UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO-MATEMATICAS

EL PODER DEL MICROPROCESADOR MC 6802

QUE PRESENTA

FRANCISCO GONZALO MORENO RODRIGUEZ

EN OPCION A TITULO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

MONTERREY, N. L. ABRIL DE 1986

Donado por:
Lic. Fernando Fonseca.
NOV. 23 87

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE CIENCIAS FISICO-MATEMATICAS

EL PODER DEL MICROPROCESADOR MC6802

QUE PRESENTA

FRANCISCO GONZALO MORENO RODRIGUEZ

EN OPCION A TITULO DE

LICENCIADO EN CIENCIAS COMPUTACIONALES

MONTERREY, N. L. ABRIL DE 1986

A G R A D E C I M I E N T O S

A mis Padres:

Con cariño y amor para mis padres por el apoyo que me han brindado durante mi trayectoria realizada y mi meta lograda

Sr. Daniel Moreno Ortega

Sra. Guadalupe Rodríguez de Moreno

A mi Abuelo:

Sr. Conrado Moreno Serna

A mis Hermanos:

Gracias por el apoyo y comprensión

Ma. Guadalupe

Elvia Armando Ma. Cristina

J. Graciela Daniel

Miguel Angel

Olga Nelly Elia Marïsela

A mis Maestros:

Por los conocimientos transmitidos durante el transcurso de la Carrera. GRACIAS

Con afecto y Estimación al Maestro y Asesor:

Ing. Aurelio Ramírez Granados

A mis Amigos y Compañeros:

Laura Felipe Hortencia Rogelio

Manzano Jorge S. Jorge R.

Alfredo Luis

Andrés

Julieta Julia

Isidro Reynaldo Ramiro

EL PODER DEL MICROPROCESADOR MC 6802

I N D I C E

Pag.

Pag.

Pag.

1.- INTRODUCCION A LOS MICROPROCESADORES

1.1.- HISTORIA

La historia de la computadora comienza con la invensión - del ABACO (figura 1.1), originario de muchos siglos pasados siendo una forma de computador digital. Este dispositivo de - cálculo puede ser usado para sumas, sustracciones, multiplicaciones y divisiones. Cada columna contiene 2 cuentas arriba - del travesaño y 5 cuentas abajo de él. Cada cuenta arriba del travesaño representa 5 unidades y cada cuenta de abajo representa 1 unidad.

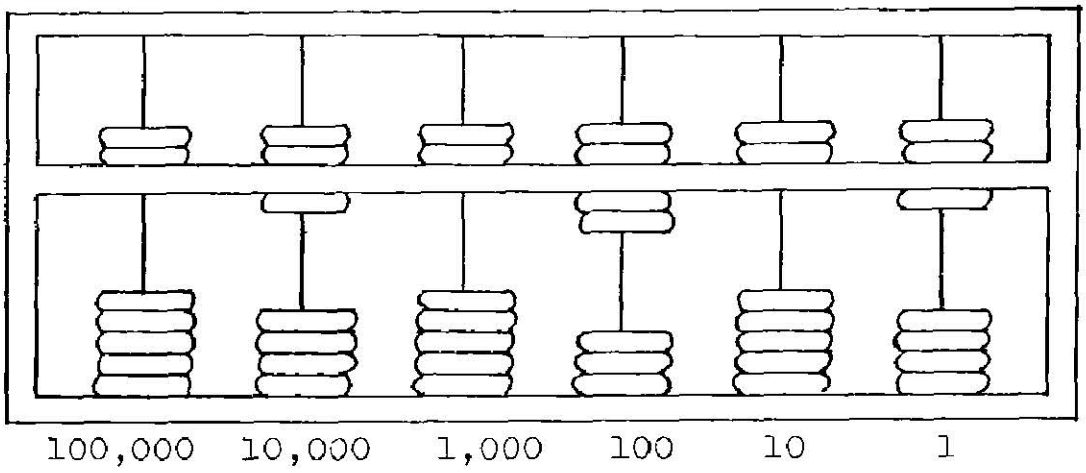


Figura 1.1 El ABACO

Note que el valor de cada columna es como se muestra abajo -- del ABACO. El número representado es el 10,201.
Si queremos ilustrar el número 60,201, una cuenta de arriba - del travesaño en la columna de 10,000 debe levantarse.

La primera máquina mecánica sumadora fue desarrollada en- 1600s. Algunos otros de máquinas sumadoras y varias formas -- rudimentarias de computadores digitales fueron inventadas en- los próximos 350 años. En 1940s el primer computador "electrónico" real fue introducido, para ellos usaron tubos al vacío- (Bulbos). Tales bulbos no fueron realmente la respuesta, -- puesto que, son bastante grandes (1 a 3 pulg.), requieren - grandes cantidades de poder y mucho espacio, generando tremenda cantidad de calor.

Con la invensión del TRANSISTOR, dispositivo semiconduc-- tor, en 1940s y cerca de los 1950s, muchos de los problemas - asociados con bulbos fueron resueltos. Los Transistores son -

pequeños comparados con los bulbos (1/4 pulg.) y requieren - relativamente menos poder. Hacen esto posible para el diseño- y la estructura de computadores digitales los cuales pueden - ser alojados en armarios propios de un cuarto modesto.

En los pasados 1960s y cerca de 1970s, nació el MICROPROCESADOR. Con ésta nueva tecnología, reemplaza hábilmente un - módulo conteniendo de 5 a 10,000 transistores con una pequeña pieza de material semiconductor menor de 1/4 pulg.^{2} ; por -- eso, el término de MICROPROCESADOR.

Aunque los principios básicos de computación digital cambian muy poco este pequeño chip de material semiconductor, seguido referido como un "circuito integrado", abre la puerta - a toda una era de Microeléctricos. Nuevas aplicaciones de computadoras controladas que están más allá de cualquier sueño - de años atrás ahora empieza a emerger.

Las calculadoras electrónicas de retención manual -un artículo de lujo para un promedio de personas en años atrás- están ahora disponibles para cada persona. Los juegos de TV -- electrónica, como ping-pong, son también un producto de Microprocesador.

MPU y MICROPROCESADOR: Estos términos son usados intercambiablemente.

1.2.- MODELO DE COMPUTADORA

Primero desarrollaremos un modelo de computadora para --- ilustrar estas bases: El grabado te sienta en un gran escritorio. La camisa que usted lleva tiene un bolsillo en cada lado. A su izquierda está una pila de tarjetas con cara hacia abajo con una instrucción impresa en cada cara. A su derecha está - un pequeño pizarrón, con una pieza de gis y un borrador. Di-- rectamente enfrente de usted están 2 bandejas, una de entrada y la otra de salida. En la pared hay un gran reloj.

Tenemos ahora creada una situación que es análoga a un -- sistema de Microcomputadora. Usted puede actuar como un MPU.- Puede escoger una tarjeta suelta del bonche de tarjetas cada-

vez que se ejecuta la instrucción impresa en esa cara. El bonche de tarjetas puede hacer referencia a la ROM (Read Only -- Memory), una vez con este contenido no puede cambiarse. Cada- tarjeta en esta ROM representa una instrucción en memoria.

El pizarrón, el gis y el borrador en su derecha pueden -- también hacer referencia a un tipo de memoria. Sin embargo, - en esta memoria podemos escribir y borrar cosas pasadas. Esta memoria hace referencia a la RAM (Random Access Memory), puesto que podemos escribir o cambiar en ella. Esta puede ser usada para almacenar datos (escribir información).

El tiempo (cada minuto) puede ser determinado por el reloj de la pared. Cualquier operación ejecutada por usted(MPU) debe ser hecha en un orden secuencial de tiempo.

Las 2 bandejas enfrente de usted son las bandejas de entrada y salida. La información de usted (MPU) debe llegar --- através de la bandeja de entrada. Así mismo, la salida del -- MPU debe ir através de la bandeja de salida. La bandeja de -- Entrada, en este sistema, está ocupada con tarjetas que contienen números al azar entre 1 y 10.

Los bolsillos en su camisa contienen cada uno una hoja de papel bastante grande para escribir un número y un lápiz con- un borrador. Su bolsillo izquierdo puede hacer referencia a - un acumulador A. Un ACUMULADOR es similar a la RAM excepto -- que puede retener solo un número, mientras que la RAM puede - almacenar varios números. Su bolsillo derecho puede hacer referencia a un acumulador B. Ambos Acumuladores son idénticos- y ejecutan funciones idénticas, esto es, proveen localidades- temporales para el almacenamiento de números.

Las instrucciones en un programa ROM, están dirigidas al- MPU para acumularlas, para la suma de números de la bandeja - de entrada, con 5 números impares mayores que 100 en su pizarrón RAM. Estas instrucciones en las primeras 8 tarjetas leídas son como sigue:

Tarjeta 1.- Lee el número de la bandeja de entrada y escribe- su número en el papel del bolsillo izquierdo.

Tarjeta 2.- Es el número impar? Si es Yes salta las tarjetas- 3, 4 y 5.

Tarjeta 3.- Lee el número de la bandeja de entrada y escribe- su número en el papel del bolsillo derecho.

Tarjeta 4.- Suma los números del bolsillo derecho e izquierdo y pone el resultado en el bolsillo izquierdo.

Tarjeta 5.- Es el número impar? Si es No ir al reverso de la- tarjeta 3.

Tarjeta 6.- Es el número mayor de 100? Si es No, regresar a - la tarjeta 3.

Tarjeta 7.- Escribir el número localizado en el bolsillo iz-- quierdo en el pizarrón.

Tarjeta 8.- Hay 5 números en el pizarrón? Si es No, regresar- a la tarjeta 1.

Justo como una instrucción de su bonche de tarjetas - El- programa en memoria ROM es leído y ejecutado por usted en secuencia (a menos que el MPU ejecute un Branch (salto) para -- alguna otra instrucción) durante un intervalo de tiempo, talcomo lo hace un sistema de microcomputador para ejecutar esas instrucciones. En su sistema análogo, los datos son almacenados en localidades temporales (bolsillos y pizarrón). Un sistema de microcomputador también contiene localidades de almacenamiento temporales, llamados ACUMULADORES y RAM. El sistema de microcomputador también tiene un reloj para controlar - la secuencia de eventos, y las bandejas entrada y salida de - su modelo de sistema son similares en función a los del sistema de microcomputador.

Note que como cada instrucción es leída, debemos tomar -- una decisión lógica basada en los contenidos de la instruc--- ción o la ejecución de algunos cálculos. Esas Acciones son la función del Microprocesador en un Microcomputador. Sin embargo, en el pasado, las instrucciones fueron codificadas de alguna manera. Recordando que las computadoras digitales responden solo a lenguajes binarios (1s y Os). Por lo tanto, cada instrucción en su sistema análogo debe representarse por un - número binario.

Un sistema de MICROCOMPUTADOR consiste de 5 elementos:

- Un Microprocesador (MPU)
- Un Random Access Memory (RAM)
- Un Read-Only Memory (ROM)
- Un Reloj
- Alguna técnica para obtener datos dentro y fuera del sistema (INPUT/OUTPUT)

1.3.- EL MICROPROCESADOR

El Microprocesador es un conjunto de dispositivos electrónicos que pueden realizar las funciones de una computadora -- convencional, es decir entrada, procesamiento, almacenamiento y salida de información. Utilizando componentes electrónicos- constituídos con técnica L.S.I. (Large Scale Integration).

El papel de un chip de microprocesador es análogo a la -- parte que juega en el computador. Este es tan pequeño que puede ser detenido en la mano. Puesto que el MPU debe comunicarse con otros dispositivos en el sistema, debe tener algunos - caminos para decidir en cuales de ellos - RAM, ROM, INPUT, -- OUTPUT - va a ser'direccionado'. Este problema se resuelve -- uniendo varios alambres, llamados Líneas, para cada dispositivo en el sistema. Esas Líneas de Dirección con las cuales se- comunica el MPU son referenciadas como BUS de Dirección (Ductos de dirección). Muchos MPUs contienen líneas de 16 direc-- ciones. Para cargarlas muchas de ellas en un alto voltaje(1s lógicos) y muchos de bajo voltaje(Os lógicos) generan una dirección. El dispositivo en ésta dirección puede comunicarse - entonces con el MPU. Por ejemplo, consideremos 16 líneas que- toma el Ducto de dirección de la figura 1.2

Asumimos que deseamos comunicarnos con el dispositivo en- cual la dirección es 8321_{16}. Su patrón binario es 1000 0011 - 0010 0001. Si cada línea que representa un 1 binario la cargamos de 5 V y cada línea que representa un 0 binario la cargamos con 0 V, solo un dispositivo en la terminación de esos -- alambres puede responder a esa dirección y comunicarse con -- el MPU. Note que las flechas en el Bus de dirección en la fi

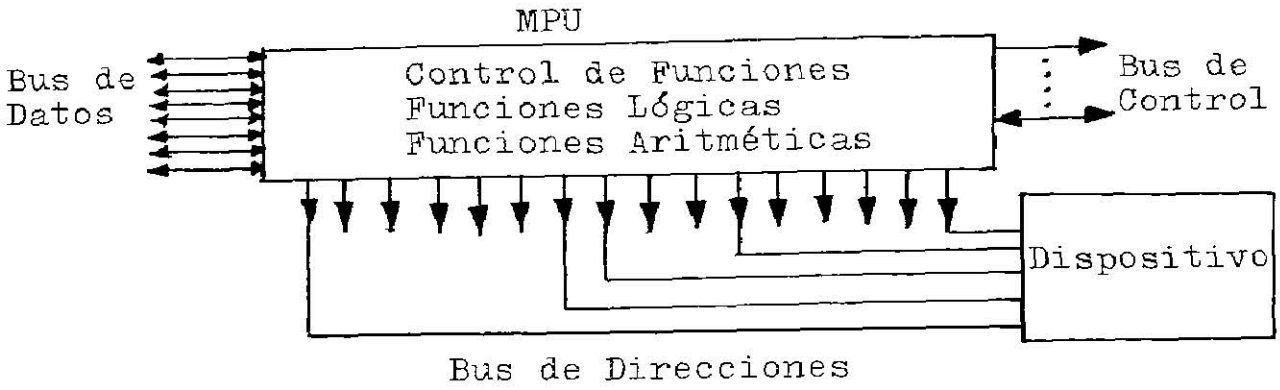


Figura 1.2 El MPU

gura 1.2 todas van dirigidas del MPU en dirección a los demás dispositivos, indicando así que todas las direcciones son generadas por el MPU. Note también que las líneas individuales están numeradas de 0 a 15. Por lo tanto, cada línea tiene un alto ó bajo voltaje representado por 1s ó 0s. Las líneas pueden hacer referencia a Bits un término derivado de la palabra Binary Digit. Las 16 líneas, entonces se refieren a 16 bits.- Los patrones prácticos hacen referencia a la primera línea -- (el bit menos significativo) como el bit 0, la segunda línea- es el bit 1 y así sucesivamente, la línea 16 (el bit más significativo) es referido como el bit 15. Es también común en-- la práctica hacer referencia a un grupo de 8 bits como un --- Byte. Por lo tanto, el bus de dirección de 16 bits de ancho - puede ser de 2 Bytes de ancho. Del bit 0 al bit 7 hacen el -- Byte menos significativo y del bit 8 al bit 15 hacen el Byte- más significativo, figura 1.3.

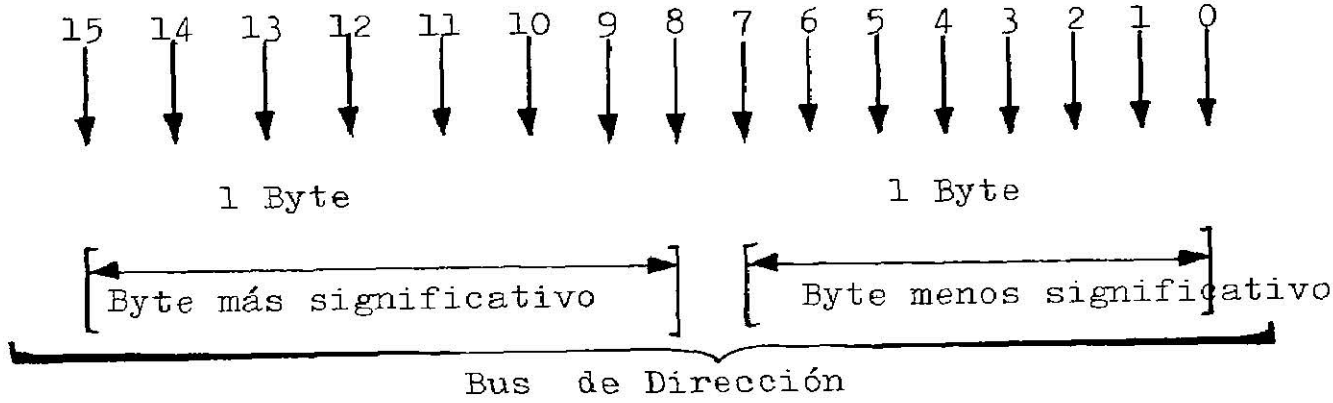


Figura 1.3 Bus de Dirección

Justo como el MPU debe direccionar el dispositivo que desea para comunicarse con él, debe también tener algunos canales através del cual pueda enviar datos o recibirlos, el dispositivo tiende a ser direccionado. Similarmente cuando sintonizamos una dirección (número) en un teléfono, debemos tener algunos caminos de envío y recibimiento de datos (hablar o escuchar). En un computador, esa transacción toma lugar en una serie de alambres (líneas) referidas a un Bus de Datos. Note que la doble dirección de las flechas de líneas de datos, figura 1.2 indican que los datos pueden ser enviados o recibidos por el MPU. De nuevo un alto voltaje en la línea de datos indica un 1 binario y un bajo voltaje un 0 binario.

El tercer tipo de Bus del microprocesador es el Bus de Control. Estas líneas controlan la secuencia de eventos para el sistema total. Por ejemplo, cuando hay datos en el Bus de Datos, el trabajo del Bus de Control es informar a un dispositivo si el MPU está listo para obtener los datos de este o si el dispositivo puede almacenar los datos en el bus de datos, esto es, si se puede Leer o Escribir. Varias funciones separadas son ejecutadas por el Bus de Control de cualquier sistema de Microcomputador.

Como se muestra en el modelo, el MPU debe proveer áreas para almacenamiento temporal de datos. Algunos MPUs tienen un acumulador, pero muchos tienen 2 o más. Nuestro sistema consta de 2 acumuladores A y B. Los Acumuladores son representados como Registros. Otros registros encontrados en el MPU son los siguientes:

- Contador del Programa (PC)
- Registro de Indice (IR o IX)
- Apuntador de Pila (SP)
- Registro de Código de Condiciones (CC)

1.4.- RANDOM ACCESS MEMORY (RAM)

Es un área de memoria en la cual se almacena información que en un momento dado puede ser borrada y volver a grabar información en el mismo lugar.

El bus de datos, figura 1.4, es el mismo que el del MPU.- Note que 3 chip select(selección de pastilla) marcados como - S1, S2 y S3. También note la word select(selección de palabra) líneas marcadas A0 a A6 que son las líneas de localización de memoria o direccionamiento, la función del chip select se pue de comparar con el seleccionamiento de página en un libro y - la word select se puede comparar con el seleccionamiento de - una línea de esa página.

Ambas el chip select y word select especifican un direc-- cionamiento sobre la memoria RAM. Estas líneas conectarán al- Bus de direccionamiento del MPU. Cuando se aplica una señal - àlta en S1 y S2 de la figura y se aplica una señal baja en S3 esta RAM particular puede ser direccionado.

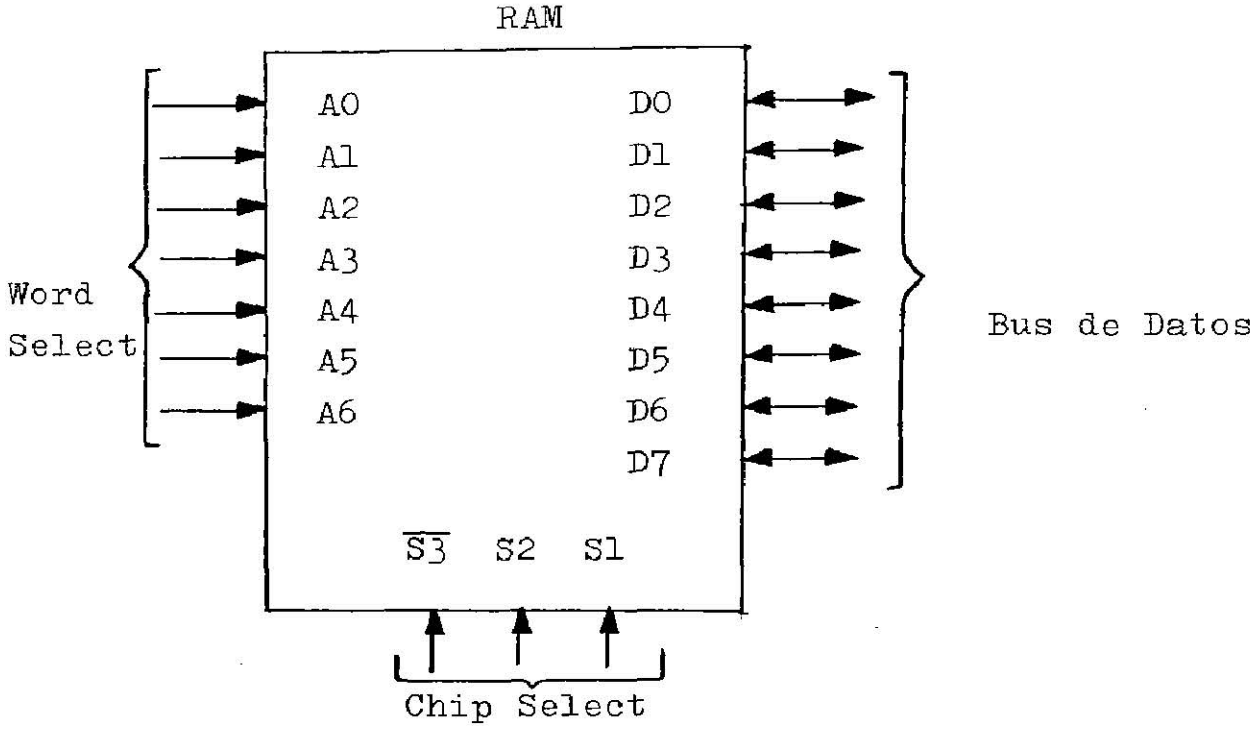


Figura 1.4 la RAM

Por ejemplo, si S1 y S2 fueran conectados al bit 14 y bit 15 del bus de dirección, respectivamente, y $\overline{S3}$ fuera conecta do al bit 13, la dirección de esa RAM puede ser 1100 0000 --- 0XXX XXXX(asumiendo que esa dirección es para un solo disposi tivo). Las Xs en la dirección pueden ser conectadas a las lí- neas de word select para direccionar una palabra individual - de esa RAM.

Por lo tanto, hay 7 líneas de word select, el número de palabras que pueden seleccionarse en un rango particular RAM es desde 000 0000(todas las líneas con 0 V) a 111 1111(todas las líneas de altò voltaje). La longitud de palabra de los datos aceptable para la RAM es de 8 bits(1 Byte) el cual puede ser colocado en el bus de datos.

La relación entre el MPU y la RAM se muestra en la figura 1.5, la línea R/W muestra una señal de línea del MPU informando a dispositivos externos que el MPU quiere (1) leer datos, si la línea es alta o (2) enviar datos, si es baja.

Para extender el proceso de word select de una RAM, asumimos que el chip select y word select son ligados al bus de dirección del MPU como se muestra en la figura 1.5. Si A14 y A15 son puestas en un estado alto y A13 en un estado bajo mientras de A0 a A6 están en un estado bajo, el contenido de las localidades de memoria 000 0000 puede entrar por el bus de datos si el MPU solicita los contenidos de esa localidad (la línea R/W es alta). Si el MPU desea leer el contenido de la localidad 0011 0111, la línea R/W puede estar en estado alto; las líneas de dirección A14 y A15 pueden ser altas y A13 puede ser baja y las líneas de dirección A0, A1, A2, A3, A4, y A5 pueden ser altas, figura 1.6, el contenido de esa localidad puede entonces ser colocado en el bus de datos para ser transferidos al MPU.

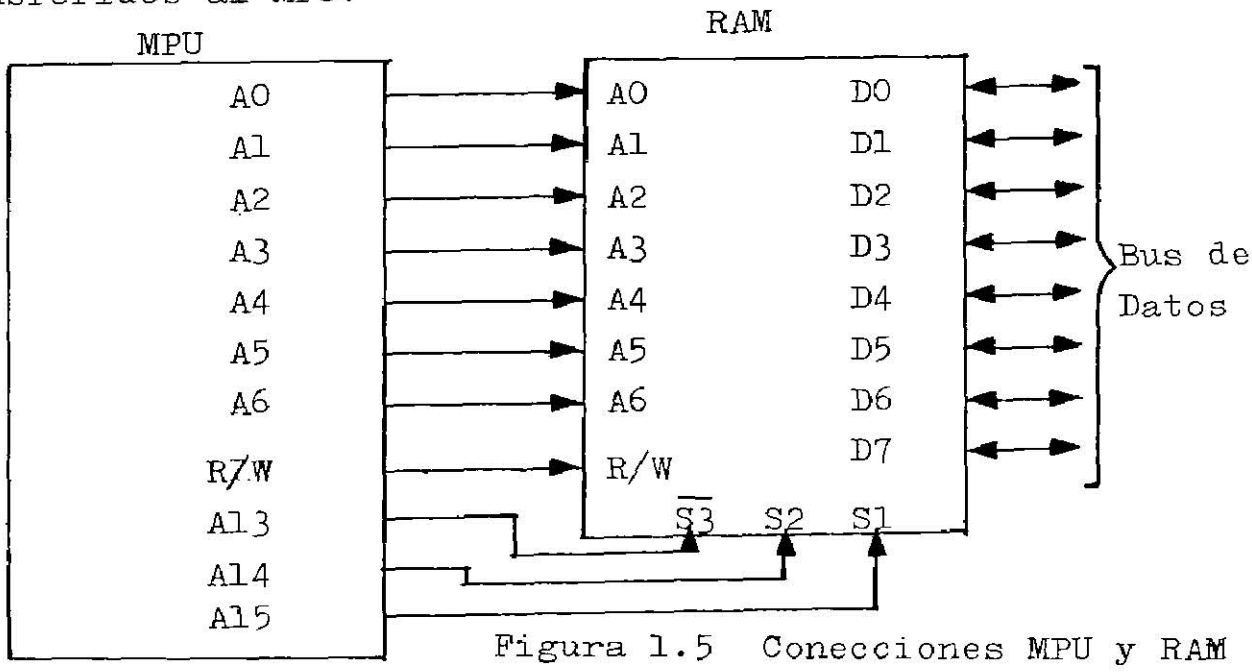


Figura 1.5 Conecciones MPU y RAM

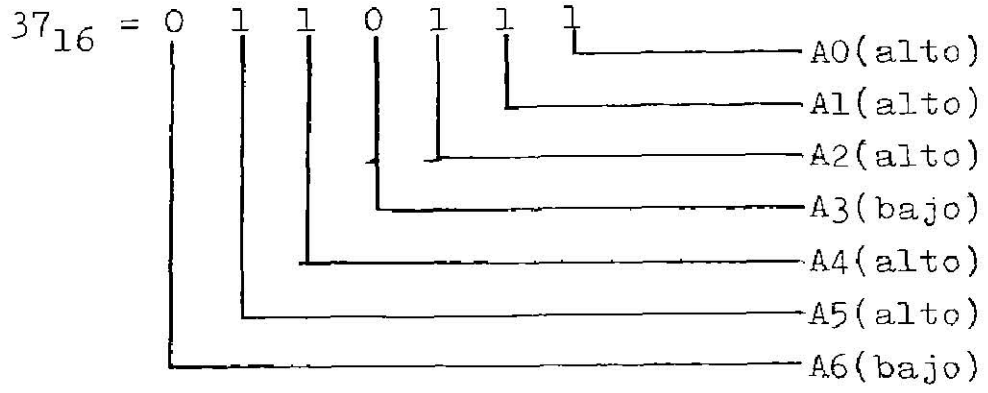


Figura 1.6 Localidades en Memoria de 37_{16}

El rango de dirección de esa RAM, como está conectada al- bus de dirección es desde 1100 0000 0000 0000(localidad 000 - 0000) a 1100 0000 0111 1111(localidad 111 1111), o desde ---- $C000_{16}$ a $C07F_{16}$.

Puede ser obvio que todas las A0s hasta A6s son ligadas a las A0 hasta A6 del bus de dirección, si tuvieramos más de - una RAM en el sistema. Sin embargo, las líneas del chip select pueden ligarse a diferentes líneas de dirección para pro-- veer a cada RAM con una unica dirección.

La RAM tenemos que considerarla con 128 localidades de -- memoria, cada 8 bits de ancho. Otros tipos varían de 1024 bi- ts cada uno a 16,384 bits cada uno.

La RAM tiene 2 tipos básicos, llamados Estáticos y Dinámi_ cos. Las RAMs dinámicas usualmente tienen más capacidad de -- almacenamiento que las estáticas puesto que sus celdas son -- más pequeñas. Sin embargo, éstas conocidas como "señal renova_ da" deben ser aplicadas a las celdas una vez cada milisegundo o el dato puede perderse. Esta señal proviene del reloj del - MPU, el cual causa que el MPU corra a una razón muy lenta. -- Las RAMs estáticas no necesitan de esta señal renovada. Tan - grande como sea el poder original aplicado, la RAM estática - retiene este dato.

1.5.- READ-ONLY MEMORY (ROM)

Tiene un esquema de dirección casi igual que la RAM. El programa explica al MPU lo que hace éste almacenado en localidades consecutivas en la ROM, pero no puede cambiar(MPU) los contenidos de las localidades ROM como las puede cambiar las localidades RAM.

Los programas están fijos en la memoria ROM mediante varias técnicas diferentes. Una es llamado Mask Programmable. Después que el usuario escribe su programa, suministra este a la manufactura de la ROM. La manufactura entonces produce la ROM, usualmente en cantidades grandes, con la estructura del programa dentro.

Una ROM típica contiene 1024 localidades separadas cada una de 8 bits de ancho(1024 x 8), figura 1.7. Las ROMs varían en tamaño como las RAMs.

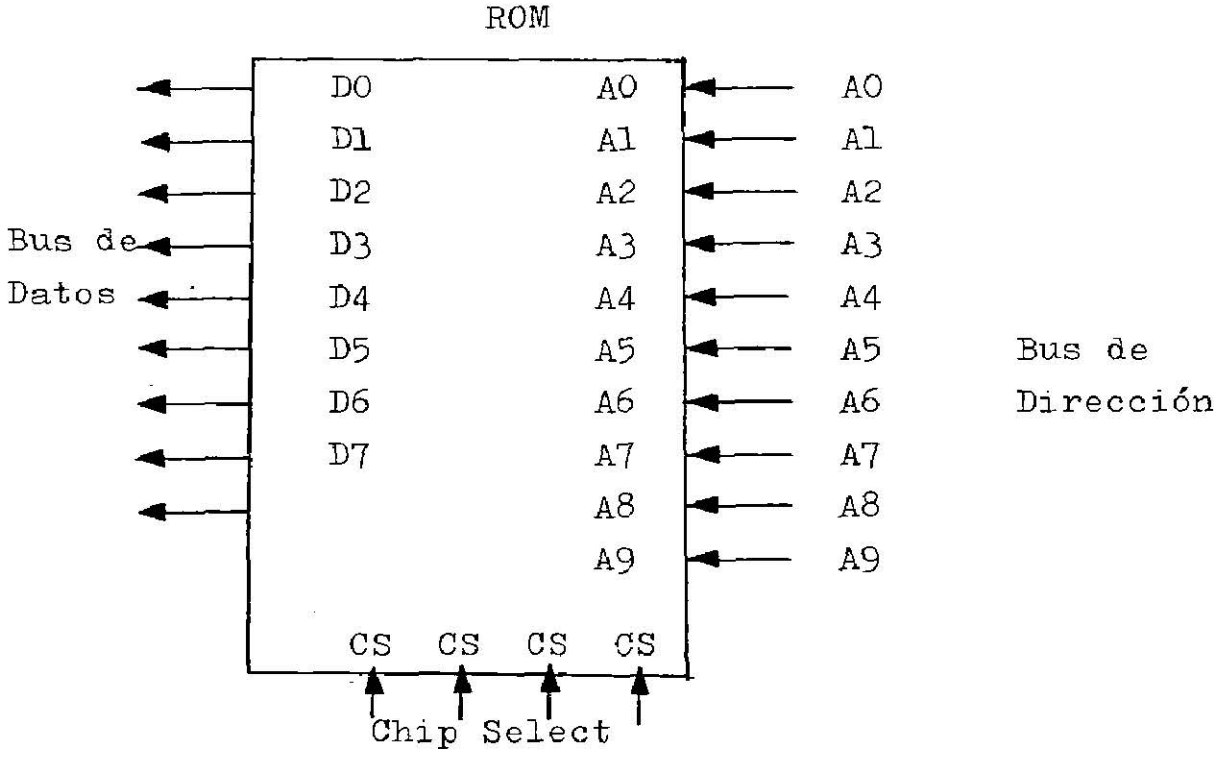


Figura 1.7 La ROM

Note que las líneas de datos están directamente hacia el MPU. Los datos pueden fluir solo desde la ROM a el MPU. Como vemos los requerimientos de la ROM deben ser seleccionados por la propia señal en el chip select desde el bus de direc--

ción, después del cual la localidad de memoria individual --- debe direccionarse através de A0 - A9 del bus de dirección.

1.6.- INPUT/ OUTPUT

El sistema de microcomputador debe proveer un camino para la obtención de datos cargándolos dentro y fuera del sistema, usualmente por medio de 2 chips llamados Input/Output "Puer-- tos o Chips de Interfase Periférica". El programa puede entonces dirigir al MPU a leer los estados de las líneas de entrada en esos chips o para enviar algunos datos a sus líneas de- salida.

1.7.- EL RELOJ

Justo como el modelo de computador desarrollado anteriormente dependiendo de un reloj en el camino para determinar -- cuantos eventos sucedieron, todos los sistemas de microcomputador son también regulados por señales de reloj(tiempo). Esas señales son encaminadas através del MPU permitiéndole la ejecución de instrucciones desde la ROM de una manera oportuna - y ordenada.

1.8.- SISTEMA DE MICROCOMPUTADOR

La relación entre todos los elementos del sistema de micro computador es:

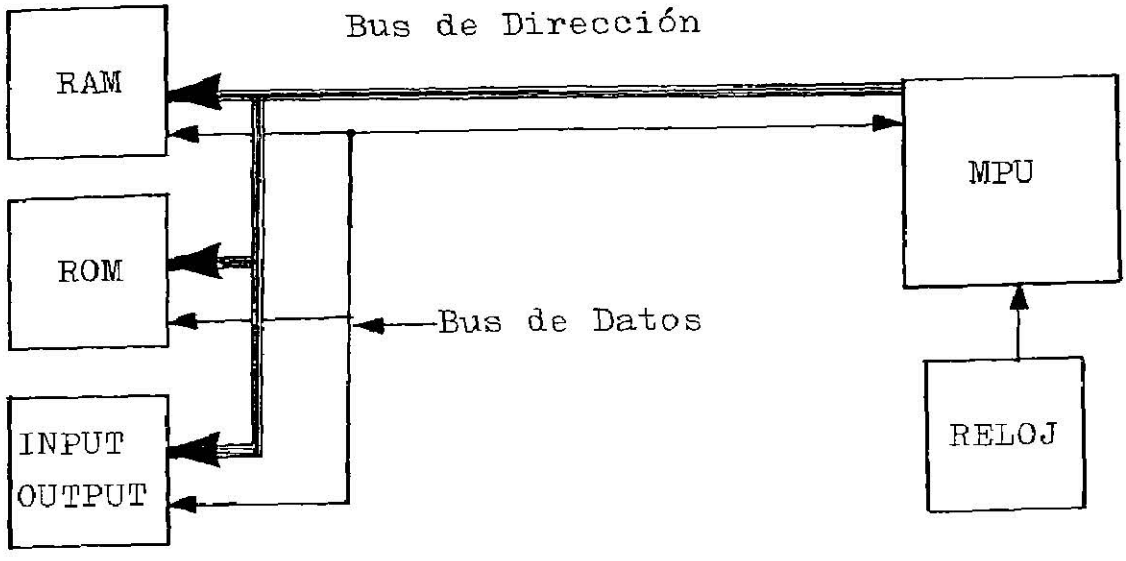


Figura 1.8 Sistema de Microcomputador

1.9.- INTERRUPTS

En el bus de control del MPU una de las líneas es llamada Line Interrupt(línea de interrupción). Interrupts es una técnica que tiene el microprocesador la cual interrumpe las actividades primarias para que el microprocesador ejecute tareas de más importancia.

El MPU monitorea algunas líneas de entrada, hace cálculos y almacena datos cuando ocurre una situación de emergencia. - Cuando el MPU entra en una situación de emergencia por una -- interrupción, éste termina la instrucción que se está ejecu-- tando y almacena el contenido en registros internos y va a -- programas internos para salvar la dificultad.

1.10.- CONTROL DE 3 ESTADOS

Es un término usado frecuentemente, aunque no entendido.- Es una técnica que permite más de un dispositivo para formar un ducto común, pero no al mismo tiempo.

En la figura 1.9, el dispositivo 1 normalmente liga al -- ducto que sirve como entrada para el MPU. Asumimos que el estado de salida del dispositivo 2 y 3 son necesarios para el - MPU. Si S1 es abierto y S2 es cerrado, el estado del dispositivo 2 puede ser alimentado dentro del MPU. Así mismo, si S1 y S2 son abiertos y S3 cerrado, el estado de S3 puede ser leído por el MPU. Cada dispositivo tiene 3 estados 1, 0 y un -- circuito abierto. Para controlar S1, S2 y S3 el dispositivo - ligado al ducto puede ser controlado, figura 1.10.

El dispositivo puede aparecer como un circuito abierto -- al bus cuando la entrada de 3 estados sea baja(0). Sin embargo, si es alta(1), la salida del dispositivo(1 o 0) puede ser ligada al ducto. La figura 1.10 muestra solo una línea. El -- dispositivo 1 puede representar una pieza de equipo(Typewrite) que en realidad tiene 8 líneas de salida ligada a 8 líneas de bus de datos. En esta situación pueden estar 8 compuertas -- idénticas para cada línea del dispositivo 1, del cual las líneas de control de 3 estados de cada compuerta puede ser con-

trolada por el mismo origen tal que la salida de las 8 líneas deben ser compuertas para el bus de datos simultáneamente.

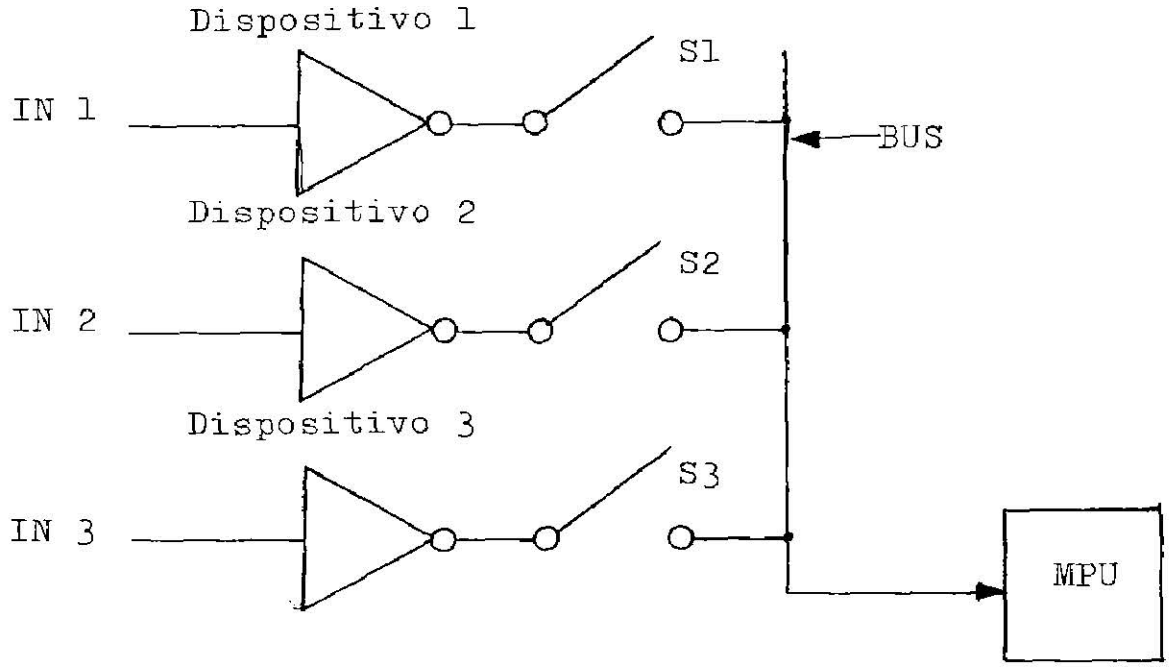


Figura 1.9 Control de 3 Estados

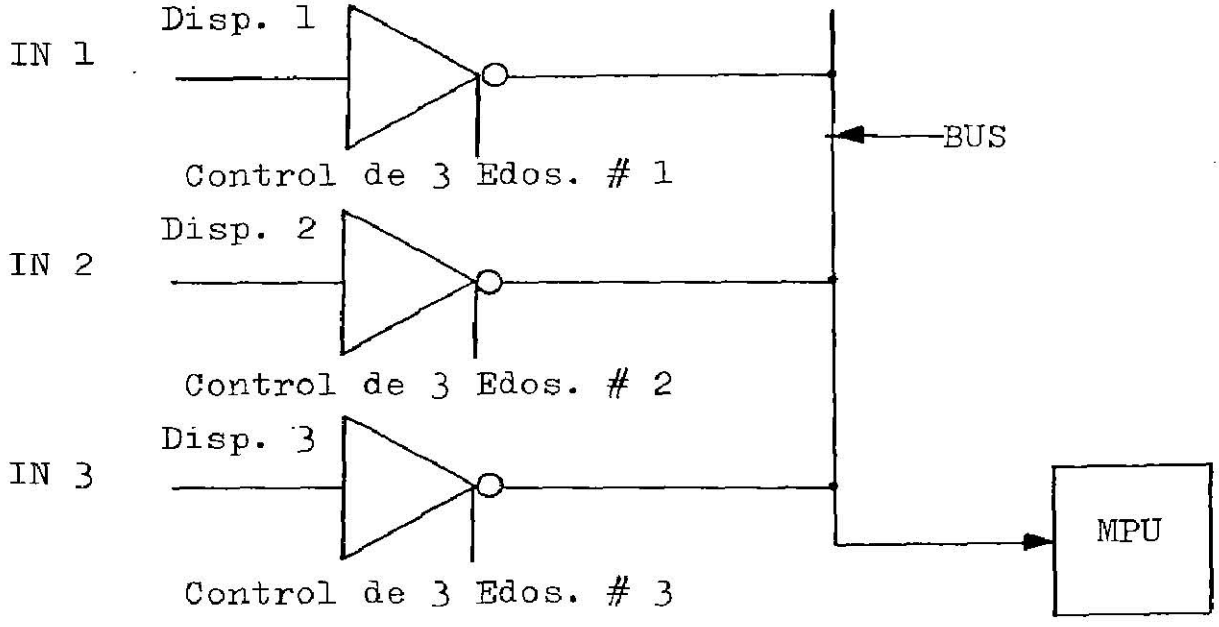


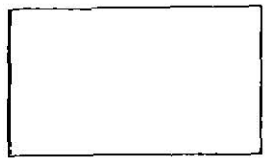
Figura 1.10 Control de 3 Estados

Muchos microprocesadores tienen un pin de entrada disponible en el MPU para ese propósito. El MC 6800 tiene un pin llamado TSC(control de 3 estados). Cuando este pin está en estado bajo, el bus de dirección y la línea R/W están en su estado normal. Sin embargo cuando una señal alta es aplicada a -- este pin, el bus de dirección y la línea R/W es abierta.

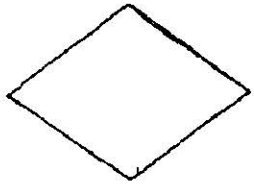
2.- P R O G R A M A C I O N

2.1.- DIAGRAMA DE FLUJO

Es una representación gráfica que ilustra los pasos lógicos, cálculos y decisiones en secuencia, que deben ser ejecutados para realizar algún trabajo específico. Los símbolos -- básicos usados son:



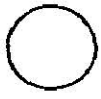
Proceso o Acción



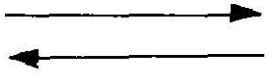
Decisión

Entrada o Salida

Continuación o Término



Conector



Líneas de Flujo

2.2.- MNEMONICOS

Como sabemos todos los computadores digitales operan solo con números binarios. No entienden los estatutos en inglés -- o cualquier otro lenguaje. Sin embargo, muchas manufactureras de microcomputadoras dan un código de 2, 3 o 4 letras que describen la función de cada instrucción. Este código conocido -

como Mnemonico, tiene un número hexadecimal equivalente(o binario) para representar la función. Por ejemplo del modelo de computadora anterior una de las instrucciones era "sumar el - acumulador A al acumulador B y colocar el resultado en el acumulador A", el mnemonico para este caso es ABA(suma B a A). - La operación ABA puede ser representada por el código hex 1B- el cual aparece en la ROM como 0001 1011. Así cuando el MPU - decodifica la instrucción, esta dice,"debo sumar los acumuladores A y B y colocar el resultado en el acumulador A, e ir - entonces a la próxima localidad secuencial en memoria por la- próxima instrucción".

Definición: Mnemonico es un código simple, usualmente alfabético, que es representativo de la función de la instrucción - que está presente.

2.3.- ENSAMBLADOR

La pregunta que tenemos ahora es ¿ Cómo vamos de la codificación mnemonica al lenguaje binario del computador? Los -- programas se escriben con códigos mnemonicos llamados Programas Fuente.

Lenguaje de Máquina: Otros términos usados para el lenguaje binario(1s y 0s), el cual es el único lenguaje que el -- computador digital reconoce.

Una vez que el código fuente es escrito para la ejecución de un trabajo, la conversión al lenguaje de máquina puede lograrse usando 2 caminos. Uno es haciendo referencia al manual de programación de la manufactura y manualmente mirando el -- lenguaje de máquina equivalente para cada código mnemonico en la lista fuente. Esto es muy tedioso, ya que se consume mucho tiempo y requiere cálculos de muy alta precisión.

El segundo método de conversión de código mnemonico a lenguaje de máquina es con un Ensamblador. Un Ensamblador es un- programa independiente diseñado para convertir un programa -- fuente(hecho en código mnemonico) en un programa de lenguaje- de máquina, figura 2.1. Cada tipo de microprocesador disponi-

ble debe tener un Ensamblador ligado al MPU. El Ensamblador - para convertir el lenguaje fuente de la MC 6800 al lenguaje - de máquina debe de correrse en una IBM 360.

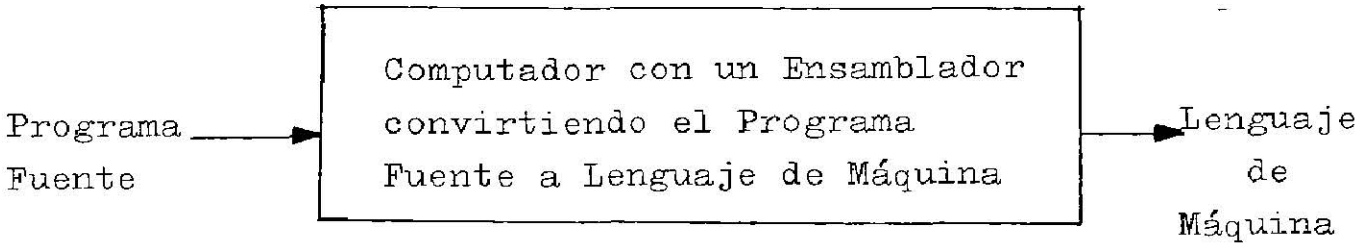


Figura 2.1 El Ensamblador.

2.4.- ASCII

El código ASCII proviene de la American Standard Code for Information Interchange. Este es un código estandard usado -- para el intercambio de información, entrada y salida, hacia - o desde los dispositivos como typewrite e impresoras en línea. Note en la figura 2.2, que el código ASCII para una A es ---- 1000001. Este nos auxilia, por ejemplo, si queremos encadenar el typewrite al sistema, escribiendo una A, debemos transmi-- tir la señal 1000001 en las líneas de datos para la typewrite.

C	ASCII	C	ASCII	C	ASCII	C	ASCII
@	1000000	N	1001110	\	1011100	$	0100100
A	1000001	O	1001111	]	1011101	%	0100101
B	1000010	P	1010000	↑	1011110	&	0100110
C	1000011	Q	1010001	NULL	0000000	'	0100111
D	1000100	R	1010010	HORIZ TAB	0001001	(	0101000
E	1000101	S	1010011	LINE FEED	0001010	)	0101001
F	1000110	T	1010100	VER TAB	0001011	*	0101010
G	1000111	U	1010101	FROM FEED	0001100	+	0101011
H	1001000	V	1010110	CARRIAGE RETURN	0001101	,	0101100
I	1001001	W	1010111	RUBOUT	1111111	-	0101101
J	1001010	X	1011000	SPACE	0100000	.	0101110
K	1001011	Y	1011001	!	0100001	/	0101111
L	1001100	Z	1011010	"	0100010	0	0110000

C	ASCII	C	ASCII	C	ASCII	C	ASCII
M	1001101	[	1011011	#	0100011	1	0110001
2	0110010	3	0110011	4	0110100	5	0110101
6	0110110	7	0110111	8	0111000	9	0111001
:	0111010	;	0111011	<	0111100	=	0111101
>	0111110	?	0111111				

Figura 2.2 El Código ASCII.

2.5.- MODELO DE PROGRAMACION.

La figura 2.3 muestra un esquema en el cual se encuentran representados los registros internos del MPU, los cuales son:

- Acumulador A
- Acumulador B
- Registro de Indice
- Contador del Programa
- Apuntador de Pila
- Registro de Código de Condiciones

Esta representación es muy útil desde el punto de vista de un usuario que le interese solo la programación del MPU, y es conocida con el nombre de Modelo de Programación.

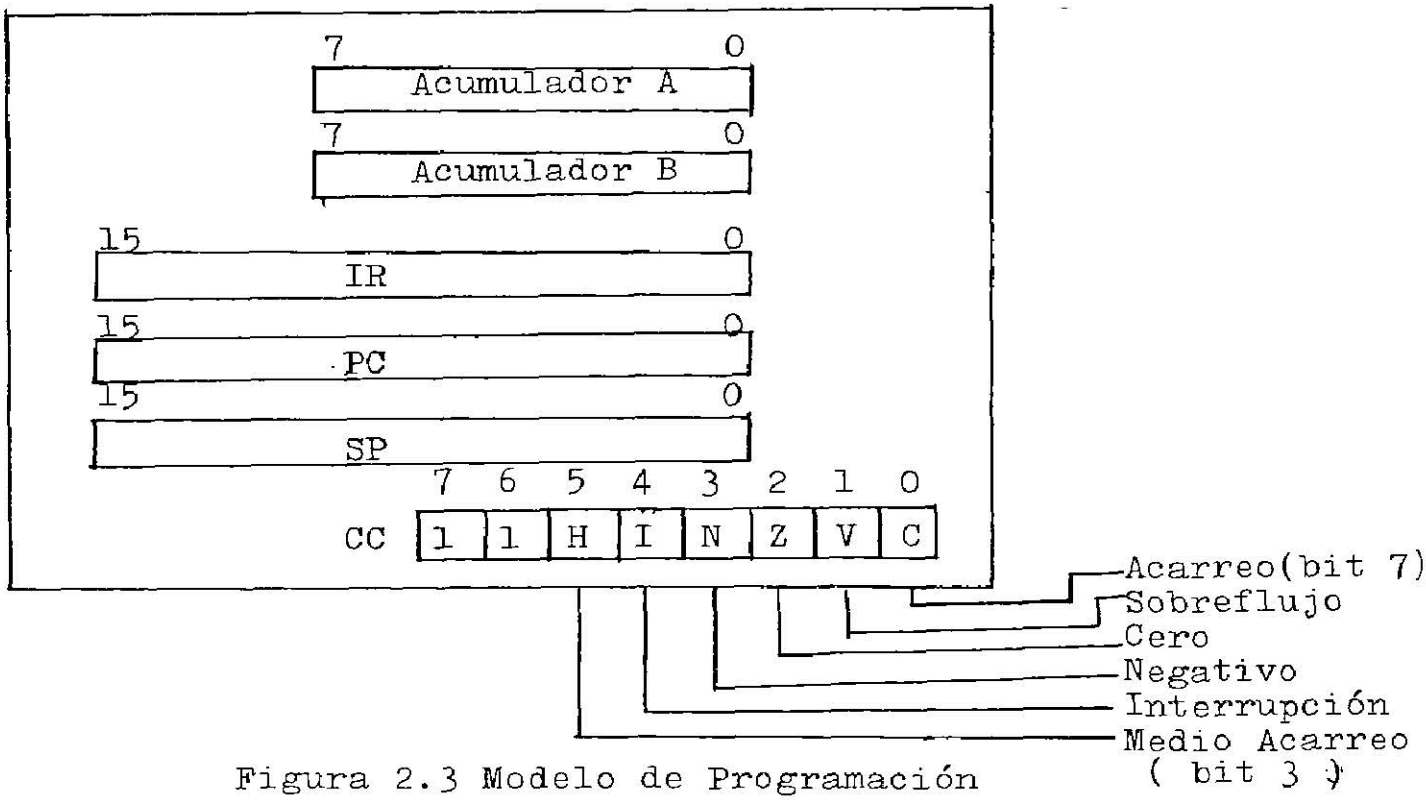


Figura 2.3 Modelo de Programación

3.- M O D O S D E D I R E C C I O N A M I E N T O

Existen 7 modos de direccionamiento, los cuales están en-relación directa con el tipo de instrucciones que realiza el-MPU; estos modos son:

- Con operando implícito
  + De Acumulador
  + Inherente o Implicado
- Con operando explícito
  + Relativo
  + Inmediato
  + Directo
  + Extendido
  + Indicado

3.1.- ACUMULADOR

Las instrucciones en las que solo se utiliza uno de los - dos acumuladores, tienen este tipo de direccionamiento.

Ejemplo:

Mnemonico	Código	Operación
DEC A	4C	AccA←--AccA - 1

3.2.- INHERENTE

Tienen este modo de direccionamiento, aquellas instrucciones que únicamente emplean los elementos de la unidad de micro procesamiento de registro de código: registro de código de -- condiciones, apuntador de pila, registro de índice, acumula-- dor A y acumulador B ala vez.

Ejemplo:

Mnemonico	Código	Operación
ABA	1B	AccA←-AccA + AccB

3.3.- RELATIVO

Este modo de direccionamiento está asociado a las instrucciones de ruptura de secuencia(con excepción del JMP) condicionales o no. El siguiente byte del código de operación contiene un número expresado en complementos a 2 el cual se suma al contador del programa, después de que este se ha incrementado para direccionar la siguiente instrucción. Esto permite-

efectuar saltos a los 126 bytes anteriores a la instrucción - de derivación, así como a los 129 posteriores.

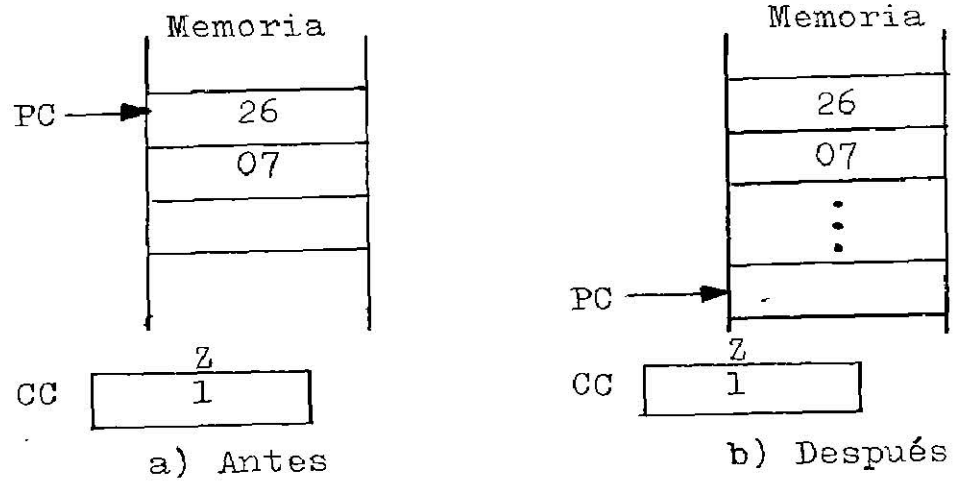
La dirección D alcanzada a partir de la instrucción de salto localizada en D1, con un desplazamiento DESP la podemos -- expresar como:

D = (D1+2) + DESP con -128 DESP 127

Ejemplo:

	Mnemonico	Código	Operación
	BNE ALFA	26 07	Si z=0, PC←-PC+02+07

ALFA	--------		

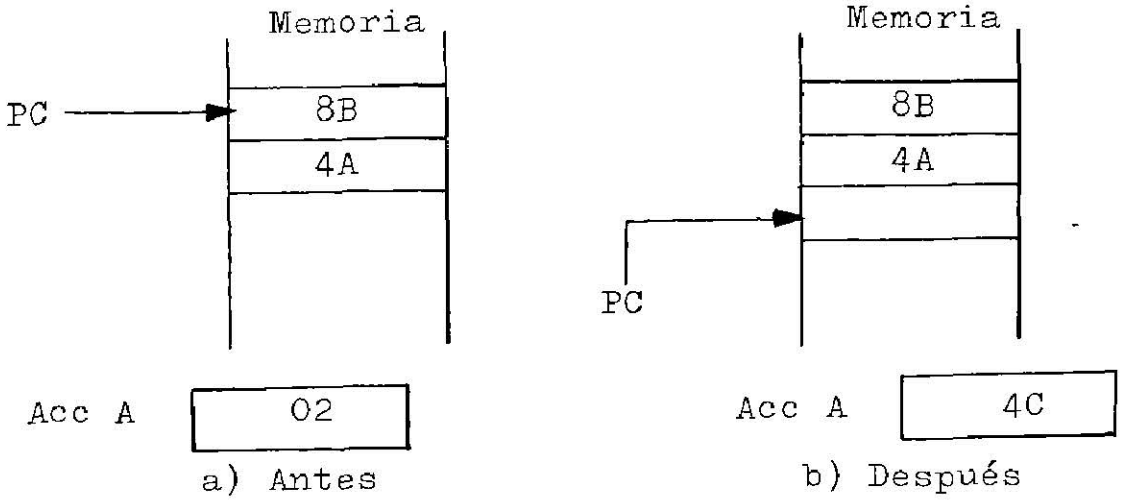


3.4.- INMEDIATO

En el direccionamiento inmediato la instrucción está formada por 2 bytes, el primero es el código de operación y el - segundo el operando. Excepciones a esto son las instrucciones LDS, LDX(cargar el apuntador de pila y el registro de índice- respectivamente) y CPX(comparar el registro de índice con un- número de 16 bits), los cuales contienen el operando en el -- segundo y tercer byte de la instrucción.

Ejemplo:

Mnemonico	Código	Operación
ADD A #$4A	8B 4A	AccA←-AccA + 4A

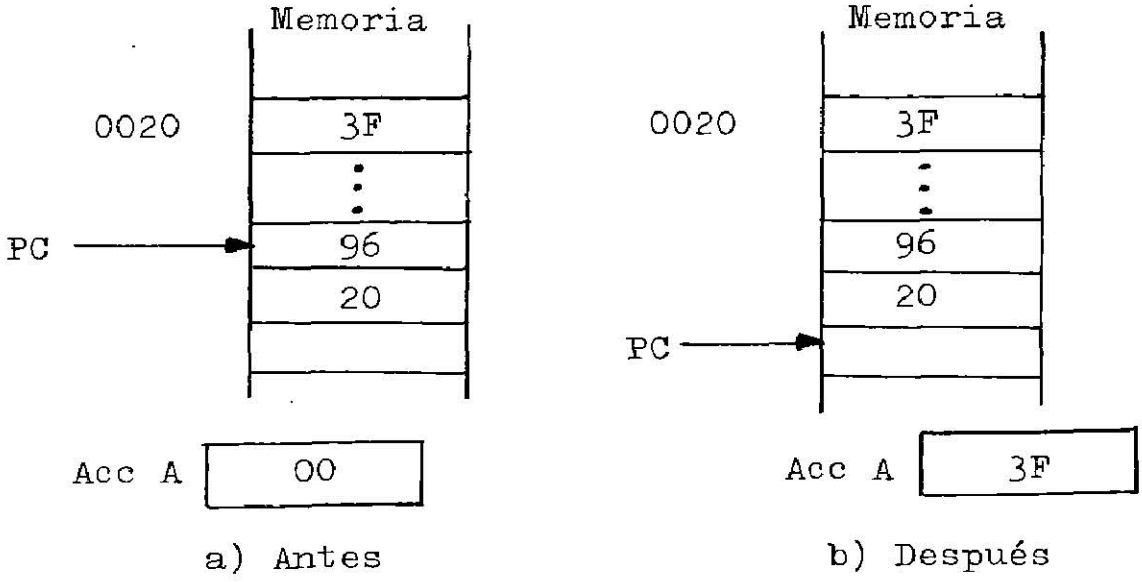


3.5.- DIRECTO

En este modo de direccionamiento, la instrucción está formada por 2 bytes. El primero contiene el código de operación y el segundo la dirección del operando. Esto permite accesar las primeras 256 bytes de la memoria.

Ejemplo:

Mnemonico	Código	Operación
LDA A $20	96 20	AccA <-- (20)



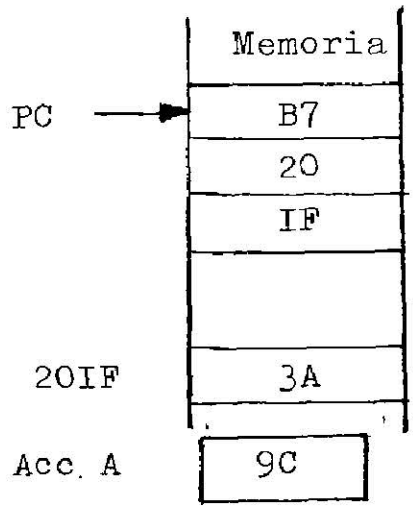
3.6.- EXTENDIDO

Una instrucción bajo este modo de direccionamiento, está- constituida por 3 bytes, de los cuales el primero contiene el código de operación y los dos siguientes la dirección del operando. Los 8 bits más significativos de esta dirección están- contenidos en el segundo byte.

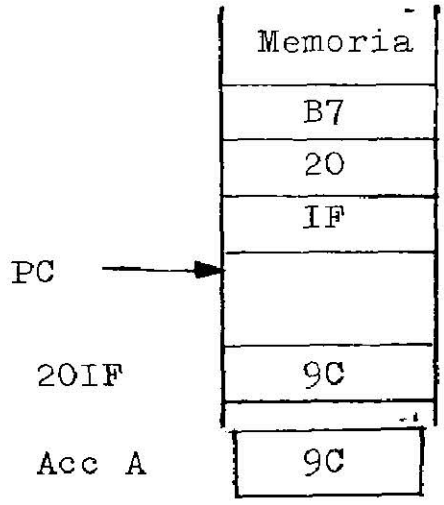
Con este modo de direccionamiento es posible accesar toda la memoria del microcomputador.

Ejemplo:

Mnemonico	Código	Operación
STA A $20IF	B7 20IF	(20IF) ← Acc A



a).- Antes



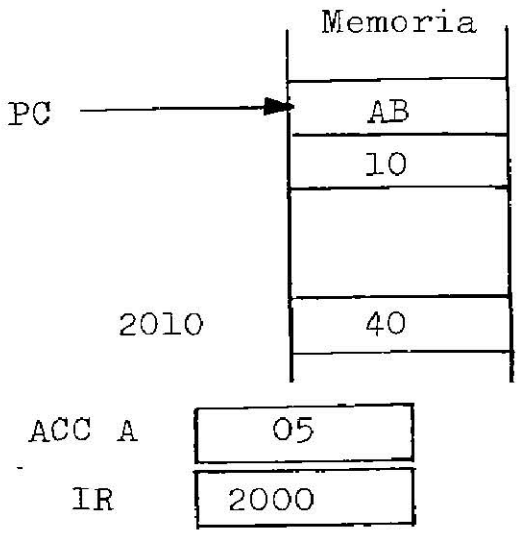
b).- Después

3.7.- INDICADO

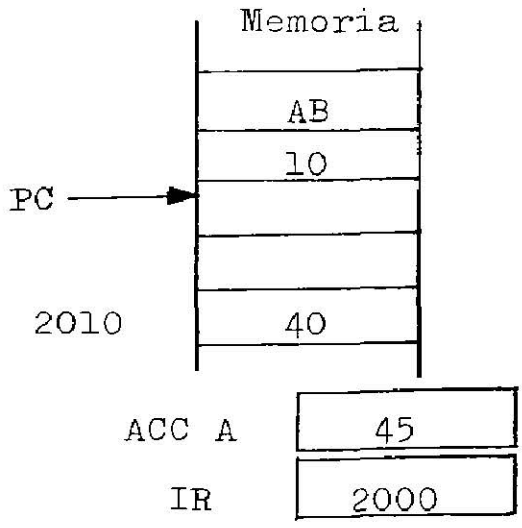
En este modo de direccionamiento la instrucción está formada por 2 bytes. El primero contiene el código de operación- y el segundo un número en complementos a 2 que se suma al contenido del registro de índice para formar la dirección del -- operando. Esta dirección se forma en un registro temporal y - el contenido del registro de índice no se modifica.

Ejemplo:

Mnemonico	Código	Operación
ADD A $10,X	AB 10	AccA ←-AccA+(IR+10_{16})



a) Antes



b) Después

4.- H A R D W A R E

4.1.- ARQUITECTURA DEL MEK 6802 - D5

El MEK 6802-D5 es un sistema de bajo costo diseñado para evaluar la capacidad de la familia de componentes del MPU MC-6800. Puede ser utilizado como una herramienta para aprender técnicas de programación y desarrollar aplicaciones de microprocesadores.

El sistema consiste de un microprocesador MC 6802, un sistema operativo en una ROM, despliegue de 6 indicadores de 7 - segmentos, un teclado de 25 teclas, un entrefaz estandard para cinta magnética y una entrefaz adaptadora de periféricos - (dispositivo de entrada/salida en paralelo). También contiene un circuito de reloj controlado por un cristal de una frecuencia de oscilación de 3.58 Mhz que se utiliza para sincronizar todos los elementos del sistema.

4.1.1.- EL MICROPROCESADOR MC 6802

La parte más importante del sistema MEK 6802-D5 es el microprocesador MC 6802, que tiene el mismo conjunto de instrucciones que el MC 6800. El MC 6802 también contiene 128 bytes de RAM y un circuito de reloj interno controlado por un cristal.

La figura 4.1 muestra un diagrama a bloques general del - Microprocesador MC 6802. La memoria RAM de 128x8 bits fué --- añadida al microprocesador básico. Los primeros 32 bytes pueden ser operados en un modo de baja potencia por medio de un voltaje Vcc de retención(Vcc Standby). Estos 32 bytes pueden almacenarse por tiempo indefinido puesto que el voltaje de retención puede ser independiente del voltaje de alimentación - del MPU.

En la figura 4.2 se muestran los registros del MPU a los que el usuario tiene acceso por programa; a este diagrama se le conoce como "modelo de programación":

- Contador del Programa(PC).- El PC es un registro de 16 bits que contiene la dirección del siguiente byte de la instruc-

ción que será buscada en memoria para su ejecución. Cuando el valor actual del PC se coloca en el ducto de direcciones el PC se incrementa automáticamente.

- Apuntador de Pila(SP).- El SP es un registro de 16 bits que contiene la dirección de la localidad disponible en una pila externa, que es normalmente una RAM que puede tener cualquier dirección conveniente. Cada vez que se introduce un dato en la pila, el SP se decrementa, y cada vez que se saca un dato, se incrementa.

- Registro de Indice(IR).- El IR es un registro de 16 bits que se usa para almacenar datos o una localidad de memoria de 16 bits utilizada en un modo de direccionamiento de memoria. Este registro puede ser decrementado, incrementado, cargado, almacenado o comparado.

- Acumuladores A y B.- El microprocesador MC 6802 contiene 2 acumuladores(A y B) de 8 bits que se usan como registros temporales que contienen los operandos y resultados de las operaciones efectuadas en la Unidad Aritmética Lógica.

- Registro de Código de Condiciones(CC).- El CC es un registro de 8 bits que indica si el resultado de una operación en la Unidad Aritmética Lógica fué: Negativo(N), cero(Z), Sobreflujo(V), Acarreo del Bit 7(C) y Medio acarreo del Bit 3(H). Estos bits se utilizan para probar el estado del MPU en las instrucciones de salto condicional. El bit 4(I) del CC es el bit de Máscara de Interrupción. Los bits 6 y 7 del CC siempre son "1". Debe señalarse que las diferentes instrucciones afectan los bits de este registro de varias maneras. Para determinar la forma en que una instrucción particular afecta cada bit.

  + Acarreo(C).- Este bit(0) del CC se coloca a "1" si, después de la ejecución de ciertas instrucciones, hay un acarreo del bit más significativo del resultado de la operación que se está realizando. De otra manera, se coloca en "0".

+ Sobreflujo(V).- Este bit(1) del CC se coloca en "1" cuando un sobreflujo en complementos a 2 resulta de una operación aritmética y se coloca en "0" si el sobreflujo mencionado no ocurre en ese tiempo. Generalmente ocurre sobreflujo en complementos a 2 si la última operación resulta ser un número mayor que el rango de $\pm$ 127 de un registro de 8 bits.

+ Cero(Z).- Este bit(2) del CC se coloca en "1", si el resultado de la operación lógica o aritmética es cero; de otra manera se coloca en "0".

+ Sign(N).- Este bit(3) del CC se coloca en "1", si el bit-7 del resultado de una operación lógica o aritmética es - "1", si el bit 7 del resultado mencionado es "0" entonces el bit N es colocado en "0".

+ Máscara de Interrupción(I).- Este bit(4) del CC deshabilita las solicitudes de interrupción causadas por la línea $\overline{IRQ}$ si se coloca en "1". Si el bit I se coloca en "0", el microprocesador puede ser interrumpido por un estado bajo de la línea $\overline{IRQ}$. Este bit puede ser colocado en "1" o en- "0" por un programa utilizando instrucciones especiales - para ello.

+ Medio Acarreo(H).- Este bit(5) del CC se coloca en "1", si durante una operación aritmética, hay un acarreo del bit-3 al bit 4 en el resultado. Si no se presenta el acarreo- mencionado, el bit H se coloca en "0".

4.1.1.1.- DESCRIPCION DE LAS SEÑALES DEL MPU MC 6802

Debido a que la operación del MPU debe estar acompañada - por la operación de los dispositivos conectados a él, se requiere que ciertas señales de control y temporización sean -- proporcionadas por el MPU a dichos dispositivos, además el -- MPU debe estar monitoriando ciertas señales de entrada, las - cuales intervienen en la operación del MPU se ilustran en la- figura 4.1 y son:

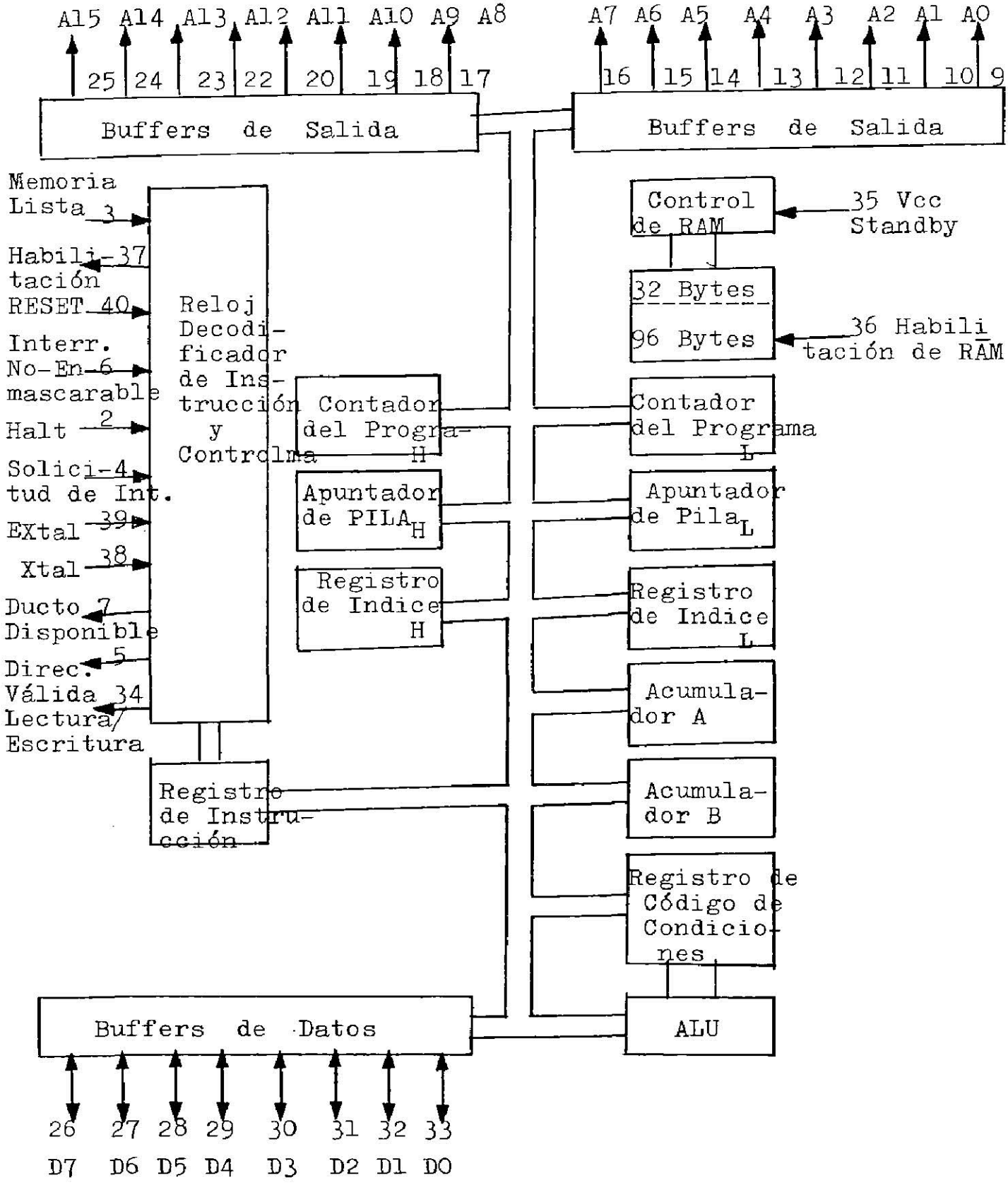

A15 A14 A13 A12 A11 A10 A9 A8
25 24 23 22 20 19 18 17
A7 A6 A5 A4 A3 A2 A1 A0
16 15 14 13 12 11 10 9
Buffers de Salida
Buffers de Salida
Memoria Lista 3
Habili-37 tación
RESET 40
Interr. No-En-6 mascarable
Halt 2
Solici-4 tud de Int.
EXtal 39
Xtal 38
Ducto 7 Disponible
Direc. 5
Válida 34
Lectura/ Escritura
Reloj Decodificador de Instrucción y Control
Contador del Progra- H
Apuntador de PILA H
Registro de Indice H
Registro de Instrucción
Control de RAM
35 Vcc Standby
32 Bytes
96 Bytes
36 Habili tación de RAM
Contador del Programa L
Apuntador de Pila L
Registro de Indice L
Acumula- dor A
Acumula- dor B
Registro de Código de Condicio- nes
ALU
Buffers de Datos
26 27 28 29 30 31 32 33
D7 D6 D5 D4 D3 D2 D1 DO

Figura 4.1

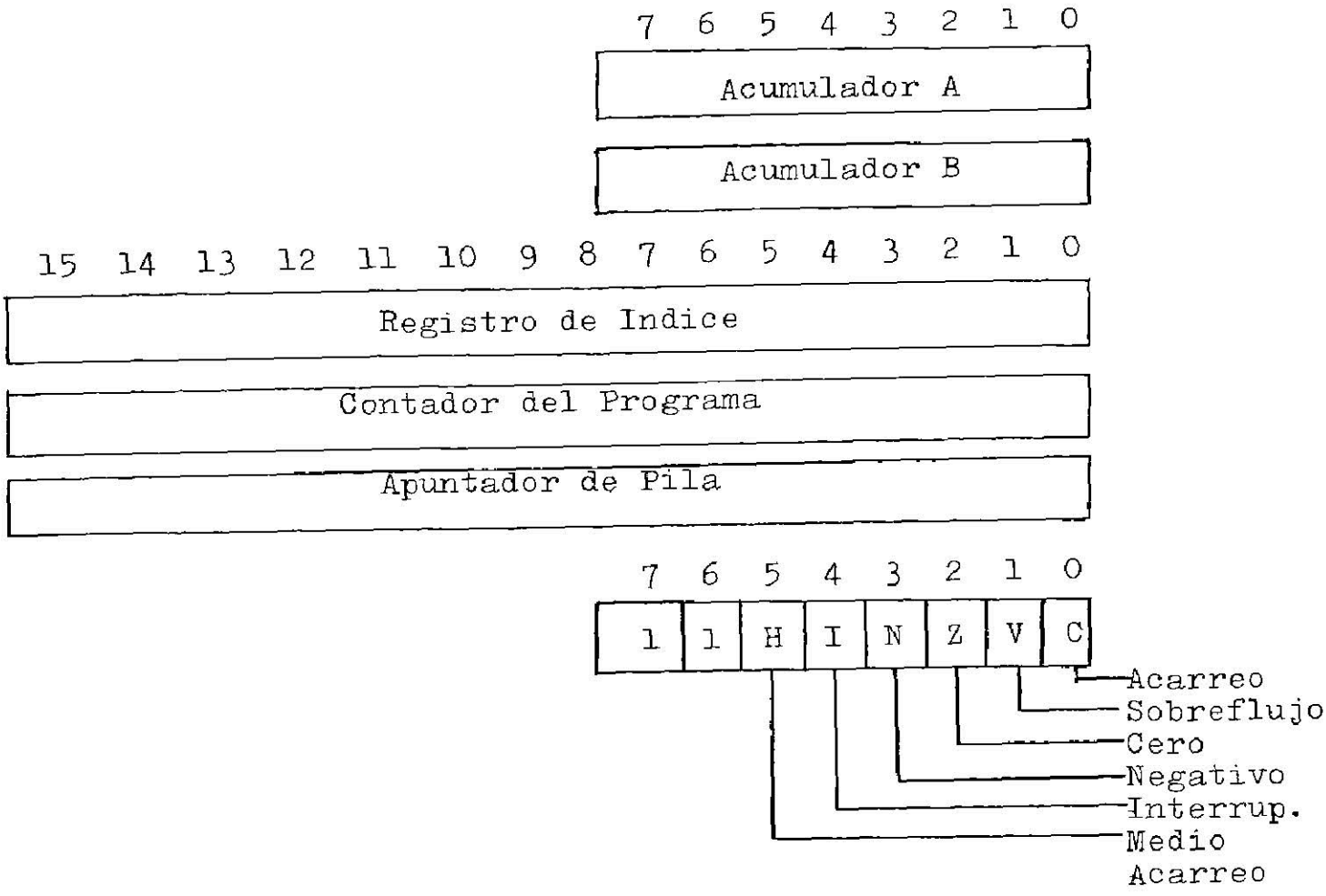


Figura 4.2 Registros del MPU MC 6802

- Ducto de Direcciones(A0-A15).- Estas 16 líneas de salida se usan para seleccionar dispositivos externos al MPU. Las direcciones son generadas por el propio MPU. Las salidas de este ducto tienen "buffers" de tres estados capaces de manejar una carga TTL y 90 pF de carga capacitiva.

- Ducto de Datos(D0-D7).- Consta de 8 líneas bidireccionales y se usa para transferir datos a un dispositivo periférico desde el MPU y viceversa. Este ducto contiene "buffers" de tres estados capaces de manejar una carga TTL y 130 pF de carga capacitiva.

- $\overline{\text{HALT}}$.- Cuando la entrada de esta línea tiene un nivel alto, el MPU busca y ejecuta las instrucciones. Cuando se pone en estado bajo, el MPU terminará la ejecución de la instrucción que está procesando y después se parará, en estas condiciones la línea BA estará en nivel alto, la línea VMA en --

nivel bajo y el ducto de datos tendrá un estado de alta --- impedancia. El ducto de direcciones tendrá la dirección de- la siguiente instrucción.

Para la operación de una sola instrucción, la transición ne gativa de la línea $\overline{\text{HALT}}$ no debe de ocurrir durante los últi mos 200 nseg. de la señal E y la línea $\overline{\text{HALT}}$ debe tener nivel alto durante un ciclo de reloj.

Si una interrupción($\overline{\text{IRQ}}$ o $\overline{\text{NMI}}$) ocurre mientras el MPU está- en estado de paro, la interrupción será "bloqueada" dentro- del MPU, hasta que éste salga del modo de paro, en ese ins- tante atenderá la interrupción. La línea $\overline{\text{HALT}}$ permite que - una fuente externa controle la ejecución del programa, eje- cutándose una instrucción a la vez. Esta capacidad es parti cularmente útil durante la revisión de un programa. En ope- ración normal del sistema, ésta línea se conecta a 5 V.

- Lectura/Escritura(R/W).- Esta señal generada por el MPU in- dica a todos los dispositivos externos que éste se encuen-- tra en estado de lectura(alto) o en estado de escritura(ba- jo). El estado normal es el de lectura, mismo que se presen ta cuando el MPU está en condición de paro. Esta línea de - salida es capáz de manejar una carga TTL y 90 pF de carga - capacitiva.

- Dirección Válida(VMA).- Esta salida indica a los dispositi- vos periféricos que la dirección presente en el ducto de di recciones es válida. En operación normal, ésta señal debe - ser utilizada para habilitar dispositivos periféricos tales como el PIA o el ACIA. Esa señal no es de tres estados y -- cuando se activa(nivel alto) puede manejar directamente una carga TTL y 90 pF de carga capacitiva.

- Ducto Disponible(BA).- Normalmente, esta línea de salida -- tendrá un nivel bajo, indicando que el ducto de direcciones está bajo el control del MPU; cuando tenga un nivel alto, - el ducto de direcciones está disponible(pero no en condi--- ción de alta impedancia), esto ocurre si la línea de $\overline{\text{HALT}}$ - tiene un nivel bajo, o el MPU se encuentra en una condi ---

ción de espera como resultado de la ejecución de una ins-- trucción de espera. La línea BA es capáz de manejar una -- carga TTL y 30 pF de carga capacitiva.

- Solicitud de Interrupción($\overline{IRQ}$).- Esta línea de entrada solicita que una secuencia de atención a interrupciones sea generada dentro del sistema. El MPU espera hasta que se com-- plete la instrucción que está siendo ejecutada antes de reconocer la solicitud. En este momento, si el bit de máscara de interrupción del registro CC no es "1", el MPU comienza la secuencia de interrupción: El IR, PC, los Acumuladores y el CC se almacenan en pila. Después de esto, el MPU responde a la solicitud de interrupción colocando un "1" al bit - de la máscara de interrupción para que no puedan ocurrir -- más interrupciones. Luego de esto, una dirección de 16 bits es cargada al contador del programa con el contenido de las localidades de memoria FFF8 y FFF9. En estas localidades se coloca la dirección de inicio de la rutina de interrupción. La línea $\overline{HALT}$ debe tener un estado alto para que las inte-- rrupciones sean atendidas; si tiene un nivel bajo, las interrupciones son "bloqueadas" dentro del MPU.

- Restablecimiento($\overline{RESET}$).- Esta entrada se usa para restablecer e inicializar el MPU después de una condición de encendido o cuando se desee inicializar todo el sistema. Cuando- tiene un nivel bajo, el MPU está inactivo y la información- de los registros se pierde. Si un nivel alto se detecta por esta entrada, indica al MPU que inicie una secuencia de restablecimiento; para esto, las últimas 2 localidades de memoria(FFFE, FFFF) contienen la dirección(parte alta y baja -- respectivamente) de inicio de esta secuencia.
  Durante la rutina de restablecimiento, el bit de la máscara de interrupción se coloca en "1" y debe colocarse en "0" -- después de dicha rutina para que el MPU pueda ser interrumpido por la línea $\overline{IRQ}$.

- Habilitación de RAM(RE).- Esta línea de entrada controla la RAM del MPU MC 6802. Cuando RE se coloca en estado alto, la

RAM del MPU es habilitada para responder a los controles -- del MPU; si se coloca en estado bajo, dicha RAM es deshabilitada. Esta terminal también puede ser utilizada para deshabilitar las operaciones de lectura y escritura en la RAM- durante la operación de encendido del sistema.

- Conexiones de Cristal(EXTAL, XTAL).- El MPU MC 6802 tiene - un circuito de reloj interno que puede ser controlado por - un cristal. Estas salidas del MPU deben conectarse en paralelo a un cristal de resonancia fundamental. Este cristal - no puede ser sustituido por un circuito RC ó LC.

- Habilitación(E).- Esta salida proporciona la señal de reloj del MPU para el sistema. Esta señal es de una sola fase y - compatible con TTL, puede estar condicionada por la señal - de memoria lista(MR). E es equivalente a la señal Ø2 del -- MPU MC 6800.

- Memoria Lista(MR).- MR es una señal de entrada de control - compatible con TTL, la cual permite el alargamiento de la - señal E. Cuando MR tiene un nivel alto, la señal E opera -- normalmente, pero cuando tiene un nivel bajo, la señal E -- puede ser alargada en múltiplos de mitades de períodos de - reloj permitiendo así la interconección con memorias lentas. El alargamiento máximo de E es de 10 microsegundos.

- Voltaje de Retención(Vcc Standby).- En esta terminal se proporciona el voltaje para los primeros 32 bytes de RAM, el - rango es de 4.0 V a 5.25 V, y sirve para retener los datos- de esta porción de memoria con una condición de encendido,- de apagado o de retención. La corriente máxima que necesita es de 8 mA a 5.25 V.

- Interrupción No-Enmascarable($\overline{\text{NMI}}$).- Una transición negativa en esta entrada provoca que una secuencia de interrup--- ción no enmascarable sea generada dentro del MPU. Tal como- lo hace con $\overline{\text{IRQ}}$, el MPU completa la ejecución de la instrucción que está siendo procesada y después reconoce y atiende la señal $\overline{\text{NMI}}$. El bit de máscara de interrupción del CC - no tiene efecto en NMI.

El registro de índice, el contador del programa, los acumuladores y el registro de código de condiciones se almacenan en la pila. Después de esto, el contador del programa es -- cargado con el contenido de las direcciones de memoria FFFC y FFFD, así que la localidad de inicio de la rutina de atención a $\overline{NMI}$ debe estar almacenada en las direcciones antes - citadas.

Las entradas $\overline{IRQ}$ y $\overline{NMI}$ son líneas de interrupción por "hardware" que son muestreadas cuando la señal E tiene un estado alto y comienza la rutina de interrupción cuando la señal E tiene un estado bajo al término de la ejecución de una instrucción. $\overline{IRQ}$ y $\overline{NMI}$ deben de conectarse a 5 V si no son -- usadas.

La figura 4.3 muestra un diagrama de flujo de las interrupciones y la tabla 4.1 muestra los vectores de interrupción del MPU.

4.2.- MEMORIA DEL SISTEMA MEK 6802 -D5

Además de los 128 bytes de RAM que contiene el MPU MC --- 6802, el sistema MEK 6802-D5 contiene otros 128 bytes de RAM utilizados para el funcionamiento del sistema, y 2048 bytes - de ROM programados con el sistema operativo "D5BUG". Esta memoria ROM, que contiene un programa(conocido como "Monitor")- permite al usuario facilidades para desarrollar, verificar y corregir sus propios programas. Esto es posible gracias a que cuenta con rutinas que manejan el teclado y el despliegue así como a la entrefaz para cinta magnética.

Además el sistema cuenta con 2 conectores para RAM del tipo MCM2114P45, permitiendo tener 1024 bytes de RAM para el usuario. También cuenta con un conector para una memoria pro-- gramable ROM o EPROM. Como se nota en la tabla 4.2, esta memoria ROM o EPROM puede ser localizada en el tope superior delmapa de memoria si así se desea ; en esta posición, el sistema comienza con la operación del programa del usuario después del restablecimiento(reset). La ROM/EPROM del usuario debe - controlar todas las rutinas de interrupción,(todas las demás-

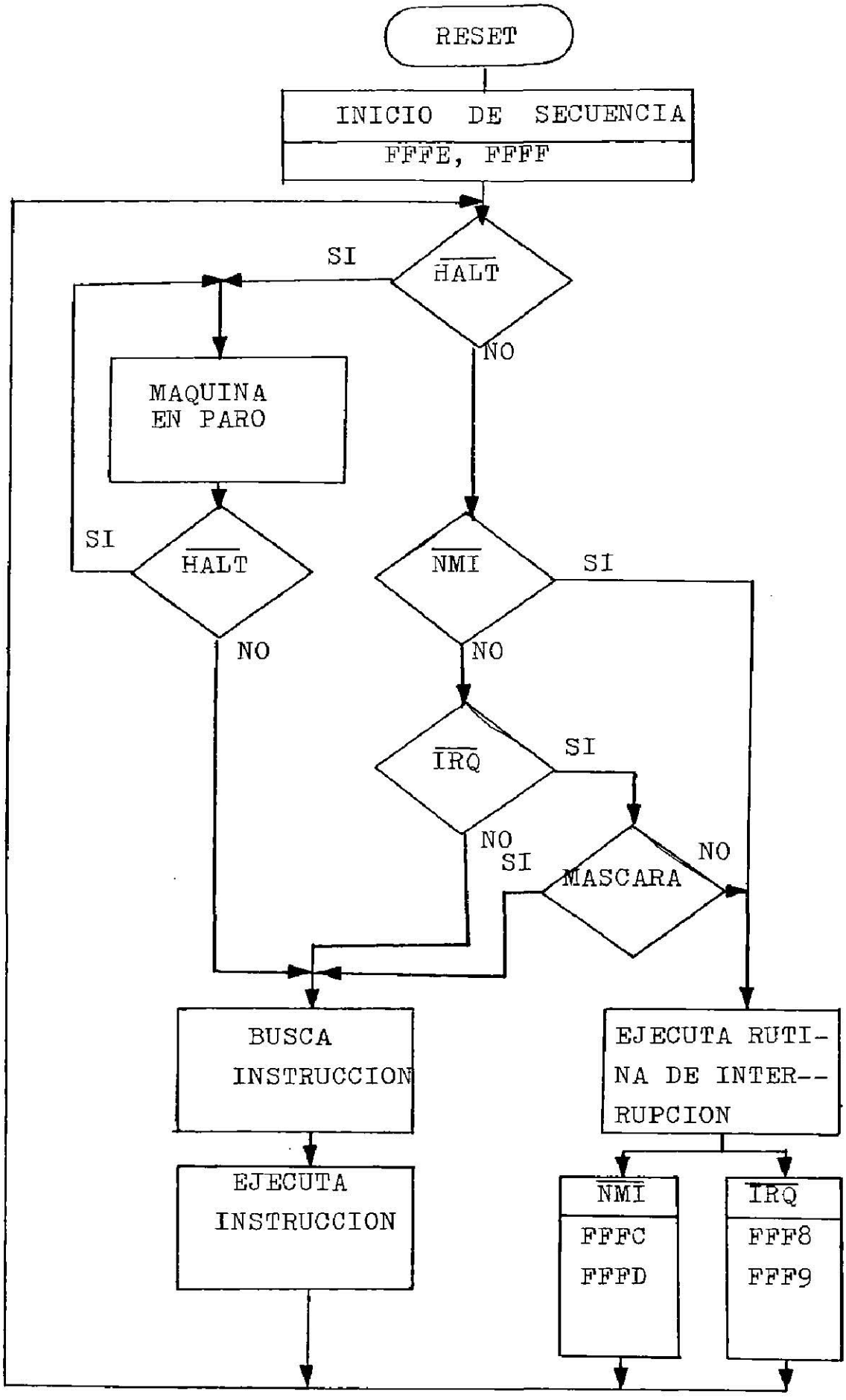


Figura 4.3 Diagrama de Flujo de Interrupciones

VECTOR		DESCRIPCION
BYTE ALTO	BYTE BAJO	
FFFE	FFFF	Restablecimiento
FFFC	FFFD	Interrupción No Enmascarable
FFFA	FFFB	Interrupción de Software
FFF8	FFF9	Solicitud de Interrupción

Tabla 4.1 Mapa de Memoria de los Vectores de Interrupción

rutinas del ROM "D5BUG" deben permanecer disponibles).

La otra opción que tiene el usuario es de colocar su ROM- o EPROM inmediatamente después de la posición de la ROM del- sistema operativo "D5BUG", de esta forma, después del resta-- blecimiento, el sistema comienza con la operación del sistema operativo y el programa del usuario puede ser llamado por me- dio de un comando del teclado.

La capacidad de memoria puede ser expandida por medio de- tabletas adicionales, esto requiere la instalación de buffers para las líneas de dirección, datos y control. Cuando esta -- opción se coloca, la RAM del MPU puede deshabilitarse, esto - se hace fácilmente cambiando un "puente" en la tableta.

La entrefaz para cinta magnética representa un dispositi- vo de mayor almacenamiento el cual complementa la memoria del sistema esta entrefaz la permite al usuario almacenar sus pro- gramas desarrollados para usarlos posteriormente.

FFFF FC00	Opción del Usuario	Como está proporcionado, "D5BUG"responde a estas direcciones. Cambiando un "puen-- te", la ROM/EPROM del usuario puede estar localizada aquí.
FBFF F800	Opción del Usuario	Como está proporcionado, la memoria no -- responde a estas direcciones. Cambiando - un "puente", la ROM/EPROM del usuario pue de ser localizada aquí.
F7FF F000	ROM de "D5BUG"	Sistema Operativo(ROM del programa monitor)
EFFF E800	ROM del Usuario	ROM/EPROM del usuario. Si esta opción se- selecciona, "D5BUG" sigue respondiendo al vector de restablecimiento.
E7FF E702		Reservado para el sistema

E701 E700	ACIA Opcional del Sistema	Permite la conexión de una entrefaz -- RS232C para la conexión de un teletipo. En este caso se tendrá que quitar el - teclado. Este puerto serie no estará - soportado por el monitor "D5BUG".
E6FF E488		Reservado para el sistema
E487 E484	PIA del Sistema	El MC 6821 se usa exclusivamente para- el teclado, el despliegue y la entre- faz para cinta magnética
E483 E480	PIA del Usuario	El MC 6821 está completamente disponi- ble para el usuario.
E47F E400	RAM del Sistema	El MC 6810 en esta posición proporcio- na 128 bytes para banderas, datos y á- rea de pila para el sistema operativo- D5BUG. 24 bytes de esta RAM son propor cionados para la pila del usuario.
E3FF E000	RAM del Usuario Opc.	RAM del usuario(1024 bytes) instalando 2 pastillas MCM2114
OFFF 0080	Memoria Externa	El sistema puede expanderse por medio- de memoria externa y módulos de I/O
007F 0000	Memoria Interna ó Externa	Como está implementado, los 128 bytes- de RAM del MC 6802 responden a esta di- rección. Esta RAM debe deshabilitarse- si se usa memoria externa.

Tabla 4.2 Mapa de Memoria del Sistema MEK 6802 -D5

RAM DE 1024 x 4 (MCM 2114)

La memoria MCM 2114 es una memoria de lectura/escritura-- de 1024 palabras de 4 bits fabricada con tecnología NMOS de - alta densidad. Para facilidad del usuario, esta memoria opera con una sola fuente de alimentación(+5 V), es directamente --

compatible con dispositivos TTL y no requiere de señales de - "refrescamiento" ya que es estática. Los datos de salida tienen la misma polaridad que los datos de entrada.

Esta memoria cuenta con una terminal de selección de pastilla($\overline{S}$) la cual permite seleccionar un circuito integrado -- cuando varios esten conectados a un mismo ducto de datos. La- figura 4.4 muestra un diagrama a bloques de esta memoria.

Cuando esta memoria no está direccionada, sus líneas de - datos(I/O) presentan alta impedancia.

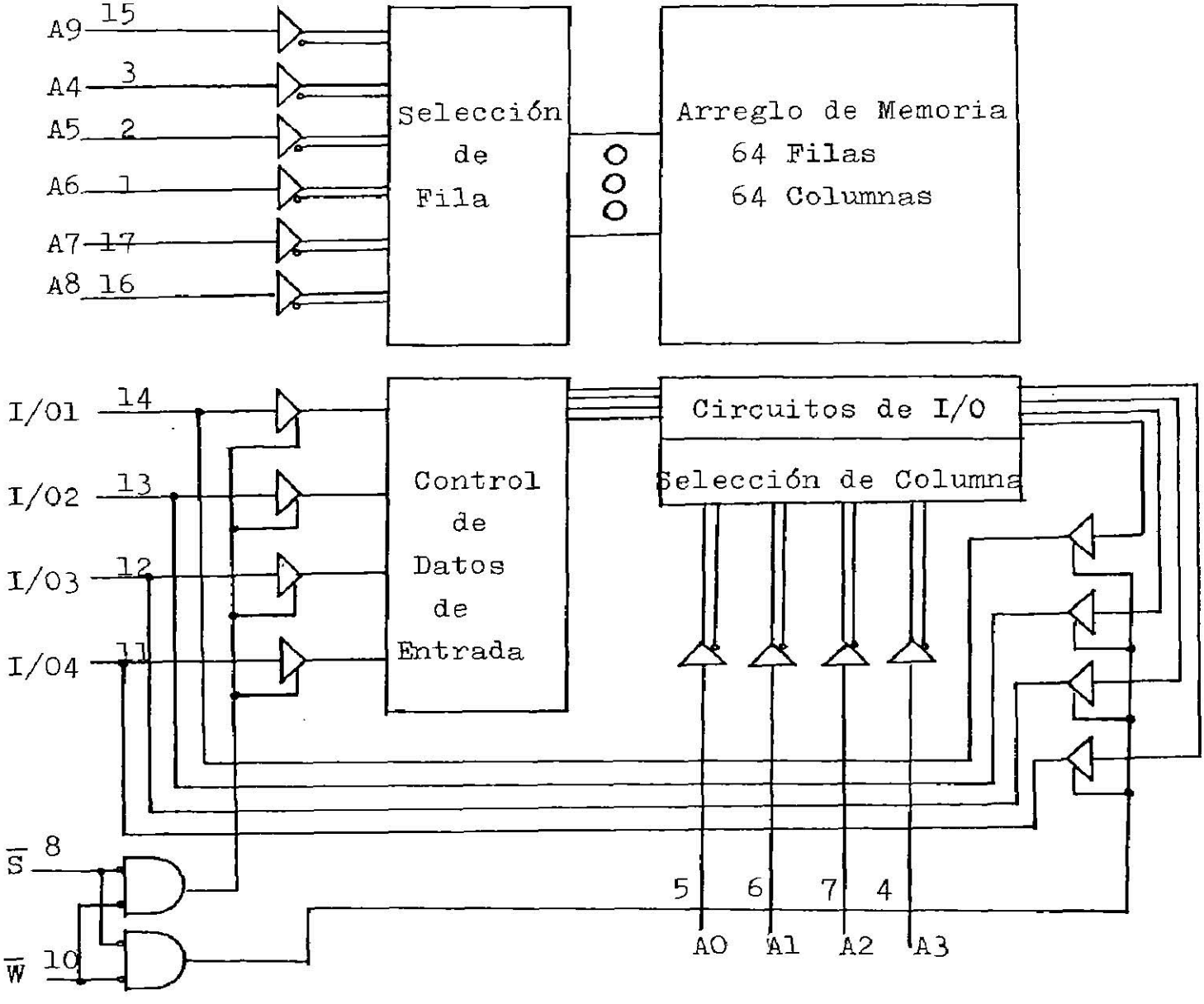


Figura 4.4 Diagrama a Bloques de la Memoria MCM 2114

RAM DE 128 x 8 (MCM 6810)

La memoria MCM 6810 es una memoria de lectura/escritura - de 128 palabras de 8 bits fabricada con tecnología NMOS, sus- salidas tienen la capacidad del tercer estado(alta impedancia) Como se puede apreciar en la figura 4.5, además de las líneas de datos y las líneas de direcciones tiene 6 líneas de selección, las que conectan al ducto de dirección del MPU, de tal- manera que una sola memoria sea direccionada a un solo tiem-- po.

Para direccionar una memoria de este tipo se requiere que se conecte un nivel bajo en 4 líneas de selección($\overline{CS1}$, $\overline{CS2}$, - $\overline{CS4}$, $\overline{CS5}$) y un nivel alto en 2 líneas de dirección(CSO, CS3). Para leer un dato de una localidad de memoria, la línea R/W - debe tener un nivel alto y para almacenarlo un nivel bajo. -- Cuando la memoria no está seleccionada, las líneas de datos - presentan una alta impedancia.

4.3.- DUCTOS DE INFORMACION

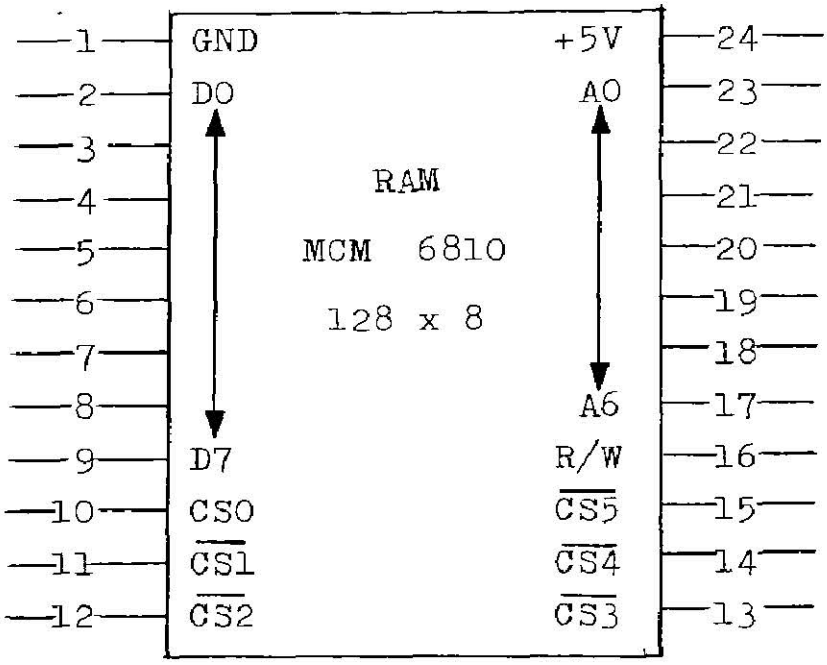
El microprocesador del sistema MEK 6802 direcciona a to-- dos los dispositivos del sistema por medio de un ducto de di- recciones de 16 bits, y transfiere los datos a través de un - ducto de datos de 8 bits.

