

# **ORGANIZACIÓN DE LA COMPUTADORA DIGITAL**

## **LA COMPUTADORA DIGITAL (A)**

Arquitectura de las computadoras: Es el comportamiento funcional de un sistema computacional desde el punto de vista de un programador.

Aspectos considerados(ejemplos):

- Tamaño y Tipo de datos.
- Tipos de operaciones que son soportadas.

Organización de las computadoras: Se refiere a las relaciones estructurales que no son visibles al programador.

Ejemplos:

- Interfases para los dispositivos periféricos.
- Frecuencia del reloj(clock).
- Tecnología usada para las memorias.

### **Niveles En Arquitectura De Computadoras**

Existen varios niveles o perspectivas en los cuales se puede considerar a la computadora. Cada uno de los niveles representa una abstracción de la computadora.

Una de las razones del suceso de las computadoras digitales es la independencia entre cada nivel:

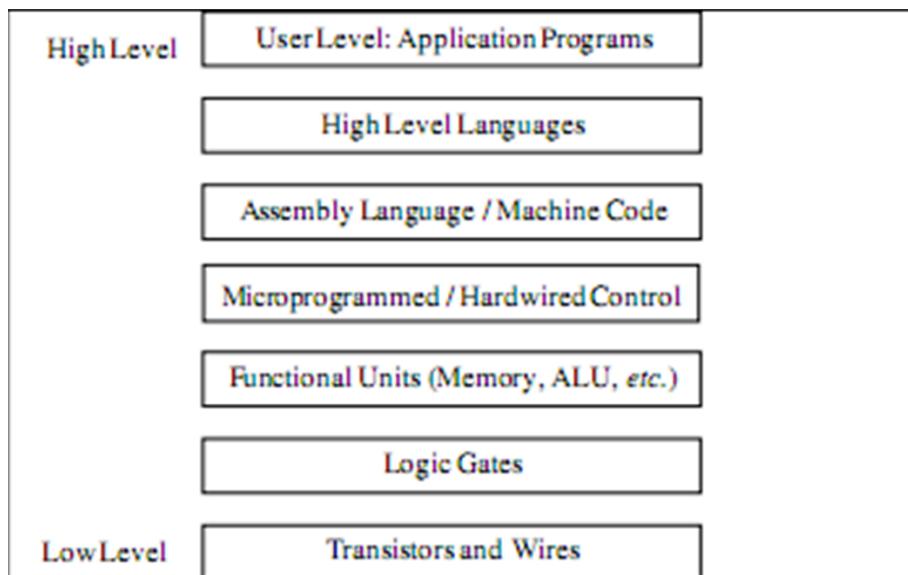
- Un usuario que corre un procesador de texto en la computadora no necesita conocer como está programado este.
- Un programador no necesita ser consciente de la estructura de compuertas lógicas dentro del computador.

Esta separación de niveles ha sido explotada en el desarrollo de las máquinas de compatibilidad ascendente:

IBM crea el concepto de “familia de máquinas” en la serie 360: máquinas con una mayor capacidad en la misma familia podían correr programas escritos para las

máquinas con menos capacidades de la misma familia sin modificar a estos programas.

## Máquina Multinivel



### Usuario O Nivel De Programas De Aplicación

El usuario ve a la computadora a través de los programas (procesador de texto, planilla de cálculo, etc) que corre en ella y muy poco o nada de su estructura de bajo nivel es visible.

### Nivel De Lenguajes De Alto Nivel

El programador solo vé al lenguaje (c, c++, java) y ningún detalle de bajo nivel de la máquina. Ve tipos de datos e instrucciones del lenguaje de alto nivel.

El rol del compilador es mapear los tipos de datos e instrucciones desde el lenguaje en alto nivel al verdadero hardware del computador

“Compatibilidad De Código Fuente”: Los programas escritos en lenguajes de alto nivel pueden ser recopilados para varias máquinas, correrán igual y darán los mismos resultados.

### Lenguaje Ensamblador/Código De Máquina

El compilador traslada desde el código fuente al “código de máquina”.

El código de máquina es un conjunto de ceros y unos conocido como “código binario de máquina” o “código binario”. Trata asuntos del hardware como registros y la transferencia de datos entre ellos.

Al conjunto de instrucciones de máquina para una determinada máquina se denomina “set de instrucciones”

El lenguaje ensamblador traslada mnemotécnicos como “move data, acc” en su correspondiente código de máquina (conjunto de ceros y unos).

“Compatibilidad Binaria”: máquinas que comparten el mismo set de instrucciones. por ejemplo:

El código de máquina que corre en una ibm 360 modelo 35 correrá sin cambios en una ibm 360 modelo 50

## Nivel Unidad De Control

Unidad De Control: Genera señales de control que permiten transferir los datos desde un registro hacia otro registro, a través de circuitos lógicos que crean una vía para su interconexión, interpretando a cada una de las instrucciones de máquina.

Puede ser:

- Unidad de control cableada:
  - ❖ Bloque de componentes de lógica digital.
  - ❖ Ventaja: Su velocidad de funcionamiento.
  - ❖ Desventaja: Complejo diseño y difícil de modificar.
- Unidad de control microprogramada:
  - ❖ Tiene un muy pequeño programa (microprograma de control) escrito en lenguaje de bajo nivel que está implementado en hardware, esto se conoce como firmware.
  - ❖ Un microcontrolador ejecuta cada una de las microinstrucciones.
  - ❖ A la instrucción de máquina se la denomina macroinstrucción y cada una se ejecuta por medio de la ejecución de un conjunto de microinstrucciones (microprograma específico).
  - ❖ Ventaja: Es más fácil modificar el set de instrucciones de la máquina (cambiando el microprograma de control).
  - ❖ Desventaja: La ejecución de una macroinstrucción es más lenta que en una unidad cableada.

## Nivel De Unidades Funcionales

**Unidades Funcionales:** Llamadas así porque realizan alguna función que es importante para la operación de la computadora. Incluyen:

- Registros internos de la cpu
- Unidad lógica aritmética(ALU)
- Memoria principal

### **Nivel De Compuertas Lógicas**

Implementan el más bajo nivel de operaciones lógicas sobre las cuales depende el funcionamiento del computador.

Las unidades funcionales se construyen con compuertas lógicas.

### **Nivel De Transistores Y Cables**

El más bajo nivel es el de los componentes como los transistores y los cables que conforman las compuertas lógicas.

En este nivel el funcionamiento de la computadora se pierde en detalles como:

- Tensiones
- Corrientes
- Retardo en la propagación de señales
- Efectos cuánticos
- Otros asuntos de bajo nivel

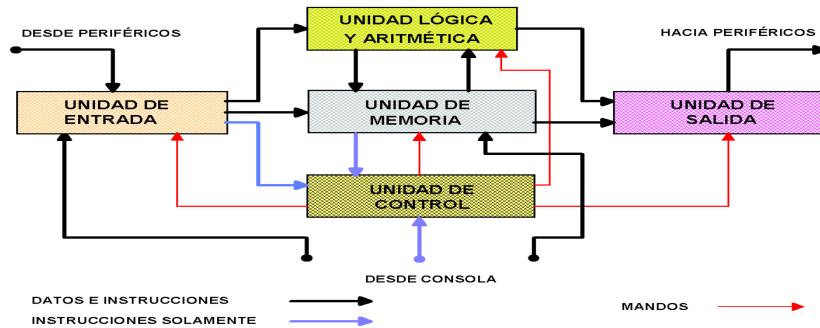
### **Definición De Computadora**

Máquina calculadora electrónica, constituida por un conjunto de dispositivos especializados, que operan en forma sincronizada mediante un programa común, que permiten sin intervención del hombre, efectuar complejas operaciones aritméticas y lógicas.

### **Principios De Von Neumann**

1. Unidad de memoria única capaz de almacenar datos e instrucciones, sin diferenciarlos
2. Unidad lógica-aritmética capaz de operar en binario
3. Unidad de control única, capaz de interpretar las instrucciones
4. Equipamiento de entrada/salida operado por la unidad de control

## Modelo De Von Neumann



Datos: Cantidades sobre las cuales debe operarse.

Instrucciones: Son las operaciones que deben realizarse sobre los datos.

### Misión De Cada Unidad De Una Computadora Digital

- La unidad de control interpreta las instrucciones y comanda al resto de la máquina.
- La unidad aritmética es la encargada de realizar las operaciones indicadas por las instrucciones.
- La unidad de memoria es la encargada de almacenar todos los datos e instrucciones.
- Las unidades de entrada y de salida son las encargadas de conectar la máquina al mundo exterior.

### Conformación De Cada Unidad

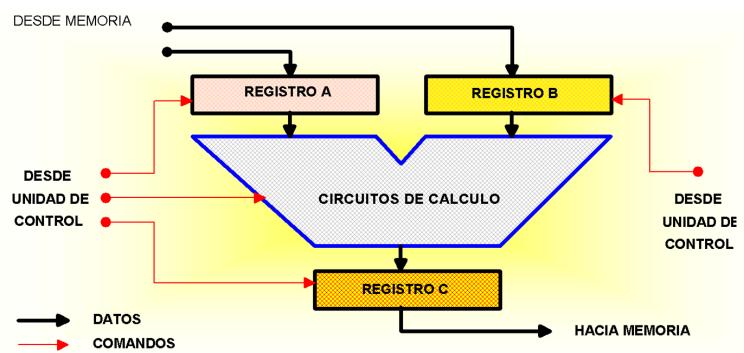
Son todas diferentes entre sí por su especialización.

Cada una está conectada a las otras de diversas maneras, según su función.

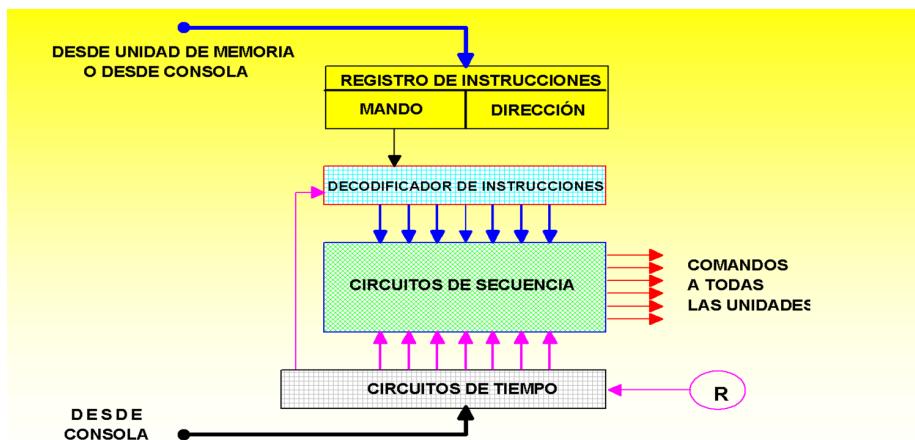
Solo se conectan al exterior las unidades de entrada y de salida.

La unidad de control puede tener accesos directos desde una consola.

