15
20

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

ARAGON

DISEÑO DE UN BANCO DE MEMORIA INTERFAZADO A UNA COMPUTADORA PERSONAL Y A UN SISTEMA DE DESARROLLO DE CONTROLADORES MICROPROGRAMADOS

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

FELIX ALBERTO MENDOZA CASTILLO

A S E S O R :

ING. DAVID JAIME GONZALEZ MAXINEZ

ENEP ARAGON

MEXICO, D. F.

1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Contenido

Prólogo

El objetivo de este proyecto es diseñar un banco de memoria de acceso aleatorio interconectado a una Computadora Personal y realizar la programación necesaria que permita la transferencia de datos, para que posteriormente forme parte de un equipo de desarrollo experimental de microcontroladores Bit-Slice.

Dicho equipo de desarrollo experimental de microcontroladores Bit-Slice, intenta mostrar al estudiante de circuitos electrónicos digitales de la ENEP Aragón, el uso de circuitos modulares en el diseño de controladores y microprocesadores basados en tecnología Bit-Slice.

La finalidad del estudio de circuitos electrónicos digitales es el desarrollo de sistemas que resulten provechosos en algún sentido. Tales sistemas pueden tener la meta de ser comercializados o usados internamente;

en cualquier caso, el diseño es similar, aunque la secuencia de pasos es diferente de un lugar a otro.

La mayoría de las ideas para productos nuevos, comerciales o experimentales se generan con frecuencia a partir del conocimiento de las necesidades del mercado y la experiencia técnica. Cada producto nuevo debe definirse cuidadosamente, para así lograr un adecuado diseño del mismo; éste proceso debe ser enriquecido con las sugerencias de los futuros usuarios, para asegurar que el éxito de las pruebas del prototipo sean reflejadas durante la operación o consumo del sistema final.

En este trabajo se presenta un diseño que trata de alcanzar lo anteriormente expresado; esto es, un sistema que cumpla con las especificaciones requeridas. Por ello resulta necesario analizar el ambiente en el cual el sistema va a operar y la forma en que va a interactúar con él.

Por tal razón, en el diseño del Banco de Memoria se presentan las bases teóricas sobre los Controladores Microprogramados, y también sobre las Computadoras Personales.

Una vez que se entiende el ambiente operativo del sistema, es ya posible diseñar apropiadamente el banco de memoria y escribir el programa de aplicación, lo cual constituye el objeto central de ésta tesis.

Finalmente, la integración del hardware y software se lleva a cabo cuando las pruebas del prototipo, en tarjetas experimentales, se efectúen adecuadamente, y en ese momento se habrá alcanzado el propósito del presente trabajo.

1

Sistema de Controladores Microprogramados

Introducción

El proceso de diseño lógico es una tarea compleja. Múltiples empresas han desarrollado diversas técnicas de diseño automático basadas en computador para facilitar el proceso de diseño. Sin embargo las especificaciones para el sistema y el desarrollo de procedimientos algorítmicos para lograr las tareas requeridas no pueden ser automatizados y requieren del razonamiento del diseñador humano.

La parte de mayor desafío y creatividad del diseño, es el establecimiento de objetivos de diseño y la formulación de algoritmos y procesos involucrados. Esta tarea requiere una cantidad considerable de experiencia e ingenio por parte del diseñador.

Un algoritmo es un procedimiento para obtener una solución a un problema. Un algoritmo diseñado es un procedimiento para configurar tal solución con un

conjunto de materiales[1]. El desarrollo del algoritmo diseñado no puede comenzar hasta que el diseñador esté seguro de dos cosas. Primero, el problema entre manos debe comprenderse completamente. Segundo, se debe asumir una configuración inicial del equipo para conformar el procedimiento. A partir del enunciado del problema y de la disponibilidad de equipo se busca una solución y se forma un algoritmo. El algoritmo se enuncia mediante un número finito de pasos bien definidos.

En este capítulo se presenta la teoría relacionada con los controladores, ya que el sistema para el cual se va a diseñar el banco de memoria, pertenece a tal categoría. Una vez logrado ello, se plantean las necesidades requeridas y se propone el banco de memoria como una solución.

1.1 Diseño de lógica de control

La información binaria encontrada en un sistema digital[2] se almacena en un procesador o registros de memoria y puede se constituida por datos o información de

---

[1] Dispositivos electronicos específicos: memorias, compuertas AND, OR; decodificadores, multiplexores, resistencias, etc.

[2] Un sistema digital es una combinación de dispositivos (eléctricos, mecánicos, fotoeléctricos, etc.) ensamblados a fin de desempeñar ciertas funciones en las cuales las cantidades se representan en forma digital.

Algunos de los sistemas digitales más comunes son las computadoras digitales, las calculadoras digitales, los voltímetros digitales y la maquinaria controlada en forma numérica. En estos sistemas las cantidades eléctricas y mecánicas varían solamente en etapas discretas.

En términos generales, los sistemas digitales ofrecen las ventajas de programabilidad, mayor velocidad y exactitud, y la capacidad de memoria. Además los sistemas digitales son menos susceptibles que los sistemas analógicos a fluctuaciones en las características de los componentes del sistema.

control. Los datos son elementos discretos de información que se manipulan por microoperaciones. La información de control suministra señales de mandos para especificar la secuencia de microoperaciones. La lógica de diseño de un sistema digital es un proceso para deducir los circuitos digitales que realizan los procesamientos de datos y los circuitos digitales que suministran señales de control.

La sincronización de todos los registros en un sistema digital se controla por medio de un generador de pulsos de reloj maestro. Los pulsos de reloj se aplican a todos los registros en el sistema. Los pulsos continuos de reloj no cambian el estado de un registro a no ser que el registro se habilite por la señal de control. Las variables binarias, que controlan las variables de selección y las entradas de habilitación de los registros se generan en una unidad llamada de control, también denominada como **controlador**. Las salidas de la unidad de control seleccionan y habilitan la parte del procesador de datos del sistema y también determinan el siguiente estado de la unidad de control en sí misma.

La relación entre la unidad de control y el procesador de datos (o Unidad aritmética) en un sistema digital se muestra en la figura 1-1. La parte del procesador de datos puede ser una unidad procesadora de propósito general, o puede consistir de registros individuales y funciones digitales asociadas. El control inicia todas las microoperaciones en el procesamiento de datos. Es la lógica de control quien genera las señales para dar secuencia a las microoperaciones en un circuito secuencial cuyos estados internos indican las funciones de control del sistema. En un tiempo dado, el estado de control secuencial inicia un conjunto de microoperaciones preseleccionadas. El control secuencial pasa el siguiente estado o inicia otras microoperaciones dependiendo de las condiciones presentes y otras entradas. Así, el circuito digital actúa como la lógica de control, suministra una secuencia de tiempo de señales para iniciar las microoperaciones en la parte del procesador de datos del sistema.

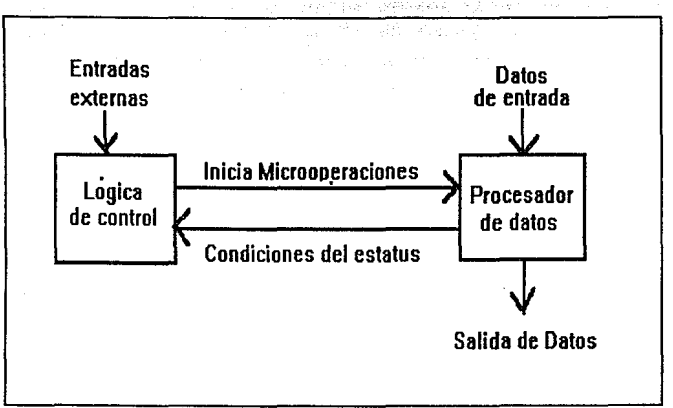


Figura 1-1. *Interacción entre el control y el procesador de datos.*

El diseño de un sistema digital que requiere una secuencia de control comienza con la suposición de la disponibilidad de variables de tiempo. Se diseña cada variable en la secuencia por medio de un estado y luego se forma un diagrama de estado o una representación equivalente para la transición entre estados. Paralelamente con el desarrollo de la secuencia de control se hace una lista de microoperaciones que se van a iniciar, para cada estado de control.

La secuencia de control y las relaciones de transferencia entre registros pueden deducirse directamente de la especificación en palabras del problema. Sin embargo es conveniente algunas veces usar una representación intermedia para describir la secuencia necesaria de operaciones del sistema. Dos representaciones, útiles en el diseño de sistemas digitales que necesitan control, son los diagramas de tiempo y los flujogramas.

Un diagrama de tiempo clarifica la secuencia de tiempo y otras relaciones entre las diferentes señales de control del sistema, ver figura 2-10. En un circuito secuencial con reloj, los pulsos de reloj sincronizan todas las operaciones incluyendo las señales de transición en las variables de control. En un sistema asincrónico una señal de transición en una variable de control puede causar un cambio a otra variable de control. Un diagrama de tiempo es muy útil en un control asincrónico ya que provee una representación ilustrativa de los cambios requeridos y las transiciones de todas las variables de control.

Un flujograma es una manera conveniente de especificar la secuencia de pasos de procedimiento y formas de decisión para un algoritmo, ver figura 2.21. Un flujograma para un algoritmo diseñado usaría normalmente los nombres de las variables de los registros definidos en la configuración del equipo. Este traslada un algoritmo de su enunciado en palabras a un diagrama de flujo de información que enumera la secuencia de operaciones de transferencia entre registros conjuntamente con las condiciones necesarias para su ejecución.

Un flujograma es un diagrama que consiste de bloques conectados por medio de líneas directas. Dentro de los bloques se especifican los pasos para configurar el algoritmo. Las líneas directas entre b'oques indican el camino que se va a tomar de un paso al siguiente. Se usan dos tipos mayores de bloques: un bloque rectangular indica un bloque de función dentro del cual se listan las microoperaciones. Un bloque en forma de rombo es un bloque de decisión dentro del cual se lista una condición actual dada. Un bloque de decisión tiene dos más caminos alternos y el camino que se toma depende del valor de la condición de estado especificada dentro del bloque.

Un flujograma es muy similar a un diagrama de estado. Cada bloque de función en el flujograma es equivalente a un estado en un diagrama de estado. El bloque de decisión en el flujograma es equivalente a la información binaria escrita por conducto de las líneas dirigidas que conectan dos estados en un diagrama de estado.

Como consecuencia, es conveniente algunas veces expresar un algoritmo por medio de un flujograma del cual se puede deducir el diagrama de estado de control.

Existen múltiples configuraciones para una unidad de control. Por ello existen varios procedimientos disponibles para el diseño de control lógico. El principal objetivo del diseño de lógica de control debe ser el desarrollo de un circuito que configure la secuencia de control deseada de una manera lógica y directa. Por ello los diseñadores con experiencia lógica usan métodos que pueden ser considerados como una extensión del método lógico secuencial clásico combinado con el método de transferencia entre registros. Podemos enumerar algunos métodos de diseño de organización de control.

1. Método de un flip-flop por estado.
2. Método del registro de secuencia y el decodificador.
3. Método del microprograma.

En las secciones siguientes se explica cada método en términos generales.

Cabe señalar que el objeto de presentar tales métodos ayudará al lector a ubicar el contexto y la importancia del sistema de desarrollo de microcontroladores Bit-Slice, además de que construye las bases para el diseño del banco de memoria y su interface a la computadora personal.

1.2 Método de un flip-flop por estado.

Este método usa un flip-flop por estado en el circuito secuencial de control. Solamente se pone a uno un flip-flop en un tiempo dado, los demás se ponen a cero. Se hace programar un solo bit de un flip-flop a otro, bajo el control de la lógica de decisión. En tal arreglo cada flip-flop representa un estado y se activa solamente cuando el bit de control se transfiere a éste.

Cabe señalar que este método no usa un número mínimo de flip-flops para el circuito secuencial. De hecho, éste usa un número máximo de flip-flops. Por ejemplo, un circuito secuencial con 12 estados requiere un mínimo de cuatro flip-flops pues $2^3 < 12 < 2^4$. Aun por medio de este método el circuito de control usa 12 flip-flops.

La ventaja de éste método es la simplicidad con la cual se diseña. Este tipo de controlador puede diseñarse por inspección a partir de un diagrama de estado que describe la secuencia de control. A primera vista, parece que este método aumentará el costo del sistema ya que se necesita un mayor número de flip-flops, pero, este método ofrece otras ventajas que no son aparentes a primera vista. Por ejemplo, éste método ofrece un ahorro de esfuerzos en el diseño, un aumento en la simplicidad operacional y una disminución potencial en los circuitos combinacionales requeridos para configurar el circuito secuencial completo.

Sin embargo, este método es poco práctico en la mayoría de los casos complejos debido al gran número de estados que el circuito de control puede tener. Además, los circuitos de control obtenidos por éste método requieren por lo general de un número excesivo de flip-flops y compuertas, lo cual implica el uso de tecnología SSI. Este tipo de configuración puede ser ineficiente con respecto al número de circuitos integrados (CI) que se usan y a la cantidad de alambres que deben ser interconectados. Por otro lado, el esfuerzo de minimizar el número de circuitos

tendería a producir una configuración irregular, lo cual haría difícil para cualquier persona diferente al diseñador, el reconocimiento de la secuencia de eventos por los cuales pasa el control. Como consecuencia podría ser difícil dar servicio y mantenimiento al equipo cuando esté en operación. Una desventaja adicional es que si se llegaran a necesitar alteraciones o modificaciones, los circuitos se deben alambrar de nuevo para cumplir con las nuevas especificaciones.

La figura 1-2 muestra la configuración de una lógica de control secuencial de cuatro estados, que usa cuatro flip-flops tipo D: un flip-flop por estado, t_i, $i = 0,1,2,3$. En cualquier intervalo de tiempo dado entre dos pulsos de reloj solamente un flip-flop es igual a 1, el resto será igual a 0. La transición del estado presente al siguiente es una función del presente t_i, que es 1 y de ciertas condiciones de entrada. El siguiente estado se manifiesta cuando el flip-flop anterior se borra y el nuevo se pone a uno. Cada una de las salidas del flip-flop se conecta a la sección de procesamiento de datos del sistema digital para iniciar ciertas microoperaciones. Las otras salidas de control mostradas en el diagrama son una función de las t y de las entradas externas. Estas salidas pueden también iniciar microoperaciones.

Si el circuito de control no necesita entradas externas para su funcionamiento, entonces, se reduce a un circuito de desplazamiento simple con un solo bit que se desplaza de una posición a la siguiente. Si la secuencia de control debe repetirse una y otra vez, el control se reduce a un contador de anillo.

Un contador de anillo es un registro de desplazamiento con la salida del último flip-flop conectado a la entrada del primer flip-flop. En un contador de anillo un solo bit se desplaza continuamente de una posición a la siguiente de manera circular.

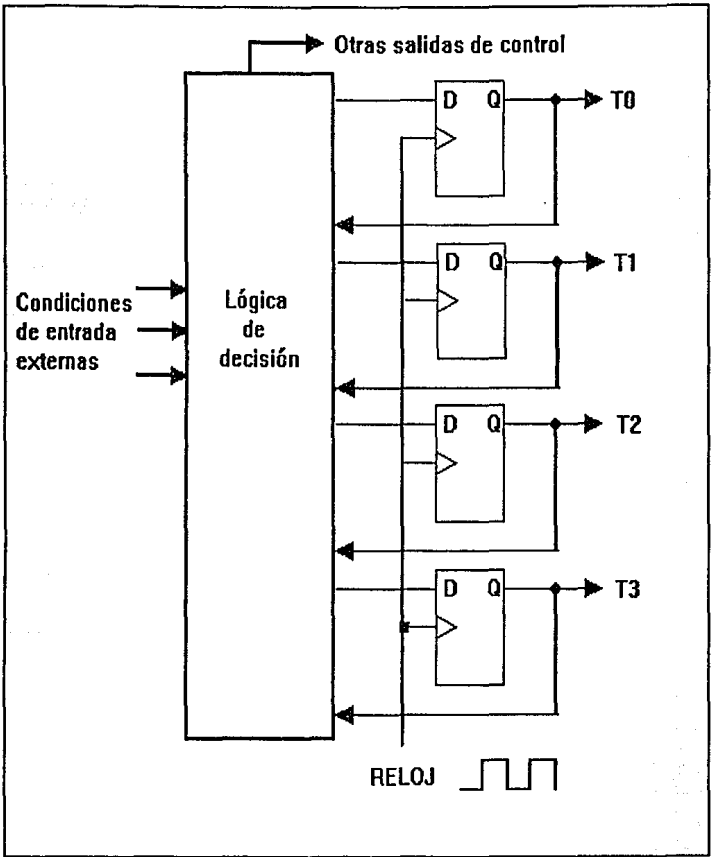


Figura 1-2. *Lógica de control con un flip-flop por estado.*

1.3 Registro de secuencia y método del decodificador.

Este método usa un registro para darle secuencia a los estados de control. El registro se decodifica para suministrar una salida por cada estado. El circuito tendrá 2^n estados y el decodificador 2^n salidas, para n flip-flops en el registro de secuencia. Por ejemplo, un registro de 4 bits puede estar en cualquiera de sus 16 estados. Un decodificador de 4 x 16 tendrá 16 salidas, una para cada estado del registro. Tanto el registro de secuencia como el decodificador son componentes MSI (Mediana Escala de Integración).

La figura 1-3 muestra la configuración de una lógica de control secuencial de cuatro estados. El registro de secuencia tiene dos flip-flops y el decodificador establece salidas separadas para cada estado en el registro. La transición al siguiente estado en el registro de secuencia es una función del estado presente y de las condiciones de entrada externas. Como salidas del decodificador están de alguna manera disponibles, es conveniente usarlas como variables de estado presente en vez de usar directamente las salidas de los flip-flops. Otras salidas que son función del estado presente y de las entradas externas pueden iniciar microoperaciones en adición a las salidas del decodificador.

Si el registro de control de la figura 1-3 no necesita entradas externas, el registro de secuencia se reduce a un contador que continuamente hace secuencias por los cuatro estados. Por ésta razón, el método es llamado algunas veces un método decodificador contador.

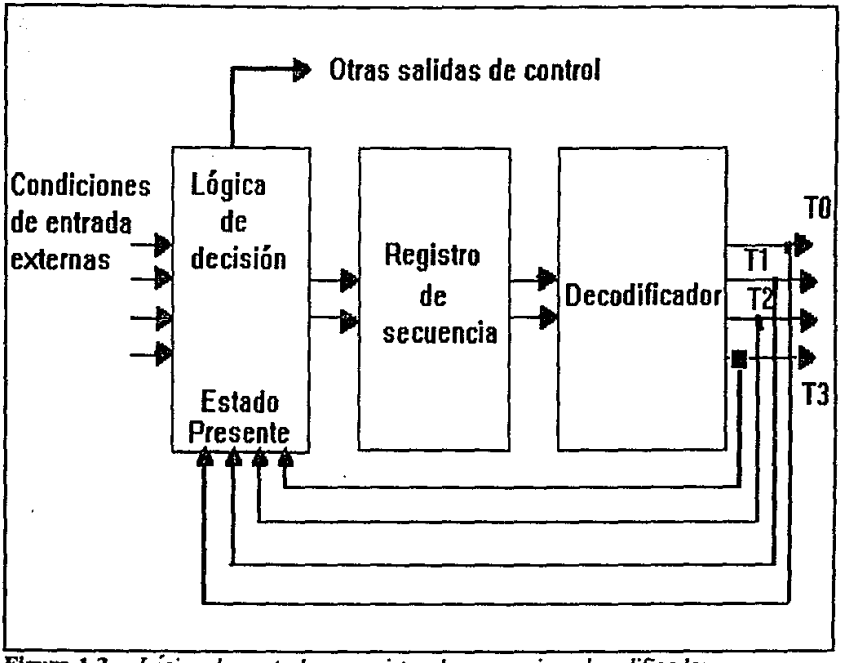


Figura 1-3. *Lógica de control con registro de secuencia y decodificador.*

1.4 Método del microprograma

El propósito de un controlador lógico es iniciar una serie de pasos secuenciales de microoperaciones. Durante cualquier tiempo dado se deben iniciar ciertas operaciones mientras que otras permanecen latentes. Así, las variables de control en un tiempo dado pueden ser representadas por una cadena de 1 ó 0, llamada palabra de control. Como tales, dichas palabras de control deben ser programadas para iniciar los diferentes componentes en el sistema de una manera organizada. Una unidad de control cuyas variables de control se almacenan en una memoria, se llama unidad de control microprogramada. Cada palabra de control de memoria se llama microinstrucción y una secuencia de microinstrucciones se llama microprograma. Cuando se requieren pocas alteraciones del microprograma, la memoria de control puede ser una Memoria de Solo Lectura (ROM). El uso del microprograma comprende la ubicación de todas las variables de control en palabras de la ROM para usarlas por medio de las unidades de control a través de operaciones sucesivas de lecturas. El contenido de la palabra en la ROM en una dirección dada especifica las microoperaciones del sistema.

Un desarrollo más avanzado, conocido como microprogramación dinámica permite cargar inicialmente un microprograma a partir de una consola de computador o de una memoria auxiliar tal como un disco magnético. Las unidades de control que usan microprogramación dinámica emplean una memoria de control en la cual se puede escribir (WCM Writable Control Memory). Este tipo de memoria es usada para escribir o cambiar el microprograma, y posteriormente su uso es de solo lectura. Cuando una ROM o WCM se usan en una unidad de control se los trata como memoria de control.

La figura 1-4 ilustra la configuración general de la unidad de control utilizando el método del microprograma. La memoria de control se asume como una ROM dentro de la cual se almacena permanentemente toda la información de control. El

registro de control de las direcciones de memoria especifica la palabra de control leída de la memoria de control. Se debe tener en cuenta que una ROM opera como un circuito combinacional con el valor de la dirección como entrada y la palabra correspondiente como salida. El contenido de la palabra especificada permanece en los alambres de salida por el tiempo que el valor de la dirección permanece en el registro de dirección. No se necesita señal de lectura como en una memoria de acceso aleatorio (RAM). Una palabra que sale de la ROM debe transferirse al registro separador, si el registro de direcciones cambia mientras que la palabra de ROM esté aún en uso. Si pueden ocurrir simultáneamente un cambio en dirección y una palabra de ROM no es necesario un registro separador.

La palabra leída de una memoria de control representa una microinstrucción. La microinstrucción especifica una o más microoperaciones para los componentes del sistema. Una vez que se ejecuten las operaciones, la unidad de control debe determinar la siguiente dirección. La ubicación de la siguiente microinstrucción podría ser la siguiente en secuencia o podría estar ubicada en otro lugar en la memoria de control. Por esta razón es necesario usar algunos bits de la microinstrucción para controlar la generación de la dirección para la siguiente microinstrucción. La siguiente dirección puede ser también una función de las condiciones externas de entrada. Mientras se ejecutan las microoperaciones, la siguiente dirección es computada en el circuito generador de la siguiente instrucción y luego transferida (con el siguiente pulso de reloj) al registro de control de direcciones para leer la siguiente microinstrucción. La construcción detallada del generador de la siguiente dirección depende de la aplicación particular.

En una situación práctica, la organización de los dispositivos de una unidad de control del microprograma debe tener una configuración de propósito general para adaptarse a una gran cantidad de situaciones. Una unidad de control de microprograma debe tener una memoria de control suficiente como para almacenar microinstrucciones. Se debe hacer provisión suficiente como para incluir todas las variables de control posibles en el sistema y no solamente para controlar a la Unidad Lógica

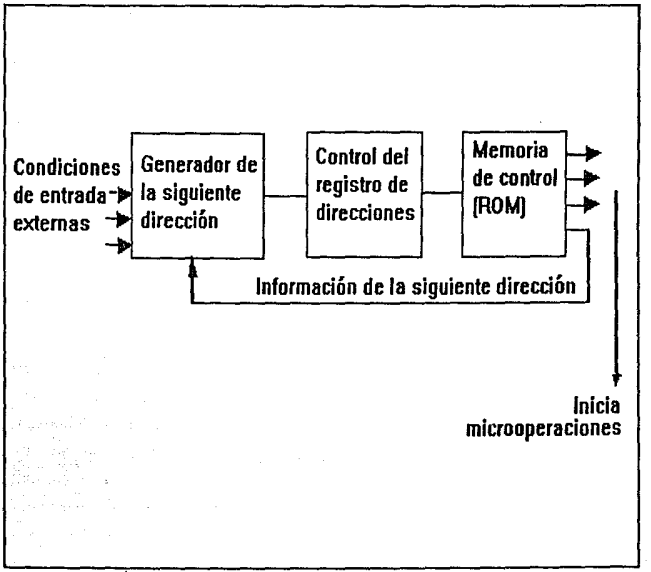


Figura 1-4. *Lógica de control con microprograma.*

Aritmética (ALU). Un multiplexor y los bits seleccionados deben incluir todos los demás bits de condición posibles que se quieran comprobar en el sistema. Se debe tener una provisión para aceptar una dirección externa para iniciar muchas operaciones.

La principal ventaja del control microprogramado es el hecho que una vez que se ha establecido una configuración de los circuitos no debe haber necesidad de

cambios posteriores de las conexiones entre los componentes. Si se quiere establecer una secuencia de control diferente para el sistema, todo lo que se necesita es especificar un conjunto diferente de microinstrucciones para la memoria de control. La configuración de los materiales no debe cambiar para las diferentes operaciones; el único cambio debe ser el microprograma que reside en la memoria de control.

Secuenciador del microprograma.

Una unidad de control de microprograma debe visualizarse como compuesta de dos partes: la memoria de control que almacena las microinstrucciones y los circuitos asociados que controlan la generación de la siguiente dirección. La parte de generación de dirección se llama algunas veces secuenciador de microprograma en vista de que da la secuencia de las microinstrucciones en la memoria de control. Un secuenciador de microprograma puede construirse con circuitos MSI para adaptarse a una aplicación particular. Sin embargo, de la misma manera que se encuentran disponibles en CI las unidades procesadoras para propósitos generales, se encuentran los CI para los secuenciadores para propósitos generales adecuados para la construcción de las unidades de control del microprograma. Para garantizar un amplio rango de uso, un secuenciador CI debe suministrar una organización interna que pueda adaptarse a un amplio rango de aplicaciones, en las cuales la modularidad es una característica esencial.

Un secuenciador de microprograma unido a la memoria de control inspecciona ciertos bits de la microinstrucción, de los cuales se determina la siguiente dirección para el control de la memoria. Un secuenciador típico presenta las siguientes características de secuenciamiento de direcciones:

1. Incrementa la dirección presente para la memoria de control.
2. Se ramifica a una dirección como se especifica en el campo de dirección de la microinstrucción.

3. Se ramifica a una dirección dada, si el bit de condición especificado es igual a 1.
4. Transfiere el control a una nueva dirección de la manera especificada por una fuente externa.
5. Tiene la facilidad para hacer subrutinas con llamadas y retornos.

En la mayoría de los casos se leen las microinstrucciones de la memoria de control en sucesión. Este tipo de secuencia puede lograrse fácilmente incrementando el registro de dirección de la memoria de control. En algunos formatos de microinstrucción, cada una de ellas contiene un campo de dirección aún para direcciones secuenciales. Esto elimina la necesidad de incrementar el registro de dirección de la memoria de control porque el campo de dirección disponible en cada microinstrucción especifica de la dirección siguiente microinstrucción. En cada caso, se debe dejar la alternativa de ramificarse a una dirección que esté por fuera de la secuencia normal.

El control debe transferirse de vez en cuando a una microinstrucción no secuencial, de manera que el secuenciador debe suministrar la capacidad de ramificarse a cualquiera de las dos direcciones dependiendo de si el bit de condición es 1 o 0. La manera más sencilla de lograr lo anterior es ramificándose a la dirección especificada por el campo de dirección de la microinstrucción si el bit de condición especificado es igual a 1 o si no pasar a la siguiente dirección en secuencia si el bit de condición es igual a 0. Esta configuración requiere la capacidad de incrementar el registro de dirección.

El secuenciador transfiere una nueva dirección para que la memoria de control comience a ejecutar la nueva microoperación. La dirección externa transfiere el control a la primera microinstrucción en una rutina de microprograma que ejecuta la macrooperación especificada.

Las subrutinas son programas usados por otras rutinas para lograr una tarea dada. Las subrutinas pueden llamarse desde cualquier punto dentro del cuerpo principal del microprograma. Frecuentemente muchos microprogramas contienen secciones idénticas de código. Las microinstrucciones pueden conservarse usando subrutinas que usan secciones comunes de microcódigo.

Los microprogramas que usan subrutinas deben tener una provisión para almacenar direcciones de retorno durante una llamada de subrutina y restaurar la dirección durante una subrutina de retorno. Esto puede lograrse colocando la dirección de retorno en un registro especial y entonces convertirse en la fuente de dirección para alistar el registro de dirección para el regreso a la subrutina principal. La mejor manera de organizar un archivo de registro que almacene direcciones para llamadas de subrutinas y que regrese, es usar una pila[3] UEPS (Ultimo en Entrar Primero en Salir).

[3] Pila es una área de memoria reservada para operaciones internas.

1.5 Evolución en el diseño de controladores

"Si tomamos la unidad fundamental de diseño como una medida de 'generaciones' en ingeniería digital, podemos considerarnos parte de la cuarta generación"[4].

En la primera generación, los bloques base de construcción fueron componentes discretos, tales como tubos de vacío y, posteriormente, transistores. La segunda generación puede ser categorizada por el uso de compuertas en circuitos integrados como la unidad básica de diseño. En la tercera generación, el diseñador trabajó con bloques preconstruidos más poderosos, tales como registros, multiplexores y unidades lógicas y aritméticas. En la cuarta generación, las unidades de diseño crecieron hasta componentes individuales altamente sofisticados, tales como microprocesadores, controladores de video y convertidores analógico-digitales. Un componente reciente de la cuarta generación es la lógica bit-slice, un dispositivo digital poderoso y flexible.

Este es el objeto de éste apartado, pues constituye parte fundamental del Sistema de Controladores Microprogramados y este, el destino del banco de memoria y su interfase, productos de la presente tesis.

Una forma común de clasificar a los microprocesadores es por el número de bits con los que su ALU puede trabajar al mismo tiempo. En otras palabras, un microprocesador con una ALU de 4 bits es llamado un microprocesador de 4 bits, sin importar el numero de líneas de dirección o el número de líneas del bus de datos que tenga. El primer microprocesador fue el Intel 4004 producido en 1971 y

---

[4] Página 3-4.
Mick John / Brick James
Bit-Slice Microprocessor Design,
McGraw-Hill Book Company,
1980.

contenía 23,000 Transistores PMOS. El 4004 fue un dispositivo de 4 bits cuya finalidad era ser usado con otros componentes para hacer una calculadora. Sin embargo algunos diseñadores notaron que éste dispositivo podía usarse para reemplazar tarjetas enteras de circuitos lógicos combinacionales y secuenciales. También la posibilidad de cambiar la función de un sistema con solo modificar un conjunto de instrucciones sin tener que rediseñar el hardware, fue muy atractivo. Esos fueron los factores que impulsaron la evolución de los microprocesadores.

En 1972 Intel presentó el microprocesador 8008, el cual era capaz de trabajar con palabras de 8 bits. Sin embargo, el 8008 requería de 20 o más dispositivos adicionales para formar un CPU funcional. En 1974 Intel anunció el 8080, un chip que tenía un conjunto de instrucciones mayor que el 8008 y solo requería de dos dispositivos adicionales para formar un CPU funcional. Por otra parte, el 8080 incorporó el uso de transistores NMOS, por lo que operaba a mayor velocidad que el 8008. Por ello el 8080 es referido como un microprocesador de segunda generación.

Poco después de que Intel produjo el 8080, Motorola lanzó al mercado el MC6800, otro CPU de propósito general de 8 bits. El 6800 tuvo la ventaja de requerir una sola fuente de energía de +5 V, a diferencia del 8080 que requería fuentes de energía de -5 V, +5 V y +12 V. Por varios años ambos microprocesadores fueron comercializados ampliamente como los líderes de 8 bits. Algunos de sus competidores fueron el MOS Technology 6502 usado como el CPU en la microcomputadora Apple II, y el Zilog Z80 usado como el CPU en la microcomputadora Radio Shack TRS-80.

Como los diseñadores encontraron más y más aplicaciones para los microprocesadores, presionaron a los industriales del silicio para desarrollar dispositivos con arquitecturas y características optimizadas para ejecutar ciertos tipos de tareas. En respuesta a las necesidades expresadas, los microprocesadores han seguido tres principales direcciones durante los últimos 10 años.

Controladores dedicados.

Una dirección ha sido hacia los controladores dedicados. Estos dispositivos son usados para controlar máquinas "inteligentes" tales como hornos de microondas, lavadoras de ropa automáticas, máquinas de coser, sistemas de ignición para autos y tornos mecánicos. Texas Instruments ha producido millones de microprocesadores de la familia TMS-1000 de 4 bits para éste tipo de aplicaciones. En 1976 Intel presentó el 8048, conteiendo un CPU de 8 bits, RAM, ROM y algunos puertos de Entrada/ Salida en un empaque de 40 patillas. Otros manufactureros han seguido la misma tendencia con productos similares. Estos dispositivos son frecuentemente conocidos como microcontroladores. Algunos de los dispositivos actualmente disponibles en esta categoría, son el Intel 8051 y el Motorola MC6801, el cual contiene contadores programables, un puerto serial (UART), así como un CPU, ROM, RAM y puertos paralelos de Entrada/ Salida. Un controlador más reciente es el Intel 8096, que contiene un CPU de 16 bits, ROM, RAM, un UART, puertos, temporizadores y un convertidor analógico - digital de 10 bits.

CPU's de propósito general.

La segunda dirección principal de la evolución del microprocesador ha sido en torno de los CPU's de propósito general, los cuales dan a una microcomputadora la mayor parte, o toda la potencia de cómputo que tenían las primeras minicomputadoras. Después de que Motorola presentó el MC6800, Intel produjo el 8085, una versión mejorada del 8080, requiriendo únicamente una fuente de +5 V. Motorola produjo entonces el MC6809, el cual tiene algunas instrucciones de 16 bits, sin dejar de ser aún un procesador de 8 bits. En 1978 Intel dió un paso muy importante con la presentación del 8086, un procesador de 16 bits. Algunos de los microprocesadores de 16 bits, tales como el National PACE y la familia 9900 de Texas Instruments, ya estaban disponibles previamente, pero el mercado industrial mundial no estaba preparado. Poco después de que Intel lanzó el 8086, Motorola desa-

rrolló su modelo MC68000 de 16 bits. El 8086 y el 68000 trabajan directamente con palabras de 16 bits, en lugar de 8 bits, pueden direccionar un millón o más bytes de memoria a diferencia de los 64 Kb direccionables por los procesadores de 8 bits. Además estos procesadores de 16 bits poseen en una sola instrucción funciones que requieren una larga secuencia de instrucciones en los procesadores de 8 bits.

La evolución a lo largo de esta ruta ha continuado hasta los procesadores de 32 bits que trabajan con giga (10^9) bytes o tera (10^{12}) bytes de memoria. Ejemplos de estos dispositivos son el Intel 80386 y 80486, el Motorola MC68020, y el National 32032.

Procesadores Bit-Slice.

Una tercera dirección de la evolución del microprocesador ha sido la de los procesadores bit-slice. Para algunas aplicaciones, los CPU's de propósito general tales como el 8080 y el 6800, no son lo suficientemente rápidos o su conjunto de instrucciones no son apropiados. Por ello, varios fabricantes han producido dispositivos que puedan ser usados en la construcción de CPU's "a la medida de las necesidades". Un ejemplo es la familia 2900 de Advanced Micro Devices. Estos dispositivos incluyen ALU's de 4 bits, multiplexores, secuenciadores y otras partes necesarias en la construcción personalizada[5] de un CPU. El término 'slice' proviene del hecho de que estos dispositivos pueden ser conectados en paralelo para trabajar con palabras de 8, 16 o de 32 bits. En otras palabras, un diseñador puede adicionar los chips que necesite para una aplicación en particular. Pero el diseñador no solo posee la libertad de optimizar el hardware del CPU, sino que también ajusta el conjunto de instrucciones a sus necesidades, usando microcódigo.

[5] Orientada a una aplicación muy particular.

Bit-slices contra microprocesadores

Existe una fuente de confusión acerca de las diferencias y relaciones entre los dispositivos bit-slice y los microprocesadores de propósito general. Las preguntas más comunes son: ¿Son los dispositivos bit-slice CPU's o miembros de la familia de CPU's?, ¿Los CPU's son construidos a partir de bit-slices?, ¿Son los bit-slices utilizables únicamente para aplicaciones de computadora?

Un hecho real entre los dispositivos bit-slices y los CPU's de propósito general, es que ambos son modernos componentes LSI (Large Scale of Integration).

Un CPU de propósito general es un dispositivo LSI que, en el mayor sentido de la palabra, es una computadora de programa-almacenado. En el chip están una Unidad Lógica Aritmética (ALU), una variedad de registros, una red de lógica secuencial para la decodificación y control de instrucciones, y frecuentemente una pequeña cantidad de memoria. Tal microprocesador tiene una arquitectura fija para su conjunto de instrucciones. La mayoría de los microprocesadores de uso general son construidos de semiconductores metal-óxido (MOS). En breve, un CPU de propósito general esta diseñado específicamente para ser parte de una computadora.

Un bit-slice, sin embargo, tiene poco en común con el CPU antes descrito, pues no es una computadora, porque no ejecuta programas de programa-almacenado, a pesar de que puede ser usado como un componente en un computador. A diferencia del microprocesador, en un dispositivo bit-slice no tiene sentido hablar de una arquitectura de conjunto de instrucciones. Sin embargo, si buscamos analogías, podríamos hablar de su arquitectura desde la definición de sus señales de control. Es cierto que, en un diseño particular, un dispositivo bit-slice puede responder a una particular arquitectura de microinstrucción, pero esto es una característica de ese particular diseño, no del dispositivo bit-slice. Por último, la mayoría de los dis-

positivos bit-slice son construidos con tecnología bipolar, de mayor velocidad que los MOS.

Para contestar a la primer pregunta anterior, podemos decir que nombrar a un bit slice como CPU o como un dispositivo en la familia de CPU's, es tan exacto como llamar microprocesador a una Unidad Lógica Aritmética.

La segunda pregunta, los CPU's no son construidos de bit slices. Por último, el rango de aplicaciones para bit slices es usualmente muy diferente al de los CPU's de propósito general. Bit slices pueden ser usados para construir procesadores, pero los CPU son procesadores. Los bit slices son dispositivos más rápidos, lo que significa que pueden ser usados como componentes de computadoras de alta velocidad. Los bit slices no solo son usados en la construcción de CPU's o procesadores de instrucciones. Ellos tienen aplicaciones en el diseño de canales de entrada/salida, controladores de disco, así como en otras aplicaciones digitales no basadas en computadora, tales como en sistemas de control de temperatura en hornos industriales.

Ahora que conocemos un poco sobre los dispositivos bit-slice podemos apreciar que ellos, al ofrecer una arquitectura abierta, permiten una gran libertad de diseño. Por ello es que fueron seleccionados para el sistema de desarrollo de controladores microprogramados Bit-Slice.

1.6 Sistema de Controladores Microprogramados

El Sistema de Controladores Microprogramados es un proyecto que tiene por objeto probar, experimentar y encontrar aplicaciones a los circuitos secuenciadores bit-slice AMD 2909 y 2910.

Dicho sistema fue desarrollado durante los años 1989 a 1990, en la ENEP Aragón, por iniciativa del Ing, David Jaime González Maxinez.

A continuación vamos a ver de que elementos se compone el Sistema de Controladores Microprogramados. La figura 1-5 muestra el diagrama de bloques funcional del Sistema de Desarrollo de Controladores Microprogramados.

La figura 1-6 esquematiza la construcción física del Sistema de Desarrollo. En él se puede apreciar los circuitos secuenciadores bit-slice 2909 y 2910. Se trata de 3 circuitos AMD 2909 y un circuito AMD 2910, conectados en cascada. También se encuentran 4 circuitos multiplexores, MUX 0 a MUX 3. Un conjunto de flip-flops son complementarios para la interconexión de dispositivos. El conjunto de resistencias y diodos emisores de luz son una herramienta cuya finalidad es ayudar al diseñador a detectar los estados del Sistema.

La figura 1-7 presenta la ubicación de la memoria EPROM, en relación al Sistema de desarrollo de Controladores Microprogramados.

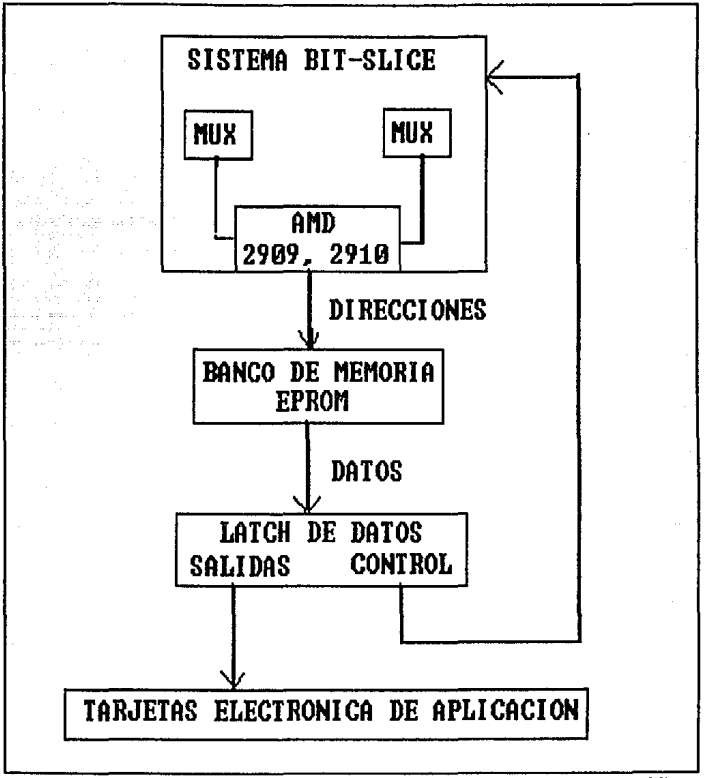


Figura 1-5. *Diagrama de bloques del Sistema de Desarrollo de Controladores Microprogramados.*

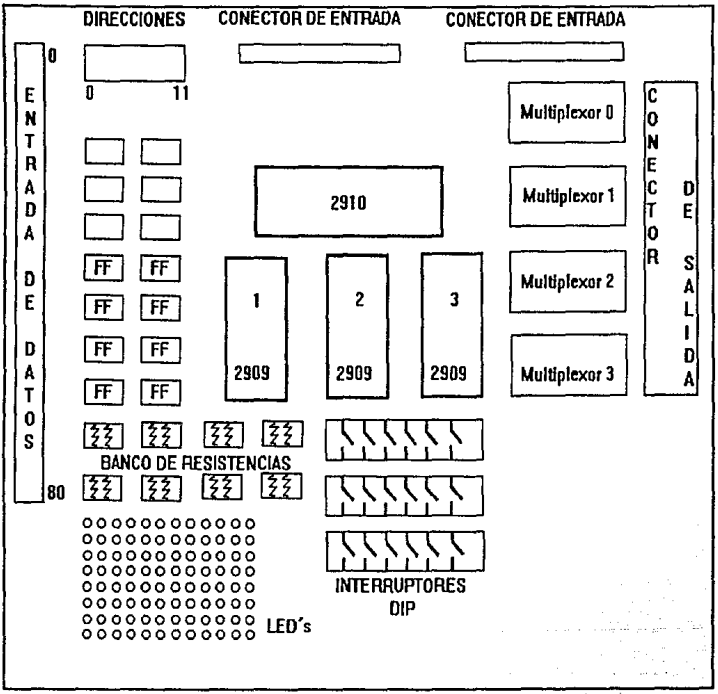


Figura 1-6. *Tarjeta terminal del Sistema Bit-Slice.*

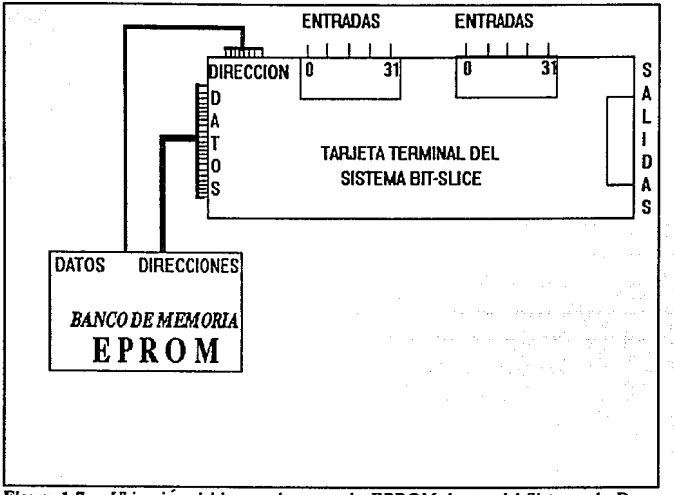


Figura 1-7. *Ubicación del banco de memoria EPROM dentro del Sistema de Desarrollo de Controladores Microprogramados.*

Identificación del problema.

La memoria actualmente usada por el Sistema de Desarrollo, es del tipo EPROM[6], la cual es una memoria que tiene posibilidades de ser reprogramada, es decir, de modificar su contenido, y por ello es que se seleccionó como parte del

[6] Memoria eléctricamente programable por el usuario; se "limpia" exponiendola a luz ultravioleta.

sistema experimental. Sin embargo, presenta severos obstáculos de eficiencia, que a continuación se detallan.

Dado que el Sistema de Desarrollo, es un proyecto experimental, la cantidad de veces que usualmente se requiere modificar el contenido de la memoria, (esto es, el microprograma) es muy elevado. Luego, considerando que el tiempo necesario para borrar una memoria EPROM asciende a unos 20 minutos, entonces hacer uso del sistema de desarrollo implica una gran cantidad de tiempo empleado en el proceso de volver a experimentar una modificación.

Si además consideramos que el lugar más cercano a la ENEP Aragón para efectuar el proceso de borrado y programación de una memoria EPROM, se encuentra en las Instalaciones del Instituto Politécnico Nacional, en Zacatenco, entonces el tiempo requerido para efectuar las modificaciones experimentales se prolonga mucho.

Ante esto, se puede optar por la adquisición un equipo para grabado de Memorias EPROM, así como del equipo de rayos ultravioleta. La limitante en este sentido es la falta de recursos Económicos, pues un equipo económico con estos dos elementos básicos asciende a $2,000 dólares (diciembre de 1989).

