ARQUITECTURA DE COMPUTADORAS

ORGANIZACIÓN DE LA COMPUTADORA DIGITAL

I.7 - ORGANIZACIÓN DE UNA COMPUTADORA DIGITAL

Una computadora digital está conformada por cinco unidades, cada una especializada en una tarea particular. Según dijimos antes, una máquina recibe información, por lo tanto debe tener un sistema de acceso, o UNIDAD DE ENTRADA, la almacena, para lo cual debe poseer una UNIDAD DE MEMORIA, luego deberá realizar sobre la información algún tipo de proceso, para lo cual debe poseer una UNIDAD DE CALCULO o UNIDAD LÓGICA Y ARITMÉTICA, una vez elaborada la información, los resultados deberán ser entregados a través de una UNIDAD DE SALIDA. Lógicamente, para realizar esto, deberá poseer un dispositivo supervisor o de control, que es denominado UNIDAD DE CONTROL.

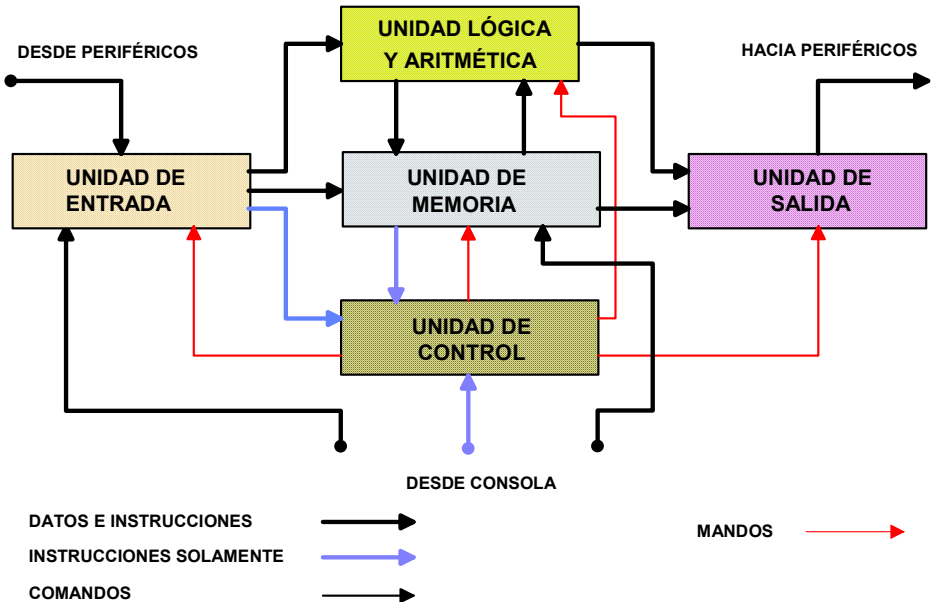


FIGURA I.1 - Esquema de una computadora digital.

En la figura I.1, se tiene el esquema de lo dicho, con las lineas de interconexión necesarias, siendo algunas de ellas para datos e instrucciones, otras solamente para instrucciones y otras para señales de control o comandos o mandos.

La máquina no diferencia entre datos e instrucciones, pues son todas secuencias de números binarios, es el programador el que define su uso, según el camino que les haga tomar o el destino al que llegarán.

Las INSTRUCCIONES son las ordenes que se dan a la unidad de control, para que ella emita comandos a todas las unidades a fin de que sea realizada una cierta operación. Los DATOS, son las cantidades sobre las que se opera.

Evidentemente, de acuerdo a lo dicho, el funcionamiento de una computadora se basa en la transferencia de la información entre distintas unidades, y según sabemos, dentro de las unidades, los soportes de la información son los registros, por tanto, este funcionamiento se basa en la trasferencia de cantidades binarias entre registros, intercalando, o no, operadores booleanos.

La organización detallada, es conocida como organización VON NEUMANN, dado que es él quién la propuso inicialmente.

De una u otra manera, todas las computadoras existentes hasta el presente, respetan esta arquitectura, si bien se han hecho, y se hacen esfuerzos para obtener nuevos modelos que sean de aplicación en algunos casos en particular.

I.8 - CONFORMACIÓN DE CADA UNA DE LAS UNIDADES COMPONENTES DE UNA COMPUTADORA DIGITAL:

Cada uno de los bloques indicados en la figura I.20, cumple una función específica, para lo cual cuenta con una composición especialmente adaptada a su trabajo.

La UNIDAD DE ENTRADA, es encargada de conectar la computadas a aquellos periféricos que pueden suministrarle información, la cual puede llegar ordenada de diversas maneras y a diferentes velocidades de transmisión, por tanto, su misión consiste en reordenarla y adecuar su velocidad a la interna de la máquina. Es evidente que la UNIDAD DE SALIDA cumplirá con la función inversa, o sea adaptar la información al periférico al que está destinada.

La UNIDAD DE MEMORIA, tendrá a su cargo el almacenamiento ordenado de toda la información que debe manejar la máquina, recordando que denominamos información tanto a los datos como a las instrucciones del programa. También es función de la memoria, la búsqueda y entrega de la información a la unidad que lo solicite.

La UNIDAD LÓGICA Y ARITMÉTICA, realizará sobre los datos todas las operaciones que el programa solicite, como asimismo, si le son alimentadas como datos, las instrucciones también serán operadas alterándolas según lo previsto.

Finalmente, LA UNIDAD DE CONTROL, toma las instrucciones y las ejecuta, o sea que interpreta cual es la tarea da ejecutar, y la lleva a cabo emitiendo órdenes a todos los elementos de la máquina que deben actuar para llevarla a cabo.

I.8.1 - UNIDAD LÓGICA ARITMÉTICA:

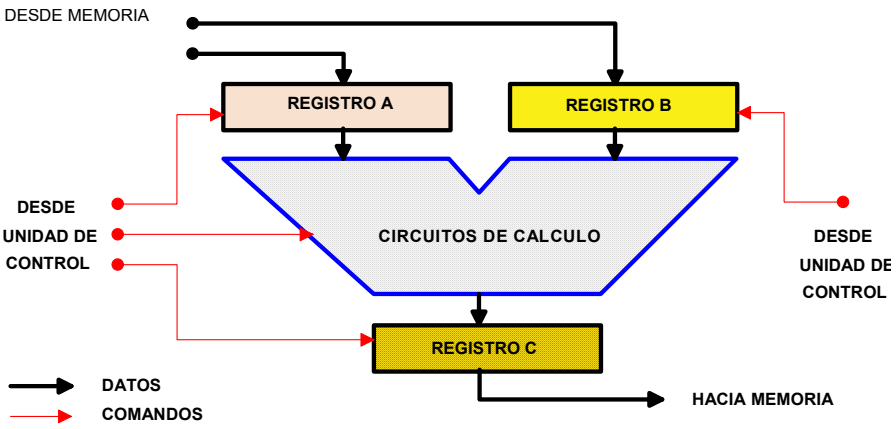


Figura I.2 - Unidad Lógica y Aritmética.

Tiene la conformación indicada en la figura I.2, donde simplemente se observa la presencia de un operador booleano programable, que puede realizar, según cual sea la máquina, varias operaciones según se lo indique la unidad de control, y algunos registros que son los soportes de la información dentro de la unidad.

I.8.2 - UNIDAD DE MEMORIA:

Ya detallamos antes sus funciones, para llevarlas a cabo, necesitará un almacén donde guardar la información, un sistema para escribir y para leer en el mismo, y un sistema para almacenar la información en un lugar predeterminado y para luego poder volver a alcanzar ese lugar para recuperar lo almacenado.

En la figura I.3, se tiene un esquema que puede ser adecuado para lo expresado, que consiste de un registro de dirección, donde se puede tener la dirección, o sea el rótulo o alguna característica que permite individualizar cada una de las posiciones o lugares del almacén donde se guarda una información, un sistema que se encarga de interpretar, o decodificar, esta dirección, el almacén y los circuitos de lectura y escritura. En algunos casos podrá faltar alguno de estos elementos por ser superfluo, o en otros serán necesarios otros dispositivos para hacer útil al sistema.

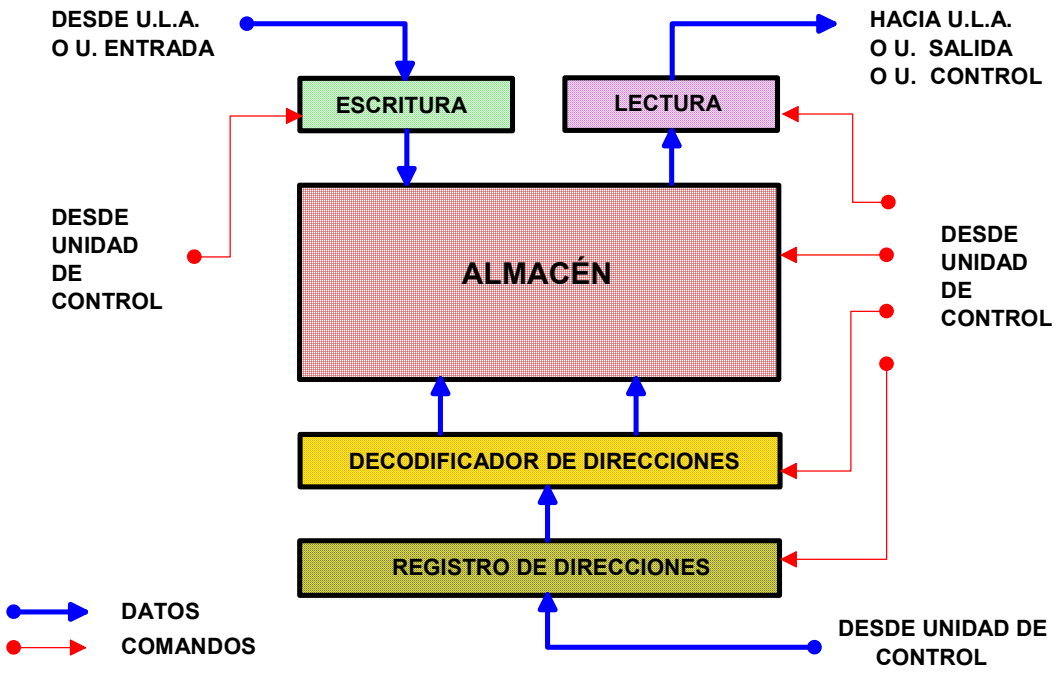
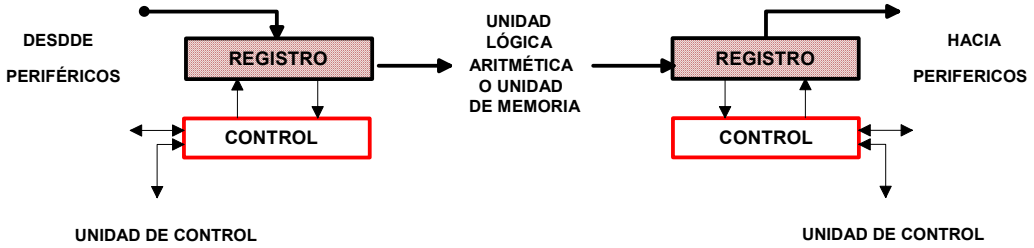


Figura I.3 - Unidad de Memoria.

I.8.3 - UNIDADES DE ENTRADA Y SALIDA:

También según hemos dicho, estas unidades son "adaptadoras" de alguna o algunas características de la información recibida, por lo tanto, deberán poseer cierta autonomía o "inteligencia" y registros. En la figura I.4, se indica la conformación más apta para el propósito.

En dicha figura, se indican con líneas delgadas las señales de comando,



tanto las correspondientes a la comunicación entre periféricos y unidades de E/S, como entre éstas unidades y la unidad de control. Las restantes, corresponden a datos e instrucciones.

FIGURA I.4 - Unidades de Entrada/Salida.

I.8.4 - UNIDAD DE CONTROL:

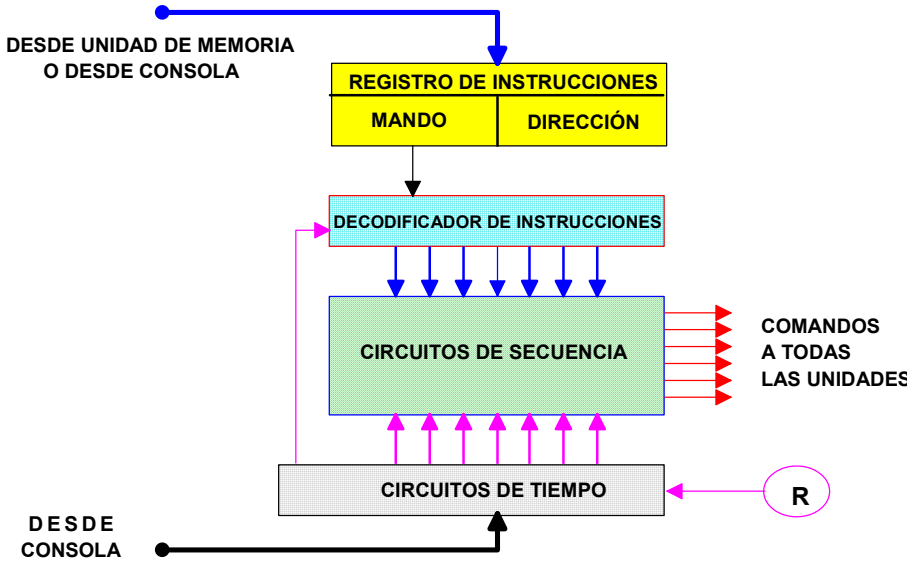


FIGURA I.5 - Unidad de Control.

La Unidad de Control es el cerebro de la máquina, debe interpretar o traducir las instrucciones para hacer que todos los circuitos operen en tiempo y forma adecuados para llevar a cabo la tarea indicada.

Estará formada por el registro de instrucciones, o mejor del registro que almacena la parte mando de una instrucción, su decodificador, y los circuitos que en el tiempo adecuado emiten las señales necesarias, en la secuencia que corresponda a la operación a ejecutar, por tanto tendrá la forma que se muestra en la figura I.5, donde se agrega el reloj, que permite el funcionamiento sincrónico de todos los componentes.

Toda instrucción está siempre formada por dos partes, una que hacer referencia a la acción a realizar, el mando, y otra al dato o a los datos sobre los cuales esa acción se debe ejercer, la cual se denomina dirección.

Los circuitos de tiempo hacen que las señales de comando lleguen en el tiempo adecuado a los componentes de las distintas unidades intervinientes en la realización de la función adecuada.

I.9 - FORMAS DE FUNCIONAMIENTO DE UNA COMPUTADORA DIGITAL:

Existen dos modos o formas principales que puede adoptar una computadora digital, una es la llamada de PROGRAMACIÓN EN HARDWARE, en la cual se interconecta un cierto conjunto de circuitos para que resuelva un dado problema, por lo que toma la forma indicada en la figura I.6. Estas conexiones pueden ser realizadas siguiendo un programa realizado en software.

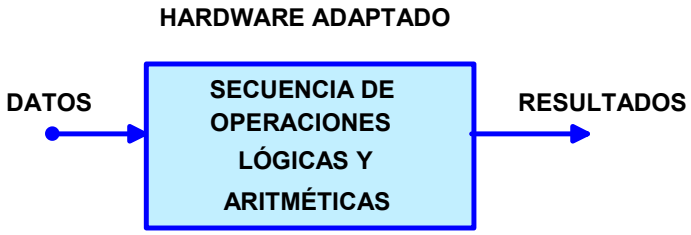


Figura I.6 - Sistema Programado en hardware.

La otra forma de funcionamiento, es mediante el llamado de PROGRAMACIÓN EN SOFTWARE, en el cual en lugar de haber un conjunto de circuitos conectado para resolver un problema, hay una serie de circuitos por los cuales pasa la información a ser transformada.

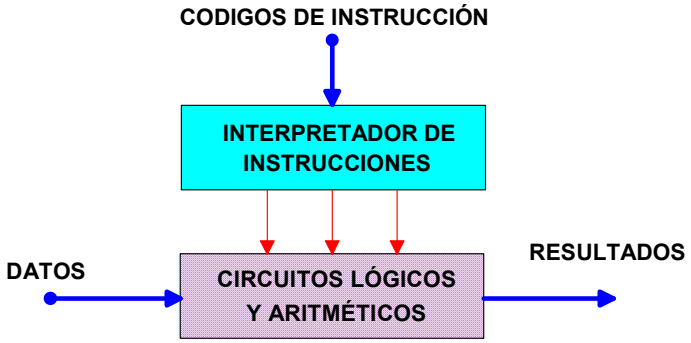


Figura I.7 - Programación en Software.

En esencia, ambas formas resuelven el mismo problema, pero la segunda es más flexible, puesto que no tiene hardware especializado como la primera, la cual resulta ser más rápida, pero también más costosa.

En la figura I.7 se indica esquemáticamente como es el segundo funcionamiento.

I.10 - INTERCONEXIÓN DE LAS UNIDADES DE UNA MÁQUINA.

Podríamos imaginar ahora a una computadora, como la agrupación de las unidades citadas anteriormente, conectadas mediante un adecuado sistema que puede ser uno en que hay una gran cantidad de compuertas y una gran cantidad de hilos que las unen. A este método de conexión podríamos llamarlo de todos contra todos, dado que hay hilos conductores entre todos los dispositivos.

Si utilizásemos este sistema, la maraña de cables sería tal que no podríamos alcanzar a desentrañarla y casi con seguridad que no sería posible construirla por tanto se prefiere una disposición en la cual todas las unidades, mediante las compuertas adecuadas, se conecta mediante un solo medio conductor, con tantos hilos como sea necesario, pero únicos para toda la máquina.

Este método es conocido como el de la barra ómnibus, también conocida más brevemente por bus, la cual recorre la máquina y tiene derivaciones a todas las unidades, tal como la red de distribución de energía eléctrica de una calle, la cual es única y posee derivaciones a todas las casas, donde cada uno puede conectar el artefacto que desee.

En la figura I.8 he diagramado escuetamente como sería la conformación de una computadora, mediante la interconexión de todas las unidades utilizando una barra ómnibus.

I.11 - FUNCIONAMIENTO DE UNA COMPUTADORA

El funcionamiento de una computadora es básicamente como sigue:

1 - La unidad de control habilita los circuitos para realizar la búsqueda en memoria de una instrucción, y una vez hallada la transfiere al registro de instrucciones.

2 - Una vez ubicada la instrucción en el registro de instrucciones, debe proceder a hacerla ejecutar.

En consecuencia, el ciclo básico de máquina consiste en una fase de búsqueda y una fase de ejecución, tal como se muestra en el diagrama de la figura I.9.

En más detalle, el ciclo básico de una computadora, es llevado a cabo en la unidad de control mediante el diagrama de flujo mostrado en la figura I.10. Lógicamente, cada una de las fases se dividirá luego en sub fases, dado que para buscar una dirección, primero hay que calcularla, luego para ejecutarla, habrá que realizar una

serie de operaciones, posiblemente en primer lugar haya que buscar un dato, para lo cual habrá que calcular la dirección del dato, y luego habrá que guardar un resultado, para lo cual también habrá que calcular la dirección donde se debe guardar al mismo.

Es asi que llegamos al diagrama de estados de la figura I.11, donde se indican separadamente los accesos a memoria de las operaciones a realizar en la unidad de proceso, o más abreviadamente CPU (Central Processing Unit o Unidad Central de Procesamiento).

La secuencia es la siguiente:

a - (CDI) *Cálculo de la dirección de la instrucción*: en forma general, la dirección efectiva de una instrucción no es contenida directamente en uno de los registros de la máquina denominado "Contador de Programa" sino que debe ser calculado en base al mismo.

b - (BI) *Búsqueda de la instrucción*: Responde a lo anteriormente visto, o sea alcanzar la posición de memoria indicada, y de traer el código de la instrucción a la Unidad de Control.

c - (DI) *Decodificación de la instrucción:* Este el paso necesario para la interpretación del código de la instrucción, dando origen a los comando necesarios para su ejecución.

d - (CDO) *Cálculo de la dirección del operando:* Cuando la instrucción hace referencia a un operando, es necesario, tal como se hizo para la dirección de la instrucción, calcularla para el mismo, indicando si está en memoria o si es obtenible via Unidad de Entrada.

e - (BO) *Búsqueda del operando:* Lo mismo que para la nueva instrucción, ahora se debe ubicar y trasladar el operando al registro indicado.

f - (EJEC) *Ejecución:* Es la realización de la operación indicada por la instrucción sobre el operando recientemente obtenido.

g - (CDR) *Cálculo de la dirección del resultado:* Es repetir el procedimiento de cálculo de una dirección.

h - (AR) *Almacenamiento del resultado:* Consiste en escribir el resultado en el registro indicado, en la memoria o disponerlo en la Unidad de Salida.

Como vemos, las dos fases antedichas, de búsqueda y ejecución, se han desdoblado en una serie de ocho subfases, que son las indicadas en la citada figura, donde se han introducido otras definiciones tales como:

- *Nueva Instrucción*: Es la instrucción que sigue en la secuencia del programa.

- *Lista o Vector*: Ocurre cuando un dato no es un solo número, sino un conjunto ordenado de los mismos, tal como una lista o una matriz.

- *Operandos múltiples*: justamente es la lista o vector andes citado.

- *Resultados múltiples* : se basa en la misma presunción anterior

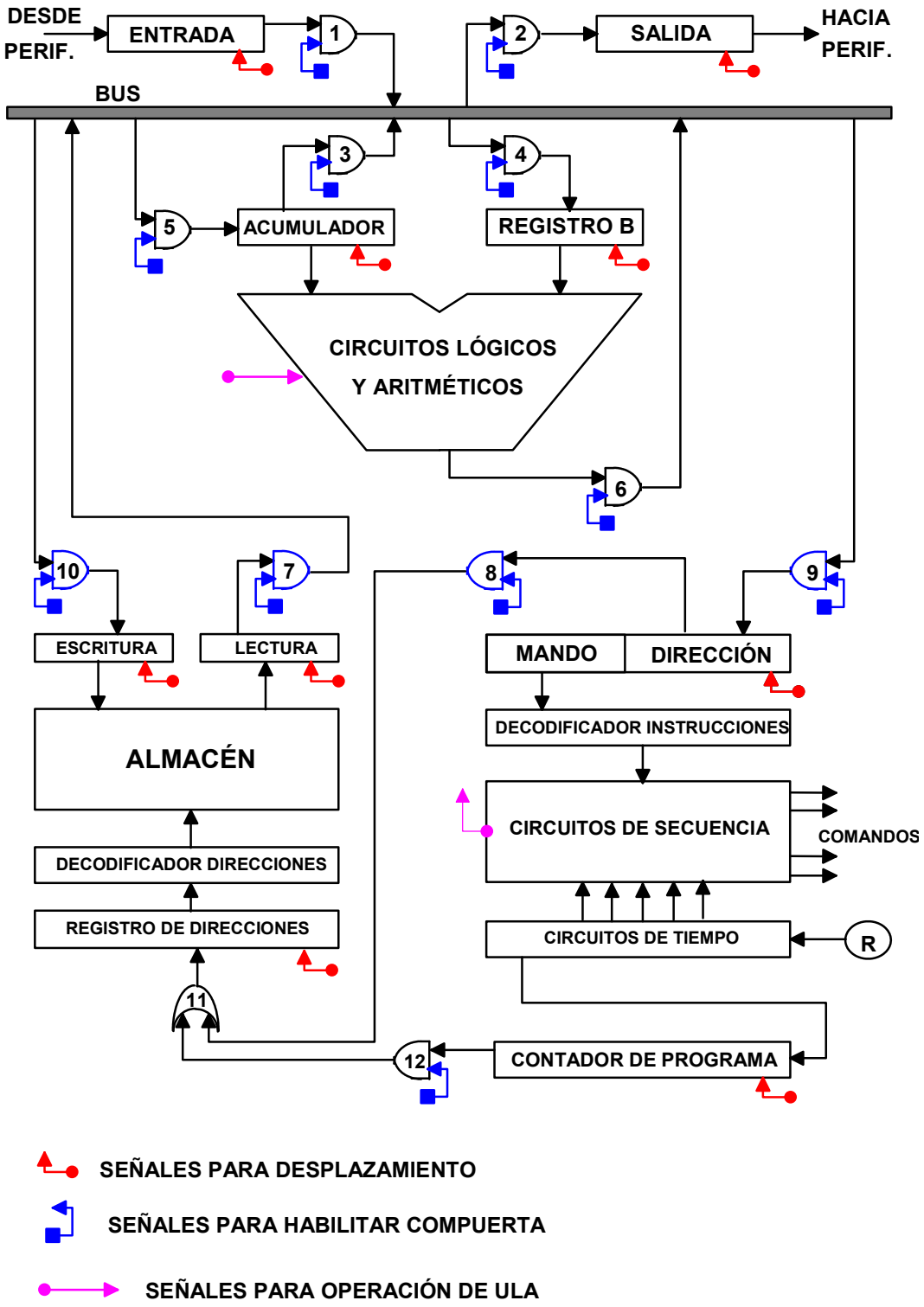


Figura I.8 - Esquema general simplificado de una computadora.

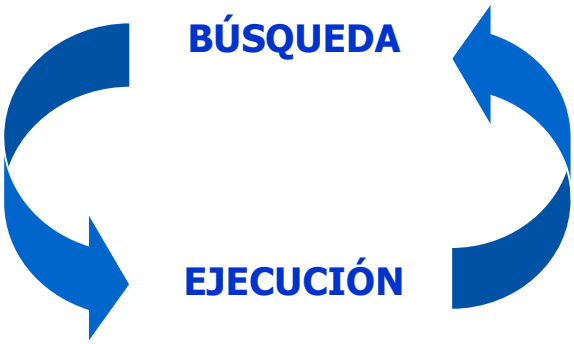


Figura I.9 - Ciclo de máquina.

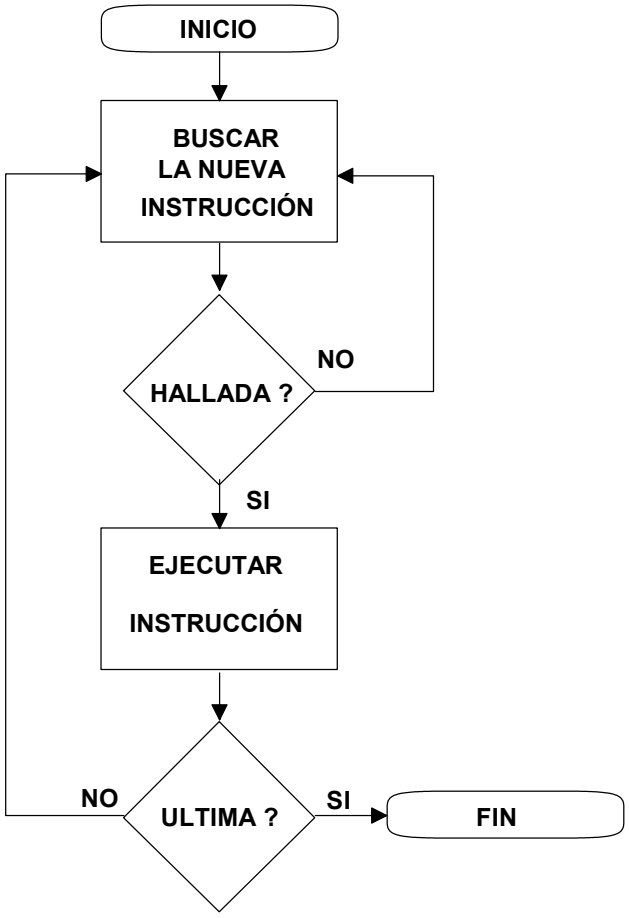


Figura I.10 - Diagrama de flujo del ciclo básico de una computadora.

I.11.1 - INTERRUPCIONES:

Todas las computadoras proveen el medio por el cual desde la Entrada, la Salida o cualquier módulo interno, se puede interrumpir el proceso que están realizando conjuntamente la Unidad de Control y la Unidad Lógica Aritmética, conjunto que es

denominado Unidad Central de Procesamiento (UCP) en castellano y (CPU) Central Processing Unit, en inglés.

