UNIVERSIDAD AUTONOMA DE GUADALAJARA

INCORPORADA A LA UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA DE INGENIERIA

Manual de Laboratorio para el Microprocesador 6800

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

SERGIO ARTURO ORTIZ DEL HOYO

GUADALAJARA, JAL., 1985

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

INTRODUCCION

Este manual tiene como propósito introducir al estudiante de ingeniería, principalmente ingeniería electrónica y en computación en el campo de la programación de microprocesadores, así como también una orientación al laboratorio a fin de integrar mejor el proceso educativo en las partes técnica y teórica.

En los programas de educación técnica o a nivel licenciatura, la enseñanza práctica debería ser una parte vital en la preparación de un ingeniero, puesto que el laboratorio debe estimular al estudiante para que razone y analice, por lo que determinará en gran medida la cadencia y profundidad de la enseñanza teórica.

Bajo esta idea se intenta preparar un manual para incrementar la importancia del laboratorio. Una oritentación para el laboratorio y los estudiantes , creando un programa para el laboratorio y su mejor administración, así como también conferir al estudiante una amplia gama de experiencias en técnicas, procesos, instrumentos y equipos; presentar aplicaciones de la teória básica; inculcar la confianza en sí mismo y desarrollar la capacidad de trabajo en equipo.

Esta tesis comprende de nueve capítulos, el primero consiste en una breve historia de la tecnología y el porque de la elección del microprocesador MC6800. El segundo capítulo es una breve descripción de caracteristicas deseadas en un laboratorio refiriendose a el personal y al material. El tercero presentará una vista general de las áreas de trabajo y las caracteristicas con las que debe contar el personal a cargo de estas áreas. El cuarto capítulo introduce al estudiante al microprocersador MC6800 en sus configuraciones mas sencillas e interrelación con componentes básicos para formar un microcomputador. El Quinto incluye las primeras 9 prácticas para la asimilación de algunas técnicas de programación desde los principios básicos. El Sexto presentará un nivel mas elevado de programación donde el estudiante mostrará su habilidad para desarrollar programas en microprocesadores, además de mostrar su conocimiento de los microprocesadores. El Séptimo se compone de 5 prácticas, las cuales daran al estudiante idea de toda la gama de aplicaciones que puede tener un microprocesador. Las conclusiones a la tesis son expuestas en el capítulo 8. Y por último el capítulo 9 presenta un apendice.

I.- CONCEPTOS GENERALES

I.- CONCEPTOS GENERALES

INTRODUCCION

Este capítulo consiste en una breve historia de la tecnología en las computadoras hasta la epoca actual. Se expondrán varios microprocesadores y el porque se ha elegido el microprocesador MC6800 para el desarrollo a través de las prácticas de esta tesis.

HISTORIA

Aunque las computadoras son de reciente inovación, su desarrollo viene de siglos de investigación, y descubrimientos. Avances en la tecnología de procesamiento de información son respuestas a la necesidad de encontrar metodos mejores, más rápidos, más baratos y más confiables de manejar datos. Entender la evolución del procesamiento de datos es de gran ayuda para entender las capacidades y limitaciones de computadoras modernas.

El inicio más realista podría ser con el desarrollo del ABACO, aunque se atribuye la creación a los chinos, realmente emergió independientemente en varias culturas.

Descubrimientos matemáticos son también parte de la historia, Napier puso los logaritmos de los números en un conjunto de rodillos de marfil (apodados "huesos de Napier". Y deslizando uno sobre otro se podía sumar y restar los números en series geométricas desarrollando multiplicaciones y divisiones.

Máquinas que desarrollaron cálculos aritméticos fueron desarrolladas en los principios del siglo 17 por Pascal y Leibnitz. La primera máquina que empleó los conceptos similares a los de una computadora fué la máquina analítica diseñada por Charles Babbage en 1803, conocido como "El Padre de las Computadoras". Esta máquina fracasó debido a que su producción estaba más alla de la tecnología de aquellos tiempos.

El uso de máquinas calculadoras electromecánicas fué implementado por primera vez por Herman Hollerit en el censo de los Estados Unidos en 1890. Estas máquinas usaban tarjetas perforadas como entrada, y desarrollaban cálculos aritméticos sencillos y ordenamiento de tarjetas. Máquinas electromecánicas de tarjetas perforadas fueron usadas intensivamente hasta mediados de 1900. Eran controladas por Panels de control manejados con alambres.

La MARK I fué la primera calculadora automática. Introducida a finales de los años 30, la MARK I usaba relevadores electromagnéticos y contadores para desarrollar operaciones. Era electromecánica, y no electrónica.

CONCEPTOS GENERALES

A mediados de los 40's, La ENIAC (del inglés "Electronic Numerical Integrator And Calculator") fue desarrollada. La ENIAC fue la primera computadora digital electrónica. Las funciones de control e interruptores fueron desarrolladas usando tubos al vacio (bulbos). La EDSAC (del inglés "Electronic Delay-Storage Automatic Computer) fué terminada en 1949, fué la primera computadora que podía almacenar programas.

La primera genereración de computadoras (1951-1958) empezó con la introducción de la UNIVACI (del inglés "Universal Automatic Computer"). La primera generación usó bulbos para el control de sus operaciones internas. Estas máquinas eran muy grandes y generaban mucho calor. Estas máquinas eran muy rápidas para las máquinas anteriores, pero demasiado lentas para estos tiempos.

La segunda generación de computadoras (1959-1964) se basó en transistores para controlar sus operaciones internas. Los transistores son mucho más pequeños, más rápidos y más confiables que los bulbos. Además, aumentos significantes en velocidad fueron obtenidos al usar núcleos magnéticos como almacenamiento interno, o memoria. Otra importante inovación durante la segunda generación fué la introducción de lenguajes de alto nivel, diseño de hardware modulares, y mejoras en componentes de entrada/salida.

La tercera generación (1965-1971) usó circuitos integrados de estado sólido mejorando el tamaño y costo de los transistores, aumentando además velocidad y confiabilidad. Estas máquinas tenían mayor capacidad de almacenamiento que las de segunda generación. Programación más sofisticada dió más fuerza a las computadoras de esta generación. Muchas operaciones que anteriormente eran manejadas manualmente se automatizaron.

La Cuarta generación se basa en integración a gran escala y continua mejorando los costos, el tamaño y la velocidad. El uso de microprocesadores ha proliferado y siendo que el costo de las computadoras está disminuyendo, mientras que la mano de obra aumenta, tener más aplicaciones con computadoras es inevitable.

MICROPROCESADORES COMERCIALES

** 6502	uprocesador de 8-bits
** 8080	uprocerador de 8-bits
** Z90	uprocesador de 8-bits
** 6800	uprocesador de 8-bits
** 8086.	uprocesador de 16-bits

Estos son algunos de los microprocesadores comerciales de mayor uso en el mercado. Tienen diferentes características, pero todos bajo un mismo principio y mostrando algunos mayor versatilidad y flexibilidad, así como rapidez, confiabilidad y costo. El microprocesador MC6800 se ha elegido en muchas escuelas por su facilidad de manejo, es un microprocesador que tiene gran versatilidad, puede aprender a usarse con gran rapidez, y además es de bajo costo.

II.- FORMULACION DEL PROGRAMA DE LABORATORIO

II.- FORMULACION DEL PROGRAMA DE LABORATORIO

INTRODUCCION

La intención de este capítulo es dar una breve descripción del personal y material básico requerido para el desarrollo satisfactorio de las prácticas que en capítulos posteriores se indican, así como también una guía que indicará la secuencia que deberá seguir la revisión y evaluación de cada práctica.

SELECCION DEL PERSONAL

La selección del personal se lleva a cabo por el instructor y por el director. En esa selección se efectuan exámenes psicométricos, de conocimiento y entrevistas personales dirigidas por los anteriormente citados, en las que se evaluará la habilidad para desarrollar un trabajo eficaz en los puestos que sean requeridos por el laboratorio.

En el siguiente capítulo haré una descripción breve de las características, por área de trabajo, con las que debe de contar el personal de laboratorio. Por ahora expondré los puntos generales para la elección de un buen personal.

Las evaluaciones realizadas serán revisadas por el departamento de personal de la escuela. Estas evaluaciones deberán mostrar características de la persona a entrevistar comos son: capacidad de organización, conocimientos administrativos y técnicos, deseos de superación y un pensamiento ético profesional para mantener la calidad en todos los deberes que a el correspondan.

SELECCION DEL MATERIAL

La selección del material requerido para la implementación de un buen laboratorio deberá hacerse tomando en cuenta todas las opciones, considerando la efectividad y la economía del equipo o material del laboratorio.

El equipo de laboratorio requerido sugerido es como sigue:

* EQUIPO DE HERRAMIENTAS

-- Taladro y accesorios.
-- Pistola para soldar y accesorios
-- Serrucho y accesorios
-- Pinzas (de presión, de punta, de corte, etc...)
-- Desarmadores (de todos tipos)
-- Martillos
-- Llaves españolas (estandar y milimétricas)
-- Llaves de estrias (estandar y milimétricas)
-- LLaves allen (estandar y milimétricas)
-- Matraca y dados
-- Limas y lijas

* EQUIPO MISCELANEO

-- Tornillos, clavos y tuercas
-- Baterías
-- Cables (de distintos calibres)
-- Extensión eléctricas
-- Focos y lámparas
-- Cintas generales y de Aislamiento
-- Guantes
-- Caimanes

* EQUIPO DE PRUEBA Y MEDICION

  -- Microcomputadora
  -- Osciloscopio
  -- Multímetro
  -- Analizador digital
  -- Fuentes de poder (varios voltajes)
  -- Fuentes de corriente (varias capacidades)
  -- Generador de Onda (oscilador)
  -- Frecuencímetro
  -- Cinta métrica
  -- Banmier
  -- Impresora y equipo periférico
     (diskettes, modems, disco duro, etc...)

* EQUIPO PARA DESARROLLO
  -- Resistencias
  -- Capacitores
  -- Bobinas
  -- Diodos
  -- Transistores
  -- Ciruitos integrados
  -- Amplificadores operacionales
  -- Microprocesadores
  -- Circuitos impresos
  -- Disipadores de potencia
  -- Osciladores
  -- LED's (diodos emisores de luz)
  -- Displays (pantallas, LCD's)
  -- Motores (AC y DC)
  -- Generadores
  -- Alternadores

Nótese que la descripción de equipo ha sido general y no hemos llegado al detalle de establecer características eléctricas o físicas de algun elemento. Por ejemplo: Circuito integrado 7400 compuerta "AND", de 14 pins y voltaje de alimentación Vcc de +5V; o capacitor Axial monolítico de 300 picofaradios con tolerancia del 10%.

La relación de fuentes donde se puede obtener información deberá ser un directorio que será disponible a todo el alumnado; este directorio contendrá información que podrá ser catalogada como sigue: Por tema, por materia, por autor, por indice alfabético. Este directorio puede ser desarrollado por el instructor del laboratorio.

Para propósitos de este manual de prácticas, existe la bibliografía de la información que puede servir como guía para resolver las prácticas que aquí se expondrán. Además podrás obtener mayor información que te dará una visión más amplia del trabajo de microprocesadores en libros de electrónica, donde se especifique el funcionamiento electrónico de circuitos integrados, transistores, amplificadores operacionales, etc...

La secuencia que se suguiere para la revisión y evaluación de cada práctica es como sigue:

A.- Se revisa el cuestionario previo de la práctica, el cual deberá ser entregado antes de iniciar la práctica, tomando en cuenta la puntuación que en cada práctica se establece y dejando a criterio del instructor el otorgar o no puntuación por un reporte bien organizado y limpio.

B.- Evaluación del proceso práctico. El instructor tomará notas del trabajo realizado por el alumno y revisará que las operaciones que deben hacerse en la practica sean realizadas satisfactoriamente. Al final de cada práctica se establece la puntuación que se otorgará por este concepto.

C.- Evaluación del reporte de la práctica: Se calificará por un reporte bien organizado, limpio y con una documentación que ponga en claro el completo entendimiento de la práctica. No se requiere de un libro de texto para explicar una situación y así es como debe documentarse cada reporte, breve y conciso .

III.- ADMINISTRACION DEL LABORATORIO

III.- ADMINISTRACION DEL LABORATORIO

INTRODUCCION

Así como en el capítulo anterior se desarrolla una guía al jefe de laboratorio, aquí se presentarán algunas notas de importancia para el mejor desarrollo del laboratorio.

Este documento es de importancia en una organización ya que lleva un procedimiento administrativo y técnico que indica la estructura del laboratorio, así como el fin que persigue, y la forma que se ha establecido para llegar a lograr ese objetivo.

Con este trabajo se pretende dejar una constancia por escrito de los lineamientos bajo los cuales se pretende que trabaje el laboratorio de manera que si existiera alguna duda, tener en donde consultar y quizas ayudar a resolver los problemas que ahí surjan y lograr una mejor captación de los deberes que requiere el puesto, así como responsibilidades de cada miembro que forma parte del laboratorio. De esta manera se pretende tener un mejor control de actividades y requerimientos de trabajo.

* CONTENIDO DEL PROGRAMA

El programa consiste en definir lo mejor posible las actividades que son realizadas en el laboratorio y principalmente definir los puntos en los cuales repercuta directamente o indirectamente al mejor aprovechamiento del laboratorio.

El objetivo del laboratorio es dar al alumno la habilidad para investigación y desarrollo de esos conceptos teóricos que le han sido impartidos en el salón de clases, para cuando este se enfrente ha problemas cotidianos en su vida profesional tenga ya los conceptos prácticos (adquiridos en el laboratorio) para resolverlos.

Es importante definir las áreas de trabajo de un laboratorio, esto es:

A.- Area de Asesoria.- Consiste en una especie de aula con las características de esta (sillas, escritorios, pizarrón, equipo audiovisual, etc...). El objetivo es presentar un resumen de los conceptos teóricos básicos necesarios para la realización de la práctica y al mismo tiempo aclarar cualquier duda al respecto. Además, presentar un sumario de las actividades que se desarrollarán en el proceso de la práctica. Es posible que durante la práctica surja algún evento relevante que sea de interés para el grupo, por lo que el instructor de laboratorio optaría por suspender temporalmente la sesión para hacer del conocimiento del alumnado el suceso y las características que este representa, llevandose a cabo en esta área de asesoría.

B. Area de Investigacion.- LLamaremos de esta manera a esa área donde se localiza la bibliografía necesaria para el desarrollo de la práctica. Aquí encontraremos todo lo referente a :

B1. Manuales para uso del equipo de laboratorio . Osciloscopios, multímetros, fuentes de poder, computadoras, etc...

B2. Manuales y libros que describan a detalle características de componentes. Resistencias, capacitores, diodos, transistores, circuitos integrados, amplificadores operacionales, microprocesadores, etc...

B3. Manuales para el uso de herramientas. Taladros, pistolas de soldar, bernier, etc...

B4. Equipo audiovisual para captar aquellas gráficas o circuitos que estén presentados en microfilm o aquel que desee tener una amplificación para poder apreciar a detalle aquellos dibujos que pudieran causar cierta confusión.

B5. Un directorio bibliográfico para hacer referencia a cierto tema de interés para el alumno y a través de este directorio se pueda encontrar material de estudio. (ejemplo: hacer referencia a libros de la biblioteca central de la universidad).

B6. En laboratorios modernos se cuenta con terminales conectadas a bancos de datos de donde se puede obtener información de proyectos. Proyectos en los que se este trabajando relacionados de acuerdo al fin que el proyecto persigue, o al procedimiento que se esté llevando a cabo o aún más a detalle la utilización de ciertos componentes en el proyecto.

Este banco de datos puede crearse internamente en la universidad, a nivel estatal, a nivel nacional y quizás en un futuro a nivel mundial.

Las ventajas que presenta este banco de datos son inumenrables, entre las más importantes puedo mencionar la de evitar trabajar en un proyecto con ciertos componentes que hayan fracasado al haberlo realizado una persona y que esta expuso en este banco de datos las características del proyecto y la razón por la cual se fracasó. Otra sería evitar que dos personas trabajen en un mismo proyecto, desperdiciando de esa manera recursos. La respuesta al encontrar el sistema de computación 2 proyectos similares sería poner en contacto a esas dos personas y coordinarlas manteniendo informados a ambos de lo ocurrido y que

trabajaran paralelamente y así obtener mejores resultados en los proyectos.

Por supuesto la confidencialidad de los proyectos quedaría restringida (si así se deseara) a los propietarios del proyecto en cuestión, siendo el sistema el único capaz de permitir acceso a ciertos archivos clasificados y/o restringidos. La clasificación del proyecto está en manos del propietario.

B7. Manuales de seguridad.

Esta área es muy delicada y requiere de una atención esmerada para tener el control del material aquí archivado, quizá sea necesario clasificar esa información para permitir o no al alumno sacar cierta información del área en cuestión.

C. Area de desarrollo.- Esta área puede estar dividida en dos secciones: Area de desarrollo para material pesado y para material delicado.

C1. Area de desarrollo para material pesado es el área en la cual se desarrollaran trabajos como el taladrar, cortar, soldar, etc..., en fin, todo ese trabajo que requiere de herramientas mecánicas que de una manera u otra provocan contaminación (ruido y basura) por lo que se requiere un área especial; Además de que la seguridad en esta área es con respecto a trabajos mecánicos en los que se presta especial atención al cuidado y/o protección de los ojos, extremidades, etc...

C2. Area de desarollo para material delicado es el área donde se protege a todos los componentes electrónicos y se les da un trato especial a los circuitos integrados. Por ejemplo: la superficie de trabajo debe estar perfectamente aislada y protegida contra cualquier efecto magnético que pudiera ocasionar algún defecto en cualquier circuito y aún en algun medio magnético (cinta, diskettes, etc...) en esta área la seguridad personal es importante y se recomienda no usar nada en las manos y brazos (anillos, reloj, pulseras, etc..) mientras se esté trabajando es recomendable además tener disponible pulseras aterrizadas (aislantes) para evitar cualquier descarga eléctrica.

En los manuales de seguridad se especificarán las precauciones para el manejo de circuitos integrados delicados que pudieran ser afectados aún con la estática del cuerpo humano.

D. Area de abastecimiento de materiales. Este almacen deberá estar perfectamente organizado para tener control del material que entra y sale, sean herramientas, componentes o equipo de prueba. Los componentes deberán estar clasificados de acuerdo a su grado de sensibilidad al trato (como material magnético), al porcentaje de uso y al costo del componente.

Clasificar el equipo de prueba y tener una lista de las personas (alumnos) autorizados y/o capaces de operar el equipo para así evitar descomposturas debido al mal uso que se le pudiera dar al equipo. Una persona designada por el instructor deberá tener un control de inventario de este almacén indicando principalmente el porcentaje de uso de componentes y herramientas así como componentes y herramientas en mal estado y generar un reporte para la adquisión de material teniendo antes que ser aprovada esta requisición por el instructor de laboratorio.

Personal del laboratorio será el único autorizado a entrar a esta área bajo un registro de entrada y salida.

Un mensaje con la leyenda "Prohibida la entrada a personal No autorizado" deberá ser colocado a la entrada de esta área.

E. Area de pruebas.- Equipo de prueba con todas las fuentes de suministro requeridas serán colocadas aquí. (p.e.: Fuentes de voltaje, fuentes de corriente de diferentes características).

Equipo básico de prueba: Osciloscopio, multímetro, generadores de onda, analizadores digitales.

Equipo Opcional: Microcomputador.

Tener un microcomputador como equipo de prueba puede ser en conjunto con el equipo básico el laboratorio ideal. Analizando las ventajas podemos observar que contar con un microcomputador se puede programar de tal manera que creamos infinidad de situaciones para probar un proyecto, almacenar en archivos del propio computador todas las lecturas a pruebas obtenidas por el equipo de prueba básico, y más tarde con otros programas tomar esos datos y obtener el porcentaje de efectividad, confiabilidad, y repetividad del proyecto.

F. Area General.- Todas las áreas deberán contar con buena iluminación, ventilación y equipo de seguridad (extinguidores, señales de precaución, hidrantes, etc...) así como salidas de emergencia y alarmas visibles y audibles de emergencia.

Es responsabilidad del instructor del laboratorio dar a conocer estas áreas a todo el alumnado antes de iniciar la primera práctica.

Tener una persona adecuada para manejar el laboratorio y sus áreas es un punto demasiado delicado en el que rara vez se puede contar con un personal que cubra con todos los aspectos con las necesidades del laboratorio.

Enunciaré algunas características deseables de un personal de laboratorio, de acuerdo a las áreas descritas anteriormente.

Area de Asesoría: En la mayoría de los casos y dependiendo de las circunstancias es deseable que el profesor del curso sea el responsable de esta área. Al responsable de esta área lo llamaremos "INSTRUCTOR DEL LABORATORIO". Entenderemos que el instructor tendrá un nivel educativo suficientemente elevado como para poder resolver cualquier duda que surja durante las prácticas y se recomienda que asesore a los responsables de las otras áreas.

Area de Investigación: Una persona con buenos conocimientos técnicos y administrativos. Habilidad administrativa para organizar y desarrollar el metodo de obtención de material (préstamo y recuperación de publicaciones). Esta persona será responsable de facilitar el equipo necesario para que el alumno investigue y obtenga la información deseada. Sería de gran ayuda una persona con conocimientos técnicos ya que el mismo podría reconocer con mayor facilidad el tipo de información que el alumno esté requiriendo y proporcionarle así el material suficiente y no caer en la desventaja de no saber donde encontrarla.

Esta persona notificará al instructor del laboratorio y al director del mismo el estado de está área considerando el desgaste o perdida de material, así como requisiciones para obtención o reposición del material. El responsable deberá tener pleno conocimiento de los sistemas que son manejados por el computador de esta área.

Area de Abastecimiento de Materiales: Un técnico en máquinas y herramientas sería la persona responsable, además de surtir el equipo tendría la responsabilidad de darle mantenimiento al mismo, siempre procurando tenerlo disponible para ejecutar las prácticas. Es muy común el desgaste en máquinas mecánicas por lo que se deberá tener un perfecto control en la reposición de inventarios. Existen también componentes que son muy delicados y deberán almacenarse en lugares protegidos. El responsable se encargará de verificar los dispositivos protectores de esta área. El responsable desarrollará un metodo de control de obtención, préstamo y recuperación de materiales. Deberá de informar al instructor de laboratorio y al director cualquier anomalia que pudiera ocasionar el no tener el material requerido disponible para alguna práctica y así tomar las medidas correctoras.

ADMINISTRACION DEL LABORATORIO

Area de desarrollo: El mismo instructor de laboratorio del que hablamos en el área de asesoría será el responsable de esta área. Las responsabilidades aquí serán distintas, cuidando principalmente que se cumpla con los reglamentos de seguridad y por otro lado soportar tecnicamente cualquier duda que pudiera surgir durante el proceso práctico.

Area de Pruebas: El instructor del laboratorio será el responsable de esta área y practicamente tendrá las mismas actividades que las realizadas en el área de desarrollo.

En esta área y debido al equipo que aquí se maneja, el instructor deberá serciorarse del buen uso que se haga del equipo, y evaluando en caso de daño si fué debido a un accidente o a una imprudencia; de esta manera tomar las acciones correctivas para evitar que sucesos similares vuelvan a ocurrir.

El instructor llevará el control del uso del microcomputador, organizando el horario de las sesiones.

Area General: El director del laboratorio, del cual hasta ahora no nos hemos referido, será el responsable de supervisar las actividades de cada responsable de área, como de personal indirecto (personal de mantenimiento y de seguridad).

Las actividades del director son varias, existe la actividad con el departamento administrativo de la escuela, con los que tratará los asuntos referentes a establecer los reglamentos de asistencias, horarios de trabajo, vestimenta, y en general las reglas de disciplina.

El director establecerá el organigrama de puestos y su respectiva descripción a detalle. Establecerá un diagrama de operaciones y el procedimiento a pasos de cada operación. Con ayuda del instructor del laboratorio determinará las formas de papelería con las que se llevará un control de reporte de prácticas, de actividades del personal del laboratorio, así como las formas de entrega y petición de material .

El director efectuará periodicamente cuestionarios y entrevistas con personal del laboratorio y estudiantes para evaluar el desarrollo del curso; y si fuera necesario reorganizar el laboratorio, elaborando el plan y llevando a cabo la reestructuración.

IV.- DESCRIPCION DEL HARDWARE

IV.- DESCRIPCION DEL HARDWARE

INTRODUCCION

Es necesario hacer una descripción del hardware para la comprensión del trabajo que se va a realizar a traves de estas prácticas. El microprocesador 6800 se encuentra en varios kits o arreglos; para propósitos educativos y de esta tesis se escogió el KIT-D2.

La descripción de este KIT consiste en diagramas, a bloques y electrónicos, características etécnicas y programación de los circuitos y componentes utilizados. Teniendo como principales componentes al mismo microprocesador MC6800, al PIA (Peripherial Communication Adapter), al ACIA (Asynchronous Communication Interfase Adapter), a los registros de memoria RAM (Random Access Memory) y ROM (Read Only Memory) e indicadores (Led's).

Un sistema puede ser ensamblado en su mas mínima configuración con 4 partes :

MPU ---> Microprocesador
RAM ---> 128X8
ROM ---> 1024X8 Read Only Memory
PIA ---> Peripherial Interface A.
8-bits

Como Expansión del Sistema se puede añadir el ACIA.

Una descripción básica de los componentes antes mencionados, es la siguiente (No intento hacer un manual de cada componente, por esa razón sere breve):

MC6800

MC6800: Es un microprocesador de 8-bits monolítico, forma el control central para la familia de Motorola de 6800s. Compatible con TTL, el MC6800, como todos los componentes de la familia, requiere solamente +5.0 Volts como fuente de suministro. El MC6800 es capaz de direccionar 64Kbytes de memoria con sus lineas de 16-bits de direccionamiento. El bus de datos de 8-bits es bidireccional al igual que tres-estados, haciendo asi direccionamiento de memoria directo y aplicaciones de multiprocesamiento fácil de realizar.

CARACTERISTICAS:

* 8-bits de procesamiento paralelo
* Bus de datos Bidireccional
* Bus de direccionamiento de 16-bits
* 72 Instrucciones -Longitud variable
* Siete modos de direccionamiento -directo, Relativo, Inmediato Indexado, Extendido, Implicito y Acumulador.
* Logitud variable del STACK
* Rearranque a base de vectores
* Vector de Interrupción enmascarada

* Interrupción no enmascarada separada --Registros internos se almacenan en el STACK
* Seis registros internos --2 Acumuladores, Registro Indice, Contador de Programa, Apuntador de STACK y Registro de Código de Condiciones
* Capacidad de Direccionamiento de memoria directo (DMA) y Multiple procesamiento
* Características de reloj simplificadas
* Velocidad de reloj de hasta 2 MHz
* Bus de interfase sin TTL
* Capacidad de ejeccución de instrucción por instrucción y Alto.

PIA

PIA: El MC6821 (Peripherial Interface Adapter) provee los estándares universales de interfase con equipo periférico a la familia 6800. Este componente es capaz de conectar al MPU con periféricos a través de 8-bits de datos periféricos bidireccionales y cuatro lineas de control. No se requiere lógica exterior para conectar la mayoria de los periféricos a este KIT.

La configuración funcional del PIA es programada por el MPU durante la inicialización del sistema. Cada una de las lineas de datos periféricos puede ser programada para actuar como entrada o salida, y cada uno de las cuatro lineas de interrupción pueden ser programadas por uno de los modos de control. Esto permite un alto grado de flexibilidad en toda la operación de interconexión.

CARACTERISTICAS

* Bus Bidireccional de 8-bits para comunicación con el MPU
* Dos Buses Bidireccionales de 8-bits para interfase con los periféricos
* Dos Registros de control programables
* Dos Registros de dirección de datos programables
* Cuatro lineas de interrupción individualmente controladas; Dos usadas también como salidas de control periférico
* Control lógico de "HANDSHAKE" para operaciónes de entrada y salida de datos
* Lineas periféricas de alta impedancia en el tercer-estado y Manejo directo de transistor
* Capacidad de programación de Interrupciónes
* Capacidad de manejar componentes CMOS en el puerto A
* Capacidad de manejar 2 TTL en ambos puertos (A y B)
* Compatible con TTL
* Operación estática

ACIA

ACIA: El MC 6850 (Asynchronous Communication Interface Adapter) provee un formato de datos y control a un periférico serial asíncrono y viceversa al MPU.

El bus de interconexión del MC6850 incluye lógica de Selección (Select), Habilitador (Enable), Leer/Escribir (read/write), interrupciones e interfase para permitir transferencia de datos en un bus direccional de 8-bits. La transmisión en paralelo del bus del sistema es transmitida a los periféricos en serie y asi mismo recibida con el formato apropiado y chequeo de errores. La configuración funcional del ACIA es programada a través del bus de datos del sistema cuando se inicializa el mismo. Un registro de control programable provee longitud de palabra variable, diferentes velocidades de transmisión controlando el reloj, control en la transmisión, control en la recepción, y control en la interrupción. El ACIA provee tambien 4 operaciones periféricas y tres lineas de control. Estas lineas permiten conectarse directamente con el modem de 0-600 bps MC6860L.

CARACTERISTICAS

* Transmisión de 8 o 9-bits
* Pariedad opcional par o non
* Chequeo de Pariedad, Sobrecorrida, y "Framing"
* Registro de control programable
* Modos del reloj opcional --/1, /16, /64
* Transmisión de hasta 1Mbps
* Detecta falso arranque
* Funciones de control periférico/modem
* Operación con uno o dos bits de alto

RAM

RAM: El MC6810 es una memoria de 128X8 bits estática diseñada en base a organización byte por byte para uso en sistemas de organizados en base a canales (buses). Es fabricado con tecnología de compuerta de silicón con canal-N. El RAM opera con una fuente de poder sencilla, es compatible con TTL y DTL, y no necesita de reloj o de realimentación por su operación estática.

Esta memoria provee acceso aleatorio en incrementos de byte por byte; ademas es posible la expansión de memoria a través de entradas de selección de chips.

CARACTERISTICAS

* Organizado en 128 bytes
* Operacion estática
* Entrada y salida bidireccional y en tres-estados
* Seis entradas de selección de chip
* Fuente de poder requerida +5.0 volts
* Compatible con TTL y DTL
* Tiempo máximo de acceso = 1.0 microsegundos para MCM6810L
  575 nanosegundos para MCM6810L-1

ROM

ROM: El MC6830 es una memoria de programación-enmascarada organizada a base de bytes para ser usada en sistemas a base de canales. Esta fabricada con tecnología de compuertas de silicón con canal-N. El ROM opera con una fuente de poder sencilla, es compatible con TTL y DTL, y no necesita de reloj o retroalimentación para refrescarse.

Este ROM provee direccionamiento de memoria en incrementos de byte por byte. Expansión de memoria es accesible a través de entradas de selección de chip.

CARACTERISTICAS

* Organizado con 1024 bytes
* Operación estática
* 4 entradas para selección del chip (opción de enmascarado)
* Fuente de poder requerida de +5.0 volts
* Compatible con TTL y DTL
* Tiempo máximo de acceso = 575 ns

V.- TECNICAS DE PROGRAMACION

V.- TECNICAS DE PROGRAMACION

INTRODUCCION

Este capítulo esta diseñado para ayudar a aprender los fundamentos dé programación en código máquina y lenguaje ensamblador del microprocesador MC6800 y sus componentes periféricos. Dirigido a esos estudiantes de los primeros años de la universidad , aunque puede ser usado exitosamente por estudiantes capaces de nivel preparatoria. Este capítulo hace énfasis en desarrollar una habilidad para diseñar y documentar la programación adecuadamente.

El capítulo consta de nueve prácticas. Las prácticas guian al estudiante desde los inicios y familiarización con el equipo hasta obtener una base fundamental en el manejo de microprocesadores.

DIVISION DE PRACTICAS

Las prácticas estan subdivididas en secciones. Estas secciones son :

+ Consideraciónes Básicas
+ Discusión Preliminar
+ Información Adicional
+ Proceso Práctico
+ Reporte
+ Evaluación de la Práctica

El instructor de laboratorio considerará el material que sea necesario entregar al alumno. Se sugiere que le sea entregado lo siguiente : Lo referente a a) Consideraciónes Básicas, b) Discusión Preliminar, y c) Proceso Práctico. La Información Adicional tendra que ser seleccionada por el instructor ya que esta información podria contener el resultado de la práctica y el objetivo de la información es tener un banco de información que pueda ayudar al alumno a realizar su práctica adecuadamente y no obtenerla ya elaborada.

El instructor mantendrá en su poder todo el material aqui descrito y que sea una ayuda al instructor para la preparación de las prácticas y capacitación del alumno y de esta manera, evaluar al alumno de la mejor forma posible.

Se sugiere al instructor plantearle al alumno un panorama de la práctica donde lo inducira a usar su astucia en la solución de estas prácticas.

Además, lo instruirá en la interpretación de los conceptos y manejo de información que el alumno deberá tomar en cuenta para resolver los problemas de estas prácticas y los que encontrará como

Ingeniero. De acuerdo a la división de prácticas algunos de estos conceptos son :

+ Discusión Preliminar
  - Planteo del Problema y Especificaciónes
  - Objetivos
  - Ventajas
  - Conceptos Técnicos
  - Aspectos que influyen en la solución deseada
  - Aspectos que excluyen la solución deseada
  - Planteo del Problema
  - Algoritmo solución

+ Consideraciónes Básicas
  - Sistemas a Usar
  - Componentes a Usar
  - Limitaciónes técnicas correspondientes
  - Conocimientos requeridos
  - Como resolver los cuestionarios previos

+ Información Adicional
  - Como han sido resueltos problemas similares
  - En donde existe información al respecto
  - Cuales son las tendencias actuales al respecto
  - Diagramas Eléctricos y a bloques
    Configuración
    Especificaciónes del material utilizado
  - Formas de Onda
  - Expresiones Booleanas
  - Tablas de Verdad
  - Sistemas Numéricos

+ Proceso Práctico (Formulación)
  - Diseño de Modelo Teórico y Modulación
    * Cálculo de Ciertos Componentes
    * Mantener diseño con situaciónes reales y prácticas
    * Facilidades de obtener el material requerido
  - Diagramas de Flujo
    * En forma de trabajo bien definido
    * Adecuado al diseño respectivo y devidamente programado en el tiempo y tomando en cuenta el factor económico
  - Areas de trabajo
  - Tiempo estimado para realizar el trabajo
  - Papeleria Anexos

+ Proceso Práctico (Realización)
  - Datos del Problema
  - Que esperar a la salida
  - Sugerencias para facilitar la práctica
  - Fuentes exteriores
    * De Alimentación
    * De Potencia
    * De Información

- Guías
  * Circuitos Eléctricos
  * A Bloques
  * Diagrama de flujo
- Realizar la Práctica
  * Conexiónes
  * Introducir Programa
  * Introducir Datos
- Pruebas
- Cuestionarios
- Conclusiónes

+ Reporte
  - Resultados experimentales
    * Pruebas, Correcciónes y Ajustes
    * Comportamiento ante los cambios
    * Descripción y Calificación del funcionamiento
    * Parámetros necesarios para conocer una aplicación del dispositivo
    * Descripción del funcionamiento mediante fotografías, gráficas, y/o tablas, explicaciónes breves.
    * Eficiencia

+ Evaluación de la Práctica
  - Conocer la salida para ejercer la función de èfectividad
  - Cumplimiento del objètivo
  - Panorama actual de las aplicaciónes
    * Descripción (Mundial y México)
    * Experimentos realizados
    * El problema del capital ($) y como mejorar la tecnología
    * Promover inquietud en el estudio
  - Evaluación Técnica
    * Comportamiento de características de diseño con las obtenidas experimentalmente
    * Mantenimiento
    * Vida del Experimento o práctica
    * Dimensiónes y ventajas del experimento o práctica
      Hardware.....Físicas
      Software.....Lógicas y Físicas
    * Tipo de Operación
      Silenciosa
      Respuesta con respecto al Tiempo
      Respuesta con respecto a la temperatura
      Respuesta con respecto a la presión
      Confiabilidad
      Seguridad
  - Conclusiónes

Pues bien, es cierto que algunos conceptos no se aplican a estas prácticas, pero el concepto de como resolver un problema en la vida

diaria para un Ingeniero es el mismo; el instructor y este manual pueden ser de gran ayuda para que el estudiante adquiera la habilidad para resolver estos problemas.

PRACTICA No.1 :

FAMILIARIZACION:
CONOCIMIENTO DEL EQUIPO,
LENGUAJE ENSAMBLADOR Y
DIRECCIONAMIENTO SIMBOLICO.

5.11.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.

* LIMITACIONES TECNICAS
  Dependera del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS
  Estudio del manual de referencia para programar el MC6800.
  Estudio del manual del sistema operativo MUDBUG.
  Estudio del manual del sistema operativo D3JBUG.
  Conocimiento perfecto del sistema hexadecimal.
  Conocimiento y capacidad para elaborar diagramas de flujo.
  Conocimiento de Mnemonicos del lenguaje ensamblador:
  NAM, ORG, OPT, END, FCC, FCB, etc..
  Conocimiento y habilidad para manejar el direccionamiento simbólico.

* CUESTIONARIO PREVIO

NOTA: Las Respuestas estan a doble densidad.
Asegúrese de identificar las respuestas.

1.- Llene los blancos en la siguiente tabla de números de 8 bits en las varias representaciones que son indicadas. Si un número dado no puede ser representado en alguno de los esquemas de representación de 8-bits, llene los blancos con "XXXX". Los números en la columna de "Valor Real" estan en decimal, y todos los otros números en la tabla se encuentran en hexadecimal.

VALOR REAL (decimal)	MAGNITUD Y SIGNO	COMPLEMENT 2's	COMPLEMENT 1's
-00	$80	XXXX	$FF
-99	$E3	$9D	$9C
-128	XXXX	$80	XXXX
-119	$F7	$89	$88

2.- Asuma que los siguientes números hexadecimales estan en representación complemento a 2, en 8-bits. Desarrollar las sumas de abajo en complemento a 2, 8-bits tal como lo haría un microprocesador. Si sobreflujo ocurre

(ej. si el valor de la respuesta no puede ser propiamente representado en forma de 8-bits complemento a 2), coloque un asterisco después de la suma de 8-bits.

$BD	$97	$74	$95
+$AB	+$E3	+$27	+$8E
$68 *	$7A	$9B *	$23 *

3.- Asuma que los siguientes números en hexadecimal estan representados en 8-bits, complemento a 1. Desarrolle las sumas de abajo tal y como las haría la computadora. Si sobreflujo aritmético ocurre, coloque un asterisco después de la suma de 8-bits.

$AB	$89	$36	$69
+$BC	+$AB	+$67	+$74
$67	$34	$9D *	$DD *
+ 1	+ 1		
$68 *	$35 *		

4.- Desarrolle las siguientes operaciónes en aritmética hexadecimal.

$54BA	$F1C0	$AB4B	$1783
+$439D	-$6BC3	+$DECC	-$84E9
$9857	$85FD	$18A17	-$6D66

5.- Suponga que cierta microcomputadora tiene 16 k de memoria. Un k en el argot de computación es igual a 1024, asi que la memoria de la computadora contiene 16,384 palabras. Suponga mas adelante que esta microcomputadora tiene 16 instrucciónes diferentes (todas estas son instrucciónes de referencia-a-memoria, de una dirección) en su conjunto de instrucciónes, de esta manera hay 16 distintos códigos operacionales. También asuma que dos bits de cada instrucción estan reservados para indicar cual (si alguno) de los 3 index registers de la computadora va a ser usado en la computación del operando, y otro bit de cada instrucción puede ser colocado en uno para indicar que direccionamiento indirecto postindexado va a ser usado. Finalmente asuma que los bits restantes de la instrucción contienen un campo de dirección sencillo que es suficiente para permitir a la computadora direccionar toda su memoria directamente.

(a) Dibuje un posible diagrama para una instrucción, mostrando todos los campos y el número de bits en cada uno.

OPCODE	INDEX	I	DIRECCION	
4	2	1	14	bits

(b) ¿Cuantos bits debe tener el MAR (memory address register) de la computadora propuesta? 14 ¿Cuantos bits

debe tener el MBR (memory buffer register) de la computadora? 21 ..

(c) ¿Cual es el rango de direcciónes de memoria (en hexadecimal) para esta maquina? $3FFF .

(d) ¿Cuantos bits debe tener el PC (Program Counter) de esta computadora? 14 .

