NOCIONES FUNDAMENTALES SOBRE LA ARQUITECTURA Y ORGANIZACIÓN DE LOS COMPUTADORES DIGITALES SECUENCIALES

ROBIRO ANTONIO ASUAJE LABARCA

UNIVERSIDAD CENTRO OCCIDENTAL “LISANDRO ALVARADO”

BARQUISIMETO, 1997

NOCIONES FUNDAMENTALES SOBRE LA ARQUITECTURA Y ORGANIZACIÓN DE LOS COMPUTADORES DIGITALES SECUENCIALES

Por

Robiro Antonio Asuaje Labarca

Trabajo de Ascenso presentado para optar
a la categoría de Agregado en el escalafón
del personal Docente y de Investigación.

UNIVERSIDAD CENTRO OCCIDENTAL “LISANDRO ALVARADO”
Decanato de Ciencias

Barquisimeto, 1997.

NOCIONES FUNDAMENTALES SOBRE LA ARQUITECTURA Y ORGANIZACIÓN DE LOS COMPUTADORES DIGITALES SECUENCIALES

Por

Robiro Antonio Asuaje Labarca

Trabajo aprobado

_____ _____

Prof. Prof.
Coordinador

_____

Prof.

Barquisimeto, de de 19___

INDICE

AGRADECIMIENTO

A los Ingenieros Ricardo Andrade, Oscar E. Giménez R., Jesús H. Canelón H. por toda la gran ayuda prestada a mi persona en el desarrollo del presente trabajo.

Al Laboratorio de Multimedia del Decanato de Ciencias, en especial a los Ingenieros José Gregorio Sánchez y Alvaro Muñoz y al Sr. César Chávez.

A mis amigos del Equipo Sinergia: Maritza, Carlos, Jorge y Hernán por toda su paciencia y espíritu de colaboración.

"La gratitud es la virtud de los que nada poseen"

NOCIONES FUNDAMENTALES SOBRE LA ARQUITECTURA Y ORGANIZACIÓN DE LOS COMPUTADORES DIGITALES SECUENCIALES

Robiro Antonio Asuaje Labarca

RESUMEN

Un computador digital consta de tres elementos básicos para efectuar sus funciones: (1) La Unidad Central de Procesamiento (CPU) formada por la unidad de control y la unidad aritmético-lógica; (2) La Memoria Principal donde residen los programas y datos y (3) Las Unidades de Entrada y Salida que permiten comunicar a la máquina con los periféricos exteriores.

Comparando al computador con el ser humano, el CPU puede asemejarse al corazón de la máquina porque mantiene el control general y el envío de información a todos los elementos. La memoria hace las veces del cerebro porque es allí donde se almacena la información, y las unidades de entrada y salida actúa como las extremidades y sentidos por los que recibe y entrega la información.

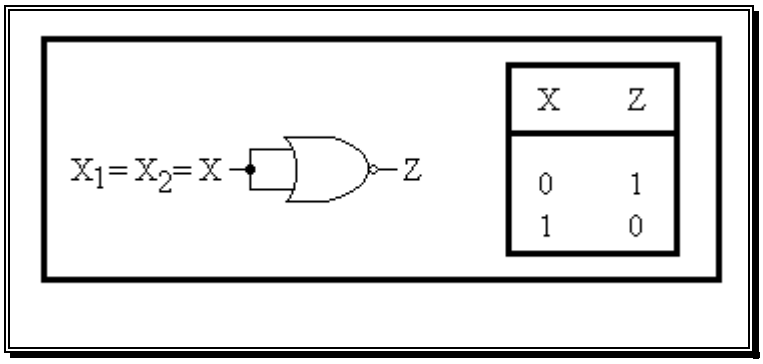
Los tres bloques están comunicados entre sí mediante buses que transportan información binaria del mismo tipo. Específicamente la *unidad de control*, uno de los principales elementos del computador se encarga de determinar la dirección de la memoria o de la unidad de entrada y de salida en la que se realiza el acceso o transferencia de información, utilizando para esta selección, los buses de direcciones. Posteriormente se necesitarán los buses de datos para transferir la información y

finalmente, el bus de control que es el que se encarga de transportar las señales auxiliares de órdenes y sincronización.

La salida de una compuerta NOR es igual 1 solamente cuando las dos entradas son iguales a 0, lo que indica que la salida corresponde a la de una compuerta OR invertida.

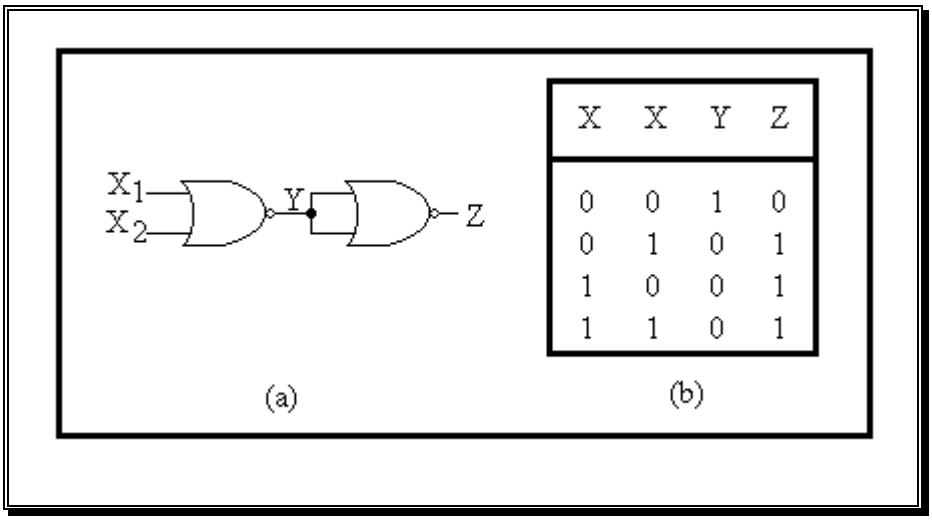
Al igual que la compuerta NAND, la compuerta NOR también constituye una compuerta universal, por lo que con ella se puede establecer cualquier sistema lógico. Por otra parte, veamos cómo mediante el uso de una compuerta NOR se puede realizar una función inversiva.

Fig. 15. (a) Compuerta NOR como inversora.
(b) Tabla de verdad.



La función OR se puede obtener mediante dos compuertas NOR como se indica a continuación:

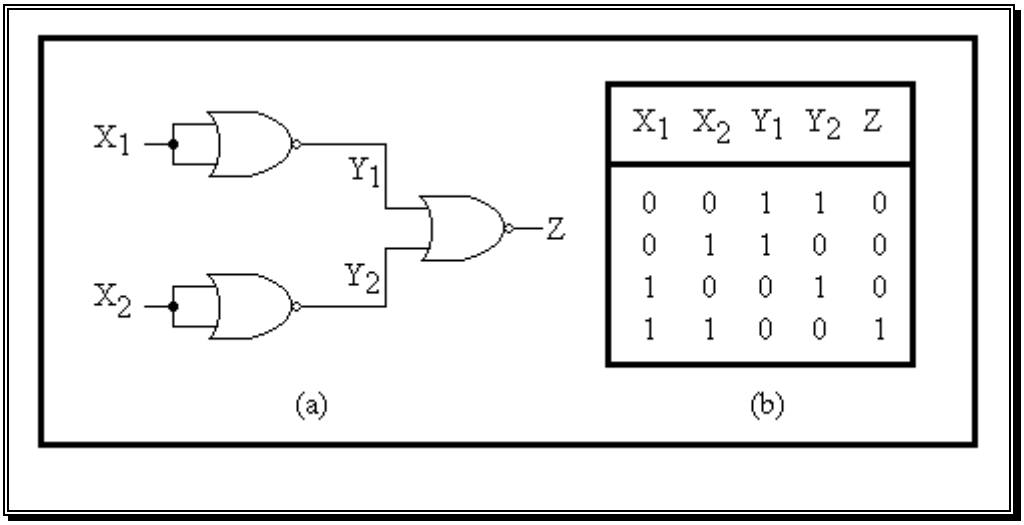
Fig. 16. (a) Uso de dos compuertas NOR para sintetizar la función OR.
(b) Tabla de verdad.



X	X	Y	Z
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

La función AND puede sintetizarse de la forma indicada en la figura 17, donde también se muestra la tabla de la verdad correspondiente.

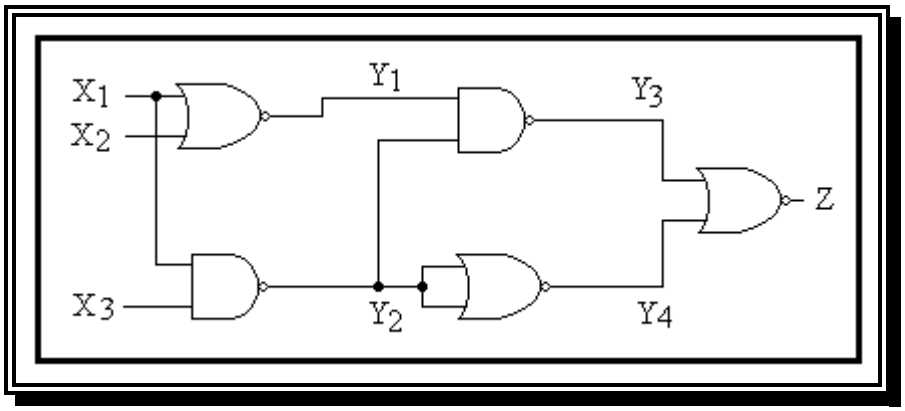
Fig. 17. (a) Implementación práctica de la función AND con compuertas NOR.
(b) Tabla de verdad.



X_1	X_2	Y_1	Y_2	Z
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Ejemplo:

Determine la tabla de la verdad para el sistema lógico mostrado:



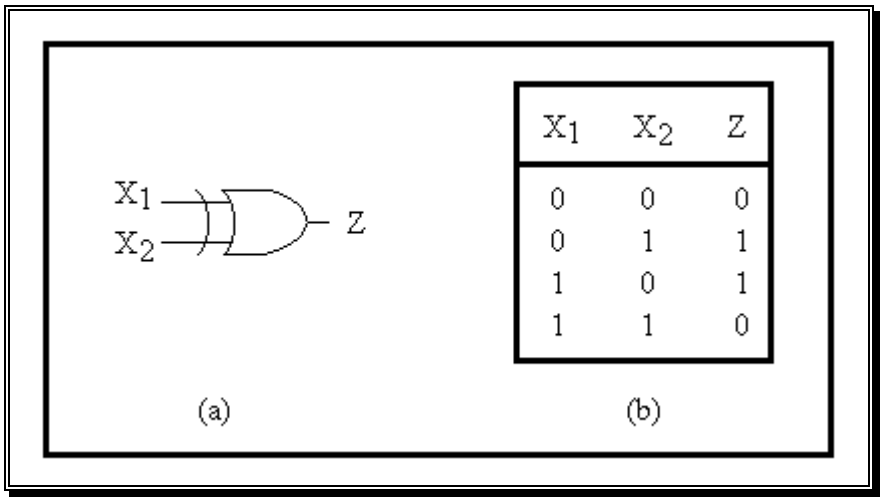
El circuito está formado por compuertas NAND y NOR. En la tabla de la verdad se presentan los valores que asumen las distintas variables:

X_1	X_2	X_3	Y_1	Y_2	Y_3	Y_4	Z
0	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1
0	1	0	0	1	1	0	0
0	1	1	0	1	1	0	0
1	0	0	0	1	1	0	0
1	0	1	0	0	1	1	0
1	1	0	0	1	1	0	0
1	1	1	0	0	1	1	0

1.7 Función Lógica XOR.

La función XOR llamada or-exclusivo, recibe su nombre debido a que arroja como resultado un valor lógico de 1 si solamente una y sólo una de sus variables de entrada tiene el valor de 1. Esto significa que el valor de una variable excluye a la otra.

Fig. 18. Compuerta XOR. (a) Símbolo lógico. (b) Tabla de verdad.

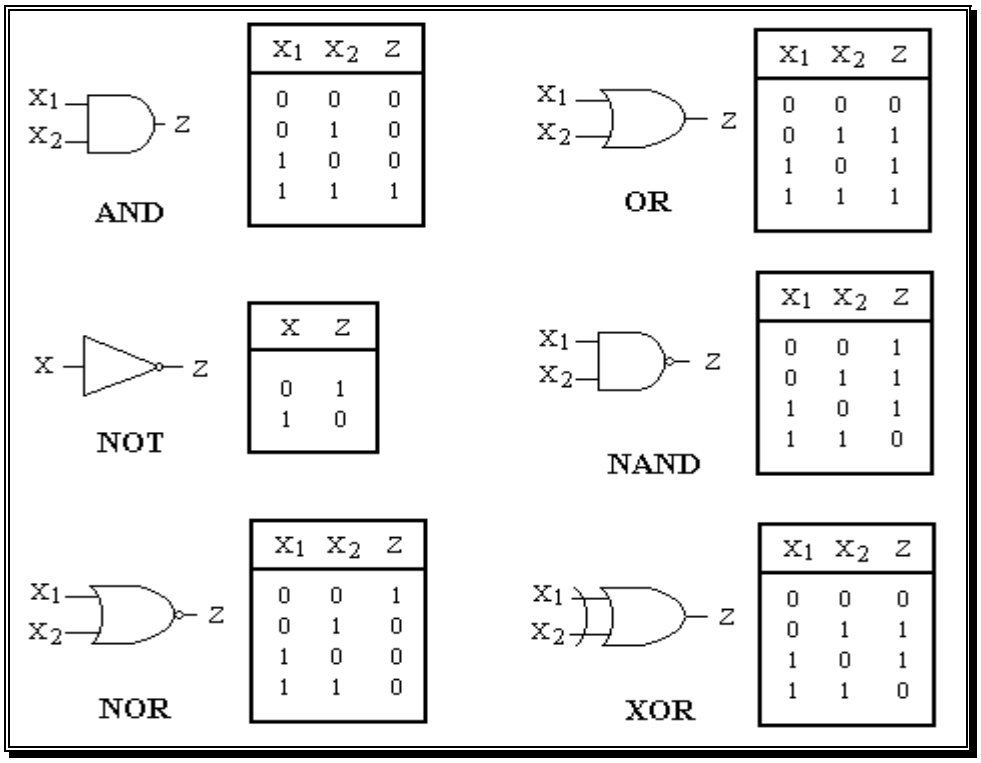


X_1	X_2	Z
0	0	0
0	1	1
1	0	1
1	1	0

Obsérvese que Z = 1 si A = 1 ó B = 1, pero no si ambas A y B son 1.

Para resumir, la figura 19 nos muestra las compuertas lógicas consideradas:

Fig. 19. Compuertas lógicas.



1.8 Circuitos Combinacionales.

Los circuitos lógicos que consideramos anteriormente, se consideran sistemas o circuitos combinacionales. Estos se caracterizan por el hecho de que su(s) salida(s) está(n) completamente determinada(s) por el estado momentáneo de sus entradas. A continuación se estudian algunos otros circuitos de este tipo que se usan normalmente en los sistemas digitales.

1.8.1 Decodificadores

Existen disponibles chip integrados que contienen un gran número de circuitos y que se utilizan en innumerables aplicaciones. En esta sección, mostramos un circuito denominado decodificador.

La característica más importante del decodificador es que para cada entrada A_1A_0, una y sólo una de sus salidas toma el valor 1 lógico; es decir, en cualquier instante una de las salidas (O_1, O_2, O_3, u O_4) se distingue de las demás. El decodificador consta solamente de puertas AND. Las entradas a las puertas se aplican directamente o después de realizar una inversión lógica en los casos que sea necesario. Cuando $A_1 = A_0 = 0$, tenemos $O_0 = 1$, mientras $O_2 = O_3 = O_1 = 0$ etc. Dicho con otras palabras cada una de las posibles combinaciones de entrada de A_1 y A_0 caracterizan una de las salidas, por lo tanto, un decodificador con n entradas necesitaría 2^n puertas y tendría 2^n salidas.

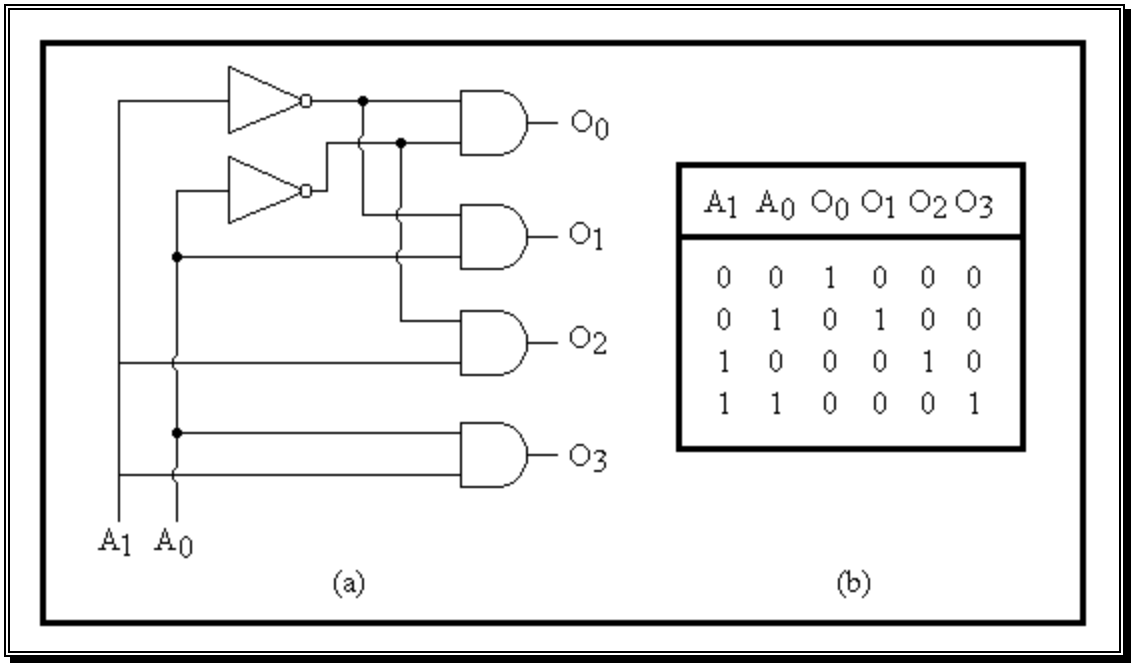
Para apreciar por qué el circuito de la figura 20 se denomina decodificador, supongamos que cada salida se conecta a un circuito individual, que permite que encienda un bombillo cuando el nivel lógico de la salida correspondiente sea 1 y supongamos también que cada bombillo ilumina una pequeña carta con un número del 0 al 3, los cuales son invisibles cuando la luz correspondiente está apagada. Entonces, el efecto neto del circuito sería hacer explícito lo que se presenta implícitamente. Dados los niveles lógicos de entrada podemos determinar fácilmente el número esperado pero si miramos directamente a la salida del sistema, el número esperado se da explícitamente, es decir, la información de entrada dada como una palabra de código ha sido decodificada.

Si es apropiado, no necesitamos suministrar todos los complementos de las 2^n salidas. Por ejemplo, supongamos que las entradas son cuatro líneas usadas para representar dígitos decimales del 0 al 9 en código decimal codificado-binario (BCD). Entonces necesitaríamos 10 luces y el decodificador debería suministrar

solamente 10 líneas de salida en lugar del número posible: $2^4 = 16$; también necesitaríamos 10 puertas AND de las que algunas no tendrán cuatro entradas, ya que hay condiciones irrelevantes correspondientes a los números del 10 al 15, que nunca se presentarán en la entrada. Por otra parte, un decodificador que deba demostrar m salidas individualmente seleccionables debe tener 2^nm.

El símbolo A para las entradas (A_1 A_0) del decodificador se ha escogido para sugerir que los bits de entrada colectivamente constituyen una dirección, es decir, lo mismo que una dirección distingue una casa entre muchas en una ciudad, así la dirección de entrada A_1A_0 señala una línea de salida entre muchas.

Fig. 20. Decodificador: (a) Circuito. (b) Tabla de verdad.



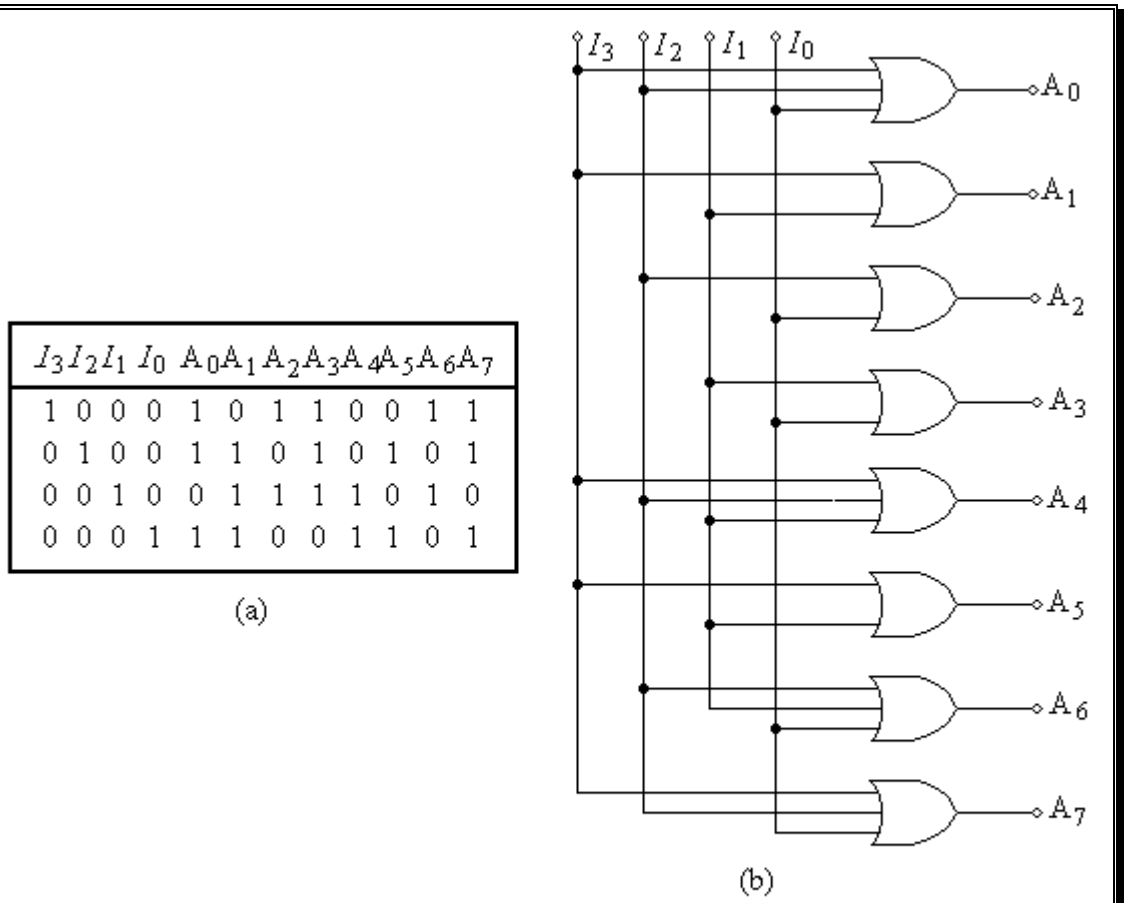
A_1	A_0	O_0	O_1	O_2	O_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

1.8.2 Codificadores.

Un codificador realiza la función inversa a la realizada por un decodificador. Un codificador se diseña para que, entre sus entradas, siempre exista una con un nivel lógico distinto a las demás (las entradas de un codificador son generalmente las salidas de un decodificador). Por cada línea de entrada aparece en las líneas de salida la palabra código correspondiente, cuyos bits son A_0, A_1,... Generalmente no se necesitan relaciones especiales entre el número de líneas de entrada y salida.

La tabla de verdad para un codificador es la de la figura 21 (a) y su implementación es la figura 21 (b).

Fig. 21 Codificador (a) Tabla de verdad. (b) Implementación



I_3	I_2	I_1	I_0	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7
1	0	0	0	1	0	1	1	0	0	1	1
0	1	0	0	1	1	0	1	0	1	0	1
0	0	1	0	0	1	1	1	1	0	1	0
0	0	0	1	1	1	0	0	1	1	0	1

(a)

Fuente: Taub. 1990

Hemos supuesto cuatro líneas de entrada que en la salida generan cuatro palabras código de ocho bits, A_0, A_1, ..., A_7. En principio, las palabras son diferentes entre sí aunque no es necesaria esta imposición. Una implementación del codificador se da en la figura anterior. Señalamos por ejemplo, que $A_0 = 1$ cuando $I_3 = 1$ o $I_2 = 1$ o $I_0 = 1$, por tanto, las entradas a la puerta OR cuya salida es A_0 son I_3, I_2 e I_0. Las entradas correspondientes a otras puertas se determinan de forma similar.

Codificador de prioridad.

Los sistemas digitales incluyen frecuentemente componentes para producir señales indicadoras de que es necesario generar alguna acción. Por ejemplo, supongamos que equipamos a un tanque de agua con un mecanismo conmutador que opera cuando el nivel de agua está excesivamente alto y hay peligro de desbordamiento. Fácilmente podemos disponer una línea que en condiciones normales esté en 0 lógico, pero que cambie a 1 lógico cuando opere el conmutador. Entonces el cambio de 0 lógico a 1 lógico es una indicación de que es necesario hacer algo, es decir, hay una petición de servicio. Este mismo sistema entonces puede tener componentes diseñados para suministrar la respuesta adecuada: atender la petición. El servicio suministrado en el caso del tanque de agua podía consistir en cerrar una válvula de entrada, abrir una válvula de salida, etc. Frecuentemente, en un sistema digital, hay líneas para peticiones de servicio y consecuentemente una serie de componentes del sistema que pueden prestar esos servicios. Los componentes que se activan dependen de la línea que presente la petición de servicio. Generalmente, cada componente de servicio se distingue de los demás por su dirección, la cual es un conjunto distintivo de bits (un código de dirección), que mediante un decodificador acceden a un componente de servicio.

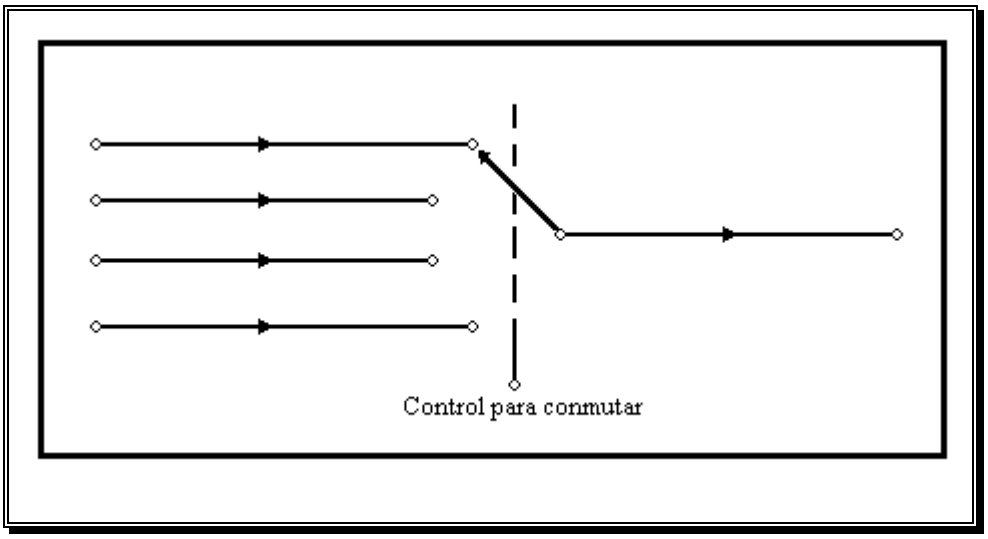
Se utilizará normalmente un codificador para aceptar como entrada las líneas de petición de servicio y obtener como salida el código de bits correspondiente a la dirección del componente que atenderá la petición, ya que el codificador se diseña para que cada vez que tenga una entrada diferente se atienda solamente una petición de servicio. Esta situación es aceptable, ya que generalmente los componentes que suministran el servicio tienen elementos comunes y por ello cada vez se puede servir solamente una petición.

Supongamos, sin embargo, que se generan dos o más peticiones de servicio al mismo tiempo. Esta situación se resuelve asignando una prioridad a cada línea de petición de servicio, si hay más de una petición simultáneamente. La salida del codificador direccionará el componente de servicio correspondiente a la petición de más alta prioridad y cuando aquélla haya sido atendida, la línea correspondiente de petición de interrupción volverá al 0 lógico y se atendería la siguiente petición de mayor prioridad.

1.8.3 Multiplexores.

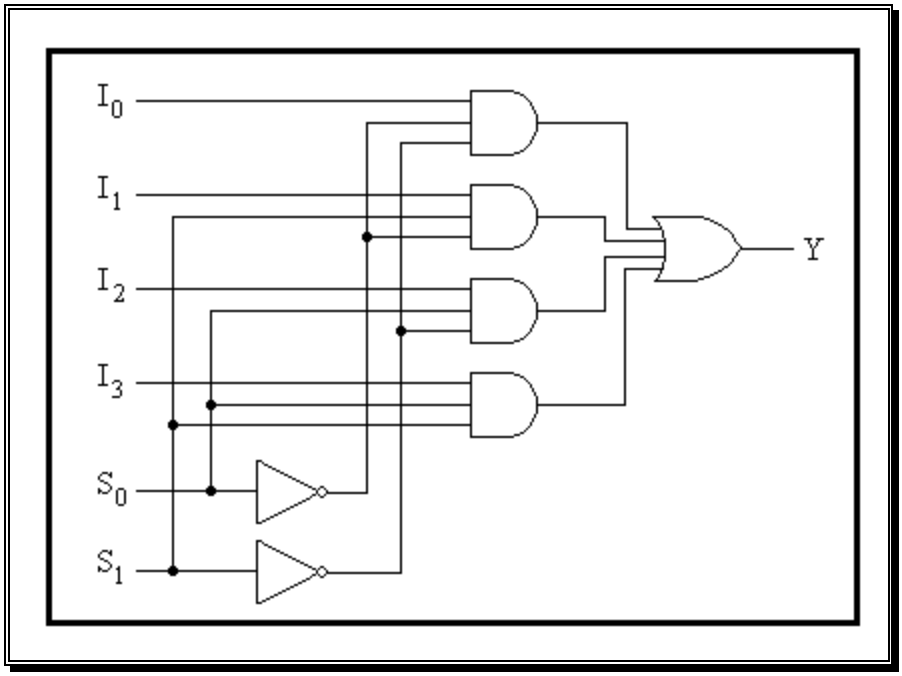
Un multiplexor realiza la función indicada en la figura No. 22, donde disponemos de una serie de líneas de entrada, y una sola línea de salida. La posición del conmutador de posición múltiple que es controlable determinará la señal de entrada que aparezca en la línea de salida. Un multiplexor consta de varias entradas y una salida y mediante un mecanismo de selección, una entrada se transfiere a la salida.

Fig. 22. La función de un multiplexor.



En la figura 23 se muestra la estructura de un multiplexor, el cual consta de cuatro entradas (I_0, ..., I_3) que pueden seleccionarse con los bits de selección S_1S_0.

Fig. 23. Multiplexor de 4 a 1 línea.



1.9 Circuitos Secuenciales.

Los circuitos lógicos cuyas entradas están alimentadas por fuentes externas sin realimentación, donde no existe conexión entre las salidas y las entradas de la puerta, son circuitos combinacionales. En estos circuitos, las salidas están determinadas únicamente por el valor de las entradas. Si en el instante $t = t_0$ cambia alguna de las entradas, consecuentemente las salidas deben de cambiar. Pero las nuevas salidas dependen solamente de las entradas después de $t = t_0$ y no del valor que tuviesen antes de ese instante. E*stos circuitos combinacionales, por lo tanto, no tienen memoria.* Las salidas actuales dependen de las entradas actuales, pero no de los valores de entradas anteriores. Por supuesto, cuando cambian las entradas hay un intervalo muy breve durante el cual no reflejan los valores de las entradas antes del cambio. Pero este breve intervalo es una consecuencia del retardo finito de propagación a través de las puertas. En principio, estos retardos de propagación pueden considerarse muy pequeños y hasta despreciables.

En otros circuitos sin realimentación las salidas dependen no sólo de las entradas actuales, sino también con alguna extensión de su historia pasada (mecanismo de memoria). Esto es, las salidas actuales dependen de la secuencia que han tenido de valores lógicos en las entradas. **Estos circuitos lógicos se denominan secuenciales.**

1.9.1 Estados.

Cada etapa que atraviesa un circuito secuencial se denomina *estado*. En cada estado el circuito almacena un recuerdo de su historia pasada, para saber qué hacer a continuación. Un estado se distingue de otro por sus recuerdos almacenados. Parecerá que, a medida que progresa el tiempo, es necesario añadir nuevos

elementos a la memoria y por consiguiente desarrollar una secuencia ilimitada de estados nuevos y diferentes. Sin embargo, parece que ni toda la historia pasada es relevante, ni todos los estados por los que progresa el sistema son diferentes entre sí y que el número total de estados es bastante limitado. Un ejemplo nos servirá para mostrar esta cuestión.

Supongamos que usted tiene la tarea de mirar una fila de cinco bombillos de luz numerados del 1 al 5 que normalmente están apagados. Un bombillo cualquiera, pero solamente uno cada vez se enciende brevemente de vez en cuando. Se le ha indicado que si se prende el bombillo 1, después el 2, y así sucesivamente hasta el número 5, debe esperar un instante y entonces hacer sonar una alarma y si el orden de destellos no sigue el de la numeración en esa secuencia, éste debe ignorarse hasta que se encienda la luz 1. Así pues, cuando finalice la secuencia 1,2,3,4,5, debe sonar la alarma. Si aparece la secuencia 1,2,4,..., después de observar el destello de la luz 4, se puede ignorar lo que ocurre a continuación ya que no se ha seguido el orden requerido y se puede seguir ignorando los destellos hasta que la luz 1 se encienda de nuevo, ya que puede ser el comienzo de la secuencia de destellos en el orden requerido.

Podemos juzgar fácilmente el número de estados diferentes y distintos que tendrá este sistema, imaginando lo que es relevante: que las luces no se han encendido, que se ha encendido la luz 1 y después la 2, y así sucesivamente, hasta que aparezca la secuencia 1,2,3,4,5, y entonces se haga sonar la alarma. Hay un total de 6 mensajes diferentes que es necesario tener en cuenta por lo tanto, hay seis recuerdos y estados diferentes.

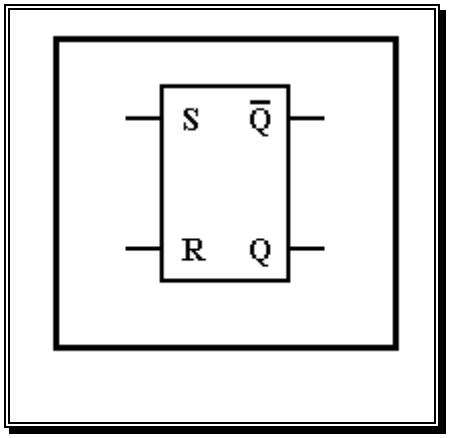
El número de diferentes secuencias de encendido que pueda observar realmente no tiene límite, pero no todas las diferentes secuencias son diferentes de una forma relevante. Supongamos por ejemplo que observa la secuencia 3,2,4,5,1,2,5,3,1,4,2,1,2. Todo lo que necesita recordar de esta secuencia es que los dos últimos encendidos fueron 1 y 2. Así pues, esta secuencia coloca al sistema en el mismo estado que la secuencia 1,2. Supongamos que se visualiza la secuencia 1,2,3,4,2,5,3,2,1,3, cuando finalice el sistema se encontrará en el mismo estado que si no se hubiese encendido ninguna luz.