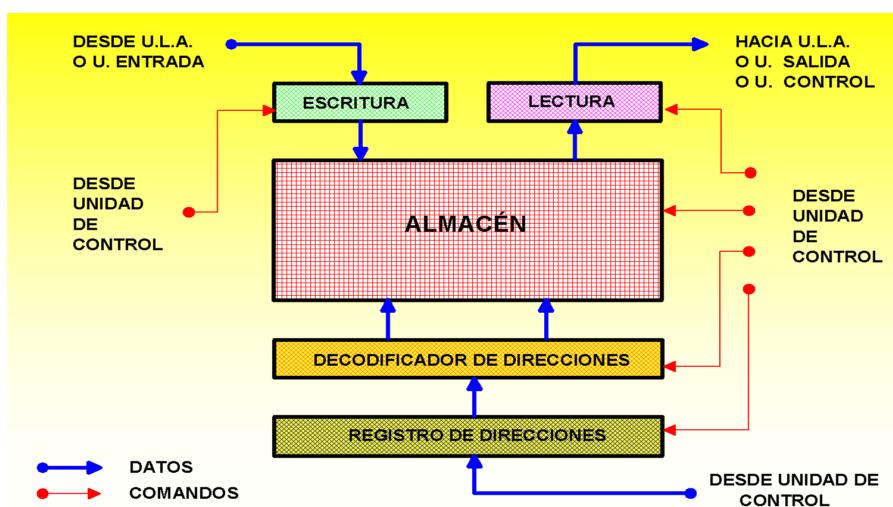
## Unidad Aritmética Y Lógica



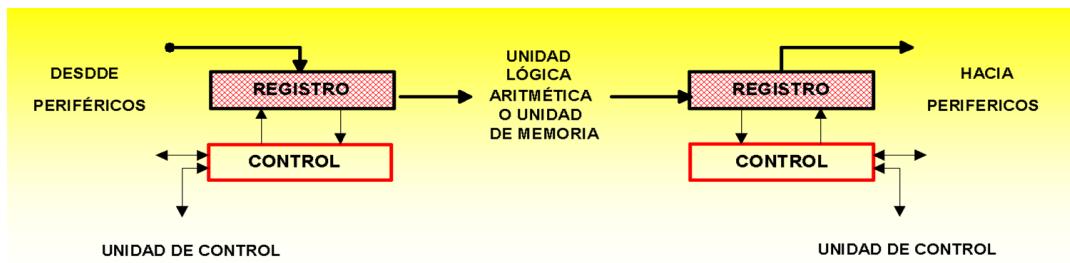
## Unidad De Control



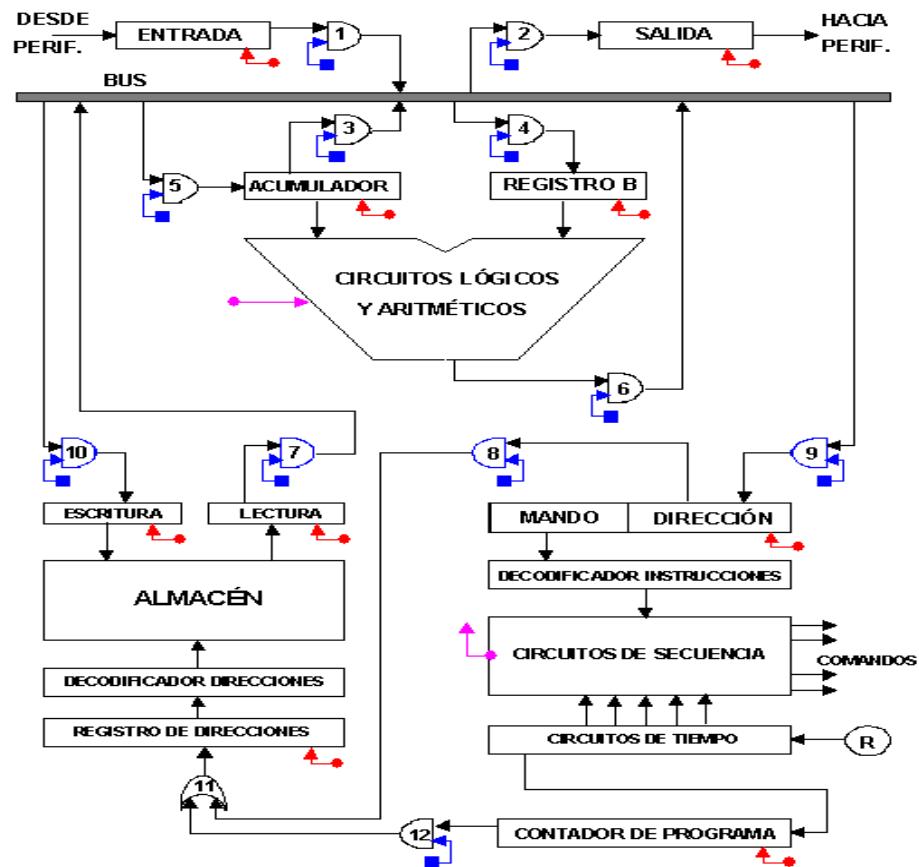
## Unidad De Memoria



## Unidades De Entrada Y Salida



## Interconexión De Las Unidades



## Funcionamiento De La Computadora

La unidad de control habilita los circuitos para buscar en memoria una instrucción y la transfiere al registro de instrucciones.

La unidad de control habilita los circuitos necesarios en la secuencia apropiada para ejecutar la instrucción.

### Registros De Computadoras

La función digital utilizada para retener información binaria en una computadora se denomina registro.

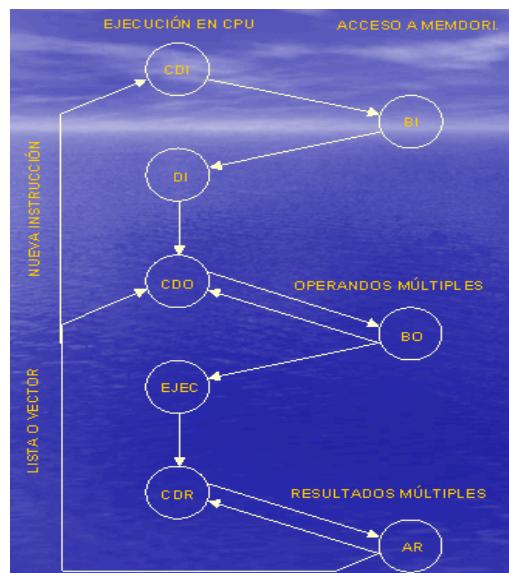
Un conjunto de flip-flops constituye un registro, ya que un flip-flop (biestable) es una celda binaria capaz de almacenar un bit de información.

Necesitan contener compuertas lógicas que realizan ciertas tareas de procesamiento de datos, generalmente controlan cuándo y cómo se debe transferir la información a los registros.

### Ciclo De Máquina Básico



### Ciclo De Máquina Como Diagrama De Estados



- <CDI> Cálculo De La Dirección De La Instrucción
  - <BI> Búsqueda De La Instrucción
  - <DI> Decodificación De La Instrucción
- <CDO> Cálculo De La Dirección Del Operando
  - <BO> Búsqueda Del Operando
  - <EJEC> Ejecución
- <CDR> Cálculo De La Dirección Del Resultado
  - <AR> Almacenamiento Del Resultado

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

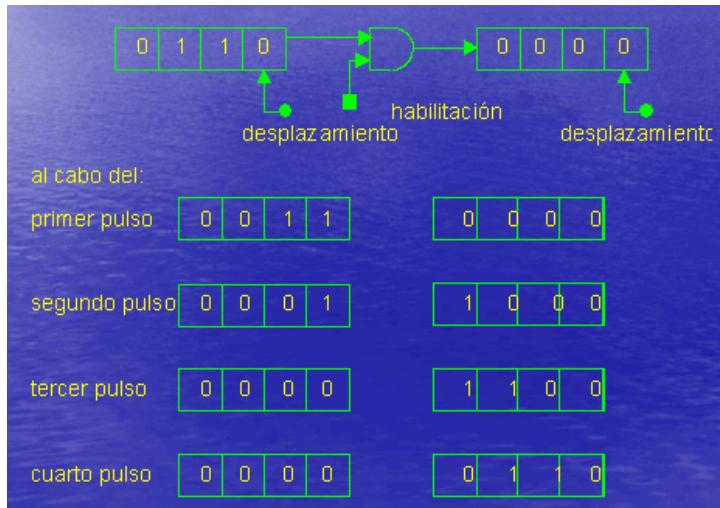
Lista O Vector: Cuando un dato no es un solo número, sino un conjunto ordenado de los mismos, tal como una lista o una matriz.

Operandos Múltiples: Es la lista o vector antes citado.

Resultados Múltiples: Se basa en la misma presunción anterior.

### Funcionamiento Simplificado De Una Computadora

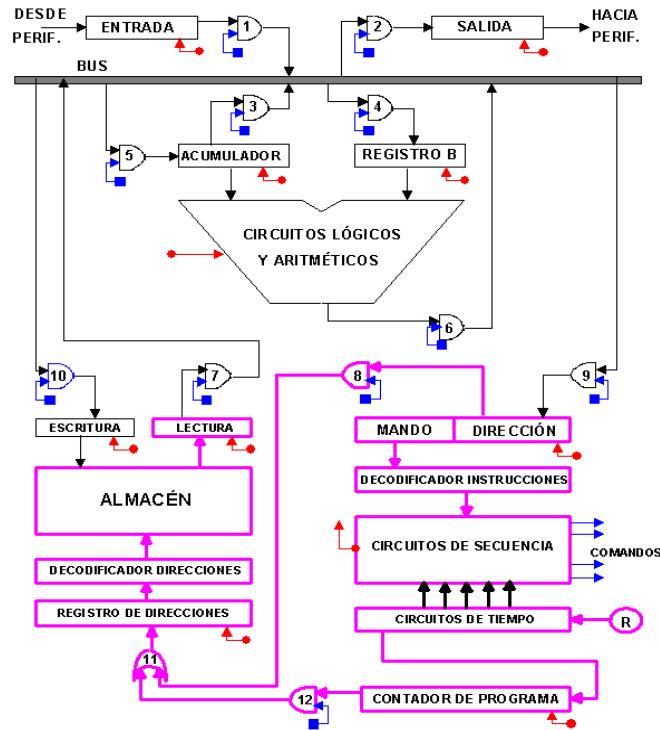
Máquina en Serie:



### CUANDO SE QUIERE EJECUTAR UNA NUEVA INSTRUCCIÓN

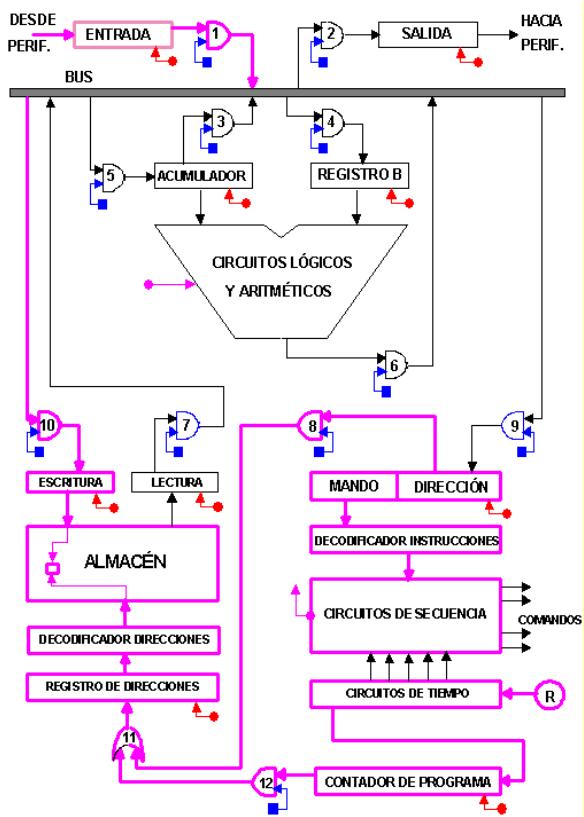
#### Fase De Búsqueda

- La UC envía la parte dirección del registro de instrucciones al Registro de direcciones de la UM habilitando la compuerta 8.
- 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.



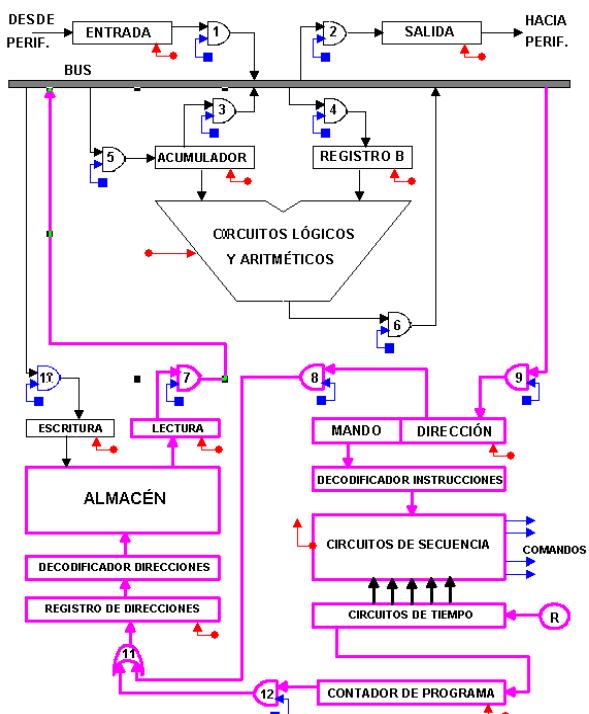
## Entrada De Datos

- La primera por la dirección de la instrucción, las restantes por el contador de programa.
- La unidad de control debe activar la compuerta 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 almácen.
- 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.



## Ejecución Del Programa

- Se carga la primera instrucción en el Registro de Instrucciones de la UC, y luego se ejecuta.
- 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 mediante la compuerta 9 es llevada al registro de instrucciones.



