UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERIA

HERRAMIENTAS PARA EL DESARROLLO DE SISTEMAS DIGITALES BASADO EN MICROPROCESADORES SIMULACION

TESIS

QUE PARA OBTENER EL TÍTULO DE:

MAESTRO EN INGENIERÍA ELÉCTRICA (ELECTRÓNICA)

PRESENTA:

JOSÉ FERNANDO GARCÍA NÚÑEZ CANO

MÉXICO, D. F. 1981

"HERRAMIENTAS PARA EL DESARROLLO DE SISTEMAS DIGITALES BASADOS EN MICROPROCESADORES SIMULACION"

JOSE FERNANDO GARCIA NUÑEZ CANO.

DIVISION DE ESTUDIOS DE POSGRADO DE LA FACULTAD DE INGENIERIA.

U N A M - 1 9 8 1

C O N T E N I D O

INTRODUCCION.

Durante los últimos 25 años las computadoras han venido a revolucionar todas las areas de la actividad humana. Esta importante aportación hecha por el hombre ha dado como resultado un gran poder para resolver una cantidad enorme de problemas que de otra manera serían casi imposibles de realizar.

Desde que aparecieron los microprocesadores hace aproximadamente 8 años, este progreso se ha hecho aún más notorio, ya que además de contar con todas las ventajas que las computadoras de por sí ofrecen, se requiere de un mínimo espacio para desarrollarlas, así como un incremento en su velocidad, confiabilidad, etc.

Cuando los primeros microprocesadores fueron introducidos a la industria electrónica fueron ignorados por un tiempo, hasta que ésta se percató de las grandes ventajas y amplias aplicaciones que éstos tenían. Desde entonces podemos encontrar microprocesadores en todas partes, como por ejemplo en las comunicaciones, en la industria controlando procesos, en cajas registradoras, en juegos de video electrónicos,etc.

El precio de los microprocesadores se reduce continuamente, no siendo éste el caso de los sistemas de desarrollo de sistemas digitales basados en microprocesadores, cuyo costo aún sigue resultando alto.

Con lo anterior y tomando en cuenta que ya se dispone de un sistema de cómputo, la adquisición de un sistema de desarrollo como el mencionado, no es rentable, por lo que se ve como una mejor alternativa, el desarrollo de programas que faci-

liten el diseño de sistemas basados en microprocesadores y que puedan correr en el sistema de cómputo ya disponible.

De todo lo anterior, se planteó como objetivo para este trabajo el realizar un programa de computadora para la minicomputadora PDP-11/40 que simulara el funcionamiento de un microprocesador Intel 8085. Este programa tendrá como entrada un archivo en disco, con código objeto para el microprocesador 8085 y deberá ofrecer a su salida el estatus del microprocesador después que se haya ejecutado cada instrucción del código objeto de entrada.

El trabajo se divide en dos partes fundamentalmente: una descripción breve acerca del microprocesador 8085 y el programa simulador que finalmente fue logrado. En esta segunda parte, se irán describiendo con detalle las características del programa simulador, para finalmente mostrarlo ya desarrollado.

EL MICROPROCESADOR INTEL 8085.

Generalidades.

En el año de 1971, Intel introdujo en el mercado de la industria electrónica el primer microprocesador de propósito general de 8 bits, el 8008. Era de tecnología MOS canal P y estaba empaquetado en un chip simple de 18 pins. Este microprocesador usaba memorias semiconductoras stándar ROM y RAM y en su mayor parte, componentes TTL para entrada y salida. Este microprocesador pronto encontró aplicaciones como en terminales y periféricos de computadoras, debido a sus características como un tiempo de ejecución de instrucción de 20 microsegundos, amplio conjunto de instrucciones y una organización de propósito general.

Con el advenimiento de las memorias RAM canal N y el empaquetado de 40 pins, Intel diseñó el microprocesador 8080A en el año de 1973. Este nuevo microprocesador fue diseñado de tal manera que fuera compatible en software con el 8008 de tal manera que los usuarios de este último, no tuvieran problemas para modificar sus programas e investigaciones y al mismo tiempo, contaran con un dispositivo más funcional con un tiempo de instrucción de sólo 2 microsegundos y que al mismo tiempo reducía la cantidad de componentes extras para implementar un sistema. Además se contaba con mayor cantidad de instrucciones, acceso directo a la memoria (DMA), direccionamiento de 16 bits y memoria para el stack externa, de tal manera que el panorama de aplicaciones se amplió aún más.

El número de componentes necesarios para formar un sistema con el 8080, fue poco a poco reduciéndose desde 30 en el año de

1973, hasta un mínimo de 15 en el año de 1976 utilizando un 8080A y periféricos optimizados.

Siguiendo esta línea de desarrollo fue como en el año de 1977 surgió el microprocesador 8085, el cual además de proveer las ventajas del 8080, lograba un incremento en funcionabilidad y en velocidad así como un decremento apreciable en la cantidad de componentes necesarios para formar un sistema, la cual ahora se redujo a sólo 3. Por otro lado, este nuevo dispositivo utiliza una sola fuente de + 5 volts y conserva en un 100 % el software implementado anteriormente, tal que los programas escritos para el 8080A correrán sin problema en el 8085A.

La reducción en la cantidad necesaria de componentes para lograr implementar un sistema es debido al alto grado de integración que ahora se tiene, pues en el mismo chip microprocesador se tiene ahora generación de la señal de reloj, control del sistema y control de interrupciones; sin tomar en cuenta que además se desarrollaron periféricos de integración a gran escala directamente compatibles con el 8085. Un examen detallado de los componentes que integran el 8085 muestra que cada uno de ellos está diseñado para proveer un mínimo de 400μA y absorber sin problema corrientes del tipo TTL lo cual evita la necesidad de buffers o drivers TTL extra, que reduciría la cantidad de componentes integrados en el mismo chip.

El 8085 y el 8080 no son compatibles, sin embargo en cuanto a la disposición de sus pins, pero esto es debido a la reducción de fuentes de alimentación y a la adición de dispositivos integrados auxiliares. Además, los pins del 8085 fueron cuidadosamente asignados para minimizar el area necesaria del circuito im-

preso, por lo tanto esta incompatibilidad no presenta mayores problemas y sobre todo tomando en cuenta las ventajas logradas de un sistema al otro.

Entre los componentes periféricos diseñados por Intel para formar un sistema mínimo, se encuentran los siguientes:

- Integrado 8155/8156: Memoria RAM de 256 bytes con 2 puertos de entrada y salida de 8 bits y uno programable de 6 bits y además, un timer programable de 14 bits.

- Integrado 8355: Memoria ROM de 2k bytes y 2 puertos de entrada y salida de 8 bits.

- Integrado 8755A: Memoria EPROM de 2k bytes con 2 puertos de 8 bits cada uno.

Los integrados anteriores, requieren de una sola fuente de voltaje de + 5 volts en un chip de 40 pins.

El 8355 y el 8755 son compatibles en cuanto a la disposición de sus pins, lo cual permite al diseñador desarrollar y probar programas en EPROM y cuando éstos son concluídos, pasarlos a memoria ROM sin necesidad de cambiar la disposición de los componentes o implementar un nuevo circuito impreso.

En la figura No. 1 se muestra un diagrama a bloques de las partes que componen al microprocesador 8085A, que como se puede apreciar, cumplen con las funciones de generación de la señal de reloj, control del bus del sistema y selección de prioridad de interrupciones, además de la ejecución del conjunto de instrucciones.

El 8085A transfiere datos en un bus bidireccional de tres estados y 8 bits (AD_0-AD_7), el cual es multiplexado en tiempo de tal manera que también pueda transmitir los ocho bits de me-

nor orden de dirección. Además, ocho líneas adicionales (A_8-A_{15}) expanden la capacidad de direccionamiento a la memoria del sistema a 16 bits, lo cual permite por consiguiente accesar hasta 64k bytes de memoria, directamente por la CPU.

La Unidad Procesadora Central (CPU) del 8085A, genera señales de control que pueden ser usadas para seleccionar los dispositivos externos apropiados y lograr operaciones de lectura y escritura, así como seleccionar entre memoria y puertos de entrada y salida.

El 8085A puede direccionar hasta 256 localidades diferentes de entrada y salida. Estas direcciones tienen los mismos valores numéricos, desde 00 hasta FF hexadecimal, que las primeras 256 direcciones de memoria, pero son distinguidas por medio de la salida IO/M de la CPU.

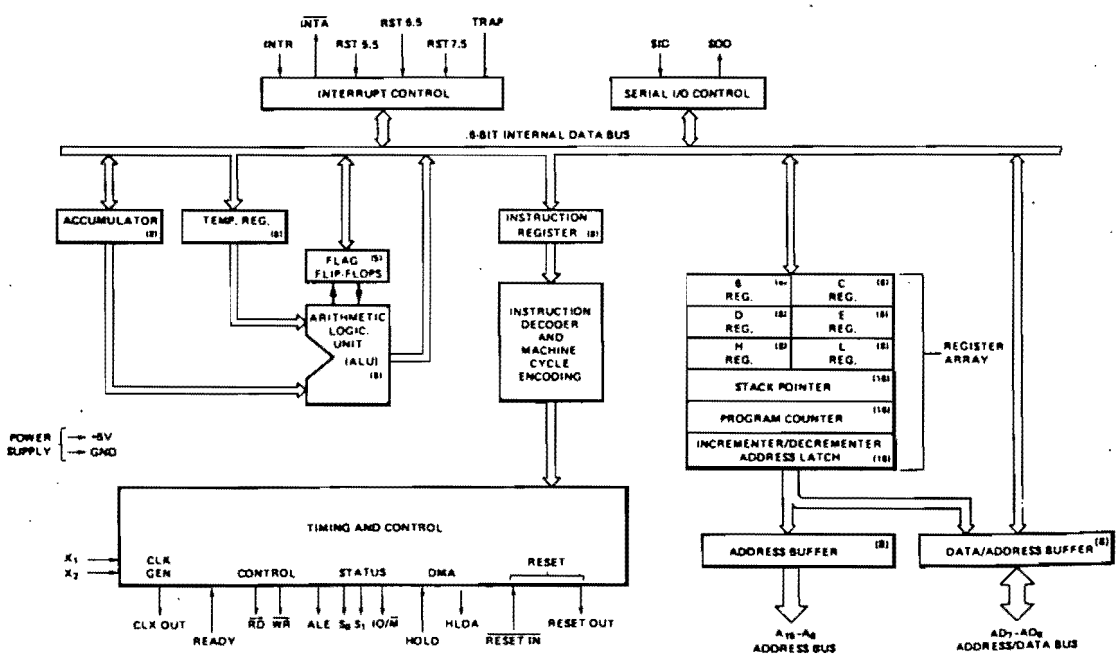


Figura No. 1
DIAGRAMA A BLOQUES FUNCIONAL DEL 8085A

Registros.

El 8085A está provisto con registros internos de 8 y 16 bits. Tiene 8 registros direccionables de 8 bits, de los cuales seis pueden ser usados ya sea como registros de 8 bits o registros pares de 16 bits. Los registros pares son tratados como si fueran registros sencillos de 16 bits, donde el byte de mayor orden le corresponde al primer registro y el byte de menor orden se localiza en el segundo registro. Además de estos registros pares, el 8085A contiene 2 registros más de 16 bits cada uno.

Los anteriores registros son distinguidos de la manera siguiente:

- El acumulador o registro A es el centro de atención de todas las instrucciones de acumulador, las cuales son de diferentes tipos como aritméticas, lógicas, de carga y almacenamiento e instrucciones de entrada y salida. Este es un registro de 8 bits sólamente.

- El contador de programa (PC), el cual siempre apunta hacia la localidad de memoria donde se encuentra la siguiente instrucción a ser ejecutada. Siempre contiene una dirección de 16 bits.

- Los registros de propósito general BC, DE y HL que pueden ser usados como 6 registros independientes de 8 bits o 3 registros pares de 16 bits cada uno, dependiendo de la instrucción especificada. El registro par HL actúa además como un apuntador de datos de dirección de memoria ya sea para fuentes o destinos en cierto número de instrucciones. Un pequeño número de instrucciones utiliza los registros pares BC y DE para direccionamiento indirecto.

- El stack pointer (SP) es un apuntador de datos especial que

siempre está apuntando hacia la parte más alta del stack o porción de memoria así llamada. Este es un registro indivisible de 16 bits.

- El registro de banderas que contiene 7 banderas de un bit, cada una de las cuales guarda información acerca del estatus del procesador para así poder controlarlo también.

Banderas.

Normalmente son consideradas sólo cinco banderas en el 8085A, sin embargo al desarrollar este trabajo se pudo constatar que en realidad son siete, (Ver bibliografía), según se indica en un artículo de la revista Electronics. Dichas banderas se muestran en la figura No. 2.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X_5	AC		P	V	CY

Figura No. 2

La bandera carry (CY) es colocada de acuerdo al acarreo que puede haber durante las operaciones aritméticas y su estado puede ser probado directamente por un programa. Por ejemplo, al efectuar una operación de suma entre dos números de 8 bits, el resultado puede requerir de 9 bits y es en este caso, donde la bandera carry actúa como el bit más significativo del resultado. Esta bandera también actúa como el borrow o préstamo en las operaciones de resta.

La bandera de carry auxiliar (AC), indica el acarreo que pueda haber del bit 3 al bit 4 del acumulador en la misma forma en que el carry lo hace en el bit 7. Esta bandera es comúnmente usada en aritmética BCD.

La bandera de signo (S) es colocada de acuerdo al estado del bit más significativo del acumulador después de la ejecución de instrucciones aritméticas y lógicas. Estas instrucciones usan el bit 7 de datos para representar el signo del número contenido en el acumulador; esto permite la manipulación de números en el rango de - 128 hasta + 127.

La bandera cero (Z) es puesta a uno lógico, si el resultado generado por ciertas instrucciones es igual a cero; si no sucede así entonces esta bandera es puesta a cero lógico.

La bandera de paridad (P) es puesta a 1 si el número de bits del acumulador es par, de lo contrario, esta bandera es borrada.

La bandera de sobreflujo (V) indica un sobreflujo en un complemento a 2 para operaciones aritméticas de 8 y 16 bits.

Y finalmente, la bandera X_5 , así llamada por su posición dentro del byte que la contiene, opera como un indicador de sobreflujo resultante de un cambio de dato de FFFF a 0000 al ejecutar la instrucción INX o al haber un cambio de dato de 0000 a FFFF al ejecutar la instrucción DCX.

El Stack.

El stack pointer mantiene siempre la dirección del último byte guardado en el stack. El stack pointer además, puede ser inicializado para usar cualquier porción de la memoria de lectura y escritura como un stack. El stack pointer es decrementado

cada vez que el dato es puesto dentro del stack y es incrementado cada vez que el dato es sacado fuera del stack; es decir que el stack crece hacia abajo en términos de la dirección de memoria. Nótese que el stack pointer es siempre incrementado o decrementado por dos bytes puesto que todas las operaciones con el stack se aplican a registros pares.

Unidad Aritmética Lógica.

La unidad aritmética lógica (ALU), contiene al acumulador y al registro de banderas, así como algunos registros temporales que son inaccesibles al programador. Las operaciones aritméticas, lógicas y de rotación son desarrolladas por la ALU. Los resultados de estas operaciones pueden ser depositadas en el acumulador o pueden ser transferidas al bus interno de datos.

Registro de Instrucción y Decodificador.

Durante una instrucción "fetch", el primer byte de una instrucción que contiene el opcode, es transferido desde el bus interno al registro de instrucción de 8 bits, de donde pasará al decodificador de instrucción. La salida del decodificador, comandada por las señales de tiempo, controla los registros, la unidad aritmética lógica y los buffers de datos y dirección. Las salidas del decodificador de instrucción junto con el generador de señal de reloj interno, producen las señales de tiempo del ciclo de máquina y las de estatus.

Generador de reloj interno.

El 8085A incorpora como ya se había mencionado antes, un ge-

nerador de reloj completo dentro del mismo chip. Este generador sólo requiere de la adición de un cristal de cuarzo para establecer los tiempos de operación. También puede aceptar un reloj externo aplicado a una de sus entradas. El cristal que sea acoplado al 8085A estándar, debe ser resonante en paralelo a una frecuencia fundamental de 6.25 Mhz o menor, que equivale a dos veces la frecuencia de reloj interna deseada. En otra versión, el 8085A-2 puede operar con cristales de hasta 10 Mhz.

El circuito de reloj genera dos señales internas de reloj sin traslape, las cuales controlan internamente al 8085 y no son puestas al alcance del usuario directamente fuera del chip. Sin embargo, una de estas señales invertida y comandada por un buffer es la señal CLK que produce el chip para el usuario, la cual tiene la mitad de frecuencia de la señal de entrada del cristal y puede ser usada para gobernar otros dispositivos del sistema.

Interrupciones.

Las cinco entradas de interrupción provistas en el 8085A son de tres tipos. INTR que es idéntica a aquella del 8080A llamada INT en cuanto a su función, es decir que es mascarable, pudiendo ser habilitada o deshabilitada por las instrucciones EI y DI respectivamente y causar que la CPU busque una instrucción RST, externamente colocada en el bus de datos, la cual apunta un salto a alguna de 8 localidades de memoria fijas. INTR también puede ser controlada por el controlador de interrupciones programable 8259, el cual genera instrucciones CALL en lugar de RSTs, y así poder realizar operaciones varias como subrutinas localizadas en cualquier parte de la memoria. Las interrupciones RST 5.5, RST 6.5

y RST 7.5, son diferentes en su función en cuanto a que éstas son mascarables a través de la instrucción SIM, la cual habilita o deshabilita estas interrupciones borrando o colocando las correspondientes banderas de máscara basadas en el dato del acumulador. Además para leer el estatus actual del enmascaramiento se cuenta con la instrucción RIM que pone la información en el acumulador.

Las interrupciones RST 5.5, 6.5 y 7.5 también están sujetas a ser habilitadas o deshabilitadas por las instrucciones EI y DI.

El tercer tipo de interrupciones es TRAP. Esta entrada no está sujeta a ningún tipo de enmascaramiento o instrucción para habilitarla o deshabilitarla.

Para ser reconocidas, las interrupciones válidas deben ocurrir al menos 160 ns antes del reconocimiento que ocurre durante el flanco descendiente de CLK, en el 8085A, o 150 ns en el 8085A-2. Además como este reconocimiento ocurre un ciclo antes del fin de la instrucción que se desarrollaba cuando la interrupción ocurre, algunas interrupciones como las RST 5.5 y 6.5 y TRAP que necesitan permanecer en nivel alto hasta ser reconocidas, deben mantenerse al menos 17 ciclos de reloj más 160 o 150 ns, suponiendo que la instrucción fuera de las que requieren de más ciclos como lo es la instrucción CALL.

Las interrupciones tienen prioridad para ser reconocidas y ésta, expresada de mayor a menor, es de la siguiente manera: TRAP, RST 7.5, RST 6.5, RST 5.5 e INTR.

Entrada y Salida en serie.

Los pins SID y SOD del 8085A, ayudan a minimizar la cuenta de

chips en pequeños sistemas, pues proveen fácil interface a puertos serie usando software para manejar en tiempo y para codificar o decodificar los datos. Cada vez que es ejecutada una instrucción RIM, el estatus del pin SID es colocado en el bit 7 del acumulador. De igual manera, cada vez que se efectúa una instrucción SIM, el bit 7 del acumulador se usa para mandar la señal al pin SOD por medio de un flip-flop interno, siempre y cuando el bit 6 del acumulador se mantenga en uno lógico.

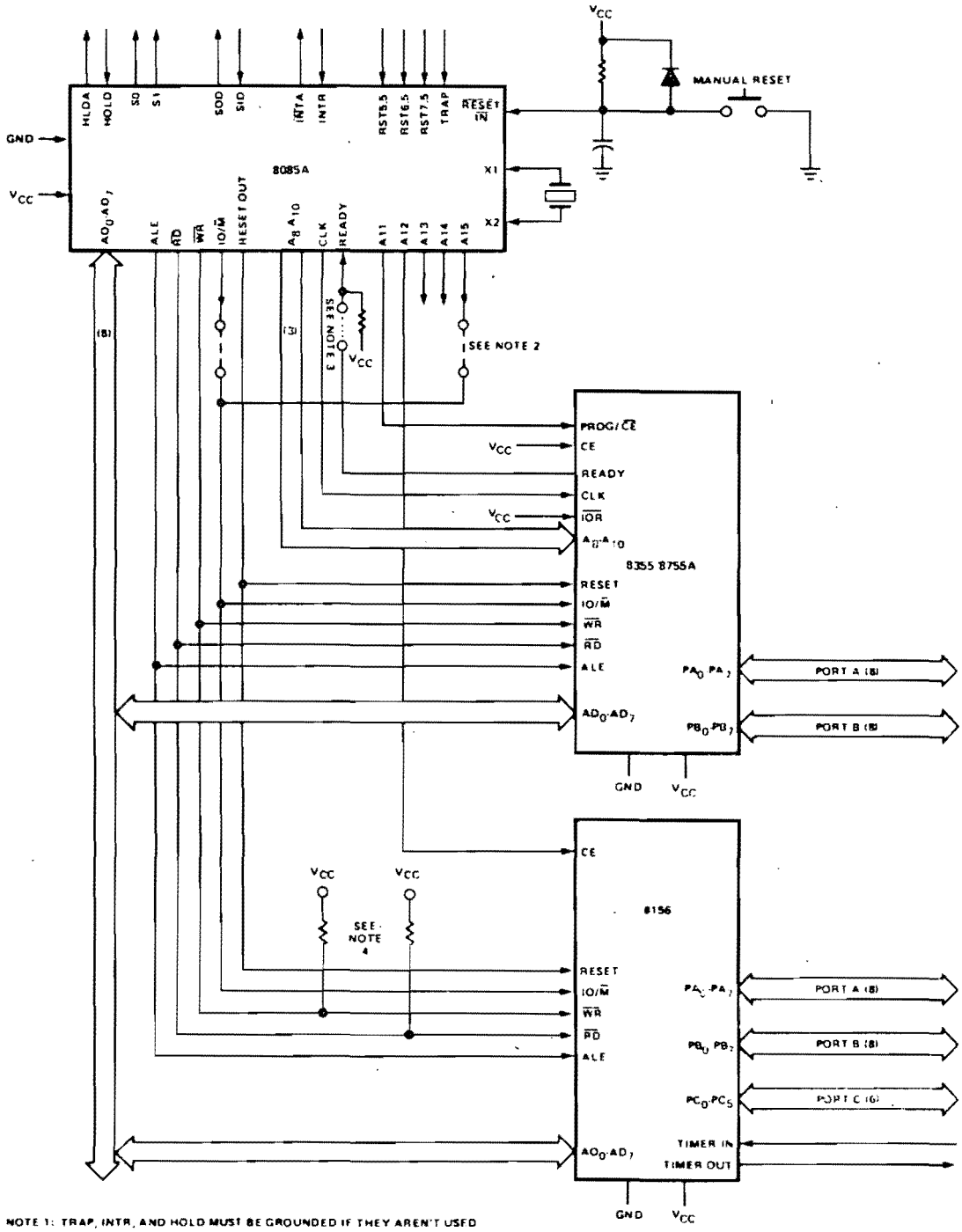
SID también puede ser usada como una entrada de prueba de propósito general y SOD puede servir como un control de salida de un bit.