Podemos tratar de implementar un sistema secuencial con personas humanas como en el ejemplo anterior, en cuyo caso dependeremos de la memoria humana, pero *si tratamos de implementar un sistema con componentes lógicos, utilizaremos flip-flops (biestables) como memoria*. Si el sistema tiene por ejemplo, seis estados de recuerdo (como el caso presentado), consecuentemente necesitaremos seis estados de flip-flops. Para disponer de seis estados necesitaremos tres flip-flops. En realidad dispondremos de ocho estados que irían de $Q_1Q_2Q_3 = 000$ a $Q_1Q_2Q_3 = 111$, y como solamente necesitamos seis, dos estados no se utilizarían.

1.9.2 Flip-flops tipo SR.

Los multivibradores bistables o flip-flops (FF) son los elementos básicos de memoria en los sistemas secuenciales. De ellos pueden considerarse varios tipos. El FF SR posee dos entradas y dos salidas. Las entradas corresponden a R (“reset”) y S ("set") mientras que las salidas, complementarias entre sí, corresponden a Q y $\overline{Q}$.

Fig. 24. Símbolo lógico del flip-flop SR.



El FF se dice que se encuentra en estado alto si Q = 1. Si Q = 0 el FF se encuentra en estado bajo.

El funcionamiento de este elemento de memoria es como sigue:

1.- A fin de garantizar una operación adecuada del FF, no se permite aplicar simultáneamente un nivel lógico alto en las entradas S y R. (S = 1 y R = 1 respectivamente).

2.- Si se aplica un nivel lógico alto a la entrada S, El FF se coloca en estado alto (Q=1).

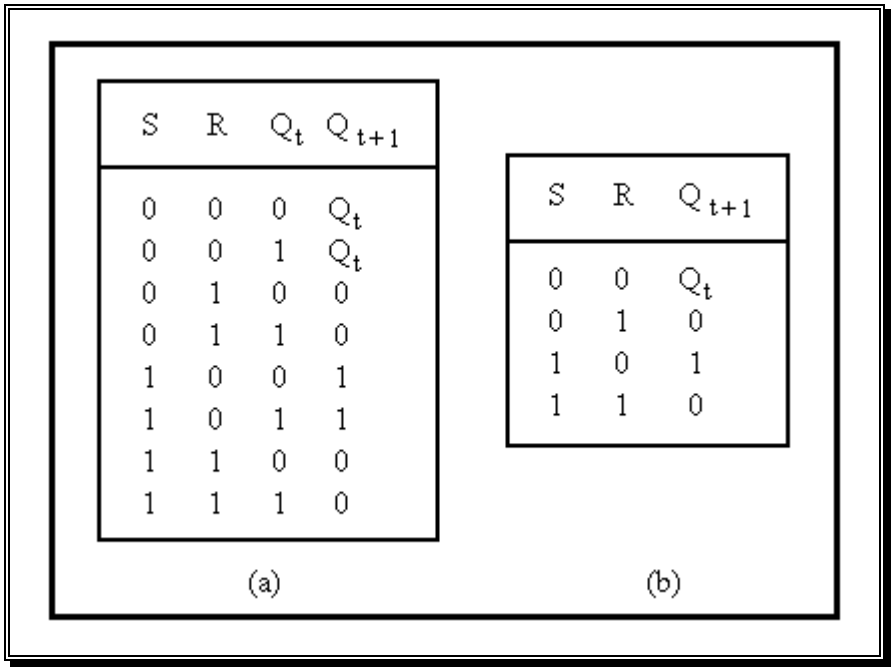
3.- Si se aplica un nivel lógico alto a la entrada R, el FF se coloca en estado bajo (Q=0).

4.- Si el FF pasa al estado alto al hacer S = 1, y luego se cambia S a 0, el FF mantiene su estado anterior correspondiente Q = 1.

5.- Si el FF pasa al estado bajo al hacer R = 1, y luego se cambia R a 0, el FF mantiene su estado anterior correspondiente Q = 0.

La figura 25 (a) muestra *la tabla de operación del FF tipo SR*. La figura 25 (b) es una versión simplificada del modo de operación de este FF. En esta tabla Q_t denota al estado presente de la salida Q del flip-flop y Q_{t+1} es el estado que resulta a la salida luego de un cambio en las entradas S y R.

Fig. 25. Tabla de verdad para un FF SR. (a) Tabla de operación.
(b) Tabla de operación simplificada



S	R	Q_t	Q_{t+1}
0	0	0	Q_t
0	0	1	Q_t
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(a)

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	0

(b)

De la tabla 25 (b) puede observarse que el FF conserva su estado anterior ($Q_{t+1} = Q_t$) cuando S = R = 0. Esta situación corresponde al estado de memoria del FF. Cuando S = 1, la salida es 1 y cuando R = 1, la salida es cero.

1.9.3 Lógica Secuencial Sincronizada.

En la sección anterior analizamos el FF SR. Para este tipo de elemento la respuesta se obtenía casi inmediatamente cuando las entradas S o R eran

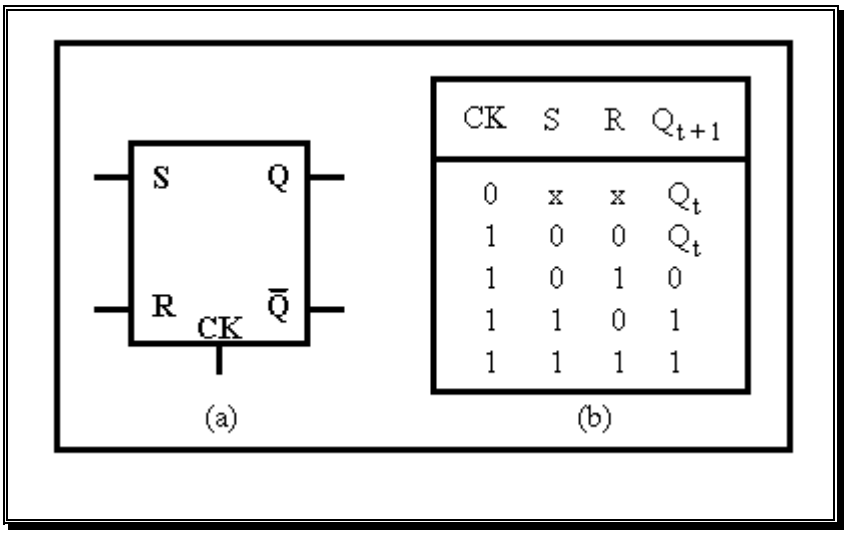
modificadas. A este tipo de lógica secuencial se le denomina **lógica asincrónica** en contraste con la **lógica secuencial sincronizada.** En este último tipo de lógica, base fundamental de los circuitos digitales modernos, los cambios se producen bajo el comando de un pulso de reloj (CK). Sólo cuando este pulso está presente puede producirse algún cambio a la salida. La lógica sincronizada ofrece la ventaja de que las transiciones de estado se producen de una manera ordenada en todas las partes del sistema digital y casi al mismo tiempo.

Básicamente existen dos maneras de iniciar cambios en un sistema sincronizado: **por nivel** y **por flanco**. En el sistema sincronizado por el **nivel del pulso de comando**, los cambios se producen cuando el pulso de reloj (CK) está bien en nivel alto o bien en nivel bajo, dependiendo del tipo de sistema. En el sistema sincronizado por el **flanco de pulso de comando,** los cambios se producen cuando el pulso de reloj está cambiando de nivel bajo a nivel alto o viceversa. En este caso solamente es importante el comienzo o el final del pulso.

1.9.4 FF Tipo SR sincronizado.

La tabla de operación y el símbolo de un FF SR sincronizado por nivel, se muestra a continuación:

Fig. 26. FF SR. (a) Símbolo lógico. (b) Tabla de operación.



Veamos como trabaja este FF:

1.- Si CK = 0, ésto inhibe cualquier cambio en las salidas del flip-flop. La primera fila de la tabla de operación de la figura corresponde a esta situación.

2.- Supongamos ahora que la entrada del reloj (CK) es igual a 1 y que las salidas sean Q = 1 y $\overline{Q}$ = 0. Asumamos además que S = R = 0. Es decir, el flip-flop mantiene su salida anterior (Segunda fila de la tabla).

3.- Si asumimos que CK = 1, S = R = 0 y que Q = 0 y $\overline{Q}$ = 1, también se conserva el estado anterior (Segunda fila de la tabla).

4.- Consideremos ahora que CK = 1 y que las entradas al flip-flop son S = 0 y R = 1. El próximo estado de Q es 0 y el de $\overline{Q}$ es 1. El flip-flop, por tanto, se pone a un nivel bajo (fila 3 de la figura 26 (b)).

5.- Si tenemos las mismas condiciones de entrada del caso anterior (CK = 1, S =0, R = 1) y las salidas son ahora Q = 0 y $\overline{Q}$ = 1, estas últimas mantienen sus niveles bajo y alto respectivamente. (fila 3 de la figura 26 (b)).

6.- Si CK = 1 y hacemos S = 1 y R = 0, lo que origina un cambio en Q y $\overline{Q}$ a 1 y 0 respectivamente (fila 4 de la figura 26 (b)).

7.- Finalmente, cuando S = R = 1 con el reloj en el nivel alto, ambas salidas pasan a nivel alto. Este estado, Q = $\overline{Q}$ = 1, aunque posible, no es deseable en el funcionamiento del FF.

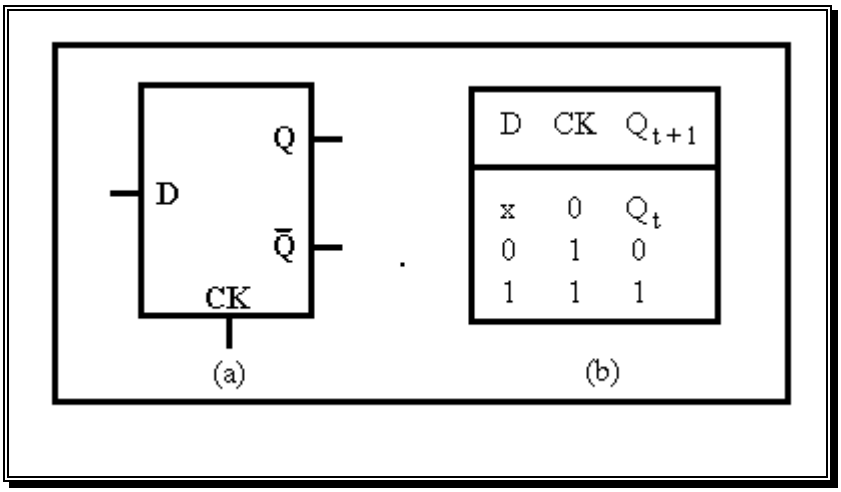
1.9.5 Flip-Flop Tipo D (Data).

El FF tipo D sincronizado por el nivel de pulso de reloj se obtiene modificando ligeramente el FF SR sincronizado. La tabla de operación y el símbolo lógico de un FF tipo D se muestra en la figura 27. En este caso, las entradas R y S se conectan entre sí para formar la entrada única D ("Data"). El FF tipo D trabaja de la manera siguiente:

1.- Cuando CK = 0, el FF mantiene la salida anterior independientemente de los cambios que pudiesen ocurrir en D.

2.- Cuando CK = 1, entonces Q = D. El FF lo que hace es transmitir la información de entrada cuando la señal de reloj está presente.

Fig. 27. FF tipo D. (a) Símbolo lógico. (b) Tabla de operación.

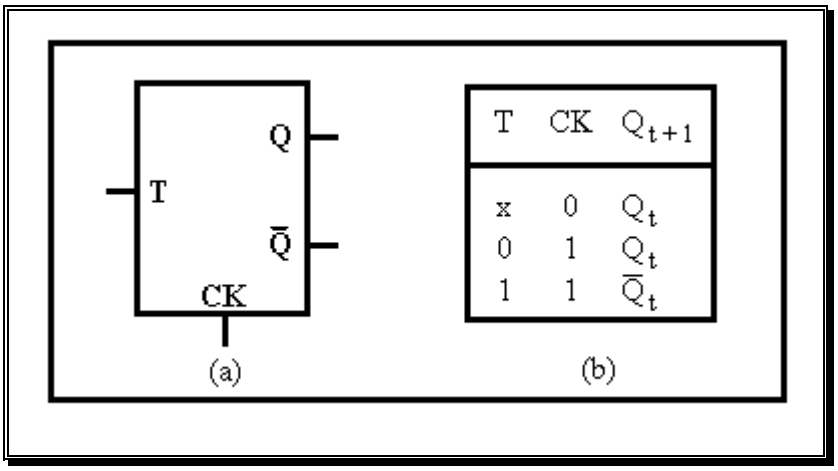


Podemos concluir que el FF tipo D es un elemento de memoria que almacena un bit de información y que cambia de estado, haciendo Q = D, bajo el control del reloj.

1.9.6 Flip-Flop Tipo T ("Toggle").

El símbolo lógico y la tabla de operación para el FF tipo T se muestra a continuación:

Fig. 28. FF tipo T. (a) Símbolo lógico. (b) Tabla de operación.



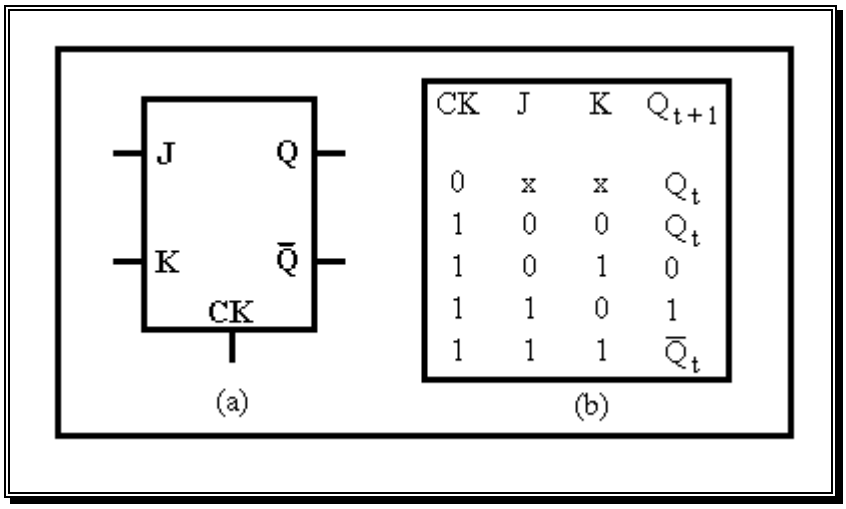
La tabla de operación muestra que si T está en nivel alto, la salida cambia de estado ($Q_{t+1} = Q_t$) cada vez que está presente un pulso de reloj (CK = 1). Si T = 0 la salida permanece igual al estado anterior, aún cuando CK cambie. Si CK = 0, $Q_{t+1} = Q_t$ independientemente del valor de T.

Nótese que el número de cambios que se produce en Q corresponde a la mitad de los cambios que se producen en el reloj CK. Se deduce entonces que el período de los pulsos de Q cuando T = 1 se duplica con relación al período de los pulsos del reloj. Dicho en otras palabras, **la frecuencia del reloj CK es dividida por 2 cuando T = 1.**

1.9.7 Flip-Flop JK.

El flip-flop JK es un multivibrador biestable cuyo símbolo lógico y tabla de operación son:

Fig. 29. FF tipo JK. (a) Símbolo lógico. (b) Tabla de operación.



Este tipo de FF tiene dos entradas, J y K y dos salidas Q y $\overline{Q}$. Cualquier cambio a la salida se produce sólo si está presente un pulso de reloj. El funcionamiento puede resumirse de la siguiente manera:

1.- Si ambas entradas son 0, el FF permanece en el estado anterior.

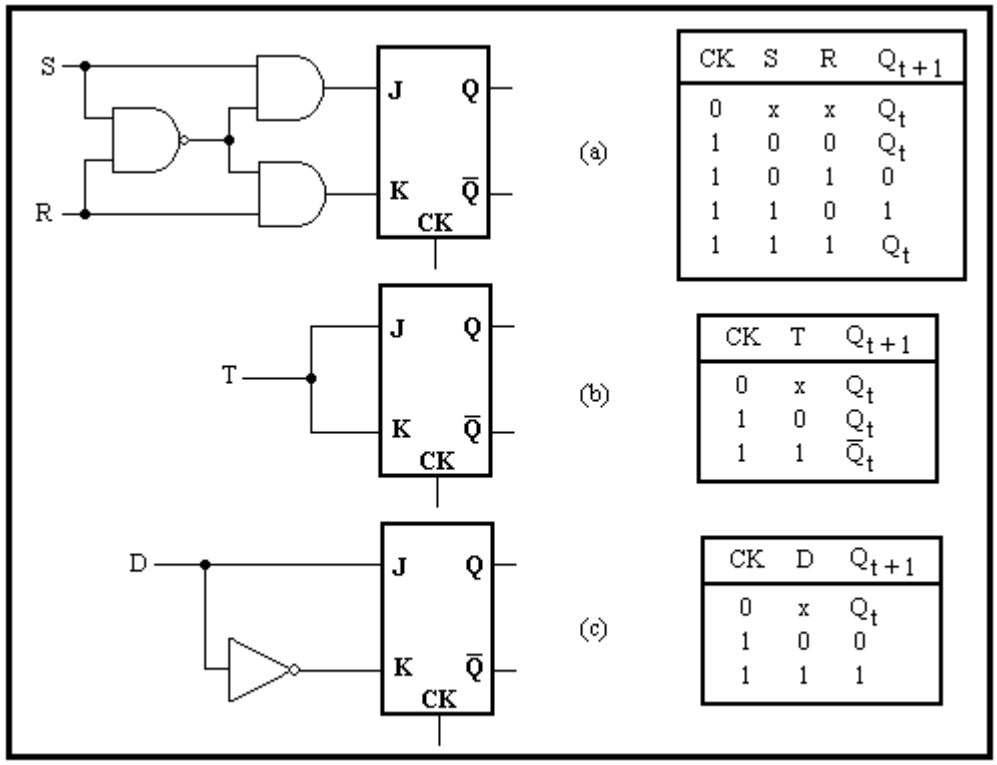
2.- Si J = 0 y K = 1, el FF cambia a 0.

3.- Si J = 1 y K = 0, el FF pasa a 1.

4.- Si J = K = 1, la salida es el complemento del estado anterior.

A partir de FF JK pueden obtenerse flip-flops tipo SR, D o T. Esto se muestra en la figura 30. En el caso de flip-flops tipo SR, la combinación S = R = 1 no produce cambios a la salida del mismo. En todos estos FF es necesario la presencia de un pulso de reloj para que se produzcan cambios a la salida.

Fig. 30. Obtención de otros tipos de FF a partir de un FF JK.
(a) Tipo SR. (b) Tipo T. (c) Tipo D



Fuente : Figuera. 1990

En los aspectos que trataremos posteriormente, veremos algunos ejemplos de circuitos secuenciales tales como una celda de bit, un ejemplo de memoria lineal simple, etc.

2. ESTRUCTURA BÁSICA DEL COMPUTADOR.

El objetivo de este capítulo es presentar algunos conceptos básicos y la terminología asociada con ellos. Se proporcionará un panorama amplio de las características fundamentales de las computadoras, para dejar su análisis más detallado y preciso para los capítulos subsecuentes.

2.1 Unidades Funcionales.

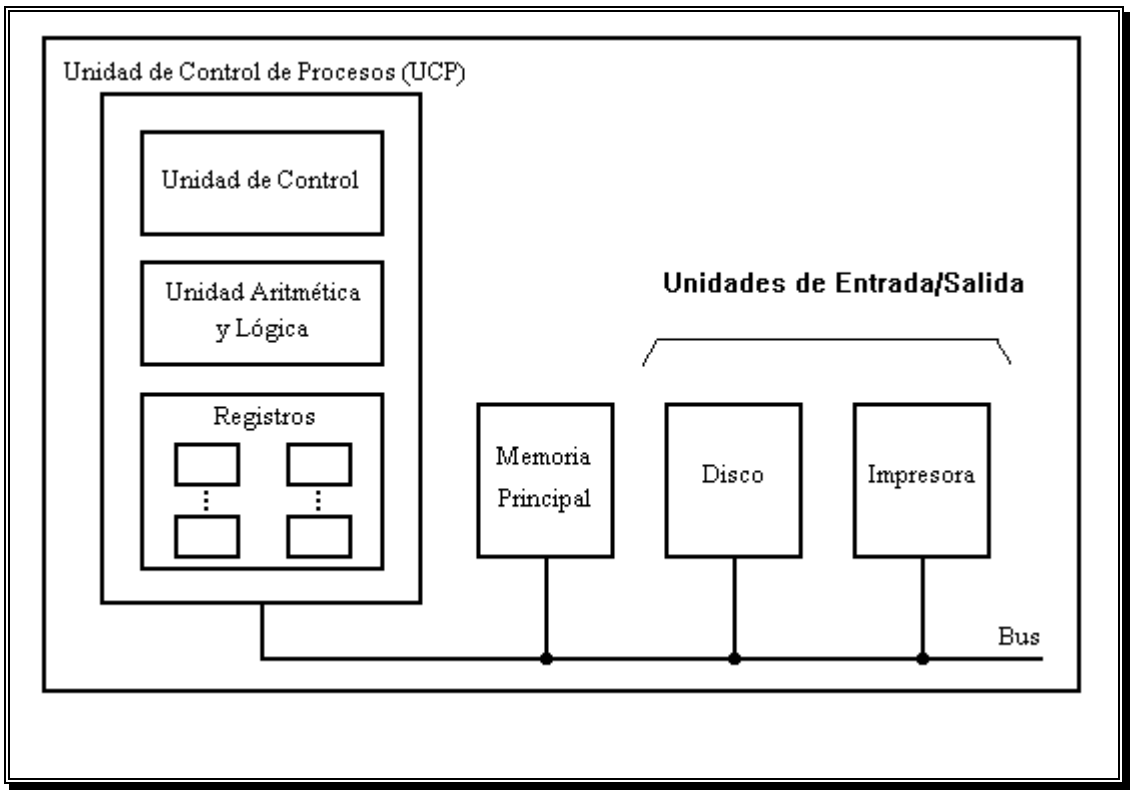
La palabra computador abarca una gran variedad de máquinas, que difieren mucho entre sí en cuanto a tamaño, velocidad y costo. En la actualidad, se utilizan palabras más específicas para referirse a cada clase de computadoras.

Sin embargo, los conceptos básicos son en esencia los mismos para todas las clases de computadores y se basan en unas cuantas ideas bien precisas que se plantean durante el resto del presente trabajo.

En su forma más simple, una computadora consiste en cinco partes principales e independientes desde el punto de vista funcional que le dan a ésta máquina las capacidades necesarias para desarrollar aplicaciones en diversos ambientes y con diferentes grados de complejidad.

A continuación se presenta un modelo sencillo:

Fig. 31. Operación de un computador básico.



La *unidad de entrada* acepta información codificada que proviene de operadores humanos, de dispositivos electromagnéticos o de otras computadoras conectadas a ella a través de líneas digitales de comunicación. La información se almacena en la memoria para consulta posterior, o bien la maneja de inmediato la *unidad aritmética y lógica*, la cual realiza las operaciones deseadas. Los pasos de procesamiento están determinados por un programa almacenado en la memoria. Por último, los resultados se envían de nuevo al mundo exterior a través de la *unidad de salida*. Todas estas acciones son coordinadas por la *unidad de control*. Es común referirse a los circuitos aritméticos y lógicos en unión con los circuitos de control como *Unidad Central de Procesamiento* (UCP o CPU: Central Processing Unit) o simplemente *Procesador*. El término "central" se utiliza para indicar que en una computadora la mayor parte de las

funciones de control se centran en una sola unidad. Los sistemas actuales contienen muchos procesadores, cada uno con cierta función específica. Por lo general, con el término *Unidad de Entrada-Salida* (E/S) se combinan los equipos de entrada y salida. Esto resulta apropiado porque hay equipo estándar que proporciona funciones tanto de entrada como de salida.

En este punto debe considerarse más de cerca la "información" que alimenta un computador. Resulta conveniente considerar tal información como de dos tipos: *instrucciones y datos*. Las instrucciones son comandos explícitos que:

- Gobiernan la transferencia de información dentro de un computador, así como entre éste y sus dispositivos de E/S.
- Especifican las operaciones aritméticas y lógicas que deben realizarse.

Un conjunto de instrucciones que realiza una tarea recibe el nombre de *programa*. El modo acostumbrado de operación consiste en almacenar un programa (o varios programas) en la memoria. Después, el procesador captura o recupera de la memoria las instrucciones correspondientes al programa y realiza con los datos las operaciones deseadas. Así, el *programa almacenado* controla totalmente el comportamiento mismo de una computadora, con excepción de la posibilidad de que el operador o los dispositivos conectados a la máquina efectúen una interrupción externa.

La información manejada por una computadora debe estar codificada en un formato adecuado. Generalmente el hardware (esto es, el equipo electrónico y

electromecánico) emplea circuitos digitales que tienen dos estados naturalmente estables los cuales son “encendido” y “apagado” y por lo cual se utiliza la codificación binaria. Esto significa que cada número carácter de texto o instrucción se codifica como una cadena de dígitos binarios (bits) en que cada una tiene cualquiera de los dos posibles valores. Por lo general, los números se representan en notación binaria posicional, según se analiza con detalle en el apéndice A. Ocasionalmente, se emplean formatos como el *Decimal Codificado en Binario* (BCD: *Binary-coded decimal*), en el que cada dígito decimal está codificado mediante cuatro bits.

Los caracteres alfanuméricos también se expresan en términos de códigos binarios. Se han desarrollado varios esquemas adecuados de codificación. Dos de los más comunes son el ASCII (*American Standard Code for Information Interchange*), en donde cada carácter se representa como un código de siete u ocho bits y el EBC-DIC (*Extended Binary-Coded Decimal Interchange Code*), en donde se emplean ocho bits para denotar un carácter.

Unidad de entrada:

Las computadoras aceptan información codificada por medio de unidades de entrada, las cuales consisten en dispositivos capaces de "leer" tales datos. La más simple de estas unidades es el teclado, el cual se encuentra conectado a la unidad de procesamiento. El teclado está cableado de manera que siempre que se oprima una tecla (o combinación de ellas), la letra o dígito correspondiente se traduzca automáticamente al código que le corresponda y pueda entonces enviarse de manera directa, ya sea a la memoria o al CPU.

Existen muchos otros tipos de dispositivos de entrada, tales como el mouse (ratón), scanner, etc.

Unidad de memoria:

La función de la unidad de memoria es almacenar programas y datos. También esta función puede realizarse con una variedad de equipos. Resulta útil distinguir entre dos clases de dispositivos de memoria que cubren el almacenamiento primario y el secundario.

El Almacenamiento Primario o Memoria Principal, es una memoria rápida en la que se almacenan programas y datos durante su ejecución. La memoria principal contiene un gran número de celdas de almacenamiento, cada una capaz de almacenar un bit de información. En estas celdas rara vez se lee o se escribe como celdas individuales. Más bien se procesa en grupos de tamaño fijo conocido como *palabras*. La memoria principal está organizada de forma que el contenido de una palabra que contenga *n* bits, pueda almacenarse o recuperarse en una operación básica.

Para proporcionar fácil acceso a cualquier palabra que se encuentre en la memoria principal, resulta útil asociar a cada localización de palabra un nombre diferente. Estos nombres son números que identifican localizaciones sucesivas, por lo que se denominan *Direcciones*. El acceso a una palabra dada se logra especificando su dirección y emitiendo un comando de control que inicie el proceso de almacenamiento o recuperación.

Al número de bits de cada palabra a menudo se le denomina *longitud de palabra*. Por lo común, las computadoras tienen 8, 16, 32 o más bits por palabra. La capacidad de la memoria principal es uno de los factores que determinan el tamaño de una computadora. Por lo general, dentro de una máquina los datos se manipulan por unidades de palabras, múltiplos de palabras o sub múltiplos de palabras. Es usual que un acceso a la memoria principal de como resultado la lectura o escritura de una palabra de datos en la memoria.

Como se mencionó antes, los programas y datos deben estar en la memoria principal durante su ejecución. Las instrucciones y datos pueden escribirse en la memoria o leerse de ella bajo el control de la unidad de procesamiento. Es esencial poder efectuar el acceso a cualquier localización de palabra dentro de la memoria principal lo más rápido posible. Las memorias en las que cualquier localización puede alcanzarse especificando se dirección reciben el nombre de *Memorias de Acceso Aleatorio* (RAM: *Random-Access Memories)*. El tiempo necesario para efectuar el acceso a una palabra recibe el nombre de *Tiempo de acceso a la memoria*. Este es un tiempo fijo, por lo general de 100 a 500 nanosegundos (ns: 10^{-9} seg.) en la mayor parte de las computadoras actuales.

Aún cuando en almacenamiento primario es esencial, tiende a ser costoso y en cierta manera limitado. Por ello, cuando tienen que almacenarse grandes cantidades de datos se utiliza *Almacenamiento Secundario* adicional, que es más barato sobre todo cuando no es necesario el acceso muy frecuente a los datos. Existe una amplia variedad de dispositivos adecuados que incluyen discos, cintas magnéticas, etc.

Unidad Aritmético-lógica:

Dentro de una computadora la ejecución de la mayoría de las operaciones tiene lugar dentro de la Unidad aritmético y lógica (UAL o ALU: *arithmetic and logic unit*). Un ejemplo representativo sería cuando dos números localizados en la memoria principal deben sumarse. Para ello se traen a la unidad aritmética en donde se realiza la adición. Después, el resultado puede almacenarse en la memoria.

De manera semejante, cualquier otra operación aritmética o lógica (por ejemplo multiplicación, división o comparación de números) se realiza trayendo los operandos necesarios a la ALU, en donde se realiza la operación necesaria. Es conveniente señalar que no todos los operandos de un cálculo que esté en proceso residen en la memoria principal, ya que es normal que los procesadores contengan un cierto número de elementos de almacenamiento de alta velocidad denominados *registros*, los cuales pueden utilizarse para el almacenamiento temporal de operandos que se usen con frecuencia. Cada uno de tales registros puede almacenar una palabra de datos. Los tiempos de acceso a los registros son, por lo general, de 5 a 10 veces más veloces que los tiempos de acceso a la memoria.

Es frecuente que tanto la unidad de control como la aritmética sean mucho más veloces en tiempo básico de ciclo que los otros dispositivos conectados a un sistema de cómputo. Por esto resulta posible diseñar sistemas de cómputos relativamente complejos que contengan varios dispositivos externos controlados por un solo procesador. Desde luego esto solamente es posible gracias a la gran diferencia de velocidades, lo que permite que el rápido procesador organice y controle la actividad de muchos dispositivos que sean más lentos que él.

Unidad de Salida:

La unidad de salida es la contraparte de la unidad de entrada. Su función consiste en devolver los resultados procesados por el CPU al mundo exterior. Hay dispositivos que proporcionan tanto una función de salida como una de entrada. Este doble papel de algunos dispositivos (disco duro, diskette) es la razón por la cual se combinen las unidades de entrada y salida bajo el nombre único de E/S.

Desde luego, existen dispositivos que se utilizan sólo para la salida, como el *monitor* (pantalla) o las *impresoras.*

Unidad de Control:

Las unidades antes descritas proporcionan las herramientas necesarias para almacenar y procesar información. Su operación debe coordinarse en alguna forma organizada, lo cual es la tarea de una unidad de control. Es efectivamente el centro nervioso que envía señales de control a las otras unidades.

Una impresora escribirá un renglón sólo si se le dan las instrucciones específicas para hacerlo. Lo común es que esto se efectúe por medio de la instrucción adecuada write (escribir) dada por el procesador. El procesamiento de esta orden implica enviar *señales de sincronización hacia y desde* la impresora, lo cual es la función de la unidad de control.

Puede decirse en general que las transferencias de E/S son controladas por instrucciones de software, las cuales identifican a los dispositivos involucrados y al

tipo de transferencias. Sin embargo, son los circuitos de control que realmente generan las señales de tiempo durante la ejecución. De manera semejante, la unidad de control también controla la transferencia de datos entre los procesadores y la memoria.

Desde el punto de vista conceptual resulta razonable considerar una unidad de control como un todo bien definido, físicamente separado, que de alguna forma interactúa con las otras partes de una máquina. En la práctica, éste rara vez es el caso. Buena parte de la circuitería de control está físicamente distribuida a través de toda la máquina. Un conjunto más bien grande de líneas de control (alambres) transmiten las señales que utilizan para medir el tiempo y sincronizan hechos en todas las unidades.

En resumen, la operación de un computador puede describirse de la siguiente forma:

- La computadora acepta información (programas y datos) a través de una unidad de entrada y la transfiere a la memoria.
- Bajo el control de un programa, la información almacenada en la memoria se recupera y se lleva a una unidad aritmética y lógica para que sea procesada.
- La información procesada sale de la computadora a través de una unidad de salida.
- La unidad de control dirige todas las actividades dentro de la máquina.

2.2 Operaciones básicas.

En la sección anterior se especificó que dentro de un computador la actividad se gobierna por medio de instrucciones. Para realizar una tarea dada, se almacena en la memoria principal un programa adecuado que consiste en un conjunto de instrucciones. Las instrucciones individuales se traen de la memoria la procesador y este ejecuta las instrucciones especificadas. Además de las instrucciones, es necesario utilizar como operandos algunos datos que también están almacenados en la memoria.

El procesador contiene circuitería aritmética y lógica como elementos principales del procesamiento. También contiene varios *Registros* que se utilizan para almacenamiento temporal de datos. Hay varios registros que resultan de particular interés. *El Registro de Instrucción* (IR: *instruction register*) contiene la instrucción que se esté ejecutando. Su salida se encuentra a disposición de los circuitos de control, los cuales generan las señales de sincronización que controlan los verdaderos circuitos de procesamiento necesarios para ejecutar la instrucción. *El Contador de Programa* (PC: *program counter*) es un registro que rastrea la ejecución de un programa. Contiene la dirección de memoria de la instrucción que en ese momento se esté ejecutando. Durante la ejecución de esta instrucción el contenido del PC se actualiza para que corresponda a la dirección de la siguiente instrucción que deba ejecutarse. Se acostumbra decir que el PC apunta hacia la instrucción que deba traerse de la memoria.

Por último, hay dos registros que facilitan la comunicación con la memoria principal. Estos son el *Registro de Dirección de Memoria* (MAR: *memory address register*) y el *Registro de Datos de Memoria* (MDR: *memory data register*). Como su nombre lo indica, el MAR contiene la dirección de localización hacia o desde la cual

deban transferirse los datos. El MDR contiene los datos que deben escribirse o leerse en ésa dirección.

Ahora se examinarán algunos pasos operativos comunes. Los programas residen en la memoria principal y por lo general llegan ahí a través de la unidad de entrada. La ejecución de un programa comienza haciendo que el PC apunte a la primera instrucción del programa. El contenido del PC se transfiere al MAR y a la memoria se envía una señal de control de lectura. Después de que transcurre cierto tiempo (que corresponde al *tiempo de acceso a la memoria*), la palabra de la dirección (en este caso la primera instrucción del programa) se lee y se saca de la memoria para cargarse en el MDR.

A continuación el contenido del MDR se transfiere al IR, momento en que la instrucción está lista para ser decodificada y ejecutada.