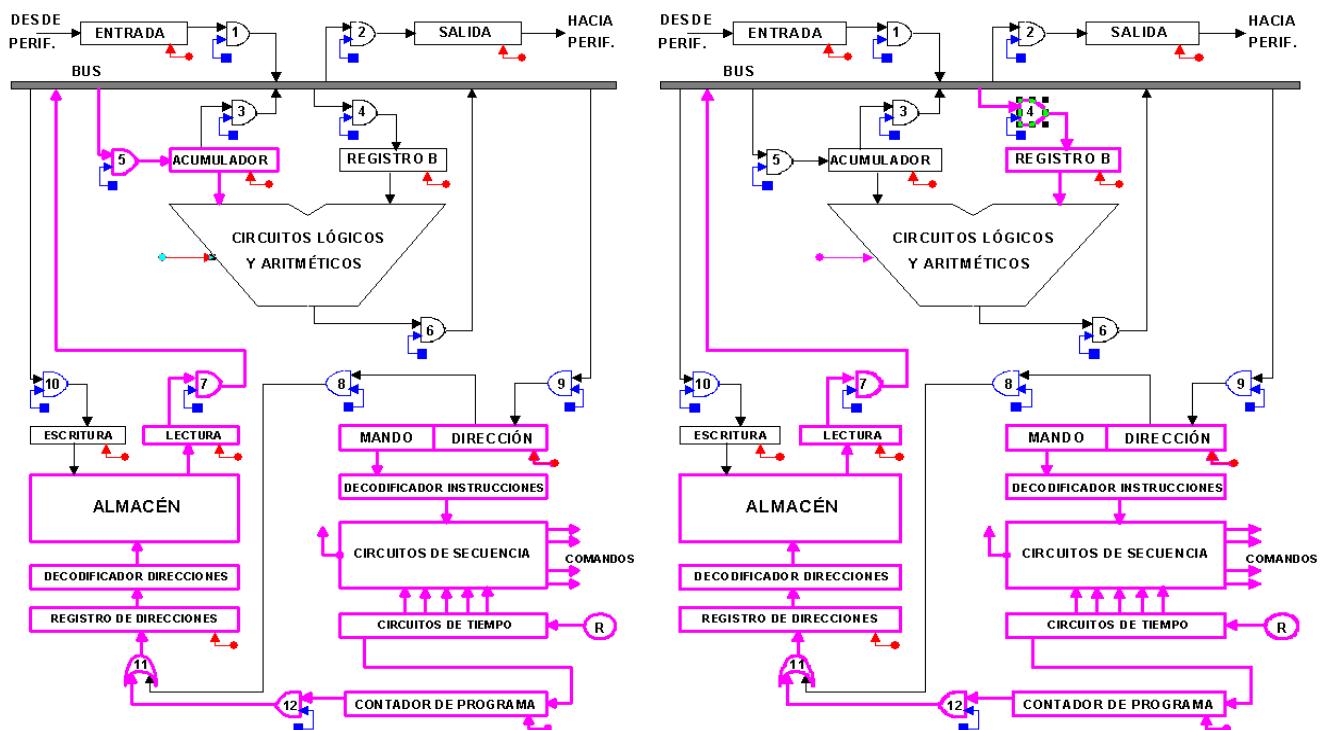
## Fase De Ejecución

La unidad de control emite señales para habilitar las compuertas necesarias, indica a la unidad aritmética la operación a realizar y remite las señales de reloj para realizar los desplazamientos.

### Alimentación Del Acumulador

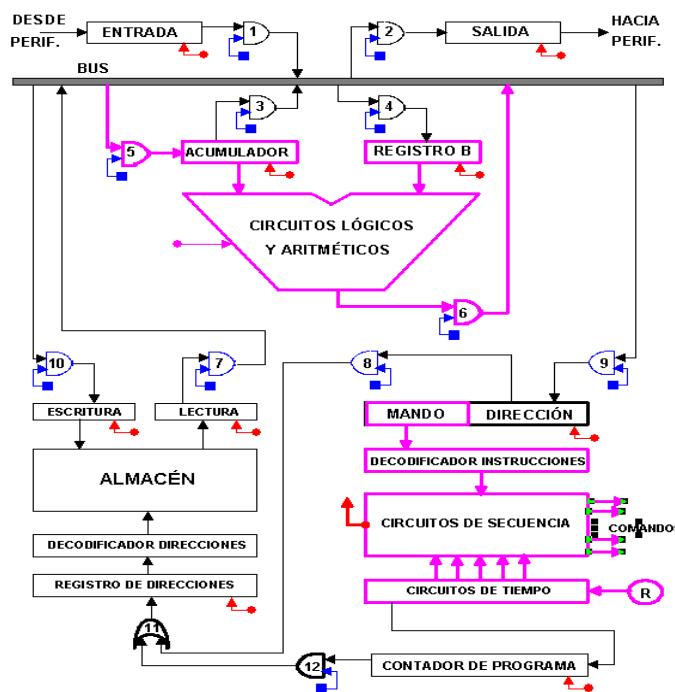
Para realizar operaciones en la ULA, se deben cargar los registros. Se habilita la compuerta 12, que llevará la dirección a la UM, y de la posición de memoria apuntada, se lleva al driver de lectura. Mediante la compuerta 7 el dato pasa al BUS y mediante la 5 puede ser pasado al Acumulador.

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



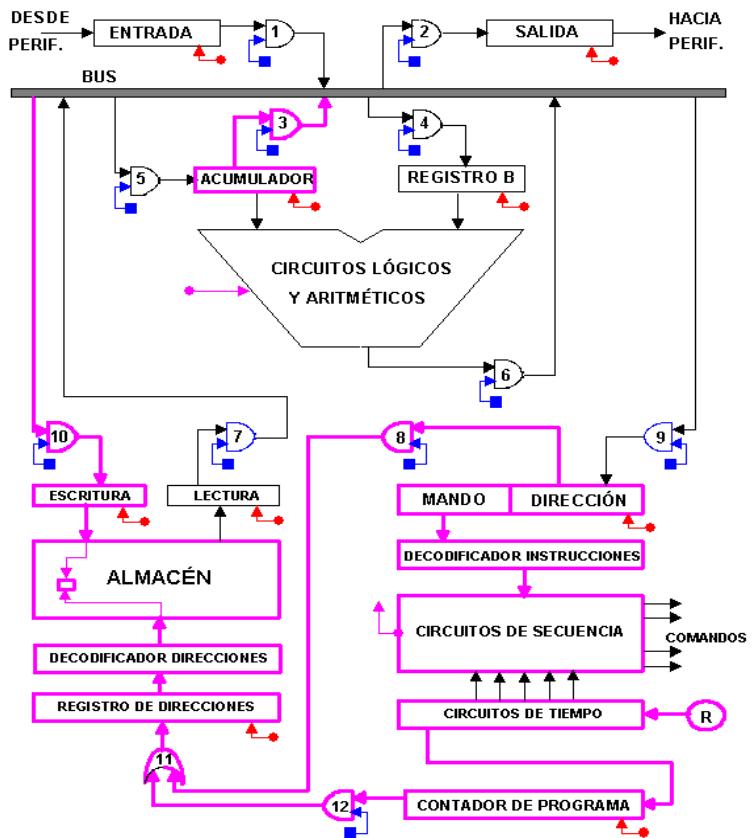
## Realización De La Operación

El mando es decodificado o interpretado, alimentando así a los circuitos de secuencia, de dónde 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.



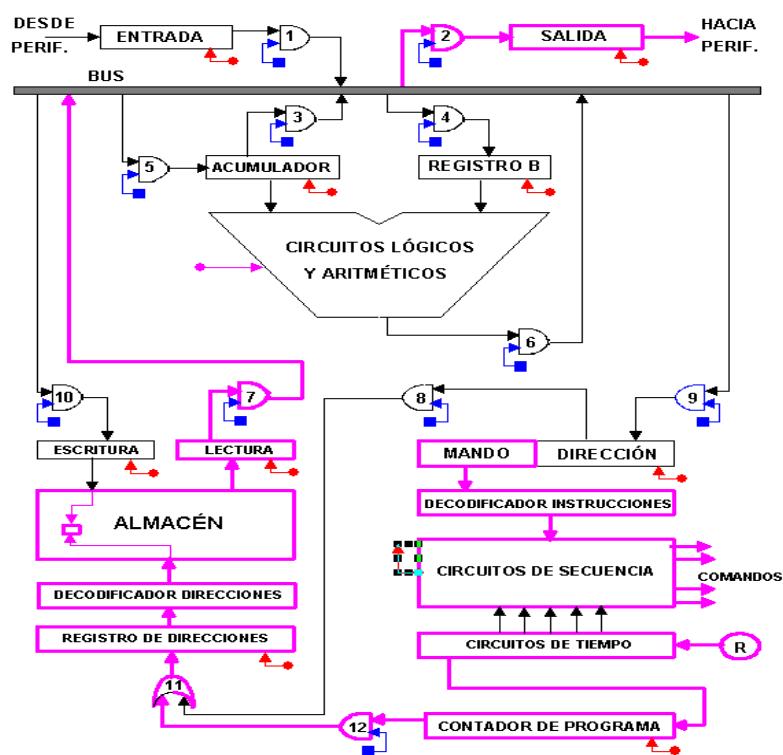
## Almacenar El Resultado

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.



## Salida De Los Resultados

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 12 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 salida.



## Programación

Toda computadora se programa en lenguaje de máquina, mediante el empleo de instrucciones de máquina.

Una instrucción de máquina es una lista de números binarios que está formada por dos partes, mando y dirección.

## Lenguaje De Máquina

Una palabra es una lista de números binarios, en la cual el registro de instrucciones separa dos partes:

El mando (indica lo que se debe hacer)

La dirección (indica sobre que se lo debe hacer)



## Modelo De Sistema De Buses

Si bien el modelo de Von Neumann permanece en las computadoras ha sido redibujado en el modelo de sistema de buses.

Se partitiona a la computadora en tres sub-unidades:

- CPU (ALU, REGISTROS Y CONTROL)
- MEMORIA PRINCIPAL
- UNIDAD DE ENTRADA/SALIDA

La comunicación entre las unidades se realiza a través de una vía de paso compartida llamada sistema de buses.

Físicamente los buses están constituidos por un conjunto de alambres que están agrupados por funciones

Sistema de buses:

- Bus De Datos: Contiene la información que se está transmitiendo.
- Bus De Direcciones: Identifica a dónde la información se está enviando
- Bus De Control: Describe aspectos de cómo y de qué manera se está enviando la información.
- Bus De Potencia O De Alimentación: Para entregar potencia eléctrica a los componentes, se da por entendida su presencia y no se dibuja.

## NIVEL LÓGICO-DIGITAL (B1) (REPRESENTACIÓN DE DATOS Y OPERACIONES CON ELLOS)

### Números

$$N = \sum_{i=0}^n a_i b^i$$

### Números En Base 10 O Decimales

En general escribimos un número de la siguiente forma:

$$a_4 a_3 a_2 a_1 a_0 , a_{-1} a_{-2} a_{-3} a_{-4}$$

LO QUE SIGNIFICA:  
 $a_4 10^4 + a_3 10^3 + a_2 10^2 + a_1 10^1 + a_0 10^0 +$   
 $a_{-1} 10^{-1} + a_{-2} 10^{-2} + a_{-3} 10^{-3} + a_{-4} 10^{-4}$

Podemos entonces resumir así:

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

### Numeración Binaria

Un número escrito como sucesión de unos y ceros tal como: 100110,101

Que será equivalente a:

$$\begin{aligned} & 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + \\ & + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ & = 32 + 4 + 2 + 0,5 + 0,125 = 38,625 \end{aligned}$$

Podemos entonces resumir así:

$$N_{(2)} = \sum_{i=-\infty}^{+\infty} a_i 2^i$$

## El Sistema Hexadecimal

Partiendo de: 5F7A,D8E

Tenemos:

$$5 \times 16^3 + 15 \times 16^2 + 7 \times 16^1 + 10 \times 16^0 + \\ + 13 \times 16^{-1} + 8 \times 16^{-2} + 14 \times 16^{-3} =$$

Es lo mismo que:

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

## Conversión De Números En Distintas Bases

### Decimal A Binario

$$N(10) = 2142$$

$$2142/2 = 1071 \text{ resto } 0$$

$$1071/2 = 535 \text{ resto } 1$$

$$535/2 = 267 \text{ resto } 1$$

$$267/2 = 133 \text{ resto } 1$$

$$133/2 = 66 \text{ resto } 1$$

$$66/2 = 33 \text{ resto } 0$$

$$33/2 = 16 \text{ resto } 1$$

$$16/2 = 8 \text{ resto } 0$$

$$8/2 = 4 \text{ resto } 0$$

$$4/2 = 2 \text{ resto } 0$$

$$2/2 = 1 \text{ resto } 0$$

$$1/2 = 0 \text{ resto } 1$$

$$N(2) = 100001011110$$

### Números fraccionarios:

$$\text{Fracción}(10) = 0,567$$

$$0,567 \times 2 = 1,134 \text{ rebose } 1$$

$$0,134 \times 2 = 0,268 \text{ rebose } 0$$

$$0,268 \times 2 = 0,536 \text{ rebose } 0$$

$$0,536 \times 2 = 1,072 \text{ rebose } 1$$

$$0,072 \times 2 = 0,144 \text{ rebose } 0$$

$$0,144 \times 2 = 0,288 \text{ rebose } 0$$

$$0,288 \times 2 = 0,576 \text{ rebose } 0$$

$$0,576 \times 2 = 1,152 \text{ rebose } 1$$

$$\text{Fracción (2)} = 0,10010001$$

### Decimal A Hexadecimal

$$N(10) = 3879,654$$

Entonces:

$$3879/16 = 242 \text{ resto } 7$$

$$242/16 = 15 \text{ resto } 2$$

$$15/16 = 0 \text{ resto F}$$

y además:

$$0,654 \times 16 = 10,464 \text{ rebose A}$$

$$0,464 \times 16 = 7,424 \text{ rebose 7}$$

$$0,424 \times 16 = 6,784 \text{ rebose 6}$$

$$0,784 \times 16 = 12,544 \text{ rebose C}$$

y así sucesivamente.

$$N(16) = F27,A76C$$

## Codificación

Es necesario codificar para representar los caracteres alfanuméricos y de control.