Y ¿Cuál sería el beneficio de tal acción? Indudablemente la reducción en tiempo sería considerable, pero el trabajo de desmontar los circuitos de memoria, introducirlos uno a uno en la cámara de rayos ultravioleta, programarlos uno a uno, y volver a montarlos en el Sistema de Desarrollo, puede convertirse en una molestia intolerable. Desde luego que se tendría la ventaja de usar tal equipo en otros proyectos experimentales, pero para el Sistema de Desarrollo, no parece la mejor solución.

Entonces, ¿existe una mejor solución?, ¿qué se requiere para la solución?, Veamos las necesidades.

Definición de las necesidades.

Se requiere diseñar la memoria de control del Sistema de Controladores Microprogramados, la cual debe cumplir con las siguientes características técnicas:

1. Debe poseer un tamaño de palabra de 80 bits
2. El numero de direcciones debe ser de 1024.
3. El tiempo de acceso debe ser menor a 210 ns.

Propuesta

La solución que se propone a la problemática anterior consiste en el empleo de memoria de acceso aleatorio (RAM) en sustitución de la memoria EPROM y que se programe a través de la Computadora Personal.

La propuesta mencionada se encuentra representada en la figura 1-8. Como puede observarse, el Banco de Memoria EPROM ha sido substituido por un Banco RAM, sin que el Sistema de desarrollo haya sido afectado en sus funciones esenciales.

La ventaja de esta propuesta radica en el hecho de que los cambios requeridos por la memoria de control se pueden llevar en forma inmediata por un procedimiento muy sencillo, dadas las facilidades que ofrece la Computadora Personal, eliminando con esto los problemas de tiempo de desmontado, borrado, y montaje de circuitos de memoria.

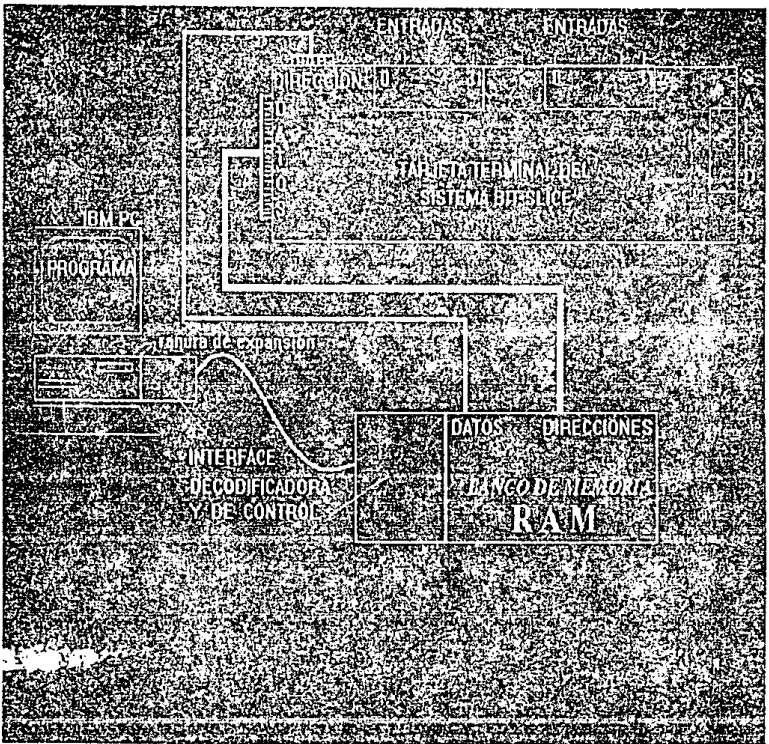


Figura 1-8. *Ubicación del Banco de Memoria y la Computadora Personal dentro del Sistema de Desarrollo.*

Ampliación de las necesidades

Para que la solución propuesta sea eficiente, la memoria diseñada debe quedar interfazada (interconectada) a una Computadora Personal, por lo cual se requiere elaborar un programa en lenguaje de alto nivel que :

1. Permita la captura de información en hexadecimal (Tal información corresponde a las microinstrucciones).
2. Proporcione la facultad de almacenar la información en disco.
3. Conceda capturar los datos por rangos de direcciones, hasta un máximo de 16.
4. Admita seleccionar el tamaño de la palabra entre las siguientes tamaños: 4, 8, 16, 32, 64 u 80 bits (número de bits por dirección).
5. Permita cargar, revisar y modificar la información contenida en disco.
6. Llevar a cabo la transferencia de datos entre la Computadora Personal y el Banco de Memoria.
7. Permita imprimir formateado el contenido de un archivo de datos, seleccionando rangos de direcciones (máximo 16), y el tamaño de la palabra.

Hasta ahora se ha descrito que es un controlador, los tipos de controladores y sus características. Se ha presentado que es un circuito bit-slice, su relación con otros circuitos LSI, sus alcances y limitaciones. La estructura del Sistema de Controladores microprogramados así como las necesidades del mismo fueron manifestadas. Finalmente, se dió una propuesta que simplifica sustancialmente el uso del Sistema de Desarrollo.

En los capítulos siguientes se dan los pasos tendientes a diseñar la solución propuesta.

2

Estudio Interno de la Computadora Personal

Introducción

La propuesta hecha en el capítulo anterior, está basada en la Computadora Personal de IBM. Para poder llegar a una solución adecuada, debemos antes saber qué es y como funciona esta máquina, comúnmente llamada PC. Este es el objeto del presente capítulo. Vamos entonces, a penetrar en los secretos de la PC para comprender la estructura fundamental de esta poderosa herramienta, a fin de lograr nuestro propósito.

2.1 Una microcomputadora de propósito general.

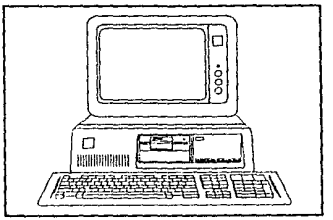


Figura 2-1 *Computadora Personal IBM.*

La computadora personal de IBM, mostrada en la figura 2,1, fue la primer microcomputadora vendida en grandes cantidades. Desde su introducción a mediados de 1981, la aceptación de la PC en el mercado ha crecido a pasos agigantados de tal manera que se ha convertido en el líder de la arquitectura de la computación personal. La clave de su éxito fue, y continúa siendo, la enorme cantidad de software de aplicación disponible para la máquina. Esto incluye aplicaciones de negocios, diseño de ingeniería, lenguajes de programación, programas educacionales, juegos, y aún sistemas operativos.

La IBM PC es un ejemplo de una computadora digital de propósito general. El término propósito general significa que fue diseñada para "correr" programas de una variedad muy amplia de aplicaciones, por computadora digital se intenta decir que es usada en el proceso de datos o información binaria. Por ejemplo, un usuario puede usar la PC con un paquete estándar de contabilidad o de control de inventarios. En este tipo de aplicación, la tarea principal de la microcomputadora es analizar y procesar grandes cantidades de datos, conocidos como base de datos. Otro usuario podría estar corriendo un paquete de software de proceso de palabras. Este es un ejemplo de una intensa actividad de entrada / salida. El usuario introduce información de texto, la cual tiene que ser reconocida por la microcomputadora y entonces verificar si está bien escrita, presentarla en pantalla con los atributos de impresión requeridos, checar si se encuentra al final de la línea o de la página, dividir palabras, grabar en disco, enviar la información a una impresora, etc. Un tercer ejemplo es donde un

programador usa un lenguaje, tal como FORTRAN, para escribir programas para aplicaciones científicas. Aquí la función fundamental de la computadora es resolver problemas matemáticos complejos. El punto que se trata de señalar es que el uso de la PC es el mismo en cada una de la aplicaciones; la única diferencia es el software que la computadora está procesando.

Ya se ha mencionado que la IBM PC es una microcomputadora. Vamos ahora a ver brevemente que es una microcomputadora y cuales son sus diferencias respecto a otras clase de computadoras.

La evolución del mercado de las computadoras durante los últimos 25 años ha sido de grandes computadoras mainframe hacia minicomputadoras y ahora a pequeñas microcomputadoras. Estas tres clases de computadoras no se reemplazan unas con otras. Todas ellas coexisten en el mercado. Actualmente, los usuarios de computadoras tienen la oportunidad de seleccionar la computadora que mejor se adapte a sus necesidades. Por ejemplo, una gran universidad o institución seleccionaría una computadora mainframe para su centro de proceso de datos. Sin embargo, un departamento de la misma universidad o en un negocio podría seleccionarse una minicomputadora para cubrir una necesidad de múltiples usuarios tal como el desarrollo de software de aplicación. Por otra parte, gerentes pueden seleccionar una microcomputadora, tal como una PC, para satisfacer necesidades tales como el procesamiento de documentos y el manejo de bases de datos.

A lo largo de la ruta de evolución de mainframes a microcomputadoras, los conceptos básicos de la arquitectura de computadoras no ha cambiado. Justo como un mainframe o una computadora, la microcomputadora es un sistema electrónico para el procesamiento de datos de propósito general usada en múltiples aplicaciones. La principal diferencia es que las microcomputadoras, tal como la IBM PC, emplea la tecnología de circuitos más novedosa: la muy alta escala de integración (VLSI) a fin de implantar un sistema de cómputo pequeño pero con suficiente capacidad y con un costo mucho menor que una minicomputadora. Y el costo precisamente, se ha

reducido debido a que la producción de tales circuitos se ha hecho en forma masiva. Sin embargo, las microcomputadoras, las cuales son diseñadas para un alto rendimiento, están capturando parte del mercado que tradicionalmente era cubierto con minicomputadoras de bajo rendimiento.

2.2 Conociendo la Computadora Personal de IBM.

Los componentes básicos de la computadora Personal de IBM son la Unidad del Sistema, el teclado y un monitor.

Ellos pueden conectarse rápidamente y estar listos para ser utilizados en unos cuantos minutos. La Unidad del Sistema contiene un microprocesador Intel de 16 bits, la memoria, y una o dos unidades de disco. El sistema es alimentado por una fuente de poder que se conecta directamente a un contacto eléctrico aterrizado estándar de 110 V ca.

La Unidad del Sistema

La Unidad del Sistema contiene el microprocesador intel de 16 bits, 8088, la memoria de sólo lectura (ROM), la memoria de acceso aleatorio (RAM), la fuente de poder, y una bocina de audio para aplicaciones musicales y de alerta, más ranuras de expansión, a fin de permitir un acceso fácil al Sistema. Una o dos unidades de disco pueden alojarse dentro del mismo gabinete.

La tarjeta del Procesador.

El corazón del Sistema es la tarjeta del procesador, comúnmente llamada tarjeta madre. Contiene muchos de los dispositivos electrónicos esenciales del sistema, incluyendo el microprocesador 8088, hasta 640 Kb (Kilobytes) de memoria de acceso

aleatorio (RAM), 40 Kb de memoria de sólo lectura (ROM), y adaptadores de entrada / salida para el teclado y la bocina de audio. Las principal característica de la tarjeta madre es el sistema de ranuras de expansión, que permite adicionar dispositivos al sistema. Y es precisamente esta cualidad que vamos a aprovechar para lograr la interfase entre la memoria de control y la Computadora Personal.

La fuente de poder

La fuente de poder provee potencia eléctrica de corriente directa (dc) a los elementos del Sistema. Esta fuente de poder provee cuatro niveles de voltaje : ± 5 y ± 12 volts. Estos niveles de energía se hallan disponibles en las ranuras de expansión y se pueden usar en las tarjetas que se agreguen. Se dispone de un total de 63.5 watts de energía dc para el Sistema y sus dispositivos.

Las unidades de disco

El sistema puede almacenar hasta dos unidades de disco flexible. Actualmente están disponibles dos tamaños. Uno es de 3 ½ y el otro es de 5 ¼ pulgadas de diámetro. Ver figuras 2-2 y 2-3, respectivamente. Por otro lado, las capacidades de almacenamiento pueden variar de 360 Kbytes a 720 Kbytes respectivamente, con formatos de doble cara y doble densidad. Para poder conectar a las unidades de disco, es necesario haber instalado una tarjeta controladora de discos en una ranura de expansión. La energía que consumen las unidades de disco es sumi-

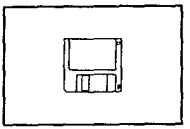


Figura 2-2 *Disco de 3.5 pulgadas.*

Figura 2-3 *Disco de 5.25 pulgadas.*

nistrada por la misma fuente de poder del sistema.

La bocina

La bocina o altavoz del sistema es controlada por programa y típicamente es usada como alarma o como un dispositivo de juegos y programas de aplicación que requieren una salida de audio.

El teclado

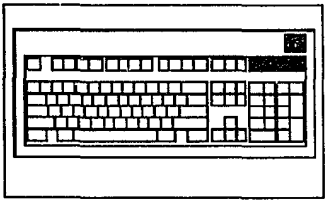


Figura 2-4 *Teclado expandido de 101 teclas.*

El teclado de la IBM PC XT actualmente contiene 101 o 102 teclas, con una distribución semejante a de las máquinas de escribir estándar. Ver figura 2-4. Existen teclas especiales de funciones programadas, las cuales son definidas por el sistema o por el programa de aplicación siendo ejecutado por el sistema. Además se dispone de un doble teclado numérico, cuya finalidad es facilitar el uso del teclado en aplicaciones contables. El teclado se encuentra interconectado al Sistema, a través de una interfase de cuatro cables que contienen tanto señales como energía. El teclado envía un código de rastreo único de 8 bits cada vez que se oprime una tecla y también cuando se libera. Si se mantiene oprimida una tecla, el teclado envía un código de rastreo al momento de oprimirla, y espera para luego enviar el mismo código a una velocidad establecida hasta que la tecla se libera. Al momento que la tecla deja de ser presionada, se envía un código diferente. Esta función permite a las aplicaciones definir el modo de operación del teclado.

El monitor

En la actualidad existen múltiples monitores ofrecidos para la IBM PC. Ellos van desde dispositivos monocromáticos de baja resolución color verde, ámbar o blanco, hasta los sofisticados monitores a color de alta resolución que tienen una definición de 1600 por 1200 pixeles, o mayor aún. Para conectar un monitor debe uno proveer a una de las ranuras de expansión de una tarjeta que controle el tipo de dispositivo deseado.

Las señales de control que suministran las tarjetas controladoras de video son las de sincronía vertical, horizontal, de video e intensidad.

Las tarjetas de funciones adicionales

A la fecha, múltiples proveedores de accesorios para la IBM PC, ofrecen tarjetas que se conectan en las ranuras de expansión para comunicarse con dispositivos que incrementan la utilidad de la Computadora Personal. Veamos algunas de ellas.

TARJETA PARA CONECTAR UNA IMPRESORA PARALELA

Esta tarjeta permite conectar la mayoría de las impresoras que usan la interfase Centronics, la cual se ha convertido en un estándar de la industria. Es capaz de soportar transferencia programada de datos y transferencia manejada por interrupciones.

Tarjeta de Puerto Serial

Esta tarjeta tiene un puerto serial Asíncrono con una interfase eléctrica que permite conexiones con un Modem para transmisión de datos sobre líneas telefónicas. Tiene una velocidad programable que va desde 50 a 19200 baudios (bits/seg) y con tamaños de palabra de 5 a 8 bits. También se permite programar los bits de paro en 1, 1 ½ o 2. La paridad puede ser seleccionada como par impar o no ser generada.

Tarjetas de expansión de Memoria

Una gran cantidad de tarjetas de expansión de memoria se encuentran en el mercado. Ellas son diseñadas para incrementar la cantidad de memoria principal del sistema hasta un límite de 640 Kbytes en una IBM PC XT.

Otros adaptadores y dispositivos

Debido a que existen muchos fabricantes que han desarrollado adaptadores e interfases, siempre es conveniente investigar si lo que deseamos ya se encuentra disponible en la industria. De ser así, también debemos investigar si es factible importarlo y el costo de tal operación. Para éste caso, en el que se requería una tarjeta que permitiera transferir datos de la memoria de la PC al banco de memoria a diseñar, se procedió a consultar con los distribuidores locales de accesorios de IBM PC, y se llegó a la conclusión de que no existía tarjeta alguna prediseñada que se ajustara a las necesidades requeridas.

2.3 La tarjeta del procesador

La tarjeta del procesador es el corazón de la PC. Identificar sus elementos y comprender sus funciones, es vital.

El procesador 8088

El 8088 es un microprocesador de 16 bits con un bus de memoria de 8 bits. Sus instrucciones pueden manipular datos de 16 bits, pero los datos e instrucciones son buscadas y escritas a memoria en 8 bits cada vez. El 8088 tiene la habilidad para direccionar hasta 1 megabyte de memoria, la cual puede contener datos o programas.

El 8088 soporta también funciones de interrupción y acceso directo a memoria, entre otras.

Los circuitos de reloj

En la arquitectura de la IBM PC XT, el 8088 opera a una velocidad de reloj de 4.77 MHz. Esto resulta en un período de operación de aproximadamente 210 nanosegundos. Como la mayoría de los ciclos del bus son de 4 ciclos de reloj, un ciclo típico de memoria es de 840 nanosegundos.

El microprocesador 8088 y muchos de las funciones de la tarjeta madre obtienen su sincronización mediante las señales que provee el oscilador del sistema. El circuito oscilador es manejado por un cristal de 14.31818 Mhz. EL chip 8284 divide esta frecuencia base por tres para dar la señal de 4.77 MHz. Posteriormente la señal anterior es dividida por un factor de cuatro para dar una señal de 1.19 MHz la cual es usada para manejar las entradas de reloj en los contadores temporizadores del sistema.

Bus del sistema

Los principales componentes funcionales de la tarjeta madre se hallan conectados al microprocesador 8088 a través del bus[1] del sistema. Este bus está hecho de varios tipos de señales: bus de datos, bus de direcciones, líneas de control, interrupción, temporización y acceso directo a memoria. El bus inicia en las terminales o pines del microprocesador. Localmente se encuentran conectados al bus los siguientes dispositivos: (1) un conector para un procesador auxiliar compatible con el intel 8087, (2) un controlador de interrupciones de 8 niveles, el circuito 8259A, (3) el controlador de bus 8288 y (4) un circuito reacondicionador de los niveles de energía del bus. Conectados al bus del sistema se hallan: (1) circuitos de soporte al microprocesador, (2) lógica decodificadora de direcciones de memoria y puertos de entrada / salida, (3) ROM, (4) RAM, (5) interruptores de configuración, (6) adaptadores integrados de entrada /salida y (7) las ranuras de expansión[2].

Memoria de Lectura (ROM)

La memoria ROM se encuentra decodificada en los 40 Kb de direcciones de memoria más altas. Por ello, no deben decodificarse las direcciones F6000 a FFFFF para aplicaciones particulares.

[1] Bus es una palabra del inglés, e indica el canal de comunicación que une diversos componentes dentro de una computadora. Generalmente se refiere al conjunto de conductores o pistas en una tarjeta de CI. El uso de ésta palabra es muy común en el lenguaje de computación del español.

[2] Traducción de la palabra inglesa ´slot´. También llamado Canal de Interfase E/S.

La memoria ROM contiene microcódigo soportando las siguientes funciones del sistema:

- Inicialización del sistema.
- Diagnósticos de encendido y revisión del sistema.
- Determinación de la configuración del sistema.
- Manejadores de dispositivos de entrada/salida, comúnmente llamados BIOS (Basic Input/Output System: Sistema Básico de Entrada/Salida).
- Carga del Sistema Operativo del disco.
- Carga de los patrones de los caracteres.

Todas las funciones anteriores residen en uno de los chips ROM de 8 Kb. Este ROM reside en el espacio de direcciones que inicia en la dirección hexadecimal FE000 y termina en FFFFF. Los listados de los programas para las funciones anteriores se pueden consultar en el Manual de Referencia Técnica de IBM.

Los restantes 32 Kb (opcionales) de ROM (en otros 4 chips de ROM) contienen el interpretador BASIC[3] de Microsoft.

Memoria Principal (RAM)

La memoria RAM del sistema reside en el espacio de direcciones del 8088 que inicia en la dirección hexadecimal 00000 y termina en 9FFFF, asumiendo que se encuentra instalada un total de 640 Kb de RAM.

El sistema de memoria opcionalmente puede ser de 9 bits de ancho; el noveno bit es de paridad.

---

[3]. Beginners All purpose Symbolic Instruction Code: Clave de instrucciones simbólicas de propósito general para principiantes.

La tarjeta del sistema usa chips de memoria dinámica con un tiempo de acceso de 250 nanosegundos. Estos chips requieren tres niveles de energía para operar: +5, -5 y +12 Volts.

La finalidad de la memoria RAM es almacenar temporalmente programas y datos para que el trabajo sea rápido, y una vez que se desee se almacenen en disco.

2.4 El microprocesador 8088

EL microprocesador 8088 es un derivado del microprocesador Intel 8086, cuya principal diferencia es el ancho del bus de datos. El 8086 tiene una ruta interna de datos y un ancho del bus de datos para memoria externa de 16 bits, por otro lado, el 8088 también tiene una ruta interna de datos de 16 bits, pero un bus de datos externos de sólo 8 bits. Debe señalarse que el código escrito para un sistema basado en un procesador o en otro correrá sin cambios en cualquiera de ellos. La diferencia se hará visible en velocidad al correr aplicaciones cuya naturaleza sea 8 o 16 bits. Es este caso uno será más rápido que el otro.

A continuación se presenta un resumen rápido de las características del microprocesador 8088:

- Arquitectura interna de 16 bits
- Soporte de 1 megabyte de memoria conectada.
- Aritmética de 8 y 16 bits sin signo, tanto en notación binaria como decimal, incluyendo multiplicación y división.
- 14 Registros de 16 bits.
- Capacidad de interrupciones enmascarable y no enmascarables.
- Capacidad de acceso directo a memoria
- Soporte de un coprocesador local

- Soporte de memoria mapeada.
- Operaciones de cadenas de caracteres.

La figura 2-5 muestra un diagrama de bloques funcional del CPU 8088. Se puede notar que algunos pines tienen dos definiciones. El 8088 tiene dos modos de operación que son seleccionados mediante la patilla 33 (+MN,-MX).Cuando esta terminal se mantiene en alto, el 8088 se encuentra en modo mínimo y sus patillas de interfase son compatibles con el microprocesador 8085. Cuando es seleccionado éste modo, no se requiere un chip controlador de bus, pero no es posible conectar un coprocesador. En el diseño de la controlador PC esta terminal se conecta a un nivel bajo, así que opera en el modo máximo.

Direccionamiento de la Memoria

Una de la características sobresalientes del microprocesador 8088 es su habilidad para direccionar más de 65,5336 bytes de datos que se pueden especificar con un campo de direcciones de 16 bits.

El 8088 tiene capacidad para manejar 20 bits en el direccionamiento de datos, lo cual le permite un tamaño físico de la memoria de 1,048,576 bytes. Pero como la mayoría de las instrucciones que hacen referencia a la memoria sólo permiten especificar y manipular un campo de direcciones de 16 bits, entonces, podría parecer como si, cuando mucho sólo 65,536 bytes de memoria pudieran ser usados.

Esto es particularmente cierto, en algún sentido, pues por ejemplo, los programadores están limitados a una región de datos de 65,536 bytes. Pero se tiene la capacidad de mover esa región a cualquier lugar en el espacio de los 1,048,576 bytes.

Esto se logra manejando los contenidos de un registro especial llamado un *registro de segmento*. El valor cargado en él es usado para localizar, en el espacio de

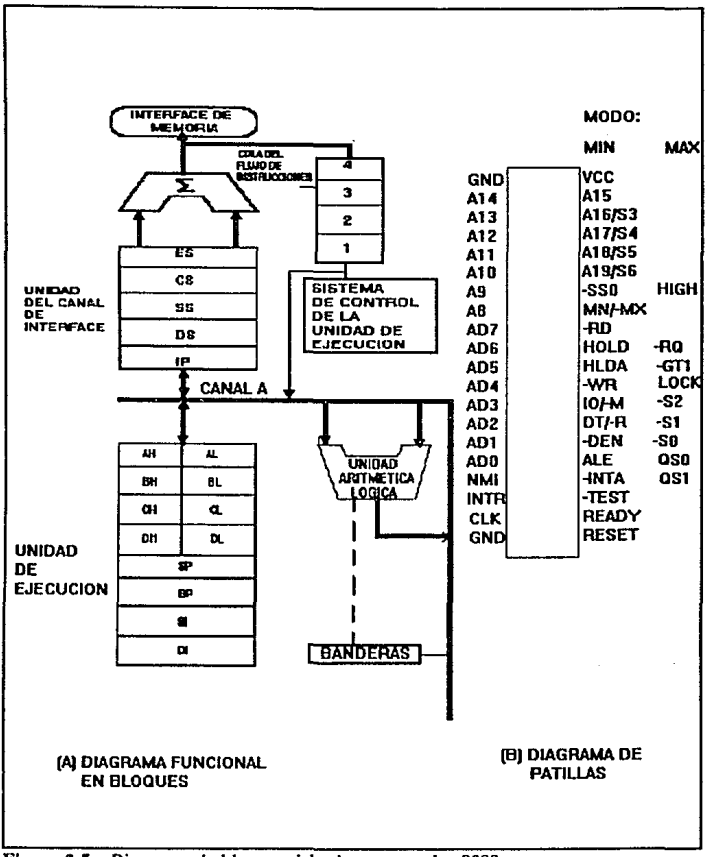


Figura 2-5 *Diagrama de bloques del microprocesador 8088.*

1 megabyte, la región de 64 Kb que las instrucciones del 8088 requieren para operar. Como este registro también tiene una longitud de 16 bits, por sí mismo tampoco puede definir el lugar deseado en el espacio de 1 megabyte. Para resolver éste problema, el registro de segmento de 16 bits sólo apunta a un inicio de un rango. Por ello, el registro de 16 bits puede especificar 65,536 regiones distintas. Es por esto que la dirección física se forma desplazando 4 bits a la izquierda los contenidos del registro de segmento y sumándolos a la dirección generada de 16 bits. La figura 2-6 ilustra la generación física de la dirección.

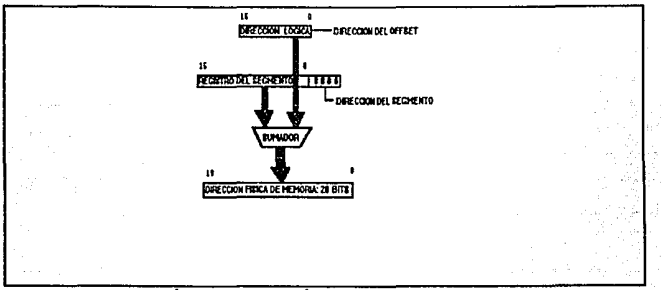


Figura 2-6 *Generación de una dirección de memoria en el 8088.*

En realidad, existen cuatro registros de segmentos. Uno es usado para la dirección del código, otro para la dirección de los datos referidos por el código y uno más es usado para los datos referidos en el código vía la pila[4], y el último es un segmento extra. El segmento extra, típicamente es usado en operaciones de movimiento donde la operación va a tomar lugar entre dos regiones diferentes de 64K

[4] Pila es la traducción de la palabra inglesa 'stack' y se refiere al área reservada de memoria para operaciones internas del microprocesador.

y existe un requerimiento tanto para el segmento fuente como para el segmento destino.

Estos cuatro registros de segmento, pueden señalar a 64 K regiones diferentes en el espacio de 1 megabyte. Por ello, una vez que los registros de segmento son fijados, las instrucciones de programa ven un espacio de 64K, ya sean datos, código, extras o stack. Si en un momento dado el programa necesita referirse fuera de estos espacios, primero debe manipular el registro de segmento apropiado. Debe señalarse que los segmentos pueden también traslaparse o apuntar todas al mismo espacio de 64 K.

El valor del segmento es frecuentemente conocido como la dirección base y una dirección dentro de un segmento es llamado un offset. Por ello, cualquier dirección en el espacio de un megabyte puede ser identificado especificando una base y un offset (compensador, balanceador). Debe notarse que diferentes combinaciones de base y offset pueden usarse para especificar una dirección de memoria.

Es usual escribir una dirección de memoria usando la nomenclatura, base-:offset, siempre que ambos sean dados como numeros hexadecimales de 4 posiciones. Por ejemplo, para definir la dirección absoluta ABC00 podemos darla como ABC0-:0000, A000:BC00.

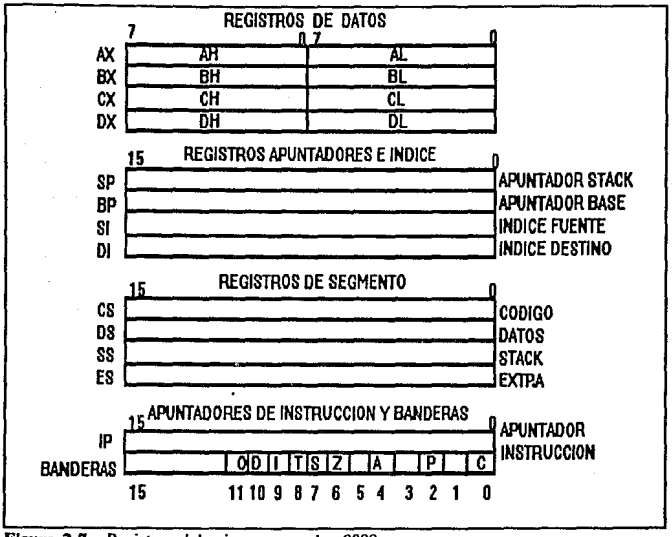


Figura 2-7 *Registros del microprocesador 8088.*

Registros del 8088

Como se mencionó antes, el 8088 tiene 14 registros de 16 bits. Estos registros pueden ser clasificados como sigue: un grupo de 4 registros de datos, un grupo de 4 registros de apuntadores y de índice, 4 registro de segmento, un registro apuntador de instrucción y un registro de banderas. La figura 2-7 es un diagrama de los registros del 8088.

Registros de datos

Estos cuatro registros de 16 bits de propósito general son usados normalmente por el conjunto de instrucciones para ejecutar operaciones aritméticas y lógicas. Estos registros pueden también ser direccionados como registros de 8 bits para operaciones por bytes.

Registros apuntadores e índice

Este conjunto de 4 registros, típicamente es usado para generar direcciones de memoria efectiva o la parte complementaria de la dirección física. Estos registros son direccionables únicamente como valores de 16 bits. También pueden ser usados por el conjunto de instrucciones para ejecutar operaciones lógicas y aritméticas en la generación de las direcciones efectivas de memoria.

Registros de segmento

Estos registros son usados para localizar segmentos de 64 K en el espacio de direcciones de 1 megabyte. El registro de segmento que es usado con una referencia específica de memoria se ilustra en la figura 2-8. Se debe notar que el hardware del microprocesador 8088 selecciona un registro por omisión. Sin embargo, en algunos casos es posible programar la sobrescritura del valor.

Registro apuntador de Instrucción

Este registro contiene las direcciones complementarías de la siguiente instrucción para el valor actual del código de segmento base.

Tipo de referencia de Memoria	Segmento base por omisión	Segmento base alternativo	Complemento
Búsqueda de instrucción	CS	Ninguno	IP
Operación con la pila	SS	Ninguno	SP
Variable (excepto siguiente)	DS	CS, ES, SS	Dirección efectiva
Cadena fuente	DS	CS, ES, SS	SI
Cadena destino	ES	Ninguno	DI
BP usado como registro base	SS	CS, DS, ES	Dirección efectiva

Figura 2-8 *Uso de los registros de segmento.*

Registro de banderas

Aunque este es un registro de 16 bits, solo 9 de sus bits son usados. Seis de ellos son bits de estado que reflejan el resultado de las operaciones aritméticas y lógicas. Los otros tres son bits de control. La figura 2-9 da un idea de los estados manejados. Para una descripción detallada puede consultarse el Manual de Referencia Técnica de IBM.

Conjunto de Instrucciones del 8088

El conjunto de instrucciones del microprocesador 8088 es muy extenso; incluye instrucciones para transferencia de datos, operaciones aritméticas y lógicas, manipulación de cadenas, control de transferencias y control del procesador.

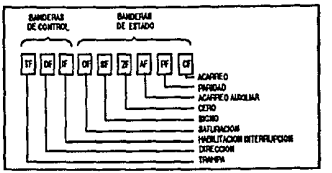


Figura 2-9 *Las banderas de 8088.*

2.5 Operaciones del bus del Sistema

La mayoría de las aplicaciones de interfases son conectadas a la PC en alguna de las ranuras de expansión. En el bus, los datos son transferidos durante lo que es llamado un ciclo del bus. En este apartado se describen los tipos de ciclos de bus posibles, y como son usados para transferir información entre la memoria, puertos de E/S y el microprocesador.

Hay dos clasificaciones generales para los ciclos de bus: manejados por el 8088 y manejados por DMA. Cuando el 8088 genera un ciclo de bus, maneja el bus del sistema con una dirección de memoria o un puerto de E/S, controla la dirección del flujo de datos y es también la fuente o el receptor de los datos. Cuando el 8088 maneja el bus, hay cinco diferentes tipos de ciclos de bus generados. El primer tipo es un ciclo de lectura de memoria, el segundo es un ciclo de escritura en memoria, el tercero es una lectura de puerto E/S, el cuarto es una escritura a un puerto E/S, el quinto es un ciclo de reconocimiento de interrupción. Este último no se cubre aquí, pues solo ocurre en el bus local del sistema.

La segunda clasificación general de los ciclos de bus son aquellos que son manejados por el controlador de DMA (el chip 8237-A). Durante las operaciones de acceso directo a memoria, el 8088 es aislado del bus del sistema y el controlador de DMA maneja los ciclos del bus. En este caso hay dos tipos de ciclos. El primero es un ciclo que lee desde una interfase y escribe los datos en una localidad de memoria que es especificada por la dirección del controlador de DMA; El segundo tipo es uno que lee datos desde una localidad de memoria especificada por el controlador de DMA y, entonces, los escribe en una interfase.

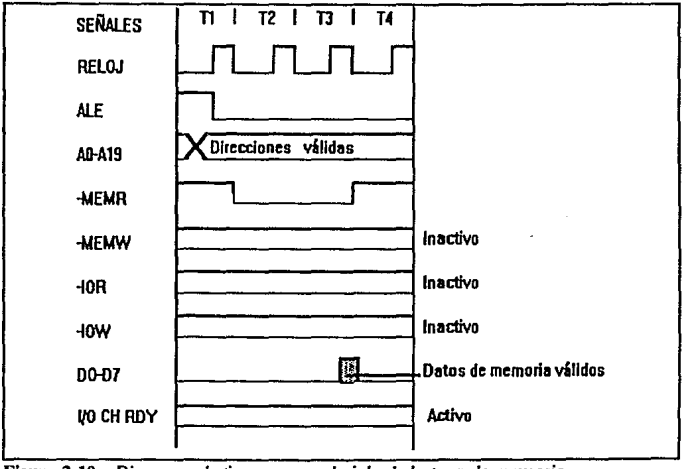


Figura 2-10 *Diagrama de tiempo para el ciclo de lectura de memoria.*

Ciclo de lectura de memoria

El ciclo de lectura de memoria es usado para buscar instrucciones y datos del sistema de memoria. Esta memoria puede estar en la tarjeta madre, en las ranuras de expansión, en ROM o RAM. Este ciclo es manejado desde el microprocesador 8088. Todos los ciclos de bus son hechos con un mínimo de 4 pulsos de reloj. Cada pulso tiene aproximadamente 210 nanosegundos de longitud. Por ello, la longitud mínima para un ciclo de lectura en memoria es de casi 840 nanosegundos. La longitud del ciclo de bus puede ser extendida por un dispositivos de memoria en las ranuras de expansión manejando la instrucción READY. El dispositivo de memoria debe manejar esta acción, de otra manera el ciclo del bus será solo de cuatro pulsos. Las señales activas del bus son manejadas por el 8088 y sus circuitos. La única excepción es el bus de datos, el cual será manejado con los datos provenientes de la localidad de la dirección de memoria. La figura 2-10 muestra las señales de tiempo básicas usadas en el ciclo para ejecutar la lectura.

El ciclo de lectura de memoria empieza durante el pulso de reloj T1 con la señal ALE siendo activa. El flanco de bajada de esta señal indica que el bus de direcciones contiene una dirección válida de memoria. Luego, la señal MEMR es activada aproximadamente al tiempo T2. Esto indica a los dispositivos conectados al bus que el ciclo es de lectura en memoria. También indica que si el dispositivo contiene memoria con una dirección que corresponde a una que se encuentra en el bus de direcciones, el dispositivo debe poner en el bus de datos su contenido. Todos los dispositivos de memoria deben decodificar la dirección presente en el bus, y por lo tanto, determinar si a ellos deben responder. El 8088 captura los datos provenientes del bus de datos al inicio del pulso de reloj T4. Poco después del inicio de T4, la señal MEMR se desactiva y el ciclo del bus finaliza al final del pulso de reloj T4.

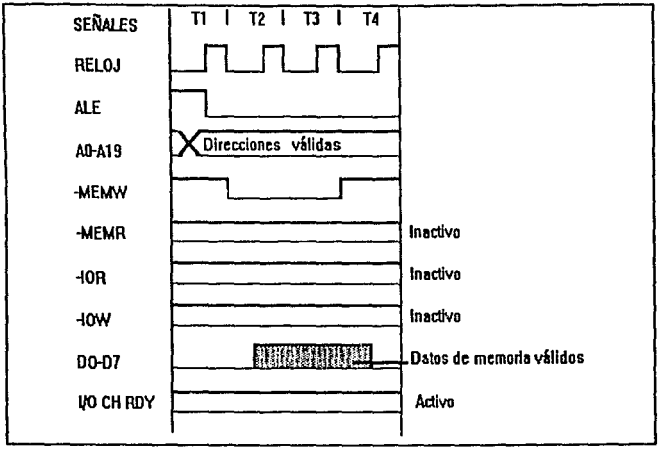


Figura 2-11 *Diagrama de tiempo para el ciclo de escritura en memoria.*

Ciclo de escritura en memoria

Este ciclo es usado siempre que una instrucción de un programa en el 8088 escribe datos en una localidad de memoria. Como en ciclo de lectura de memoria, el 8088 y sus circuitos manejan una dirección sobre el bus, indicando la dirección de la localidad de memoria que debe aceptar el dato proveniente del 8088. En adición, para manejar el bus de direcciones y las señales de control, el 8088 también pone en el bus de datos la información que va a ser escrita en la localidad de memoria seleccionada. La figura 2-11 muestra los tiempos básicos de este ciclo.

La señal ALE es activada durante el pulso T1, indicando que el bus de direcciones contiene una dirección de memoria válida. Luego, la señal MEMW se activa indicando que el ciclo es de escritura en memoria. La señal MEMW se activa aproximadamente en T2. Poco después, de que MEMW es activada, el 8088 introduce en el bus de datos un dato que es escrito en la localidad de memoria seleccionada. En T4 la señal MEMW es desactivada y el ciclo del bus se termina al final del pulso T4.

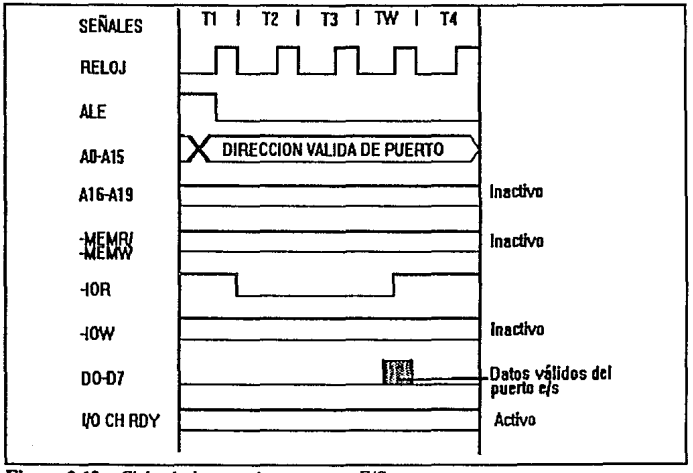


Figura 2-12 *Ciclo de lectura de un puerto E/S.*

Ciclo de lectura de un puerto E/S

Este ciclo es iniciado cada vez que la instrucción IN se ejecuta. Este ciclo es similar al de lectura en memoria. Su propósito es recolectar datos de alguna de las direcciones de puertos de E/S. En el diseño de la PC, este ciclo tiene siempre un mínimo de 5 pulsos de reloj. Un dispositivo en puerto específico de E/S puede extender la longitud del ciclo desactivando la señal READY. Durante un ciclo de lectura a un puerto E/S, el microprocesador maneja una dirección de puerto de 16 bits sobre el bus de direcciones. Debe notarse que durante este ciclo, los 4 bits de mayor orden (MSB: More Significance Bit) del bus de direcciones no son activados. La figura 2-12 muestra un diagrama de tiempo para este ciclo.

Durante el pulso de reloj T1, la señal ALE se activa, indicando que los bits 0 a 15 del bus de direcciones contienen una dirección válida de puerto E/S. Al pulso T2, la señal de control IOR se activa, indicando que el ciclo es de lectura en un puerto E/S, y que el puerto direccionado debe responder poniendo en el bus de datos sus contenidos. Al inicio del pulso T4, el procesador toma una muestra de los datos presentes en el bus y la señal IOR se desactiva. El ciclo de bus termina al final de T4. Debe notarse que un ciclo normal de lectura en un puerto E/S es de solo 4 pulsos de reloj, pero en el diseño de la PC, un pulso extra, llamado pulso TW, es insertado automáticamente en cada ciclo.

Ciclo de escritura en un puerto E/S

Este ciclo es iniciado cada vez que se ejecuta la instrucción OUT. Este ciclo escribe datos provenientes del 8088 a una dirección específica de puerto E/S. Normalmente, este ciclo es de 4 pulsos de reloj, pero el diseño de la PC inserta automáticamente un pulso TW extra. Este ciclo también puede ser expandido a un mayor número de pulsos de reloj, usando la señal READY. Otra vez, solamente los bits 0 a 15 del bus de direcciones son usados para direccionar un puerto en especial. La figura 2-13 muestra el diagrama de tiempo para este ciclo.

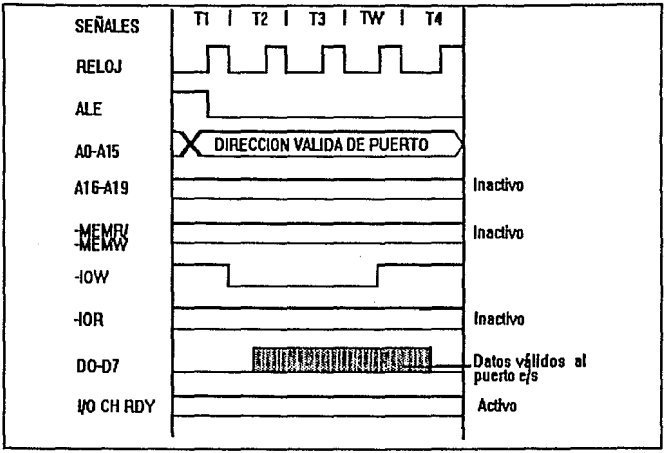


Figura 2-13 *Ciclo de escritura en un puerto E/S.*

Al igual que con los otros ciclos del bus, la señal ALE se activa durante el pulso T1 e indica que el bus de direcciones contiene una dirección válida de puerto E/S. Luego, la señal de control IOW se activa en el pulso T2, indicando que el ciclo es de escritura en un puerto E/S, y que el puerto seleccionado debe tomar los datos presentes en el bus de datos. Poco después del pulso T2, el procesador pone en el bus los datos para la dirección de puerto. Al inicio del pulso 4, la señal IOW se desactiva, El ciclo se termina al final del pulso T4.

Ciclo de DMA

Los ciclos de DMA son un poco más complicados, pues se ejecutan funciones de lectura y escritura en el mismo ciclo. Adicionalmente, el bus no es manejado por el microprocesador 8088, sino por el controlador de DMA y sus circuitos de soporte. Los ciclos no son iniciados ni por el 8088, ni por el controlador de DMA, sino que son iniciados por una solicitud proveniente de un adaptador de la interfase. Todo un conjunto de actividades del bus precede a un bus de DMA. Estas actividades no serán cubiertas, por razones de espacio. Un ciclo de DMA puede iniciarse mediante la activación de cualquiera de las tres señales DRQ1, DRQ2, ó DRQ3. La primera indicación de que una solicitud ha sido concedida y que un ciclo de DMA ha iniciado, es cuando se activa la señal AEN. Poco después de que la señal AEN se ha activado, una de las cuatro señales DACK0, DACK1, DACK2 o DACK3 se activa, indicando a la interfase cual pedido de DMA o canal esta siendo atendido en el actual ciclo de DMA.

Un ciclo normal de DMA toma únicamente cinco pulsos de reloj para ejecutarse. En el diseño de la PC se agrega un pulso. Cuando el controlador de DMA tiene el "control" del bus del sistema, sus señales de tiempo son generadas a partir de la misma señal de reloj que usa el microprocesador. Como el controlador DMA genera ligeramente diferentes temporizaciones, los pulsos de DMA son llamados pulsos "S". Los pulsos "S" indican el estado del controlador de DMA tal como los pulsos "T" indican el estado del microprocesador 8088. Cuando el controlador de DMA no se encuentra ejecutando un ciclo de DMA, se le llama estado "idle" (desocupado), ejecutando continuamente un estado de pulso "SI", mientras busca/espera pedidos de DMA. Cuando se detecta una solicitud, el controlador envía una señal al 8088 diciéndole que "suelte" el bus en un tiempo conveniente. El controlador también va al estado "S0", el cual continuamente busca una respuesta del 8088 que le informe que el bus se ha liberado para empezar un ciclo de DMA. Cuando el controlador recibe la señal HLDA proveniente del 8088, entra al estado S1, señalando el inicio del ciclo de DMA.

Ciclo de DMA para escritura en memoria

El propósito de este ciclo es tomar datos desde una interfase y escribirlos en la localidad de memoria especificada por el controlador de DMA. Después de que se inicia un ciclo de DMA, el controlador y sus circuitos de soporte ponen en el bus, la dirección de la localidad de memoria destino. Luego, la señal IOR se activa, indicando a la interfase que hizo la petición del ciclo DMA, poner sus datos en el bus. Luego, la señal MEMW se activa, indicando a la memoria que debe tomar los datos de la interfase y escribirlos en la memoria. Se debe notar que los datos provenientes de el adaptador de interfase no se retienen en ninguna parte. Es responsabilidad de la interfase mantener datos válidos hasta que la memoria pueda ejecutar la operación de escritura.

Ciclo de DMA para lectura de memoria

Este ciclo es usado para transferir datos provenientes del sistema de memoria a una interfase. Después que un ciclo de DMA, ha sido iniciado, el controlador de DMA y sus circuitos de soporte, ponen en el bus una dirección de memoria. Luego, el controlador activa la señal MEMR, indicando con ello a la memoria que debe poner en el bus de datos sus contenidos. Luego, el controlador activa la señal IOW. Esto indica a la interfase que debe tomar los datos de la memoria.