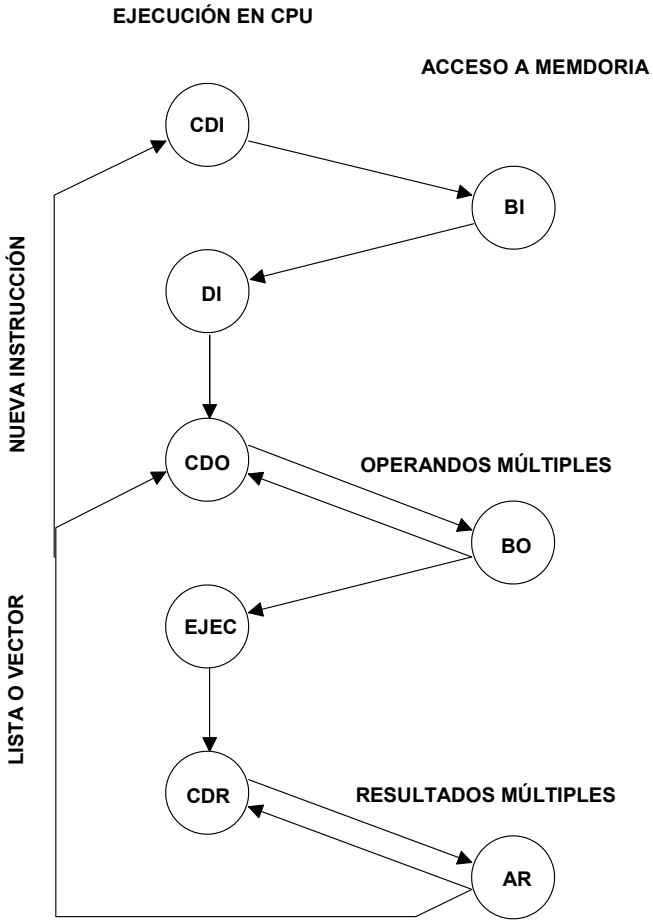


Figura I.11 - Diagrama de estados del ciclo de máquina.

En general estas interrupciones son utilizadas para mejorar la performance de los equipos, especialmente debido a que los periféricos operan a una velocidad mucho menor que el sistema central, por lo que la transferencia de datos se hace muy lenta, desaprovechando la capacidad total.

I.11.1.1 - CLASES DE INTERRUPCIONES:

Las interrupciones pueden clasificarse en cuatro grupos:

a) *Por programa:* **Bajo ciertas condiciones de ejecución de una instrucción, es posible generar una interrupción. Estas condiciones pueden ser: Rebose aritmético (overflow), división por cero, ejecución de un mando ilegal y referencias a espacios de memoria externos.**

b) ***Por tiempo:*** **Generadas por un reloj interno de la máquina, para permitir que el sistema operativo realice ciertas operaciones periódicas de control.**

c) ***De Entrada/Salida:*** **Generada por los controles de E/S para indicar funcionamiento normal o la presencia de condiciones de error.**

d) ***Fallas de Hardware:*** **Generadas por fallas en los controles de paridad, o en la alimentación.**

I.11.1.2 - INTERRUPCIONES Y CICLO DE MAQUINA:

Mientras se ejecuta una operación de E/S, que como dijese son las más lentas, es posible que la CPU siga ejecutando el programa, para ello debe contarse con un buen sistema de interrupciones. Asimismo, es necesario que la Unidad de Entrada/Salida, cuente con los registros necesarios, o bien que el periférico posea memoria.

El procedimiento es como sigue: el programa o el usuario solicitan una operación de E/S, y si el periférico está disponible, se comienza con la transacción, transfiriendo el control a la Unidad de E/S o al periférico, cuando el buffer de entrada/salida, o la memoria del periférico están completas, se emite una señal de interrupción, con lo cual se devuelve el control al usuario o al programa.

Una vez vaciado el buffer o la memoria del periférico, se vuelve, mediante otra señal de interrupción, a ceder el control, y así sucesivamente hasta terminar la transacción pedida.

En general, puede representarse el ciclo de máquina de acuerdo a lo indicado en la figura I.10, donde se tiene un camino sin interrupciones y en la figura I.11, el correspondiente diagrama de estados.

En el diagrama de estados, se ha reemplazado EJEC por E y se agregan dos estados adicionales, uno el correspondiente al control de interrupciones (CI) y otro al estado de interrupción (I). Cuando no hay interrupciones el ciclo se cierra antes, mientras que cuando hay, habrá que resolverlas antes de pasar a la próxima instrucción. El resto del diagrama es exactamente igual al anterior.

I.11.1.3 - INTERRUPCIONES MÚLTIPLES:

En este caso, mientras se ejecuta una interrupción, aparece otra, por ejemplo, mientras la impresora está recibiendo información puede presentarse un caracter ilegal, lo cual da lugar a la segunda interrupción.

Para resolver el problema, se pueden utilizar dos variantes, la primera es inhabilitar toda interrupción que se presente mientras se esté en la ejecución de la primera, y solo cuando ésta termine se pasará el control a la segunda.

Si hubiese más interrupciones anidadas, el proceso es el mismo, se irán ejecutando en orden inverso al de aparición, una vez terminada la operación en curso.

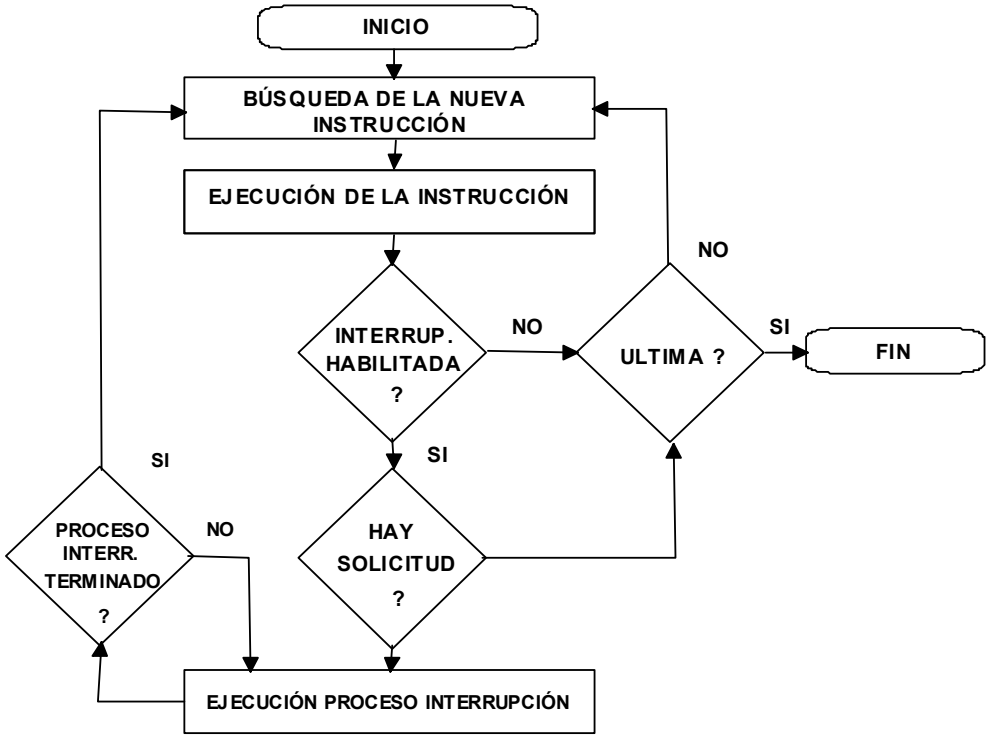


Figura I.12 - Ciclo de máquina con interrupciones.

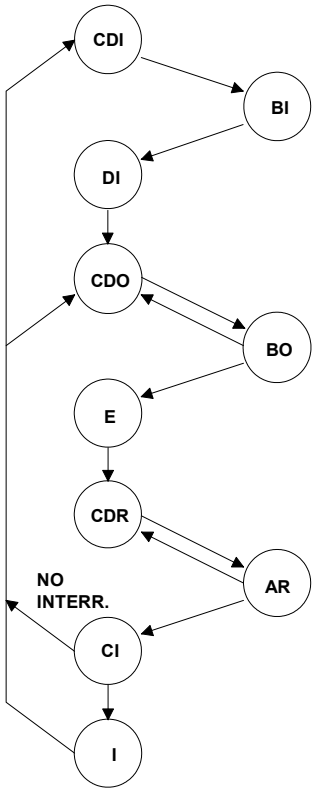
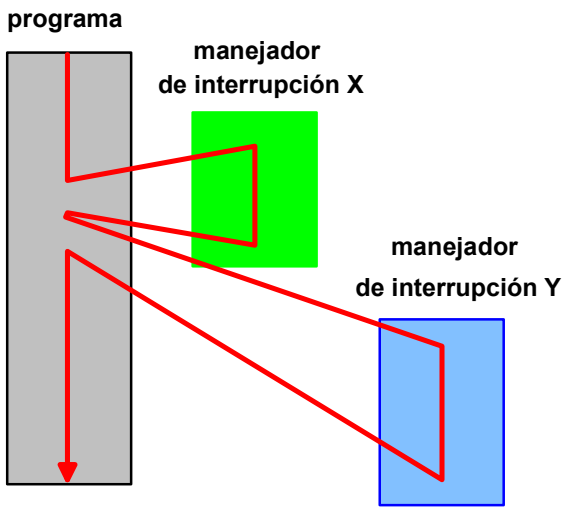
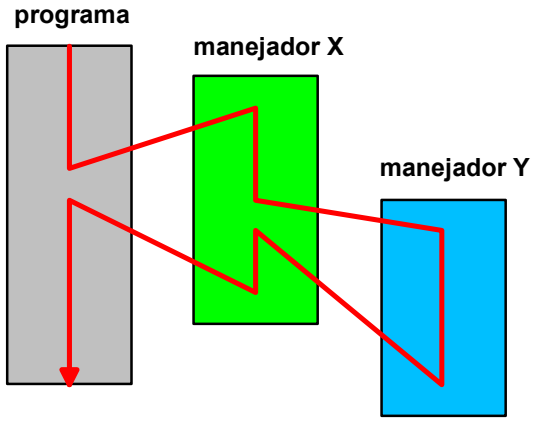


Figura I.13 - Diagrama de estados del ciclo de máquina con interrupciones.



a) Procesamiento de interrupciones secuenciales.



b) Procesamiento de interrupciones anidadas

Figura I.14 - Transferencia de control en interrupciones múltiples.

La otra variante, es la de definir prioridades para las interrupciones, con lo cual las de mayor prioridad deberán despacharse antes de las otras, aún interrumpiendolas si su prioridad es menor.

En el diagrama de la figura I.14, podemos observar las dos variantes citadas.

I.12 - FUNCIONAMIENTO SIMPLIFICADO DE UNA COMPUTADORA:

Volviendo a nuestro diagrama de la figura I.8, el primer paso, según dijimos, será el de buscar y traer la próxima instrucción desde la Unidad de Memoria.

Debemos tener en cuenta que se trata del esquema simplificado de una máquina que opera en serie, es decir que para transferir el contenido de un registro a otro, se precisa entregar a los mismos una serie de pulsos de desplazamiento, que hacen mover (desplazar) el contenido de uno al otro. El sistema empleado puede verse en la figura I.15, donde tenemos dos registros de cuatro bits cada uno, y una compuerta de unión.

La transferencia se realiza bit a bit, habilitando primero la compuerta para que exista un enlace eléctrico, y luego enviando a los dos registros una serie de cuatro pulsos de desplazamiento, en cada pulso un bit se desplaza de uno a otro registro, por ello es que al cabo de cuatro pulsos, la información ha pasado del primero a segundo registro.

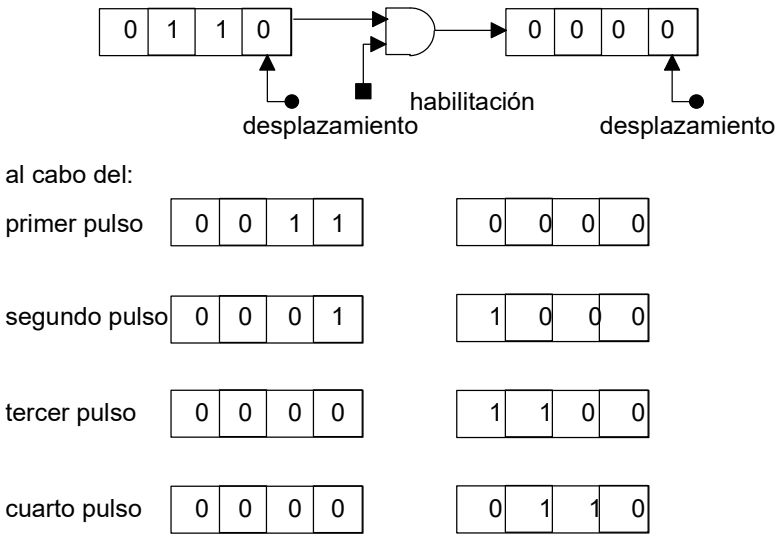


Figura I.15 - Transferencia serie.

Volviendo a nuestro ejemplo, para obtener una instrucción, que puede ser la primera de un programa, la unidad de control emitirá señales para habilitar la compuerta 12, así el contenido del contador de programa es alimentado al registro de direcciones, desde el cual es decodificado, habilitando para lectura la posición de memoria indicada.

Mediante la señal de habilitación de las compuertas 7 y 9, y enviando los pulsos de desplazamiento adecuados, el contenido de la celda de memoria direccionada es pasado al registro de instrucciones, de donde, el mando es tomado directamente por la unidad de control para su interpretación, y mediante habilitación de la compuerta 8, se envía la parte dirección al registro de direcciones, a fin de buscar un operando.

Repitiendo el proceso anterior, y habilitando la compuerta 7 y la 5, mediante los pulsos de desplazamiento adecuados, se remite el operando al registro acumulador.

Luego seguramente habrá que repetir el proceso para llevar otro operando al registro B, y en un tercer paso se realizará la operación, mediante habilitación de las compuertas 6 y 5, se llevará el resultado al acumulador.

Si es necesario guardar este resultado en memoria, la instrucción que sigue seguramente provocará la habilitación de las compuertas 3 y 10, escribiendo el dato en el casillero previamente direccionado.

De la misma forma pueden entrarse datos por el registro de entrada, habilitando la compuerta 1 y la 5 o la 10, o darse salida, mediante la compuerta 7 o 3 y la 2. Por supuesto que en todos los casos se necesitan los correspondientes pulsos de desplazamiento.

Como vemos, faltan algunos detalles, por ejemplo los controles de la entrada y la salida, que permiten conectarse con los periféricos, la entrada mediante interrupciones, y la salida mediante órdenes del programa o también por solicitudes de interrupción.

I.12.1 - LENGUAJE DE MÁQUINA:

Toda computadora funciona mediante una secuencia de instrucciones, consistente en una sucesión de ceros y unos. Dicho de otra manera, una máquina solo entiende números binarios, y solo es capaz de operar con ellos.

Escribir instrucciones en lenguaje binario, es una tarea sumamente tediosa, por otra parte, leer datos y resultados en el mismo lenguaje es sumamente laborioso, es así que para facilitarnos la tarea de programar, se han desarrollado lenguajes más amigables, o más comprensibles.

Uno de los primeros sistemas empleados, es ahora conocido como lenguaje ensamblable, en el cual se utilizan pocos símbolos, preferentemente caracteres alfanuméricos para escribir un programa, y los datos y resultados, resulten escritos en nuestro sistema decimal. De más está decir que cada carácter alfanumérico es representado mediante un código binario.

Ello significa que la máquina deberá traducir la notación alfanumérica a notación binaria, efectuar así los cálculos, y luego volver a traducir los resultados. Por suerte esto lo hace en forma automática, mediante el llamado programa ensamblador.

Entonces tenemos, para hacer una vista gráfica, lo indicado en la figura I.16, donde indico la forma en la cual se ejecuta un programa escrito en lenguaje ensamblable.

A continuación, se desarrollaron lenguajes de orden superior, o más cercanos a nuestra forma de expresarnos, ellos son por ejemplo: el Cobol, el Fortran, el Basic, el Pascal, el C, etc.

En éste caso, se dice que el lenguaje es compilable o interpretable. El lenguaje compilable necesita del auxilio de un programa compilador, el cual se encarga de traducir los programas al lenguaje de máquina. Los lenguajes interpretables, como por ejemplo el Basic, necesitan un compilador más simple, dado que a cada sentencia del

programa corresponde una instrucción en lenguaje de máquina. Mientras que en los compilables, corresponden varias instrucciones de máquina a cada sentencia.

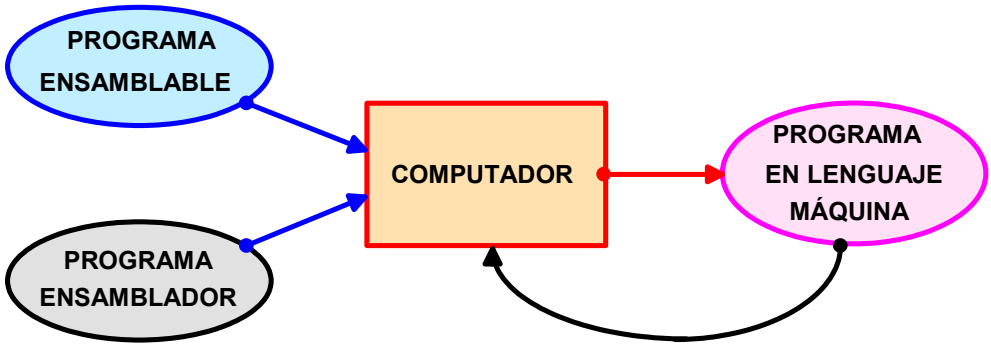


Figura I.16 - Conversión de un lenguaje ensamblable en uno ejecutable.

Asimismo, se utiliza un sistema simbólico para la búsqueda de los datos, lo que significa que a cada dato, constante o variable, se le asigna un nombre, al cual el compilador o el interprete le asignan una, o más posiciones de memoria, las cuales serán indicadas por las direcciones de la instrucción en leguaje máquina.

A veces el compilador usa un paso intermedio, dado que pasa al programa a un lenguaje ensamblable y luego un ensamblador lo pasa a lenguaje de máquina.

En la figura I.17 se indica una forma posible de implementación del sistema a compilador.

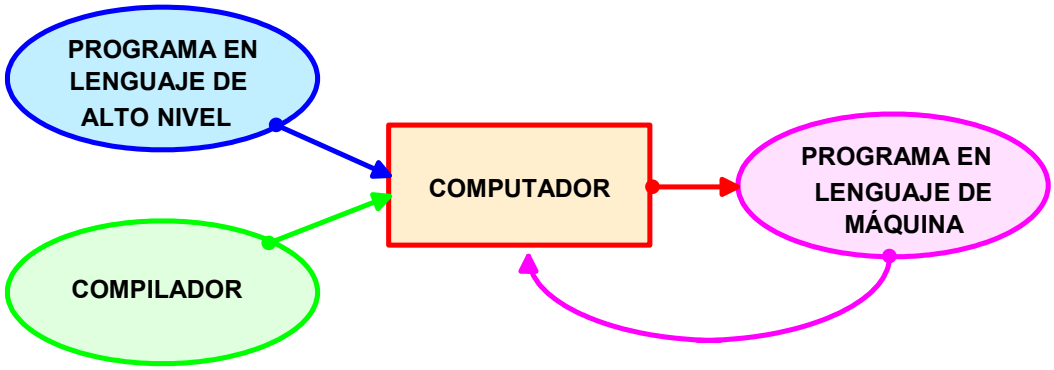


Figura I.17 - Compilación de un lenguaje de alto nivel.

Al programa en lenguaje original, ya sea en ensamblable como el de alto nivel, se lo conoce con el nombre de PROGRAMA FUENTE, mientras que al programa ejecutable, o sea al programa en lenguaje de máquina, se lo conoce como PROGRAMA OBJETO.

I.12.2 - EJEMPLO DE LENGUAJE DE MÁQUINA:

A continuación, vamos a suponer un lenguaje ensamblable y su equivalente binario, en el desarrollo de un pequeño programa.

Función	símbolo	binario
Cargar el registro acumulador desde la memoria	CRA	0 0 0 0
Cargar el registro B desde la memoria	CRB	0 0 0 1
Guardar el contenido del acumulador en memoria	GUA	0 0 1 0
Sumar el contenido del alcumulador con el del registro B	SUM	0 0 1 1
Restar el contenido del registro B del contenido del acumul.	RES	0 1 0 0
Multiplicar el contenido del acumulador y el del registro B	MUL	0 1 0 1
Dividir el contenido del acumulador por el del registro B	DIV	0 1 1 0
Poner un cero en el acumulador	ACO	0 1 1 1
Llevar el contenido del acumul. al contador de programa	LAP	1 0 0 0
Poner en cero el contador de programa	CPO	1 0 0 1
Saltar a la posición xxxxx si el contenido del acumul. es 0	SAO	1 0 1 0
Hacer el AND de los contenidos de acumulador y registro B	AND	1 0 1 1
Hacer el OR entre los contenidos del acumul. y el registro B	OR	1 1 0 0
Complementar el contenido del acumulador	COA	1 1 0 1
Intercambiar el contenido del acumulador con el del reg. B	AXB	1 1 1 0
Poner todo en cero (Hacer reset general)	PTO	1 1 1 1

Nota: Los resultados de todas las operaciones quedan siempre en el acumulador.

Supongamos además que nuestra máquina posee solamente sesenta y cuatro posiciones de memoria, o sea que necesitaremos solamente seis bits para identificarlas (o direccionarlas) a todas.

En consecuencia, una palabra instrucción, tal como es normal definir a una instrucción, tendrá una longitud de cuatro más seis o sea diez bits, su conformación se indica en la figura I.18.

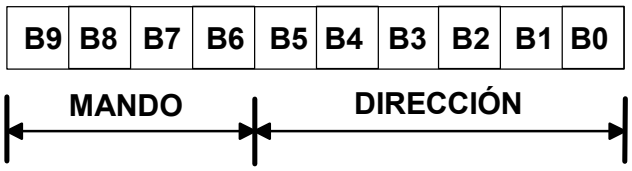


Figura I.18 - La palabra instrucción de nuestro ejemplo.

También haremos el supuesto de tener que resolver la siguiente expresión:

$$a = b \cdot (c+d)/e$$

que contiene cinco variables, a las cuales se debe asignar una posición de memoria.

Una práctica común es la de dejar las posiciones de baja numeración, a partir de la cero, para el programa, y las de mayor numeración para las variables. En la tabla siguiente muestro dicha asignación.

variable	posición	decimal
a	1 1 0 0 1 0	50
b	1 1 0 0 1 1	51
c	1 1 0 1 0 0	52
d	1 1 0 1 0 1	53
e	1 1 0 1 1 0	54

He ubicado al lado de la numeración binaria de cada celda, la numeración decimal, a fin de tener una referencia fácil.

Ahora pasaremos al desarrollo del programa, a tal fin, utilizaré algunas abreviaturas para los registros, por ejemplo, el registro acumulador será denominado como ACM y el registro B, como RB y el contador de programa como CP. Por otra parte, para indicar el contenido de un registro o posición de memoria, utilizaremos la notación de "Backus", en la cual <A> significa contenido de A, y mediante el símbolo =>, el lugar donde debe quedar el resultado de la operación realizada, así la indicación:

ACM <= <ACM> + <RB>

significa sumar el contenido del ACM y el contenido del RB y llevarlo al ACM.

Asimismo, la expresión: RB <= c ; significa llevar la variable c al RB.

Es posible utilizar una forma simbólica para escribir el programa, el que puiede quedar en la siguiente forma:

```
RB <= c                    ; cargar c en RB
ACM <= d                   ; cargar d en ACM
ACM <=  <ACM> + <RB>       ; sumar ACM y RB
RB <= e                    ; cargar e en RB
ACM <= <ACM> / <RB>        ; dividir ACM por RB
RB <= b                    ; cargar b en RB
ACM <= <ACM> x <RB>        ; multiplicar ACM por RB
a <= <ACM>                 ; guardar <ACM> en a.
```

Ahora podemos pasar a utilizar el lenguaje ensamblable que antes hemos definido, con lo que el programa queda escrito como sigue:

```
CRB, c
CRA, d
SUM
CRB, e
DIV
CRB, b
MUL
GUA, a
```

y finalmente, en lenguaje de máquina, o sea una vez ensamblado, quedará la siguiente secuencia de instrucciones de máquina:

0 0 0 1	1 1 0 1 0 0
0 0 0 0	1 1 0 1 0 1
0 0 1 1	
0 0 0 1	1 1 0 1 1 0
0 1 1 0	
0 0 0 1	1 1 0 0 1 1
0 1 0 1	
0 0 1 0	1 1 0 0 1 0

Como vemos en la tabla, hay instrucciones que hacen referencia a memoria y otras que no lo hacen, en este caso todas las que se refieren a operaciones no necesitan hacer referencia a memoria.

Y entonces, podemos llevar todo lo dicho a un mapa de memoria, quedando lo indicado en la tabla insertada a continuación, en la cual he supuesto los siguientes valores:

a = ?; b = 4; c = 32; d = 18; e = 5

Posición		Contenido
0		0 0 0 1 1 1 0 1 0 0
1		0 0 0 0 1 1 0 1 0 1
2		0 0 1 1 0 0 0 0 0 0
3		0 0 0 1 1 1 0 1 1 0
4		0 1 1 0 0 0 0 0 0 0
5		0 0 0 1 1 1 0 0 1 1
6		0 1 0 1 0 0 0 0 0 0
7		0 0 1 0 1 1 0 0 1 0
8		0 0 0 0 0 0 0 0 0 0
9		0 0 0 0 0 0 0 0 0 0
10		0 0 0 0 0 0 0 0 0 0
11		0 0 0 0 0 0 0 0 0 0
12		0 0 0 0 0 0 0 0 0 0
--		--
--		--
--		--
50		0 0 0 0 0 0 0 0 0 0
51		0 0 0 0 0 0 0 1 0 0
52		0 0 0 0 0 1 0 0 0 0
53		0 0 0 0 0 0 1 0 0 0
54		0 0 0 0 0 0 0 1 0 1
55		--
56		--
57		--
58		--
59		--

60		**--**
61		**--**
62		**--**
63		**--**
64		**--**

Lo indicado en la tabla corresponde a la situación inicial, una vez corrido el programa, en la posición 50, quedará:

50		**0 0 0 0 1 0 1 0 0 0**

Según dejé establecido al comienzo, esto no es más que un esquema simplificado de funcionamiento, en especial, faltan la forma de cargar el programa y la de obtener los resultados. Esto es llevado a cabo por una serie de instrucciones fijas ubicadas en la memoria, en posiciones también fijas, que forman el llamado FIRMWARE, y que se ubican físicamente en la llamada memoria ROM o memoria de lectura exclusiva.

En realidad, esta ROM (Read Only Memory) es direccionada por la parte de menor numeración de la memoria, como si formaran parte de la misma, y la memoria utilizable por nosotros, que en la actualidad es denominada memoria RAM (Random Access Memory), contendrá el resto de las posiciones.

Es así que instrucciones tales como cargar el programa, entregar el resultado, etc, hacen referencia a ésta ROM.

Llevando el ejemplo a nuestro diagrama de la figura 1.27, y suponiendo que el programa y los datos están cargados en la memoria, al poner al contador de programa en cero y dar una señal de inicio, señal que puede ser dada por una llave que conecta el reloj a los circuitos de tiempo, se iniciará la acción.

El primer paso será el de lanzar el ciclo automático de la unidad de control, que indica una fase de búsqueda y otra de ejecución. En la de búsqueda, que es la primera que se realiza, se habilita la compuerta 12 y se envian pulsos de desplazamiento al registro de direcciones y al contador de programa.

Dado que tenemos ahora registros de direcciones de seis bits, al cabo de seis pulsos de desplazamiento se habrá pasado todo el contenido del contador de programa al registro de direcciones de la Unidad de Memoria.

El siguiente paso es el de trasferir el contenido de la primera posición de memoria al registro de instrucciones, para lo cual se habilitan la compuerta 7 y la compuerta 9 y se envian pulsos de desplazamiento al registro de lectura y al registro de instrucciones . Ahora la cantidad de pulsos es de 10, los seis correspondientes a la parte dirección más los cuatro correspondientes al mando.

