CAPITULO 3

3.1 SISTEMAS DIGITALES DE PROGRAMA ALMACENADO

3.1.1 Estructura de Von-Newmann

Una máquina automática de cómputos de propósito general debe contener los siguientes componentes básicos:

1) **Unidad Aritmética y Lógica (ALU)** para realizar las operaciones aritméticas y lógicas elementales.
2) **Unidad de Memoria (UM)** donde almacenar los datos (información) y las órdenes (instrucciones).
3) **Unidad de Control (UC)** para ejecutar las órdenes y llevar el "control".
4) **Unidad de Entrada y Salida (UE/S)** para realizar las comunicaciones hombre-máquina o máquina-máquina.

- Una máquina de cómputos debe ser capaz de almacenar la información (datos) y órdenes (instrucciones), las cuales gobernarán una tarea determinada (rutina). Si las órdenes se reducen a un código numérico, y si la máquina, de alguna manera, distingue orden de dato, entonces es posible almacenar a ambos en la misma memoria.

- Debido a que la UM únicamente sirve para almacenar instrucciones y datos, debería existir un órgano (unidad) que interprete (ejecute) las órdenes guardadas en la UM. Esta es la Unidad de Control (UC).

- Ya que la máquina propuesta es una máquina de cómputos, debe poder realizar operaciones aritméticas elementales. Esta se lleva a cabo por la ALU. Este dispositivo es capaz de realizar operaciones lógicas y aritméticas elementales.

- Por último, debería existir algún medio de comunicación entre máquina-máquina o máquina-hombre para realizar intercambio de información. Esto lo realiza la Unidad E/S.

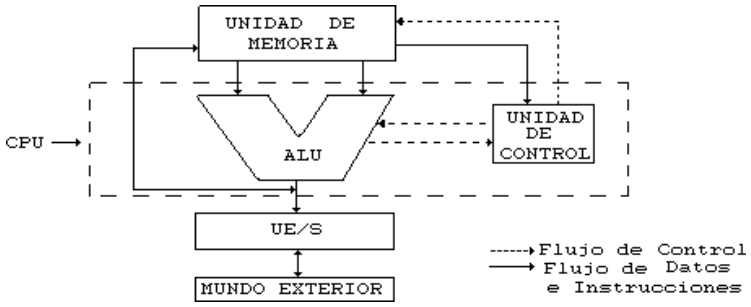


FIGURA 3.1

Las principales características de la arquitectura de Von-Newmann son:

1) Un programa con un dato se almacenan en una única memoria en forma secuencial, describiendo dicha secuencia la tarea deseada.

FIGURA 3.2

2) La memoria es un sistema unidimensional, es decir, tiene la apariencia de un vector de palabras.

3) No hay una forma explícita de distinguir entre datos e instrucciones, vista la memoria como un bloque de información. Es decir, tanto datos como instrucciones son información binaria; ambos se pueden confundir. Por lo tanto, internamente, el procesador debe poder (de alguna forma) distinguirlos.

4) Tiene un contador (registro) de instrucciones, el cual mantiene la dirección de la próxima instrucción a ser ejecutada. Este contador es implícita o explícitamente actualizado para proveer a la máquina con una secuencia de instrucciones a ejecutar, implicando un simple lugar de control. Esta es una de las fundamentales limitaciones del modelo de Von-Newmann, con respecto a la velocidad de ejecución.

Las soluciones propuestas para aumentar la ejecución de un programa son:

Pipelining: consiste en proveer dos unidades independientes:

. unidad de ejecución,

. unidad de mecanismo de decodificación y búsqueda
de la instrucción (prefetch).

Multiprocesamiento: consiste en distribuir el trabajo en numerosas máquinas Von-Newmann, compartiendo memorias.

De lo visto anteriormente se puede inferir que, para la ejecución de una tarea (instrucción), se realiza la siguiente secuencia de acciones o "ciclo de ejecución de la instrucción":

1- La UC requiere y busca de memoria la próxima instrucción a ser ejecutada.
2- La UC decodifica la instrucción: interpreta qué es lo que tiene que hacer.
3- Dependiendo del resultado del paso 2:
   a- Se busca un operando de memoria, y se lo almacena en un registro de la ALU; luego se le da la orden a la ALU para realizar una operación.
   b- Se almacena un operando en memoria como resultado de una operación realizada por la ALU.
   c- Se realiza un pedido de I/O para transferir una palabra al, o del, exterior.
4- Una vez terminado el paso 3, retornar al paso 1.

Conocidos los componentes básicos de una Máquina de Cómputos, un diseñador de hardware no tiene suficientes restricciones para un único diseño, pues habría que responder una serie de preguntas:
1) ¿Cuántas instrucciones se pueden ejecutar simultáneamente?
2) ¿Cómo y dónde guardar los datos e instrucciones en memoria?
3) ¿Cómo accedería la UC a las instrucciones y/o datos?

Una arquitectura como la de la fig.3.1 se denomina Arquitectura de Von-Newmann. Un microprocesador en sí mismo no es una arquitectura de Von-Newmann completa. Un microprocesador (llamado CPU) consiste internamente de una UC, de una ALU y de una serie de registros. Un microprocesador contiene las líneas para unirse a una UM y a una UE/S para formar dicha arquitectura. Un microprocesador así configurado se denomina microcomputadora. Actualmente, el término microprocesador contiene algunos elementos de memoria y de entrada\salida dentro del chip.

Unidad de Memoria (fig.3.3)

Consta de las siguientes líneas:

1) Datos de entrada.
2) Datos de salida.
3) Direcciones.
4) Control de escritura.
5) Habilitación (enable).
6) Estado (status).

Un concepto importante de la arquitectura Von-Newmann es que los datos y las instrucciones se encuentran entrelazados en la UM (fig.3.2).

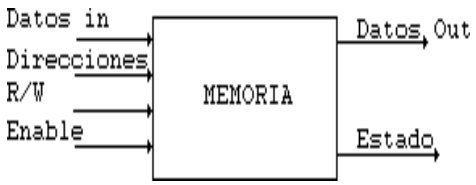


FIGURA 3.3

Unidad Aritmética y Lógica (fig.3.4)

Consta de las siguientes líneas:

1) Dos datos de entrada X e Y.
2) Datos de salida (Resultado) ZR.
3) Líneas de control (Función a realizar).
4) Registro código de condición. Generar ciertas banderas (Flags)de resultado: Z:cero, N:signo, C:carry y V:overflow.

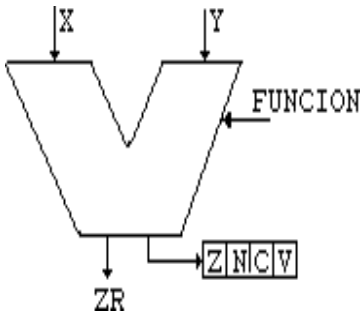


FIGURA 3.4

Unidad de Entrada/Salida (fig.3.5)

Es similar a la UM con la diferencia que ocupa menos direcciones de memoria.

La línea Control se refiere a qué tipo de transferencia (de Input o Output) se realiza.

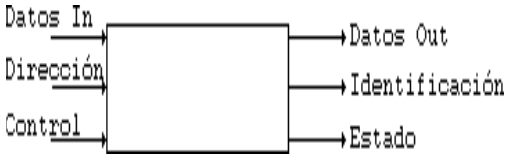


FIGURA 3.5

Unidad de Control (fig.3.6)

Es la unidad más compleja de las cuatro. Su misión es realizar la secuencia y coordinación de las señales de control para realizar la ejecución de una instrucción determinada. Es decir, interpreta cada instrucción recibida.

El registro de instrucción (IR) se encarga de mantener la instrucción que está siendo ejecutada.

El temporizado interno de señales determina cuándo una dada señal de control se debe activar. Dicho temporizado se puede realizar en forma sincrónica o asincrónica.