Ciclo de DMA para refrescar la memoria

Estos ciclos son usados por el sistema para refrescar la memoria dinámica del sistema. El hardware automáticamente genera una simulación de un ciclo de DMA para lectura de memoria cada 72 pulsos de reloj. El controlador de DMA esta programado para continuamente incrementar las direcciones de memoria en cada ciclo, para refrescar en su totalidad a la memoria dinámica. Como la lectura de datos no se usa en estos ciclos, el ciclo de DMA se acorta de tal manera que el ancho de banda del bus se reduce. Estos ciclos de DMA ocurren únicamente en el canal 0 de

DMA y son de 5 pulsos de longitud. Esto indica que el bus es utilizado un 7% del tiempo para la función de refresco.

2.6 Descripción de la señales disponibles en las ranuras de expansión.

El conjunto de señales del sistema de la PC esta disponible en 5 ranuras (slots) de expansión de 62 patillas cada una de ellas. Cada ranura posee señales idénticas en cada terminal con un nivel lógico TTL, a excepción de las líneas de potencia y tierra. Las señales disponibles se presentan ya demultiplexadas y con mayor energía que cuando son entregadas directamente por el microprocesador. Además, las ranuras contienen señales adicionales a las usadas por el microprocesador, a fin de soportar Acceso directo a memoria DMA, interrupciones, temporizaciones, control de entradas/salidas, lectura y escritura a memoria, generación de estados de espera, refresco de memoria y detección de error. A continuación se presenta una serie de descripciones funcionales detalladas de todas las 62 señales.

Definición de las señales

Osc (Oscillador: Oscilador)

Esta señal de salida tiene una frecuencia de 14.31818 MHz, y un período de aproximadamente 70 ns. Tiene un ciclo de trabajo de 50 % y es la señal de mayor frecuencia en el bus. Todas las demás señales son generadas a partir de ésta señal; su valor puede ser ajustado mediante un capacitor variable (trimmer) que esta localizado en la tarjeta base del procesador. Esta señal puede ser dividida por cuatro para dar la frecuencia de 3.58 MHz requerida por el monitor de color de T. V.

Clk (Clock: Pulsos de Reloj)

Esta señal de salida es derivada por la división por tres de la señal OSC previamente descrita, lo cual da una frecuencia de 4.77 MHz. Esta señal no es simétrica, pues tiene un tiempo alto de 70 ns y un tiempo bajo de 140 ns. Esta señal está bien sincronizada con respecto al control de lectura y escritura en la memoria, y puede ser usada para generar estados de espera. Un ciclo típico es de cuatro períodos de reloj o aproximadamente 840 ns.

Reset Drv (Reset Driver: Manejador de inicialización)

Esta señal de salida es mantenida en nivel activo-alto durante la secuencia de encendido del sistema y permanece activa hasta que todos los niveles han alcanzado sus rangos adecuados de operación, y entonces, se inactiva. En adición, si cualquier nivel de energía cae por debajo de su rango especificado de operación, ésta línea es llevada a su nivel activo-alto. Esta señal es generalmente usada para proveer una inicialización o limpia de los dispositivos conectados y llevarlos a un estado conocido antes de la operación del sistema. Esta señal es puesta activa o inactiva en el flanco de bajada de la señal OSC.

A0 a A19 (Address 0 a 19: Líneas de dirección 0 a 19)

Las líneas de dirección A0 a A19 son señales de salida únicamente y son usadas para direccionar localidades de memoria o puertos de entrada/salida E/S. A0 es el bit menos significativo y A19 es el bit más significativo. Estas 20 líneas de dirección son manejadas por el microprocesador durante los ciclos de lectura y escritura hacia memoria o puertos E/S. Sin embargo durante los ciclos de Acceso Directo a Memoria (DMA), son manejadas por el dispositivo 8237-5, el cual es un controlador de DMA. Con 20 líneas de dirección es posible direccionar un Megabyte

de localidades de memoria, pero no todo este espacio de direcciones esta ocupado por el sistema. La memoria básica, 0 a 64 Kbytes, reside en la tarjeta del procesador y obviamente este espacio de direcciones no debe ser direccionado en ningún diseño propio. Similarmente, la tarjeta del procesador ocupa un espacio de direcciones para 48 Kbytes de memoria ROM, el cual reside en la parte más alta del espacio de direcciones; otra vez, éste no debe ser utilizado en alguna aplicación.

El microprocesador a través del uso de las instrucciones IN y OUT, puede direccionar hasta 64 Kbytes de direcciones únicas de puertos de E/S. Estas direcciones de puertos son trasladadas por medio de las líneas de dirección A0 a A15; las líneas A16 a A19 no son usadas y son mantenidas inactivas durante los ciclos E/S de puertos. Sin embargo en la PC, solo las líneas de dirección A0 a A9 son usadas para direccionar puertos, por lo cual solamente las direcciones de puertos E/S en el rango 0200 h a 03FF h son válidas. Existen algunas excepciones a esta regla.

D0 a D7 (Data 0 a 7: Líneas de datos 0 a 7)

Estas 8 líneas de datos son bidireccionales y son usadas para transmitir datos entre el microprocesador, memoria, y puertos E/S. D0 es el bit menos significativo y D7 es el bit más significativo. Durante los ciclos de escritura, los datos son presentados en el bus para escribirlos en memoria o en los puertos E/S. Los datos son válidos justo antes del flanco de subida de las señales de control IOW o MEMW. Los flancos de subida de estas señales son, por lo común, usados para regular el flujo de datos entre el bus y la memoria o los registros de los puertos E/S. Durante los ciclos de lectura, la localidad de memoria o el registro de puerto direccionada, debe poner sus datos en el bus de datos antes del flanco de subida de las señales de control IOR o MEMR. Durante los ciclos de acceso directo a memoria, el bus de datos es usado para transferir datos directamente de un puerto de E/S y memoria sin la intervención del microprocesador; durante estos ciclos, el procesador es desconectado del bus y el dispositivo DMA 8237-5 controla la transferencia del bus.

ALE (Address Latch Enable: Habilitación de la dirección)

Esta señal de salida es manejada por el controlador de bus, el dispositivo 8288. Es usada para indicar que la dirección presente en el bus es válida para empezar un ciclo de bus. Esta señal cambia a nivel activo-alto justo antes de que el bus de direcciones sea válido y cae a nivel bajo justo después de que el bus de direcciones es válido. La señal es usada para retener la información de la dirección proveniente del bus local de datos/direcciones del procesador 8088. Debe señalarse que el bus del sistema de datos no contiene información de dirección y, por ello, ALE no puede ser usada para demultiplexar direcciones del bus de datos. La señal ALE es un buen punto de sincronía cuando se realizan ciclos de bus, pues empieza al principio de cada ciclo de bus. ALE no es activa durante los ciclos de acceso directo a memoria.

I/O CH CK (I/O Channel Check: Revisión del canal de Entrada / Salida)

Esta es una señal de bajo nivel usada para reportar condiciones de error en las tarjetas conectada en el bus. Cuando esta señal es puesta a nivel bajo, genera una interrupción no enmascarable NMI para el microprocesador.

I/O CH RDY (I/O Channel Ready: Canal de entrada / salida preparado)

Esta señal de entrada es usada para extender la longitud de ciclos de un bus; de esta forma, si la memoria o los puertos de E/S no son lo suficientemente rápidos como para responder a un ciclo de bus normal de 4 pulsos de reloj (840 ns), pueden trabajar mediante el adecuado uso de esta señal. Si una memoria o un puerto E/S quiere extender el ciclo de bus, entonces forzará a la línea I/O CH RDY a un nivel bajo cuando decodifica su dirección y recibe un comando MEMR, MEMW, IOR o IOW. Esta señal debe ser controlada cuidadosamente para que únicamente los estados de espera necesarios sean sumados al ciclo de bus. Manteniendo esta línea

inactiva se puede extender el ciclo de bus mediante la adición de estados de espera en incrementos de 210 ns, hasta un máximo de 10 pulsos de reloj, o 2100 ns. Esta señal debe ser manejada con una salida de colector abierto, pues está cableada con otras señales I/O CH RDY, de otras tarjetas de interfase conectadas al bus.

IRQ2 a IRQ7 (Interrupt Request 2 - 7: Solicitud de Interrupción 2 a 7)

Estas 6 señales de entrada son usadas para generar peticiones de interrupción al microprocesador desde el bus del sistema. Estas señales van directamente al control de interrupciones 8259A de la tarjeta madre. Los programas del BIOS en ROM inicializan al controlador para que la IRQ2 sea la señal de mayor prioridad e IRQ7 la de menor prioridad. Si el nivel no está enmascarado, un flanco de subida genera una solicitud de interrupción al microprocesador 8088. Una vez que se genera ésta señal, permanece activa hasta que el procesador envía una señal INTA (reconocimiento de interrupción). La señal INTA no está presente en el ducto del sistema, por ello la solicitud es usualmente limpiada con un bit en un registro de puerto E/S, usando el comando OUT enviado en la rutina de servicio a interrupciones. Si la solicitud de interrupción no se mantiene activa hasta que la señal INTA llegue, entonces se genera una interrupción de nivel 7, sin importar el nivel de prioridad original. Debe notarse que este comportamiento puede cambiarse reprogramando el parámetro de inicialización en el controlador 8259A.

IOR (I/O Read: Lectura en un puerto de E/S)

Esta señal de salida es manejada por el controlador de bus, el chip 8288. Es usada para indicar a los puertos E/S que el ciclo de bus presente es un ciclo de lectura a un puerto E/S y que la dirección en el bus de direcciones es una dirección de puerto. La dirección de puerto E/S debe responder colocando sus datos en el bus de datos del sistema. Esta señal es activo-bajo, así que el puerto E/S debe poner sus

datos en el bus aproximadamente 30 ns antes del flanco de subida de IOR, para asegurar que el procesador tome datos válidos. Cuando ocurre un ciclo de acceso directo, la señal IOR es manejada por el controlador de acceso directo a memoria (DMA), el dispositivo 8237-5. En este caso, la dirección del canal no contiene una dirección de puerto E/S, sino que contiene la dirección de la memoria donde serán escritos los datos leídos del puerto . El puerto de E/S no es seleccionado por una dirección, más bien por la señal activa DACK proveniente del controlador DMA 8237-5.

IOW (I/O Write: Escritura en un puerto de entrada /salida)

Esta señal es una salida de nivel activo alto, la cual es manejada por el controlador de bus 8788 durante los ciclos de bus, e indica al bus de direcciones que contiene una dirección de puerto E/S y que el bus de datos contiene un dato a ser escrito en el puerto E/S dado. Cuando la señal va a su estado activo-alto, el bus de datos puede no ser válido, por ello el puerto de datos debe ser sincronizado usando el flanco de subida de esta señal. Cuando ocurre un ciclo de DMA, esta señal es manejada por el controlador 8237-5. La señal IOW se usa entonces para escribir datos de la memoria, que se encuentran ahora en el canal de datos, en el puerto E/S seleccionado por el DACK. Nuevamente, los datos pueden no ser válidos en el borde de subida de ésta señal y deben ser sincronizados en el puerto usando el flanco de bajada de esta señal.

MEMW (Memory Write: Escritura en Memoria)

Esta es una señal de salida activo-bajo usada para escribir datos provenientes del bus del sistema hacia memoria. Esta señal es manejada por el controlador de bus e indica que el bus de direcciones contiene una dirección de una localidad de memoria en la cual la información sobre el bus de datos debe ser escrita. Los datos

pueden ser inválidos en el bus de datos cuando MEMW va a activo-bajo, pero son válidos justo antes de el flanco de subida de esta señal. Durante los ciclos de DMA esta señal es manejada por el controlador 8237 y es usada para escribir datos de un puerto E/S hacia memoria. Debe tenerse particular cuidado cuando esta señal es usada para un ciclo de escritura en memoria dinámica, ya que los datos escritos pueden ser inválidos cuando esta señal va a su nivel activo.

Esto puede requerir que la memoria RAM dinámica, DRAM, tenga que ser diseñada para usar la escritura retardada.

MEMR (Memory Read: Lectura en Memoria)

Esta señal es una salida de bajo nivel usada para solicitar lectura de datos en memoria. Esta señal es manejada por el controlador de bus 8288.

Indica que el bus de direcciones contiene una dirección válida de memoria y que la localidad especificada de memoria debe manejar el bus de datos con su lectura de datos. Como con la señal IOR, la memoria debe manejar datos válidos, en el bus de datos, aproximadamente 30 ns justo antes del flanco de subida de la señal MEMR para asegurar que el procesador recibe datos válidos. Durante los ciclos DMA, esta señal es manejada por el controlador 8235-A e indica que la dirección de la localidad de memoria debe responder manejando el bus de datos con su lectura de dato, para que pueda ser escrito dentro del puerto DACK seleccionado.

DRQ1 a DRQ3 (Direct memory access request 1 - 3: Solicitud de acceso directo a memoria 1 a 3)

Estas líneas de entrada son de nivel activo-alto y son usadas en interfases para solicitar ciclos DMA. Si un dispositivo o interfase lógico quiere transferir datos entre él y memoria sin la intervención del CPU, la petición es inicializada llevando a

un nivel alto a la línea DRQ. DRQ1 es la de mayor prioridad y DRQ3 es la de menor prioridad. En realidad, DRQ0 tiene la mayor de las prioridades, sin embargo no se encuentra disponible en el bus del sistema, pues es usada por el procesador para refrescar el sistema de memoria dinámica. Dependiendo del modo programado en el controlador DMA, debe tenerse cuidado en el control de estas líneas, pues si DRQ es mantenido alto por demasiado tiempo, más de un ciclo puede ser tomado. Típicamente, la señal DRQ es reestablecida por su correspondiente señal DACK.

DACK0 a DACK 3 (Direct memory access acknowledge 0-3: Aceptación de acceso directo a memoria)

Estas 4 señales de salida son de nivel activo bajo emitidas por el controlador DMA 8237 para indicar que la señal DRQ correspondiente ha sido aceptada y que el controlador DMA tomará el bus y procederá con la solicitud del ciclo DMA. No hay correspondencia con DRQ0 en el bus del sistema, por ello DACK0 es enviada solo para indicar que el ciclo presente DMA es una simulación de ciclo de lectura la cual puede se usada para refrescar sistemas de memoria dinámica. Durante los ciclos de lectura de memoria el bus de direcciones contiene incrementos de refresco válidos. Los ciclos de refresco DACK0 ocurren cada 72 pulsos de reloj o 15.12 ìs.

AEN (Address Enable: Habilitación de una dirección)

Esta señal es una salida activo-alta emitida por el controlador lógico DMA para indicar que un ciclo de bus DMA está en progreso. Dentro de la tarjeta del microprocesador, ésta señal es usada para deshabilitar los buses de direcciones, datos y control del microprocesador y para habilitar el bus de direcciones y controles del controlador DMA.

TC (Terminal Count: Conteo Terminado)

Esta señal es una salida de nivel activo-alto emitida por el controlador DMA 8237 para indicar que uno de los canales de DMA ha alcanzado su número preprogramado de ciclos de transferencia. Esta señal es típicamente usada para terminar un bloque de transferencia de datos DMA.

+5 y -5 Volts dc.

La energía de 5 volts DC se encuentra disponible en 2 patillas de cada una de las ranuras del sistema, y tiene una regulación de ±5% (4.75 a 5.25 volts).

La energía de -5 V se halla disponible con una regulación en ± 10 % (-4.75 a -5.5 volts).

+12 y -12 Volts dc.

El nivel de +12volts está disponible en una patilla y se encuentra regulada a ±5% (11.4 a 12.6 V DC)

El nivel de -12 volts está disponible en una patilla y se encuentra regulada a ±10% (-10.8 a -13.2 V DC)

GND

El sistema de tierra DC esta disponible en 3 patillas.

La figura 2-14 muestra un diagrama de la configuración existente en una ranura de expansión, con sus respectivas señales.

SEÑAL	PATILLAS DE LA RANURA		SEÑAL
GND	B1	A1	-I/O CH CK
RESET DRV	B2	A2	D7
+15 VDC	B3	A3	D6
IRQ2	B4	A4	D5
-5VDC	B5	A5	D4
DRQ2	B6	A6	D3
-12VDC	B7	A7	D2
NO USADO	B8	A8	D1
+12VDC	B9	A9	D0
GND	B10	A10	I/O CH RDY
MEMW	B11	A11	AEN
MEMR	B12	A12	A19
IOW	B13	A13	A18
IOR	B14	A14	A17
-DACK3	B15	A15	A16
DRQ3	B15	A16	A15
-DACK1	B17	A17	A14
DRQ1	B18	A18	A13
-DACK0	B19	A19	A12
CLK	B20	A20	A11
IRQ7	B21	A21	A10
IRQ6	B22	A22	A9
IRQ5	B23	A23	A8
IRQ4	B24	A24	A7
IRQ3	B25	A25	A6
-DACK2	B26	A26	A5
T/C	B27	A27	A4
ALE	B28	A28	A3
+5VDC	B29	A29	A2
OSC	B30	A30	A1
GND	B31	A31	A0

Figura 2-14 *Definición de las señales que existen en los pines de las ranuras de expansión.*

2.7 Sistema de Interrupciones

Las interrupciones pueden ser muy útiles y son frecuentemente usadas cuando se interconecta un diseño al sistema del microprocesador. Su principal ventaja es la habilidad para conseguir la atención del microprocesador por servicio, sin que requiera que el procesador continuamente le pregunte lo que desea. Esto deja al procesador en libertad de hacer otras cosas hasta que se soliciten sus servicios.

Un buen ejemplo del uso de las interrupciones es el servicio del teclado de la Computadora Personal. Cada pulsación del teclado genera al menos una interrupción al 8088. Como el microprocesador esta ejecutando un programa, no puede estar en un ciclo esperando a que el usuario oprima una tecla. Si éste fuera el caso, el programa nunca se ejecutaría pues toda la atención del procesador estaría dedicada a obtener la siguiente pulsación. Una solución simple sería, detener el programa periódicamente y ver en la interfase del teclado si se ha pulsado una tecla. Sin embargo, el programa de aplicación debe saber cuando y con que frecuencia hacerlo. Por otro lado, mucho de su tiempo de procesamiento sería usado en un búsqueda improductiva en la interfase del teclado.

La función de interrupción resuelve este problema deteniendo automáticamente el programa en algún punto conveniente. por ejemplo, la terminación de la siguiente instrucción, e indica al procesador que el programa necesita servicio cada vez que se oprime una tecla. Después de que se completa una tarea de interrupción, el hardware automáticamente regresa el control al programa que se estaba ejecutando.

La función de interrupción se usa frecuentemente en aplicaciones de interfase donde los programas requieren sincronización con eventos externos, o donde errores o condiciones de estado pueden requerir la atención del procesador o del programa.

Como muchas de las condiciones de interfase pueden requerir servicio, y posiblemente todas al mismo tiempo, las funciones de interrupción de la computadora proveen nueve niveles de interrupción. Estos nueve niveles tienen diferentes prioridades, de tal manera que cuando múltiples peticiones son activadas al mismo tiempo, se atienden en un orden secuencial preestablecido.

Sistema de Interrupciones en la PC

El microprocesador 8088 tiene 2 tipos de interrupción disponibles: una interrupción enmascarable y la otra no enmascarable (NMI). Como la NMI no es enmascarable internamente por el procesador 8088, se utiliza lógica en la tarjeta madre para enmascarar y desenmascarar la NMI. En la interrupción enmascarable el diseño de la PC conecta un controlador de interrupciones que expande a 8 los niveles de interrupción, es por ello que se tienen 9 niveles en total. Sin embargo no todos se encuentran disponibles para aplicaciones de interfase. La figura 2-15 muestra un resumen de los niveles de interrupción y su uso.

Secuencia de eventos en una interrupción

Cuando ocurre una interrupción en el sistema, toma lugar una secuencia de eventos que dirige la solicitud de interrupción hacia el programa apropiado necesario para satisfacer tal requerimiento. A continuación se presenta un breve resumen de la secuencia de eventos que ocurren cuando se activa una interrupción, incluyendo la secuencia de inicialización (esta se ejecuta con la carga del sistema operativo).

1. Inicialización del estado del apuntador de la pila[5] y restauración del área para guardar registros en el sistema de RAM.

[5] Pila o stack es un área reservada para operaciones internas. Las pilas funcionan siguiendo el fundamento de último en llegar, primero en salir.

Nivel Interrupción		Uso
Mayor nivel	NMI	Verificación de paridad de RAM, de canal de E/S y procesador numérico.
	IRQ 0	Salida del temporizador 8253-5, canal 0
	IRQ 1	Búsqueda de interrupción del teclado.
Disponible en el bus del sistema	IRQ 2	No usada
	IRQ 3	No usada
	IRQ 4	Puerto serial RS232
	IRQ 5	No usada
	IRQ 6	Estado del disco flexible
	IRQ 7	Puerto paralelo (no usado en BIOS)

Figura 2-15 *Resumen de los niveles de interrupción.*

2. Inicialización de las direcciones bajas de memoria RAM del 8088, con las direcciones de las rutinas para servicio el interrupciones. Los primero 1024 bytes del espacio de direcciones del 8088 son reservados para 256 apuntadores de 4 bytes, lo que significa que se puede dar servicio hasta de 256 interrupciones. Los cuatro bytes consisten de un valor para el apuntador de instrucción (IP) y otro para un código de segmento (CS). El BIOS solo hace esto para las interrupciones que usa.
3. Inicialización del chip controlador de interrupciones, el chip 8259.

   A. Inicialización de los comandos ICW 1 a 4 (Initialization Command Word: Palabra para inicialización).

   B. Inicialización de la OCW (palabras para control de operación). Esto significa desenmascarar los niveles de interrupción.

4. Desenmascarar el sistema de interrupciones en el 8088. (establecer el bit de estado en el registro de banderas)
5. La interfase genera una solicitud de interrupción.
6. El controlador de interrupciones 8259A toma la solicitud.
7. El controlador de interrupciones envía la solicitud al microprocesador 8088.
8. El 8088 responde con un pulso en la terminal INTA, con lo cual da la prioridad a la solicitud y pone las solicitudes en el retenedor (latch) de servicio del controlador de interrupciones.
9. El 8088 envía un segundo pulso por INTA, el cual es respondido por el controlador 8259A con el valor del nivel del apuntador. Este valor es usado para obtener la dirección, en el espacio de direcciones bajas del 8088, de la rutina de servicio de interrupción.
10. Luego, el 8088 desenmascara las interrupciones, guarda sus banderas en la pila, almacena la dirección de la siguiente instrucción (los valores IP y CS) en la pila, busca los nuevos valores de IP y CS usando el valor del apuntador, y salta a los nuevos valores de dirección IP y CS.
11. La rutina de servicio a la interrupción se introduce en la pila que usará y que podría destruir.
12. Para permitir adicionales niveles de interrupción, la rutina de interrupción emite una instrucción de fin de interrupción (EOI: end of interrupt) al controlador 8259A.
13. La rutina de servicio a la interrupción se ejecuta.
14. Cuando se completa la rutina de servicio, el registro y estado son reestablecidos extrayendolos (POP) de la pila.
15. Por último, la rutina de servicio executa una instrucción de retorno de interrupción, con lo cual desenmascara las interrupciones, toma los valores anteriores de CS e IP de la pila, y brinca a esos valores y continua la ejecución del programa donde fue interrumpido.

2.8 Sistema de Acceso Directo a Memoria (DMA)

En muchas aplicaciones de interfase, llega a ser necesario aceptar o transmitir datos desde o hacia la interfase, a velocidades mayores de las posibles con un ciclo simple programado que emplee las instrucciones IN y OUT. Un buen ejemplo de este problema es la transmisión y recepción de datos desde las unidades de disco de la PC. La velocidad con que fluyen los datos desde o hacia el adaptador de disco es lo suficientemente alta como para que el 8088 se encontrara en problemas para atenderlo y además dar servicio a otros dispositivos, tales como el teclado. Para resolver estas interfases de alta velocidad de transferencia, se provee la función especial llamada Acceso Directo a Memoria (DMA). La función DMA permite a un adaptador o interfase leer o escribir datos desde o hacia la memoria sin usar el microprocesador. Esta función la suministra el dispositivo controlador Intel 8237-5.

Concepto básico de DMA

Durante la ejecución normal de un programa, el 8088 maneja el bus del sistema, direccionando la memoria y manejando la información. Cuando una interfase desea transferir datos usando la habilidad de DMA, envía una señal de pedido al controlador de DMA. El controlador entonces otorga una prioridad a la petición y envía una petición de alto al 8088. Al final del actual ciclo del bus, el 8088 se desconecta del bus del sistema y envía una señal de reconocimiento, indicando al controlador de DMA que el bus se encuentra libre. Entonces, el controlador de DMA se conecta al bus del sistema y maneja al bus de control y de direcciones, ejecutando un ciclo de transferencia de datos entre la interfase que lo pidió y la memoria. La interfase (o adaptador) es notificada de esta acción por el controlador de DMA quien le envía una señal de reconocimiento. Se puede pensar en el controlador como si fuera un componente adicional, el cual, cuando es requerido toma el control del bus del sistema y dirige la transferencia de datos entre la memoria y el adaptador. Debe

notarse que durante las operaciones de DMA, el controlador de DMA no maneja el bus; sólo se pueden transmitir datos directamente entre la memoria y el adaptador o interfase.

Uso de DMA en la Computadora Personal

El chip controlador de DMA, 8237-5 tiene cuatro canales de DMA. Dos de ellos son usados en el diseño de la PC. El canal 0 es usado por la función de refresco para la memoria dinámica. El canal 2 es usado para la transferencia de datos entre el adaptador de discos y la memoria. Los canales 1 y 3 no son usados. Los canales 1, 2 y 3 se encuentran disponibles en el bus del sistema para ser usados en interfases instaladas en las ranuras de expansión. El BIOS de la PC inicializa el controlador de DMA para que el canal 0 tenga la mayor prioridad y el canal 3, la menor. La siguiente tabla resume las prioridades y uso de los cuatro canales de DMA de la PC.

Mayor prioridad	Canal 0	Soporta refresco de memoria
..	Canal 1	No usada
..	Canal 2	Soporta al adaptador de disco
Menor prioridad	Canal 3	No usada

Tabla 1

Operación de DMA

El siguiente procedimiento es una descripción paso a paso de las acciones tomadas por la PC durante un ciclo operacional de DMA.

1. Antes de una operación de DMA pueda ocurrir, ese debe inicializar al controlador 8237-5, para ejecutar un apropiado tipo de ciclo.

   - Seleccionar la función de memoria "lectura a" o "escritura de".
   - Tipo de transferencia: en bloque o un solo byte.
   - Cantidad de bytes a ser transferidos.
   - Dirección de memoria para inicio de la transferencia.
   - Habilitación de la señal del canal de pedido.

   Esta inicialización es realizada dando instrucciones de control al dispositivo 8237-5, usando las instrucciones para puertos de entrada/salida del microprocesador 8088.

2. El adaptador o interfase envía una señal DRQ al controlador 8237-5, indicándole que se requiere una transferencia de datos en un canal específico. Hay tres señales DRQ en el bus del sistema; uno para cada uno de los canales 1,2 y 3.

3. El controlador 8237-5 identifica la prioridad de la petición y envía una señal HRQ a la circuitería de generación de estado de espera del microprocesador.

4. Los circuitos de estado de espera rastrean las líneas de estado del microprocesador y buscan por un estado pasivo del procesador (Bus inactivo o un ciclo que vaya a terminar).

5. Cuando se detecta un estado pasivo, la lógica de control envía una señal de "no preparado" al procesador, causando que el procesador entre en un estado de espera. También se envía una señal HOLDA al controlador 8237-5, indicando que al siguiente reloj, el bus estará libre y un ciclo DMA tendrá lugar. Las señales también son enviadas al bus de direcciones, control y a los buffers de datos, los cuales remueven al microprocesador 8088 del bus del sistema. Debe notarse que el 8088 aun procede con el ciclo de bus hasta el pulso de reloj T3 y, entonces suspende el ciclo.

6. El controlador 8237-5 detectara la señal HOLDA y envía una señal DACK a la interfase que hizo la petición. Esta señal actúa como un selector de chip para el adaptador, habilitándolo en el bus del sistema.

7. El controlador 8237-5 ahora maneja un dirección sobre el bus del sistema, apuntando a la localidad de memoria donde la transferencia de datos tomará lugar. El controlador 8237-5 también tomará el control de las líneas de control del bus (MEMR, MEMW, IOR e IOW) y ejecutará las operaciones de lectura y escritura sobre el bus.

8. La interfase o adaptador, después de recibir la señal DACK, pasa la señal DRQ al controlador. El controlador, después de completar el ciclo, deja pasar la señal HRQ a los circuitos de lógica de control de estado de espera. Luego, estos circuitos pasan la señal HOLDA al controlador, indicando que el microprocesador nuevamente entrará al bus. Finalmente, los circuitos de estado de espera dejan pasar la señal "No preparado" para que el procesador rehabilite sus buffers sobre el bus.

   El ciclo de bus que fue suspendido al pulso de reloj T3 es reiniciado y el 8088 continua sus operaciones normales. Se debe notar que

cuando el ciclo de bus es reiniciado, se insertan dos pulsos de reloj extras para dar al ciclo de bus suficiente tiempo de acceso.

Las operaciones anteriores son hechas en cada ciclo de DMA. Esta operación esta hecha de esta manera para que el microprocesador pueda operar en modo máximo y soportar el procesador auxiliar 8087.

2.9 Sistema de Temporizadores y Contadores

Una facilidad que frecuentemente se requiere en el diseño de una interfase es una función de conteo y temporizado. Para soportar estas funciones, la PC tiene tres contadores temporizadores independientes en un sólo chip, el Intel 8253-5, implantado dentro de la tarjeta madre. En general, estos canales de conteo y temporizado se usan para soportar las funciones básicas de E/S de la PC que no se encuentran disponibles para uso general en un diseño de interfase.

Cada canal de contador temporizador tiene una entrada de señal de reloj y una entrada de señal de compuerta, donde la compuerta controla la entrada de reloj al contador temporizador. Cada contador tiene una señal de salida cuya función se fija por programación del modo de operación del canal. En el diseño de la PC, las entradas de reloj es el mismo en todos los canales, una señal cuadrada de 1.19318 MHz. Cada canal contador temporizador tiene una longitud de 16 bits.

Uso de los canales temporizadores y contadores

Canal 0

Este canal es utilizado como un temporizador de uso general en el sistema. Su señal de compuerta se mantiene alta todo el tiempo o "encendida" y la señal de reloj de entrada es una onda cuadrada de 1.19318 MHz. La salida de el temporizador en este canal se liga al nivel de interrupción 0, el nivel más alto de interrupción enmascarable. Este canal es configurado por el PC BIOS para generar una petición de interrupción de nivel 0, cada 54.936 milisegundos. Estos pedidos de interrupción son contados por una rutina del BIOS, que genera un reloj de tiempo real que puede ser leído o escrito. Esta rutina también usa el conteo de interrupciones para generar el retardo del apagado del motor después de una búsqueda en disco. En cada pedido de interrupción, la rutina actualiza el reloj de tiempo real, verifica si el motor de la unidad de disco flexible requiere ser apagado, e intenta invocar una rutina definida por el usuario. Esta última función puede ser de interés para alguna aplicación de interfase. Cada 54.936 milisegundos la rutina del BIOS enviará una interrupción por software al nivel de interrupción 1C hexadecimal. Esta tabla de vectores de interrupción se compone por un código de segmento y un complemento (offset) que simplemente regresa el control al BIOS. Un programa de aplicación puede ir y alterar el valor del vector de interrupciones y dirigir la interrupción periódica a una rutina del usuario. La dirección de esta rutina es entonces insertada en la tabla de vector de interrupciones para la interrupción 1C.

Canal 1

Este canal temporizador contador es usado de manera dedicada para soportar la función de refresco del sistema. El reloj de entrada esta ligado a la señal de onda cuadrada de 1.19318 MHz y la compuerta está fija a un nivel alto. La salida del canal es usada para generar una petición de ciclo para acceso directo a memoria (DMA)

en el canal 0 del DMA. Este canal de DMA es usado para refrescar el sistema de memoria dinámica, simulando ciclos de lectura de memoria cada 72 ciclos del reloj del procesador (210 nanosegundos), o sea, cada 15.12 microsegundos.

Canal 2

Este canal tiene un propósito dual en la PC. Primero, es usado para sacar datos seriales escritos a un puerto de cassettes de audio del sistema. Segundo, es usado para controlar la bocina de audio.

2.10 Direccionamiento y uso de puertos de la PC

La mayoría de los dispositivos de soporte y adaptadores de E/S [Input/Output: Entrada/Salida: E/S] en la PC son controlados y sensados usando los puertos digitales de entrada y salida. Estos puertos son direccionados usando el espacio de direcciones de E/S del microprocesador 8088. Los datos pueden ser enviados a esos puertos usando la instrucción OUT en la familia de microprocesadores 8088/8086. Por otro lado, los datos pueden ser sensados o leídos de esos puertos usando la instrucción IN. La arquitectura del microprocesador 8088 soporta un espacio de 65,536 direcciones únicas de puertos. El diseño de la PC no usa en su totalidad éste espacio de direcciones. Solamente los 10 bits menos significativos del campo de direcciones son usados en la PC. Esto significa que exclusivamente los bits 0 a 9 son usados para decodificar dispositivos o direcciones de puertos.

Es importante notar que las instrucciones IN y OUT en la familia de procesadores 8088 pueden aun así ser usadas para especificar direcciones de puertos con los bits de alto orden, pero los dispositivos actualmente diseñados tomarán en

cuenta sólo a sus líneas decodificadas y responderán, por tanto, a los bits 0 a 9 únicamente, sin ser afectados.

El bit 9 de el campo de direcciones de los puertos E/S tiene un significado especial en el diseño de dispositivos periféricos. Cuando este bit es inactivo, los datos no pueden ser recibidos en el bus de las ranuras de expansión. Cuando este bit es activo, se habilitan los datos provenientes de las ranuras de expansión. Esto quiere decir que para puertos de entrada, los 1024 direcciones de puertos soportadas en la PC, son igualmente divididos en 512 direcciones de puertos posibles en la tarjeta del sistema y 512 direcciones de puertos que pueden existir en el bus de tarjetas de expansión.

Cabe destacar que esta restricción no es aplicable a los puertos de salida; pues cualquiera de los 1024 direcciones de puertos pueden ser usadas como direcciones de puertos de salida en las tarjetas localizadas en las ranuras. Sin embargo, las direcciones de los puertos de salida que son usadas en tablero del sistema no deben ser duplicados en las tarjetas localizadas en los ranuras de expansión.

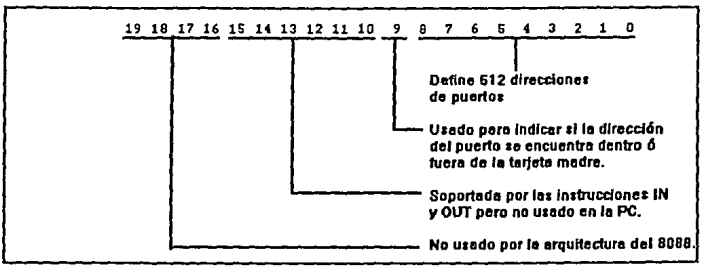


Figura 2-16 *Direccionamiento de puertos de Entrada/Salida del Microprocesador 8088.*

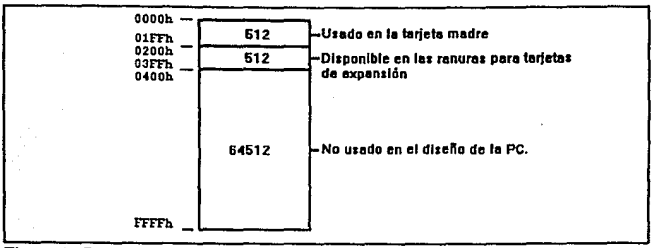


Figura 2-17 *Uso del espacio de direcciones para puertos de E/S.*

La figura 2-16 ilustra el uso de el campo de direcciones de E/S. La figura 2-17 ilustra el espacio de direcciones permitidos en el diseño de la PC.

Mapa de direcciones para puertos de E/S

El mapa de direcciones de puertos E/S puede ser dividido en dos partes. La primera parte es el espacio de direcciones 0000 a 01FF, la cual reside en el sistema de tarjeta madre.

Este rango de puertos son usados para direccionar dispositivos de soporte integrados en la tarjeta del procesador 8088. La figura 2-18 define las funciones de las direcciones de los puertos del sistema interno de la PC.

Debemos notar que el rango de direcciones 00C0h a 01FFh no son usadas ni como entradas, ni como salidas. Como previamente se enunció, estas direcciones no pueden ser usadas como puertos de entrada pero es posible decodificar esas direcciones como puertos de salida para usarlos en el diseño de la interfase.

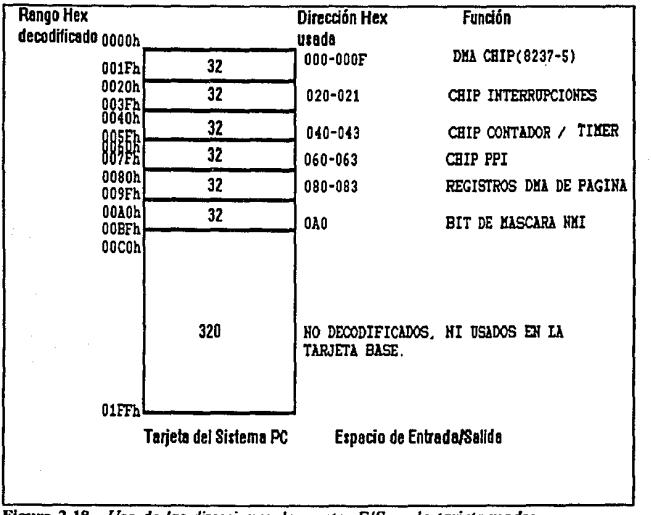


Figura 2-18 *Uso de las direcciones de puertos E/S en la tarjeta madre.*

La figura 2-19 muestra una tabla que define el uso que hace la IBM PC, de todo el espacio de direcciones de puertos I/O, localizado en el rango 000h a 03FFh.

Debe señalarse que debido a que IBM y otras compañías proveen cada vez nuevos desarrollos de tarjetas de aplicación para la PC, éstas direcciones decodificadas pueden estar usadas. Debido a que esto ocurre a una tasa muy elevada, es imposible mantener un mapa exacto del uso de los puertos. Si un sistema no usa alguno de esos dispositivos o tarjetas de desarrollo entonces nosotros podremos usar esas direcciones

RANGO HEXADECIMAL	USO
000-00F	CHIP DMA
020-021	INTERRUPCIONES 8259A
040-043	TEMPORIZADOR 8253A
060-063	PPI 8255-5
080-083	REGISTROS DE PÁGINA DMA
0Ax*	REGISTRO DE MÁSCARA NMI
0Cx	RESERVADO
0Ex	RESERVADO
100-1FF	NO USABLE
200-20F	CONTROL JUEGOS
210-217	UNIDAD DE EXPANSIÓN
220-24E	RESERVADO
278-27F	RESERVADO
280-2E7	RESERVADO
2F8-2FF	COM2
300-31F	TARJETA PROTOTIPO
320-32F	DISCO FIJO
378-37F	IMPRESORA
380-38C	COMUNICACIONES SDLC
3B0-3BF	COMUNICACIONES SINC
3A0-3A9	COMUNICACIONES SINC
3B0-3BF	MONITOR MONOCR/IMPRES
3C0-3CF	RESERVADO
3D0-3DF	CONTROL GRÁFICAS
3E0-3E7	RESERVADO
3F0-3F7	DISKETTE
3F8-3FF	COM1 (COMUNIC ASINC

Figura 2-19 *Uso del espacio de direcciones de puertos E/S en la PC.*

decodificadas en nuestros propios diseños. Esto es válido siempre que no se tengan otros planes para ocupar tales puertos.

2.11 Uso del mapa de memoria

El procesador 8088 soporta un espacio de direccionamiento de memoria de 1 megabytes de memoria. IBM ha especificado ciertas porciones de uso para este espacio. Esta sección resume la utilización de este espacio.

El diseño de la PC utiliza tanto los límites superior como el inferior del espacio de direccionamiento del CPU 8088. Adicionalmente, los adaptadores de despliegue de la PC (monitores) usan una porción del espacio de direcciones para el almacenamiento temporal de video (buffers).

En la parte más alta del espacio de direccionamiento, se decodifica la memoria ROM. Un total de 64 Kbytes es decodificado de los cuales 40 Kbytes es ocupado por la ROM. Este espacio de direcciones es decodificado en la tarjeta madre y no debe ser usado por otros dispositivos conectados en los ranuras de expansión.

La ROM contiene el PC BIOS (Basic Input / Output System), diagnósticos, el sistemas operativos de cassette, y un interpretador BASIC. Una memoria ROM adicional de 8 Kb puede ser conectada al enchufe que se incluye en la tarjeta madre. Los restantes 16 K en el bloque de 64 K que son decodificados no son usados y por lo tanto están reservados para futuras aplicaciones.

La memoria RAM esta localizada en las direcciones bajas del espacio de direccionamiento de memoria. Los primeros 640 Kb de RAM es decodificada en la tarjeta madre y cualquier RAM adicional debe ser sumada mediante una tarjeta conectada en algún slot de expansión. Cuando se carga el Sistema Operativo [DOS, Disk Operating System], se instala en los primeros 12 Kbytes del espacio de RAM del sistema. En el momento de cargar un programa, por ejemplo DEBUG, se usan unos 6 Kb adicionales. Si en lugar de ello se carga el lenguaje BASIC, se utilizan 10

Kbytes. Por ello, en un sistema de 640 Kb de memoria, tal carga de programas nos dejaría un espacio de aproximadamente 618 Kbytes, ya que,

(640 - 10 - 12) Kbytes = 618 Kbytes

debe notarse que estos son valores aproximados y que ellos cambian con cada nueva versión del sistema operativo.

Cada uno de los adaptadores de video de la PC decodifica un bloque de 32 Kbytes de almacenamiento para usarlos en la regeneración de la pantalla. La tarjeta de gráficas en color actualmente utiliza solo 16 Kbytes de los 32 K bytes que decodifica. El bloque restante de 16 Kbytes no debe ser usado por el sistema. La tarjeta monocromática de video usa sólo 4 de los 32 Kbytes que decodifica. En forma semejante al caso anterior, los restantes 28 Kbytes no deben ser usados.

Debe señalarse que cuando uno de los adaptadores de video no se encuentre instalado en el sistema, el espacio de memoria que usaría si estuviese presente, se hallaría libre y *podría ser* usado por el sistema.

El diseño de la PC ha reservado ciertas áreas del sistema de memoria para futuras mejoras. Para asegurar la compatibilidad con futuros dispositivos, se debe *evitar,* el uso de esas áreas reservadas del espacio de direccionamiento de memoria. La figura 2-20 ilustra el mapa de memoria, su uso y las áreas reservadas para el sistema de memoria de la IBM PC.

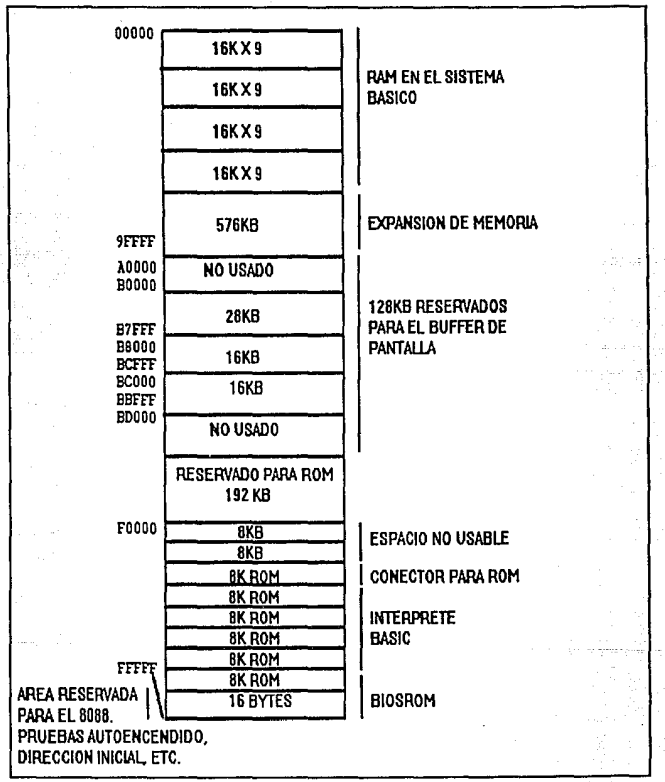


Figura 2-20 *Mapa de memoria de la IBM PC.*

2.12 Estados de espera

Un problema común cuando se conectan interfases al bus de la PC, es que la velocidad de los ciclos del bus de la PC concuerde con el diseño de la interfase. No es raro ver a una interfase operar a una velocidad menor a la que opera la PC. Por ello el diseño de la PC provee herramientas que solucionan este problema. Una señal llamada READY en el bus del sistema, puede ser usada para extender la longitud de un ciclo de bus para que se ajuste a una interfase de menor velocidad o para detener el ciclo de bus hasta que se sincronise con un ciclo de la interfase.

Como fue señalado inicialmente, todos los ciclos de bus del 8088 son normalmente de una longitud de 4 pulsos de reloj y son descritos como T1 a T4. En algunos ciclos de bus, el hardware de la PC automáticamente inserta un pulso de reloj extra, llamado TW. La señal READY puede ser usada para insertar nuevos o adicionales estados de espera.

Como el diseño que se va a desarrollar en este trabajo de tesis va a operar a velocidades superiores a las de la PC, no es necesario entrar en detalles de la generación de estados de espera.

2.13 Transferencia de datos en alta velocidad

Una de las principales consideraciones en muchos tipos de aplicaciones de interfase es la velocidad de transferencia entre la PC y el dispositivo. Es importante determinar las capacidades de transferencia de la PC al usar diferentes técnicas, pues el diseño debe hacer uso del método que proporcione mejor rendimiento a la aplicación.

La técnica que se usa con mayor frecuencia es la programación simple de E/S. Usando ésta técnica, la transferencia de datos es hecha bajo el control de un programa.