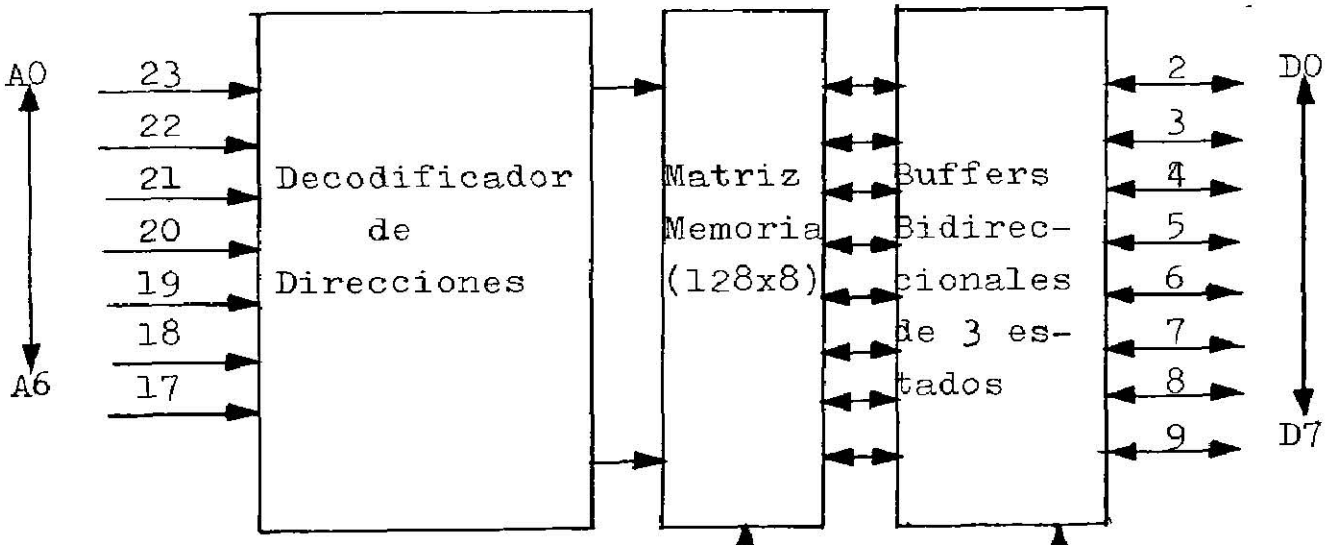
El MPU genera todas las señales que conduce el ducto de - direcciones. Como este es un ducto de 16 bits, se pueden pro- ducir direcciones de la 0000_{16} a la $FFFF_{16}$. Cuando el MPU - genera una dirección, solamente el dispositivo con esa direc- ción se comunicará con él. Los 16 bits son designados como -- AO hasta A15.

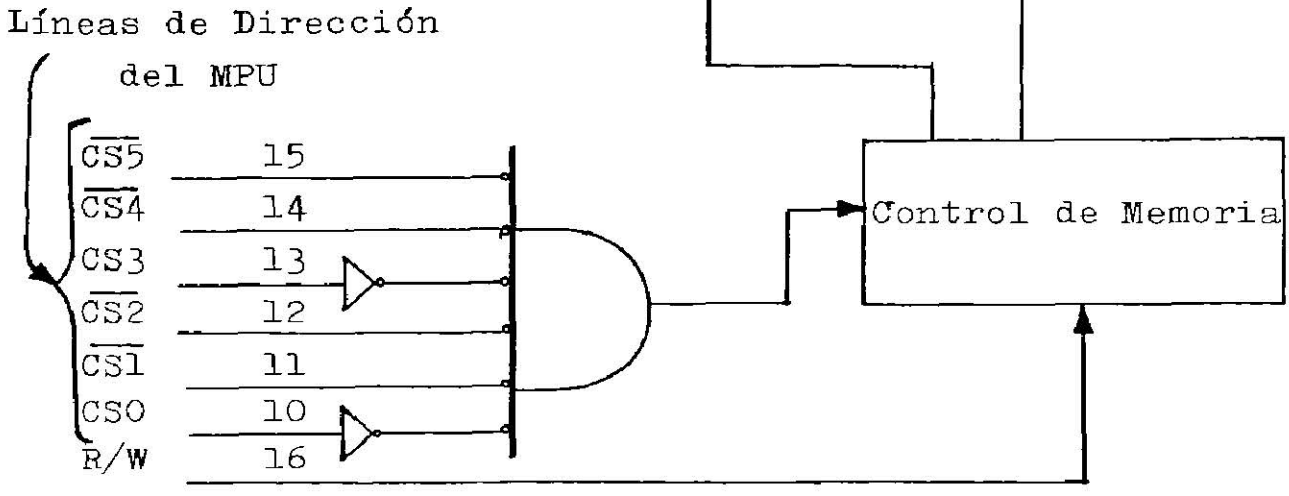
El ducto de datos de 8 bits es común a todos los disposi- tivos que integran el sistema. La figura 4.6 muestra un dia-- grama de bloques del sistema MEK 6802-D5, en el cual se ilus- tran los ductos de direcciones y de datos. Se puede observar- que la dirección de los datos fluye entre cada dispositivo y- el MPU MC 6802, como indican las flechas. El MPU puede leer - instrucciones y datos de la ROM/EPROM, pero no puede enviar--

datos a la ROM/EPROM. Sin embargo, las RAM, PIA y ACIA tienen la capacidad de enviar y recibir datos del MPU. Las líneas -- individuales del ducto de datos son designadas como DO hasta-D7.

Figura 4.5 Diagrama a Bloques de la Memoria MCM 6810







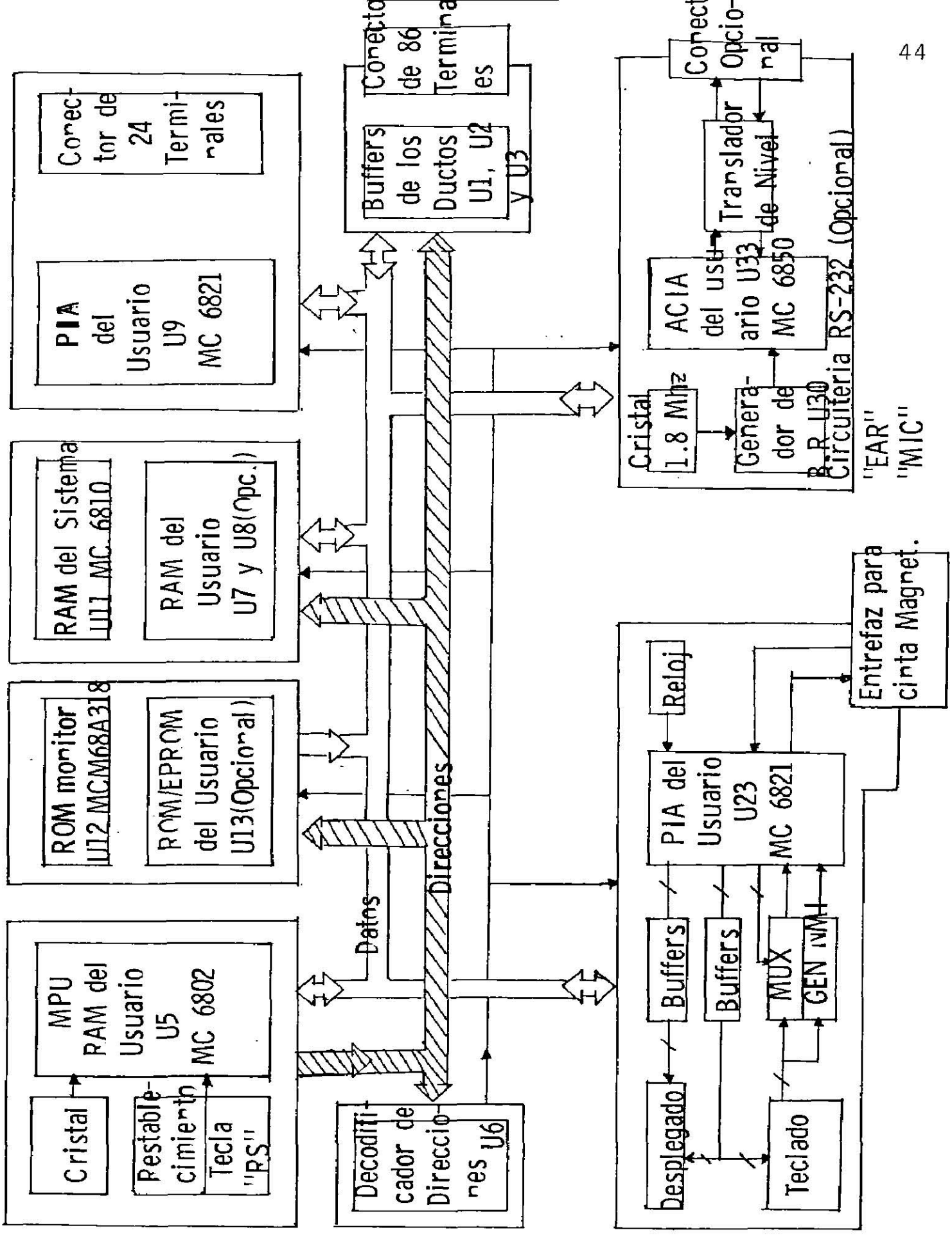


Figura 4.6 Diagrama a Bloques del Sistema MEK 6802 - D5

4.4.- ENTRADA Y SALIDA DEL SISTEMA

Todos los sistemas de microprocesadores deben proporcionar un medio para obtener datos del exterior del sistema, usualmente por medio de una o más pastillas llamadas Puertos de Entrada/Salida o Pastilla de Entrefaz. El MPU puede por programa, leer el estado de las líneas de entrada de esas pastillas o enviar algunos datos a sus líneas de salida.

El sistema MEK 6802 -D5 tiene un PIA(Entrefaz Adaptadora de Periféricos) para comunicarse con dispositivos externos que manejan datos en paralelo; también se tiene la opción de conectarse un ACIA(Entrefaz Adaptadora de Comunicación Asincrónica), que sirve para comunicar al MPU con dispositivos externos que manejan datos en serie.

4.4.1.- ENTREFAZ ADAPTADORA DE PERIFERICOS(PIA)

El PIA MC 6821 es un dispositivo NMOS encapsulado en un módulo de 40 terminales, usado como medio de interconexión de equipos periféricos y señales externas con el MPU. El PIA se comunica con el MPU a través del mismo ducto de datos bidireccional de 8 bits al que están conectadas las RAM y ROM. El PIA tiene 2 grupos de líneas de datos bidireccionales de 8 bits para interconectarse con el lado exterior. Estas 16 líneas pueden ser programadas para actuar como líneas de entrada o de salida, estas líneas se muestran en la figura 4.7.

Además de las 24 líneas de ductos de datos(8 hacia el MPU y 16 al exterior), el PIA tiene 3 terminales de selección de pastilla(CS0, CS1, $\overline{CS2}$), una terminal de restablecimiento ($\overline{RES}$), dos de interrupción($\overline{IRQA}$ y $\overline{IRQB}$), una de lectura/escritura(R/W), cuatro líneas de control(CA1, CA2, CB1,CB2), una terminal de habilitación(E), dos de selección de registro (RS0, RS1) y dos terminales de alimentación(+5V y GND), como se muestra en la figura 4.8.

El PIA MC 6821 tiene 2 partes llamadas Puertos(lado A y lado B). Cada lado tiene un registro de datos periféricos (PDR), un registro de dirección de datos(DDR), y un registro de control(CR).

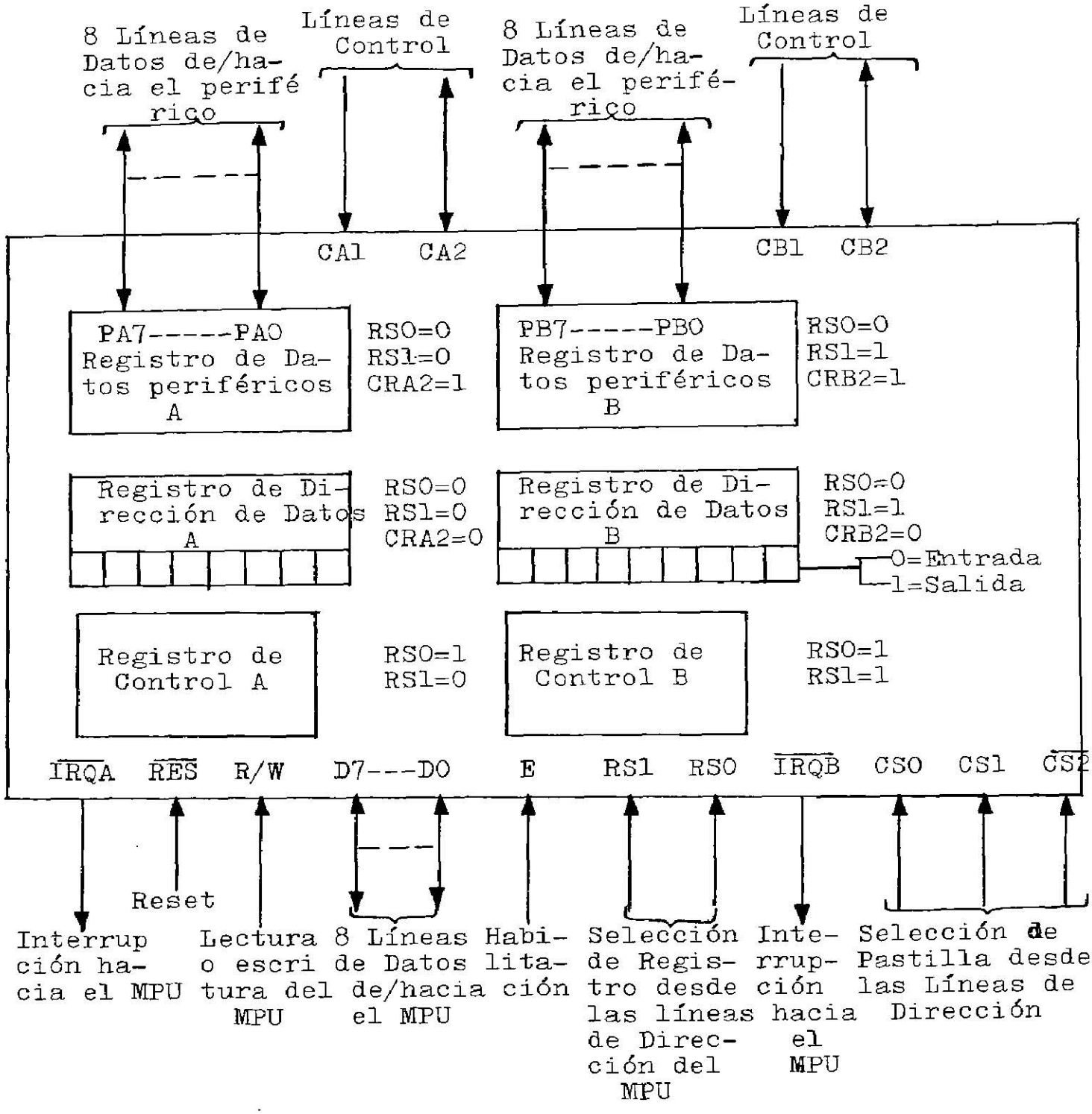


Figura 4.7 Entrefaz Adaptadora de Periféricos

Cada registro de datos periféricos sirve de entrefaz entre el PIA y el lado exterior. Este registro de 8 bits contiene - la información del estado de las líneas de datos periféricos.

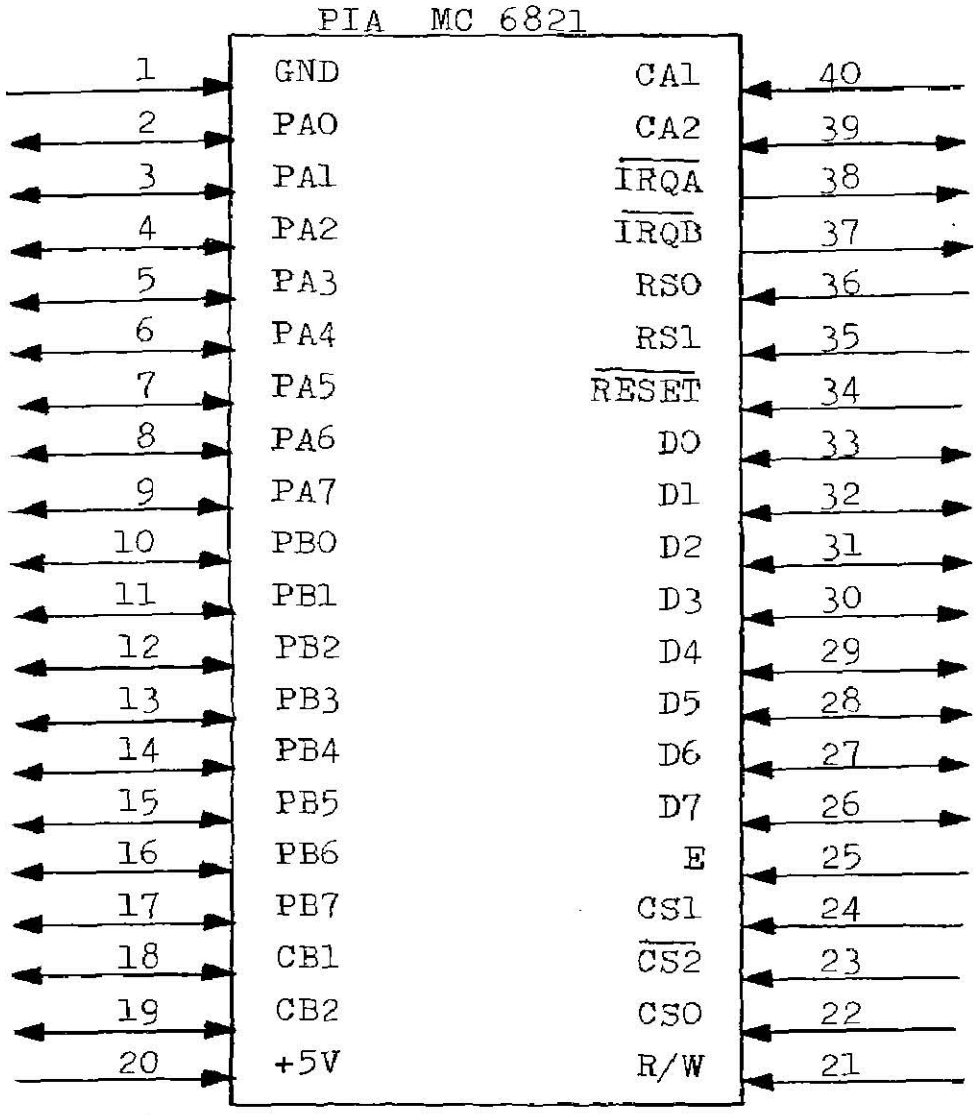


Figura 4.8 Terminales del PIA MC 6821

El registro de dirección de datos es usado por el programador para definir cada línea periférica como una línea de -- entrada o de salida. Cuando a un bit de este registro de 8 -- bits se le coloca un "1", la línea correspondiente del registro de datos periférico está definida como línea de salida y- cuando se le coloca un "0", está definida como entrada.

El registro de control se usa para permitir que el MPU -- controle la operación de las líneas periféricas CA1, CA2, CB1 y CB2. Este registro también se usa para controlar las líneas de interrupción y verificar el estado de las banderas de in-- terrupción. El bit 2 del registro se usa para determinar cuando se selecciona el registro de datos periférico o el regis--

tro de dirección de datos.

4.4.1.1.- SEÑALES DE ENTREFAZ DEL PIA

La figura 4.9 muestra un diagrama a bloques del PIA, en - él se muestran las señales de entrefaz que conectan al PIA -- con el MPU y con los dispositivos externos:

- Líneas de Datos Periféricos del lado A(PA0-PA7).- Cada una- de estas 8 líneas de datos periféricos, que interconectan - al MPU con dispositivos externos al sistema, pueden ser programadas para actuar como una línea de entrada o de salida. Durante la operación de lectura de las líneas de datos periféricos realizadas por el MPU, el dato de las líneas periféricas programadas como entradas aparecen directamente en -- las líneas del ducto de datos del MPU.
  En otra forma, cuando el MPU ejecuta una instrucción de "almacenar" un ducto en el registro de datos periféricos A(PDR A), el dato aparece en las líneas de datos periféricos que- fueron programadas como salidas. Un "1" lógico escrito en - el PDR A causa un nivel alto en la línea de dato periférica correspondiente y un "0" lógico, causa un nivel bajo.
  El dato de las líneas periféricas, que fueron programadas - como salida, puede ser leído por el MPU, leyendo el registro de datos periférico A, si el voltaje de la línea periférica es mayor de 2 volts, se lee como un "1" lógico, y si - es menor de 0.8 volts, se lee como un "0" lógico. Si las líneas periféricas son "cargadas", puede suceder oue el dato- leído por el MPU sea diferente al dato almacenado en el registro de datos periféricos A.
- Líneas de Datos Periféricos del lado B(PB0-PB7).- Las 8 líneas de datos, que interconectan el lado B del PIA con el - exterior del sistema, pueden ser programadas para actuar -- como entradas o salidas en forma similar al lado A. Los "Buffers" de salida que manejan a estas líneas tienen la capacidad del tercer estado, permitiendo ponerlas en estado de- alta impedancia cuando la línea de dato periférica se usa - como una entrada. El dato de las líneas periféricas progra-

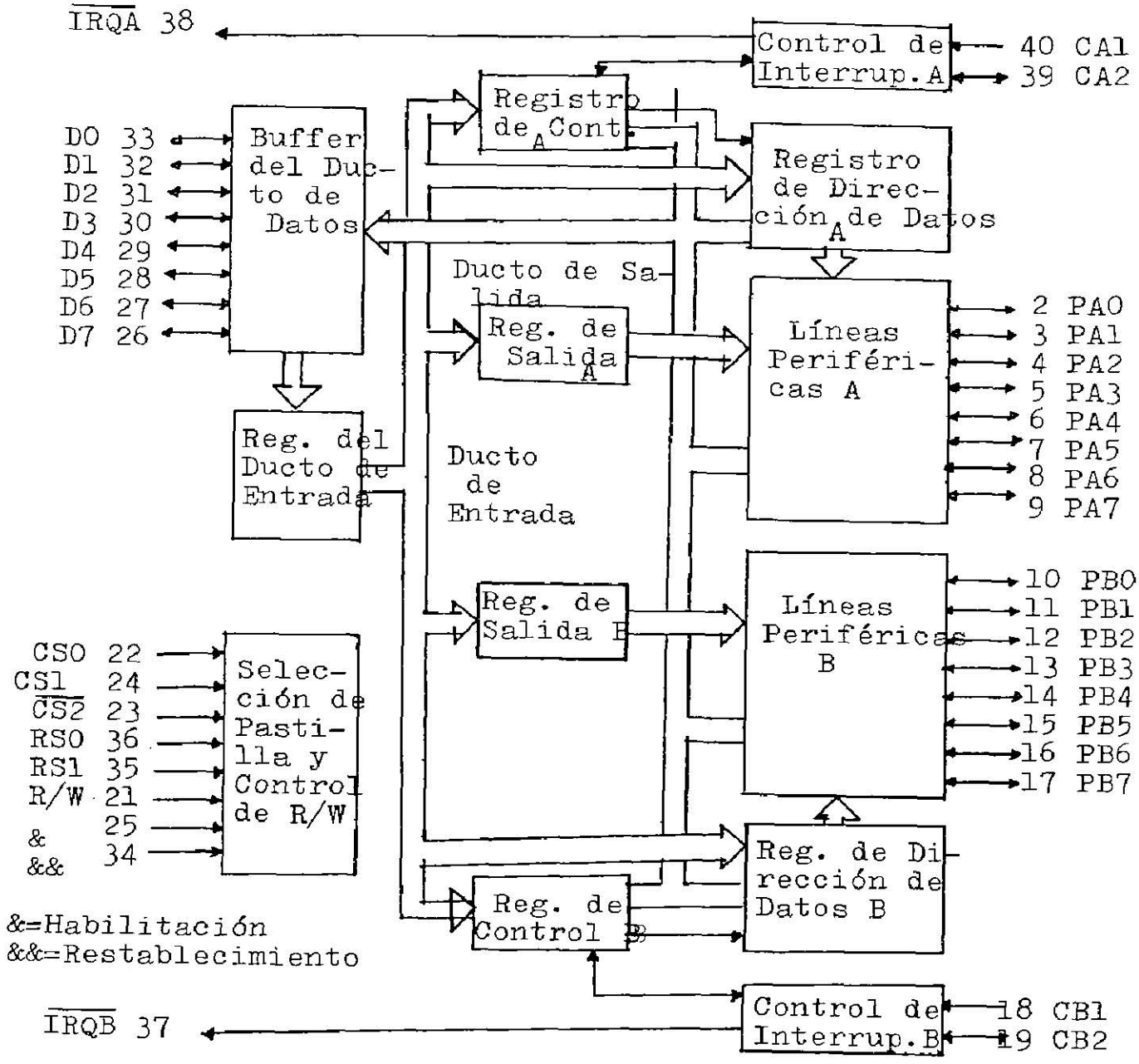


Figura 4.9 Diagrama a Bloques del PIA MC 6821

madas como salidas puede ser leído por el MPU, leyendo el registro de Datos Periférico B, si el voltaje de la línea es mayor que 2 volts, se lee como un "1" lógico, y si es menor de 0.8 volts, se lee como un "0" lógico. A diferencia del lado A, si las líneas periféricas del lado B son "cargadas", el dato leído por el MPU será el mismo almacenado en el registro de Datos Periféricos B.

- Líneas de Datos Bidireccionales(D0-D7).- Las 8 líneas bidireccionales de datos permiten la transferencia de datos entre el MPU y el PIA. Los "Buffers" de salida del ducto de datos del PIA son dispositivos de tres estados que permanecen en el estado de alta impedancia excepto cuando el MPU ejecuta una operación de lectura del PIA.

- Líneas de Selección de Pastilla(CS0, CS1, $\overline{CS2}$).- Estas 3 líneas de entrada se usan para seleccionar al PIA. Para esto CS0 y CS1 deben tener un estado alto y $\overline{CS2}$ un estado bajo.- Una vez que el PIA ha sido direccionado por el MPU, en las de selección se deben mantener los estados mencionados durante el pulso de habilitación(E), el cual es la única señal de temporización proporcionada por el MPU al PIA. Este pulso de habilitación(E) es normalmente la señal Ø2 del reloj. Una de las líneas de selección debe ser la salida de una compuerta "Y", la cual tendrá en sus entradas una línea de dirección y la línea VMA del MPU.

- Línea de Habilitación del PIA(E).- El pulso de habilitación (E) del MPU sirve de señal de sincronización al PIA y por tal razón debe de conectarse a esta línea de habilitación del PIA.

- Línea de Restablecimiento($\overline{RES}$).- Esta línea, la cual cuando está activa coloca a todos los registros del PIA a cero, primordialmente se usa durante un reinicio o una operación de encendido. Esta línea de restablecimiento tiene normalmente un nivel alto. La transición negativa(nivel alto a nivel bajo) restablece a todos los registros del PIA, causando que las líneas PA0-PA7, PB0-PB7, CA2 y CB2 funcionen-

como entradas y que se deshabilitan las interrupciones.

- Líneas de Lectura/Escritura(R/W).- Esta señal generada por el MPU controla la dirección de los datos que se transfieren en el ducto de datos. Un nivel bajo en esta línea de entrada habilita a los "Buffers" de las líneas del ducto de datos del PIA y se efectúa la transferencia del dato desde el MPU hacia el PIA(operación de escritura) en la transición negativa de la señal E. Un nivel alto en la línea de R/W prepara una transferencia hacia el ducto de datos del MPU(operación de lectura).

- Líneas de Solicitud de Interrupción($\overline{IRQA}$ y $\overline{IRQB}$).- Estas líneas del PIA, que generalmente están conectadas a las líneas $\overline{IRQ}$ o $\overline{NMI}$ del MPU , son el medio con que cuenta el PIA para solicitar un servicio de interrupción. Cuando las líneas $\overline{IRQA}$ o $\overline{IRQB}$ tienen un nivel bajo, pueden causar una interrupción al MPU.
  Cada línea de solicitud de interrupción tiene asociados dos banderas que le puede colocar un nivel bajo. Cada bit de bandera está asociado con una línea particular de interrupción periférica. Además el PIA cuenta con 4 bits de habilitación de interrupción los cuales se usan para habilitar una interrupción de un dispositivo externo.
  El servicio de atención a una interrupción al MPU, puede ser acompañado por una rutina de "software" que en una forma priorizada, secuencialmente, lee y prueba los dos registros de control del PIA, para saber que líneas de interrupción periférica fue activada y así saber que dispositivo externo está solicitando un servicio de interrupción.
  Los bits de las banderas de interrupción son borrados como un resultado de una operación de lectura del registro de datos periférico ejecutada por el MPU.

- Líneas de Entrada de Interrupción(CA1 y CB1).- Estas líneas sirven para producir interrupciones; las banderas de interrupción asociadas se ponen a "1" cuando se detecta la transición activa de estas líneas que es programada mediante los registros de control del PIA.

- Línea de Control Periférica(CA2).- La línea de control periférica CA2 puede ser programada para actuar como una entrada de interrupción o como una salida de control periférica. La función de esta línea se programa con el registro de control A.

- Línea de Control Periférica(CB2).- La línea de control periférica CB2 puede ser programada para actuar como una entrada de interrupción o como una salida de control periférica. Al actuar como una entrada de interrupción esta línea presenta una alta impedancia. La función de esta línea se programa con el registro de control B.

- Líneas de Selección de Registros del PIA(RS0 y RS1).- Estas dos líneas se usan en combinación con el registro de control para seleccionar uno de los registros internos del PIA que va a ser utilizado en una operación de lectura o escritura.

4.4.1.2.- MANEJO DE LAS ENTRADAS DE INTERRUPCION DEL PIA

En el sistema MEK 6802 - D5 las localidades de memoria con las cuales direccionan al PIA del usuario son de la E480 a la E483 y la asignación de estas localidades a los registros internos del PIA es la siguiente:

Localidad de Memoria		Asignación
E481	-	CRA(Registro de Control A)
E480	-	DDRA(Registro de Dirección de Datos A). (Bit 2 de CRA=0).
E480	-	PDRA(Registro de Datos Periférico A). (Bit 2 de CRA=1).
E483	-	CRB(Registro de Control B).
E482	-	DDRB(Registro de Dirección de Datos B). (Bit 2 de CRB=0).
E482	-	PDRB(Registro de Datos Periféricos B). (Bit 2 de CRB=1).
E43C	-	Vector de Interrupción IRQ.

Para programar las interrupciones se realiza la siguiente secuencia:

1.- Se habilitan los registros DDRA y DDRB como entradas o -- como salidas. Se coloca un "0" en la línea donde se quiera entrada y un "1" en la línea donde se desee salida.

2.- Se coloca un "1" en el bit 2 de los registros de control- CRA y CRB para accesar los registros de datos(PDR A y PDR B).

3.- Se colocan los bits 5, 4, 3, 1 y 0 de los registros de -- control CRA y CRB de acuerdo a las siguientes tablas:

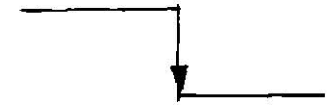
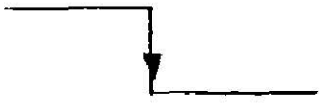
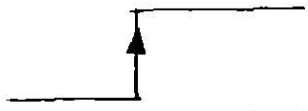
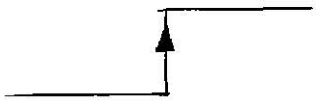
7	6	5	4	3	2	1	0
IRQA1	IRQA2	CA2	CONTROL		DDR A	CA1	CONTROL

CRA

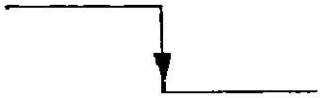
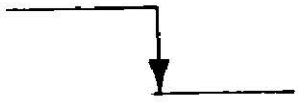
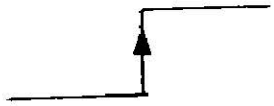
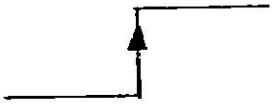
7	6	5	4	3	2	1	0
IRQB1	IRQB2	CB2	CONTROL		DDR B	CB1	CONTROL

CRB

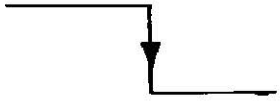
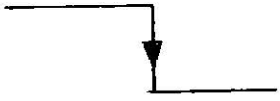
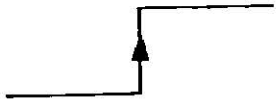
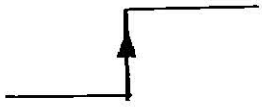
Control de la Línea de Interrupción CA1.

Transición de la Línea de Interrupción	Bit de CRA(1)	Bit de CRA(0)	Estado de Línea $\overline{IRQA}$
	0	0	Alto(máscara)
	0	1	Bajo(solicitud interrupción)
	1	1	Alto(máscara)
	1	1	Bajo(solicitud interrupción)

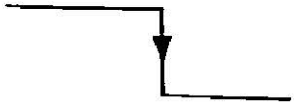
Control de la Línea de Interrupción CB1.

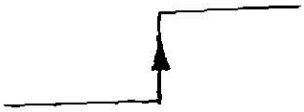
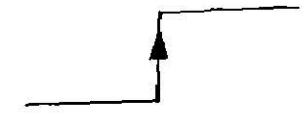
Transición de la Línea de Interrupción	Bit 1 de CRB	Bit 0 de CRB	Estado de Línea $\overline{IRQB}$
	0	0	Alto(máscara)
	0	1	Bajo(solicitud interrupción)
	1	0	Alto(máscara)
	1	1	Bajo(solicitud interrupción)

Control de la Línea de Interrupción CA2.

Transición de la Línea CA2	Bit 5 de CRA	Bit 4 de CRA	Bit 3 de CRA	Estado de la Línea $\overline{IRQA}$
	0	0	0	Alto(máscara)
	0	0	1	Bajo(solicitud interrupción)
	0	1	0	Alto(máscara)
	0	1	1	Bajo(solicitud interrupción)

Control de Línea de Interrupción CB2.

Transmición de la Línea CB2	Bit 5 de CRB	Bit 4 de CRB	Bit 3 de CRB	Estado de la Línea $\overline{IRQB}$
	0	0	0	Alto(máscara)

Transmición de la Línea CB2	Bit 5 de CRB	Bit 4 de CRB	Bit 3 de CRB	Estado de la Línea $\overline{IRQB}$
	0	0	1	Bajo(solicitud interrupción)
	0	1	0	Alto(máscara)
	0	1	1	Bajo(solicitud interrupción)

Programación de CA2 como Línea de Salida.

Para que CA2 actúe como línea de salida se coloca un "1"- en el bit 5 del registro de control CRA, de esta manera se -- tienen 4 opciones:

1.- Modo Protocolo(Handshake Mode)- Bits 5, 4, 3 de CRA=100

Este modo se usa cuando un periférico está transmitiendo- datos al MPU. El periférico debe decirle al MPU que un dato - está disponible para su lectura y el MPU debe contestarle al- periférico que ya tomó ese dato. La secuencia es como sigue:

a).- El periférico envía una señal por medio de la línea de - interrupción CA1 para colocar a "1" a la bandera IRQA1 - (bit 7) del registro de control A, con lo cual le dice - al MPU que hay un dato para él.

b).- Cuando la bandera IRQA1 es colocada a "1", la línea de - salida CA2 se coloca en un nivel alto.

c).- Después de que el MPU lee el contenido del registro de - datos A(operación de lectura del MPU) la línea de salida CA2 se coloca en un nivel bajo. Con esto, está avisado - el dispositivo periférico que el MPU tomó el dato y que está listo para recibir más datos.

2.- Modo Pulso- Bits 5, 4, 3 de CRA=101

En este modo, se dice al periférico que el dato contenido en el registro de datos A ha sido leído por el MPU, se utiliza cuando no es necesario un protocolo completo. El dispositivo periférico puede estar proporcionando datos continuamente al MPU, pero debe saber cuando éste toma los datos. Se produce un pulso invertido en la línea CA2 al mismo tiempo de la ejecución de una instrucción de lectura de registro de datos-A.

3.- Modo Bajo.- Bit 5, 4, 3 de CRA=110

En este modo, la salida CA2 siempre tiene nivel bajo.

4.- Modo Alto.- Bit 5, 4, 3 de CRA=111

En este modo, la salida CA2 siempre tiene nivel alto.

Programación de CB2 como Línea de Salida.

Para tener a la línea CB2 como línea de salida se coloca un "1" en el bit 5 del registro CRB, de esta manera se tienen 4 opciones:

1.- Modo Protocolo(Handshake Mode).- Bits 5, 4, 3 de CRB=100

Este modo se usa cuando el MPU envía datos a un dispositivo periférico, éste debe decirle al MPU que está listo para recibir el dato. Después de que el MPU envía el dato al registro de datos B, envía una señal al dispositivo periférico diciéndole que el dato está disponible en ese registro. Después de que el MPU envía el dato al registro de datos B, envía una señal al dispositivo periférico diciéndole que el dato está disponible en ese registro. Después de que el periférico toma el dato, puede pedir más datos, y la secuencia se repite. La secuencia típica se describe a continuación:

a).- Para decirle al MPU que quiere un dato, el periférico envía una señal por medio de la línea de interrupción CB1 para colocar a "1" la bandera IRQB1(bit 7) del registro CRB.

b).- Cuando la bandera IRQB1 es colocada a "1", la línea CB2 se coloca en nivel alto.

c).- Después de que el MPU envía el dato al registro de datos B, la línea CB2 se coloca en un nivel bajo, señalándole- al dispositivo periférico con esto que el dato está disponible para que lo tome.

2.- Modo Pulso.- Bits 5, 4, 3 de CRB=101

Este modo se usa para indicarle al periférico que el dato está disponible en el registro de datos B. Por CB2 se produce un pulso invertido al mismo tiempo de que se ejecuta una instrucción de escritura al registro de datos B.

3.- Modo Bajo.- Bits 5, 4, 3 de CRB=110

En este modo, la salida CB2 siempre tiene un nivel bajo.

4.- Modo Alto.- Bits 5, 4, 3 de CRB=111

En este modo, la salida CB2 siempre tiene un nivel alto.

Nota: Si se van a utilizar las líneas CA2 o CB2 como líneas - de salida, se deben programar con el modo bajo y des--- pués con el modo elegido, esto es necesario ya que inicialmente CA2 se encuentra en estado alto y CB2 en tercer estado(alta impedancia).

4.4.2.- ENTREFAZ ADAPTADORA DE COMUNICACION ASINCRONICA(ACIA)

4.4.2.1.- GENERALIDADES

En la sección anterior se mostró como el MPU se comunica- con dispositivos externos através del PIA, los datos son en-- viados o recibidos desde el MPU de 8 bits, a un mismo tiempo, es decir, en paralelo. Para este propósito, se necesitan co-- nectar 8 líneas entre el PIA y el dispositivo periférico.

Cuando es necesario enviar o recibir datos desde una distancia muy grande, las 8 líneas que deben conectarse entre el PIA y el dispositivo periférico deben tener esa longitud. El- ACIA(Asynchronous Communications Interface Adapter) permite - la transmisión de datos en forma serie a través de una sola - línea, funciona como un convertidor de serie a paralelo o ---

como un convertidor de paralelo a serie, figura 4.10.

Los datos se envían al ACIA por el ducto de datos(líneas-D0-D7), el ACIA los convierte en una serie de unos y ceros y-después los envía por una sola línea al dispositivo receptor. En el caso inverso, los datos en serie son recibidos por el - ACIA desde un dispositivo externo, los convierte a forma paralelo y los envía al MPU sobre el ducto de datos(D0-D7).

En cualquier tipo de transmisión de datos se utilizan dos términos: Síncrono y Asíncrono; esto es para referirse al tipo sincronización usado en dicha transmisión.

En una transmisión síncrona, el receptor y el transmisor-deben ser sincronizados el uno con el otro. Usualmente un dispositivo solicita un dato al otro, espera un período de tiempo fijo, y después lee el dato(asumiendo que el dato fue colocado en la línea durante el período de espera).

En una transmisión asíncrona, se agregan unos bits llamados de "inicio" y "paro" al dato para permitirle al receptor-saber cuando comienza y finaliza un dato. Después de que el - receptor detecta el bit de paro sabe que el dato transmitido-está completo y espera el bit de inicio del siguiente dato. - Los datos no estan dentro de un sistema sincronizado.

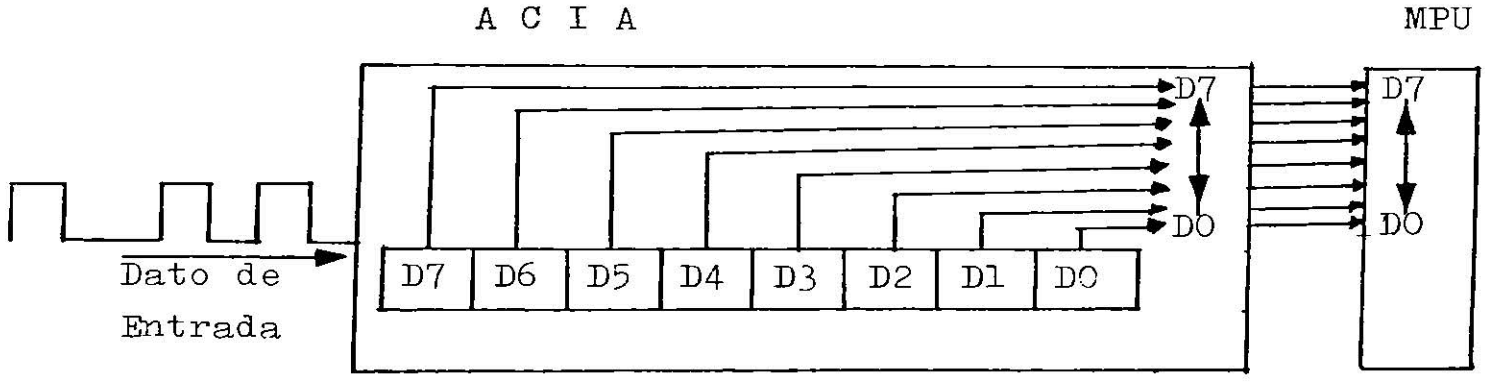
El término "razón de baud"(Baud Rate) se usa frecuentemente en la transmisión de datos en serie pero a veces es mal -- interpretado. Un "baud" está definido como el recíproco de la duración del pulso más corto de un dato y que incluye los bits de inicio, paro y paridad. Frecuentemente este término es considerado como "bits por segundo" el cual expresa sólo el número de bits de datos transferidos por segundo. A continuación-se definen algunos términos usados en la transferencia de datos en serie.

Bit de Inicio.- Es el primer bit de una palabra dato en serie que indica el comienzo de la transmisión de un conjunto de -- bits de datos. Este bit se detecta generalmente como una transición negativa de la línea de transmisión.

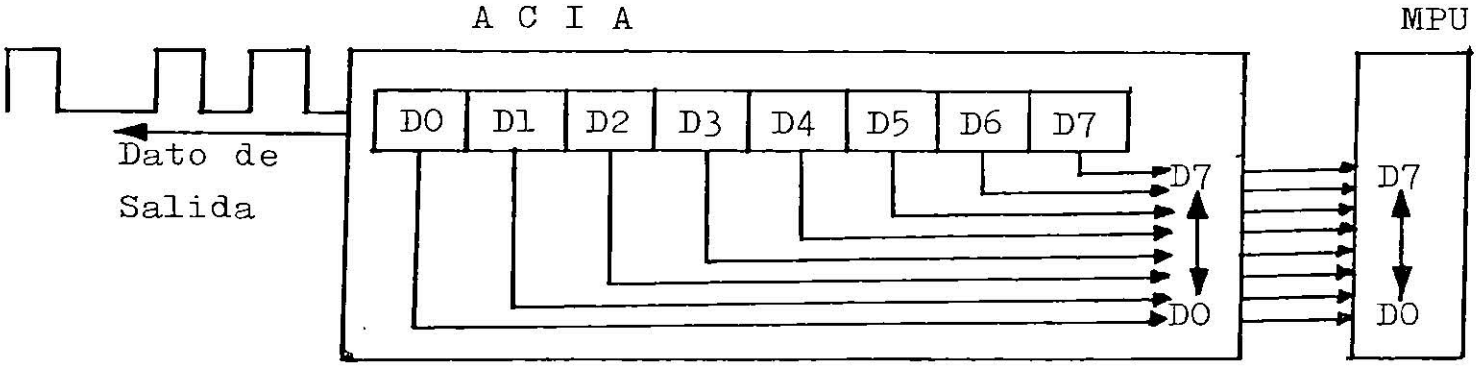
Bit de Paro.- Es el último bit de una palabra dato en serie - que indica el fin de esa palabra. Este bit tiene usualmente - un nivel alto.

Bit de Paridad.- Cuando se está transmitiendo una serie de -- bits, es común que el transmisor agregue lo que es conocido - como un "bit de paridad" a los bits del dato transmitido.

Se utilizan 2 tipos de paridad: paridad par e impar. Si - se usa la paridad impar, la suma de los ls transmitidos, in-- cluyendo el bit de paridad, debe ser impar. Por ejemplo, si - el dato contiene tres ls, el bit de paridad debe ser cero.



a).- Serie a Paralelo



b).- Paralelo a Serie

Figura 4.10 Funciones de Conversión del ACIA.

Si cuatro ls están en el dato, el transmisor debe agregar un "1" para el número total de ls transmitidos sea impar. El-

mismo principio se usa para paridad par, un "1" o un "0" se- debe colocar en el bit de paridad para que la suma de los 1s- transmitidos sea un número par.

El receptor, en ambos casos, verifica para asegurarse que se recibió el número correcto de 1s.

La función del bit de inicio, el bit de paro y de los bits de paridad está ilustrada en la figura 4.11. La relación de - la razón de baud, la razón de palabra de dato, y el número de bits de dato transmitidos por segundo se muestra en seguida:

Razón de Baud = 1/Tiempo del Bit = 1/9.09 mseg.= 110 Baud

Tiempo para transmitir una palabra =(11 bits)x(9.09mseg/bit)= 0.1 seg.

Razón de Palabra Dato = 1/0.1 seg. = 10 Palabra Dato/seg.

Razón de Dato = (10 Palabra Dato/seg.)x(8 Bits/Palabra Dato)
= 80 Bits/seg. (incluyendo el bit de paridad)

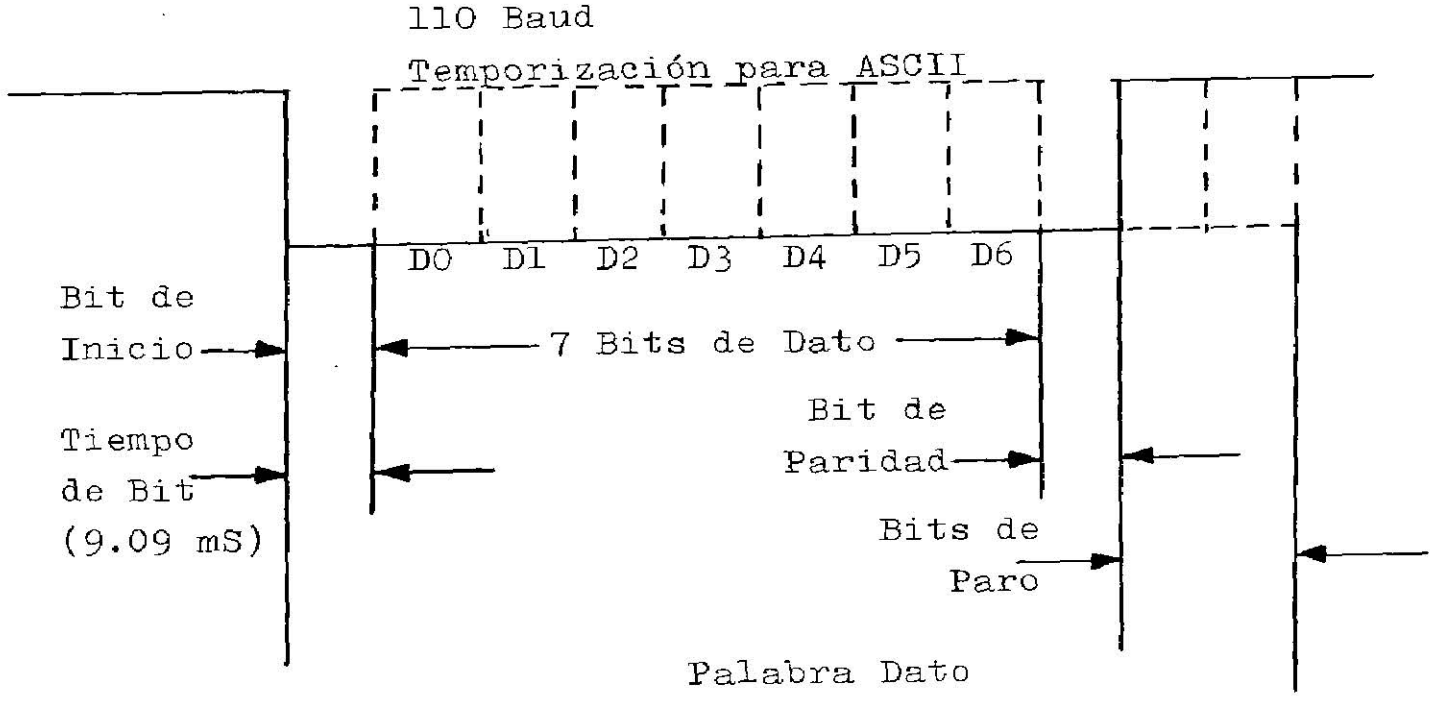


Figura 4.11 Funciones de los Bits de Inicio y Paro y Paridad

Notese que la razón de baud y el número de bits de datos-transmitido por segundo no son los mismos. La razón de baud - de 110 incluye los bits de inicio y de paro, además de los -- bits de dato, mientras que la razón de dato de 80 bits por segundo solo incluye los bits de información(dato) y el bit de-paridad.

En el ACIA existen varias opciones para la transmición de datos, como son el número de bits del dato(siete u ocho), paridad par o impar y el número de bits de paro(uno o dos). La-siguiente tabla muestra el tiempo del bit, el tiempo de la -- palabra dato, el número de palabra dato por segundo, y los -- bits por segundo del dato, para varios valores de razón de -- baud.

Razón de Baud	110	150	300	1200
Tiempo del Bit (mseg.).	9.09	6.66	3.33	0.833
+Tiempo de la Palabra Dato.	0.1 seg	0.73seg	0.0366seg	0.0092seg
Palabra Dato/seg.	10	13.7	27.32	108.7
Bits de Dato/seg.	80	110	218.6	870.

+ Asumiendo un bit de inicio, 8 bits de datos(incluyendo - un bit de paridad y dos bits de paro), en total once --- bits por palabra dato.

Tabla 4.3 Datos de Varios Valores de Razón de Baud

Como las señales digitales son transmitidas por una sola-línea, es posible realizar lecturas erróneas debido a ruido - en la línea. Para minimizar la probabilidad de esos errores,-se usa una técnica de muestreo que determina cuando es válido el bit de inicio. Después de que el bit de inicio ha sido considerado como válido, cada uno de los bits de la palabra dato

son muestreados aproximadamente en el centro del bit.

La técnica mencionada consiste de estar muestreando el -- bit de inicio varias veces para determinar si es válido y --- después muestrear cada bit con un pulso corto, aproximadamente, en el centro del bit disminuyendo con esto la probabili-- dad de error ya que el pulso del ruido debe de estar presente precisamente en el punto donde ocurre el muestreo. El muestreo se efectúa añadiendo una señal de reloj, en el caso del ACIA MC 6850, esta señal se proporciona internamente por medio de un divisor que puede programarse para lograr frecuencias - de 1, 16 o 64 veces la razón de baud. Estas frecuencias se conocen como los modos: 1, : 16, y : 64.

4.4.2.2.- DESCRIPCION DEL ACIA

El ACIA(MC 6850, Asynchronous Communications Interface -- Adapter), es un dispositivo NMOS, encapsulado en una pastilla de 24 terminales, se usa como un medio de recepción y transmisión de datos de 8 bits en un formato serie. El MPU se comunica(envía y recibe datos) con el ACIA por medio de 8 líneas bidireccionales(ducto de datos) tal como lo hace con las RAM, - ROM y PIA. Figura 4.12.

El ACIA tiene 4 registros que pueden ser direccionados -- por el MPU: el Registro de Estado(SR) y el Registro Receptor de Datos(RDR) son registros de "solo lectura". El Registro -- Transmisor de Datos(TDR) y el Registro de Control(CR) son registros de solo "escritura".

Además de estos 4 registros, el ACIA tiene 3 líneas de selección de pastilla(CS0, CS1,$\overline{CS2}$), una línea de selección de registro(RS), una línea de solicitud de interrupción($\overline{IRQ}$), -- una línea de habilitación(E), una línea de Lectura/Escritura (R/W), 8 líneas de datos(D0-D7) y 7 líneas de control(RXC, -- TXC, $\overline{DCD}$, $\overline{RTS}$, RXD, TXD, y $\overline{CTS}$).

Líneas Entre el MPU y el ACIA

- Líneas de Datos Bi-direccionales(DO-D7).- Estas 8 líneas de datos bidireccionales permiten la transferencia de datos -- entre el ACIA y el MPU.
  Las líneas de datos DO-D7 contienen "buffers" de tres estados que permanecen en un estado de alta impedancia excepto- cuando el MPU ejecuta una operación de lectura de algún registro del ACIA.

- Líneas de Selección de Pastilla(CSO, CS1, $\overline{CS2}$).- Estas líneas sirven para direccionar al ACIA y están conectadas al- ducto de direcciones. Para direccionar al ACIA, las líneas- CSO y CS1 deben tener un nivel alto y la línea $\overline{CS2}$, un nivel bajo; estos niveles deben de estar presentes durante el tiempo que tarda el pulso de habilitación(E), el cual es la única señal de temporización que proporciona el MPU al ACIA.

- Línea de Habilitación(E).- Esta línea de entrada es compatible con TTL y presenta una alta impedancia de entrada, sirve para habilitar los "buffers" de las líneas de entrada y- de salida del ACIA. Esta línea de entrada se conecta generalmente a la línea de la señal Ø2 del reloj del MPU.

- Línea de Lectura/Escritura(R/W).- Esta línea de entrada es- compatible con TTL y presenta una alta impedancia de entrada, se usa para controlar la dirección del flujo de información entre las 8 líneas de datos(DO-D7) del ACIA y el MPU.- Cuando la línea de lectura/escritura tiene un nivel alto -- (operación de lectura), se habilitan los "buffers" de las - líneas de salida del ACIA y el MPU puede entonces leer el - registro interno seleccionado del ACIA. Cuando la línea de- lectura/escritura tiene un nivel bajo(operación de escritura), se inhibe a los "buffers" de las líneas de salida del- ACIA y en ese momento el MPU puede efectuar una operación - de escritura en el registro seleccionado.

- Selección de Registro(RS).- La línea de selección de registro es compatible con TTL y presenta una alta impedancia de entrada, se usa para seleccionar, en forma conjunta con la- línea de lectura/escritura, los registros transmisor/recep-

tor de datos, o los registros de control/estados del ACIA, los cuales se muestran en la figura 4.13. Esta línea de entrada debe conectarse a una línea del ducto de direcciones del MPU. Un nivel alto en esta línea selecciona los registros transmisor/receptor de datos, un nivel bajo a los registros de control/estados.

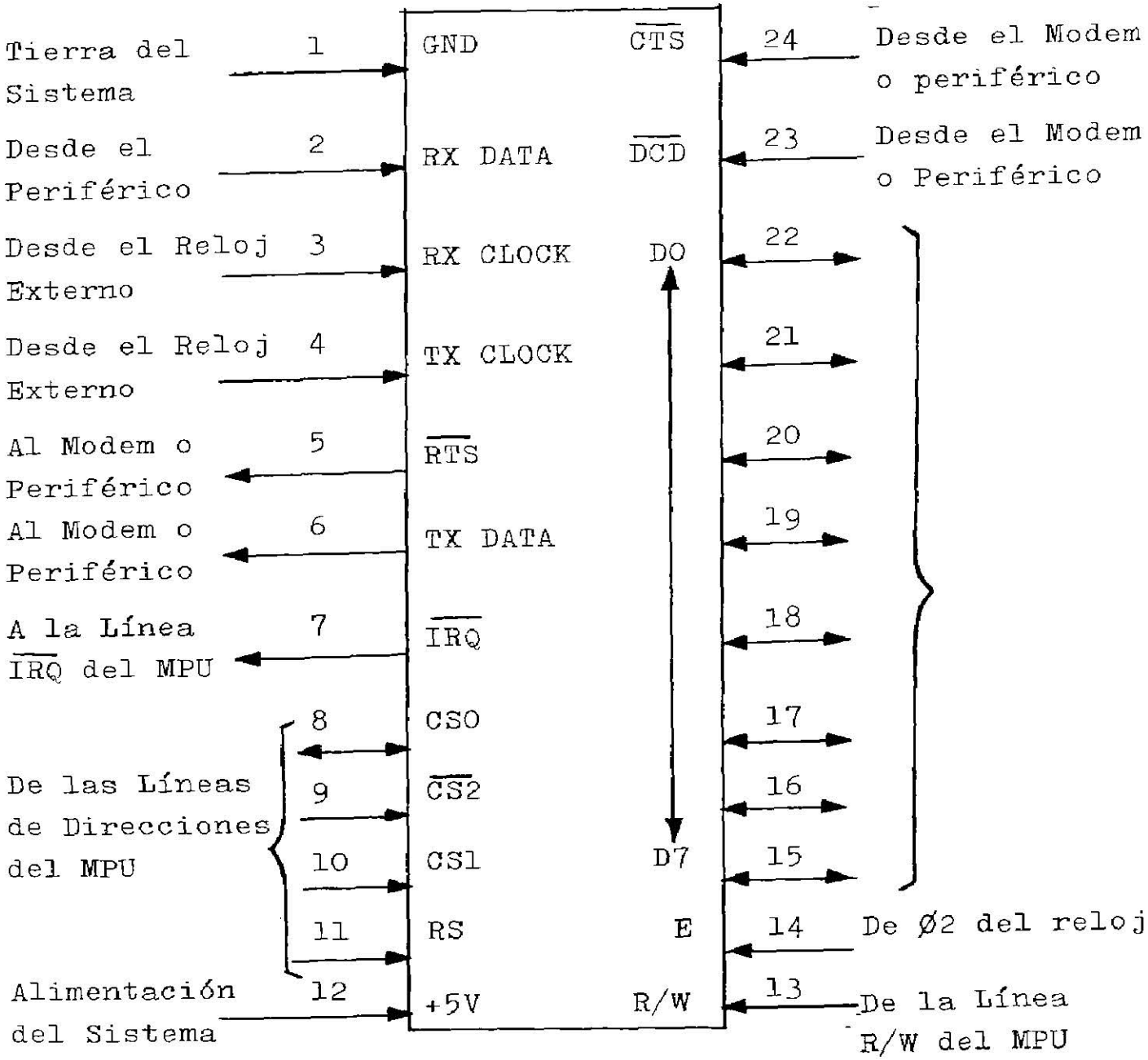


Figura 4.12 Terminales del ACIA MC 6850

- Solicitud de Interrupción($\overline{IRQ}$).- Esta línea es compatible con TTL y se activa con un nivel bajo, sirve para solicitar un servicio de interrupción al MPU. La solicitud de interrupción puede ser provocada tanto por la sección de recepción de datos como por la transmisión de datos, ambas colocan un "1" en el bit 7 del registro de estados cuando ocacionan una solicitud de interrupción.
  La sección de transmisión de datos ocaciona una solicitud de interrupción al término del envío del dato presente en el registro transmisor de datos siempre y cuando esté habilitada la solicitud de interrupción de transmisión en el registro de control.
  La sección de recepción de datos ocaciona una solicitud de interrupción cuando el registro receptor de datos se llena y el registro de control esté habilitada la solicitud de interrupción de recepción.
  La línea $\overline{IRQ}$ puede tener un nivel bajo también debido a una omisión de lectura del MPU por traslapamiento de datos en el registro receptor de datos(condición de "overrun").

Líneas de Control de un MODEM.

Los datos en serie que son transmitidos a través de líneas telefónicas deben ser enviadas primeramente a un MODEM(modulador-de-modulador) el cual los prepara para su transmisión. El ACIA proporciona tres líneas de control para un MODEM, las cuales se describen a continuación, figura 4.14

- Solicitud de Envío($\overline{RTS}$).- Esta línea de salida del ACIA la usa el MPU para solicitar al Modem un envío de datos. La señal $\overline{RTS}$ es de nivel activo bajo y su estado se determina por medio del registro de control.
- Limpiar para Envío($\overline{CTS}$).- Esta línea de entrada es compatible con TTL y presenta alta impedancia de entrada, se usa para detectar cuando el Modem terminó la transmisión de una cadena de datos, se activa con un nivel bajo proporcionado por la línea de salida $\overline{CTS}$ del Modem.

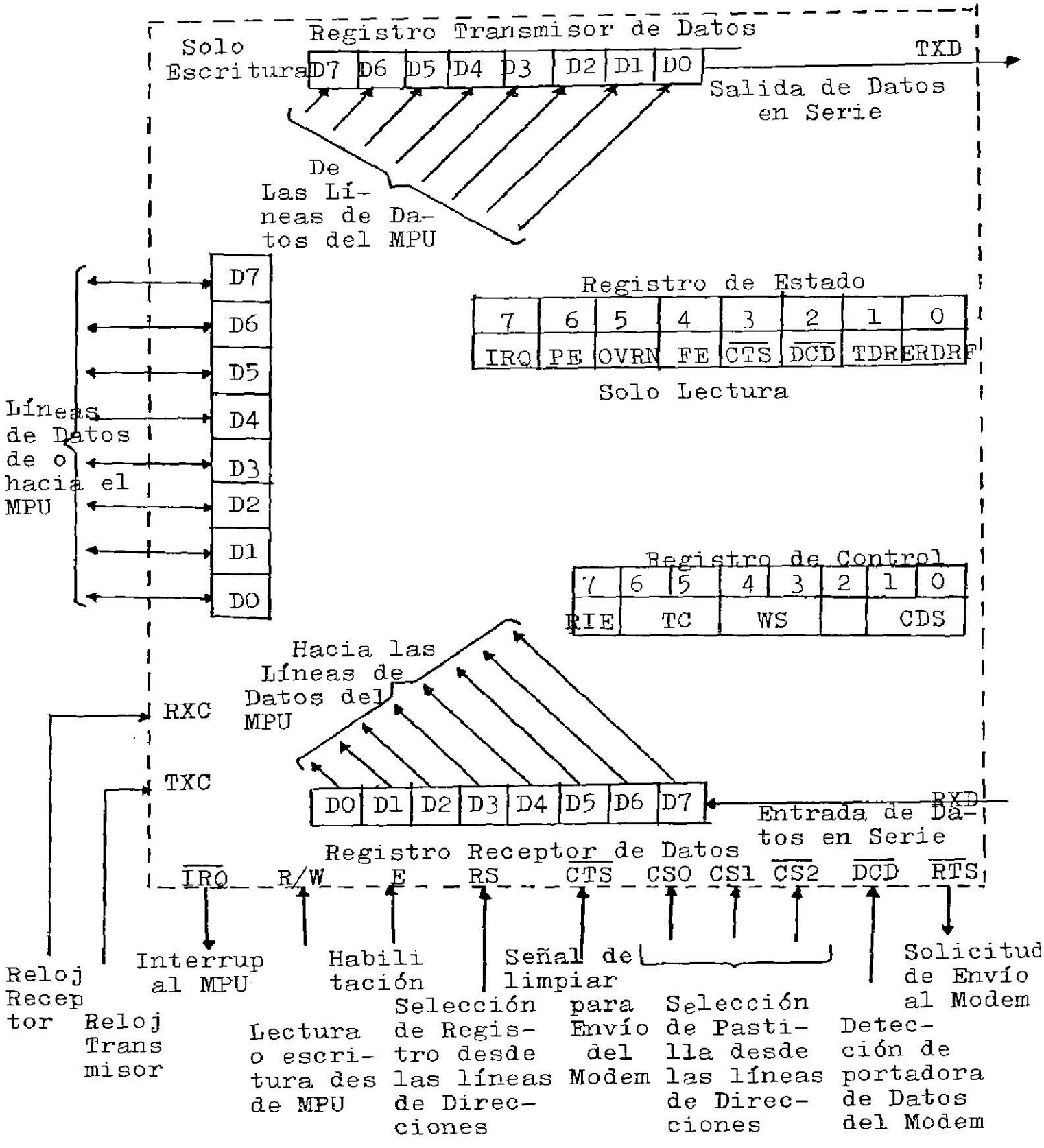


Figura 4.13 Líneas de Entrefaz del ACIA MC 6850

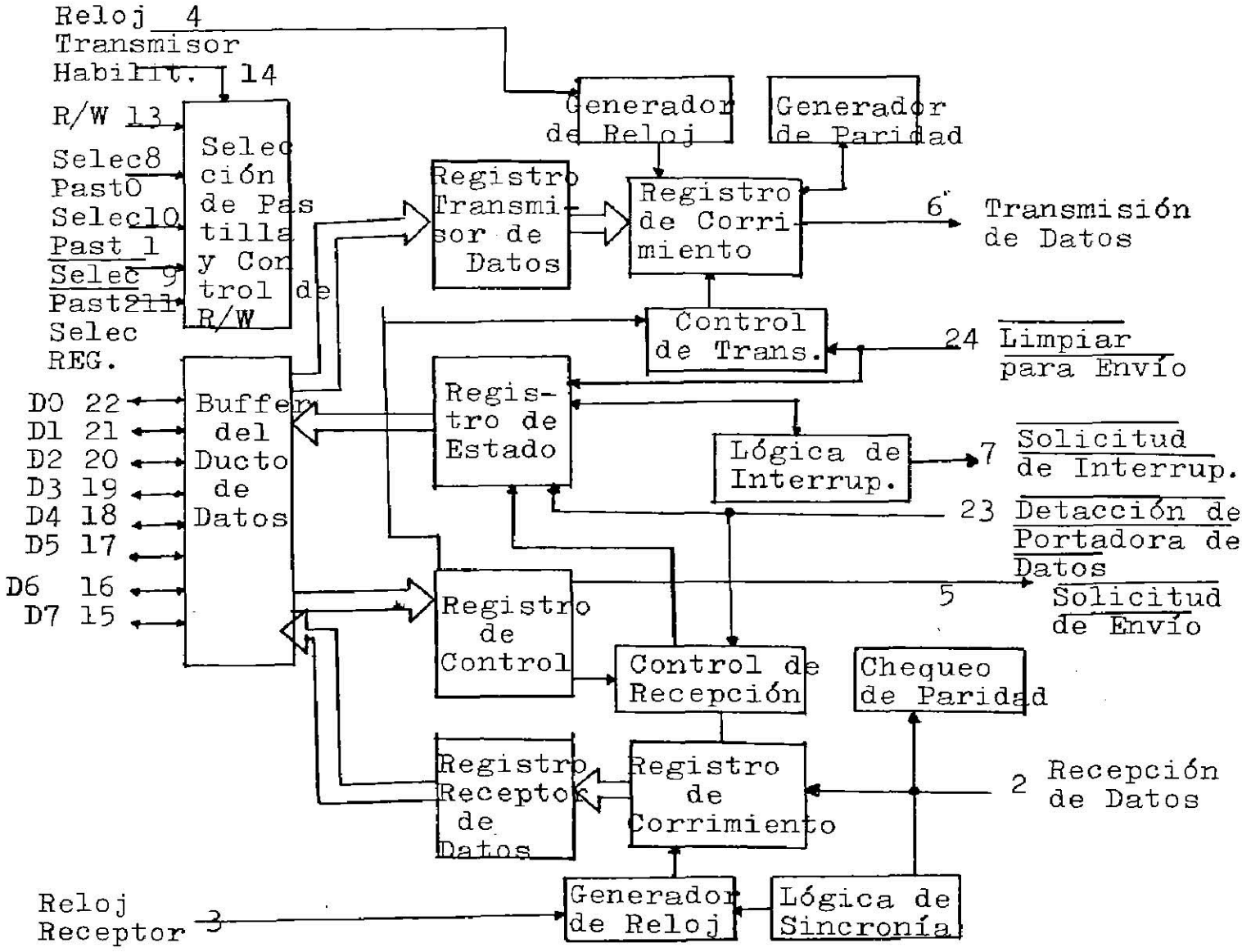


Figura 4.14 Diagrama a Bloques del ACIA MC 6850

- Detección de Portadora de Datos($\overline{DCD}$).- Esta línea de entrada es compatible con TTL y presenta alta impedancia de entrada, se usa para saber cuando el Modem terminó la recepción de una cadena de datos, normalmente está conectada a la línea de salida $\overline{DCD}$ del Modem y se activa con un nivel bajo. Cuando se le coloca un nivel alto, se inhibe e inicializa la sección de recepción de datos del ACIA, y con la transición positiva de esta señal se provoca una solicitud de interrupción al MPU siempre y cuando esté habilitada en el registro de control.

Líneas de Datos en Serie.

El ACIA tiene 2 líneas para la transferencia de datos en serie: la línea de transmisión de datos(TX DATA) que se usa para enviar, y la línea de recepción de datos(RX DATA) que se usa para recibir datos desde un dispositivo externo. Antes de enviar un dato, el ACIA le agrega el bit de inicio automáticamente. El número de bits de paro y la paridad, par o impar, se especifica en el registro de control(bits 2, 3 y 4).

Como el dato se recibe por una sola línea, el ACIA usa el bit de paridad para comprobar la exactitud del mismo número de ls recibidos; el ACIA quita el bit de inicio, el de paro y el de paridad de la palabra dato al recibirla, antes de convertirla a la forma paralela para transferirla al MPU por medio del ducto de datos.

- Línea de Recepción de Datos(RX DATA).- Esta línea de entrada es compatible con TTL y presenta una alta impedancia de entrada, se usa para recibir los datos en serie provenientes de un dispositivo externo. La recepción de datos con sincronización interna puede hacerse con razones de reloj de 16 o 64 veces la razón de dato. Con sincronización externa es posible tener unrango de 0 a 500,000 bits por segundo de razón de dato(utilizando el modo $\div$ 1).

- Línea de Transmisión de Datos(TX DATA).- Esta línea de salida se usa para transmitir datos en serie hacia un Modem o un dispositivo periférico externo. Es posible realizar una razón de dato de un rango de 0 a 500,000 bits por segundo con una sincronización externa(utilizando el modo ÷1).

Líneas de Entrada de Reloj.

El ACIA proporciona 2 líneas de entrada de reloj que es compatible con TTL y presenta alta impedancia de entrada, se utilizan para la sincronización de la transmisión y recepción de datos. Pueden ser seleccionadas frecuencias de reloj de 1, 16 o 64 veces la razón de dato.

- Reloj para el Transmisor(TXC).- Esta entrada se usa para sincronizar la transmisión de datos. La transmisión del dato se inicia en la transición negativa de la señal del reloj externo.
- Reloj para el Receptor(RXC).- Esta entrada se usa para sincronizar la recepción de los datos en serie. El receptor toma el dato en la transición positiva de la señal de reloj externo(utilizando el modo ÷1, la señal de reloj y el dato deben estar sincronizados externamente).

Registro Transmisor de Datos(TDR).

El Registro Transmisor de Datos es un registro de 8 bits usado para retener el dato(convertido de forma paralela a forma serie) hasta que sea transmitido. El dato se escribe en el registro transmisor de datos en la transición negativa de la señal de habilitación(E) después de que el ACIA fué direccionado por medio de las líneas CS0, CS1 y $\overline{CS2}$, y la línea RS tenga un nivel alto y la línea R/W un nivel bajo.

Si el transmisor está desocupado y no se está transmitiendo algún dato, entonces la transmisión del dato comienza un tiempo de duración de un bit después de la terminación del comando de escritura. Si un dato se está transmitiendo en el momento de la escritura, el nuevo dato comenzará su transmi-

sión tan pronto como termine la del dato previo.

Registro Receptor de Datos(RDR).

El Registro Receptor de Datos es un registro de 8 bits -- usado para retener el dato que fué transferido desde un Modem o dispositivo periférico externo hacia el ACIA. Después de -- que este registro esté lleno, el dato está listo para transferirlo al MPU por las líneas del ducto de datos; en estas condiciones, el registro provoca una solicitud de interrupción,- colocándole un nivel bajo en la línea de salida $\overline{IRQ}$, siempre- y cuando esté habilitada en el registro de control.

El registro receptor de datos mantiene su información después de una operación de lectura del MPU a este registro y solamente es cambiado cuando se recibe otro dato desde el Modem.

Registro de Estados(SR).

El Registro de Estados es un registro de 8 bits de solo - lectura que contiene información de las actividades internas- del ACIA. Para leer su contenido, el ACIA debe ser seleccionada por medio de las líneas CS0, CS1 y $\overline{CS2}$, la línea RS debe - tener nivel bajo y la línea R/W nivel alto. A continuación se detallan las funciones de los bits de este registro, figura - 4.15.

7	6	5	4	3	2	1	0
IRQ	PE	OVRN	FE	$\overline{CTS}$	$\overline{DCD}$	TDRE	RDRF

Figura 4.15 Registro de Estados

Bit 0.- Registro Receptor de Datos Lleno(RDRF).

"1": a).- Indica que el registro de datos está lleno.

b).- Si está habilitado, el bit IRQ se coloca "1" y -- así permanece hasta que se efectúe una lectura -- del RDR por el MPU.

"0": a).- Indica que el contenido del registro receptor de-datos fué leído por el MPU. El dato es retenido - en el registro.

b).- Si hay una pérdida de portadora, la línea $\overline{DCD}$ --- tendrá un nivel alto y este bit se colocará a "0" indicando que el contenido del RDR no es el co--- rrecto.

c).- Debido a un retablecimiento general.

Bit 1.- Registro de Transmisor de Datos Vacío(TDRE).

"1": a).- Indica que el contenido del registro transmisor - de datos fué transferido y que el registro está - listo para recibir más datos.

b).- Si está habilitado, el bit IRQ también se coloca- a "1" y así permanece hasta que el MPU realice -- una operación de escritura al TDR.

"0": a).- Indica que el registro transmisor de datos está - lleno.

b).- Cuando un nivel alto esté presente en la línea de entrada $\overline{CTS}$ colocará a "0" a ese bit.

Bit 2.- Detección de Portadora de Datos($\overline{DCD}$).

"1": a).- Indica que no hay señal portadora proveniente del Modem.

b).- Si está habilitado, el bit IRQ también se coloca- a "1" y así se mantiene hasta que el MPU ejecuta- una operación de lectura del registro de estados- y del registro receptor de datos o hasta que se - presente un restablecimiento del ACIA.

c).- Un "1" en este bit provoca que el bit RDRF sea -- colocado a "0", deshabilitando las posibles soli- citudes de interrupción del RDRF.

"0": a).- Indica que sí hay señal portadora proveniente del Modem.

Bit 3.- Limpiar para Envío($\overline{CTS}$).

"1": a).- Indica que la línea limpiar para envío proveniente del Modem tiene nivel alto, indicando con esto que el Modem no está listo para transmitir datos.

"0": a).- Indica que la línea limpiar para envío proveniente del Modem tiene nivel bajo, indicando con esto que el Modem está listo para transmitir datos.

Bit 4.- Error de Estructura(FE).

"1": a).- Indica que el formato de la palabra dato está incorrecto debido a los bits de inicio o de paro. - Este error detectado por la ausencia del primer - bit de paro, lo cual indica un error de sincronización, por falla en la transmisión, o por una -- "caida de línea". Este bit es colocado a "1" o a- "0" durante el tiempo de transferencia del dato - recibido y está presente durante todo el tiempo - que el dato asociado está disponible.

"0": a).- Indica que el formato de la palabra dato está correcto.

Bit 5.- Omisión de Lectura en el Receptor(OVRN).

"1": a).- Indica que uno o más datos de una cadena de datos fué perdido, esto es, un dato o un número de da-- tos fueron recibidos, pero no leídos del registro receptor de datos(RDR). La condición de omisión - de lectura comienza en la mitad del último bit -- del segundo dato recibido después de que no haya- ocurrido la lectura del RDR. La omisión de lectura no se indica en el registro de estados hasta - que el dato válido anterior a la omisión es leído. La bandera de la omisión es limpiada después de - la lectura del dato por parte del MPU, también es limpiada con un restablecimiento.

"0": a).- Indica que no ha ocurrido una omisión de lectura-del MPU.

Bit 6.- Error de Paridad(PE).

"1": a).- Indica que el número de ls del dato recibido no - está de acuerdo con la paridad preseleccionada. - La indicación de error de paridad está presente - todo el tiempo que permanezca el dato en el registro receptor de datos(RDR). Si no se selecciona - alguna paridad, la salida del generador de pari-- dad del transmisor y la parte del chequeo de paridad del receptor son deshabilitadas.

"0": a).- Indica que no hay error de paridad.

Bit 7.- Solicitud de Interrupción(IRQ).

"1": a).- Indica que hay una solicitud de interrupción la - cual ocacionó un nivel bajo a la línea de salida- $\overline{IRQ}$. La solicitud de interrupción se borra por -- una operación de lectura al RDR, por una opera--- ción de escritura al TDR, o por una operación de- lectura al SR, seguida por una operación de lectura al RDR si fué causado por $\overline{DCD}$. Un restableci-- miento general también borra a este bit.

"0": a).- Indica que no hay solicitud de interrupción.

Registro de Control (CR).

El registro de control(CR) es un registro de 8 bits usado por el MPU para controlar la transmisión y la recepción de -- datos en serie, figura 4.16. Para que el MPU pueda escribir - en el CR, el ACIA debe estar seleccionado por medio de las líneas CS0, CS1 y $\overline{CS2}$, y las líneas de RS y R/W deben tener nivel bajo.

Figura 4.16 Registro de Control

7	6	5	4	3	2	1	0
RIE	Control de Transmisión		Selección de Palabra			División del Contador	

Bit 0 y 1.- Bits de Selección de División del Contador(CDS).

Estos bits determinan las razones de división utilizada - en las secciones de transmisiones y recepciones del ACIA. También se usan para ejecutar un restablecimiento del ACIA, el - cual limpia el registro de estados(excepto para condiciones - externas de $\overline{CTS}$ y $\overline{DCD}$) e inicializa al receptor y al transmisor. El restablecimiento no afecta a los otros bits del registro de control. Después de una falla de energía o reinicio -- del sistema, el ACIA debe estar restablecido antes de colocar la razón de división de reloj. Las funciones de los bits CR0- y CR1 se muestran a continuación:

CR1	CR0	Función
0	0	÷1
0	1	÷16
1	0	÷64
1	1	Restablecimiento del ACIA

Bits 2, 3, 4.- Bits de Selección de Palabra(WS).

El programador tiene la opción de seleccionar la longitud de datos, el número de bits de paro, y el tipo de paridad,u-- sando la combinación apropiada de los bits 2, 3 y 4 del registro de control que a continuación se ilustran:

CR4	CR3	CR2	Longitud del Dato	+Paridad	+Bits de Paro
0	0	0	7	Par	2
0	0	1	7	Impar	2
0	1	0	7	Par	1
0	1	1	7	Impar	1
1	0	0	8	Ninguna	2
1	0	1	8	Ninguna	1
1	1	0	8	Par	1
1	1	1	8	Impar	1

Bits 5 y 6.- Bits de Control de Transmisor(TC)

El estado de los bits 5 y 6 del registro de control proporcionan el control para la solicitud de interrupción causada por la condición de registro transmisor de datos vacío(TDRE), para la salida de solicitud de envío($\overline{RTS}$), y para la transmisión de un nivel de ruptura(espacio), como se muestra adelante:

CR6	CR5	Función
0	0	La salida $\overline{RTS}$ tiene un nivel bajo y la solicitud de interrupción del transmisor se deshabilita. Este es el código usado cuando se solicita que el canal de comunicación sea colocado.
0	1	La salida $\overline{RTS}$ tiene un nivel bajo y el canal de comunicación es colocado. Además, este código se usa para generar solicitudes de interrupción por medio del bit TRDE del registro de estados.
1	0	La salida $\overline{RTS}$ tiene un nivel alto y la solicitud de interrupción del transmisor se deshabilita. Este código se usa para "tirar" el canal de comunicación.
1	1	La terminal $\overline{RTS}$ tiene un nivel bajo y una señal de ruptura se transmite. Este código se usa para provocar una interrupción en el sistema remoto.

Bit 7.- Habilitación de Solicitud de Interrupción del Receptor (RIE)

"1": a).- Habilita las solicitudes de interrupción causadas por:

- El bit de Registro Receptor de Dato Lleno(RDRF)
- Una transición positiva de la línea $\overline{DCD}$

"0": a).- Des-habilita las solicitudes de interrupción ocacionadas por el bit RDRF o por la pérdida de portadora de datos.

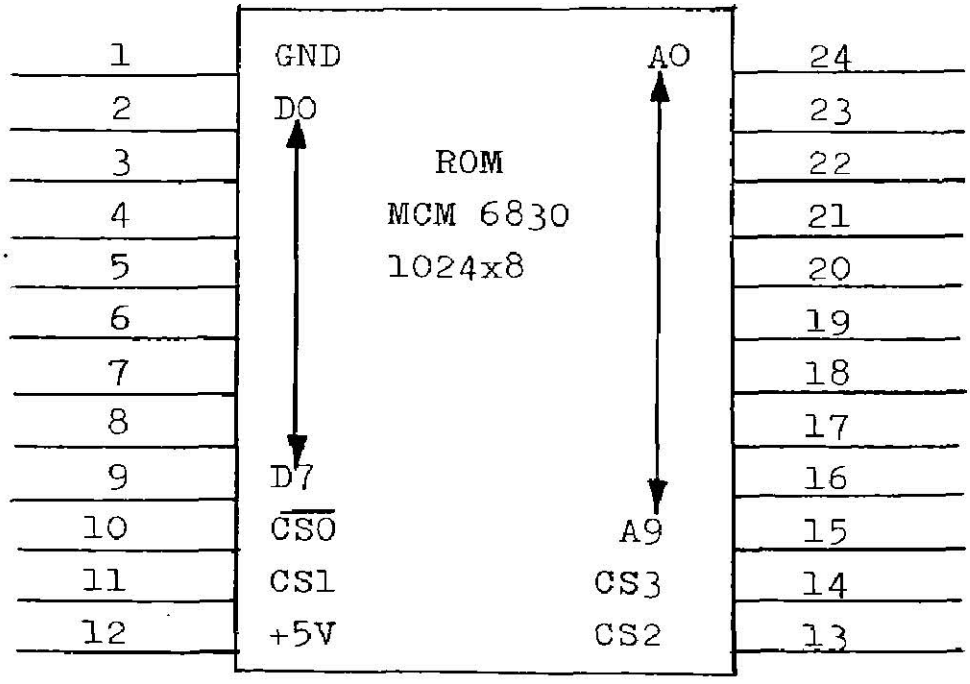
4.4.3.- RANDOM ACCESS MEMORY(RAM).

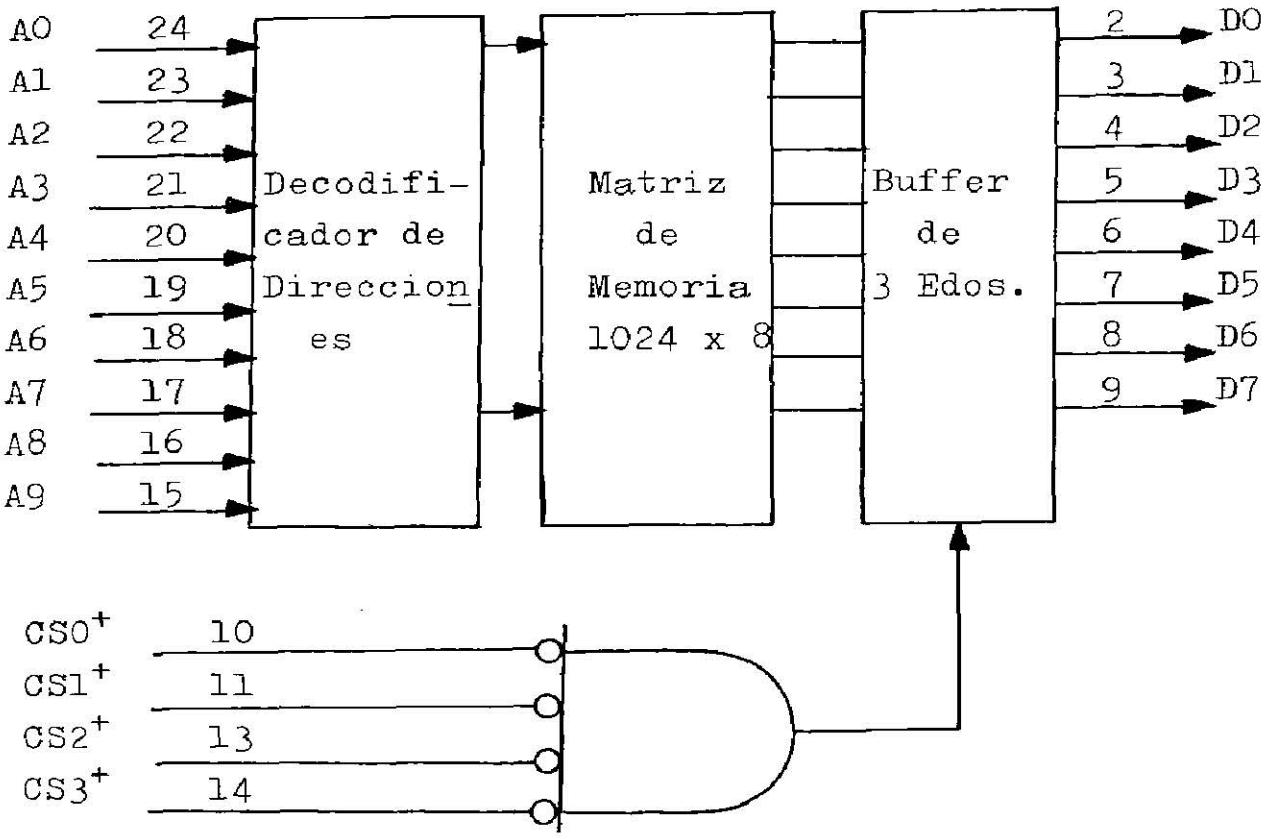
El MCM 6810 es un dispositivo NMOS de 3 estados, compatible con TTL conteniendo 128 palabras de 8 bits(128 bytes) alojados en paquetes de 24 terminales, figura 4.5. 8 terminales de ductos de datos(DO-D7), ligados a las terminales de ducto de datos del MPU(DO-D7). Las 7 terminales de dirección de la RAM(AO-A6) son ligados a las líneas de dirección del MPU(AO--A6). Estas 7 líneas son usadas por el MPU para seleccionar -- una palabra de 8 bits en un chip particular direccionado. Si hay más de una RAM MCM 6810 en el sistema todos los AOs son - ligados a la línea de dirección AO del MPU, todos los Als ligados a la línea de dirección Al del MPU, y así sucesivamente. La RAM también tiene 6 líneas de selección de pastilla. Esta selección de pastilla pueden ser ligados al ducto de dirección del MPU de tal manera que solo una pastilla o línea de la RAM es direccionada a un tiempo. Para direccionar una RAM se requiere de un nivel bajo en 4 de las líneas de selección de -- pastilla y un nivel alto en 2 de las líneas restantes. La línea R/W es la misma línea R/W del MPU discutida anteriormente. Para leer datos desde una localidad RAM, la línea R/W debe estar en estado alto; para almacenar datos en la localidad RAM, la línea R/W debe estar en estado bajo. Cuando una RAM no esdireccionada, el ducto de datos de la RAM vá a la condición - del tercer estado(alta impedancia).

4.4.4.- READ-ONLY MEMORY(ROM).

La ROM MCM 6830 es compatible con TTL. Es un dispositivo NMOS de tres estados conteniendo 1024 palabras de 8 bits(1024 bytes) alojadas en un paquete de 24 terminales, figura 4.17.- Las 8 líneas del ducto de datos(DO-D7), son ligados a las líneas del ducto de datos del MPU(DO-D7). Las 10 líneas de di-- cciones(AO-A9) van ligadas a las líneas del ducto de direc---

ción del MPU(AO-A9). Estas 10 líneas de dirección son usadas- por el MPU para elegir una palabra de 8 bits en un chip particular seleccionado. Si hay más de una ROM en el sistema, el - AO de cada ROM es ligado a la línea AO del MPU, el Al de cada ROM a la línea Al del MPU, y así sucesivamente. La ROM tiene- 4 líneas de selección de pastilla disponibles. Estas 4 líneas deben ser definidas por el usuario, positivas(CS) y negativas ($\overline{CS}$). Puesto que este dispositivo es mask-programmable, la línea CS puede ser manufacturada dentro del dispositivo. Pueden ser ligados en el ducto de dirección del MPU de tal manera -- que una ROM sea direccionada a un tiempo. Para seleccionar -- una ROM una señal de +2V(o mayor) debe ser aplicada a cada línea de selección de pastilla positiva(CS) y una señal de OV - para cada uno negativo($\overline{CS}$). Cuando la ROM no es direccionada- el ducto de datos de la ROM va a la condición de 3 estados -- (alta impedancia).





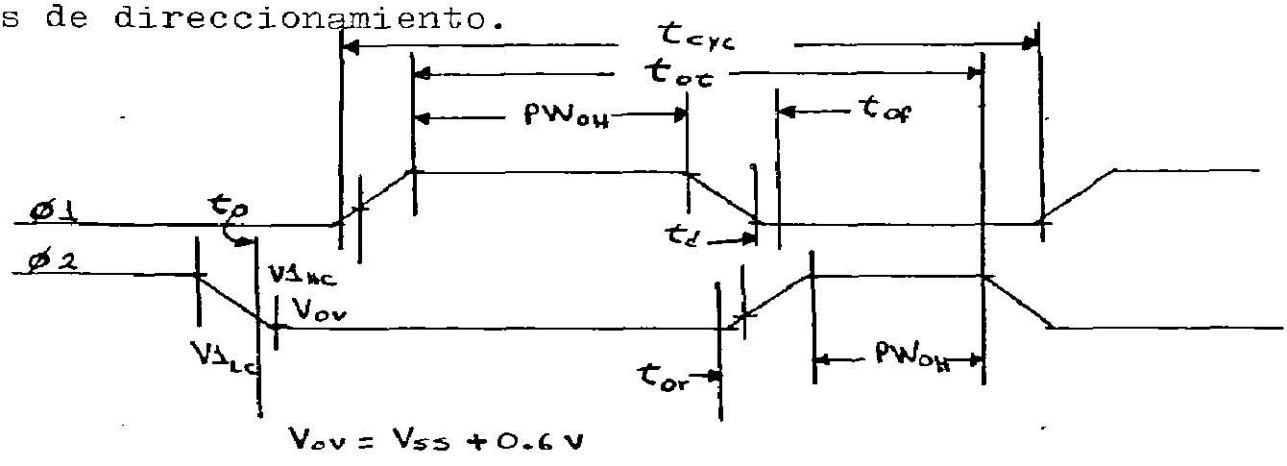
+ Niveles definidos por el usuario.

Figura 4.17 Diagrama a Bloques de la ROM MCM 6830

4.4.5.- RELOJ(CLOCK)

Todos los sistemas digitales deben tener una señal de reloj, de tal modo que las funciones puedan ser completadas a un tiempo, de manera ordenada. Este requiere de 2 fases de reloj capaces de operar desde un sistema de poder de 5V, figura 4.18

En el interior del MPU hay un registro llamado "registro de instrucción(IR)", cuyo propósito es decodificar la instrucción tal que el MPU la reconosca y pueda usarla en varios modos de direccionamiento.



Tiempo de Reloj	Minimo			Max.	Unidad
Tiempo de Ciclo	t_{cyc}	1.0	--	10	µs
Longitud de Pulso de Reloj (medido en Vcc - 0.6V)	PW_{oH}	400 400	-- --	9500 9500	ns
Tiempo Total Ø1 y Ø2	t_{Ot}	940	--	--	ns
Tiempo de Caida y Elevación (entre Vss-0.4V y Vcc-0.6V)	t_{Or}, t_{Of}		--	100	ns
Tiempo de Demora o Separación DE Tiempo(medido en-- Vov = Vss+0.6V)	t_d	0	--	9100	ns
Vcc=5V ± 5%, Vss=OV					

Figura 4.18 Especificaciones de Reloj

Cuando la fase 1(Ø1) de la señal de reloj es alta, el contenido del PC es transferido al ducto de dirección. Mientras este toma su lugar, el VMA puede estar alto(1), indicando una dirección válida. En la caída del borde del Ø1, el PC puede ser incrementado en 1. Cuando la fase 2(Ø2) es alta(asumimos que la señal de reloj Ø2 es usada como una línea de selección de pastilla o una capacitada), los datos de la localidad de memoria direccionada es colocada en el ducto de dirección, y durante la caída del borde del Ø2, los datos son encerrados en el MPU. Esta secuencia general, figura 4.19 ocurre en cada tiempo, el MPU direcciona una localidad de memoria y los datos son transferidos.

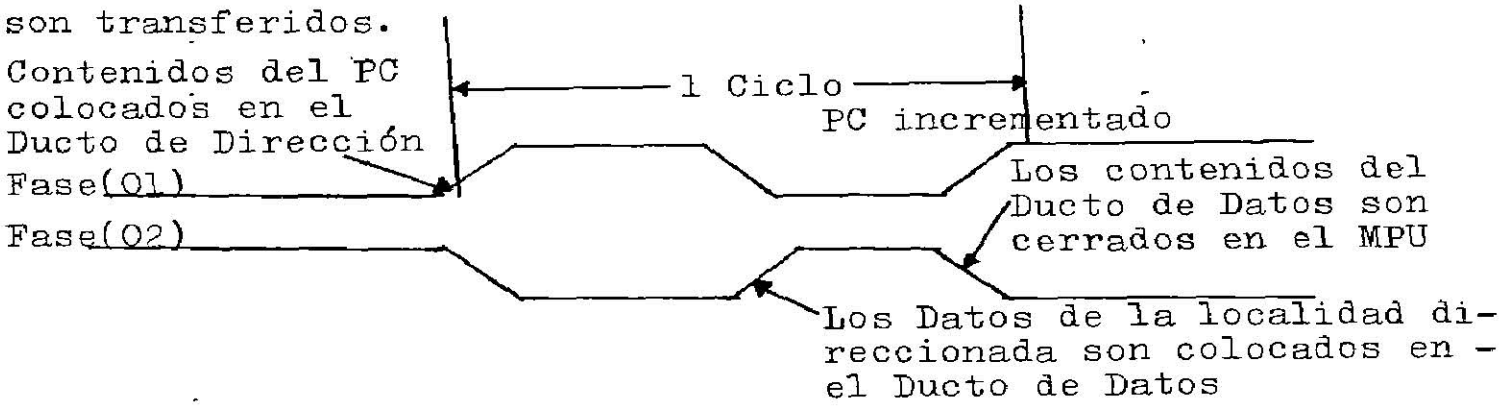


Figura 4.19 Instrucción del Registro de Secuencia

Para ilustrar la parte del sistema de reloj en la ejecución de instrucciones por el MPU, investigemos varias instrucciones de un programa simple y veamos que sucede durante cada ciclo de reloj:

Dirección	Contenido	Descripción
100	86	Carga A Inmediato
101	25	Datos
102	D6	Carga B Directo
103	37	Dirección
104	B7	STA A Extendido
105	40	Dirección
106	02	

- Las Direcciones 100 y 101 (2 ciclos)

Ciclo 1: a).- El contenido del contador P(100) es colocado en el ducto de dirección.

b).- El contador P es incrementado en 1 (a 101).

c).- El contenido de la dirección 100(86) es colocado en el ducto de datos, encerrados en el MPU el registro de instrucción(IR) y decodificada como una instrucción Carga A Inmediato (R/W y VMA son "1").

Ciclo 2: a).- El contenido del contador P(101) es colocado en el ducto de dirección.

b).- El contador P es incrementado en 1 (a 102).

c).- El contenido de la dirección 101(25) es colocado en el ducto de datos y encerrado en el Acumulador A(R/W y VMA son "1").

- Las Direcciones 102 y 103 (3 ciclos).

Ciclo 1: a).- El contenido del contador P(102) es colocado en el ducto de dirección.

b).- El contador P es incrementado en 1 (a 103).

c).- El contenido de dirección 102(D6) es colocado en el ducto de datos, encerrado en el MPU el registro de instrucción (IR) y decodificado -

como una instrucción Carga B Directo(R/W y-- VMA son "1").

Ciclo 2: a).- El contenido del contador P(103) es colocado en el ducto de dirección.

b).- El contador P es incrementado en 1 (a 104).

c).- El contenido de dirección 103(35) es colocado en el ducto de datos y encerrado en el -- MPU el ducto de dirección registrado(R/W y - VMA son "1").

Ciclo 3: a).- El contenido del ducto de dirección registrado (35) es colocado en el ducto de dirección.

b).- El contenido de la dirección 35 es colocado- en el ducto de datos y encerrados en el Acumulador B(R/W y VMA son "1").

- Las Direcciones 104-106 (5 ciclos).

Ciclo 1: a).- El contenido del contador P(104) es colocado en el ducto de dirección.

b).- El contador P es incrementado en 1 (a 105).

c).- El contenido de dirección 104(B7) es colocado en el ducto de datos, encerrado en el MPU el registro de instrucción(IR) y decodificada como una instrucción Almacena A Extendida (VMA y R/W son "1").

Ciclo 2: a).- El contenido del contador P(105) es colocado en el ducto de dirección.

b).- El contador P es incrementado en 1 (a 106).

c).- El contenido de dirección 105(40) es colocado en el ducto de Datos y encerrado en el -- MPU el ducto de dirección de registro(byte - más significativo)(VMA y R/W son "1").

Ciclo 3: a).- El contenido del contador P(106) es colocado- en el ducto de dirección.

b).- El contador P es incrementado en 1 (a 107).

c).- El contenido de dirección 106(02) es colocado en el ducto de datos y encerrada en el MPU el ducto de dirección registrada(byte menos significativo)(VMA y R/W son "1").

Ciclo 4: a).- El contenido del ducto de dirección registrada (4002) es colocado en el ducto de direc--- ción.

b).- El contenido del Acumulador A(25) es leído -- para transferirse al ducto de datos(VMA es -- "0" y R/W es "1").

Ciclo 5: a).- La dirección 4002 es accesada, la línea R/W - es colocada en estado bajo(escritura), VMA es "1" y el contenido del Acumulador A entra en- el ducto de datos y entonces los almacena en- la dirección accesada(4002).

4.5.- USO DE LA TARJETA MEK 6802 - D5

4.5.1.- GENERALIDADES

Se dan en este punto las instrucciones de operación de la tarjeta microcomputadora MEK 6802-D5 necesarias para que el - usuario pueda crear, correr y depurar los programas realiza-- dos con instrucciones de máquina.

El kit de evaluación de la microcomputadora MEK 6802-D5 - proporciona la circuitería (Hardware) y los programas residen tes en ROM (Firmware) necesarios para la formación de un sistema basado en el MPU MC 6802. Con este sistema el usuario -- puede realizar sus programas en lenguaje de máquina y tiene - la facilidad para expanderlo si así lo requiere.

4.5.2.- DESCRIPCION

La microcomputadora MEK 6802-D5 es un sistema diseñado -- para evaluar la familia de componentes del MPU MC 6800. Es -- útil en el desarrollo de aplicaciones de microprocesadores -- así como en el aprendisaje de técnicas de programación.

El sistema consiste de una pastilla MC 6802, un sistema - operativo en una ROM(2k bytes), 6 despliegues luminosos de 7- segmentos, un teclado de 25 contactos, un PIA y un entrefaz - estándard para cinta magnética. Tiene 2 bases para circuitos- integrados en los cuales se pueden conectar 2 pastillas de -- RAM para formar 1k byte, también tiene una base para pastilla de EPROM y 3 para los "buffers" de las líneas del conector de 86 terminales. La figura 4.20 muestra un esquema de los dis-- positivos del sistema MEK 6802-D5.

4.5.3.- COMPONENTES

EL MICROPROCESADOR MC 6802(U5)

El microprocesador genera la señal de reloj para todo el- sistema, ejecuta los programas del sistema operativo y tam--- bién los programas del usuario contenidos en la RAM del sis-- tema. El MPU contiene 128 bytes de RAM que pueden ser utiliza dos por el usuario.

ROM MC 68A316E(U12)

Esta memoria tiene grabado el programa del sistema operativo, llamado también programa Monitor, que controla al sistema MEK 6802-D5.