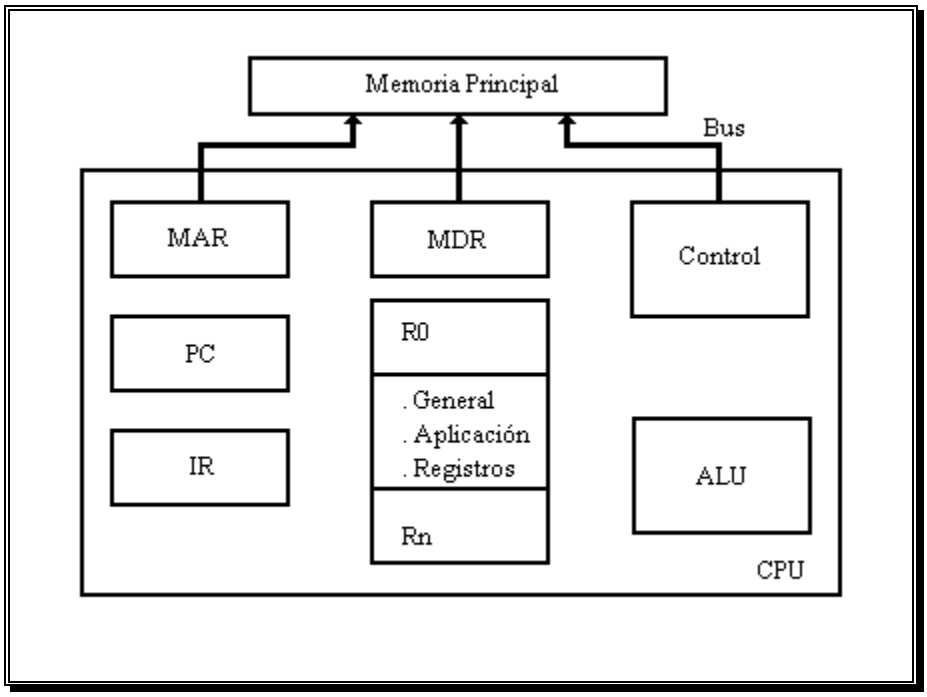
Si la instrucción incluye una operación que deba ser realizada por la ALU, entonces será necesario obtener los operandos necesarios. Si un operando reside en la memoria (también podría estar en un registro general del procesador), tendrá que traerse enviando su dirección al MAR e iniciando un *ciclo de lectura*. Cuando el operando ha sido leído de la memoria y llevado al MDR, puede transferirse del MDR a la ALU. Una vez que se trajo uno o más operandos de esta forma, la ALU puede realizar la operación necesaria. Si el resultado de esta operación debe almacenarse en la memoria, antes tendrá que enviarse al MDR. La dirección de la localización en donde el resultado debe almacenarse se envía la MAR y se inicia un *ciclo de escritura*. Mientras tanto, el contenido del PC se incrementa para apuntar hacia la siguiente

instrucción que deba ejecutarse. Así, en cuanto se culmine la ejecución de la instrucción que se esté procesando, podrá traerse una nueva instrucción.

Además de que se transfieran datos entre la memoria principal y el procesador, es necesario que se cuente con capacidad para aceptar datos provenientes de dispositivos de entrada y para enviarlos a dispositivos de salida. Por esto deben proporcionarse algunas instrucciones de máquinas que sean capaces de manejar transferencias de E/S.

Algunas veces es posible alterar la ejecución normal de un programa. A menudo sucede que algún dispositivo requiere de un servicio urgente. Para manejar tales situaciones con suficiente velocidad, el flujo normal del programa que el procesador está ejecutando debe interrumpirse. Para lograr esto, el dispositivo puede enviar una *señal de interrupción*. Una *Interrupción* es una solicitud de servicio que proviene de un dispositivo de E/S y que se envía al procesador. El CPU proporciona el servicio solicitado ejecutando la *rutina de interrupción-servicio* que resulte adecuada, mediante el uso del sistema operativo. Ya que tales distracciones pueden alterar el estado interno del procesador, es esencial que su situación se guarde en la memoria principal, antes de dar servicio a la interrupción. Normalmente esto implica almacenar tanto el contenido del PC, de los registros generales, como cierta información de control. Al concluir la rutina de interrupción-servicio, el estado del procesador se restaura, de manera que pueda continuar la ejecución del programa interrumpido.

Fig. 32. Conexiones entre el CPU y la memoria principal.



2.3 Buses. Estructuras de conexión.

Hasta aquí se han analizado las características funcionales de las partes individuales que constituyen una computadora. Para que formen un sistema operacional deberán conectarse entre sí de alguna forma organizada. Existen muchas formas de hacerlo. Se tomarán en cuenta tres estructuras de las más conocidas.

Para que logre una velocidad de operación razonable, una computadora debe estar organizada en forma paralela. Esto significa que todas las unidades pueden manejar una palabra completa de datos en un momento dado. También significa que las transferencias de datos entre las unidades deben hacerse en paralelo, lo cual implica que se requiere un número considerable de alambres (líneas) para establecer las conexiones necesarias. Un conjunto de alambres que tiene cierta identidad común,

recibe el nombre de *bus*. Además de los alambres que transportan los datos, es esencial disponer de algunas líneas para fines de direccionamiento y control.

En la figura 33 se muestra la forma más simple de computadora estructurada en dos buses. El procesador interactúa con la memoria a través de un bus de memoria. Las funciones de entrada y salida se manejan por medio de un bus de E/S que pasa a través del procesador en su camino a la memoria. En tales configuraciones, las transferencias de E/S por lo general están bajo control directo del procesador, el cual inicia transferencias y vigila su progreso hasta la conclusión.

En la figura 34 se proporciona una versión ligeramente diferente de estructura de dos buses. Las posiciones relativas del procesador y la memoria están invertidas. Aquí también hay un bus de memoria para la comunicación entre ellos. Sin embargo, las transferencias de E/S se efectúan directamente hacia o desde la memoria. Ya que la memoria tiene poca circuitería capaz de controlar tales transferencias, es necesario establecer un mecanismo de control distinto. Una técnica estándar consiste en proporcionar *canales de E/S* como parte del equipo de E/S. Un canal de E/S tiene la capacidad necesaria para controlar las transferencias. En realidad, se trata de un procesador de aplicación especial, a menudo llamado *procesador periférico*.

Fig. 33. Estructura de dos Buses.

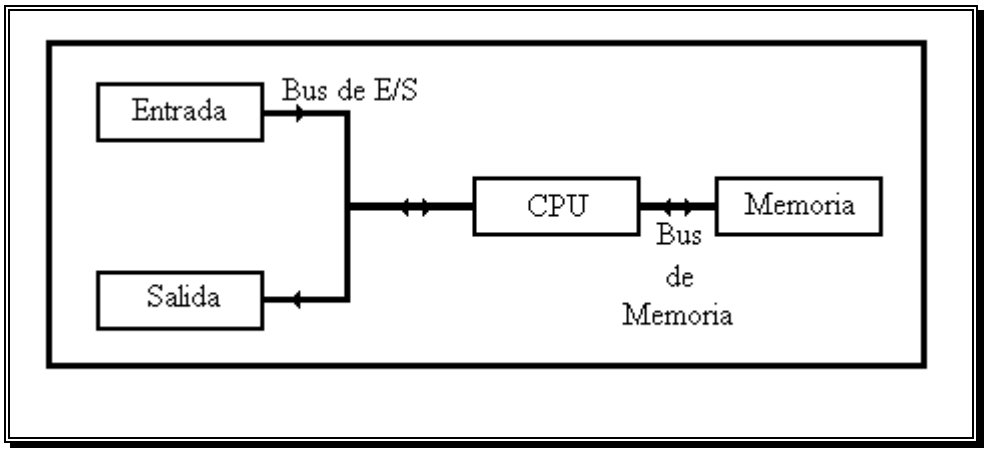
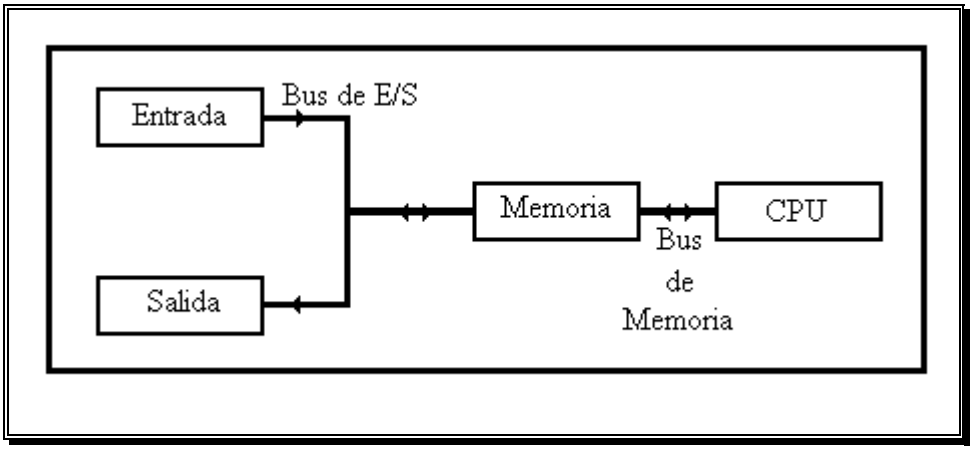


Fig. 34. Estructura alternativa de dos buses.



Un procedimiento común consiste en hacer que el procesador inicie una transferencia pasando la información requerida al canal de E/S, que la toma a su cargo y controla la transferencia propiamente dicha.

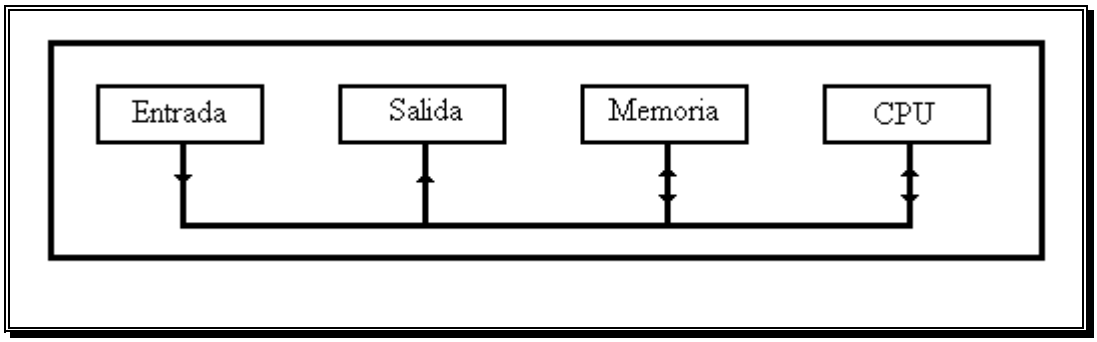
Ya se mencionó que un bus consiste en un conjunto de líneas distintas, que sirven a diferentes propósitos. *Existen 3 agrupamientos principales de líneas: de datos, de dirección y de control.* Las líneas de datos se utilizan para la transmisión de datos. Por lo tanto, el número de tales líneas corresponde al número de bits de la

palabra que manipula el procesador. Para efectuar el acceso a los datos contenidos en la memoria, las líneas de dirección indican su localización. Las líneas de control se utilizan para indicar la dirección de la transferencia de datos y coordinar la temporización de los hechos que suceden durante una transferencia.

Muchas máquinas tienen varios buses distintos, de manera que en realidad se las podría tratar como máquinas de *bus múltiple*. Sin embargo, su operación se describe en forma adecuada mediante los ejemplos de 2 buses, ya que la razón principal de que se incluyan buses adicionales es aumentar la velocidad de operación a través de aún más paralelismo.

Una estructura muy diferente, tiene un *bus único*, se muestra en la figura 35. Todas las unidades están conectadas a este bus que proporciona el único medio de interacción. Ya que el bus puede utilizarse únicamente para una transferencia cada vez, puede deducirse que en un momento dado sólo dos unidades pueden estar usando activamente el bus. Las líneas de control del bus se emplean para mediar entre las unidades que soliciten el uso del bus. La principal virtud del bus único es su bajo costo y flexibilidad para conectar dispositivos periféricos. A cambio de esto, su velocidad de operación es más baja.

Fig. 35. Estructura de bus único.



Las diferencias entre las estructuras de buses tienen un marcado efecto en el rendimiento de las computadoras. Sin embargo, desde el punto de vista conceptual las diferencias no son cruciales en ninguna descripción funcional. Además, los principios fundamentales de operación de las computadoras son esencialmente independientes de su estructura específica de bus.

En general, la transferencia de información por un bus no puede realizarse a una velocidad directamente comparable con la velocidad de operación de todos los dispositivos conectados al bus. Algunos dispositivos son relativamente lentos como por ejemplo, los terminales y las impresoras. Otros tales como los discos y las cintas magnéticas son considerablemente más veloces. La memoria principal y los procesadores operan a velocidades considerables, por lo que son las partes más veloces de una computadora. Ya que todos estos dispositivos deben comunicarse entre sí a través de un bus, es necesario proporcionar un mecanismo de transferencia que sea eficiente, que no esté restringido por los dispositivos lentos y que pueda utilizarse para suavizar las diferencias de temporización que haya entre los procesadores, las memorias y los dispositivos externos.

Una práctica muy común es incluir *registros buffer* en los dispositivos para que contengan la información durante las transferencias. Esta técnica se puede ejemplificar con la transferencia de un carácter codificado desde un procesador, hasta una impresora de caracteres en donde debe ser escrito. El procesador efectúa la transferencia enviando el carácter por el bus al buffer de salida de la impresora. Ya que el buffer es un registro electrónico, esta transferencia requiere relativamente de poco tiempo. Una vez que el buffer está cargado, la impresora puede empezar a imprimir sin la posterior intervención del procesador. En este momento, el bus y el procesador ya no son necesarios y quedan disponibles para ocuparse de otra actividad. La impresora efectúa la escritura del carácter en su buffer y no queda disponible para transferencias posteriores hasta que no se complete este proceso. Para resumir, los registros buffer suavizan las diferencias de temporización que haya entre los diferentes procesadores, memorias y dispositivos de E/S que deben comunicarse entre sí en un sistema completo de computadora. Tales registros impiden que un procesador de alta velocidad quede bloqueado por un dispositivo de E/S lento durante una secuencia de transferencia de datos. Esto no permite al procesador conmutar rápidamente de un dispositivo a otro, y que su actividad de procesamiento se entrelace con transferencias de datos desde y hacia diferentes dispositivos.

En el análisis precedente se consideraron varios aspectos generales de la estructura y operación de las computadoras. Los capítulos siguientes proporcionarán definiciones completas de los términos empleados.

3. LA MEMORIA.

3.1 Concepto.

Llamaremos memoria a todo dispositivo electrónico encargado del almacenamiento de la información en el computador. Usualmente emplean el almacenamiento binario, es decir, la información elemental que registra es el bit.

3.2 Función.

La memoria viene siendo el lugar principal de trabajo de una computadora, debido a que todo el procesamiento tiene lugar en ella, esto es, se usa para almacenar los programas que debe ejecutar el procesador y también los datos que debe manipular. Las instrucciones que provienen de los dispositivos periféricos se copian en la memoria y el procesador se encarga de extraerlas para ejecutarlas. Al seguir las direcciones de las instrucciones en la memoria, la computadora recibe indicaciones para dar entrada en la memoria a los datos que provienen de los dispositivos periféricos.

Una vez que la información ha sido escrita en la memoria, el computador puede copiarla, compararla o modificarla, es decir, procesarla.

3.3 Características.

Estas son las que definen y diferencian los distintos tipos de memoria. Sus características se pueden agrupar según su:

a) **Modo de retener la información:** Esto es de acuerdo a la permanencia de la información grabada en las memorias. Estas pueden ser:

- *Memorias permanentes*: Son llamadas también memorias no volátiles o duraderas. Son memorias que contienen información y no pueden borrarse, por ejemplo, los discos y la memoria ROM (incluyendo el caso de los CD ROM).
- *Memorias No permanentes*: Son llamadas también memorias volátiles o temporales. Son memorias a las que se les borra la información cuando se le corta el suministro de energía.

b) **Modo de lectura:**

- *Memorias de lectura destructiva*: Son aquellas que cuando se lee la información tiende a desaparecer. Requieren de una regeneración del contenido.
- *Memorias de lectura no destructiva*: La información aún persiste a pesar de haber ocurrido una operación de lectura sobre la misma.

c) **Modo de acceso:** Es la manera cómo se puede accesar la información, la cual puede ser por palabra o por bloque.

- *Por palabra*: Es de acceso rápido y el tiempo de acceso es fijo independientemente de la dirección de la información. Es utilizada en la memoria principal.
- *Por bloque*: Es de acceso lento y el tiempo de acceso varía de acuerdo a la ubicación de la información. Es utilizada en memoria secundaria y es ideal para almacenar grandes cantidades de información.

d) **Velocidad:** Se refiere al tiempo en que tarda en realizar una operación de lectura o de escritura.

e) **Capacidad o tamaño:** Es la cantidad de información que puede almacenar en memoria, la cual se puede expresar en unidades de bits, bytes o palabras.

3.4 Jerarquización de las Memorias.

Pueden distinguirse tres niveles básicos de memorias en cuanto a su utilidad y localización en la arquitectura de la máquina.

a) *Memoria Caché*: Es una memoria auxiliar de poca capacidad y de alta velocidad, se añade a la memoria principal para acelerar su funcionamiento, ya que ella trae y mantiene información que existe en la memoria principal para que el procesador la tenga disponible. Su tiempo de acceso es mínimo (de 20 a 100 ns), por lo tanto reduce los tiempos de espera.

b) *Memoria principal*: Es de acceso aleatorio, capacidad muy alta (ella almacena los programas, sistema operativo, etc.) y su tiempo de acceso es de 20 a 21000 nanosegundos.

c) *Memoria Secundaria*: Posee una gran capacidad de almacenamiento con una velocidad comprendida entre 10 y 100 milisegundos, por lo tanto es más lenta que la memoria principal. Su acceso suele ser secuencial o aleatorio.

4. MEMORIA PRINCIPAL.

4.1 Concepto.

Es un dispositivo electrónico que permite almacenar información a través de secciones de almacenamiento, las cuales no están fijadas por límites físicos, pues pueden variar de una aplicación a otra. Esto significa que un espacio físico específico puede ser utilizado para guardar datos de entrada en una aplicación, resultados de salida en otra e instrucciones de un programa.

4.2. Descripción.

La memoria está constituida por un cierto número de celdas (células o posiciones), cada una de las cuales es capaz de almacenar una porción de información que puede ser de dos tipos: las instrucciones de un programa y los datos con los que opera esas instrucciones. Esta memoria sólo puede realizar dos operaciones básicas: lectura y escritura.

Cabe mencionar que la unidad más pequeña de almacenamiento dentro de la memoria es el bit, representado por un dígito binario (0 ó 1); conforman en un conjunto de ocho bits lo que se denomina byte, lo que constituye un carácter de información.

Obsérvese la implementación de un bit (circuito de tipo secuencial) usando un flip-flop de tipo SR, en donde sus componentes:

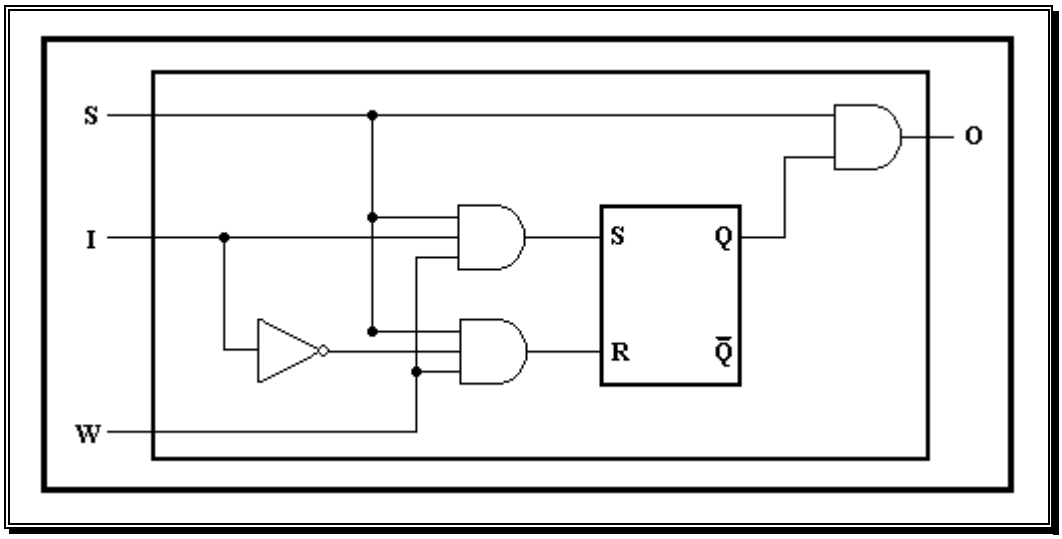
S (Select): Toma el valor lógico 1, si la celda pertenece a una palabra de memoria seleccionada.

I (Input): Valor del dato de entrada, si es el caso de una operación de escritura en la memoria.

W (Write): Señal que indica el tipo de operación. Toma el valor lógico 0 si la operación es de lectura y toma un valor lógico 1 cuando la operación es de escritura.

O (Output): Muestra el valor almacenado en la celda o bit.

Fig. 36. Celda de memoria o bit.

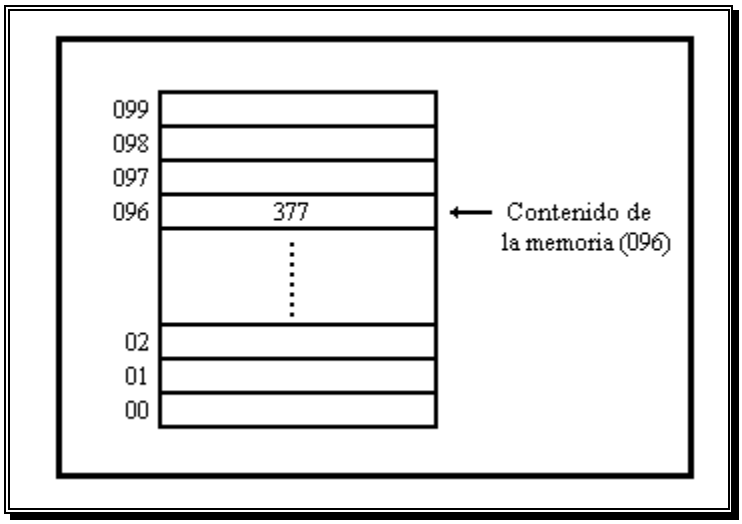


Según se señalo anteriormente, existen dos conceptos básicos asociados a cada byte o posición de memoria: su dirección y su contenido.

La dirección permanece constante y es única, mientras que el contenido puede cambiar mientras se ejecuta un programa.

Obsérvese una memoria de computadora que consta de 100 posiciones en memoria con direcciones de 0 a 99 y muestra además el contenido de una dirección específica:

Fig. 37. Memoria principal de un computador.



El número máximo posible de celdas de memoria se conoce como tamaño de memoria o espacio de direcciones; éste siempre será una potencia de 2 debido a que la representación interna de la dirección de memoria es binaria. Si se usan n bits para representar la dirección binaria sin signo, entonces se tendrá un máximo 2^n celdas de memoria y sus direcciones serán los valores 0,1,2,...,2^n - 1. Así, la memoria de una computadora puede tener desde unos centenares de bytes hasta millones de ellos.

El tamaño máximo de la memoria principal que puede utilizarse en cualquier computadora lo determina su esquema de direccionamiento.

4.2.1 Palabras y Contenido.

Es un determinado número de bits contenidos en una celda o posición de memoria. Es el medio del cual se dispone para almacenar la información en la

memoria. Es el tamaño mínimo para expresar dicha información en la memoria principal; por esta razón, es que podemos decir que la *palabra* es la unidad de información más pequeña direccionable. Una palabra de la memoria consta de n bytes que pueden direccionarse en forma individual. La palabra es el tamaño privilegiado principal de un computador, mientras que el byte y otros tamaños menores deben dividirla en forma exacta; ésta puede representar un operando, una instrucción, un grupo de caracteres alfanuméricos o cualquier información codificada en lenguaje binario a la que se le denomina *contenido.*

4.2.2 Longitud de palabra.

Es el número de bits que contiene cada palabra de memoria también se le denomina tamaño de palabra y por lo general comprende 8, 16 o 32 bits. Está estrictamente sujeta a la arquitectura del computador y a la forma en que este fue diseñado.

La longitud de palabra es de gran importancia ya que de ella depende la rapidez con que opera un computador. Por ejemplo, un computador de ocho bits, lo cual equivale a un byte, podrá mover o manipular los datos byte a byte, mientras que un computador de 32 bits, podrá manipular los datos en grupos de cuatro bytes.

Mientras más grande sea el tamaño de la palabra, el computador será capaz de direccionar mayor memoria principal y podrá aceptar instrucciones más grandes.

4.2.3. Unidades de Medida.

Son las unidades mediante las cuales se determina el tamaño de la información almacenada en la memoria. Estas unidades son las siguientes:

- Un conjunto de 4 bits = Nibble.
- Un conjunto de 8 bits = Byte.
- Un conjunto de 1024 byte = Kilobyte.
- Un conjunto de 1024 kilobyte = Megabyte.
- Un conjunto de 1024 megabyte = Gigabyte.
- Un conjunto de 1024 gigabyte = Terabyte.

4.3 Ciclo de Máquina.

La unidad central de procesamiento (CPU) de una computadora viene siendo el "cerebro" de la misma, ya que dicha unidad se encarga de controlar y supervisar las operaciones del sistema en su totalidad y de realizar las funciones aritméticas y lógicas.

La CPU se compone de dos partes: La unidad de control (UC) y la unidad aritmético-lógica (UAL). La UC tiene como función obtener una instrucción de la memoria principal, examinarla y codificarla. El tiempo transcurrido en este paso se denomina *tiempo de instrucción*. Una vez que la UC determina el tipo de instrucción, le entrega el control a la UAL para que ejecute la operación correspondiente lo cual genera

el *tiempo de ejecución*. La culminación de estos dos pasos es lo que se denomina Ciclo de Máquina.

4.3.1 Ciclo de Memoria.

Es el tiempo transcurrido desde que se solicita un dato a la memoria hasta que ésta, se encuentre en disposición de efectuar una nueva operación de lectura y escritura.

Para ejecutar una instrucción, es necesario que los circuitos de control de la CPU provoquen la transferencia de la instrucción desde la memoria principal hasta la CPU. También es necesario mover operandos y resultados entre la memoria principal y la CPU. Es evidente entonces que existe la necesidad de las dos operaciones básicas en que participa la memoria principal: Traer información (lectura) y almacenar información (escritura).

4.3.1.1 Operación de Lectura Desde Memoria.

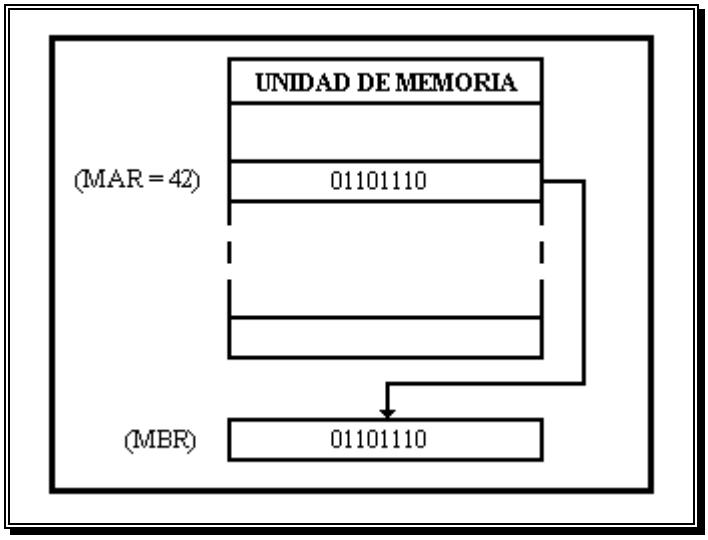
La secuencia de operaciones necesarias para comunicarse con la unidad de memoria con el propósito de transferir una palabra es:

1. - Transferir los bits de dirección de la palabra selecciona-da al registro de direcciones de memoria MAR.
2. - Activar la entrada de control de lectura.

El resultado de la operación de lectura se ilustra en la figura 38.

La información almacenada en el registro de memoria (42) se transfiere al registro de datos de memoria (MBR).

Fig. 38. Operación de lectura de memoria.



4.3.1.2 Operación de Escritura en Memoria.

La operación de escritura en memoria tiene similitud a la de lectura; cuando la unidad de memoria recibe una señal de control de escritura, el control interno interpreta el contenido del registro separador como la configuración de bits de la palabra que se va a almacenar en un registro de memoria. Existe una secuencia de operaciones o pasos a seguir para almacenar una palabra en memoria. Dichos pasos son:

1.- Se transfieren los bits de dirección de la palabra al MAR.

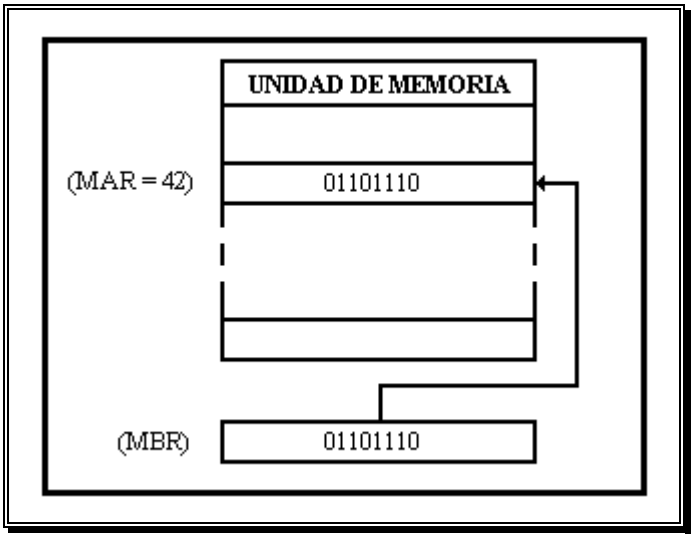
2.- Se transfieren los bits de datos de la palabra al MBR.

3.- Se activa la entrada de control de escritura.

4.- Se graba en la dirección indicada por MAR el contenido de MBR.

Para comunicar los registros (MBR, MAR y el registro de memoria) y realizar las operaciones de lectura y escritura hacemos uso de los buses.

Fig. 39. Operación de escritura en memoria.



4.3.2 Tiempo de Acceso.

El parámetro más importante asociado a una operación de memoria es el tiempo que ésta requiere para efectuarse. El tiempo de acceso es el tiempo que transcurre desde la emisión de una requisición de lectura o de escritura hasta que ese valor se encuentre en el MBR y esté a disposición para ser procesado.

4.3.2.1 Tiempo de Acceso del Sistema.

Es el tiempo que pasa desde que el procesador solicita información a la memoria principal hasta que este finalice la operación y esté listo para realizar otra actividad.

4.3.2.2 Tiempo de Acceso a la Memoria.

Es el tiempo transcurrido desde que el procesador solicita información a la memoria (enviando la dirección a través del bus de direcciones) hasta que el procesador recibe la información solicitada (a través del bus de datos).

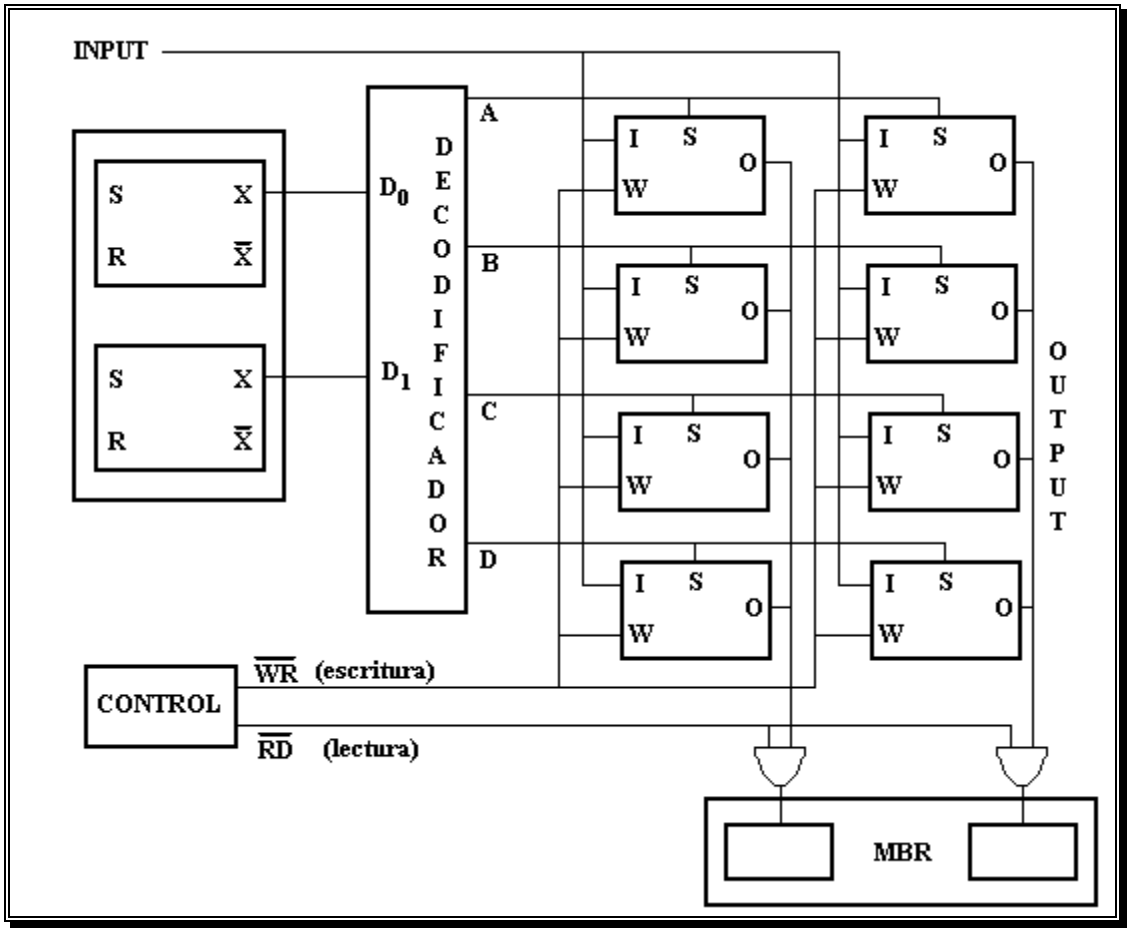
Se puede decir entonces que el tiempo de acceso de memoria está dentro del tiempo de acceso del sistema.

Tenemos que si el tiempo de acceso a la memoria es menor que el tiempo de acceso del sistema, el microprocesador puede trabajar a máxima velocidad sin problemas (retardos); pero en el caso de que el tiempo de acceso a memoria sea mayor que el tiempo de acceso del sistema, el sistema debe tener la capacidad adicional (a través de algún dispositivo electrónico) de detener temporalmente al microprocesador para darle tiempo a la memoria a que entregue una información válida.

Para finalizar, se muestra el diseño lógico de una memoria lineal simple de cuatro palabras (identificadas por las direcciones A,B,C y D) de dos bits cada una. Obsérvese la distribución de los diferentes componentes, entre ellos

el decodificador y las celdas de bits cuyos modelos fueron mostrados con anterioridad.

Fig. 40. Memoria lineal simple.



5. MEMORIA RAM.

5.1. Definición.

La memoria RAM (Random Access Memory), es un dispositivo de memoria temporal, en los que se puede leer o escribir información en forma inmediata. Dicha información puede ser retirada, modificada y almacenada nuevamente.

5.2. Características.

5.2.1 Acceso Aleatorio.

Llamado también acceso al azar o directo. Cuando se hace un acceso a cualquier posición de la memoria siempre es el mismo.

5.2.2 Operación Lectura/Escritura.

Una vez grabada la información en las posiciones de la memoria RAM, éstas pueden ser leídas o escritas. Su contenido puede ser modificado cuando no se necesita un programa; puede ser reemplazado por otro.

5.2.3 Volatilidad.

El contenido (información y/o programas) se pierde cuando se corta el suministro de energía, es decir cuando se apaga el computador.

5.2.4 Capacidad.

Tiene una capacidad base (estándar) de 512 KB a 640 KB, con flexibilidad de expansión.