En la figura I.38 indico los elementos involucrados en la parte búsqueda del ciclo de máquina, resaltándolos del resto.

Ahora el contenido del registro de instrucciones de la Unidad de Control es:

0 0 0 1 1 1 0 1 0 0

Donde los primeros cuatro bits son el mando, y cuyo contenido significa trasferir al Registro B de la Unidad Aritmética y Lógica el contenido de la posición de memoria donde se encuentra el valor de c. Esto implica un ciclo de memoria, en el cual primero se habilita la compuerta 8 y se remiten pulsos de desplazamiento a la parte dirección del registro de instrucciones y al registro de direcciones.

Luego se habilitan las compuertas 7 y 4 y se remiten pulsos de desplazamiento a los registro de lectura y B. Con ésto se termina la fase de ejecución, y se vuelve a una fase de búsqueda.

En ésta, se pasa el contenido del contador de programa al registro de direcciones y se busca la segunda instrucción, se la transfiere al registro de instrucciones, quedando al final de la fase, la palabra que sigue en dicho registro.

0 0 0 0 1 1 0 1 0 1

Con el significado de transferir el contenido de la dirección de memoria 110101 al registro acumulador. Al final del proceso de transferencia, tendremos el dato c en el registro B y el dato d en el registro ACM.

La nueva fase de búsqueda, llevará al registro de instrucciones la instrucción siguiente, que es la de sumar el contenido de ambos registros, B y ACM y guardar el contenido en en ACM.

El proceso sigue en la misma forma, en la fase de búsqueda transfiriendo el contenido de la posición de memoria indicada por el contador de programa, al registro de instrucciones, y en la fase de ejecución, ejecutando el mando contenido en registro de instrucciones sobre el dato referido por la parte dirección, también contenida en el mismo, cuando la instrucción hace referencia a memoria.

Finalmente, una vez cumplidas todas las instrucciones del programa, para lo cual al final del mismo habrá que indicar que allí termina, el resultado de la operación queda en la posición de memoria que lleva el rótulo:

1 1 0 0 1 0

Si se desea dar lectura la mismo, es necesario agregar las instrucciones necesarias para ello, buscandolo en la memoria y trasfiriendolo al registro de salida para su dusposición por el usuario. En especial, la instrucción de transferencia al registro de la Unidad de Salida no se ha incluído en el listado de la página 42.

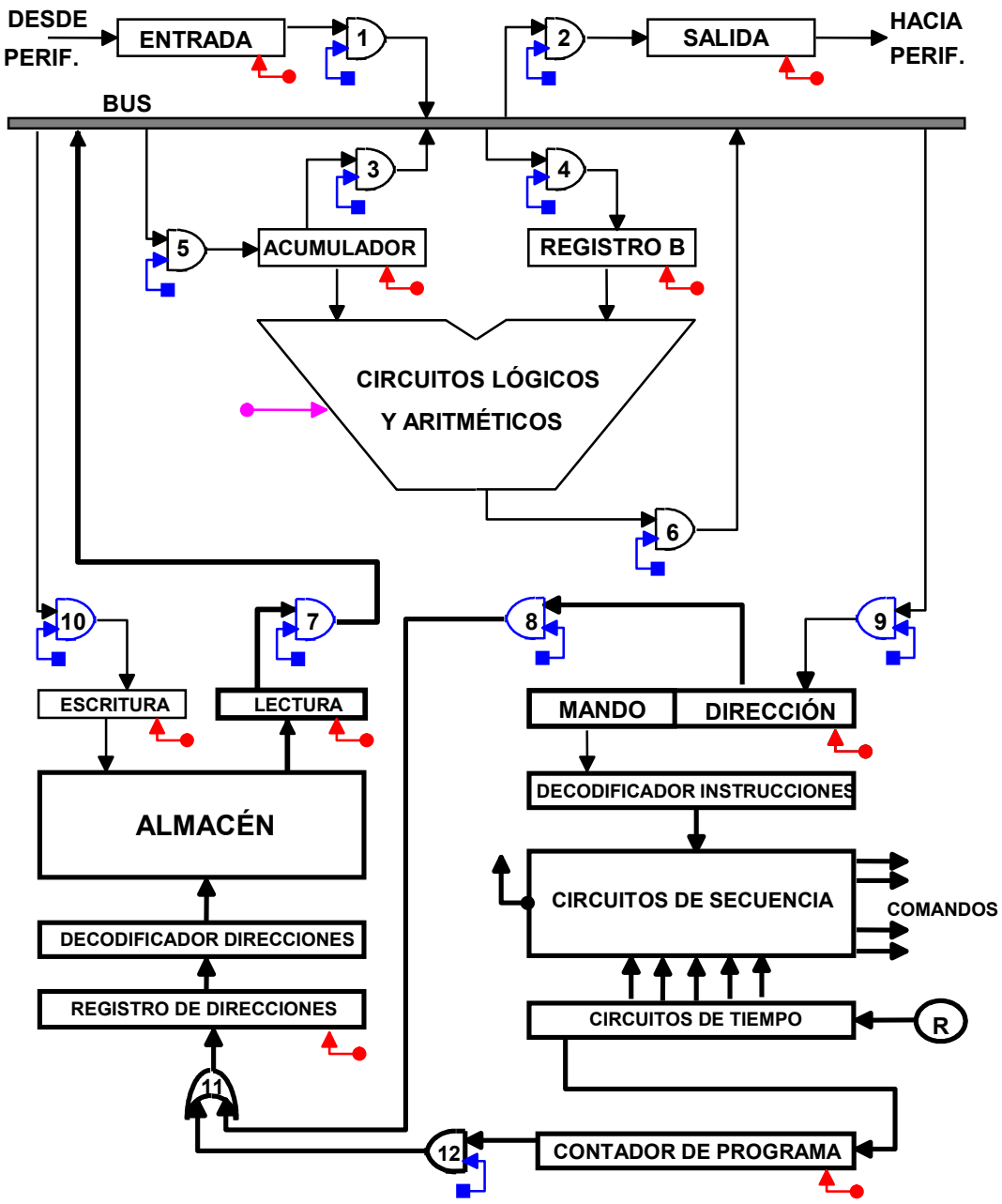


Figura I.19 - Elementos involucrados en una fase de búsqueda.

En la figura I.19 se nuestran como remarcados todos los elementos involucrados en una fase de búsqueda, donde, según ya dijimos, el proceso es el siguiente:

1 – La UC envía la parte direccion del regitro de instrucciones al Registro de direcciones de la UM habilitando la compuerta 8.

2 – La dirección es decodificada en la UM y es tomado el contenido de la casilla direccionada y colocado en el registro de lectura de la misma.

3 – Del registro de lectura, la Unidad de control lo enviará al BUS, activando la compuerta 7.

A continuación deberá ocurrir una fase de ejecución, donde la UC toma el mando del Registro de Instrucciones y después de decodificarlo, remite las señales correspondientes a todas las unidades de la máquina que deben intervenir para la resolución del mismo mando.

Supongamos en primer lugar que el mando correspondía a la carga desde la UE de uno o más datos, entonces los elementos que toman parte en el proceso son los remarcados en el diagrama mostrado en la figura I.20, donde se observa, que la unidad de control debe activar las compuertas 8, para mandar la dirección a la UM, luego la compuerta 1 y 10 para enviar el dato entrado al driver de escritura y finalmente dar la señal de desplazamiento al mismo para su paso al almacén.

Si ahora, teniendo los datos en la memoria, deseamos cargar un programa, no deberemos más que continuar con el mismo procedimiento remarcado en la figura I.20.

Una vez que contamos con los datos y el progama, se puede pasar a la ejecución del mismo, así tal como lo hiciesemos con el programita en assembler, podemos pasar a ejecutar el programa, para lo cual se carga la primera instrucción en el Registro de Instrucciones de la UC, y luego se ejecuta. Esto se verifica en la forma indicada en la figura I.21.

Seguramente ésta primera instrucción indicará cargar un dato en la ULA, en la cual tenemos dos registros involucrados, el acumulador y el B. En las figuras I.22, se muestra la carga del acumulador.

En tercer lugar, llegará el momento de operar entre los datos cargados en la ULA, lo cual ocurre según lo mostrado en la figura I.23, donde los circuitos aritméticos se interconectan, bajo la orden de la UC, para realizar la operación.

De aquí pueden partir dos caminos, o guardar el resultado en memoria, para lo cual se debe pasar el contenido del Registro Acumulador a una dada posición de memoria, o se recarga el Registro B para continuar la operación.

En la figura I.24 se tienen indicados los elementos que operarán en el caso de almacer el resultado. Finalmente, en la figura I.25 tenemos la disposición para dar salida a un resultado a traves de la US. En todos los casos, la UC ha remitido las señales de habilitación de las correspondients compuertas, y ha enviado los pulsos de desplazamiento a los registros indicados.

TODAS LAS COMPUTADORAS DIGITALES OPERAN DE ÉSTA MANERA

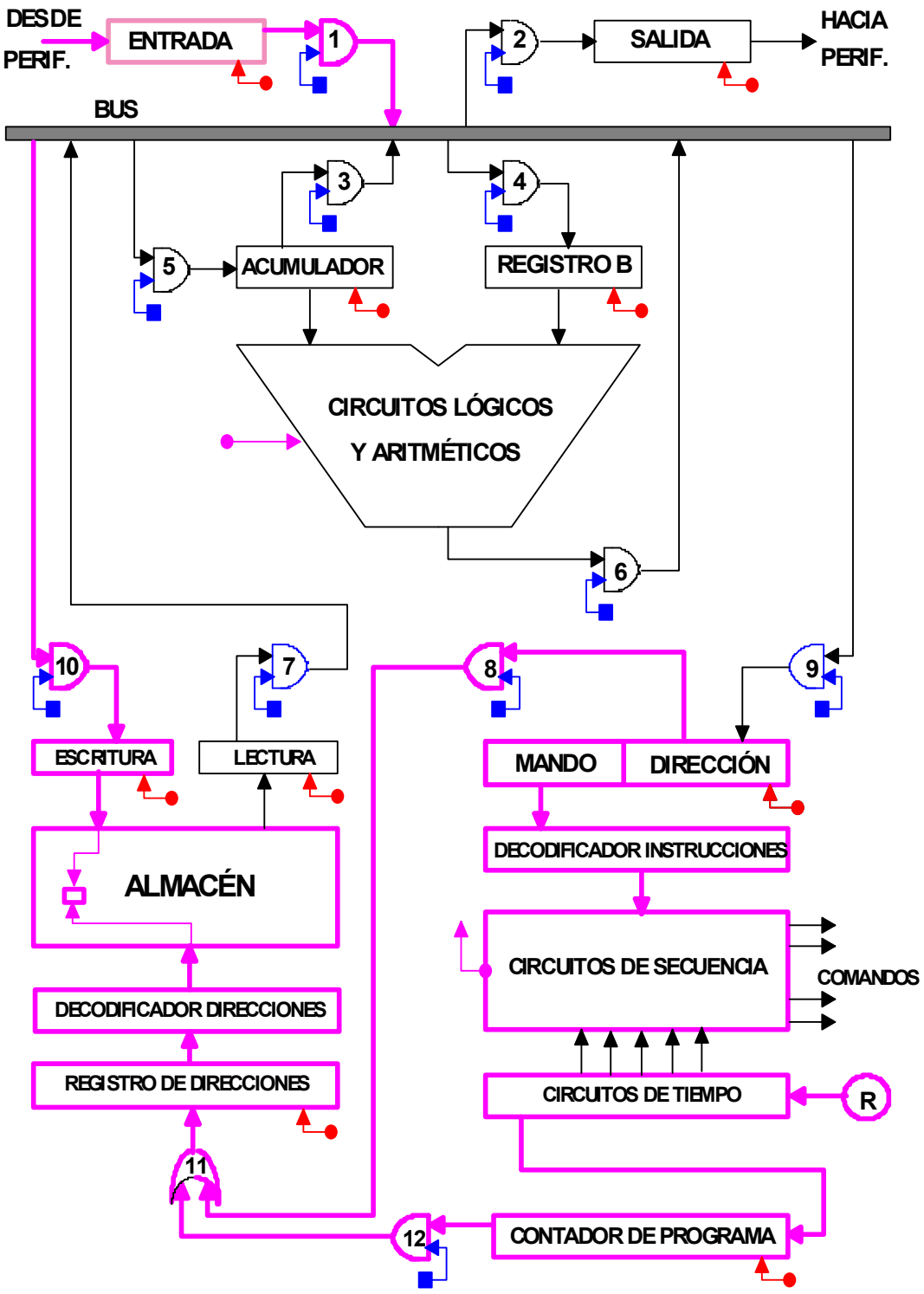


Figura I.20 – Carga de datos e instrucciones en la memoria.

El Contador de Programa contiene la dirección de memoria donde debe guardarse un dato o una instrucción. La instrucción actual es la de cargar cantidades en la memoria. Como la carga es desde la entrada, se deberá habilitar la compuerta 1, el BUS y la compuerta 10, llevando así la entrada al driver de escritura, de donde pasará a la posición indicada por el contador de programa.

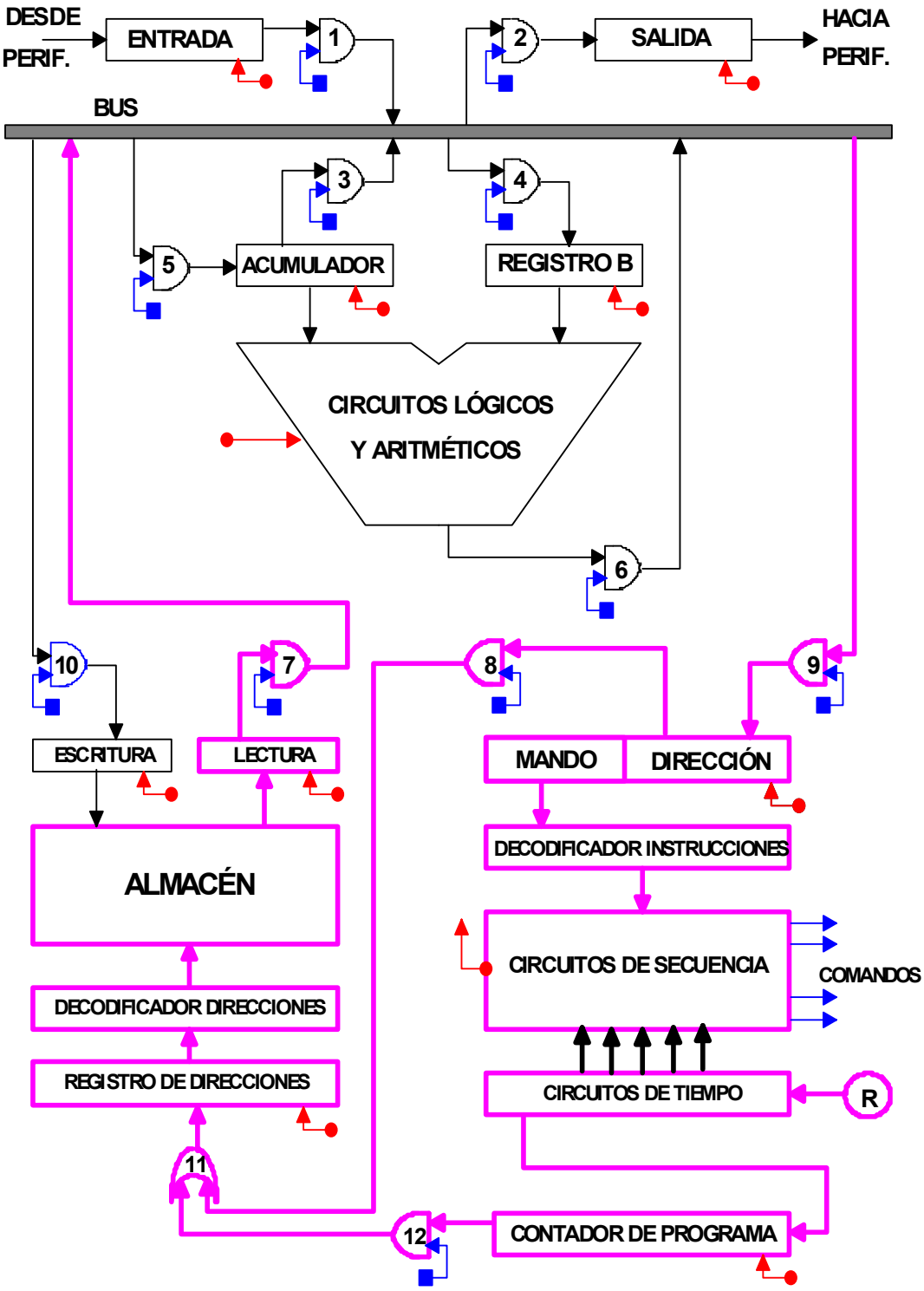


Figura I.21 – Busqueda de la nueva instrucción.

En éste caso, el Contador de Programa apunta a la primera instrucción, mediante la compuerta 12 se pasa la dirección a la Unidad de Memoria, de donde el contenido de esa posición de memoria es pasado al driver de lectura. De aquí, mediante la compuerta 7 se alimenta al Bus y finalmente la mediante la compuerta 9 es llevada al registro de instrucciones.

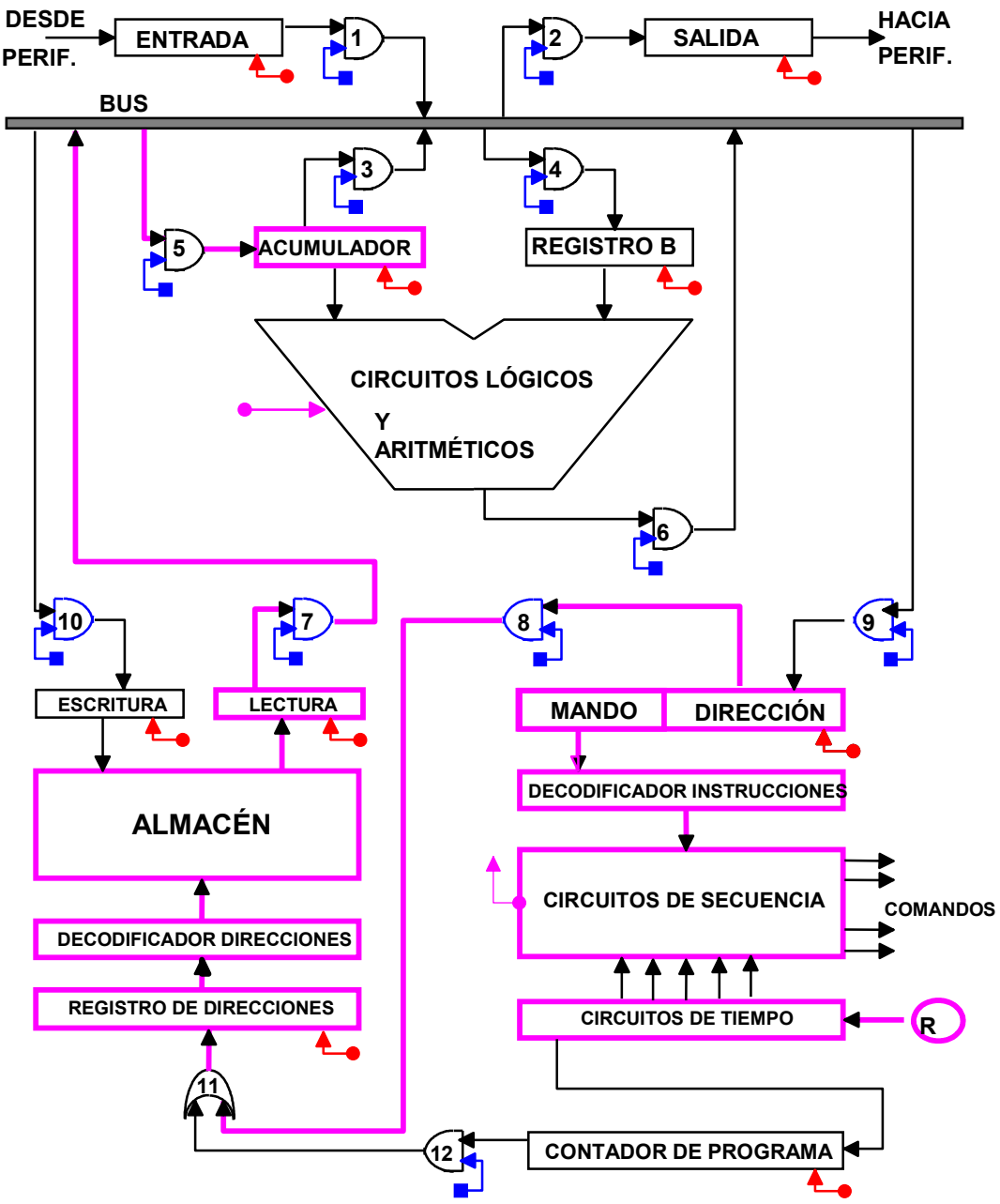


Figura I.22 – Carga del Acumulador.

Para realizar operaciones en la ULA, se deben cargar los registros, ello puede llevarse a cabo la habilitación de la compuerta 8, que llevará la dirección a la UM, y de la posición de memoria apuntada, se lleva al driver de lectura. Mediante la compuerta 7 el dato pasa al BUS y mediante la 5 puede ser pasado al Acumulador.

Si es necesario llevar el dato al registro B, se utilizará la compuerta 4 en vez de la compuerta 5.

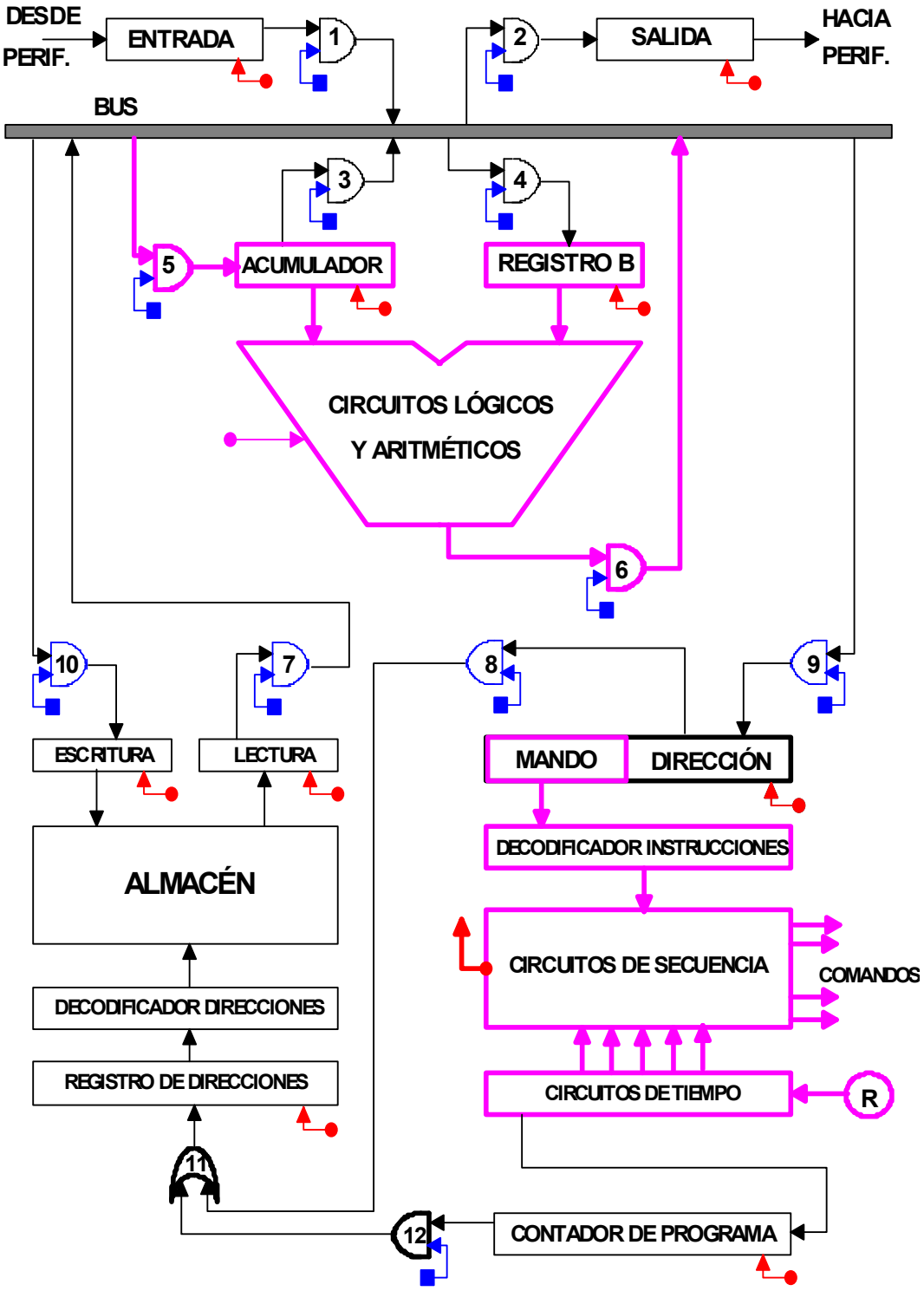


Figura I.23 - Realización de la operación.

Ahora el mando es decodificado o interpretado, alimentando así a los circuitos de secuencia, de donde saldrán los comandos en forma adecuada. Uno de los comandos, indicará a la ULA como disponer sus circuitos para realizar la operación pedida. Los contenidos del Acumulador y del Registro B, son tomados para hacer la operación, y como el ACM queda desocupado, mediante la compuerta 6 se pasa al BUS y mediante la 5 del BUS al ACM.

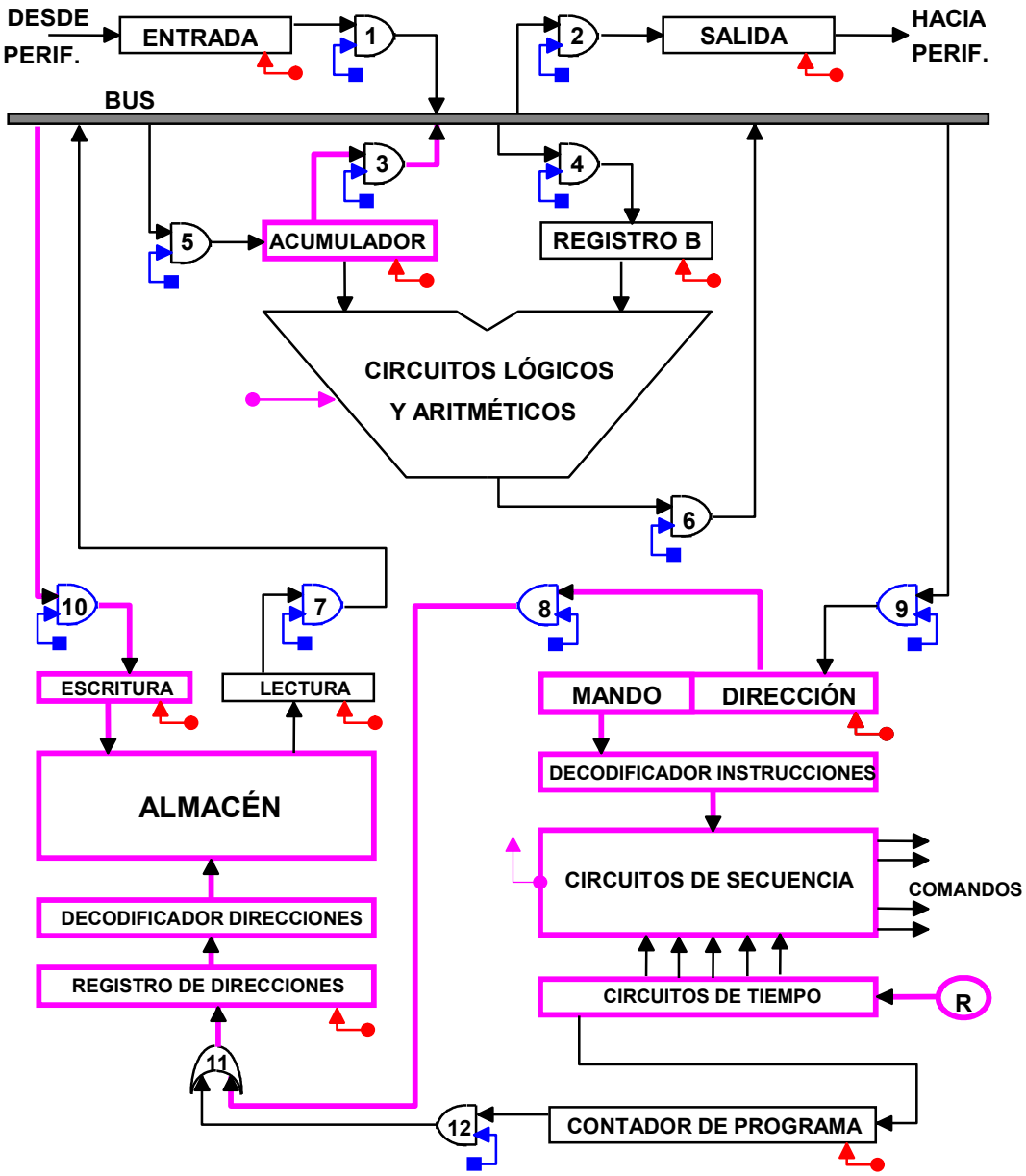


Figura I.24 – Almacenando el resultado.

