18

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

CAMPUS ARAGÓN

DISEÑO, CONSTRUCCIÓN E IMPLEMENTACIÓN DE UN SISTEMA MINIMO, BASADO EN EL MICROPROCESADOR Z80 PARA LA METERIA DE MICROCOMPUTADORAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN COMPUTACIÓN

P R E S E N T A N:

ROSA HERNANDEZ VELARDE
BENITO VARGAS CALIXTO

ASESOR DE TESIS :
ING. ANTONIO NAVARRO GONZALEZ

MÉXICO 2000

18

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

CAMPUS ARAGÓN

DISEÑO, CONSTRUCCIÓN E IMPLEMENTACIÓN DE UN SISTEMA MINIMO, BASADO EN EL MICROPROCESADOR Z80 PARA LA METERIA DE MICROCOMPUTADORAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN COMPUTACIÓN

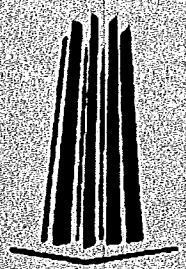
P R E S E N T A N:

ROSA HERNANDEZ VELARDE

BENITO VARGAS CALIXTO

ASESOR DE TESIS :

ING. ANTONIO NAVARRO GONZALEZ



MÉXICO 2000

AGRADECIMIENTOS

A DIOS:

El reconocimiento y agradecimiento por darnos la voluntad e inteligencia para concluir esta etapa de nuestras vidas; olvidando así nuestra imperfecta actitud filial Ante todo por su gran amor.

A nuestros padres, familiares y amigos:

Gracias por dejarnos ser lo que somos, por habernos brindado su apoyo, comprensión y cariño Sobre todo por habernos dado una palabra de aliento cuando más lo necesitábamos.

En especial al Ing. Antonia Navarro González:

Nuestro más sincero agradecimiento por dedicarnos su tiempo con empeño para encaminarnos a concluir esta tesis.

A Rosa Hernández Velarde:

Gracias por brindarme tu amistad, por haber creído en mí, por ser mi fuente de inspiración y motivo de superación.

A Benito Vargas Calixto:

Gracias por ser un buen amigo, por creer en mí, por tu gran paciencia y por haber motivado nuestra superación.

A nuestros Profesores:

Quienes lograron con sus enseñanzas una formación, para enfrentarnos hoy a los retos de nuestra profesión.

ÍNDICE GENERAL

INTRODUCCIÓN

"...sugerimos que la implementación, mantenimiento y modificación se minimizarían si el sistema pudiera diseñarse de tal manera que sus elementos fueran pequeños, relacionados fácilmente a la aplicación y con relativa independencia unos de otros. Esto significa, por lo tanto, que el buen diseño es un ejercicio de particionamiento y organización de los elementos del sistema.

Por particionamiento queremos decir la división del problema en subprogramas más pequeños, de manera que cada subprograma corresponde posteriormente a un elemento del sistema. Las preguntas son: ¿Dónde y cómo debemos dividir el problema?, ¿Qué aspectos del problema pertenecen a la misma parte del sistema y qué aspectos pertenecen a partes diferentes?. El diseño estructurado responde a estas preguntas con dos principios básicos

1) Las partes altamente interrelacionadas del problema deben estar en el mismo elemento del sistema, es decir, cosas que se necesiten entre ellos deben ir juntas.
2) Las partes no relacionadas del problema deben residir en elementos no relacionados del sistema. Es decir, cosas que no tienen nada que ver una con otra no deben ir juntas "

Yourdon, E. y L. L. Constantine[1]

Nadie puede negar que a partir del nacimiento del primer microprocesador (INTEL 4004 en 1971) comenzó una era de grandes revoluciones tecnológicas, industriales y de servicios. Además de la modificación de antiguas formas de laborar, ya que se requirió de personal más capacitado, dejando a un lado las actividades puramente manuales y obligando a los directivos a capacitar a su personal. De esta forma, directa e indirectamente los procesos de manufactura sufrieron un cambio radical, ya que el mejoramiento de los procesos de fabricación de productos se volvieron altamente especializados, dando por ende una reducción en los costos de producción, repercutiendo en un decremento del precio en el producto final. A partir de este momento, mayor cantidad de personas tuvieron acceso a dichos productos.

Gracias al rápido incremento del potencial de los microprocesadores, el ser humano tuvo la capacidad y la oportunidad de conocer y explorar nuevos terrenos que anteriormente le eran desconocidos. Algunos ejemplos muy claros se pueden ver en los sistemas de navegación automáticos, sistemas de exploración submarina, sistemas de control de procesos y desgraciadamente, sobre todo en la balística (simulación de armamentos teledirigidos, simulación de naves aéreas de ataque, etc.). Amén del control automático de satélites espaciales y robots de reconocimiento, además de los sistemas de exploración médica, tales como la tomografía computarizada y demás relativos. Sin embargo, también se ha visto una gran aplicación en áreas tales como los videojuegos, realidad virtual, efectos luminosos en los grandes eventos, etc.

Esto le ha permitido al ser humano, tomar conciencia de su entorno y del medio en que habita, permitiéndole penetrar a mundos que hasta entonces le eran desconocidos.

---

[1] Yourdon, E., et al. Diseño Estructurado. Fundamentos de una disciplina de programas de computadora y diseño de sistemas. Englewood Cliffs, New Jersey, Prentice-Hall, 1979

En suma, el uso de los microprocesadores ha abarcado áreas tan extensas que podría decirse que en la actualidad, todo está controlado directa o indirectamente por estos dispositivos.

Como se puede ver de lo anteriormente expuesto, es menester del ser humano (y sobre todo de las nuevas generaciones de profesionistas, especialmente de los ingenieros) tomar nota y aprender a utilizar estos poderosos dispositivos.

De esta forma, esta tesis surge de la necesidad de vincular los conocimientos teóricos adquiridos en varias materias de la carrera de Ingeniería en Computación (Diseño Lógico, Diseño de Sistemas Digitales y Microprocesadores por mencionar sólo algunos) en el diseño de los pilares de una microcomputadora: un Sistema Mínimo.

De igual forma, surge de la necesidad de sentar las bases e incitar a las futuras generaciones de ingenieros a continuar el estudio y diseñó de sistemas digitales basados en el uso de los microprocesadores para acceder a un mercado laboral altamente competitivo, con los fundamentos teóricos y prácticos del uso de esta tecnología en constante expansión.

Para tener una cabal y clara conciencia de lo que implica el manejo de microprocesadores, es necesario realizar un estudio detallado de los elementos básicos de que constan estos dispositivos: su forma de arquitectura interna, el juego de instrucciones de que consta, la forma de realizar su interconexión con el "mundo exterior", los dispositivos en los que se alojará la información que lo ha de instruir en las actividades que va a realizar y la forma de realizar dicha actividad.

Todo proyecto de construcción de circuitos electrónicos requiere cierto nivel de conocimiento de los materiales que se vayan a utilizar, ello simple y sencillamente por el hecho de que se requiere saber el funcionamiento de cada elemento en forma individual y su forma de interconexión con otros elementos, de tal forma que se obtenga un dispositivo terminal y funcional (sobre todo que cumpla con los requerimientos demandados por el usuario o público consumidor). No es suficiente el hecho de poderse guiar por obras ya concluidas, ya que en la mayoría de los casos, estos proyectos terminales tuvieron un objetivo concreto o personal, y que por ende, no cumple con los requerimientos actuales, y que en el peor de los casos, los materiales con los que fueron construidos, han caído en desuso o no están al alcance del público constructor.

Así, el diseño de esta tesis se basa en los siguientes objetivos:

1) Elaborar un Sistema Mínimo con dispositivos periféricos de entrada/salida y sus interfaces de comunicación entre éstos y el microprocesador.
2) Diseñar y programar un pequeño sistema operativo que permita controlar las funciones de los distintos elementos de este Sistema Mínimo.
3) Diseñar un programa que permita a los usuarios introducir al Sistema Mínimo sus propios programas y corroborar su funcionamiento.
4) Mostrar a los estudiantes y público en general, la forma de diseño y conexión de un Sistema Mínimo basado en un microprocesador, así como la programación de su sistema operativo.

5) Reducir el costo de diseño de un sistema mínimo mediante la utilización de circuitos integrados convencionales y al alcance del público en general

Para un cumplimiento cabal de los objetivos de la tesis, ésta se divide en los siguientes capítulos:

En el capítulo I, se explica brevemente los conceptos generales de microprocesadores, microprogramas, microinstrucciones, buffers y en forma global, de todos los elementos funcionales que forman parte de las microcomputadoras.

En el capítulo II, se habla en particular de la arquitectura y elementos de que consta el microprocesador Z80 (ALU, buffers, registros, nombre y función de las señales de entrada/salida, etc.), finalizando en el mismo capítulo con las características eléctricas del dispositivo.

El capítulo III se enfoca a la forma de la interconexión de memorias y puertos de entrada/salida al microprocesador Z80 (conexión con el "mundo externo"), así como de las distintas formas de llevarlas a cabo.

En el capítulo IV, se explica en forma detallada los dispositivos periféricos que se van a conectar al sistema mínimo, los cuales permiten la comunicación entre el usuario y el microprocesador. Así como su funcionamiento y la forma en que se diseña la interfaz entre el mismo dispositivo periférico y el Sistema Mínimo.

En el capítulo V, se procede a la descripción del diseño del Sistema Mínimo que se requiere para hacer funcional al microprocesador Z-80; es decir, un microprocesador por sí mismo es incapaz de realizar una actividad mínima, tal como controlar el sentido de giro y la velocidad de un motor. Para realizar tal actividad, el dispositivo de control (microprocesador) requiere de elementos externos (circuitos de potencia) que le permitan controlar al motor.

Finalmente, en el capítulo VI se procede a la descripción del sistema operativo, la cual es la encargada de "hablar" el mismo lenguaje que los dispositivos periféricos, y de controlar los eventos que se generan a partir de los datos obtenidos por el usuario o por el medio ambiente.

Debido a que el proceso de diseño de un Sistema Mínimo basado en microprocesador no es una actividad que involucre una gran cantidad de información (investigación documental), sino que por el contrario, exige una gran cantidad de inversión de tiempo para probar los distintos elementos de que consta el Sistema Mínimo, así como de los elementos que se requieren para ir corroborando el funcionamiento del sistema (fuentes de potencia, analizadores lógicos, osciloscopios, etc), esta tesis se apoya más en la investigación de campo para elaborar un producto final y que realice el trabajo esperado.

Esperamos que esta tesis beneficie al público en general y sobre todo, en especial a los estudiantes de ingeniería, ya que les muestra en forma metódica el proceso de diseño de un Sistema Mínimo, así como los circuitos integrados existentes en el mercado, los cuales permiten disminuir el costo del sistema al mínimo.

CAPITULO I:
CONCEPTOS GENERALES

"...estaba sentado en un cuarto de la Sociedad Analítica, en Cambridge, mi cabeza inclinada en la mesa en una especie de gracioso sueño, con una tabla de logaritmos descansando atrás de mí. Otro miembro se introdujo al cuarto, y viéndome medio dormido gritó, bien Babage, ¿qué sueñas ahora? A lo que contesté estaba pensando que esas tablas (señalando a los logaritmos) pueden ser calculadas por máquinas"

Charles Babage

1.1 Procesador

No existe ningún otro concepto más difícil de definir como el término **Procesador**. A grandes rasgos, los procesadores se caracterizan por ser sistemas que están formados por varias etapas a saber: la primera etapa está formada por un medio de entrada de datos, la segunda etapa por un medio o dispositivo que le indica cómo procesar los datos y finalmente la última etapa consiste en un medio de salida de los datos procesados. En esta última etapa, la salida del procesador se puede ver como la entrada a otro procesador, cuya finalidad puede ser diferente a la actual, pero que requiere de los servicios de la primera.

Existe una gama infinita de procesadores (incluso el mismo ser humano). Un ejemplo puede ser un extractor manual de cítricos, la entrada al procesador es los cítricos en si mismo, el medio que le indica cómo debe procesar estos cítricos está dada por la persona que presiona los dos mangos, ejerciendo una fuerza sobre los cítricos a través de una palanca que se encuentra al final del procesador, mediante esta acción, el procesador arrojará como salida el jugo de los cítricos, el cual, dependiendo de la finalidad de la persona puede tener muchos usos.

La realización de esta tarea mecánica y monótona (extracción de jugo de un cítrico) a la larga termina por fastidiar a un ser humano, de igual forma paulatinamente va menoscabando su ego, su amor propio y su deseo innato de adquirir cultura mental.

Sin embargo, gracias a su intelecto (y sobre todo a su deseo de comodidad y ley del menor esfuerzo), el hombre ha inventado dispositivos que suprimen o aligeran su carga. Remitiéndonos al ejemplo de los cítricos, en la actualidad existen máquinas que lo sustituyen en esta labor. A pesar del invento de estos dispositivos, aún debe estar presente el hombre para vigilar el momento en que la máquina esté lista para comenzar a trabajar, además tiene que proporcionarle las entradas al procesador (de cítricos) y retirar el producto final.

Para evitar tales inconvenientes, la humanidad inventó y desarrolló los procesadores (refiriéndonos al término más genérico) automáticos y programables. Es decir, dispositivos que cuentan en su interior con la información que le indican cómo debe proceder de acuerdo a los

cambios físicos de su entorno (estos cambios físicos son captados a través de sensores y digitalizados mediante **convertidores analógico-digitales** o **ADCs**)

Los primeros procesadores programables que vieron la luz, eran demasiado grandes y costosos, consumían mucha potencia y eran demasiado lentos. Dadas las características de los primeros procesadores programables, éstos llegaban a ocupar cuartos enteros y requerían de grandes sistemas de ventilación para evitar su sobrecalentamiento. Además, eran creados con el fin de realizar funciones específicas (cálculos científicos y balísticos). Sin embargo, gracias a la miniaturización de los componentes, estos gigantescos monstruos fueron reduciendo su tamaño y por ende su costo (llegando a ser accesible para muchas personas), incrementando adicionalmente su potencia de cálculo.

Resumiendo, podemos decir que un procesador se puede definir como un sistema que explora secuencialmente una información (programa) almacenada en un dispositivo especial (memoria), interpreta sus instrucciones y las ejecuta.

Cuando este gigantesco procesador se miniaturizó y se pudo grabar en una simple oblea de silicio, nació la era de los microprocesadores. Genéricamente, este dispositivo "inteligente" es conocido con las siglas de **CPU (Central Process Unit)** o **Unidad Central de Procesos**.

1.2 Microprocesador

A partir del momento en que se pudo grabar en una diminuta oblea de silicio los componentes mínimos necesarios (transistores[1]) para formar circuitos integrados que realizaban una acción específica, dieron pie para que naciera el concepto de microprocesador. Estos dispositivos se encuentran dentro de la gama de los circuitos integrados de **Gran Escala de Integración (LSI)**[2], tecnologías más avanzadas se encuentran dentro del concepto de **Muy Grande Escala de Integración (VLSI)** o **Ultra Gran Escala de Integración (ULSI)**.

El microprocesador cuenta con varias características, entre las cuales se pueden mencionar:

a) Programabilidad: Es la capacidad de un microprocesador de actuar de acuerdo a unas instrucciones almacenadas en un medio o dispositivo (normalmente en una memoria **ROM**)

[1] La base de la tecnología de los circuitos integrados (y por ende de los microprocesadores) se ubica en los transistores, ya que estos se graban en una placa de silicio basado en impurezas y crecimientos epitaxiales de determinadas zonas de la placa de silicio.

[2] El término escala de integración se refiere al número de compuertas mínimas formadas básicamente a partir de transistores que se pueden grabar en una simple oblea de silicio. Este número de componentes varía dependiendo de la técnica utilizada. Las escalas más conocidas son:

Mnemónico	Descripción	Número de compuertas por circuito integrado
SSI	(Small Scale Integration) Pequeña Escala de Integración.	1 - 10
MSI	(Medium Scale Integration) Mediana Escala de Integración.	10 - 100
LSI	(Large Scale Integration) Gran Escala de Integración.	100 - 2,000
VLSI	(Very Large Scale Integration) Muy Grande Escala de Integración.	2,000 - 10,000
ULSI	(Ultra Large Scale Integration) Ultra Gran Escala de Integración.	10,000 en adelante

b) Universalidad: Es la capacidad que tiene el microprocesador de fungir como elemento "inteligente" o "cerebro" en aplicaciones que van desde una computadora personal hasta un rastreador de satélites. Esta caracteristica se debe a los numerosos componentes que se encuentran grabados dentro del microprocesador tales como flip-flops, contadores, registros, comparadores, decodificadores, etc. Estos componentes están formados a través de transistores.

Dependiendo del número de bits de datos que maneje, un microprocesador puede ser clasificado en:

1) **Microprocesadores de 4 bits**: El más conocido es el antiquisimo 4004 de Intel. Este microprocesador nació a finales del año 1971 (Noviembre). Entre las caracteristicas más sobresalientes de este microprocesador se encuentran:
   a) **Bus de Datos** de 4 bits y **Bus de Instrucciones** de 8 bits.
   b) 1 Kb de **Memoria de Datos** y 4 Kb de **Memoria de Programa**.
   c) Registro **Contador de Programa** (**Program Counter** o **PC**) de 12 bits
   d) 16 **Registros de Propósito General** de 4 bits u 8 **Registros de Propósito General** de 8 bits.
   e) 16 Instrucciones ejecutándose a una frecuencia de reloj de 740 KHz.
   f) 2300 transistores grabados en su interior, contenidos en un chip de 16 pines.
   g) Su principal aplicación se ubicó en las calculadoras.

   Para el año de 1972, Intel introdujo el 4040 como una versión mejorada del 4004. En ésta, el conjunto de instrucciones se incrementa en 14, la **Memoria de Programa** se expande a 8 Kb y se introduce la capacidad de manejar interrupciones.
2) **Microprocesadores de 8 bits**: Pilares de esta familia son los mundialmente conocidos 8008 (el cual era una versión mejorada del 4040) y el 8080. Estos microprocesadores se caracterizan por un tener un **Bus de Datos** de 8 bits, además de un conjunto de **Registros** de 8 bits (**A**, **B**, **C**, **D**, **E**, **H** y **L**). Estos **Registros Individuales** se pueden agrupar por pares para formar **Registros** de 16 bits (**BC**, **DE** y **HL**). Se adicionan al microprocesador circuiteria para direccionar 256 **Puertos de Entrada/Salida**. Cuentan con un **Bus de Dirección** de 16 bits, un registro **SP** (**Stack Pointer** o **Puntero de Pila**) de 16 bits (el cual sustituye a las antiguas **Stack Multinivel** de los microprocesadores de 4 bits). El microprocesador **8080** fue usado en el **Altair 8800**, la primera "Computadora Personal". Posteriores mejoras a estos microprocesadores se encuentran el **8085** (1976), el **Z80** (julio de 1976, el cual fue utilizado por primera vez en el **TRS-80 Model 1**). Aún cuando otras compañías crearon diferentes versiones de los microprocesadores anteriores, éstas fueron las más representativas.
3) **Microprocesadores de 16 bits**: Tal vez el microprocesador **TMS 9900** de Texas Instruments sea el primer microprocesador real de 16 bits que apareció (si no se toma en cuenta la propiedad **Bit-Slice** de los microprocesadores **IMP-16** de National Semiconductor o el

microprocesador **2901** de Advanced Micro Device) Su desarrollo se basó en el minicomputador **TI 990**. Este microprocesador alojaba circuiteria para un **Bus de Direcciones** de 15 bits, un par de **Registros Internos** de 16 bits, además de un **Bus de Datos** de 16 bits (características representativas de esta familia de circuitos integrados). Además del microprocesador anterior, también se dieron a conocer las siguientes compañías con sus respectivos microprocesadores: Intel (**8086**, **80286**), Zilog (**Z-8000**), entre otras

4) **Microprocesadores de 32 bits**: El primer microprocesador que tuvo una circuiteria interna de **Bus de Datos** de 32 bits fue el **68000** de Motorola (Septiembre, 1979), sin embargo su **Bus de Datos** externo era de sólo 16 bits. Otros ejemplos de microprocesadores de 32 bits fueron el **32032** de National Semiconductor, el **80386SX** de Intel, entre otros. Los microprocesadores que realmente tenían un **Bus de Datos** de 32 bits fueron el **80386DX**, el **80486** (ambas de Intel), el **Z80000** de Zilog, el **68020** de Motorola, entre otras.
5) **Microprocesadores de 64 bits**: Probablemente el microprocesador de 64 bits que sea más conocido es el Pentium de Intel (con este nombre, Intel cambia toda una gama de generaciones de microprocesadores 80xxx). Éstas se caracterizan por tener un **Bus de Datos** de 64 bits, una arquitectura superescalar, predicción de rama dinámica, enrutamiento por tuberías, unidad de punto flotante, ejecución mejorada de instrucciones, cachés de código y de datos separada, caché de datos de retroescritura, enrutamiento de ciclo de bus, paridad de dirección y verificación de paridad interna, entre otras características tendientes a mejorar el rendimiento de los sistemas.

   Otros microprocesadores que se ubican dentro de esta familia son: **U-SPARC**, **R10000**, **Alpha**, etc.

Gracias al incremento del número de bits de datos que un microprocesador puede manejar, se acelera su velocidad de procesamiento, puesto que requiere de menos ciclos de reloj para ejecutar una instrucción

Además de los microprocesadores de longitud de datos estándar, existen los microprocesadores por trozos (**Bit-Slice**) también denominados **En Cascada**. Estos microprocesadores se caracterizan por tener su parte común (**Contador de Programa**, **Registro de Dirección de Memoria**, **Registro de Dirección de Operando**, **Registro de Instrucción**, **Decodificador de Operación y Memoria de Control**) en un chip separado y la parte de cálculo (**Acumulador**, **Unidad Aritmético/Lógica** y **Registro de Datos de Memoria**) se encuentran localizados dentro de chips que están relacionados directamente con la longitud de los datos. Éstos permiten procesar datos de longitud deseada y no normalizada (mediante acoplamiento); es decir, si un chip tiene la capacidad de procesar 4 bits de datos y se desea procesar 20 bits de datos, se interconectan 5 chips controlados por la parte común. Éstos se interconectarán en paralelo, teniendo en cuenta cada uno los procesos del otro. Siempre debe de tomarse con ellos

las mayores precauciones al momento de la programación y sobre todo con los bits de acarreo de los chips individuales.

Debido a las características anteriores, la creación de aplicaciones con microprocesadores **Bit-Slice** es muy compleja ya que el diseñador debe de configurar la **CPU** y el hardware del sistema, además de la microprogramación del conjunto de microinstrucciones. Otra desventaja es que no existe mucha ayuda de desarrollo de software disponible

Dentro de sus ventajas, se puede mencionar entre otras

a) La mayoría son bipolares (siendo más rápidos que la tecnología contraparte **MOS**).
b) Se pueden eliminar problemas de disipación de potencia al repartir la lógica bipolar sobre varios chips.
c) Se pueden construir sistemas de cualquier longitud de palabra y crear un conjunto de instrucciones óptimo para una aplicación en particular.

1.2.1 Unidades Funcionales Básicas de un Microprocesador

El microprocesador está conformado por las siguientes unidades funcionales básicas **Unidad de Control, Unidad Aritmético/Lógica y Registros de trabajo**, tal como se puede apreciar en la siguiente figura:

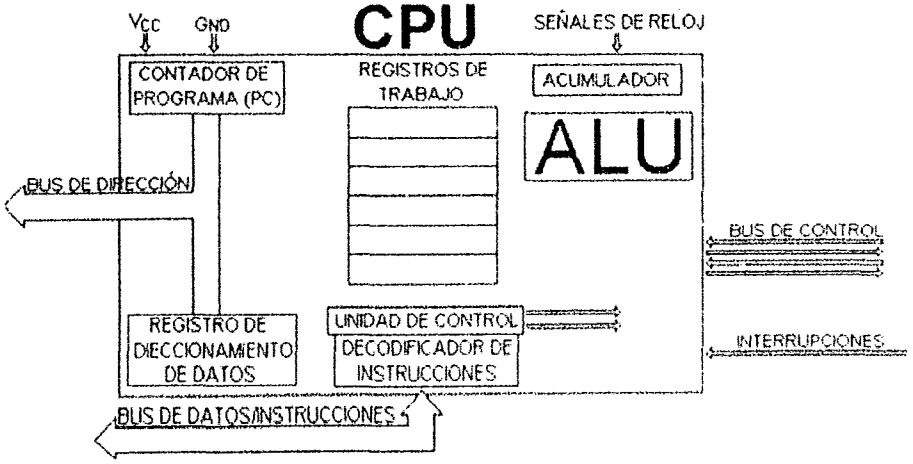


Fig. 1.1 ESTRUCTURA FUNDAMENTAL DE UN MICROPROCESADOR

Para que el microprocesador pueda operar en forma óptima, requiere de componentes adicionales; tales como una fuente de alimentación, una serie o tren de pulsos de reloj (siendo este pulso de reloj unifase o bifase, dependiendo de las características eléctricas del microprocesador) y unos canales o pistas por donde se mueve la información, estas pistas a su vez interconectan a cada una de las partes. Estas pistas son conocidas con el nombre de **Bus** (genéricamente, estos buses se consideran externos al microprocesador, ya que cumplen con la función de interconectar al propio microprocesador con dispositivos externos o periféricos, sin

embargo, también existen los buses internos, los cuales interconectan a cada una de las partes de los microprocesadores).

Dependiendo del tipo de información que manejen, estos buses pueden ser clasificados en

a) **Bus de Direcciones**: Son unidireccionales y generalmente triestado[3], tienen la función de direccionar localidades de **Memoria** o **Puertos de Entrada/Salida**. Dependiendo del número de bits de este bus será la capacidad de memoria que podrá direccionar, por ejemplo, si el microprocesador cuenta con un **Bus de Direcciones** de 16 líneas (numeradas de A_0 hasta A_{15}), puede direccionar hasta 65536 (64 Kb) localidades diferentes de **Memoria**

b) **Bus de Control**: Son unidireccionales, siendo de entrada o de salida y generalmente triestado tienen la función de contener las señales que han de controlar las operaciones a realizar con los **Puertos de Entrada/Salida**, ejemplos de tales señales de control pueden ser $\overline{WR}$[4] (indica una petición de escritura en la **Memoria** o en un **Puerto de Entrada/Salida**), $\overline{RD}$ (indica la petición de lectura de una localidad de **Memoria** o de un **Puerto de Entrada/Salida**), etc.

c) **Bus de Datos**: Son bidireccionales y generalmente triestado, tienen la función de contener los datos e instrucciones que indicarán cómo ha de operar el sistema. De igual forma, permiten la comunicación bidireccional con los componentes del sistema, tales como la **Memoria Principal** o los **Dispositivos Periféricos** conectados mediante los **Puertos de Entrada/Salida** al microprocesador. El número de líneas en el **Bus de Datos** determina el tamaño de la palabra que el microprocesador puede manejar. Dependiendo del número de bits que se maneje en este bus, será su forma de numeración. Por ejemplo, si cuenta con 8 bits de datos, estos serán numerados desde D_0 hasta D_7.

En resumen y de acuerdo a sus principales componentes, un microprocesador es conocido como un ***sistema digital de proceso síncrono*** o ***procesador síncrono***[5].

---

[3] El término Tri-Estado, se refiere al estado eléctrico de los buses, es decir, a la presencia o ausencia de datos en el mismo. Para tal efecto, en los buses puede existir un estado "0", el cual indica una línea desactivada, un estado "1" indica una línea activada, y finalmente, un estado de alta impedancia. Este último estado le indica a cualquier dispositivo que así lo requiera, que el bus de direcciones se encuentra libre para que pueda utilizarlo sin temor a tener conflictos con el microprocesador. En resumen, este último estado indica que el microprocesador no esta utilizando el bus indicado.

[4] El guion sobre la señal de control indica que su estado está normalmente en alto (esta siempre presente una señal "1", es decir, por dicha línea se encuentra presente una diferencia de potencial de aproximadamente 5 volts) y que se activa cuando se va a un estado bajo (se presenta una señal de "0" o una ausencia de voltaje).

[5] Un procesador síncrono, es un sistema que cambia de estado en respuesta a los pulsos sincronizados de reloj y al conjunto de instrucciones que se encuentran almacenados en una memoria físicamente conectada al sistema. Los componentes fundamentales de un procesador digital o síncrono son:

a) Unidad de memoria. Es un sistema síncrono secuencial formado por dispositivos biestables (flip-flops) que almacenan los datos.
b) Unidad Aritmético/Lógica: Recibe información procedente de la memoria, realiza con ella cálculos adecuados y los devuelve al sistema.
c) Unidad de Control: Está constituida por un sistema secuencial síncrono. Evoluciona entre estados de una forma síncrona controlada por los impulsos del generador, siguiendo un determinado diagrama de flujo. En cada uno de los estados, genera las señales adecuadas para que la unidad de control y la memoria realicen la operación adecuada.

1.2.1.1 Unidad Aritmética/Lógica

La Unidad Aritmética/Lógica es la parte medular del microprocesador, en esta sección se realizan las operaciones trascendentales del chip, tales como las operaciones de suma, resta, complemento, operaciones lógicas tales como ANDs, ORs, NORs, etc. En algunos sistemas más complejos también se realiza la operación de multiplicación. Las operaciones que se realizan dentro de la **ALU** están determinadas por las instrucciones presentes en la **Memoria de Programa**.

El diagrama esquemático de una **ALU** se puede visualizar en la siguiente figura

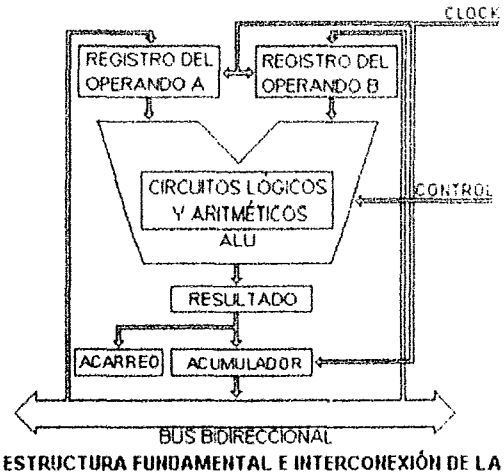


Fig. 1.2 ESTRUCTURA FUNDAMENTAL E INTERCONEXIÓN DE LA ALU

Como se puede apreciar en la figura anterior, la **ALU** se compone principalmente de unos **Circuitos Lógicos y Aritméticos** y unos **Registros**, donde se han de almacenar tanto los operandos como el resultado de las operaciones que se hayan realizado sobre ellas (normalmente, la **ALU** siempre lleva asociado un registro que funge como un medio de almacenamiento temporal de los resultados de las operaciones realizadas por el microprocesador; a este **Registro Especial** se le denomina **Acumulador**). Todos los componentes de la **ALU** están internamente conectadas por un **Bus Interno**. Los **Registros** tienen la función de tomar desde y enviar los datos hacia el **Bus Bidireccional de Datos**. Todos estos procesos están sincronizados por la **Unidad de Control**.

Para propósitos didácticos, en la actualidad existe una amplia gama de circuitos integrados que internamente están construidos como **ALUs**, ejemplos de tales dispositivos es el 74LS181, entre otros.

1.2.1.2 Unidad de Control

La **Unidad de Control** (conocida generalmente con el término de "**corazón del sistema**") genera la secuencia de impulsos adecuada para que el microprocesador procese la información. Esta unidad recibe secuencialmente las instrucciones por el **Bus de Instrucciones**, almacenándose en un registro especial denominado **IR (Instruction Register)** o **Registro de Instrucciones**. El registro **IR** deposita las instrucciones en el **Decodificador de Instrucciones**, el cual se encarga de interpretarlas y producir una serie de impulsos de gobierno en el **Generador de Impulsos** o **Secuenciador** (también es conocida como **Generador de Ciclos de Máquina** o **GCM**). Al mismo tiempo, la **Unidad de Control** incrementa sincrónicamente el registro **Contador de Programa (PC)** cada vez que se ejecuta una instrucción.

En la siguiente figura se pueden apreciar los componentes básicos de una Unidad de Control, así como su distribución:

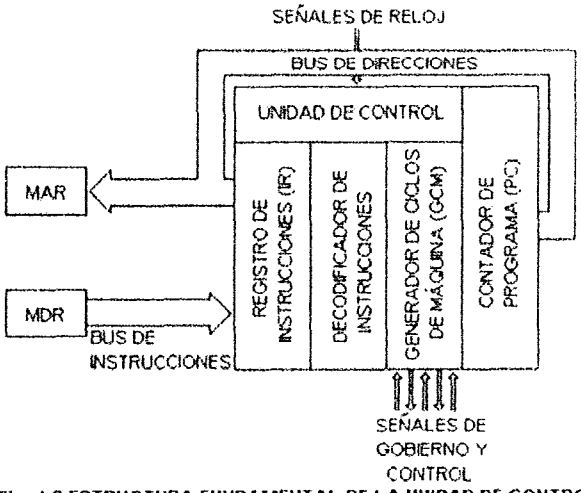


Fig. 1.3 ESTRUCTURA FUNDAMENTAL DE LA UNIDAD DE CONTROL

Cuando se realiza una operación que involucre una dirección externa al microprocesador, se obtiene desde la **Unidad de Control** una dirección de la memoria de datos.

1.2.1.3 Registros

Los registros son utilizados por el microprocesador para almacenar información de datos, direcciones y de los estados internos (**Flag**) del mismo microprocesador. Los registros son dispositivos lógicos secuenciales que están compuestos en su forma más simple por flip-flops; es decir: los registros almacenan cada bit de información en un flip-flop. De esta forma, dependiendo del número de bits de información que pueda manejar el registro, será el número de flip-flops que tenga agrupados en su interior.

Algunos registros pueden ser referenciados implícitamente como campos de operandos o como parte de una especificación de dirección. Otros por el contrario, sólo pueden ser referenciados de modo implícito (tales como el registro de Estado o **Flag**). Algunos permanecen ocultos totalmente y no pueden ser referenciados implícita o explícitamente.

Dentro del microprocesador existen dos grandes grupos de registros, a saber:

1) **Registros de Propósitos Específicos** o **Dedicados**. Estos registros sólo son accesibles y modificables directamente por el microprocesador, de esta forma estos registros permiten al microprocesador realizar sus funciones básicas, tales como el registro de la siguiente instrucción a ejecutarse, los datos/instrucciones que se han de accesar, el código de operación de la instrucción que se va a ejecutar o el estado interno del microprocesador después de ejecutarse una instrucción. Dentro de la categoría de los **Registros Dedicados** se ubican los siguientes:
   a) **Contador de Programa (PC)**. Este registro almacena la dirección absoluta de la siguiente instrucción que se ha de ejecutar. Normalmente, las instrucciones se encuentran en localidades sucesivas de memoria, dado lo cual, al momento de efectuarse una instrucción el registro **PC** se incrementa en una unidad (siempre y cuando las instrucciones requieran de un sólo ciclo de máquina, en caso de ser instrucciones de más ciclos, en esa misma medida el registro **PC** será incrementado). Al momento de ejecutar instrucciones de bifurcación, el registro **PC** guarda la dirección de la posición de memoria a bifurcar y cuando se efectúe una instrucción de **RET**orno, el mismo registro **PC** debe contener la dirección de memoria siguiente como si no se hubiera efectuado ningún salto.
   b) **Registro de Dirección de Memoria (MAR)**: Este registro contiene la dirección de memoria absoluta de los datos o instrucciones que se han de accesar. El registro **MAR** se conecta al **Bus de Direcciones** del sistema y su tamaño determina la capacidad de memoria que se puede direccionar.
   c) **Registro de Datos de Memoria (MDR)**: También denominado como **Registro Buffer de Memoria** o **Memory Buffer Register(MBR)** por sus siglas en inglés. Este registro almacena los datos de entrada o salida desde y hacia la memoria central, o a los dispositivos periféricos conectados al sistema. Se conecta al **Bus de Datos** del sistema y su tamaño determina la unidad máxima de transferencia de datos. A mayor capacidad del registro **MDR**, los ciclos de máquina necesarios para la ejecución de una instrucción son menores.
   d) **Registro de Instrucción (IR)**: El código de operación de una instrucción se almacena en este registro. Posteriormente, es decodificado y da lugar a la generación de señales de control del microprocesador.

e) **Acumulador**: Este registro funge como área de trabajo del microprocesador (más en particular de la **ALU**). Este registro siempre contiene el resultado de la última operación efectuada. La capacidad de este registro va acorde con la capacidad del registro **MDR**

f) **Registro de Estado (Flag)**: Después de una operación, este registro guarda el estado interno del microprocesador. Para tal efecto, el registro de estado comúnmente cuenta con los siguientes campos:

1) **Estado nulo del resultado (Z)**: Se activa al obtenerse como resultado un entero cero en la última instrucción ejecutada

2) **Signo del Resultado (S)**: Se activa al obtenerse como resultado un número negativo en la última operación efectuada. En caso contrario, si el resultado es positivo, este bit se desactiva.

3) **Paridad del resultado (E)**: Este bit se activa cuando el resultado de la última operación efectuada es un número par, si el resultado es impar, el bit se desactiva.

4) **Acarreo (C)**: Este bit se activa cuando se genera un acarreo en la última operación efectuada. Normalmente se toma en cuenta en las operaciones aritméticas y de corrimiento.

5) **Desbordamiento (O)**: Este bit se activa cuando el resultado de la última operación excede la capacidad del acumulador.

g) **Puntero de Pila (SP)**: Este registro contiene la dirección del último valor inscrito en la pila de direcciones. Es decir, apunta a la parte superior de la pila. Tiene la función de salvaguardar los **Registros de Propósito General** al momento de efectuarse una llamada a una subrutina, así como mantener la dirección actual del registro **PC**.

2) **Registros de Propósito General**: Son registros básicos de almacenamiento (también se les denomina **Memoria de Trabajo**) accesibles y modificables directamente por el usuario, tienen como función almacenar los datos de los usuarios. Para tal efecto, en el microprocesador se encuentran interconstruidos registros para la transferencia de datos o para efectuar un desplazamiento de datos, para la conversión serial (bit a bit) a la forma paralela (octeto a octeto), etc.

Gracias a los **Registros de Propósito General**, se elimina la necesidad de mover los datos entre la memoria y el acumulador, aumentando en forma considerable la velocidad de ejecución de las instrucciones.

Los registros de Propósito General serán tratados con mayor profundidad en el capitulo 2, razón por la cual no entraremos en detalles sobre este tipo de registros.

A grandes rasgos, la ejecución de una instrucción por parte del microprocesador se divide en dos fases.

1) **Fase de búsqueda (Fetch) de la instrucción**. En esta fase, se generan las señales necesarias para buscar una instrucción en la memoria y cargarla a los registros del microprocesador. Esta fase se subdivide en varias etapas, a saber:

   a) El contenido del registro **Contador de Programa o PC** (dirección de la posición de memoria en donde se encuentra el código de la instrucción) se transfiere por medio del bus de direcciones al registro **MAR (Registro de Direcciones de Memoria)**. Dicho proceso se puede ver gráficamente en la siguiente figura

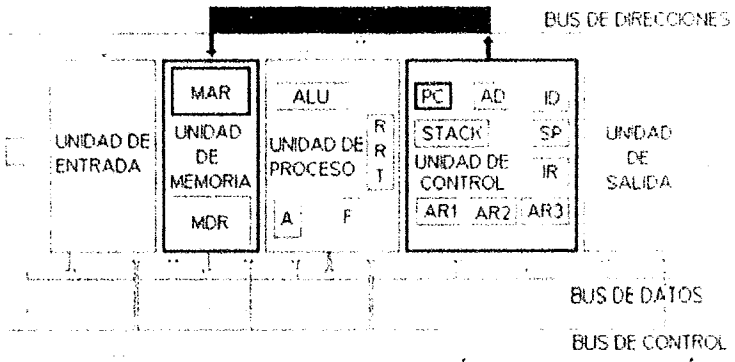


Fig. 1.4 PRIMERA ETAPA DE LA FASE FETCH (BÚSQUEDA) DE INSTRUCCIÓN

   b) Decodificada y seleccionada la posición de memoria, ésta aparece en el registro de datos de memoria. Posteriormente es transferido por el bus de instrucciones al **Registro de Instrucciones(IR)** en la **Unidad de Control**. Dicho proceso se muestra en la figura siguiente:

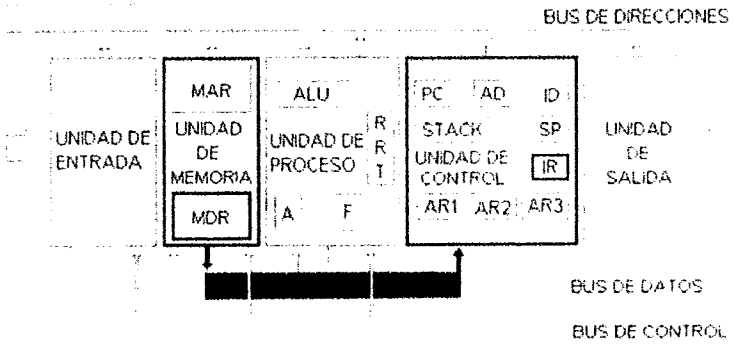


Fig. 1.5 SEGUNDA ETAPA DE LA FASE FETCH (BÚSQUEDA) DE INSTRUCCIÓN

2) **Fase de ejecución (Execute) de la instrucción** En esta fase, se ejecuta la instrucción cargada en los registros del microprocesador, y se reinicia la secuencia de generación de señales para la búsqueda y ejecución de la siguiente instrucción. Para efectos de claridad en la explicación, esta fase suele subdividirse en las siguientes etapas:

a) Decodificación. La parte de la instrucción llamada código de operación (**Opcode**) es transferido al **Decodificador de Instrucciones** (ID), en esta etapa se incrementa el contenido del registro **PC**. El proceso se visualiza en la siguiente gráfica

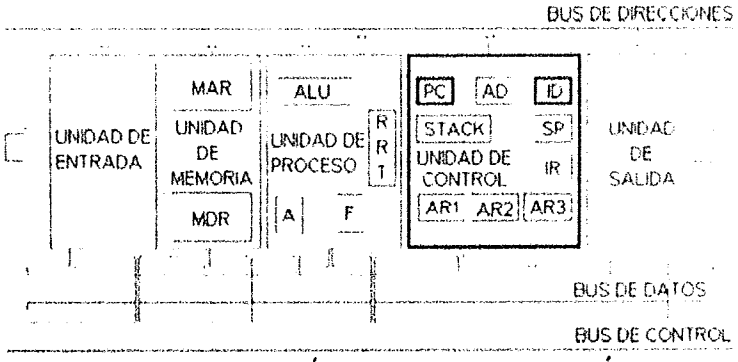


Fig. 1.6 FASE DE EJECUCIÓN (EXECUTE) DE LA INSTRUCCIÓN

b) Decodificada la instrucción, la unidad de control genera las señales o impulsos necesarios para llevar a cabo la instrucción.

1.3 Memoria

Dado que un microprocesador es un dispositivo que ejecuta una serie de instrucciones en forma secuencial, obteniéndose unos resultados parciales (resultados que pueden ser las entradas para posteriores instrucciones) debe de existir algún medio o dispositivo en el que se puedan almacenar estas instrucciones y resultados. Tales medios o dispositivos de almacenamiento (datos o instrucciones) reciben el nombre genérico de memorias.

Las memorias pueden ser clasificadas de diferentes formas. Algunos tipos de clasificación se basan en:

1) Por el tipo de material usado: Semiconductores vs Magnéticos.
2) Capacidad de retención de la información cuando la alimentación es suprimida: Volátil vs No volátil.
3) Forma de acceso a los datos: Serial vs Aleatorio.

1.3.1 Memoria de Programa

Antes de entrar en detalle acerca de los tipos de dispositivos de memoria ROM existentes en el mercado, procedamos a explicar en forma más clara el concepto de **Memoria de Programa**

Recuerde de las definiciones anteriores, que un microprocesador es un sistema síncrono secuencial y como tal, tiene como elementos principales una **Unidad de Memoria**, una **Unidad Aritmético/Lógica** y finalmente una **Unidad de Control** Estos elementos en conjunto son los que permiten que un microprocesador ejecute una serie de instrucciones en secuencia y sea el elemento "inteligente" en cualquier sistema computarizado

En este caso la memoria **ROM** cumple la función de fungir como **Memoria de Programa** En este dispositivo se almacena el conjunto de instrucciones que debe ejecutar el microprocesador, normalmente esta serie de instrucciones son fijas y no cambian a voluntad del usuario del sistema El tamaño de cada localidad de la memoria **ROM** está acorde con el tamaño de la palabra del microprocesador (éste puede tener una longitud de datos de ocho bits, formando un octeto de datos[6] o ser la unión de varios octetos para formar palabras de 16, 32 y 64 bits en los sistemas computadores más modernos).

Un punto principal a recordar es que no cualquier instrucción puede ser programado en la **Memoria de Programa**, esto es: no se puede extraer arbitrariamente de otro sistema el contenido de su memoria **ROM**, insertar en un sistema computador diferente y esperar que funcione (a menos que sea un sistema estándar, y todos los elementos tengan el mismo funcionamiento, de tal forma que sean intercambiables y reemplazables entre sí), dado que las instrucciones de los microprocesadores son particulares para cada compañía y cumplen funciones específicas dependiendo de la arquitectura del propio microprocesador.

De acuerdo a la complejidad del sistema computador, la **Memoria de Programa** puede estar compuesta por memorias **ROM**, **PROM**, **EPROM** o **EEPROM** (éstos tipos de memorias serán definidos en la siguiente subsección). Este último tipo de memoria **ROM** es útil cuando no se desea extraer la memoria del circuito para reprogramarlo, ya que puede programarse y verificar su funcionamiento en el mismo circuito.

En los modernos sistemas de cómputo, la **Memoria de Programa** no comprende al sistema operativo en sí, sino sólo una pequeña parte la cual se encarga de "reconocer" los principales dispositivos periféricos conectados a ella (tal como el teclado), las cuales son imprescindibles para su funcionamiento (de hecho, en los modernos sistemas de cómputo, si algún elemento esencial falla, tal como el teclado o alguno de los discos duros conectados el sistema no bootea), después de haber reconocido los dispositivos periféricos esenciales, las instrucciones programadas en la memoria ROM buscan un drive de la cual va a cargar el sistema operativo en memoria **RAM**. Esta

[6] Un octeto de datos u ocho bits forman un byte de datos

pequeña **Memoria de Programa** que se encarga de las funciones básicas del sistema se denomina **BIOS (Basic Input/Output System)** o **Sistema Básico de Entrada/Salida**

De esta forma, la **Memoria de Programa** puede tener muchas y variadas funciones acorde a las necesidades y requerimientos del sistema computador. Sin embargo, algo que caracteriza a las **Memorias de Programa** es que su información no puede modificarse arbitrariamente, ni mucho menos almacenar información que constantemente se esté modificando.

1.3.2 Memoria ROM

Las siglas de las memorias **ROM** se derivan de las palabras anglosajonas **Read Only Memory (Memoria de Sólo Lectura)**; es decir, son dispositivos que retienen su información aún en el caso extremo de que se les suprima la fuente de energía. De igual forma, es imposible escribir datos en estos dispositivos por medios normales, requiriéndose de métodos especiales y de dispositivos auxiliares (grabadores de memoria) para grabar información en sus celdas.

Estos dispositivos de almacenamiento son usados cuando no se desea que la información grabada en sus celdas se pierda al suprimir la fuente de alimentación.

Las memorias **ROM** se suelen subdividir en:

a) **ROM (Read Only Memory) Memoria de Sólo Lectura**: Estas memorias son programadas permanentemente por el fabricante del semiconductor. Este tipo de memoria es conocido como mascarable, para diferenciarlos de otros tipos de memoria. En los últimos pasos finales del proceso de manufactura, el patrón binario es introducido mediante una técnica llamada ***enmascaramiento***.

b) **PROM (Programmable Read Only Memory) Memoria de Sólo Lectura Programable**: Este tipo de dispositivo, a diferencia del anterior, no necesita programarse por el fabricante. El mismo usuario puede programarlo. Su diseño se basa en el método de fusibles enlazados, que pueden ser fundidos o no dependiendo del patrón de bits. La desventaja de este tipo de memoria es su nula capacidad de reprogramación; razón por la cual, debe asegurarse que el programa sea 100% seguro (libre de errores).

c) **EPROM (Erasable Programmable Read Only Memory) Memoria de Sólo Lectura Reprogramable**: Estos tipos de memorias pueden ser programadas, borrar la información y volver a programarlas (a esto se le denomina **reprogramación**). La información contenida en la **EPROM** puede ser borrada aplicando luz ultravioleta durante aproximadamente un periodo de 20 minutos en la ventanilla que tienen para tal fin. Estos dispositivos operan en el rango de +5 volts de alimentación.

   Este tipo de memoria presenta las siguientes desventajas:

   1) Para ser borrados, necesitan ser removidos del circuito.

   2) Requieren un tiempo grande para ser borrados.

   3) Tienen un ciclo de vida limitado (100 a 1000 borrados y reprogramación).

4) El dispositivo completo es borrado (borrado por bloques), y no por bytes individuales

d) **EEPROM (Electrically Erasable Programmable Read Only Memory) Memoria de Sólo Lectura Programable Eléctricamente Borrable** Estos dispositivos de almacenamiento permiten borrar su información y ser reprogramados mediante un pulso eléctrico. Esta característica permite su manipulación sin necesidad de removerlo del circuito.

Aún cuando estos dispositivos operan en el rango de +5 volts, es necesario aplicar un voltaje en el rango de +17 a +20 volts para poder borrar y programar la memoria.

1.3.3 Memoria de Datos

Dado que los sistemas computadores manejan información que cambia constantemente, esto es, normalmente el microprocesador recibe datos de entrada desde dispositivos periféricos conectados mediante **Puertos de Entrada/Salida** tales como teclados, convertidores analógico/digitales, etc. Efectúa cálculos sobre estos datos y los almacena para posteriores acciones. Además de lo anteriormente dicho, los microprocesadores cuentan con **Registros de Propósito General** que le permiten almacenar datos intermedios, sin embargo estos registros son limitados, razón por la cual los sistemas microcomputadores deben de tener un medio en el cual puedan almacenar una gran cantidad de información que está en continuo cambio. A este medio se le denomina **Memoria de Datos**. La **Memoria de Datos** consiste en su parte más esencial por memorias **RAM** (este termino será definido más profundamente en la siguiente subsección). A diferencia de la **Memoria de Programa**, la **Memoria de Datos** puede tener una longitud de palabra deseada (esto es, si el microprocesador maneja una palabra de un octeto de datos y la **Memoria de Datos** tiene capacidad para manejar palabras de dos octetos de datos, se puede usar una localidad para almacenar dos bytes de datos).

Un punto importante a tomar en cuenta para la **Memoria de Datos**, es la velocidad de acceso a los datos, ya que si esta velocidad es inferior a la velocidad del microprocesador, se debe de diseñar un circuito que permita introducir estados de espera (**NOP**) en el microprocesador hasta que la memoria **RAM** haya decodificado la dirección de la cual va a extraer el dato y lo haya puesto a disposición del microprocesador en el **Bus de Datos**.

Para este caso, la tecnología más común es la utilización de memorias **RAM NMOS**, las cuales tienen una velocidad de acceso a los datos de 100 ns o menos.

En sistemas microcomputadores más modernos, suelen usarse memorias **RAM Estáticas** para fungir como memoria cache y memorias **RAM Dinámicas** para fungir como memoria principal, ya que al carecer de elementos de refresco, las memorias **RAM Estáticas** son más rápidas, sin embargo tienen el inconveniente de ser más caras y de difícil manejo cuando sobrepasan los 640Kb de memoria.

De la misma forma, al poder disponer de mayor capacidad de direccionamiento y por ende de mayor capacidad de memoria, los modernos sistemas microcomputadores suelen dividir la memoria RAM en:

a) **Memoria Convencional**: Está formado por los primeros 640 Kb de memoria **RAM**, éstos pueden ser directamente direccionados por el sistema operativo. Las siguientes 384 Kb direcciones de memoria(denominados **Memoria de Área Superior**) hasta el límite de 1 Mb son utilizados para manejar funciones del vídeo e instrucciones de la **ROM**.

b) **Memoria Expandida**: Al tener el sistema operativo hasta un límite de 640 Kb de memoria para ejecutar un programa y no poder direccionar más allá de 1 Mb, las compañías Lotus, Intel y Microsoft desarrollaron un estándar para una tarjeta de memoria adicional llamada **EMS (Especificación de Memoria Expandida)**, esta memoria no tiene direcciones que correspondan a la memoria física y fue desarrollada para sobrepasar los límites de la memoria física de las computadoras XT. Debido a que los programas que usan memoria expandida la acceden en bloques de 64 Kb, el uso de este tipo de memoria es más lento.

c) **Memoria Extendida**: Esta zona de memoria se ubica en las localidades a partir de 1 Mb y requiere de un administrador de memoria extendida, tal como **HIMEM**.

De la misma forma en sistemas computadores modernos, se usa la **Memoria de Datos** para alojar el sistema operativo (conjuntando programa y datos en un solo tipo de memoria)

1.3.4 Memoria RAM

Las siglas de las memorias **RAM** se derivan de las palabras anglosajonas **Random Accces Memory (Memoria de Acceso Aleatorio)**. Este tipo de dispositivo se ubica dentro de la categoría de **Memorias Volátiles**(al ser suprimida su fuente de alimentación, pierden su contenido). En una memoria **RAM**, sus celdas de información pueden ser accesadas para lectura o escritura.

Dependiendo de la forma de almacenar la información en sus celdas, las memorias **RAM** se pueden subdividir en:

a) **RAM Estática**: Este tipo de memoria almacena los datos en flip-flops. Mientras tenga una fuente de alimentación, la información es retenida en los flip-flops, al momento de desconectar la fuente, la información en éstos se pierde.

b) **RAM Dinámica**: Este tipo de memoria almacena los datos en un capacitor. Después de cierto periodo de tiempo, el capacitor se descarga, razón por la cual, la información en la memoria se pierde. Para evitar este tipo de contratiempos, es necesario refrescar periódicamente la memoria. La mayoría de las memorias dinámicas requieren un refresco periódico de aproximadamente 2 ms. Aunque algunos dispositivos requieren un refresco de hasta 4 ms. Para tal efecto, todos los circuitos que utilicen este tipo de memoria, necesitan de una circuitería adicional para proporcionar el refresco a las memorias.

1.4 Puertos de Entrada/Salida

Antes de proceder a mencionar los diversos dispositivos que pueden conectarse a un sistema microcomputador y que fungirán como dispositivos periféricos de entrada/salida de datos, debe saber que éstos no tienen una comunicación directa con el microprocesador y que requieren de un elemento (interfaz) que le permita comunicarse con dicho microprocesador. Sin embargo, para que pueda existir la comunicación entre el microprocesador y los dispositivos periféricos a través de las interfaces, éstas deben de conectarse a una dirección del bus de direcciones y que además se encuentren ubicadas dentro del rango del mismo bus de direcciones o del rango que esté definido en el microprocesador para ser usado como puerto de entrada/salida (en el caso de los **Puertos de Entrada/Salida Aisladas**). A la combinación de la dirección y de la interfaz conectada a ella para comunicarse con un dispositivo periférico se le denomina **Puerto**. A partir de este concepto, se definen los **Puertos de Entrada** como la combinación de una dirección y de una interfaz que permita la entrada de datos hacia el microprocesador desde un dispositivo periférico(**Lectura de Puertos**) y el **Puerto de Salida** como la combinación de una dirección y de una interfaz que permitan la salida de datos desde el microprocesador hacia un dispositivo periférico(**Escritura de Puertos**). Debido a que la combinación de las señales de control son las que determinan la operación que se hará sobre un puerto, este mismo puerto puede fungir tanto de entrada como de salida.

Dependiendo de las señales de control usadas y de las direcciones a las cuales se conectan las interfaces, los **Puertos de Entrada/Salida** se pueden clasificar en:

a) **Puertos de Entrada/Salida Mapeadas**.
b) **Puertos de Entrada/Salida Aisladas**.

Estos tipos de **Puertos de Entrada/Salida** serán estudiados con mayor profundidad en secciones posteriores.

Estos conceptos se aplican a todos los sistemas de cómputo, ya que los dispositivos periféricos requieren forzosamente de una interfaz que les permitan "hablar" el mismo idioma que el microprocesador, además de almacenar la información enviada si el microprocesador se encuentra ocupado y posterga su atención hacia el dispositivo periférico que envió la información.

1.5 Dispositivos de Entrada

Son conocidos popularmente como dispositivos periféricos de entrada. Cumplen la función de servir como medios de comunicación entre el usuario o el medio ambiente y el microprocesador. Su principal objetivo es la introducción de datos al sistema.

Dentro de este rubro se consideran como dispositivos de entrada a los siguientes elementos.

1) **Teclado**. En su parte más esencial, este dispositivo está formado por una matriz de teclas decodificadas por un microcontrolador diseñado para tal fin (circuito integrado **8048**, o afín). Este chip utiliza una técnica de exploración para monitorear la matriz del teclado. Cada una de las teclas está conectada a una de las intersecciones renglón-columna, las cuales emiten un nivel alto si una tecla no está presionada. Al ser presionada una tecla, el procesador recibe una señal de nivel bajo. La tecla presionada se decodifica y se le hace corresponder con un carácter en la **ROM DE CARACTERES**. El patrón de bits se envía a través del cable de datos a la tarjeta del sistema (interfaz de teclado). La siguiente figura muestra en forma gráfica la distribución de los elementos que conforman a un teclado.

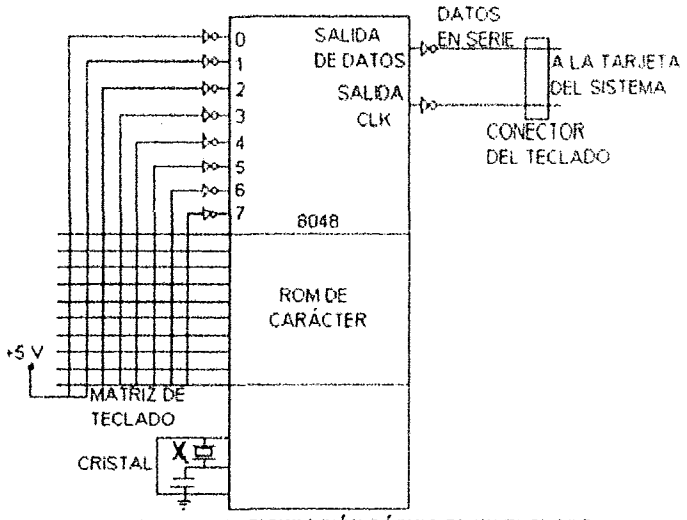


Fig. 1.7 CONFIGURACIÓN BÁSICA DE UN TECLADO

Para eliminar los rebotes que se producen al momento de presionar una tecla y al momento de soltarlo, existen dos métodos:

1. **En los Teclados no Codificados**: Un resistor y un capacitor están conectados como un filtro para reducir las oscilaciones y el efecto de rebote.
2. **En los Teclados Codificados**: Se utiliza un retardo de unos pocos milisegundos antes de que el golpe de la tecla se codifique. El retardo se logra con un ciclo programado que inserta el retardo (a esto se le denomina **eliminación de rebotes**).

El chip 8048 genera una interrupción durante el tiempo en que el voltaje del teclado oscila.

Los teclados cuentan con un subgrupo de teclas a saber:

a) **Teclado principal**: Agrupa las teclas típicas de una máquina de escribir tales como letras, números, signos de puntuación, etc.

b) **Teclado numérico** Contiene los diez dígitos (0-9) y algunas teclas de funciones tales como "+", "-", "*", "/"

c) **Teclas de funciones programables** Contienen teclas que pueden ser programadas por el usuario, dichas teclas son F1 F1 F12

Los teclados se clasifican de acuerdo a los contactos de las teclas y a su diseño ergonómico

De acuerdo a los contactos de sus teclas, los teclados se clasifican en

1. **Contactos por Lámina** Son los más baratos y más comerciales Usan dos membranas de fibra plástica en las que se imprimen los contactos y sus conexiones Éstas se recubren de un material aislante a fin de que sus superficies de contacto permanezcan ligeramente separadas. Al presionar la tecla se cierran los contactos y sus circuitos Una pieza de espiral es ocupada para levantar la tecla presionada
2. **Contactos Auténticos de Metal** Su funcionamiento es parecido al anterior con la salvedad de que son más caros Al pulsar una tecla, se retraen unas lenguetas metálicas Para evitar la corrosión de dichas lenguetas, éstas se bañan en oro u otro material protector.
3. **Teclas Carentes de Contactos**: Se utilizan tres métodos diferentes:
   a) **Piezo Eléctrico**: debajo de cada tecla se encuentra una pequeña pieza de cristal, la cual al ser presionada genera una pequeña tensión electrica
   b) **Opto Eléctrico**: Se utiliza un indicador luminoso debajo de cada tecla, la cual se encarga de producir la señal de la tecla pulsada
   c) **Magneto Mecánico**: Un imán permanente es colocado bajo las teclas, al ser pulsada la tecla se conecta a un sensor magnético, y desde él, se genera un cambio del estado eléctrico que el teclado se encarga de reflejar.

Para realizar la conexión del teclado con la interfaz presente en la tarjeta madre, se utiliza un conector **DIN** de 5 pines o un conector **MINIDIN** de 6 pines

2) **Mouse**: Es un dispositivo de la tecnología de las computadoras, el cual es semejante a un ratón en apariencia y movimiento, que permite al usuario controlar el movimiento del cursor en una pantalla de video, rodando el dispositivo sobre una superficie Se usa también para ejecutar comandos, seleccionar bloques de texto y otras funciones variadas

En su parte más elemental, están compuestos por rodillos o LEDs y una superficie metálica graduada. El movimiento del dispositivo hace que los rodillos, los potenciómetros internos o el LED (dependiendo de su forma de construcción) permitan detectar las líneas graduadas. Estos cambios de voltaje son utilizados para detectar los movimientos relativos del ratón y rastrear la posición relativa del cursor sobre la pantalla. El movimiento se convierte a valores digitales y son usados por el microprocesador para calcular la dirección y magnitud del movimiento del ratón.

El ratón normalmente se mueve sobre una superficie plana llamada mouse-pad. La computadora mantiene un registro de posición actual del ratón que se incrementa o diminuye por medio del movimiento del mismo. El ratón normalmente tiene uno o más botones que se usan para activar comandos.

Los ratones pueden ser subdivididos en.

a) **Ratón Óptico**: La posición de rastreo del ratón óptico se realiza por medio del conteo del número de líneas (o puntos) que se cruzan sobre un cojín especial. Al no existir contacto entre el ratón y la superficie y como no existen partes en movimiento, el ratón óptico es confiable. Este dispositivo no requiere de mantenimiento aún cuando requiera de una almohadilla especial para su funcionamiento óptimo.

Algunos movimientos de rastreo del ratón óptico tienen relación con la orientación de la almohadilla. El ratón no puede girar más de 90° en cualquier dirección.

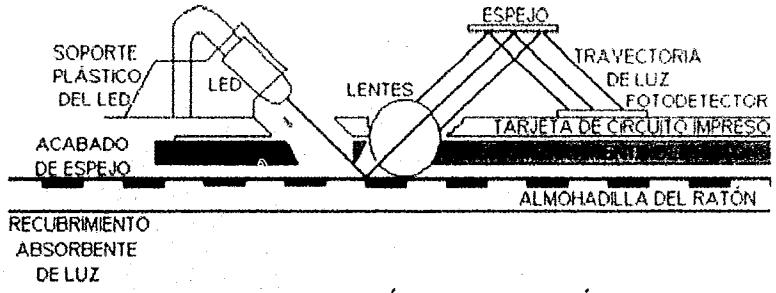


Fig. 1.8 DIAGRAMA ESQUEMÁTICO DE UN MOUSE ÓPTICO

b) **Ratón Mecánico**: Está compuesto de una esfera mecánica para controlar dos codificadores de varilla mecánica ortogonal. Ésta es una tecnología barata y le permite movimientos de tracción en cualquier superficie, sin embargo tiene una vida útil limitada (debido a la conmutación mecánica de los codificadores) y requieren de una limpieza periódica.

c) **Ratón Opto-Mecánico**: Está compuesto de una esfera para impulsar dos codificadores de varilla óptica ortogonal. Esta tecnología es barata y permite una vida útil más prolongada (comparada con la mecánica) y le permite rodar en cualquier superficie, aunque también requieren de limpieza periódica.

d) **Acústico-Mecánico**: Está compuesto de medidores de tensión para determinar la dirección y un transductor piezoeléctrico para determinar la magnitud. No tiene partes en movimiento, no necesita mantenimiento y el ratón puede deslizarse casi en cualquier superficie. La resolución depende del tipo de superficie en la cual se utiliza el ratón y la presión aplicada en éste.

e) **Ratón Analógico**: Esta tecnología de diseño puede utilizar una esfera o dos ruedas metálicas montadas ortogonalmente que impulsan la varilla de un potenciómetro. Entre las

desventajas de este método se pueden mencionar las siguientes: tienen una vida útil limitada y una cantidad acotada de recorrido en cualquier dirección debido al uso del potenciómetro.

En general, se usa cualquier clase de almohadilla o tapete con un ratón no óptico para reducir la fricción, el ruido audible y para proteger el escritorio. Esta almohadilla con frecuencia es rígida.

Los ratones pueden ser conectados a las computadoras mediante una interfaz serie o paralelo. Para la interfaz paralelo, se utiliza una codificación en cuadratura para el movimiento y codifica las depresiones del interruptor como un bajo voltaje TTL.

Para la interfaz serie, se conecta a un puerto RS-232.

3) **Scanner**: Es un dispositivo análogo a una fotocopiadora, con la diferencia de que las imágenes son digitalizadas y convertidas en un formato que las microcomputadoras pueden entender. Existen una gran variedad de scanners, entre las cuales se pueden encontrar:
   a) **Scanners Basados en Cámara**: Utilizan tecnología fotográfica para producir digitalizaciones de muy alta resolución en objetos bidimensionales o tridimensionales. La cabeza del scanner podría semejarse a una cámara de 35 mm que puede montarse en una torre móvil sobre una plataforma plana.
   b) **Scanners de Alimentación**: Son semejantes a las impresoras de hojas sencillas. Están diseñados para explorar documentos de tamaño carta, aunque pueden soportar tamaños más grandes y más pequeños.
   c) **Scanners de Cama Plana**: son utilizados para digitalizar objetos planos bidimesionales como documentos y fotografías. Algunos ofrecen un aditamento de alimentación de páginas múltiple para la exploración automática de un gran número de páginas.
   d) **Scanners de Deslizamiento**: Este tipo de scanners son utilizados para exploración de alta resolución a color de las diapositivas de 35 mm. Soportan salidas de nivel de 256 en una escala de grises para conversión de escala de grises a diapositivas a color o en blanco y negro.
   e) **Scanners de Tipo Aéreo**: utilizan una plataforma de digitalización con la cabeza de exploración fija arriba de la plataforma. La plataforma permite la exploración de pequeños objetos tridimensionales.
4) **Lector Óptico**: la mayoría de los **Lectores Ópticos** consisten de un cilindro de plástico del tamaño de un lápiz con un sensor de luz en un extremo. El otro extremo se conecta a la computadora por medio de un cable. A medida que se posiciona la punta del lápiz sensible a la luz para seleccionar un punto en la pantalla. El lápiz óptico envía un pulso cuando esta área de la pantalla es brillante. El pulso producido por el lápiz óptico se utiliza entonces para calcular la posición de la pantalla en donde estaba el lápiz.

El lápiz detecta la ráfaga de luz fluorescente emitida cuando el haz de electrones está bombardeando al fósforo. La luz se emite durante el dibujo del elemento de imagen. La salida del lápiz normalmente está conectada a la lógica de control del sistema de tal manera que el microprocesador se detiene ejecutando comandos cuando el lápiz detecta la luz.

Además de los anteriores periféricos de entrada, existe una gama infinita de dispositivos que pueden ser utilizados como elementos de entrada de datos al sistema computador, tales como los convertidores analógicos-digitales, etc.

1.6 Dispositivos de salida

Dentro de los dispositivos de salida que con mayor frecuencia se encuentra dentro de un equipo de cómputo, se ubican:

1) **Impresoras**: existe una gran variedad de impresoras, las cuales se pueden subdividir en:
   a) **Impresora de Margarita**: Su principio de impresión es parecido a las máquinas de escribir. La margarita es un pequeño disco dividido en pequeños segmentos. Al final de cada segmento se encuentra la forma de un carácter. Esta rueda sobre sí misma hasta la posición correcta. Seguidamente, un martillo golpea el tipo de impresión contra la cinta entintada, imprimiendo el carácter sobre el papel.
   b) **Impresora de Bola (o Esfera)**: Su principio de funcionamiento es análogo a la impresora de margarita, a diferencia de que utiliza una bola de tipos, o cabeza de impresión esférica que contiene los tipos de escritura en la superficie, la bola rueda sobre sí misma para alinear los caracteres, posteriormente la impresora la golpea contra la cinta para dejar impreso el carácter en el papel.
   c) **Impresoras Matriciales (de Impacto)**: Crean el carácter moviendo en forma selectiva un conjunto de diminutas agujas para formar un patrón de puntos en la forma del carácter deseado. Típicamente se utilizan de siete a nueve agujas en impresoras de baja resolución y de 18 a 24 agujas en impresoras de alta resolución. Entre más agujas tenga la unidad, más se aproximará el mecanismo de impresión a la resolución de la calidad de imprenta para los caracteres totalmente formados.

      Los caracteres que serán impresos se almacenan en una **ROM**, ésta determina cuáles agujas de la matriz van a golpear en la cinta e imprimir el carácter en el papel.

      Las impresoras de matriz de puntos utilizan dos tecnologías para la alimentación del papel: la primera consiste de un rodillo de presión que impulsa el papel por medio de la fricción, la segunda utiliza un conjunto de tractores que jalan el papel longitudinalmente con los dientes que se acoplan a los agujeros del papel.

      El papel puede ser de formas continuas (con o sin orificios para el tractor) en rollos u hojas sueltas.

d) **Impresoras de Inyección de Tinta**. Son una variante de las impresoras matriciales. diminutas gotitas de tinta son forzadas a pasar al papel. La cabeza se desplaza en sentido horizontal gracias a la acción de un motor, mientras el papel va desplazándose en sentido vertical accionado por un tractor o dos rodillos. La cabeza dispone de varias toberas dispuestas verticalmente que dan paso a diminutas gotas de una tinta especial, que se esparcen sobre el papel.

e) **Impresoras Lasser**: Al proceso de impresión con tecnología lasser también se le denomina **electrofotografía**. Dentro de ésta, existen varios procesos de impresión, a saber:

1) **Térmica**: Los elementos calentados escriben sobre papel térmicamente sensible.
2) **Electrosensible**. Cargas eléctricas queman una cubierta metálica delgada para exponer una cubierta inferior negra.
3) **Electrostática**: El papel recubierto se carga selectivamente por un estilete y luego se pasa a través de un tonner que es atraído por la carga.

2) **Monitores**: El monitor al igual que el teclado son los dispositivos periféricos más utilizados en los sistemas de computador. Se encuentran internamente formados de:

a) **Un Tubo de Rayos Catódicos (CRT) o Válvula Termoiónica de Alto Vacío** en el que un rayo catódico barre la totalidad de la superficie a razón de 25 veces por segundo.

Este barrido produce 625 líneas, mediante las cuales el choque de los electrones sobre la capa interior fosforescente produce una trama visible de 625 líneas, cuya modulación de luminosidad produce la imagen.

b) **Etapa Elevadora de Tensión**: Está compuesta por un oscilador de frecuencia de 16625 Hz. Esta salida amplificada es aplicada a un transformador de núcleo de ferrita que se encarga de obtener las diferentes tensiones que requiere el **CRT** para el enfoque, aceleración del rayo catódico y la muy alta tensión (alrededor de 12000 V), que crea el campo eléctrico para atraer hacia la pantalla los electrones que parten del cátodo.

c) **Circuitos de Gobierno**: Encargados de coordinar los elementos mencionados y que además cumplen con las funciones de amplificar la señal a presentar en la pantalla, generar la frecuencia de línea y de cuadro, separar la señal de vídeo y de sincronismo y controlar el brillo y la luminosidad de la pantalla.

Generalmente, los monitores se pueden clasificar dentro de las siguientes categorías:

a) **Monocromáticos**: Oscilan a una frecuencia de 18.432 KHz y a una frecuencia vertical de 50 Hz. Están disponibles en color ámbar o verde. estos monitores mostrarán un carácter de textos de 9x16 pixeles con 25 líneas de 80 columnas.

Los adaptadores de vídeo monocromáticos estándar exhiben gráficos de 729x348 pixeles, pudiendo emular pantalla de gráficos a color (CGA), gráficos mejorados o de matriz de gráficos de vídeo (VGA).

b) **RGB**: Comúnmente se denominan monitores **CGA**, tienen una frecuencia horizontal de 15.75 KHz y una vertical de 60 Hz. Las entradas separadas del rojo, verde y azul y la intensidad permiten un total de 16 colores exhibidos. Los caracteres son de 8 pixeles de altura por 8 pixeles de ancho y una exhibición en pantalla de 80x25 caracteres. Los gráficos se pueden mostrar hasta un máximo de 640x200 pixeles.

c) **Monitores de frecuencia dual**: Es una pantalla que puede operar ya sea a una frecuencia de sincronía monocromática o color. Normalmente tiene una pantalla monocromática y producen tonalidades cuando se operan como una pantalla a color. Pueden cambiar de frecuencias de sincronia monocromática (18.432 KHz horizontal, 50 Hz vertical) a frecuencias de sincronia a color (15.75 KHz horizontal, 60 Hz vertical).

d) **Monitores EGA**: Exhiben caracteres de 8 x 14 pixeles en una pantalla de 80x25. Pueden ser controlados mediante adaptadores **CGA** o **EGA** para entregar resoluciones gráficas desde 320x200 pixeles hasta 640x350 pixeles. La frecuencia de sincronia vertical es constante a 60 Hz. Se puede mostrar un total de 16 colores de 64 posibles. Los colores se generan a través de entradas primarias y secundarias del rojo, verde y azul.

e) **Monitores VGA**: Funcionan a una frecuencia horizontal de 3.15 KHz y a una frecuencia vertical de 70 Hz. Proporcionan una resolución de 640x480 pixeles. Utilizan un esquema de entrada analógica **RGB** para obtener los 262144 colores diferentes disponibles en los modos **VGA**.

3) **Plotters**: El plotter tiene la función de sustituir al dibujante técnico: traza lineas desplazando una pluma sobre el papel. Este dispositivo es utilizado normalmente en el área de dibujo técnico. La mayoría de los plotters llevan incorporada una función de cambio de plumas para poder dibujar gráficos con distintos colores.

Además de los dispositivos de salida anteriores, existe una gama infinita de dispositivos periféricos que actúan como tales, y a medida que avanza la tecnologia, diariamente se van incrementando más dispositivos que tienen como fin mostrar los resultados obtenidos por un sistema cornputador.

1.7 Resumen

Los microprocesadores surgen como una necesidad del ser humano de liberarse de las tareas monótonas y repetitivas. Sin embargo, la potencia de cálculo de estos dispositivos ha ido en constante aumento, de tal forma que en ocasiones ha sustituido al ser humano o ha ampliados sus capacidades físicas y mentales para procesar datos donde el ser humano no puede.

Las características fundamentales de los microprocesadores son su Programabilidad y su Universalidad, las cuales les permiten fungir como elementos "inteligentes" en las aplicaciones de sistemas digitales.

Los microprocesadores se clasifican de acuerdo al número de bits que puede manejar en su bus de datos como:

1) Microprocesadores de 4 bits.
2) Microprocesadores de 8 bits.
3) Microprocesadores de 16 bits.
4) Microprocesadores de 32 bits.
5) Microprocesadores de 64 bits.

Estructuralmente hablando, los microprocesadores están formados por las siguientes unidades básicas:

1) Unidad de Control: Esta sección genera la serie de pulsos adecuados para controlar las operaciones del microprocesador.
2) Unidad Aritmético/Lógica: Esta sección funge como el "cerebro" del sistema, permitiendo realizar operaciones de cálculo u operaciones lógicas.
3) Registros de Trabajo: Esta sección permite el almacenamiento temporal de datos, incrementando la rapidez de las microoperaciones.

Para que un sistema microcomputador pueda realizar todo tipo de operaciones, requiere forzosamente que el conjunto de instrucciones que debe ejecutar se encuentre almacenado en un medio que no permita perder los datos aún cuando se le suprima su alimentación (**Memoria de Programa**) y un medio donde pueda almacenar información temporal (**Memoria de Datos**).

Además de lo anterior, todo sistema microcomputador requiere de un medio de entrada de datos (**Dispositivos Periféricos de Entrada**) y de un medio de salida de datos (**Dispositivos Periféricos de Salida**), la interconexión de estos **Dispositivos Periféricos de Entrada/Salida** de datos se comunican con el microprocesador mediante una interfaz. Los elementos mencionados permiten al ser humano introducir datos en una microcomputadora, la cual realiza operaciones sobre estos datos y devuelve los resultados al mundo exterior en forma visual o impresa.

CAPITULO II:
MICROPROCESADOR Z80

"Me pongo muy nervioso leyendo la prensa comercial. Las revistas pretenden hacerle creer que el único microprocesador viable incluso para las aplicaciones simples es una 80486 a 50 MHz. o una latosa máquina RISC. ¿Cuántos diseñadores usan estos CPUs en sus propios diseños?"

Jack G. Ganssle

2.1 Arquitectura del Microprocesador Z80

El microprocesador Z80 es un circuito integrado diseñado y manufacturado por la Cía. Zilog. Éste es una versión mejorada del microprocesador 8080 de la compañía Intel. La siguiente figura muestra el diagrama a bloques de la arquitectura interna del microprocesador Z80.