5.2.5 Velocidad.

Posee una velocidad de acceso de aproximadamente 500 ns.

5.2.6 Tecnología.

Se clasifican en estáticas y dinámicas.

5.3 Uso e Importancia de la Memoria RAM.

La memoria principal del computador es de memoria RAM. El sistema operativo se carga en la memoria RAM, de manera que el procesador pueda comenzar a ejecutar las instrucciones que lo componen.

Para que el procesador pueda ejecutar un programa, una aplicación, etc., se requiere fundamentalmente de la memoria RAM, así como también almacenar, leer o modificar ésta en el momento que se requiera.

5.4 Clasificación de la Memoria RAM Según su Tecnología.

5.4.1 RAM Estática.

Definición: Son memorias de semiconductores basadas en puntos o celdas de memoria tipo biestable (Flip-Flop) que tiene la propiedad de retener el contenido tanto tiempo como interconectado se encuentre la fuente de alimentación de energía.

Características.

a) Permite operaciones de Lectura/Escritura

b) Son de direccionamiento aleatorio

c) Son volátiles

d) Son costosas y ocupan mucho espacio

5.4.2 RAM Dinámica (DRAM).

Definición: En esta memoria se utiliza la permanencia de una superficie magnética que se hace pasar ante un cabezal de Lectura/Escritura para registrar sobre ella información. Los diferentes bits de información se almacenan en forma de cargas eléctricas en un condensador de tipo MOS (Metal Oxide Semiconducter: Semiconductor de Óxido Metálico).

Es la que se emplea normalmente en los computadores, ésta tiende a perder gradualmente la información que contiene debiendo *"Refrescarse"* o *"Restaurarse"*

cada cierto tiempo (es comparado con baterías recargables). El "refrescamiento" se obtiene con un ciclo de lectura en falso que se efectúa simultáneamente sobre varias direcciones de memoria y los datos leídos se ignoran y son generados automáticamente por el hardware.

Características.

a) Cada bit de la memoria dinámica RAM tiende a reescribirse, mediante una unidad de refrescamiento.

b) Posee direccionamiento aleatorio o de acceso directo.

c) Permite almacenar más puntos de información en chips más pequeños.

5.4.3 Comparación.

a) La RAM estática es más rápida que la dinámica.

b) En la RAM estática, el diseño de su circuito es más simple.

c) La RAM estática está constituida por un mínimo de 4 a 6 transistores mientras que la dinámica puede estar formada por un sólo transistor, por lo tanto, el consumo de la RAM estática es mayor que la dinámica.

d) Las RAM estáticas son más sencillas de interconectar. La expansión de la estática es más sencilla dado que la memoria dinámica necesita un sistema adicional de refresco.

e) La tecnología usada en la memoria RAM estática consume más energía, es menos densa y además es más costosa que la empleada en la memoria dinámica. Si se utilizan memorias estáticas en los computadores estos consumirían más corriente.

5.5 Estructura Básica de un Chip de Memoria RAM.

La memoria RAM está constituida por un conjunto de celdas de memoria, capaz de almacenar un bit de información. Ya sabemos que un conjunto de celdas de memoria en forma de arreglo forman una palabra de memoria. Estas palabras son seleccionadas por el **Decodificador de Direcciones**. Las líneas que unen las celdas de memoria se denominan *líneas de bits* conectado a un **circuito de detectar/escribir** y a su vez ese circuito se conecta a las **líneas de entrada y salida**.

La entrada y salida de datos de cada circuito de detectar/escribir están conectadas a una sola *línea de datos bidireccional*. Además de las líneas de direcciones y datos se proporciona dos líneas de control:

a) **Entrada R/$\overline{W}$:** que define el tipo de operación.

R/$\overline{W}$=1 : implica una operación de lectura.

R/$\overline{W}$=0 : implica una operación de escritura.

Como puede apreciarse el hecho que los datos sean leídos o escritos dependerá del estado de la señal de control **R/$\overline{W}$.**

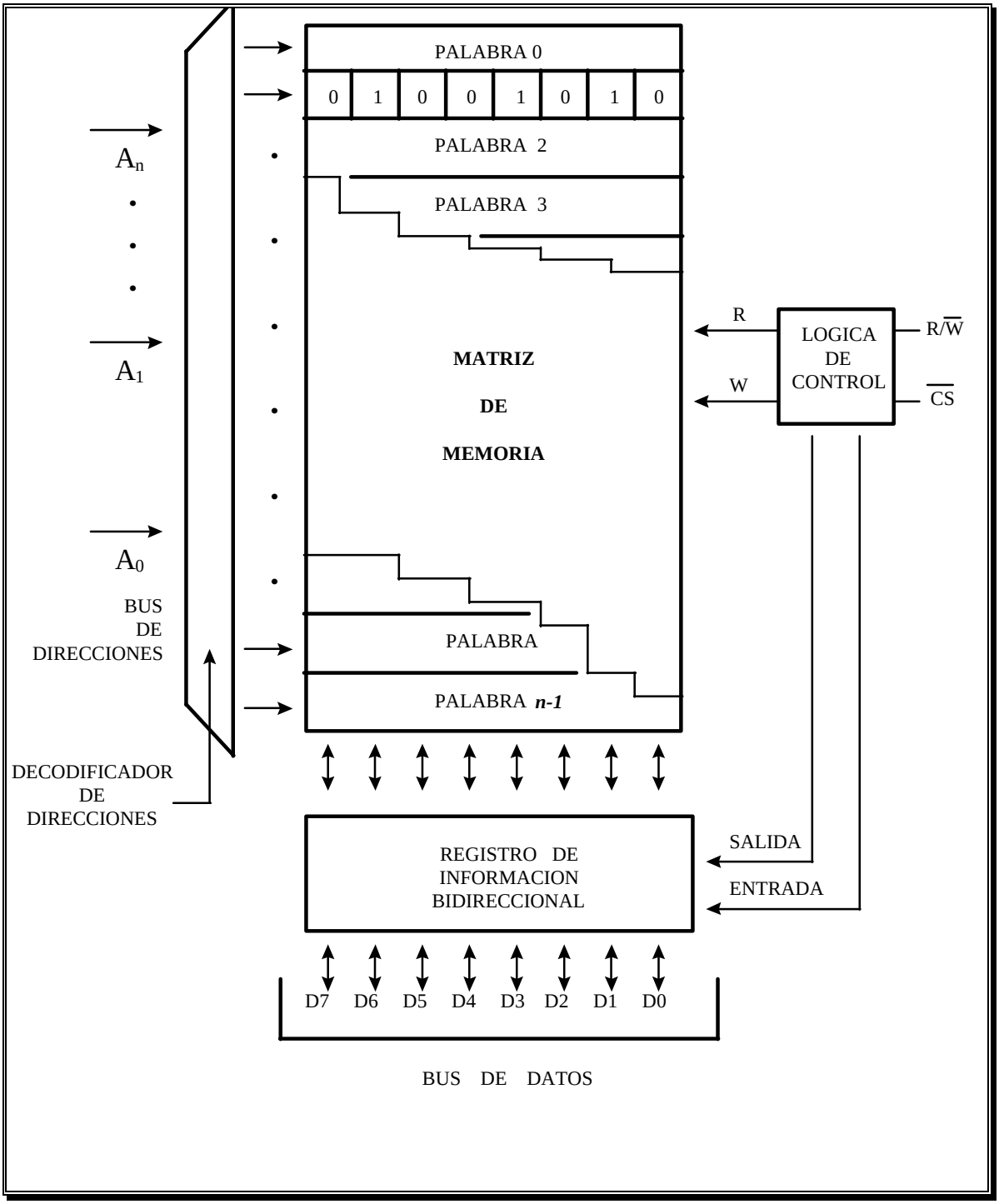
b) $\overline{CS}$**:** Selección (activación) de un chip en particular:

$\overline{CS}$ **= 0:** Permite una conexión efectiva de la Unidad de memoria a los buses de datos y direcciones.

$\overline{CS}$ **= 1:** Las líneas de conexión de la memoria a los citados buses imposibilitan su funcionamiento.

El microprocesador controla y sincroniza las operaciones de lectura/escritura y de selección de chip.

Fig. 41. Estructura de un Chip de RAM.



5.6 Principio de Funcionamiento de la Memoria RAM.

5.6.1 Lectura:

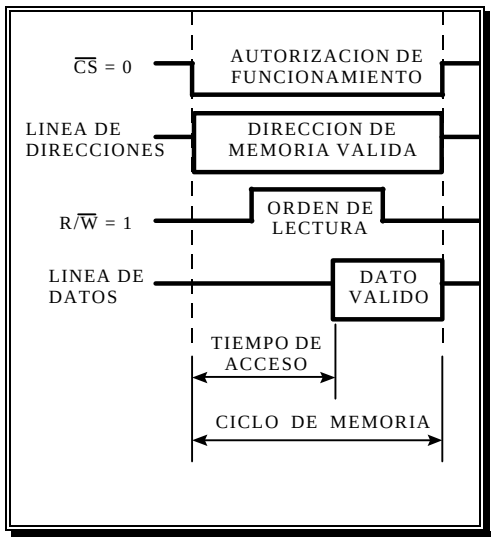
Se realiza de la siguiente manera:

a) Autorizar la actualización de la unidad de memoria posicionando adecuadamente la entrada de control $\overline{CS} = 0$.

b) Llevar la dirección de la palabra de memoria cuyo contenido se desee extraer, a la entrada de direccionamiento de la unidad de memoria.

c) Enviar la orden de lectura a través de la entrada **R/$\overline{W}$=1**. Seguidamente, el contenido de la posición de memoria direccionada pasará al registro de información, y de éste al bus de datos (Ver Figura 42). Luego, el contenido de la posición de memoria direccionada pasará al registro de información, y de éste al bus de datos.

Fig. 42. Principio de Funcionamiento: Lectura



5.6.2 Escritura:

Su procedimiento es el que sigue:

a) Colocar en el bus de direcciones la configuración binaria que identifica a la posición de memoria donde deseamos almacenar el dato.

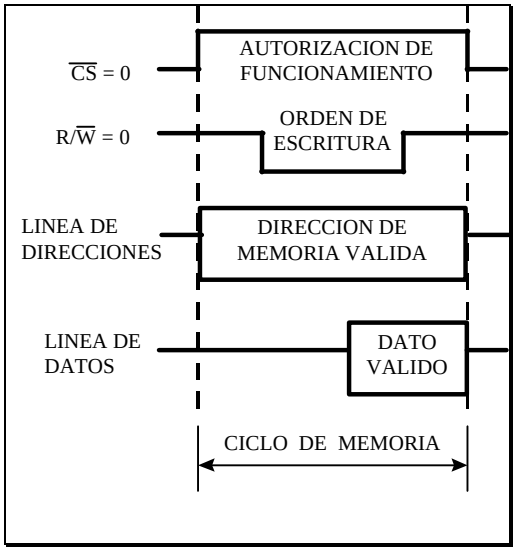
b) Llevar al bus de datos la información a escribir en la celda de memoria.

c) Autorizar la activación de la unidad de memoria mediante $\overline{CS}$**=0**.

d) Mandar la orden de escritura **R/**$\overline{W}$**=0**.

Una vez realizados estos pasos, la información presente en el bus de datos pasará al interior de la posición de memoria direccionada:

Fig. 43. Principio de Funcionamiento. Escritura



6. MEMORIA ROM.

6.1 Definición.

La memoria ROM (Read Only Memory; Memoria de sólo lectura), es un dispositivo electrónico donde se almacena información fija en forma binaria que ha sido grabada en el proceso de fabricación del circuito integrado. Esto funciona como complemento de la memoria total del sistema.

6.2 Características.

- Este tipo de memoria es de sólo lectura.
- Es permanente o no volátil, ya que la información que contiene no se borra al perder el suministro de energía eléctrica.
- Es de acceso aleatorio: Se puede accesar en forma arbitraria a los bits almacenados en una dirección cualquiera, de manera que el tiempo que tarda en llegar a la salida de bits leídos desde cualquier dirección, es aproximadamente el mismo.
- Cuando una de sus posiciones de memoria es direccionada, el contenido que se lee siempre es el mismo.

6.3 Usos.

- Es empleada para almacenar programas o rutinas standard de aplicación específica.

- La aplicación principal de la memoria ROM es la destinada a guardar los programas de arranque de los computadores, esto se lleva a cabo a través de algunos programas no modificables (por ejemplo, ROM BIOS en Pc's), que permiten a la CPU interpretar determinadas órdenes y comandos externos y ejecutar subrutinas definidas por la visualización de datos y la gestión de interrupciones.

El ROM BIOS consiste en un juego de rutinas de manejo de teclado, drives y monitor que proporcionan la interfaz entre el software y el hardware del computador.

6.4 Estructura Básica de la Memoria ROM.

Una memoria ROM está constituida por la siguiente configuración:

6.4.1 Matriz de Memoria.

Es un bloque formado por celdas de memoria donde está almacenada la información en forma binaria (bits) y se fija en la matriz desde el momento de su fabricación. Cada celda es un biestable, es decir, capaz de almacenar los estados "0" y "1".

6.4.2 Decodificador de Direcciones.

Tiene como misión seleccionar la palabra a leer dentro de la matriz de memoria.

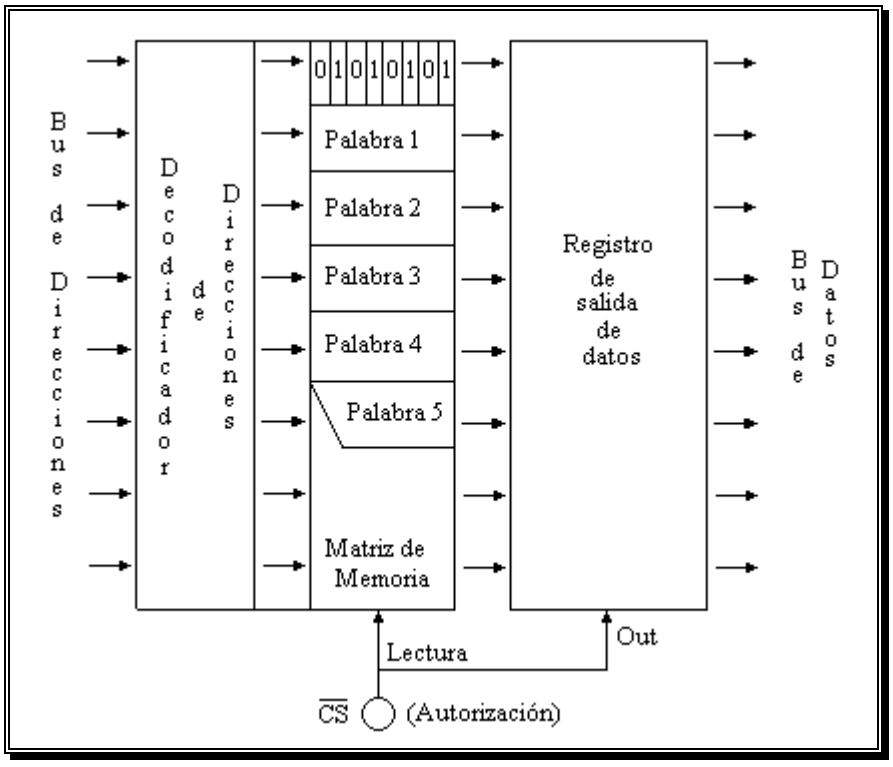
6.4.3 Buffer de Datos.

Consiste en un registro de información de salida que tiene la misión de memorizar temporalmente los datos que van a ser leídos de la posición de memoria direccionada.

6.4.4 Bus de Direcciones.

Las líneas de dirección indican la localización de los datos en la memoria para poder accesar de este modo su contenido. Cuando está completo es un conjunto de m líneas que transporta la dirección y que permite accesar 2^m posiciones de memoria diferentes.

Fig. 44. Estructura básica de la memoria ROM.



6.5 Principio de Funcionamiento.

Para ejecutar la lectura en la memoria ROM se llevan a cabo los siguientes pasos:

- El primer paso es direccionar la posición de memoria cuyo contenido se va a leer, la cual es llevada a través del bus de direcciones (para poder accesar el contenido de lo que deseamos leer); él se encargará de accesar dicho contenido en la matriz de memoria.

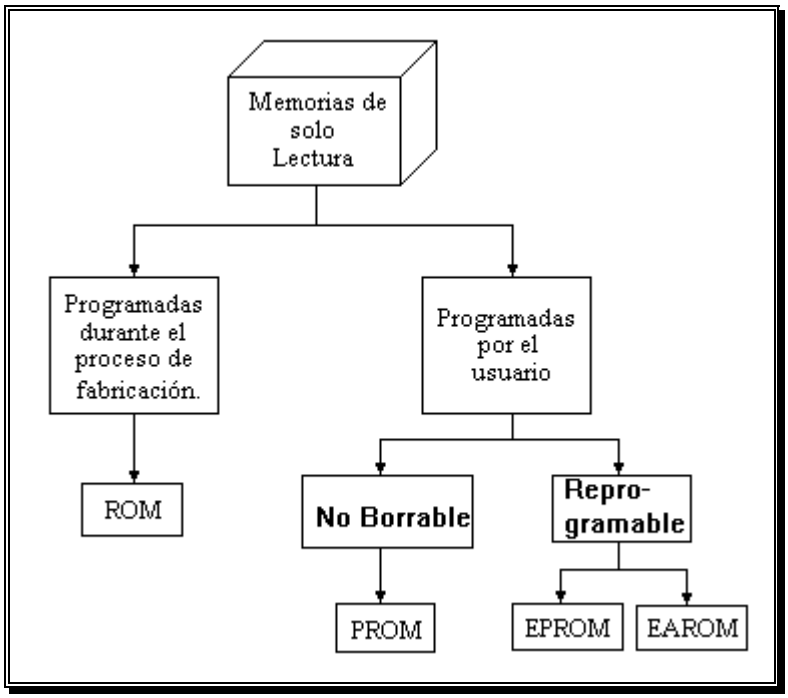
- Autorizar la actuación de la memoria ROM, posicionando la señal $\overline{cs}$ (selector del Chip) para permitir la conexión efectiva de las unidades de memoria a los buses de datos y direcciones.

- A continuación, transcurrido el tiempo de acceso, el dato pasará al registro de salida donde se almacenará temporalmente para luego ser transportado por el bus de datos.

6.6 Tipos de ROM.

En vista de la necesidad de modificar la información almacenada en una memoria de sólo lectura, se han desarrollado algunos tipos de ROM que permiten la posibilidad de posteriores reprogramaciones. Dentro de esta categoría están incluidas las llamadas memorias PROM, EPROM, EAROM.

Fig. 45. Tipos de ROM



6.6.1 PROM (Programable Read-Only Memory).

Memoria programable de sólo lectura. Esta memoria a pesar de que sólo puede ser leída, puede programarse una sola vez, lo cual lo hace el usuario mediante un dispositivo especial llamado "Programador de ROM" el cual funciona estando la tarjeta virgen compuesta por unos (1) de manera que este dispositivo inserta ceros (0) mediante información transmitida por el usuario, a través de pulsos de alta corriente. Este proceso de programación sobre la ROM es irreversible, es decir, sólo permite una única programación lineal.

Las PROM proporcionan una flexibilidad y una conveniencia que no ofrece la ROM, ya que son más rápidas y mucho menos costosas.

Características:

- Sólo permite la lectura.
- Son de acceso aleatorio.
- Son permanentes o no volátiles: La información no puede borrarse.
- La PROM son una alternativa más rápida y menos costosa ya que el usuario puede programarla directamente.

6.6.2 EPROM (Erasable Programable Read-Only Memory).

Memoria de sólo lectura programable y borrable. Esta memoria puede ser programada por el usuario mediante un equipo especial y ser reprogramada (aunque el tiempo que conllevan estas operaciones es extremadamente largo). Esto significa que los códigos (programas) almacenados en la memoria pueden ser programados (por el usuario), borrados y reprogramados para códigos diferentes. El equipo especial que se necesita para el borrado puede ser una instalación de luz ultravioleta.

Es necesario retirar del sistema el módulo de memoria para borrarlo y reprogramarlo. Una vez borrado los datos del EPROM, se necesita disponer de una grabado especial para introducir nuevos datos.

Características:

- Pueden borrarse mediante radiaciones ultravioleta.

- Proporciona flexibilidad durante la fase del desarrollo del sistema digital, porque son capaces de retener la información almacenada durante largo período.
- Permiten sucesivas reprogramaciones y de esta forma se posibilita la depuración del programa hasta lograr un prototipo definitivo, momento en el cual es sustituida por una PROM programada.
- Son programables eléctricamente por el usuario.

6.6.3 EAROM (Electrically Alterable Read-Only Memory).

Memoria de sólo lectura eléctricamente alterable, la cual puede ser programada y borrada mientras se encuentra en el sistema. Las posiciones de memoria deben borrarse antes de ser escritas.

Características:

- Las posiciones de memoria deben borrarse antes de que puedan ser escritas.
- El proceso de borrado puede ser selectivo. Existen dos formas: borrado de palabra y borrado de bloque.
- Son no volátiles y su contenido puede ser alterado ocasionalmente en el circuito, lo que da una característica más eficaz que la EPROM y ROM.
- Desde el punto de vista económico son las más costosas.
- El inconveniente de este tipo de memoria es el número de distintos niveles de tensión que deben ser aplicados para lograr el borrado.

7. MEDIOS DE MEMORIA SECUNDARIA.

7.1 Definición.

Es una memoria auxiliar que permite el almacenamiento de grandes cantidades de datos usando dispositivos más económicos, pero más lentos que los usados en la construcción de la memoria principal.

7.2 Características Globales.

- Estas memorias son controladas por unidades de entrada/salida.
- Son dispositivos en los cuales se lee, se graba información o ambas.
- Tienen gran capacidad de almacenamiento.
- Sólo se borran si se manipulan para tal actividad.
- El acceso a cualquier información por parte del CPU es más complejo que acceso a memoria principal.

7.3 Almacenamiento Masivo.

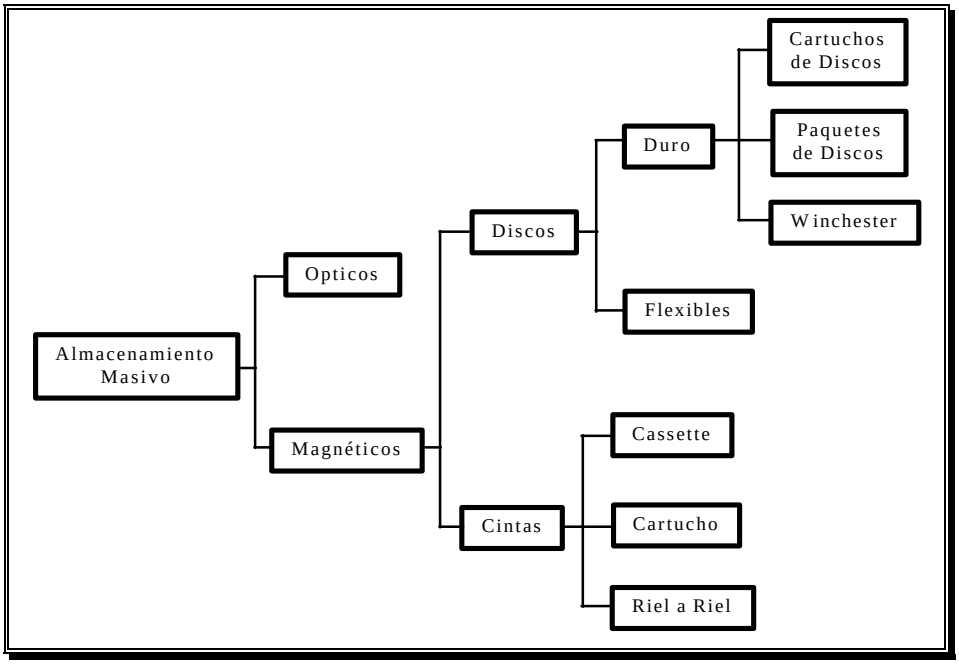
Este término se refiere a una cantidad de datos e información más grande que la cantidad disponible en la memoria principal del computador en un momento determinado. Así, el propósito de un dispositivo de almacenamiento de este tipo es proveer un medio permanente (no volátil) para que la información pueda ser

almacenada, recuperada y manipulada. En efecto, el almacenamiento masivo brinda un lugar para almacenar datos y programas antes y después de su procesamiento.

Un concepto relacionado con este aspecto es lo que se conoce como *archivo*. Cuando estamos usando alguna aplicación desarrollada en cualquier lenguaje de programación, seguramente necesitaremos almacenar la información y datos en algún sitio. Un archivo es un conjunto organizado de información, estructurada en algunos casos en forma de registros.

Es importante tener claro ambos términos ya que en algunos casos se pueden prestar a confusión. Un archivo es una interpretación lógica de una información que se encuentra en un medio de almacenamiento masivo, independiente de la forma en cómo se encuentra esta información en el medio masivo y cómo ésta es manipulada.

Fig. 46. Tipos de Dispositivos de Almacenamiento Masivo



7.4 Medios de Acceso Secuencial.

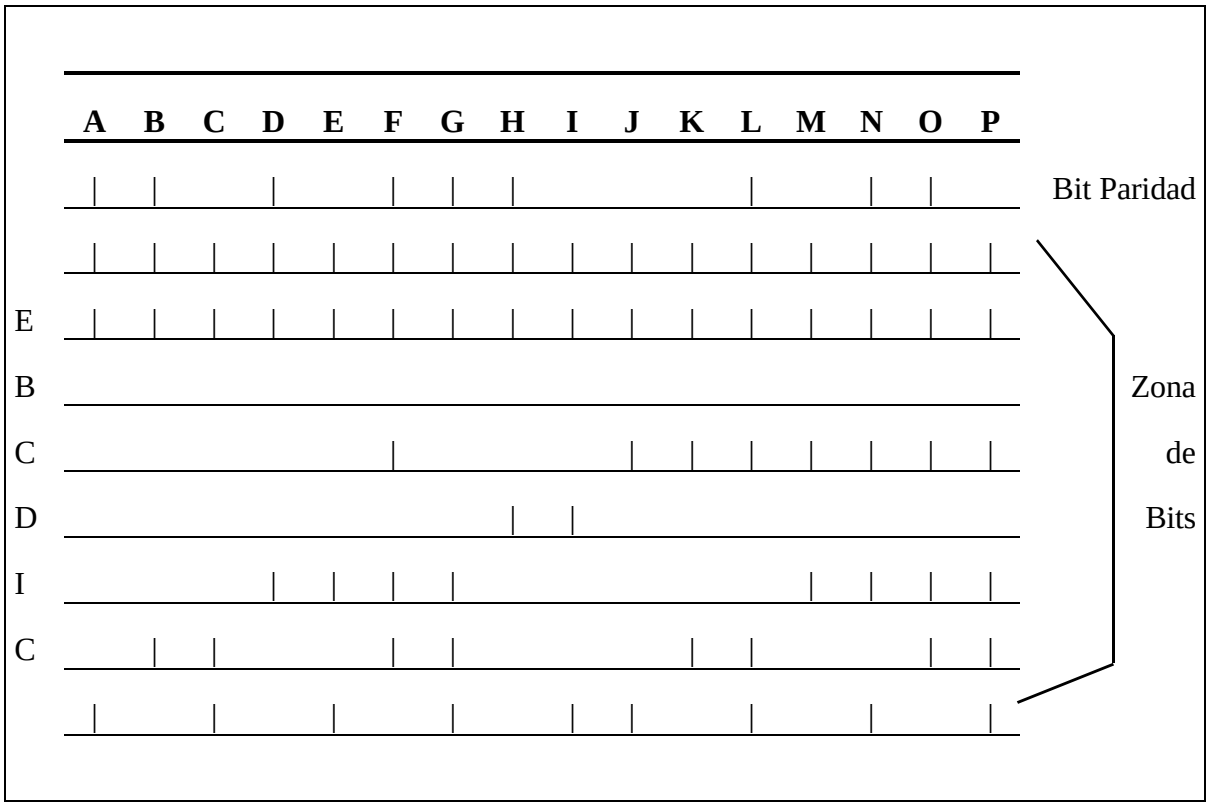
7.4.1 Cinta Magnética.

Una cinta magnética es una tira contínua de plástico enrollado en un carrete (Tape). Esta cinta plástica es tratada con una capa de óxido de hierro que puede ser magnetizada. Generalmente tiene un ancho de ½ de pulgada (1.27 cm. aproximadamente) y tiene una longitud de 400 a 3200 pies (122 a 976 metros aproximadamente). Algunas se encuentran disponibles en formas de cartuchos y cassettes para su uso en el computador.

Los datos en este medio, están almacenados como pequeños puntos magnetizados sobre su cubierta. Aunque estos puntos pueden ser leídos (detectados) por el computador no son visibles al ser humano, como en la mayoría de los dispositivos actuales. Grandes volúmenes de información pueden ser almacenados en una cinta; la densidad de grabación común es de 1600 caracteres por pulgada pudiendo alcanzar la cantidad de 6250 caracteres por pulgada en algunos casos.

El método más común para representar los datos en una cinta magnética usa un esquema de codificación de nueve (9) pistas. La cinta se divide en 9 filas horizontales que se denominan *pistas*, y de esta manera los datos están representados verticalmente en forma de columnas, un carácter por columna. De esta manera, ocho de los nueve bits representa el carácter y el restante se utiliza como bit de paridad, para el chequeo y control de errores. El método de codificación que utiliza es idéntico al EBCDIC (Extended Binary Coded Decimal Interchange Code).

Fig Nro. 47. Vista de una cinta de nueve pistas con paridad par.



La cinta magnética se instala en un tape drive (manejador de cintas). Este mecanismo tiene un cabezal de lectura/escritura (actualmente electromagnético) que crea o detecta los bits magnetizados mientras la cinta al moverse pasa frente a él. Cuando el cabezal se encuentra leyendo los datos, detecta los puntos magnetizados y los convierte en pulsos eléctricos que son enviados al CPU. Cuando se escriben datos, el cabezal magnetiza los puntos apropiados sobre la cinta mientras borra cualquier dato almacenado allí previamente.

Los registros individuales que conforman un archivo, son separados mediante Inter-record Gap (IRG). Esta "brecha" o "intervalo" no contiene datos pero cumple

una función importante ya que a medida que el cabezal se encuentra leyendo sobre la cinta, el IRG le indica el final de un determinado registro, lo que lleva al tape drive a detenerse.

El tamaño de un IRG depende de la velocidad del tape drive. Si el tape drive es muy rápido se necesitan IRG largos y si es muy lentos se requieren brechas cortas.

Fig. Nro. 48. Vista de una cinta magnética con Inter record Gap.

....	IRG	Regis-tro 1	IRG	Regis-tro 2	IRG	Regis-tro 3	IRG	Regis-tro 4	IRG	....

Puede ocurrir que los registros almacenados en la cinta sean muy cortos (pequeños) y los IRG'S muy largos lo cual conduce un desperdicio de la cinta y causa que el tape drive se detenga y arranque en forma constante. Para evitar esta situación, los registros pueden ser agrupados. Estos bloques de registro están separados por Inter-block Gaps (IBG). En lugar de leer un registro corto y parar y así sucesivamente, el cabezal de lectura/escritura lee un bloque de registros a la vez y se detiene y continua de esa forma.

Fig. Nro. 49 Vista de una cinta magnética con Inter Block Gap .

....	IBG	Regis-tro	Regis-tro	Regis-tro	IBG	Regis-tro	Regis-tro	Regis-tro	IBG	....

Usando el método con IBG, tiene dos ventajas importantes sobre el método IRG:

1. La cantidad de almacenamiento disponible en la cinta es usado en forma más eficiente.
2. Se reduce significativamente el número de operaciones de lectura/escritura requeridas.

Las ventajas de usar cintas magnéticas para el almacenamiento de información son las siguientes:

- Los datos pueden transferirse entre la cinta y el CPU a altas velocidades.
- Tienen una alta densidad de grabación, de manera que pueden almacenar gran cantidad de datos en un espacio pequeño.
- Pueden borrarse y usarse continuamente (reusable).
- Su costo es relativamente bajo.
- Es perfecta cuando se requiere accesar la información en forma secuencial.

El uso de la cinta magnética presenta las siguientes desventajas:

- Como es un medio de acceso secuencial, la cinta completa debe leerse de principio a fin, cuando se necesita hacer cambios sobre los datos. Esto representa una cantidad significativa de tiempo.
- Cada cinta debe estar debidamente identificada en cuanto a su contenido para evitar usos indebidos.

- Ciertos factores ambientales pueden distorsionar los datos almacenados como por ejemplo el polvo, temperaturas extremas, electricidad estática, etc. Esto genera la necesidad de tener copias de respaldo para prevenir la pérdida de los datos.

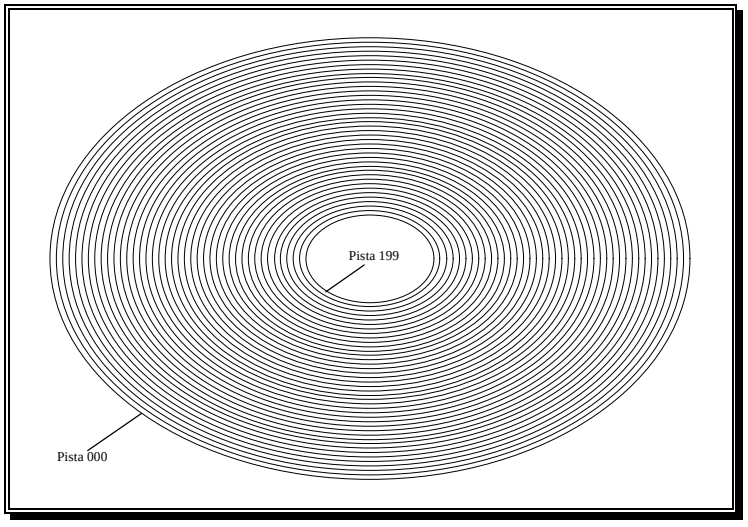
7.5 Medios de Acceso Directo.

7.5.1 Disco Magnético.

Un disco magnético es un plato metálico circular cubierto en ambas caras (lados) con un material magnetizable tal como el óxido de hierro.

Para almacenar y recuperar información, el disco se encuentra girando mientras un *cabezal de lectura/escritura* se posiciona encima de su superficie. Esta información se organiza en círculos concéntricos denominados *pistas*. Una pista nunca coincide con otra, es decir, son independientes.

Fig. Nro. 50. Vista de un disco que posee 200 pistas concéntricas



Generalmente se utilizan un conjunto de discos magnéticos dispuestos en forma de fila y unidos mediante un eje único. Bajo esta forma de organización, se involucran varios elementos y conceptos que la identifican:

- Pistas: Como señalamos anteriormente, son círculos concéntricos en el cual se registra la información. En la mayoría de los casos los círculos más internos tienen que manejarse con mayor densidad de grabación.

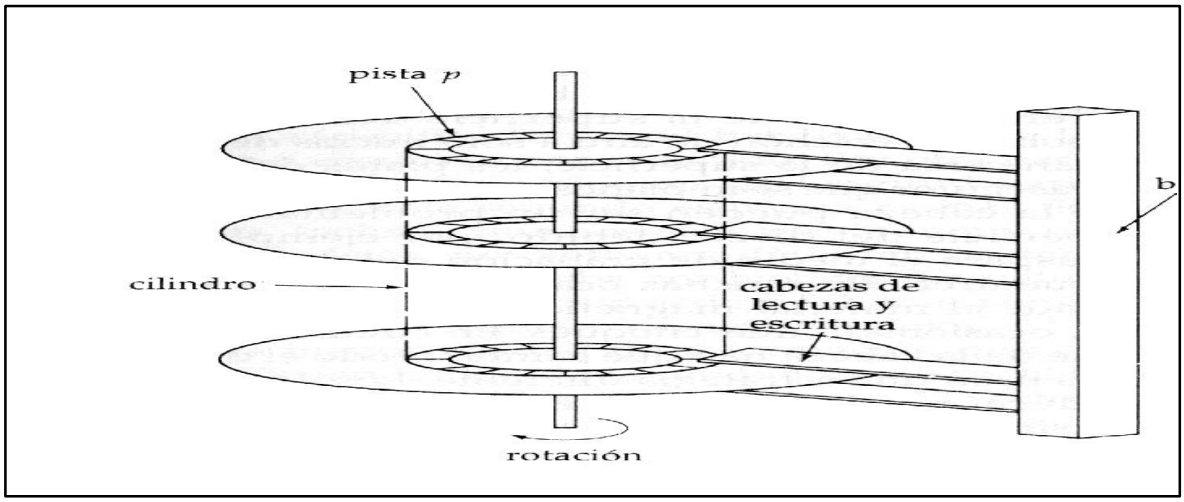
- Sectores: Es la forma en cómo se encuentra dispuesta la información en una pista. Suele haber de 10 a 100 sectores por pista. Pueden tener tamaño fijo o variable. Se utilizan los IRG para separar los sectores.