- Asincrónica: .permite mejor empleo del tiempo,
  .más compleja en la práctica,
  .problemas de sincronización con dispositivos externos.

- Sincrónica: .los temporizados se implementan en base a la señal de un reloj
  .las señales de control se activan por un número proporcional al periodo del reloj,
  .todos los microprocesadores trabajan en forma sincrónica.

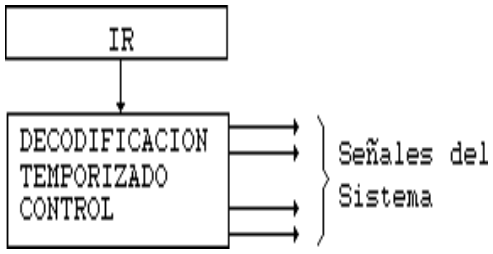


FIGURA 3.6

3.2 ARQUITECTURA DE UNA UNIDAD PROCESADORA CENTRAL

Un diagrama en bloque de una microcomputadora, de acuerdo a las definiciones anteriores, sería como el de la fig.3.7. El microprocesador tiene una serie de líneas conocidas como canales (bus) del microprocesador. Los buses se dividen en tres tipos:

a) Canal de Control.

b) Canal de Datos.

c) Canal de Direcciones.

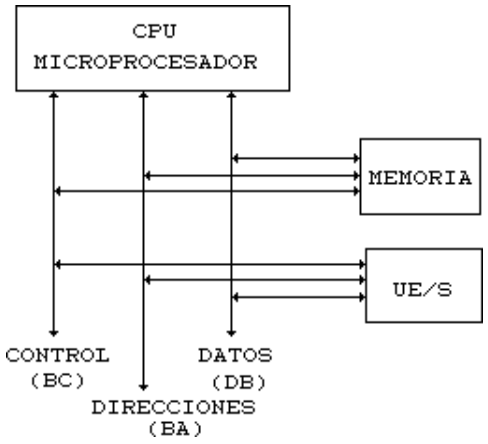


FIGURA 3.7

El bus de direcciones identifica un lugar de memoria o un periférico con el cual se quiere comunicar la CPU. No es necesario separar las direcciones para la UM o la UE/S. El bus de direcciones define la longitud del mapa de memoria.

El bus de datos es el medio por el cual se obtienen los datos y las instrucciones a ser ejecutadas. Es un medio de transmisión bidireccional. Determina la longitud de la palabra.

El bus de control está formado por señales que le permiten al procesador comunicarse con la memoria o los periféricos en una forma "disciplinada". Por ejemplo, una línea de control podría indicar que se está realizando una transferencia de información, y otra, la dirección de la transferencia. También hay líneas enviadas por la memoria o el periférico para indicar la finalización de un ciclo de lectura o escritura.

3.3 ORGANIZACION INTERNA DE UNA CPU

Una CPU consta de tres grandes partes (fig.3.8):

1) Unidad de Control.

2) ALU.

3) Arreglo de Registros.

FIGURA 3.8

Un único bus de datos (IDB) interno es el medio de transporte de datos entre los distintos registros internos dentro del procesador.

3.3.1 Funciones de la CPU

1- Búsqueda de la instrucción y/o datos de la memoria.

2- Decodificación de la instrucción.

3- Generación de las señales de temporizado y control a los restantes componentes de la microcomputadora (en respuesta a una instrucción de memoria).

4- Transferencia de datos con los dispositivos de I/O.

5- Realización de las operaciones Aritméticas y Lógicas.

A continuación se describirán las tres secciones:

Sección de Registros

Las operaciones más comunes en el interior de la CPU son las transferencias binarias entre registros. Una pequeña cantidad de registros se necesita para hacer eficiente la ejecución de las instrucciones. No todos los microprocesadores tienen los mismos registros, sin embargo hay un conjunto de registros que podemos considerar básicos.

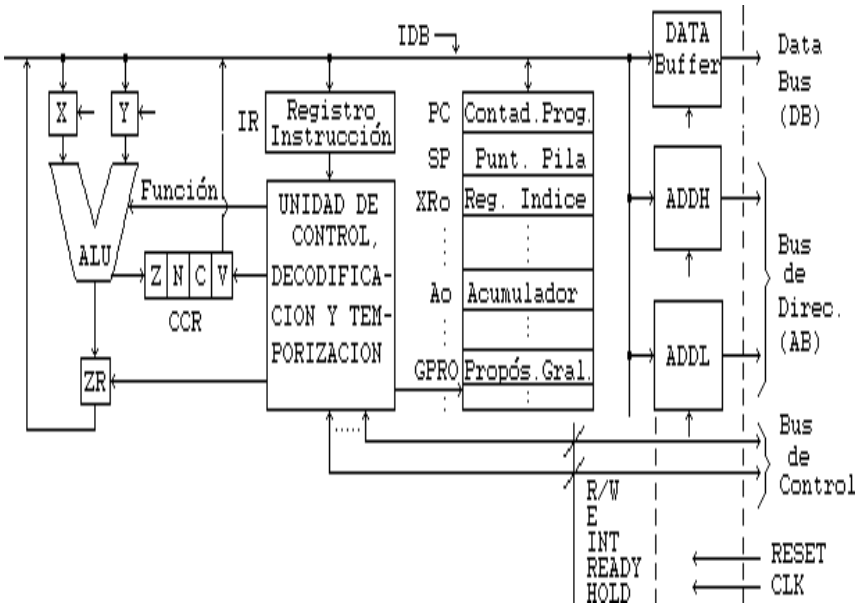


FIGURA 3.9

- Registro de Instrucción (IR)

Cuando la CPU busca una instrucción de memoria, ésta se almacena en el IR. Esta permanece en el IR mientras el circuito de decodificación de instrucción determina cuál es la instrucción a ejecutar. Este registro no está a disposición del usuario. La longitud de palabra es la misma que el IDB.

- Contador de Programa (PC: Program Counter)

El PC siempre contiene la direcciòn de memoria de la próxima instrucción a ser ejecutada o del próximo lugar de memoria, dependiendo de la cantidad (longitud) de bytes de la instrucción presente. Es decir, el PC apunta a la próxima instrucción; si la misma es de más de 1 byte de longitud, deberá recorrer todos los bytes que la componen para poder leer la parte del operando de la instrucción.

El PC es automáticamente incrementado, generalmente después de cada lectura de memoria. Su contenido es alterado, **por software,** mediante instrucciones de salto (JMP)(cond., incond.) o llamados a subrutina y, **por hardware,** por la línea 'reset' o por líneas de interrupciones, en cuyo caso el PC toma un valor particular. El número de bits generalmente determina la longitud del mapa de memoria.

- Acumuladores (A)

Este es un registro que está involucrado en la mayoría de las operaciones aritméticas y lógicas ejecutadas en la ALU. Además es, generalmente, fuente y

destino de la mayoría de los resultados de la ALU. Otros usos: almacenaje de datos leídos de un periférico, contador, registro de propósito general como operaciones de desplazamiento, etc. El ancho de palabra es el mismo que el del bus de datos. Hay procesadores que tienen más de un acumulador (ejemplo: el µp 6800 con dos acumuladores, A y B).

- Registros de Propósito General (GPR)

Son los registros que se pueden usar como acumuladores, registros índices, contadores (generador de retardos), registros de direccionamiento indirecto, etc. Las operaciones más importantes son:
GPR<--M; M<--GPR; [GPR]<--Reg; Reg<--[GPR]; INC [GPR]; DEC [GPR].