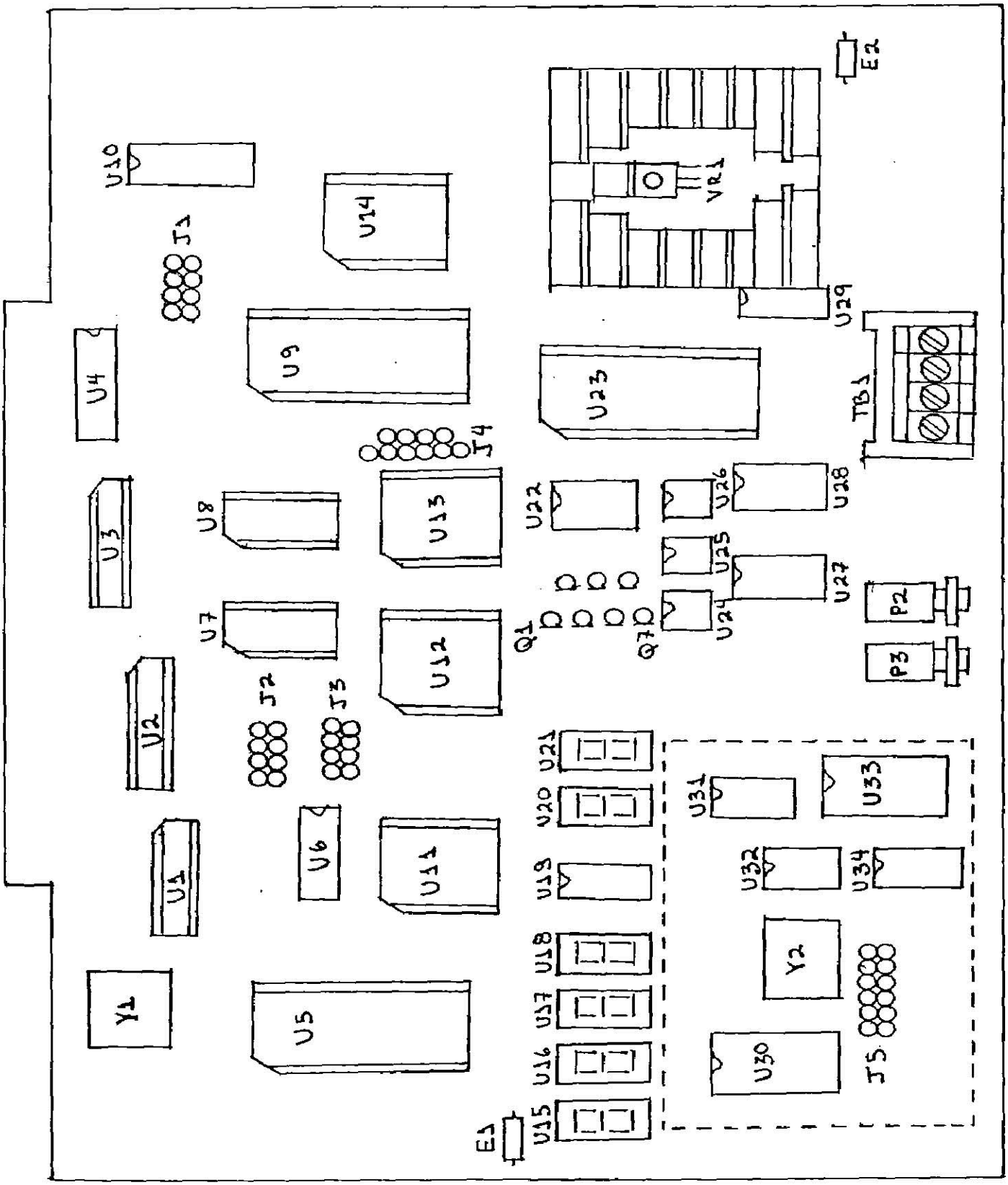
ENTREFAZ ADAPTADORA DE PERIFERICOS(PIA) MC 6821(U23)

Esta entrefaz se utiliza para interconectar al teclado y- los despliegues luminosos con el MPU.

ENTREFAZ ADAPTADORA DE PERIFERICOS(PIA) MC 6821(U9)

Este componente sirve para conectar al MPU con dispositivos externos al sistema, proporciona 2 puertos bidirecciona--

Figura 4.20 Disposición de los Componentes del Sistema MEK 6802 - D5



les de 8 bits y 4 líneas de control, conectados a una base de 24 terminales la cual se encuentra junto a este componente -- (U14).

RAM MC 6810(U11)

Este dispositivo es una memoria estática de lectura/escritura usada por el sistema MEK 6802-D5 como memoria auxiliar - al sistema monitor. Contiene 128 bytes utilizados como área - de pila, almacén de datos y banderas.

DESPLIEGUES(U15, U16, U17, U18,U20, U21)

El desplegado del sistema consiste de 6 indicadores luminosos de 7 segmentos. El PIA del sistema(U23) utiliza 7 líneas para manejar los ánodos y 6 líneas para los cátodos de - los indicadores luminosos.

TECLADO

El teclado sirve para introducir datos a la microcomputadora, indicándole al sistema operativo que es lo que debe hacer por medio de los comandos. Cada tecla tiene un código, el cual es reconocido por el MPU.

DISPOSITIVOS OPCIONALES

RAM MC 2114(U7, U8)

El sistema cuenta con 2 bases de 18 terminales cada una - para poder instalar dos memorias de tipo MC 2114(1kx4 bits).- Con esto se expande la RAM para el usuario a 1k x 8 bits.

MANEJADORES ("BUFFERS") DE DUCTO(U1, U2, U3, U10)

Estas unidades interconectan a todas las líneas del MPU - con el conector de 86 terminales utilizado para la expansión- del sistema. Estos dispositivos son escenciales en dicha ex-- pansión debido a que las líneas del MPU no son capaces de manejar a toda la circuitería conectada externamente.

4.5.4.- PRECAUCIONES DE OPERACION E INSTALACION

Debido a los dispositivos MOS tiene una impedancia de entrada extremadamente alta, son suceptibles de sufrir daños cuando son expuestos a altas descargas eléctricas estáticas. Para evitar estos posibles daños a los dispositivos del sistema durante la manipulación, prueba u operación deben de tomarse en cuenta las siguientes consideraciones.

a).- Los dispositivos deben de estar en contacto con un material conductor, excepto cuando estén probando a estén en operación normal, para evitar las descargas estáticas.

b).- Todos los equipos utilizados en la prueba u operación de los dispositivos deben estar conectados a tierra.

c).- Los dispositivos no deben de insertarse o removerse cuando el sistema tenga energía, debido a que los voltajes transitorios pueden causar un daño permanente.

d).- Las señales externas no deben ser aplicadas a las entradas de los dispositivos mientras éstos no estén energisados.

e).- Todas las entradas de los dispositivos que no sean utilizadas deben de conectarse ya sea a la alimentación(Vcc) o a la tierra(GND) según convenga a la circuitería lógica involucrada.

4.5.5.- DESCRIPCION DE LAS SEÑALES DEL CONECTOR

La figura 4.21 muestra las terminales de este conector así como sus correspondientes señales que a continuación se detallan.

- Ducto de Dirección de 16 líneas(A0-A15).- Sirve para seleccionar a una localidad de memoria de un bloque de 64 bytes.
- Ducto de Datos de 8 Líneas(D0-D7).- Estas líneas bidireccionales sirven de portadoras de los datos de y hacia el MPU.

Número de Pin	Descripción	Número de Pin	Descripción
1	+5 VDC	A	+5 VDC
2	+5 VDC	B	+5 VDC
3	+5 VDC	C	+5 VDC
4	HALT	D	IRQ
5	RESET	E	NMI
6	R/W	F	VMA
7		H	
8	+12 V GND(ref)	J	E(Ø2)
9	+12 V GND(ref)	K	+12 V GND(ref)
10		L	
11	-12 V (ref)	M	-12 V (ref)
12		N	
13		P	BA
14		R	MR
15		S	
16	+12 V (ref)	T	+12 V (ref)
17		U	
18		V	
19		W	
20		X	
21		Y	
22		Z	
23		A	
24		B	
25	-5 V (ref)	C	
26		D	
27		E	
28		F	
29	D1	H	D3
30	D5	J	D7
31	DØ	K	D2
32	D4	L	D6
33	A15	M	A14
34	A12	N	A13
35	A11	P	A10
36	A8	R	A9
37	A7	S	A6
38	A4	T	A5
39	A3	U	A2
40	AØ	V	A1
41	GND	W	GND
42	GND	X	GND
43	GND	Y	GND

Figura 4.21 Terminales de Conector de la Tableta del Sistema MEK 6802 - D5

Líneas de Control(9 líneas)

a).- E(Habilitación).- Esta señal se utiliza como señal de reloj del sistema. El MPU utiliza un cristal externo de una frecuencia de oscilación de 3.579 MHz, para generar la señal de reloj del sistema de 894.8 KHz.

b).- R/W(Lectura/Escritura).- Esta señal sirve para controlar el sentido de la información en el ducto de datos, es decir, determina si es una operación de lectura o de escritura dependiendo de su estado lógico. Cuando tiene un estado alto, los datos fluyen hacia el MPU(lectura) y cuando tiene un estado bajo, los datos salen del MPU.

c).- VMA(Dirección Válida).- Si esta línea de salida tiene un nivel bajo indicará que el ducto de direcciones tiene una dirección no válida; será válida si la línea VMA presenta un nivel alto.

d).- MR(Memoria Lista).- Esta línea de entrada sirve para "alargar" la señal de habilitación(E). Cuando la línea MR tiene un nivel alto, la señal E estará en operación normal; cuando tenga un nivel bajo, la señal E se mantendrá en un nivel durante todo el tiempo que tarde la línea MR en estado bajo. Esta condición es útil para conectar al MPU con dispositivos que operan lentamente.

e).- $\overline{\text{RESET}}$(Restablecimiento).- Esta línea sirve para realizar el restablecimiento del MPU. Su nivel activo es bajo.

f).- BA(Ducto Disponible).- Esta línea sirve para indicar que el MPU está en la condición de Paro y el ducto de direcciones está disponible(pero no en estado de alta impedancia).

g).- $\overline{\text{HALT}}$(Paro).- Esta línea con nivel activo bajo, sirve para colocar al MPU en la condición de Paro.

h).- $\overline{\text{IRQ}}$(Solicitud de Interrupción).- Esta señal provoca que se genere una secuencia de atención de interrupción dentro del MPU. En el registro de código de Condiciones ex-

iste un bit de interrupción que sirve para habilitar o -- deshabilitar la función de esta línea de entrada.

i).- $\overline{\text{NMI}}$(Interrupción No-Enmascarable).- Esta línea funciona- de la misma manera que $\overline{\text{IRQ}}$ excepto que a $\overline{\text{NMI}}$ no le afecta el estado del bit de interrupción del Registro de Código de Condiciones.

4.5.6.- ENTREFACES DEL SISTEMA

La tableta del sistema MEK 6802-D5 cuenta con una fuente- de alimentación que suministra la potencia necesaria para alimentar al sistema mínimo, es decir, a todos los dispositivos- montados en la tableta. Si se necesita expander el sistema o- conectar algún dispositivo externo a través del PIA se tiene- que conectar una fuente de alimentación externa de +5V CD. de mayor corriente. Al realizar esto, se tiene que quitar el --- "puente" E2 de la tableta.

A la fuente de alimentación de la tableta se le tiene que conectar un transformador de 18V CA con derivación central. - Dicha conexión mostrada en la figura 4.22 nos ilustra lo anterior.

Si se quiere conectar una fuente de alimentación de +5VCD externa se tiene que desconectar el transformador y hacer la- conexión mostrada en la figura 4.23.

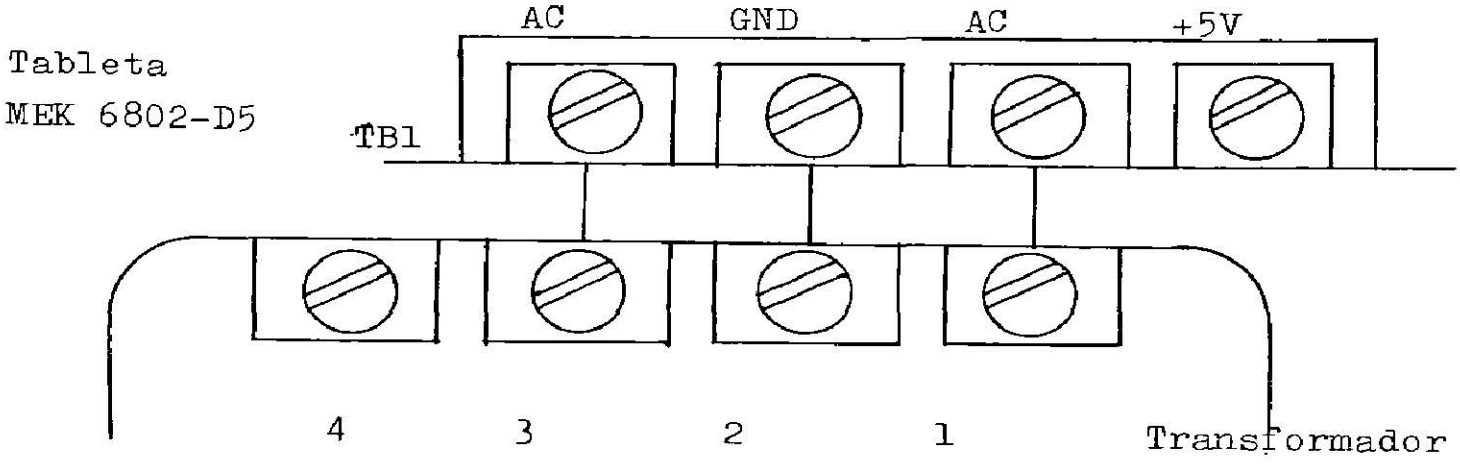


Figura 4.22 Conexión del Transformador de 18 VCA

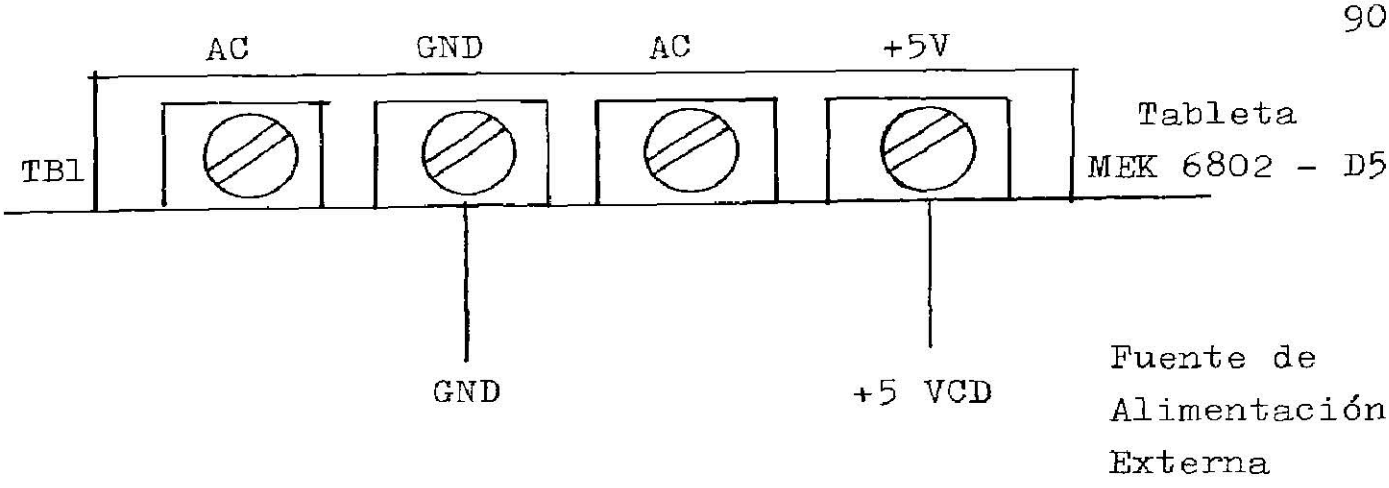


Figura 4.23 Conexión de Fuente de Alimentación Externa

PIA del Usuario

La tableta del sistema cuenta con una base para circuito-integrado(U14) de 24 terminales que tienen conectadas las señales de entrefaz del PIA del usuario(U9). La tabla 4.4 muestra el número de la terminal de la base y su correspondiente-señal de entrefaz del PIA.

No. de Terminal	Señal de Entrefaz	No. de Terminal	Señal de Entrefaz
1	PA6	13	GND
2	PA7	14	Sin conexión
3	PB0	15	PA5
4	PB1	16	PA4
5	PB2	17	PA3
6	PB3	18	PA2
7	PB4	19	PA1
8	PB5	20	PA0
9	PB6	21	CA2
10	PB7	22	CA1
11	CB1	23	Sin Conexión
12	CB2	24	+5V

Tabla 4.4 Señales de Entrefaz del PIA del Usuario en la Base (U14)

Las líneas de solicitud de interrupción IRQA e IRQB del - PIA del usuario pueden conectarse a cualquiera de las entra-- das de solicitud de interrupción $\overline{NMI}$ o $\overline{IRQ}$ del MPU. Para esto se cuenta con un conjunto de terminales para "puentes" donde- se pueden conectar las opciones seleccionadas. La figura 4.24 muestra las terminales para los "puentes" mencionados.

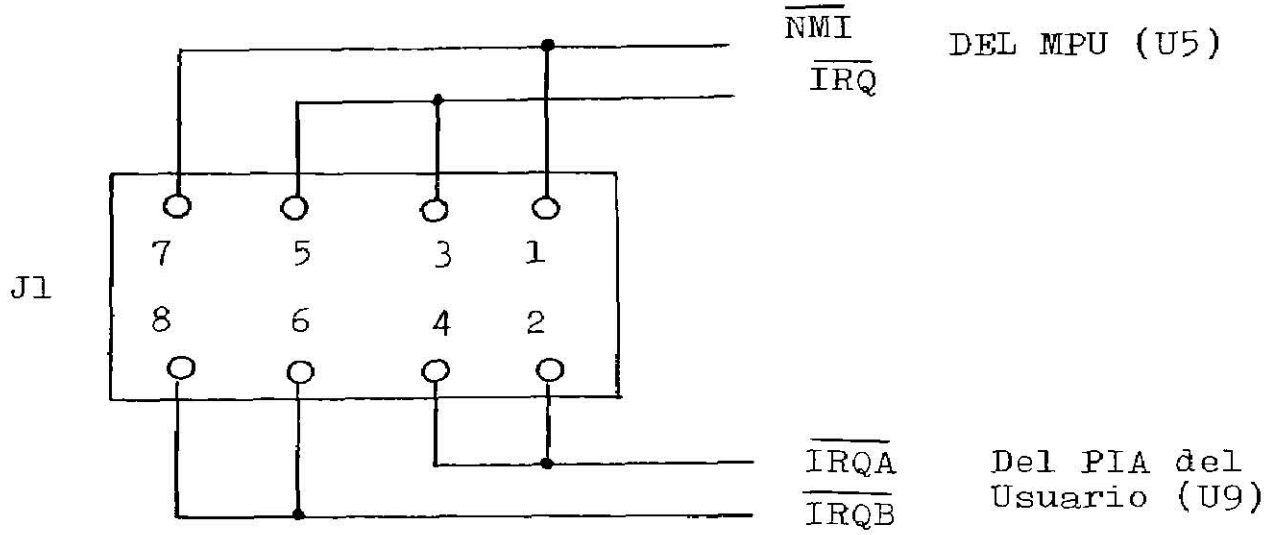


Figura 4.24 Conexiones para las Entradas de Solicitud de Interruoción del PIA del Usuario

El PIA del usuario está decodificado para responder a las siguientes direcciones:

Registro de Datos lado A	$E480
Registro de Dirección de Datos lado A	$E480
Registro de Control lado A	$E481
Registro de Datos lado B	$E482
Registro de Dirección de Datos lado B	$E482
Registro de Control lado B	$E483

Entrefaz para Cinta Magnética

El sistema cuenta con una entrefaz para cinta magnética,- es decir, el sistema puede almacenar programas en cinta magnética. La tableta es compatible con muchas grabadoras de cinta de bajo costo, la salida de la grabadora debe conectarse a la

entrada "EAR"(P3) de la tableta, ésta puede aceptar señales - tan bajas como de 2 Vpp.

Para grabar los programas debe conectarse la salida "MIC" (P2) de la tableta a la entrada "MIC" de la grabadora de cinta. El MEK 6802-D5 envía una señal de 50 mVpp de 1200 Hz y -- 2400 Hz para ceros y unos respectivamente.

OPCIONES

RAM del Usuario

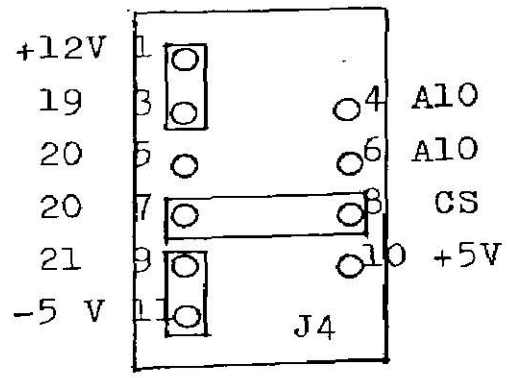
Como se mencionó anteriormente, el sistema cuenta con una opción que permite añadir 1k x 8 bits de RAM del usuario. Esta memoria está decodificada de forma que responda en las direcciones $E000-$E3FF. Si se selecciona esta opción se deben- colocar 2 RAM del tipo MCM 2114 en las bases U7 y U8.

ROM del Usuario

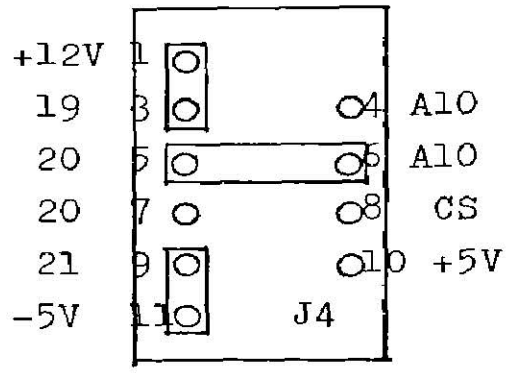
El sistema cuenta con opción de colocar una ROM del usuario, es decir, una memoria donde se tengan grabados los pro-- gramas del usuario; para esto se tiene que insertar la ROM -- mencionada en la base U13. La decodificación del sistema causa que la ROM sea seleccionada por 2 conjuntos de direcciones de la $E800 a $EFFF ó de la $F800 a la $FFFF. Si el usuario - quiere que en su memoria se contengan los vectores de inter-- rupción y restablecimiento del sistema, debe de seleccionar - el conjunto de las direcciones comprendidas entre la $F800 a- la $FFFF y debe deshabilitarse la ROM del sistema para que no pueda responder a estas direcciones.

Además de poder seleccionar las direcciones, el usuario - puede conectar varios tipos de ROM ó EPROM. La tabla 4.5 muestra las conexiones de la EPROM más comúnes.

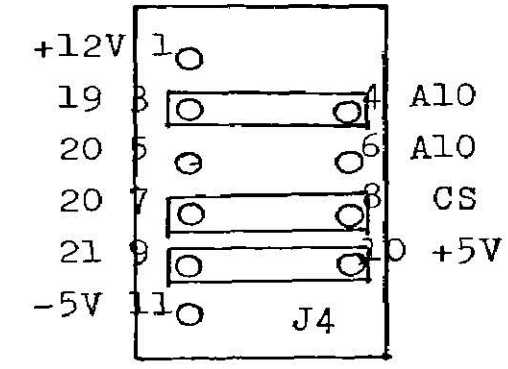
En la tableta se encuentran los conjuntos de terminales - para "puentes"(J2, J3 y J4) que sirven para seleccionar el tipo de memoria, así como su dirección. Las figuras 4.25 y 4.26 muestran la colocación de los "puentes" para las diferentes - opciones.



a).- Conexiones para la EPROM de 1Kx8 con 3 fuentes



b).- Conexiones para la EPROM de 2Kx8 con 3 fuentes

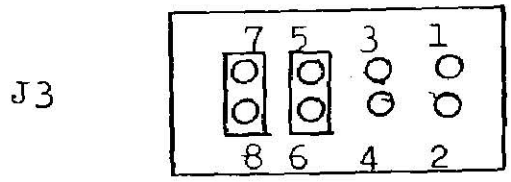


c).- Conexiones para la EPROM de 2Kx8 con una fuente

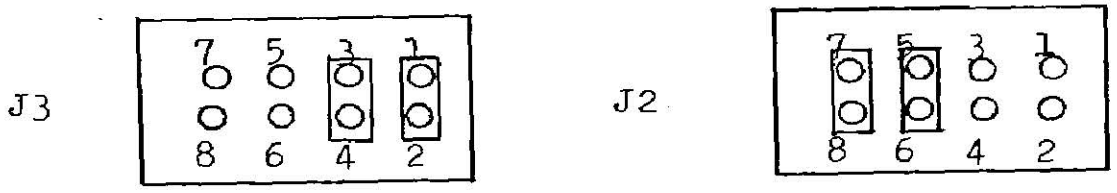
Figura 4.25 Configuración de los "Puentes" para los diferentes tipos de EPROM

TIPOS	ORGANIZACION	TERMINALES			
		21	20	19	18
MCM 2708	1 K x 8	-5V	CS	+12V	GND
MCM 2716	2 K x 8	+5V	CS	A10	GND
TMS 2716	2 K x 8	-5V	A10	+12V	CS

Tabla 4.5 Conexiones de EPROM



a).- Opción 1, Selección del conjunto de direcciones $E800 - $EFFF



b).- Opción 2, Selección del conjunto de direcciones $F800 - $FFFF

Figura 4.26 Configuración de los "Puentes" para seleccionar el conjunto de direcciones

Si se usa la opción 2 de la figura 4.26, la ROM del sistema debe deshabilitarse para que no responda en las direcciones $F800-$FFFF, esto se logra quitando el "puente" de las terminales 1-2 de J2.

Expansión

El sistema cuenta con la posibilidad de expanderse. Para esto se tiene que colocar "buffers" en las líneas de los ductos de dirección y de datos, así como en las de control. Dichos "buffers" se colocan de la siguiente forma: 2 "buffers" del tipo 74LS244 en las bases U1 y U2 y uno del tipo 74LS245 en la base U3. Recuérdese que si se va a expander el sistema se tiene que deshabilitar la RAM interna del MPU MC 6802. Esto se realiza colocando un "puente" en E1. Las figuras 4.27 y 4.28 muestran el diagrama de las conexiones del sistema MEK - 6802-D5.

4.5.7.- OPERACION

El sistema MEK 6802-D5 cuenta con un sistema operativo que sirve, entre otras cosas, para comunicarse con el usuario; quien proporciona los comandos de operación necesarios a través de un teclado. Cuenta también con un conjunto de 6 indicadores de 7 segmentos que ayudan al usuario en el "diálogo" con el sistema.

A continuación se describe la operación del teclado y sus funciones dentro del sistema.

Restablecimiento

Para restablecer el sistema se presiona la tecla "RS" y aparece un guión en el primer despliegue de la izquierda, indicando que el sistema se encuentra en monitor.

Despliegue o Cambio de una Localidad de Memoria

Para desplegar o cambiar el contenido de una localidad de memoria, se parte del monitor, después, se introduce la direc

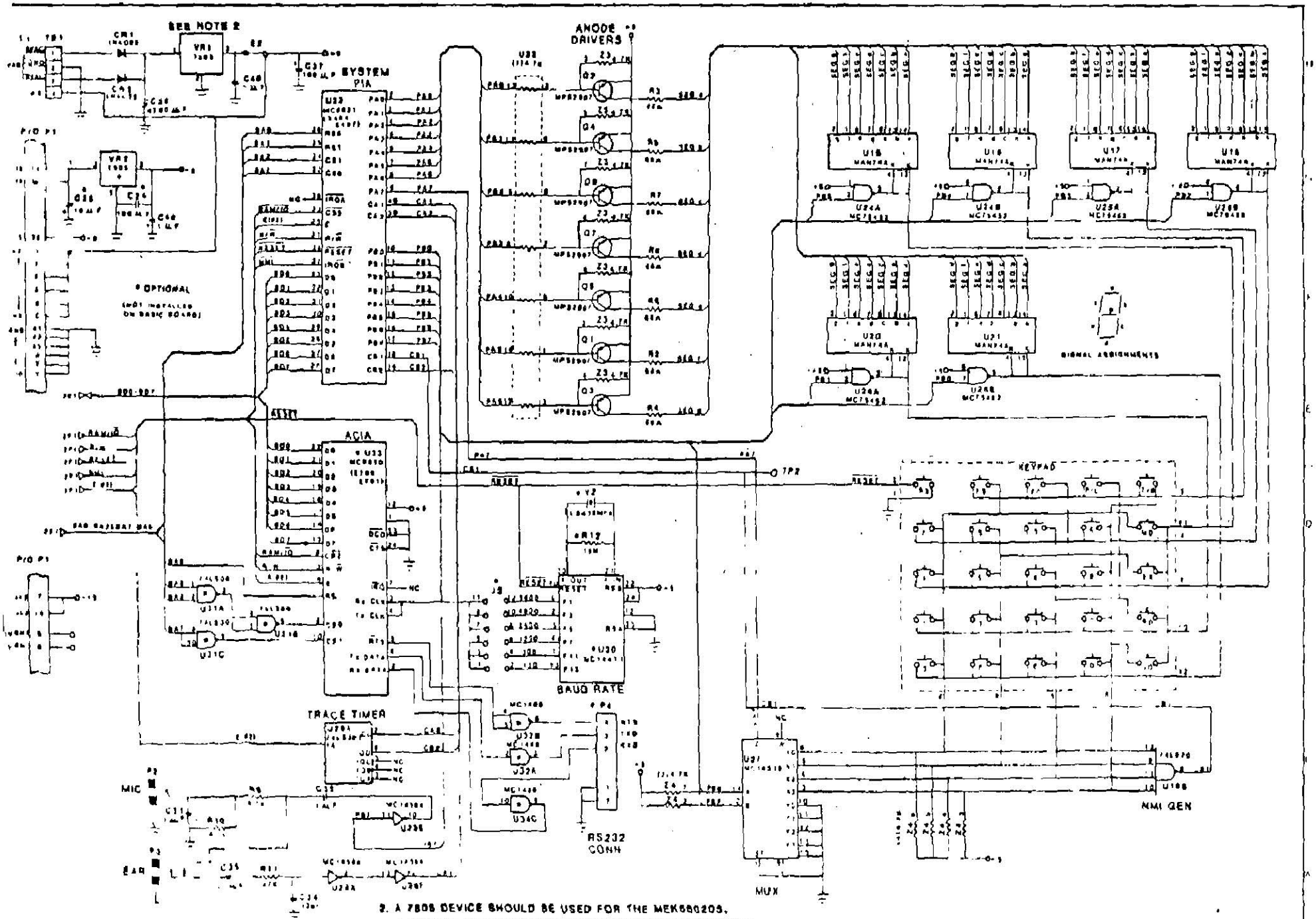


FIGURE 5.1 SCHEMATIC DIAGRAM
(SHEET 3 OF 3) 5-7/5-8

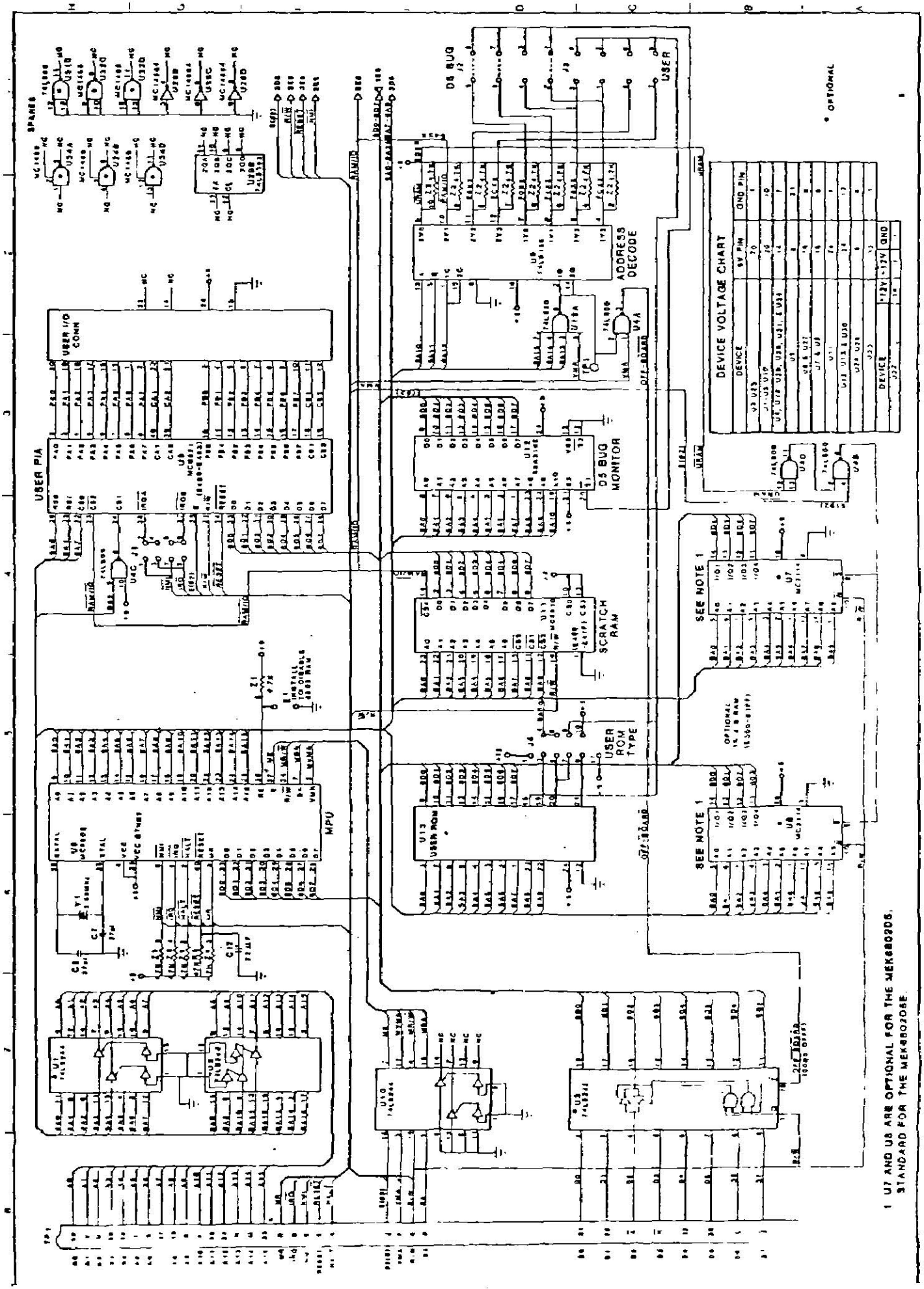


FIGURE 5.1 SCHEMATIC DIAGRAM (SHEET 2 OF 3) 5-5/5-6

ción en código hexadecimal a través del teclado. En los 4 primeros despliegues aparecerá esta dirección. El primer dígito que se introdusca aparecerá en el cuarto despliegue precedido por ceros, al introducir el segundo dígito de la dirección, el primero se recorre a la izquierda y el segundo aparecerá en el cuarto despliegue y así sucesivamente hasta introducir los demás dígitos.

Después de que esté desplegada la dirección correcta se oprime la tecla "M", con esto aparecerá el contenido en código hexadecimal de la dirección en los últimos 2 despliegues.

Si se desea cambiar el contenido de dicha dirección en este momento se puede introducir el nuevo dato por medio del teclado.

Si se desea desplegar la siguiente localidad de memoria se presiona la tecla "GO" y aparecerá la siguiente dirección con su respectivo contenido. Si se quiere cambiar dicho contenido, basta introducirlo en este momento. Si se quiere desplegar la dirección anterior a la mostrada en los despliegues, se oprime la tecla "M" y aparecerá la dirección anterior y su correspondiente contenido.

Despliegues o Cambio de Registros

Para obtener la información de los registros internos del MPU, se presiona la tecla "RD" estando en el monitor; después aparecerá en los primeros 4 despliegues el contenido del contador del programa y en el quinto y sexto las letras "PC". En este momento se puede cambiar, si se desea, el contenido del contador del programa. Para observar el siguiente registro -- (Acumulador A), se presiona la tecla "GO" y aparecerá en el tercer y cuarto despliegues, el contenido del acumulador A y en el sexto la letra "A".

El siguiente registro es el Acumulador B y aparece en la misma forma que el Acumulador A presionando la tecla "GO", ahora en el sexto despliegue aparecerá la letra "B".

Después del acumulador B sigue el registro de índices X,- luego el apuntador de la pila y al final el registro de código de condiciones. La tabla 4.6 muestra los registros internos del MPU y el orden y la forma en que aparecen en los despliegues:

pc	pc	pc	pc	p	c	Contador del Programa
		a	a		a	Acumulador A
		b	b		b	Acumulador B
X	X	X	X	I	d	Registro de Indice X
sp	sp	sp	sp	s	p	Apuntador de Pila
		c	c	c	c	Registro de Código de Condiciones

Tabla 4.6 Despliegues de los Registros Internos del MPU

Si se despliega el registro de código de condiciones y -- después se presiona la tecla "GO", aparecerá el primer registro, es decir, el contador del programa y su contenido.

Para desplegar el registro anterior al mostrado en los -- despliegues, se presiona la tecla "M", el orden en que aparecen es el inverso al mostrado en la tabla 4.6. Si está desplegado el contador del programa y se presiona la tecla "M", aparecerá el registro de código de condiciones. Para salirse de esta rutina se presiona la tecla "EX" y aparecerá el guión indicador de monitor.

Corrida de Programas

Para ejecutar un programa del usuario, se parte del monitor, se introduce por medio del teclado la dirección inicial del programa y se presiona la tecla "GO".

Continuar

Este comando es igual en funcionamiento a la corrida de un programa, excepto que, en este comando, no se le coloca dirección inicial de la corrida, o sea que se toma como direc--

ción inicial a la contenida en el contador del programa. El - procedimiento es el siguiente: se parte desde el monitor y se presiona la tecla "GO".

Salida del Programa("Escape")

Para detener la ejecución de un programa, se presiona la- tecla "EX" y en los despliegues aparecerá el contenido del -- contador del programa y las letras "PC". Dicho contenido indi cará la dirección donde se detuvo la ejecución del programa;- el sistema ejecuta la rutina de muestra los registros inter-- nos del MPU. Esto ocurre si se estaba ejecutando un programa- del usuario en el momento que se presionó la tecla "EX", pero si en dicho momento se estaba ejecutando alguna rutina del -- sistema operativo, entonces retornará al monitor.

Cálculo de Desplazamiento("Offset")

En las instrucciones de salto(excepto JSR y JMP) se tiene que colocar un desplazamiento, es decir, un número de localidades de memoria que el contador del programa tiene que "saltar". El sistema MEK 6802-D5 cuenta con una rutina que calcula el desplazamiento que hay entre 2 localidades de memoria.- El procedimiento es el siguiente:

1.- Se coloca el sistema en monitor.
2.- Se introduce la dirección correspondiente al byte de desplazamiento.
3.- Se presiona la tecla "M", aparecerá el contenido de dicha dirección.
4.- Se presiona la tecla "FS", se desplegará solamente la letra "A" en el sexto despliegue.
5.- Se introduce la dirección destino, se desplegará en los - primeros 4 despliegues acompañada por la letra "A" ante-- rior.
6.- Se presiona la tecla "GO", el desplazamiento calculado aparecerá en el quinto y sexto despliegues. Si está fuera- de rango permitido($+129_{10}$; -127_{10}), aparecerá la palabra "Bad".

7.- Una vez calculado el desplazamiento, se presiona la tecla "GO" y se almacenará en la dirección correspondiente del- byte de la instrucción de salto y el sistema pasa a la rutina de despliegue o cambio de una localidad de memoria - desplegandose la siguiente dirección y su contenido.

8.- Si no quiere almacenar el desplazamiento calculado, des-- pués del paso 6 se presiona la tecla "FC" y el sistema regresará a la rutina de despliegue o cambio de una localidad de memoria , desplegando la dirección del byte de desplazamiento y su contenido.

9.- Si en el paso 6 aparece la palabra "Bad", se presiona la- tecla "M" y el sistema regresará a la rutina de despliegue o cambio de una localidad de memoria, desplegándose - la dirección del desplazamiento que se quiso calcular.

Ejecución de un Programa Instrucción por Instrucción ("TRACE SINGLE STEP")

El procedimiento para ejecutar un programa instrucción -- por instrucción es el siguiente:

1.- Se coloca el sistema en monitor.

2.- Se presiona la tecla "RD", desplegándose el contenido del contador del programa y las letras "PC".

3.- Se introduce la dirección inicial del programa a ejecutar la cual se desplegará junto con las letras "PC".

4.- Se presiona la tecla "T/B" y se ejecutará la primera instrucción desplegándose la dirección de la siguiente ins-- trucción a ejecutar al igual que las letras "PC".

5.- En este momento el sistema se encuentra en la rutina de - despliegue o cambio de registro. Si se desea observar o - cambiar el contenido de éstos, basta presionar la tecla - "GO" o "M".

6.- Si se desea observar o cambiar el contenido de alguna localidad de memoria se pasa el sistema a la rutina que realiza esto, se presiona la tecla "EX", desplegándose el -- guión en el primer despliegue, se introduce la dirección-

de memoria que se desea observar, se presiona la tecla -- "M" y aparecerá su contenido. Si se requiere se puede cambiar dicho contenido.

7.- Para regresar a la rutina de instrucción por instrucción- se presiona la tecla "EX" y "RD" desplegándose el contenido del contador del programa y las letras "PC".

8.- Para ejecutar la siguiente instrucción se presiona la tecla "T/B" desplegándose la dirección de la siguiente instrucción a ejecutar.

9.- Por cada instrucción a ejecutar se presiona la tecla "T/B".

Puntos de Detención("Breakpoints")

En la ejecución de algunos programas a veces es necesario detenerlos después de haber ejecutado una porción de ellos, - ya sea para analizar los registros internos del MPU o algunas localidades de memoria, o aplicar señales externas necesarias para el propósito del programa.

Para ésto, el sistema cuenta con una rutina que coloca -- puntos de detención de ejecución de programas en localidades- de memoria. El procedimiento para colocarlos es el siguiente:

1.- El sistema deberá estar en monitor.

2.- Se presiona la tecla "FS", desplegándose "-FS".

3.- Se presiona la tecla "T/B", desplegándose solamente un cero en el sexto despliegue. En ese momento, el sistema seencuentra en la rutina que coloca los puntos de detención.

4.- El cero desplegado en el paso anterior indica que no hayningún punto de detención colocado en las localidades dememoria. Se introduce la dirección del primer punto de detención, es decir, la dirección donde se desea parar la - ejecución del programa, después se presiona la tecla "FS" se desplegará la dirección del primer punto de detención- y en el quinto y sexto despliegues aparecerá "01" indicando que se ha introducido un punto de detención.

5.- Si se desea colocar otro punto de detención, solamente se tiene que introducir en este momento la dirección donde -

se requiere y presionar la tecla "FS", con esto se desplegará la dirección del segundo punto de detención y en el quinto y sexto despliegues aparecerá "02" indicando que hay 2 puntos de detención colocados.

6.- Esta misma secuencia se continúa hasta llegar a 5 puntos de detención que son los que acepta como máximo el sistema. Para salir de esta rutina se oprime la tecla "EX" y el sistema retorna al monitor.

7.- Para observar el número y las direcciones de los puntos de detención, el sistema debe estar en la rutina de punto de detención, es decir, debieron haberse presionado las teclas "FS" y "T/B" desde el monitor. Con esto se despliega la dirección de uno de los puntos de detención y en el sexto despliegue aparecerá el número de ellos.

8.- Para observar la dirección del siguiente punto de detención se presiona la tecla "GO" y así sucesivamente. Al hacer esta operación, el número indicador de puntos de detención no se altera.

9.- Si se desea introducir un nuevo punto de detención, basta llamar desde el monitor a la rutina mencionada presionando las teclas "FS" y "T/B", enseguida se introduce la dirección del punto de detención y se presiona la tecla "FS". El número indicador de puntos de detención se incrementará.

10.-Para cancelar un punto de detención, se parte del monitor, se presionan las teclas "FS" y "T/B", después se presiona la tecla "GO" tantas veces como sea necesario hasta que sea desplegada la dirección del punto de detención que se quiere anular. Se presiona después la tecla "FC". En ese momento se desplegará la dirección del punto de detención anterior, y el número indicador de puntos el cual se decrementará.

11.-El restablecimiento del sistema cancela a todos los puntos de detención.

Funciones del Usuario

El sistema MEK 6802-D5 cuenta con una rutina que le permite al usuario ejecutar funciones especiales por medio del teclado, es decir, desde el teclado puede mandar los comandos para que el sistema realice una de las 16 funciones especiales que fueron previamente introducidas.

Esto significa que el usuario puede ejecutar una a la vez, de 16 programas que previamente fueron introducidos en la memoria del sistema.

Se introduce una tabla que indica las direcciones de los 16 programas que se quieren ejecutar, en un apuntador de tabla se tiene que colocar la dirección de inicio de dicha tabla. El apuntador de la tabla lo forman las direcciones $E43F y $E440.

Por ejemplo si se tienen 5 programas independientes en las direcciones $0010, $0030, $E020 y $E040; la tabla estará en la dirección $0050, entonces en el apuntador de la tabla se tiene que colocar el número $0050. El apuntador de la tabla y la tabla de direcciones quedará como sigue:

Dirección	Contenido	Comentario
E43F	00	Dirección de Inicio
E440	50	De la tabla
Tabla		
0050	00	Dirección del
0051	30	Primer programa
0052	00	Dirección del
0053	10	Segundo programa
0054	E0	Dirección del
0055	20	Tercer programa
0056	E0	Dirección del
0057	40	Cuarto programa
0058	E0	Dirección del
0059	00	Quinto programa

Después que se hayan introducido los 5 programas y la tabla anterior, se coloca al sistema en monitor, se presiona la tecla "FS" y después un dígito hexadecimal: un "0", para ejecutar el primer programa, un "1" para el segundo, un "2" para el tercero, un "3" para el cuarto y un "4" para ejecutar el quinto programa.

Como se mencionó anteriormente, el usuario puede ejecutar como máximo 16 programas que corresponden a las teclas "0" y "F". Deben tenerse en cuenta que el restablecimiento(RS) del sistema coloca a cero al apuntador de la tabla.

Grabación de un Programa en Cinta Magnética

Para grabar un programa en cinta magnética se debe colocar la grabadora como se menciona en la sección 4.5.6. El procedimiento de grabado es el siguiente:

Se parte desde el monitor del sistema, se oprime la tecla "P/L" y en los despliegues aparecerá "0000bb"; en estas condiciones se introduce la dirección inicial del programa. Después se presiona la tecla "GO", en los despliegues aparecerá "0000EE", después se introduce la última dirección del programa que se quiera grabar, enseguida se habilita la grabación en la grabadora y se presiona la tecla "GO".

El sistema inicia la grabación del programa enviando los caracteres "FF" durante los primeros 30 segundos, enseguida envía la dirección inicial y la final. Después de esto, envía la información contenida en las direcciones comprendidas entre la inicial y la final, inclusive. Al término de la grabación del sistema retorna a monitor.

Carga de un Programa de Cinta Magnética a Memoria

Cuando se desea pasar un programa grabado en cinta magnética a la memoria del sistema se sigue la siguiente secuencia:

Se parte con el sistema en monitor, se presiona la tecla "FS" y en los despliegues aparecerá "-FS", después se presiona la tecla "P/L", luego se deberá habilitar la reproducción-

de la cinta en la grabadora. Al término de la carga del pro-- grama, si no hubo ningún error, el sistema retorna al monitor.

Si ocurre un error en la carga del programa, el sistema - desplegará el aviso "FAIL??". Presionando la tecla "EX" el -- sistema pasará al monitor, el registro de índice contendrá la dirección donde ocurrió el error y el acumulador A contendrá- el último byte leído desde la cinta.

Verificación de un Programa

El sistema cuenta con una rutina que verifica si un pro-- grama de la memoria del sistema es igual a otro que está gra- bado en cinta magnética. El procedimiento para efectuar dicha rutina es el siguiente:

Se parte desde el monitor del sistema, se presiona la te- cla "FS" y "RD"; a continuación se habilita la reproducción - en la grabadora y en ese momento, el sistema verifica los pro gramas, si no coinciden manda el mensaje "FAIL??" a través de los despliegues. Presionando "EX" se retorna al monitor del - sistema y el registro de índice contendrá la dirección de --- error y el acumulador A contendrá el contenido del último --- byte leído de la cinta.

4.5.7.- RUTINAS DEL SISTEMA

La tabla 4.7 muestra las direcciones de las rutinas del - sistema MEK 6802-D5.

Nombre de la Rutina	Dirección	Descripción
RESET	$F000	Restablecimiento.
PROMPT	$F024	Inicialización.
GET	$F04E	Rutina para leer una te cla.
PUT	$F0BB	Muestra un dato en los- despliegues.
DYSCOO	$F120	Decodifica de hex a 7 seg.

Nombre de la Rutina	Dirección	Descripción
DLY25	$F169	Produce un retardo de 25-mseg.
DLY1	$F171	Produce un retardo de 1--mseg.
DLYX	$F179	Produce un retardo basado en el registro de índice-(0.5 seg. máximo).
ADDAX	$F183	Suma el registro A al registro X.
CLRDS	$F195	Borra los despliegues debido a una máscara del registro A.
ROLL2	$F1AA	Rutina de entrada numérica para 2 dígitos(1 byte)
ROLL4	$F1CC	Rutina de entrada numérica para 4 dígitos(2 Byte)
RDKEY	$F1EF	Lee y reconoce una tecla.
TIN	$F533	Lee 1 byte de la entrefaz para cinta magnética.
PNCHB	$F630	Estructura y graba un archivo.
LOAD	$F69C	Carga un archivo o verifica un archivo.
MNPTR	$E419,A	Apuntador de subprograma-activo.
KEY	$E41B	Introduce un código de tecla.
KYFLG	$E41C	Bandera indicadora de tecla pendiente.
DISBUF	$E41D-$E422	6 bytes que corresponden-a los despliegues de 7 segmentos.

Nombre de la Rutina	Dirección	Descripción
ROLPAS	$E423	Bandera indicadora de la-entrada del primer dígito.
HEXBUF	$E42C,D,E	3 bytes para información-hex. Cada uno corresponde a 2 despliegues.
USP	$E42F,30	Localidad donde se almace na el apuntador de pila - del usuario.
UCC	$E431	Localidad donde se almace na el registro de código- de condiciones.
UB	$E432	Localidad donde se almace na el acumulador B.
UA	$E433	Localidad donde se almace na el acumulador A.
UX	$E434,5	Localidad donde se almace na el IR o IX.
UPC	$E436,7	Localidad donde se almace na el PC.
UlRQU	$E43C,D	Apuntador de la dirección de inicio de la rutina de servicio IRQ del usuario.
FNCFL	$E43E	Bandera indicadora de fun ción especial.
FNCPNT	$E43F,40	Apuntador de dirección de tabla de función especial.
BYTE	$E459	Byte de dato leído desde- cinta o para grabar cinta.
BEGAD	$E460,1	Indicador de dirección de inicio para grabar.
ENDAD	$E462,3	Indicador de última direc ción para grabar.

Tabla 4.7 Direcciones de los Programas D5BUG

4.6.- CONVERSIONES

4.6.1.- CONVERSION DE BINARIO A HEXADECIMAL

Se mostrará con este ejemplo el manejo de indicadores luminosos de 7 segmentos; la microcomputadora hará el papel de un decodificador de un número binario de 4 bits a código de 7 segmentos representando en el indicador, dígitos hexadecimales. El manejo de interruptores e indicador se hará a través del PIA como se indica en la figura 4.27.

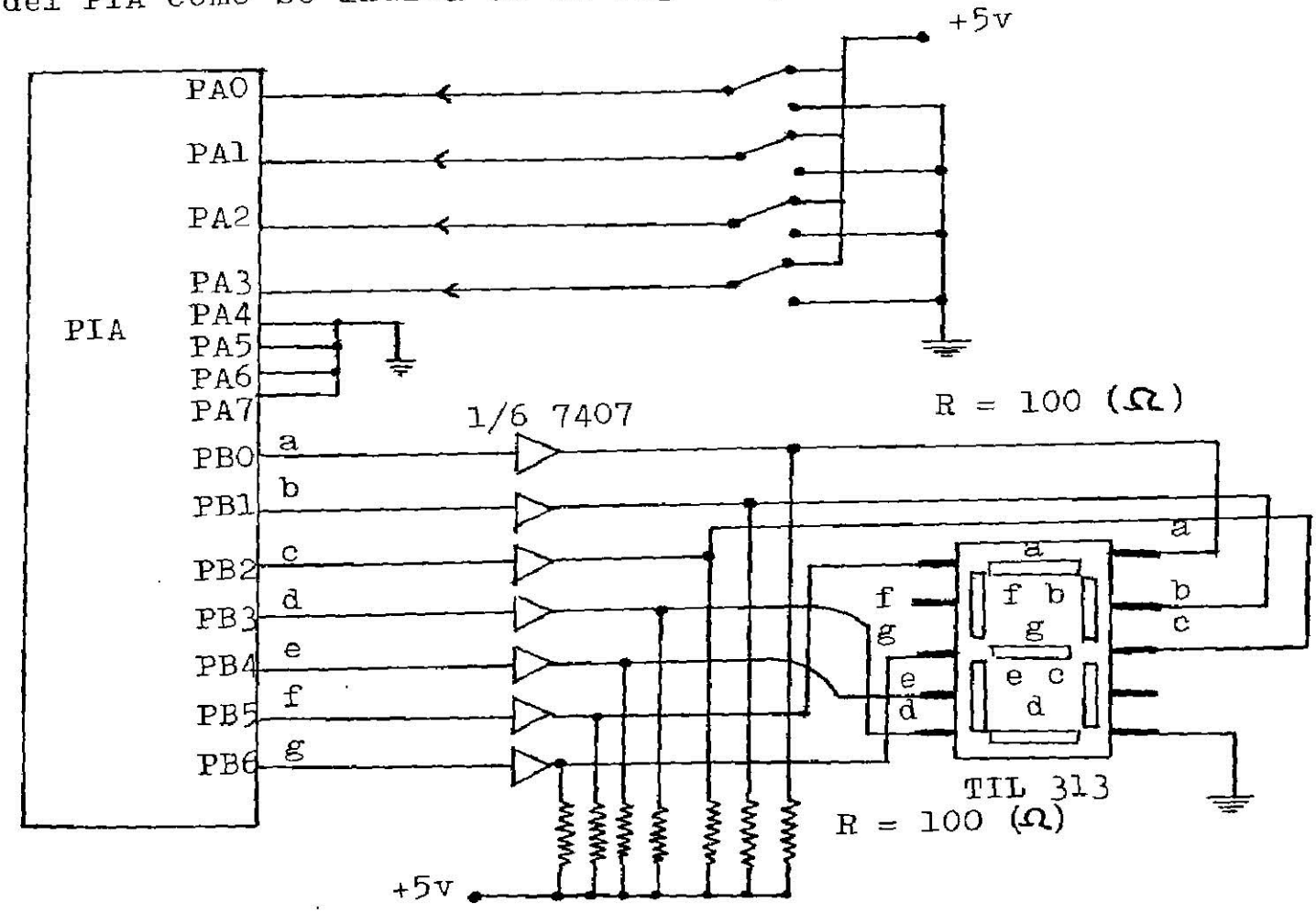


Figura 4.27

El algoritmo de codificación se basa en la búsqueda del equivalente de 7 segmentos en una tabla. El número binario, formado por los interruptores, es leído en el lado A del PIA; este número, sumado a la posición inicial de la tabla forman la dirección que contiene el código de 7 segmentos equivalente al número leído. El diagrama de flujo que describe el programa completo de conversión se muestra en la figura 4.28.

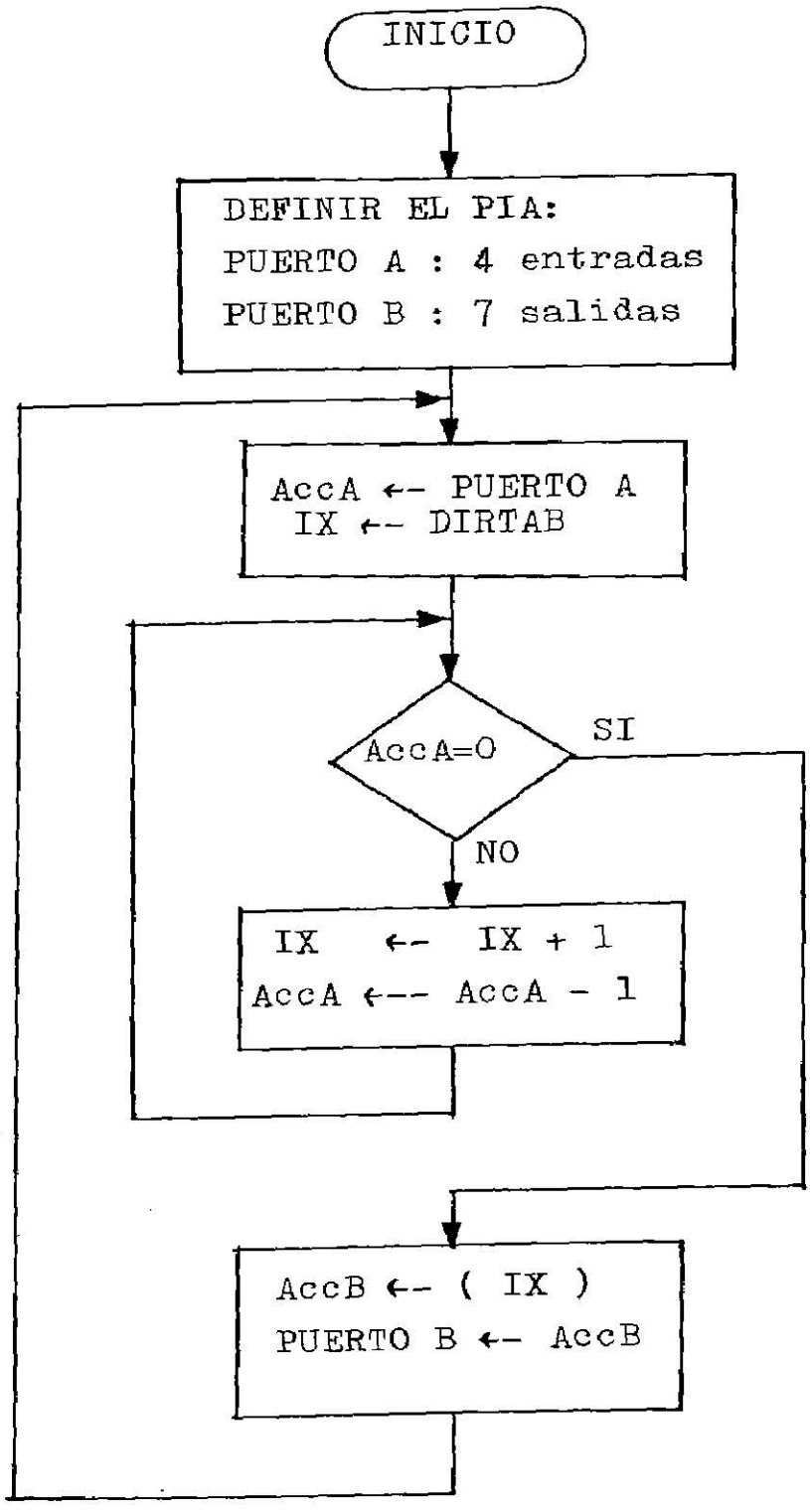

INICIO
DEFINIR EL PIA:
PUERTO A : 4 entradas
PUERTO B : 7 salidas
AccA ← PUERTO A
IX ← DIRTAB
AccA=0
SI
NO
IX ← IX + 1
AccA ← AccA - 1
AccB ← (IX)
PUERTO B ← AccB

Figura 4.28

El programa, escrito en lenguaje simbólico, que efectúa - lo descrito en el diagrama de flujo se dá a continuación:

```
DIRTAB    RMB2
          CLR     CRA
          CLR     CRB
          CLR     DDRA      PUERTO A : 8 ENTRADAS
          LDA A   #$04
          STA A   CRA
          LDA B   #FF
          STA B   DDRB      PUERTO B : 8 SALIDAS
          STA A   CRB       FIN DEFINICION DEL PIA
OTRO      LDA A   PRA       LECTURA DE ENTRADAS
          LDX     DIRTAB
BUSC      TST A             SUMA EL NUMERO LEIDO A
          BEQ     SAKA
          INX               LA DIRECCION DE
          DEC A             LA TABLA
          BRA     BUSC
SAKA      LDA B   $00,X     OBTIENE EQ7 DE TABLA
          STA B   PRB       LO ESCRIBE EN EL PUERTO
          BRA     OTRO      DE SALIDA
```

4.6.2.- MANEJO DE VARIOS INDICADORES DE 7 SEGMENTOS

Se presenta en esta parte una de las técnicas más usadas- para el manejo de varios indicadores de 7 segmentos que evita el uso de más de un PIA. Esta técnica consiste en mandar a todos los indicadores, por un ducto común, la misma información en un instante dado y activar el cátodo(o ánodo) del indicador que debe mostrar tal información, figura 4.29. Es necesario dejar encendido por un instante el indicador en cuestión- para después apagarlo y presentar otro dato correspondiente - a otro indicador de 7 segmentos. Este proceso se repite encendiendo secuencialmente y en forma cíclica cada uno de los exibidores luminosos dado el efecto visual de un despliegue continuo de información. A este proceso se le llama refrescamiento.

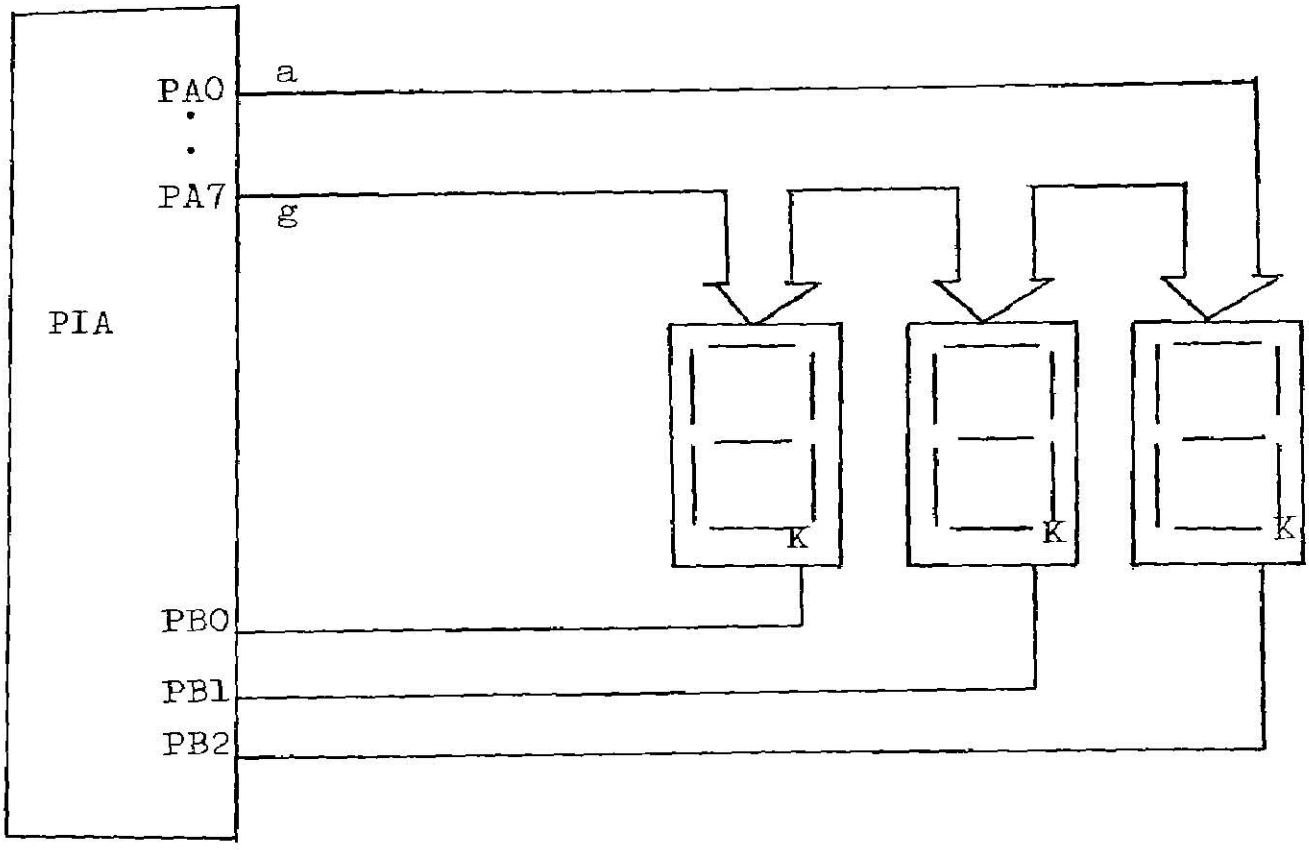


Figura 4.29

Presentamos aquí un ejemplo, en el que se realiza la conversión de un número binario de 8 bits ubicado en memoria, a un número decimal de 3 dígitos representado en los indicadores luminosos.

Las tareas a efectuar por el programa se plantean con ayuda del diagrama de flujo de la figura 4.30. Cada una de esas tareas se detallan más adelante con otro diagrama de flujo y se dá el subprograma escrito en ensamblador; al final se presenta el programa principal con los llamados a las subrutinas.

4.6.3.- CONVERSION DE NUMERO BINARIO A BCD

En esta parte se transforma el número binario de 8 bits (NUM), a un número en BCD expresado en 3 bytes(BCDi; i=1,2,3), pasando por un formato de 2 bytes, figura 4.31.

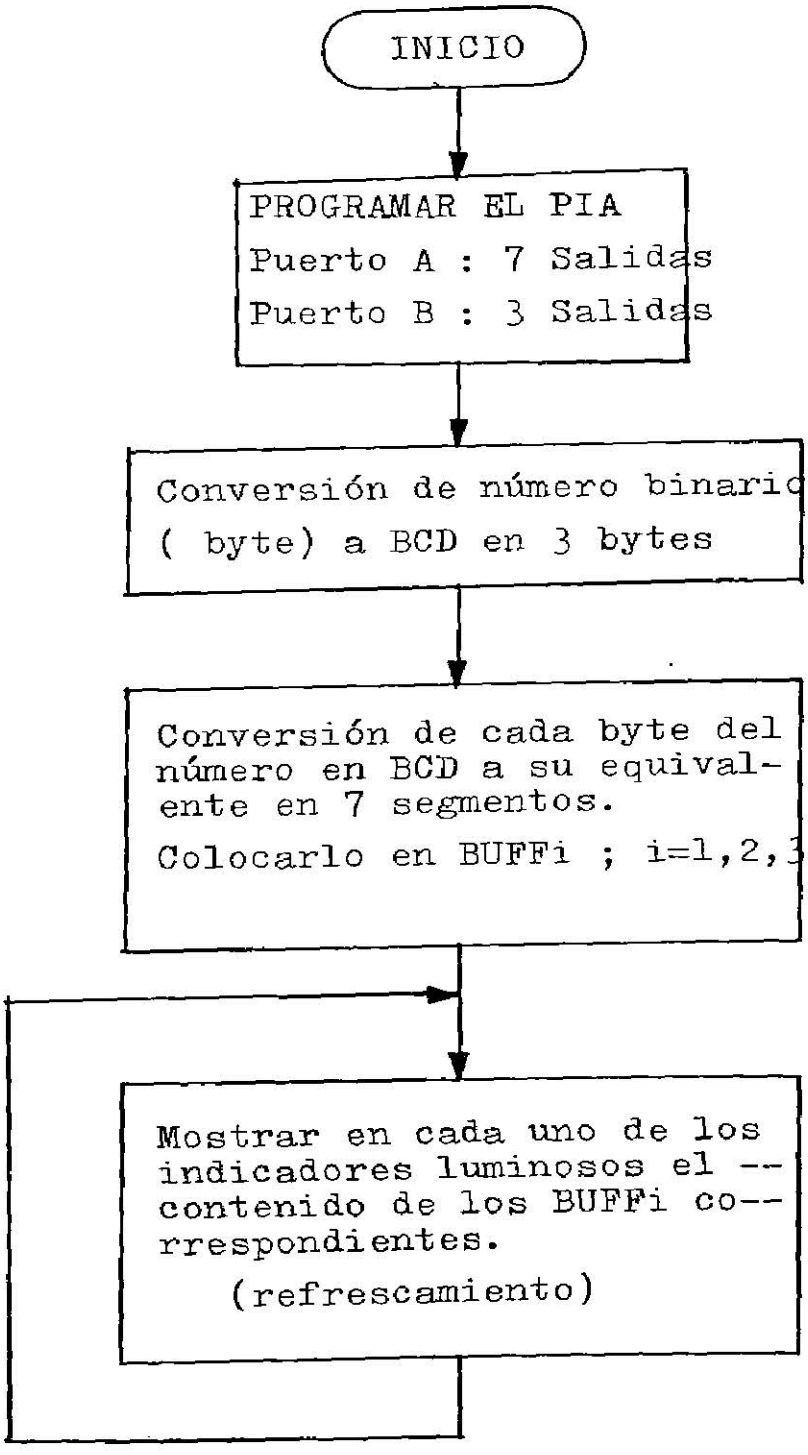

INICIO
PROGRAMAR EL PIA
Puerto A : 7 Salidas
Puerto B : 3 Salidas
Conversión de número binario
(byte) a BCD en 3 bytes
Conversión de cada byte del
número en BCD a su equival-
ente en 7 segmentos.
Colocarlo en BUFFi ; i=1,2,3
Mostrar en cada uno de los
indicadores luminosos el --
contenido de los BUFFi co--
rrespondientes.
(refrescamiento)

Figura 4.30

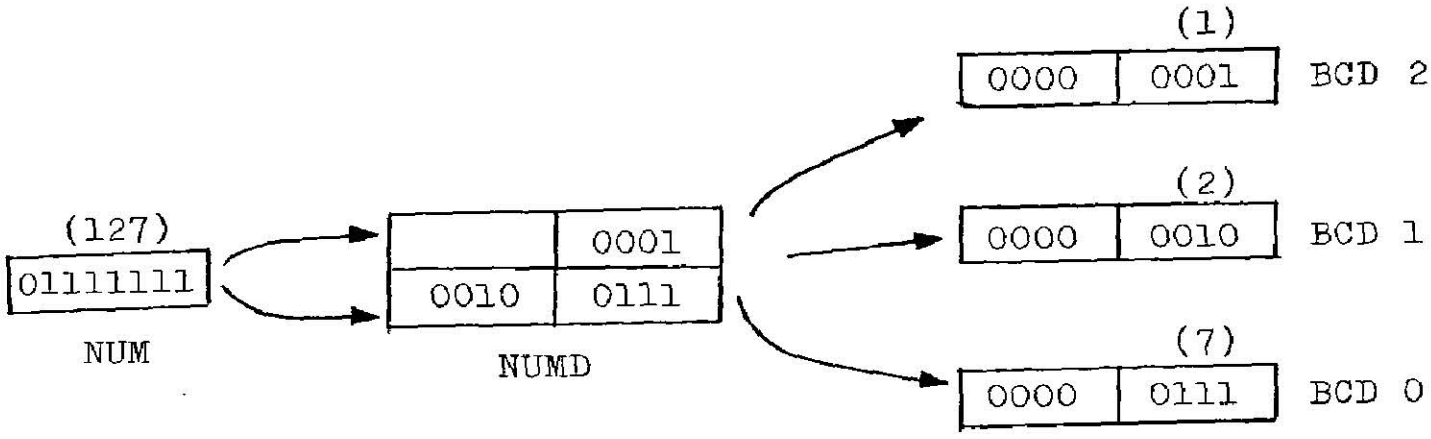


Figura 4.31

En la figura 4.32 se describe el algoritmo que realiza esta conversión con ayuda de diagramas de flujo. El cambio de base y de formato se lleva a cabo en 2 partes: una primera parte transforma el número en binario puro contenido en NUM, a un número en BCD de 16 bits, contenido en NUMD y NUMD+1 (NUMD contiene el número de mayor peso) y la segunda parte transforma el contenido no nulo de NUMD y NUMD+1 a BCD2, BCD1 y BCD0.

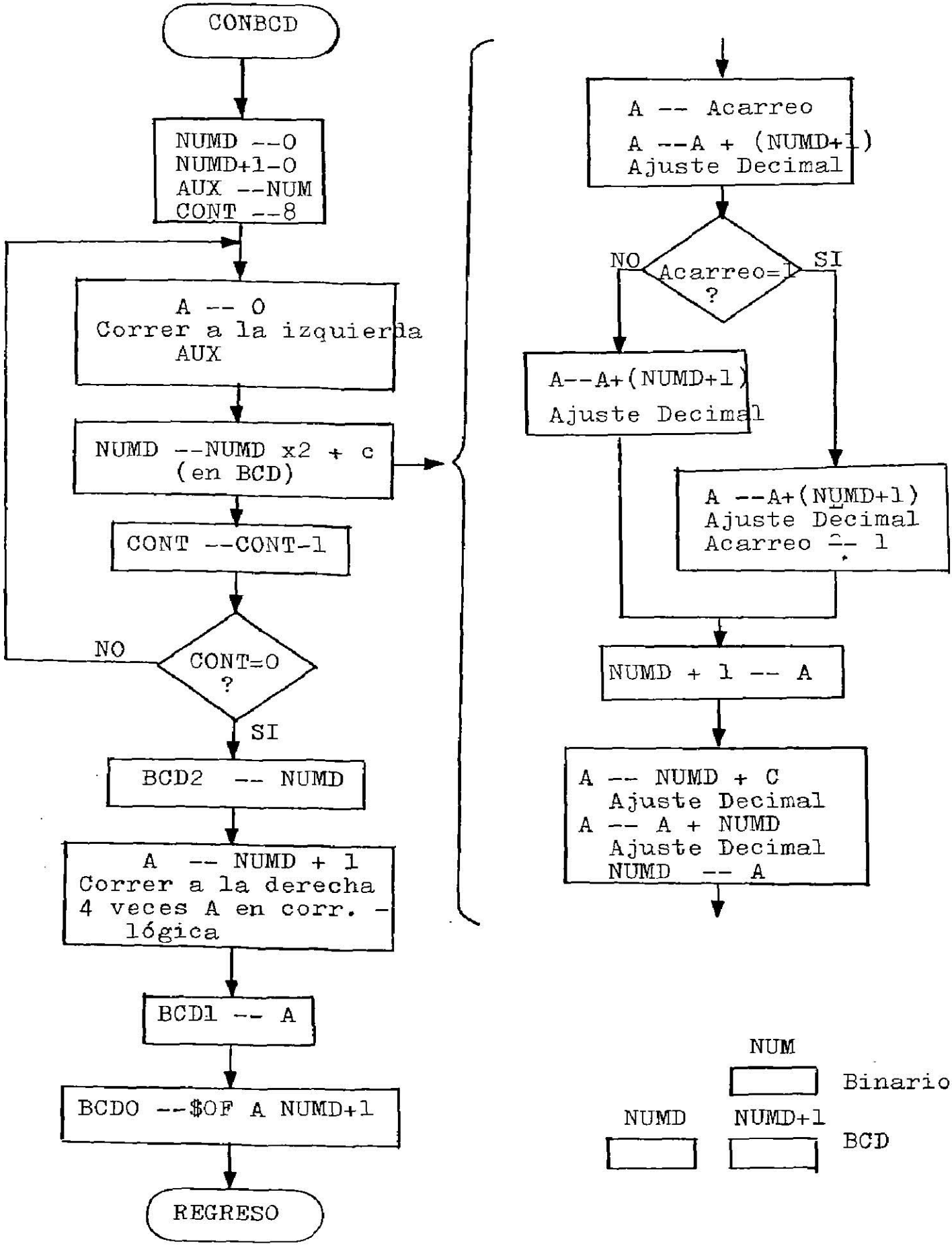
La primera transformación, que es propiamente la conversión binaria a BCD, se basa en la evaluación de un número binario en una nueva base que está dada por la expresión

$$NUMD = b_7.2^7 + b_6.2^6 + \ldots\ldots + b_0.2^0$$

donde $b_7.b_6....b_0$ son los dígitos del número binario que en este caso es de 8 bits; esta expresión se puede poner bajo la forma siguiente:

$$NUMD = (((((((b_7.2+b_6).2+b_5).2+b_4).2+b_3).2+b_2).2+b_1).2+b_0$$

que nos dá el algoritmo de conversión: multiplicaciones por 2 y sumas sucesivas que, aplicando la operación de ajuste decimal en cada resultado parcial de suma o multiplicación, nos conduce a un resultado final expresado en BCD. La programación de ésta rutina en lenguaje ensamblador se dá a continuación.


CONBCD
NUMD --0
NUMD+1-0
AUX --NUM
CONT --8
A -- 0
Correr a la izquierda
AUX
NUMD --NUMD x2 + c
(en BCD)
CONT --CONT-1
NO
CONT=0
?
SI
BCD2 -- NUMD
A -- NUMD + 1
Correr a la derecha
4 veces A en corr. -
lógica
BCD1 -- A
BCDO --$OF A NUMD+1
REGRESO
A -- Acarreo
A --A + (NUMD+1)
Ajuste Decimal
NO
Acarreo=1
?
SI
A--A+(NUMD+1)
Ajuste Decimal
A --A+(NUMD+1)
Ajuste Decimal
Acarreo -- 1
NUMD + 1 -- A
A -- NUMD + C
Ajuste Decimal
A -- A + NUMD
Ajuste Decimal
NUMD -- A
NUM
Binario
NUMD
NUMD+1
BCD

```
CONBCD    CLR     NUMD
          CLR     NUMD+1
          LDA A   NUM
          STA A   AUX
          LDA B   #$08          CONT = ACCB
RET       CLR A
          ASL AUX
          ADC A   #00           ACCA --- ACARREO
          ADD A   NUMD+1
          DAA
          BCS     PON
          ADD A   NUMD+1
          DAA
          BRA     SUM
PON       ADD A   NUMD+1
          DAA
          SEC
SUM       STA A   NUMD+1
          LDA A   NUMD
          ADC A   #00
          DAA
          ADD A   NUMD
          DAA
          STA A   NUMD
          DEC B
          BNE     RET
          STA A   BCD2          CONVERSION NUMD -- BCD1
          LDA A   NUMD+1
          LSR A
          LSR A
          LSR A
          LSR A
          STA A   BCD1
```

```
LDA A   NUMD+1
AND A   #$OF
STA A   BCOO

RST
```

4.6.4.- OBTENCION DE LOS EQUIVALENTES EN 7 SEGMENTOS

Esta decodificación se lleva a cabo extrayendo de una tabla el equivalente en 7 segmentos asociado al número en BCD;- los códigos equivalentes están ordenados en orden creciente - del número en BCA y están ubicados en localidades consecuti-- vas de memoria, figura 4.33. DIRTAB y DIRTAB+1 contienen la - dirección inicial de la tabla.

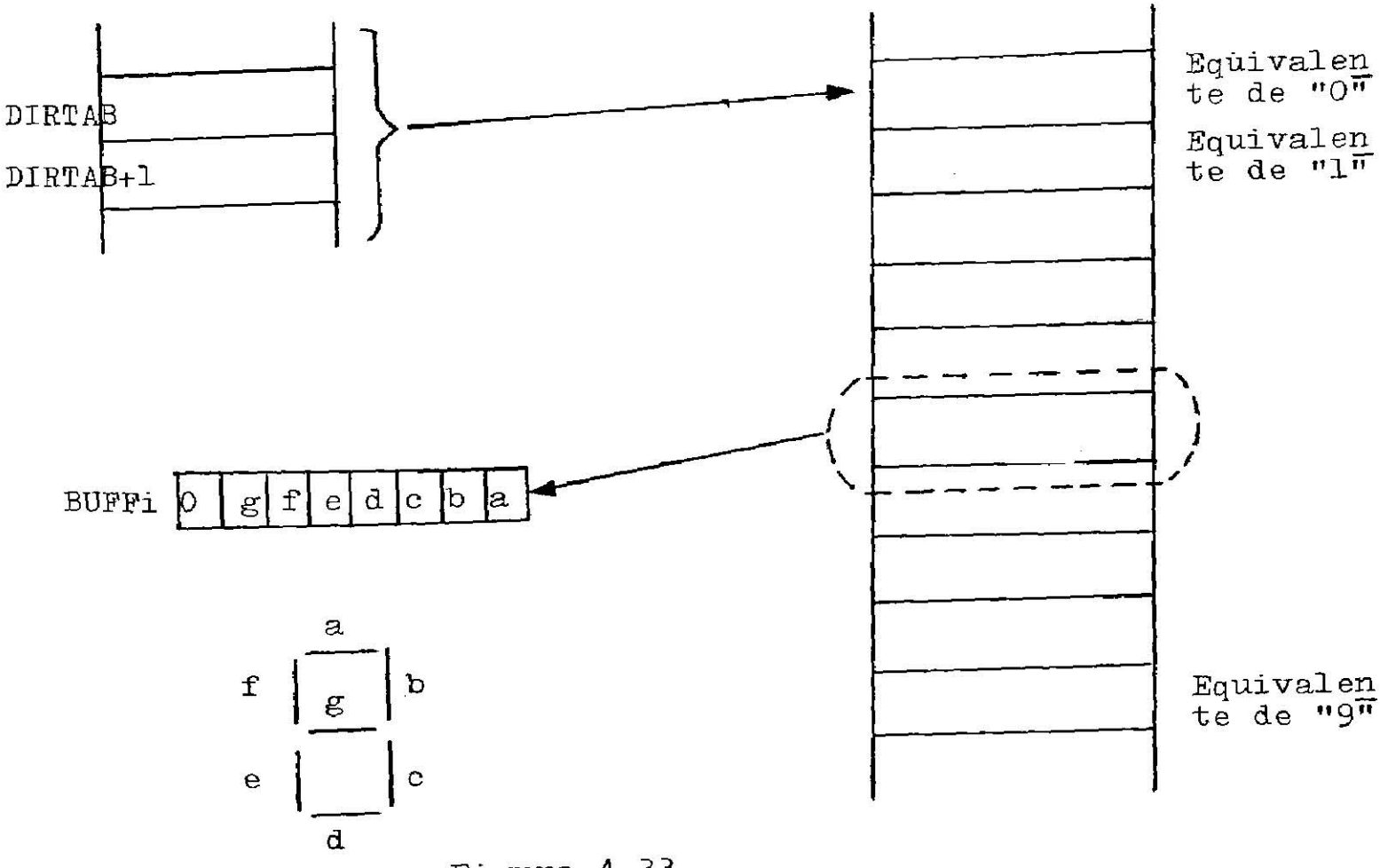


Figura 4.33

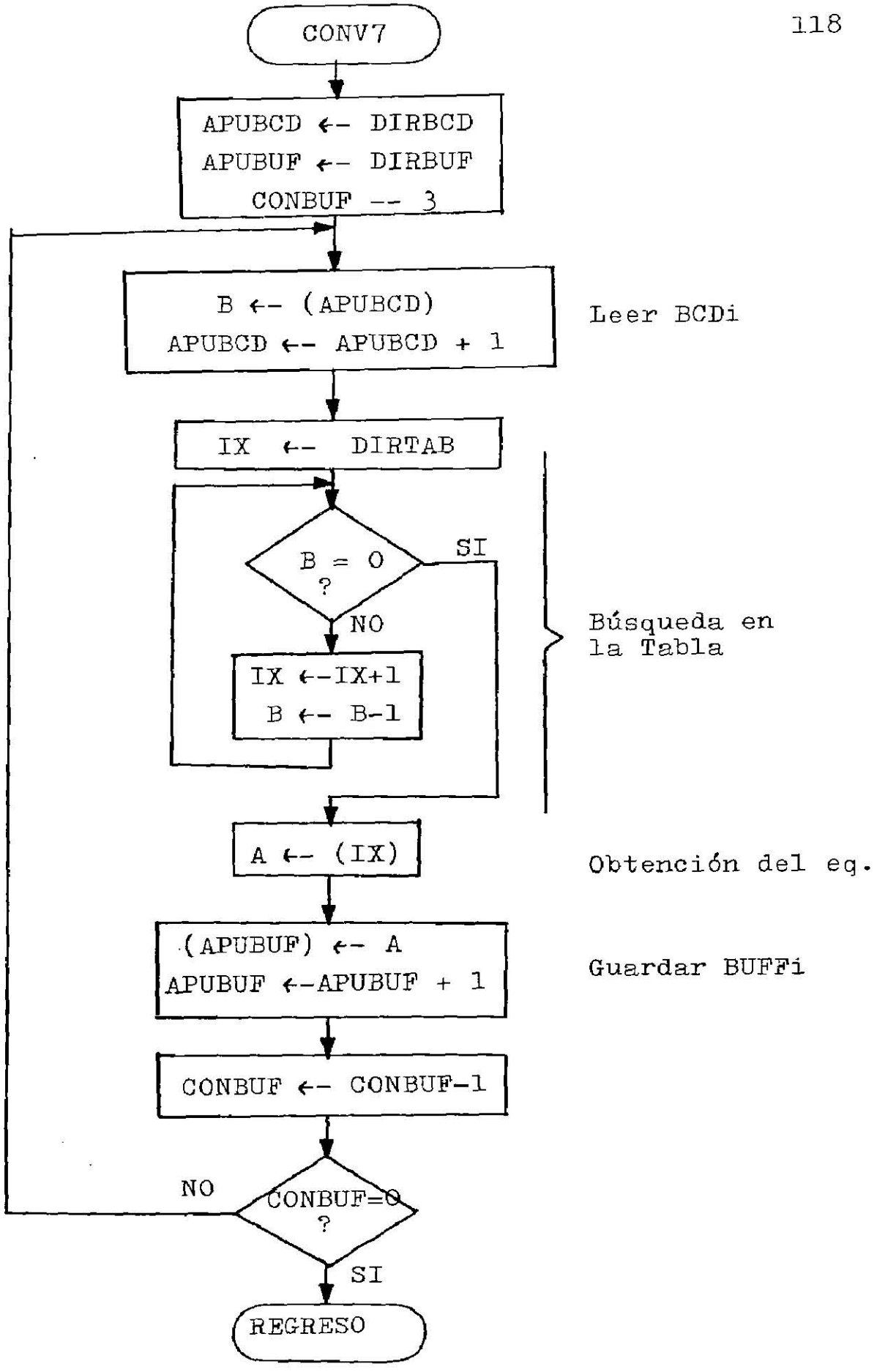

CONV7
APUBCD ← DIRBCD
APUBUF ← DIRBUF
CONBUF -- 3
B ← (APUBCD)
APUBCD ← APUBCD + 1
Leer BCDi
IX ← DIRTAB
B = 0 ?
SI
NO
IX ←-IX+1
B ← B-1
Búsqueda en la Tabla
A ← (IX)
Obtención del eq.
(APUBUF) ← A
APUBUF ←-APUBUF + 1
Guardar BUFFi
CONBUF ← CONBUF-1
NO
CONBUF=0 ?
SI
REGRESO

Figura 4.34

El algoritmo para efectuar esta decodificación está descrito mediante el diagrama de flujo mostrado en la figura 4.34. Las direcciones iniciales de las BCDi y los BUFFi se encuentran en las variables DIRBCD y DIRBUF. Se utilizan apuntadores para direccionar los BCDi a convertir(APUBCD) y para direccionar las localidades(BUFFi) donde se depositan los equivalentes en 7 segmentos(APUBUF). CONBUF contiene el número de indicadores de 7 segmentos utilizados, que en el caso actuales tres. A continuación se dá el programa en ensamblador de esta rutina.

```
CONV7    LDX    DIRBCD      INICIA APUNTADORES
         STX    APUBCD
         LDX    DIRBUF
         STX    APUBUF
         LDA A  #03
         STA A  CONBUF
OTRBCD   LDX    APUBCD
         LDA B  00,X        B ---(APUBCD)
         INX
         STX    APUBCD
BUSQ     TST B              BUSQUEDA EN LA TABLA
         BEQ    TENGO
         INX
         DEC B
         BRA    BUSQ
TENGO    LDA A  00,X        OBTENCION
         LDX    APUBUF
         STA A  00,X        GUARDA BUFFi
         INX                INCREMENTA APUBUF
         STX    APUBUF
         DEC    CONBUF
         BNE    OTRBCD
         RST
```

4.6.5.- REFRESCAMIENTO DE LOS INDICADORES

Esta parte del programa saca el contenido de los BUFFi -- por el puerto A, en el cual se encuentran conectados los ánodos de los indicadores luminosos. Se debe activar el cátodo - del indicador i cuando el contenido del BUFFi sea depositado- en el puerto A y se debe dejar activado durante un tiempo suficiente(1 ms), para poder apreciar el efecto de "encendido - contínuo" durante el "refrescamiento".

El proceso de refrescamiento se lleva a cabo haciendo un- llamado contínuo a una rutina que muestre en los exibidores - de 7 segmentos el contenido de los tres BUFFi. Dicha rutina - se describe mediante el diagrama de flujo de la figura 4.35 - y la programación en lenguaje simbólico se dá a continuación.

```
REFRES    LDX    DIRBUF      CARGA APUNTADOR
          LDA B  #$FE        CARGA UN "0" EN EL
                             PRIMER BIT
          LDA A  #$03        INICIA CONTADOR
          STA A  CONREF
SAKA      LDA A  00,X        ESCRIBE EN EL PUERTO A
          STA A  PIADRA      EL CONTENIDO DE BUFFi
          STA B  PIADRB      HABILITA EL CATODO
          INX                INCREMENTA APUNTADOR
          SEC                CORRE EL CERO PARA
          ROL B              HABILITAR
          JSR    RETIM
          CLR    PIADRA      RETARDA
          DEC    CONREF
          BNE    SAKA
          RTS
```

4.6.6.- PROGRAMA PRINCIPAL

El programa principal responde al diagrama de flujo de la figura 4.30 y está compuesto de una parte de programación del PIA y de los llamados a las subrutinas definidas anteriormente, este programa escrito en lenguaje ensamblador es el siguiente:

```
        CLR     CRA         DEFPIA
        LDA A   #$7F
        STA A   DDRA        PUERTO A : 7SALIDAS
        LDA B   #$7F
        STA B   CRA
        CLR     CRB
        LDA A   #$07
        STA A   DDRB        PUERTO B : 3 SALIDAS
        STA B   CRB
        JSR     CONBCD      BINARIO --BCD
        JSR     CONV7        BCD-7 SEGMENTOS
MUESTR  JSR     REFRES       REFRESCAMIENTO
        BRA     MUESTR
```

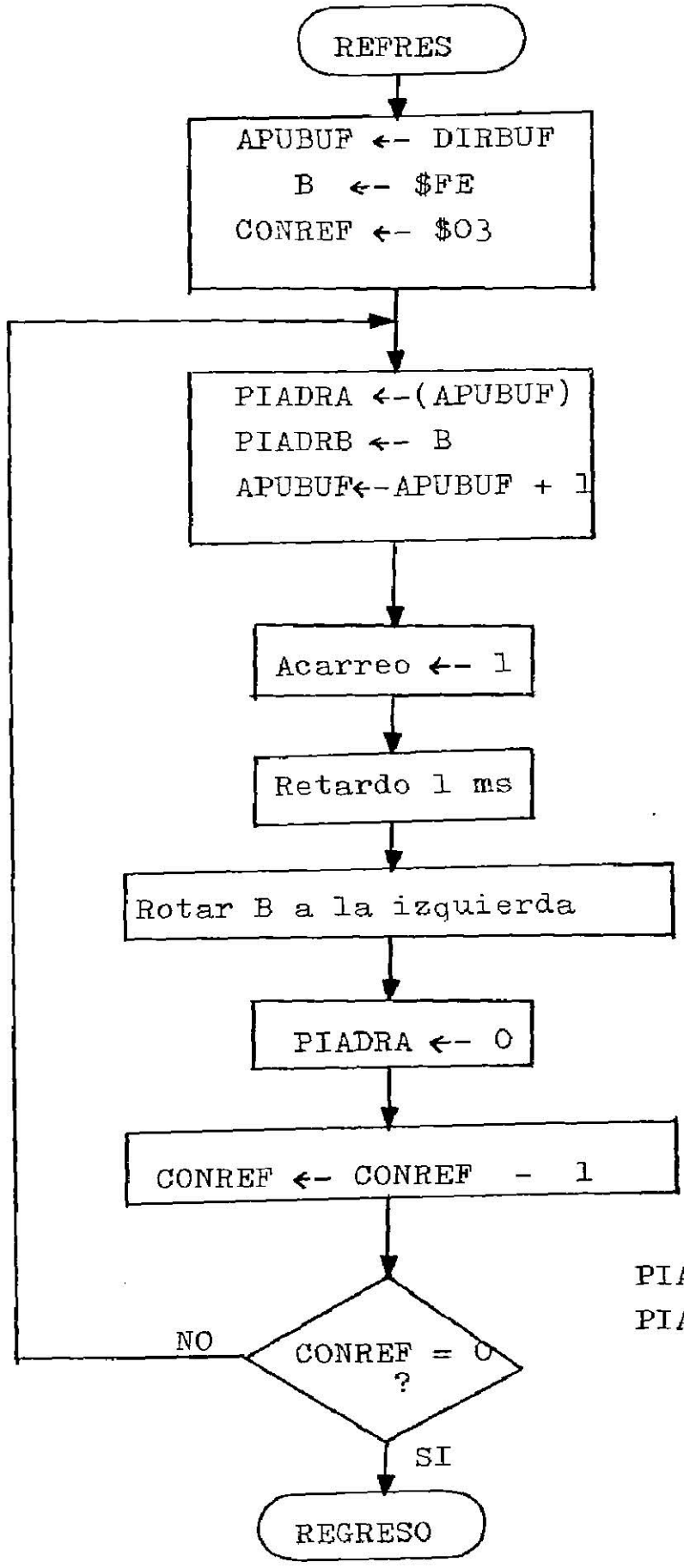
4.6.7.- CONVERSION DIGITAL - ANALOGICA

En esta sección se mostrará el uso de un convertidor digital-analógico integrado de 8 bits MC1408L8 para generar algunas formas de ondas y producir señales de un valor determinado. La figura 4.36 muestra un circuito en el cual figuran el PIA el convertidor integrado y un circuito adicional para transformar la salida del convertidor a una señal de voltaje.

La calibración del circuito se realiza como sigue:

a).-Sin conectar la red R1 P1 ponga en las entradas digitales niveles altos. Con el potenciómetro P2 ajuste la salida a 10 V.

b).- Conecte la red R1 P1 donde se indica con líneas punteadas y ajuste P1 hasta obtener en la salida 5 V.



PIADRA : Lado A del PIA
PIADRB : Lado B del PIA

Figura 4.35

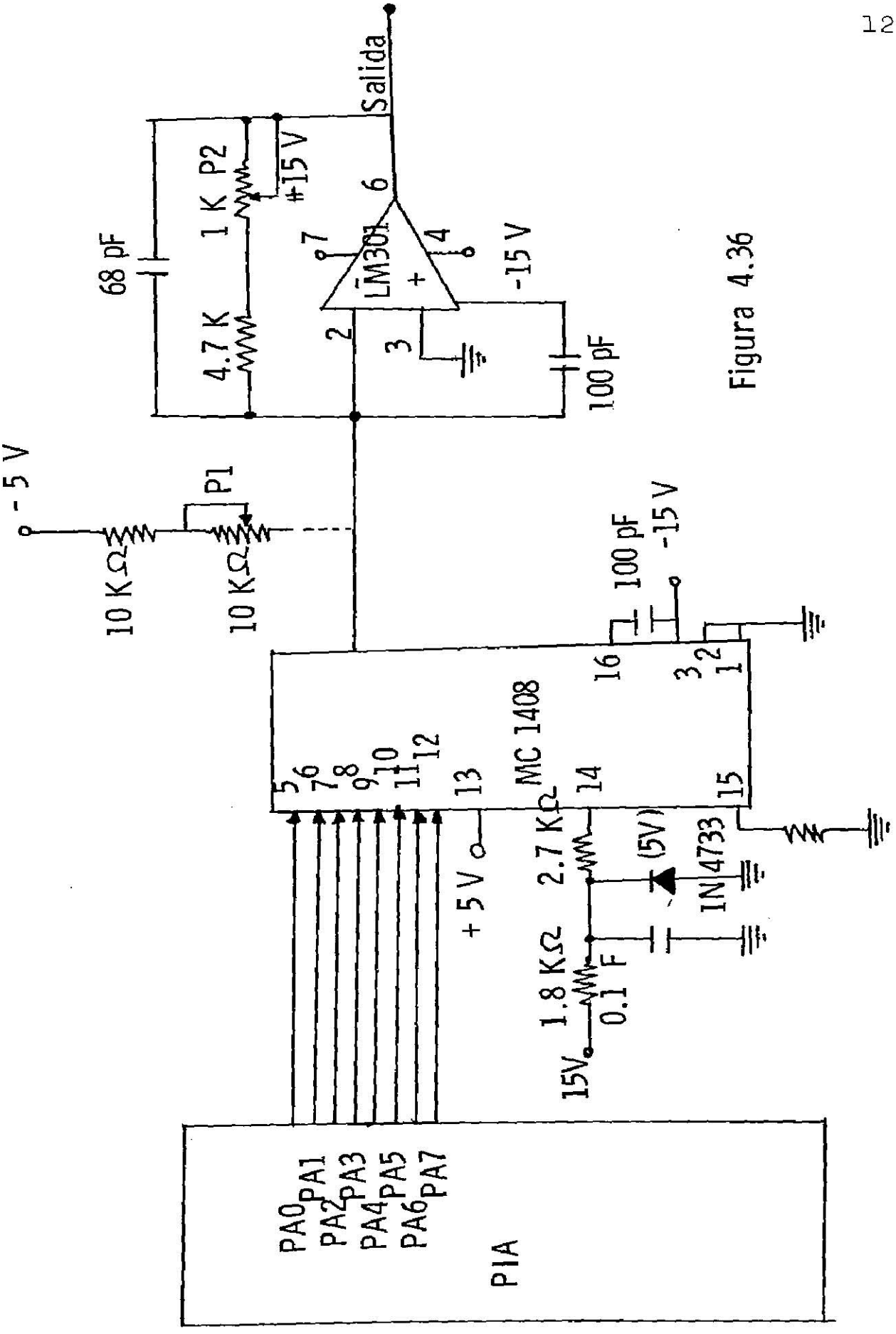

Salida
+15 V
1 K P2
68 pF
4.7 K
LM301
7
6
4
2
3
-15 V
100 pF
- 5 V
P1
10 KΩ
10 KΩ
100 pF
-15 V
16
3
2
1
5
6
7
8
9
10
11
12
13
MC 1408
14
15
2.7 KΩ
(5V)
IN4733
+ 5 V
1.8 KΩ
0.1 F
15V
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PIA

Figura 4.36

4.6.7.1.- GENERACION DE ALGUNAS FORMAS DE ONDA

Una vez calibrado el circuito, se pueden generar señales analógicas en el rango de -5 a +5 V, escribiendo en el lado A del PIA configuraciones desde $00 hasta $FF. El convertidor posee una característica valor digital-valor analógico como la mostrada en la figura 4.37.

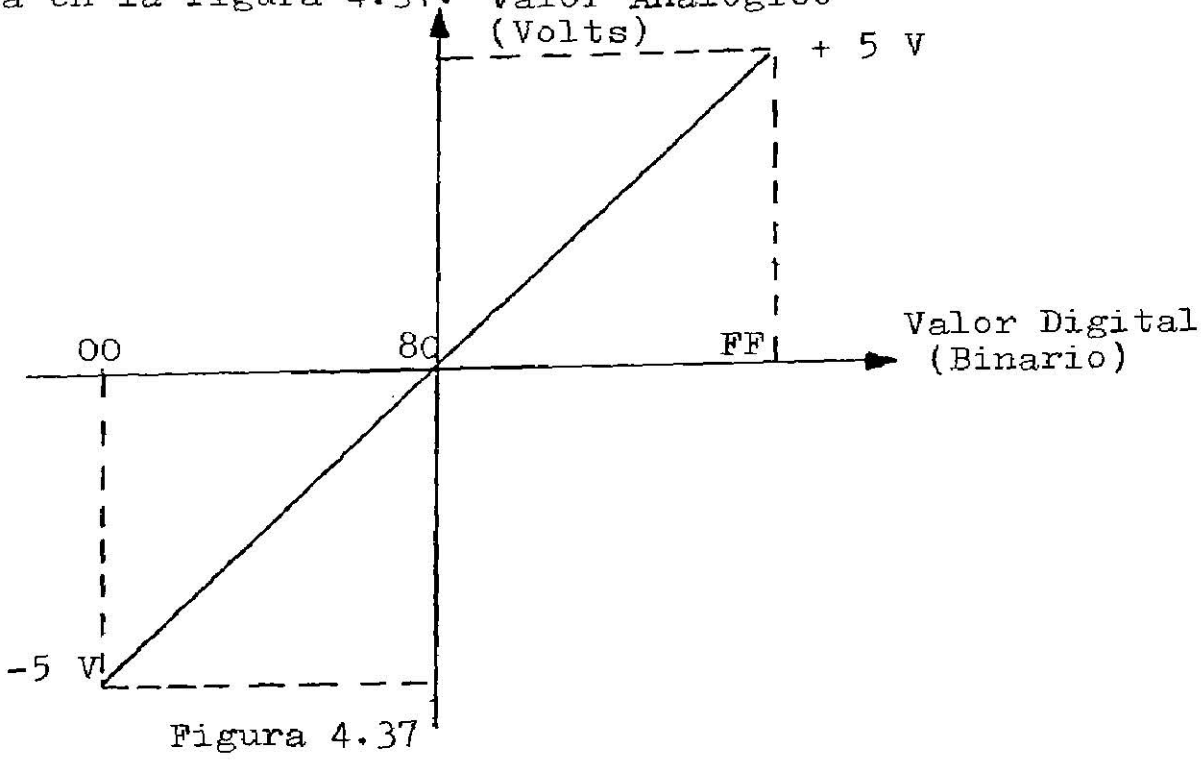


Figura 4.37

i).- Generación de una señal "diente de sierra". Esta señal mostrada en la figura 4.38 se puede generar mediante un programa bastante simple que es descrito en el diagrama de flujo de la figura 4.39. El programa escrito en lenguaje ensamblador se dá a continuación;

```
        CLR     CRA
        LDA A   #$FF
        STA A   DDRA        LADO A : SALIDAS
        LDA A   #04
        STA A   CRA
        CLR A
OTRA    STA A   PRA         INICIA CONTADOR
        INC A
        JSR     RETXM
        BRA     OTRA        RETARDO X MILISEG.
```

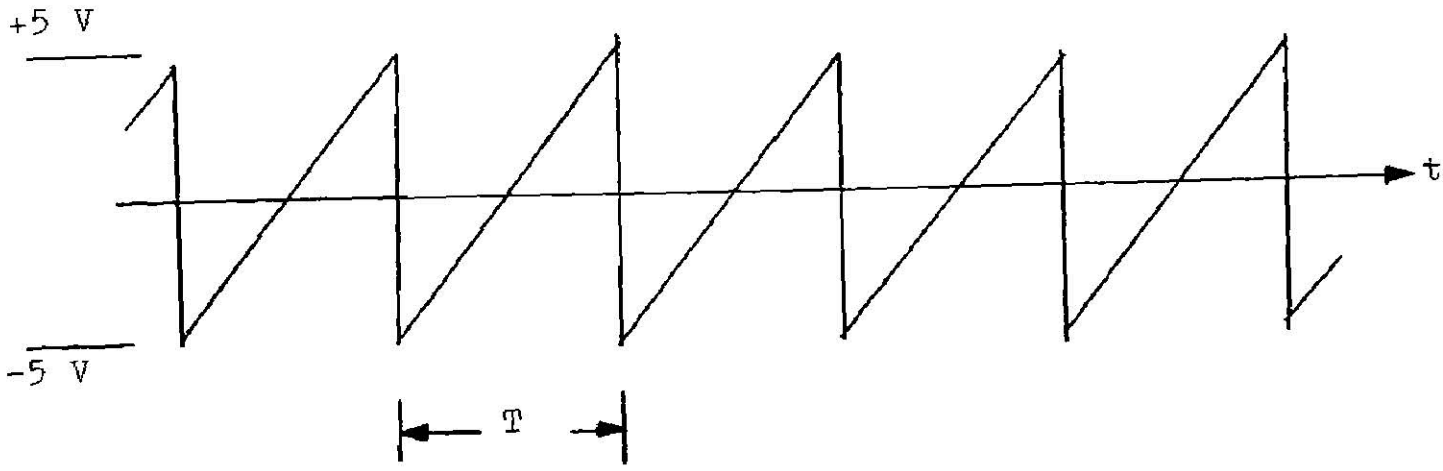


Figura 4.38

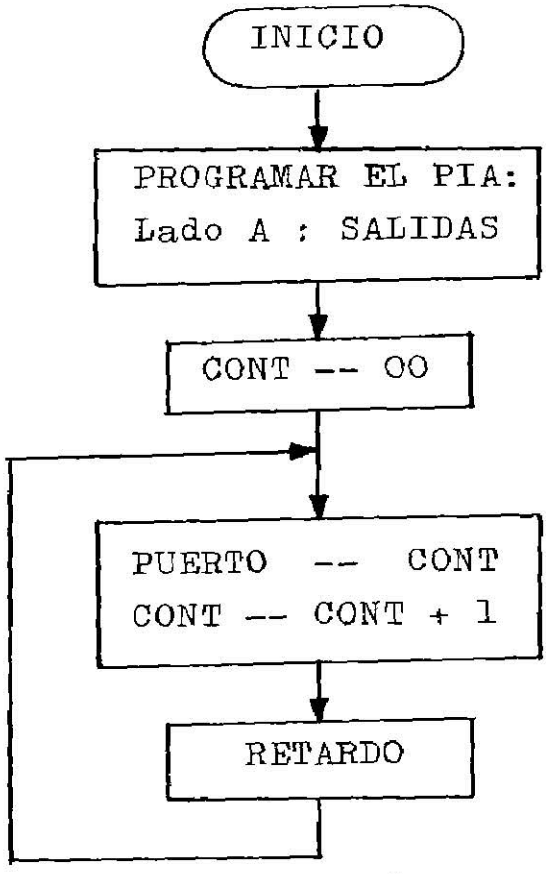


Figura 4.39

El período T es función de los tiempos de ejecución de -- las diferentes instrucciones utilizadas y el retardo, mediante el cual se puede ajustar T.

ii).- Generación de una señal triangular, una señal como la - mostrada en la figura 4.40 puede ser producida mediante la ejecución de un programa, que responde al diagrama - de flujo de la figura 4.41. El algoritmo consiste básicamente en la generación de 2 rampas, una con pendiente positiva(incremento del contador) y otra con pendiente- negativa(decremento del contador) al detectar el cambio FF--- 00.