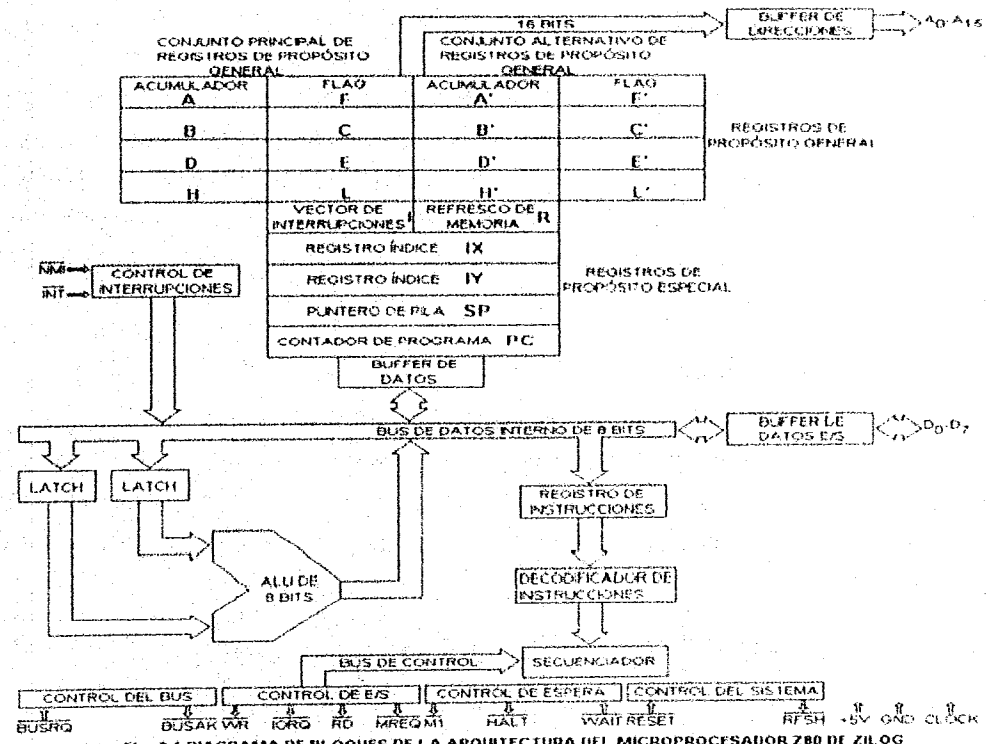


Fig. 2.1 DIAGRAMA DE BLOQUES DE LA ARQUITECTURA DEL MICROPROCESADOR Z80 DE ZILOG

El microprocesador Z80 a diferencia de su contraparte 8080, tiene un conjunto de instrucciones que llegan hasta los 158 (¡más del doble del 8080!). Además de requerir de un ciclo

de reloj unifase(el 8080 requiere de un ciclo de reloj bifase). Su velocidad de procesamiento oscila entre 2.5 MHz. y 4.0 MHz. (Z80A). Para su alimentación, requiere una sola fuente de +5 volts.

Además de las características anteriores, el microprocesador Z80 cuenta con la lógica interna para manipular interrupciones de dispositivos periféricos y refresco de memoria dinámica.

El microprocesador Z80 está construido con la tecnología **MOS (Metal Oxid Semiconductor)** de **canal N**. Por lo que se requiere un cuidado extremo en su manejo, para evitar dañarlo en forma involuntaria con la electricidad estática del cuerpo humano.

Como se puede ver en la gráfica anterior, este microprocesador cuenta con las siguientes secciones: **Unidad Aritmética/Lógica (ALU)**, **Registros de la CPU (Registros de Propósito General, Registros de Propósito Especial)**, **Registro de Instrucciones**, más una sección que se encarga de decodificar las instrucciones recibidas y controlar la dirección colocada en el **Bus de Direcciones (Decodificador de Instrucciones)**.

El microprocesador Z80 cuenta con un **Bus Bidireccional** de datos de 8 bits y un **Bus Unidireccional de Direcciones** de 16 bits. Gracias al número de bits de capacidad del **Bus de Direcciones**, el microprocesador Z80 puede direccionar hasta 65536 (64 Kb) localidades diferentes de memoria.

El microprocesador Z80 cuenta con un conjunto total de 208 bits de **Registros Internos** que pueden ser objeto de lectura o escritura, siendo éstas accesibles por el programador. El número total de registros se distribuye en un conjunto de 8 **Registros Principales** de **Propósito General** de 8 bits cada uno, un conjunto de 8 **Registros Alternos** de 8 bits cada uno, un conjunto de 4 **Registros de Propósito Especial** de 16 bits cada uno y finalmente, un registro **Contador de Programa (PC)** de 16 bits.

2.2 Registros del Microprocesador Z80

El microprocesador Z80 está compuesto por un conjunto de **Registros Principales**, a saber: un **Acumulador (A)**, un **Registro de Estado** (registro F o Flag), y seis **Registros de Propósito General (B, C, D, E, H y L)**. Adicionalmente, cuenta con un conjunto de **Registros Alternos**, que duplican los registros anteriores (A', F', B', C', D', E', H' y L') como se puede apreciar en la siguiente figura:

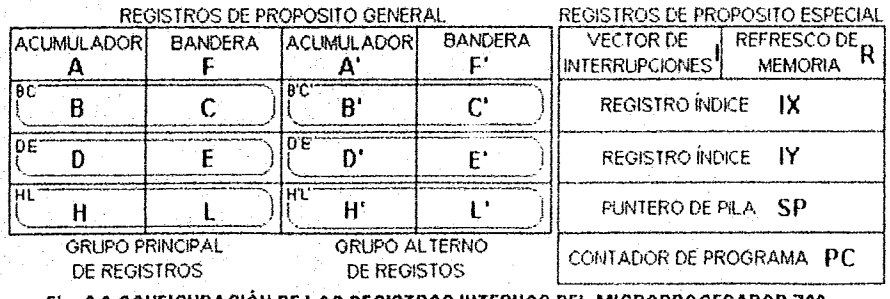


Fig. 2.2 CONFIGURACIÓN DE LOS REGISTROS INTERNOS DEL MICROPROCESADOR Z80

Los registros anteriores sólo pueden ser usados una vez en cada momento, es decir, para poner un conjunto de datos en el registro B' debemos primero tener los datos en el registro B y posteriormente usar una instrucción de intercambio que opere en ambos registros. La tabla siguiente muestra el mnemónico y el valor en binario asociado a cada registro:

Registro	Código	Registros	Código
B	000	BC	00
C	001	DE	01
D	010	HL	10
E	011	SP	11
H	100		
L	101		
A	111		

Tabla 2.1 CÓDIGO BINARIO DE LOS REGISTROS

Los **Registros de Propósito General** se pueden agrupar por pares para formar un conjunto de registros de 16 bits cada uno (formando los **Registros Pares BC, DE y HL**). Otro tanto puede ocurrir con los **Registros Alternos** (formando los **Registros Pares B'C', D'E' y H'L'**).

Dentro del grupo de **Registros de Propósito Especial** se encuentran:

1) **Vector de Interrupciones (I)**: Está formado por un byte u ocho bits. Su función principal es servir a las interrupciones originadas por los dispositivos periféricos. Al ser originada una interrupción de un dispositivo periférico, el microprocesador brinca a una localidad de memoria que contiene la rutina cuya función es la de brindar los servicios requeridos al dispositivo periférico solicitante. Al originarse la interrupción, un dispositivo especial denominado **PIC (Programmable Interruption Controller)** o **Controlador Programable de Interrupciones** proporciona los 8 bits de menor peso de la dirección de 16 bits, en tanto que el **Vector de Interrupciones** origina los restantes 8 bits de mayor peso, para conformar la dirección completa.
2) **Registro de Refresco de Memoria (R)**: Está formado por un byte u ocho bits. Su función principal es la de refrescar la memoria dinámica. Este proceso de refresco se realiza en las etapas de decodificación y ejecución de la instrucción. Después de cada instrucción de búsqueda, 7 bits del registro son incrementados, en tanto que el octavo bit se programa mediante la instrucción **LD R, A.** En el proceso de refresco de la memoria, se activa la señal correspondiente ocasionando que el contenido del registro **R** sea colocado en los 8 bits de menor peso del **Bus de Direcciones**. Además, el contenido del registro **I** es colocado en los 8 bits de mayor peso del **Bus de Direcciones**.
3) **Registro Indice IX**: Tiene una longitud de dos bytes ó 16 bits. Su función principal es la de apuntar a una localidad de memoria externa en las instrucciones de direccionamiento indirecto. La localidad de memoria activa se suma al contenido del **Registro Índice IX** y a un desplazamiento entero **d**.

4) **Registro Indice IY**: Tiene la misma longitud que el registro par **IX**, de igual forma su función es análogo al anterior registro. Los registros índices pares **IX** e **IY** son independientes uno de otro.
5) **Puntero de Pila (SP)**: Tiene una longitud de dos bytes ó 16 bits, su función principal es almacenar los 16 bits de dirección de una pila **LIFO (Last Input First Output)** o último en entrar primero en salir en una memoria externa. La introducción y la extracción de datos de la pila se realizan a través de las instrucciones **PUSH** y **POP**.
6) **Contador de Programa (PC)**: Tiene una longitud de dos bytes ó 16 bits. La función principal del registro **PC (Program Counter)** es almacenar la dirección de la siguiente instrucción a ser ejecutada. Dependiendo del número de bytes de una instrucción, el registro **PC** se incrementa en ese valor. Por ejemplo, si una instrucción tiene una longitud de un byte, el registro **PC** se incrementa en uno (**PC=PC+1**), si tiene dos bytes de longitud, el registro **PC** se incrementa en dos (**PC=PC+2**), etc. Si una instrucción **JUMP** se ejecuta, el registro **PC** almacena la dirección a la cual el programa deberá brincar. Al momento de ejecutar la instrucción **RET**, el registro **PC** deberá contener la dirección de la siguiente instrucción a ser ejecutado como si ninguna instrucción de salto se hubiera efectuado.
7) **Registro de Estados (Flag)**: El microprocesador Z80 cuenta con una par de registros que continuamente muestran el estado del microprocesador: **F** y **F'**. La longitud de ambos registros es de 8 bits cada uno. Cada uno de los bits de los registros anteriores muestran un estado diferente del microprocesador.

Los estados de los registros **F** y **F'** son activados o desactivados después de cualquier operación aritmética ó alguna operación que actualice los datos contenidos en los restantes registros del microprocesador. La distribución de los bits de estado, su mnemónico y la acción que realiza se pueden ver en la siguiente tabla:

Bit	Flag	Descripción	Acción
0	C	Carry	Se activa si el resultado de una operación lógica o aritmética excede la capacidad del registro (acarreo del bit más significativo del acumulador)
1	N	Substract	Se activa si la última operación efectuada fue una substracción (operación en BCD)
2	P/V	Parity/Overflow	Se activa Parity si existe un número par de bits. Se activa Overflow si el resultado de un complemento a 2 excede la capacidad de un registro
3	F3	Undocumented	Copia del bit 3 (Indeterminado)
4	H	Half Carry	Acarreo del bit 4 en operaciones de BCD
5	F5	Undocumented	Copia del bit 5 (Indeterminado)
6	Z	Zero	Se activa si el resultado de una operación es cero
7	S	Sign	Se activa si el resultado de una operación (complemento a 2) es negativo

Tabla 2.2 ESTRUCTURA DE LA BANDERA DE ESTADOS (FLAG) DEL MICROPROCESADOR Z80

2.3 Señales de Control del Microprocesador Z80

Para poder dirigir las operaciones que se han de realizar sobre los distintos **Dispositivos Periféricos** conectados mediante los **Puertos de Entrada/Salida** al sistema, así como sobre el acceso a la **Memoria de Programa** o la **Memoria de Datos** y la sincronización de tales operaciones, el microprocesador Z80 cuenta con un conjunto de señales que se activan dependiendo de la acción que se va a realizar.

2.3.1 Señales de salida

Las señales de salida son generadas por el microprocesador, sólo el propio microprocesador puede manejar estas señales. Una descripción más exacta de tales señales, así como de la función que realizan se da en la siguiente tabla:

Mnemónico	Descripción	Acción
A_0-A_{15}	Bus de Dirección	Permite direccionar hasta 65536 bytes de memoria (64 Kb). Adicionalmente, direcciona 256 **Puertos de Entrada/Salida** en los 8 bits de menor peso. El **Bus de Direcciones** tiene una salida triestado activa en ALTO.
$\overline{MREQ}$	Señal de Petición de Memoria	Activa en BAJO. Indica que la dirección presente en el **Bus de Direcciones** está lista para realizar cualquier operación de lectura o escritura en la memoria.
$\overline{IOREQ}$	Señal de Petición de Entrada/Salida	Triestado, activa en BAJO. Indica que una operación de entrada/salida está tomando lugar. El byte de menor peso del **Bus de Direcciones** indica el puerto que ha sido seleccionado. El contenido del **Acumulador (A)** se coloca en el byte de mayor peso mientras está activa la señal. También se activa cuando existe un reconocimiento de petición de interrupción e indica al periférico que puede colocar el vector de interrupción en el **Bus de Datos** (byte de menor peso de la dirección que contiene la rutina que da servicio al periférico.
$\overline{RD}$	Señal de Lectura	Triestado, activa en BAJO. Indica que está tomando lugar una operación de lectura en la memoria o en los **Puertos de Entrada/Salida**.
$\overline{WR}$	Señal de Escritura	Triestado, activa en BAJO. Indica que está tomando lugar una operación de escritura en la memoria o en los **Puertos de Entrada/Salida**.
$\overline{BUSAK}$	Señal de Reconocimiento de Bus	Activa en BAJO. Normalmente se usa en combinación con la señal **BUSREQ** e indica al dispositivo que le solicita, que los buses del **CPU** están en alta impedancia y que por tanto puede tomar el control de ellos.
$\overline{M1}$	Ciclo de Máquina 1	Señal activa en BAJO. Indica que el **CPU** está en la porción de obtención del código de operación del ciclo de instrucción/ejecución.
$\overline{RFSH}$	Señal de Refresco de Memoria Dinámica	Activa en BAJO. Indica que los siete bits de menor peso del **Bus de Direcciones** contienen una dirección de refresco para la memoria dinámica.

Tabla 2.3 SEÑALES DE SALIDA DEL MICROPROCESADOR Z80

2.3.2 Señales de Entrada

Las señales de entrada son generadas por dispositivo periféricos externos conectados al sistema mediante **Puertos de Entrada/Salida**, el propio microprocesador actúa dependiendo del nivel lógico de las señales que reciba en estos pines. Una descripción más exacta de tales señales, así como de la función que realizan se muestra en la tabla número 2.4.

Mnemónico	Descripción	Acción
CLK	Señal de Reloj	Proporciona el medio mediante el cual se reciben los pulsos de reloj necesarios para sincronizar los procesos de la CPU. El nivel de esos pulsos está acorde con los dispositivos TTL. La frecuencia de oscilación se encuentra entre 2 MHz. y 4 MHz.
$\overline{INT}$	Señal de Interrupción	Activa en BAJO. Indica al CPU que un dispositivo periférico ha originado una petición de interrupción. El CPU atiende dicha petición al final del ciclo de instrucción actual siempre y cuando los flip-flops de interrupción estén activados.
$\overline{NMI}$	Señal de Operación de Interrupción No Mascarable	Activa en BAJO. Indica que el CPU debe atender inmediatamente la petición de interrupción al final del ciclo de instrucción actual, sin tomar en cuenta el estado de los flip-flops de interrupción. Siempre brinca a la localidad de memoria **0066H** (donde se debe colocar la rutina de tratamiento de interrupción no mascarable).
$\overline{HALT}$	Señal de Paro	Activa en BAJO. Indica que una instrucción de paro está siendo ejecutada. Mientras se encuentre activa esta señal, la CPU ejecuta continuamente instrucciones **NOP**, estando en espera de una señal de interrupción.
$\overline{BUSRQ}$	Señal de Requerimiento de Bus	Activa en BAJO. Indica que un dispositivo periférico tomará el control del **Bus de Datos**, **Direcciones** y **Control**, por lo que la CPU debe mandarlos a alta impedancia. Esta señal tiene una prioridad más alta que **NMI**, por lo que siempre es atendida al final de la ejecución de la instrucción actual.
$\overline{RESET}$	Señal de Re-Inicio	Activa en BAJO. Habilita los flip-flops de interrupción, borra el registro **Contador de Programa** (carga el registro PC con 0000H), borra los registros I y R.
$\overline{WAIT}$	Señal de Espera	Activa en BAJO. Indica que la memoria direccionada o el dispositivo periférico conectado al sistema mediante los **Puertos de Entrada/Salida** no están listos para efectuar una transferencia de datos.

Tabla 2.4 SEÑALES DE ENTRADA DEL MICROPROCESADOR Z80

2.3.3 Señales de Entrada/Salida

Las señales de entrada/salida son un grupo especial de señales que manejan datos bidireccionales. Estas señales pueden ser canalizadas tanto por el microprocesador como por los dispositivos periféricos conectados al sistema mediante **Puertos de Entrada/Salida**. Una descripción más detallada de las señales de entrada/salida, así como de las funciones que realizan se detalla en la siguiente tabla:

Mnemónico	Descripción	Acción
D_0-D_7	Bus de Datos	Activo en ALTO, bidireccional. Contiene el código de datos y/o instrucciones que intercambia la CPU con la memoria y los dispositivos periféricos conectados al sistema mediante **Puertos de Entrada/Salida**.

Tabla 2.5 SEÑALES DE ENTRADA/SALIDA DEL MICROPROCESADOR Z80

2.4 Características Eléctricas del Microprocesador Z80

La tabla 2.6 muestra las características eléctricas y valores típicos del microprocesador Z80 Éstas deben de ser tomadas en cuenta para un correcto funcionamiento del sistema. Para una comprensión más clara de la tabla, tome en cuenta las siguientes notas:

a) El dato debe estar disponible para el **Bus de Datos** del Z80 cuando la señal $\overline{RD}$ se active. Durante un reconocimiento de interrupción, el dato debe estar disponible cuando se activen las señales $\overline{M1}$ e $\overline{IORQ}$.

b) Todas las señales de control son internamente sincronizadas, así que pueden ser totalmente asíncronas con respecto a los pulsos de reloj.

c) La señal de $\overline{RESET}$ debe estar activa por un periodo mínimo de 3 ciclos de reloj.

d) Retardo de salida vs Capacitancia de carga:
TA=70°C Vcc=+5V+/- 5%, añade 10 nanosegundos de retardo para cada 50 pf de incremento en la carga hasta un máximo de 200 pf para el **Bus de Datos** y 100 pf para el **Bus de Direcciones y Líneas de Control**.

e) Aún cuando es estático por diseño, tiene garantizado un $t_{W(\phi H)}$ de μ200 segundos máximo.

(1) $t_{acm}=t_{W(\phi H)}+t_f-75$

(2) $t_{aci}=t_c-80$

(3) $t_{ca}=t_{W(\phi L)}+t_f-40$

(4) $t_{caf}=t_{W(\phi L)}+t_f-60$

(5) $t_{dcm}=t_c-210$

(6) $t_{dci}=t_{W(\phi L)}+t_f-210$

(7) $t_{cdf}=t_{W(\phi L)}+t_f-80$

(8) $t_{W(MRL)}=t_c-40$

(9) $t_{W(MRH)}=t_{W(\phi H)}-t_f-30$

(10) $t_{W(WRL)}=t_c-40$

(11) $t_{mr}=2t_c+t_{W(\phi H)}+t_f-80$

(12) $t_c=t_{W(\phi H)}+t_{W(\phi L)}+t_r+t_f$

Señal	Simbolo	Parámetro	Min.	Max.	Unidad	Condición
ϕ	T_C	Periodo de los pulsos de reloj	.4	(12)	µsec	
	$T_w(\phi H)$	Ancho de los pulsos de reloj a un nivel lógico alto	180	(E)	nsec	
	$T_w(\phi L)$	Ancho de los pulsos de reloj a un nivel lógico bajo	180	200	nsec	
	T_{rf}	Flanco de subida y bajada del pulso de reloj		30	nsec	
A_0-A_{15}	$t_D(AD)$	Retardo de la salida de los pulsos de direcciones		145	nsec	C_L=50pF
	$t_F(AD)$	Retardo para el estado de alta impedancia		110	nsec	
	t_{acm}	Estabilidad de dirección antes de MREQ	(1)		nsec	
	t_{aci}	Estabilidad de dirección antes de IORQ, RD o WR	(2)		nsec	
	t_{ca}	Estabilidad de dirección de RD, WR, IORQ o MREQ	(3)		nsec	
	t_{caf}	Estabilidad de dirección de RD o WR en alta impedancia	(4)		nsec	
D_0-D_7	$t_D(D)$	Retardo de la salida de datos		230	nsec	C_L=50pF
	$t_f(D)$	Retardo de alta impedancia durante un ciclo de escritura		90	nsec	
	$t_{sa}(D)$	Activación de datos en flanco de subida en un ciclo M1	50		nsec	
	$t_{s\overline{a}}(D)$	Activ. de datos en flanco de bajada en un ciclo M2-M5	60		nsec	
	t_{dcm}	Estabilidad de datos antes de activarse WR	(5)		nsec	
	t_{dci}	Estabilidad de datos antes de activarse WR (ciclo de I/O)	(6)		nsec	
	t_{cdf}	Estabilidad de datos de WR	(7)			
	t_H	Tiempo detenido para una activación	0		nsec	
$\overline{MREQ}$	$t_{DL\phi(MR)}$	Retardo de MREQ (bajo) desde un flanco de bajada		100	nsec	C_L=50pF
	$t_{DH\phi(MR)}$	Retardo de MREQ (alto) desde un flanco de subida		100	nsec	
	$t_{D\overline{H}\phi(MR)}$	Retardo de MREQ (alto) desde un flanco de bajada		100	nsec	
	$t_{W(MRL)}$	Ancho del pulso (MREQ bajo)	(8)		nsec	
	$t_{W(MRH)}$	Ancho del pulso (MREQ alto)	(9)		nsec	
$\overline{IORQ}$	$t_{DL\phi(IR)}$	Ret. de IORQ desde el flanco de subida del reloj, IORQ bajo		100	nsec	C_L=50pF
	$t_{D\overline{L}\phi(IR)}$	Ret. de IORQ desde el flanco de bajada del reloj, IORQ bajo		130	nsec	
	$t_{DH\phi(IR)}$	Ret. de IORQ desde el flanco de subida del reloj, IORQ alto		100	nsec	
	$t_{D\overline{H}\phi(IR)}$	Ret. de IORQ desde el flanco de bajada del reloj, IORQ alto		110	nsec	
$\overline{RD}$	$t_{DL\phi(RD)}$	Retardo de RD (bajo) desde un flanco de subida		100	nsec	C_L=50pF
	$t_{D\overline{L}\phi(RD)}$	Retardo de RD (bajo) desde un flanco de bajada		130	nsec	
	$t_{DH\phi(RD)}$	Retardo de RD (alto) desde un flanco de subida		100	nsec	
	$t_{D\overline{H}\phi(RD)}$	Retardo de RD (alto) desde un flanco de bajada		110	nsec	
$\overline{WR}$	$t_{DL\phi(WR)}$	Retardo de WR (bajo) desde un flanco de subida		80	nsec	C_L=50pF
	$t_{D\overline{L}\phi(WR)}$	Retardo de WR (bajo) desde un flanco de bajada		90	nsec	
	$t_{DH\phi(WR)}$	Retardo de WR (alto) desde un flanco de bajada		100	nsec	
	$t_{W(WRL)}$	Ancho del pulso, WR bajo	(10)		nsec	
$\overline{M1}$	$t_{DL(M1)}$	Retardo de M1 desde el flanco de subida del reloj, M1 bajo		130	nsec	C_L=50pF
		Retardo de M1 desde el flanco de subida del reloj, M1 alto		130	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	Retardo de RFSH desde el flanco de subida del reloj, RFSH bajo		180	nsec	C_L=50pF
	$t_{DH(RF)}$	Retardo de RFSH desde el flanco de subida del reloj, RFSH alto		150	nsec	
$\overline{WAIT}$	$t_s(WT)$	Tiempo de activación de WAIT para un flanco de bajada del reloj	70		nsec	
HALT	$t_{D(HT)}$	Retardo de HALT desde un flanco de bajada del reloj		300	nsec	C_L=50pF
$\overline{INT}$	$t_s(IT)$	Tiempo de activación de INT para un flanco de bajada del reloj	80		nsec	
NMI	$t_w(NML)$	Ancho de pulso, MNI bajo	80		nsec	
BUSRQ	$t_s(BQ)$	BUSRQ	80		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	Retardo de BUSAK (bajo)desde un flanco de subida del reloj		120	nsec	C_L=50pF
	$t_{DH(BA)}$	Retardo de BUSAK (alto)desde un flanco de bajada del reloj		110	nsec	
$\overline{RESET}$	$t_s(RS)$	Tiempo de activación de RESET para flanco de subida de reloj	90		nsec	
	tF(C)	Tiempo para una alta impedancia (MREQ, IORQ, RD y WR)		100	nsec	
	t_{mr}	Estabilidad de M1 antes de IORQ (Reconocimiento de Int.)	(11)		nsec	

Tabla 2.6 CARACTERÍSTICAS ELÉCTRICAS DEL MICROPROCESADOR Z80

2.5 Temporización Básica de la CPU

El pulso de reloj suministrado al microprocesador Z80 produce un tren de pulsos llamados "**Ciclos-T**" (ciclos de reloj). Estos a su vez, se agrupan en periodos llamados "**Ciclos-M**" (ciclos máquina) para cada ciclo de instrucción, como se puede apreciar en la siguiente figura:

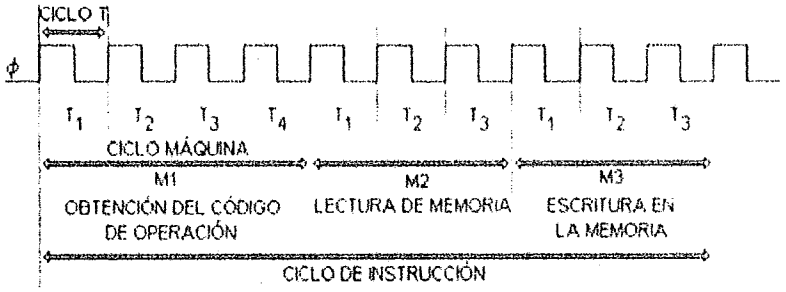


Fig. 2.3 DIAGRAMA DE TIEMPOS BÁSICO DE UN CICLO DE INSTRUCCIÓN DEL Z80

El ciclo de máquina M1 es el periodo de búsqueda de instrucción, éste puede tener de 4 a 6 ciclos de reloj. Durante los ciclos de máquina M2 y M3, se realiza una operación de lectura de memoria o entrada/salida de datos.

Durante el ciclo de búsqueda de la instrucción de código de operación, el registro **Contador de Programa (PC)** contiene la dirección de la siguiente instrucción. Los contenidos del registro **PC** son colocados en el **Bus de Direcciones** (A_0-A_{15}) durante la primera mitad del **Ciclo M1**, como se puede apreciar en la siguiente figura:

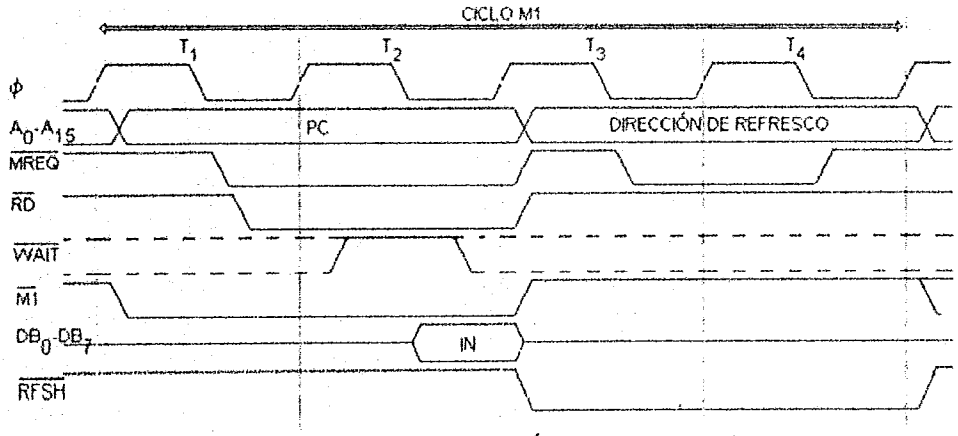


Fig. 2.4 DIAGRAMA DE TIEMPO DEL CICLO DE MÁQUINA M1 DEL MICROPROCESADOR Z80

Cuando se encuentra en un ciclo **Fetch**, esto es, en el estado de lectura de alguna localidad de memoria, las señales $\overline{MREQ}$ y $\overline{RD}$ son activadas a un nivel lógico bajo. Esto indica a la memoria que una operación de lectura está tomando lugar desde una localidad cuya dirección se encuentra en el **Bus de Direcciones**.

La línea WAIT se muestrea durante este periodo. Si el dispositivo de memoria es lento, puede generar una señal de espera para detener la operación. Si una señal de espera se encuentra durante este muestreo (esto es, durante cada ciclo T), entonces la **CPU** va a introducir otro estado de espera. Cuando el dispositivo está listo para transferir los datos, la señal de espera desaparece y el **Bus de Datos** obtiene el dato desde la localidad de memoria.

Durante la última parte del ciclo M1 (entre T3/T4), la dirección de refresco de memoria se coloca en los siete bits de menor peso del bus de direcciones y se genera una señal RFSH. Esto permite el refresco de las memorias dinámicas de estado sólido.

Durante el momento del ciclo M1 que los contenidos del registro **PC** están en el bus de direcciones, la señal M1 se encuentra activo en bajo.

El ciclo de máquina M1 va a ser tan largo tanto como una señal de espera esté presente. Usando la línea WAIT nos permite sincronizar la **CPU** y los dispositivos externos.

Los ciclos de máquina M2 y M3 son usados para leer o escribir en una localidad de memoria. La siguiente figura muestra esto último:

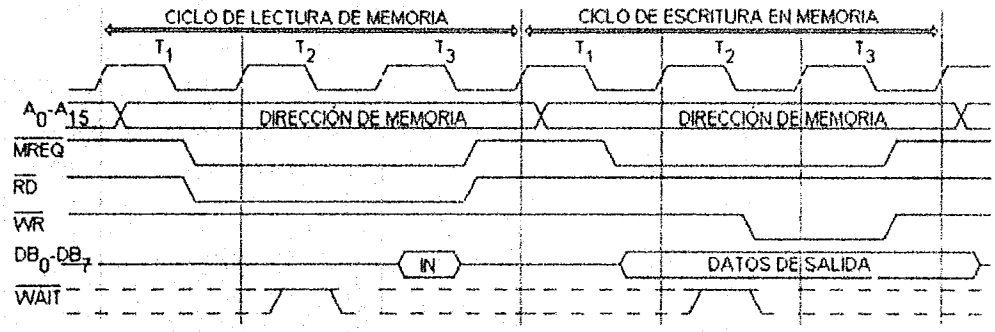


Fig. 2.5 DIAGRAMA DE TIEMPOS DE UN CICLO DE LECTURA/ESCRITURA DE MEMORIA DEL Z80

Las principales señales utilizadas en este proceso son MREQ, WR y RD. Si se requiere de una operación de lectura de memoria, entonces una dirección se coloca en el **Bus de Direcciones** (A_0-A_{15}) durante el ciclo de máquina M2. Durante este periodo, las líneas MREQ y RD se activan en bajo. La señal MREQ no se activa hasta que el dato en el **Bus de Direcciones** sea estable.

Las operaciones de escritura de memoria causan que el dato de la **CPU** sea escrito en localidades específicas de memoria. Esto ocurre durante el ciclo de máquina M3. En esta operación, las señales MREQ y WR se activan. La señal MREQ no se activa hasta que el dato presente en el **Bus de Datos** sea estable. Nuevamente, la dirección de la localidad especificada se coloca en el **Bus de Direcciones**.

Al igual que en un ciclo de búsqueda (Fetch), un estado de espera puede ser generado. Si la señal WAIT está en un nivel bajo, entonces la CPU continua introduciendo estados de espera hasta que la señal sea desactivada. La señal WAIT puede ser usada para sincronizar la CPU con la memoria.

La siguiente figura muestra el diagrama para un ciclo de lectura/escritura de puertos:

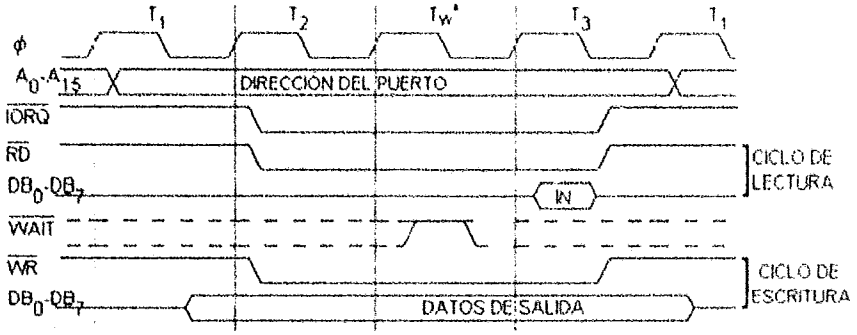


Fig. 2.6 DIAGRAMA DE TIEMPOS DE UN CICLO DE LECTURA/ESCRITURA DE PUERTO DEL Z80

Durante cada uno de estos tipos de operaciones, la señal IORQ debe de activarse a un nivel bajo. Si la operación es un ciclo de lectura, entonces la señal RD también debe de activarse. Si la operación es un ciclo de escritura, entonces la señal WR debe activarse junto con IORQ. En los ciclos de entrada/salida, la dirección del puerto se coloca en el byte de menor peso del bus de direcciones. Dado que la dirección del puerto consta de 8 bits, se pueden especificar hasta 256 direcciones o puertos diferentes (00H-FFH).

Durante una operación de entrada (lectura de puertos), las señales IORQ y RD se activan durante T2 y T3, y el dato del puerto de entrada se coloca en el **Bus de Datos**.

Durante una operación de salida (escritura en los puertos), las señales IORQ y WR se activan durante los mismos ciclos T1 y T3. El dato del acumulador se coloca en el **Bus de Datos** y también en el puerto cuya dirección está contenida en el byte de menor peso del **Bus de Direcciones**. Sin embargo, la señal IORQ no se activa inmediatamente, permitiendo que el dato presente en el **Bus de Datos** se estabilice antes de que la operación se lleve a cabo.

La señal BUSRQ se usa para que los dispositivos externos obtengan el control del **Bus de Direcciones, Datos y control** de la **CPU**. Esto permite accesos directos a la memoria, como se puede apreciar en la siguiente gráfica:

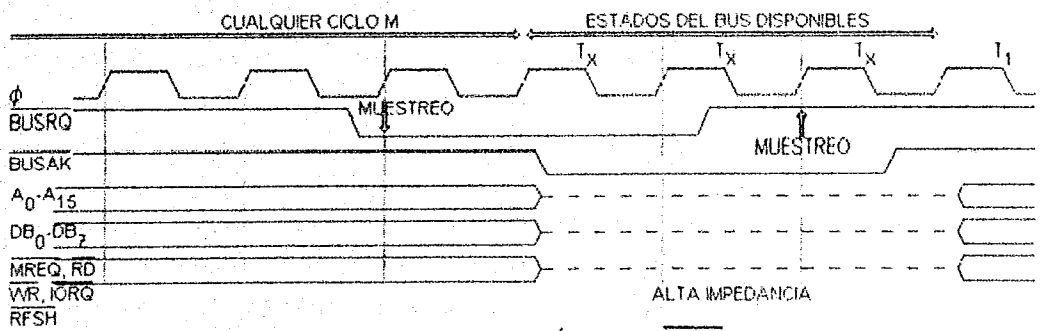


Fig. 2.7 DIAGRAMA DE TIEMPOS DE UN CICLO DE PETICIÓN DE BUS (BUSRQ) DEL MICROPROCESADOR Z80

La **CPU** muestrea la señal $\overline{BUSRQ}$ durante el último **Ciclo T** de cualquier **Ciclo M**. Si el requerimiento de bus está activo, la **CPU** completa la instrucción actual, y posteriormente atiende la petición. Después del último **Ciclo T** del último **Ciclo M**, la **CPU** se pone en un estado de alta impedancia. Las líneas del **Bus de Dirección**, **Bus de Datos** y las líneas de control $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ y $\overline{RFSH}$ se colocan en un estado de alta impedancia, desconectándolos del circuito externo. Esto permite al dispositivo externo tomar el control de las líneas, para introducir datos a localidades de memoria sin pasar por la **CPU**. Cuando las líneas de la **CPU** están en el estado de alta impedancia, ésta genera una señal $\overline{BUSAK}$ (reconocimiento de bus) indicándole al dispositivo que los buses están disponibles para su uso.

Cuando el dispositivo externo ha finalizado su intercambio de datos con la memoria, desactiva la señal $\overline{BUSRQ}$, indicándole a la **CPU** que tome el control nuevamente.

La habilidad de atender interrupciones permite a la **CPU** usar ciertos tipos de dispositivos externos en forma más eficiente. La **CPU** puede realizar otras acciones mientras el dispositivo más lento se encuentra procesando la información, o puede realizar otras acciones mientras espera que raras situaciones ocurran. La señal $\overline{INT}$ es el encargado de realizar una petición de interrupción. Esta línea se muestrea por la **CPU** durante el flanco de subida del último estado T de cada **Ciclo M**, como se puede apreciar en la siguiente figura:

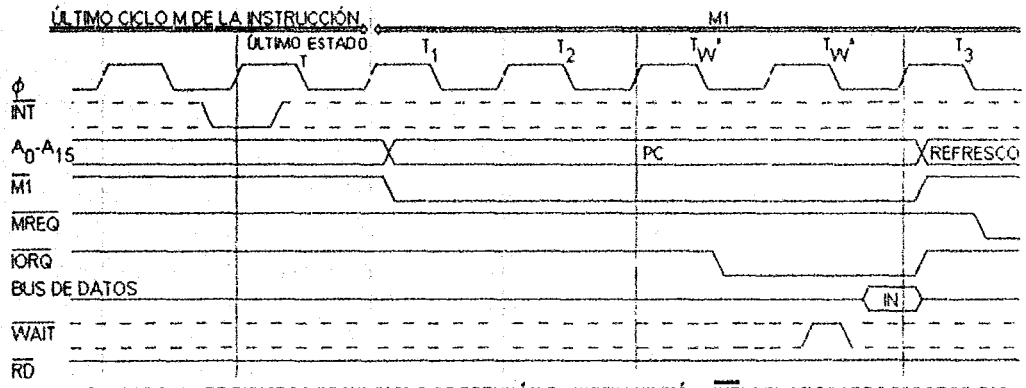


Fig. 2.8 DIAGRAMA DE TIEMPOS DE UN CICLO DE PETICIÓN DE INTERRUPCIÓN ($\overline{INT}$) DEL MICROPROCESADOR Z80

Estas interrupciones pueden ser enmascaradas por software dado que la **CPU** no acepta ninguna petición a menos que el flip-flop interno de la **CPU** esté activado. Este flip-flop de interrupción se controla mediante comandos de software. Las interrupciones son también ignoradas si la línea $\overline{BUSRQ}$ está activada a un nivel bajo.

Si la **CPU** acepta la petición de interrupción, entonces un estado especial M1 se genera, de esta forma la línea $\overline{M1}$ se activa en bajo. El **Bus de Direcciones** recibe los contenidos del registro **Contador de Programa (PC)**, así que la **CPU** puede regresar al programa original después de que la interrupción sea servida. La dirección de la siguiente instrucción, para ser ejecutada siguiente a la terminación de la interrupción, se almacena en una pila de memoria externa.

Una vez que los contenidos del registro **PC** son almacenados, la señal $\overline{WAIT}$ puede ser usada para extender los tiempos al introducir la **CPU** estados de espera. Si la linea $\overline{WAIT}$ se activa cuando es muestreada, la **CPU** introduce estados de espera. Si la señal está inactiva, entonces ningún estado de espera se genera y la **CPU** continua la ejecución del programa.

Ciertos tipos de situaciones de interrupción no pueden esperar a que el software comience la ejecución y la activación del flip-flop. Tales interrupciones podrían ser una condición de alarma en un proceso de manufactura o en un equipo médico. Estas situaciones requieren una interrupción no mascarable, tal como puede apreciarse en la siguiente figura:

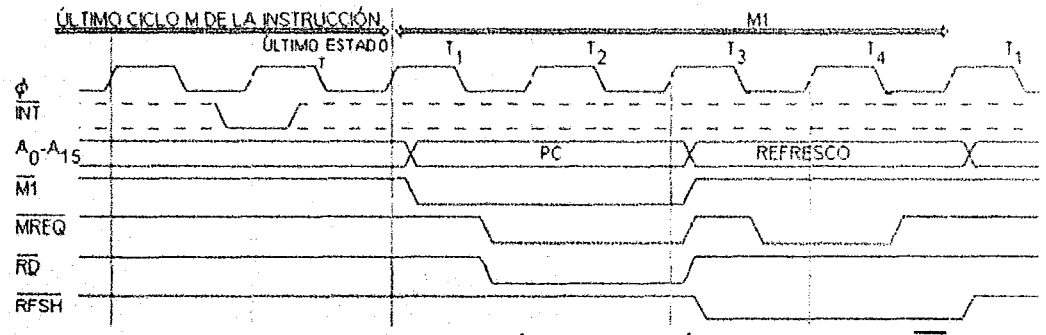


Fig. 2.9 DIAGRAMA DE TIEMPO DE UN CICLO DE PETICIÓN DE INTERRUPCIÓN NO ENMASCARABLE ($\overline{NMI}$) DEL Z80

Este tipo de ciclo de interrupción es muy similar a la interrupción regular, excepto que no depende del flip-flop de interrupción controlado por software. Este tipo de interrupción es atendido cuando el presente ciclo de instrucción sea completado. Los contenidos del contador de programa son almacenados en una pila de memoria externa y la **CPU** efectúa un brinco automático a la localidad de memoria **0066H** para buscar el programa que brinda servicios a la interrupción.

Si una instrucción de software **HALT** es ejecutada, la **CPU** va a continuar ejecutando instrucciones **No-Operación (NOP)** hasta que una de las dos situaciones siguientes ocurra:

1) Una interrupción no mascarable sea recibida.
2) Una interrupción mascarable sea recibida y el flip-flop interno de interrupciones del microprocesador Z80 está activo.

Si las líneas de interrupción ($\overline{INT}$ o $\overline{NMI}$) están activas cuando son muestreadas durante la porción T4 del ciclo M1, entonces la condición **HALT** es terminada después de T4. La linea $\overline{HALT}$ entonces se desactiva poniéndose en un estado alto. Lo anterior se muestra en la siguiente gráfica:

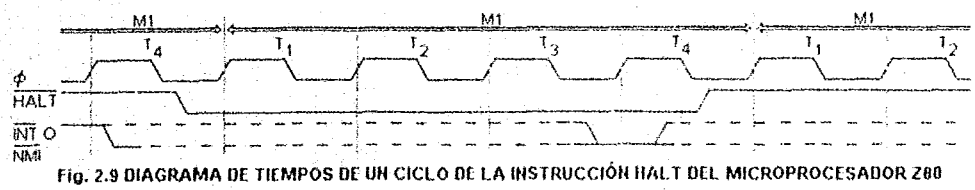


Fig. 2.9 DIAGRAMA DE TIEMPOS DE UN CICLO DE LA INSTRUCCIÓN HALT DEL MICROPROCESADOR Z80

2.6 Modos de Direccionamiento del microprocesador Z80

Una de las mayores ventajas del microprocesador Z80 con respecto a otros microprocesadores de 8 bits, es sin lugar a dudas su gran cantidad de modos de direccionamiento, gracias a éstos, el microprocesador puede extraer el controlador de proceso simple, organizar y crear una computadora real activa.

Los modos de direccionamiento del microprocesador Z80 son:

1) **Direccionamiento Inmediato**: En este tipo de direccionamiento, el operando sigue al código de operación en localidades secuenciales de memoria y el operando se carga en la localidad seleccionada inmediatamente. Un ejemplo de tal forma de direccionamiento puede estar dado por las instrucciones **ADD A,n** y **SUB A,n**. En estas instrucciones, el operando n se suma (o substrae) a los contenidos del **Acumulador**, y el resultado se almacena en el **Acumulador**. El formato para las instrucciones de direccionamiento inmediato es de la forma:

Byte 1	**op-code**	**código de operación**
Byte 2	**(n)**	**Dato**

La principal utilidad del modo de direccionamiento inmediato es la carga de datos en registros específicos o localidades, también son usados para realizar operaciones aritméticas usando constantes.

2) **Direccionamiento Inmediato Extendido**: Es análogo al modo de direccionamiento anterior, excepto que sólo puede manipular datos de 2 bytes, dado lo cual, requiere de tres bytes (el código de operación y dos bytes de datos siguiente). El formato de este tipo de instrucción es como se muestra a continuación:

Byte 1	**op-code**	**Código de Operación**
Byte 2	**(n-1)**	**Byte de Menor Peso**
Byte 3	**(n-2)**	**Byte de Mayor Peso**

Existen también instrucciones similares para los otros registros pares de 16 bits, tales como:

LD BC,nn; LD DE,nn y **LD SP,nn.**

3) **Direccionamiento de Página-Cero Modificado**: Este tipo de instrucción permite al programador efectuar una bifurcación a cualquier localidad de memoria en la página cero (primeras 256 direcciones de memoria, iniciando en la localidad **0000H**). Un ejemplo de este modo de direccionamiento está dado por la instrucción **RST p**, el cual dependiendo del operando p, va a resetear el contador de programa a cualquiera de las siguientes direcciones en la página cero de la memoria: **0000H, 0008H, 0010H, 0018H, 0020H, 0028H, 0030H** y **0038H**. En esta instrucción, los contenidos actuales del contador de programa se introducen en una pila de memoria externa. El byte de mayor peso del registro **Contador de Programa** se

carga con **00H**, mientras que el byte de menor peso se carga con un byte que selecciona cualquiera de las ocho localidades de memoria mencionadas. Por ejemplo, la carga del byte de menor peso del registro **Contador de Programa** con **CFH** puede ejecutar una instrucción **RST,08**.

El principal uso de este modo de direccionamiento es permitir el servicio de subrutinas con una instrucción de llamada de un solo byte, tal como se puede apreciar en la siguiente figura:

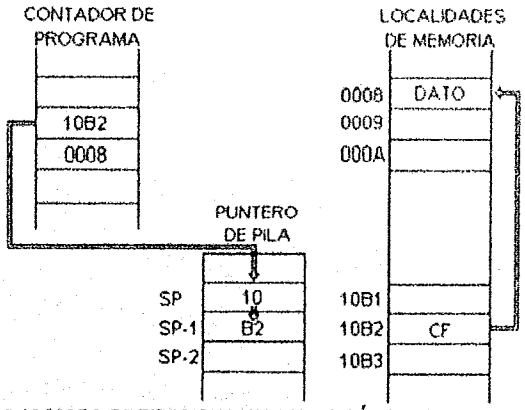


Fig. 2.10 MODO DE DIRECIONAMIENTO DE PÁGINA-CERO EXTENDIDA

La ejecución del programa comienza en la página cero, y en la localidad **10B2H** encuentra una instrucción **CF** durante el ciclo de búsqueda de instrucción. Esto ocasiona que el microprocesador Z80 bifurque a una subrutina en la localidad **0008H** (el registro **Contador de Programa** contiene **10B2H** al momento de ser encontrada la instrucción). El byte de mayor peso del registro **PC** se introduce en la pila en la localidad apuntada por **SP-1** y el byte de mayor peso en la localidad apuntada por **SP-2**. Cuando el programa retorna de la subrutina, el registro **Contador de Programa** recupera el dato de la pila y se incrementa a la siguiente instrucción localizada en la localidad de memoria **10B3H**.

4) **Direccionamiento Relativo**: Este modo de direccionamiento ejecuta instrucciones de bifurcación de dos bytes, los cuales permiten un desplazamiento en un valor de **e**. La bifurcación se efectúa a una localidad de memoria que se ubica desde -126_{10} hasta $+128_{10}$ localidades de memoria a partir de la dirección actual. Estas instrucciones requieren de dos bytes, así que el valor de **e** puede estar entre -128_{10} y $+127_{10}$. El valor del desplazamiento **e** siempre es un número con signo en complemento a dos, de esta forma puede tomar valores en binario desde **10000000B** hasta **01111111B** (desde **80H** hasta **7FH**). Dado que éste es una instrucción de dos bytes, la bifurcación no puede ocurrir hasta que la instrucción actual haya terminado, el registro **Contador de Programa** (PC) se incrementa en un par antes de que la bifurcación ocurra.

El formato para este tipo de instrucciones es como se muestra a continuación:

Byte 1	**op-code**	**Código de Operación**
Byte 2	**e(-128_{10} a +127_{10})**	**Desplazamiento Entero**

Un ejemplo de este tipo de direccionamiento puede ser el que se muestra en la siguiente figura:

LOCALIDAD	DATO	COMENTARIO
00 00		
00 01		
00 02		
00 03		
00 04		PC DESPUÉS DE LA EJECUCIÓN
00 05		
00 06		
00 07		
00 08		
00 09	18	CÓDIGO DE OPERACIÓN PARA JR, e
00 0A	FA	COMPLEMENTO A DOS PARA -6
00 0B		

Fig. 2.11 INSTRUCCIÓN DE MODO DIRECCIONAMIENTO RELATIVO

Las instrucciones ejecutan una bifurcación incondicional a una subrutina localizada en un desplazamiento **e** de un código de operación. El valor para el segundo byte va a ser **e-2**. El código de operación **18H** en la localidad **0009H** y el desplazamiento entero **FAH** (complemento a dos para –6) en la localidad **000AH**. Después de la ejecución de esta instrucción, el registro **Contador de Programa** va a contener la nueva dirección **0004H**.

5) **Direccionamiento Extendido**: En este modo de direccionamiento se puede usar dos enteros **nn** de 8 bits cada uno para crear una dirección de 16 bits. Existe un código de operación de uno o dos bytes seguidos por una dirección de dos bytes u operandos. En cualquier caso, el primer byte es el de menor peso y el segundo es el byte de mayor peso. El formato para esta instrucción es:

Byte 1	**op-code**	**Código de Operación**
Byte 2		**Posible Código de Operación Adicional)**
Byte 3	**n1**	**Primer operando (Byte de Menor Peso)**
Byte 4	**n2**	**Segundo operando (Byte de Mayor Peso)**

Un ejemplo de este tipo de direccionamiento está dado por la instrucción **LD A,(nn)**, el cual efectúa la carga del **Acumulador** con el byte ubicado en la localidad de memoria dada por el operando de dos bytes **nn**.

6) **Direccionamiento Indexado**: Este modo de direccionamiento utiliza los dos registros índice de 16 bits (**IX** e **IY**), más un valor de desplazamiento después del código de operación, usado para calcular la dirección real de bifurcación. En una instrucción de direccionamiento indexado típico, existe un código de operación de dos bytes seguidos por un desplazamiento entero **d**.

El formato para esta instrucción es el siguiente:

Byte 1	**(op-code)**	**Código de Operación**
Byte 2	**(op-code)**	**Código de Operación**
Byte 3	**(d)**	**Desplazamiento d**

Un ejemplo para este modo de direccionamiento es la instrucción **LD A, (IX+d)**, la cual carga el **Acumulador** con el dato existente en la localidad de memoria direccionada por los contenidos del registro índice **IX** y el desplazamiento entero **d**.

7) **Direccionamiento por Registro**: Este modo de direccionamiento permite transferir datos entre diferentes registros del microprocesador Z80. Tomando como ejemplo la instrucción **LD r,r'** Los registros **A, B, C, D, E, H y L** pueden ser usados como **r** o **r'**. El código de operación de un byte se forma usando el código de los registros en el código de operación, usando un formato como se muestra a continuación:

0 1|⇐r⇒|⇐r'⇒|

En el ejemplo anterior, los primeros dos bits corresponden a la instrucción, los siguientes tres bits corresponden al código binario del registro destino y finalmente los tres restantes bits corresponden al código binario del registro fuente.

8) **Direccionamiento por Registro Implicado**: En este modo de direccionamiento se usan **Registros Especiales** que siempre usan los mismos registros de la **CPU** para almacenar los operandos. Un ejemplo de este modo de direccionamiento es la instrucción **LD R,A**, el cual carga el registro **R (Refresco de Memoria)** con los contenidos del **Acumulador**.

9) **Direccionamiento por Registro Indirecto**: Este modo de direccionamiento permite las poderosas transferencias de datos entre la CPU y las localidades de memoria apuntadas por los contenidos de los registros pares de 16 bits. Un ejemplo de este modo de direccionamiento es la instrucción **LD (DE),A**, el cual carga la localidad de memoria apuntada por los contenidos del registro par **DE** con los contenidos del **Acumulador**.

10) **Direccionamiento por Bit**: Este modo de direccionamiento permite la manipulación individual de bits de cualquier registro, con acciones tales como activación, desactivación y verificación. El código de operación para este modo de direccionamiento esta dado por:

Byte 1	**(op-code)**	**Código de Operación**
Byte 2	**01 b r**	

En caso de una verificación de un bit de cualquier registro, el bit Z del registro F se activa si el bit verificado es 0 y se desactiva si es 1.

2.7 Interrupciones del microprocesador Z80

Una interrupción es un proceso que detiene la ejecución del programa principal de su computadora y comienza a ejecutar un programa diferente localizado en cualquier otra parte de la memoria, esto ocurre en respuesta a un estímulo externo. Tales estímulos pueden ser por ejemplo detectores de humo, detectores de nivel de líquido en sistemas de control u otros procesos. Cuando se recibe la petición de interrupción, la computadora ejecuta inmediatamente el programa que brinda servicios al dispositivo que realizó dicha petición.

Los ejemplos más comunes de dispositivos que manejan el intercambio de información con el microprocesador son aquellos cuyos datos no están siempre presentes, o los cuales se actualizan durante ciertos intervalos de tiempo, tales como un reloj de tiempo real o un temporizador. Ejemplos adicionales pueden ser los dispositivos DAC, teclados, impresoras, etc.

Antes de emitir una señal de reconocimiento de interrupción, la CPU finaliza la ejecución de la instrucción actual (además de efectuar un retardo entre la petición de interrupción y el momento en que la **CPU** se encuentra lista para procesar la petición). Para indicar al dispositivo periférico que haya generado la interrupción el momento en que la CPU se encuentra lista para atenderlo, el microprocesador Z80 muestrea la línea $\overline{INT}$ en el flanco de subida del último pulso de reloj de la instrucción actual. Si la línea $\overline{INT}$ se encuentra activa a un nivel bajo, la CPU responde generando una señal $\overline{IORQ}$ en el siguiente ciclo de máquina M1. De esta forma, se puede utilizar la combinación simultánea de las señales $\overline{IORQ}$ y $\overline{M1}$ para formar una señal de reconocimiento de interrupción.

La siguiente figura muestra en forma gráfica una forma de realizar un dispositivo de reconocimiento de interrupción:

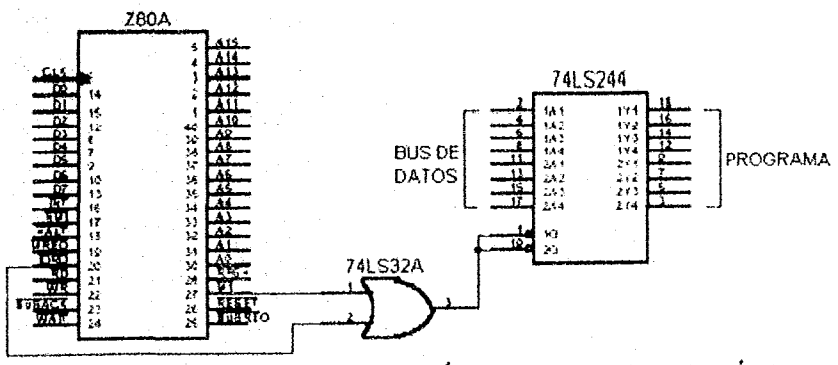


Fig. 2.12 RECONOCIMIENTO DE INTERRUPCIÓN PARA UN DISPOSITIVO PERIFÉRICO

El microprocesador Z80 cuenta con dos grupos de interrupciones

a) **Mascarable**: Este tipo de interrupción es controlado mediante software usando las instrucciones **EI, DI, IM0, IM1 e IM2**. Aún cuando este tipo de interrupciones se inicia enviando a un nivel bajo la línea $\overline{INT}$ del microprocesador Z80, esto no es suficiente, ya que requiere de una activación anticipada del flip-flop **IFF1**, dado que si se encuentra inactiva, la señal $\overline{INT}$ es mascarada e ignorada por la CPU. Este flip-flop se activa con las instrucciones **IM0, IM1, IM2** o **EI**. Si se desea desactivarlos se puede utilizar el pulso de $\overline{RESET}$ o ejecutar una instrucción **DI**.

Las instrucciones mascarables se dividen en:

1) **Modo 0**: Este modo tiene un funcionamiento análogo a la interrupción del microprocesador **8080A** de Intel, sin embargo existen diferencias importantes entre ellos en lo que respecta a la temporización. Este modo se selecciona cuando se aplica un pulso de $\overline{RESET}$ o cuando se ejecuta la instrucción **IM0**.

El **Modo de Interrupción 0** requiere que el dispositivo que solicitó la interrupción coloque una instrucción en el **Bus de Datos** del microprocesador Z80 al momento en que sea generada la señal de reconocimiento de interrupción, la cual en la mayoría de los casos es una instrucción **RESTART** de un byte. Estas instrucciones efectúan un salto incondicional a la localidad de memoria ubicada en la página 0 como se muestra en la siguiente tabla:

Mnem.	Dirección	Cod. Binario $B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$	Cod. Hex.
RST 00	00 00H	1 1 0 0 0 1 1 1	C7
RST 08	00 08H	1 1 0 0 1 1 1 1	CF
RST 10	00 10H	1 1 0 1 0 1 1 1	D7
RST 18	00 18H	1 1 0 1 1 1 1 1	DF
RST 20	00 20H	1 1 1 0 0 1 1 1	E7
RST 28	00 28H	1 1 1 0 1 1 1 1	EF
RST 30	00 30H	1 1 1 1 0 1 1 1	F7
RST 38	00 38H	1 1 1 1 1 1 1 1	FF

Tabla 2.7 INSTRUCCIONES RESTART DELMICROPROCESADOR Z80

La rutina que brinda servicios a la interrupción debe estar localizada en la memoria donde la instrucción **RESTART** transfiere el control.

La figura número 2.13 muestra un ejemplo de la forma de operación de una **Interrupción de Modo 0**.

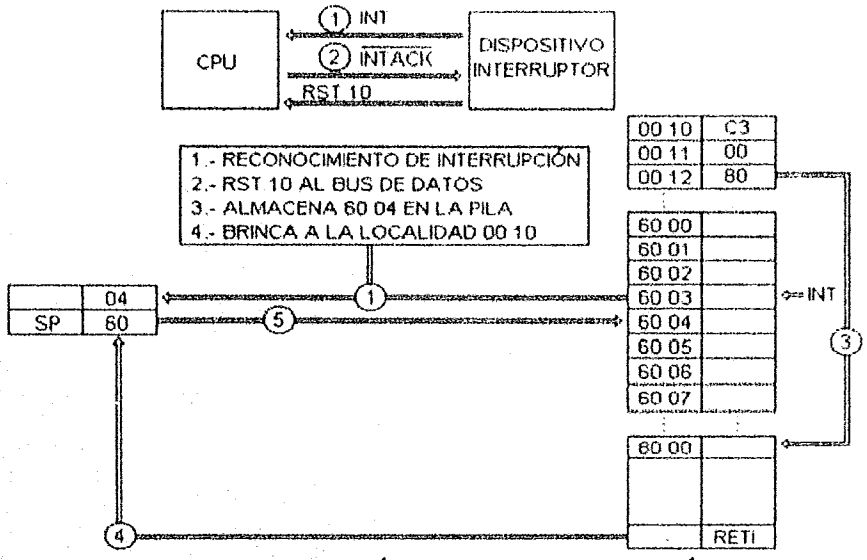


Fig. 2.13 FORMA DE OPERACIÓN DEL MODO DE INTERRUPCIÓN 0 DEL Z80

Esta forma de interrupción opera como sigue:

a) La señal $\overline{INT}$ ocurre durante la ejecución de la instrucción localizada en la dirección **6003H**. Ésta es reconocida por el registro **CPU** durante el último ciclo de reloj de esta instrucción.

b) En el siguiente pulso, la **CPU** reconoce la interrupción activando a un nivel bajo las señales $\overline{IORQ}$ y $\overline{M1}$.

c) Cuando la señal de reconocimiento de interrupción es recibida, el dispositivo periférico que generó la petición coloca un código **RST 10** en el bus de datos de la CPU.

d) La **CPU** ejecuta la instrucción **RST 10** incrementando el contenido del registro **PC** a **6004H**, almacenando su contenido en la pila de memoria externa y finalmente efectuando un brinco incondicional a la localidad de memoria **0010H**.

e) En la localidad de memoria **0010H**, la instrucción es un brinco incondicional a la localidad **8000H**, donde se encuentra almacenado el programa de servicio de interrupción.

f) Se almacenan las variables de memoria mediante las instrucciones **EX**, **EXX** o **PUSH**. Si se decide por éste último método, se debe ejecutar en secuencia las siguientes instrucciones:

```
80 00   PUSH AF
80 01   PUSH BC
80 02   PUSH DE
80 03   PUSH HL
80 04   PUSH IY
80 05   PUSH IX
```

Posterior a estas últimos, deben seguir las instrucciones propias de la rutina. Cuando haya terminado la ejecución del programa, se debe ejecutar instrucciones **POP** para restaurar las variables de ambiente desde la pila y finalmente ejecutar una instrucción **RETI**:

```
80 xx  POP IX
80 xx  POP IY
80 xx  POP HL
80 xx  POP DE
80 xx  POP BC
80 xx  POP AF
80 xx  RETI
```

g) Después de la instrucción **RETI**, la **CPU** reemplaza los contenidos del registro **PC** con el dato almacenado en la pila (**6004H**). Esta dirección es la siguiente en secuencia en el programa principal como si ninguna interrupción hubiese ocurrido. La ejecución normal del programa se ubica en la dirección **6004H**.

2) **Modo 1**: Este modo de interrupción es análogo a la interrupción no mascarable, excepto en que es mascarable y efectúa un salto incondicional a la localidad de memoria **0038H** en lugar de la localidad **0066H**.

Este modo de interrupción requiere la activación del flip-flop **IFF1** mediante la ejecución de las instrucciones **EI** e **IM1**. Este modo no requiere de alguna lógica adicional para efectuar la instrucción **RESTART**, por lo cual éste es más rápido que el modo 0.

La siguiente figura muestra un ejemplo del modo de interrupción 1:

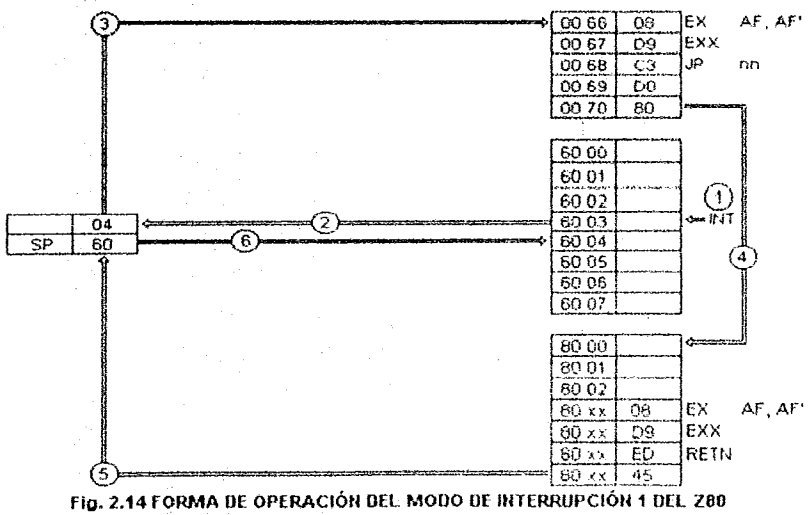


Fig. 2.14 FORMA DE OPERACIÓN DEL MODO DE INTERRUPCIÓN 1 DEL Z80

El Modo de Interrupción 1 mostrado en la figura anterior tiene una teoria de funcionamiento como sigue:

a) La señal $\overline{INT}$ se genera mientras se está ejecutando la instrucción localizada en la dirección **6003H**.

b) El registro **Contador de Programa** (PC) se incrementa de **6003H** a **6004H** y su contenido se almacena en la pila de memoria externa.

c) El registro **PC** se carga con **0038H**, transfiriendo el control del programa a esta dirección. Antes de comenzar a ejecutar las instrucciones propias del programa que brinda servicios a la interrupción almacena las variables de ambiente mediante la ejecución de las instrucciones **EX** y **EXX**.

d) Efectúa un brinco incondicional a la localidad de memoria donde se ubican las instrucciones del programa que brinda servicios a la interrupción. Al final de la instrucción se ejecuta una instrucción **RETN**, el cual indica al registro **CPU** retornar el control al programa principal.

e) La ejecución de la instrucción **RETN** restaura los contenidos del registro **PC** desde la pila de memoria externa. Al momento de que el registro **PC** contiene la dirección **6004H**, el programa efectúa un brinco incondicional a esta localidad de memoria y ejecuta las instrucciones ubicadas a partir de esta dirección. Esta dirección de memoria es la siguiente en secuencia como si no se hubiera efectuado alguna interrupción. Debe tomarse en cuenta que antes de ejecutar la instrucción **RETN** se deben restaurar las variables de ambiente ejecutando nuevamente las instrucciones **EX** y **EXX**.

3) **Modo 2**: Este modo de interrupción es uno de los más potentes del microprocesador Z80, ya que permite hasta 128 niveles de interrupción vectorizadas, como contraposición a los únicamente ocho niveles del **Modo de Interrupción 0** y sólo un nivel de los **Modos de Interrupción 1 y no mascarable**.

Este vector consiste de una palabra simple de un byte que apunta a cualquier localidad de memoria. La dirección de 16 bits del programa que brinda los servicios de interrupción se almacenan en una tabla de direcciones de interrupción localizadas en alguna parte de la memoria. La ubicación de esta tabla está apuntada por una palabra de dos bytes formadas por los contenidos del **Registro de Interrupción** (I) y de una palabra de un byte suministrado por el dispositivo que generó la interrupción. El byte de mayor peso de este puntero de 16 bits es suministrado por el registro I, el cual debe ser precargado por programa. El byte de menor peso es suministrado por el dispositivo que generó la interrupción.

Sin embargo, existe una restricción al respecto: el contenido de la tabla debe comenzar en una localidad de memoria par. El primer byte corresponde al byte de menor peso de la dirección y el segundo byte corresponde al byte de mayor peso de la dirección. De esta

forma, el bit menos significativo del byte suministrado por el dispositivo interruptor debe ser 0, tal como se puede ver en la siguiente gráfica:

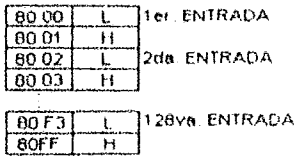


Fig. 2.15 TABLA DE VECTORES

En la gráfica anterior, se elige la página 8 para ubicar la tabla, comenzando con **8000H** La primer entrada se ubica en **8000H** y **8001H**, en estas localidades se almacenan los bytes de menor peso y de mayor peso de la dirección donde se ubica el programa que brinda servicios de interrupción. La primera parte de esta dirección (**80H**) se almacena en el registro I, la segunda parte es suministrada por el dispositivo periférico.

La figura número 2.16 muestra la forma de operación del **Modo de Interrupción 2**. En esta figura, el programa principal se ubica en la página 4, la tabla de vectores en la página 8 y la subrutina de interrupción para el dispositivo en la página 6. El registro I contiene **80H**, el dispositivo interruptor está configurado para introducir **04H** en el bus de datos cuando la señal de reconocimiento de interrupción sea recibida:

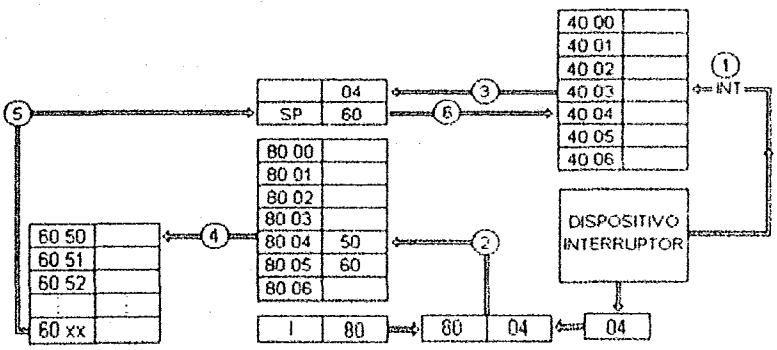


Fig. 2.16 FORMA DE OPERACIÓN DEL MODO DE INTERRUPCIÓN 2 DEL Z80

El modo de interrupción 2 opera de la siguiente forma

1) El dispositivo periférico envia una señal de interrupción ($\overline{INT}$) a la CPU.
2) Cuando la señal de reconocimiento de interrupción es recibida, el periférico coloca **04H** en el bus de datos. Esta se combina con **80H** desde el registro I para formar **8004H**. Esta localidad de memoria contiene la dirección del programa que brinda servicios a la interrupción requerida por el periférico.
3) El registro **PC** se incrementa y su contenido se almacena en la pila de memoria externa.
4) El registro **PC** se carga con la dirección encontrada en la dirección **8004H (6005H)**, ocasionando que se efectúe un brinco incondicional a esta localidad de memoria.

5) Después de la ultima instrucción (**RETI**) del programa, los datos del registro **PC** almacenada en la pila son restauradas.
6) El programa principal efectúa un brinco a la localidad de memoria **4004H**

Debe tomarse siempre en cuenta que deben almacenarse las variables de ambiente al iniciar la rutina de servicio de interrupción y restaurarlas antes de efectuar la instrucción **RETI** final.

b) **No Mascarable**: La interrupción se ejecuta en secuencia sin tomar en cuenta cualquier otra consideración. La línea $\overline{NMI}$ se encuentra a disposición de este tipo de interrupciones, ésta es muestreada por el microprocesador durante el último pulso de reloj de cada ciclo máquina. Si esta línea se encuentra activa en bajo, la **CPU** comienza la secuencia de interrupción en el siguiente pulso de reloj. La interrupción no mascarable es semejante a un reinicio de hardware o una instrucción **RST 66H** (efectúa un salto incondicional a la dirección **0066H** y ejecuta las instrucciones que se encuentren a partir de esta localidad de memoria), además de que se encuentra implementada por hardware dentro del microprocesador. Para ejemplificar este tipo de interrupción, refiérase al modo de interrupción 1, tomando en cuenta las siguientes dos consideraciones:

1. La línea que se utiliza para aceptar la interrupción es $\overline{NMI}$.
2. El brinco incondicional lo efectúa a la localidad de memoria **0066H**.

2.8 Conjunto de Instrucciones del Microprocesador Z80

El microprocesador Z80 cuenta con un conjunto total de 158 instrucciones, sin embargo cuando son combinadas en sus diversas formas se pueden contar más de 400 instrucciones. Cuando se agrupan, éstas pueden ser clasificadas en:

1) **Instrucciones de Carga e Intercambio**: Las instrucciones de carga son usadas para transferir datos entre una localidad de memoria a otra. Más específicamente, existen dos tipos básicos de instrucciones de carga: a) transferencias de datos entre registros internos del microprocesador y b) transferencias de datos entre registros internos y localidades externas de memoria. Las instrucciones de carga pueden ser de 8 ó de 16 bits, las cuales manipulan bytes simples de datos y usan registros o localidades de memoria de 8 bits. Las instrucciones de carga de ocho bits se enlistan a continuación:

Mnem.	Cód. de Operación $B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$	Ciclos	Flags Afectados	Descripción
LD r, r'	0 1 r r r r r' r'	1	Ninguno	Almacena el dato de un registro r' en un registro r
LD r, n	0 0 r r r 1 1 0 n n n n n n n n	2	Ninguno	Almacena el dato n en el registro r

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
LD R, (HL)	0 1 r r r 1 1 0	2	Ninguno	Almacena el dato contenido en la localidad de memoria direccionada por el registro par HL en el registro r
LD R, (IX+d)	1 1 0 1 1 1 0 1 0 1 r r r 1 1 0 d d d d d d d d	5	Ninguno	Almacena el dato contenido en la localidad de memoria direccionada por la suma del registro índice IX y el byte d al registro r
LD R,(IY+d)	1 1 1 1 1 1 0 1 0 1 r r r 1 1 0 d d d d d d d d	5	Ninguno	Almacena el dato contenido en la localidad de memoria direccionada por la suma del registro índice IY y el byte d al registro r
LD (HL), r	0 1 1 1 0 r r r	2	Ninguno	Almacena el dato del registro r en la localidad de memoria direccionada por el registro par HL
LD (IX+d), r	1 1 0 1 1 1 0 1 0 1 1 1 0 r r r d d d d d d d d	5	Ninguno	Almacena el contenido del registro r en la localidad de memoria direccionada por la suma del registro índice IX y el byte d
LD (IY+d), r	1 1 1 1 1 1 0 1 0 1 1 1 0 r r r d d d d d d d d	5	Ninguno	Almacena el contenido del registro r en la localidad de memoria direccionada por la suma del registro índice IY y el byte d
LD (HL), n	0 0 1 1 0 1 1 0 n n n n n n n n	3	Ninguno	Almacena el dato n en la localidad de memoria direccionada por el registro par HL
LD (IX+d), n	1 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 d d d d d d d d n n n n n n n n	5	Ninguno	Almacena el dato n en la localidad de memoria direccionada por la suma del registro índice IX y el byte d
LD (IY+d), n	1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 0 d d d d d d d d n n n n n n n n	5	Ninguno	Almacena el dato n en la localidad de memoria direccionada por la suma del registro índice IY y el byte d
LD A, (BC)	0 0 0 0 1 0 1 0	2	Ninguno	Almacena el dato contenido en la localidad de memoria direccionada por el registro par BC en el **Acumulador**
LD A, (DE)	0 0 0 1 1 0 1 0	2	Ninguno	Almacena el dato contenido en la localidad de memoria direccionada por el registro par DE en el **Acumulador**
LD A, (nn)	0 0 1 1 1 0 1 0 n n n n n n n n n n n n n n n n	2	Ninguno	Almacena el dato contenido en la localidad de memoria direccionada por el operando par nn en el **Acumulador**
LD (BC), A	0 0 0 0 0 0 1 0	2	Ninguno	Almacena el dato del **Acumulador** en la localidad de memoria direccionada por el registro par BC
LD (DE), A	0 0 0 1 0 0 1 0	2	Ninguno	Almacena el dato del **Acumulador** en la localidad de memoria direccionada por el registro par DE

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
LD (nn), A	0 0 1 1 0 0 1 0 n n n n n n n n n n n n n n n n	4	Ninguno	Almacena el contenido del **Acumulador** en la localidad de memoria direccionada por el operando par nn
LD A, I	1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 1	2	S: Si I<0 S=1 Si I>=0 S=0 Z: Si I=0 Z=0 Si I<>0 Z=1 H: H=0 N: N=0 P/V: P/V=IFF2	Almacena el contenido del registro I en el **Acumulador**
LD A, R	1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 1	2	S: Si R<0 S=1 Si R>=0 S=0 Z: Si R=0 Z=0 Si R<>0 Z=1 H: H=0 N: N=0 P/V: P/V=IFF2	Almacena el contenido del registro R en el **Acumulador**
LD I, A	1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 1	2	Ninguno	Almacena el contenido del **Acumulador** en el registro I
LD R, A	1 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1	2	Ninguno	Almacena el contenido del **Acumulador** en el registro R

Las instrucciones de carga de 16 bits usan los registros pares **AF**, **BC**, **DE**, **HL**, **SP**, **IX** e **IY**. También usan direcciones de memoria de 16 bits para especificar las localidades de dos bytes de datos que van a ser enviados u obtenidos de la CPU. En todas las instrucciones de carga debe especificarse un origen y un destino de los datos, los cuales pueden ser registros internos o localidades de memoria. El grupo de instrucciones de carga de 16 bis se muestra a continuación:

Mnem.	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
LD dd, nn	0 0 d d 0 0 0 1 n n n n n n n n n n n n n n n n	3	Ninguno	Almacena el dato de dos bytes nn en el registro par **dd** (BC, DE, HL o SP)
LD IX, nn	1 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 n n n n n n n n n n n n n n n n	4	Ninguno	Almacena el dato de dos bytes nn en el registro indice IX
LD IY, nn	1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 1 n n n n n n n n n n n n n n n n	6	Ninguno	Almacena el dato de dos bytes nn en el registro indice IY
LD HL, (nn)	0 0 1 0 1 0 1 0 n n n n n n n n n n n n n n n n	5	Ninguno	Almacena el contenido de la localidad de memoria direccionada por el operando par nn en el registro L y el contenido de **nn+1** en el registro H
LD dd, (nn)	1 1 1 0 1 1 0 1 0 1 d d 1 0 1 1 n n n n n n n n n n n n n n n n	6	Ninguno	Almacena el contenido de la localidad de memoria nn en el byte de menor peso del registro **dd** y el contenido de **nn+1** al byte de mayor peso del registro **dd**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
LD IX, (nn)	1 1 0 1 1 1 0 1 0 0 1 0 1 0 1 0 n n n n n n n n n n n n n n n n	6	Ninguno	Almacena el contenido de la localidad de memoria direccionada por el operando par **nn** al byte de menor peso del registro IX y el contenido de **nn+1** al byte de mayor peso de IX
LD IY, (nn)	1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 0 n n n n n n n n n n n n n n n n	6	Ninguno	Almacena el contenido de la localidad de memoria direccionada por el operando par **nn** al byte de menor peso del registro IY y el contenido de **nn+1** al byte de mayor peso de IY
LD (nn), HL	0 0 1 0 0 0 1 0 n n n n n n n n n n n n n n n n	5	Ninguno	Almacena los contenidos del registro **L** a la localidad de memoria direccionada por el operando par **nn** y **H** en **nn+1**
LD (nn), dd	1 1 1 0 1 1 0 1 0 1 d d 0 0 1 1 n n n n n n n n n n n n n n n n	6	Ninguno	Almacena el contenido del byte de menor peso del registro **dd** en la localidad de memoria direccionada por el operando par **nn** y el de mayor peso en **nn+1**
LD (nn), IX	1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0 n n n n n n n n n n n n n n n n	6	Ninguno	Almacena el contenido de menor peso del registro índice IX en la localidad de memoria direccionada por el registro par **nn** y el de mayor peso en **nn+1**
LD (nn), IY	1 1 1 1 1 1 0 1 0 0 1 0 0 0 1 0 n n n n n n n n n n n n n n n n	6	Ninguno	Almacena el contenido de menor peso del registro índice IY en la localidad de memoria direccionada por el operando par **nn** y el de mayor peso en **nn+1**
LD SP, HL	1 1 1 1 1 0 0 1	1	Ninguno	Almacena el contenido del registro par HL en el registro par SP
LD SP, IX	1 1 0 1 1 1 0 1 1 1 1 1 1 0 0 1	2	Ninguno	Almacena el contenido del registro índice IX en el registro par SP
LD SP, IY	1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1	2	Ninguno	Almacena el contenido del registro índice IY en el registro par SP
PUSH qq	1 1 q q 0 1 0 1	3	Ninguno	Almacena el contenido de mayor peso del registro par **qq** en la parte superior de la pila, decrementa SP y almacena el byte de menor peso en la nueva dirección de SP (pila LIFO)
PUSH IX	1 1 0 1 1 1 0 1 1 1 1 0 0 1 0 1	3	Ninguno	Almacena el contenido de mayor peso del registro índice IX en la parte superior de la pila decrementa SP y almacena el byte de menor peso en la nueva dirección de SP (pila LIFO)
PUSH IY	1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1	4	Ninguno	Almacena el contenido de mayor peso del registro índice IY en la parte superior de la pila, decrementa SP y almacena el byte de menor peso en la nueva dirección de SP (pila LIFO)

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
POP qq	1 1 q q 0 0 0 1	3	Ninguno	Extrae el contenido de menor peso del registro par qq de la dirección actual de la pila direccionada por SP, incrementa SP y extrae el byte de mayor peso de la nueva dirección apuntada por SP Vuelve a incrementar SP
POP IX	1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 1	4	Ninguno	Extrae el contenido de menor peso del registro índice IX de la dirección actual de la pila direccionada por SP, incrementa SP y extrae el byte de mayor peso de la nueva dirección apuntada por SP Vuelve a incrementar SP
POP IY	1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1	4	Ninguno	Extrae el contenido de menor peso del registro índice IY de la dirección actual de la pila direccionada por SP, incrementa SP y extrae el byte de mayor peso de la nueva dirección apuntada por SP Vuelve a incrementar SP

Las instrucciones de intercambio son usados para intercambiar los contenidos de cualquier par de registros. Las instrucciones mencionadas son:

Mnem.	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
EX DE, HL	1 1 1 0 1 0 1 1	1	Ninguno	Intercambia los datos de dos bytes del registro par HL con el registro par DE
EX AF, AF'	0 0 0 0 1 0 0 0	1	Ninguno	Intercambia los datos de dos bytes del registro par AF' con el registro par AF
EXX	1 1 0 1 1 0 0 1	1	Ninguno	Intercambia los datos de dos bytes de los registros BC, DE y HL con los registros BC', DE' y HL'
EX (SP), HL	1 1 1 0 0 0 1 1	5	Ninguno	Intercambia el byte de menor peso del registro par HL con la localidad de memoria direccionada por el registro par SP y el byte de mayor peso por SP+1
EX (SP), IX	1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1	6	Ninguno	Intercambia el byte de menor peso del registro índice IX con la localidad de memoria direccionada por el registro par SP y el byte de mayor peso por SP+1
EX (SP), IY	1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 1	6	Ninguno	Intercambia el byte de menor peso del registro índice IY con la localidad de memoria direccionada por el registro par SP y el byte de mayor peso por SP+1

2) **Instrucciones de Búsqueda y Transferencia por Bloque**. Las instrucciones de bloques usan tres de los registros pares en su ejecución como se muestra a continuación:

HL **Dirección de la localidad fuente**
DE **Dirección de la localidad destino**
BC **Byte usado como contador**

En cualquier programa que usa estas instrucciones son necesarios estos registros para inicializar los valores requeridos. Cuando se efectúa las instrucciones por bloques, estos registros son automáticamente incrementados para apuntar a la siguiente localidad de memoria. Las instrucciones que pertenecen a este grupo son:

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
LDI	1 1 1 0 1 1 0 1 1 0 1 0 0 0 0 0	4	H H=0 N N=0 P/V Si BC-1=0 P/V=1 Si BC-1<>0 P/V=0	Almacena un byte de datos de la localidad de memoria direccionada por el registro par **HL** en la localidad de memoria direccionada por **DE**, incrementa **HL**, **DE** y decrementa el registro par **BC** (contador)
LDIR	1 1 1 0 1 1 0 1 1 0 1 1 0 0 0 0	BC<>0: 5 BC=0: 4	H H=0 N N=0 P/V P/V=0	Almacena un byte de datos de la localidad de memoria direccionada por el registro par **HL** en la localidad de memoria direccionada por **DE**, incrementa **HL**, **DE** y decrementa el registro par **BC** (contador) Si después del decremento de **BC** éste es 0 se termina la ejecución, si es diferente de 0 se repite la instrucción decrementando a su vez el registro PC
LDD	1 1 1 0 1 1 0 1 1 0 1 0 1 0 0 0	4	H H=0 N N=0 P/V Si BC-<>0 P/V=1 Si BC-1=0 P/V=0	Almacena un byte de datos de la localidad de memoria direccionada por el registro par **HL** en la localidad de memoria direccionada por **DE**, incrementa **HL**, **DE** y decrementa el registro par **BC** (contador)
LDDR	1 1 1 0 1 1 0 1 1 0 1 1 1 0 0 0	BC<>0: 5 BC=0: 4	H H=0 N N=0 P/V P/V=0	Almacena un byte de datos de la localidad de memoria direccionada por el registro par **HL** en la localidad de memoria direccionada por **DE**, incrementa **HL**, **DE** y decrementa el registro par **BC** (contador) Si después del decremento de **BC** éste es 0 se termina la ejecución, si es diferente de 0 se repite la instrucción decrementando a su vez el registro PC

Mnem.	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
CPI	1 1 1 0 1 1 0 1 1 0 1 0 0 0 0 1	4	S Si Resultado<0 S=1 Si Resultado>=0 S=0 Z Si A=(HL) Z=1 Si A<>(HL) Z=0 H Sin Carry B_4 H=1 Con Carry B_4 H=0 N N=1 P/V Si BC-1<>0 P/V=1 Si BC-1=0 P/V=0	Compara el byte de datos de la localidad de memoria direccionada por el registro par HL con el acumulador, si son iguales se activa el bit respectivo del **Flag** Posteriormente se incrementa HL y se decrementa BC (contador)
CPIR	1 1 1 0 1 1 0 1 1 0 1 1 0 0 0 1	Si BC<>0 y A<>(HL) 5 Si BC=0 ó A=(HL) 4	S Si Resultado<0 S=1 Si Resultado>=0 S=0 Z Si A=(HL) Z=1 Si A<>(HL) Z=0 H Sin Carry B_4 H=1 Con Carry B_4 H=0 N N=1 P/V Si BC-1<>0 P/V=1 Si BC-1=0 P/V=0	Compara el byte de datos de la localidad de memoria direccionada por el registro par HL con el **Acumulador**, si son iguales se activa el bit respectivo del **Flag** Posteriormente se incrementa HL y decrementa BC (contador) Si BC=0 ó A=(HL) se termina la instrucción, si BC<>0 y A<>(HL) el registro PC se decrementa en 2 y se repite la instrucción
CPD	1 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1	4	S Si Resultado<0 S=1 Si Resultado>=0 S=0 Z Si A=(HL) Z=1 Si A<>(HL) Z=0 H Sin Carry B_4 H=1 Con Carry B_4 H=0 N N=1 P/V Si BC-1<>0 P/V=1 Si BC-1=0 P/V=0	Compara el byte de datos de la localidad de memoria direccionada por el registro par HL con el **Acumulador**, si son iguales se activa el bit respectivo del **Flag** Posteriormente se decrementa HL y BC (contador)
CPDR	1 1 1 0 1 1 0 1 1 0 1 1 1 0 0 1	Si BC<>0 y A<>(HL) 5 Si BC=0 ó A=(HL) 4	S Si Resultado<0 S=1 Si Resultado>=0 S=0 Z Si A=(HL) Z=1 Si A<>(HL) Z=0 H Sin Carry B_4 H=1 Con Carry B4 H=0 N N=1 P/V Si BC-1<>0 P/V=1 Si BC-1=0 P/V=0	Compara el byte de datos de la localidad de memoria direccionada por el registro par HL con el acumulador, si son iguales se activa el bit respectivo del **Flag** Posteriormente se decrementa HL y BC (contador) Si BC=0 ó A=(HL) se termina la instrucción, si BC<>0 y A<>(HL) el registro PC se decrementa en 2 y se repite la instrucción

3) **Instrucciones Aritméticas y Lógicas**: Existen dos grupos básicos de instrucciones lógicas y aritméticas: de 8 y de 16 bits. En el primer grupo se encuentran las instrucciones aritméticas **ADD, ADC, SUB, SBC, CP, INC** y **DEC** y las instrucciones lógicas **AND, OR** y **XOR**. Las instrucciones aritméticas y lógicas también pueden ser empleadas para manipular datos de 16 bits, usando los registros pares **HL, IX, IY, BX** y **SP**.