- Registros Indices (XR)

Los registros índices se pueden usar como contadores o como registros que almacenan direcciones de datos (muy importante en el manejo de tablas o arreglos). Las operaciones posibles sobre estos registros son: cargarle un valor, incrementar o decrementar en 1 su valor. La idea básica de direccionamiento indexado se refiere a que la dirección efectiva se forma por la suma del contenido del registro índice más un desplazamiento u "offset" dado como operando en la instrucción. El número de bits puede ser:

- 16 bits como en el caso del µp 6800
- 8 bits como en el caso del µp 6502

Hay microprocesadores que tienen más de un registro índice, y otros distinguen entre registro índice fuente (SI) y registro índice destino (DI), para el manejo de cadenas de datos.

- Registro de Estado (CCR: Código de Condición)

Consiste en un conjunto de bits individuales llamados "flags" que indican el estado de una condición particular del microprocesador, normalmente actualizados por las operaciones de la ALU. Los valores de estos flags se pueden examinar bajo el control de un programa para la toma de decisiones (resultados de sucesivos decrementos, comparaciones de positivo o negativo, etc.). La mayoría de las instrucciones de salto condicionales se refieren al valor de los flags. Por ejemplo: JZ (saltar si es cero), JC (saltar si carry es "1"), etc. Los flags típicos son: C:carry, Z:cero, V:overflow, P:paridad, N:signo, HC:half carry, B:borrow, I:interrupciones habilitadas.

- Registro Puntero de Pila (SP: Stack Pointer)

La pila de un sistema es una porción de memoria RAM que se reserva para almacenaje temporario de los contenidos de los registros, de datos o de direcciones de la CPU. Normalmente se llena en orden decreciente de direcciones y se vacía en forma inversa. (Estructura tipo LIFO).

El SP es un registro de dirección de memoria que sirve para este fin, pues si se desea guardar un dato en el stack, éste se decrementa en 1. Por el contrario, si se desea quitar un dato del stack, éste se incrementa en 1.

Usos: en saltos a subrutinas, interrupciones, pasaje de parámetros a subrutinas.

- Registro "Buffer" (DB - AB)

Hay dos tipos de registros buffer: de direcciones (AB) y de datos (DB). Estos son los registros de salida del canal de datos y direcciones. El AB contiene la dirección de un lugar de memoria o periférico de I/O, y el DB, el dato a ser transferido. En ambos casos, los contenidos se mantienen estables durante un cierto tiempo (tiempo de acceso) para asegurar la transferencia.

Hay situaciones especiales en las cuales es necesario otro tipo de registro relacionado con las direcciones de memoria, el cual se denomina MAR registro de direcciones de memoria. Este es un registro temporario que se usa para mantener las direcciones de datos de memoria, en las cuales la CPU debe leer o escribir. Por ejemplo, en instrucciones del tipo ADD [dirección] (A<--A+[dirección]), la CPU debe poner la dirección del operando de ADD en el MAR, y luego el contenido del MAR se carga en el AB para obtener el dato en la operación de suma con el acumulador.

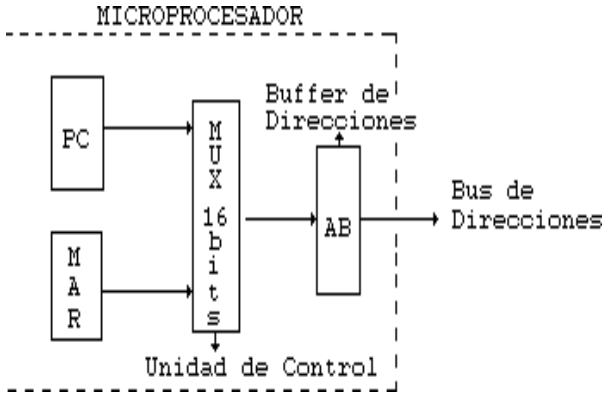


FIGURA 3.10

Otra alternativa para poder ejecutar la instrucción anterior sería almacenar la dirección en registros temporarios, como X e Y en la arquitectura de la fig 3.9, y luego enviarlos al buffer de direcciones (AB), para traer el dato de memoria.

Registros Especiales

Algunos microprocesadores tienen registros de propósito especial, cuya función permite facilitar el desarrollo del hardware (Refresh Register) o del software (Banco de Registros) de un sistema.

- Registro de Refresco de Memoria (MRC)

Algunos microprocesadores (Z80) tienen este tipo de registro. Dicho registro contiene la dirección actual de refresco de memoria dinámica; esto tiene la ventaja de no necesitar un controlador de memoria dinámica o de facilitar su implementación. El procesador se encarga de presentar las direcciones de refresco en los momentos en que no se usa el canal de datos, además genera una línea RSH para indicar el refresco. Esta actividad se realiza durante la parte del ciclo Fetch (búsqueda de la instrucción) en que la CPU comienza la decodificación y ejecución de la instrucción, siendo totalmente transparente al programa.

- Registros Segmentos

Hasta ahora, cuando se accedía a memoria principal, para buscar el código de operación de una instrucción o un dato, la dirección física de memoria coincidía con: la dirección efectiva calculada por la instrucción, o con el contenido de algún registro de datos, o con el contenido del PC. Esto es cierto para la mayoría de los microprocesadores de 8 bits. Sin embargo, para los microprocesadores de 16 bits esto no es así, ya que emplean de alguna manera registros especiales llamados "segmentos" cuando acceden a la memoria principal.

A la dirección calculada por una instrucción se la denomina "dirección lógica de programa", y a la que apunta a la memoria se la denomina "dirección física". El mecanismo para traducir una dirección lógica en dirección física se llama "mecanismo de mapeo". Este mecanismo se implementa en una unidad llamada MMU (Memory Management Unit), fig.3.11, la cual puede estar dentro de la CPU o como unidad separada.

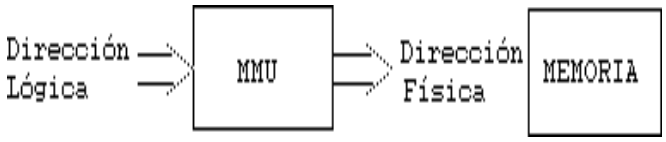


FIGURA 3.11

Los segmentos permiten dividir el espacio de memoria en diferentes áreas llamadas segmentos. Un segmento es una unidad lógica de memoria contigua de hasta 64 Kbytes de longitud.

Ejemplo : 8086.

El mecanismo de trasformación para el 8086 es el siguiente. El 8086 tiene un PC o una dirección efectiva de 16 bits y un canal de direcciones de 20 bits; es decir, permite direccionar 1 Mbyte.

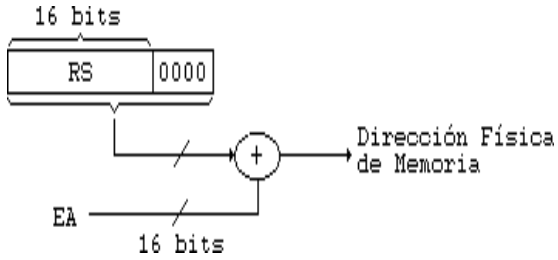


FIGURA 3.12

El 8086 posee cuatro Registros Segmentos CD, DS, ES, SS, que dan la base del segmento (principio del segmento) para códigos, datos y pila. Los segmentos pueden ser disjuntos, parcialmente o totalmente solapados.