Cada una de las pendientes pueden ser modificadas ajustando cada retardo. El programa escrito en lenguaje simbólico -- queda como sigue;

```
                CLR     CRA         PROGRAMA DEL PIA
                LDA A   #$FF        LADO A : SALIDAS
                STA A   DDRA
                STA A   CRA
SUBE            LDA A   #$00        INICIA CONTADOR
INCRE           STA A PRA
                INC A
                BEQ     BAJA
                JSR     RETXM       RETARDO 1
                BRA     INCRE
BAJA            LDA A   #$FE
DECRE           STA A   PRA
                DEC A
                BEQ     SUBE
                JSR     RETXM       RETARDO 2
                BRA     DECRE
```

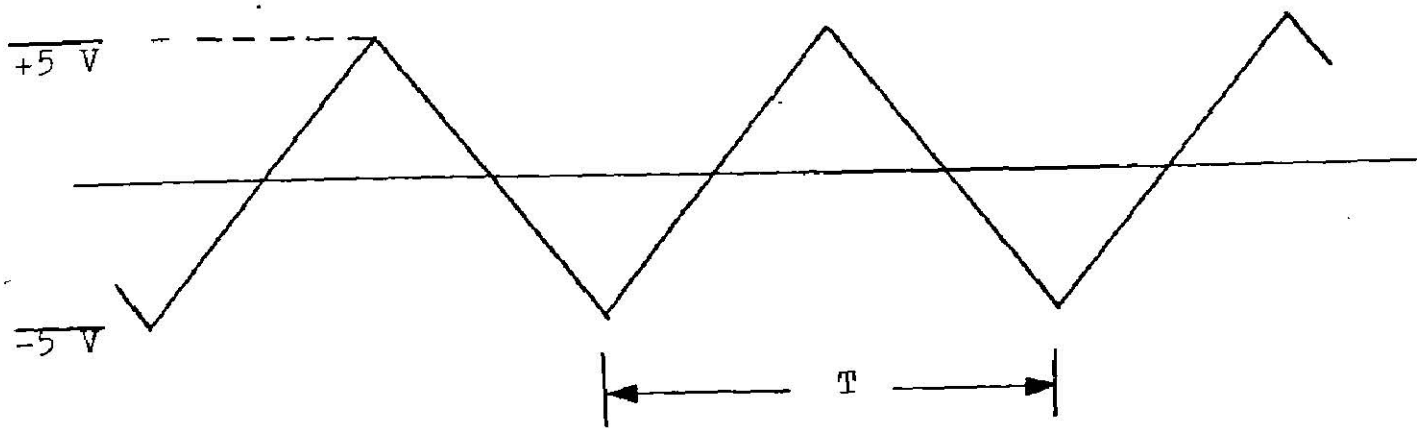


Figura 4.40

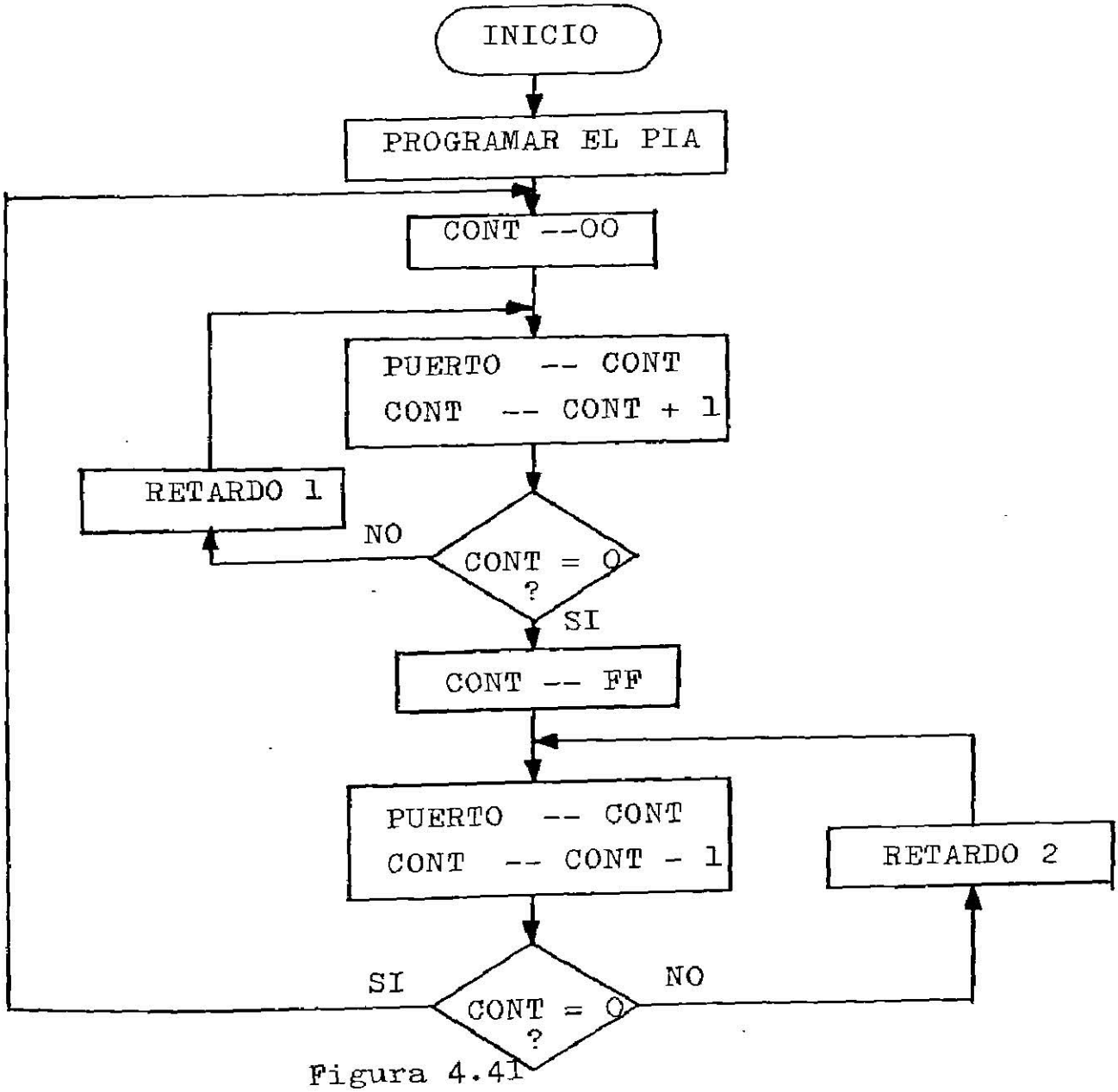


Figura 4.41

4.6.7.2.- CONVERSION DE UN NUMERO REAL A UNA SEÑAL DE VOLTAJE

Hasta ahora solo hemos generado las señales mediante el convertidor D/A sin importar su amplitud, es decir, no se ha determinada con precisión un valor de voltaje a producir. Si queremos, por ejemplo, generar una señal con un valor 2.5 volts, es necesario poner en la entrada digital la configuración $00 y no $02.80, que es el número real 2.5 expresado en complemento a 2 en 16 bits.

El problema es, entonces, convertir un número real en punto fijo expresado en complemento a 2 en un formato de 2 bytes a un entero de un byte que responde a las características de la figura 4.38; este problema se soluciona al emplear la transformación lineal:

$$Y = (X + 5).128/5$$

donde X es el número real y Y el entero.

4.6.8.- CONVERSION ANALOGICA A DIGITAL

En esta sección se ve la forma de transformar un valor de una señal analógica a su equivalente numérico digital expresado en complemento a 2 en un formato de punto fijo en 2bytes.- Dicha transformación puede ser llevada a cabo mediante el uso de un convertidor A-D integrado o utilizando alguna de las técnicas de conversión A-D por programación.

El uso de un convertidor integrado no presenta problema alguno, solo es necesario conocer las especificaciones de las señales de entrada y de alimentación y el formato de la información digital de salida.

La conversión A-D por programación, utilizando un convertidor A-D integrado representa la ventaja de que el conjunto es más barato, aunque el tiempo de conversión puede ser relativamente lento. Presentaremos una técnica llamada de "aproximaciónes sucesivas" para la cual es necesario un circuito como el mostrado en la figura 4.42. Se utiliza el convertidor D-A de la sección anterior y además un circuito comparador --

cuya salida se conecta a una entrada(PBO) del PIA. Las entradas del comparador son: la salida del convertidor D-A y señal de voltaje a convertir.

El algoritmo de conversión por aproximaciones sucesivas - se describe en el diagrama de flujo de la figura 4.43 y consiste basicamente en una búsqueda dicotómica del voltaje de - entrada, que converge en un número finito de pasos igual al - número de bits del convertidor D-A.

El programa en lenguaje simbólico que realiza las tareas- descritas en el diagrama de flujo se dá a continuación. Para- obtener el equivalente digital en un formato de punto fijo de 2 bytes expresado en complemento a 2 (X) solo es necesario -- efectuar la siguiente transformación lineal X=5/128.Y-5.

```
            CLR A  CRA
            LDA A  #$FF          LADO A : SALIDAS
            STA A  DDRA
            LDA A  #04

            STA A  CRA
            CLR    CRB
            LDA B  #FE           LADO B : PBO ENTRADA
            STA B  DDRB
            STA A  CRB

            LDA B  #$08
            CLR A                INICIA CONTADOR
            STA A  LIMINF
            COM A
            STA A  LIMSUP

MEDIA       LDA A  LIMSUP
            SUB A  LIMINF
            LSR A
            STA A  Y             DIVISION POR 2
            STA A PRA            SACAR Y AL PUERTO A

            LDA A  PRB
```

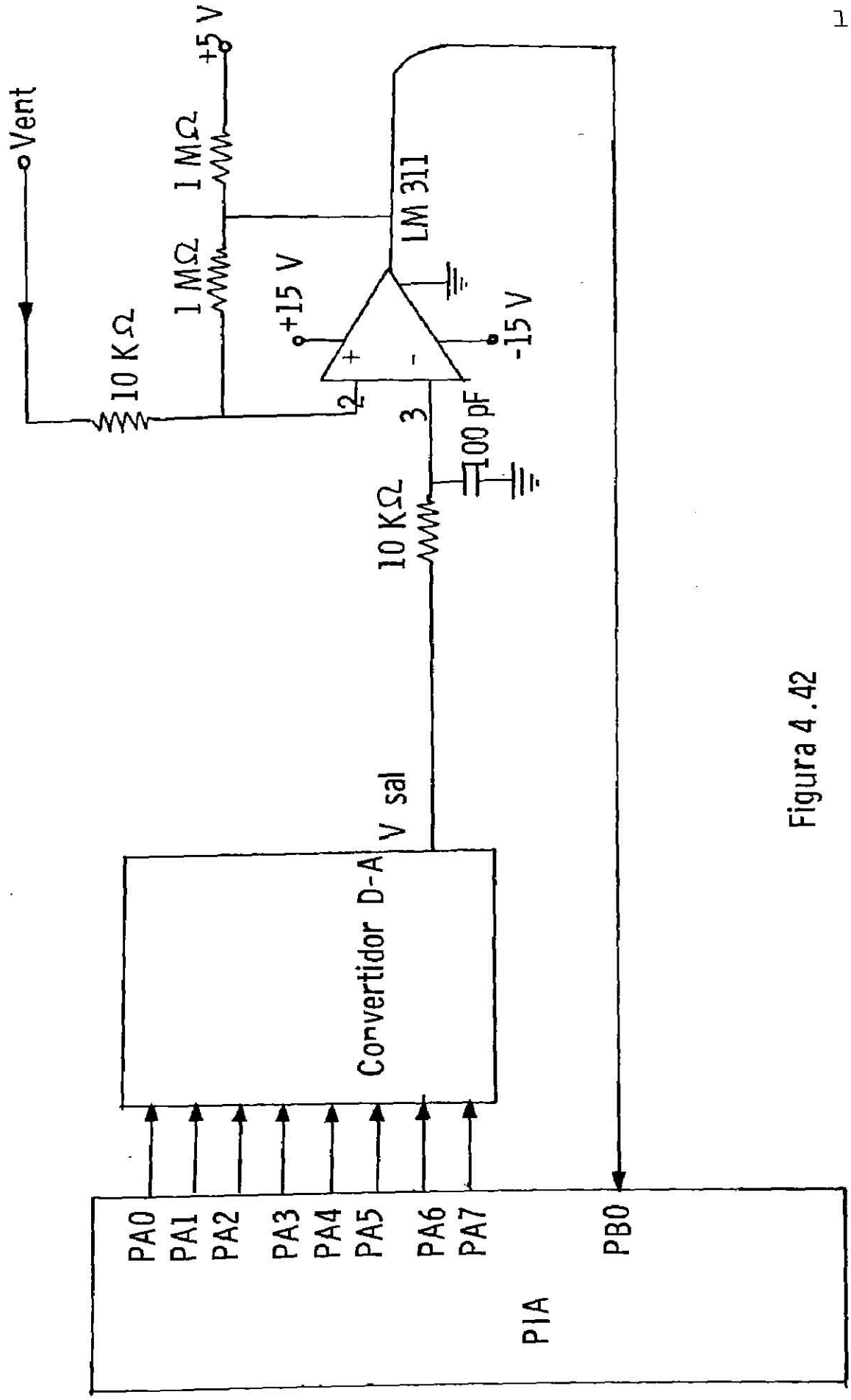
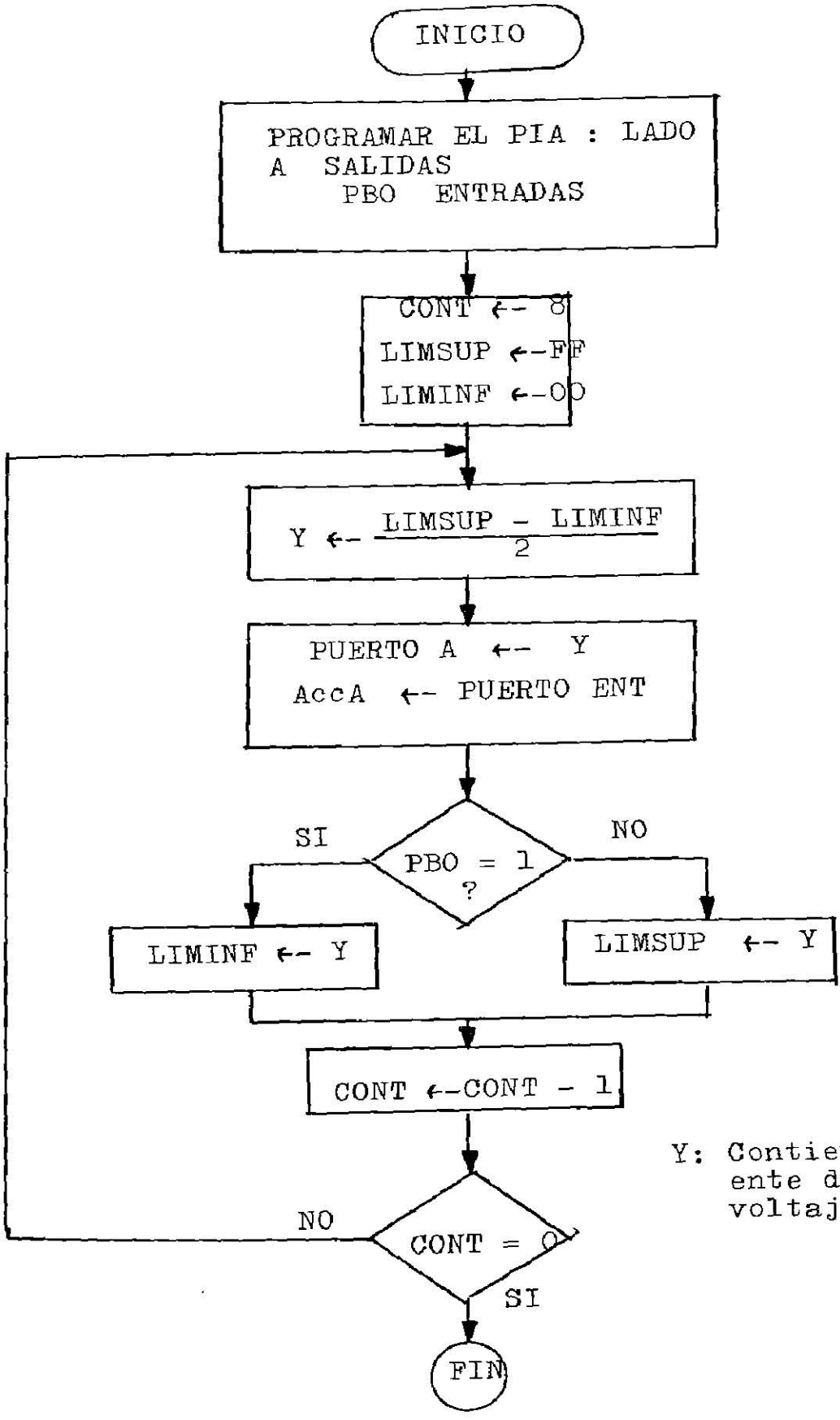

Vent
+5 V
1 MΩ
1 MΩ
10 KΩ
+15 V
LM 311
-15 V
2
3
100 pF
10 KΩ
V sal
Convertidor D-A
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PIA

Figura 4.42



Y: Contiene el equivalente digital del voltaje de entrada

Figura 4.43

```
                BIT A   #01         PBO = 1 ?
                BNE     CAMB
                LDA A   Y
                STA A   LIMSUP
                BRA     DECRE
CAMB            LDA A   Y
                STA A   LIMINF
DECRE           DEC B
                BNE     MEDIA
                SWI
```

5.- E L S O F T W A R E

D E L M C 6 8 0 2

5.1.- SOFTWARE Y DESCRIPCION

Este capítulo describe las subrutinas del D5BUG, el software que contiene y otra información útil para el MEK 6802 -- D5.

Descripción y Subrutinas:

- Reset: ($F000)
  Rutina: Reset por hardware
  Descripción: Enmascara o inhibe la interrupción
  Borra todo el RAM de $E400- E47F
  Inicializa el sistema de PIA
  Estabiliza por default el uso del SP($E418)
  Pasa el control a la rutina PROMPT

- Prompt: ($F024)
  Rutina: Reset por software
  Descripción: El sistema inicializa el estock pointer($E47E)
  Por software pone las siguientes banderas
  ROLPAS = 1
  UPROG = 0
  ROIFLG = 0
  KYFLG = 0
  FNCFL = 0
  Borra los 7 segmentos de los displays
  Despliega el símbolo (-)
  Estabiliza FUNSEL como programa activo principal

- GET: ($F04E)
  Rutina: Subrutina para leer y descifrar
  Efectos: Todos los registros son alterados por esta rutina
  El registro A a su vez obtiene el código de la tabla
  En RAM KYFLG tendrá $01 a su vez se localiza - el código de tecla oprimida

Rutinas usadas:

DLY25

Requerimientos del Stack: 4 Bytes

Descripción: Las teclas del tablero están eléctricamente arregladas dentro de 6 filas y 4 columnas. La tecla "RS"(reset) no forma parte de este arreglo. El GET se llama cuando una tecla se oprime y retorna 25 mseg. después de que la tecla se deja de oprimir.

Cuando la tecla se oprime, el GET busca en la matriz para encontrar su fila y su columna. La información de la fila/columna se usa para direccionar la tabla de la cual obtiene el código de la tecla.

Este código es almacenado en RAM y también en el acumulador A cuando el GET retorna. Una localización RAM llamada KYFLG es puesta $01 para indicar que una tecla entra, cuando otra -- rutina reorganiza esta bandera, esta deberá -- leer el código de la tecla y poner en KYFLG -- $00.

TECLAS DE LAS FUNCIONES

TECLA	CODIGO
MD	80
EX	81
RD	82
GO	83
FS	84
FC	85
P/L	86
T/B	87

Nota: Los códigos de las funciones tienen MSB para poder examinar en forma negativa el Código de Condición.

SUMARIO DE LAS TECLAS Y CODIGOS

TECLA	FILA	COLUMNA	CODIGO
0	0	0	00
F	0	1	0F
E	0	2	0E
D	0	3	0D
1	1	0	01
2	1	1	02
3	1	2	03
C	1	3	0C
4	2	0	04
5	2	1	05
6	2	2	06
B	2	3	0B
7	3	0	07
8	3	1	08
9	3	2	09
A	3	3	0A
FS	4	0	84
FC	4	1	85
P/L	4	2	86
T/B	4	3	87
MD	5	0	80
EX	5	1	81
RD	5	2	82
GO	5	3	83

- PUT: ($FOBB)

  Rutina: Programa de utilidad para mostrar información en los displays de 7 segmentos y llamar al programa principal activo como una subrutina una vez cada milisegundo.

  Efectos: Los acumuladores A y B y el registro de índice se alteran por la rutina PUT.

Rutinas usadas:

DLY1, rutina especificada por MNPTR.

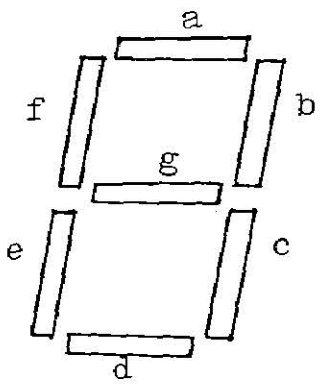
Requerimientos del Stack: 1 además de los requerimientos del programa principal(mínimo 3 bytes).

Descripción: La rutina PUT es el corazón del sistema operativo porque los displays deben borrarse contínuamente. La información del ánodo para el display deseado se presenta a las líneas del ánodo en paralelo y el cátodo apropiado se habilita encendiendo los segmentos correspondientes al dato. Un dígito dado permanece encendido alrededor de 1 milisegundo después el cátodo se deshabilita, la información del ánodo cambia al siguiente dígito y entonces se habilita el siguiente cátodo.

Durante el corto período entre dígitos se ejecuta una subrutina cuya dirección está almacenada en la localidad MNPTR de la RAM. Sobre el RESET o PROMPT, el MNPTR se coloca en la dirección de la rutina del Function Select(FUNSEL). Cuando el FUNSEL desea pasar el control a alguna otra rutina, almacena la dirección de esa rutina en el MNPTR el cual activa la nueva rutina(y desactiva el FUNSEL). Otra forma de hacer esto es decir que la rutina de display llama a "algún programa" una vez por milisegundo y ese "algún programa" es el programa cuya dirección se almacena en la dirección MNPTR de RAM.

La rutina del PUT obtiene el segmento de información de un buffer de 6 bytes llamado DISBUF, los cuales contienen 7 segmentos de datos para los displays correspondiendo el primer byte a el display de más a la izquierda. Además el acumulador B lleva la cuenta de cuál dígito será habilitado con el bit tercero de la izquierda correspondiendo a el display de más a la izquierda y el bit de menor peso corresponde a el display de más a la derecha.

Nota: Debido a que las interfases del ánodo son de lógica negativa la rutina PUT invierte el dato en el DISBUF.



El punto decimal no se usa en el MEK 6802D5

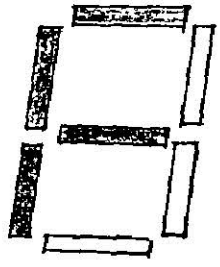
MSB							LSB
No se usa	g	f	e	d	c	b	a

Si un bit tiene un uno su segmento correspondiente estará encendido(Para DISBUF solamente).

Ejemplo: Para mostrar el carácter "F" el dato en el DISBUF será:

a = 1
b = 0
c = 0
d = 0
e = 1
f = 1
g = 1
No se
usa = 0

01111 0001 6 $71



- DISCOD: ($F120)
  Rutina: convierte un dato hexadecimal del HEXBUF en un código de 7 segmentos y lo almacena en el DISBUF.
  Efectos: El contenido de los acumuladores A y B, y el registro índice se guardan y realmacenan antes de retornar al -- DYSCOD.
  A su vez la localidad de RAM del DISBUF al DISBUF+5 contiene los códigos de los 7 segmentos para el dato hexadecimal- el HEXBUF a HEXBUF+2.
  Rutinas usadas: ADDAX
  Requerimientos del Stack: 9 Bytes
  Descripción: Los 6 códigos de 4 bits de información hexadecimal del HEXBUF a el HEXBUF+2 se convierten en un byte de- información para los ánodos(7 segmentos) la cual se almacena en el buffer del display de 6 bytes(DISBUF). Los 4 bits- más significativos de una localidad del HEXBUF corresponden a la localidad del DISBUF y los 4 bits menos significativos del HEXBUF+2 corresponden a la localidad DISBUF+5.
  Resultando una información de 7 segmentos que se muestra en los 6 displays del MPU por la rutina PUT. El primer byte -- del DISBUF corresponde al display de más a la izquierda y - el sexto byte(DISBUF+5) corresponde al display de más a la- derecha.

- DLY25, DLY1, DLYX ($F169) ($F171) ($F179)
  Rutina: Estas rutinas causan retardos de 25 milisegundos, - un milisegundo, o un retardo variable basado en el valor -- de el registro índice.
  Efectos: El contenido del registro de índice se guarda y se vuelve a almacenar antes de retornar de las rutinas. Los -- acumuladores A y B no se alteran por las rutinas de retardo.
  Requerimientos del Stack: 2 Bytes
  Descripción: Al llamar al DLY25 o el DLY1 se ocaciona un retardo en el tiempo de ejecución de cerca de 25 milisegundos o un milisegundo respectivamente. Al llamar al DLYX se ocaciona un retardo basado en el valor contenido en el regis--

tro índice de acuerdo con lo siguiente:

(X-1) (8CIC) (1.11763 US/CIC) + (27.94 US)

Un retardo mínimo de 27.94 nos resulta cargando el registro índice con valor de 1($0001) y un valor máximo de 0.5842 segundos resulta cargando el registro índice con valor de cero($0000)-1 = $FFFF =65535.

- ADDAX: ($F183)
  Rutina: Suma el acumulador A con el registro índice.
  Efectos: El acumulador A y el registro de índice se alteran por ADDAX.
  Requerimientos del Stack: 2 Bytes
  Descripción: El valor del registro A es sumado al valor del registro índice y el resultado se suma al registro índice.

- CLRDS: ($F195)
  Rutina: Borra los dígitos del display numérico para inhibir en el registro A.
  Efectos: El registro A es alterado por CLRDS. Los contenidos del registro índice se guardan y vueltos a almacenar antes de ser regresados del CLRDS.
  Requerimientos del Stack: 2 Bytes
  Descripción: Los 6 bits de menor peso del acumulador A corresponden a uno de los displays del MPU donde el bit de menor peso corresponde al dígito de más a la derecha. Para cada bit que se encuentra en un 1 lógico resultará un display o un dígito apagado.
  Ejemplo: 0011 1100($3C) borrará los primeros 4 dígitos del-display.

- ROLL2 ($F1AA)
  Rutina: Rutina de entrada numérica para valores de 2 dígitos (1 byte)
  Requerimientos: Los 4 bits menos significativos del acumulador contienen el nuevo dígito hexadecimal y los 4 bits más-significativos deben ser borrados.

La bandera de ROLPAS(cuando está puesta) indica "el primer-paso" el cual resulta en la localidad que ha sido borrada - anteriormente para el barrido o corrimiento del dígito nuevo que entra por la derecha.
Efectos: El acumulador A se altera por el ROLL2, el contenido del registro de índice se guarda y almacena antes de retornar del ROLL2.
Requerimientos del Stack: 2 Bytes
Descripción: La dirección de la localidad a ser operada por ROLL2 es especificada en HEXBUF y HEXBUF+1. Un dígito nuevo en hexadecimal se pasa a ROLL2 en los 4 bits menos significativos del acumulador A. Este nuevo dígito se desliza de - la derecha tal que el dígito que estaba en los 4 bits menos significativos de la localidad se encuentran ahora en los 4 bits más significativos. El dígito que estaba en los 4 bits más significativo se pierde. Si la bandera de ROLPAS no fué igual a cero entonces la localidad se borra antes de recircular el nuevo dígito.
El ROLPAS se borra para señalar que no durará más el primer paso.

- ROLL4: ($F1CC)
Rutina: Rutina de entrada numérica para valores de 4 dígi-- tos(2 Bytes).
Requerimientos: La bandera de ROLPAS(cuando está puesta) -- indica el "primer paso" el cual resulta en las localidades-HEXBUF y HEXBUF+1 que han sido borrados anteriormente para-recircular el dígito nuevo de la derecha. Los 4 bits más -- significativos del acumulador A deben borrarse.
Efectos: El acumulador A se altera por el ROLL4, el acumulador B se guarda y es recobrado antes de regresar del ROLL4.
Requerimientos del Stack: 3 Bytes
Descripción: Las localidades hacia las cuales el ROLL4 desliza el nuevo dígito son HEXBUF y HEXBUF+1 el dígito nuevo-es pasado al ROLL4 en los 4 bits menos significativos del - acumulador A. Este nuevo dígito se desliza hacia los primeros 2 bytes del HEXBUF tal que el dígito nuevo finaliza en-

los 4 bits menos significativos del HEXBUF+1.
Si la bandera ROLPAS no fué igual a cero entonces el HEXBUF y el HEXBUF+1 se borrarán antes para deslizar el dígito nuevo y el ROLPAS se borrará para indicar que no durará más el primer paso.

- RDKEY: ($FIEF)

Rutina: Lee y reconoce una tecla del tablero.
Requerimientos: Normalmente el RDKEY se llamará debido a -- que la bandera del KYFLG fué colocada indicando que una tecla a sido presionada.
Efectos: El valor de la tecla regresa al acumulador A el -- KYFLG se borra para reconocer que la tecla ha sido identificada.
Requerimientos del Stack: 2 Bytes
Descripción: Carga el código de la tecla del KEY en el acumulador A y borra la bandera del teclado KYFLG para reconocer que la tecla ha sido identificada.

- TIN: ($F533)

Rutina: Lee un byte de datos de la cinta.
Efectos: El contenido de los acumuladores A y B se altera-- por el TIN. A su vez la localidad del byte($E459) contiene-los caracteres que fueron recobrados de la cinta. El acumulador A también contiene los datos recobrados.
Rutinas usadas: FEDGE
Requerimientos del Stack: 4 Bytes
Descripción: El TIN es usado para leer un caracter en un -- formato(Kansas City Standard) de la cinta del cassette.
El caracter del formato es:
a).- "0" lógico tiempo de bit = 4 ciclos de 1200 Hz.
b).- "1" lógico tiempo de bit = 8 ciclos de 2400 Hz.
c).- Los caracteres consisten de:
    - Un 0 lógico como un bit de inicio.
    - 8 bits de datos tiempo de bit(iniciando con LSB).
    - Al menos dos "1" lógicos como bit de "alto".

Para asegurar una recuperación de datos confiables se usan-

técnicas de sincronía por software y algoritmos de tolerancia de error. La rutina TIN controla desde antes del bit de inicio hasta después del octavo bit de datos en serie.
Durante el bit de STOP los datos anteriores se procesan por otro software a una velocidad de 300 Baud el bit de STOP dura 13.3 milisegundos. Cuando se leen caracteres sucesivos - de datos, el software que procesa el byte leído previamente debe terminar a tiempo, para llamar el TIN por lo menos un- ciclo completo antes de que el próximo bit de sincronía inicie. Esto permitirá al TIN recobrar sincronización con el - flujo de datos en serie.
El tiempo requerido de el último ciclo de instrucciones que llama al TIN hasta la primer prueba es exactamente de 47 ciclos del procesador o 52.5 microsegundos.
El tiempo desde el fin del último ciclo del último bit hasta terminar las instrucciones de TINs RTS es de 100 ciclos- o 111.2 microsegundos. Generalmente será fácil procesar el- caracter previo a tiempo para regresar al TIN para los ca-- racteres sucesivos.

- PNCHB: ($F608)
Rutina: Formatea y coloca un dato de un BYTE hacia la interfase de Cassette.
Efectos: El contenido de los acumuladores A y B se alteran- durante esta subrutina.
Rutinas usadas: BIT 0, BIT 1, INVRT
Requerimientos del Stack: 6 Bytes
Descripción: Referirse a la rutina TIN para el formateo de- los caracteres de la cinta. Se usan técnicas de sincronía - por software para crear un flujo serie de ciclos de dura--- ción propia para formar señales de audio del standard de -- Kansas City. En los bytes de datos que se están grabando es vital mantener una sincronía entre bytes sucesivos no necesariamente entre el caracter. Esta subrutina consume exactamente 35 ciclos de máquina desde el llamado de la subrutina hasta la primer transición del bit enviado.

La sincronía se controla automáticamente por la subrutina - mientras se envía el caracter.
Después de transmitir la segunda última transición del último bit del stock el PNCHB regresa. La transición la cual -- marca la terminación del último ciclo del bit de stock también marca el principio del siguiente bit de inicio para el siguiente caracter. Es proporcinal como la primer transi--- ción transmitida por la siguiente llamada PNCHB. El tiempo- permitido del último ciclo de PNCHBs RTS hasta la transi--- ción siguiente debe ser transmitida en 159 ciclos del MPU.- Los 159 menos 35 para entrar al PNCHB el tiempo siguiente - deja 124 ciclos los cuales deben de consumirse por un software externo. Por ejemplo, si toma 50 ciclos para prepàrarse para el siguiente caracter los 74 ciclos que quedan(124--50) deben de consumirse antes de llamar al PNCHB de nuevo.

- PUNCH: ($F630)

Rutina: Formatea y graba un archivo o un programa en un cassette.
Efectos: Los contenidos de los acumuladores A y B se alteran durante esta subrutina.
Rutinas usadas: PNCHB
Requerimientos del Stack: 8 Bytes
Descripción: Esta subrutina está sincronizada por software. Antes de iniciar la subrutina almacena la dirección del inicio en la localización(E460, 1) y la dirección final(E462,-3) el formato de salida es como sigue:

a).- 30 segundos de caracteres($FF).
b).- Un block de caracter inicial "S"($53).
c).- Dirección de inicio, byte de mayor peso.
d).- Dirección de inicio, byte de menor peso.
e).- Dirección final, byte de mayor peso.
f).- Dirección final, byte de menor peso.
g).- Flujo binario de datos a partir de la dirección inicial, hasta la dirección final.

h).- Un byte(checksum), el Checksum(chequeo general) es el- complemento a 2 de todos los bytes de datos más los ca racteres de inicio y fin. Al recobrar la información - la suma afirmada de los datos incluyendo los bits de - inicio del Checksum deben de resultar igual a cero.

- LOAD: ($69C)

Rutina: Carga en memoria y además verifica un archivo graba do en cassette.

Efectos: El contenido de los acumuladores A, B y el regis-- tro índice se alteran por esta subrutina.

Rutinas usadas: TIN

Requerimientos del Stack: 6 Bytes

Descripción: Esta rutina se usa para leer un archivo guarda do en cinta TAPE hacia memoria o verificar un archivo de -- cinta de nuevo con los contenidos de memoria. Si el bit me- nos significativos(LSB) de la bandera FNCFL($E43E) es uno - se realiza una carga, de no ser así se realiza una verifica ción.

Normalmente si se detecta un error en el Checksum el dis--- play de 7 segmentos la indicará.

Cuando se ha reiniciado la carga el bit Z del CC si la car- ga fué aceptada, si Z es igual a uno la carga fué buena.

Nombre de Rutina	Addr en D5BUG	Descripción
MNPTR	$E419,A	Pointer to active sub-program.
KEY	$E41B	Entered key code from keypad.
KYFLG	$E41C	Flag to indicate a key is pending.
DISBUF	$E41D - E422	(6) Bytes correspond to (6) 7 -- segment displays. Contains 7 seg ment codes.
ROLPAS	$E423	Flag to indicate first digit en- try.
HEXBUF	$E42C,D,E	3-Byte buffer for hexadecimal in formation. Each byte correspond- to (2) 7 segment display.

Nombre de Rutina	Addr en D5 BUG	Descripción
USP	$E42F,30	User stack pointer pseudo-register.
UCC	$E431	User condition code pseudo-register.
UB	$E432	User B-register pseudo-register.
UA	$E433	User A-register pseudo-register.
UX	$E434,5	User X-register pseudo-register.
UPC	$E436,7	User program counter pseudo-register.
UIRQV	$E43C,D	Points to user's IRQ service routine.
FNCFL	$E43E	Flag to indicate special(or Alternater) function.
FNCPNT	$E43F,40	Points of ADDR of user's special - function table.
BYTE	$E459	Data byte read from cassette or to be punched to cassette.
BEGAD	$E460,1	Beginning address(for punch).
ENDAD	$E462,3	Ending address(for punch).

Tabla 5.1 Resumen de las localidades RAM más usadas por el D5BUG.

5.2.- REGISTROS DEL MPU

Los registros internos del MPU fueron explicados anteriormente en el punto 4.1.1 y son:

- Acumuladores A y B.
- Registro de índice(IR o IX).
- Contador del programa(PC).
- Apuntador de Pila(SP).
- Registro de Código de Condiciones(CC).

5.3.- INSTRUCCIONES SET

Sabemos que una instrucción puede tener 1, 2, 3 o 4 modos de direccionamiento diferentes. Cada modo de direccionamiento es una manera diferente de decirle al MPU donde encontrar los datos.

La instrucción en M6800 incluye 72 instrucciones separadas. Sin embargo, puesto que cada instrucción tiene más de un modo de direccionamiento, hay 197 códigos de máquina válidos. Por ejemplo, la instrucción ADD A tiene un código de operación 8B para el modo inmediato, 9B para el modo directo, BB para el modo extendido, y un AB para el modo indexado. En el modo inmediato los datos están en el próximo byte de memoria mientras que en los otros 3 modos de datos están en alguna otra dirección de memoria.

Todos los modos de direccionamiento para cada instrucción son sumarizados y el código de operación hexadecimal para cada modo de direccionamiento es listado. Algunos ejemplos ilustran el contenido de los registros aplicados antes y después de la ejecución de una instrucción.

Cuando se muestra el signo(#) la instrucción aplicada está en modo "inmediato" y el número seguido del signo(#) es localizado en el próximo byte de memoria. Cuando el signo($) se muestra indica que el próximo número está en hexadecimal. Sino se presenta el signo(#), entonces el número seguido al($) está en una dirección hexadecimal en memoria. El modo de direccionamiento indexado se muestra con un número hex, seguido por una coma y una X, por ejemplo, $10,X. El número seguido del signo($) es sumado al contenido del IR para formar una nueva dirección efectiva. Por ejemplo:

1.- LDA A #$25 Indica que el dato de 25 está en hex(modo inmediato).

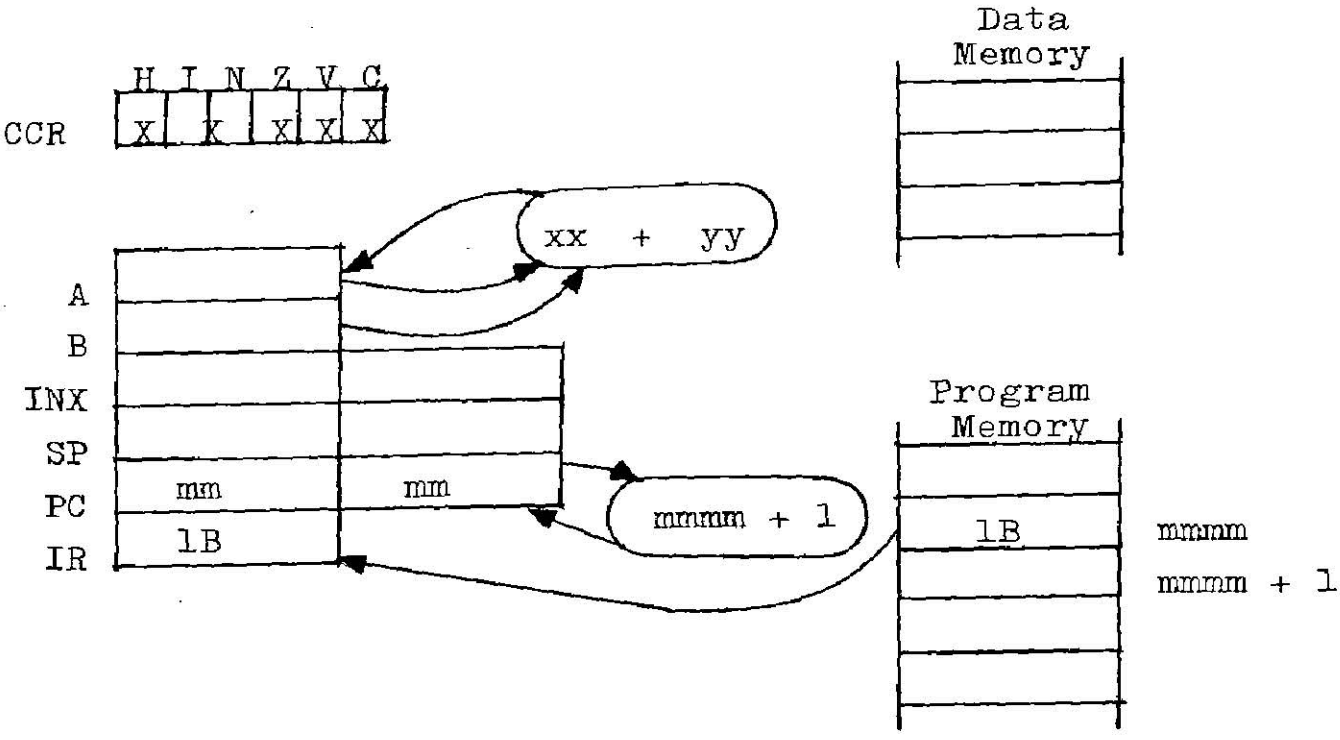
2.- LDA A $25 Indica que en el 25 hex está una dirección--(modo directo).

3.- LDA A $2525 Indica que en el 2525 hex está una dirección (modo extendido).

4.- LDA A $25,X Indica que la dirección está formada por el- 25 hex a el contenido del IR para formar una nueva dirección(modo indexado).

Esta sección contiene el conjunto de instrucciones completas para el microprocesador M6800.

ABA - Suma el Acumulador B al Acumulador A



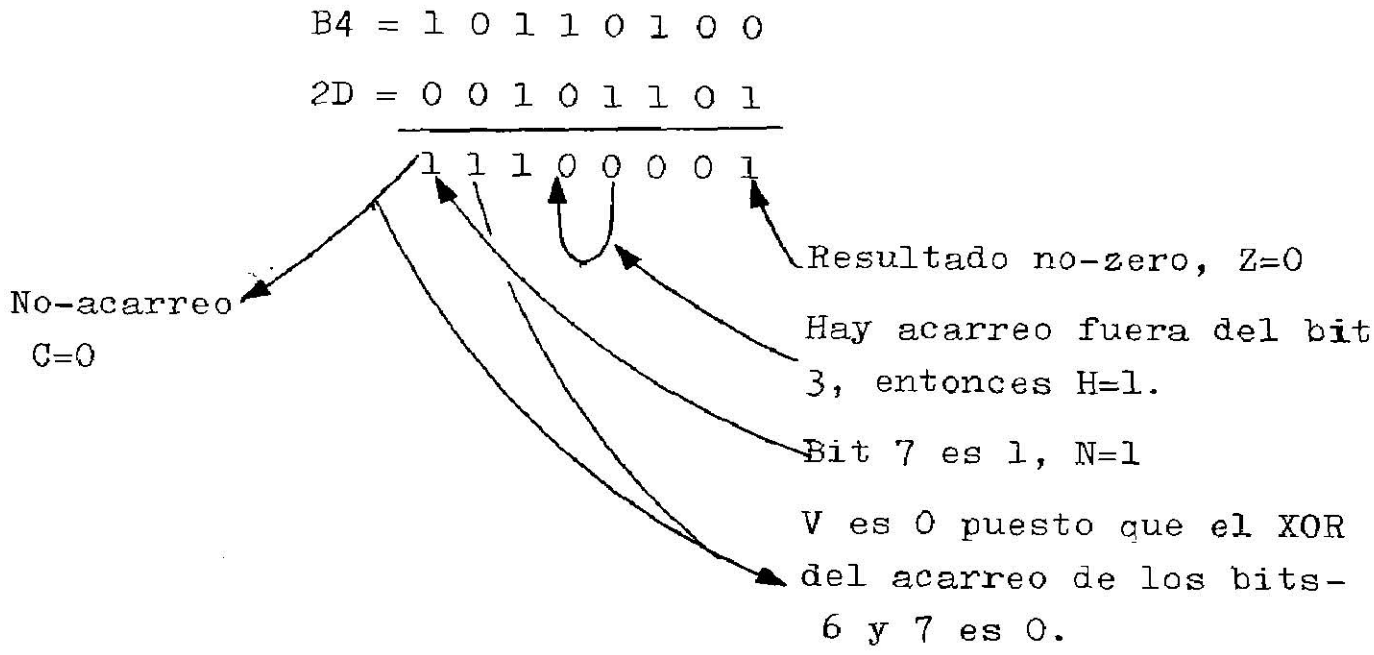
ABA - 1B

Suma el contenido del acumulador B a el contenido del acumulador A, y almacena, el resultado, en el acumulador A.

Si xx=$B4_{16}$ y yy=$2D_{16}$, entonces después de ejecutar la instrucción:

ABA

el Acumulador A contiene $E1_{16}$.

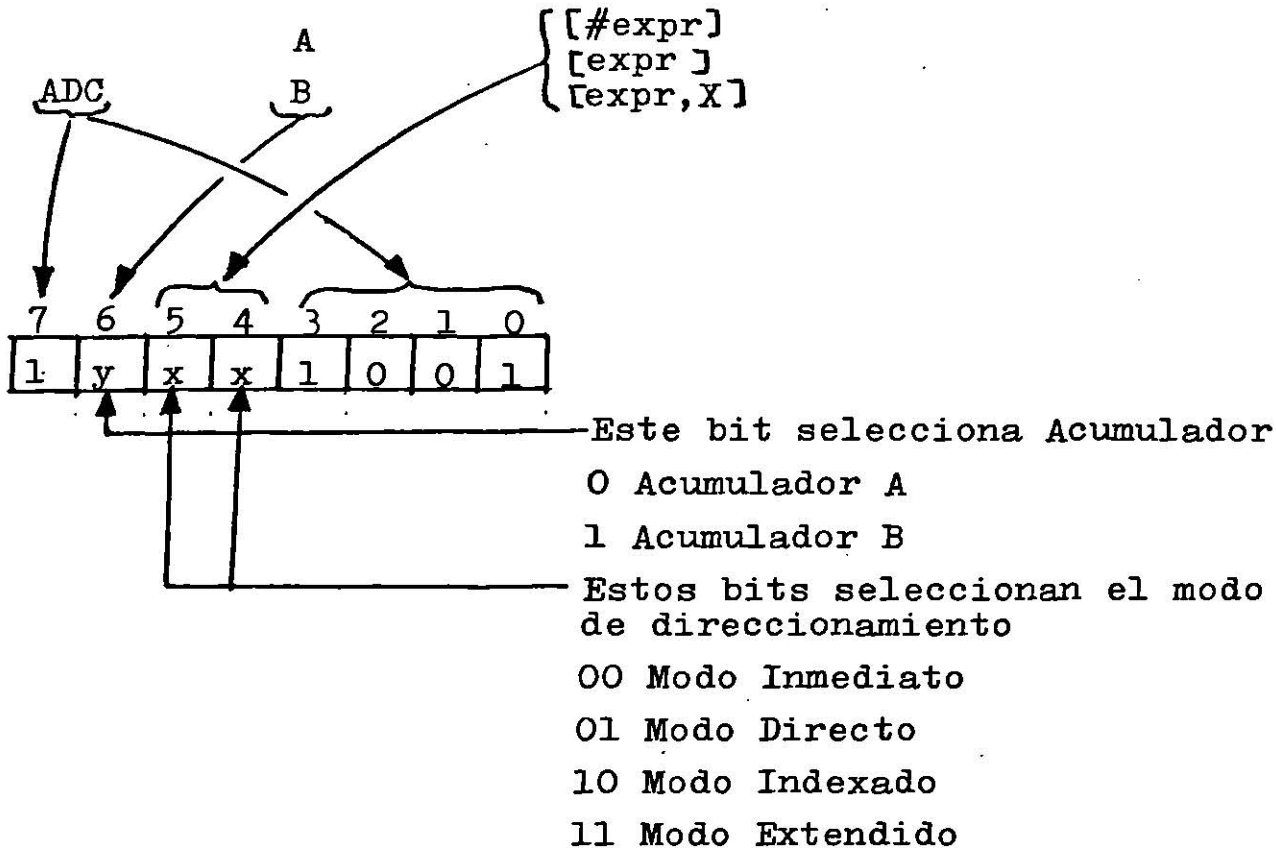


ADC - Suma Memoria , con Acarreo, al Acumulador A o B

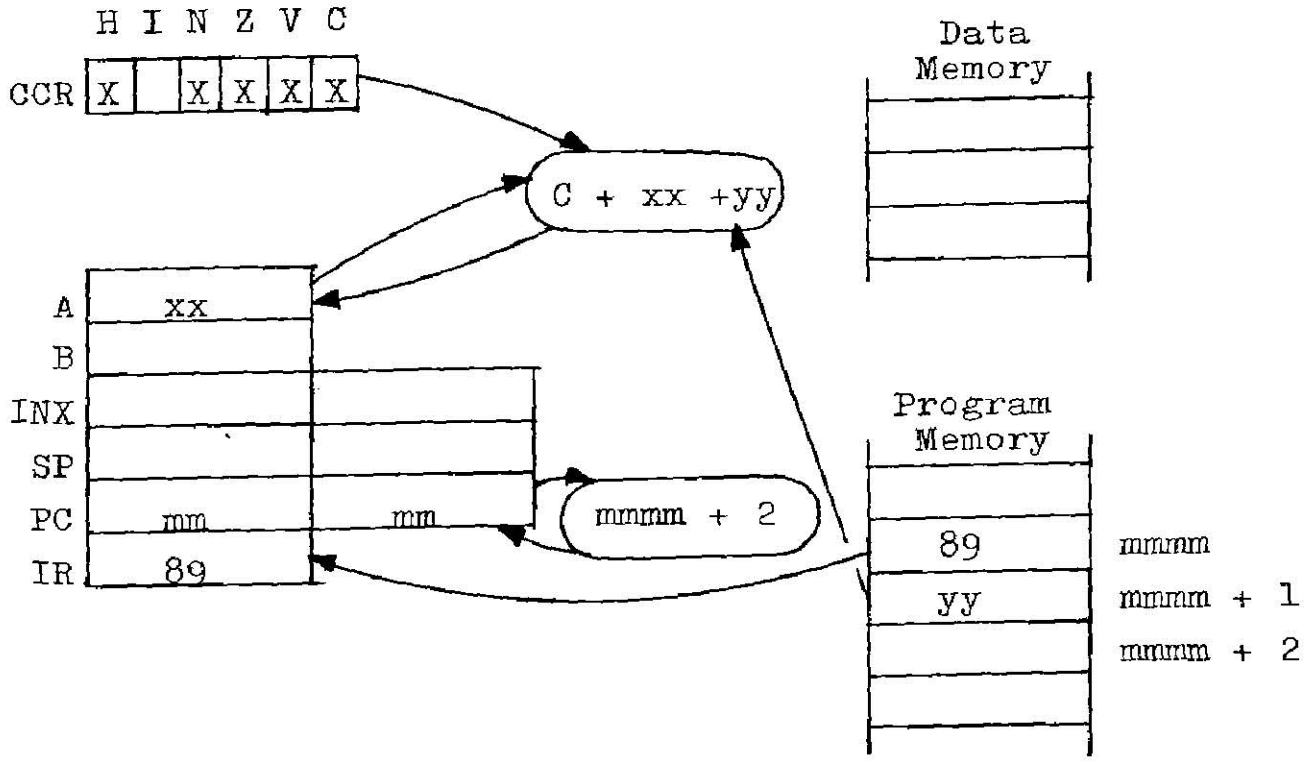
Esta instrucción usa 4 modos de direccionamiento de datos - de memoria y permite que los contenidos de los datos de memoria y el estado de acarreo sean sumados al acumulador A o el- B. Los 4 métodos son:

- Inmediato
- Extendido
- Directo
- Indexado

El primer byte del código objeto determina cual opción -- de direccionamiento es seleccionada.



Primero, consideremos la ejecución de la adición con acarreo- usando datos inmediatos.



Al acumulador A,(seleccionado por el bit 6 del byte en el registro de instrucción), suma el contenido del byte del programa,(modo de direccionamiento seleccionado por los bits 5 y 4- del byte en el registro de instrucción) y el estado Carry. Suponer $xx=3A_{16}$, $yy=7C_{16}$, C=1. Después que la instrucción:

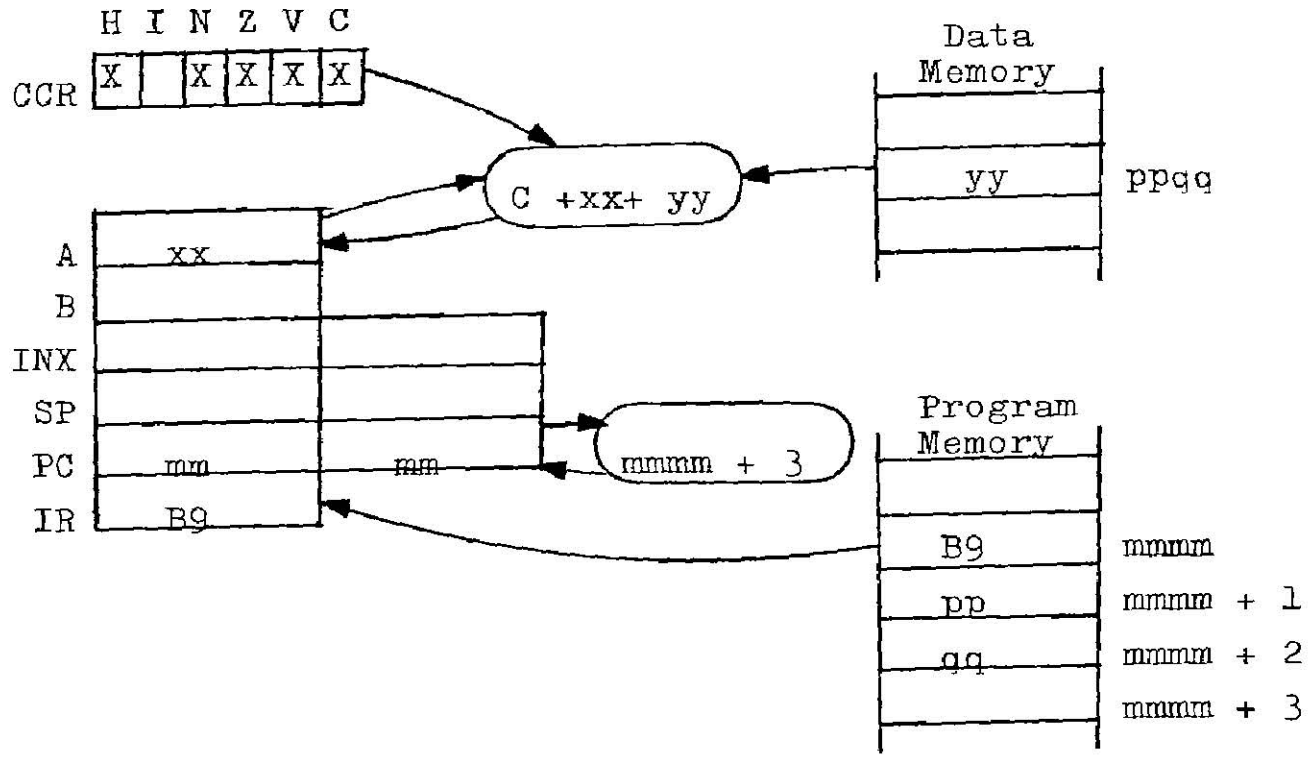
ADC A #$7C

es ejecutada, el acumulador contiene $B7_{16}$

3A = 0 0 1 1 1 0 1 0
7C = 0 1 1 1 1 1 0 0
Carry = 1
1 0 1 1 0 1 1 1 ← Resultado no-cero, Z=0

C=0
N=1
0 ⊻ 1 = 1, V=1

La suma con acarreo usando el modo Extendido trabaja de manera similar al modo Directo.

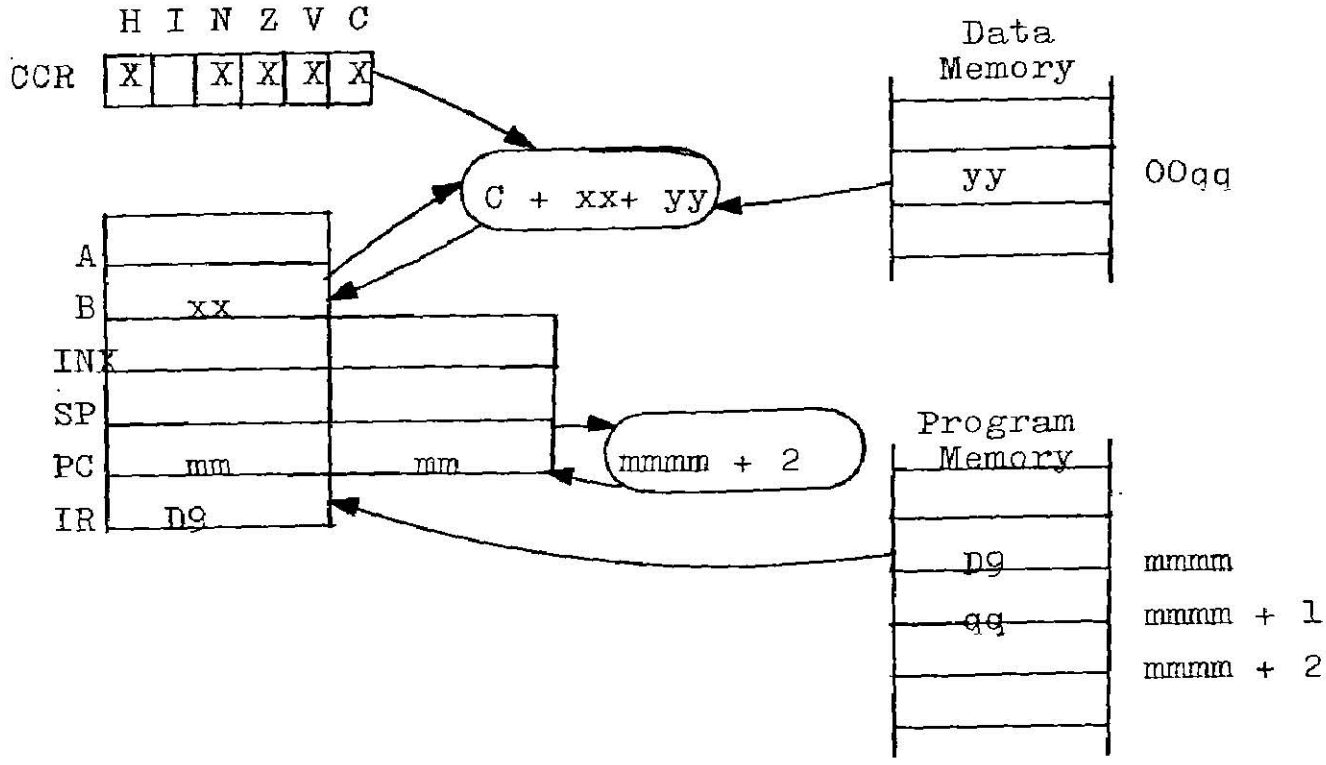


Al acumulador A, le suma el estado Carry y el contenido de la memoria de datos direccionada por los próximos dos bytes del programa de memoria,(el byte de alto orden está en la dirección del segundo byte del código objeto; mmmm + 1, y el byte de bajo orden está en la dirección del tercer byte del código objeto, mmmm + 2). Note que los 2 bytes del programa pueden direccionar los bytes de la memoria de datos en el rango de 0 ppqq $65,355_{10}$. Si xx=$3A_{16}$, pp=50_{16}, qq=23_{16}, yy=$7C_{16}$ y C=1, la ejecución de la instrucción:

ADC A $5023

produce el mismo resultado que la ejecución de la instrucción ADC A #$7C.

Consideremos la suma uando el modo Directo:

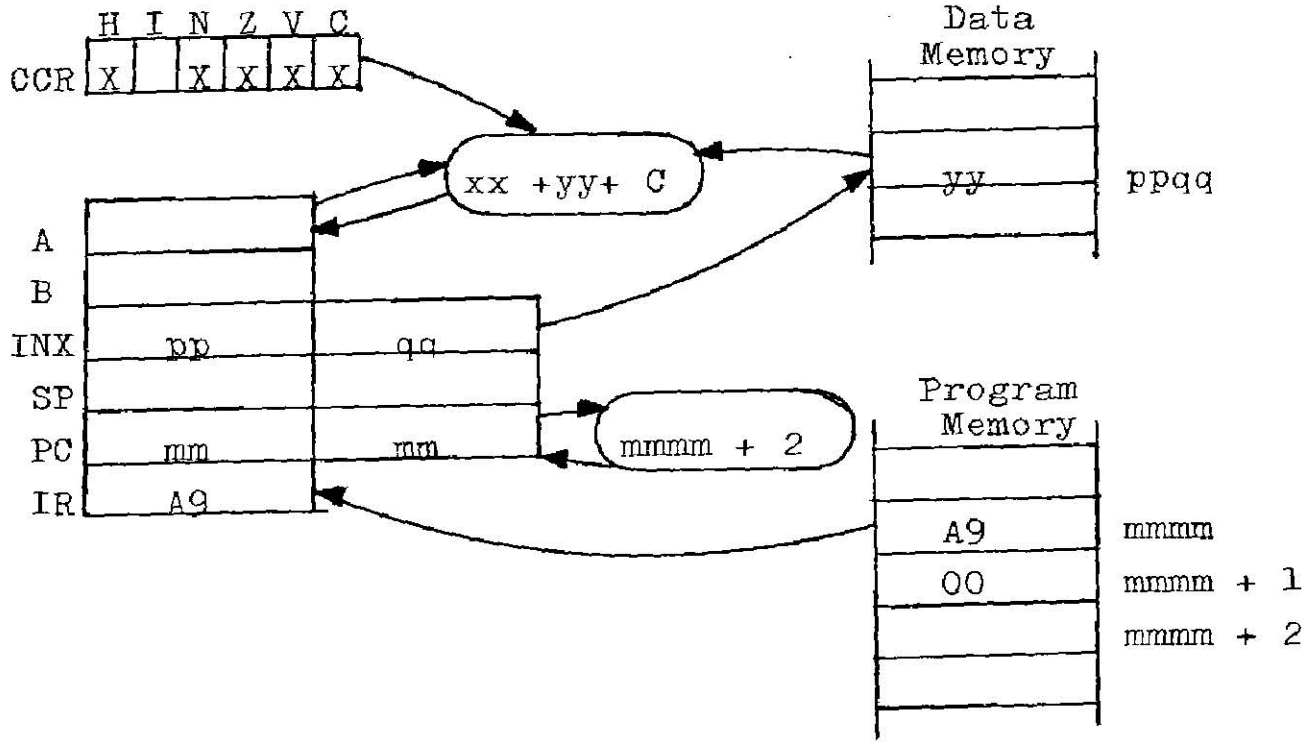


Al acumulador B le suma el estado Carry y el contenido de la-memoria de datos direccionada por el próximo byte de memoria-de datos(mmmm + 1). Note que el próximo byte del programa de-memoria puede direccionar el byte de memoria de datos en el - rango 0_{10} qq 255_{10}. Si xx=$3A_{16}$, qq=$1F_{16}$, yy=$7C_{16}$ y C=1, - entonces al ejecutar la instrucción:

ADC B $1F

genera el mismo resultado que la ejecución de la instrucción-ADC A #$7C, con la excepción que el resultado es almacenado - en el acumulador B en lugar del acumulador A.

El modo Indexado toma 2 formas diferentes:con desplazamiento- y con no-desplazamiento. El modo Indexado con no-desplazamien to.

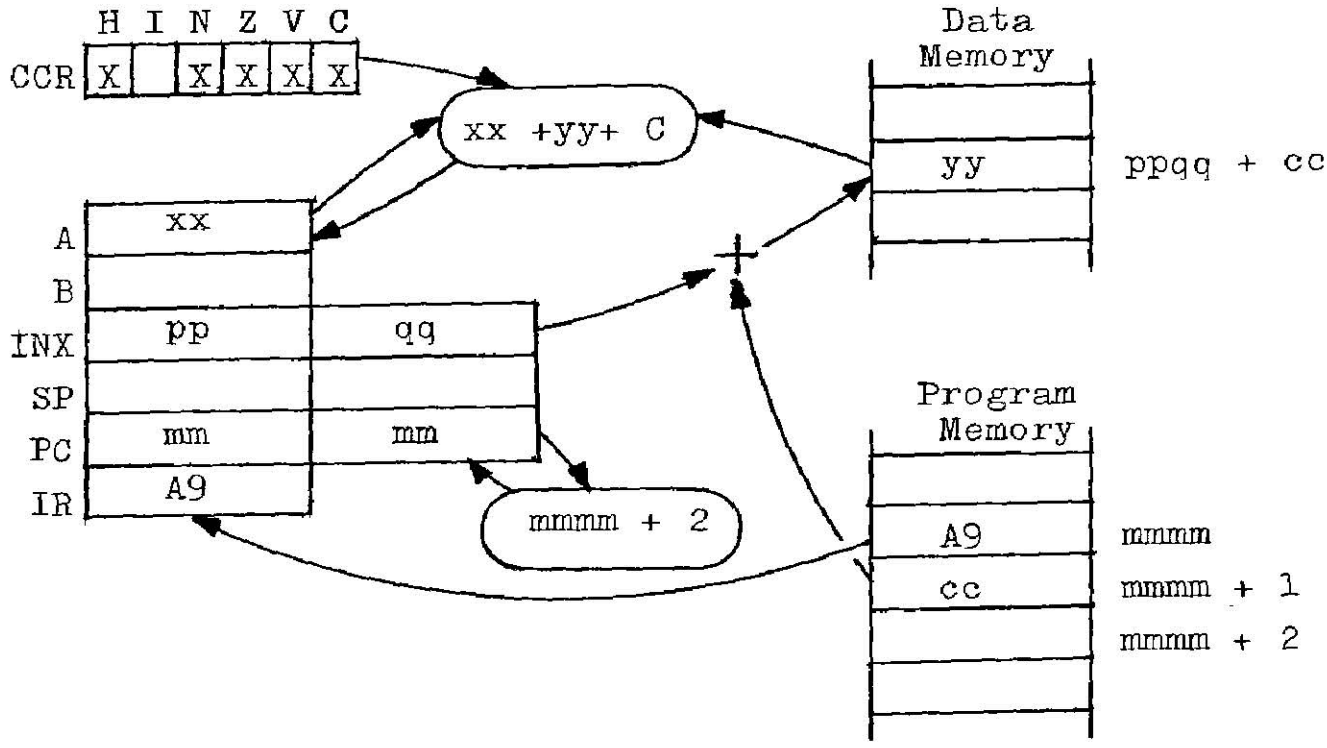


Al acumulador A, le suma el estado Carry y el contenido de la memoria de datos direccionada por el registro de índice. Note que el registro de índice puede direccionar los bytes de la - memoria de datos en el rango de 0 ppqq $65,355_{10}$. Si $xx=3A_{16}$ ppqq=(el contenido del INX)=5023_{16}, $yy=76_{16}$ y C=1, entonces - la ejecución de la instrucción:

ADC A X

produce el mismo resultado de la instrucción ADC A $5023.

El modo Indexado con desplazamiento permite un desplazamiento en el byte seguido a la instrucción a ser sumada al contenido del registro de índice.



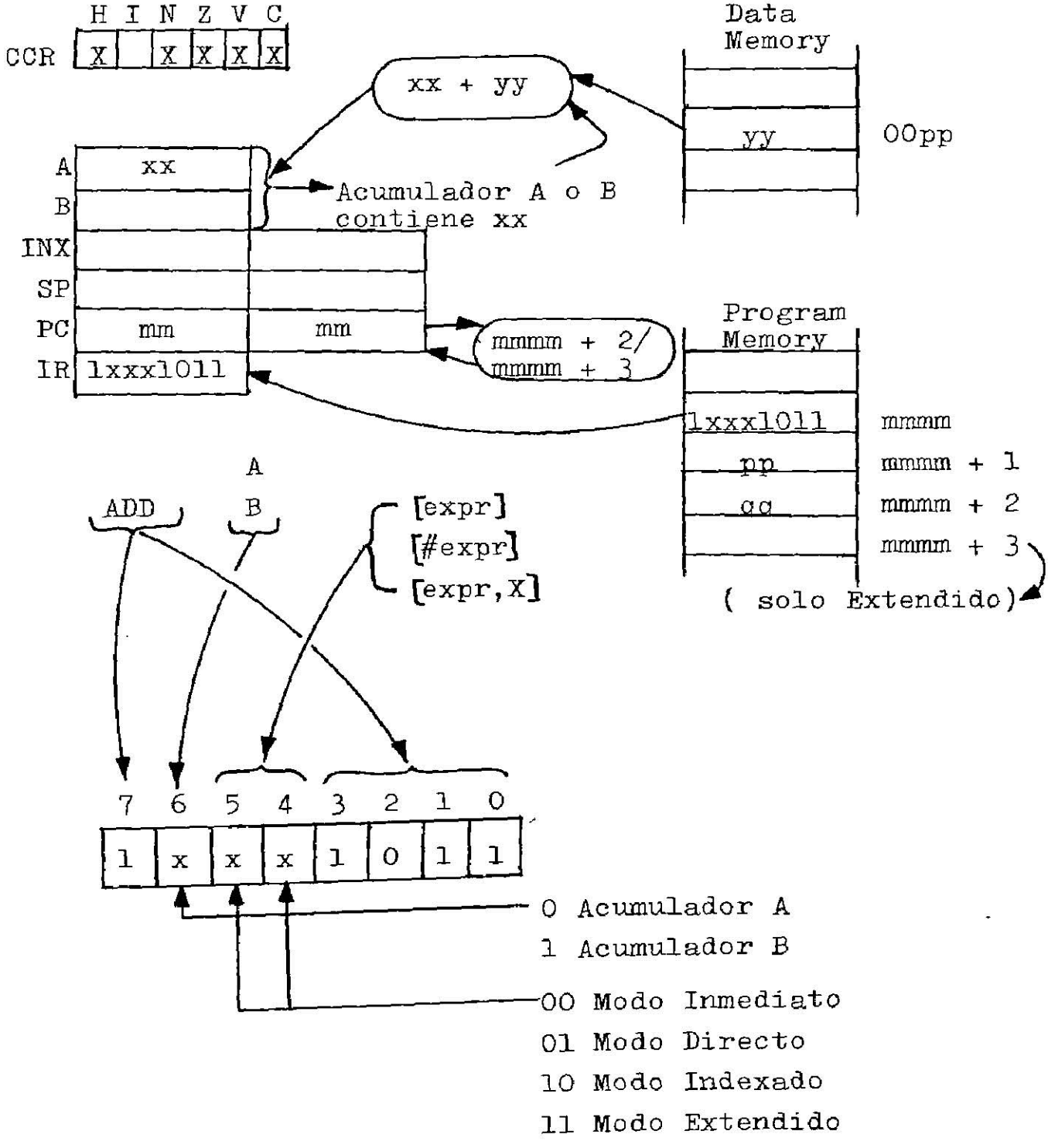
Al acumulador A, le suma el contenido de memoria direccionada por la suma del registro de índice y el byte del programa de memoria seguido del código de instrucción, (Note que en el ejemplo anterior mmmm + 1 fué 0), y el estado Carry. El valor en mmmm + 1 es tratado como un entero de 8 bits cuando la suma con el registro de índice es ejecutado. Si $xx=3A_{16}$, $ppqq=500D_{16}$, $cc=16_{16}$, $yy=76_{16}$ y C=1, la instrucción:

```
ADC A $16,X
```

genera el mismo resultado que la instrucción ADC A $5023.

ADD - Suma Memoria al Acumulador

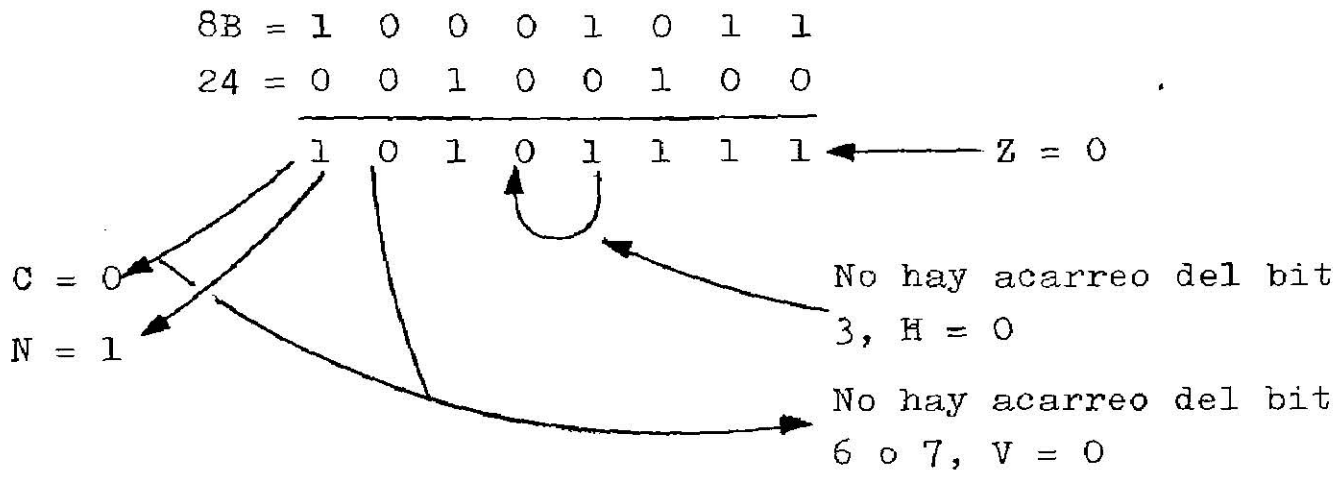
Esta instrucción suma el contenido de una localidad de memoria al Acumulador A o al Acumulador B. Esta instrucción ofrece la misma opción de direccionamiento de memoria que la instrucción ADC, y puede ser ilustrada usando el modo Directo.



Al ser seleccionado el acumulador, suma el contenido del byte de memoria seleccionado. Suponer $xx=24_{16}$, $yy=8B_{16}$, $pp=43_{16}$ y C=1. Después que la instrucción:

ADD A $43

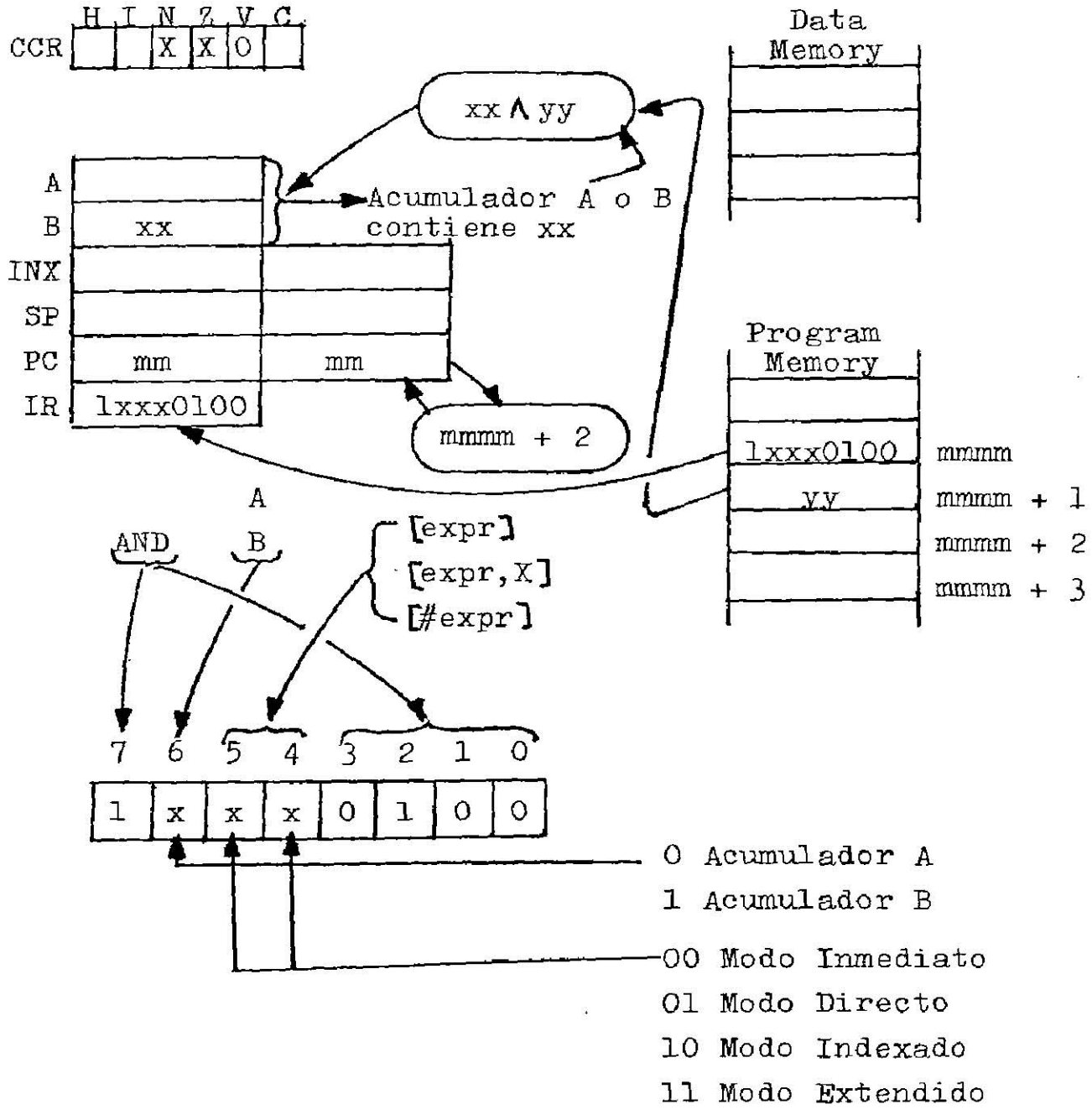
es ejecutada, el acumulador A contiene AF_{16}.



ADD es la instrucción de adición binaria usada normalmente, en operaciones de un solo byte.

AND - Aplicación de AND de Memoria con el Acumulador

Esta instrucción ANDs el contenido de una localidad de memoria con el contenido del acumulador A o B. Esta instrucción ofrece la misma opción de direccionamiento de memoria que la instrucción ADC y puede ilustrarse usando el modo inmediato.



Aplica el AND al contenido del byte de memoria seleccionado--con el acumulador seleccionado y almacenando el resultado en-el acumulador seleccionado. Suponer xx=FC_{16}, yy=13_{16}.Después-que la instrucción:

```
AND B #$13
```

es ejecutada, el acumulador B contiene 10_{16}.

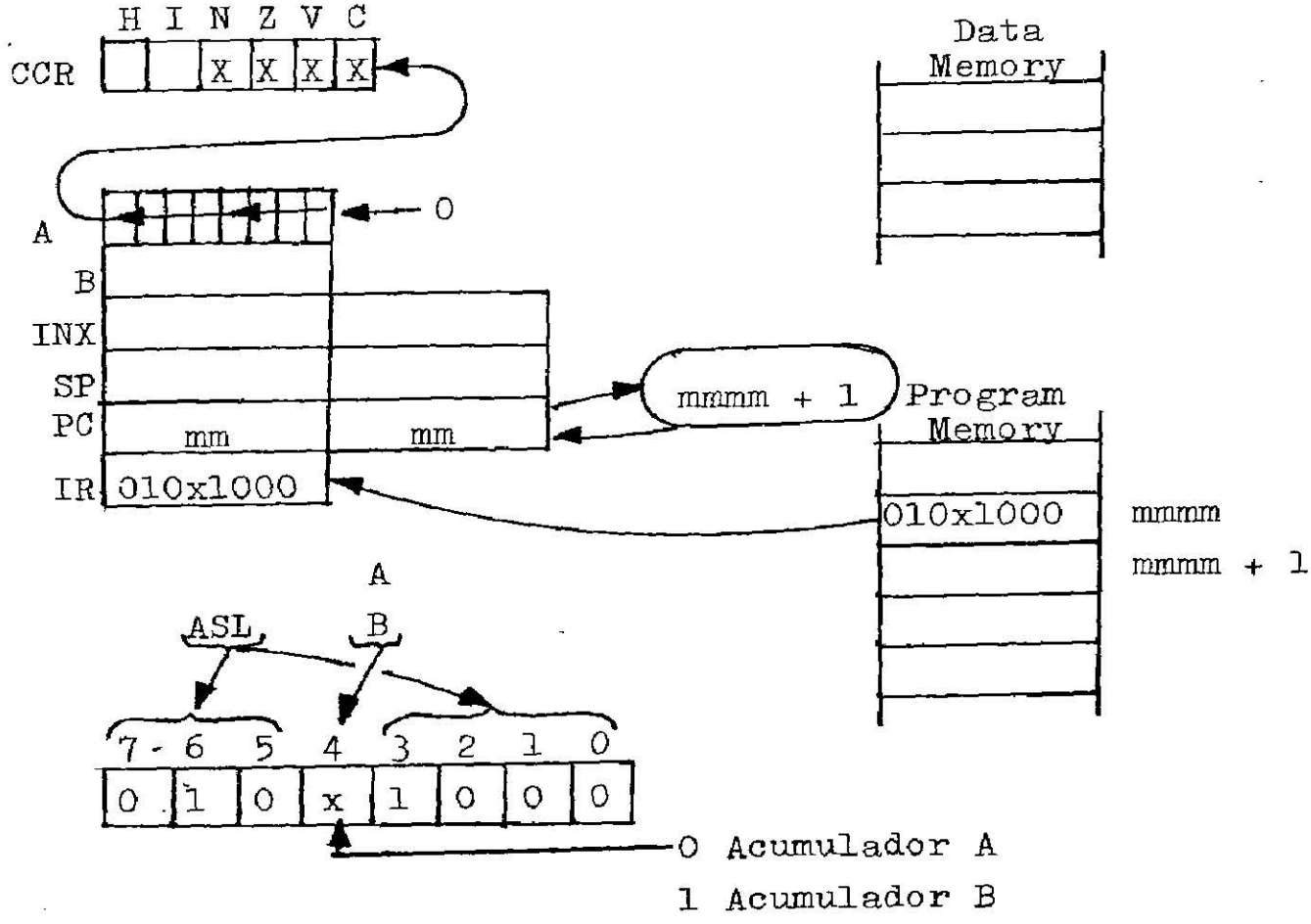
```
FC = 1 1 1 1 1 1 0 0
13 = 0 0 0 1 0 0 1 1
     -----------------
     0 0 0 1 0 0 0 0 <-- Coloca Z=0

0 en el bit 7                V es limpiado a 0
coloca N = 0
```

AND es una instrucción lógica frecuentemente usada.

ASL - Cambia el Acumulador o Memoria un Byte a la Izquierda

Ejecuta un bit aritmético a la izquierda cambiando el contenido del acumulador A o el B, o el contenido de un byte de-memoria seleccionado.

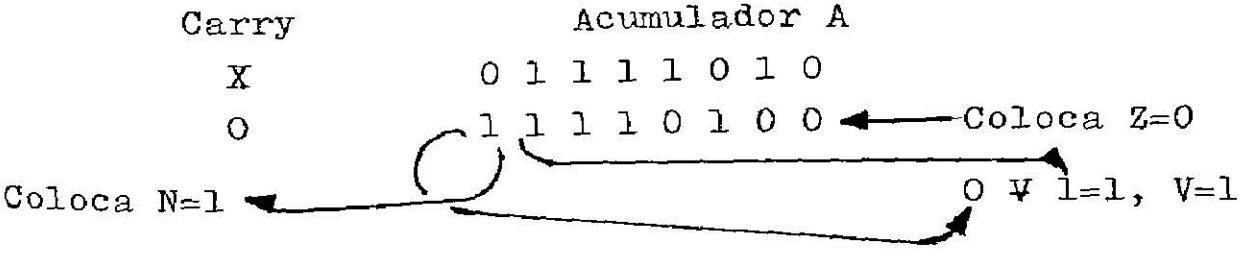
Primero, consideremos el cambio en el Acumulador.



Suponer que el acumulador B contiene $7A_{16}$. Ejecutando una:

ASL A

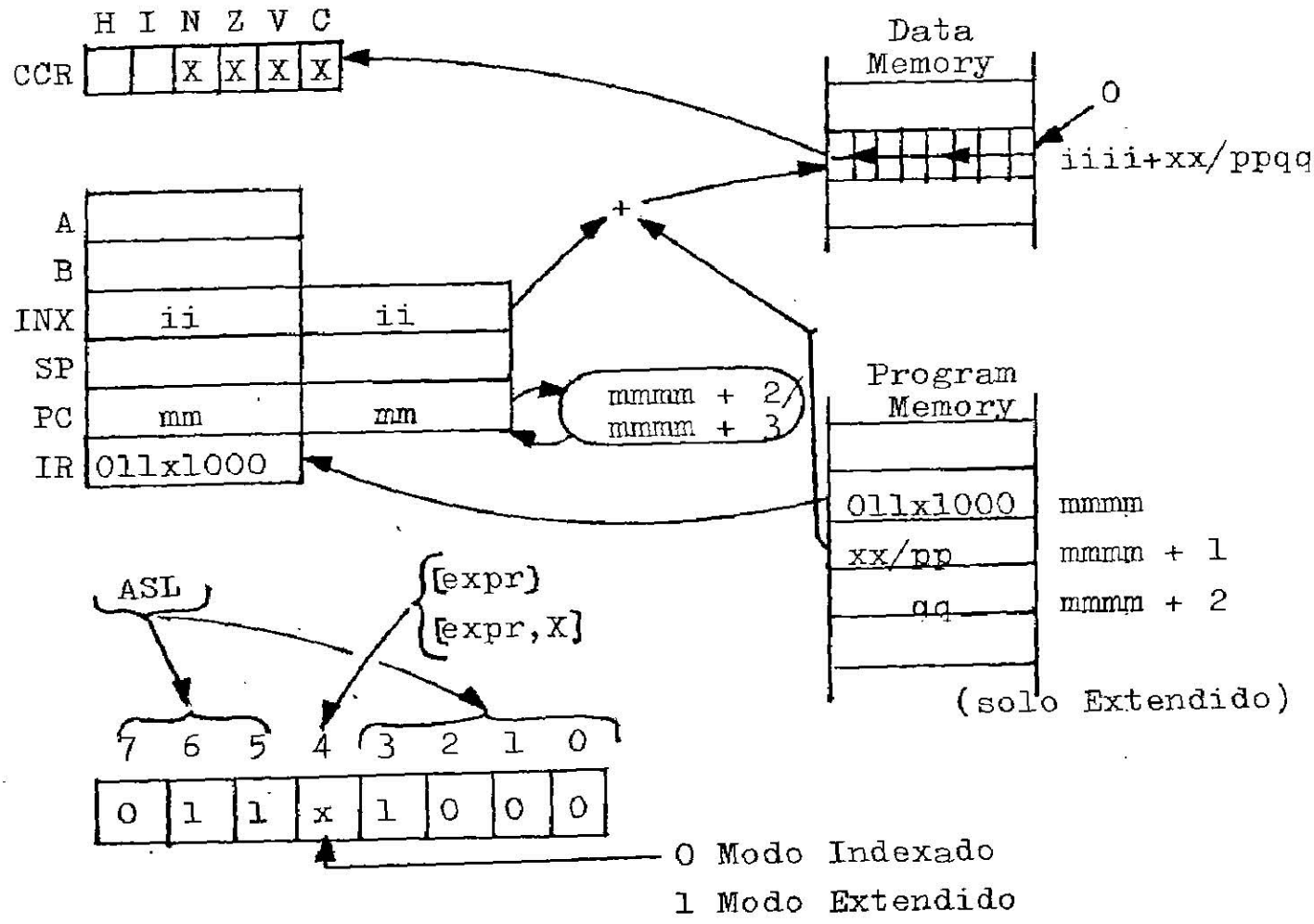
puede colocar el estado Carry en 0, el estado Sign en 1, el - estado Overflow en 1, el estado Zero en 0 y almacena $F4_{16}$. - en el acumulador A.



La instrucción ASL usa 2 opciones de modo de direccionamiento de memoria de datos:

-Extendido

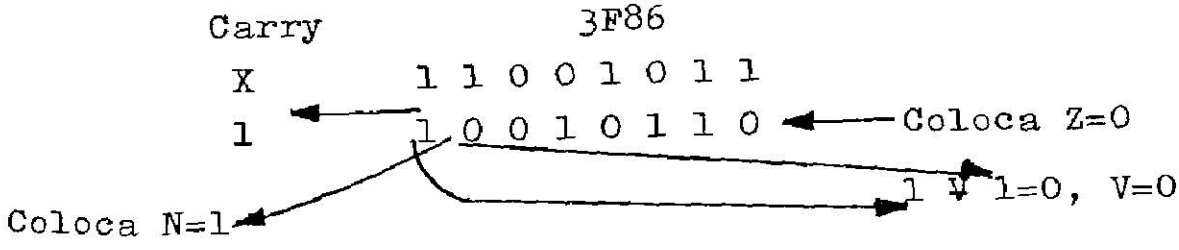
-Indexado



Suponer que el modo indexado es usado, iii=$3F3C_{16}$, xx=$4A_{16}$, ppqq=$3F86_{16}$ y el contenido de ppqq es CB_{16}. Después que se ejecuta la instrucción:

```
ASL   $4A,X
```

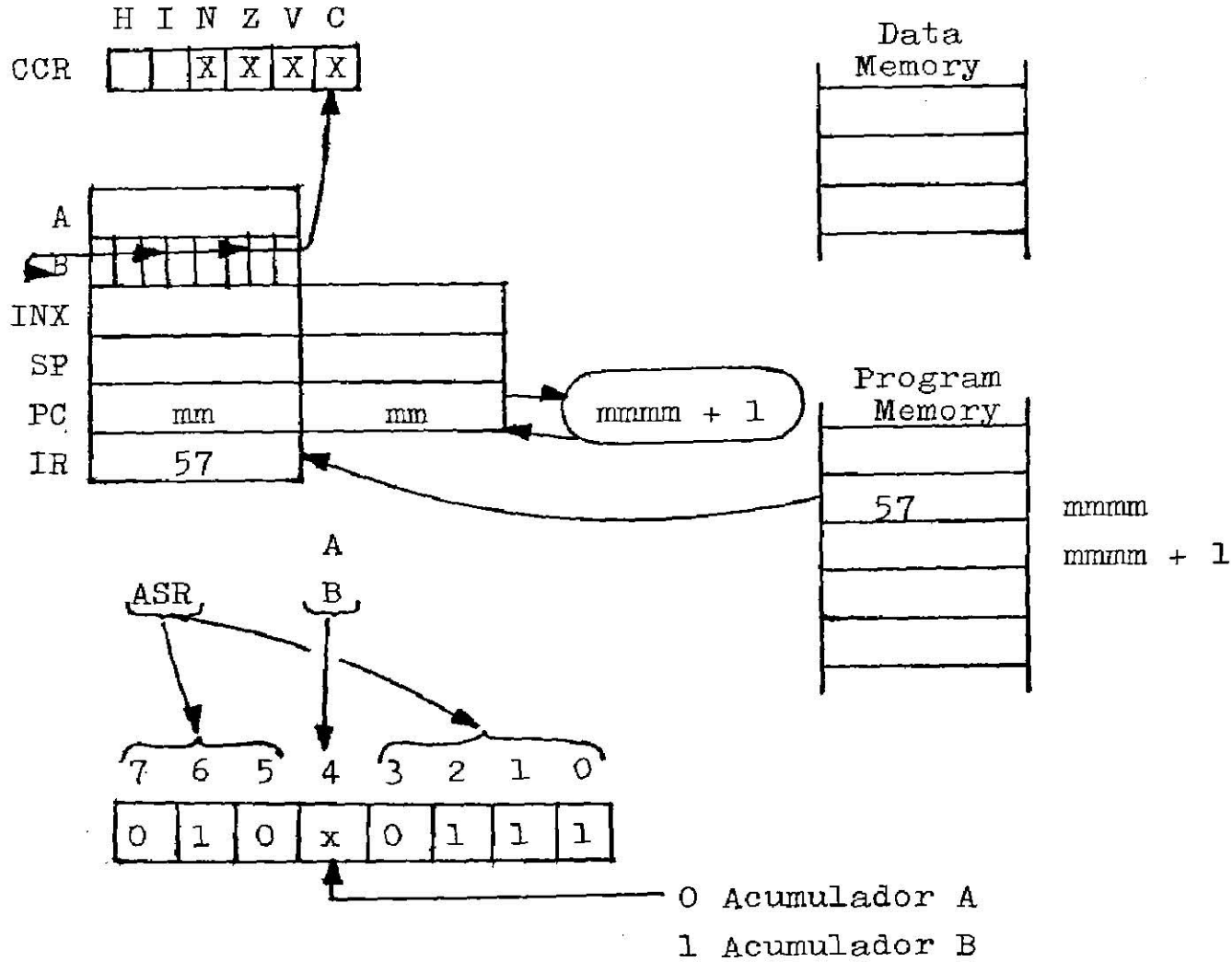
el contenido de ppqq es alterado a 96_{16} y el Carry es 1.



ASR - Cambia el Acumulador o Memoria un Byte a la Derecha

Ejecuta un bit aritmético a la derecha cambiando el contenido del acumulador A o el B, o el contenido de un byte de memoria seleccionado.

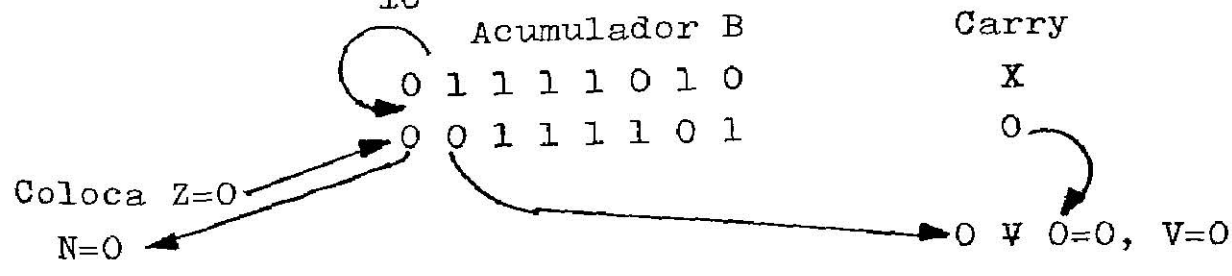
Primero, consideremos el cambio en el Acumulador.



Suponer que el acumulador B contiene $7A_{16}$. Ejecutando una instrucción:

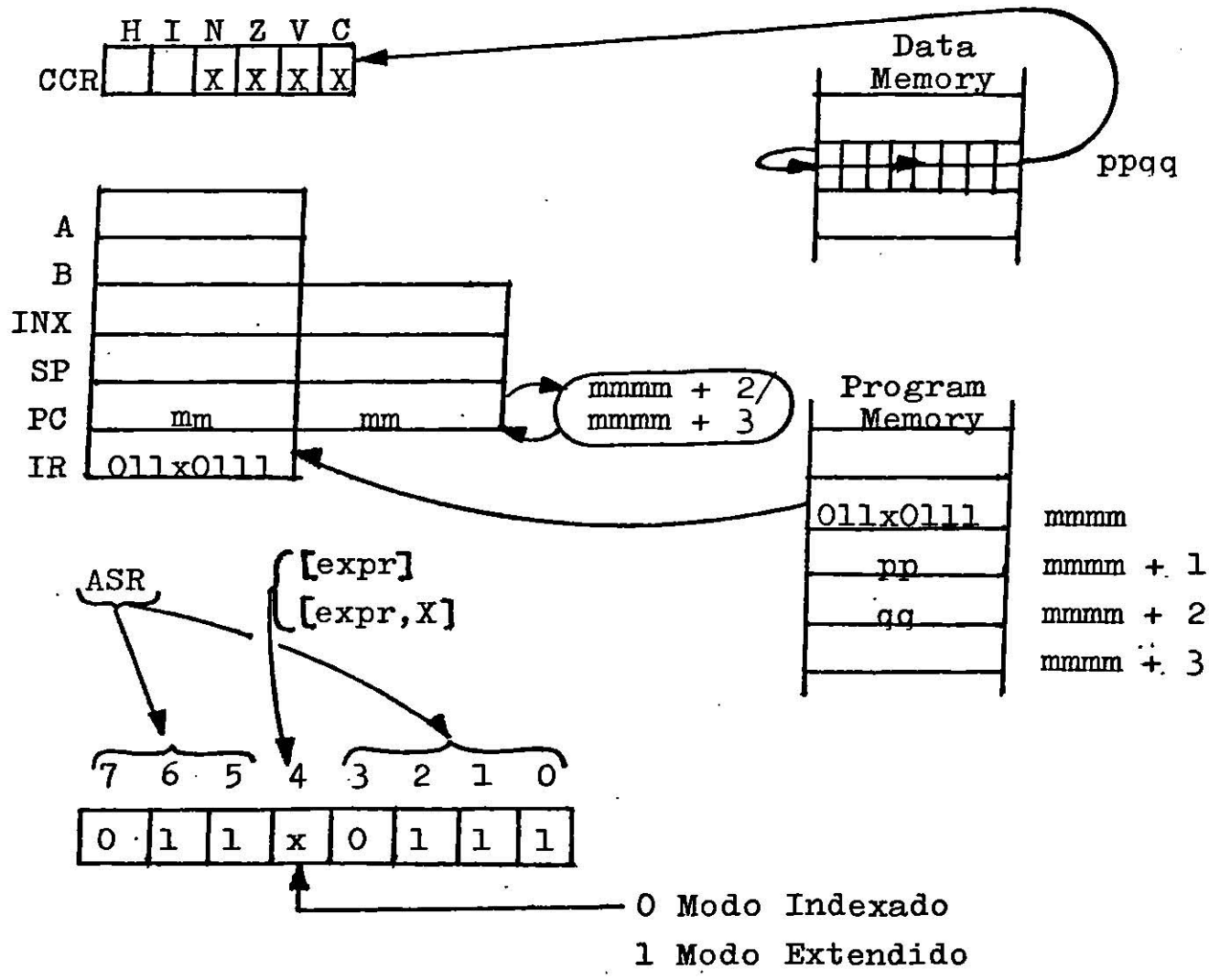
ASR B

coloca el Carry en 0, el Sign en 0, el Overflow en 0, el Zero en 0 y almacena $3D_{16}$ en el Acumulador B.



La instrucción ASR tiene 2 opciones de modo de direccionamiento de memoria de datos:

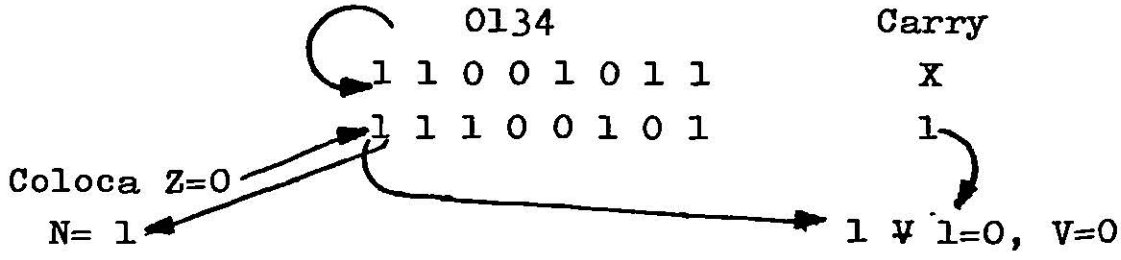
- Extendido
- Indexado



Suponer que usamos el modo extendido, pp=01_{16}, qq=34_{16} y el contenido de 0134_{16} es CB_{16}. Ejecutando la instrucción:

```
ASR   $0134
```

altera el contenido de la localidad de memoria 0134_{16} a $E5_{16}$.