En este grupo también se ubican instrucciones que permiten realizar operaciones de precisión múltiple en número binarios **BCD** con signo y sin signo, o el complemento a dos de números con signo.

También se ubican en este grupo las instrucciones **CPL, NEG, CCF** y **SCF**, como se muestra a continuación:

Mnem	Cód. de Operación $B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$	Ciclos	Flags Afectados	Descripción
ADD A, r	1 0 0 0 0 r r r	1	S: Si Resultado<0, S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0, Z=1 Si Resultado<>0 Z=0 H:Con Carry B_4 H=1 Sin Carry B4 H=0 N: N=0 C:Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V: Overflow P/V=1 Sin Overflow P/V=0	Suma el contenido del registro r al **Acumulador** y almacena el resultado en el **Acumulador**
ADD A, n	1 1 0 0 0 1 1 0 n n n n n n n n	2	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0, Z=1 Si Resultado<>0 Z=0 H:Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 C:Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V: Overflow P/V=1 Sin Overflow P/V=0	Suma el operando n al contenido del **Acumulador** y almacena el resultado en el **Acumulador**
ADD A, (HL)	1 0 0 0 0 1 1 0	2	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0, Z=1 Si Resultado<>0 Z=0 H:Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 C:Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V:Overflow P/V=1 Sin Overflow P/V=0	Suma el byte de datos de la localidad de memoria direccionada por el registro par **HL** con el **Acumulador**, almacena el resultado en el **Acumulador**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
ADD A,(IX+d)	1 1 0 1 1 1 0 1 1 0 0 0 0 1 1 0 d d d d d d d d	5	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H:Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 C:Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V:Overflow P/V=1 Sin Overflow P/V=0	Suma el byte de datos de la localidad de memoria direccionada por la suma del registro índice IX y el byte **d** con el **Acumulador**, almacena el resultado en el **Acumulador**
ADD A,(IY+d)	1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 d d d d d d d d	5	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H:Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 C:Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V:Overflow P/V=1 Sin Overflow P/V=0	Suma el byte de datos de la localidad de memoria direccionada por la suma del registro índice IY y el byte **d** con el **Acumulador**, almacena el resultado en el **Acumulador**
ADC A, r	1 0 0 0 1 r r r	1	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H:Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 C:Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V:Overflow P/V=1 Sin Overflow P/V=0	Suma el byte de datos del registro **r** y el **Bit de Acarreo** con el **Acumulador** almacena el resultado en el **Acumulador**
ADC A, n	1 1 0 0 1 1 1 0 n n n n n n n n	2	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H:Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 C:Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V:Overflow P/V=1 Sin Overflow P/V=0	Suma el byte de datos del operando **n** y el **Bit de Acarreo** con el **Acumulador**, almacena el resultado en el **Acumulador**
ADC A, (HL)	1 0 0 0 1 1 1 0	2	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H:Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 C:Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V:Overflow P/V=1 Sin Overflow P/V=0	Suma el byte de datos de la localidad de memoria direccionada por el registro par **HL** y el **Bit de Acarreo** con el **Acumulador**, almacena el resultado en el **Acumulador**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
ADC A, (IX+d)	1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Suma el byte de datos de la localidad de memoria direccionada por la suma del registro índice IX y el byte **d** además del **Bit de Acarreo** con el **Acumulador** almacena el resultado en el **Acumulador**
ADD A,(IY+d)	1 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Suma el byte de datos de la localidad de memoria direccionada por la suma del registro índice IY y el byte **d** además del **Bit de Acarreo** con el **Acumulador** almacena el resultado en el **Acumulador**
SUB r	1 0 0 1 0 r r r	1	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos del registro **r** con el **Acumulador**, almacena el resultado en el **Acumulador**
SUB n	1 1 0 1 0 1 1 0 n n n n n n n n	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos del operando **n** con el **Acumulador** almacena el resultado en el **Acumulador**
SUB (HL)	1 0 0 1 0 1 1 0	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos de la localidad de memoria direccionada por el registro par **HL** con el **Acumulador**, almacena el resultado en el **Acumulador**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
SUB (IX+d)	1 1 0 1 1 1 0 1 1 0 0 1 0 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos de la localidad de memoria direccionada por la suma del registro índice IX y el byte d con el **Acumulador** almacena el resultado en el **Acumulador**
SUB (IY+d)	1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos de la localidad de memoria direccionada por la suma del registro índice IY y el byte d con el **Acumulador** almacena el resultado en el **Acumulador**
SBC A, r	1 0 0 1 1 r r r	1	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos del registro r y el **Bit de Acarreo** con el **Acumulador**, almacena el resultado en el **Acumulador**
SBC A, n	1 1 0 1 1 1 1 0 n n n n n n n n	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos del operando n y el **Bit de Acarreo** con el **Acumulador**, almacena el resultado en el **Acumulador**
SBC A, (HL)	1 0 0 1 1 1 1 0	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos de la localidad de memoria direccionada por el registro par **HL** y el **Bit de Acarreo** con el **Acumulador** almacena el resultado en el **Acumulador**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
SBC A, (IX+d)	1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos de la localidad de memoria direccionada por la suma del registro índice IX y el byte **d** además del **Bit de Acarreo** con el **Acumulador** almacena el resultado en el **Acumulador**
SBC A,(IY+d)	1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 0 d d d d d d d d	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Resta el byte de datos de la localidad de memoria direccionada por la suma del registro índice IY y el byte **d** además del **Bit de Acarreo** con el **Acumulador** almacena el resultado en el **Acumulador**
AND r	1 0 1 0 0 r r r	1	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectua una operación **AND** lógica del registro **r** con el **Acumulador**, almacena el resultado en el **Acumulador**
AND n	1 1 1 0 0 1 1 0 n n n n n n n n	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectua una operación **AND** lógica del operando **n** con el **Acumulador**, almacena el resultado en el **Acumulador**
AND (HL)	1 0 1 0 0 1 1 0	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectua una operación **AND** lógica del byte de datos de la localidad de memoria direccionada por el registro par **HL** con el **Acumulador**, almacena el resultado en el **Acumulador**
AND (IX+d)	1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectua una operación **AND** lógica del byte de datos de la localidad de memoria direccionada por la suma del registro índice **IX** y el byte **d** además con el **Acumulador**, almacena el resultado en el **Acumulador**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
AND (IY+d)	1 1 1 1 1 1 0 1 1 0 1 0 0 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectúa una operación **AND** lógica del byte de datos de la localidad de memoria direccionada por la suma del registro índice **IY** y el byte **d** además con el **Acumulador** almacena el resultado en el **Acumulador**
OR r	1 0 1 1 0 r r r	1	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectúa una operación **OR** lógica del registro **r** con el **Acumulador** almacena el resultado en el **Acumulador**
OR n	1 1 1 1 0 1 1 0 n n n n n n n n	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectúa una operación **OR** lógica del operando **n** con el **Acumulador** almacena el resultado en el **Acumulador**
OR (HL)	1 0 1 1 0 1 1 0	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectúa una operación **OR** lógica del byte de datos de la localidad de memoria direccionada por el registro par **HL** con el **Acumulador** almacena el resultado en el **Acumulador**
OR (IX+d)	1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectúa una operación **OR** lógica del byte de datos de la localidad de memoria direccionada por la suma del registro índice **IX** y el byte **d** con el **Acumulador** almacena el resultado en el **Acumulador**
OR (IY+d)	1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectúa una operación **OR** lógica del byte de datos de la localidad de memoria direccionada por la suma del registro índice **IY** y el byte **d** con el **Acumulador** almacena el resultado en el **Acumulador**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
XOR r	1 0 1 0 1 r r r	1	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectua una operación XOR lógica del registro r con el **Acumulador** almacena el resultado en el **Acumulador**
XOR n	1 1 1 0 1 1 1 0 n n n n n n n n	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectua una operación XOR lógica del operando n con el **Acumulador**, almacena el resultado en el **Acumulador**
XOR (HL)	1 0 1 0 1 1 1 0	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectúa una operación XOR lógica del byte de datos de la localidad de memoria direccionada por el registro par HL con el **Acumulador**, almacena el resultado en el **Acumulador**
XOR (IX+d)	1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectúa una operación XOR lógica del byte de datos de la localidad de memoria direccionada por la suma del registro índice IX y el byte d con el **Acumulador** almacena el resultado en el **Acumulador**
XOR (IY+d)	1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H H=1 N N=0 C C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Efectua una operación XOR lógica del byte de datos de la localidad de memoria direccionada por la suma del registro índice IY y el byte d con el **Acumulador**, almacena el resultado en el **Acumulador**
CP r	1 0 1 1 1 r r r	1	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=1 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Compara el contenido del registro r con el **Acumulador**, si la comparación es verdadera se activa el bit respectivo

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
CP n	1 1 1 1 1 1 1 0 n n n n n n n n	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=1 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Compara el contenido del operando **n** con el **Acumulador**, si la comparación es verdadera se activa el bit respectivo
CP (HL)	1 0 1 1 1 1 1 0	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=1 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Compara el contenido de la localidad de memoria direccionada por el registro par **HL** con el acumulador, si la comparación es verdadera se activa el bit respectivo
CP (IX+d)	1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=1 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Compara el contenido de la localidad de memoria direccionada por la suma del registro índice **IX** y el byte **d** con el **Acumulador**, si la comparación es verdadera se activa el bit respectivo
CP (IY+d)	1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 d d d d d d d d	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=1 C Con Carry B_7 C=1 Sin Carry B_7 C=0 P/V Overflow P/V=1 Sin Overflow P/V=0	Compara el contenido de la localidad de memoria direccionada por la suma del registro índice **IY** y el byte **d** con el **Acumulador**, si la comparación es verdadera se activa el bit respectivo
INC r	0 0 r r r 1 0 0	1	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H Con Carry B_4 H=1 Sin Carry B_4 H=0 N N=0 Si P/V r=7Fh antes de operación P/V=1 Si R<>7Fh antes de operación P/V=0	Incrementa el contenido del registro r

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
INC (HL)	0 0 1 1 0 1 0 0	3	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H: Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 P/V: Si (HL)=7Fh antes de operación P/V=1 Si (HL)<>7Fh antes de operación P/V=0	Incrementa el contenido de la localidad de memoria direccionada por el registro par HL
INC (IX+d)	1 1 0 1 1 1 0 1 0 0 1 1 0 1 0 0 d d d d d d d d	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H: Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=0 P/V: Si (HL)=7Fh antes de operación P/V=1 (HL)<>7Fh antes de la operación P/V=0	Incrementa el contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d
INC (IY+d)	1 1 1 1 1 1 0 1 0 0 1 1 0 1 0 0 d d d d d d d d	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H: Con Carry B_4 H=1 Sin Carry B4 H=0 N: N=0 P/V: Si (HL)=7Fh antes de operación P/V=1 Si (HL)<>7Fh antes de la operación P/V=0	Incrementa el contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d
DEC r	0 0 r r r 1 0 1	1	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H: Con Carry B_4 H=1 Sin Carry B4 H=0 N: N=1 P/V: Si r=80h antes de la operación P/V=1 Si r<>80h antes de operación P/V=0	Decrementa el registro r
DEC (HL)	0 0 1 1 0 1 0 1	3	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H: Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=1 P/V: Si r=80h antes de la operación P/V=1 Si r<>80h antes de operación P/V=0	Decrementa el contenido de la localidad de memoria direccionada por el registro par HL

Mnem.	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
DEC (IX+d)	1 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 d d d d d d d d	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H: Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=1 P/V: Si r=80h antes de la operación P/V=1 Si r<>80h antes de operación P/V=0	Decrementa el contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d
DEC (IY+d)	1 1 1 1 1 1 0 1 0 0 1 1 0 1 0 1 d d d d d d d d	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultad0=0 Z=1 Si Resultado<>0 Z=0 H: Con Carry B_4 H=1 Sin Carry B_4 H=0 N: N=1 P/V: Si r=80h antes de la operación P/V=1 Si r<>80h antes de operación P/V=0	Decrementa el contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d

Las instrucciones aritméticas de 16 bis se muestran a continuación.

Mnem.	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
ADD HL, ss	0 0 s s 1 0 0 1	3	H: Con Carry de B_{11} H=1 Sin Carry de B_{11} H=0 N: H=0 C: Con Carry de B_{15} C=1 Sin Carry de B_{15} C=0	Suma el contenido del registro par **ss** al contenido de **HL**. Almacena el resultado en **HL**
ADD HL, ss	1 1 1 0 1 1 0 1 0 1 s s 1 0 1 0	4	S: Si Resultado<0 S=1 Si Resultado>=0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: Con Carry B_{11} H=1 Sin Carry B_{11} H=0 N: N=0 C: Con Carry B_{15} C=1 Sin Carry B_{15} C=0 P/V: Overflow P/V=1 Sin Overflow P/V=0	Suma el contenido del registro par **ss** y el bit de **Acarreo** al contenido de **HL**. Almacena el resultado en **HL**
SBC HL, ss	1 1 1 0 1 1 0 1 0 1 s s 0 0 1 0	4	S: Si Resultado<0 S=1 Si Resultado>=0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: Sin Carry B_{12} H=1 Con Carry B_{12} H=0 N: N=1 C: Con Carry Neg C=1 Sin Carry Neg C=0 P/V: Overflow P/V=1 Sin Overflow P/V=0	Resta el contenido del registro par **ss** y el bit de **Acarreo** del contenido del registro par **HL**. Almacena el resultado en **HL**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
ADD IX, pp	1 1 0 1 1 1 0 1 0 0 p p 1 0 0 1	4	H Con Carry de B_{11} H=1 Sin Carry de B_{11} H=0 N H=0 C Con Carry de B_{15} C=1 Sin Carry de B_{15} C=0	Suma el contenido del registro par pp al contenido del registro índice IX Almacena el resultado en IX
ADD IY, rr	1 1 1 1 1 1 0 1 0 0 r r 1 0 0 1	4	H Con Carry de B_{11} H=1 Sin Carry de B_{11} H=0 N H=0 C Con Carry de B_{15} C=1 Sin Carry de B_{15} C=0	Suma el contenido del registro par rr al contenido del registro índice IY Almacena el resultado en IY
INC ss	0 0 s s 0 0 1 1	1	Ninguno	Incrementa el contenido del registro **ss**
INC IX	1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 1	2	Ninguno	Incrementa el contenido del registro índice IX
INC IY	1 1 1 1 1 1 0 1 0 0 1 0 0 0 1 1	2	Ninguno	Incrementa el contenido del registro índice IY
DEC ss	0 0 s s 1 0 1 1	1	Ninguno	Decrementa el registro **ss**
DEC IX	1 1 0 1 1 1 0 1 0 0 1 0 1 0 1 1	2	Ninguno	Decrementa el contenido del registro índice IX
DEC IY	1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 1	2	Ninguno	Decrementa el contenido del registro índice IY

4) **Instrucciones de Rotación y Desplazamiento**: Estas instrucciones mueven bits específicos a la derecha o izquierda acorde a determinadas reglas. El conjunto de instrucciones de este grupo se muestra a continuación:

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
RLCA	0 0 0 0 0 1 1 1	1	H H=0 N N=0 C C=B_7 de A	Gira a la izquierda el contenido del **Acumulador** El B_7 se copia en C y en B_0
RLA	0 0 0 1 0 1 1 1	1	H H=0 N N=0 C C=B_7 de A	Gira a la izquierda el contenido del **Acumulador** El contenido de C se copia en B_0 y B_7 en C
RRCA	0 0 0 0 1 1 1 1	1	H H=0 N N=0 C C=B_0 de A	Gira el contenido del **Acumulador** hacia la derecha El B_0 se copia en el B_7 y en C
RRA	0 0 0 1 1 1 1 1	1	H H=0 N N=0 C C=B_0 de A	Gira a la derecha el contenido del **Acumulador** El bit C se copia en el B_7 y B_0 en C
RLC r	1 1 0 0 1 0 1 1 0 0 0 0 0 r r r	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H H=0 N N=0 C C=B_7 de r P/V Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido del registro r El B_7 se copia en C y en B_0

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
RLC (HL)	1 1 0 0 1 0 1 1 0 0 0 0 0 1 1 0	4	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido de la localidad de memoria direccionada por el registro par HL. El B_7 se copia en C y en B_0
RLC (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 0 0 0 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d. El bit B_7 se copia en C y en B_0
RLC (IY+D)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 0 0 0 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C:C=B_7 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d. El bit B_7 se copia en C y en B_0
RL r	1 1 0 0 1 0 1 1 0 0 0 1 0 r r r	2	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido del registro r. El bit C se copia en B_0 y B_7 se copia en C
RL (HL)	1 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0	4	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido de la localidad de memoria direccionada por el registro par HL. El bit C se copia en B_0 y B_7 se copia en C

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
RL (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 0 1 0 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d. El bit C se copia en B_0 y B_7 se copia en C
RL (IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 0 1 0 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d. El bit C se copia en B_0 y B_7 se copia en C
RRC r	1 1 0 0 1 0 1 1	2	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la derecha el contenido del registro r. El bit B_0 se copia en B_7 y en C
RRC (HL)	1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 0	4	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la derecha el contenido de la localidad de memoria direccionada por el registro par HL. El bit B_0 se copia en B_7 y en C
RRC (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 0 0 1 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 del registro fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Gira a la derecha el contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d. El bit B_0 se copia en B_7 y en C

Mnem	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
RRC (IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 0 0 1 1 1 0	6	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H H=0 N N=0 C C=B_7 del registro fuente P/V Paridad par P/V=1 Paridad impar P/V=0	Gira a la izquierda el contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d. El bit B_0 se copia en B_7 y en C
RR r	1 1 0 0 1 0 1 1 0 0 0 1 1 r r r	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H H=0 N N=0 C C=B_0 del registro fuente P/V Paridad par P/V=1 Paridad impar P/V=0	Gira a la derecha el contenido del registro r. El bit C se copia en B_7 y B_0 se copia en C
RR (HL)	1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0	4	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H H=0 N N=0 C C=B_7 del registro fuente P/V Paridad par P/V=1 Paridad impar P/V=0	Gira a la derecha el contenido de la localidad de memoria direccionada por el registro par HL. El bit C se copia en B_7 y B_0 se copia en C
RR (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 0 1 1 1 1 0	6	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H H=0 N N=0 C C=B_7 del registro fuente P/V Paridad par P/V=1 Paridad impar P/V=0	Gira a la derecha el contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d. El bit C se copia en el B_7 y B_0 se copia en C
RR (IY+d)	0 0 0 1 1 1 1 0 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 0 1 1 1 1 0	6	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H H=0 N N=0 C C=B_7 de la fuente P/V Paridad par P/V=1 Paridad impar P/V=0	Gira a la derecha el contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d. El bit C se copia en B_7 y B_0 se copia en C
SLA r	1 1 0 0 1 0 1 1 0 0 1 0 0 r r r	2	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H H=0 N N=0 C C=B_7 de la fuente P/V Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento aritmético a la izquierda del contenido del registro r. El bit B_7 se copia en C

Mnem	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
SLA (HL)	1 1 0 0 1 0 1 1 0 0 1 0 0 1 1 0	4	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento aritmético a la izquierda del contenido de la localidad de memoria direccionada por el registro par HL. El bit B_7 se copia en C
SLA (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 1 0 0 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 de la fuente P/V: Paridad par P/V=1 Paridad impar: P/V=0	Realiza un desplazamiento aritmético a la izquierda del contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d. El bit B_7 se copia en C
SLA (IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 1 0 0 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_7 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento aritmético a la izquierda del contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d. El bit B_7 se copia en C
SRA r	1 1 0 0 1 0 1 1 0 0 1 0 1 r r r	2	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 de la fuente P/V: Paridad par P/V=1 Paridad impar: P/V=0	Realiza un desplazamiento aritmético a la derecha del contenido del registro r (sin modificar el bit B_7). El bit B_0 se copia en C
SRA (HL)	1 1 0 0 1 0 1 1 0 0 1 0 0 1 1 0	4	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento aritmético a la derecha del contenido de la localidad de memoria direccionada por el registro par HL (sin modificar el bit B_7). El bit B_0 se copia en C
SRA (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 1 0 1 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento aritmético a la derecha del contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d (sin modificar el bit B_7). El bit B_0 se copia en C

Mnem	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
SLA (IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 1 0 0 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento aritmético a la derecha del contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d (sin modificar el bit B_7). El bit B_0 se copia en C
SRL r	1 1 0 0 1 0 1 1 0 0 1 1 1 r r r	2	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento a la derecha del contenido del registro r. El bit B_0 se copia en C y se pone a 1 el bit B_7
SRL (HL)	1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0	4	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento a la derecha del contenido de la localidad de memoria direccionada por el registro par HL. El bit B_0 se copia en C y se pone a 1 el bit B_7
SRL (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 1 1 1 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento aritmético a la derecha del contenido de la localidad de memoria direccionada por la suma del registro índice IX y el byte d. El bit B_0 se copia en C y se pone a 1 el bit B_7
SRL (IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 0 1 1 1 1 1 0	6	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 C: C=B_0 de la fuente P/V: Paridad par P/V=1 Paridad impar P/V=0	Realiza un desplazamiento aritmético a la derecha del contenido de la localidad de memoria direccionada por la suma del registro índice IY y el byte d. El bit B_0 se copia en C y se pone a 1 el bit B_7
RLD	1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1	5	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 P/V: Paridad par P/V=1 Paridad impar P/V=0	Copia el nibble de menor peso al nibble de mayor peso de la localidad de memoria direccionada por el registro par HL. El anterior nibble de mayor peso se copia en el nibble de menor peso del **Acumulador** y el nibble anterior de menor peso del **Acumulador** se copia en el nibble de menor peso de la localidad de memoria, no se modifica el nibble de mayor peso del **Acumulador**

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
RRD	1 1 1 0 1 1 0 1 0 1 1 0 0 1 1 1	5	S Si Resultado<0 S=1 Si Resultado<>0 S=0 Z Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H H=0 N N=0 P/V Paridad par P/V=1 Paridad impar P/V=0	Copia el nibble de menor peso de la localidad de memoria direccionada por el registro par HL al nibble de menor peso del **Acumulador** El nibble de menor peso del **Acumulador** se copia en el nibble de mayor peso de la localida de memoria y el nibble de mayor peso de esta localidad de memoria se copia en el nibble de menor peso de la misma No se modifica el nibble de mayor peso del **Acumulador**

5) **Instrucciones de Manipulación de Bits**: Estas instrucciones permiten el examen de bits individuales de un registro o de una localidad de memoria, activarlos o desactivarlos. Se puede usar para ello el direccionamiento indexado o por registro indirecto para ubicar el dato en la localidad de memoria. De igual forma, se puede utilizar cualquiera de los ocho bits, los siete diferentes registros o cualquier dato ubicado en la localidad de memoria direccionada por los registros pares **HL**, **IX** e **IY**. El conjunto de instrucciones que pertenecen a este grupo son:

Mnem.	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
BIT B, r	1 1 0 0 1 0 1 1 0 1 b b b r r r	2	S: S=Desconocido Z:bit=0 Z=1, bit<>0 Z=0 H: H=1 N: N=0 P/V P/V=Desconocido	Almacena el complemento del bit b indicado dentro del registro r en el bit Z del registro F
BIT B, (HL)	1 1 0 0 1 0 1 1 0 1 b b b 1 1 0	3	S S=Desconocido Z bit=0 Z=1, bit<>0 Z=0 H: H=1 N: N=0 P/V: P/V=Desconocido	Almacena en el bit Z del registro F el complemento del bit b indicado contenido en la localidad de memoria direccionada por el registro par HL
BIT B, (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 1 b b b 1 1 0	5	S: S=Desconocido Z:bit=0 Z=1, bit<>0 Z=0 H H=1 N: N=0 P/V: P/V=Desconocido	Almacena en el bit Z del registro F el complemento del bit b indicado contenido en la localidad de memoria direccionada por el registro índice IX más el complemento a 2 del byte d
BIT b,(IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 0 1 b d d 1 1 0	5	S: S=Desconocido Z: bit=0.Z=1, bit<>0 Z=0 H H=1 N: N=0 P/V P/V=Desconocido	Almacena en el bit Z del registro F el complemento del bit b indicado contenido en la localidad de memoria direccionada por el registro índice IY más el complemento a 2 del byte d
SET b, r	1 1 0 0 1 0 1 1 1 1 b b b r r r	2	Ninguno	Activa (pone a "1") el bit b indicado en el registro r
SET B, (HL)	1 1 0 0 1 0 1 1 1 1 b b b 1 1 0	4	Ninguno	Activa (pone a "1") el bit b indicado en la localidad de memoria direccionada por el registro par HL

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
SET b, (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 1 1 b b b 1 1 0	6	Ninguno	Activa (pone a "1") el bit b indicado en la localidad de memoria direccionada por la suma del registro índice IX y el complemento a 2 del byte d
SET b, (IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 1 1 b b b 1 1 0	6	Ninguno	Activa (pone a "1") el bit b indicado en la localidad de memoria direccionada por la suma del registro índice IY y el complemento a 2 del byte d
RES b, r	1 1 0 0 1 0 1 1 1 0 b b b r r r	4	Ninguno	Desactiva (pone a "0") el bit b en el registro r
RES B, (HL)	1 1 0 0 1 0 1 1 1 0 b b b 1 1 0	4	Ninguno	Desactiva (pone a "0") el bit b en la localidad de memoria direccionada por el registro par HL
RES b, (IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 1 0 b b b 1 1 0	6	Ninguno	Desactiva (pone a "0") el bit b de la localidad de memoria direccionada por la suma del registro índice IX con el complemento a 2 del byte d
RES b, (IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d d d d d d d d 1 0 b b b 1 1 0	6	Ninguno	Desactiva (pone a "0") el bit b de la localidad de memoria direccionada por la suma del registro índice IY con el complemento a 2 del byte d

6) **Instrucciones de Bifurcación, Llamada y Retorno**: Dado que una computadora ejecuta las instrucciones en forma secuencial, el contador de programa se incrementa en cada ocasión que una instrucción se ejecuta. El número de incrementos se determina por el número de bytes requeridos para un tipo particular de instrucción. Aún cuando esta ejecución secuencial es uno de los aspectos más poderosos de una computadora, éste limita el rango de los posibles problemas que pueden resolver este procesamiento secuencial. Es imposible realizar muchas de las operaciones que requieren tomar una decisión simple. Tales operaciones podrían ser por ejemplo un puerto de entrada que está conectada a un teclado. Se puede crear un ciclo usando una instrucción **JUMP** que verifique el bit de strobe (usualmente el bit 7) y si no está activo, efectuar una bifurcación al principio del ciclo. En caso contrario, si este se encuentra activo el programa podría ejecutar la siguiente instrucción. Sin embargo, muchos programas requieren decisiones lógicas que no podrían realizarse sin el uso de las instrucciones **JUMP**, **CALL** y **RETURN**. Las instrucciones que pertenecen a este grupo son:

Mnem.	Cód. de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
JP nn	1 1 0 0 0 0 1 1 n n n n n n n n n n n n n n n n	3	Ninguno	Almacena el operando par nn en el registro par PC y apunta a la dirección de la siguiente instrucción a ejecutar

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
JP cc,nn	1 1 cc cc cc 0 1 0	3	Ninguno	Almacena el operando par nn en el registro PC si la condición cc es verdadera comienza la ejecución del programa de esta dirección Si la condición es falsa el registro PC se incrementa en uno y continúa la ejecución normal
JR e	0 0 0 1 1 0 0 0 e-1	3	Ninguno	Añade el valor del desplazamiento e al registro PC y ejecuta la siguiente instrucción de la nueva localidad Tiene un margen de −126 a +129 bytes
JR C, e	0 0 1 1 1 0 0 0 e-2	Si C=1 3 Si C=0 12	Ninguno	Añade el desplazamiento e al registro PC si la condición es verdadera y ejecuta las instrucciones a partir de la nueva dirección Si la condición es falsa las instrucciones se toman a partir de la localidad siguiente a esta instrucción
JR NC,e	0 0 1 1 0 0 0 0 e-2	Si C=1 3 Si C=0 12	Ninguno	Añade el desplazamiento e al registro PC si la condición es verdadera y ejecuta las instrucciones a partir de la nueva dirección Si la condición es falsa las instrucciones se toman a partir de la localidad siguiente a esta instrucción
JR Z, e	0 0 1 0 1 0 0 0 3-2	Si C=1 3 Si C=0 12	Ninguno	Añade el desplazamiento e al registro PC si la condición es verdadera y ejecuta las instrucciones a partir de la nueva dirección Si la condición es falsa las instrucciones se toman a partir de la localidad siguiente a esta instrucción
JR NZ, e	0 0 1 0 0 0 0 0	Si C=1 3 Si C=0 12	Ninguno	Añade el desplazamiento e al registro PC si la condición es verdadera y ejecuta las instrucciones a partir de la nueva dirección Si la condición es falsa las instrucciones se toman a partir de la localidad siguiente a esta instrucción
JP (HL)	1 1 1 0 1 0 0 1	1	Ninguno	Almacena el contenido de HL en el registro PC y ejecuta las instrucciones a partir de la nueva dirección
JP (IX)	1 1 0 1 1 1 0 1 1 1 1 0 1 0 0 1	2	Ninguno	Almacena el contenido de IX en el registro PC y ejecuta las instrucciones a partir de la nueva dirección
JP (IY)	1 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1	2	Ninguno	Almacena el contenido de IY en el registro PC y ejecuta las instrucciones a partir de la nueva dirección

Mnem	Cód de Operación $B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$	Ciclos	Flags Afectados	Descripción
DJNZ	0 0 0 1 0 0 0 0 e-2	Si B<>0 3 Si B=0 2	Ninguno	Decrementa el registro B si el resultado es distinto de 0 añade e al registro PC y ejecuta las instrucciones a partir de la nueva dirección Si B es 0 las instrucciones se toman a partir de la siguiente dirección a esta instrucción
CALL nn	1 1 0 0 1 1 0 1 n n n n n n n n n n n n n n n n	5	Ninguno	Almacena el contenido del registro PC en la parte superior de la pila, carga el operando par nn en el registro PC y ejecuta las instrucciones a partir de la nueva dirección
CALL cc, nn	1 1 cc cc cc 1 0 0 n n n n n n n n n n n n n n n n	Si cc=V 5 Si cc=F 3	Ninguno	Almacena el contenido del registro PC en la parte superior de la pila si la condición es verdadera y carga el operando par nn en el registro PC Ejecuta las instrucciones a partir de la nueva dirección
RET	1 1 0 0 1 0 0 1	3	Ninguno	Extrae el contenido del registro PC de la pila y ejecuta las instrucciones a partir de la nueva dirección
RET cc	1 1 cc 0 0 0	Si cc=V 3 Si cc=F 1	Ninguno	Extrae el contenido del registro PC de la parte superior de la pila si la condición es verdadera y ejecuta las instrucciones a partir de la nueva dirección Si la condición es falsa las instrucciones se ejecutan en la dirección siguiente a la instrucción actual
RETI	1 1 1 0 1 1 0 1 0 1 0 0 1 1 0 1	4	Ninguno	Desactiva los flip-flops IFF1 e IFF2, restaura el registro PC e indica que ha terminado la rutina de interrupción
RETN	1 1 1 0 1 1 0 1 0 1 0 0 0 1 0 1	4	Ninguno	Ejecuta una instrucción de retorno incondicional de una interrupción no mascarable Reproduce el estado de IFF2 análogo al de IFF1 al estado anterior a la aceptación de una interrupción NMI
RST p*	1 1 t t t 1 1 1	3	Ninguno	Almacena el contenido del registro PC en la pila, añade el operando p al registro PC y ejecuta las instrucciones a partir de la nueva dirección

* Para una información más completa acerca de esta instrucción, consulte el manual de usuario del microprocesador Z80

7) **Instrucciones de Entrada/Salida**: Estas instrucciones permiten la transferencia de datos entre la CPU y los **Puertos de Entrada/Salida**. El microprocesador Z80 usa para ello diferentes registros e instrucciones de I/O que le permiten usar estos registros sin requerir primero transferirlos al **Acumulador**. Las instrucciones que pertenecen a este grupo se mencionan a continuación:

Mnem	Cód. de Operación $B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$	Ciclos	Flags Afectados	Descripción
IN A, (n)	1 1 0 1 1 0 1 1	3	Ninguno	Selecciona una dirección del byte inferior del **Bus de Direcciones**, coloca el contenido del **Acumulador** en el byte superior. Un byte es escrito por el puerto y se almacena en el **Acumulador**
IN r, (C)	1 1 1 0 1 1 0 1 0 1 r r r 0 0 0	3	S: Si Resultado<0 S=1 Si Resultado<>0 S=0 Z: Si Resultado=0 Z=1 Si Resultado<>0 Z=0 H: H=0 N: N=0 P/V: Overflow P/V=1 Sin Overflow P/V=0	Coloca el contenido del registro C en el byte inferior del **Bus de Direcciones**, el contenido del registro B en el byte superior del mismo bus, un byte de datos es colocado por el puerto seleccionado y almacena este dato en el registro r del microprocesador
INI	1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 0	4	S: S=desconocido Z: Si B-1=0 Z=1 Si B-1<>0 Z=0 H: H=desconocido N: N=1 P/V: P/V=desconocido	Coloca el contenido del registro C en el byte inferior del **Bus de Direcciones**, el contenido del registro B en el byte superior. El puerto seleccionado coloca un byte en el **Bus de Datos** y se almacena en el microprocesador central. Posterior a ello, almacena el dato en la localidad de memoria direccionada por el registro par HL. Finalmente decrementa el contador (registro B) y HL
INIR	1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 0	Si B<>0: 5 Si B=0: 4	S: desconocido Z: Z=1 H: H=desconocido N: N=1 P/V: P/V=desconocido	Coloca el contenido del registro C en el byte inferior del **Bus de Direcciones**, el contenido del registro B en el byte superior. El puerto seleccionado coloca un byte en el **Bus de Datos** y se almacena en el microprocesador central. Posterior a ello, se almacena en la localidad de memoria direcionada por el registro par HL. Finalmente decrementa HL y B, si el resultado es 0, el registro PC se decrementa en 2 y se repite la instrucción

Mnem.	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
IND	1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0	4	S S=desconocido Z Si B-1=0 Z=1 Si B-1<>0 Z=0 H H=desconocido N N=1 P/V P/V=desconocido	Coloca el registro C en el byte inferior del **Bus de Direcciones** el contenido del registro B en el byte superior El puerto seleccionado coloca un byte en el **Bus de Datos**, almacenándose en el microprocesador central Posterior a ello, se almacena en la localidad de memoria direccionada por el registro par HL Decrementa B (contador) y HL
INDR	1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0	4	S S=desconocido Z Z=1 H H=desconocido N N=1 P/V P/V=desconocido	Coloca el registro C en el byte inferior del **Bus de Direcciones** el contenido del registro B en el byte superior El puerto seleccionado coloca un bye en el **Bus de Datos** y se almacena en el microprocesador central Posterior a ello, se almacena en la localidad de memoria direcionada por HL Finalmente decrementa HL y B, si el resultado es 0, el registro PC se decrementa en 2 y se repite la instrucción
OUT (n), A	1 1 0 1 0 0 1 1 n n n n n n n n	3	Ninguno	Coloca el operando n en el byte inferior del **Bus de Direcciones**, el contenido del **Acumulador** en el byte superior Posteriormente se coloca el contenido del **Acumulador** en el **Bus de Datos** y se escribe en el puerto sleccionado
OUT (C), r	1 1 1 0 1 1 0 1 0 1 r r r 0 0 1	3	Ninguno	Coloca el contenido del registro C en el byte inferior del **Bus de Direcciones**, el contenido del registro B en el byte superior El dato de r se coloca en el **Bus de Datos** y se escribe en el puerto seleccionado
OUTI	1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1	4	S S=desconocido Z Si B-1=0 Z=1 Si B-1<>0 Z=0 H H=desconocido N N=1 P/V P/V=desconocido	Coloca el contenido del registro par HL en el **Bus de Direcciones**, almacena al dato de esta localidad de memoria en el microprocesador Decrementa el registro B (contador) Posterior a ello, coloca el contenido del registro C en el byte inferior del **Bus de Direcciones**, el contenido del registro B en el byte superior El byte de datos se coloca en el **Bus de Datos** y se escribe en el puerto seleccionado Finalmente incrementa HL

Mnem.	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
OTIR	1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1	Si B<>0: 5 Si B=0: 4	S: S=desconocido Z: Z=1 H: H=desconocido N: N=1 P/V: P/V=desconocido	Coloca el registro par HL en el **Bus de Direcciones**, almacena el dato de esta localidad de memoria en el microprocesador. Decrementa el registro B (contador). Posterior a ello, coloca el registro C en el byte inferior del **Bus de Direcciones**, B en el byte superior. El byte de datos se coloca en el **Bus de Datos** y se escribe en el puerto seleccionado. Finalmente incrementa HL. Si B decrementado no es 0, el registro PC se decrementa en 2 y se repite la instrucción.
OUTD	1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 1	4	S: S=desconocido Z: Si B-1=0: Z=1 Si B-1<>0: Z=0 H: H=desconocido N: N=1 P/V: P/V=desconocido	Coloca el registro par HL en el **Bus de Direcciones**, almacena el dato de esta localidad de memoria en el microprocesador. Decrementa el registro B (contador). Posterior a ello, coloca el registro C en el byte inferior del **Bus de Direcciones**, B en el byte superior. El byte de datos se coloca en el **Bus de Datos** y se escribe en el puerto seleccionado. Finalmente incrementa HL. Si B decrementado no es 0, el registro PC se decrementa en 2 y se repite la instrucción.
OTDR	1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1	Si B<>0: 5 Si B=0: 4	S: S=desconocido Z: Z=1 H: H=desconocido N: N=1 P/V: P/V=desconocido	Coloca el registro par HL en el **Bus de Direcciones**, almacena el dato de esta localidad de memoria en el microprocesador. Decrementa el registro B (contador). Posterior a ello, coloca el registro C en el byte inferior del **Bus de Direcciones**, B en el byte superior. El byte de datos se coloca en el **Bus de Datos** y se escribe en el puerto seleccionado. Finalmente incrementa HL. Si B decrementado no es 0, el registro PC se decrementa en 2 y se repite la instrucción.

8) **Instrucciones de Control de la CPU**: Existen varias instrucciones del Z80 usados exclusivamente para el control de la CPU: **NOP, HALT, DI, EI, IM1, MI2**. El grupo total de instrucciones en este grupo se muestra a continuación.

Mnem	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
CPL	0 0 1 0 1 1 1 1	1	H: H=1 N: N=1	Complementa a 1 el **Acumulador**
NEG	1 1 1 0 1 1 0 1 0 1 0 0 0 1 0 0	2	S: Si resultado<0 S=1 Si resultado<>0 S=0 Z: Si resultado=0 Z=1 Si resultado<>0 Z=0 H: Con carry B_4 H=1 Sin carry B_4 H=0 N: N=1 C: Si A=00h antes de la operación: C=1 Si A<>00h antes de la operación: C=0 P/V: Si A=80h antes de la operación: P/V=1 Si A<>80h antes de la operación: P/V=0	Complementa a 2 del **Acumulador**
CCF	0 0 1 1 1 1 1 1	1	H: H=Carry anterior N: N=0 C: Si CY=0 antes de la operación: C=1 Si CY<>0 antes de la operación: C=0	Niega (Invierte) el bit C del **Flag**
SCF	0 0 1 1 0 1 1 1	1	H: H=0 N: N=0 C: C=1	Activa (pone a "1") el bit C del **Flag**
NOP	0 0 0 0 0 0 0 0	1	Ninguno	Detiene las operaciones del procesador central en este ciclo
DAA	0 0 1 0 0 1 1 1	1	S: B_7=1 de A después de la operación: S=1 Si $B_{7<>1}$ de A después de la operación: S=0 Z: Si A=0 después de la operación: Z=1 Si A<>0 después de la operación: Z=0 H: * C: * P/V: Paridad A par después de la operación P/V=1 Paridad A impar después de la operación: P/V=0	Ajusta condicionalmente el contenino del **Acumulador** para las operaciones de suma y resta en **BCD**
HALT	0 1 1 1 0 1 1 0	1	Ninguno	Suspende las operaciones del microprocesador central con instrucciones **NOP** para regenerar la memoria hasta recibir una interrupción

* Para más información acerca de esta instrucción consulte el manual del usuario del microprocesador Z80

Mnem.	Cód de Operación B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0	Ciclos	Flags Afectados	Descripción
DI	1 1 1 1 1 0 1 1	1	Ninguno	Inhibe la interrupción mascarable desactivando los flip-flops de interrupción (IFF1, IFF2)
EI	1 1 1 1 1 0 1 1	1	Ninguno	Habilita la interrupción mascarable activando los flip-flops de interrupción (IFF1, IFF2)
MI 0	1 1 1 0 1 1 0 1 0 1 0 0 0 1 1 0	2	Ninguno	Activa el Modo de Interrupción 0
MI 1	1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0	2	Ninguno	Activa el Modo de Interrupción 1
IM 2	1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0	2	Ninguno	Activa el Modo de Interrupción 2

2.9 Resumen

El microprocesador Z80 es el más poderoso dentro de la familia de los microprocesadores de 8 bits ya que cuenta con más del doble de instrucciones de su contraparte 8080 de Intel, otra de sus ventajas es que sólo requiere de una fuente de alimentación unifase de 5 volts, cuenta además, con un bus bidireccional de datos de 8 bits, un bus de direcciones de 16 bits (lo cual le permite direccionar hasta 64 Kb de memoria), además de un conjunto de registros de Propósito Dedicado y un conjunto de registros de Propósito General.

Los registros de Propósito Dedicado se encuentra se encuentra conformados por los registros I, R, IX, IY, SP, PC y F.

Los registros de Propósito General se dividen (para una mayor claridad en su operación) en un conjunto de registros principales (A, F, B, C, D H y L) y un conjunto de registros alternos (A', F', B', C', D', E', H' y L') los cuales duplican los registros anteriores.

Estos registros pueden ser agrupados por pares para formar registros de 16 bits (AF, BC, DE y HL), así como sus correspondientes registros pares alternos (A'F', B'C', D'E' y H'L').

Para poder realizar diversas operaciones de cálculo, lectura/escritura de datos, el microprocesador Z80 cuenta con diversas señales de control, las cuales son:

1) Señales de Entrada: Estas señales son generadas por el microprocesador e indican que se va a realizar una operación sobre un dispositivo externo al microprocesador.
2) Señales de Salida: Estas señales son generadas por los dispositivos periféricos, e indican la petición de una operación por parte del microprocesador
3) Señales de Entrada/Salida: Estas señales son usadas para intercambiar información entre los dispositivos periféricos de entrada/salida y el microprocesador.

Las características más importantes del microprocesador Z80 y las cuales le dan fama entre los mismos microprocesadores de su familia son sin lugar a dudas su capacidad de manejar diversos tipos de interrupciones y su capacidad de manejar diversos tipos de direccionamiento de memoria.

Gracias a su capacidad de manejar interrupciones, el microprocesador Z80 puede intercambiar información entre éste y los dispositivos periféricos con una mayor rapidez y fiabilidad que los microprocesadores análogos de esta familia.

De igual forma, gracias a su capacidad de manejar los direccionamientos de memoria, se puede realizar cualquier operación sobre un segmento de memoria o sobre un dato específico.

Tampoco puede quedar atrás su conjunto de instrucciones, ya que mediante una serie de combinación de argumentos, éstos sobrepasan las 158 instrucciones mencionadas con anterioridad.

CAPITULO III: INTERCONEXIÓN DE MEMORIA Y PUERTOS

"Finalmente he comprendido lo que significa compatibilidad hacia arriba: mantener los errores del pasado"

Dennis Van Tassel

3.1 Interconexión de Memorias

Antes de explicar la forma en que el microprocesador Z80 utiliza los **Buses de Dirección, Datos y Control** para comunicarse con los **Dispositivos Periféricos** mediante los **Puertos de Entrada/Salida**, explicaremos en forma breve la notación genérica de la forma de interconexión de cualquier dispositivo periférico a un microprocesador.

Los dos tipos de memoria existentes en el mercado y que forman parte de las microcomputadoras son las memorias **ROM** y **RAM** (indispensables para su funcionamiento como sistema, lo cual ya se explicó en un apartado anterior).

Dependiendo de la forma de conexión entre un microprocesador y los dispositivos de memoria, existen dos tipos:

1) El primer tipo se forma a través de **Conexiones Diferentes**, tal como se muestra en la siguiente figura:

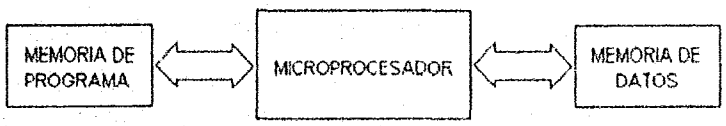


Fig. 3.1 CONEXIÓN DE LAS MEMORIAS AL MICROPROCESADOR EN FORMA INDEPENDIENTE

Esta forma de conexión presenta la desventaja de tener un número elevado de terminales de la **Unidad Central de Procesos (CPU)**. A pesar de ello, si se toma en cuenta el buen diseño de la **Unidad de Control**, esta forma de conexión realiza procesos de búsqueda y ejecución de instrucciones en paralelo **(Parallel Proccesing)**, permitiendo reducir en forma considerable el tiempo de ejecución de las instrucciones.

2) El segundo tipo involucra una **Conexión Común**, tal como se indica en la siguiente figura:

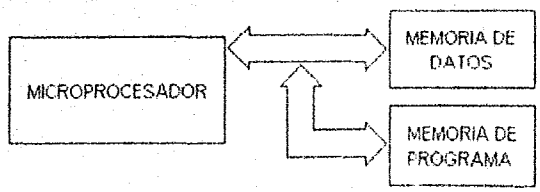


Fig. 3.2 CONEXIÓN DE LAS MEMORIAS AL MICROPROCESADOR EN FORMA COMÚN

Esta forma de conexión permite el intercambio de información entre el microprocesador y uno de los dispositivos de memoria a la vez. Además, permite la reducción de los elementos de control existentes en el microprocesador.

Para realizar con éxito una conexión común del microprocesador con las memorias, debe tomarse en cuenta los siguientes elementos básicos de los microprocesadores:

a) Las **Líneas de Dirección** que permiten seleccionar la localidad de memoria a la cual se desea accesar.
b) Las **Líneas de Datos** a través de las cuales se puede leer y escribir información.
c) Las **Líneas de Control** que permiten seleccionar el tipo de operación que se efectúa sobre las memorias (selección, lectura y escritura).

La utilización simultanea de estas líneas permiten diversas operaciones sobre las memorias tales como selección, lectura y escritura.

Profundizando un poco más, podemos decir que existen varias formas de conexión de las memorias al microprocesador, las cuales son:

1) Al ser la memoria **RAM** un dispositivo de lectura y escritura, utiliza los tres tipos de líneas mencionadas, razón por la cual su forma de conexión es la siguiente:

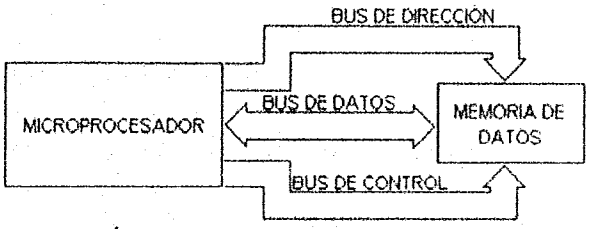


Fig. 3.3 CONEXIÓN DE UNA MEMORIA RAM A UN MICROPROCESADOR

Para poder realizar este tipo de conexión se requiere que el microprocesador también cuente con los tres tipos de líneas. Estas líneas constituyen el **Bus de Direcciones, Datos** y **Control** los cuales se activan en instantes de tiempo diferentes para poder efectuar procesos de lectura o escritura sobre determinadas localidades específicas de memoria (localidades determinadas por el conjunto de **Líneas de Dirección** activas).

2) Para disminuir el número de pines existentes en un microprocesador se pueden compartir las conexiones comunes entre los tres tipos de líneas. Basado en la forma de uso en común de los **Buses de Direcciones, Datos y Control** existen dos tipos principales:
   a) **Los Datos y las Direcciones Comparten el Mismo Bus**: Para utilizar con éxito este tipo de conexiones, alguna de las dos señales debe de almacenarse en el exterior del microprocesador. Normalmente, la información se almacena internamente en el microprocesador: si es una instrucción se almacena en el registro IR, si es un dato en un registro auxiliar. En el exterior del microprocesador se coloca un **Registro de Dirección** como se muestra en la figura número 3.4.

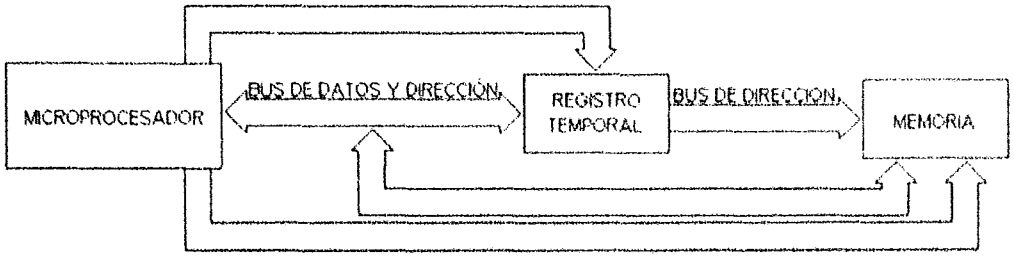


Fig. 3.4 CONEXIÓN DE UNA MEMORIA Y MICROPROCESADOR CON BUS DE DATOS Y DIRECCIÓN COMÚN

Gracias a este método, se permite la disminución del número de pines, logrando con ello integrar en una sola pastilla de silicio todos los elementos de una microcomputadora: **CPU, Memoria y Puertos de Entrada/Salida Aisladas**.

b) **Los Datos y las Señales de Control Comparten el Mismo Bus**: Para este caso su teoria de funcionamiento es análoga al anterior.

Los datos se almacenan en el interior del microprocesador en tanto que las señales de control se almacenan en un registro externo, tal como se puede apreciar en la siguiente figura:

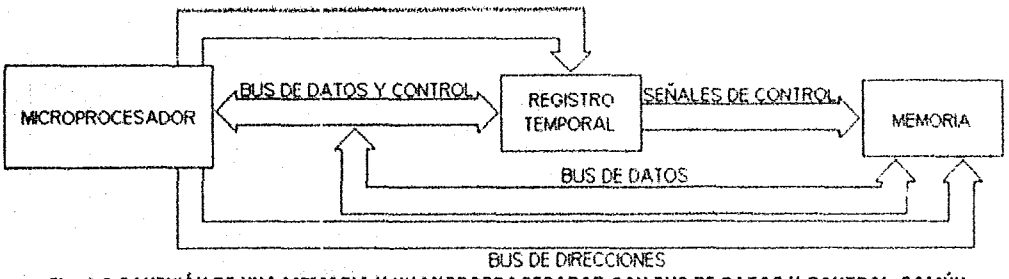


Fig. 3.5 CONEXIÓN DE UNA MEMORIA Y UN MICROPROCESADOR CON BUS DE DATOS Y CONTROL COMÚN

Aún cuando existen otros tipos de estructuras, estos no serán mostrados puesto que caen fuera del propósito de este trabajo.

3.2 Interconexión de Dispositivos Periféricos

Como ya se ha mencionado anteriormente, los dispositivos periféricos se conectan a un sistema microcomputador mediante un **Puerto de Entrada/Salida**, el cual se forma esencialmente de una dirección ubicada en el rango del **Bus de Direcciones (Puertos de Entrada/Salida Mapeadas)** del microprocesador o en el rango de las **Líneas de Dirección** que estén configuradas para direccionar **Puertos de Entrada/Salida (Puertos de Entrada/Salida Aisladas)**, así como de una interfaz para permitir la comunicación de datos entre el microprocesador y los dispositivos periféricos. En lo sucesivo se supondrá que cuando se habla de la interconexión de cualquier dispositivo periférico a un microprocesador, éste contará con una interfaz y una dirección asignada para permitir el intercambio de datos entre el mismo dispositivo periférico y el microprocesador.

Al igual que existen diferentes formas de conectar los dispositivos de memoria al microprocesador, también existen para la conexión de los dispositivos periféricos. La forma genérica de interconexión de un dispositivo periférico con el microprocesador lo muestra la siguiente figura:

Fig. 3.6 CONEXIÓN DE UN MICROPROCESADOR Y UN DISPOSITIVO PERIFÉRICO

Básicamente existen dos formas de conexión entre el microprocesador y los dispositivos periféricos:

1) **Utilizando Conexiones Independientes Para Cada Dispositivo Periférico**: Ésta forma de conexión permite una comunicación única entre el microprocesador y los dispositivos periféricos ya que asigna un **Bus de Datos, Direcciones y Control** para cada dispositivo periférico conectado al sistema, tal como se muestra en la siguiente figura:

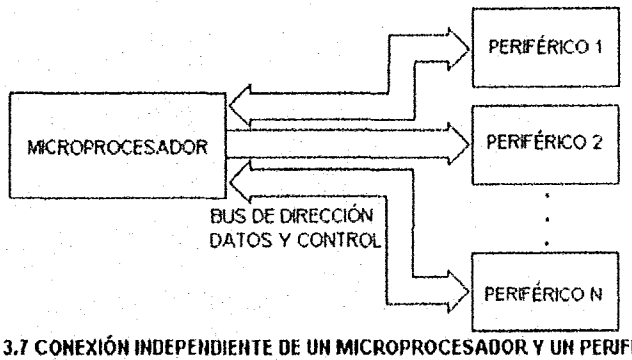


Fig. 3.7 CONEXIÓN INDEPENDIENTE DE UN MICROPROCESADOR Y UN PERIFÉRICO

La complejidad de este tipo de conexión radica en el número de dispositivos periféricos elevados, debido sobre todo al incremento del número de líneas requeridas. Este tipo de conexión es adecuado cuando el microprocesador puede enviar información simultánea a varios dispositivos periféricos (procesadores grandes y complejos).

2) **Utilizando una Conexión Común a Todos los Dispositivos Periféricos:** A través de la cual envía en secuencia la información correspondiente a varios de estos dispositivos periféricos. Esta forma de conexión permite reducir la complejidad del sistema, pero disminuye la velocidad de intercambio de información entre el microprocesador y los dispositivos periféricos (figura 3.8).

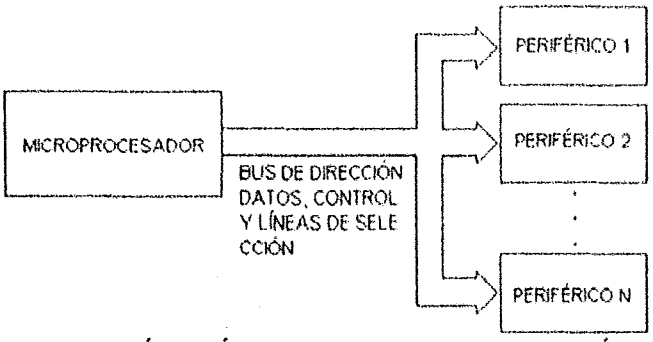


Fig. 3.8 CONEXIÓN COMÚN DE UN MICROPROCESADOR Y UN PERIFÉRICO

A partir de lo anteriormente explicado surge el concepto de **Selección**; esto es, además de la información, dirección y control, el microprocesador debe enviar a los dispositivos periféricos una señal de **Selección** para indicar el dispositivo periférico con el cual desea establecer comunicación. Estas señales de **Selección** pueden generarse a través de varias formas:

a) **El Microprocesador Genera una Señal de Selección Para Cada dispositivo Periférico**: Esto es, si existe un número n de dispositivos periféricos, el microprocesador genera n señales de las cuales se activa una a la vez, esta línea activada corresponde al dispositivo periférico deseado. Esta forma sólo es práctica si el número de dispositivos periféricos es reducido (figura 3.9).

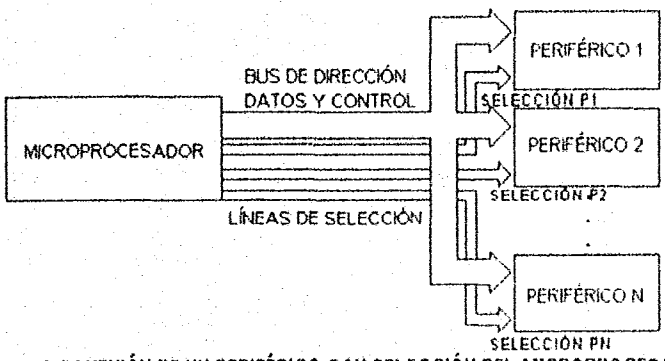


Fig. 3.9 CONEXIÓN DE UN PERIFÉRICO CON SELECCIÓN DEL MICROPROCESADOR

b) **El Microprocesador Genera una Combinación Binaria de Selección Diferente Para Cada Dispositivo Periférico**: Éstas señales son enviadas por terminales comunes a todos. Existen dos formas de combinación binaria de **Selección** desde el microprocesador central a los dispositivos periféricos:

1) A través de terminales independientes a las de información, tal como se puede apreciar en la figura número 3.10.

En este tipo de uso de línea de **Selección**, el microprocesador activa simultáneamente los datos y la línea de **Selección**, esto eleva los números de terminales, sin embargo incrementa la velocidad de transferencia de datos entre el microprocesador y los dispositivos periféricos.

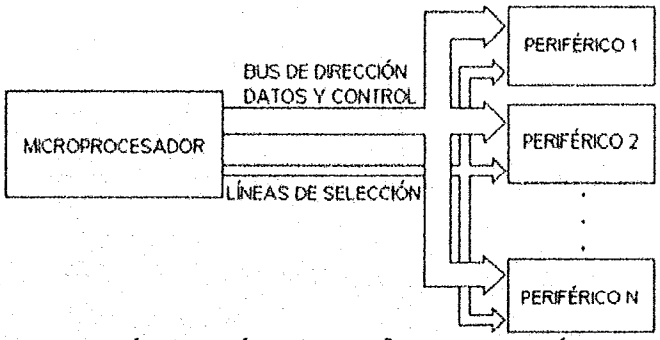


Fig. 3.10 CONEXIÓN DE PERIFÉRICOS CON SEÑALES DE SELECCIÓN COMUNES Y TERMINALES INDEPENDIENTES

2) Otra forma de transmisión de la línea de **Selección** consiste en utilizar los mismos terminales que la información y disponer de un terminal adicional cuyo estado lógico indica al periférico cuál es el tipo de combinación presente en los terminales comunes. Un ejemplo de tal configuración se puede apreciar en la figura número 3.11.

La señal **I/S** en un nivel alto indica que la combinación presente en los terminales es la información que se envía a los dispositivos periféricos, en tanto que en un nivel bajo indica que la combinación presente es la de **Selección** de dispositivo periférico.

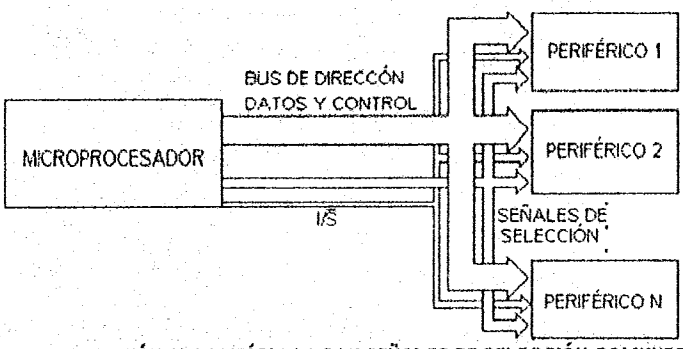


Fig. 3.11 CONEXIÓN DE PERIFÉRICOS CON SEÑALES DE SELECCIÓN COMUNES, TERMINALES INDEPENDIENTES Y SEÑAL DE ESTADO

Para la transmisión de datos del microprocesador hacia los dispositivos periféricos se puede utilizar el formato serie o paralelo:

1) **Serie**: Este tipo de transmisión de datos se logra a través de una secuencia de instrucciones o programa de serialización (división del tamaño de la palabra de datos en bits individuales, éstos se almacenan en buffers o registros para facilitar la manipulación individual de cada bit)

que transmite el contenido de un registro del microprocesador. Es conveniente que el propio microprocesador posea una terminal de entrada y salida de datos en formato serie. De esta forma la conexión del microprocesador con el periférico se reduce a dos líneas. En forma análoga, el periférico debe entregar y recibir la información en el mismo formato que el microprocesador (Fig. 3.12).

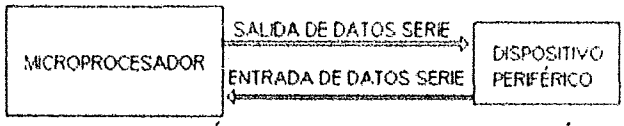


Fig. 3.12 COMUNICACIÓN SERIAL MICROPROCESADOR-PERIFÉRICO

Si el microprocesador no cuenta con líneas dedicadas a ello, se puede utilizar un dispositivo transmisor-receptor serie-paralelo conectado a éste. El microprocesador envía los datos en paralelo al transmisor, el cual se encarga de convertirlos en formato serie y de transmitirlos al periférico.

La recepción en serie la realiza el mismo dispositivo transmisor-receptor y cuando ha recibido un byte completo de información, lo convierte en formato paralelo emitiendo una señal al microprocesador del dato disponible, este último lee el dato y realiza procesos con ella, posteriormente borra el dato del transmisor-receptor (Fig. 3.13).

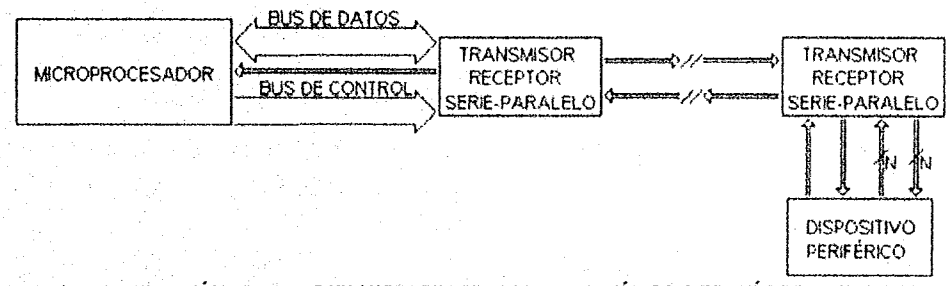


Fig. 3.13 COMUNICACIÓN SERIAL DE UN MICROPROCESADOR Y PERIFÉRICO A TRAVÉS DE UN TRANSMISOR RECEPTOR SERIE-PARALELO

2) **Paralelo**: El bus de datos es el más utilizado para intercambiar información entre un microprocesador con sus dispositivos periféricos. Suelen utilizarse como elementos intermedios (interfaz) dispositivos de tres estados con entradas de habilitación (Buffers). Cuando estos dispositivos triestado no están habilitados, permanecen en un estado de alta impedancia liberando al bus para que pueda ser utilizado por otro dispositivo periférico o por el mismo microprocesador. Cuando se desea activar el dispositivo, el microprocesador genera las señales de control adecuadas permitiendo al dispositivo periférico colocar los datos en el **Bus de Datos** como se puede apreciar en la figura número 3.14.

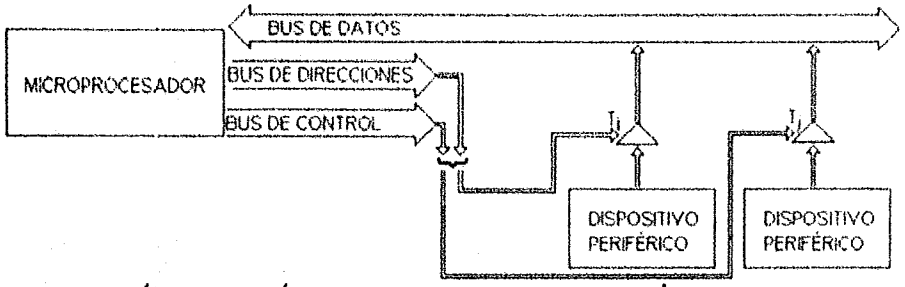


Fig. 3.14 CONEXIÓN DE UN PERIFÉRICO A UN MICROPROCESADOR A TRAVÉS DE UN BUFFER TRI-ESTADO

Cuando el microprocesador desea enviar los datos, coloca éstos en el **Bus de Datos** y genera las **Señales de Control** adecuadas. Para que los datos no se pierdan si el dispositivo periférico es lento, se utiliza un registro que almacena la información manteniéndolos presente en las entradas del dispositivo periférico, como se puede ver en la siguiente figura:

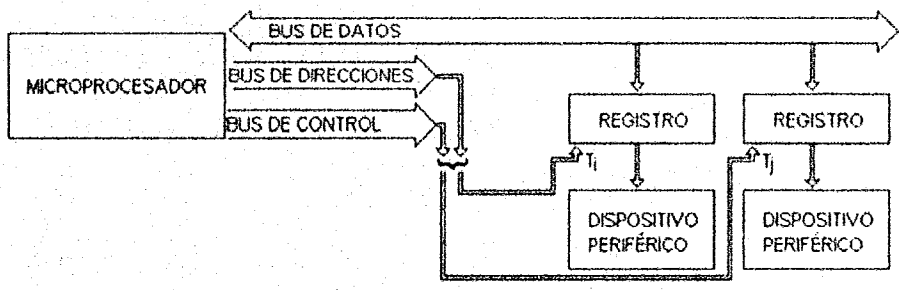


Fig: 3.15 CONEXIÓN DE UN PERIFÉRICO A UN MICROPROCESADOR A TRAVÉS DE UN REGISTRO

La señal de registro T_n se activa cuando el microprocesador ejecuta una instrucción de salida de información hacia el dispositivo periférico (combinando en forma adecuada las señales de control).

Se debe tener siempre presente que cuando el microprocesador realiza una instrucción de transferencia de información con la memoria, no hace lo propio con el dispositivo periférico y viceversa.

Las señales de control tienen las funciones de:

a) Diferenciar entre el intercambio de información con la memoria y con los dispositivos periféricos.
b) Diferenciar entre el intercambio de información desde el microprocesador hacia los dispositivos periféricos (incluyendo memorias) y viceversa.
c) Generar las **Señales de Control** adecuadas mientras la información se encuentra en el **Bus de Datos**.

Por la forma de controlar el intercambio de información con la memoria y los dispositivos periféricos, ésta puede clasificarse en dos formas:

1) **Intercambio de Información Controlada por el Microprocesador**. El intercambio de información se realiza mediante la ejecución de instrucciones del microprocesador. Dentro de este rubro, se ubican los siguientes métodos:

   a) **Intercambio de Información Iniciada por el Microprocesador**: Se puede realizar en forma síncrona cuando el dispositivo periférico no envía ninguna señal de control al microprocesador. Para este tipo de intercambio de información el dispositivo periférico procesa la información o la coloca en el **Bus de Datos** en el mismo instante de tiempo que el microprocesador tarda en ejecutar una instrucción de entrada o de salida.

   Si el dispositivo periférico es más lento que el microprocesador, se utiliza el intercambio de información con **Detención del Proceso** actual. Esto se realiza haciendo que el microprocesador deje de ejecutar el proceso actual y espere a que el dispositivo periférico coloque o lea el dato desde el **Bus de Datos**. Esta forma es viable sólo cuando el proceso en ejecución es sencillo y el microprocesador puede estar inactivo mientras espera la respuesta del dispositivo periférico.

   Una forma de lograr la **Detención del Proceso** es por programa, ejecutando ciclos de retado de duración adecuados. Esta técnica es adecuada con dispositivos periféricos de salida lenta tales como las impresoras.

   El ciclo de retardo debe ser superior al tiempo que el dispositivo periférico tarda en aceptar el dato. Sin embargo presenta el inconveniente de hacer lento el intercambio de información aún cuando el tiempo que tarda en aceptar los datos el dispositivo periférico es variable.

   Otra forma de detener el programa es por activación de la **Línea de Paro**. Esta línea va unida a la **Unidad de Control** del microprocesador, siendo su forma de acción como la siguiente: en el nivel lógico inactivo de la **Línea de Paro**, el microprocesador ejecuta el programa en secuencia; cuando cambia el estado lógico hacia activo de la **Línea de Paro**, la **Unidad de Control** termina el proceso en curso y se detiene hasta que la **Línea de Paro** vuelve al estado inactivo. Esta técnica requiere de un dispositivo biestable (flip-flop) en la **Unidad de Interfaz**. Cuando el microprocesador ejecuta la instrucción de intercambio de información, activa el biestable (flip-flop) el cual se encuentra unida a la **Línea de Paro**. Adicionalmente, la instrucción activa la **Unidad de Control** del dispositivo periférico para que realice las operaciones requeridas. Al terminar de procesar la información, el dispositivo periférico genera una señal que hace cambiar de estado el biestable (flip-flop), permitiendo que el microprocesador reanude la ejecución del programa.

   Otra forma de intercambiar la información entre el microprocesador y los dispositivos periféricos con **Consulta Periódica**, esto evita las limitaciones de la **Detención del Programa**.

[illegible]

3) **Intercambio de información iniciada por el Dispositivo Periférico.** Este método evita que el microprocesador y el dispositivo periférico se mantengan en un ciclo de espera.

Para este tipo de intercambio de información el microprocesador debe contar con dos terminales: **Líneas de interrupción**, en donde los dispositivos periféricos le harán saber que desean o tienen una acción pendiente en su interfaz.

1) Al encontrarse en el estado inactivo, la **Unidad de Control** analiza el proceso en proceso buscando y ejecutando la siguiente instrucción.

2) Si se encuentra activa, la **Unidad de Control** deja de ejecutar las instrucciones en proceso actual y atiende al dispositivo periférico que desea el acceso activo de la **Terminal de Interrupción**.

Ante estas circunstancias, existen dos formas de identificar a quien dar servicio:

1) **Varias Terminales de Interrupción.** Este método requiere la disponibilidad de acceder un número elevado de terminales de microprocesador en relación con el número total de dispositivos periféricos.

2) **Terminal Única de Interrupción.** Si más de un dispositivo periférico puede generar una interrupción al microprocesador y existe sólo una terminal para recibirlo, el microprocesador debe poseer alguna forma de identificar el dispositivo periférico que efectuó la **Petición de Interrupción**. Cuando el microprocesador acepta la interrupción, debe comunicárselo al dispositivo periférico, dado lo cual el microprocesador debe poseer dos terminales o métodos que provean las interrupciones.

   a) **Terminal de Reconocimiento de Interrupción.** Cuando el microprocesador ha reconocido una interrupción, también debe tener un método para identificar al dispositivo periférico que solicitó servicio, esto se realiza mediante alguna de las siguientes acciones:

      1. **Identificación por Programa**. En esta forma de identificación, el microprocesador no posee ningún circuito especial para identificar al dispositivo periférico que generó la señal de **Petición de Interrupción** e incluso podría no requerir de las señales de **Reconocimiento de Interrupción**.

[illegible]

D) **Intercambio de información iniciada por el Dispositivo Periférico**. Este modo puede ser [illegible]

Para este tipo de intercambio de información el microprocesador debe tener dos líneas especiales: **Línea de interrupción**, las cuales son consultadas por el [illegible] cada ciclo de ejecución y tomar una acción basado en su estado.