En éste caso el contenido del ACM debe pasar a la memoria, a la dirección indicada por el registro de instrucciones en la parte correspondiente, mientras el mando indica Guardar. Mediante la compuerta 8 se lleva la dirección a la UM quién habilita la posición direccionada, y luego mediante la compuerta 3 se lleva el contenido del ACM al BUS y mediante la 10 del ´BUS al driver de escritura.

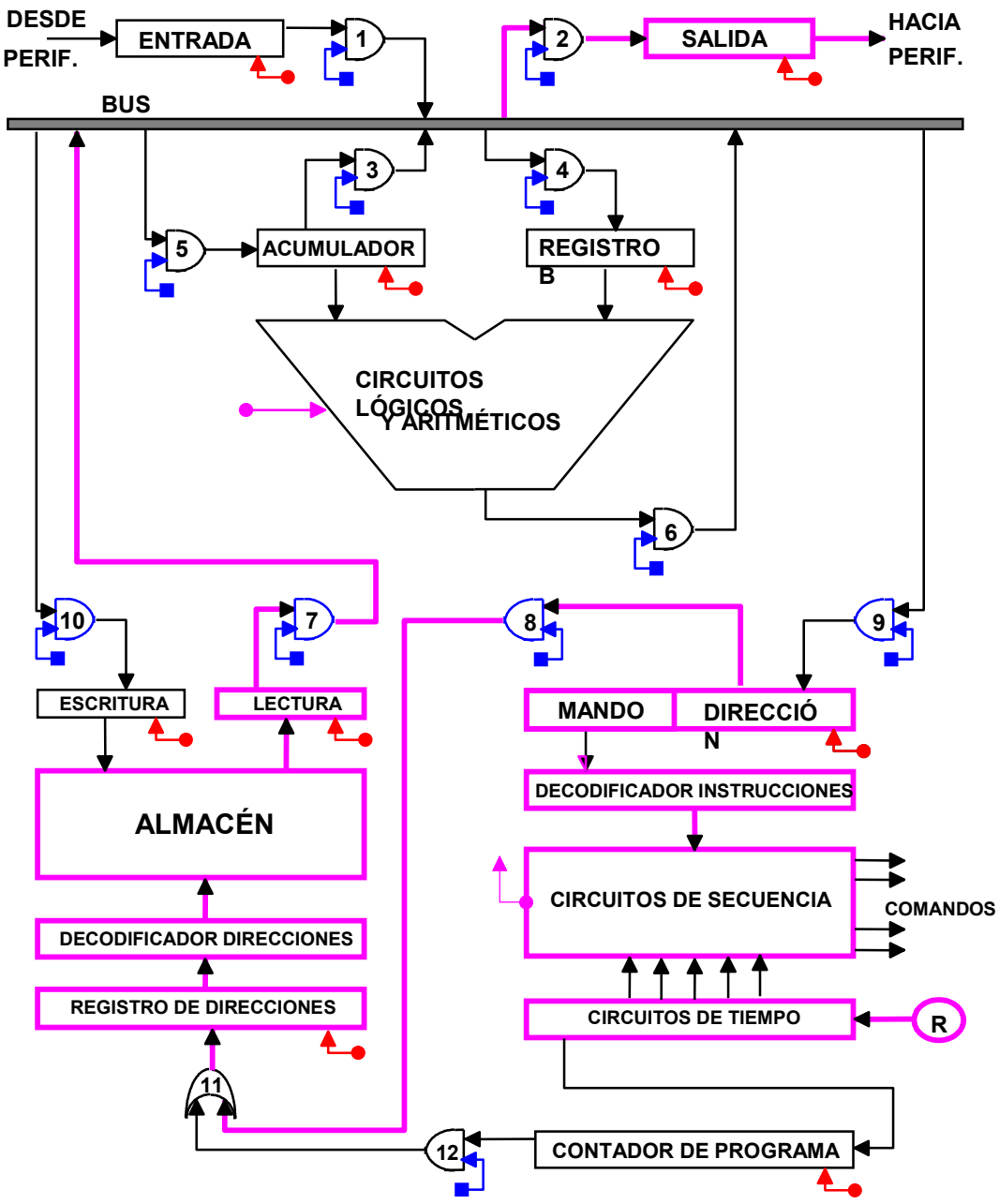


Figura I.25 – Dando salida al resultado.

Ahora el mando dirá enviar a la US el contenido de la posición de memoria indicada en la parte dirección del registro de instrucciones, mediante la compuerta 8 es llevada a la UM, donde mediante el registro de direcciones y el decodificador de direcciones se habilita la posición direccionada, encargándose la UM de enviarla al driver de lectura. La compuerta 7 llevará el dato al BUS, y la compuerta 2 al registro de sa.lida.

I.12.3 - SISTEMA OPERATIVO:

Todos los programas que hacen al funcionamiento de una máquina, a que pueda aceptar datos, entregar resultados, dar advertencias sobre el funcionamiento, presentar las pantallas, traducir programas, supervisar el funcionamiento, etc., es denominado SISTEMA OPERATIVO, sin el cual ninguna máquina puede funcionar, salvo en lenguaje de máquina.

En realidad, la comunicación entre el hombre y la máquina se lleva a cabo mediante dicho sistema operativo. Por tanto nosotros no nos enteramos del funcionamiento interno del computador, sino que nos comunicamos con el sistema operativo y solo debemos saber su funcionamiento para utilizarlo.

Actualmente los sistemas operativos ha evolucionado hacia modos de operación gráficos, lo cual nos aleja aún más de las intimidades de la máquina, presentandosenos como una forma de visualizar datos y confeccionar programas muy sencilla, completamente intuitiva y al alcance de cualquier persona. Pero por supuesto, el sistema operativo se programa en la forma habitual, mediante sentencias en algún lenguaje de alto nivel.

El sistema operativo está almacenado normalmente en varios lugares, cada uno de los cuales tiene un objetivo puntual. Una primera parte, en una memoria ROM que tiene la misión de inicializar la máquina, o sea dar los parámetros iniciales para el arranque luego del encendido. Una segunda, contenida en una memoria RAM, normalmente de la familia C-MOS para que sea de muy bajo consumo, que contiene los parámetros del sistema, tales como cuales periféricos hay conectados, y de que tipo son, y una tercera parte en un disco rígido, que es la encargada fundamentalmente de la presentación.

I.13 - SISTEMA DE COMPUTACIÓN:

La computadora por si sola no puede comunicarse con el ser humano, sino que es necesario disponer los periféricos adecuados tanto para entregarle información, como para obtener los resultados. Es así que se conforma un sistema, cuyo centro es la máquina, pero a su alrededor se disponen una gran cantidad de equipos tales como: pantallas, teclados, lectores de cinta, lectores de discos, graficadores (plotters), impresoras, dispositivos para comunicaciones, sensores varios, etc.

En la figura I.26, se tiene un ejemplo de conformación de un centro de procesamiento de datos basado en una mainframe, o sea en un computador de gran capacidad.

Como puede observarse, el computador se comunica con los controladores de cada periférico, así es menor la tarea que debe desarrollar para la trasferencia de datos.

Asimismo, se han dispuestos dos buses normalizados, también denominados interfaces o interfases, como abreviatura de los términos ingleses INTERFACE SPECIFICATIONS que significan cuales son las especificaciones tanto mecánicas como eléctricas que se deben satisfacer para la unión de los dispositivos analizados.

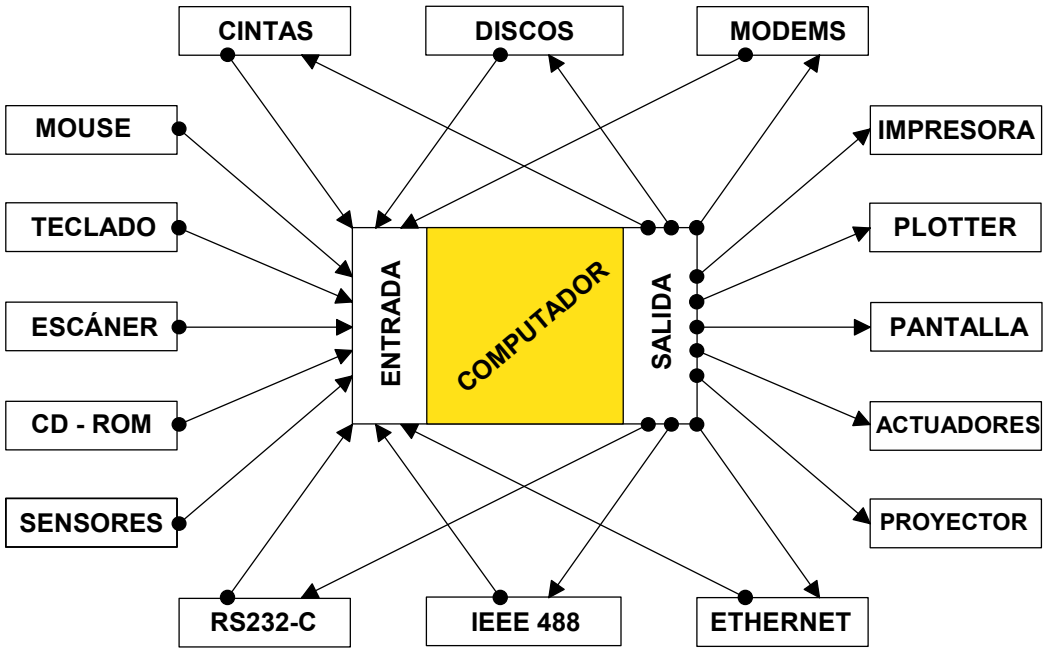
En un caso se ha dispuesto el interfase serie RS 232 C, que es utilizado fundamentalmente para comunicaciones, mientras que en el otro se ha indicado el IEEE 488, que es paralelo y se lo utiliza principalmente para mediciones.

Existe una amplia variedad de buses, cada uno de ellos previsto para una aplicación específica, en este caso solo hemos nombrado los dos mas empleados en aplicaciones comunes.

En la actualidad el dispositivo de salida más empleado es el tubo de rayos catódicos, con pantalla color, y para almacenamiento externo el disco magnético rígido. Lo normal para el intercambio de programas es el disco flexible, y para aplicaciones multimedia el disco óptico, mientras que en caso de necesitar respaldos de mucha capacidad se aplican cartuchos de cinta magnética.

Se usan plotters para obtener gráficos o planos e impresoras de matriz de puntos, chorro de tinta o laser, a fin de tener documentos escritos de las elaboraciones realizadas.

Figura I.26 - Esquema general de un sistema de computación.



Finalmente, si la computadora se utiliza para el control de procesos, aparte de los medios de comunicación con el hombre, se necesitan sensores para determinar el estado del proceso, y actuadores para modificarlo de acuerdo a las necesidades del mismo.

Justamente cuando se emplea la máquina para el control de procesos, o el control de maquinarias, es común implementar un sistema integrado, en el cual hay varias computadoras interconectadas en red, para proceder al proyecto (CAD,

Computer Aided Design o Diseño Asistido por Computador) y a la manufactura (CAM, Computer Aided Manufacturing o Manufactura Asistida por Computador).

Asimismo, debido a la creciente cantidad de instrucciones de un programa y a la enorme cantidad de cálculos que se debe realizar sobre una gran cantidad de datos, se emplean grandes sistemas de computación formados por gran cantidad de computadores, interconectados de diversas maneras entre ellos.

En general toda supercomputadora actual está formada por una gran cantidad de procesadores (que no son más que computadores elementales, aunque de gran capacidad), capaces de operar conjuntamente en paralelo. Es así que por ejemplo la serie RS6000 de IBM está conformada por máquinas que alcanzan a tener hasta 512 procesadores. De ésta serie es derivada la "DEEP BLUE" máquina que juega al ajedrez y ha batido hacia mediados del año 1997 al entonces campeón mundial de ese deporte.

En la figura se tiene una fotografía de la supercomputadora VPP 700 E, fabricada por Fujitsu Limited de Japón. La misma puede tener hasta 256 procesadores, alcanzar velocidades de cálculo de hasta 614,4 Gigaflops, y memoria de hasta 512 Gigabytes. Es una de las máquinas comerciales más poderosas y económicas (con relación a las prestaciones) de la actualidad.

1.B.1. REPRESENTACIÓN DE DATOS

1.B.1.1- CODIFICACIÓN:

Dado que no es conveniente emplear directamente el sistema binario de numeración, por la dificultad que el tiene para su interpretación, y además por cuanto es necesario no solo tener números en la máquina, sino también otros elementos, tales como caracteres alfabéticos, de control, tipográficos, etc.

Para representar los diez dígitos decimales, se necesitan diez combinaciones distintas de dígitos binarios, ello solo es posible cuando se utilizan 4 de estos, lo que nos da dieciséis combinaciones posibles, de las cuales solo se eligen diez.

1.B.1.1.1 - CÓDIGO BCD:

BCD significa Binary Coded Decimal, o sea Decimal Codificado en Binario, es el más simple y utilizado de los códigos numéricos, dado que sigue la numeración binaria convencional del 0 al 9, descartando las combinaciones correspondientes a los decimales 10 al 15, o sea A a F en hexadecimal.

El código queda:

0 = 0 0 0 0	**5 = 0 1 0 1**
1 = 0 0 0 1	**6 = 0 1 1 0**
2 = 0 0 1 0	**7 = 0 1 1 1**
3 = 0 0 1 1	**8 = 1 0 0 0**
4 = 0 1 0 0	**9 = 1 0 0 1**

1.B.1.1.2 - CÓDIGOS POR PESO:

A veces, por determinados motivos, tales como facilidad para hallar el complemento, o por requerir un menor consumo de menor energía para su transmisión, etc., se utiliza el criterio de asignar un peso o valor a cada una de las posiciones binarias, así podemos decir que en el BCD, los valores de cada posición, tomando de izquierda a derecha, son: 8; 4; 2; y 1.

Dicho de otra manera, el primer binario encontrado pesa 8 unidades, el segundo pesa cuatro, el tercero dos y el cuarto una.

Por tanto, para leer el código 0 1 1 0, podemos decir:

$$0x8+1x4+1x2+0x1 = 6$$

1.B.1.1.3 - CÓDIGOS ALFANUMÉRICOS:

Según dijera antes, no solo es necesario representar números en la máquina, sino también caracteres alfabéticos, de control y otros símbolos.

En general, se prefiere utilizar un sistema de dos dígitos hexadecimales para tener las diferentes combinaciones, además que ello implica el uso de ocho bits, o

sea una unidad bien establecida y normalizada en computación, que es denominada "Byte".

Entre los varios sistemas desarrollados, podemos, en la actualidad, citar dos, el EBCDIC (Extended Binary Coded Decimal Interchange Code) o código de intercambio de información BCD extendido, que ha sido utilizado por IBM en todas sus máquinas, y el ASCII (American Standard Code for Information Interchange) o Código Americano Standard para el Intercambio de Información, que es leído como ASKI, y en la figura 1.B.1 se indica en forma de tabla.

b_7				0	0	0	0	1	1	1	1
b_6				0	0	1	1	0	0	1	1
b_5				0	1	0	1	0	1	0	1
b_4	b_3	b_2	b_1								
0	0	0	0	NUL	DLE	SP	0	@	P	``	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	“	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	&	5	E	U	e	u
0	1	1	0	ACK	SYN	%	6	F	V	f	v
0	1	1	1	BEL	ETB	´	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	-	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	_	o	DEL

Figura 1.B.1 - Código ASCII

En dicha tabla la lectura se hace por filas y columnas. Las columnas dan la primera parte o primer nibble (incompleto) del byte, pues se utiliza el código de siete bits y las filas el restante. Así un símbolo ubicado en la columna 3 y fila 4, es el que lleva la numeración 34, que en binario es el:

0 1 1 0 1 0 0

contendrá el símbolo: 4.

De otra manera, si tenemos que representar el símbolo +, vemos que corresponde a la columna 2 (010) y a la fila 11 (1011).

Los símbolos que están formados por combinaciones de letras, en general corresponden a caracteres de control, con el significado dado a continuación:

BS = (Backspace) retroceso
HT = (Horizontal Tab) tabulación horizontal
LF = (Line Feed) alimentación de línea
SOH = (Start of Heading) comienzo del encabezamiento

STX = (Start of Text) inicio del texto
ETX = (End Of Text) fin del texto
EOT = (End of Trasmission) fin de la trasmisión
ENQ = (Enquiry) solicitud de respuesta
ACK = (Aknowledge) reconocimiento del mensaje
VT = (Vertical Tab) tabulación vertical
FF = (Forma Feed) alimentación de la nueva página
CR = (Carriage Return) retorno del carro de la impresora
NAK = (Negative Aknowledgment) reconocimiento negativo
SYN = (Syncronism) señal de sincronismo
ETB = (End of Trasmission Block) fin de la transmisión del bloque de datos.
FS = (File Separator) seaprador de archivos
GS = (Group Separator) separador de grupos
RS = (Record Separator) separador de grabaciones
US = (United Separator) separador de todo
NUL = (Null) nada, carácter vacío
BEL = (Bell) campanilla
SO = (Shift Out) los caracteres que siguen deben ser interpretados como fuera del código.
SI = (Shift In) los caracteres que siguen pertenecen al código
DEL = (Delete) borrar
SP = (Space) espacio
DLE = (Data Link Escape) carácter que cambia el significado de lo que sigue
DC1, DC2, DC3, DC4 = (Device Control #) controlador de dispositivos nº
CAN = (Cancel) cancelar
EM = (End of Medium) final del medio utilizado para lectura o escritura
SUB = (Substitute) sustituto
ESC = (Escape) se usa para dar a los caracteres que siguen un significado alternativo.

Dado que el código ASCII solo permite representar 128 caracteres diversos, la empresa IBM ha desarrollado el EBCDIC fundamentalmente para la transmisión de datos entre sus supercomputadoras. Dado que como se observa en la tabla incluída como Anexo I al presente capitulo, se tienen algunos espacios vacíos, los cuales pueden ser empleados para la representación de nuevos caracteres o controles. Asimismo, en el Anexo II se muestra otra manera de expresar la tabla de código ASCII, incluyendo la forma empleada en transmisiones html (HiperText Markup Language, o sea Lenguaje Aumentado para Hipertextos)

Finalmente, dada la existencia de variadas formas de expresión de la escritura en los diversos países, los códigos de 7-8 bits no alcanzan para representarlos a todos, se ha desarrollado el denominado UNICODE, o UNIversal CODE, que está formado por una secuencia de 16 bits. Ver página Web en la siguiente dirección: http://www.unicode.org.

1.B.1.1.4 - OPERACIONES ENTRE NÚMEROS CODIFICADOS.

Cuando se realizan operaciones con números codificados en binario, generalmente en el código BCD, al realizar una suma se presentan tres condiciones:

a) que el resultado esté comprendido entre 0 y 9, por lo que será correcto.

b) que el resultado esté comprendido entre 10 y 15, lo que deberá ser detectado y corregido.

c) que el resultado sea 16 o más, lo cual también implicará una corrección.

Veamos como sucede haciendo algunos ejemplos: supongamos la suma de 3+4 = 7, que es:

```
   3 = 0 0 1 1
 + 4 = 0 1 0 0
 -------------
   7 = 0 1 1 1
```

obteniendo un resultado correcto.

En cambio, si sumamos 6+5 = 11, tendremos como respuesta:

```
   6 = 0 1 1 0
 + 5 = 0 1 0 1
 -------------
  11 = 1 0 1 1
```

Si bien la suma en binario es correcta, no es posible traducirla al decimal, pues corresponde a un código no válido, en realidad en binario debería leerse:

0 0 0 1 0 0 0 1 = 11

Vemos que si agregamos 6 al resultado tenemos la respuesta correcta:

```
    1 0 1 1
  + 0 1 1 0
 -----------
1   0 0 0 1
```

El tercer caso ocurre por ejemplo, cuando sumamos 8+9 = 17

```
  8  = 1 0 0 0
+ 9  = 1 0 0 1
--------------
 17 = 1 0 0 0 1
```

Por lo que utilizando el criterio anterior, se leería el decimal 11, entonces corrigiendo, agregando seis como antes, tendremos:

```
  1 0 0 0 1
+   0 1 1 0
-----------
  1 0 1 1 1
```

que con lo antedicho, quedará:

0 0 0 1 0 1 1 1

que es el 17 en BCD.

1.B.1.2 - NÚMEROS EN COMA FIJA Y EN COMA FLOTANTE:

En nuestro sistema de operar, utilizamos la notación denominada de coma fija (punto fijo para los anglosajones), en el cual una cantidad es escrita observando siempre la posición de la coma decimal, por ejemplo:

234,56
1.244.322,987.6
0,25

y así sucesivamente.

Este sistema es el ideal para los contadores, dado que ellos tienen que tener en cuenta todas las cifras significativas, por pequeñas que sean.

En cambio, en los cálculos científicos, pueden tenerse o bien cifras muy grandes, como por ejemplo las distancias astronómicas, o muy pequeñas, como los espacios interatómicos, por lo cual si utilizásemos una notación como la anterior, tendríamos algo así:

7.845.670.000.000.000

o algo así:

0,000.000.000.000.000.045.6

Entonces es conveniente utilizar una notación que puede ser denominada científica, o mas convenientemente, notación en coma flotante.

Así las cantidades anteriores quedarán expresadas como:

$$0{,}784.567 \times 10^{16}$$

Lo que significa que se debe correr la coma decimal dieciséis lugares a la derecha, para tener la cantidad expresada en coma fija.

En el segundo caso tendremos que escribir:

$$0{,}456 \times 10^{-16}$$

lo que significa que se deben anteponer entre la coma y el primer dígito significativo la cantidad de dieciséis ceros.

Si expresáramos los números anteriores mediante el código BCD, tendríamos:

0111 1000 0100 0101 0110 0111 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000

y para el segundo:

0000, 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0100 0101 0110

Como es posible observar, son cantidades muy difíciles de leer a primera vista, así como ocupan muchos espacios con ceros en la máquina, por tanto la notación en coma flotante es mucho más práctica.

En una computadora, la cantidad es soportada por un registro, en el cual hay un espacio para la mantisa, o sea la parte significativa del número, y otra para el exponente, pues la base es siempre la misma, diez.

Además, se utiliza un sistema denominado normalizado, en el cual la primera cifra significativa siempre es escrita a la derecha de la coma decimal, por tanto tampoco es necesario tener un espacio para la misma.

Por otra parte, las cantidades pueden llevar signo, para lo cual se reserva una posición binaria en el registro, si allí hay un cero el signo es positivo, si hay un uno, el signo es negativo. Asimismo se utiliza otra posición para el signo del exponente, a menos que se utilice un sistema relativo, en el cual los exponente se escriben sin signo, pero si son mayores de una cierta cantidad se dice que son positivos y negativos cuando ocurre lo contrario, ello implica que para su determinación haya que realizar un cálculo.

En consecuencia, una cantidad en la máquina, por ejemplo la segunda, se tendrá en un registro en la siguiente forma:

1 0001 0110 0000 0000 0100 0101 0110 0000 0
Se exponente mantisa Sm

vemos que el registro tiene el primer bit para signo del exponente, luego admite un exponente de cuatro cifras y una mantisa también de cuatro cifras, más otro bit para signo de ésta.

A veces, no se utiliza signo para el exponente, sino que el mismo se dice polarizado, por cuanto se supone que es cero cuando tiene un cierto valor medio entre el mínimo y el máximo que puede escribirse en el número de bits previsto para el mismo.

1.B.1.2.1 - OPERACIONES EN COMA FLOTANTE:

Para realizar sumas o restas, deben primero igualarse los exponentes, a fin de que a ambas mantisas les quede alineada la coma decimal. Luego se puede operar y finalmente normalizar el resultado.

Por ejemplo:

$$0{,}234 \times 10^{7} + 0{,}12 \times 10^{-2}$$

para igualar los exponentes a partir del menor, deberemos ir desplazando a la derecha el segundo número tantas veces como lo indique su diferencia. En este caso, es:

7 - (-2) = 9

por lo tanto deberemos agregar nueve ceros después de la coma, por lo que quedará:

$$0{,}234x10^{7} + 0{,}000.000.000.12x10^{7} = 0{,}234.000.000.12x10^{7}$$

En el caso del producto o la división, se puede operar directamente, pero el exponente del resultado será igual a la suma de los exponentes en el caso del producto, y a la resta del correspondiente al dividendo menos el del divisor. En ambos casos luego se deberá normalizar el número nuevamente. Por ejemplo, supongamos tener:

$$0{,}234x10^{7} \text{ x } 0{,}45x10^{-4} = 0{,}1053x10^{3}$$

o también:

$$0{,}98x10^{-4} / 0{,}653x10^{2} = 1{,}543307x10^{-6}$$

y normalizando:

$$0{,}1543307 \text{ x } 10^{-5}$$

1.B.1.2.3 - NORMAS DEL IEEE PARA NÚMEROS EN COMA FLOTANTE:

En la norma 755 del IEEE, se definen las dos mas importantes representaciones de las cantidades en coma flotante, el formato simple y el formato doble, ambas además con una variante denominada extendida. Esta normalización sirve para mejorar la portabilidad de los programas.

En el formato simple, se tiene, de izquierda a derecha:

Bit de signo	Exponente polarizado	Fracción
1 bit	8 bits	23 bits

mientras que en el formato doble, es:

Bit de signo	Exponente polarizado	Fracción
1 bit	11 bits	52 bits

Por otra parte, en la variante extendida la palabra alcanza los 43 bits en la variante simple, y 79 bits en la doble.

El exponente polarizado significa que el valor cero corresponde a un cierto valor medio, dado que el mismo no lleva signo. En la variante simple, el exponente alcanza los valores de 0 a 253 y en la doble de 0 a 2045. Ello significa que los valores pueden ir de -126 a +127 en el primer caso y de -1022 a + 1023 en el segundo. La polarización es de 127 y 1023 respectivamente.

Mientras que en el caso de la variante extendida, el exponente puede alcanzar o superar los 11 bits en el caso simple y los 15 en la doble. Y la parte significativa o mantisa, los 31 y 63 bits respectivamente.

1.B.1.3 - COMPLEMENTACIÓN DECIMAL Y BINARIA:

Supongamos tener un disco, en cuya superficie se han marcado 100 divisiones perfectamente equidistantes, con lo cual nos sirve para contar de 0 a 100.