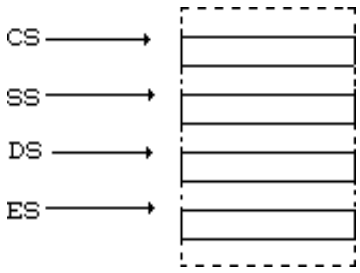


FIGURA 3.13

Con la configuración anterior es posible, aún teniendo un PC de 16 bits, lograr un direccionamiento mayor que 64K,(por ejemplo 1Mbyte). Esto se realiza segmentando el mapa de memoria (en módulos (segmentos) de 64K lugares). La

dirección física de memoria se forma de la siguiente forma:

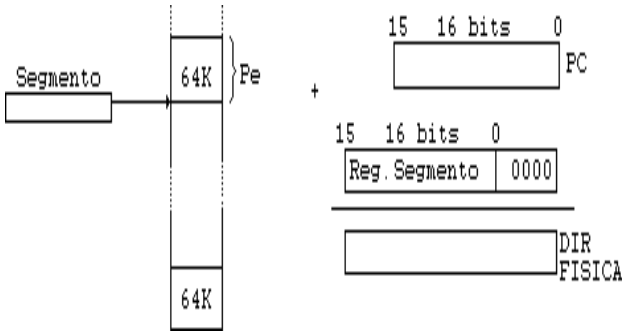


FIGURA 3.14

El registro segmento se adiciona a la dirección efectiva (EA) (offset) formada de los distintos modos de direccionamiento para generar la dirección física. Un segmento tiene asignado una dirección base (por software) la cual es el comienzo del mismo en memoria. Este punto es múltiplo de 16, ya que los 4 bits menos significativos de la dirección base son siempre cero.

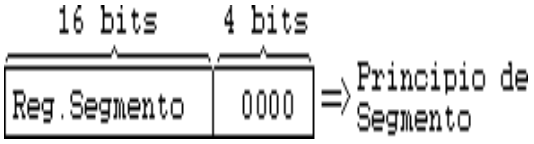


FIGURA 3.15

Los 16 bits que forma el microprocesador es un registro offset de 16 bits, que se refiere como dirección lógica. Una dirección física es un valor de 20 bits que identifica un byte de memoria.

Una ventaja de esta forma de direccionamiento (además de la mayor capacidad de direccionamiento: 1Mb) es que un programa que no haga uso de los segmentos, se puede ejecutar en cualquiera de los módulos, es decir, es dinámicamente reubicable (relocatable).

- Banco de Registros

Hay microprocesadores que tienen 2 conjuntos de registros idénticos, llamados bancos de registros. Esto es muy útil cuando se necesitan conservar los contenidos en ciertos registros, y se debe ejecutar una rutina prioritaria (una rutina de atención de interrupción), entonces, en lugar de guardar todos los contenidos en memoria, uno podría cambiar el banco de registros y usar los registros del otro

banco. Ejemplos: microcontroladores 8048, 8051...

Banco 0: r0,r1,r2,r3
Banco 1: r0,r1,r2,r3

- Registro de Interrupción (o Vector de Interrupción)

La función de este registro es la de formar una dirección de entrada a una tabla donde se hallan las direcciones de las distintas rutinas de interrupción. La CPU utiliza este puntero para realizar un salto indirecto a la dirección donde se halla la primera instrucción de la rutina de interrupción.

- Registro de Página

Hay ciertos microprocesadores, como el 6809, que poseen un registro llamado Registro de Página. Este registro contiene los 8 bits más significativos de una dirección de 16 bits, para todos los modos de direccionamiento directo. Este modo consiste en direccionar un lugar de memoria especificando únicamente los 8 bits menos significativos, ya que los 8 bits más significativos provienen del registro de página. Este modo de direccionamiento a veces se lo llama *paginado*.

Sección ALU (Unidad Aritmética y Lógica)

Una ALU es un circuito combinacional que realiza operaciones aritméticas y lógicas, involucrando 1 o 2 operandos.

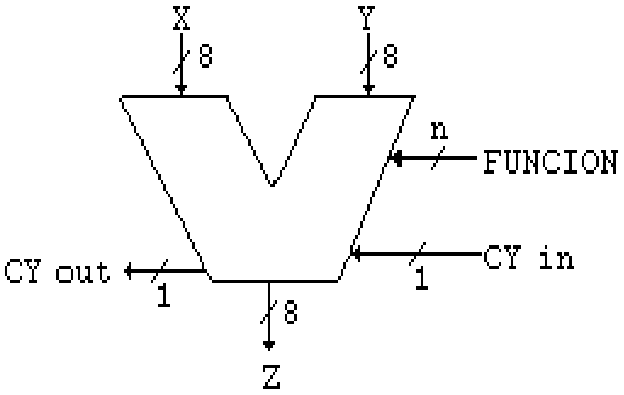


FIGURA 3.16

La ALU realiza la operación seleccionada por las líneas de control (FUNCION) sobre el o los operandos X e Y, de acuerdo al tipo de operación seleccionada. El resultado se obtiene en la salida Z. Evidentemente, los operandos X e Y pueden

tener numerosas fuentes, ya que generalmente están conectados al bus de datos (Acc, XLow, [M]).

Entre las operaciones de un <u>único operando</u> se pueden mencionar: Complemento Z=/X, Clear Z=0, Incremento y Decremento en 1 (usado para operaciones de reg A, X y PC), Desplazamiento: rotaciones con o sin carry. Con <u>dos operandos</u>: suma A+[M], A+R; resta A-[M] y A-R, (donde debe actualizar el carry); compara A-[M] (importante en los saltos condicionales); funciones lógicas AND, OR, XOR, XNOR; con respecto a la resta de números, es muy común que usen la forma en complemento a1 o a2.

Generalmente, en microprocesadores, el resultado se almacena nuevamente en el acumulador (A), es por ello que en la fig.3.9 se podría quitar el registro ZR.

Con respecto a las operaciones en BCD, hay resultados que es necesario corregirlos, pues las operaciones se realizan en binario. La mayoría de los microprocesadores tienen una lógica de ajuste decimal que se encarga de ello.

En los microprocesadores de 16 bits, se provee el manejo de números con y sin signo en 8 y 16 bits, incluyendo multiplicación y división. En microprocesadores de 32 bits se provee el hardware para punto flotante.

Con respecto a la generación de los flags de estado (Z, N, C, etc.), el único flag que se genera por sí solo es el de carry, pues los restantes se realizan por una lógica externa (fig.3.17)

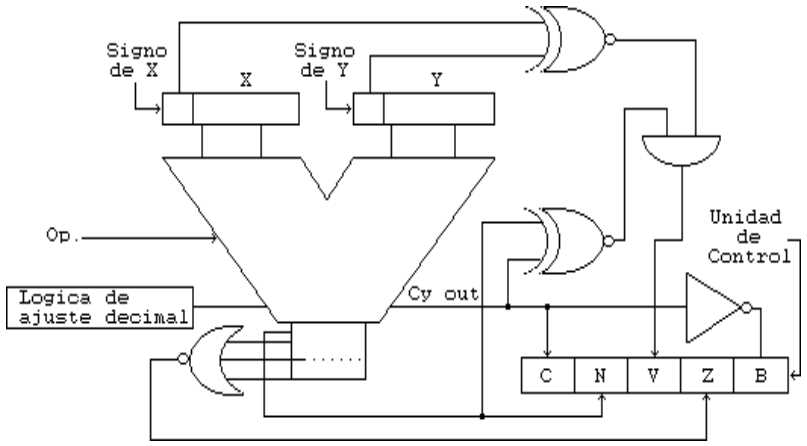


FIGURA 3.17

<u>Unidad de Control</u>

Esta unidad de la CPU incluye:

-Decodificación de instrucción.

-Temporizado.

-Circuitos de Control.

La función de esta unidad es la de determinar la secuencia de tareas a realizar, y la de generar el temporizado de las señales de control necesarias para ejecutar una determinada instrucción.

- Decodificación: significa interpretación de la instrucción; es decir, conocer qué señales de control intervienen en la ejecución de una instrucción y su secuencia de generación.
- Temporizado: significa determinar cuánto tiempo (medido en períodos de reloj) cada una de las señales de control debe estar activa.