Sistema microcomputador mínimo (MCS-85).

Como ya se había mencionado antes, la alta escala de integración usada en la fabricación del microprocesador 8085A y periféricos compatibles, hace posible la implementación de un sistema mínimo de sólamente 3 chips. En la figura No. 3 puede notarse el esquema de uno de estos sistemas mínimos, que como se puede apreciar emplea muy pocos componentes para construirse: 1 8085A, 1 - 8355/8755A, 1 8156, 1 cristal, 4 resistencias, 1 capacitor, 1 diodo y una fuente de voltaje de + 5 volts.

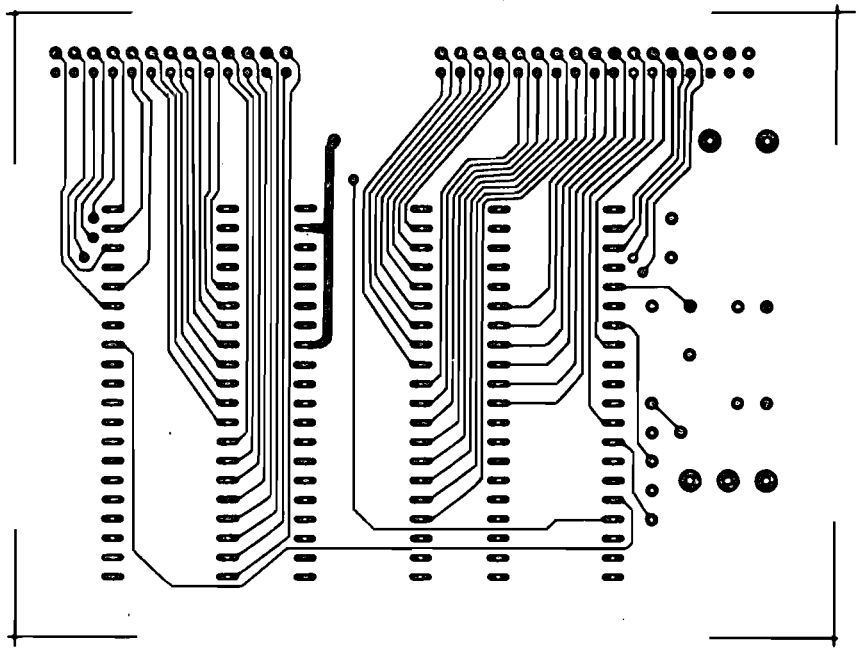
Con este mínimo sistema microcomputador, se logran las siguientes funciones: 1 CPU con un ciclo de reloj menor o igual a 320 ns, 2048 bytes de memoria EPROM o ROM, 256 bytes de memoria RAM, 38 líneas de entrada y salida, 5 interrupciones, 1 contador/timer programable, 1 cristal y oscilador, 1 reloj y un reset para la alimentación. Por otro lado, en la figura No. 4 se puede ver el trazado del circuito impreso para este sistema mínimo, el cual es extrema-

damente sencillo y eficiente.

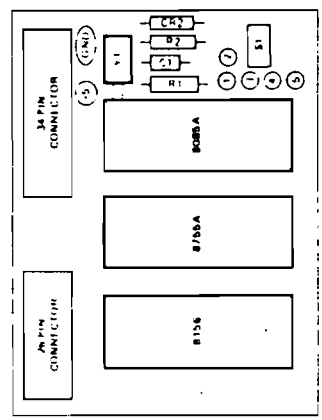


NOTE 1: TRAP, INTR, AND HOLD MUST BE GROUNDED IF THEY AREN'T USED
NOTE 2: USE IO/M FOR STANDARD I/O MAPPING. USE A15 FOR MEMORY MAPPED I/O
NOTE 3. CONNECTION IS NECESSARY ONLY IF ONE T_{WAIT} STATE IS DESIRED.
NOTE 4. PULL-UP RESISTORS RECOMMENDED TO AVOID SPURIOUS SELECTION WHEN $\overline{RD}$ AND $\overline{WR}$ ARE 3-STATED. THESE RESISTORS ARE NOT INCLUDED ON THE PC BOARD LAYOUT OF FIGURE 3.)

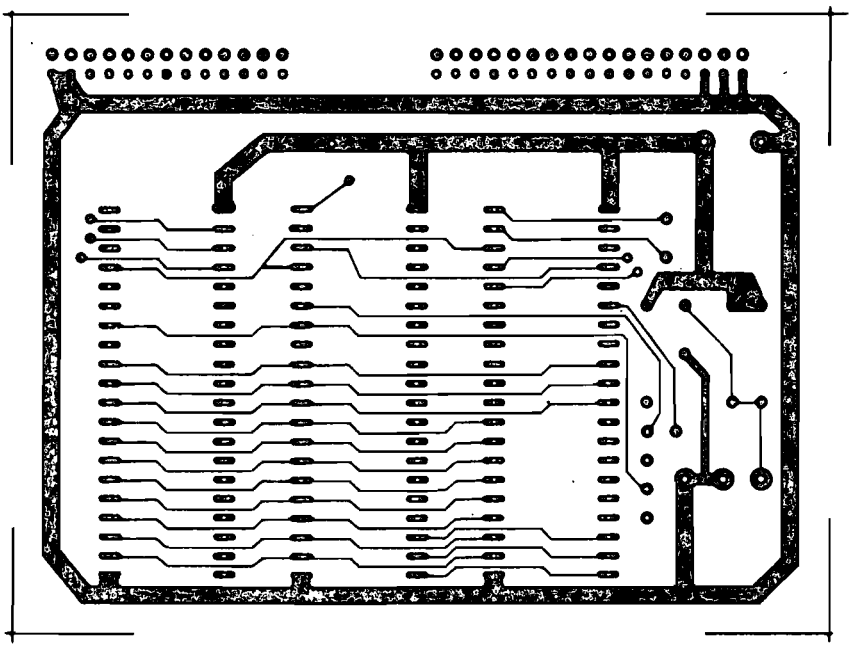
Figura No. 3
ESQUEMA DEL SISTEMA MINIMO 8085



COMPONENT SIDE
SCALE: ≈1:1



COMPONENT LAYOUT



SOLDER SIDE
SCALE: ≈1:1

Figura No. 4
CIRCUITO IMPRESO PARA EL SISTEMA MINIMO 8085

Conjunto de Instrucciones.

Una computadora, no importa que tan sofisticada sea, sólo puede hacer aquello que se le ordena y para lo cual fue construída. Un programa es una secuencia de instrucciones, las cuales son reconocidas por la computadora y causa que ésta desarrolle una determinada operación. Una vez que un programa es colocado en un espacio de la memoria, éste será accesible a la CPU y se podrá correr el programa tantas veces como uno quiera para resolver el mismo tipo de problemas o realizar la misma función. El conjunto de instrucciones al cual la CPU del 8085A responde, es permanentemente fijado durante el diseño del chip.

Cada instrucción de computadora nos permitirá iniciar el desarrollo de una operación específica.

El 8085A tiene un grupo de instrucciones que mueve datos entre registros, entre registros y memoria, y entre un registro y un puerto de entrada y salida. También tiene instrucciones aritméticas y lógicas, instrucciones de salto condicionales e incondicionales, e instrucciones de control de la máquina. La CPU reconoce estas instrucciones sólo cuando están codificadas en forma binaria.

La memoria usada en el sistema MCS-85, está organizada en bytes (8 bits). Cada byte tiene una localización única en la memoria física. Esa localización es descrita mediante una dirección de 16 bits, por lo cual se posible direccionar hasta 65,536 bytes de memoria, que puede ser de acceso aleatorio de lectura y escritura (RAM) o del tipo de sólo lectura ROM.

Los datos en el 8085A son almacenados en la forma de 8 bits binarios enteros dentro de una palabra, donde el bit 0 situa-

do en el extremo izquierdo de la palabra, representa al bit menos significativo y el bit 7 situado en el otro extremo, representa al bit más significativo.

Una instrucción de programa para el 8085A puede ser de uno, dos o tre bytes de longitud. Las instrucciones de varios bytes deben ser almacenadas en localidades de memoria sucesivas, donde la dirección del primer byte es utilizada siempre como la dirección de la instrucción. El formato exacto de cada instrucción dependerá de la operación en particular que se desea ejecutar. El primer byte en las instrucciones de 2 y 3 bytes, es el correspondiente al opcode.

Considerando las 10 instrucciones no especificadas normalmente por el fabricante, y que sin embargo forman parte del conjunto de instrucciones del 8085A según se demuestra en el artículo de Electronics mencionado anteriormente, se tiene un total de -- 256 instrucciones válidas para el 8085A.

Al final de este capítulo se presentan algunas hojas de datos donde se podrá ver también el listado de las instrucciones que posee el 8085A, junto con el mnemónico usado, el código de operación y los ciclos de reloj que requieren.

Modos de Direccionamiento.

Frecuentemente los datos que van a ser operados, se encuentran almacenados en la memoria. Existen en el 8085A, cuatro diferentes modos para direccionar los datos almacenados en la memoria o en los registros:

- Directo. Los bytes 2 y 3 de la instrucción, contienen la dirección exacta de memoria donde se almacena el dato. Los bits

de menor orden de la dirección están en el byte 2 y los de mayor orden en el byte 3.

- Registro. La instrucción especifica el registro o registro par en el cual se localiza el dato.

- Registro Indirecto. La instrucción especifica un registro par, el cual contiene la dirección de memoria donde se localiza el dato. El byte de mayor orden está en el primer registro y el de menor orden en el segundo.

- Inmediato. La instrucción contiene el dato mismo. Este puede ser una cantidad de 8 o de 16 bits. El byte menos significativo primero y el más significativo en segundo lugar.

En las instrucciones de salto, la dirección de la siguiente instrucción a desarrollar, puede especificarse de dos maneras:

- Directo. La instrucción de salto contiene la dirección de la siguiente instrucción a ser ejecutada; excepto para la instrucción RST, el byte 2 contiene los bits de dirección de menor orden y el byte 3 a los de mayor orden.

- Registro Indirecto. La instrucción de salto indica un registro par, el cual contiene la dirección de la próxima instrucción por ejecutar. Los bits de mayor orden de la dirección están en el primer registro y los de menor orden en el segundo.

La instrucción RST, es una instrucción especial de llamado de un byte, usada normalmente durante secuencias de interrupción, que contiene un campo de tres bits que al ser multiplicado por ocho, nos da la dirección de la siguiente instrucción por desarrollar.

Conclusiones.

De todo lo expuesto anteriormente, se pueden hacer las si-

guientes observaciones:

- El sistema formado con el microprocesador 8085A es bastante rápido y supera la velocidad de operación de su predecesor el 8080A.

- Es directamente compatible en cuanto al software con el sistema anterior, por lo que los programas e investigaciones anteriores no se pierden en absoluto.

- El alto nivel de integración permite formar un sistema mínimo de sólo 3 dispositivos integrados.

- Los niveles de voltaje y corriente para las señales de interface, son capaces de manejar grandes cargas de hasta 40 dispositivos MOS o 1 dispositivo TTL schottky.

- El sistema MCS-85 es eficiente tomando en cuenta que las ocho líneas menores de dirección son multiplexadas en tiempo con el bus de datos, lo cual salva 7 pins (el octavo pin es usado para la señal ALE) y permite la integración de mayor número de componentes en un solo chip.

- La disposición de los pins del microprocesador y sus periféricos permite el diseño de un circuito impreso sencillo que asegura que las señales fluyan fácilmente de un chip a otro.

- Requiere una sola fuente de alimentación de +5 volts.

- Incorpora 4 entradas de interrupción adicionales a la ya disponible en el 8080A.

- Es un sistema de bajo costo y gran versatilidad por lo que tiene ante sí un amplio horizonte de aplicaciones.

Hojas de datos.

A continuación se presentan algunas de las hojas de datos

proporcionadas por el fabricante, donde se pueden apreciar aspectos como la disposición de los pins y su función, prioridad de interrupciones con dirección de restart y sensitividad, máximos rangos de operación y características de AC y DC.

Además se presentan tablas con una lista completa del conjunto de instrucciones del 8085A en orden alfabético, en secuencia del código de operación y por grupos funcionales, junto con la lista de las 10 instrucciones no especificadas por el fabricante obtenidas de un artículo de la revista Electronics. En estas tablas se pueden apreciar además, el mnemónico usado, el código de operación y los ciclos de reloj que requiere cada instrucción.

Para mayor información respecto al sistema 8085A refiérase a la primera referencia de la Bibliografía.

8085A INSTRUCTION SET INDEX
Table 4-1

Instruction		Code	Bytes	T States	Machine Cycles	Page
ACI	DATA	CE data	2	7	F R	4-7
ADC	REG	1000 1SSS	1	4	F	4-6
ADC	M	8E	1	7	F R	4-7
ADD	REG	1000 0SSS	1	4	F	4-6
ADD	M	86	1	7	F R	4-6
ADI	DATA	C6 data	2	7	F R	4-6
ANA	REG	1010 0SSS	1	4	F	4-9
ANA	M	A6	1	7	F R	4-10
ANI	DATA	E6 data	2	7	F R	4-10
CALL	LABEL	CD addr	3	18	S R R W W	4-13
CC	LABEL	DC addr	3	9/18	S R/S R R W W	4-14
CM	LABEL	FC addr	3	9/18	S R/S R R W W	4-14
CMA		2F	1	4	F	4-12
CMC		3F	1	4	F	4-12
CMP	REG	1011 1SSS	1	4	F	4-11
CMP	M	BE	1	7	F R	4-11
CNC	LABEL	D4 addr	3	9/18	S R/S R R W W	4-14
CNZ	LABEL	C4 addr	3	9/18	S R/S R R W W	4-14
CP	LABEL	F4 addr	3	9/18	S R/S R R W W	4-14
CPE	LABEL	EC addr	3	9/18	S R/S R R W W	4-14
CPI	DATA	FE data	2	7	F R	4-11
CPO	LABEL	E4 addr	3	9/18	S R/S R R W W	4-14
CZ	LABEL	CC addr	3	9/18	S R/S R R W W	4-14
DAA		27	1	4	F	4-9
DAD	RP	00RP 1001	1	10	F B B	4-9
DCR	REG	00SS S101	1	4	F	4-8
DCR	M	35	1	10	F R W	4-8
DCX	RP	00RP 1011	1	6	S	4-9
DI		F3	1	4	F	4-17
EI		FB	1	4	F	4-17
HLT		76	1	5	F B	4-17
IN	PORT	DB data	2	10	F R I	4-16
INR	REG	00SS S100	1	4	F	4-6
INR	M	34	1	10	F R W	4-8
INX	RP	00RP 0011	1	6	S	4-9
JC	LABEL	DA addr	3	7/10	F R/F R R	4-13
JM	LABEL	FA addr	3	7/10	F R/F R R	4-13
JMP	LABEL	C3 addr	3	10	F R R	4-13
JNC	LABEL	D2 addr	3	7/10	F R/F R R	4-13
JNZ	LABEL	C2 addr	3	7/10	F R/F R R	4-13
JP	LABEL	F2 addr	3	7/10	F R/F R R	4-13
JPE	LABEL	EA addr	3	7/10	F R/F R R	4-13
JPO	LABEL	E2 addr	3	7/10	F R/F R R	4-13
JZ	LABEL	CA addr	3	7/10	F R/F R R	4-13
LDA	ADDR	3A addr	3	13	F R R R	4-5
LDAX	RP	000X 1010	1	7	F R	4-5
LHLD	ADDR	2A addr	3	16	F R R R R	4-5

Instruction		Code	Bytes	T States	Machine Cycles	Page
LXI	RP,DATA16	00RP 0001 data16	3	10	F R R	4-5
MOV	REG,REG	0100 0SSS	1	4	F	4-4
MOV	M,REG	0111 0SSS	1	7	F W	4-4
MOV	REG,M	0100 0110	1	7	F R	4-4
MVI	REG,DATA	00DD D110 data	2	7	F R	4-4
MVI	M,DATA	36 data	2	10	F R W	4-4
NOP		00	1	4	F	4-17
ORA	REG	1011 0SSS	1	4	F	4-10
ORA	M	B6	1	7	F R	4-11
ORI	DATA	F6 data	2	7	F R	4-11
OUT	PORT	D3 data	2	10	F R O	4-16
PCHL		E9	1	6	S	4-15
POP	RP	11RP 0001	1	10	F R R	4-15
PUSH	RP	11RP 0101	1	12	S W W	4-15
RAL		17	1	4	F	4-12
RAR		1F	1	4	F	4-12
RC		D8	1	6/12	S/S R R	4-14
RET		C9	1	10	F R R	4-14
RIM		20	1	4	F	4-17
RLC		07	1	4	F	4-11
RM		F8	1	6/12	S/S R R	4-14
RNC		D0	1	6/12	S/S R R	4-14
RNZ		C0	1	6/12	S/S R R	4-14
RP		F0	1	6/12	S/S R R	4-14
RPE		E8	1	6/12	S/S R R	4-14
RPO		E0	1	6/12	S/S R R	4-14
RRC		0F	1	4	F	4-12
RST	N	11XX X111	1	12	S R R	4-14
RZ		C8	1	6/12	S/S R R	4-14
SBB	REG	1001 1SSS	1	4	F	4-7
SBB	M	9E	1	7	F R	4-8
SBI	DATA	DE data	2	7	F R	4-8
SHLD	ADDR	22 addr	3	16	F R R W W	4-5
SIM		30	1	4	F	4-18
SPHL		F9	1	6	S	4-16
STA	ADDR	32 addr	3	13	F R R W	4-5
STAX	RP	000X 0010	1	7	F W	4-6
STC		37	1	4	F	4-12
SUB	REG	1001 0SSS	1	4	F	4-7
SUB	M	96	1	7	F R	4-7
SUI	DATA	D6 data	2	7	F R	4-7
XCHG		EB	1	4	F	4-6
XRA	REG	1010 1SSS	1	4	F	4-10
XRA	M	AE	1	7	F R	4-10
XRI	DATA	EE data	2	7	F R	4-10
XTHL		E3	1	16	F R R W W	4-16

Machine cycle types:

F	Four clock period instr fetch	X	Variable or optional binary digit
S	Six clock period instr fetch	DDD	Binary digits identifying a destination register: B = 000, C = 001, D = 010, E = 011, H = 100, L = 101, Memory = 110, A = 111
R	Memory read	SSS	Binary digits identifying a source register: B = 000, C = 001, D = 010, E = 011, H = 100, L = 101, Memory = 110, A = 111
I	I/O read	RP	Register Pair: BC = 00, HL = 10; DE = 01, SP = 11
W	Memory write		
O	I/O write		
B	Bus idle		

Figura No. 5
CONJUNTO DE INSTRUCCIONES POR ORDEN ALFABETICO

8085A CPU INSTRUCTIONS IN OPERATION CODE SEQUENCE
Table 4-2

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	28	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	–
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	SIM	5B	MOV E,E	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SP,D16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	–
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	–	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAX B	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	–	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	–	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	–
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	–	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAX D	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DCR E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET Adr	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	–	F6	ORI D8
20	RIM	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	–
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28	–	53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

D8 = constant, or logical/arithmetic expression that evaluates to an 8 bit data quantity.

D16 = constant, or logical/arithmetic expression that evaluates to a 16 bit data quantity.

Adr = 16-bit address.

Figura No. 6
INSTRUCCIONES EN SECUENCIA DEL CODIGO DE OPERACION

NEW CONDITION CODES V = bit 1
X5 = bit 5

Condition code format

S	Z	X5	AC	0	P	V	C

DSUB (double subtraction)

(H) (L) = (H) (L) − (B) (C)

The contents of register pair B and C are subtracted from the contents of register pair H and L. The result is placed in register pair H and L. All condition flags are affected.