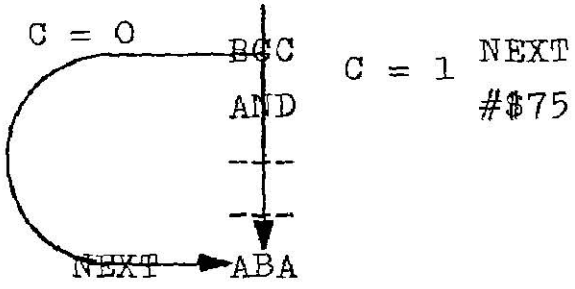


BCC - Salto si el Carry está Limpio

BCC - 24

Esta instrucción es idéntica a la instrucción BRA excepto que el salto(branch) es solo ejecutado si el estado Carry es-igual a cero, de otro modo la próxima instrucción es ejecutada.

En la siguiente secuencia de instrucción:



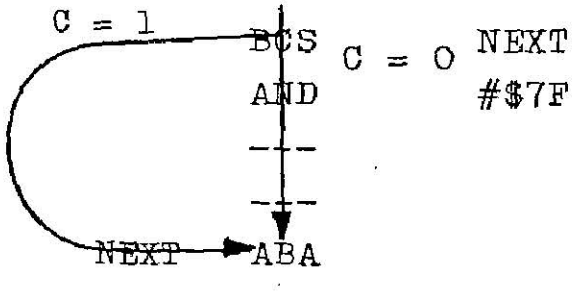
Después de la instrucción BCC, la instrucción ABA es ejecutada si el estado Carry es igual a cero. La instrucción AND es-ejecutada si el estado Carry es igual a uno.

BCS - Salto si el Carry es Colocado

BCS - 25

Esta instrucción es idéntica a la instrucción BRA excepto que el salto es ejecutado solo si el Carry es igual a uno, de otro modo la próxima instrucción es ejecutada.

En la siguiente secuencia de instrucción:

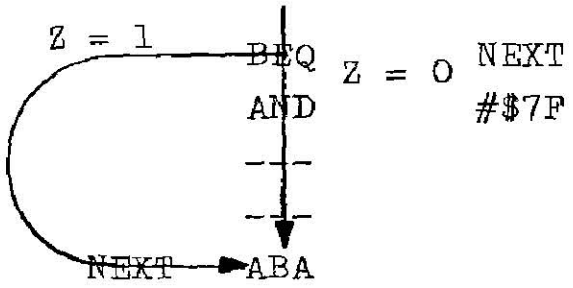


Después de la instrucción BCS, la instrucción ABA es ejecutada si el estado Carry es igual a uno. La instrucción AND es - ejecutada si el estado Carry es igual a cero.

BEQ - Salto si es Igual

BEQ - 27

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si el estado Zero es igual a uno; de otro modo es ejecutada la próxima instrucción.
En la siguiente secuencia de instrucción:

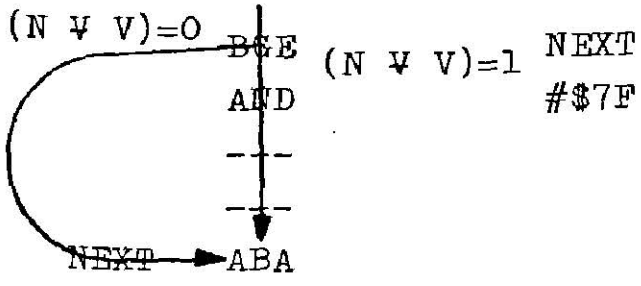


Después de la instrucción BEQ, la instrucción ABA es ejecutada si el estado Zero es igual a uno. La instrucción AND es ejecutada si el estado Zero es igual a cero.

BGE - Salto si es Mayor que o Igual a Zero

BGE - 2C

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado si el OR-exclusivo de los estados Sign(N) y Overflow(V) son cero; esto es, Sign y Overflow son ambos 1 o son ambos 0; de otro modo se ejecuta la próxima instrucción.
En la siguiente secuencia de instrucción:



Después de la instrucción BGE, la instrucción ABA es ejecutada si los estados Sign y Overflow son ambos 1 o 0. La instrucción AND es ejecutada si el estado Sign no es igual al Overflow.

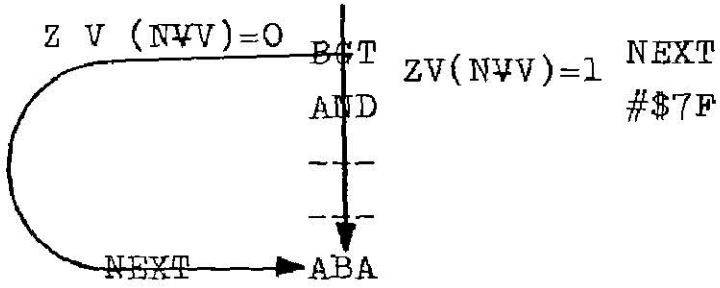
Esta instrucción es usada para ejecutar complementos a 2 mayores o iguales al salto(branch).

BGT - Salto si es Mayor que Zero

BGT - 2E

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si se cumple uno de los siguientes puntos:
- El estado Zero es 0 y el Sign y Overflow son 0.
- El estado Zero es 0 y el Sign y Overflow son 1.

De otro modo, la próxima instrucción es ejecutada.
En la siguiente secuencia de instrucción:

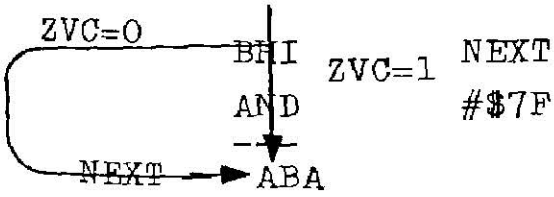


Después de la instrucción BGT, la instrucción ABA es ejecutada si la bandera Zero es 0 y el OR-exclusivo de los estados - Sign y Overflow son 0. De otro modo, la instrucción AND es -- ejecutada.
Esta instrucción es usada para implementar complementos a 2 - mayor que la capacidad del salto.

BHI - Salto si es muy Grande

BHI - 22

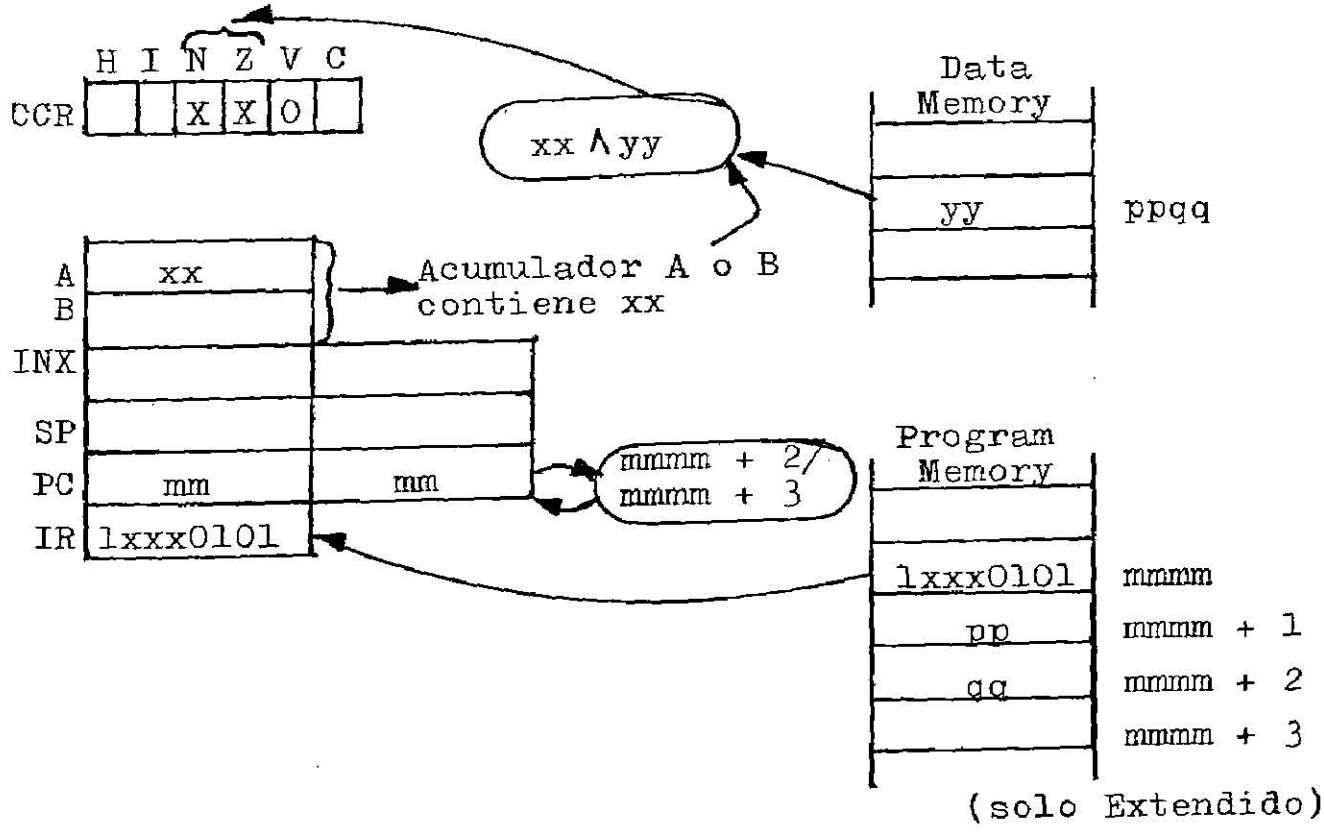
Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si el estado Zero y el Carry son ceros; de otro modo, ejecuta la próxima instrucción.
En la siguiente secuencia de intrucción:

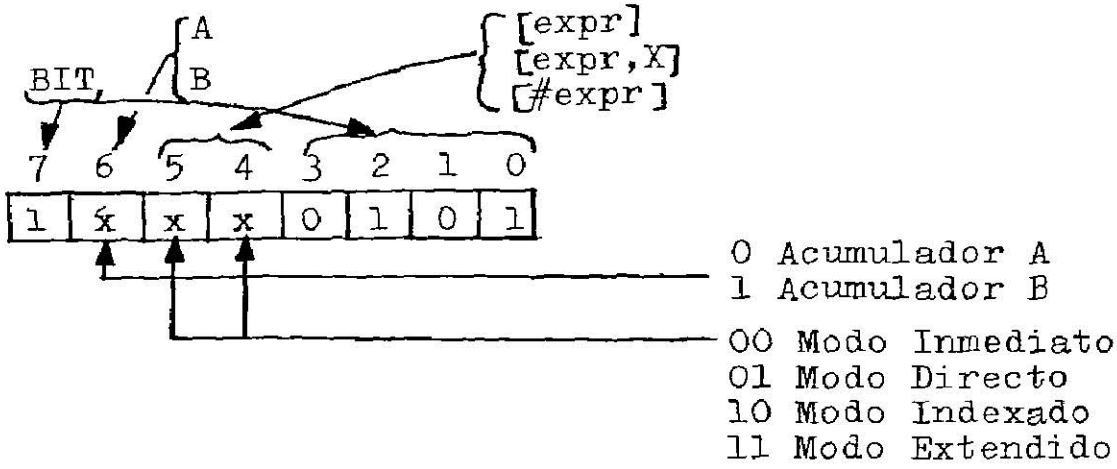


Después de la instrucción BHI, la instrucción ABA es ejecutada si los estados Zero y Carry son 0. La instrucción AND es ejecutada si cualquiera de los dos estados Zero y Carry son uno o si ambos son uno.

BIT - Bit de Prueba

Esta instrucción ANDs el contenido del acumulador A o B con el contenido de una localidad de memoria seleccionada, colocando las banderas de condición por consiguiente, pero sin alterar el contenido del acumulador o el byte de memoria. Esta instrucción ofrece la misma opción de direccionamiento de memoria que la instrucción ADC. Esta instrucción puede ser ilustrada usando el modo Extendido.





Aplica el AND al contenido del acumulador especificado con el contenido de una localidad de memoria y coloca las banderas Sign y Zero por consiguiente. Suponer xx=$A6_{16}$, yy=$E0_{16}$ y ppqq=1641_{16}. Después que la instrucción:

BIT A $1641

es ejecutada, el acumulador A contiene $A6_{16}$, la localidad ppqq contiene $E0_{16}$ pero los estados pueden modificarse:

A6 = 1 0 1 0 0 1 1 0
E0 = 1 1 1 0 0 0 0 0
1 0 1 0 0 0 0 0 ← Coloca Z=0

Coloca N=1 ←

V es limpiado a 0

La instrucción BIT generalmente precede a instrucciones condicionales Branch.

BLE - Salto si es Menor o Igual a Zero

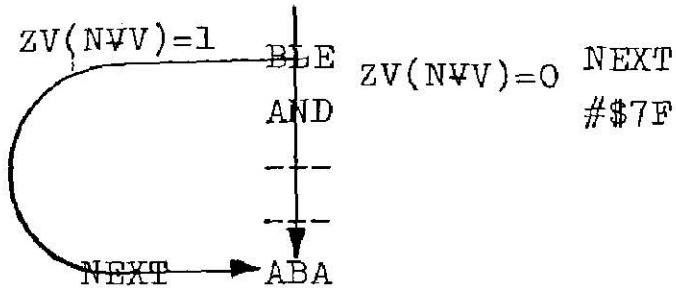
BLE - 2F

Esta instrucción es idéntica a la instrucción BRA con la excepción que el salto es ejecutado solo si una o más de las 3 condiciones siguientes se cumplen:

- El estado Zero es 1.
- El estado Overflow es 1 y el estado Sign es 0.
- El estado Overflow es 0 y el estado Sign es 1.

De otro modo, la próxima instrucción es ejecutada.

En la siguiente secuencia de instrucción:



Después de la instrucción BLE, la instrucción ABA es ejecutada si el estado Zero es 1 o si el OR-exclusivo de los estados Sign y Overflow es 1. La instrucción AND es ejecutada si el estado Zero es 0 y el OR-exclusivo de los estados Sign y Overflow es 0.

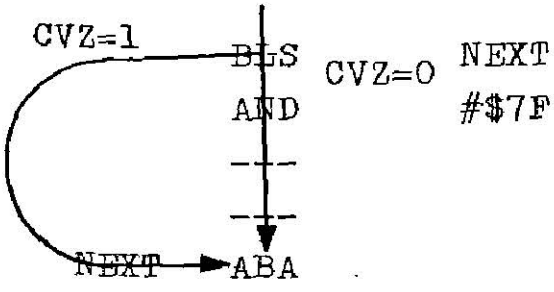
Esta instrucción provee al programador de complementos a 2 -- menores o iguales al branch.

BLS - Salto si es Menor o Igual

BLS - 23

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si cualquiera de los estados Carry o Zero son 1; de otro modo, la próxima instrucción es ejecutada.

En la siguiente secuencia de instrucción:



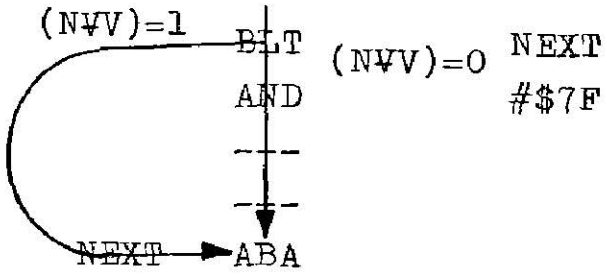
Después de la instrucción BLS, la instrucción ABA es ejecutada si el estado Carry o el Zero es igual a 1. La instrucción AND es ejecutada si ambos estados son 0.

Esta instrucción es útil como un señalamiento menor o igual al branch.

BLT - Salto si es Menor que Zero

BLT - 2D

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo cuando el OR-exclusivo de los estados Sign y Overflow es 1; de otro modo, la próxima instrucción es ejecutada.
En la siguiente secuencia de instrucción:

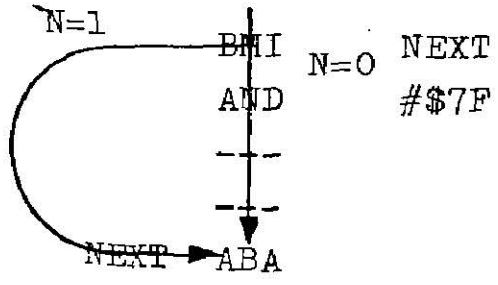


Después de la instrucción BLT, la instrucción ABA es ejecutada si los estados Sign y Overflow no son iguales. La instrucción AND es ejecutada si los estados Sign y Overflow son iguales.
Esta instrucción provee de complementos a 2 Menores que la capacidad del Branch.

BMI - Salto si es Menor

BMI - 2B

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si el estado Sign es 1; de otro modo, se ejecuta la próxima instrucción.
En la siguiente secuencia de instrucción:

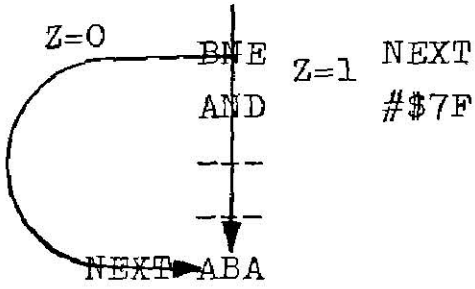


Después de la instrucción BMI, la instrucción ABA es ejecutada si el estado Sign es 1. La instrucción AND es ejecutada si el estado Sign es 0.

BNE - Salto si no es Igual

BNE - 26

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si el estado Zero es 0; deotro modo, la próxima instrucción es ejecutada.
En la siguiente secuencia de instrucción:

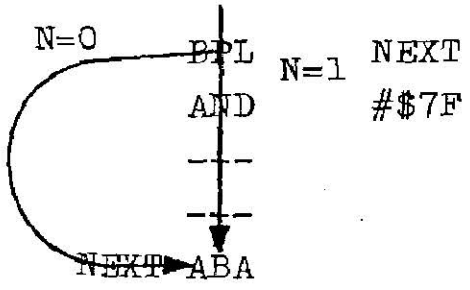


Después de la instrucción BNE, la instrucción ABA es ejecutada si el estado Zero es 0. La instrucción AND es ejecutada si el estado Zero es 1.

BPL - Salto si es Más(Positiva)

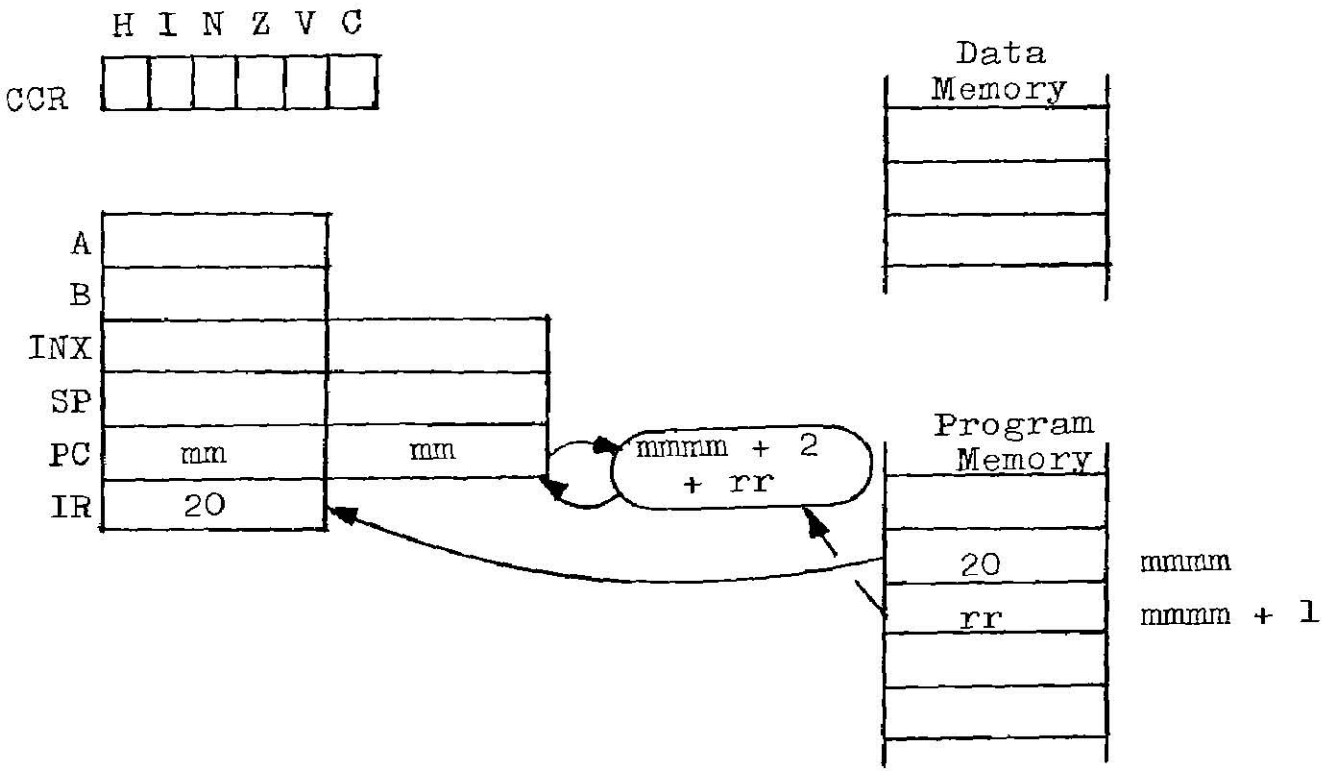
BPL - 2A

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si el estado Sign es 0; deotro modo, la próxima instrucción es ejecutada.
En la siguiente secuencia de instrucción:



Después de la instrucción BPL, la instrucción ABA es ejecutada si el estado Sign es 0. La instrucción AND es ejecutada si el estado Sign es 1.

BRA - Salto a la Instrucción Identificada en el Operando



BRA disp

20 rr

Esta instrucción suma los contenidos del segundo byte del código objeto(tomandolo como desplazamiento de 8 bits) al contenido del PC + 2; ésta llega a ser la dirección de memoria - para la próxima instrucción a ser ejecutada. Los contenidos - anteriores del PC se pierden.

En la siguiente secuencia de instrucción:

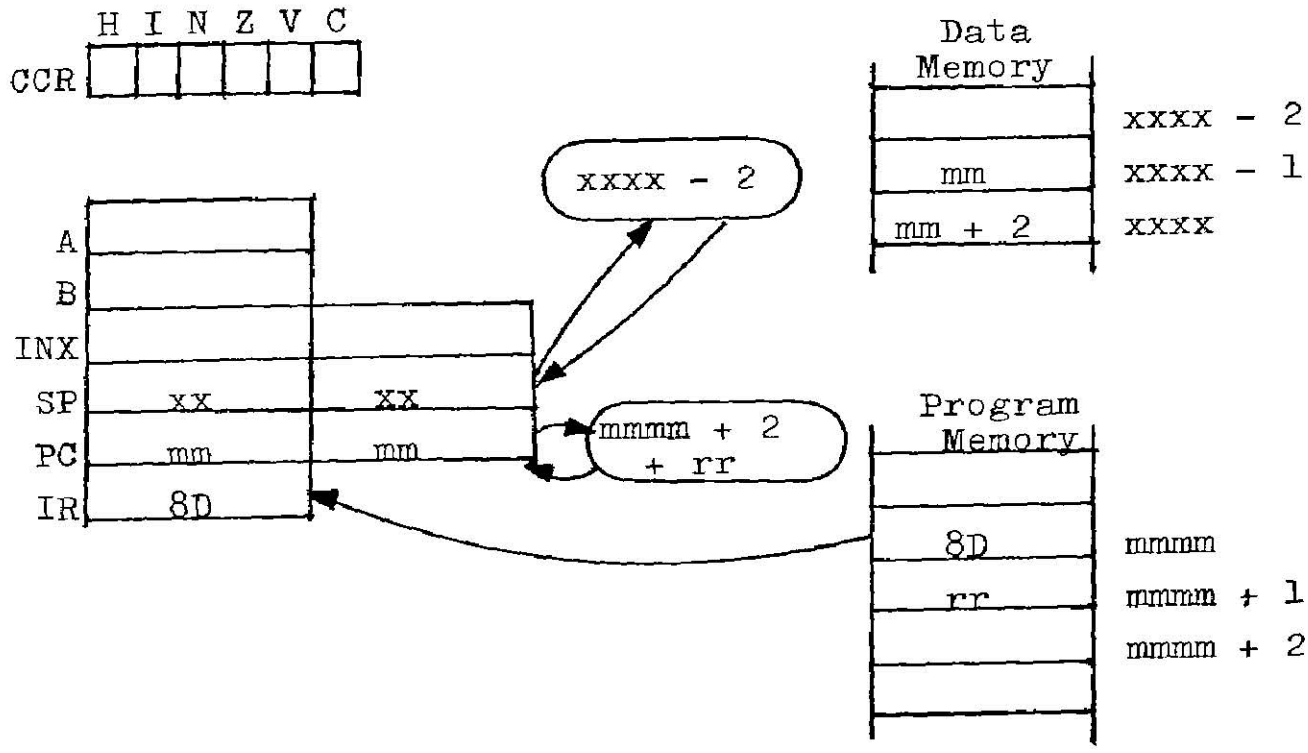
```
        BRA   NEXT
        AND   #$7F
        ---
        ---
NEXT    ABA
```

Después de la instrucción BRA, la instrucción ABA es ejecutada. La instrucción AND no puede ser ejecutada a menos que una instrucción Branch o Jump rompa la secuencia de la instrucción.

La instrucción de Salto(Branch) usa direccionamientos de Salto Relativos, los cuales son similares al Programa Relativo Paginado. La excepción es que el contenido del Contador del Programa es incrementado para apuntar a la próxima instrucción antes que el desplazamiento de 8 bits señalado sea sumado. Por lo tanto, el contenido del Contador del Programa es reemplazado por:

PC + 2 + rr

BSR - Salto a la Subrutina Identificada en el Operando



BSR	disp
8D	rr

Almacena la dirección de la instrucción seguida del BSR - en el tope de la pila; el tope de la pila es un byte de datos de memoria direccionada por el SP. Entonces sustrae 2 del SP- en orden para direccionar en el nuevo tope de la pila. Suma - el contenido del segundo byte de la instrucción más 2 para el PC y comenzar la ejecución.

Considere la secuencia de instrucción:

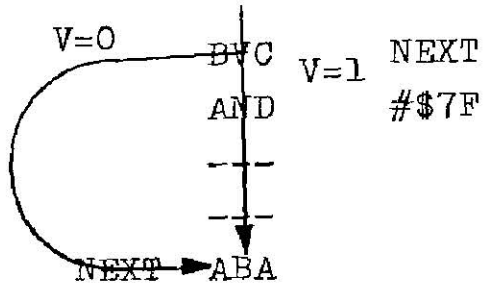
```
        BSR   SUBR
        AND   #$7F
        ---
        ---
SUBR    ABA
```

Después que la instrucción BSR es ejecutada, la dirección de-la instrucción AND es salvada en el tope de la pila. El SP es decrementado en 2. La instrucción ABA puede ser la próxima en ejecutarse.

BVC - Salto si el Overflow está Limpio

BVC - 28

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si el estado Overflow es 0; de otro modo, la próxima instrucción es ejecutada.
En la siguiente secuencia de instrucción:

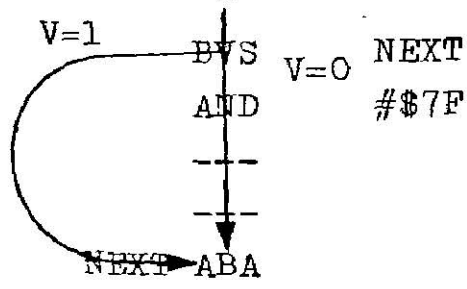


Después de la instrucción BVC, la instrucción ABA es ejecutada si el Overflow es 0. La instrucción AND es ejecutada si el Overflow es 1.

BVS - Salto si el Overflow es Colocado

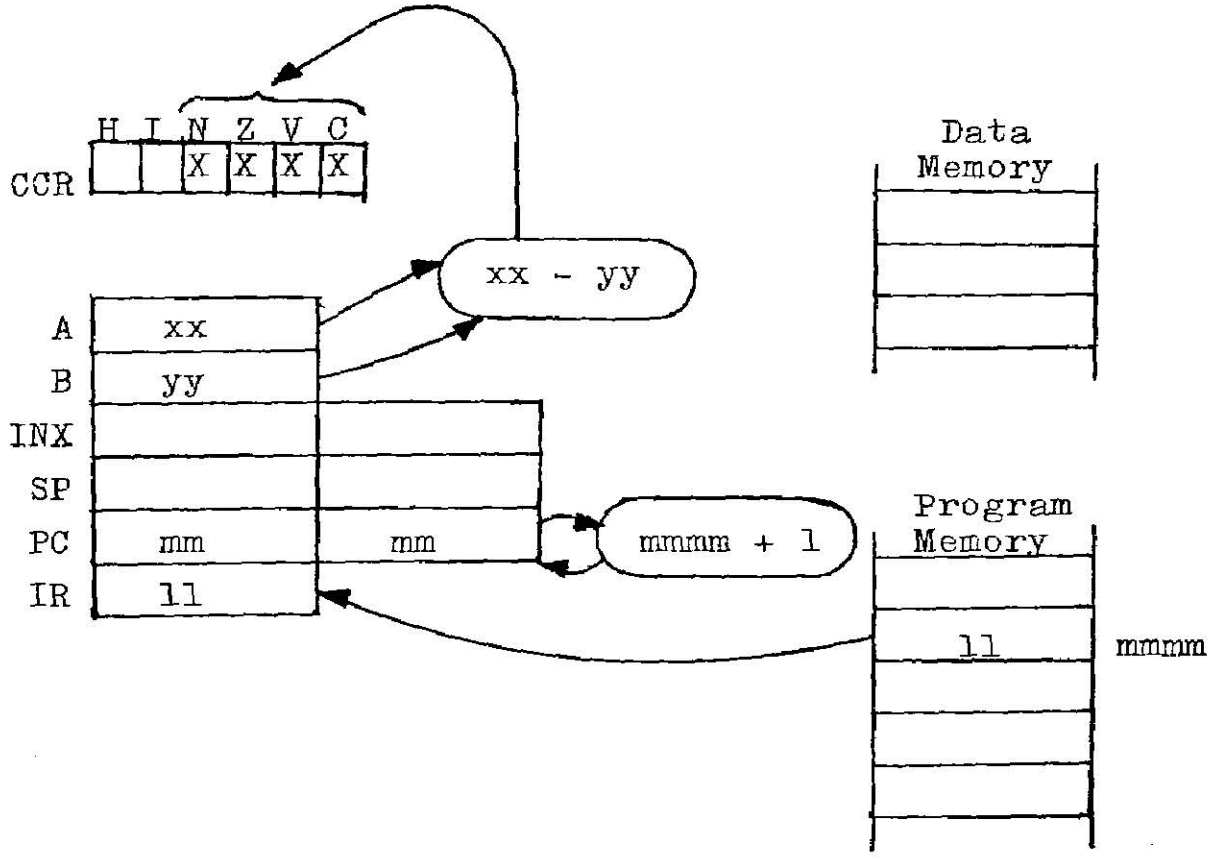
BVS - 29

Esta instrucción es idéntica a la instrucción BRA, excepto que el salto es ejecutado solo si el Overflow es 1; de o-- tro modo, la próxima instrucción es ejecutada.
En la siguiente secuencia de instrucción:



Después de la instrucción BVS, la instrucción ABA es ejecutada si el estado Overflow es igual a 1. La instrucción AND es-ejecutada si el Overflow es 0.

CBA - Compara Acumuladores



CBA - 11

Sustrae el contenido del acumulador B del contenido del - acumulador A. Descartando el resultado, esto es, no afectando el contenido de cualquier acumulador, pero modificando las -- banderas de estados que reflejan el resultado de la operación. Suponer xx=$E3_{16}$ y yy=$A0_{16}$. Después que la instrucción:

CBA

es ejecutada, el Acc A contiene $E3_{16}$ y el Acc B contiene $A0_{16}$ pero los estados son modificados.

E3 = 1 1 1 0 0 0 1 1

Compl.a 2 de A0 = 0 1 1 0 0 0 0 0

0 1 0 0 0 0 1 1 ← Coloca Z=0

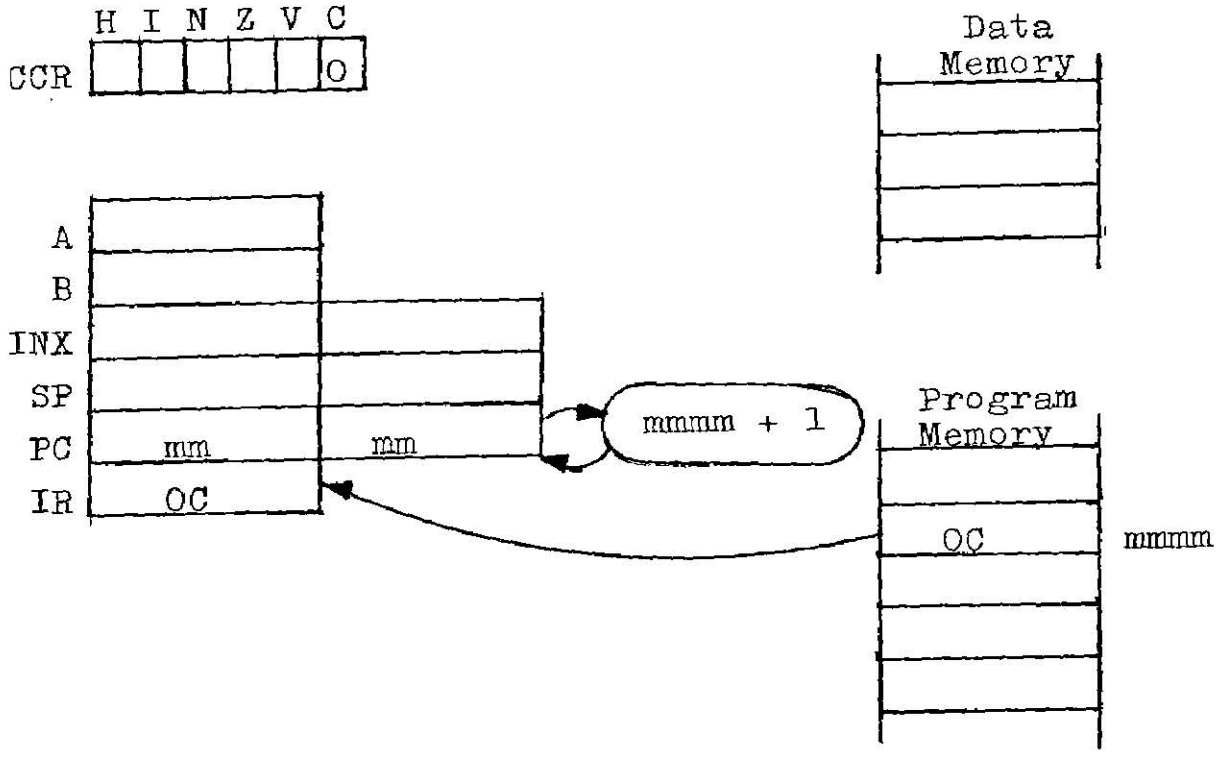
Coloca C=0

N=0

1 ⊻ 1=, Coloca V=0

Note que el Carry resultante es complementado.

CLC - Limpia el Carry

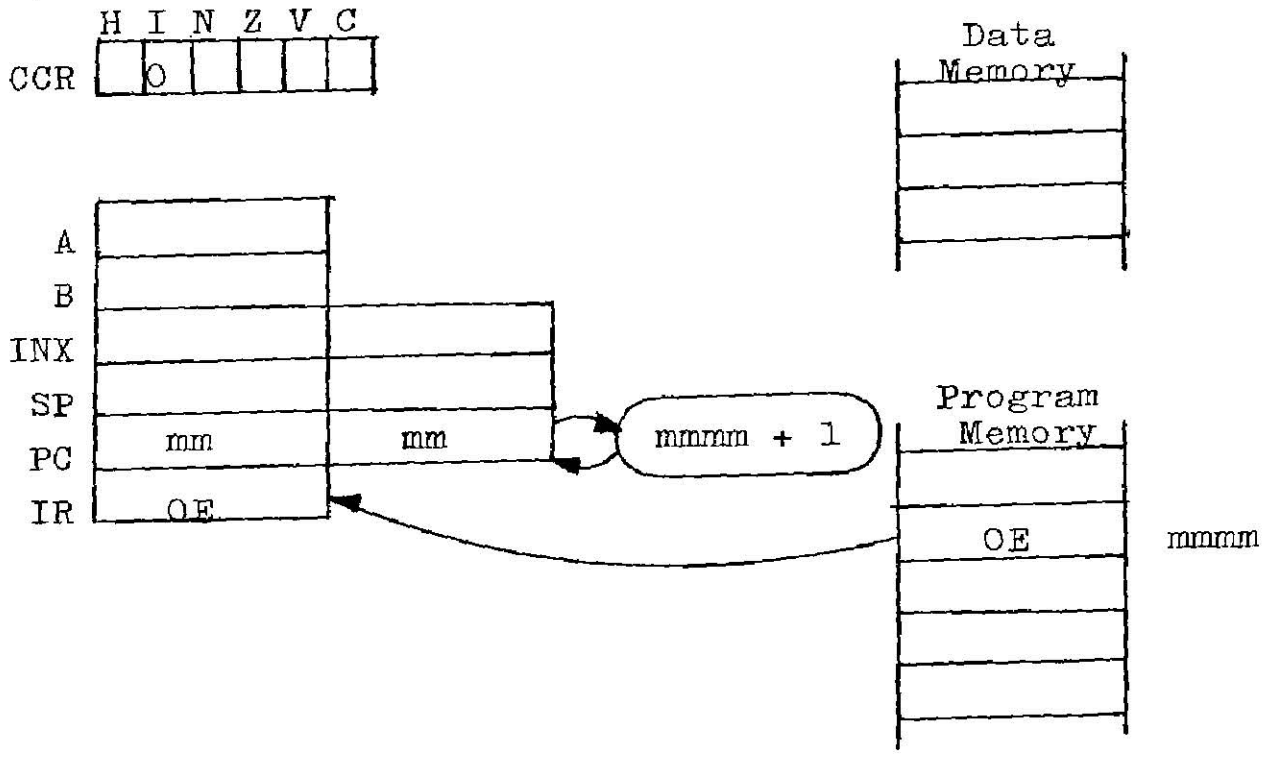


CLC - OC

Limpia el estado Carry. Ninguno de los contenidos de los estados restantes son afectados.

El estado Carry es también limpiado por instrucciones CLR y TST.

CLI - Limpia la Máscara de Interrupción

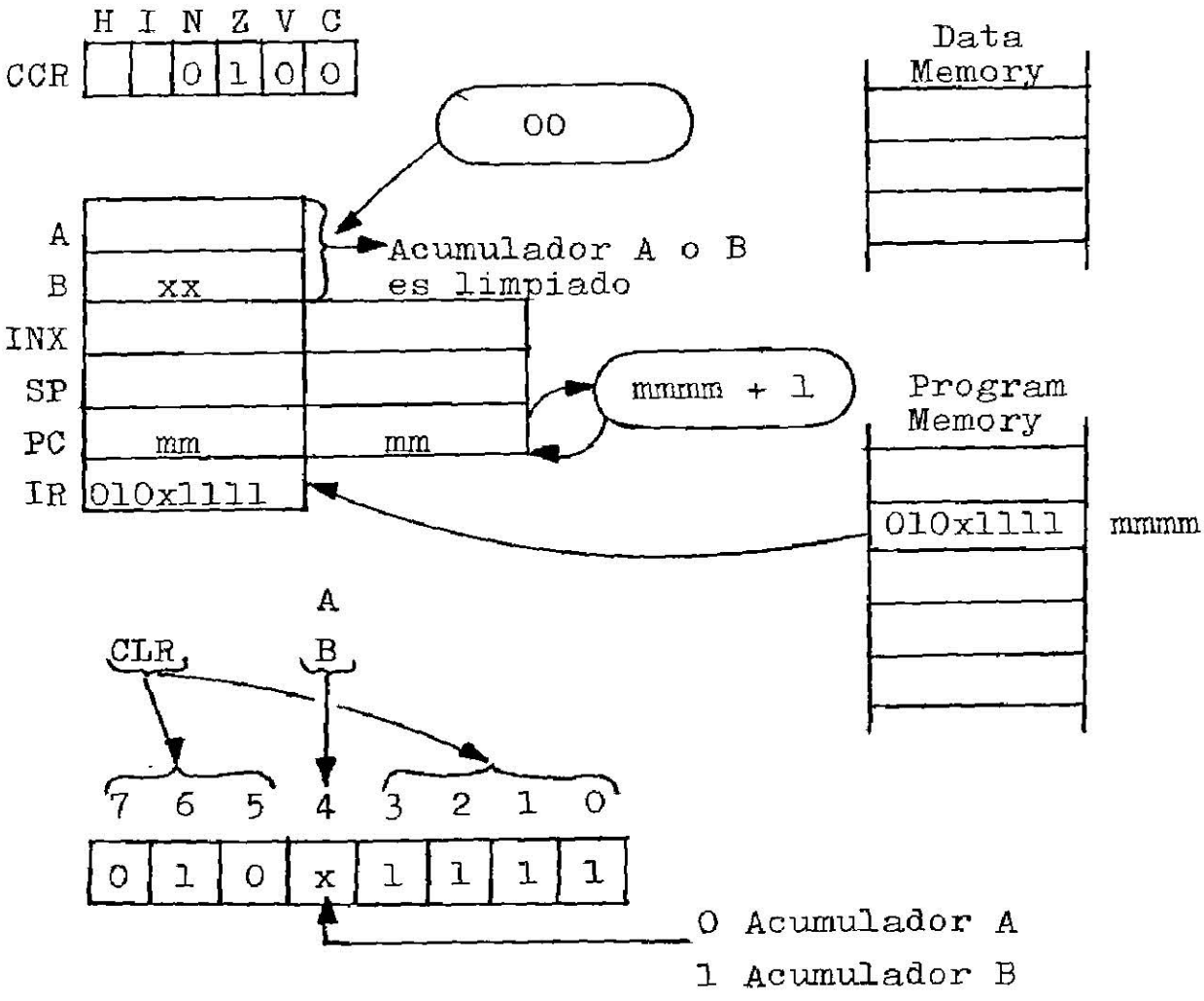


CLI - OE

Limpia el bit de máscara de interrupción en el registro - de código de condiciones. Esta instrucción habilita el servicio de habilidad de interrupción del MC 6800, esto es, el MC-6800 puede responder a la línea de control de interrupción -- requerida. Ningún otro estado es afectado.

CLR - Limpia el Acumulador o la Memoria

Esta instrucción limpia un acumulador específico o un --- byte de memoria seleccionada. La bandera Zero es 1; los estados Sign, Carry y Overflow son 0.
Primero limpiaremos el Acumulador:

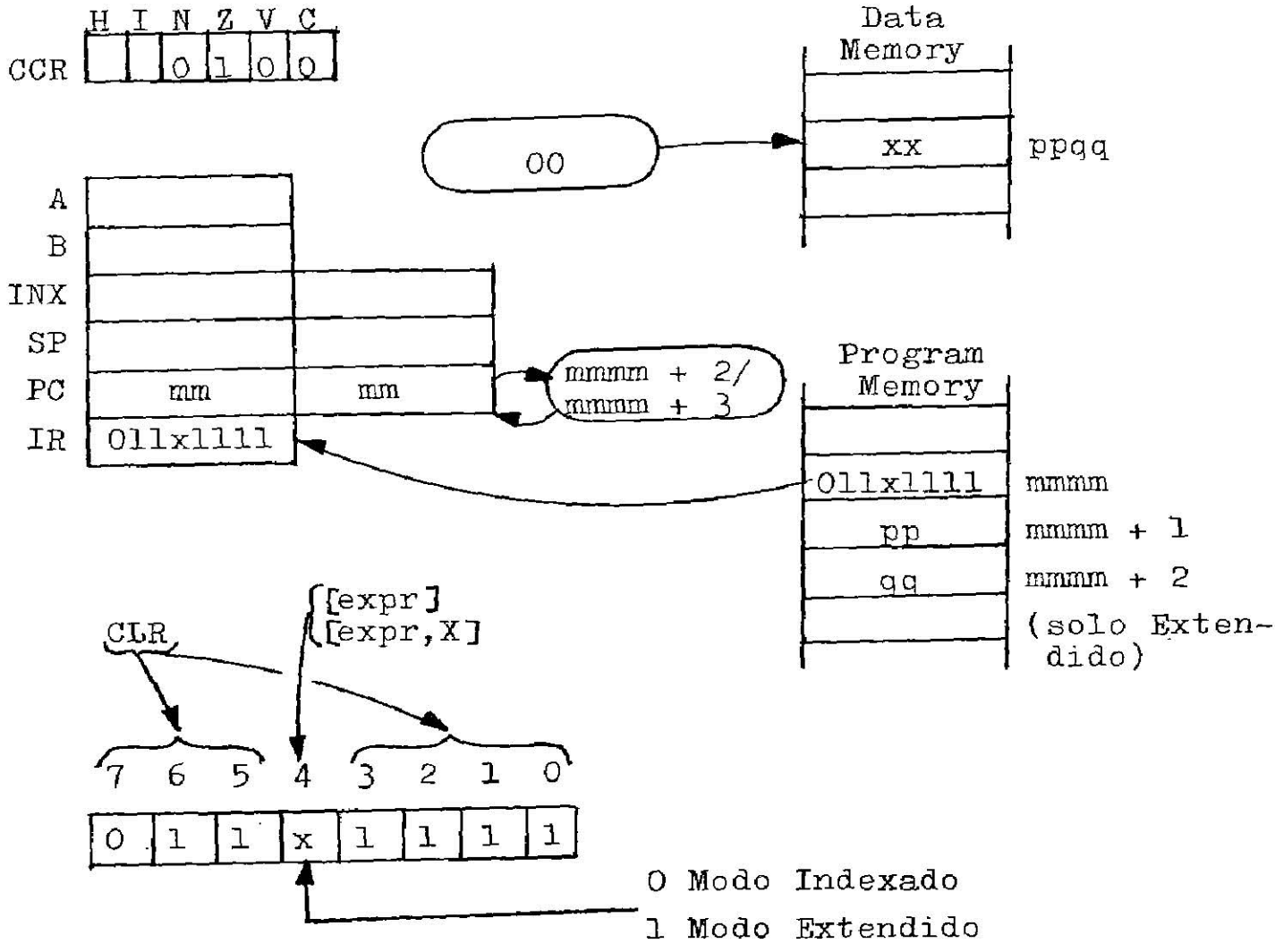


Limpia el contenido de un acumulador específico. Suponer el - acumulador B igual a 43_{16}. Después que la instrucción:

CLR

es ejecutada, el acumulador B contiene 00. Además Sign, Overflow y Carry son 0; Zero es 1.

La instrucción CLR también tiene 2 modos de direccionamiento-Indexado y Extendido.

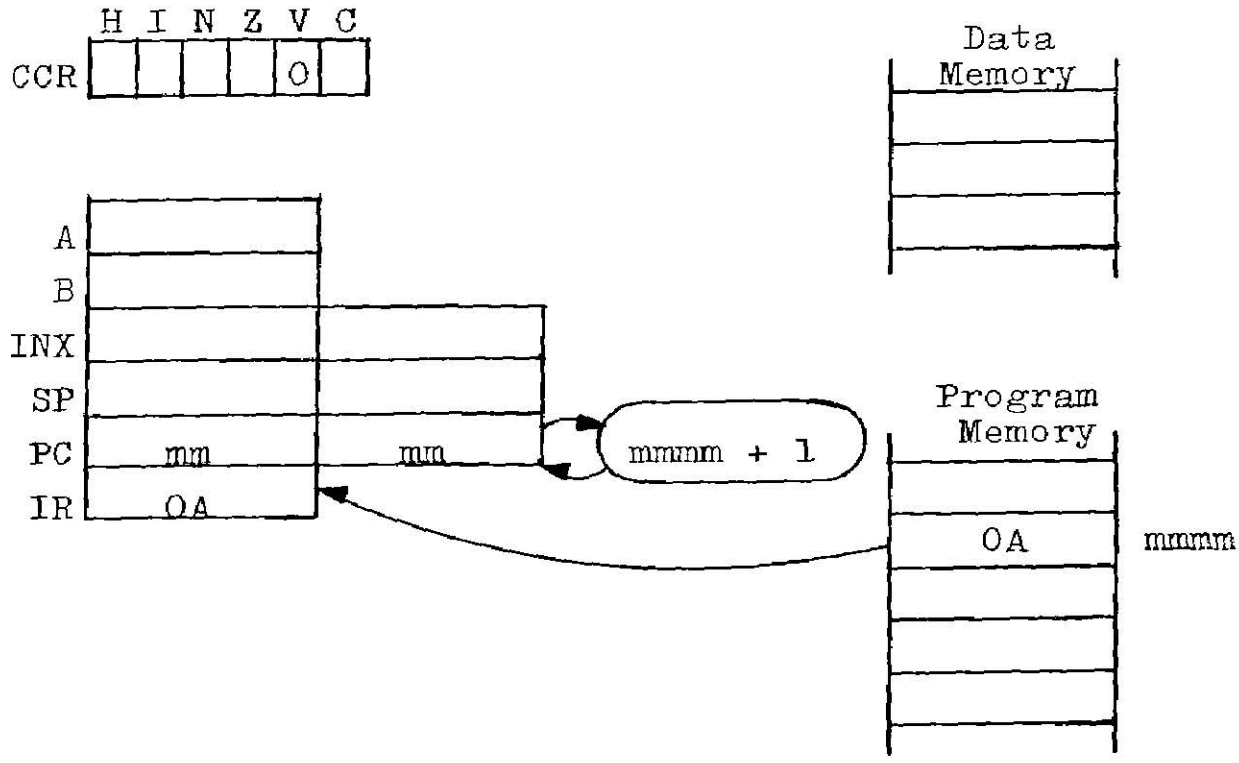


Suponer pp=43_{16}, qq=14_{16} y xx=05_{16}. Después que se ejecuta:

CLR $4314

el contenido de la localidad de memoria 4314 es 00 y la bandera del estado puede modificarse.

CLV - Limpia el Overflow

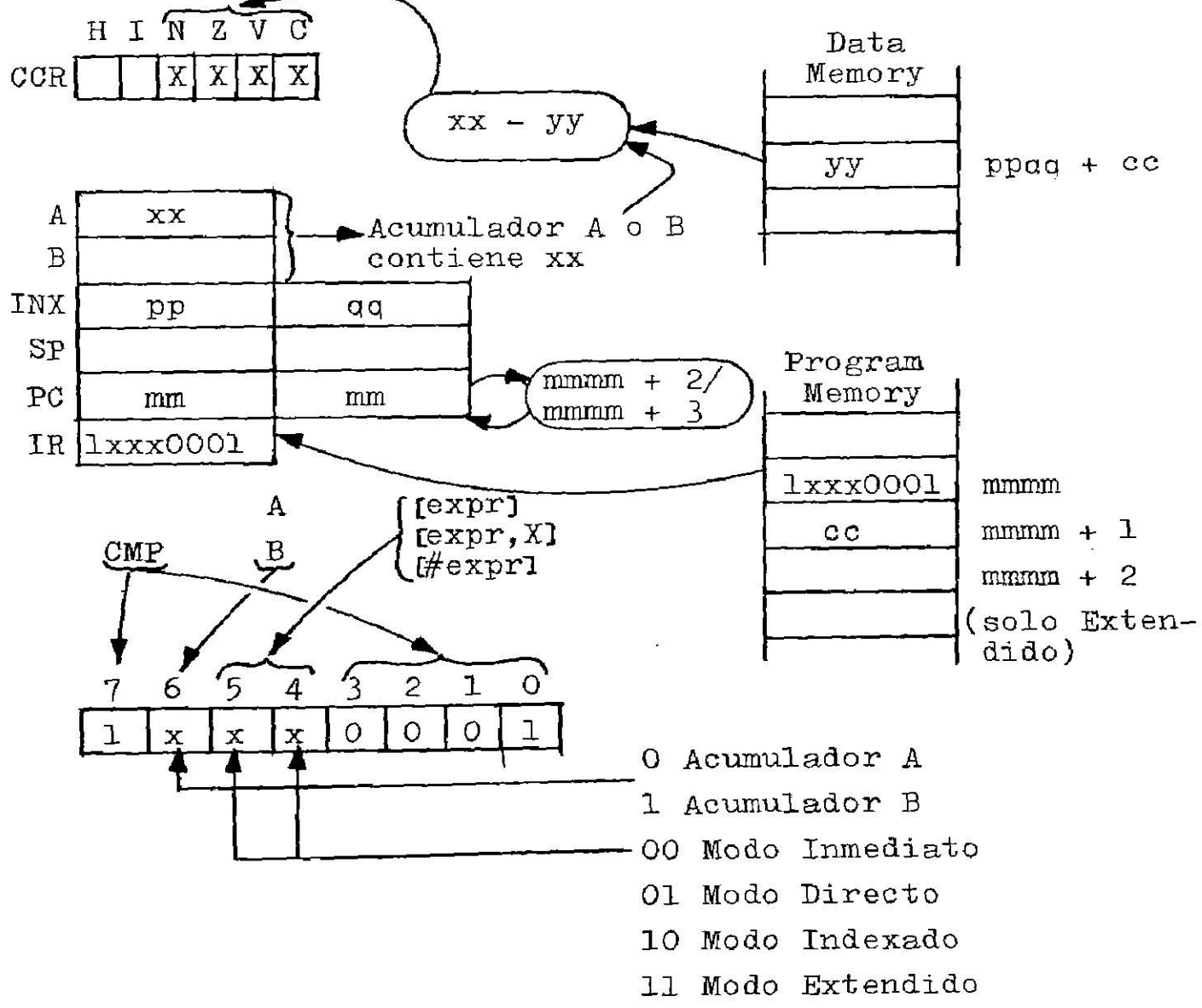


CLV - OA

Limpia el bit de Overflow del registro de código de condiciones. Ningún otro registro o estado es afectado.

CMP - Compara el Acumulador con Memoria

Esta instrucción sustrae el contenido de una localidad de memoria seleccionada al contenido del acumulador A o el B, colocando las banderas de condición por consiguiente, pero sin-alterar el contenido del acumulador o el byte de memoria. Esta instrucción ofrece la misma opción de direccionamiento de-memoria que la instrucción ADC. Esta instrucción puede ser -- ilustrada usando el modo indexado.



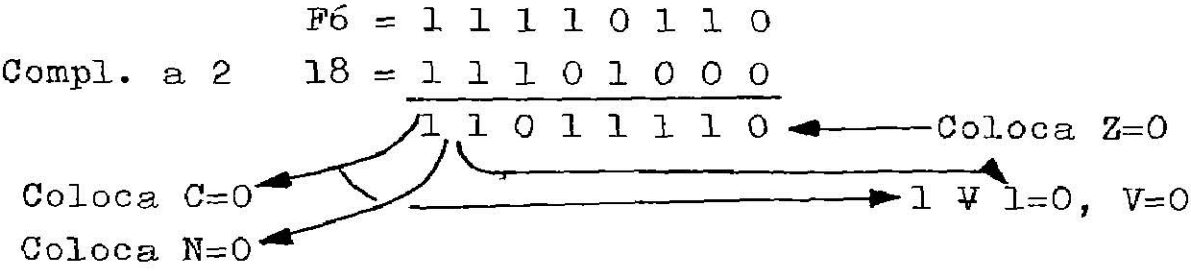
Sustrae el contenido del byte de memoria seleccionada del contenido de un acumulador específico y coloca los estados -- Sign, Zero, Overflow y Carry reflejando el resultado de la -- sustracción.

Suponer xx=$F6_{16}$, yy=18_{16} y cc=43_{16}.

Después que la instrucción:

```
CMP B  $43,X
```

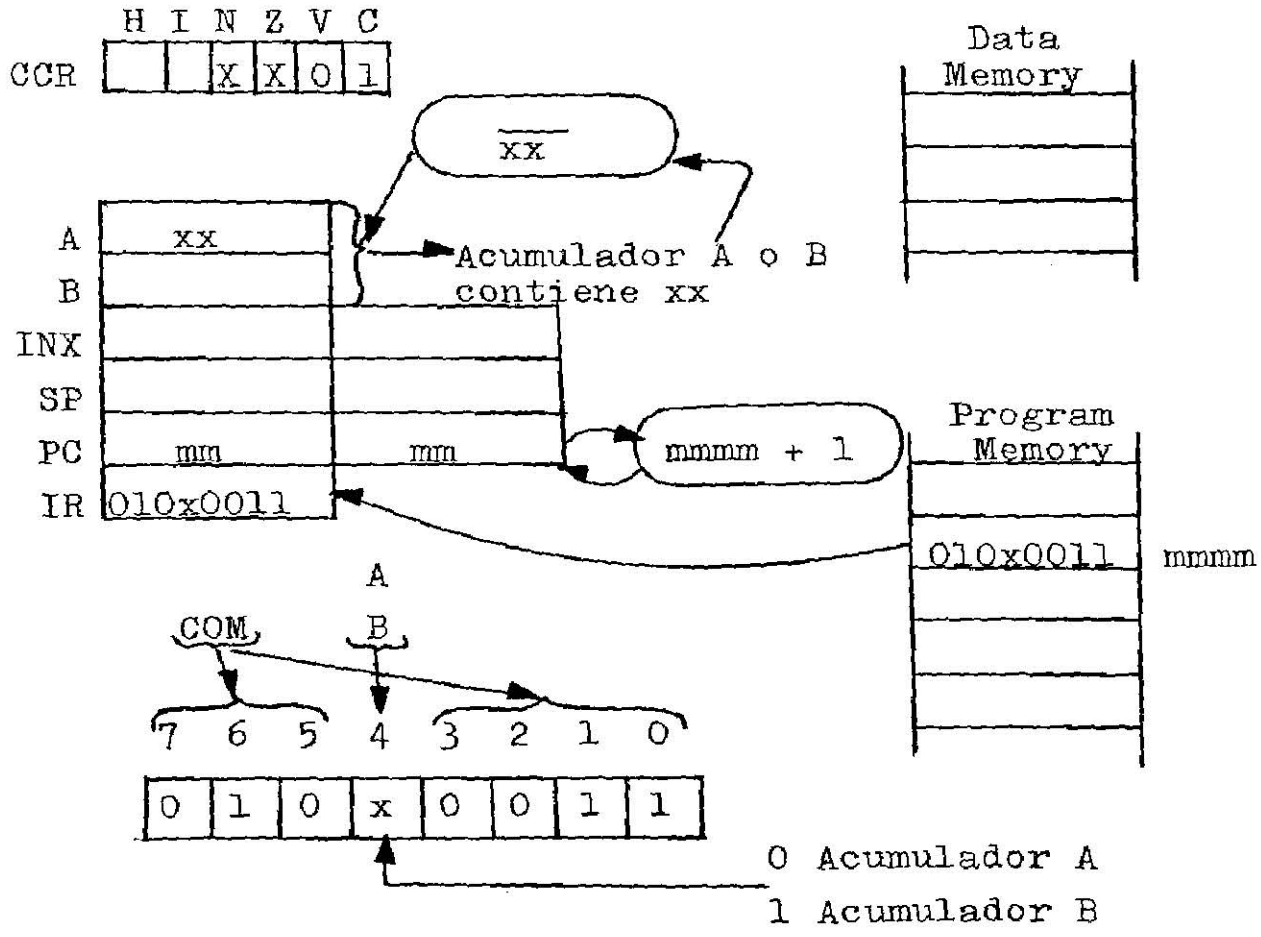
es ejecutada, el acumulador B contiene aún $F6_{16}$ y la localidad ppqq+43_{16} contiene aún 18_{16} pero los estados pueden ser - modificados.



Note que C es el complemento del Carry resultante.

COM - Complementa el Acumulador o Memoria

Esta instrucción complementa un acumulador específico o un byte de memoria seleccionado.
Primero, consideremos el complemento del Acumulador.



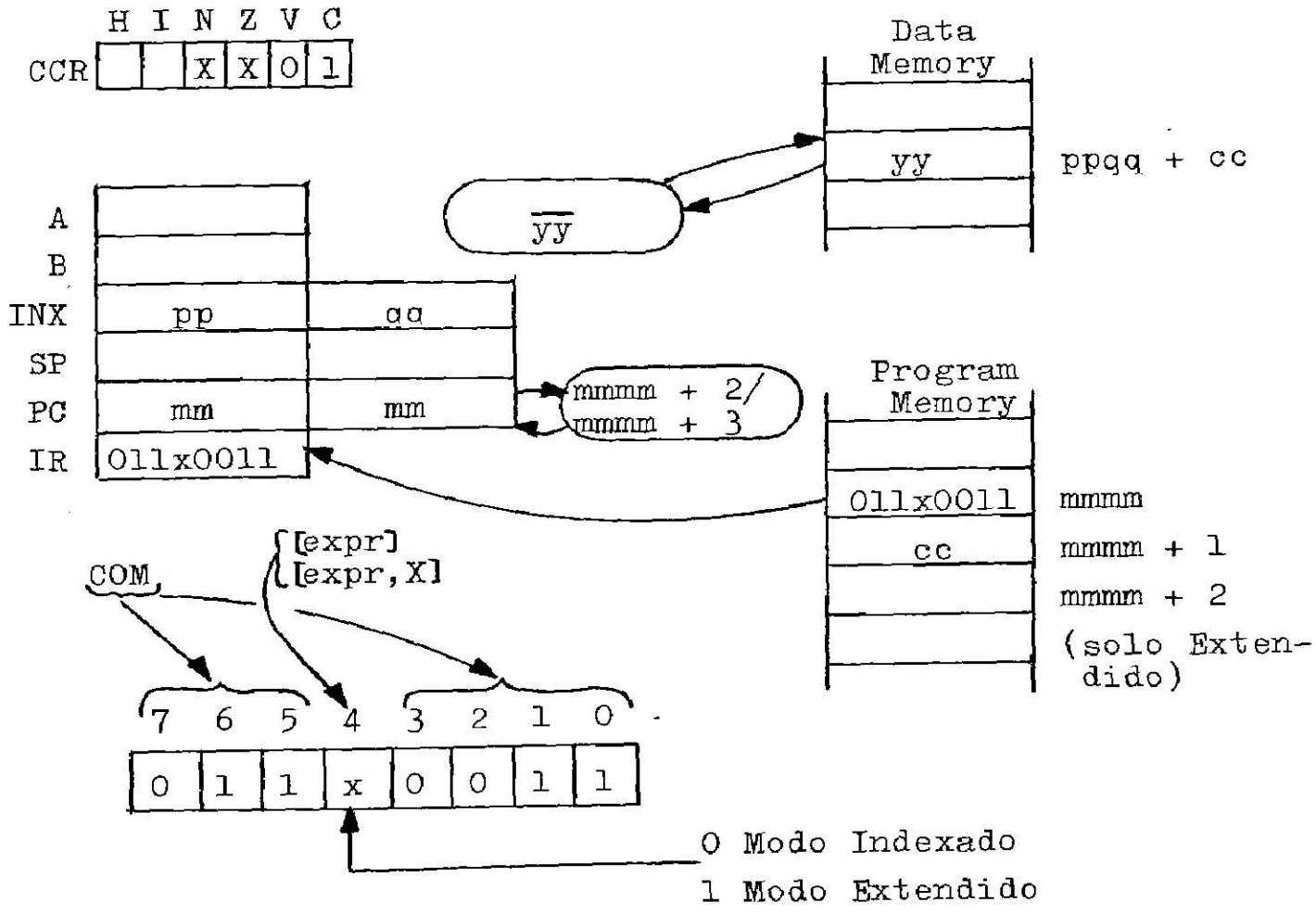
Complementa el contenido de un acumulador específico. Ningún otro estado o registro es afectado.
Suponer que el Acc. B contiene $3A_{16}$. Después que la instrucción:

COM B

es ejecutada, el Acumulador B contiene $C5_{16}$.

3A = 0 0 1 1 1 0 1 0
Complemento = 1 1 0 0 0 1 0 1 ← Coloca Z=0
Coloca C=1 ← V=0
N=1

La instrucción COM también tiene 2 modos de direccionamiento- Extendido e Indexado.



Suponer que el contenido del Registro de Indice es 0100_{16} y-- el contenido de la localidad de memoria 0113_{16} es 23_{16}. Des-- pués que la instrucción:

```
COM   $13,X
```

es ejecutada, la localidad de memoria 0113_{16} contiene DC_{16}.

23 = 0 0 1 0 0 0 1 1

Complemento= 1 1 0 1 1 1 0 0 ← Coloca Z=0

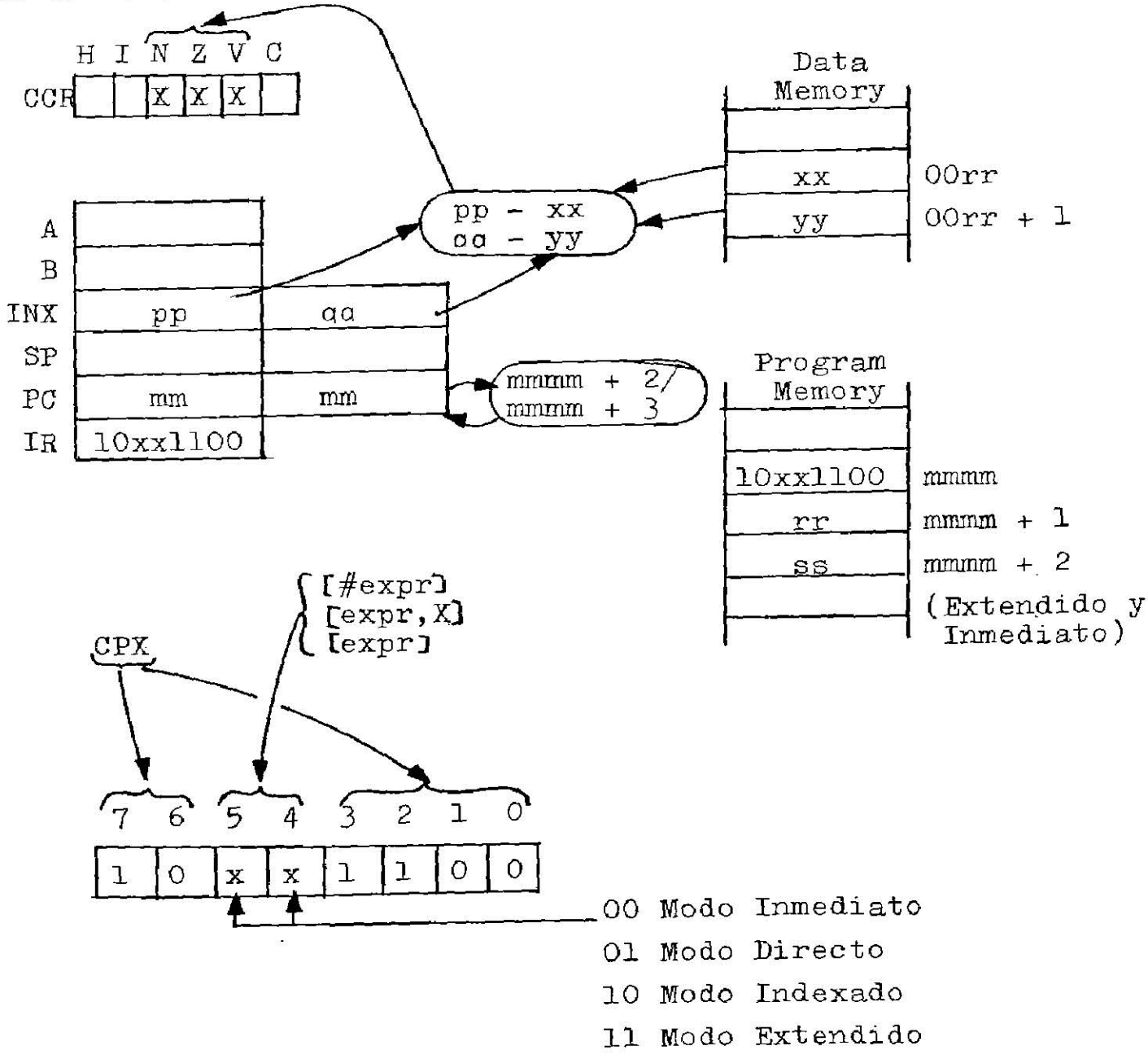
V es limpiado

Coloca C=1

N=1

CPX - Compara el Registro de Indice

Esta instrucción compara el contenido del registro de índice con el contenido de 2 localidades de memoria seleccionada. Esta instrucción ofrece la misma opción de direccionamiento de memoria que la instrucción ADC. Se puede ilustrar usando el modo de direccionamiento Directo.

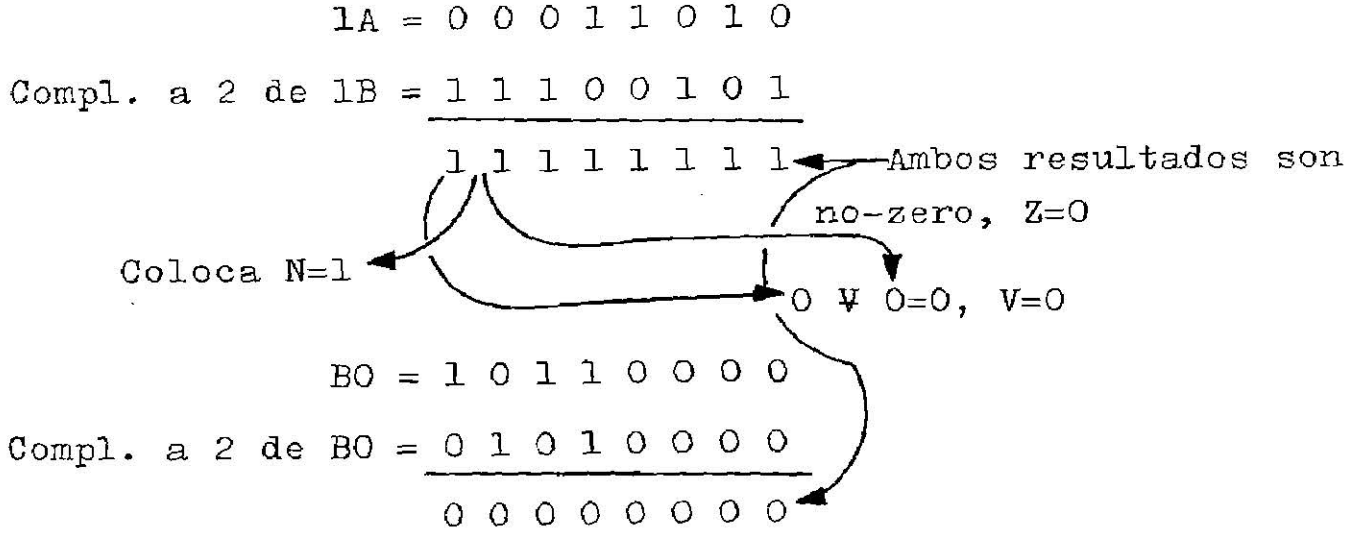


Sustrae el contenido del byte de memoria inmediatamente seguido del byte de memoria seleccionado desde un byte abajo del registro de índice, y descarta el resultado. Sustrae el contenido del byte de memoria seleccionado desde el byte alto del registro de índice. Coloca los estados Sign, y Overflow de acuerdo al resultado, coloca el estado Zero en 1 si el resultado de ambas sustracciones fué 0, y descarta el resultado.

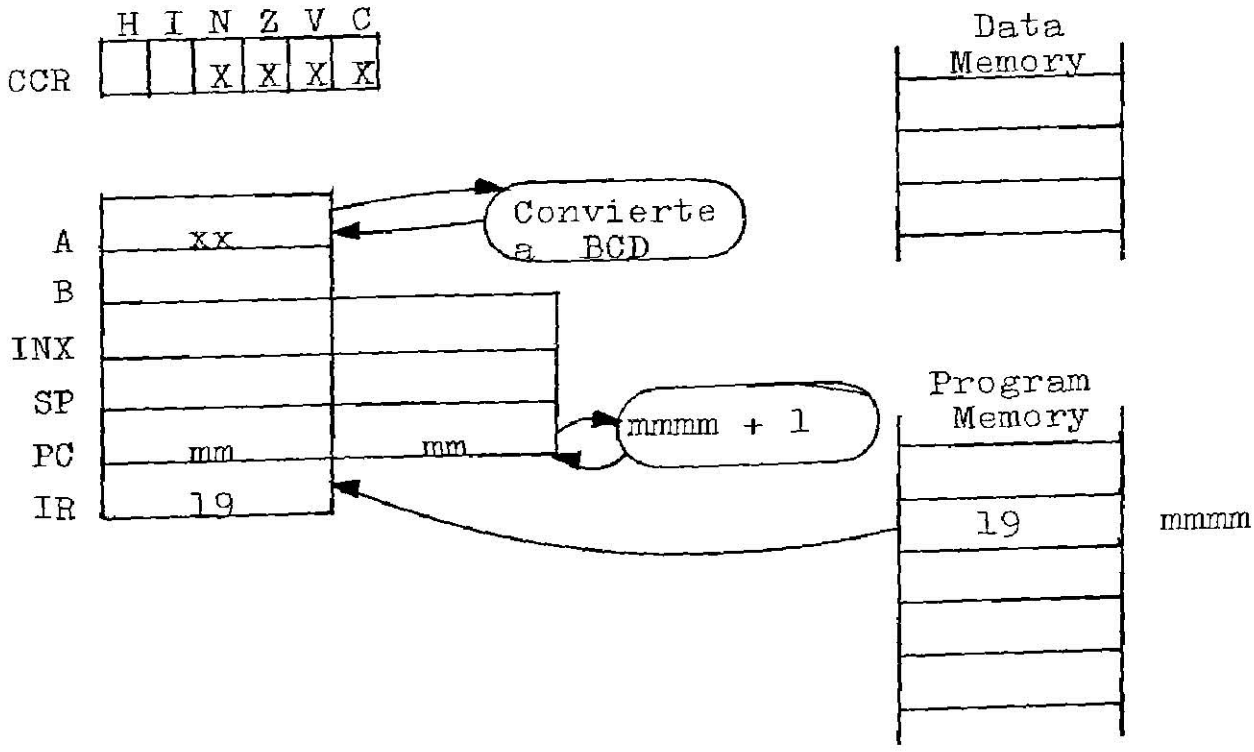
Suponer pp=$1A_{16}$, qq=BO_{16}, xx=$1B_{16}$, yy=BO_{16} y rr=43_{16}. Después que la instrucción:

```
CPX  $43
```

es ejecutada, el registro de índice contiene $1ABO_{16}$, la localidad 0043 contiene $1B_{16}$ y la localidad 0044 contiene BO_{16} pero los estados Sign, Zero y Overflow pueden modificarse como sigue:



DAA - Ajuste Decimal al Acumulador



DAA - 19

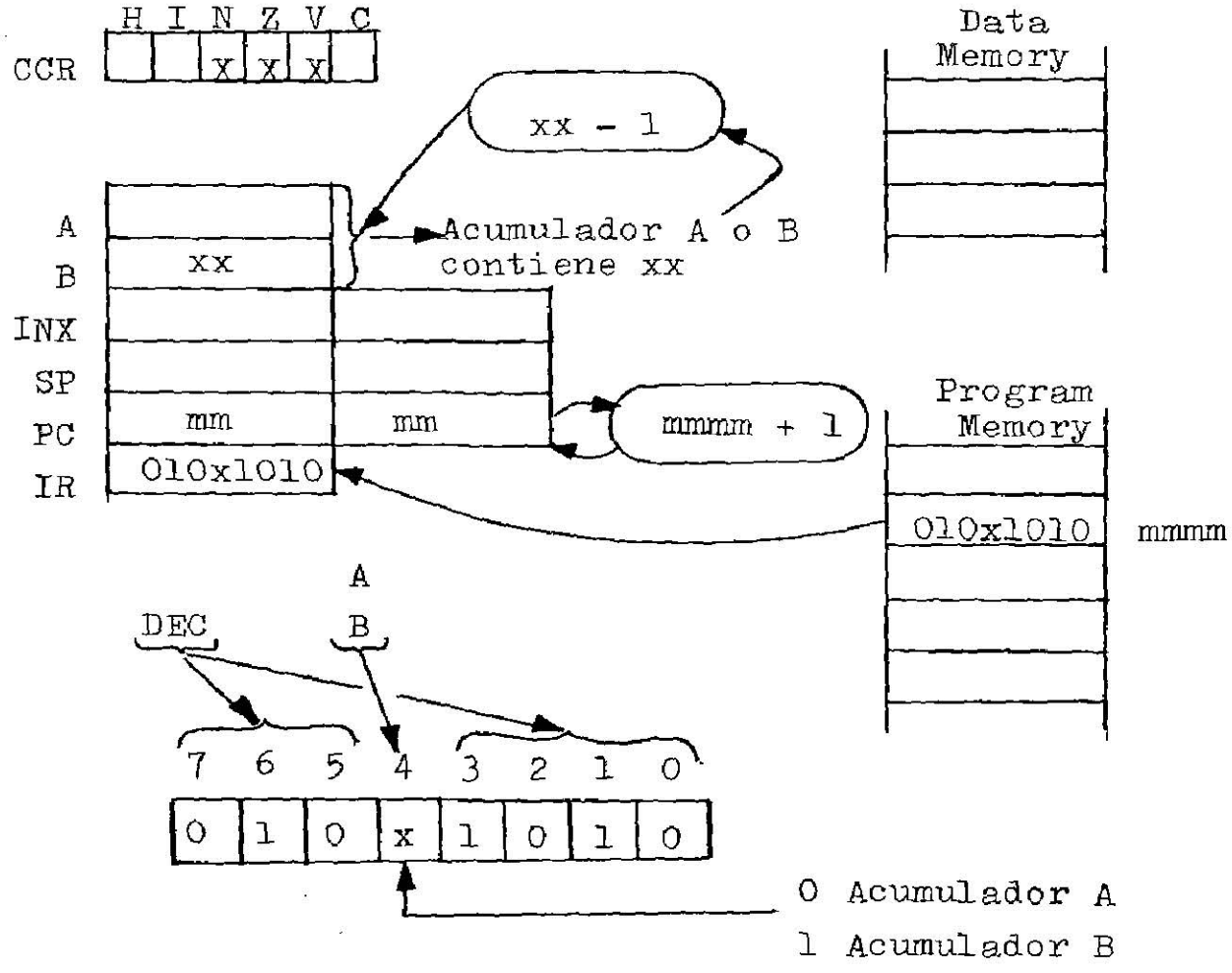
Convierte el contenido del acumulador A a la forma Código Binario-Decimal. Esta instrucción debe ser usada solo después de sumar 2 números BCD, esto es, mirando ABA DAA o ADD A DAA- o ADC A DAA como instrucción de adición decimal compuesta la- cual opera en la fuente BCD para generar respuestas BCD. Suponer Acc. A = 39_{16} y Acc. B = 47_{16}. Después que la instrucción:

ABA
DAA

es ejecutada, el acumulador contiene 86_{16} no 80_{16}. Las banderas SIGN y Zero son modificadas reflejando el estado que representan. El estado Overflow es destruido y el estado Carry- es colocado o recolocado como si tomara lugar una adición BCD hipotética.

DEC - Decrementa el Acumulador o Memoria

Esta instrucción decrementa un acumulador específico o un byte de memoria seleccionado.
Primero, consideremos el decremento del Acumulador.



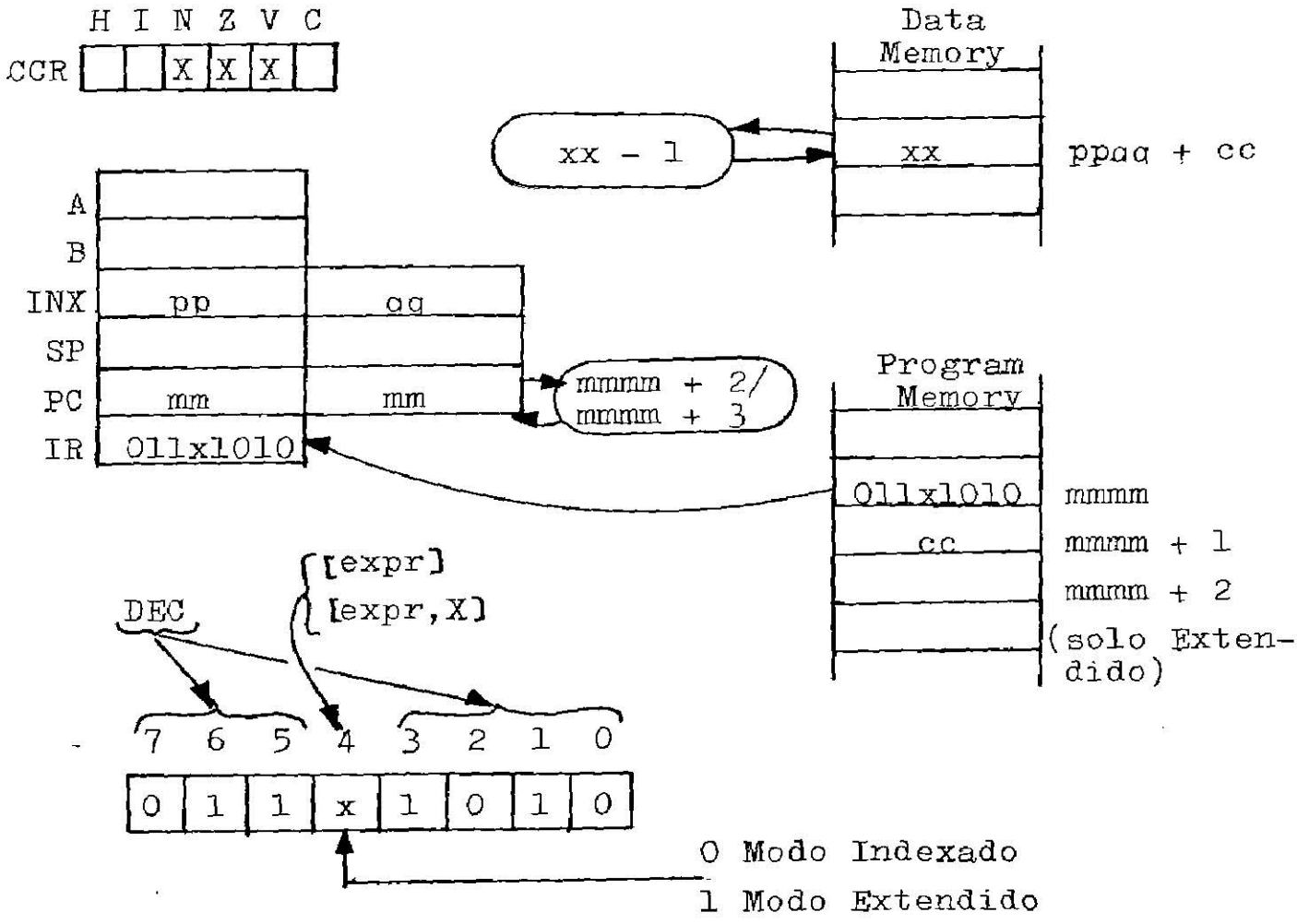
Sustrae 1 del contenido del Acumulador específico.
Suponer Acc. B = $3A_{16}$. Después que la instrucción:

DEC B

es ejecutada, el acumulador B contiene 39_{16}.

3A = 0 0 1 1 1 0 1 0
Compl. a 1 de 1= 1 1 1 1 1 1 1 1
0 0 1 1 1 0 0 1 ← Coloca Z=0
Coloca N=0 ← → 1 ∨ 1=0, V=0

La instrucción DEC también tiene 2 modos de direccionamiento, Extendido e Indexado.



Decrementa el contenido del byte de memoria seleccionada. Si xx= $A5_{16}$, ppqq= 0100_{16} y cc=$0A_{16}$, después de la ejecución de la instrucción:

DEC $0A,X

la localidad de memoria 0100_{16} contiene $A4_{16}$.

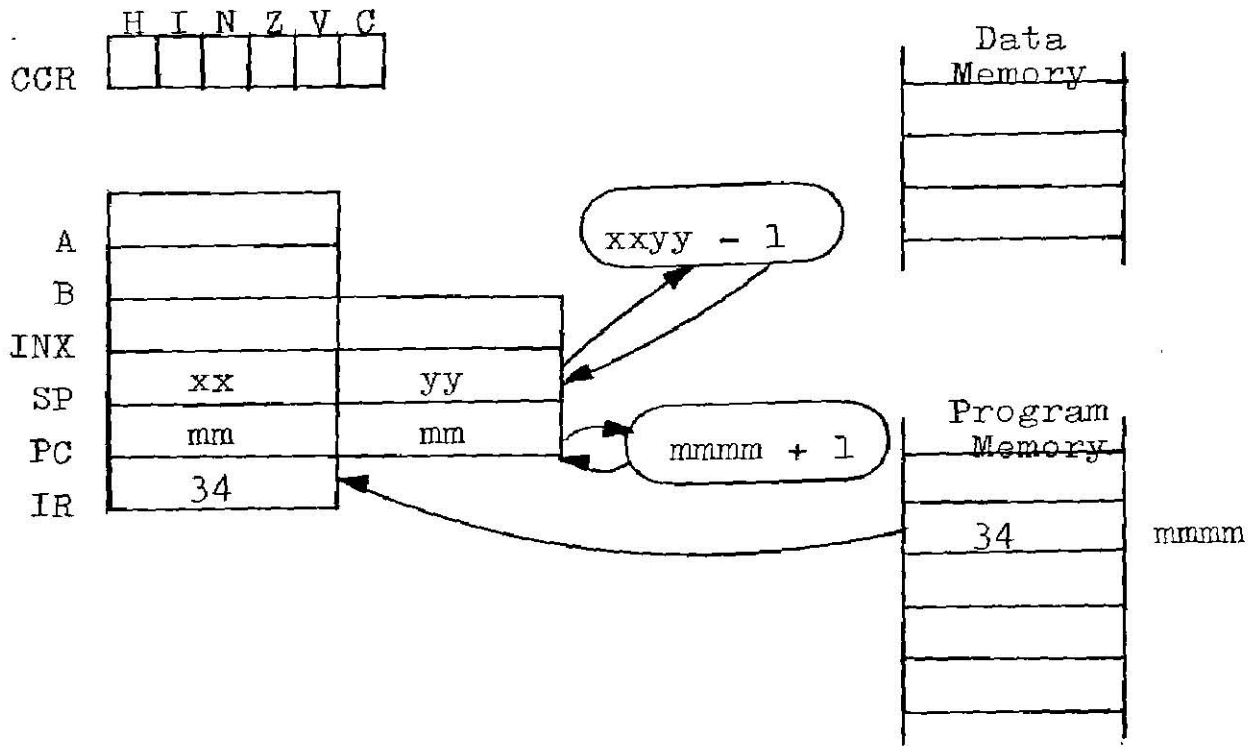
A5 = 1 0 1 0 0 1 0 1

Compl. a 1 de 1= 1 1 1 1 1 1 1 1

1 0 1 0 0 1 0 0 ← Coloca Z=0

Coloca N=1 ← (1) → 1 ∀ 1, V=0

DES - Decrementa el Apuntador de Pila



DES - 34

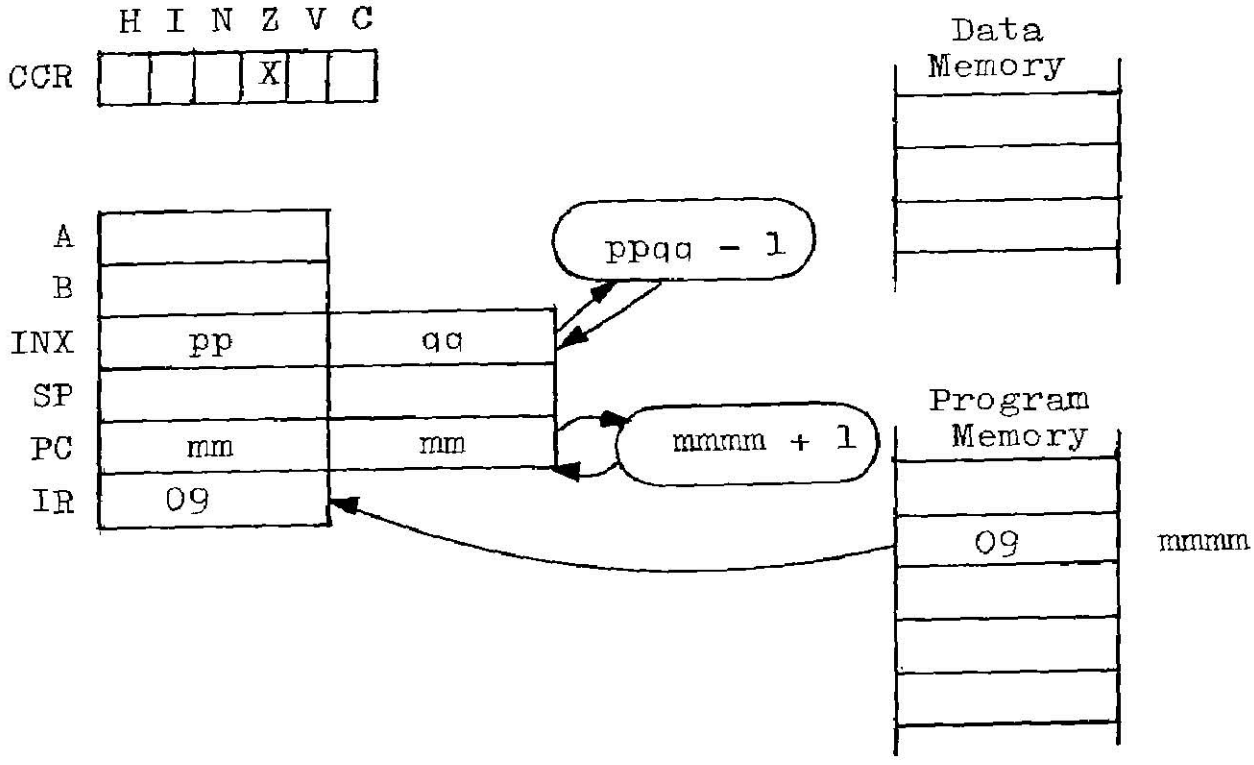
Sustrae 1 al valor de 16 bits del Apuntador de Pila. Ningún otro registro o estado es afectado.

Suponer que el apuntador de la pila contiene $2F7A_{16}$. Después que la instrucción:

DES

es ejecutada, el Apuntador de Pila contiene $2F79_{16}$.

DEX - Decrementa el Registro de Indice



DEX - 09

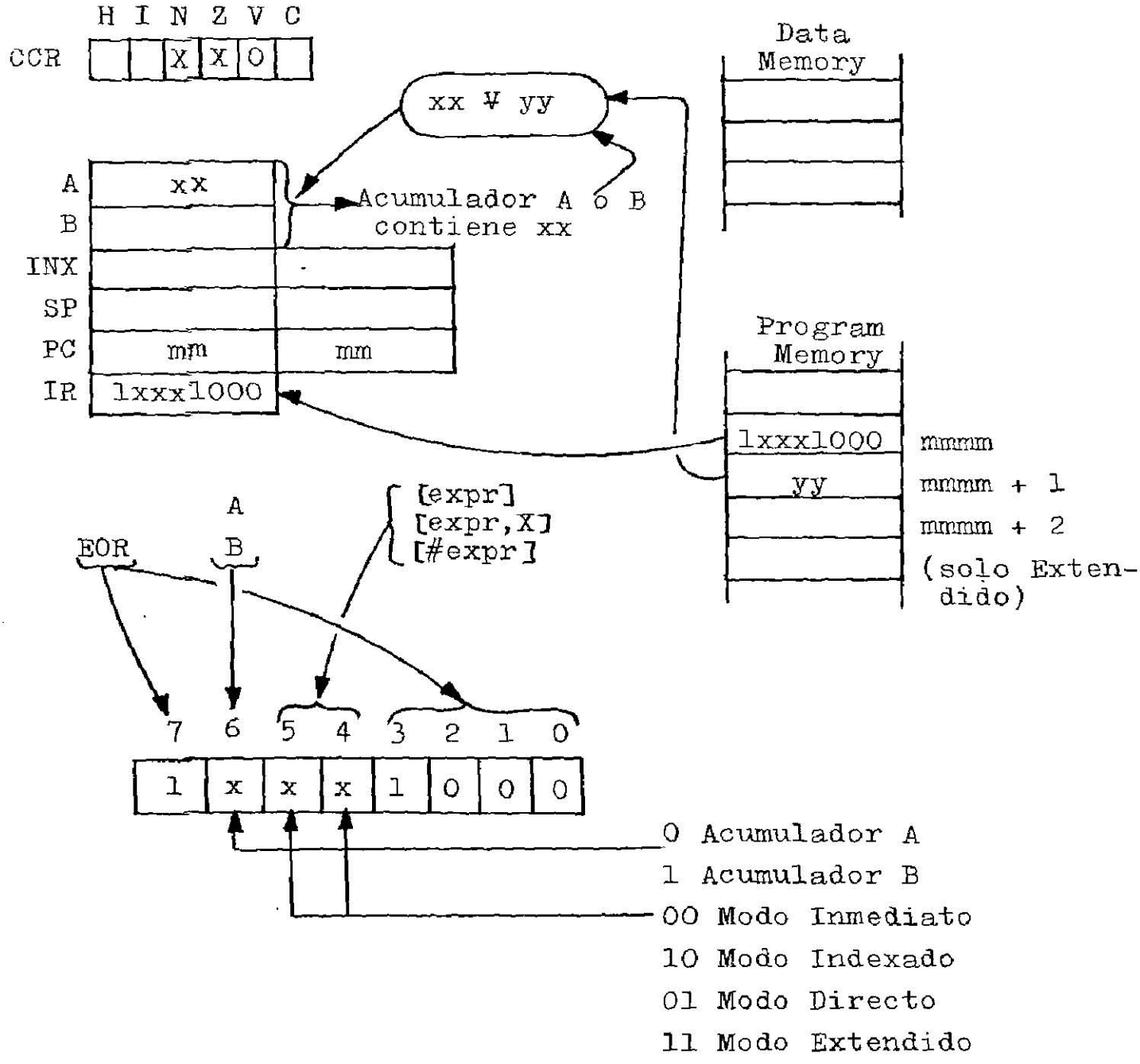
Sustrae 1 del valor de 16 bits del Registro de Indice. El estado Zero es 1 si el resultado de los 16 bits es 0. Suponer que el registro de índice contiene $310C_{16}$. Después -- que la instrucción:

DEX

es ejecutada, el Registro de Indice contiene $310B_{16}$ y el estado Zero es 0.

EOR - OR-exclusivo del Acumulador con Memoria

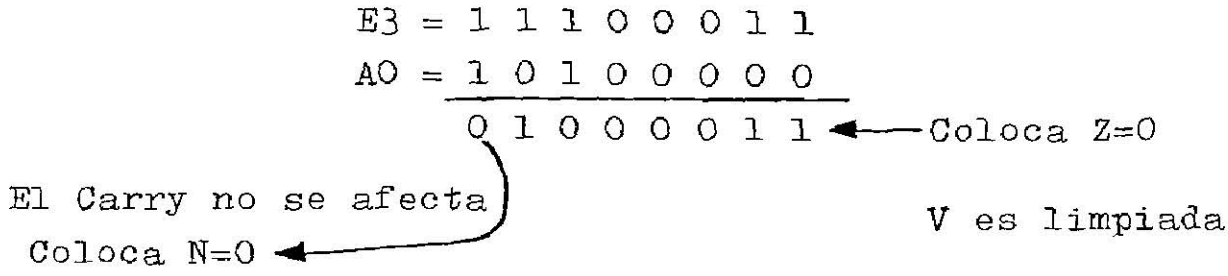
Aplica el OR-exclusivo al contenido del acumulador A o B- con el contenido de un byte de memoria seleccionada. Esta --- instrucción ofrece la misma opción de direccionamiento de memoria que la instrucción ADC y es ilustrada usando el modo -- inmediato.



Aplica el OR-exclusivo al contenido de un acumulador específico con el contenido de una localidad de memoria seleccionada, tratando ambos operandos como simples datos binarios. Suponer que xx=$E3_{16}$ y yy=AO_{16}. Después que la instrucción:

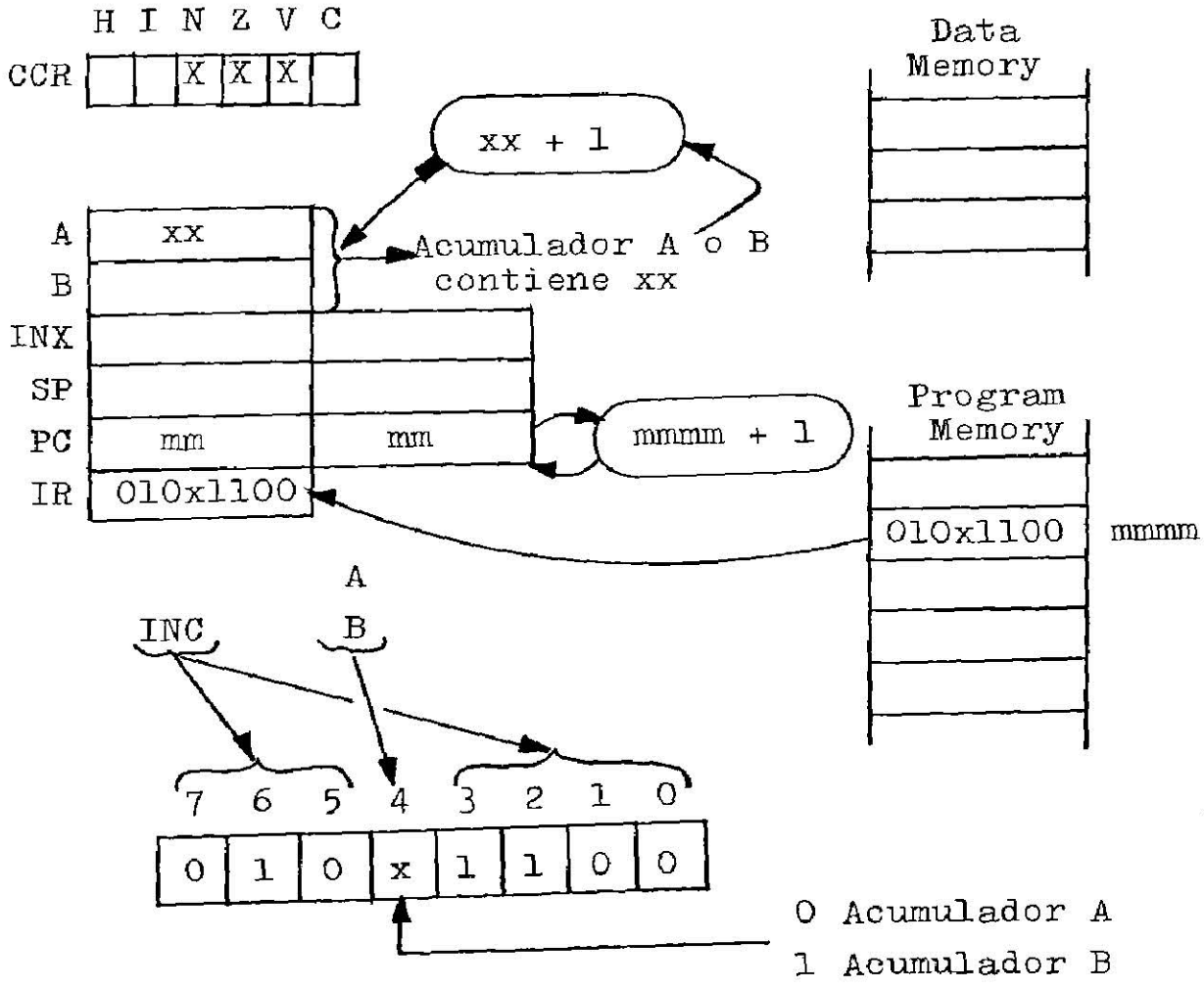
```
EOR A  #$AO
```

es ejecutada, el acumulador A contiene 43_{16}.



INC - Incrementa el Acumulador o la Memoria

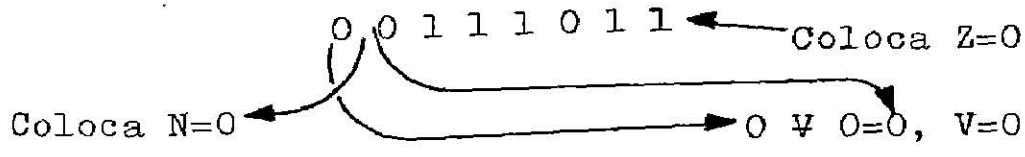
Esta instrucción incrementa un acumulador específico o un byte de memoria seleccionado.
Primero, consideremos el incremento en el Acumulador.