I) Al encontrarse con un estado inactivo, la **Unidad de Control** continua el proceso de proceso buscando y ejecutando la siguiente instrucción.

II) Si se encuentran activas, la **Unidad de Control** deja de buscar las instrucciones del proceso actual y atiende al dispositivo periférico que generó la señal activa de la **Terminal de Interrupción**.

Ante estas circunstancias, existen dos formas de operación y atención con periféricos.

1) **Varias Terminales de Interrupción**. Este método presenta la desventaja de necesitar un número elevado de terminales del microprocesador en relación con el número total de dispositivos periféricos.

2) **Terminal Única de Interrupción**. Si más de un dispositivo periférico puede enviar una interrupción al microprocesador y existe sólo una terminal para [illegible] al microprocesador, debe existir alguna forma de identificar el dispositivo periférico que efectuó la **Petición de Interrupción**. Cuando el microprocesador acepta la interrupción, debe comunicársele al dispositivo periférico, dado lo cual el microprocesador debe poseer dos terminales o métodos que procesen las interrupciones:

   a) **Terminal de Reconocimiento de Interrupción**. Cuando el microprocesador ha reconocido una interrupción, también debe tener un método para identificar al dispositivo periférico que solicitó servicio, esto se realiza mediante alguna de las siguientes acciones:

      1. **Identificación por Programa**. En esta forma de identificación, el microprocesador no posee ningún circuito especial para identificar al dispositivo periférico que genero la señal de **Petición de Interrupción** e incluso podría no requerir de las señales de **Reconocimiento de Interrupción**.

Para ello se dota a la **Unidad de Interfaz** de un dispositivo biestable (flip-flop) de estado interno que puede ser consultado por el microprocesador. Cuando el dispositivo periférico se encuentra preparado para realizar un intercambio de información, se lo informa al microprocesador colocando el biestable (flip-flop) en un estado activo. El microprocesador consulta el estado de dicho biestable y realiza las acciones adecuadas. En la práctica esto sólo es posible cuando la ejecución del programa completo es inferior al tiempo máximo admisible entre dos transferencias de datos.

b) **Intercambio de Información Iniciada por el Dispositivo Periférico**: Esta técnica permite una gran flexibilidad a la par que evita las limitaciones de la **Consulta Periódica**.

Para este tipo de intercambio de información el microprocesador debe contar con varias terminales (**Líneas de Interrupción**), las cuales son consultadas en el último estado del ciclo de ejecución y tomar una acción basado en su estado:

1) Al encontrarse con un estado inactivo, la **Unidad de Control** continúa la ejecución del proceso buscando y ejecutando la siguiente instrucción.
2) Si se encuentran activos, la **Unidad de Control** deja de buscar las instrucciones del proceso actual y atiende al dispositivo periférico que provoco el estado activo de la **Terminal de Interrupción**.

Ante estas circunstancias, existen dos formas de demandar y atender una interrupción:

1) **Varias Terminales de Interrupción**: Este método presenta la desventaja de necesitar un número elevado de terminales del microprocesador en relación con el número total de dispositivos periféricos.
2) **Terminal Única de Interrupción**: Si más de un dispositivo periférico puede efectuar una interrupción al microprocesador y existe solo una terminal para indicárselo al microprocesador, debe existir alguna forma de identificar el dispositivo periférico que efectuó la **Petición de Interrupción**. Cuando el microprocesador acepta la interrupción, debe comunicárselo al dispositivo periférico, dado lo cual el microprocesador debe poseer dos terminales o métodos que procesen las interrupciones:
   a) **Terminal de Reconocimiento de Interrupción**: Cuando el microprocesador ha reconocido una interrupción, también debe tener un método para identificar al dispositivo periférico que solicitó servicio, esto se realiza mediante alguna de las siguientes acciones:
      1. **Identificación por Programa**: En esta forma de identificación, el microprocesador no posee ningún circuito especial para identificar al dispositivo periférico que genero la señal de **Petición de Interrupción** e incluso podría no requerir de las señales de **Reconocimiento de Interrupción**.

Cuando se activa la señal de **Petición de Interrupción**, se introduce en el registro **IR** una instrucción cuyo código de operación es de salto a la subrutina y cuyo campo de dirección es una combinación binaria correspondiente a una posición determinada de memoria (puede ser las primeras o las últimas posiciones). Para este caso, contamos con dos opciones: la primera es colocando la **Subrutina de Interrupción** a partir de la posición actual y la segunda es colocando en la posición de memoria actual una instrucción de salto con direccionamiento indirecto y colocar la **Subrutina de Interrupción** a partir de la dirección indirecta indicada en la instrucción.

2. Colocando en el interior del microprocesador circuitos que permitan introducir en un registro un vector de identificación externo. Este vector puede ser introducido a través del mismo **Bus de Datos**.

   Al reconocer la **Petición de Interrupción**, el microprocesador genera una señal de reconocimiento que se aplica a la interfaz, ocasionando que éste coloque el vector asignado al dispositivo periférico (Fig. 3.16).

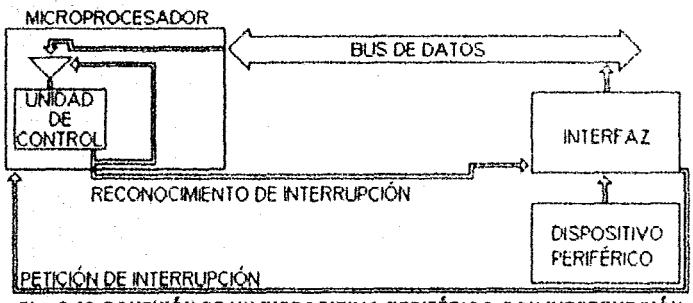


Fig. 3.16 CONEXIÓN DE UN DISPOSITIVO PERIFÉRICO CON INTERRUPCIÓN

Al existir diversos dispositivos periféricos debe asignar un nivel de prioridad a cada uno de ellos. De acuerdo con los siguientes criterios:

a) Se debe tomar siempre en cuenta que al momento de realizar una interrupción, los dispositivos periféricos no admiten que transcurra más de un tiempo *t* hasta el momento de recibir atención, tal como suele suceder con los convertidores analógico-digitales, el tiempo máximo entre la emisión de **Petición de Interrupción** y su reconocimiento no debe ser superior al tiempo de **conversión** *t*.

b) **Procesos en Tiempo Real**: El microprocesador atiende a un proceso físico elaborando respuestas en un tiempo máximo, unos dispositivos periféricos tienen mayor prioridad de atención que otros. Un ejemplo podría ser un sistema que controla una alarma contra incendios, si ocurriera algún conato de incendio, el sistema proporciona la mayor prioridad a este suceso, relegando a segundo término otras interrupciones.

La prioridad de las interrupciones puede ser de varios tipos:

1) **Fijo**: La prioridad de las interrupciones permanece fija a lo largo de la ejecución del programa Las prioridades fijas pueden ser realizadas mediante:
   a) **Programa**: El microprocesador consulta el estado de los biestables (flip-flops) de **Petición de Interrupción** de los dispositivos periféricos en un orden determinado.
   b) **Circuito de Prioridad**: Se usa en el caso de que los dispositivos periféricos se identifiquen mediante un **Vector de Interrupción**. El circuito se realiza de tal forma que cuando más de un dispositivo periférico emite una interrupción simultáneamente, sólo aquel que posee una mayor prioridad coloca su **Vector de Interrupción** en el **Bus de Datos** al momento de ser reconocida la interrupción.

      Existen dos forma de realizar el circuito para una **Prioridad de Interrupciones Fija**:

      1. **Circuito de Prioridad Fija no Modular**: Este circuito es común a todos los dispositivos periféricos y posee una entrada para cada uno de ellos. Este circuito se muestra en la siguiente figura:

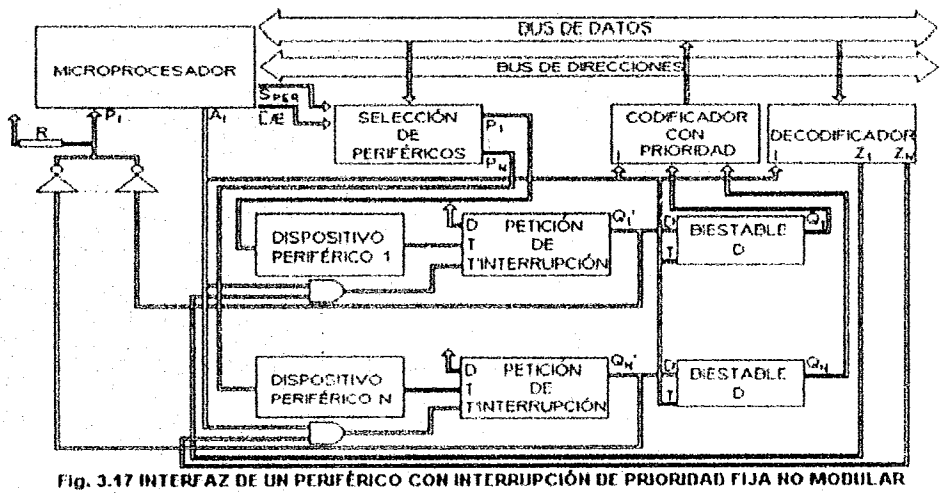


Fig. 3.17 INTERFAZ DE UN PERIFÉRICO CON INTERRUPCIÓN DE PRIORIDAD FIJA NO MODULAR

Consiste esencialmente de un codificador con prioridad y un decodificador. Cada dispositivo periférico se conecta a un **Biestable (Flip-Flop) de Interrupción** la cual se activa al aplicársele un flanco activo a su entrada T.

Las salidas Q' de los dispositivos de **Petición de Interrupción** se conectan a través de un inversor de colector abierto a la entrada P_i de **Petición de Interrupción**. También se conectan las salidas de éstas a las entradas de los biestables (flip-flops) tipo D cuyas entradas T están controladas por las señales de **Reconocimiento de Interrupción** A_i. Cuando la señal A_i no está activada, T permite que la salida Q del biestable (flip-flop) tenga el estado de su entrada D. Al aceptar la interrupción, el microprocesador activa la señal A_i y los biestables (flip-flops) almacenan el estado de la variable D. Las salidas de los biestables (fli-flops) se conectan a las entradas de un codificador con prioridad que

La prioridad de las interrupciones puede ser de varios tipos:

1) **Fijo**: La prioridad de las interrupciones permanece fija a lo largo de la ejecución del programa. Las prioridades fijas pueden ser realizadas mediante:
   a) **Programa**: El microprocesador consulta el estado de los biestables (flip-flops) de **Petición de Interrupción** de los dispositivos periféricos en un orden determinado.
   b) **Circuito de Prioridad**: Se usa en el caso de que los dispositivos periféricos se identifiquen mediante un **Vector de Interrupción**. El circuito se realiza de tal forma que cuando más de un dispositivo periférico emite una interrupción simultáneamente, sólo aquel que posee una mayor prioridad coloca su **Vector de Interrupción** en el **Bus de Datos** al momento de ser reconocida la interrupción.

      Existen dos forma de realizar el circuito para una **Prioridad de Interrupciones Fija**:

      1. **Circuito de Prioridad Fija no Modular**: Este circuito es común a todos los dispositivos periféricos y posee una entrada para cada uno de ellos. Este circuito se muestra en la siguiente figura:

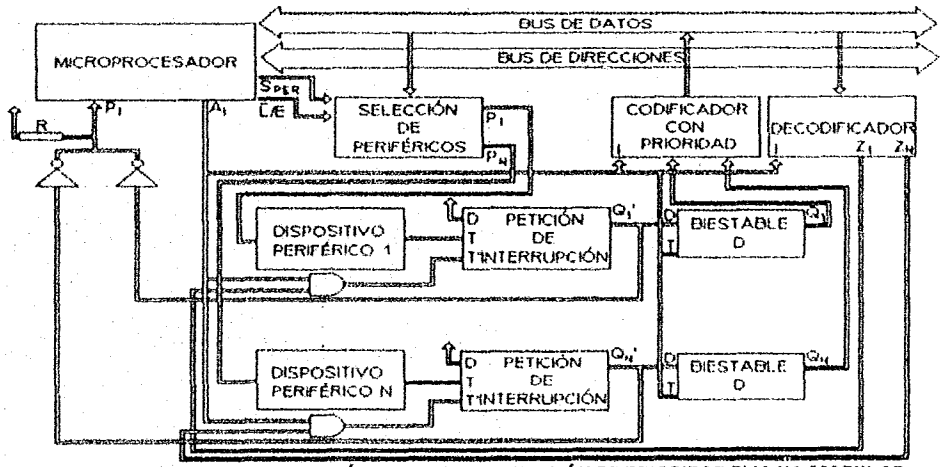


Fig. 3.17 INTERFAZ DE UN PERIFÉRICO CON INTERRUPCIÓN DE PRIORIDAD FIJA NO MODULAR

Consiste esencialmente de un codificador con prioridad y un decodificador. Cada dispositivo periférico se conecta a un **Biestable (Flip-Flop) de Interrupción** la cual se activa al aplicársele un flanco activo a su entrada T.

Las salidas Q' de los dispositivos de **Petición de Interrupción** se conectan a través de un inversor de colector abierto a la entrada P_i de **Petición de Interrupción**. También se conectan las salidas de éstas a las entradas de los biestables (flip-flops) tipo D cuyas entradas T están controladas por las señales de **Reconocimiento de Interrupción** A_i. Cuando la señal A_i no está activada, T permite que la salida Q del biestable (flip-flop) tenga el estado de su entrada D. Al aceptar la interrupción, el microprocesador activa la señal A_i y los biestables (flip-flops) almacenan el estado de la variable D. Las salidas de los biestables (fli-flops) se conectan a las entradas de un codificador con prioridad que

posee salidas de tres estados conectados al **Bus de Datos**. La línea A_i está conectada a la entrada de **Inhibición I** del **Decodificador de Prioridad**. Mientras esta línea esté desactivada, las salidas del codificador se encuentran en alta impedancia, cuando la línea A_i se activa, aparece la combinación binaria a la salida del codificador, el cual corresponde a la entrada de mayor peso del dispositivo periférico que ha generado una **Petición de Interrupción**.

Esta combinación binaria constituye el **Vector de Interrupción** para este dispositivo periférico, el cual aparece en el **Bus de Datos** y es aceptada por el microprocesador. El microprocesador calcula a partir estos datos la dirección de la subrutina que brinda servicios al dispositivo periférico que emitió la interrupción.

Al atender el microprocesador la interrupción, desactiva el circuito de **Petición de Interrupción** correspondiente al dispositivo periférico que acaba de ser atendido.

En la figura anterior, esta acción se realiza gracias a un decodificador cuyas entradas de datos están conectadas al **Bus de Datos** y cada una de sus salidas están conectadas a través de circuitos lógicos **AND** a la entrada T' de un dispositivo de **Petición de Interrupción**.

2. **Prioridad Fija Modular**: Se basa en que la **Señal de Interrupción** de cada dispositivo periférico sólo debe ser aceptada si no existe ninguna de prioridad superior a la que está solicitando atención. Debido a esto, el circuito de cada dispositivo periférico debe recibir una señal del dispositivo periférico anterior y emitir otra al posterior como se puede apreciar en la siguiente figura:

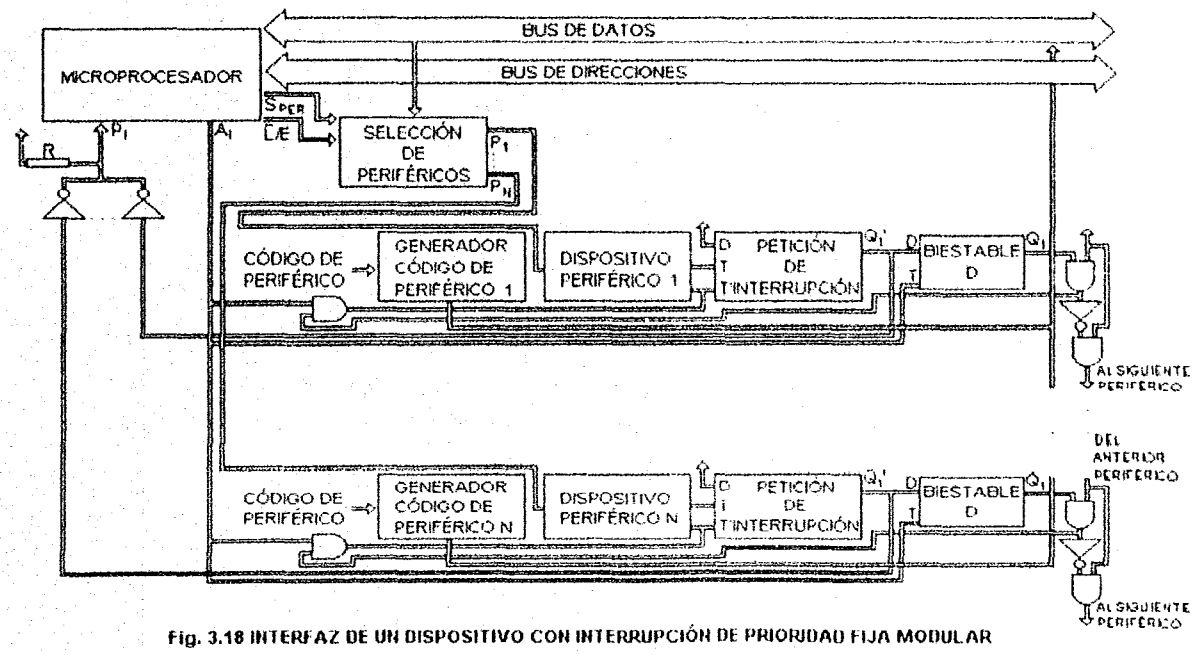


Fig. 3.18 INTERFAZ DE UN DISPOSITIVO CON INTERRUPCIÓN DE PRIORIDAD FIJA MODULAR

2) **Variable**: Esta forma de manejar la prioridad de las interrupciones es de mayor utilidad en microprocesadores que trabajan en tiempo real. La forma de realizar esto es mediante:

a) **Programa**: El circuito para este método puede ser como el que se muestra en la siguiente figura:

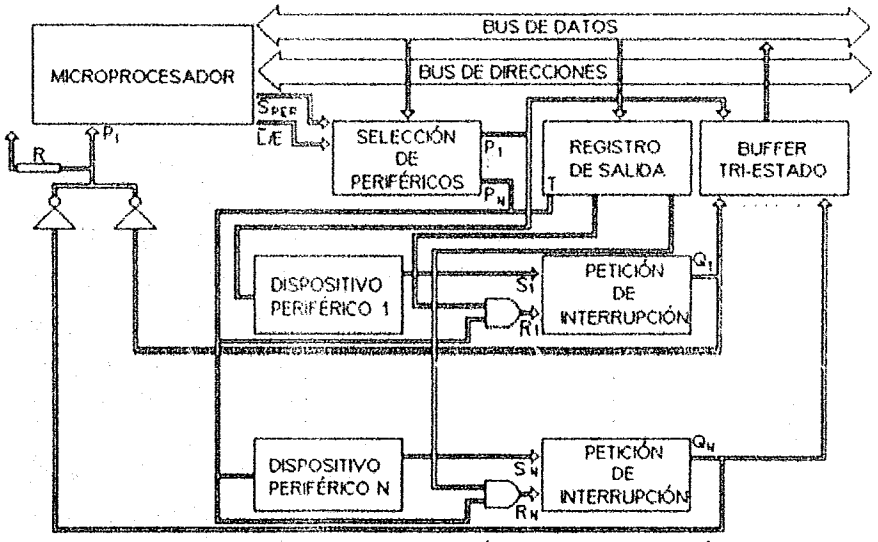


Fig. 3.19 INTERFAZ DE UN DISPOSITIVO PERIFÉRICO CON INTERRUPCIÓN VARIABLE

Se dedica un número **n** de bits de una posición de memoria o de un registro para almacenar 2^n combinaciones binarias asignadas a otras tantas **prioridades de interrupción** distintas. En cada parte del programa que se desea tener una determinada prioridad, se coloca en éste el lugar destinado a la combinación citada.

b) **Circuito de Prioridad**: Consiste en su parte más esencial por un **Circuito Combinacional Programable (ROM, PROM o EPROM)** o una **Matriz Lógica Programable**. Este circuito recibe las **Señales de Interrupción** de cada dispositivo periférico a través de un biestable (flip-flop) tipo D. Según el estado de las **Líneas de Interrupción** este circuito dará a la salida una respuesta distinta para los mismos estados de las demás **Líneas de Interrupción**. La entrada de este biestable (flip-flop) se conecta a una línea del **Bus de Datos**. Se puede cambiar la prioridad colocando en los lugares adecuados del programa una instrucción de salida al dispositivo periférico de un registro cuya línea correspondiente del **Bus de Datos** está unido a la entrada del biestable (flip-flop) tipo D, para modificar esta prioridad se coloca un 1 ó 0 según la prioridad deseada.

Dado que pueden existir procesos que requieran realizar por completo una subrutina antes de atender a otra interrupción que se produzca durante la ejecución de los procesos de la misma. De igual forma, si dos dispositivos periféricos de diferente prioridad interrumpen simultáneamente y se atiende al de mayor prioridad, es necesario impedir que se atienda al otro dispositivo periférico hasta que termine la subrutina actual. De esta forma debe existir algún método para inhibir las interrupciones, las cuales pueden ser por:

1) **Inhibición Global de las Interrupciones**: Esto ocurre cuando cualquier dispositivo periférico puede emitir una interrupción. Para deshabilitar estas interrupciones, se requiere de hardware y software. El hardware consiste de un biestable (flip-flop) situado en el interior del microprocesador, el cual al encontrarse en un determinado estado impide que la señal de **Petición de Interrupción** sea tomada en cuenta o muestreada por el microprocesador. Como se puede apreciar en la siguiente figura:

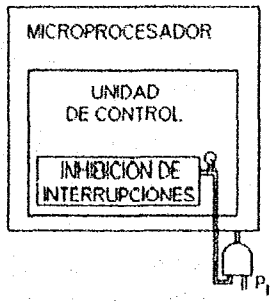


Fig. 3.20 INHIBICIÓN DE INTERRUPCIÓN

Para poder activar este biestable (flip-flop), el microprocesador debe poseer instrucciones de software adecuadas. Para este caso existen dos alternativas:

a) El biestable (flip-flop) de **Inhibición de Interrupción** no está asociado a ningún otro. El juego de instrucciones debe poseer una instrucción adecuada para ponerlo en modo activo y no activo.

b) El biestable está agrupado con otros, constituyendo un registro o palabra de estado (**Flag**), como puede apreciarse en la siguiente figura:

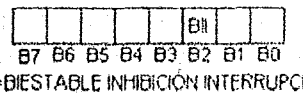


Fig. 3.21 FLAG DE ESTADO DE UN MICROPROCESADOR

Para actuar sobre los biestables (flip-flops) del **Flag**, el microprocesador debe poseer instrucciones para activar y desactivar dicho byte. Este método ofrece nula flexibilidad porque sólo permite la inhibición de todas las interrupciones generadas por los dispositivos periféricos o de ninguno.

2) **Inhibición Individual de las Interrupciones**: Esto se logra colocando un biestable (flip-flop) destinado a esta función en la interfaz del dispositivo periférico. El biestable (flip-flop) de **Inhibición de Interrupción** individual se agrupa formando un registro (**Registro de Máscara**), constituyendo un periférico del microprocesador, como se puede apreciar en la siguiente figura:

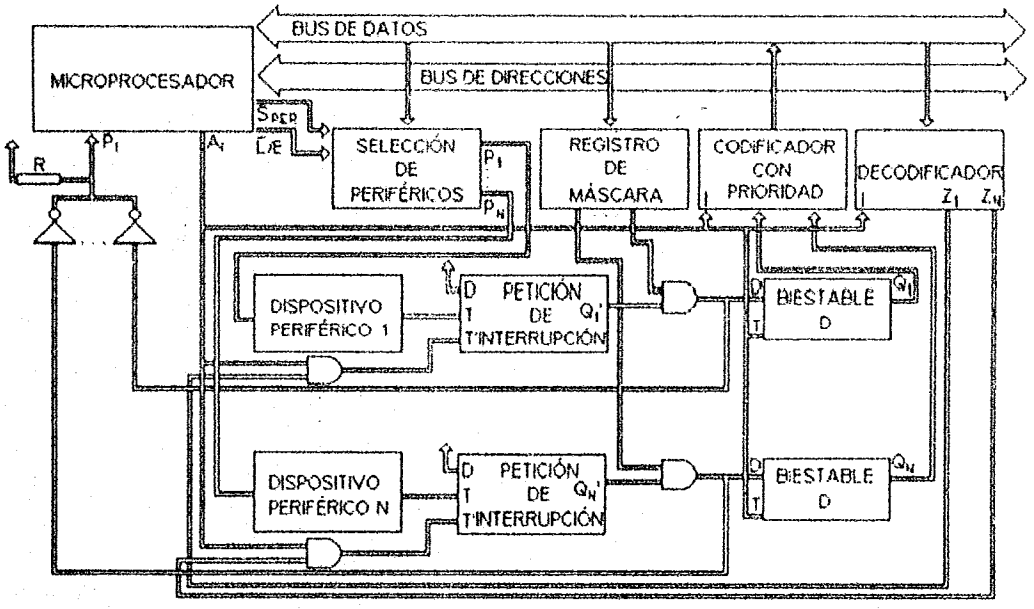


Fig. 3.22 INTERFAZ DE UN DISPOSITIVO PERIFÉRICO CON INTERRUPCIÓN E INHIBICIÓN INDIVIDUAL

La salida de este dispositivo se conecta a una compuerta lógica **AND**, cuya línea adicional está conectada al dispositivo de **Petición de Interrupción**. Esta compuerta lógica **AND** se conecta a la entrada del biestable (flip-flop) tipo D y a la entrada P_I del microprocesador. De esta forma la interrupción generada por un dispositivo periférico no es detectada por el microprocesador hasta que el **Bit de Inhibición** correspondiente del **Registro Máscara** se encuentre activado.

Otra forma mediante el cual se efectúa el intercambio de información entre los dispositivos periféricos y el microprocesador o la memoria es el denominado **Acceso Directo a Memoria (Direct Access Memory o DMA)**. En ocasiones suele ser necesario transferir una gran cantidad de información entre un dispositivo periférico y la memora **RAM**. Un posible ejemplo es la carga de un programa de aplicación general (por ejemplo un traductor) entre un disco magnético y una computadora. En este caso es necesario disminuir al máximo el tiempo que tarda en transferir toda la información, aún cuando con ello se incremente la complejidad de la unidad de interfaz.

En una interfaz DMA el dispositivo periférico se conecta directamente con la memoria sin que el microprocesador ejecute las instrucciones de transferencia de datos. Para esto es necesario que el microprocesador y los dispositivos periféricos se conecten con los mismos buses, como puede apreciarse en la siguiente figura:

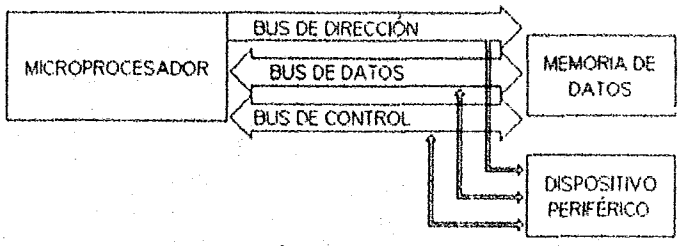


Fig. 3.23 INTERFAZ DE PERIFÉRICO CON ACCESO DIRECTO A MEMORIA

El método de **Acceso Directo a Memoria (DMA)** puede ser iniciado por el microprocesador o por el periférico.

Por las posiciones de memoria con las cuales se realiza la transferencia de información, la interfaz con **Acceso Directo a Memoria (DMA)** puede ser clasificada por:

1) **Posiciones de Memoria Fija**: El dispositivo periférico intercambia información con la memoria **RAM** en unas posiciones de memoria fijas y repetitivas.

   Para este caso, la **Unidad de Control** de **Acceso Directo a Memoria** es simple, como ejemplo podemos citar la adquisición de un número elevado de datos procedentes de un proceso industrial y procesarlos con una secuencia adecuada de instrucciones.

2) **Posiciones de Memoria Variable**: El dispositivo periférico intercambia información con la memoria **RAM** en diferentes posiciones de la misma. En este caso, generalmente el microprocesador decide iniciar el **Acceso Directo a Memoria** cuando encuentra la instrucción adecuada durante la ejecución del programa. La **Unidad de Control** de **Acceso Directo a Memoria** es programable y antes de iniciar la transferencia, el microprocesador le indica la posición de memoria a partir de la cual deben realizarse la transferencia y el número de posiciones (cantidad de datos a transferir) o la posición inicial y final.

Por la forma en que se realiza la **Unidad de Control** de DMA, estas pueden clasificarse en:

1) **Dedicación Exclusiva**: Durante el tiempo que el dispositivo periférico accede a la memoria, el microprocesador no puede hacer lo propio, por tanto, no ejecuta proceso alguno. Este método es adecuado cuando se desea realizar la transferencia de datos lo más rápido posible y se puede detener el proceso actual durante todo el tiempo que dure la trasferencia.

2) **Robo de Ciclo**: Mientras el microprocesador no accede a la memoria, puede hacerlo la unidad de control del dispositivo periférico conectado por DMA. De esta forma se combina el **Acceso Directo a la Memoria** entre el microprocesador y el dispositivo periférico sin disminuir la velocidad de acceso.

Otra forma de lograr el **Robo de Ciclo** es multiplexando el microprocesador y el dispositivo periférico y realizar sistemáticamente un acceso por parte del microprocesador y otra por parte del dispositivo periférico. Un ejemplo de dicho circuito se muestra a continuación.

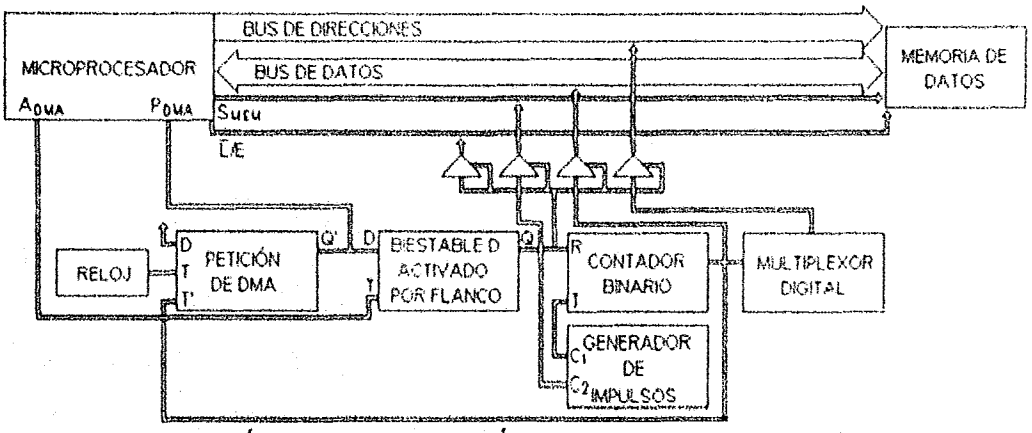


Fig. 3.24 CONEXIÓN DE UN DISPOSITIVO PERIFÉRICO CON DMA Y POSICIONES FIJAS DE MEMORIA

En el circuito de la gráfica anterior, se dispone de un dispositivo de **Petición de DMA** cuya entrada D se activa a un nivel lógico alto y la entrada T a un reloj generador de pulsos que genera un flanco activo cada cierto intervalo de tiempo. Su salida Q' se conecta a la entrada P_{DMA} (activo a un nivel lógico alto) del microprocesador. Cuando el microprocesador detecta la señal, termina la ejecución de la instrucción en curso y genera una señal A_{DMA} de **Reconocimiento de DMA**, poniendo en alta impedancia su **Bus de Datos**, **Dirección** y **Control**. Esta señal se aplica a la entrada T de un biestable (flip-flop) tipo D activado por flancos, cuya entrada D está unida al circuito de **Petición de DMA**. De esta forma, al producirse dicho impulso el biestable (flip-flop) pasa al estado activo y hace desaparecer la puesta a cero del contador binario, que comienza a contar los impulsos del generador. La entrada R del biestable (flip-flop) se une a la salida Q del dispositivo de **Petición de DMA** para asegurar su estado inactivo mientras no se produce la petición de **DMA**.

La salida Q del biestable (flip-flops) se conecta también a la entrada de inhibición de los dispositivos trisecado que unen los elementos de la interfaz a los buces del microprocesador. De esta forma se produce la aceptación de **DMA**, el estado de los buces del microprocesador se controla por la interfaz.

Cuando el circuito alcanza el final del último estado genera un flanco activo que se aplica a la entrada T' del dispositivo de **Petición de DMA** haciendo que vuelva al estado inactivo. De esta forma desaparece la señal de **Petición de DMA** y el microprocesador continúa la ejecución del programa donde lo dejó.

3.3 Mapeo de la Memoria del Sistema

Como se ha mencionado anteriormente, cualquier aplicación de un sistema digital basado en un microprocesador requiere la transferencia de datos entre circuitos externos al microprocesador y él mismo. La información acerca del mundo exterior debe ser reunida y procesada por el sistema, una vez procesada, los resultados deben ser mostrados y enviados a dispositivos periféricos para ser mostrados nuevamente al mundo exterior. Estas transferencias de información constituyen las operaciones de **Entrada/Salida(E/S)** o **Input/Output (I/O)**.

Dada la gran variedad de dispositivos periféricos que pueden ser conectados a un sistema digital y las características especiales, tanto eléctricas como funcionales de cada uno de ellos, la transferencia de información entre el microprocesador y el periférico no se efectúa de manera directa sino a través de otros elementos externos, los cuales reciben la información proveniente del microprocesador y la envían a los dispositivos de salida o recaban la información originada en los dispositivos de entrada y la transmiten al microprocesador. A estos elementos se les da el nombre de **puertos de Entrada/Salida**.

Los **puertos de entrada/salida** están formados básicamente a partir de registros externos. Algunos microprocesadores proporcionan **Señales de Control** que permiten que los registros externos que forman los **Puertos de E/S** ocupen un espacio de direcciones separado, es decir, distinto del espacio de direcciones de los registros externos que componen la memoria. Cuando los puertos tienen asignados un espacio de direcciones separado, se dice que están en modo de **Entrada/Salida Aislada** o **E/S Estándar**. Por el contrario, cuando se ubican dentro del mismo espacio de direcciones que la memoria, se dice que están en modo de **Entrada/Salida Mapeada** o **Entrada/Salida Proyectada en Memoria**.

Además, un **Puerto de E/S** contiene circuitos para el control de la transferencia de datos, así como para el acoplamiento (interfaz) con el dispositivo periférico externo. Es importante señalar que el microprocesador únicamente tiene relación con el **Puerto de Entrada/Salida**, no tiene comunicación directa con el dispositivo periférico; es el **Puerto de Entrada/Salida** el que tiene relación directa con el dispositivo periférico.

3.3.1 Puertos de Entrada/Salida Aislados

Para que un microprocesador pueda implementar el modo de **E/S Aislada (Isolated I/O)** son indispensables las siguientes condiciones:

1) El microprocesador debe proporcionar **Señales de Control** que permitan distinguir entre una operación con un **Puerto de Entrada/Salida** y una referencia a memoria.
2) El código de instrucciones debe tener instrucciones especiales con las que se pueda leer (entrada) o escribir (salida) datos en los **Puertos de Entrada/Salida**.

El microprocesador Z80 cumple con los requisitos anteriores por lo tanto, permite la interconexión de **Puertos de Entrada/Salida** en el modo de **E/S Aislada**.

Como normalmente un sistema no necesita un número muy grande de **Puertos de Entrada/Salida**, es común que no se utilicen todas las líneas del **Bus de Direcciones** del microprocesador en el acceso a **Puertos de Entrada/Salida Aislada**. En el caso del microprocesador Z80, éste usa únicamente las ocho líneas menos significativas del **Bus de Direcciones** (A_0-A_7) para direccionar un **Puerto de Entrada/Salida**. Con esto el sistema puede accesar un máximo de 256 direcciones asociadas con **Puertos de Entrada/Salida**.

El microprocesador Z80 tiene un poderoso grupo de instrucciones con las que puede realizar transferencias de datos con los **Puertos Entrada/Salida Aislada**. Estas instrucciones se pueden dividir en:

1) **Instrucciones de E/S con el Acumulador**: Las más simples de estas instrucciones son las de **Entrada /Salida** con el **Acumulador**. Sus **Mnemónicos** son:

   IN A, (N)y OUT(N), A

   Ambas instrucciones son de dos bytes; el primer byte especifica el código de operación (**IN=DBH** y **OUT=D3H**) y el segundo proporciona el número o dirección del **Puerto de Entrada/Salida Aislada**. El número del **Puerto de Entrada/Salida** se encuentra en el intervalo de 00H hasta FFH (0 a 255).

   La instrucción **IN A, (N)** ordena cargar en el **Acumulador** el contenido del **Puerto de Entrada/Salida** "N". Durante esta instrucción el número del **Puerto de Entrada/Salida** se envía por las líneas A_0-A_7 del **Bus de Direcciones** y en las líneas restantes (A_8-A_{15}) aparece el contenido del **Acumulador**.

   La instrucción **OUT (N), A** ordena transferir el contenido del **Acumulador** al **Puerto de Entrada/Salida** "N". Durante esta instrucción el número del **Puerto de Entrada/Salida** se envía por las líneas A_0-A_7 del **Bus de Direcciones**, mientras que en el **Bus de Datos** y en las líneas A_8-A_{15} del **Bus de Direcciones** aparece el contenido del **Acumulador**. La ejecución de cualquiera de estas instrucciones no afecta las banderas de estado (**Flag**) de la **CPU**.

   Por ejemplo, con el siguiente programa se lee el contenido del **Puerto de Entrada 8FH**, el dato leído se carga en la dirección de memoria **5000H** y se envía el mismo dato al **Puerto de Salida 21H**:

```
IN A, (8FH)          ; Acumulador = Dato del Puerto 8FH
LD (5000H), A        ; Localidad 5000H = Dato Leido
OUT (21H), A         ; Puerto 21H = Lato Leido
```

2) **Instrucciones de E/S Usando el Registro C**: Un **Puerto de Entrada** puede tener la misma dirección asignada que un **Puerto de Salida**, ya que su acceso está controlado por instrucciones distintas.

   Las instrucciones de **E/S** usando el registro **C** permiten la transferencia de datos entre cualquiera de los siete **Registros de Propósito General** y el **Puerto de Entrada/salida**

especificado en el registro C. Es decir, el registro C actúa como un apuntador que contiene la dirección del **Puerto de Entrada/Salida** asociada con la operación de E/S. Como el registro C es de un byte, el número del **Puerto de Entrada/Salida** se encuentra en el rango de **00H** hasta **FFH**. Las instrucciones y **Mnemónicos** para este caso son:

IN r, (C) OUT (C), r

La instrucción **IN r,(C)** carga el registro "r" con el contenido del **Puerto de Entrada/Salida** indicado en el registro C y la instrucción **OUT (C),r** carga el contenido del registro "r" en el **Puerto de Entrada/Salida** especificado en el registro C; el registro "r" puede ser cualquier de los **Registros de Propósito General** A, B, C, D, E, H o L. La ejecución de la instrucción **IN r,(C)** altera el estado de las banderas **S, Z** y **P/V**.

Durante la ejecución de estas instrucciones, el contenido del registro C se envía por las líneas A_0-A_7 del **Bus de Direcciones** y el contenido del registro B se envía por las líneas A_8-A_{15} del mismo **Bus de Direcciones**. El contenido del registro B se puede utilizar para transmitir información adicional o simplemente se puede ignorar.

En el siguiente ejemplo se lee el dato del **Puerto de Entrada 70H** y se envía al **Puerto de Salida 71H**.

```
LD C,70H        ; Cargar apuntador con numero del puerto de entrada
IN D,(C)        ; Leer en D el dato del puerto 70H
INC C           ; Apuntar ahora al puerto de salida
OUT (C),D       ; Enviar el dato al puerto 71H
```

3) **Instrucciones de E/S de Bloques**: Además de lo anteriormente explicado, el microprocesador Z80 cuenta con ocho instrucciones que permiten la transferencia de bloques de 1 a 256 bytes de datos entre dispositivos periféricos de entrada/salida y localidades de memoria apuntadas por el registro **HL**. Igual que en el caso de las instrucciones de búsqueda de datos y transferencia de bloques de datos, existen instrucciones para E/S de bloques de datos en forma automática y semiautomática. Sus **Mnemónicos** son:

INI	**INIR**	**IND**	**INDR**
OUTI	**OTIR**	**OUTD**	**OTDR**

En estas instrucciones el registro **HL** señala la dirección de la localidad de memoria que recibe o envía el dato, el registro **C** especifica el número del **Puerto de Entrada/Salida** deseada y el registro **B** realiza la función de contador para indicar cuantos bytes de datos se van a transferir. Durante la ejecución de las instrucciones, el contenido del registro C se envía por las líneas A_0-A_7 del **Bus de Direcciones** y el contenido del registro B se envía por las líneas A_8-A_{15} del **Bus de Direcciones**.

La instrucción **INI** ordena cargar el contenido del **Puerto de Entrada/Salida** especificado en el registro C hacia la localidad de memoria direccionada por el registro **HL**. Después de la

transferencia de un byte, el contenido del registro HL se incrementa en una unidad y el contenido del registro B se decrementa en una unidad. La bandera Z del registro F se pone en un nivel 1 lógico cuando el registro B llega a cero.

La instrucción INI es muy conveniente en las transferencias de varios datos, ya que lleva el control del controlador y del apuntador de memoria. Por ejemplo, la siguiente subrutina lee 100 datos (64H) del Puerto de Entrada 51H y los almacena en las localidades de memoria a partir de la dirección DATOS:

```
ENTRADA:  LDHL, DATOS    ; Dirección inicial de memoria
          LDBC, 6451H    ; Inicializar contador B=100 datos y apuntador del puerto
                         ; C=51H
LAZO:     INI            ; Almacenar en memoria el dato del puerto
          JP NZ, LAZO    ; Saltar si faltan de transferir datos
          RET            ; Regresa
```

Las instrucciones INI y JP NZ,LAZO reemplazan a las siguientes instrucciones en un programa utilizando instrucciones comunes:

```
LAZO:  IN A, (C)
       LD(HL), C
       INC HL
       DJNZ LAZO
```

La instrucción INIR es semejante a la instrucción INI, excepto que la ejecución termina hasta que se ha transferido el número de bytes especificado en el registro B, es decir, hasta que B sea cero. Cada iteración (transferencia de un byte) de la instrucción INIR toma 8.4 microsegundos para un reloj de 2.5 MHz. Se puede especificar un bloque de datos de 1 a 256 bytes.

La siguiente subrutina lee 90H datos del Puerto de Entrada 80H y los almacena en las localidades de memoria a partir de la dirección DATOS.

```
ENTRADA:  LDHL, DATOS    ; (HL) = dirección inicial de memoria
          LDBC, 9080H    ; (B) = 90H y (C) = 80H
          INIR           ; Leer 90H datos
          RET            ; Regresar al terminar la lectura
```

Las instrucciones IND e INDR operan en la misma forma que las instrucciones INI e INIR, con la excepción de que el registro HL se decrementa en una unidad, con lo cual la transferencia se realiza a localidades de memoria comenzando con la dirección más alta del bloque y terminando con la dirección más baja del mismo.

La instrucción OUTI transfiere el contenido de la localidad de memoria direccionada por el registro HL al Puerto de Entrada/Salida cuya dirección se especifica en el registro C. Después de la transferencia, el contenido del registro HL se incrementa en una unidad y el contenido del registro B se decrementa en uno. La bandera Z del registro F se pone en un nivel 1 lógico si el

registro B llega a cero. Las instrucciones **OUTI** e **INI** son semejantes en funcionamiento, excepto por el sentido de la transferencia de los datos.

En la siguiente subrutina se envía un bloque de 50 datos (32H), que se encuentran a partir de la localidad de memoria con dirección **BLOQUE**, al dispositivo periférico conectado al **Puerto de Salida 0DH**:

```
SALIDA:   LDHL, BLOQUE   ; Dirección del bloque de datos
          LD BC, 320DH   ; Inicializar contador B=50 datos y apuntador del puerto
                         ; C=0DH
LAZO:     OUTI           ; Enviar un dato
          JP NZ, LAZO    ; Saltar si faltan datos
          RET            ; Regresa
```

La instrucción **OTIR** es similar a la instrucción **OUTI**, con la diferencia de que en la primera la ejecución termina hasta después de transferir el número de bytes especificado en el registro **B**, es decir, hasta que el registro **B** sea cero.

El ejemplo que se presenta a continuación transfiere el bloque de **90H** datos, que se encuentran a partir de la localidad de memoria con dirección **TABLA**, al **Puerto de Salida 35H**:

```
SALIDA:   LDHL, TABLA    ; Dirección del bloque de datos
          LD BC, 9035H   ; Contador (B)=90H, Apuntador (C)=35H
          OTIR           ; Transferencia del bloque
          RET            ; Regresa
```

Las instrucciones **OUTD** y **OTDR** son semejantes a las instrucciones **OUTI** y **OTIR**, excepto que el apuntador **HL** se carga inicialmente con la dirección final del bloque de datos a transferir, decrementando el registro par **HL** en cada transferencia.

Durante la ejecución de cualquiera de las instrucciones de **E/S** explicadas en las secciones anterior, el microprocesador realiza un ciclo de máquina en el cual se lleva a cabo la transferencia de información entre el **Puerto de Entrada/Salida** y el microprocesador.

Si se trata de una instrucción de entrada (**IN**), a este ciclo de máquina se le denomina **Ciclo de Entrada (Input Cycle)**. En el transcurso de la ejecución de estas instrucciones, el microprocesador lee el dato proveniente del **Puerto de Entrada**. En el caso contrario, es decir, cuando se trata de una instrucción de salida(**OUT**), al ciclo de máquina en que el microprocesador envía el dato al **Puerto de Salida** se le llama **Ciclo de Salida (Output Cycle)**

Los ciclos de máquina de entrada o salida son análogos a los ciclos de lectura o escritura en memoria respectivamente. La principal diferencia radica en las señales de control que genera el microprocesador para indicar una operación con memoria, las cuales son distintas a las generadas cuando se realiza una operación de hacia un **Puerto de Entrada/Salida**.

Así como el microprocesador Z80 produce la señal $\overline{\text{MREQ}}$ durante los ciclos de máquina hacen referencia a la memoria, de manera similar, al ejecutar una operación con un **Puerto de Entrada/Salida**, el microprocesador genera una señal llamada $\overline{\text{IORQ}}$ (Input/Output Request).

La señal $\overline{IORQ}$ se activa en 0 lógico y proviene de una terminal de salida triestado del microprocesador. Cuando ésta se activa, indica que la parte baja del **Bus de Direcciones** (A_0-A_7) contiene un número de **Puerto de Entrada/Salida** válido para una operación de entrada o salida

La señal $\overline{IORQ}$ se activa al principio de T_2, permaneciendo en un nivel 0 lógico hasta la mitad de T_3. Además, en un ciclo de entrada se activa la señal $\overline{RD}$ desde el inicio de T_2 hasta la mitad de T_3, siendo usada para que el **Puerto de Entrada/Salida** direccionado ponga el dato en el **Bus de Datos**, de donde es leído por el microprocesador al bajar el reloj en T_3. Por otro lado; en un ciclo de salida la señal que se activa es $\overline{WR}$, desde el comienzo de T_2 hasta la mitad de T_3, siendo utilizada como señal de reloj para grabar el dato en el **Puerto de Entrada/Salida** al volver a un nivel 1 lógico. En este caso, el microprocesador pone el dato en el **Bus de Datos** a partir de la mitad de T_1 y lo mantiene estable hasta finalizar T_3.

Durante las operaciones de **E/S**, el microprocesador inserta automáticamente un estado de espera (**Tw***) después de T2. Este estado adicional da un margen más amplio para que un **Puerto de Entrada/Salida** decodifique las **Líneas de Dirección**. La razón para esto es que durante las operaciones de E/S, el tiempo desde que la señal $\overline{IORQ}$ baja a un nivel 0 lógico hasta que el microprocesador examina la línea $\overline{WAIT}$ es muy corto. Sin este estado de retardo extra no hay tiempo suficiente para que un **Puerto de Entrada/Salida** decodifique su dirección y active la línea $\overline{WAIT}$, si ésta tiene un nivel 0 lógico, se inserta un nuevo estado de espera **Tw**.

En el microprocesador Z80, las señales que determinan los espacios de direccionamiento de **Puertos de Entrada/Salida Aislada** son las **Líneas de Dirección** A_0-A_7 y las **Señales de Control** $\overline{IORQ}$, $\overline{RD}$ y $\overline{WR}$.

Para controlar **Puertos de Entrada**, los circuitos externos combinan las señales $\overline{IORQ}$, $\overline{RD}$ y la dirección del **Puerto de Entrada/Salida**, generando un pulso de selección exclusivo para cada **Puerto de Entrada/Salida**. Este pulso ocurre en el momento en que el microprocesador está listo para leer el dato del **Puerto de Entrada/Salida** durante el ciclo de entrada de una instrucción **IN** que especifique ese **Puerto de Entrada/Salida**.

Para **Puertos de Salida**, los circuitos externos combinan las señales $\overline{IORQ}$, $\overline{WR}$ y la dirección del **Puerto de Entrada/Salida** generan un pulso de selección único para cada **Puerto de Entrada/Salida**. Este pulso actúa a la manera de un pulso de escritura para grabar el dato en el **Puerto de Entrada/Salida** durante el ciclo de salida de una instrucción **OUT** que especifique ese **Puerto de Entrada/Salida**.

Hay que insistir en la importancia de que a cada **Puerto de Entrada/Salida** le corresponda una dirección distinta. Aún en situaciones en las que un solo dispositivo periférico de entrada tiene asociados más de un **Puerto de Entrada**, cada uno de ellos debe tener una dirección diferente; el motivo sigue siendo el mismo; la prevención de conflictos en el **Bus de Datos** originados por la activación simultanea de varios buffers de tres estados con sus salidas conectadas a un punto común.

Cuando se trata de **Puertos de Salida**, normalmente cada uno se identifica con un número diferente; sin embargo, es posible y frecuentemente útil que varios de ellos sean seleccionados al mismo tiempo. Como en este caso es el microprocesador el que controla la transferencia y el que envía el dato, no hay posibilidad de conflicto. La única limitación en este tipo de conexión es que a la hora de efectuar una transferencia a ese número de **Puerto de Entrada/Salida**, todos recibirán idéntica información.

Nótese que puede haber un **Puerto de Entrada** con el mismo número que un **Puerto de Salida** sin que ocurra ningún problema, ya que nunca estarán activos simultáneamente. Esto se debe a que para generar el pulso de selección del **Puerto de Entrada** es necesaria la señal $\overline{RD}$ y para que se genere el pulso de selección del **Puerto de Salida** se requiere la señal $\overline{WR}$.

El diseño de la circuitería de selección varía, dependiendo del numero de dispositivos periféricos de E/S conectadas al sistema y de las características de los circuitos que componen los **Puertos de Entrada/Salida.** Si únicamente se requiere un **Puerto de Entrada** y un **Puerto de Salida**, no es necesario decodificar las **Líneas de Dirección**. Para generar los pulsos de selección de estos **Puertos de Entrada/Salida**, bastan las señales $\overline{I/OR}$[1] e $\overline{I/OW}$[2]. En este caso el número de **Puerto de Entrada/Salida** no importa, pero de cualquier manera no puede omitirse en la instrucción.

Cuando el sistema requiere más de un **Puerto de Entrada** o **de Salida**, entonces es necesario decodificar las **Líneas de Dirección** con el fin de generar los pulsos de selección para cada **Puerto de Entrada/Salida** en particular.

La forma más simple de decodificación de las **Líneas de Dirección** es la **Selección Lineal**, ya que utiliza el menor número de compuertas. En este método una sola **Línea de Dirección** se asocia exclusivamente con cada **Puerto de Entrada/Salida**, combinándose con las señales $\overline{IORQ}$ y $\overline{RD}$ o $\overline{WR}$ para generar el pulso de selección correspondiente. Como únicamente son ocho las **Líneas de Dirección** para los **Puertos de Entrada/Salida**, sólo pueden seleccionarse hasta ocho **Puertos de Entrada** y ocho **Puertos de Salida** con este método. Sin embargo, la eliminación de los circuitos que se requieren para decodificar las **Líneas de Dirección** se convierten en un ahorro importante en sistemas pequeños.

Una desventaja de la **Selección Lineal** es la posibilidad de que un error de programación dañe los circuitos si accidentalmente dos **Puertos de Entrada** utilizan el **Bus de Datos** simultáneamente. Suponga, por ejemplo, que un sistema particular que usa **Selección Lineal** contiene un **Puerto de Entrada** en la dirección **04H** y otro **Puerto de Entrada** en la dirección **08H**, seleccionados por las **Líneas de Dirección** A_2 y A_3, respectivamente. Si se programa una

[1] Por razones de simplicidad, se utiliza esta nomenclatura en lugar de indicar las señales que se activan (IOREQ, RD así como la dirección del **Puerto de Entrada/Salida** si fuera necesario) cuando ocurre una transferencia de datos entre los dispositivos periféricos y el microprocesador (lectura de datos).

[2] Por las mismas razones, se utiliza esta nomenclatura en lugar de indicar las señales que se activan (IOREQ, WR así como la dirección del **Puerto de Entrada/Salida** si fuera necesario) cuando ocurre una transferencia de datos entre el microprocesador y los dispositivos periféricos (escritura de datos).

instrucción IN A,(0CH), en lugar de una instrucción **IN A,(08H)**, su ejecución causará que los **Puertos de Entrada 04H y 08H** sean seleccionados al mismo tiempo, lo cual muy probablemente dañará los buffers tnestado.

Para seleccionar un numero mayor de dispositivos periféricos de **Entrada/Salida**, es necesario decodificar las ocho **Lineas de Dirección** que correspondan a los **Puertos de Entrada/Salida**. Con **Decodificación Completa** o **Exhaustiva**, el máximo número de pulsos de selección que pueden ser generados son 512:256 asociados con instrucciones **IN** y 256 asociados con instrucciones **OUT**. Para efectuar la decodificación, los circuitos decodificadores con entradas de habilitación (**Chip Enable**) son los más convenientes porque reducen el número de compuertas.

Si una aplicación requiere cuatro o menos pulsos de selección para entrada y cuatro o menos pulsos de selección para salida, se puede utilizar un circuito integrado 74LS139 que es un circuito con dos decodificadores de 2 a 4 lineas. Sus salidas permanecen en un nivel 1 lógico mientras no sean seleccionadas; cuando una de ellas es seleccionada, ésta pasa a un nivel 0 lógico Sus entradas de habilitación son activas en un nivel 0 lógico.

La entrada de habilitación de uno de los decodificadores se conecta a la señal $\overline{I/OR}$ y las entradas de selección a las **Líneas de Dirección** A_0 y A_1, produciendo cuatro pulsos de selección para el mismo número de **Puertos de Entrada** en las salidas del decodificador. Las cuatro salidas están en un nivel 1 lógico hasta que aparece la señal $\overline{I/OR}$, entonces en la salida seleccionada por A_0 y A_1 se genera un pulso con un nivel 0 lógico. Las otras salidas permanecen en un nivel 1 lógico. La entrada de habilitación del otro decodificador se conecta a la señal $\overline{I/OW}$ y las entradas de selección de A_0 y A_1; con esto se obtienen cuatro pulsos de selección para el mismo número de **Puertos de Salida**.

Como únicamente las dos líneas menos significativas del **Bus de Direcciones** (A_0 y A_1) están conectadas al decodificador, el estado lógico de las seis líneas más significativas del primer octeto del **Bus de Direcciones** (A_2 a A_7) es una condición "no importa". El efecto de esto es la generación de los mismos pulsos de selección para instrucciones **IN** u **OUT** con diferentes direcciones de **Puertos de Entrada/Salida**. Por ejemplo, el pulso de selección para el **Puerto de Entrada 00H (PE00H)** puede generarse ejecutando la instrucción **IN A,(00H)** o cualquiera de las siguientes: **IN A,(04H)**; **IN A,(08H)**; **IN A,(0CH)**; **IN A,(10H)**; **IN A,(14H)**; etc.

Los circuitos decodificadores que tienen varias entradas de habilitación, algunas activas en un nivel 0 lógico y otras activas en un nivel 1 lógico, proporcionan aún más flexibilidad. Los circuitos integrados 8205 y 74LS138 son decodificadores de 3 a 8 lineas que poseen 3 entradas de habilitación (chip enable), dos de ellas activas en un nivel 0 lógico y una de ellas activa en un nivel 1 lógico. Con uno de estos circuitos se pueden generar hasta ocho pulsos de selección activos en un nivel 0 lógico y no se requieren compuertas adicionales para implantar las señales $\overline{I/OR}$ e $\overline{I/OW}$ porque éstas se obtiene con lógica interna del circuito.

Los circuitos decodificadores con entradas de habilitación se pueden conectar en cascada para generar un mayor número de pulsos de selección. Una cantidad de 17 circuitos integrados 74LS154 que son decodificadores de 4 a 16 líneas proporcionan 256 pulsos de selección.

Otra forma de generar pulsos de selección en sistemas con un número reducido de **Puertos de Entrada/Salida** es usando circuitos comparadores de magnitud. Con este método se tiene la ventaja de poder ubicar a cada **Puerto de Entrada/Salida** en un lugar específico del espacio de direccionamiento, sin necesidad de decodificar exhaustivamente las ocho **Líneas de Dirección** para **Puertos de Entrada/Salida**. Sin embargo, tiene la desventaja de que se requiere cuando menos un comparador para cada **Puerto de Entrada/Salida**, lo cual, al crecer el número de **Puertos de Entrada/Salida** implica un aumento considerable de la circuitería de selección.

Por ejemplo, el circuito integrado 74LS648 es un comparador binario de dos números de 8 bits, P y Q, que cuenta con dos salidas activas en 0 lógico: P=Q y P>Q. El circuito activa la salida apropiada, de acuerdo a la relación obtenida de la comparación de los dos números.

Suponiendo que el sistema a diseñar requiriera de un **Puerto de Entrada** cuya dirección sea **57H** y un **Puerto de Salida** con dirección **E1H**, se pueden utilizar dos comparadores 74LS684 para generar los dos pulsos de selección.

3.3.2 Puertos de Entrada/Salida Mapeados

El modo de **Entrada/Salida Mapeada a Memoria (Memory Mapped I/O)** se basa en que tanto las localidades de memoria como los **Puertos de Entrada/Salida** se consideran como registros externos desde el punto de vista del microprocesador. Entonces, las instrucciones que hacen referencia a la memoria también pueden transferir datos entre un dispositivo periférico y el microprocesador, siempre y cuando el **Puerto de Entrada/Salida** que los interconecta se encuentre dentro del espacio de direccionamiento de memoria, es decir, controlado por las **Señales de Control** para memoria, que en el caso del microprocesador Z80 son $\overline{MREQ}$, $\overline{RD}$ y $\overline{WR}$. De esta forma, el registro asociado con el **Puerto de Entrada/Salida** es tratado simplemente como una localidad de memoria más.

Una característica importante es que las operaciones de entrada y salida usando **Puertos de Entrada/Salida Mapeada a Memoria** no están limitadas a los registros internos. Por ejemplo, algunas de las instrucciones del microprocesador Z80 que pueden utilizarse para leer de **Puertos de Entrada Mapeados a Memoria** son:

Instrucción	Interpretación para E/S mapeada a memoria
LD r, (HL)	Entrada de un puerto a cualquier registro
LD A, (pq)	Entrada de un puerto a un acumulador
LD HL, (pq)	Entrada de dos puertos a los registros H y L
ADD A, (HL)	Entrada de un puerto al acumulador con operación aritmética
AND (HL)	Entrada de un puerto al acumulador con operación lógica
BIT b, (HL)	Examinar estado lógico del bit b de un puerto de entrada

Algunas instrucciones que pueden enviar información a **Puertos de Salida Mapeados a Memoria** son:

Instrucción	Interpretación para E/S mapeada a memoria
LD (HL), r	**Salida de cualquier registro a un puerto**
LD (pq), A	**Salida del acumulador a un puerto**
LD (pq), HL	**Salida de los registros H y L a dos puertos**
LD (HL), n	**Salida del valor inmediato n a un puerto**
SET b, (HL)	**Poner en 1 lógico el bit b de un puerto de salida**

Para generar los pulsos de selección de **Puertos de Entrada/Salida Mapeados a Memoria** es necesario decodificar las 16 líneas del **Bus de Direcciones** (A_0-A_{15}) junto con las señales de control $\overline{MREQ}$ y $\overline{RD}$ o $\overline{WR}$, dependiendo si es un **Puerto de Entrada** o un **Puerto de Salida**.

Una manera sencilla de implantar una estructura de **Puerto de Entrada/Salida Mapeada a Memoria** es utilizando la línea más significativa del **Bus de Direcciones** (A_{15}), para indicar si la dirección especificada en una instrucción se refiere a una localidad de memoria o a un **Puerto de Entrada/Salida**. Cuando la **Línea de Dirección** A_{15} está en un nivel 0 lógico, entonces las líneas A_0-A_{14} direccionan una localidad de memoria. Si la **Línea de Dirección** A_{15} está en un nivel 1 lógico, esto significa que la dirección mostrada por las **Líneas de Dirección** A_0-A_{14} corresponden a un **Puerto de Entrada/Salida**. De esta forma, se dedican los primeros 32K a memoria y quedan disponibles los restantes 32K para direccionar **Puertos de Entrada/Salida**.

3.3.3 Comparación entre Puerto de E/S Aislado y Puerto de E/S Mapeado

El modo de **Puerto de Entrada/Salida Aislada** presenta las siguientes ventajas:

1) Como se usan instrucciones especiales para realizar procesos de **Entrada/Salida** en un programa, éstas pueden distinguirse fácilmente de las instrucciones que hagan referencia a memoria.
2) Como solo se utilizan ocho líneas en el direccionamiento de un **Puerto de Entrada/Salida**, se necesitan menos circuitos para su decodificación.
3) Como el número de **Puerto de Entrada/Salida** se puede representar en un byte, las instrucciones son mas cortas.
4) Como los **Puertos de Entrada/Salida** están asignados a un espacio separado de la memoria, se tiene disponible la capacidad total de direccionamiento del microprocesador para circuitos de memoria.

Las desventajas de este método de conexión de **Puertos de Entrada/Salida** son:

1) La capacidad de procesamiento y flexibilidad de las instrucciones de **Entrada/Salida** es en general muy restringida.
2) Se debe dedicar al menos una terminal del circuito integrado del microprocesador para la señal de control que distingue las operaciones con **Puertos de Entrada/Salida** de las operaciones que se realizan con la memoria.

Por otra parte, el modo de **Puertos de Entrada/Salida Mapeados a Memoria** tiene como ventajas las siguientes:

1) Permite la utilización de la gran variedad de instrucciones que hacen referencia a la memoria para la transferencia de información y la ejecución de operaciones aritméticas o lógicas directamente en los **Puertos de Entrada/Salida**, sin necesidad de transferir los datos a los registros internos del microprocesador.
2) Reduce el número de **Líneas de Control** que debe tener el microprocesador.

Las desventajas para los **Puertos de Entrada/Salida Mapeados a Memoria** son las siguientes:

1) Cada **Puerto de Entrada/Salida** implantado de este modo disminuye en uno las direcciones disponibles para la memoria.
2) Es necesario decodificar las 16 Líneas del **Bus de Direcciones** para seleccionar el **Puerto de Entrada/Salida**.
3) Las instrucciones que hacen referencia a la memoria requieren dos bytes para representar la dirección, por lo tanto son mas largas y también pueden ser más lentas.

De lo anterior se concluye que ninguno de los dos modos de implantar los **Puertos de Entrada/Salida** es claramente mejor que el otro. Por lo tanto, la decisión de cual utilizar depende de las características particulares que tenga el sistema en cuestión.

Si el sistema exige que el espacio total destinado al almacenamiento esté ocupado por circuitos de memoria, entonces la única alternativa para la implantación de **Puertos de Entrada/Salida** es la **Aislada**. Por otro lado, si los requerimientos de memoria son reducidos, el empleo de la técnica de **Puertos de Entrada/Salida Mapeados a Memoria** puede resultar adecuado.

Los pulsos de selección de **Puertos de Entrada/Salida** también son útiles como pulsos de control en aplicaciones donde se controlan dispositivos externos, con los que no se pretende efectuar ninguna transferencia de datos. Es preferible la instrucción **OUT** para generar los pulsos de control, ya que la instrucción **IN** provoca que el dato contenido en el **Bus de Datos** pase al **Acumulador** (o alguno de los registros internos).

Un ejemplo de esto puede ser el control del encendido y apagado de un motor, a través de pulsos de selección generados por programa, que ponen en un nivel 0 o 1 lógico un flip-flop.

Inicialmente se supone que la salida Q del flip-flop está desactivada, por lo tanto, el nivel lógico del buffer está en 0 lógico. Debido a lo anterior la bobina del relevador no esta energizada y como el relevador es del tipo "normalmente abierto", entonces no existe flujo de corriente en el circuito **CA**, lo que ocasiona que el motor se encuentre desconectado.

La ejecución de una instrucción **OUT (00H), A** pone el flip-flop en un nivel 1 lógico y hace que la salida del buffer sea de 5V, activando la bobina del relevador, la que a su vez cierra los contactos. De esta forma se establece el flujo de corriente en el circuito **CA**, poniendo al motor en funcionamiento. La ejecución de una instrucción **OUT(01H),A** regresa la salida Q del flip-flop a un nivel 0 lógico, apagando el motor. Nótese que la información que sale al **Bus de Datos** durante la

ejecución de las instrucciones OUT, es decir, el contenido del **Acumulador**, no es enviada a ningún lado ni afecta al sistema.

Si se quiere mantener encendido el motor por un tiempo determinado, se puede hacer uso de las técnicas para la generación de tiempos de espera por programa.

La siguiente rutina puede servir para este propósito:

```
MOTOR       (10)    LD BC, NN       ; Cargar contador con valor inicial
ENCENDER    (11)    OUT (00H), A    ; Prender el motor
LAZO        (6)     DEC BC          ; Decrementar contador
            (4)     LD A, C         ; Revisar si el contador es cero
            (4)     OR B
            (10)    JP NZ, LAZO     ; Si no es cero brincar al lazo
APAGAR      (11)    OUT (01H), A    ; Si es cero apagar motor
```

El tiempo en que el motor permanece encendido se calcula a partir del numero **NN** con que se carga inicialmente el contador y de la duración de cada una de las instrucciones dentro del ciclo. Los números encerrados entre paréntesis en la rutina anterior representan los estados que dura cada instrucción. Un estado (t_s) equivale a 400 ns para un cristal de 2.5 MHz.

Así pues, el tiempo de funcionamiento esta dado por:

$T=[NN*(6+4+4+10)+11]*t_s$

$T=[24*NN+11]*400$ ns para $0<NN<65,535$

si NN=1 se obtiene T_{min} = 14 microsegundos

si NN=0 se obtiene T_{max} = 0.629 segundos

3.3.4 Circuitos Para la Implantación de Puertos de Entrada/Salida

Los **Puertos de Entrada/Salida** pueden implantarse con circuitos **SSI**, **MSI** o **LSI**. Sin embargo, para minimizar el número de componentes en general se usan circuitos **MSI** o **LSI**.

Un **Puerto de Salida** de 8 bits se puede implantar con un circuito de tipo cerrojo (latch) de 8 bits como el circuito integrado 74LS373 o el 74LS374. El circuito integrado 74LS373 contiene 8 cerrojos disparados en el nivel positivo con una entrada común de reloj activa en 1 lógico. El circuito integrado 74LS374 contiene 8 flip-flops tipo D disparados en la transición positiva (0 a 1) con una entrada común de reloj.

Ambos circuitos poseen salidas con buffers de tres estados. Los buffers tienen una entrada de habilitación activas en un nivel 0 lógico. Cuando alguno de estos circuitos se usa para implantar un **Puerto de Salida**, los buffers se mantienen activos todo el tiempo o son controlados por el dispositivo periférico de salida.

El circuito integrado 9334 de National Semiconductors proporciona 8 cerrojos direccionables individualmente, los cuales pueden utilizarse como 8 **Puertos de Salida** de un bit para propósitos de control. La entrada de datos del circuito integrado 9334 se conecta a una de las líneas del **Bus de Datos**. Cada uno de los cerrojos es seleccionado como un **Puerto de Salida** independiente por medio de tres **Líneas de Dirección**, con lo cual se puede cambiar el estado de un bit sin afectar el

estado de los otros. Con este circuito solamente se requieren dos instrucciones para poner un bit en un nivel 0 o en 1 lógico: una de ellas carga el **Acumulador** con **00H** o **01H**; la otra es una instrucción de salida para el **Puerto de Entrada/Salida** (bit) seleccionado, suponiendo que se encuentra conectado un circuito integrado 9334 al microprocesador Z80.

En contraste, para cambiar un solo bit sin alterar ninguno de los otros bits de un **Puerto de Salida** en donde todos los bits son manejados simultáneamente, por ejemplo un circuito integrado 74LS373 o un 74LS374, se vuelve necesario mantener en memoria una copia de la ultima información enviada al **Puerto de Entrada/Salida**.

Los cerrojos y los buffers triestado del circuito integrado 74LS373 y del 74LS374 permiten que estos circuitos se puedan utilizar también como **Puertos de Entrada**. Cuando un dispositivo periférico de entrada tiene su propio cerrojo pero sin salidas de tercer estado se le añade un circuito buffer con salidas de tres estados. Por ejemplo, el circuito integrado 74LS240 y el 74LS244 son buffers octales con salidas de tres estados. Los buffers triestado en cada uno de ellos están divididos en dos grupos de cuatro, cada grupo con una entrada común de habilitación activa en un nivel 0 lógico. Además las entradas de datos son gatillos Schmitt para mejorar su inmunidad al ruido.

Un circuito integrado especialmente útil y flexible para implantar puertos de 8 bits, tanto de entrada como de salida es el 8212 de Intel. Este es un circuito integrado de 24 terminales fabricado con tecnología bipolar Schottky (figura número 3.25).

El circuito integrado 8212 contiene un registro de 8 bits que puede ser activado de diversas maneras. Los ocho cerrojos que componen el registro se disparan en el nivel positivo. La salida de cada cerrojo (latch) va conectada a un buffer de tres estados. Como puede observarse en el diagrama lógico de la figura número 3.25, la señal de reloj que controla los cerrojos así como la señal de habilitación de los buffers de salida, se derivan de cuatro entradas (MD, $\overline{DS1}$, DS2 y STB) a través de un circuito combinacional. Normalmente las líneas de entrada $\overline{DS1}$ y DS2 van asociadas a la selección del circuito (device select); MD (mode) define su modo de operación, y STB (strobe) se utiliza como entrada de reloj.

Cuando un circuito integrado 8212 se usa como **Puerto de Salida**, el modo de operación se indica con MD=1. La señal de reloj C se obtiene de las entradas de selección ($\overline{DS1}$ y DS2). Cuando $\overline{DS1}$ está a un nivel 0 lógico y DS2 se encuentra en un nivel 1 lógico, se selecciona el circuito y el reloj está en un nivel 1 lógico. Mientras C=1 las salidas de datos siguen a las entradas de datos. La información de las entradas se graba en la transición del nivel 1 al nivel 0 de la señal de reloj. Dependiendo de que se use $\overline{DS1}$ o DS2, el **Puerto de Entrada/Salida** se puede activar con un pulso de selección activo en un nivel 0 lógico o activo en un nivel 1 lógico, respectivamente. Como se trata de un **Puerto de Salida**, los buffers triestado se mantienen activos todo el tiempo. Cuando se usa como **Puerto de Entrada**, el circuito integrado 8212 proporciona un buffer de tercer estado controlado por el microprocesador y un registro controlado por el dispositivo periférico de entrada.

El modo de operación se indica con MD=0: el estado del buffer lo determinan $\overline{DS1}$ y DS2. La señal de reloj de los cerrojos proviene de la entrada STB.

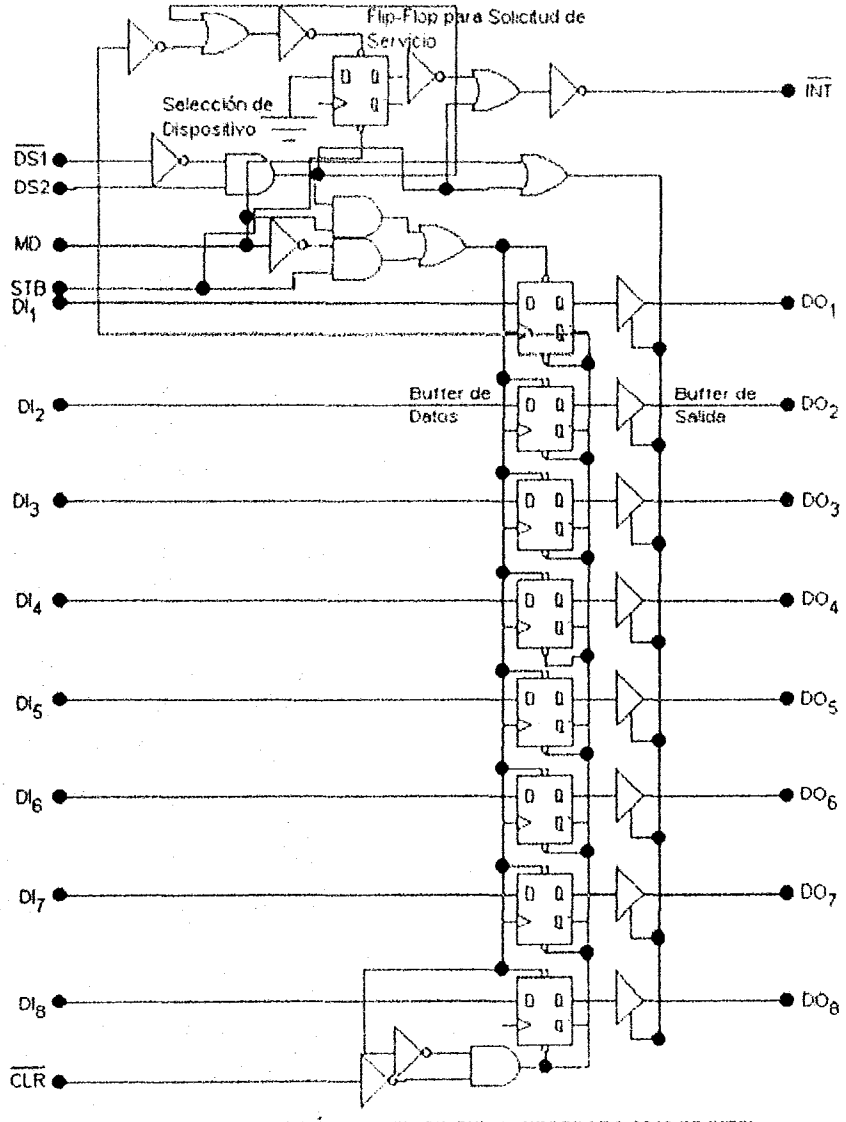


Fig. 3.25 DIAGRAMA LÓGICO DEL CIRCUITO INTEGRADO 8212 DE INTEL

3.4 Resumen

Antes de proceder a realizar las conexiones de los diversos elementos que van a integrar el Sistema Mínimo, es esencial tener una noción de los diversos tipos de conexión de los dispositivos periféricos a los microprocesadores.

Las formas de conexión de los dispositivos periféncos pueden ser mediante líneas comunes o independientes, estas formas de conexión se eligen de acuerdo al número de pines existentes en el microprocesador o la distancia existente entre el microprocesador y los dispositivos periféricos (serial si existe una distancia grande entre ambos dispositivos o paralela si la distancia entre ambos es corta y no existe la necesidad de serializar los datos).

De acuerdo a los dispositivos periféricos conectados al sistema, se puede elegir el intercambio de información iniciada por el microprocesador o por el dispositivo periférico (mediante el uso de líneas de interrupción).

Finalmente, es ineludible el conocimiento de las formas de mapear la memoria del microprocesador Z80, con el objeto de elegir la mejor forma de conectar los dispositivos periféricos al Sistema Mínimo, con el fin de realizar una mejor elección en cuanto a la forma de direccionamiento de los **Puertos de Entrada/Salida**, con la consecuente elección de los circuitos integrados que permitan realizar una buena comunicación entre los dispositivos periféricos y el microprocesador al menor costo posible.

CAPITULO IV:
DISPOSITIVOS PERIFÉRICOS

"Una respuesta de un programador a una persona que no conoce de computadoras todo se puede, que no podamos o que no sepamos es diferente"

Ivan Edo Rodriguez

4.1 Interfaz serie

Ha pensado alguna vez, cómo es posible que se pueda efectuar una transmisión de datos desde una distancia tan lejana como Europa, Asía o el continente africano. Aún más, si el **Bus de Datos** de un sistema computarizado utiliza un tamaño mínimo de 8 bits (puesto que los modernos equipos Pentium y similares utilizan un **Bus de Datos** de 64 bits), en el remoto caso de que se utilizará un canal de transmisión análogo al bus de los sistemas PCs, ello implicaría tener un tendido de cables en paralelo de 8 líneas, adicionalmente deben de tener unas líneas tales como la fuente de potencia, tierra y señales que permitan efectuar una sincronización de los datos (es decir, señales que indiquen al transmisor si el receptor está encendido, si se encuentra ocupado, si su buffer se encuentra lleno, si ha recibido bien los datos, etc.), además de una señal **Strobe** que indique al receptor que los datos están disponibles y son válidos. Por si esto fuera poco, los cables que se utilizan como medios de conducción de las señales actúan como resistores que paulatinamente van transformando esta señal en calor y la van disipando, ello implica que a ciertos intervalos de distancia se debe de poner unos retransmisores que permitan regenerar las señales y volverlas a emitir.

Como puede verse de lo anteriormente expuesto, tener un canal de comunicación de tal índole repercutiría en forma grave en el costo de implementar tal canal, puesto que éste mismo se incrementaría a proporciones inaceptables para las empresas, entidades gubernamentales instituciones educativas, etc. Lo cual repercutiría en el usuario final, puesto que sería inaccesible para la mayoría de las personas.

Para disminuir en la mayor medida de la posible los gastos que acarrearía tal proyecto, la primer opción viable que se eligió fue utilizar los servicios de los medios de comunicación existentes en el mundo: el teléfono. Sin embargo, las líneas telefónicas no cuentan con un cableado de tantas líneas.