- Cilíndros: Se refiere a la superficie imaginaria formada por todas las pistas directamente encima y debajo una de otra. En otras palabras, es el conjunto de pistas que se pueden accesar sin necesidad de mover el cabezal de lectura/escritura.

- Cabezal de lectura/escritura: Se encarga de colocar o extraer la información del disco. En la mayoría de los casos se tiene un cabezal por cada superficie de disco que tenga.

- Brazo móvil: Es el que está unido al cabezal de lectura/escritura. Mientras el disco gira, el recorre cada una de las pistas haciendo que el cabezal se mueva.

Fig. 51. Mecanismo de un disco de cabezal móvil



Por otra parte, este conjunto de discos (disk pack) se coloca en la unidad de disco (disk drive) para que la data almacenada en él pueda ser procesada. El disk drive hace girar todos los discos simultáneamente a velocidades superiores de 3600 revoluciones por minuto. En algunos modelos, el disk pack es removible; en otros permanece montado en la unidad. En los disk pack removibles, los discos pueden extraerse de la unidad cuando la data contenida ya no se necesita en un momento dado.

Referente a su funcionamiento, cuando la información almacenada en la superficie de un disco se necesita para su uso, todos los cabezales se mueven hacia las pistas correspondientes sobre las superficies de otros discos ya que ellos están contenidos en el mismo mecanismo. Como todos los cabezales de lectura/escritura se mueven juntos, ellos se posicionan sobre las mismas pistas de todas las

superficies de los discos al mismo tiempo. Sin embargo, algunas unidades de disco tienen un cabezal de lectura/escritura por cada una de las pistas. El acceso es mucho más rápido con este tipo de unidad ya que el mecanismo de acceso no tiene que moverse de una pista a otra.

El computador localiza la información almacenada en el disco magnético mediante su número de superficie en el disco, número de pista y número de registro. Estos elementos definen la dirección en disco de la data. La dirección a disco de un registro está almacenada inmediatamente antes del registro. Los registros en el disco están separados por brechas (gaps) similares al caso de la cinta magnética. Igualmente, la presencia de gaps en cada pista reduce la cantidad de datos que puede ser almacenada en el disco.

Este mecanismo provee acceso directo, lo cual hace que los datos que se usan frecuentemente se almacenen en él. Dependiendo del disk drive es posible manejar cerca de 800.000 caracteres por segundo en una operación de lectura o escritura.

Ahora bien, ¿cómo se leen y escriben los bits de información en un disco?. Los pasos son:

- Antes de escribir cualquier dato en un disco, las partículas de hierro están dispersas según un patrón aleatorio dentro de una película magnética que recubre la superficie del disco. Para organizar las particulas en datos, la electricidad se desplaza a través de una bobina de alambre enrollada en un núcleo ferromagnético en el cabezal de lectura/escritura del mecanismo de la unidad; el cabezal está suspendido sobre la superficie del disco. La

electricidad convierte el núcleo en un electroimán que puede imantar las moléculas del revestimiento.

- Al pasar sobre el disco, la bobina induce un campo magnético en el núcleo. El campo, a su vez, magnetiza las moléculas de hierro del revestimiento del disco, de forma que sus polos positivos apuntan hacia el polo negativo del cabezal de lectura/escritura, y sus polos negativos apuntan hacia el polo positivo del cabezal.
- Después que el cabezal crea una banda magnética en el disco que gira, se forma una segunda banda a su lado. Conjuntamente, las dos bandas representan el elemento discreto menor de los datos que un computador puede manipular: un bit. Si éste va a representar un 1 binario, después de crear la primera banda, la corriente de la bobina se invierte, de manera que los polos magnéticos del núcleo se intercambian y las moléculas de la segunda banda se magnetizan en dirección opuesta. Si el bit es un 0 binario, las moléculas de ambas bandas se alinean en la misma dirección.
- Cuando se almacena un segundo bit, la polaridad de su primera banda es siempre la opuesta de la banda precedente, con el fin de indicar que comienza un nuevo bit. Incluso la unidad más lenta tarda sólo una fracción de segundo en crear cada banda.
- Para leer los datos, no se envía nada de corriente al cabezal de lectura/escritura cuando pasa sobre el disco. En vez de eso, tiene lugar la inversión magnética del proceso de escritura. Los grupos de moléculas polarizadas del revestimiento del disco son, por sí mismas, diminutos imanes que crean un campo magnético a través del cual pasa el cabezal de lectura/escritura. El movimiento del cabezal a través del campo magnético

genera una corriente eléctrica que se desplaza en una u otra dirección a través de los hilos que salen del cabezal. La dirección en que fluye la corriente depende de las polaridades de las bandas. Mediante la detección de las direcciones del desplazamiento de la corriente, el computador puede saber si el cabezal de lectura/escritura pasa sobre un 1 o un 0.

Algunas ventajas de los discos magnéticos en relación a las cintas magnéticas son:

- Los archivos pueden ser organizados de manera que su acceso pueda ser secuencial o en forma directa.
- Por su tiempo de acceso rápido, permite que los datos se puedan acceder casi en forma inmediata.
- Con un software apropiado, una transacción simple puede simultáneamente actualizar o cambiar varios archivos almacenados en el disco.

Entre las desventajas se incluyen:

- Los disk pack son un medio de almacenamiento relativamente costoso el cual puede ser diez veces mayor que el costo de una cinta magnética en términos de costo por carácter almacenado. No obstante los diskettes reducen significativamente este costo.
- El almacenamiento requiere una programación más complicada para manejar el dispositivo y poder lograr los accesos a los registros y actualizar los archivos. Se necesita personal altamente especializado para mantener esta característica.

- La facilidad relativa para lograr el acceso a la data almacenada puede crear problemas de seguridad.

7.5.2 Discos Flexibles.

El disco flexible (floppy diskette; diskette) apareció en 1973 para reemplazar a las tarjetas perforadas como medio de entrada de datos pero adicionalmente, para almacenar programas y archivos de datos. En la actualidad, los diskettes son producidos en dos tamaños, 5¼ pulgadas y 3½ pulgadas. Están fabricados con material plástico y cubiertos con una sustancia de óxido magnetizable. Su uso se ha popularizado debido a que es reusable, fácil de almacenar, transportar y por su bajo peso.

Fig Nro. 52. (a) Partes de un Floppy Disk (3 1/2”)

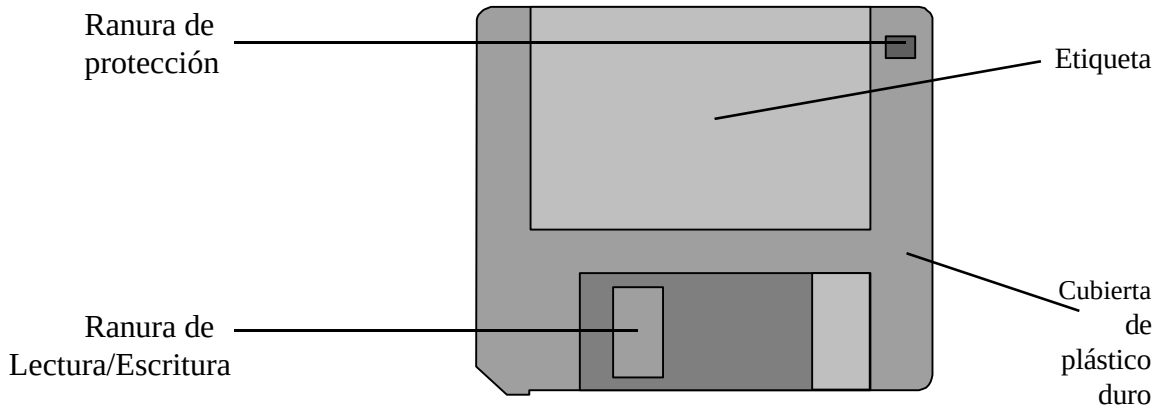
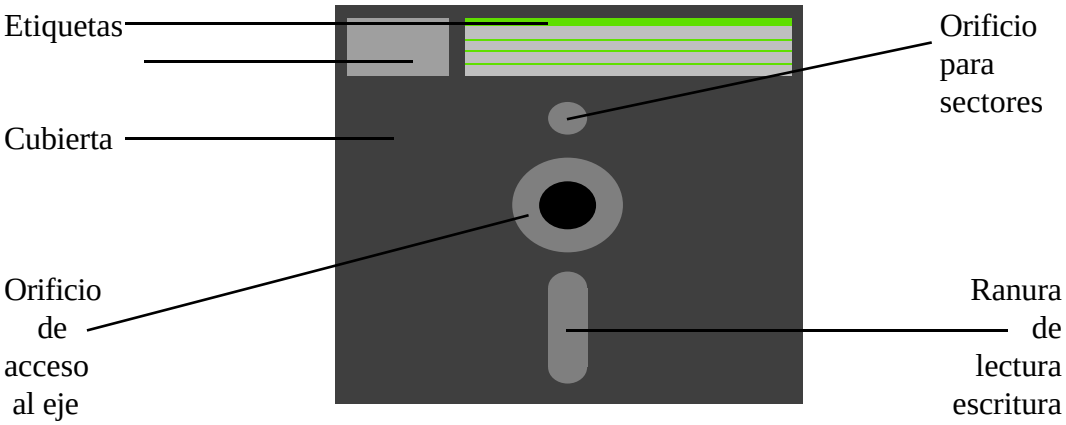


Fig Nro 52(b). Partes de un Floppy Disk (5 1/2”) Cont.



Los datos son almacenados en el floppy disk mediante puntos magnetizables sobre las pistas como en el caso de los discos magnéticos; los elementos son direccionados mediante su número de pista y número de sector. El cabezal de lectura/escritura accede al disco a través de un orificio en forma rectangular que se encuentra en la cubierta del diskette y la cual se denomina ranura de entrada/salida. El cabezal se mueve hacia adelante y hacia atrás para leer o escribir la data sobre el disco. A diferencia de los sistemas de disco duro, el cabezal roza la superficie del disco. El disco gira a una velocidad aproximada de 360 revoluciones por segundo.

La densidad de grabación en este medio se encuentra definida por dos factores: la densidad por pista que se encuentra medida por el número de pistas por pulgada y la densidad lineal la cual es el número de bytes por pulgada por pista. Con la tecnología actual estas densidades se han ido incrementando paulatinamente. La capacidad de almacenamiento de los diskettes de 3½ pulgadas es mayor que los de 5¼ pulgadas. Esto se debe a que el floppy disk más pequeño tiene mayor densidad de grabación.

7.5.3 Disco Duro (Hard Disk).

Este dispositivo almacenará la información en unos platos rígidos de aluminio cubierto con un material óxido-magnetizable. Su formato ha variado en tamaños de 14, 8, 5¼ y 3½ pulgadas. Estos dos últimos se utilizan sobre todo en computadores personales y modelos portátiles. Los discos de 14 y 8 pulgadas se encuentran en algunos sistemas minicomputadores y mainframes.

Las capacidades de un hard disk han variado desde 10 millones de bytes (10 MB) a tamaños que superan actualmente al billón de bytes (1 GB).

Para los usuarios de computadores personales, la ventaja de capacidad de un disco duro sobre el floppy disk, ofrece el acceso a cientos de programas y datos sin necesidad de intercambiar varios diskettes. Adicionalmente, el tamaño máximo de un archivo es mayor en un disco duro y este opera 20 veces más rápido que los floppy disk. La desventaja radica en que la información almacenada debe ser respaldada regularmente y que son más sensibles a movimientos bruscos que en el caso de los discos flexibles.

Debido a que los discos duros son rígidos, varios se colocan en forma de pila en un mismo eje tal como vimos en el caso del disk pack. Al igual que en el floppy disk, el número de pista y la densidad varía según el modelo.

Otras formas de dispositivos magnéticos incluye los drives Winchester que son un tipo de disco duro en el cual el disco y el cabezal de lectura/escritura viven encerrados en una misma caja, lo cual reduce el efecto de agentes atmosféricos contaminantes; los cartuchos de discos, muy similares a las cintas de video que pueden ser insertados y removidos del computador de la misma manera que los floppy disk y finalmente los arreglos de disco, en el cual se combinan y reorganizan múltiples disk drive dentro de una misma unidad.

7.6 Métodos Modernos de Almacenamiento.

7.6.1 RAM Disk

El acceso a los datos es relativamente bajo comparado con la velocidad en la cual un microprocesador puede manipularlos. Los discos de memoria de acceso aleatorio (RAM Disks) son un tipo de dispositivo de almacenamiento cuya velocidad puede aproximarse a la velocidad del microprocesador.

Consiste en una tarjeta RAM que contiene chips de RAM y se coloca en el computador en la misma ranura (Slot) donde se inserta la tarjeta manejadora del disco. Esto hace que la computadora trate a la tarjeta de RAM como si fuese un disk drive. Aunque los RAM disks no son discos físicos separados, ellos funcionan como los diskettes.

Algunos de estos dispositivos, usados en los microcomputadores pueden almacenar data por encima de los 128 MB y tienen una tasa de recuperación 50 veces más rápido que un floppy disk.

Su ventaja básica es la velocidad. La data almacenada en RAM puede ser transferida de una parte de la RAM a otra en forma más rápida que si la transferencia se hiciera desde un disco hacia RAM. La desventaja es que requiere una fuente de energía contínua como cualquier memoria RAM: cuando la fuente falla o se desconecta, los datos se pierden. No obstante, algunos fabricantes proveen baterías de respaldo para ser usadas en el caso de que ocurra lo antes señalado.

7.6.2 Tecnología Láser.

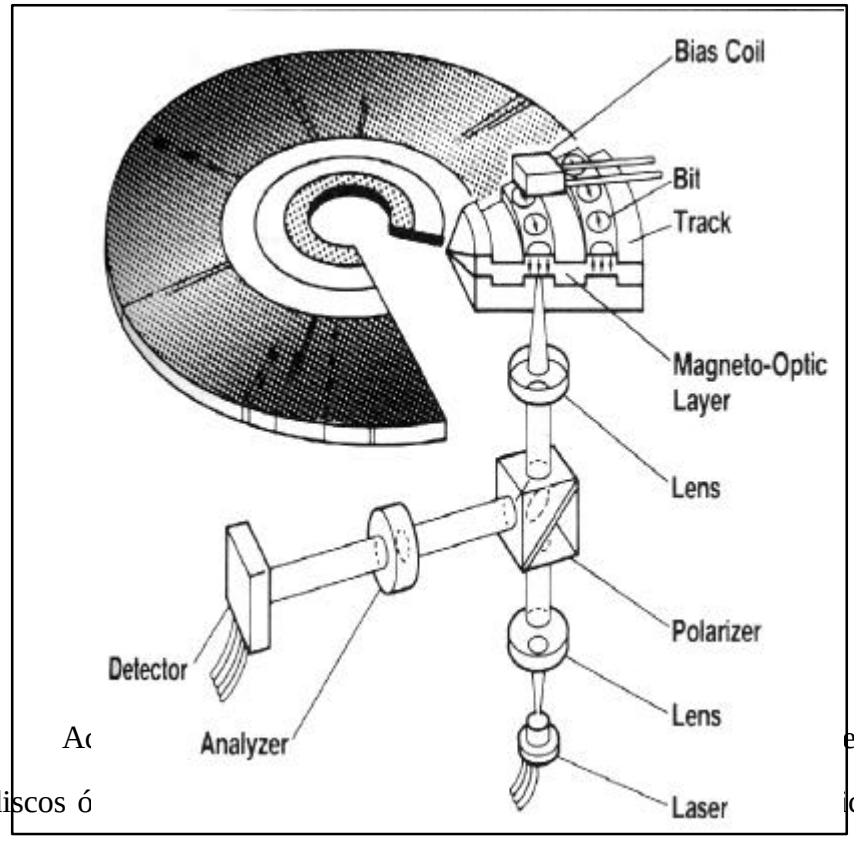
El uso de la tecnología del láser provee una oportunidad de almacenar datos masivos a un costo altamente reducido. Un sistema de almacenamiento con láser puede almacenar cerca de 128 billones de caracteres con cerca de una décima parte del costo de un medio magnético. En este mecanismo, los datos son grabados cuando un rayo láser forma unos patrones en la superficie de una hoja de poliéster cubierta con una fina capa de rodio (metal que se encuentra mezclado con platino).

Para leer los datos, el láser refleja una luz sobre la superficie reconstruyéndolo en una cadena de dígitos binarios (bits). Cuando la información es almacenada con un láser es resistente a las alteraciones, no se deteriora con el uso excesivo, es inmune a la radiación electromagnética y no sufre daños si hay fallas en la unidad de alimentación eléctrica.

Un desarrollo en tecnología láser es el disco óptico o disco láser. Es mucho más rápido que los discos duros pero ligeramente más lento comparado con los

RAM disks. Su gran ventaja es su extensa capacidad de almacenamiento, que pueden almacenar más de 600 MB de datos. Los bits de datos son almacenados con la presencia o ausencia de un pequeño pit (agujero) hecho en el disco mediante un rayo láser de punto. Una línea de una pulgada de longitud contiene cerca de 5000 pits (bits de dato).

Fig. Nro. 53. Mecanismo de Tecnología Láser.



Ac es versiones de discos ó icos de una sola escritura y (3) Discos ópticos borrables.

7.6.2.1 Discos Ópticos de Solo Lectura.

Estos discos tienen una igualdad funcional a la memoria de solo lectura (ROM). La versión más popular de este tipo de disco emplea la misma tecnología de un disco compacto (CD) que ha llegado a ser muy utilizado y popular para sonido. La tecnología es digital y está basada en un disco óptico de 4¾ de pulgadas que puede almacenar 540 MB en un lado. El dispositivo es comúnmente llamado CD-ROM.

Los CD-ROM'S son fabricados en forma similar a las grabaciones fonográficas convencionales. Un disco maestro se utiliza para crear las copias del mismo.

El funcionamiento de una unidad de CD-ROM es el siguiente:

- Un motor varía constantemente la velocidad de giro del disco CD-ROM, para que, con independencia de la situación donde se encuentre en cada instante un componente llamado detector, en relación con el radio del disco, la parte de éste que se encuentra inmediatamente encima de ese detector esté moviéndose siempre a la misma velocidad.
- El láser proyecta un haz de luz concentrado, que es enfocado luego por una bobina de enfoque.
- El rayo láser atraviesa la capa protectora de plástico e incide en la capa reflectora del fondo del disco, que se asemeja a una lámina de aluminio.
- La superficie de la capa reflectora presenta alternativamente entradas y salientes. Los salientes (lands) son zonas planas; los entrantes (pits) son diminutas concavidades en la capa reflectora. Estos dos tipos de superficies son los registros de los 1 y 0 usados para almacenar los datos.

- La luz que da en un entrante se dispersa, pero la luz que incide en un saliente se refleja de nuevo en el detector, donde pasa a través de un prisma que desvía el rayo láser reflejado hacia un diodo fotosensible.
- Cada impulso luminoso que llega al diodo fotosensible genera una pequeña corriente eléctrica. Estas corrientes se cotejan con un circuito regulado, generando una cadena de 1 y 0 suceptible de ser interpretada por el computador.

7.6.2.2 Discos Ópticos de una Sola Escritura.

También denominados WORM, son discos en blanco o vírgenes que son grabados por el usuario. Para escribir la data, un poderoso rayo de luz láser crea pequeños pits o marcas sobre la cubierta que envuelve la superficie del disco. Una vez hecho esto, los pits o marcas no pueden ser borradas. Para recuperar la data, se utiliza otro tipo de láser menos poderoso para leer el patrón de bits.

En la actualidad se utilizan para reemplazar el almacenamiento en microfilms y su versatilidad permite almacenar documentos, fotos, dibujos y música en forma digitalizada.

7.6.2.3 Discos Ópticos Borrables.

Estos discos usan lásers para leer y escribir información hacia y desde el disco. Adicionalmente usan un material magnético sobre la superficie del

disco y un cabezal de escritura magnética para conseguir la capacidad de que sea borrable la información. Para escribir sobre tales discos, un rayo láser abre una pequeña marca (spot) sobre él y entonces un campo magnético se aplica para reversar la polaridad magnética del spot. Debido a esta propiedad, estos discos son conocidos como discos ópticos-magnéticos.

8. Memoria Caché

8.1 Definición.

Es una memoria auxiliar de gran velocidad y baja capacidad, que se "añade" a la memoria principal para acelerar su funcionamiento y optimizar su rendimiento. Suele ser de 5 a 10 veces más rápida que la memoria RAM y su capacidad estándar va de 8 KB a 256 KB.

8.2 Características. Funcionamiento General.

El análisis de un gran número de programas típicos ha demostrado que las referencias a la memoria en cualquier intervalo de tiempo dado tienden a quedar confinadas dentro de unas cuantas zonas localizadas en la memoria. Este fenómeno se conoce como *localidad de referencia*. La razón de ser de esta propiedad se puede entender considerando que un programa de computadora típico influye en forma secuencial, donde se encuentran con frecuencia ciclos y subrutinas. Cuando se ejecuta un ciclo de un programa, el CPU se refiere en repetidas ocasiones al conjunto de instrucciones contenido en la memoria que constituye el ciclo. Cada vez que se hace un llamado a una subrutina dada, su conjunto de instrucciones se captura en la memoria. Por lo tanto, los ciclos y subrutinas tienden a localizar las referencias a la memoria para instrucciones de captura. En menor grado, las referencias de la memoria a datos también tienden a ser localizadas. Los procedimientos de búsqueda en tablas se refieren en repetidas ocasiones, a una porción de la memoria donde se almacena la tabla de elementos. Los procedimientos iterativos se refieren a localidades comunes de la memoria y arreglo de números confinados a una porción local de la memoria. El resultado de todas estas observaciones es la propiedad de *localidad de referencia* que

asevera que en un intervalo de tiempo dado, las direcciones generadas por un programa se refiere a unas cuantas zonas localizadas de la memoria en repetidas ocasiones mientras en resto de la memoria observa accesos relativamente poco frecuentes.

Si la porción activa del programa y los datos se colocan en una memoria compacta y veloz, se puede reducir el tiempo promedio de acceso a la memoria, con lo cual se reduce el tiempo de ejecución total del programa. Dicha memoria compacta y veloz se conoce como memoria caché, y está ubicada entre el CPU y la memoria principal. El tiempo de acceso a la memoria caché es menor que el de la memoria principal en un factor de 5 a 10. La memoria caché es el componente más veloz en la jerarquía de la memoria y se aproxima a la velocidad de componentes del CPU.

La operación básica de la caché es como sigue. Cuando el CPU necesita tener acceso a la memoria, se examina la caché. Si se encuentra la palabra en la caché, se lee de la memoria de alta velocidad. Si la palabra direccionada por el CPU no se encuentra en la caché, se tiene acceso a la memoria principal para la lectura de la palabra. Después se transfiere un bloque de palabras que contiene la que se acaba de acceder de la memoria principal a la caché. El tamaño del bloque puede variar de una palabra (la recién accedida) a cerca de 16 palabras adyacentes a ésta. En esta forma, se transfieren algunos datos a la caché de modo que referencias futuras a la memoria puedan hallar la palabra requerida en la caché de alta velocidad.

El *desempeño de la memoria* caché se mide frecuentemente en términos de una cantidad llamada razón de aciertos. Cuando al CPU se refiere a la memoria y

encuentra la palabra en la caché, se dice que produce un *acierto*. Si la palabra no se encuentra, entonces se encuentra en la memoria principal y se cuenta como un *desacierto*. La proporción entre el número de aciertos y las referencias totales de la CPU a la memoria (aciertos más desaciertos) es la *razón de aciertos*. La razón de aciertos se mide en forma experimental ejecutando programas representativos en la computadora y midiendo el número de aciertos y desaciertos durante un intervalo de tiempo dado. Se ha informado razones de aciertos de 0.9 y mayores. Esta razón de aciertos elevada verifica la validez de la propiedad de la localidad de referencia.

La característica básica de esta memoria es su rápido tiempo de acceso. Por lo tanto, se desperdicia muy poco tiempo cuando se buscan palabras en la caché. La transformación de datos de la memoria principal a la memoria cache se conoce como proceso de *mapeo*. Existen tres tipos de procedimientos de mapeo que son de interés práctico al considerar la organización de la memoria caché:

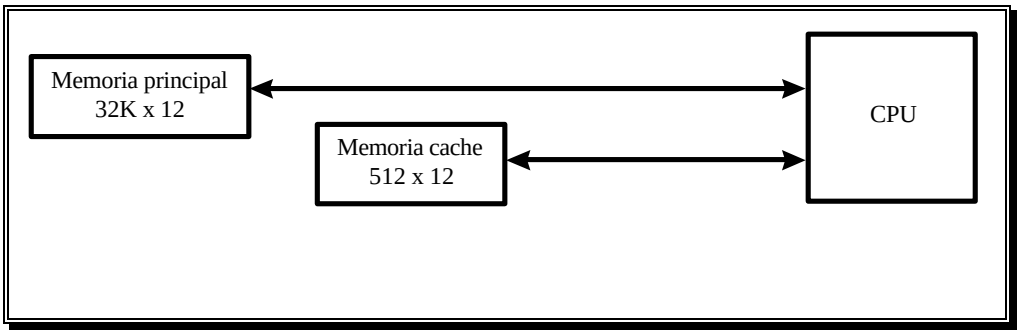
1.- Mapeo Asociativo.

2.- Mapeo Directo.

3.- Mapeo Asociativo de Conjunto.

Para ayudar en la explicación de estos tres procedimientos de mapeo utilizaremos un ejemplo específico de una organización de memoria que se ilustra en la figura 54.

Fig. 54. Ejemplo de memoria cache.



Para este ejemplo, la memoria principal puede alojar 32K palabras de 12 bits cada una y la caché puede almacenar 512 de estas palabras en cualquier momento dado. Para toda palabra almacenada en la caché existe una copia duplicada en la memoria principal. El CPU se comunica con ambas memorias. Primero, envía una dirección de 15 bits a la caché. Si hay un acierto, la CPU acepta la palabra de datos de 12 bits de la caché. Si hay un desacierto, la CPU lee la palabra de la memoria principal y luego esta se transfiere a la caché.

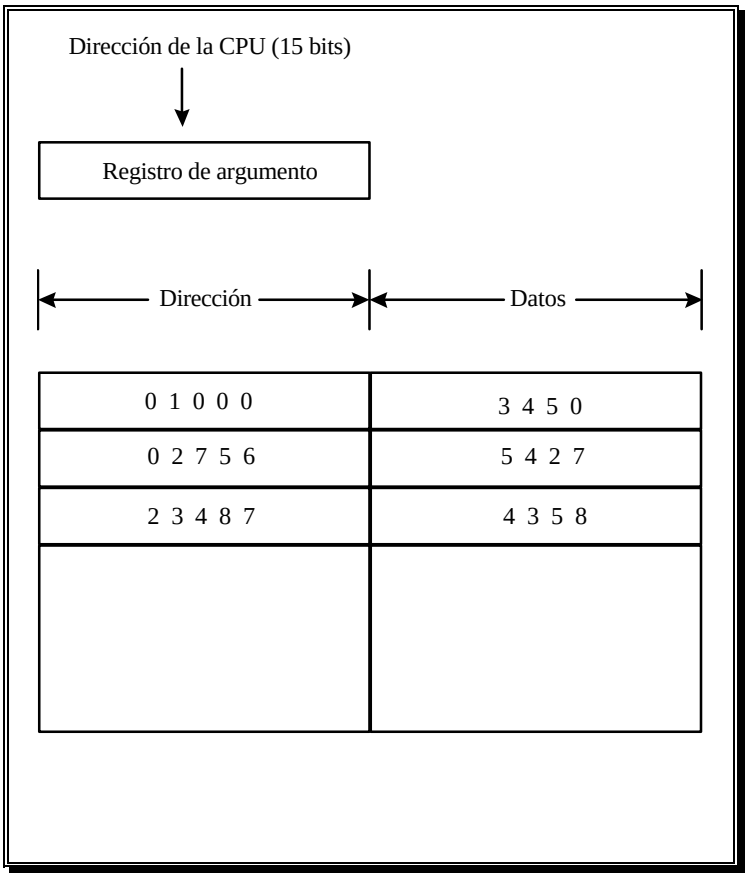
8.3 Tipos de Memoria Cache.

8.3.1 Mapeo Asociativo.

La organización de la memoria cache más veloz y más flexible utiliza una memoria asociativa que aloja la dirección y el contenido (datos) de la palabra de la memoria. Esto hace posible que cualquier localidad de la caché almacene cualquier palabra de la memoria principal. El diagrama presenta tres palabras almacenadas en la cache. El valor de dirección de 15 bits se muestra como un número octal de cinco dígitos y su palabra de datos de 12 bits correspondiente se ilustra como un número octal de cuatro dígitos. Una dirección de 15 bits de la

CPU se coloca en el registro de argumento y en la memoria asociativa se busca una dirección coincidente. Si se encuentra la dirección, los datos de 12 bits correspondientes se leen y se envían al CPU. Si no hay coincidencia, se tiene acceso a la memoria principal para encontrar la palabra. Después, el par dirección-datos se transfiere a la memoria caché asociativa. Si se encuentra llena, debe desplazarse un par dirección-datos para dejar espacio para un par que se necesite pero que no esté en ese momento en la caché. La decisión concerniente a qué par desplazar se determina a partir del algoritmo de reemplazo que el diseñador escoge. Un procedimiento sencillo consiste en sustituir las palabras de la caché en orden de eliminación recíproca siempre que se solicite una nueva palabra de la memoria principal. Esto constituye un criterio de remplazo donde el primero en entrar es el primero en salir (PEPS; FIFO).

Fig. 55. Caché de mapeo asociativo (todos los números están expresados en octal)

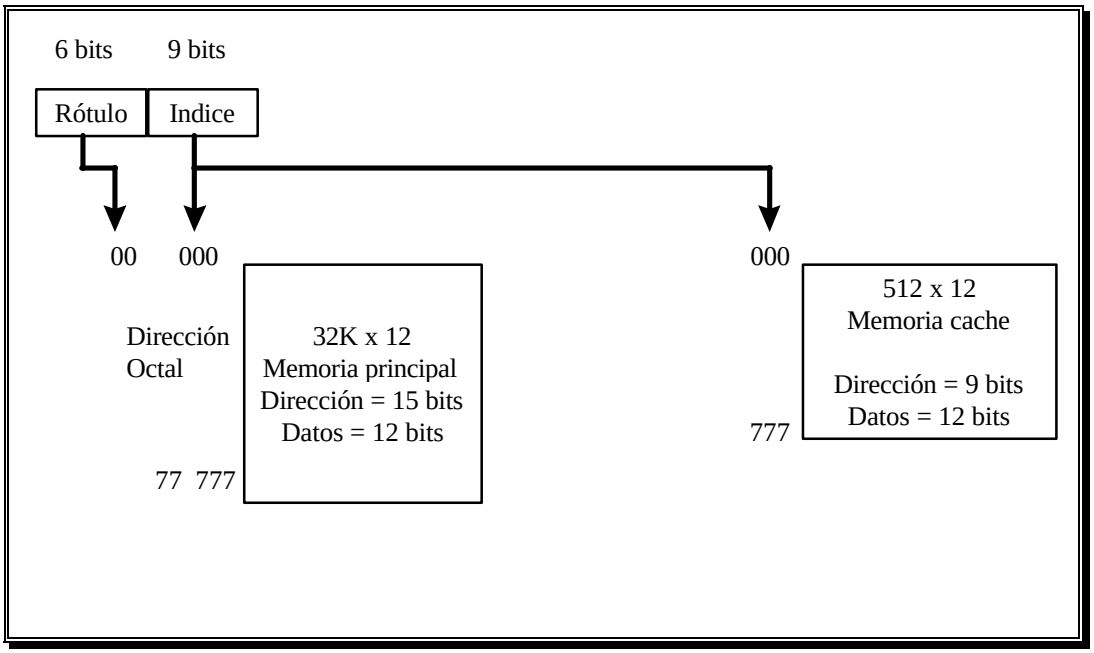


8.3.2 Mapeo Directo.

Las memorias asociativas son costosas comparadas con las memorias de acceso aleatorio debido a la memoria agregada que está asociada con cada celda. La posibilidad de utilizar una memoria de acceso aleatorio para la caché se investiga en la figura 56. La dirección de 15 bits de la CPU se divide en dos campos. Los nueve bits de orden inferior constituyen el campo índice y los seis bits restantes forman el campo rótulo. La figura muestra que la memoria principal necesita una dirección que incluya los bits índice y rótulo. El número de

bits en el campo índice es igual al número de bits de dirección que se requieren para lograr el acceso a la memoria cache.

Fig. 56. Relaciones de direccionamiento entre la memoria principal y la caché



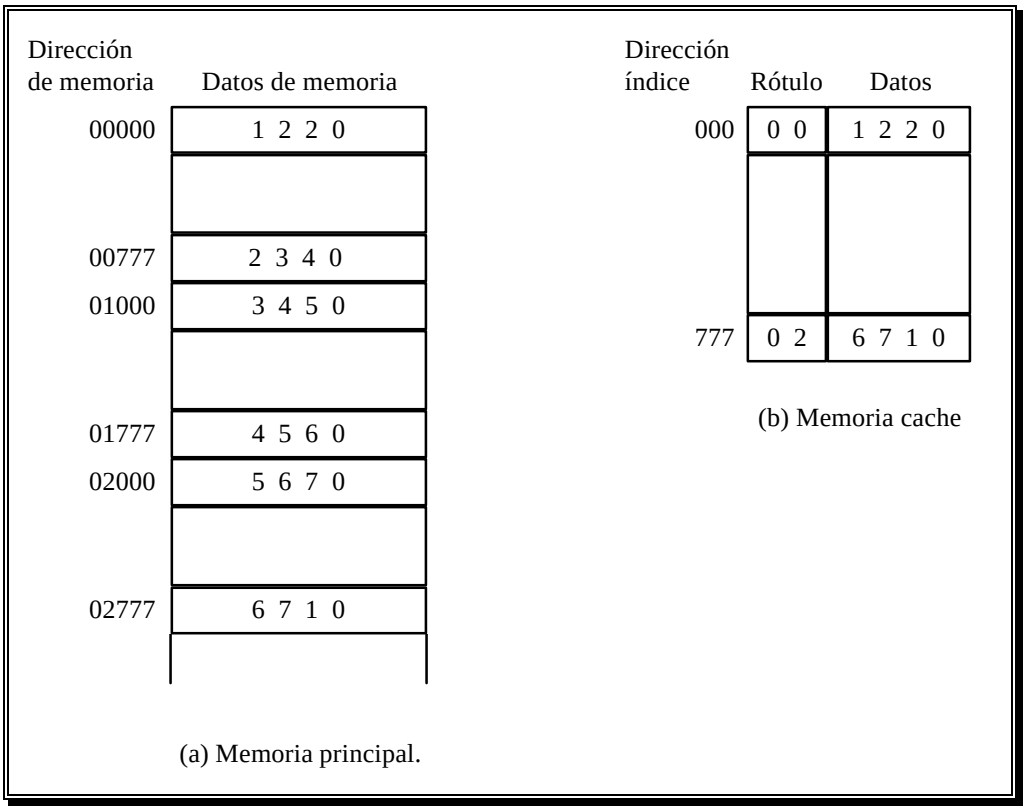
En el caso general, hay 2^k palabras en la memoria cache y 2^n en la memoria principal. La dirección de memoria de n bits se divide en dos campos: K bits del campo índice y n-k bits para el campo rótulo. La organización de la memoria caché de mapeo directo emplea la dirección de n bits para tener acceso a la memoria principal y el índice de k bits para tener acceso a la caché. La organización interna de las palabras en la memoria caché se ilustra en la figura 57. Cada palabra contenida en la caché consta de la palabra de datos y su rótulo asociado. Cuando se trae una palabra nueva a la caché, los bits del rótulo están almacenados junto con los bits de datos. Cuando el CPU genera una solicitud de memoria, el campo índice se utiliza para que la dirección tenga acceso a la

caché. El campo rótulo de la dirección de la CPU se compara con el campo rótulo de la palabra leída de la reserva. Si coinciden los dos rótulos hay un acierto y la palabra de datos que se busca se encuentra en la caché. Si no hay coincidencia hay un desacierto y la palabra requerida se lee de la memoria principal. Después se almacena en la caché junto con el nuevo rótulo, reemplazando al valor anterior.