6.- Eescribe las instrucciónes en lenguaje ensamblador para inicializar la dirección simbólica "Contador" con el valor hexadecimal 4A.

```
LDA A =$4A
STA A CONTADOR
```

7.- Liste los 4 directivos necesarios en lenguaje ensamblador para un programa.

```
NAM     ----> Nombre del Programa.
OPT O,S----> Opción, para requerir un archivo
             objeto y tabla de símbolos.
ORG     ----> Origen. Designa la dirección de inicio.
END     ----> Fin de Programa.
```

8.- Liste los comandos en lenguaje ensamblador para formar un Byte y caracter constante.

```
FCB   ----> Form Constant Byte.
FDB   ----> Form Double Byte.
FCC   ----> Form Constant Character.
```

9.- Se tiene un programa fuente en lenguaje ensamblador con los siguientes formatos :

```
              NAM     FAMILIAR
              OPT     O,S
              ORG     $0200
L1            FCB     13,$2C
L2            FCC     /A/B/
L3            FDB     $453F,$12
                |
                      Comandos de Programa
                |
              END
```

(a) Demuestre el programa objeto generado por el ensamblador :

```
                      NAM     FAMILIAR
                      OPT     O,S
                      ORG     $0200
0200    0D      L1    FCB     13,$2C
0201    2C
0202    45      L2    FDB     $453F,$12
```

```
0203    3F
0204    00
0205    12
0206    41      L3      FCC     /A/B/
0207                    END
```

10.- ¿Porque es deseable usar direccionamiento simbolico en lugar de direccionamiento absoluto?.

RESPUESTA

Direccionamiento absoluto, e.j. 1A24 para almacenamiento de datos puede tener ciertos inconvenientes como son : (a) Hasta que el programa es ensamblado, las direcciónes disponibles pueden ser desconocidas. (b) Si muchas direcciónes son usadas para diferentes propositos, se hace dificil recordar el propósito de cada dirección mientras se prepara el programa. (c) Si un programa es modificado mas tarde, ciertas direciónes son usadas para almacenar datos pueden estar no disponibles y es necesario reasignar las direcciónes de almacenamiento.

5.12.- DISCUSION PRELIMINAR

Durante la primera parte de tu laboratorio, aprenderas a operar el equipo disponible en el laboratorio, como son: Terminales, Microprocesadores, impresoras, modems y otros componentes.

Fuentes de poder, módulos de microcomputadora, terminales, y en algunos casos, MODEMS (modulador/demodulador), se conectan todos juntos y en este orden de encendido. En el caso que existan modems, este deberá ser puesto en la selección correcta de operación de transmisión -baud rate-. Si tú no haces todo correctamente, puedes llegar a quemar algun componente, así que por favor tén cuidado al hacer las conexiónes. Busca siempre ayuda del instructor de laboratorio, a menos que entiendas perfectamente lo que estas haciendo.

Tu instructor de laboratorio discutirá el equipo de laboratorio, y demostrará como hacer uso de las facilidades del laboratorio. El instructor de laboratorio responderá a cualquier pregunta del sistema operativo o del manual de instrucciónes del MC6800 que tengas en mente despues de que tu hayas leido y estudiado los manuales.

Debes estudiar las funciónes de las teclas del kit-2 "P", "L", "N", "V", "M", "E", "R", "G". Y si el sistema operativo esta disponbible, debes estudiar y entender los comandos "*", "A", "B", "C", "E", "G", "H", "I", "M", "Q", "X", y "Z" antes del primer laboratorio.

Los estudiantes que rapidamente llegan a ser diestros con todos los comandos MUDBUG, invariablemente encuentran que su trabajo a través del semestre es substancialmente disminuido. Asegúrate de preguntar a tu instructor de laboratorio cualquier punto que no este claro porque te beneficiarás aprendiendo ahora (la manera facil) en lugar de aprender mas tarde (la manera dificil).

El estudiante irá a una estación de microcomputadora o al microprocesador, y usara la tecla "M" si usa el kit directamente o el comando "C" si usa MUDBUG para introducir el siguiente programa hexadecimal en lenguaje máquina en localidades de memoria consecutivas empezando con la localidad en la cual es igual al equivalente hexadecimal de la suma decimal de los dígitos de tu número de credencial.

```
86    19
8B    23
8B    32
8B    11
97    03
D6    03
C4    F0
D7    04
3F
20    ED
```

Cada 2 números representan un valor de 8-bits que debe ser puesto en una palabra de la memoria de la microcomputadora. Los valores han sido agrupados para hacerte mas fácil la traducción de lenguaje máquina a lenguaje ensamblador. Cuando este programa es ejecutado, la ejecución empieza con la instrucción que aparece al principio del area de memoria del programa. La ejecución termina cuando el programa ejecuta la instrucción SWI que aparece cerca del fin de programa.

Para introducir un valor a una localidad de memoria se prosigue como sigue:

Usando MUDBUG. Teclee "C", MUDBUG desplegará un espacio en blanco, demostrando que esta listo. Entonces deberás teclear la dirección de memoria deseada (en hexadecimal) seguida por un <CARRIAGE RETURN> (Retorno de carro). La dirección y los contenidos de la localidad deseada serán mostrados en hexadecimal en la próxima linea. Para cambiar los contenidos de las localidades, teclée los contenidos hexadecimales seguidos por un punto. El comando "C" y todos los otros comandos de MUDBUG son explicados y con mucho mayor documentación, por supuesto, en tu manual de MUDBUG. Después de haber estudiado tu manual, no dudarás en decidir que el punto no es la mejor manera de terminar una linea y querer seguir introduciendo un programa en localidades de memoria consecutivas. ?Cual es la mejor manera?.

Una vez que hayas introducido el programa dentro de la memoria, (1) use el comando "M" del MUDBUG para obtener un listado de memoria. (2) use el comando "C" del MUDBUG terminando la linea con retorno de carro <CR> para ir paso a paso a través de la memoria, y examinar cada localidad a su paso. (3) Use el comando "C" terminado con asterisco (*) para examinar las localidades de tu programa en orden inverso.

Use el comando "G" para ejecutar el programa como sigue. Tecleé "G" seguido por la dirección de inicio de ejecución del programa (ej. la direccion de la primera instruccion). El control sera transferido a esa localidad y el programa sera ejecutado. Cuando la instrucción SWI en el programa es ejecutada, el control regresa a MUDBUG (la instrucción SWI siempre devuelve el control a MUDBUG, así que puede pensar que es una instrucción de alto -HALT-). Después de que la instrucción SWI ha sido ejecutada, MUDBUG imprimirá los valores hexadecimales de los registros AR, BR, XR, PR, y SP (Stack Pointer), e imprimirá los códigos de condición (CCR) en binario. Los códigos de condición son H,I,N,Z,V y C.

Tu probablemente habrás notado que el programa afecta los códigos de condición del MC6800. Usa el comando "E" varias veces para establecer los códigos de condición con varios valores, y después de cada comando "Q" verifica que los códigos de condición están establecidos como tú lo especificaste. Hay otro comando MUDBUG además del "E" que puede ser usado para borrar los códigos de condición. Demuestre este comando e incluye una sección en tu reporte en la cual explicas el comando que usaste aquí.

Ahora establezca los códigos de condición en un valor inicial y observe que pasa. ?Exactamente cual es el valor final del código de condición?.

Tu te preguntarás porqué hay una instrucción BRANCH después de la instrucción SWI. Note que la instrucción BRANCH en cuestión apunta al inicio de programa, y también note que el PR siempre apunta a la instrucción BRANCH cuando SWI devuelve control a MUDBUG. La instrucción BRA esta añadida al programa para hacer más fácil la ejecucción del programa una y otra vez. Explique como la instrucción BRA te permite usar el comando "H" para ejecutar el programa después de haberlo ejecutado por vez primera. El comando "H" es designado para eliminar la necesidad de recordar la dirección de inicio cada vez que tengas que correr el programa.

Finalmente , usa el comando de MUDBUG "I" para borrar las localidades de memoria que has estado usando. Entonces pon el programa otra vez en la memoria sin usar el comando "C", (se creativo), y después obtén un listado de memoria para verificar que has introducido tu programa corréctamente. Explica en tu reporte como cargaste tu programa de nuevo en la memoria sin el uso del comando "C".

Antes de que tu vengas al laboratorio, traduce el programa en lenguage-máquina a un programa completo en lenguage-ensamblador con los comentarios adecuados. Después elabora el diagrama de flujo del programa a nivel instrucciones, esto es, usa notación con flechas en el flujo de tu diagrama y demuestra que pasa exactamente al nivel de instrucción. Será requerido que demuestres tu diagrama de flujo y el programa en lenguaje ensamblador a tu instructor de laboratorio.

El reporte de este laboratorio deberá contener, (1) La traducción en lenguaje ensamblador con comentarios completos; (2) El diagrama de flujo a nivel instrucción; (3) La respuesta a todas las preguntas que tiene esta práctica y (4) conclusiones.

5.13.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, F y G.

Se puede encontrar información adicional en la bibliografía con referencia 1,8, y 9.

5.14.- PROCESO PRACTICO

El próceso práctico consiste en lo siguiente:

* Llevar al laboratorio el cuestionario previo
* Llevar al laboratorio el diagrama de flujo
* Llevar al laboratorio el programa codificado
* Realizar la práctica y demostrarlo al instructor siguiendo los pasos que indica el capítulo anterior

5.15.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

FAMILIARIZACION:
Conocimiento del Equipo,
Lenguaje ensamblador y
Direccionamiento Simbólico.

Nombre del alumno
No. de Credencial
Materia
Lab 1
Fecha

COMANDOS A ENTENDER:

* - Un asterisco introduce comentarios. Puede ser usado con otro comando para invertir el sentido de visualización de las localidades de memoria.

A - Imprime el valor hexadecimal del registro AR, y acepta un nuevo valor para AR.

B - Imprime el valor hexadecimal del registro BR, y acepta un nuevo valor para BR.

C - Cambia (después de imprimir) los contenidos de la localidad de memoria INICIO (START).

E - Establece un nuevo valor del Estado de condiciones (CCR) (después de imprimir el valor existente del CCR).

G - Va a la localidad 'INICIO' (START) para ejecutar el programa de usuario.

H - Alto y Regreso a continuar con el proceso que inicia donde este apuntando el Contador de Programa (Program Counter) PR.

I - Inicializa localidades desde 'INICIO' (START) hasta 'ALTO' (STOP) con el parámetro indicado (KEY).

M - Proporciona un listado de memoria de la localidad 'INICIO' hasta 'ALTO' en terminal.

Q - Despliega los registros en Terminal o impresora. Estos registros son AR, BR, PR, SP, IX, y CCR (HINZVC).

X - Imprime el valor hexadecimal del registro IX, y acepta un nuevo valor para IX.

Z - Pone ceros en los registros AR, BR, IX, PR y CCR; y además inicializa el Apuntador de Stack (SP).

1).- TRADUCCION

Traducción del lenguaje ensamblador con sus respectivos comentarios. Para cuestiones de esta práctica se tomará el número de credencial 000010.

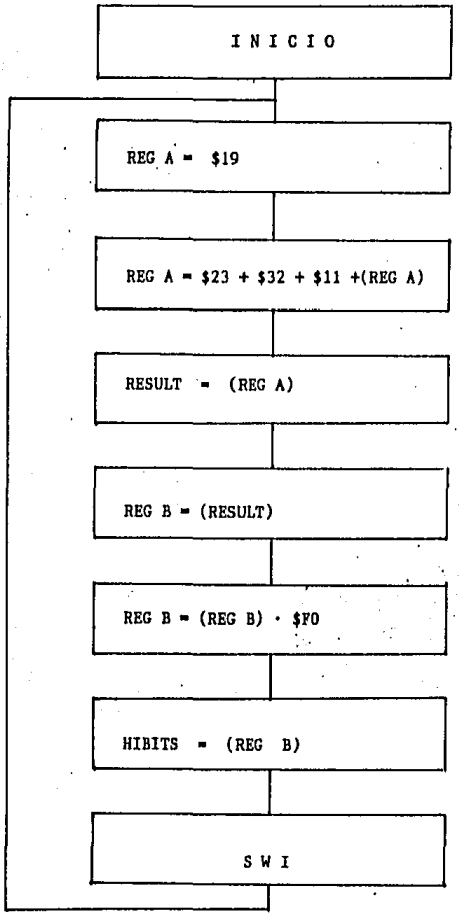
LINEA	ETIQUETA	COD OP	LENG ENSM	COMENTARIOS
			ORG $000A	
000A	INICIO	86 19	LDAA =$19	Carga registro A con el valor hex.19 Se coloca la etiqueta 'INICIO' para hacer referencia al principio del programa.
000C		8B 23	ADDA =$23	Se efectúa la operación de adición a
000E		8B 32	ADDA =$32	registro AR en estas 3 instrucciones
0010		8B 11	ADDA =$11	El resultado esta en AR.
0012		97 1E	STAA RESULT	El resultado se almacena en la localidad de etiqueta RESULT, correspondiente a la localidad $1E.
0014		D6 1E	LDAB RESULT	El registro BR lo utilizamos para to
0016		C4 F0	ANDA =$F0	mar el resultado, enmascarar los 4
0018		D7 1D	STAB HIBITS	bits de mayor significancia y los al macenamos en la localidad HIBITS.
001A		3F	SWI	Alto, Software Interrupt. Regresa el control al sistema operativo. En est caso MUDBUG. y es tiempo de revisar el estado final de los registros.
001B		20 ED	BRA INICIO	Al volver al programa, el Contador d Programa PR apunta al incio de esta instrucción y usando el comando 'H' de MUDBUG, el control del programa empieza aquí. Con esta instrucción por lo tanto se regresa al inicio de programa.
001D	HIBITS	00	RMB 1	Se reserban estas 2 localidades para
001E	RESULT	00	RMB 1	almacenamiento temporal de datos.
			END	Fin de Programa. No genera código op

```
NAM     SUMA
OPT     O,S
```

Programa SUMA, suma 4 números, $19, $23, $32 y $11 utilizando el registro AR y almacenando el resultado en la localidad de memoria RESULT. Después utilizando el acumulador B se almacenan los 4 bits de mayor significancia en la localidad HIBITS.

TECNICAS DE PROGRAMACION

2).- DIAGRAMA DE FLUJO



3).- CUESTIONARIO

a) Cual es la mejor manera de terminar una linea.

La mejor manera es usar Retorno de Carro (Carriage Return) CR.

b) ?Cual comando puede ser usado para borrar los códigos de condición?.

El comando "Z".

c) ?Cual es el valor final del Código de Condiciones?.

	H	I	N	Z	V	C	AR	BR
VALOR INICIAL	X	X	X	X	X	X	X	X
LDAA =$19	X	X	0	0	0	X	$19	X
ADDA =$23	0	X	0	0	0	0	$3C	X
ADDA =$32	0	X	0	0	0	0	$6E	X
ADDA =$11	0	X	0	0	0	0	$7F	X
STAA RESULT	0	X	0	0	0	0	$7F	X
LDAB RESULT	0	X	0	0	0	0	$7F	$7F
ANDB =$F0	0	X	0	0	0	0	$7F	$70
STAB RESULT	0	X	0	0	0	0	$7F	$70
SWI								

VALOR FINAL: SI I = 0 entonces CCR = $00
SI I = 1 entonces CCR = $10

d) Explique como es usado eel comando "H" de MUDBUG en este programa.

El comando "H" continúa o ejecuta la instrucción apuntada por el PC y en ese punto el PC marca la loclidad $001B y esta instrucción hara que el programa continúe su ejecución brincando al inicio del programa y por lo tanto ejecutándolo de nuevo.

e) Como colocar el programa de nuevo en memoria sin usar el comando 'C' de MUDBUG.

Se usa el comando 'PO' (POKE). Este comando sirve para llevar a cabo la escritura en memoria. No permite la lectura del contenido de las localidades. Este comando hace lo siguiente:

```
          PO  0A
y aparece  000A  y se teclea 86  (CR)
y aparece  000B  y se teclea 19  (CR)
y asi sucesivamente hasta introducir todo el programa.
Para salir de este comando se usa la terminación punto(.) .
```

5.16.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere que se haga tomando en cuenta los puntos a continuación se mencionan:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. Un punto por cada pregunta.

b) Respecto al Proceso Práctico: Se otorgarán 20 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa.

c) Con Respecto al Reporte: Se otorgarán 70 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

20 puntos por la parte (1)
20 puntos por la parte (2)
20 puntos por la parte (3)
10 puntos por la parte (4)

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....20 puntos
Punto (c)....70 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.2 :

REGISTRO INDICE,
'BRANCH' EN CODIGO MAQUINA Y
'BRANCH' EN LENGUAJE ENSAMBLADOR.

5.21.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal
  Codigo ASCII

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS
  Uso del Registro Indice.
  Uso del 'OFFSET' (Apendice-N) en registro Indice.
  Instrucciones de 'BRINCO' relativo (BRANCH).
  Uso y cálculo del 'OFFSET' en las instrucciones (BRANCH).
  Uso de etiquetas locales.

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

1.- Inicializar 2 Bytes de memoria $0030 y $0031, con el valor hexadecimal $1A3F. Incluya el código máquina.

```
CE 1A3F      LDX  =$1A3F
FF 0030      STX  $0030
```

2.- Escriba las instrucciones en lenguaje ensamblador para mover los contenidos de las localidades $0100 y $0101 a las localidades $0120 y $0121. Suponga que el registro índice IX contiene $0100; y deje el registro índice con su valor inicial.No use el registro BR para realizar esta operación.

```
LDAA  0,X
STAA  $20,X
LDAA  1,X
STAA  $21,X
```

3.- Escriba las instrucciones en lenguaje ensamblador para incrementar los contenidos de la localidad 'CONTADOR'

la cual almacena el valor hex. $91FF. Y Cual sera el nuevo contenido (2 Bytes) despues de realizar lo anterior.

```
            LDX     CONTADOR
            INX
            STX     CONTADOR
            ::
            ::
CONTADOR    RMB     2
```

EL NUEVO CONTENIDO SERA = CONTADOR / $9200

4.- Para el programa de abajo asigne las direcciones apropiadas, empezando en $1017. Las 2 primeras direcciones ya han sido asignadas.

```
LINEA       ETIQUETA        MNEMONICO
1017        INIT            LDAA =$2A
1019                        STAA VALFIN
101C                        LDX  =$0015
101F                        STX  MEMOR
1022                        RTS
1023        VALFIN          RMB  1
1024        MEMOR           RMB  2
```

5.- Para el programa de abajo calcule el 'OFFSET' requerido para brincar a la localidad 'AQUI'.

```
LINEA     COD OP        ETIQUETA      MNEMONICO
2345      8C 1017                     CPX  =INIT
2348      26 ____                     BNE  AQUI
234A      CE 2A30                     LDX  =PUNT
234D      BD 2A35                     JSR  MENSA
2350      39            AQUI          RTS
```

RESPUESTA: El contador de Programa apunta a $234A después de la instrucción 'BRANCH'. La localidad a la cual queremos llegar (si se cumple la condición) es la $2350, por lo tanto ...

$2350 - $234A = $0006

El 'OFFSET' es igual a $06

6.- Ahora calcule el 'OFFSET' del programa siguiente:

```
LINEA     COD OP        ETIQUETA      MNEMONICO
0115      86 13                       LDAA =$13
0117      5C            AQUI          INCB
0118      F7 811C                     STAB CONTA
011B      4A                          DECA
011C      26 ____                     BNE  AQUI
011E      39                          RTS
```

```
Localidad Destino  = $0117
Localidad    PC    = $011E

$0117 - $011E      = $FFF9
```

7.- Ejecute la acción del compilador manualmente, ensamblando manualmente el siguiente programa. Calculando el 'OFFSET' de cada instrucción 'BRANCH'. El código máquina para JSR ADCHR es BD 1F00 y para JSR TERM es BD 1F03. Empieze en la localidad $0220.

```
          PRCHR     JSR     ADCHR
                    LDX     MEMORIA
                    INX
                    STX     MEMORIA
                    LDAA    0,X
                    CMPA    =$0E
                    BEQ     FINLINEA
                    JSR     TERM
                    BRA     PRCHR
          FINLINEA  RTS
          MEMORIA   RMB     2
```

```
$0220   BD  1F00          PRCHR     JSR     ADCHR
$0223   FE  0236                    LDX     MEMORIA
$0226   08                          INX
$0227   FF  0236                    STX     MEMORIA
$022A   A6  00                      LDAA    0,X
$022C   B1  0E                      CMPA    =$0E
$022E   27  05                      BEQ     FINLINEA
$0230   BD  1F03                    JSR     TERM
$0233   20  EB                      BRA     PRCHR
$0235   39                FINLINEA  RTS
$0236   00002             MEMORIA   RMB     2
```

5.22.- DISCUSION PRELIMINAR

PROCESO DE LA PRACTICA.

Esta práctica tiene como finalidad demostrar al alumno algunas de las aplicaciones usando el registro índice, valiendose de instrucciones 'BRANCH' para agilizar el proceso de las mismas aplicaciones.

La práctica consiste en realizar 2 programas en lenguaje ensamblador y su correspondiente código máquina para así introducirlo al KIT-D2 y mostrar al instructor del laboratorio la correcta operación de tu programa. Tu reporte deberá contener el programa en código máquina y lenguaje ensamblador, así como también un listado de la memoria afectada, Resultado final de los registros y las limitaciones de tu propio programa.

El primer programa consiste en almacenar el mensaje "HEXADECIMAL NO VALIDO" en memoria RAM cuando este tiene la etiqueta MENS (mensaje) y con terminación de un nulo (0).(Use el ensamblador como ayuda). Asuma que la instrucción -JSR PRINT- llama a una subrutina, la cual imprime los contenidos del Registro A en impresora. Por ahora la subrutina PRINT sera como sigue:

```
$0400  FE  040A   PRINT    LDX   IMPPNT     Actualiza apuntador
 0403  08                   INX               de impresion.
 0404  FF  040A             STX   IMPPNT
 0407  A7  00               STAA  0,X         Almacena contenido de
                                              Registro A en memoria
 0409  39                   RTS               Regreso de subrutina
 040A             IMPPNT    RMB   2           Reserva 2 Bytes.
```

Ahora bien, tu programa deberá imprimir tal mensaje. No olvides de inicializar el apuntador de impresión. Se sugiere inicializar IMPPNT con el valor $0010. Tu programa inicia en la localidad $0200 del PSEUDORAM (ROM simulado), El programa sugerido por el instructor tomo 53 Bytes de PSEUDORAM, 22 Bytes de RAM para almacenamiento de datos y el "STACK POINTER" (apuntador de Stack) no fue utilizado.

NOTA: El sitema operativo permite la impresión del mensaje por medio de una subrutina de utilería, en caso de no localizarla use la subrutina indicada arriba.

El segundo programa consiste en checar si el caracter en el registro A es un caracter hexadecimal válido. Esto es, que sea del 0 al 9 o de la A a la F. Introduzca el caracter ASCII al registro A. Al final de la rutina el registro A debera contener el equivalente hexadecimal en 4 bits si este es válido. Si no es válido el caracter use el programa anterior para imprimir el mensaje ("HEXADECIMAL NO VALIDO"). Iniciar el programa en la localidad $0250. El programa sugerido ocupo 21 Bytes de PSEUDORAM, 0 Bytes de RAM y no fue usado el "STACK POINTER".

5.23.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, B, F, G, H y I.

Se puede encontrar información adicional en la bibliografía con referencia 1,4, 8 y 9.

5.24.- PROCESO PRACTICO

Se realizará la práctica con el KIT-D2 del laboratorio o con el emulador de microprocesadores instalado en la computadora. El estudiante verificará el corrrecto funcionamiento de sus 2 programas y después de esto los demostrará a su instructor de laboratorio.

A cualquier pregunta no dude en pedir ayuda a su instructor.

5.25 REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

REGISTRO INDICE,
'BRANCH' código Máquina y
'BRANCH' Lenguaje Ensamblador.

Nombre del alumno
No. de Credencial
Materia
Lab 2
Fecha

IMPMEN... Este programa consiste en imprimir un mensaje el cual esta almacenado en memoria. Llama a la subrutina PRINT para cada caracter que se desea imprimir. Se emplea el registro A y el X; además la subrutina PRINT.

LINEA	ETIQUETA	COD OP	LENG	ENSM	COMENTARIOS
			NAM	IMPMEN	
			OPT	O,S	
			ORG	$0200	
$0200	IMPMEN	CE 021F	LDX	=MENS-1	Inicia Programa con la etiqueta del nombre del programa.
$0203		FF 021D	STX	POINTR	Inicializa el apuntador POINTR con la dirección del mensaje.
$0206		CE 0010	LDX	=$0010	Inicializa apuntador de
$0209		FF 040A	STX	IMPNT	impresión. Donde empezará a imprimir.
$020C	1H	FE 021D	LDX	POINTR	Actualiza el apuntador
$020F		08	INX		usando el registro Indice.
$0210		FF 021D	STX	POINTR	
$0213		A6 00	LDAA	0,X	Carga un caracter del mensaje en el Registro A.
$0215		27 05	BEQ	FIN	Si es NULO entonces brinca al fin de programa.
$0217		BD 0400	JSR	PRINT	Si es caracter entonces imprime ese caracter del mensaje actualmente en el Registro A.
$021A		20 0E	BRA	1B	Regresa a la etiqueta 1 atrás para analizar el siguiente caracter.
$021C	FIN	3F	SWI		Fin de Programa

```
                                        "Software Interrupt"

$021D   POINTR  RMB     2               Reserva 2 localidades de
                                        memoria.  Esa localidad
                                        apunta al primer caracter
                                        del mensaje.

$021F   MENS            FCC             'HEXADECIMAL NO VALIDO'
$0234           00      FCB  0          Nulo
```

MEMORIA AFECTADA POR EL PROGRAMA:

```
      0 1 2 3 4 5 6 7 8 9 A B C D E F
$0010 H E X A D E C I M A L   N O   V
$0020 A L I D O
```

Con el correspondiente código ASCII de cada caracter.

Resultado final de los registros:

AR/$00 BR/INICIAL XR/$0234

PC/$0210 SP/INICIAL CC/$40

```
   H I N Z V C
   ' ' 0 1 0 '
```

LIMITACIONES:

Depende de Cada programa. El alumno inidicará si existe o no alguna limitante en su programa.

CHECKHEX... Este programa checa si el caracter en el registro A es un valor hexadecimal válido. El registro A contiene al final el equivalente hexadecimal en 4 bits si este es válido, de otra manera desplegará el mensaje del programa anterior.

LINEA	ETIQUETA	COD OP	LENG	ENSM	COMENTARIOS
			NAM	CHEKHEX	
			OPT	O,S	
			ORG	$0250	
$0250	CHEKHEX	86 XX	LDA	=CHAR	Aqui se introduce el caracque va a ser analizado.
$0252		80 30	SUBA	=$30	Ya que 30 o 37 tienen que ser restados la substracción se hace desde el principio.
$0254		2B AA	BMI	IMPMEN	Si el resultado de la substracción es negativo indica que el caracter tiene un valor menor de 30, por lo tanto menor que cero.
$0256		81 09	CMPA	=09	Se compara si el caracter
$0258		2F 0A	BLE	FIN	esta entre 0-9. brinca a fin si es cierto. Caracter válido.
$025A		80 07	SUBA	=07	Se efectúa una substracción
$025C		81 0F	CMPA	=$0F	para verificar los caracte-
$025E		22 A0	BHI	IMPMEN	res de A a F. 46-30-7 = 0F. Se compara con 0F y si es mayor entonces no es un caracter hexadecimal. brinca a rutina de mensaje.
$0260		81 09	CMPA	=09	Se compara con el valor de
$0262		2F 9C	BLE	IMPMEN	A. 41-30 = 0A. Si es menor que A es un caracter no válido.
$0264	FIN	3F	SWI		
$0265		20 E9	BRA	CHEKHEX	Regresa a inicio de programa para correr el progrma una vez más. (después de accionar el comando : CONTINUAR)

MEMORIA AFECTADA POR EL PROGRAMA:

Si el caracter analizado no es un caracter hexadecimal, la memoria afectada sera la misma que se afecta en el programa anterior.

Resultado final de los registros:

AR/Caracter equivalente hex en 4 bits.

BR/INICIAL

XR/INICIAL

PC/$0265

SP/INICIAL

CC/$40

H	I	N	Z	V	C
'	'	X	X	0	0

LIMITACIONES:

Depende de Cada programa. El alumno inidicará si existe o no alguna limitante en su programa.

5.26.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada pregunta es 1.42 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 20 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos de los 2 programas, es decir 10 puntos por cada programa y se calificarán los siguientes tópicos en cada programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

c) Con Respecto al Reporte: Se otorgarán 70 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

PARTE 1.- PROGRAMA 1

* Lenguaje ensamblador y código máquina 20 puntos. Por cada byte usado de mas se tomará 1 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

* Reporte del contenido final de los registros 5 puntos.

* Reporte de las limitaciones 5 puntos.

La Suma TOTAL = 35 puntos.

PARTE 2 .- PROGRAMA 2
Los mismos conceptos que en la parte 1.
TOTAL = 35 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....20 puntos
Punto (c)....70 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.3 :
MAS ACERCA REGISTRO INDICE
E INSTRUCCIONES 'BRANCH'

5.31.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Número Hexadecimal
  Codigo ASCII

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS
  Uso del Registro Indice.
  Uso del 'OFFSET' (Apendice-N) en registro Indice.
  Instrucciones de 'BRINCO' relativo (BRANCH).
  Uso y calculo del 'OFFSET' en las instrucciones (BRANCH).
  Uso de etiquetas locales.

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

Cada Conjunto de números hexadecimales en la columna "ANTES" demuestra los valores que contienen los registros y las localidades de memoria del MC6800, al principio de la ejecucción de cada instruccidn. Tu problema es llenar la columna de "DESPUES" con los valores correspondientes despuds de que la instruccidn ha sido ejecutada. Considera cada instruccidn independiente de las otras. Demuestra todos los valores en hexadecimal. También escribe el lenguaje ensamblador equivalente de cada instruccidn de máquina que es dada en cada inciso. Note que el parentesis significa "contenido de", así, por ejemplo, "($0200)" significa los contenidos de la localidad $0200.

	ANTES		DESPUES		INSTRUCCION Y DESCRIPCCION
(1)					
	(PR) =	$F100	(PR) =	$F103	CPX =$34CB
	($F100) =	$8C	($F100) =	$8C	
	($F101) =	$34	($F101) =	$34	Xrl--] $CB
	($F102) =	$CB	($F102) =	$CB	Xrh--] $34
	(XR) =	$34DE	(XR) =	$34DE	
	(CCR) =	$D5	(CCR) =	$D1	
(2)					
	(PR) =	$0125	(PR) =	$00F6	BGE $FFCF + * + 2
	($0125) =	$2C	($0125) =	$2C	= BGE $00F6
	($0126) =	$CF	($0126) =	$CF	Banderas cumplen con
	(BR) =	$80	(BR) =	$80	condiciones...
	(CCR) =	$DB	(CCR) =	$DB	N or(ex) V = 0
					ocurre brinco

(3)

(PR)	=	$EF01	(PR)	=	$EE8F	BLE $FF8C + * + 2
($EF01)	=	$2F	($EF01)	=	$2F	= BLE $ EE8F
($EF02)	=	$8C	($EF02)	=	$8C	Condiciones:
(AR)	=	$00	(AR)	=	$00	Z + (N or(ex) V) = 1
(CCR)	=	$D3	(CCR)	=	$D3	Si brinca

(4)

(PR)	=	$EF00	(PR)	=	$EF02	BGT $FFA0 + * + 2
($EF00)	=	$2E	($EF00)	=	$2E	= BGT $ FFA0
($EF01)	=	$A0	($EF01)	=	$A0	Condiciones:
(AR)	=	$CA	(AR)	=	$CA	Z + (N or(ex) V) = 0
(CCR)	=	$D3	(CCR)	=	$D3	No brinca

(5)

(PR)	=	$0400	(PR)	=	$0402	ADCB $FF,X
($0400)	=	$E9	($0400)	=	$E9	
($0401)	=	$FF	($0401)	=	$FF	Se toma el contenido
(BR)	=	$ED	(BR)	=	$DB	de:
(XR)	=	$8081	(XR)	=	$8081	($FF + $8081) = ($8180)
($8080)	=	$7C	($8080)	=	$7C	($8180) $ED
($8081)	=	$8E	($8081)	=	$8E	(BR) $ED
($8180)	=	$ED	($8180)	=	$ED	+ C 1
(CCR)	=	$C5	(CCR)	=	$N/R	$DB

NOTA: El asterisco en las instrucciones 'BRANCH' significa la localidad donde empieza la instrucción que se esta ejecutando.

5.32.- DISCUSION PRELIMINAR

Esta práctica tiene como finalidad dar al alumno la oportunidad de demostrar la habilidad que este posee para manejar el registro índice (IX) y el uso de las instrucciones de brinco (BRANCH).

La práctica consiste en realizar 2 programas en lenguaje ensamblador y su correspondiente código máquina para así introducirlo al KIT-D2 y mostrar al instructor de laboratorio la correcta operación de tu programa.

Tu reporte deberá contener el programa en código máquina y lenguaje ensamblador, así como también un listado de la memoria afectada, resultado final de los registros y limitaciones del programa.

El primer programa consiste en borrar las localidades de memoria de la localidad 00 a la localidad 4F. Los registros A y B deben contener su valor inicial al final de la rutina, es decir no deberán ser afectados. El programa solución toma 10 bytes de memoria. Inicie su programa en la localidad $0200.

El segundo programa consiste en contar el número de veces que un valor entre +/- 1F (hex) inclusive aparecen dentro de un rango de memoria. De la localidad $0000 a la $0100 inclusive. Cheque manualmente que su programa funcione para valores de +/- 1F y +/- 20 antes de demostrarlo al instructor de laboratorio. Inicie su programa en la localidad $0200. El programa solución toma 33 Bytes PSEUDORAM y 3 Bytes de RAM, No se usa el STACK POINTER. El resultado deberá ser almacenado en la localidad "CONTADOR". Recuerde que limpieza en su reporte final será apreciada.

5.23.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, B, F, G, H e I.

Se puede encontrar información adicional en la bibliografía con referencia 1, 4, 8 y 9.

5.34.- PROCESO PRACTICO

Se realizará la práctica con el KIT-D2 del laboratorio o con el emulador en computadora. El estudiante verificará el correcto funcionamiento de su programa y después de esto lo demostrará al instructor de laboratorio.

Para fines de verificar si los programas están correctos pueden usar los 2 programas indicados adelante como programas maestros.

El instructor puede colocar los siguientes programas en memoria y así verificar que los programas del alumno sean correctos.

PROGRAMA MAESTRO 1.- Para checar que las localidades del 00 al 4F esten en ceros.

PASO 1... Introducir el siguiente programa:

```
0300        LDX  =$50
SIGUE       DEX
            BLT  FIN
            LDAA 0,X
            BEQ  SIGUE
FIN         SWI
```

PASO 2... Corra el programa del alumno, tecleando
G 0200

PASO 3... Corra el programa maestro, tecleando
G 0300

PASO 4... Verifique si el registro A esta en cero "0".
Si es así el programa esta correcto.
Si el registro A muestra cualquier otro valor entonces el programa es incorrecto.

PROGRAMA MAESTRO 2.- Para checar que el contéo es correcto.

PASO 1... Introducir el siguiente prógrama:

```
                org $0300
0300            CLRA
                LDX  =$0100
SIGUE           STAA 0,X
                INCA
                DEX
                BNE   SIGUE
                STAA  0,X
                SWI
```

PASO 2... Corra el programa maestro, tecleando
G 0300

PASO 3... Corra el programa del alumno, tecleando
G 0200

PASO 4... Después cheque el Registro A. Sí este contiene 3F entonces el programa esta correcto. de otra manera no lo esta.

5.35.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

MAS ACERCA REGISTRO INDICE
E INSTRUCCIONES 'BRANCH'

Nombre del alumno
No. de Credencial
Materia
Lab 3
Fecha

BORMEM.... Este programa consiste en borrar cierta región de memoria. De la localidad $00 a la localidad $4F. Los registros A y B no son afectados.

LINEA	ETIQUETA	COD OP	LENG	ENSM	COMENTARIOS
			NAM	BORMEM	
			OPT	O,S	
			ORG	$0200	
$0200	BORMEM	CE 0200	LDX	=$4F	Inicia Programa en la localidad $0200 e inicializa registro índice con el valor de la localidad mas alta que deseamos borrar.
$0203	SIGUE	6F 00	CLR	0,X	Borra la localidad marcada
$0205		09	DEX		con el index register y se
$0206		26 FB	BNE	SIGUE	borran así en orden decreciente hasta que el valor del IX sea cero.
$0208		6F 00	CLR	0,X	La localidad 00 es borrada en este momento.
$020A	FIN	3F	SWI		Interrupción del programa.
$0			END		Fin de programa.

MEMORIA AFECTADA POR EL PROGRAMA:

Listado de memoria. De la $00 a la $4F en ceros.

Resultado final de los registros:

AR/INICIAL	BR/INICIAL	XR/$00
PC/$020B	SP/INICIAL	CC/$04

H I N Z V C
' ' 0 1 0 0

LIMITACIONES:

Depende de Cada programa. El alumno inidicará si existe o no alguna limitante en su programa.

CHECKMEM.... Este programa cuenta el número de veces que valores entre +/-1F aparecen de la localidad $0000 a la $0100. El resultado se almacena en la localidad "CONTADOR".

LINEA	ETIQUETA	COD OP	LENG	ENSM	COMENTARIOS
				NAM CHECMEM	
				OPT O,S	
				ORG $0200	
$0200	CHECKMEM	86 1F		LDAA =$1F	Inicia Programa en la localidad $0200.
$0202		97 06		STAA LIMMAY	Se marca el límite mayor ($1F), y se asigna una localidad de memoria.
$0204		40		NEGA	Se marca el límite menor
$0205		97 05		STAA LIMMEN	(-1F), y se le asigna una localidad de memoria.
$0207		7F 0005		CLR CONTADOR	Se borra el contador.
$020A		CE 0000		LDX =$00-1	Se inicializa el registro con el valor -1 de la localidad donde se empezará a analizar.
$020D	SIGUIENT	08		INX	Incrementa en uno al apuntador de memoria.
$020E		A6 00		LDAA 0,X	Se almacena el valor de la localidad en el registro A para poder ejecutar las operaciones.
$0210		91 06		CMPA LIMMAY	Compara con límite mayor y
$0212		2E 07		BGT CARGA	si es mayor, el programa brinca para analizar la siguiente localidad.
$0214		91 05		CMPA LIMMEN	Compara con límite menor y
$0216		2D 03		BLT CARGA	si es menor, el programa brinca para analizar la siguiente localidad.
$0218		7C 0007		INC CONTADOR	Suma uno al contador.
$021B	CARGA	8C 0100		CPX =$0100	Compara la última localidad a analizar.
$021E		26 EE		BNE SIGUIENT	Si no es la última regresa en 'LOOP' para analizar otra localidad.

```
$0220   FIN       3F      SWI          Interrupción Software.
$0005   LIMMEN    RMB     1            Reserva bytes de memoria
$0006   LIMMAY    RMB     1            para trabajo de almacena-
$0007   CONTADOR  RMB     1            miento.

                  END                  Fin de Programa.
```

MEMORIA AFECTADA POR EL PROGRAMA:

Listado de memoria.

```
$0005    LIMMEN   -$1F
$0006    LIMMAY    $1F
$0007    CONTADOR $3F
```

Resultado final de los registros:

AR/$0100 BR/INICIAL XR/$0100

PC/$0221 SP/INICIAL CC/$04

```
H I N Z V C
' ' 0 1 0 0
```

LIMITACIONES:

Depende de Cada programa. El alumno inidicará si existe o no alguna limitante en su programa.

5.36.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 30 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos de los 2 programas, es decir 10 puntos por cada programa y se calificarán los siguientes tópicos en cada programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

c) Con Respecto al Reporte: Se otorgarán 60 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

PARTE 1.- PROGRAMA 1

* Lenguaje ensamblador y código máquina 15 puntos. Por cada byte usado de más se tomará 1 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

* Reporte del contenido final de los registros 5 puntos.

* Reporte de las limitaciones 5 puntos.

TOTAL = 30 puntos.

PARTE 2 .- PROGRAMA 2
Los mismos conceptos que en la parte 1.
TOTAL = 30 puntos.

Las partes están señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....30 puntos
Punto (c)....60 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.4 :

APUNTADOR DE 'STACK'
"STACK POINTER"

5.41.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numerico Hexadecimal,y Decimal

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS
  Instrucciones para uso del STACK:

-LDS	LoaD Stack pointer
-STS	STore Stack pointer
-TXS	Transfer IndeX to Stack
-TSX	Transfer Stack to IndeX
-INS	INcrement Stack pointer
-DES	DEcrement Stack pointer
-PSH	PuSH data to stack
-PUL	PULl data from stack

  Manejo del Stack Pointer

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

1.- Inicialize el stack pointer a $A027, entonces almacene los contenidos del Registro A y del registro B en el STACK en ese orden. Indíquelo en lenguaje ensamblador.

```
LDS   #$A027
PSHA
PSHB
```

2.- Ahora suponga que el registro A contiene $3F y el registro B contiene $20. Iniciando el programa con la respuesta anterior y siguiendo con lo que se muestra abajo:

```
INCA
DECB
PSHB
PSHA
PULA
PULB
 ::
 ::
```

Indique el contenido final de los registros:

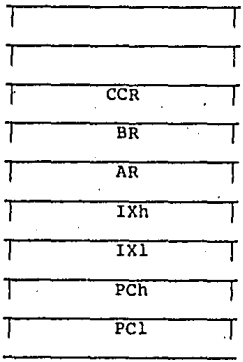
```
AR ... $1F
BR ... $40
SP ... $A025
```

3.- Asuma que 5 bytes han sido almacenados en el Stack. Se desea incrementar el tercero de estos 5 bytes sin afectar al apuntador de Stack o cualquier otro dato dentro del Stack. Escriba las instrucciones necesarias para hacerlo. Use solamente 2 instrucciones.

```
TSX
INC 2,X
```

4.- Indique como se ve afectado el Stack después de que se ha interrumpido el sistema por medio de un requerimiento de interrupción (IRQ).