0 0 0 0 1 0 0 0 (08)

cycles: 3
states: 10
addressing: register
flags: Z, S, P, CY, AC, X5, V

ARHL (arithmetic shift of H and L to the right)

(H7=H7); (Hn−1) = (Hn)
(L7=Ho); (Ln−1) = (Ln); (CY) = (Lo)

The contents of register pair H and L are shifted right one bit. The uppermost bit is duplicated and the lowest bit is shifted into the carry bit. The result is placed in register pair H and L. Note: only the CY flag is affected.

0 0 0 1 0 0 0 0 (10)

cycles: 2
states: 7
addressing register
flags: CY

RDEL (rotate D and E left through carry)

(Dn+1) = (Dn); (Do) = (E7)
(CY) = (D7); (En+1) = (En); (Eo) = (CY)

The contents of register pair D and E are rotated left one position through the carry flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. Only the CY and the V flags are affected.

0 0 0 1 1 0 0 0 (18)

cycles: 3
states: 10
addressing: register
flags: CY, V

LDHI (load D and E with H and L plus immediate byte)

(D) (E) = (H) (L) + (byte 2)

The contents of register pair H and L are added to the immediate byte. The result is placed in register pair D and E. Note: no condition flags are affected.

0 0 1 0 1 0 0 0 (28)
data

cycles 3
states 10
addressing immediate register
flags none

LDSI (load D and E with SP plus immediate byte)

(D) (E) = ((SPH)(SPL)) + (byte 2)

The contents of register pair SP are added to the immediate byte. The result is placed in register pair D and E. Note: no condition flags are affected.

0 0 1 1 1 0 0 0 (38)
data

cycles: 3
states 10
addressing: immediate register
flags none

2's complement overflow
Underflow (DCX) or overflow (INX)
X5 = 01·02 · 01·R + 02·R, where
01 = sign of operand 1, 02 = sign of operand 2,
R = sign of result. For subtraction and comparisons, replace 02 with $\overline{02}$.

RSTV (restart on overflow)

If (V):
((SP)−1) = (PCH)
((SP)−2) = (PCL)
(SP) = (SP)−2
(PC) = 40 hex

If the overflow flag V is set, the actions specified above are performed; otherwise control continues sequentially.

1 1 0 0 1 0 1 1 (CB)

cycles: 1 or 3
states: 6 or 12
addressing: register indirect
flags: none

SHLX (store H and L indirect through D and E)

((D)(E)) = (L)
((D)(E)+1) = (H)

The contents of register L are moved to the memory location whose address is in register pair D and E. The contents of register H are moved to the succeeding memory location.

1 1 0 1 1 0 0 1 (D9)

cycles: 3
states: 10
addressing: register indirect
flags: none

JNX5 (jump on not X5)

If (not X5):
(PC) = (byte 3) (byte 2)

If the X5 flag is reset, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise control continues sequentially.

1 1 0 1 1 1 0 1 (DD)
low-order address
high-order address

cycles: 2 or 3
states: 7 or 10
addressing: immediate
flags: none

LHLX (load H and L indirect through D and E)

(L) = ((D)(E))
(H) = ((D)(E)+1)

The content of the memory location whose address is in D and E, are moved to register L. The contents of the succeeding memory location are moved to register H.

1 1 1 0 1 1 0 1 (ED)

cycles 3
states: 10
addressing: register indirect
flags: none

JX5 (jump on X5)

If (X5):
(PC) = (byte 3) (byte 2)

If the X5 flag is reset, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction, otherwise control continues sequentially.

1 1 1 1 1 1 0 1 (FD)
low-order address
high-order address

cycles: 2 or 3
states 7 or 10
addressing immediate
flags: none

Figura No. 7
NUEVAS INSTRUCCIONES PARA EL 8085A

8085A INSTRUCTION SET SUMMARY BY FUNCTIONAL GROUPING
Table 4-3

Mnemonic	Description	Instruction Code(1) D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock(2) Cycles
MOVE, LOAD, AND STORE										
MOV r1,r2	Move register to register	0	1	D	D	D	S	S	S	4
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
STACK OPS										
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	12
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	12
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	12
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	12
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	6
JUMP										
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	7/10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	7/10
JZ	Jump on zero	1	1	0	0	1	0	1	0	7/10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	7/10
JP	Jump on positive	1	1	1	1	0	0	1	0	7/10
JM	Jump on minus	1	1	1	1	1	0	1	0	7/10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	7/10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	7/10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	6
CALL										
CALL	Call unconditional	1	1	0	0	1	1	0	1	18
CC	Call on carry	1	1	0	1	1	1	0	0	9/18
CNC	Call on no carry	1	1	0	1	0	1	0	0	9/18
CZ	Call on zero	1	1	0	0	1	1	0	0	9/18
CNZ	Call on no zero	1	1	0	0	0	1	0	0	9/18
CP	Call on positive	1	1	1	1	0	1	0	0	9/18
CM	Call on minus	1	1	1	1	1	1	0	0	9/18
CPE	Call on parity even	1	1	1	0	1	1	0	0	9/18
CPO	Call on parity odd	1	1	1	0	0	1	0	0	9/18
RETURN										
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	6/12
RNC	Return on no carry	1	1	0	1	0	0	0	0	6/12
RZ	Return on zero	1	1	0	0	1	0	0	0	6/12
RNZ	Return on no zero	1	1	0	0	0	0	0	0	6/12
RP	Return on positive	1	1	1	1	0	0	0	0	6/12
RM	Return on minus	1	1	1	1	1	0	0	0	6/12
RPE	Return on parity even	1	1	1	0	1	0	0	0	6/12
RPO	Return on parity odd	1	1	1	0	0	0	0	0	6/12
RESTART										
RST	Restart	1	1	A	A	A	1	1	1	12
INPUT/OUTPUT										
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
INCREMENT AND DECREMENT										
INR r	Increment register	0	0	D	D	D	1	0	0	4
DCR r	Decrement register	0	0	D	D	D	1	0	1	4
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	6
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	6
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	6
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	6
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	6
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	6
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	6
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	6
ADD										
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
SUBTRACT										
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4

Figura No. 8
SUMARIO DEL CONJUNTO DE INSTRUCCIONES POR GRUPOS FUNCIONALES DEL 8085A

8085A INSTRUCTION SET SUMMARY (Cont'd)
Table 4-3

Mnemonic	Description	Instruction Code(1) D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock(2) Cycles
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
LOGICAL										
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive OR register with A	1	0	1	0	1	S	S	S	4
ORA r	OR register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive OR memory with A	1	0	1	0	1	1	1	0	7
ORA M	OR memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive OR immediate with A	1	1	1	0	1	1	1	0	7
ORI	OR immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
ROTATE										
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
SPECIALS										
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
CONTROL										
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable Interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4
HLT	Halt	0	1	1	1	0	1	1	0	5
NEW 8085A INSTRUCTIONS										
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	4
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	4

NOTES: 1. DDD or SSS: B 000, C 001, D 010, E 011, H 100, L 101, Memory 110, A 111.
2. Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

Figura No. 9
CONTINUACION DEL SUMARIO DE INSTRUCCIONES
POR GRUPOS FUNCIONALES

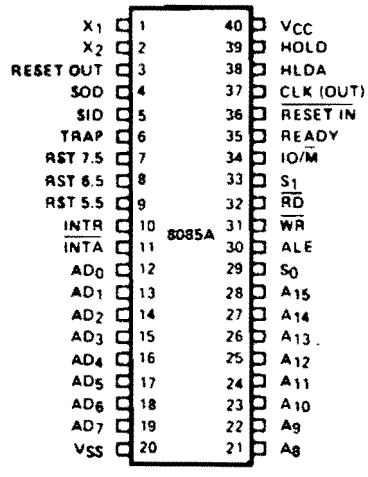


Figure 2. 8085A Pinout Diagram

8085A FUNCTIONAL PIN DEFINITION

The following describes the function of each pin:

Symbol	Function
A_8–A_{15} (Output, 3-state)	Address Bus: The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET.
AD_{0-7} (Input/Output, 3-state)	Multiplexed Address/Data Bus: Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles.
ALE (Output)	Address Latch Enable: It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.
S_0, S_1, and IO/$\overline{M}$ (Output)	Machine cycle status:

IO/$\overline{M}$	S_1	S_0	Status
0	0	1	Memory write
0	1	0	Memory read
1	0	1	I/O write
1	1	0	I/O read
0	1	1	Opcode fetch
1	1	1	Interrupt Acknowledge
*	0	0	Halt
*	X	X	Hold
*	X	X	Reset

* = 3-state high impedance
X = unspecified

Symbol	Function
	S_1 can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$, S_0 and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.
$\overline{RD}$ (Output, 3-state)	READ control: A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer. 3-stated during Hold and Halt modes and during RESET.
$\overline{WR}$ (Output, 3-state)	WRITE control: A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET.
READY (Input)	If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the cpu will wait an integral number of clock cycles for READY to go high before completing the read or write cycle.
HOLD (Input)	HOLD indicates that another master is requesting the use of the address and data buses. The cpu, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data, $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.
HLDA (Output)	HOLD ACKNOWLEDGE: Indicates that the cpu has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The cpu takes the bus one half clock cycle after HLDA goes low.
INTR (Input)	INTERRUPT REQUEST: is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

Figura No. 10
DISPOSICION DE LOS PINS Y SU FUNCION

8085A FUNCTIONAL PIN DESCRIPTION (Continued)

Symbol	Function
$\overline{\text{INTA}}$ (Output)	INTERRUPT ACKNOWLEDGE: Is used instead of (and has the same timing as) $\overline{RD}$ during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.
RST 5.5 RST 6.5 RST 7.5 (Inputs)	RESTART INTERRUPTS: These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupts is ordered as shown in Table 1. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.
TRAP (Input)	Trap interrupt is a nonmaskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. (See Table 1.)
$\overline{\text{RESET IN}}$ (Input)	Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. $\overline{\text{RESET IN}}$ is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay. The cpu is held in the reset condition as long as $\overline{\text{RESET IN}}$ is applied.
RESET OUT (Output)	Indicates cpu is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
X_1, X_2 (Input)	X_1 and X_2 are connected to a crystal, LC, or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
CLK (Output)	Clock Output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.
SID (Input)	Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
SOD (Output)	Serial output data line. The output SOD is set or reset as specified by the SIM instruction.
V_{CC}	+5 volt supply.
V_{SS}	Ground Reference

TABLE 1. INTERRUPT PRIORITY, RESTART ADDRESS, AND SENSITIVITY

Name	Priority	Address Branched To (1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled
RST 7.5	2	3CH	Rising edge latched
RST 6.5	3	34H	High level until sampled
RST 5.5	4	2CH	High level until sampled
INTR	5	See Note (2).	High level until sampled.

NOTES:

(1) The processor pushes the PC on the stack before branching to the indicated address.
(2) The address branched to depends on the instruction provided to the cpu when the interrupt is acknowledged

Figura No. 11
CONTINUA LA FUNCION DE LOS PINS Y TABLA DE PRIORIDAD DE INTERRUPCIONES

TABLE 6. A.C. CHARACTERISTICS

T_A = 0°C to 70°C; V_{CC} = 5V ± 5%; V_{SS} = 0V

Symbol	Parameter	8085A[2]		8085A-2[2] (Preliminary)		Units
		Min.	Max.	Min.	Max.	
t_{CYC}	CLK Cycle Period	320	2000	200	2000	ns
t_1	CLK Low Time	80		40		ns
t_2	CLK High Time	120		70		ns
t_r, t_f	CLK Rise and Fall Time		30		30	ns
t_{XKR}	X_1 Rising to CLK Rising	30	120	30	100	ns
t_{XKF}	X_1 Rising to CLK Falling	30	150	30	110	ns
t_{AC}	A_{8-15} Valid to Leading Edge of Control[1]	270		115		ns
t_{ACL}	A_{0-7} Valid to Leading Edge of Control	240		115		ns
t_{AD}	A_{0-15} Valid to Valid Data In		575		350	ns
t_{AFR}	Address Float After Leading Edge of $\overline{READ}$ ($\overline{INTA}$)		0		0	ns
t_{AL}	A_{8-15} Valid Before Trailing Edge of ALE[1]	115		50		ns
t_{ALL}	A_{0-7} Valid Before Trailing Edge of ALE	90		50		ns
t_{ARY}	READY Valid from Address Valid		220		100	ns
t_{CA}	Address (A_8-A_{15}) Valid After Control	120		60		ns
t_{CC}	Width of Control Low ($\overline{RD}$, $\overline{WR}$, $\overline{INTA}$) Edge of ALE	400		230		ns
t_{CL}	Trailing Edge of Control to Leading Edge of ALE	50		25		ns
t_{DW}	Data Valid to Trailing Edge of $\overline{WRITE}$	420		230		ns
t_{HABE}	HLDA to Bus Enable		210		150	ns
t_{HABF}	Bus Float After HLDA		210		150	ns
t_{HACK}	HLDA Valid to TRailing Edge of CLK	110		40		ns
t_{HDH}	HOLD Hold Time	0		0		ns
t_{HDS}	HOLD Setup Time to Trailing Edge of CLK	170		120		ns
t_{INH}	INTR Hold Time	0		0		ns
t_{INS}	INTR, RST, and TRAP Setup Time to Falling Edge of CLK	160		150		ns
t_{LA}	Address Hold Time After ALE	100		50		ns
t_{LC}	Trailing Edge of ALE to Leading Edge of Control	130		60		ns
t_{LCK}	ALE Low During CLK High	100		50		ns
t_{LDR}	ALE to Valid Data During Read		460		270	ns
t_{LDW}	ALE to Valid Data During Write		200		120	ns
t_{LL}	ALE Width	140		80		ns
t_{LRY}	ALE to READY Stable		110		30	ns

Figura No. 12
CARACTERISTICAS DE A.C.

TABLE 6. A.C. CHARACTERISTICS (Cont.)

Symbol	Parameter	8085A[2]		8085A-2[2] (Preliminary)		Units
		Min.	Max.	Min.	Max.	
t_{RAE}	Trailing Edge of $\overline{READ}$ to Re-Enabling of Address	150		90		ns
t_{RD}	$\overline{READ}$ (or $\overline{INTA}$) to Valid Data		300		150	ns
t_{RV}	Control Trailing Edge to Leading Edge of Next Control	400		220		ns
t_{RDH}	Data Hold Time After $\overline{READ}$ $\overline{INTA}$ [7]	0		0		ns
t_{RYH}	READY Hold Time	0		0		ns
t_{RYS}	READY Setup Time to Leading Edge of CLK	110		100		ns
t_{WD}	Data Valid After Trailing Edge of $\overline{WRITE}$	100		60		ns
t_{WDL}	LEADING Edge of WRITE to Data Valid		40		20	ns

Notes:
1. A_8-A_{15} address Specs apply to IO/$\overline{M}$, S_0, and S_1 except A_8-A_{15} are undefined during T_4-T_6 of OF cycle whereas IO/$\overline{M}$, S_0, and S_1 are stable.
2. Test conditions: t_{CYC} = 320ns (8085A)/200ns (8085A-2); C_L = 150pF.
3. For all output timing where C_L = 150pF use the following correction factors:
   25pF ≤ C_L < 150pF: -0.10 ns/pF
   150pF < C_L ≤ 300pF: +0.30 ns/pF
4. Output timings are measured with purely capacitive load.
5. All timings are measured at output voltage V_L = 0.8V, V_H = 2.0V, and 1.5V with 20ns rise and fall time on inputs.
6. To calculate timing specifications at other values of t_{CYC} use Table 7.
7. Data hold time is guaranteed under all loading conditions.

Figura No. 13
CONTINUACION CARACTERISTICAS DE A.C.

TABLE 4. ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	−65°C to +150°C
Voltage on Any Pin With Respect to Ground	−0.5V to +7V
Power Dissipation	1.5 Watt

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

TABLE 5. D.C. CHARACTERISTICS

(T_A = 0°C to 70°C; V_{CC} = 5V ±5%; V_{SS} = 0V; unless otherwise specified)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	-0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}+0.5	V	
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400μA
I_{CC}	Power Supply Current		170	mA	
I_{IL}	Input Leakage		±10	μA	V_{in} = V_{CC}
I_{LO}	Output Leakage		±10	μA	0.45V ≤ V_{out} ≤ V_{CC}
V_{ILR}	Input Low Level, RESET	-0.5	+0.8	V	
V_{IHR}	Input High Level, RESET	2.4	V_{CC}+0.5	V	
V_{HY}	Hysteresis, RESET	0.25		V	

Figura No. 14
MAXIMOS RANGOS DE OPERACION Y CARACTERISTICAS DE D.C.

PROGRAMA SIMULADOR DEL MICROPROCESADOR 8085.

Generalidades.

La simulación es una área de la técnica moderna que se emplea frecuentemente como una herramienta muy útil en el diseño de sistemas. Mediante ésta y con ayuda de diversos modelos, se simula el comportamiento real de sistemas que caben dentro de todas las actividades humanas.

Dentro de la Ingeniería se pueden simular sistemas de todo tipo, por ejemplo: se hacen simulacros de presas para descubrir los fenómenos asociados a éstas y después poder construir una presa confiable, se efectúan modelos a escala de sistemas mecánicos para probar sus capacidades y limitaciones, o se simula el comportamiento de un dispositivo electrónico mediante el uso de modelos que se asemejan al dispositivo real en cuanto a su comportamiento se refiere.

Actualmente la simulación con ayuda de las computadoras tiene un gran auge debido a las ventajas inherentes a éstas.

En la Ingeniería Electrónica este hecho ha sido de gran utilidad, pues se ha podido diseñar en base a la simulación de dispositivos como transistores, capacitores, resistencias, etc. Además, se pueden simular circuitos completos que de esta manera pueden ser estudiados más fácilmente sin la necesidad de complicados cálculos hechos a mano que nos llevarían una cantidad enorme de tiempo y esfuerzo.

En el presente trabajo se hace la simulación de un sistema electrónico digital que en la última década ha tenido un desarrollo extraordinario debido a sus innumerables ventajas: el microprocesador.

En las siguientes secciones se hará una descripción del trabajo desarrollado, la secuencia del mismo, los problemas para efectuar la simulación y las características y limitaciones del simulador finalmente logrado.

Establecimiento del Problema.

El objetivo de este trabajo, como se había mencionado al principio del mismo, es el de realizar un programa de computadora para la minicomputadora PDP-11/40 que simule el funcionamiento del microprocesador Intel 8085.

Este programa simulador tendrá como entrada un archivo en disco con código objeto para el microprocesador 8085 y ofrecerá a su salida el estatus del microprocesador después que se haya ejecutado cada instrucción del código objeto de entrada.

Además deberá contar con una serie de comandos por medio de la cual el usuario podrá programar el número de instrucciones a ejecutar, la dirección de inicio del programa simulado y el modo de operación, es decir, si debe imprimir el estatus del microprocesador después de ejecutar cada instrucción o sólo imprimir el estatus final al término del total de instrucciones.

En vista de lo anterior, se planteó como necesidad inicial el conocer perfectamente cada una de las dos partes que iban a intervenir en el proyecto, es decir conocer el funcionamiento y el conjunto de instrucciones de cada una de ellas, así como la relación existente entre las mismas.

Por un lado, se estudiaron el total de instrucciones que realiza el microprocesador 8085A y cómo afectan éstas al estatus del mismo; y por el otro, el conjunto de instrucciones de la PDP-11/40

y la forma en que éstas pudieran simular el comportamiento del primero.

Después de haber realizado lo anterior, se llegó a la forma de solución que se presenta en las siguientes secciones.

Estructura del Programa.

En la figura No. 15 de la siguiente hoja, se presenta un diagrama de flujo donde se puede apreciar en forma general, el desarrollo del algoritmo propuesto para este trabajo.

Después de haber inicializado el estatus del microprocesador simulado, lo primero que se realiza es pasar el archivo de disco que contiene el código de operación de un programa desarrollado para el 8085, a una porción de memoria dentro del programa simulador denominada BUFFER.

Los caracteres almacenados en BUFFER están codificados en - ASCII, por lo que será necesario entonces tomar cada uno de los códigos de operación y convertirlos a hexadecimal, ya que éste es el código con el que trabaja el 8085. Al mismo tiempo, se - checa la validez de la sintaxis, es decir que el opcode contenga exclusivamente caracteres hexadecimales.

Después de esta etapa de decodificación, el código de operación es transferido a la memoria simulada del 8085, a partir de la dirección indicada por el usuario.

Cualquier error ocurrido durante el desarrollo de los pasos anteriores, será indicado por el programa y en caso de ser necesario se terminará la simulación para que se corrija el error correspondiente.