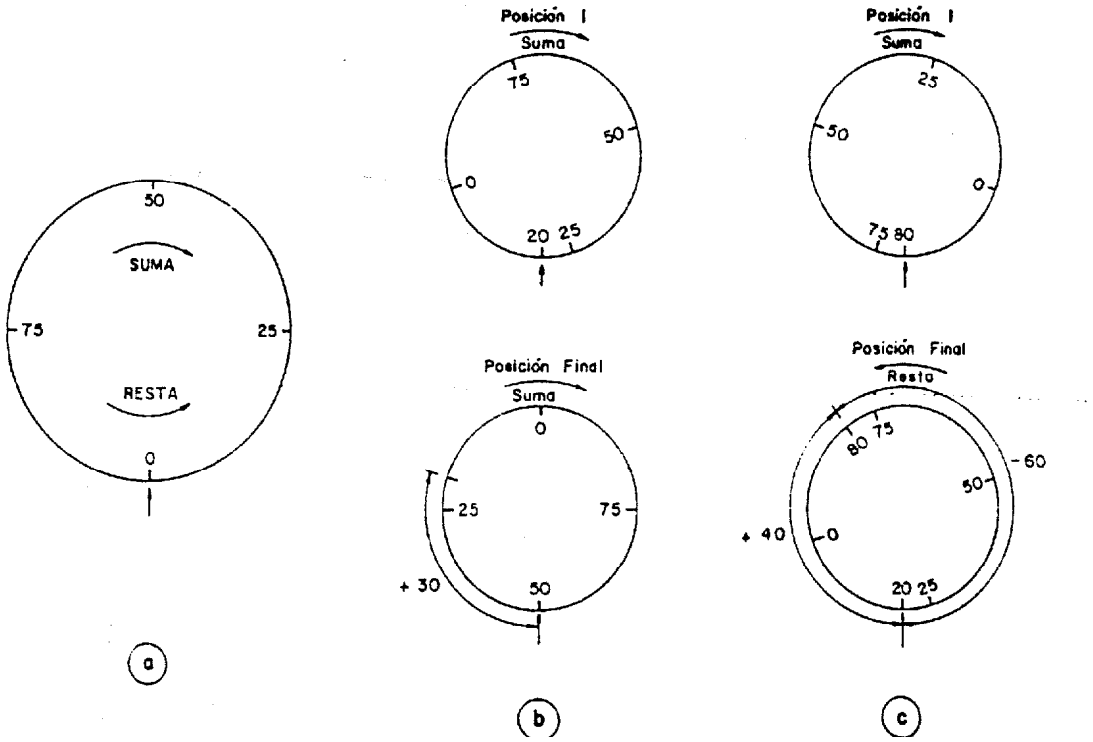


Figura 1.B.2 - Complementación.

Pongamos un índice para marcar una cierta posición, y un eje alrededor del cual puede girar, tal como se muestra en la figura 1.B.2. Al hacer girar el disco en sentido horario, la cantidad que apunta hacia el índice aumenta, y disminuye si lo hacemos girar en sentido contrario.

De ésta manera, el disco nos sirve como ayuda para la suma algebraica, pues si queremos hacer, por ejemplo, 20 + 30, hacemos girar primero el disco hasta la posición 20, luego le agregamos el giro de 30 posiciones más y nos queda indicado el número 50. (Ver Figura 1.B.2.b)

En cambio, si partimos de 80 y lo giramos en sentido inverso al de las agujas del reloj en 60 posiciones, el índice quedará marcando 20, que es el resultado de la resta, y es igual a la suma de 80 + 40, en éste disco. (Ver Figura 1.B.2.c)

Quiere decir que podremos sumar cualquier cantidad, pero que llegue solamente hasta 100, o podemos restar cualquier cantidad de 100, hasta llegar a cero. A éste número máximo de nuestro disco, se lo denomina MÓDULO, por cuanto es una cantidad máxima representable en el sistema.

Por otra parte, vemos que para sumar hay que girar en un sentido, y para restar en el otro, lo cual significa tener un motor reversible.

Supongamos ahora que posicionamos al disco en la posición 80, y le sumamos 100 - 60 = 40, la posición final será nuevamente 20, o sea que hemos realizado la resta girando al disco siempre en el sentido de la suma.

Pues bien, a la cantidad 40, se la denomina COMPLEMENTO A 100 DE 60, y generalizando, podemos decir que:

COMPLEMENTO DE N = MÓDULO - N

En sentido general, el complemento de un cierto número, será:

10000 - 4573 = 5427

En nuestros sistemas, dada la cantidad de cifras a representar, por ejemplo 8, en cualquier calculadora de mano, no podemos hablar de un módulo en sentido general, sino en sentido restricto, o sea cifra por cifra. Es así que en un sistema decimal hablamos de un módulo 10, y en números binarios, de módulo 2, con lo cual igualamos el módulo a la base de numeración.

Entonces, en sistema decimal, el complemento a 10 de una cierta cantidad, se calculará restando de diez la cifra menos significativa, y las restantes de nueve.

Tomando el ejemplo anterior, hacemos:

```
   9 9 9 10
 - 4 5 7 3
C10 = 5 4 2 7
```

que es el mismo resultado.

También podríamos haber dicho que deseamos el complemento a nueve, en cuyo caso se restan todas las cifras de 9, con lo cual nos da un complemento una unidad menor que el complemento a diez:

```
   9 9 9 9
 - 4 5 7 3
C9 = 5 4 2 6
```

Según veremos más adelante, para nosotros será lo mismo uno que otra complemento, basta saber cual es el utilizado.

Al tratar con números binarios, tenemos el complemento a 2, que será:

```
   1 1 1 10   Mód = 16
 - 0 1 0 1  =  5
C2 = 1 0 1 1  =11
```

y el complemento a 1:

```
   1 1 1 1   Mód. = 16
```

- 0 1 0 1 = 5
C1 = 1 0 1 0 = 10

Entonces, lo mismo que antes, el complemento a 1 es igual al complemento a dos menos una unidad.

Cuando los números están codificados, se debe trabajar con los cuatro bits del mismo simultáneamente, como si estuviésemos en un sistema decimal.

1.B.1.4 - APLICACIÓN DE LOS COMPLEMENTOS:

Según ya vimos, los mismos son útiles en la realización de operaciones aritméticas entre números con signo, para utilizar siempre el mismo circuito aritmético de suma.

1.B.1.4.1 - SUMA ALGEBRAICA:

Denominamos suma algebraica a la suma de números con signo, existiendo varias formas de llevarla a cabo, pero en este lugar solo nos interesa el uso de complementos a 10 y a 9.

1.B.1.4.1.1 - SUMA ALGEBRAICA CON COMPLEMENTOS A 10:

Se presenta el caso cuando las cantidades a sumar poseen signos diferentes, siendo conveniente la Complementación del que lleva signo negativo, para luego realizar la suma, siguiendo el procedimiento dado como ejemplo.

Sea sumar:

+8.276 -> +8.276
-5.421 -> C10 = +4.579
2.855 1 2.855 = 2.855

Como vemos hay un rebose, el que debe dejarse de lado, y entonces el resultado es el correcto.

No es este el único caso, pues puede ocurrir que el resultado sea negativo, lo que ocurre cuando el valor absoluto del número con signo negativo es mayor al de signo positivo.

+4.423 -> +4.423
-8.156 ->C10= +1.844
-3.733 6.267 ->C10 =-3.733

Podemos decir como conclusión, que a su vez son reglas a tener en cuenta para operar con complementos:

a) Cada vez que se complementa se cambia de signo.

b) Cuando hay un rebose en el resultado, éste se descarta y lo que queda es lo correcto.

c) Cuando no hay rebose, se debe volver a complementar.

1.B.1.4.1.2 - SUMA ALGEBRAICA CON COMPLEMENTOS A 9:

Las reglas son similares a las anteriores, se complementa el número signo negativo y se suma, luego habrá que ver que ocurre con el rebose o arrastre decimal.

a) Con arrastre decimal: Se suma éste en la cifra menos significativa, y se tiene el resultado, por ejemplo:

```
+ 8.726   ->          + 8.726
- 5.421  -> C9 ->     + 4.578
+ 3.305               +13.304
        Se corrige  +        1
                      + 3.305
```

b) Sin arrastre decimal: Se debe volver a complementar a nueve y cambiar de signo.

```
- 4.738 -> C9  -> + 5.261
+ 2.245   ->      + 2.245
- 2.493           + 7.506 -> C9 -> - 2.493
```

1.B.1.4.1.3 - SUMA ALGEBRAICA ENTRE NÚMEROS BINARIOS:

Al igual que la suma entre números decimales, la suma algebraica entre números binarios se puede realizar mediante complementos, que en este caso será a 2 y a 1.

1.B.1.4.1.3.1 - CON COMPLEMENTO A 2.

Tal como en el caso de los decimales, si hay arrastre se ignora, y si no lo hay se debe complementar nuevamente el resultado.

Ejemplos:

```
+ 1 0 1 0 0   ->    + 1 0 1 0 0
- 0 1 0 0 1 -> C2 ->- 1 0 1 1 1
+ 0 1 0 1 1         1 0 1 0 1 1 -> 1 0 1 1

- 1 0 1 0 1 -> C2 -> + 0 1 0 1 1
+ 1 0 0 1 0   ->    + 1 0 0 1 0
- 0 0 0 1 1         + 1 1 1 0 1 -> C2 -> - 1 1
```

1.B.1.4.1.3.2 - CON COMPLEMENTOS A 1:

Nuevamente se repite el mismo caso de los números decimales, se hay arrastre, se suma a la cifra menos significativa, si no hay, se complementa de nuevo y se cambia de signo.

Ejemplos:

```
+ 1 1 1 0 1      ->      + 1 1 1 0 1
- 0 1 0 1 0 -> C1 ->    + 1 0 1 0 1
+ 1 0 0 1 1              1 1 0 0 1 0
                      +       1
                       + 1 0 0 1 1
```

```
 - 1 0 1 1 0 -> C1 ->     + 0 1 0 0 1
+ 1 0 0 1 0   ->          + 1 0 0 1 0
 - 0 0 1 0 0              + 1 1 0 1 1 -> C1 -> - 1 0 0
```

1.B.1.5 - MULTIPLICACIÓN Y DIVISIÓN:

En este caso haremos un breve repaso de los métodos conocidos para realizar las operaciones citadas, tanto entre decimales, como entre binarios.

1.B.1.5.1 - MULTIPLICACIÓN DECIMAL:

Aparte del normal método de multiplicar y desplazar, que volvemos a mostrar mediante un ejemplo, hay otros dos utilizados en máquinas, el de las componentes derechas e izquierdas, y el de las sumas repetidas.

1.B.1.5.1.1 - MÉTODO NORMAL:

Solo haré el ejemplo, indicando los nombres dados a cada elemento:

```
  8543        Multiplicando
 x 234        Multiplicador
 34172        1er producto parcial
25629         2do producto parcial
17086         3er producto parcial
1999062       Producto
```

1.B.1.5.1.2 - MÉTODOS DE LAS COMPONENTES DERECHAS E IZQUIERDAS:

Cuando multiplicamos como en la forma anterior, obtenemos por lo general algún arrastre, en éste método se trata de anotar los arrastres aparte de los productos parciales, y sumar entre sí los productos parciales sin arrastres y los arrastres, para luego sumar los resultados conjuntamente.

```
Ejemplo:                  8543
                          x 234
                  3211    2062
Componentes       2110    4529    Componentes
Izquierdas        1100    6086    derechas
                  134311  655952
                          134311
                          1999062
```

1.B.1.5.1.3 - MULTIPLICACIÓN POR SUMAS REPETIDAS:

Es el método mas primitivo, utilizado desde las primeras calculadoras mecánicas, se trata de sumar el multiplicando tantas veces como el diga el multiplicador, en la siguiente forma:

```
  8543
  8543
  8543
  8543  4        =   4
 85430
 85430
 85430  3 x 10   =  30
854300
 854300  2 x 100 = 200
1999062            234
```

1.B.1.5.2 - MULTIPLICACIÓN BINARIA:

Aplicando las correspondientes tablas, que podemos aquí repetir en la siguiente forma:

```
x 0 1
0 0 0
1 0 1
```

De donde leemos: 0x0=0; 0x1=0; 1x0=0; 1x1=1, y no hay arrastres, por lo que ese método no es aplicable.

1.B.1.5.2.1 - MÉTODO NORMAL:

Partiremos con el ejemplo:

```
      1 0 1 1      11
    x 0 1 1 0       6
    0 0 0 0
  1 0 1 1
1 0 1 1
0 0 0 0
1 0 0 0 0 1 0      66
```

Es tan sencillo que no precisa ningún comentario.

1.B.1.5.2.2 - POR SUMAS REPETIDAS:

En realidad no hay tales sumas, dado que solo se suman una vez o nada, o sea que es directamente como el anterior, si en el multiplicador hay un 1, el número se suma y luego se desplaza, si hay un cero solamente se desplaza.

1.B.1.5.3 - DIVISIÓN DECIMAL:

También aquí tenemos dos métodos, el común, ya conocido por todos y el por restas sucesivas, que se puede hacer de dos formas, la restaurativa y la no restaurativa.

1.B.1.5.3.1 - MÉTODO COMÚN:

Consiste en probar cuantas veces cabe el dividendo en una porción del divisor, haremos aquí un ejemplo con el solo propósito de recordar los nombres dados a los diversos elementos.

Dividendo	**452980**	**|365**	**Divisor**
1er resto parcial	**879**	**1241**	**Cociente**
2do resto parcial	**1498**		
3er resto parcial	**380**		
Resto	**15**		

1.B.1.5.3.2 - DIVISIÓN POR RESTAS SUCESIVAS, MÉTODO RESTAURATIVO:

En este caso se alinean por la izquierda el dividendo y el divisor, y se resta hasta que aparece un resto negativo, allí se procede a sumar o reponer (restaurar) el resto anterior, se desplaza el divisor una posición a la derecha y se vuelve a repetir el proceso. La cantidad de veces que se restó sin que aparezca el resto negativo es el dígito correspondiente del cociente.

Ejemplo:

452980 |365
-365000
87980
-365000
-277020 Resto negativo, se debe
+365000 reponer. Cantidad de
87980 restas = 2 - 1 = 1
- 36500 Luego se debe desplazar
51480 el dividendo y restar.
- 36500
14980
- 36500
- 78480 Resto negativo, se debe
+ 36500 reponer. Cantidad de
14980 restas = 3 - 1 = 2
- 3650 Se desplaza nuevamente
11330 y se vuelve a restar.
- 3650
7680
- 3650
4030
- 3650
380

- 3650
- 3270 Resto negativo, se debe
+ 3650 reponer. Cantidad de
380 restas = 5 - 1 = 4
- 365 Se desplaza y se resta.
15
- 365
- 350 Resto negativo, reponer
+ 365 Cantidad de restas =
15 2 - 1 = 1

En consecuencia el resultado es: Cociente = 1241
Resto = 15

Si seguimos dividiendo, hallaremos los decimales, tantos como deseemos, si bien lo normal en operaciones contables es de dos.

1.B.1.5.3.3 - MÉTODO NO RESTAURATIVO:

Ahora, la reposición se reemplaza por la suma del divisor desplazado una posición a la derecha, y el dígito del cociente corresponderá al complemento a diez del número de sumas para llegar al primer resto no negativo. Las restas se reemplazan por la suma del divisor

Ejemplo:

452980 |365
- 365000
87980
-365000
-277020 Restas 2 - 1 = 1 (Primer dígito del cociente)
+ 36500 Se desplaza el divisor y se suma
-240520
+ 36500
-204020
+ 36500
-167520
+ 36500
-131020
+ 36500
- 94520
+ 36500
- 58020
+ 36500
- 21520
+ 36500 Sumas 8, el complemento a 10 de 8 = 2 (Segundo dígito)
+ 14980 Ahora hay resto positivo, por lo que se desplaza y se resta

- 3650
+11330
- 3650
+ 7680

- 3650
+ 4030
- 3650
+ 380
- 3650 **Restas 5 -1 = 4 (Tercer dígito del cociente)**
- 2540
+ 365 **Se desplaza y se suma nuevamente.**
- 2175
+ 365
- 1810
+ 365
- 1445
+ 365
- 1080
+ 365
- 715
+ 365
- 350
+ 365 **Sumas 7, Complemento a 10 de 7 = 3 (Cuarto dígito)**
+ 15

Hemos llegado al final, a menos que deseemos tener parte fraccionaria.

El cuarto dígito del cociente será entonces el complemento a diez de 7, o sea 3, por lo que el cociente total será: 1243

Lo mismo puede hacerse empleando complementos, y entonces en vez de restar, se sumarán. Se debe tener cuidado en aplicar correctamente las reglas de la suma de complementos.

1.B.1.5.4 - DIVISIÓN BINARIA:

Es tan sencilla como la multiplicación binaria, y puede hacerse también por restas repetidas, que no son tales, pues a lo suma será una o ninguna.

Ejemplo:

```
1011010 |1110      90 |14
 1110     110       -6  6
 100010
  1110
  00110
```

1.B.1.6 - NOTACIÓN EN COMPLEMENTO A DOS:

Cuando se opera con números con signo, se presentan dos inconvenientes, la suma algebraica es altamente ineficiente, según veremos más adelante, y el cero tiene dos representaciones, +0 y -0.

En la notación en complemento a dos, que consiste en incorporar el bit de signo al número, en forma tal que:

$$N = -2^{n-1} a_{n-1} + \sum_{n=0}^{n-2} 2^i a_i$$

En la cual para enteros positivos es a_{n-1}=0, por lo que el término:

$$-2^{n-1} a_{n-1} = 0$$

y corresponderá al primer dígito del número considerado. Por otra parte, cuando a_{n-1} = 1, el término considerado valdrá:

$$-2^{n-1} a_{n+1} = -2^{n-1}$$

En consecuencia, cuando el número es negativo, la primera cifra es un 1.

Esto también resulta práctico para diferenciar los números positivos de los negativos.

Decimal	*en magnitud y signo*	*en complemento a dos*
+7	**0111**	**0111**
+6	**0110**	**0110**
+5	**0101**	**0101**
+4	**0100**	**0100**
+3	**0011**	**0011**
+2	**0010**	**0010**
+1	**0001**	**0001**
+0	**0000**	**0000**
-0	**1000**	**----**
-1	**1001**	**1111**
-2	**1010**	**1110**
-3	**1011**	**1101**
-4	**1100**	**1100**
-5	**1101**	**1011**
-6	**1110**	**1010**
-7	**1111**	**1001**
-8	**----**	**1000**

Como vemos, la formación de los números negativos, según la expresión anterior es lograda tomando el primer bit con su valor negativo, y restandole los que le siguen, de acuerdo a su valor.

Además, si tomamos directamente el complemento a dos según vimos antes, tenemos exactamente lo mismo.

Otra forma para determinar dicho complemento, es tomar un sistema semigráfico, con tantas casillas como dígitos se deseen para hacerlo. Por ejemplo, supongamos el complemento a dos de 7, lo cual sería expresar -7, pero en un sistema de ocho bits, tendríamos:

-128	64	32	16	8	4	2	1

-128	64	32	16	8	4	2	1
1	0	0	1	0	0	1	1

-128 +16 +2 +1 = -109

-128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	0

-128 +8 = -120

Tabla I.1 - Caja para determinar el valor de los números en complemento a dos.

1.B.1.6.1 - PRODUCTO DE NÚMEROS EN NOTACIÓN COMPLEMENTO A DOS.

Se presentan inconvenientes cuando el multiplicador es negativo y presentado como complemento a dos. En el producto, se requiere que los productos parciales sean negativos, para que el resultado también lo sea. Veamos un ejemplo:

```
    0110        =  6
  x 1001        = -7
    0110
 0110000
 0110110        = 54
```

lo que es incorrecto.

Una de las posibles soluciones, es la de hacer positivos a ambos números, complementando (en éste caso descomplementando al negativo).

```
    0110
  x 0111
    0110
   0110
  0110
 0000
 0101010    = 42
```

Pero, el mismo debe ser complementado nuevamente por cuando debe ser negativo, con lo cual tendremos:

```
 1   0   1   0   1   1   0
-64  +16     +4  +2      = -42
```

Sin embargo, sería preferible utilizar alguna técnica que no requiera de la última transformación, para lo cual puede aplicarse el denominado método de Booth, que consiste en llenar con unos todos los espacios de los registros de los productos parciales, que deben ser de longitud doble a la de los multiplicandos.

Por ejemplo:

```
          1 0 0 1 = -7
        x 0 0 1 1 =  3
  ---------------
1 1 1 1 1 0 0 1 = -7
1 1 1 1 1 0 0 1 0 =-14
1 1 1 1 0 1 0 1 1  = -128+64+32+8+2+=-21
```

De la misma forma se puede actuar con la división, obteniendo así un método práctico para realizar operaciones en complemento a dos.

1.B.2 – IMPLEMENTACIÓN DE LAS OPERACIONES LÓGICAS Y ARITMÉTICAS

1.B.2.1 - INTRODUCCION:

Según vimos en el capitulo I (tema A), la Unidad Lógica y Aritmética es la encargada de realizar todas las operaciones de ese tipo sobre los operandos, para ello, según lo mostrado en la figura I.5, consta de circuitos de cálculo y registros.

En este item analizaremos como se interconectan los circuitos de cálculo a fin de realizar las diferentes operaciones tanto en binario como en decimal codificado en binario (BCD).

Los circuitos que veremos son de tipo funcional y no constructivo, dado que tienen por objeto detallar la forma en que pueden realizarse las principales operaciones aritméticas y lógicas. También acompañaré los diagramas de flujo correspondientes a las operaciones indicadas.

1.B.2.2 - COMPLEMENTADORES:

Se denomina complemento de un número a la cantidad que le falta a ese número para llegar al módulo, siendo EL MÓDULO la mayor de las cantidades expresable con esa cantidad de dígitos, más una unidad.

A fin de simplificar, y no tener que realizar operaciones con números de longitud variable, definiendo en cada caso el modulo correspondiente, se recurre a un módulo restringido, cifra por cifra, y en el caso de los binarios tenemos el módulo a uno o a dos, según que la cifra menos significativa, se reste de uno o de dos. En caso de los números decimales, tendremos, de igual modo, el complemento a nueve y el complemento a diez.

Al calcular los complementos, en el caso de los binarios, el complemento a dos es igual al complemento a uno más uno, y en el caso de los decimales, el complemento a diez es igual al complemento a nueve más uno.

1.B.2.2.1 - COMPLEMENTADORES BINARIOS:

El más simple es el serie, que emplea un inversor y un registro de desplazamiento, tal como en la figura 1.B.3.a, mientras que en la figura 1.B.3.b se tiene el complementador paralelo.

De cualquier manera, recordemos que en la implementación práctica de los registros, se utilizan multivibradores, los cuales tienen dos salidas, la normal y la complementaria, por tanto, para hallar el complemento a uno de un número binario, solamente es necesario obtener las salidas complementarias de los multivibradores, y así se lo muestra en 1.B.3.c.

1.B.2.2.2 - COMPLEMENTADOR DECIMAL:

En este caso, y según dijéramos antes, tomaremos solamente el complementador a nueve, que consiste en restar cada dígito de nueve, dado que el complemento a diez será el mismo más una unidad.

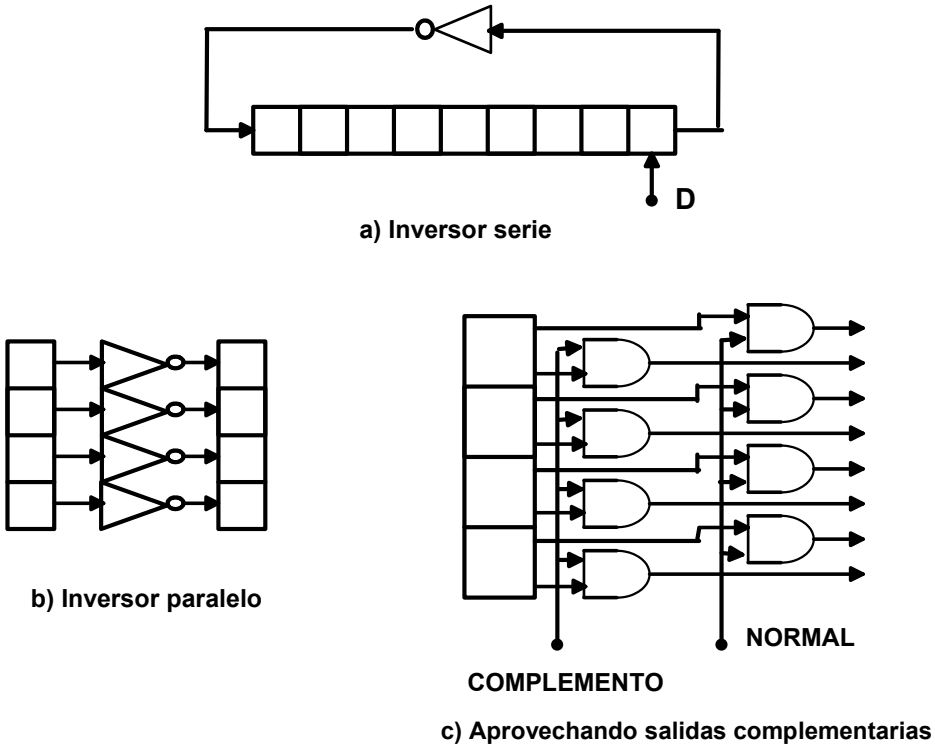


Figura 1.B.3- Complementadores Binarios.

En primer lugar, haremos una tabla de la verdad para hallar las funciones necesarias, luego minimizaremos y finalmente diagramaremos el circuito, todo ello es lo que se muestra en la figura 1.B.4.

Recordemos que estos complementos son útiles para la realización de cálculos, dado que permiten simplificar las operaciones de resta y de división.

1.B.2.3 - COMPARADORES:

Otra de las funciones necesarias en una máquina es la comparación de cantidades para saber si son iguales.

Directamente una compuerta que permita das salida cuando ambos dígitos de entrada son iguales, por ejemplo una NOR-EXCLUSIIA, que en definitiva es representada por el circuito de la figura 1.B.5.

La comparación entre números decimales se reduce, a la comparación de códigos binarios, por tanto será cuestión de disponer varios comparadores binarios en paralelo.

Dec	B_3	B_2	B_1	B_0	C_3	C_2	C_1	C_0	Dec
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	9
1	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	8
2	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	7
3	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	6
4	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	5
5	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	4
6	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	3
7	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	2
8	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	1
9	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	0

a) Tabla de la verdad del complementador decimal.

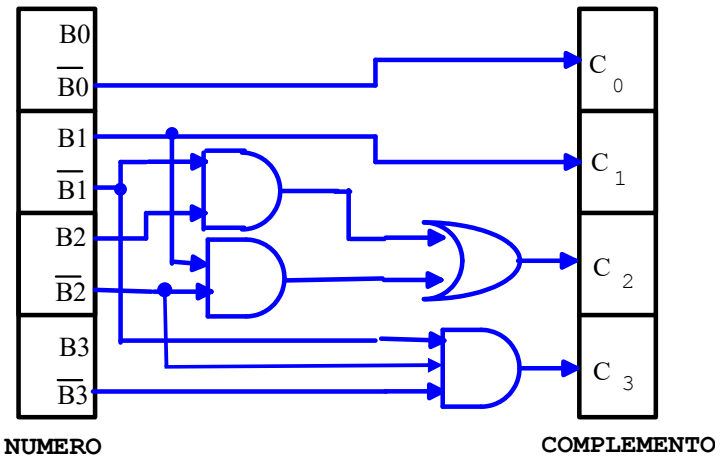
$C_0 = 0,2,4,6,8 + TNV$ **$C_0 = \overline{B}_0$** **$C_1 = B_1$**

$C_1 = 2,3,6,7 + TNV$

$C_2 = 2,3,4,5 + TNV$ **$C2 = \overline{B}_1 B_2 + B_1 \overline{B}_2$** **$C_3 = \overline{B}_1 \overline{B}_2 \overline{B}_3$**

$C_3 = 0,1 + TNV$

b) Funciones y su minimización.



c) Circuito

Figura 1.B.4- Complementador decimal.

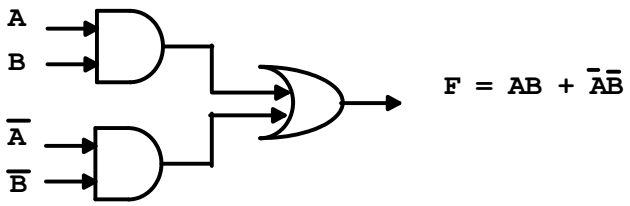


Figura 1.B.5- Comparador binario.

1.B.2.4 - SUMADOR BINARIO:

El circuito que sigue en orden de complejidad, es el semisumador binario, que es el complementario del comparador binario, pues recordemos que en la suma de dos dígitos binarios ocurre lo siguiente:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1 0

Dicho en otros términos, cuando los dígitos son diferentes hay salida, mientras que no la hay cuando son iguales, debiendo tener en cuenta que cuando ambos valen 1, el resultado es 10.