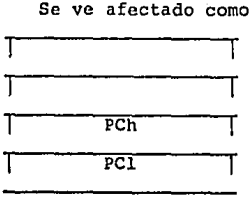
Se ve afectado como sigue:



El Requerimiento de Interrupción almacena los registros indicados dentro del Stack.

5.- Indique como se ve afectado el Stack después de que se ha ejecutado una instrucción "JUMP TO SUBRUTINA (JSR)".

Se ve afectado como sigue:



El Contador de programa (PC) es almacenado en el Stack. Primero PC Low Byte y despues PC High Byte.

5.42.- DISCUSION PRELIMINAR

Esta práctica tiene como finalidad dar al estudiante la oportunidad de demostrar sus conocimientos en las funciones del Apuntador de Stack y sus ventajas.

La práctica consiste en realizar un programa en lenguaje ensamblador y su correspondiente código máquina para así introducirlo al KIT-D2 y mostrar al instructor la correcta operación de tu programa.

Tu reporte deberá contener el programa en código máquina y lenguaje ensamblador; así como también un listado de la memoria afectada, Resultado final de los registros y las limitaciones del programa.

Con el objeto de mantener mas ventajas en el uso del apuntador de Stack se deberá desarrollar el programa con el menor numero de instrucciones ya que por este concepto se evaluará una parte de la práctica. 2 puntos menos por cada byte mas que haya sido usado, tomando en comparasión que el programa solución ha tomado solamente 9 Bytes de PSEUDORAM Y cero Bytes de RAM, haciendo uso por supuesto del Stack Pointer.

El programa consiste en lo siguiente: Se desea obtener una tabla de datos en orden descendente. Esto es, comenzando en la localidad $0000 con $FF, después en la localidad $0001 con $FE y así sucesivamente hasta llegar a la localidad $00FF y tener almacenado ahí mismo $00. Comienze su programa en la localidad $0200.

5.43.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, H,y I.

Se puede encontrar información adicional en la bibliografía con referencía 1 y 4.

5.44.- PROCESO PRACTICO

Se realizará la práctica con el KIT-D2 del laboratorio o con el emulador de la computadora. El estudiante verificará el perfecto funcionamiento de su programa y después lo demostrará al instructor de laboratorio.

Para verificar su programa, cheque los puntos críticos y algún otro valor aleatorio.

PASO 1... Cheque la localidad $0000 debe contener $FF
PASO 2... Cheque la localidad $0001 debe contener $FE
PASO 3... Cheque la localidad $00FE debe contener $01
PASO 4... Cheque la localidad $00FF debe contener $00
PASO 5... Cheque la localidad $007F debe contener $80

Si su programa cumple con estas condiciones, tiene un 90% de seguridad de que el programa es correcto.

El instructor deberá checar mas localidades y así hacer la evaluación correcta. De ser posible se recomienda checar al menos 30 localidades de memoria aleatoriamente para comprobar el programa.

5.45.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

APUNTADOR DE 'STACK'
"STACK POINTER"

Nombre del alumno
No. de Credencial
Materia
Lab 4
Fecha

TABDAT... Este programa consiste en colocar una tabla de datos en orden descendente. Comenzando en la localidad $00 y terminando en la localidad $FF. Se uso el registro A y el Apuntador de Stack. Inicia en la localidad $0200.

LINEA	ETIQUETA	COD OP	LENG	ENSM	COMENTARIOS
			NAM	TABDAT	
			OPT	O,S	
			ORG	$0200	
$0200	TABDAT	8E 00FF	LDS	=$FF	Inicia Programa en la localidad $0200 y coloca al apuntador de Stack apuntando a la localidd $FF. Razón empezar la tabla en la localidad $00FF.
$0203		4F	CLRA		Borra el Registro A. Se coloca el primer valor
$0204	NEXT	36	PSHA		Coloca el primer valor en la tabla. $00 en $00FF.
$0205		4C	INCA		Incrementa registro A en uno. Se coloca el siguiente valor en el registro A.
$0206		26 FC	BNE	NEXT	Basándose en que el registro A tiene solamente 8 bits después de haber colocado $FF en la localidad $0000, El reg. A es incrementado eso hace que se coloque en ceros y dando así fin al loop.
$0208	FIN	3F	SWI		Interrupción Software.
			END		Fin de Programa.

MEMORIA AFECTADA POR EL PROGRAMA:

Listado de memoria. De la $00 a la $FF de la siguiente manera.

```
$0000 ....FF
$0001 ....FE
$0002 ....FD
  ::      ::
  ::      ::
$00FD ....02
$00FE ....01
$00FF ....00
```

Resultado final de los registros:

AR/$00	BR/INICIAL	XR/INICIAL
PC/$0209	SP/$FFFF	CC/$04

```
   H I N Z V C
   ' ' 0 1 0 0
```

NOTA: Los programas de los alumnos pueden tener otro status final de los registros, dependiendo por supuesto de su programa.

LIMITACIONES:

Depende de Cada programa. El alumno inidicará si existe o no alguna limitante en su programa.

En este programa las limitaciones son que puede ocurrir una interrupción al programa y por lo tanto causaría anomalías en el resultado final.

5.46.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Practico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 5.44 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

c) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 20 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 10 puntos.

* Reporte del contenido final de los registros 10 puntos.

* Reporte de las limitaciones 5 puntos.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causaran baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

PRACTICA No.5 :

PRINCIPIOS DE LA CALCULADORA
EN UNA MICROCOMPUTADORA.

5.51.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Número Hexadecimal,y Binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS
  Instrucciones para uso del STACK:

-ADD	ADD
-ADC	ADd with Carry
-SUB	SuBstract
-SBC	SuBstract with Carry
-ROL	ROtate Left
-ROR	ROtate Right
-ASL	Arithmetic Sifht Left
-ASR	Arithmetic Shift Right
-LSR	Logic Shift Right

  Instrucciones para uso de Subrutinas:

-JSR	Jump to SubRutine
-BSR	Branch to SubRoutine
-RTS	ReTurn from Subroutine

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

Cada Conjunto de números hexadecimales en la columna "ANTES" demuestra los valores que contienen los registros y las localidades de memoria del MC6800, al principio de la ejecucción de cada instrucción. Tu problema es llenar la columna de "DESPUES" con los valores correspondientes despues de que la instrucción ha sido ejecutada. Considera cada instrucción independiente de las otras. Demuestra todos los valores en hexadecimal. También escribe el lenguaje ensamblador equivalente de cada instrucción de máquina que es dada en cada inciso. Note que el paréntesis significa "contenido de", así, por ejemplo, "($0200)" significa los contenidos de la localidad $0200.

	ANTES		DESPUES		INSTRUCCION Y DESCRIPCCION
(1)					
	(PR) = $0300		(PR) = $0301		RORA
	($0300) = $46		($0300) = $46		
	(AR) = $87		(AR) = $C3		CCR--} AR bit 7
	(CCR) = $D5		(CCR) = $C9		bit 7--} bit 6 etc... AR bit 0 --} CCR
(2)					
	(PR) = $2000		(PR) = $2003		ADDB $0155
	($2000) = $FB		($2000) = $FB		
	($2301) = $01		($2301) = $01		BR + ($0155)---} BR
	($2002) = $55		($2002) = $55		
	($0155) = $BC		($0155) = $8C		
	(BR) = $D9		(BR) = $95		
	(CCR) = $C3		(CCR) = $E9		
(3)					
	(PR) = $0300		(PR) = $0302		ADCA $95,X
	($0300) = $A9		($0300) = $A9		
	($0301) = $95		($0301) = $95		C + AR + (($R)+$95))--}AR
	($0095) = $B1		($0095) = $B1		
	($D095) = $E7		($D095) = $E7		
	(AR) = $85		(AR) = $6D		
	(XR) = $D000		(XR) = $D000		
	(CCR) = $F5		(CCR) = $D3		
(4)					
	(PR) = $2100		(PR) = $2102		SBCB $D5
	($2100) = $D2		($2100) = $D2		
	($2101) = $D5		($2101) = $D5		BR - ($D5) - C --} BR
	($00D5) = $86		($00D5) = $86		
	(BR) = $95		(BR) = $0E		
	(CCR) = $C7		(CCR) = $C0		

(5)

(PR)	=	$4000	(PR)	=	$4002	SBCA $55,X
($4000)	=	$A2	($4000)	=	$A2	
($4001)	=	$55	($4001)	=	$55	AR - (XR + $55) - C --) AR
($0053)	=	$BC	($0053)	=	$BC	
(AR)	=	$E9	(AR)	=	$2D	
(XR)	=	$FFFE	(XR)	=	$FFFE	
(CCR)	=	$C2	(CCR)	=	$C0	

5.52.- DISCUSION PRELIMINAR

Esta práctica tiene como finalidad dar al alumno la oportunidad de demostrar su habilidad en los principios aritméticos con los que funciona una calculadora usando microprocesadores.

La práctica consiste en realizar una subrutina (Programa) en lenguaje ensamblador y su correspondiente código máquina para así introducirlo al KIT-D2 y mostrar al instructor la correcta opercaión de tu programa.

Tu reporte deberá contener la subrutina en código máquina y lenguaje ensamblador; así como también un listado de la memoria afectada, resultado final de los registros y las limitaciones del programa. Buena documentación a nivel función sera evaluada.

La subrutina consiste en lo siguiente:

Elaborar un programa que multiplique 2 números binarios sin signo de 8-bits, produciendo un resultado de 16-bits. Reserve 4 bytes de memoria, uno para el multiplicando (M1), otro para el multiplicador (M2) y 2 bytes para el resultado (R1 y R2). Reserva mas Bytes si así lo requieres, pero indique la razón de reservar mas bytes y la función de éstos en su reporte. Inicie su programa en la localidad $0200.

Recuerda colocar las limitaciones de tu programa en el reporte, son muy importantes para el resultado y la evaluación de tu práctica.

El programa sugerido tomo 36 Bytes de PSEUDORAM, 5 Bytes de RAM , Se usaron ambos registros A y B, y no se hizo uso del Stack Pointer.

5.53.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, B,e I.

Se puede encontrar información adicional en la bibliografía con referencia 1 y 4.

5.54.- PROCESO PRACTICO

Se realizará la práctica con el KIT-D2 del laboratorio o con el emulador de Computadora. El estudiante verificará el correcto funcionamiento de su programa y después de esto lo demostrará al instructor del laboratorio.

Para verificar que el programa este correcto, el estudiante y/o el instructor podrán disponer del siguiente programa -que llamaremos el programa MAESTRO- y seguir las instrucciones siguientes.

PASO 1: Introducir el programa del alumno en el KIT-D2, iniciando en la localidad $0200.

PASO 2 : Introducir el programa maestro en el KIT-D2 iniciando en la localidad $0300.

```
PROGRAMA MAESTRO
         NAM   MASTER
         OPT   O,S
         ORG   $0300
         CLRA
         STAA  M1
         STAA  M2
         JSR   MULT   ------)  Subrutina del alumno
         LDX   R2
         STX   PRUEBA1
         LDAA  =$7F
         STAA  M1
         STAA  M2
         JSR   MULT   -------)  Subrutina del alumno
         LDX   R2
         STX   PRUEBA2
         LDAA  =$FF
         STAA  M1
         STAA  M2
         JSR   MULT   -------)  Subrutina del alumno
         LDX   R2
         STX   PRUEBA3
         SWI
         END
```

```
PRUEBA1     RMB     2
PRUEBA2     RMB     2
PRUEBA3     RMB     2
```

PASO 3 : Checar las direcciones de las localidades para que las variables M1, M2 y R2 de ambos programas sean iguales.

PASO 4 : Correr el programa MAESTRO.

```
G 0300
```

PASO 5 : Checar los valores de las siguientes localidades despuds de ejecutar el programa MAESTRO:

(PRUEBA1) = $0000
(PRUEBA2) = $3FFF
(PRUEBA3) = $FFFF

Si es correcto el programa funciona.

PASO 6 : Si el instructor cree necesario verificar con otros valores, se le alienta a hacerlo y así obtener un mejor punto de vista para la correcta evaluación.

5.55.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

PRINCIPIOS DE LA CALCULADORA
EN UNA MICROCOMPUTADORA.

Nombre del alumno
No. de Credencial
Materia
Lab 5
Fecha

MULT.... Este programa es una subrutina que consiste en multiplicar 2 números no signados de 8-bits, M1 y M2, produciendo un resultado de 16-bits, en la memoria con etiquetas R1 y R2. Se inicia en la localidad $0200.

LINEA	ETIQUETA	COD OP	LENG	ENSM	COMENTARIOS
			NAM	MULT	
			OPT	O,S	
			ORG	$0200	Inicia programa en la localidad $0200
$0200	MULT	4F		CLRA	Borra Registro A para así
$0201		97 12		STAA LTRA	borrar las áreas de resul-
$0203		97 14		STAA R1	tado y de trabajo.
$0205		97 13		STAA R2	
$0207		96 11		LDAA M2	M2 = Multiplicador
$0209		20 06		BRA LOOP1	
$020B	LOOP2	78 0010		ASL M1	Arithmetic Shift left al
$020E		79 0012		ROL LTRA	Multiplicando. Byte supe-
					rior del multiplicando.
$0211	LOOP1	44		LSRA	Se mueve el mulptiplicador
$0212		24 0D		BCC NOSUM	a la derecha, el lsb se almacena en el Carry bit, si es cero no sume/no ejecute las siguentes instrucciones.
$0214		D6 14		LDAB R1	Suma el Multiplicando al
$0216		DB 10		ADDB M1	Byte menos significativo.
$0218		D7 14		STAB R1	
$021A		D6 13		LDAB R2	Suma con carry el byte mas
$021C		D9 12		ADCB LTRA	significativo.
$021E		D7 13		STAB R2	
$0220		4D		TSTA	Prueba si el multiplicador

```
$0221   NOSUM    26 E8    BNE    LOOP2    es cero. si lo es entonces
                                          la operación ha terminado.

$0223             39       RTS             Regreso a Programa
                                           Principal.

                                NOTESE QUE NO EXISTE EL
                                SOFTWARE INTERRUPT.

$0010   M1        RMB      1               MULTIPLICANDO
$0011   M2        RMB      1               MULTIPLICADOR
$0012   LTRA      RMB      1               LOCALIDAD DE TRABAJO
$0013   R2        RMB      1               BYTE SUPERIOR DEL RESULTADO
$0014   R1        RMB      1               BYTE INFERIOR DEL RESULTADO
```

MEMORIA AFECTADA POR EL PROGRAMA:

Después de correr el programa maestro :

```
$0020 ....$0000
$0022 ....$3FFF
$0024 ....$FFFF
```

Resultado final de los registros:

AR/$00	BR/$FF	XR/$FFFF
PC/$032D	SP/INICIAL	CC/$48 - H I N Z V C 1 ' 0 1 0 0

NOTA: Los programas de los alumnos pueden tener otro status final de los registros, dependiendo por supuesto de su programa.

LIMITACIONES:

Depende de Cada programa. El alumno inidicará si existe o no alguna limitante en su programa.

5.56.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 5.54 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

c) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 20 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 10 puntos.

* Reporte del contenido final de los registros 10 puntos.

* Reporte de las limitaciones 5 puntos.

TOTAL = 45 puntos.

Las partes están señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.6 :

SUBRUTINAS

5.61.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numerico Hexadecimal.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  Instrucciones para uso de Subrutinas:
  -JSR Jump to SubRutine
  -BSR Branch to SubRoutine
  -RTS ReTurn from Subroutine

  Conocimiento de la frecuencia del crystal usado en el KIT.

  Conocimiento del número de ciclos del MPU.
  Interpretación y localización de esta información por cada instrucción.

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

Considere el programa de abajo en lenguaje ensamblador para el MC6800. (nota: El programa no intenta desarrollar una función útil cuando es ejectuado. Solo sirve para darte mayor comprensión del funcionamiento del M6800). La instrucción SWI al final actúa como un alto y regresa el control al sistema operativo. Todas las otras instrucciones las encuentras en el manual de referencia para programadores del MC6800.

```
LINEA   ETIQUETA  COD OP    LENG ENSM    COMENTARIOS
                            NAM  MULT
                            OPT  O,S
                            ORG  $0100   Inicia en la localidad 100
$0100   INICIO    CE 0035   LDX  =$35    XR [---$0035
$0103             0D        SEC          C [---1
$0104             86 BA     LDAA =-106   AR [--$BA
$0106             C6 20     LDAB =32     BR [--$20
$0106   LOOP      E7 0D     STAB 13,X    (XR+($0D)) [-- BR
$010A             5A        DECB         BR [-- BR - 1
$010B             09        DEX          XR [-- XR - 1
$010C             56        RORB         ROTAR BR A LA DERECHA
$010D             4C        INCA         AR [--AR + 1
$010E             2F F8     BLE  LOOP    VA A LOOP SI A MENOR o = 0
$0110   FIN       3F        SWI          ALTO, REGRESA A SIST. OP.
                            END
```

1.- Traduzca el programa de lenguaje ensamblador a lenguaje máquina propia del microprocesador MC6800. Los valores en lenguaje-máquina deberán ser expresados en código hexadecimal.

RESPUESTA ARRIBA

2.- Documente el programa.

RESPUESTA ARRIBA

3.- Suponga que el programa es cargado en las localidades de memoria indicadas y ejecutado. ?Cuales son los valores hexadecimales finales. (Después de haberse ejecutado la instrucción SWI) de los siguientes registros y localidades de memoria. Asuma que el código de condiciones (CCR) es igual a $C0, cuando empieza la ejecucción y no considere el efecto de la instrucción SWI en el registro (CCR).

(PR) = $0111 (AR) = $01 (CCR) = $C0 ($15) = $20

(XR) = $FFEE (BR) = $42 ($E) = $85 ($27) = $20

4.- ?Que otros registros o localidades de memoria son afectados con el programa?. Si es asi, especifique cuales, y como son afectados.

```
>M 0,100

0000  63 C7 8F 20 42 85 0B 18 31 63 C7 8F 20 42 85 0B
0010  18 31 63 C7 8F 20 42 85 0B 18 31 63 C7 8F 20 42
0020  85 0B 18 31 63 C7 8F 20 42 85 0B 18 31 63 C7 8F
0030  20 42 85 0B 18 31 63 C7 8F 20 42 85 0B 18 31 63
0040  C7 8F 20 00 00 00 00 00 00 00 00 00 00 00 00 00
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

FFF0 ----------------------------------85 0B 18 31
```

Las localidades de $FFFC a $FFFF no son cambiadas en en el KIT-D2.

5.- Use la tarjeta de programación del MC6800 y determine exactamente cuanto tarda la corrida del programa. Asuma un ciclo igual a 1.0 microsegundo. Incluya tiempo de ejecucción de la instrucción SWI.

TIEMPO = 1441 MICROSEGUNDOS.

5.62.- DISCUSION PRELIMINAR

Esta práctica tiene como finalidad dar al alumno la oportunidad de demostrar sus conocimientos en las funciones de subrutinas, aplicaciones ventajas y sus desventajas.

Un programa puede ser hecho de una serie de llamadas a subrutinas, cada una causando ejecucción de una subrutina en particular para desarrollar una tarea especifica. Cada subrutina debe contener solamente un punto de entrada y otro de salida. Entrada y Salida deben estar bien documentadas . Cada subrutina puede ser probada individualmente y entonces usada con confianza cuando es llamada del programa principal.

El plan de programa debe ser en formato de "arriba-a-abajo", con las tareas completamente definidas y de esto las subrutinas bien definidas. Cada tarea puede ser asignada a una subrutina, la cual puede a su vez llamar a otra subrutina de menor nivel. Las llamadas a subrutinas pueden tener varios niveles de profundidad, si fuera necesario.

Después de esta breve descripción de subrutina y con lo aprendido en el salón de clases, el alumno desarrollará una subrutina que produzca un retardo de N milisegundo, donde N es el contenido binario del registro A. Esta subrutina deberá llamar a la subrutina MILISEC que produce un retardo de 1 milisegundo cada vez que es llamada. Asuma 1 microsegundo por cada ciclo del MPU.

NOTA : Investigue en el laboratorio la frecuencia del reloj usado en el KIT y utilícela para sus cálculos del programa.

Inicia tu subrutina en la localidad $0200. La subrutina tomo 13 Bytes de Pseudoram, 0 Bytes de RAM, se usaron los registros A y B y no se uso el Apuntador de Stack. Minimiza tu programa, el número de Bytes que uses en tu programa sera tomado en consideración.

Entonces , esta práctica consiste en realizar una subrutina en lenguaje ensamblador y su correspondiente código máquina para así introducirlo al KIT- D2 y mostrar al instructor la correcta operación de tu programa.

Tu reporte deberá contener la subrutina en código máquina y lenguaje ensamblador, así como también un listado de la memoria afectada. Resultado final de los registros y las limitaciones del mismo.

5.63.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, D, E, F, H y I.

Se puede encontrar información adicional en la bibliografía con referencia 1, 5 y 9.

5.64.- PROCESO PRACTICO

Se realizará la práctica con el KIT-D2 del laboratorio o con el emulador en la computadora. El estudiante verificará el correcto funcionamiento de su programa y después de esto lo demostrará al instructor del laboratorio.

Para verificar que el programa este correcto, el estudiante y/o el instructor tomarán el programa y lo analizarán haciendo los cálculos correspondientes.

Una manera de verificar el programa es colocar el siguiente programa MAESTRO, el cual llama 2 veces a la subrutina realizada por el alumno (SUBRET) y con el reloj en la mano se podra verificar si el programa es correcto.

PROCEDIMIENTO :

PASO 1 : Coloque la subrutina del alumno. Desde la localidad $0200.

PASO 2 : Introduzca el programa MAESTRO.

MAESTRO... Para verificar subrutina de retardo elaborado por el alumno.

```
        NAM   MAESTRO
        OPT   O,S
        ORG   $0300
        LDAA  #10
        JSR   SUBRET    $0200
        LDAA  #100
        JSR   SUBRET    $0200
        SWI
        END
```

PASO 3 : Con reloj en mano arranque el programa MAESTRO.

```
G 0300
```

PASO 4 : Al instante aparecer "=" debe de haber transcurrido 1 segundo.

PASO 5 : Continúe esta prueba cambiando el valor de 10 del programa MAESTRO por multiplos de 10 (es decir 20, 30, 40,....250) y se obtendrá un retardo aproximado de 1, 2, 3 .....25 segundos.

5.65.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

SUBRUTINAS

Nombre del alumno
No. de Credencial
Materia
Lab 6
Fecha

SUBRET... Esta subrutina consiste en producir un retardo de N milisegundos, donde N es el contenido binario del registro A. Se inicia en la localidad $0200.

```
LINEA   ETIQUETA  COD OP    LENG ENSM     COMENTARIOS
                            NAM  SUBRET
                            OPT  O,S
                            ORG  $0200    Inicia en la localidad 200
                                          el valor de N ya está en el
                                          registro A.

$0200   SUBRET    BD 0207   JSR  MILSEC   Llama a la subrutina de
$0203             4A        DECA          retardo de 1 milisegundo.
$0204             26 FA     BNE  SUBRET   De regreso checa si el va-
                                          lor de A es cero. Si no es
                                          cero, ejecuta de nuevo.

$0206             39        RTS           Regresa a programa
                                          principal.

        * MILSEC ...Subrutina de retardo de 1 milisegundo.
        *
        *
$0207   MILSEC    C6 A2     LDAB  =162    Inicializa el contador.
$0209   MASDEC    5A        DECB          Reg. B y lo decrementa hasta
$020A              26 FD     BNE   MASDEC cero, entonces debido a los
                                          ciclos de tiempo que requie-
                                          re esta instrucción, ha
                                          sido provocado un retardo
                                          de 1 milisegundo.

$020C              39        RTS           Regresa a subrutina SUBRET
                             END
```

MEMORIA AFECTADA POR EL PROGRAMA:

MEMORIA RAM NO SE VE AFECTADA.

Resultado final de los registros:

AR/$00	BR/$00	XR/INICIAL
PC/NOTA	SP/INICIAL	CC/$04

```
H I N Z V C
' ' 0 1 0 '
```

NOTA : De donde fue llamado, es decir si se corrío el programa maestro entonces el resultado final del PC sera $030B.

LIMITACIONES:

Este programa tiene un retardo de .999 milisegundos. lo que quiere decir que en vez de realizar el retardo de 1 milisegundo existe un error de 1 microseegundo.

Otra limitante. es que el programa no desarrolla correctamente si N es cero. Pero esto es válido ya que no es necesario usar una subrutina para retardar cero milisegundos. ?o si?.

5.66.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 5.64 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

c) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 20 puntos. Por cada byte usado de mas se tomara 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

* Reporte del contenido final de los registros 10 puntos.

* Reporte de las limitaciones 10 puntos.

TOTAL = 45 puntos.

Las partes están señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.7 :
LA COMPUTADORA Y LA COMUNICACION CON EL USUARIO.

5.71.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Número Hexadecimal.
  Sistema Número Binario.
  Sistema Número BCD.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  Conceptos, Operaciones y Manejo de los sistemas Binario y BCD.

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

1.- Represente los siguientes números en el sistema BCD.

BASE	NUMERO X		CONVERSION BCD
(10)	15	=	0001 0101
(2)	0101011	=	0100 0011
(16)	3A	=	0101 1000
(16)	24	=	0011 0110
(2)	1101101	=	1 0000 1001
(10)	32	=	0011 0010

2.- Indique el significado de las siglas BCD.

BINARY CODED DECIMAL

3.- Cual es la instrucción que hace los ajustes necesarios para corregir resultados inválidos al esperar un resultado decimal después de que se ha ejecutado alguna operación en el KIT-D2.

DAA .- Decimal Adjust accA.

4.- Ejecute las siguientes operaciones en código BCD.

```
  100001101001          10001000
+ 100010101010        - 00110111
-------------         ----------
 1100000100100          1010001
```

5.- Ahora las siguientes :

```
    10010101 --)   85
X       101 --) X  5
               ---
               425 ---) 0100 0010 0101

  010101110100 --)  576
X          11 --) X  3
                  ----
                  1728 ---) 1 0111 0010 1000
```

5.72.- DISCUSION PRELIMINAR

Esta práctica tiene como finalidad dar al alumno la oportunidad de demostrar sus conocimientos en el manejo de información por parte de la computadora y la decodificación para que el usuario pueda interpretar fácilmente esa información.

Principalmente se sabe que las computadoras basan su manejo de información a base de 1's y 0's, es decir bajo el sistema binario. Pues bien hay cadenas largas de 1's y 0's que resultan difícil de interpretar para un usuario y para que este pueda interpretar la información, es conveniente presentársela en un sistema que el pueda reconocer, esto es, el sistema decimal.

Ya que el proceso para representar números decimales es largo y con un grado de dificultad mas allá de lo aprendido hasta ahora, nos concentraremos en lograr un programa para convertir un número en sistema binario a un número en sistema BCD.

Sin usar la instrucción DAA realize :

La práctica consiste entonces en realizar un programa convertidor del sistema binario al sistema BCD en lenguaje ensamblador y su correspondiente código máquina para así introducirlo al KIT-D2 y mostrar al instructor la correcta operación. Inicia tu programa en la localidad $0200. El número binario sera de 16 bits y el resultado sera almacenado en 3 registros. El programa sugerido tomo 57 (hex) Bytes de memoria PSEUDORAM y 5 Bytes de RAM, uso de los registros A y B y no se uso el Apuntador de Stack.

Tu reporte deberá tener el programa en código máquina y lenguaje ensamblador; así como también un listado de la memoria afectada, resultado final de los registros y las limitaciones del programa. Documenta tu programa.

5.73.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, F, G, H e I.

Se puede encontrar información adicional en la bibliografía con referencia 1, 4, 8 y 9.

5.74.- PROCESO PRACTICO

Se realizará la práctica con el KIT-D2 del laboratorio o con el emulador en computadora. El estudiante verificará el correcto funcionamiento de su programa y después de esto lo demostrará al instructor del laboratorio.

Para verificar que el programa este correcto, el estudiante y el instructor tomarán el prógrama y lo analizarán haciendo los cálculos correspondientes.

El instructor puede probar con valores críticos; se recomienda usar el valor inicial binario 11111111 (base 2) y se espera obtener en las localidades de memoria correspondientes a los registros de resultado lo siguiente:

```
          0010 0101  0101  ----)   255  (base 10)
Reg --)   1    2     3
```

5.75.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

LA COMPUTADORA Y
LA COMUNICACION CON EL USUARIO

Nombre del alumno
No. de Credencial
Materia
Lab 7
Fecha

CONVERT... Este programa consiste en convertir un número binario de 16 bits a una representación BCD almacenando el resultado en 3 Registros, de 2 Bytes BCD cada uno.

```
LINEA   ETIQUETA  COD OP    LENG ENSM     COMENTARIOS
                            NAM  SUBRET
                            OPT  O,S
                            ORG  $0000     Número binario inicial
        MSB                 RMB  1        8 bits mayor significancia
        LSB                 RMB  1        8 bits menor significancia

                            ORG  $0010     RESULTADO EN BCD
        UNIDEC              RMB  1        Unidades y decenas
        CIENMIL             RMB  1        Cientos y miles
        DECMILL             RMB  1        Decenas de millar

                            ORG  $0200     Inicia en la localidad 200
$0200           7F 0010     CLR  UNIDEC   Borra los registros donde
$0203           7F 0011     CLR  CIENMIL  se almacena el resultado
$0206           7F 0012     CLR  DECMIL   inicia XR con 16 (dec)
$0209           CE 0010     LDX  =$0010

$020C   INICIO  96 10       LDAA UNIDEC   Se efectua comparación
$020E           16          TAB           de unidades.
$020F           84 0F       ANDA =$0F
$0211           80 05       SUBA =$05
$0213           2B 02       BMI  AT
$0215           CB 03       ADDB =03

$0217   AT      16          TBA           Comparación de Decenas.
$0218           84 F0       ANDA =$F0
$021A           80 50       SUBA =$50
$021C           2B 02       BMI  BT
$021E           CB 30       ADDB =$30
$0220   BT      D7 10       STAB UNIDEC

$0222           96 11       LDAA CIENMIL  Comparación de Cientos.
$0224           16          TAB
$0225           84 0F       ANDA =$0F
```

```
$0227           80 05    SUBA =05
$0229           2B 02    BMI CT
$022B           CB 03    ADDB =03

$022D   CT      16       TBA              Comparación de Miles.
$022E           84 F0    ANDA =$F0
$0230           80 50    SUBA =$50
$0232           2B 02    BMI  DT
$0234           CB 30    ADDB =$30
$0236   DT      D7 11    STAB CIENMIL

$0238           96 12    LDAA DECMILL     Comparación de decenas de
$023A           16       TAB              Millar.
$023B           80 05    SUBA =05
$023D           2B 02    BMI  ET
$023F           CB 03    ADDB =03
$0241   ET      D7 12    STAB DECMILL

$0243           78 0001  ASL  LSB         Los registros son recorridos
$0246           79 0000  ROL  MSB         para ajustar el valor.
$0249           79 0010  ROL  UNIDEC      y termina cuando el registro
$024C           79 0011  ROL CIENMIL      índice marca cero, es decir
$024F           79 0012  ROL DECMILL      ha ejecutado esta acción
$0252           09       DEX              16 veces.
$0253           26 AB    BNE  INICIO

$0255   FIN     3F       SWI              Interrupción Software.
                         END              Fin de conversión.
```

MEMORIA AFECTADA POR EL PROGRAMA:

MEMORIA RAM SE AFECTAN 5 BYTES :
$0000, $0001, $0010, $0011 y $0012

Resultado final de los registros:

AR/DEPENDE DE INICIAL	BR/DEPENDE DE INICIAL	XR/$0000
PC/$0256	SP/INICIAL	CC/$04

H I N Z V C
' ' 0 1 0 0

LIMITACIONES:

Depende del programa del alumno, este programa en particular no tiene limitaciones.

5.76.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 5.74 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

c) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 20 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.
* Reporte del contenido final de los registros 10 puntos.
* Reporte de las limitaciones 10 puntos.

TOTAL = 45 puntos.

Las partes están señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

PRACTICA No.8 :

ANALIZADOR DE BYTE

5.81.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico Binario.
  Sistema Numérico BCD.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  Uso de los comandos del sistema operativo MUDBUG

* CUESTIONARIO PREVIO

  NOTA: Las Respuestas están subrayadas y a doble densidad. Asegúrese de identificar las respuestas.

Existe una serie de Preguntas dentro de la sección de Proceso Práctico (5.84), cuando se realize la práctica y con los conocimientos adquiridos por el alumno se deberán ir contestando y se entregarán en el reporte de la práctica.

5.82.- DISCUSION PRELIMINAR

Tu misión en este laboratorio es escribir y correr un programa del MC6800 que examine y cataloge cada cuarto de byte del registro AR. Considera al registro AR compuesto de 4 cuartos de Byte. Bits 7 y 6 del AR forman el cuarto de byte mas-a-la-izquierda, bits 5 y 4 forman otro cuarto de Byte, bits 3 y 2 forman otro cuarto de Byte y el cuarto de Byte mas-a-la-derecha lo forman los bits 1 y 0. Tu programa deberá poner el registro BR al número de cuartos de Byte del registro AR que sean igual a %11 (p.e. 3), y además deberá poner el registro XR igual al número de cuartos de Byte que sean igual a %00. Tu deberás también asegurarte que AR contenga su valor inicial antes de terminar la ejecucción de tu programa.

Escribe este programa y todos los programas futuros para que sean re-arrancables. Un programa es re-arrancable si puede ser ejecutado una y otra vez cada vez que sea llamado (corrido) a su localidad inicial sin tener que recargar el programa o inicializar cualquiera de sus variables o registros. En otras palabras, programas re-arrancables automaticamente hacen su propio mantenimiento. ?Puedes dar tu un ejemplo de algun programa que no sea autoarrancable?.

Además de hacer tu programa re-arrancable, codifícalo (y todos tus futuros programas) para correr en ROM (Read Only Memory). Un programa puede correr en ROM sí y solo si este nunca modifica cualquier palabra de su propia area de memoria, así, en particular, un programa en ROM nunca puede desarrollar cualquier modificación a sus propias instrucciones. También, cualquier almacenamiento de variables que sea requerido debe ser mantenido en un area separada del RAM (Read/Write Memory) o en el STACK, el cual esta en RAM. Tu actualmente corres tus programas en PSEUDOROM (RAM) pero considera como si lo hicieras en ROM.

Asume que tu tienes ROM para tus programas empezando desde la localidad $2000, y asume que tu tienes RAM para uso temporal empezando en la localidad $0180. Tu quizás requieras hacer uso del STACK, el cual es otra area del RAM. Si tu usas el STACK, recuerda que MUDBUG convenientemente inicializa el apuntador de stack por ti, de esta manera tu no necesitas de la instrucción LDS. También recuerda que este programa y virtualmente todos los programas deben ser escritos para conservar el apuntador de stack en su posición inicial. En otras palabras si tu colocas datos en el STACK, deberás quitarlos antes de terminar tu programa.

Una vez que hayas escrito tu programa Analizador de Byte en lenguaje ensamblador, manualmente conviértelo en lenguaje máquina como lo haría un ensamblador del MC6800. Escribe la versión de lenguaje máquina en hexadecimal a la izquierda de las instrucciones en lenguaje ensamblador, y también incluye una columna para indicar las localidades de memoria de la primera palabra de cada instrucción.

La solución de este laboratorio tomó 17 localidades de memoria en ROM y usa el STACK como almacenamiento temporal.

5.83.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, F, G, H e I.

Se puede encontrar información adicional en la bibliografía con referencia 1, 8 y 9.

5.84.- PROCESO PRACTICO

Cuando vayas al laboratorio, dirígete a la estación de microcomputadora (KIT-D2 o emulador en la computadora) e introduce tu programa a la memoria usando el comando "C" de MUDBUG, el cual ya deberas de haber entendido antes de venir al laboratorio. Despues de que has cargado tu programa, genera un listado de memoria para verficar que tu verdaderamente has introducido tu programa correctamente. Entonces establece tus valores inciales para los registros AR, BR y XR, y prueba tu programa. Correlo varias veces haciendo varias pruebas variando los valores de tus registros para estar seguros de que tu programa trabaja correctamente para todos los casos. En particular, prueba tu programa que corra con valores de AR igual a 0, -1, +1, y $80, y además pruebalo con valores aleatorios.

Si tu programa falla (no corre perfectamente) tu deberás corregir esa "basura" que existe dentro de tu programa (debug). El comando "T" de MUDBUG es una ayuda invaluable para esa labor. Asegúrate de estudiar tu manual de MUDBUG considerando el comando "T" antes de venir al laboratorio. También asegúrate de preguntar a tu instructor de laboratorio cualquier duda en puntos que no entiendas completamente porque será importante para futuros laboratorios (y también otros comandos , pero especialmente el comando "T").

Si tu programa trabaja perfectamente desde la primera vez que tu lo corriste y no requieres de hacer un "debug", demuestra tu habilidad en el manejo del comando "T" usándolo varias veces con tu programa. También usa el comando "*" para anotar tus resultados si estas usando un terminal-impresora. Explica como puedes usar el comando "T" para ir a través de tu programa paso a paso una instrucción por cada paso si fuera necesario para hacer el "debug"; de cualquier manera usa el comando "T" para hacer una traza de las primeras 10 instrucciones. Después usa el comando "O" y el comando "N" para hacer lo mismo.

Quizás tu programa contenga varios "loops". Si es así, tu deberás ser capáz de usar el comando "T" para trazar la ejecucción de tu programa a través de varias iteraciones de un "loop". Explica y demuestra como el comando "T" es conveniente para problemas de "debug" dentro de un "loop". También explica que sucede cuando usas los parámetros de este comando iguales (INICIO = FIN).

Una vez que tu programa trabaja correctamente, usa el comando "W" para escribir una cinta objeto para el. Entonces usa el comando "I" para borrar el area de memoria que has estado usando, y lista el area para verificar que de hecho se ha borrado. También usa el comando "F" (con un enmascarado de cero) para probar que tu memoria ha sido borrada, y después usa el comando "F" 8 veces (con valores diferentes de cero) para probar que tu memoria esta borrada. (Asegúrate de explicar en tu reporte como usas 8 veces el comando "F"). Finalmente usa el comando "L" para cargar tu programa de nuevo en memoria y has un listado de memoria para probar que tu programa ha sido leido en memoria

correctamente. Explica que sucede si tratas de cargar tu programa con un parámetro igual a -$2000. ¿Correrá tu programa correctamente en su nueva localizacion?. ?Porqué si o Porqué no?. Pregunta a tu instructor de laboratorio cualquier duda y que hacer si tu terminal no tiene unidad de cinta para realizar los comandos "W" y "L".

Quizás tu tengas en tu programa una o mas instrucciones de brinco "BRANCH", y ese tipo de instrucciones requiere de direccionamiento relativo. Usa el comando "R" para localizar la dirección destino de una o mas instrucciones de brinco, e incluye una explicación en tu reporte que demuestre que tu entiendes el comando "R" y como ayuda en lograr un seguimiento lógico de tu programa.

Ahora deberás estar listo para usar el comando "R" para marcar la dirección destino de las instrucciones de brinco. Usa el comando "C" y el "R" para crear instruciones BRA en localidades $1AD y $1AE con dirección destino en $1F8, y tecléa "G 1AD" para verificar que tu instrucción de brinco realmente brinca ala localidad $1F8. Después sin usar el comando "R", usa solamente el comando "C" para modificar la instrucción BRA en las localidades $1AD y $1AE para hacer el brinco a la localidad $175. Después pon una instrucción SWI en la localidad $175, y teclea "G 1AD" para ver si tu modificaste la instrucción BRA correctamente" Este ejercicio deberá darte una apreciación para el manejo y ventajas del comando "R".

Finalmente usa el comando "K" "Calculadora" para la suma de varios números hexadecimales incluyendo números positivos y negativos. Tu ocacionalmente encontrarás la utilidad de este comando de mucha ayuda tenerlo a la mano para realizar operaciones aritméticas en-linea durante tu trabajo de "debug". Explica porque actualizar el parámetro INICIO (START) es conveniante al usar el comando "K".

Después de este laboratorio deberás entender los comandos de MUDBUG "F", "K", "L", "N", "O", "R", "T" y "W". Ademas de los aprendidos anteriormente.

Para este laboratorio tu reporte deberá contener: Una versión de tu programa en lenguaje ensamblador completamente documentada y su traducción a lenguaje máquina. Incluye todos los comandos necesarios de un programa ORG, EQU, RMB y END. Un diagrama de flujo de tu programa a nivel instrucción.

5.85.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

ANALIZADOR DE BYTE

Nombre del alumno
No. de Credencial
Materia
Lab 8
Fecha

Un ejemplo sencillo de un programa que no es re-arrancable es cualquiera que modifique sus propias instrucciones.