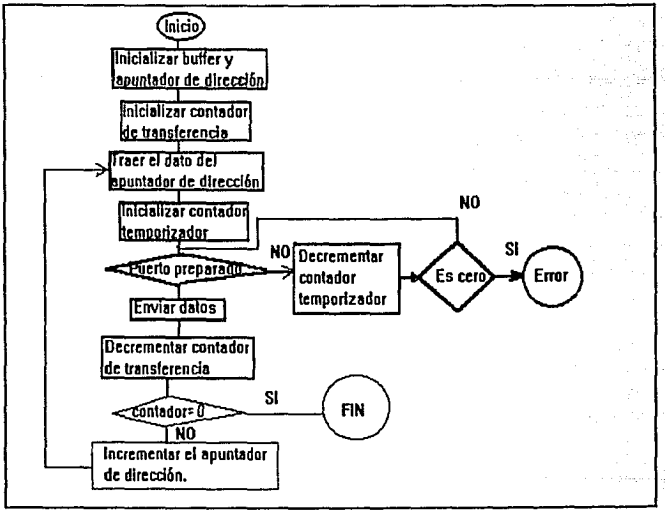


Figura 2-21 *Flujograma mostrando un ciclo típico de transferencia.*

Un ciclo típico de transferencia es mostrado en la figura 2-21. Ahí los datos son transferidos desde un almacenamiento en memoria (buffer) de la PC hacia una dirección de puerto de E/S. La malla tiene varias funciones para llevar a cabo el envío simple de datos. Primero, debe mantener una dirección de memoria apuntando al buffer y entonces incrementar esta dirección en cada transferencia. Segundo, el

ciclo debe mantener un contador de el número deseado de bytes de datos que van a ser transferidos y probar el ciclo de conteo para una condición de fin. Tercero, si el puerto de E / S no puede aceptar los datos tan rápido como el ciclo puede proveer de datos, se deben tomar las consideraciones necesarias para que el ciclo pregunte por una señal que le permita saber si el dato fue tomado.

En muchos casos, la transferencia de datos puede ser tan lenta que el BASIC puede llevarla a cabo. Por ejemplo, podemos usar la siguiente rutina en BASIC.

```
10 DIM BUFFER(1000)
20 FOR CONTADOR = 0 TO 1000
30 WAIT &H3BC,&H01
40 BUFFER(CONTADOR) = INP(&H3BD)
50 OUT &H3BC,0
60 NEXT
70 END
```

Este ciclo transfiere 1000 bytes de datos del puerto de entrada cuya dirección es 03DB en hexadecimal y los almacena en el arreglo llamado BUFFER. La línea 30 es usada para esperar hasta que el bit cero en el puerto de entrada 03BC sea activo. Este bit es usado para indicar que el dato en el puerto de entrada es válido y que puede ser leído. El enunciado de la línea 40 lee el dato presente en el puerto y lo almacena en el buffer. La línea 50 limpia la señal de dato-preparado en el registro 03BC, indicando al adaptador que el dato ha sido leído y que se puede cargar un nuevo dato en el puerto de entrada.

La eficiencia en la transferencia de datos puede ser mejorada usando un programa compilado. Sin embargo, para los casos en los que se desea una alta velocidad se puede optar por usar el lenguaje ensamblador del 8088. Finalmente, se puede usar la facilidad de DMA, con la cual es posible la transferencia de datos a una velocidad de 476 kilobytes por segundo.

2.14 Puertos y tarjetas para interfase

Para concluir este capítulo se presentan dos interfases que existen en forma estándar en la computadora personal IBM. Una de ellas es la interfase serie RS-232-C; el otro es el puerto paralelo para impresora. Para muchos proyectos de interfase, puede ser posible usar estos puertos.

Interfase RS-232

RS-232 es un estándar muy popular, típicamente usado para conectar equipo de proceso de datos (DTE: Data Terminal Equipment) a equipo de comunicación (DCE: Data Communication Equipment). Por ejemplo, terminales a modems. El estándar RS-232 cubre las características funcionales mecánicas y eléctricas de esta interfase. Dispositivos tales como terminales, plotters, analizadores lógicos, unidades de cinta e impresoras típicamente tienen una interfase RS-232. Si se desea conectar una aplicación de interfase al RS-232, será necesario convertir las señales de nivel lógico TTL a un nivel no TTL, requerido por este estándar.

El tipo se transmisiones soportadas por éste estándar incluyen tanto simplex, medio-duplex y duplex-total[6], para operación síncrona y asíncrona[7].

RS-232 opera a velocidades de hasta 20 Kbaudios por segundo (Kbps). Las velocidades estándares son 1200, 2400, 4800, 9600 y 19200 bps. La longitud

---

[6] Simplex: comunicación en una sola dirección. Media-duplex: comunicación en dos sentidos pero no al mismo tiempo. Duplex-total: transferencia de datos en ambos sentidos al mismo tiempo.

[7] En las comunicaciones síncronas los caracteres se transmiten en grupos; esto se logra con la implementación de una señal de reloj común en los equipos transmisores y receptores. La comunicación asíncrona consiste en la transmisión y reconocimiento de un solo carácter a la vez, el cual puede empezar y terminar en forma aleatoria.

recomendada es de 15 metros o menor, aunque es común encontrar longitudes de 30 metros. Con mucho cuidado en la calidad del cable (baja capacitancia) se puede trabajar hasta con 150 metros.

Puerto paralelo

El puerto paralelo, también llamada interfase Centronics tiene un conector de 36 conductores. Su uso esta enfocado a la transmisión de datos hacia impresoras.

El puerto paralelo también puede ser usado como un punto de entradas y salidas digitales de propósito general para interfazar dispositivos. Esta interfase provee un registro de salida de 8 bits que también puede ser usado como entrada[8], un registro de salida de 4 bits que puede ser cambiado a un registro de entrada, un registro de entrada de 5 bits, y un registro de salida que puede ser habilitado para generar una interrupción de nivel 7. La tarjeta esta diseñada para que su dirección pueda ser cambiada con otro puerto de impresión en el mismo sistema (LPT1 o LPT2).

El puerto paralelo normalmente decodifica las direcciones hexadecimales de puertos E/S 0378, 0379 y 037A. Estas direcciones pueden ser transladadas a 0278, 0279 y 027A, mediante interruptores que provee la tarjeta.

[8] Esta posibilidad implica cambiar la conexión física de algunos dispositivos que integran la tarjeta de puerto paralelo.

3
Diseño del Banco de Memoria y su Interfase

Introducción

En éste capítulo se desarrolla el análisis y diseño del banco de memoria y su interfase. Para ello se presentan las técnicas de decodificación de puertos y memoria. Se examina y diseña la interfase entre el banco de memoria y la ranura de expansión de la computadora personal. Así mismo se realiza el acoplo hacia el sistema de desarrollo de controladores microprogramados.

3.1 Técnicas de interfases digitales

El método más usado para interconectar dispositivos digitales a una computadora personal, es a través del uso de registros de entrada/salida programados.

Con registros digitales de salida, el microprocesador puede escribir datos en el registro, tratando el registro como un puerto E/S o como una localidad de memoria. En forma similar, son tratados los registros de entrada. Estos se usan para tomar muestras del estado de las señales conectadas a sus terminales. Por ejemplo, si un programa quiere determinar si un interruptor ha sido abierto, tal interruptor debe ser ligado como entrada del registro y ser leído para poder determinarlo. Puede considerarse a un registro digital de entrada como una localidad de memoria o como un puerto de E/S que ha sido cableado a localidades individuales de bits. En general, los registros de E/S permiten al procesador sensar información relacionada al mundo exterior y emitir señales que causen acciones fuera de la computadora.

La figura 3-1 muestra los componentes típicos de un diseño general de interfase y las funciones disponibles del microprocesador para implementarla. Como se ilustra en el diagrama, las tres funciones principales usadas en un interfase a un microprocesador son: (1) interrupciones usadas para sincronizar eventos externos, (2) DMA usado para transferencia de datos a velocidades altas a o hacia memoria, y (3) registros digitales de entrada/salida usados para sensar y controlar los circuitos de interfase o la interfase directamente conectada. Frecuentemente hay funciones que no pueden fácilmente ser realizadas por el programa del sistema usando registros digitales de entrada/salida directamente. Esto es porque no es lo suficientemente rápido o porque requiere demasiado software para una ejecución eficiente. El último elemento de un diseño para interfase es la conexión al mundo real.

Desafortunadamente, para los diseñadores, el mundo real muchas veces no es digital y las señales desde y hacia tales dispositivos finales requieren conversión a otras

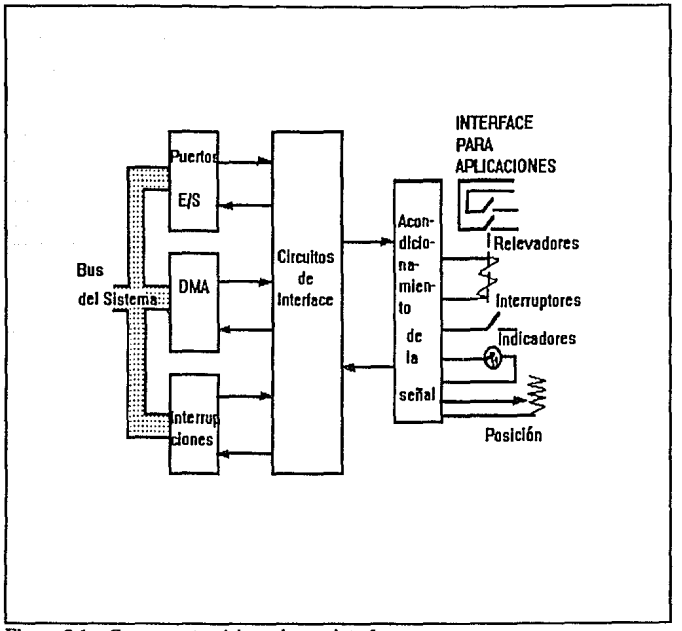


Figura 3-1 *Componentes típicos de una interfase.*

formas, tales como relevadores, sensores de interrupción, indicadores, niveles de voltaje no digitales, etc. En la figura 3.1 se incluye ello en la sección de acondicionamiento de la señal.

Ahora bien, en el diseño del banco de memoria se puede hacer uso o no, de los tres funciones principales mencionadas. Esto va a depender del enfoque que se le dé al problema. Vamos entonces a analizar en detalle la problemática que presenta el diseño del banco de memoria y su interfase.

3.2 Especificación del problema

En el capítulo 1 se definieron las características que el banco de memoria debe poseer,

1. Un tamaño de palabra de 80 bits.
2. 1024 direcciones.
3. El tiempo de acceso debe ser menor a 210 ns.

El banco de memoria RAM sustituye a un banco de memoria EPROM, que debe acoplarse a las señales del bus de expansión de la PC y conectarse al Sistema de Desarrollo de Controladores.

Estas necesidades involucran interfases digitales programadas. Esto significa que no se requiere convertir señales digitales a otro tipo, tales como analógicas.

Un factor importante a definir es la distancia a la cual se encontrará el Sistema de Desarrollo. Debido a que se trata de un sistema experimental, y dado que su tamaño físico es reducido, es posible tenerlo a un lado de la PC. Por ello, la longitud del cable se puede limitar a 50 cms.

En éste capítulo se diseñará el banco de memoria. En el próximo capítulo se desarrollará el programa que dará vida al banco.

3.3 Análisis del problema

El banco de memoria debe ser una conjunto de chips de memoria conectados en paralelo. El tamaño de palabra de 80 bits se obtendrá al contabilizar las líneas de datos de todos los dispositivos de memoria que forman el banco. Por ello, si usamos dispositivos de memoria idénticos, para obtener la cantidad de chips requeridos, debemos dividir el tamaño de la palabra total entre el tamaño de palabra de cada uno de los dispositivos, esto es:

Cantidad de Chips de memoria = 80 / (# Líneas de datos en cada chip)

Ahora bien, en cuanto a las localidades de dirección, podemos decir que para tener acceso a 1024 localidades, debemos hacer uso de la ecuación siguiente:

$$1024 \text{ localidades} = 2^{(\text{líneas de dirección del chip})}$$

lo cual nos lleva a requerir un chip que posea 10 líneas de dirección.

El tamaño común de chips de memoria más cercano a las necesidades es de 2K x 8. con el cual se necesitarían 10 chips de memoria. Un modelo comercial es el HM6716, una memoria estática con organización de 2048 x 8 bits, y tiempo de acceso de 25/30 ns.

Ahora que hemos seleccionado un tipo específico de CI de memoria, podemos empezar a imaginar el banco de memoria, sin embargo, la conexión en paralelo de los dispositivos de memoria da como resultado un problema: ¿Cómo aislar las señales provenientes de la PC, de las señales enviadas al Sistema de Desarrollo?

Si usamos la memoria HM6716, cierto es que sus líneas de datos pueden ponerse en alta impedancia, pero que va a pasar cuando desde la PC se estén escribiendo datos?. La respuesta es sencilla, las líneas de datos no van a estar en alta impedancia, y por ello por el bus de datos que conecta también al Sistema de Desarrollo, se van a transferir las señales que pueden ocasionar salidas no previsibles del Sistema.

La solución a éste problema es usar algún dispositivo electrónico que permita aislar a la PC del Sistema de Desarrollo. Tal circuito es un separador (buffer), como el 74LS244. Sin embargo, se debe usar un separador entre la PC y la memoria y otro separador entre la memoria y el Sistema de Desarrollo. Obviamente este separador es por línea de datos, de control y de dirección. Esto implica un mayor número tanto de circuitos como de conexiones necesarias.

Por razones de velocidad, el banco de memoria debe localizarse físicamente mas cerca del Sistema de Desarrollo que de la PC. Esto es porque el reloj de la PC opera a 4.77 MHz, y el Sistema va a trabajar a velocidades superiores.

Es por esto que el diseño del banco de memoria debe hacerse en dos módulos. Uno de ellos será básicamente el conjunto de dispositivos de memoria, y el otro contendrá a los circuitos de interfase, decodificación y soporte.

Una vez que el banco de memoria se haya implementado en el Sistema de Desarrollo, tomará energía eléctrica del mismo. Esto permite que la PC puede ser desconectada sin afectar el funcionamiento del Sistema de Desarrollo.

Desde el punto de vista del mapa de memoria de la PC, el banco de memoria podemos localizarlo en un espacio de direcciones desocupado. Si analizamos el mapa de memoria de la PC, figura 2-20, del capítulo anterior, podemos darnos cuenta que el rango de direcciones A0000 a AFFFF (hex) se encuentra desocupado. En la misma

situación se encuentra el rango de C0000 a CFFFF. Cada uno de estos rangos tiene una longitud de 64K direcciones.

Ahora bien, el banco de memoria es de 1024 x 80 bits. Si descomponemos el banco en los diez chips de 1024 x 8, entonces podemos localizar cada uno de los chips en uno de los rangos de direcciones libres de la PC, lo cual ocuparía solo 10K direcciones de las 64K disponibles. De ésta manera los datos se podrían escribir en el banco de memoria en forma directa, dado que se podría decodificar y usar como una extensión de la memoria interna de la PC. Ello facilitaría la transferencia de datos.

Aunque el banco de memoria no use todo el espacio de direcciones de 64K, es preferible que todo el rango seleccionado esté libre. ¿Por qué?, bueno por que el primer chip quedaría decodificado en las direcciones X0000 a X03FF, el segundo chip de X1000 a X13FFF, y así sucesivamente, hasta X9000 a X93FF. Esto con el fin de que la decodificación desde el punto de vista hardware sea sencilla.

Hasta ahora se ha discutido la estructura de la memoria, veamos que pasa con las señales de control.

Para llevar a cabo el control del banco de memoria se pueden decodificar algunos puertos. Uno debe ser de entrada para sensar las condiciones presentes en el banco de memoria. El otro puerto debe ser de salida para controlar tanto la escritura de datos desde la PC, como la lectura de datos hacia el Sistema de Desarrollo.

A continuación se procede a entrar en detalle en el diseño de cada etapa del Banco de Memoria y su interfase.

3.4 Diseño del Banco de Memoria y su Interfase

Decodificación de Memoria

El microprocesador 8088 tiene un bus de direcciones de 20 bits, así que puede direccionar 2^{20} o 1,048,576 direcciones. Cada dirección representa un byte almacenado. Las líneas de dirección A0 a A19 son usadas para seleccionar la localidad de memoria deseada.

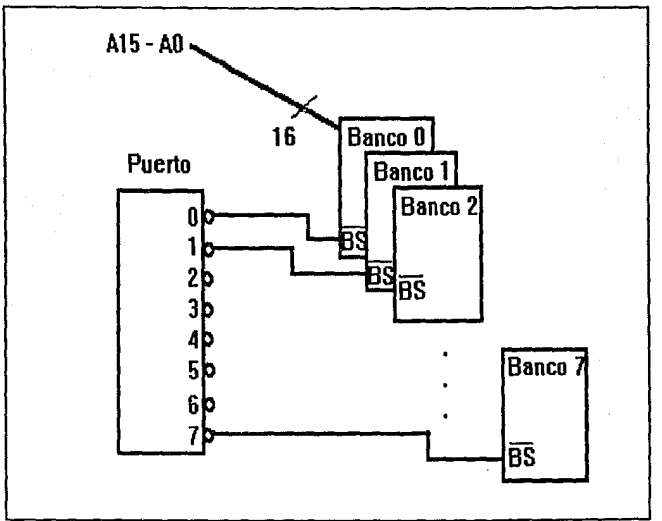


Figura 3-2 *Método para expandir memoria.*

Un método tradicional para expandir o decodificar memoria en una microcomputadora es el denominado intercambio de bancos. Un sistema que tiene solo 20 líneas de dirección puede direccionar 1MB de memoria. Como se demuestra en la figura 3-2, la implementación de hardware simple de selección puede permitir a un sistema accesar hasta 8 bancos de memoria de 1MB cada uno. Los circuitos son configurados de tal forma que cuando se inicia el sistema, éste usa el banco 0. Para cambiarse al banco 1, un byte en el puerto de selección deshabilita el banco 0 y otorga acceso al banco 1. En la práctica, éste tipo de diseños 'truco' son frecuentemente necesarios para conseguir mayores ventajas del diseño de los microprocesadores.

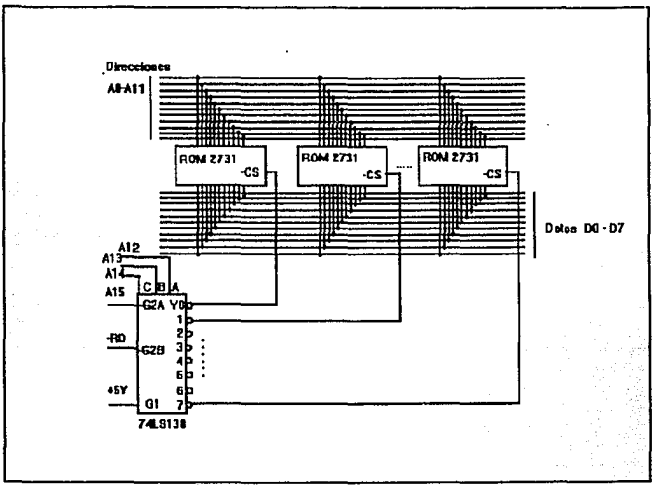


Figura 3-3 *Una variación del método para decodificar memoria.*

Una variación importante de éste método es el que se muestra en la figura 3-3. Esta figura muestra como ocho dispositivos de memoria pueden se conectadas en paralelo en un bus común de direcciones y en un bus común de datos. Viendo el esquema se puede notar que a diferencia del método anterior, aquí se usan líneas de dirección para la decodificación del banco de memoria adecuado. Este diseño reduce el número de direcciones al cual se puede expandir los bancos de memoria, pero al mismo tiempo simplifica la circuitería para la decodificación de puertos de E/S.

Asumiendo que el banco de memoria se desea decodificar como extensión de la memoria de la PC, entonces debemos tomar el método apropiado para decodificar la memoria.

La forma mas fácil para decodificar una dirección o un grupo de direcciones de memoria comienza con inspeccionar el actual uso del mapa de direcciones de memoria y encontrar un bloque de direcciones no usadas, y entonces, construir el apropiado circuito decodificador, fijo, para tales direcciones. La mayoría de las tarjetas de aplicación actuales usan esta técnica.

El problema con tal diseño es que el rango seleccionado de decodificación puede en el futuro ser ocupado por alguna aplicación comercial que el usuario desee agregar a su sistema. Es precisamente por ello que debemos desarrollar un método de decodificación que nos permita situar nuestra aplicación en diferentes rangos del espacio de direcciones, con tan solo cambiar la posición de algunos interruptores.

Por ello, se propone un diseño que se basa en el comparador octal SN/74LS688. En un lado del circuito comparador las líneas de dirección A16 a A19 son conectadas. En el otro lado, la salida de los microinterruptores DIP (Dual In Package) interruptores son conectados. Cuando el valor puesto en los interruptores DIP es igual al valor en el bus de direcciones, la salida comparadora es activada y usada para activar otros dispositivos, tal como el circuito decodificador SN/74LS138. Cuando un interruptor es abierto o cerrado, entonces se compara a un nivel alto en la asociada dirección del bus de direcciones.

Un esquema que muestra la estructura tal diseño se muestra en la figura 3-4.

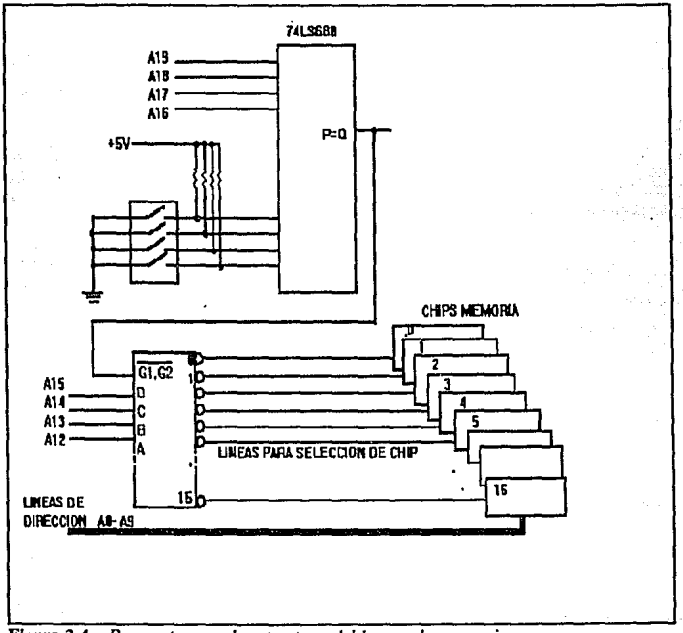


Figura 3-4 *Propuesta para la estructura del banco de memoria.*

Este enfoque nos permite situar el banco de memoria en 16 posibles rangos de direcciones, basados en las líneas de dirección A19, A18, A17 y A16, como se muestra en la tabla 1.

TABLA 1

NUMERO DE RANGO	A19	A18	A17	A16	RANGO HEXADECIMAL
0	0	0	0	0	00000 - 0FFFF
1	0	0	0	1	10000 - 1FFFF
2	0	0	1	0	20000 - 2FFFF
3	0	0	1	1	30000 - 3FFFF
4	0	1	0	0	40000 - 4FFFF
5	0	1	0	1	50000 - 5FFFF
6	0	1	1	0	60000 - 6FFFF
7	0	1	1	1	70000 - 7FFFF
8	1	0	0	0	80000 - 8FFFF
9	1	0	0	1	90000 - 9FFFF
10	1	0	1	0	A0000 - AFFFF
11	1	0	1	1	B0000 - BFFFF
12	1	1	0	0	C0000 - CFFFF
13	1	1	0	1	D0000 - DFFFF
14	1	1	1	0	E0000 - EFFFF
15	1	1	1	1	F0000 - FFFFF

Claro que no se debe usar un rango ya ocupado. Por ello se recomienda usar los rangos A0000-AFFFF ó C0000-CFFFF, que no son utilizados en el diseño de la PC.

Por su lado, las líneas A15 a A12 se utilizan para decodificar el número de chip adecuado. Con éstas 4 líneas de dirección podemos activar hasta 16 dispositivos de memoria, ver tabla 2. Claro que sólo necesitamos 10 de ellas, estas diez serán de la 0 a la 9.

TABLA 2

NUMERO DE CHIP ACTIVADO	VALOR DE LAS LÍNEAS DE DIRECCIÓN			
	A15	A14	A13	A12
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1

NUMERO DE CHIP ACTIVADO	VALOR DE LAS LÍNEAS DE DIRECCIÓN			
14	1	1	1	0
15	1	1	1	1

Las líneas A9 a A0 pueden entonces ser utilizadas para decodificar las 1024 localidades de dirección del banco de memoria. Esto es válido, pues 2^{10} da como resultado 1024 combinaciones posibles.

De ésta manera, todas las líneas de dirección son empleadas para comunicarse al banco de memoria.

Para lograr la escritura de la memoria desde la PC, también es necesario hacer uso de las señal de control MEMW. Con ésta señal será posible escribir en los datos en el banco de memoria.

Ahora bien, como fue analizado durante la sección anterior, se requiere de circuitos buffer que permitan separar las señales provenientes de la PC del Sistema de Desarrollo. Estos dispositivos electrónicos deben ser controlados desde el programa de aplicación, para que se activen en el momento adecuado. Esto se puede lograr mediante el uso puertos de Entrada/salida.

Decodificación de Puertos.

La decodificación de direcciones de puertos E/S es muy similar a la decodificación de direcciones de memoria. La principal diferencia es la inclusión de la señal AEN para indicar que se trata de una decodificación de puertos y no de memoria. Por ésta similitud, podemos entonces usar la técnica basada en microinterruptores DIP para situar las interrupciones en direcciones seleccionables.

Recordando lo dicho en el capítulo anterior, sobre el direccionamiento de puertos, tenemos que solamente las líneas de dirección A0 a A15 pueden ser usadas con este propósito. Además, las direcciones de puertos E/S disponibles en las ranuras de expansión se encuentran en el rango de 0200 a 03FF (Hex). Por tanto, en la decodificación de los puertos E/S, usaremos exclusivamente las líneas de dirección A0 a A9.

De éste conjunto de 10 líneas de dirección, las 7 líneas de mayor significado, esto es A9 a A3 serán usadas como entradas al circuito comparador octal SN74LS688, por un lado; por el otro lado se conectará los microinterruptores DIP.

La figura 3-5 muestra el diseño para la decodificación de puertos seleccionables de E/S. En este diseño, se permite direccionar un bloque de ocho direcciones de puertos que pueden ser movidas en el espacio de direcciones simplemente ajustando un nuevo valor en los interruptores DIP. Cuando el valor fijado en los interruptores es igual al presente en el bus de direcciones, la salida del comparador se activa y puede ser usada como la señal de control que selecciona un grupo de direcciones. Esta señal activa puede ser manejada en conjunto con las señales IOR y OWR para generar señales habilitadoras de registros entrada/salida.

Específicamente la salida del comparador se usa para habilitar el dispositivo SN74LS245, separador[1] bidireccional de datos y, también para activar los circuitos decodificadores SN74LN138 que actúa con las señales A0, A1, A2, IOR e IOW. La señal IOR también es usada para definir la dirección activada en el separador bidireccional SN74LN245.

[1] El separador es dispositivo electrónico que puede permitir o no, el paso de una señal digital, 1 ó 0, en función de una señal de control. Cuando no se permite el paso de la señal, la salida del dispositivo presenta el estado de "alta impedancia".

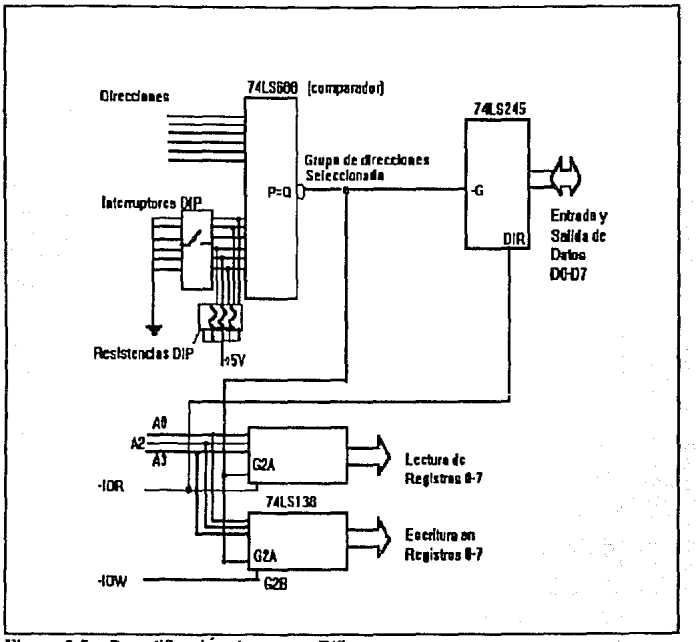


Figura 3-5 *Decodificación de puertos E/S.*

Esta configuración permite situar los puertos de E/S en un amplio rango de direcciones, tal como lo muestra la tabla 3.

Tabla 3

A9	A8	A7	A6	A5	A4	A3	RANGO DECODI-FICADO
0	0	0	0	0	0	0	000 A 007
0	0	0	0	0	1	0	010 A 017
0	0	0	0	0	1	1	018 A 01F
0	0	0	0	1	0	0	020 A 027
..	..	..	..	..	..	..	..
1	0	0	0	0	0	0	200 A 207
1	..	..	..	..	..	..	..
1	0	0	0	1	0	0	220 A 227
1	..	..	..	..	..	..	..
1	0	1	0	0	0	0	280 A 287
1	..	..	..	..	..	..	..
1	1	0	0	0	0	0	300 A 307
1	..	..	..	..	..	..	..
1	1	1	1	1	0	1	3E8 A 3EF
1	1	1	1	1	1	0	3F0 A 3F7
1	1	1	1	1	1	1	3F8 A 3FF

Como se podrá observar, los rangos de puertos decodificados constan de 8 direcciones. Es necesario recalcar que aún cuando sea posible decodificar una extensa cantidad de puertos, no deben usarse localidades menores a 200 (hex). Se recomienda hacer uso de los rangos de puertos siguientes: 200 a 207, 280 a 287, ó 300 a 307, por no estar ocupados en el diseño de la IBM PC. Sin embargo, para cada PC en

particular se debe consultar su manual técnico, a fin de evitar tener dos dispositivos tratando de operar con la misma dirección, lo cual puede ocasionar desperfectos del sistema.

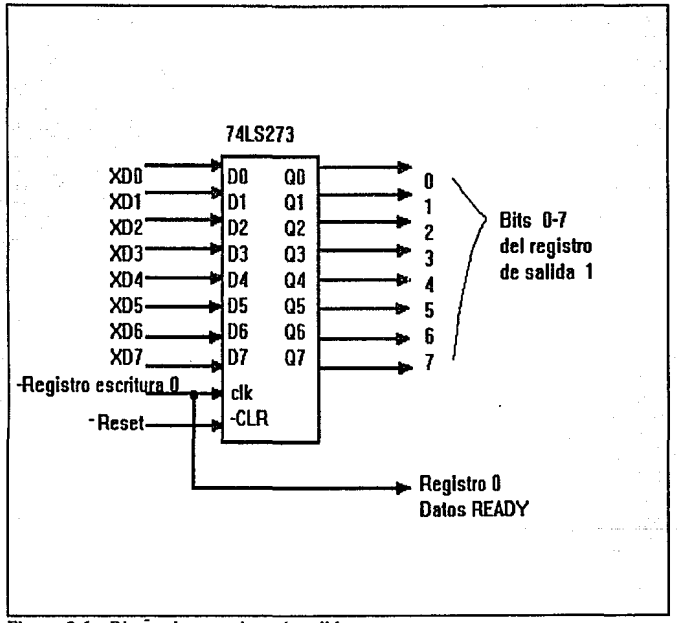


Figura 3-6 *Diseño de un registro de salida.*

Diseño del registro de salida

La figura 3-6 muestra la forma en que se puede diseñar un registro de salida. Aquí un dispositivo retenedor (latch) octal el SN74LS273, se usa para capturar el dato enviado por la instrucción OUT del ensamblador, o por la instrucción PORT del Turbo Pascal. El dato del bus es escrito en el flanco de bajada de la señal de control proveniente del circuito decodificador SN74LS138. Esta señal se activa al ejecutarse la instrucción OUT ó PORT, y se decodifica éste puerto.

Para escribir datos a un registro, primero se carga la dirección del puerto en el registro AL y entonces se ejecuta la instrucción OUT con la variable del puerto. Los datos que están en el registro AL aparecen entonces en el registro de salida. El siguiente es un ejemplo de código que escribe la cadena AA (hex) al puerto hexadecimal 0300.

```
MOV DX,0300    CARGA EL REG DX CON LA DIRECCIÓN DEL PUERTO
MOV AL,AA      CARGA EL ACUMULADOR CON EL DATO
OUT DX,AL      ESCRIBE EL DATO AL PUERTO
```

El mismo resultado se puede obtener usando Turbo Pascal[2].

Usando éstos comandos con los registros de salida es posible, bajo el control del programa manipular las señales de salida del registro y controlar el Banco de Memoria.

---

[2] Pascal es un lenguaje de programación de alto nivel, inventado por el Suizo Niklaus Wirth; recibió el nombre de PASCAL, en honor de Blaise Pascal (1623-1662) quien diseño y contruyó la primer calculadora. Turbo Pascal es una marca registrada del compilador de Pascal de la compañía norteamericana Borland International.

Cada uno de los bits de salida del registro será usado para establecer las condiciones de operación adecuadas para la transferencia de datos al banco de memoria, y después para la operación de la memoria con el Sistema de Desarrollo. La tabla 4 muestra el uso de cada una de la líneas de datos del registro de salida.

Tabla 4

LÍNEA DE DATO	FUNCIÓN
SP0	Activación buffers selección Chip de memoria a 5 V
SP1	Activación buffers de dirección A0-A9 de la PC
SP2	Activación buffers de dirección A0-A9 del Sistema
SP3	Activación buffers selección Chip de memoria de la PC
SP4	Activación buffers selección Chip de memoria a tierra
SP5	Activación buffers control escritura memoria de la PC
SP6	Activación buffers control lectura del Sistema
SP7	Activación buffers salida datos hacia el Sistema

El registro de salida debe ser llevado a un estado conocido cuando el sistema sea encendido. Esto es para que cuando el sistema encienda no provoque la actividad accidental de los dispositivos conectados. Esto se puede realizar usando la señal RESET, con lo cual los bits de salida se inicializan a cero en el momento de encendido de la PC.

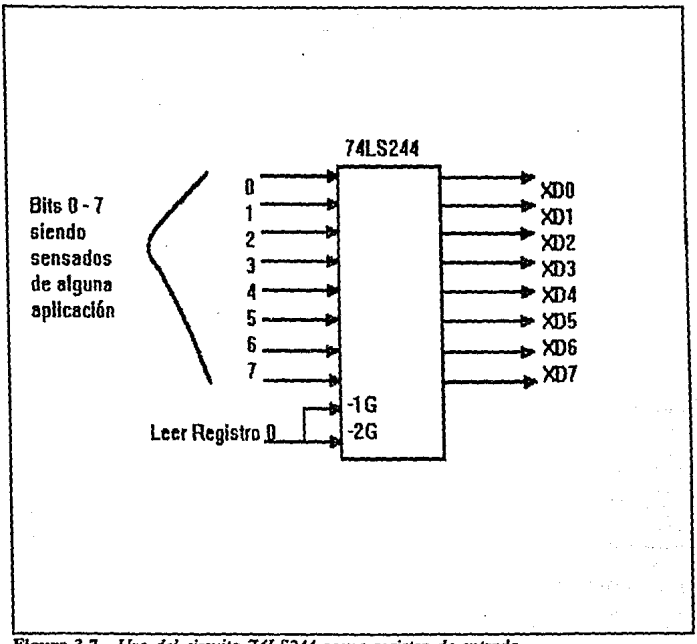


Figura 3-7 *Uso del circuito 74LS244 como registro de entrada.*

Diseño del registro de entrada

Sensar información desde una interfase normalmente es llevada a cabo usando un registro digital de entrada. La forma en que se puede diseñar éste tipo de registro se muestra en la figura 3-7. En éste diseño, los datos provenientes del mundo externo son reconocidos por el sistema de la PC cuando se decodifica el puerto y ocurre la señal IOR. Estas condiciones se pueden lograr al ejecutar la instrucción IN del ensamblador, o con la instrucción PORT del Turbo Pascal. Tales instrucciones toman el byte presente en el puerto decodificado y lo pone en un registro interno del microprocesador, donde puede ser operado por el programa.

Es importante notar que el dato recolectado a la entrada de éste tipo de registro sólo es una muestra tomada en el momento de ejecución de la instrucción de lectura. Por ello, cuando los eventos que se desean monitorear son de muy corta duración, es posible que se recolecten datos erróneos. En ese caso se debe usar un diseño de registro con retención. Dado que los datos que se van a sensar en el Sistema de Desarrollo son muy estables[3], no se analizará tal diseño con retención.

El flanco de bajada de la señal de escritura IOW puede ser usado como reloj para admitir el paso del bus de datos hacia el registro de salida en el SN74LS273. De igual manera la señal IOR es usada para admitir el paso de los datos provenientes del circuito sensor SN74LS244 hacia el bus de datos.

> Hay dos observaciones sobre el tiempo cuando se decodifican direcciones de puertos. La primera es al inicio de un ciclo de bus para puertos E/S. Si la decodificación del puerto tiene mucho retraso, puede ocurrir después de que las señales de control IOR u

[3] Son estables en el sentido de que se trata de señales discretas y no analógicas o contínuas. Además de que específicamente van a medir si existe polarización o no. No son no señales que continuamente cambien, provientes de alguna aplicación física.

IOW sean válidas. Esto puede resultar en la decodificación momentánea de otra dirección de puerto. Si ocurre ésta decodificación inválida, y se usa con otras señales de control, entonces se puede escribir en direcciones de puertos erróneos. El diseño de la PC permite un máximo de 92 nanosegundos en la decodificación.

La segunda consideración es sobre el tiempo es al final del ciclo de bus de E/S. Aquí, si la señal IOW es retardada y la decodificación es muy rápida, el flanco de bajada de la señal de escritura IOW puede escribir datos en una dirección que es decodificada por el siguiente ciclo del bus!. En el diseño de la PC, el retardo en la señal IOW debe ser menor a 200 nanosegundos. Sin embargo, el mayor tiempo de retardo está dado por la relación del flanco de bajada de la señal IOW para validar datos del bus. Si la señal IOW se retarda más de 120 nanosegundos, la dirección del puerto puede ser escrita con datos inválidos. Similarmente, si la señal IOR se retrasa, se reducirá el tiempo de acceso para lectura disponible para el puerto. Los diagramas de tiempo presentados en el capítulo 2 deben ser consultados para mayor detalle.

Leer datos provenientes de un puerto de entrada, es similar a la escritura de datos en un puerto de salida. El siguiente es el programa en Turbo Pascal necesario para leer el dato presente en el puerto 0301, y asignarlo a la variable *valorleido* de tipo *byte*.

```
PROGRAM LECTURA_DE_PUERTO;
VAR VALORLEIDO : BYTE;
BEGIN
  VALORLEIDO := PORT[$301];
END.
```

El mismo resultado se puede obtener con ensamblador.

Las líneas de datos tendrán significado especial que permitirá conocer el estado del Banco de Memoria, en cuanto a su polarización, y por tanto controlar la operación del Sistema de Desarrollo. La tabla 4 presenta el significado de las líneas de interés.

Tabla 4

LÍNEA DE DATO	FUNCIÓN
D0	Detectar polarización del Sistema (Vcc)
D1	Detectar tierra del Sistema (Gnd)
D2	No usada
D3	No usada
D4	No usada
D5	No usada
D6	No usada
D7	No usada

En base al análisis llevado a cabo en la sección anterior y con los conocimientos obtenidos respecto al funcionamiento de la PC y del Sistema de desarrollo, se proponen ahora los diagramas de configuración del banco de memoria y su interfase.

Las figuras en las páginas siguientes muestran tal propuesta.

PÁGINA EN BLANCO

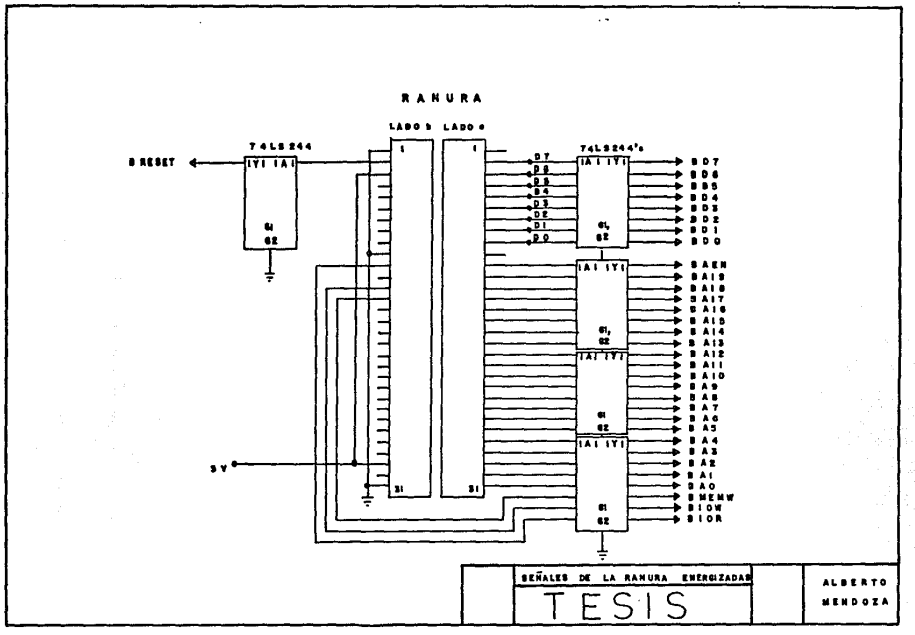


Figura 3-8 *Señales de la ranura energizadas a través de circuitos 74LS244.*

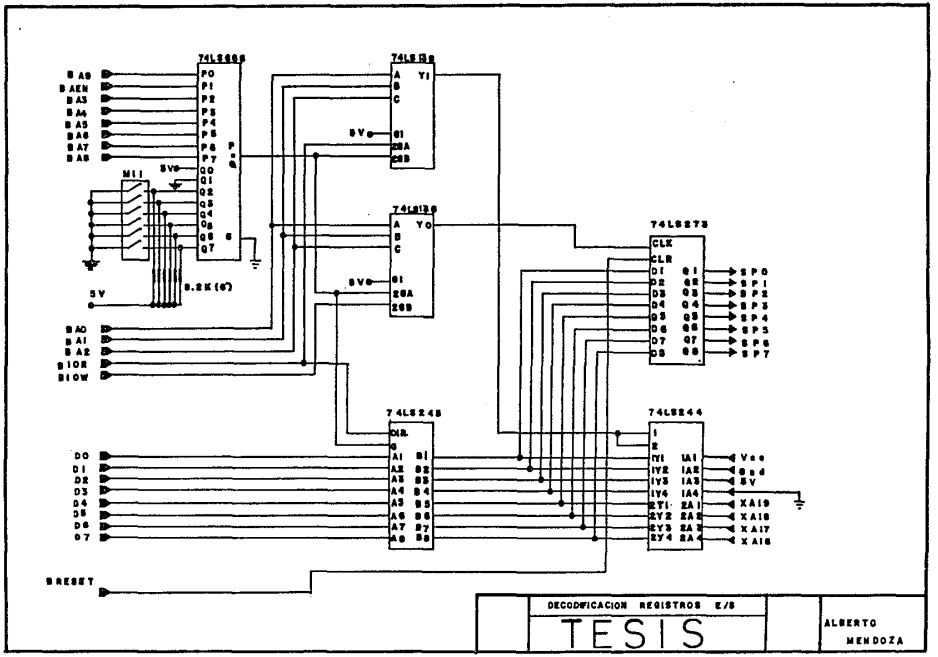


Figura 3-9 *Decodificación de registros de entrada/salida.*

La figura 3-8 muestra las señales provenientes de una ranura de expansión de la PC, que se utilizan en el diseño. Dichas señales se hacen transitar por un dispositivo (el 74LS244) que las energiza, habilitándolas para ser usadas en el diseño.

En la figura 3-9 se presenta el diagrama para la decodificación de registros de entrada/salida. Se puede notar que se dispone de un interruptor DIP, a fin de seleccionar un rango de direcciones desocupadas en una PC en particular, tal como se explicó en apartados anteriores. También cabe resaltar que únicamente se hace uso de un puerto de lectura y otro para escritura, dando libertad de usar los 14 restantes para el uso deseado.

Las señales para la decodificación y control de la memoria son presentadas en la figura 3-10. En tal figura, se puede notar la configuración de los circuitos 74LS244, los cuales se han interconectado de tal forma que permitan el manejo independiente de los circuitos de memoria.

El banco de memoria se presenta en la figura 3-11. Se puede notar el uso extensivo de circuitos 74LS244, quienes se encargan de separar las señales involucradas durante los procesos de escritura desde la PC, de las señales usadas durante el proceso de lectura desde el Sistema de desarrollo de controladores microprogramados.

Sin duda, el seguir una señal a través de los diagramas es difícil. Por ello, en la figura 3-12 presenta una vista global del diseño del banco de memoria y su interfase. En este diagrama he intentado mostrar la mayor cantidad de señales que permitan comprender fácilmente la operación del diseño propuesto.