La desventaja del mapeo directo es que la razón de aciertos puede disminuir considerablemente si dos o más palabras son direcciones que tengan el mismo índice pero rótulos diferentes se acceden en repetidas ocasiones. Sin embargo, esta posibilidad se minimiza por el hecho de que tales palabras están relativamente muy apartadas en el intervalo de direcciones (múltiplos de 512 localidades en este ejemplo).

Para ver cómo opera el esquema de mapeo directo, considérese el ejemplo numérico que se ilustra en la figura 57. La palabra en la dirección cero se almacena en ese momento en la caché (índice = 000; rótulo = 00; datos = 1220). Supóngase que la CPU desea ahora tener acceso a la palabra en la dirección 02000. La dirección índice es 000, de tal suerte que se utiliza para lograr el acceso a la cache. Después se comparan los dos rótulos. El rótulo de la cache es 00 pero el rótulo de la dirección es 02, lo cual no produce una coincidencia. Por lo tanto, se tiene acceso a la memoria principal y la palabra de datos 5670 se transfiere a la CPU. La palabra de la caché en la dirección índice 000 se reemplaza luego por un rótulo de 02 y datos de 5670.

Fig. 57. Organización de caché de mapeo directo.

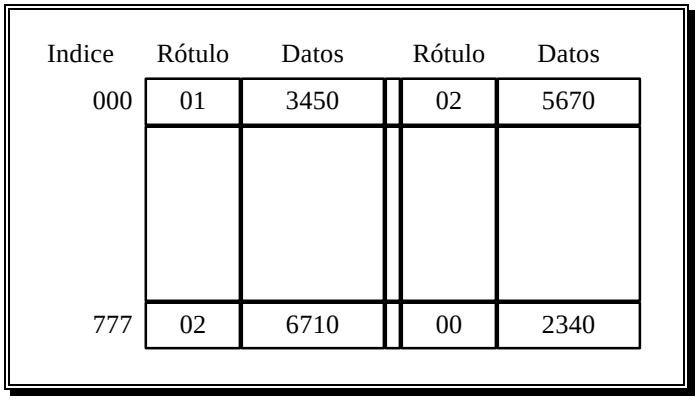


8.3.3 Mapeo Asociativo de Conjunto.

Ya se dijo antes que la desventaja del mapeo directo es que dos palabras con el mismo índice pero con valores de rótulo diferentes no pueden residir en la memoria caché al mismo tiempo. Un tercer tipo de organización llamado mapeo asociativo de conjunto, es una mejora en relación con la organización del mapeo directo en que cada palabra de la caché puede almacenar dos o más palabras de memoria con la misma dirección índice. Cada palabra de datos se almacena junto con su rótulo, y se dice que el número de elementos de datos rótulos en una palabra de la caché forma un conjunto. Un ejemplo de una organización de caché asociativa de conjunto en relación con un tamaño de conjunto de dos se ilustra en la figura 58. Cada dirección índice se refiere a dos palabras de datos y

sus rótulos asociados. Cada rótulo requiere seis bits y cada palabra de datos tiene 12 bits, de modo que la longitud de la palabra es 2(6 + 12) = 36 bits. Una dirección índice de nueve bits puede dar cabida a 512 palabras. Por lo tanto, el tamaño de la memoria caché es de 512 x 36. Esta puede alojar 1024 palabras de memoria principal porque cada palabra de caché contiene dos palabras de datos. En general, una caché asociativa de conjunto con tamaño de conjunto k dará cabida a k palabras de memoria principal en cada palabra de la caché.

Fig. 58. Caché de mapeo asociativo de conjunto con tamaño de conjunto de dos.



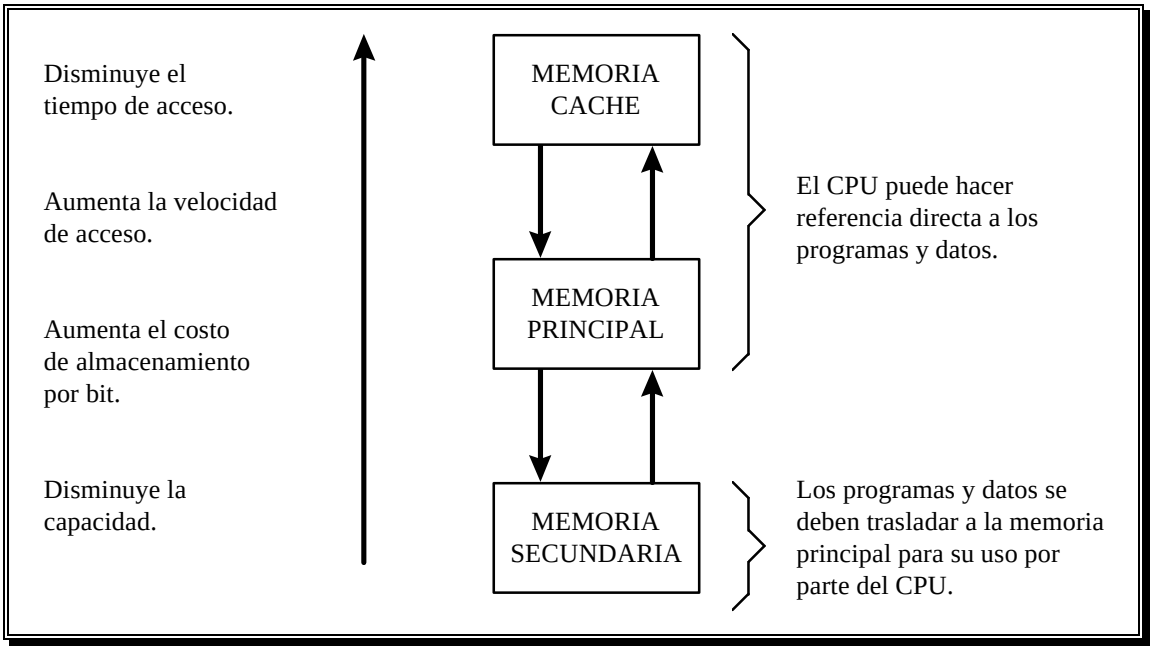
Indice	Rótulo	Datos	Rótulo	Datos
000	01	3450	02	5670
777	02	6710	00	2340

Los números octales que aparecen en la figura 58 se refieren al contenido de la memoria principal que se ilustra en la figura 57. Las palabras almacenadas en las direcciones 01000 y 02000 de la memoria principal se almacenan en la memoria caché de la dirección índice 000. En forma análoga, las palabras en las direcciones 02777 y 00777 se almacenan en la caché en la dirección índice 777. Cuando la CPU genera una solicitud de memoria, el valor índice de la dirección se utiliza para lograr el acceso a la caché. El campo rótulo de la dirección de la CPU se compara después con ambos rótulos en la caché para determinar si hay una coincidencia. La lógica de comparación se ejecuta a través de una búsqueda

asociativa de los rótulos en el conjunto semejante a una búsqueda en la memoria asociativa; de aquí el nombre de asociativa de conjunto. La razón de aciertos mejorará a medida que aumente el tamaño del conjunto porque en la caché pueden residir más palabras con el mismo índice pero rótulos diferentes. Un aumento en el tamaño del conjunto incrementa el número de bits en las palabras de la caché y requiere lógica de comparación más compleja.

En resumen, la organización jerárquica de los tipos de memoria presentes en los computadores digitales y algunas comparaciones entre elementos comunes a ellos se presentan a continuación:

Fig. 59. Organización jerárquica de la memoria.



9. UNIDAD DE CONTROL.

9.1 Definición.

Es el órgano del computador encargado de hacer que el desarrollo de las instrucciones sea posible. Desde ella se controlan y gobiernan todas las operaciones: localiza, analiza y dirige la ejecución de todas las instrucciones de un programa.

La unidad de control recibe a través del bus de datos el código binario de la instrucción en ese momento, el cual lo registra convenientemente. Después el decodificador que es el que se encarga de extraer y analizar el código de la operación de la instrucción, selecciona las posiciones de una memoria ROM llamada memoria de control, que corresponde a dicha instrucción y en las que se encuentra grabado los códigos de las señales que controlan cada uno de los pasos elementales en las que se descompone la instrucción y que recibe el nombre de *microinstrucción*. Por último, el secuenciador que es el que se encarga de sacar y distribuir a los elementos del sistema, emite las correspondientes señales de control de cada microinstrucción, para de esta manera ejecutar ordenadamente la instrucción en curso.

Dentro de la unidad de control se encuentra el contador de programa (PC), encargado de enviar por el bus de direcciones la posición de la memoria donde se encuentra la siguiente instrucción.

9.2 Funciones de la Unidad de Control.

La unidad de control tiene como función básica la ejecución de las siguientes secuencias :

- Tomar de la memoria principal la instrucción apuntada por el contador de programas e incrementar adecuadamente este contador.
- Interpretar o decodificar la instrucción dada.
- Ejecutar la instrucción.

Además de estas tres funciones básicas de lectura, decodificación y ejecución de las instrucciones, la unidad de control se encarga de resolver las situaciones anormales o de conflicto que puedan ocurrir en el computador y de controlar o comunicarse con los periféricos.

Principalmente la información que utiliza la Unidad de Control para realizar esta función es la siguiente:

- Las instrucciones.
- Registros de estado.
- Contador de períodos.
- Señales de entrada y salida.

Adicionalmente la unidad de control emplea la información del contador de programa para producir el secuenciamiento de las instrucciones. Ella misma se encarga de ir incrementando adecuadamente este registro y de cargarlo con nuevos valores.

El *código de operación* indica la operación que debe realizarse, así como los modos de direccionamiento que hay que utilizar. La unidad de control deberá localizar los operandos, mandarlos a la unidad aritmético-lógica y almacenar el resultado.

El *registro de estado* contiene información sobre resultados de operaciones anteriores, así como posibles situaciones especiales tales como desbordamientos, interrupciones, etc. y que exigen la acción de la unidad de control. En términos generales, se puede decir que la información de estado se emplea para hacer rupturas condicionales en la secuencia del programa en curso.

Las *señales de entrada y salida* permiten el diálogo con los periféricos.

La ejecución de cada instrucción requiere realizar una serie de pequeños pasos que llamaremos *operadores elementales* tales como: lectura de operandos, incremento del contador de programa, ejecución de una operación aritmética, etc.

La ejecución de cada una de estas operaciones elementales, requiere la activación de una serie de señales de control que genera la unidad de control a través de su secuenciador. Por lo tanto, el objetivo de la unidad de control será la generación de las señales de control que permitan realizar las distintas operaciones elementales de cada instrucción.

Las *Operaciones Elementales* son una serie ordenada de pequeños pasos de los cuales se descompone la ejecución de cada instrucción. Se clasifican en dos grupos:

- *Operaciones de Transferencia*: En ellas, se comienza seleccionando los elementos que participan (posición de memoria o de registro) uno de los cuales actúa como origen y el otro como destino. Luego se establece un camino de comunicación entre las salidas del origen y las entradas del destino. Finalmente se envía al destino una señal para que se cargue con la información que exista en su entrada.
- *Operaciones de Proceso*: Tienen un planteamiento básico idéntico a las de transferencia; la diferencia fundamental es que la información de origen se hace pasar a través de un operador en su camino hacia el destino. Si la operación es diádica (dos operandos), se parte simultáneamente de dos orígenes u operandos que se unen en el operador que produce el resultado. Si la operación es monádica (un operando) se parte de un solo origen.

9.3 Diseño de la Unidad de Control.

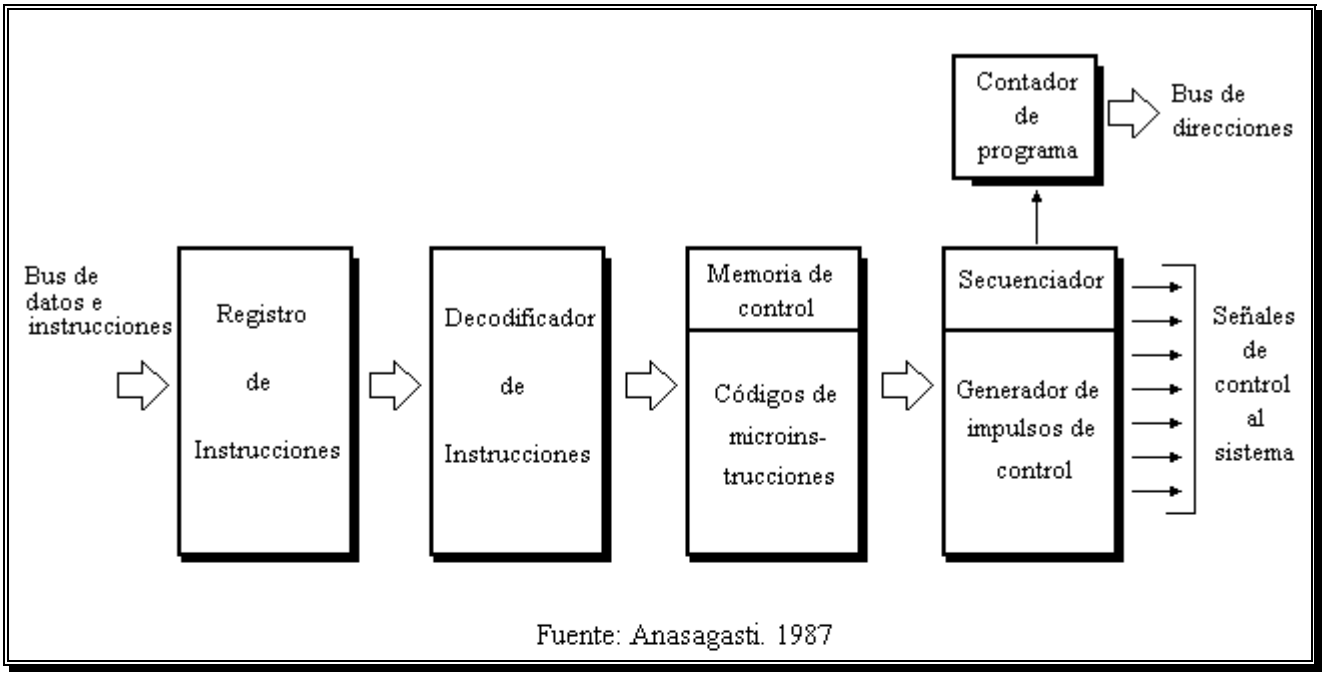
Sabemos que la unidad de control es un traductor que convierte una instrucción de máquina en señales de control, por eso su diseño o construcción exige:

- Haber definido previamente las señales que hay que activar en cada una de las instrucciones de máquina que debe ser capaz de interpretar la unidad de control.
- Toda instrucción se encarga de iniciar la siguiente, o sea , el secuenciamiento de instrucciones se encuentra incluido en cada instrucción.
- La duración de las señales viene determinada por un reloj u oscilador.

Con esto se puede tener una idea de la dimensión y complejidad que reviste el diseño de la Unidad de Control del computador.

A todo ello debe añadirse el análisis y ajuste de los retardos de todos los distintos caminos internos de la unidad de control, lo cual hace que el diseño de este tipo de circuitos no sea sencillo.

Fig. 60. Estructura general de la unidad de control.



Existen dos métodos para construir una unidad de control:

- Mediante Lógica Cableada.
- Mediante Lógica Almacenada.

9.3.1 Unidad de Control en Lógica Cableada.

Los aspectos que la caracterizan son:

1.- Se construye mediante puertas lógicas y se diseña siguiendo algunos de los métodos clásicos de diseño lógico.

2.- Dada la complejidad del circuito su diseño y puesta a punto son actividades laboriosas.

3.- Es muy difícil de modificar puesto que el hacerlo exige un rediseño completo del circuito.

4.- Este tipo de diseño dota al circuito de una gran rapidez consiguiendo una velocidad de funcionamiento mayor que usando lógica almacenada. Por esta razón, las computadoras muy potentes usan lógica cableada.

A pesar de las dificultades, el diseño de la Unidad de Control Cableada está tomando un nuevo auge. Esto se debe a las modernas técnicas de diseño por computador para circuitos de alta escala de integración (VLSI), que resuelven automáticamente la mayor parte de las dificultades del diseño de lógica cableada.

Este tipo de diseño ofrece una buena solución a la carrera de los fabricantes hacia máquinas cada vez más rápidas.

9.3.2 Unidad de Control Microprogramada.

9.3.2.1 Microprogramas (Firmware).

Es un conjunto ordenado de microinstrucciones, que constituyen el cronograma de una instrucción.

Ejecutar un microprograma consiste en ir leyendo cada una de las microinstrucciones que lo forman enviando las señales leídas al computador como señales de control, donde llamaremos microinstrucción a cada palabra que define un período de una instrucción.

Se emplea el término de Firmware para referirse globalmente al conjunto de los microprogramas de una máquina; es un concepto de difícil equivalencia castellana que en algunas ocasiones puede traducirse por microcódigo.

9.3.2.2 Estructura Básica de la Unidad de Control Microprogramada.

1.- Ha de tener una memoria de control con capacidad suficiente para almacenar todos los microprogramas correspondientes a todas las microinstrucciones de máquina.

2.- Ha de tener un procedimiento para hacer corresponder a cada instrucción de máquina su microprograma, o lo que es lo mismo, ha de tener un procedimiento para convertir el código de operación

de la instrucción en la dirección de la memoria de control donde empieza su microprograma.

3.- Ha de tener un mecanismo para ir leyendo las sucesivas microinstrucciones del microprograma en curso y para bifurcar en un nuevo microprograma, cuando termina el que se está ejecutando.

Existen dos alternativas para resolver las condiciones 2 y 3 que denominaremos: Secuenciamiento Explícito y Secuenciamiento Implícito.

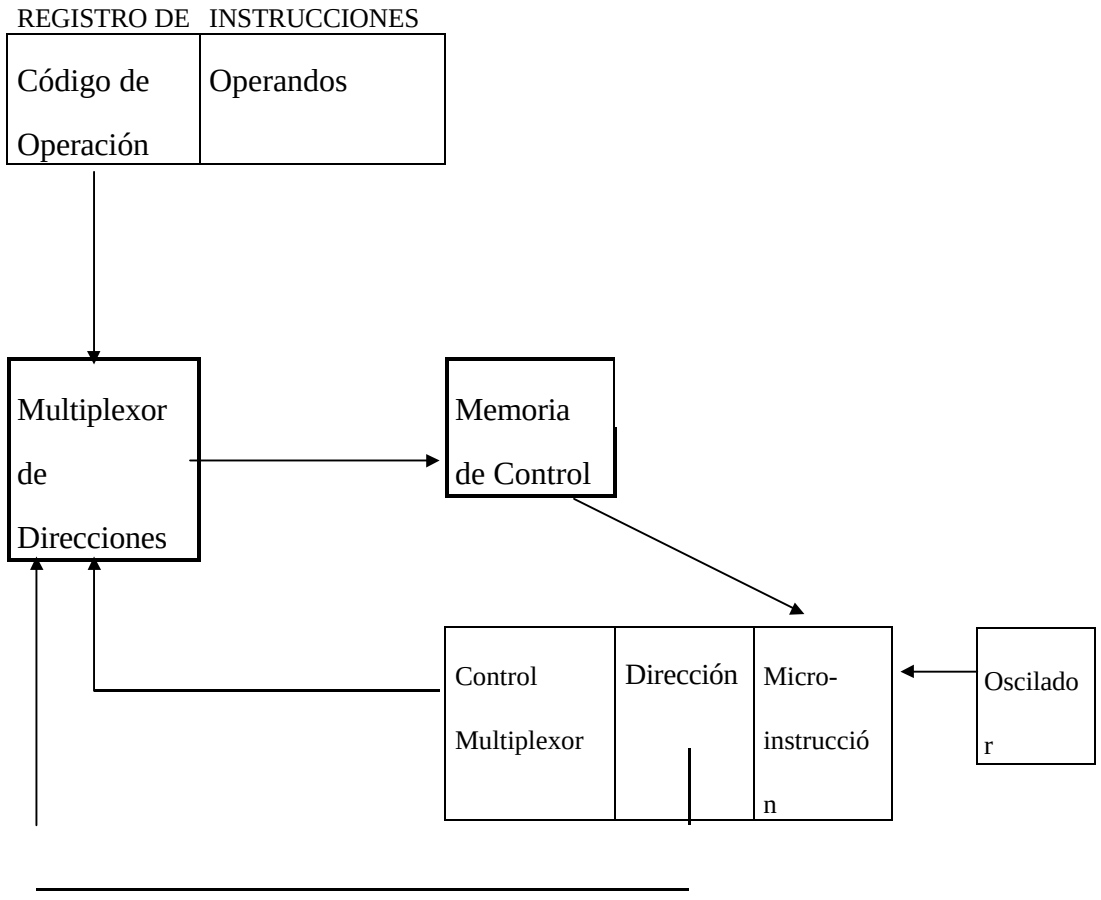
9.3.2.3 Secuenciamiento Explícito.

Consiste en incluir en cada microinstrucción la dirección de la microinstrucción siguiente. En este caso, las microinstrucciones de un microprograma pueden estar desordenadas; no exigen ningún mecanismo de secuenciamiento, puesto que cada microinstrucción suministra la nueva dirección que se tiene que usar en la memoria de control. El encadenamiento con una nueva instrucción se resuelve mediante una señal de control, que toma como dirección de la siguiente microinstrucción el nuevo código de operación. El mayor inconveniente de esta solución es la gran cantidad de memoria empleada en el secuenciamiento de las microinstrucciones (memoria utilizada para guardar las direcciones).

En la figura 61 , se ve como el oscilador genera la señal de "carga" en el registro de microinstrucciones, reestableciendo el período básico de la

máquina. La parte de dirección se realimenta a la memoria de control para iniciar la lectura de la siguiente microinstrucción. El multiplexor de direcciones permite seleccionar entre el código de operación de la nueva instrucción o la dirección de la microinstrucción. Este mecanismo gobernado por una señal de control, se encarga de que la última microinstrucción de un microprograma enganche con la primera microinstrucción del siguiente.

Fig. Nro. 61. Esquema de la U.C. con Secuenciamiento Explícito.



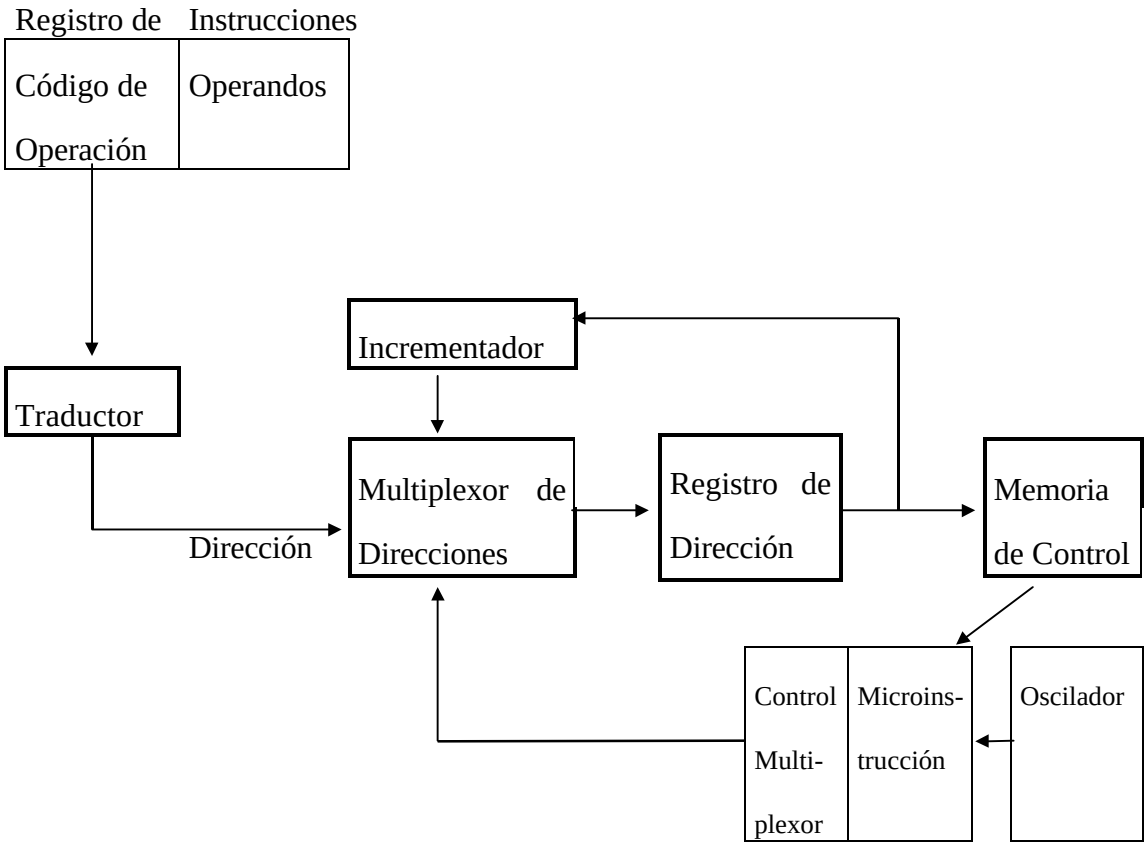
9.4.2.4 Secuenciamiento Implícito.

El secuenciamiento implícito obliga a tener ordenadas las microinstrucciones de cada microprograma, en posiciones consecutivas de la memoria de control. De esta forma ya no es necesario que las microinstrucciones contengan la dirección de la siguiente, con el evidente ahorro de memoria. Como contrapartida, se necesita establecer un mecanismo que resuelva las condiciones 2 y 3 antes mencionadas (ver 9.3.2.2).

La primera se resuelve introduciendo una etapa traductora entre el código de operación y el multiplexor de direcciones de la memoria de control. Este traductor no es más que una memoria ROM.

La segunda condición se resuelve dotando a esta unidad de control de un registro de microdirecciones y de un incrementador. El encadenamiento de microprogramas es my similar al anterior. Una señal de control activa el multiplexor de forma tal que, la última microinstrucción de un microprograma, selecciona como siguiente dirección la información que proviene de la ROM. (Ver gráfico No. 62).

Fig. Nro. 62. Esquema de la U.C. con Secuenciamiento Implícito.



APÉNDICE "A"

SISTEMAS NUMÉRICOS

A.1. Definición de Sistema Numérico.

Es un conjunto de símbolos utilizados para la representación de cantidades, así como las reglas que rigen dicha representación.

Un sistema numérico se caracteriza por dos elementos:

La Base $(B \geq 1)$, que indica el número de símbolos que utiliza y que se caracteriza por ser el coeficiente que determina cuál es el valor de cada símbolo dependiendo de su posición y,

El Alfabeto (A), que constituye los símbolos del sistema. Sus valores varían desde cero (0) hasta Base-1 (B-1).

1.- Ejemplos

- Sistema Binario: Base = 2
  Alfabeto = 0, 1

- Sistema Octal: Base = 8
  Alfabeto = 0, 1, 2, 3, 4, 5, 6, 7

- Sistema Decimal: Base = 10
  Alfabeto = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- Sistema Hexadecimal: Base = 16

  Alfabeto = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

A,	B,	C,	D,	E,	F
↑	↑	↑	↑	↑	↑
10	11	12	12	14	15

Los sistemas numéricos actuales son sistemas posicionales, en los que el valor relativo que presenta cada símbolo o cifra depende de su valor absoluto y de la posición relativa que ocupa dicha cifra con respecto a la coma decimal, íntimamente ligada al valor de la base del sistema de numeración utilizado.

Así, bajo esta filosofía, cualquier número se puede expresar de la siguiente manera:

(1) $$N^{o} = \sum_{i=-d}^{n} D_i \times B^i = X_n B^n + X_{n-1} B^{n-1} + \ \ldots \ + X_0 B^0 + X_{-1} B^{-1} + X_{-2} B^{-2} + \ldots \ + X_{-d} B^{-d}$$

donde:

B: Valor de la base.

i: Posición del dígito respecto a la coma decimal.

d: Nº de dígitos a la derecha de la coma.

n: Nº de dígitos a la izquierda de la coma.

D: Cada uno de los dígitos que componen al número.

Esta fórmula corresponde al Teorema Fundamental de la Numeración, que relaciona una cantidad expresada en cualquier sistema de numeración con la misma cantidad equivalente expresada en el sistema decimal.

Por ejemplo:

$$1992 = 1\times10^3 + 9\times10^2 + 9\times10^1 + 2\times10^0$$
$$= 1000 + 900 + 90 + 2$$

$$3.1415 = 3\times10^0 + 1\times10^{-1} + 4\times10^{-2} + 1\times10^{-3} + 5\times10^{-4}$$
$$= 3 + 0.1 + 0.04 + 0.001 + 0.0005$$

A.2. Conversiones entre Sistemas.

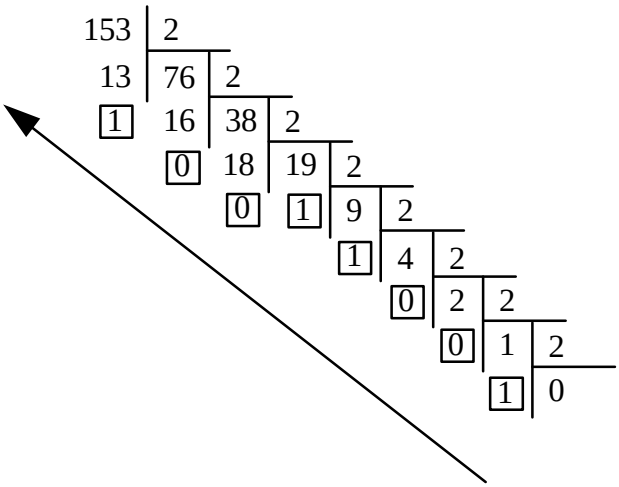
A.2.a. Decimal a Binario.

Un método de conversión consiste en dividir la PARTE ENTERA del número decimal entre 2 y anotar en cada caso el residuo. El número binario en su parte entera se obtiene agrupando todos los residuos obtenidos en orden inverso de aparición, es decir el último residuo se convierte en la cifra más significativa y el primer residuo en la cifra menos significativa. El proceso de división culminará cuando el dividendo es menor que el divisor (2). Por otra parte, para la PARTE FRACCIONARIA, se multiplica el número (en su parte real) por el número 2 y se anota el valor 0 ó 1, que se genera de la primera posición correspondiente a la parte entera. Si esta cifra es igual a 1, se resta 1 del resultado. Si el número que resulta en la posición del primer entero es 0, se anota un 0 y el resultado no se modifica. El número fraccionario binario se obtiene agrupando todos los acarreos generados en la posición de los enteros y tomados en orden creciente de aparición.

Ejemplo:

i) $(153.84)_{10} \rightarrow (?)_2$

Parte Entera:



Parte Fraccionaria:

					Acarreo
0.84	x	2	=	1.68	1
0.68	x	2	=	1.36	1
0.36	x	2	=	0.72	0
0.72	x	2	=	1.44	1
0.44	x	2	=	0.88	0
⋮				⋮	⋮

Resultado: $(153.84)_{10} = (10011001.11010...)$

"Note que la fracción no es exacta."

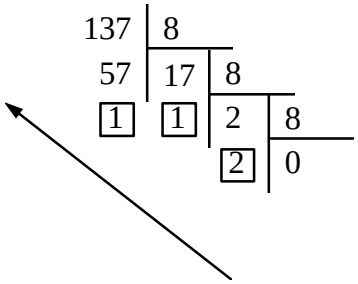
A.2.b. Decimal a Octal.

En este caso, se requieren los mismos pasos que en el caso anterior, pero dividiendo la PARTE ENTERA del número decimal entre 8 y la PARTE FRACCIONAL multiplicándola por 8, que es el número que representa la Base Octal.

Ejemplo:

$(137.67)_{10} \rightarrow (?)_8$

Parte Entera:



Parte Fraccionaria:

					Acarreo
0.67	x	8	=	5.36	5
0.36	x	8	=	2.88	2
0.88	x	8	=	7.04	7
0.04	x	8	=	0.32	0
⋮				⋮	⋮

Resultado: $(137.67)_{10} = (211.5270...)_8$

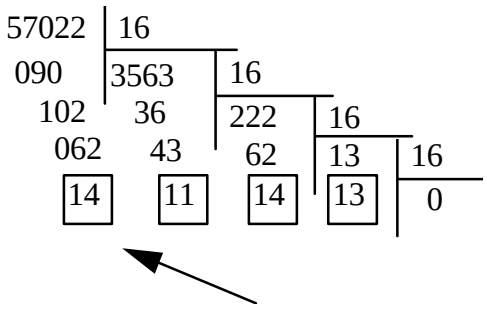
A.2.c. Decimal a Hexadecimal.

El proceso es idéntico a los dos casos anteriores, pero se utiliza como número base el 16.

Ejemplo:

$(57022.27)_{10} \rightarrow (?)_{16}$

Parte Entera:



Parte Fraccionaria:

					Acarreo
0.27	x	16	=	4.32	4
0.32	x	16	=	5.12	5
0.12	x	16	=	1.92	1
0.92	x	16	=	14.72	14
0.72	x	16	=	11.52	11
⋮				⋮	⋮

Resultado: $(57022.27)_{10} = (DEBE.451EB...)_{16}$

A.2.d. Binario a Decimal.

- Método de las Sumas de las potencias de 2.

Este método se basa en la aplicación directa del teorema fundamental de la numeración y se determina usando la relación (1) señalada iniciando este tema.

Así $(110.0101)_2 \rightarrow (?)_{10}$

Aplicando la fórmula tendríamos:

$$N_{10} = \sum_{i=-4}^{2} a_i \bullet (2)^i = 1\times 2^{-4} + 0\times 2^{-3} + 1\times 2^{-2} + 0\times 2^{-1} + 0\times 2^{0} + 1\times 2^{1} + 1\times 2^{2}$$

$$= 0.0625 + 0 + 0.25 + 0 + 0 + 2 + 4 = 0.3125 + 6 = 6.3125$$

Resultado: $(110.0101)_2 = (6.3125)_{10}$

Una forma de hacer la conversión más rápida es:

$$\begin{array}{cccccccccccccccc} 1 & & 1 & & 0 & . & 0 & & 1 & & 0 & & 1 \\ \downarrow & & \downarrow & & \downarrow & \downarrow & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 1\times2^{2} & + & 1\times2^{1} & + & 0\times2^{0} & + & 0\times2^{-1} & + & 1\times2^{-2} & + & 0\times2^{-3} & + & 1\times2^{-4} \end{array}$$

Luego, $(110.0101)_2 = 2^2 + 2^1 + 2^{-2} + 2^{-4} = (6.3125)_{10}$

A.2.e. Binario a Octal.