Cada instrucción se trae de memoria y se carga en el IR, la cual es decodificada por el "decodificador de instrucciones". La unidad de control, además de generar y testear las señales de control para manejar el flujo de información en el interior de la CPU, debe generar y testear un cierto número de señales de control (que conforman el bus de control) que le permitan gobernar y "conocer" las operaciones que se desarrollan en su exterior (Ejemplo: búsqueda del código de operación, operaciones de lectura/escritura en memoria, etc.)

Las líneas más comunes a cualquier procesador son:

. Reset
. Enable
. R/W
. Ready (wait)
. Hold
. Interrupción

Reset: Esta señal carga al PC con una dirección fija, en la cual se halla la primera instrucción a ser ejecutada. Usualmente, en esa dirección se coloca una memoria ROM, y se la denomina *Vector de Direción Reset.*

Enable:Indica que se está realizando una transferencia en el canal de datos. En otras palabras, que la información en el canal de datos es válida.

R/W: Señala la dirección de la transferencia (operación de lectura o de escritura). A veces, está desdoblada en 2 líneas R y W con sincronismo incluido.

Ready: Esta señal le indica al procesador que debe suspender la ejecución de

la instrucción, manteniendo el canal de datos, direcciones y control sin cambios por un tiempo indeterminado. Se utiliza para conectar dispositivos con tiempo de acceso mayor al de la CPU.

Hold: Esta señal pone al canal de datos, direcciones y algunas líneas del canal de control en tres estados. Se usa en operaciones de I/O mediante acceso directo a memoria (DMA), o sea, cuando alguien inteligente va a acceder al canal.

Interrupciones: Esta señal se utiliza para "interrumpir" la ejecución secuencial de un programa, estableciendo una dirección en el PC para realizar un salto a una dirección de memoria donde se halla una rutina, llamada de "Servicio de Interrupción", que debe ser ejecutada con urgencia. Una vez ejecutada esta rutina, debe retornar al programa que se estaba ejecutando.

Clock: reloj que determina el temporizado interno y externo.

3.4 REPRESENTACION DE DATOS E INSTRUCCIONES

En una computadora, la unidad más elemental (primitiva) de información es el *bit*. Debido a la poca información que se puede manejar con un solo bit, la unidad primaria de información en una computadora es un grupo de bits ("String") denominado palabra (word) de computadora. El ancho de la "palabra" de una computadora puede ser de 4 (nybble), 8 (byte), 16, 32 bits. Un procesador de 8 bits, procesa datos e instrucciones de 8 bits de ancho de palabra. Por supuesto que el bus de datos del procesador debe ser igual de ancho que la palabra de memoria.

Los datos almacenados en la memoria de un computador se pueden clasificar en 3 tipos (dependiendo de qué interpretación se le dé a la misma):

- palabras de datos binarios puros,
- palabras de datos codificados,
- palabras de instrucción.

- Palabras de Datos Binarios

Son palabras que representan una cantidad numérica en el sistema binario. Ejemplo: $01100001_2 = 97_{10}$ puede ser cualquier información dependiendo del significado de la información

Datos con signo: no sería muy útil contar con números positivos únicamente, por lo tanto es muy usual trabajar en complemento a 2.

Datos de varias palabras (de ancho) Multiword: muy a menudo, el rango posible de una dada longitud de palabra puede ser excedido. Por ejemplo, si se necesita trabajar en complemento a 2 con una palabra de 8 bits -un rango de -127 a 127, pero con 16 bits (doble palabra- un rango de -32767 a 32767). Para lograr esto, es usual utilizar lugares consecutivos de memoria para cada dato. Por ejemplo: 1000100101101010.

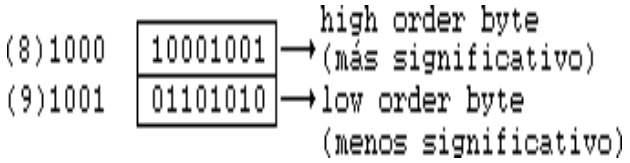


FIGURA 3.18

Representación Octal y Hexadecimal: es usual utilizar este tipo de representación para la entrada y "display" de datos por medio del usuario de la computadora.

- Palabras de Datos Codificados

Los datos procesados por una computadora no deben ser únicamente binarios. Es usual la utilización del *código BCD* donde cada cadena de 4 bits representa a un número. En una palabra de 8 bits se pueden representar 2 números BCD. Muchas computadoras tienen instrucciones para el manejo de números en BCD.

Los datos almacenados en la memoria pueden ser caracteres alfa- numéricos, tal como el *código ASCII* (7 bits datos + 1 de paridad).

Una palabra de datos cualquiera ($01100001_2 = 61_H = 97_{10} = a_{ascii} = 61_{BCD}$) puede tener numerosas representaciones y su interpretación depende exclusivamente de las intenciones del programador.

- Palabra de Instrucción

Hasta ahora se han examinado las palabras de datos. Ahora se verá la representación de las órdenes (comandos o instrucciones) del microprocesador.

Un concepto importante en la arquitectura de Von-Newmann es que los datos e instrucciones ocupan la misma área de memoria. Externamente, no hay forma de distinguir datos de instrucciones, sin embargo, si el PC apunta a una instrucción, la cual es decodificada, su interpretación provee la referencia para la próxima instrucción, distinguiendo entre datos e instrucciones a lo largo de toda la

memoria de programa. Por lo tanto, una instrucción deberá proveer a la Unidad de Control de la siguiente información:

- tipo de operación a realizar,
- información referida a los argumentos (variables):
  - .tipo de direccionamiento,
  - .cantidad de bytes de la instrucción,
  - .con registro o con memoria,
- dónde almacenar los resultados.

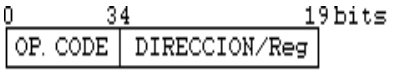
Nota: se entiende por *argumento* a los datos, direcciones o registros involucrados en la instrucción.

Los códigos de las instrucciones son distintas de máquina a máquina, sin embargo, las palabras de instrucción de cualquier máquina se dividen en 2 unidades básicas:

1_ El Código de Operación (op. code): el cual determina la operación a ejecutar y provee, además, los distintos modos de direccionamientos.
2_ La Dirección del Operando: la cual especifica la dirección de memoria con cuyo contenido se va a operar. Esta parte puede o no existir, dependiendo del tipo de instrucción. La dirección del operando puede ser directamente el dato a operar (MVI A, DATO). Las instrucciones pueden estar contenidas en:

a- una palabra,
b- varias palabras de memoria.

a- en una palabra:



*16 bits
*16 bits de direccionamiento
*memoria de 20 bits de palabra

FIGURA 3.19

requiere menos lugares de memoria, pero mayor longitud de palabra;

b- varias palabras: como la mayoría de los microprocesadoresmanejan memoria de palabra de datos de 8 bits, y siendo necesario el OP CODE de 8 bits (esto es válido para microprocesadores de 8 bits) de longitud para tener un buen 'set'

(conjunto) de instrucciones, se determinan 3 tipos de formatos de instrucciones:

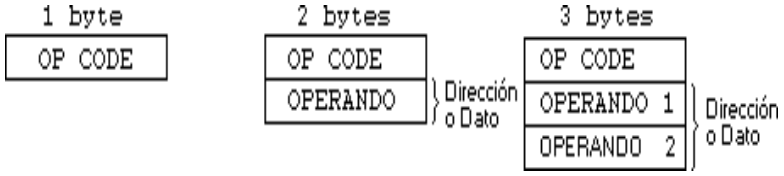


FIGURA 3.20

3.4.1 Formato de Instrucciones

Un OP CODE de 8 bits genera 2^8=256 tipos distintos de instrucciones. Sin embargo, el número de instrucciones de un microprocesador es menor (aproximadamente 200). Generalmente el OP CODE se divide en varios campos (conjuntos de bits) dependiendo su significado de cómo fue diseñada la unidad de control para interpretarlos. Por ejemplo

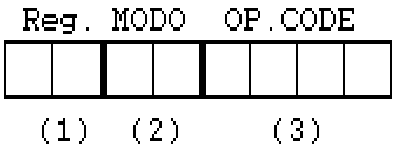


FIGURA 3.21

(1) Selecciona el tipo de registro a operar.
(2) Selecciona el Modo de operaciones.
(3) Selecciona el código de operaciones propiamente dicho (ADD, SUB,...).