Para evitar tal contratiempo, y disminuir el costo de implementar un canal de comunicación entre equipos PCs ubicados a distancias lejanas, se decidió utilizar una transmisión de datos en serie (bit a bit) para lo cual, los datos en paralelo son serializados. Después de tener los datos en serie, estos son transmitidos bit por bit en un solo hilo de par trenzado. Para una correcta transmisión de los datos, se utilizan señales de **Inicio** y de **Parada** (**Start** y **Stop**), los cuales tienen la misión de indicar al receptor que los datos que se están emitiendo son válidos. Gracias a

ésta técnica, surgen los conceptos de **Half-Duplex** o **Semi-Duplex** y **Full-Duplex** o **Duplex Completo**

De igual forma, lo anterior se aplica para las comunicaciones del microprocesador con sus dispositivos periféricos tales como las impresoras, teclados, pantallas de video, módems, entre otros.

Para poder efectuar una comunicación paralelo-serie y serie-paralelo, se puede utilizar dispositivos tan comunes como los registros de desplazamiento, tales como los circuitos integrados 74LS194, 74LS322, entre otras. Sin embargo, al utilizar estos dispositivos se debe de tener en mente las caracteristicas eléctricas de los periféricos emisores, de tal suerte que se consiga adaptar los registros de corrimiento a los emisores y perder algunos datos tales como los bits de **Start, Stop** y de **Paridad**.

Otra posibilidad es utilizar circuitos integrados especializados que permitan una transformación paralelo-serie y serie-paralelo, que permitan además captar todos los bits utilizados en la transmisión de los datos. Tales dispositivos se denominan **UART (Universal Asincronic Receptor/Transmisor** o **Receptor/Transmisor Asíncrono Universal)**, ejemplos de tales dispositivos son los típicos **UART SMC COM2017, AY-5-1013A, AY-1015** o **TR1602** (estos dos últimos funcionan con una sola fuente de alimentación de +5V, en tanto que los demás requieren una alimentación extra de –12V).

Básicamente, están formados internamente por transmisores paralelo-serie y receptores serie-paralelo independientes, los cuales se encuentran unidos por terminales de programación comunes.

Normalmente, la transmisión de datos en serie sigue el formato asincrono ilustrado en la siguiente figura:

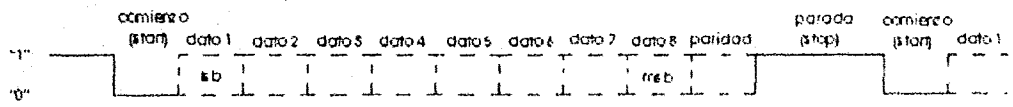


Fig. 4.1 FORMATO DE UN OCTETO DE DATOS EN UNA TRANSMISIÓN EN SERIE

Como se puede ver en la anterior figura cuando el emisor no está transmitiendo dato alguno, la línea de datos se mantiene en una señal de nivel alto ó 1 lógico, en espera de una señal **Start**. Esta última está representada por una transición de nivel alto a un nivel bajo, posterior a ésta, siguen ocho bits de datos. Al final de éstas, se emiten unas señales de **Paridad** y de **Parada (Stop)** del sistema, este mismo principio vuelve a repetirse para cada byte de datos transmitido.

En el extremo contrario (receptor), otra **UART** controla la línea de entrada en serie para el bit de comienzo. Al ocurrir este hecho, se almacenan los bits de datos en un registro y se verifica la paridad de los mismos. Al terminar de recibir los datos y verificar la paridad, la **UART** genera una señal de datos listos, la cual puede ser utilizada como señal de **Strobe** para el microprocesador

(esta señal puede ser aplicada a la linea $\overline{INT}$, la cual detiene el proceso actual del microprocesador, obligándolo a atender la petición de interrupción).

Sin embargo, para que el receptor pueda seguir recibiendo datos, es menester indicarle que el microprocesador ha leido los datos y **Resetear** la linea de datos disponibles (rehabilitando la función del **UART**).

Además de lo anterior, se necesita conocer la velocidad de transmisión de los datos, la cual puede ser expresa como bits por segundo. Algunas velocidades tipicas de transmisión de datos estándares en la industria son.

110 bps
150 bps
300 bps
600 bps
1200 bps
2400 bps
4800 bps
9600 bps

Tabla 4.1 VELOCIDADES ESTÁNDARES DE TRANSMISIÓN DE DATOS

Para poder generar estas velocidades de transmisión podemos utilizar circuitos integrados generadores de baudio o diseñar alguno que se adapte a nuestros requerimientos.

Sin embargo, existe una cuestión más: ¿Cuál es el nivel de voltaje que se debe utilizar para que la comunicación entre el microprocesador y los dispositivos periféricos se realice con éxito?, si se toma en cuenta que ambas utilizan distintos niveles de voltaje. Para resolver tal conflicto, se diseñó la norma **EIA RS-232C** (ampliamente utilizada en los equipos módems), gracias a esta norma conocemos los niveles que deben prevalecer para un nivel 0 lógico y un nivel 1 lógico, asi como sus tipos de conexiones, las asignaciones de los terminales, las impedancias de las fuentes y de carga así como otras funciones adicionales.

Los niveles de la norma **RS-232C** son bipolares, oscilan entre –3 y –15 volts para representar un nivel 1 lógico y una tensión entre +3 y +15 volts para representar un nivel 0 lógico. La región entre –3 y +3 volts permiten una mejor inmunidad al ruido

Para poder manejar estos niveles de tensión se debe de proporcionar al sistema un controlador **TTL a RS-232C** (tal como el circuito integrado **MC1488**) en la parte transmisora. Otro tanto ocurrirá en la parte receptora, pero en forma inversa; es decir, se le debe de proporcionar un controlador **RS-232C a TTL** (tal como el circuito integrado **MC1489**).

Como sustituto de los circuitos integrados anteriores, se puede utilizar un arreglo de transistores y resistencias para poder manejar tales niveles de voltaje, como ejemplos se pueden mencionar los conocidos transistores **2N2219** y **2N2222**.

[1] Bits por segundo

4.2 Teclados

Como se ha explicado anteriormente, los teclados son dispositivos periféricos de entrada de datos al microprocesador que en su parte más elemental constan de una matriz de teclas, decodificadas por un microcontrolador diseñado para tal fin (circuito integrado 8048, o afín). Este chip utiliza una técnica de exploración para monitorear la matriz del teclado. Cada una de las teclas está conectada a una de las intersecciones renglón-columna, las cuales emiten un nivel alto lógico si una tecla no está presionada. Al ser presionada una tecla, el procesador recibe una señal de nivel lógico bajo. La tecla presionada es decodificada y se le hace corresponder con un carácter en la **ROM de Caracteres**. El patrón de bits se envía a través del cable de datos a la tarjeta del sistema (**Interfaz de Teclado**).

4.2.1 Tipos de Teclado

Los tipos más comunes de teclados existentes en el mercado se denominan **XT** y **AT**. Los teclados **XT** son los primeros que aparecieron en las computadoras personales y se caracterizan sobre todo por que tienen una interfaz unidireccional (el teclado sólo emite datos a la interfaz).

Las características más sobresalientes de los teclados **XT** son:

a) Generan 2 bits de inicio, 8 bits de datos uno de los cuales es un bit de **Make/Break Code**[2], y un bit de **Parada (Stop)**.
b) Se utiliza un bit **Make/Break** para indicar el estado de las teclas (Un nivel lógico 0 indica que la tecla está presionada y un nivel lógico 1 indica que la tecla ha sido soltada).
c) Tiene un código de teclas distinto a los teclados **AT**.
d) No aceptan comandos de la interfaz de teclado.
e) El teclado es reseteado llevando conjuntamente a un nivel alto las **Líneas de Datos** y **de Reloj**.
f) El código para un **Make Code** (tecla presionada) se genera con el último bit del byte puesto a cero. Cuando en este bit se recibe un nivel lógico alto (1), se transmite un **Break Code** (código de tecla soltada), en suma, se obtiene un **Break Code** de un **Make Code** al sumar al primero un valor de **80H**.

Del último punto tratado, huelga decir que el código enviado no tiene nada que ver ni es concordante con el **Código ASCII**, razón por la cual se debe de encontrar un método para decodificar el código enviado por el teclado, para obtener el código **ASCII Estándar**.

Los códigos **Make Code** y **Break Code** para los teclados **XT** se muestran en la figura número 4.2.

El teclado **AT** se caracteriza sobre todo por permitir una comunicación bidireccional con la interfaz presente en la microcomputadora.

[2] Estos códigos se generan cuando una tecla es presionada (make code) y cuando es soltada (break code).

El protocolo de comunicación del teclado AT se basa en el formato serie controlado por sus propios **Impulsos de Reloj**, transmite a la par que los **Pulsos de Reloj** 11 bits. De los cuales, existe un bit de **Inicio** (transición de un nivel alto a un nivel bajo, de "1" a "0"), 8 bits de datos (con el bit menos significativo en primer lugar), un bit de **Paridad Impar** y un bit de **Parada** (transición de un nivel bajo a un nivel alto, "0" a "1").

La frecuencia de oscilación del reloj se encuentra dentro del orden de 20-30 KHz, variando de teclado a teclado.

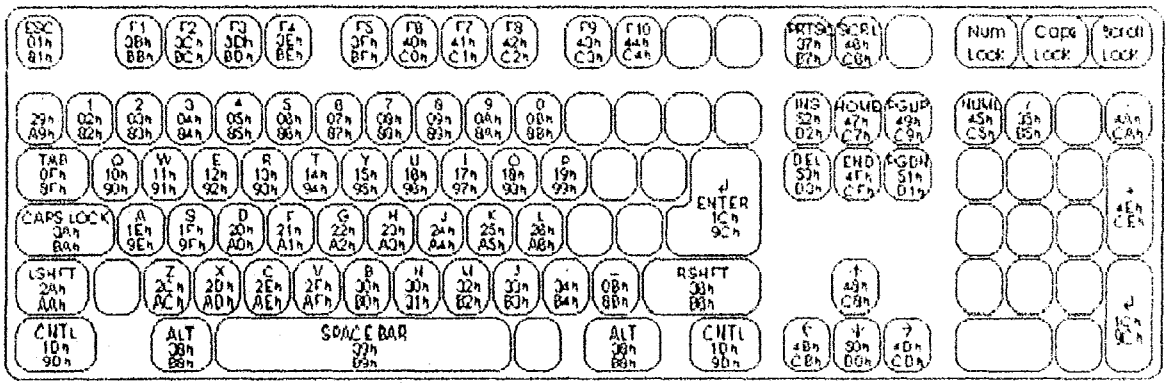


Fig. 4.2 DISPOSICIÓN FÍSICA Y CÓDIGOS DE "MAKE/BREAK" PARA LAS TECLAS DEL TECLADO XT

Dado que la cadencia de los bits de datos varía de teclado a teclado, es necesario utilizar una **Interfaz Controlada por Reloj** (**SPI** en términos de Motorola) a asíncronas (**SCI**) con una entrada de datos de un bit por **Pulso de Reloj**. Si existieran contratiempos al respecto, lo recomendable es conectar la **Señal de Reloj** del teclado a la línea $\overline{\text{INT}}$ del microprocesador y leer los datos bit a bit en el borde de bajada del **Pulso de Reloj**.

Las **Líneas de Datos** y **de Reloj** se implementan en los teclados con una salida de **Colector Abierto** y una resistencia **Pull-Up** conectada a la línea de +5 volts. Siendo recomendable e imprescindible conectar a tierra estas líneas, tal como puede verse en la siguiente figura.

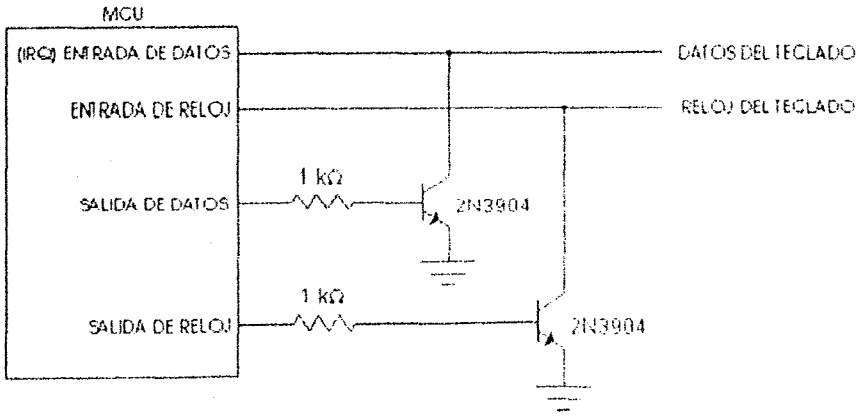


Fig. 4.3 INTERCONEXIÓN DEL TECLADO CON UN MICROCONTROLADOR

El teclado transmitirá pulsaciones de datos tantas veces como una tecla se encuentre presionada (o haya sido soltada) si las **Líneas de Datos** y **de Reloj** se encuentran en un nivel lógico alto (1). Si la **Línea de Reloj** se encuentra en un nivel lógico bajo (0), el teclado almacenará los datos en un buffer hasta que la **Señal de Reloj** se vaya a un nivel lógico alto nuevamente (la **Señal de Reloj** actúa como una línea **RTS (Request To Send)**. Si la **Línea de Datos** se encuentra en un nivel lógico bajo, el teclado se prepara para recibir 11 bits de datos de la computadora vía la interfaz.

Si la **Línea de Reloj** es puesta a un nivel bajo por la interfaz mientras el teclado se encuentra transmitiendo datos por al menos 60 μs, el teclado suspenderá la transmisión y se preparará para recibir los datos del teclado. Al finalizar la recepción de datos, el teclado procederá a retransmitir el dato que fue interrumpido.

La habilidad del teclado de permitir una comunicación bidireccional, permite a la interfaz controlar los LEDs de estado del teclado, tales como las teclas **Num Lock**, **Caps Lock** y **Scroll Lock**. La transmisión de datos desde la interfaz hacia el teclado se inicia enviando la **Línea de Reloj** a un nivel lógico bajo al menos durante 60 μS. Transcurrido este tiempo, la **Línea de Datos** se debe de poner a un nivel lógico bajo y la **Línea de Reloj** a un nivel lógico alto. Aproximadamente 10 milisegundos después, el teclado comenzará a generar las **Señales de Reloj**, en cada transición de nivel alto a bajo, el teclado va a transmitir un nuevo bit.

Después de que se ha verificado el bit de **Paridad**, la interfaz debe de resetear (enviando a un nivel lógico alto) la **Línea de Datos** y esperar a que el teclado envíe otro **Pulso de Reloj** (será la décima transición de un nivel alto a un nivel bajo, si no se toma en cuenta la transición inicial enviada por el controlador). El teclado pondrá entonces la **Línea de Datos** a un nivel bajo antes de que ocurra la decimoprimera transición del pulso de reloj para reconocer la recepción del byte de comando.

Si la **Línea de Datos** no es reseteada, entonces el teclado comenzará a emitir **Pulsos de Reloj** hasta que la **Línea de Datos** sea soltada. Al ocurrir esto, el teclado envía a un nivel lógico bajo la **Línea de Datos** y transmite un comando de **RESEND (FEH)**

La tabla 4.2 muestra los comandos que se pueden enviar al teclado y las que este puede enviar a la computadora mediante la interfaz (datos concernientes a un teclado bidireccional **AT**)

Comando	Origen	Destino	Acción
F5H	Interfaz	Teclado	Deshabilitar por default. Resetea el teclado, regresa un código **ACK** y suspende el escaneo de las teclas, espera por otro comando. Deja intacto los leds de estados del teclado.
EEH	Interfaz	Teclado	Petición de eco. Responde con un código de eco (EEH).
F4H	Interfaz	Teclado	Comando habilitar. Borra el buffer de salida, habilita el teclado y envía un código **ACK**.
F2H	Interfaz	Teclado	Lectura de ID. Responde con un código **ACK** y dos bytes de ID (83H y ABH), rehabilita el escaneo incluso si estaba deshabilitado.
FEH	Interfaz	Teclado	Retransmisión. Retransmite el último byte datos (**Scan Code**) enviado.
FFH	Interfaz	Teclado	Reset. Resetea el **CPU** del teclado, inicia el chequeo de encendido, responde con un byte de chequeo de encendido.
F0H	Interfaz	Teclado	Selección de conjunto de **Scan Code**. Responde con **ACK** y espera hasta que el microcontrolador envíe un byte (01, 02 ó 03) especificando el conjunto de **Scan Code** a utilizar. Si un código 00 es enviado, el teclado responde con **ACK** seguido por el conjunto de **Scan Code** utilizado actualmente.
F7H	Interfaz	Teclado	Activa todos los **Typematic** de las teclas, responde con un **ACK**.
F8H	Interfaz	Teclado	Activa todos los **Make/Break** de las teclas, responde con un **ACK**.
F9H	Interfaz	Teclado	Activa todos los **Make** de las teclas, responde con un **ACK**.
FAH	Interfaz	Teclado	Activa todos los **Typematic/Make/Break**, también controla los **Scan Code** transmitidos desde el teclado, responde con un **ACK**.
F6H	Interfaz	Teclado	Activación por default. Responde como el comando **F5H** pero no inhibe el scaneo de las teclas. De igual forma no afecta el estado de los LEDs indicadores de estado.
FBH	Interfaz	Teclado	Activa el **Typematic** de una tecla.
FCH	Interfaz	Teclado	Activa el **Make/Break** de una tecla.
EDH	Interfaz	Teclado	Activa/desactiva los indicadores de estado. Este comando le permite controlar los LEDs de estado del teclado. El teclado responde con un código **ACK** y espera un byte de opción, el cual es codificado en un byte como sigue: b0=**Scroll Lock**, b1=**Num Lock**, b2=**Caps Lock**, b3 ... b7=0. Un nivel lógico 1 en los bits anteriores activa el LED, un nivel lógico 0 los desactiva.
F3H	Interfaz	Teclado	Activa la razón/retardo de autorepetición. El teclado responde con un **ACK** y espera un byte que indique la razón o el retardo de repetición, el cual es codificado en un byte como sigue: b7=no usado, b6...b5=retardo de la repetición (00=250 ms, 11=1000 ms), b4...b0=Razón de repetición (0000=30x/Seg, 1111=2x/Seg), el teclado responde con un código **ACK** después de recibir cada byte de opción.
FAH	Teclado	Interfaz	**ACK** (reconocimiento), respuesta a muchos comandos.
AAH	Teclado	Interfaz	Chequeo de encendido con éxito.
EEH	Teclado	Interfaz	Respuesta del comando **ECO** (**EEH**).
00H ó FFH	Teclado	Interfaz	Detección de error: de una tecla, también indica un overflow en el buffer.
FEH	Teclado	Interfaz	Re-envío: Envía en respuesta a la recepción de un código de comando invalidado o un código con un bit de paridad erróneo.

Tabla 4.2 COMANDOS DISPONIBLES EN LOS TECLADOS AT

El controlador de teclado más conocido es el circuito integrado 8042 de la compañía Intel. A grandes rasgos las características de este microcontrolador como interfaz de teclado son:

a) El circuito integrado 8042 tiene dos líneas de **Colector Abierto** llamados **Reloj** y **Datos**
b) El teclado tiene las mismas características que el circuito integrado 8042.
c) Ambos extremos pueden manejar las líneas de comunicación bidireccional al mismo tiempo. Efectuando este proceso mediante interrupciones.
d) Cuando la microcomputadora desea enviar información al teclado, éste envía un comando al chip 8042, el cual le indica que el siguiente dato que recibirá del **CPU** deberá ser enviado sin modificación al teclado a través de la **Línea de Datos** seriales.
e) Las líneas de comunicación bidireccional están en estado inactivo al encontrarse en un nivel lógico alto (1).
f) El chip 8042 envía a un nivel bajo la **Línea de Datos** para inhibir cualquier transmisión del teclado.
g) Cuando el chip 8042 desea transmitir un dato al teclado, envía la **Línea de Datos** a un nivel lógico bajo (0).
h) El 8042 libera la **Línea de Reloj** y espera a que el teclado envíe la **Línea de Reloj** a un nivel lógico bajo. Cuando ha ocurrido esto, el circuito integrado 8042 coloca sus primeros bits de datos en la **Línea de Datos**.
i) A cada transición de un nivel lógico alto a un nivel lógico bajo de la **Señal de Reloj**, es colocado un bit de datos en la **Línea de Datos**.
j) El último bit en el último **Pulso de Reloj** es un bit de datos que regresa el teclado. Después de que el controlador ve este último bit, inhibe el teclado poniendo la **Línea de Reloj** a un nivel lógico bajo.
k) Durante la transmisión, el controlador puede abortar la operación manteniendo la **Línea de Reloj** en un nivel lógico bajo.
l) Para que se pueda transmitir datos del teclado al chip 8042, es menester mantener las **Líneas de Datos y de Reloj** en un nivel lógico alto. Si ambas líneas no están en alto, el teclado no puede enviar datos. Si la **Línea de Reloj** se mantiene en un nivel lógico bajo por el controlador, éste no desea ningún dato, si la **Línea de Datos** se mantiene en un nivel lógico bajo por el mismo controlador, este desea enviar datos al teclado el cual debe de comenzar a recibirlos con los **Pulsos de Reloj**.
m) El teclado recibe los datos en forma inversa a como los manda, es decir, se envía un nuevo dato cuando el **Pulso de Reloj** tiene una transición de un nivel lógico bajo a un nivel lógico alto ("0" a "1"). Las figuras 4.6.a y 4.b muestran en forma visual lo explicado anteriormente.

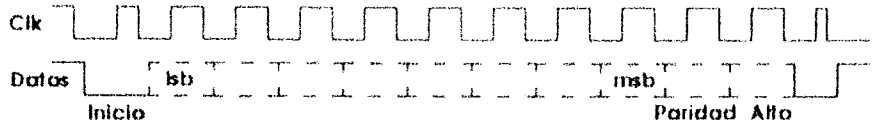


Fig. 4.6 a) TRANSMISIÓN DE DATOS DEL TECLADO AL MICROCONTROLADOR 8042

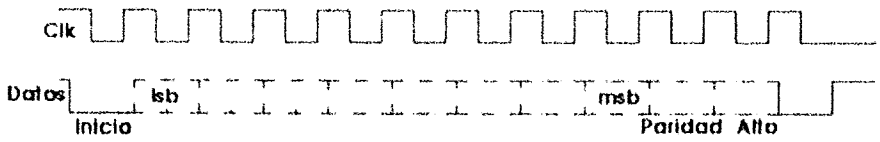


Fig. 4.6 b) TRANSMISIÓN DE DATOS DEL MICROCONTROLADOR 8042 AL TECLADO

4.2.3 Diseño del Controlador de Teclado

Los códigos de **Make Code** y **Break Code** para el teclado **AT/XT** que se va a utilizar en la implementación del sistema mínimo son:

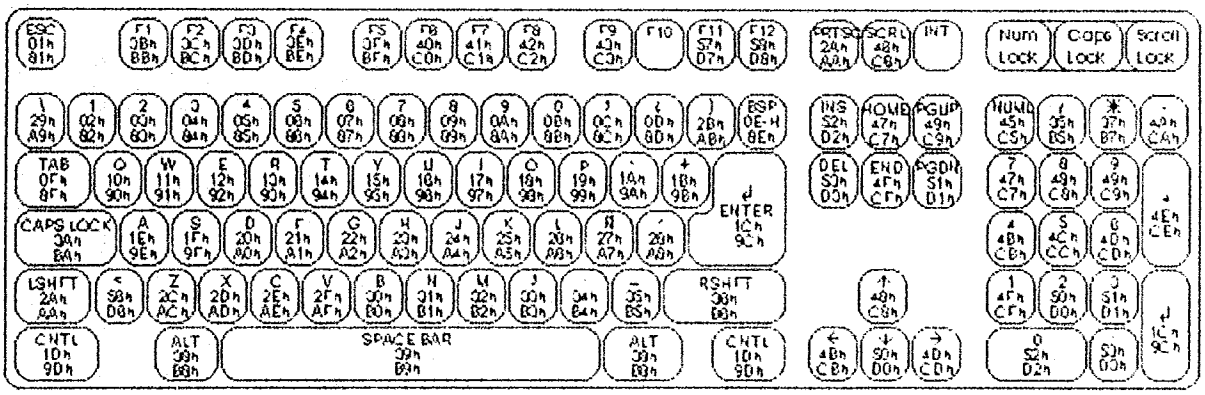


Fig. 4.7 DISPOSICIÓN FÍSICA Y CÓDIGOS DE "MAKE/BREAK" PARA EL TECLADO USADO EN EL SISTEMA MÍNIMO

Como se había mencionado anteriormente y debido a los bajos requerimientos de costos y simplicidad de diseño, en lugar de utilizar circuitos integrados especiales o microcontroladores, se decidió por la implementación de una interfaz de teclado con una serie de circuitos integrados TTLs comunes y corrientes. La gráfica número 4.8 muestra el diseño de la interfaz de teclado unidireccional, la cual tiene una teoría de funcionamiento como se describe a continuación:

1. La **Señal de Reloj** proveniente del teclado es retardada dos periodos de reloj del sistema e invertida por un par de flip-flops tipo latch (74LS175).
2. La nueva **Linea de Reloj** es usada para desplazar los bits de datos del teclado dentro de un registro de desplazamiento serie-paralelo (74LS322). Los desplazamientos son controlados por el microprocesador 8048 del teclado.

3. Cuando los 8 bits de datos han sido introducidos, la línea A/QA es invertida para proporcionar una señal $\overline{INT}$ hacia el microprocesador.
4. La señal $\overline{INT}$ indica al microprocesador que un byte de datos se encuentra en el buffer, y que debe de leerlo.

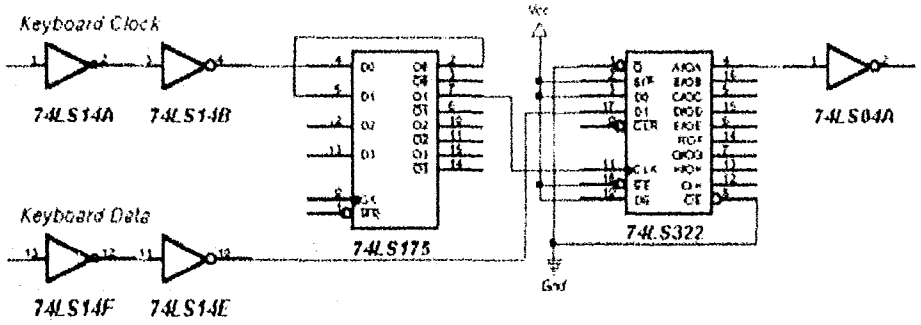


Fig. 4.8 INTERFAZ TECLADO-CPU

Debido a su naturaleza triestado, el circuito integrado 74LS322 puede ser conectado directamente al **Bus de Datos** del sistema, simplificando el hardware y el software necesarios para su control.

Habiendo probado la interfaz del teclado, el siguiente paso es determinar la forma en que se va a controlar las teclas especiales del teclado, tal como la tecla **CAPS-LOCK**. Para este fin, se puede controlar directamente por el programa almacenado en la **Memoria de Programa** del microprocesador. Sin embargo, esto conduce al rápido incremento del tamaño de dicho programa. Tomando también en cuenta la limitada capacidad de direccionamiento del microprocesador Z80, nos enfocaremos en el diseño de un circuito que permita controlar esta tecla.

El diseño del circuito que se encarga de verificar el estado de la tecla **CAPS-LOCK** se muestra en la siguiente figura:

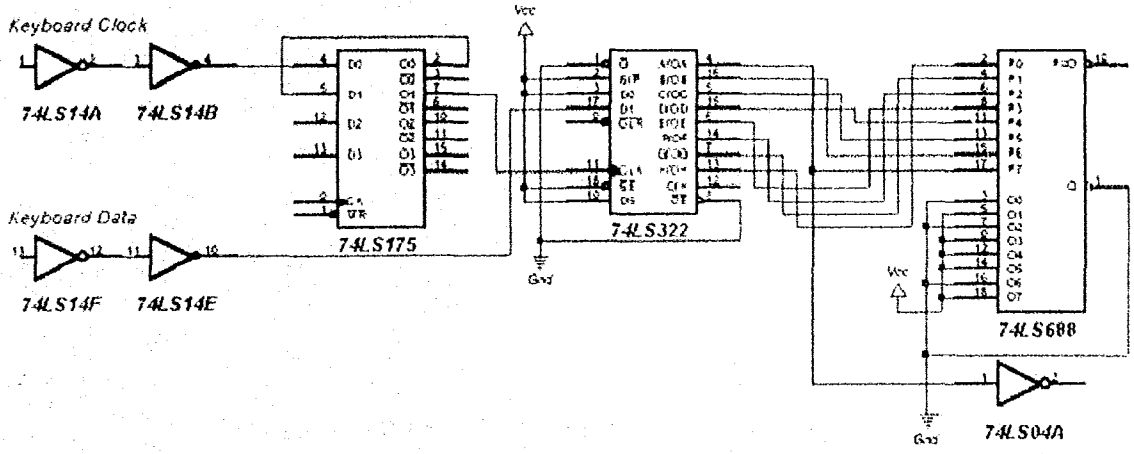


Fig. 4.9 INTERFAZ TECLADO-CPU CON CONTROL DE LA TECLA CAPS-LOCK

Como se puede ver, el circuito que se encarga de verificar el estado de la tecla **CAPS-LOCK** consiste en su parte más esencial por un circuito integrado comparador de 8 bits (74LS688). Una de las líneas del byte de datos se conecta directamente con las salidas del registro de desplazamiento (74LS322), y la otra parte del byte de datos se conecta directamente a las fuentes de alimentación con un arreglo tal que los datos son comparados con el código de la tecla **CAPS-LOCK (BAH)**. Esto significa que cuando la tecla mencionada es presionada y posteriormente liberada, el comparador genera una señal de bajo nivel lógico ($\overline{P=Q}$). Esta señal es usada como **Strobe de CAPS-LOCK** activada.

Gracias al método anteriormente mencionado, se puede ahorrar una gran cantidad de bytes de **Memoria de Programa** para otros propósitos. Sin embargo, como se mencionó anteriormente, los bytes de datos que emite el teclado no están codificados en código **ASCII**, razón por la cual se debe diseñar otro circuito análogo al que se utilizó para verificar el estado de la tecla **CAPS-LOCK**. La función de dicho circuito será la de decodificar las señales enviadas por el teclado a código **ASCII** que puedan ser manipuladas directamente por el microprocesador.

El diseño del circuito que se mencionó se muestra en la siguiente figura:

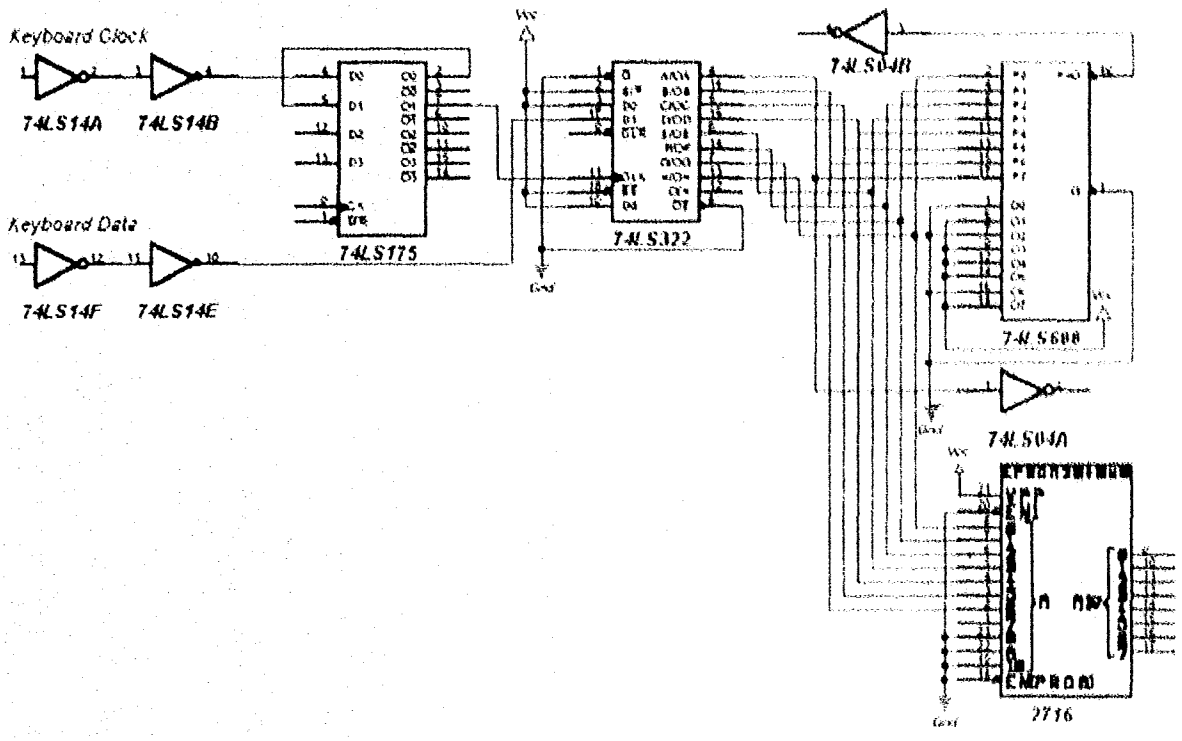


Fig. 4.10 INTERFAZ TECLADO-CPU CON CONTROL DE CAPS-LOCK Y DECODIFICADOR ASCII

En este circuito, la memoria **ROM 2716** cumple la función de almacenar en sus celdas el **Código ASCII** de los datos enviados por el teclado. En la mayoría de los casos el código decodificado de las teclas presionadas se duplica, excepto para el **Código ASCII** de los caracteres alfabéticos. Puesto que el **Código ASCII** para éstos últimos son distintos y deben ser tomados en cuenta dada la forma de diseño de la interfaz de teclado con control de la tecla **CAPS-LOCK**. Para propósitos de este trabajo, se utilizaron las primeras **70H** localidades de memoria para almacenar el **Código ASCII** de los números, caracteres especiales y las letras minúsculas. A partir de la localidad de memoria **80H** se almacenaron los **Códigos ASCII** duplicado de los números, caracteres especiales y letras mayúsculas.

Gracias a las siete líneas de decodificación se puede decodificar hasta 128 teclas, suficientes para proporcionar un **Código ASCII** a cada tecla. Sin embargo, también se debe tomar en cuenta la diferencia entre las letras mayúsculas y minúsculas que se mencionó anteriormente. De igual forma, debe tomarse en consideración la señal de **CAPS-LOCK** activada que se mostró en el apartado anterior.

Para conseguir los objetivos mencionado anteriormente, se necesita recurrir al auxilio de la lógica de diseño de sistemas secuenciales, con el objeto de poder diseñar un sistema secuencial que se adapte a los requerimientos previstos.

La figura número 4.10 muestra el diagrama de flujo que ha de seguir el sistema secuencial.

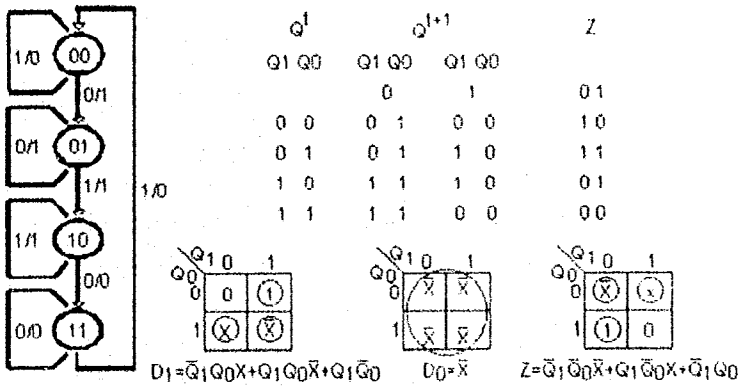


Fig. 4.10 DIAGRAMA DE FLUJO DEL SISTEMA SECUENCIAL

Habiendo determinado el diagrama de flujo del sistema secuencial, el siguiente paso es determinar el tipo de flip-flop a utilizar. De igual forma y por razones de simplicidad, los circuitos a considerar son los flip-flops tipo D, tal como el 74LS175. Además, dado que ya contamos con un circuito integrado especificado en el procedimiento anterior, nos valdremos de él para elaborar el circuito que determina la secuencia de flujo del sistema secuencial.

La siguiente figura muestra el diagrama final del circuito interfaz de teclado, con todas sus características de control mencionadas anteriormente:

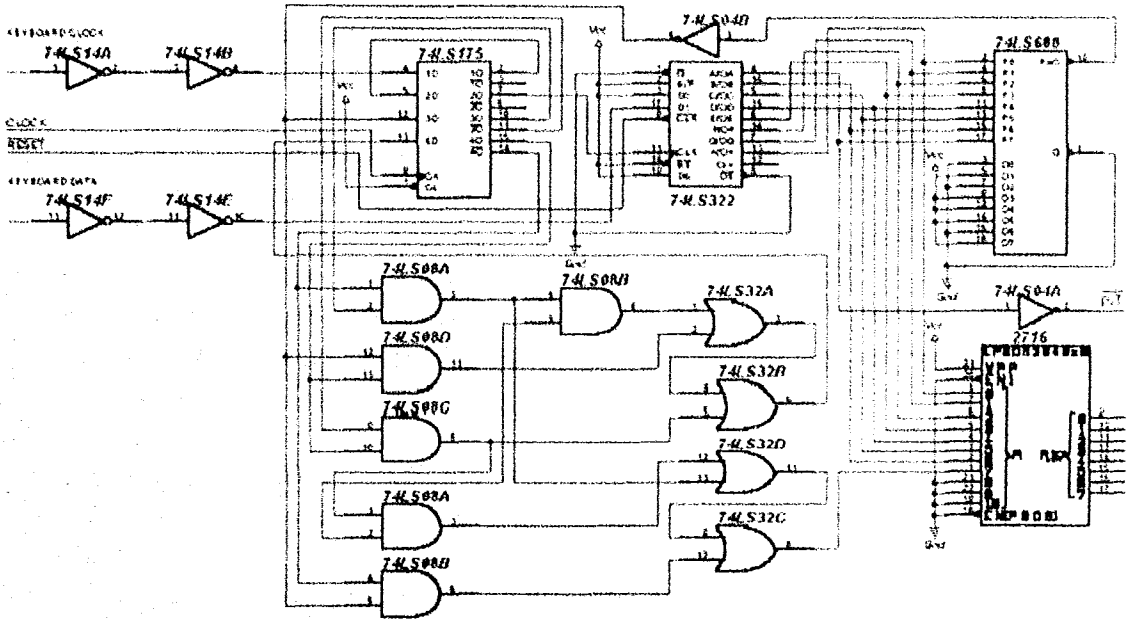


Fig. 3.12 DIAGRAMA FINAL DE LA INTERFAZ UNIDIRECCIONAL TECLADO-CPU

4.3 Monitores

Debido al objetivo del presente trabajo, el diseño del hardware y el software para el control de un monitor normal cae fuera del presupuesto de un estudiante, razón por la cual se requiere de un dispositivo más económico y de fácil implementación: las **Pantallas de Cristal Liquido** o **LCD** (término derivado de las palabras anglosajonas **Liquid Crystal Display**). Debido a estas y otras razones, nos enfocaremos en el diseño de un control para una pantalla **LCD**.

4.3.1 Breve Fundamento Histórico

Los materiales de **Cristal Líquido** (término sugerido por el físico O. Lehman en 1889) fueron descubiertos en 1888 por un botánico Austriaco llamado Friedrich Rheinitzer, sin embargo solo en los últimos 25 años estos materiales han sido desarrollados lo suficiente como para ser usados en las pantallas. El **Cristal Líquido** es un término que indica el estado de una sustancia que no es ni sólido ni líquido. Las moléculas son típicamente **Moities** orgánicos en forma de barra de alrededor de 25 armstrongs de longitud y su orden está en función de la temperatura. En 1963, mientras trabajaba para la compañía RCA, un investigador llamado Williams descubrió que la forma en que la luz pasa a través del **Cristal Líquido** cambia cuando éste es estimulado por un cambio de corriente eléctrica. Cinco años después, otro investigador de la RCA, llamado Heilmeyer y sus colegas crearon un prototipo de pantalla en el cual se aplicaban estos conceptos. El éxito de este

prototipo marcó el inicio de la tecnología moderna de las **Pantallas de Cristal Liquido** (LCD). Al principio, las **Pantallas de Cristal Liquido** fueron muy inestables para ser utilizadas como material de manufactura de las unidades de **Display**, creando varios problemas de mercado para la tecnología de los **LCDs**, hasta que un profesor de la universidad de Hull (George W. Gray) en el Reino Unido hizo un importante aporte científico al descubrir un material de **Cristal Liquido** estable: el **Biphenil**, pilar fundamental de la tecnología de los futuros **LCDs**. En 1973 se introdujo el EL-8025 (figura 4.13), la primer calculadora electrónica del mundo con características de **Pantalla LCD**. La tecnología siguió el desarrollo del EL-8025, la cual aún forma parte de la base de los productos **LCD** actuales.

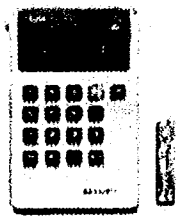


Fig. 4.13 PRIMER CALCULADORA CON LCD

Las aplicaciones para los **LCDs** se extienden a las calculadoras, relojes, procesadores de palabras, televisores, sistemas de video y otros (Fig. 4.14).

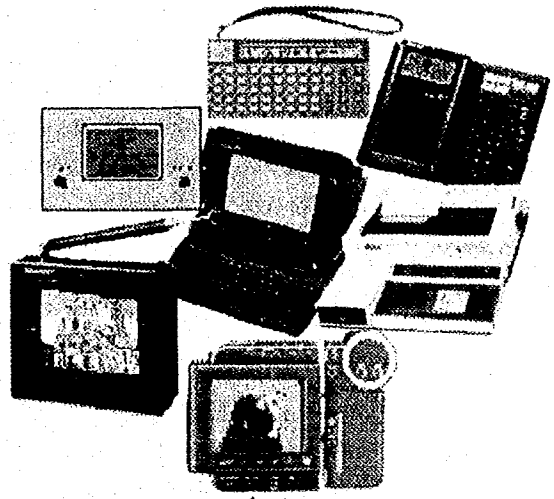


Fig. 4.14 APLICACIONES TÍPICAS DE LAS PANTALLAS LCD

La gran expansión de los **LCDs** es el resultado de los recientes descubrimientos hechos en las siguientes cuatro áreas de la tecnología de los **LCDs**:

1. Capacidades avanzadas de mostrar sólo caracteres alfanuméricos a gráficos.
2. Capacidades avanzadas de manejar sólo tonalidades monocromáticas a colores.

3. Características avanzadas de mostrar solo imágenes estáticas a visualizar gráficos en movimiento
4. Capacidades avanzadas de cambiar desde pequeñas pantallas hasta grandes pantallas.

Hasta fechas muy recientes, muchos dispositivos visualización de datos usaban **Tubos de Rayos Catódicos (CRT)**. Los **CRTs** sin embargo, requieren altos niveles de voltaje para la emisión y el control del ángulo de los electrones. La tecnología de las pantallas de panel plano son la respuesta perfecta a las demandas para pantallas más compactas con gran ahorro de energía e incremento de la eficiencia.

Las pantallas de panel plano están siendo desarrolladas bajo los siguientes objetivos:

a) Unidades más pequeñas.
b) Bajo voltaje de operación y bajo consumo de potencia (objetivos no fácilmente logrados con los **CRTs**).

Existe una gran diversidad de sistemas con diferentes tipos de materiales y métodos de desplegado que pueden usarse:

1) **LCD (Liquid Cristal Display).**
2) **PDP (Plasma Display).**
3) **LED (Light Emisor Display).**
4) **EL (Electroluminiscent) Panel.**
5) **VFD (Vaccum Fluorescent Display).**
6) **DMD (Digital Micromirror Device).**
7) **FED (Field Emision Display).**

De todos estos sistemas, las **Pantallas de Cristal Líquido** son preferidas por considerarse las más prometedoras. Por ser más pequeñas y ligeras, estas pantallas funcionan con bajos voltajes que pueden ser manejadas por dispositivos **LSI**. Dado que consumen poca potencia, pueden ser alimentadas por grandes periodos de tiempo con baterías. Otras ventajas sobre otros diversos tipos de pantallas planas incluyen:

a) Adaptabilidad a colores reales.
b) Bajo costo.
c) Gran potencial para departamentos de desarrollo.

En el pasado, la calidad de los gráficos en las pantallas **LCDs** eran consideradas inferiores a los **CRTs**. De igual forma la fabricación de **LCDs** mayores a 10 pulgadas no era posible. Esto cambió dramáticamente en 1988 cuando un prototipo con las siguientes características fue introducido:

a) Pantalla de 14 pulgadas.
b) 27 mm de densidad.
c) 1.8 Kg de peso.
d) Calidad de gráficos comparable a los monitores **CRTs**.

4.3.2 Principios de Operación de los LCDs

Las moléculas de **Cristal Líquido** tienden a estar en fórma desordenada de acuerdo con su eje longitudinal, a menos que se pongan en una superficie acanalada en una dirección fija. Las moléculas de **Cristal Líquido** se alinean en forma paralela a los canales.

Fig. 4.15 ALINEACIÓN DE LAS MOLÉCULAS DE CRISTAL LÍQUIDO

Cuando los las moléculas de **Cristal Líquido** son intercaladas entre una placa superior y una inferior, éstas se alinean con los canales en las direcciones a y b, respectivamente. Forzando a las moléculas de **Cristal Líquido** a efectuar una torsión de 90 grados.

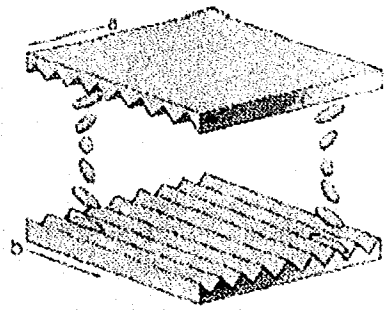


Fig. 4.16 ALINEACIÓN DE MOLÉCULAS EN SUPERFICIES PERPENDICULARES

La luz pasa entre el arreglo de moléculas de Cristal Líquido, siguiendo la dirección de las éstas. Cuando las moléculas de **Cristal Líquido** giran 90 grados, la luz efectúa el mismo giro de 90 grados para pasar entre las moléculas.

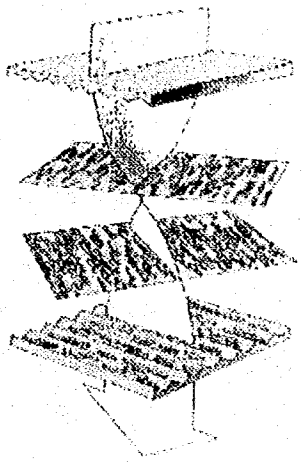


Fig. 4.17 PASO DE LA LUZ ENTRE LAS MOLÉCULAS DE CRISTAL LÍQUIDO

Cuando se aplica un voltaje, las moléculas de **Cristal Liquido** se acomodan verticalmente, acorde con el campo eléctrico, de esta forma la luz pasa en forma lineal entre las moléculas de **Cristal Liquido**.

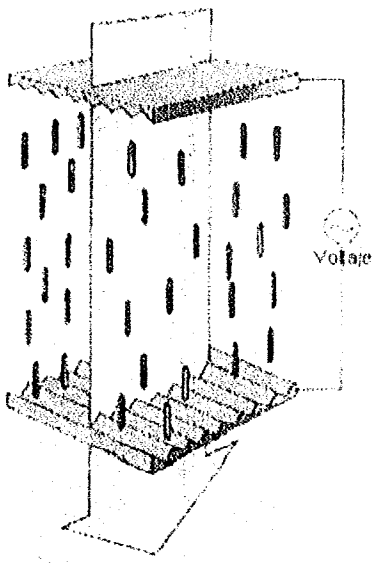


Fig. 4.18 ALINEACIÓN VERT. DE LAS MOLÉCULAS DE CRISTAL LÍQUIDO

Cuando el voltaje se aplica a una combinación de **Filtros Polarizadores** y moléculas de **Cristal Líquido** torcidos, se forma una **Pantalla de Cristal Líquido**.

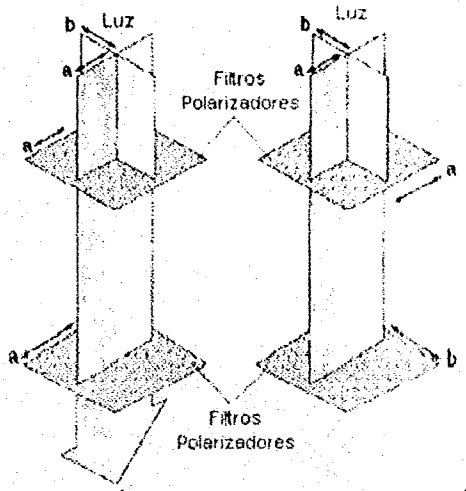


Fig. 4.19 FORMACIÓN DE UNA PANTALLA DE CRISTAL LÍQUIDO

Si los **Filtros Polarizadores** se arreglan como muestra la figura a la izquierda, la luz pasa. Cuando el arreglo se realiza en la forma indicada en la figura a la derecha, la luz es bloqueada.

La estructura helicoidal mostrada en la figura anterior tiene la habilidad de controlar la luz Cuando se polariza aleatoriamente, la luz pasa por el polárizador frontal y se polariza linealmente De esta forma, al atravesar el vidrio frontal, la luz se rota por las moléculas de **Cristal Líquido** y atraviesa el vidrio posterior. Si el **Analizador** se rota 90 grados con respecto al **Polarizador**, la luz pasa entre el **Analizador**, siendo reflejado a la celda. El observador verá el fondo de la pantalla, que en este caso será un fondo gris plateado.

Cuando un **Controlador** de señal apropiado se aplica a los electrodos, se genera un campo eléctrico entre las celdas. Las moléculas de **Cristal Líquido** son rotadas en la dirección del campo eléctrico. La luz polarizada linealmente pasa entre las celdas no afectadas, siendo absorbida por el **Analizador** posterior. El observador ve un carácter negro en un fondo gris plateado. Cuando se retira el campo eléctrico, las moléculas de **Cristal Líquido** regresan a su estado anterior de torsión (90 grados). Como se puede visualizar en la siguiente figura número:

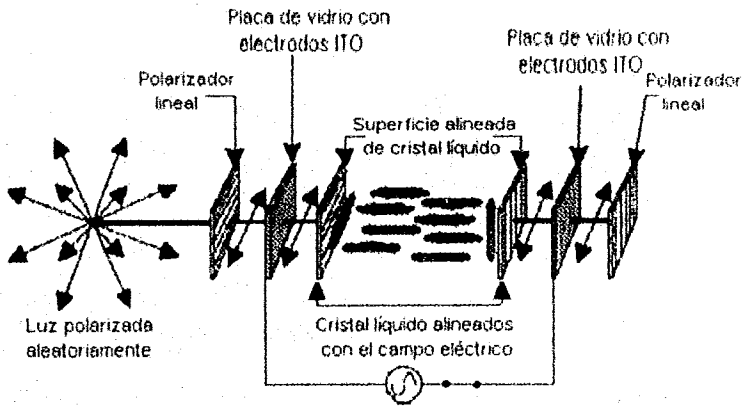


Fig. 4.20 ESTRUCTURA FUNCIONAL DE UNA PANTALLA LCD

4.3.3 Construcción de las Pantallas LCD

Como se ha mencionado anteriormente, una **Pantalla de Cristal Líquido** está compuesto de múltiples capas. Primero, una hoja de vidrio se reviste con una película de óxido metálico transparente el cual actúa como un electrodo. Esta película puede ser modelada para formar los renglones y las columnas de una pantalla de **Matriz Pasiva** o el pixel individual de una pantalla de **Matriz Activa**. Estos electrodos son usados para activar el voltaje entre las celdas necesaria para la transición de la orientación. Posteriormente, se aplica una capa de **Polímero** alineado. Esta capa experimenta un proceso de frotamiento el cual deja una serie de surcos microscópicos paralelos en la película. Estos surcos ayudan a alinear las moléculas de **Cristal Líquido** en una dirección referida, con su eje longitudinal paralelo a los surcos Esto obliga a las moléculas alinearse conforme a las capas y ayuda a forzar el torcimiento de las moléculas de Cristal Líquido entre las capas de alineación. Dos hojas de vidrio son preparadas, una de ellas es revestida con una capa de granos de polímero. Estos granos mantienen un hueco uniforme entre las hojas de

La estructura helicoidal mostrada en la figura anterior tiene la habilidad de controlar la luz Cuando se polariza aleatoriamente, la luz pasa por el polarizador frontal y se polariza linealmente De esta forma, al atravesar el vidrio frontal, la luz se rota por las moléculas de **Cristal Líquido** y atraviesa el vidrio posterior. Si el **Analizador** se rota 90 grados con respecto al **Polarizador**, la luz pasa entre el **Analizador**, siendo reflejado a la celda. El observador verá el fondo de la pantalla, que en este caso será un fondo gris plateado.

Cuando un **Controlador** de señal apropiado se aplica a los electrodos, se genera un campo eléctrico entre las celdas. Las moléculas de **Cristal Líquido** son rotadas en la dirección del campo eléctrico. La luz polarizada linealmente pasa entre las celdas no afectadas, siendo absorbida por el **Analizador** posterior. El observador ve un carácter negro en un fondo gris plateado. Cuando se retira el campo eléctrico, las moléculas de **Cristal Líquido** regresan a su estado anterior de torsión (90 grados). Como se puede visualizar en la siguiente figura número:

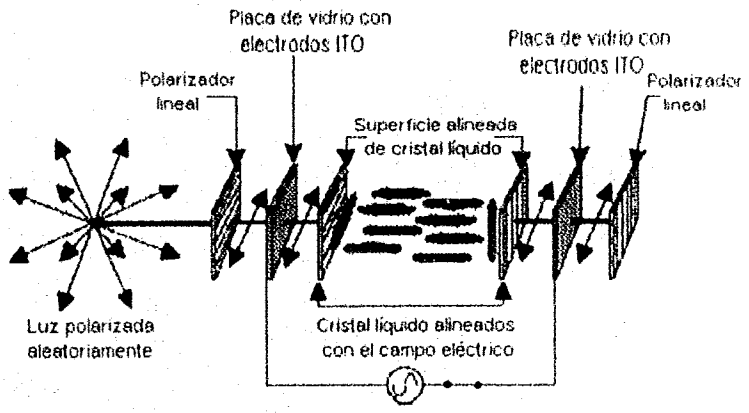


Fig. 4.20 ESTRUCTURA FUNCIONAL DE UNA PANTALLA LCD

4.3.3 Construcción de las Pantallas LCD

Como se ha mencionado anteriormente, una **Pantalla de Cristal Líquido** está compuesto de múltiples capas. Primero, una hoja de vidrio se reviste con una película de óxido metálico transparente el cual actúa como un electrodo. Esta película puede ser modelada para formar los renglones y las columnas de una pantalla de **Matriz Pasiva** o el pixel individual de una pantalla de **Matriz Activa**. Estos electrodos son usados para activar el voltaje entre las celdas necesaria para la transición de la orientación. Posteriormente, se aplica una capa de **Polímero** alineado. Esta capa experimenta un proceso de frotamiento el cual deja una serie de surcos microscópicos paralelos en la película. Estos surcos ayudan a alinear las moléculas de **Cristal Líquido** en una dirección referida, con su eje longitudinal paralelo a los surcos Esto obliga a las moléculas alinearse conforme a las capas y ayuda a forzar el torcimiento de las moléculas de Cristal Líquido entre las capas de alineación. Dos hojas de vidrio son preparadas, una de ellas es revestida con una capa de granos de polímero. Estos granos mantienen un hueco uniforme entre las hojas de

vidrio donde las moléculas de **Cristal Líquido** son colocadas. Las dos hojas de vidrio son entonces colocadas juntas y los bordes sellados con **Epoxy**. Una esquina se deja sin sellar de tal forma que las moléculas de **Cristal Líquido** puedan ser inyectados dentro del tubo. Una vez que la pantalla ha sido llenada de moléculas de **Cristal Líquido**, el borde es sellado y los **Polarizadores** son aplicados a la superficie de vidrio. En una pantalla **TN(Twisted Nematic)**, las capas de alineación están posicionadas con su dirección de frotamiento perpendicular y los **Polarizadores** son aplicados para hacer coincidir la orientación de las capas de alineación. En una pantalla **STN (Super-Twisted Nematic)** las capas de alineación son colocadas con su dirección de frotamiento en una variedad de ángulos para crear una torsión de 180 a 270 grados, los polarizadores no son aplicados paralelos a las capas de alineación. La forma de unión de las diversas capan que contiene una pantalla **LCD** se muestra en la siguiente figura número:

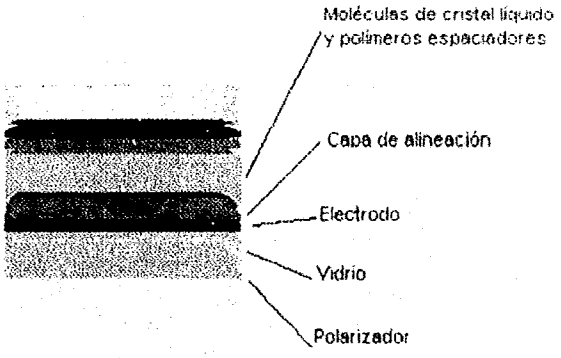


Fig. 4.21 PROCESO DE CONSTRUCCIÓN DE UN LCD

La pantalla es finalmente terminada completando la conexión del circuito controlador el cual controla el voltaje aplicado a varias áreas del **Display** (pixeles).

4.3.4 Direccionamiento de las Pantalla LCD

El **Direccionamiento** es el proceso mediante el cual los pixeles son activados o desactivados para formar una imagen. Existen dos métodos principales de **Direccionamiento**: **Directo** y **Multiplexado**. El **Direccionamiento Directo** es conveniente para pantallas donde existen pocos elementos que van a ser activados. Con **Direccionamiento Directo** cada pixel en la pantalla tiene su propio circuito de control. Un microprocesador debe aplicar individualmente un voltaje a cada elemento. Una aplicación común del **Direccionamiento Directo** son las pantallas de **Cristal Liquido** de siete segmentos, los cuales se encuentran en relojes y dispositivos similares. En un **Direccionamiento Multiplexado** se encuentra envuelto un gran número de pixeles. Cuando los elementos están en un orden regular, pueden ser direccionados por sus renglones y columnas en lugar de controlar cada elemento separadamente. Esto reduce la complejidad del circuito porque cada pixel no necesita su propio controlador. Sin embargo, si se usa **Direccionamiento**

Multiplexado, se necesita 20 controladores, uno para cada renglón y uno para cada columna. Esto es tremendamente desventajoso, especialmente en pantallas que serán muy grandes.

4.3.5 Pantallas de Matriz Activa VS Pantallas de Matriz Pasiva

Las pantallas de **Matriz Pasiva** son direccionadas por un conjunto de electrodos multiplexores transparentes perpendiculares uno de otro, ubicados bajo y sobre la capa de **Cristal Liquido** en una formación de renglón-columna. Los electrodos están construidos típicamente de **Óxido de Lata de Indio (ITO)** el cual es un material conductor semitransparente.

Un **Pixel Pasivo** se direcciona cuando existe un suficiente voltaje que causa que las moléculas del **Cristal Liquido** se alineen paralelas al campo eléctrico. Una pantalla puede tener más de un pixel en cualquier momento por el tiempo de respuesta del material de **Cristal Liquido**. Cuando se direcciona un pixel por un periodo de duración de tiempo corto, las moléculas de **Cristal Liquido** se alinean de tal forma que crean un **Pixel Opaco**. Cuando se remueve el voltaje, el pixel se asemeja a un capacitor descargando, se apaga lentamente hasta disipar la carga y las moléculas de **Cristal Liquido** retornan a su orientación indefinida.

Por su tiempo de respuesta, una pantalla puede escanear la matriz de pixeles y activar los pixeles apropiados para formar una imagen. Como el tiempo para escanear la matriz es más corto que el tiempo de activación, una imagen de múltiples pixeles puede ser desplegado.

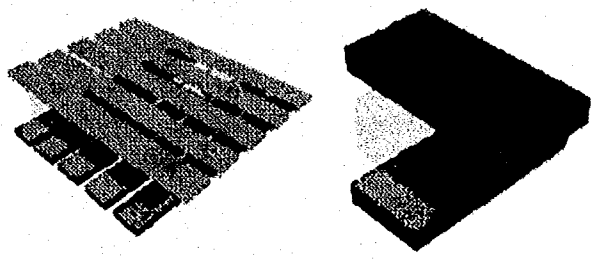


Fig. 4.22 ESTRUCTURA DE UN LCD DE MATRIZ PASIVA

En las pantallas de **Matriz Activa**, el **Direccionamiento** toma lugar detrás de la pelicula de **Cristal Liquido**. La superficie frontal de la pantalla se reviste con un electrodo continuo mientras que la parte posterior de la superficie del electrodo se modela en pixeles individuales. Una **Delgada Pelicula de Transistor (TFT)** actúa como un switch para cada pixel. El **TFT (Thin Field Transistor)** se direcciona por un conjunto de delgados electrodos multiplexados (lineas de **Compuerta** y de fuente) corriendo a lo largo de boquetes entre los pixeles. Un pixel se direcciona aplicando corriente a una linea de compuerta el cual switchea el TFT y permite cambiar la linea de **Fuente** para fluir en la parte posterior del electrodo. Esto activa un voltaje entre los pixeles y lo activa. Una imagen se crea en forma similar a las pantallas de **Matriz Pasiva:** escaneando la matriz. Una pantalla de **Matriz Activa** no sufre las limitaciones de las pantallas de **Matriz Pasiva**. Puede ser visto hasta en un ángulo de 45 grados, además de que tiene un contraste de 40:1, esto

indica que la brillantez de un **Pixel Activo** es 40 veces más grande que un **Pixel Inactivo**. Sin embargo, las pantallas de **Matriz Activa** requieren un sistema de luz más intensa dado que las líneas de **Compuerta** y de **Fuente** de los TFT no son muy transparentes bloqueando una fracción de la luz.

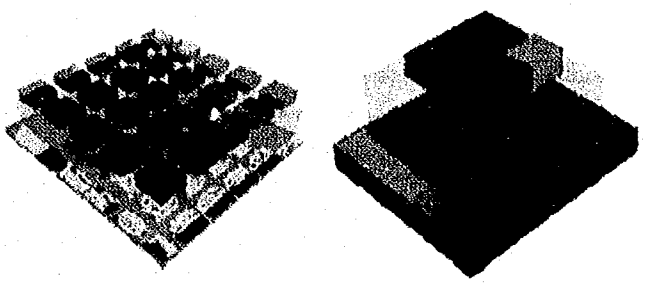


Fig. 4.23 ESTRUCTURA DE UN LCD DE MATRIZ ACTIVA

4.3.6 Pantallas de Colores

Para tener alcance a un color, es necesario primero tener una pantalla que sea de color blanco en un instante y negro en el otro. Esto es porque algunos tipos de pantalla (display antiguos tipo **STN** por ejemplo) pueden tener un color amarillo sobre un color azul, apariencia que no va a permitir producir todos los colores. En una pantalla blanca, todas las longitudes de onda pasan y sin embargo, todas las longitudes de onda pueden ser manipuladas para crear el color deseado. Para obtener todos los colores, cada pixel individual se divide en tres subpixeles: **Rojo**, **Verde** y **Azul** (**RGB**). Esto es, para cada color del pixel se emplean los tres distintos tipos de pixel. Estos subpixeles se crean aplicando filtros de color, los cuales únicamente permiten pasar cierta longitud de onda mientras que absorben el resto. Con una combinación de color rojo y color verde (subpixeles) de varias intensidades, un pixel puede tener la apariencia de distintos colores. Esto es análogo al manejo del color en los **Tubos de Rayos Catódicos** (**CTR**) de una televisión o de un monitor de computadora en el cual diferentes resplandores de fósforo de color verde, rojo y azul son excitados por cañones de electrones. El número de colores que puede se crean se logra a través de la combinación de los subpixeles rojo, verde y azul dependiendo del número de distintas escalas de gris (intensidad) que puede alcanzar la pantalla.

4.3.7 Iluminación de los LCD y Clasificación de Displays

Para que una **Pantalla LCD** muestre información, debe tener una fuente de luz. Algunos **Displays** usan sólo la luz ambiental y emplean una superficie reflejante montada atrás del **Display** (muchas calculadoras y relojes utilizan este método). Estos **Displays** no son muy brillantes porque la luz debe pasar dentro de múltiples **Polarizadores**, los cuales disminuyen severamente la intensidad de la luz, aunado a que varias capas del **Display** son construidas de materiales semitransparentes. Sin embargo, una fuente más intensa se emplea en la forma de un sistema de

luz de resplandor. Los bulbos de luz montados atrás de los bordes del **Display** reemplazan la luz ambiental reflejada. Esto resulta en **Displays** más brillantes por dos razones:

1) La luz no tiene que venir de adentro y por lo tanto no pierde parte de la intensidad.
2) Los sistemas luminosos pueden ser más intensos que la luz ambiental.

Las luces de respaldo tienen la desventaja de seguir siendo muy intensos potencialmente. Los sistemas de luces de respaldo son muy usados en **Displays** complejos tales como las pantallas de las computadoras laptop.

Las diferentes tecnologías existentes análogas a los monitores y a las pantallas **LCD**, como se mencionó anteriormente son:

1) **PDP (Plasma Display)**: Este es un tipo de **Tubo al Vacío** de dos electrodos, los cuales operan bajo el mismo principio que las lamparas fluorescentes caseras. Un gas inerte tal como el neón o el argón se introduce entre dos placas de vidrio cuyos electrodos transparentes han sido formados con anterioridad, el vidrio se ilumina generando descargas. Estos altos contrastes son relativamente fáciles de aplicar a las pantallas de color.
2) **LED (Light Emisor Display)**: Debido a la unión de dos tipos de semiconductores, los **LEDs** emiten luz cuando una corriente pasa entre ellos. Estos dispositivos son capaces de convertir un flujo de electrones en luz. Debido a su brillantez, son frecuentemente usados en señales al aire libre.
3) **Electroluminiscent Panel**: Estos dispositivos utilizan sulfuro de zinc u otro material que emita fluorescencia cuando se le aplica voltaje. Las pantallas utilizan una delgada capa de fósforo, el cual emite luz cuando se sujeta a un campo eléctrico. Esta pantalla ofrece las siguientes ventajas sobre los **LCDs** o los **PDPs**:
   a) Alta resolución y gran tamaño (escala de 16 niveles de gris, 1024 x 768 puntos).
   b) Alto contraste.
   c) Alta velocidad de respuesta.
   d) Angulo de vista ancho.
   e) Ligero y compacto.
   f) Bajo consumo de potencia.

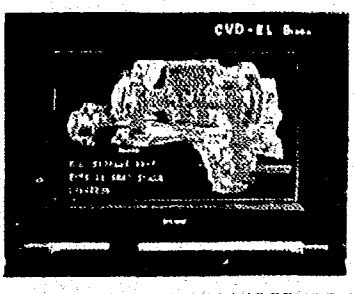


Fig. 4.24 PANEL ELECTROLUMINISCENTE (EL PANEL)

4) **VFD (Vaccum Fluorescent Display)**: Este es un tipo de **Tubo al Vacío** compuesto de electrodos enrejillados. Los electrodos positivos y el material fluorescente forman unidades de pantalla en una sustancia de vidrio, los termoelectrones son emitidos de filamentos de electrodos negativos ubicados en otro lugar. Estos termoelectrones son acelerados por rejillas localizadas entre ellos e impactadas en la pantalla, ocasionado la iluminación de ésta última.
5) **DMD (Digital Micromirror Device)**: Un **DMD** es una estructura en la cual un chip **SRAM** se cubre con espejos de aluminio de 16 micras cuadradas. Los extremos opuestos de cada espejo están soportadas por columnas, y rotan mientras mantienen un ángulo de 10 grados respecto del eje horizontal. Los espejos controlan el volumen reflejante de luz de su fuente, proyectando una imagen en la pantalla. Debido a que los **DMDs** operan bajo circuitos digitales, la imagen no se distorsiona por el ruido.

   Un proyector **DMD** consiste de un **DMD** de 768 x 576 pixeles, una lámpara Xe Arc, un filtro de color y unos lentes. El chip **DMD** se irradia con una luz de aproximadamente 2,000 lumens de intensidad e imágenes de 640 x 480 pixeles son proyectadas. 60 imágenes de 480 lineas se proyectan en una pantalla de 100 pulgadas cada 1/60 de segundo.

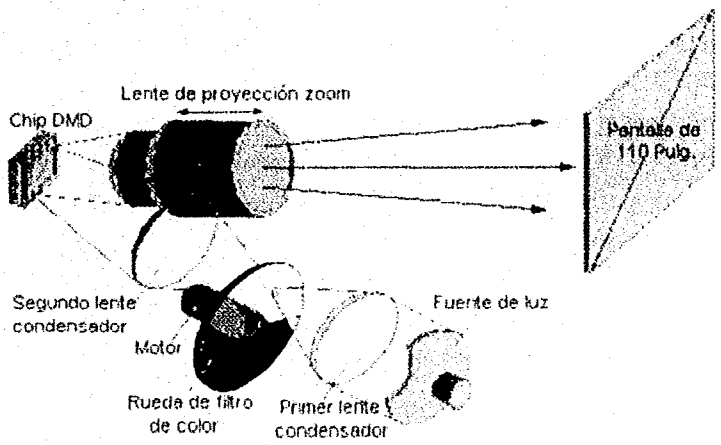


Fig. 4.25 ESTRUCTURA DE UN PROYECTOR DMD

6) **FED (Field Emision Display)**: Un **FED** proyecta imágenes usando el mismo principio que las pantallas de **CRT**. Un **FED** remueve electrones del cátodo y crea colisiones con el material fluorescente del ánodo, emitiendo luz. Mientras el cátodo de un **CRT** usa una fuente de electrones de un punto, un **FED** usa una fuente de electrones de superficie (figura 4.26).

4.3.8 Controladores de Pantallas LCD

Debido al rápido avance de la tecnologia, se puede encontrar en el mercado con relativa facilidad microcontroladores de **Pantallas LCD** en una sola oblea de silicio empacada en un chip, los cuales están microprogramados para realizar funciones especificas para desplegar caracteres alfanuméricos, en los nuevos dispositivos inclusive imágenes gráficas.

En la mayoría de los casos, los microcontroladores de **Pantallas LCD** están configurados para conectarse directamente al control de microprocesadores de 4 o de 8 bits. Además, las distintas marcas de fabricantes de microcontroladores comparten características comunes tales como la **ROM Generadora de Caracteres** (CGROM) y la **RAM Generadora de Caracteres** (CGRAM). La primera tiene la función de almacenar la serie de caracteres que puede mostrar la **Pantalla LCD**, estos caracteres no pueden ser modificadas directamente por el programador, en contraparte, la segunda tiene como función almacenar la serie de caracteres modificables que puede mostrar la **pantalla LCD**. Estos caracteres pueden ser programados a voluntad por el diseñador del sistema.

La mayoria de los microcontroladores son compatibles con los productos de Hitachi (HD 4478 y HD 66780). Ello equivale a diseñar una interfaz para un producto y cambiar de microcontrolador si así lo requiriera el caso.

La **CGROM** normalmente se enmascara con un patrón predeterminado, dado lo cual los diseñadores pueden personalizar su propio juego de caracteres. Entre las ventajas de una **CGROM** enmascarada y personalizada se encuentran:

1) Conjunto de caracteres personalizados.
2) Un conjunto máximo de 256 caracteres.

Si una **CGROM** personalizada se emplea, se deben de tomar en cuenta las siguientes consideraciones:

1) La instrucción **Clear Display** cuyo código es **20H** debe ser un carácter blanco.
2) Si son programados más de 240 caracteres en la **ROM**, entonces el número de caracteres disponibles en la **CGRAM** se reduce considerablemente. El espacio físico disponible en la **RAM** está disponible para ser usado como memoria, sin embargo no será suficiente para asociarle un código de carácter.
3) La **ROM de Caracteres** implementada en un chip particular se suele indicar por un subíndice de dos caracteres unidos al nombre del dispositivo, tal como SED $1278F_{DA}$.

La **CGRAM** normalmente cuenta con 64 bytes lo cual le permite al usuario programar hasta 8 caracteres.

4.3.9 Diseño del Controlador de Pantalla LCD

El diseño del controlador de **Pantalla LCD** se basa en el microcontrolador **SED 1278**, el cual tiene las siguientes características:

1) Controlador de **Pantalla LCD** de **Matriz de Puntos** el cual está diseñado para mostrar caracteres.

2) Cuenta con una capacidad para mostrar hasta 80 caracteres bajo el control de un microprocesador de 4 o de 8 bits.
3) Cuenta con una **ROM Generadora de Caracteres (CGROM)** interna, el cual tiene una capacidad de almacenamiento de 240 caracteres diferentes, cada una de las cuales se genera en una fuente de 5 X 10 puntos compatibles con un **Duty** de 1/11. Adicionalmente, el microcontrolador **SED 1278F/D** contiene 64 bytes de **RAM Generadora de Caracteres (CGRAM)**, en el cual el usuario puede almacenar 8 diferentes caracteres, cada uno consistiendo de 558 puntos. Esta característica de memoria ofrece alta flexibilidad al mostrar caracteres en la pantalla.
4) El voltaje mínimo garantizado para manejar el **LCD** es de 3 volts, otorgando al microcomtrolador **SED 1278F/D** la capacidad para controlar **Pantallas LCD** de bajos voltajes.

El diagrama a bloques del microcontrolador SED 1278F/D es:

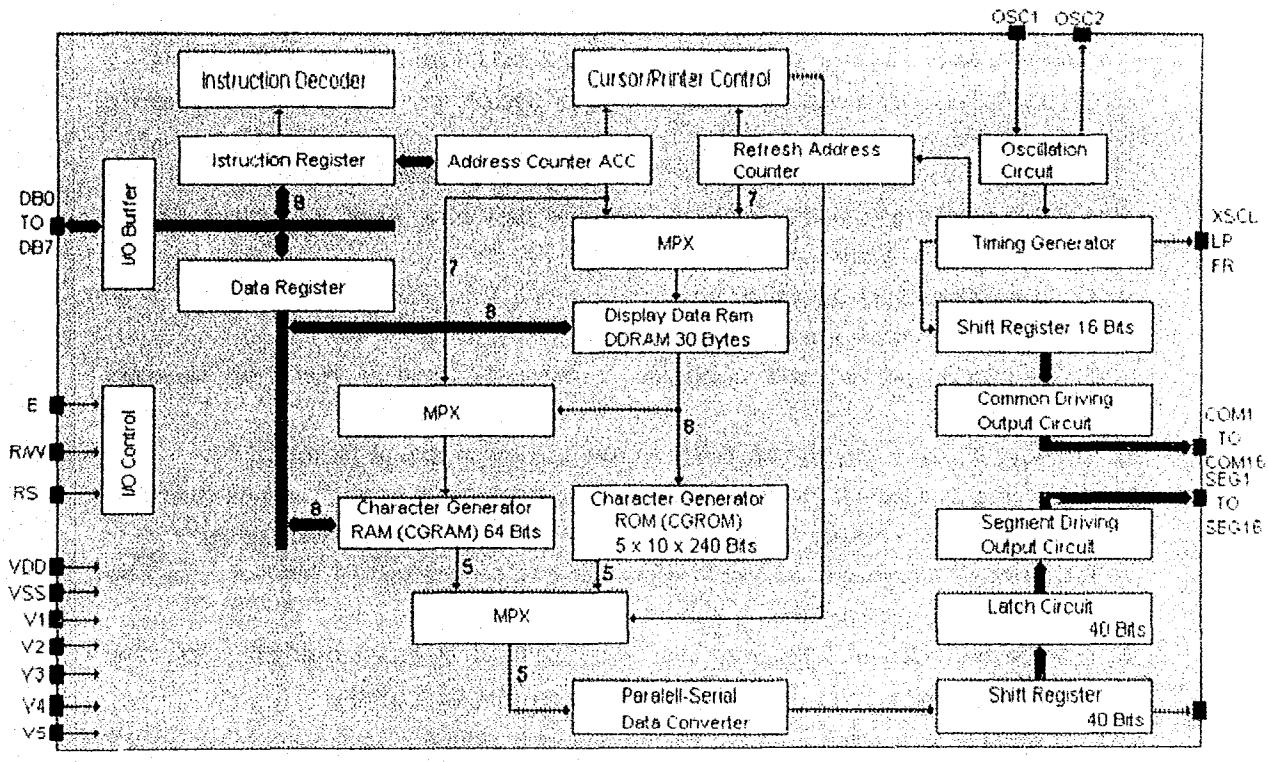


Fig. 4.26 DIAGRAMA A BLOQUES DEL MICROCONTROLADOR PARA LCD 1278F/D

La siguiente tabla muestra el número de pin y su descripción para el microcontrolador.

Pin		Pin		Pin		Pin	
Número	Nombre	Número	Nombre	Número	Nombre	Número	Nombre
1	SEG22	21	SEG2	41	DB2	61	COM15
2	SEG21	22	SEG1	42	DB3	62	COM16
3	SEG20	23	GND	43	DB4	63	SEG40
4	SEG19	24	OSC1	44	DB5	64	SEG39
5	SEG18	25	OSC2	45	DB6	65	SEG38
6	SEG17	26	V1	46	DB7	66	SEG37
7	SEG16	27	V2	47	COM1	67	SEG36
8	SEG15	28	V3	48	COM2	68	SEG35
9	SEG14	29	V4	49	COM3	69	SEG34
10	SEG13	30	V5	50	COM4	70	SEG33
11	SEG12	31	LP	51	COM5	71	SEG32
12	SEG11	32	XSCL	52	COM6	72	SEG31
13	SEG10	33	VDD	53	COM7	73	SEG30
14	SEG9	34	FR	54	COM8	74	SEG29
15	SEG8	35	D0	55	COM9	75	SEG28
16	SEG7	36	RS	56	COM10	76	SEG 27
17	SEG6	37	R/W	57	COM11	77	SEG26
18	SEG5	38	E	58	COM12	78	SEG25
19	SEG4	39	DB0	59	COM13	79	SEG24
20	SEG3	40	DB1	60	COM14	80	SEG23

Tabla 4.3 CONFIGURACIÓN DE PINES DEL CONTROLADOR SED 1278F/D

La siguiente tabla muestra la descripción de cada uno de los segmentos de pines del microcontrolador:

PINE	DESCRIPCIÓN
RS	Señal de entrada **Register Select** Selecciona entre el registro de dato y de instrucciones durante un acceso al **CPU** Con RS=0; ciclo de acceso al **Registro de Instrucciones** Con RS=1; ciclo de acceso al **Registro de Datos**
R/W	Esta señal de entrada selecciona en los registros del **1278F/D** entre los ciclos de lectura y escritura Con R/W=0; ciclo de **Escritura de Registro** Con R/W=1; ciclo de **Lectura de Registro**
E	Señal **Strobe de Entrada**, habilita el proceso de lectura/escritura
DB0-DB7	Línea de entrada/salida de datos de nivel TTL se conectan directamente al **Bus de Datos** del microprocesador
COM1-COM16	Controlador común (Common). Señales de salida hacia el **Panel de LCD**
SEG1-SEG40	Controlador de segmento. Señales de salida hacia el **Panel de LCD**
OSC1	Si se usa el oscilador **RC Interno** para generar las señales de control del **LCD**, la **Resistencia Feedback (Rf)** se conecta a este pin. Si se usa una fuente de reloj externa, el reloj se conecta a este pin
OSC2	Si se usa el oscilador **RC Interno** para generar las señales de control del **LCD**, la **Resistencia Feedback (Rf)** se conecta a este pin. Si se usa una fuente de reloj externa, este pin está abierto
LP	Pulso de salida latch de datos para un controlador X externo
XSCL	Pulsos de reloj de salida de cambio de datos para un controlador X externo
FR	Controlador de forma de onda AC del **LCD** para un controlador X externo.
D0	Señal de desplegado de datos para un controlaor X externo.