A continuación, se toma el opcode de la instrucción a ejecu-

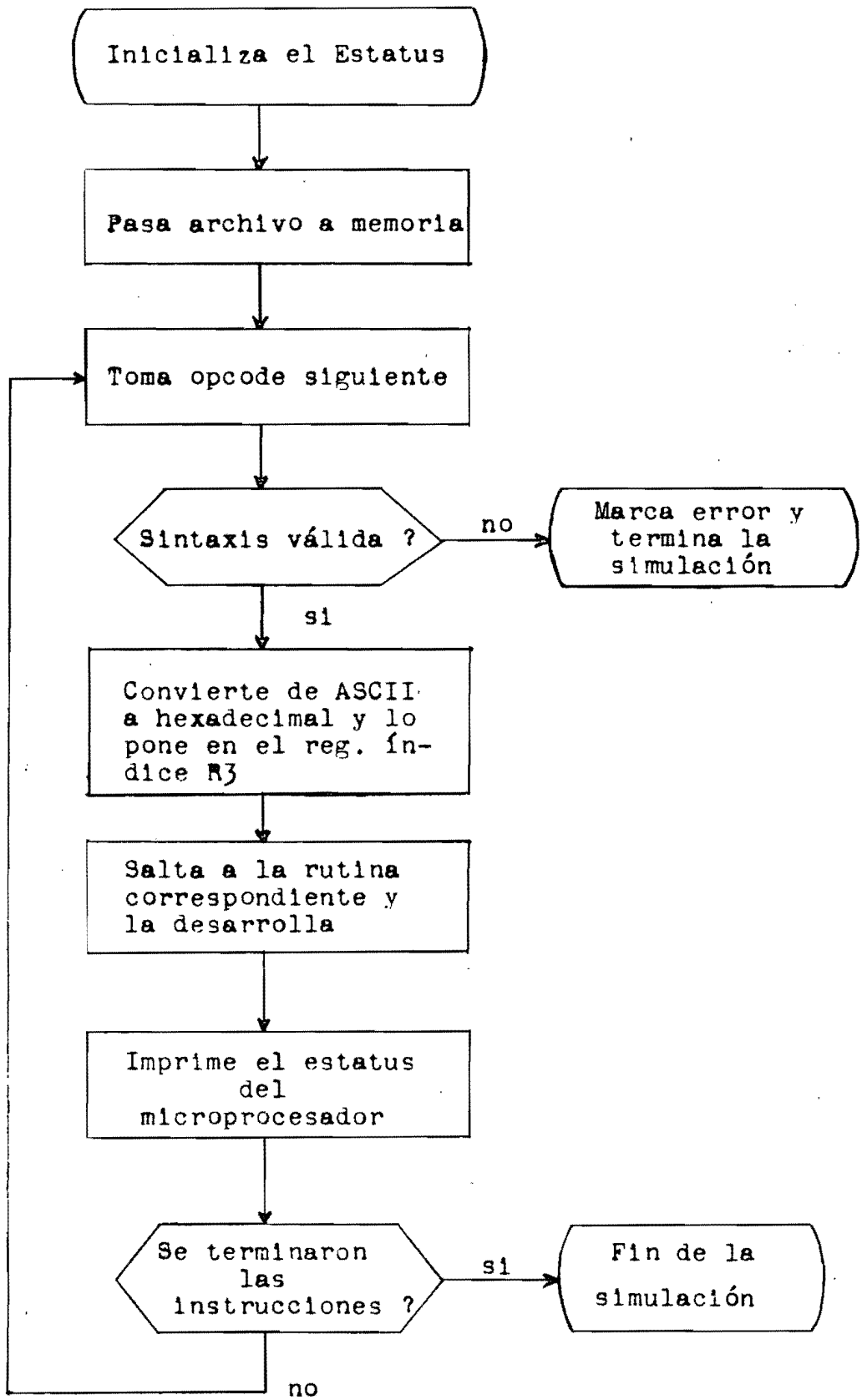


Figura No. 15
DIAGRAMA DE FLUJO DEL PROGRAMA SIMULADOR

tar que puede tener cualquier valor comprendido entre 00 y FF hexadecimal, ya que el 8085 tiene un total de 256 instrucciones.

De acuerdo a este valor y con ayuda de una tabla de direcciones, se salta a la rutina correspondiente al código de entrada.

En esta rutina se encuentra propiamente la simulación de la instrucción que se desarrollaría en el 8085 para el código de operación especificado.

Inmediatamente después de ejecutar la rutina, se pasa a otra que nos imprimirá el estatus del microprocesador simulado para posteriormente tomar el siguiente código de operación y efectuar la siguiente instrucción del programa.

Esta secuencia continúa hasta que se ejecuten todas las instrucciones del programa objeto inicialmente almacenado en disco, tras de lo cual se dará por terminado el programa simulador mostrando el estatus final del microprocesador simulado.

Como se ve, la filosofía general del simulador es bastante sencilla y el problema principal estriba en la simulación de cada una de las instrucciones del 8085.

En la siguiente sección se hará una descripción de los problemas principales que se encontraron durante el desarrollo del programa simulador.

Problemas de Simulación.

El primer obstáculo que se presentó para desarrollar el programa simulador fue el hecho de pasar el archivo de disco a la memoria del programa. Este problema se resolvió gracias a la ayuda de las pseudoinstrucciones .LOCK, .FETCH, .LOOKUP y .READW ("Programmed Requests"), las cuales definen ciertos parámetros

como son: el nombre del dispositivo empleado (en este caso el disco 0), el número del canal por donde se va a efectuar la transición, el nombre del programa objeto, la cantidad de datos para leer, etc.

Después de efectuadas estas operaciones, el archivo que contiene el código de operación del programa objeto, es almacenado en una porción de memoria del programa simulador denominada BUFFER. Nótese que, puesto que cada caracter producido por el teletipo está codificado en ASCII y requiere de un byte para representarlo, cada palabra del BUFFER consistente en 2 bytes, corresponderá a un byte de instrucción del 8085.

Otro de los problemas que se presentaron, fue la representación de los registros del 8085. Estos registros son de una longitud de un byte, a excepción del program counter y el stack pointer, por lo tanto todas las instrucciones referentes a los registros deberían efectuarse en forma de bytes por la PDP. Puesto que la PDP trabaja por lo general con palabras de 16 bits para tener direccionamiento par, se decidió representar a los registros del A al L del 8085 en una palabra (16 bits) de la memoria de la PDP cada uno. De esta manera, se trabajó sólo con la parte menor de la palabra y manteniendo un valor binario de cero en el byte mayor de la palabra.

Para representar el program counter del 8085, se eligió el registro R1 de la PDP y para representar el stack pointer se tomó el registro R2. Además se utilizó el registro R4 como contador del número de instrucciones a ejecutar y el registro R3 como apuntador o índice para saltar a la rutina de simulación de instrucción correspondiente. De esta manera, los registros R0 y R5

se utilizaron como registros de propósito general durante el programa simulador. Finalmente, los registros R6 y R7 fueron utilizados de acuerdo a su función específica dentro de la PDP como apuntador del stack y contador del programa respectivamente.

Uno de los problemas mayores que se presentaron fue la simulación de las banderas. Primero, por la diferencia entre las de la PDP y las del 8085, ya que la PDP sólo tiene 4 banderas que son:N (signo), Z (cero), V (sobreflujo) y C (acarreo), mientras que el 8085, tomando en cuenta las 2 banderas mencionadas en un artículo de la revista Electronics (ver Bibliografía), cuenta con un total de 7 banderas que son: S (signo), Z (cero), X5 (sobreflujo en incrementos), AC (acarreo auxiliar), P (paridad), V (sobreflujo) y CY (acarreo). Y por otro lado, porque la posición de las banderas no era la misma dentro de la palabra que las contenía a unas y otras.

Además, al efectuar la simulación se desarrollaban instrucciones que afectaban irremediablemente el estado de las banderas, por lo tanto las banderas de la PDP no podían servir para representar sus equivalentes del 8085.

Lo anterior se resolvió implementando rutinas que modifican el estatus de las banderas simuladas,contenidas en un byte específico de la memoria denominado PSW. Estas rutinas prueban el estado del resultado obtenido y de acuerdo a éste colocan las banderas al valor que les corresponde.

Lo anterior se cumple para la mayoría de las banderas simuladas excepto para V y CY, las cuales se colocan en el estado de sus análogas V y C de la PDP inmediatamente después de efectuada la instrucción para no afectar el estado de estas últimas.

Otro de los problemas principales que se presentó, fue la simulación de la memoria del 8085. Esto se resolvió apartando un bloque de palabras dentro del programa simulador. El espacio de este bloque denominado MEMORY es el que se utiliza entonces, como la memoria direccionada por el 8085.

Dentro de esta memoria se almacena el código de operación del programa objeto ya decodificado de ASCII a hexadecimal, a partir de la localidad que indique el usuario. Además, el apuntador del stack simulado siempre deberá contener una dirección que caiga dentro de este espacio de memoria. Inicialmente, el stack pointer (SP) es cargado con la dirección mayor de la memoria, pero esta dirección puede modificarse a los deseos del usuario por medio de las instrucciones del programa objeto.

El simulador no distingue entre datos y direcciones o espacio de stack, por lo tanto toca al usuario preveer problemas respecto a esto dentro de su programa objeto.

Por otro lado, el program counter simulado (PC) debe apuntar siempre dentro del espacio de memoria a partir del inicio del programa. Nótese que existe un direccionamiento relativo ya que la dirección de inicio de MEMORY, que tendrá determinado valor, corresponderá a la dirección 0000 de la memoria simulada.

Capacidades, Características y Limitaciones.

A continuación se ennumeran las características propias del programa simulador que finalmente fue desarrollado.

-En primer término, como se indicó antes, el tamaño del BUFFER que recibe el archivo de disco tiene una capacidad máxima de 270 palabras que equivalen a 256 bytes de instrucción, más 14 pala-

bras disponibles donde poder meter caracteres como carriage return y line feed, cuya función es únicamente la de permitir un arreglo práctico y fácil de apreciar en la pantalla, del código objeto.

Esta magnitud del BUFFER podría modificarse de acuerdo a las necesidades del usuario, por ejemplo cuando las instrucciones del programa objeto incluyen instrucciones de salto que producen lazos múltiples y aumenten la cantidad de instrucciones necesarias para cubrir el programa.

-Por otro lado, se implementó una rutina por medio de la cual el programa simulador proporciona al usuario la facilidad de escoger el número de instrucciones a ejecutar, el modo de obtener el estatus del microprocesador y la dirección de inicio del programa objeto. En la siguiente sección se verá con detalle la forma de utilizar esta rutina.

-Para representar el estatus del microprocesador simulado, se toman en cuenta los registros del mismo, estos son: el registro A o acumulador, los registros pares BC, DE y HL, el registro de banderas o "processor status word" (PSW), el program counter (PC) y el stack pointer (SP). Además se indica el contenido de los 2 bytes situados en la parte alta del stack y el contenido de la memoria direccionada por el registro par HL.

-Cualquier error en el procesamiento del archivo de disco, la entrada de comandos o la decodificación, se indica visualmente especificando el tipo de error cometido y tomando las medidas necesarias hasta que la causa del error desaparezca.

-En el paquete de rutinas para simular las instrucciones del 8085, se incluyeron aquellas especificadas en el artículo de E-

lectronics mencionado, que suman un total de 10. Por lo tanto - cualquier código de operación comprendido entre 00 y FF es válido para el archivo en disco.

-Hubo 6 instrucciones que no se simularon, que son: EI, DI, SIM, RIM, IN port y OUT port, ya que éstas tienen que ver con el hardware del sistema y no afectan a las variables mostradas como estatus del microprocesador. Sin embargo, en investigaciones posteriores se ve la factibilidad de poder hacerlo y así tener una simulación exacta del 8085.

-Para las operaciones con memoria, se dispone de 4096 bytes que son apartados inicialmente dentro de la memoria de la PDP. Note que lo anterior delimita el direccionamiento de memoria a los valores hexadecimales comprendidos entre 0000 y 0FFF.

-Finalmente, el tamaño de la memoria utilizada en la PDP para contener al programa simulador es de 14,801 bytes, por lo tanto se ve la posibilidad de ampliar el tamaño de la memoria simulada para cubrir el direccionamiento completo del 8085 y/o añadir rutinas que ayuden a simular en forma total el microprocesador.

Instrucciones de Manejo.

-El primer paso que se debe dar para poder utilizar el programa simulador almacenado en la minicomputadora PDP-11/40 de la - DEPFI, es editar el programa objeto y guardarlo en el disco 0 con el nombre RK0:FERNA.OBJ.

Para que el archivo sea correcto, deben almacenarse en forma secuencial las instrucciones codificadas en hexadecimal y deberá ponerse siempre el símbolo "/" al final del programa. Además de los caracteres mencionados sólo es permitido el uso de carriage

return-line feed con objeto de poder ordenar el archivo de una manera práctica y objetiva.

El número máximo de instrucciones de un byte es de 256. Si se utilizan instrucciones de 2 o 3 bytes recuerde sólamente que no debe sobrepasar el máximo de 256 bytes, es decir 512 caracteres hexadecimales.

-A continuación corra el programa simulador denominado RK1: JFGNC.333. El simulador imprimirá los encabezados y después el aviso de espera de comandos.

Los comandos deben especificarse por medio del teletipo de la siguiente manera:

XXX/M/YYYY ?

donde XXX se refiere al número de instrucciones a ejecutar y puede tener un valor decimal comprendido entre 000 y 256 inclusive; M indica el modo de operación y sólo puede tomar 2 valores 0 o 1, y YYYY que indica la dirección de inicio del programa objeto, pudiendo sólamente tener valores comprendidos entre 0000 y 0F00 hexadecimales.

La variable M con un valor cero (0), indica que el programa debe dar el estatus del 8085 inmediatamente después que se haya efectuado cada una de las instrucciones. Con un valor de uno (1), el programa sólo imprimirá el estatus final del 8085 después de haber efectuado el total de instrucciones especificadas en XXX o al término del programa si éste ocurre antes.

Cualquier variación a la sintaxis de los comandos, representará un error que será indicado por el programa para luego esperar una nueva secuencia de comandos correcta, tras de lo cual pasará a desarrollar las instrucciones del programa objeto hasta

finalizar.

Cada vez que se efectúe una instrucción, el simulador imprimirá el mnemónico de ésta y al finalizar el programa se imprime el mensaje de "fin de la simulación", pasando el control de la máquina al programa monitor.

El estatus inicial del microprocesador simulado se coloca con los registros A, BC, DE, HL y PSW a un valor cero, PC (R1) con la dirección inicial del programa objeto en la memoria, SP (R2) con la dirección inmediata superior al fin del bloque de memoria de 4096 bytes y el contenido del stack pointer y la localidad de memoria direccionada por HL serán aleatorios.

Finalmente, cabe hacer la aclaración que las instrucciones HLT y NOP se modificaron ligeramente con el fin de hacerlas útiles en el desarrollo del programa. Cuando se simula la instrucción HLT, se detiene la ejecución de instrucciones, se imprime el estatus del microprocesador hasta ese momento y el programa espera que el usuario teclee un carriage return para continuar ejecutando las siguientes instrucciones.

La instrucción NOP se utiliza como indicación de terminación del programa, por lo tanto al encontrarse en la memoria el correspondiente código el programa lo interpretará como la última instrucción imprimiendo el estatus final del microprocesador y dando por terminada la simulación.

Los pasos descritos en esta sección deben repetirse en forma secuencial si se desea volver a correr el programa o algún otro, para lo cual se requeriría cargar este último en un archivo del disco 0 como se indica al principio.

En la última parte de este trabajo se muestra una copia del

programa completo, donde aparecen algunos comentarios laterales que pueden servir de ayuda para comprender la secuencia de operaciones desarrolladas; y algunos ejemplos prácticos de aplicación del mismo.

CONCLUSIONES.

El programa simulador planteado originalmente como el objetivo de este trabajo fue desarrollado en la Minicomputadora -- PDP-11/40 perteneciente al Laboratorio de Cálculo Automatizado para el Diseño de la DEPFI y corrió satisfactoriamente simulando la operación del 8085.

Salvo algunas excepciones, se logró simular la mayor parte de las instrucciones del 8085 y también se logró visualizar y comprender más objetivamente el funcionamiento tanto de la PDP como del microprocesador 8085.

Se ve la necesidad de implementar instrucciones tales como las de interrupción, interface con periféricos y control de éstas, con lo cual se simularía en forma total el microprocesador. Asimismo, aprovechar al máximo la capacidad de direccionamiento de 64K del 8085 y la alternativa de poder simular programas de más de 256 instrucciones.

Se creó una fuerte inquietud para seguir investigando en esta interesante área y aplicar el método de simulación sobre otro tipo de microprocesadores o sistemas afines.

Por otro lado, se comprueba el planteamiento original de buscar la simulación como un método de reducir el costo en la implementación de sistemas de desarrollo basados en microprocesadores,

sobretodo cuando ya se dispone de un sistema de cómputo como en este caso lo es la PDP 11/40.

Finalmente, se vislumbra una amplia aplicación para la simulación de microprocesadores, como herramienta para el diseño de sistemas digitales.

BIBLIOGRAFIA.

- Intel Corporation, MCS-85 USER'S MANUAL.
September 1978.

- Richard H. Eckhouse Jr. & L. Robert Morris, MINICOMPUTER SYSTEMS. Organization, Programming and Applications (PDP-11).
Second Edition, Prentice Hall, 1979.

- John J. Donovan, SYSTEMS PROGRAMMING.
New York, McGraw-Hill, 1972.

- Wolfgang Dehnhardt & Villy M. Sorensen, UNSPECIFIED 8085 OP-CODES ENHANCE PROGRAMMING.
"Electronics", January 18, 1979.

- Digital Equipment Corporation, RT-11 SYSTEM REFERENCE MANUAL.
1976.