El tipo de formato de campos del OP CODE de la fig. 3.21 no es único para cada microprocesador, sino que tiene distintos formatos. Generalmente, el número de formatos es pequeño, pues aumenta la complejidad del hardware de la unidad de control.

De acuerdo al formato de las instrucciones es posible distinguir dos arquitecturas bién diferenciadas:

- Procesador orientado a registro (8085)
- Procesador orientado a memoria (6800)

Un procesador orientado a registro es aquél que usa el formato de las instrucciones para direccionar sus registros internos, es decir, que la mayoría de

las instrucciones requieren acceder, al menos, a un argumento que es un registro.

Un procesador orientado a memoria es aquél que usa el formato de las instrucciones para seleccionar los distintos modos de direccionamiento de memoria, es decir, la mayoría de las instrucciones requieren acceder, al menos, a un argumento de memoria.

Los procesadores de 8 bits, con una longitud de palabra limitado, deben elegir entre uno u otro modo entre las distintas arquitecturas. Sin embargo, los procesadores de 16 bits tienen un ancho de palabra suficientemente grande para acomodar las distintas arquitecturas.

3.5 LENGUAJES DE PROGRAMACION

3.5.1 Lenguaje Máquina (LM)

Para que un programa pueda ser ejecutado por una computadora es necesario, de alguna forma, traducirlo al único lenguaje que una computadora entiende, el lenguaje de ceros y unos, llamado *lenguaje máquina*. Al programa en lenguaje máquina almacenado en memoria, se lo denomina *programa objeto.*

La programación en lenguaje máquina en forma binaria es muy tediosa. Si se usa notación hexadecimal u octal, en lugar de ceros y unos ($00100100_2 = 24_H = 044_8$), es posible reducir, en cierta manera, el número de errores cometidos al programar.

3.5.2 Lenguaje Ensamblador (Assembler)

Definición

Este lenguaje es la expresión simbólica (mnemónica) del lenguaje máquina. Utiliza caracteres alfanuméricos para expresar cada una de las instrucciones y no su forma binaria o hexadecimal, lo cual lo hace mucho más fácil de recordar.

$00100100_2 = 24_H$ MOV A,B transfiere el registro B el contenido del reg. A

A <--- B

Función

Es la de traducir programas escritos en lenguaje mnemónico a su código máquina ejecutable.

Ventajas

Las direcciones se pueden referir como nombres o etiquetas (labels), haciendo innecesario el cálculo de las direcciones de saltos; provee códigos de operación simbólicos, mayor facilidad en la realización de programas, facilidad en la modificación, etc.

Características

Las características más importantes son:

1_ Expresión simbólica de las instrucciones.
2_ Expresión simbólica de las direcciones: referencias como "labels" o etiquetas.
3_ Conjunto de Directivas o Pseudo-operaciones.
4_ Direccionamiento relativo.
5_ Definición de Macro instrucciones. Subrutinas.
6_ Ensamblado condicional.
7_ Generación de código reubicable.
8_ Vinculación externa de programas.
9_ Segmentos.

Sintaxis

Una sentencia escrita en assembler consta de 4 campos:

Label	Código de operación	Operando	Comentarios
(1)	(2)	(3)	(4)

(1) Define en forma simbólica la dirección donde se ha de ensamblar la instrucción.
(2) Este campo puede ser:
- mnemónico de la instrucción, que se traduce a código máquina y es

interpretada por el microprocesador en tiempo de ejecución.
- directiva o pseudo-operación, que se interpreta por el ensamblador y se ejecuta en tiempo de ensamblado.
(3) Referencia simbólica o absoluta de los operandos que intervienen en la operación. Estos pueden no existir, o tener uno, dos o tres operandos, separados por comas [,]. Finalmente, el operando puede ser una expresión aritmética y/o lógica (+ - *.AND..OR..MOD. < >).

Directivas

Una directiva es una instrucción para el ensamblador, que éste usará mientras genera código objeto. O sea, no es una orden para el microprocesador, por lo tanto no se traduce a código máquina. Esta se ubica en la misma posición que las instrucciones.

Las directivas o seudo-instrucciones más comunes son:
EQU, END, NAME, ORG, DB, DW, DS, PUBLIC (o GLOBAL), EXTERN, LOCALS.

Directiva EQU:

Esta directiva asigna a un símbolo, una constante (o una dirección o una expresión), es decir, cuando se encuentre dicho símbolo en el programa fuente, se usará la constante equivalente.

```
símbolo     EQU   expresión
```

La expresión puede especificar una constante, una dirección, un registro o aún el mnemónico de una instrucción. El símbolo especifica el nombre que se empleará para representar la expresión.

Directiva ORG:

Esta directiva especifica el lugar de memoria absoluto, a partir del cual comenzará a ensamblarse el programa. Es decir, determina el origen del programa cuando se lo carga en memoria. En un programa puede haber más de una directiva ORG.

```
            ORG   expresión
```

La expresión especifica el nuevo valor del contador de programa.

Directiva END:

Esta directiva identifica el fin del archivo fuente. Por lo tanto, se debe ubicar al final del mismo.

```
                END   expresión (opcional)
```

La expresión especifica la dirección del programa desde la cual comenzará la ejecución.

Directivas DB,DW,DS:

Estas directivas se emplean para realizar reservas de lugares de memoria, especificando el tipo del símbolo (byte, doble byte,etc.).

```
     símbolo     DB    expresión
                 DW
                 DS
```

El símbolo especifica el nombre de la variable, la cual tiene asignado un lugar de memoria en la posición de ensamblado. La expresión determina el valor inicial del símbolo. La directiva DB reserva bytes de almacenaje, y la DW, dos bytes. DS reserva la cantidad de bytes determinada por la expresión.

Ejemplo de directivas de micros de 8 bits

6800	8085
END : End of Program	END
EQU : Equate Symbol	EQU
NAM : Name of Program	NAME
RMB : Reserve Memory Bytes	DS : Define Storage
ORG : Origin	ORG
FCB : Form Constant Bytes	DB : Define Byte (1)
FCC : Form Constant Characters	DB : Define Byte (2)
FDB : Form Double Bytes	DW : Define Word (3)

(1) Almacena byte dato en consecutivos lugares de memoria.

(2) Idem (1) en ASCII.

(3) Idem (1) con 2 bytes.

```
                    ORG  2000  : origen del programa. El programa se ensambla
                                  a partir de la dirección 2000.
               PATO EQU  28  : asigna a la variable PATO el valor 28.
```

```
                    ORG  2000
       2000         LDA  C,PATO
                    MOV  A,C
             LAZO:  ADD  C
                    DCR  C
                    JNZ  LAZO
```

Ejemplo:

```
                ORG             1000H
VAR1            DB              00H
TABLA           DB              0,0,0,0,4,F,0,8H
LONGT           EQU             $  -  TABLA

                ORG             2000H
                MVI             C, LONGT                    INIC. CONTADOR
                LDA             VAR1
                LXI             H,TABLA                     INIC. PUNTERO
;
ACA:            CMP             M
                INX             H
                RZ
                DCR             C
                JNZ             ACA
                RET
;
                END
```

Direccionamiento Relativo:

Esta característica permite realizar un direccionamiento relativo a la posición actual del contador del programa. En el Assembler, el Contador de Programa se especifica por $ ó *.

```
               JMP $+8      (salta 8 lugares adelante de la dirección de
                            ensamblado de  la instrucción JMP).
```

Macroinstrucciones y Subrutinas

Cuando un programador tiene que utilizar un grupo de instrucciones varias veces dentro del programa principal, puede generar un subprograma, el cual será llamado todas las veces que sea necesario, en lugar de recodificarlo reiteradamente. Para ello dispone de dos alternativas: Subrutinas o Macroinstrucciones.