El código más simple es el BCD (Binary Coded Decimal o Decimal Codificado En Binario)

### Código BCD

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

### Códigos Alfanuméricicos

Actualmente hay tres en uso:

- ASCII (American Standard Code For Information Interchange) (hay de 7 y de 8 bits, el octavo bit es el de paridad usado para control)
- EBCDIC (Extended Binary Coded DecimaL Interchange Code) (Es propio de ibm y de 8 bits 00-ff)
- UNICODE (Universal Code) (Usa dos bytes por carácter)

### Operaciones Entre Números Codificados

Supongamos una máquina que opera en bcd, al realizar una suma se presentan una de tres condiciones:

- Resultado comprendido 0 – 9 que es correcto
- Resultado entre 10 – 15 erróneo
- Resultado mayor de 16 erróneo

$$4 + 3 = 7 \quad 0100 + 0011 = 0111 \quad \text{Correcto}$$

$$6 + 8 = 14 \quad 0110 + 1000 = 1110 \quad \text{Pero en BCD el 14 se debe escribir como:} \\ 0001 \ 0100 \text{ por lo que se debe corregir}$$

$$8 + 9 = 17 \quad 1000 + 1001 = 10001 \quad \text{Que se debería escribir } 0001 \ 0111$$

## Corrección

Cuando pasa de 9, se suma el binario 0110 que equivale al 6

Entonces:  $14 + 6 \rightarrow 1110 + 0110 = 0001\ 0100$

Cuando es 16 o más, también sumamos 6, pero el arrastre ya lo tenemos:

$17 + 6 \rightarrow 0001\ 0001 + 0000\ 0110 = 0001\ 0111$

## Números En Coma Flotante

Es la que también se llama notación científica.

$$N = \text{MANTISA} \times 10^{(\text{EXPONENTE})}$$

EJEMPLO:

$$N = 0,312 \times 10^8$$

Que es igual a: 31.200.000

El exponente puede ser positivo o negativo:

- Cuando es positivo, es la cantidad de lugares hacia la derecha que debe moverse la coma decimal.
- Cuando es negativo, es la cantidad de lugares hacia la izquierda que debe moverse la coma decimal.

La mantisa es la cantidad significativa del número. Puede ser positiva o negativa, lo que hace al número positivo o negativo. Por convención siempre es fraccionaria, o sea que su valor estará comprendido entre 0 y 1.

## Operaciones En Coma Flotante

Para sumar y restar es necesario primero alinear las mantisas, lo cual significa modificar los exponentes.

$$\begin{aligned}0,234 \times 10^{-5} + 0,81 \times 10^{-4} &= \\0,0234 \times 10^{-4} + 0,81 \times 10^{-4} &= 0,8334 \times 10^{-4} \\0,556 \times 10^5 - 0,467 \times 10^3 &= 0,556 \times 10^5 - \\-0,00467 \times 10^5 &= 0,55133 \times 10^5\end{aligned}$$

25

En el caso del producto y la división, las mantisas se operan normalmente y los exponentes se suman o se restan según corresponda.

$$0,24 \times 10^{-3} \times 0,45 \times 10^4 = 0,108 \times 10^1$$

$$0,32 \times 10^5 / (0,71 \times 10^8) = 0,45 \times 10^{-3}$$

## Normas IEEE Para Números En coma Flotante

IEEE = Instituto De Ingenieros En Electrónica Y Electricidad (USA)

Son normas universalmente adoptadas

- Formato Simple: 1 BIT de signo, 8 BITS de exponente polarizado y 23 BITS para la mantisa o fracción
- Formato Doble: 1 BIT de signo, 11 BITS para el exponente polarizado y 52 BITS para la mantisa.

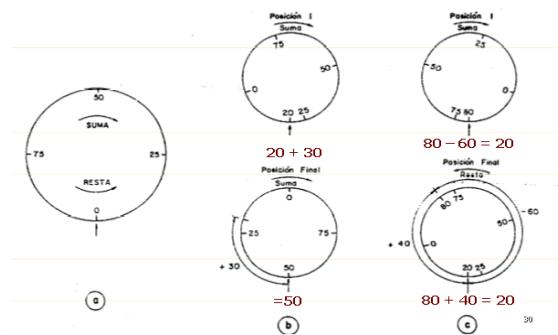
## Exponente Polarizado

Significa que no se emplea el signo del mismo sino que se fija un cierto valor como exponente cero.

Para valores mayores el exponente es positivo, para valores menores de ese valor el exponente es negativo.

Ejemplo: Si el valor es 100, el valor medio es 50, y el exponente 40 significa que el valor es -10, mientras que para el exponente igual a 75, el valor será +25

## Complementación



El esquema muestra una rueda numerada desde 0 hasta 99

Si se gira en el sentido horario se suma y si se gira en sentido antihorario se resta

En la figura (b) se muestra el caso de  $20 + 30 = 50$

En la figura (c) se muestra que se puede obtener 20 por la resta 80 - 60 o sumándole a 80 el complemento a 100 de 60 que es 40,  $80 + 40 = 120$  pero al pasar de 100 vuelve a cero por lo que queda 20 de resultado (implica despreciar el 1 de acarreo)

Complemento de N = MÓDULO – N

Módulo = Máxima cantidad expresable con esa cantidad de cifras + 1

### Complemento Restringido

Al usar el complemento según la definición, siempre tendríamos cuentas sobre operadores muy largos, por ejemplo: en un sistema de 12 dígitos, el módulo sería 1.000.000.000 y el complemento de 45 sería 999.999.955

Es más conveniente definir el complemento restringido, que puede ser a 10 o a 9.

En el complemento a diez se restan todas las cifras del número de 9 y la menos significativa de diez.

En el complemento a nueve se restan todas las cifras del número de 9.

### Aplicaciones De Complemento

Se utiliza para convertir restas en sumas.

Se el número con signo negativo, y se procede a sumarlos.

- En el caso del complemento a 10, si la suma tiene arrastre, se desprecia el mismo y lo que queda es el resultado. Cuando no tiene arrastre, el resultado es el complemento de la suma.

$$\begin{array}{r}
 +8.276 \rightarrow +8.276 \\
 -5.421 \rightarrow C10 = +4.579 \\
 2.855 \quad (1) 2.855 = 2.855 \\
 \text{El acarreo (1) no se tiene en cuenta} \\
 \\ 
 +4.423 \rightarrow +4.423 \\
 -8.156 \rightarrow C10 = +1.844 \\
 -3.733 \quad 6.267 \rightarrow C10 = -3.733 \\
 C10(C10(N)) = N, \text{ me indica cuál número negativo está representando.}
 \end{array}$$

- En el caso del complemento a 9, si hay arrastre se suma un uno al resultado, y si no tiene arrastre se vuelve a complementar a 9.

$$\begin{array}{r}
 +8.726 \rightarrow +8.726 \\
 -5.421 \rightarrow C9 \rightarrow +4.578 \\
 +3.305 \quad +1.3.304 \\
 \text{Se corrige sumando el acarreo} +1 \\
 +3.305 \\
 \\ 
 -4.738 \rightarrow C9 \rightarrow +5.261 \\
 +2.245 \rightarrow +2.245 \\
 -2.493 \quad +7.506 \rightarrow C9 \rightarrow -2.493 \\
 C9(C9(N)) = N, \text{ me indica cuál número negativo}
 \end{array}$$

## Complemento En Binarios

Es lo mismo que entre números decimales, pero el complemento restringido será a dos ó a uno.

- En el caso del complemento a 2, se busca de derecha a izquierda el primer 1 el cual se mantiene y a partir de ahí los demás cambian, los 0 → 1 y los 1 → 0.

### COMPLEMENTO A DOS

$$\begin{array}{r} +10100 \rightarrow +10100 \\ -01001 \rightarrow C2 \rightarrow -10111 \\ +01011 \quad (1)01011 \rightarrow 1011 \\ \text{El acarreo (1) no se tiene en cuenta} \end{array}$$

$$\begin{array}{r} -10101 \rightarrow C2 \rightarrow +01011 \\ +10010 \rightarrow +10010 \\ -00011 \quad +11101 \rightarrow C2 \rightarrow -11 \\ C2(C2(N)) = N, \text{ me indica cuál número binario negativo está representando.} \end{array}$$

- En el caso del complemento a 1, se cambian todos los 0 → 1 y los 1 → 0.

### COMPLEMENTO A 1

$$\begin{array}{r} +11101 \rightarrow +11101 \\ -01010 \rightarrow C1 \rightarrow +10101 \\ +10011 \quad (1)10010 \\ \text{Se corrige sumando el acarreo} \quad \begin{array}{r} +1 \\ +10011 \end{array} \end{array}$$

$$\begin{array}{r} -10110 \rightarrow C1 \rightarrow +01001 \\ +10010 \rightarrow +10010 \\ -00100 \quad +11011 \rightarrow C1 \rightarrow -100 \\ C1(C1(N)) = N, \text{ me indica cuál número binario negativo está representando.} \end{array}$$

## Operaciones Entre Números Decimales

### Suma Y Resta

No hay ninguna variante respecto de los métodos normales.

### Multiplicación

- Método Normal

<b>8543</b>	<b>Multiplicando</b>
<b>x 234</b>	<b>Multiplicador</b>
<b>34172</b>	<b>1er producto parcial</b>
<b>25629</b>	<b>2do producto parcial</b>
<b>17086</b>	<b>3er producto parcial</b>
<b>1999062</b>	<b>Producto</b>

- 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 por un lado los productos parciales sin arrastres y por otro los arrastres, para luego sumar los resultados conjuntamente.

Ejemplo:	<b>8543</b>
	<b>x 234</b>
	<b>3211</b>
	<b>2062</b>
Componentes	<b>2110</b>
Izquierdas	<b>1100</b>
	<b>134311</b>
Componentes	<b>4529</b>
derechas	<b>6086</b>
	<b>655952</b>
	<b>134311</b>
	<b>1999062</b>

- Sumas Repetidas

Se trata de sumar el multiplicando tantas veces como diga el multiplicador, de la siguiente forma:

$$\begin{array}{r} 8543 \\ 8543 \\ 8543 \\ 8543 \quad 4 \quad = \quad 4 \\ 85430 \\ 85430 \\ 85430 \quad 3 \times 10 \quad = \quad 30 \\ 854300 \\ \hline 854300 \quad 2 \times 100 \quad = \quad 200 \\ \hline 1999062 \quad \quad \quad \quad \quad \quad 234 \end{array}$$

## División

- Método Normal

<b>Dividendo</b>	<b>452980</b>	<u><b>365</b></u>	<b>Divisor</b>
<b>1er resto parcial</b>	<b>879</b>	<b>1241</b>	<b>Cociente</b>
<b>2do resto parcial</b>	<b>1498</b>		
<b>3er resto parcial</b>	<b>380</b>		
<b>Resto</b>		<b>15</b>	

- 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:

<u>452980</u>	<u>365</u>	
- <u>365000</u>		
<u>87980</u>		
- <u>365000</u>		
<u>-277020</u>	Resto negativo, se debe reponer.	
+ <u>365000</u>	Cantidad de restas = 2 - 1 = 1	
<u>87980</u>		
- <u>36500</u>	Luego se debe desplazar el dividendo y restar.	
<u>51480</u>		
- <u>36500</u>		
<u>14980</u>		
- <u>36500</u>		
<u>-78480</u>	Resto negativo, se debe reponer.	
+ <u>36500</u>	Cantidad de restas = 3 - 1 = 2	
<u>14980</u>		
- <u>3650</u>	Se desplaza nuevamente y se vuelve a restar.	
<u>11330</u>		
- <u>3650</u>		
<u>7680</u>		
- <u>3650</u>		
<u>4030</u>		
- <u>3650</u>		
<u>380</u>		
- <u>3650</u>		
<u>-3270</u>	Resto negativo, se debe reponer.	
+ <u>3650</u>	Cantidad de restas = 5 - 1 = 4	
<u>380</u>		
- <u>365</u>	Se desplaza y se resta.	
<u>15</u>		
- <u>365</u>		
<u>-350</u>	Resto negativo, reponer	
+ <u>365</u>	Cantidad de restas = 2 - 1 = 1	
<u>15</u>		

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.