La conversión de binario a octal, y viceversa simplifica el intercambio de información de los sistemas digitales con el mundo exterior. Acelera además la conversión de los números decimales al sistema binario, ya que la conversión directa de decimal a binario requiere más operaciones que la conversión de decimal a octal y de este a binario. Para ello, se agrupan los dígitos binarios de tres en tres a partir del punto decimal hacia la izquierda y hacia la derecha, sustituyendo cada trío de dígitos binarios por su equivalente dígito actual.

Ejemplo:

$(11100.1011)_2 \rightarrow (?)_8$

$$\begin{array}{cccccc} 011 & 100 & . & 101 & 100 \\ \downarrow & \downarrow & & \downarrow & \downarrow \\ 3 & 4 & . & 5 & 4 \end{array}$$

Así, $(11100.1011)_2 \rightarrow (34.54)_8$

Obsérvese que se añadieron ceros a la izquierda y a la derecha de los bits más y menos significativos respectivamente; para completar grupos de tres dígitos binarios.

A.2.f. Binario a Hexadecimal.

El sistema hexadecimal encuentra uso extensivo en los microprocesadores, donde los números binarios son manejados en grupos de cuatro dígitos. Para convertir un número binario a hexadecimal, se utiliza un mecanismo similar al caso anterior pero agrupando los dígitos binarios de cuatro en cuatro y sustituyendo cada cuarteto por su correspondiente dígito hexadecimal.

Ejemplo:

$(1111010.11011)_2 \rightarrow (?)_{16}$

0111	1010	.	1101	1000
↓	↓		↓	↓
7	A	.	D	8

Respuesta: $(1111010.11011)_2 \rightarrow (7A.D8)_{16}$

A.2.g. Octal a Binario.

Para esta conversión, se expresa cada número octal mediante tres dígitos binarios según la siguiente tabla:

Dígito Octal	Dígitos Binarios
0	000
1	001
2	010

3	011
4	100
5	101
6	110
7	111

Ejemplo:

$(7532.1064)_8 \rightarrow (?)_2$

$$\underbrace{7}_{111}\ \underbrace{5}_{101}\ \underbrace{3}_{011}\ \underbrace{2}_{010}\ \bullet\ \underbrace{1}_{001}\ \underbrace{0}_{000}\ \underbrace{6}_{110}\ \underbrace{4}_{100}$$

Respuesta: $(7532.1064)_8 \rightarrow (111101011010.001000110100)_2$

A.2.h. Hexadecimal a Binario.

Para convertir un número hexadecimal a binario, se sustituye cada dígito hexadecimal por su representación binaria con cuatro dígitos según la siguiente tabla:

Dígito Hexadecimal	Dígitos Binarios
0	0000
1	0001
2	0010

3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Ejemplo:

$(CA.FE)_{16} \rightarrow (?)_2$

C	A	.	F	E
↓	↓		↓	↓
1100	1010	.	1111	1110

Así, $(CA.FE)_{16} \rightarrow (11001010.11111110)_2$

A.2.i. Conversión Octal a Hexadecimal.

Esta conversión realiza un paso intermedio utilizando el sistema binario. Primero se convierte el número octal en binario y éste se pasa a hexadecimal.

Ejemplo:

$(FA.DB)_{16} \rightarrow (?)_8$

F	A	.	D	B
↓	↓		↓	↓
1111	1010	.	1101	1011

Así, $(FA.DB)_{16} \rightarrow (11111010.11011011)_2$

Luego, $(11111010.11011011)_2 \rightarrow (?)_8$

011	111	010	.	110	110	110
↓	↓	↓		↓	↓	↓
3	7	2	.	6	6	6

Así, $(11111010.11011011)_2 \rightarrow (372.666)_8$

Por lo tanto: $(FA.DB)_{16} = (372.666)_8$

A.3. Suma y Resta en el Sistema Binario.

La suma de dos números en el sistema binario sigue las mismas reglas que en el sistema decimal. Los casos que se pueden presentar son los siguientes:

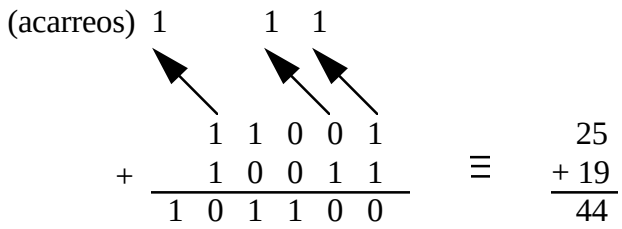
Tabla del 0 Tabla del 1

0 + 0 = 0 1 + 0 = 1

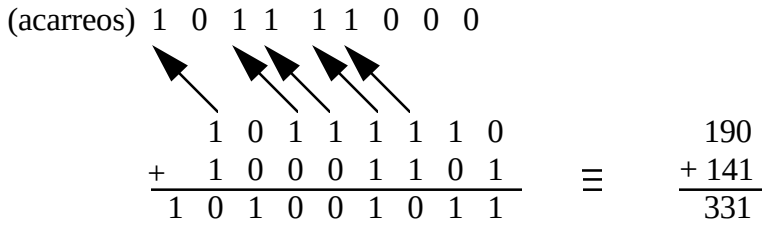
0 + 1 = 1 1 + 1 = 0 (*)

(*) y se lleva 1 (acarreo) a la suma parcial siguiente hacia la izquierda.

Ejemplo: Sumar los números binarios 11001 y 10011



Ejemplo:



Aunque la sustracción puede ser efectuada en binario al igual que en decimal, usualmente los computadores no la efectúan en esa forma. De hecho la mayoría de los computadores no pueden efectuar sustracciones. En su lugar, ellos calculan el resultado de una sustracción por medio de una suma "*especial*". Esta técnica es llamada SUMA DEL COMPLEMENTO. Para poder ver esta instrucción vamos a introducirnos en la manera en cómo podemos representar números negativos.

A.4. Representación de Números Negativos.

Los computadores digitales utilizan principalmente tres métodos para representar números enteros, estos son los siguientes:

a. Representación Signo-Magniud.
b. Complemento de la Base.
c. Complemento de la base Disminuida.

A.4.a. Representación Signo-Magnitud.

La representación de números decimales que usamos a diario es lo que se denomina sistema de signo-magnitud. En este sistema, un número consiste de una magnitud y un símbolo indicativo si la magnitud es positiva o negativa. Así, interpretamos los números decimales +98, -57, +123.5 y -13 en la manera usual. En el sistema numérico decimal, usamos la convención que el signo del número es "+", si ningún símbolo está presente en la magnitud. Hay además dos posibles representaciones del cero, "+0" y "-0", pero ambas tienen el mismo valor.

La representación signo-magnitud puede ser aplicada a los números binarios de una forma bastante fácil mediante el uso de una posición (bit) extra para representar el signo. Tradicionalmente, el bit más significativo (MSB) es usado para representar el signo (0=positivo; 1=negativo) y así, los bits restantes (es decir, los menos significativos) contienen la magnitud. Por ejemplo: usando 8 bits.

$(00101011)_2 = (+43)_{10}$ $(10101011)_2 = (-43)_{10}$

$(00000000)_2 = (+0)_{10}$ $(10000000)_2 = (-0)_{10}$

La representación de números en el sistema signo-magnitud contiene igual número de representación de números positivos y negativos. De esta manera, un número en signo-magnitud de "*n*" bits cae en el rango $-(b^{n-1}-1)$ hasta $+(b^{n-1}-1)$, con dos posibles representaciones para el cero (b=valor de la base).

Para sumar dos números en signo-magnitud, aplicamos las reglas que aprendimos en la escuela. Si los signos son los mismos, sumamos las magnitudes y colocamos al resultado el mismo signo. Si los signos son diferentes, restamos la magnitud más pequeña de la más grande y le colocamos al resultado el signo de la magnitud más grande. Para la sustracción, podemos cambiar el signo del sustraendo y proceder como una adición.

A.4.b. Sistemas Numéricos en Complemento (Rep. Negativa).

Los números negativos en un computador están usualmente representados mediante sistemas numéricos en complemento (Complement Number Systems). Mientras en un sistema signo-magnitud se manipula el número mediante el cambio de signo, un sistema numérico en complemento maneja un número tomando su complemento definido por el sistema. Tomar el complemento es más difícil que cambiar el signo, pero los dos números en un sistema numérico en complemento pueden ser sumados o restados directamente sin necesidad de hacer los chequeos de signo y magnitud requeridos por el sistema numérico de signo-magnitud. Describiremos dos sistemas numéricos en complemento, denominados Complemento de la Base y Complemento de la Base Disminuida.

En cualquier sistema numérico, tratamos siempre con un número fijo de dígitos "n". Asumiremos que la base es "b" y que los números tienen la forma:

$$D = d_{n-1}\, d_{n-2} \ldots d_1\, d_0.,$$

De manera que el punto de la base está a la derecha y el número es un entero. Si un número D es complementado dos veces, el resultado será siempre D.

A.4.b.1. Representación en Complemento de la Base.

En general, en el sistema de complemento de la base, el complemento de un número D de *n*-dígitos se obtiene restándolo de b^n (b^n - número). Si D está entre 1 y b^n-1, esta sustracción resultará en otro número entre 1 y b^n-1. Si D es cero, el resultado de la sustración es b^n, este tiene la forma 1000...00, donde hay un total de n+1 dígitos. Descartamos el bit extra más significativo y obtendremos el resultado de 0. Así, solamente hay una representación del cero en un sistema de complemento de la base.

En el caso del sistema decimal, se denomina complemento a 10 (C a 10) y sería 10^n-número.

Ejemplos:

a. C a 10 de $(52520)_{10}$ es 10^5-52520 = 47480 n=5
b. C a 10 de $(100)_{10}$ es 10^3-100 = 900 n=5
c. C a 10 de $(0.3267)_{10}$ es 1^5-0.3267 = 0.6733 n=0

En este sistema, el rango de valores representables para una precisión de n dígitos es:

$$-(b^{n-1}) \text{ hasta } +(b^{n-1}+1)$$

A.4.b.2. Representación en Complemento a Dos.

Para los números binarios, el complemento de la base se denomina complemento a dos. En este sistema, un número es positivo si el bit más significativo es cero (0) y negativo si es igual a uno (1).

Algunos ejemplos de números de 8 bits y su representación en complemento a dos son los siguientes:

a.

$$\begin{aligned} (119)_{10} &= (01110111)_2 \\ &\quad\ 10001000 \quad \text{(compl. bits)} \\ &+ \quad\ 1 \\ &\quad\ (10001001)_2 = (-199)_{10} \end{aligned}$$

b.

$$\begin{aligned} (0)_{10} &= (00000000)_2 \\ &\quad\ 11111111 \quad \text{(compl. bits)} \\ &+ \quad\ 1 \\ \text{Se descarta} \rightarrow &\ (\mathbf{1}00000000)_2 = (0)_{10} \end{aligned}$$

c.

$$\begin{aligned} (-128)_{10} &= (10000000)_2 \\ &\quad\ 01111111 \quad \text{(compl. bits)} \\ &+ \quad\ 1 \\ &\quad\ (10000000)_2 = (+128)_{10} \end{aligned}$$

A.4.b.3. Adición y Substracción en Complemento a Dos.

La adición en complemento a dos se realiza mediante una acción binaria común (como fue explicada anteriormente) ignorando cualquier acarreo que ocurra después del bit más significativo.

Estableceremos la siguiente convención para los siguientes números:

$A=(A_0, A_1, A_2, \dots , A_n)$ y $A^*=(A_1, \dots , A_n)$

$B=(B_0, B_1, B_2, \dots , B_n)$ y $B^*=(B_1, \dots , B_n)$

$C=(C_{-1}, C_0, C_1, C_2, \dots , C_n)$ y $C^*=(C_1, \dots , C_n)$

Donde A_0, B_0 y C_0 representan los bits para el signo de los números A, B y C respectivamente. Nótese que para el número C se colocó una posición adicional C_{-1} después del bit de signo, ya que en C vamos a considerarlo como el resultado de la operación. Este sistema adicional va a permitir el manejo de un acarreo final si este ocurre al sumar los dígitos más significativos de los números A y B.

Es importante señalar que en estas operaciones se incluye el bit del signo de los números en cuestión.

Pueden aparecer tres casos posibles:

<u>Caso No. 1</u> : *A es positivo y B es positivo.*

			Notación Signo-Magnitud		Notación Complemento a Dos
1)	A	=	+ 0011	≅	00011
	B	=	+ 0100		00100
	C	=	+ 0111		00111
2)	A	=	+1011	≅	01011
	B	=	+1100		01100
	C	=	10111		10111

Acarreo ↑ — ↑ Overflow

<u>Caso No. 2</u> : (a) *A es positivo y B es negativo,* $A^* \geq B^*$.

			Notación Signo-Magnitud		Notación Complemento a Dos
1)	A	=	+ 0111	≅	00111
	B	=	- 0011		11101
	C	=	+ 0100		100100

Acarreo se Descarta ↑

=>	C	=			00100

2)	A	=	+ 0101	≅	00101
	B	=	- 0101		11011
	C	=	+ 0000		100000

Acarreo se Descarta ↑

=>	C	=			00000

(b) *A es positivo, B es negativo,* $A^* < B^*$.

		Notación Signo-Magnitud		Notación Complemento a Dos
A	=	+ 0100	≅	00100
B	=	- 1000		11000
C	=	- 0100		11100

Caso No. 3 : *A es positivo, B es negativo.*

			Notación Signo-Magnitud		Notación Complemento a Dos
1)	A	=	- 0011	≅	11101
	B	=	- 0100		11100
	C	=	- 0111		111001

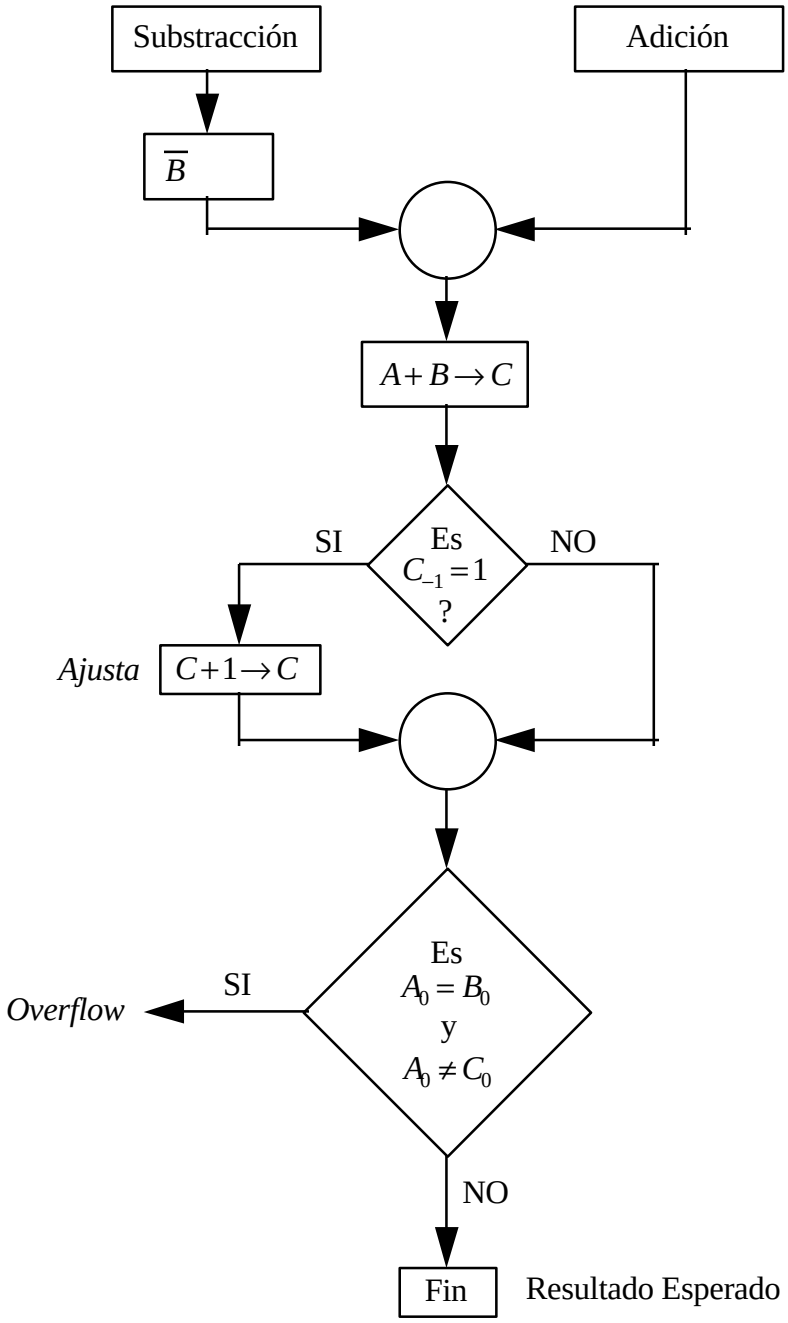
Acarreo se Descarta

=>	C	=	11001

			Notación Signo-Magnitud		Notación Complemento a Dos
2)	A	=	- 1000	≅	11000
	B	=	- 1001		10111
	C	=	10001		101110

Acarreo — Se Descarta — Overflow

En general, el algoritmo para adición y sustracción en complemento a dos es el siguiente:



Como se puede observar, el overflow ocurre cuando en una operación de adición de dos números que tienen el mismo signo, el resultado excede el rango permisible. Lógicamente, la adición de dos números con signos diferentes nunca podrán producir un overflow. Una regla práctica para

*

Acarreo
se Ajusta

			Notación Signo-Magnitud		Notación Complemento a Dos
2)	A	=	+ 0011	≅	00011
	B	=	- 0011		+ 11100
	C	=	+ 0000		11111
					(Cero Negativo)

(b) *A es positivo, B es negativo,* $A^* < B^*$.

		Notación Signo-Magnitud		Notación Complemento a Dos
A	=	+ 0011	≅	00011
B	=	- 1100		10011
C	=	- 1001		10110

<u>Caso No. 3</u> : (a) *A es negativo, B es negativo.*

			Notación Signo-Magnitud		Notación Complemento a Dos
1)	A	=	- 0100	≅	11011
	B	=	- 0111		11000
	C	=	- 1011		110011
					+ 1
	C	=			10100

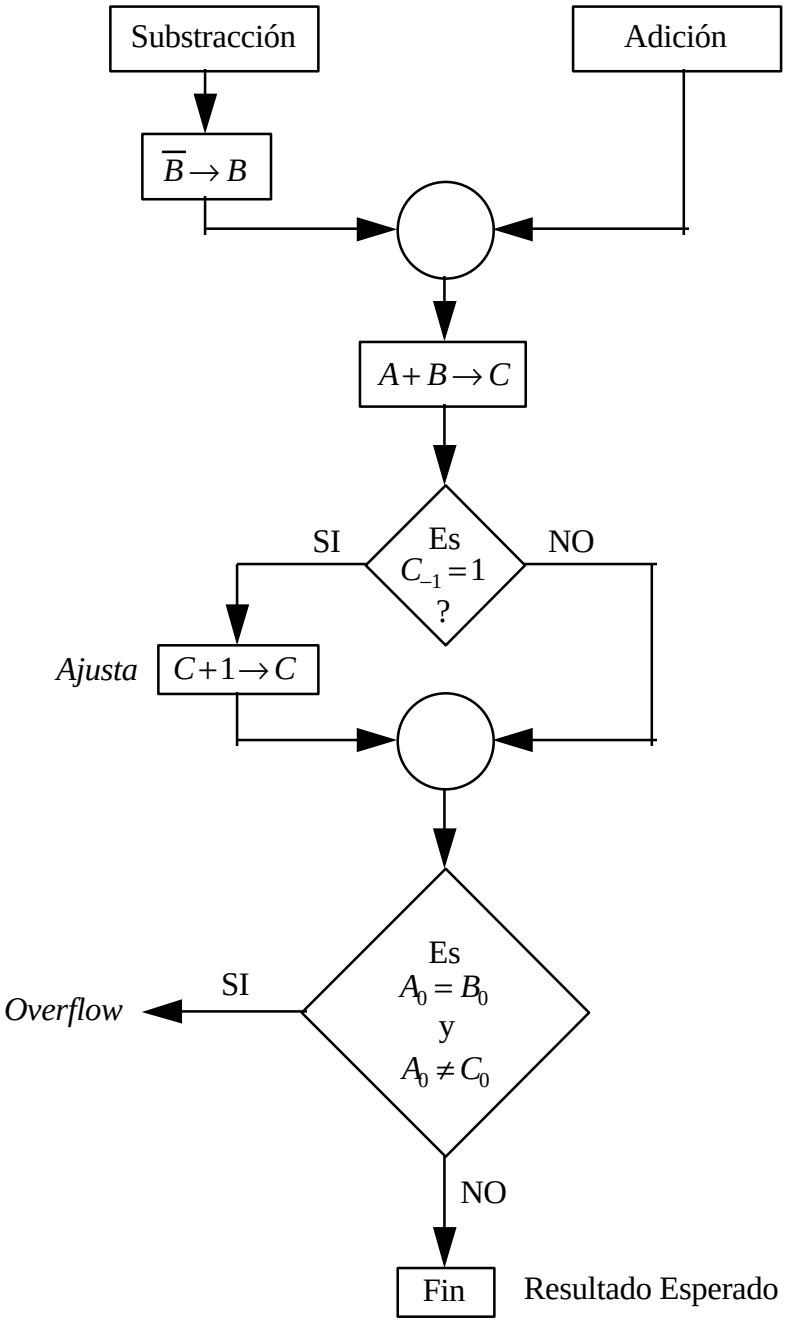
Acarreo se Ajusta

2)	A	=	- 1000	≅	10111
	B	=	- 1000		10111
	C	=	10000		101110
					+ 1
	C	=			01111

Acarreo

Acarreo se Ajusta — Overflow

En general, el algoritmo para la adición y sustracción en complemento a uno es el siguiente:


Substracción
Adición
$\overline{B} \rightarrow B$
$A + B \rightarrow C$
SI
Es
$C_{-1} = 1$
?
NO
Ajusta
$C + 1 \rightarrow C$
Es
$A_0 = B_0$
y
$A_0 \neq C_0$
SI
Overflow
NO
Fin
Resultado Esperado

A.5. Operaciones en el Sistema Hexadecimal.

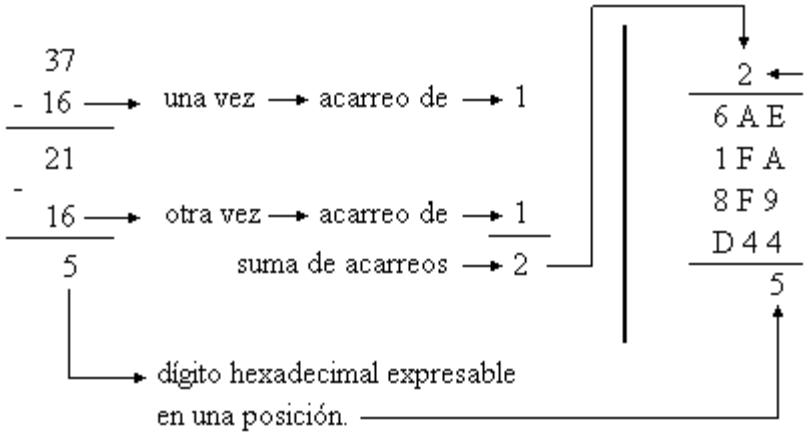
A.5.a. Suma Hexadecimal.

Si se quisiera sumar las cantidades hexadecimales siguientes, se tendría:

```
  6 A E   Sumando la primera posición se tiene:  E16 ——— 1410
+ 1 F A                                        + A16 ——— 1010
  8 F 9                                          916 ——— + 910
  D 4 4                                          416 ———   410
                                                           3710
```

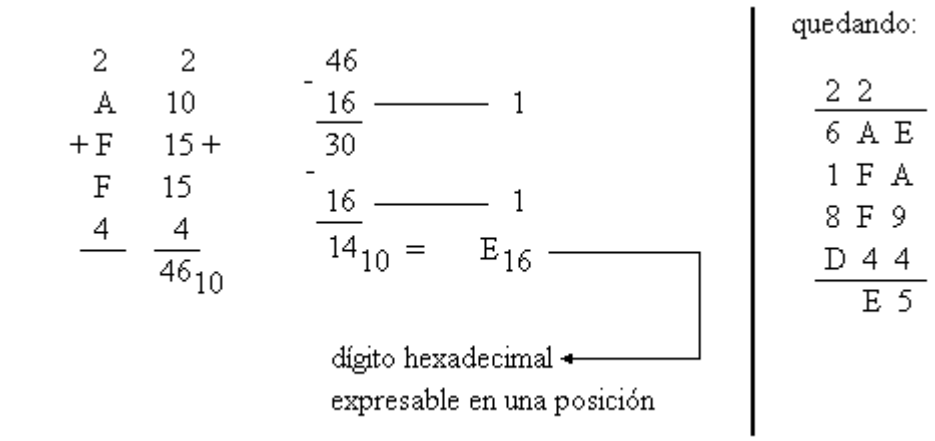
Como el dígito máximo expresable en una posición hexadecimal es F (15), se tiene que calcular, para determinar el valor del acarreo, cuántas veces el número obtenido comprende la base 16. Así:

a)



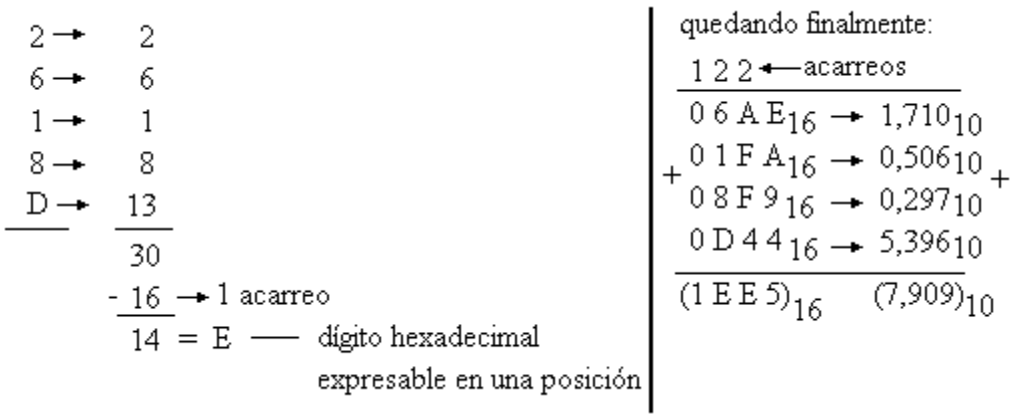
Sumando la segunda posición

b)



Sumando la tercera posición:

c)



Si se quisiera restar las cantidades hexadecimales siguientes:

$$\begin{array}{r} 8\,F\,9\,A \\ - \\ 7\,1\,8\,D \\ \hline \end{array}$$

se vería que, al restar los dígitos de la primera posición, el dígito hexadecimal A es menor que el dígito hexadecimal D. Debe hacerse, por lo tanto, un pedido de uno al dígito del minuendo inmediato a la izquierda. Uno que representa, en el sistema decimal, 10 unidades; en el sistema binario, 1; y en este caso, 16. Por lo tanto:

$$\begin{array}{l} A \rightarrow 10 \longrightarrow 16 + 10 = 26 \\ -D \rightarrow 13 \longrightarrow \quad -13 \\ \hline \qquad\qquad\quad 13_{10} = D_{16} \end{array}$$

Ahora, para aplicar la disminución del uno (1) pedido, se añade 1 al dígito correspondiente del sustraendo, resultando así: 8+1 = 9.

En la segunda posición se resta 9-9 quedando:

$$\begin{array}{r} 8\,F\,9\,A \\ - \\ 7\,1\,8\,D \\ \hline 0\,D \end{array}$$

Las dos últimas posiciones no presentan problema, ya que en la tercera sería $(15)_{10}$ - $(1)_{10}$ = $(14)_{10}$ $\longrightarrow$ E_{16} sin acarreo y finalmente 8-7 = 1_{16}, obteniendo como resultado final:

$$\begin{array}{l} 8\,F\,9\,A \longrightarrow 36{,}762 \\ 7\,1\,8\,D \longrightarrow 29{,}069 \\ \hline 1\,E\,0\,D_{16} \longrightarrow \quad 7{,}693_{10} \end{array}$$

Si cada dígito hexadecimal representa un grupo de 4 bits y un byte es un conjunto de 8 bits, quiere decir que el contenido binario de un byte queda representado por dos dígitos hexadecimales.

A.6. Representación General de Números en Punto Flotante.

Los sistemas de representación que hemos presentados son apropiados para aquellas aplicaciones o programas que tratan con cantidades numéricas enteras.

Por otra parte, existen programas que deben manipular cantidades numéricas que presentan parte fraccional. En estas circunstancias, es conveniente utilizar otro mecanismo que se adopte mejor a los valores a representar. Para ello se cuenta con el sistema de representación de números en punto flotante, donde el valor de un número cualquiera se representa de la forma $M \times b^E$, donde M representa la mantisa y E el exponente.

Así M es una fracción con signo; b es la base del sistema numérico con el cual se trabaja y E es un valor entero con signo que determina la posición real del punto sobre la fracción M.

Algunos ejemplos de números en punto flotante son:

a) $+ 0.015843 \times 10^{-5}$
b) $- 7.12CB2 \times 16^{10}$
c) $- 0.2175527 \times 8^{4}$
d) $+ 1101.0101 \times 2^{-3}$

El computador nos establece un límite sobre el rango y la precisión de los valores numéricos que se pueden representar. Por ejemplo, la arquitectura del computador puede limitarnos a trabajar con 2 dígitos para el exponente y 7 dígitos para la mantisa. Según estas restricciones, los números posibles a representar (en el sistema decimal, en este caso) estarían comprendidos entre:

Valor Máximo = $+ 9999999 \times 10^{+99}$

Valor Mínimo = $- 9999999 \times 10^{-99}$

Es necesario en la mayoría de los casos, transformar los valores a ser representados en el formato fijado por el computador.

A.6.1. Formato de Punto Flotante

Una gran variedad de formatos son usados en diferentes computadores debido a las dificultades alternativas que se pueden tener en relación al tamaño de la mantisa y el exponente, posición del punto decimal en la mantisa y representación de números negativos. Independientemente de las características, un formato típico presenta la siguiente forma:

Signo Mantisa (SM)	Exponente (E)	Mantisa (M)

De esta forma tenemos un sistema versátil ya que podemos:

(i) Obtener una adecuada precisión si incrementamos o disminuimos el número de bits para la mantisa.

(ii) Para representar números positivos y negativos, contamos con un bit que representa el signo de la mantisa.

(iii) Para manipular el rango de valores posibles a representar, podemos aumentar o rebajar la cantidad de bits para el exponente

(iv) Para representar fracciones, podemos utilizar exponentes positivos y negativos.

Se pueden presentar dos situaciones en relación a los números a manipular: los números normalizados y los números desnormalizados.

Los números tienen siempre un dígito distinto de cero como el dígito de mayor peso (el primer dígito a la derecha del punto fraccional) como el caso de los números -0.417 y +0.219. Al normalizar números, estos serán siempre menores que 1. Los números desnormalizados representan el caso contrario. Como por ejemplo: +0.0092 y -0.00017.

Generalmente los procesadores representan los números en punto flotante con sus mantisas normalizadas. Así, las operaciones a realizar sobre estos valores representan ventajas tales como la simplicidad en el diseño y una precisión más grande de la mantisa.

Suponiendo el siguiente formato:

1 bit para el signo de la mantisa (SM),

6 bits para el exponente que se representa en complemento a uno (E),

10 bits para la mantisa (M) y

base de exponenciación (b) = 2,

Veamos algunos números decimales y sus equivalentes normalizados en punto flotante:

a) $-171.875 = -0.171875 \times 10^{3}$

SM	E	M
1	000011	0010110000

a) $0.00533203125 = +0.533203125 \times 10^{-2}$

SM	E	M
0	111101	1000100010

A.6.2. Operaciones en Punto Flotante

Ya que hemos visto las consideraciones que se tienen que tener presente en el manejo de los números en este tipo de representación, veremos cómo se realizan las operaciones básicas que incluyen la adición, sustracción, multiplicación y división.

Aunque las reglas exactas y precisas para cualquiera de estas operaciones dependen del formato del punto flotante y de la organización del computador, presentamos algunos pasos básicos que son utilizados universalmente.

El algoritmo para la adición y la sustracción tiene 3 pasos mínimos:

1.- El punto fraccional de los números debe ser "alineados" de manera que ambos números tengan el mismo exponente. Esto es realizado mediante

la "desnormalización" del operando más pequeño para lograr esta situación.

2.- Las mantisas se suman o se restan según el tipo de operación.

3.- Se normaliza el resultado (si es necesario), ya que la adición puede producir un resultado mayor que 1 y la susstracción puede producir un valor mucho menor a 0.5

La multiplicación y la división son más sencillas y tienen los siguientes aspectos básicos:

1.- Los exponentes se suman (si la operación es multiplicación) o se restan (para el caso de la división).

2.- Las mantisas se multiplican o se dividen.

3.- Se normaliza el resultado.

APÉNDICE "B"

CÓDIGOS DE REPRESENTACIÓN.

B.1 Códigos Decimales.

El sistema de numeración binaria es el más natural para una computadora, pero las personas están acostumbradas al sistema decimal. Una manera de resolver esta diferencia consiste en convertir los números decimales a binarios, realizar todas la operaciones aritméticas en sistema binario y luego convertir los resultados binarios de nuevo a sistema decimal. Este método requiere que se almacenen los números decimales en la computadora en una forma en que se puedan convertir a binario. Como la computadora sólo puede aceptar valores binarios, debemos representar las cifras decimales a través de un código que contenga unos y ceros (1 y 0). También es posible realizar las operaciones aritméticas directamente con números decimales cuando están almacenadas en la computadora en forma codificada.

Un código binario es un grupo de *n* bits que suponen hasta 2^n combinaciones distintas de unos y ceros, donde cada combinación representa un elemento del conjunto que se codifica. Un conjunto de cuatro elementos se puede codificar con dos bits, donde a cada elemento se le asigna una de las siguientes combinaciones de bits: 00, 01, 10, 11. Un conjunto de ocho elementos requiere un código de tres bits y uno de dieciséis elementos requiere un código de cuatro bits. La combinación de bits de un código de *n* bits se determina a partir de la cuenta en sistema binario de 0 a 2^{n-1}. A cada elemento se le deben asignar una combinación de bits binarios única y desde luego dos elementos

no pueden tener el mismo valor; en caso contrario, la asignación del código será ambigua.

Un código binario tendrá algunas combinaciones de bits no asignadas si el número de elementos en el conjunto no es una potencia de dos. Las diez cifras decimales forman dicho conjunto. Un código binario que distingue entre diez elementos debe contener al menos cuatro bits, pero seis de las dieciséis posibles combinaciones se mantendrán no asignadas.

Tabla B.1. Decimal Codificado Binario (BCD)

Símbolo Decimal	Dígito BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Se pueden obtener numerosos códigos binarios diferentes disponiendo cuatro bits en diez combinaciones distintas. El código que se utiliza más comúnmente para representar los dígitos decimales es la asignación binaria directa, según se representa en la tabla B.1. A éste se le llama *Decimal Codificado Binario* y se conoce comúnmente como BCD. Puede haber otros códigos decimales y algunos de ellos se representarán más adelante en esta sección.

La tabla B.1 representa el código de cuatro bits para cada dígito decimal. Un número con *n* dígitos decimales requerirá 4*n* bits en BCD. El decimal 396 se representa en BCD con doce bits como 0011 1001 0110, donde cada grupo de cuatro bits representa un dígito decimal. Un número decimal en BCD es el mismo que su número binario equivalente sólo cuando el número está entre 0 y 9. Un número BCD mayor que 10 se ve diferente de su número binario equivalente aunque ambos contengan unos y ceros (1 y 0). Además, las combinaciones binarias de la 1010 a la 1111 no se utilizan y no tienen significado alguno en el código BCD. Considérese el número decimal 185 y su valor correspondiente en BCD y en binario.