- Subrutinas:

Una subrutina es un conjunto de instrucciones, que ocupan un determinado lugar de memoria, y que pueden ser ejecutadas desde distintos puntos del programa. Para ello se dispone de instrucciones especiales, CALL, que quiebran la ejecución secuencial del programa, transfiriendo el control a ese conjunto de instrucciones (subrutina), guardando el contenido del contador de programa en la pila. La última instrucción de las subrutinas es el código RETURN, que retorna el control a la instrucción siguiente al CALL que produjo el llamado. Una subrutina es un mecanismo soportado por el microprocesador.

- Macroinstrucción:

Una macroinstrucción es un nombre definido por el usuario que asocia un símbolo a una secuencia de instrucciones. Este nombre es usado como una nueva instrucción y aparece como un código de operación en la sintaxis del assembler. El ensamblador se encarga de insertar dicha secuencia de instrucciones en el programa fuente cada vez que encuentre el nombre de la macro (expansión de la macro).

```
símbolo     MACRO       parámetros
```

El símbolo es el nombre de la macro, y los parámetros son los argumentos que ella emplea en el desarrollo.

```
LABEL   MACRO  X,Y,Z
        MOV    A,X
        RLC
        Y
Z       JNZ    FIN
        AND    0F
```

```
FIN     NOP
        ENDM
```

La diferencia entre macro y subrutina es una relación de compromiso entre tiempo y memoria. La ejecución de una macro es más rápida que la equivalente de una subrutina. Esto se debe a que no es necesario el uso de instrucciones CALL y RET, para las llamadas. Sin embargo, cuando el tiempo no es un factor primordial, el uso de subrutinas es una mejor elección, sobre todo desde el punto de vista de ahorro de memoria.

Ensamblador Condicional

Algunos ensambladores permiten ignorar o incluir una parte del programa dependiendo de las condiciones existentes en el tiempo de ensamblado (como el resultado de la evaluación de una expresión). El ensamblado condicional se controla por medio de directivas o pseudo-instrucciones como IF...ENDIF; EQUAL; etc.

```
IFZ   A     si A = 0, entonces ensambla
 .
 .
ENDIF
```

Generación de Código Reubicable:

Un programa objeto generado por el ensamblador puede ser "absoluto" o "reubicable". Se dice "absoluto", cuando está ensamblado en las direcciones de memoria donde va a ser cargado y ejecutado. Se dice "reubicable", cuando se puede generar código de programa, independientemente de las direcciones de memoria absolutas en las cuales residirá finalmente el mismo. Las direcciones finales de memoria se calculan en tiempo de carga por un programa conocido como cargador-vinculador (loader-linker).

Una de las ventajas más importantes del empleo de código reubicable, es que permite crear programas ensamblados en forma independiente uno del otro (o módulos de programa), y luego ser vinculados (link) para formar una unidad ejecutable, que pueder ser cargable en memoria.

En el momento de carga se resuelven las referencias de un módulo a otro. La

conexión entre los módulos se realiza a través de las referencias y definiciones externas. Las definiciones externas se especifican por medio de la directiva GLOBAL, quedando disponibles para el uso en otros módulos. Las referencias externas se especifican por medio de la directiva EXTERN, y son aquéllas que se emplean en el mismo módulo, pero fueron definidas en otro.

Segmentos

Un segmento de programa es aquella parte que posee su propio contador de programa y es lógicamente distinguible. En el momento de ensamblado, las direcciones para cada segmento se pueden especificar por separado. Los tipos de segmentos son:

- Segmento de código (CODE).
- Segmento de datos (DATA).
- Segmento de pila (STACK).
- Segmento absoluto.

Naturalmente, el lenguaje Ensamblador no puede ser cargado en memoria sino que debe ser traducido a lenguaje máquina. Ello lo realiza el programa llamado "Assembler". Este puede ser ejecutado en el mismo procesador o en otra computadora; en este último caso lo se llama "Cross-Assembler".

Tipos de Instrucciones

Las instrucciones de los microprocesadores se encuentran dentro de las categorías siguientes:

1- Instrucciones Aritméticas.
2- Instrucciones Lógicas.
3- Instrucciones de Desplazamiento (Shift).
4- Instrucciones de Transferencia.
5- Instrucciones de Entrada-Salida.
6- Instrucciones de Salto.
7- Instrucciones de Manejo de Bit de Status.
8- Instrucciones de Pila.
9- Instrucciones de Control de Máquina.

10- Instrucciones de Búsqueda en Tablas.

1- Instrucciones Aritméticas

Son las relativas a las operaciones aritméticas. Este tipo de instrucciones afecta a los flags (se actualizan, pues utilizan la ALU). La cantidad de bytes de estas instrucciones puede ser 1 o 2.

Mnemónico	Descrip.	Mnemónico	Descrip.
ADD B	A <-- A+B	INC A	A <-- A+1
ADC B	A <-- A+B+Cy	DEC A	A <-- A-1
ADI DATO	A <-- A+DATO	COMP B	A <-- B
SUB B	A <-- A-B	ADD M	A <-- A+(M)
SBB B	A <-- A-C-B	SUB M	A <-- A-(M)
SBI DATO	A <-- A-DATO	MUL M	A.Rg

2- Instrucciones Lógicas

Son las relativas a las operaciones Booleanas. Este tipo de instrucciones afecta a los flags (se actualizan, pues utilizan la ALU). Estas se realizan bit a bit entre operandos. La cantidad de bytes de estas instrucciones puede ser 1 o 2.

Mnemónico	Descrip.	Mnemónico	Descrip.
CLR A	A <-- 0	CMP A	A <-- A+1
AND B	A <-- A.B	SET A	A <-- FF
ANI DATO	A <-- A.DATO	XNOR B	A <-- A.EXOR.B
ORA B	A <-- A+B	AND M	A <-- A.M
ORI DATO	A <-- A+DATO	ORA M	A <-- A+M
XOR B	A <-- A.EXOR.B		

3- Instrucciones de Desplazamiento

Estas instrucciones permiten desplazar los acumuladores hacia ambos lados. Se utilizan en operaciones aritméticas (Multiplicación). Muy importante para conocer el estado de un bit en particular. Cada instrucción realiza el desplazamiento de 1 bit. La cantidad de bytes de estas instrucciones es normalmente 1.

Mnemónico	Descrip.

RAL	Rotar a izq. sin Cy (fig.3.22a)
RLC	Rotar a izq. con Cy (fig.3.22b)
RAR	Rotar a der. sin Cy
RRC	Rotar a der. con Cy
ASR	Desplazamiento aritmético a la derecha (fig.3.22c)

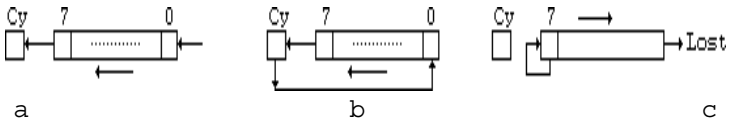


FIGURA 3.22

4- Instrucciones de Transferencia

Incluyen las instrucciones de cargar registro (LOAD), almacenar registro (STORE) y mover contenido de un registro a otro (MOVE). Se usan generalmente para inicializar. La cantidad de bytes de estas instrucciones puede ser 1, 2 o 3.