EJEMPLO:

```
                ORG  $100
$0100           ADDA #$11
$0102           STAA $0100
                SWI
```

Funcionamiento del comando "T" en loops: Es conveniente por sus 4 parámetros, y así nosotros le podemos decir al microcomputador cuantas veces queremos hacer la rutina y cuantas veces queremos que se nos muestre el "status". Por lo tanto tenemos un completo conocimiento de lo que esta sucediendo en esa pequeña rutina y como están siendo afecrados los registros.

Lo que pasa si uso el comando "T" y coloco los parámetros de tal manera que INICIO = ALTO, el programa no ejecuta nada a menos que tenga activos los parámetros 3 y 4. Algo que realmente sucede es que se coloca en en el modo de paso a paso.

El porque usar el comando "F" 8 veces para probar que la memoria ha sido borrada es muy sencillo, ya que al hacerlo 8 veces se trata de enmascarar un bit por cada paso (sin repetir el bit) y eso nos hace verificar los 8 bits con los que cuenta cada palabra y nos muestra si efectivamente la memoria esta en ceros.

PREGUNTA: Que pasa si trato de cargar el programa de regreso a la memoria con un desplazamiento de -$2000.

RESPUESTA: Ya que mi programa estaba originalmente en la localidad $2000, entonces el "offset" pondrá a mi programa en la localidad $0000.

PREGUNTA:¿Correrá el programa correctamente en su nueva direccion?

RESPUESTA: SI, porque la localidad $0000 es también RAM y viendo el mapa de memoria y la memoria que afecta el programa mismo no altera el funcionamiento del mismo.

?Como es usado el comando "R"? Es usado para determinar y/o establecer las direcciones de destino en las instrucciones de brinco. La segunda palabra de una instrucción de tipo de brinco en el MC6800 siempre contiene un valor de desplazamiento relativo signado, y de esa manera, usando el comando "R" sabemos exactamente a donde es dirigido el programa después de la instrucción de brinco. Es posible además cambiar el "offset" de la instrucción de brinco con este comando. La terminación es similar a la del comando "C". (blanco tab / * , . o (CR)).

La manera como se usa el comando "C" y el "R" para poder crear una insttrucción BRA en la localidad $1AD y $1AE con destino a la localidad $1F8 es como sigue: Primero se ejecuta el comando "C" con el paramentro $1AD y se coloca la instrucción BRA, entonces (CR) (retorno de carro) y obtengo la localidad $1AE y establezco cualquier "offset" positivo; entonces con el comando "R" observo a que localidad estoy mandando el programa y después de esto si es necesario cambio el "offset" y ejecutando el comando "R" hasta obtener la localidad deseada (en este caso $1F8).

El comando "K" es muy útil en realización de operaciones como suma de varios números hexadecimales, ya sean positivos o negativos. también es muy útil porque coloca el valor resultante de la suma en el parámetro de INICIO y así poder usar el comando "G" sin necesidad de recordar el parametro, esto viene después de haber usado el comando "N", ya que podemos sumar el parámetro usado en el comando "N" y su propio comando de paso , así con el comando "K" tenemos un punto de INICIO bastante acertado.

CONTAR... Este programa cuenta el número de veces que existe un par de ceros y/o de unos en el registro A, al iniciar el programa.

```
LINEA   ETIQUETA  COD OP   LENG ENSM      COMENTARIOS
                           NAM  CONTAR
                           OPT  O,S
                           ORG  $2000
        INICIO            PSHA           Almacena valor del reg A
                                         para al final recobrarlo.

                          CLRB           Borra contador de pares
                                         de 1's.

                          LDX  # 4       Coloca como si existieran
                                         4 veces pares de 0's

        LOOP              ASLA           Este Loop hace lo siguiente
                          BVS  1F        Mueve a la izquierda el
                          BCC  2F        contenido de AR y checa si
                          INCB           existe sobreflujo. si esto
        1H                DEX            ocurre significa que hay un
        2H                ASLA           1 y un 0. Después checa
                          BNE  LOOP      por carry en cero, si esto
                                         ocurre, hay un par de ceros.
                                         de otra manera significa que
                                         hay un par de unos y que in-
                                         crementará el contador de
                                         1'S (reg B) y decrementará
                                         el de ceros (reg X). Cuando
                                         el registro A es cero enton-
                                         ces el análisis ha terminado

                          PULA           Recobra el valor inicial de
                                         el registro A.

                          SWI            Interupción de Programa.
```

MEMORIA AFECTADA POR EL PROGRAMA:

MEMORIA RAM SE AFECTAN DESDE LA $2000 a la $2011 de programa.

Resultado final de los registros:

AR/INICIAL	BR/DEPENDE DE INICIAL DE AR	XR/DEPENDE DE INICIAL DE AR
PC/$0212	SP/INICIAL	CC/$04

H	I	N	Z	V	C
'	'	0	1	0	0

LIMITACIONES:

Depende del programa del alumno, este programa en particular no tiene limitaciones.

5.86.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Proceso Practico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 5.84 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 20 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

* Reporte del contenido final de los registros 10 puntos.

* Reporte de las limitaciones 5 puntos.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....55 puntos
Punto (b)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.9 :

ANALISIS DE MEMORIA

5.91.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico Binario.
  Sistema Numérico BCD.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  Manejo del registro índice

  Perfecto conocimiento de las instrucciones de brinco.

* CUESTIONARIO PREVIO

  NOTA: Las Respuestas están a doble densidad.
  Asegúrese de identificar las respuestas.

Existe una serie de Preguntas dentro de la sección de Proceso Práctico (5.94), cuando se realize la práctica y con los conocimientos adquiridos por el alumno se deberán ir contestando y se entregarán en el reporte de la práctica.

5.92.- DISCUSION PRELIMINAR

El principal propósito de este ejercicio de laboratorio es introducirte a algunas técnicas elementales de instrucciones indexadas y de brinco. Antes de presentarte al laboratorio deberás escribir un programa que procese una lista de números como se especifica abajo, y entonces deverás usar MUDBUG y un microprocesador en el laboratorio para demostrar tu programa y si es necesario limpiarlo de errores "debug".

Tu programa deberá examinar cada localidad en una lista de números para determinar si la localidad contiene un valor impar, un valor par negativo, un valor par positivo, o cero. Si un valor de la lista es originalmente impar, invierta cada bit del valor y almacene el valor invertido en la misma localidad. Si un valor de la lista es originalmente par y negativo, divida el valor entre 2. Si un valor de la lista es par y positivo (mayor que cero), coloque el mismo valor pero negativo. Finalmente, si el valor de la lista es cero, cámbielo a -1. La siguiente tabla muestra como debe de desarrollarse tu programa para una lista de valores elegidos aleatoriamente.

ANTES DE EJECUCCION	CLASIFICACION	DESPUES DE EJECUCCION	CAMBIO REALIZADO
$00	Cero	$FF	Cambio a -1
$99	Impar	$66	Invierte
$A4	Par negativo	$D2	Divide en 2
$7E	Par positivo	$82	Negación
$7F	Impar	$80	Invierte
$08	Par positivo	$F8	Negación
$80	Par negativo	$C0	Divide en 2

Cuando tu programa ha terminado de procesar todos los valores en la lista, deberá detenerse con la instrucción SWI. La instrucción SWI no detiene totalmente al microprocesador, sin embargo, actúa efectivamente como una instrucción de alto "HALT" regresando el control a MUDBUG.

Tu programa deberá procesar una lista de 25 números que están en RAM de la localidad $F2 a la $10A. Tu puedes usar las localidade RAM de la $00 a la $1F para almacenamiento temporal, y también podras usar el STACK. Todas las otras localidades de RAM (especialmente de la $F1 a la $10B) deberán permanecer sin alteraciones por la operación de tu programa.

Asume que tienes tu programa iniciando en la localidad $2000, y escribe tu programa de tal manera que pueda correr en una area ROM. Como de costumbre deberás hacer tu programa re-arrancable, así lo prodrás correr una y otra vez sin necesidad de recargarlo o de re-inicializarlo.

Una vez que hayas escrito tu programa en lenguaje ensamblador, manualmente conviertelo a código máquina como lo haría un ensamblador. Escribe la versión hexadecimal del código-máquina de tu programa a la izquierda de tus instrucciones en lenguaje ensamblador, y también incluye una columna indicando la dirección de memoria de la primera palabra de cada instrucción.

5.93.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, F, G, H e I.

Se puede encontrar información adicional en la bibliografía con referencia 1, y 4.

5.94.- PROCESO PRACTICO

Cuando te presentes al laboratorio, dirígete a la estación de microprocesador e introduce tu programa a la memoria usando el comando "C" de MUDBUG. Después de que tu programa ha sido cargado, genera un listado de memoria para verificar que realmente tu programa ha sido cargado correctamente. Y entonces podras probar tu programa como se indica a continuación:

Pon algunos valores en tus localidades asignadas para la lista y genera un listado para grabar esos valores en un formato sencillo de leer (apuntalos en un papel si es que no estas usando una terminal-impresora). Entonces corre tu programa, y genera otro listado después de haber terminado la ejecución de tu programa. De nuevo, usa los comandos de MUDBUG para generar tu listado de memoria y checar que tu programa ha trabajado correctamente. Asegúrate de checar varias localidades fuera de las localidades asignadas (antes y después de la lista) y así estarás completamente seguro de que tu programa no altera localidades extra. Repite esta operación varias veces con una lista de valores inteligentemente seleccionados y demuestra la correcta operación de tu programa.

Ya que tu programa sera corrido varias veces, tu encontrarás que es conveniente que tu programa sea automaticamente re-arrancable vía el comando "H" de MUDBUG. Tu puedes hacer esta mejora simplemente mediante el uso del comando "C" y el "R" para añadir la instrucción "BRA INICIO", siendo esta parte de tu programa. La instrucción "BRA INICIO" es solamente una ayuda para pruebas.

Si tu programa no trabaja correctamente, deberás usar MUDBUG para corregirlo. En particular, encontrarás que cualquier error en tu programa puede ser facilmente separado mediante el uso del comando "T". Aun si tu programa corre perfectamente desde el principio (y lo hara si eres cuidadoso), deberás usar el comando "T" con tu programa para mejorar y demostrar tu habilidad en corrección de errores. Deberás además practicar con otros comandos de MUDBUG hasta que los uses con maestría. Se recomienda que uses el comando "R" especialmente.

Para esta práctica entrega lo siguiente:

1.- La versión totalmente documentada de tu programa en lenguaje ensamblador y su traducción en lenguaje máquina. Incluye todos los comandos necesarios ORG, EQU, RMB y END.

2.- Como parte de tu documentación, explica brevemente como funciona tu programa, y documenta el estado final de los registros, el código de condiciones, y las localidades de memoria afectadas. También presenta características relevantes de tu programa, y muestra claramente como sería tu programa para trabajar en un listado general

de N valores; donde N va de 1 a 50,000. Tu necesitas variar solamente una ecuación para ejecutar tu programa en cualquier rango de valores dado.

La solución a este programa tomo $13 Bytes (19 decimal) de memoria ROM y no uso ninguna localidad RAM.

5.95.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

ANALIZADOR DE MEMORIA

Nombre del alumno
No. de Credencial
Materia
Lab 9
Fecha

ANALIZADOR DE MEMORIA.... Este programa analiza y procesa un listado de memoria de acuerdo a las condiciones establecidas bajo cada valor. Modifica y regresa a la misma localidad de memoria el nuevo valor.

LINEA	ETIQUETA	COD OP	LENG	ENSM	COMENTARIOS
		NAM	ANMEMOR		
		OPT	O,S		
	LISTA	EQU	25		
	LIST	EQU	$F2		
		ORG	$2000		
	INICIO	LDX	#LISTA		Carga el reg XR con el número de valores a analizar.
	LOOP	LDAA	LIST,X		Carga valores de la lista.
		ASR	LIST,X		Recorre a la derecha y ejecuta para pares negativos
		BLS	1F		Checa por impares o ceros.
		BMI	2F		Checa por pares negativos.
		DECA			Realiza operación para pares positivos.
	1H	COMA			Realiza operación para impares y ceros.
		STAA	LIST,X		Regresa el valor a su localidad de memoria.
	2H	DEX			Decrementa XR y cuando llegue a cero es que ha teminado de analizar la lista.

```
BNE   LOOP    Checa si XR es cero.
              y si no lo es, regresa para
              analizar el siguiente valor.

SWI           Interrupción de programa.
```

Cambiando los valores iniciales en LISTA y LIST, se puede hacer el rango como se desée.

MEMORIA AFECTADA POR EL PROGRAMA:

MEMORIA RAM SE AFECTAN DESDE LA $2000 a la $2012 de programa.

Resultado final de los registros:

AR/DEPENDE DE LISTADO	BR/INICIAL	XR/$00
PC/$0213	SP/INICIAL	CC/$04

H I N Z V C
' ' 0 1 0 0

LIMITACIONES:

Depende del programa del alumno, este programa en particular no tiene limitaciones.

5.96.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Proceso Práctico: Se otorgarán 55 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 5.94 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 20 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

* Reporte del contenido final de los registros 10 puntos.

* Reporte de las limitaciones 5 puntos.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....55 puntos
Punto (b)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

VI.- PROGRAMACION AVANZADA

VI.- PROGRAMACION AVANZADA

INTRODUCCION

Este capítulo cubre considerablemente la programación de componentes de entrada/salida, lo cual es una parte escencial para las aplicaciones de microporcesadores, siendo estas las de controlador y procesador de datos. Además en este capítulo se cubrirá aspectos relevantes del ACIA y del PIA mostrando a detalle sus modos de operación, entre los que se destaca sus modos de operación en tiempo real. Además trataré de establecer las bases para comprender el funcionamiento de los sistemas operativos, lógistica y operación.

El capítulo consta de nueve prácticas. Las prácticas guían al estudiante desde los inicios y familiarización con el equipo periférico hasta obtener la habilidad para desarrollar proyectos con alto grado de dificultad, y considerando la elaboración de sistemas operativos de una importancia especial.

DIVISION DE PRACTICAS

Las prácticas están subdivididas en secciones. Estas secciones son :

+ Consideraciones Básicas
+ Discusión Preliminar
+ Información Adicional
+ Proceso Práctico
+ Reporte
+ Evaluación de la Práctica

Como se describe en la introducción del capítulo 6 el instructor de laboratorio considerará el material que sea necesario entregar al alumno. Se sugiere que le sea entregado lo siguiente : Lo referente a a) Consideraciones Básicas, b) Discusión Preliminar, y c) Proceso Práctico. La Información Adicional tendrá que ser seleccionada por el instructor ya que esta información podría contener el resultado de la práctica y el objetivo de la información es tener un banco de información que pueda ayudar al alumno a realizar su práctica adecuadamente y no obtenerla ya elaborada.

Considere los aspectos indicados en la introducción del capítulo anterior válidos para este capítulo.

PROGRAMACION AVANZADA

PRACTICA No.1 :

CONVERSION DE BASE

6.11.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico desde BASE 2 a BASE 10

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  Conocimiento de los sitemas numéricos y sus conversiones.

  Conocimiento de subrutinas del sistema operativo MUDBUG.

* CUESTIONARIO PREVIO

  NOTA: Las Respuestas están a doble densidad.
  Asegúrese de identificar las respuestas.

1.- realice las siguientes conversiones, llenando los espacios vacios.

BASE2	BASE3	BASE4	BASE5	BASE8	BASE10	BASE16
101010	1120	222	132	52	42	2A
100010001	101010	10101	2043	421	273	111
10001000100	1111110	101010	13332	2104	1092	444
110010110111	11110120	302313	101010	6267	3255	CB7
1111110010	1101102	33302	13020	1762	1010	3F2
1000000010000	12122022	1000100	112422	10020	4112	1010

2.- Haga las conversiones necesarias y llene los espacios vacios. Todas las conversiones deben ser exactas, no redondear.

BASE2	BASE4	BASE8	BASE10	BASE16
11011011.0110101	3123.1222	333,324	219.4140625	DB.6A
11111010.10101	3322.222	372,52	250.65625	FA.A8
10011100.1011000010	2130.230022	274.5412	156.68994140625	9C.B0A
1101110011.10101101	31303.2231	1563.532	883.67578125	373.AD
11101101.10101100111	3231.223032	355.5316	237.67529296875	ED.ACE

6.12.- DISCUSION PRELIMINAR

Este laboratorio te dará la oportunidad de aplicar tu conocimiento de las relaciones entre números en diferentes bases. Antes de venir al laboratorio, deberás escribir un programa que convierte un número hexadecimal a cualquier base especificada en el rango de 2 a 10. Entonces usarás el sitema operativo MUDBUG y una estación de microcomputadora en el laboratorio para realizar correcciones, depurarlo y demostrarlo.

Brevemente, tu programa leera un valor de 16 bits hexadecimal desde la terminal, convertira el valor de entrada como un entero sin signo a una base que fue especificada por el valor inicial del registro A, y sacar por la terminal el valor convertido.

Cuando tu programa recive el control, el registro AR contiene el parámetro que indica la base destino. Si el valor inicial de AR no es válido, que no se encuentre dentro de rango de 2 a 10, tu programa deberí desplegar una diagonal "\" y hacer sonar una campana en la terminal y después regresar el control a MUDBUG. Si el valor inicial de AR es válido, por el otro lado, tu programa deberá mover el cursor al principio de una nueva linea y entonces esperar para que el usuario introduzca el número hexadecimal que queremos convertir.

Ahora el usuario puede introducir el número que deseamos convertir. El número de entrada deberá ser un número en formato estandar de MUDBUG, y con terminación de MUDBUG estandar. Si el usuario teclea un caracter de terminación sin valor alguno, tu programa deberá usar cero como valor. Si el usuario teclea "/" para finalizar con la introducción del dato, el programa deberá regresar al control de MUDBUG. De otra manera deberá continuar como sigue abajo:

Una vez que has recivido el valor inicial, el programa deberá desplegar un signo igual "=" en la terminal. Entónces tu programa deberí convertir el valor de entrada (entero sin signo) a la base especificada y desplegar el valor convertido en la terminal. El valor de cero deberá ser desplegado siempre con un solo cero., y cualquier otro valor deberá ser mostrado sin signo y sin ceros antes del valor significativo.

Después de desplegar el valor convertido, tu programa deberá avanzar el cursor al inicio de una nueva linea y regresar a la rutina en la cual se pide el valor inicial, para otro número. Por lo tanto tu programa convertirá tantos números sean introducidos a la base especificada, hasta que el usuario teclee un caracter de terminación de ejecución "\".

El siguiente ejemplo ayudará para clarificar las especificaciones de este programa. Ilustra una ejecucción típica del programa:

```
>A 00 0A
>G 2000.
FFFF.=65535
-1.=65535
8000.=32768
10.=16
.=0
64.=100
2710.=10000
2711.=10001
-3CB0.=50000
C355.=50005
FF.=255
80.=128
/
```

El manual de MUDBUG contiene algunas rutinas internas que será de mucha utilidad para ti, y se permite que las uses para esta práctica. Por ejemplo: Puedes considerar de utilidad la subrutina CRLF, la rutina de ERROR, subrutina NEWVAL, subrutina NUMBI, y/o la subrutina OUTCHR. Puedes usar ecuaciones para definir los puntos de entrada para las rutinas de MUDBUG como "offsets" desde el principio y en base al ROM de MUDBUG, y entonces tu programa usará las rutinas como se especifica en el manual.

Asume que tienes ROM iniciando desde la localidad $2000, y escribe tu programa de tal manera que pueda correr en el area de ROM. Puedes usar RAM para almacenamiento temporal y si tu programa lo requiere usa también el STACK. Como es usual, tu programa deberá ser re-arrancable de manera que puedas arrancarlo una y otra vez sin necesidad de cargar de nuevo el programa o inicializar alguna localidad de memoria (excepto AR). Naturalmente tu programa deberá preservar el valor del STACK POINTER, y los otros registros podrán modificarse como se requiera.

Una vez que hayas escrito tu programa en lenguaje ensamblador, manualmente convierte lo a lenguaje máquina como lo haría un ensamblador. Escribe la versión de lenguaje máquina a la izquierda de la versión en lenguaje ensamblador y además incluye una columna con las direcciones de las localidades correspondientes. Asegúrate de incluir las instrucciones como EQU, RMB, ORG que sean necesarios para que tu programa sea completo y flexible.

6.13.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, F, G, H e I.

6.14.- PROCESO PRACTICO

Cuando te presentes al laboratorio, dirígete a la estación de microprocesador e introduce tu programa a la memoria usando el comando "C" de MUDBUG. Después de que tu programa ha sido cargado, genera un listado de memoria para verificar que realmente tu programa ha sido cargado correctamente. Y entonces podrás probar tu programa como se indica a continuación: Pruébalo con diferentes entradas para verificar que el programa responde correctamente a todas las posibles entradas y con todas las bases destino especificadas. Asegúrate de que esto sea cierto antes de demostrarlo a tu instructor de laboratorio y que el detecte los errores.

Para esta práctica entrega lo siguiente:

1.- La versión totalmente documentada de tu programa en lenguaje ensamblador y su traducción en lenguaje máquina. Incluye todos los comandos necesarios ORG, EQU, RMB y END.

2.- Como parte de tu documentación, explica brevemente como funciona tu programa, y documenta el estado final de los registros, el código de condiciones, y las localidades de memoria afectadas.

La solución a este programa tomo $4D Bytes (77 decimal) de memoria ROM y RAM, y uso del STACK.

6.15.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

CONVERTIDOR DE BASE

Nombre del alumno
No. de Credencial
Materia
Lab 1
Fecha

CONVERTIDO DE BASE... Este programa convierte un valor inicial hexadecimal a un valor en una base especificada por el registro AR.

```
LINEA   ETIQUETA  COD OP   LENG ENSM     COMENTARIOS
                           NAM  CONVERT
                           OPT  O,S
        CERO               EQU  $0000
                           ORG  $0000
        MENOR              RMB    1
                           ORG  $000A
        MEMAR              RMB    1
                           ORG  $2000
        INICIO             CMPA =$02     Compara la base que esta en
                           BLT  1AD      AR. La base debe estar en el
                           CMPA =$0A     rango de 2 a 10. Si no cum-
                           BLE  2F       ple entonces el programa
        1AD                JMP  ERROR    imprime una diagonal. y re-
                                         regresa el control a MUDBUG
                                         Se ayuda por la subrutina
                                         de MUDBUG ERROR.

        2F                 STAA MEMAR    Almacena la base de AR en
                                         la localida $0A. porque
                                         algunas rutinas de MUDBUG
                                         afectan al registro AR.

        3F                 JSR  CRLF     Mueve el cursor al inicio
                                         de la siguiente linea.

                           LDX  =CERO    Inicializa el Registro ín-
                           CLR  0,X      dice y localidades de meмo-
                           CLR  1,X      ria que serán usadas para
                                         almacenar el valor de
                                         entrada.
```

```
        JSR NEWVAL      Subrutina que accepta 4
                        valores hexadecimales de la
                        terminal y los almacena en
                        la dirección indicada por el
                        el registro índice. También
                        carga el registro AR con el
                        Byte de mayor significancia
                        y BR con el de menor.

        CLRB            Borra el registro BR porque
                        sera usado como contado.
                        Contará las veces que el
                        programa use el Apuntador
                        de Stack.

        LDAA  #"="      Estos 2 pasos son para des-
        JSR   OUTCHR    plegar el símbolo "=" en
                        la terminal ya que la subru-
                        tina OUTCHR despliega carac-
                        teres en la terminal.

4F      LDAA  MENOR     Carga el byte de menor
                        significancia en el registro
                        AR.
                        Incrementa el contador y
1AT     INX             resta hasta que el byte
        SUBA MEMAR      LSB sea menor que la base
        BCC  1AT        apuntada en AR, actualmente
                        en la localidad MEMAR.

        TEST CERO       Checa que el byte MSB del
        BEQ  5AD        valor almacenado en $0000
        DEC  CERO       sea cero o no. Si es cero,
        BRA  1AT        brinca a la rutina de
                        almacenamiento. Por otro
                        lado, decrementará en uno
                        y regresa a 1 ATras para
                        comparar de nuevo.

5AD     ADDA  MEMAR     Para recobrar el valor in-
        PSHA            mediato menor que la base
        INCB            obtenida por la substrac-
                        ción, el progama añade la
                        base al LSB. Almacena el
                        valor en el STACK e incre-
                        menta en uno el número de
                        veces que el STACK es usa-
                        do ayudándose del BR.

        DEX             Decrementa en uno el conta-
        BEQ 6AD         dor, si es cero va a la
                        rutina de despliege.
```

```
        STX  CERO      Esta rutina pone el valor
        LDX =CERO      del contador como un nuevo
        BRA 4F         valor en las localidades de
                       memoria CERO.  Borra el
                       contador y regresa al inicio
                       para convertir otro número.

6AD     PULA           Rutina para desplegar. En
        ADDA  =$30     esta rutina el programa to-
        JSR OUTCHR     ma el contenido del STACK
        DECB           en AR y $30 es sumado para
        BNE  6AD       ajustar el valor al código
                       ASCII y entonces desplegar.

        BRA  3B        Regresa a alimentación de
                       linea y espera por el si-
                       guiente valor.

        END
```

Cambiando los valores iniciales en LISTA y LIST, se puede hacer el rango como se desee.

MEMORIA AFECTADA POR EL PROGRAMA:

MEMORIA RAM SE AFECTAN DESDE LA $2000 a la $2012 de programa.

Resultado final de los registros:

AR/DEPENDE DE USUARIO	BR/VARIABLE	XR/VARIABLE
PC/$200D	SP/INICIAL	CC/$04 H I N Z V C ' ' 0 1 0 0

LIMITACIONES:

Depende del programa del alumno, este programa en particular no tiene limitaciones.

6.16.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 5 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.14 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 20 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

* Reporte del contenido final de los registros 10 puntos.

* Reporte de las limitaciones 5 puntos.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.2 :

PRINCIPIOS DEL FUNCIONAMIENTO DEL ACIA
ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER

6.21.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.
  Osciloscopio

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.
  Resolución del Osciloscopio. No relevante.

* CONOCIMIENTOS REQUERIDOS

  El ACIA MC68050
  - ESTRUCTURA
  - CARACTERISTICAS
  - FUNCIONAMIENTO
  - PROGRAMACION

  USO DEL OSCILOSCOPIO

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

1.- Cual es la definición básica del ACIA.

El ACIA actua como una interfase entre un componente serial y la computadora. Provee un formato para los datos y control al periférico.

2.- Enliste las características del ACIA MC68050

* 8- y 9-bits de transmisión
* Pariedad Par y Non opcional
* Chequeo de Pariedad, error de "overrun" y error de "framing"
* Registro de control programable
* Reloj con opciones de 1, 1/16, 1/64
* Transmisión de 1.0 Mbps
* Borrado de bit en arranque en falso
* Funciones para controlar periféricos/modems
* Doble "buffer"
* Operación con uno o dos bits de alto.

3.- Dibuje el chip MC68050 con su asignación a pins.

4.- Explique la función de Entrada/Salida de las lineas "Chip select" CS0, CS1, $\overline{CS2}$.

Estas 3 lineas de entrada son de alta impedancia compatibles con TTL y son usadas para direccionar al ACIA.

El ACIA es seleccionado cuando CS0 y CS1 están en alto "1" y $\overline{CS2}$ es bajo "0".

La transferencia de datos a través del ACIA puede ser desarrollada bajo el control de la señal "enable (E)", "Read/Write (R/$\overline{W}$)" y "Register Select (RS)".

ENABLE.....Habilitador
READ/WRITE.....Leer/Escribir
REGISTER SELECT.....Seleccionador de Registro

5.- Indique que es posible controlar al programar el registro de control.

* Longitud variable de la palabra (8- o 9-bits)
* Modo de reloj (1 , 1/16, 1/64)
* Control de transmisión
* Control de recepción
* Control de interrupción.

6.22.- DISCUSION PRELIMINAR

Esta práctica tiene como finalidad dar al alumno la oportunidad de demostrar su conocimiento del ACIA "Asynchronous Communications Interface Adapter".

Una computadora, para desarrollar una función útil, debe ser capaz de comunicarse con el exterior. Esto es, a través de componentes externos, como son : teclados, impresoras, teminal de video, otras computadoras, etc...

Dos formas de transferencia de información son posibles: serie y paralelo. En forma paralela se requieren de 8-bits transmitidos al mismo tiempo, pero además de las 8 lineas por cada bit se requieren lineas de control. Para transmisión de datos de mas de 30 metros este tipo de transmisión resulta inpráctico. En tales casos, la transmisión en serie es preferible, donde solo una linea es requerida.

El ACIA actúa como interfase entre un componente serie y la computadora. comunicandose con el componente serie en formato serie y con la computadora en formato paralelo.

Direccionamiento simbólico es preferible cuando se trabaja con el ACIA. Por ejemplo: La sentencia... SERBUF EQU $7FF5

dirige al ensamblador a sustituir 7FF5 por la dirección simbólica SERBUF donde quiera que aparezca dentro del programa.

Para cuestiones de esta práctica y en aquellas donde se use el ACIA, nombraremos "SERBUF" a la localidad de memoria que sirve como "buffer" de datos del ACIA, y llamaremos "SERCSR" a la localidad de memoria donde tenemos el control y el registro de estados, donde viene "SERCSR" (Serial Control and Status Register). Le corresponde al alumno investigar que direcciones tienen acceso al ACIA. Este, puede preguntar al instructor de laboratorio. En el diagrama 1 presentado en el apéndice J , se podrá apreciar que el ACIA es seleccionado a través de CS2, hoja 2, E8, y en la hoja 1, A7, se aprecia el enlace con la leyenda "ACIA DIS" . El chip 16, un 74LS138 CORRESPONDE A UN DECODIFICADOR /demultiplexor 1 a 8. Pues bien, en la hoja 1, A8, las entradas al chip 16 ABC vienen del bus de direccionamiento del MPU, pins A13, A14, y A15. Ya que se ha establecido el canal de selección del MPU con el ACIA, es necesario saber bajo que condiciones y que dirección hará que el ACIA sea seleccionado. La combinación en ABC = 011 hará que Y3 sea seleccionado y por lo tanto el ACIA. Esta combinación corresponde a A15 = 0, A14 = 1 y A13 = 1 lo que nos daría la dirección $6000. Y para saber si nos vamos a comunicar con SERBUF o SERCSR checamos la hoja 2 en E8 y encontramos el seleccionador de registros RS, el cual esta conectado al bus de direccionamiento del MPU

por A0. Entonces escogeremos para SERCSR la dirección $6000 y para SERBUF la dirección $6001.

Como hemos visto estas direcciones estan dadas por el hardware del sistema, y cualquier kit puede llegar a tener modificaciones, es por eso que se les recomienda preguntar a su instructor de laboratorio las direcciones usadas por el ACIA del kit que se vaya a usar.

Por ahora el alumno conectará a la salida del ACIA del kit D2 un osciloscopio para observar las diferentes modalidades de transmisión.

Ademas realizara un programa en el que continuamente este enviando una señal, variando la configuracion de salida, es decir, modificando el numero de bits de datos, la velocidad y el tipo de paridad.

Incluya en su reporte:

1.- Un diagrama de conecciones del kit D2 a el osciloscopio.

2.- Su programa a nivel ensamblador y a nivel lenguaje máquina perfectamente documentado.

3.- Una tabla con los posibles modos de transmición y un dibujo de la señal por cada modo de transmición.

6.23.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, D, F, G, H, I y J.

Se puede encontrar información adicional en la bibliografía con referencia 1, 2, 5, 8 y 9.

6.24.- PROCESO PRACTICO

El alumno conectará el kit D2 a un osciloscopio para poder observar la salida. Se requiere conocimiento del funcionamiento del osciloscopio para hacer las conecciones correspondientes y los ajustes necesarios par poder observar la señal.

Si el laboratorio contara con un Analizador Digital, el proceso de esta práctica sería logrado al 100%, ya que la actividad sería observable con mayor facilidad.

Introduzca el programa el Kit D2 y proceda a correr el programa checando la señal de salida, vaya paso a paso chacando cada modo posible y dibujando la salida, mostrando el bit de arranque, el bit de pariedad y el o los bits de alto, así como los bits de datos.

Demuestre al instructor de laboratorio sus resultados sus resultados, y si tiene alguna duda pregunte a su instructor y presente además cualquier punto que usted crea de interes en la realización de la práctica.

6.25.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

PRINCIPIOS DE FUNCIONAMIENTO
DEL ACIA

Nombre del alumno
No. de Credencial
Materia
Lab 2
Fecha

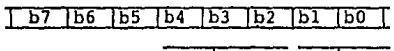
1.- Diagrama de conexiones:

PROGRAMA: TRANSMISION DE DATOS USANDO EL ACIA.

LINEA	ETIQUETA	COD OP	LENG ENSM	COMENTARIOS
			NAM TRANSMT	
			OPT O,S	
			ORG $0200	
	SEÑAL		EQU X	Señal Arbitraria
	SERCSR		EQU $6000	Direccionamiento simbólico
	SERBUF		EQU $6001	localidades del buffer y
	INICIO		LDAA =3	Reset Maestro
			STAA SERCSR	
			CLRA	Inicializa registros y carga
			LDAB =SEÑAL	valor de la señal en el registro B. En el registro A se selecciona el modo 1.
	LOOP		BSR TRANS	Brinca a subrutina de transmisión de datos.
			INCA	Incrementa uno al registro
			CMPA =$20	A para seleccionar el siguiente modo de transmisión comparando antes si ha sido ya analizado el último modo. Los modos de interes se hacen variar en los primeros 5 bits del registro de control, es por eso que al comparar el bit 6 y sea verdadero, entonces habremos analizado ya todos los modos posibles.
			BNE LOOP	
			SWI	Interrupción.
	SUBRUTINA TRANS		STAA SERCSR	Al registro de control donde se le indica el modo de transmisión.
			STAB SERBUF	La señal al buffer de datos
	LP1		LDAA SERCSR	Se checa el registro de
			BITA =2	estado para el bit TDRE que
			BEQ LP1	indica que el registro de transmisión de datos esta vacio. La lines CTS controla este bit.

RTS Regreso a subrutina.

3.- Registro de control:



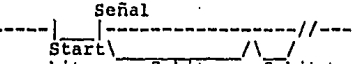
Longitud de palabra, pariedad y selector de bit de alto.

Divisor de Señal de reloj

b1	b0	funcion
0	0	/1
0	1	/16
1	0	/64
1	1	Reset Maestro

b4	b3	b2	long. + Pariedad + bit de alto
0	0	0	7 Par 2
0	0	1	7 Non 2
0	1	0	7 par 1
0	1	1	7 Non 1
1	0	0	8 Ninguna 2
1	0	1	8 Ninguna 1
1	1	0	8 par 1
1	1	1	8 Non 1

MODO	b4	b3	b2	b1	b0
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
:	:	:	:	:	:
31	1	1	1	1	1



Reset Maestro

6.26.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.24 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Punto 1 Diagrama de Conecciones del Osciloscopio acreditará 15 puntos.

* Punto 2 Lenguaje ensamblador y código máquina 15 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Punto 3 Representación del registro de control con sus posibles funciones y diagramas de señales acreditará 15 puntos.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.3 :
PROGRAMACION DEL ACIA

6.31.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.
  Impresora

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  El ACIA MC68050
  - ESTRUCTURA
  - CARACTERISTICAS
  - FUNCIONAMIENTO
  - PROGRAMACION

  USO DE IMPRESORA

* CUESTIONARIO PREVIO

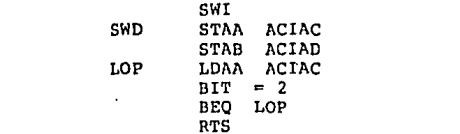
NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

dado un sistema MC6800 con ACIA en direcciones ACIAC (control) y ACIAD (datos), anteriormente llamdos SERCSR y SERBUF respectivamente, con el reloj de entrada de recepción y transmisión conectados a 19,200 Hz. Defina los patrones (por bits) de salida producidos por los datos transmitidos incluyendo todos los bits añadidos cuando la siguiente secuencia de instruccciones es ejecutada:

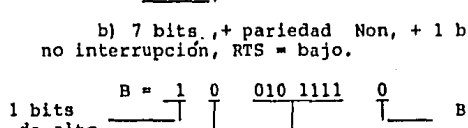
```
TEST   LDAA  = 3
       STAA ACIAC
       CLRA
       LDAB =%1011111
       BSR  SWD
       LDAA =%1101
       BSR  SWD
       LDAA =%11010
       BSR  SWD
       SWI
SWD    STAA  ACIAC
       STAB  ACIAD
LOP    LDAA  ACIAC
       BIT  = 2
       BEQ  LOP
       RTS
```

Defina la configuración de bits producido por la salida.

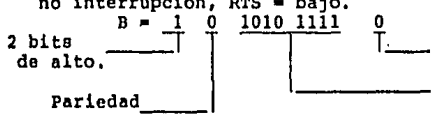
a) 7 bits, + pariedad Par, + 2 bit de alto, reloj dividido entre 1, no interrupción y RTS = "bajo".



b) 7 bits ,+ pariedad Non, + 1 bit de alto, reloj /16, no interrupción, RTS = bajo.



c) 8 bits, + pariedad Par, + 1 bit de alto, reloj /64, no interrupción, RTS = bajo.



2.- Calcule el "Baud Rate" y el "Character Rate" para cada salida transmitida.

	BAUD RATE	CHARACTER RATE
A)	19.2 K	1746 Char/sec
B)	1.2 K	120 Char/sec
C)	300	27 Char/sec

3.- Cual es el máximo Baud Rate que puede ser generado por el ACIA.

Mas de 1.0 MHz.

4.- Cual es la primera cosa que tiene que hacerse para inicializar el ACIA.

El Reset Maestro, esto se logra colocando 1's en los bits 0 y 1 del control del ACIA. Se puede usar las siguientes instrucciones:

```
LDAA  = 3
STAA   ACIAC
```

5.- Explique la función de la linea serial de entrada/salida CTS ("Clear To Send).

Provée control automatico al fin de transmisión de una via de comunicaciones encadenada. Por medio de un MODEM, CTS activo cuando transmisión en bajo, inhibiendo el "registro de transmisión de Datos vacio (TDRE)".

6.32.- DISCUSION PRELIMINAR

Como es sabido, procesamiento de datos serial por el ACIA sigue escencialmente las especificaciones del RS-232-C, dictadas por la Asociación de Industrias Electrónicas (EIA). Niveles de voltaje, fuente y resistencias de carga, tipo conector, asignación de pins para señales de datos y de control están contenidas en estas especificaciones. Algunas de estas señales de control son producidas por el ACIA para el componente serial y otras por el componente serial para el ACIA.

Una señal de control es RTS ("Request to Send" Requerimiento para envio), la cual es producida por el ACIA cuando requiere permiso del componente serial, por ejemplo un impresor, -para enviarle datos-. Esta señal es activa cuando "0", por lo tanto se nombra RTS testada, la barra indicaría testada o inversión y así RTS = 0 y RTS testada = 1. RTS es determinado por los bits 5 y 6 del registro de Control.

La respuesta usual por el componente serial (impresor) al recibir RTS testado = 0 es para activar una linea de control a el ACIA llamada CTS testada (Clear To Send), también actua cuando "0".

Este intercambio de señales de control, usualmente preceden a la transmisión de datos, y es frecuentemente llamado "Hand Shacking" y puede ser usado para permitir transferencia de datos solo cuando el componente esta encendido y en estado operacional.

Pues bien, La práctica consiste en elaborar un programa para enviar un mensaje a una impresora serial. El mensaje será a la elección del alumno, pero el ACIA deberá ser programado con transmision de 7 bits, pariedad Non y 2 stop bits. Data Rate de 600 bps. (para este caso tomamos la frecuencia del oscilador igual a 38400 bps).

Se requiere hacer las conexiones correctas entre el ACIA del kit D2 y el impresor. Consulte a su instructor para cualquier aclaración.

Inicie su programa en la localidad $2000. El programa resuelto tomo 47 bytes de memoria PSEUDORAM.

Su reporte deberá contener lo siguiente:

1.- Su programa en lenguaje ensamblador y su respectivo código máquina perfectamente documentado.

2.- El estado final del registro de Control y del registro de estados.

6.33.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, F, G, H e I.

Se puede encontrar información adicional en la bibliografía con referencia 1, 4, 5, 8 y 9.

6.34.- PROCESO PRACTICO

El alumno conectará el kit D-2 a un impresor, asegurándote de colocar los interrruptores del impresor de la manera correcta (baud rate, pariedad, etc..) y conectando los cables a las entradas y salidas respectivas. No dude en preguntar a su instructor si no estas seguro de lo que estas haciendo.

Introduce tu programa al kit D2 y procede a correrlo. Si surge cualquier anomalía anótala en tu reporte, trata de corregirlo y prueba de nuevo.

Demuestra a tu instructor de laboratorio los resultados obtenidos.