- División Por Restas Sucesivas, 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:

$$\begin{array}{r} 452980 \mid 365 \\ -365000 \\ \hline 87980 \\ -365000 \\ \hline -277020 \text{ Restas } 2 - 1 = 1 \text{ (Primer dígito del cociente)} \\ + 36500 \text{ Se desplaza el divisor y se suma} \\ \hline -240520 \\ + 36500 \\ \hline -204020 \\ + 36500 \\ \hline -167520 \\ + 36500 \\ \hline -131020 \\ + 36500 \\ \hline -94520 \\ + 36500 \\ \hline -58020 \\ + 36500 \\ \hline -21520 \\ + 36500 \text{ Sumas } 8, \text{ el complemento a } 10 \text{ de } 8 = 2 \text{ (Segundo dígito)} \\ + 14980 \text{ Ahora hay resto positivo, por lo que se desplaza y se resta} \\ \hline -3650 \\ +11330 \\ -3650 \\ \hline + 7680 \\ -3650 \\ \hline + 4030 \\ -3650 \\ \hline + 380 \\ -3650 \text{ Restas } 5 - 1 = 4 \text{ (Tercer dígito del cociente)} \\ -2540 \\ + 365 \text{ Se desplaza y se suma nuevamente.} \\ -2175 \\ + 365 \\ -1810 \\ + 365 \\ -1445 \\ + 365 \\ -1080 \\ \hline + 365 \\ -350 \\ + 365 \text{ Sumas } 7, \text{ Complemento a } 10 \text{ de } 7 = 3 \text{ (Cuarto dígito)} \\ + 15 \end{array}$$

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

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.

## Operaciones Entre Números Binarios

### Suma Y Resta

No hay ninguna variante respecto de los métodos normales.

### Multiplicación

La tabla de multiplicar entre binarios es la siguiente:

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

- Método Normal

$$\begin{array}{r} \begin{array}{r} 1 & 0 & 1 & 1 & & 1 \\ \times & 0 & 1 & 1 & 0 & \\ \hline 0 & 0 & 0 & 0 & & \\ 1 & 0 & 1 & 1 & & \\ 1 & 0 & 1 & 1 & & \\ \hline 0 & 0 & 0 & 0 & & \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 66 \end{array} \end{array}$$

- Sumas Repetidas

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.

## División

Es tan sencilla como la multiplicación binaria, y puede hacerse también por restas repetidas.

$$\begin{array}{r} 1011010 \quad | 1110 \quad 90 \quad | 14 \\ \underline{1110} \quad \quad 110 \quad \quad 6 \quad 6 \\ 100010 \\ \underline{1110} \\ 00110 \end{array}$$

## Notación En Complemento A Dos

Es usada en máquinas de palabra muy corta, por ejemplo de 8 BITS

Consiste en incorporar el bit de signo al número

Permite aumentar la cantidad de valores distintos

Por Ejemplo:

- En palabras de 8 BITS, el BIT de mayor peso se supone que representa el signo.
- Así pueden representar 128 valores positivos (incluyendo al cero) y 128 negativos aprovechando al máximo la capacidad del BYTE.

#### EN CASO DE PALABRAS DE CUATRO BITS

DECIMAL	MAGNITUD Y SIGNO	COMPLEMENTO A 2
+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

#### DETERMINACIÓN GRÁFICA

-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

#### Multiplicación De Números En Complemento A Dos

Se presentan inconvenientes cuando el multiplicador es negativo. Por Ejemplo:

$$\begin{array}{r}
 \underline{\phantom{0}0110} \quad = 6 \\
 \times \underline{\phantom{0}1001} \quad = -7 \\
 \hline
 \phantom{0}0110 \\
 \underline{\phantom{0}011000} \\
 \phantom{0}0110110 \quad = 54
 \end{array}$$

**Corrección:** Una de las formas es la de descomplementar al número negativo, multiplicar y luego volver a complementar el resultado dado que el mismo debe ser negativo.

$$\begin{array}{r}
 & 0110 \\
 1001 \text{ DESC. } & \underline{x \ 0111} \\
 & 0110 \\
 & 0110 \\
 & 0110 \\
 & 0000 \\
 \hline
 & 0101010 = 42
 \end{array}$$

COMPLEMENTANDO

$$\begin{array}{r}
 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \\
 -64 \quad +16 \quad +4 \quad +2 \quad 0 \\
 \hline
 & = -42
 \end{array}$$

### Método De Booth

Es utilizado para simplificar el proceso anterior.

Consiste en llenar todos los espacios de los registros de los productos parciales con unos.

Estos registros deben tener longitud doble a la de los operandos.

$$\begin{array}{r}
 1 \ 0 \ 0 \ 1 = -7 \\
 \times 0 \ 0 \ 1 \ 1 = 3 \\
 \hline
 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 = -7 \\
 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 = -14 \\
 \hline
 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 = -128 + 64 + 32 + 8 + 2 + 1 = -21
 \end{array}$$

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

## NIVEL LÓGICO-DIGITAL (B2) (IMPLEMENTACIÓN DE LAS OPERACIONES)

### Compuertas Lógicas

Tabla de verdad	
A	X
0	0
1	1

BUFFER



Tabla de verdad	
A	X
0	1
1	0

NOT



Tabla de verdad AND		
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

AND



Tabla de verdad NAND		
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

NAND



Tabla de verdad OR		
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

OR



Tabla de verdad NOR		
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

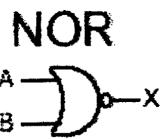


Tabla de verdad X-OR		
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

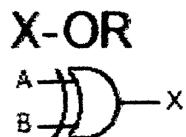
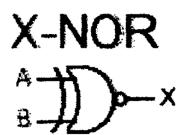


Tabla de verdad X-NOR		
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

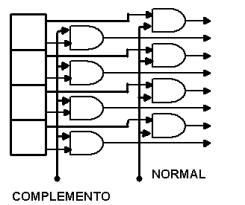
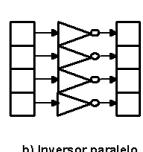
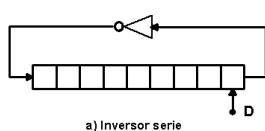


## Complementadores

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.

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.

## Complementadores Binarios



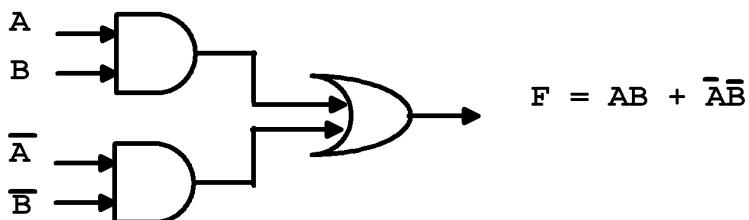
En la figura (a) muestra el inversor serie al cuál debe aplicarse una cantidad de pulsos de desplazamiento (d) que corresponde con el número de bits del registro.

En la figura (b) cada bits del registro tiene colocado un inversor por lo cual se obtiene el complemento a uno en el registro de la derecha.

En la figura (c) se utiliza la salida complementada que tienen los multivibradores conectando cada bit mediante una compuerta and usada como control para el complemento a uno o la salida común también controlada por una compuerta and, en cada bit, para obtener el valor normal.

### Complementadores Decimales

#### Comparador Binario



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 <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	C <sub>3</sub>	C <sub>2</sub>	C <sub>1</sub>	C <sub>0</sub>	Dec
-----	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----

- Tabla de la verdad del complementador decimal.

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

- Funciones

$$C_0 = 0, 2, 4, 6, 8 + TNV$$

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

$$C_2 = 2, 3, 4, 5 + TNV$$

$$C_3 = 0, 1 + TNV$$

- Minimizando

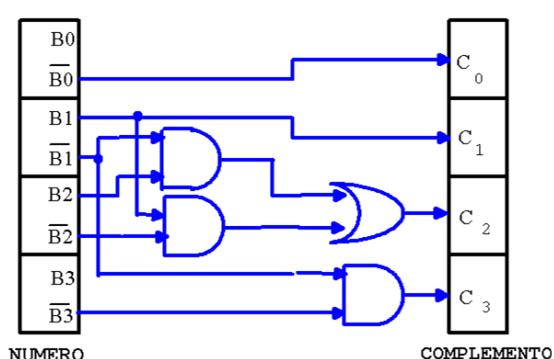
$$C_0 = \overline{B}_0$$

$$C_1 = B_1$$

$$C_2 = \overline{B}_1 B_2 + B_1 \overline{B}_2$$

$$C_3 = \overline{B}_1 \overline{B}_2 \overline{B}_3$$

- Esquema Del Circuito

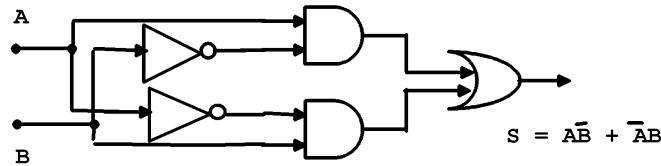


Sumador Binario

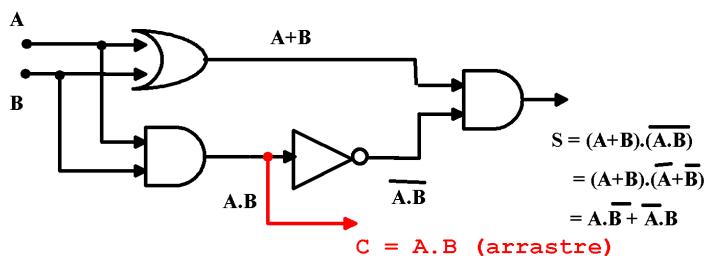
+

0	0	0
0	1	1
1	0	1
1	1	10

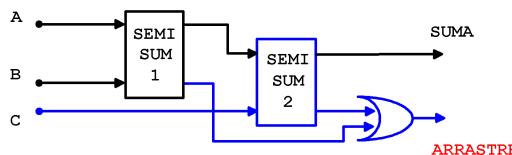
### Semisumador



### Semisumador Con Indicación De Arrastre



### Sumador Completo Implementado Con Bloques Semisumadores



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.

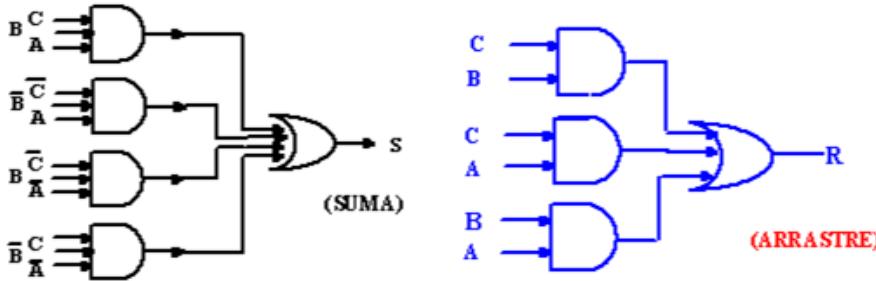
### Sumador Completo Implementado Con Compuertas

Contempla solo dos niveles de compuertas.

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

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

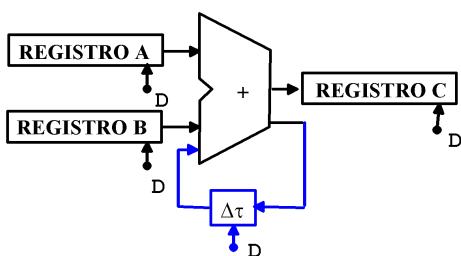
$$S = 1, 2, 4, 7 = \overline{ABC} + \overline{ABC} + \overline{ABC} + ABC$$



### Mecanización De La Suma Entre 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.

#### Sumador Serie Binario



Se tiene a un sumador serie, que consta de tres registros ( A, B, C), un sumador completo y un elemento de retardo ( $\Delta\tau$ ) para aplicar el arrastre en la próxima suma.

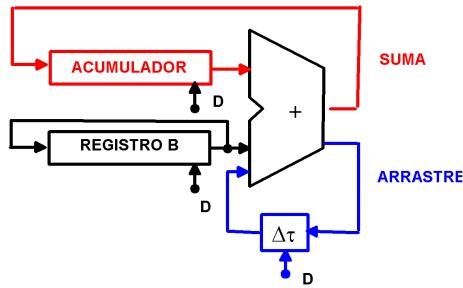
En el registro A se dispone un sumando, en el registro B, otro y en cada pulso de desplazamiento (D), los BITS menos significativos se suman, conjuntamente con el arrastre de la suma anterior, para ir quedando los resultados en el registro C, entrando por el BIT más significativo del registro.

A medida que los pulsos de desplazamiento (D) 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 lo tanto es posible reemplazar a este último por alguno

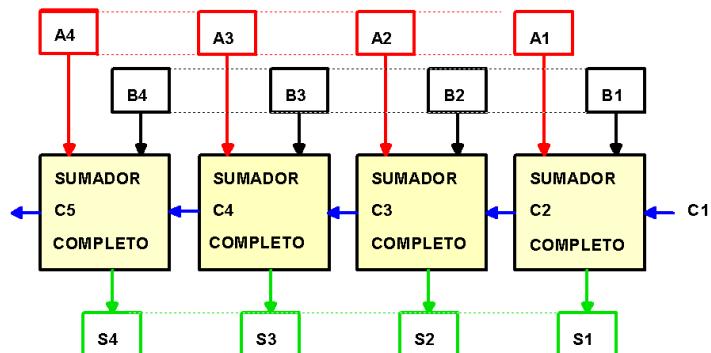
de los primeros. El registro donde se van acumulando los resultados se denomina registro acumulador.