- Digital Equipment Corporation, PDP-11 PROCESSOR HANDBOOK.
1975.

```
$
;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

;             "PROGRAMA SIMULADOR DEL MICROPROCESADOR 8085"

;                AUTOR: JOSE FERNANDO GARCIA NUNEZ CANO.


;        DIVISION DE ESTUDIOS DE POSGRADO DE LA FACULTAD DE INGENIERIA

;                          U N A M - 1 9 8 1

;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

.TITLE SIMULADOR DEL INTEL 8085
.MCALL ..V2..,.REGDEF,.TTYIN,.TTYOUT,.TTINR
.MCALL .PRINT,.EXIT,.LOCK,.FETCH,.LOOKUP,.READW
..V2..
.REGDEF

;-----------------------------------------------------------------------
;        RUTINA QUE DA INSTRUCCIONES PARA LEER UN ARCHIVO DE DISCO
;-----------------------------------------------------------------------
META1:  .LOCK
        .FETCH #HUECO,#DISCO
        BCS ERFTCH
        .LOOKUP #AREA,#5,#FILINP
        BCC META2
        TSTB @#52
        BNE ERLKFN
        .PRINT #ELOOKO
        .EXIT

;-----------------------------------------------------------------------
;        RUTINAS PARA MARCAR ERRORES EN EL PROCESAMIENTO DEL DISCO
;-----------------------------------------------------------------------
ERFTCH: .PRINT #EFETCH
        .EXIT

ERLKFN: .PRINT #ELOOK1
        .EXIT

ERREAD: CMPB @#52,#1
        BEQ ERREHW
        BHI ERREFN
        .PRINT #ERREAO
        .EXIT

ERREHW: .PRINT #ERREA1
        .EXIT

ERREFN: .PRINT #ERREA2
        .EXIT

;-----------------------------------------------------------------------
;        RUTINA QUE LEE EL ARCHIVO E IMPRIME ENCABEZADOS
;-----------------------------------------------------------------------
META2:  .READW #AREA,#5,#BUFFER,#270.,#0
        BCS ERREAD
        .PRINT #ENCAB1          ;Imprime encabezados.
        .PRINT #SIGREN
        .PRINT #ENCAB2
```

```
        .PRINT #SIGREN
        .PRINT #ENCAB3
        .PRINT #SIGREN
        .PRINT #ENCAB1
        .PRINT #SIGREN
        MOV #META1,SP

;-------------------------------------------------------------------------
;                RUTINA QUE ESPERA LOS COMANDOS
;-------------------------------------------------------------------------
        .PRINT #TEXTO1          ;Imprime aviso de espera de comandos.
        .PRINT #TEXTO2
LIMPIA: .TTINR                  ;Limpia el buffer del teletipo.
        BCC LIMPIA
        .TTYIN                  ;Espera el primer caracter.
        JSR PC,SINTX1           ;Checa la sintaxis y convierte a binario.
        MOV R0,R5
        MUL #100.,R5            ;Le da el valor de centenas.
        .TTYIN                  ;Espera el segundo caracter.
        JSR PC,SINTX1           ;Checa la sintaxis y convierte a binario.
        MOV R0,R3
        MUL #10.,R3             ;Le da el valor de decenas.
        ADD R3,R5               ;Suma decenas y centenas.
        .TTYIN                  ;Espera el tercer caracter.
        JSR PC,SINTX1           ;Checa que sea numero.
        ADD R0,R5               ;Obtiene el No. de instrucciones.
        CMP R5,#256.            ;Comprueba que no sea mayor de 256 y si
        BHI ERR1                ;lo es manda mensaje de error.
        INC R5
        MOV R5,R4               ;Inicializa el contador de instrucciones.
        .TTYIN                  ;Espera el caracter "/" y si no es el si-
        CMPB R0,#57             ;guiente que llega, manda un mensaje de
        BNE ERR1                ;error.
        .TTYIN                  ;Espera el siguiente caracter que debe -
        CMPB R0,#61             ;ser "0" o "1", indicando el modo de ope-
        BHI ERR1                ;racion del programa. Si no cumple lo an-
        CMPB R0,#60             ;terior, manda un mensaje de error.
        BLO ERR1
        SUB #60,R0              ;Convierte a binario.
        MOVB R0,MODO            ;Inicializa la bandera "MODO".
        .TTYIN
        CMPB R0,#57
        BNE ERR1
        .TTYIN
        JSR PC,SINTX0
        MOV R0,R1
        ASH #12.,R1
        .TTYIN
        JSR PC,SINTX0
        ASH #8.,R0
        ADD R0,R1
        .TTYIN
        JSR PC,SINTX0
        ASH #4,R0
        ADD R0,R1
        .TTYIN
        JSR PC,SINTX0
        ADD R0,R1
        CMP R1,#7400
        BHI ERR1
        ADD #MEMORY,R1
```

```
        .TTYIN                  ;Espera el siguiente caracter que debe
        CMPB R0,#15             ;ser un "carriage return". Si no lo es
        BNE ERR1                ;manda un mensaje de error.
        MOV #BUFFER,R2
        MOV R1,R5
OTR1:   MOV (R2)+,R0
        JSR PC,SINTAX
        MOVB R0,R3
        ASH #4,R3
        SWAB R0
        JSR PC,SINTAX
        ADD R0,R3
        MOVB R3,(R5)+
        JMP OTR1
EJEC:   MOV #MEMORY+4096.,R2
        .PRINT #STAINI          ;Imprime el status inicial del micro-
        JSR PC,ESTADO           ;procesador simulado.
        JMP NOIMP

;-------------------------------------------------------------------
;       RUTINAS PARA BRINCAR A LA INSTRUCCION CORRESPONDIENTE
;-------------------------------------------------------------------
META3:  TSTB MODO               ;Prueba la bandera "MODO" y si es di-
        BNE NOIMP               ;ferente de cero no imprime el estatus.
        JSR PC,ESTADO
NOIMP:  DEC R4                  ;Decrementa el contador de instrucciones
        BEQ FIN                 ;y si es cero, termina la simulacion.
OTRO:   MOVB (R1)+,R3           ;Trae el codigo de operacion.
        BIC #177400,R3
        TSTB M                  ;Prueba la bandera "M" para ver si uti-
        BEQ MULT1               ;liza la rutina OTRO como subrutina y si
        RTS PC                  ;asi es regresa a donde fue llamada.

MULT1:  ASL R3                  ;Da direccionamiento par.
        JMP @TABLA(R3)          ;Salta a la rutina correspondiente.

;-------------------------------------------------------------------
; RUTINAS PARA INFORMAR ERRORES DE SINTAXIS Y TERMINACION DEL PROGRAMA
;-------------------------------------------------------------------
ERROR:  .PRINT #CINV            ;Hubo caracteres invalidos en el opcode
        .EXIT                   ;e imprime el mensaje de error.

ERR1:   .PRINT #TEXERR          ;Hubo caracteres invalidos en los co-
        JMP LIMPIA              ;mandos e imprime el mensaje de error.

FIN:    .PRINT #STATUS
        JSR PC,ESTADO
        .PRINT #ACABO           ;Se termina la simulacion e imprime el
        .EXIT                   ;mensaje correspondiente.

;-------------------------------------------------------------------
;       RUTINAS PARA CHECAR LA VALIDEZ DEL OPCODE Y COMANDOS
;-------------------------------------------------------------------
SINTAX: CMPB R0,#60             ;En esta rutina se verifica que los ca-
        BHIS SIGUE              ;racteres del opcode correspondan solo a
        CMPB R0,#57             ;digitos hexadecimales, a un "carriage
        BEQ EJEC                ;return" (al cual no toma en cuenta), o
        CMPB R0,#15             ;al simbolo "/" que significara la ter-
        BNE ERROR               ;minacion del opcode. Si no se cumple lo
        JMP OTR1                ;anterior imprime mensaje de error.
SIGUE:  CMPB R0,#106
```

```
        BHI ERROR
        CMPB R0,#71
        BLOS NUM
        CMPB R0,#101
        BLO ERROR
        SUB #7,R0
NUM:    SUB #60,R0
        RTS PC

SINTX0: CMPB R0,#60
        BLO ERR1
        CMPB R0,#106
        BHI ERR1
        CMPB R0,#71
        BLOS NUME
        CMPB R0,#101
        BLO ERR1
        SUB #7,R0
NUME:   BIC #177760,R0
        RTS PC

SINTX1: CMPB R0,#60             ;En esta rutina se comprueba que los
        BLO ERR1                ;caracteres en los comandos indican-
        CMPB R0,#71             ;do el numero de instrucciones a eje-
        BHI ERR1                ;cutar solo sean numeros (0-9) y des-
        BIC #177760,R0          ;pues los convierte de ASCII a binario.
        RTS PC

;-------------------------------------------------------------------------
;       RUTINA QUE ORDENA SE IMPRIMA EL STATUS DEL 8085
;-------------------------------------------------------------------------
ESTADO: .PRINT #REGA            ;En esta rutina se imprime el status del
        MOVB A,SALE             ;microprocesador, para lo cual se van
        JSR PC,SALIDA           ;enviando uno a uno los contenidos de
        .PRINT #REGBC           ;los registros simulados al byte "SALE"
        MOVB B,SALE             ;en donde son convertidos a ASCII para
        JSR PC,SALIDA           ;imprimirlos.
        MOVB C,SALE
        JSR PC,SALIDA
        .PRINT #REGDE
        MOVB D,SALE
        JSR PC,SALIDA
        MOVB E,SALE
        JSR PC,SALIDA
        .PRINT #REGHL
        MOVB H,SALE
        JSR PC,SALIDA
        MOVB L,SALE
        JSR PC,SALIDA
        .PRINT #CMHL
        JSR PC,DIRMEM
        MOVB @R0,SALE
        JSR PC,SALIDA
        .PRINT #STPSW
        MOVB PSW,SALE
        JSR PC,SALIDA
        .PRINT #STPC
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,SALE
        JSR PC,SALIDA
```

```
        SWAB R1
        MOVB R1,SALE
        JSR PC,SALIDA
        ADD #MEMORY,R1
        .PRINT #STSP
        SUB #MEMORY,R2
        SWAB R2
        MOVB R2,SALE
        JSR PC,SALIDA
        SWAB R2
        MOVB R2,SALE
        JSR PC,SALIDA
        ADD #MEMORY,R2
        .PRINT #CSP             ;Imprime el contenido del SP simulado
        INC R2                  ;correspondiente a los dos ultimos by-
        MOVB @R2,SALE           ;tes accesados al stack y tomando como
        JSR PC,SALIDA           ;digitos mas significativos aquellos
        MOVB -(R2),SALE         ;de direccion mayor.
        JSR PC,SALIDA
        .PRINT #SIGREN
        .PRINT #SIGREN
        RTS PC

;-------------------------------------------------------------------------
;       RUTINA QUE IMPRIME EL STATUS DEL 8085
;-------------------------------------------------------------------------
SALIDA: MOVB SALE,R0            ;Convierte el contenido del byte "SALE"
        ASH #-4,R0              ;a codigo ASCII para imprimirlo.
        BICB #360,R0
        ADD #60,R0
        CMPB R0,#71
        BLOS SALID1
        ADD #7,R0
SALID1: .TTYOUT                 ;Sale el primer digito hexadecimal.
        MOVB SALE,R0
        BICB #360,R0
        ADD #60,R0
        CMPB R0,#71
        BLOS SALID2
        ADD #7,R0
SALID2: .TTYOUT                 ;Sale el segundo digito hexadecimal.
        RTS PC

;-------------------------------------------------------------------------
;       RUTINAS DE INSTRUCCIONES CORRESPONDIENTES AL 8085
;-------------------------------------------------------------------------
NOP:    .PRINT #TEX00           ;Imprime el mnemonico de la instruccion.
        JMP FIN
LXIB16: .PRINT #TEX01
        INCB M                  ;Realiza la simulacion de la instruccion:
        JSR PC,OTRO             ;carga el registro par BC con el dato de
        MOV R3,C                ;2 bytes que sigue al opcode. Como uti-
        JSR PC,OTRO             ;liza la subrutina OTRO, inicialmente
        MOV R3,B                ;pone la bandera M en uno y despues la
        CLRB M                  ;regresa a su valor normal de cero.
        JMP META3               ;Regresa por el siguiente opcode.
STAXB:  .PRINT #TEX02
        MOV B,R0                ;Almacena en la memoria el contenido del
        SWAB R0                 ;acumulador. La direccion de la memoria
        ADD C,R0                ;esta dada por el contenido del registro
        ADD #MEMORY,R0          ;par BC.
```

```
        MOVB A,@R0
        JMP META3
INXB:   .PRINT #TEX03
        INCB C                  ;Incrementa en uno el contenido del re-
        BNE SALT1               ;sistro par BC. La unica bandera que es
        INCB B                  ;afectada es "X5".
SALT0:  BNE SALT1
        BISB #40,PSW            ;Coloca la bandera X5 en uno.
        JMP META3
SALT1:  BICB #40,PSW            ;Coloca la bandera X5 en cero.
        JMP META3
INRB:   .PRINT #TEX04
        MOV B,R3
        MOV #1,R5
        JSR PC,CAAUX1
        INCB B                  ;Incrementa en uno el registro B. Afec-
INRB0:  JSR PC,BAND1            ;ta todas las banderas, exceptuando "Cy".
        MOV B,R0
        JSR PC,BAND2
        JMP META3
DCRB:   .PRINT #TEX05
        DECB B                  ;Decrementa en uno al registro B y pone
        JMP INRB0               ;las banderas correspondientes.
MVIB8:  .PRINT #TEX06
        INCB M                  ;Mueve el dato de un byte que sigue al
        JSR PC,OTRO             ;opcode, al registro B.
        MOV R3,B
        CLRB M
        JMP META3
RLC:    .PRINT #TEX07
        ROLB A                  ;El contenido del acumulador es rotado
        JSR PC,BAND3            ;a la izquierda una posicion. El carry
        BCC ACC1                ;y el bit menor son colocados al valor
        BISB #1,A               ;que tenia el bit mayor antes de la ro-
        JMP META3               ;tacion.
ACC1:   BICB #1,A
        JMP META3
DSUB:   .PRINT #TEX08
        MOV H,R0                ;Resta el contenido del registro par -
        SWAB R0                 ;BC al contenido del registro par HL.
        ADD L,R0
        MOV B,R5
        SWAB R5
        ADD C,R5
        SUB R5,R0
        JSR PC,BAND1            ;Afecta todas las banderas.
        JSR PC,BAND3
        JSR PC,ZERO
        TST R0
        BPL NCER
        BISB #200,PSW
        JMP NUNO
NCER:   BICB #200,PSW
NUNO:   MOVB R0,L
        SWAB R0
        MOVB R0,H
        JMP META3
DADB:   .PRINT #TEX09
        MOV H,R0                ;El contenido del registro par BC es
        SWAB R0                 ;sumado al contenido del registro par
        ADD L,R0                ;HL. Solo se afecta la bandera "Cy".
```

```
        MOV B,R5
        SWAB R5
        ADD C,R5
        ADD R5,R0
SUMD:   JSR PC,BAND3            ;Pone el carry al valor adecuado y en-
        MOVB R0,L               ;via el resultado al registro par HL.
        SWAB R0
        MOVB R0,H
        JMP META3
LDAXB:  .PRINT #TEX0A
        MOV B,R0                ;Carga el acumulador con el contenido
        SWAB R0                 ;de la localidad de memoria cuya direc-
        ADD C,R0                ;cion esta dada por el registro par BC.
        ADD #MEMORY,R0
        MOVB @R0,A
        JMP META3
DCXB:   .PRINT #TEX0B
        TSTB C                  ;Decrementa en uno el contenido del re-
        BNE SALTC               ;gistro par BC. Solo afecta a la bandera
        TSTB B                  ;"X5".
        BNE SALTB
        DECB B
        BISB #40,PSW            ;Pone en uno a la bandera "X5".
        JMP SALT2
SALTB:  DECB B
SALTC:  BICB #40,PSW            ;Pone en cero a "X5".
SALT2:  DECB C
        JMP META3
INRC:   .PRINT #TEX0C
        MOV C,R3
        MOV #1,R5
        JSR PC,CAAUX1
        INCB C
INRC0:  JSR PC,BAND1
        MOV C,R0
        JSR PC,BAND2
        JMP META3
DCRC:   .PRINT #TEX0D
        DECB C
        JMP INRC0
MVIC8:  .PRINT #TEX0E
        INCB M
        JSR PC,OTRO
        MOV R3,C
        CLRB M
        JMP META3
RRC:    .PRINT #TEX0F
        RORB A                  ;El contenido del acumulador es rotado
        JSR PC,BAND3            ;a la derecha una posicion. El carry y
        BCC ACC2                ;el bit mayor adquieren el valor que
        BISB #200,A             ;tenia el bit menor antes de la rotacion.
        JMP META3
ACC2:   BICB #200,A             ;Solo es afectada la bandera "Cy".