Tabla 4.4 DESCRIPCIÓN DE LOS PINES DEL MICROCONTROLADOR SED 178F/D

Para poder controlar las funciones y características de despliegue de caracteres en el panel, el microcontrolador cuenta con los siguientes comandos:

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Clear Display	0 0 0 0 0 0 0 1	01H	0

Acciones

1. Carga todas las direcciones de la **RAM de Despliegue de Datos (DDRAM)** con **20H**
2. Borra los contenidos del contador de direcciones a **0H**.
3. Activa la pantalla a un cambio de cero caracteres.
4. Activa el **Contador de Direcciones** para apuntar al **DDRAM**.
5. Si el cursor está activado, mueve el cursor al carácter más a la izquierda en la pantalla o, si una pantalla de dos líneas está siendo usada, mueve el cursor al carácter más a la izquierda en la primer línea.
6. Activa el **Contador de Direcciones** para incrementarse en cada uno de los accesos a la **DDRAM** o **CGRAM**.

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Cursor Home	0 0 0 0 0 0 1 *	02H 03H	0

Acciones

1. Borra el contenido del **Contador de Direcciones a 0H**.
2. Activa el **Contador de Direcciones** para apuntar al **DDRAM**.
3. Activa la pantalla a un cambio de cero caracteres.
4. Si el cursor está activado, lo mueve al carácter más a la izquierda. Si está siendo usada una pantalla de dos líneas, lo mueve al carácter más a la izquierda de la primera línea.

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Entry Mode Set	0 0 0 0 0 1 I/D S	04H-07H	0

Acciones

1. El **Bit I/D** selecciona la forma en la cual los contenidos del **Contador de Direcciones** son modificados después de cada acceso al **DDRAM** o **CGRAM**.
   - I/D=0: el **Contador de Direcciones** se incrementa.
   - I/D=1: el **Contador de Direcciones** se decrementa.
2. El **Bit S** habilita el desplazamiento de la pantalla en lugar del desplazamiento del cursor después de cada escritura o lectura al **DDRAM**.
   - S=1: Desplazamiento de la pantalla habilitada.
   - S=0: Desplazamiento del cursor habilitado.

La dirección en la cual se desplaza la pantalla es opuesta al desplazamiento del cursor. Por ejemplo, si S=0 e I/D=1 el cursor debe desplazarse un carácter a la derecha después de una escritura del microprocesador al **DDRAM**. Sin embargo, si S=1 e I/D=1 la pantalla debe desplazarse un carácter a la izquierda y el cursor mantiene su posición en el panel.

El cursor va a ser desplazado en la dirección seleccionada por **I/D** durante la lectura del **DDRAM**, sin considerar el valor de **S**. De forma similar, la lectura y escritura del **CGRAM** siempre desplaza el cursor. Nótese que si una pantalla de dos líneas es desplegada, ambas líneas van a ser desplazadas simultáneamente.

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Display ON/OFF	0 0 0 0 1 D C B	08H-0FH	0

Acciones

Esta instrucción controla varias características de la pantalla

1. El **Bit D** pone la pantalla completa en **Off** u **On**
   - D=1: La pantalla se pone a **On**
   - D=0: La pantalla se pone a **Off**
2. El **Bit C** pone el cursor en **Off** u **On**
   - C=1: El cursor se pone a **On**
   - C=0: El cursor se pone a **Off**
3. El **Bit B** habilita el parpadeo de los caracteres con los que se superpone el cursor
   - B=1: Parpadeo habilitado
   - B=0: Parpadeo deshabilitado

El parpadeo se logra alternando entre el desplegado normal y el ennegrecimiento total de un carácter. El periodo de parpadeo es activado a 204800 de la frecuencia de oscilación. Por ejemplo si la fosc=250 KHz, el cursor va a parpadear con un periodo de 0.8192 segundos, cerca de 1.2 Hz.

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Cursor/Display shift	0 0 0 1 S/C R/L * *	10H-1FH	0

Acciones

Esta instrucción desplaza la pantalla y/o mueve el cursor un carácter a la izquierda o a la derecha independientemente de una lectura/escritura de **DDRAM**

1. El **Bit S/C** selecciona el movimiento del cursor o el movimiento simultáneo del cursor y de la pantalla
   - S/C=1: Desplazamiento simultáneo de cursor y de pantalla
   - S/C=0: Desplazamiento únicamente del cursor
2. El **Bit R/L** selecciona entre el movimiento de retroceso o avance de la pantalla y/o del cursor
   - R/L=1: Desplazamiento hacia la derecha un carácter.
   - R/L=0: Desplazamiento hacia la izquierda un carácter.

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
System Set	0 0 1 IF N F * *	20H-3FH	0

Acciones

Esta instrucción inicializa el sistema, por lo cual debe ser la primera instrucción ejecutada después del suministro de potencia.

1. El **Bit IF** selecciona entre una interface para un microprocesador de 4 o de 8 bits
   - IF=1: Interface para un microprocesador de 8 bits usando DB0-DB7
   - IF=0: Interface para un microprocesador de 4 bits usando DB4-DB7

Los bits **N** y **F** seleccionan el número de líneas de pantalla y el correspondiente **Duty Cycle**, como se muestra a continuación:

N	F	Número de líneas	Relación "Duty"	Señales de salida Common	Señales de salida Common no seleccionadas
0	0	1	1/8	COM1-COM8	COM9-COM16
0	1	1	1/11	COM1-COM11	COM12-COM16
1	*	2	1/16	COM1-COM16	---

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Set CGRAM Address	0 1 ACGR	40H-7FH	0

Acciones

Esta instrucción

1. Carga una nueva dirección de 6 bits en el **Contador de Direcciones**
2. Activa el contador de direcciones para direccionar la CGRAM

Cada vez que esta instrucción es ejecutada, los contenidos del **Contador de Direcciones** van a ser automáticamente modificados después de cada acceso al **CGRAM**, como se ha determinado por la instrucción **Entry Mode Set**

Si se emite esta instrucción por el sistema microprocesador mientras la pantalla está habilitada y el cursor está habilitado o el parpadeo está habilitado, aparecerán pseudo-cursores o pseudo-parpadeos. Para prevenir esto, se necesita desactivar el cursor y el parpadeo antes de cargar una nueva dirección de CGRAM

La capacidad activa del **Contador de Direcciones** cuando éste direcciona la **CGRAM** es de 6 bits, así que el contador va a direccionar de **00H** hasta **3FH** si se escribe más de 64 bytes de datos en la **CGRAM**.

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Set DDRAM Address	1 Add	80H-CFH ..... 1 línea 80H-A7H línea 1 ... línea 2 C0H-E7h línea 2 ... línea 2	0

Acciones

1. Carga una nueva dirección de 7 bits dentro del **Contador de Direcciones**.
2. Activa el contador de direcciones para apuntar al **DDRAM**.

Cada vez que esta instrucción es ejecutada, los contenidos del **Contador de Direcciones** son automáticamente modificados después de cada acceso a la **CGRAM**, como haya sido configurado por la instrucción **Entry Mode Set**.

El **SED 1278** tiene únicamente 80 localidades. El espacio de dirección válida para varias configuraciones de pantalla se muestra a continuación:

Número de líneas		Caracteres	Dirección
1		80	00H-4Fh
2	1ra. Línea	40	00h-27h
	2da. Línea	40	40h-67h

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Write Data	Datos	Datos	1

Acciones

Esta instrucción escribe el dato presente en DB7-DB0 hacia la **CGRAM** o **DDRAM**. El espacio de **RAM** (**CG** o **DD**) y las direcciones en este espacio que son accesados depende de si una instrucción **Set CGRAM Address** o **Set DDRAM Address** ha sido recientemente ejecutada y en los parámetros de esta instrucción.

Los contenidos del **Contador de Direcciones** son automáticamente modificados después de cada instrucción **Write Data** como se ha determinado por la instrucción **Entry Mode Set**. Cuando el dato es escrito a la **CGRAM** los bits DB7, DB6 y DB5 no se despliegan directamente como caracteres.

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Ready Bussy Flag Address Counter	BF ACC	BF ACC	1

Acciones

Esta instrucción lee el **Registro de Instrucciones**, proporciona el valor actual del **Contador de Direcciones** y el estado del **Flag de Ocupado**. Esta instrucción debe ser ejecutada anterior a cualquier otra instrucción.

1. **Acc**, el valor del contador de direcciones apunta a una localidad del **CGRAM** o **DDRAM** dependiendo del tipo de instrucción **Set RAM Address** que haya sido recientemente ejecutada. En la instrucción **Busy Flag Check** inmediatamente después de ejecutar una instrucción **RAM Address Set**, un valor de **Contador de Direcciones** válido puede ser leído 5 ciclos de reloj después de que el registro **BF** (**Busy Flag**) se va a un nivel bajo. En la instrucción **Busy Flag Check** inmediatamente después de la ejecución de una instrucción **Write Data**, un valor de **Contador de Direcciones** válida puede ser leída en cuanto el registro **BF** se vaya a un nivel bajo.
2. El **Bit FB** muestra el estado de1a bandera **Busy**
   - BF=1. El microcontrolador **SED 1278** se encuentra ocupado.
   - BF=0. El microcontrolador **SED 1278** se encuentra listo para la siguiente instrucción.

Instrucción	Código binario B7 B6 B5 B4 B3 B2 B1 B0	Código hexadecimal	Estado RS
Read Data	Dato	Dato	1

Acciones

Esta instrucción lee los datos de la **CGRAM** o **DDRAM**, dependiendo del tipo de instrucción **Set RAM Address** que haya sido ejecutada recientemente. La dirección en este espacio depende de los parámetros de la instrucción **Set RAM Address**. Inmediatamente antes de ejecutar una instrucción **Read Data**, debe ejecutar una instrucción **Set CGRAM Address** o una instrucción **Set DDRAM Address**.

Los contenidos del **Contador de Direcciones** son modificados después de cada instrucción **Read Data**, como ha sido determinado por la instrucción **Entry Mode Set**. El desplazamiento de la pantalla no es ejecutado independientemente de la instrucción **Entry Mode Set**.

La siguiente tabla muestra un resumen de los comandos que existen para el microcontrolador

Instrucción	Código RS R/W B7 B6 B5 B4 B3 B2 B1 B0	Descripción	Ciclos (máximo)
Clear Display	0 0 0 0 0 0 0 0 0 1	Borra todos los datos en la pantalla y activa el **Contador de Direcciones** de la **DDRAM** a **00H**.	410
Return Home	0 0 0 0 0 0 0 0 1 *	Activa el **Contador de Direcciones** de la **DDRAM** a **00H**. También retorna cualquier dato desplazado a **Home**. Los contenidos de la **DDRAM** permanecen sin cambio.	410
Entry Mode Set	0 0 0 0 0 0 0 1 I/D S	Especifica la dirección en la cual el cursor se mueve y si la pantalla puede desplazarse o no cuando los datos son leídos o escritos en la pantalla.	10
Display On/Off	0 0 0 0 0 0 1 D C B	Pone la pantalla total en **On/Off**, así como el estado del cursor en **On/Off**. Adicionalmente el parpadeo de los caracteres sobre los que se superpone el cursor.	10
Cursor/Display Shift	0 0 0 0 0 1 S/C R/L * *	Mueve el cursor y desplaza la pantalla sin cambiar el contenido los contenidos de la **DDRAM**.	10
System Set	0 0 0 0 1 IF N F * *	Activa la longitud de la **Interface de Datos** (IF), el número de caracteres a desplegar (N) y la fuente del carácter (F).	10
Set CGRAM Address	0 0 0 1 ACG	Activa las direcciones del **CGRAM**, seguido por una transferencia de datos del **CGRAM**.	10
Set DDRAM Address	0 0 1 ADD	Activa las direcciones del **DDRAM**, seguido por una transferencia de datos del **DDRAM**.	10
Read Bus Flag and Address	0 1 BF ACC	Lee el registro **Busy Flag** (BF) el cual indica la operación interna y los contenidos del **Contador de Direcciones**.	0
Write Data to CG o DD RAM	1 0 WRITE DATA	Escribe datos al **DDRAM** o **CGRAM**.	10
Read Data from CG o DD RAM	1 1 READ DATA	Lee datos del **DDRAM** o **CGRAM**.	10

Tabla 4.5 COMANDOS DEL MICROCONTROLADOR SED 1278F/D

El principio de operación del microcontrolador **SED1278F/D** se basa en las siguientes características:

1) **Busy Flag**: El microcontrolador toma entre 10 y 410 ciclos de reloj para ejecutar las instrucciones. Durante este periodo, instrucciones adicionales pueden no efectuarse. El dispositivo contiene una bandera de estado **Ocupado** para permitirle al usuario verificar el estado interno del chip. La bandera **BF** debe de estar en un estado 0 antes de que otra instrucción sea ejecutada.

   Si la bandera **Ocupado** no se verifica entre las instrucciones el usuario debe de efectuar retardos mayores a los necesarios para la instrucción actual antes de ejecutar otra instrucción.

La siguiente tabla muestra un resumen de los comandos que existen para el microcontrolador

Instrucción	Código RS R/W B7 B6 B5 B4 B3 B2 B1 B0	Descripción	Ciclos (máximo)
Clear Display	0 0 0 0 0 0 0 0 0 1	Borra todos los datos en la pantalla y activa el **Contador de Direcciones** de la **DDRAM a 00H**.	410
Return Home	0 0 0 0 0 0 0 0 1 *	Activa el **Contador de Direcciones** de la **DDRAM a 00H**. También retorna cualquier dato desplazado a **Home**. Los contenidos de la **DDRAM** permanecen sin cambio.	410
Entry Mode Set	0 0 0 0 0 0 0 1 I/D S	Especifica la dirección en la cual el cursor se mueve y si la pantalla puede desplazarse o no cuando los datos son leídos o escritos en la pantalla.	10
Display On/Off	0 0 0 0 0 0 1 D C B	Pone la pantalla total en **On/Off**, así como el estado del cursor en **On/Off**. Adicionalmente el parpadeo de los caracteres sobre los que se superpone el cursor.	10
Cursor/Display Shift	0 0 0 0 0 1 S/C R/L * *	Mueve el cursor y desplaza la pantalla sin cambiar el contenido los contenidos de la **DDRAM**.	10
System Set	0 0 0 0 1 IF N F * *	Activa la longitud de la **Interface de Datos (IF)**, el número de caracteres a desplegar **(N)** y la fuente del carácter **(F)**.	10
Set CGRAM Address	0 0 0 1 ACG	Activa las direcciones del **CGRAM**, seguido por una transferencia de datos del **CGRAM**.	10
Set DDRAM Address	0 0 1 ADD	Activa las direcciones del **DDRAM**, seguido por una transferencia de datos del **DDRAM**.	10
Read Bus Flag and Address	0 1 BF ACC	Lee el registro **Busy Flag (BF)** el cual indica la operación interna y los contenidos del **Contador de Direcciones**.	0
Write Data to CG o DD RAM	1 0 WRITE DATA	Escribe datos al **DDRAM** o **CGRAM**.	10
Read Data from CG o DD RAM	1 1 READ DATA	Lee datos del **DDRAM** o **CGRAM**.	10

Tabla 4.5 COMANDOS DEL MICROCONTROLADOR SED 1278F/D

El principio de operación del microcontrolador **SED1278F/D** se basa en las siguientes características:

1) **Busy Flag**: El microcontrolador toma entre 10 y 410 ciclos de reloj para ejecutar las instrucciones. Durante este periodo, instrucciones adicionales pueden no efectuarse. El dispositivo contiene una bandera de estado **Ocupado** para permitirle al usuario verificar el estado interno del chip. La bandera **BF** debe de estar en un estado 0 antes de que otra instrucción sea ejecutada.

   Si la bandera **Ocupado** no se verifica entre las instrucciones el usuario debe de efectuar retardos mayores a los necesarios para la instrucción actual antes de ejecutar otra instrucción.

2) **Interface para un Microprocesador de 4 Bits**: Si se ejecuta una instrucción **System Set** con el **Bit 4** puesto a 0, el microcontrolador **SED 1278F/D** va a operar con una interface de **Bus de Datos** para un microprocesador de 4 bits.

   Si se usa una interface de 4 bits, las instrucciones de 8 bits son escritas nibble a nibble; con el nibble de mayor orden primero, seguido por el nibble de menor orden. No es necesario verificar el estado de del registro **BF** entre la escritura de nibbles separados de las instrucciones individuales.

   La lectura del registro **Busy Flag/Address Counter** proporciona el nibble de mayor orden primero, seguido por el nibble de menor orden.

3) **Iniciación del sistema**: Aún cuando el microcontrolador **SED1278** no tiene una entrada de reset externa, éste va a resetearse automáticamente cuando la potencia sea suministrada. La secuencia inicia una vez que V_{DD}<4.5 V.

   Mientras que el microcontrolador SED1278 se esté reseteando, el registro **BF** se pone a un nivel lógico 1. El reset toma cerca de 3,750 **Ciclos de Reloj**. Por ejemplo si fosc=250 KHz, la secuencia de reset tarda alrededor de 30 ms.

   La acción de reset coloca el microcontrolador **SED1278** en un estado donde:

   a) La pantalla es borrada
   b) La configuración del sistema corresponde a:
      1. IF=1: Interface para un microprocesador de 8 bits.
      2. N=0: Una línea de desplegado de datos.
      3. N=0: **Duty Cycle** de 1/8.
   c) La configuración de la pantalla corresponde a:
      1. D=0: Pantalla en **Off**.
      2. C=0: Cursor en **Off**.
      3. B=0: Parpadeo en **Off**.
   d) El modo de entrada está puesto a:
      1. I/D=1: Incremento.
      2. S=1: Sin desplazamiento de pantalla.

Durante la iniciación, el reseteo involucra varios factores inestables relacionados a fluctuaciones de salida de la fuente de potencia. Por esta razón es altamente recomendable que una secuencia de iniciación de software se lleve a cabo. La siguiente figura muestra la secuencia de iniciación de software para una conexión con un microprocesador de 8 bits en su **Bus de Datos**:

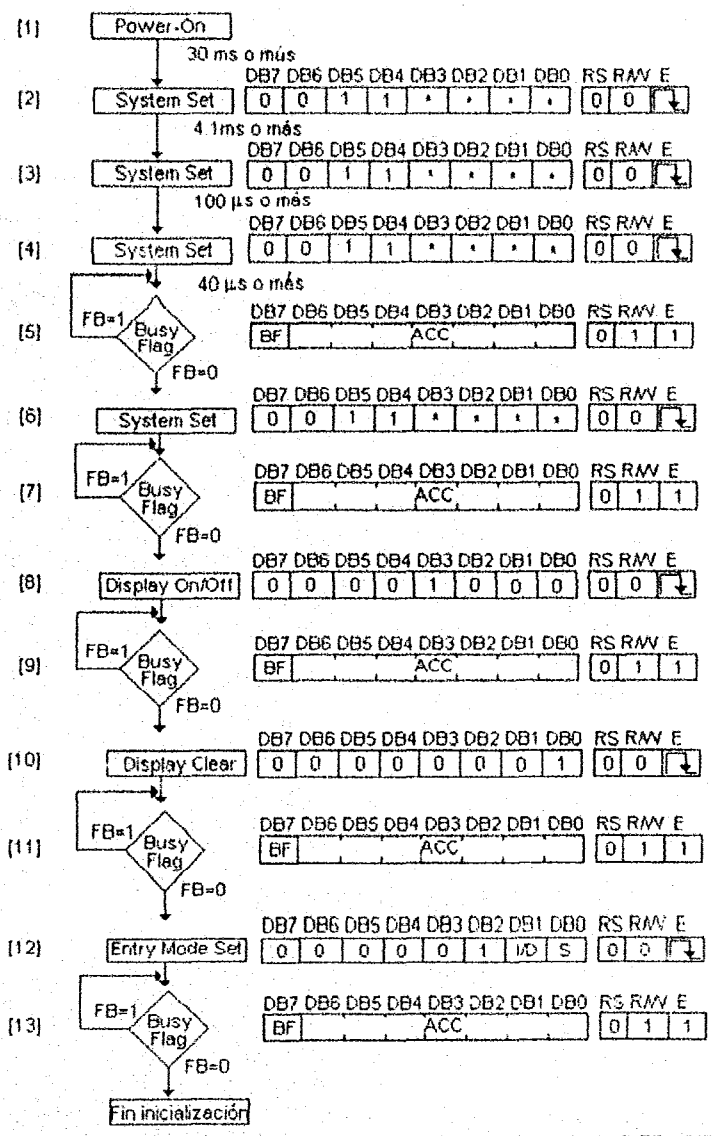


Fig. 4.27 SECUENCIA DE INICIACION PARA UN MICROPROCESADOR DE 8 BITS

Si desea conectar el microcontrolador a un microprocesador de 4 bits en su **Bus de Datos**, la secuencia de iniciación varía muy ligeramente de la anterior, como puede verse en la siguiente figura:

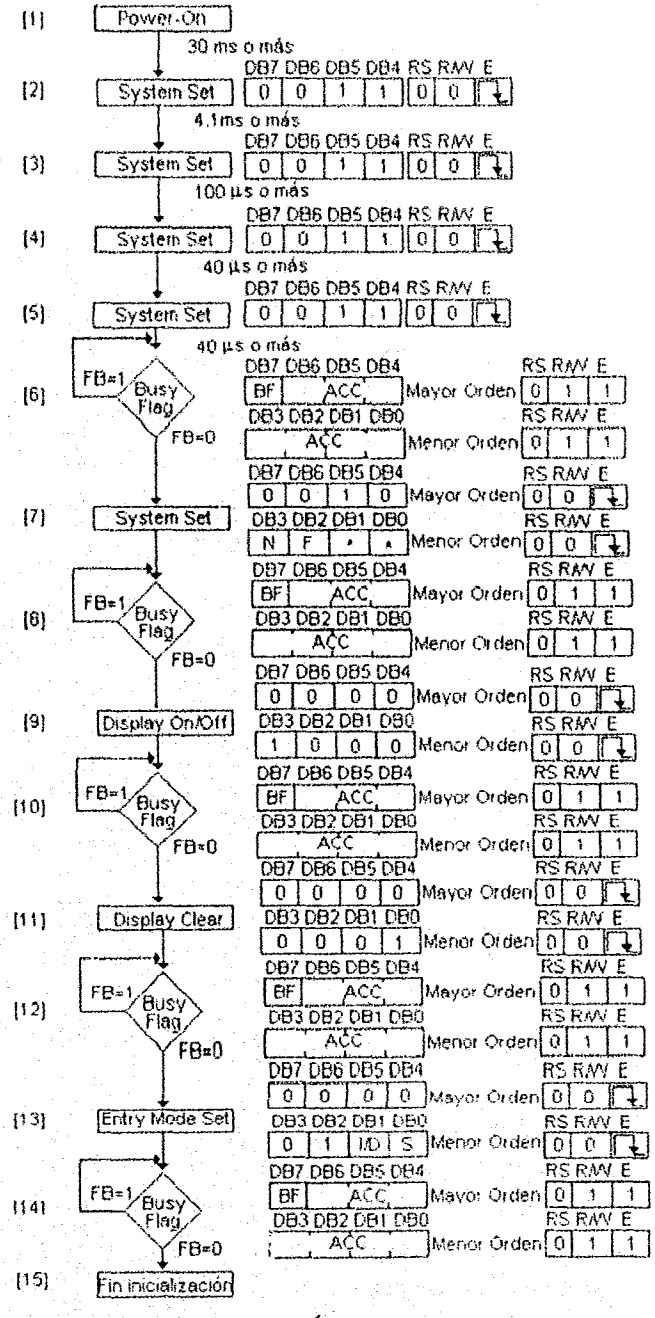


Fig. 4.28 SECUENCIA DE INICIACIÓN PARA UN MICROPROC. DE 4 BITS

4) **Interface del LCD**: Las líneas segment y common generan señales de control usando los voltajes suministrados en los pines V_1, V_2, V_3, V_4 y V_5. Los niveles de voltaje en estos pines dependen del **Duty Cycle** de la pantalla. La forma más simple para producir estos voltajes es usando un circuito divisor de voltaje. Las siguientes figuras muestran ejemplos de configuraciones **Duty Cycle** para 1/8, 1/11 y 1/16:

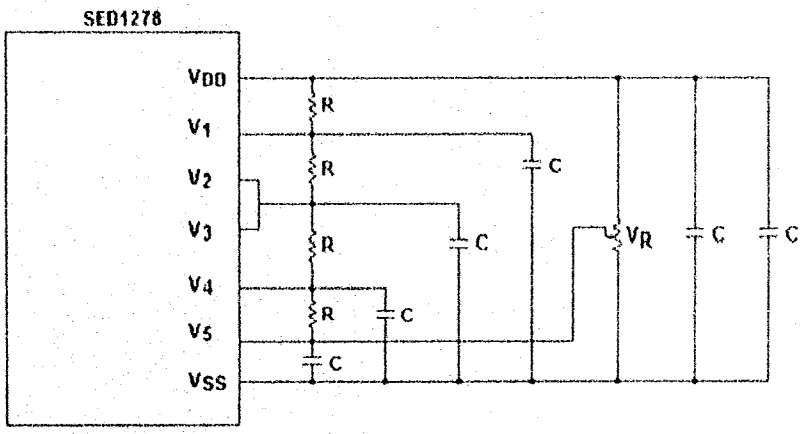


Fig. 4.29 RED DE CONTROL DE VOLTAJE PARA UN LCD DE 1/8 O 1/11 "DUTY CYCLE"

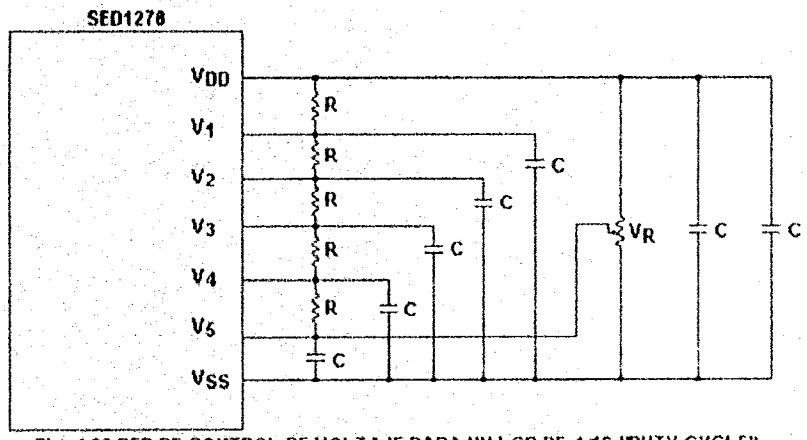


Fig. 4.30 RED DE CONTROL DE VOLTAJE PARA UN LCD DE 1/16 "DUTY CYCLE"

Cuando se implementen las configuraciones anteriores, debe de tomarse en consideración lo siguiente:

a) V_5 se activa usando un potenciómetro y (V_{DD}-V_{SS})

b) La fuente de potencia para el microcontrolador **SED1278** debe ser filtrada con un capacitor **CO** de al menos 0.1 μF colocado tan cerca del chip tanto como sea posible.

5) **Configuración de la Interfaz del LCD**: El microcontrolador **SED1278** tiene 16 salidas Common y 40 Segment de control, permitiendo al chip controlar hasta 16 caracteres. La capacidad de control puede ser expandida hasta 80 caracteres, usando un controlador de segmento externo **SED1181F**$_{LA}$.

Las siguientes figuras muestran la configuración de las **Líneas de Control** para diversas características del **Panel de LCD**:

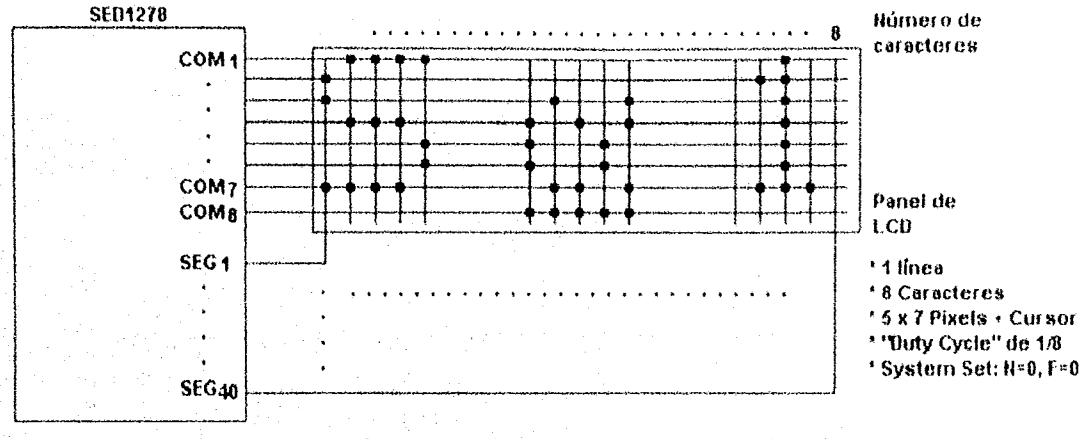


Fig. 4.31 CONFIGURACIÓN DE LAS SEÑALES DE CONTROL PARA 1 LÍNEA, 8 CARÁCTERES

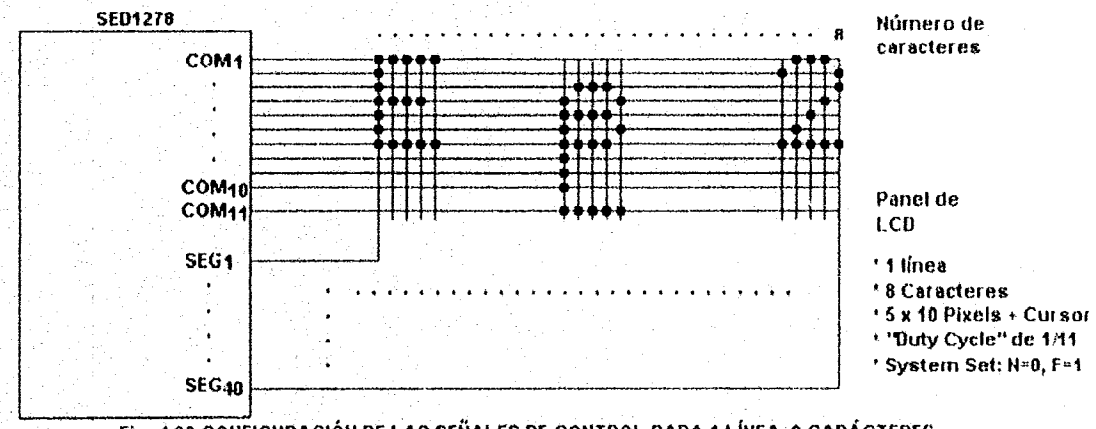


Fig. 4.32 CONFIGURACIÓN DE LAS SEÑALES DE CONTROL PARA 1 LÍNEA, 8 CARÁCTERES

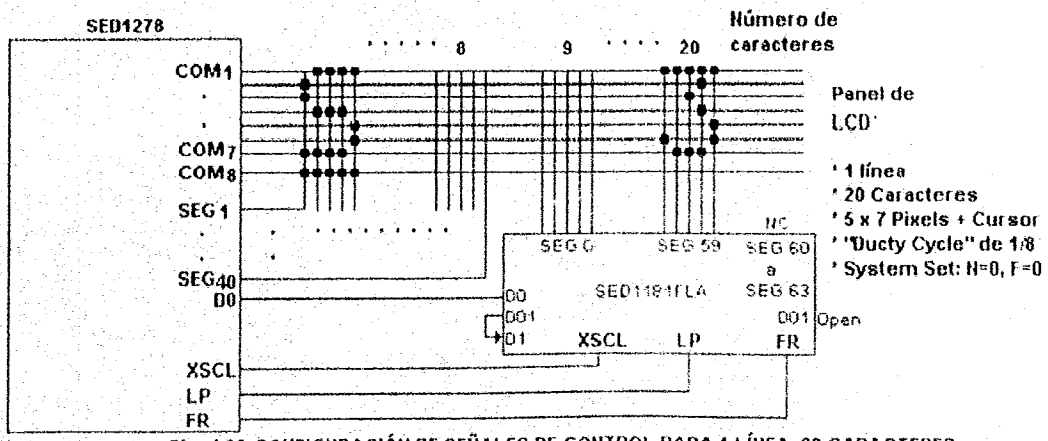


Fig. 4.33 CONFIGURACIÓN DE SEÑALES DE CONTROL PARA 1 LÍNEA, 20 CARACTERES

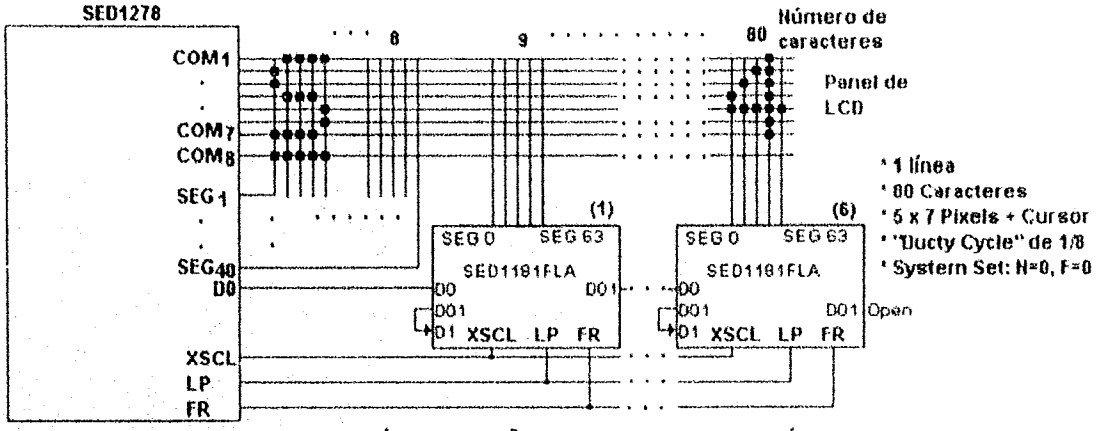


Fig. 4.34 CONFIGURACIÓN DE LAS SEÑALES DE CONTROL PARA 1 LÍNEA, 80 CARACTERES

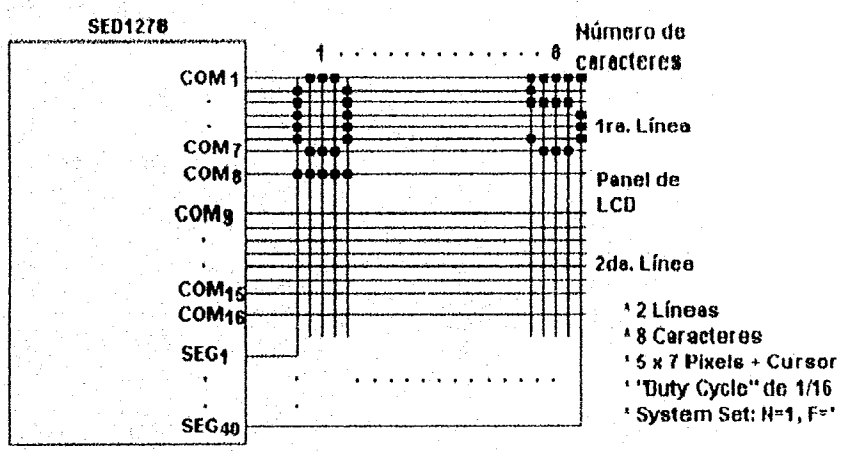


Fig. 4.35 CONFIGURACIÓN DE LAS SEÑALES DE CONTROL PARA 2 LÍNEAS, 8 CARACTERES

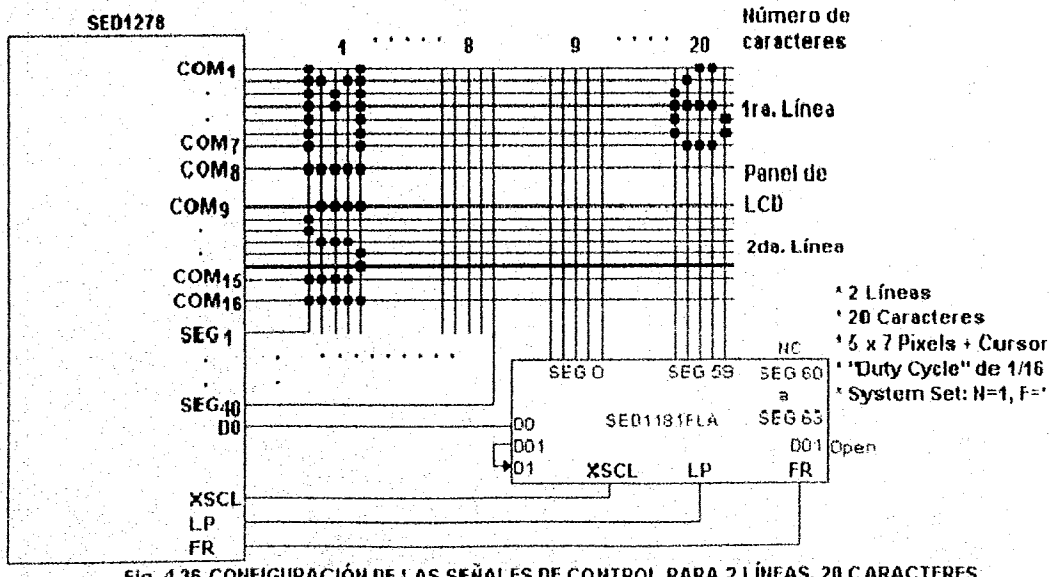


Fig. 4.36 CONFIGURACIÓN DE LAS SEÑALES DE CONTROL PARA 2 LÍNEAS, 20 CARACTERES

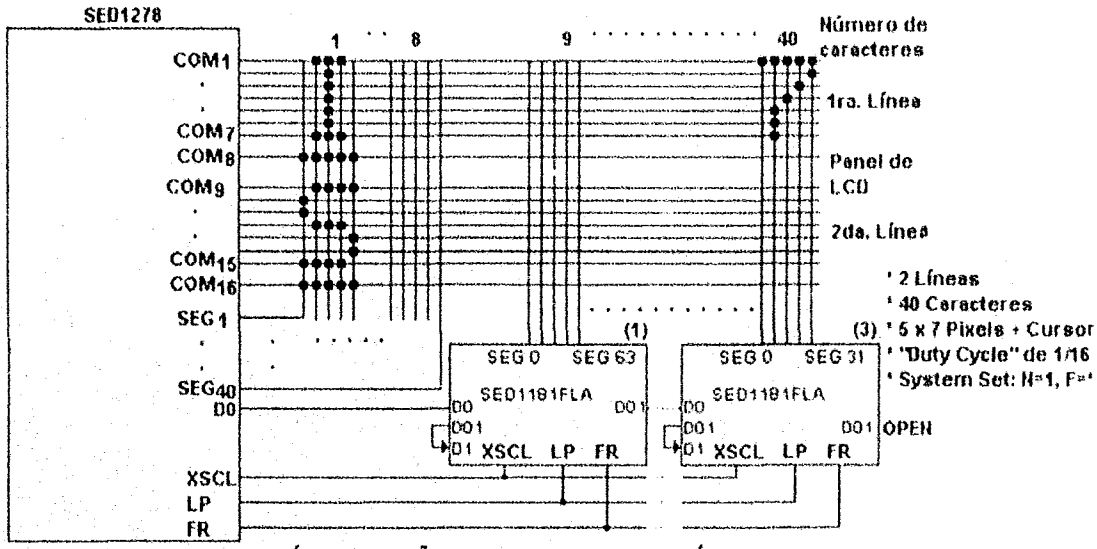


Fig. 4.37 CONFIGURACIÓN DE LAS SEÑALES DE CONTROL PARA 2 LÍNEAS, 40 CARACTERES

6) **Interface Hacia un Microprocesador**: Como se había mencionado anteriormente, el microcontrolador **SED1278** puede ser gobernado por un microprocesador con un **Bus de Datos** 4 bits o por un microprocesador con un **Bus de Datos de** 8 bits. La siguiente figura muestra un ejemplo de conexión a un microprocesador que maneja un **Bus de Datos** de 8 bits:

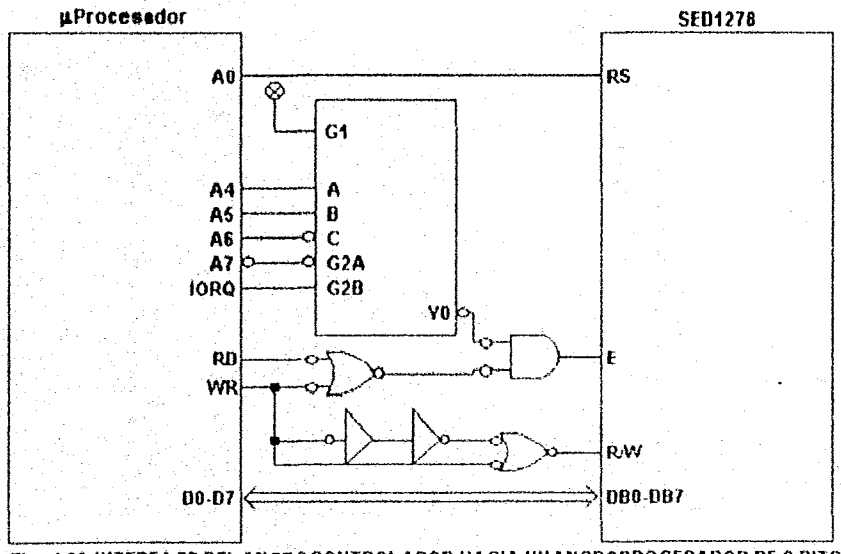


Fig. 4.38 INTERFACE DEL MICROCONTROLADOR HACIA UN MICROPROCESADOR DE 8 BITS

7) **Especificaciones**: Para poder manejar el microcontrolador sin incurrir en riesgos de dañarlo, se debe diseñar el sistema de tal forma que se ajuste a las siguientes características:

a) **Valores Absoluto Máximos**: La siguiente tabla muestra la relación de valores máximo que puede soportar el microcontrolador **SED1278**:

Parámetro	Símbolo	Valor	Unidad
Fuente de Voltaje (1)	V_{DD}	-3.0 a 7.0	V
Fuente de Voltaje (2)	V_1-V_5	-0.3 a VDD+0.3	V
Voltaje de entrada	V_{IN}	-0.3 a VDD+0.3	V
Temperatura de operación	T_{OPR}	-20 a +75	°C
Temperatura de almacenamiento	T_{STG}	-65 a +150	°C
Temperatura de soldado x Tiempo (3)	T_{SOL}	260, 10	°C, S
Disipación de potencia	P_D	300	mW

Tabla 4.6 VALORES MÁXIMOS DEL MICROCONTROLADOR SED1278F/D

Notas:

1. $V_{DD} >> V_1 >> V_2 >> V_3 >> V_4 >> V_5 >> V_{SS}$
2. Un producto con empaquetado **Flat** puede ser menos resistente a la humedad si se le expone a temperaturas extremas. Cuando se monta este dispositivo en una tarjeta de circuito impreso, use una técnica de soldadura de tal forma que evite excesiva carga termal en la resina del paquete.
3. Todos los voltajes asumen que V_{DD}=0.

b) **Caracteristicas de CD**: De igual forma, la alimentación de energía del sistema debe de ajustarse a los siguientes requerimientos:

Parámetro	Símbolo	Condición	Valor			Unidad	Pines aplicables
			Min.	Typ.	Max.		
V_{IN} H (1) (TTL)	V_{IH1}		2.0	-	V_{DD}	V	DB0-DB7, E
V_{IN} L (1) (TTL)	V_{IL1}		V_{SS}	-	0.8	V	R, S, R/W
V_{IN} H (2) (CMOS)	V_{IH2}		V_{DD}-1.0	-	V_{DD}	V	OSC1
V_{IN} L (2) (CMOS)	V_{IL2}		V_{SS}	-	1.0	V	
V_{OUT} H (1) (TTL)	V_{OH1}	$-I_{OH}$=0.250 mA	2.4	-	-	V	DB0-DB7
V_{OUT} L (1) (TTL)	V_{OL1}	I_{LO}=1.6 mA	-	-	0.4	V	
V_{OUT} H (2) (CMOS)	V_{OH2}	$-I_{OH}$=0.04 mA	0.9V_{DD}	-	-	V	XSCL, LP, DO
V_{OUT} L (2) (CMOS)	V_{HL2}	I_{OH}=0.04 mA	-	-	0.1V_{DD}	V	
Resistor Controlador (COM)		$\|V_{COM}$-$V_H\|$=0.5 V	-	2	10	KΩ	COM1-COM16
Resistor Controlador (SEG)			-	2.5	10	KΩ	SEG1-SEG40
Corriente de I/O		$\|V_{SEG}$-$V_N\|$=0.5 V	-	-	1	µA	
Corriente Pull-Up		V_{IN}=0 a V_{DD}	50	125	250	µA	DB0-DB7, R, S, R/W
Fuente de Corriente		F_{OSC}=f_{OP}=270 KHz.	-	0.5	0.8	µA	V_{DD}

Tabla 4.7 VALORES V_{CD} PARA EL MICROCONTROLADOR SED1278

c) **Caracteristicas de CA**: La siguiente figura muestra el diagrama de tiempos para un ciclo de lectura/escritura del microcontrolador:

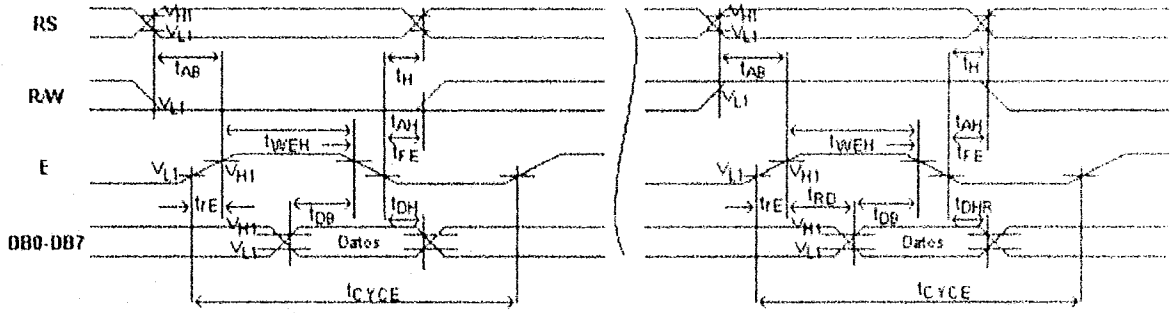


Fig. 4.39 DIAGRAMA DE TIEMPO PARA UN PROCESO DE ESCRITURA/LECTURA DESDE EL MICROPROCESADOR

La siguiente tabla muestra la definición de los elementos que componen el diagrama anterior:

Parámetro/Escritura	Símbolo	Condición	Valores		Unidad
			Min	Max	
Enable Cycle Time	T_{CYCE}		500	-	ns
Enable 'H' level pulsewidth	t_{WEH}		220	-	ns
Enable rise/fall time	T_{rE}, t_{fE}		-	25	ns
RS, R/W setup time	T_{AS}		40	-	ns
RS, R/W address hold time	T_{AH}		10	-	ns
Data setup time	T_{DS}		60	-	ns
Write data hold time	T_{DH}		10	-	ns
Parámetro/Lectura	Símbolo	Condición	Valores		Unidad
			Min	Max	
Enable Cycle Time	T_{CYCE}		500	-	ns
Enable 'H' level pulsewidth	t_{WEH}		220	-	ns
Enable rise/fall time	T_{rE}, t_{fE}		-	25	ns
RS, R/W setup time	t_{AS}		40	-	ns
RS, R/W address hold time	t_{AH}		10	-	ns
Read data setup time	t_{RD}	C_L=100pF	0	120	ns
Read data hold time	t_{DHR}		20	-	ns

Tabla 4.8 DESCRIPCIÓN DE LOS ELEMENTOS DE CICLO DE ESCRITURA/LECTURA

Como se puede ver de lo descrito anteriormente, diseñar un sistema que permita controlar una **Pantalla LCD** es una tarea tan ardua como diseñar un sistema mínimo completo. Ante todo por los enormes conocimientos técnicos que se requieren para lograr un buen diseño efectivo y funcional. Pero sobre todo que se pueda ajustar al presupuesto de cualquier persona que desee ahondar en este maravilloso mundo de los microprocesadores. Para reducir el costo y simplificar el diseño del sistema mínimo, existen en el mercado **Pantallas LCD Recuperadas**, las cuales vienen a auxiliar en una forma sin igual al diseñador, ya que permiten a un costo relativamente bajo adquirir en una sola tarjeta impresa el microcontrolador y la **Pantalla LCD**.

Ante esta perspectiva, el diseño del sistema mínimo se basará en la utilización del sistema **SMC1622**, el cual contiene en su diseño la utilización del microcontrolador **SED1278** (descrito con anterioridad) y del circuito integrado **KS0061**, el cual funge como adición al microcontrolador con el

objeto de incrementar su capacidad y poder controlar sus funciones para mostrar un total de 80 caracteres distribuidos en dos líneas de caracteres en una sola pantalla.

La siguiente figura muestra una imagen del sistema **SMC1622**.

Fig. 4.40 PANTALLA DE LCD SMC1622

A grandes rasgos las características de la tarjeta **SMC1662** son:

1. Bajo voltaje de potencia: 5 V.
2. Controlador en la tarjeta: **SED1278**.
3. Interfaz directa hacia un microprocesador de 4 u 8 bits.
4. 11 comandos de control.
5. 80 mm de largo X 36 mm de ancho X 12 mm de grosor.
6. Tamaño de los caracteres: 2.95 mm de largo X 5.55 mm de ancho.
7. Area de vista: 64.5 mm de largo X 13.8 mm de ancho.
8. Apertura Bezel: 64.5 mm de largo X 13.8 mm de ancho.
9. Tamaño del punto: 0.58 mm de largo X 0.78 mm de ancho.
10. Diapasón del punto: 0.63 mm de largo X 0.83 mm de ancho.

Las siguientes tablas muestran las características más relevantes respecto a óptica, ubicación y descripción de los pines de la tarjeta **SMC1622**:

Elemento	Símbolo	Min.	Tip	Max	Unidad
Ángulo de vista	Φ	-10	25	40	grados
Contraste	K	-	3.0	-	-
Encendido	TON	-	200	400	ms
Apagado	TOFF	-	250	400	ms

Tabla 4.9 CARACTERÍSTICAS OPTICAS DEL SMC1622

Número de pin	Señal	Función
1	GND	Tierra
2	V_{CC}	Fuente de potencia de 5 V
3	V_0	Voltaje de control del LCD
4	RS	"H": entrada de datos "L": entrada de comandos
5	R/W	Read/Write
6	E	Enable
7	DB0	Bus de Datos DB0-DB7: operaciones de 8 bits DB4-DB8: operaciones de 4 bits
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	

Tabla 4.10 CONFIGURACIÓN DE LOS PINES DEL SMC1622

La siguiente figura muestra la forma de conexión de los pines **VDD**, **V0** y **GND** para controlar el contraste de los caracteres:

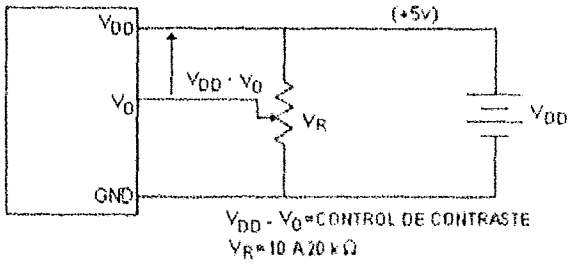


Fig. 4.41 DIAGRAMA DE CONEXIÓN PARA CONTROL DE CONTRASTE

Gracias a que el trabajo de diseño y los costos derivados se verán considerablemente disminuidos por la utilización de una **Pantalla LCD** con microcontrolador integrado, el hardware requerido para dicho circuito consta sólo de los elementos necesarios para que funja como interfaz o medio de realizar una comunicación entre este elemento y el microprocesador. Además de lo anteriormente dicho, el pequeño sistema operativo del sistema mínimo tiene programado una fase de retardo con el tiempo suficiente de tal forma que pueda sustituir en todo la ardua tarea de realizar procesos cíclicos verificando el estado del microcontrolador y procediendo a no permitir ninguna comunicación con éste, hasta la finalización de su proceso actual (verificación del **Bit BF**).

La siguiente figura muestra la interfaz final **Microprocesador-Pantalla LCD**. La parte esencial del sistema radica en los circuitos integrados tipo latch octal (74LS373), los cuales retienen los datos y los comandos dirigidos hacia la **Pantalla LCD**. Posteriormente, se habilita la señal **ENALCD** permitiendo a la **Pantalla LCD** realizar las instrucciones almacenadas en los circuitos tipo latchs anteriores.

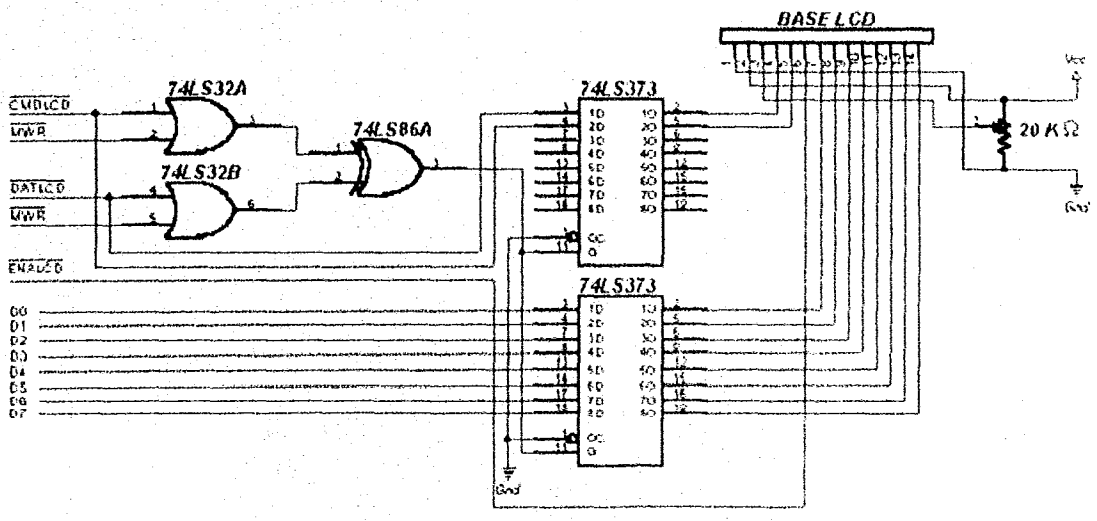


Fig. 4.42 DIAGRAMA DE LA INTERFAZ UNIDIRECCIONAL MICROPROCESADOR-PANTALLA LCD

4.4 Resumen

La transferencia de datos entre el microprocesador y el teclado se realiza en forma serial. Esta transferencia de datos se inicia mediante la generación de una señal START, seguido de 8 bits de datos, con el bit LSB en primer instancia y el bit MSB al final, cuando se han transferido los 8 bits de datos, se genera un bit de PARIDAD y finalmente un bit de parada (STOP).

Al usar el teclado en forma directa, es indispensable conocer el código que se genera al momento de presionar y liberar una tecla, ya que esto permite diseñar la interfaz para que se decodifique el código de la tecla al correspondiente código ASCII.

Sin lugar a dudas, los dispositivos periféricos de salida de datos más barato y fáciles de manejar son las Pantallas de Cristal Líquido, gracias a sus bajos requerimientos de hardware para controlarlos, de igual forma, requieren de una fuente de alimentación muy baja (normalmente, 5 volts en contraposición a los altos voltajes que se tienen que generar dentro de los monitores convencionales CRT).

En este aspecto, es mejor la utilización de Pantallas de Cristal Líquido "recuperados", los cuales provienen de sistemas digitales dañados, pero a los cuales aún le quedan piezas en buen estado. El uso de este tipo de pantallas reduce en forma considerable el costo total del sistema, puesto que se puede adquirir una pantalla por menos de la mitad de su precio comercial.

CAPITULO V:
DISEÑO DEL SISTEMA MÍNIMO

"El invento más grande del hombre es sin duda la computadora"

Anónimo

Después de haber diseñado el sistema de control e interfaz para los dispositivos periféricos del sistema mínimo, el siguiente paso que procede es el diseño del sistema mínimo en sí. Cada etapa del proceso de construcción se describirá en forma extensa, sin embargo se recomienda que cada una de las etapas sea probada antes de pasar a la etapa posterior, solucionando los problemas que se presenten en su momento y no retroceder cuando ya se lleve una etapa muy avanzada.

El proceso de diseño del sistema mínimo se divide en cuatro etapas principales a saber:

1) Configuración de los buses del microprocesador Z80.
2) Configuración de las señales de control del microprocesador Z80.
3) Configuración de la memoria y decodificación para la selección de los **Puertos de Entrada/Salida**.
4) Reconocimiento de interrupción.

La siguiente figura muestra un diagrama a bloques del diseño del sistema mínimo final:

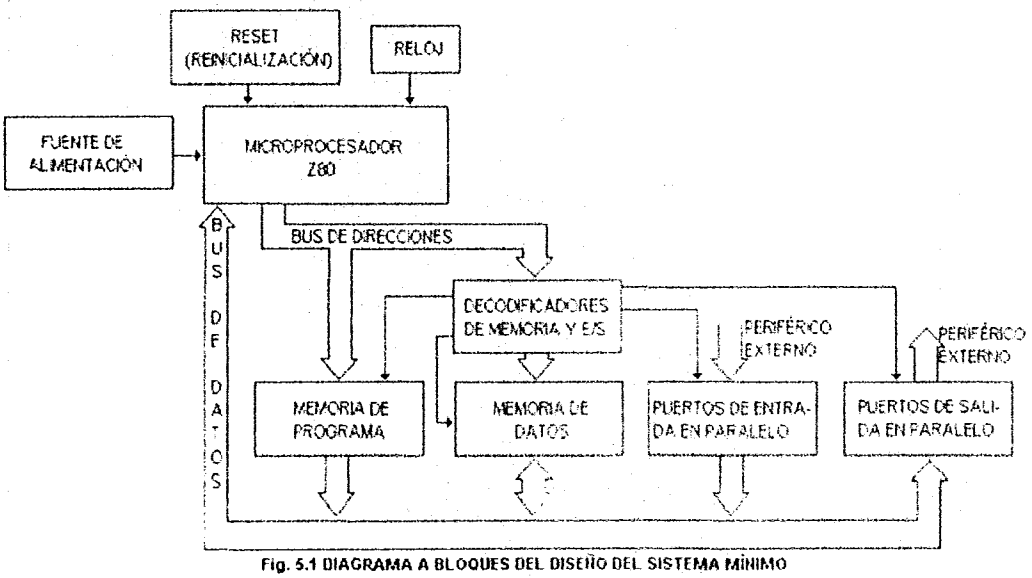


Fig. 5.1 DIAGRAMA A BLOQUES DEL DISEÑO DEL SISTEMA MÍNIMO

5.1 Diseño del "Corazón" (Reloj) del Microprocesador

El diseño del sistema mínimo se basa en un reloj unifase de 4 MHz, con niveles TTL (microprocesador Z80A). Estos **Pulsos de Reloj** pueden ser generados de varias formas, el método utilizado en este diseño se muestra en la figura 5.2. Como puede apreciarse, la frecuencia de oscilación se encuentra determinada por el cristal de 4 MHz. Sin embargo, para que el circuito trabaje en una forma óptima, el producto **R1C1** debe ser mayor que el periodo de oscilación. Por ejemplo, asumiendo que deseamos la frecuencia mostrada. Esto representa un periodo de 4×10^{-6} segundos. De esta forma, el producto **R1C1** debe ser mayor a 4×10^{-6}, si se toma un valor R1 constante de 330Ω. Entonces, el valor de C1 debe ser igual o mayor a $4 \times 10^{-6}/330$ farads o aproximadamente 121.12×10^{-12} F (éstos resultados fueron redondeados a 120 pF en valores comerciales). Este valor del producto **R1C1** permite un cambio aceptable de fase a la frecuencia de oscilación del circuito de reloj, para simplificar aún más el diseño, el valor de R2 se toma análogo al de R1.

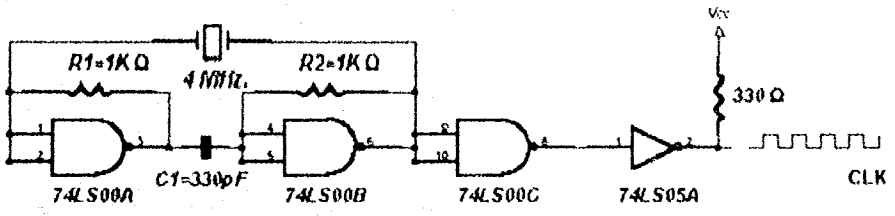


Fig. 5.2 GENERADOR DE PULSOS DE RELOJ PARA EL MICROPROCESADOR Z80A

Una característica muy importante a tomar en cuenta es el inversor de **Colector Abierto** y el resistor, los cuales en conjunto fungen como elementos de activación de 330Ω para +5 V. Esto satisface las necesidades de reloj de C. C. y C. A.

5.2 Diseño del Circuito de Reset del Sistema Mínimo

El circuito de $\overline{\textbf{Reset}}$ es un tipo especial de interrupción el cual fuerza al microprocesador a comenzar la ejecución de un programa desde una localidad específica en la memoria del sistema. Esta señal de entrada es aplicada al momento de ser alimentado el sistema con una fuente de potencia o cuando se desea que el microprocesador reinicie la ejecución del programa desde el principio. Esta necesidad de reinicio del sistema se debe en cierta medida a los posibles errores que tenga un programa y que necesiten ser verificados en forma continua. Esta señal interrumpe la ejecución del programa y carga el registro **Contador de Programa** (PC) con **0000H** (la más baja dirección de memoria).

Para diseñar el circuito de $\overline{\textbf{Reset}}$, se puede elegir entre uno manual o uno automático e incluso uno que combine ambas formas.

La siguiente figura muestra un tipo de circuito configurado como **Reset Manual**

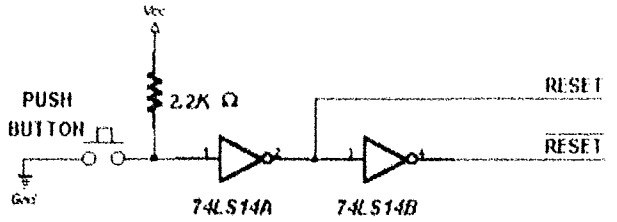


Fig. 6.3 CIRCUITO DE RESET MANUAL PARA EL MICROPROCESADOR Z80A

Este circuito tiene una forma de operación como se describe a continuación. la señal de salida de este circuito suele estar en un nivel lógico alto (1) hasta que se pulse el botón, pasando inmediatamente a un nivel lógico bajo. De esta forma, el microprocesador Z80 permanecerá en un estado de **Reset** mientras se mantenga oprimido el botón y no volverá a comenzar a ejecutar las instrucciones hasta que el botón sea liberado. Este circuito es de gran utilidad en las fases de diseño y revisión del sistema.

Si por el contrario, se desea que el sistema ejecute las instrucciones sin necesidad de un pulso de **Reset** previo por parte del usuario, el circuito de **Reset Manual** suele no emplearse y puede ser usado en su lugar un circuito de **Reset Automático**. La siguiente figura muestra la configuración de este circuito:

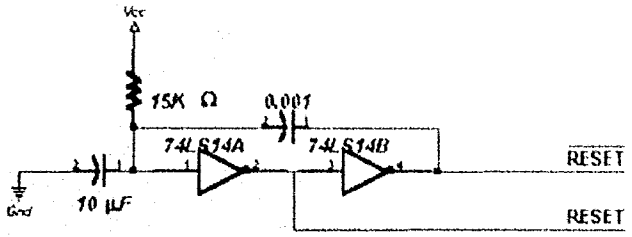


Fig. 6.4 CIRCUITO DE RESET AUTOMÁTICO PARA EL Z80A

Este circuito opera como sigue: cuando la alimentación de energía se aplica al sistema, el capacitor de 10 µF se descarga completamente. El nivel lógico 0 en la entrada de la terminal 1 del circuito integrado 74LS14 se mantiene en ese mismo estado durante unos 50ms. El régimen largo de carga del capacitor genera a su vez un nivel lógico 0 (condición de **Reset**) al sistema mínimo hasta que el nivel de la entrada se eleve aproximadamente a 2 V (nivel lógico 1 TTL). Cuando la potencia total del sistema toma lugar, el tiempo que tarda el circuito de **Reset Automático** en alcanzar 2V constituye un pulso de **Reset** de encendido para el sistema mínimo de unos 35 ms. Para que el sistema mínimo regrese a sus condiciones iniciales, es preciso desconectar la alimentación.

Para evitar los contratiempos anteriores, se requiere de un sistema que combine el sistema de **Reset Automático y Manual**. La siguiente figura muestra la combinación de ambos tipos de circuitos de **Reset**:

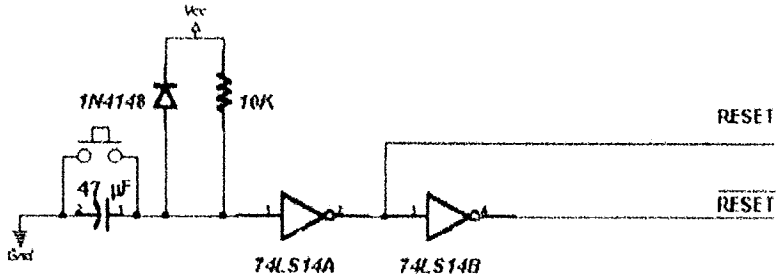


Fig. 5.5 CIRCUITO DE RESET AUTOMÁTICO Y MANUAL COMBINADO

Este circuito permite al sistema mínimo comenzar la ejecución del programa inmediatamente después del encendido. La ejecución de las instrucciones pueden ser detenidas y reiniciadas pulsando el botón de **Reset**. Los inversores 74LS14 con disparador **Schmitt** aumentan la fiabilidad de diseño del circuito. Cuando se desconecta la alimentación, el empleo del diodo para descargar rápidamente el capacitor asegura que se genere un pulso si se vuelve a aplicar repentinamente la alimentación. Puesto que las interferencias de baja frecuencia (**Glitches**) de la línea de alimentación suelen tener una corta duración, el régimen de descarga del capacitor debe ser lo bastante rápido para que deje de generarse un pulso de **Reset** una vez que se haya restablecido la alimentación de energía.

5.3 Buffering

Como se ha mencionado anteriormente, el microprocesador Z80 tiene la capacidad para direccionar hasta un total de 56536 localidades diferentes de memoria física (denominados genéricamente como 64Kb) y hasta 256 **Puertos de Entrada/Salida** individuales. Para tal fin, el microprocesador cuenta con un **Bus de direcciones** de 16 bits, etiquetados desde A_0 hasta A_{15}. Donde A_0 es el **LSB (Less Significative Bit o Bit Menos Significativo)** y A_{15} es el **MSB (More Significative Bit o Bit Más Significativo)**.

La **Sección de Control** del microprocesador activa al registro **Contador de Programa (PC)** para la próxima instrucción a ser ejecutada, en el ciclo de **Búsqueda de Instrucción (Fetch)** coloca el contenido del registro PC en el **Bus de Direcciones**. Durante las instrucciones de entrada/salida, los ciclos de temporización adicionales colocan la dirección del **Puerto de Entrada/Salida** en los 8 bits de menor peso (A_0-A_7) del **Bus de Direcciones**. Dado que este bus debe controlar las entradas y salidas de muchos **Puertos de Entrada/Salida** en paralelo, todos los cuales consumen potencia de entrada, el **Bus de Direcciones** debe tener una corriente de salida que pueda satisfacer la demanda de carga de los dispositivos periféricos conectados al sistema.

Para este propósito, el microprocesador puede absorber una carga máxima de 1.8 mA o una carga TTL en cada terminal. Debido a estas circunstancias, precisaría utilizar memorias de baja potencia y circuitos integrados interfaz de dispositivos periféricos, debido a que estos dispositivos son costosos y van en contra de la filosofía de diseño del sistema mínimo se descartará esta elección.

El empleo de dispositivos TTL y circuitos integrados de baja densidad para funciones de decodificación requieren más alimentación procedente de los bus del microprocesador Z80. La siguiente tabla indica la carga requerida para varios dispositivos:

Dispositivo	Corriente de entrada
TTL Normal (7404, 7442, etc.)	1.6 mA
TTL Schottky de baja potencia (74LS04, 74LS42, etc.)	0.18 mA
2708 (EPROM 1K x 8)	10μA
2114 (PROM 1K x 4)	10μA
2716 (EPROM 2K x 8)	10μA
2102 (PROM 1K x 1)	10μA
8212 (Latch de 8 bits)	0.25 mA
8T97 (Controlador de 6 bits)	1.0 mA

Tabla 5.1 CARGA DE CONSUMO DE DIVERSOS DISPOSITIVOS

Como puede verse a simple vista, los dispositivos que consumen una gran cantidad de carga son los circuitos integrados TTL, los circuitos integrados LSTTL ahorran potencia. Sin embargo, si se desea utilizar estos elementos se debe hacer tal sustitución en todo el sistema mínimo.

Con una carga de excitación de 1.8 mA se puede emplear dispositivos LSTTL para la decodificación de **Direcciones de Memoria**, pero existe el inconveniente de limitar el abanico de salidas **Fanout** en cada **Línea de Dirección** a 9 entradas. Sin embargo, este procedimiento tampoco es recomendable por las limitantes expuestas anteriormente.

Mejor que intentar optimizar el diseño hasta un grado tal que el usuario se vea precisado a tomar en consideración cada μA consumido por el sistema, es más fácil añadir **Buffers** que aumentar la potencia de salida de los bus del sistema mínimo hasta un punto en el que la carga no sea un factor importante. Además de lo anterior, permite al usuario añadir sus propios circuitos integrados TTL sin llegar a preocuparse excesivamente por la carga de los buses del sistema mínimo.

Para conseguir una salida de alta potencia a partir del **Bus de Direcciones**, se utiliza un dispositivo de **Amplificación Intermedia** (proceso denominado **Buffering**).

La figura número 5.6 es el diagrama y tabla de verdad del **Controlador de Bus** 74LS367 (un **Controlador de Bus** equivalente es el 8T97). Este circuito integrado es un buffer triestado capaz de controlar 15 cargas de unidad TTL o 60 cargas Schottky de baja potencia cuando los pines de control **Enable** se encuentra en un nivel lógico bajo (0). Cuando la señal Enable se encuentre en

un nivel lógico alto (1), las salidas de datos son forzadas a un estado de alta impedancia o estado inactivo.

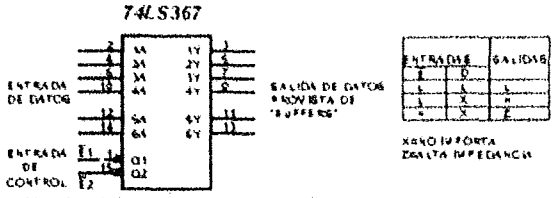


Fig. 5.6 DIAGRAMA Y TABLA DE VERDAD DEL CI 74LS367

5.3.1 Bus de Direcciones con Buffer

La siguiente figura muestra el circuito con buffer del **Bus de Direcciones**:

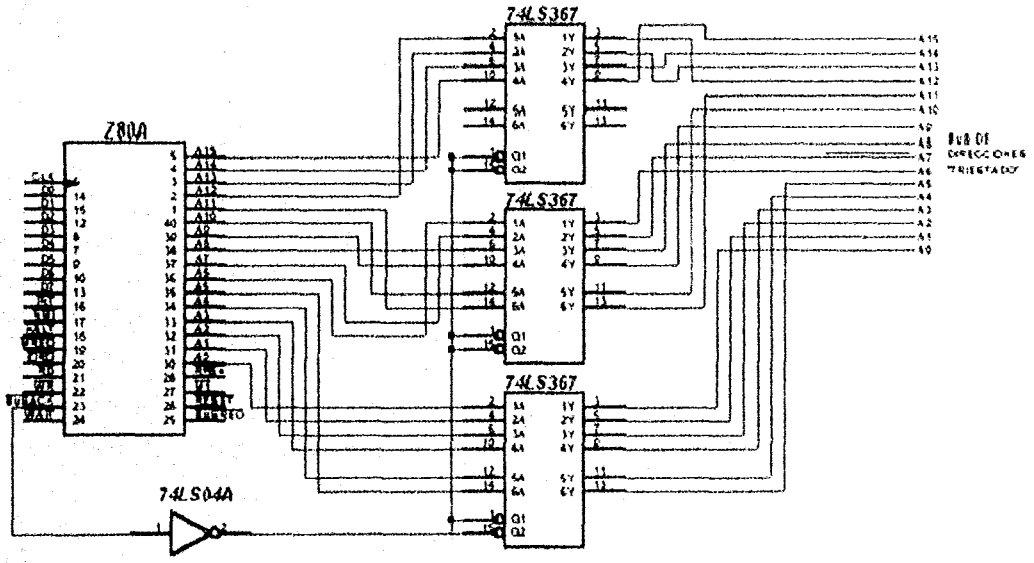


Fig. 5.7 BUS DE DIRECCIONES CON BUFFERS "TRIESTADO"

La función triestado de los dispositivos buffers (74LS367) se controlan mediante la señal **Busack**. Esta señal conmuta el control del **Bus de Direcciones** con un dispositivo externo durante las operaciones de **Acceso Directo a Memoria** (DMA o Direct Memory Access). Cuando no existe una situación de DMA, la señal **Busack** se encuentra en un nivel lógico alto (1), habilitando a los buffers para que permitan pasar todas las salidas procedentes del microprocesador Z80. Cuando se recibe una señal de petición DMA, la señal **Busack** pasa a un lógico nivel bajo poniendo a los buffers en un modo de alta impedancia. Esto permite que un dispositivo externo pueda acceder directamente a la memoria sin necesidad de utilizar el microprocesador como un elemento intermedio. De esta forma se consiguen grandes velocidades de transferencia de datos superiores a las que puede manejar el microprocesador.

5.3.2 Bus de Datos con Buffer

Por las mismas razones que se necesita para proveer un buffer al **Bus de Direcciones**, se aplica un buffer al **Bus de Datos**. Excepto que se requiere una ligera modificación, ya que el **Bus de Datos** es bidireccional, ello indica que los datos fungen tanto de entrada como de salida hacia/desde el microprocesador.

Cuando el microprocesador Z80 escribe un octeto de datos en una posición determinada de la memoria, los datos fluyen desde el microprocesador hacia la memoria. En forma contraria, cuando el microprocesador efectúa una lectura de un octeto de datos de la memoria, los datos fluyen de la memoria hacia el microprocesador.

De esta forma, se requiere que los controladores de **Bus de Datos** sean bidireccionales o estar conectados de tal forma que puedan realizar la misma función.

Para lograr este propósito, pueden interconectarse 3 circuitos integrados 74LS367 o si se desea minimizar la cantidad de chips utilizados, pueden interconectarse dos chips 8212. La siguiente figura muestra el diagrama y la descripción de los pines más importantes de dicho circuito (8212):

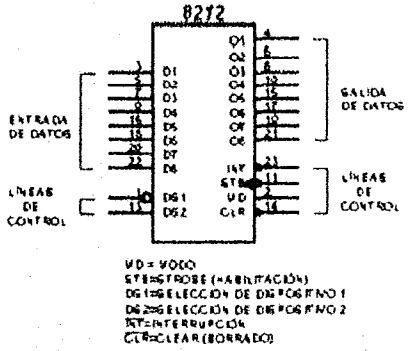


Fig. 5.8 DIAGRAMA DEL BUFFER DE 8 BITS 8212 DE INTEL

El circuito integrado 8212 es concebido y producido por Intel como un dispositivo de entrada o de salida de datos. Está construido internamente como un registro tipo latch de 8 bits. El chip puede enclavarse continuamente de modo que los datos puedan fluir a través del mismo, o puede desconectarse para bloquear el flujo de datos, además de presentar una salida triestado, idónea para la filosofía de diseño del sistema mínimo. Además de las características mencionadas, este circuito integrado cuenta con las siguientes características:

a) Registro de datos para 8 bits totalmente en paralelo y memoria intermedia.
b) Flip-flop para petición de servicio en la generación de interrupciones.
c) Corriente de carga reducida –0.25 mA máximo.
d) Salidas triestado.
e) Corriente de retorno a la salida de 15 mA.
f) Tensión de salida de 3.65 V para nivel alto

g) Puesta a cero asíncrona del registro

h) Sustitución de memoria intermedia, circuito de enclavamiento (latch) y multiplexores en los sistemas de microcomputadoras

i) Reduce la cuenta de encapsulamiento en cada sistema.

La siguiente figura muestra la forma de conexión de ambos circuitos integrados al microprocesador Z80 para fungir como **Bus de Datos** bidireccional con buffer.