Para realizar operaciones encadenadas uno de los sumandos se debe conservar, esto se logra colocando una realimentación en el registro B, como se muestra en la figura.

### Sumador Serie Binario Con Acumulador



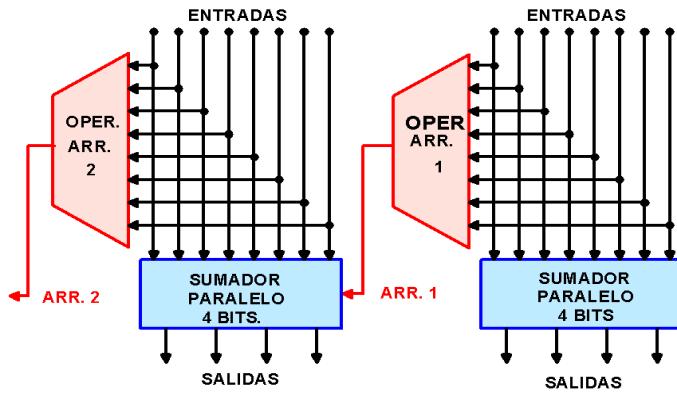
Sumador Paralelo



Se pueden colocar sumadores completos entre dos registros para realizar la suma en paralelo de cada conjunto de bits.

Hay que tener en cuenta el tiempo necesario para la propagación del arrastre, ya que cada suma no será correcta hasta que se le agregue el arrastre que le corresponde. Estos retardos introducidos, además de aumentar el tiempo de ejecución puede dar lugar a errores de lectura (si no son considerados).

Sumador Paralelo Con Anticipo Del Arrastre

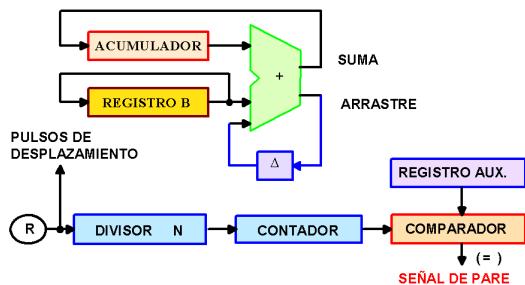


Para acelerar la suma, es necesario de alguna manera tratar que el arrastre no demore su cálculo, una forma, es anticiparlo en bloques cortos de sumadores, por ejemplo de cuatro unidades, de esta manera, el retardo siempre será solamente el correspondiente a la determinación de tres arrastres, pues el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

### Multiplicación De Números 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.

### Multiplicación Por Sumas Sucesivas



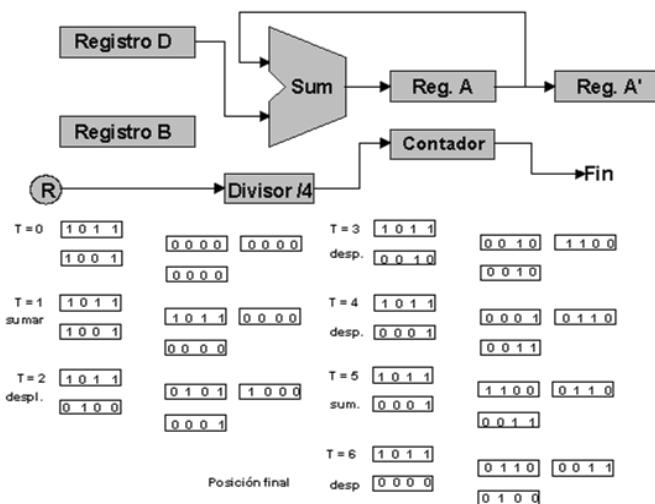
Consiste en sumar tantas veces el multiplicando como lo diga el multiplicador.

El multiplicando se coloca en el registro B, el multiplicador en el registro auxiliar, cada pulso del reloj (pulso de desplazamiento) que se recibe se suma un BIT del registro B con un BIT del acumulador

El divisor genera un pulso cada vez que transcurren tantos pulsos como posiciones binarias tenga el registro B (N). El contador contará la cantidad de sumas, cuando

esta sea igual al contenido del registro auxiliar, el comparador genera una señal y el proceso se detiene.

### Multiplicación Acelerada



El multiplicando se coloca en el registro D y se mantiene en el mismo.

El multiplicador se coloca en el registro B

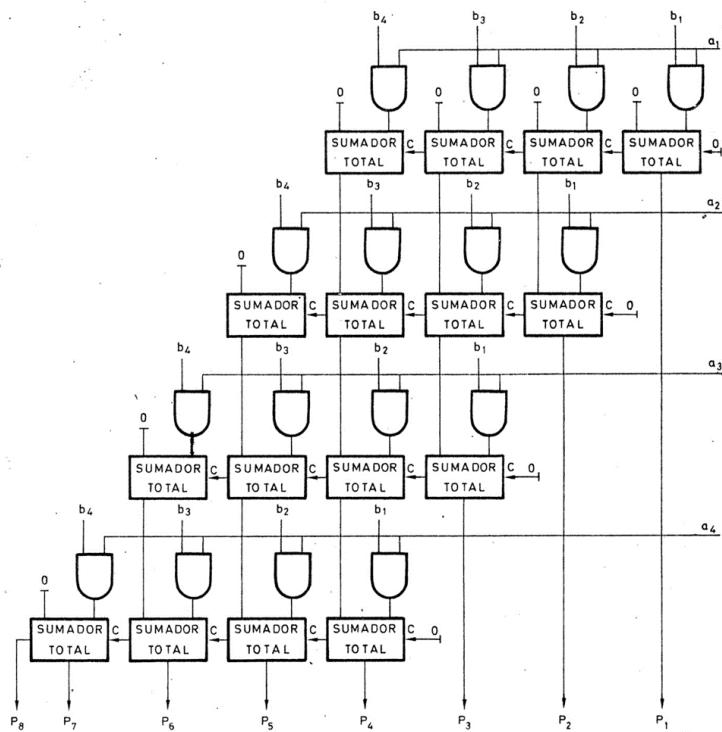
El registro A, para el resultado, tiene doble longitud, lo cual es lógico pues el producto de dos cantidades tiene una longitud igual al doble de la de cada uno de los operandos.

Cuando el bit menos significativo (LSB) del contenido del registro B es “1”, la acción provocada es la suma de  $\langle A \rangle$  y  $\langle D \rangle$ , 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.

Cuando el bit menos significativo (LSB) del contenido del registro B es “0”, la acción provocada es desplazar una posición a la derecha al registro B, A y A', al mismo tiempo que el contador cuenta el desplazamiento habido.

Los datos iniciales en decimal eran:  $11 \times 9 = 99$ , el resultado es 01100011, lo que significa:  $1 \times 2^6 + 1 \times 2^5 + 1 \times 2^1 + 1 \times 2^0 = 64 + 32 + 2 + 1 = 99$

## Multiplicador Celular



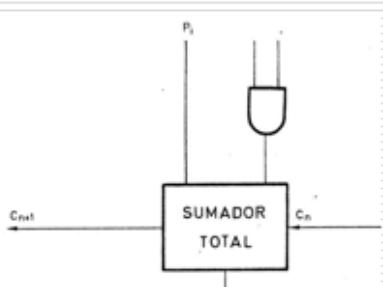
Multiplicando b4 b3 b2 b1

Multiplicador a4 a3 a2 a1

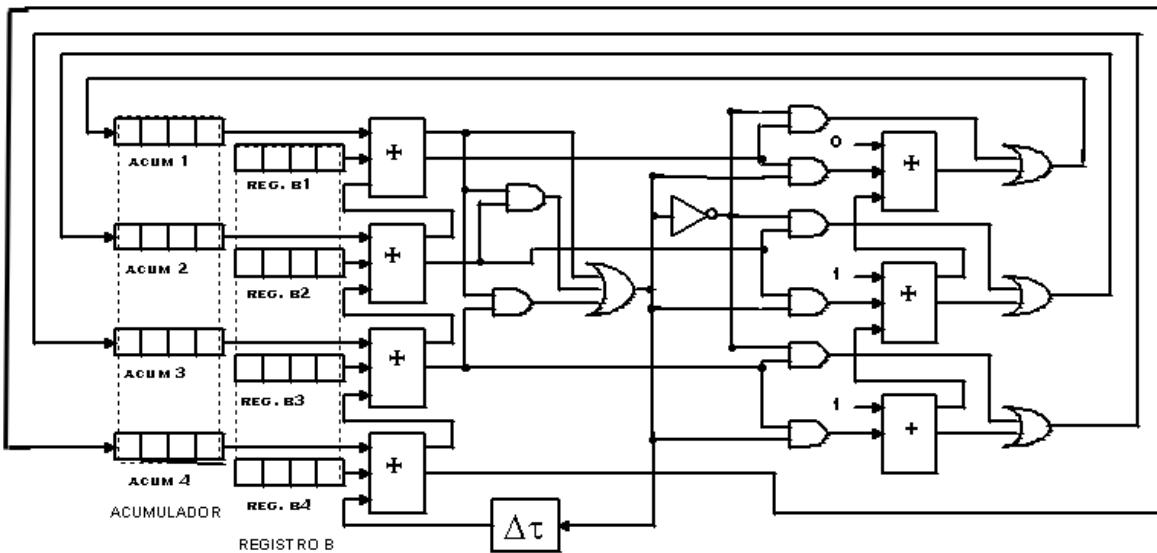
P8 P7 P6 P5 P4 P3 P2 P1 Es el resultado

LA COMPUERTA AND Y EL SUMADOR TOTAL SE UTILIZA COMO ELEMENTO BÁSICO (CÉLULA) PARA REALIZAR EL MULTIPLICADOR.

PODRÍA REALIZARSE CON MENOS ELEMENTOS, POR EJEMPLO LA PRIMER FILA DE SUMADORES TOTALES NO SON NECESARIOS POR EL ALGORITMO, ADEMÁS TODOS LOS SUMADORES TOTALES QUE TIENEN UNA ENTRADA FIJA CON CERO PODRÍAN REEMPLAZARSE CON SEMISUMADORES, PERO SE TENDRÍAN DOS TIPOS DE CIRCUITOS Y NO SE UTILIZARÍA EL BLOQUE CELULAR, QUE FACILITA LA CONSTRUCCIÓN DEL CIRCUITO



## Sumador BCD



Registro acumulador de cuatro dígitos BCD

Registro B de cuatro dígitos BCD

Cada uno de los registros en realidad está formado por cuatro sub-registros, en los cuales se almacena cada uno de los dígitos BCD, que como sabemos es de cuatro bits cada uno.

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 arrastre del sumador dispuesto en la parte superior, corresponde a cualquier resultado que pase de 15, o sea el que llamamos arrastre de 16.

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.

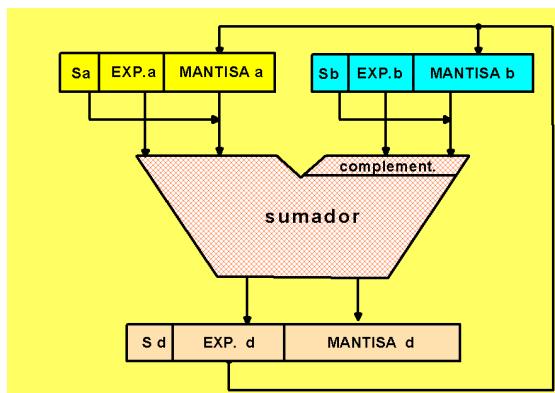
Si no ocurre lo anterior, y la suma es la correcta, las compuertas la derivan directamente al acumulador, que como siempre es el encargado de guardar los resultados.

El dígito binario de menor peso del resultado será siempre correcto, pues no pasa por un sumador adicional, dado que se le suma cero.

Mecanización De Las Operaciones Con Números Expresados En Coma Flotante  
 Disposición de los campos de un registro para números en coma flotante, según la norma IEEE 755.

## S | EXPONENTE | MANTISA

Diagrama Lógico Simplificado De La Ula Para Números En Coma Flotante



### Operaciones

- Multiplicación Y División

Se realiza el producto o la división de las mantisas.

Si es producto se suman los exponentes y si es división se restan los exponentes (el del divisor del dividendo)

- Suma Y Resta

En primer lugar se igualan los exponentes, para lo cual se restan: E<sub>a</sub> - E<sub>b</sub> = C<sub>d</sub>

```

SI ES Cd = 0, ENTONCES SON IGUALES
SI ES Cd > 0, ENTONCES ES: Ea > Eb
SI ES Cd < 0, ENTONCES ES: Eb > Ea

```

- ❖ Si son iguales se deja todo como está.
- ❖ Si son distintos, se toma la mantisa del menor y se la lleva a un registro de desplazamiento Rd.