PÁGINA EN BLANCO

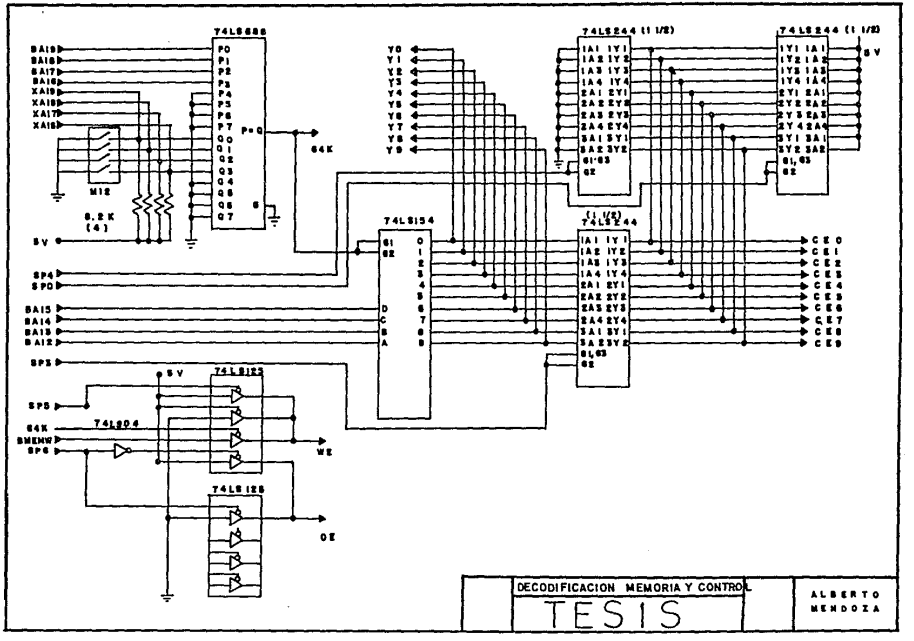


Figura 3-10 *Decodificación de la memoria y sus señales de control.*

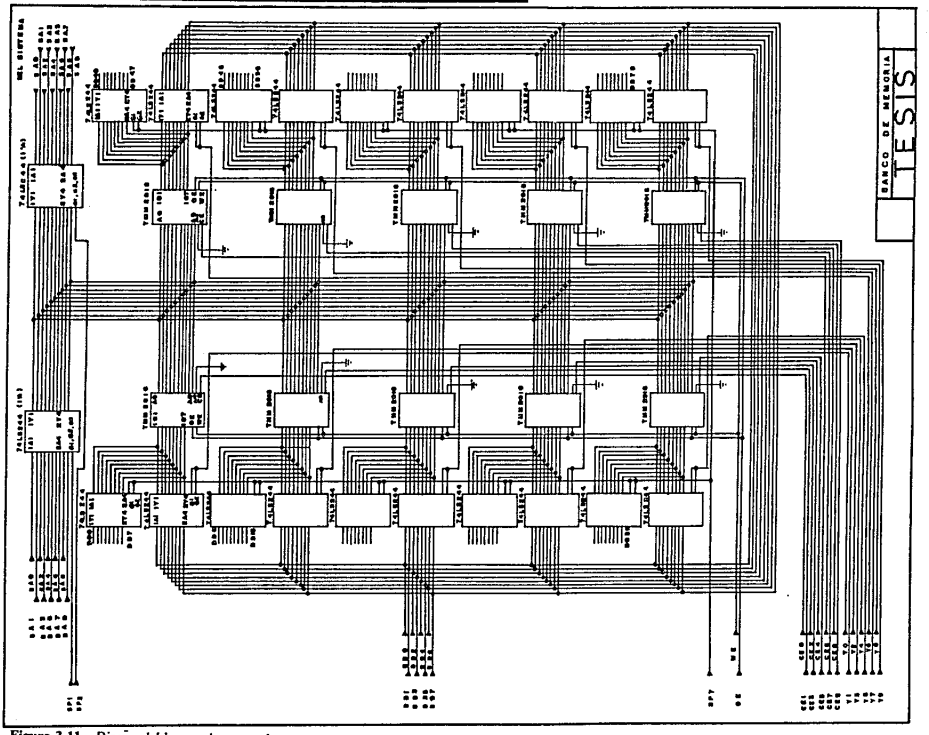


Figura 3-11 *Diseño del banco de memoria.*

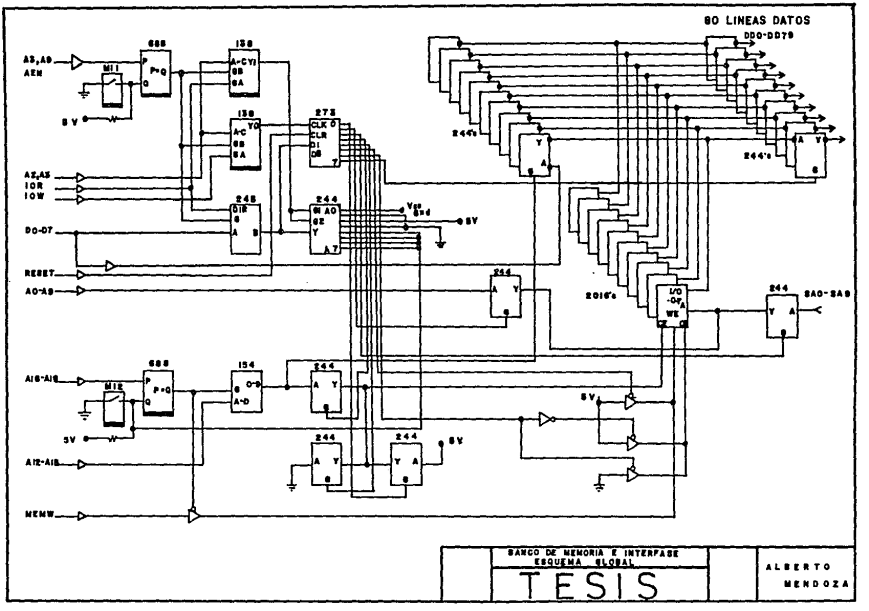


Figura 3-12 *Diagrama global del banco de memoria y su interfase.*

El diseño propuesto es fruto del análisis a detalle y del conocimiento de las necesidades. Al final de este trabajo, bajo el título **Materiales**, se presentan las hojas técnicas de los dispositivos utilizados en el diseño, así como también una lista de los materiales básicos requeridos para la construcción física del mismo.

3.5 Conexión al Sistema de Desarrollo y a la PC

El banco de memoria ha sido diseñado, y para comprobar su funcionamiento se alambró un circuito parcial en tarjeta experimental. Ahora se analiza como conectarlo al sistema de desarrollo.

Como se vio en el capítulo anterior, la conexión al sistema de desarrollo, consiste en la interconexión de las señales de dirección, datos y de control entre el banco de memoria y el sistema de desarrollo de controladores microprogramados.

Dado que en el Sistema de desarrollo ya se cuenta con dos conectores de entrada, uno para los datos y el otro para las direcciones, entonces basta con proveer al banco de memoria con dos salidas que "empaten" con sus correspondientes líneas. Se propone usar dos cables planos, con los conectores correspondientes en ambos extremos de cada cable.

De la misma manera, la interconexión entre la PC y el banco de memoria se propone hacerse con dos cables planos, en uno de ellos las líneas de dirección y en el otro, las líneas de datos. Las líneas de control serán repartidas entre ambos conductores.

4

Desarrollo del software de aplicación

Introducción

En este capítulo se desarrolla el programa que satisface los requisitos de transferencia de datos provenientes de la computadora personal hacia el banco de memoria.

Se inicia con una breve introducción a la programación de computadoras, así como la función que tienen los lenguajes de programación. Se presenta al Pascal como el lenguaje de programación con el cual se pretende desarrollar la aplicación. Luego se describen las técnicas empleadas en el diseño de programas.

Se procede entonces a realizar un análisis de los módulos en que se puede dividir el sistema, y las principales variables involucradas. Se avanza luego en la descripción las actividades que cada módulo debe realizar, separándolos en procedimientos y funciones cuando es necesario. Finalmente, se presenta un listado completo del programa fuente obtenido.

4.1 Programación de computadoras

Los programas para computadoras son un conjunto de instrucciones que le indican a la computadora como realizar una tarea específica.

La computadora ejecuta las instrucciones en forma secuencial[1], a menos que una instrucción la haga "saltar" a otra parte del programa y continuar el proceso desde ese punto.

Lenguaje de máquina

Lenguaje de máquina es el conjunto de códigos o instrucciones que "entiende" en forma directa la Unidad Central de Proceso (CPU).

Los códigos de las instrucciones se escriben en forma numérica en sistemas binarios, octales, decimales o hexadecimales. Estas códigos representan instrucciones relacionadas con el equipo, tales como "sumar los contenidos de dos registros" ó "brincar a una dirección".

Las primeras computadoras requerían que los programas se escribieran en este lenguaje porque era lo único que podían "entender". Pero hacer esto implica un esfuerzo mental sorprendente por parte de los diseñadores, lo cual conduce a múltiples errores y eleva los costos de producción de aplicaciones.

[1] Esto puede no ser válido en computadoras con múltiples procesadores.

Lenguaje ensamblador

Es una serie de programas que permiten traducir códigos mnemónicos, más fáciles de recordar (que una cadena de unos y ceros) como ADD, SUB y MOV, al código numérico "entendible" por el CPU.

El código resultante, llamado código objeto, es una serie de valores binarios que pueden ejecutarse directamente por la computadora.

Debido a que existe una relación directa entre el código nmemónico o código fuente y el código objeto, éste último producido por el ensamblador es muy compacto y eficiente. Sin embargo la escritura de código fuente es tediosa, necesita mucho tiempo y es propensa a errores.

Lenguajes de alto nivel

Un lenguaje de programación que se aleja bastante de la operación interna de la computadora y que genera varias instrucciones de bajo nivel por cada instrucción en el programa fuente es un lenguaje de alto nivel.

Los lenguajes de computación de alto nivel con frecuencia se parecen al inglés y fueron desarrollados para satisfacer varias necesidades, por ello es que existen múltiples lenguajes, tales como FORTRAN, COBOL, C, BASIC, RPG, etc.

Algunas de las ventajas de los lenguajes de alto nivel son:

- Transportar las aplicaciones de una máquina a otra sin tener que reescribir el programa fuente.

- Permitir que personas que no necesariamente sean expertas en los equipos de cómputo, escriban programas de aplicación.

- Eliminar la codificación repetitiva de funciones usadas comúnmente (funciones de entrada /salida, funciones trigonométricas, etc.).

- Aumentar la productividad de los programadores.

- Usar una plataforma de instrucciones consistentes con el tipo de aplicaciones a desarrollar.

Los programas escritos en un lenguaje de alto nivel deben convertirse a código ejecutable por la computadora para que puedan correr en ella. Esto lo hace un interprete o un compilador. Ellos son programas escritos para cada tipo de computadora

Interpretes

Los interpretes toman el código fuente y lo traducen directamente a la computadora utilizada, sentencia a sentencia y ejecutando cada una de ellas al traducirla. Cuando se requiere ejecutar nuevamente una sentencia, debe reconvertirse a código máquina. Esto se significa una ejecución más lenta, pero permite facilidad en el desarrollo de programas.

Compiladores

A diferencia del código fuente para un ensamblador, el código fuente para un compilador, no está relacionado directamente con el código objeto que produce. Como resultado, el código objeto es normalmente más abultado y menos eficiente que el producido por un ensamblador. Por otra parte, el código fuente puede desarrollarse mucho más rápidamente y mantenerse más fácilmente.

A un compilador se le da como datos un archivo que contiene el código fuente del programa de alto nivel y genera un programa en código de máquina. Ya convertido en código de máquina, el programa puede ejecutarse en forma independiente del compilador y del programa fuente. Existen diversos tipos de compiladores, cada uno de ellos identificado por el tipo de código de salida que genera. Aquí no se presentarán.

4.2 El Pascal como lenguaje

El pascal lo inventó el Profesor Niklaus Wirth, al final de los años 60's, en Suiza, y recibe el nombre del filósofo del siglo XVII, Blaise Pascal. Su objetivo era la sencillez y la sistematización enfocado a un entorno educativo. El resultado fue un lenguaje poderoso.

En los años siguientes, la potencia del lenguaje se hizo evidente para las compañías fabricantes de computadoras y para los diseñadores de programas. Actualmente, existe una proliferación del lenguaje Pascal y se han realizado muchas extensiones sobre la definición estándar del lenguaje.

Turbo Pascal

Turbo Pascal es una marca registrada de fábrica para un compilador de Pascal de la compañía Borland International.

Pero Turbo Pascal es mucho más que un simple compilador. Combina un editor, un compilador y un depurador, creando un entorno de desarrollo de software muy productivo.

Este es el lenguaje con el cual se va a desarrollar el programa de aplicación, objetivo de esta tesis.

El pascal como lenguaje estructurado.

El pascal es un lenguaje estructurado con algunas similitudes con el Algol y el C. La característica que distingue a un lenguaje estructurado es su capacidad para separar y almacenar del resto del programa toda la información e instrucciones necesarias para realizar un tarea definida. La base de éste método es el concepto de programas estructurados en bloques.

De esta manera, se utilizan módulos pequeños, fáciles de entender, para construir programas más grandes y complejos.

Las funciones y procedimientos son los bloques que constituyen el Pascal. Cada uno de ellos se hace tan independiente como sea posible mediante la declaración de variables locales que son conocidos exclusivamente en ese subprograma.

Este método se programación tiene una ventaja indiscutible. Los subprogramas pueden aprovecharse del método "divide y vencerás" aplicable a la solución de problemas. Con ello, un problema se divide en trozos cada vez más pequeños hasta que se trabaje con una rutina que pueda codificarse y probarse con facilidad.

Este método llamado descendente o refinamiento por pasos, en general se inicia identificando el problema. Luego se precisan las tareas detalladas, hasta reducir el problema al lenguaje de programación que se va ha utilizar. En otras palabras, se define el problema como una serie de bloques funcionales; luego se define cada bloque hasta alcanzar el nivel más simple.

Y es precisamente con ésta filosofía en mente, como se pretende dar solución a la problemática de captura y transferencia de datos hacia el banco de memoria.

4.3 Especificación del problema

En el capítulo 1, se mencionaron las características que debe posee el programa, para que la solución propuesta sea eficiente,

1. Permitir la captura de información en hexadecimal, realizando una validación de los datos introducidos. (Tal información corresponde a las microinstrucciones).
2. Proporcionar la facultad de almacenar la información en disco.
3. Conceder capturar los datos por rangos de direcciones, hasta un máximo de 16.
4. Admita seleccionar el tamaño de la palabra entre las siguientes tamaños: 4, 8, 16, 32, 64 u 80 bits (número de bits por dirección).
5. Permitir cargar, revisar y modificar la información contenida en disco.
6. Llevar a cabo la transferencia de datos entre la Computadora Personal y el Banco de Memoria.
7. Permitir imprimir formateado el contenido de un archivo de datos, seleccionando rangos de direcciones (máximo 16), y el tamaño de la palabra.

De las tareas mencionadas en la lista se puede deducir que los puntos 3 y 6, constituyen las partes fundamentales del programa.

4.4 Análisis del Problema

Empezaremos el análisis descomponiendo el problema en módulos funcionales, integrados por un menú principal, para luego construir submódulos basados en procedimientos y funciones que ejecuten las acciones específicas.

La lógica general del programa se puede definir en términos generales, de la siguiente forma:

> El usuario define los parámetros de la memoria, tanto de la PC como del banco, y los rangos que desea usar, para luego capturar las microinstrucciones, imprimirlas si lo desea y luego realizar la transferencia de datos hacia el banco de memoria. Finalmente almacena en disco su información para futuros usos.

Por ello, el sistema se puede dividir en los siguientes módulos:

0. Ayuda
1. Parámetros de la memoria
2. Rangos de trabajo
3. Captura de microinstrucciones
4. Impresión de microinstrucciones
5. Transferencia de datos al banco de memoria
6. Manejo de archivos
9. Terminar

Se debe notar que en esta clasificación se ha considerado una opción de ayuda y otra para finalización adecuada del programa.

A continuación se presenta una descripción más cercana de los conceptos que involucra cada módulo de este menú propuesto.

Los *parámetros de la memoria* consisten en la fijación de valores de variables, para su uso en otros módulos. Entre las variables consideradas se encuentra el espacio de direcciones libres que tiene la PC, los puertos de escritura y lectura posibles de usar, la dirección máxima del banco y los bits de salida para el banco. La dirección máxima del banco de memoria, es el espacio de direcciones que se desea accesar. Los bits de salida es el tamaño de la "palabra" para una microinstrucción.

Los *rangos* son una definición que van ha permitir llevar a efecto las operaciones de edición, impresión y transferencia dentro de ciertos limites del espacio de direcciones. Esto es muy útil cuando se requiera efectuar modificaciones a los datos en un determinado lugar, o cuando el diseño de una aplicación del sistema de controladores solamente involucre ciertas áreas de memoria. Por tanto, debe indicar cuantos rangos se van a utilizar y definir sus direcciones limite inferior y superior. Las especificaciones indican la existencia de 16 rangos como máximo.

Llevar a cabo la *captura* de las microinstrucciones o datos supone la definición previa de la cantidad de rangos y sus limites, así como de los parámetros de memoria. La entrada de datos deberá realizarse en forma secuencial, esto es, primero el rango uno, después el rango dos y así sucesivamente. Y dentro de cada rango, primero una dirección y luego incrementar esa dirección en uno y así hasta el limite mayor de ese rango.

Para la captura de una microinstrucción en particular se deben permitir la entrada de caracteres válidos, esto es de '0 'a '9' y de 'A' a 'F'. Cada uno de los datos se debe validar en el momento en que se introduzca.

La **impresión** de datos se llevará a cabo para los rangos que se han definido, con los parámetros especificados. La impresión se hará en forma matricial, direcciones de memoria contra microinstrucciones; todo en hexadecimal.

La *transferencia* de datos se efectuará en los rangos definidos y con los parámetros de memoria especificados. El usuario deberá preparar el sistema de desarrollo de controladores (dando energía al mismo, revisando direcciones de puertos E/S y de memoria, y verificando interruptores y conectores) para que la recepción de datos se realice adecuadamente. En caso de haber seleccionado el bloque de memoria disponible como $8000 se permitirá una transferencia hacia la memoria interna de la PC para un posible análisis.

Finalmente, se deseará hacer un *manejo de archivos* en disco con el fin de almacenar los datos introducidos, y tener la facilidad para recuperar un archivo previamente almacenado.

Ahora que se tiene una vista panorámica de las funciones que cada módulo debe poseer, se puede dar un esquema más detallado del sistema.

MÓDULOS DETALLADOS DEL SISTEMA

0. AYUDA
1. PARÁMETROS DE MEMORIA
    RANGO DE 64 KB DE DIRECCIONES LIBRES
        $8000, $A000, $D000
    PUERTOS E/S LIBRES
        LECTURA
            $221, $281, $301
        ESCRITURA
            $220, $280, $300
    DIRECCIÓN MÁXIMA
        $00F, $0FF, $1FF, $2FF, $3FF
    BITS DE SALIDA
        8, 16, 24, 32, 40, 48, 64, 80
    INICIALIZACIÓN DE MATRIZ DE DATOS
        0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
2. RANGOS
    CANTIDAD DE RANGOS
        1 A 16
    ESPECIFICACIÓN DE LOS RANGOS
        DIRECCIÓN INICIAL
            MAYOR O IGUAL A $000
        DIRECCIÓN FINAL
            MENOR A DIRECCIÓN MÁXIMA
3. ADICIÓN
    ENTRADA DE DATOS
        SIGUIENTE RANGO
        TERMINAR
4. IMPRESIÓN
    IMPRIMIR
5. TRANSFERENCIA DE DATOS
    TRANSFERIR
        AL BANCO, A MEMORIA INTERNA
6. ARCHIVO
    CAMBIAR NOMBRE DE ARCHIVO
    RECUPERAR ARCHIVO
    ALMACENAR ARCHIVO
9. TERMINAR

4.5 Organización de los datos y variables

Una vez que se tiene definido el problema que se va a resolver, el siguiente paso es identificar los tipos y variables principales para el programa empleando instrucciones en Turbo Pascal.

Identificación de los datos.

El esquema del sistema basado en módulos propuesto en la sección anterior es la base para identificar las variables significativas del programa.

Algunas de la variables son fácilmente deducibles, pero en otros casos, es necesario un poco más de tiempo para identificar de que tipo deben ser. Aquí analizaremos las variables cruciales.

Empezaremos por las variables que van a contener a las microinstrucciones. Qué tipo de estructura de datos será la más adecuada para almacenar ésta información?

Sabemos que vamos a introducir datos hexadecimales. Cada una de las microinstrucciones puede ser de hasta 80 bits, o sea, 20 caracteres hexadecimales. Por otro lado, sabemos que la cantidad de microinstrucciones máximas será de 1024. Esto nos sugiere que un arreglo[2] de caracteres podría ser el tipo ideal de estructura para almacenar las microinstrucciones:

```
Type    Matriz_datos = array [0..1023,0..19] of char;
```

[2] Arreglo es la traducción de **array** que se refiere a una serie de datos por renglones o por matrices.

y cada uno de los datos, o caracteres hexadecimales puede ser una variable de tal tipo:

```
Var     Dato : Matriz_datos;
```

Ahora bien, la definición que hemos hecho, almacena los datos, y en circunstancias generales, se requeriría de un proceso de comunicaciones para la transmisión de ellos hacia el banco de memoria. Sin embargo, la estructura del banco, fue diseñada para recibir los datos en forma directa. Esto significa que podemos escribir los datos en él, sin un proceso adicional.

Para ello podemos usar una declaración de Turbo Pascal que permite indicarle al compilador, donde deseamos que almacene una variable. Se trata de la palabra reservada ABSOLUTE, la cual tiene la siguiente estructura:

```
absolute segmento:offset;
```

donde segmento y offset son constantes de la dirección de memoria del segmento base y del desplazamiento dentro del segmento.

Entonces, los datos se pueden escribir en forma directa, chip por chip, y byte por byte, creando una variable:

```
Chip_memo_i : Array [0..1023] of Byte Absolute Despl:$i000;
```

donde i, es el número de chip del banco de memoria (del 0 al 9), y despl es el segmento base donde se encuentra decodificado el banco de memoria.

Debe notarse que en la declaración anterior "i", no puede ser una variable, dado que la estructura de ABSOLUTE indica que tanto el segmento como el desplazamiento deben ser constantes, por ello, se requiere crear una lista de 10 declaraciones de este tipo, uno para cada dispositivo de memoria.

Además, Despl debe fijarse como una constante:

```
const    Despl = $A000;
```

por lo cual si deseamos poder direccionar a diferentes segmentos de memoria, se debe repetir la declaración de variables. Esto implica código inevitablemente duplicado. Afortunadamente son pocas líneas.

Otro conjunto de datos importantes que debemos definir, es el de los rangos. Puesto que se requieren de hasta 16 rangos, cada uno de ellos con su dirección inicial y final, podemos crear un tipo:

```
Type    arreglo_rango = array [1..16] of integer;
```

para luego definir variables de las direcciones iniciales y finales, respectivamente:

```
var     arreglo1, arreglo2 : arreglo_rango;
```

Las variables anteriores, entonces, van a contener direcciones de tipo entero. Sin embargo, el tipo de direcciones a que el usuario microprogramador está acostumbrado son de tipo hexadecimal. Y ya que las direcciones válidas van de 000 a 3FF hex, se puede definir un tipo de arreglo:

```
Type    codigo_hexa = string[3]
        intervalos  = array[1..16] of codigo_hexa;
```

y luego una variable de ese tipo:

```
var     dirminimas, dirmaximas : intervalos;
```

En cierta forma, los rangos quedan doblemente definidos, pero al hacerlo así, se tiene la ventaja de que la entrada y despliegue en pantalla de los mismos es más fácil de manejar. De igual manera podemos proceder con otros datos cuya entrada sea hexadecimal y tengan representación decimal, por ejemplo:

```
var     despla : string[5];
```

Tomando como base todo lo anterior, las variables restantes son más fáciles de definir, por lo cual en la tabla 4-1 se presentan con su descripción y tipo

Tabla 4-1

VARIABLE	TIPO	DESCRIPCIÓN
maximadireccion	string[3]	Cadena hexadecimal de la mayor dirección a usar en el banco de memoria.
num_maximadireccion	integer	Valor decimal de la mayor dirección del banco
puerto_lectura	string[4]	Cadena hexadecimal del numero de puerto de entrada
num_puerto_lectura	word	Numero del puerto de entrada
puerto_escritura	string[4]	Cadena hexadecimal del número de puerto de salida
num_puerto_escritura	word	Número del puerto de escritura
maximosbits	string[2]	Cadena para la cantidad de bits de salida del banco de memoria
num_maximosbits	integer	Numero de bits de salida del banco
maximosrangos	string[2]	Cadena para el numero de rangos
num_maximosrangos	integer	Número de rangos
caracterrelleno	char	Carácter para inicializar la matriz de datos

Puede notarse que los nombres de las variables son extensos, pero su nombre revela el tipo de dato que se almacena y por ello es fácil reconocerlos. De igual manera se procederá al nombrar variables locales, procedimientos y funciones en cada uno de los módulos del sistema.

En las secciones que siguen se va a presentar un análisis y descripción de las funciones específicas de cada módulo. Sin embargo, no se pretende cubrir todo en detalle, sino más bien, destacar lo más relevante, describiendo el objetivo de los procedimientos y funciones involucrados. Como podrá deducirse del listado del programa fuente, muchos aspectos no requieren más que conocer el lenguaje pascal, para lograr una rápida comprensión.

4.6 Procedimientos generales

Pantalla

Llegando a éste punto en que empezaremos a desarrollar cada una de las actividades requeridas por el sistema, es necesario considerar la existencia de un problema adicional: el control de la pantalla.

El manejo de la pantalla de la PC, muchas veces resulta complejo. Para facilitar esto, se puede hacer uso de algunos procedimientos incluidos en el paquete de Turbo Pascal. En éste caso vamos a usar los procedimientos *Initialize, Openwindow y Closewindow*.

Initialize lleva a cabo la inicialización de variables apuntadores que requieren los otros dos procedimientos. *Openwindow* abre una "ventana" en la pantalla, y *Closewindow* cierra la última ventana abierta, redibujando la pantalla que existía antes de que *Openwindow* actuara.

Para poder hacer uso de tales procedimientos, se debe hacer la declaración siguiente:

```
Uses    Win;
```

Cadenas hexadecimales - enteros decimales

Por otro lado, dado que durante todo el programa haremos uso de cadenas hexadecimales que tendrán uso en ciclos como contadores, debemos implementar funciones de conversión cadena hexadecimal - entero decimal y viceversa.

Afortunadamente se dispone de una librería en Turbo Pascal, que nos elimina tal trabajo adicional. Por esto, haremos uso de las funciones **DECI** y **HEX**, incluidos en la caja de herramientas de Turbo Pascal, para transformar cadenas hexadecimales a enteros decimales y viceversa, respectivamente.

Esta caja de herramientas junto con las utilerías de ventanas pueden incluirse en el programa fuente, con la siguiente declaración:

```
{$I WINDOWS}
```

Ayuda en línea

El procedimiento de ayuda instantánea lo llamaremos **ponayuda**, y únicamente escribirá conjuntos de líneas de texto en la pantalla, dependiendo desde donde lo invoquemos. Este procedimiento será incluido en los módulos que se considere necesario.

Con la ayuda de todas éstas utilerías, entonces, nos enfocaremos a la solución de cada uno de los objetivos dentro de cada módulo, sin pérdida de tiempo.

4.7 Programa principal

El programa principal consistirá de unas cuantas líneas, gracias a que se han modularizado todas las actividades.

Empezaremos inicializando las variables globales mediante un procedimiento llamado *iniciavariablesglobales*, el cual dará valores por omisión a las variables principales. Esto es necesario debido a que cuando las variables son creadas, sus contenidos son inciertos, además de que algunas de ellas deben contener ciertos valores iniciales conocidos.

A continuación se muestra una tabla de módulos a los que se puede tener acceso, y corresponde precisamente el menú principal, ver tabla 4-2.

Tabla 4-2

Módulo	Procedimientos
Ayuda	ponayuda
Parámetros de memoria	menuparametros
Rangos de trabajo	menurangos
Captura de datos	menuedicion
Impresión	menuimpresion
Transferencia	menutransferencia
Archivo	menuarchivo
Terminar	-ninguno-

Cada una de las opciones del menú principal consiste de un procedimiento que deberá ser llamado si es seleccionado por el usuario. Los procedimientos serán descritos en las secciones siguientes.

Al terminar de ejecutar un módulo cualquiera, se debe regresar al menú principal para volver a seleccionar otra opción. Cuando se desee terminar, se podrá pulsar la opción 9, la cual debe pedir verificación de salida, a fin de evitar la terminación indeseada del programa.

4.8 Parámetros de memoria

Como puede apreciarse en el esquema del menú principal expuesto en secciones precedentes, la opción de parámetros de memoria sólo implica la selección de opciones prefijadas en el programa. Por ello es recomendable que a fin de que el usuario no cometa errores en la entrada, se le presente una lista numerada de opciones de las cuales deba seleccionar alguna de ellas.

Podemos normalizar las pantallas con un formato similar, a fin de que el usuario se acostumbre rápidamente al sistema. Por ello se opta por que la opción "0" sea siempre la de ayuda, y la opción "9" sea la de salida, tal como en el menú principal.

De esta forma, disponemos del "1" al "8" para posibles opciones. Así que ahora estamos en posibilidades de enumerar los submenúes y sus opciones:

LISTA DE OPCIONES PARA EL MODULO DE PARÁMETROS DE MEMORIA

0. AYUDA
1. RANGO DE 64 KB DE DIRECCIONES LIBRES
    1. $8000
    2. $A000
    3. $D000
2. PUERTOS E/S LIBRES
    1. LECTURA
        1. $221
        2. $281
        3. $301
    2. ESCRITURA
        1. $220
        2. $280
        3. $300
3. DIRECCIÓN MÁXIMA
    1. $00F
    2. $0FF
    3. $1FF
    4. $2FF
    5. $3FF
4. BITS DE SALIDA
    1. 8
    2. 16
    3. 24
    4. 32
    5. 40
    6. 48
    7. 64
    8. 80
5. INICIALIZACIÓN DE MATRIZ DE DATOS
    [0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F]
9. SALIR

Este menú lo llamaremos *menuparametros* y consistirá en presentar al usuario cada uno de los menúes descritos para que introduzca su selección. Esta selección se limita a caracteres válidos con los ciclos repet-until. Una vez que sea pulsado un carácter válido las variables involucradas se actualizarán.

Debe notarse que en la opción de inicialización de la matriz de datos, no se han enumerado las teclas válidas, por ello se debe validar al carácter introducido, y preguntar al usuario si desea escribir en toda la matriz de datos, el dato tecleado. Esta última verificación se lleva a efecto con el procedimiento **rellenarmatriz.**

4.9 Definición de rangos

Lo primero que se debe hacer en este menú, al que llamaremos *menurangos* es poder cambiar la cantidad de rangos, desde 1 hasta 16, y estos deben darse en decimal.

A continuación se deben poder listar los rangos actuales, es decir sus direcciones inicial y final de cada uno de ellos.

La última opción de éste menú es para poder cambiar las direcciones de cada uno de los rangos. Se debe presentar una lista numerada de cada uno de los rangos, desplegando sus direcciones. De ahí el usuario deberá seleccionar el número del rango que desee modificar. Con el rango seleccionado se le debe pedir que introduzca la dirección inicial y final. Estas direcciones deben validarse a ser hexadecimales y estar comprendidas en el rango de 000 a Num_maximadirección (máximo 3FF hex, o sea 1024 decimal). En caso de que se introduzcan valores erróneos el programa no debe permitir avanzar. Como puede notarse, esto es un proceso largo, por lo cual se puede separar en otro procedimiento. Lo llamaremos *cambiardefiniciones*.

Así que el menú de rangos queda definido de la siguiente manera:

OPCIONES PARA EL MODULO PARA DEFINICIÓN DE RANGOS

0. AYUDA
1. CANTIDAD DE RANGOS
   [1 A 16]
2. VER DEFINICIONES DE RANGOS
3. ESPECIFICACIÓN DE LOS RANGOS
   1. RANGO 1
   2. RANGO 2
   3. RANGO 3
   4. RANGO 4
   5. RANGO 5
   6. RANGO 6
   7. RANGO 7
   8. RANGO 8
   9. RANGO 9
   10. RANGO 10
   11. RANG0 11
   12. RANGO 12
   13. RANGO 13
   14. RANGO 14
   15. RANGO 15
   16. RANGO 16
   99. SALIR
   DIRECCIÓN INICIAL
      [$000...$NUM_MAXIMADIRECCION]
   DIRECCIÓN FINAL
      [$000...$NUM_MAXIMADIRECCION]
9. SALIR

4.10 Captura de Datos

La entrada de las microinstrucciones es una de las funciones principales del programa, pero que no requiere de múltiples submenúes, puesto que en los dos menúes anteriores ya se han fijado los parámetros de trabajo.

La captura de los datos debe ser programada para que permita la captura de una microinstrucción cuya longitud ya ha sido determinada con la selección de parámetros. Cada microinstrucción pertenece a una dirección específica del banco de memoria. Por ello se debe recorrer cada una de las direcciones de los rangos en forma secuencial, esto es, incrementado la dirección en uno. Así, hasta terminar con todos los rangos.

Por otro lado, se deben identificar errores a la hora de introducir los datos y permitir corregirlos. Además se debe conceder avanzar al siguiente rango de direcciones y tener la opción de terminar en cualquier momento.

En la práctica, puede resultar complejo, pues se debe identificar cada tecla pulsada y actuar en consecuencia.

Vamos a llamar *menuedicion* al procedimiento que lleva a cabo esta importante labor.

Su esquema podemos representarlo a continuación:

OPCIONES PARA EL MODULO DE CAPTURA DE MICROINSTRUCCIONES

ENTRADA DE LA MICROINSTRUCCIÓN
[1..9,A..F]

<ESC> ó <TOTALINTRODUCCIÓN DE MICROINSTRUCCIÓN>

<ENTER>,S	.	DATO CORRECTO
<-,N	.	DATO INCORRECTO
<ESC>	.	SIGUIENTE DATO
R	.	SIGUIENTE RANGO
T	.	TERMINAR

Lamentablemente, esto no es tan claro como en los menúes anteriores. La lucidez tiene que ser sacrificada ya que si las opciones se dan como números, el usuario puede inadvertidamente seguir tecleando pensando que esta introduciendo datos, cuando en realidad se encuentre seleccionando opciones equivocadas.

Así que para este menú se propone usar teclas adicionales: <ESC>o escape y "<-"o flecha izquierda, y las teclas alfabéticas "R", "T", "S" y "N".

Como puede apreciarse en el esquema, este pequeño submenú solamente se debe presentar si el usuario pulsa la tecla <ESC>o si ya ha terminado de introducir toda la microinstrucción.

Una última consideración acerca del procedimiento menuedicion, es la referente a la posición en que se van a introducir los datos. Dado que las microins-

trucciones se codifican en hexadecimal, sería posible introducir primero los caracteres menos significativos o al final. Para ser más coherentes con la forma en que se codifica un carácter hexadecimal, esto es, los bits más significativos a la izquierda,

$$0111 = 7$$

podemos entonces también introducir primero los caracteres más significativos (MSB).

4.11 Impresión de datos

El procedimiento encargado de la impresión de los datos va ha ser llamado *menuimpresion*, el cual básicamente tendrá una opción:

OPCIONES DEL MODULO PARA IMPRIMIR MICROINSTRUCCIONES

0. AYUDA
1. IMPRIMIR
9. SALIR

Como puede verse, el único camino es imprimir o no imprimir. Y como deseamos imprimir, entonces, mandaremos a la impresora las microinstrucciones correspondientes a cada dirección. Esto se puede lograr creando un procedimiento especial que se dedique a realizar tal actividad; lo llamaremos *impresión_de_datos*. Y consistirá de ciclos cuya salida a impresora sean las direcciones con sus respectivas microinstrucciones.

Un aspecto importante que debe considerarse antes de tratar de imprimir es verificar que la impresora se encuentra en condiciones de realizar tal operación. Esta

función la podemos llamar *impresoralista* y debe indicarnos si el procedimiento impresión_de_datos puede ejecutarse. En caso contrario, un mensaje debe mandarse al usuario informándole de tal situación.

4.12 Transferencia de datos

La transferencia de datos hacia el banco de memoria es un proceso muy importante del sistema. Tal operación debe realizarse después de que se hayan capturado las microinstrucciones.

Como se mencionó en la sección 4-5, el diseño del banco de memoria fue concebido para que la transferencia de datos fuera directa. Esto significa que no se requiere de un proceso de comunicación de datos entre la Computadora PC y el banco de memoria.

¿Cómo se logra entonces la transferencia de datos? Bueno, el banco de memoria es accesado como si fuera parte de la propia PC, y por ello al hacer las declaraciones de variables dirigidas a localidades absolutas de memoria, la transferencia es 'transparente' al programa. Esto quiere decir que el programa no se da cuenta de que va escribir en el banco de memoria externo.

Ahora bien, las microinstrucciones en hexadecimal fueron capturadas y almacenadas como caracteres, en código ASCII extendido (American Standard Code For Information Interchange: Código Normalizado Americano para el Intercambio de Información), y se deben transferir al banco de memoria como el equivalente binario del carácter hexadecimal, por ejemplo, veamos la tabla 4-3; para el carácter "D":

Tabla 4-3

	HEX	BINARIO	DECIMAL
Código ASCII	44	010001000	68
Representación deseada	D	1101	13

como puede observarse, no existe equivalencia directa.

Sin embargo, se pueden concatenar los datos en grupos de dos caracteres hexadecimales, y luego hacer que el compilador de Turbo Pascal los considere como un número hexadecimal y almacenarlos en variables de tipo BYTE.

Así entonces, si tenemos la microinstrucción siguiente:

0102030405060708090A

podemos descomponerla en los subgrupos siguientes:

01 02 03 04 05 06 07 08 09 0A

y luego concatenar cada subgrupo con el carácter "$":

$01 $02 $03 $04 $05 $06 $07 $08 $09 $0A

esto con el fin de convertir cada subgrupo en una variable numérica *dato_final* de tipo BYTE, con lo cual los subgrupos de dos caracteres se transforman en un número que ocupa 8 bits, ver la tabla 4-4:

Tabla 4-4

	DATO_FINAL	
CADENA	HEXADECIMAL	BINARIO
$01	01	00000001
$02	02	00000010
...	..	..
$0A	0A	00001010

El paso final consiste en colocar cada dato_final en la posición adecuada del banco de memoria.

Sabiendo que el banco de memoria está compuesto de 10 'chips' de 1K x 8 bits, entonces, podemos asignar cada dato_final a una variable de localidad absoluta, en función de cual dato estamos enviando:

CHIP_MEMO_i[DIR] := DATO_FINAL;

donde i es el número de chip de memoria, el cual va del 0 al 9; y DIR es la dirección en decimal de la fracción de microinstrucción actualmente asignada, la cual va desde 0 hasta 1023.

Todo esto lo podemos agrupar en un procedimiento al cual le llamaremos *seg1*.

Lo anterior debe repetirse hasta transferir la totalidad de la microinstrucción, y para todas las direcciones contenidas en los rangos.

En este momento, o quizá antes, podríamos preguntarnos por qué entonces en la captura de datos no se dirigió la transferencia directa al banco de memoria. Esto es debido a los beneficios que la modularidad nos brinda.

Por ejemplo, se pueden primero capturar todas las microinstrucciones en una PC que no posea el banco de memoria y después efectuar la transferencia en la PC que tenga instalado el banco de memoria.

Una consideración importante que debe ser señalada, es la referente a que la transferencia puede realizarse en dos modalidades:

A MEMORIA INTERNA
AL BANCO DE MEMORIA

Esto lo debe decidir el programa en función de que segmento de memoria seleccionó como destino en *menuparametros*, así entonces, si :

SEGMENTO	DESTINO
$8000	Memoria Interna
$A000	Banco de Memoria
$D000	Banco de Memoria

Cuando el destino sea la memoria interna, se puede analizar como quedan distribuidos los datos en una memoria consecutiva, la de la PC. Este análisis se puede llevar a cabo con el comando del DOS, DEBUG, y luego desplegando los segmentos de memoria,

```
-d $8000:0000
```

cuando se desea éste análisis es importante no tener cargados programas residentes en memoria.

Sin embargo, para permitir que el destino sea opcional se debe repetir el código del procedimiento *seg1*, debido a que la palabra reservada ABSOLUTE no admite que el segmento y desplazamiento sean variables, así que debemos fijar una constante, desp, con el valor del segmento, y posteriormente definir las variables de chip:

```
CONST
    DESPL = $D000;
VAR
    CHIP_MEMO_1 : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$1000;
    ...
```

Además, como Turbo Pascal no permite declarar variables en medio de código, entonces, completamente todo el código del procedimiento *seg1*, será repetido en los procedimientos *seg2 y seg3*. Donde la única diferencia será en la declaración de la constante DESPL.

Con ésta situación debemos crear otro procedimiento que engloble a tales procedimientos similares. Lo llamaremos procedimiento *transfiere*, y lo que hará es seleccionar el procedimiento de segmento (seg1, seg2, seg3) que se desee.

Hasta ahora se ha considerado que el banco de memoria se encuentra en perfectas condiciones de operación en el momento de iniciar la transferencia, pero el programa debe verificar que se cumplan las condiciones de operación básicas del banco de memoria: energía y direcciones de interrupciones y decodificación de memoria.

Lograremos tal verificación de condiciones de operación del banco de memoria con un procedimiento que llamaremos *hardware*, el cual leerá el puerto seleccionado en menuparametros y comprobará las condiciones. En caso de que no existan las condiciones adecuadas de operación, se debe enviar un mensaje al usuario.

Una vez que se encuentren las condiciones adecuadas en el banco de memoria, se debe proceder a abrir los dispositivos "buffer" para la transferencia, y cerrar las comunicaciones hacia el resto de los circuitos del Sistema de Desarrollo de Controladores Microprogramados. Esto se limita escribir en el puerto adecuado el número de control $F5. Pese a que se trata de una sola instrucción lo asignaremos a un procedimiento llamado *Da_control_a_PC*, para mayor claridad.

Finalmente, ya que se concluya la transferencia de datos, debemos abrir los "buffers" que comunican la memoria con el Sistema de adores, y cerrar las comunicaciones con la PC. Esto lo realizará un procedimiento llamado *Regresa_control_a_bitslices*, y consiste en escribir en el puerto de escritura el número de control $0B.

Todos los procedimientos nuevos nombrados en ésta sección podemos agruparlos en el procedimiento llamado *menutransferecia* el cual debe presentar las siguientes opciones:

OPCIONES DEL MODULO DE TRANSFERENCIA DE DATOS

0. AYUDA
1. CONTINUAR
9. MENÚ PRINCIPAL

'Continuar', realizará la transferencia involucrando los procedimientos necesarios y nos debe informar el resultado de tal operación. ¿Complejo?, quizá no, pero llegar a tal solución consume mucho tiempo..

4.13 Almacenado y Recuperación de archivos

Una vez que se han capturado o modificado las microinstrucciones, se deseará almacenarlas en disco para futuras revisiones o transferencias. El procedimiento que va a realizar tal actividad será llamado *menuarchivo*, y tendrá las siguientes opciones:

OPCIONES DEL MODULO PARA EL MANEJO DE ARCHIVOS

0. AYUDA
1. NOMBRE DE ARCHIVO
2. GUARDAR ARCHIVO
3. RECUPERAR ARCHIVO
9. SALIR

La opción de cambiar el nombre al archivo, debe permitir la entrada de un nombre de archivo, que incluya la unidad de disco y subdirectorio en el cual se desea localizar al archivo indicado. Por ejemplo, si se quiere accesar al archivo datos.dat, en la unidad de disco b, en el directorio programa, se debe introducir:

B:\PROGRAMA\DATOS.DAT

La opción de guardar archivo se puede alojar en un procedimiento cuyo nombre sea *archivado*, el cual debe verificar si existe el nombre de archivo indicado, y de no ser así escribirlo; en caso de que ya exista debe entonces preguntar al usuario si se desea sobreescribir tal archivo y actuar en función de la respuesta, sobreescribiéndolo ó no. Por tal razón conviene hacer un procedimiento llamado *escribeaarchivo* cuyo objetivo sea la escritura del archivo en disco, y que sea llamado cuando sea necesario.

Para la opción de recuperar archivo, podemos proceder de la misma manera que en el caso anterior, creando un procedimiento *recuperado*, que verifique si existe el archivo pedido, y que nos informe en caso de no existir. Si el archivo existe, entonces debe llamar a un procedimiento llamado *recuperaarchivo*, que lea el archivo deseado.