En la figura 1.B.6 se tiene el circuito que cumple con la primera parte, que es denominado SEMISUMADOR, mientras que en la figura 1.B.7, se tiene el circuito del un SUMADOR COMPLETO, dado que se detecta la presencia de los dos unos para indicarlo en una salida particular, denominada de ARRASTRE.

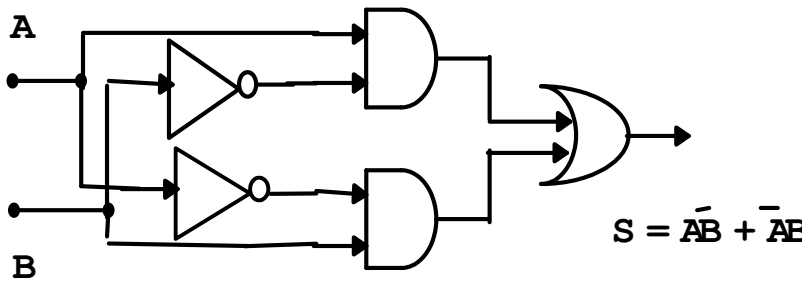


Figura 1.B.6- Semisumador.

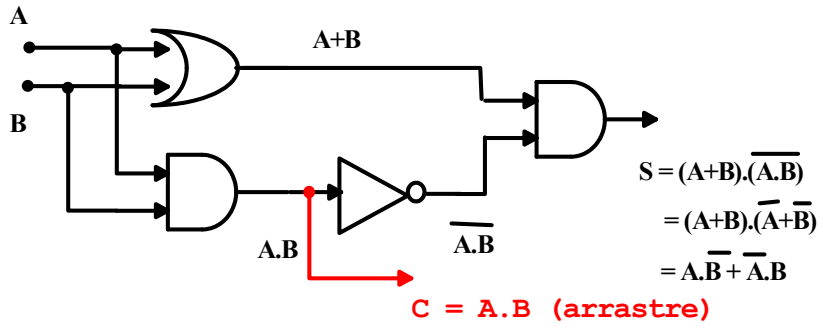


Figura 1.B.7- Semisumador modificado para dar el arraste.

Otra forma de implementar el sumador completo, es a través de semisumadores tales como el indicado en la figura 1.B.6, quedando así el circuito de la figura 1.B.8.

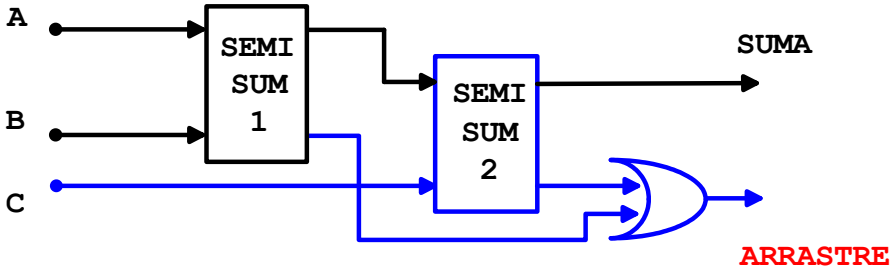


Figura 1.B.8- Sumador completo.

Debemos ver que en todos estos circuitos de sumadores completos, los niveles son más de dos, por lo que tienen retardos mayores que en los diseños tradicionales.

En la figura 1.B.9 se muestra un sumador completo realizado mediante el método tradicional de diseño, que contempla solo dos niveles de compuertas.

Es de notar en este último caso que se han seguido todos los pasos, planteo del problema, obtención de las funciones, minimización, obtención de las funciones minimizadas y trazado del esquema del circuito.

1.B.2.5 - MECANIZACION DE LA SUMA ENTRE NUMEROS BINARIOS:

En este caso se trata de máquinas que operan en binario, por lo que pueden presentarse dos casos, de suma serie y de suma paralelo. La primera es más económica en circuitos, necesitando más tiempo de ejecución, mientras que la segunda es más cara en circuitos, pero muchísimo más rápida en su ejecución.

1	1	1	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1
0	1	0	1	0
0	0	1	1	0
0	0	0	0	0
A	B	C	S	R

$R = 3,5,6,7 = AB+BC+AC$

$S = 1,2,4,7 = \overline{A}\overline{B}C+\overline{A}B\overline{C}+A\overline{B}\overline{C}+ABC$

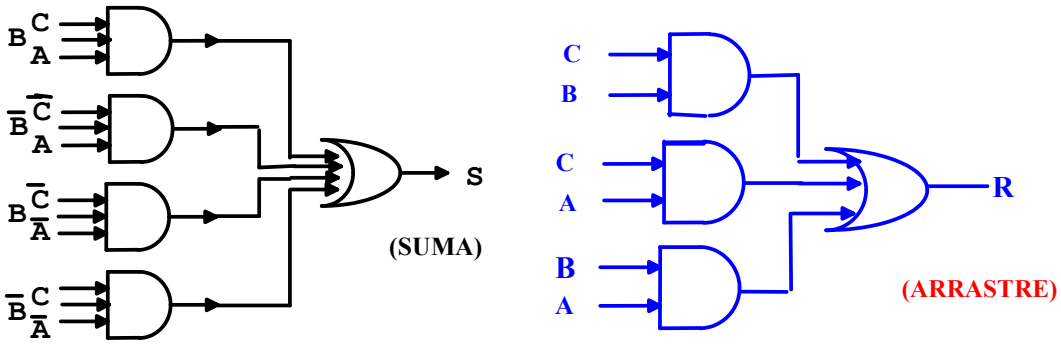


Figura 1.B.9 - Sumador completo, realizado por método tradicional.

II.5.1 - SUMA SERIE:

En la figura 1.B.11, se tiene al circuito de este tipo de sumador, donde vemos que consta de tres registros, un sumador completo y un elemento de retardo para aplicar el arrastre en la próxima suma.

En la figura 1.B.10, se tiene el diagrama de flujo a cumplir para realizar la suma de dos números binarios.

En el registro A se dispone un sumando, en el registro B, otro, y en cada pulso de desplazamiento, los bits menos significativos se suman, conjuntamente con el arrastre de la suma anterior, para ir quedando los resultados en el registro C, entrando por el bit más significativo.

Es evidente que a medida que los pulsos de desplazamiento van actuando, se van liberando desde la izquierda los registros A y B, y viceversa, se va ocupando desde la izquierda el registro C, por tanto es posible reemplazar a este último por alguno de los primeros. El registro donde se van acumulando los resultados se denomina REGISTRO ACUMULADOR o directamente ACUMULADOR.

Muchas veces es necesario realizar operaciones encadenadas, para lo cual uno de los sumandos debe ser conservado, ello se logra mediante realimentación del otro registro, el B. En la figura 1.B.12, se tiene este caso.

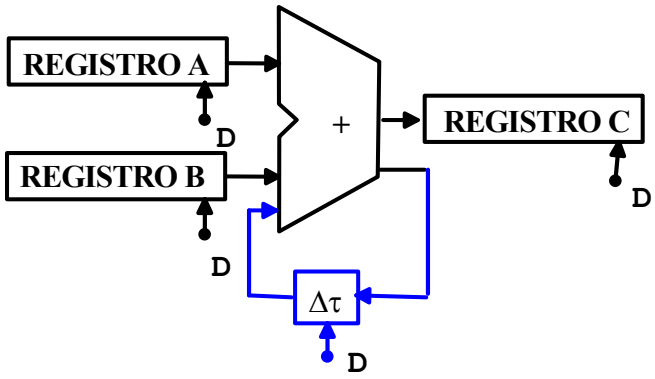


Figura 1.B.11- Sumador serie sin acumulador.

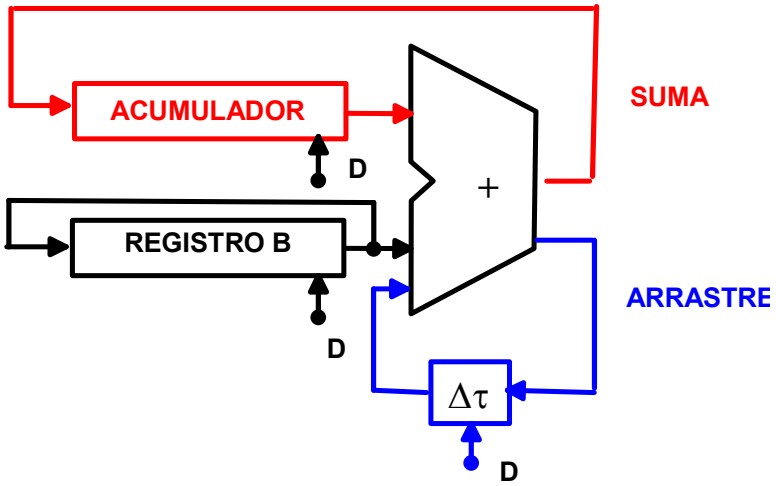


Figura 1.B.12- Sumador con Acumulador.

1.B.2.5.2 - SUMA PARALELO:

Es posible agrupar diversos sumadores completos entre dos registros, a fin de que la suma se realice en paralelo, solamente debe tenerse en cuenta un detalle, el de la propagación del arrastre, pues una suma no será correcta hasta que el arrastre que se agregue, también sea correcto. Lógicamente esto va introduciendo retardos en las señales, los cuales pueden, aparte de aumentar el tiempo de ejecución, dar lugar a errores de lectura.

En la figura 1.B.13 se tiene el circuito propuesto para este tipo de sumador.

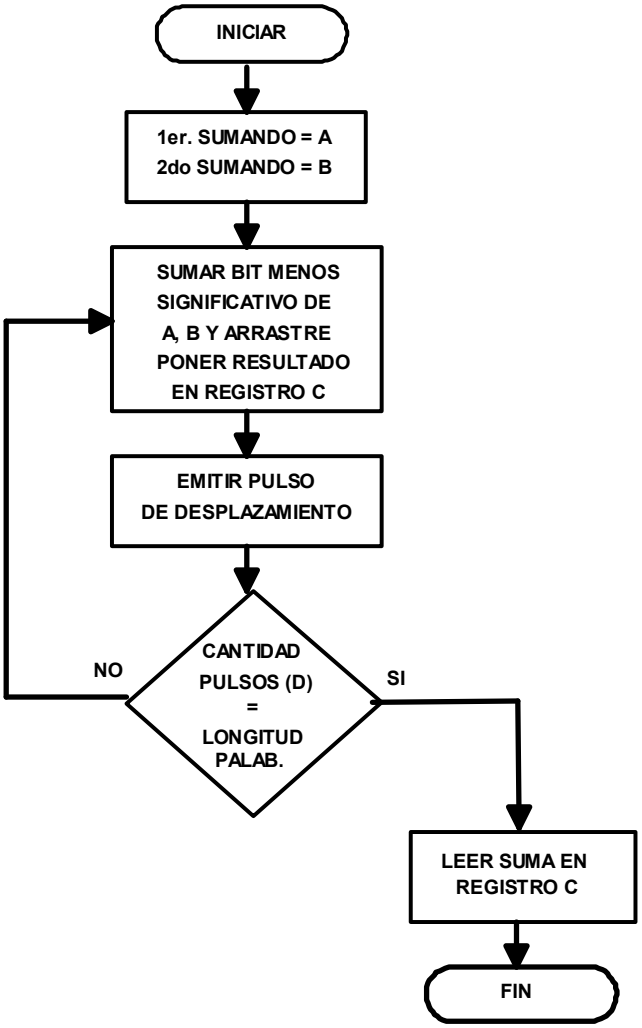


Figura 1.B.10- Diagrama de Flujo para la suma serie binaria.

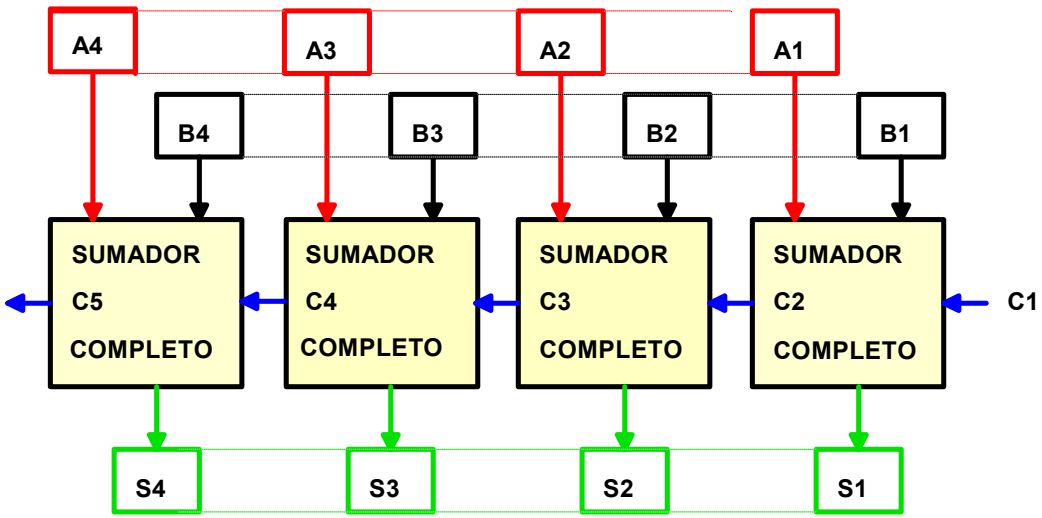


Figura 1.B.13- Sumador paralelo.

1.B.2.5.2.1 - SUMADOR PARALELO CON ANTICIPO DEL ARRASTRE:

Según dejara establecido antes, para acelerar la suma, es necesario de alguna manera tratar que el arrastre no demore su cálculo, una forma, es anticiparlo en bloques cortos de sumadores, por ejemplo de cuatro unidades, de esta manera, el retardo siempre será solamente el correspondiente a la determinación de tres arrastres, pues el cuarto es dado mediante un circuito combinacional.

En la figura 1.B.14 tenemos el caso indicado, donde suponemos qu el arrastre deja de influir en la demora para obtener los resultados, evitando errores de lectura en los mismos.

1.B.2.6 - MULTIPLICACION DE NUMEROS BINARIOS.

La multiplicación entre números binarios, puede hacerse fundamentalmente de tres maneras, por sumas sucesivas, acelerada por suma y desplazamiento y por un sistema matricial o celular.

1.B.2.6.1 - MULTIPLICACION BINARIA POR SUMAS SUCESIVAS:

Es la más elemental de las formas, y consiste en sumar tantas veces el multiplicando como lo diga el multiplicador. Una manera sencilla de llevarla a cabo es provocar la suma del acumulador y el registro B, previo almacenamiento en él del multiplicando en éste y del multiplicador en un registro auxiliar, luego se cuenta el número de sumas que se va almacenando en un contador, y se lo compara con el registro auxiliar, cuando ambos contenidos sean iguales, se detendrá el proceso.

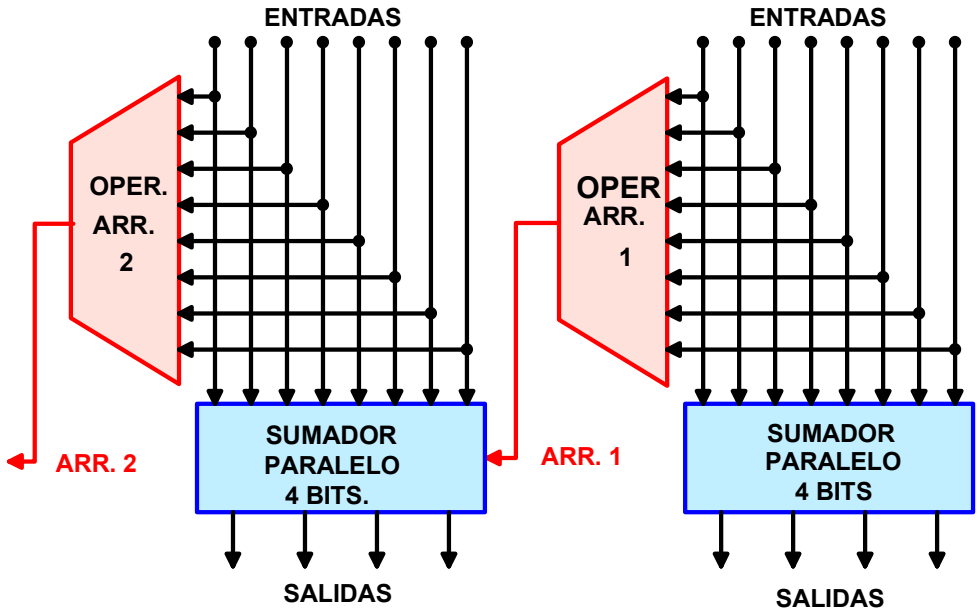


Figura 1.B.14 - Suma paralelo con anticipo del arrastre.

En la figura 1.B.15 se tiene el diagrama de flujo que se debe cumplir, y en la figura 1.B.16 el esquema del circuito que lo puede llevar a cabo.

El divisor en encargado de emitir un pulso cada vez que se realiza una suma, o sea cada vez que transcurre un tiempo palabra, o dicho de otra manera cada vez que transcurren tantos pulsos como posiciones binarias tenga un registro. De esta manera el contador luego contará solamente la cantidad de sumas, y cuando esta sea igual al contenido del registro auxiliar, el comparador debiera emitir una señal que producirá la detención del proceso.

1.B.2.6.2 - MULTIPLICACION ACELERADA:

En este caso se busca que cuando haya un 1 en el multiplicador el multiplicando se sume y se desplace, mientras que cuando haya un cero se desplace solamente.

En la figura 1.B.17, se tiene el diagrama de flujo de este tipo de multiplicador, mientras que en la figura 1.B.18, se tiene el esquema adecuado.

Obsérvese que el registro A, para el resultado, tiene doble longitud, lo cual es lógico pues el producto de dos cantidades tiene una longitud igual al doble de la de cada uno de los operandos.

La operación es como sigue: Cuando el bit menos significatIIo del contenido del registro B es "1", la acción provocada es la suma de <A> y <B>, guardando el resultado en A, y desplazando luego una posición a la derecha al registro B, A y A', al mismo tiempo que el contador cuenta el desplazamiento habido.

Ello es lo que se observa en las figuras correspondientes a T=0 que es la posición inicial, T=1, la suma y T=2 el desplazamiento. Luego, para T=3, dado que en B ahora se presenta un "0", se deberá desplazar únicamente a los registros antes citados y aumentar en una unidad el contenido del contador.

El proceso sigue igual, hasta que se produce el desplazamiento número 4, puesto que esa es la cantidad de bits de los registros, y en ese momento se tendrá el resultado leyendo conjuntamente el contenido de A y A'.

Los datos iniciales en decimal eran: 11 x 9 = 99, el resultado es 01100011, lo que significa $1x2^6 + 1x2^5 + 1x2^1 + 1x2^0 = 64 + 32 + 2 + 1 = 99$, por lo hemos actuado correctamente.

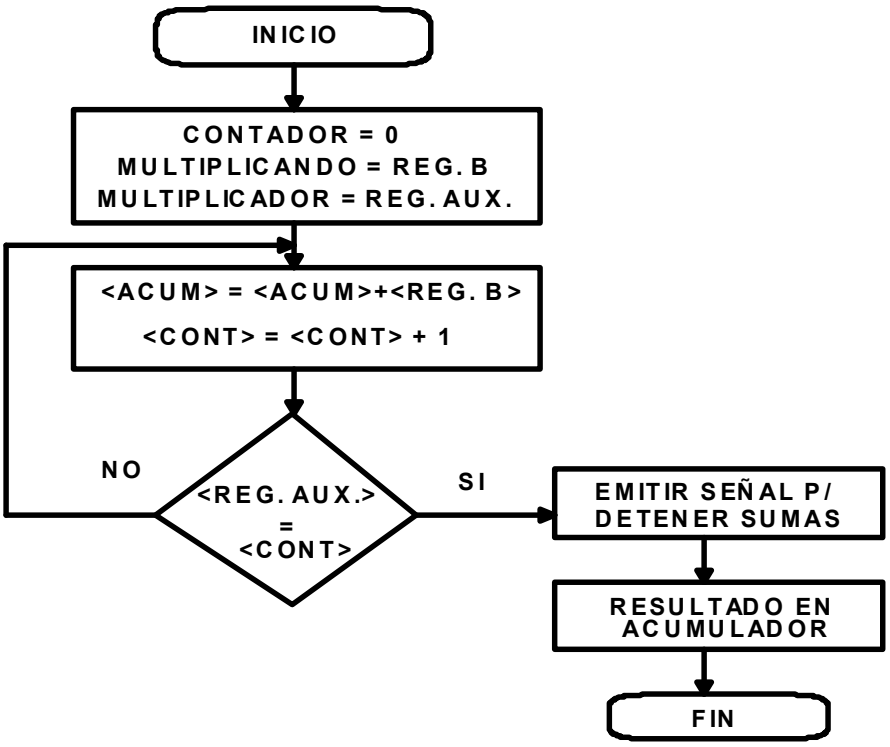


Figura 1.B.15 - Diagrama de flujo de la multiplicación por sumas sucesivas.

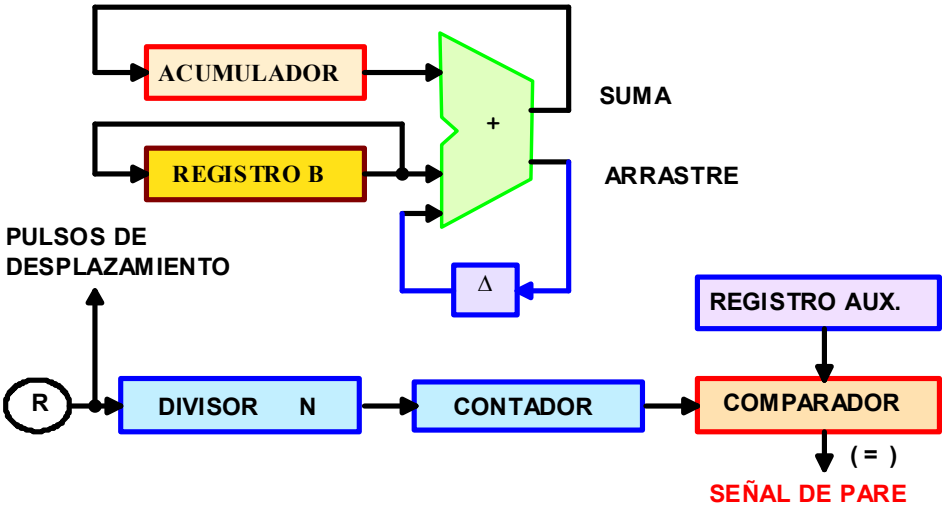


Figura 1.B.16 - Esquema para la multiplicación binaria por sumas sucesivas.

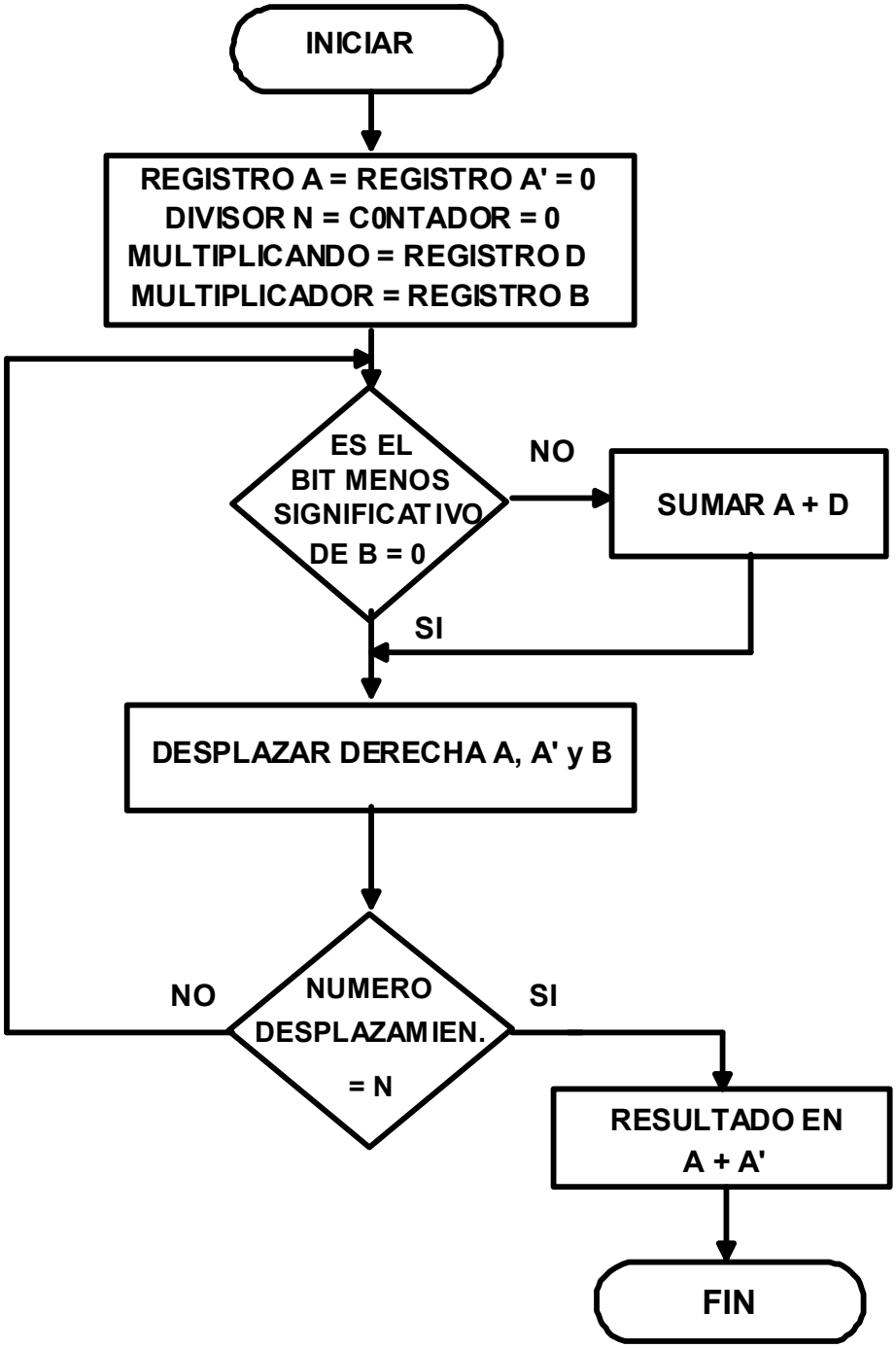


Figura 1.B.17 - Diagrama de flujo de la multiplicación binaria acelerada.

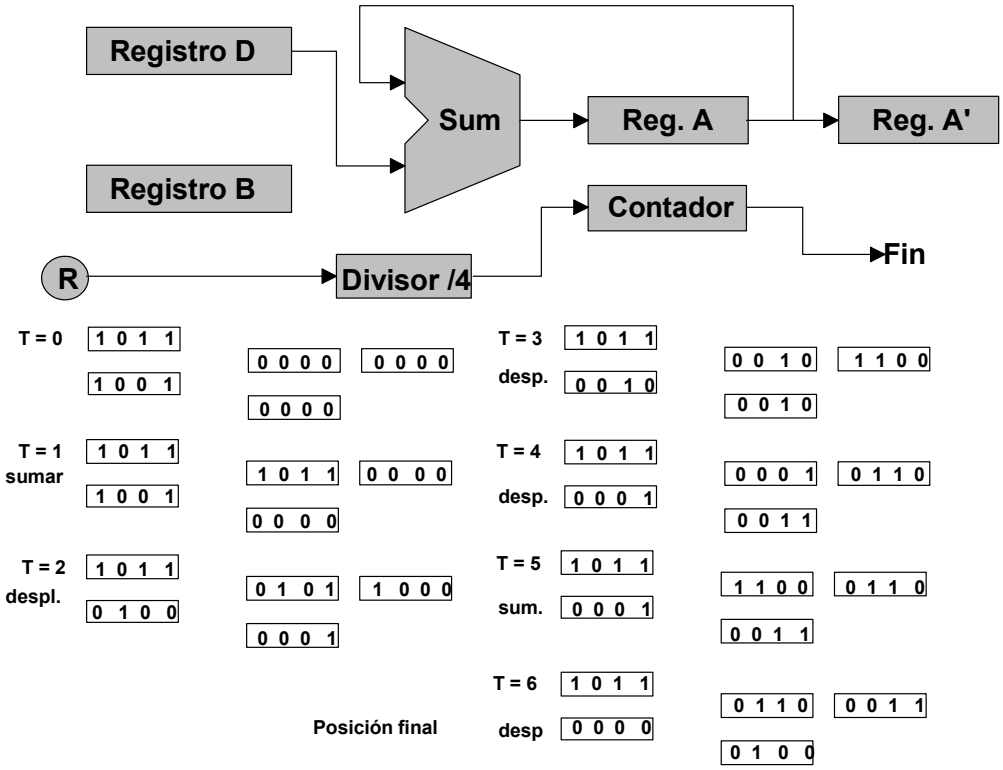


Figura 1.B.18 - Multiplicador binario por suma y desplazamiento.