Se coloca Cd en un contador, y a cada pulso de reloj se desplaza un lugar a la derecha la mantisa y se va descontando una unidad en el contador, cuando este llega a cero, tenemos igualadas las mantisas y podemos operar.

### Normalización

Luego de efectuar la operación indicada, suma o resta, se deberá normalizar el número, o sea llevar la mantisa a la expresión normal, que es fraccionaria, y colocar el exponente debido.

# UNIDAD DE CONTROL

## UNIDAD DE CONTROL (A)

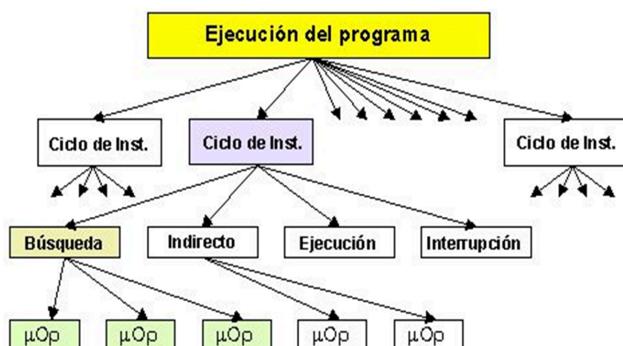
### Micro-Operaciones

El ciclo de máquina está formado por dos subciclos: búsqueda y ejecución

Cada subciclo a su vez origina cierta cantidad de pequeñas operaciones tales como: habilitación de compuertas, remisión de pulsos de desplazamiento, etc.

Este conjunto de órdenes, necesarias para el cumplimiento de un ciclo de instrucción, puede ser tomado como formado por un conjunto de micro-operaciones.

### Elementos De Ejecución De Un Programa



La ejecución de un programa consiste en una secuencia de ciclos de instrucción con una instrucción de máquina por ciclo.

Cada ciclo de instrucción puede subdividirse en cuatro sub-ciclos: búsqueda, indirecto, ejecución e interrupción , de los cuales búsqueda y ejecución siempre ocurren.

### Subciclo De Búsqueda

Es utilizado para buscar instrucciones en memoria.

Los registros involucrados en la operación son:

- REGISTRO DE DIRECCIONES DE MEMORIA (Memory Address Register (MAR)): Conectado a las líneas del bus de direccionamiento y especifica la dirección de memoria donde se debe escribir o leer.

- REGISTRO BUFFER DE MEMORIA (Memory Buffer Register (MBR)): Está conectado a las líneas del bus de datos, conteniendo el valor a ser almacenado o el último valor leído en memoria.
- CONTADOR DE PROGRAMA (Program Counter (PC)): Retiene la dirección de la próxima instrucción a ser buscada.
- REGISTRO DE INSTRUCCIONES (Instrucción Register (IR)): Retiene la última instrucción buscada.

Secuencia de eventos:

- La dirección de la próxima instrucción a ser ejecutada es tomada del pc, y entregada al bus de direcciones, al cual está conectado el mar.
- Se emite desde la unidad de control el mando leer, el que es entregado al bus de control. Es ubicada la dirección en la memoria y el contenido de la correspondiente posición es entregado al MBR, el cual está conectado al bus de datos.
- Se suma una unidad al contenido del pc.
- Se pasa el contenido del MBR por medio del bus de datos al IR.

### Paralelismo

Los eventos indicados, cuando no se superponen, pueden realizarse en forma simultánea, o sea al mismo tiempo.

La notación t1, t2, t3, representa distintas unidades de tiempo.

### Simbólicamente Sub-Ciclo Búsqueda

t1 : MAR <-- (PC)

t2 : MBR <-- (Memoria)

PC <-- (PC) + 1

t3 : IR <-- (MBR)

### Sub-Ciclo Indirecto

- El campo dirección del Registro de instrucciones es transferido al registro de dirección de memoria
- Se emite desde la unidad de control el mando LEER, el que es entregado al bus de control. Es ubicada la dirección en la memoria y el contenido de la correspondiente posición es entregado al MBR.
- Se pasa el contenido del MBR por medio del bus de datos al IR, actualizando su campo de dirección.

El IR está ahora en el mismo estado como si el direccionamiento indirecto no hubiera sido usado.

Corresponde a la búsqueda del o de los operandos cuando la instrucción especifica un direccionamiento indirecto.

t1 : MAR <-- (IR(Dirección))

t2 : MBR <-- (Memoria)

t3 : IR(Dirección) <-- (MBR(Dirección))

### Subciclo De Ejecución

En esta etapa las instrucciones son resueltas, pero no todos los tipos de instrucción se resuelven de la misma forma, ya que la forma de utilizar el hardware dependerá de la función de cada una de ellas, en general tenemos cuatro tipos de instrucciones:

**Instrucciones de movimiento de bits:** En el cual se manipula el orden de los bits que contienen el dato.

**Instrucciones aritméticas:** Donde se realizan operaciones matemáticas y también lógicas, estas se solucionan en las llamadas ALU o unidades aritmético-lógicas

**Instrucciones de salto:** En la que se cambia al siguiente el valor del contador de programa, lo que permite utilizar el código de manera recursiva.

**Instrucciones a memoria:** Son con las que el procesador lee y escribe la información de la memoria del sistema.

Para cada uno de los códigos operativos (opcodes) hay una secuencia específica de micro-operaciones para lograr la ejecución de la instrucción que dicho código representa.

Comienza con IR conteniendo al código operativo de la instrucción ADD R1,X

- El campo de dirección del IR es cargado en el MAR.
- El contenido de la posición de memoria referenciada es leído quedando cargado en el MBR.
- El contenido del registro R1 es sumado al contenido del MBR por la ALU y el resultado es guardado en el registro R1.

La misma puede expresarse como:

ADD R1,X (suma el contenido de la posición de memoria X al registro R1)

t1 : MAR <-- (IR(Dirección))

t2 : MBR <-- Memoria

t3 : R1 <-- (R1) + (MBR)

## Instrucción ISZ (Increment and Skip if Zero)

### ISZ X

Comienza con IR conteniendo al código operativo de la instrucción ISZ X ( el contenido de la locación x es incrementado en 1, si el resultado es cero, la próxima instrucción es saltada)

- El campo de dirección del IR es cargado en el MAR.
- El contenido de la posición de memoria referenciada es leído quedando cargado en el MBR.
- El contenido del MBR es sumado uno por la ALU y el resultado es guardado en MBR.
- El contenido del MBR es grabado en la posición de memoria referenciada por el MAR. El contenido del PC es incrementado en 1 si el contenido del MBR es cero.

La posible secuencia de micro-operaciones, puede ser:

t1 : MAR <-- (IR(dirección))

t2 : MBR <-- Memoria

t3 : MBR <-- (MBR) + 1

t4 : Memoria <-- (MBR)

Si ((MBR)=0) luego (PC<--(PC)+1)

## Instrucción BSA (Branch and Save Address)

Bifurcar para ir a subrutina.

Comienza con IR conteniendo al código operativo de la instrucción BSA X ( la dirección de la instrucción que sigue a la instrucción BSA es salvado en la locación X, y la ejecución continúa en la locación X+1. La dirección salvada será usada para el retorno)

- El campo de dirección del IR es cargado en el MAR.
- El contenido del PC es cargado en el MBR
- El campo de dirección del IR es cargado en el PC. El contenido del MBR es grabado en la posición de memoria referenciada por el MAR.
- El contenido del PC es incrementado en 1.

La dirección en el PC al comienzo de la instrucción es la dirección de la próxima instrucción en secuencia. Esta es salvada en la dirección designada en el IR. Esta última dirección es además incrementada para proporcionar la dirección de la instrucción para el siguiente ciclo de instrucción.

t1 : MAR <-- (IR(Dirección))

MBR <- (PC)  
t2 : PC <- (IR(Dirección))  
Memoria <- (MBR)  
t3 : PC <- (PC) + 1

### Sub-Ciclo De Interrupción

Al finalizar el sub-ciclo de ejecución se realiza un test para determinar si ha ocurrido alguna interrupción habilitada, si es así, se realiza el sub-ciclo de interrupción:

- El contenido del PC es transferido al MBR, de manera que pueda ser salvado para retornar de la interrupción.
- El MAR es cargado con la dirección en la cual el contenido del PC será salvado y el PC es cargado con la dirección de comienzo de la rutina de tratamiento de la interrupción.
- Se almacena el contenido del MBR, el cual contiene el viejo valor del PC, en la memoria.

t1 : MBR <- (PC)  
t2 : MAR <- Guarda-dirección  
PC <- Dirección-rutina  
t3 : Memoria <- (MBR)

### Ciclo De Instrucción

Cada fase del ciclo de instrucción puede ser descompuesta en una serie de micro operaciones elementales.

Para cada subciclo hay una secuencia preestablecida, por tanto hay una secuencia de microoperaciones para cada código operativo

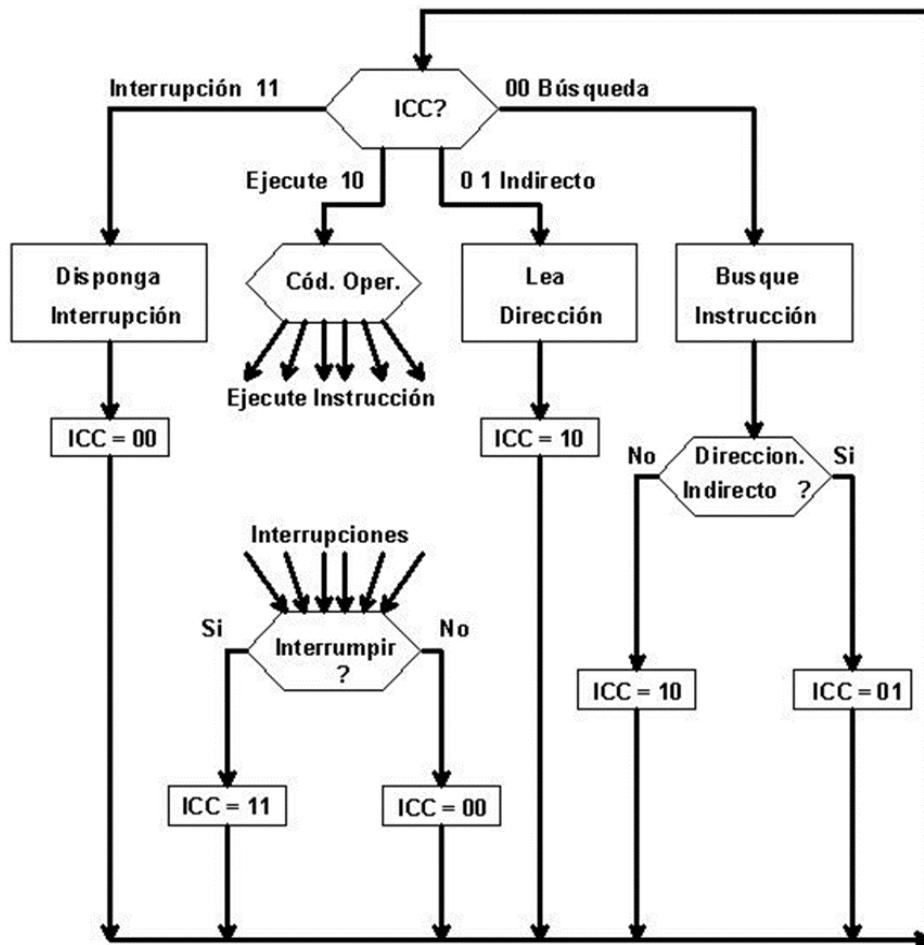
#### Código De Ciclo De Instrucción (Instruction Cycle Code (ICC))

Es un registro de dos BITS que indica el estado de la CPU para cada porción del ciclo que se está ejecutando.

Códigos:

- 00 : Búsqueda
- 01 : Indirecto
- 10 : Ejecute
- 11 : Interrupción

## Diagrama De Flujo Para El Ciclo De Instrucción



Al final de cada uno de los cuatro sub-ciclos, el registro ICC es cargado apropiadamente.

El sub-ciclo indirecto siempre es seguido por el de ejecución.

El sub-ciclo de interrupción siempre es seguido por el de búsqueda y finalmente para ambos, el de búsqueda y el de ejecución, el próximo sub-ciclo depende del estado del sistema.

## Control De La CPU

Para reducir a operaciones elementales, es necesario conocer cuales son los requerimientos funcionales de la misma.

### **Pasos Para Caracterizar La Unidad De Control**

- 1 - Definir los elementos básicos de la UCP (CPU).

2 - Describir las micro-operaciones que la UCP (CPU) realiza.

3 - Determinar las funciones que la unidad de control debe cumplir para provocar la ejecución de las micro-operaciones indicadas.