Mnemónico	Descrip.
LDA ADD	A <-- (ADD) (acceso a variable de 8 bits)
LXI H,ADD	HL <-- ADD (inicialización de reg. de16bits)
STA ADD	(ADD) <-- A (acceso a variable de 8 bits)
MOV A,B	A <-- B
MVI A,DATO	A <-- DATO (inicialización de reg. de 8bits)
LDX ADD	Cy <-- ADD
XCHG Reg	(Intercambio entre reg. HL <-->DE)
XCHG M	
LHLD ADD	HL <-- (ADD) (acceso a variable de 16 bits)

5- Instrucciones de Entrada/Salida

Los dispositivos externos se pueden conectar de 2 formas:

I) Las direcciones de los dispositivos son parte del mapa de memoria.

II) Los dispositivos tienen su propio espacio de direccionamiento, los cuales se seleccionan por líneas (IO) especiales, que se manejan por instrucciones de I/O.

FIGURA 3.23

```
IN  PORT      A <-- (PORT)
OUT PORT      A --> (PORT)
```

La cantidad de bytes de estas instrucciones es generalmente 2.

No todos los procesadores tienen esta característica.

6- Instrucciones de Salto

Las instrucciones de salto son las que alteran el PC, por lo tanto se cambia el flujo de ejecución del programa. Estas se dividen en 2 tipos:

I) Instrucciones de salto sin guardar el PC en la pila:

a) Saltos condicionales: la toma de decisiones en un programa se lleva a cabo por los saltos condicionales.

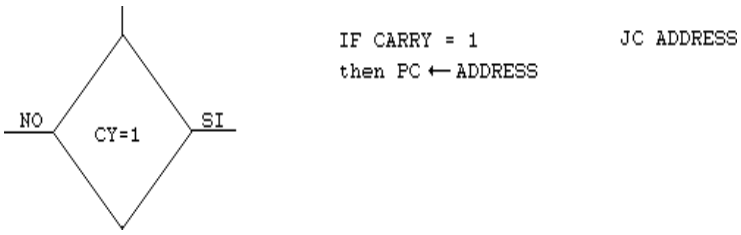


FIGURA 3.24

```
JC  ADD      salta si Cy=1
JZ  ADD      salta si Z=1
JM  ADD      salta si N=1
JP  ADD      salta si Paridad
JGE ADD      salta si resultado >=0
JLE ADD      salta si resultado <=0
JNC
INC
```

JPLUS

JPO,JPE

Dentro de este grupo se puede agregar la DJNZ.reg, DIR, que hace:

Decrementa Reg en 1 Reg = Reg-1

Si Z = 1 --> PC=DIR debe decir distinto

Si Z = 1 --> sigue

b) Saltos incondicionales: JMP ADD PC <-- ADD

II) Instrucciones de salto que guardan el PC antes de saltar: salto a subrutina. Si una secuencia de instrucciones se debe ejecutar repetidas veces en un programa, a menudo se realiza en un módulo independiente llamado subrutina. (Ej.: multiplicación). Cada vez que el programa necesita un resultado de esa tarea, llama por medio de instrucciones especiales (CALL). Cuando el microprocesador termina su ejecución, la vuelta al programa principal se realiza por instrucciones de retorno de subrutina (RET). La dirección de retorno de la subrutina se almacena en la Pila.

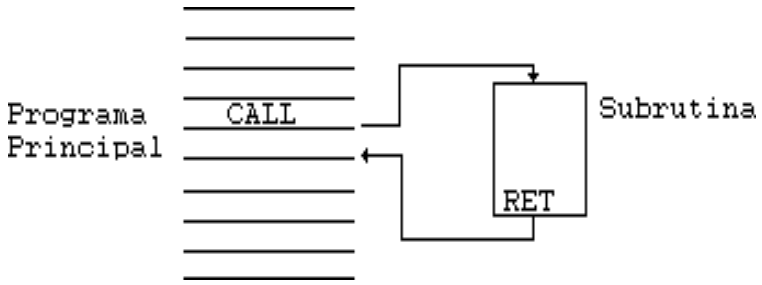


FIGURA 3.25

CALL ADD PC --> (SP), SP <-- SP-1

PC <-- ADD

RET PC <-- (SP+1), SP<--SP+1

Saltos relativos: los saltos relativos son aquéllos que bifurcan el programa a direcciones relativas a la posición actual del contador de programa. El operando es un número en complemento a 2, permitiendo desplazamientos hacia adelante o hacia atrás de dicha posición.

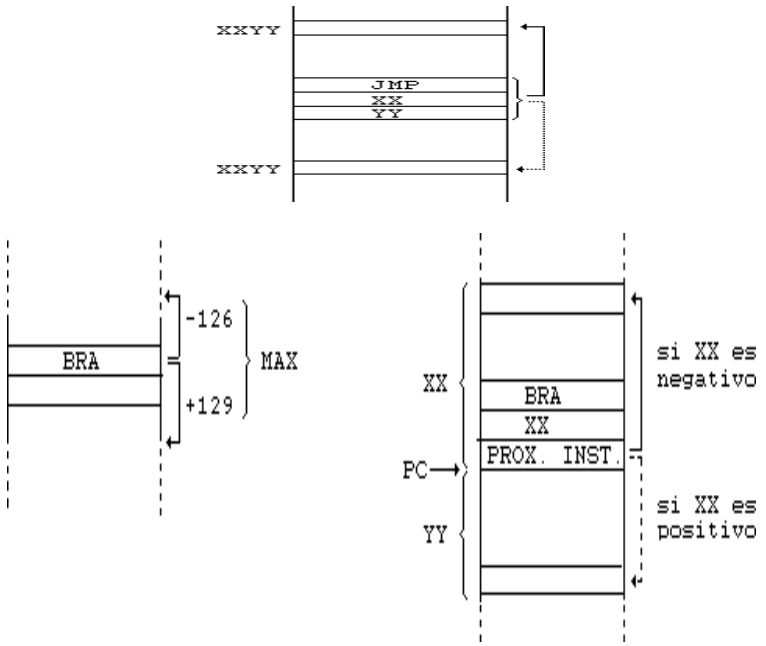


FIGURA 3.26

7- Instrucciones de Manejo de Bit de Status

Este tipo de instrucciones se refieren al empleo (manejo) de los bits del Registro Código de Condición (RCC): Cy, Z, V, S, P, etc. A través de estas instrucciones se pueden establecer los bits individualmente como:

```
SET   CARRY    CY=1
CMPL  CARRY    /CY
CLEAR CARRY    CY=0
SET   CARRY    set overflow
```

Dentro del RCC hay, en algunos microprocesadores, bits relacionados con el manejo de interrupciones como en el caso del 6800, que contiene el bit Set Interrupt Mask, para habilitar el uso de las interrupciones.

8- Instrucciones de Pila

Una pila es un área de memoria RAM que se emplea para almacenaje temporario de los registros de la CPU. Se maneja por medio del SP.

```
PUSH  A          (SP) <-- A;   SP <-- SP-1
POP   A         A <-- (SP+1); SP <-- SP+1
XTHL              HL <--> (SP)
```

9- Instrucciones de Control de Máquina

Estas instrucciones están relacionadas con el control de la ejecución de un programa. Están incluídas en este grupo:

NOP	No operación
HALT	Detiene ejecución

Suelen agruparse en este tipo, las instrucciones de manejo de interrupciones, pues, de alguna manera, intervienen en el control de la máquina.

EI	habilita interrupciones
DI	deshabilita interrupciones
SIM	establecimiento de máscara de interrupciones

10- Instrucciones de Búsqueda en Tablas (Look Up Table)

Son aquellas instrucciones que buscan un dato en una tabla por medio de un único acceso a la misma.

XLAT	AL <-- ((Bx)+AL)	(8086)

En el microprocesador 8031:

MOV A, @A+DPTR	A <-- (DPTR + A)
MOV A, @A+PC	A <-- (PC + A)