6.35.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

PROGRAMACION DEL ACIA

Nombre del alumno
No. de Credencial
Materia
Lab 3
Fecha

PROGRAMA: IMPRESION DE DATOS USANDO EL ACIA.

```
LINEA   ETIQUETA   COD OP   LENG ENSM      COMENTARIOS
                            NAM  IMPRE
                            OPT  O,S
                            ORG  $2000
            INICIO          LDAA  = 3      Reset maestro del ACIA.
                            STAA   ACIAC

                            LDAA  =%110    Programa el registro de
                            STAA   ACIAC   control del ACIA con 7 bits
                                           pariedad Non,2 bits de alto
                                           y reloj/64.

                            LDX  =MENS-1   Inicia la dirección del
                            STX  MENSDIR   mensaje que se desea enviar

            MAS             LDX  MENSDIR   Actualiza la dirección del
                            INX            mensaje durante cada ciclo.
                            STX  MENSDIR

                            LDAA  0,X      Toma el caracter de memoria
                                           donde esta siendo direccio-
                                           nado por el registro índice

                            BEQ    FIN     Si el caracter es un nulo,
                                           se ha alcanzado el fin de
                                           mensaje y va a "FIN" de
                                           programa.

            LISTOTX         LDAB  ACIAC    Carga el contenido del re-
                            ANDB  =2       gistro de estados del ACIA
                                           para obtener el bit 1, es
                                           decir si el componente de
                                           salida esta listo para re-
                                           cibir la señal. El bit que
```

```
                                es checado es el "transmi-
                                sión lista".

            BEQ   LISTOTX       Si no esta lista la trans-
                                misión, el componente a
                                recibir no esta listo, en-
                                tonces repite la rutina
                                y checa hasta que lo este.

            STAA  ACIAB         Transmite un caracter a la
                                impresora.

            BRA   MAS           Regresa por el siguiente
                                caracter.

MENSDIR     RMB   2             Reserva 2 bytes de memoria
                                para el apuntador de la
                                dirección del mensaje.

MENS        FCC   /HOLA/        Reserva y forma caracteres
                                constantes en las localida-
                                des de memoria sucesivas,
                                incluyendo la presente
                                con el código hexadecimal
                                correspondiente al ASCII,
                                por caracter del mensaje.

            FCB   0             Nulo, indica fin de mensaje

FIN         SWI                 Interrupción de programa.
            END
```

El registro de Control contiene lo siguiente: %0000 0110 = $06

El registro de Estados contiene lo siguiente:

El alumno sera capaz de definir la situación en la que se encuentra el registro de estados al leer el registro y compararlo con la tabla del formato de este registro.

La tabla del formato del registro de estados del ACIA la puede encontrar en el apéndice de esta tesis.

6.36.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.34 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Punto 1 Lenguaje ensamblador y código máquina 15 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Punto 2 Estado final del registro de Control acreditará 20puntos..

* Punto 3 Representacion del registro de estados con acreditará 10 puntos.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.4 :

PROGRAMACION DEL PIA
PERIPHERIAL INTERFACE ADAPTER

6.41.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  El PIA MC6821
  - ESTRUCTURA
  - CARACTERISTICAS
  - FUNCIONAMIENTO
  - PROGRAMACION

  LOCALIDADES DE MEMORIA QUE PERTENECEN AL DIRECCIONAMIENTO DEL PIA

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

1.- Indique cuantos registros tiene el PIA y cuales son:

El PIA tiene 4 registros. 2 Registros de control Programables y 2 registros de dirección de datos también programables.

2.- Indique a) cuantos pins para selección de registros tiene el PIA, b) como son estos representados y c) de una breve descripción.

a) 2

b) RS0 y RS1

c) Las 2 lineas de selección de registros son usadas para seleccionar varios registros dentro del PIA. Estas 2 lineas son usadas en conjunto con registros de control internos para seleccionar un registro en particular que va a ser escrito o leido.

Las lineas de selección de registro y de selección de "chip" deben ser estables por la duración del pulso E (enable-Habilitador) mientras dure el ciclo de lectura o escritura.

3.- Aunque el lado A y el lado B del PIA son idénticos en la mayoría de los conceptos, hay 2 diferencias principales, explique cuales son:

a) Una diferencia es la construcción interna I/O.

El lado A esta diseñado para manejar CMOS a niveles normales del 30% a 70%, e incorpora un componente interno "pull-up" que permanece conectado aun en el modo de entrada.

El lado B usa un buffer normal de 3-estados NMOS el cual puede levantar niveles de CMOS sin resistores externos. El lado B puede manejar circuitos Darlington sin ningun problema.

Cuando el PIA sale del "reset" (reestablecimiento) el puerto A representa entradas con resitencias "pull-up", y el puerto B (en su modo de entrada tambien) flotará alto y bajo (0 y 1) dependiendo de la carga a la que este conectado.

b) El lado A no tiene capacidad para manejar el modo "seguimiento de bit".

La operación de leer cuando se esta en modo de transmisión (salida de datos del PIA), Cuando se lee el puerto A, el pin actual es leido, cuando el lado B es leido viene de un ciclo de salida (output latch), adelante del pin actual.

4.- Escriba las instrucciones necesarias para asegurar que todas las lineas de datos para el puerto A serán lineas de entrada:

```
PIABFA      EQU     $7FF0       Registro Buffer
PIACRA      EQU     $7FF1       Registro de Control

            LDAA    PIACRA      Borra bit 2 para dar
            ANDA    =%11111011  Acceso a DDR y poner
            STAA    PIACRA      a puerto A como
            CLRA    PIABFA      entradas.
```

5.- Escribe las instrucciones para una rutina de reestablecimiento para colocar al puerto A como entradas y al puerto B como salidas. Esta rutina deberá dejar al PIA listo para cargar y almacenar datos.

Suponga lo siguiente:

```
PIABFA      EQU     $7FF0
PIACRA      EQU     $7FF1
PIABFB      EQU     $7FF2
PIACRB      EQU     $7FF3


LADOA       LDAA    PIACRA
            ANDA    =%11111011  Borra bit 2 para
            STAA    PIACRA      accesar al DDR

            CLR     PIABFA      Coloca puerto a para
                                entradas.

            ORAA    =%00000100  Bit 2 = 1 para datos
            STAA    PIACRA

LADOB       LDAA    PIACRB
            ANDA    =%11111011  Borra bit 2 para
            STAA    PIACRB      accesar DDR

            LDAA    =%11111111  Coloca puerto B para
                                para salidas.

            STAA    PIABFB      Toma el registro de
            LDAA    PIACRB      control y pone bit2 =1
            ORAA    =%00000100  para datos.
            STAA    PIACRB
```

6.42.- DISCUSION PRELIMINAR

En esta práctica el alumno podrá demostrar su habilidad en la programación del PIA.

El PIA, "Peripherial Interface Adapter", es un componente que puede recibir y transmitir datos en paralelo y a una velocidad no especificada.

El PIA esta compuesto de 2 secciones casi idénticas, A y B, cada una es capaz de manejar 8-bits de datos. Para cada sección hay un registro de Control (CR) y un registro buffer de datos (BF), además un registro de dirección de datos (DDR) el cual determina cuales bits del buffer de datos son entradas y cuales salidas. El buffer de datos y el registro de dirección de datos comparten la misma dirección de memoria, la selección de los 2 depende del estado del bit 2 del registro de control.

Bit 2 = 0 . servicio con DDR
Bit 2 = 1 . servicio con Buffer de Datos

Existen varios manuales técnicos y de referencia que pueden ayudar a entender el funcionamiento del PIA. En este caso el M6821. Consúltelos y verifique la tabla de referencia para programación de los registros del PIA en el apéndice E.

Investiga con tu instructor de laboratorio las localidades de memoria que accesan a los registros del PIA.

A.- Tu primera misión es escribir el programa necesario para desplegar un "1" en U1, un "2" en U2, un "3" en U3 y un "4" en U4. (U significa el display de 7 segmentos del kit). Despliegue cada valor por 2 ms y después cheque si una tecla de la columna de la derecha ha sido accionada y después vuelva a la rutina de desplegar. Brinco a la rutina DLY2, la cual regresa con todos los registros preservados (excepto CCR) después de 2 ms puede ser de utilidad. Si se encuentra cualquier tecla de la columna derecha (G, V, A, B, C o D) presionada en 2 ciclos consecutivos, brinque a una rutina switch.

Pregunte al instructor de laboratorio las localidades de memoria que accesan a estas unidades o "displays".

B.- Tu segunda mision es escribir la rutina switch la cual determina cual tecla ha sido presionada y brinca a la localidad con el nombre de la tecla (G, V, A, B, C o D) para desarrollar la función asociada a cada tecla.

Inicie su programa en la localidad $2000, el programa resuelto tomó 111 Bytes de PSEUDORAM.

Tu reporte deberá contener lo siguiente:

1.- El programa en lenguaje ensamblador y su respectivo código máquina perfectamente documentado.

Recuerde que su programa con el mínimo de localidades de memoria ocupadas causará mayor puntuación al alumno.

6.43.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, E, F, G, H, I y J.

Se puede encontrar información adicional en la bibliografía con referencia 1, 4, 5, 6, 8 y 9.

6.44.- PROCESO PRACTICO

El alumno realizará la práctica con el kit D2 del laboratorio. El alumno introducirá el programa y verificará el correcto funcionamiento de este y después lo demostrará al instructor de laboratorio.

No se olvide que hacer una buena presentación al instructor lo cual causar una mejor impresión y por lo tanto mejor evaluación. Explique el funcionamiento del programa.

6.45.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

PROGRAMACION DEL PIA

Nombre del alumno
No. de Credencial
Materia
Lab 4
Fecha

PROGRAMA: DISPLAY ...Asumo para este programa modelo las localidades dememoria indicadas en el programa para los registros de control y de datos del PIA. Con el puerto A se manejarán los LED's y con el puerto B los DISPLAYS.

```
LINEA   ETIQUETA  COD OP   LENG ENSM    COMENTARIOS
                           NAM  IMPRE
                           OPT  O,S
                           ORG  $2000    Inicializa en la localidad
                                         $2000 .

                           LDX  #$8020   Se inicializa el XR para
                                         trabajar con direccionamien-
                                         to indexado.
        PIABFA             EQU  0        Buffer de datos A
        PIACRA             EQU  1        Control puerto A
        PIABFB             EQU  2        Buffer de datos B
        PIACRB             EQU  3        Control puerto B

        * INICIO DE PROGRAMA

        INICIO             CLR  PIACRA,X Borra los registros de con-
                           CLR  PIACRB,X trol y se asegura que el
                                         bit 2 es igual a cero y así
                                         comunicarse con el DDR.

                           LDAA #$7F     Se inicializa el puerto A
                           STAA PIABFA,X como salidas para controlar
                                         a los LED's del display X.
                                         El bit 7 se selecciona como
                                         entrada para checar el
                                         teclado.

                           LDAA #$FF     El display X será seleccio-
                           STAA PIABFB,X nado por el puerto B y aqui
                                         se inicializa para tener a
```

```
                                este puerto como salidas.

                LDAB =4         Ahora se carga un bit en el
                STAB PIACRA,X   bit 2 para indicarle al PIA
                STAB PIACRB,X   que cuando se comunique con
                                el buffer de datos sera
                                para manejo de datos.

                CLR PIABFB,X    Se asegura que no este
                                seleccionado ningun display.

IAQUI           CLR  BANDERA    Esta bandera se inicializa,
                                y el propósito de ésta es
                                llevar un control del núme-
                                ro de ciclos que dura la
                                misma tecla presionada.

BUSCA           LDAB =$20       Selecciona el display 1
                LDAA =6         (puerto B) y el número 1
                BSR  TRAN       (codigo para LED's) (puerto
                                A), y brinca a subrtuina
                                de transmisión.

                LDAA =$5B       Selecciona el No. 2 y
                BSR  TRAN       brinca a subrutina.

                LDAA =$4F       Selecciona el No. 3 y
                BSR TRAN        brinca a subrutina.

                LDAA =$56       Selecciona el No. 4 y
                BSR TRAN        brinca a subrutina.

                LDAA =$FF       Inicia la busqueda por
                STAA PIABFB,X   teclas de la columna
                LDAA PIABFA,X   derecha presionadas. Manda
                                pulso a todas las columnas
                                a través del puerto B y
                                recibe por el puerto A
                                (bit 7) la respuesta a algu-
                                na actividad en la  columna
                                de la derecha.
                                actividad = 0.

                BMI  IAQUI      Si no hay actividad, signi-
                                fica que hay un 1 en el bit7
                                un número negativo y regresa
                                al ciclo BUSCA

                INC  BANDERA    De otra manera, hay activi-
                                dad y se incrementa la ban-
                                dera.
```

```
LDAA BANDERA
CMPA =2
BLT  BUSCA
```

Compara el contenido de la bandera con 2 y si es menor regresa al cilo BUSCA . Significa que solo hubo una falsa actividad.

```
BRA  SWITCH
```

De otra manera, la tecla ha estado presionada por 2 ciclos y es necesario ejecutar la operación de la tecla. brinca a la rutina switch.

* SUBRUTINA DE TRANSMISION

```
TRAN        COMA          Se completa el contenidó
            STAA PIABFA,X de A, ya que los LED's se
            STAB PIABFB,X activan con cero.
                          Se envia la señal al
                          puerto A y al puerto B.

            BSR  DLY2     Se envia a la subrutina de
                          retardo de 2 ms.

            LDAA =$FF     Borra todos los LED's del
            STAA PIABFA,X display.

            LSRB          Con esta instrucción se
                          cambia al siguiente display

            RTS           regreso a programa
                          principal.
```

* Rutina Switch

```
SWITCH      LDAB =$E0      Inicia busqueda de tecla de
            BSR  PRUEBA    la columna derecha. se
                           carga $E0 para ir
                           seleccionando los
                           renglones. brinca a la
                           subrutina que compara con
                           el valor de cada tecla.

            BCC  G         Si existe carry al regresar
                           de la subrutina PRUEBA,
                           indica que se presionó la
                           tecla G y brinca a la
                           localidad de la función G.

            BSR  PRUEBA    Brinca a checar la siguien-
                           te tecla.

            BCC  V         Ahora a la función de la V

            BSR  PRUEBA    Compara con la tecla A.
            BCC  A

            BSR  PRUEBA    Compara con la tecla B.
            BCC  B

            BSR  PRUEBA    Compara con la tecla C.
            BCC  C

            BSR  PRUEBA    Compara con la tecla D.
            BCC  D

            BRA  BUSCA     En caso de error brinca a
                           Subrutina BUSCA.
```

Subrutina PRUEBA... La tecla con el valor correspondiente. El primer valor corresponde al primer renglon y así se va modificando ese valor para seleccionar renglón por renglón.

```
PUEBA    STAB PIABFB,X  Se manda un pulso al
                        renglón correspondiente.

         ANDB  =$3F     Se efectúan estas operacio-
         LSRB            nes para ajustar el valor
         ORAB  =$C0     para el siguiente renglón.

         LDAB PIABFA,X  Se lee el bit 7 y este
                         indicará si hay valor o no
                         activo en el renglón
                         correspondiente.

         RORA           Se mueve el resultado al bit
                         indicador de carry para
                         hacer la comparación en la
                         rutina switch.

         RTS            Regresa a SWITCH.
```

6.46.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.44 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Punto 1 Lenguaje ensamblador y código máquina 45 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.5 :

PROGRAMANDO UN SISTEMA

6.51.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.
  Interruptores

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  El PIA MC6821
  DISPLAYS Y LEDS DEL KIT D2

* CUESTIONARIO PREVIO

NOTA: Las Respuestas están a doble densidad.
Asegúrese de identificar las respuestas.

1.- Diga cuantas lineas de control tiene el lado A del PIA y como es su función con respecto a la entrada y salida.

Hay 2 lineas de control: CA1 (entrada al PIA) y CA2 (salida del PIA).

Dentro del registro de control del PIA hay un bit similar a un bit del ACIA, es decir tiene la misma función. Indique como se denomina ese bit, y además considerando del bit 0 al bit 7 diga a cual se refiere, y por último si indique si es posible programarlo.

Se denomina "READY" bit y se refiere al bit 7, y no es posible programarlo ya que es borrado automaticamente cuando datos son leidos del buffer de datos .

3.- Describa las lineas de interrupción del PIA, IRQA y IRQB.

Estas lineas, las cuales interrumpen al MPU ya sea directa o indirectamente a través de un circuito de prioridad de interrupción, son "fuente abierta" (no hay carga sobre el chip). Son capaces de manejar una corriente de 3.2 mA de una fuente externa, por lo tanto permite todas las lineas de requerimiento de interrupción atarlas juntas en la configuración "OR ALAMBRADO" (WIRE OR).

Las interrupciones tienen servicio por medio de rutinas de software que secuencialmente leen y prueban, en base a prioridades los bits 6 y 7 .(los que esten activos).

Cuando el MPU lee los registros periféricos, las banderas de interrupción (bit 6 y 7) son borrados al mismo tiempo que el requerimiento de interrupción.

4.- Bajo que condiciones el bit 7 del PIACRA es activado ("1").

1.- Una transición negativa en la linea CA1 sea detectada y el bit 1 del PIACRA sea cero "0".

2.- Una transición hacia arriba (positiva) en la linea CA1 sea detectada y el bit 1 del PIACRA sea uno "1".

5.- Escriba las instrucciones necesarias para programar el PIA lado B como sigue: CB1 como entrada, CB2 como linea de salida en modo "HANDSHAKE". CB1 se active en transición negativa.

```
 : :
LDAA   =%0010 0101
STAA   PIACRB
 : :
```

6.52.- DISCUSION PRELIMINAR

En esta práctica el alumno tratará de reconocer un problema y manejarlo por medio de la programación del microprocesador MC6800 y del PIA MC6821.

La práctica consiste en desarrollar un programa que pueda resolver la siguiente situación:

El microprocesador en conjunto con el PIA puede ser usado en un sensor de automóvil y un sistema de alarma.

Suponga que tiene los siguientes parámetros como entradas al sistema:

bit	FUNCION (Monitor de)	ESTADOS 0	1
0	Cinturón de Seguridad	desconectado	sujeto
1	Puerta	cerrada	abierta
2	Presión de aceite	baja	normal
3	Ignición	apagado	encendido
4	Palanca de Velocidades	neutral/Park	Otras
5	Motor	no encendido	encendido
6	Dia/Noche	noche	dia
7	Luces	apagadas	encendidas

y los siguientes parámetros como salidas:

bit	FUNCION (Monitor de)	ESTADOS 0	1
0	Buzzer	apagado	encendido
1	Campanas	apagadas	sonando
2	Alarma luminosa en el pánel	apagado	encendido
3	Control de arranque	apagado	encendido

Pues bien, escriba las instrucciones necesarias para habilitar el encendido del motor si y solo si el cinturón de seguridad esta sujeto, el carro se encuentra en neutral y la puerta esta cerrada.

Después escriba las instrucciones dentro del mismo y con la lógica adecuada para encender la luz en el pánel si el motor esta encendido y la presión de aceite baja.

Y por último, hacer sonar la campana si el motor esta apagado y las luces han quedado encendidas.

Documente su programa y entregue un reporte claro y limpio.

Para aquellos alumnos aventajados, podrán implementar las rutinas de la práctica anterior y así desplegar mensajes a través del display del KIT-D2.

6.53.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, E, F, G, H, I y J.

Se puede encontrar información adicional en la bibliografía con referencia 1, 4, 5, 8 y 9.

6.54.- PROCESO PRACTICO

El alumno investigará las localidades de memoria disponibles para las salidas y entradas del PIA.

La idéa, por ahora es conectar interruptores a las entradas del PIA y así establecer las condiciones pedidas en la práctica, simulando los sensores.

Para simular las salidas, bien se pueden conectar LED's y así obtener la respuesta deseada de acuerdo a las condiciones requeridas.

Demuestre su programa y ejecucción del mismo. No haga perder el tiempo a su instructor y asegúrese que su programa funcione correctamente antes de demostrarlo a su instructor.

Por otro lado este preparado para cualquier pregunta por parte de su instructor ya que se tomará en cuenta para su evaluación.

6.55.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

PROGRAMANDO UN SISTEMA

Nombre del alumno
No. de Credencial
Materia
Lab 5
Fecha

CHECANDO UN AUTOMOVIL....Programa para checar la actividad de un automóvil y tener un sistema de alarma y protección.

```
LINEA   ETIQUETA  COD OP   LENG ENSM     COMENTARIOS
                           NAM  IMPRE
                           OPT  O,S
                           ORG  $2000     Inicializa en la localidad
                                          $2000 .

                           LDX  =$XXXX    Se direcciona las localida-
        PIABFA             EQU  0         des disponibles del PIA pa-
        PIACRA             EQU  1         ra accesar y transmitir
        PIABFB             EQU  2         información.
        PIACRB             EQU  3

                           CLR  PIACRA,X  Borra los registros de
                           CLR  PIACRB,X  control y se asegura que el
                                          bit 2 es igual a cero y así
                                          comunica con el DDR.

                           CLR  PIABFA,X  Se inicializa el puerto A
                                          como entradas.

                           LDAA =$FF      Se inicializa el puerto B
                           STAA PIABFB,X  como salidas.

                           LDAA =4        Ahora se carga un 1 en el
                           STAA PIACRA,X  bit 2 para indicarle al PIA
                           STAA PIACRB,X  que el PIABFB y PIABFA
                                          seran usados para menejo de
                                          datos. La programación de
                                          interrupción es como el
                                          programador lo requiera.
```

```
                CLR PIABFB,X    Se asegura que no haya
                                nada a la salida.

CHECANDO        LDAA PIABFA,X   Carga datos de sensores
                                para la primera aplicación.
                                encendido.

                BITA =1         Cinturón asegurado (bit
                BEQ CHECANDO 0).

                BITA =$12       Puerta cerrada (bit 1),
                BNE CHECANDO    velocidad en park o neutral
                                (bit 4).

                LDAA PIABFB,X   Puede arrancar. habilita
                ORAA =%1000     encendido. (ignicion).
                STAA PIABFB,X

ACEITE          LDAA PIABFA,X   Carga datos de sensores
                                para la segunda aplicación.
                                Presión de aceite.

                BITA =$20       Motor encendido (bit 5).
                BEQ  LUZ        Si apagado, va a rutina de
                                luces.

                BITA =%100      Presión de aceite (bit 2).
                BEQ  ALARMA

                BRA  ACEITE

LUZ             BITA =$80       Luces (bit 7). Si
                BNE CAMPANAS    encendidas hace sonar las
                BRA CHECANDO    campanas. De otra manera
                                regresa a rutina de checar
                                el encendido.

ALARMA          LDAA PIABFB,X   Hace encender la alarma
                ORAA =%100      luminosa debido a presión
                STAA PIABFB,X   de aceite baja.
                BRA  ACEITE

CAMPANAS        LDAA PIABFB,X   Hace sonar las campanas
                ORAA =%10       debido a motor apagado y
                STAA PIABFB,X   luces encendidas.
                BRA ACEITE

                END             Fin de programa.
```

Como podrán observar este programa es un ciclo continuo de pruebas y para detenerlo se usará el reset maestro.

6.56.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Cuestionario Previo: Se otorgarán 10 puntos por el hecho de entregar el cuestionario contestado correctamente. el valor de cada respuesta es de 2 puntos.

b) Respecto al Proceso Práctico: Se otorgarán 45 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.54 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

Punto 1 Lenguaje ensamblador y código máquina 45 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....10 puntos
Punto (b)....45 puntos
Punto (c)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.6 :

SUBRUTINAS

6.61.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.
  Impresora Serial

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  INSTRUCCIONES PARA SOLICITAR SUBRUTINAS Y REGRESO DE SUBRUTINAS
  JSR, BSR, RTS
  EL ACIA

* CUESTIONARIO PREVIO

El programa sugerido en esta práctica es considerado con un grado de dificultad tal que se requiere mayor tiempo del que se ha estado estimado, por lo tanto no se requerirá un cuestionario previo.

6.62.- DISCUSION PRELIMINAR

En prácticas previas se han usado llamdas a subrutinas, las cuales causan a un grupo de instrucciones a ser ejecutadas y terminadas por la instrucción RTS "Return to subroutine".

Un programa puede ser hecho de una serie de llamadas a subrutina, cada una causando la ejecucción de una subrutina en particular, para así llevar a cabo una tarea específica.

Cada subrutina cdeberá tener un solo punto de entrada y un solo punto de salida. Condiciones de entrada y salida deberán ser documentadas. Cada subrutina puede ser probada individualmente y entonces usada con confianza cuando sea llamada por el programa principal.

El plan del programa deberá ser en formato de "arriba-a-abajo", con sus tareas definidas completamente desde el principio, y de estas tareas definir las subtareas. Cada tarea puede entonces ser asignada a una subrutina, la cual a su vez puede llamar a otra subrutina de un nivel inferior y así manejar las subtareas.

Llamadas de subrutinas pueden tener varios niveles de profundidad, si es necesario, esas al nivel mas bajo, pueden ser responsables de la tarea mas simple.

El resultado es una estructura piramidal o jerárquica, los niveles superiores siguen siendo globales o generales y los niveles inferiores siendo a detalle .

La práctica consiste en escribir una subrutina llamada "PAGINA" la cual imprime una página de datos, la primera dirección de los datos esta en el registro índice cuando "PAGINA" es llamada. El formato es como sigue:

--Una "PAGINA" esta compuesta de 16 lineas.

-- Una "LINEA" esta compuesta de un retorno de carro (CR) y alimentación de linea (Line Feed) (para empezar una nueva linea) seguida por 8 "PALABRA"s, cada "PALABRA" separada por un espacio.

-- Una "PALABRA" comprende 4 bytes, de memoria, cada byte es impreso como 2 caracteres ASCII. Ejemplo: 0011 1101 en memoria causaría imprimir "3D".

Use el formato de "arriba-a-abajo" para resolver este problema.

Inicialice el ACIA para transmitir la información a una impresora.

Asegúrese de conectar la impresora correctamente al Kit.

Su reporte deberá contener una gráfica de su estructura piramidal de sus subrutinas. Además deberá contener su programa en lenguaje ensamblador.

6.63.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, D, F, G, H, I y J.

Se puede encontrar información adicional en la bibliografía con referencia 1, 4, 5, 8 y 9.

6.64.- PROCESO PRACTICO

El alumno instalará el sistema de impresora y kit D2, investigando las localidades de memoria disponibles para la transmisión por el ACIA.

SUGERENCIA: El alumno podrá basarse en las prácticas anteriores referentes al ACIA para inicializar su programa principal.

Para fines de demostración, imprima las localidades de memoria de $0000 hasta completar una página. El funcionamiento de su programa y la corrida .

Su instructor evaluará cuando la demostración sea satisfactoria y quizás no sea necesario imprimir todas las localidades de memoria.

6.65.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

SUBRUTINAS

Nombre del alumno
No. de Credencial
Materia
Lab 6
Fecha

La inicializacíon del ACIA y la secuencia de instrucciones necesarias para tener un programa principal que llame a la siguiente subrutina puede establecerse de acuerdo a las necesidades y en base a prácticas anteriores.

Para no repetir las secuencia de instrucciones de inicializar el ACIA para salida a impresora y en modo de operacíon indicaré lo siguiente: INICIAACIA.

```
LINEA   ETIQUETA  COD OP   LENG ENSM      COMENTARIOS
                           NAM  PRINCIP
                           OPT  O,S
                           ORG $2000      Inicializa en la localidad
                                          $2000 .

        PRINCIP            INICIAACIA
                           LDX =$0000     Direccíon de inicio de
                           JSR  PAGINA    programa y brinca a subru-
                           SWI            tina.
```

PAGINA....Subrutina que imprime una página de 16 lineas de datos desde la memoria. El registro índice indicará el primer caracter a ser impreso. LLama a la subrutina linea.

```
        PAGINA             DEX            Uno menos del primer carac-
                           STX  APUNTA    ter. inicia apuntador.
                           LDAA =16
                           STAA LINCON    Establece contador de linea

        OTRALIN            JSR  LINEA     Imprime linea
                           DEC  LINCON    última linea?
                           BNE  OTRALIN   No. Imprime otra linea.
                           RTS            Regresa a PINCIPAL.

        LINCON             RMB  1         Contador de lineas
        APUNTA             RMB  2         Apuntador
```

LINEA...Subrutina para imprimir 64 caracteres de 32 localidades de memoria. Llama a la subrutina PALABRA, y a subrutina CRLF (retorno de carro y alimentoación de linea).

```
LINEA       JSR  CRLF      Inicia nueva linea.
            LDAA =8
            STAA PALCON    Contador de palabras.

OTRAPAL     JSR  PALABRA   Imprime palabra.
            DEC  PALCON    Ultima palabra?
            BNE  OTRAPAL   No. Otra palabra.
            RTS            Regresa a PAGINA.

PALCON      RMB  1         Contador de palabras.
```

PALABRA...Subrutina para imprimir los contenidos de 4 localidades de memoria como 8 caracteres hexadecimales. LLama a subrutinas OBYTE y ESPACIO.

```
PALABRA     LDAA =4
            STAA BYTCON    Inicia contador de Bytes.

OTROBYT     JSR  OBYTE     Imprime un byte como 2 ca-
            DEC  BYTCON    racteres. último byte?
            BNE  OTROBYT   No. Otro byte.
            JSR  ESPACIO   Si. Entonces ESPACIO.
            RTS            Regresa a LINEA.

BYTCON      RMB  1         Contador de Bytes.
```

OBYTE...Subrutina para imprimir el contenido de una localidad de memoria como 2 caracteres ASCII. LLama a subrutina HEXPRT.

```
OBYTE   LDX  APUNTA   Obtiene la dirección de
        INX           inicio de página.
        STX  APUNTA

        LDAA 0,X      Obtiene un byte, y copia
        STAA TEMP     el byte para uso posterior.

        ASRA          Coloca los 4 bits del lado
        ASRA          izquierdo en el lado
        ASRA          derecho.
        ASRA

        ANDA =$0F     Elimina el lado izquierdo y
        JSR  HEXPRT   por lo tanto los bits mas
                      mas significativos serán
                      impresos.

        LDAA  TEMP    Recobra valor inicial del
        ANDA =$0F     contenido de memoria y
        JSR  HEXPRT   ahora imprime los 4 bits
                      menos significativos.

        RTS           Regresa a palabra.
```

HEXPRT...Subrutina para imprimir un caracter ASCII. Llama a subrutina IMPRINT.

```
HEXPRT  ADDA =$30     Convierte el valor ASCII.
        CMPA =$39     Es número?
        BLS  SALIDA   Si es número va a SALIDA

        ADDA =7       es letra, suma 7 para con-
                      vertir en ASCII.

SALIDA  JSR  IMPRINT  a IMPRESION

        RTS           Regresa a OBYTE.
```

IMPRINT...Subrutina para imprimir caracteres si el componente (impresora) esta en-linea, via CTS = 1 (CTS=0). el código ASCII esta en el registro A.

```
ACIACSR         EQU  $7FF4
ACIABUF         EQU  $7FF5

IMPRINT         LDAB ACIACSR  CTS  no igual a "0"
                BITB  =8      No.  Trata de nuevo.
                BNE  IMPRINT

                BITB  =2      Listo.?
                BNE  IMPRINT  No.  Trata de nuevo.

                STAA ACIABUF  imprime a través del ACIA.
                RTS           Regresa a HEXPRT.
```

ESPACIO...Subrutina para avanzar un espacio imprimiendo el caracter equivalente a espacio. LLama a subrutina IMPRINT.

```
ESPACIO         LDAA =$20     ASCII para ESPACIO.
                JSR  IMPRINT
                RTS           Regresa a PALABRA
```

CRLF...Subrutina para liberar un (CR) retorno de carro y alimentación de linea (LF) a la impresora. LLama a subrutina IMPRINT. Ajusta cada uno con 2 caracteres nulos.

```
CRLF            LDAA =$0D     CR  Retorno de carro.
                JSR  IMPRINT  e imprime .

                CLRA          Salida de dos nulos.
                JSR  IMPRINT
                JSR  IMPRINT

                LDAA  =$0A    LF  Alimentación de linea.

                CLRA          Salida de dos nulos ,
                JSR  IMPRINT
                JSR  IMPRINT
                RTS           Regresa a LINEA.
```

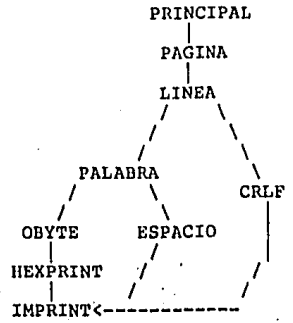


DIAGRAMA DE FLUJO.

6.66.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Proceso Práctico: Se otorgarán 40 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.64 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado

c) Con Respecto al Reporte: Se otorgarán 60 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Punto 1 Lenguaje ensamblador y código máquina 60 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

TOTAL = 60 puntos.

Las partes están señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....40 puntos
Punto (b)....60 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de dos semanas.

PRÁCTICA No.7 :

MANEJO DE MAPAS DE MEMORIA

6.71.- CONSIDERACIONES BÁSICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  CONCEPTOS SOBRE MAPAS DE MEMORIA

* CUESTIONARIO PREVIO

El programa sugerido en esta práctica es considerado con un grado de dificultad tal que se requiere mayor tiempo del que se ha estado estimado, por lo tanto no se requerirá un cuestionario previo.

6.72.- DISCUSION PRELIMINAR

Para este laboratorio escribirás un paquete de subrutinas en lenguaje ensamblador, y usarás el ensmablador MC6800 para ensamblarlo. Entonces usarás MUDBUG para corregir y demostrar tus subrutinas. Tu paquete de subrutinas para este laboratorio contendrá 4 subrutinas de fácil acceso al usuario para manejo de mapas de bits, y quizás tu subrutinas necesiten de subrutinas propias.

Un mapa de bits es una estructura de almacenamiento de datos en memoria que es muy útil para guardar información de variables (como variables lógicas) que tengan unicamente 2 posibles estados (verdadero o falso). Necesitamos unicamente un bit de memoria para almacenar una variable lógica, de esta manera 8 variables lógicas pueden ser empacadas en una palabra sencilla de RAM en un sistema MC6800. Un bloque de memoria que contiene variables lógicas empacadas es llamado entonces un mapa de bits.

Un mapa de bits de 32-palabras en un sistema MC6800 puede contener hasta 8*256=256 diferentes variables lógicas. Cada variable lógica en un mapa de bits es indexada de acuerdo a su posición relativa, así las variables lógicas en un mapa de bits de 32-palabras es indexado desde el bit 0 hasta el bit 255. Bit cero de la palabra cero es el bit cero de todo el mapa de memoria, y el bit 7 de la palabra 31 es el bit 255 de todo el mapa de bits. El resto de un mapa de bits es como se muestra enseguida:

DIRECCION DE MEMORIA	BITS							
BITMAP + 0	7	6	5	4	3	2	1	0
BITMAP + 1	15	14	13	12	11	10	9	8
BITMAP + 2	23	22	21	20	19	18	17	16
BITMAP + 3	31	30	29	28	27	26	25	24

Cualquier programa que requiera un gran número de variables lógicas es un candidato potencial para usar un mapa de bits. Siempre y cuando la velocidad de ejecucción no sea de consideración crítica, un mapa de bits puede ser usado exitosamente en un sistema MC6800 para reducir los requerimientos de RAM para variables lógicas por un factor de 8. Cualquier variable que deba ser accesada frecuentemente y sea requerida en un tiempo determinado puede ser codificada individualemnte y todas las otras variables lógicas sean manejadas por el mapa de bits.

Programas que usan variables lógicas de un solo bit comunmente prueban el valor de una variable, establecen una variable, borran una variable, o cambian el valor de la variable (invierten). Tu paquete de subrutinas, deberá contener subrutinas PRUEBA, SET, BORRAR, CAMBIAR, las cuales proveerán un metodo combeniente para desarrollar cada una de estas operaciones para cualquier variable lógica del mapa de bits de

32-palabras. Las especificaciones de las subrutinas son dadas a continuación:

Subrutina PRUEBA... prueba el valor de un bit específico del mapa de bits, y regresa el control a la rutina que lo mando llamar con el bit Z del registro de condiciones (CCR) en uno o en cero para reflejar el valor del bit que fue probado. La secuencia para llamar a la subrutina PRUEBA es como sigue:

```
JSR   PRUEBA              * o BSR
FCB   numero de bit       * del 0 al 255
```

La palabra que inmediatamente esta después de la instrucción JSR (o BSR)en memoria es llamada la palabra parámetro, y contiene el número indexado sin signo del bit en particular en el bit map que va a ser probado.

Subrutina PRUEBA siempre preserva el valor de los registros AR, BR, XR y apuntador de STACK, y regresa control a la rutina de control con el bit Z del CCR reflejando el valor de la prueba. bits H e I son preservados. bits N, V y C no son importantes.

Subrutina SET ... establece un bit especificado en uno del mapa de bits. La secuencia es la siguiente:

```
          JSR   SET               * o BSR
          FCB   numero de bit     * del 0 al 255
ejemplo:
OPEN      EQU   213
          JSR   SET
          FCB   OPEN
```

El CCR debe terminar después de esta rutina como sigue: Z será colocado en 0. bits H e I son preservados. bits N, V y C no son importantes.

Subrutian BORRAR... borra un bit específico, lo pone en cero, del mapa de bits, y la secuencia es la siguiente:

```
JSR   BORRAR              * o BSR
FCB   numero de bit       * del 0 al 255
```

El CCR debe terminar después de esta rutina como sigue: Z será colocado en 1. bits H e I son preservados. bits N, V y C no son importantes.

Subrutina CAMBIAR... intercambia el valor de un bit específico del mapa de memoria (si es cero lo pone en uno, y el uno lo pone en cero). y la secuencia es como sigue:

```
JSR   CAMBIAR             * o BSR
FCB   numero de bit       * del 0 al 255
```

El CCR debe terminar después de esta rutina como sigue: Z reflejará el valor del cambio realizado. bits H e I son preservados. bits N, V y C no son importantes.

Las especificaciones de estas subrutinas quizás parezcan formidables, al principio, pero el código para implementarlas es preferible que sea directo. Las especificaciones del código de condiciones, las cuales parecen ser complejas son casi automaticamente realizadas si haces tu trabajo con cuidado.

Asume que tienes ROM para tu paquete de subrutinas empezando en la localidad de memoria $2000, y escribe tus subrutinas para que puedan correr en area ROM. Puedes usar las localidades del $180 al $19F para tu mapa de bits, y si requieres almacenamiento temporal puedes usar las localidades del $1A0 al $1AF; y por supuesto puedes usar el STACK.

La solución para este programa tomó $5B localidades de memoria ROM, 2 localidades RAM y use 5 localidades del STACK.

PROGRAMA MAESTRO PARA PROBAR TUS SUBRUTINAS

```
BIT     EQU  0          Esta ecuación define un bit en particular
                        en el mapa de bits, y maneja el programa
                        maestro para ejercicios.

        ORG $23F7
INICIO  JSR  SET        Establece el bit, entonces se detiene así
        FCB  BIT        el usuario puede convenientemente checar
        SWI             los valores de los registros y si es
                        necesario listar la memoria y checar los
                        valores del mapa de bits.

        JSR BORRAR      Borra el bit, se detiene para verificar
        FCB  BIT        los valores en los registros.
        SWI

        JSR CAMBIA      Intercambia el bit a cero o a uno, y se
        FCB  BIT        detien para que el usuario verifique.
        SWI

        JSR PRUEBA      Prueba el bit y el usuario checará los
        FCB  BIT        registros y verificará que el mapa de
        SWI             bits no ha cambiado.

        JSR CAMBIA      Intercambia el bit a cero o a uno, y se
        FCB  BIT        detien para que el usuario verifique.
        SWI

        JSR PRUEBA      Prueba el bit y el usuario checará los
        FCB BIT         registros y verificará que el mapa de
        SWI             bits no ha cambiado.

        BRA INCIO       Reinicia la secuencia completamente.

        END
```

6.73.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, D, F, G, H, I y J.

Se puede encontrar información adicional en la bibliografía con referencia 1, 8 y 9.

6.74.- PROCESO PRACTICO

Usa el programa maestro para correr tu programa y si es necesario hacer correcciones y demuestres el buen funcionamiento de tu programa al instructor. Corre el programa maestro con los siguientes valores BIT=0, luego con BIT=$7F, luego con BIT=$80, luego con BIT=$95, luego con BIT=$FF y verifica la correcta operación de cada una de tus subrutinas para cada corrida. Asegúrate de modificar los valores antes de correr con un nuevo valor para BIT y de esa manera estarás seguro de que tu programa funciona correctamente.

Para este laboratorio deberás organizar tu reporte como sigue:

1.- Entrega un listado de tus subrutinas completamente comentadas y en lenguaje ensamblador .

2.- Dibuja un diagrama de flujo a nivel función de cada una de tus subrutinas incluyendo ademas subrutinas internas.

3.- Indica cuantas localidades de ROM , RAM, y STACK fueron usadas por tus subrutinas. Asegúrate de incluir las dos localidades de STACK que son usadas por tus subrutinas para mantener la dirección de regreso al programa principal.