        JMP META3
ARHL:   .PRINT #TEX10
        MOV H,R0                ;El contenido del registro par HL es ro-
        SWAB R0                 ;tado a la derecha una posicion. Esta -
        ADD L,R0                ;instruccion es equivalente a dividir en-
        ASR R0                  ;tre dos el—contenido del acumulador.
        JMP SUMD
LXID16: .PRINT #TEX11
```

```
        INCB M
        JSR PC,OTRO
        MOV R3,E
        JSR PC,OTRO
        MOV R3,D
        CLRB M
        JMP META3
STAXD:  .PRINT #TEX12
        MOV D,R0
        SWAB R0
        ADD E,R0
        ADD #MEMORY,R0
        MOVB A,@R0
        JMP META3
INXD:   .PRINT #TEX13
        INCB E
        BNE SALT1A
        INCB D
        JMP SALT0
SALT1A: BICB #40,PSW
        JMP META3
INRD:   .PRINT #TEX14
        MOV D,R3
        MOV #1,R5
        JSR PC,CAAUX1
        INCB D
INRD0:  JSR PC,BAND1
        MOV D,R0
        JSR PC,BAND2
        JMP META3
DCRD:   .PRINT #TEX15
        DECB D
        JMP INRD0
MVID8:  .PRINT #TEX16
        INCB M
        JSR PC,OTRO
        MOV R3,D
        CLRB M
        JMP META3
RAL:    .PRINT #TEX17
        JSR PC,CAROT
        ROLB A                  ;El contenido del acumulador es rotado
        JSR PC,BAND3            ;a la izquierda a traves del carry.
        JMP META3
CAROT:  MOVB PSW,R0
        BIC #177776,R0
        BEQ CYCER
        SEC
        RTS PC
CYCER:  CLC
        RTS PC
RDEL:   .PRINT #TEX18
        MOV D,R0                ;El contenido del registro par DE es
        SWAB R0                 ;rotado a la izquierda a traves del ca-
        ADD E,R0                ;rry una posicion.
        JSR PC,CAROT
        ROL R0
        JSR PC,BAND1            ;Afecta dos banderas: "V" y "Cy".
        JSR PC,BAND3
        MOVB R0,E
        SWAB R0
```

```
        MOVB R0,D
        JMP META3
BAND3:  BCC CCER                ;Esta subrutina sirve para, con ayuda
        BISB #1,PSW             ;del carry de la PDP, colocar en su es-
        RTS PC                  ;tado correspondiente a la bandera "Cy".
CCER:   BICB #1,PSW
        RTS PC
DADD:   .PRINT #TEX19
        MOV H,R0
        SWAB R0
        ADD L,R0
        MOV D,R5
        SWAB R5
        ADD E,R5
        ADD R5,R0
        JMP SUMD
LDAXD:  .PRINT #TEX1A
        MOV D,R0
        SWAB R0
        ADD E,R0
        ADD #MEMORY,R0
        MOVB @R0,A
        JMP META3
DCXD:   .PRINT #TEX1B
        TSTB E
        BNE SALTE
        TSTB D
        BNE SALTD
        DECB D
        BISB #40,PSW
        JMP SALT3
SALTD:  DECB D
SALTE:  BICB #40,PSW
SALT3:  DECB E
        JMP META3
INRE:   .PRINT #TEX1C
        MOV E,R3
        MOV #1,R5
        JSR PC,CAAUX1
        INCB E
INRE0:  JSR PC,BAND1
        MOV E,R0
        JSR PC,BAND2
        JMP META3
DCRE:   .PRINT #TEX1D
        DECB E
        JMP INRE0
MVIE8:  .PRINT #TEX1E
        INCB M
        JSR PC,OTRO
        MOV R3,E
        CLRB M
        JMP META3
RAR:    .PRINT #TEX1F
        JSR PC,CAROT
        RORB A                  ;El contenido del acumulador es rotado
        JSR PC,BAND3            ;a la derecha a traves del carry.
        JMP META3
RIM:    .PRINT #TEX20
        JMP META3
LXIH16: .PRINT #TEX21
```

```
        INCB M
        JSR PC,OTRO
        MOV R3,L
        JSR PC,OTRO
        MOV R3,H
        CLRB M
        JMP META3
SHLDADR:.PRINT #TEX22
        INCB M
        JSR PC,OTRO
        MOV R3,R5
        JSR PC,OTRO
        SWAB R3
        ADD R3,R5
        ADD #MEMORY,R5
        MOVB L,(R5)+
        MOVB H,@R5
        CLRB M
        JMP META3
INXH:   .PRINT #TEX23
        INCB L
        BNE SALT1B
        INCB H
        JMP SALTO
SALT1B: BICB #40,PSW
        JMP META3
INRH:   .PRINT #TEX24
        MOV H,R3
        MOV #1,R5
        JSR PC,CAAUX1
        INCB H
INRHO:  JSR PC,BAND1
        MOV H,RO
        JSR PC,BAND2
        JMP META3
DCRH:   .PRINT #TEX25
        DECB H
        JMP INRHO
MVIH8:  .PRINT #TEX26
        INCB M
        JSR PC,OTRO
        MOV R3,H
        CLRB M
        JMP META3
DAA:    .PRINT #TEX27
        MOV A,RO            ;El contenido del acumulador es ajustado
        BICB #360,RO        ;para formar dos digitos BCD de 4 bits,
        CMPB RO,#9.         ;de la manera siguiente:
        BLOS DAA1
DAAO:   MOV #6,RO
        JSR PC,CAAUX
        SWAB A              ;Si el valor de los 4 bits menos signi-
        ADD #3000,A         ;ficativos es mayor que 9 o si la bande-
        JSR PC,BAND1        ;ra AC vale uno, se suma el numero 6 bi-
        JSR PC,BAND3        ;nario al acumulador.
        SWAB A
        MOV A,RO            ;Ahora, si el valor de los 4 bits mas
        JSR PC,BAND2        ;significativos es mayor que 9 o si la
        JMP DAA2            ;bandera Cy vale uno, se suma el numero
DAA1:   MOVB PSW,RO         ;6 binario a estos 4 bits del acumulador.
        BIC #177757,RO      ;Se afectan todas las banderas.
```

```
        BNE DAA0
DAA2:   MOV A,R0
        ASH #-4,R0
        CMPB R0,#9.
        BLOS DAA4
DAA3:   SWAB A
        ADD #60000,A
        JSR PC,BAND1
        JSR PC,BAND3
        SWAB A
        MOV A,R0
        JSR PC,BAND2
        JMP META3
DAA4:   MOVB PSW,R0
        BIC #177776,R0
        BNE DAA3
        JMP META3
LDHI:   .PRINT #TEX28
        INCB M
        JSR PC,OTRO
        CLRB M
        MOV H,R0
        SWAB R0
        ADD L,R0
        ADD R3,R0
        MOVB R0,E
        SWAB R0
        MOVB R0,D
        JMP META3
DADH:   .PRINT #TEX29
        MOV H,R0
        SWAB R0
        ADD L,R0
        ASL R0
        JMP SUMD
LHLDADR:.PRINT #TEX2A
        INCB M
        JSR PC,OTRO
        MOV R3,R5
        JSR PC,OTRO
        SWAB R3
        ADD R3,R5
        ADD #MEMORY,R5
        MOVB (R5)+,L
        MOVB @R5,H
        CLRB M
        JMP META3
DCXH:   .PRINT #TEX2B
        TSTB L
        BNE SALTL
        TSTB H
        BNE SALTH
        DECB H
        BISB #40,PSW
        JMP SALT4
SALTH:  DECB H
SALTL:  BICB #40,PSW
SALT4:  DECB L
        JMP META3
INRL:   .PRINT #TEX2C
        MOV L,R3
```

```
        MOV #1,R5
        JSR PC,CAAUX1
        INCB L
INRL0:  JSR PC,BAND1
        MOV L,R0
        JSR PC,BAND2
        JMP META3
DCRL:   .PRINT #TEX2D
        DECB L
        JMP INRL0
MVIL8:  .PRINT #TEX2E
        INCB M
        JSR PC,OTRO
        MOV R3,L
        CLRB M
        JMP META3
CMA:    .PRINT #TEX2F
        COMB A
        JMP META3
SIM:    .PRINT #TEX30
        JMP META3
LXISP16:.PRINT #TEX31
        INCB M
        JSR PC,OTRO
        MOV R3,R2
        JSR PC,OTRO
        SWAB R3
        ADD R3,R2
        ADD #MEMORY,R2
        CLRB M
        JMP META3
STAADR: .PRINT #TEX32
        INCB M
        JSR PC,OTRO
        MOV R3,R5
        JSR PC,OTRO
        SWAB R3
        ADD R3,R5
        ADD #MEMORY,R5
        MOVB A,@R5
        CLRB M
        JMP META3
INXSP:  .PRINT #TEX33
        INC R2
        JMP SALTO
INRM:   .PRINT #TEX34
        JSR PC,DIRMEM
        MOVB @R0,R3
        MOV #1,R5
        JSR PC,CAAUX1
        INCB @R0
INRM0:  JSR PC,BAND1
        MOVB @R0,R0
        BIC #177400,R0
        JSR PC,BAND2
        JMP META3
DCRM:   .PRINT #TEX35
        JSR PC,DIRMEM
        DECB @R0
        JMP INRM0
MVIM8:  .PRINT #TEX36
```

```
        INCB M
        JSR PC,OTRO
        CLRB M
        JSR PC,DIRMEM
        MOVB R3,@R0
        JMP META3
STC:    .PRINT #TEX37
        BISB #1,PSW
        JMP META3
LDSI:   .PRINT #TEX38
        INCB M
        JSR PC,OTRO
        ADD R2,R3
        MOVB R3,E
        SWAB R3
        MOVB R3,D
        CLRB M
        JMP META3
DADSP:  .PRINT #TEX39
        MOV H,R0
        SWAB R0
        ADD L,R0
        ADD R2,R0
        JMP SUMD
LDAADR: .PRINT #TEX3A
        INCB M
        JSR PC,OTRO
        MOV R3,R5
        JSR PC,OTRO
        SWAB R3
        ADD R3,R5
        ADD #MEMORY,R5
        MOVB @R5,A
        CLRB M
        JMP META3
DCXSP:  .PRINT #TEX3B
        TST R2
        BNE SALT5
        BISB #40,PSW
        JMP SALT6
SALT5:  BICB #40,PSW
SALT6:  DEC R2
        JMP META3
INRA:   .PRINT #TEX3C
        MOV A,R3
        MOV #1,R5
        JSR PC,CAAUX1
        INCB A
INRA0:  JSR PC,BAND1
        MOV A,R0
        JSR PC,BAND2
        JMP META3
DCRA:   .PRINT #TEX3D
        DECB A
        JMP INRA0
MVIA8:  .PRINT #TEX3E
        INCB M
        JSR PC,OTRO
        MOV R3,A
        CLRB M
        JMP META3
```

```
CMC:    .PRINT #TEX3F
        MOV #1,R0
        MOVB PSW,R5
        XOR R0,R5
        MOVB R5,PSW
        JMP META3
MOVBB:  .PRINT #TEX40
        JMP META3
MOVBC:  .PRINT #TEX41
        MOV C,B
        JMP META3
MOVBD:  .PRINT #TEX42
        MOV D,B
        JMP META3
MOVBE:  .PRINT #TEX43
        MOV E,B
        JMP META3
MOVBH:  .PRINT #TEX44
        MOV H,B
        JMP META3
MOVBL:  .PRINT #TEX45
        MOV L,B
        JMP META3
MOVBM:  .PRINT #TEX46
        JSR PC,DIRMEM
        MOVB @R0,B
        JMP META3
MOVBA:  .PRINT #TEX47
        MOV A,B
        JMP META3
MOVCB:  .PRINT #TEX48
        MOV B,C
        JMP META3
MOVCC:  .PRINT #TEX49
        JMP META3
MOVCD:  .PRINT #TEX4A
        MOV D,C
        JMP META3
MOVCE:  .PRINT #TEX4B
        MOV E,C
        JMP META3
MOVCH:  .PRINT #TEX4C
        MOV H,C
        JMP META3
MOVCL:  .PRINT #TEX4D
        MOV L,C
        JMP META3
MOVCM:  .PRINT #TEX4E
        JSR PC,DIRMEM
        MOVB @R0,C
        JMP META3
MOVCA:  .PRINT #TEX4F
        MOV A,C
        JMP META3
MOVDB:  .PRINT #TEX50
        MOV B,D
        JMP META3
MOVDC:  .PRINT #TEX51
        MOV C,D
        JMP META3
MOVDD:  .PRINT #TEX52
```

```
        JMP META3
MOVDE:  .PRINT #TEX53
        MOV E,D
        JMP META3
MOVDH:  .PRINT #TEX54
        MOV H,D
        JMP META3
MOVDL:  .PRINT #TEX55
        MOV L,D
        JMP META3
MOVDM:  .PRINT #TEX56
        JSR PC,DIRMEM
        MOVB @R0,D
        JMP META3
MOVDA:  .PRINT #TEX57
        MOV A,D
        JMP META3
MOVEB:  .PRINT #TEX58
        MOV B,E
        JMP META3
MOVEC:  .PRINT #TEX59
        MOV C,E
        JMP META3
MOVED:  .PRINT #TEX5A
        MOV D,E
        JMP META3
MOVEE:  .PRINT #TEX5B
        JMP META3
MOVEH:  .PRINT #TEX5C
        MOV H,E
        JMP META3
MOVEL:  .PRINT #TEX5D
        MOV L,E
        JMP META3
MOVEM:  .PRINT #TEX5E
        JSR PC,DIRMEM
        MOVB @R0,E
        JMP META3
MOVEA:  .PRINT #TEX5F
        MOV A,E
        JMP META3
MOVHB:  .PRINT #TEX60
        MOV B,H
        JMP META3
MOVHC:  .PRINT #TEX61
        MOV C,H
        JMP META3
MOVHD:  .PRINT #TEX62
        MOV D,H
        JMP META3
MOVHE:  .PRINT #TEX63
        MOV E,H
        JMP META3
MOVHH:  .PRINT #TEX64
        JMP META3
MOVHL:  .PRINT #TEX65
        MOV L,H
        JMP META3
MOVHM:  .PRINT #TEX66
        JSR PC,DIRMEM
        MOVB @R0,H
```

```
        JMP META3
MOVHA:  .PRINT #TEX67
        MOV A,H
        JMP META3
MOVLB:  .PRINT #TEX68
        MOV B,L
        JMP META3
MOVLC:  .PRINT #TEX69
        MOV C,L
        JMP META3
MOVLD:  .PRINT #TEX6A
        MOV D,L
        JMP META3
MOVLE:  .PRINT #TEX6B
        MOV E,L
        JMP META3
MOVLH:  .PRINT #TEX6C
        MOV H,L
        JMP META3
MOVLL:  .PRINT #TEX6D
        JMP META3
MOVLM:  .PRINT #TEX6E
        JSR PC,DIRMEM
        MOVB @R0,L
        JMP META3
MOVLA:  .PRINT #TEX6F
        MOV A,L
        JMP META3
MOVMB:  .PRINT #TEX70
        JSR PC,DIRMEM
        MOVB B,@R0
        JMP META3
MOVMC:  .PRINT #TEX71
        JSR PC,DIRMEM
        MOVB C,@R0
        JMP META3
MOVMD:  .PRINT #TEX72
        JSR PC,DIRMEM
        MOVB D,@R0
        JMP META3
MOVME:  .PRINT #TEX73
        JSR PC,DIRMEM
        MOVB E,@R0
        JMP META3
MOVMH:  .PRINT #TEX74
        JSR PC,DIRMEM
        MOVB H,@R0
        JMP META3
MOVML:  .PRINT #TEX75
        JSR PC,DIRMEM
        MOVB L,@R0
        JMP META3
HLT:    .PRINT #TEX76
        JSR PC,ESTADO
LIMPBU: .TTINR
        BCC LIMPBU
        .TTYIN
        JMP NOIMP
MOVMA:  .PRINT #TEX77
        JSR PC,DIRMEM
        MOVB A,@R0
```

```
        JMP META3
MOVAB:  .PRINT #TEX78
        MOV B,A
        JMP META3
MOVAC:  .PRINT #TEX79
        MOV C,A
        JMP META3
MOVAD:  .PRINT #TEX7A
        MOV D,A
        JMP META3
MOVAE:  .PRINT #TEX7B
        MOV E,A
        JMP META3
MOVAH:  .PRINT #TEX7C
        MOV H,A
        JMP META3
MOVAL:  .PRINT #TEX7D
        MOV L,A
        JMP META3
MOVAM:  .PRINT #TEX7E
        JSR PC,DIRMEM
        MOVB @R0,A
        JMP META3
MOVAA:  .PRINT #TEX7F
        JMP META3
DIRMEM: MOV H,R0
        SWAB R0
        ADD L,R0
        ADD #MEMORY,R0
        RTS PC
ADDB:   .PRINT #TEX80
        MOV B,R0
SUMA:   JSR PC,CAAUX
        SWAB A
        SWAB R0
SUM0:   ADD R0,A
SUM1:   JSR PC,BAND1
        JSR PC,BAND3
        SWAB A
SUM2:   MOV A,R0
        JSR PC,BAND2
        JMP META3
ADDC:   .PRINT #TEX81
        MOV C,R0
        JMP SUMA
ADDD:   .PRINT #TEX82
        MOV D,R0
        JMP SUMA
ADDE:   .PRINT #TEX83
        MOV E,R0
        JMP SUMA
ADDH:   .PRINT #TEX84
        MOV H,R0
        JMP SUMA
ADDL:   .PRINT #TEX85
        MOV L,R0
        JMP SUMA
ADDM:   .PRINT #TEX86
        JSR PC,DIRMEM
        MOVB @R0,R0
        JMP SUMA
```

```
ADDA:   .PRINT #TEX87
        MOV A,R0
        JMP SUMA
ADCB:   .PRINT #TEX88
        MOV B,R5
SUMB:   MOVB PSW,R0
        BIC #177776,R0
        ADD R5,R0
        JSR PC,CAAUX
        SWAB A
        SWAB R0
        BIC #377,R0
        JMP SUM0
ADCC:   .PRINT #TEX89
        MOV C,R5
        JMP SUMB
ADCD:   .PRINT #TEX8A
        MOV D,R5
        JMP SUMB
ADCE:   .PRINT #TEX8B
        MOV E,R5
        JMP SUMB
ADCH:   .PRINT #TEX8C
        MOV H,R5
        JMP SUMB
ADCL:   .PRINT #TEX8D
        MOV L,R5
        JMP SUMB
ADCM:   .PRINT #TEX8E
        JSR PC,DIRMEM
        MOVB @R0,R5
        JMP SUMB
ADCA:   .PRINT #TEX8F
        MOV A,R5
        JMP SUMB
SUBB:   .PRINT #TEX90
        MOV B,R0
RESTA:  SWAB A
        SWAB R0
REST0:  SUB R0,A
        JMP SUM1
SUBC:   .PRINT #TEX91
        MOV C,R0
        JMP RESTA
SUBD:   .PRINT #TEX92
        MOV D,R0
        JMP RESTA
SUBE:   .PRINT #TEX93
        MOV E,R0
        JMP RESTA
SUBH:   .PRINT #TEX94
        MOV H,R0
        JMP RESTA
SUBL:   .PRINT #TEX95
        MOV L,R0
        JMP RESTA
SUBM:   .PRINT #TEX96
        JSR PC,DIRMEM
        MOVB @R0,R0
        BIC #177400,R0
        JMP RESTA
```

```
SUBA:   .PRINT #TEX97
        CLR A
        BICB #203,PSW
        BISB #100,PSW
        JMP META3
SBBB:   .PRINT #TEX98
        MOV B,R0
RESTB:  MOVB PSW,R5
        BIC #177776,R5
        SUB R5,R0
        SWAB A
        SWAB R0
        BIC #377,R0
        JMP REST0
SBBC:   .PRINT #TEX99
        MOV C,R0
        JMP RESTB
SBBD:   .PRINT #TEX9A
        MOV D,R0
        JMP RESTB
SBBE:   .PRINT #TEX9B
        MOV E,R0
        JMP RESTB
SBBH:   .PRINT #TEX9C
        MOV H,R0
        JMP RESTB
SBBL:   .PRINT #TEX9D
        MOV L,R0
        JMP RESTB
SBBM:   .PRINT #TEX9E
        JSR PC,DIRMEM
        MOVB @R0,R0
        JMP RESTB
SBBA:   .PRINT #TEX9F
        MOV A,R0
        JMP RESTB
CAAUX:  MOV R0,R3
        MOV A,R5
CAAUX1: ASH #12.,R3
        ASH #12.,R5
        ADD R3,R5
        BCC CACER
        BISB #20,PSW
        RTS PC
CACER:  BICB #20,PSW
        RTS PC
BAND1:  BVC VCER
        BISB #2,PSW
        RTS PC
VCER:   BICB #2,PSW
        RTS PC
BAND2:  TSTB R0
        BPL SCER
        BISB #200,PSW
        JMP ZERO
SCER:   BICB #200,PSW
ZERO:   TST R0
        BNE ZCER
        BISB #104,PSW
        RTS PC
ZCER:   BICB #100,PSW
```

```
PARID:  MOV #8.,R3
        CLR R5
LAZO:   ROLB R0