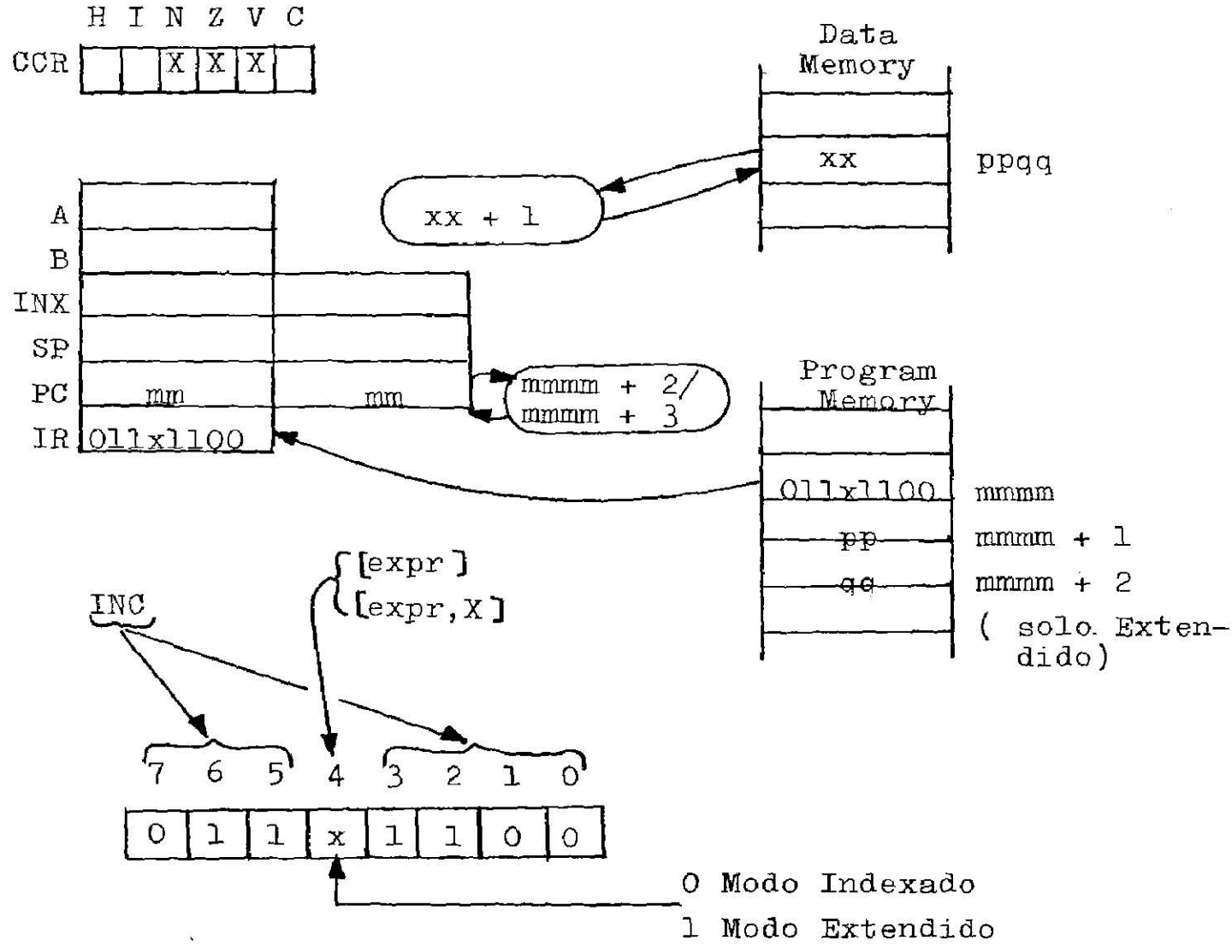
Suma 1 al acumulador seleccionado. Suponer xx=$3A_{16}$. Después - que la instrucción:

INC A

es ejecutada, el acumulador A contiene $3B_{16}$.



La instrucción INC también ofrece 2 modos de direccionamiento Extendido e Indexado.

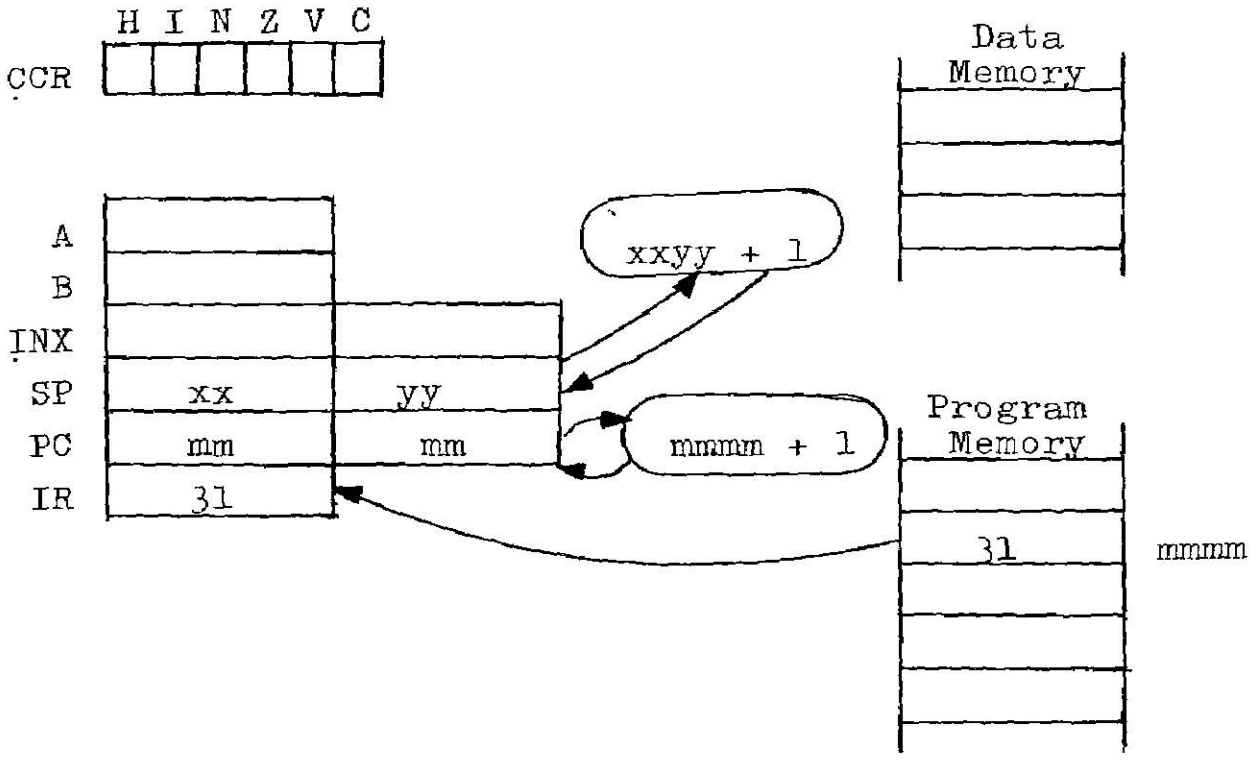


Incrementa el byte de memoria seleccionada. Suponer $pp=01_{16}$, $qq=A2_{16}$ y $xx=C0_{16}$. Después que se ejecuta la instrucción:

INC $01A2

el contenido de la localidad $01A2_{16}$ se incrementa a $C1_{16}$.
La instrucción INC puede ser usada para proveer un contador en una variedad de aplicaciones, esto es, contando las ocurrencias de un evento o como un contador iterativo el cual especifica el número de veces que un trabajo va a ser ejecutado.

INS - Incrementa el Apuntador de Pila



INS - 31

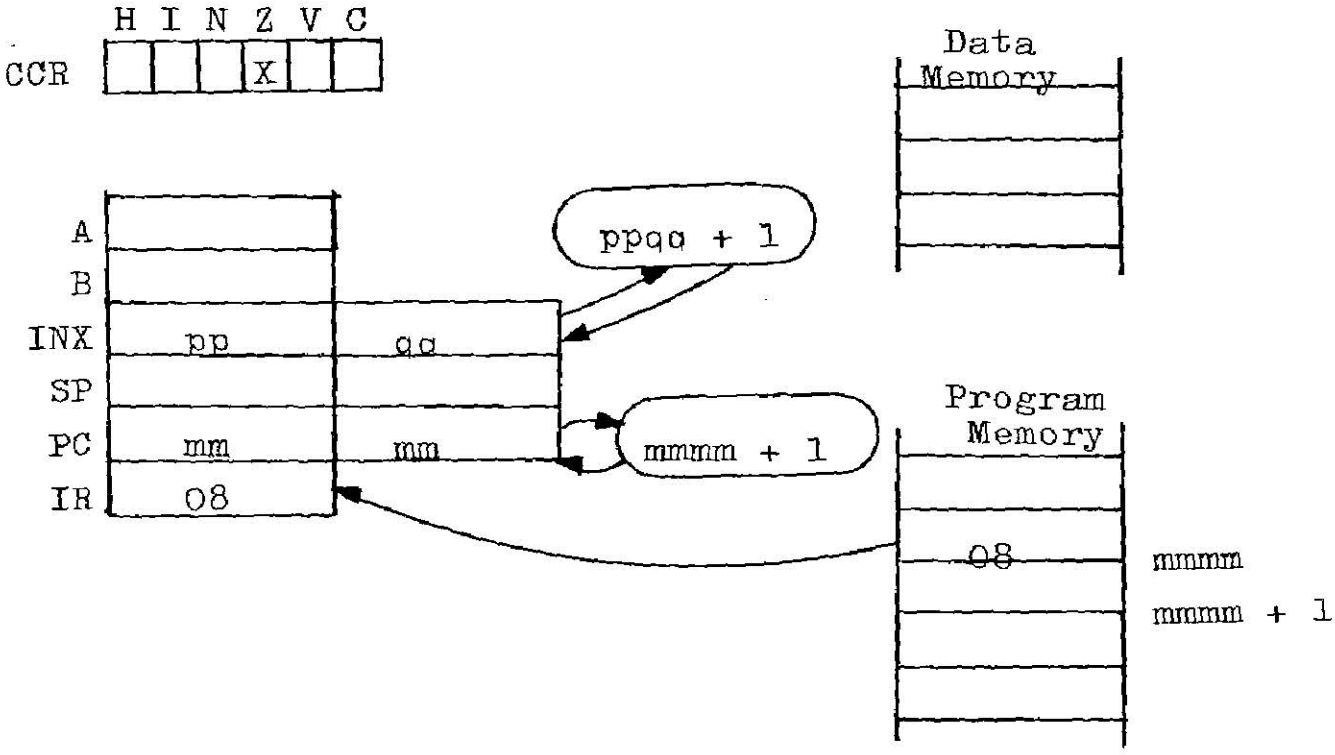
Suma 1 al valor de 16 bits del Apuntador de Pila. Ningún otro estado o registro es afectado.

Suponer que el Apuntador de Pila contiene $2F7A_{16}$. Después que la instrucción:

INS

es ejecutada, el Apuntador de Pila contiene $2F7B_{16}$.

INX - Incrementa el Registro de Indice



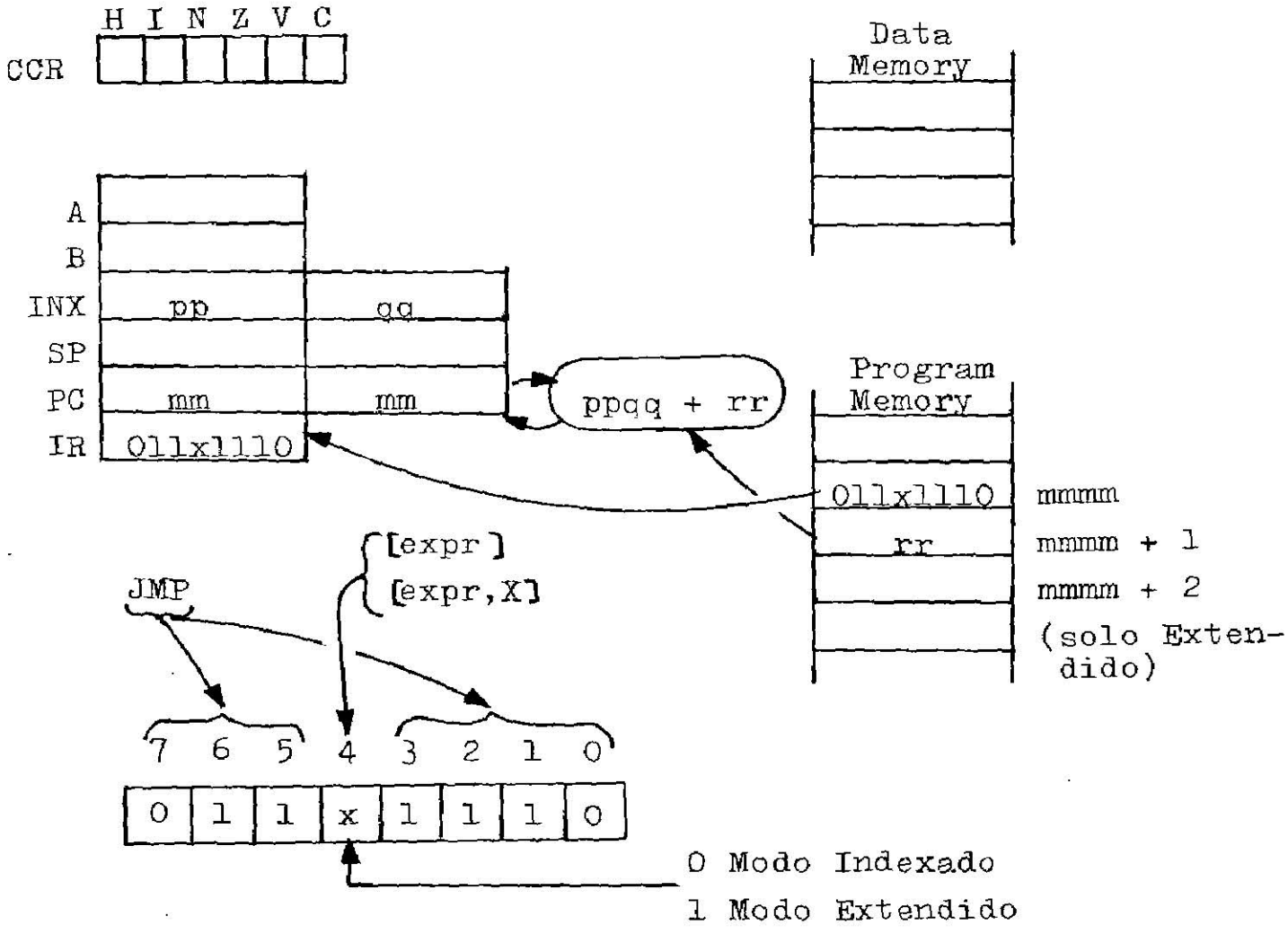
INX - 08

Suma 1 al valor de 16 bits del Registro de Indice. El estado Zero es 1 si el resultado de los 16 bits es 0. Suponer - que el Registro de Indice contiene $310C_{16}$. Después que la instrucción:

INX

es ejecutada, el Registro de Indice contiene $310D_{16}$.

JMP - Salto Vía con Direccionamiento Indexado o Extendido



Salta a la instrucción especificada por el operando cargando la dirección del byte de memoria en el Contador del Programa. En la siguiente secuencia de instrucción:

```
        LDX   #JPTBL
        JMP   2,X
        ---
        ---
JPTBL   BRA   NEWDATA
        BRA   PROCESSDATA
        BRA   FLAGDATA
```

Si el registro de índice contiene la dirección del JPTBL, entonces la instrucción ejecutada seguida de:

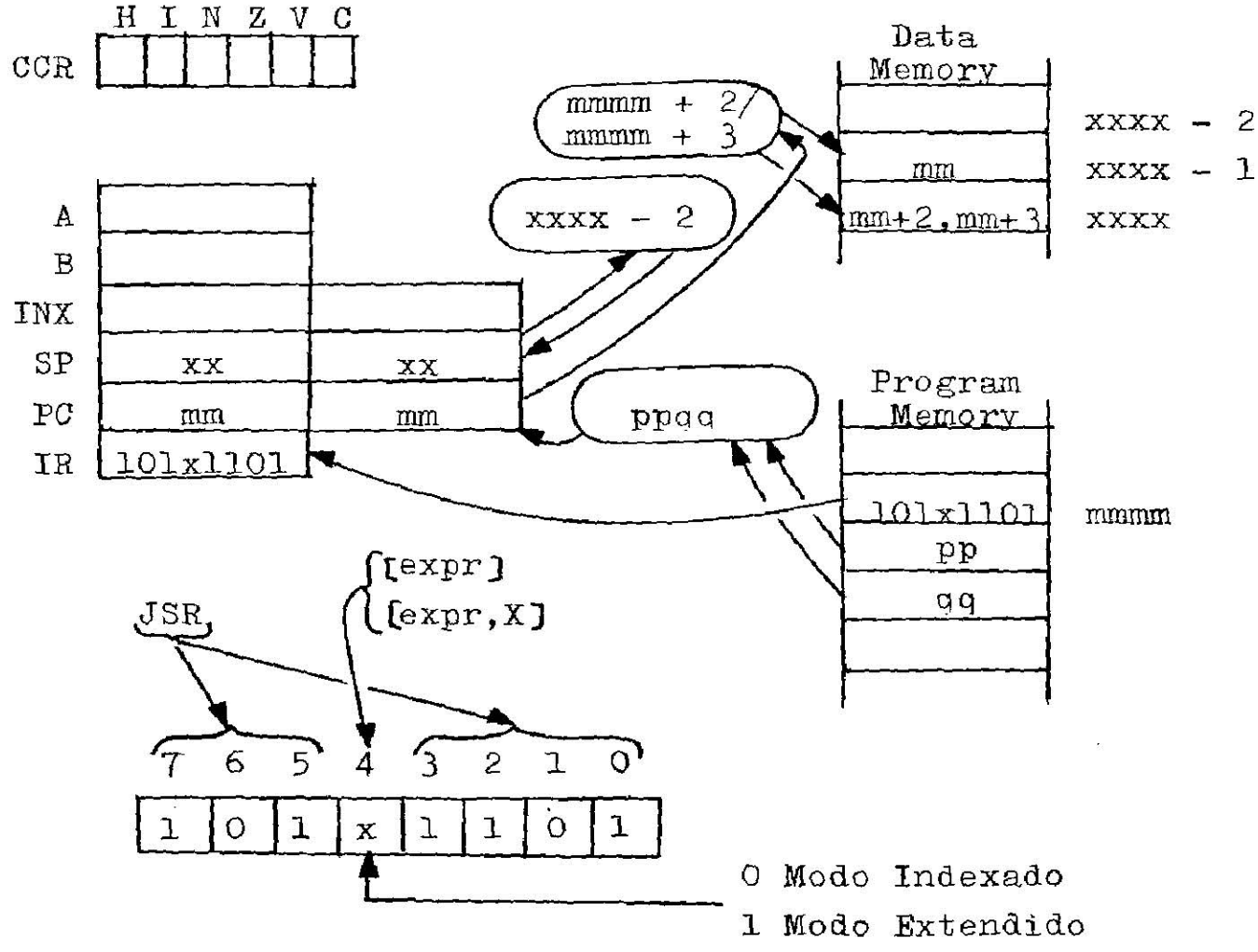
JMP 2,X

es la instrucción BRA PROCESSDATA.

Frecuentemente, la instrucción JMP usa el modo de direccionamiento Extendido. En ese caso, el segundo byte de la instrucción es cargado al byte alto del Contador del Programa, y el tercer byte cargado al byte bajo del Contador del Programa. La ejecución de la instrucción continúa desde esa dirección.

JSR - Salto a Subrutina usando Direccionamiento Indexado o Extendido

Esta instrucción puede ser ilustrada usando direcciona--- miento Extendido.



El Contador del Programa es incrementado en 3(si el direccio namiento extendido es usado), o 2(si el direccionamiento in-- dexado es usado), y entonces guardado en la pila. El Apunta-- dor de Pila es ajustado para apuntar a la próxima localidad - vacía en la pila. La dirección del byte de memoria selecciona do es entonces almacenado en el Contador del Programa. La ex- clusión continúa desde ese punto.

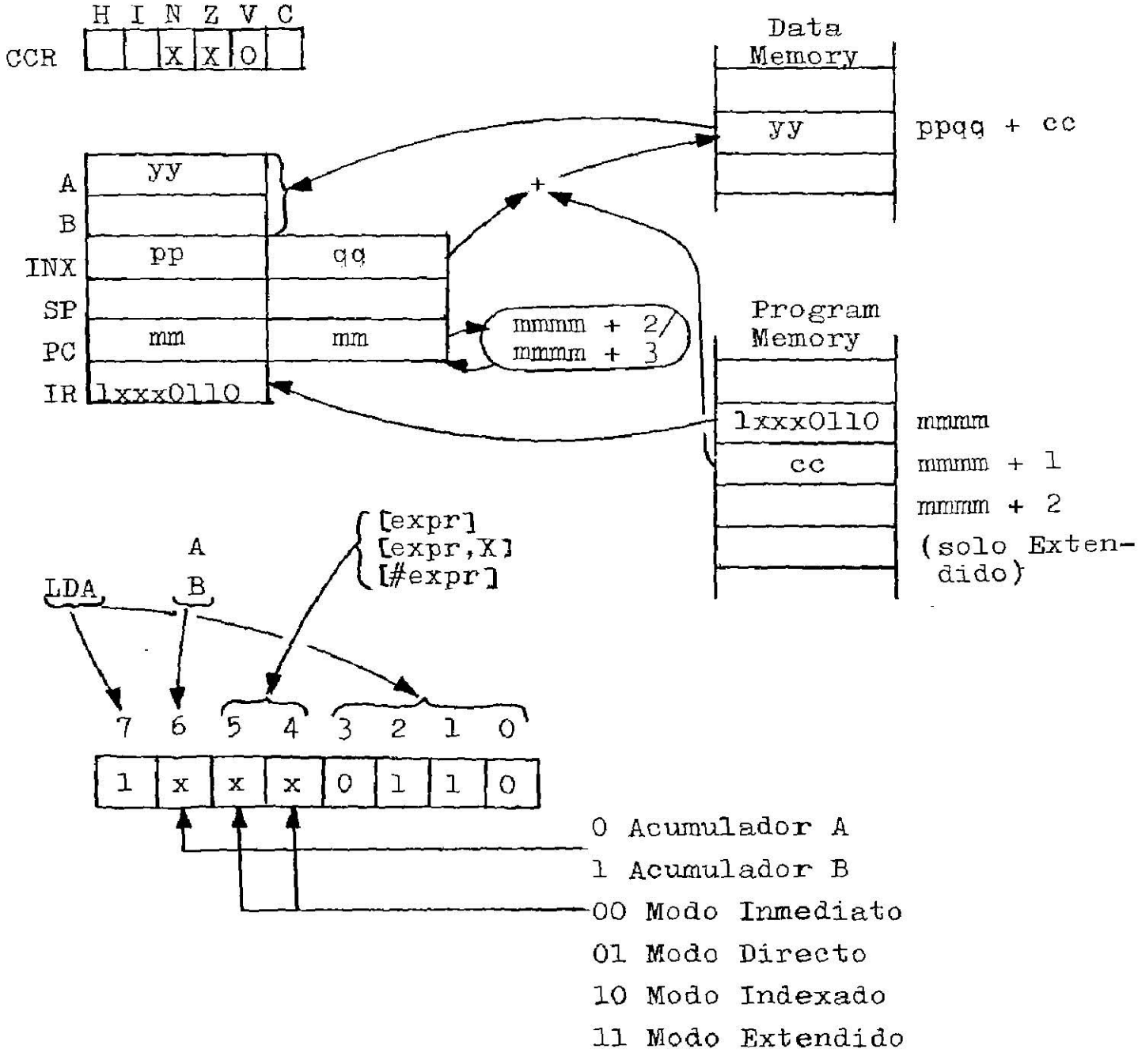
Consideremos la siguiente secuencia de instrucción:

```
        JSR   SUBR
        AND   #$7F
        ---
        ---
SUBR    ABA
```

Después que la instrucción JSR es ejecutada, la dirección de-la instrucción AND es salvada en el tope de la pila. La ins-- trucción ABA es la próxima instrucción a ser ejecutada.

LDA - Cargar el Acumulador desde Memoria

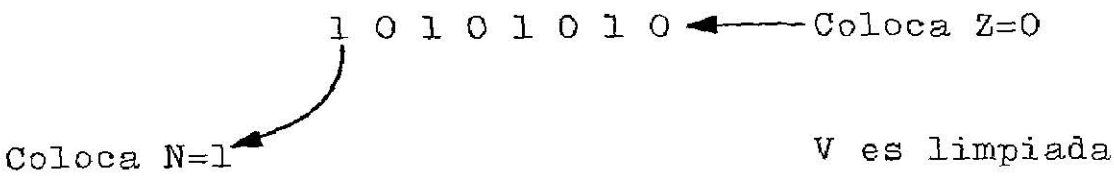
Carga el contenido del byte de memoria seleccionado en el acumulador específico. Esta instrucción ofrece la misma opción de direccionamiento de memoria que la instrucción ADC y puede ser ilustrada usando el modo Indexado.



Carga el contenido del byte de memoria seleccionado al acumulador específico. Suponer que el registro de índice contiene- 0800_{16} y cc=43_{16}. Si la localidad de memoria 0843_{16} contiene- AA_{16} . Después que la instrucción:

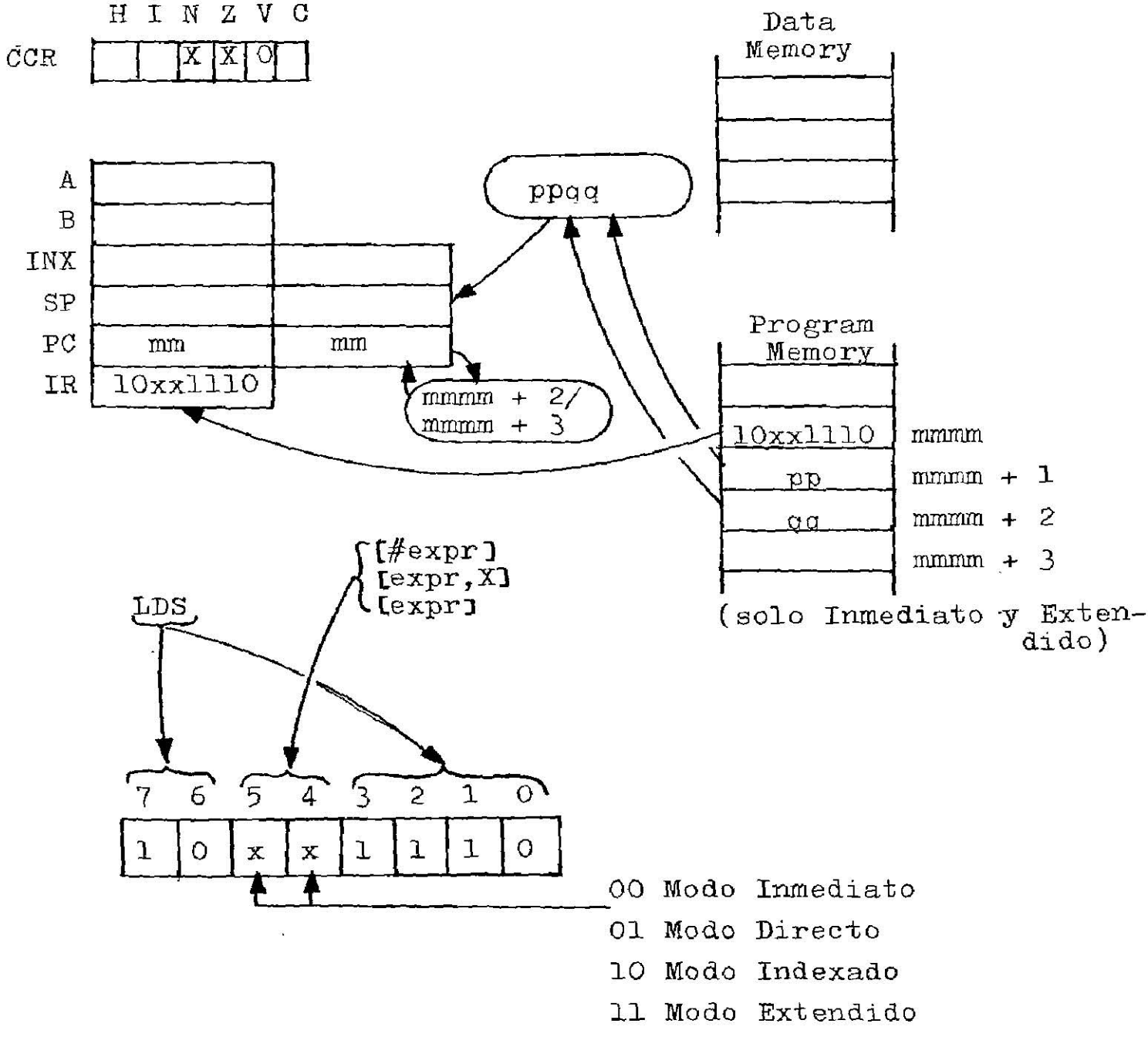
```
LDA A  $43,X
```

es ejecutada, el acumulador A contiene AA_{16}.



LDS - Carga el Apuntador de Pila

Carga el contenido de 2 localidades de memoria seleccionada en el Apuntador de Pila. Esta instrucción ofrece la misma opción de direccionamiento de memoria que la instrucción ADC, y puede ser ilustrada usando el direccionamiento Inmediato.

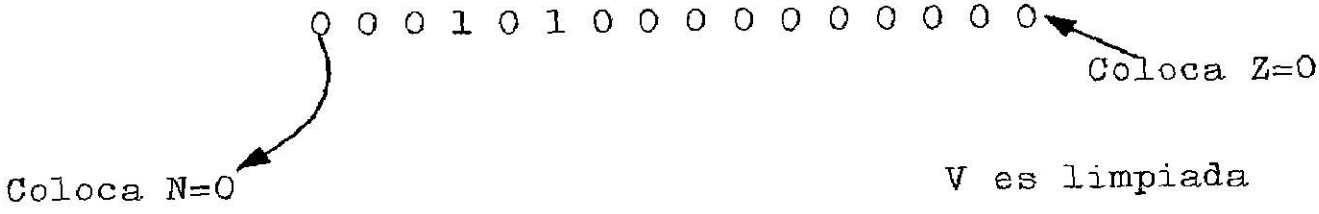


Carga el contenido del byte de memoria seleccionada en el --- byte alto del Apuntador de Pila. Carga el contenido del byte- de memoria inmediatamente seguido del byte de memoria en el - byte bajo del Apuntador de Pila. Coloca el estado Sign si el- byte más significativo es colocado; coloca el estado Zero si- todos los 16 bits cargados son 0. Limpia la bandera Overflow.

Suponer pp=14_{16} y qq=00_{16}. Después que la instrucción:

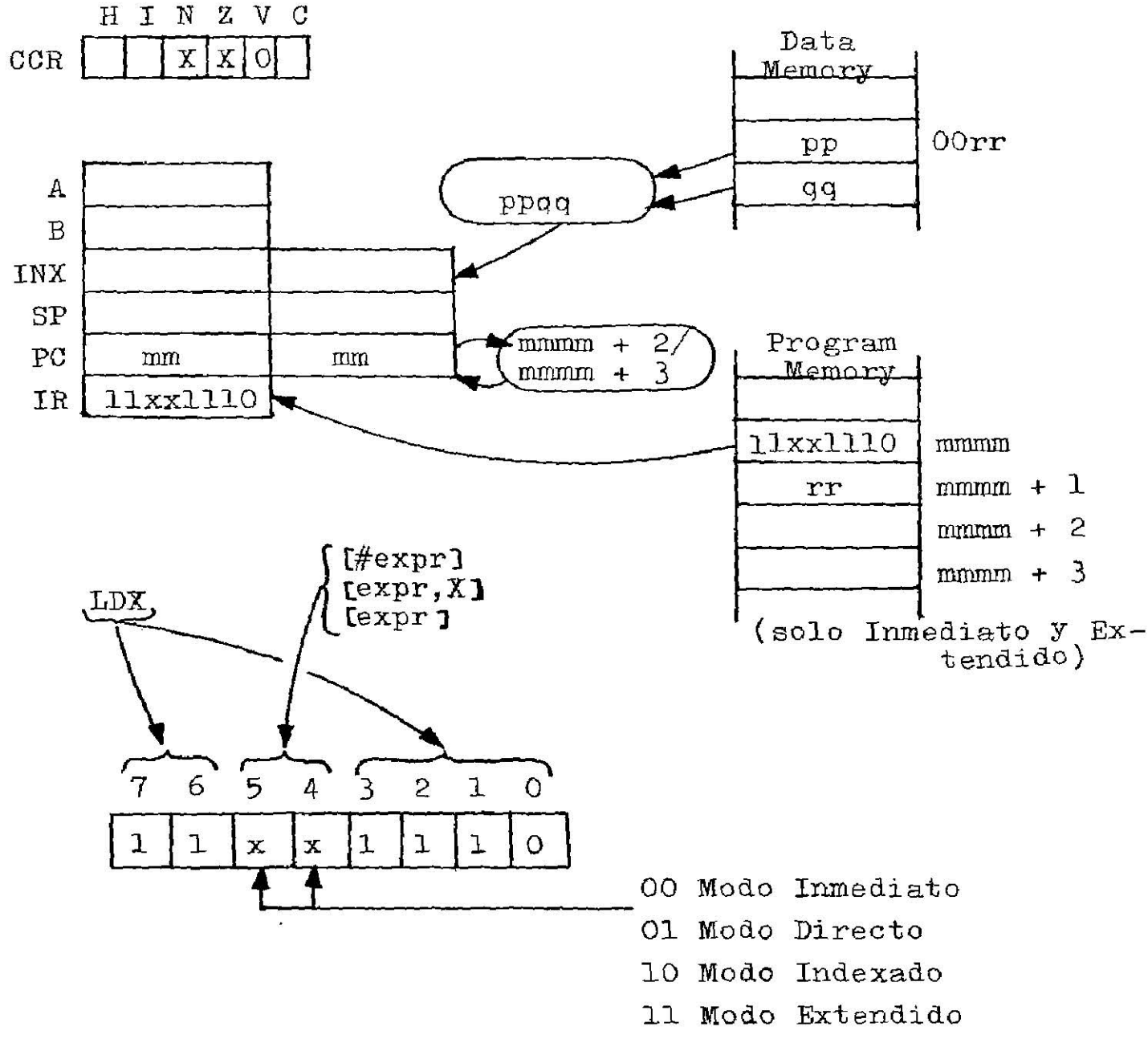
```
LDS  #$1400
```

es ejecutado, el Apuntador de Pila contiene 1400_{16}.



LDX - Carga el Registro de Indice

Carga el contenido de 2 localidades de memoria en el Re-- gistro de Indice. Esta instrucción ofrece la misma opción de- direccionamiento de memoria que la instrucción ADC, y puede - ser ilustrada usando el modo Directo.

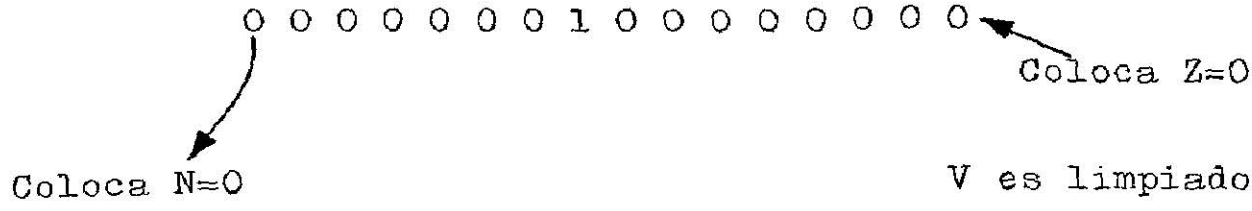


Carga el contenido del byte de memoria seleccionado en el --- byte alto del Registro de Indice. Carga el contenido del byte de memoria inmediatamente seguido del byte de memoria selec-- cionado en el byte bajo del Registro de Indice. Coloca el es- tado Sign si el bit más significativo del Registro de Indice- es 1. Coloca el estado Zero si todos los 16 bits del Registro de Indice son 0. El estado Overflow es 0.

Suponer que $rr=90_{16}$, $pp=01_{16}$ y $qq=00_{16}$. Ejecutando:

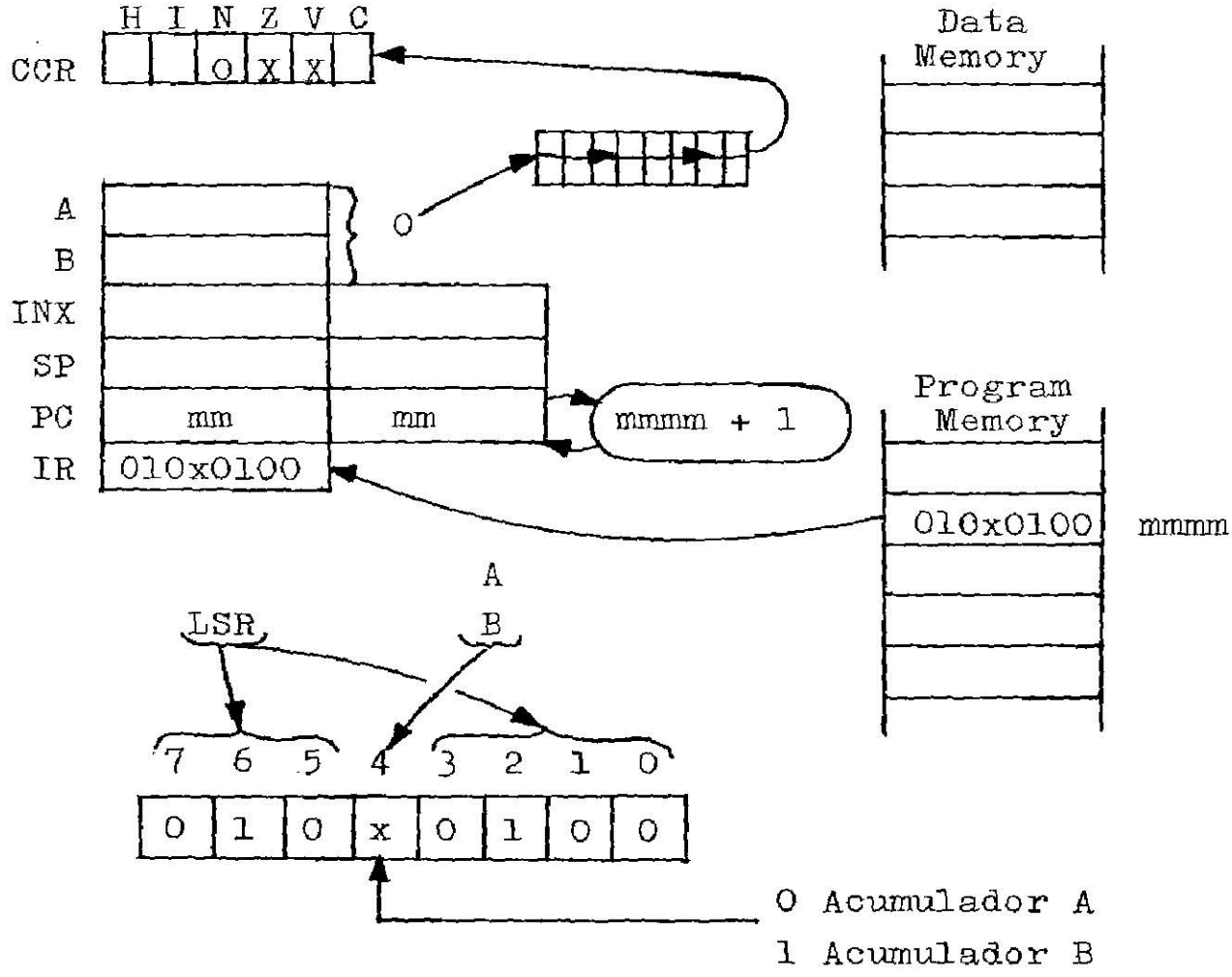
```
LDX   $90
```

carga 0100_{16} en el Registro de Indice.



LSR - Corrimiento Lógico a la Derecha del Acumulador o de Memoria

Esta instrucción ejecuta un bit lógico corriéndolo a la - derecha de un acumulador específico o un byte de memoria se-- leccionado.

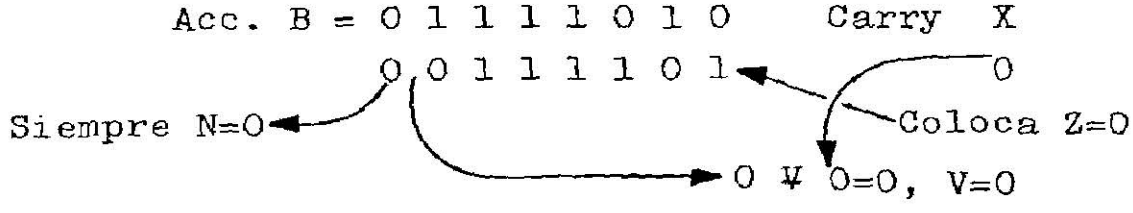


Cambia el contenido del acumulador seleccionado un bit a la - derecha. Cambia el bit de bajo orden al estado Carry. Cambia- con 0 el bit de alto orden.

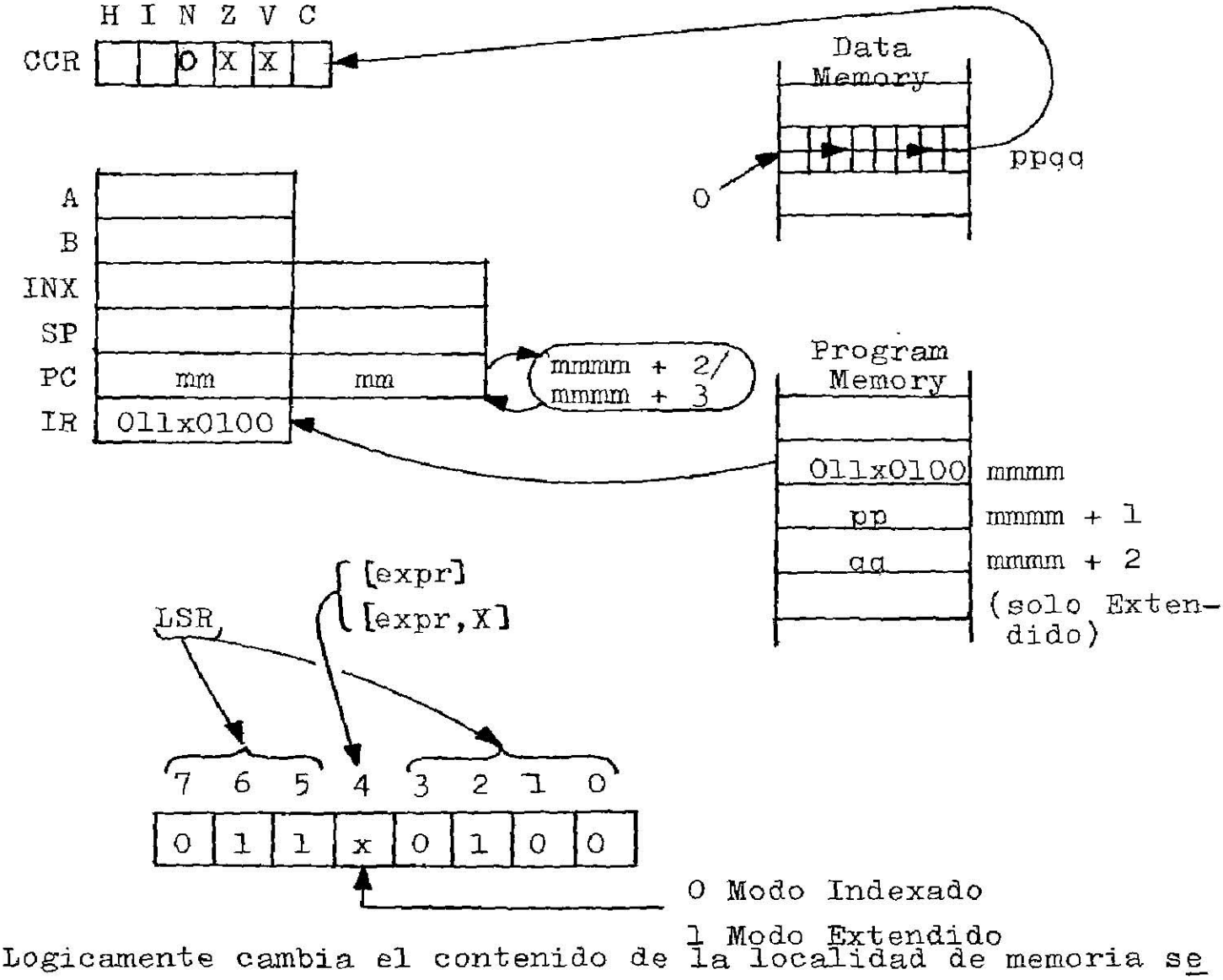
Suponer Acc. B= 7A . Después que la instrucción:

LSR B

es ejecutada, el Acumulador B contiene 3D y el estado Carry es 0.



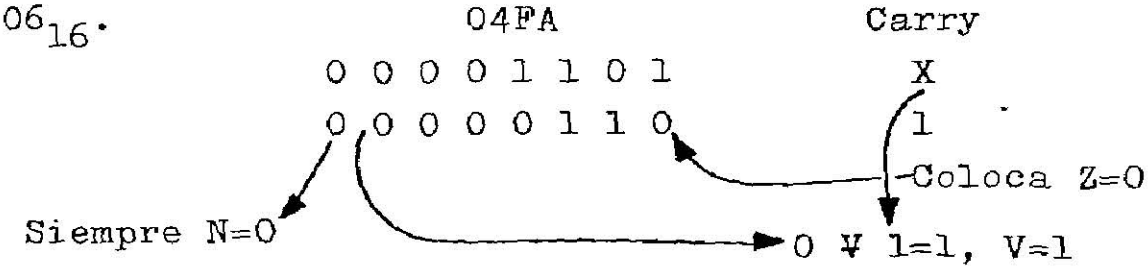
La instrucción LSR también tiene 2 modos de direccionamiento- Indexado y Extendido.



Logicamente cambia el contenido de la localidad de memoria seleccionada un bit a la derecha. Suponer pp=04_{16}, qq=FA_{16} y el contenido de la localidad de memoria $04FA_{16}$ es $0D_{16}$. Después- que la instrucción:

LSR $04FA

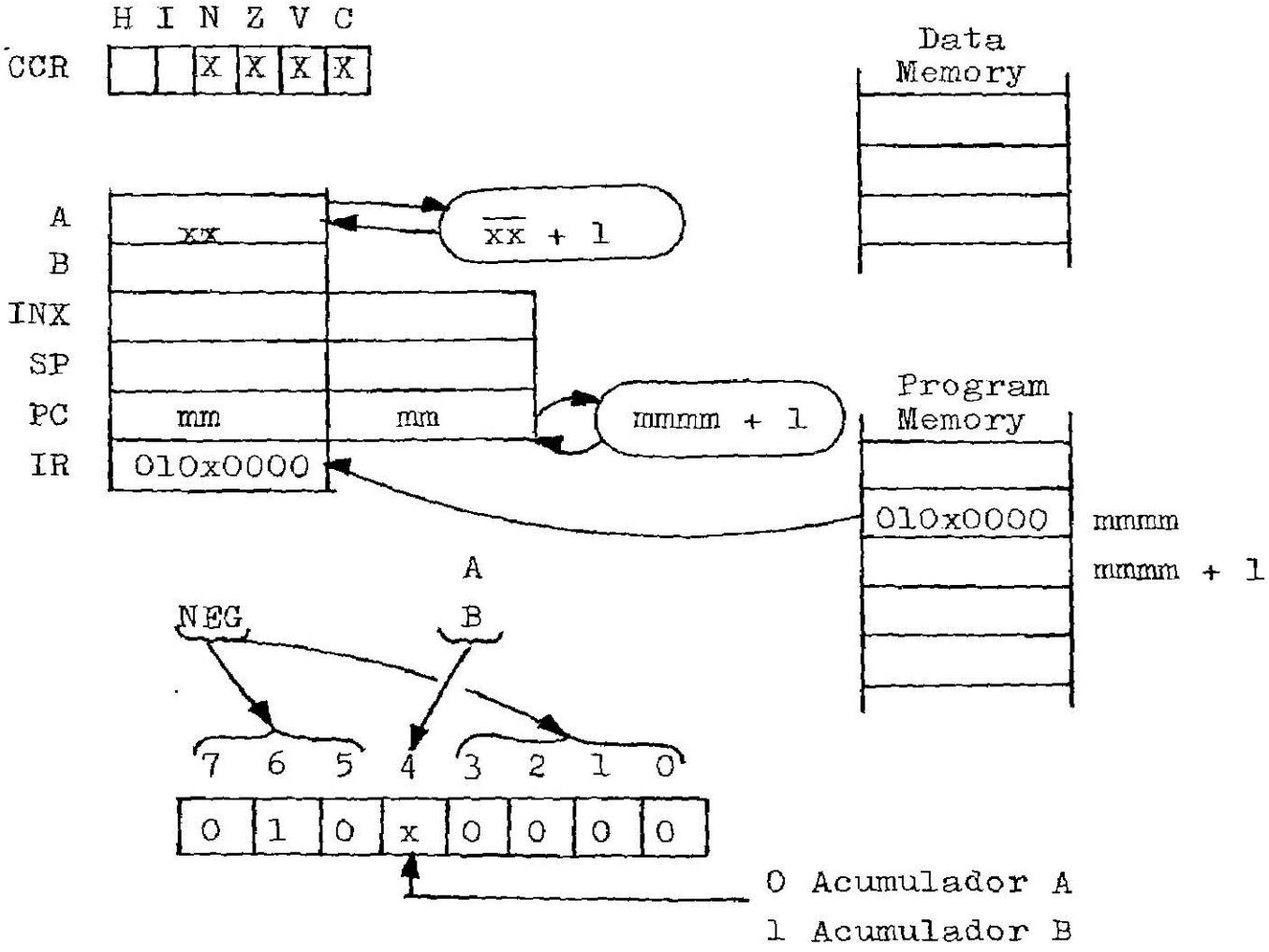
es ejecutada, el estado Carry es 1 y el contenido de $04FA_{16}$ es 06_{16}.



NEG - Negativo del Acumulador o de Memoria

Esta instrucción niega el contenido de un acumulador específico o un byte de memoria seleccionado remplazando el acumulador o el byte de memoria con el complemento a 2 de estos -- contenidos.

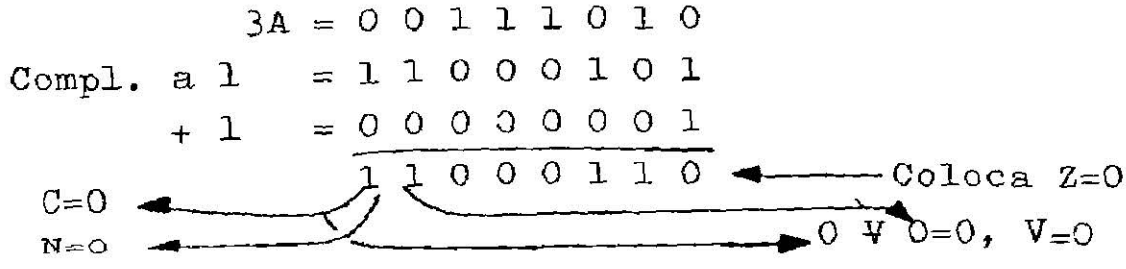
Primeros, consideremos la negación del Acumulador.



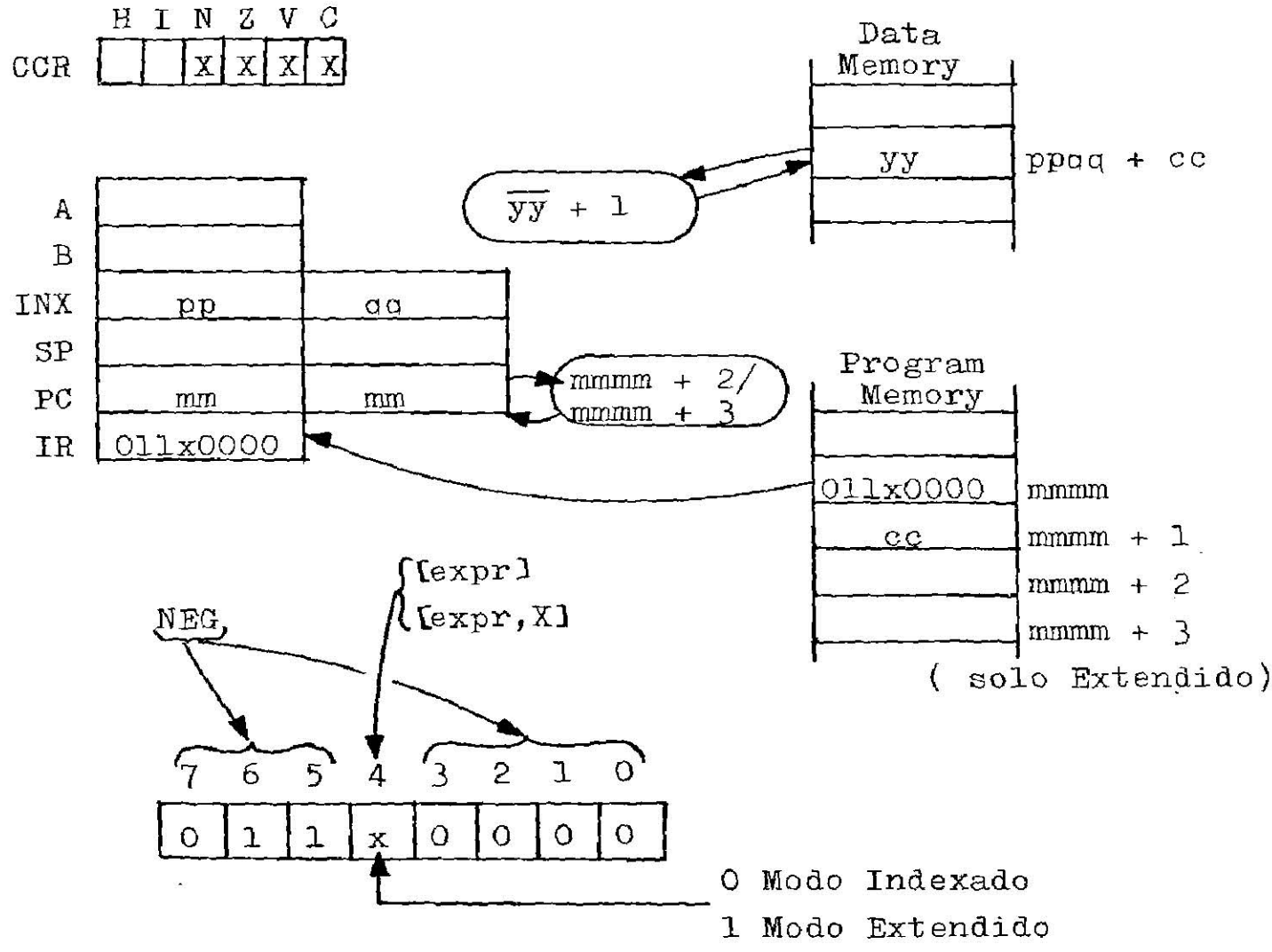
Niega el contenido del acumulador específico tomando el complemento a 1 del contenido del acumulador y sumandole 1 al resultado. Suponer Acc. A=$3A_{16}$. Después que la instrucción:

NEG A

es ejecutado, el Acumulador A contiene $C6_{16}$.



La instrucción NEG también tiene 2 modos de direccionamiento- Indexado y Extendido.



Suponer el contenido del Registro de Indice como 0100_{16}, cc=$1D_{16}$ y la localidad $011D_{16}$ como AA_{16}. Después que la instrucción:

NEG $1D,X

es ejecutada, la localidad $011D_{16}$ es alterada a 56_{16}.

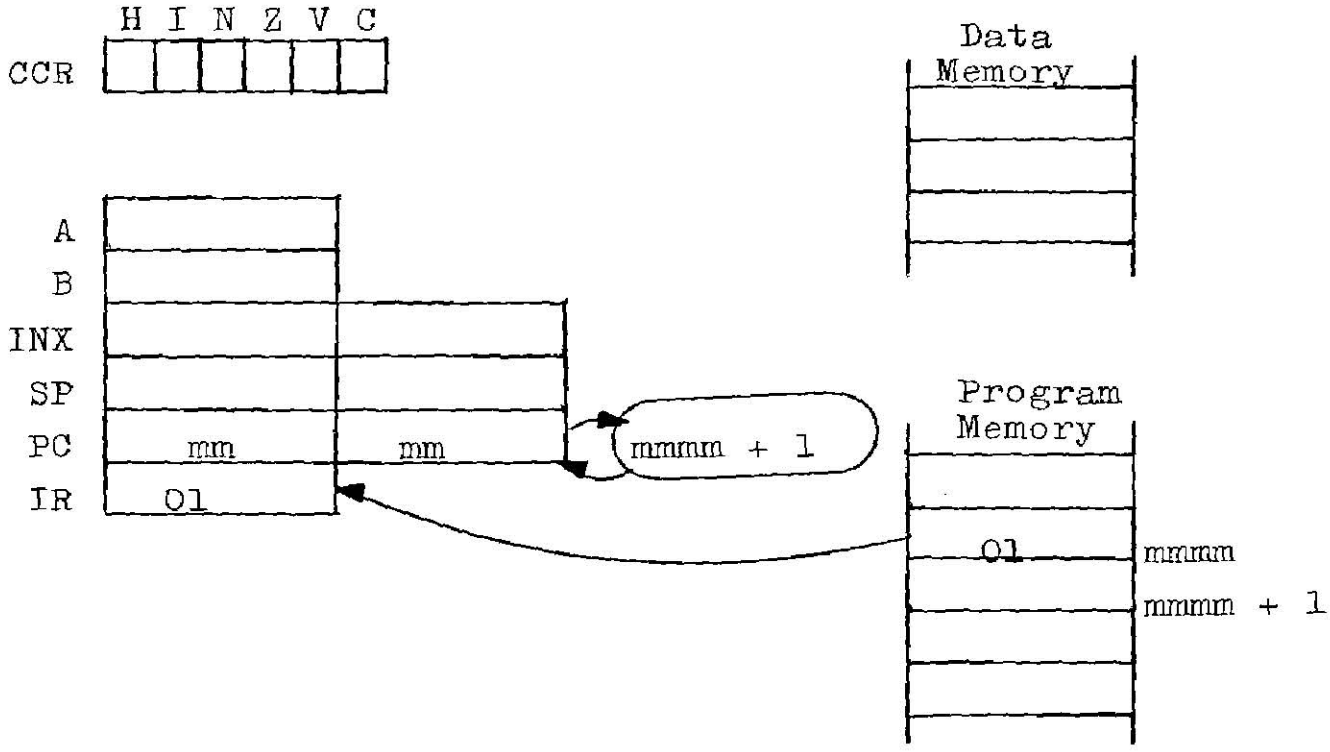
```
        AA = 1 0 1 0 1 0 1 0
Compl. a 1 = 0 1 0 1 0 1 0 1
     + 1   = 0 0 0 0 0 0 0 1
             ---------------
             0 1 0 1 0 1 1 0  <-- Coloca Z=0
```

Coloca C=0

0 ⊻ 0=0, V=0

N=0

NOP - No Operación



NOP - 01

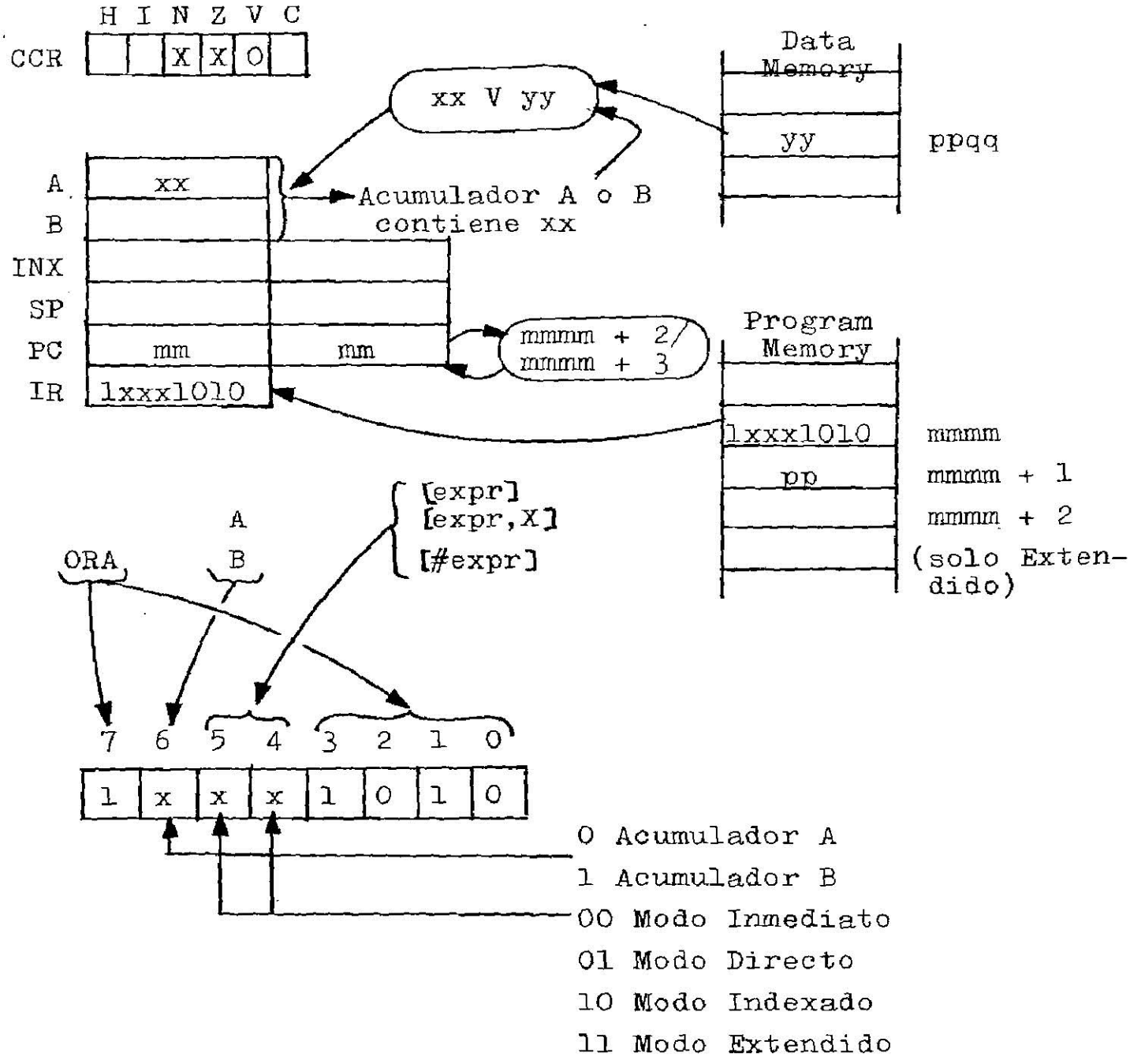
Esta es una instrucción de 1 byte la cual ejecuta una no-operación excepto que el Contador del Programa es incrementado. Esta instrucción está presente por 2 razones:

- Permite dar una etiqueta a un byte de programa objeto:
  HERE NOP
- Para afinar el tiempo de demora. Cada instrucción NOP suma-2 ciclos para una demora.

NOP no es muy frecuentemente usada.

ORA - Aplicación del OR del Acumulador con Memoria

Esta instrucción aplica el OR al contenido del acumulador A o B con el contenido de un byte de memoria seleccionada. Esta instrucción ofrece la misma opción de direccionamiento de-memoria que la instrucción ADC, y puede ilustrarse usando el-modo Extendido.



Aplica el OR al contenido del acumulador específico con el -- contenido de la localidad de memoria seleccionada, tratando - ambos operandos como simples datos binarios.

Suponer que pp=16_{16}, qq=23_{16}, xx=$E3_{16}$ y yy=AB_{16}. Después que- la instrucción:

```
ORA A  $1623
```

es ejecutada, el Acumulador A contiene EB_{16}.

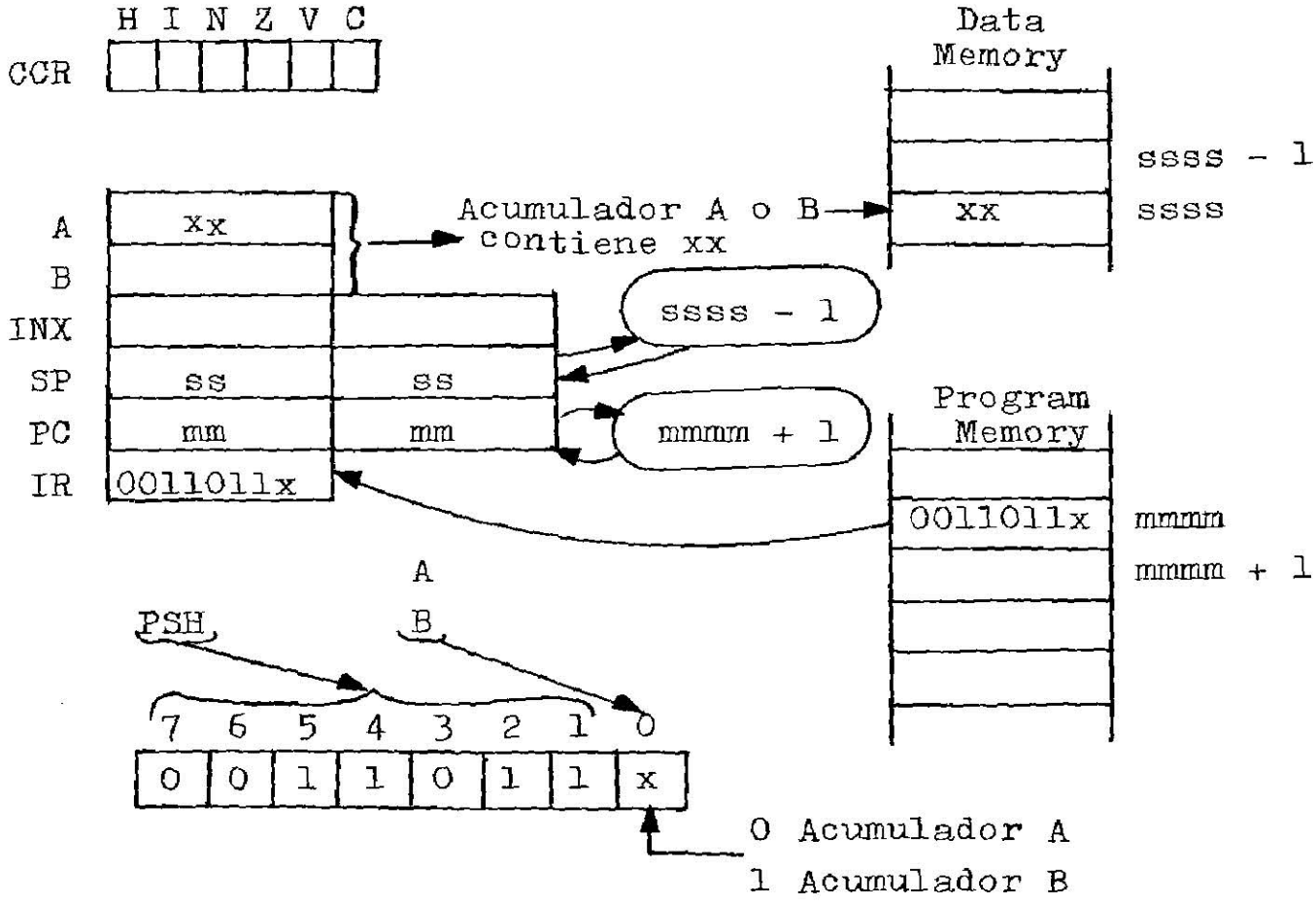
```
E3 = 1 1 1 0 0 0 1 1
AB = 1 0 1 0 1 0 1 1
     1 1 1 0 1 0 1 1 <---Coloca Z=0

Coloca N=1 <---                Limpia el V
```

Esta es una instrucción lógica; es seguido usada para "pren - der" los bits en turno. Por ejemplo, la instrucción:

```
ORA A  #$80
```

coloca incondicionalmente el bit de alto orden del Acumulador A en 1.

PSH - Coloca el Acumulador en la Pila



Coloca el contenido del Acumulador seleccionado en el tope de la pila. El Apuntador de Pila es decrementado en 1. Ningún otro estado o registro es afectado.

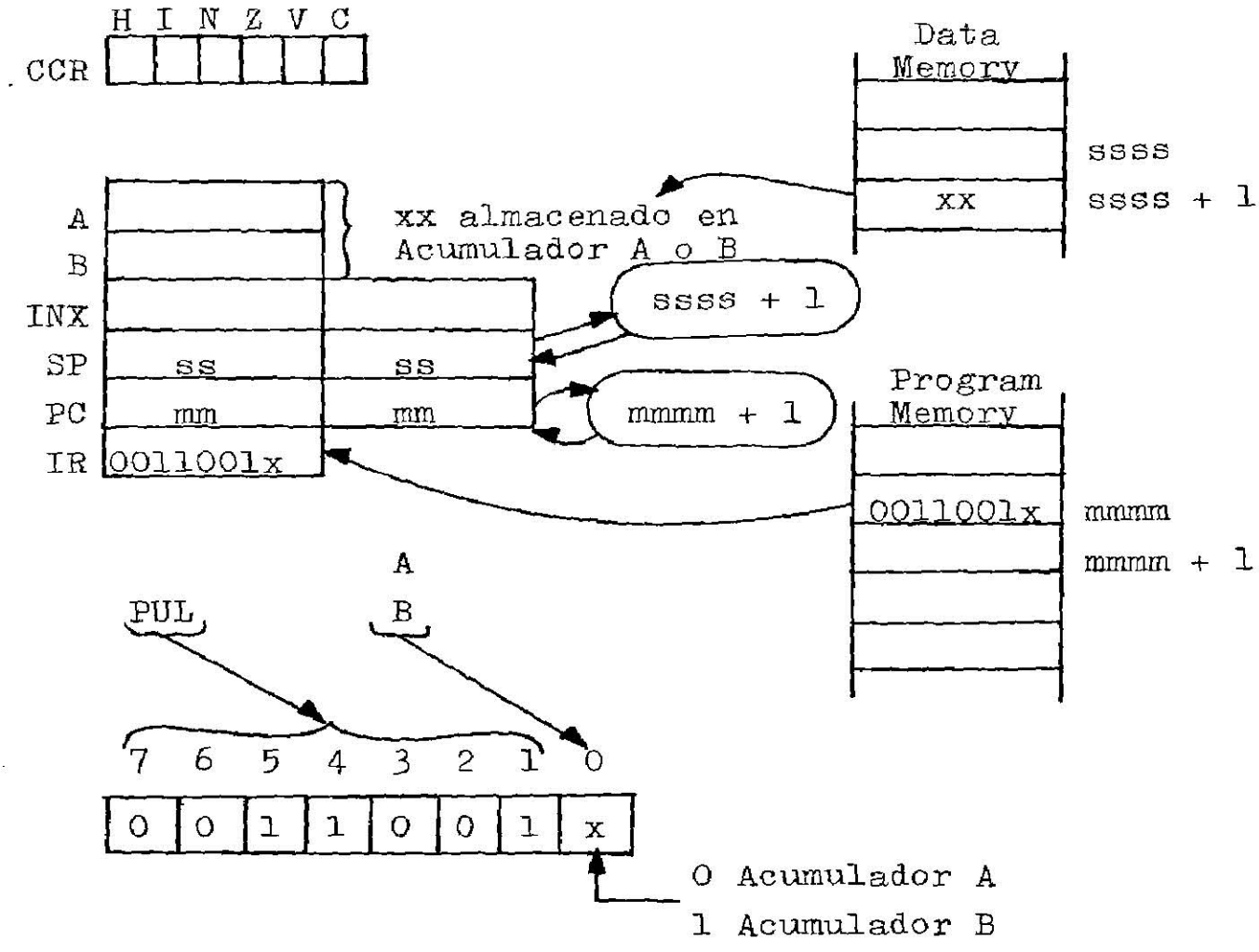
Suponer que el Acumulador A contiene $3A_{16}$ y el Apuntador de Pila contiene $2AF7_{16}$. Después que la instrucción:

PSH A

es ejecutada, 3A es almacenado en la localidad $2AF7_{16}$ y el Apuntador de Pila es alterado a $2AF6_{16}$.

La instrucción PSH es muy frecuentemente usada para salvar el contenido del acumulador, por ejemplo, antes de servir a una interrupción.

PUL - Saca datos de la Pila



Incrementa el Apuntador de Pila, entonces saca del tope de la pila un byte del acumulador seleccionado. Ningún otro estado o registro es afectado.

Suponer que el Apuntador de Pila contiene $2AF6_{16}$ y la localidad $2AF7_{16}$ contiene CE_{16}. Después que la instrucción:

PUL B

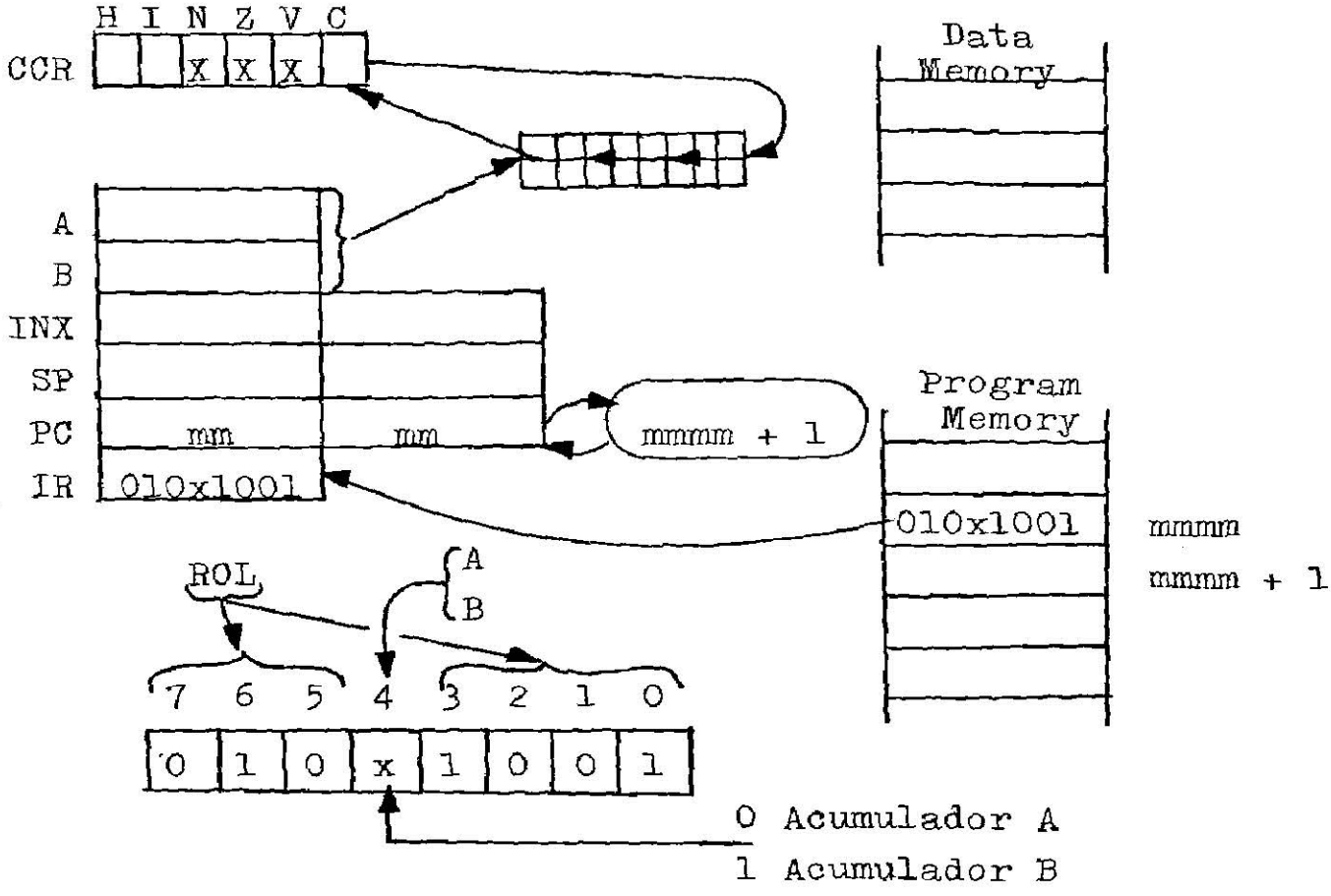
es ejecutada, el Acumulador B contiene CE_{16} y el Apuntador de Pila contiene 2AF7.

La instrucción PUL es muy frecuentemente usada para realmacenar el contenido del acumulador que es salvado en la pila, -- por ejemplo, después de servir un interruptor.

ROL - Rotación del Acumulador o Memoria a la Izquierda a través del Carry

Esta instrucción rota el acumulador específico o un byte de memoria seleccionado un bit a la izquierda a través del Carry.

Primero, consideremos rotar el Acumulador.

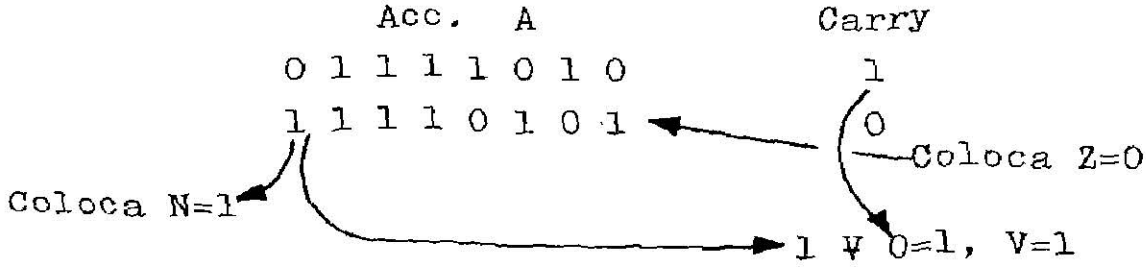


Rotación del contenido del Acumulador seleccionado un bit a la izquierda a través del estado Carry.

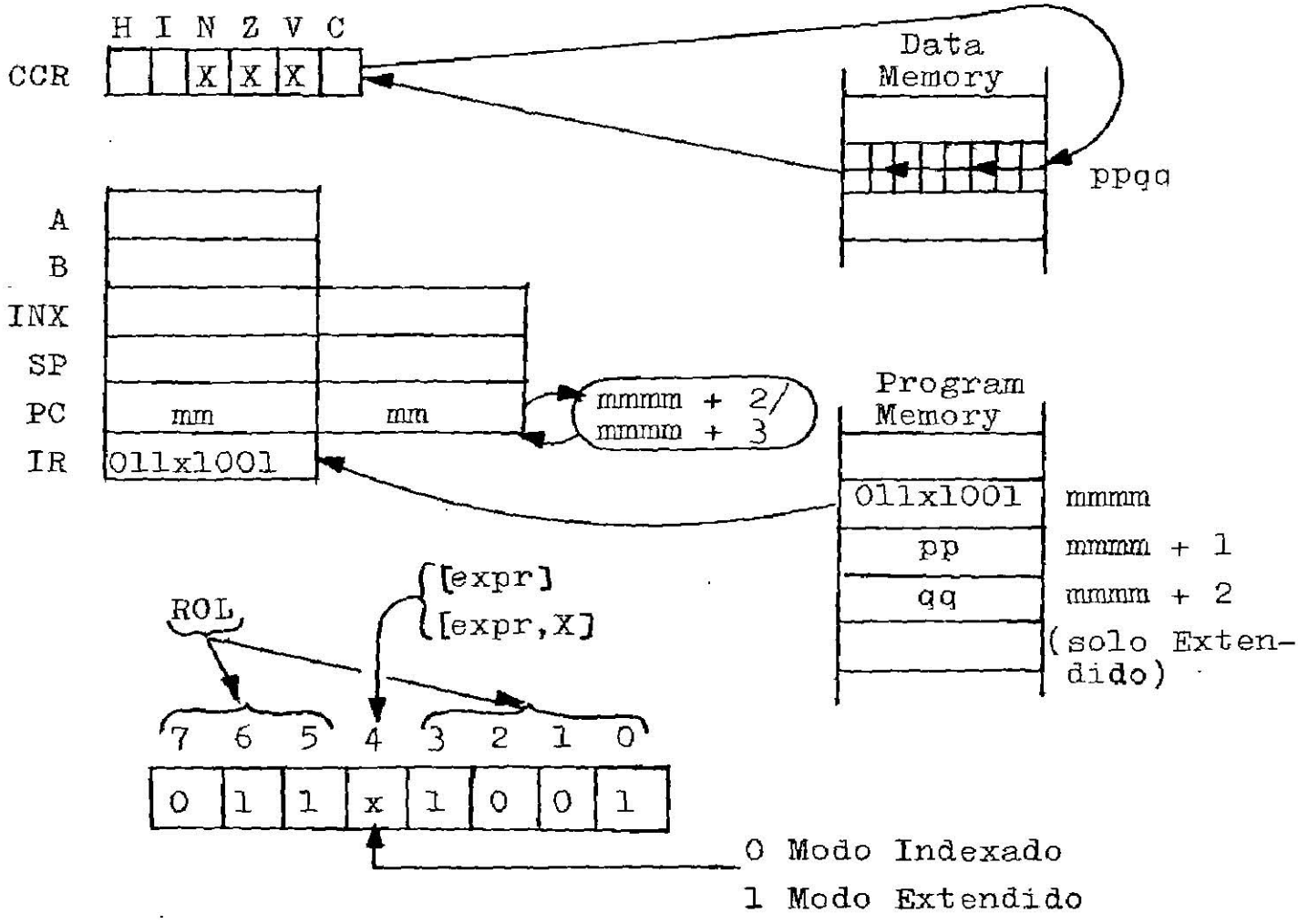
Suponer Acc. A=$7A_{16}$ y el Carry en 1. Después que la instrucción:

ROL A

es ejecutada, el acumulador A contiene $F5_{16}$ y el Carry un 0.



La instrucción ROL tiene 2 modos de direccionamiento, Extendido e Indexado.

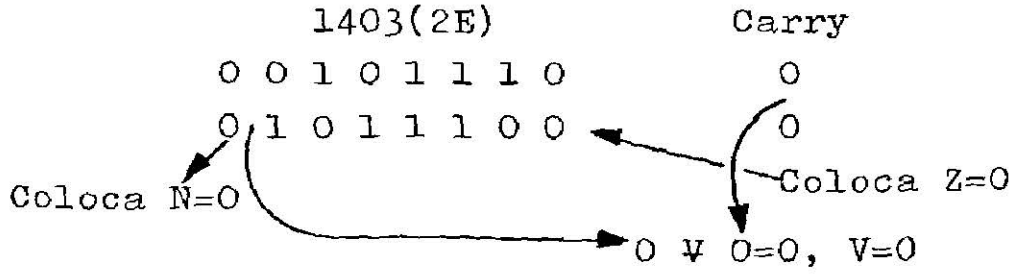


Rotación del byte de memoria seleccionado un bit a la izquierda a través del estado Carry.

Suponer pp=14_{16}, qq=03_{16}, el contenido de la localidad de memoria 1403_{16} como 2E y el Carry como 0. Después que la instrucción:

ROL $1403

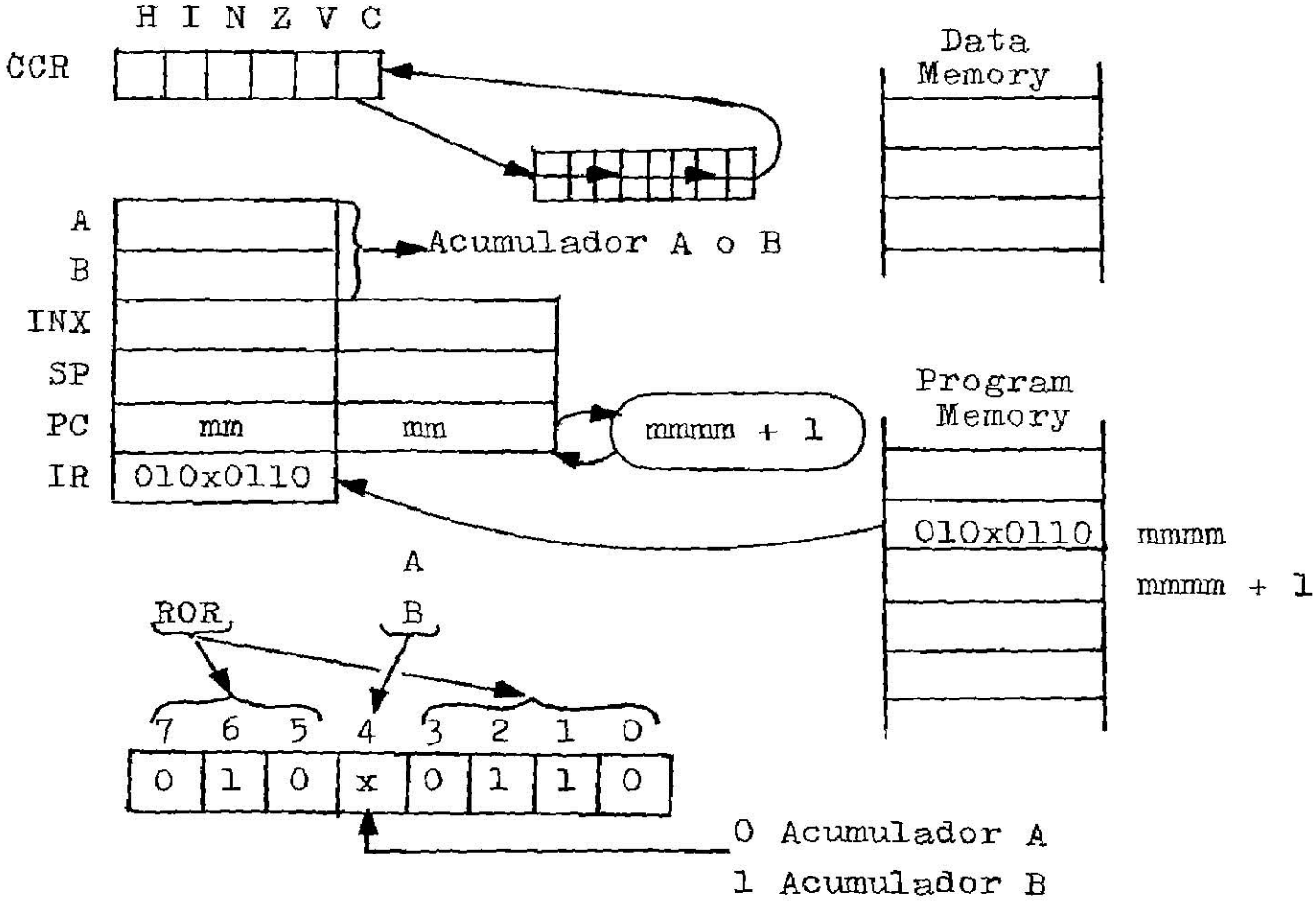
es ejecutado, la localidad 1403_{16} contiene $5C_{16}$.



ROR - Rotación del Acumulador o la Memoria a la Derecha
a través del Carry

Esta instrucción rota un acumulador específico o un byte de memoria seleccionado un bit a la derecha a través del estado Carry.

Primero, consideremos la rotación del Acumulador.

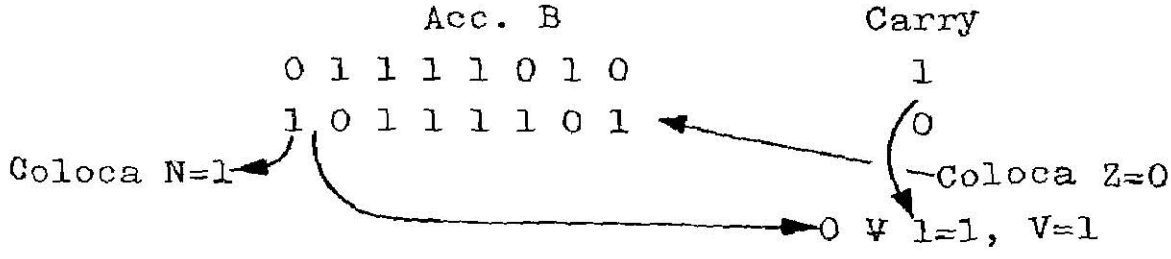


Rotación del contenido del acumulador específico un bit a la derecha a tavés del estado Carry.

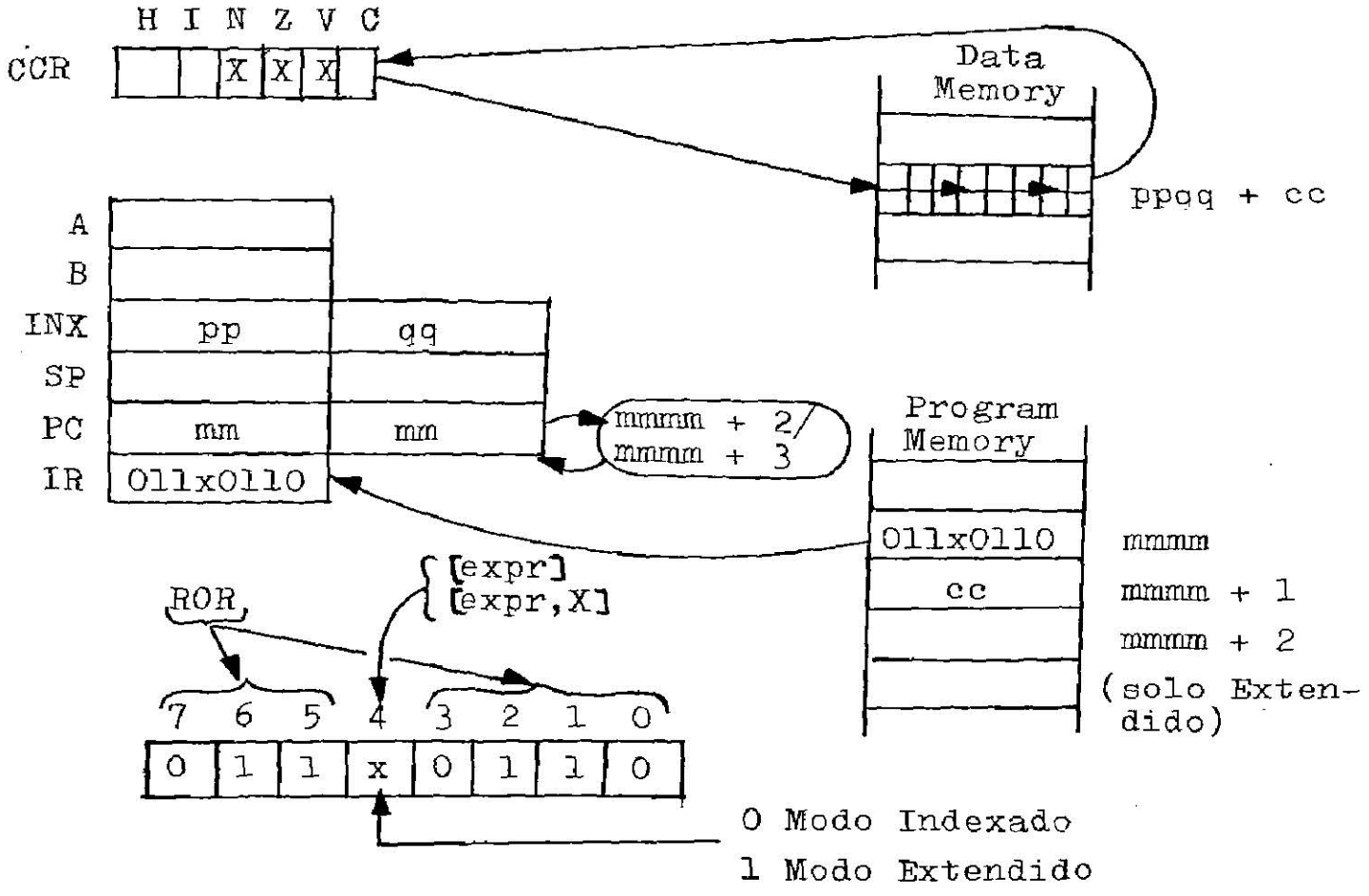
Suponer el Acc. B=$7A_{16}$ y el Carry = 1. Al ejecutar:

ROR B

puede producir estos resultados; Acc. B=BD_{16} y Carry =0.



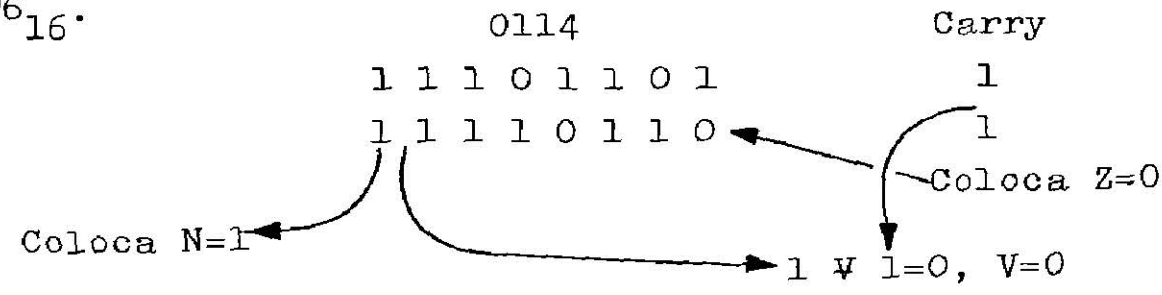
La instrucción ROR tiene 2 modos de direccionamiento, Indexado y Extendido.



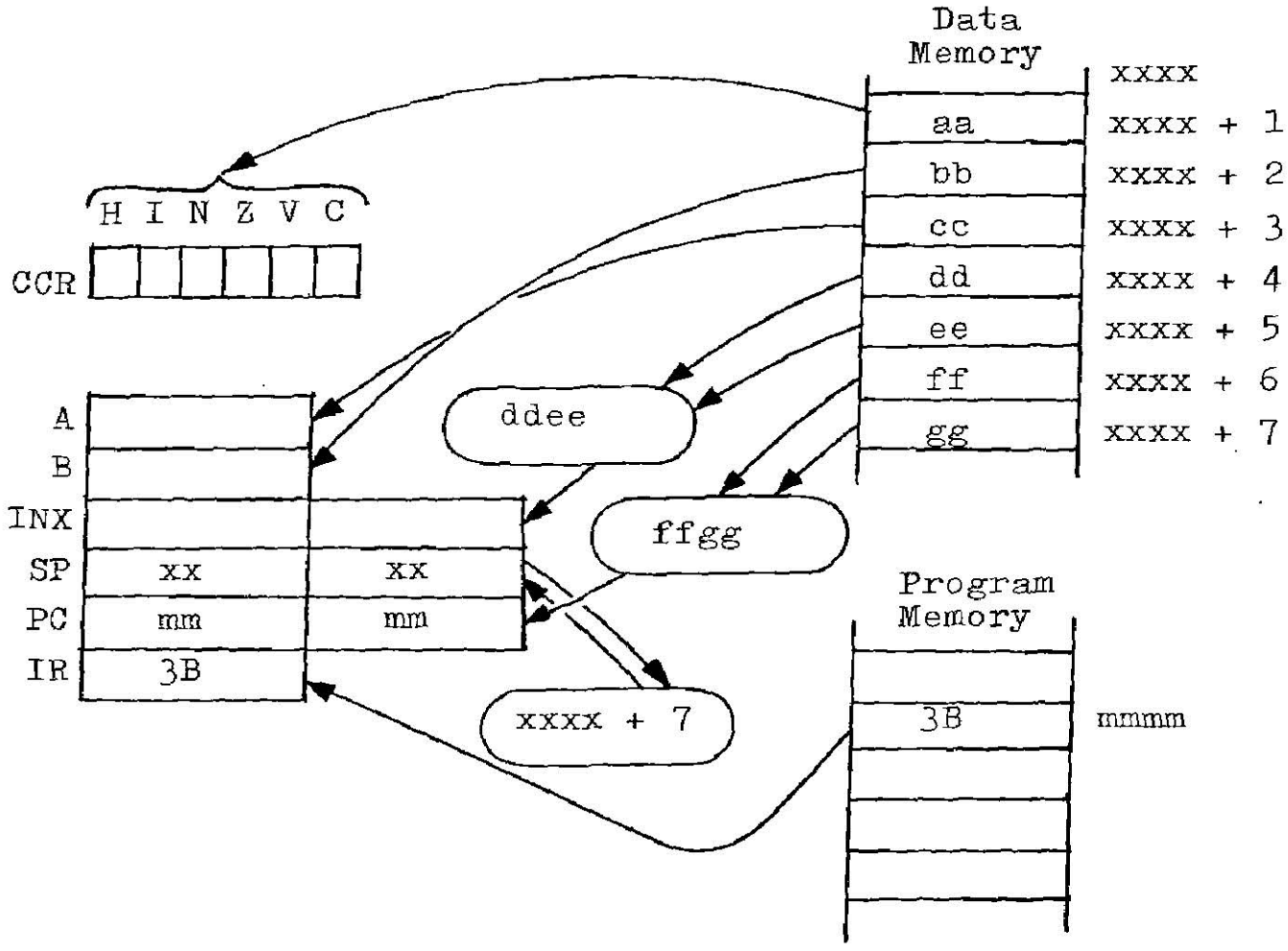
Suponer que cc=14_{16}, el contenido del Registro de Indice es 0100_{16}, el contenido de la localidad 0114_{16} es ED_{16} y el Carry es 1. Después que la instrucción:

```
ROR  $14,X
```

es ejecutada, el Carry contiene 1 y la localidad 0114_{16} es $F6_{16}$.



RTI - Retorno desde la Interrupción



RTI - 3B

El Registro de Código de Condiciones, los Acumuladores, - el Registro de Indice y el Contador del Programa tienen valores sacados de la pila. Los registros y las localidades co--- rrespondientes de la pila de la cual son sacados son como sigue:

Localidad de Memoria	Registro
XXXX+1 (Bit 5-0)	Registro de Código de Condiciones
XXXX+2	Acumulador B
XXXX+3	Acumulador A
XXXX+4	Byte alto del Registro de Indice
XXXX+5	Byte bajo del Registro de Indice
XXXX+6	Byte alto del Contador del Programa
XXXX+7	Byte bajo del Contador del Programa

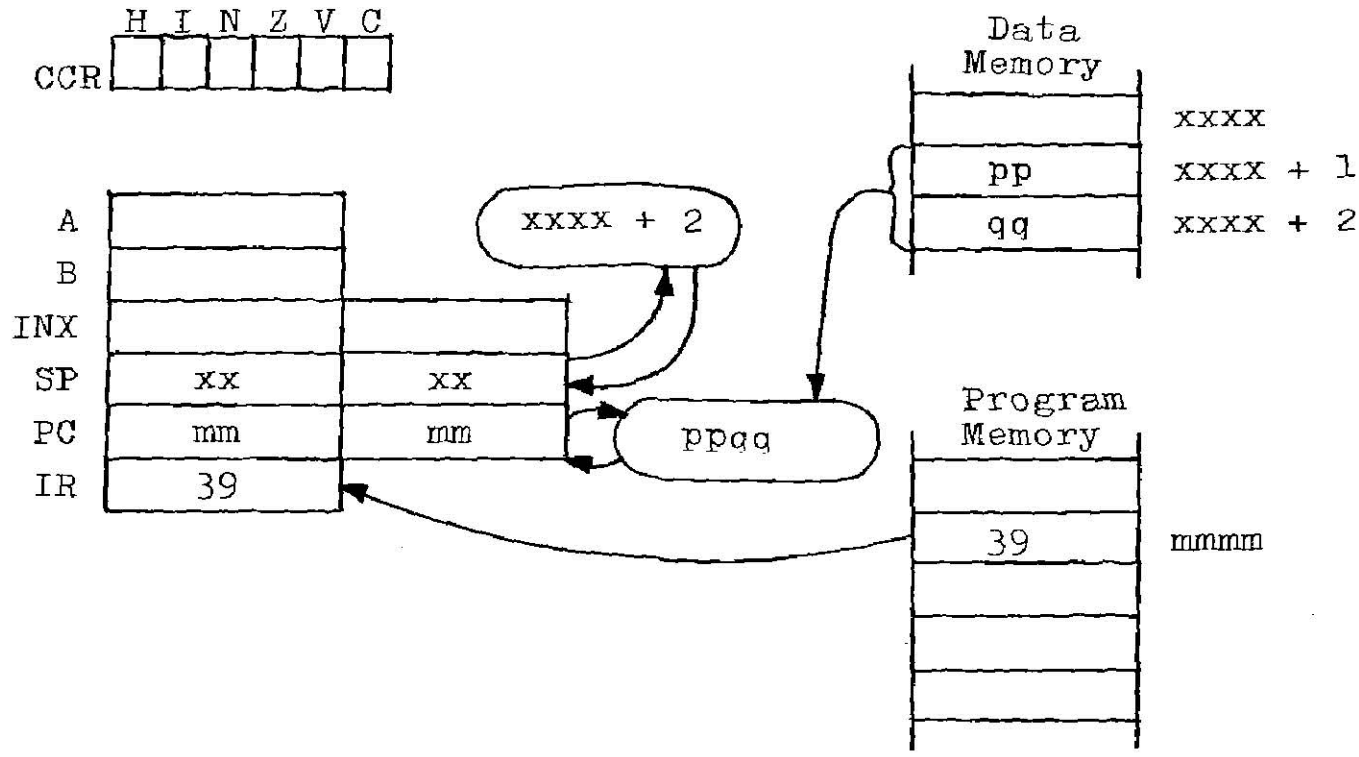
La ejecución continúa desde la dirección obtenida del Contador del Programa.
Suponer que el Apuntador de Pila contiene $100F_{16}$, aa=CB_{16}, bb=14_{16}, cc=00_{16}, dd=01_{16}, ee=00_{16}, ff=09_{16} y qq=$A2_{16}$. Después que la instrucción:

RTI

es ejecutada, el Acumulador A contiene 00_{16}, el Acumulador B contiene 14_{16}, el Registro de Indice contiene 0100_{16}, el Apuntador de Pila contiene 1016_{16} y el Contador del Programa contiene $09A2_{16}$(esta es la dirección desde la cual la ejecución de la instrucción puede ejecutarse). Además el Registro de Código de Condiciones aparece como sigue:

	H	I	N	Z	V	C
CB = 1 1	0	0	1	0	1	1

RTS - Retorno de Subrutina

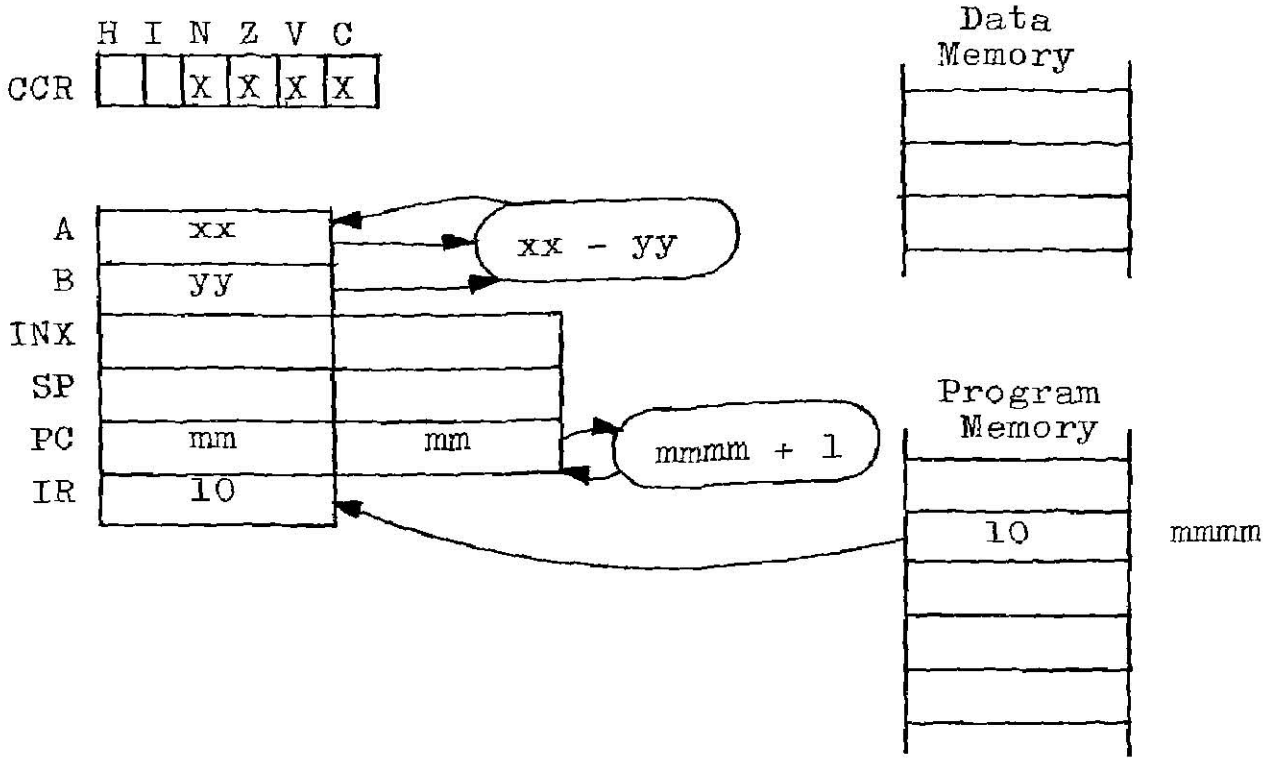


RTS - 39

Mueve el contenido del tope de la pila de 2 bytes al Contador del Programa; esos 2 bytes proveen la dirección de la próxima instrucción a ser ejecutada. El contenido anterior del Contador del Programa se pierde. Se incrementa el Apuntador de Pila en 2 para direccionar el nuevo tope de la pila.