6.75.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

MANEJO DE MAPAS DE MEMORIA

Nombre del alumno
No. de Credencial
Materia
Lab 7
Fecha

```
LINEA  ETIQUETA  COD OP   LENG ENSM      COMENTARIOS
                          NAM  BITMAP
                          OPT  O,S
                          ORG  $0180     USO DE RAM
                                         $2000 .

       PRINCIP            RMB  32
                          ORG  $2000     INICIA PROGRAMA
       BIMPAP             EQU  $0180     Inicializa variables y
       SAVEX              EQU  $01A0     reserva localidades de
       SAVE1              EQU  $01A2     memoria.
       SAVE2              EQU  $01A4

       INIT               STX  SAVEX     Inicia subrutina para
                          TSX            poder preservar los valores
                          STX  SAVE1     de los registros, almace-
                          PSHA           nando en el STACK.
                          PSHB           Esta es una subrutina
                          LDAA  =RES     interna.
                          PSHA
                          STX  SAVE2
       1H                 ROR  BIMAP + -BITE
                          DECA
                          BNE   1B
                          LDX  SAVE1
                          TXS
                          RTS            Regresa a subrutina que lo
                                         mando llamar.

       PRUEBA             BSR  INIT      Subrutina  PRUEBA.
                          BHS  2F
       1H                 SEC
```

```
                SK1
2H              CLC
                BRA  RETURN


SET             BSR INIT        Subrutina  SET
                BRA 1B


BORRAR          BSR INIT        Subrutina BORRAR
                BRA  2B


CAMBIAR         BSR INIT        Subrutian CAMBIAR.
                BHS  1B
                BRA  2B


RETURN          LDX  SAVE2      Para regresar al programa
                TXS             principal. ajustes para
                PULA                    saber a que linea
1H              ROL  BIMAP +    BITE    tenemos que
                DECA                    regresar y que
                BNE  1B         los registros hayan quedado
                LDX  SAVE1      preservados.
                INX
                INX
                TXS
                PULA
                PULA
                ADDB  =1
                ADCA  =0
                PSHB
                PSHA
                LDX. SAVE2
                INX
                TXS
                PULB
                PULA
                LDX  SAVE1
                INX
                INX
                TXS
                LDX  SAVEX
                RTS             Regreso a programa principal
```

PROGRAMACION AVANZADA

FLUJO DE SUBRUTINA PRUEBA

INICIO

* ALMACENA REGISTROS A Y B EN EL STACK

* BRINCA A SUBRUTNA INIT

-------VALOR ORIGINAL DEL REGISTRO INDICE A LOCALIDAD SAVEX

-------LOCALIDAD DE MEMORIA ACTUALIZADA PARA PODER REGRESAR A PROGRAMA PRINCIPAL

-------REGISTRO A TOMA EL VALOR DE BIT DEL PROGRAMA PRINCIPAL Y LO PROCESA PARA ASI TENER LA LOCALIDAD DE MEMORIA DE LA PALABRA QUE SE VA A ALTERAR EN EL MAPA DE BITS Y EL BIT QUE SE VA A ALTERAR ESTA EN EL REGISTRO B.

------>PRUEBA EL BIT QUE ES INDICADO POR LA PALABRA Y POR EL REGISTRO B

* REGRESA A SUBRUTINA

-------ALMACENA EL VALOR DEL REGISTRO B EN EL MAPA DE BITS

-------REGRESA LOS VALORES INICIALES A LOS REGISTROS A Y B

-------REGRESA EL VALOR INICIAL AL REGISTRO INDICE DEL LA LOCALIDAD SAVEX

* REGRESA A PROGRAMA PRINCIPAL.

FLUJO DE SUBRUTINA SET

INICIO

* ALMACENA REGISTROS A Y B EN EL STACK

* BRINCA A SUBRUTNA INIT

-------VALOR ORIGINAL DEL REGISTRO INDICE A LOCALIDAD SAVEX

-------LOCALIDAD DE MEMORIA ACTUALIZADA PARA PODER REGRESAR A PROGRAMA PRINCIPAL

-------REGISTRO A TOMA EL VALOR DE BIT DEL PROGRAMA PRINCIPAL Y LO PROCESA PARA ASI TENER LA LOCALIDAD DE MEMORIA DE LA PALABRA QUE SE VA A ALTERAR EN EL MAPA DE BITS Y EL BIT QUE SE VA A ALTERAR ESTA EN EL REGISTRO B.

------>COLOCA EL VALOR DE UNO AL BIT EN CUESTION A TRAVEZ DEL REGISTRO B

* REGRESA A SUBRUTINA

-------ALMACENA EL VALOR DEL REGISTRO B EN EL MAPA DE BITS

-------REGRESA LOS VALORES INICIALES A LOS REGISTROS A Y B

-------REGRESA EL VALOR INICIAL AL REGISTRO INDICE DEL LA LOCALIDAD SAVEX

* REGRESA A PROGRAMA PRINCIPAL.

PROGRAMACION AVANZADA

FLUJO DE SUBRUTINA BORRAR

INICIO

* ALMACENA REGISTROS A Y B EN EL STACK

* BRINCA A SUBRUTNA INIT

-------VALOR ORIGINAL DEL REGISTRO INDICE A LOCALIDAD SAVEX

-------LOCALIDAD DE MEMORIA ACTUALIZADA PARA PODER REGRESAR A PROGRAMA PRINCIPAL

-------REGISTRO A TOMA EL VALOR DE BIT DEL PROGRAMA PRINCIPAL Y LO PROCESA PARA ASI TENER LA LOCALIDAD DE MEMORIA DE LA PALABRA QUE SE VA A ALTERAR EN EL MAPA DE BITS Y EL BIT QUE SE VA A ALTERAR ESTA EN EL REGISTRO B.

------>COLOCA EL VALOR DE CERO AL BIT EN CUESTION A TRAVEZ DEL REGISTRO B

* REGRESA A SUBRUTINA

-------ALMACENA EL VALOR DEL REGISTRO B EN EL MAPA DE BITS

-------REGRESA LOS VALORES INICIALES A LOS REGISTROS A Y B

-------REGRESA EL VALOR INICIAL AL REGISTRO INDICE DEL LA LOCALIDAD SAVEX

* REGRESA A PROGRAMA PRINCIPAL.

PROGRAMACION AVANZADA

FLUJO DE SUBRUTINA CAMBIAR

INICIO

* ALMACENA REGISTROS A Y B EN EL STACK

* BRINCA A SUBRUTNA INIT

-------VALOR ORIGINAL DEL REGISTRO INDICE A LOCALIDAD SAVEX

-------LOCALIDAD DE MEMORIA ACTUALIZADA PARA PODER REGRESAR A PROGRAMA PRINCIPAL

-------REGISTRO A TOMA EL VALOR DE BIT DEL PROGRAMA PRINCIPAL Y LO PROCESA PARA ASI TENER LA LOCALIDAD DE MEMORIA DE LA PALABRA QUE SE VA A ALTERAR EN EL MAPA DE BITS Y EL BIT QUE SE VA A ALTERAR ESTA EN EL REGISTRO B.

------>INVIERTE LOS VALORES DEL BIT EN CUESTION A TRAVEZ DEL REGISTRO B

* REGRESA A SUBRUTINA

-------ALMACENA EL VALOR DEL REGISTRO B EN EL MAPA DE BITS

-------REGRESA LOS VALORES INICIALES A LOS REGISTROS A Y B

-------REGRESA EL VALOR INICIAL AL REGISTRO INDICE DEL LA LOCALIDAD SAVEX

* REGRESA A PROGRAMA PRINCIPAL.

6.76.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Proceso Práctico: Se otorgarán 55 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.74 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 25 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

* Reporte de Los diagramas de flujo de las subrutinas es igual a 15 puntos

* Reporte de las limitaciones 5 puntos.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....55 puntos
Punto (b)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.8 :

DIRECCIONAMIENTO DE REGISTROS

6.81.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  DIRECCIONAMIENTO INDEXADO

* CUESTIONARIO PREVIO

El programa sugerido en esta práctica es considerado con un grado de dificultad tal que se requiere mayor tiempo del que se ha estado estimado, por lo tanto no se requerira un cuestionario previo.

6.82.- DISCUSION PRELIMINAR

Tu mision para este laboratorio es escribir, depurar y demostrar un paquete de subrutinas que mejorará las instrucciones LDAA y STAA del MC6800 por medio de un nuevo método de modos de direccionamiento para estas instrucciones. Los nuevos modos de direccionamiento que tu implementarás ya están disponibles en el microprocesador MC6809, pero esos no existen en este microprocesador. Aunque los modos de direccionamiento que manejen tus subrutinas son nuevos, deverán ser consistentes con las instrucciones actuales LDAA y STAA.

En resumen, tu paquete de subrutinas implementará tres tipos de direccionamiento indexado. Un modo permitirá el uso del registro AR con un "offset" de 8 bits con signo (en complemnto a 2) para el registro índice (XR), y otro modo permitirá el uso de BR como "offset" de XR. El tercer modo permitirá el uso de doble presición usando el AR como (MSB) y el BR como (LSB) y asi tener un offset de 16 bits para el XR.

Tu paquete de subrutinas contendrá 6 subrutinas de fácil acceso para el usuario las cuales seran 3 versiones del LDAA y 3 versiones del STAA. Estas subrutinas, nombradas LDAAAX, LDAABX, LDAADX, STAAAX, STAABX, y STAADX operarán como las instrucciones LDAA y STAA, excepto por los cambios en los registros y en el CCR. Los siguientes parrafos describirán la función de cada subrutina.

Subrutina LDAAAX implementa la instrucción "LDAA %A,X" donde la anotación "%A" indica que el valor signado que esta inicialmente en AR será usado como "offset" por el XR en la computación para direccionar el operando. Por ejemplo: Si el valor inicial en AR es $FF y el valor en XR es $8000, la dirección del operando es $FFFF + $8000 = $7FFF. Cuando la subrutina LDAAAX regresa control a la rutina que lo llamo, el valor apropiado del operando se encontrará en AR, y los códigos de condición son actualizados como si fueran actualizados por una función normal de LDAA con el mismo valor de operando. Los registros BR, XR y SP son preservados.

Subrutina LDAABX implementa la instrucción "LDAA %B,X" donde la anotación "%B" indica que el valor signado que esta inicialmente en BR sera usado como "offset" por el XR en la computación para direccionar el operando. Por ejemplo: Si el valor inicial en BR es $01 y el valor en XR es $7FFF, la dirección del operando es $0001 + $7FFF = $8000. Cuando la subrutina LDAABX regresa control a la rutina que lo llamo, el valor apropiado del operando se encontrará en AR, y los códigos de condición son actualizados como si fueran actualizados por una función normal de LDAA con el mismo valor de operando. Los registros BR, XR y SP son preservados.

Subrutina LDAADX implementa la instrucción "LDAA %D,X" donde la anotación "%D" indica que el valor signado de 16 bits que esta inicialmente en doble presición, AR como (MSB) y BR como (LSB) serán usados como "offset" por el XR en la computación para direccionar el operando. Por ejemplo: Si el valor inicial de AR es $20, y el valor inicial en BR es $F7 y el valor en XR es $FFFF, la dirección del operando es $2000 + $00F7 + $FFFF = $20F6. Cuando la subrutina LDAADX regresa control a la rutina que lo llamó, el valor apropiado del operando se encontrará en AR, y los códigos de condición son actualizados como si fueran actualizados por una función normal de LDAA con el mismo valor de operando. Los registros BR, XR y SP son preservados.

Subrutina STAAAX implementa la instrucción "STAA %A,X" donde la anotación "%A" indica que el valor signado que esta inicialmente en AR sera usado como "offset" por el XR en la computación para direccionar el operando. Por ejemplo: Si el valor inicial en AR es $80 y el valor en XR es $6100, la dirección del operando es $FF80 + $6100 = $6080. Cuando la subrutina STAAAX regresa control a la rutina que lo llamó, el valor que se encontraba en AR se encontrará en la localidad de memoria apropiada , y los códigos de condición son actualizados como si fueran actualizados por una función normal de STAA con el mismo valor de operando. Los registros BR, XR y SP son preservados.

Subrutina STAABX implementa la instrucción "STAA %B,X" donde la anotación "%B" indica que el valor signado que esta inicialmente en AR sera usado como "offset" por el XR en la computación para direccionar el operando. Por ejemplo: Si el valor inicial en BR es $40 y el valor en XR es $FFC0, la dirección del operando es $0040 + $FFC0 = $0000. Cuando la subrutina STAABX regresa control a la rutina que lo llamo, el valor que se encontraba en AR se encontrará en la localidad de memoria apropiada , y los códigos de condición son actualizados como si fueran actualizados por una función normal de STAA con el mismo valor de operando. Los registros BR, XR y SP son preservados.

Subrutina STAADX implementa la instrucción "STAA %D,X" donde la anotación "%D" indica que el valor signado de 16 bits que esta inicialmente en doble presición, AR como (MSB) y BR como (LSB) serán usados como "offset" por el XR en la computación para direccionar el operando. Por ejemplo: Si el valor inicial de AR es $80, y el valor inicial en BR es $20 y el valor en XR es $5000, la dirección del operando es $8000 + $0020 + $5000 = $D020. Cuando la subrutina STAADX regresa control a la rutina que lo llamó, el valor del AR se encontrará en la localidad de memoria apropiada, y los códigos de condición son actualizados como si fueran actualizados por una función normal de STAA con el mismo valor de operando. Los registros BR, XR y SP son preservados.

Una de las ventajas de estas subrutinas es que se provee un direccionamiento indexado donde el offset del XR es determinado dinamicamente en tiempo de ejecucción. Con el direccionamiento indexado

del MC6800 a su nivel normal, el offset debe ser determinado al tiempo de ensamblar.

Asuma que tienes ROM empezando en la localidad $2000, y escribe tu subrutrna como si fueran a correr en ROM. Puedes usar RAM empezando en la localidad $0020 para almacenamiento temporal. Codifica tu subrutina para que sea invulnerable a interrupciones.

6.83.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, F, G, H, I y J.

Se puede encontrar información adicional en la bibliografía con referencia 1,4,5, 8 y 9.

6.84.- PROCESO PRACTICO

Usa tu ingenio para desarrollar un buen procedimiento de prueba para tu paquete de subrutinas, y deberás preparar una presentación convincente a tu instructor. durante el periodo de clases.

Para esta práctica favor de entregar:

1.- Un listado de tu paquete de subrutinas completamentamente documentado en lenguaje ensamblador.

2.- Asegúrate de documentar la cantidad de RAM, ROM y stack usado en tu paquete y además explica brevemente como funciona tu paquete.

6.85.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

DIRECCIONAMIENTO DE REGISTROS

Nombre del alumno
No. de Credencial
Materia
Lab 8
Fecha

```
LINEA  ETIQUETA  COD OP   LENG ENSM     COMENTARIOS
                          NAM  LDARUTS
                          OPT  O,S
                          ORG  $2000     Inicio en la localidad
                                         $2000 .

       INICIO            SEI            Inhibe interrupciones
       LDAABX            TBA            Inicia Subrutina LDAABX

       LDAAAX            STX  SAVEX     Efectúa la subrutina LDAABX
                         TSTA           y LDAAX simultaneamente ya que
                         BEQ  2F        tienen las mismas característi-
       1H                INX            cas, excepto quel valor de B se
                         DECA           traspasa a A para ejecutar el
                         BNE  1B        offset.
       2H                BRA  3F

       LDAADX            PSHB           Inicia la subrutina para
                         STX  SAVEX     LDADX.
                         TSTA
                         BNE  1F
                         TSTB
                         BEQ  2F
       1H                INX
                         DECB
                         BLS  1B
                         DECA
                         BNE  1B
       2H                PULB

       3H                LDAA 0,X       Inicia el regreso a pro-
                         PSHA           grama principal y regre-
                         TPA            sa los registros a su
                         LDX  SAVEX     valor inicial.
                         TAP
```

```
                PULA
                RTS             Regreso a programa principal

STAABX          PSHA            Subrutina STAABX
                TBA             Pone el valor de BR en AR
                BRA 1F

STAAAX          PSHA            Inicia subrutina STAAAX y
1H              STX  SAVEX      efectua simultaneamente las
                TSTA            2 subrutinas de STAA de
                BEQ  2F         8-bits. pone lo que hay en
1H              INX             AR en la localidad indicada
                DECA            por el XR mas el offset de
                BNE  1B         BR o AR.
2H              BRA  3F

STAADX          PSHA            Inicia la subrutina de
                PSHB            almacenamiento tomando el
                STX  SAVEX      doble offset y almacenando
                TSTA            lo que hay en AR en las
                BEQ  2F         localidades establecidas
1H              INX             por requerimientos de
                DECB            programa.
                BLS  1B
                DECA
                BNE  1B
2H              PULB

3H              PULA            Estructura el regreso al
                STAA 0,X        programa principal y
                PSHA            regresa los registros a su
                TPA             valor incial.
                LDX  SAVEX
                TAP
                PULA

                RTS             Regresa a programa principal
```

6.86.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Proceso Práctico: Se otorgarán 55 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.84 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un numero mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 40 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

TOTAL = 45 puntos.

Las partes están señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....55 puntos
Punto (b)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

PRACTICA No.9 :

CREACION DE INSTRUCCION

6.91.- CONSIDERACIONES BASICAS

* SISTEMA A USAR
  Sistema Numérico Hexadecimal.
  Sistema Numérico binario.

* COMPONENTES A USAR
  Kit D-2. Microprocesador MC6800.
  Sistema Operativo MUDBUG o D3JBUG.

* LIMITACIONES TECNICAS
  Dependerá del sistema operativo a usar, es decir el MUDBUG o el D3JBUG.

* CONOCIMIENTOS REQUERIDOS

  USO DE SUBRUTINAS

  DIRECCIONAMIENTO INDEXADO

* CUESTIONARIO PREVIO

El programa sugerido en esta práctica es considerado con un grado de dificultad tal que se requiere mayor tiempo del que se ha estado estimado, por lo tanto no se requerirá un cuestionario previo.

6.92.- DISCUSION PRELIMINAR

Uno de los mas notables deficiencias al conjunto de instrucciones del MC6800 es la falta de instrucciones PSHX y PULX. Ya que el MC6800 es una máquina orientada para ser manejada por el STACK, el fabricante probablemente tenga que proveer de estas instrucciones. De hecho, estas instrucciones son disponibles en ediciones posteriores de la familia MC6800, pero están faltos en el 6800. En esta práctica tu harás las funciones PSHX y PULX disponibles para este sistema.

La subrutina PSHX (el cual es análogo a la instrucción PSHA o PSHB) pone el contenido de XR en el STACK y decrementa el apuntador de STACK dos veces en este proceso. La subrutina PULA (la cual es análoga a la instrucción PULA o PULB) toma el valor del STACK de las dos últimas localidades y las coloca en el registro índice XR, restando en dos el apuntador de STACK. Como las instrucciones originales PSHA y PULA, las subrutinas PSHX y PULX preservan el contenido de todos los registros del registro de condiciones. También preservan el AR y el BR, y la subrutina PSHX en especial preserva el XR. Cada subrutina desarrolla su función sin causar efectos posteriores indeseados.

Cuando la subrutina PSHX coloca los 16-bits del XR en el STACK, deberá almacenar el valor en forma regular con el byte de menor significancia almacenado primero y después el byte de mayor significancia. Esto es, la palabra de stack con la mayor localidad de memoria deberá contener el byte de menor significancia de la parte del XR y la menor localidad el byte de mayor significancia. La subrutina PULX, por supuesto deberá usar el mismo formato sacando el valor del STACK.

Las subrutinas PSHX Y PULX no son tan triviales para implementarse como parece. Ambas subrutinas deben de manipular el stack, y deberán hacerlo, por supuesto, sin dejar alternativa a una interrupción inesperada. Un problema que debes resolver es el hecho que cada subrutina regresará a un lugar erroneo en lo que a ti concierne. La subrutina PSHX, por ejmemplo, quiere colocar el valor de XR en el lugar que contiene la localidad de memoria de regreso al programa. Similarmente, la subrutina PULX quiere sacar el valor para XR de las dos localidades que estan arriba de las localidades que contiene el valor de regreso al programa principal. Naturalmente cada subrutina deberá ajustar el apuntador de stack propiamente antes de salir, y con este requerimiento tu diseño se ve ahora más complicado.

Asume que tienes ROM para tus subrutinas empezando en la localidad $2000, y escribe tus subrutinas de manera que puedan correr en esta area. Podrás usar el area de RAM empezando en la localidad $180 y terminando en $1FF para almacenamiento temporal si fuera necesario. el uso del stack es obvio.

Usa el programa maestro indicado a continuación para que demuestres tu paquete de subrutinas.

PROGRAMA MAESTRO PARA EJERCITAR LAS SUBRUTINAS PSHX Y PULX

```
INICIO1   ORG   $2400      Inicializa los registros para
          CLRA             predeterminar los estados. El AR y BR
          LDAB  =5         serán usados como contadores, pero el
          LDX   =$3203     valor de XR es arbitrario.

LOOP1     DEX              Modifica el valor de Xr, y actualiza los
          INCA             contadores en AR y BR. Entonces pone el
          DECB             valor de XR en el STACK, y se detiene, de
          JSR   PSHX       esa manera el usuario podrá checar sus
          SWI              registros, incluyendo el CCR y el STACK.

INICIO2   BNE   LOOP1      Regresa a hacer un ciclo para continuar
                           con el ejercicio de esta subrutina hasta
                           que el contador de BR este en cero. Note
                           que el programa se verá dañado si la
                           subrutina PSHX no preserva los registros
                           y los códigos de condición.

LOOP2     LDX =$FFFF       Establece el XR con un valor arbitrario
          INCB             el cual no debe estar en el STACK, y
          DECA             actualiza el AR y el BR. entonces saca
          JSR   PULX       el siguiente valor para el XR del STACK,
          SWI              y se detiene para que el usuario pueda
                           examinar convenientemente los registros y
                           el stack.

INICIO3   BNE   LOOP2      Regresa a hacer un ciclo para continuar
          BRA INICIO1      con el ejercicio de esta subrutina hasta
                           que el contador de BR este en cero. Note
                           que el programa se verá dañado si la
                           subrutina PULX no preserva los registros
                           y los códigos de condición. Además anote
                           que puede regresar si se usa el comando
                           "H" de MUDBUG.

          END              FIN DE PROGRAMA MAESTRO.
```

6.93.- INFORMACION ADICIONAL

Se puede encontrar información adicional para esta práctica en el apéndice A, D, F, G, H, I y J.

Se puede encontrar información adicional en la bibliografía con referencia 1, 4, 5, 6, 7, 8 y 9.

6.94.- PROCESO PRACTICO

Para esta práctica favor de entregar:

1.- Un listado de tu paquete de subrutinas completamentamente documentado en lenguaje ensamblador.

2.- Asegúrate de documentar la cantidad de RAM, ROM y stack usado en tu paquete y además explica brevemente como funciona tu paquete.

6.95.- REPORTE

FORMULACION DE LA PRACTICA.
PROCESO PRACTICO Y
RESULTADOS DE LA PRACTICA

CREACION DE INSTRUCCIONES

Nombre del alumno
No. de Credencial
Materia
Lab 9
Fecha

LINEA	ETIQUETA	COD OP	LENG	ENSM	COMENTARIOS
			NAM	PULPSH	
			OPT	O,S	
			ORG	$2000	INICIO DE PROGRAMA $2000 .
	RAM		EQU	$0180	
	ROM		EQU	$2000	
	SAVEX		EQU	$0180	
	PSHX		DES		Subrutina PSHX empieza aquí, con las instrucciones DES creo un espcacio donde pondré la localidad de regreso.
			DES		
			PSHA		Estas instrucciones son para almacenar el registro AR, el registro CCR y el XR, así podrán ser preservados cuando finalize la subrutina.
			TPA		
			PSHA		
			STX	SAVEX	
			TSX		Se transfiere el contenido del apuntador de satck al XR y usando las instrucciones LDA y STA en modo indexado se coloca la dirección de regreso al programa principal en el espacio anteriormente establecido, por lo tanto se deja las direcciones altas del stack listas para almacenar el contenido del
			LDAA	5,X	
			STAA	3,X	
			LDAA	4,X	
			STAA	2,X	

registro índice.

```
        LDAA SAVEX+1
        STAA 5,X
        LDAA SAVEX
        STAA 4,X
```

Se obtiene el valor de XR de su localidad de almacenamiento temporal y usando el registro AR se almacena en el STACK realizando asi la instrucción PSHX.

```
        LDX  SAVEX
        PULA
        TAP
        PULA
        RTS
```

Se recuperán los valores de los registros. estos són el AR, el XR y el CCR, y después regresa a programa principal.

```
PULX    PSHA
        TPA
        PSHA
```

Estas instrucciones almacenan el valor de AR y CCR en el STACK y antes de finalizar con esta subrutina será recuperados. el registro índice no necesita ser almacenado temporalmente ya que tomará el valor del stack.

```
        TSX
        LDX  4,X
        STX  SAVEX
```

Se obtiene el valor para XR del STACK y lo almacena en una localidad temporal mientras se actualiza el STACK en los siguientes pasos.

```
        LDAA 3,X
        STAA 5,X
        LDAA 2,X
        STAA 4,X
        LDX SAVEX
```

Se mueve la dirección de retorno a la localidad mas alta del apuntador de stack, así la última instrucción de esta subrutina correspondera a las localidades de regreso a programa principal y dejando 2 localidades menos que cuando la subrutina fue llamada.

```
        PULA
        TAP
        PULA
```

Se recuperan los valores de los registros AR y CCR.

```
        INS
        INS
        RTS
```

Brinca 2 localidades en el stack para ajustar el número de localidades de

memoria en el stack y hacer que el apuntador de stack apunte a las localidades de retorno a programa principal.

6.96.- EVALUACION DE LA PRACTICA

La secuencia para evaluar la práctica se sugiere se haga como se indica a continuación:

a) Respecto al Proceso Práctico: Se otorgarán 55 puntos al presentar satisfactoriamente al instructor de laboratorio lo realizado en laboratorio. El instructor verificará el programa y todos los pasos del programa, se recomienda al instructor seguir los pasos de la sección 6.92 para verificar el programa del alumno. Checar los siguientes tópicos en el programa:

* Obtener el Resultado deseado incluye:
  - Memoria Afectada
  - Resultado final de los registros.

De acuerdo al programa del estudiante ya que es posible que su programa ocupe un número mayor o menor de bytes de PSEUDOROM y/o use otros registro para lograr el objetivo.

b) Con Respecto al Reporte: Se otorgarán 45 puntos al entregar el reporte con todas las partes acreditadas. Esto es :

* Lenguaje ensamblador y código máquina 40 puntos. Por cada byte usado de mas se tomará 2 punto y viceversa , por cada byte menos que la respuesta del instructor se otorgará un punto extra.

* Memoria afectada.- Reporte de la memoria afectada 5 puntos.

TOTAL = 45 puntos.

Las partes estan señaladas al final de la práctica. Limpieza y Presentación del reporte causarán baja de hasta 10 puntos sobre la evaluación final de la práctica.

SUMARIO

Punto (a)....55 puntos
Punto (b)....45 puntos

TOTAL.....100 puntos

El tiempo estimado para realizar este proceso es de una semana.

VII.- PROGRAMACION APLICADA

VII.- PROGRAMACION APLICADA

INTRODUCCION

Este capítulo esta dirigido para aquellos estudiantes o ingenieros miembros de la comunidad técnica que tienen contacto con el diseño de microcomputadoras y esas aplicaciones mecánicas, hidráulicas que ahora pueden ser llevadas a cabo mas efectivamente y confiablemente con los microprocesadores.

Una de las metas de este capítulo es clarificar algunos detalles para determinar si un sistema de microcomputadora es apropiado, y si es así, ayudar a entender los problemas de diseño y debugging. La forma de lograr esto es presentando 5 prácticas con un contenido de simulación de sistemas operativos, electrónicos y mecánicos.

Buscando por la posiblidad de usar microcomputadoras para reemplazar otras técnicas en controladores industriales, uno de los primeros requisitos para convertir controladores mecánicos a un diseño de microcomputadora es proveer la interfase electromecánica. Una vez que la señal mecánica puede ser controlada por una señal eléctrica, usando solenoides o motores, y el desarrollo pueda ser monitoreado por sensores eléctricos, la microcomputadora, entonces, podrá ser usada.

Las secuencias funcionales de control usadas anteriormente en el diseño mecánico pueden ser traducidas a instrucciones de computadora.

Es clásico, cuando se diseña un controlador por microcomputadora, proveer la interfase mas simple para que cuando sea "ENCENDIDO" (+5V), el motor arranque, la válvula de seguridad X se abra, la alarma suene, y cuando sea cero volts, esos mismos componentes se encuentren en su estado pasivo. Todas esas funciones de abrir, cerrar, arrancar, limitar, etc.., están delegadas a la microcomputadora y a la lógica dentro del "SOFTWARE". Esta técnica frecuentemente resulta en el diseño de un sistema con un costo mínimo en lo referente a circuitos "HARDWARE". Si el microcomputador puede manejar todas las tareas encomendadas en el tiempo requerido, esto puede resultar en el diseño mas apropiado debido al costo.

Sin embargo, es posible que algunas rutinas de "SOFTWARE" tomen mucho tiempo y algunos sistemas electrónicos puedan estar construidos para ser seguros contra fallas, en otras palabras, algunos componentes se requiere que sean desconectados al momento de una falla eléctrica, por lo tanto, el diseñador requiere incluir algun tipo de lógica "HARDWARE" para tener esta protección.

Si el control de la máquina que ha sido construida es hecho con una microcomputadora y su "SOFTWARE", en lugar de lo tradicional, la computadora proveera mas aplicaciones, siendo así mas flexible, ser mas confiable y también con menor costo.

PROGRAMACION APLICADA

Un sistema de control tradicional, diseñado con componentes electromecánicos como son interruptores, relevadores, etcc.., son diseñados normalmente para una aplicación específica. Si los requisitos de la aplicación cambiaran o una aplicación nueva es requerida, el sitema de control ("HARDWARE") deberá ser reconstruido o al menos realizar una reconección entre chips. En un sistema de control en base a microcomputadoras aún los grandes cambios en las funciones del sistema pueden ser soportadas por cambios en la programación.

CONSIDERACIONES

Existen 3 procesos esenciales que deben ser seguidos desde la posibilidad de usar microprocesadores en un proceso hasta decidir firmemente el uso de una microcomputadora en tus sistemas:

1.- EVALUACION DEL PROCESO.

2.- VERIFICACION DEL DISEÑO DEL PROCESO.

3.- DISEÑO DETALLADO DEL PROCESO.

EVALUACION DEL PROCESO:

Es necesario analizar si una microcomputadora puede ser usada para realizar una aplicación específica y además familiarizarse con todas las capacidades de la microcomputadora y así tener la evaluación adecuada.

Entre las capacidades de la microcomputadora se consideran monitorear datos, hacer cálculos, tomar desiciones y tener señales de control a la salida.

El diseñador tomará en cuenta si puede usarse el microcomputador en base a la velocidad de respuesta requerida por el proceso, la eficacia y el costo del uso de una microcomputadora.

VERIFICACION DEL DISEÑO DEL PROCESO

En este paso se seleccionan las capacidades de determinada familia de chips en base a los siguientes puntos:

1.- La habilidad de los componentes periféricos de esa familia. (Requerida para interfase con componentes externos).

2.- La habilidad y capacidad de desarrollo de un sistema en sus dos conceptos: Material (Hardware) y Programación (Software).

3. Requisitos y posibilidades para manejar lenguajes de alto nivel. Ejemplo: BASIC, FORTRAN, PASCAL, ENSAMBLADOR.

En base a esto se determina si un diseño práctico puede ser implementado.

DISEÑO DETALLADO DEL PROCESO

Una de las desiciones mas importantes para realizar el diseño de un sistema en base a microcmputadoras es:

ANALISIS DEL COSTO

CRITERIOS GENERALES PARA EL DISEÑO.

* Planteo del problema y especificaciones.
* Aspectos que influyen en la solución deseada.
* Aspectos que excluyen la solución deseada.
* Algoritmo solución.
* Información
  -- Como han sido resueltos problemas similares.
  -- En donde existe información al respecto.
  -- Cuales son las tendencias actuales al respecto.
  -- Obtención de diagramas eléctricos y a bloques que describan:
     - o La configuración del sistema disponible.
     - o Especificaciones del material disponible o actual.

-- Respecto a sistemas matemáticos y/o físicos.

* Formulación del Proyecto.

-- Diseño del modelo teórico y validación.
  - o Cálculo de ciertos componentes.
  - o Diseñar de acuerdo a situaciones reales y prácticas.

-- Elaboración de diagramas de flujo.
  - o En forma de trabajo bien definido.
  - o Debidamente programado con respecto al tiempo.
  - o Consideraciones relativas al factor económico.

* Realización del proyecto.

-- Realización técnica.
  - o Configuración de materiales.
  - o Programación.

-- Pruebas y correcciones.
  - o Simulaciones.
  - o Puesta a punto.

* Implementación del proyecto.

-- Pruebas, correcciones y ajustes.

-- Comportamiento ante los cambios.

-- Evaluación de funcionamiento y ajustes.

-- Documentación del proyecto.
  - o Descripción del funcionamiento y eficiencia.

DIVISION POR PRACTICAS

Cambiando el formato de las prácticas que se han venido desarrollando, se presentarán varios problemas y algunas sugerencias de los puntos que habrían de tomarse en cuenta para así resolver adecuadamente y con ayuda de microprocesadores el problema en cuestión. Como usted se dará cuenta, las primeras 3 prácticas no presentan mayor problema para aquellas personas que han trabajado y han llegado a este punto, no necesitando de sugerencias para realizar las mismas.

De ahora en adelante el estudiante será capaz de desarrollar programas en microprocesadores y con las bases obtenidas no solamente en el MC6800, sino que con una breve vista y análisis de los comandos de otros microprocesadores será capaz de utilizarlos con maestría.

PROGRAMACION APLICADA

PRACTICA No.1 :

APLICACION DE SISTEMA OPERATIVO

PRESENTACION DEL PROBLEMA

Para esta práctica tu escribirás un programa modular que lee e interpreta entradas de el teclado de 24 caracteres en la microcomputadora KIT-D2 del laboratorio. Tus pruebas y depuración de programa para esta práctica sera de tu agrado porque podrás observar el resultado de tu programa al tiempo que se este ejecutando interactivamente. Iremos a las especificaciones del programa pero primero discutimeros material de soporte.

Teclado y componentes de entrada similares están interconectados a microprocesadores en un gran número de aplicaciones. De hecho, casí todo artículo basado en microprocesadores que requiere o acepta entradas de personas, tendrá entradas en forma de teclado o en algun equivalente.

Si tu quieres que un microprocesador lea entradas de un teclado, deberás primero de alguna manera decodificar y evitar la repetición de tecla de las entradas. Consideremos primero la decodificación. Si una persona presiona una tecla del teclado y asi cierra un interruptor, el microprocesador primero deberá reconocer el hecho de que una tecla ha sido presionada. Entonces el microprocesador decodificará la entrada para determinar que tecla en particular fue presionada. Hay varias maneras de decodificar la entrada de un teclado. Por un lado, el microprocesador puede desarrollar la decodificación via "software" (programación), y por el otro lado la decodificación puede ser hecha enteramente por circuitería "hardware" que simplemente presente un código de 8-bits (código ASCII) en paralelo o en serie a través de un puerto para el microprocesador. Así entre los dos extremos incluimos espacio en ROM y posibilidad de ejecución contra espacio en el circuito, consumo de potencia, y costo de partes. Adicionando que uno debe considerar el costo del diseño, desarrollo, documentación y mantenimiento del "software" contra los costos análogos de diseño, desarrollo , documentación y mantenimiento del "hardware".

Además, decodificar entradas desde un teclado, el diseñador del sistema deberá controlar el rebote de la tecla. Considera una tecla mecánica sencilla que le permita a una persona cerrar y abrir un circuito ya sea presionando o liberando la tecla. Tipicamente, el contacto entre puntos en el interruptor rebotan por varios milisegundos cuando la tecla es presionada. Similarmente, arcos eléctricos pueden causar que el interruptor rebote cuando es liberada.

En algunas aplicaciones, El rebote de los interruptores no es de importancia relevante. Por ejemplo, Un interruptor que rebota por varios milisegundos normalmente no puede causar algun impacto si es usado en una acción mecánica. Si la tecla esta siendo leida por el

microprocesador ,el rebote del interruptor puede causar serios problemas. Ya que un microprocesador es demasiado rápido, este probablemente reaccionará como si el interruptor (tecla) hubiera sido presionada varias veces. Imagina tratar de teclear entradas al microprocesador en un teclado que no fue diseñado contra rebotes. Si tu tecleas el 3, por ejemplo, el microprocesador pensará que tu has tecleado el 33, 333, o 3333 por el rebote de la tecla.

El efecto del rebote en las teclas puede ser eliminado haciendo mejoras al "hardware", o puede ser eliminado por medio de programación. Cualquier metodo es igualmente efectivo, y usualmente hay una mezcla de las dos soluciones para optimizar el resultado.

Cada KIT-D2 tiene 24 teclas que son interconectadas a el microprocesador a través del PIA. Un PIA es un componente muy poderoso y complejo que provee puertos de entradas y salidas en paralelo. El PIA consiste de 2 lados, el A y el B, los cuales son casi equivalentes.

Cuando el PIA es inicializado propiamente, el usuario puede liberar datos de salida mediante la sencilla operación de almacenar el valor en cierta localidad de memoria que corresponde a uno de los registros del PIA. Similarmente, un usuario puede leer entradas en paralelo del PIA simplemente leyendo una localidad de memoria en particular que correspondería a uno de los registros del PIA. El PIA es un componente altamente flexible que puede ser controlado dinamicamente a través de programación, pero tu usaras el PIA como un puerto de entrada y salida para propósitos de esta práctica.

El diagrama que contiene el teclado en un KIT-D2 se muestra en el apéndice J. Tu te darás cuenta que no hay decodificación y antirebote en el circuito hardware. Por lo tanto tu programa deberá proteger contra rebote y decodificar las entradas con una pequeña ayuda del hardware mostrado.

Note que en el circuito eléctrico los bits PB0, PB1, PB2, PB3, PB4, y PB5 son salidas desde el PIA a el teclado. (Si, tu programa tendrá salidas al teclado, aunque parezca extraño al principio). Estos 6 bits de datos del puerto B corresponden a los bits 0 a 5 de la localidad BDATA (ej.$8022), y tu programa sacará datos al teclado simplemente al almacenar valores a esa localidad.

Note también que en el circuito, los bits PB6 y PB7 son salidas para el lado B del PIA y son conectados a un multiplexor de 4 canales MC14539, conocido también como selector de datos, (adjunto especificaciones del multiplexor) Los valores de PB6 y PB7 son usados para seleccionar de una a cuatro lineas de entrada (X0, X1, X2 o X3) a la salida Z del MC14539. Tu programa podrá por lo tanto seleccionar cualquiera de las lineas en el el MC14539 almacenando el valor apropiado en la localidad BDATA.

Finalmente, note que en el circuito la salida Z del MC14539 esta conectada a la entrada PA7, la cual es el bit 7 del lado A del PIA. PA7 corresponde a la localidad ADATA (ej:$8020), de esa manera tu programa podrá leer el valor de la salida Z del MC14539 sencillamente al tomar datos de la localidad ADATA.

Como ilustra el circuito, el teclado es organizado electricamente en matriz de 6 por 4. Presionando una tecla conecta un renglón con una columna, así la señal la señal del renglón es pasada a la columna cunado la tecla es pulsada. Los renglones son controlados por las salidas del PB0 al PB5 y las columnas son entradas al selector MC14539.

Si tu piensas en lo que has leido, seras capaz de inventar un metodo sencillo para decodificar las entradas del teclado. Tu programa puede controlar los valores de los renglones y además controlar el 14539 para leer los valores en las columnas, una columna a la vez. Por lo tanto, tu programa deverá ser capaz de determinar cual tecla en particular fue pulsada.

Ahora que has entendido como decodificar las entradas, tu entenderás el siguiente algoritmo para evitar el rebote en las teclas.

ALGORITMO ANTIREBOTE.- Este algoritmo evita el rebote al pulsar y liberar una tecla.

D1.- (Tecla presionada) Si no es presionada, esperar hasta que alguna lo sea. Cuando una tecla es presionada, decodificarla y proceder con el paso D2.

D2.- (inicializa para antirebote) Pone K <--6. (K es el contador que sera usado para controlar el ciclo de antirebote que se indica a continuación).

D3.- (Esperar y leer la tecla) Retrasa 2 milisegundos. Después checa para verificar si la tecla que fue decodificada sigue aun presionada. Si no lo esta, regresa el control a D1. De otra manera, continua con el paso D4. (el retardo de tiempo permite el rebote mecánico. Si la misma tecla no ha sido cerrada después del retardo, la tecla esta rebotando. En este caso, el algoritmo empieza de nuevo en el paso para que una tecla se detenga de rebotar).

D4.- (itera) Pone K <-- K-1. Si K>0, cicla de nuevo al paso D3. De otra manera, procede al paso D5. (si la tecla no ha sido leida en rebotes por 12 milisegundos, el algoritmo considera que la tecla fue firmemente pulsada).

D5.- (Transfiere la entrada) Encuentra el caracter ASCII que corresponde a la tecla, e imprime el caracter en la terminal. (Casi cualquier tipo de proceso puede ser desarrollado aquí).

D6.- (inicializa el antirebote) Pone K<--6. (K sera usado para controlar el ciclo de antirebote en la liberación de la tecla).