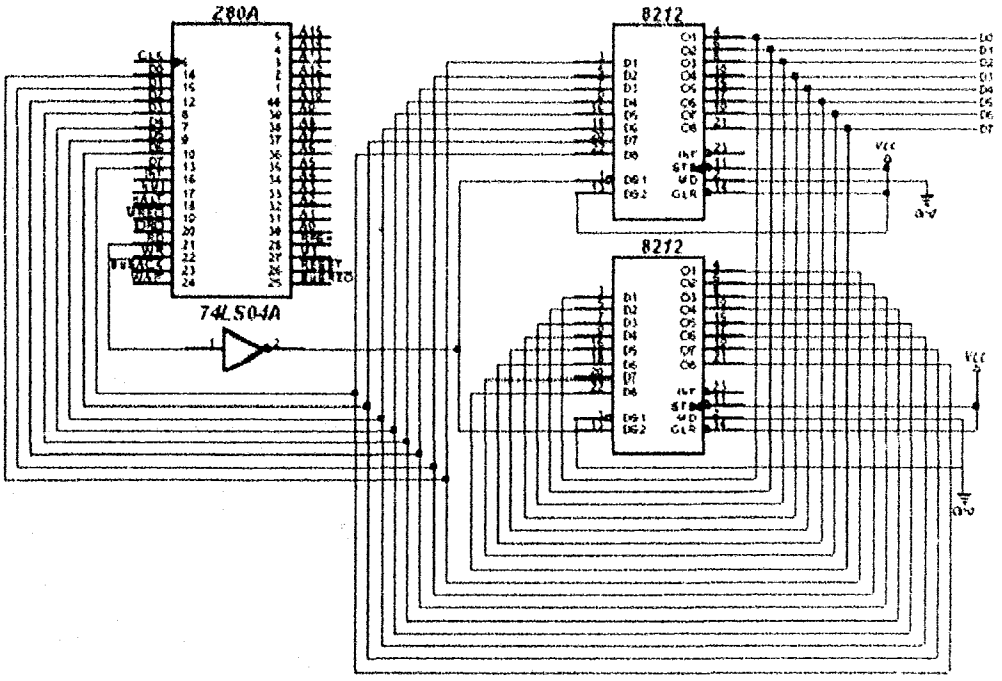


Fig. 5.8 BUS DE DATOS BIDIRECCIONAL CON BUFFER DE 8 BITS 8212 DE INTEL

Los dos circuitos están cableados en sentidos opuestos. El circuito integrado superior dirige los datos desde el microprocesador central hacia la memoria, mientras que el circuito integrado inferior canaliza los datos hacia el microprocesador central. El control se ejerce a través de un solo inversor conectado a la señal de salida de control $\overline{RD}$ del microprocesador Z80. Cuando se encuentra inactiva, la señal $\overline{RD}$ se encuentra en un estado normalmente de nivel lógico bajo (0), salvo en las operaciones de escritura. Esto hace posible que el circuito integrado superior se desactive, poniéndose en un modo triestado de alta impedancia, activando el circuito integrado inferior, lo cual permite el flujo de datos desde la memoria o **Puertos de Entrada/Salida** hacia el procesador central. Cuando la señal $\overline{RD}$ adquiere un nivel lógico alto (1) durante una petición de escritura de memoria, el sistema se invierte activando el circuito integrado superior y desactivando el chip inferior, lo cual permite el flujo de datos desde el microprocesador central hacia la memoria o hacia los **Puertos de Entrada/Salida**.

La configuración del buffer anterior se basa en la suposición natural de que cuando el microprocesador central no está escribiendo datos, debe de estar realizando una operación de lectura.

Si se desea disminuir aún más el costo del diseño del sistema mínimo, se puede reemplazar el par de circuitos integrados 8212 por tres circuitos integrados 74LS367. El único inconveniente es la adición de cableado y espacio requerido para alojar otro circuito integrado. Sin embargo, la adición de otro circuito integrado se compensa con el mayor tamaño de los circuitos integrados 8212. La siguiente figura muestra como pueden interconectarse tres circuitos integrados 74LS367 para reemplazar al par de circuitos integrados 8212:

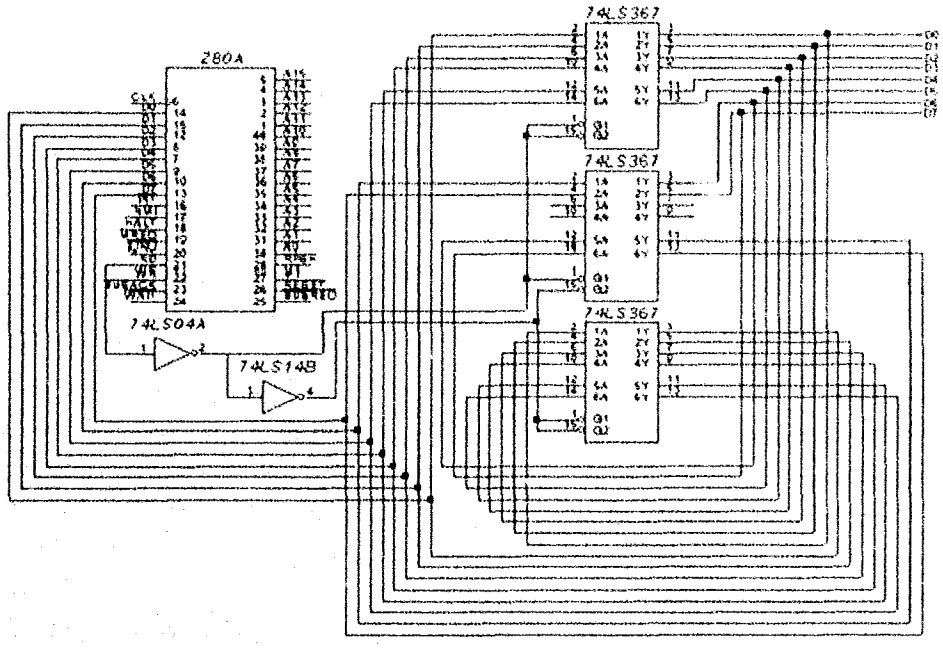


Fig. 5.10 BUS DE DATOS BIDIRECCIONAL CON BUFFER DE 4 BITS 74LS367

5.3.3 Bus de Control con Buffer

La parte final de las conexiones directas al microprocesador son las señales del **Bus de Control**, las cuales coordinan los **Puertos de Entrada/Salida**, los datos de los canales y finalmente direcciona los datos hacia y desde el microprocesador central.

Los motivos por los cuales son necesarios proveer de buffer al **Bus de Control** son análogos al **Bus de Datos** y al **Bus de Direcciones**. Además, las señales de **Control Invertidas y no Invertidas** proporcionan al diseñador del sistema la facilidad para una expansión rápida y fácil de las capacidades y los dispositivos periféricos añadidos al sistema.

Las **Señales de Control** con buffer se muestran en la siguiente figura.

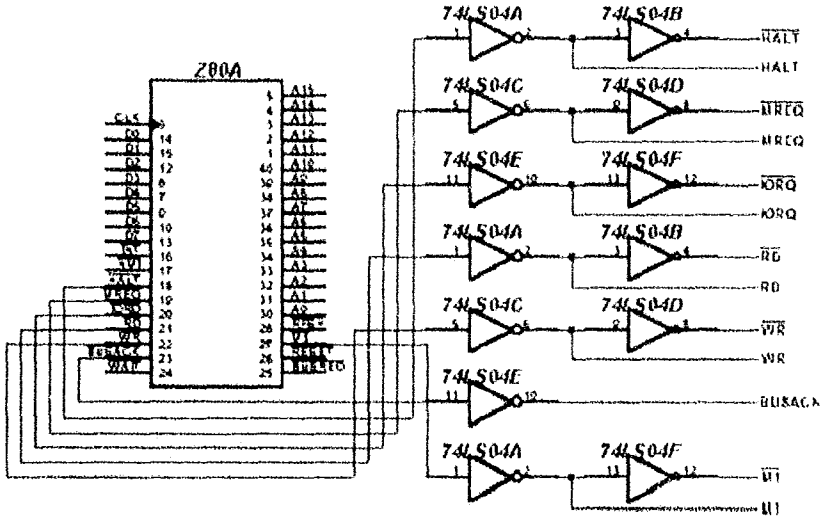


Fig. 5.11 BUS DE CONTROL CON BUFFERS UTILIZANDO INVERSORES DOBLES

Después de haber estudiado individualmente cada sección e implementado su buffer respectivo, la figura 5.12 muestra el diagrama que combina los elementos anteriores.

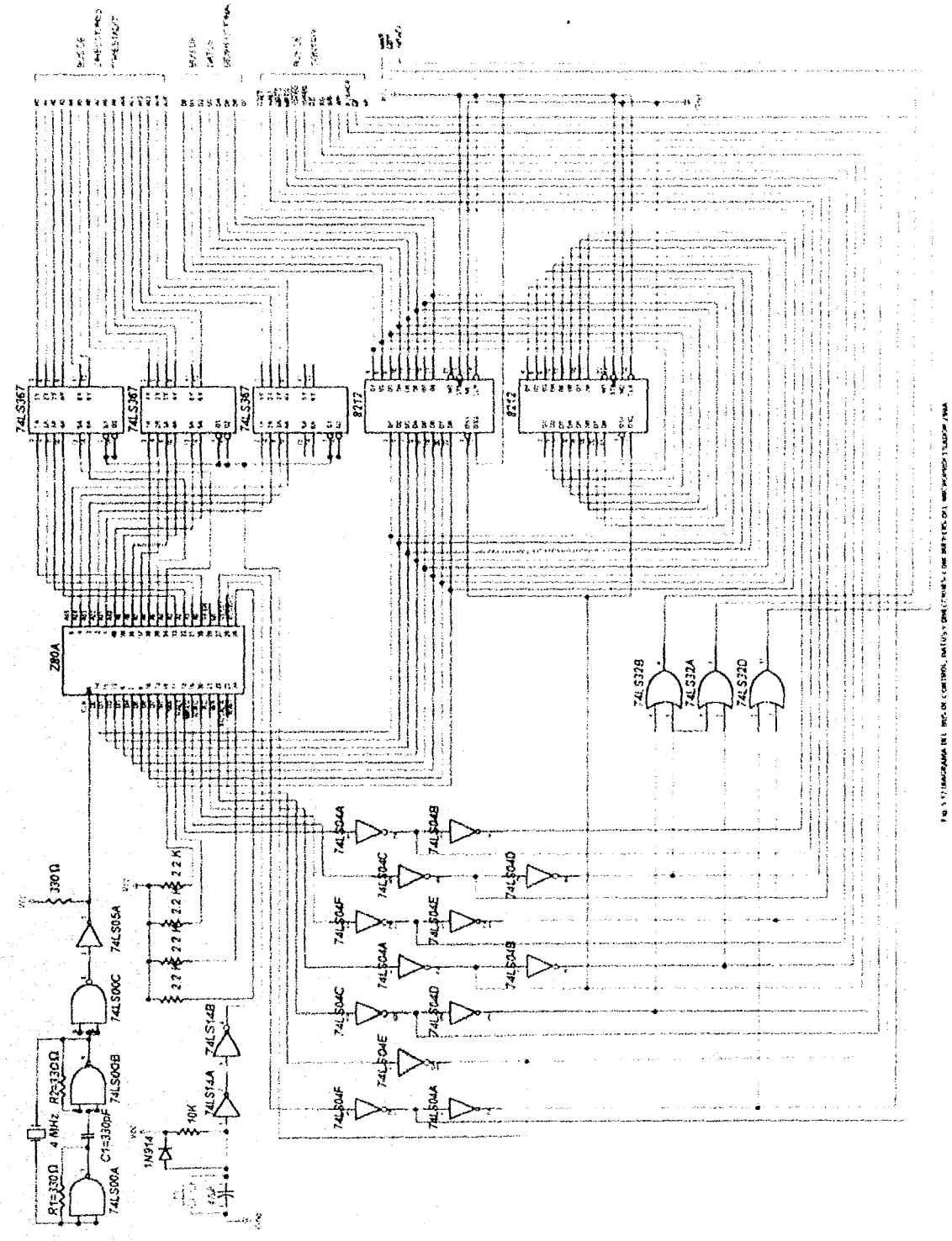
5.4 Pruebas

Como se había mencionado anteriormente, antes de proseguir con el proceso de diseño del sistema mínimo, se debe de probar el funcionamiento de cada una de las partes básicas que integran el sistema y corregir en su momento las posibles fallas que pudieran surgir por malas conexiones o por dispositivos dañados. Para este caso, se recomienda seguir la siguiente secuencia de verificación:

a) **Reloj**: Para comprobar el funcionamiento integral del circuito **Generador de Pulsos de Reloj**, se requiere de un osciloscopio o de un frecuenciómetro para registrar la frecuencia de pulsos exacta.

b) **Circuito de $\overline{\text{Reset}}$**: Independientemente del diseño elegido, el subsistema debe tener una salida normalmente alta (excepto en el sistema de **$\overline{\text{Reset}}$ Automático**). Cuando se aplique la alimentación de energía o se emita una pulsación del botón, éste debe de pasar a un nivel lógico bajo, indicando su buen funcionamiento. Este nivel lógico bajo hará que el sistema se vaya a un estado de **$\overline{\text{Reset}}$**.

c) **Bus de Control**: Dado que éste es el buffer más simple de los subsistemas mencionados con anterioridad, solo requiere aplicar una señal conocida en las entradas y verificar su correspondiente señal análoga en las salidas. Por ejemplo, si se aplica una entrada de nivel lógico bajo al terminal 1 del primer circuito integrado, la terminal 2 debe de registrar un nivel lógico alto y la terminal 4 un nivel lógico bajo (indicando su correcto funcionamiento).


74LS367
74LS367
74LS367
8212
8212
Z80A
74LS32B
74LS32A
74LS32D
74LS04A
74LS04B
74LS04C
74LS04D
74LS04E
74LS04F
74LS04A
74LS04B
74LS04C
74LS04D
74LS04E
74LS04F
330Ω
74LS05A
74LS00C
74LS00B
74LS00A
R=330Ω
R=330Ω
C1=330pF
4 MHz
74LS14B
74LS14A
10K
1N914
2.2K
2.2K
2.2K
2.2K

d) **Bus de Direcciones con Buffer**: Para la verificación del correcto funcionamiento de este subsistema, se realiza el siguiente procedimiento con sumo cuidado se suprime el circuito integrado Z80A. Se pone a masa el pin número 13 del segundo circuito integrado del **Bus de Control**, hecho esto, las salidas de los circuitos integrados 74LS367 deben tener un nivel lógico alto (indicando que se encuentran en un estado de alta impedancia). Seguidamente, se procede a unir el mismo terminal del circuito integrado del **Bus de Control** a través de una resistencia de 2.5 KΩ a 5 V+, esto activará todos los buffers del **Bus de Direcciones**. Inicialmente, todas las salidas están a un nivel lógico alto, se procede secuencialmente a poner a masa las líneas del **Bus de Direcciones** A_0 hasta A_{15}, esta señal debe observarse al final del **Bus de Direcciones** provista de buffer. Cuando las 16 líneas del **Bus de Direcciones** se hayan verificado con éxito, se procede a realizar lo mismo en el **Bus de Datos**.

e) **Bus Bidireccional de Datos con Buffer**: La forma de probar este bus es análoga a probar el **Bus de Direcciones**, excepto por una ligera variante. Se debe probar el flujo de datos desde el microprocesador hacia la memoria y los **Puertos de Entrada/Salida** y desde éstos hacia el microprocesador. Para realizar la comprobación del **Bus Bidireccional de Datos**, se procede a realizar lo siguiente: se pone a masa el pin número 1 del segundo circuito integrado del **Bus de Control** simulando con ello un estado de lectura. En esta posición, los datos deben de fluir desde la memoria y los **Puertos de Entrada/Salida** hacia el microprocesador, para lo cual se aplica alternadamente niveles lógicos bajos y altos al **Bus de Direcciones** provista de buffer (más concretamente, se conecta en forma alternada a masa y a 5 V+ a través de un resistor de 2.2 KΩ las terminales de datos de los circuitos integrados 8212). Para verificar las operaciones de escritura de datos hacia la memoria y los **Puertos de Entrada/Salida**, se procede a unir mediante una resistencia de 2.2 KΩ el mismo terminal 1 del segundo circuito integrado del **Bus de Control** a 5 V+ y se verifica aplicando alternadamente niveles lógicos altos y bajos al **Bus de Datos** procedentes de la memoria y de los **Puertos de Entrada/Salida**.

5.5 Lectura/Escritura de Memoria y Puertos Mapeados

Como se ha mencionado anteriormente, existe una ligera diferencia entre controlar **Puertos de Entrada/Salida Mapeados** y **Aislados** (diferencia marcada por las **Señales de Control** utilizadas para su lectura/escritura, número máximo de **Puertos de Entrada/Salida** que pueden controlarse dado el número de **Bits de Direcciones** a las que se conectan, etc.).

Dado que la técnica elegida para el diseño del sistema mínimo es la utilización de **Puertos de Entrada/Salida Mapeados**, nos centraremos en las **Señales de Control** para éste.

Recuerde que las **Señales de Control** tienen toda la información necesaria para realizar la operación requerida sobre un **Puerto de Entrada/Salida** externo mediante la selección de las señales adecuadas. Para controlar las operaciones básicas de lectura/escritura de la memoria y de los **Puertos de Entrada/Salida**, las señales de control requeridos para este caso son $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ y $\overline{\text{WR}}$, además de la activación de la localidad de memoria a la cual se desea leer o

escribir o la dirección del **Puerto de Entrada/Salida** con el cual se desea establecer una operación de lectura o escritura. Recuerde nuevamente las definiciones de las señales mencionadas

1) $\overline{\text{MREQ}}$ **(Petición de Memoria)**. Siempre que sucede una transacción entre el microprocesador central y la memoria, esta señal se va a un nivel lógico bajo
2) $\overline{\text{IORQ}}$ **(Petición de Entrada/Salida)**: Siempre que suceda una transacción entre el microprocesador central y los **Puertos de Entrada/Salida Aislados**, esta señal se va a un nivel lógico bajo.
3) $\overline{\text{RD}}$ **(Petición de Lectura)**: Siempre que el microprocesador central lea datos de entrada de la memoria o de los **Puertos de Entrada/Salida**, esta señal se va a un nivel lógico bajo
4) $\overline{\text{WR}}$ **(Petición de Escritura)**: Siempre que el microprocesador central escribe datos a la memoria o a los **Puertos de Entrada/Salida**, esta señal se va a un nivel bajo.

Para diferenciar los procesos de lectura o de escritura de los **Puertos de Entrada/Salida** y de la memoria, las señales $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ y $\overline{\text{WR}}$ se conectan como muestra la siguiente figura

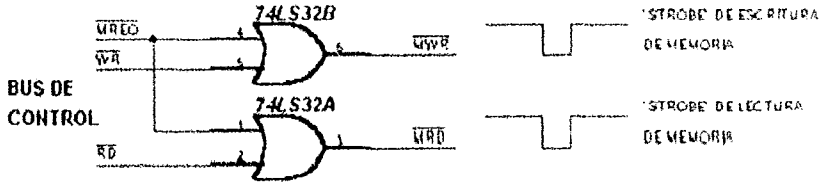


Fig. 5.13 DECODIFICACÓN DE LAS SEÑALES DE CONTROL PARA ENTRADA/SALIDA DE MEMORIA

Las dos señales de **Strobe** decodificadas definen las operaciones de lectura de la memoria o de los **Puertos de Entrada/Salida Mapeados** ($\overline{\text{MRD}}$), escritura de la memoria o de los **Puertos de Entrada/Salida Mapeados** ($\overline{\text{MWR}}$)

5.6 Mapeo de la memoria

La parte fundamental del proceso de diseño del sistema mínimo se centra en este proceso, ya que determina la cantidad de memoria de solo lectura **(Memoria de Programa)**, memoria de lectura/escritura **(Memoria de Datos)** y número de **Puertos de Entrada/Salida** que se pueden direccionar. Al elaborar el mapeo del sistema, se determina en primer instancia si los **Puertos de Entrada/Salida** que se conectarán al sistema son **Mapeados** o **Aislados**, el tamaño del sistema operativo que controlará los recursos del sistema, la cantidad de memoria de lectura/escritura requerida para el intercambio de datos temporales entre el microprocesador y los **Puertos de Entrada/Salida**, así como el número mínimo y máximo de éstos

La importancia de esta etapa estriba en la mejor elección de los circuitos integrados de control para decodificar las direcciones, los dispositivos de interfaz entre el microprocesador y los dispositivos periféricos, los niveles lógicos para activar y desactivar las interfaces, etc.

La siguiente tabla muestra el mapeo de la memoria del sistema[1] mínimo:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Función
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16 Kb de ROM
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8 Kb de RAM
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Habilitación de Teclado
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Hab. De comandos LCD
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Reset del Teclado
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	Hab. De lectura/escritura LCD
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Habilitación de decodificador
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Indefinido
1	0	0	0	0	0	0	0	0	0				1	0	0	
1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	Indefinido
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Habilitación de Datos LCD

Tabla 5.2 MAPEO DE LA MEMORIA DEL SISTEMA MÍNIMO

Como se puede ver de la tabla de mapeo anterior, se puede direccionar 16 Kb de **Memoria de Programa**, 8 Kb de **Memoria de Datos**, habilitar distintas funciones de dos **Puertos de Entrada/Salida** y dejar libres varias líneas para añadir otros **Puertos de Entrada/Salida**. La explicación completa del mapeo de memoria del sistema se explicará con mayor detalle en los siguientes subtemas.

5.6.1 Memoria de Programa

A partir del análisis de la tabla 5.2, se deduce que para poder direccionar 16 Kb de memoria **ROM** se requiere de la utilización de 14 bits del **Bus de Direcciones** (A_0-A_{13}). Además de lo anterior, el dispositivo de memoria **EPROM** que se va a utilizar es el D2732A. La configuración de dicho dispositivo se muestra a continuación:

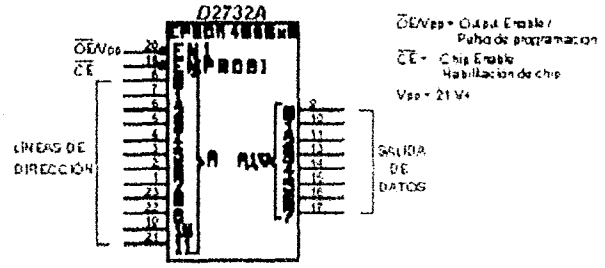


Fig. 5.14 DIAGRAMA DE LA MEMORIA EPROM D2732A DE INTEL

Como se puede ver en la gráfica, la memoria **EPROM D2732A** de Intel tiene la capacidad para almacenar 4,096 (4 Kb) octetos de datos en su estructura, para tal fin cuenta con doce **Líneas de Dirección** (numeradas de A_0-A_{11}), cuenta además con una línea de habilitación de circuito, una

[1] Para determinar la cantidad de localidades de memoria direccionables con una cantidad determinada de bits se calcula el valor de 2 elevado a la potencia del mismo número de bits. Por ejemplo, si deseamos calcular cuanta memoria EPROM podemos direccionar con 14 bits, efectuamos la siguiente operación:

línea de habilitación de datos (el cual funge también como **Strobe** de programación al proporcionársele un pulso de voltaje de 21 V+) y ocho líneas de datos.

Dado que el dispositivo puede almacenar solo 4 Kb de datos, se precisa interconectar 4 dispositivos semejantes para cumplir los requisitos de diseño. Además, como sólo se requiere de 12 bits para direccionar hasta 4 Kb de memoria (A_0-A_{11}), esto deja libres dos **Líneas de Dirección** para elegir el banco de memoria que se desee (A_{12} y A_{13}), la elección del banco en cuestión se basa en la siguiente tabla:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Rango	Función
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000h	Banco
0	0	0	0													-	Memoria
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFFh	EPROM 0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000h	Banco
0	0	0	1													-	Memoria
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFh	EPROM 1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000h	Banco
0	0	1	0													-	Memoria
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFFh	EPROM 2
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3000h	Banco
0	0	1	1													-	Memoria
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFF	EPROM 3

Tabla 5.3 MAPA Y RANGO DE DIRECCIONES DE LAS MEMORIAS EPROM

Como se puede ver de la tabla anterior, se requiere de dos **Líneas de Dirección** para seleccionar entre activar la memoria ROM, la memoria RAM y los **Puertos de Entrada/Salida**. Así como de dos **Líneas de Dirección** para elegir entre los cuatro bancos de memoria ROM. Esto permite pensar en utilizar el decodificador/demultiplexor 74LS139 de Motorola, el cual tiene la siguiente configuración:

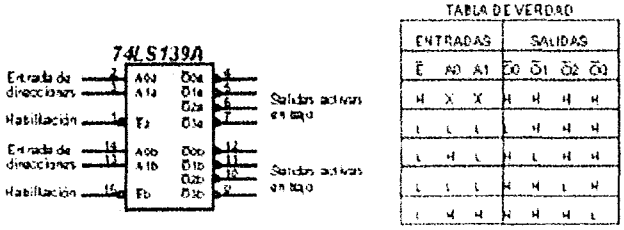


TABLA DE VERDAD

ENTRADAS			SALIDAS			
$\bar{E}$	A0	A1	$\bar{O}0$	$\bar{O}1$	$\bar{O}2$	$\bar{O}3$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = NIVEL DE VOLTAJE ALTO
L = NIVEL DE VOLTAJE BAJO
X = CONDICIÓN DE NO IMPORTA

Fig. 5.15 DIAGRAMA DEL DECODIFICADOR 74LS139 DE MOTOROLA

2^{13}=16,384 localidades de memoria direccionables.

La siguiente figura muestra la forma de interconectar los bancos de memoria ROM al microprocesador central, junto con las funciones de lectura y habilitación de chip:

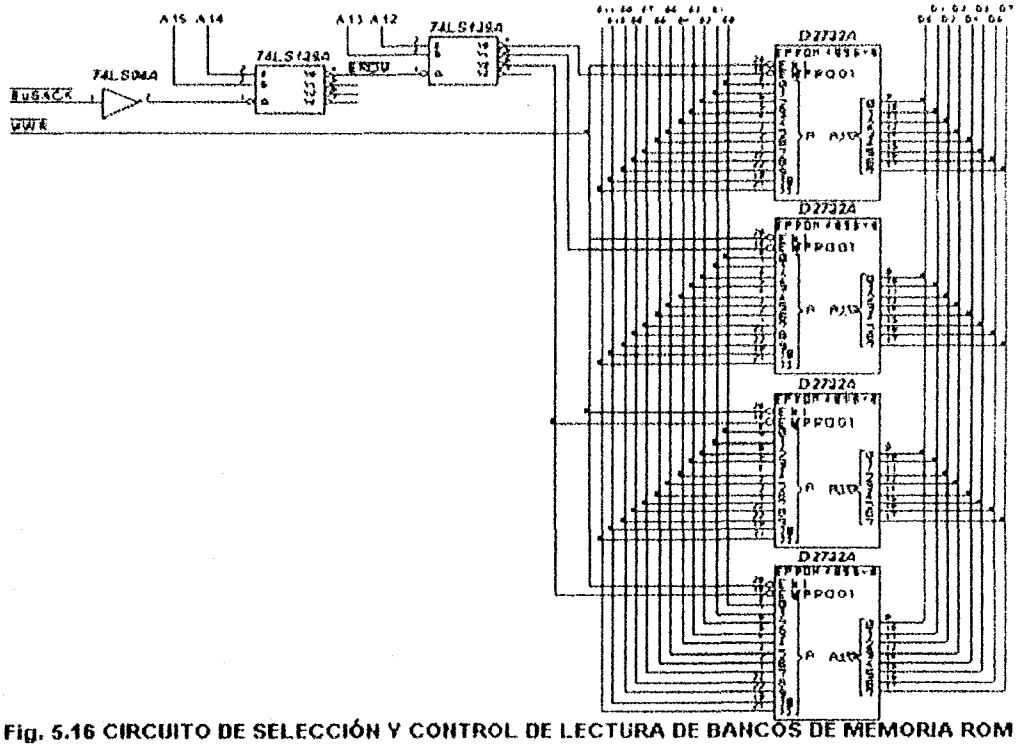


Fig. 5.16 CIRCUITO DE SELECCIÓN Y CONTROL DE LECTURA DE BANCOS DE MEMORIA ROM

Se debe prestar especial atención al pin número 1 del primer circuito integrado 74LS139. La señal $\overline{\text{BUSACK}}$ de entrada le permite controlar la memoria y liberarla cuando ocurra una petición de **DMA** por un dispositivo periférico externo. Al ocurrir este hecho, la señal $\overline{\text{BUSACK}}$ se va a un nivel lógico bajo, indicando que el microprocesador ha reconocido un **DMA** y que está llevando a un estado inactivo de alta impedancia los **Buses de Datos, Dirección y Control** para que puedan ser manipulados por el dispositivo periférico externo que solicito el **DMA**.

5.6.2 Memoria de Datos

En forma análoga a la interconexión de los dispositivos de memoria **ROM**, se conectará la memoria **RAM** con unas ligeras variantes a saber: en primer instancia se utilizará dispositivos de memoria **RAM HM6116L** (dispositivos equivalentes son los circuitos integrados 2128), los cuales tienen una capacidad de almacenamiento de 2 Kb, las características de este dispositivo se muestran en la siguiente figura:

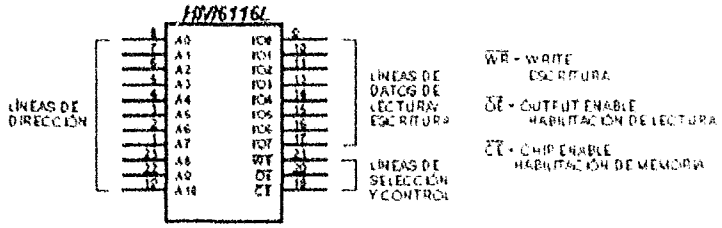


Fig. 5.17 DIAGRAMA DE LA MEMORIA RAM HM6116L DE 2KB OCTETOS DE LECTURA/ESCRITURA

Para poder direccionar un banco de memoria individual, se requiere de 11 **Líneas de Dirección** (A_0-A_{10}).

La siguiente tabla muestra la forma de mapeo de memoria para la RAM.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Rango	Función
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000h	Banco
0	1	0	0	0	...											-	Memoria
0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	47FFh	RAM 0
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	4800h	Banco
0	1	0	0	1	...											-	Memoria
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFFh	RAM 1
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000h	Banco
0	1	0	1	0	...											-	Memoria
0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	57FFh	RAM 2
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	5800h	Banco
0	1	0	1	1	...											-	Memoria
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFFF	RAM 3

Tabla 5.4 MAPA Y RANGO DE DIRECCIONES DE LAS MEMORIAS RAM

De igual forma, se cuenta con dos **Líneas de Direcciones** para seleccionar entre activar los bancos de memoria **ROM**, **RAM** o los **Puertos de Entrada/Salida** (A_{15} y A_{14}). Para elegir el banco de memoria RAM, se cuenta con dos **Líneas de Dirección** (A_{12} y A_{11}), lo cual permite volver a pensar en utilizar el circuito integrado 74LS139 para decodificar las direcciones de memoria **RAM**. La figura número 5.17 muestra la forma de interconexión de dichos dispositivos para que puedan ser controlados por el microprocesador central.

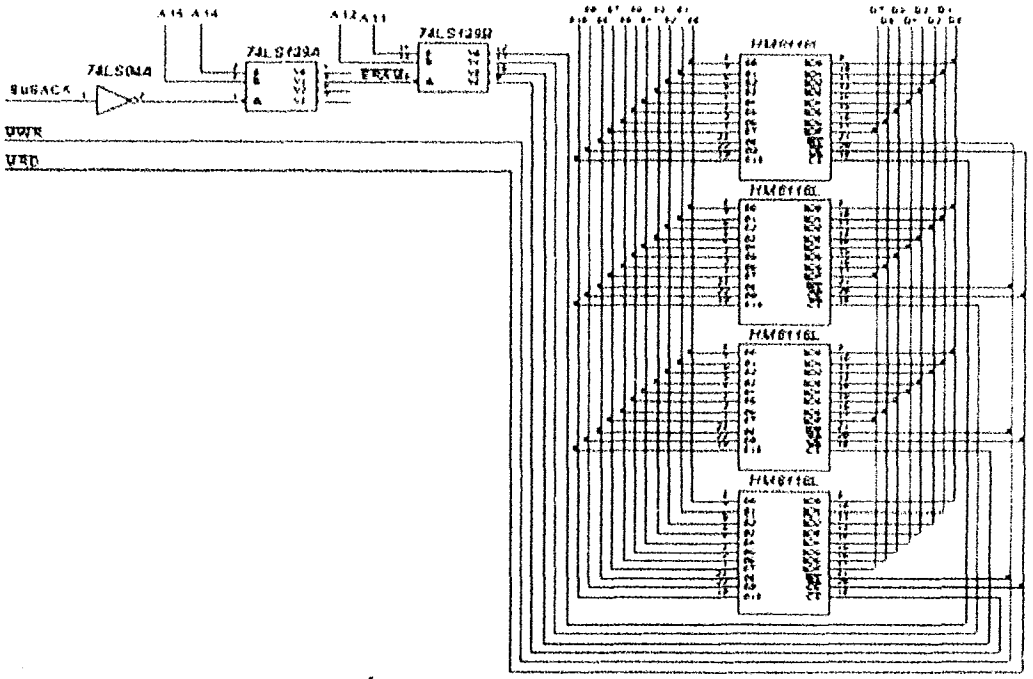


Fig. 5.17 CIRCUITOS DE SELECCIÓN Y CONTROL DE LECTURA/ESCRITURA DE MEMORIAS RAM

5.6.3 Puertos de Entrada/Salida

La siguiente fase de diseño del sistema mínimo concierne en la etapa de interconexión de los circuitos integrados convenientes que controlen las funciones de lectura/escritura de los **Puertos de Entrada/Salida**. Para tal fin, la siguiente tabla muestra el mapeo de memoria de los **Puertos de Entrada/Salida**:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Rango	Función
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H	SKBD
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8001H	CMDLCD
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8002H	CLRKBD
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	8003H	ENLCD
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	8004H	ENADEC
1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	8004H	INDEF
1	0	0	0	0	0	0	0	0	0	.	.	.	1	0	0	-	INDEF
1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	803CH	INDEF
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000H	DATLCD

Tabla 5.4 MAPA Y RANGO DE DIRECCIONES DE LOS PUERTOS DE ENTRADA/SALIDA

Dado que el circuito integrado 74LS139 tiene interconstruido dos decodificadores independientes, es conveniente pensar en utilizar uno de ellos para que funja como sistema de control de algunas caracteristicas de los **Puertos de Entrada/Salida**. Sin embargo, si se pensara en una posterior expansión del número de **Puertos de Entrada/Salida** que pueda controlar el sistema, sería menester añadir más circuitos decodificadores y utilizar las **Lineas de Direcciones**

libre del **Bus de Direcciones**. Pensado en esa posibilidad, se ha añadido un circuito integrado decodificador/demultiplexor 74LS138, cuyas características se muestran a continuación.

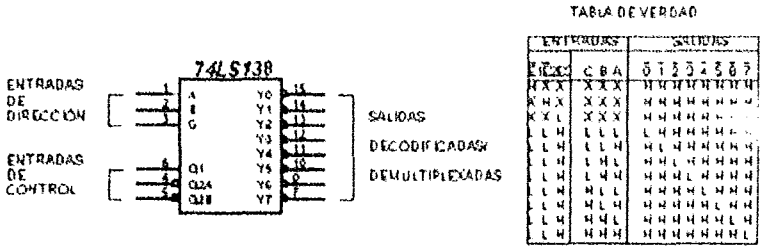


Fig. 5.18 DIAGRAMA DEL DECODIFICADOR/DEMULTIPLEXOR 74LS138 DE MOTOROLA

La siguiente figura muestra la forma de interconexión de los dispositivos mencionados anteriormente para controlar los **Puertos de Entrada/Salida**:

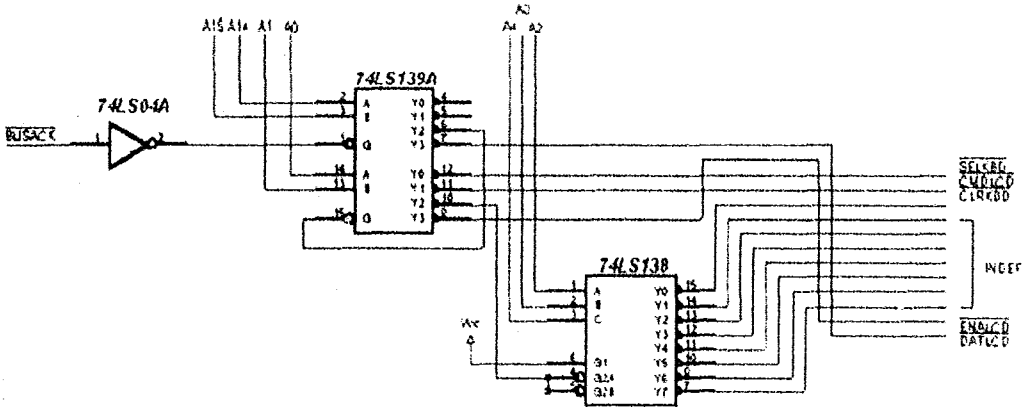


Fig. 5.19 CIRCUITO DE SELECCIÓN Y CONTROL DE LOS PUERTOS DE ENTRADA/SALIDA

Después de haber realizado la conexión individual de cada una de las secciones anteriores del sistema mínimo, la figura número 5.20 muestra el diagrama final de las conexiones analizadas anteriormente en conjunto.

5.7 Reconocimiento de Interrupciones

La última etapa de diseño del sistema mínimo comprende al circuito que se encarga de aceptar las peticiones de interrupción del teclado. Para poder diseñar en forma óptima este circuito, se debe proceder a enumerar algunos aspectos de su diseño:

a) El circuito de reconocimiento de interrupción se crea acorde con la filosofía de diseño del sistema mínimo mencionados anteriormente.

b) El circuito se enfoca hacia el modo de interrupción 2 del microprocesador Z80.

c) El circuito está creado con el fin de demostrar la forma de implementar el reconocimiento de interrupción, dado lo cual éste es sencillo y sólo manipula un único dispositivo periférico (el teclado).

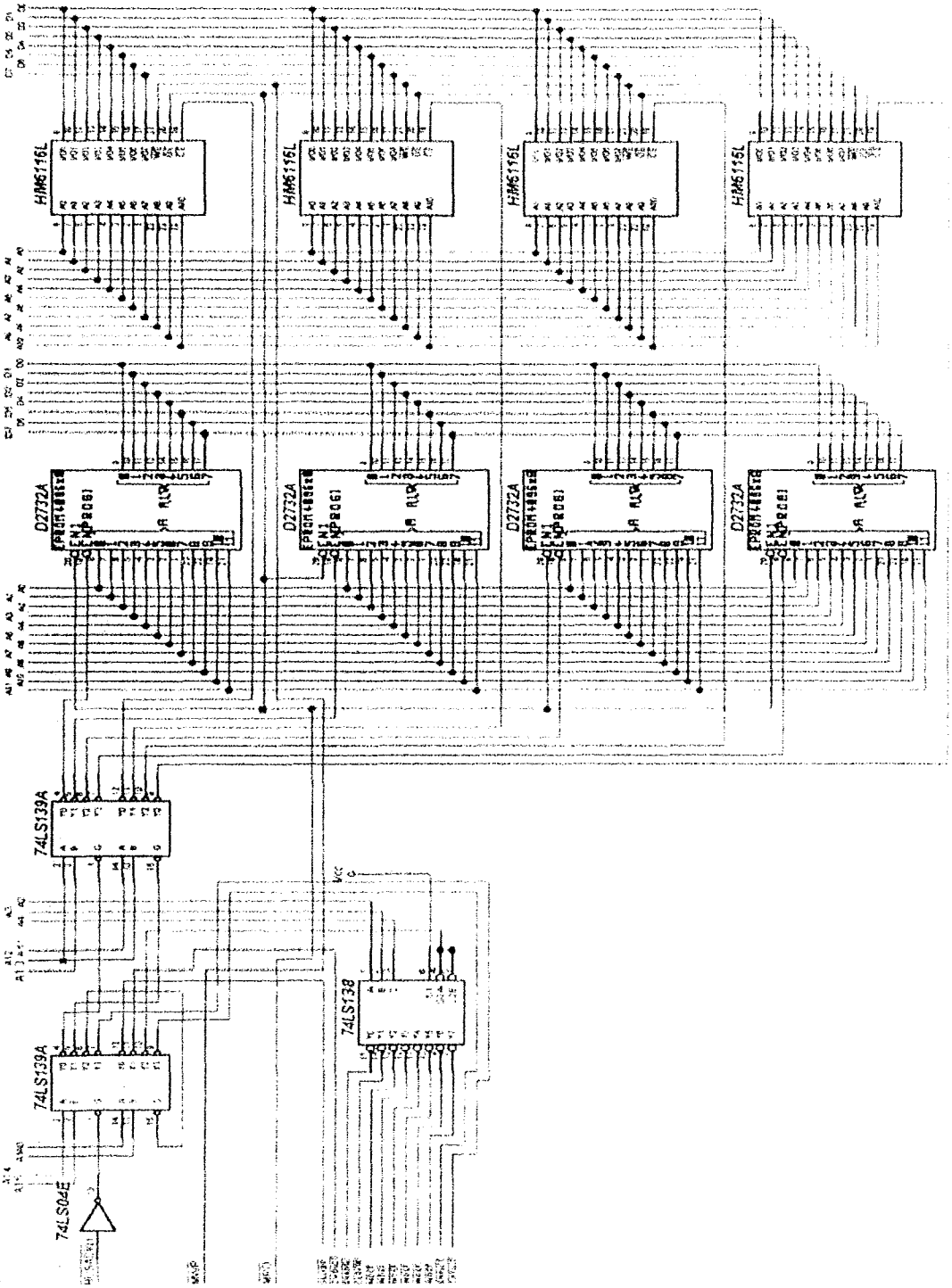


Fig. 5.21 DIAGRAMA DEL MAPEO DE MEMORIA PARA LOS PUERTOS DE ENTRADA/SALIDA DEL MICROPROCESADOR Z80A

De acuerdo con lo anteriormente dicho y al hecho de que para reconocer una interrupción el microprocesador Z80 genera en forma simultánea la activación a un nivel lógico bajos las señales $\overline{IORQ}$ y $\overline{M1}$, este circuito es el de mayor facilidad de implementación, como se puede observar en la siguiente figura:

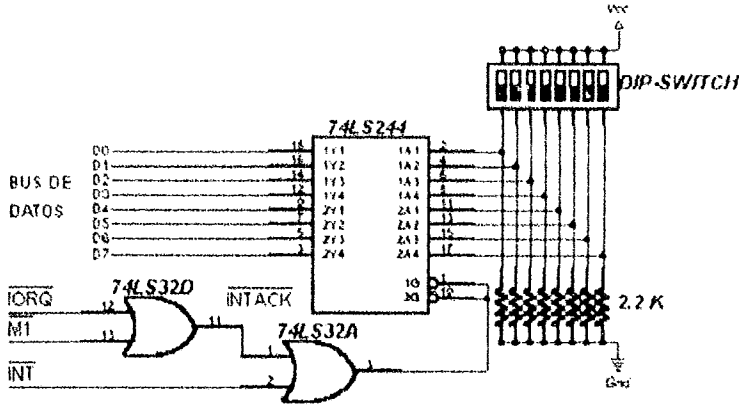


Fig. 5.21 CIRCUITO DE RECONOCIMIENTO DE INTERRUPCIONES DEL TECLADO

Se debe prestar especial atención al DIP-SWITCH conectado al buffer 74LS244, ya que permite modificar la tabla de **Vectores de Interrupción** a voluntad del diseñador. De igual forma, se debe observar con atención a la forma de implementar el par de circuitos lógicos **OR** (74LS32), ya que permiten que una **Señal de Interrupción** esté siempre activa hasta que el microprocesador genere las señales de **Reconocimiento de Interrupción** y se ejecute una secuencia de código para resetear la señal de interrupción ($\overline{INT}$). Estas características serán comentadas con mayor profundidad en el diseño del sistema operativo para el sistema mínimo.

5.8 Pruebas

Después de que se hayan añadido los componentes de las figuras 5.20 y 5.21 a la figura 5.12, se estará en posibilidades de probar las decodificaciones de **Entrada/Salida** y **Memoria**.

Para realizar las pruebas de los subsistemas, se debe insertar todos los circuitos integrados, exceptuando el microprocesador Z80A y los dispositivos de memoria. Se debe poner a un nivel lógico bajo el pin número 11 del segundo circuito integrado del **Bus de Control** ($\overline{BUSACK}$), esto hará que todas las salidas de los circuitos integrados decodificadores/demultiplexores (74LS139 y 74LS138) se vayan a un nivel lógico alto. Aplicando un nivel lógico bajo a la misma entrada del circuito integrado de control, se estará en condiciones de probar el cambio de nivel lógico de las salidas. Para tal efecto, se debe conectar secuencialmente y en forma de conteo binario (tomando la **Línea de Direcciones** 14 como el **LSB** y la **Línea de Direcciones** 15 como el **MSB**) a un nivel lógico alto los últimos bits de **Bus de Direcciones**, esto hará que las líneas de salida 4, 5, 6 y 7 del primer circuito decodificador/demultiplexor (74LS139) tomen niveles lógicos bajos en forma secuencial (indicando la activación de las señales $\overline{EROM}$, $\overline{ERAM}$, $\overline{SELDIS}$, $\overline{DATLCD}$).

Para probar las **Líneas de Selección** del banco de memoria **ROM**, se sigue aplicando el nivel lógico alto al pin número 11 del primer chip 74LS139. Posteriormente se aplica niveles lógicos bajos a las entradas 14 y 15 del **Bus de Direcciones**, finalmente se aplica secuencialmente y en forma de conteo binario (tomando la **Línea de Direcciones** 12 como el **LSB** y la **Línea de Direcciones** 13 como el **MSB**) a un nivel lógico alto, esto hará que las lineas de salida 4, 5, 6 y 7 del segundo circuito integrado decodificador/demultiplexor (74LS139) tomen niveles lógicos bajos en forma secuencial (indicando la activación de las señales **SROM0**, **SROM1**, **SROM2** y **SROM3**)

Para probar las **Líneas de Selección** del banco de memoria **RAM**, se sigue aplicando el nivel lógico alto al pin número 11 del primer chip 74LS139. Posteriormente se aplica un nivel lógico bajo al pin número 15 y un nivel lógico alto al pin número 14 del **Bus de Direcciones**, finalmente se aplica secuencialmente y en forma de conteo binario (tomando la **Línea de Direcciones** 11 como el **LSB** y la **Línea de Direcciones** 12 como el **MSB**) a un nivel lógico alto, esto hará que las lineas de salida 12, 11, 10 y 9 del segundo circuito integrado decodificador/demultiplexor (74LS139) tome niveles lógicos bajos en forma secuencial (indicando la activación de las señales **SRAM0**, **SRAM1**, **SRAM2** y **SRAM3**).

Para probar las **Líneas de Selección y Control** de los **Puertos de Entrada/Salida**, se sigue aplicando el nivel lógico alto al pin número 11 del primer chip 74LS139. Posteriormente se aplica un nivel lógico alto al pin número 15 y un nivel lógico bajo al pin número 14 del **Bus de Direcciones**, finalmente se aplica secuencialmente y en forma de conteo binario (tomando la **Línea de Direcciones** 0 como el **LSB** y la **Línea de Direcciones** 1 como el **MSB**) a un nivel lógico alto, esto hará que las lineas de salida 12, 11, 10 y 9 del primer circuito integrado decodificador/demultiplexor (74LS139) tomen niveles lógicos bajos en forma secuencial (indicando la activación de las señales **SELKBD**, **CMDLCD**, **ENADEC** y **DATLCD**). En este mismo punto, al activar la señal **ENADEC**, se estará en posibilidades de seleccionar en forma secuencial las salidas del circuito integrado 74LS138. Para tal efecto, se aplica secuencialmente y en forma de conteo binario (tomando la **Línea de Direcciones** 2 como el **LSB** y la **Línea de Direcciones** 4 como el **MSB**) a un nivel lógico alto, esto hará que las lineas de salida 14, 13...9 y 7 tomen niveles lógicos bajos en forma secuencial (indicando la activación de las señales **CLRKBD**, **INDEF**... **INDEF**).

Después de haber realizado en forma satisfactoria las pruebas anteriores, se procede a insertar todos los circuitos integrados exceptuando el microprocesador Z80A (incluyendo los circuitos mencionados en el capitulo anterior, excepto la pantalla de cristal liquido). Posteriormente, se activan las señales **DATLCD** y **CMDLCD**, se introduce una palabra de comando o de datos al **Bus de Datos**, posteriormente se activan las señales **CMDLCD** y **MWR** o **DATLCD** y **MWR** esto hará que dichas señales sean retenidas por los dispositivos latchs. En forma análoga, se activa la señal **ENALCD** y se verifican las señales correspondientes en el zócalo de la **Pantalla LCD**.

Para verificar el correcto funcionamiento de la interfaz de teclado, se conecta éste mismo al conector **DIM** y se presiona cualquier tecla conocida, se activan las señales **MRD** y **SKBD** esto

hará que los datos decodificados a **Código ASCII** aparezcan en el **Bus de Datos** del microprocesador.

Finalmente, para comprobar en forma dinámica el correcto funcionamiento del sistema mínimo, se prueba a programar y grabar los comandos de iniciación para la **Pantalla de Cristal Líquido** y lectura del teclado. Un ejemplo de dicho programa podría ser el que se muestra a continuación

```
0000                    ORG 00H
0000  21 FE 5F          LD  HL,5FFEH        ; STACK POINTER
0003  F9                LD  SP,HL
0004  3E 0E             LD  A,0EH           ; INT VECTOR
0006  ED 47              LD  I,A
0008  C3 50 01           JP  0150H
000B  C5                DELAY:   PUSH BC
000C  CD 23 00           CALL DLY
000F  06 4E              LD  B,+(4000/50)-2
0011  C3 14 00    LDLP:  JP  LDLY1
0014  C3 17 00    LDLY1: JP  LDLY2
0017  C3 1A 00    LDLY2: JP  LDLY3
001A  C6 00        LDLY3: ADD A,0
001C  10 F3               DJNZ LDLP
001E  3A 0B 00            LD  A,(DELAY)
0021  C1                  POP BC
0022  C9                 RET
0023  3D           DLY:   DEC A
0024  C8                 RET Z
0025  06 4F              LD  B,+(4000/50)-1
0027  C3 2A 00     DLP:   JP  DLY1
002A  C3 2D 00     DLY1:  JP  DLY2
002D  C3 30 00     DLY2:  JP  DLY3
0030  C6 00        DLY3:  ADD A,0
0032  10 F3               DJNZ DLP
0034  C3 37 00            JP  DLY4
0037  C3 3A 00     DLY4:  JP  DLY5
003A  00           DLY5:  NOP
003B  C3 23 00            JP  DLY
003E  00           ABDEL: NOP
003F  3E FA        ABDCH: LD  A,0FAH
0041  CD 0B 00            CALL DELAY
0044  10 F9               DJNZ ABDCH
0046  C9                  RET
0047  3E 00        INIKBD: LD  A,00H
0049  32 00 40            LD  (4000H),A       ; CLEAR BUFFER SIZE FIRST LINE
004C  32 02 40            LD  (4002H),A       ; CLEAR BUFFER SIZE SECOND LINE
004F  32 80 00            LD  (80H),A         ; CLEAR FLAGS LCD
0052   3C                 INC A
0053  32 01 40            LD  (4001H),A       ; INITIAL POSITION CURSOR FIRST LINE
0056  32 03 40            LD  (4003H),A       ;INITIAL POSITION CURSOR SECOND LINE
0059  C9                  RET
005A  C5           WRCHAR: PUSH BC
005B  32 00 C0            LD  (0C000H),A      ; LOAD DATA AND INSTRUCTION
005E  3E 05               LD  A,05H           ; DELAY 5 MSEC.
0060  CD 0B 00            CALL DELAY
0063  32 03 80            LD  (8003H),A       ; ENABLE LCD
0066  C1                  POP BC
0067  C9                  RET
0068  B7           WRLINE: OR  A             ; SIZE = 0?
0069  C8                  RET Z               ; EXIT
006A  47                  LD  B,A             ; TO LOOP
006B  7E           LPWRLN: LD  A,(HL)
006C  CD 5A 00            CALL WRCHAR
```

```
006F  23          INC HL
0070  10 F9       DJNZ LPWRLN
0072  C9          RET
0073  F5    EXITRD: PUSH AF
0074  3A 00 40     LD  A,(4000H)      ; LOAD SIZE
0077  21 05 40     LD  HL,4005H       ; LOAD ADDRESS BASE EXECUTE
                                       ; COMMAND
007A  21 01 80     LD  HL,8001H
007D  36 01        LD  (HL),01H       ; CLEAR LCD
007F  3E 05        LD  A,5H           ; DELAY 5 MSEC.
0081  CD 0B 00     CALL DELAY
0084  32 03 80     LD  (8003H),A      ; ENABLE LCD
0087  F1           POP AF
0088  CD 47 00     CALL INIKBD
008B  C9           RET
008C  C5    BACKSP: PUSH BC
008D  3A 00 40     LD  A,(4000H)      ; LOAD SIZE
0090  B7           OR  A              ; SIZE = 0?
0091  C8           RET Z              ; EXIT
0092  3D           DEC A
0093  32 00 40     LD  (4000H),A
0096  21 05 40     LD  HL,4005H       ; LOAD ADDRESS BASE
0099  CD 68 00     CALL WRLINE
009C  C1           POP BC
009D  C9           RET
009E  F5    DELLN: PUSH AF
009F  21 00 40     LD  HL,4000H
00A2  46           LD  B,(HL)
00A3  CD 8C 00 LPDEL: CALL BACKSP
00A6  10 FB        DJNZ LPDEL
00A8  F1           POP AF
00A9  C9           RET
00AA  E5    LCDINI: PUSH HL
00AB  C5           PUSH BC
00AC  21 01 80     LD  HL,8001H       ; COMMAND TO LCD
00AF  3E 32        LD  A,32H          ; DELAY 50 MSEC.
00B1  CD 0B 00     CALL DELAY
00B4  36 3C        LD  (HL),3CH       ; SYSTEM SET
00B6  3E 10        LD  A,10H          ; DELAY 10 MSEC.
00B8  CD 0B 00     CALL DELAY
00BB  32 03 80     LD  (8003H),A      ; ENABLE LCD
00BE  36 3C        LD  (HL),3CH
00C0  3E 10        LD  A,10H          ; DELAY 10 MSEC.
00C2  CD 0B 00     CALL DELAY
00C5  32 03 80     LD  (8003H),A      ; ENABLE LCD
00C8  36 3C        LD  (HL),3CH
00CA  3E 32        LD  A,32H          ; DELAY 50 MSEC.
00CC  CD 0B 00     CALL DELAY
00CF  32 03 80     LD  (8003H),A      ; ENABLE LCD
00D2  36 3C        LD  (HL),3CH
00D4  3E 32        LD  A,32H          ; DELAY 50 MSEC.
00D6  CD 0B 00     CALL DELAY
00D9  32 03 80     LD  (8003H),A      ; ENABLE LCD
00DC  36 0F        LD  (HL),0FH       ; DISPLAY ON/OFF
00DE  3E 32        LD  A,32H          ; DELAY 50 MSEC.
00E0  CD 0B 00     CALL DELAY
00E3  32 03 80     LD  (8003H),A      ; ENABLE LCD
00E6  36 01        LD  (HL),01H       ; CLEAR DISPLAY
00E8  3E 32        LD  A,32H          ; DELAY 50 MSEC.
00EA  CD 0B 00     CALL DELAY
00ED  32 03 80     LD  (8003H),A      ; ENABLE LCD
00F0  36 06        LD  (HL),06H       ; ENTRY MODE SET
00F2  3E 32        LD  A,32H          ; DELAY 50 MSEC.
```

```
00F4  CD 0B 00          CALL DELAY
00F7  32 03 80          LD  (8003H),A       ; ENABLE LCD
00FA  21 DD 02          LD  HL,LCDBR        ; "<BENNY && ROSSY>" BY DEFAULT
                                            ; FIRST LINE
00FD  3E 10             LD  A,010H          ; 16 CHARACTERS
00FF  CD 68 00          CALL WRLINE
0102  21 01 80          LD  HL,8001H
0105  36 C0             LD  (HL),0C0H       ; GO TO SECOND LINE
0107  3E 32             LD  A,32H           ; DELAY 50 MSEC.
0109  CD 0B 00          CALL DELAY
010C  32 03 80          LD  (8003H),A       ; ENABLE LCD
010F  21 ED 02          LD  HL,LCDSI        ; "<>SYSTEMS INC.<>"
0112  3E 10             LD  A,10H           ; 16 CHARACTERS
0114  CD 68 00          CALL WRLINE
0117  06 28             LD  B,028H
0119  CD 3E 00          CALL ABDEL          ; DELAY 10 SEC.
011C  36 01             LD  (HL),01H        ; CLEAR DISPLAY
011E  3E 32             LD  A,32H           ; DELAY 50 MSEC.
0120  CD 0B 00          CALL DELAY
0123  32 03 80          LD  (8003H),A       ; ENABLE LCD
0126  C1                POP BC
0127  E1                POP HL
0128  C9                RET
0129                    ORG 0150H
0150  CD AA 00          CALL LCDINI
0153  CD 47 00          CALL INIKBD
0156  FB                EI
0157  ED 5E             IM  2
0159  76                HALT
015A                    ORG 0200H           ; KEYBOARD INTERRUPTION
0200  F3                DI
0201  F5                PUSH AF
0202  C5                PUSH BC
0203  D5                PUSH DE
0204  E5                PUSH HL
0205  21 04 40          LD  HL,4004H        ; RETRIEVE LCD FLAGS
0208  CB 46             BIT 0,(HL)
020A  28 02             JR  Z,FLKB
020C  20 76             JR  NZ,SLKB
020E  ED 4B 00 40 FLKB: BC,(4000H)          ; RETRIEVE NUM CHAR'S
0212  06 00             LD  B,00H           ; FROM MSB
0214  21 05 40          LD  HL,4005H
0217  09                ADD HL,BC
0218  3A 00 80          LD  A,(8000H)       ; RETRIEVE DATA FROM KEYBOARD
021B  32 02 80          LD  (8002H),A       ; CLEAR KEYBOARD INTERFACE
021E  FE 0D             CP  0DH             ; ENTER KEY?
0220  20 05             JR  NZ,FLBS
0222  CD 73 00          CALL EXITRD
0225  28 AD             JR  Z,RETURN        ; EXIT
0227  FE 08       FLBS:    CP  08H          ; BACK SPACE KEY?
0229  20 05             JR  NZ,FLDK
022B  CD 8C 00          CALL BACKSP         ; DEL LAST CHAR
022E  28 A4             JR  Z,RETURN
0230  FE 7F       FLDK: CP  7FH             ; DEL KEY?
0232  20 05             JR  NZ,FLSC
0234  CD 9E 00          CALL DELLN          ; DEL LINE FULL
0237  28 9B             JR  Z,RETURN
0239  77          FLSC: LD  (HL),A          ; SAVE CHAR
023A  CD 5A 00          CALL WRCHAR
023D  21 00 40          LD  HL,4000H
0240  34                INC (HL)
0241  21 01 40          LD  HL,4001H
0244  34                INC (HL)
```

```
0245  7E              LD  A,(HL)
0246  FE 10            CP  10H
0248  20 0D            JR  NZ,FLSL
024A  21 01 80         LD  HL,8001H
024D  36 07            LD  (HL),07H        ; SHIFT DISPLAY
024F  3E 05            LD  A,05H           ; DELAY 5 MSEC.
0251  CD 0B 00         CALL DELAY
0254  32 03 80         LD  (8003H),A       ; ENABLE LCD
0257  FE 28      FLSL: CP  28H             ; GO 2ND LINE
0259  20 79            JR  NZ,RETURN
025B  21 01 80         LD  HL,8001H
025E  36 02            LD  (HL),02H        ; GO HOME
0260  3E 05            LD  A,05H           ; DELAY 5 MSEC.
0262  CD 0B 00         CALL DELAY
0265  32 03 80         LD  (8003H),A       ; ENABLE LCD
0268  36 C0            LD  (HL),00C0H      ; GO SECOND LINE
026A  3E 05            LD  A,05H           ; DELAY 5 MSEC.
026C  CD 0B 00         CALL DELAY
026F  32 03 80         LD  (8003H),A       ; ENABLE LCD
0272  36 06            LD  (HL),06H        ; ENTRY MODE SET
0274  3E 05            LD  A,05H           ; DELAY 5 MSEC.
0276  CD 0B 00         CALL DELAY
0279  32 03 80         LD  (8003H),A       ; ENABLE LCD
027C  21 80 00         LD  HL,80H
027F  CB C6            SET 0,(HL)
0281  C3 D4 02         JP  RETURN
0284  ED 4B 02 40 SLKB: LD  BC,(4002H)
0288  06 00            LD  B,00H
028A  21 2D 40         LD  HL,402DH
028D  09               ADD HL,BC
028E  3A 00 80         LD  A,(8000H)
0291  32 02 80         LD  (8002H),A       ; CLEAR KEYBOARD INTERFACE
0294  FE 0D            CP  0DH             ; ENTER KEY?
0296  20 05            JR  NZ,SLBS         ; EXIT
0298  CD 73 00         CALL EXITRD
029B  28 37            JR  Z,RETURN
029D  FE 08      SLBS: CP  08H             ; BACK SPACE KEY?
029F  20 05            JR  NZ,SLDK         ; DEL LAST CHAR
02A1  CD 8C 00         CALL BACKSP
02A4  28 2E            JR  Z,RETURN
02A6  FE 50      SLDK: CP  50H             ; DEL KEY?
02A8  20 05            JR  NZ,SLSC         ; DEL LINE FULL
02AA  CD 9E 00         CALL DELLN
02AD  28 25            JR  Z,RETURN
02AF  77         SLSC: LD  (HL),A          ; SAVE CHAR
02B0  CD 5A 00         CALL WRCHAR
02B3  21 02 40         LD  HL,4002H
02B6  34               INC (HL)
02B7  21 03 40         LD  HL,4003H
02BA  34               INC (HL)
02BB  7E               LD  A,(HL)
02BC  FE 10            CP  10H
02BE  20 0D            JR  NZ,SLSL
02C0  21 01 80         LD  HL,8001H
02C3  36 07            LD  (HL),07H        ; SHIFT DISPLAY
02C5  3E 05            LD  A,05H           ; DELAY 5 MSEC.
02C7  CD 0B 00         CALL DELAY
02CA  32 03 80         LD  (8003H),A       ; ENABLE LCD
02CD  FE 28      SLSL: CP  28H
02CF  20 03            JR  NZ,RETURN
02D1  CD 8C 00         CALL BACKSP
02D4  E1         RETURN: POP HL
02D5  D1               POP DE
```

```
02D6    C1                  POP  BC
02D7    F1                  POP  AF
02D8    FB                  EI
02D9    ED 5E               IM   2
02DB    ED 4D               RETI
02DD    3C 42 45 4E  LCDBR: DB   "<","B","E","N"
02E1    4E 59 20 26         DB   "N","Y"," ","&"
02E5    26 20 52 4F         DB   "&"," ","R","O"
02E9    53 53 59 3E         DB   "S","S","Y",">"
02ED    3C 3E 53 59  LCDSI:    DB   "<",">","S","Y"
02F1    53 54 45 4D         DB   "S","T","E","M"
02F5    53 20 49 4E         DB   "S"," ","I","N"
02F9    43 2E 3C 3E         DB   "C",".","<",">"
```

Si el funcionamiento del programa anterior es el esperado, se tiene una microcomputadora activa y funcional. El siguiente paso que procede es la programación del sistema operativo que permita controlar los dispositivos periféricos que integran el sistema mínimo, así como los que puedan ser agregados.

5.9. Resumen

El proceso de diseño del Sistema Mínimo debe forzosamente dividirse en etapas, para realizar un proceso de verificación y corrección de los errores que vayan surgiendo. Así, es recomendable seguir la siguiente secuencia de diseño del Sistema Mínimo:

1) Diseño del "corazón" (reloj) del microprocesador.
2) Diseño del circuito de reset del microprocesador.
3) Colocación de los buffers requeridos por el microprocesador.
4) Selección de las señales que han de controlar las operaciones sobre la memoria (lectura/escritura).
5) Mapeo de la memoria del Sistema Mínimo para direccionar los diversos dispositivos periféricos conectados al Sistema Mínimo.
6) Conexión de la memoria de datos y de programa del Sistema Mínimo
7) Conexión de las interfaces para los dispositivos periféricos y para el reconocimiento de interrupciones

Recuerde que cada uno de estos pasos deben ser probados en forma estática mediante la aplicación de una señal conocida a la entrada del sistema y su correspondiente verificación a la salida del mismo.

Cuando haya verificado y corregido todas las etapas del diseño del Sistema Mínimo, diseñe un pequeño programa que permita interactuar todos los elementos del sistema (prueba dinámica), y si el resultado es el esperado, ya tiene en sus manos en pequeño microcomputador funcional y listo para ser ampliado con la adición de más dispositivos periféricos o la programación y mejora del sistema operativo para el Sistema Mínimo.

CAPITULO VI:
SISTEMA OPERATIVO

"No debería permitirse a nadie programar en ensamblador"

Jim Issak

6.1 Nociones Básicas de un Sistema Operativo

Se puede ver a un Sistema Operativo como una colección organizada de extensiones de software del hardware, el cual consiste esencialmente de rutinas de control que hacen funcionar e interactuar todos los elementos de un sistema computador, proporcionando además un entorno para la ejecución de programas. Estos programas se apoyan en las facilidades proporcionadas por el Sistema Operativo para tener acceso a los recursos del computador, tales como acceso a archivos, dispositivos de entrada/salida (E/S), etc. Estos recursos son accedidos mediante **llamadas al sistema operativo**[1]. Además de las llamadas al sistema operativo, los usuarios interactuan con el propio Sistema Operativo por medio de la ejecución de comandos u órdenes del sistema operativo. De esta forma, el sistema operativo proporciona la interfaz primaria entre los dispositivos de hardware del computador y el usuario.

La capacidad y velocidad de un Sistema Operativo depende en gran medida por las necesidades y características del entorno físico que está destinado a soportar.

De esta forma, el sistema operativo actúa como gestor de los recursos del sistema informático tales como el microprocesador, la memoria, los archivos y los dispositivos de entrada/salida. Al actuar de esta forma, el sistema operativo lleva una bitácora de cada uno de los recursos y elige al usuario al que se le proporcionará un recurso, así como el periodo de tiempo que tendrá asignado este recurso. En equipos de cómputo con ejecución concurrente de programas, el Sistema Operativo debe resolver las peticiones de recursos de manera que se mantenga la integridad del sistema, optimizando al máximo el rendimiento final de los recursos del equipo.

El papel fundamental de los Sistemas Operativos es incrementar la productividad de los recursos y de la capacidad de procesamiento del microprocesador fungiendo como una interfaz para el usuario, compartir el hardware entre varios usuarios, permitir a los mismos usuarios compartir datos entre ellos mismos, prevenir que un usuario interfiera con otro, planear los recursos del usuario, facilitar las operaciones de entrada/salida, llevar una bitácora de los recursos asignados a los usuarios, recuperar información errónea, facilitar las operaciones en paralelo, organizar los datos para una mayor seguridad y una mayor velocidad de acceso, manejar comunicaciones en red, etc.

6.2 Funciones de un Sistema Operativo

Los sistemas operativos controlan básicamente dos tipos de funciones, las cuales se dividen en:

1) **Funciones del Usuario**: Éstas se encargan de proveer al usuario un entorno para la ejecución de programas. Dentro de este rubro se pueden encontrar las siguientes clases de funciones
   a) **Control del Programa**: Le permiten al usuario manipular programas y datos en un sistema de cómputo.
   b) **Operaciones de Entrada/Salida**: Debido a la complejidad de comunicación entre el microprocesador y los **Puertos de Entrada/Salida**, el sistema operativo debe proveer funciones de entrada/salida de datos que sean simples, potentes y amigables con el usuario.
   c) **Manipulación de Sistema de Archivos**: El sistema de archivos normalmente es usado para almacenar grandes cantidades de datos y programas, de esta forma, el sistema operativo permite al usuario acceder y manipular los datos almacenados mediante nombres simbólicos más que por direcciones físicas de los medios de almacenamiento.
2) **Funciones del Sistema**: Éstas tienen como función manejar los recursos internos del sistema, tales como la memoria, los **Puertos de Entrada/Salida**, las interrupciones, etc. Dentro de este rubro se pueden encontrar las siguientes subclasificacioes:
   a) **Manejo de Memoria**: Éste permite la manipulación de áreas de memoria superiores a las direcciones físicas del **Bus de Direcciones** del microprocesador.
   b) **Protección**: El sistema operativo debe proveer un nivel de privacidad y no-interferencia cuando exista comunicación con diversos dispositivos periféricos al mismo tiempo, controlando la interacción entre los programas del usuario y los programas del mismo sistema operativo.
   c) **Planificación y Asignación de Recursos**: Éste permite a varios programas que se estén ejecutando simultáneamente no tener conflictos, asignando recursos a cada uno de ellos, tratando lo mayor posible de optimizar los recursos asignados.
   d) **Estadísticas**: El sistema operativo debe llevar una bitácora de los recursos asignados a cada programa y asignar nuevos recursos dependiendo de las bitácoras.

Los Sistemas Operativos pueden procesar los datos en forma serial o concurrente; esto es, los recursos del sistema pueden ser asignados a un solo programa y ser liberados hasta que éste haya terminado o pueden ser reasignados en forma dinámica entre diferentes programas en diferentes etapas de su ejecución. A los Sistemas Operativos que pueden ejecutar múltiples programas en forma intercalada se les denominan sistemas de multiprogramación, los cuales se definen en forma breve a continuación.

[1] Tipo de interrupción de software, el cual permite a un programa realizar un proceso sobre un dispositivo de hardware sin una manipulación directa, esta manipulación es realizada por el sistema operativo y devuelve los resultados al proceso que generó la interrupción.

1) **Batch Serial**: Fueron los primeros sistemas operativos de propósito general, diseñados para planificar las entradas y salidas del sistema. Los programas eran leídos desde una lectora de tarjetas perforadas, ejecutadas, y posteriormente las salidas eran escritas a impresoras o dispositivos análogos. Estas operaciones eran repetidas para cada programa, posteriores mejoras permitieron la ejecución de varios programas habilitando las funciones de lectura de programas desde lectoras de tarjetas y escritura de datos en forma concurrente.

   Los Sistemas Operativos por lotes precisan que el programa, los datos y las órdenes adecuadas al sistema sean introducidos al computador en forma de trabajo. Esto permiten poca o ninguna interacción con los usuarios y los programas en ejecución.

   Los trabajos asignados al Sistema Operativo son procesados en su orden de llegada, esto es, el primero en llegar es el primero en ejecutarse.

   Normalmente el área de memoria se divide en dos: una de ellas está permanentemente ocupada por la parte residente del Sistema Operativo y la otra es usada para cargar los programas transitorios durante su ejecución, al terminarse la ejecución del programa transitorio, un nuevo programa es cargado en la misma área de memoria.

   Al existir un solo programa en ejecución en la memoria, no se requiere de alguna gestión crítica de dispositivos de entrada/salida, permitiendo utilizar éstos dispositivos mediante programa.

2) **Multiprogramación y Multitarea**: El funcionamiento de ambos tipos se basa en lo siguiente: cuando la **CPU** tiene unas operaciones de entrada/salida, éstos switchean a un programa diferente, el cual maneja estas operaciones de entrada/salida, de esta forma se incrementa el rendimiento del sistema, dado que el cuerpo principal del programa puede quedar residente en memoria y listo para efectuar llamadas a subprogramas.

   Los Sistemas Operativos Multitarea ejecutan instancias de programas (denominados procesos o tareas), de ésta forma pueden ejecutar en forma concurrente dos o más procesos activos. Estos Sistemas Operativos se implementan manteniendo el código y los datos de varios procesos simultáneamente en memoria y multiplexando el microprocesador y los Dispositivos de Entrada/Salida entre ellos. Existe en éstos tipos de Sistema Operativo un soporte de hardware y software para proteger la memoria con el fin de evitar que procesos erróneos corrompan la memoria y el comportamiento de otros procesos.

   Los Sistemas Operativos Multiprogramación además de soportar Multitarea, proporcionan un sistema de protección de memoria, fuerzan el control de la concurrencia cuando los procesos acceden a dispositivos de entrada/salida y archivos compartidos. De igual forma, soportan generalmente varios usuarios (adquiriendo el término de Sistema Operativo Multiusuario), éstos proporcionan facilidades para mantenimiento del entorno de los usuarios individuales, requieren de validación de cada usuario para asegurar la protección del sistema y proporcionan una contabilidad de uso de cada uno de los recursos asignados por usuario.

Dentro de este rubro también se ubican los sistemas Operativos Multiacceso, los cuales permiten el acceso simultaneo a un sistema informático desde varias terminales tales como los sistemas de reservación aérea que pueden soportar centenares de terminales activos bajo el control de un programa central.

Otro tipo de Sistema Operativo que se puede ubicar dentro de este grupo es el denominado Multiprocesamiento o Multiprocesador, los cuales gestionan la operación de sistemas informáticos que incorporan varios microprocesadores. Por definición, éstos se ubican dentro del grupo de Multitarea ya que soportan la ejecución simultánea de múltiples tareas (procesos) sobre diferentes procesadores.

3) **Tiempo compartido**: Éstos son representantes populares de los Sistemas Operativos Multiprogramados, Multiusarios. En este tipo de Sistemas Operativos, un usuario puede interactuar con un programa por medio de una terminal. El sistema operativo asigna periodos cortos de tiempo a cada usuario, dando la sensación de que el usuario es el único que se encuentra en sesión en el sistema. El Sistema Operativo intenta lograr una compartición equitativa de los recursos comunes. La mayoría de los Sistemas Operativos de Tiempo Compartido utilizan una planificación por reparto (circular) del tiempo, según el cual los programas se ejecutan con prioridad rotativa que se incrementa durante la espera y disminuye después de que se les ha concedido el servicio, con el fin de evitar que los programas monopolicen el procesador. Un programa que se ejecute más tiempo que el intervalo definido por el sistema operativo se interrumpe y se coloca al final de la cola de programas en espera. La gestión de memoria para este tipo de sistema operativo proporciona aislamiento y protección a los programas residentes en memoria, además de requerir una sofisticación en la gestión de Entrada/Salida de datos para tratar con múltiples usuarios y dispositivos, de igual forma, debe proporcionar protección y control de acceso hacia archivos.

4) **Tiempo real**: Este tipo de Sistemas Operativos se usa en computadores donde los datos deben de ser aceptados y procesados en un gran numero de procesos, la mayoría de los cuales son externos al sistema, en un intervalo de tiempo corto o en plazos breves. Ejemplos de tales aplicaciones pueden ser por ejemplo: control industrial, conmutación telefónica, control de vuelo y simulaciones de procesos en tiempo real. Estos sistemas operativos también son usados con gran éxito en aplicaciones militares.

El objetivo primordial de los Sistemas Operativos en Tiempo Real es proporcionar tiempos de respuesta rápidos, satisfaciendo los plazos de planificación. Aparecen procesos explícitos definidos y controlados por el programador, cada uno de estos procesos está a cargo del manejo de un único suceso externo. El proceso se activa tras la ocurrencia del suceso en cuestión, el cual viene frecuentemente señalado por una interrupción. La operación de Multitarea se consigue planificando los procesos para ser ejecutados independientemente unos de otros. Cada proceso tiene asignado un nivel de prioridad que se corresponde con la

importancia del suceso al que sirve. Los procesos de mayor prioridad expropian la ejecución de los procesos de menor prioridad (Planificación Basada en Prioridades).

En Sistemas Operativos en Tiempo Real, la gestión de memoria es menor que en otros tipos de Sistemas Operativos de Multiprogramación, ya que muchos procesos residen permanentemente en memoria con el fin de lograr tiempos de respuesta rápidos. La gestión de archivos en sistemas operativos de tiempo real tiene como fin incrementar la velocidad de acceso antes que la utilización eficiente en medios magnéticos.

5) **Combinado**: Normalmente, estos sistemas operativos son comerciales y proporcionan una combinación de los tipos de Sistemas Operativos descritos. Por ejemplo, en centros universitarios es frecuente encontrar tanto desarrollo interactivo de programas como simulación de eventos naturales.
6) **Distribuido**: Éste se compone de una colección de sistemas informáticos autónomos capaces de existir comunicación mediante interconexiones de hardware y software. Éstos evolucionaron a partir de las redes de computadoras en las que un número muy grande de equipos independientes están conectados mediante enlaces y protocolos de comunicación.

Los sistemas Operativos Distribuidos gobiernan la operación de un sistema informático distribuido y proporciona una abstracción de máquina en forma virtual en los usuarios, su objetivo esencial es la transparencia de datos. Proporcionan medios para la compartición global de los recursos del sistema, tales como la capacidad de cálculo, los archivos y los dispositivos de entrada/salida. Además de los servicios típicos, los Sistemas Operativos Distribuidos proporcionan el acceso a recursos remotos, comunicación remota de procesos y la distribución de los cálculos.

6.2 Definición de Funciones del Sistema Operativo para el Sistema Mínimo

Como se ha mencionado en la sección anterior, la función de cualquier sistema operativo es la de proporcionar a los usuarios un conjunto de herramientas para ayudarle a desarrollar, depurar, ejecutar y hacer un buen uso de un programa de computadora. Para ello, el sistema operativo gestiona los recursos de la computadora creando una interfaz entre el usuario y los recursos físicos de la misma computadora. Dependiendo de los recursos con los que cuenta la computadora, así como los fines para las que fue creada, en esa misma medida será la complejidad del sistema operativo. Por ejemplo, para sistemas pequeños como el mostrado en el presente trabajo, sólo proporciona un medio para introducir y leer datos de 8 bits desde los dispositivos de memoria y desde un único dispositivo periférico; para sistemas grandes y complejos, el sistema operativo suele controlar en forma dinámica los recursos del sistema.

Para poder operar en forma óptima los recursos del sistema mínimo del presente trabajo, se debe considerar los siguientes elementos del sistema mínimo:

1) Microprocesador Z80A.
2) 16384 octetos (localidades) de memoria **EPROM** (16Kb).
3) 8192 octetos (localidades) de memoria **RAM** (8 Kb)
4) Teclado estándar AT/XT e interfaz.
5) **Pantalla de Cristal Liquido (LCD)** e interfaz.

El sistema operativo del sistema mínimo debe proporcionar acceso a estos recursos y permitir al usuario un medio de controlarlos durante la ejecución de los programas. De esta forma, el sistema operativo del sistema mínimo incluye los siguientes medios y funciones de control:

1) Arranque en frío.
2) Reconocimiento de interrupciones.
3) Lectura de datos/instrucciones del teclado
4) Visualizador de datos e instrucciones (comandos)
5) Reconocimiento de datos/instrucciones
6) Ejecución de comandos

Las funciones y modos de operar de cada una de ellas serán explicadas con mayor profundidad en las siguientes secciones.

6.2.1 Arranque en Frío

La operación de arranque en frío consiste en la disposición del sistema operativo al momento de la aplicación de potencia al sistema mínimo. Esto puede efectuarse mediante el almacenamiento en algún medio con capacidad de almacenamiento de una pequeña rutina de autoiniciación ("bootstrap"). Esta rutina es utilizada para cargar el sistema operativo desde otro dispositivo tal como una unidad de disquete (floppy) u otro medio (el cual puede ser un dispositivo de almacenamiento magnético de mayor capacidad y mayor velocidad que este último, tal como un disco duro). Para el caso del sistema mínimo, el sistema operativo se encuentra alojado en forma permanente en varios chips de memoria **EPROM**, siempre en estado de espera para ser ejecutado inmediatamente después de la aplicación de energía y se active el botón de **Reset**. Esta última acción permite poner a cero el contenido del registro **PC** del microprocesador Z80A.