        ADC R5
        DEC R3
        BNE LAZO
        BIC #177776,R5
        BEQ PAR
        BICB #4,PSW
        RTS PC
PAR:    BISB #4,PSW
        RTS PC
ANAB:   .PRINT #TEXA0
        MOVB B,R0
ANA0:   COMB R0
        BICB R0,A
ANA1:   BICB #3,PSW
        BISB #20,PSW
        JMP SUM2
ANAC:   .PRINT #TEXA1
        MOVB C,R0
        JMP ANA0
ANAD:   .PRINT #TEXA2
        MOVB D,R0
        JMP ANA0
ANAE:   .PRINT #TEXA3
        MOVB E,R0
        JMP ANA0
ANAH:   .PRINT #TEXA4
        MOVB H,R0
        JMP ANA0
ANAL:   .PRINT #TEXA5
        MOVB L,R0
        JMP ANA0
ANAM:   .PRINT #TEXA6
        JSR PC,DIRMEM
        MOVB @R0,R0
        JMP ANA0
ANAA:   .PRINT #TEXA7
        JMP ANA1
XRAB:   .PRINT #TEXA8
        MOV B,R0
XOR0:   XOR R0,A
XOR1:   BICB #23,PSW
        JMP SUM2
XRAC:   .PRINT #TEXA9
        MOV C,R0
        JMP XOR0
XRAD:   .PRINT #TEXAA
        MOV D,R0
        JMP XOR0
XRAE:   .PRINT #TEXAB
        MOV E,R0
        JMP XOR0
XRAH:   .PRINT #TEXAC
        MOV H,R0
        JMP XOR0
XRAL:   .PRINT #TEXAD
        MOV L,R0
        JMP XOR0
XRAM:   .PRINT #TEXAE
```

```
        JSR PC,DIRMEM
        MOVB @R0,R0
        BIC #177400,R0
        JMP XOR0
XRAA:   .PRINT #TEXAF
        CLR A
        BICB #223,PSW
        BISB #100,PSW
        JMP META3
ORAB:   .PRINT #TEXB0
        MOVB B,R0
ORA0:   BISB R0,A
        JMP XOR1
ORAC:   .PRINT #TEXB1
        MOVB C,R0
        JMP ORA0
ORAD:   .PRINT #TEXB2
        MOVB D,R0
        JMP ORA0
ORAE:   .PRINT #TEXB3
        MOVB E,R0
        JMP ORA0
ORAH:   .PRINT #TEXB4
        MOVB H,R0
        JMP ORA0
ORAL:   .PRINT #TEXB5
        MOVB L,R0
        JMP ORA0
ORAM:   .PRINT #TEXB6
        JSR PC,DIRMEM
        MOVB @R0,R0
        JMP ORA0
ORAA:   .PRINT #TEXB7
        JMP XOR1
CMPB:   .PRINT #TEXB8
        MOV B,R5
CMP0:   CMPB A,R5
        JSR PC,BAND1
        JSR PC,BAND3
        MOV A,R0
        SUB R5,R0
        BIC #177400,R0
        JSR PC,BAND2
        JMP META3
CMPC:   .PRINT #TEXB9
        MOV C,R5
        JMP CMP0
CMPD:   .PRINT #TEXBA
        MOV D,R5
        JMP CMP0
CMPE:   .PRINT #TEXBB
        MOV E,R5
        JMP CMP0
CMPH:   .PRINT #TEXBC
        MOV H,R5
        JMP CMP0
CMPL:   .PRINT #TEXBD
        MOV L,R5
        JMP CMP0
CMPM:   .PRINT #TEXBE
        JSR PC,DIRMEM
```

```
        MOVB @R0,R5
        JMP CMP0
CMPA:   .PRINT #TEXBF
        BICB #203,PSW
        BISB #100,PSW
        JMP META3
RNZ:    .PRINT #TEXC0
        MOVB PSW,R0
        BIC #177677,R0
        BEQ RET
        JMP META3
POPB:   .PRINT #TEXC1
        MOVB (R2)+,C
        MOVB (R2)+,B
        JMP META3
JNZADR: .PRINT #TEXC2
        MOVB PSW,R0
        BIC #177677,R0
        BEQ JMPADR
        ADD #2,R1
        JMP META3
JMPADR: .PRINT #TEXC3
        INCB M
        JSR PC,OTRO
        MOV R3,R5
        JSR PC,OTRO
        SWAB R3
        ADD R3,R5
        MOV R5,R1
        ADD #MEMORY,R1
        CLRB M
        JMP META3
CNZADR: .PRINT #TEXC4
        MOVB PSW,R0
        BIC #177677,R0
        BEQ CALLADR
        ADD #2,R1
        JMP META3
PUSHB:  .PRINT #TEXC5
        MOVB B,-(R2)
        MOVB C,-(R2)
        JMP META3
ADI8:   .PRINT #TEXC6
        INCB M
        JSR PC,OTRO
        CLRB M
        MOV R3,R0
        JMP SUMA
RST0:   .PRINT #TEXC7
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
        MOV #MEMORY,R1
        JMP META3
RZ:     .PRINT #TEXC8
        MOVB PSW,R0
        BIC #177677,R0
        BNE RET
        JMP META3
```

```
RET:    .PRINT #TEXC9
        MOVB (R2)+,R1
        BIC #177400,R1
        MOVB (R2)+,R0
        BIC #177400,R0
        SWAB R0
        ADD R0,R1
        ADD #MEMORY,R1
        JMP META3
JZADR:  .PRINT #TEXCA
        MOVB PSW,R0
        BIC #177677,R0
        BNE JMPADR
        ADD #2,R1
        JMP META3
RSTV:   .PRINT #TEXCB
        MOVB PSW,R0
        BIC #177775,R0
        BNE VONE
        JMP META3
VONE:   SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
        MOV #MEMORY+64.,R1
        JMP META3
CZADR:  .PRINT #TEXCC
        MOVB PSW,R0
        BIC #177677,R0
        BNE CALLADR
        ADD #2,R1
        JMP META3
CALLADR:.PRINT #TEXCD
        INCB M
        JSR PC,OTRO
        MOV R3,R5
        JSR PC,OTRO
        SWAB R3
        ADD R3,R5
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
        MOV R5,R1
        ADD #MEMORY,R1
        CLRB M
        JMP META3
ACI8:   .PRINT #TEXCE
        INCB M
        JSR PC,OTRO
        CLRB M
        MOV R3,R5
        JMP SUMB
RST1:   .PRINT #TEXCF
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
```

```
        MOV #MEMORY+8.,R1
        JMP META3
RNC:    .PRINT #TEXD0
        MOVB PSW,R0
        BIC #177776,R0
        BEQ RET
        JMP META3
POPD:   .PRINT #TEXD1
        MOVB (R2)+,E
        MOVB (R2)+,D
        JMP META3
JNCADR: .PRINT #TEXD2
        MOVB PSW,R0
        BIC #177776,R0
        BEQ JMP1
        ADD #2,R1
        JMP META3
JMP1:   JMP JMPADR
OUT8:   .PRINT #TEXD3
        JMP META3
CNCADR: .PRINT #TEXD4
        MOVB PSW,R0
        BIC #177776,R0
        BEQ CALLADR
        ADD #2,R1
        JMP META3
PUSHD:  .PRINT #TEXD5
        MOVB D,-(R2)
        MOVB E,-(R2)
        JMP META3
SUI8:   .PRINT #TEXD6
        INCB M
        JSR PC,OTRO
        CLRB M
        MOV R3,R0
        JMP RESTA
RST2:   .PRINT #TEXD7
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
        MOV #MEMORY+16.,R1
        JMP META3
RC:     .PRINT #TEXD8
        MOVB PSW,R0
        BIC #177776,R0
        BNE RET1
        JMP META3
RET1:   JMP RET
SHLX:   .PRINT #TEXD9
        MOV D,R0
        SWAB R0
        ADD E,R0
        ADD #MEMORY,R0
        MOVB L,(R0)+
        MOVB H,@R0
        JMP META3
JCADR:  .PRINT #TEXDA
        MOVB PSW,R0
        BIC #177776,R0
```

```
        BNE JMP2
        ADD #2,R1
        JMP META3
JMP2:   JMP JMPADR
IN8:    .PRINT #TEXDB
        JMP META3
CCADR:  .PRINT #TEXDC
        MOVB PSW,R0
        BIC #177776,R0
        BNE CALL1
        ADD #2,R1
        JMP META3
CALL1:  JMP CALLADR
JNX5:   .PRINT #TEXDD
        MOVB PSW,R0
        BIC #177737,R0
        BEQ JMPX1
        ADD #2,R1
        JMP META3
JMPX1:  JMP JMPADR
SBI8:   .PRINT #TEXDE
        INCB M
        JSR PC,OTRO
        CLRB M
        MOV R3,R0
        JMP RESTB
RST3:   .PRINT #TEXDF
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
        MOV #MEMORY+24.,R1
        JMP META3
RPO:    .PRINT #TEXE0
        MOVB PSW,R0
        BIC #177773,R0
        BEQ RET2
        JMP META3
RET2:   JMP RET
POPH:   .PRINT #TEXE1
        MOVB (R2)+,L
        MOVB (R2)+,H
        JMP META3
JPOADR: .PRINT #TEXE2
        MOVB PSW,R0
        BIC #177773,R0
        BEQ JMP3
        ADD #2,R1
        JMP META3
JMP3:   JMP JMPADR
XTHL:   .PRINT #TEXE3
        MOV H,R0
        SWAB R0
        ADD L,R0
        MOVB (R2)+,L
        MOVB @R2,H
        SWAB R0
        MOVB R0,@R2
        SWAB R0
        MOVB R0,-(R2)
```

```
        JMP META3
CPOADR: .PRINT #TEXE4
        MOVB PSW,R0
        BIC #177773,R0
        BEQ CALL2
        ADD #2,R1
        JMP META3
CALL2:  JMP CALLADR
PUSHH:  .PRINT #TEXE5
        MOVB H,-(R2)
        MOVB L,-(R2)
        JMP META3
ANI8:   .PRINT #TEXE6
        INCB M
        JSR PC,OTRO
        CLRB M
        MOVB R3,R0
        JMP ANA0
RST4:   .PRINT #TEXE7
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
        MOV #MEMORY+32.,R1
        JMP META3
RPE:    .PRINT #TEXE8
        MOVB PSW,R0
        BIC #177773,R0
        BNE RET3
        JMP META3
RET3:   JMP RET
PCHL:   .PRINT #TEXE9
        MOV H,R1
        SWAB R1
        ADD L,R1
        ADD #MEMORY,R1
        JMP META3
JPEADR: .PRINT #TEXEA
        MOVB PSW,R0
        BIC #177773,R0
        BNE JMP4
        ADD #2,R1
        JMP META3
JMP4:   JMP JMPADR
XCHG:   .PRINT #TEXEB
        MOV D,R0
        MOV H,D
        MOV R0,H
        MOV E,R0
        MOV L,E
        MOV R0,L
        JMP META3
CPEADR: .PRINT #TEXEC
        MOVB PSW,R0
        BIC #177773,R0
        BNE CALL3
        ADD #2,R1
        JMP META3
CALL3:  JMP CALLADR
LHLX:   .PRINT #TEXED
```

```
        MOV D,R0
        SWAB R0
        ADD E,R0
        ADD #MEMORY,R0
        MOVB (R0)+,L
        MOVB @R0,H
        JMP META3
XRI8:   .PRINT #TEXEE
        INCB M
        JSR PC,OTR0
        CLRB M
        MOV R3,R0
        JMP XOR0
RST5:   .PRINT #TEXEF
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
        MOV #MEMORY+40.,R1
        JMP META3
RP:     .PRINT #TEXF0
        MOVB PSW,R0
        BIC #177577,R0
        BEQ RET4
        JMP META3
RET4:   JMP RET
POPPSW: .PRINT #TEXF1
        MOVB (R2)+,PSW
        MOVB (R2)+,A
        JMP META3
JPADR:  .PRINT #TEXF2
        MOVB PSW,R0
        BIC #177577,R0
        BEQ JMP5
        ADD #2,R1
        JMP META3
JMP5:   JMP JMPADR
DI:     .PRINT #TEXF3
        JMP META3
CPADR:  .PRINT #TEXF4
        MOVB PSW,R0
        BIC #177577,R0
        BEQ CALL4
        ADD #2,R1
        JMP META3
CALL4:  JMP CALLADR
PUSHPSW:.PRINT #TEXF5
        MOVB A,-(R2)
        MOVB PSW,-(R2)
        JMP META3
ORI8:   .PRINT #TEXF6
        INCB M
        JSR PC,OTR0
        CLRB M
        MOVB R3,R0
        JMP ORA0
RST6:   .PRINT #TEXF7
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
```

```
        SWAB R1
        MOVB R1,-(R2)
        MOV #MEMORY+48.,R1
        JMP META3
RM:     .PRINT #TEXF8
        MOVB PSW,R0
        BIC #177577,R0
        BNE RET5
        JMP META3
RET5:   JMP RET
SPHL:   .PRINT #TEXF9
        MOV H,R2
        SWAB R2
        ADD L,R2
        ADD #MEMORY,R2
        JMP META3
JMADR:  .PRINT #TEXFA
        MOVB PSW,R0
        BIC #177577,R0
        BNE JMP6
        ADD #2,R1
        JMP META3
JMP6:   JMP JMPADR
EI:     .PRINT #TEXFB
        JMP META3
CMADR:  .PRINT #TEXFC
        MOVB PSW,R0
        BIC #177577,R0
        BNE CALL5
        ADD #2,R1
        JMP META3
CALL5:  JMP CALLADR
JX5:    .PRINT #TEXFD
        MOVB PSW,R0
        BIC #177737,R0
        BNE JMPX2
        ADD #2,R1
        JMP META3
JMPX2:  JMP JMPADR
CPI8:   .PRINT #TEXFE
        INCB M
        JSR PC,OTRO
        CLRB M
        MOV R3,R5
        JMP CMP0
RST7:   .PRINT #TEXFF
        SUB #MEMORY,R1
        SWAB R1
        MOVB R1,-(R2)
        SWAB R1
        MOVB R1,-(R2)
        MOV #MEMORY+56.,R1
        JMP META3

;-------------------------------------------------------------------------
;       TABLA PARA DEFINIR DIRECCIONES DE INSTRUCCIONES
;-------------------------------------------------------------------------
        .NLIST BIN
TABLA:  .WORD NOP,LXIB16,STAXB,INXB,INRB,DCRB,MVIB8,RLC
        .WORD DSUB,DADB,LDAXB,DCXB,INRC,DCRC,MVIC8,RRC
        .WORD ARHL,LXID16,STAXD,INXD,INRD,DCRD,MVID8,RAL
```

```
        .WORD RDEL,DADD,LDAXD,DCXD,INRE,DCRE,MVIE8,RAR
        .WORD RIM,LXIH16,SHLDADR,INXH,INRH,DCRH,MVIH8,DAA
        .WORD LDHI,DADH,LHLDADR,DCXH,INRL,DCRL,MVIL8,CMA
        .WORD SIM,LXISP16,STAADR,INXSP,INRM,DCRM,MVIM8,STC
        .WORD LDSI,DADSP,LDAADR,DCXSP,INRA,DCRA,MVIA8,CMC
        .WORD MOVBB,MOVBC,MOVBD,MOVBE,MOVBH,MOVBL,MOVBM,MOVBA
        .WORD MOVCB,MOVCC,MOVCD,MOVCE,MOVCH,MOVCL,MOVCM,MOVCA
        .WORD MOVDB,MOVDC,MOVDD,MOVDE,MOVDH,MOVDL,MOVDM,MOVDA
        .WORD MOVEB,MOVEC,MOVED,MOVEE,MOVEH,MOVEL,MOVEM,MOVEA
        .WORD MOVHB,MOVHC,MOVHD,MOVHE,MOVHH,MOVHL,MOVHM,MOVHA
        .WORD MOVLB,MOVLC,MOVLD,MOVLE,MOVLH,MOVLL,MOVLM,MOVLA
        .WORD MOVMB,MOVMC,MOVMD,MOVME,MOVMH,MOVML,HLT,MOVMA
        .WORD MOVAB,MOVAC,MOVAD,MOVAE,MOVAH,MOVAL,MOVAM,MOVAA
        .WORD ADDB,ADDC,ADDD,ADDE,ADDH,ADDL,ADDM,ADDA
        .WORD ADCB,ADCC,ADCD,ADCE,ADCH,ADCL,ADCM,ADCA
        .WORD SUBB,SUBC,SUBD,SUBE,SUBH,SUBL,SUBM,SUBA
        .WORD SBBB,SBBC,SBBD,SBBE,SBBH,SBBL,SBBM,SBBA
        .WORD ANAB,ANAC,ANAD,ANAE,ANAH,ANAL,ANAM,ANAA
        .WORD XRAB,XRAC,XRAD,XRAE,XRAH,XRAL,XRAM,XRAA
        .WORD ORAB,ORAC,ORAD,ORAE,ORAH,ORAL,ORAM,ORAA
        .WORD CMPB,CMPC,CMPD,CMPE,CMPH,CMPL,CMPM,CMPA
        .WORD RNZ,POPB,JNZADR,JMPADR,CNZADR,PUSHB,ADI8,RST0
        .WORD RZ,RET,JZADR,RSTV,CZADR,CALLADR,ACI8,RST1
        .WORD RNC,POPD,JNCADR,OUT8,CNCADR,PUSHD,SUI8,RST2
        .WORD RC,SHLX,JCADR,IN8,CCADR,JNX5,SBI8,RST3
        .WORD RPO,POPH,JPOADR,XTHL,CPOADR,PUSHH,ANI8,RST4
        .WORD RPE,PCHL,JPEADR,XCHG,CPEADR,LHLX,XRI8,RST5
        .WORD RP,POPPSW,JPADR,DI,CPADR,PUSHPSW,ORI8,RST6
        .WORD RM,SPHL,JMADR,EI,CMADR,JX5,CPI8,RST7

;-------------------------------------------------------------------------
;       DEFINICION DE PARAMETROS
;-------------------------------------------------------------------------
HUECO:  .BLKW 256.
DISCO:  .RAD50 /DK /
AREA:   .BLKW 8.
FILINP: .RAD50 /DK FERNA OBJ/
A:      .WORD 0
B:      .WORD 0
C:      .WORD 0
D:      .WORD 0
E:      .WORD 0
H:      .WORD 0
L:      .WORD 0
M:      .BYTE 0
PSW:    .BYTE 0
SALE:   .BYTE 0
MODO:   .BYTE 0
BUFFER: .BLKW 270.
MEMORY: .BLKW 2048.

;-------------------------------------------------------------------------
;                           CONJUNTO DE TEXTOS
;-------------------------------------------------------------------------
        .NLIST BIN
EFETCH: .ASCIZ /ERROR EN EL FETCH/
ELOOK0: .ASCIZ /ERROR EN EL LOOKUP, CANAL ACTIVO/
ELOOK1: .ASCIZ /ERROR EN EL LOOKUP, ARCHIVO NO ENCONTRADO/
ERREA0: .ASCIZ /ERROR EN EL READW, SE LEYO DESPUES DE EOF/
ERREA1: .ASCIZ /ERROR EN EL READW, DE HARDWARE/
ERREA2: .ASCIZ /ERROR EN EL READW, CANAL NO ABIERTO/
```

```
TEXERR: .ASCIZ /CARACTERES NO VALIDOS, DAME OTROS/
CINV:   .ASCIZ /CARACTER INVALIDO EN EL OPCODE/
ACABO:  .ASCIZ /FIN DEL PROGRAMA SIMULADOR/
ENCAB1: .ASCIZ /XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX/
ENCAB2: .ASCIZ /     PROGRAMA SIMULADOR DEL MICROPROCESADOR 8085/
ENCAB3: .ASCIZ /          AUTOR: J.FERNANDO GARCIA N. CANO/
TEXTO1: .ASCIZ /ESPERO EL NO. DE INSTRUCCIONES A EJECUTAR, EL MODO Y/
TEXTO2: .ASCIZ /LA DIRECCION DE INICIO DEL PROGRAMA./
STAINI: .ASCIZ /ESTATUS INICIAL:/
STATUS: .ASCIZ /ESTATUS FINAL:/
REGA:   .ASCII /A=/
        .BYTE 200
REGBC:  .ASCII / BC=/
        .BYTE 200
REGDE:  .ASCII / DE=/
        .BYTE 200
REGHL:  .ASCII / HL=/
        .BYTE 200
CMHL:   .ASCII / (HL)=/
        .BYTE 200
STPSW:  .ASCII / PSW=/
        .BYTE 200
STPC:   .ASCII / PC=/
        .BYTE 200
STSP:   .ASCII / SP=/
        .BYTE 200
CSP:    .ASCII / (SP)=/
        .BYTE 200
SIGREN: .ASCIZ / /
TEX00:  .ASCIZ /NOP/
TEX01:  .ASCIZ /LXI B,D16/
TEX02:  .ASCIZ /STAX B/
TEX03:  .ASCIZ /INX B/
TEX04:  .ASCIZ /INR B/
TEX05:  .ASCIZ /DCR B/
TEX06:  .ASCIZ /MVI B,D8/
TEX07:  .ASCIZ /RLC/
TEX08:  .ASCIZ /DSUB/
TEX09:  .ASCIZ /DAD B/
TEX0A:  .ASCIZ /LDAX B/
TEX0B:  .ASCIZ /DCX B/
TEX0C:  .ASCIZ /INR C/
TEX0D:  .ASCIZ /DCR C/
TEX0E:  .ASCIZ /MVI C,D8/
TEX0F:  .ASCIZ /RRC/
TEX10:  .ASCIZ /ARHL/
TEX11:  .ASCIZ /LXI D,D16/
TEX12:  .ASCIZ /STAX D/
TEX13:  .ASCIZ /INX D/
TEX14:  .ASCIZ /INR D/
TEX15:  .ASCIZ /DCR D/
TEX16:  .ASCIZ /MVI D,D8/
TEX17:  .ASCIZ /RAL/
TEX18:  .ASCIZ /RDEL/
TEX19:  .ASCIZ /DAD D/
TEX1A:  .ASCIZ /LDAX D/
TEX1B:  .ASCIZ /DCX D/
TEX1C:  .ASCIZ /INR E/
TEX1D:  .ASCIZ /DCR E/
TEX1E:  .ASCIZ /MVI E,D8/
TEX1F:  .ASCIZ /RAR/
```

```
TEX20:  .ASCIZ /RIM/
TEX21:  .ASCIZ /LXI H,D16/
TEX22:  .ASCIZ /SHLD Adr/
TEX23:  .ASCIZ /INX H/
TEX24:  .ASCIZ /INR H/
TEX25:  .ASCIZ /DCR H/
TEX26:  .ASCIZ /MVI H,D8/
TEX27:  .ASCIZ /DAA/
TEX28:  .ASCIZ /LDHI/
TEX29:  .ASCIZ /DAD H/
TEX2A:  .ASCIZ /LHLD Adr/
TEX2B:  .ASCIZ /DCX H/
TEX2C:  .ASCIZ /INR L/
TEX2D:  .ASCIZ /DCR L/
TEX2E:  .ASCIZ /MVI L,D8/
TEX2F:  .ASCIZ /CMA/
TEX30:  .ASCIZ /SIM/
TEX31:  .ASCIZ /LXI SP,D16/