D7.- (espera y lee la tecla) Retrasa 2 milisegundos. Después checa si la tecla que fue presionada y decodificada en el paso D1 sigue siendo presionada. Si lo está, regresa al paso D6. De otra manera va al paso D8. (El control permanece en ciclo en los pasos D6 y D7 hasta que la tecla sea liberada).

D8.- (itera) Pone K <-- K-1. Si K > 0, regresa al paso D7. De otra manera termina el algoritmo. (si la tecla fue encontrada liberada por espacio de 12 milisegundos, el algoritmo considera a la tecla completamente suelta).

Finalmente estamos listos para discutir las especificaciones para tu programa para esta práctica. Tu programa deberá hacer lo siguiente:

1.- Inicializa el PIA. El PIA será utilizado para buscar, decodificar el teclado.

2.- Usa la subrutina de MUDBUG CRLF para mandar un retorno de carro y alimentación de linea a la terminal.

3.- Leer la entrada desde el teclado, y use la subrtuina disponible en MUDBUG para desplegar el caracter en la terminal.

4.- Cicle al paso 3. Tu programa estará en un ciclo infinito, pero tu podrás terminar tu programa con una interrupción no enmascarda NMI.

En alguna parte de tu programa deverás tener una rutina para retardo de 2 milisegundos, Hay muchas maneras para implementarla con contadores/timers exteriores e interrupciones, pero no queremos implementar esas técnicas en este laboratorio. Por lo tanto, usarás un metodo efectivo, ciclar para efectuar el retardo de tiempo. Cada instrucción en el MC6800 requiere de un número de ciclos de reloj para su ejecución, y el tiempo de reloj por ciclo es de 1.6276 microsegundos. Por lo que podemos implementar un ciclo de retardo con las siguientes instrucciones:

```
RETARDO     LDX  =COUNT      3 CICLOS
CICLO       DEX              4 CICLOS
            BNE  CICLO       4 CICLOS
```

614.4 CICLOS = 1.0 milisegundo. Las dos instrucciones en el ciclo son ejecutadas COUNT veces si COUNT es mayor que cero.

Tiempo = (3 + COUNT * 8) * (1.6276) microsegundos.

Si ponemos el Tiempo en 2.0 milisegundos encontramos que COUNT sera igual a 153. Si quieres hacer tu retardo de tiempo con diferentes

instrucciones, deverás calcular nuevamente el retardo que esta generará.

Asume que tienes ROM para tu programa iniciando en la localidad $2000, y puedes usar localidades de RAM de la $0 a la $FF, por supuesto podrás usar el STACK.

Deverás demostrar tu programa funcionando al instructor y no se aceptará el reporte sin haberlo presentado al instructor en tiempo de laboratorio.

Entregar para el laboratorio el programa en código máquina y lenguage ensamblador. Indique cuantas localidades de memoria fueron ocupadas en ROM y en RAM.

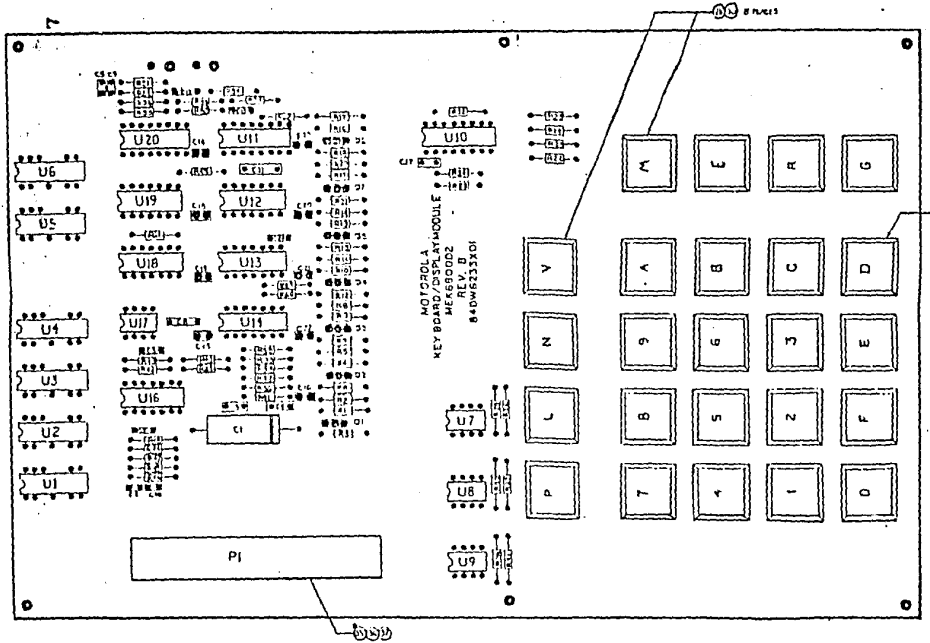


FIGURE A2-a. Keyboard/Display Module Assembly

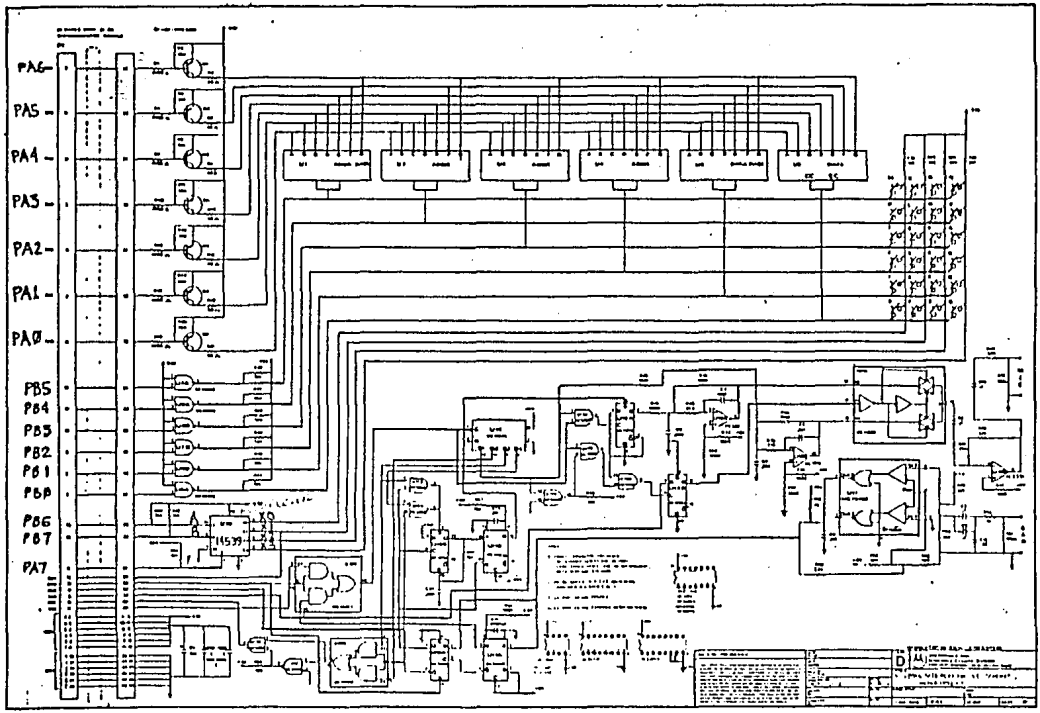


FIGURE A3-1. Keyboard/Display Module Schematic

MULTIPLEXOR MC 14539B

DUAL 4-CHANNEL DATA SELECTOR/MULTIPLEXER

The MC14539B data selector/multiplexer is constructed with MOS P-channel and N-channel enhancement mode devices in a single monolithic structure. The circuit consists of two sections of four inputs each. One input from each section is selected by the address inputs A and B. A "high" on the Strobe input will cause the output to remain "low".

This device finds primary application in signal multiplexing functions. It permits multiplexing from N-lines to 1-line, and can also perform parallel-to-serial conversion. The Strobe input allows cascading of n-lines to n-lines.

- Quiescent Current = 5.0 nA/package (typical) @ 5 Vdc
- Noise Immunity = 45% of V_{DD} typical
- High Fanout: > 50
- Input Impedance = 10^{12} ohms typical
- Supply Voltage Range = 3.0 Vdc to 18 Vdc
- Capable of Driving Two Low-power TTL Loads, One Low-power Schottky TTL Load or Two HTL Loads Over the Rated Temperature Range

MAXIMUM RATINGS (Voltages referenced to V_{SS})

Rating	Symbol	Value	Unit
DC Supply Voltage	V_{DD}	-0.5 to +18	Vdc
Input Voltage, All Inputs	V_{in}	-0.5 to V_{DD} + 0.5	Vdc
DC Current Drain per Pin	I	10	mAdc
Operating Temperature Range — AL Device CL, CP Device	T_A	-55 to +125 -40 to +85	°C
Storage Temperature Range	T_{stg}	-65 to +150	°C

TRUTH TABLE

ADDRESS INPUTS		DATA INPUTS					OUTPUTS
B	A	X3	X2	X1	X0	ST,ST'	Z, W
X	X	X	X	X	X	1	0
0	0	X	X	X	0	0	0
0	0	X	X	X	1	0	1
0	1	X	X	0	X	0	0
0	1	X	X	1	X	0	1
1	0	X	0	X	X	0	0
1	0	X	1	X	X	0	1
1	1	0	X	X	X	0	0
1	1	1	X	X	X	0	1

X = Don't Care

CMOS MSI

ILOW POWER COMPLEMENTARY MOS!

DUAL 4-CHANNEL DATA SELECTOR/MULTIPLEXER

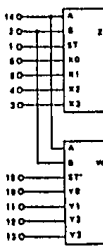
L SUFFIX
CERAMIC PACKAGE
CASE 620

P SUFFIX
PLASTIC PACKAGE
CASE 648

ORDERING INFORMATION

MC14539B — Suffix Denotes
- L — Ceramic Package
- P — Plastic Package
- A — Extended Operating Temperature Range
- C — Limited Operating Temperature Range

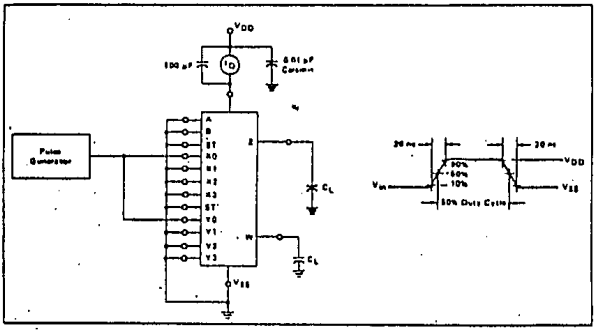
BLOCK DIAGRAM



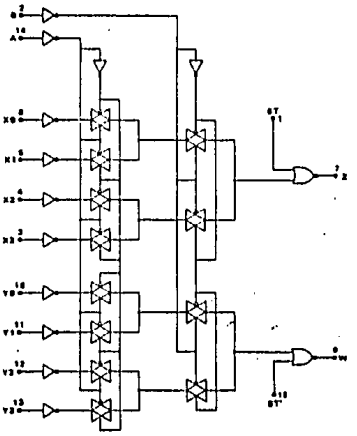
V_{DD} = Pin 16
V_{SS} = Pin 8

DIAGRAMA LOGICO DEL MULTIPLEXOR MC 14539B

FIGURE 3 – POWER DISSIPATION TEST CIRCUIT AND WAVEFORM



LOGIC DIAGRAM



PROGRAMACION APLICADA

PRACTICA No.2 :

APLICACION COMERCIAL

PRESENTACION DEL PROBLEMA

Para este laboratorio tu crearás un probador de reflejos como si fuera un juguete, extendiendo el programa que escribiste en el laboratorio anterior. Tu probador de reflejos se comunicará con el jugador a través de números hexadecimales de 16 bits, quizás tu juguete no sea de gran aceptación comercial. Sin embargo, tu juguete demostrará algunas características que comunmente se encuentran en productos comerciales que son algunas veces anunciados en televisión. Niños o adultos usarán tu juguete para probar sus reflejos (y su paciencia), y el juguete evaluará su efectividad y los premiará o castigará segun sea el caso.

Cuando tu pones a ejecutar tu juguete por primera vez, deberá inicializar el PIA que controla el teclado y deberá también desplegar una introducción en una nueva linea en la terminal: "Hola, Yo soy rapidín, tu juguete de reflejos." Y entonces hara lo siguiente:

1. Sacar un CRLF a la terminal. Esta salida sirve para notificar al jugador que el juguete de reflejos esta listo para arrancar con la prueba. Observa que la subrutina CRLF en MUDBUG será de mucha utilidad aquí. En general, tu desearás usar las subrutinas de MUDBUG para tareas de I/O de este laboratorio.

2. Retarde y no hacer nada por un periodo de tiempo en un rango aproximado de tiempo de 0 a 64 segundos. Tu puedes efectuar este retardo de tiempo muy fácil como sigue: Primero, usa el generador de números aleatorios anexo a esta práctica para computar un número aleatorio en el rango de 0 a 255, y entonces retrasa 250 milisegundos el número aleatorio de veces. Si el NUMERO = 0 sera mejor que se use como si fuera el valor de 256.

3. Selecciona un caracter aleatorio de entre los caracteres que existen en el teclado del KIT- D2. Entonces despliega el caracter seleccionado y 2 espacios en blanco a la terminal. Tu puedes facilmente seleccionar un número aleatorio como sigue: Primero, use el generador de números aleatorios anexo a esta práctica para computar un nuevo número en el rango de 0 a 255. Entonces, ya que hay 24 teclas en el teclado, compute el modulo 24 para obtener un número aleatorio en el rango de 0 a 23. Este número aleatorio designa uno de los 24 caracteres del teclado.Note que hay dos diferentes "E"'s, así que la posibilidad de seleccionar la "E" es doble que cualquier posibilidad de seleccionar otra tecla. Precaución: No cometa el error de usar el número aleatorio del paso 2 en este paso; un programa que usa solamente un número aleatorio es fácil de predecir para jugadores maestros.

4. Ponga REFLEJO <---0 . REFLEJO es una variable de doble presisión que es usada para contar aproximadamente cuantas centecimas de segundo se requieren para que el jugador pueda identificar el caracter que ha sido desplegado a raiz del paso 3 y en presionar la tecla correspondiente en el teclado. El valor de el tiempo de REFLEJO sera actualizado automaticamente por una rutina de interrupción de tiempo-real que tu deberás escribir como parte del proyecto de esta práctica, y REFLEJO funcionará como un reloj de tiempo real con unidades de centecimas de segundo. La interrupción causada por el reloj sera discutida mas tarde.

5. Examine el teclado. Si tu encuentras que ninguna tecla ha sido presionada, proceda al paso 6. Si alguna tecla ha sido presionada , por otro lado, haga sonar una alarma con la campana de la terminal. imprima los caracteres "tramposos nunca prosperan" en la terminal, y entonces vaya al paso 9. Una tecla es considerada presionada solo si pasa completamente la lógica anti-rebote en tu programa buscador-de-tecla-presionada, así que una tecla rebotando no se considera como si fuera realmente presionada. Este paso detecta a la gente que hace trampa presionando una tecla antes de tiempo colocando un tiempo fenomenal.

6. Si REFLEJO < LIMITE (donde LIMITE = 3,000), proceda al paso 7. De otra manera, imprima en la terminal "Tu perdiste", y vaya al paso 9. Recuerda que el valor de REFLEJO esta siendo actualizado automaticamente por tu rutina de interrupción de tiempo-real, de esta manera encontrarás que REFLEJO >= LIMITE después de 30.00 segundo de tiempo. Se especialmente cuidadoso para asegurarte que tu prueba para REFLEJO < LIMITE es certera para todos los casos posibles, tu puedes codificar facilmente un error que puede pasar posiblemente en un sistema manejado por interrupciones con eventos asíncronos. Una interrupción puede ocurrir entre dos instrucciones, de esa manera el valor en doble presisión de REFLEJO Puede cambiar entre dos instrucciones cualquiera que estas sean. Tu programa deberá ser escrito de tal manera que pueda modificar una sola linea de código y así cambiar el valor de 3,000 en este paso si así se deseara y colocar un valor del rando de 0 < LIMITE < 65,536.

7. Examine el teclado. si no hay tecla presionada, regrese al paso 6. Si la tecla es presionada, proceda al paso 8. Note de nuevo que una tecla es considerada presionada si pasa la prueba lógica de anti-rebote. Si el jugador toca una tecla muy rapidamente en lugar de hacerlo suavemente, la tecla rebotara por un largo tiempo y asi el jugador resultará penalizado. Note que este paso puede facilmente requererir hasta 12 milisegundos, y el valor de REFLEJO puede avanzar de 2,999 a 3,001 mientras el paso esta siendo ejecutado. De hecho, este paso puede requerir mas de 12 milisegundos porque varias teclas puedieran estar rebotando al mismo tiempo. El punto es que tu programa quizás nunca encuentre REFLEJO = LIMITE en el paso 6, así que la instrucción CPX no funcionará allá.

8. Cheque para ver si la tecla que fue presionada fue la tecla correcta. (no olvides que hay doe "E"'s y que ambas respuestas son correctas como respuesta). Si la tecla erronea es presionada, imprima "lo echaste a perder". De otra manera, use las subrutinas apropiadas de MUDBUG para imprimir "xxxx centésimas de segundo" donde "xxxx" es el valor hexadecimal del tiempo de REFLEJO. Se cuidadoso para asegurarte de que la variable de tiempo REFLEJO no sea actualizada cuando se imprima su valor, y asegúrate que el valor que imprimes es bastante cierto y refleje el tiempo del paso 4 al inicio del paso 8.

9. Avance a una nueva linea en la terminal, e imprima "Quieres jugar de nuevo?" seguido de 2 espacios. Entonces lee el caracter del teclado de la terminal, no del teclado del KIT-D2. Si la entrada es "S" o "s" entonces regresa al paso 1 y probarás de nuevo el reflejo. De otra manera simplemente detente.

Nota que tu programa nunca checa por la liberación de la tecla en el teclado ya que el programa esta solamente tomando en cuenta la reacción del jugador para presionar la tecla. Así que parte de tu práctica anterior podrá ser descartada.

Ahora considera la interrucpión de reloj a tiempo real en algo mas de detalle. Cada KIT-D2 en el laboratorio tiene un reloj libre de 300 Hertz que esta conectado a la entrada CA1 (pin 40) del PIAque ocupa las localidades de memoria del $8004 al $8007. Inicializando el PIA propiamente, puedes hacer generar al PIA una señal de IRQA (interrupción) en pin 38 del PIA cada vez que se detecte una caida en el pin del CA1. La salida de la interrupción IRQA del PIA es conectada al IRQ del MPU (pin 4 del MC6800), así podrás facilmente programar el PIA y el MPU para proveer un reloj libre de 300-Herts con interrupción a tiempo-real.

Tu deberás estudiar la documentación del PIA para obtener completo entendimiento de la entrada del CA1, la salida IRQA, y de varios bits de los registros de control del PIA. Tu deberás también estudiar tu documentación en el M6800 para refrescar tu memoria con respecto a las interrupción IRQ y IRQ enmascarada. Encontrarás que hay al menos dos maneras de habilitar y deshabilitar el reloj que controla la interrupción.

Cuando tu escribas el programa, deberás particularmente considerar cuando habilitar y deshabilitar la interrupción. Una interrupción que es mantenida viva (habilitada) cuando se deba de desactivar causará errores aleatorios que quizás no aparezcan durante tus pruebas. El bit de interrupción enmascarada es considerado con basura cuando tu programa empieza a ejecutarse, así que deberás inicializar ese bit como una de tus primeras operaciones.

Cuando escribas la rutina que responde al manejo de interrupciones y así al reloj de interrupciones deberás tener ciertas consideraciones en mente. Primero que nada, recuerda que una rutina de manejo de

interrupciones es ejecutada asincronamente con respecto al resto del programa, así su ejecucción completa puede ocurrir entre la ejecucción de 2 instrucciones cualquiera de tu programa principal. Una rutina de manejo de interrupciones tiene que ser tan corta y tan rápida como sea posible para minimizar los efectos de tiempo que su ejecucción tiene en el resto del programa, y obviamnete deberás preservar todos los registros y valores del código de condiciones.

Tu rutina de manejo de interrupciones será ejectuda 300 veces por segundo cuando el reloj de interrupciones de tiempo-real este habilitado, y si tu quieres incrementar el valor de REFLEJO 100 Veces por segundo, por lo tanto, tu deberás convertir la interrupción de 300 Herts a interrupciones de 100 Hertz.

Si tu estudias tu manual de MUDBUG cuidadosamente aprenderás como tu programa puede establecer un vector de interrupción indirecto en el area de RAM y así permitir que tu propio programa recupere el control cuando una interrupción IRQ aparezca. Recuerda que que MUDBUG automaticamnete reinicializa todos los vectores indirectos de interrupción o el manejo de interrupciones en general.

Asume que tienes ROM para tu programa (incluyendo todas las tablas, subrutinas, mensajes, y rutinas de manejo de interrupción) empezando en la localidad $2000, y escribe tu programa de manera que pueda correr en area ROM. Podrás usar localidades de la $0 a la $FF para almacenamiento temporal si así lo requieres.

GENERADOR DE NUMEROS ALEATORIOS

Esta subrutina genera un número aleatorio de 8-bits y lo pone en AR. La subrutina RNG genera una secuencia de números aleatorios a través de metodos lineales con un multiplicador de 261, y constantes aditivas de 13,849, y módulos de 65,5636. Estos valores han sido cuidadosamente seleccionados para proveer una secuencia aleatoria con correlación serial mínima.

Secuencia a llamar : JSR RNG

Condiciones de regreso: El número aleatorio esta en AR, y los bits mas bajos del registro CCR son destruidos. El BR, XR , SP y el CC.I son preservados.

La subrutina RNG puede ser colocada para empezar donde se desée.

```
            ORG   RAM
RANDOM      RMB   2
SAVEX       RMB   2
            ORG   ROM
RNG         PSHB
            STX   SAVEXR
            CLRB
            CLRA
            LDX   =261

1H          ADDB  RANDOM+1
            ADCA  RANDOM+0
            DEX
            BNE   1B

            ADDB  =C.AND.$FF
            ADCA  =C.LSR.8
C           EQU   13849
            STAB  RANDOM+1
            STAA  RANDOM+0

            PULB
            LDX   SAVEXR
            RTS
```

PROGRAMACION APLICADA

PRACTICA No.3 :

APLICACION DE SISTEMA OPERATIVO

PRESENTACION DEL PROBLEMA

Tu misión en esta práctica es escribir, ensamblar, depurar y demostrar una subrutina "utilidad" que pueda ser usada para imprimir un programa o una parte del programa en formato de lenguaje máquina. El listado del lenguaje máquina que tu subrtuina reproducirá será similar en formato a la porción objeto de un listado en ensamblador.

La subrutina DECODE usa tres parámetros, y estos parámetros son INICIO, ALTO, y ERRORALTO. Los parámetros INICIO y ALTO le dicen a la subrutina DECODE que area de memoria deberá ser procesada, y el parámetro ERRORALTO le dice a la subrtuina DECODE que hacer si encuentra un byte con código operacional que no contiene un código operacional válido.

El parámetro INICIO especifica la dirección de memoria del primer byte de la primera instrucción que la subrutina DECODE debe procesar. El parámetro ALTO similarmente especifica la última dirección que debe ser procesada del area de memoria descrita. Tu subrutina deberá continuar procesando instrucciones hasta que haya procesado la instrucción que incluye el parámetro ALTO. Note que el parámetro ALTO puede especificar una dirección que sea la segunda o tercera palabra de la última instrucción a procesar, y no necesariamente la primera palabra de la última instrucción.

El parámetro ERRORALTO le dice a la subrutina DECODE que deberá hacer si encuentra un código operacional que no contiene un valor válido para códigos operacionales. Si el parámetro ERRORALTO es cero, la subrutina DECODE deberá seguir adelante procesando hasta haber procesado toda el area descrita por los parámetros INICIO y ALTO. Si el parámetro ERRORALTO no es cero, deberá detenerse tan pronto y haya procesado la última instrucción.

Cuando la subrutina DECODE recibe control, El Registro índice contiene la dirección de memoria de un bloque de 5 bytes de memoria. Los primeros 2 bytes de ese bloque contienen el parametro INICIO, los bytes 3 y 4 contienen el valor del parametro ALTO y el último byte contiene el valor del parámetro ERRORALTO.

Al regresar de la subrutina, El XR y el BR deberán ser preservados, y el registro AR deberá contener una bandera indicadora de estado. El valor de AR deberá ser cero si la subrutina DECODE no encontro códigos operacionales inválidos, y un valor diferente de cero si llega ha encontrar alguno. El bit Z del CCR deberá reflejar el estado final de valor en AR, y los otros bits del CCR no son importantes,

Los siguientes ejemplos ilustran el formato de la salida que tu subrutina deberá generar. En este ejemplo INICIO=$2600, STOP=$2609, y ERRORALTO=0.

```
2600  36
2601  37
2602  FF 0184
2605  CE 0180
2608  5F
2609  6F 01
```

Note que la subrutina imprime cada instrucción al inicio de una nueva linea. Cada linea contiene la dirección del primer byte de la instrucción seguida por un espacio en blanco seguido del byte del código operacional. Si es una instrucción de 2 o 3 palabras, la subrutina brinca otro espacio y despliega el valor del operando como se ilustra arriba.

Si la subrtuina DECODE encuentra un código operacional inválido, deberá sacar un mensaje especial de error en la linea con el código operacional erroneo. El código operacional inválido deberá ser seguido por un espacio, seis asteriscos y un campanazo. Considere el ejemplo siguiente:

```
2600  36
2601  38  ******[suena campana]
2602  FF 0184
2605  CE 0180
2608  52  ******[suena campana]
2609  6F 01
```

Note que código operacional inválido es indicado visiblemente y auditivamente para que el usuario lo tome en cuenta. Si el parametro ERROPALTO hubiera sido diferente de cero, la subrutina se hubiera detenido después de la segunda linea en el ejemplo anterior.

Cuando escribas la subrutina DECODE considèra el uso de una tabla de seguimiento. Asegúrate de examinar el mapa de códigos operacionales para determinar que reglas puedan minimizar el tamaño de tu tabla, y también checa esas excepciones a la regla.

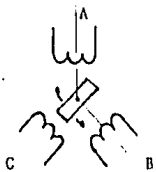
Puedes usar cualquier subrutina que este documentada en el manual de MUDBUG. RAM inicia en la localidad $0180.

PRACTICA No.4 :

APLICACION MECANICA

PRESENTACION DEL PROBLEMA

Un motor de paso es otra aplicacion de un PIA. Imagine 3 electromagnetos o devanados, A, B, y C colocados en angulos iguales alrededor de un magneto el cual es libre de girar.



Cada electromagneto A, B y C estan bajo control de un bit del Buffer de datos del PIA, como se muestra abajo.

Un magneto esta "ENCENDIDO" cuando el bit apropiado esta en el estado 1, y "APAGADO" cuando esta EN ESTADO 0. Energizar el magneto C causa que el polo norte del magneto central rote a el polo sur del magneto C.

Buffer de Datos PIA

Para colocar el polo norte del magneto central apuntando hacia A, y asumiendo que el PIABFB ha sido inicializado para salida, entonces con el siguiente par de instrucciones se ejecutara lo deseado:

```
LDAA  =$01
STAA  PIABFB
```

BIT 0 (Electromagneto A) esta "ENCENDIDO".

Se pueden presentar varias opciones con las que quisieramos trabajar:

1) ?Como colocar el polo norte del magneto central entre 2 puntos?, suponga A y B. Escriba las instrucciones.

SUGERENCIA: Que pasa si se activan los magnetos A y B.

2) ?Que programa realizarias para hacer que el magneto central gire continuamente?. Escriba las instrucciones.

SUGERENCIA: Use una subrutina de retardo entre cada cambio para hacer mas lento los cambios de la computadora, y el motor pueda girar a una velocidad aceptable.

3) ?Como modificaría la velocidad angular para este motor de pasos, bajo el control de un programa?.

SUGERENCIA: Variando la constante de la subrutina de retardo via el teclado.

SUGERENCIA: Para llevar a la práctica este programa, cheque los rangos de voltaje con los que trabaja el motor de paso y vea el rango de voltaje y corriente que el kit puede manejar. Es posible que se requiera un circuito intermedio (interfase) para acoplar el motor de pasos y el kit.

PRACTICA No.5 :
APLICACION ELECTRONICA (INSTRUMENTACION)

PRESENTACION DEL PROBLEMA

Una aplicación de la operación IRQ (interrupt request) esta en controlar las operaciones específicas de una computadora con respecto al tiempo.

Por ejmplo: Un voltímetro digital quiza sea requerido para hacer indicaciones de laboratorio o en procesos de control a un porcentaje de 10 mediciones por segundo. Además de la ineficacia de los loops de tiempo para control de estas mediciones, la computadora no esta disponible para otras tareas.

La solución es usar un componente, "Reloj a tiempo real", el cual produce interrupciones a un promedio de tiempo específico.

Escriba la rutina de inicialización y la rutina de servicio IRQ para este reloj, el cual causa que el voltimetro digital realice 10 mediciones por segundo.

SUGERENCIA: Puede colocar un interruptor a la linea IRQ del microprocesador para así activar manualmente la rutina de servicio IRQ y probar que la velocidad y el procentaje de mediciones por segundo dependerá del componente externo que maneja las interrupciones.

CONCLUSIONES

CONCLUSIONES

Con los adelantos de la tecnología es posible que vengan nuevos y mejores kits de microprocesadores en los cuales las ventajas sean inumerables comparándolos con el que hasta ahora ha sido uno de los microprocesadores mas manejables y fácil de aprender, el MC 6800.

No obstante las prácticas que aqui se describen podrán ser aplicadas a otros procesadores y quizás las respuestas que se han dado en esta tesis no tengan similitud alguna, pero creo que si podrán actualizarse y dar una idea de lo que son los microprocesadores y sus aplicaciones.

APENDICE A

CODIGO HEXADECIMAL, DECIMAL Y BINARIO

HEXADECIMAL	DECIMAL	BINARIO
00	00	0000
01	01	0001
02	02	0010
03	03	0011
04	04	0100
05	05	0101
06	06	0110
07	07	0111
08	08	1000
09	09	1001
0A	10	1010
0B	11	1011
0C	12	1100
0D	13	1101
0E	14	1110
0F	15	1111
10	16	10000

APENDICE B

CODIGO ASCII

Bits 4 a 6 / BITS 0 a 3	0	1	2	3	4	5	6	7
	0	NUL	DLE	SP	0	@	P		p
	1	SOH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	'	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L	/	l	/
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N		n	~
	F	SI	US	/	?	O	_	o	DEL

APENDICE C

GLOSARIO

ACUMULADOR.- Registro donde son mantenidos resultados aritméticos y lógicos .

ACIA.- Asynchronous Comunication Interface Adapter, Comunicación entre periféricos y MPU asincrona, en serie.

ALU.- Unidad Aritmética y Lógica. La parte del MPU donde funciones aritméticas y lógicas son desarrolladas.

ASCII.- American Standard Code for Information Interchange. Código binario para representar caracteres alfanuméricos, especiales y de control.

ASINCRONO.- Operaciones que son iniciadas inmediatamente después de la finalización de la anterior, no estan controladas por reloj alguno.

BAUD.- Medida de velocidad de transmisión. Número de veces que una linea de transmisión cambia de estado en un segundo. Igual a bits por segundo si cada estado de la linea es representado por un 1 o un 0.

BCD.- Binary Coded Decimal. Representación de números decimales donde cada dígito es representado por su equivalente binario.

BINARIO.- Sistema numérico en base 2.

BIT.- Un dígito binario.

BUFFER.- Circuito que provee separación entre alguna parte del sistema y el resto del mismo.

BUG.- Un error de programa que causa malfuncionamiento.

BUS.- La interconección entre un sistema que lleva datos en forma paralela. Varios componentes pueden usar el mismo "BUS" pero solamente puede ser usado en un tiempo como transmisor y en otro como receptor.

BYTE.- Un grupo de bits. Comunmente 8 bits.

COMPILADOR.- Programa el cual convierte comandos en lenguaje de alto nivel a comandos en lenguaje ensamblador o lenguaje máquina.

CONTADOR DE PROGRAMA.- PC Registro el cual mantiene la dirección de la siguiente instrucción (o dato) del programa que esta siendo ejecutado.

CPU.- Unidad Central de Procesamiento. Parte del sistema la cual desarrolla funciones de cálculo y manejo de datos.

CROM. Control de la memoria ROM.

CRT.- Tubo de rayos catódicos.

DEBUG.- Proceso de Checar y corregir cualquier error de programa.

DIRECCIONAMIENTO DIRECTO.- Un modo de direccionamiento donde la dirección del operando esta contenida en la instrucción.

DIRECCIONAMIENTO INDEXADO.- Una forma de direccionamiento indirecto el cual usa el registro índice para mantener la dirección del operando.

DIRECCIONAMIENTO INMEDIATO.- Modo de direccionamiento el cual usa parte de la misma instrucción como dato de operación.

DIRECCIONAMIENTO INDIRECTO.- Modo de direccionamiento donde la dirección de la localidad donde la dirección del operando puede ser encontrado es contenido en esta instrucción.

DIRECCIONAMIENTO RELATIVO.- Modo de direccionamiento donde el lugar donde encontrará la instrucción o dato dependerá de la localización del contador de programa y el valor de desplazamiento indicado por la instrucción que esta siendo ejecutada.

DMA.- Acceso de memoria directo.

DUPLEX.- Transferencia de datos en dos direcciones.

ENSAMBLADOR.- Programa el cual convierte comandos en lenguaje ensamblador a comandos en lenguaje máquina, y checa si es válida la instrucción o si esta completa la definición.

EPROM.- ROM Electricamente programable. Memoria puede ser borrada (luz ultravioleta) y programada por medios eléctricos.

FIRMWARE:- Instrucciones o datos almacenados permanentemente en ROM.

FLAG.- Un Flip-Flop que puede ser puesto en estado 1 o 0 mediante programación.

FLIP-FLOP.- Componente de dos estados que cambia cuando el reloj del componente cambie de estado, el cual es controlado por un reloj externo.

GLITCH.- Pulso de ruido.

HANDSHAKE.- Sistema de transferencia de datos entre CPU y periféricos, donde el CPU pide al periférico que le acepte datos y los enviará solo si el receptor le contesta "si".

HARDWARE.- Todos los componentes electrónicos y mecánicos que forman un sistema.

HEXADECIMAL.- Sistema numérico en base 16.

INICIALIZAR.- Puesta a punto de todos los registros, definición inicial.

INSTRUCCION.- Patrón de bits el cual indica al MPU la ejecucción de cierta función.

INTERFASE.- Circuito el cual conecta diferentes partes de un sistema desarrollando y procesando señales para hacer una transferencia perfecta entre dos o mas componentes.

INTERRUPCION.- Una señal del MPU la cual obliga a cambiar de una tarea a otra.

I/O.- Entrada/Salida.

K.- Abreviación de 1024.

LENGUAJE.- Significado sistemático de comunicación con el MPU.

LENGUAJE ENSAMBLADOR.- Representación de comandos en MNEMONICOS y manejo conveniente de memoria manejando ciertos símbolos.

LENGUAJE MAQUINA.- El nivel mas bajo de programación. El único lenguaje que el MPU puede entender sin necesidad de un traductor.

LENGUAJE DE ALTO NIVEL.- Lenguaje de computadora que es fácil de usar, el cual requiere compilarse a lenguaje máquina antes de que pueda ser usado por el MPU.

LIFO.- Last In First Out, Ultimo en entrar primero en salir. describe el uso del STACK.

MASK.- Patrón de bits que es usado en conjunto para seleccionar uno o varios bits de una palabra en especial.

MAPA DE MEMORIA.- Cuadro que muestra la localización de memoria de un sitema.

MICROCICLO.- Paso de Programa sencillo del MPU. El nivel mas pequeño de pasos en un programa.

MICROPROCESADOR.- Un CPU implementado con circuitos de gran escala de integración. Frecuentemente consiste de un solo módulo.

MNEMONICO.- Una palabra o frase que hace referencia a otra frase de mayor longitud, lo cual hace que sea recordada con facilidad.

SUBRUTINAS DISPONIBLES

ADDR ($FFDE) Substrae 2 números de 16-bits almacenados en 813A y 813C y pone el resultado en "A" y "B". Resultado = (813C,813D)-(813A,813B).

ADDXA ($F953) Añade el acumulador A al registro X.

CLRD ($F849) Borra dígitos como esten indicados en el acumulador A.

DISNMI ($F8A7) Deshabilita NMI's del teclado.

ENNMI ($F8B2) Habilita NMI's del teclado.

DLY ($F94F) Loop de retardo, dependiendo del contenido del registro X. XR = $6D = 1 milisegundo.

DLY1 ($F94C) Retarda un milisegundo.

DLY25 ($F947) Retarda 25 segundos.

DYSCOD ($FC11) Decodifica 4 bytes hexadecimales almacenados desde HEXBUF a HEXBUF+3 a un código de display de 7-segmentos los cuales son almacenados Desde DISBUF hasta DISBU+7 para ser desplegados.

EFPI ($F83C) permite que haya interrucpión de teclado cuando cualquier tecla sea presionada.

EXTST ($F842) Busca la presencia de $55AA en la localidad $F000 usada para ver si existe expansión de ROM.

GETIT ($FACA) Lee el teclado y determina si alguna tecla de función fue presionada. Regresa al monitor de D3JBUG si la tecla escape es presionada.

IRQ ($FFEA) Entrada/Salida a el ACIA ($800B).

LOAD ($FED9) Carga de cinta formateada a un BAUD de 1200.

S300LD ($FEFB) Carga de cinta formateada a un BAUD de 300.

S1200P ($FFB7) Manda a cinta formateada a un BAUD de 1200.

S300P ($FF7F) Manda a cinta formateada a un BAUD de 300.

ROLENT ($FB4A) Empaca bytes hexadecimales en 1 o 2 bytes.

MODEM.- Modulador/Demodulador usada para enviar y recivir datos en serie sobre una red de audio.

OBJETO, Código.- Un patrón de bits que son presentados al CPU como una instrucción y dato.

PALABRA.- Colección paralela de dígitos binarios (bits) parecidos a un byte.

PARIEDAD.- Bit de prueba que es añadido al patrón de bits en una transmisión.

PERIFERICO.- Equipo para entrada o salida de datos de un sistema, generalmente para comunicarse con un usuario.

PIA.- Interfase adaptadora de periféricos.

PUERTO.- Una terminal que el CPU usa para comunicarse con el exterior.

PROM.- ROM Programable, Es un módulo ROM que puede ser programado por el usuario.

RAM.- Memoria de acceso aleatorio. Datos pueden ser escritos y leidos desde cualquier y en cualquier lugar de memoria.

REGISTRO.- Localidad de memoria del MPU de propósito general que sirve para mantener una palabra del MPU.

ROM.- Memoria de lectura solamente. Módulo de Memoria que de acuerdo a los procedimientos de manufactura su contenido ha sido establecido y estos no pueden ser cambiados, solamente leidos.

SIMPLEX.- Transmisión de datos en un solo sentido.

SOFTWARE.- Programa almacenado en cualquier medio.

STACK.- Medio de almacenamiento que sigue el protocolo de último dato en entrar sera el primero en salir. Genralmente se le asigna un espacio en memoria RAM.

SUBRTUINA.- Una secuencia de instrucciones que desarrollan una función específica y que pueden ser llamadas desde cualquier lugar del programa, esto ahorra espacio de memoria.

TRES-ESTADOS.- Descripción de un estado lógcico logrado por ciertos componentes al elevar la impedancia de su salida, logrando así deshabilitar la salida.

TTY.- Teletipo.

UART.- Transmisor/Receptor Universal asíncrono.

APENDICE D

Referencia Técnica del ACIA.

FORMATO DEL REGISTRO DE CONTROL DEL ACIA.

BIT 0 y BIT 1.- Promedio de conteo y RESET Maestro. se usa en las dos secciones, transmisión y recepción.

b1	b0	Funcion
0	0	ciclos de reloj entre 1
0	1	ciclos de reloj entre 16
1	0	ciclos de reloj entre 64
1	1	RESET MAESTRO.

BIT 2, BIT 3 y BIT 4.- Longitud de palabra, Pariedad y Selección de bit de Parada.

b4	b3	b2	Longitud de palabra	Pariedad	Bits de parada
0	0	0	7	PAR	
0	0	1	7	NON	
0	1	0	7	PAR	
0	1	1	7	NON	
1	0	0	8	NINGUNA	2
1	0	1	8	NINGUNA	1
1	1	0	8	PAR	1
1	1	1	8	NON	1

BIT 5 y BIT 6.- Bits de control de transmisión: Controla la interrupción a la salida y RTS y provee de tranmisión de interrupción.

b6	b5	Funcion
0	0	RTS = 0 e inhibe interrupciones Tx
0	1	RTS = 0 y permite la interrupción a Tx
1	0	RTS = 1 e inhibe la interrupción Tx
1	1	RTS = 1 transmite una interrupción y permite interrupcción Tx.

BIT 7 .- Habilitar para recivir interrupciones.