Cada subrutina debe contener al menos una instrucción RETURN esta es la última instrucción ejecutada dentro de la subrutina y causa el retorno al programa que lo llamo.

SBA - Resta los Acumuladores



SBA - 10

Sustrae el contenido del Acumulador B del contenido del Acumulador A.

Suponer que xx=$3A_{16}$ y yy=$7C_{16}$. Después que la instrucción:

SBA

es ejecutada, el Acumulador A contiene BE_{16} y el Acumulador B contiene $7C_{16}$.

3A = 0 0 1 1 1 0 1 0

Compl. a 2 de 7C = 1 0 0 0 0 1 0 0

1 0 1 1 1 1 1 0 ← Coloca Z=0

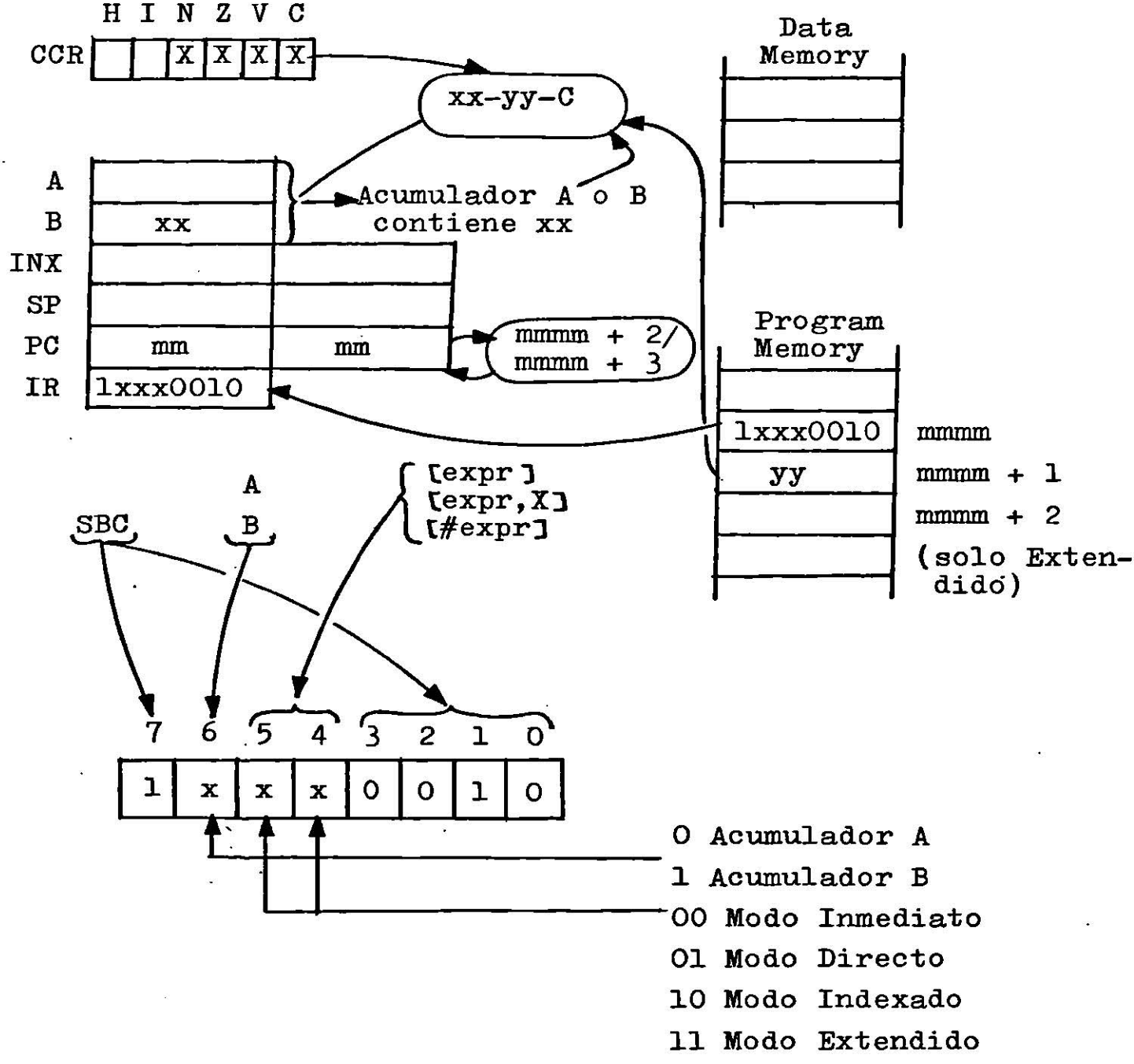
Coloca C=1 ←

N=1 ←

0 ∨ 0=0, V=0

SBC - Resta la Memoria del Acumulador con Borrow

Sustrae el contenido de un byte de memoria seleccionada y el contenido de la bandera Carry del acumulador especificado. Esta instrucción ofrece la misma opción de direccionamiento - de memoria que la instrucción ADC, y puede ser ilustrada usando el modo Inmediato.



Sustrae el contenido del byte de memoria seleccionada, y el - estado Carry, desde el acumulador especificado, tratando todo el contenido de los registros como simples datos binarios.

Suponer que xx=14_{16}, yy=34_{16} y el Carry es 1. Después que la- instrucción:

SBC B #$34

es ejecutada, el contenido del Acumulador B es alterado a --- DF_{16}.

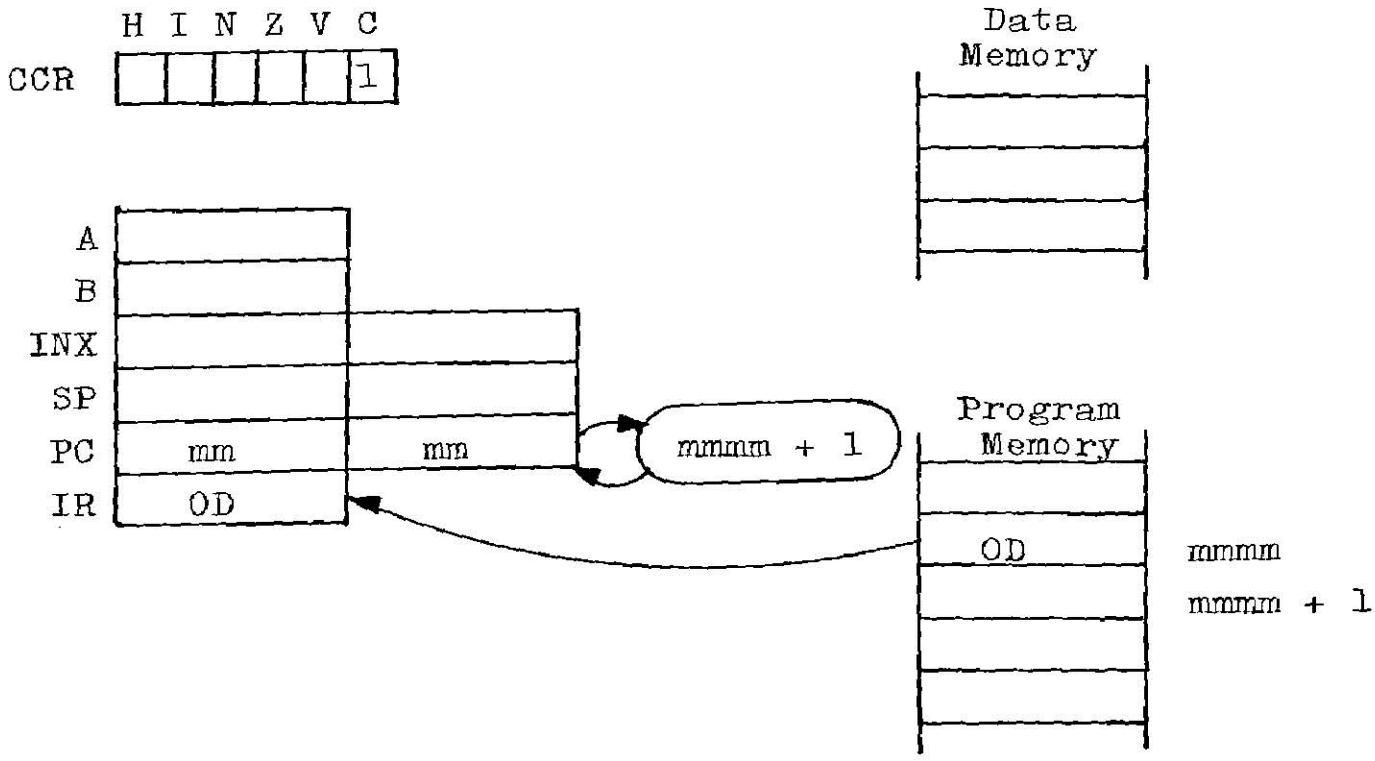
14 = 0 0 0 1 0 1 0 0
Compl. a 2 34 = 1 1 0 0 1 1 0 0
Compl. a 2 1 = 1 1 1 1 1 1 1 1

1 1 0 1 1 1 1 1 ← Coloca Z=0

Coloca C=0 ←
N=1 ←
1 ¥ 1=0, V=1 ←

Note que el resultado del Carry es complementado.

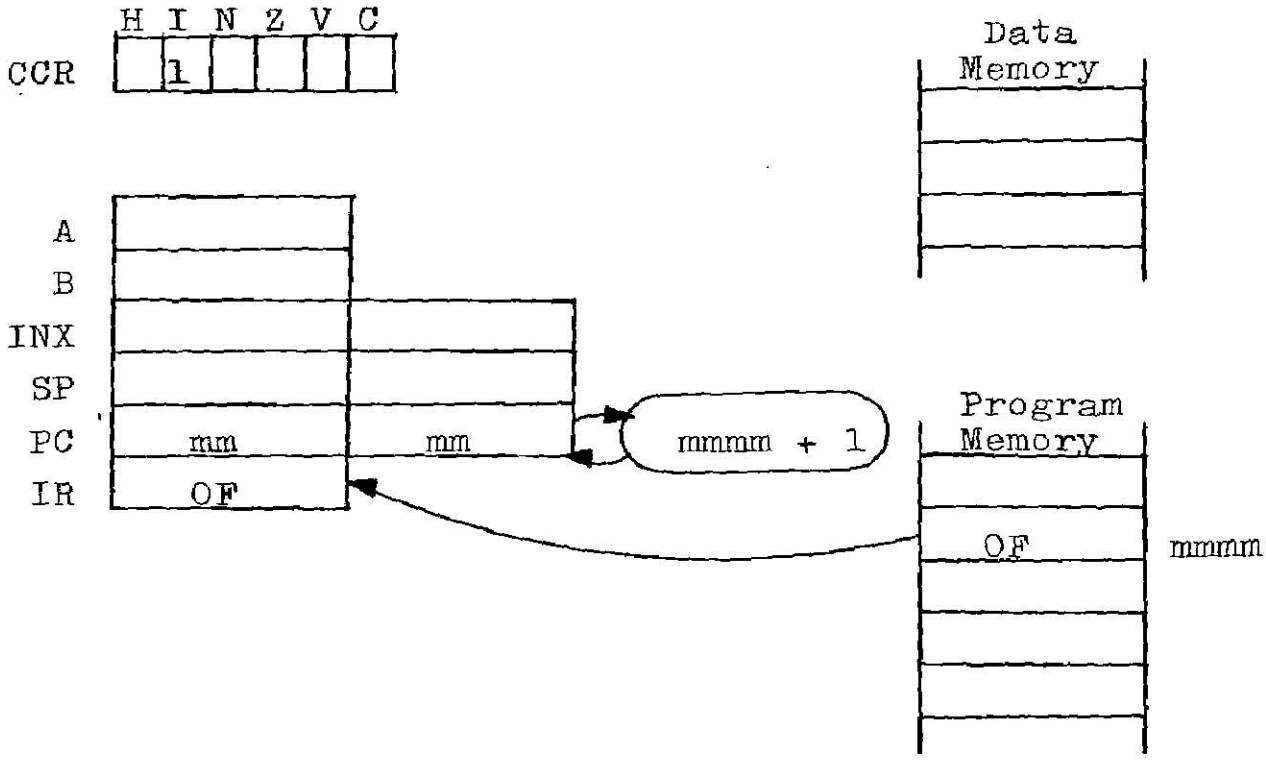
SEC - Coloca el Carry



SEC - OD

Cuando la instrucción SEC es ejecutada, el estado Carry - es 1, sin hacer caso al valor anterior. Ningún otro estado o- registro es afectado.

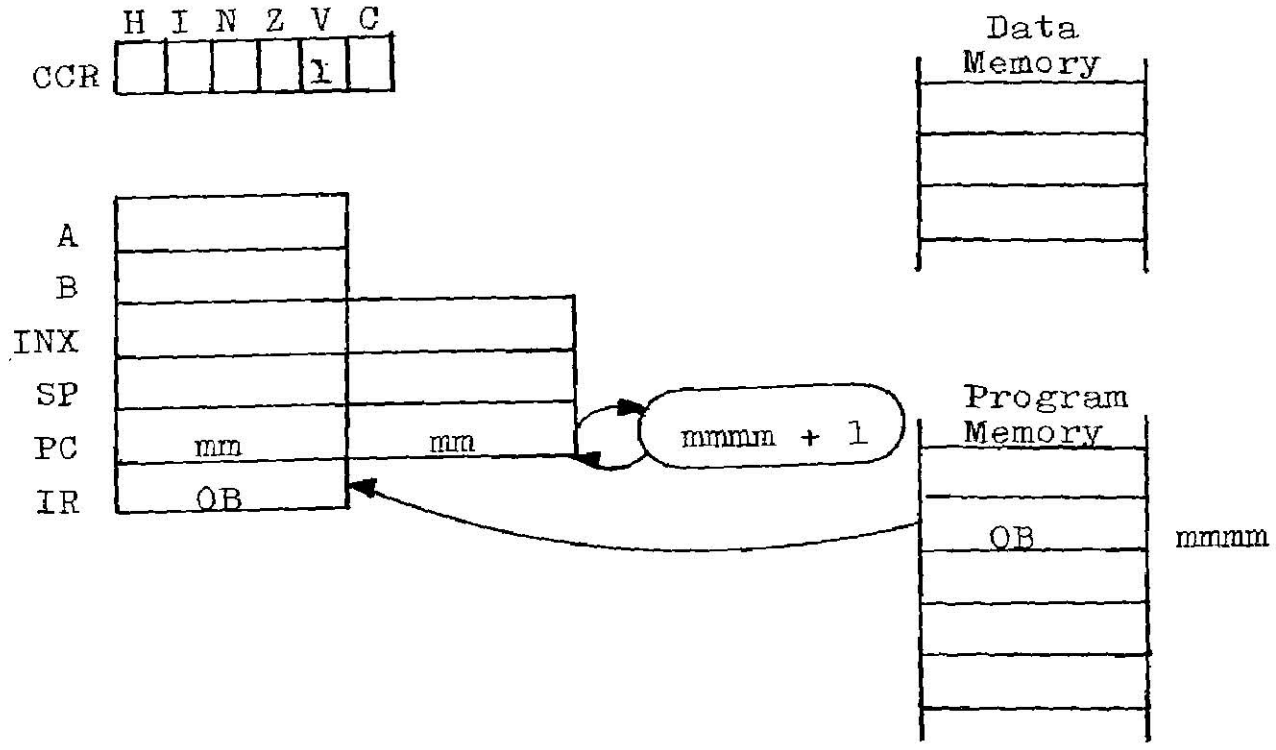
SEI - Coloca la Máscara de Interrupción



SEI - OF

Después que esta instrucción es ejecutada, el MPU es inhibido desde el servicio de una interrupción y puede continuar ejecutando instrucciones, sin responder a interrupciones hasta que el estado Interrupt sea limpiado. Ningún otro estado o registro es afectado.

SEV - Coloca el Estado Overflow

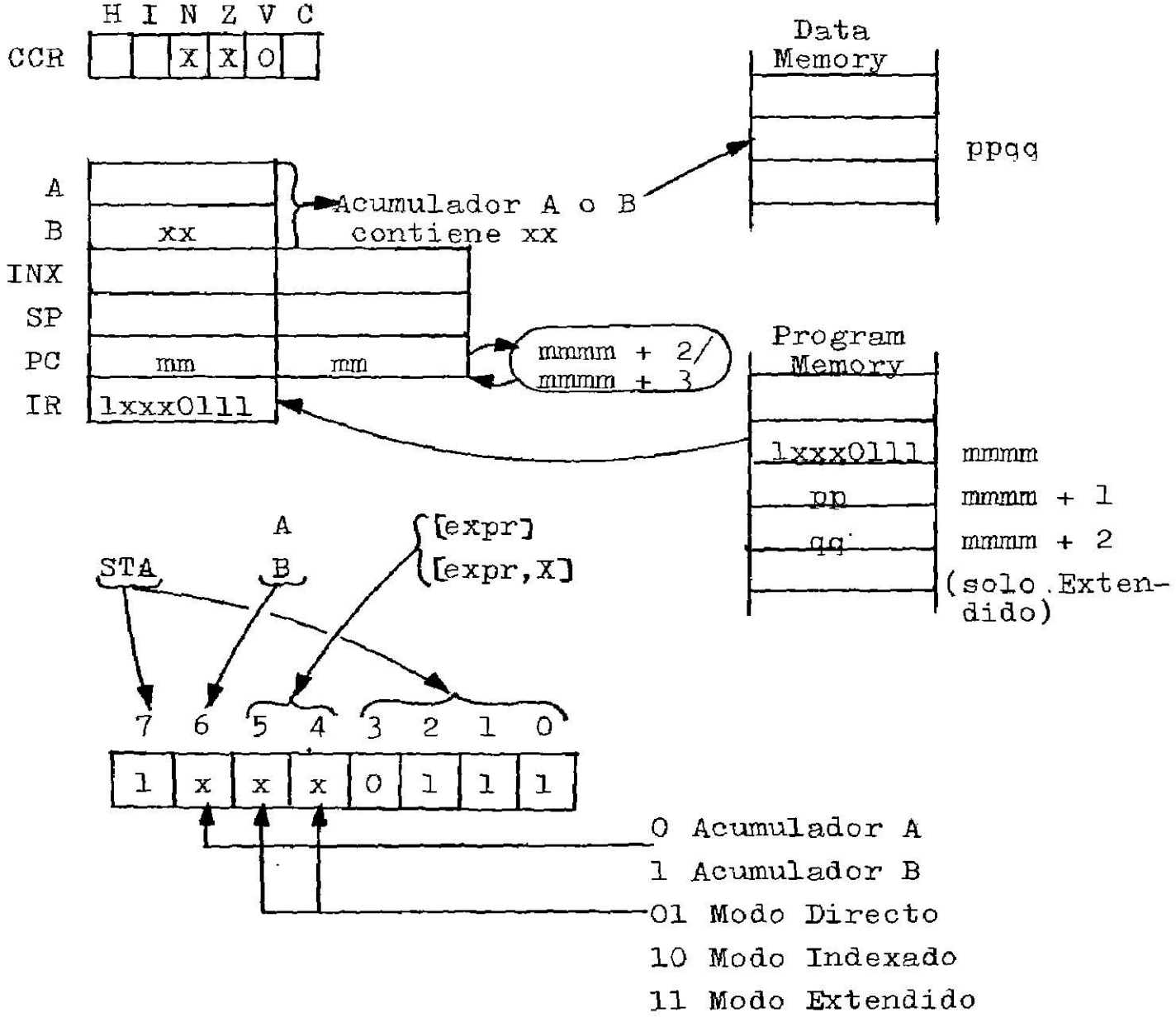


SEV - OB

Cuando la instrucción SEV es ejecutada, el estado Over--- flow es colocado en 1, sin tomar en cuenta su valor anterior. Ningún otro estado o registro es afectado.

STA - Almacena el Acumulador en Memoria

Almacena el contenido de un acumulador seleccionado en -- una localidad de memoria seleccionada. Esta instrucción ofrece la misma opción de direccionamiento de memoria que la instrucción ADC, con la excepción de que el modo de direcciona-- miento inmediato no está disponible. Esta instrucción puede - ilustrarse usando el modo Extendido.

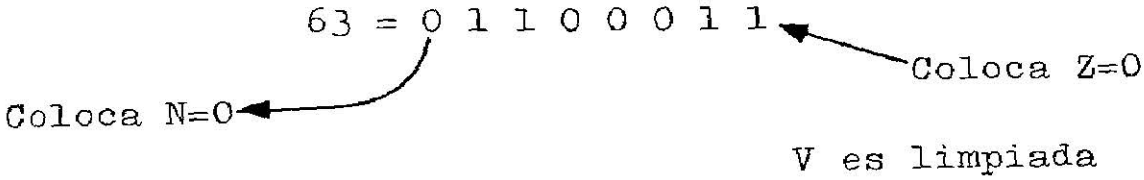


Almacena el Acumulador especificado en Memoria.

Suponer xx=63_{16}, pp=05_{16}, qq=$3A_{16}$. Después que la instruc --- ción:

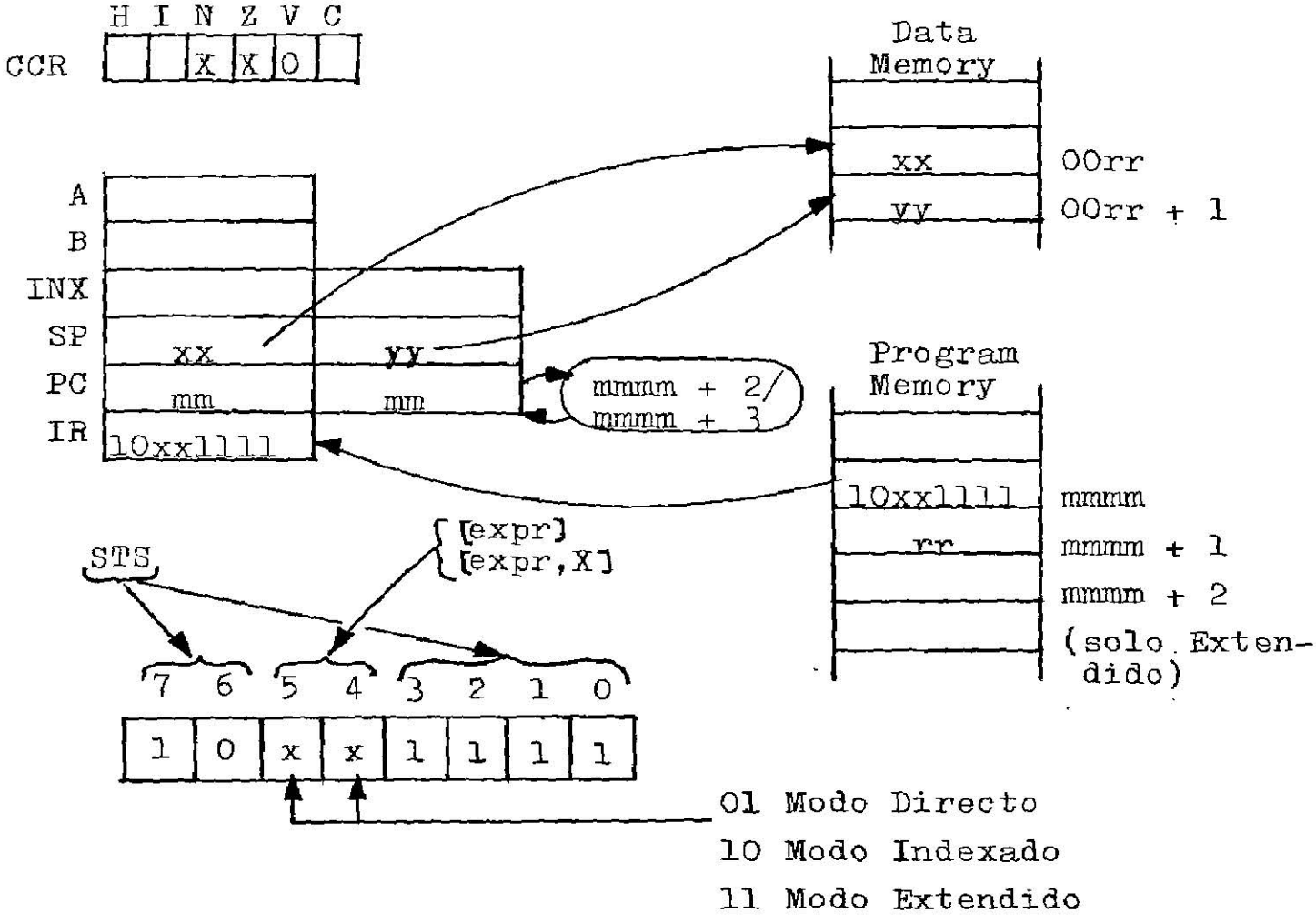
```
STA B  $053A
```

es ejecutada, el contenido de la localidad de memoria $053A_{16}$ - es 63_{16}.



STS - Almacena el Apuntador de Pila

Almacena el contenido del Apuntador de Pila en 2 contí--- nuas localidades de memoria. Como la instrucción STA, esta -- instrucción ofrece los modos de direccionamiento Directo, Indexado y Extendido. Puede ser ilustrada usando el Directo.



Almacena el byte alto del Apuntador de Pila en el byte de memoria seleccionado. Almacena el byte bajo del Apuntador de Pila en el byte de memoria inmediatamente seguido.
Suponer que el Apuntador de Pila contiene $28FF_{16}$ y rr=80_{16}. - Después que la instrucción:

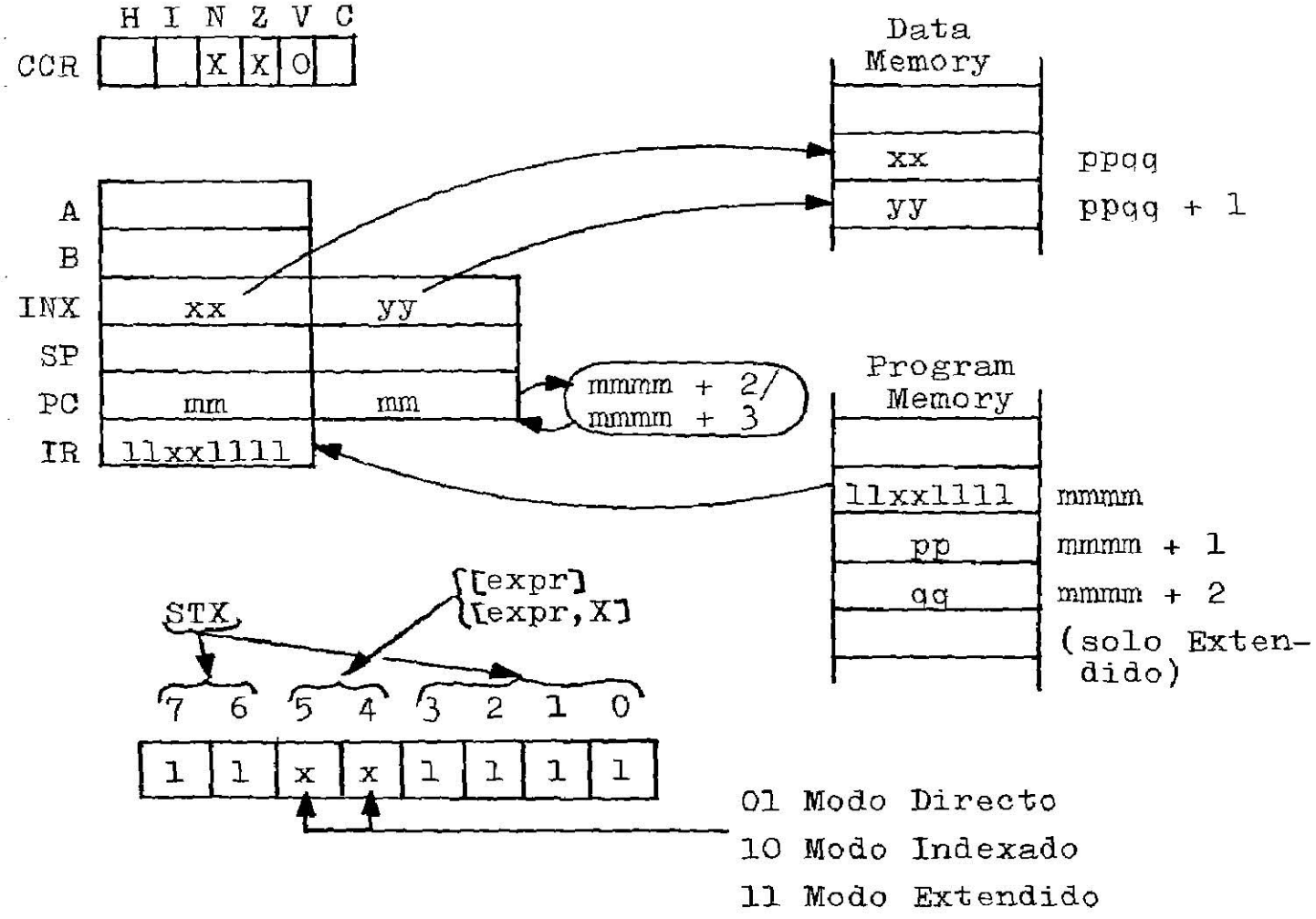
STS

es ejecutada, la localidad 0080_{16} contiene 28_{16} y la localidad 81_{16} contiene FF_{16}.

Coloca N=0 ← 0 0 1 0 1 0 0 0 1 1 1 1 1 1 1 1 → Coloca Z=0
V es limpiada

STX - Almacena el Registro de Indice

Almacena el contenido del Registro de Indice en 2 localidades contínuas de memoria. Como la instrucción STA, no ofrece direccionamiento Inmediato, pero si los otros modos. Puede ser ilustrado usando el modo Extendido.



Almacena el byte alto del Registro de Indice en un byte de memoria seleccionado. Almacena el byte bajo del Registro de Indice en el byte inmediatamente seguido de la localidad de memoria seleccionada. Suponer que el Registro de Indice contiene 0100_{16}, pp=14_{16} y qq=30_{16}. Después de:

STX

la localidad 1430_{16} contiene 01_{16} y la 1431_{16} contiene 00_{16}.

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

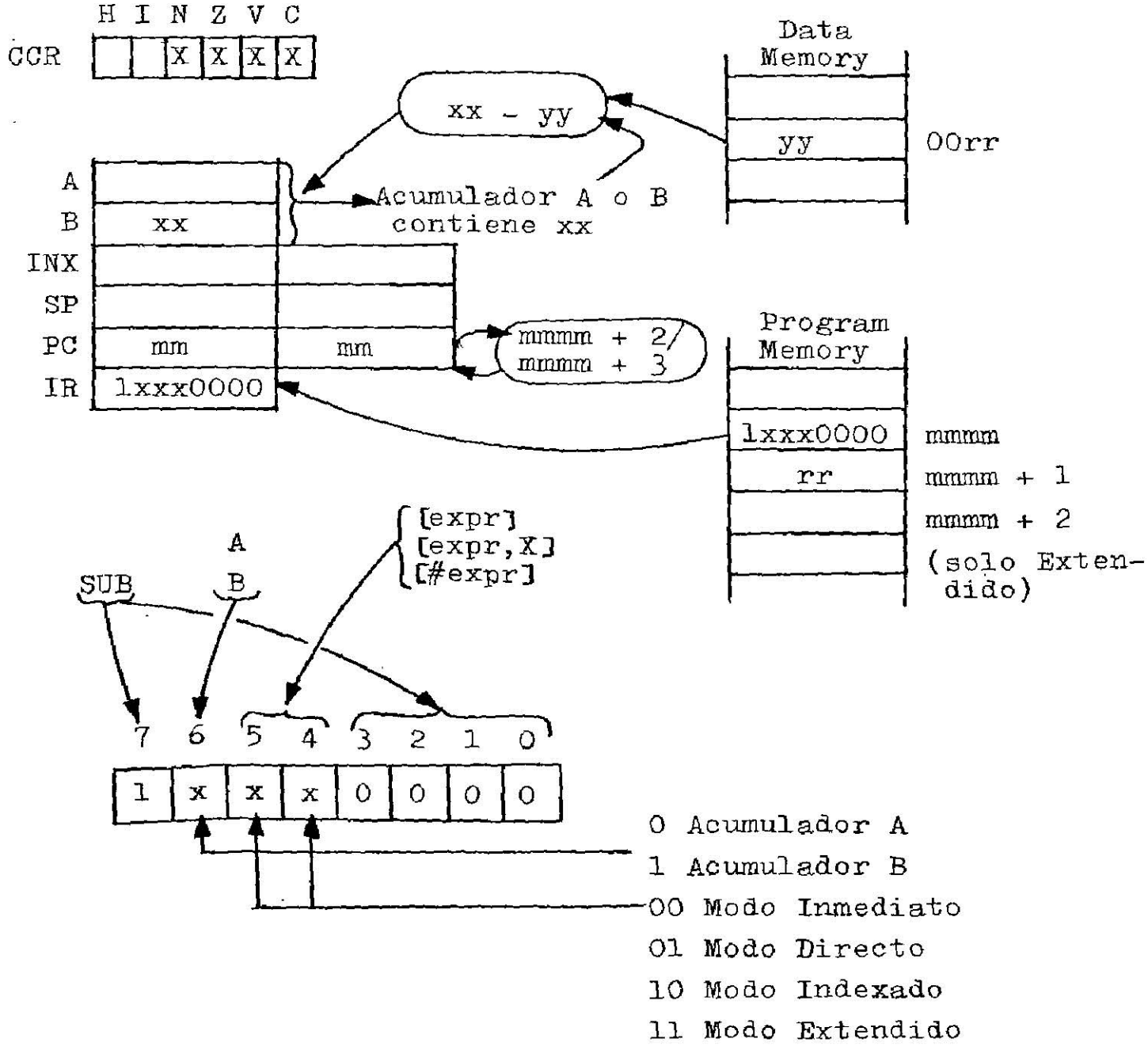
Coloca N=0

Coloca Z=0

V es limpiada

SUB - Resta la Memoria del Acumulador

Sustrae el contenido de un byte de memoria seleccionado - del contenido del acumulador A o el B. Esta instrucción ofrece la misma opción de direccionamiento oue la instrucción ADC, y puede ser ilustrada usando el modo Directo.

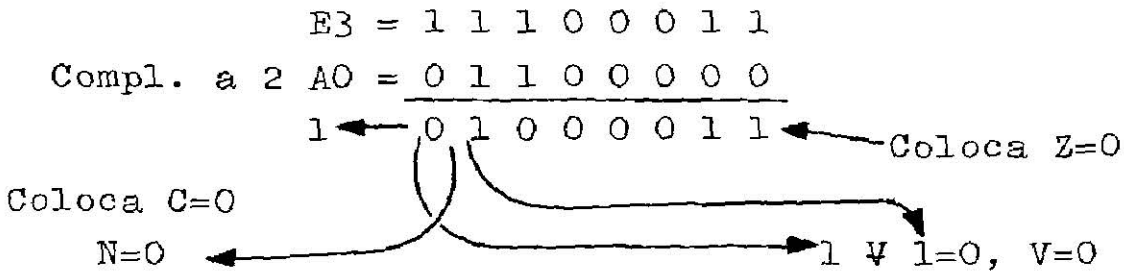


Sustrae el contenido del byte de memoria seleccionado del contenido del acumulador específico, tratando ambos operandos como simples datos binarios.

Suponer que xx=$E3_{16}$, yy=AO_{16} y rr=31_{16}. Después que la instrucción:

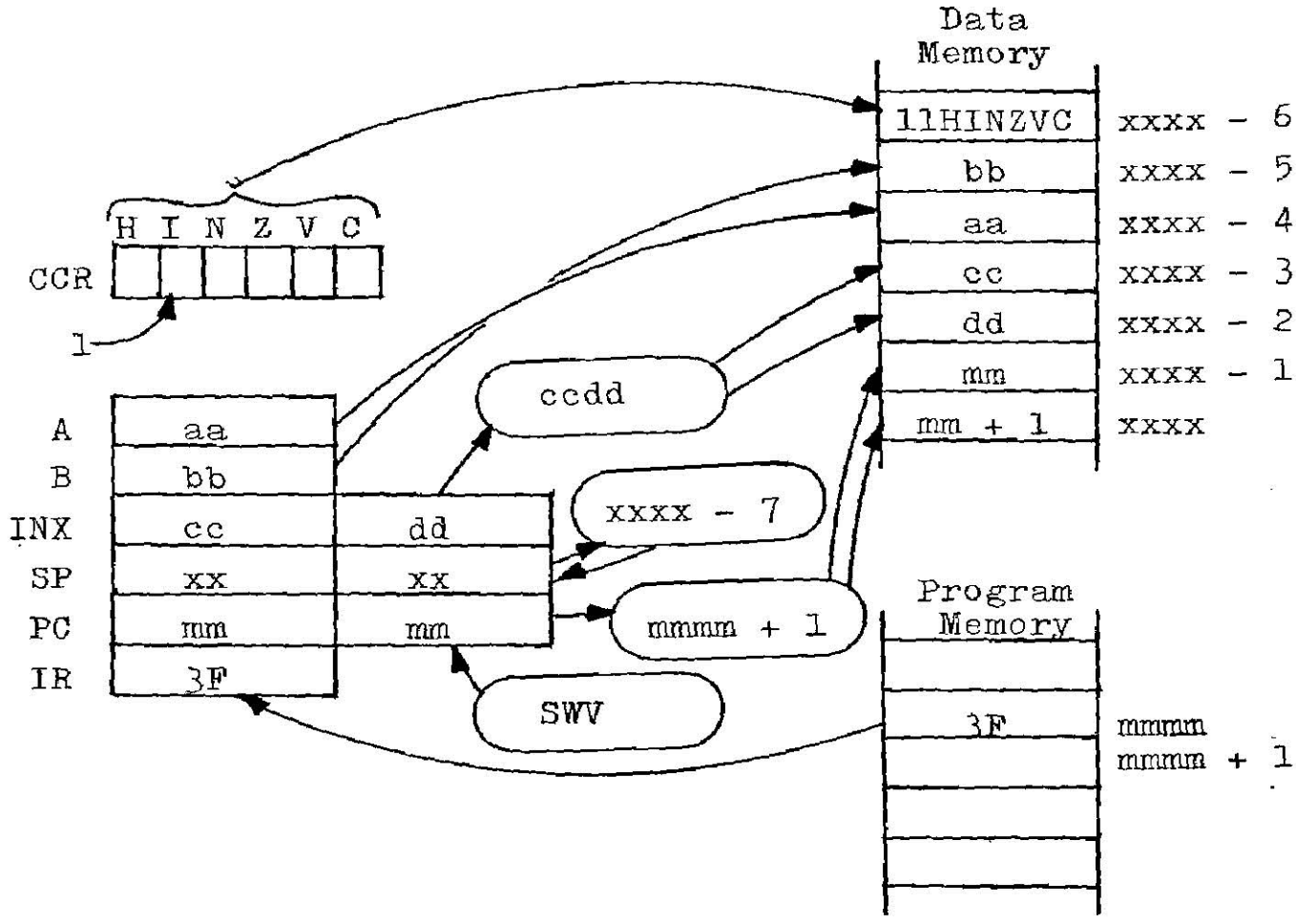
```
SUB B  $31
```

es ejecutada, el contenido del acumulador B es 43_{16}



La instrucción SUB es usada para ejecutar sustracciones de un solo byte.

SWI - Interrupción del Software



SWI - 3F

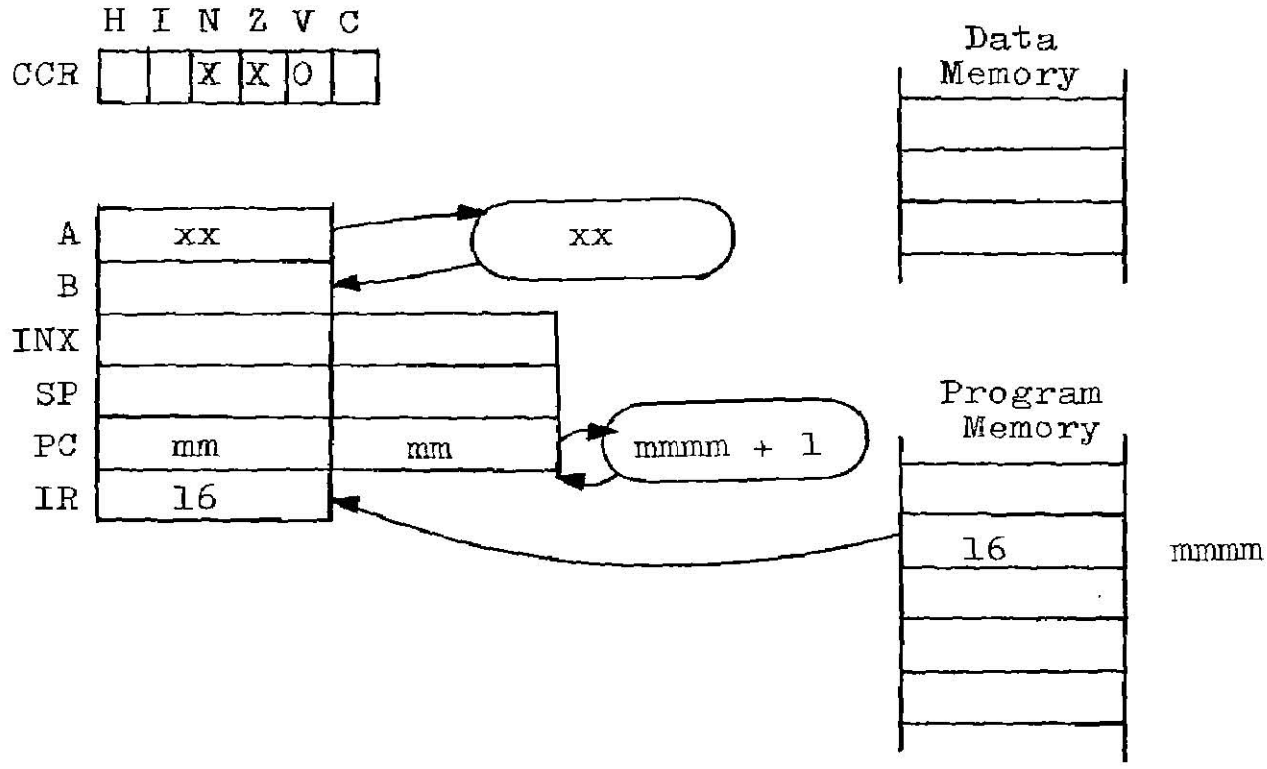
El Contador del Programa es incrementado en 1, entonces - el Contador del Programa, el Registro de Indice, los Acumuladores y el Registro de Código de Condiciones son todos insertados en la pila. Los registros y sus localidades correspon-- dientes en el cual se insertan son los siguientes:

Localidades de Memoria	Registro
(SP es XX al empezar la ejecución de la instrucción)	
XXXX	Byte bajo del Contador - del Programa
XXXX-1	Byte alto del Contador - del Programa
XXXX-2	Byte bajo del Registro - de Indice
XXXX-3	Byte alto del Registro - de Indice
XXXX-4	Acumulador A
XXXX-5	Acumulador B
XXXX-6	Registro de Código de -- Condiciones

El bit de máscara de interrupción es colocado en 1. Este incapacita el servicio de interrupción del MC 6800, esto es, el - procesador no puede responder a una interrupción desde un dispositivo periférico. El contenido del SWV(Software Interrupt-Pointer) es cargado en el Contador del Programa.

La instrucción SWI puede ser usada para una variedad de funciones. La dirección del punto de entrada para un grupo de -- sistemas de subrutinas o la dirección del punto de entrada -- para un sistema operativo de disco o la dirección de cualquier paquete software pueden ser insertados en el Software - Interrupt Pointer. Al ejecutarse una instrucción SWI cualquiera de esos sistemas Software pueden entrar.

TAB - Mueve el Acumulador A al Acumulador B



TAB - 16

Mueve el contenido del Acumulador A a el Acumulador B. Colocando los estados Sign y Zero por consiguiente, limpiando - el estado Overflow. Suponer que $xx=00_{16}$. Después que la instrucción:

TAB

es ejecutado, los Acumuladores A y B contienen 0.

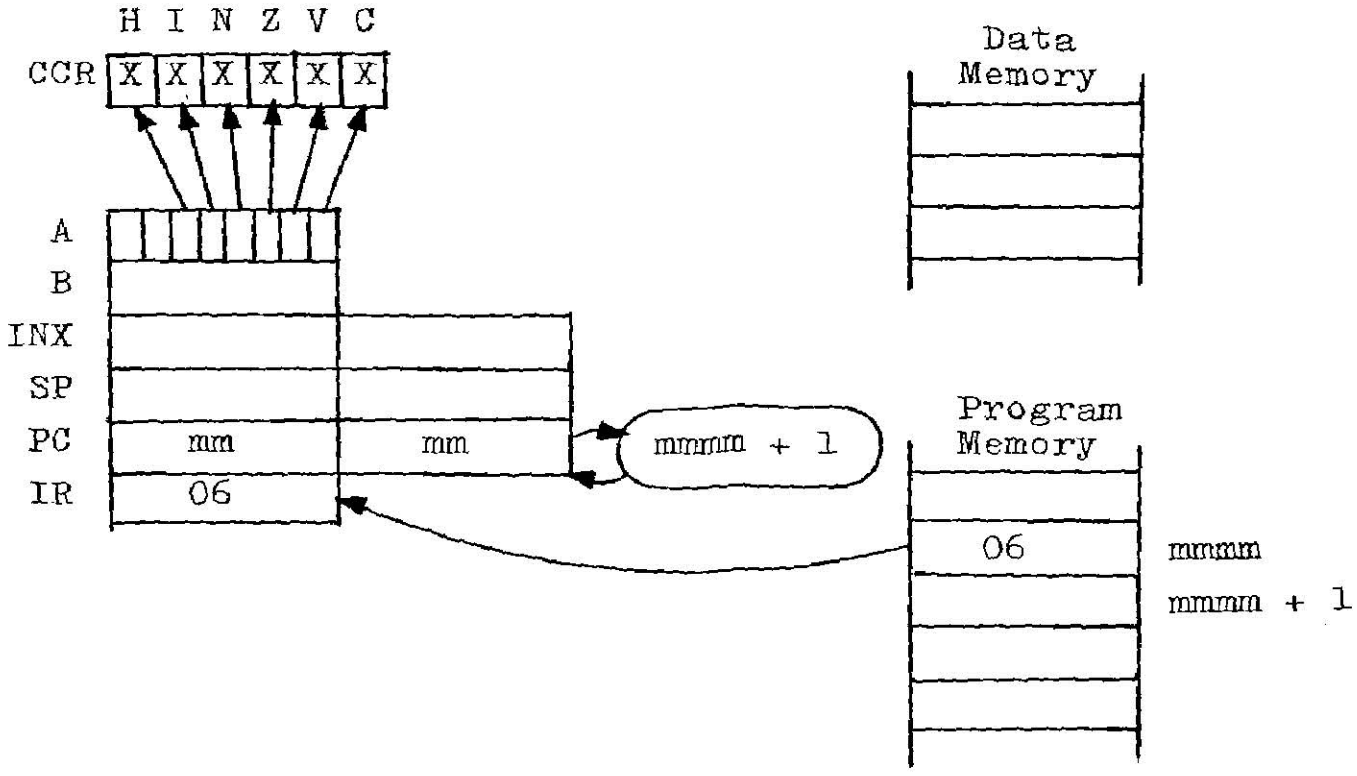
0 0 0 0 0 0 0 0 ← Coloca Z=1

Carry no afectado

N=0

V es limpiado

TAP - Mueve el Acumulador al Registro de Código de Condiciones

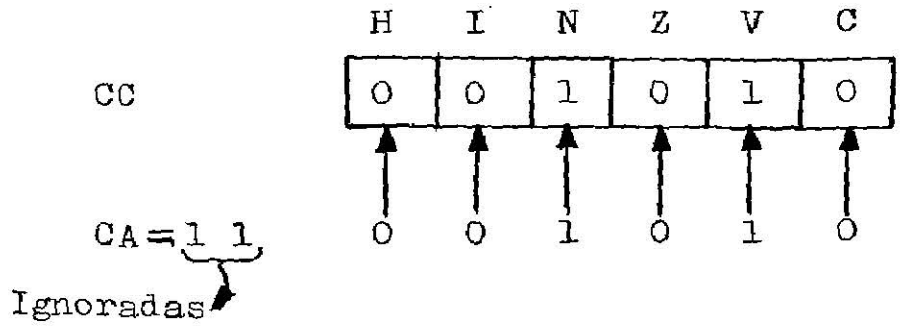


TAP - 06

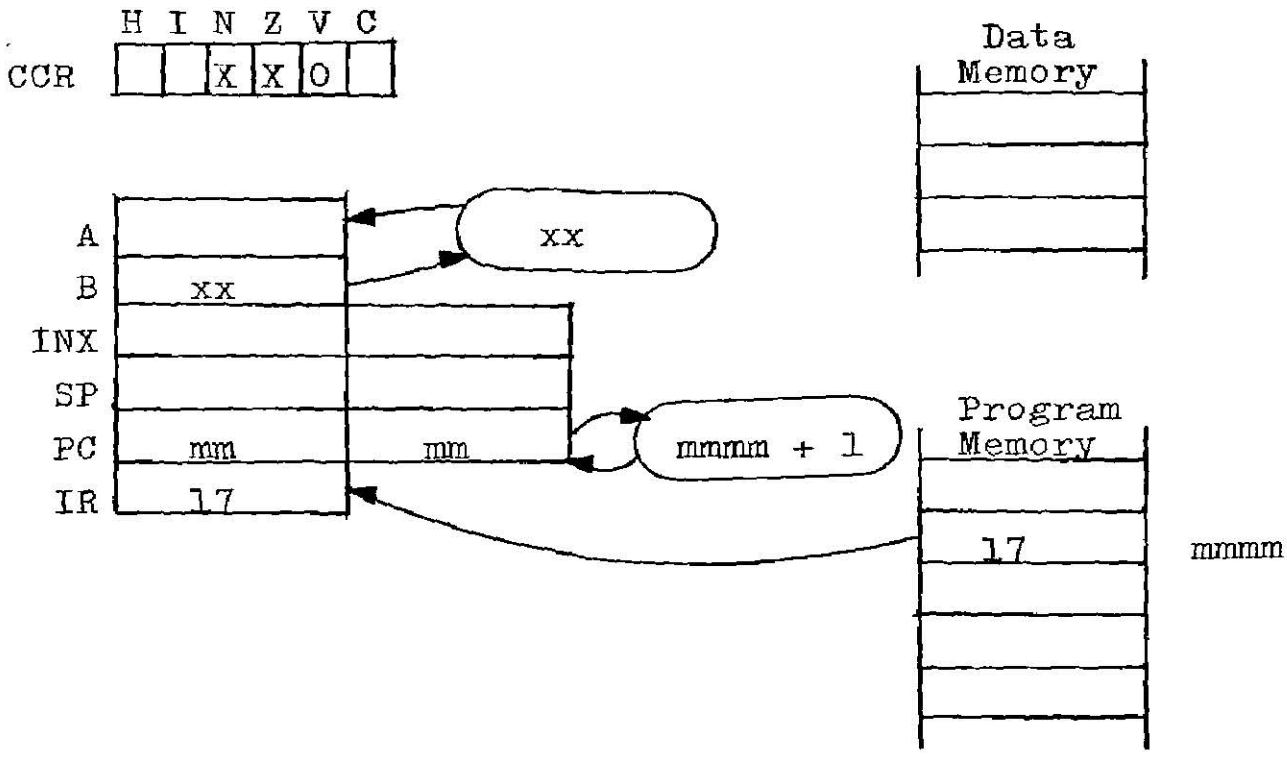
Mueve los bits 5-0 del Acumulador A al Registro de Código de Condiciones. Suponer que el Acumulador A contiene CA_{16}. -- Después de ejecutar:

TAP

El Registro de Código de Condiciones contiene lo siguiente:



TBA - Mueve el Acumulador B al Acumulador A



TBA - 17

Mueve el contenido del Acumulador B al Acumulador A. Coloca los estados Sign y Zero por consiguiente. Limpia el estado Overflow. Suponer que xx=$C3_{16}$. Después de ejecutar:

TBA

los Acumuladores A y B contienen $C3_{16}$.

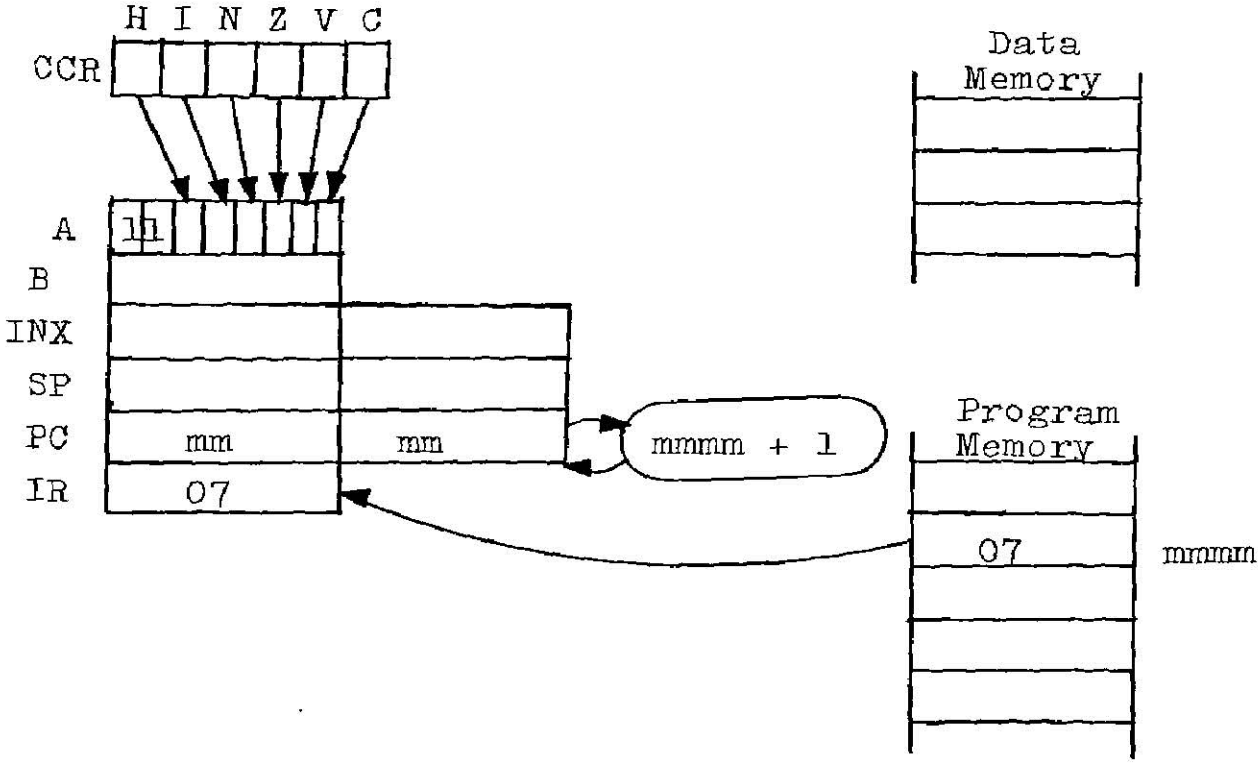
1 1 0 0 0 0 1 1 ← Coloca Z=0

Carry no afectado

V es limpiado

N=1

TPA - Mueve el Registro de Código de Condiciones al Acumulador



TPA - 07

Mueve el contenido del Registro de Código de Condiciones- al Acumulador A, los bits 0-5. Coloca los bits 6 y 7 del Acumulador A en 1. Suponer el Registro CC como sigue:

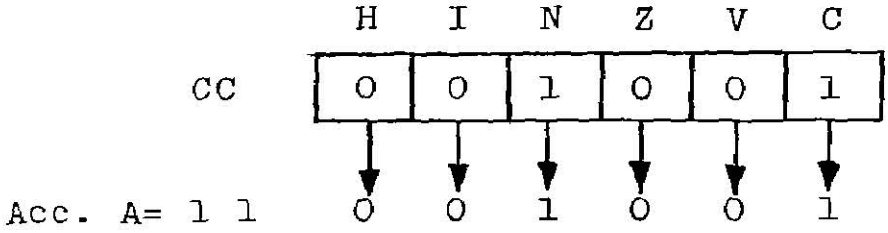
N y C son 1.

H, I, Z y V son 0.

Después de ejecutar:

TPA

el Acumulador A contiene $C9_{16}$.

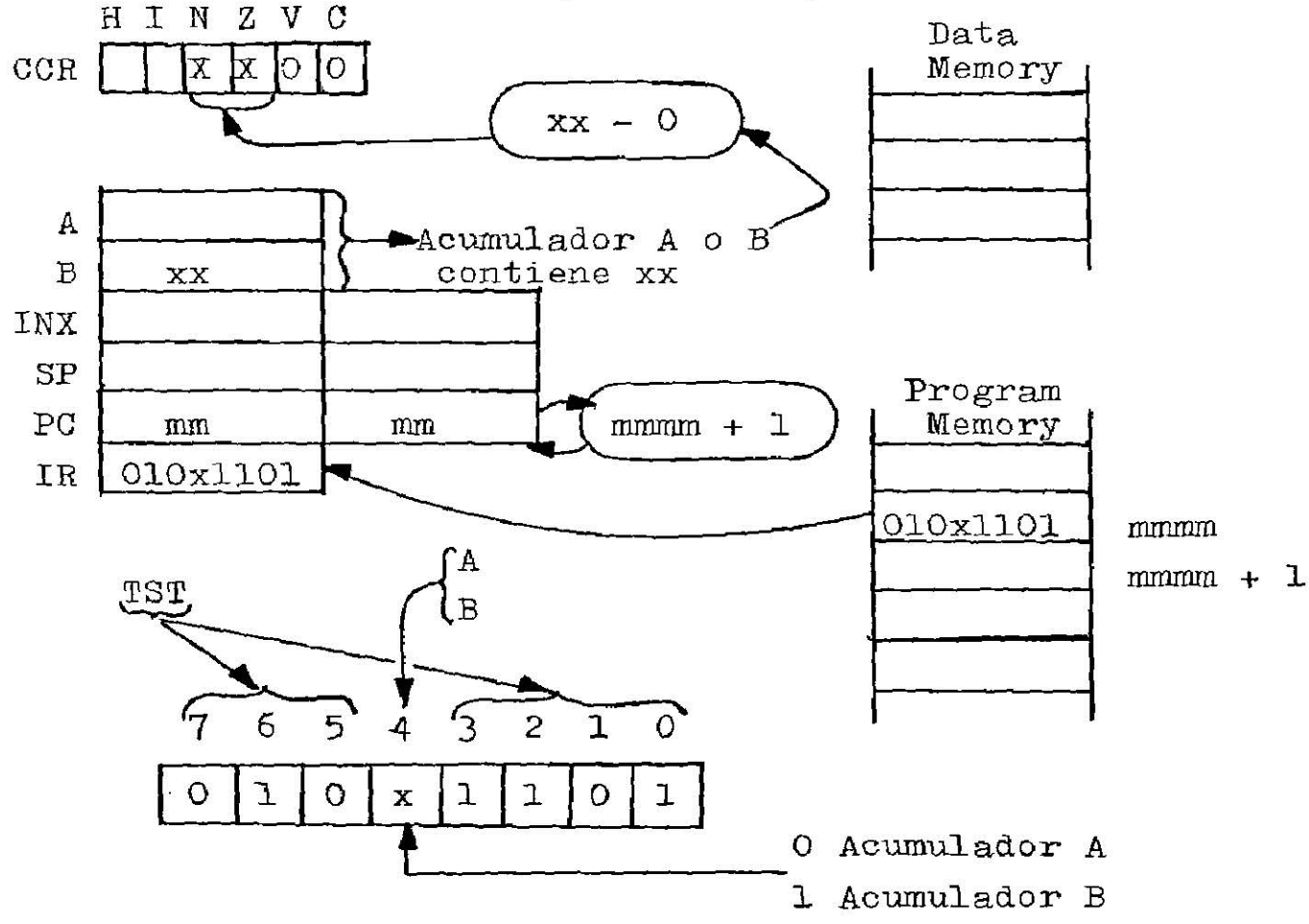


El Acumulador A es el único registro afectado.

TST - Prueba el Contenido del Acumulador o de Memoria

Coloca las banderas Sign y Zero dependiendo del contenido del acumulador especificado o de un byte de memoria seleccionado.

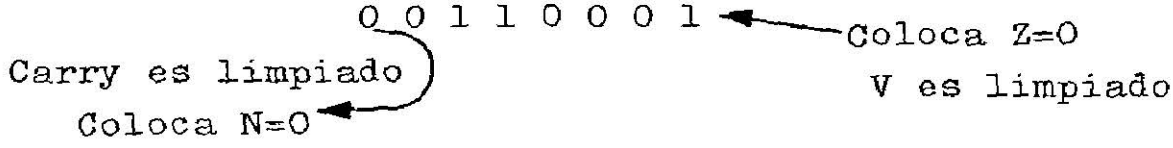
Primero, consideremos la prueba del Acumulador.



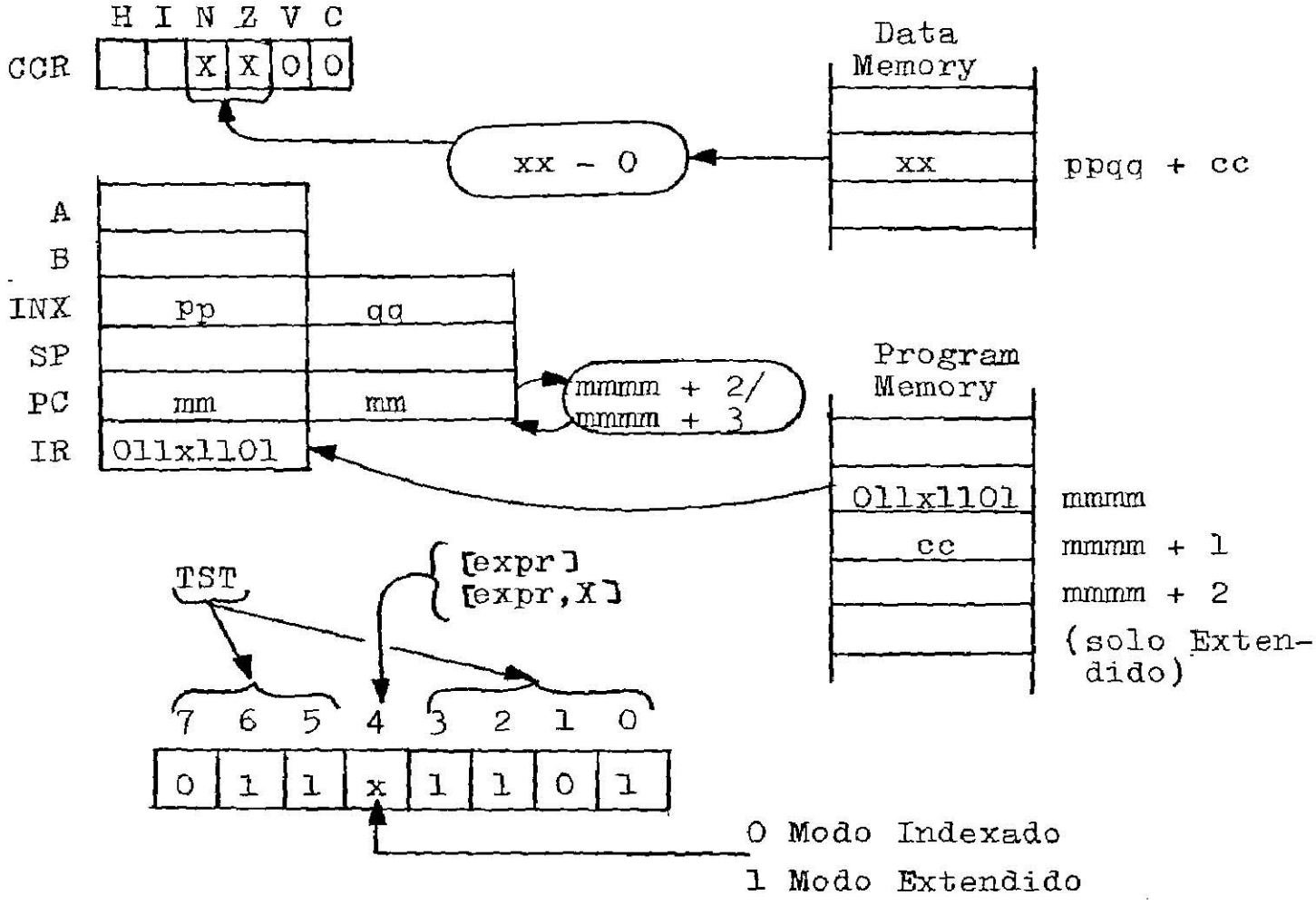
Coloca las banderas Sign y Zero dependiendo del resultado de-sustraer 00_{16} del Acumulador A o B. Limpia las banderas Overflow y Carry. Suponer que $xx=31_{16}$. Después de ejecutar:

TST B

los estados Sign, Zero, Overflow y Carry son 0.



La instrucción TST ofrece 2 modos de direccionamiento, Indexado y Extendido.

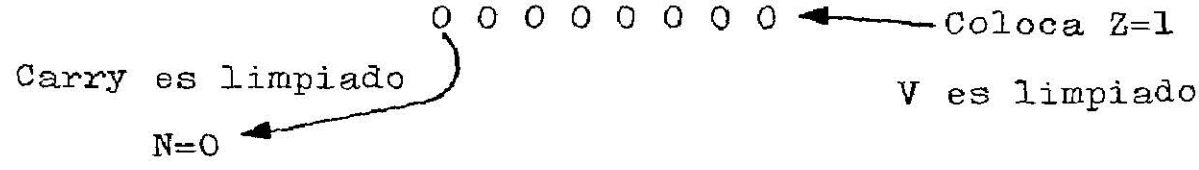


Prueba el byte de memoria seleccionada restando 00_{16} de ese contenido. Coloca las banderas Sign y Zero por consiguiente, y limpia los estados Overflow y Carry.

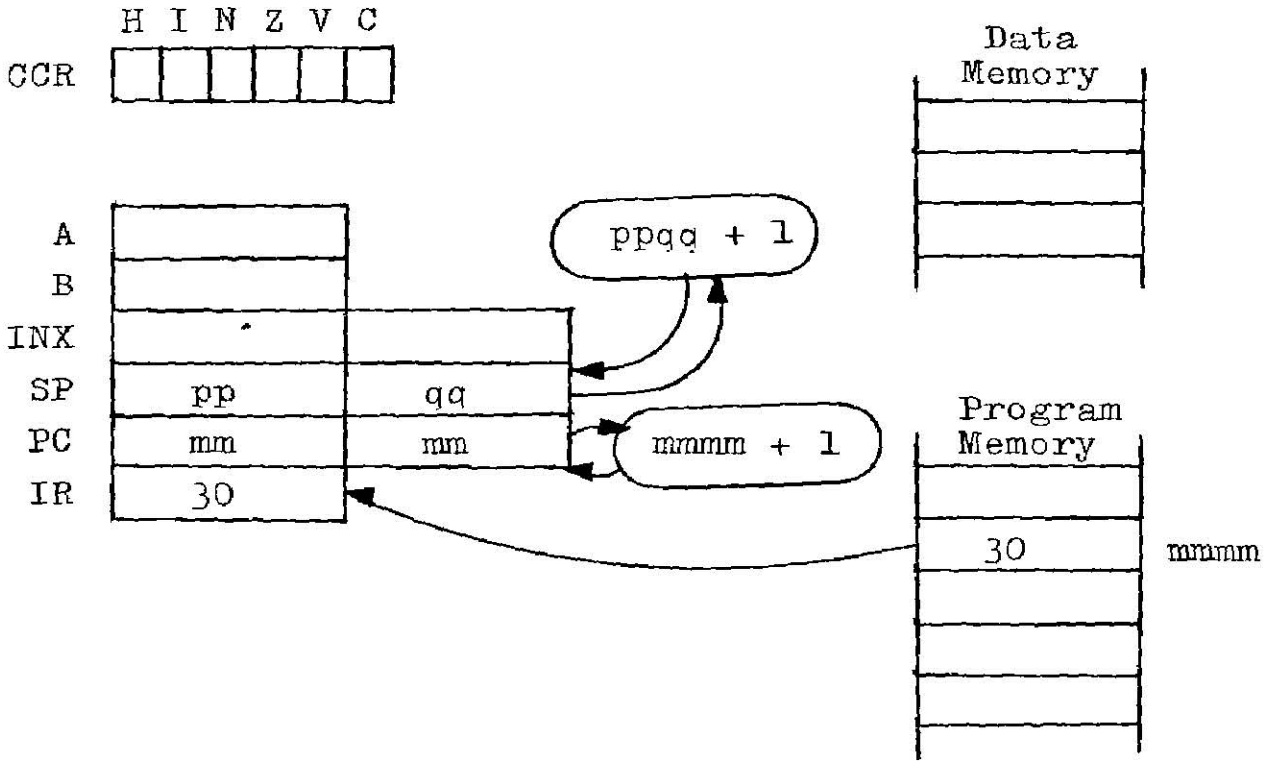
Suponer que el Registro de Indice contiene 0100_{16}, cc=02_{16} y la localidad 0102 contiene 00_{16}. Al ejecutar:

```
TST  2,X
```

coloca las banderas Sign, Overflow y Carry en 0 y la bandera Zero en 1.



TSX - Mueve el Apuntador de Pila al Registro de Indice



TSX - 30

Mueve el contenido del Apuntador de Pila al Registro de Indice y lo incrementa en 1.

Suponer ppqq=$2AF7_{16}$. Después que la instrucción:

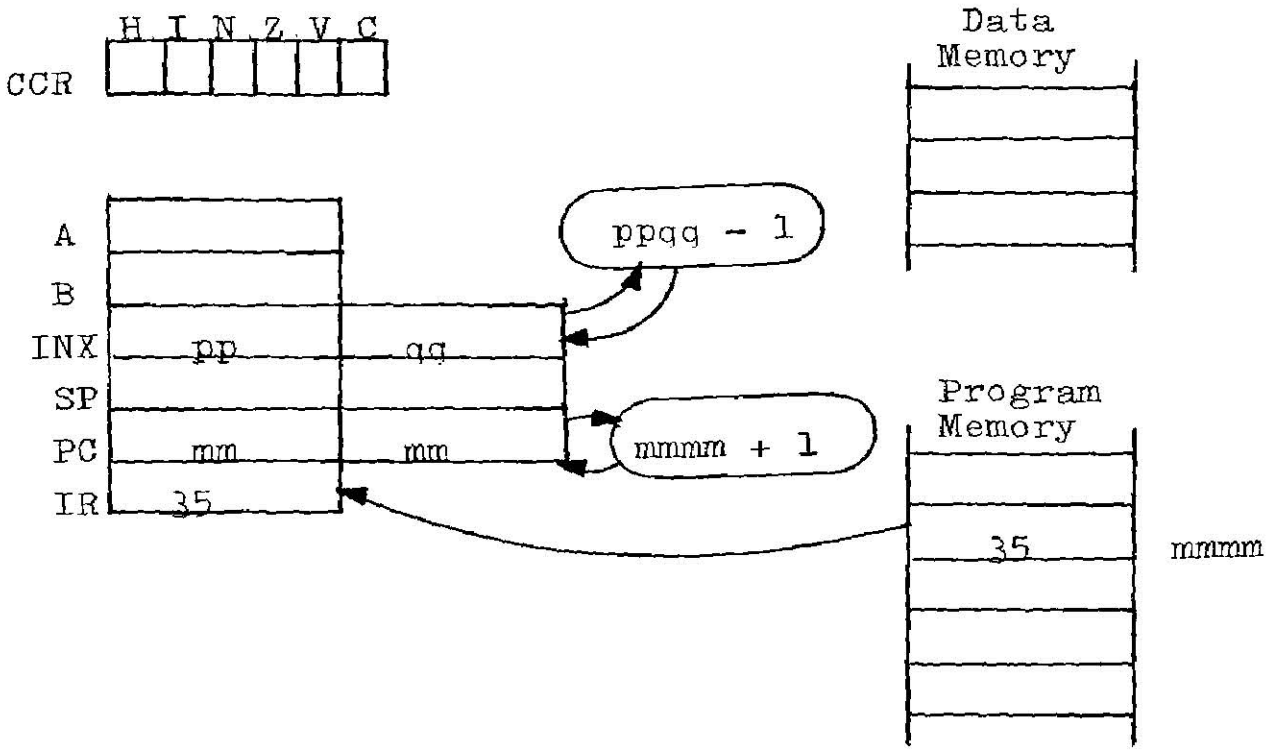
TSX

es ejecutada, el Registro de Indice contiene $2AF8_{16}$.

La razón de que el Registro de Indice sea cargado con el contenido del Apuntador de Pila más uno es para seguir el Registro de Indice en un punto directamente al fondo de la pila. - Recordando que empleando MC 6800 se decrementa después de escribir, incrementa antes de leer la pila.

Ningún otro registro o estado es afectado.

TXS - Mueve el Registro de Indice al Apuntador de Pila



TXS - 35

Mueve el contenido del Registro de Indice al Apuntador de Pila y lo decrementa en 1.

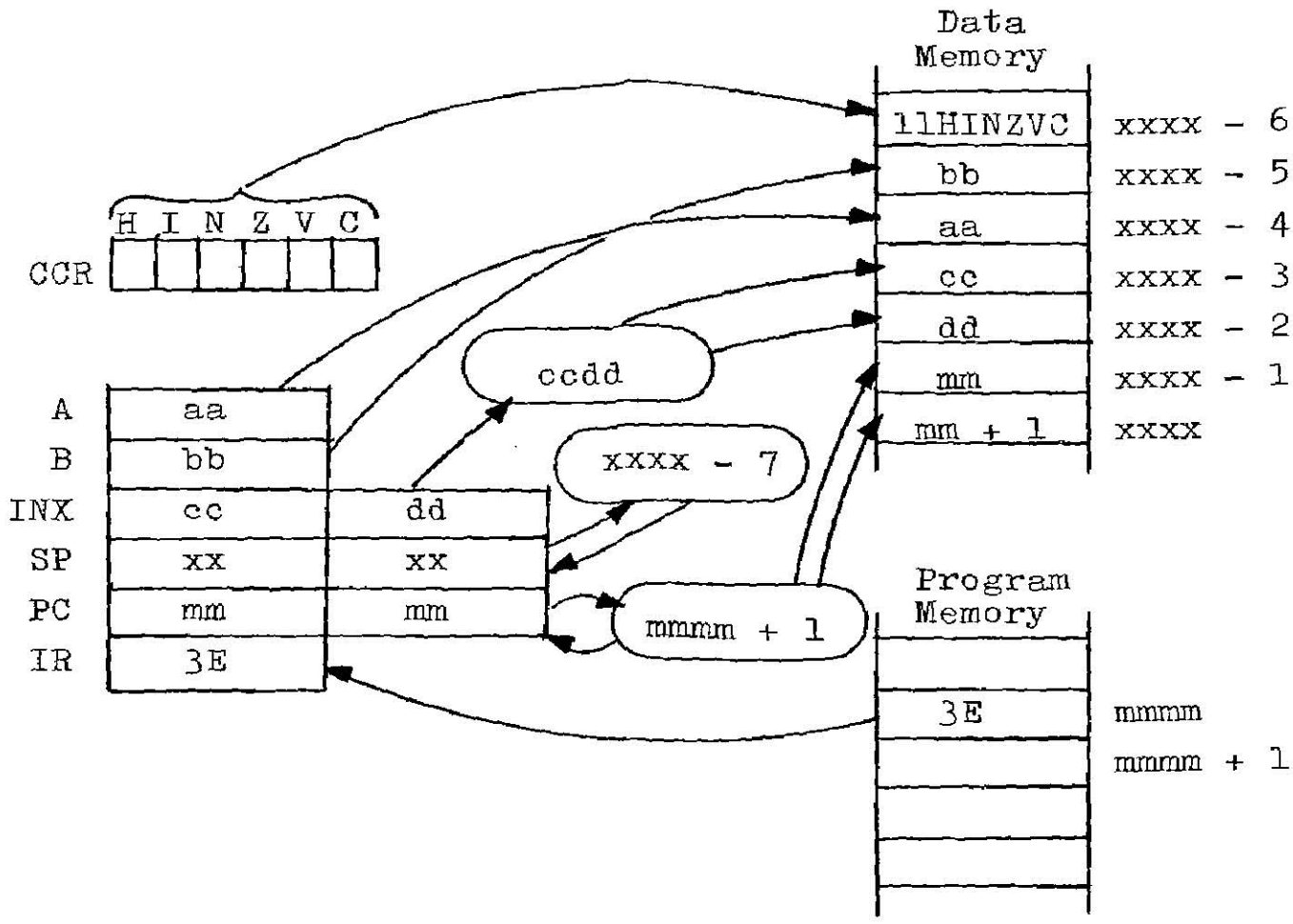
Suponer ppqq=$2AF8_{16}$. Después que la instrucción;

TXS

es ejecutada, el Apuntador de Pila contiene $2AF7_{16}$.

Ningún otro estado o registro es afectado.

WAI - Espera por una Interrupción



WAI - 3E

El Contador del Programa es incrementado en 1, entonces - el Contador del Programa, el Registro de Indice, los Acumuladores A y B, y el Registro de Código de Condiciones son insertados en la pila. Los registros y localidades de memoria co-- rrespondientes en el cual son insertados son los siguientes:

Localidad de Memoria	Registro

(el SP es XXXX al empezar la ejecución de la instrucción)

XXXX	Byte bajo del Contador - del Programa
XXXX-1	Byte alto del Contador - del Programa
XXXX-2	Byte bajo del Registro - de Indice
XXXX-3	Byte alto del Registro - de Indice
XXXX-4	Acumulador A
XXXX-5	Acumulador B
XXXX-6	Registro de Código de - Condiciones

Después que los estados del sistema han sido salvados en la - pila, la ejecución es detenida hasta que el dispositivo periférico solicite una interrupción. Cuando una interrupción es- solicitada, el bit de máscara de interrupción es colocado en- 1 y un salto es realizado en la dirección contenida en el -- EXTERNAL INTERRUPT normal.

5.4.- ENSAMBLADOR DEL MICROPROCESADOR

Un Ensamblador es un programa que procesa los estatutos - de un programa fuente escrito en un lenguaje también llamado- Ensamblador y traduce este programa fuente a programas objeto ejecutables por un determinado procesador. En este tipo de -- traductores cada línea del programa fuente genera una traducción en lenguaje de máquina.

5.4.1.- FORMATO DE UN ESTATUTO FUENTE

Los programas escritos en lenguaje ensamblador consisten- de una secuencia de estatutos fuente que son una secuencia de caracteres ASCII, terminando con el caracter " carriage re-- turn". Cada estatuto fuente puede estar formado hasta de 5 -- campos que son:

- Número de Línea.- Consiste de 5 dígitos decimales(cuyo va-- lor no exceda de 65,535) al inicio de la línea fuente. El - número de línea debe estar precedido de un espacio. Estos - números son opcionales, pero se debe ser consistente al utilizarlos: los tiene todo el programa o no.
- Campo de Etiquetas.- Se ubica después del número de línea - (si existe) y puede tomar una de las siguientes formas:
  + Un asterisco (*) como primer caracter indica que el resto de la línea es un comentario.
  + Un espacio como primer caracter indica que el campo de -- etiquetas está vacío.
  + Un símbolo consistente de 1-6 caracteres alfanuméricos, - siendo el primero una literal.
- Campo de Operaciones.- Está ubicado después del campo de -- etiquetas, la operación puede ser de 2 tipos:
  + El mnemónico de la instrucción de máquina correspondien-- te a una instrucción MC 6800.

Una directiva; operación especial que reconoce el ensam-- blador y controla el proceso de ensamble.

- Campo de Operandos.- La interpretación del campo de operandos depende del campo de operaciones, es decir, si existe - un mnemónico o una directiva. Para los mnemónicos del M6800 el campo de operando debe especificar el modo de direccionamiento. El formato para el campo de operandos en el caso de operaciones código de máquina y el correspondiente modo de-direccionamiento son como sigue:

Formato del Operando	Modo de Direccionamiento
Sin Operando	Inherente y Acumulador
Expresión	Directo o Extendido
# Expresión	Inmediato
Expresión, X	Indicado

Las directivas del ensamblador pueden tomar otra forma.

- Campo de Comentarios.- Esta separado del campo de operandos por uno o más espacios. Este campo es opcional y es ignorado por el ensamblador en el proceso de traducción. Sin em-- bargo, es importante en la documentación del programa.

5.4.2.- EXPRESIONES

Una expresión aritmética es una combinación de símbolos - y/o números separados por uno de los operadores aritméticos - (+, -, * o /).

El ensamblador evalúa las expresiones algebraicas sin a-- grupamientos con paréntesis. No existe jerarquía entre los -- operadores aritméticos. Un resultado fraccionario será trunca do a un valor entero.

- Constantes
  Decimales : número
  Hexadecimales : $ número o número H
  Octales : número o número Q
  Binarios : % número o número B

- Literales ASCII
  ' caracter (apóstrofe seguido de un carácter ASCII). El resultado es el valor numérico carácter --- ASCII.

5.4.3.- DIRECTIVAS

Las Directivas que reconoce el ensamblador M6800 son:

DIRECTIVA	FUNCION
De Control de Ensamble	
NAM	Asigna un nombre al programa.
ORG	Origen(dirección inicial) del programa.
END	Fin del programa.
Definición de Datos/ Localización de Almacenamiento	
FCC	Cadena de Caracteres.
FCB	Dato de un Byte.
FDB	Dato de 2 Bytes.
RMB	Reserva Bytes de Memoria.
Definición de Símbolos	
EQU	Asigna un valor permanente.

DIRECTIVA	FUNCION
Control de Listado	
PAGE	Inicio de Página.
SPC n	Salta n espacios.
OPT NOO	No se genera un archivo objeto en cinta.
OPT O (Objet Tape)	Se genera archivo objeto(esta opción se supone por omisión).
OPT M (Memory File)	El ensamblador escribe el código de - máquina de memoria.
OPT NOM	El ensamblador no guarda el programa-objeto ensamblado.
OPT S	El ensamblador imprime los símbolos - al final de la "pasada" no. 2.
OPT NOS	No imprime los símbolos.
OPT NOL	No imprime un listado de los datos -- del ensamblador.
OPT L	Se imprime un listado de datos Ensamblador.
	Inhibe el formato de páginas del listado del ensamblador.
OPT P	El listado será paginado(selección -- por omisión).
OPT NOG	Causa que solo una línea de datos sea listada de las directivas FCC, FCB y- FCF.
OPT G	Todos los datos generados por las directivas FCC, FCB y FDF serán impre-- sos.

5.5.- MUESTRA DE UN PROGRAMA FUENTE

El siguiente programa parcial ilustra como muchas de las-reglas discutidas son aplicadas:

```
00100  NAM    RONB
00105  OPT M, S, L
00110 PIA1AC  EQU   $4005
00115 TEMP    EQU   $251
00120  ORG    $10
00125 COIN    RMB   5
00130  ORG    $20
00135 TAB FCB $10, $20, $30
00140  ORG    $30
00145 TAB1 FDB $1000, $2000, $3000
00150  PAGE
00155  ORG    $60
00160 TYPE FCC * HELP *
00165  SPC  6
00170 * THIS IS START OF MAIN PROGRAM
00175  ORG    $300
00180 NEW LDA A PIA1AC
00185  STA A  COIN
00190 AGAIN LDA B  TAB1+1
00195  STA B  COIN+1
  -
  -
00220  BRA NEW
  -
  -
  -
00300  END
```

Note que todos los estatutos originales, excepto para etiquetas y comentarios, tienen 2 espacios entre el número de línea y el estatuto fuente. Los estatutos de etiquetas y comentarios solo requieren un espacio después del número de línea.

- Línea 100.- Asigna el nombre RONB al programa.
- Línea 105.- Esas opciones causan que el ensamblador almacene el código de máquina de memoria(M), para imprimir el símbolo de tabla(S), e imprime el programa ensamblado(L).
- Línea 110.- La etiqueta PIA1AC es colocada igual al hexadecimal 4005.
- Línea 115.- La etiqueta TEMP es colocada igual al hexadecimal 251.
- Líneas 120 y 125.- Estas 2 líneas dirigen al ensamblador -- asignandole a la etiqueta COIN la dirección hexadecimal 10, a la etiqueta COIN+1 la dirección hexadecimal 11, etc. Esas etiquetas son solo usadas en el programa principal.
- Líneas 130 y 135.- Estas 2 líneas dirigen al ensamblador -- asignandole a la etiqueta TAB la dirección hexadecimal 20.- El hexadecimal 10 puede cargarse en la dirección 20, el hexadecimal 20 en la dirección 21 y el hexadecimal 30 en la dirección 22.
- Líneas 140 y 145.- Estas 2 líneas dirigen al ensamblador -- asignándole a la etiqueta TAB1 la dirección 30 y causa los-siguientes números hexadecimales a ser cargados en las localidades de memoria:

Dirección	Número hex Almacenado
30	10
31	00
32	20
33	00
34	30
35	00

- Línea 150.- El ensamblador brinca el resto de la página y empieza imprimiendo al principio de la próxima página.

- Líneas 155 y 160.- Estas 2 líneas dirigen el ensamblador asignandole a la etiqueta TYPE la dirección hexadecimal 60 y almacena el ASCII binario equivalente de HELP en localidades de memoria consecutivas empezando en la dirección hexadecimal 60 como sigue:

Dirección	Número Hex Almacenado(ASCII)	
60	48	H
61	45	E
62	4C	L
63	50	P

- Línea 165.- El ensamblador deja los próximos 6 espacios en-blanco.

- Línea 170.- Dirige al ensamblador para imprimir esa línea en el listado de salida.

- Línea 175.- Dirige al ansamblador asignandole a la etiqueta NEW la dirección hexadecimal 300 y el LDA A PIAlAC código original a la dirección hexadecimal 300, 301 y 302 como sigue:

Dirección	Contenido
300	B6
301	40
302	05(al PIAlAC se le asigno el valor 4005)

- Línea 185.- Dirige al ensamblador asignandole a el STA A COIN(modo directo) la dirección hexadecimal 303 y 304 comosigue:

Dirección	Contenido
303	97
304	10(a COIN le fué asignada la dirección 10)

- Línea 190.- Dirige al ensamblador asignándole a la etiqueta AGAIN la dirección hexadecimal 305 y LDA B TAB+1 código de-máquina la dirección 305 y 306 como sigue:

Dirección	Contenido
305	D6(TAB1 se le asignó la - dirección 30)
306	31(TAB1+1 se le asignó la dirección 31, etc.)

- Línea 195.- Dirige al ensamblador asignándole a el STA B -- COIN+1 código de máquina la dirección hexadecimal 307 y 308 como sigue:

Dirección	Contenido
307	D7
308	11(COIN+1 se le asignó la- dirección 11)

- Línea 220.- Dirige al ensamblador a calcular el número de - instrucciones sobre el cual el MPU está brincando. La salida ensamblada es como sigue:

Dirección	Contenido
309	20
30A	F5

Nota: F5 esta en complementos a 2 del hexadecimal OB(direc--- ción presente + 2 [30B] menos la dirección de la etiqueta -- NEW 300).

- Línea 300.- Terminación del programa.

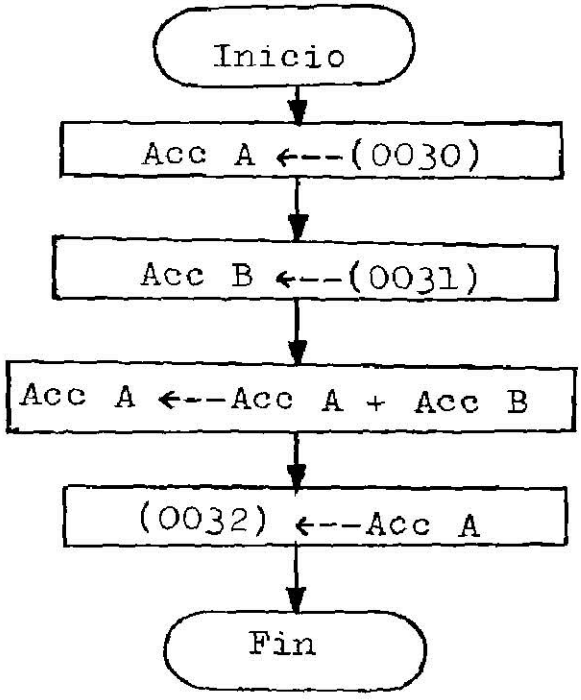
5.6.- EJEMPLOS DE PROGRAMACION

A continuación se presentan algunos programas simples que sirven de ejemplo para el uso de instrucciones del MPU y de - algunas de las facilidades proporcionadas por el Ensamblador- descrito anteriormente.

5.6.1.- SUMA DE DOS NUMEROS DE 8BITS

Efectuar la suma de 2 números de 8 bits ubicados en las - localidades 0030 y 0031 cada uno. El resultado puede ser depo sitado en la dirección 0032.
Supongase que la suma no excede la capacidad de representa--- ción de un byte para números expresados en complemento a 2.

Diagrama de Flujo:



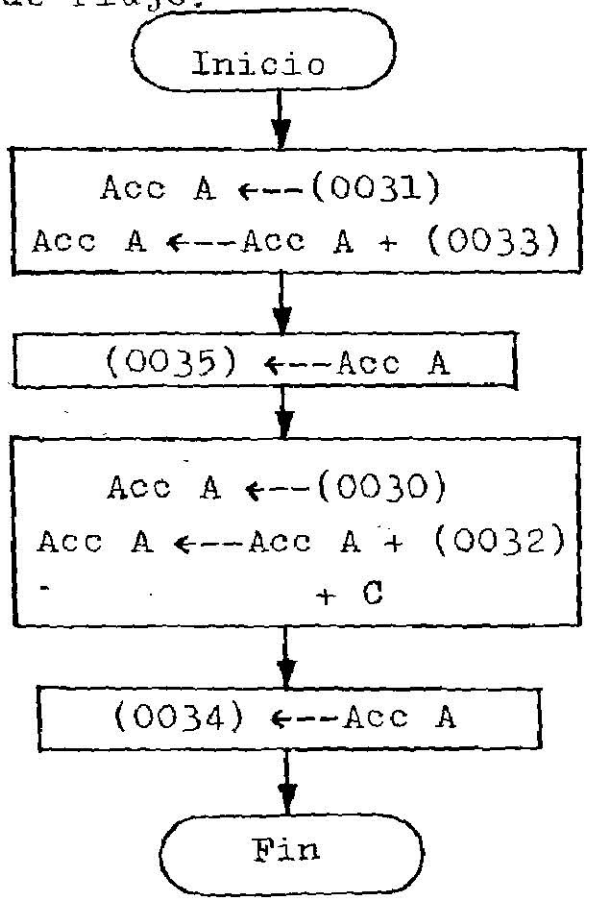
Programa en Lenguaje Ensamblador y Codificación.

Dirección	Código			
			ORG $0000	
0000	96	30	LDA A $30	+ Obtiene el 1er sumando
0002	D6	31	LDA B $31	+ Obtiene el 2o. sumando
0004	1B		ABA	+ Realiza la suma
0005	97	32	STA A $32	+ Deposita el resultado
0007	3F		SWI	+ Alto

5.6.2.- SUMA DE DOS NUMEROS DE 16 BITS

Se requiere efectuar la suma de 2 números de 16 bits cada uno, que estarán formados por 2 bytes. Los operandos estarán ubicados en localidades consecutivas a partir de la dirección 0030. El resultado de 2 bytes deberá depositarse a partir de la localidad 0034.

Diagrama de Flujo:



+ Suma los 8 bits menos significativos.

+ Guarda el resultado de menor peso.

+ Suma los 8 bits más -- significativos.

+ Guarda el resultado de mayor peso.

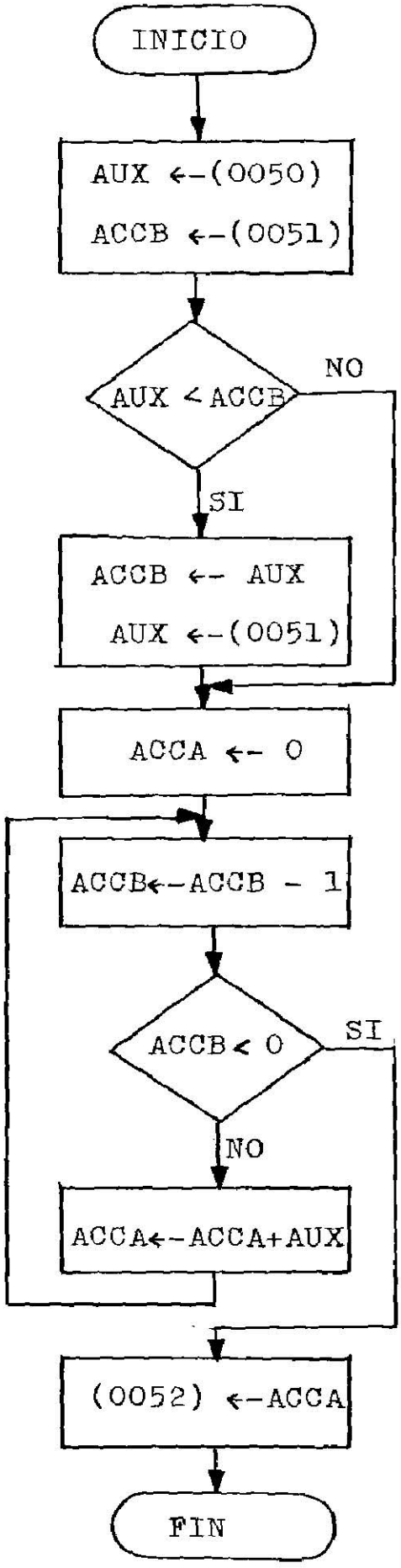
Programa en Lenguaje Ensamblador y Codificación.

```
Dirección   Código
                        ORG   $0000
  0000      96  31      LDA A $31   + Obtiene la parte baja del
                                      1er.
  0002      9B  33      ADD A $33   + sumando y le suma la del
                                      2o.
  0004      97  35      STA A $35   + Guarda el resultado de -
                                      menor peso.
  0006      96  30      LDA A $30   + Obtiene la parte alta del
                                      1er.
  0008      99  32      ADC A $32   + sumando y le suma la del
                                      2o. con acarreo.
  000A      97  34      STA A $34   + Guarda el resultado de -
                                      mayor peso.
  000C      3F          SWI         + Alto.
```

5.6.3.- MULTIPLICACION DE DOS NUMEROS DE 8 BITS

Multiplicar 2 números enteros positivos de 8 bits ubica-- dos en localidades de memoria consecutivas a partir de la dirección 0050. El algoritmo a utilizar, en este caso, será el- de efectuar la suma de un operando tantas veces el valor del- otro. Se debe tratar de reducir el número de sumas eligiendo- el menor operando para contar las veces que se suma el otro.

Diagrama de Flujo: Se muestra en la figura 5.6, describiendo- las acciones a efectuar con diferentes registros y localida-- des de memoria para llevar a cabo lo especificado en el enunciado. Se define una variable AUX1 que contiene el operando - mayor.



+ Se toman los operandos.

+ Se pregunta por el menor para dejarlo en ACCB.

+ Se realizan las sumas.

+ Se guarda el resultado.

Figura 5.6

Programa en Lenguaje Ensamblador y Codificación.

```
Dirección  Código
                                ORG     $0000
  0000                  AUX     RMB1
  0001      96  50              LDA A   $50     + Lectura de
  0003      97  53              STA A   AUX     + los operandos
  0005      D6  51              LDA B   $51
  0007      D1  53              CMP B   AUX     + Pregunta por el
  0009      23  06              BLS   NOCAMB    + menor para contar
  000B      D6  53              LDA B   AUX     + con él
  000D      96  51              LDA A   $51
  000F      97  53              STA A   AUX
  0011      4F          NOCAMB  CLR A
  0012      5A          CUEN    DEC B           + Ciclo para las
  0013      2B  04              BMI   FIN       + sumas
  0015      9B  53              ADD A   AUX
  0017      20  F9              BRA   CUEN
  0019      97  52  Fin         STA A   $52     + Se guarda el resul-
                                                  tado.
  001B      3F                  SWI
```

5.6.4.- TRANSFERIR UN BLOQUE DE BYTES

Se necesita transferir un bloque de bytes de un lugar a otro de la memoria. Se dan las direcciones inicial y final (DINIC y DFINAL respectivamente) del arreglo a mover y la dirección a partir de la cual se debe cargar dicho bloque(DIDEST).

Diagrama de Flujo: Es el mostrado en la figura 5.7. Se utilizan 2 variables auxiliares a manera de apuntadores(cada uno de 2 bytes) que contienen las direcciones fuente y destino del byte a transferir.