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$$

El valor BCD tiene doce bits pero el número binario equivalente necesita únicamente ocho bits. Es evidente que un número BCD necesita más bits que su valor binario equivalente. Sin embargo, hay una ventaja en el uso de números decimales porque los datos de entrada y salida de una computadora son generados por personas que emplean el sistema decimal.

Es importante entender que los números BCD son números *decimales y no* números binarios, aunque se representen en bits. La única diferencia entre un número decimal y un BCD es que los decimales se escriben con los símbolos 0, 1, 2, ... , 9 y los números BCD utilizan los códigos binarios 0000, 0001, ... , 1001, pero el valor del número es exactamente el mismo. El 10 decimal se representa en BCD con ocho bits como 0001 0000 y el 15 decimal, como 0001 0101. Los valores binarios correspondientes son 1010 1111 y tienen cuatro bits.

B.1.1 Adición BCD.

Considérese la adición o suma de dos dígitos decimales en BCD, junto con un posible acarreo desde un par anterior de dígitos menos significativos. Como cada dígito no es mayor que nueve, la suma no puede ser mayor que 9+9+1=19, donde 1 es el acarreo. Supóngase que se suman los dígitos BCD como si se tratara de números binarios. La suma binaria dará un resultado en el intervalo de 0 a 19. En binario, el intervalo será de 0000 a 10011 pero en BCD debe ser de 0000 a 11001; el primer uno es un acarreo y los cuatro bits que siguen son las sumas de los dígitos BCD. Cuando la suma binaria es igual a, o menor que 1001 (sin acarreo), el dígito BCD correspondiente es correcto. Pero cuando la suma binaria es mayor que o igual a 1010, el resultado es un dígito BCD no válido. La suma del 6 binario $(0110)_2$, a la suma lo convierte en el dígito correcto y produce así mismo un acarreo, según se requiere. Esto se debe a que la diferencia entre un acarreo en la posición 2^4 de la suma binaria y una acarreo decimal es 16-10=6. Considérense las tres adiciones BCD siguientes:

4	0100	4	0100	8	1000
+ 5	+ 0101	+ 8	+ 1000	+ 9	1001
9	1001	12	1100	17	1 0001
			+ 0110		+ 0110
			1 0010		1 0111

En cada caso, los dos dígitos BCD se suman como si fueran dos números binarios. Si la suma binaria es mayor que o igual a 1010, se suma 0110 para obtener la suma de dígitos BCD correcta y un acarreo. En el primer ejemplo, la suma es igual a nueve y es la suma de dígitos BCD correcta. En el segundo ejemplo, la suma binaria produce un dígito BCD no válido. La suma de 0110 produce la suma de dígitos BCD correcta 0010 (2) y un acarreo. En el tercer caso, la suma binaria

produce un acarreo. Esta condición ocurre cuando la suma es mayor que, o igual a 16. Aunque los otros 4 bits son menores que 1001, la suma binaria requiere una corrección porque hay un acarreo. Al sumar 0110, se obtiene la suma de dígitos BCD requerida 0111 (7) y un acarreo BCD.

La suma de dos números BCD sin signo de *n* dígitos sigue el mismo procedimiento. Considérese la adición 184 + 576 = 760 en BCD.

Acarreo BCD	1	1		
	0001	1000	0100	184
	+ 0101	0111	0110	+ 576
Suma Binaria	0111	10000	1010	
Suma de 6		0110	0110	
Suma BCD	0111	0110	0000	760

El primer par de dígitos BCD menos significativos da una suma de dígitos BCD de 0000 y un acarreo para el siguiente par de dígitos. El segundo par de dígitos BCD, más un acarreo previo, dan una suma de dígitos de 0110 y un acarreo para el siguiente par. El tercer par de dígitos, más un acarreo, da una suma binaria 0111 que no requiere corrección.

B.1.1 Aritmética Decimal.

La representación de números decimales con signo en BCD es similar a la representación de números con signo en binario. Podemos emplear el conocido sistema de la magnitud con signo o bien el sistema del complemento con signo. El signo de un número decimal suele representarse con 4 bits para coincidir con el

código de 4 bits de los dígitos decimales. Es costumbre designar un signo más con cuatro ceros y un signo menos con el equivalente BCD de 9 que es 1001.

El sistema de magnitud con signo es difícil de utilizar con computadoras. El sistema de complemento con signo puede ser el complemento a 9's o a 10's, pero este último es el que se utiliza con mayor frecuencia. Para obtener el complemento a 10's de un número BCD, se calcula el complemento a 9's y se suma 1 al dígito menos significativo. El complemento a 9's se calcula restando cada dígito a 9.

Los procedimientos desarrollados para el sistemas del complemento a 2's con signo (ver apéndice A) se aplican también al sistema del complemento a 10's con signo para números decimales. La adición se efectúa sumando todos los dígitos, incluyendo el del signo, y desechando el acarreo final. Claramente, ello supone que todos los números negativos están en forma de complemento a 10's. Considérese la adición (+375) + (-240) = +135, efectuada en el sistema de complemento con signo.

$$\begin{array}{r} 0\ 375 \\ +\ \underline{9\ 760} \\ 0\ 135 \end{array}$$

El 9 de la última posición a la izquierda, del segundo número, representa un signo de menos y 9760 es el complemento a 10's para 0240. Los dos números se suman y se desecha el acarreo final para obtener +135. Desde luego, los números decimales contenidos en la computadora deben estar en BCD, incluyendo los dígitos de los signos. La suma se efectúa con dígitos BCD como se describió antes.

La resta o sustracción de números decimales, sin signo o en el sistema de complemento a 10's con signo, es la misma que en el sistema binario. Calcúlese el complemento a 10's del sustraendo y súmese al minuendo. Muchas computadoras tienen *hardware* especial para efectuar operaciones aritméticas directamente con números decimales en BCD. El usuario de la computadora puede especificar, a través de instrucciones programadas, que la operación aritmética se debe realizar con números decimales sin convertirlos a binarios.

B.2 Otros Códigos Decimales.

Los códigos binarios de dígitos decimales requieren un mínimo de cuatro bits por dígito. Numerosos códigos diferentes se pueden formular disponiendo 4 bits en diez posibles combinaciones diversas. El código BCD y otros tres códigos representativos se presentan en la tabla B.2. Cada código utiliza sólo diez combinaciones de bits de 16 posibles que se pueden disponer con 4 bits. Las seis combinaciones que no se usan en cada caso no tienen significado alguno y deben evitarse.

Los códigos BCD y 2421 son ejemplos de códigos *ponderados*. En un código ponderado, a cada posición de bit se le asigna un factor de ponderación en tal forma que cada dígito se puede evaluar sumando los valores ponderados de todos los unos de la combinación codificada. El código BCD tiene ponderaciones de 8, 4, 2, 1 que corresponden a la potencia de dos valores de cada bit. La asignación de bits 0110, por ejemplo, es interpretada por las ponderaciones como aquella que representa el número 6 decimal puesto que 8x0+4x1+2x1+1x0 = 6. La combinación de bits 1101, cuando se pondera con los dígitos respectivos 2421, da el equivalente decimal 2x1+4x1+2x0+1x1 =

7. Nótese que algunos dígitos pueden codificarse en dos formas posibles en el código 2421. Al 4 decimal se le pueden asignar las combinaciones de bits 0100 o 1010, ya que ambas se suman para hacer un valor ponderado total de 4, pero en la tabla B.2 se eligió a 0100.

Los códigos 2421 y exceso 3 son ejemplos de *códigos autocomplementarios*. Tales códigos tienen la propiedad de que el complemento a 9`s de un número decimal se obtiene directamente cambiando unos por ceros y ceros por unos en el código. Por ejemplo, el decimal 395 se representa en el código de exceso 3 como 0110 1100 1000. El complemento a 9's (604) se representa como 1001 0011 0111 que se obtiene sencillamente completando cada bit de código (como en el complemento a 1's para lo números binarios).

El código de acceso 3 se ha utilizado en algunas computadoras más antiguas debido a su propiedad de autocomplementación. Es un código no ponderado donde cada combinación codificada se obtiene a partir del valor binario correspondiente más 3. Nótese que el código BCD no es autocomplementario.

El código 84-2-1 es un ejemplo de asignación de ponderaciones positivas y negativas a un código decimal. En este caso, la combinación de bits 0110 se interpreta como el 2 decimal y se calcula a partir de 8x0+4x1+(-2)x1+(-1)x0 = 2.

Tabla B.2. Cuatro códigos binarios diferentes para los dígitos decimales.

Dígito Decimal	**BCD 8421**	**2421**	**Exceso-3**	**84-2-1**
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Combinaciones de bits no empleadas	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Es importante entender la diferencia entre la *conversión* de un número decimal en binario y la *codificación* en sistema binario de un número decimal. En cada caso el resultado final es una cadena de bits. Los bits que se obtienen de la conversión son dígitos binarios. Los bits que se obtienen de la codificación son combinaciones de unos y ceros dispuestos de acuerdo con las reglas del código usado. En consecuencia, es en extremo importante entender que cada cadena de bits en una computadora representa a veces un número binario y en otras ocasiones una cadena de bits representa otra información, especificada por un código binario dado.

B.3 Códigos Alfanuméricos.

Muchas aplicaciones de las computadoras digitales requieren manejar datos no solo de números, sino también de letras. Para representar los nombres y otra información pertinente, es necesario formular un código binario para las letras del alfabeto. Además, el mismo código binario debe representar numerales y caracteres especiales con $. Un conjunto de caracteres alfanuméricos es un conjunto de elementos que incluye los 10 dígitos decimales, las 26 letras del alfabeto y un número de caracteres especiales. Dicho conjunto contiene 36 y 64 elementos si solo se contemplan letras mayúsculas, o bien entre 64 y 128 elementos si se incluyen letras mayúsculas y minúsculas. En el primer caso se necesita un código binario de 6 bits y en el segundo se necesita uno de 7 bits.

Los códigos binarios desempeñan un papel importante en las computadoras digitales. Los códigos deben estar en sistema binario porque las computadoras únicamente pueden almacenar unos y ceros. Debe entenderse que los códigos binarios solamente cambian los símbolos, no el significado de los elementos de información que representa. Si inspeccionamos los bits de una computadora al azar, observaremos que la mayor parte del tiempo representan algún tipo de información codificada y no números binarios.

B.3.1 Código de Caracteres ASCII.

El código binario estándar de los caracteres alfanuméricos es ASCII (American Standard Code for Information Interchange). Este emplea siete bits para codificar 128 caracteres, según se ilustra en la tabla B.3. Los siete bits del código se designan por b_1 a b_7 , donde b_7 es el bit más significativo. La letra A, por ejemplo,

se representa en ASCII como 1000001 (columna 100, renglón 0001). El código ASCII contienen 94 caracteres gráficos que se pueden imprimir y 34 caracteres no imprimibles que se utilizan para desempeñar diversas funciones de control. Los caracteres gráficos constan de las 26 letras de la A a la Z, las 26 letras minúsculas, los diez numerales del 0 al 9, y 32 caracteres imprimibles especiales como %, * y $.

Tabla B.3. American Standard Code for Information Interchange (ASCII)

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	´	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	¦
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Tabla B.3. Caracteres de Control (Cont.).

NUL	Nulo	DLE	Escape	de

			enlace de datos
SOH	Inicio de encabezado	DC1	Control del dispositivo 1
STX	Inicio de texto	DC2	Control del dispositivo 2
ETX	Fin de texto	DC3	Control del dispositivo 3
EOT	Fin de transmisión	DC4	Control del dispositivo 4
ENQ	Investigación	NAK	Reconocimien-to negativo
ACK	Reconoci-miento	SYN	Inactividad sincrónica
BEL	Campana	ETB	Fin de bloque de transmisión
BS	Retroceso de espacio	CAN	Cancelación
HT	Tabulador horizontal	EM	Fin del medio
LF	Cambio de línea	SUB	Sustituto
VT	Tabulador vertical	ESC	Escape
FF	Cambio de forma	FS	Separador de archivos
CR	Retorno del carro	GS	Separador de grupos
SO	Fijación de mayúsculas	RS	Separador de registros
SI	Liberación de mayúsculas	US	Separador de unidades
SP	Espacio	DEL	Supresión

Los 34 caracteres de control están designados en la tabla ASCII con los nombres abreviados. Estos se vuelven a citar debajo de la tabla con sus nombres funcionales completos. Los caracteres de control se utilizan para enviar datos y disponer el texto impreso en un formato preestablecido. Existen tres tipos de caracteres de control: caracteres de formato, separadores de información y caracteres de control de comunicación. Los caracteres de formato son caracteres que controlan la presentación de la impresión. Entre estos se encuentran los

conocidos controles de la máquina de escribir como el retroceso de espacios (BS), tabulación horizontal (HT) y el retorno del carro (CR). Los separadores de información se utilizan para separar datos de divisiones como párrafos y páginas. Entre estos se cuentan caracteres como el separador de registros (RS) y el separador de archivos (FS). Los caracteres de control de comunicaciones son útiles durante la transmisión del texto entre terminales remotos. Algunos ejemplos de caracteres de control de comunicación son STX (inicio de texto) y ETX (fin de texto), que se utilizan para enmarcar un mensaje de texto cuando se transmite a través de alambres telefónicos.

B.3.2 Bit de Paridad.

ASCII es un código de siete bits, pero la mayoría de los computadores manipulan una cantidad de ocho bits como una sola unidad llamada byte. Por lo tanto, los caracteres ASCII se almacenan más frecuentemente uno por byte, donde el bit más significativo se inicializa en cero. El bit extra se utiliza algunas veces con fines específicos, dependiendo de la aplicación. Por ejemplo, algunas impresoras reconocen otros 128 caracteres ASCII de 8 bits donde el bit más significativo se inicializa en uno. Estos caracteres permiten a la impresora producir símbolos adicionales como el alfabeto griego o la fuente de tipo itálico.

Para la comunicación de datos, a veces se utiliza un octavo bit para indicar la paridad del carácter. Un *bit de paridad* es un bit extra incluído para hacer que el número total de unos sea par o impar. Considérense los dos caracteres siguientes y su paridad par o impar:

			Con paridad par	**Con paridad impar**
ASCII A	=	1000001	01000001	11000001
ASCII T	=	1010100	11010100	01010100

En cada caso, se utiliza el bit extra en la última posición a la izquierda del código con el objeto de producir un número par de unos en el caracter para que haya paridad par o bien un número impar de unos en el caracter para que se presente paridad impar. En general, se adopta una paridad o la otra, donde la paridad par es la común.

El bit de paridad es útil para detectar errores durante la transmisión de información de un sitio a otro. Esto se lleva a cabo de la manera siguiente:

- Se genera un bit de paridad par en la parte emisora de cada caracter.
- Los caracteres de ocho bits que incluyen bit de paridad se transmiten a su destino.
- Después se verifica la paridad de cada caracter en la parte receptora.
- Si la paridad del caracter recibido no es par, esto quiere decir que al menos un bit ha cambiado de valor durante la transmisión.

Este método detecta cualquier número impar de errores en cada caracter que se transmite. Un número par de errores no es detectado. Se necesitan otros códigos de detección de errores para manejar un número par de errores.

Lo que se hace después de que se detecta un error depende de la aplicación en particular. Una posibilidad es solicitar la retransmisión del mensaje con la suposición de que el error fue aleatorio y que no volverá a ocurrir. Por lo tanto, si el receptor detecta un error de paridad, devuelve un control de caracter NAK (reconocimiento negativo) que consta de ocho bits de paridad par 10010101 (ver tabla B.3.). Si no se detecta ningún error, el receptor devuelve un caracter de control ACK (reconocimiento) 00000110. La parte emisora responderá a un NAK transmitiendo el mensaje una vez más hasta que se reciba la paridad correcta. Si después de varios intentos la transmisión sigue estando en error, se puede enviar un mensaje al operador para verificar la existencia de fallas en la trayectoria de la transmisión.

B.3.3 Otros Códigos Alfanuméricos.

Otro código alfanumérico que se utiliza en equipos IBM es EBCDIC (*Extended BCD Interchange Code*). Este emplea 8 bits para cada carácter y un noveno bit para la paridad, si éste se utiliza. EBCDIC tiene los mismos símbolos de caracteres que ASCII, pero la asignación de bits para caracteres es diferente. Como el nombre lo indica, el código binario de las letras y los numerales son una extensión del código BCD. Esto significa que los primeros 4 bits y los últimos 4 del código varían de 0000 a 1001, como en BCD.

Cuando se usan caracteres en el interior de una computadora para el procesamiento de datos (no para fines de transmisión), a veces conviene utilizar un código de 6 bits para representar 64 caracteres. Un código de 6 bits puede

especificar 64 caracteres que constan de las 26 letras mayúsculas, los 10 numerales y hasta 28 caracteres especiales. Este conjunto de caracteres suele ser suficiente para procesar datos. El uso de menos bits para codificar caracteres tiene la ventaja de reducir el espacio que se necesita para almacenar grandes cantidades de datos alfanuméricos.

Un código desarrollado en las etapas iniciales de la transmisión de datos es el código Baudot de 5 bits. Aunque 5 bits pueden especificar sólo 32 caracteres, el código Baudot representa 58 caracteres utilizando dos métodos de operación. En el modo llamado letras, los 5 bits codifican 26 letras del alfabeto. En el modo llamado cifras, los 5 bits codifican los numerales y otros caracteres. Existen dos caracteres especiales que son conocidos por ambos modos de operación y se emplean para cambiar de un modo al otro. El carácter de *cambio de letras* coloca a la estación receptora en el modo de letras, después de los cuales todos los códigos de caracteres subsiguientes se interpretan como letras. El carácter *de cambio de cifras* coloca al sistema en el modo de cifras. La operación de cambio es análoga a la de una máquina de escribir con una tecla para mayúsculas.

Haciendo un poco de historia, cuando se transfería información alfanumérica a la computadora mediante el uso de tarjetas perforadas, los caracteres alfanuméricos se codificaban con 12 bits. Los programas y datos del pasado se elaboran en tarjetas perforadas utilizando el código Hollerith. Una tarjeta perforada consta de 80 columnas y 12 renglones. Cada columna representa un carácter alfanumérico de 12 bits con orificios perforados en los renglones apropiados. Un orificio se percibe como un 1 y la ausencia de un orificio se percibe como un 0. Los

12 renglones se marcan comenzando desde la parte superior, como 12, 11, 0, 1, 2, ... , 9 perforaciones. Las tres primeras se conocen como perforaciones de zona y las nueve últimas se denominan perforaciones numéricas. Los dígitos decimales se representan por un solo orificio en una perforación numérica. Las letras del alfabeto se representan por dos orificios en una columna, uno en la parte de la zona y el otro en la perforación numérica. Los caracteres especiales se representan con uno, dos o tres orificios en una columna. El código de la tarjeta de 12 bits es ineficiente en cuanto al uso que hace de los bits. En consecuencia, la mayoría de las computadoras que reciben entrada de una lectora de 30 tarjetas convierten el código de tarjeta de 12 bits de entrada en un código de 6 bits interno para conservar bits de almacenamiento.

GLOSARIO DE TÉRMINOS.

- **Acarreo:** Unidad adicional que se produce al sumar dos dígitos cuyo resultado excede a la base.

- **Acceso:** Proceso de almacenar, modificar o recuperar información en algún archivo. Es posible tener acceso a las memorias, a las impresoras y a otros elementos de la computadora. Dentro de este término están los conceptos de "Clave de Acceso" (clave o código del usuario para ponerse en contacto, con la computadora) y "Tiempo de Acceso" (el tiempo que necesita la memoria o el disco para dar su respuesta).

- **Acceso Aleatorio:** Una memoria es de acceso aleatorio cuando el tiempo de acceso a cualquier palabra de información almacenada es de valor constante. Este es el sistema mediante el cual es posible lograr acceso a las informaciones que se encuentran en el dispositivo de almacenamiento.

- **Acceso Directo:** Caracteriza a una memoria en la que puede leerse o escribirse con el mismo tiempo de acceso, cualquiera que sea la dirección de la célula direccionada.

- **Acceso Secuencial:** Es aquel donde los datos han de ser accedidos uno detrás del otro.

- **ASCII:** Son las siglas de *American Standard Code for Information Interchange*. Representa los caracteres (alfabéticos, simbólicos, etc.) en la memoria del computador.

- **Banco de Registros:** Elemento que bajo el control del CPU sirve para conservar datos temporalmente.
- **Bifurcación:** Operación por medio de la cual se escoge dependiendo de una determinada condición, una de dos alternativas (Salto).

- **Binario:** Sistema de numeración en base 2 que consta de los dígitos 0 y 1, usado comúnmente en computadora.

- **Bipolar:** Tecnología de diseño de circuitos microelectrónicos que utiliza transistores los cuales se utilizan para la interrupción y la amplificación.

- **Bit (Dígito Binario):** El componente más pequeño de la clave binaria. El nombre se deriva de la contracción de Binary y Digit. Un bit es un uno (1) o un cero (0). Los bits se usan universalmente en los sistemas electrónicos para codificar la información, las instrucciones y los datos. Los bits se agrupan en unidades que posteriormente son procesadas y almacenadas por la computadora. Según sea su longitud, estos grupos son denominados como caracteres, bytes o palabras.

- **Bloque:** Conjunto de registros que se manipulan de forma unitaria en operaciones de entrada y salida.

- **Bus:** Conductor eléctrico que proporciona una comunicación entre dos o más dispositivos, tales como el CPU, memoria o periféricos.

- **Bus de Control:** Ruta o canal entre dispositivos de hardware, que permite la sincronización de actividades entre diversos elementos del computador.

- **Bus de Datos:** Ruta o canal entre dispositivos de hardware, que permite la transferencia de instrucciones y datos del CPU con el exterior.

- **Bus de Direcciones:** Ruta o canal entre dispositivos del hardware, que permite localizar los datos en la memoria.

- **Byte:** Unidad de almacenamiento que equivale a ocho (8) bits. Se usa universalmente para representar un carácter. Se le conoce también como octeto.

- **Canal de E/S:** Es la unidad encargada de realizar las transacciones de información entre la unidad de control y los periféricos. Su utilidad radica en que descargan a la unidad central de proceso del control directo de la entrada y salida de datos.

- **Capacidad:** Es el número de información que puede contener la memoria.

- **Celda:** Unidad elemental de almacenamiento. Circuito o registro que almacena una cantidad de información que se maneja como un todo.

- **Chip:** Pastilla o circuito integrado: Es una plaquita rectangular, de una sustancia llamada silicio, que llevan impresos una cantidad de transistores y componentes electrónicos microscópicos (circuitos electrónicos miniaturizados).

- **Ciclo:** Intervalo entre los flancos de subida y bajada de dos pulsos consecutivos de reloj.

- **Ciclo de Refresco:** Ciclo en el cual se mantiene la carga de los bytes por un dispositivo llamado “Refresh” encargado de dicho proceso. El refrescamiento se realiza periódicamente.

- **Cilíndro:** Conjunto de las pistas de un disco que tienen un mismo radio.

- **Circuito Lógico:** Circuito que procesa señales lógicas.

- **Código de Operación:** Código que especifica la operación que se ha de ejecutar en una instrucción.

- **Código de Paridad:** Es el que añade a cada dato un bit adicional. Este bit llamado de paridad será (0) si el número de unos en el dato es par, y será (1) en caso contrario.

- **Complemento:** Noción que se utiliza para obtener la representación negativa de un número dado. Por ejemplo, el complemento de cero es uno y de uno es cero.

- **Complemento a Dos:** Forma de representar un número negativo en el sistema de numeración binario.

- **Complemento a Uno:** Complemento restringido en el sistema binario.

- **Compuertas Lógicas:** Es un elemento cuya salida depende de las combinaciones de las entradas. Ejemplo AND, OR, XOR, NAND, etc.

- **Contador de Programa (PC):** Registro de la unidad de control de un computador utilizado como contador para mantener al día la dirección de la próxima instrucción por ejecutar.

- **CPU (Unidad Central de Procesamiento,** *Central Processing Unit*). El CPU se encuentra formado por la unidad de control, la unidad aritmético-lógica y algunos registros. También se le denomina *procesador*.

- **Datos:** Unidades de información que pueden ser definida con exactitud. En un sentido más general, los datos son las materias primas que una vez procesadas, dan lugar a la información total.

- **Decodificador de Direcciones:** Selecciona la celda cuya dirección ingresa en la unidad de memoria a través de las líneas de direccionamiento. Las líneas de direccionamiento proceden de la CPU a través del bus de direcciones.

- **Desbordamiento:** Parte del resultado de una operación aritmética que sobrepasa las posibilidades de resultado. Corresponde generalmente a la pérdida de los dígitos más significativos.

- **Desplazamiento:** Movimiento de una información digital un cierto número de posiciones a la derecha o a la izquierda. Si la información tiene un número limitado de posiciones se pierde la información salvo si el desplazamiento es circular.

- **Desplazamiento a la Derecha:** Se corren los bits más hacia la derecha y las posiciones se rellenan de ceros. El bit que sale por la derecha se almacena en el registro especial de estados, el bit de acarreo. Con estos tipos de desplazamiento existe pérdida de información.

- **Desplazamiento a la Izquierda:** En este tipo de desplazamiento se corre el bit más a la izquierda y en la posición vacante se coloca un 0. El bit que sale por la izquierda se almacena en un bit de registro especial de estados, el bit de acarreo.

- **Diádica:** Operación que involucra dos operandos.

- **Dirección:** Información que identifica biunívocamente un registro, una posición de memoria, una unidad periférica.

- **DOS (Disk Operating System):** Sistema operativo del disco. Un conjunto de programas que se cargan desde el diskette o disco duro al encender el computador y posibilitan la utilización de los distintos dispositivos que forman una máquina.

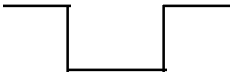
- **Drive:** Mecanismo empleado en los almacenamientos periféricos para soportar y conducir dispositivos tales como las cintas o los discos magnéticos.

- **Fetch:** Etapa de búsqueda de una instrucción en la memoria principal y traslado al microprocesador.

- **Firmware:** Memoria Fija o Soporte Lógico Inalterable: Software permanente, que debe permanecer aún en el caso de una interrupción en el suministro de energía eléctrica.

- **Flanco:** En la representación gráfica de un ciclo: lado ascendente o descendente, o sea, la oscilación del ciclo, ejemplo:

  Flancos



- **Flip-Flop (Biestable) :** Componente lógico electrónico que tienen una salida que sólo puede tomar los valores: falso o verdadero.

- **Instrucción:** Palabra reconocible para la unidad de control, de la que se extrae la información sobre qué función se ejecutará y qué elementos se utilizarán.

- **Interrupción:** Suspensión del proceso normal de operación que puede, sin embargo, volver a iniciarse más tarde en el mismo lugar en el que se había producido la parada, en este periodo es posible realizar cualquier otra acción.

- **Kbyte:** Es un Kilobyte, conjunto de 1024 byte.

- **Lenguaje:** Es un conjunto de claves o códigos mediante el cual el hombre comunica al computador el conjunto de acciones que debe realizar.

- **LSI (Large Scale Integration):** Alta Escala de Integración, se refiere a los microcircuitos formados por un gran número de componentes electrónicos.
- **Mbyte:** Es un Megabyte, conjunto de 1024 Kbyte.

- **Memoria de Semiconductor:** Es el tipo de memoria empleada como memoria principal, basada en biestables electrónicos como puntos de memoria.

- **Memoria MOS:** Chip de memoria o semiconductor de óxido de metal.

- **Memoria Principal:** Es la memoria donde se almacenan los programas, los datos y los resultados implicados en la ejecución de un proceso.

- **Memoria Volátil:** Sistema de almacenamiento que, si pierde la energía eléctrica pierde asimismo su contenido.

- **Microinstrucción:** Instrucción elemental que conjuntamente con otras compone una instrucción.

- **Microlenguaje:** Lenguaje de microprogramación.

- **Microprograma:** Programa escrito en microlenguaje, correspondiente a la ejecución de una instrucción de lenguaje de máquina en un computador microprogramado.

- **Microprogramación:** Designa a una técnica de realización del secuenciador central de un computador, en la que cada instrucción es interpretada y ejecutada bajo el control de un programa llamado microprograma.

- **Monádica:** Operación que involucra un solo operando tal como la negación.

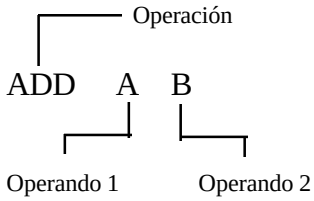
- **MOSFET (Metal Oxide Semiconductor Field Effect Transistor):** Transistor de efecto de campo de semiconductor a base de metal y óxido transistor empleado en circuitos integrados MOS.

- **Multiplexor:** Dispositivo que permite distribuir la información de una vía entre varias vías, o por el contrario, concentrar informaciones provenientes de varias vías en una sola.

- **Operación:** Es la ejecución de un cálculo determinado sobre una o varias entidades con el objeto de hallar un resultado.

- **Operador:** Es un circuito electrónico capaz de realizar una operación aritmética (suma, resta, etc.) o lógica (and, or, etc.).

- **Operando:** Dato sobre el cual se efectúa una operación, ejemplo.



- **Palabra:** Toda cadena de bits utilizada para representar un único ente de información.

- **Periféricos:** Conjunto de unidades constituidas de un sistema de proceso de datos que funcionan bajo el control de la Unidad Central de Proceso, generalmente están conectados a un canal y este a su vez al CPU.

- **Procesador:** ver CPU.

- **Procesamiento en Serie:** Se procesa bit a bit en forma secuencial.

- **Procesamiento en Paralelo:** Se procesa simultáneamente todos los bits de información.

- **Procesamiento en Paralelo-Serie:** Se procesa por grupos, los grupos se procesan en serie y los bits que componen los grupos en paralelo.

- **Programas:** Secuencia o grupo de instrucciones cuya finalidad es la de señalar a la computadora el modo de realizar una determinada función. Todo programa está formado por instrucciones, variables y constantes. Por lo general, los programas son diseñados de uno a tres niveles: binario, lenguaje ensamblador o lenguaje de alto nivel.

- **Puntos de Memoria:** Son los elementos biestables que retienen la información almacenada, hasta que esta es modificada por una nueva operación o hasta que se desconecta la alimentación.

- **Registro de Condición:** Son unos biestables que indican alguna condición especial surgida al ejecutar una operación, y que pueden ser utilizadas por el programa para tomar determinadas decisiones. Son alterados por la unidad aritmético-lógica.

- **Registro de Datos de Memoria (MDR o MBR):** Registro que permite intercambiar la información entre el procesador y la memoria principal.

- **Registro de Dirección en Memoria (MAR):** Registro que permite direccionar la memoria principal de un computador.

- **Registro de Instrucciones (IR):** Registro del procesador de un computador que recibe y conserva la instrucción en un curso durante su interpretación (decodoficación) y ejecución.

- **Registro de Tipo General:** Se utiliza para almacenar instrucciones y datos de manera temporal. Pueden ser accedidos tanto por el programador como por el procesador.

- **Registro de Uso Específico:** Son aquellos que tienen una función básica dentro del procesador y solo pueden ser accesados por éste.

- **Reloj u Oscilador:** Dispositivo que genera los impulsos eléctricos que permiten sincronizar las operaciones de la computadora.

- **Salto:** Instrucción que permite romper el encadenamiento de las instrucciones colocadas en posiciones sucesivas de memoria y seleccionar otra parte del programa.

- **Secuenciador:** Circuito integrado cuya función es producir la secuencia de las direcciones de los microprogramas.

- **Sincronización:** Son los medios que utilizan una computadora para coordinar sus actividades con aquellos dispositivos externos que estén conectados.

- **Sistema:** Es un conjunto de elementos que interactúan entre sí, con el fin de alcanzar determinado objetivo.

- **Subciclos:** Parte o división de un ciclo, por ejemplo, cada flanco del ciclo (subida o bajada) representa un subciclo.

- **Sumador:** Es un circuito combinacional capaz de sumar dos dígitos binarios más el posible acarreo de la etapa anterior, produciendo el dígito de suma y el de acarreo de la etapa siguiente.

- **Tabla de Verdad:** Tabla que define a una función o circuito por su valor para cada posible combinación de las variables implicadas.

- **Tiempo de Acceso:** Es el intervalo de tiempo transcurrido desde que se solicita un dato a la unidad de memoria hasta que ésta lo entrega a la unidad de control.

- **UCP:** ver CPU.

- **Unidad de Control:** Unidad de un computador encargada de la búsqueda y la interpretación de las instrucciones, del cálculo de dirección y de la búsqueda de los operandos y del posicionamiento de los circuitos para ejecutar las instrucciones.

- **Volátil:** Sistema de memoria en el cual los datos se pierden cuando se desconecta la alimentación.

BIBLIOGRAFIA.

- ANASAGASTI, PEDRO. 1988. Fundamentos de los Computadores. 1ra. Edición. Editorial Paraninfo. Madrid.
- BLAS, JOSÉ M. DE. 1989. IBM PS/2: Características y Configuraciones. 1ra. Edición. Editorial Anaya Multimedia. España.
- BLISSMER, ROBERT H. 1991. Introducing Computers. 1ra. Edición. John Wiley & Sons. U.S.A.
- BOYCE, JIM. 1994. Conserve Viva su PC. 1ra. Edición. Prentice-Hall. México.
- BYTE . MAY 1996. McGraw-Hill Company. U.S.A.
- BYTE VENEZUELA. Julio 1997. Organización Transamérica C.A. Caracas.
- FIGUERA, JUPITER. 1990. Circuitos Lógicos de Computación. 1ra. Edición. EDUVEN. Caracas.
- HAMACHER, V. CARL, VRANESIC, ZVONKO G. y SAFWAT ZAKY. 1986. Organización de Computadoras. 2da. Edición. McGraw-Hill. México.
- MANDELL, STEVEN L. 1992. Computers and Information Processing. 6ta. Edición. West Publishing Company. St. Paul, MN.
- MANO, M. MORRIS. 1991. Ingeniería Computacional: Diseño del Hardware. 1ra. Edición. Prentice-Hall. México.
- MEINADIER, JEAN P. 1975. Estructura y Funcionamiento de los Computadores Digitales. 1ra. Edición. Editorial AC. Madrid.
- PÁEZ M., GERARD. 1993. Arquitectura y Organización del Computador. 1ra. Edición. Editorial Senda Sol. Mérida.
- PC MAGAZINE. Mayo 1997. Ziff Comunications Company. México.
- PC MAGAZINE. Julio 1997. Ziff Comunications Company. México.
- PC MAGAZINE. Agosto 1997. Ziff Comunications Company. México.

- SILBERSCHATZ, ABRAHAM, JAMES L. PETERSON y PETER B. GALVIN. 1994. Sistemas Operativos. 3a. Edición. Addison-Wesley Iberoamericana. Wilmington. U.S.A.
- TANENBAUM, ANDREW. 1985. Organización de Computadoras: Un Enfoque Estructurado. 2da. Edición. Prentice-Hall. México.
- TAUB HERBERT. 1983. Circuitos Digitales y Microprocesadores. 1ra. Edición. McGraw-Hill. U.S.A.
- TORRES P., MANUEL. 1989. Microprocesadores y Microcontroladores Aplicados a la Industria. 1ra. Edición. Editorial Paraninfo. Madrid.
- VELÁSQUEZ G., NELLY. 1989. Computadores & Microcomputadores: Arquitectura y Microprogramación. 1ra. Edición. U.C.L.A. Barquisimeto.
- WHITE, RON. 1996. Cómo Funcionan Las Computadoras. 2da. Edición. Prentice-Hall. U.S.A.