4.14 Listado del Programa

A continuación se presenta un listado completo del programa fuente resultante, bautizado como ***MICROPRO.PAS***. La compilación fue realizada en una Computadora Personal compatible con IBM y 640 KB de memoria. La versión del Compilador Turbo Pascal fue 5.5.

```
program Tesis;
{-------------------------------------------------------------------------}
{ Este programa permite capturar microinstrucciones en hexadecimal
  y transferirlas al banco de memoria para usarlas en el diseño de
  controladores microprogramados basados en bit-slices.
  Compilado con Turbo Pascal, version 5.5.
  Vea la documentación que se encuentra en el trabajo de tesis.
  Félix Alberto Mendoza Castillo, Junio de 1991. UNAM ENEP-ARAGON
  Para cualquier duda, llame al 765 08 59                          }
{-------------------------------------------------------------------------}
{Unidades usadas en el programa: }
{-------------------------------------------------------------------------}
USES
   Crt, Win, Printer, Dos;
{-------------------------------------------------------------------------}
{Declaraciones de tipos usados:  }
{-------------------------------------------------------------------------}
TYPE
   codigo_hexa4      =          string[4];
```

```
  codigo_hexa      =      string[3];
  matriz_datos     =      array[0..1023,0..19] of char;
  intervalos       =      array[1..16] of codigo_hexa;
  arreglo_rango    =      array[1..16] of integer;
{-------------------------------------------------------------------}
{Declaracion de variables:  }
{-------------------------------------------------------------------}
VAR
  dato                                :      matriz_datos;
  dirminimas,
  dirmaximas                          :      intervalos;
  arreglo1,
  arreglo2                            :      arreglo_rango;
  desp                        :       integer;
  nombrearchivo               :       string;
  despla                      :       string[5];
  maximadireccion             :       string[3];
  puerto_lectura,
  puerto_escritura            :       string[4];
  maximosbits,
  maximosrangos               :       string[2];
  num_maximosrangos,
  num_maximosbits,
  num_maximadireccion,
  x,y                                 :      integer;
  caracterrelleno,opcion      :       char;
  num_puerto_lectura,
  num_puerto_escritura                :      word;
  terminar                            :      boolean;
{-------------------------------------------------------------------}
{llamada a las utilerias de Turbo Pascal para el manejo de ventanas y
  procedimientos generales: }
{-------------------------------------------------------------------}
{$I WINDOWS}
{-------------------------------------------------------------------}
{Procedimiento para inicializar las variables globales: }
{-------------------------------------------------------------------}
```

```
procedure iniciavariablesglobales;
var
  contador          :          byte;
  i,j               :          integer;
begin
  initialize;
  terminar:=false;
  despla:='A000'; desp:=2;
  maximadireccion:='00F'; num_maximadireccion:=deci(maximadireccion);
  maximosbits:='8'; num_maximosbits:=8;
  maximosrangos:='1'; num_maximosrangos:=1;
  puerto_lectura:='$221';num_puerto_lectura:=$221;
  puerto_escritura:='$220';num_puerto_escritura:=$220;
  nombrearchivo:='NOMBRE.DAT';
  caracterrelleno:='0';
  for i:=0 to 1023 do for j:=0 to 19 do dato[i,j]:=caracterrelleno;
  dirminimas[1]:='000'; arreglo1[1]:=0;
  dirmaximas[1]:='00F'; arreglo2[1]:=15;
  for contador:=2 to 16 do
    begin
        dirminimas[contador]:='000';
        arreglo1[contador]:=0;
        dirmaximas[contador]:='000';
        arreglo2[contador]:=0;
    end;
end;
{-------------------------------------------------------------------------}
{Procedimiento para ayuda en linea: }
{-------------------------------------------------------------------------}
procedure ponayuda(nivelayuda:integer);
begin
 openwindow(10,5,70,20,'Ayuda',115,115);clrscr;
 case nivelayuda of
  1:begin
      writeln; writeln('Ayuda Menu Principal');writeln;
      writeln('  Seleccione su opcion del menu principal introduciendo');
      writeln('el número de opción deseada y oprimiendo la tecla ');
```

```
        writeln('<ENTER>.');
        writeln(' ');
        writeln(' ');
        writeln('  Dentro de cada una de las opciones se puede encontrar');
        writeln('ayuda específica a ellas.                    ');
        writeln(' ');
        write  ('  Pulse <ENTER> para continuar..');
        write  ('');
    end;
  2:begin
        writeln; writeln('Ayuda en Parámetros de Memoria'); writeln;
        writeln('  En espacio disponible y puertos E/S se presenta una ');
        writeln('lista de opciones, de las cuales deben seleccionarse ');
        writeln('aquellas que se no encuentren usadas por otros disposi-');
        writeln('tivos (mouse, discos, etc.) instalados en su PC. Con-');
        writeln('sulte el manual técnico de su PC ante cualquier duda.');
        writeln('  En dirección máxima se presenta una lista de la');
        writeln('cual puede seleccionar la mayor requerida por el banco.');
        writeln('  En bits de salida, se debe seleccionar el numero ');
        writeln('de acuerdo al tamaño de palabra de la microinstrucción. ');
    end;
  3:begin
        writeln;writeln('Ayuda en Rangos de Memoria');writeln;
        writeln('  Se debe introducir un numero determinado de rangos,');
        writeln('desde 1 hasta 16 como máximo, en decimal.');
        writeln('  Es posible ver como se encuentran delimitados cada uno');
        writeln('de los rangos con opción 2. ');
        writeln('  Si se desea modificar los rangos, entonces, al pulsar');
        writeln('la opción 3, se presentará una lista numerada con los ');
        writeln('rangos existentes, y se debe introducir el número de ');
        writeln('a modificar; finalmente se pide introducir los limites ');
        writeln('inferior y superior del rango, en hexadecimal.');
        writeln('No se permiten valores mayores a la "dirección máxima".');
    end;
  5:begin
        writeln;writeln('Ayuda en Impresión');writeln;
        writeln('  Seleccione la opción de Continuar (1) en el momento ');
```

```
        writeln('en que la impresora se encuentre correctamente conectada ');
        writeln('y en linea. En caso de que ocurra un error, el programa ');
        writeln('puede terminar inesperadamente, por lo que se recomienda ');
        writeln('salvar los datos antes de proceder a imprimir.');
        writeln('  La impresion se ejecutará con los parámetros de  ');
        writeln('memoria actuales  y con los rangos definidos. ');
        writeln(' ');
    writeln(' ');
        writeln(' ');
  end;
6:begin
        writeln;writeln('Ayuda en transferencia');writeln;
        writeln('  La transferencia se llevara a cabo con los parámetros');
        writeln('de memoria actuales y con los rangos definidos');
    writeln(' ');
        writeln('  El programa verifica si se encuentran las condiciones ');
        writeln('básicas para realizar la transferencia, pero el usuario');
        writeln('siempre debe checar de antemano las mismas. Por ejemplo,');
        writeln('que la energía del sistema y los interruptores de');
        writeln('direcciones y puertos no interfieran con otros ');
        writeln('dispositivos de la PC. ');
        writeln(' ');
  end;
7:begin
        writeln;writeln('Ayuda en Archivo');writeln;
        writeln('  La opción de nombre permite cambiar el nombre al ');
        writeln('archivo de datos y puede contener la ruta completa ');
        writeln('del archivo, por ejemplo: c:\pascal\datos\archivo.dat.');
        writeln('  Es importante nombrar el archivo con caracteres ');
        writeln('aceptables por el DOS, para evitar conflictos. ');
        writeln('  Una vez que el nombre de archivo es correcto, se ');
        writeln('puede proceder a guardarlo, con la opcion 2.');
        writeln('');
        writeln('  También se puede recuperar un archivo previamente');
        writeln('almacenado con la opción 2, y trabajar con él.');
  end;
end;
```

```
  repeat  until KeyPressed;
  closewindow;
end;
{-------------------------------------------------------------------------}
{Procedimiento para cambio de parametros: }
{-------------------------------------------------------------------------}
procedure menuparametros;
var
   opcion,opcion1    :          char;
   x,y,x1,y1                    :          byte;
   codigo            :          integer;
{-------------------------------------}
   procedure rellenarmatriz;
   var
     i,j,x,y  :          integer;
     opcion             :          char;
   begin
   openwindow(3,10,77,20,'Confirme inicializar matriz',115,115);clrscr;
   writeln('Precaución, este proceso modifica los datos actuales');
   writeln('Proceder a inicializar matriz con : ',caracterrelleno);
   write('Confirme  S/N  ? --> '); x:=wherex; y:=wherey;
   repeat
            gotoxy(x,y);
            opcion:=readkey;
            opcion:=upcase(opcion);
   until (opcion in ['S','N',#27]);
   if opcion  = 'S' then
            begin
                     for i:=0 to 1023 do
                              for j:=0 to 19 do
                                       dato[i,j]:=caracterrelleno;
                     writeln;
                     write('Inicialización concluida.. pulse una tecla');
                     repeat until keypressed;
            end;
   closewindow;
 end;
```

```
{-------------------------------------}
begin
 opcion:='0';
 repeat
   openwindow(25,6,60,21,'Parámetros de Memoria',115,115);clrscr;
   writeln('0. Ayuda');
   writeln;
   writeln('1. Espacio Disponible  $',despla);
   writeln('2. Puerto Lectura      ',puerto_lectura);
   writeln('3. Puerto Escritura    ',puerto_escritura);
   writeln('4. Dirección máxima    ',MAXIMADIRECCION);
   writeln('5. Bits de salida      ',MAXIMOSBITS);
   writeln('6. Caracter de inicio  ',caracterrelleno);
   writeln;
   writeln('9. Menú principal');
   writeln;write('Opción --> '); x:=wherex; y:=wherey;
   repeat
        gotoxy(x,y); opcion:=readkey;
   until opcion in ['0'..'6','9'];
   case opcion of
   '0': ponayuda(2);
   '1': begin
            openwindow(48,8,68,22,'Espacio Disponible',115,115);clrscr;;
            writeln;
            writeln('1. 8000');writeln('2. A000'); writeln('3. D000');
            writeln; writeln('9. No cambiar'); writeln;
            write('Opción --> '); x1:=wherex; y1:=wherey;
            repeat
                 gotoxy(x1,y1); opcion1:=readkey;
            until opcion1 in ['1'..'3','9'];
            case opcion1 of
                      '1': begin despla:='8000'; desp:=1; end;
                      '2': begin despla:='A000'; desp:=2; end;
                      '3': begin despla:='D000'; desp:=3; end;
                      '9':
                      end;
            closewindow;
```

```
        end;
'2': begin
        openwindow(48,8,68,22,'Puerto de Lectura',115,115);clrscr;;
        writeln;
        writeln('1. $221');writeln('2. $281'); writeln('3. $301');
        writeln; writeln('9. No cambiar'); writeln;
        write('Opción --> '); x1:=wherex; y1:=wherey;
        repeat
            gotoxy(x1,y1); opcion1:=readkey;
        until opcion1 in ['1'..'3','9'];
        case opcion1 of
            '1': begin
                    num_puerto_lectura:=$221;
                    puerto_lectura:='$221';
                end;
            '2': begin
                    num_puerto_lectura:=$281;
                    puerto_lectura:='$281';
                end;
            '3': begin
                    num_puerto_lectura:=$301;
                    puerto_lectura:='$301';
                end;
            '9':
            end;
        closewindow;
    end;
'3': begin
        openwindow(48,8,68,22,'Puerto de Escritura',115,115);clrscr;;
        writeln;
        writeln('1. $220');writeln('2. $280'); writeln('3. $300');
        writeln; writeln('9. No cambiar'); writeln;
        write('Opción --> '); x1:=wherex; y1:=wherey;
        repeat
            gotoxy(x1,y1); opcion1:=readkey;
        until opcion1 in ['1'..'3','9'];
        case opcion1 of
```

```
                '1': begin
                        num_puerto_escritura: = $220;
                        puerto_escritura: = '$220';
                    end;
                '2': begin
                        num_puerto_escritura: = $280;
                        puerto_escritura: = '$280';
                    end;
                '3': begin
                        num_puerto_escritura: = $300;
                        puerto_escritura: = '$300';
                    end;
                '9':
                end;
        closewindow;
    end;
'4': begin  {DIRECCION maxima}
        openwindow(48,8,68,22,'Dirección máxima',115,115);clrscr;
        writeln;
        writeln('1. 00F');writeln('2. 0FF'); writeln('3. 1FF');
        writeln('4. 2FF');writeln('5. 3FF');
        writeln; writeln('9. No cambiar'); writeln;
        write('Opción --> '); x1: = wherex; y1: = wherey;
        repeat
                gotoxy(x1,y1); opcion1: = readkey;
        until opcion1 in ['1'..'5','9'];
        case opcion1 of
                '1': maximadireccion: = '00F';
                '2': maximadireccion: = '0FF';
                '3': maximadireccion: = '1FF';
                '4': maximadireccion: = '2FF';
                '5': maximadireccion: = '3FF';
                '9':
                end;
        num_maximadireccion: = deci(maximadireccion);
    closewindow;
      end;
```

```
'5': begin  {cantidad de bits de salida}
            openwindow(48,8,68,22,'Bits de salida',115,115);clrscr;;
            writeln;
            writeln('1. 8');writeln('2. 16'); writeln('3. 24');
            writeln('4. 32');writeln('5. 40');
            writeln('6. 48');writeln('7. 64');writeln('8. 80');
            writeln; writeln('9. No cambiar'); Writeln;
            write('Opción --> '); x1:=wherex; y1:=wherey;
            repeat
                  gotoxy(x1,y1); opcion1:=readkey;
            until opcion1 in ['0'..'8','9'];
            case opcion1 of
                        '1': maximosbits:='8';
                        '2': maximosbits:='16';
                        '3': maximosbits:='24';
                        '4': maximosbits:='32';
                        '5': maximosbits:='40';
                        '6': maximosbits:='48';
                        '7': maximosbits:='64';
                        '8': maximosbits:='80';
                        '9':
                        end;
             val(maximosbits,num_maximosbits,codigo);
             if codigo <>0 then
              begin
                    writeln('error convirtiendo maximosbits');
                    repeat until keypressed;
              end;
            closewindow
          end;
'6': begin
            openwindow(48,8,68,22,'Caracter de inicio',115,115);clrscr;;
            writeln;
            writeln('  De un caracter ');
            writeln('hexadecimal');
            writeln('con el cual toda la matriz ');
            writeln('de datos se inicializará');
```

```
            writeln;
            write('Opción --> '); x1:=wherex; y1:=wherey;
            repeat
                gotoxy(x1,y1); opcion1:=readkey; opcion1:=upcase(opcion1);
            until opcion1 in ['0'..'9','A'..'F'];
            caracterrelleno:=opcion1;
            rellenarmatriz;
            closewindow;
          end;
    '9':
    end; {case opcion 1..4, 9}
    closewindow;
  until opcion ='9';
end;
{-------------------------------------------------------------------}
{Procedimiento para cambio de rangos: }
{-------------------------------------------------------------------}
procedure menurangos;
var
  opcion,opcion1    :        char;
  x,y,x1,y1         :        byte;
  i,j,k,numero,
  codigo            :        integer;
  cadena2           :        string[2];
  cadena3           :        string[3];
{-------------------------------------}
 procedure cambiardefiniciones;
 var
   i,k,direccion1,direccion2    :        integer;
 begin
   repeat
    openwindow(48,8,79,22,'Cambiar definiciones de rangos',115,115);clrscr;
    writeln('Inicio Final   Inicio-Final');
    i:=1;
    repeat
          write(i,'. ' ,dirminimas[i],'-',dirmaximas[i],'       ');
          if i<num_maximosrangos
```

```
            then writeln(i+1,'.' ,dirminimas[i+1],'-',dirmaximas[i+1]);
        i:=(i+2);
until (i > num_maximosrangos);
writeln('99. Terminar');
write('Rango a cambiar --> ');x1:=wherex;y1:=wherey;
repeat
        gotoxy(x1,y1);
        readln(cadena2);
        val(cadena2,numero,codigo);
until (codigo=0);
case numero of
1..16: begin
            openwindow(10,17,70,22,'Cambio de espacios de rangos',115,115);
            clrscr;
            writeln('Rango ',numero);
            write('Dirección inicial --> ');x1:=wherex;y1:=wherey;
            write(dirminimas[numero]);
            repeat
                    gotoxy(x1,y1);
                    readln(cadena3);
                    if cadena3='' then cadena3:=dirminimas[numero];
                    for k:=1 to 4 do cadena3[k]:=upcase(cadena3[k]);
                    codigo:=0; k:=1;
                    while ((codigo=0) and (k<4)) do
                      begin
                            if cadena3[k] in ['0'..'9','A'..'F'] then codigo:=0
                              else codigo:=1;
                            k:=k+1;
                      end;
                    dirminimas[numero]:=cadena3;
                    direccion1:=deci(dirminimas[numero]);
                    if (direccion1 >=0 ) and
                          (direccion1 <=num_maximadireccion) then codigo:=0
                          else codigo:=1;
            until (codigo=0);
            dirminimas[numero]:=cadena3;
            write('Dirección final  --> ');x1:=wherex;y1:=wherey;
```

```
            write(dirmaximas[numero]);
            repeat
                    gotoxy(x1,y1);
                    readln(cadena3);
                    if cadena3 = '' then cadena3: = dirmaximas[numero];
                    for k: = 1 to 3 do cadena3[k]: = upcase(cadena3[k]);
                    codigo: = 0; k: = 1;
                    while ((codigo = 0) and (k < 4)) do
                      begin
                            if cadena3[k] in ['0'..'9','A'..'F'] then codigo: = 0
                            else codigo: = 1;
                            k: = k + 1;
                      end;
                    dirmaximas[numero]: = cadena3;
                    direccion2: = deci(dirmaximas[numero]);
                    if (direccion1 < = direccion2) and
                          (direccion2 < = num_maximadireccion) then codigo: = 0
                          else codigo: = 1;
            until (codigo = 0);
            dirmaximas[numero]: = cadena3;
            write('Rango modificado. Presione cualquier tecla..');
            repeat until keypressed;
            closewindow
          end;  {1..16}
   99:
   end;  {case numero}
   closewindow;
  until(numero = 99);
  for i: = 1 to 16 do
        begin
          arreglo1[i]: = deci(dirminimas[i]);
          arreglo2[i]: = deci(dirmaximas[i]);
        end;
 end;   {cambiar definiciones de rangos}
{-------------------------------------}
begin
  opcion: = '0';
```

```
repeat
        openwindow(25,6,60,21,'Rangos',115,115);clrscr;
        writeln('0. Ayuda');
        writeln;
        writeln('1. Cantidad de rangos   ',MAXIMOSRANGOS);
        writeln('2. Ver definiciones     ');
        writeln('3. Cambiar definiciones ');
        writeln;
        writeln('9. Menú principal');
        writeln; write('Opción --> '); x:=wherex; y:=wherey;
        repeat
          gotoxy(x,y); opcion:=readkey;
        until opcion in ['0'..'3','9'];
        case opcion of
        '0': ponayuda(3);
        '1': begin     {cantidad de rangos}
                  openwindow(48,8,79,22,'Cantidad de rangos',115,115);clrscr;
                  writeln;
                  writeln('Escriba la cantidad de rangos,');
                  write('Mínimo 1, máximo 16 --> '); x1:=wherex; y1:=wherey;
                  repeat
                    gotoxy(x1,y1);
                    readln(maximosrangos);
                    val(maximosrangos,numero,codigo);
                  until ((codigo=0) and ((numero < 17) and (numero > 0)));
                  val(maximosrangos,num_maximosrangos,codigo);
                  if codigo <>0 then
                   begin
                    writeln('error convirtiendo maximosrangos');
                    repeat until keypressed;
                   end;
          closewindow;
            end;
        '2': begin   {ver definiciones de rangos}
             openwindow(48,8,79,22,'Definiciones de rangos',115,115);clrscr;
             write('Inicio-Final     Inicio-Final');writeln;
             i:=1;
```

```
                repeat
                        write(i,'. ' ,dirminimas[i],'-',dirmaximas[i],'        ');
                        if i<num_maximosrangos
                          then
                          writeln(i+1,'. ' ,dirminimas[i+1],'-',dirmaximas[i+1]);
                        i:=(i+2);
                until (i > num_maximosrangos);
                writeln;write('Pulse cualquier tecla...');
                repeat until keypressed;
                closewindow;
              end;
          '3': cambiardefiniciones;
          '9':
          end;
          closewindow;   {cierra ventana menu rangos}
    until opcion ='9';
end; {menurangos}
{-------------------------------------------------------------------------}
{procedimiento para la captura o edición de microinstrucciones: }
{-------------------------------------------------------------------------}
procedure menuedicion;
var
   carent,opcion,opcion1        :        char;
   h,i,j,k,x,y,x1,y1,direccion   :        integer;
   datocorrecto,terminar,
   continuar,continuar2                  :        boolean;
begin
   opcion:='0'; k:=0;
   openwindow(3,10,77,20,'Entrada de datos',115,115);clrscr;
   h:=1;
   continuar:=true; continuar2:=true;
   while ((h < num_maximosrangos+1) and continuar ) do
   begin
     direccion:=arreglo1[h];
     continuar2:=true;
     while ((direccion <=arreglo2[h]) and continuar2 and continuar) do
           begin
```

```
repeat
datocorrecto:=false; terminar:=false; k:=0;
gotoxy(1,1);
writeln;              writeln('Rango      ',h);
writeln(' ');         writeln('Dirección    MSB');writeln;
write(hex(direccion),'          '); x:=wherex; y:=wherey;
gotoxy(1,y+2);
writeln('<ESC>:Salir        Teclas válidas: 0123456789ABCDEF');
gotoxy(x,y);
for j:=0 to num_maximosbits div 4 -1 do write(dato[direccion,j]);
gotoxy(x,y);
while (k < (num_maximosbits div 4)) do
begin
  carent:=readkey; carent:=upcase(carent);
  case carent of
   '0'..'9','A'..'F': begin
                                write(carent);
                                dato[direccion,k]:=carent;
                                k:=k+1;
                              end;
   #27,#08,#28:       k:=num_maximosbits div 4;{para que salga}
  end;
end; {while, termina de contar los bits}
openwindow(50,12,78,22,'Verifique datos',115,115);clrscr;
writeln;
writeln('<ENTER>,S.Correcto');
writeln('<-,N    . Volver a entrar');
writeln('<ESC>   . Siguiente dato');
writeln('R       . Siguiente rango');
writeln('T       . Terminar'); writeln;
write('Dato correcto? --> '); x1:=wherex; y1:=wherey;
repeat
      gotoxy(x1,y1); opcion1:=readkey;
      opcion1:=upcase(opcion1);
until(opcion1 in [#13,#0,#8,#72,#77,#27,'S','N','T','R']);
case opcion1 of
  #13,'S'                          datocorrecto:=true;
```

```
            #0,#8,#72,#77,'N'      :        k:=0;
            #27                    :        terminar:=true;
            'T'                    :        continuar:=false;
            'R'                    :        continuar2:=false;
            ELSE
              write(char(#07));
            end;
          closewindow; {Ya tomando la verificacion}
          until(datocorrecto or terminar or not continuar or not continuar2);
          direccion:=direccion+1
    end;  { while (direccion <=arreglo2[h])  }
    h:=h+1;
  end;  { while (h <num_maximosrangos+1) }
closewindow;
end;
{-------------------------------------------------------------------------}
{Procedimiento para la impresión de las microinstrucciones: }
{-------------------------------------------------------------------------}
procedure menuimpresion;
var
opcion,opcion1      :      char;
x,y                 :      byte;
h                   :      integer;
{-------------------------------------}
 function impresoralista:boolean;
 VAR
   parms            :       registers;
 begin
  parms.ax:=$200; {funcion 2}
  parms.dx:=0;   {numero de la impresora}
  intr($17,parms);
  impresoralista:=(hi(parms.ax)=$90);{prueba de los bits 7 y 4}
 end;
{-------------------------------------}
 PROCEDURE IMPRESION_DE_DATOS(VAR  DIR_INICIO,DIR_FINAL:integer);
 VAR
   LINEA,COL,
```

```
  LI,CO,
  LIN,COLU      :          INTEGER;
BEGIN
  CLRSCR;
  LI:=0;
  WRITELN(LST);
  WRITELN(LST);
  WRITELN(LST,'        ','DIRECCION','        ','D A T O');
  WRITELN(LST);
  FOR LINEA:=0 TO (DIR_FINAL - DIR_INICIO)  DO
      BEGIN
        WRITE(LST,'          ',HEX(DIR_INICIO+LINEA),'          ');
        FOR COL:=0 TO ((num_maximosbits div 4)-1) DO
              WRITE(LST,dato[DIR_INICIO+LINEA,COL]);
        WRITELN(LST);
      END;
END;
{-------------------------------------}
begin
  repeat
      openwindow(25,6,60,21,'Impresión',115,115);clrscr;
      writeln('0. Ayuda');
      writeln;
      writeln('1. Continuar ');
      writeln;
      writeln('9. Menú principal');
      writeln; write('Opción --> '); x:=wherex; y:=wherey;
      repeat
        gotoxy(x,y); opcion:=readkey;
      until opcion in ['0'..'1','9'];
      case opcion of
      '0': ponayuda(5);
      '1': if impresoralista then
          begin
              openwindow(48,8,79,22,'Imprimiendo',115,115);clrscr;
              writeln;write('Imprimiendo...'); x:=wherex;y:=wherey;
              writeln(lst,'');
```

```
                writeln(lst,
                  '    Universidad Nacional Autónoma de México',#013,#10,
                  'Escuela Nacional de Estudios Profesionales Aragón',
                  #013,#10,
                  '                TESIS DE INGENIERIA',#013,#10);
                FOR h := 1 TO  num_maximosrangos DO
                  IMPRESION_DE_DATOS(ARREGLO1[h],ARREGLO2[h]);
                writeln(lst,'',#12);
                openwindow(55,15,77,19,'Fin de Impresion',115,115);clrscr;
                writeln('Fin de impresion.');
                write('Presione una tecla...');
                repeat until keypressed;
                closewindow;
                closewindow;
        end
            else
            begin
                openwindow(10,12,70,17,'Impresora no preparada',115,115);
                clrscr;
                writeln;
                write('Error al intentar imprimir',#10,#13,
                                        'Presione una tecla');
                repeat until keypressed;
                closewindow;
            end;
        '9':
        end;
        closewindow;
    until opcion = '9';
end;
{-------------------------------------------------------------------------}
{Procedimiento para la transferencia de datos hacia el banco de memoria : }
{-------------------------------------------------------------------------}
procedure menutransferencia;
var
  opcion,opcion1     :          char;
  x,y                :          byte;
```

```
 m                              :          integer;
 correcto          :     boolean;
{-------------------------------------}
 procedure hardware(var correcto:boolean);
 var
  valorleido        : byte;
 begin
  correcto:= false;
  valorleido:= port[$221];
  {no todos los bits deben ser unos:}
  if ( (port[num_puerto_lectura] and $ff ) <>$FF) then
  {podemos checar todos los bits juntos: }
  {AA es la condicion que deseamos, y cuando damos XOR y el resultado
   es 00 entonces se ha logrado }
    if ( (port[num_puerto_lectura] xor $((insert('$',delete(despla,2,2),1)) or $05)) =  $00 ) then
correcto:= true;
 end;
{-------------------------------------}
 Procedure Da_control_a_PC;
 begin
  port[num_puerto_escritura]:= $F5;
 end;
{-------------------------------------}
 procedure Regresa_control_a_bitslices;
 begin
  port[num_puerto_escritura]:= $0B;
 end;
{-------------------------------------}
 procedure transfiere(VAR DIR_INICIAL,DIR_FINAL:integer);
 VAR
   CHI,DATO_FINAL,PRUEBA,
   CHI_FIN,chip_final      :     INTEGER;
   DATO_CADENA             :     STRING[3];
{-------------------------}
   procedure seg1;
   const
   despl = $8000;
```

```
var
 dir,chip                : integer;
 CHIP_MEMO_0      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$0000;
 CHIP_MEMO_1      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$1000;
 CHIP_MEMO_2      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$2000;
 CHIP_MEMO_3      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$3000;
 CHIP_MEMO_4      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$4000;
 CHIP_MEMO_5      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$5000;
 CHIP_MEMO_6      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$6000;
 CHIP_MEMO_7      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$7000;
 CHIP_MEMO_8      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$8000;
 CHIP_MEMO_9      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$9000;
begin
chip_final: = (Num_maximosbits div 4) - 1;
CHI_FIN : = ((CHIP_FINAL + 1) DIV 2)-1 ;
FOR DIR: = DIR_INICIAL TO DIR_FINAL DO
  BEGIN
       FOR CHIP: = 0 TO CHI_FIN DO   {CHI_FIN ES COMO MAXIMO = 9}
         BEGIN
              CHI: = CHIP * 2;
              {LOS DATOS SE  CONCATENAN DE DOS EN DOS PARA FORMAR
              UN BYTE DE UN CHIP EN PARTICULAR, EN UNA DIRECCION DADA}
              DATO_CADENA : = CONCAT ('$',dato [DIR,CHIP_FINAL-CHI-1],
                                          dato [DIR,CHIP_FINAL- CHI]);
              VAL (DATO_CADENA,DATO_FINAL,PRUEBA);
              {EL DATO SE ENVIA AL CHIP ADECUADO}
              CASE CHIP OF
              0  : CHIP_MEMO_0[DIR] : = DATO_FINAL;
              1  : CHIP_MEMO_1[DIR] : = DATO_FINAL;
              2  : CHIP_MEMO_2[DIR] : = DATO_FINAL;
              3  : CHIP_MEMO_3[DIR] : = DATO_FINAL;
              4  : CHIP_MEMO_4[DIR] : = DATO_FINAL;
              5  : CHIP_MEMO_5[DIR] : = DATO_FINAL;
              6  : CHIP_MEMO_6[DIR] : = DATO_FINAL;
              7  : CHIP_MEMO_7[DIR] : = DATO_FINAL;
              8  : CHIP_MEMO_8[DIR] : = DATO_FINAL;
              9  : CHIP_MEMO_9[DIR] : = DATO_FINAL;
```

```
                END; {CASE}
            END; {FOR CHIP:= 0 TO CHI_FIN}
    END;    { FOR DIR:= DIR_INICIAL TO DIR_FINAL}
 end;
{---------------------------}
 procedure seg2;
 const
  despl = $A000;
 var
  dir,chip                  : integer;
  CHIP_MEMO_0       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$0000;
  CHIP_MEMO_1       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$1000;
  CHIP_MEMO_2       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$2000;
  CHIP_MEMO_3       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$3000;
  CHIP_MEMO_4       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$4000;
  CHIP_MEMO_5       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$5000;
  CHIP_MEMO_6       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$6000;
  CHIP_MEMO_7       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$7000;
  CHIP_MEMO_8       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$8000;
  CHIP_MEMO_9       : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$9000;
begin
 chip_final:= (Num_maximosbits div 4) - 1;
 CHI_FIN :=((CHIP_FINAL + 1) DIV 2)-1 ;
 FOR DIR:= DIR_INICIAL TO DIR_FINAL DO
   BEGIN
       FOR CHIP:= 0 TO CHI_FIN DO   {CHI_FIN ES COMO MAXIMO = 9}
         BEGIN
             CHI:= CHIP * 2;
             {LOS DATOS SE  CONCATENAN DE DOS EN DOS PARA FORMAR
             UN BYTE DE UN CHIP EN PARTICULAR, EN UNA DIRECCION DADA}
             DATO_CADENA := CONCAT ('$',dato [DIR,CHIP_FINAL-CHI-1],
                                        dato [DIR,CHIP_FINAL- CHI]);
             VAL (DATO_CADENA,DATO_FINAL,PRUEBA);
             {EL DATO SE ENVIA AL CHIP ADECUADO}
             CASE CHIP OF
             0   : CHIP_MEMO_0[DIR] := DATO_FINAL;
             1   : CHIP_MEMO_1[DIR] := DATO_FINAL;
```

```
                    2  : CHIP_MEMO_2[DIR] := DATO_FINAL;
                    3  : CHIP_MEMO_3[DIR] := DATO_FINAL;
                    4  : CHIP_MEMO_4[DIR] := DATO_FINAL;
                    5  : CHIP_MEMO_5[DIR] := DATO_FINAL;
                    6  : CHIP_MEMO_6[DIR] := DATO_FINAL;
                    7  : CHIP_MEMO_7[DIR] := DATO_FINAL;
                    8  : CHIP_MEMO_8[DIR] := DATO_FINAL;
                    9  : CHIP_MEMO_9[DIR] := DATO_FINAL;
                    END; {CASE}
            END;
    END;
  end;
{-------------------------}
 procedure seg3;
 const
   despl = $D000;
 var
   dir,chip                 : integer;
   CHIP_MEMO_0      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$0000;
   CHIP_MEMO_1      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$1000;
   CHIP_MEMO_2      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$2000;
   CHIP_MEMO_3      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$3000;
   CHIP_MEMO_4      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$4000;
   CHIP_MEMO_5      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$5000;
   CHIP_MEMO_6      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$6000;
   CHIP_MEMO_7      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$7000;
   CHIP_MEMO_8      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$8000;
   CHIP_MEMO_9      : ARRAY [0..1023] OF BYTE ABSOLUTE DESPL:$9000;
 begin
   chip_final: = (Num_maximosbits div 4) - 1;
   CHI_FIN : = ((CHIP_FINAL + 1) DIV 2)-1  ;
   FOR DIR: = DIR_INICIAL TO DIR_FINAL DO
     BEGIN
          FOR CHIP: = 0 TO CHI_FIN DO   {CHI_FIN ES COMO MAXIMO = 9}
            BEGIN
                 CHI: = CHIP * 2;
                 {LOS DATOS SE  CONCATENAN DE DOS EN DOS PARA FORMAR
```

```
                    UN BYTE DE UN CHIP EN PARTICULAR, EN UNA DIRECCION DADA}
                    DATO_CADENA := CONCAT ('$',dato [DIR,CHIP_FINAL-CHI-1],
                                              dato [DIR,CHIP_FINAL- CHI]);
                    VAL (DATO_CADENA,DATO_FINAL,PRUEBA);
                    {EL DATO SE ENVIA AL CHIP ADECUADO}
                    CASE CHIP OF
                    0  : CHIP_MEMO_0[DIR] := DATO_FINAL;
                    1  : CHIP_MEMO_1[DIR] := DATO_FINAL;
                    2  : CHIP_MEMO_2[DIR] := DATO_FINAL;
                    3  : CHIP_MEMO_3[DIR] := DATO_FINAL;
                    4  : CHIP_MEMO_4[DIR] := DATO_FINAL;
                    5  : CHIP_MEMO_5[DIR] := DATO_FINAL;
                    6  : CHIP_MEMO_6[DIR] := DATO_FINAL;
                    7  : CHIP_MEMO_7[DIR] := DATO_FINAL;
                    8  : CHIP_MEMO_8[DIR] := DATO_FINAL;
                    9  : CHIP_MEMO_9[DIR] := DATO_FINAL;
                    END; {CASE}
              END;
    END;
  end;
{-------------------------}
 begin
          case desp of
                    1 : seg1;
                    2 : seg2;
                    3 : seg3;
          end;
 end; {procedure transfiere}
{-------------------------------------}
begin
   repeat
        openwindow(25,6,60,21,'Transferencia',115,115);clrscr;
        writeln('0. Ayuda');
        writeln;
        writeln('1. Continuar ');
        writeln;
        writeln('9. Menú principal');
```

```
writeln; write('Opción --> '); x:=wherex; y:=wherey;
repeat
        gotoxy(x,y);  opcion:=readkey;
until opcion in ['0'..'1','9'];
case opcion of
'0': ponayuda(6);
'1': begin
     correcto:=false;
     if despla <>'8000' then hardware(correcto);
     if ( (correcto) or (despla='8000') ) then
          begin
            openwindow(48,8,79,22,'Transferencia',115,115);clrscr;
            writeln;write('Transferencia...'); x:=wherex;y:=wherey;
     Da_control_a_PC;
            FOR m := 1 TO  num_maximosrangos DO
                    transfiere(ARREGLO1[m],ARREGLO2[m]);
            Regresa_control_a_bitslices;
            openwindow(55,15,77,19,'Fin de Transferencia',115,115);
            clrscr;
            writeln('Fin de transferencia.');
            write('Presione una tecla...');
            repeat until keypressed;
            closewindow; {fin de transferencia}
            if (despla ='8000') then
              begin
                openwindow(12,17,68,24,'Transferencia Interna',115,115);
                clrscr;
                writeln('La transferencia se efectúo en la memoria');
                writeln('interna de la PC; puede analizarla invocando');
                writeln('el comando DEBUG del DOS, desde línea de co-');
                writeln('mando y al indicador: - dar: d 8000:0000');
                delay(9000);
                repeat until keypressed;
                closewindow;  { transferencia interna }
              end;
            closewindow;     {transferencia}
          end  {if hardware correcto}
```

```
                    else
                    begin
                      openwindow(5,18,75,24,'Error de Hardware',115,115);clrscr;
                      writeln('Verifique el Sistema de Controladores:');
                      writeln('* Números de Puertos E/S');
                      writeln('* Energía eléctrica');
                      write('* Bloque de Memoria Seleccionada');
                      repeat until keypressed;
                      closewindow;
                    end;
                end;
    '9':
        end;
  closewindow;
  until opcion = '9';
end;
{-------------------------------------------------------------------------}
{Procedimiento para recuperar/almacenar datos de/en disco: }
{-------------------------------------------------------------------------}
procedure menuarchivo;
var
  opcion,opcion1:char;
  x,y:byte; i,j,k:integer;
  cadena:string;
{-------------------------------------}
procedure archivado;
var
  archivodatos :    file of matriz_datos;
  archivonuevo :          boolean;
  i,j,x,y   :       integer;
  opcion  :         char;
{-------------------------}
  procedure escribearchivo;
  begin
        rewrite(archivodatos);
        write(archivodatos,dato);
        close(archivodatos);
```

```
        writeln('Archivo escrito');
        write('Pulse una tecla..');
        repeat until keypressed;
 end;
{---------------------------}
 begin
   assign(archivodatos,nombrearchivo);
   {$i-} reset(archivodatos); {$i+}
   archivonuevo:= (ioresult <>0);  {0 si ya existe => archivonuevo=true}
   if archivonuevo then escribearchivo
   else
     begin
          openwindow(10,12,70,17,'Verifique sobreescritura',115,115);clrscr;
          writeln('El archivo ',nombrearchivo,' ya existe ..');writeln;
          write('Sobreescribir  S/N ? --> ');x:=wherex; y:=wherey;
          repeat
                gotoxy(x,y); opcion:=readkey;
          until (opcion in ['S','s','N','n',#27]);
          writeln;
          if upcase(opcion) = 'S' then escribearchivo;
          closewindow;
     end;
 end;
{---------------------------------------}
 procedure recuperado;
 var
   archivodatos     :        file of matriz_datos;
   archivonuevo     :        boolean;
  i,j,x,y           :        integer;
   opcion           :        char;
{---------------------------}
   procedure recuperaarchivo;
   begin
    read(archivodatos,dato);
    close(archivodatos);
    writeln;
    writeln('Archivo recuperado');
```

```
    write('Pulse una tecla.. --> ');
    repeat until keypressed;
  end;
{-----------------------------}
 begin
   assign(archivodatos,nombrearchivo);
   {$i-} reset(archivodatos); {$i+}
   archivonuevo:= (ioresult <>0);  {0 si ya existe  =>archivonuevo=false}
   if not archivonuevo  then  recuperaarchivo
   else
        begin
          openwindow(10,12,70,17,'Error de lectura',115,115);clrscr;
          writeln('El archivo ',nombrearchivo,' no se encontró');writeln;
          write('Pulse una tecla para continuar..');
          repeat until keypressed;
          closewindow;
        end;
 end;  {archivado}
{-----------------------------------}
begin
   repeat
        openwindow(25,6,60,21,'Archivo',115,115);clrscr;
        writeln('0. Ayuda');
        writeln;
        writeln('1. Nombre    ',NOMBREARCHIVO);
        writeln('2. Guardar   ');
        writeln('3. Recuperar ');
        writeln;
        writeln('9. Menú principal');
        writeln; write('Opción --> '); x:=wherex; y:=wherey;
        repeat
        gotoxy(x,y); opcion:=readkey;
        until opcion in ['0'..'3','9'];
        case opcion of
        '0': ponayuda(7);
        '1': begin
             openwindow(3,15,77,19,'Nombre Archivo',115,115);clrscr;
```

```
            writeln('Nombre de archivo con ruta');write('--> ');
            x:=wherex;y:=wherey;
            write(NOMBREARCHIVO);
            gotoxy(x,y);
            readln(CADENA);
            if (NOMBREARCHIVO <>CADENA) AND (CADENA <>'') then
                  nombrearchivo:=cadena;
            closewindow;
           end;
        '2': archivado;

        '3': Recuperado;
        '9':
        end;
   closewindow;
   until opcion='9';
end;          {menu archivo}
{-------------------------------------------------------------------}
{Programa principal: }
{-------------------------------------------------------------------}
begin
 iniciavariablesglobales;
 while (terminar = false) do
  begin
        openwindow(2,4,35,20,'Menu principal',115,115); clrscr;
        writeln('0. Ayuda');
        writeln('1. Parámetros de Memoria');
        writeln('2. Rangos de trabajo');
        writeln('3. Captura de datos');
        writeln('4. Impresión');
        writeln('5. Tranferencia');
        writeln('6. Archivo');
        writeln; writeln('9. Salir');writeln;
        write('Opción --> '); x:=wherex; y:=wherey;
              REPEAT
                gotoxy(x,y);
                opcion:=readkey;
```

```
                UNTIL (opcion in ['0'..'6','9']);
        case opcion of
        '0': ponayuda(1);
        '1': menuparametros;
        '2': menurangos;
        '3': menuedicion;
        '4': menuimpresion;
        '5': menutransferencia;
        '6': menuarchivo;
        '9': begin
                openwindow(3,15,77,19,'Confirme salida',115,115);clrscr;
                writeln; write('  Salir s/n --> ');
                opcion := readkey;
                if (opcion in ['s','S']) then terminar := true;
                closewindow;
            end;
        end; {case opcion}
        closewindow;
  end; {while }
  Textbackground(0);
  Textcolor(7);
  clrscr;
end.
{-------------------------------------------------------------------}
```

5

Pruebas y Caracterización del Prototipo

Introducción

Este capítulo constituye una guía para el usuario del sistema. Se presentan los pasos a seguir para lograr un correcto uso del banco de memoria. Esto incluye tanto el manejo del programa MICROPRO desarrollado en el capítulo anterior, como la configuración del banco de memoria.

Este capítulo no pretende enseñar a utilizar la computadora. No se hallará aquí una indicación de dónde se encuentra el interruptor de la computadora, monitor o impresora. Tampoco es objetivo intentar revelar los secretos del sistema operativo. Y aunque no se necesite ser un experto conocedor del DOS, sí es conveniente estar familiarizado con algunas ordenes simples como DIR, COPY y CHKDSK.

En cambio, sí se indican las precauciones a tomar para evitar fallas que pudieran ocurrir, ya sea en la PC o en el banco de memoria.

5.1 Equipo necesario

A diferencia de los productos electrónicos de consumo que son comprados y usados sin modificaciones, el hardware y el software de las computadoras puede y/o debe ajustarse uno en función del otro.

Por ejemplo, si se deseará ejecutar una aplicación que requiera 2 MB de memoria y 20 MB en disco, en una PC que únicamente posee 640 KB y 10 MB en disco, entonces, se deben adicionar los elementos de hardware que permitan alcanzar la configuración necesaria. Por otro lado, si se desea convertir a una PC en un "Bridge" para una red local (LAN), entonces será necesario incrementar una tarjeta de comunicaciones. Pero, que sucede si todas las ranuras de expansión ya se encuentran ocupadas?. Indudablemente se tendría la necesidad de quitar alguna tarjeta y en su lugar, colocar la nueva. Un caso peor es, cuando, aunque existen ranuras de expansión disponibles, todas las interrupciones y canales de DMA ya han sido ocupadas. En este caso, nuevamente sería necesario quitar alguna tarjeta, o bien usar otro equipo.

Las computadoras personales son similares a los automóviles. Un carro puede estar equipado con inyección de combustible o ser turbo cargado; las computadoras personales pueden tener memorias de diferentes tamaños y diferentes impresoras y monitores. Los automóviles y las computadoras personales difieren en que un carro es manufacturado por una firma y vendido con beneficios instalados, mientras que una computadora personal es manufacturada por una firma y vendida independientemente del software, el cual es manufacturado por otra firma.

Que se necesita para ejecutar Micropro

Micropro, el programa de aplicación desarrollado en el capítulo anterior, puede ejecutarse en cualquier computadora personal que sea 100% compatible con IBM PC, PC XT y PC AT. Requiere una unidad de disco flexible y monitor monocromático, CGA o RGB. La computadora debe tener entre 512KB y 640KB de memoria RAM. Además, se debe contar los elementos listados a continuación:

Sistema operativo

Micropro se ejecuta bajo ambiente DOS (Sistema Operativo de Disco). Se debe tener instalada una versión de DOS 3.0 o superior.

Impresora

Una impresora de matriz de puntos, chorro de tinta, térmica o láser servirá para obtener los listados del programa Micropro.

5.2 Instalación de la tarjeta

Debido a que se requiere instalar la tarjeta (experimental, ahora) de interfase al banco de memoria en una de las ranuras de expansión, a continuación se presentan las precauciones que se deben tomar para la adecuada instalación de la misma.

Precauciones

La precaución #1 es la relacionada a la electricidad estática (la chispa que salta del cuerpo después de caminar sobre una alfombra, usando zapatos con suela de cuero). Es el mayor eliminador de circuitos integrados, CI's o CHIPs. Las pequeñas chispas representan voltajes del orden de 1,000 a 10,000 volts de descarga eléctrica, que pueden fácilmente dañar un dispositivo electrónico construido para

operar entre 5 y 20 volts. Así que se debe ser muy cuidadoso al manejar cualquier tarjeta de la PC.

Un método adecuado a usar cuando se instalen o remuevan estos dispositivos electrónicos, es tocar algún metal conectado a tierra, antes de manejar tal artículo. Cualquier parte metálica, no pintada y limpia del gabinete de la PC, es un buen lugar para tocar; por ejemplo podemos tocar el chasis de la fuente de poder. Una vez que la tarjeta está en las manos, no se deben tocar los componentes electrónicos.

La precaución #2 es la relativa a la seguridad de las manos cuando se remueven o instalan tarjetas. La cara posterior de las tarjetas, donde no hay componentes, tiene muchos puntos filosos sobresaliendo. Tomar una tarjeta con violencia puede resultar en cortaduras en las manos. Además, muchas veces las tarjetas tienen conexiones "puente" que pueden ser dañadas. La mejor solución cuando se intenta instalar o quitar una tarjeta *difícil*, es, balancear la tarjeta de atrás hacia adelante durante el proceso.

Generalmente las ranuras de expansión localizadas en el interior de la PC son iguales, lo que significa que se puede instalar la tarjeta en cualquier ranura y debe trabajar correctamente. Sin embargo, algunas tarjetas requieren cables que se conectan a otros dispositivos en la computadora, tales como unidades de disco, o la tarjeta de interfase al banco de memoria; así que es preferible instalar la tarjeta en la ranura más cercana del dispositivo al que será conectado.

Configuración de la tarjeta de interfase

Muchas tarjetas contienen medios de comunicación con dispositivos externos tales como modems, impresoras, plotters o "mouse". Estas áreas de comunicación son llamados puertos. Cada puerto tiene una localidad de memoria específica, llamada dirección que es reservada para su uso exclusivo.

Si dos tarjetas en el mismo sistema de cómputo tratan de comunicarse usando la misma dirección de memoria, puede ocurrir un conflicto que causará fallas en los dispositivos. Esto es como la situación donde dos personas tratan de pasar por una sola puerta angosta al mismo tiempo.

Algunas tarjetas tienen sus direcciones de puertos en forma "cableada", por lo que no se pueden modificar fácilmente. Otras tarjetas, como la de interfase al banco de memoria, usan interruptores DIP (Dual In Package), o "jumpers" (pequeñas terminales interconectables) que permiten cambiar fácilmente la dirección de puertos. El punto principal aquí, es estar seguro que dos dispositivos no usen la misma dirección.

Por ello cuando se instale la tarjeta de interfase al banco de memoria, se tiene que verificar que las direcciones de puertos E/S y decodificación de memoria seleccionados, no tienen conflicto con los demás dispositivos ya instalados en la PC. El manual que viene con la PC y los dispositivos periféricos contienen la información necesaria para realizar la verificación. Consulte dichos manuales.

Debido a la normalización de los productos para PC, se pueden recomendar los siguientes valores para los interruptores de selección de puertos (Hex):

Puerto de Lectura	221
Puerto de Escritura	220
Dirección para decodificación de memoria	A000

Pero estos pueden variar de una computadora a otra.

Procedimiento para instalación

La tarjeta que se instala en una de las ranuras de la PC, puede quedar ahí por siempre, sin que ello afecte el funcionamiento de los programas instalados en la PC, para ello siga los siguientes pasos:

1. Apagar y desconectar la computadora, así como todos los dispositivos conectados a ella.
2. Observar las precauciones sobre la electricidad estática.
3. Abrir la unidad central.
4. Localizar una ranura vacía.
5. Quitar la pequeña lamina que cubre el orificio que da al exterior.
6 Estar seguro que los interruptores han sido correctamente configurados.
7. Insertar la tarjeta en la ranura de expansión. Balancear la tarjeta suavemente si fuera necesario.
8. Fijar la tarjeta al gabinete, instalando un tornillo.
9. Instalar el conector que da hacia el banco de memoria.
10. Poner la tapa de la unidad central
11. Realizar las conexiones que se eliminaron en el punto 1.
12. Conectar los extremos de los cables planos (2) a las terminales localizadas en la parte exterior de la tarjeta recién instalada. Note que cada cable plano se conecte con su correspondiente (1 con 1, 2 con 2), además la patilla 1, de cada cable se encuentra marcada.
13. Conectar los otros extremos de los cables planos al Banco de Memoria, observando las notas del punto anterior.

Procedimiento para quitar la tarjeta

Si por alguna razón desea remover la tarjeta, imite los pasos del procedimiento de instalación, en sentido inverso:

1. Apagar la computadora así como todos los dispositivos conectados a ellos.
2. Observar las precauciones sobre la electricidad estática.
3. Desconectar los cables planos.
4. Quitar el tornillo que sujeta la tarjeta.
5. Jalar suavemente pero firmemente la tarjeta, balanceándola si fuera necesario.

5.3 Ejecución del programa Micropro.

La ejecución del programa Micropro puede hacerse con el fin de capturar microinstrucciones, imprimir y guardar los datos en disco, para posteriormente transferirlos al banco de memoria instalado en otra computadora.