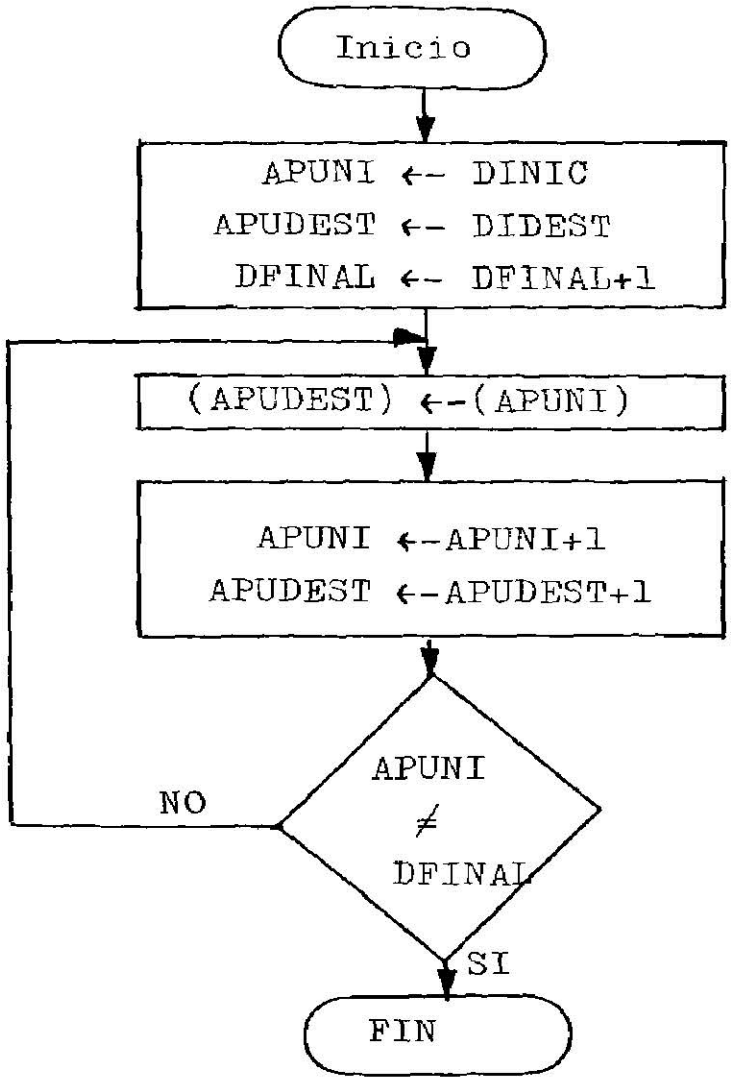


Figura 5.7

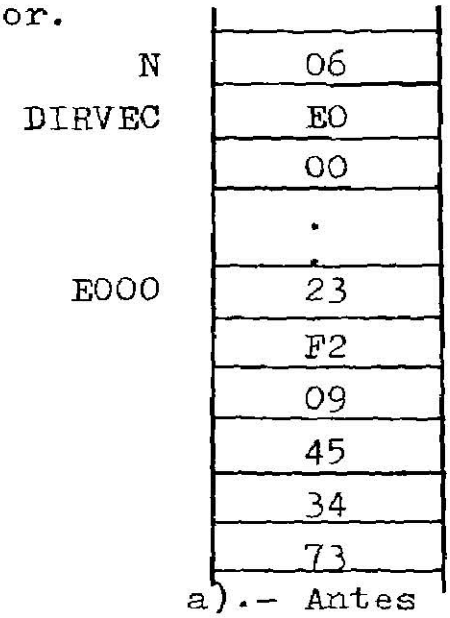
Programa en Lenguaje Ensamblador y Codificación.

```
                        ORG    $0000
0000            DINC    RMB2
0002            DFINAL  RMB2
0004            DDESTI  RMB2
0006            APUINI  RMB2
0008            APUDEST RMB2
001A DE  00             LDX    DINIC     APUINI -- DINIC
001C DF  06             STX    APUINI
001E DE  06             LDX    DDESTI    APUDEST --DDESTI
0020 DF  08             STX    APUDEST
0022 DE  02             LDX    DFINAL    DFINAL -- DFINAL + 1
```

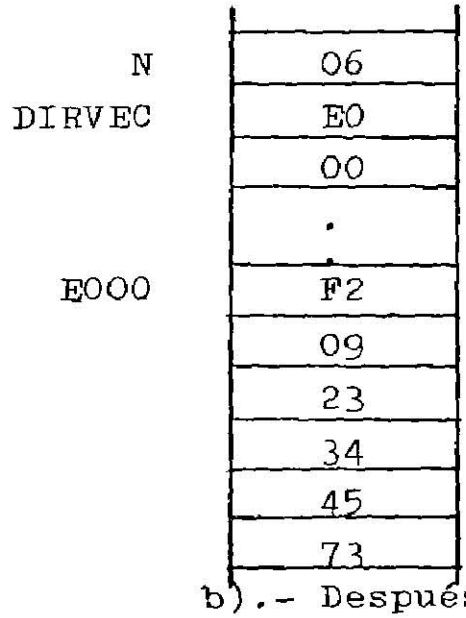
```
0024 08                INX
0025 DF  02            STX  DFINAL
0027 DE  06  CAMBIO    LDX  APUNI     (APUDEST) -- (APUNI)
0029 A6  00            LDA A 00,X
002B DE  06            LDX  APUDEST
002D A7  00            STA A 00,X
002F DE  08            LDX  APUDEST   APUDEST -- APUDEST + 1
0031 08                INX
0032 DF  08            STX  APUDEST
                  *
0034 DE  06            LDX  APUNI     APUNI -- APUNI + 1
0036 08                INX
0037 DF  06            STX  APUNI
                  *
0039 9C  02            CPX  DFINAL    APUNI   DFINAL
003B 26  EA            BNE  CAMBIO
003D 3F                SWI            ALTO
```

5.6.5.- ORDENAMIENTO DE UN ARREGLO

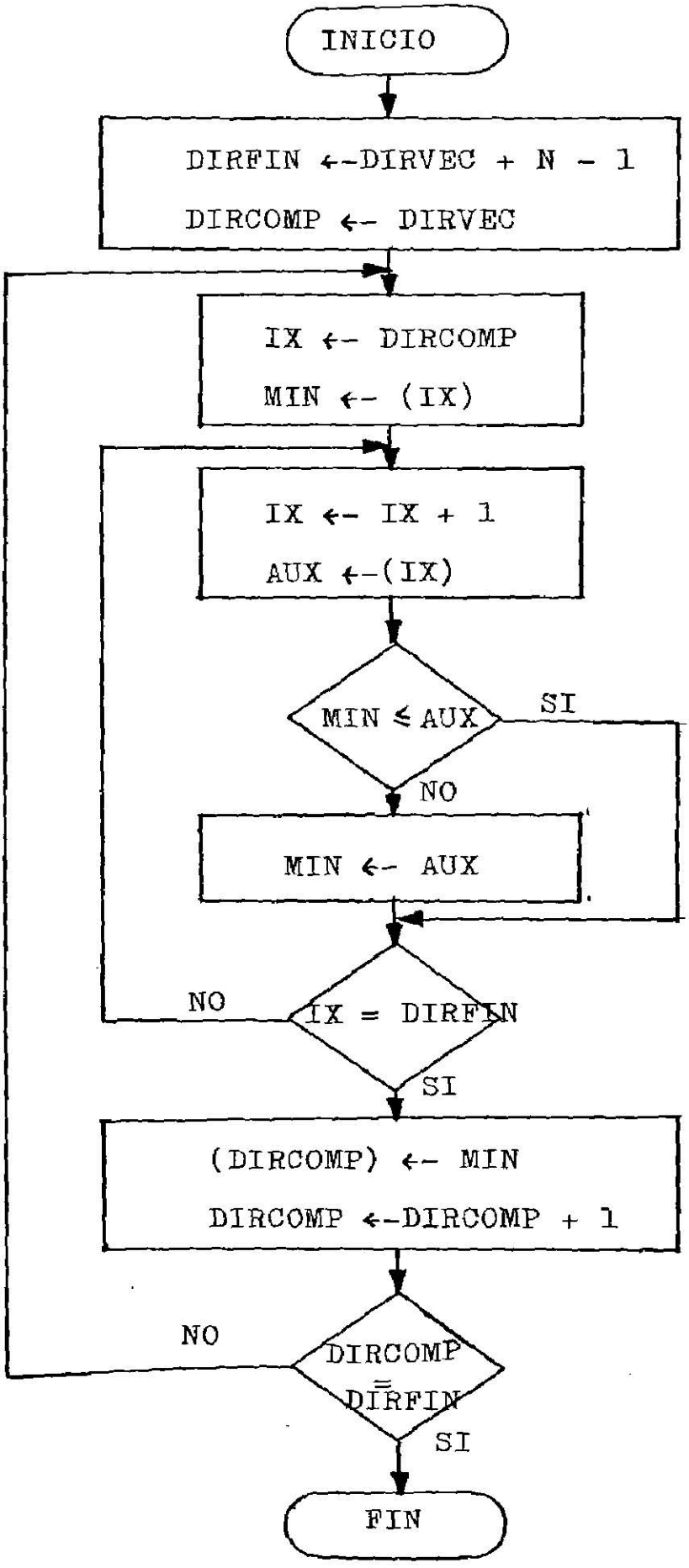
Se requiere ordenar un conjunto de números enteros de un-byte en forma creciente: dichos números están ubicados en localidades consecutivas formando un vector. La dirección inicial del arreglo estará contenida en dos bytes el primero apuntado por DIRVEC; N contiene el número de elementos del vector.



a).- Antes



b).- Después


INICIO
DIRFIN ←-DIRVEC + N - 1
DIRCOMP ←- DIRVEC
IX ←- DIRCOMP
MIN ←- (IX)
IX ←- IX + 1
AUX ←-(IX)
MIN ≤ AUX
SI
NO
MIN ←- AUX
NO
IX = DIRFIN
SI
(DIRCOMP) ←- MIN
DIRCOMP ←-DIRCOMP + 1
NO
DIRCOMP ≡ DIRFIN
SI
FIN

Programa en Lenguaje Ensamblador.

```
                ORG   $0000

DIRVEC    RMB       2
DIRFIN    RMB       2
DIRCOM    RMB       2
MIN       RMB       2
          LDA A     N          * DIRFIN --DIRVEC + N - 1
          DEC A
          ADD A     DIRVEC + 1
          STA A     DIRFIN + 1
          LDA A     DIRVEC
          ADC A     #$00
          STA A     DIRFIN

          LDX DIRVEC
          STX DIRCOMP          * DIRCOMP -- DIRNEC
INIC      LDX DIRCOMP          * IX -- DIRCOMP
          LDA A     00,X       * MIN -- IX
          STA A     MIN
BUSMIN    INX                  * IX -- IX + L
          LDA B     00,X       * AUX -(1X); (AUX=ACCB)
          CMP B     MIN        * MIN   ACCB ?
          BPL NOCAMB           *
          STA B     MIN        * MIN -- ACCB
NOCAMB    CPX DIRFIN           * IX = DIRFIN
          BNE BUSMIN
          LDX DIRCOMP          * (DIRCOMP) -- MIN
          LDA A     MIN
          STA A     00,X
          INX                  * DIRCOMP --DIRCOMP + 1
          STX DIRCOMP
          CPX DIRFIN           * DIRCOMP = DIRFIN ?
          BNE INIC
          SWI
```

A P E N D I C E I

PROGRAMAS DE EJEMPLOS

PROGRAMA QUE LIMPIA LAS LOCALIDADES DE MEMORIA 00_{16} A FF_{16}

Proposito: Este programa muestra como grandes bloques de memoria pueden ser limpiados(en este caso, las localidades de 00_{16} a FF_{16}).

Programa Fuente:

```
        NAM     CLM
        ORG     $500
        LDX     #$00
        CLR A
AGAIN   STA A   0,X
        INX
        CPX     #$100
        BNE     AGAIN
        SWI
```

Programa Objeto:

Dirección de Memoria (Hex)	Contenido de Memoria (Hex)	
0500	CE	0000
0503	4F	
0504	A7	00
0506	08	
0507	8C	0100
050A	26	F8
050C	3F	

PROGRAMAS QUE CARGAN LA MEMORIA CON TABLAS DE DATOS

- ASCENDENTE
- DESCENDENTE

Proposito: Estos programas ilustran como las tablas de datos-mostradas a continuación pueden ser cargadas en -- las localidades de memoria.

Tabla Ascendente		Tabla Descendente	
Dirección	Dato	Dirección	Dato
0000	00	0000	FF
0001	01	0001	FE
0002	02	0002	FD
0003	03	0003	FC
.	.	.	.
.	.	.	.
.	.	.	.
00FD	FD	00FD	02
00FE	FE	00FE	01
00FF	FF	00FF	00

Programa Fuente (Modo Ascendente):

```
        NAM     STT
        ORG     $200
        LDX     #$00
        CLR A
NEXT    STA A   0,X
        INC A
        INX
        CPX     #$100
        BNE     NEXT
        SWI
```

Programa Objeto (Modo Ascendente):

Dirección de Memoria (Hex)	Contenido de Memoria (Hex)	
0200	CE	0000
0203	4F	
0204	A7	00
0206	4C	
0207	08	0100
020A	8C	
020B	26	F9
020D	3F	

Programa Fuente (Modo Descendente):

```
        NAM   STT
        ORG   $300
        LDS   #$FF
        CLR A
AGAIN   PSH A
        INC A
        BNE   AGAIN
        SWI
```

Programa Objeto (Modo Descendente):

Dirección de Memoria (Hex)	Contenido de Memoria (Hex)	
0300	8E	00FF
0303	4F	
0304	36	
0305	4C	
0306	26	FC
0308	3F	

PROGRAMA ADICION DE MULTIPLE PRECISION

Proposito: Sumar 2 números binarios de bytes múltiples. La -- longitud de los números (en Bytes) está en el acumulador B, la dirección de inicio de los números - está en las localidades de memoria 0042 y 0043, y- 0044 y 0045, y la dirección de inicio del resultado está en las localidades de memoria 0046 y 0047. Todos los números empiezan con los bits menos significativos.

Problema de Ejemplo:

Acc B = 04
(0042 y 0043) = 0048
(0044 y 0045) = 004C
(0046 y 0047) = 0050
(0048) = 30
(0049) = 4A
(004A) = 52
(004B) = 3A
(004C) = F8
(004D) = 25
(004E) = C0
(004F) = 1B

Resultado: (0050) = 28
(0051) = 70
(0052) = 12
(0053) = 56

o sea,

```
  3A524A30
+ 1BC025F8
  56127028
```

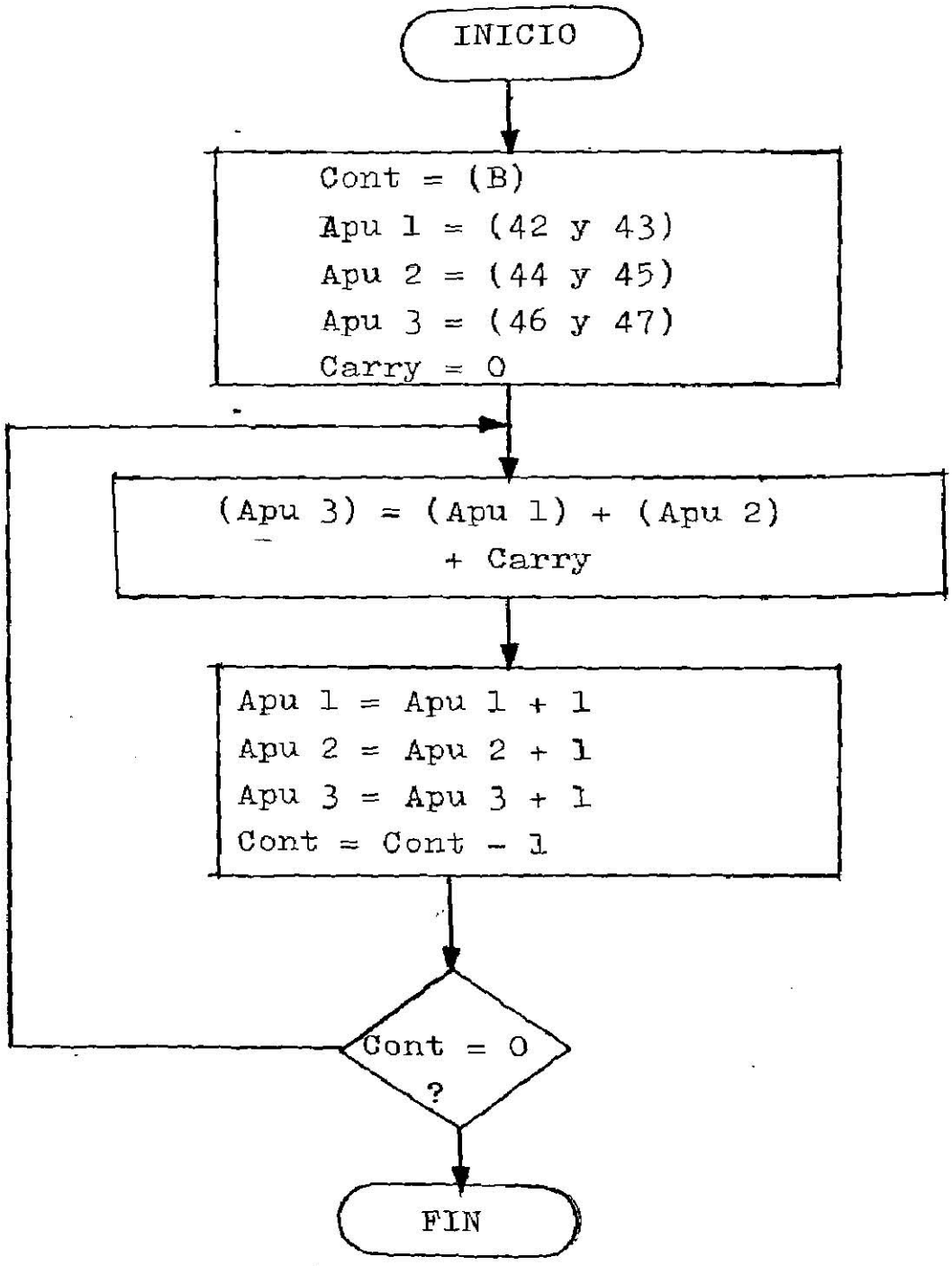


Diagrama de Flujo

Programa Fuente: El programa principal inicia con la Pila (Stack) en la localidad 0058, coloca las direcciones iniciales de los números --- 0048, 004C y 0050, respectivamente, ob-- tiene la longitud de los números desde - la localidad 0040, y llama a la subruti- na de Adición de Múltiple Precisión.

```
        ORG     0
        STS     $3E       SAVE MONITOR STACK POINTER
        LDS     #$58      START STACK AT LOCATION 0058
        LDX     #$48      GET STARTING ADDRESS OF FIRST NUMBER
        STX     $42
        LDX     #$4C      GET STARTING ADDRESS OF SECOND NUMBER
        STX     $44
        LDX     #$50      GET STARTING ADDRESS OF RESULT
        STX     $46
        LDA B   $40       GET LENGTH OF NUMBERS
        JSR     MPADD
        LDS     $3E       RESTORE MONITOR STACK POINTER
        SWI

Subrutina:
        ORG     $20
MPADD   CLC               CLEAR CARRY TO START
MP1     LDX     $42
        LDA A   X         GET BYTE FROM FIRST NUMBER
        INX
        STX     $42
        LDX     $44
        ADC A   X         ADD BYTE FROM SECOND NUMBER
        INX
        STX     $44
        LDX     $46
```

```
        STA A  X        STORE RESULT
        INX
        STX    $46
        DEC B
        BNE    MP1
        RTS
```

Programa Objeto:

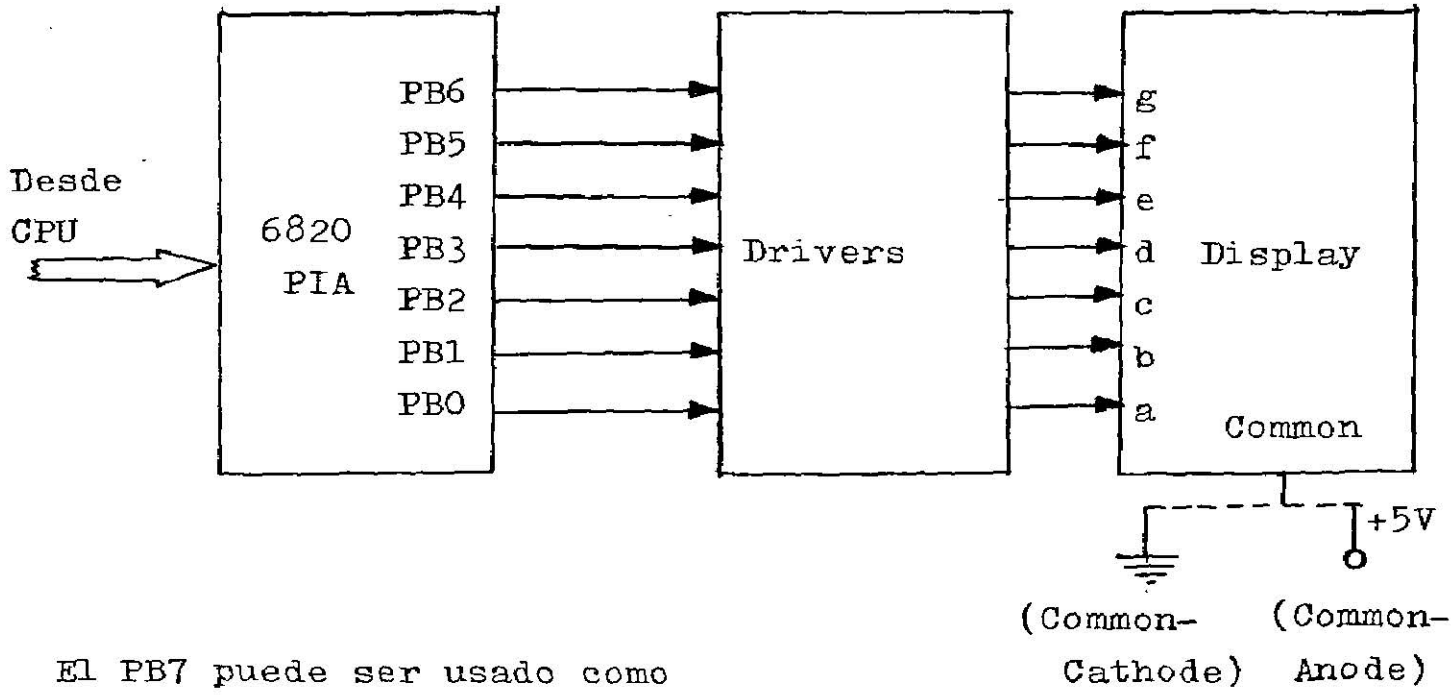
Dirección de Memoria (Hex)	Contenido de Memoria (Hex)
1)Programa Principal	
0000	9F 3E
0002	8E 0058
0005	CE 0048
0008	DF 42
000A	CE 004C
000D	DF 44
000F	CE 0050
0012	DF 46
0014	D6 40
0016	BD 0020
0019	9E 3E
001B	3F
2)Subrutina	
0020	0C
0021	DE 42
0023	A6 00
0025	08
0026	DF 42
0028	DE 44
002A	A9 00
002C	08
002D	DF 44
002F	DE 46
0031	A7 00

Dirección de Memoria (Hex)	Contenido de Memoria (Hex)
0033	08
0034	DF 46
0036	5A
0037	26 E8
0039	39

PROGRAMA DESPLIEGUE DEL LED DE 7-SEGMENTOS

Proposito: Desplegar la interface de un LED de 7-segmentos en el MPU 6800. Este despliegue puede estar en el modo de Common-Anode(Negativo Lógico) o el modo de - Common-Cathode(Positivo Lógico).

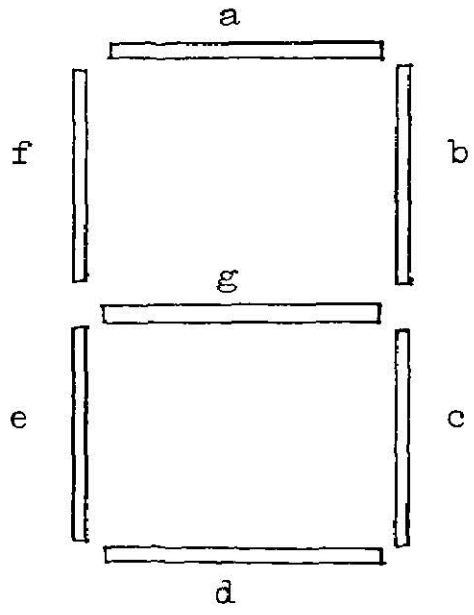
La figura I muestra la circuitería para la inter-- face en un despliegue de 7-segmentos. Cada segmento puede tener uno, dos o más LEDs juntos en un -- mismo lugar. Hay 2 modos de conectar los despliegues. Una es ligando todos los cátodos juntos a -- tierra(Fig. IIa); este es un despliegue "common--- catode" un uno lógico en un anodo ilumina un seg-- mento. El otro modo es ligando todos los ánodos -- con un suministro de voltaje positivo(fig. IIb); - este es un despliegue "common-anode", y un cero -- lógico en un cátodo ilumina un segmento. Así, el - despliegue del common-anode usa lógica negativa y- el despliegue del common-cathode usa lógica positiva.



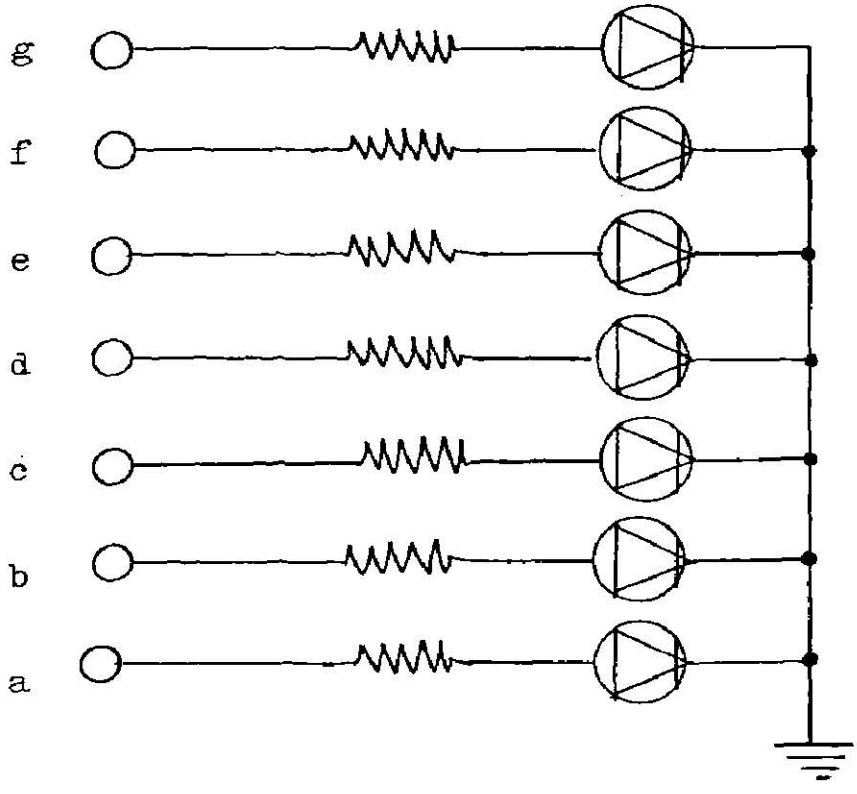
El PB7 puede ser usado como punto decimal.

Figura I.- Despliegue de Interface de 7- Segmentos

La línea Common del despliegue es ligada a tierra o a +5V. Los segmentos desplegados son usualmente etiquetados.



a).- Common-Cathode



b).- Common-Anode

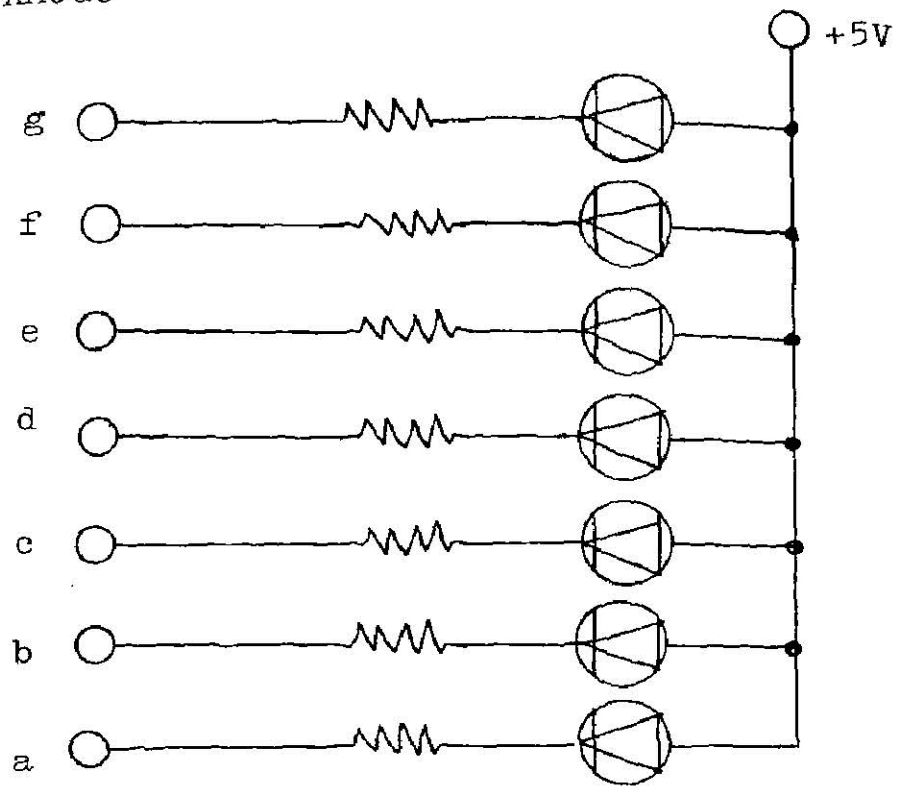
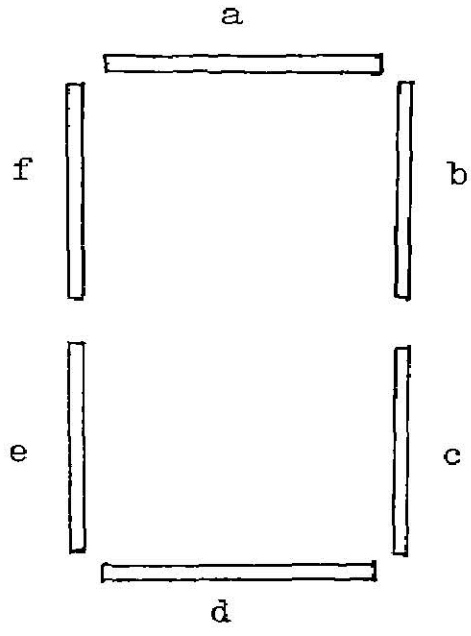


Figura II.- Organización de Despliegue de 7-Segmentos

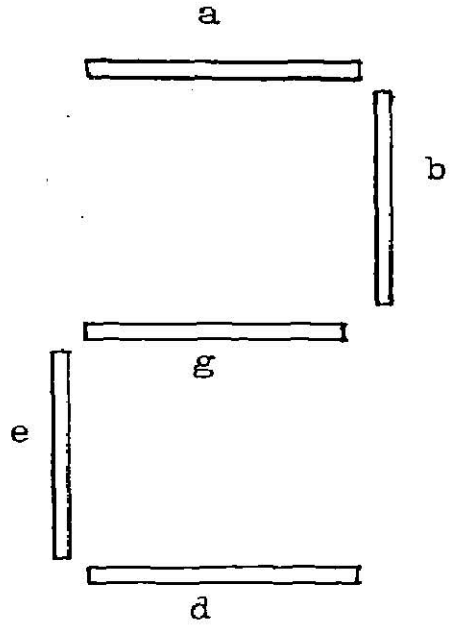
Nota: El despliegue de 7-segmentos es ampliamente usado por - que contiene el más pequeño número de segmentos controlados - separadamente que pueden suplir reconocidas representaciones- de todos los dígitos decimales(Fig. III y Tabla I). Los des-- pliegues de 7-segmentos pueden también producir algunas le--- tras y otros caracteres(Tabla II). Mayores representaciones - requieren un número sustancialmente grande de segmentos y más circuitos. Dado que los despliegues de 7-segmentos son tan po pulares, el bajo costo del decodificador/manejador de 7-seg-- mentos tiende a estar disponible. Los más populares de estos- dispositivos son el conductor 7447 Common-Anode y el conduc-- tor 7448 Common-Cathode; estos dispositivos tienen Lamp Test- Input(el cual coloca todos los segmentos en "on") y blanquea- las entradas y salidas(para blanquear lleva o arrastra ceros)

0:Segments f, e, d, c, b, a <u>on</u>

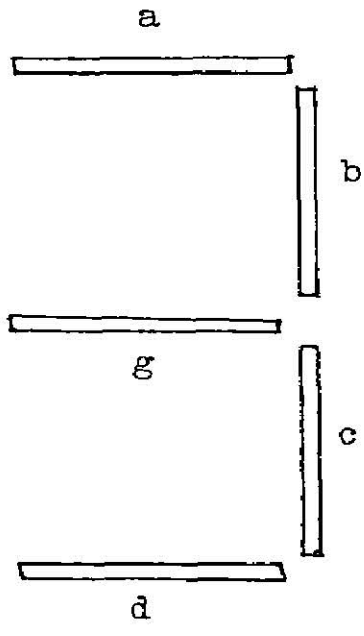


1:Segments c, b <u>on</u>

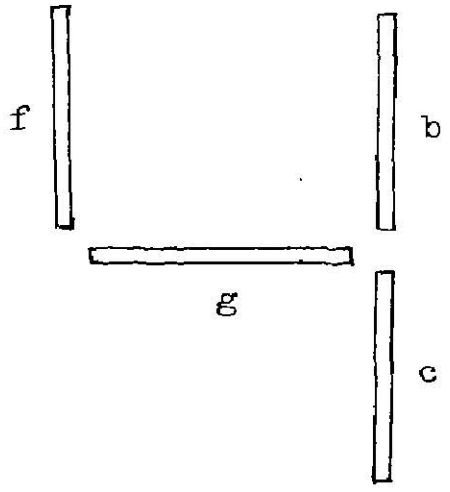
2:Segments g,e,d,b,a <u>on</u>



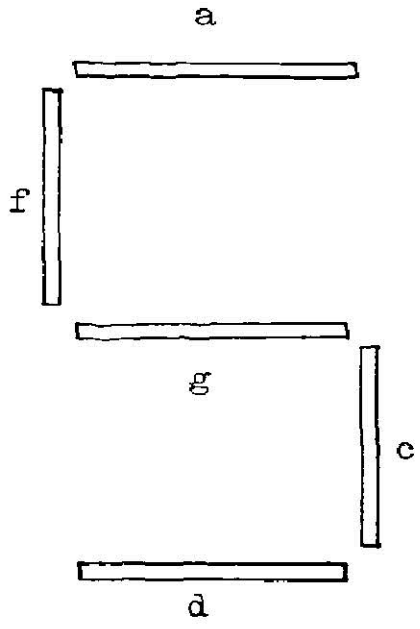
3:Segments g,d,c,b,a <u>on</u>



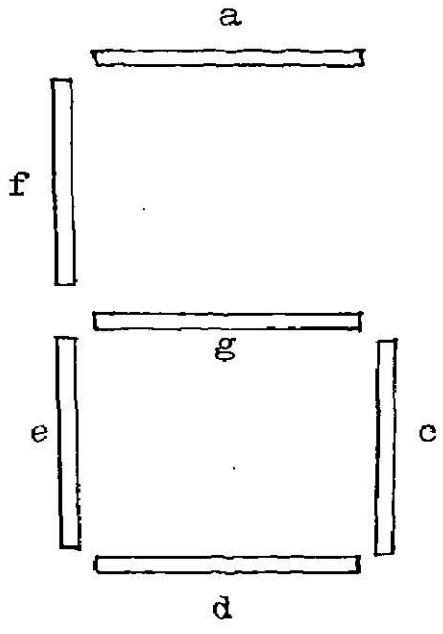
4: Segment g, f, c, b on



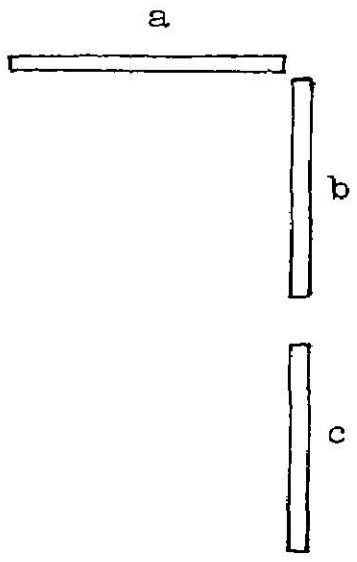
5: Segments g, f, d, c, a on



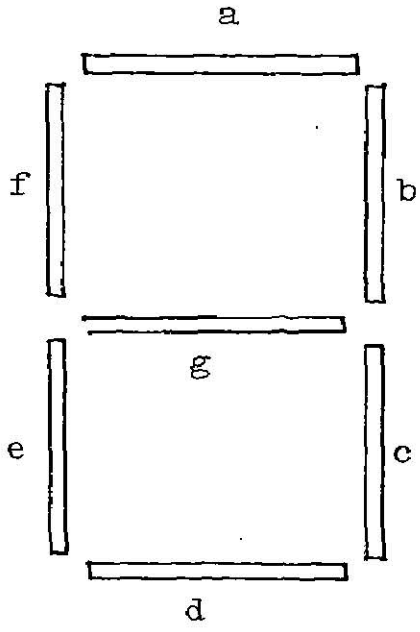
6: Segments g, f, e, d, c, a on



7: Segments c, b, a on



8:Segments g,f,e,d,c,b,a on



9:Segments g,f,c,b,a on

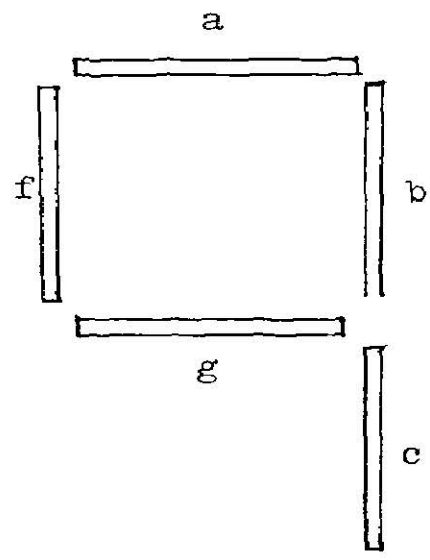


Figura III.- Representación en 7-Segmentos de Dígitos Decimales

	Representación Hexadecimal	
Numero	Common-Cathode	Common-Anode
0	3F	40
1	06	79
2	5B	24
3	4F	30
4	66	19
5	6D	12
6	7D	02
7	07	78
8	7F	00
9	67	18

El bit 7 es siempre 0 y los otros bits g,f,e,d,c,b, y a están en orden decreciente de significancia.

Tabla I.- Representación en 7-Segmentos de los Números Decimales

Letras Mayúsculas

Letra	Representación Hexadecimal Common-Cathode	Common-Anode
A	77	08
C	39	46
E	79	06
F	71	0E
H	76	09
I	06	79
J	1E	61
L	38	47
O	3F	40
P	73	0C
U	3E	41
Y	66	19

Letras Minúsculas y Caracteres Especiales

Caracter	Representación Hexadecimal Common-Cathode	Common-Anode
b	7C	03
c	58	27
d	5E	21
h	74	0B
n	54	2B
o	5C	23
r	50	2F
u	1C	63
-	40	3F
?	53	2C

Tabla II.- Representación en 7-Segmentos de Letras y Símbolos

Programa de Ejemplo.

Proposito: Desplegar el contenido de la localidad de memoria-0041 en despliegues de 7-segmentos si contiene un-dígito decimal. De otro modo, despliega blancos.

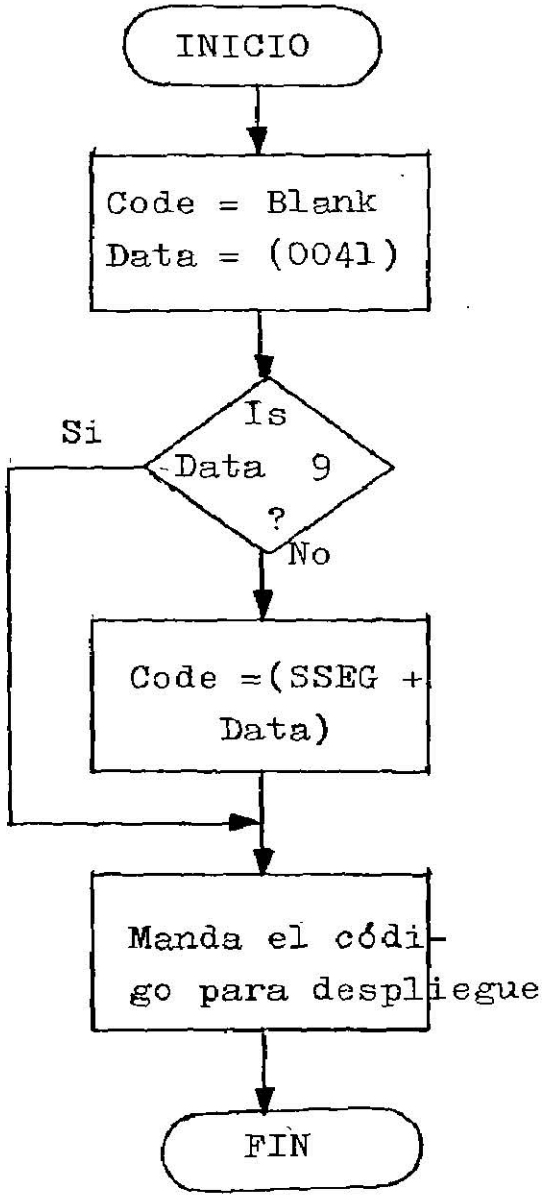


Diagrama de Flujo

Programa Fuente:

```
        CLR     PIACRB      CLEAR OUT CONTROL REGISTER
        LDA A   #$FF        ALL LINES OUTPUTS
        STA A   PIADDRB
        LDA A   #%00000100  ACCESS DATA REGISTER
        STA A   PIACRB
        LDA B   #BLANK      GET BLANK CODE
        LDA A   $41         GET DATA
        CMP A   #9          IS DATA GREATER THAN 9?
        BHI     DSPLY       YES, DISPLAY BLANKS
        CLR     $40
        LDX     $40         GET OFFSET ADDRESS FOR TABLE
        LDA B   SSEG,X      GET SEVENT-SEGMENT CODE
DSPLY   STA B   PIADRB      CODE TO DISPLAY
        SWI
```

Programa Objeto:

Dirección de Memoria	Contenido de Memoria	
0000	7F	PIACRB
0003	86	FF
0005	B7	PIADDRB
0008	86	04
000A	B7	PIACRB
000D	C6	BLANK
000F	96	41
0011	81	09
0013	22	07
0015	7F	0040
0018	DE	40
001A	E6	20
001C	F7	PIADRB
001F	3F	
0020-0029		SSEG

Dirección de Memoria	Contenido de Memoria
001A	08
001B	5A
001C	26 F4
001E	20 ED

Ejemplo;

(0040) = 66
(0041) = 3F
(0042) = 7F
(0043) = 7F
(0044) = 06
(0045) = 5B
(0046) = 07
(0047) = 4F
(0048) = 6D
(0049) = 7D

Despliegue 4088127356

PROGRAMA DEL RELOJ DIGITAL

Proposito: Este programa muestra en el display un reloj de -- tiempo real en horas, minutos y segundos.

Programa Fuente:

```
            NAM     TIME
            OPT     LLEN=80,CREF
            ORG     $00
      * RUTINA D5BUG
       PUT  EQU     $F0BB
     MNPTR  EQU     $E419
    HEXBUF  EQU     $E42C
    DISCOD  EQU     $F120
       RHR  RMB     1          HORAS
      RMIN  RMB     1          MINUTOS
      RSEC  RMB     1          SEGUNDOS
      TICK  RMB     2          CONTADOR-PASOS; CASI 1ms/PASO
       *
     START  LDX     #TIME      DIRECCION DE LA RUTINA TIME
            STX     MNPTR      ESTABLECE COMO ACTIVO SUB
            JMP     PUT        OBTIENE EL COMIENZO
       *
      TIME  LDX     TICK      # DE PASOS
            INX               INCREMENTA EL # DE PASOS
            STX     TICK      ACTUALIZA
            CPX     #535      PASOS/SEGUNDO
            BNE     TBOT      ABAJO DEL PROGRAMA
            LDX     #$0000
            STX     TICK      COLOCA EL CONTADOR TICK
            INC     RSEC      INCREMENTA LOS SEGUNDOS
            LDA A   RSEC
            ORA A   #$F5      CHECA EL SOBREFLUJO EN BCD
            COM A
            BNE     TBOT      SI ES NO $xA SEGUNDOS
            LDA A   RSEC
            ADD A   #6        INCREMENTA DIEZ DIGITOS
```

```
        STA A   RSEC     ACTUALIZA
        CMP A   #$60     60 SEGUNDOS AUN?
        BNE     TBOT     SI ES NO; EXITO
        CLR     RSEC     COLOCA EL CONTADOR DE SEGUNDOS
        INC     RMIN     NUEVO MINUTO
        LDA     RMIN
        ORA A   #$F5     CHECA EL SOBREFLUJO EN BCD
        COM A
        BNE     TBOT     SI ES NO; EXITO
        LDA A   RMIN
        ADD A   #6       INCREMENTA DIEZ DIGITOS
        STA A   RMIN     ACTUALIZA
        CMP A   #$60     60 MINUTOS AUN?
        BNE     TBOT     SI ES NO; EXITO
        CLR     RMIN     COLOCA EL CONTADOR DE MINUTOS
        INC     RHR      NUEVA HORA
        LDA A   RHR
        ORA A   #$F5     CHECA SOBREFLUJO EN BCD
        COM A
        BNE     CHKHR    CHECA EL FINAL DEL DIA
        LDA A   RHR
        ADD A   #6       INCREMENTA DIEZ DIGITOS
        STA A   RHR      ACTUALIZA LAS HORAS
CHKHR   LDA A   RHR
        CMP A   #$24     FIN DE UN DIA?
        BNE     TBOT
        CLR     RHR      COLOCA LAS HORAS
 TBOT   LDX     RHR      HORAS/MINUTOS
        STX     HEXBUF   AL BUFFER HEX
        LDA A   RSEC     SEGUNDOS
        STA A   HEXBUF+2 AL ULTIMO DIGITO
        JMP     DYSCOD   CONVERSION A 7-SEGMENTOS
* PUEDE RETORNAR PARA COLOCARSE EN LA CONCLUSION DEL
  DYSCOD
        END
```

Programa Objeto:

Dirección de Memoria	Contenido de Memoria	
0005	CE	000E
0008	FF	E419
000B	7E	F0BB
000E	DE	02
0010	08	
0011	DF	03
0013	8C	0217
0016	26	4C
0018	CE	0000
001B	DF	03
001D	7C	0002
0020	96	02
0022	8A	F5
0024	43	
0025	26	3D
0027	96	02
0029	8B	06
002B	97	02
002D	81	60
002F	26	33
0031	7F	0002
0034	7C	0001
0037	96	01
0039	8A	F5
003B	43	
003C	26	26
003E	96	01
0040	8B	06
0042	97	01
0044	81	60
0046	26	1C
0048	7F	0001
004B	7C	0000

Dirección de Memoria	Contenido de Memoria
004E	96 00
0050	8A F5
0052	43
0053	26 06
0055	96 00
0057	8B 06
0059	97 00
005B	96 00
005D	81 24
005F	26 03
0061	7F 0000
0064	DE 00
0066	FF E42C
0069	96 02
006B	B7 E42E
006E	7E F120

A P E N D I C E II

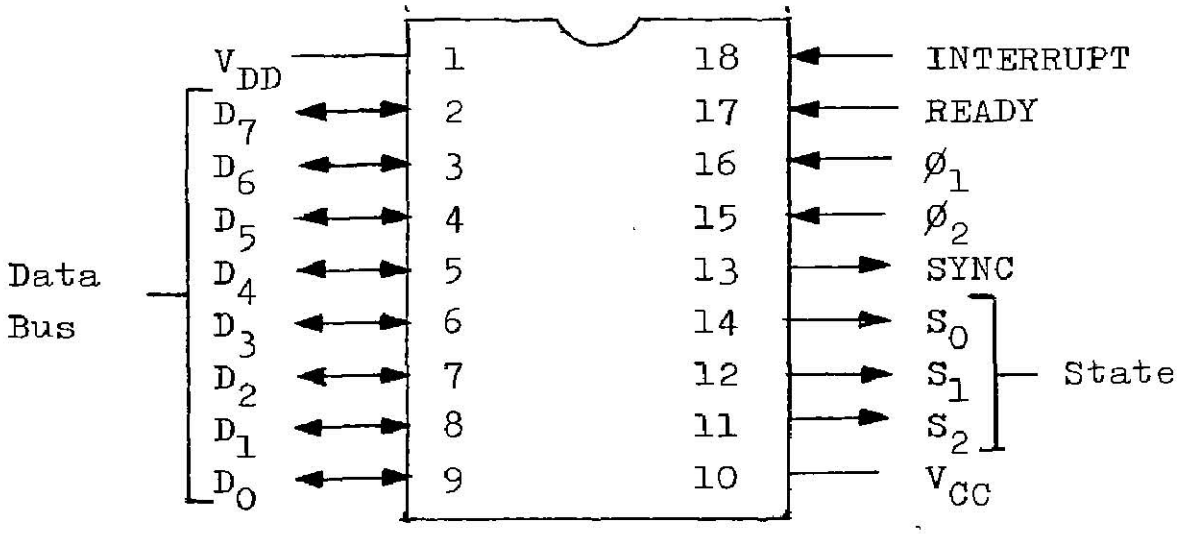
ALGUNAS FAMILIAS

DE

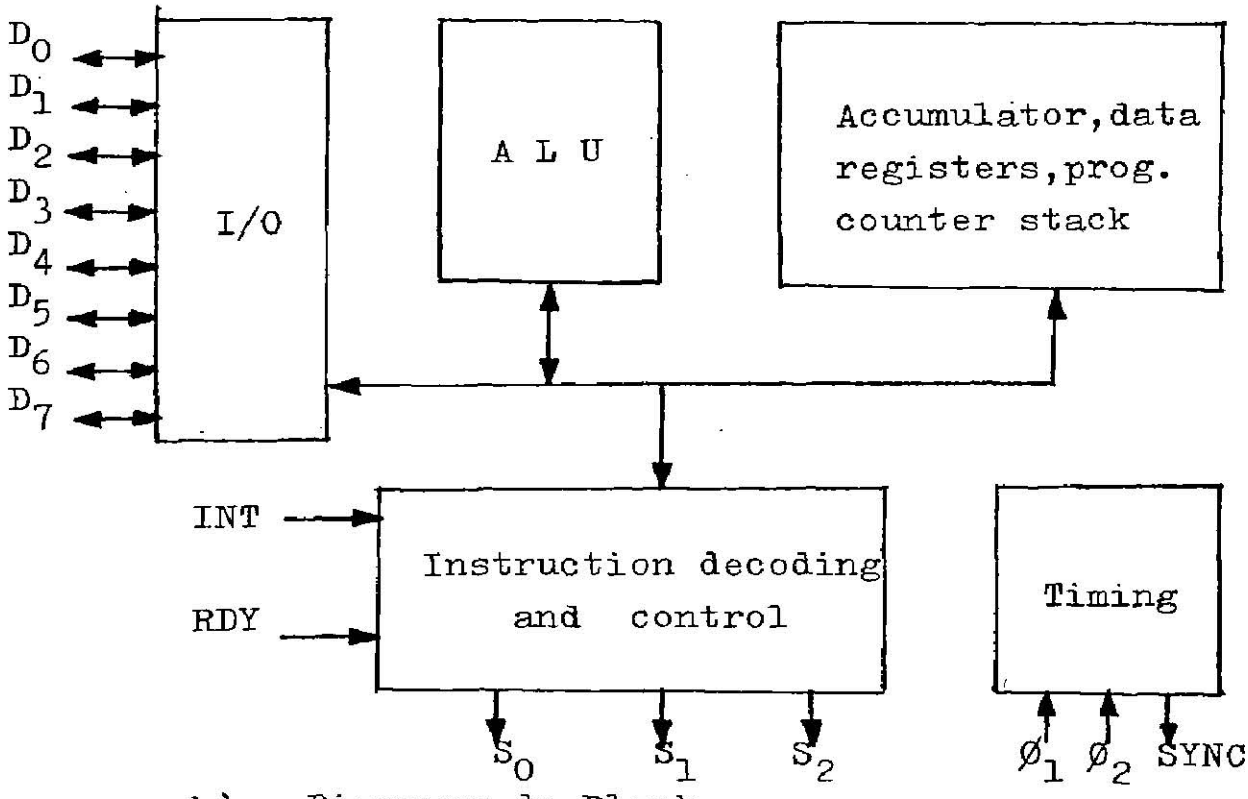
MICROPROCESADORES

EL MICROPROCESADOR 8008

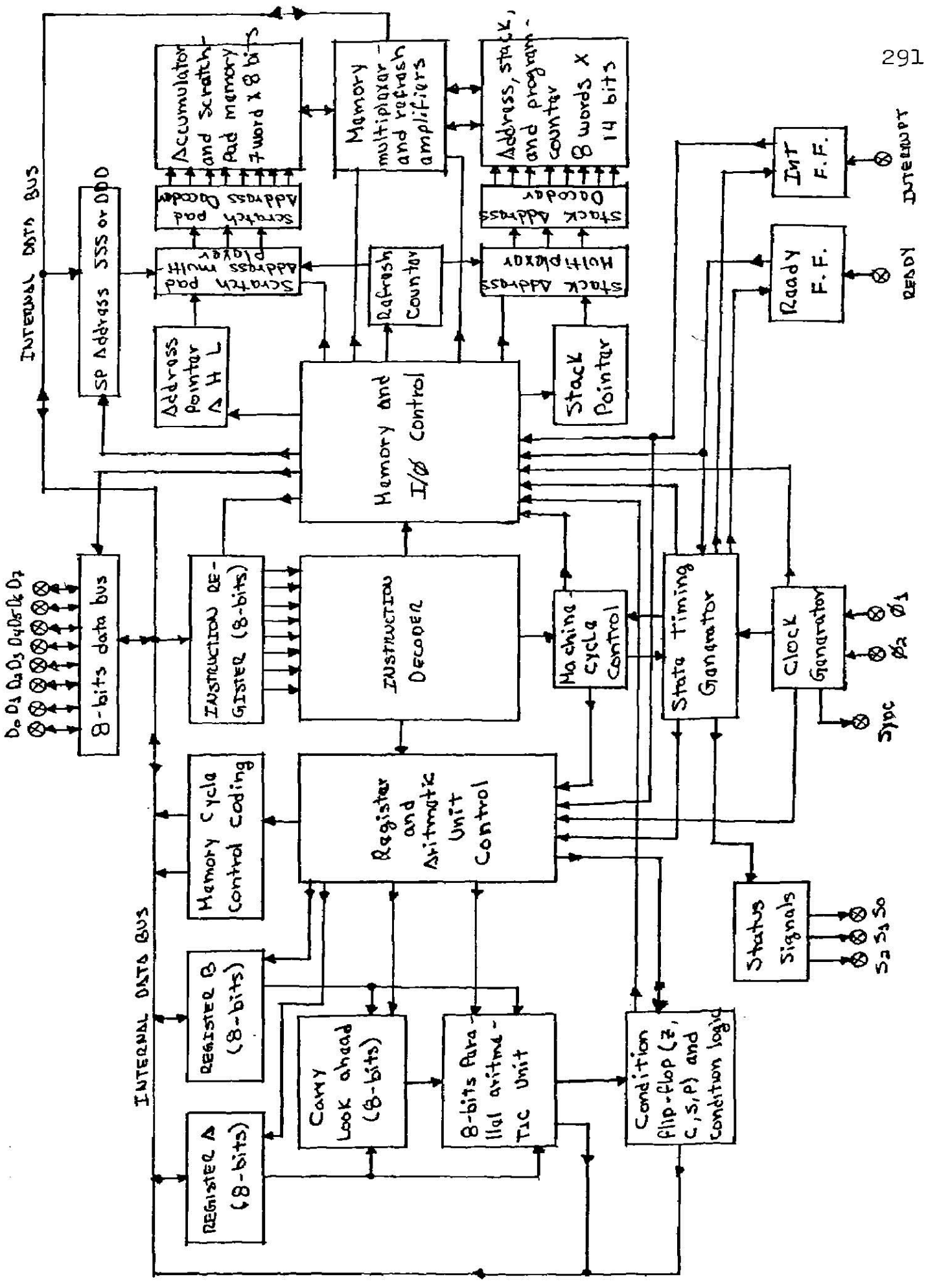
El 8008 es un microprocesador de 8 bits y arquitectura muy -- simple, tiene un contador de programas(PC) de 14 bits, lo -- cual le permite accesar solo 16 k bytes de memoria. El manejo de sus registros es sencillo y su conjunto de instrucciones - limitado, por lo que fué rápidamente superado por su descen-- diente directo el 8080.



a).- Configuración del Pin



b).- Diagrama de Block



Arquitectura del MPU 8008

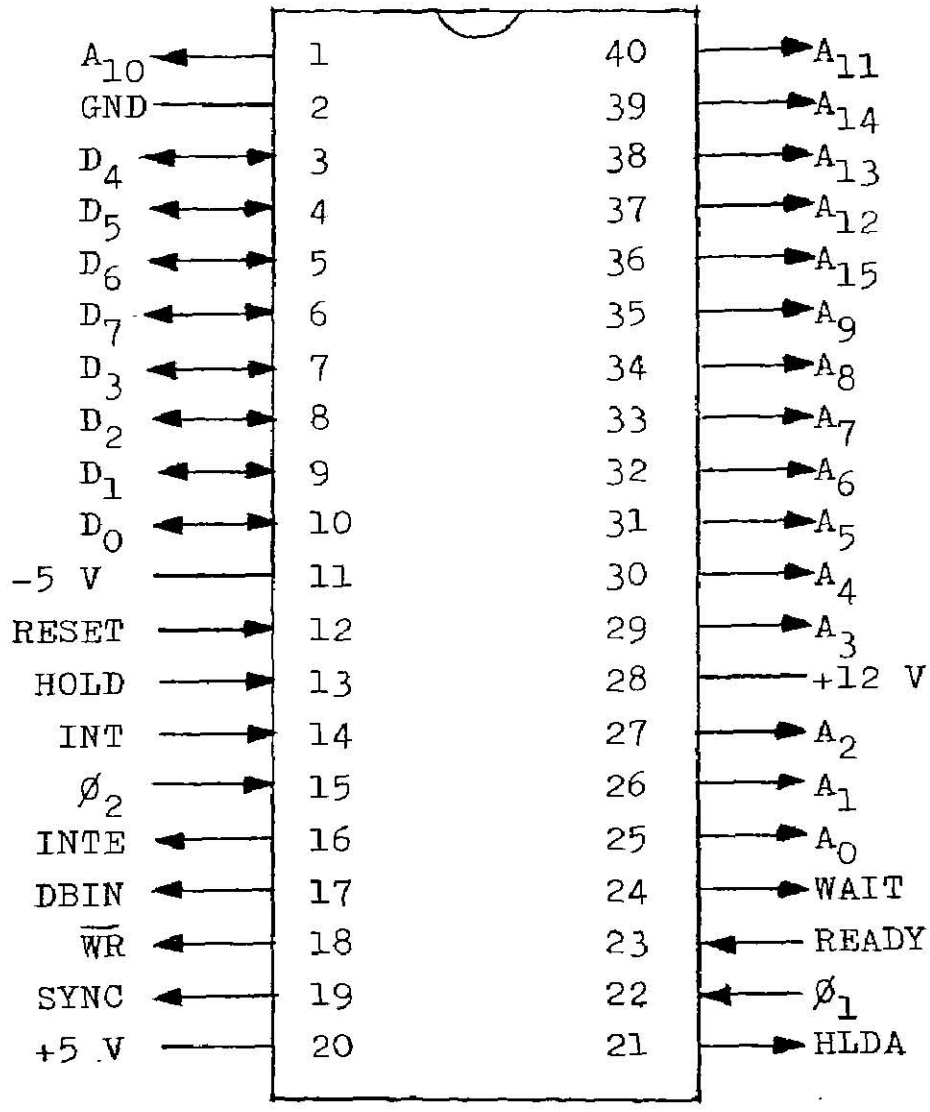
EL MICROPROCESADOR 8080

El microprocesador 8080 presentó muchas ventajas con respecto a su antecesor. Tiene la capacidad de direccionar hasta 64 k-bytes de memoria y su conjunto de instrucciones es más gran-- de y poderoso. Otra ventaja la presenta el hecho de que el -- chip del 8080 tiene 40 pins o patitas, lo cual le permite pre sentar la dirección y dato simultáneamente, que se traduce en menor ciclos por referencia a memoria y el consiguiente aumen to en la velocidad del procesador.

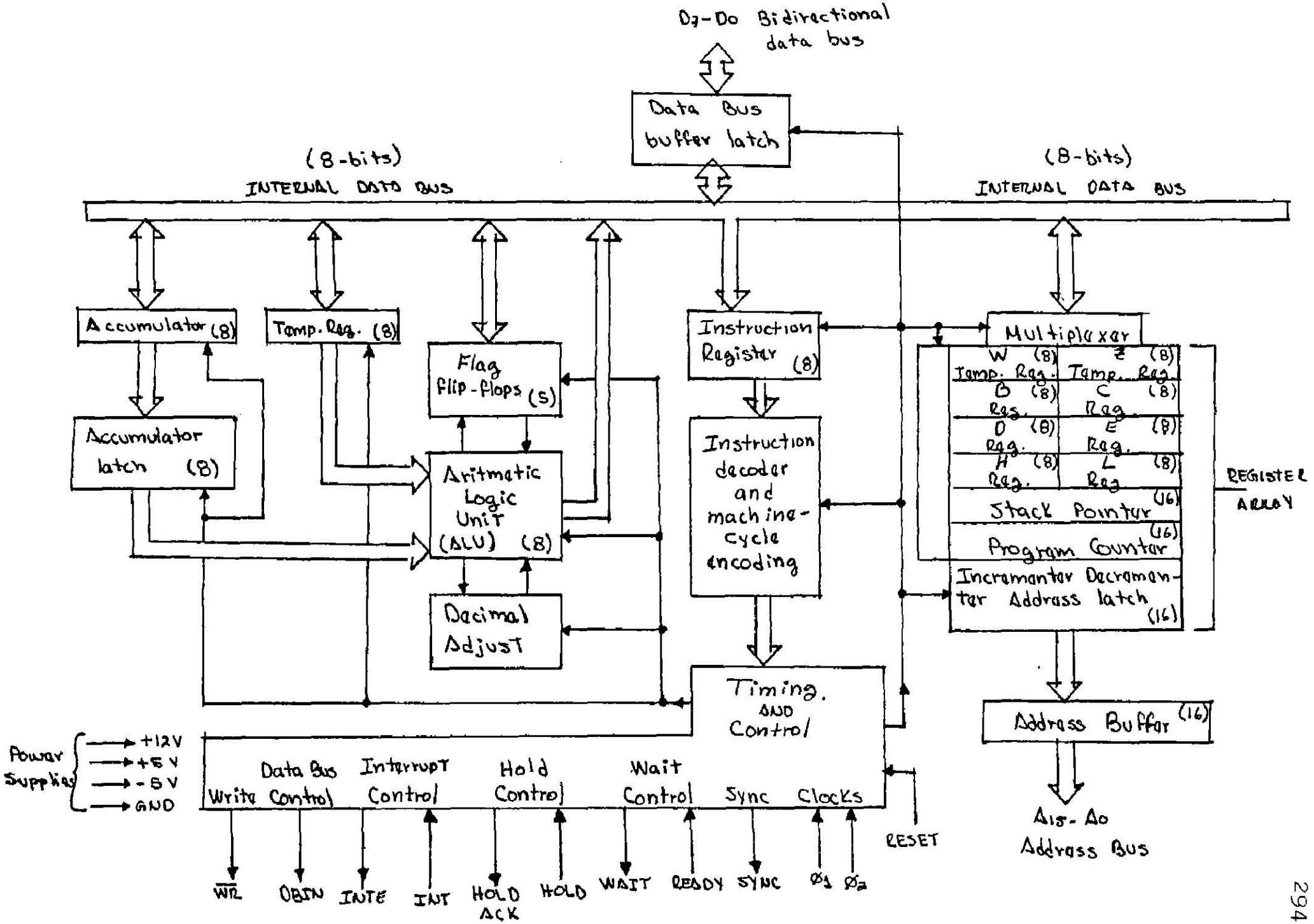
Debido a que el 8080 conservó la mayoría de los registros del 8008, un programa escrito para éste último puede correr en el 8080 si se realizan ciertas modificaciones.

El 8080 tiene algunas instrucciones de 16 bits para manejo de direcciones, pero es considerado como un procesador de bits, - debido a que la mayoría de sus operaciones son de bits. El -- 8080 se convirtió en uno de los microprocesadores más popula- res y aparte del Intel existen al menos 8 compañías que tam-- bién se dedican a fabricarlo, por lo que existe gran variedad de equipo de soporte compatible con él, desde controladores - de DMA(Direct Access Memory) hasta controladores de floppy -- disk.

También el soporte en Software ha sido bastante diversificado y existen disponibles varios editores, ensambladores y lengua jes de alto nivel.



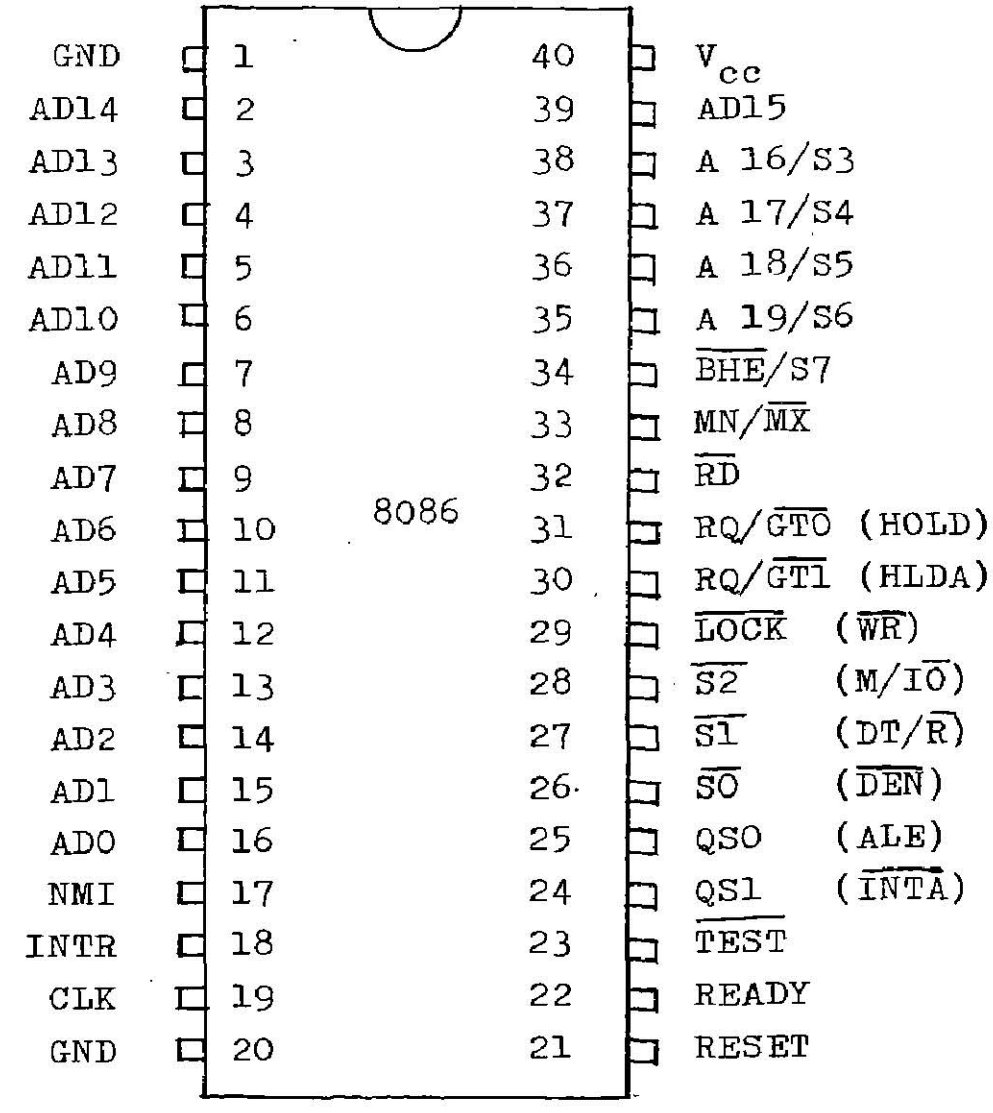
a).- Configuración del Pin 8080



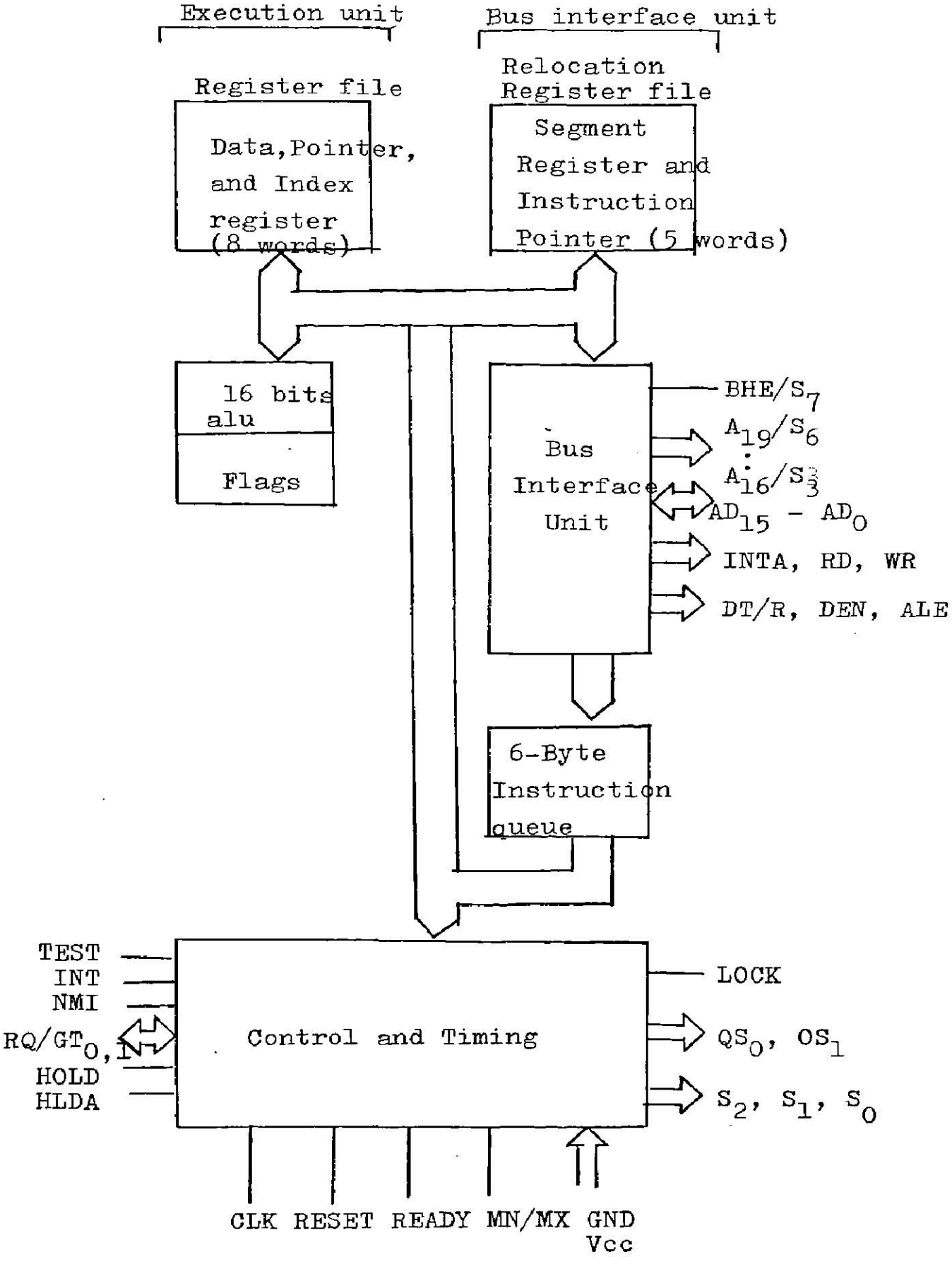
Arquitectura del MPU 8080

EL MICROPROCESADOR 8086

El 8086 es el primer microprocesador de 16 bits de Intel. Aunque en diseño está basado en el 8080, no es directamente compatible con él y presenta demasiadas características que lo sitúan muy aparte. Es un procesador muy sofisticado y entre sus principales ventajas está el poder direccionar hasta un Megabyte de memoria externa. El 8086 viene en un chip de 40 pins y es compatible con muchos de los chips de soporte del 8080.



a).- Configuración del Pin del 8086



a).- Arquitectura del MPU 8086

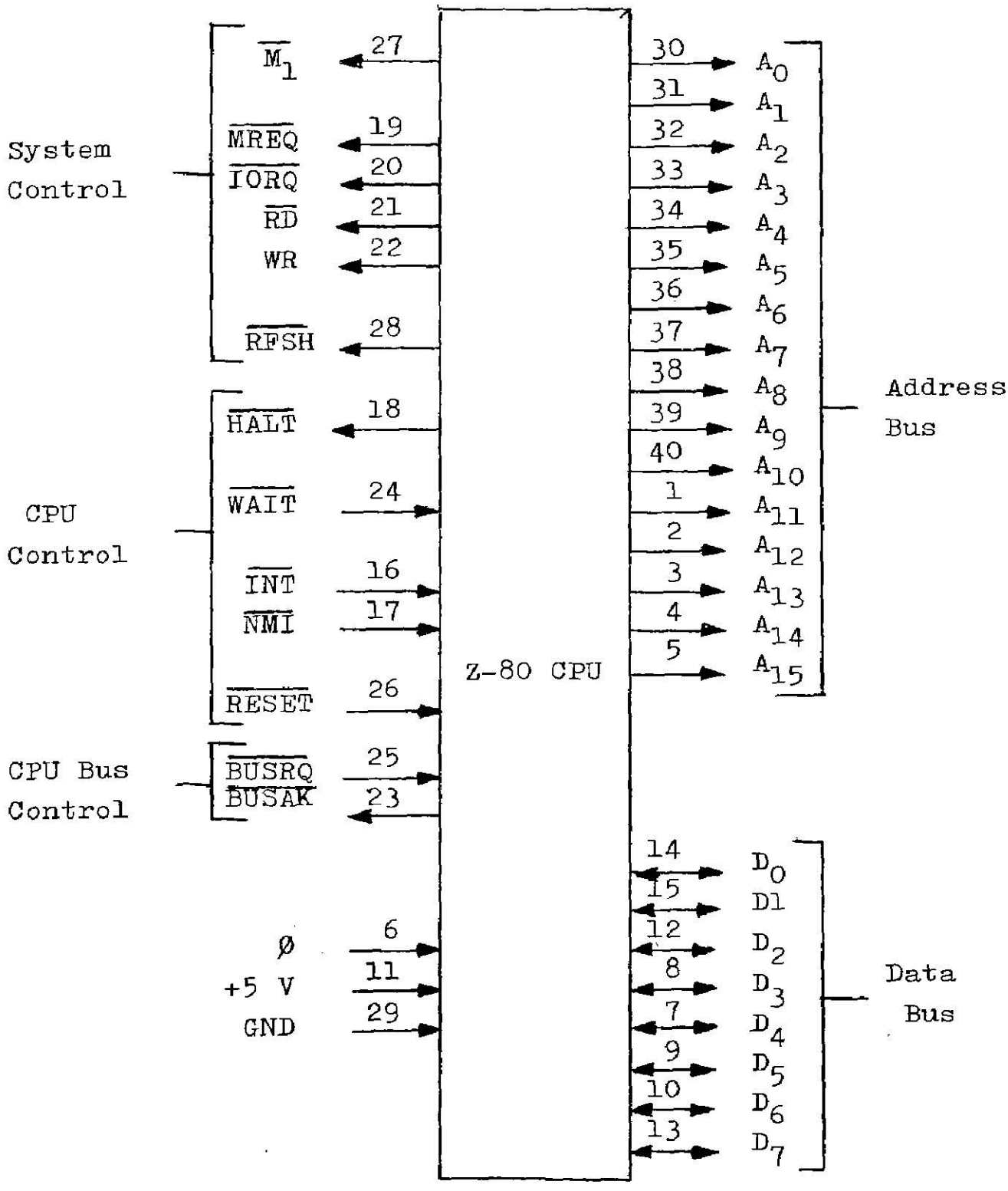
EL MICROPROCESADOR Z-80

El Z-80 es un microprocesador que nació directamente del 8080 pero fué producido por otra compañía, la Zilog, con la idea - de mejorar sustancialmente el funcionamiento del 8080, pero - conservando la compatibilidad con él. Esto se logró y el Z-80 puede ejecutar todas las instrucciones del 8080 y muchas otras más.

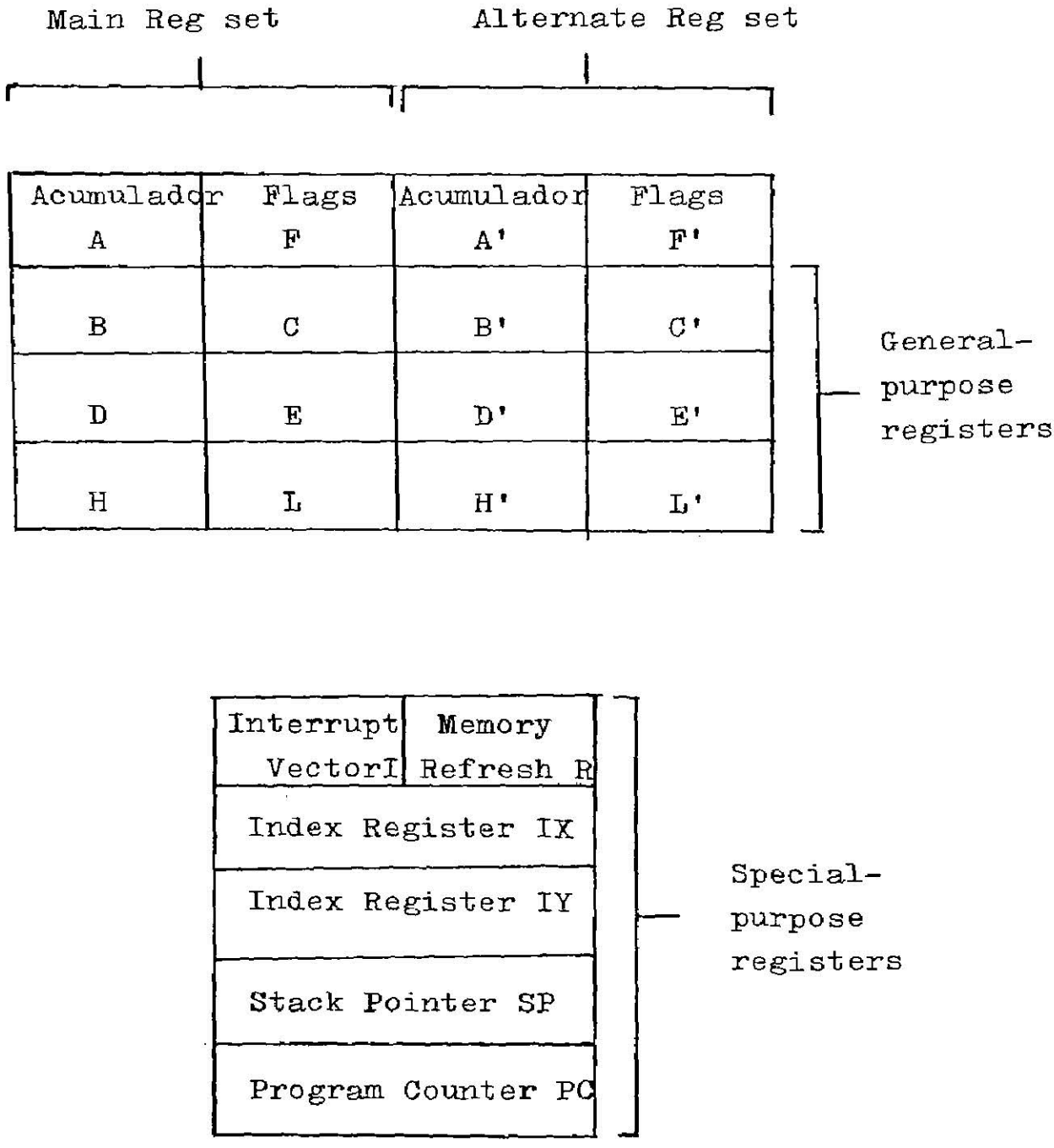
Esto significa que un programa escrito en el 8080 puede correr en el Z-80(Hay contadas excepciones) y los programas escritos expresamente para el Z-80 pueden ser más eficientes. Física-- mente el Z-80 presenta también ventajas significativas: Incor pora al chip del procesador la función de reloj, eliminando - la necesidad de contar con un chip de soporte en reloj. Aun-- que en este aspecto el 8085 puede competir con él, no puede - eliminar la ventaja que representa el conjunto adicional de - registros que presenta el Z-80.

El Z-80 presenta características que lo hacen bueno para fun- ciones de procesamiento de datos y además conserva las carac- terísticas de control del 8080. El soporte en Software para - el Z-80 es bastante grande y puede utilizar varios lenguajes- de alto nivel.

La arquitectura interna del Z-80 es muy similar a la del --- microprocesador 8080 con la muy notable diferencia que tiene- el doble de registros internos.



a).- Configuración del Pin Z-80

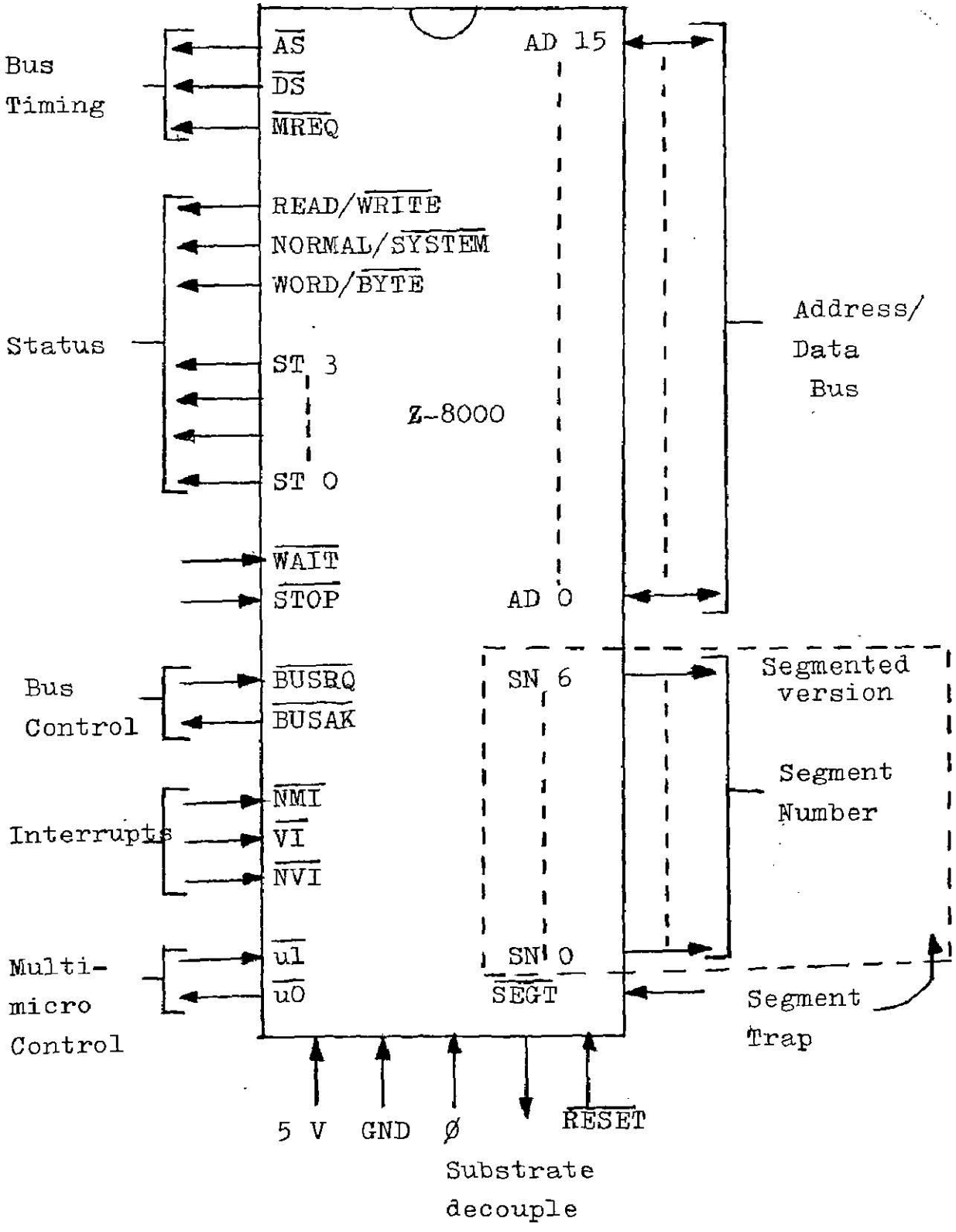


b).- Configuración de Registros del Z-80

EL MICROPROCESADOR Z-8000

La compañía Zilog produjo el microprocesador de 16 bits ---- Z-8000, tomando un camino diferente al de la mayoría de los - fabricantes de microprocesadores de 16 bits. En lugar de ba-- sarse en un microprocesador de 8 bits y tratar de mejorarlo - o de basarse en alguna microcomputadora y tratar de producir- una versión reducida de ella, Zilog adoptó algunas de las -- avanzadas características de los computadores grandes y de -- las minicomputadoras de tamaño mayor, así produjo un micro -- procesador muy potente. El Z-8000 es el primer microprocesa-- dor que se introduce en la competencia de la parte alta del - rango de las minicomputadoras.

El Z-8000 no es compatible con el Z-80, en Software. Tiene un conjunto de 110 instrucciones que incluyen poderosos macro--- instrucciones.



a).- Configuración del Pin de Z-8000

GLOSARIO:

ACUMULADOR
Registro en el cuál se almacenan los resultados de una operación del computador.

ACCESS TIME
Tiempo que toma un byte determinado en memoria, para estar -- disponible al procesador.

ACIA
Adaptador de interface para comunicaciones asincrónicas. In-- terface entre el MPU y un periférico asincrónico.

ADDER
Un circuito combinacional que suma números binarios.

ALFANUMERICO
Conjunto de caracteres que contienen letras del alfabeto y nú meros.

ALMACENAMIENTO NO VOLATIL
Sistema de memoria que retiene datos sin necesidad de energía eléctrica.

ALU
Un circuito lógico combinacional que puede efectuar operaciones lógicas y aritméticas en una computadora digital.

ASINCRONO
Operación del computador que tiene lugar siempre y cuando apa resca información de entrada. El flip-flop RS, por ejemplo, - es un circuito asíncrono.

ASSEMBLER
Programa del computador que convierte automáticamente un con- junto de mnemónicos en lenguaje máquina.

BASIC
Siglas del Beginner's All-Purpose Symbolic Instruction Code. Un lenguaje muy sencillo, usado en muchas computadoras perso- nales.

BAUD
Velocidad de transmisión, se define como bit/seg. que es el - cambio de 0 y 1 lógicos.

BAUDOT CODE
Código de 5 bits para representar caracteres alfanuméricos.

BCD
Ver Binario Codificado a Decimal.

BENCH MARK
Una tarea común para la implementación con programas que pueden ser escritos por diferentes MPUs para determinar un orden de la eficiencia de los diferentes MPUs en una aplicación particular.

BIESTABLE
Un circuito electrónico o mecanismo que tiene dos estados de- operación, ejemplos de estos circuitos pueden ser un switch - mecánico, una lámpara indicadora, un flip-flop, etc.

BINARIO
Sistema numérico cuya base es 2. También un término general - usado para describir una condición o un circuito electrónico- los cuales tienen solamente dos estados(on-off).

BINARIO CODIFICADO A DECIMAL
Sistema numérico usado en computadoras digitales y calculadoras que asigna un número binario a cada uno de los diez dígitos decimales.

BIT
Abreviación común de dígito binario.

BIT DE PARIDAD
Bit que agregado a una palabra binaria hace que el número total de unos sea par o non.

BORRAR
Quitar o remover un dato de una memoria.

BRINCO
Procedimiento de un programa que transfiere el control de una instrucción a otra instrucción en cualquier parte del programa.

BRANCH
El procedimiento de un programa de computadora que transfiere el control de una instrucción en cualquier parte del programa.

BREAKPOINT
Direcciones del programa en el cual la ejecución es bloqueada para permitir el debuger o dato de entrada.

BUFFER
Parte de un computador que aisla un circuito de otro circuito.

BUG
Un error. Puede ser un error en el programa o un defecto en la operación de un computador.

BUS
Uno o más conductores eléctricos que transmiten energía o datos binarios a varias secciones de un computador.

BYTE
Un grupo de 8 bits.

CALCULADORA
Microprocesador básico utilizado principalmente para resolver problemas matemáticos.

CAMPO
Categoría particular o agrupación de datos o instrucciones.

CARACTER
Cualquier letra, número o símbolo que un computador digital puede entender, almacenar o procesar.

CHIP
Una capa delgada de silicio de unas cuantas décimas de pulgada cuadra con un circuito integrado, el cual contiene desde docenas hasta miles de componentes electrónicos, en su superficie.

CICLO
Intervalo de tiempo específico durante el cual tiene lugar una secuencia regular de eventos de un computador.

CINTA DE PAPEL
Cinta de papel que contiene datos binarios en forma de perforaciones.

CIRCUITO
Conjunto de partes electrónicas y conductores eléctricos que ejecutan algunas operaciones útiles.

CIRCUITO INTEGRADO
Circuito Electrónico formado sobre la superficie de un pequeño chip de silicio.

CIRCUITO LOGICO
Una compuerta u otro circuito que responde a señales de dos estados.

CODIGO
Método con el que se representan letras, números, símbolos y datos con números binarios.

CODIGO ASCII
Siglas del Código Standard Americano para el Intercambio de Información. Un código binario que representa caracteres alfanuméricos y varios símbolos(American Standard Code for Information Interchange).

CODIGO FUENTE
Lista de instrucciones que forman el programa fuente.

CODIGO OBJETO
Conjunto de bits que representan instrucciones o datos en el MPU.

CODIGO DE OPERACION
Un conjunto de bits que especifica una operación de máquina en el CPU.

COMPILADOR
Programa del computador que convierte el superlenguaje en ---

lenguaje de máquina.

COMPUERTA
Circuito que realiza alguna operación lógica.

COMPUTADOR
Mecanismo electrónico que procesa señales(datos) discretos y- análogos. Ver Computador Analógico y Computador Digital.

COMPUTADOR ANALOGICO
Computador que utiliza voltajes variables para representar - cantidades numéricas. Un computador analógico específico es - frecuentemente utilizado para resolver un número relativamente pequeño de problemas.

COMPUTADOR DIGITAL
Computador que usa señales discretas para representar cantidades numéricas, casi todas las computadoras digitales modernas son máquinas de dos estados binarios. Estas pueden ser programadas para resolver una amplia variedad de problemas.

COMPUTADOR PERSONAL
Microcomputador con una tarjeta de entrada designado para su- fácil manejo y máxima economía.

CONTADOR
Parte del flip-flop que cuenta en binario.

CPU
Ver Unidad Central de Procesamiento

CRT
Siglas de tubos de rayos catódicos. La pantalla de video usada en sets de Televisión y en muchas terminales de computadora.

CUTS
Usado en sistemas de grabación en una computadora, definición de un sistema de almacenamiento de datos en un cassette con - una serie de tonos para representar 1s y 0s.

DATO
Números, hechos, información, resultados, señales y casi cualquier cosa que pueda ser procesada por un computador.

DEBUG
El proceso de encontrar y corregir un error en un programa o la función actual del computador. Comúnmente toma más tiempo de bugear un programa que escribirlo.

DECIMAL
Sistema numérico cuya base es diez.

DECISION
Operación del computador que compara dos palabras o checa el estado de una palabra o un bit para después tomar un curso específico.

DECODIFICADOR
Circuito combinacional que convierte datos binarios en algún otro sistema numérico.

DECREMENTAR
Disminuir el valor de un número por algún otro valor fijo, -- comúnmente uno.

DEMULTIPLEXER
Circuito combinacional que aplica el estadò lógico de una sola entrada a una de las múltiples salidas.

DIAGRAMA DE FLUJO
Diagrama que muestra los principales pasos u operaciones que se llevan a cabo en un programa.

DIGITO
Caracter en un sistema numérico que representa una cantidad - específica.

DIGITO BINARIO
Los dígitos binarios 0 y 1.

DIRECCION
Número binario que identifica una localización de memoria de almacenamiento específica.

DIRECCIONAMIENTO DIRECTO
Un modo de direccionar donde la dirección del operando está - contenida en la instrucción.

DIRECCIONAMIENTO INDEXADO
Una forma de direccionamiento indirecto en el cual se usa un- registro indexado para mantener la dirección del operando.

DIRECCIONAMIENTO INDIRECTO
Modo de direccionamiento en el cual la dirección de la localidad de memoria en donde se encuentra la dirección del operando, está contenida en la instrucción.

DIRECCIONAMIENTO INMEDIATO
Un modo de direccionamiento en la cual usa como parte de la - misma instrucción el dato del operando.

DOCUMENTACION
Una parte importante en el diseño del computador y el desarrollo del programa. El proceso de grabar en un formato organizado, una lista detallada de consideración de operación y pro-- gramación.

EJECUTAR
Cumplir con o actuar sobre una instrucción en un programa de- un computador digital.

ENCODIFICADOR
Circuito combinacional que convierte un dato de algún otro -- sistema numérico a binario.

ESCRIBIR
Colocar información en una memoria o en un registro.

ESTADO LOGICO
Condición que indica prendido o apagado, verdadero o falso, - si o no, etc. Los estados lógicos están representados por los dígitos binarios 0 y 1 en un computador digital.

ESTADO SOLIDO
Componentes electrónicos o circuitos hechos de materiales sólidos como Silicio y el Germanio.

FIRMWARE
Instrucciones o datos permanentes almacenados en ROM.

FLAG
Un flip-flop que puede estar en set o reset bajo un control de software.

FLIP-FLOP
Circuito lógico secuencial básico. Circuito que está siempre en uno de los dos estados básicos.

FLOPPY DISK
Ver memoria de disco magnético.

GLITCH
Un pulso lógico no deseado, producido cuando dos circuitos lógicos interconectados cambian de estado a un tiempo ligeramente diferente(Ruido Eléctrico).

HALF DUPLEX
Transfiere datos en 2 direcciones pero solo una a un tiempo.

HAND SHAKE
Protocolo de comunicación. Sistema de transferir datos entre el CPU y periférico. Siempre el CPU "pregunta" al periférico si este lo puede aceptar y solo transfiere el dato si la contestación es sí.

HARD COPY
Una impresión en papel de datos o resultados del computador.

HARDWARE
Los circuitos electrónicos en un computador.

HARD WIRE
Circuitos que componen el alambrado a través de compuertas lógicas.

HASH
Señal de ruido.

HEXADECIMAL
Sistema numérico cuya base es 16.

HIGH WAY
Como el bus.

HOUSEKEEPING
Operaciones que tienen lugar en un computador o en un programa del computador que borra memorias, checa el estado de los registros, ordena datos y prepara al sistema para su operación

INITIALICE
Iniciar todos los registros, banderas, etc. para las condiciones definidas.

INSTRUCCION
Bit patrón en el cual puede ser alimentada por el MPU, esto causa que una función particular sea ejecutada.

INSTRUCCION SET
El listado de instrucciones dadas para que el MPU pueda ejecutarlas.

INTERPRETADOR
Programa del computador que convierte y ejecuta otro programa paso por paso.

INTERRUPCION
Las operaciones de un computador digital se detienen temporalmente para atender o dar servicio a un evento externo.

IMPRESORA
Mecanismo de salida que imprime la información en papel.

I/O
Entrada/Salida.

(K)

Esta letra representa la capacidad de memoria de un computa-- dor. Corresponde a 2^{10} (1024). Por lo tanto una memoria de 4K almacena 4096 bits.

KEYBOARD

Un arreglo de switches similar a una máquina de escribir, usado para alimentar datos manualmente a un computador digital.

LATCH

Memoria para retener un estado de una entrada previa hasta -- que sea nuevamente re-escrita.

LAZO

Secuencia de instrucciones que son repetidas una y otra vez - hasta que se obtenga el resultado deseado.

LECTOR DE CINTA(DE PAPEL)

Unidad de entrada de un computador que lee datos de una cinta de papel.

LECTOR DE TARJETAS

Un mecanismo de entrada al computador el cual lee la información contenida en tarjetas perforadas.

LEER

Proceso de recuperar datos de cintas y discos magnéticos o de tarjetas perforadas. O de poner a disposición de otros circuitos de información contenida en una memoria.

LENGUAJE

Los símbolos, frases, caracteres, y números usados para comunicarse con un computador digital.

LENGUAJE DE ALTO NIVEL

Lenguaje de computadora que es fácil de usar , lo cual quiere decir compilación para crear un lenguaje de máquina.

LENGUAJE ASSEMLER
El paso anterior al lenguaje máquina. Sustituye fácilmente -- mnemónicos tales como LDA y CLR por instrucciones en lenguaje máquina binario como 01100111.

LENGUAJE MAQUINA
Lenguaje binario que entiende un computador.

LINE PRINTER
Una impresora que imprime todo un renglón de caracteres en -- una sola operación.

LOGICA COMBINACIONAL
Conjunto de compuertas lógicas que responden a la entrada de- información con un pequeño retardo de tiempo.

LOGICA NEGATIVA
Sistema lógico donde el bit "0" es representado por un nivel- alto de voltaje y el bit "1" por un nivel bajo de voltaje.

LOGICA POSITIVA
Sistema lógico donde el bit "1" se representa por un nivel -- alto de voltaje y el bit "0" por un nivel bajo de voltaje.

LOGICA SECUENCIAL
Conjunto de componentes lógicos que responden a la entrada de información solo cuando se recibe un impulso de reloj. Los -- circuitos lógicos secuenciales hacen uso de flip-flops así -- que cada operación es afectada por operaciones previas.

LOOP
Una secuencia de instrucciones de computadora la cual es repetida en una o más veces hasta que un resultado deseado es llevado a cabo.

MACROINSTRUCCION
Instrucción del computador compuesta de una secuencia de mi-- croinstrucciones.

MAPA DE MEMORIA
Es un diagrama que muestra la organización de un sistema.

MASK
Un bit patrón para ser usado en conjunción con una operación-lógica, para seleccionar un bit particular o los bits de una-palabra máquina.

MECANISMO DE ALMACENAMIENTO
Una memoria (del computador).

MEMORIA
Parte de un computador digital en la cual se almacenan datos.

MEMORIA DE ACCESO AL AZAR
Una memoria que ofrece igual tiempo de acceso a cualquier localización de almacenamiento

MEMORIA DE CINTA MAGNETICA
Sistema de memoria que almacena y recobra datos binarios en - una cinta magnética.

MEMORIA DE DISCO MAGNETICO
Sistema de memoria que almacena o recobra datos binarios so-- bre un disco de metal o de plástico cubierto con un material-magnético.

MEMORIA DE LECTURA/ESCRITURA
Memoria que contiene información que puede ser modificado o - borrado. Generalmente llamada RAM.

MEMORIA SOLO PARA LEER
Memoria que contiene datos permanentes los cuales no pueden - ser alterados o borrados. Generalmente llamada ROM.

MEMORIA MAGNETICA
Un anillo de material que puede almacenar un dígito binario.

MEMORIA VOLATIL
Sistema de memoria que retiene datos solo cuando está presente la alimentación eléctrica.

MICROCOMPUTADOR
Computador digital hecho con una combinación de un microprocesador con uno o más circuitos de memoria. Los microcomputadores de un solo chip también son válidas.

MICROINSTRUCCION
La operación más básica que tiene lugar en un computador digital.

MICROPROCESADOR
La unidad central de procesamiento completa de un computador-digital(unidad aritmética lógica, sección de control y algu-- nos registros en un solo chip).

MICROPROGRAMA
Programa dentro del MPU el cual controla el chip del MPU du-- rante la secuencia básico de ejecución.

MNEMONICO
Abreviaturas o siglas de una instrucción.

MODEM
Modulador(Demodulador usado para transmitir y recibir señales de datos en serie).

MULTIPLEXER
Circuito combinacional que aplica el estado lógico a una de - las varias entradas a una sola salida.

NIBBLE
Es la mitad de un byte, una palabra de 4 bits.

NUCLEO MEGNETICO
Pequeño aro de metal que puede almacenar un bit.

NUMERO
La representación de una cantidad. En un computador digital - los números pueden representar datos, caracteres, instrucciones, etc.

OCTAL
Sistema numérico de base 8.

OPERACION ILEGAL
Instrucción de un programa que no puede ser ejecutado.

OPERANDO
Dato usado para operaciones a máquina.

PALABRA
Línea de bits que representan un número, un caracter o una -- instrucción en un computador digital. Las palabras pueden tener cualquier longitud.

PARALELO
Transferencia de datos de 2 o más bits a un mismo tiempo.

PARALLEL PROCESSING
Operación de un dato multibit a la vez.

PARIDAD
Checar el bit de la suma de datos puede ser paridad par o impar. Una suma de paridad impar de datos ls + el bit de pari-- dad es impar.

PERIFERICOS
Equipo para entrada o salida de un sistema(Ejm., Teletipo).

PIA
Adaptador de Interfase Periférico.

PORT
Una terminal la cual el MPU usa para comunicarse con el mundo externo.

PRIMERA GENERACION
Computadores digitales hechos con bulbos.

PROCESADOR
Computador digital.

PROGRAMA
Un listado de instrucciones que dicen al computador que hacer y como hacerlo.

PROGAMA FUENTE
Programa escrito en un lenguaje como el Assembler o el Basic.

PROGRAMA OBJETO
Programa escrito o expresado en lenguaje máquina.

PROM
Programable ROM. El PROM es un ROM especial que puede ser programado por el usuario.

PUSH
Operación de poner datos en el STACK.

RAM
Ver memoria de Lectura/Escritura.

REGISTRO
Parte de un flip-flop que almacena una palabra de un dato binario. Un registro es una memoria temporal.

REGISTRO DE ESTADO
Registro que es usado para almacenar la condición de los acumuladores después de que una instrucción ha sido ejecutada.

REGISTRO DE INSTRUCCION
Registro del MPU, el cual es usado para mantener la instrucción tomada de la memoria.

RELOJ
Circuito que produce una secuencia de pulsos eléctricos regularmente espaciados que sincronizan la operación de varios circuitos en un computador digital.

ROM
Ver memoria solo para leer.

SECCION DE CONTROL
Nervio electrónico central de un computador digital, circuito que decodifica las instrucciones de entrada y activa las varias secciones de un computador en perfecta sincronía. Es parte de la Unidad Central de Procesamiento(CPU).

SECCION DE SALIDA
Una impresora, una terminal de video u otro mecanismo que muestre la información procesada por un computador a un operador o a un mecanismo electrónico.

SEGUNDA GENERACION
Computadoras hechas con transistores.

SERIE
Transfiere datos de un bit a un tiempo.

SERIAL PROCESSING
Operando sobre un dato de un bit a un tiempo.

SIMPLEX
Transmisión de datos en una sola dirección.

SINCRONO
Operación del computador que tiene lugar bajo el control del reloj.

SINTAX
La gramática de lenguaje de programación.

SOFTWARE
Papel de trabajo, como programas y documentación de estos asociada con la operación de un computador.

SUBRUTINA
Secuencia de instrucciones en un programa que es usada más de una vez por el programa principal.

SUMADOR
Circuito lógico combinacional oue suma números binarios.

STACK
Lugar de almacenamiento de los registros del MPU.

TARJETA
Una tarjeta de papel que contiene perforaciones y que almacena datos del computador o instrucciones de un programa.

TARJETA LECTORA
Un dispositivo de entrada en una computadora que lee el contenido de las tarjetas.

TELETIPEWRITER
Aparato similar a una máquina de escribir que puede ser usado para alimentar y programar a un computador e imprimir la información de salida(de un computador) en una tira de papel. (Tele impresora o Teletipo).

TERCERA GENERACION
Computadoras hechas con circuitos integrados.

TIEMPO DE EJECUCION
El tiempo tomado para ejecutar una instrucción en términos de ciclos de reloj.

TRANSISTOR
Mecanismo electrónico de estado sólida que puede ser usado como amplificador o como un switch on-off. Un circuito integrado es una red compleja de transistores y otros componentes sobre la superficie del chip.

TRAP
Localización pre-definida en memoria en el cual el procesador deberá leer como resultado de una operación.

TRI-STATE
Descripción de dispositivos lógicos cuyas salidas pueden ser deshabilitadas, colocando en ellas un estado de alta impedancia.

UNIDAD ARITMETICA LOGICA

Circuito lógico combinacional que ejecuta operaciones aritméticas y lógicas en un computador digital.

UNIDAD CENTRAL DE PROCESAMIENTO

La unidad aritmética lógica y las secciones de control de un computador digital.

VART

Transmisor receptor asincrónico universal.

VECTOR

Direcciones de memoria que son proporcionadas al procesador para direccionarse a una nueva área de memoria.

VOLATILE

Dispositivo de memoria que pierde el dato contenido si se deja de alimentar con corriente.(Ejm., la RAM).

B I B L I O G R A F I A

6800 Assembly Language Programming

Autor : Lance A. Leventhal

Apuntes para el Curso de Microprocesadores

Volumen II

Autores: Ernesto López

Javier Carrillo

Manuel Méndez

Manual de: Programación y Aplicaciones del MEK 6802-D5

Autor: Ing. Cesar A. Leal