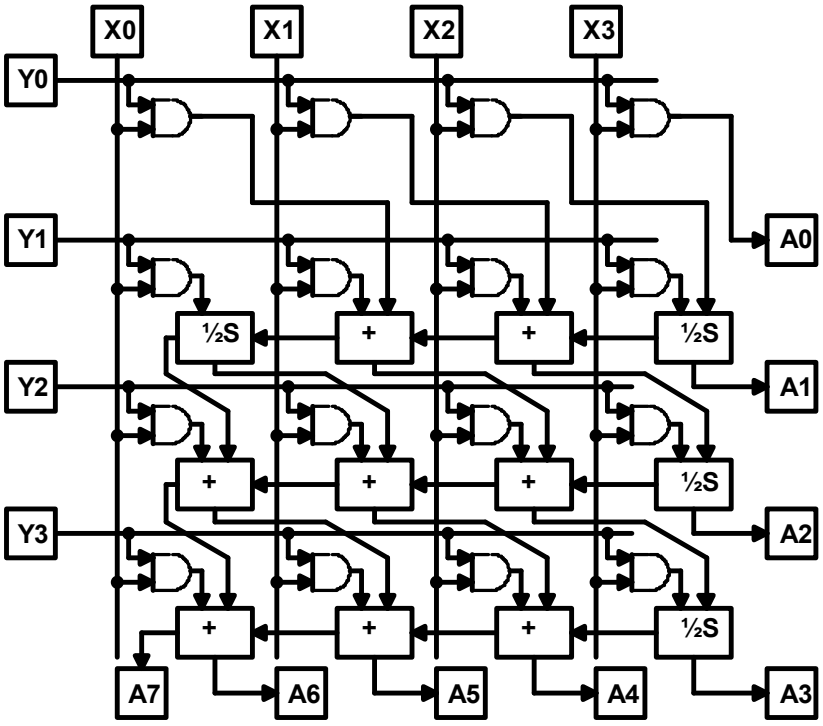


Figura 1.B.19 - Multiplicador celular.

1.B.2.6.3 - MULTIPLICADOR CELULAR:

En la figura 1.B.19, tenemos un multiplicador paralelo celular, en el cual se obtiene el producto final directamente en las salidas, con las demoras introducidas por los circuitos solamente.

Como puede verse, el sistema se basa en sumas y desplazamientos, que son realizados todos por hardware, por lo que resulta bastante mas costoso que los anteriores, solo se justifica en caso de necesitar elevadísimas velocidades de cálculo.

1.B.2.7 - MECANISMOS PARA LA DIVISION ENTRE NUMEROS BINARIOS:

La división entre números binarios, se realiza en las mismas formas que la multiplicación, puede ser por restas sucesivas, por resta y desplazamiento y mediante un sistema celular. Sabemos que en algunos casos se puede reemplazar la resta por la suma del complemento, pero en esencia el proceso es el mismo.

1.B.2.7.1 - DIVISION BINARIA POR RESTAS SUCESIIAS:

El diagrama de flujo representativo del algoritmo que permite efectuar la división por restas sucesivas, es el mostrado en la figura 1.B.20(II.17), que como vemos es muy parecido al de la multiplicación por sumas sucesivas, pues se trata de restar el divisor del dividendo tantas veces como se pueda, contando el número de restas, lo que será el cociente, o sea el resultado de la división.

En nuestro caso, la operación deberá detenerse cuando desaparezca el arrastre, lo cual es llevado a cabo mediante una compuerta, tal como se muestra en la figura 1.B.21, que representa el esquema del circuito necesario para llevar a cabo la operación.

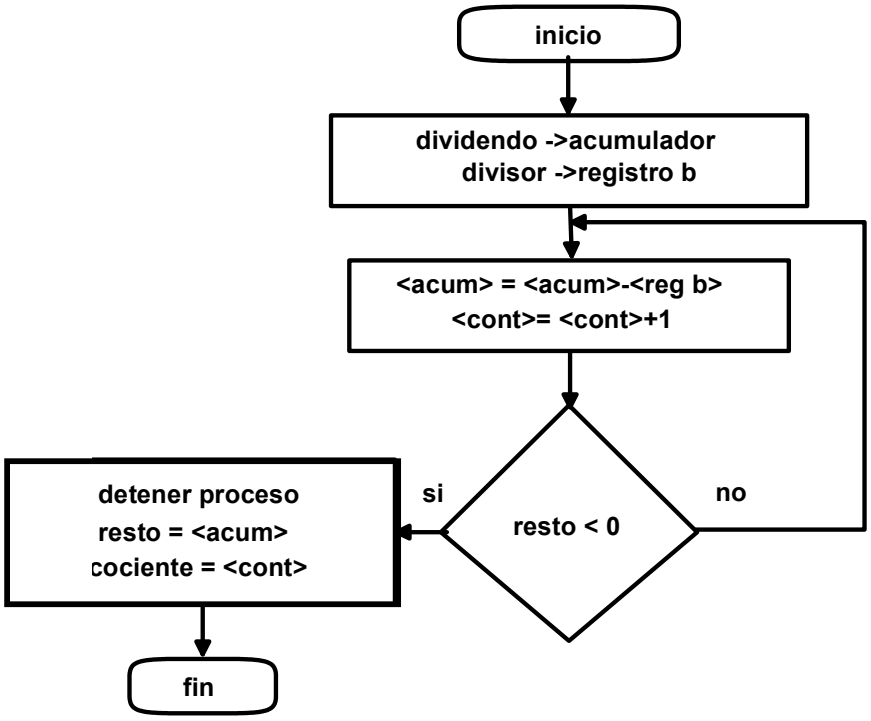


Figura II.17 - Diagrama de flujo de la División por restas sucesivas.

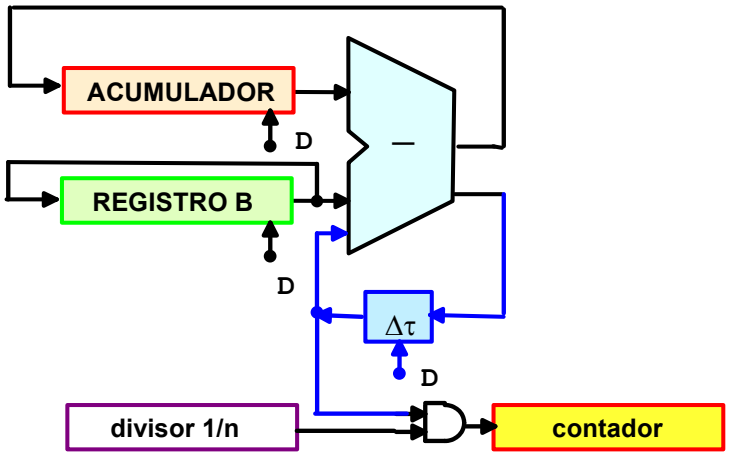


Figura 1.B.21- Divisor por restas sucesivas.

1.B.2.7.2 -DIVISION ACELERADA ENTRE NUMEROS BINARIOS:

Ahora se trata de restar y desplazar, para ello en primer lugar hará falta un registro para el dividendo de doble longitud, y se desplazará al mismo totalmente a la izquierda para comenzar. Luego se prueba si es posible efectuar la resta, o sea si el contenido del registro acumulador es mayor que el contenido del registro B, estando alineados por su dígito mas significativo.

En caso de que ocurra así, puede realizarse la resta, y el primer dígito del cociente será un 1, si no puede restarse, directamente se desplaza a la izquierda el

contenido del acumulador y del cociente, y se vuelve a empezar, en este caso se pondrá un 0 en el lugar correspondiente del cociente.

El proceso sigue hasta que la cantidad de desplazamientos sea igual a un número prefijado, donde se dará por terminada la operación, en el acumulador se leerá el resto, en el registro M, se leerá el cociente.

En general, la cantidad de desplazamientos N, será igual a la longitud deseada del cociente.

En la figura 1.B.22 se tiene el diagrama de flujo y en la 1.B.23 el esquema del circuito.

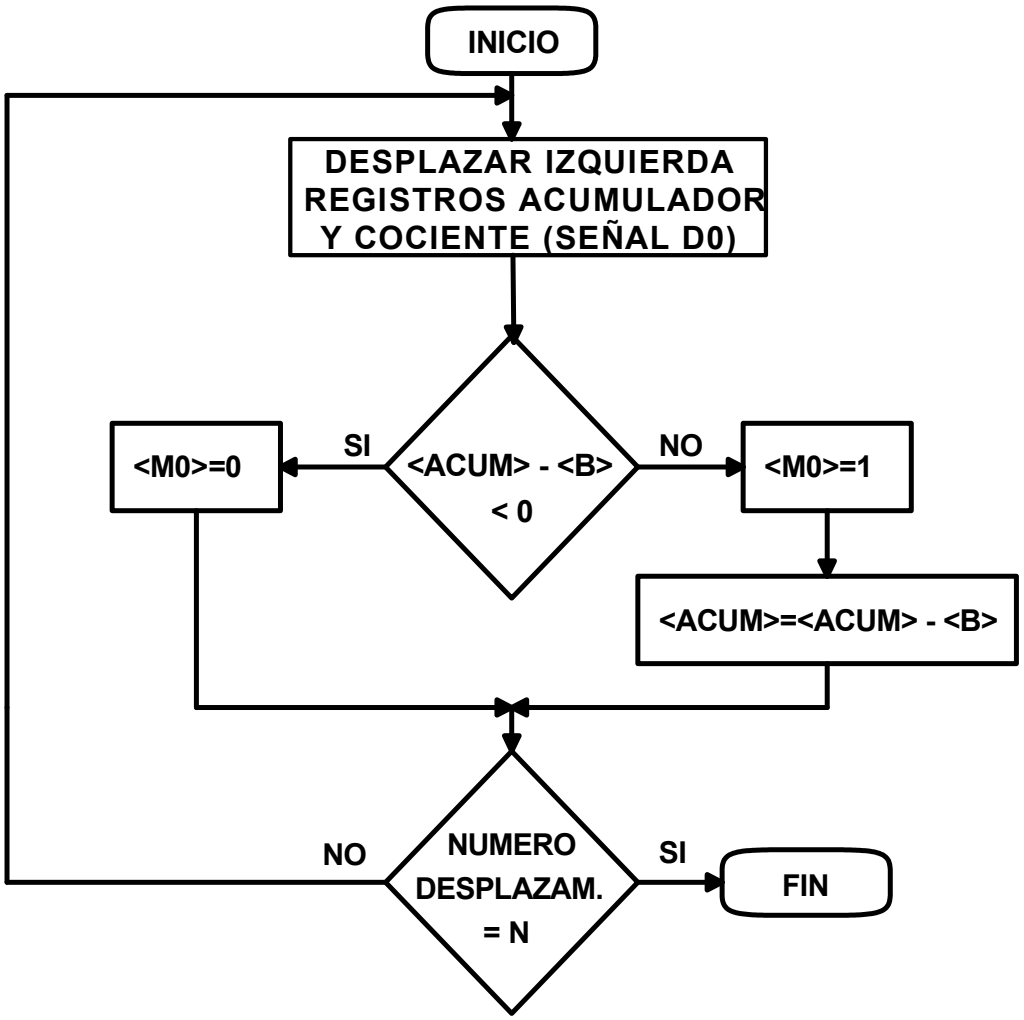


Figura 1.B.22- Diagrama de flujo para la División binaria acelerada.

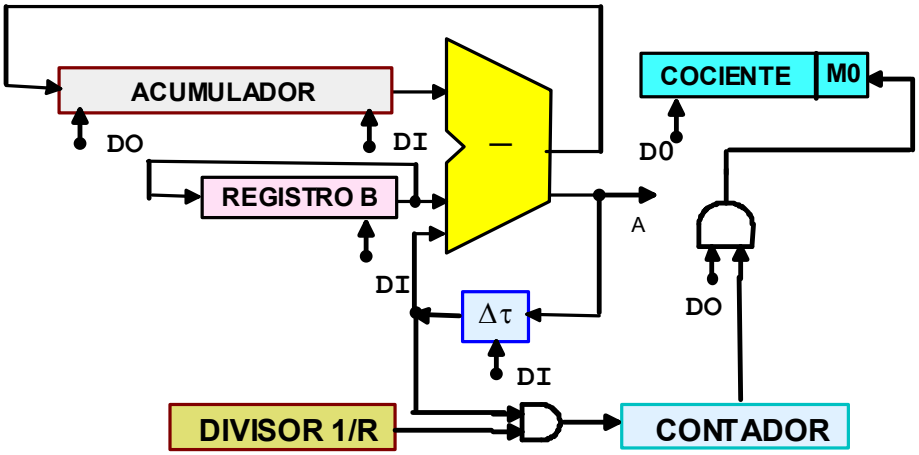


Figura 1.B.23 - Circuito para la mecanización de la división acelerada.

1.B.2.7.3 - DIVISOR CELULAR:

Tal como en el caso de la multiplicación, también es posible realizar la división por restas y desplazamiento mediante un sistema celular, o sea casi totalmente paralelo.

1.B.2.8 - OPERACIONES ENTRE NÚMEROS CODIFICADOS EN BINARIO:

Toca ahora el turno a las cantidades decimales, que empero están expresadas en binario, en el código más común que es el BCD.

Recordemos también que la representación normal en computadora es mediante el código ASCII, el que en su versión completa consiste en un grupo de ocho bits por cada caracter, sin embargo, para las cifras solamente son necesarios cuatro bits, que son los últimos, utilizando los primeros cuatro para indicar en que campo de la representación cae.

Para las cifras se emplean los binarios que representan los hexadecimales 30 a 39, mientras que para las letras va del 41 para la A al 5A para la Z y del 61 para la a al 7A para la z.

En consecuencia los primeros cuatro bits, o sea el primer "nibble", cuando es 3, indica que lo que sigue en el segundo "nibble" o segundo grupo de cuatro bits, es una cifra comprendida entre 0 y 9.

1.B.2.8.1 - SUMA ENTRE DECIMALES CODIFICADOS:

En definitiva, para representar cada cifra se utiliza un grupo de cuatro binarios, por tanto los registros deberán ser capaces de soportar esta cantidad de información, suministrándola en paralelo al sumador.

Dado que el código admite más de diez alternativas, exactamente 16, las que van del 10 al 15 no nos servirán, pero durante las operaciones se presentarán errores, por ejemplo en la suma de ocho y cinco, cuyo resultado es 13, el cual es perfectamente representable para el código, pero no para nosotros, por lo que nos hará falta corregir estos resultados mayores a nueve y menores que 16. Esta corrección se consigue sumando un seis binario y tomando un arrastre decimal para la próxima suma. Cuando el resultado es mayor que 16, el proceso es igual, salvo que la detección se debe realizar por el arrastre de esa cantidad.

Ejemplos:

```
   3  0011     5  0101     9  1001
+2  0010   +6  0110   +8  1000
   5  0101    11  1011   17 10001
          +0110      +0110
               1 0001     1 0111
```

En el primer caso no hace falta corrección, en el segundo se suma seis y se obtiene el resultado correcto, primer nible vale el decimal 1 y el segundo también el decimal 1, por lo que queda el 11 buscado. En el tercer caso, el resultado intermedio sería el 11 decimal, al sumarle seis se obtiene el 17 buscado.

En la figura 1.B.24, se tiene el diagrama de flujo que cumple con las funciones necesarias para éste cálculo, y en la 1.B.25, el diagrama del circuito.

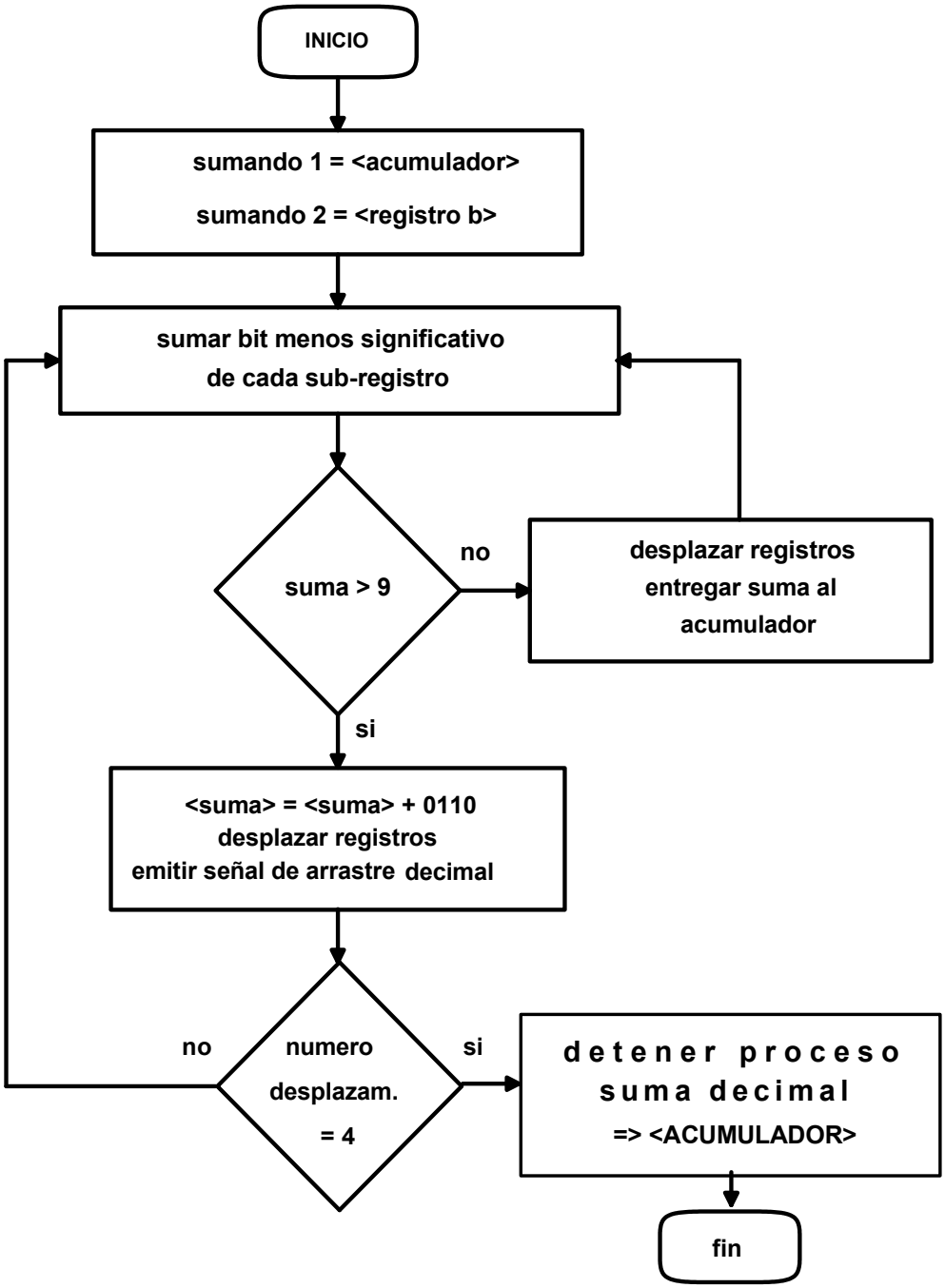


Figura 1.B.24 - Diagrama de flujo del sumador BCD.

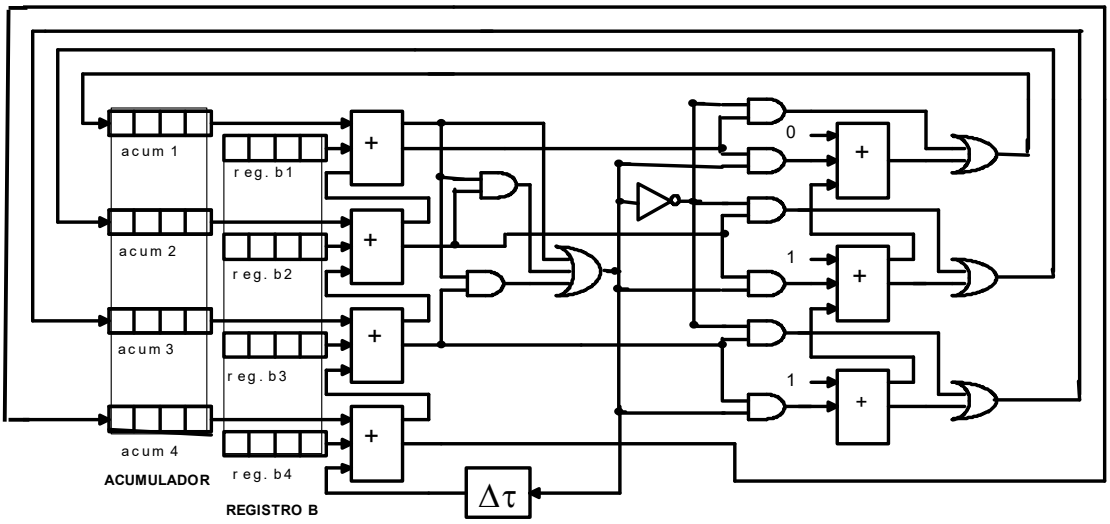


Figura 1.B.25 - Circuito sumador/corrector BCD.

Como vemos, cada uno de los registros en realidad está formado por cuatro sub-registros, en los cuales se almacena cada uno de los decimales codificados, que como sabemos es de cuatro bits cada uno. A los que se los denomina al igual que antes, acumulador y registro B, y en forma vertical estarán los cuatro bits de cada código.

Las compuertas ubicadas a continuación de cada sumador determinan el estado de error de la combinación de salidas, que ocurrirá al presentarse los binarios que van del 1010, que es el decimal 10, al 1111 que es el 15. Mientras que el arraste del sumador dispuesto en la parte superior, corresponde a cualquier resultado que pase de 15, o sea el que llamamos arrastre de 16.

Con estas condiciones, mediante el siguiente nivel de compuertas se derivan las señales para que pasen por los sumadores encargados de agregar 0110 a los resultados errados, al mismo tiempo que generan un arrastre para la suma de los próximos dígitos, lo que indica que la cantidad anterior era 10 o más.

En caso de no ocurrir lo anterior, y la suma es la correcta, las compuertas la derivan directamente al acumulador, que como siempre es el encargado de guardar los resultados.

Otra de las cosas que podemos observar, es que el dígito binario de menor peso del resultado será siempre correcto, pues no pasa por un sumador adicional, dado que se le suma cero.

1.B.2.8.2 - SUMADOR ALGEBRAICO:

En este caso se trata de números con signo, por tanto deberán mezclarse sumas y restas, sumando los de signo positivo y restando los de signo negativo.

El signo de un número codificado está representado por un dígito binario ubicado en una casilla determinada, en nuestro caso vamos a suponer que al final del

registro que lo contiene. La convención es tal que cuando allí hay un 1 el número es negativo, y cuando hay un 0, el número es positivo.

En la figura 1.B.27, se encuentra indicado el circuito para realizar la suma algebraica, mientras que en la figura 1.B.26 se muestra el diagrama de flujo que cumple.

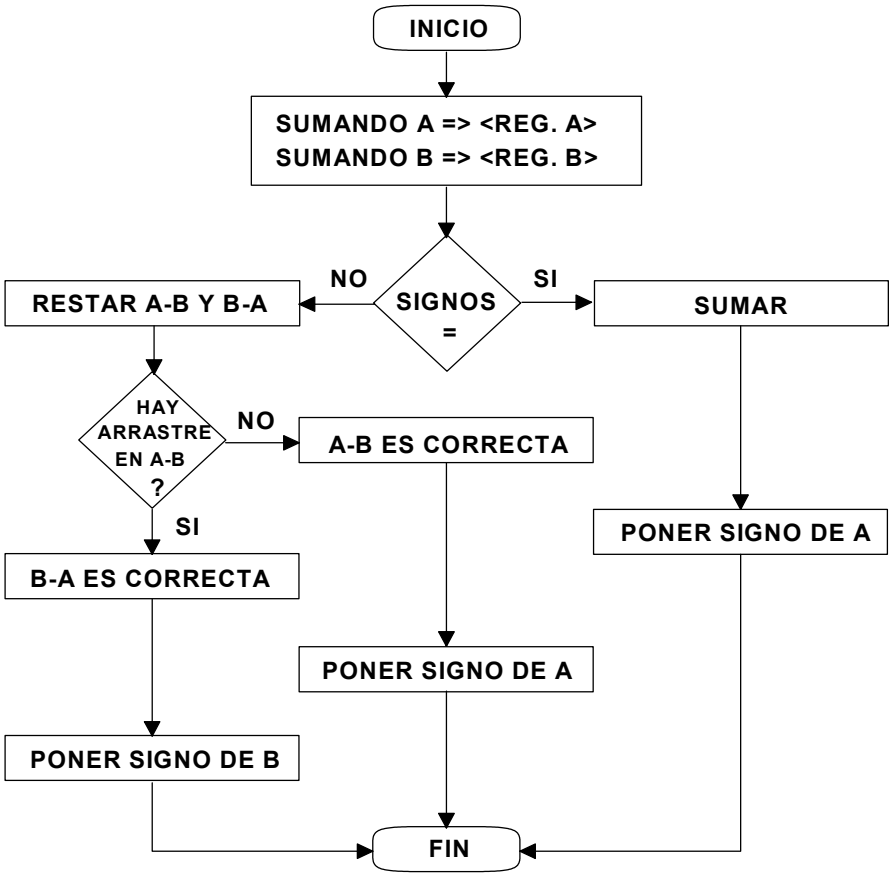


Figura 1.B.26 - Diagrama de flujo de la suma algebraica.

En primer lugar se prueba si los signos son iguales, en cuyo caso la operación a realizar es la suma, y el signo del resultado es igual a cualquiera de los dos signos, en nuestro caso supondremos el de A.

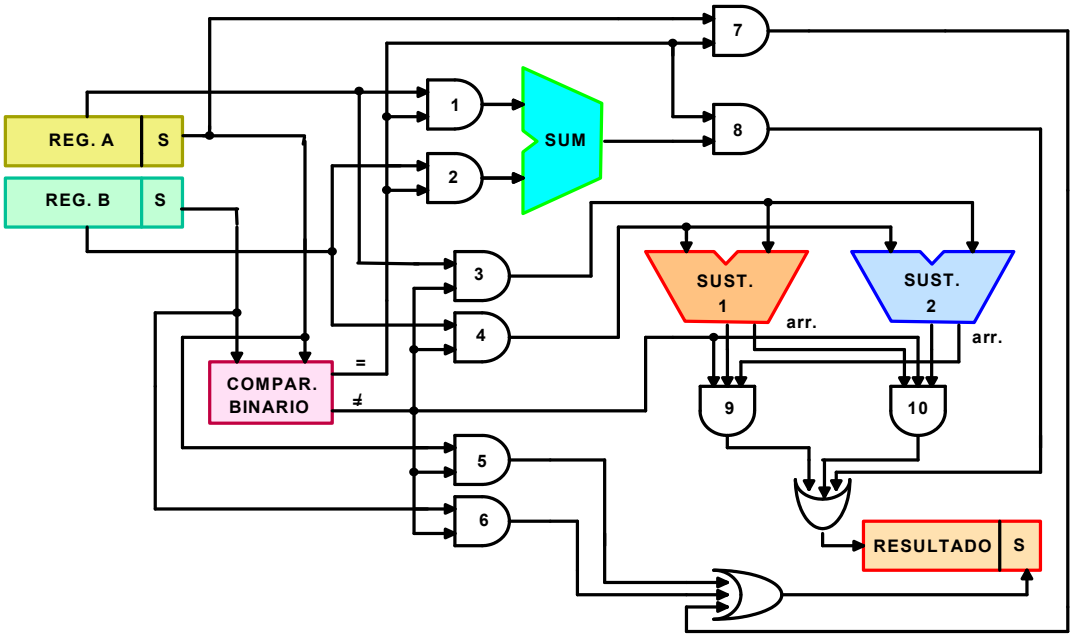


Fig. 1.B.27 - Sumador Algebraico.