TEX32:  .ASCIZ /STA Adr/
TEX33:  .ASCIZ /INX SP/
TEX34:  .ASCIZ /INR M/
TEX35:  .ASCIZ /DCR M/
TEX36:  .ASCIZ /MVI M,D8/
TEX37:  .ASCIZ /STC/
TEX38:  .ASCIZ /LDSI/
TEX39:  .ASCIZ /DAD SP/
TEX3A:  .ASCIZ /LDA Adr/
TEX3B:  .ASCIZ /DCX SP/
TEX3C:  .ASCIZ /INR A/
TEX3D:  .ASCIZ /DCR A/
TEX3E:  .ASCIZ /MVI A,D8/
TEX3F:  .ASCIZ /CMC/
TEX40:  .ASCIZ /MOV B,B/
TEX41:  .ASCIZ /MOV B,C/
TEX42:  .ASCIZ /MOV B,D/
TEX43:  .ASCIZ /MOV B,E/
TEX44:  .ASCIZ /MOV B,H/
TEX45:  .ASCIZ /MOV B,L/
TEX46:  .ASCIZ /MOV B,M/
TEX47:  .ASCIZ /MOV B,A/
TEX48:  .ASCIZ /MOV C,B/
TEX49:  .ASCIZ /MOV C,C/
TEX4A:  .ASCIZ /MOV C,D/
TEX4B:  .ASCIZ /MOV C,E/
TEX4C:  .ASCIZ /MOV C,H/
TEX4D:  .ASCIZ /MOV C,L/
TEX4E:  .ASCIZ /MOV C,M/
TEX4F:  .ASCIZ /MOV C,A/
TEX50:  .ASCIZ /MOV D,B/
TEX51:  .ASCIZ /MOV D,C/
TEX52:  .ASCIZ /MOV D,D/
TEX53:  .ASCIZ /MOV D,E/
TEX54:  .ASCIZ /MOV D,H/
TEX55:  .ASCIZ /MOV D,L/
TEX56:  .ASCIZ /MOV D,M/
TEX57:  .ASCIZ /MOV D,A/
TEX58:  .ASCIZ /MOV E,B/
TEX59:  .ASCIZ /MOV E,C/
TEX5A:  .ASCIZ /MOV E,D/
TEX5B:  .ASCIZ /MOV E,E/
TEX5C:  .ASCIZ /MOV E,H/
```

```
TEX5D:  .ASCIZ /MOV E,L/
TEX5E:  .ASCIZ /MOV E,M/
TEX5F:  .ASCIZ /MOV E,A/
TEX60:  .ASCIZ /MOV H,B/
TEX61:  .ASCIZ /MOV H,C/
TEX62:  .ASCIZ /MOV H,D/
TEX63:  .ASCIZ /MOV H,E/
TEX64:  .ASCIZ /MOV H,H/
TEX65:  .ASCIZ /MOV H,L/
TEX66:  .ASCIZ /MOV H,M/
TEX67:  .ASCIZ /MOV H,A/
TEX68:  .ASCIZ /MOV L,B/
TEX69:  .ASCIZ /MOV L,C/
TEX6A:  .ASCIZ /MOV L,D/
TEX6B:  .ASCIZ /MOV L,E/
TEX6C:  .ASCIZ /MOV L,H/
TEX6D:  .ASCIZ /MOV L,L/
TEX6E:  .ASCIZ /MOV L,M/
TEX6F:  .ASCIZ /MOV L,A/
TEX70:  .ASCIZ /MOV M,B/
TEX71:  .ASCIZ /MOV M,C/
TEX72:  .ASCIZ /MOV M,D/
TEX73:  .ASCIZ /MOV M,E/
TEX74:  .ASCIZ /MOV M,H/
TEX75:  .ASCIZ /MOV M,L/
TEX76:  .ASCIZ /HLT/
TEX77:  .ASCIZ /MOV M,A/
TEX78:  .ASCIZ /MOV A,B/
TEX79:  .ASCIZ /MOV A,C/
TEX7A:  .ASCIZ /MOV A,D/
TEX7B:  .ASCIZ /MOV A,E/
TEX7C:  .ASCIZ /MOV A,H/
TEX7D:  .ASCIZ /MOV A,L/
TEX7E:  .ASCIZ /MOV A,M/
TEX7F:  .ASCIZ /MOV A,A/
TEX80:  .ASCIZ /ADD B/
TEX81:  .ASCIZ /ADD C/
TEX82:  .ASCIZ /ADD D/
TEX83:  .ASCIZ /ADD E/
TEX84:  .ASCIZ /ADD H/
TEX85:  .ASCIZ /ADD L/
TEX86:  .ASCIZ /ADD M/
TEX87:  .ASCIZ /ADD A/
TEX88:  .ASCIZ /ADC B/
TEX89:  .ASCIZ /ADC C/
TEX8A:  .ASCIZ /ADC D/
TEX8B:  .ASCIZ /ADC E/
TEX8C:  .ASCIZ /ADC H/
TEX8D:  .ASCIZ /ADC L/
TEX8E:  .ASCIZ /ADC M/
TEX8F:  .ASCIZ /ADC A/
TEX90:  .ASCIZ /SUB B/
TEX91:  .ASCIZ /SUB C/
TEX92:  .ASCIZ /SUB D/
TEX93:  .ASCIZ /SUB E/
TEX94:  .ASCIZ /SUB H/
TEX95:  .ASCIZ /SUB L/
TEX96:  .ASCIZ /SUB M/
TEX97:  .ASCIZ /SUB A/
TEX98:  .ASCIZ /SBB B/
TEX99:  .ASCIZ /SBB C/
```

```
TEX9A:  .ASCIZ /SBB D/
TEX9B:  .ASCIZ /SBB E/
TEX9C:  .ASCIZ /SBB H/
TEX9D:  .ASCIZ /SBB L/
TEX9E:  .ASCIZ /SBB M/
TEX9F:  .ASCIZ /SBB A/
TEXA0:  .ASCIZ /ANA B/
TEXA1:  .ASCIZ /ANA C/
TEXA2:  .ASCIZ /ANA D/
TEXA3:  .ASCIZ /ANA E/
TEXA4:  .ASCIZ /ANA H/
TEXA5:  .ASCIZ /ANA L/
TEXA6:  .ASCIZ /ANA M/
TEXA7:  .ASCIZ /ANA A/
TEXA8:  .ASCIZ /XRA B/
TEXA9:  .ASCIZ /XRA C/
TEXAA:  .ASCIZ /XRA D/
TEXAB:  .ASCIZ /XRA E/
TEXAC:  .ASCIZ /XRA H/
TEXAD:  .ASCIZ /XRA L/
TEXAE:  .ASCIZ /XRA M/
TEXAF:  .ASCIZ /XRA A/
TEXB0:  .ASCIZ /ORA B/
TEXB1:  .ASCIZ /ORA C/
TEXB2:  .ASCIZ /ORA D/
TEXB3:  .ASCIZ /ORA E/
TEXB4:  .ASCIZ /ORA H/
TEXB5:  .ASCIZ /ORA L/
TEXB6:  .ASCIZ /ORA M/
TEXB7:  .ASCIZ /ORA A/
TEXB8:  .ASCIZ /CMP B/
TEXB9:  .ASCIZ /CMP C/
TEXBA:  .ASCIZ /CMP D/
TEXBB:  .ASCIZ /CMP E/
TEXBC:  .ASCIZ /CMP H/
TEXBD:  .ASCIZ /CMP L/
TEXBE:  .ASCIZ /CMP M/
TEXBF:  .ASCIZ /CMP A/
TEXC0:  .ASCIZ /RNZ/
TEXC1:  .ASCIZ /POP B/
TEXC2:  .ASCIZ /JNZ Adr/
TEXC3:  .ASCIZ /JMP Adr/
TEXC4:  .ASCIZ /CNZ Adr/
TEXC5:  .ASCIZ /PUSH B/
TEXC6:  .ASCIZ /ADI D8/
TEXC7:  .ASCIZ /RST 0/
TEXC8:  .ASCIZ /RZ/
TEXC9:  .ASCIZ /RET/
TEXCA:  .ASCIZ /JZ Adr/
TEXCB:  .ASCIZ /RSTV/
TEXCC:  .ASCIZ /CZ Adr/
TEXCD:  .ASCIZ /CALL Adr/
TEXCE:  .ASCIZ /ACI D8/
TEXCF:  .ASCIZ /RST 1/
TEXD0:  .ASCIZ /RNC/
TEXD1:  .ASCIZ /POP D/
TEXD2:  .ASCIZ /JNC Adr/
TEXD3:  .ASCIZ /OUT D8/
TEXD4:  .ASCIZ /CNC Adr/
TEXD5:  .ASCIZ /PUSH D/
TEXD6:  .ASCIZ /SUI D8/
```

```
TEXD7:  .ASCIZ /RST 2/
TEXD8:  .ASCIZ /RC/
TEXD9:  .ASCIZ /SHLX/
TEXDA:  .ASCIZ /JC Adr/
TEXDB:  .ASCIZ /IN D8/
TEXDC:  .ASCIZ /CC Adr/
TEXDD:  .ASCIZ /JNX5/
TEXDE:  .ASCIZ /SBI D8/
TEXDF:  .ASCIZ /RST 3/
TEXE0:  .ASCIZ /RPO/
TEXE1:  .ASCIZ /POP H/
TEXE2:  .ASCIZ /JPO Adr/
TEXE3:  .ASCIZ /XTHL/
TEXE4:  .ASCIZ /CPO Adr/
TEXE5:  .ASCIZ /PUSH H/
TEXE6:  .ASCIZ /ANI D8/
TEXE7:  .ASCIZ /RST 4/
TEXE8:  .ASCIZ /RPE/
TEXE9:  .ASCIZ /PCHL/
TEXEA:  .ASCIZ /JPE Adr/
TEXEB:  .ASCIZ /XCHG/
TEXEC:  .ASCIZ /CPE Adr/
TEXED:  .ASCIZ /LHLX/
TEXEE:  .ASCIZ /XRI D8/
TEXEF:  .ASCIZ /RST 5/
TEXF0:  .ASCIZ /RP/
TEXF1:  .ASCIZ /POP PSW/
TEXF2:  .ASCIZ /JP Adr/
TEXF3:  .ASCIZ /DI/
TEXF4:  .ASCIZ /CP Adr/
TEXF5:  .ASCIZ /PUSH PSW/
TEXF6:  .ASCIZ /ORI D8/
TEXF7:  .ASCIZ /RST 6/
TEXF8:  .ASCIZ /RM/
TEXF9:  .ASCIZ /SPHL/
TEXFA:  .ASCIZ /JM Adr/
TEXFB:  .ASCIZ /EI/
TEXFC:  .ASCIZ /CM Adr/
TEXFD:  .ASCIZ /JX5/
TEXFE:  .ASCIZ /CPI D8/
TEXFF:  .ASCIZ /RST 7/

;-----------------------------------------------------------------------------

        .END META1


;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
$
EJEMPLO 1.
PROGRAMA PARA ORDENAR DE MENOR A MAYOR (ORDEN ASCENDENTE) 4 NUMEROS
BINARIOS DE 8 BITS ALMACENADOS EN LA MEMORIA DE 0050 A 0053. EL RE-
SULTADO ES ALMACENADO EN LA MEMORIA A PARTIR DE LA DIRECCION 0000 Y
EN LOS REGISTROS B, C, D Y E. NO SE MODIFICAN LOS CONTENIDOS INICIA-
LES DE LAS LOCALIDADES 0050 A 0053.
*


 ERRK0:FERNA.OBJ$R$/L$$
215300360120360220360320360446 2C4E2C562C5E78B9DA1C01414F79
BADA2901CA2A014A57C315017ABBDA3601CA3601535FC31C016C702C71
2C722C737600/
*


 RUN RK1:JFGNC.333
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

     PROGRAMA SIMULADOR DEL MICROPROCESADOR 8085

          AUTOR: J.FERNANDO GARCIA N. CANO

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

ESPERO EL NO. DE INSTRUCCIONES A EJECUTAR, EL MODO Y
LA DIRECCION DE INICIO DEL PROGRAMA.
256/0/0100
ESTATUS INICIAL:
A=00 BC=0000 DE=0000 HL=0000 (HL)=22 PSW=00 PC=0100 SP=1000 (SP)=5245

LXI H,D16
A=00 BC=0000 DE=0000 HL=0053 (HL)=FA PSW=00 PC=0103 SP=1000 (SP)=5245

MVI M,D8
A=00 BC=0000 DE=0000 HL=0053 (HL)=01 PSW=00 PC=0105 SP=1000 (SP)=5245

DCR L
A=00 BC=0000 DE=0000 HL=0052 (HL)=8A PSW=00 PC=0106 SP=1000 (SP)=5245

MVI M,D8
A=00 BC=0000 DE=0000 HL=0052 (HL)=02 PSW=00 PC=0108 SP=1000 (SP)=5245

DCR L
A=00 BC=0000 DE=0000 HL=0051 (HL)=B7 PSW=00 PC=0109 SP=1000 (SP)=5245

MVI M,D8
A=00 BC=0000 DE=0000 HL=0051 (HL)=03 PSW=00 PC=010B SP=1000 (SP)=5245

DCR L
A=00 BC=0000 DE=0000 HL=0050 (HL)=E9 PSW=04 PC=010C SP=1000 (SP)=5245

MVI M,D8
A=00 BC=0000 DE=0000 HL=0050 (HL)=04 PSW=04 PC=010E SP=1000 (SP)=5245

MOV B,M
A=00 BC=0400 DE=0000 HL=0050 (HL)=04 PSW=04 PC=010F SP=1000 (SP)=5245

INR L
A=00 BC=0400 DE=0000 HL=0051 (HL)=03 PSW=00 PC=0110 SP=1000 (SP)=5245

MOV C,M
```

```
INR L
A=00 BC=0403 DE=0000 HL=0052 (HL)=02 PSW=00 PC=0112 SP=1000 (SP)=5245

MOV D,M
A=00 BC=0403 DE=0200 HL=0052 (HL)=02 PSW=00 PC=0113 SP=1000 (SP)=5245

INR L
A=00 BC=0403 DE=0200 HL=0053 (HL)=01 PSW=04 PC=0114 SP=1000 (SP)=5245

MOV E,M
A=00 BC=0403 DE=0201 HL=0053 (HL)=01 PSW=04 PC=0115 SP=1000 (SP)=5245

MOV A,B
A=04 BC=0403 DE=0201 HL=0053 (HL)=01 PSW=04 PC=0116 SP=1000 (SP)=5245

CMP C
A=04 BC=0403 DE=0201 HL=0053 (HL)=01 PSW=00 PC=0117 SP=1000 (SP)=5245

JC ADR
A=04 BC=0403 DE=0201 HL=0053 (HL)=01 PSW=00 PC=011A SP=1000 (SP)=5245

MOV B,C
A=04 BC=0303 DE=0201 HL=0053 (HL)=01 PSW=00 PC=011B SP=1000 (SP)=5245

MOV C,A
A=04 BC=0304 DE=0201 HL=0053 (HL)=01 PSW=00 PC=011C SP=1000 (SP)=5245

MOV A,C
A=04 BC=0304 DE=0201 HL=0053 (HL)=01 PSW=00 PC=011D SP=1000 (SP)=5245

CMP D
A=04 BC=0304 DE=0201 HL=0053 (HL)=01 PSW=00 PC=011E SP=1000 (SP)=5245

JC ADR
A=04 BC=0304 DE=0201 HL=0053 (HL)=01 PSW=00 PC=0121 SP=1000 (SP)=5245

JZ ADR
A=04 BC=0304 DE=0201 HL=0053 (HL)=01 PSW=00 PC=0124 SP=1000 (SP)=5245

MOV C,D
A=04 BC=0302 DE=0201 HL=0053 (HL)=01 PSW=00 PC=0125 SP=1000 (SP)=5245

MOV D,A
A=04 BC=0302 DE=0401 HL=0053 (HL)=01 PSW=00 PC=0126 SP=1000 (SP)=5245

JMP ADR
A=04 BC=0302 DE=0401 HL=0053 (HL)=01 PSW=00 PC=0115 SP=1000 (SP)=5245

MOV A,B
A=03 BC=0302 DE=0401 HL=0053 (HL)=01 PSW=00 PC=0116 SP=1000 (SP)=5245

CMP C
A=03 BC=0302 DE=0401 HL=0053 (HL)=01 PSW=00 PC=0117 SP=1000 (SP)=5245

JC ADR
A=03 BC=0302 DE=0401 HL=0053 (HL)=01 PSW=00 PC=011A SP=1000 (SP)=5245

MOV B,C
A=03 BC=0202 DE=0401 HL=0053 (HL)=01 PSW=00 PC=011B SP=1000 (SP)=5245

MOV C,A
A=03 BC=0203 DE=0401 HL=0053 (HL)=01 PSW=00 PC=011C SP=1000 (SP)=5245

MOV A,C
```

```
CMP D
A=03 BC=0203 DE=0401 HL=0053 (HL)=01 PSW=85 PC=011E SP=1000 (SP)=5245

JC ADR
JMP ADR
A=03 BC=0203 DE=0401 HL=0053 (HL)=01 PSW=85 PC=0129 SP=1000 (SP)=5245

MOV A,D
A=04 BC=0203 DE=0401 HL=0053 (HL)=01 PSW=85 PC=012A SP=1000 (SP)=5245

CMP E
A=04 BC=0203 DE=0401 HL=0053 (HL)=01 PSW=04 PC=012B SP=1000 (SP)=5245

JC ADR
A=04 BC=0203 DE=0401 HL=0053 (HL)=01 PSW=04 PC=012E SP=1000 (SP)=5245

JZ ADR
A=04 BC=0203 DE=0401 HL=0053 (HL)=01 PSW=04 PC=0131 SP=1000 (SP)=5245

MOV D,E
A=04 BC=0203 DE=0101 HL=0053 (HL)=01 PSW=04 PC=0132 SP=1000 (SP)=5245

MOV E,A
A=04 BC=0203 DE=0104 HL=0053 (HL)=01 PSW=04 PC=0133 SP=1000 (SP)=5245

JMP ADR
A=04 BC=0203 DE=0104 HL=0053 (HL)=01 PSW=04 PC=011C SP=1000 (SP)=5245

MOV A,C
A=03 BC=0203 DE=0104 HL=0053 (HL)=01 PSW=04 PC=011D SP=1000 (SP)=5245

CMP D
A=03 BC=0203 DE=0104 HL=0053 (HL)=01 PSW=00 PC=011E SP=1000 (SP)=5245

JC ADR
A=03 BC=0203 DE=0104 HL=0053 (HL)=01 PSW=00 PC=0121 SP=1000 (SP)=5245

JZ ADR
A=03 BC=0203 DE=0104 HL=0053 (HL)=01 PSW=00 PC=0124 SP=1000 (SP)=5245

MOV C,D
A=03 BC=0201 DE=0104 HL=0053 (HL)=01 PSW=00 PC=0125 SP=1000 (SP)=5245

MOV D,A
A=03 BC=0201 DE=0304 HL=0053 (HL)=01 PSW=00 PC=0126 SP=1000 (SP)=5245

JMP ADR
A=03 BC=0201 DE=0304 HL=0053 (HL)=01 PSW=00 PC=0115 SP=1000 (SP)=5245

MOV A,B
A=02 BC=0201 DE=0304 HL=0053 (HL)=01 PSW=00 PC=0116 SP=1000 (SP)=5245

CMP C
A=02 BC=0201 DE=0304 HL=0053 (HL)=01 PSW=00 PC=0117 SP=1000 (SP)=5245

JC ADR
A=02 BC=0201 DE=0304 HL=0053 (HL)=01 PSW=00 PC=011A SP=1000 (SP)=5245

MOV B,C
A=02 BC=0101 DE=0304 HL=0053 (HL)=01 PSW=00 PC=011B SP=1000 (SP)=5245

MOV C,A
A=02 BC=0102 DE=0304 HL=0053 (HL)=01 PSW=00 PC=011C SP=1000 (SP)=5245
```

```
CMP D
A=02 BC=0102 DE=0304 HL=0053 (HL)=01 PSW=85 PC=011E SP-1000 (SP)-5245

JC ADR
JMP ADR
A=02 BC=0102 DE=0304 HL=0053 (HL)=01 PSW=85 PC=0129 SP-1000 (SP)-5245

MOV A,D
A=03 BC=0102 DE=0304 HL=0053 (HL)=01 PSW=85 PC=012A SP-1000 (SP)-5245

CMP E
A=03 BC=0102 DE=0304 HL=0053 (HL)-01 PSW-85 PC-012B SP-1000 (SP)-5245

JC ADR
JMP ADR
A=03 BC=0102 DE=0304 HL=0053 (HL)=01 PSW-85 PC-0136 SP-1000 (SP)-5245

MOV L,H
A=03 BC=0102 DE=0304 HL=0000 (HL)=22 PSW=85 PC=0137 SP=1000 (SP)=5245

MOV M,B
A=03 BC=0102 DE=0304 HL=0000 (HL)=01 PSW=85 PC=0138 SP-1000 (SP)=5245

INR L
A=03 BC=0102 DE=0304 HL=0001 (HL)=5C PSW-01 PC-0139 SP=1000 (SP)=5245

MOV M,C
A=03 BC=0102 DE=0304 HL-0001 (HL)-02 PSW-01 PC-013A SP-1000 (SP)-5245

INR L
A=03 BC=0102 DE=0304 HL=0002 (HL)-1B PSW=01 PC-013B SP-1000 (SP)=5245

MOV M,D
A=03 BC=0102 DE=0304 HL=0002 (HL)=03 PSW=01 PC=013C SP-1000 (SP)=5245

INR L
A=03 BC=0102 DE=0304 HL=0003 (HL)=13 PSW=05 PC=013D SP-1000 (SP)-5245

MOV M,E
A=03 BC=0102 DE=0304 HL=0003 (HL)-04 PSW=05 PC=013E SP-1000 (SP)-5245

HLT
A-03 BC-0102 DE=0304 HL-0003 (HL)=04 PSW=05 PC=013F SP-1000 (SP)-5245


NOP
ESTATUS FINAL:
A=03 BC=0102 DE-0304 HL=0003 (HL)-04 PSW=05 PC=0140 SP=1000 (SP)=5245

FIN DEL PROGRAMA SIMULADOR

..


  RUN RK1:JFGNC.333
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

     PROGRAMA SIMULADOR DEL MICROPROCESADOR 8085

          AUTOR: J.FERNANDO GARCIA N. CANO

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
257/0/0100
CARACTERES NO VALIDOS, DAME OTROS
256/2/0100
CARACTERES NO VALIDOS, DAME OTROS
256/0/0F01
CARACTERES NO VALIDOS, DAME OTROS
256/1/0100
ESTATUS INICIAL:
A=00 BC=0000 DE=0000 HL=0000 (HL)=01 PSW=00 PC=0100 SP=1000 (SP)=0245

LXI H,D16
MVI M,D8
DCR L
MVI M,D8
DCR L
MVI M,D8
DCR L
MVI M,D8
MOV B,M
INR L
MOV C,M
INR L
MOV D,M
INR L
MOV E,M
MOV A,B
CMP C
JC ADR
MOV B,C
MOV C,A
MOV A,C
CMP D
JC ADR
JZ ADR
MOV C,D
MOV D,A
JMP ADR
MOV A,B
CMP C
JC ADR
MOV B,C
MOV C,A
MOV A,C
CMP D
JC ADR
JMP ADR
MOV A,D
CMP E
JC ADR
JZ ADR
MOV D,E
MOV E,A
JMP ADR
MOV A,C
CMP D
JC ADR
JZ ADR
MOV C,D
MOV D,A
JMP ADR
MOV A,B
CMP C
JC ADR
MOV B,C
```

```
CMP D
JC ADR
JMP ADR
MOV A,D
CMP E
JC ADR
JMP ADR
MOV L,H
MOV M,B
INR L
MOV M,C
INR L
MOV M,D
INR L
MOV M,E
HLT
A=03 BC=0102 DE=0304 HL=0003 (HL)=04 PSW=05 PC=013F SP=1000 (SP)=5245

NOP
ESTATUS FINAL:
A=03 BC=0102 DE=0304 HL=0003 (HL)=04 PSW=05 PC=0140 SP=1000 (SP)=5245

FIN DEL PROGRAMA SIMULADOR
```

```
$
EJEMPLO 2.
PROGRAMA PARA SUMAR 2 NUMEROS BCD DE 4 DIGITOS CADA UNO ALMACENADOS
A PARTIR DE 0000 Y DEJAR EL RESULTADO EN ORDEN ASCENDENTE DE VALOR
A PARTIR DE 0010. EL RESULTADO TAMBIEN APARECE EN LOS REGISTROS A,
B Y C.
*


 ERRKO:FERNA.OBJ$R$/L$$
21030036902D36252D36102D36967E2E0286273210003A01002C8E2732
11003E00CE003212002E104E2C467600/
*


 RUN RK1:JFGNC.333
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

     PROGRAMA SIMULADOR DEL MICROPROCESADOR 8085

          AUTOR: J.FERNANDO GARCIA N. CANO

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

ESPERO EL NO. DE INSTRUCCIONES A EJECUTAR, EL MODO Y
LA DIRECCION DE INICIO DEL PROGRAMA.
256/0/0F00
ESTATUS INICIAL:
A=00 BC=0000 DE=0000 HL=0000 (HL)=22 PSW=00 PC=0F00 SP=1000 (SP)=5245

LXI H,D16
A=00 BC=0000 DE=0000 HL=0003 (HL)=13 PSW=00 PC=0F03 SP=1000 (SP)=5245

MVI M,D8
A=00 BC=0000 DE=0000 HL=0003 (HL)=90 PSW=00 PC=0F05 SP=1000 (SP)=5245

DCR L
A=00 BC=0000 DE=0000 HL=0002 (HL)=1B PSW=00 PC=0F06 SP=1000 (SP)=5245

MVI M,D8
A=00 BC=0000 DE=0000 HL=0002 (HL)=25 PSW=00 PC=0F08 SP=1000 (SP)=5245

DCR L
A=00 BC=0000 DE=0000 HL=0001 (HL)=5C PSW=00 PC=0F09 SP=1000 (SP)=5245

MVI M,D8
A=00 BC=0000 DE=0000 HL=0001 (HL)=10 PSW=00 PC=0F0B SP=1000 (SP)=5245

DCR L
A=00 BC=0000 DE=0000 HL=0000 (HL)=22 PSW=44 PC=0F0C SP=1000 (SP)=5245

MVI M,D8
A=00 BC=0000 DE=0000 HL=0000 (HL)=96 PSW=44 PC=0F0E SP=1000 (SP)=5245

MOV A,M
A=96 BC=0000 DE=0000 HL=0000 (HL)=96 PSW=44 PC=0F0F SP=1000 (SP)=5245

MVI L,D8
A=96 BC=0000 DE=0000 HL=0002 (HL)=25 PSW=44 PC=0F11 SP=1000 (SP)=5245

ADD M
A=BB BC=0000 DE=0000 HL=0002 (HL)=25 PSW=84 PC=0F12 SP=1000 (SP)=5245
```

```
STA ADR
A=21 BC=0000 DE=0000 HL=0002 (HL)=25 PSW=15 PC=0F16 SP=1000 (SP)=5245

LDA ADR
A=10 BC=0000 DE=0000 HL=0002 (HL)=25 PSW=15 PC=0F19 SP=1000 (SP)=5245

INR L
A=10 BC=0000 DE=0000 HL=0003 (HL)=90 PSW=05 PC=0F1A SP=1000 (SP)=5245

ADC M
A=A1 BC=0000 DE=0000 HL=0003 (HL)=90 PSW=80 PC=0F1B SP=1000 (SP)=5245

DAA
A=01 BC=0000 DE=0000 HL=0003 (HL)=90 PSW=01 PC=0F1C SP=1000 (SP)=5245

STA ADR
A=01 BC=0000 DE=0000 HL=0003 (HL)=90 PSW=01 PC=0F1F SP=1000 (SP)=5245

MVI A,D8
A=00 BC=0000 DE=0000 HL=0003 (HL)=90 PSW=01 PC=0F21 SP=1000 (SP)=5245

ACI D8
A=01 BC=0000 DE=0000 HL=0003 (HL)=90 PSW=00 PC=0F23 SP=1000 (SP)=5245

STA ADR
A=01 BC=0000 DE=0000 HL=0003 (HL)=90 PSW=00 PC=0F26 SP=1000 (SP)=5245

MVI L,D8
A=01 BC=0000 DE=0000 HL=0010 (HL)=21 PSW=00 PC=0F28 SP=1000 (SP)=5245

MOV C,M
A=01 BC=0021 DE=0000 HL=0010 (HL)=21 PSW=00 PC=0F29 SP=1000 (SP)=5245

INR L
A=01 BC=0021 DE=0000 HL=0011 (HL)=01 PSW=04 PC=0F2A SP=1000 (SP)=5245

MOV B,M
A=01 BC=0121 DE=0000 HL=0011 (HL)=01 PSW=04 PC=0F2B SP=1000 (SP)=5245

HLT
A=01 BC=0121 DE=0000 HL=0011 (HL)=01 PSW=04 PC=0F2C SP=1000 (SP)=5245


NOP
ESTATUS FINAL:
A=01 BC=0121 DE=0000 HL=0011 (HL)=01 PSW=04 PC=0F2D SP=1000 (SP)=5245

FIN DEL PROGRAMA SIMULADOR

..
```