Para cargar Micropro se procede de la siguiente manera:

1. Introduzca el disco del Sistema Operativo DOS en la unidad A. Conecte y encienda la computadora.
2. Espere el indicador del sistema en pantalla (por ejemplo, *a:>*).
3. Teclear **a:micropro** y pulsar RETURN.

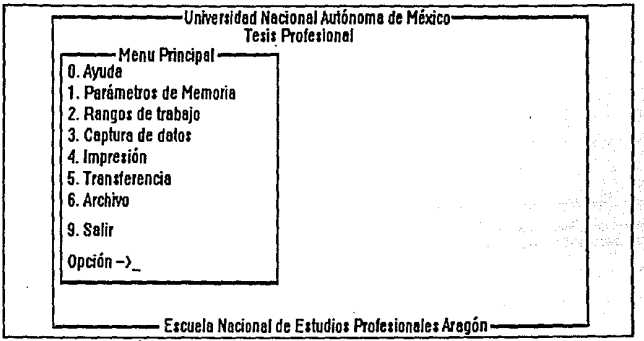


Figura 5-1 *Menú principal del programa micropro.*

Si se ha cargado correctamente Micropro, verá una pantalla como en la figura 5-1. Esta pantalla es el menú principal del sistema.

El menú principal es uno de los muchos menús con los que tendrá que trabajar en Micropro. Básicamente un menú es una lista de elecciones simples. en la que cada opción del menú está representada por un número.

Para seleccionar una opción del menú, lo único que debe hacer es teclear el número que representa dicha elección.

0. Obtención de ayuda

La utilidad de ayuda en Micropro consiste en un ventana de información en pantalla. Esta facilidad de ayuda se ha incluido en casi todas las opciones de los menús.

Para obtener ayuda se debe seleccionar la opción 0 en los menús que tengan tal opción.

A continuación se presentarán las funciones de las opciones del menú principal.

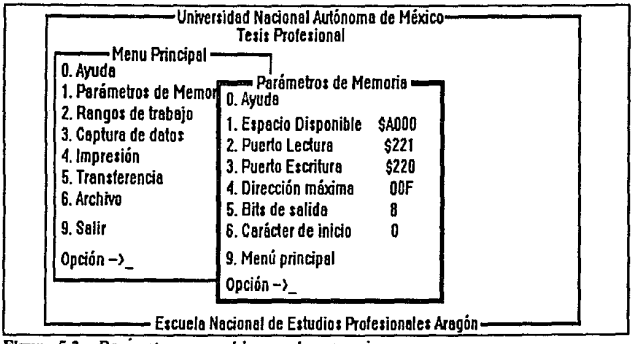


Figura 5-2 *Parámetros para el banco de memoria.*

1. Parámetros de Memoria

Seleccione la opción 1 del menú principal. Verá entonces una ventana con los actuales parámetros del banco de memoria, como la que se muestra en la figura 5-2.

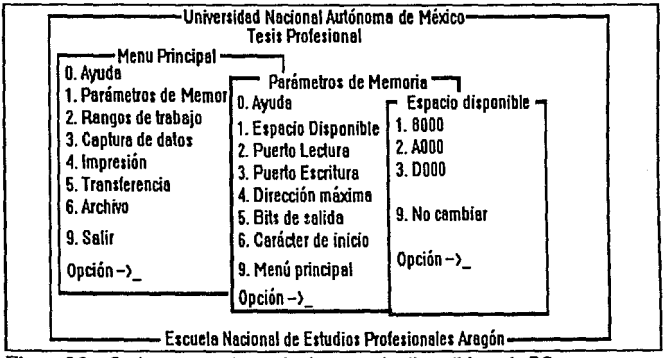


Figura 5-3 *Opciones para el espacio de memoria disponible en la PC.*

Espacio disponible

Al Seleccionar la opción de espacio disponible, aparece una ventana con opciones para seleccionar el espacio deseado, figura 5-3. Pulse el número con la opción de espacio deseado. Tal espacio corresponde a un conjunto de 40K direcciones de memoria en el cual el banco de memoria va a ser decodificado. El espacio selec-

cionado se toma como la base. Por ejemplo, al seleccionar $A0000, se hará uso de las direcciones $A0000 hasta $A93FF.

Debe recordarse sin embargo, que no todo el conjunto de 40 Kb de direcciones son completamente usadas. Más bien se usan solo 10 Kb, aunque repartidos dentro de los 40 Kb. También, la dirección base seleccionada deberá coincidir con la posición de los interruptores del banco de memoria.

Cuando se opta por la dirección base 8000 y la microcomputadora tiene instalados un total de 640 KB de memoria RAM, entonces la transferencia se efectúa en memoria interna.

Puerto de lectura, escritura

Para seleccionar el puerto de lectura o escritura, se debe considerar la posición de los interruptores del banco de memoria.

Los valores por omisión son los siguientes:

Lectura: 221
Escritura: 220

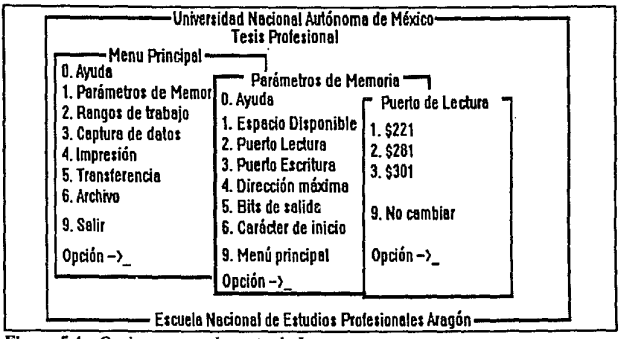


Figura 5-4 *Opciones para el puerto de Lectura.*

También, se debe consultar previamente el manual técnico de la computadora siendo usada, para verificar que los valores de puertos no son usados por otro dispositivo. Las figuras 5-4 y 5-5 muestran las opciones existentes para los puertos de lectura y escritura respectivamente.

Dirección máxima.

Las opciones para la dirección máxima se refieren al rango de memoria con el que se va a trabajar en el banco. Este valor va a limitar los valores posibles de los rangos de direcciones en la opción 2 del menú principal. El valor por omisión es de 00F. Todas las opciones se refieren a direcciones en hexadecimal. La figura 5-6 muestra todas las opciones para la dirección máxima.

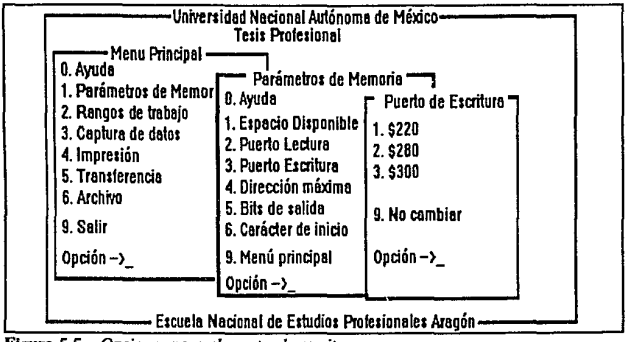


Figura 5-5 *Opciones para el puerto de escritura.*

Bits de salida

La opción de bits de salida determina que tan larga será la salida efectiva del banco de memoria. El valor definido aquí, va a tener influencia durante la captura, impresión y transferencia de datos.

Los valores posibles van desde 8 hasta 80. El valor por omisión es de 8 bits. La figura 5-7 muestra la ventana que aparece al definirla.

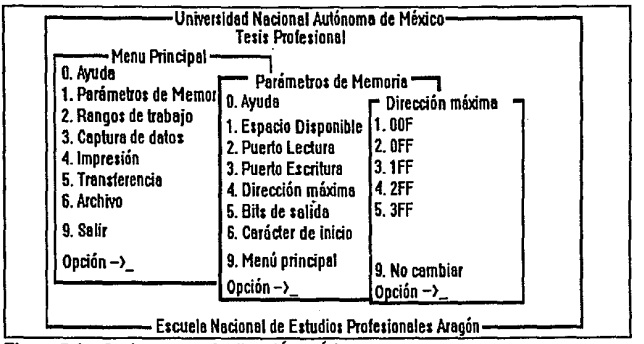


Figura 5-6 *Opciones para la dirección máxima.*

Carácter de inicio

Al entrar en esta opción, aparecerá una ventana, como la de la figura 5-8. Se debe teclear un carácter de valor hexadecimal [0..F]. Luego se pide una confirmación para rellenar la matriz de datos con tal dato. Si se procede, el nuevo valor sustituirá a los datos existentes que serán perdidos. Tal proceso es irreversible. Por ello se recomienda salvar los datos en disco antes de confirmar.

2. Rangos de trabajo

Seleccione la opción 2 del menú principal. En la pantalla verá aparecer una ventana con las tres opciones correspondientes a los rangos de trabajo. La figura 5-9 muestra las opciones de éste módulo.

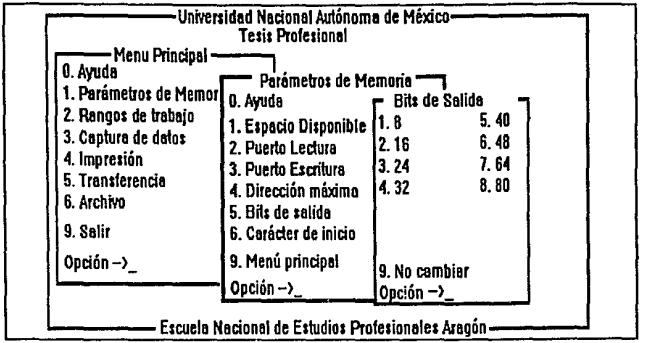


Figura 5-7 *Opciones para la cantidad de bits de salida.*

Cantidad de rangos

Al seleccionar ésta opción, se pedirá teclear la cantidad de rangos con los que se desea trabajar. Se debe introducir entonces un número entre 1 y 16 inclusive, y luego RETURN. La cantidad inicial de rangos es 1.

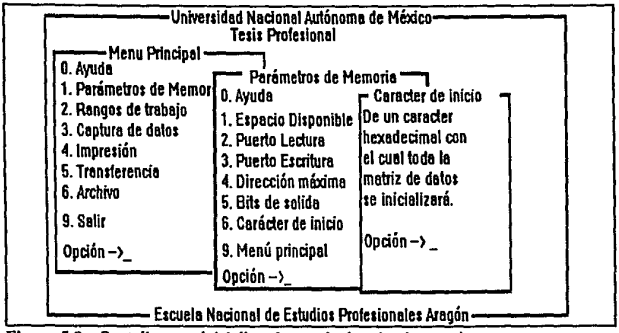


Figura 5-8 *Pantalla para inicializar la matriz de microinstrucciones.*

Ver definiciones

Esta opción permite ver en que direcciones opera cada uno de los rangos. No se permiten modificaciones.

Cambiar definiciones.

Esta opción permite modificar el espacio de direcciones definidos en cada rango. Se debe seleccionar el número de rango que se desea modificar. Entonces se abrirá otra ventana que pedirá las direcciones inicial y final para ese rango. Las direcciones deberán introducirse en hexadecimal y no deben exceder en valor a la dirección máxima definida en el módulo de parámetros de memoria. La figura 5-10

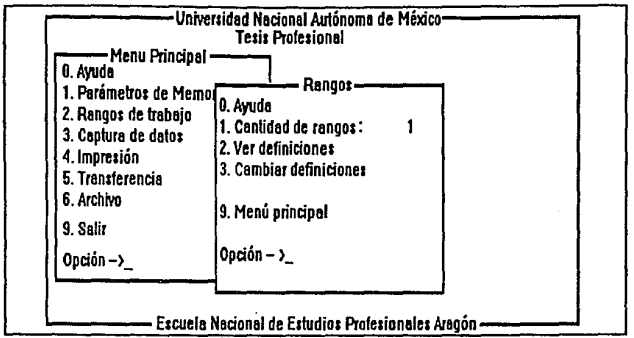


Figura 5-9 *Opciones del módulo de rangos.*

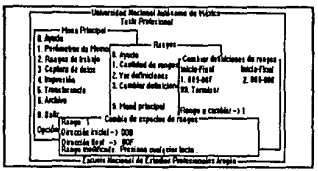


Figura 5-10 *Pantalla obtenida al terminar de modificar un rango.*

muestra la pantalla que se observa una vez cambiado un rango.

La definición de los rangos es una de las actividades más importantes. De ésta operación van a depender las funciones de captura de microinstrucciones, impresión y transferencia de datos. La modificación los rangos de ninguna manera afecta a los datos en sí.

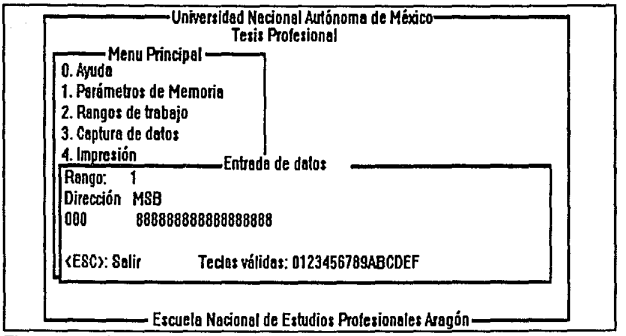


Figura 5-11 *Pantalla inicial para la captura de microinstrucciones.*

3. Captura de datos

Al seleccionar la opción 3 del menú principal, se abre una ventana para la entrada de microinstrucciones. En tal ventana se indica el rango actualmente accesado, la dirección correspondiente, así como las teclas habilitadas, ver figura 5-11. Se debe entonces introducir la microinstrucción, como una cadena de caracteres hexadecimales.

Cuando termine de introducir toda la microinstrucción aparecerá una ventana pidiendo que verifique el dato. Si el dato ha sido correctamente tecleado, pulse ENTER. La dirección avanzará entonces para que se introduzca el dato correspondiente. Este proceso se repetirá para todas las direcciones contenidas en los rangos previamente definidos.

En caso de cometer un error, se puede pulsar la tecla ESC y aparecerá la ventana con las opciones posibles. Oprima N, y vuelva a entrar el dato.

Si al aparecer la ventana de opciones se oprime la tecla R, entonces se avanzará al siguiente rango.

En caso de desear terminar sin pasar por todas las direcciones se puede pulsar la tecla T, con lo cual se regresará al menú principal.

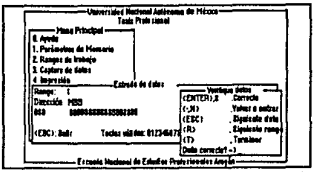


Figura 5-12 *Pantalla obtenida durante la captura de microinstrucciones.*

En la figura 5-12 se muestra la pantalla que se obtiene en éste módulo.

4. Impresión

Para proceder a imprimir, pulse la opción 4 del menú principal. Aparecerá una ventana con el menú de impresión. Prepare entonces la impresora y pulse la tecla 1 para continuar.

De esta manera se producirá un listado de los datos contenidos en cada dirección de memoria. El tamaño de la lista dependerá de la definición de los rangos.

La figura 5-13 presenta un listado obtenido, cuando se tiene un solo rango con direcciones 000 a 00F.

En caso de que el programa detecte que la impresora no se encuentra preparada, enviará un mensaje informando de tal situación. En este caso se deben revisar las conexiones de la impresora, y volver a intentar imprimir.

Universidad Nacional Autónoma de México
Escuela Nacional de Estudios Profesionales Aragón
TESIS DE INGENIERIA

DIRECCION	DATO
0000	8888888888888888
0001	8888888888888888
0002	16543F78EA0034656F
0003	0000000000000000
0004	89567FA888867AEBBB
0005	812012381AB1230E43
0006	AAAAAAAAAAAAAAAA
0007	121212001232218BFF
0008	98032BE121F12E12AB
0009	9A131BEC23C2659900
000A	FFFFFFFFFFFFFFFF
000B	2222222222222222
000C	3333333333333333
000D	AAAAAAAAAAAAAAAA
000E	3333333333333333
000F	AAAAAAAAAAAAAAAA

Figura 5-13 *Listado de microinstrucciones obtenido en una impresora.*

5. Transferencia

Antes de proceder a transferir los datos ya capturados, debe asegurarse de que los cables planos que conectan la tarjeta de interfase y el banco de memoria hayan sido correctamente conectados; así como también se deberá encender la energía del Sistema de Desarrollo. Se recomienda que todo ello se realice antes de iniciar el programa *MICROPRO*, cuando la PC este apagada. En caso de haber llegado hasta aquí, sin haber efectuado lo anterior, entonces puede Guardar sus datos en un archivo (opción 6), salir de MICROPRO, apagar la PC, realizar lo anteriormente recomendado, ingresar nuevamente a *MICROPRO* y llegar hasta este punto para proseguir.

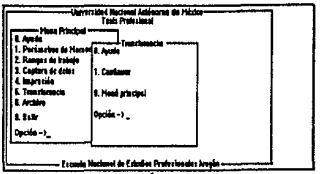


Figura 5-14 *Menú de opciones para la transferencia de microinstrucciones desde la PC hacia el banco de memoria.*

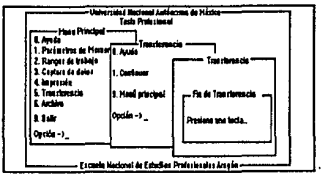


Figura 5-15 *Pantalla obtenida durante una transferencia de datos exitosa.*

Pulse la opción 5 del menú principal. Se abrirá una ventana como en la figura 5-14. En este momento se deben realizar las últimas verificaciones en el banco de memoria hasta estar completamente seguro que todo se encuentra correctamente. A continuación se debe seleccionar la opción de continuar, con lo cual la transferencia de datos se realizará. En el momento en que aparezca en la pantalla la indicación de que la transferencia a concluido, pulse cualquier tecla, figura 5-15.

En caso de que el programa detecte errores en el banco de memoria, se abrirá una ventana informando de tal situación, figura 5-16. En tal situación se debe revisar el banco de memoria a fin de detectar la falla, y luego volver a intentar.

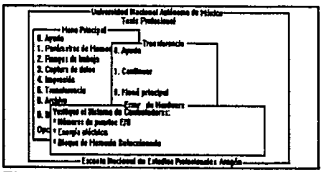


Figura 5-16 *Mensaje de error al intentar la transferencia de microinstrucciones.*

Cuando la computadora en la cual se esté operando tenga 640 KB de memoria y se haya seleccionado el espacio disponible $8000, la transferencia se efectuará hacia la memoria interna, figura 5-17. En una sección posterior se indica como analizar éste tipo de transferencia.

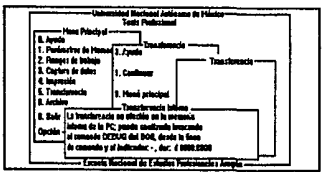


Figura 5-17 *Transferencia de microinstrucciones en memoria interna.*

6. Archivo

La opción de archivo se obtiene pulsando el número 6 en el menú principal. Se abrirá una ventana con tres opciones.

Se debe pulsar la opción de nombre para asignar un nombre al archivo que se desea accesar, figura 5-18. Este nombre puede incluir una ruta completa, sin exceder de 80 caracteres. Los siguientes son nombres válidos de archivos:

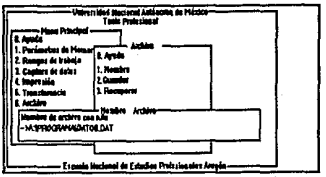


Figura 5-18 *Ventana de captura del nombre de archivo.*

datos1.dat

c:\tesis\programa\datos\datos2.dat

La opción de guardar se debe pulsar para almacenar en disco, los datos que se hayan introducido. Si el nombre de archivo ya existe se pedirá confirmación para sobrescribir el archivo. Esto eliminará el contenido previo del archivo.

La opción de recuperar intentará leer el archivo desde el disco. En caso de que no encuentre el archivo, se abrirá una ventana informando tal situación, figura 5-19.

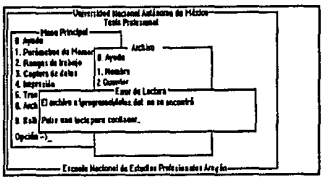


Figura 5-19 *Mensaje de error al no encontrar el archivo especificado.*

9. Salir de Micropro

Cuando haya terminado la sesión de Micropro, no debe desconectar la computadora y el monitor sin más, no señor(a,ita). Deberá salir correctamente del programa y volver al DOS.

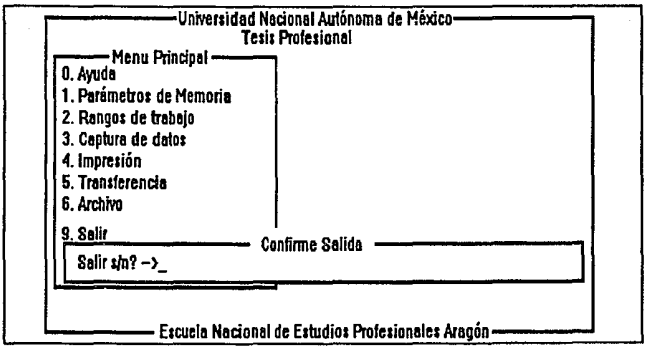


Figura 5-20 *Pantalla para confirmar la salida del programa micropro.*

Para salir de Micropro, pulse la opción 9 del menú principal. Aparecerá entonces una ventana pidiendo la confirmación para salir, figura 5-20.

Puesto que lo que desea es salir de Micropro, teclee s para escoger Sí. Volverá al DOS y ahora sí puede desconectar la computadora.

Nota: Si lo que hace es simplemente apagar la computadora sin salir de Micropro en la forma correcta, el banco de memoria puede ser dañado.

5.4 Transferencia en memoria interna

Si la transferencia de datos se realizó internamente, entonces se puede analizar cómo es que se distribuyen los datos en la memoria de la PC.

Una vez que se haya concluido una transferencia interna se puede salir de Micropro y con ayuda del comando DEBUG del DOS analizar los datos.

Como es sabido, el orden en el cual los CI de memoria RAM del banco de memoria están dispuestos es el siguiente:

MSB LSB

C9 C8 C7 C6 C5 C4 C3 C2 C1 C0

Donde *C#* = numero de Chip.

Y se utiliza el rango de memoria comprendido entre:

8000 - 893FF (Hex)

Sin embargo este rango no se utiliza en forma consecutiva, sino que únicamente los siguientes rangos específicos (Hex):

80000 A 803FF
81000 A 813FF
82000 A 823FF
83000 A 833FF
84000 A 843FF

85000 A 853FF
86000 A 863FF
87000 A 873FF
88000 A 883FF
89000 A 893FF

Entrando al depurador del DEBUG podemos apreciar como están organizados los datos en memoria.

1. Introduzca un disco del DOS en la unidad A: y teclee
2. A:DEBUG <ENTER>
3. Aparecerá el indicador:
4. -

Ahora, si deseáramos localizar (simulando) el dato contenido específicamente en un CHIP, podemos hacerlo de la siguiente forma:

1. Pedimos desplegar memoria con el comando dump (d):
2. d 8000:caaa <ENTER>
3. Aparecerá un listado de direcciones con sus datos.

donde *caaa* es un valor del offset o desplazamiento que podemos calcularlo como sigue:

c Es el numero de Chip, C# (véase distribución de CHIP's)

aaa Es la dirección en la cual esta almacenado el dato.

Por ejemplo si deseáramos saber el contenido del Chip # 5 en la dirección 0201, entonces daríamos ante el indicador del debug, el comando siguiente

- d 8000:5201

El cual nos desplegaría un conjunto de datos iniciando con esa dirección.

La forma en la cual hemos descrito la posición de los datos, puede hacer creer que almacenamos la información en forma desordenada. Sin embargo, esto no es así.

5.5 Separando el Banco de la PC.

Una vez que se ha completado la transferencia de datos, se puede desear desconectar el banco de memoria de la tarjeta adaptadora, para que opere libremente, con la fuente de poder del Sistema de Desarrollo.

Para ello, es recomendable que, si se desea separar el Sistema de desarrollo de la PC, una vez que se haya salido del programa MICROPRO, y se encuentre ante el indicador del sistema (por ejemplo, *a:>*) ponga el interruptor del sistema en la posición de *apagado*, y luego de unos 5 segundos, proceda a desconectar el banco de memoria de la tarjeta adaptadora.

Ahora el Sistema de Controladores tiene ya su banco de memoria con las microinstrucciones almacenadas, y por ello está listo para operar. Refiérase a la guía del usuario del Sistema de Desarrollo de Controladores Microprogramados para obtener mayor información, sobre el uso de dicho sistema.

Conclusiones

Se pueden citar dos ventajas esenciales de los controladores microprogramados:

1. La flexibilidad de uso, puesto que se puede obtener el conjunto de instrucciones requerido.

2. La posibilidad de realizar los sistemas rápidamente: Con el sistema de lógica alambrada, una operación compleja requiere implementar múltiples circuitos con una configuración particular. Con el sistema microprogramable, esa misma operación es ejecutada por una instrucción.

Sin embargo, la microprogramación presenta el inconveniente de ser poco práctica para ser manejada por el programador. En efecto, se requiere un perfecto conocimiento de la arquitectura del controlador, puesto que cada microinstrucción es un mando de registros y de bus.

El Sistema de Desarrollo de Controladores Microprogramados permite experimentar rápidamente en el diseño de sistemas de control microprogramado. El banco de memoria interfazado a la PC, otorga a tal Sistema un uso más eficiente.

La Computadora Personal es una herramienta ampliamente empleada en gran cantidad de ámbitos de la vida humana. Su facilidad de operación y su capacidad permiten asignarle múltiples usos. Un factor determinante en todo ello, es el referente a la amplia difusión de la literatura que se ha escrito sobre ella, así como la incorporación de nueva tecnología en su arquitectura. Por ello es que se eligió como parte de la solución en el banco de memoria.

Las tres funciones básicas generalmente usadas en una interfase con un microprocesador son: interrupciones, DMA y registros digitales de entrada/salida.

Nuestro diseño no hace uso de interrupciones debido a que no se requiere sincronizar eventos externos. DMA tampoco es usado, pues no se ejecutan transferencias a altas velocidades de/hacia la memoria de la PC. En cambio si se emplean registros de entrada/salida para sensar y controlar la operación del banco de memoria.

En el diseño presentado se partió de una idea simple: usar las señales implementadas en la PC que direccionan memoria, para crear un banco manejado por esas mismas señales, y localizarlo en un espacio no ocupado. Esta idea simplificó en cierta medida la circuitería, aunque, por supuesto, hubo necesidad de hacer uso de gran cantidad de buffers, dado lo amplio del tamaño de palabra del banco de memoria.

Ahora bien, el hardware no lo es todo, necesita de instrucciones que le digan que hacer, y cuando. El programa MICROPRO presentado, aunque aparece separado en un capítulo, está sumamente ligado al hardware. El

desarrollo de tal software implicó un trabajo agotador, pese a su sencillez. La depuración, corrección de fallas y cambio de prioridades presentaron una increible inversión de tiempo.

El usuario del banco de memoria y MICROPRO, puede fácilmente emplearlos correctamente, pues se muestran las pantallas que le presentará el programa, y se indican las precauciones que deberá observar en el manejo del hardware. La documentación y manuales de usuario son materiales indispensables en cualquier producto o servicio que va ha ser aprovechado por personal que no participó en el proyecto.

Considero que los objetivos del presente trabajo fueron muy ambiciosos, diseñar hardware y software, parece ser lo ideal dada su estrecha relación, sin embargo, productivamente no es recomendable. No obstante, la experiencia que he obtenido a lo largo, y sobre todo después de este proyecto, me presentan un panorama más completo sobre aspectos en la relación hardware-software.

Así entonces, sin pretender que es un trabajo perfecto, me considero satisfecho del estudio realizado, de las metas alcanzadas, pues cumplen lo encomendado.

Glosario

A

Acceso- almacenar, modificar o recuperar información en algún dispositivo o archivo, o bien hacer uso de él.

Acceso aleatorio- método de acceso a la información contenida en un dispositivo de almacenamiento. Implica que el tiempo requerido para accesar (en lectura y eventualmente escritura) una determinada localidad es invariable; no puede depender de las localidades anteriormente accesadas, ni del tipo de información leída o escrita. El funcionamiento de una cinta magnética es el opuesto y recibe el nombre de acceso secuencial.

Algoritmo- método para resolver un problema; conjunto de acciones que deben ejecutarse en un orden específico.

Analógico- representación contínua de los eventos del mundo real.

And, or, not (y, o, no)- compuertas primarias de lógica booleana usadas en el diseño de circuitos digitales.

ANSI- American National Standards Institute. Organización dedicada al desarrollo de normas para la industria de E.U.A.

Archivo- conjunto de registros relacionados.

Arreglo- serie de datos ordenados por renglones o por matrices.

ASCII- American Standard Code for Information Interchange. Modelo

para intercambio de información. Es un código de 7 bits, que permite 128 posibles combinaciones llamadas caracteres, 32 de los cuales están destinados a control de comunicaciones. Los caracteres ASCII con frecuencia se almacenan en Bytes, en donde el octavo bit puede emplearse como bit de paridad.

B

Basic- lenguaje de programación diseñado para resolver problemas matemáticos y de negocios. Se considera que BASIC es uno de los lenguajes de programación más fáciles de aprender.

BCD- decimal codificado en binario. Técnica para almacenamiento de números decimales en las computadoras.

Binario- dos; binario puede referirse a un sistema de numeración de base 2 en el que solo existen dos dígitos, 1 y 0; también puede referirse a la información o al software que, por tener forma de una serie de unos y ceros es por naturaleza entendible por la máquina.

Bipolar- categoría de diseño de circuitos microelectrónicos. El nombre proviene del uso simultáneo de cargas tanto positivas como negativas dentro del transistor.

Bit- dígito binario. Componente más pequeño del código binario; un bit es un solo dígito (0 o 1) en un número binario. Puede concebirse un bit como un estado de bombilla eléctrica que puede estar encendida o apagada.

Bit-Slice - procesador en rebanadas. Pastillas lógicas diseñadas para ser modificadas e integradas por el consumidor; los procesadores en rebanadas están diseñados como bloques elementales de construcción para el fabricante de equipo. Contienen circuitos que efectúan operaciones elementales y son susceptibles de ser microprogramados para crear un lenguaje de máquina especializado. Por lo general los procesadores Bit-Slice se construyen para trabajar en porciones de 4 bits de datos, que pueden configurarse para constituir procesadores más grandes (8 bits, 12 bits, etc).

Buffer- área de almacenamiento que conservainformacióntemporalmente.

Bus- ducto, ruta o canal común entre dispositivos del hardware.

Byte- unidad de almacenamiento equivalente a 8 bits o a un carácter de información; el byte es una unidad común de almacenamiento en un sistema de cómputo y es sinónimo de carácter de datos o de texto.

C

Cableado- interconstruído. Se aplica a circuitos de una computadora que son diseñados para realizar cierta función y que no pueden modificarse con facilidad.

Cadena- datos o texto alfanuméricos. Conjunto adyacente de caracteres tratado como una unidad

Campo- unidad definida de datos o información en un registro. Puede tener uno o más bytes de información.

Carácter- elemento alfanumérico; un carácter es una letra del alfabeto, un dígito numérico o un símbolo especial como el punto decimal o la coma.

Chip- pastilla. Circuitos electrónicos miniaturizados; contiene desde unos cuantos hasta varios cientos de miles de componentes electrónicos (transistores, resistencias, etc). Los términos pastilla, chip, circuito integrado y microelectrónico son sinónimos.

Ciclo de máquina- ciclo interno en el procesador de una computadora; los ciclos de máquina gobiernan los tiempos de las operaciones elementales dentro de la computadora.

Circuito- dispositivo electrónico o canal de comunicación para señales eléctricas.

Circuito electrónico- conjunto de componentes elctrónicos que realizan una función. Definen una serie de trayectorias a través de las que fluye la electricidad, siguiendo un esquema predefinido para la realización de una cierta tarea.

Circuito impreso- (tarjeta de). Tarjeta plana que contiene microcircuitos y otros componentes electrónicos conectados por trayectorias impresas eléctricamente conductoras.

Circuito lógico. Circuito electrónico que realiza funciones de procesamiento. Los circuitos lógicos conforman los procesadores y las unidades de control. Se diseñan como combinaciones de compuertas lógicas que son los bloques de construcción de la lógica booleana.

Codigo fuente- Instrucciones escritas en lenguaje original de programación.

Código objeto- instrucciones que pueden ser ejecutadas directamente por la computadora. Un código objeto es la salida de los ensambladores y compiladores. Equivale al lenguaje de máquina.

Compilador- Traductor de lenguajes de programación de alto nivel. Es un programa que traduce un lenguaje de alto nivel al lenguaje de máquina de una computadora en particular.

Compuerta- dispositivo de apertura y cierre. Combinación de interruptores diseñados de acuerdo con la lógica booleana.

Compuerta lógica- componente de los circuitos lógicos digitales. Son combinaciones de transistores que detectan la presencia o ausencia de pulsos y son utilizadas en el diseño de circuitos lógicos.

Computadora- Máquina programable para el procesamiento de información. Se constituye de hardware y software.

Computadora analógica- computadora que acepta y procesa señales electrónicas análogas a las del mundo real (como fluctuaciones de voltaje).

Computadora Personal- Computadora empleada para uso doméstico o personal. Es una microcomputadora de propósito general.

Constante- información en un programa con valores fijos.

D

Datos- unidades de información que pueden definirse con precisión. Técnicamente, es la materia prima que al ser procesada da lugar a la información.

Digital- discontinuo, discreto, controlado por computadora digital.

DIP interruptor- Dual In Package Interuptor. Interruptores de dos posiciones, en línea.

Dirección- número de posición particular de la memoria o sistema de almacenamiento; cada posición particular en la memoria (palabra o byte) tiene una dirección o número únicos, como sucede con los apartados postales.

DMA- Acceso directo a memoria; método rápido para intercambiar datos con la memoria.

DOS- Disk Operating System. Sistema Operativo de Disco.

E

EBCDIC- Extended Binary Coded Decimal Interchange Code. Código extendido de intercambio decimal

codificado en binario. Se constituye de 8 bits (un byte), que permite 256 combinaciones de caracteres posibles.

EPROM- Erasable and Programable Read Only Memory. Memoria de lectura exclusiva reprogramable.

E/S- Entrada/Salida. Transferencia de datos entre un CPU y un dispositivo periférico.

Escritura- Información grabada o registrada en un dispositivo de almacenamiento.

Espacio direccionable- cantidad de memoria a la que se puede tener acceso.

Estado sólido- circuitos electrónicos fabricados con materiales sólidos (microcircuitos). Contrasta con cintas.

F

Firmware- Software permanente que debe permanecer aun en el caso de una interrupción en el suministro de energía eléctrica.

Flujograma- Diagrama de flujo. Imagen gráfica de la secuencia de operaciones de un programa o de un sistema de información.

Frecuencia- ciclos y oscilaciones por segundo.

H

Hardware- la maquinaria. El cpu y todos los periféricos. Cualquier dispositivo que contrasta con el software.

Hertz- unidad de medida de la frecuencia; el número de oscilaciones eléctricas por unidad de tiempo se mide en HERTZ. El nombre de esta unidad se deriva del apellido de Heinrich 'Hertz, quien en 1883 detectó las ondas electromangéticas.

Hexadecimal- número basado en 16 digitos; Este sistema de numeración es usado como un medio abreviado para representar número binarios. Cada 4 bits se convierten en un solo dígito hexadecimal, ejemplo,

Decimal	Binario	Hexadecimal
0	0000	0
1	0001	1
15	1111	F

I

IBM- International Business Machines Corporation.

IC- Integrated Circuit. véase CHIP.

Instrucción- comando dirigido a la computadora.

Interface- interfaz. Interconexión entre elementos de hardware y/o de software; las interfaces de hardware son trayectorias físicas que deben

conectar e intercambiar señales electrónicas en un orden preestablecido.

Interprete- traductor de lenguaje de programación de alto nivel a codigo binario, al momento de ejecutar.

Interrupción- Un suspension en el flujo normal de un programa que permite continuar el flujo posteriormente.

L

Lectura- se refiere a la transferencia a la memoria de la computadora, de una copia de información proveniente de un dispositivo como un disco.

Lenguaje de alto nivel- lenguaje orientado a la solución de problemas. Están diseñados para permitir al programador concentrarse en la lógica del problema a resolver, liberándolo de la necesidad de un conocimiento profundo del lenguaje de máquina de la computadora.

Lenguaje C- Lenguaje de programación estructurado y de alto nivel, desarrollado por Bell Laboratories; El C es un lenguaje de compilador, notorio por sus posibilidades para el manejo de condiciones que normalmente necesitan escribirse en lenguaje ensamblador.

Lenguaje de máquina- Lenguaje original de la computadora, directamente ejecutable o entendible por ella.

Lenguaje de programación- Lenguaje empleado por los programadores para desarrollar instrucciones para la computadora.

Lógica Booleana- Concepto primario en el diseño de computadoras digitales. Los pulsos eléctricos generados por el reloj de la computadora son dirigidos a circuitos constituidos por componentes lógicos, denominados compuertas. Las salidas de unas compuertas son las entradas de otras, y así sucesivamente. Las reglas que rigen el comportamiento de estos pulsos al entrar y salir de las compuertas, son booleanas, nombre que reciben en honor del matemático inglés George Boole, quien a mediados del siglo XIX desarrollo las matemáticas de la lógica.

Lenguaje ensamblador- Lenguaje de programación que permite escribir programas a nivel de lenguaje de máquina.

LSI- Large Scale Integration. Alta escala de integración. Se refiere a los microcircuitos formados por un gran

número de componentes electrónicos. La complejidad de un circuito LSI varía de 3,000 a 100,000 transistores en solo chip.

M

Matriz- arreglo de renglones y columnas.

Memoria- almacenamiento de trabajo de la computadora.

Menú- lista de las opciones disponibles en un programa interactivo.

Microcomputadora- computadora de pequeño tamaño.

Microprocesador- procesador completo en una sola pastilla.

Microprogramado- traducción de instrucciones de máquina. El microprogramado es una característica arquitectónica del diseño de digital, que permite que se desarrollen y agreguen con mayor facilidad instrucciones adicionales. Las nuevas instrucciones se diseñan de acuerdo con las reglas de microprogramado establecidas y se agregan a la memoria de firmware (el banco de memoria, en esta tesis). De esta forma, no es necesario volver a diseñar los circuitos electrónicos. Al proceso de desarrollo de intrucciones en microprogramado se le denomina microprogramación.

Mnemónico- nombre simbólico asignado a programas y datos.

Modelo- representación matemática de un dispositivo o proceso.

MSI- Medium Scale Integration. Mediana escala de integración; se refiere a un número pequeño de componentes electrónicos incorporados a un solo chip. La complejidad de un MSI va desde 100 hasta 3,000 transistores.

P

Palabra- unidad interna de almacenamiento de la computadora. La palabra tiene un cierto número de bits y determina la velocidad general de procesamiento. Mientras más grande sea la palabra, más instrucciones o información se procesan como una sola unidad.

Paquete de software- programa o serie de programas ya empacados, como un producto listo para su venta y uso.

Parámetro- Cualquier valor de una variable.

Paridad- técnica de detección de errores. Consiste en agregar un sólo bit a cada byte, basándose en el nú-

mero total de 0s y 1s que éste contiene.

Pascal- Lenguaje de programación de alto nivel, notable por su simplicidad y su diseño de programación estructurada.

PC- véase Computadora Personal.

Pila- Area reservada para operaciones internas. Funciona siguiendo una secuencia de último en llegar primero en salir.

Procedimiento- Serie de pasos para realizar una tarea específica.

Procesador- Sección de la computadora constituida por el CPU. Con frecuencia los términos procesador y CPU se utilizan como sinónimos.

Proceso, procesar- manipular información.

Programa- grupo de instrucciones que indica a la computadora como realizar una función específica; consta de instrucciones, variables y constantes.

Programable- capaz de aceptar un programa.

Programación- desarrollo de un programa de computadora.

Programa de aplicación- programa específico del usuario. En contraste se tienen los programas de sistemas, como los sistemas operativos y sistemas de manejo de bases de datos. Los programas de aplicación corren bajo el control de un sistema operativo.

PROM- Programmable Read Only Memory. Memoria de lectura exclusiva/programable. Microcircuito de memoria permanente para el almacenamiento de programas.

Prototipo- Sistema de desarrollo con participación del usuario.

Puerto- Interfaz de canal de comunicaciones. La conexión entre un dispositivo de hardware y un canal de entrada/salida.

R

RAM- Random Access Memory. Memoria de lectura/escritura. En estas memorias, cada byte individual de información puede ser introducido o extraído de manera independiente del resto de la información contenida en la memoria. Las celdas de almacenamiento (bits) de una memoria RAM requieren cierta energía para retener su contenido; si falla la energía, la información se pierde.

RAM dinámica- Chips de memoria que necesitan ser refrescadas continuamente, para conservar sus datos.

RAM estática- Son memoria RAM que requieren energía para conservar su contenido. Sin embargo, a diferencia de las RAM dinámicas, no requieren de una generación continua del contenido. El contenido binario de la celda de almacenamiento se mantiene por medio de una fuente regulada de energía.

Registro- grupo de campos de datos relacionados; es un conjunto de datos sobre un sujeto o tema.

Registro- circuitos contadores y memorias buffer en el procesador que vigilan las secuencias de varias actividades.

Reloj- Dispositivo que genera un número uniforme de pulsos eléctricos por segundo (a menudo se utiliza un cristal de cuarzo); Estos pulsos se emplean para determinar los límites de los ciclos de máquina y dar origen a los pulsos digitales que cruzan los circuitos electrónicos.

ROM- Read Only Memory. Memoria de lectura exclusiva. Pastilla de memoria permanente. Se almacenan datos o instrucciones en el momento de su fabricación y no pueden alterarse.

S

Silicio- material básico para la fabricación de circuitos microelectrónicos.

Semiconductor- Sustancias cuyas características eléctricas pueden ser notablemente alteradas mediante la inyección de ciertas impurezas. Un semiconductor se encuentra en algún punto entre conductor y un aislante. Un semiconductor puede actuar como un interruptor que se puede activar eléctricamente.

Sistema- conjunto de componentes y eventos relacionados que interactúan unos con otros para ejecutar una tarea.

Sistema Operativo- Programa de control principal que determina la operación de la computadora. El sistema operativo es el primer programa que se copia en la memoria de la computadora a partir de un disco o cinta, después de que ésta se enciende por primera vez.

SLSI- Super Large Scale Integration. Super gran escala de integracion. Se refiere a los chips de densidad ultra-alta que contiene más de 1 millón de transistores.

Software- Instrucciones de computadora. **Software de aplicación-** uso específico de la computadora.

SSI- Small Scale Integration. Pequeña escala de integración; se refiere a chips fabricados con un pequeño número de componentes electrónicos. La complejidad de los SSI varía aproximadamente de 2 a 100 transistores.

Subrutina- grupo de instrucciones en un programa que realizan una función específica; equivale a un módulo de programa.

T

Transistor- elemento semiconductor primario; los transistores son los bloques físicos empleados en la construcción de circuitos digitales y memoria. También se emplean para amplificar señales. En las computadoras digitales, el transistor se emplea principalmente como un interruptor eléctrico.

Traslape- (Overlay). Segmento secundario de programa que se copia encima de instrucciones ya existentes en la memoria (pero que se pueden copiar nuevamente desde disco o cinta en caso de necesitarse)

Tiempo de Acceso- tiempo de respuesta del disco o memoria; el tiempo de acceso a la memoria se refiere a la velocidad de la memoria (por ejemplo, qué tan rápido se puede intercambiar información entre el procesador y la memoria).

U

Unidad Aritmética y Lógica (ALU)- es el conjunto de circuitos de una computadora que realiza las funciones aritméticas, las funciones lógicas y las funciones de comparación.

Unidad de control- controlador; componentes de hardware que dirigen a la computadora y/o a las actividades de dispositivos periféricos.

V

Ventana- zona de visión en una pantalla de video.

VLSI- Very Large Scale Integration. Muy alta escala de integración. Se refiere al gran número de componentes electrónicos interconstruídos en un solo chip. La complejidad VLSI varía aproximadamente de 100,000 a 1 millón de transistores en una sola pastilla.

Materiales

En las páginas siguientes se presenta una lista de los materiales básicos que se emplearon en el diseño del banco de memoria y su interfase. Después se presentan las hojas técnicas de los principales dispositivos electrónicos que forman parte del diseño propuesto.

Materiales utilizados en el diseño del banco de memoria

La tabla 1, presenta una lista de los circuitos integrados usados en el diseño del banco de memoria y su interfase. Por otra parte, la tabla 2 muestra otros componentes importantes del diseño, tales como cables y conectores.

Tabla 1

Código	Descripción	Cantidad
74LS04	Inversores	1
74LS125	Buffers de 3 estados	2
74LS138	Decodificador 3 x 8	2
74LS154	Decodificador 4 x 16	1
74LS244	Buffers de 3 estados	35
74LS245	Buffers bidireccionales de 3 estados	1
74LS273	Circuito retenedor	1
74LS688	Comparador octal	2
TMM2016AP-12	Memoria RAM 2K x 8	10

Tabla 2

Descripción	Cantidad
Interruptores DIP de 8 vias	2
Resistencias de 8.2 K Ohms, 1/4 W	10
Cable Plano 16 vias	2 metros
Conector para cable plano "hembra"	2
Conector para cable plano "macho"	2
Terminales para "Jumper" de 16 pines duales	2

National Semiconductor Corporation

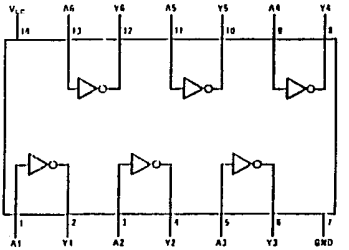
DM54LS04/DM74LS04 Hex Inverting Gates

General Description

This device contains six independent gates each of which performs the logic INVERT function.