B7 = 1 Habilita interrupción cuando esta reciviendo datos.

B7 = 0 No permite interrupciones cuando esta reciviendo.

FORMATO DEL REGISTRO DE ESTADOS DEL ACIA.

BIT 0.- REGISTRO DE RECEPCION DE DATOS LLENO.
b0 = 0 : Indica que esta vacio.
b0 = 1 : Indica que se ha recivido datos.

BIT 1.- REGISTRO DE TRANSMISION DE DATOS VACIO.
b1 = 0 : Indica que esta vacio.
b1 = 1 : Indica que tiene datos por enviar.

BIT 2.- INDICA PRESENCIA DE CARRY
b2 = 0 : Indica que el carry esta presente.
b2 = 1 : Indica que no hay carry presente.

BIT 3.- BORRADO LISTO PARA ENVIAR
Este bit refleja el estado de entrada que usa el MPU para hacer interfase con el MPU.

BIT 4.- ERROR DE PATRON DE DATOS FUERA DE CUADRO. (FRAMING ERROR).
Indica la ausencia del primer bit de parada, lo cual es el resultado de un error de sincronia, transmisión defectuosa o una condición de alto.

BIT 5.- ERROR DE SOBRERECCEPCION. (OVERRUN ERROR).
Indica que un caracter o un número de caracteres fueron recividos pero no leidos del registro de recepción de datos antes de que llegaran otros carateres.

BIT 6.- ERROR DE PARIEDAD.
Indica que un error de pariedad existe.

BIT 7.- REQUERIMIENTO DE INTERRUPCION.
Es el complemento de la salida de $\overline{IRQ}$. Cualquier interrupción que es puesta y habilitada estará disponible en este registro.

APENDICE E

Referencia Técnica del PIA.

DDR.- REGISTRO DE DIRECCION DE DATOS.
Es accesado a través del bit 2 del registro de control. el bit 2 mostrará "0".

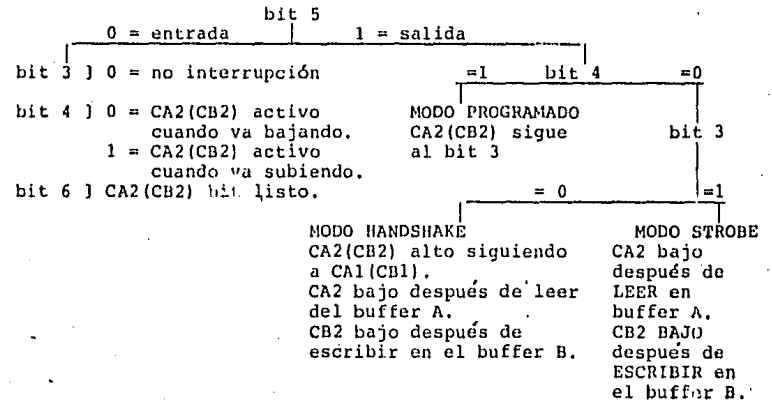
Para cada una de las 8 lineas de datos: 1 = salida
0 = entrada

FORMATO DEL REGISTRO DE CONTROL DEL PIA.

bit 0] cero para no interrupcidn.

bit 1] 0 = CA1(CB1) activo cuando va bajando el ciclo de reloj.
1 = CA1(CB1) activo cuando va subiendo el ciclo de reloj.

bit 2] 0 = DDR se accesa a través del buffer de datos.
1 = Acceso al Buffer de datos.



NOTA: bajo significa cero lógico.
alto significa uno lógico.

FORMATO DEL REGISTRO DE ESTADOS DEL PIA.

BIT 0: CA1(CB1) Habilitador /Deshabilitador de Interrupción.

BIT 1: Determina transmisión activa CA1(CB1) para establecer la bandera de interrupción en el bit 7.

BIT 2: Determina si esta direccionado el registro de dirección de datos o el registro de salida.

BIT 3, BIT 4 y BIT 5: CONTROL DE CA2(CB2)
Si bit 5 = 0 entonces bit 3 y 4 se comportan como bit 0 y 1 pero para CA2(CB2).
Si bit 5 = 1 y bit 4 = 0, entonces sucede lo siguiente:

para CA2 bit 3 = 0...Lee con CA1 almacenado
bit 3 = 1...Lee con E almacenado

para CB2 bit 3 = 0...Escribe con CB1 almacenado
bit 3 = 1...Escribe con E almacenado

Si bit 5 = 1 y bit 4 = 1, entonces bit 3 establece o no a CA2(CB2), lo pone en uno o cero.

APENDICE F

SISTEMA OPERATIVO D3JBUG

DESCRIPCION GENERAL: Es un programa monitor de 2K x 8 bits, el cual reside en el MC6846 del kit MEK6802D3, el cual es una combinación de ROM, timer e interfase. D3JBUG requiere de 128 bytes para operación de su stack. El cual esta localizado de la dirección $8100 a la $817F. Este stack RAM es usado como un espacio para almacenamiento temporal para el procesador 6800 cuando una operación de stack es requerido como PUSH o PULL, o cuando una subrutina o interrucpión es encontrada.

Algunas localidades de RAM son usadas para almacenamiento temporal de datos y para algunas banderas que mantienen comunicación entre varias rutinas. Las localidades para esas rutinas de utilería son las siguientes:

NMPTR ($8100) Contiene la direcccidn de programa que es ejecutada durante la rutina PUT.

UHASH ($8102) Apuntador de usuario para la función especial de buscar-en-tablas.

UNMIV ($8104) Vector de usuario no-enmascarable, almacena la dirección del programa de la interrupción NMI de usuario.

UIRQV ($8106) Vector de requerimiento de interrupción del usuario.

USWIV ($8108) Vector de interrupción de software de usuario.

DIGEN ($8115) Bit habilitador de rotación de dígito del display.

KEY ($8117) Decodifica el valor de la tecla a la interrupcción del teclado.

HEXBUF ($8118) Buffer hexadecimal de 4 bytes para el LED del display.

DISBUF ($811C) Buffer display de 8-bytes (codigo de 7 segmentos).

USP ($812B) Apuntador de stack del usuario.

ROWCOL ($812E) Contiene el código de la tecla que fue presionada derivada de la rutina GET, la cual interroga por una interrupcción de teclado.

ROLFLG ($8134) Bandera usada por la subrutina ROLENT para indicar a la rutina que ceros son requeridos a la izquierda cuando el cero es almacenado. Si el uno es puesto, la rutina mueve el display y coloca el último valor en el lado derecho del acumulador A.

FLG24 ($8135) Bandera usada por la subrutina ROLENT para indicar que displays deben de actualizarse.

TOFT ($8140) Un apuntador de 2-bytes es usado por la subrutina de punto-de-freno que mantiene la dirección del apuntador de la tabla de punto-de-freno .

SUMARIO DE COMANDOS

TECLA	DIRECCION	DESCRIPCION
M	FAE2	PERMITE QUE LAS LOCALIDADES DE MEMORIA SEAN VISUALIZADAS Y CAMBIADAS.
EX	F85A	ESCAPE - PERMITE REENTRAR AL D3JBUG. NO BORRA "BREAKPOINTS".
RD	F9E3	PERMITE AL USUARIO CAMBIAR Y VISUALIZAR LOS REGISTROS
GO	FC6A	DA EL CONTROL A LA DIRECCION QUE SE ENCUENTRA EN EL CONTADOR DE PROGRAMA.
FS	FC52	PONE LA BANDERA DE FUNCION ESPECIAL.
FC	FC5F	BORRA LA BANDERA DE FUNCION ESPECIAL.
P/L	FE22	PUNCH/LOAD/VERIFY EL BAUD DE LA CINTA EN 1200/300.
T/B	FCD6	PERMITE VISUALIZAR, AÑADIR Y BORRAR HASTA 8 "BREAKPOINTS".

APENDICE G

SISTEMA OPERATIVO MUDBUG

DESCRIPCION GENERAL: Es un programa de utilería de propósito general para el microprocesador MC6800. Fue diseñado y desarrollado en la Universidad Estatal de Arizona, se usa en un chip EPROM de 2K (2,048 palabras) de espacio de ROM. MUDBUG requiere además del bloque de ROM un bloque de RAM de 128-bytes para stack y variables internas. Además MUDBUG requiere un ACIA para comunicación RS-232 con terminales para el usuario. La localidad para el registro de control del ACIA esta definida por ACONT y el registro de datos del ACIA como ADATA.

MUDBUG esta diseñado de tal manera que las últimas 8 localidades del espacio en ROM funcionen como vectores de interrupción para el sistema, y los vectores de interrupción son como siguen:

INTERRUPCION DE HARDWARE.- (IRQ) Brinca a RAM y RAM+1.

INTERRUPCION SOFTWARE.- (SWI) Brinca a RAM+2 Y RAM+3.

INTERRUPCION NO ENMASCARABLE.- (NMI) Brinca a RAM+4 y RAM+5.

INTERRUPCION DE REINICIO.- (RSI) Inicializa el sistema MUDBUG como si lo hubieran encendido.

CONTINUACION DE APENDICE G

SUMARIO DE COMANDOS
--MNEMONICO DEL COMANDO

|--Numero de Parametros

DESCRIPCION:

* 0 Un asterico introduce una linea de comentario.
A 0 Imprime el valor de AR en hex, y accepta un nuevo valor para AR.
B 0 Imprime el valor de BR en hex, y accepta un nuevo valor para BR.
C 1 Cambia (después de imprimir) el contenido de la localidad START.
E 0 Establece un nuevo valor para CC después de imprimirlo.
F 4 Encuentra KEY bajo los bits de MASK desde la localidad START hasta STOP.
G 1 Va a la localidad START para iniciar el programa de usuario.
H 0 Se detiene y regresa a la localidad inidicada por CONT DE PROG.
I 3 Inicializa localidades desde START hasta STOP con el valor de KEY.
K 2 Calculador. Pone START<-- START + STOP, e imprime START.
L 1 Carga(.) o Compara(,) un modulo objeto a memoria, Despliega START.
M 2 Listado de Memoria desde START hasta STOP.
N 1 Paso N. Ejecuta instrucciones por pasos, inicia en START.
O 1. Un paso. Ejecuta una instrucción a la vez, inicia en START.
PE 1 PEEK en la memoria, Lee de la memoria.
PO 1 POKE en la memoria, escribe pero no lee de la memoria.
PR 1 Imprime el valor del Contador de Programa y acepta nuevo valor.
Q 0 Despliega los registros AR,BR,XR,PR,SP,HINZVC.
R 1 Dirección relativa: Imprime o estable la dirección destino de una instrucción branch.
S 0 Imprime el valor del stack pointer y acepta nuevo valor.
T 4 Va a START y da el control cuando ha encontrado la llave KEY las veces indicadas hasta la localidad STOP si no encuentra.
V 4 Verifica el programa ROM, igual que T pero en ROM.
W 2 Escribe desde localidad START hasta STOP en la cinta en forma objeto.
X 0 Imprime el valor del registro Indice y acepta nuevo valor.
Z 0 Pone cero a los registros AR, BR, XR, PR y CC, inicializa SP.

SUBRUTINAS DISPONIBLES

CRLF	ROM+$00	CRLF a la terminal (retorno de carro y alimentación)
CRLF4H	ROM+$03	Sale CRLF a la terminal y el valor de XR.
ERROR	ROM+$06	Salida un DIAGONAL y campana, y regresa a MUDBUG.
INCHR	ROM+$09	Admite caracter sencillo del teclado a AR.
MUDBUG	ROM+$0C	Regresa control directamente al inicio de MUDBUG.
NEWVAL	ROM+$0F	Lee un número de 4-bytes de XR y XR+1.
NUMBI	ROM+$12	Lee un número de 4-bytes de AR(MSB) y BR (LSB).
OUT2H	ROM+$15	Salida de 2-bytes en hexadecimal de XR.
OUT2HB	ROM+$18	Salida de 2-bytes en hex. de BR.
OUT2HS	ROM+$1B	Salida de 2-bytes en hex. de XR y un espacio.
OUT4HS	ROM+$1E	Salida de 4-bytes en hex. de XR y XR+1 y un espacio.
OUTCHR	ROM+$21	Salida de un caracter ASCII del registro AR.
OUTS	ROM+$24	Salida de un espacio en blanco a la terminal.
POWERUP	ROM+$30	Reinicio como si fuera arranque en frio.
PRTXM	ROM+$27	Imprime el valor de 4-bytes de XR y el del contenido de la localidad indicada por XR.
READPT	ROM+$2A	Lee un caracter de cinta y lo pone en AR.
STAAB	ROM+$2D	STAA 0,X; STAB 1,X; RTS

ACCUMULATOR AND MEMORY OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	↕	3				A + M → A	↕	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	↕	3				B + M → B	↕	↕	↕	↕	↕	↕
Add Accums	ABA													1B	2	1	A + B → A	↕	↕	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	↕	3				A + M + C → A	↕	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	↕	3				B + M + C → B	↕	↕	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	↕	3				A · M → A	↔	↕	↕	↕	R	↔
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	↕	3				B · M → B	↔	↕	↕	↕	R	↔
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	↕	3				A · M	↔	↕	↕	↕	R	↔
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	↕	3				B · M	↔	↕	↕	↕	R	↔
Clear	CLR							6F	7	2	7F	6	3				00 → M	↔	↕	R	S	R	R
	CLRA													4F	2	1	00 → A	↔	↕	R	S	R	R
	CLRB													5F	2	1	00 → B	↔	↕	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	↕	3				A − M	↔	↕	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	↕	3				B − M	↔	↕	↕	↕	↕	↕
Compare Accums	CBA													11	2	1	A − B	↔	↕	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M → M	↔	↕	↕	↕	R	S
	COMA													43	2	1	A → A	↔	↕	↕	↕	R	S
	COMB													53	2	1	B → B	↔	↕	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	↔	↕	↕	↕	ⓥ	ⓒ
	NEGA													40	2	1	00 − A → A	↔	↕	↕	↕	ⓥ	ⓒ
	NEGB													50	2	1	00 − B → B	↔	↕	↕	↕	ⓥ	ⓒ
Decimal Adjust A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	↔	↕	↕	↕	ⓘ	ⓒ
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	↔	↕	↕	↕	ⓥ	↔
	DECA													4A	2	1	A − 1 → A	↔	↕	↕	↕	ⓥ	↔
	DECB													5A	2	1	B − 1 → B	↔	↕	↕	↕	ⓥ	↔
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	↕	3				A ⊕ M → A	↔	↕	↕	↕	R	↔
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	↕	3				B ⊕ M → B	↔	↕	↕	↕	R	↔
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	↔	↕	↕	↕	ⓥ	↔
	INCA													4C	2	1	A + 1 → A	↔	↕	↕	↕	ⓥ	↔
	INCB													5C	2	1	B + 1 → B	↔	↕	↕	↕	ⓥ	↔
Load Accum	LDAA	86	2	2	96	3	2	A6	5	2	B6	↕	3				M → A	↔	↕	↕	↕	R	↔
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	↕	3				M → B	↔	↕	↕	↕	R	↔
Or Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	↕	3				A ∨ M → A	↔	↕	↕	↕	R	↔
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	↕	3				B ∨ M → B	↔	↕	↕	↕	R	↔
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	↔	↔	↔	↔	↔	↔
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	↔	↔	↔	↔	↔	↔
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	↔	↔	↔	↔	↔	↔
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	↔	↔	↔	↔	↔	↔
Rotate Left	ROL							69	7	2	79	6	3				M	↔	↕	↕	↕	ⓘ	↕
	ROLA													49	2	1	A	↔	↕	↕	↕	ⓘ	↕
	ROLB													59	2	1	B	↔	↕	↕	↕	ⓘ	↕
Rotate Right	ROR							66	7	2	76	6	3				M	↔	↕	↕	↕	ⓘ	↕
	RORA													46	2	1	A	↔	↕	↕	↕	ⓘ	↕
	RORB													56	2	1	B	↔	↕	↕	↕	ⓘ	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	↔	↕	↕	↕	ⓘ	↕
	ASLA													48	2	1	A	↔	↕	↕	↕	ⓘ	↕
	ASLB													58	2	1	B	↔	↕	↕	↕	ⓘ	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	↔	↕	↕	↕	ⓘ	↕
	ASRA													47	2	1	A	↔	↕	↕	↕	ⓘ	↕
	ASRB													57	2	1	B	↔	↕	↕	↕	ⓘ	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	↔	↕	R	↕	ⓘ	↕
	LSRA													44	2	1	A	↔	↕	R	↕	ⓘ	↕
	LSRB													54	2	1	B	↔	↕	R	↕	ⓘ	↕
Store Accum	STAA				97	4	2	A7	6	2	B7	5	3				A → M	↔	↕	↕	↕	R	↔
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	↔	↕	↕	↕	R	↔
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	↕	3				A − M → A	↔	↕	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	↕	3				B − M → B	↔	↕	↕	↕	↕	↕
Subtract Accums	SBA													10	2	1	A − B → A	↔	↕	↕	↕	↕	↕
Subtr with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	↕	3				A − M − C → A	↔	↕	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	↕	3				B − M − C → B	↔	↕	↕	↕	↕	↕
Transfer Accums	TAB													16	2	1	A → B	↔	↕	↕	↕	R	↔
	TBA													17	2	1	B → A	↔	↕	↕	↕	R	↔
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	↔	↕	↕	↕	R	R
	TSTA													4D	2	1	A − 00	↔	↕	↕	↕	R	R
	TSTB													5D	2	1	B − 00	↔	↕	↕	↕	R	R

INDEX REGISTER AND STACK POINTER OPERATIONS	MNEMONIC	IMMED			DIRECT			INDEX			EXTND			INHER			BOOLEAN/ARITHMETIC OPERATION	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	3	3				(IX_H:IX_L) − (M:M+1)	⇕	⇕	ⓒ	⇕	ⓒ	⇕
Decrement Index Reg	DEX													09	4	1	X − 1 → X	•	•	⇕	⇕	•	•
Decrement Stack Pnt	DES													34	4	1	SP − 1 → SP	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	X + 1 → X	•	•	•	⇕	•	•
Increment Stack Pnt	INS													31	4	1	SP + 1 → SP	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M → X_H, (M + 1) → X_L	•	⇕	ⓡ	⇕	R	•
Load Stack Pnt	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M → SP_H, (M + 1) → SP_L	•	•	ⓡ	⇕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				X_H → M, X_L → (M + 1)	•	•	ⓡ	⇕	R	•
Store Stack Pnt	STS				9F	5	2	AF	7	2	BF	6	3				SP_H → M, SP_L → (M + 1)	•	•	ⓡ	⇕	R	•
Indx Reg → Stack Pnt	TXS													35	4	1	X − 1 → SP	•	•	•	•	•	•
Stack Pnt → Indx Reg	TSX													30	4	1	SP + 1 → X	•	•	•	•	•	•

JUMP AND BRANCH OPERATIONS	MNEMONIC	RELATIVE			INDEX			EXTND			INHER			BRANCH TEST	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z ∨ (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C ∨ Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z ∨ (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C ∨ Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		———		⑯		———	
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	⑮	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	ⓒ	•	•	•	•

CONDITION CODE REGISTER OPERATIONS	MNEMONIC	INHER			BOOLEAN OPERATION	5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Accum A → CCR	TAP	06	2	1	A → CCR	———		(13)		———	
CCR → Accum A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES:
(Bit set if test is true and cleared otherwise)

(1) (Bit V) Test: Result = 10000000?
(2) (Bit C) Test: Result = 00000000?
(3) (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set)
(4) (Bit V) Test: Operand = 10000000 prior to execution?
(5) (Bit V) Test: Operand = 01111111 prior to execution?
(6) (Bit V) Test: Set equal to result of N ⊕ C after shift has occurred
(7) (Bit N) Test: Sign bit of most significant (MS) byte of result = 1?
(8) (Bit V) Test: 2's complement overflow from subtraction of LS byte?
(9) (Bit N) Test: Result less than zero? (Bit 15 = 1)
(10) (ALL) Load Condition Code Register from Stack. (See Special Operations)
(11) (Bit I) Set when interrupt occurs. If previously set, a Non Maskable Interrupt is required to exit the wait state.
(12) (ALL) Set according to the contents of Accumulator A.

LEGEND

OP	Operation Code (Hexadecimal);	00	Byte = Zero;
~	Number of MPU Cycles;	H	Half-carry from bit 3;
#	Number of Program Bytes;	I	Interrupt mask
+	Arithmetic Plus;	N	Negative (sign bit)
−	Arithmetic Minus;	Z	Zero (byte)
:	Boolean AND;	V	Overflow, 2's complement
M_SP	Contents of memory location pointed to be Stack Pointer;	C	Carry from bit 7
⊕	Boolean Inclusive OR;	R	Reset Always
⊕	Boolean Exclusive OR;	S	Set Always
M̄	Complement of M;	⇕	Test and set if true, cleared otherwise
→	Transfer Into;	•	Not Affected
0	Bit = Zero;	CCR	Condition Code Register
		LS	Least Significant
		MS	Most Significant

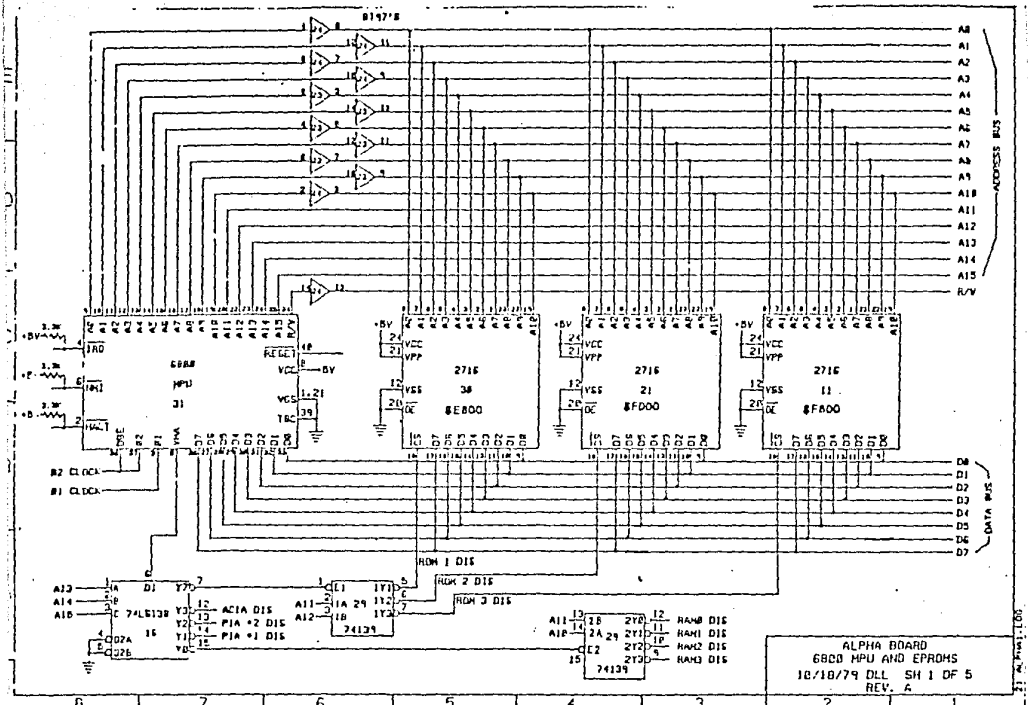
00	*		40	NEG	A		80	SUB	A	IMM	C0	SUB	B	IMM
01	NOP		41	*			81	CMP	A	IMM	C1	CMP	B	IMM
02	*		42	*			82	SBC	A	IMM	C2	SBC	B	IMM
03	*		43	COM	A		883	*			C3	*		
04	*		44	LSR	A		84	AND	A	IMM	C4	AND	B	IMM
05	*		45	*			85	BIT	A	IMM	C5	BIT	B	IMM
06	TAP		46	ROR	A		86	LDA	A	IMM	C6	LDA	B	IMM
07	TPA		47	ASR	A		88	*			C7	*		
08	INX		48	ASL	A		88	EOR	A	IMM	C8	EOR	B	IMM
09	DEX		49	ROL	A		89	ADC	A	IMM	C9	ADC	B	IMM
0A	CLV		4A	DEC	A		8A	ORA	A	IMM	CA	ORA	B	IMM
0B	SEV		4B	*			8B	ADD	A	IMM	CB	ADD	B	IMM
0C	CLC		4C	INC	A		8C	CPX	A	IMM	CC	*		
0D	SEC		4D	TST	A		8D	BSR		REL	CD	*		
0E	CLI		4E	*			8E	LDS		IMM	CE	LDX		IMM
0F	SEI		4F	CLR	A		8F	*			CF	*		
10	SBA		50	NEG	B		90	SUB	A	DIR	D0	SUB	B	DIR
11	CBA		52	*			91	CMP	A	DIR	D1	CMP	B	DIR
12	*		52	*			92	SBC	A	DIR	D2	SBC	B	DIR
13	*		53	COM	B		93	*			D3	*		
14	*		54	LSR	B		94	AND	A	DIR	D4	AND	B	DIR
15	*		55	*			95	BIT	A	DIR	D5	BIT	B	DIR
16	TAB		56	ROR	B		96	LDA	A	DIR	D6	LDA	B	DIR
17	TBA		57	ASR	B		97	STA	A	DIR	D7	STA	B	DIR
18	*		58	ASL	B		98	EOR	A	DIR	D8	EOR	B	DIR
19	DAA		59	ROL	B		99	ADC	A	DIR	D9	ADC	B	DIR
1A	*		5A	DEC	B		9A	ORA	A	DIR	DA	ORA	B	DIR
1B	ABA		5B	*			9B	ADD	A	DIR	DB	ADD	B	DIR
1C	*		5C	INC	B		9C	CPX		DIR	DC	*		
1D	*		5D	TST	B		9D	*			DD	*		
1E	*		5E	*			9E	LDS		DIR	DE	LDX		DIR
1F	*		5F	CLR	B		9F	STS		DIR	DF	STX		DIR
20	BRA	REL	60	NEG	IND		A0	SUB	A	IND	E0	SUB	B	IND
21	*		61	*			A1	CMP	A	IND	E1	CMP	B	IND
22	BHI	REL	62	*			A2	SBC	A	IND	E2	SBC	B	IND
23	BLS	REL	63	COM	IND		A3	*			E3	*		
24	BCC	REL	64	LSR	IND		A4	AND	A	IND	E4	AND	B	IND
25	BCS	REL	65	*			A5	BIT	A	IND	E5	BIT	B	IND
26	BNE	REL	66	ROR	IND		A6	LDA	A	IND	E6	LDA	B	IND
27	BEQ	REL	67	ASR	IND		A7	STA	A	IND	E7	STA	B	IND
28	BVC	REL	68	ASL	IND		A8	EOR	A	IND	E8	EOR	B	IND
29	BVS	REL	69	ROL	IND		A9	ADC	A	IND	E9	ADC	B	IND
2A	BPL	REL	6A	DEC	IND		AA	ORA	A	IND	EA	ORA	B	IND
2B	BMI	REL	6B	*			AB	ADD	A	IND	EB	ADD	B	IND
2C	BGE	REL	6C	INC	IND		AC	CPX		IND	EC	*		
2D	BLT	REL	6D	TST	IND		AD	JSR		IND	ED	*		
2E	BGT	REL	6E	JMP	IND		AE	LDS		IND	EE	LDX		IND
2F	BLE	REL	6F	CLR	IND		AF	STS		IND	EF	STX		IND
30	TSX		70	NEG	EXT		B0	SUB	A	EXT	F0	SUB	B	EXT
31	INS		71	*			B1	CMP	A	EXT	F1	CMP	B	EXT
32	PUL	A	72	*			B2	SBC	A	EXT	F2	SBC	B	EXT
33	PUL	B	73	COM	EXT		B3	*			F3	*		
34	DES		74	LSR	EXT		B4	AND	A	EXT	F4	AND	B	EXT
35	TXS		75	*			B5	BIT	A	EXT	F5	BIT	B	EXT
36	PSH	A	76	ROR	EXT		B6	LDA	A	EXT	F6	LDA	B	EXT
37	PSH	B	77	ASR	EXT		B7	STA	A	EXT	F7	STA	B	EXT
38	*		78	ASL	EXT		B8	EOR	A	EXT	F8	ADC	B	EXT
39	RTS		79	ROL	EXT		B9	ADC	A	EXT	F9	ADC	B	EXT
3A	*		7A	DEC	EXT		BA	ORA	A	EXT	FA	ORA	B	EXT
3B	RTI		7B	*			BB	ADD	A	EXT	FB	ADD	B	EXT
3C	*		7C	INC	EXT		BC	CPX		EXT	FC	*		
3D	*		7D	TST	EXT		BD	JSR		EXT	FD	*		
3E	WAI		7E	JMP	EXT		BE	LDS		EXT	FE	LDX		EXT
3F	SWI		7F	CLR	EXT		BF	STS		EXT	FF	STX		EXT

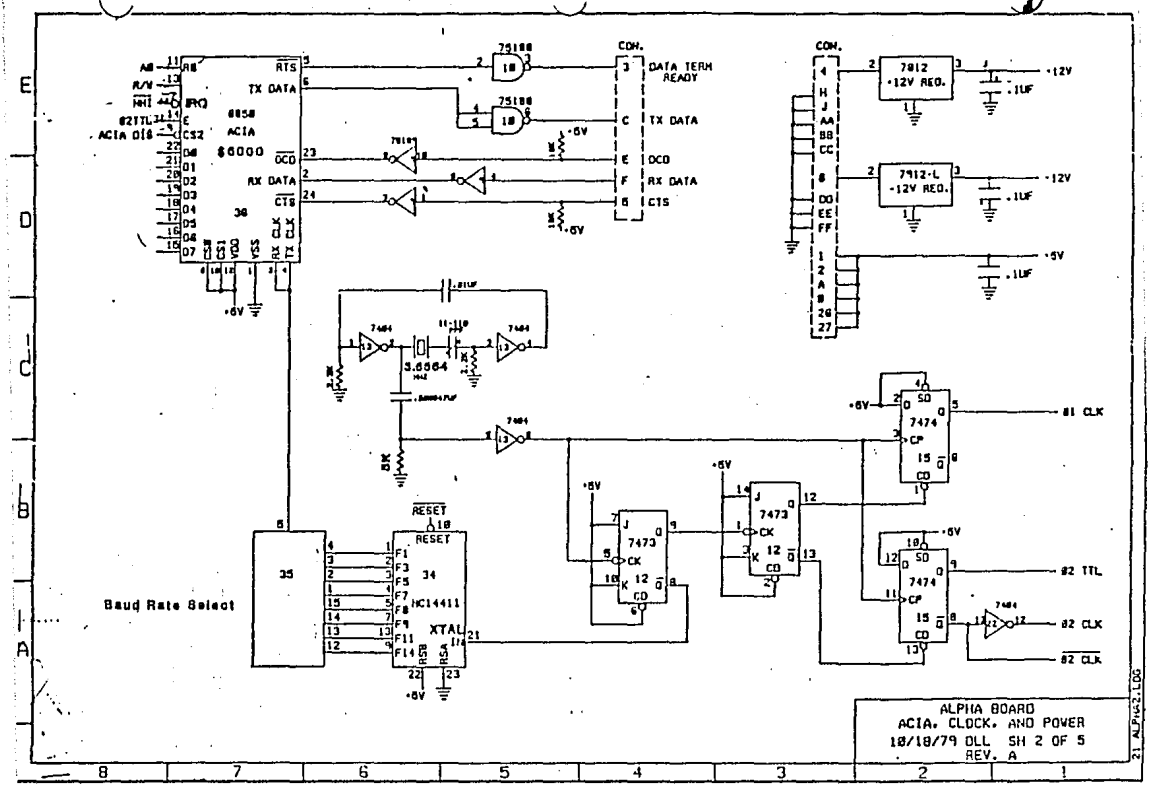
Notes: 1. Addressing Modes: A = Accumulator A, B = Accumulator B, IMM = Immediate, DIR = Direct, EXT = Extended, REL = Relative, IND = Indexed

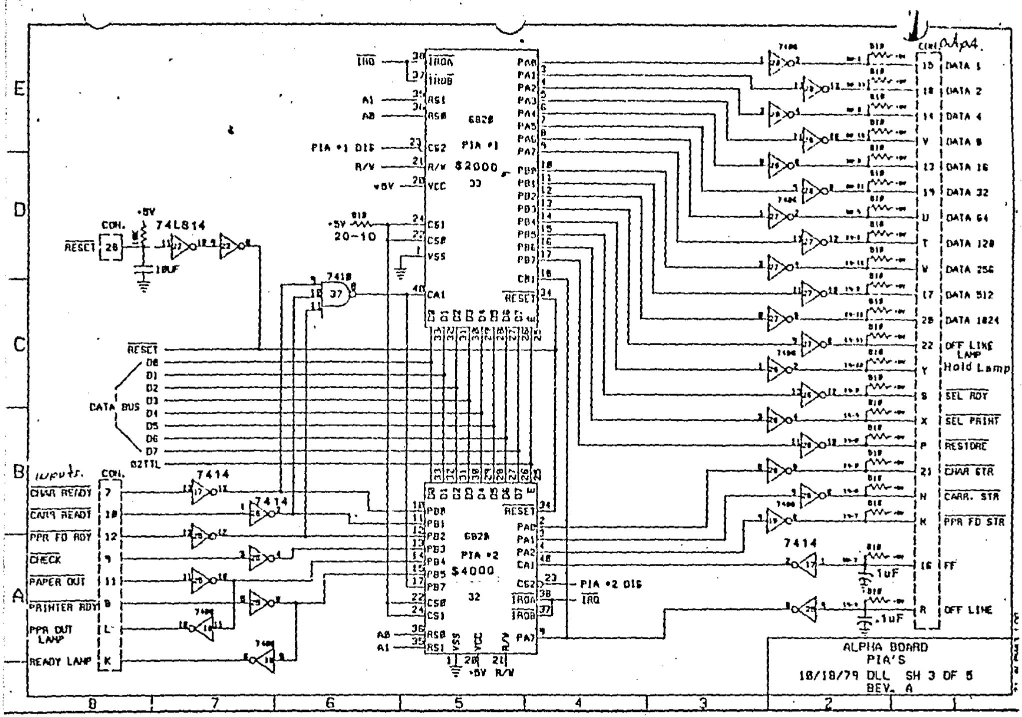
2. Unassigned code indicated by"*"

Hexadecimal Values of Machine Codes

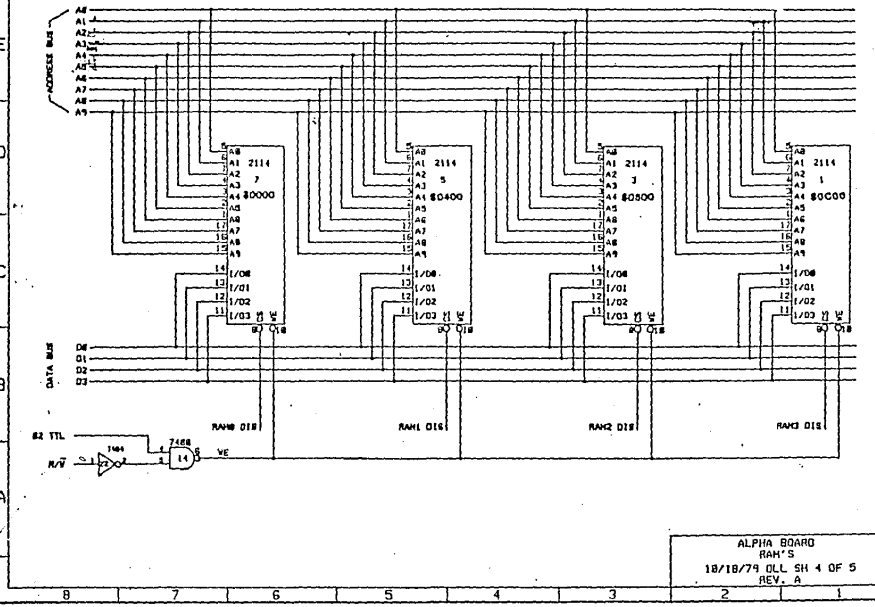
APPENDICE J


8T97'S
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
R/W
ADDRESS BUS
6800
MPU
2716
$E800
$F000
$F800
D0
D1
D2
D3
D4
D5
D6
D7
DATA BUS
ROM 1 DIS
ROM 2 DIS
ROM 3 DIS
ACIA DIS
PIA #2 DIS
PIA #1 DIS
RAM0 DIS
RAM1 DIS
RAM2 DIS
RAM3 DIS
74LS138
74139
ALPHA BOARD
6800 MPU AND EPROMS
10/10/79 DLL SH 1 OF 5
REV. A

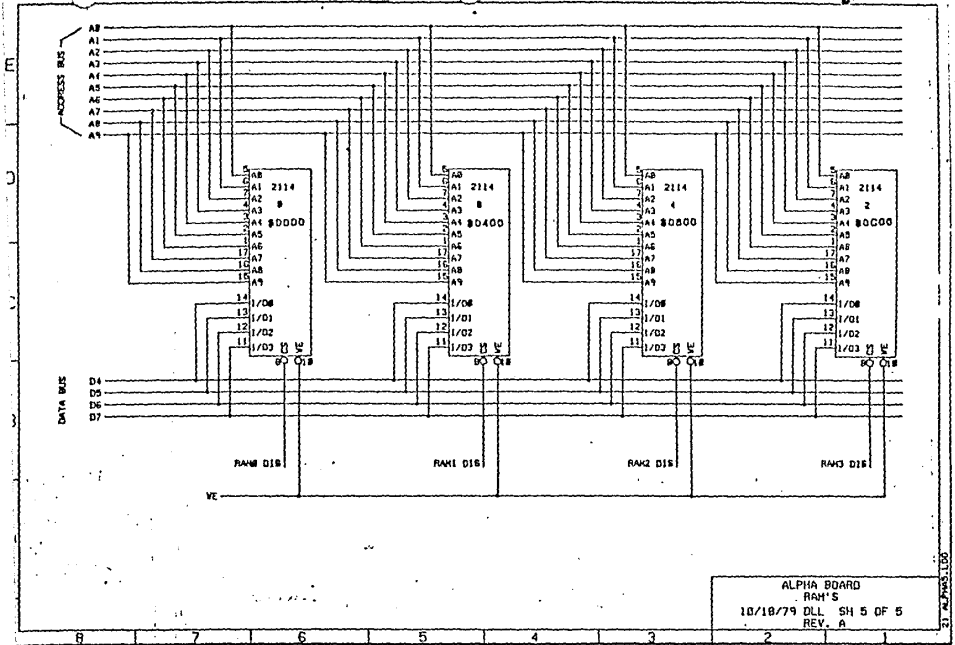

CON.
DATA TERM READY
TX DATA
DCD
RX DATA
CTS
6850 ACIA
$6000
7912 -12V REG.
7912-L -12V REG.
+12V
-12V
+5V
.1UF
3.6864
7474
7473
HC14411
XTAL
RESET
Baud Rate Select
Ø1 CLK
Ø2 TTL
Ø2 CLK
Ø2 CLK
ALPHA BOARD
ACIA, CLOCK, AND POWER
10/10/79 DLL SH 2 OF 5
REV. A
21 ALPHA2.LOG



ALPHA BOARD
PIA'S
10/18/79 DLL SH 3 OF 5
REV. A



ALPHA BOARD
RAM'S
10/10/79 OLL SH 4 OF 5
REV. A


ADDRESS BUS
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
DATA BUS
D4
D5
D6
D7
2114
$0000
$0400
$0800
$0C00
I/O0
I/O1
I/O2
I/O3
CS
WE
RAM0 DIS
RAM1 DIS
RAM2 DIS
RAM3 DIS
VE
ALPHA BOARD
RAM'S
10/10/79 DLL SH 5 OF 5
REV. A
21 ALPHA5.1000

BIBLIOGRAFIA

[1] BASIC MICROPROCESSORS AND THE 6800
RON BISHOP.
ED. HAYDEN
USA 1979

[2] WHAT EVERY ENGINEER SHOUL KNOW ABOUT MICROCOMPUTERS
WILLIAM S. BENNET AND CARL F. EVERT, Jr.
ED. MARCEL DEKKER.
USA 1980

[3] WHAT EVERY ENGINEER SHOULD KNOW ABOUT MICROCOMPUTER SYSTEMS
DESIGN AND DEBUGING
BILL WRAY AND BILL CRAWFORD
ED. MARCEL DEKKER
USA 1984

[4] PROGRAMMING THE 6800 MICROPROCESSOR
BOB SOUTHERN
MOTOROLA SEMICONDUCTOR PRODUCTS INC.
CANADA 1977

[5] MICROPROCESSORS DATA MANUAL
CENTRO DE INFORMACION TECNICA
MOTOROLA INC.
USA 1981

[6] SCHOTTKY TTL DATA BOOK
CENTRO DE INFORMACION TECNICA
MOTOROLA INC.
USA 1981

[7] MEMORY DATA MANUAL
CENTRO DE INFORMACION TECNICA
MOTOROLA INC.
USA 1982

[8] MUDBUG 2D USER'S MANUAL
DAVID C. PHEANIS
CUARTA EDICION
USA 1983

[9] D3JBUG USER'S MANUAL
CENTRO DE INFORMACION TECNICA
MOTOROLA INC.
USA 1979