Cuando los signos son distintos, se efectúan las dos restas posibles, la verdadera será la que no posee arrastre, en consecuencia se utiliza el arrastre de la incorrecta para habilitar la operación válida, y el signo será el signo del mayor en valor absoluto.

1.B.2.9 - MULTIPLICADOR DECIMAL:

La operación se realiza en la misma forma que en el caso de la multiplicación binaria acelerada, o sea por suma y desplazamiento.

En la figura 1.B.28, que contiene el diagrama de flujo previsto, indica que debe cargarse un contador en descuento con cada uno de los dígitos del multiplicador, luego de cada desplazamiento. Ahora se comienza a sumar el multiplicando con el registro A tantas veces como lo indica el contador, el cual al llegar a cero provoca un desplazamiento y el reinicio de las sumas.

Igual que en el caso del multiplicador binario, el registro para el resultado es de doble longitud, y el proceso se detendrá cuando hayan transcurrido tantos desplazamientos como sea la longitud de los registros.

En la figura 1.B.29, se tiene el esquema del circuito que cumple con lo dicho, siendo el tiempo palabra igual en pulsos a la longitud de los registros, en bytes.

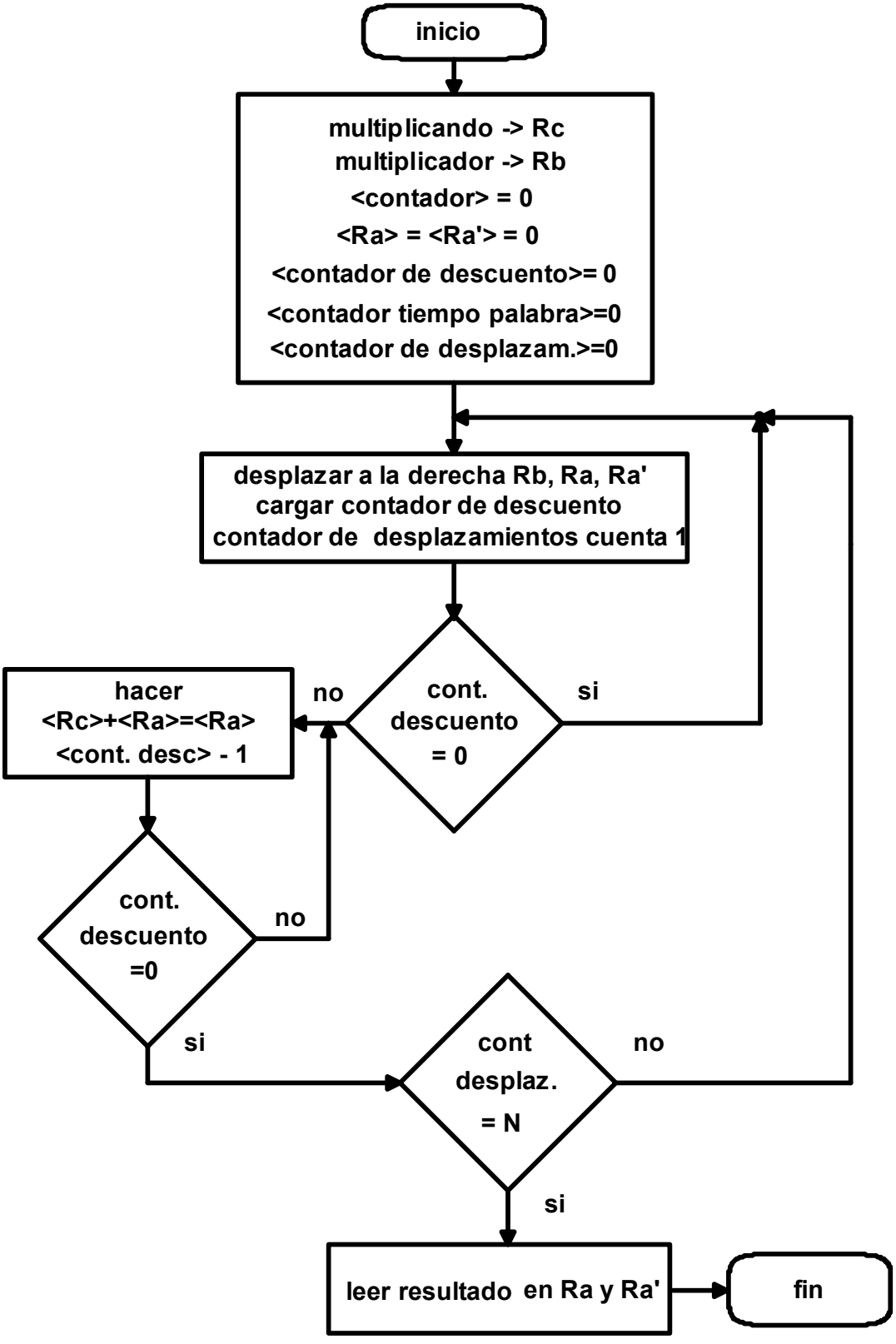


Figura 1.B.28 - Diagrama de flujo del multiplicador decimal.

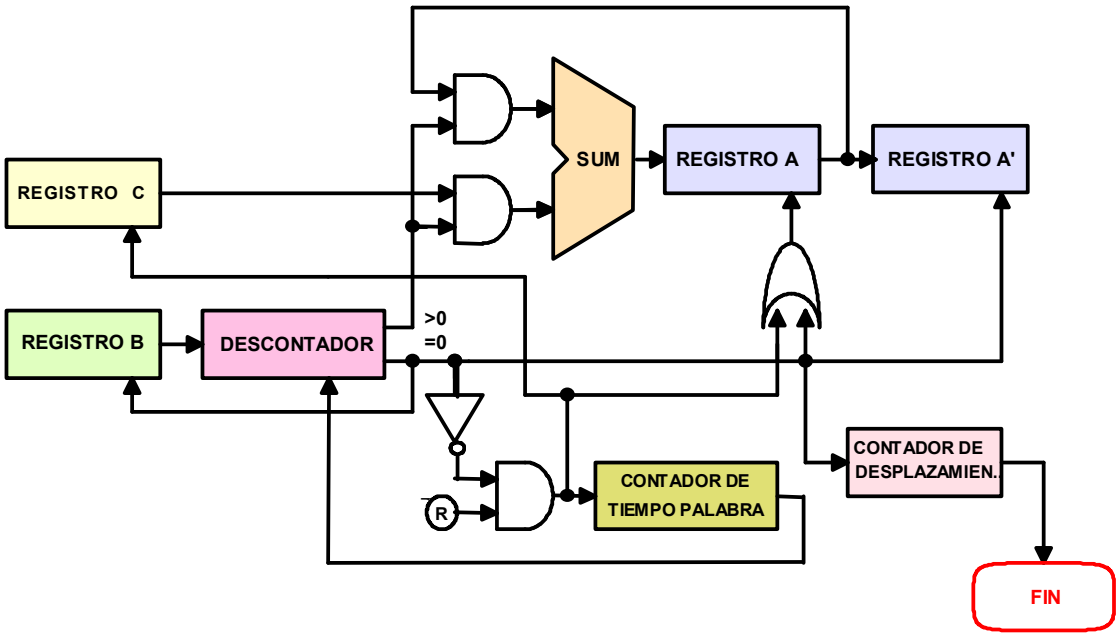


Figura 1.B.29 - Multiplicador decimal.

1.B.2.10 - MECANIZACIÓN DE LAS OPERACIONES CON NÚMEROS EXPRESADOS EN COMA FLOTANTE:

Recordemos que la expresión de cantidades en coma flotante, o en notación científica, según se quiera considerar, consiste en representar un número mediante una mantisa y una base elevada a un cierto exponente, por supuesto que uno o los dos con su signo.

En la figura 1.B.30, tenemos la disposición de los campos de un registro para números en coma flotante, según la norma IEEE 755. En primer lugar, a partir de la izquierda, tenemos el signo del número, luego el exponente polarizado, y finalmente la mantisa o fracción.

Figura 1.B.30 - Campos de un registro para escritura de cantidades en coma flotante.

Recordemos que si está en formato simple se un bit de signo, ocho para el exponente y veintitrés para la mantisa, en total 32 bits para el registro. En cambio, en formato doble, el registro debe tener 64 bits, divididos en uno para el signo, once para el exponente y 52 para la mantisa. Si hablamos en bytes, ello significa, en formato simple, un byte para el exponente, y tres bytes para el signo y la mantisa, y en formato doble, la palabra es de ocho bytes, donde hay un byte y medio, o sea 12 bits para el signo y el exponente, y seis bytes y medio para la mantisa.

Cuando se realizan operaciones de suma y resta, primero se deben igualar los exponentes y luego operar, mientras que en el caso de multiplicaciones y divisiones, los exponentes se suman o restan respectivamente.

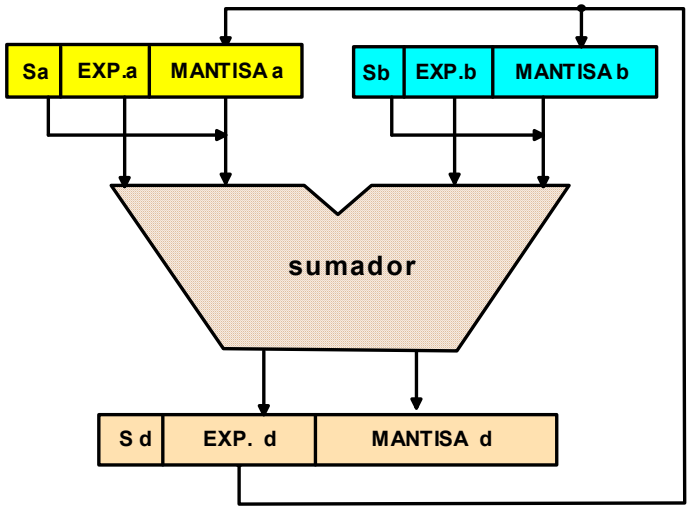


Figura 1.B.31 - Esquema lógico del circuito posible para operaciones en coma flotante.

Finalmente, una vez obtenido el resultado, el mismo debe ser normalizado, en forma tal que el primer dígito significativo ocupe el primer casillero del registro, lo que también significa que el primer dígito significativo se encuentra justamente detrás de la coma decimal.

En la figura 1.B.31 se tiene un esquema del circuito, simplificado, para llevar a cabo la operación de suma, que también puede ser resta, dado que la diferencia radica en la forma de calcular los arrastres.

La secuencia de operaciones, tal como se indica en el diagrama de flujo de la figura 1.B.32, consiste en:

1 - Comparación de exponentes, lo que se hace por diferencia. Cuando Cd = 0 los exponentes son iguales, cuando Cd > 0; Ea > Eb, y cuando Cd < 0 será: Ea < Eb. Se toma el menor de ellos, se lleva a un registro de desplazamiento, denominado Rd y se va desplazando la mantisa a la derecha, incrementando el valor de su exponente hasta que Cd=0, lo que indica su igualdad.

2 - Se opera sobre las mantisas, o sea se suman o se restan, quedando el resultado en Rd.

3 - Se normaliza el número, buscando la primera cifra significativa y se la ubica después de la coma decimal, recordemos que en esta notación, ello significa ubicarla en primer lugar o como dígito más significativo del registro. Cuando hay rebose, ello significa que hay parte entera del número, por ello es que se lo debe llevar a la derecha, aumentando el valor de su exponente. Se desplaza a la izquierda hasta que se tenga un dígito significativo, y se disminuye el exponente.

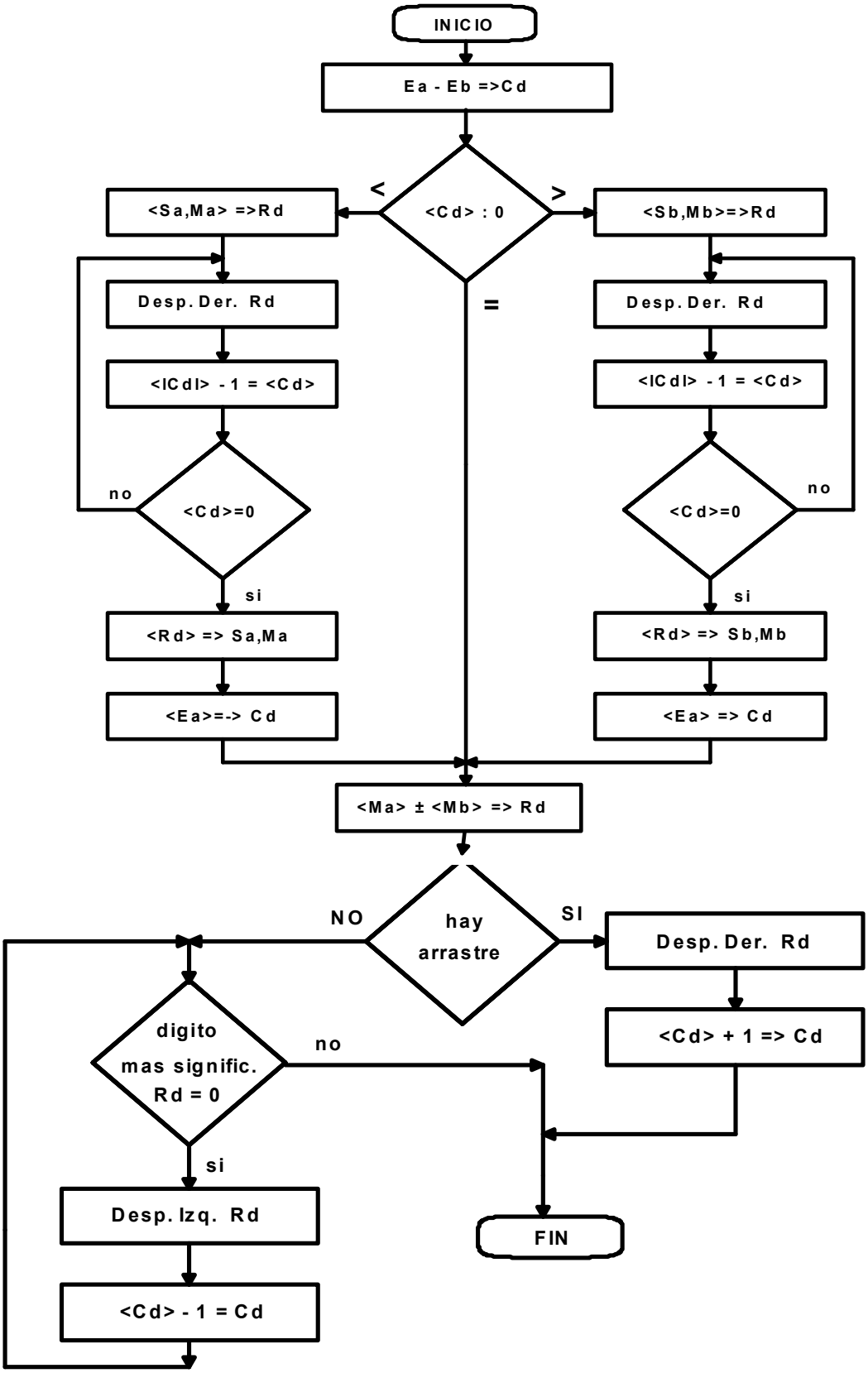


Figura 1.B.32 - Diagrama de flujo para la realización de la suma o resta entre cantidades expresadas en notación científica.

Dec	Hex	Code	Dec	Hex	Code	Dec	Hex	Code	Dec	Hex	Code
0	00	NUL	32	20		64	40	space	96	60	-
1	01	SOH	33	21		65	41		97	61	/
2	02	STX	34	22		66	42		98	62	
3	03	ETX	35	23		67	43		99	63	
4	04		36	24		68	44		100	64	
5	05	HT	37	25	LF	69	45		101	65	
6	06		38	26	ETB	70	46		102	66	
7	07	DEL	39	27	ESC	71	47		103	67	
8	08		40	28		72	48		104	68	
9	09		41	29		73	49		105	69	
10	0A		42	2A		74	4A	[	106	6A	|
11	0B	VT	43	2B		75	4B	.	107	6B	,
12	0C	FF	44	2C		76	4C	<	108	6C	%
13	0D	CR	45	2D	ENQ	77	4D	(	109	6D	_
14	0E	SO	46	2E	ACK	78	4E	+	110	6E	>
15	0F	SI	47	2F	BEL	79	4F	| !	111	6F	?
16	10	DLE	48	30		80	50	&	112	70	
17	11		49	31		81	51		113	71	
18	12		50	32	SYN	82	52		114	72	
19	13		51	33		83	53		115	73	
20	14		52	34		84	54		116	74	
21	15		53	35		85	55		117	75	
22	16	BS	54	36		86	56		118	76	
23	17		55	37	EOT	87	57		119	77	
24	18	CAN	56	38		88	58		120	78	
25	19	EM	57	39		89	59		121	79	‘
26	1A		58	3A		90	5A	!]	122	7A	:
27	1B		59	3B		91	5B	$	123	7B	#
28	1C	IFS	60	3C		92	5C	*	124	7C	@
29	1D	IGS	61	3D	NAK	93	5D	)	125	7D	‘
30	1E	IRS	62	3E		94	5E	;	126	7E	=
31	1F	IUS	63	3F	SUB	95	5F	^	127	7F	"

ANEXO 1 - Tabla del código EBCDIC – Parte 1.

Dec	Hex	Code	Dec	Hex	Code	Dec	Hex	Code	Dec	Hex	Code
128	80		160	A0		192	C0	{	224	E0	\
129	81	a	161	A1	~	193	C1	A	225	E1	
130	82	b	162	A2	s	194	C2	B	226	E2	S
131	83	c	163	A3	t	195	C3	C	227	E3	T
132	84	d	164	A4	u	196	C4	D	228	E4	U
133	85	e	165	A5	v	197	C5	E	229	E5	V
134	86	f	166	A6	w	198	C6	F	230	E6	W
135	87	g	167	A7	x	199	C7	G	231	E7	X
136	88	h	168	A8	y	200	C8	H	232	E8	Y
137	89	i	169	A9	z	201	C9	I	233	E9	Z
138	8A		170	AA		202	CA		234	EA	
139	8B		171	AB		203	CB		235	EB	
140	8C		172	AC		204	CC		236	EC	
141	8D		173	AD		205	CD		237	ED	
142	8E		174	AE		206	CE		238	EE	
143	8F		175	AF		207	CF		239	EF	
144	90		176	B0		208	D0	}	240	F0	0
145	91	j	177	B1		209	D1	J	241	F1	1
146	92	k	178	B2		210	D2	K	242	F2	2
147	93	l	179	B3		211	D3	L	243	F3	3
148	94	m	180	B4		212	D4	M	244	F4	4
149	95	n	181	B5		213	D5	N	245	F5	5
150	96	o	182	B6		214	D6	O	246	F6	6
151	97	p	183	B7		215	D7	P	247	F7	7
152	98	q	184	B8		216	D8	Q	248	F8	8
153	99	r	185	B9		217	D9	R	249	F9	9
154	9A		186	BA		218	DA		250	FA	
155	9B		187	BB		219	DB		251	FB	
156	9C		188	BC		220	DC		252	FC	
157	9D		189	BD		221	DD		253	FD	
158	9E		190	BE		222	DE		254	FE	
159	9F		191	BF		223	DF		255	FF	

ANEXO 1 - Tabla del código EBCDIC – Parte 2.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

ANEXO II – Código ASCII completo para uso en Internet.

APÉNDICE: SISTEMAS DE NUMERACIÓN

AB.1 - INTRODUCCIÓN:

Cuando hablamos de números, solo pensamos en el sistema decimal, que es el que nos enseñaron en la escuela, sin embargo, desde un punto de vista general, un número es expresable mediante la relación:

$$N = \sum_{i=0}^{n} a_i b^i \qquad \textbf{(2.1)}$$

dónde: N **= número considerado**
a_i **= cualquier cifra entre 0 y** b**-1**
b **= base del sistema de numeración**
i **= exponente**

De esta manera, para diferentes bases podemos especificar diferentes sistemas de numeración. Así tendríamos una infinita cantidad de sistemas, pero solo algunos son útiles, en especial el binario, el octal, el decimal y el hexadecimal, lo que significa tener base dos, ocho, diez y dieciséis, respectivamente.

Por otra parte, en las máquinas no se puede operar con circuitos que tengan más de dos estados, dado que solo así se comportan adecuadamente los componentes electrónicos que los componen. Por tanto en el interior de la computadora solo es posible tener una representación binaria, pero ello haría que las traducciones fuesen dificultosas, por tanto se hace uso de códigos, que mediante una numeración binaria permite la representación de cantidades en decimal, o en cualquier otra base, así como que mediante combinaciones de números decimales es posible representar no solo el alfabeto, sino también otros signos.

Finalmente debemos observar como automatizamos las operaciones aritméticas, pues no es lo mismo hacerlas a mano que mediante una máquina automática, que además trabaja con códigos y en una base de numeración diferente a la aplicada por nosotros.

AB.2 - EL SISTEMA DECIMAL:

Es evidente que este sistema deriva de la cantidad de dedos que poseen nuestras manos, si hubiésemos tenido seis en cada una, el nuestra base sería duodecimal.

Fundamentalmente consiste en expresar una lista de cifras, que por su ubicación significarán unidades, decenas, centenas, etc., o si están a la derecha de la coma decimal, décimos, centésimos, milésimos etc.

Ello quedaría indicado así:

$$\ldots a_3\ a_2\ a_1\ a_0,\ a_{-1}\ a_{-2}\ a_{-3}\ \ldots.$$

donde podemos escribir:

$$\ldots a_3 \times 10^3 + a_2 \times 10^2 + a_1 \times 10^1 + a_0 \times 10^0 + a_{-1} \times 10^{-1} + a_{-2} \times 10^{-2} + a_{-3} \times 10^{-3} + \ldots$$

que es lo indicado por la expresión 2.1.

En resumen, un número N en base diez, se expresa como:

$$N_{(10)} = \sum_{i=-\infty}^{+\infty} a_i 10^i$$

Donde: *i* puede ser un entero comprendido entre - y + infinito
***a* cualquier valor entre 0 y 10-1=9.**

AB.3 - EL SISTEMA BINARIO:

Como ya dijimos, el sistema binario consiste en utilizar una base dos, y por tanto las cifras serán solo 0 y 1, por lo que una expresión tal como:

110011,01101

deberá leerse:

$$1x2^5 + 1x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} + 1x2^{-3} + 0x2^{-4} + 1x2^{-5}$$

Que expresado en decimal, será:

```
1x32+1x16+0x8+0x4+1x2+1x1+0x(1/2)+1x(1/4)+1x(1/8)+0x(1/16)+1x(1/32)=

=32+16+2+1+0,25+0,125+0,03125 = 51,40625
```

La gran ventaja de éste sistema de numeración, es que solo se necesitan dos estados para cada cifra, además las tablas de sumar y multiplicar se vuelven muy simples, lo cual reduce las habilidades que debe poseer un mecanismo para realizar el cálculo.

AB.3 - EL SISTEMA OCTAL:

Ahora, la base es 8, con lo cual solo tenemos cifras del 0 al 7, y cualquier número, tal como:

1564,23

quedará:

$$1x8^3 + 5x8^2 + 6x8^1 + 4x8^0 + 2x8^{-1} + 3x8^{-2}$$

que en decimal será:

512+320+48+16+3+1/4+3/64 = 899,29687

Este sistema se utilizó en máquinas con palabra muy corta, de tan solo tres bits, dado que con ellos se alcanza a representar ocho estados diferentes.

AB.4 - EL SISTEMA HEXADECIMAL:

Justamente para aprovechar al máximo la capacidad binaria de una máquina, sería necesario contar siempre en hexadecimal, pues hacen falta cuatro bits para representar las diez cifras distintas de un sistema decimal, por tanto si nos limitáramos solo a ello, nos sobrarían seis estados. En este caso además de las diez cifras decimales, del cero al nueve, se utilizan las letras A=10; B=11; C=12; D=13; E=14; y F=15.

Así una cantidad expresada como:

5F7A,D8E

será: $5x16^3+15x16^2+7x16^1+10x16^0+13x16^{-1}+8x16^{-2}+14x16^{-3}$

que en decimal vale:

20480+3480+896+10+0,8125+0,03125+0,0034179=24839,847167

AB.5 - CONVERSIÓN ENTRE NÚMEROS DADOS EN DIFERENTES BASES.

Una parte del problema ya ha sido indicada en el inciso anterior, como pasaje del número en cualquier base a decimal. Ahora nos queda la variante opuesta, o sea determinar el número en base 2, 8 o 16, a partir de la expresión en decimal.

AB.5.1 - DECIMAL A BINARIO:

El método consiste en dividir sucesivamente al numero por 2, y se van anotando los residuos, la sucesión de los mismos es en orden inverso al de la división.

Por ejemplo:

$N_{(10)} = 2142$

Dónde $N_{(10)}$ significa N en base 10.

Para convertir, la operación que se realiza es una división, y se va anotando el resto:

2142/2 = 1071 resto 0
1071/2 = 535 resto 1
535/2 = 267 resto 1
267/2 = 133 resto 1
133/2 = 66 resto 1
66/2 = 33 resto 0
33/2 = 16 resto 1
16/2 = 8 resto 0
8/2 = 4 resto 0
4/2 = 2 resto 0
2/2 = 1 resto 0

1/2 = 0 resto 1

Por lo que el binario equivalente se tendrá leyendo los restos en orden inverso:

$$N_{(2)} = 100001011110$$

En caso que el número sea fraccionario, en vez de dividir se debe multiplicar, y en vez de tomar el resto, se toma el rebose, o la parte entera como dígito binario, por ejemplo:

$$X_{(10)} = 0,567$$

0,567 x 2 = 1,134 rebose 1
0,134 x 2 = 0,268 rebose 0
0,268 x 2 = 0,536 rebose 0
0,536 x 2 = 1,072 rebose 1
0,072 x 2 = 0,144 rebose 0
0,144 x 2 = 0,288 rebose 0
0,288 x 2 = 0,576 rebose 0
0,576 x 2 = 1,152 rebose 1

y así sucesivamente, recordando que a mayor cantidad de cifras, o sea de productos, mayor precisión tendrá el resultado.

La otra diferencia con relación a la conversión de enteros, es que ahora la lectura se hace en el mismo sentido en que se realizan los productos.

$$X_{(2)} = 0,10010001.....$$

En caso de números reales, o sea con parte entera y fraccionaria, la primera se divide por dos y la segunda se multiplica por la misma cantidad.

AB.5.2 - DECIMAL A OCTAL:

El procedimiento es siempre el mismo, la parte entera se divide por la base, y la parte fraccionaria se multiplica por la misma, anotando en un caso los residuos o restos y en el otro los reboses o desbordes.

Ejemplo: $N_{(10)} = 987,34$

Procedimiento:

987/8 = 123 resto 3
123/8 = 15 resto 3
15/8 = 1 resto 7
1/8 = 0 resto 1

y además:
0,34 x 8 = 2,72 rebose 2
0,72 x 8 = 5,76 rebose 5
0,76 x 8 = 6,08 rebose 6
0,08 x 8 = 0,64 rebose 0

y así sucesivamente.

Por tanto el resultado es: $N_{(8)}$ = 1733,256

AB.5.3 - DECIMAL A HEXADECIMAL:

Siguiendo el mismo procedimiento, tomamos el número:

$N_{(10)}$ = 3879,654

Entonces:

3879/16 = 242 resto 7
242/16 = 15 resto 2
15/16 = 0 resto F

y además:

0,654 x 16 = 10,464 rebose A
0,464 x 16 = 7,424 rebose 7
0,424 x 16 = 6,784 rebose 6
0,784 x 16 = 12,544 rebose C

y así sucesivamente, con lo que el resultado será:

$N_{(16)}$ = F27,A76C