Connection Diagram

Dual-In-Line Package



TL/F/6345–1

Order Number DM54LS04J, DM74LS04M or DM74LS04N
See NS Package Number J14A, M14A or N14A

Function Table

$Y = \overline{A}$

Input	Output
A	Y
L	H
H	L

H = High Logic Level
L = Low Logic Level

Absolute Maximum Ratings (Note)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM54LS	−55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: The "Absolute Maximum Ratings" are those values beyond which the safety of the device can not be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	DM54LS04			DM74LS04			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−0.4			−0.4	mA
I_{OL}	Low Level Output Current			4			8	mA
T_A	Free Air Operating Temperature	−55		125	0		70	°C

Electrical Characteristics over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions		Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max	DM54	2.5	3.4		V
			DM74	2.7	3.4		
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IH} = Min	DM54		0.25	0.4	V
			DM74		0.35	0.5	
		I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	µA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	mA
			DM74	−20		−100	
I_{CCH}	Supply Current with Outputs High	V_{CC} = Max			1.2	2.4	mA
I_{CCL}	Supply Current with Outputs Low	V_{CC} = Max			3.6	6.6	mA

Switching Characteristics at V_{CC} = 5V and T_A = 25°C (See Section 1 for Test Waveforms and Output Load)

Symbol	Parameter	R_L = 2 kΩ				Units
		C_L = 15 pF		C_L = 50 pF		
		Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time Low to High Level Output	3	10	4	15	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	3	10	4	15	ns

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

National Semiconductor Corporation

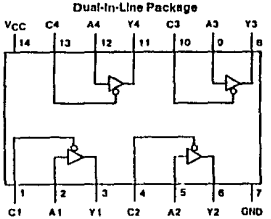
DM54LS125A/DM74LS125A Quad TRI-STATE® Buffers

General Description

This device contains four independent gates each of which performs a non-inverting buffer function. The outputs have the TRI-STATE feature. When enabled, the outputs exhibit the low impedance characteristics of a standard LS output with additional drive capability to permit the driving of bus lines without external resistors. When disabled, both the output transistors are turned off presenting a high-impedance state to the bus line. Thus the output will act neither as a significant load nor as a driver. To minimize the possibility that two outputs will attempt to take a common bus to opposite logic levels, the disable time is shorter than the enable time of the outputs.

Connection Diagram

Dual-In-Line Package



TL/F/6387-1

Order Number DM54LS125AJ, DM74LS125AM or DM74LS125AN
See NS Package Number J14A, M14A or N14A

Function Table

Y = A

Inputs		Output
A	C	Y
L	L	L
H	L	H
X	H	HI-Z

H = High Logic Level
L = Low Logic Level
X = Either Low or High Logic Level
HI-Z = TRI-STATE (Outputs are disabled)

Absolute Maximum Ratings (Note)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM54LS	−55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.*

Recommended Operating Conditions

Symbol	Parameter	DM54LS125A			DM74LS125A			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−1			−2.6	mA
I_{OL}	Low Level Output Current			12			24	mA
T_A	Free Air Operating Temperature	−55		125	0		70	°C

Electrical Characteristics over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions		Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min		2.4	3.4		V
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max	DM54		0.25	0.4	V
			DM74		0.35	0.5	
		I_{OL} = 12 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	µA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.4	mA
I_{OZH}	Off-State Output Current with High Level Output Voltage Applied	V_{CC} = Max, V_O = 2.4V, V_{IH} = Min, V_{IL} = Max				20	µA
I_{OZL}	Off-State Output Current with Low Level Output Voltage Applied	V_{CC} = Max, V_O = 0.4V, V_{IH} = Min, V_{IL} = Max				−20	µA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	mA
			DM74	−20		−100	
I_{CC}	Supply Current	V_{CC} = Max (Note 3)			11	20	mA

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 3: I_{CC} is measured with the data control (C) inputs at 4.5V and the data inputs grounded.

Switching Characteristics at V_{CC} = 5V and T_A = 25°C (See Section 1 for Test Waveforms and Output Load)

Symbol	Parameter	R_L = 667Ω				Units
		C_L = 50 pF		C_L = 150 pF		
		Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time Low to High Level Output		15		21	ns
t_{PHL}	Propagation Delay Time High to Low Level Output		18		22	ns
t_{PZH}	Output Enable Time to High Level Output		25		35	ns
t_{PZL}	Output Enable Time to Low Level Output		25		40	ns
t_{PHZ}	Output Disable Time from High Level Output (Note 1)		20			ns
t_{PLZ}	Output Disable Time from Low Level Output (Note 1)		20			ns

Note 1: C_L = 5pF.

National Semiconductor Corporation

DM54LS138/DM74LS138, DM54LS139/DM74LS139 Decoders/Demultiplexers

General Description

These Schottky-clamped circuits are designed to be used in high-performance memory-decoding or data-routing applications, requiring very short propagation delay times. In high-performance memory systems these decoders can be used to minimize the effects of system decoding. When used with high-speed memories, the delay times of these decoders are usually less than the typical access time of the memory. This means that the effective system delay introduced by the decoder is negligible.

The LS138 decodes one-of-eight lines, based upon the conditions at the three binary select inputs and the three enable inputs. Two active-low and one active-high enable inputs reduce the need for external gates or inverters when expanding. A 24-line decoder can be implemented with no external inverters, and a 32-line decoder requires only one inverter. An enable input can be used as a data input for demultiplexing applications.

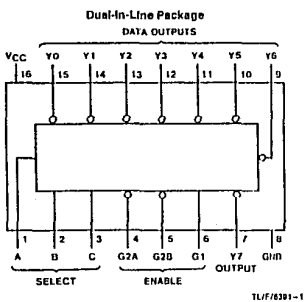
The LS139 comprises two separate two-line-to-four-line decoders in a single package. The active-low enable input can be used as a data line in demultiplexing applications.

All of these decoders/demultiplexers feature fully buffered inputs, presenting only one normalized load to its driving circuit. All inputs are clamped with high-performance Schottky diodes to suppress line-ringing and simplify system design.

Features

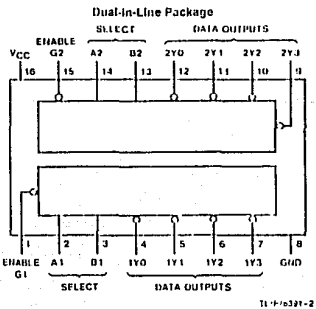
- Designed specifically for high speed:
  - Memory decoders
  - Data transmission systems
- LS138 3-to-8-line decoders incorporates 3 enable inputs to simplify cascading and/or data reception
- LS139 contains two fully independent 2-to-4-line decoders/demultiplexers
- Schottky clamped for high performance
- Typical propagation delay (3 levels of logic)
  - LS138 21 ns
  - LS139 21 ns
- Typical power dissipation
  - LS138 32 mW
  - LS139 34 mW

Connection Diagrams



TL/F/6391–1

Order Number DM54LS138J, DM74LS138M or DM74LS138N
See NS Package Number J16A, M16A or N16A



TL/F/6391–2

Order Number DM54LS139J, DM74LS139M or DM74LS139N
See NS Package Number J16A, M16A or N16A

Absolute Maximum Ratings (Note)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM54LS	−55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	DM54LS138			DM74LS138			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−0.4			−0.4	mA
I_{OL}	Low Level Output Current			4			8	mA
T_A	Free Air Operating Temperature	−55		125	0		70	°C

'LS138 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions		Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	DM54	2.5	3.4		V
			DM74	2.7	3.4		
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min	DM54		0.25	0.4	V
			DM74		0.35	0.5	
		I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	µA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	mA
			DM74	−20		−100	
I_{CC}	Supply Current	V_{CC} = Max (Note 3)			6.3	10	mA

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 3: I_{CC} is measured with all outputs enabled and open.

'LS139 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions		Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	DM54	2.5	3.4		V
			DM74	2.7	3.4		
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min	DM54		0.25	0.4	V
			DM74		0.35	0.5	
		I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	µA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	mA
			DM74	−20		−100	
I_{CC}	Supply Current	V_{CC} = Max (Note 3)			6.6	11	mA

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 3: I_{CC} is measured with all outputs enabled and open.

'LS139 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C (See Section 1 for Test Waveforms and Output Load)

Symbol	Parameter	From (Input) To (Output)	R_L = 2 kΩ				Units
			C_L = 15 pF		C_L = 50 pF		
			Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time Low to High Level Output	Select to Output		18		27	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Select to Output		27		40	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Enable to Output		18		27	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Enable to Output		24		40	ns

Function Tables

LS138

Inputs					Outputs							
Enable		Select										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

* G2 = G2A + G2B

H = High Level, L = Low Level, X = Don't Care

LS139

Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = High Level, L = Low Level, X = Don't Care

LS138 Switching Characteristics

V_{CC} = 5V and T_A = 25°C (See Section 1 for Test Waveforms and Output Load)

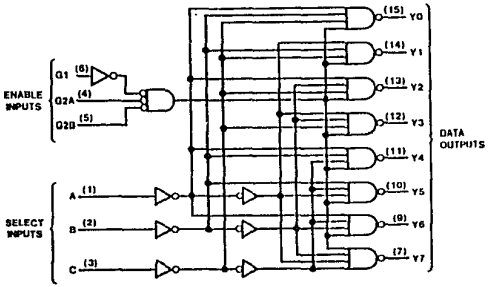
Symbol	Parameter	From (Input) To (Output)	Levels of Delay	R_L = 2 kΩ				Units
				C_L = 15 pF		C_L = 50 pF		
				Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time Low to High Level Output	Select to Output	2		18		27	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Select to Output	2		27		40	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Select to Output	3		18		27	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Select to Output	3		27		40	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Enable to Output	2		18		27	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Enable to Output	2		24		40	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Enable to Output	3		18		27	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	Enable to Output	3		28		40	ns

Recommended Operating Conditions

Symbol	Parameter	DM54LS139			DM74LS139			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−0.4			−0.4	mA
I_{OL}	Low Level Output Current			4			8	mA
T_A	Free Air Operating Temperature	−55		125	0		70	°C

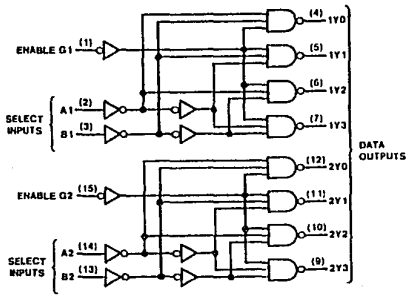
Logic Diagrams

LS138



TL/F/6391-3

LS139



TL/F/6391-4

National Semiconductor Corporation

DM54LS154/DM74LS154 4-Line to 16-Line Decoders/Demultiplexers

General Description

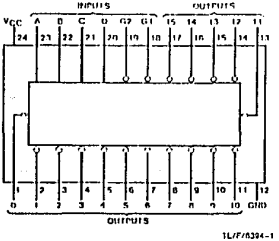
Each of these 4-line-to-16-line decoders utilizes TTL circuitry to decode four binary-coded inputs into one of sixteen mutually exclusive outputs when both the strobe inputs, G1 and G2, are low. The demultiplexing function is performed by using the 4 input lines to address the output line, passing data from one of the strobe inputs with the other strobe input low. When either strobe input is high, all outputs are high. These demultiplexers are ideally suited for implementing high-performance memory decoders. All inputs are buffered and input clamping diodes are provided to minimize transmission-line effects and thereby simplify system design.

Features

- Decodes 4 binary-coded inputs into one of 16 mutually exclusive outputs
- Performs the demultiplexing function by distributing data from one input line to any one of 16 outputs
- Input clamping diodes simplify system design
- High fan-out, low-impedance, totem-pole outputs
- Typical propagation delay
  3 levels of logic 23 ns
  Strobe 19 ns
- Typical power dissipation 45 mW

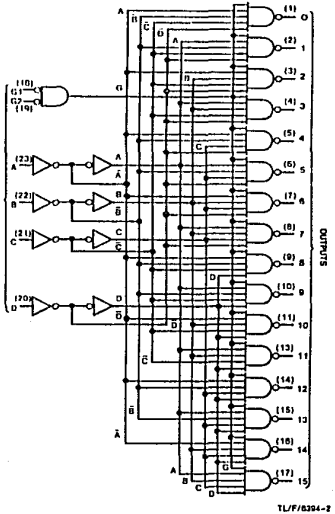
Connection and Logic Diagrams

Dual-In-Line Package



TL/F/6394–1

Order Number DM54LS154J,
DM74LS154WM or DM74LS154N
See NS Package Number J24A, M24B or N24A



TL/F/6394–2

ıte Maximum Ratings (Note)

tions for Military/Aerospace products are not in this datasheet. Refer to the associated electrical test specifications document.

tage	7V
ge	7V
Free Air Temperature Range	
	−55°C to +125°C
	0°C to +70°C
ımperature Range	−65°C to +150°C

Note: *The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.*

ımended Operating Conditions

Parameter	DM54LS154			DM74LS154			Units
	Min	Nom	Max	Min	Nom	Max	
Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
High Level Input Voltage	2			2			V
Low Level Input Voltage			0.7			0.8	V
High Level Output Current			−0.4			−0.4	mA
Low Level Output Current			4			8	mA
Free Air Operating Temperature	−55		125	0		70	°C

cal Characteristics over recommended operating free air temperature range (unless otherwise noted)

Parameter	Conditions		Min	Typ (Note 1)	Max	Units
Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	DM54	2.5	3.4		V
		DM74	2.7	3.4		
Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min	DM54		0.25	0.4	V
		DM74		0.35	0.5	
	I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.1	mA
High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	µA
Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.4	mA
Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	mA
		DM74	−20		−100	
Supply Current	V_{CC} = Max (Note 3)			9	14	mA

picals are at V_{CC} = 5V, T_A = 25°C

more than one output should be shorted at a time, and the duration should not exceed one second

measured with all outputs open and all inputs grounded.

ing Characteristics at V_{CC} = 5V and T_A = 25°C (See Section 1 for Test Waveforms and Output Load)

Parameter	From (Input) To (Output)	R_L = 2 kΩ				Units
		C_L = 15 pF		C_L = 50 pF		
		Min	Max	Min	Max	
Propagation Delay Time Low to High Level Output	Data to Output		30		35	ns
Propagation Delay Time High to Low Level Output	Data to Output		30		35	ns
Propagation Delay Time Low to High Level Output	Strobe to Output		20		25	ns
Propagation Delay Time High to Low Level Output	Strobe to Output		25		35	ns

Function Table

Inputs						Outputs														
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = High Level, L = Low Level, X = Don't Care

MOTOROLA

SN54/74LS240
SN54/74LS241
SN54/74LS244

DESCRIPTION — The SN54LS/74LS240, 241 and 244 are Octal Buffers and Line Drivers designed to be employed as memory address drivers, clock drivers and bus-oriented transmitters/receivers which provide improved PC board density.

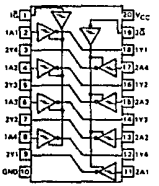
- HYSTERESIS AT INPUTS TO IMPROVE NOISE MARGINS
- 3-STATE OUTPUTS DRIVE BUS LINES OR BUFFER MEMORY ADDRESS REGISTERS
- INPUT CLAMP DIODES LIMIT HIGH-SPEED TERMINATION EFFECTS

OCTAL BUFFER/LINE DRIVER WITH 3-STATE OUTPUTS

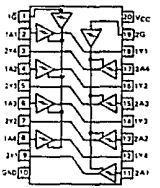
LOW POWER SCHOTTKY

LOGIC AND CONNECTION DIAGRAMS DIP (TOP VIEW)

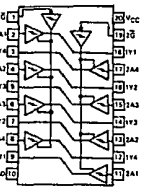
SN54LS/74LS240



SN54LS/74LS241



SN54LS/74LS244



TRUTH TABLES

SN54LS/74LS240

INPUTS		OUTPUT
1G̅,2G̅	D	
L	L	H
L	H	L
H	X	(Z)

SN54LS/74LS244

INPUTS		OUTPUT
1G̅,2G̅	D	
L	L	L
L	H	H
H	X	(Z)

SN54LS/74LS241

INPUTS		OUTPUT	INPUTS		OUTPUT
1G̅	D		2G̅	D	
L	L	L	H	L	L
L	H	H	H	H	H
H	X	(Z)	L	X	(Z)

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial
Z = HIGH Impedance

J Suffix — Case 732-03 (Ceramic)
N Suffix — Case 738-03 (Plastic)

GUARANTEED OPERATING RANGES

SYMBOL	PARAMETER		MIN	TYP	MAX	UNIT
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54,74			−3.0	mA
		54 74			−12 −15	mA
I_{OL}	Output Current — Low	54 74			12 24	mA

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
$V_{T+} - V_{T-}$	Hysteresis		0.2	0.4		V	V_{CC} = MIN
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54,74	2.4	3.4		V	V_{CC} = MIN, I_{OH} = −3.0 mA
		54,74	2.0			V	V_{CC} = MIN, I_{OH} = MAX
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	I_{OL} = 12 mA, V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 24 mA, V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
I_{OZH}	Output Off Current HIGH				20	µA	V_{CC} = MAX, V_{OUT} = 2.7 V
I_{OZL}	Output Off Current LOW				−20	µA	V_{CC} = MAX, V_{OUT} = 0.4 V
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V
I_{IL}	Input LOW Current				−0.2	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current		−40		−225	mA	V_{CC} = MAX
I_{CC}	Power Supply Current Total, Output HIGH				27	mA	V_{CC} = MAX
	Total, Output LOW	LS240			44		
		LS241/244			46		
	Total at HIGH Z	LS240			50		
		LS241/244			54		

= 5.0 V

		LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
	Delay, Data to Output		9.0 12	14 18	ns	
	Propagation Delay, Data to Output		12 12	18 18	ns	C_L = 45 pF, R_L = 667 Ω
	Output Enable Time to HIGH Level		18	23	ns	
	Output Enable Time to LOW Level		20	30	ns	
	Output Disable Time from LOW Level		15	25	ns	C_L = 5.0 pF R_L = 667 Ω
	Output Disable Time from HIGH Level		10	18	ns	

AC WAVEFORMS

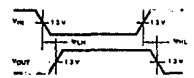


Fig. 1

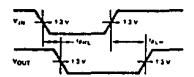


Fig. 2

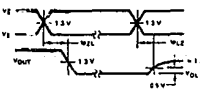


Fig. 3

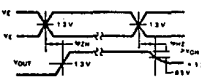
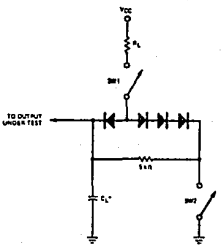


Fig. 4



SWITCH POSITIONS

SYMBOL	SW1	SW2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Open
t_{PLZ}	Closed	Closed
t_{PHZ}	Closed	Closed

Fig. 5

MOTOROLA

SN54/74LS245

OCTAL BUS TRANSCEIVER

LOW POWER SCHOTTKY

DESCRIPTION — The SN54LS/74LS245 is an Octal Bus Transmitter/Receiver designed for 8-line asynchronous 2-way data communication between data buses. Direction Input (DR) controls transmission of Data from bus A to bus B or bus B to bus A depending upon its logic level. The Enable input (Ē) can be used to isolate the buses.

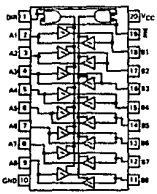
- HYSTERESIS INPUTS TO IMPROVE NOISE IMMUNITY
- 2-WAY ASYNCHRONOUS DATA BUS COMMUNICATION
- INPUT DIODES LIMIT HIGH-SPEED TERMINATION EFFECTS

TRUTH TABLE

INPUTS		OUTPUT
Ē	DIR	
L	L	Bus B Data to Bus A
L	H	Bus A Data to Bus B
H	X	Isolation

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

LOGIC AND CONNECTION DIAGRAM
DIP (TOP VIEW)



J Suffix — Case 732-03 (Ceramic)
N Suffix — Case 738-03 (Plastic)

SN54/74LS245

GUARANTEED OPERATING RANGES

SYMBOL	PARAMETER		MIN	TYP	MAX	UNIT
V_{CC}	Supply Voltage	54	4.5	5.0	5.5	V
		74	4.75	5.0	5.25	
T_A	Operating Ambient Temperature Range	54	−55	25	125	°C
		74	0	25	70	
I_{OH}	Output Current — High	54,74			−3.0	mA
		54			−12	mA
		74			−15	
I_{OL}	Output Current — Low	54			12	mA
		74			24	

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
$V_T+ - V_T-$	Hysteresis		0.2	0.4		V	V_{CC} = MIN
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54,74	2.4	3.4		V	V_{CC} = MIN, I_{OH} = −3.0 mA
		54,74	2.0			V	V_{CC} = MIN, I_{OH} = MAX
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	I_{OL} = 12 mA; V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 24 mA
I_{OZH}	Output Off Current HIGH				20	μA	V_{CC} = MAX, V_{OUT} = 2.7 V
I_{OZL}	Output Off Current LOW				−200	μA	V_{CC} = MAX, V_{OUT} = 0.4 V
I_{IH}	Input HIGH Current	A or B, DIR or $\overline{E}$			20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
		DIR or $\overline{E}$			0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V
		A or B			0.1	mA	V_{CC} = MAX, V_{IN} = 5.5 V
I_{IL}	Input LOW Current				−0.2	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Output Short Circuit Current		−40		−225	mA	V_{CC} = MAX
I_{CC}	Power Supply Current Total Output HIGH				70	mA	V_{CC} = MAX
	Total, Output LOW				90		
	Total at HIGH Z				95		

AC CHARACTERISTICS: T_A = 25°C, V_{CC} = 5.0 V

SYMBOL	PARAMETER	LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS
t_{PLH} t_{PHL}	Propagation Delay Data to Output		8.0 8.0	12 12	ns	C_L = 45 pF, R_L = 667 Ω
t_{PZH}	Output Enable Time to HIGH Level		25	40	ns	
t_{PZL}	Output Enable Time to LOW Level		27	40	ns	
t_{PHZ}	Output Disable Time from LOW Level		15	25	ns	C_L = 5.0 pF, R_L = 667 Ω
t_{PHZ}	Output Disable Time from HIGH Level		15	25	ns	

SN54/74LS273

OCTAL D FLIP-FLOP WITH CLEAR

LOW POWER SCHOTTKY

- HIGH SPEED
- LOW POWER
- COMMON CLOCK AND COMMON RESET
- CLAMP DIODES FOR HIGH-SPEED TERMINATION
- ESD > 3500 V

PIN NAMES

		LOADING (Note a) HIGH	LOW
CP	Clock (Active HIGH Going Edge) Input	0.5 U.L.	0.25 U.L.
D_0–D_7	Data Inputs	0.5 U.L.	0.25 U.L.
MR	Master Reset (Active LOW) Input	0.5 U.L.	0.25 U.L.
Q_0–Q_7	Register Outputs (Note b)	10 U.L.	5(2.5) U.L.

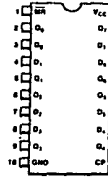
NOTES:
a. 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW
b. The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

TRUTH TABLE

MR	CP	D_n	Q_n
L	X	X	L
H	↑	H	H
H	↑	L	L

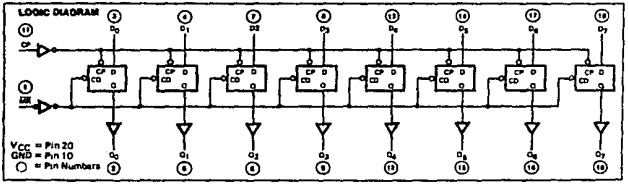
H = High Logic Level
L = Low Logic Level
X = Immaterial

CONNECTION DIAGRAM
DIP (TOP VIEW)



J Suffix — Case 732-03 (Ceramic)
N Suffix — Case 738-03 (Plastic)

LOGIC DIAGRAM



NA HIPTION — Th… …ck and cc Master …set.

Wh… he $\overline{MR}$ input is LOW, the Q outputs are … of the other inp… meeting the setup … time requirements of the D input is … outp… …sition of the clock input.

GUAR… TED OPERATING RANGES

SYMBOL	PARAMETER		MIN	TYP	MAX	UNIT
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54,74			−0.4	mA
I_{OL}	Output Current — Low	54 74			4.0 8.0	mA

DC CHARACTERISTICS OVER OPERATING …PERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.5		V	
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	I_{OL} = 4.0 mA; V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA
I_{IH}	Input HIGH Current				20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V
I_{OS}	Short Circuit Current		−20		−100	mA	V_{CC} = MAX
I_{CC}	Power Supply Current				27	mA	V_{CC} = MAX

= 5.0 V

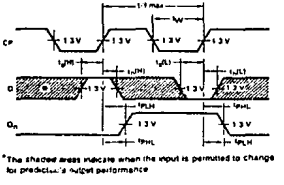
SYMBOL		MIN		MAX		
f_{MAX}	Maximum Clock Frequency	30	40		MHz	Fig. 1
t_{PHL}	Propagation Delay, Clock to Output			27	ns	Fig. 2
t_{PLH} t_{PHL}	Propagation Delay, Clock to Output		17 18	27 27	ns	Fig. 1

= 25°C, V_{CC} = 5.0 V

SYMBOL			LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS
t_w	Pulse	Clock or Clear	20			ns	Fig. 1
t_s	Data		20			ns	Fig. 1
t_h	Hold		5.0			ns	Fig. 1
t_{rec}	Rec		25			ns	Fig. 2

AC WAVEFORMS

CLOCK TO OUTPUT DELAYS, CLOCK PULSE WIDTH, FREQUENCY, SETUP AND HOLD TIMES DATA TO CLOCK



*The shaded areas indicate when the input is permitted to change for predictable output performance

Fig. 1

MASTER RESET

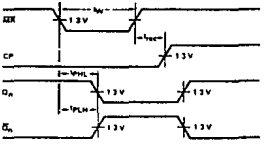


Fig. 2

DEFINITION OF TERMS:

SETUP TIME (t_s) — is defined as the minimum time required for the correct logic level to be present at the logic input prior to the clock transition from LOW-to-HIGH in order to be recognized and transferred to the outputs.

HOLD TIME (t_h) — is defined as the minimum time following the clock transition from LOW-to-HIGH that the logic level must be maintained at the input in order to ensure continued recognition. A negative HOLD TIME indicates that the correct logic level may be released prior to the clock transition from LOW-to-HIGH and still be recognized.

RECOVERY TIME (t_{rec}) — is defined as the minimum time required between the end of the reset pulse and the clock transition from LOW-to-HIGH in order to recognize and transfer HIGH data to the Q outputs.

MOTOROLA

SN54/74LS682
SN54/74LS684
SN54/74LS688

8-BIT MAGNITUDE COMPARATORS

LOW POWER SCHOTTKY

DESCRIPTION — The SN54LS 74LS682, 684, 688 are 8-bit magnitude comparators. These device types are designed to perform comparisons between two eight-bit binary or BCD words. All device types provide $\overline{P=Q}$ outputs and the LS682 and LS684 have $\overline{P>Q}$ outputs also.

The LS682, LS684 and LS688 are totem pole devices. The LS682 has a 20 kΩ pullup resistor on the Q inputs for analog or switch data.

TYPE	$\overline{P=Q}$	$\overline{P>Q}$	OUTPUT ENABLE	OUTPUT CONFIGURATION	PULLUP
LS682	yes	yes	no	totem-pole	yes
LS683	yes	yes	no	open-collector	yes
LS684	yes	yes	no	totem-pole	no
LS685	yes	yes	no	open-collector	no
LS686	yes	yes	yes	totem pole	no
LS687	yes	yes	yes	open-collector	no
LS688	yes	no	yes	totem-pole	no
LS689	yes	no	yes	open collector	no

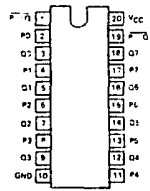
FUNCTION TABLE

INPUTS			OUTPUTS	
DATA	ENABLES			
P, Q	$\overline{G}$, $\overline{GT}$	$\overline{G2}$	$\overline{P=Q}$	$\overline{P>Q}$
P = Q	L	L	L	H
P > Q	L	L	H	L
P < Q	L	L	H	H
X	H	H	H	H

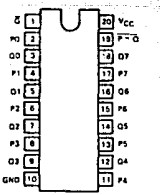
H = high level, L = low level, X = irrelevant

CONNECTION DIAGRAMS
(TOP VIEW)

SN54LS/74LS682/684

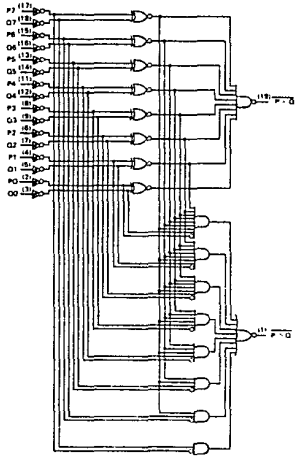


SN54LS/74LS688

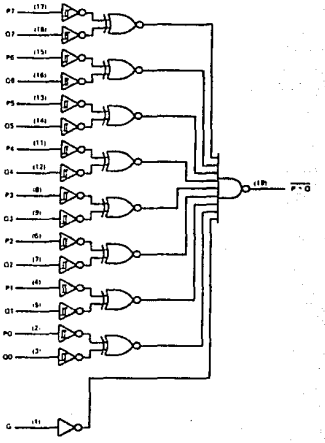


J Suffix — Case 732-03 (Ceramic)
N Suffix — Case 738-03 (Plastic)

LOGIC DIAGRAMS



SN54LS/74LS682 thru LS684



SN54LS/74LS688

GUARANTEED OPERATING RANGES

SYMBOL	PARAMETER		MIN	TYP	MAX	UNIT
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54,74			−0.4	mA
I_{OL}	Output Current — Low	54 74			12 24	mA

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

SYMBOL	PARAMETER		LIMITS MIN	LIMITS TYP	LIMITS MAX	UNITS	TEST CONDITIONS
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.5		V	
V_{OL}	Output LOW Voltage	54,74		0.25	0.4	V	I_{OL} = 12 mA, V_{CC} = V_{CC} MIN., V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 24 mA
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
		LS682-Q Inputs			0.1	mA	V_{CC} = MAX, V_{IN} = 5.5 V
		Others			0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V
I_{IL}	Input LOW Current	LS682-Q Inputs			−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V
		Others			−0.2	mA	
I_{OS}	Short Circuit Current		−30		−130	mA	V_{CC} = MAX
I_{CC}	Power Supply Current	LS682			70	mA	V_{CC} = MAX
		LS684			65	mA	
		LS688			65	mA	

SN54/74LS682 • SN54/74LS684 • SN54/74LS688

AC CHARACTERISTICS: $T_A = 25°C$

SN54LS/74LS682

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P = Q}$		13 15	25 25	ns	$V_{CC} = 5.0\ V$ $C_L = 45\ pF$ $R_L = 667\ \Omega$
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P = Q}$		14 15	25 25	ns	
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P > Q}$		20 15	30 30	ns	
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P > Q}$		21 19	30 30	ns	

SN54LS/74LS684

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P = Q}$		15 17	25 25	ns	$V_{CC} = 5.0\ V$ $C_L = 45\ pF$ $R_L = 667\ \Omega$
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P = Q}$		16 16	25 25	ns	
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P > Q}$		22 17	30 30	ns	
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P > Q}$		24 20	30 30	ns	

SN54LS/74LS688

SYMBOL	PARAMETER	LIMITS			UNITS	TEST CONDITIONS
		MIN	TYP	MAX		
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P = Q}$		12 17	18 23	ns	$V_{CC} = 5.0\ V$ $C_L = 45\ pF$ $R_L = 667\ \Omega$
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P = Q}$		12 17	18 23	ns	
t_{PLH} t_{PHL}	Propagation Delay, $\overline{G}$, $\overline{G1}$ to $\overline{P = Q}$		12 13	18 20	ns	

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MCM2016A

Fast 16K Bit Static RAM

N PACKAGE
PLASTIC
CASE 724

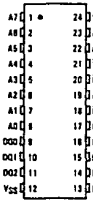
The MCM2016A is a 16,384 bit static random access memory organized as 2048 words by 8 bits, fabricated using Motorola's high-performance silicon-gate MOS (HMOS) technology. It uses an innovative design approach which combines the ease-of-use features of fully static operation (no external clocks or timing strobes required) with the reduced standby power dissipation associated with clocked memories. To the user this means low standby power dissipation without the need for address setup and hold times, nor reduced data rates due to cycle times that are longer than access times. Perfect for cache and sub-100 ns buffer memory systems, this high speed static RAM is intended for applications that demand superior performance and reliability.

Chip enable ($\overline{E}$) controls the power-down feature. It is not a clock but rather a chip control that affects power consumption. In less than a cycle time after $\overline{E}$ goes high, the part automatically reduces its power requirements and remains in this low-power standby mode as long as $\overline{E}$ remains high. This feature provides significant system-level power savings.

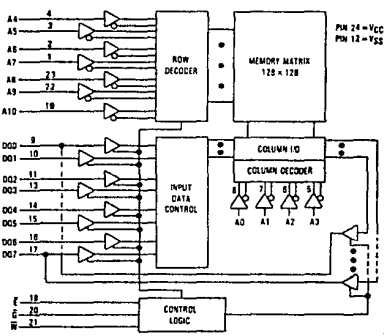
The MCM2018A is in a 24-pin dual-in-line 300 mil wide package with the industry standard JEDEC approved pinout.

- Single +5 V Operation, ±10%
- Fully Static: No Clock or Timing Strobe Required
- Fast Access Time: MCM2016A-35 = 35 ns (Maximum)
  MCM2016A-45 = 45 ns (Maximum)
- Power Supply Current: 135 mA Maximum (Active)
  20 mA Maximum (Standby)
- Three-State Output

PIN ASSIGNMENT



BLOCK DIAGRAM



PIN NAMES

A0-A10	Address Input
DQ0-DQ7	Data Input/Output
$\overline{W}$	Write Enable
$\overline{G}$	Output Enable
$\overline{E}$	Chip Enable
V_{CC}	+5 V Power Supply
V_{SS}	Ground

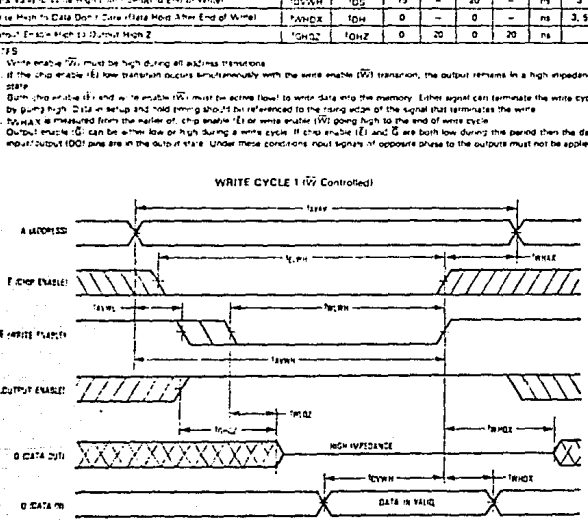
WRITE CYCLE (See Notes 1 and 2)

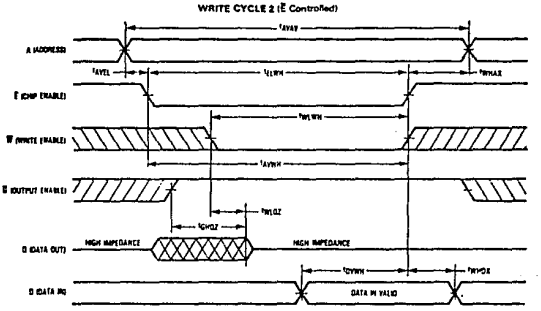
Parameter	Symbol		MCM2016A-35		MCM2016A-45		Units	Notes
	Standard	Alternate	Min	Max	Min	Max		
Address Valid to Address Valid (Write Cycle Time)	t_{AVAV}	t_{WC}	35	–	45	–	ns	
Chip Enable Low to Write High (Chip Enable to End of Write)	t_{ELWH}	t_{EW}	30	–	40	–	ns	
Address Valid to Chip Enable Low (Address Setup to Chip Enable)	t_{AVEL}	t_{AS}	0	–	0	–	ns	
Address Valid to Write Low (Address Setup to Write)	t_{AVWL}	t_{AS}	0	–	0	–	ns	
Address Valid to Write High	t_{AVWH}	t_{AW}	30	–	40	–	ns	3
Write Low to Write High (Write Pulse Width)	t_{WLWH}	t_{WP}	30	–	35	–	ns	
Write High to Address Don't Care (Address Hold After End of Write)	t_{WHAX}	t_{WR}	0	–	0	–	ns	4
Write High to Output Don't Care (Output Active After End of Write)	t_{WHQX}	t_{WLZ}	0	–	0	–	ns	5
Write Low to Output High Z (Write to Enable to Output Disable)	t_{WLQZ}	t_{WHZ}	0	20	0	20	ns	5
Data Valid to Write High (Data Setup to End of Write)	t_{DVWH}	t_{DS}	15	–	20	–	ns	3
Write High to Data Don't Care (Data Hold After End of Write)	t_{WHDX}	t_{DH}	0	–	0	–	ns	3, 5
Output Enable High to Output High Z	t_{GHQZ}	t_{OHZ}	0	20	0	20	ns	

NOTES

1. Write enable ($\overline{W}$) must be high during all address transitions.
2. If the chip enable ($\overline{E}$) low transition occurs simultaneously with the write enable ($\overline{W}$) transition, the output remains in a high impedance state.
3. Both chip enable ($\overline{E}$) and write enable ($\overline{W}$) must be active (low) to write data into the memory. Either signal can terminate the write cycle by going high. Data in setup and hold timing should be referenced to the rising edge of the signal that terminates the write.
4. t_{WHAX} is measured from the earlier of chip enable ($\overline{E}$) or write enable ($\overline{W}$) going high to the end of write cycle.
5. Output enable ($\overline{G}$) can be either low or high during a write cycle. If chip enable ($\overline{E}$) and $\overline{G}$ are both low during this period then the data input/output (DQ) pins are in the output state. Under these conditions input signals of opposite phase to the outputs must not be applied.

WRITE CYCLE 1 ($\overline{W}$ Controlled)





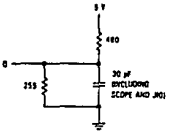
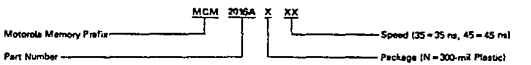


Figure 1. Output Load

ORDERING INFORMATION
(Order by Full Part Number)



Full Part Numbers – MCM2016AN35
MCM2016AN45

MODE SELECTION

Mode	$\overline{E}$	$\overline{G}$	$\overline{W}$	V_{CC} Current	DQ
Standby	H	X	X	I_{SB}	High Z
Read	L	I	H	I_{CC}	Q
Write Cycle	L	x	L	I_{CC}	D

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal p .cautions be taken to avoid application of any voltage higher than maximum rated voltages to tha high-impedance circuit.

ABSOLUTE MAXIMUM RATINGS (See Note)

Rating	Symbol	Value	Unit
Power Supply Voltage	V_{CC}	−0.5 to +7.0	V
Voltage on Any Pin With Respect to V_{SS}	V_{in}, V_{out}	−0.5 to +7.0	V
DC Output Current	I_{out}	±20	mA
Power Dissipation	P_D	1.1	Watt
Temperature Under Bias	T_{bias}	−10 to +80	°C
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−65 to +150	°C

NOTE: Permanent device damage may occur if ABSOLUTE MAXIMUM RATINGS are exceeded. Functional operation should be restricted to RECOMMENDED OPERATING CONDITIONS. Exposure to higher than recommended voltages for extended periods of time could affect device reliability.

DC OPERATING CONDITIONS AND CHARACTERISTICS

(V_{CC} = 5.0 V ± 10%, T_A = 0 to 70°C, Unless Otherwise Noted)

RECOMMENDED OPERATING CONDITIONS

Parameter	Symbol	Min	Typ	Max	Unit
Supply Voltage (Operating Voltage Range)	V_{CC}	4.5	5.0	5.5	V
	V_{SS}	0	0	0	V
Input Voltage	V_{IH}	2.0	3.0	6.0	V
	V_{IL}	−0.5*	0	0.8	V

*The device will withstand undershoots to the −2.5 volt level with a maximum pulse width of 50 ns. This is periodically sampled rather than 100% tested.

DC CHARACTERISTICS

Parameter	Symbol	Min	Max	Unit
Input Leakage Current (V_{CC} = 5.5 V, V_{in} = GND to V_{CC})	$I_{lkg(I)}$	−1.0	1.0	µA
Output Leakage Current ($\overline{E}$ = V_{IH} or $\overline{G}$ = V_{IH}, $V_{I/O}$ = GND to V_{CC})	$I_{lkg(O)}$	−1.0	1.0	µA
Operating Power Supply Current ($\overline{E}$ = V_{IL}, $I_{I/O}$ = 0 mA)	I_{CC}	—	135	mA
Standby Power Supply Current ($\overline{E}$ = V_{IH})	I_{SB}	—	20	mA
Output Low Voltage (I_{OL} = 8.0 mA)	V_{OL}	—	0.4	V
Output High Voltage (I_{OH} = −4.0 mA)	V_{OH}	2.4	—	V

CAPACITANCE (f = 1.0 MHz, T_A = 25°C, Periodically Sampled Rather Than 100% Tested)

Characteristic		Symbol	Typ	Max	Unit
Input Capacitance	All Inputs Except $\overline{E}$ and DQ	C_{in}	3	5	pF
	$\overline{E}$		5	7	
I/O Capacitance	DQ	$C_{I/O}$	5	7	pF

AC OPERATING CONDITIONS AND CHARACTERISTICS

(V_{CC} = 5 V ± 10%, T_A = 0 to +70°C, Unless Otherwise Noted)

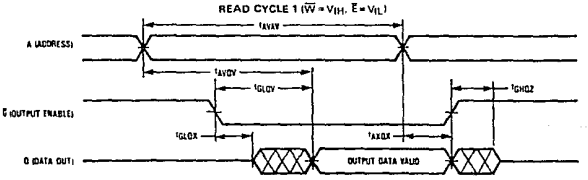
Input Pulse Levels . . . . . 0 and 3.0 V
Input Rise and Fall Times . . . . . 5 ns
Input and Output Timing Measurement Reference Levels . . . 1.5 V
Output Load . . . . . See Figure 1

READ CYCLE (See Note 1)

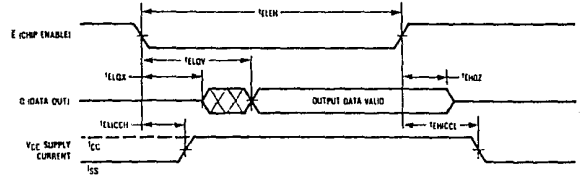
Parameter	Symbol		MCM2016A-35		MCM2016A-45		Units	Notes
	Standard	Alternate	Min	Max	Min	Max		
Address Valid to Address Valid (Read Cycle Time)	t_{AVAV}	t_{RC}	35	–	45	–	ns	
Address Valid to Output Valid (Address Access Time)	t_{AVQV}	t_{AC}	–	35	–	45	ns	
Chip Enable Low to Chip Enable High (Read Cycle Time)	t_{ELEH}	t_{RC}	35	–	45	–	ns	
Chip Enable Low to Output Valid (Chip Enable Access Time)	t_{ELQV}	t_{ACS}	–	35	–	45	ns	
Output Enable Low to Output Valid (Output Enable Access Time)	t_{GLQV}	t_{OE}	–	20	–	20	ns	
Chip Enable Low to Output Invalid (Chip Enable to Output Active)	t_{ELQX}	t_{CLZ}	5	–	5	–	ns	2
Chip Enable High to Output High Z (Chip Disable to Output Disable)	t_{EHQZ}	t_{CHZ}	0	20	0	20	ns	2
Output Enable Low to Output Invalid (Output Enable to Output Active)	t_{GLQX}	t_{OLZ}	0	–	0	–	ns	2
Output Enable High to Output High Z (Output Disable to Output Disable)	t_{GHQZ}	t_{OHZ}	0	20	0	20	ns	2
Address Invalid to Output Invalid (Output Hold Time)	t_{AXQX}	t_{OH}	5	–	5	–	ns	
Chip Enable Low to Power Up	t_{ELICCH}	t_{PU}	0	–	0	–	ns	
Chip Enable High to Power Down	t_{EHICCL}	t_{PD}	–	20	–	20	ns	

NOTES:
1. Transition time specification applies for all input signals. In addition to meeting the transition rate specification, all input signals must transition between V_{IL} and V_{IH} (or between V_{IH} and V_{IL}) in a monotonic manner.
2. Transition is measured ± 200 mV from the steady state output voltage with the output loading specified in Figure 1.
3. In read cycle 2, all addresses are valid prior to or coincident with chip enable ($\overline{E}$) transition low.

READ CYCLE 1 ($\overline{W}$ = V_{IH}, $\overline{E}$ = V_{IL})



READ CYCLE 2 ($\overline{W}$ = V_{IH}, $\overline{G}$ = V_{IL}; See Note 3 Above)



Bibliografía

Anderson Charles,
The Visible Computer: 8088,
Software Masters, Inc.,
Texas, 1985.

Borland International,
Turbo Pascal 3.3, User Reference Manual,
Scotts Valley, CA, USA. 1983, 1984, 1985.

Brenner C. Robert
IBM-PC Troubleshooting & Repair Guide,
Howard W. Sams & Company
Indianapolis, Indiana 1989.

Eggebrecht Lewis C.
Interfacing to the IBM Personal Computer,
Howard W. Sams & Company.
Indianapolis, Indiana USA.

Hall V. Douglas,
Microprocessors and Interfacing,
McGraw-Hill International Editions
Computer Science Series
Singapore, 1986.

Hennefeld Julien
Using Turbo Pascal 3.0, 4.0 and 5.0
PWS Publishers,
USA, 1989.

International Business Machines Corp.,
Personal Computer Technical Reference Manual.
USA, 1981.

Jamsa Kris / Nameroff Steven
Turbo Pascal, Biblioteca de Programas,
McGraw-Hill,
España, 1989.

Mick John / Brick James
Bit-Slice Microprocessor Design,
McGraw-Hill Book Company,
1980.

Motorola Incorporation
Fast and LS TTL Data
Revision 4
Evans Press
USA 1989.

Motorola Incorporation
Motorola Memory Data
Revision 4
Evans Press
USA 1989.

Nath, Sanjiva
Assembly Languaje Interfacing in Turbo Pascal,
Management Information Source, Inc., MIS : PRESS,
Portland, Oregon, 1987.

National Semiconductor Corporation
LS/S/TTL Logic Databook.
Santa Clara, California, USA, 1987.

Norton Peter
Inside the IBM PC: Access to Advanced Features and Programming,
Robert J. Brandy Company.

Philippot P.
Turbo Pascal para IBM-PC y Compatibles,
Procedimientos y Funciones,
Gustavo Gili, S. A.
Barcelona, 1987.

Schildt Herbert
Programación y Técnicas. Turbo Pascal Avanzado.
McGraw-Hill,
México, 1989.

Singh Avtar / Triebel Walter A.
The 8088 microprocessor : Programming, Interfacing, Software, and Applications
New Jersey, USA, 1989.

Texas Instruments Inc.
The TTL Data Book for Design engineers,
1981.

Thorne Michael
Programming the 8086/8088. For the IBM PC and compatibles
Addison - Wesley Publishing Company Inc.
1987.

Tocci Ronald J.
Digital Systems / Principles and Applications,
Prentice Hall, Inc.

Tood Gail
Using WordPerfect Series 5 Edition
McGraw-Hill Inc.
1988.

Wiatrowski Claude A. y House Charles H.
Circuitos Lógicos y Sistemas de Microcomputadoras,
Limusa.
México, 1987.

Wood Steve

Turbo Pascal Version 3.0

McGraw-Hill

México, 1989.

Unisys

Product Interconnect and Wiring Guide

Unisys, 1989.

U S America.