Después de haberse efectuado la operación de **Reset**, el microprocesador Z80A comienza la ejecución de la instrucción situada en la localidad **0000H** de memoria. El sistema operativo del sistema mínimo proporciona las instrucciones para comenzar la ejecución del sistema operativo. Esta serie particular de instrucciones de programa constituye el procedimiento de arranque en frío y establece las condiciones requeridas de puesta a punto para el sistema operativo. De esta forma, se inicia el registro **SP** a una zona de la memoria de lectura/escritura para mantener la pila **LIFO (Last Input/First Output)**. Este pila es requerida para la ejecución de cualquiera de las instrucciones **Restart** y **Call**, además de las instrucciones de reconocimiento de interrupciones. La

falta de iniciación del registro **SP** antes de la ejecución de estas instrucciones es impredecible y en ocasiones desastrosa para el correcto funcionamiento del sistema mínimo. Para el sistema operativo del sistema mínimo descrito en el presente trabajo, el puntero de pila se ubica en la localidad de memoria de lectura/escritura **5FFEH**.

6.2.1.1 Iniciación del Sistema Operativo del Sistema Mínimo

El proceso de iniciación del sistema operativo comprende los pasos de definir el área de **memoria RAM** a partir de la cual se van a alojar los datos de la **Pila LIFO**, como se mencionó anteriormente, el área asignada en el sistema para este propósito se ubica a partir de la localidad 5FFEH (localidad superior de la memoria **RAM**), recuerde sin embargo, que a diferencia del concepto de **Pila** universalmente conocida (la **Pila** se incrementa conforme se le van añadiendo datos), en nuestro sistema la **Pila LIFO** decrece (esto indica que cuando se introduce un dato en la Pila, el puntero de pila o **registro SP** se decrementa y apunta a la localidad de memoria anterior, en tanto que cuando se elimina un dato de la pila, el **registro SP** se incrementa y apunta a la siguiente localidad de memoria). Debe tomar esto en consideración cuando se diseñen programas a verificar, ya que si se usan en forma indiscriminada los accesos a Pila, podría existir conflictos entre los programas del usuario y el contenido del **registro SP**.

El siguiente paso que procede es la iniciación de la **Pantalla de Cristal Líquido**, con el objeto de crear un dispositivo periférico de salida visual, ello permite mostrar las operaciones internas del sistema, así como una forma de visualización de los resultados de los cálculos efectuados por el mismo sistema.

A partir de la figura 4.27, el diagrama de flujo para la secuencia de iniciación de **la Pantalla de Cristal Líquido** se muestra en la siguiente figura:

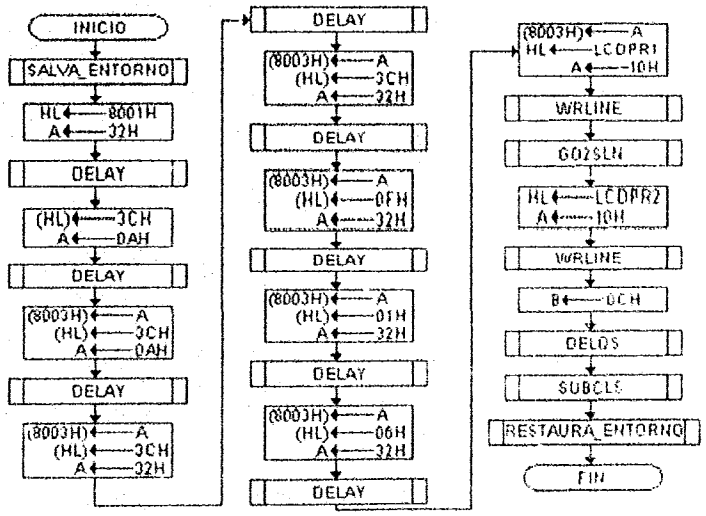


Fig. 6.1 SECUENCIA DE INICIACIÓN PARA LA PANTALLA LCD

Habiendo efectuado el proceso de iniciación de la Pantalla de Cristal Liquido, se debe corroborar el óptimo funcionamiento de los dispositivos de memoria RAM, ya que van a jugar un papel fundamental en el funcionamiento del sistema.

Antes de proceder a describir el proceso de verificación de la memoria RAM, es menester describir la subrutina que verifica áreas especificas de memoria. Esta subrutina tiene un funcionamiento como el mostrado en el siguiente diagrama de flujo:

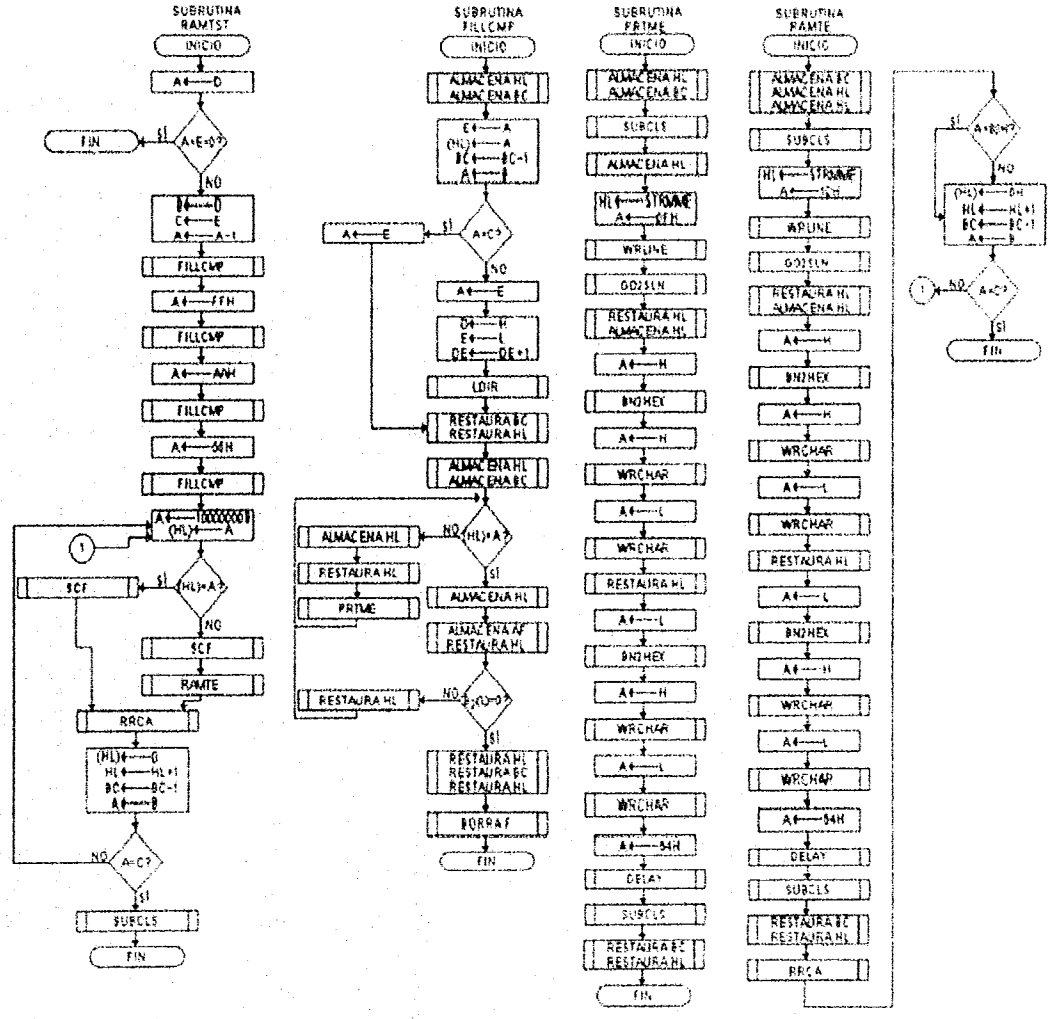


Fig. 6.2 SUBRUTINA DE VERIFICACIÓN DE LOCALIDADES DE MEMORIA RAM

Después de haber mostrado como funciona la subrutina de verificación de áreas de memoria RAM, el paso que procede es la descripción del proceso de verificación de la memoria RAM total. Como se había mencionado con anterioridad, el sistema mínimo utiliza dispositivos de memoria RAM 6116, los cuales se encuentran internamente construidos para almacenar 2Kb octetos de datos. De igual forma, el sistema tiene un banco de memoria total de 16 Kb, deduciéndose que el

área de memoria RAM está compuesta por cuatro bloques consecutivos, ello permite la verificación individual de cada bloque de memoria RAM. Para realizar este proceso, el punteo de pila (registro SP) se va modificando para ocupar localidades de memoria que no estén siendo verificadas, esto último se podrá apreciar mejor en la gráfica número 6.4 (recuerde que la manipulación del puntero de PILA no afecta su definición, puesto que al terminar el proceso de verificación del área de memoria RAM, se restablece su área de memoria inicial).

Finalmente, el último paso que procede en la iniciación del sistema es la definición del área de memoria RAM en la cual se van a almacenar los datos que se introduzcan por teclado. Para ello, el siguiente cuadro muestra las definiciones de estas áreas de memoria para los datos que se introduzcan por el teclado:

Localidad de Memoria	Función
04000H	Almacenar el número de caracteres de la primera línea
04001H	Almacenar la posición del cursor de la primer línea
04002H	Almacenar el número de caracteres de la segunda línea
04003H	Almacenar la posición del cursor de la segunda línea
04004H	Flag de estados de la pantalla LCD
04005H	Buffer de caracteres de la primer línea
0402DH	Buffer de caracteres de la segunda línea

Tabla 6.1 DEFINICIÓN DE ÁREAS DE MEMORIA PARA CARACTERES DEL TECLADO

Habiendo definido las áreas de memoria RAM en la cual se van a almacenar los datos procedentes del teclado, el siguiente paso es la iniciación de estas áreas de memoria a valores conocidos, de tal forma que no se produzcan errores durante la ejecución de las instrucciones, tal como se puede apreciar en la siguiente figura:

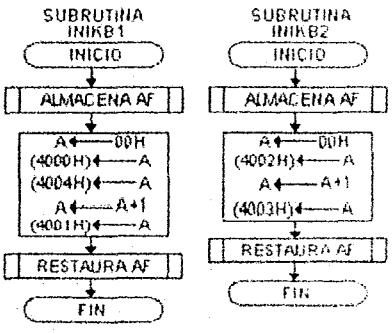


Fig. 6.3 SUBRUTINAS DE INICIACIÓN DEL TECLADO

Para este caso, se implementaron dos subrutinas de iniciación de variables del teclado, puesto que la primer subrutina es usada en el modo de ejecución de comandos y la segunda subrutina es usada en el modo de programación del sistema.

El proceso completo de iniciación del sistema operativo del sistema mínimo se puede apreciar en la siguiente figura:

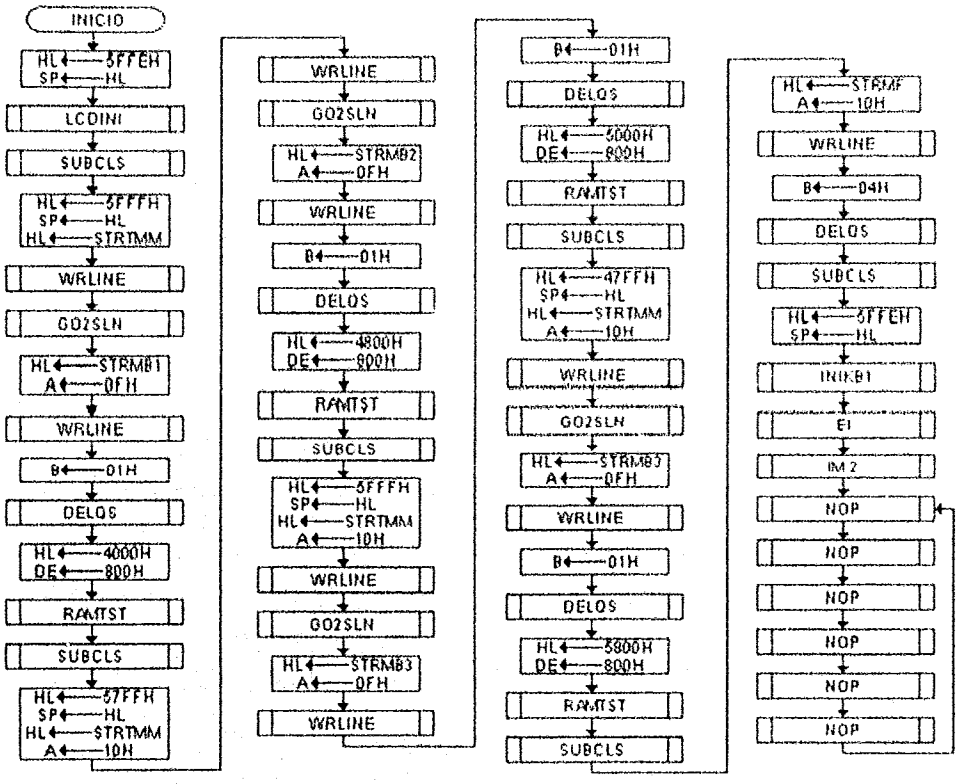


Fig. 6.4 SECUENCIA DE INICIACIÓN PARA EL SISTEMA OPERATIVO DEL SISTEMA MÍNIMO

Preste atención especial a las instrucciones de habilitación de interrupción y habilitación del modo de interrupción, ya que éstas son fundamentales para permitir la introducción de datos procedentes del teclado.

6.2.2 Reconocimiento de Interrupciones de Hardware

Como se había mencionado anteriormente, para poder realizar las instrucciones en forma apropiada para un evento de interrupción desde un dispositivo de hardware se requiere como pasos antecesores los siguientes puntos:

1) Diseño de un circuito para reconocer las interrupciones de hardware.
2) Diseño de un circuito para introducir la dirección de brinco para el hardware interruptor.
3) Iniciación del registro **Puntero de Pila (SP)** a una localidad de memona de lectura escritura (**RAM**).
4) Activación de los flip-flops de interrupción (si fuese necesario).
5) Programación de la subrutina de atención al dispositivo interruptor.
6) Almacenamiento de las variables de estado actuales del programa

7) Programación de las instrucciones para efectuar un brinco a la localidad de memoria del punto anterior y ejecución de las instrucciones subsecuentes.
8) Recuperación de las variables de estado del programa principal.
9) Retorno del control al programa principal.

Para poder programar con éxito los puntos anteriores en el sistema operativo del sistema mínimo, el puntero de pila se inicia a la localidad de memoria **5FFEH** mencionada anteriormente. Además, dado que se implementará el modo de interrupción 2 (**Interrupciones Vectorizadas**), el **Vector de Interrupciones** se programa a partir de la localidad de memoria EPROM **0EF0H** (bloque número 1 de memoria **EPROM**), lo cual nos deja margen para programar los subsecuentes 127 vectores de interrupción que es capaz de controlar el microprocesador Z80. En esta localidad de memoria se programa la dirección destino del programa que proporciona servicios al dispositivo periférico (**0B00H**), recuerde que el byte de menor peso de la dirección destino se programa primero, así que en la localidad de memoria **0EF0H** se coloca el byte de menor peso de la dirección mencionada (**00H**) y en la dirección siguiente (**0EF1H**) se coloca el byte de mayor peso (**0BH**). Gracias a lo mencionado anteriormente, al momento de efectuarse una interrupción y ser reconocida ésta por el microprocesador, el dispositivo periférico coloca en el **Bus de Datos** el byte **F0H**, el cual se une con el contenido del registro **I** (**0EH**) y forma la dirección **0EF0H**. Esta última se carga en el registro **PC** (antes de ello, el contenido del registro **PC** incrementado se almacena en la pila de memoria externa). El dato que se encuentre en esta localidad de memoria y en la siguiente localidad se cargan en el registro **PC**, el cual efectúa de esta forma un brinco incondicional a la nueva localidad del registro **PC**. Hecho lo anterior, se procede a la ejecución de las instrucciones localizadas a partir de la dirección actual de memoria hasta que ocurra la ejecución explícita de una instrucción **RET**.

6.2.3 Lectura de Datos/Instrucciones del Teclado

El método de lectura de datos e instrucciones del teclado proporciona la interfaz primaria entre el sistema mínimo y el usuario. A la entrada, comienza a leer datos desde el puerto del teclado a partir de una señal de interrupción que éste último envía (en cada petición de interrupción, es leída una tecla del teclado).

El proceso de introducción de datos procedentes del teclado mediante interrupción se puede apreciar en la siguiente figura:

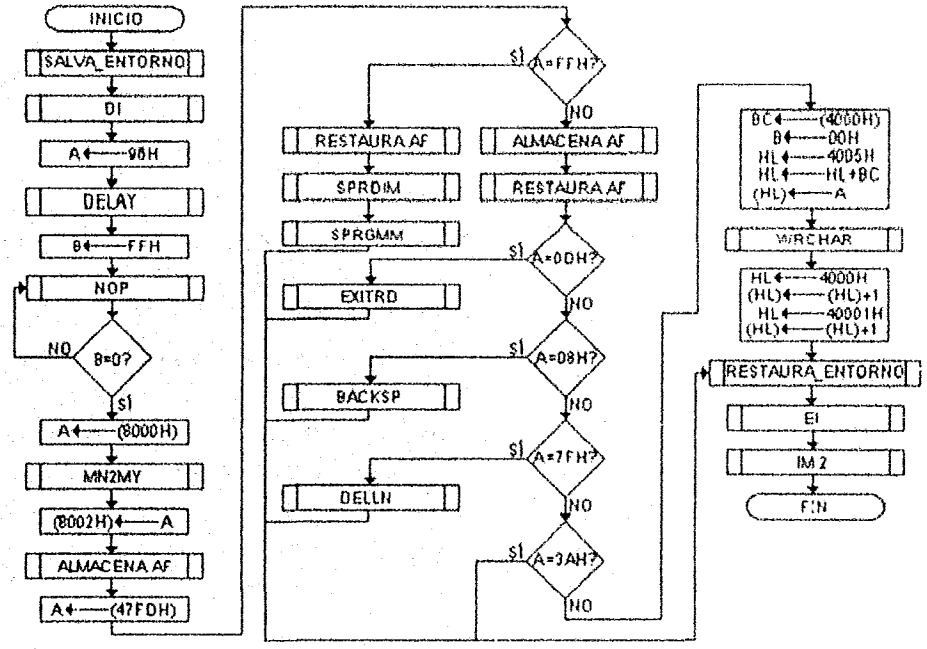


Fig. 6.5 SECUENCIA DE RECEPCIÓN DE DATOS PROCEDENTES DEL TECLADO

Preste especial atención a los siguientes puntos:

1) Almacenamiento de las variables de ambiente en la pila de memoria externa.
2) Deshabilitación de los servicios de interrupción: Esto permite procesar los datos y no permitir ninguna dato adicional hasta que se haya procesado el actual.
3) Ciclo corto de retardo: Esto permite estabilizar los datos procedentes del teclado antes de aceptar su inserción y manipulación por parte del sistema.
4) Verificación de estado de programación: Esto permite saltar a la subrutina que se encarga de aceptar los datos del programa del usuario para su posterior ejecución (si el usuario quisiera introducir datos desde el teclado en su programa, se podría lograr esto con unas ligeras modificaciones en esta parte del sistema).

5) Recuperación de las variables de ambiente almacenadas en la pila de memoria externa: Esto permite continuar la ejecución del programa desde la cual fue llamada la subrutina de interrupción.
6) Re-habilitacion de los servicios de interrupción y del modo de interrupción: Esto nos permite seguir aceptando datos procedentes del teclado mediante interrupciones.

6.2.4 Visualización de Datos/Instrucciones

Este módulo junto con la de lectura de datos/instrucciones del teclado y de reconocimiento de interrupciones de hardware constituyen el eje central de la interfaz entre el microprocesador y el usuario.

Para poder determinar el estado de la **Pantalla de Cristal Liquido**, así como la ubicación del cursor en la pantalla y otras cuestiones más se utilizará una localidad de memoria (**4004H**), la cual fungirá como medio de almacenamiento de los estados de la **Pantalla de Cristal Liquido**. La configuración de dicho **Flag de Estados**, así como la función de cada uno de los bits se muestra en la siguiente tabla:

Bit	Flag	Descripción	Acción
0	NL	Número de línea	Almacena la ubicación del cursor en la **Pantalla LCD**. Si está inactivo, esta se encuentra en la primer línea, en caso contrario se ubica en la segunda línea
1	ND	Indefinido	
2	ND	Indefinido	
3	ND	Indefinido	
4	ND	Indefinido	
5	ND	Indefinido	
6	ND	Indefinido	
7	ND	Indefinido	

Tabla 6.1 ESTRUCTURA DE LA BANDERA DE ESTADOS (FLAG) DE LA PANTALLA DE CRISTAL LÍQUIDO

El módulo de visualización de datos/instrucciones opera de la siguiente forma:

1. El dato o instrucción es cargado en el buffer tipo latch de la **Pantalla de Cristal Liquido** mediante una instrucción **LD (0C00H), A**.
2. El microprocesador ejecuta una serie programada de retardos con el fin de permitir al controlador de **Pantalla LCD** terminar su procesamiento actual.
3. Al final de este periodo de retardo, la **Pantalla de Cristal Liquido** es activada mediante una instrucción **LD (8003H), A**.

6.2.5 Reconocimiento de Datos/Instrucciones

Este módulo en su parte más esencial se compone de una subrutina de comparación de cadena, el cual compara la cadena introducida por el usuario desde el teclado.

Como se puede apreciar en la figura número 6.5, cuando se reconoce una tecla **ENTER** procedente del teclado el programa bifurca hacia una subrutina, la cual se encarga de verificar los datos/instrucciones del usuario contra una base de datos almacenada, la cual le permite determinar si es una instrucción o un dato.

La subrutina de verificación de datos/instrucciones opera como se muestra en la siguiente figura:

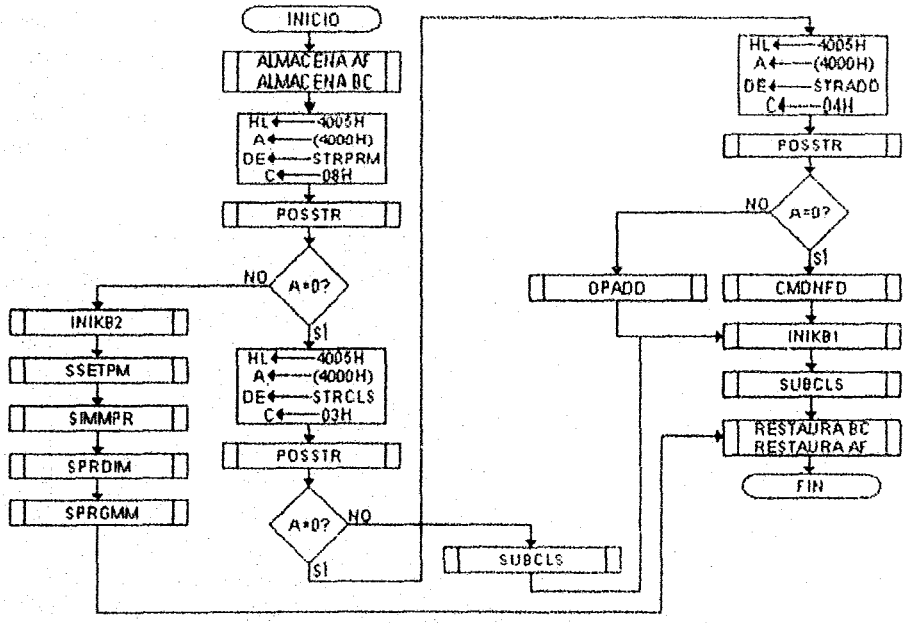


Fig. 6.6 SECUENCIA DE RECONOCIMIENTO DE COMANDOS/INSTRUCCIONES DEL USUARIO

6.2.6 Ejecución de Comandos

Habiendo realizado con éxito el reconocimiento de datos/instrucciones del módulo anterior, el siguiente paso que procede es la ejecución del comando a partir de los datos alojados en la memoria **RAM**.

Para ello, el sistema cuenta con las siguientes subrutinas (las cuales pueden ser usadas por los usuarios para verificar sus programas introducidos mediante el teclado):

Nombre:	Función:			
DELAY	Efectúa retardos programados por el usuario, los retardos se efectúan en milésimas de segundo			
Argumento(s):				
Registro A	Almacena el número de milésimas de segundos de retardo			
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL DELAY	CD	07	00
Nota(s):				
Registro A	La programación de 0 equivale a la programación de 255: A=0=255 ms de retardo			
Ejemplo:	Mnemónico	Código Hexadecimal		
	LD A, 64H	3E	64	
	CALL DELAY	CD	07	00

Nombre:	Función:			
DELQS	Efectúa retardos programados por el usuario, los retardos se efectúan a razón de 250 (0.25 ms) milésimas de segundo			
Argumento(s):				
Registro B	Almacena el número de 0.25 milésimas de segundos de retardo			
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL DELQS	CD	3A	00
Nota(s):				
Ninguno				
Ejemplo:	Mnemónico	Código Hexadecimal		
	LD B, 04H	06	04	
	CALL DELQS	CD	3A	00

Nombre:	Función:			
WRCHAR	Imprime un carácter en la pantalla de cristal liquido			
Argumento(s):				
Registro A	Almacena el carácter a imprimir en la pantalla de cristal liquido			
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL WRCHAR	CD	44	00
Nota(s):				
Ninguno				
Ejemplo:	Mnemónico	Código Hexadecimal		
	LD A, 44H	3E	44	
	CALL WRCHAR	CD	44	00

Nombre:	Función			
WRLINE	Imprime una línea de caracteres en la pantalla de cristal líquido			
Argumento(s):				
Registro par HL	Almacena la dirección de inicio de los caracteres a imprimir en la pantalla de cristal líquido			
Registro A	Almacena el número de caracteres a imprimir			
Resultado(s)				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL WRLINE	CD	54	00
Nota(s):				
	Aún cuando se puede programar hasta la impresión de 255 caracteres, la pantalla de cristal líquido puede mostrar un máximo de 16 caracteres por línea			
Ejemplo:	Mnemónico	Código Hexadecimal		
	LD HL, STRDTI	21	C0	0C
	LD A, 08H	3E	08	
	CALL WRLINE	CD	54	00

Nombre:	Función:			
GO2SLN	Mueve el cursor a la segunda línea de la pantalla de cristal líquido			
Argumento(s):				
Ninguno				
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL GO2SLN	CD	62	00
Nota(s):				
Ninguno				
Ejemplo:	Mnemónico	Código Hexadecimal		
	CALL GO2SLN	CD	62	C0

Nombre:	Función:			
SUBCLS	Borra los caracteres impresos en la pantalla de cristal líquido y coloca el cursor en la posición más a la izquierda de la pantalla			
Argumento(s):				
Ninguno				
Resultado(s)				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL SUBCLS	CD	72	00
Nota(s)				
Ninguno				
Ejemplo:	Mnemónico	Código Hexadecimal		
	CALL SUBCLS	CD	72	00

Nombre:	Función			
WRLINE	Imprime una línea de caracteres en la pantalla de cristal líquido			
Argumento(s):				
Registro par HL	Almacena la dirección de inicio de los caracteres a imprimir en la pantalla de cristal líquido			
Registro A	Almacena el número de caracteres a imprimir			
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL WRLINE	CD	54	00
Nota(s):				
	Aún cuando se puede programar hasta la impresión de 255 caracteres, la pantalla de cristal líquido puede mostrar un máximo de 16 caracteres por línea			
Ejemplo:	Mnemónico	Código Hexadecimal		
	LD HL, STRDTI	21	C0	0C
	LD A, 08H	3E	08	
	CALL WRLINE	CD	54	00

Nombre:	Función:			
GO2SLN	Mueve el cursor a la segunda línea de la pantalla de cristal líquido			
Argumento(s):				
Ninguno				
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL GO2SLN	CD	62	00
Nota(s):				
Ninguno				
Ejemplo:	Mnemónico	Código Hexadecimal		
	CALL GO2SLN	CD	62	00

Nombre:	Función:			
SUBCLS	Borra los caracteres impresos en la pantalla de cristal líquido y coloca el cursor en la posición más a la izquierda de la pantalla			
Argumento(s):				
Ninguno				
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL SUBCLS	CD	72	00
Nota(s):				
Ninguno				
Ejemplo:	Mnemónico	Código Hexadecimal		
	CALL SUBCLS	CD	72	00

Nombre:	Función:			
WRLINE	Imprime una línea de caracteres en la pantalla de cristal líquido			
Argumento(s):				
Registro par HL	Almacena la dirección de inicio de los caracteres a imprimir en la pantalla de cristal líquido			
Registro A	Almacena el número de caracteres a imprimir			
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL WRLINE	CD	54	00
Nota(s):				
	Aún cuando se puede programar hasta la impresión de 255 caracteres, la pantalla de cristal líquido puede mostrar un máximo de 16 caracteres por línea			
Ejemplo:	Mnemónico	Código Hexadecimal		
	LD HL, STRDTI	21	C0	0C
	LD A, 08H	3E	08	
	CALL WRLINE	CD	54	00

Nombre:	Función:			
GO2SLN	Mueve el cursor a la segunda línea de la pantalla de cristal líquido			
Argumento(s):				
Ninguno				
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL GO2SLN	CD	62	00
Nota(s):				
Ninguno				
Ejemplo:	Mnemónico	Código Hexadecimal		
	CALL GO2SLN	CD	62	00

Nombre:	Función:			
SUBCLS	Borra los caracteres impresos en la pantalla de cristal líquido y coloca el cursor en la posición más a la izquierda de la pantalla			
Argumento(s):				
Ninguno				
Resultado(s):				
Ninguno				
Forma de llamada:	Mnemónico	Código Hexadecimal		
	CALL SUBCLS	CD	72	00
Nota(s):				
Ninguno				
Ejemplo:	Mnemónico	Código Hexadecimal		
	CALL SUBCLS	CD	72	00

Nombre	Función			
BN2HEX	Convierte un patrón de 8 bits binarios a dos caracteres hexadecimales			
Argumento(s)				
Registro A	Patrón binario a convertir en formato hexadecimal			
Resultado(s)				
Registro H	Valor más significativo del conjunto de caracteres hexadecimales resultantes			
Registro L	Valor menos significativo del conjunto de caracteres hexadecimales resultantes			
Forma de llamada	Mnemónico	Código Hexadecimal		
	CALL SUBCLS	CD	8D	00
Nota(s)				
Ninguno				
Ejemplo	Mnemónico	Código Hexadecimal		
	LD A, 0CFH	3E	CF	
	CALL BN2HEX	CD	8D	00

Nombre	Función			
HEX2BN	Convierte un patrón de datos hexadecimales en un patrón de datos BCD (Binario Codificado en Decimal)			
Argumento(s)				
Registro HL	Patrón de datos hexadecimales			
Resultado(s)				
Registro A	Patrón de datos BCD resultante			
Forma de llamada	Mnemónico	Código Hexadecimal		
	CALL HEX2BN	CD	B0	07
Nota(s)				
Ninguno				
Ejemplo	Mnemónico	Código Hexadecimal		
	LD HL, 0F6H	21	F6	00
	CALL HEX2BN	CD	B0	07

Nombre	Función			
MN2MY	Convierte un carácter en minúscula a un carácter en mayúscula			
Argumento(s)				
Registro A	Almacena el carácter en minúscula			
Resultado(s)				
Registro A	Almacena el carácter en mayúscula			
Forma de llamada	Mnemónico	Código Hexadecimal		
	CALL SUBCLS	CD	A2	00
Nota(s)				
Ninguno				
Ejemplo	Mnemónico	Código Hexadecimal		
	LD A, 64H	3E	64	
	CALL MN2MY	CD	A2	00

Nombre	Función			
BN2DEC	Convierte una serie de dígitos binarios a decimales			
Argumento(s)				
Registro DE	Almacena el patrón de bits binarios a convertir en hexadecimal			
Registro HL	Almacena la dirección de memoria en la cual se va a alojar la longitud del resultado			
Resultado(s)				
Localidad de memoria 40A0h	Dirección de memoria donde inicia el patrón de caracteres decimales resultantes			
Localidad de memoria 409FH	Dirección de memoria en la cual se almacena la longitud del patrón de caracteres decimales resultantes			
Forma de llamada	Mnemónico	Código Hexadecimal		
	CALL BN2DEC	CD	6F	01
Nota(s)				
Localidad de memoria 4060H	Almacena la dirección de memoria en la cual se va a almacenar la longitud de del resultado			
Localidad de memoria 4065H	Almacena la longitud actual del buffer			
Localidad de memoria 4066H	Almacena el signo del número, los primeros 15 bits del patrón binario se toman como el dato y el decimosexto bit se toma como el signo (si está activo, se toma como un número negativo, si está inactivo se toma como un número positivo)			
Ejemplo	Mnemónico	Código Hexadecimal		
	LD DE, 0111111111111111B	11	FF	7F
	LD HL, 409FH	21	9F	40
	CALL BN2DEC	CD	6F	01

Nombre	Función			
DEC2BIN	Convierte un patrón de datos decimales a un patrón de datos en binario			
Argumento(s)				
Registro HL	Almacena la localidad de memoria donde inicia el patrón de datos decimales			
Registro B	Almacena el número de datos en formato decimal			
Resultado(s)				
Registro HL	Almacena el resultado en formato binario			
Forma de llamada	Mnemonico	Código Hexadecimal		
	CALL DEC2BN	CD	B6	01
Nota(s)				
Ninguno				
Ejemplo	Mnemónico	Código Hexadecimal		
	LD HL, 4800H	21	00	48
	LD B, 04H	06	04	
	CALL DEC2BN	CD	B6	01

Nombre	Función		
POSSTR	Busca un subpatrón de caracteres en un patrón de caracteres		
Argumento(s)			
Registro HL	Almacena la localidad de memoria donde inicia el patrón de caracteres		
Registro A	Almacena el número de caracteres del patrón de caracteres		
Registro DE	Almacena la localidad de memoria donde inicia el subpatrón de caracteres		
Registro C	Almacena el número de caracteres del subpatrón de caracteres		
Resultado(s)			
Registro A	Almacena la posición a partir de la cual se encuentra la subcadena en la cadena de caracteres, si el resultado es 0, la subcadena no fue encontrada en la cadena		
Forma de llamada:	Mnemónico	Código Hexadecimal	
	CALL POSSTR	CD	02	02
Nota(s)			
Localidad de memoria 4067H	Almacena la dirección de inicio del patrón de caracteres		
Localidad de memoria 4069H	Almacena la dirección de inicio del subpatrón de caracteres		
Localidad de memoria 406CH	Almacena la longitud de la subcadena de caracteres		
Localidad de memoria 406DH	Almacena el índice actual (carácter) de la cadena de caracteres que está siendo verificada		
Ejemplo:	Mnemónico	Código Hexadecimal	
	LD HL, 4800H	21	00	48
	LD A, 0FH	3E	0F	
	LD DE, STRPRM	11	A6	0B
	LD C, 08H	0E	08	
	CALL POSSTR	CD	02	02

Habiéndose descrito las subrutinas que componen las funciones que pueden ser empleadas por los usuarios en la depuración de sus programas, a continuación se describen comandos que pueden ser ejecutadas directamente por el usuario sin necesidad de programarlos:

Nombre	Función
CLS	Efectúa una llamada a la función SUBCLS (descrita con anterioridad)
Nota(s)	
Ninguna	
Ejemplo:	CLS

Nombre	Función
SUMA	Efectúa la suma de dos números decimales introducidos como argumento en la línea de órdenes del sistema
Nota(s)	• Los argumentos deben estar separados del comando por un espacio en blanco y entre ellos por una coma, sin espacio en blanco entre ambos. • El resultado se muestra durante un lapso de tiempo de 2 segundos, después del cual los resultados son borrados de la pantalla • Si los argumentos son incorrectos, se muestra un mensaje en la pantalla indicando cual es el formato del comando y de sus argumentos
Ejemplo:	SUMA 129,345

Nombre:	Función:
PROGRAMA	Inicia el modo de programación del sistema
Nota(s)	• La localidad de memoria destinada para los programas de los usuarios inician a partir de 4800h • El formato de los programas debe estar en binario • El sistema convierte los datos binarios a hexadecimal para que puedan ser corroborados por los usuarios • Si se desea predefinir etiquetas (texto), estos deben de estar al inicio del segmento de memoria destinado para los programas y efectuar un brinco incondicional al final de las etiquetas • Se puede eliminar un carácter a la izquierda con la tecla de retroceso • Se puede eliminar la línea entera con la tecla DELETE • No se encuentran activadas el segmento de teclas numéricas
Ejemplo:	PROGRAMA

Nombre:	Función:
VOLCADO	Inicia el proceso de volcado del programa del usuario
Nota(s)	• El formato de las instrucciones contenidas en el programa del usuario se muestra en hexadecimal. • El contenido de cada localidad de memoria a partir de 4800H y hasta el final del programa se muestra en la pantalla durante un segundo, tiempo después del cual se borra la pantalla y se muestra el siguiente dato • Este comando sólo es reconocido cuando el sistema se encuentra en modo de programación del sistema
Ejemplo:	VOLCADO

Nombre:	Función:
CAMBIA	Permite cambiar el contenido de la localidad de memoria de programa del usuario
Nota(s)	• El argumento del comando debe estar en formato hexadecimal y separado por un solo espacio en blanco del comando • El sistema no verifica la dirección, dado lo cual se debe tener cuidado para no programar una localidad de memoria diferente a la dirección de programa del usuario • Este comando sólo es reconocido cuando el sistema se encuentra en modo de programación del sistema
Ejemplo:	CAMBIA 4805H

Nombre:	Función:
EJECUTA	Inicia la ejecución del programa del usuario
Nota(s)	• La ejecución se inicia efectuando un brinco incondicional a la localidad de memoria 4800H • El sistema añade una instrucción de salto incondicional al programa principal después de que se hayan ejecutado las instrucciones de los usuarios • El sistema no cuenta con una función de aborto de programa • Si ocurrieran errores en el programa, se necesita reiniciar el sistema • Si no se encuentra ningún programa cargado en la memoria, se muestra un mensaje de error • Este comando sólo es reconocido cuando el sistema se encuentra en modo de programación del sistema
Ejemplo:	EJECUTA

Nombre	Función
NUEVO	Reinicia el modo de programación del sistema
Nota(s)	• Reinicia el Contador de Programa del usuario • Al ejecutar un comando de reinicio de programación, los datos del programa anterior se pierden • Este comando sólo es reconocido cuando el sistema se encuentra en modo de programación del sistema
Ejemplo	NUEVO

Nombre	Función
DETEN	Detiene el modo de programación del sistema y vuelve al estado de ejecución de comandos del usuario
Nota(s)	• Resetea el Contador de Programa del usuario • Se efectúa una llamada a la subrutina SUBCLS • Al ejecutar un comando de detención del modo de programación del sistema los datos del programa se pierden • Este comando sólo es reconocido cuando el sistema se encuentra en modo de programación del sistema
Ejemplo	DETEN

La siguiente tabla muestra una forma que muestra como se puede programar las localidades de memoria destinadas para tal fin:

Programa número:	Titulo:
Descripción:	Notas:

Dirección	Mnemónico	Cod. Hex.	Cod. Bin.	Dirección	Mnemónico	Cod. Hex.	Cod. Bin.
4800H				481AH			
4801H				481BH			
4802H				481CH			
4803H				481DH			
4804H				481EH			
4805H				481FH			
4806H				4820H			
4807H				4821H			
4808H				4822H			
4809H				4823H			
480AH				4824H			
480BH				4825H			
480CH				4826H			
480DH				4827H			
480EH				4828H			
480FH				4829H			
4810H				482AH			
4811H				482BH			
4812H				482FH			
4816H				4830H			
4817H				4831H			
4818H				4832H			
4819H				4833H			

Tabla 6.2 FORMATO DE DISEÑO DE PROGRAMAS DEL SISTEMA

6.3. Resumen

El último paso del diseño del Sistema Mínimo es el diseño del sistema operativo. Este sistema operativo debe estar acorde con los requerimientos y los dispositivos periféricos que puede controlar el sistema.

El sistema operativo del Sistema Mínimo debe proporcionar un control de los recursos del sistema, mediante la programación de:

1) Funciones del Usuario: Los cuales proveerán a los usuarios un entorno para la ejecución de sus propios programas.
2) Funciones del sistema: Los cuales proveerán los medios para manejar los recursos internos del sistema.

Recuerde que antes de programar el sistema operativo para el Sistema Mínimo, debe definir sus funciones, basado en los dispositivos que controla el Sistema Mínimo.

Para finalizar, el sistema operativo del Sistema Mínimo cuenta con un conjunto de funciones que pueden ser usados por los usuarios en sus programas o ejecutados directamente, sin necesidad de efectuar una llamada a estas funciones dentro de un programa.

CONCLUSIONES

El rápido incremento de la potencia de cálculo y uso de los microprocesadores le ha exigido a los profesionistas una actualización en el manejo de los sistemas automáticos con el objeto de poder acceder hacia áreas de trabajo mayor especializadas y con una mejor fuente de ingresos.

A los ingenieros en especial, este hecho les ha puesto en el camino de la actualización en sus áreas de conocimiento ya que les obliga a mantenerse al día en las nuevas tecnologías derivadas de la utilización de los microprocesadores con el objeto de poder brindar una asesoría y un apoyo técnico eficiente a las empresas o instituciones que asi lo requieran.

Aún cuando los microprocesadores tienen diferentes potencias de cálculo dependiendo del número de bits que pueda manejar en su bus de datos y de la magnitud de frecuencia a la cual oscila el reloj que le brinda los pulsos, el microprocesador Z80 se elige por su economia, potencia de calculo, mayor número del juego de instrucciones y simplicidad de uso y manejo comparado con otros microprocesadores de 8 bits existentes en el mercado.

Si bien, existen diversos microcontroladores que permiten extender las capacidades del microprocesador, existen circunstancias en las cuales el proceso de creación de la interfaz entre el microprocesador y cualquier microcontrolador que no haya sido elaborado específicamente para éste, en este caso es conveniente el uso de circuito integrados convencionales, además del uso de la lógica digital tradicional para permitir la comunicación entre el microprocesador y el dispositivo periférico.

El uso de estos circuitos integrados convencionales permite a los diseñadores tener una idea más clara respecto a sus objetivos y forma de realizar las conexiones pertinentes para reducir al mínimo el número de circuitos integrados requeridos, disminuyendo de esta forma el costo total del producto final.

Para economizar más el diseño y la implementación de minisistemas computadores, en lugar de usar los costosos monitores CRT, se pueden usar las pantallas LCD, las cuales aventajan a los primeros con relación a su costo, hardware requerido para su control, tamaño, peso, asi como la potencia aplicada necesaria para que pueda funcionar.

La elaboración de un pequeño sistema operativo para controlar el hardware del Sistema Mínimo permite añadir una disminución del costo del sistema, asi como también permite al diseñador adquirir nociones elementales del diseño de sistemas operativos y sentar las bases para posteriores mejoras del sistema.

La elaboración del programa que le permita al usuario probar sus propios programas reduce en forma drástica la necesidad del diseño de otro Sistema Mínimo a partir de cero, permitiéndole además, la depuración de sus propios programas in site antes de programarlas en un dispositivo de memoria ROM.

CONCLUSIONES

Al elaborar el Sistema Mínimo con las características mencionadas, se deja abierto el tema para que pueda ser retomado por cualquier persona con iniciativa propia y pueda añadirle periféricos adicionales tales como una interfaz serie, un controlador de acceso directo a memoria (DMA), cargar el sistema operativo desde un drive externo, extender las capacidades del sistema operativo, etc.

BIBLIOGRAFIA

1. Harland, David, Z80 Machine code & interfacing, Editorial Edward Arnold, Great Britain, 1990
2. Carr, Joseph, Z80 Users manual, Editorial Reston Publishing Company, Reston Virginia, USA. 1980.
3. Ciarcia, Steve, Construya una microcomputadora basado en el Z80, Editorial Byte Books/McGraw-Hill; Naucalpan de Juárez, Edo. de Mex. 1984.
4. Mandado, Enrique, Procesadores programables. El microprocesador, Editorial Publicaciones Marcombo, S. A., México, D.F., 1983.
5. F. Driscoll, Frederick, Microprocessor-microcomputer technology, Editorial Breton Publishers, Massachusetts, USA, 1983.
6. M. Deitel, Harvey, Operating Systems, 2nd. Ed., Editorial Addison-Wesley, Massachusetts, USA, 1990.
7. Milenkovic, Milan, Sistemas Operativos Conceptos y Diseño, 2da. Ed., Editorial McGraw-Hill, Madrid, España, 1994.

APENDICE A:
USO Y MANEJO DE WINDRAFT

> Si quieres entender acerca de otras dimensiones de existencia, deja que tu imaginación explore dentro de ellas.
>
> Idenfuye

Poder realizar diseños fiables y altamente competitivos de hardware de computadora implica contar con los conocimientos técnicos y económicos suficientes para poder determinar los elementos más fiables y de menor costo global, que permitan competir con diseños existentes en el mercado. Además de contar con software que permitan realizar modificaciones en el hardware y software antes de poderlos construir a gran escala, con la certeza de que funcionarán sin problemas.

Para poder realizar los diseños de hardware y software en el menor tiempo posible, actualmente existe una gran variedad de productos en el mercado, los cuales permiten emular la forma de los dispositivos electrónicos, calcular la potencia de carga que consumirán cada uno de estos dispositivos electrónicos, la potencia total consumida así como simular mediante computadora la ejecución de las instrucciones con las que cuenta el microprocesador elegido.

De tal suerte que iniciarse en el arte del diseño de sistemas digitales basado en microprocesadores, implica una breve pero minuciosa investigación de los productos existentes en el mercado y que tendrán como objetivo acelerar el proceso de diseño, prueba, modificación y elaboración del producto final.

Esta breve etapa implica tomar en cuenta el costo del software, la capacidad de diseño del mismo, los posibles errores de programación que pudiera tener así como las soluciones recomendadas por los fabricantes del producto a estos errores y finalmente, se debe tener en consideración la facilidad de instalación en los equipos de cómputo con los que se cuenta, ya que esto será un factor determinante para no incrementar el costo del diseño de los sistemas digitales al no contar con la infraestructura adecuada para un buen inicio en este arte.

Una de tales herramientas de desarrollo es conocida como Windraft, este software permite diseñar los sistemas de hardware en la microcomputadora, así como modificar el diseño del circuito electrónico, realizar conexiones con subdiagramas, obtener una versión impresa del circuito electrónico, etc. Este software de diseño de circuitos electrónicos es desarrollado y puesto a la venta por la empresa Ivex, para mayor referencia del software se puede contactar a esta empresa en el siguiente URL:

http://www.ivex.com

Una breve guía del uso del software se muestra a continuación.

Al seleccionar el icono de Windraft, se abre una ventana como la siguiente

Fig. 7.1 INICIO DE WINDRAFT

Preste especial atención a la capacidad de pines, ya que si desea realizar el diseño de un sistema con un mayor número de pines a su capacidad, no le permitirá guardar sus archivos (ello se soluciona adquiriendo la licencia para manejar una mayor cantidad de pines). Windraft cuenta con los siguientes elementos de menú

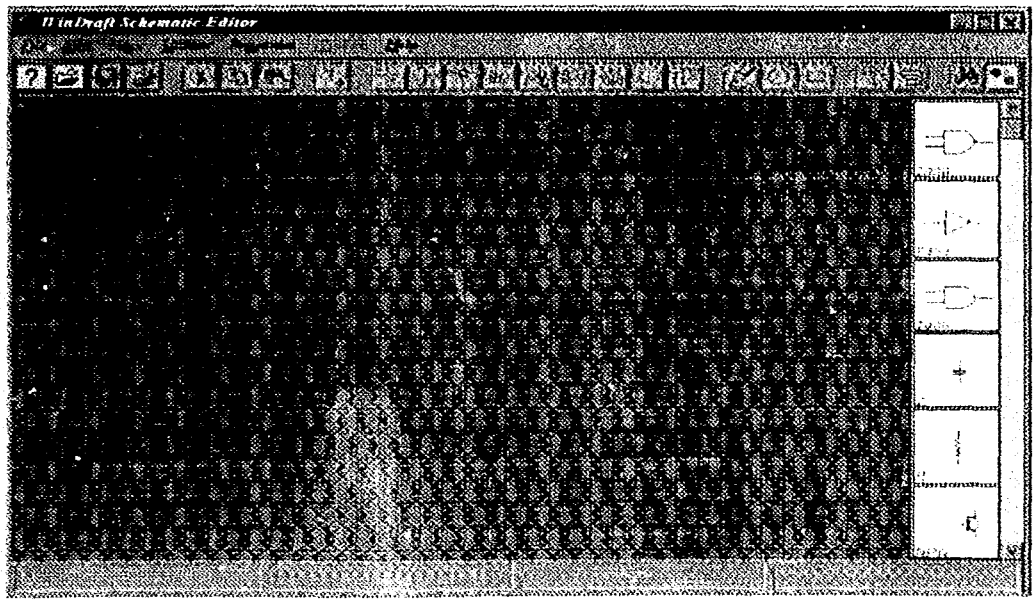
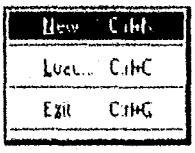


Fig. 7.2 AMBIENTE DE TRABAJO DE WINDRAFT

Los elementos del menú tienen un funcionamiento como sigue.

a) File

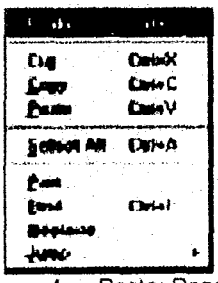


La directiva New es usada para crear un nuevo diagrama, librería o un bloque. La directiva Load es usada para cargar en el ambiente de trabajo de Windraft un diagrama, librería o bloque. Éstas se abren mediante cuadros de diálogo que permiten seleccionar el archivo deseado.

Al tener abierto un archivo, las opciones del menú File se incrementan con las siguientes opciones:

1. Save: Guarda el archivo actual en uso.
2. Save as: Guarda el archivo en uso con el nombre proporcionado por el usuario.
3. Print: Configura e imprime un archivo en una impresora local.
4. Print preview: Visualiza el formato final del archivo antes de imprimirlo.
5. Save all: Almacena todos los archivos en uso.
6. Close all: Cierra todos los archivos en uso.
7. Exit: Cierra la ejecución del programa Windraft.

b) Edit

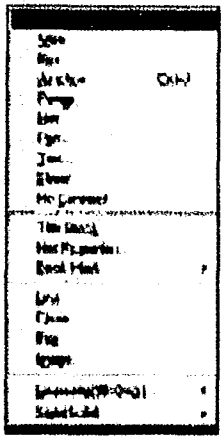


Permite realizar acciones tales como copia, selección y eliminación de objetos. Además de ser parte del menú Edit, para este tipo de acciones existen hot keys (Ctrl-X para cortar, Ctrl-C para copiar y Ctrl-V para pegar).

El menú Edit tiene las siguientes directivas:

1. Undo: Cancela la última operación realizada.
2. Cut: Elimina un objeto seleccionado.
3. Copy: Copia en el buffer un objeto seleccionado.
4. Paste: Pega el objeto almacenado en la memoria en el archivo.
5. Select all: Selecciona todos los objetos de la ventana actual.
6. Selected object: Edita y modifica el objeto seleccionado.
7. Find: Buscar un texto en el archivo, posicionando el cursor en el elemento encontrado.
8. Replace: Modifica en forma global un texto.
9. Jump: Mueve el cursor a la posición deseada en el archivo.

c) Place



Este menú contiene los elementos para diseñar circuitos electrónicos, los cuáles son:

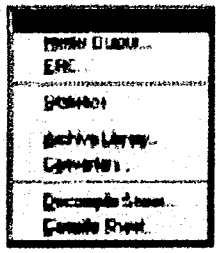
1. Part: Coloca un objeto en el archivo o en la paleta
2. Wire: Realiza conexiones eléctricas en los objetos del circuito eléctrico.
3. Bus: Coloca un bus gráfico en el diseño del circuito eléctrico actual.
4. Junction: Coloca un elemento de unión para varios objetos del diseño del circuito eléctrico actual.
5. Power: Une a tierra o Vcc un objeto del diseño del circuito eléctrico.
6. Sheet Net: Realiza conexiones eléctricas o buses a subdiseños eléctricos.
7. Port: Interconecta eléctricamente señales, incluyendo cables y buses.
8. Text: Incluye información sobre objetos, tales como etiquetas, comentarios, etc.
9. Sheet symbols: Son usados en diseños jerárquicos, cada símbolo representa un subdiseño del sistema, cada subdiseño contiene conectores que permiten la conexión de señales entre un diseño y sus subdiseños en la jerarquía.
10. No connect: Identifica a un pin o un dispositivo que permite una desconexión intencional.
11. Title block: Pone en el archivo un title block creado con anterioridad.
12. Net properties: Modifica las características de configuración.
13. Book mark: Coloca un book mark en la posición deseada en el circuito electrónico y se puede volver a él mediante el comando Jump.
14. Line: Coloca una línea en el diseño del circuito eléctrico actual.
15. Circle: Coloca un círculo en el diseño del circuito eléctrico actual.
16. Box: Coloca un cuadro en el diseño del circuito eléctrico actual.
17. Image: Coloca una imagen en formato bmp en el diseño del circuito eléctrico actual.
18. Geometry: Es usado para elegir el ángulo de torsión de las líneas que se colocan en el diseño.
19. Style: Se usa para determinar el tipo de línea que se va a colocar en el diseño: punteado o continúo.

d) Tools

Este menú contiene los siguientes elementos:

1. Annotate: Se usa para escanear un diseño con todos sus sublementos y actualizar automáticamente las referencias de todas las partes en el diseño. Actualiza también el número de pines de las partes que tienen múltiples dispositivos en el paquete. De igual forma, sobreescribe todas las referencias actuales o puede ser usada para incrementar las referencias que aún no han sido actualizadas.
2. Back annotate: Actualiza las referencias a partes del diseño que están dadas en un archivo de texto que lista las viejas y las nuevas referencias. Este archivo puede ser creado con un simple editor de textos, éste debe ser creado antes de una ejecución del comando Back annotate, su extensión por default debe ser .BAN.
3. Cleanup: Se usa para escanear el diseño actual y checar conexiones, buses y uniones y otros objetos colocados unos sobre otros y remover los translapes.

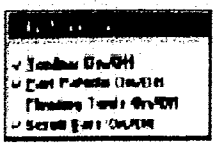
e) Utilities



Este menú contiene las siguientes herramientas:

1. BOM: Crea una lista resumida de todas las partes usadas en el diseño.
2. Net Output: Es usado para generar un archivo netlist.
3. ERC: Se usa para checar el diseño de conformidad con las leyes eléctricas básicas.
4. Statistics: Muestra estadísticas de los diseños abiertos.
5. Archive library: Escanea el diseño completo y genera un simple archivo binario que contiene todas las partes usadas en el diseño.
6. Converters: Convierte los diseños OrCAD/SDT de 16 o 32 bits.
7. Decompile sheet: Permite decompilar los archivos binarios Windraft en archivos ASCII equivalentes.
8. Compile sheet: Compila un archivo Windraft en formato ASCII a su equivalente formato archivo binario.

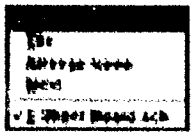
f) Properties



Las opciones de este menú permiten definir los parámetros del sistema, tal como se muestra a continuación:

1. Preferences: Se usa para activar los parámetros de Windraft y las opciones del usuario.
2. Tool bar On/Off: Muestra u oculta la barra de herramientas.
3. Part palette On/Off: Muestra u oculta la paleta de partes.
4. Floating tools On/Off: Coloca la barra de herramientas en un lugar distinto al habitual.
5. Scroll bars On/Off: Muestra u oculta la barra de desplazamiento del área de trabajo.

g) Windows



Este menú contiene herramientas para modificar la forma de mostrar los archivos abiertos.

h) Help

Este menú contiene la ayuda propia del sistema, así como otras utilerías que se muestran a continuación:

1. Shortcuts: Muestra los shortcuts definidos para el teclado.
2. Tips: Despliega el cuadro de diálogo para los tips del sistema.
3. Registration: Permite introducir los datos de registro del software para incrementar su capacidad de manejo de pines.
4. About: Muestra un cuadro de dialogo con la versión del sistema y el password de acceso.

Sin lugar a dudas, el elemento más importante es la barra de herramientas, ya que permite simplificar el proceso de diseño de circuitos electrónicos al habilitar la ejecución de comandos mediante un icono.

La barra de herramientas y el menú de herramientas flotantes se muestra a continuación:

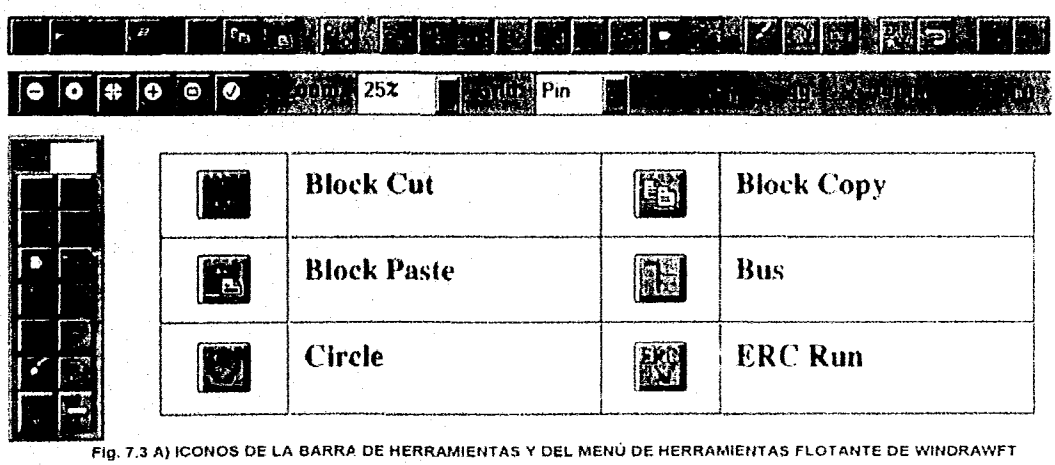


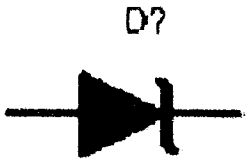
Fig. 7.3 A) ICONOS DE LA BARRA DE HERRAMIENTAS Y DEL MENÚ DE HERRAMIENTAS FLOTANTE DE WINDRAWFT

	File Open		File Save
	Find Text		Geometry
	Ground		Help
	Junction		Line
	Module Port		No Connect
	Part		Part Palette
	Power		Print
	Preferences		Sheet Net
	Rectangle		Repeat
	Text		Undo
	Wire		
	Center Screen		Refresh Screen
	Zoom in		Zoom to Fit
	Zoom out		Zoom Window

Fig. 7.3 B) ICONOS DE LA BARRA DE HERRAMIENTAS Y DEL MENÚ DE HERRAMIENTAS FLOTANTE DE WINDRAWFT

Los elementos usados en el diseño de un sistema electrónico son:

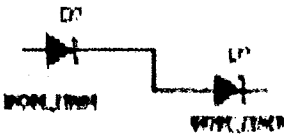
1) Parts:



Son objetos gráficos que se colocan en el área de trabajo para representar los dispositivos electrónicos en el circuito electrónico. Windraft incluye varias librerías de dispositivos electrónicos y un editor de librerías para crear nuevos objetos Parts.

Para colocar un Part en el área de trabajo, use el comando Place|Part en el menú o seleccione el icono de la barra de herramientas (toolbar).

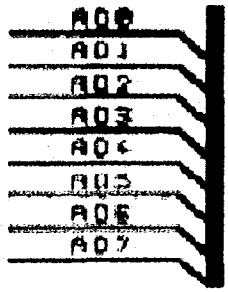
2) Wires:



Coloca líneas en el diagrama del circuito electrónico para representar una unión entre los objetos, tales como los pines o Parts y los objetos de fuente de potencia.

Para colocar una línea en un diagrama, use el comando Place|Wire del menú de comandos o presione el icono de la barra de herramientas y se dibuja la línea de un pin a otro pin.

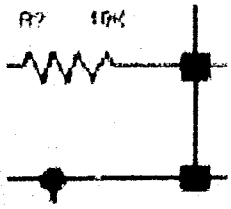
3) Buses:



Los Buses se usan para representar un arreglo de señales como una unidad simple en un diagrama.

Para colocar un bus en un diagrama, use el comando Place|Bus del menú o el icono de la barra de herramientas y se dibuja el bus.

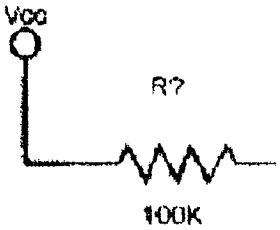
4) Junctions:



Los objetos Junctions indican una conexión física entre líneas (Wires). Se puede elegir entre tres tipos diferentes de objetos Junctions (Cuadrados, Circulares y en forma de Diamante), además de tres tamaños diferentes (pequeño, mediano y grande).

Para colocar un objeto Junction en un diagrama, use el comando Place|Junction o el icono de la barra de herramientas y presione el botón izquierdo del mouse para colocar el objeto en la posición deseada. Windraft coloca automáticamente un objeto Junction donde una línea (Wire) termina y otra línea (no donde los objetos Wire se cruzan)

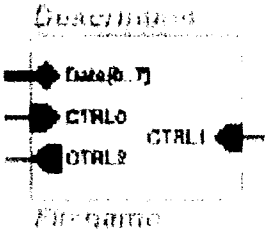
5) Power Objetcs.



Los objetos Power representan una conexión a una fuente de potencia. Windraft tiene varios tipos de objetos Power tales como el circulo de Vcc mostrado en la gráfica.

Para colocar un objeto Power en un diagrama, use el comando Place|Power del menú o seleccione el icono de la barra de herramientas, posteriormente seleccione el tipo de objeto, introduzca un nombre y coloque el objeto en la posición deseada usando el mouse

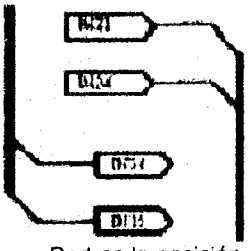
6) Net Symbol:



Los objetos Net Symbol son usados para indicar una conexión a un módulo de Port o a un subdiagrama jerárquico

Para colocar un objeto Net Symbol en un diagrama, use el comando Place|Net del menú o el icono de la barra de herramientas, seleccione el tipo (Entrada, Salida, E/S o No Especificado) y coloque el objeto Net Symbol en la posición deseada

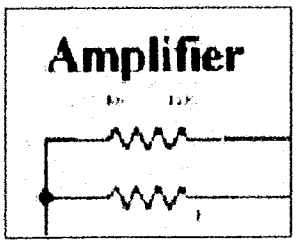
7) Port:



Los objetos Port (también llamados módulos Port) son usados dentro de un diagrama dentro de múltiples diagramas en un proyecto para conectar eléctricamente señales entre subdiagramas.

Para colocar un objeto Module Port, use el comando Place|Port del menú o el icono de la barra de herramientas, después seleccione el tipo (Entrada, Salida, E/S o No Especificado) y finalmente coloque el objeto Port en la posición deseada del diagrama.

8) Text:



El objeto Text es usado para incluir información, etiquetas, comentarios y ligas de comandos en el área de trabajo

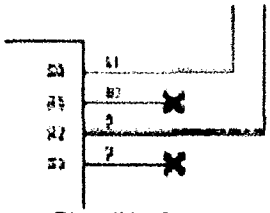
Para colocar un objeto Text en el diagrama, use el comando Place|Text o el icono de la barra de herramientas, introduzca el texto deseado en el cuadro de diálogo y coloque el objeto Text usando el mouse en la posición deseada del diagrama.

9) Sheet Symbols:

Los objetos Sheet Symbols son simbolos de bloques formados que representan subdiagramas un diseño. Cada objeto Sheet Symbol representa un subdiagrama.

Para colocar un objeto Sheet, use el comando Place|Sheet del menú, introduzca un nombre para el objeto colóquelo en la posición deseada con el mouse.

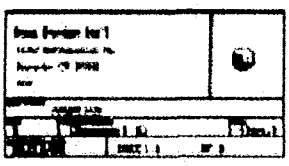
10) No Connect Symbol.



El objeto No Connect Symbol identifica los pines en un dispositivo que van a quedar desconectados. Este simbolo ocasiona que la herramienta ERC ignore los pines cuando se genera un reporte de pines no conectados.

Para colocar un objeto No Connect en el diagrama, use el comando Place|No Connect o el icono  de la barra de herramientas flotante y coloque el objeto al final del pin deseado.

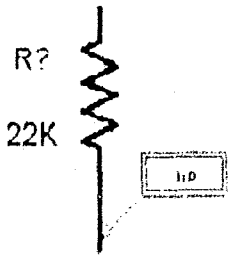
11) Title Block:



Un objeto Title Block se usa para identificar cierto tipo de información en el diagrama. Éste contiene información tal como el nombre de la compañia, la dirección, el titulo del diagrama, el número, el tamaño y la revisión.

Para colocar un objeto Title Block, use el comando Place|Title Block del menú, seleccione el objeto Title Block deseado de los archivos disponibles y colóquelo en el área de trabajo.

12) Net Properties:



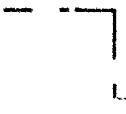
El objeto Net Properties coloca un simbolo gráfico junto a una linea (Wire) deseada para modificar algunas de sus propiedades.

Para colocar un simbolo Net Properties, use el comando Place|Net Properties del menú, y coloque el símbolo NP de tal forma que el cursor permita seleccionar la linea (Wire) a la que se le desea asociar las nuevas caracteristicas.

13) Book Mark:

Activa posiciones invisibles en el diagrama. Éste puede ser usado para brincar a las posiciones especificadas por el comando Book Mark. Se puede especificar hasta 5 diferentes Book Marks en un diagrama. Para colocar un Book Mark, use el comando Place|Book Mark del menú y coloque el objeto Book Mark en la posición deseada del área de trabajo.

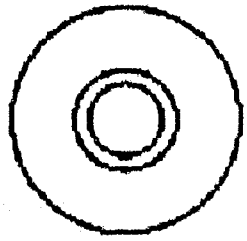
14) Line:



Los objetos Line son usados en los diagramas para dibujar lineas diferentes a las conexiones eléctricas

Para colocar un objeto Line en un diagrama, use el comando Place|Line del menú o el icono  en la barra de herramientas y dibuje la linea en la posición deseada del área de trabajo. Éstas lineas pueden ser sólidas o punteadas.

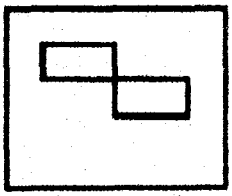
15) Circle:



Los objetos Circle son usados en los diagramas para dibujar circulos diferentes de las conexiones eléctricas

Para colocar un objeto Circle en un diagrama, use el comando Place|Circle del menú o el icono  de la barra de herramientas y dibuje el circulo en la posición deseada del área de trabajo. Los circulos pueden ser sólidos o punteados.

16) Box:



Los objetos Box son colocados en el diagrama para dibujar cuadros diferentes de las conexiones eléctricas.

Para colocar un objeto Box en el diagrama, use el comando Place|Box del menú o el icono  de la barra de herramientas y dibuje el cuadro en la posición deseada del área de trabajo. Estos cuadros pueden ser sólidos o punteados.

APENDICE B:
USO Y MANEJO DE
Z80 SIMULATOR

La siguiente herramienta que apoya mucho el proceso de diseño de sistemas digitales basados en el microprocesador Z80 es el software denominado Z80 Simulator, cuya primordial función es simular la ejecución de instrucciones del microprocesador Z80 en una computadora. Esto permite corregir los errores que pudieran ocurrir en un programa antes de que sea programado en una memoria ROM,

El software Z80 Simulator es producido y puesto a la venta por la compañía Barleywood, para obtener más información respecto a este software y obtener una versión de prueba con vigencia de 30 días del mismo software, se puede contactar a la compañía en el siguiente URL:

http://www.barleywood.com

Una breve guía del uso de este software se muestra a continuación:

Al seleccionar el icono z80en11, se abre una ventana como la siguiente:

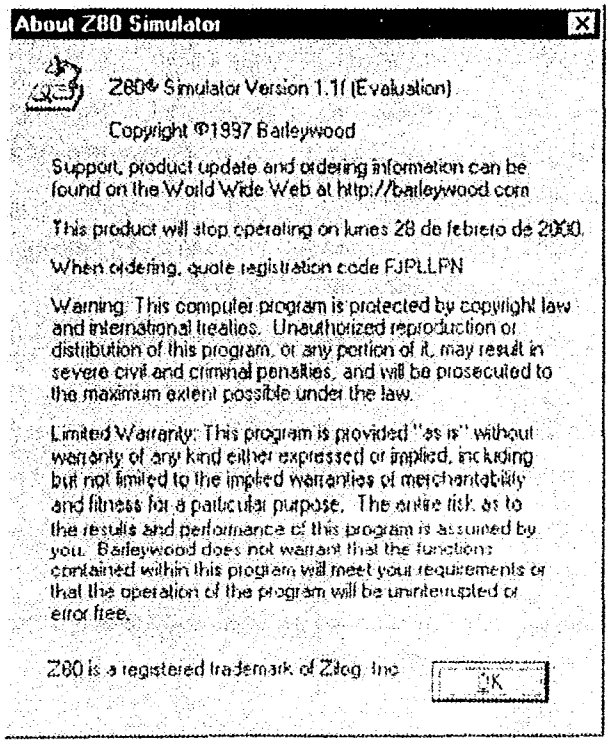


Fig. 7.4 INICIO DE Z80 SIMULATOR

Preste especial atención al código de registro ya que si desea obtener la versión comercial para esta versión de prueba, debe proporcionar este código

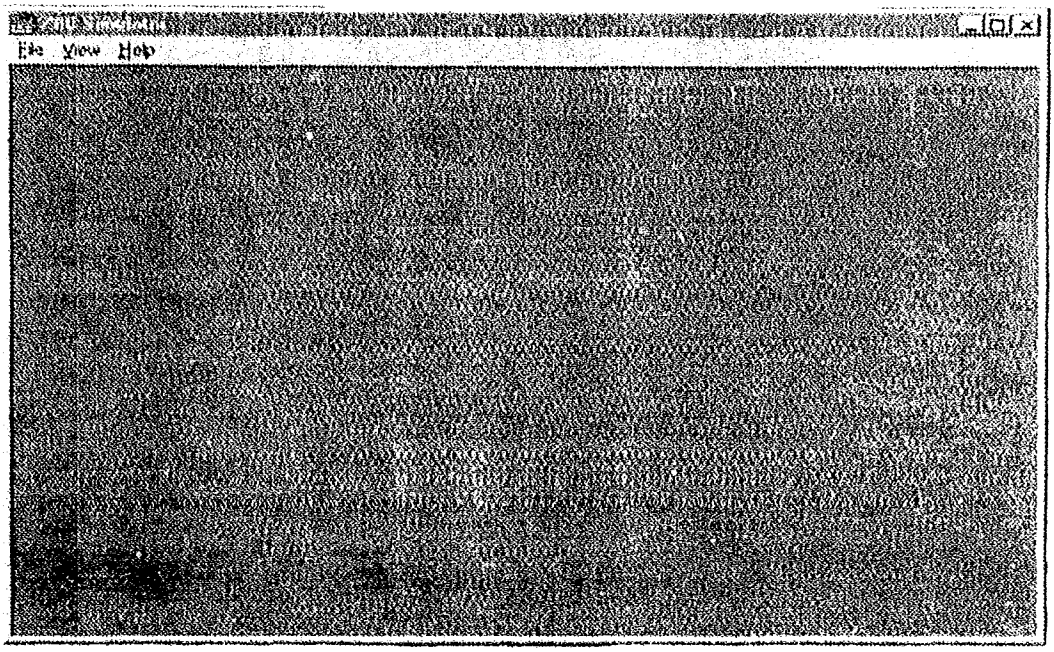


Fig. 7.3 AMBIENTE DE TRABAJO DE Z80 SIMULATOR

Los elementos del menú tienen un funcionamiento como sigue:

a) File: Permite realizar las operaciones más comunes tal como carga de un archivo almacenamiento de un archivo, etc. Al tener abierto un archivo, los elementos del menú se incrementa con las siguientes opciones:

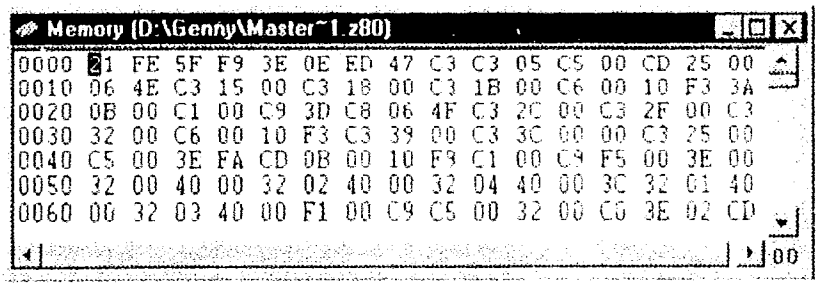
   1. Open: Abre un archivo mediante deseado, otra forma de ejecutar este comando es mediante la presión simultánea de las teclas Control+O.
   2. Save: Almacena un archivo deseado en una ruta deseada, se puede ejecutar este comando mediante la presión simultánea de las teclas Control+S.
   3. Save as: Guarda un archivo con un nombre diferente en una ruta deseada otra forma de ejecución del mismo comando es mediante la presión simultánea de las teclas Control+A.
   4. Print: Imprime el archivo, también se puede ejecutar el mismo comando mediante la presión simultánea de las teclas Control+P.
   5. Close: Cierra el archivo actual, también se puede ejecutar el mismo comando mediante la presión simultánea de las teclas Control+F4.
   6. Exit: Cierra el programa.

b) Edit: Permite realizar las operaciones más comunes sobre un archivo, tales como deshacer el último cambio efectuado, copiar un segmento del programa, etc. Las opciones de este menu son:

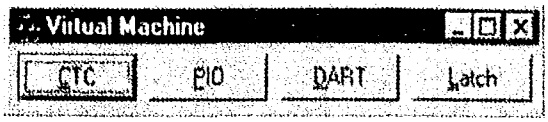
1. Undo: Deshace el último cambio efectuado sobre el archivo, se puede ejecutar este comando mediante la presión simultánea de las teclas Control+Z.
2. Cut: Elimina el segmento seleccionado del archivo, también puede ejecutarse el comando mediante la presión simultánea de las teclas Control+X.
3. Copy: Copia el segmento seleccionado del archivo en el buffer, otra forma de ejecución del comando es mediante la presión simultánea de las teclas Control+C.
4. Paste: Coloca el segmento almacenado en el buffer en la posición actual del cursor en el archivo, también se puede ejecutar el comando mediante la presión simultánea de las teclas Control+F.
5. Find: Busca una cadena de caracteres en el archivo actual, también permite al mismo tiempo la sustitución de la cadena de caracteres por otra. Otra forma de ejecución del comando es mediante la presión simultánea de las teclas Control+F.
6. Find Next: Reinicia la búsqueda de la misma cadena de caracteres si ésta ya ha sido encontrada, se puede ejecutar el mismo comando mediante la presión de la tecla F3.

c) View: Permite abrir un nuevo archivo o una herramienta de desarrollo. Las opciones de este menú son:

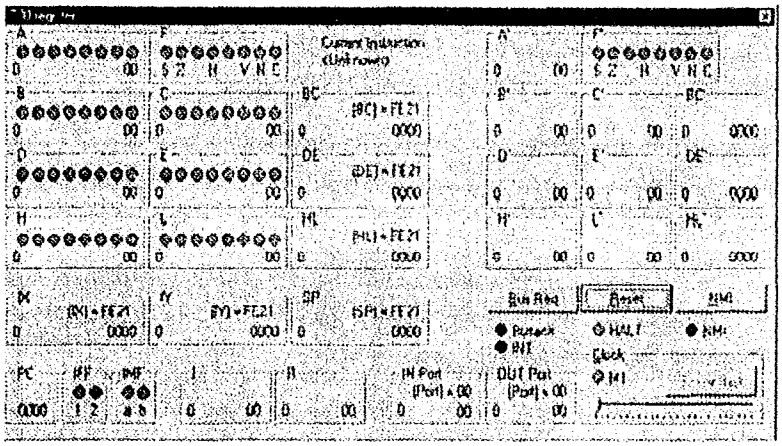
1. New Assembler Source: Abre un nuevo archivo, este comando también puede ser ejecutado mediante la presión simultánea de las teclas Control+N.
2. Memory Viewer: Muestra el contenido de la memoria en formato hexadecimal. Los datos contenidos en la memoria son el resultado de la compilación del programa o de la apertura de un archivo previamente compilado. Los datos de esta memoria son semejantes a las que se van a grabar en un dispositivo físico de memoria.



3. Virtual Machine. Muestra los microcontroladores pertenecientes a la familia del microprocesador Z80. La selección de cada uno de ellos muestra el microcontrolador respectivo. Al estar activo el microcontrolador, éste puede ser programado para su simulación simultánea con el programa fuente.



4. Registers: Muestra todos los registros contenidos en el microprocesador Z80, los puertos de Entrada/Salida y las señales de control más importantes del microprocesador, las cuales permiten el inicio o la detención de la simulación de la ejecución de las instrucciones programadas en el archivo antes de ser grabadas en una memoria ROM. Para iniciar la secuencia de ejecución de las instrucciones programadas en el archivo, se aplica un reset y se elige la velocidad de ejecución de las instrucciones.



d) Windows: Conmuta entre las diversas ventanas abiertas en el área de trabajo, permitiendo la edición simultánea de varios programas, visualización de los contenidos de la memoria o el estado de la ejecución del programa actual.

e) Tools: Presenta varias opciones, dependo de la ventana activa en ese momento, las opciones más comunes para éste son:

1. Assemble: Genera el código máquina en formato hexadecimal de las instrucciones programadas en el archivo fuente.
2. Toggle Breakpoint: Coloca un punto de ruptura en la posición actual del cursor, éste permite la detención de la simulación de ejecución de las instrucciones programadas en el archivo fuente.
3. Ascii Display: Visualiza el contenido de la memoria en formato hexadecimal y su correspondiente código ASCII.
4. Disassemble: Desensambla el contenido de la memoria, mostrando las instrucciones originales del archivo fuente.

5. Break Before IN: Detiene la ejecución de las instrucciones programadas en el archivo fuente antes de efectuar una lectura de un Puerto de Entrada/Salida
6. Break Before ON: Detiene la ejecución de las instrucciones programadas en el archivo fuente antes de ejecutar una escritura en un Puerto de Entrada/Salida
7. Break Before Ram Write: Detiene la ejecución de las instrucciones programadas en el programa fuente antes de ejecutar la escritura en una localidad de memoria RAM

f) Help: Muestra la ayuda del Simulador, las opciones de este menú son:

1. Contents: Muestra una ayuda adicional para los contenidos generales del simulador. Se puede activar este comando mediante la presión de la tecla F1.
2. Obtaining Technical Support: Muestra información sobre centros de asesoría en caso de tener algún problema con el simulador.
3. About Z80 Simulator: Muestra los datos generales del simulador, tales como compañía distribuidora, fecha de expiración (en caso de ser software de prueba), código de registro, etc.