#### Elementos de la UCP (CPU):

- ULA (ALU)
- UNIDAD DE CONTROL
- REGISTROS
- RUTAS DE DATOS INTERNAS
- RUTAS DE DATOS EXTERNAS

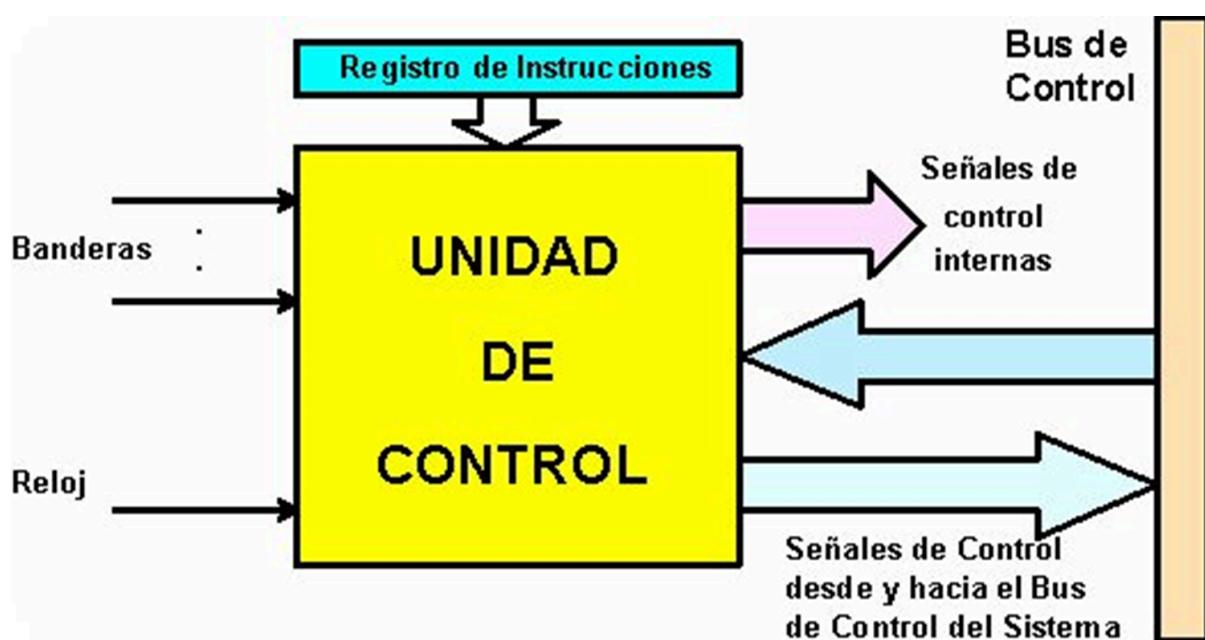
#### Micro-operaciones que debe ejecutar:

- Transferir datos desde uno a otro registro.
- Transferir datos de un registro a un módulo de e/s
- Transferir datos de un módulo de e/s a un registro
- Llevar a cabo una operación lógica o aritmética, utilizando registros para la entrada y la salida.

#### Tareas Básicas De La Unidad De Control:

Secuenciamiento: La Unidad de control hace que la UCP (CPU) vaya pasando por una secuencia de micro-operaciones, en la secuencia apropiada, basada en el programa que se está ejecutando.

Ejecución: La unidad de control hace que cada micro-operación sea cumplida.



## Señales De Control

Las entradas de la UC son:

- Reloj(Clock): La unidad de Control provee una o una serie de micro-operaciones en cada pulso de reloj, por tanto esto es denominado "ciclo del procesador" o "ciclo de reloj".
- Registro de Instrucciones (Instruction Register): El código operativo de la presente instrucción es utilizado para determinar cuáles son las micro-operaciones a llevar a cabo durante el ciclo de ejecución.
- Banderas (Flags): Estas son señales necesarias para que la unidad de control determine el estado de la UCP (CPU) y de los resultados anteriormente obtenidos en la ULA (ALU). Por ejemplo, en la instrucción ISZ (Increment-and-Skip-if-Zero), la unidad de control debe incrementar al PC si la bandera "0" está activa.
- Señales de Control del Bus de Control: Un sector del bus de control debe proveer las señales de solicitud de interrupción y de reconocimiento.

Las salidas de la UC son:

- Señales de control para la UCP (CPU): de las cuales hay dos tipos, las que provocan el movimiento de datos entre registros, y las que activan funciones específicas de la ULA (ALU).
- Señales de control para el bus de control: Son de dos tipos, las que son destinadas a la memoria y las que actúan sobre los módulos de E/S.

Los nuevos elementos introducidos en la figura del modelo general de unidad de control, son las señales de control, de las cuales, según dijimos, hay tres tipos: Las que activan funciones de la ULA (ALU), las que activan rutas de datos y las señales que son para el bus de control del sistema o para las interfaces externas.

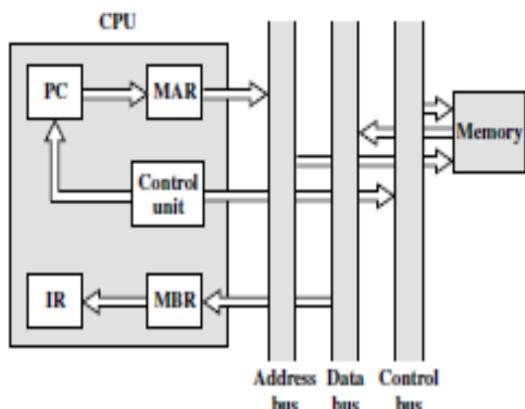
Todas estas señales son aplicadas directamente como señales binarias a la entrada de compuertas lógicas.

## Flujo De Datos Y De Control

Durante el ciclo de búsqueda, una instrucción es leída desde memoria. La figura muestra el flujo de datos durante este ciclo.

El PC contiene la dirección de la próxima instrucción a ser buscada. Esta dirección es llevada al MAR y colocada en el bus de direcciones (address bus).

La unidad de control pide una lectura a la memoria (señal de control colocada en el bus de control), y su resultado es colocado en el bus de datos y copiado en el MBR y luego es llevado al IR. Mientras tanto el PC es incrementado en uno, preparándolo para la próxima búsqueda



MBR = Memory buffer register

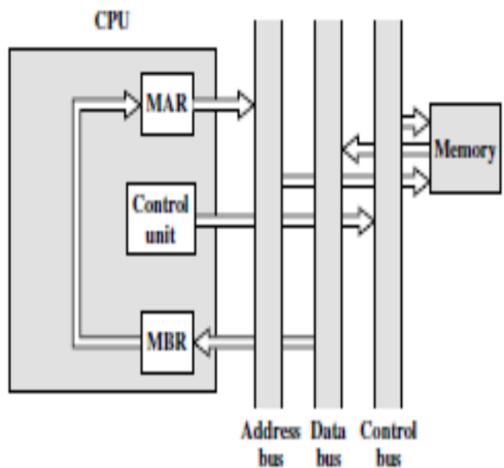
MAR = Memory address register

IR = Instruction register

PC = Program counter

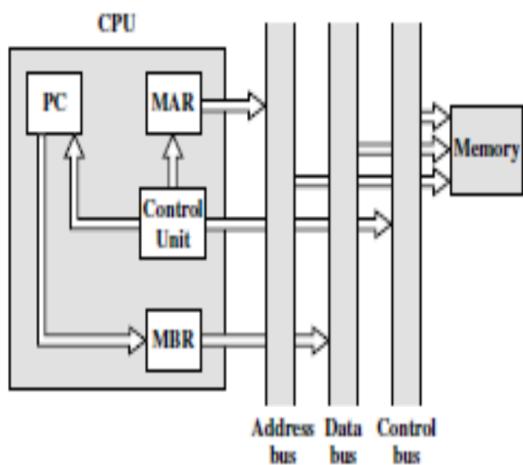
Una vez que un ciclo de búsqueda termina, la unidad de control examina el contenido del IR para determinar si se realiza el ciclo indirecto.

Como muestra la figura los BITS más a la derecha del MBR, los que contienen la dirección de referencia, son transferidos al MAR. Entonces la unidad de control solicita una lectura a la memoria (señal de control colocada en el bus de control), el resultado es la dirección del operando que es colocada en el bus de datos y de allí cargada en el MBR.



La figura muestra el flujo de datos del ciclo de interrupción.

El contenido en curso del PC debe ser salvado de forma que la UCP(CPU) pueda restaurar su actividad luego de la interrupción. Entonces, el contenido del PC es transferido al MBR y colocado en el bus de datos. Una locación de memoria especial reservada para este propósito es cargada en el MAR desde la unidad de control y luego es colocada en el bus de direcciones. La unidad de control solicita una escritura a la memoria (señal de control colocada en el bus de control). El PC es cargado con la dirección de comienzo de la rutina de tratamiento de interrupciones



### Implementaciones De La Unidad De Control

Hay una amplia variedad de técnicas que pueden ser aplicadas, la mayoría de las cuales cae en una de las siguientes dos categorías:

- IMPLEMENTACIÓN CABLEADA.
- IMPLEMENTACIÓN MICROPROGRAMADA.

## Control Cableado

Es esencialmente un circuito combinacional, donde las entradas son transformadas en un conjunto de señales lógicas, que son las señales de control.

Su control está basado en una arquitectura fija, es decir, que requiere cambios en el cableado si el conjunto de instrucciones es modificado o cambiado.

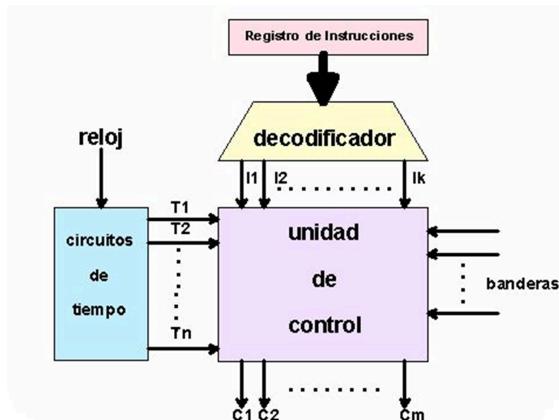
Consideremos ahora el registro de instrucciones (IR). La Unidad de Control hace uso del código operativo para llevar a cabo diferentes acciones para diferentes instrucciones. Mediante un decodificador, es posible lograr que para cada código, haya una sola salida. En general, el decodificador puede tener entradas binarias y dar  $2^n$  salidas también binarias, en consecuencia, habrá una salida por cada uno de los códigos operativos previstos. En la tabla siguiente tenemos un ejemplo de cómo ocurre este proceso, Decodificador con cuatro entradas (Inputs) y dieciséis salidas (Outputs):

$I_1$	$I_2$	$I_3$	$I_4$	$O_1$	$O_2$	$O_3$	$O_4$	$O_5$	$O_6$	$O_7$	$O_8$	$O_9$	$O_{10}$	$O_{11}$	$O_{12}$	$O_{13}$	$O_{14}$	$O_{15}$	$O_{16}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Un decodificador para una Unidad de Control, debe ser algo más sofisticado que el indicado, por cuanto se debe tener en cuenta las diferentes longitudes de los códigos operativos o su presencia en diferentes partes de la instrucción. De cualquier manera, de ésta forma es posible analizarlo.

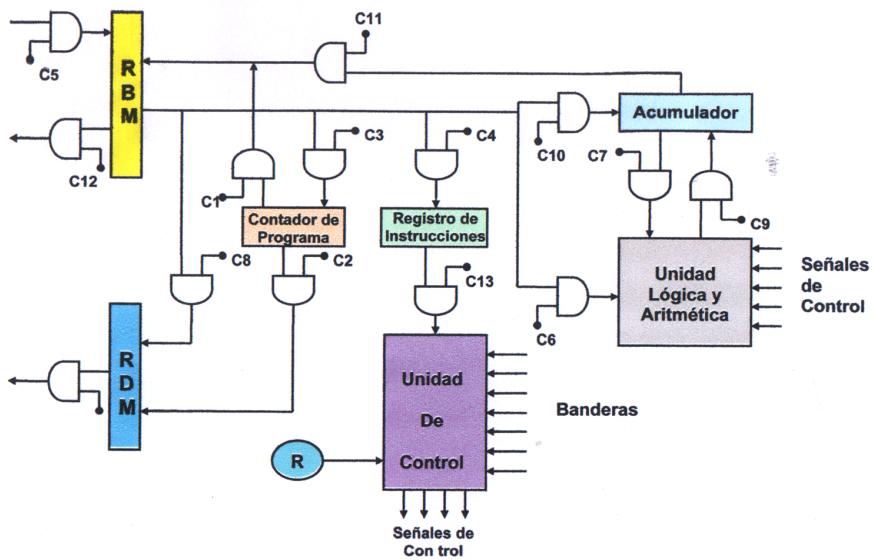
La sección de reloj de la Unidad de control, entrega una secuencia continua de pulsos, los cuales se emplean para determinar la duración de las micro-operaciones. Pero además, es necesario emitir diferentes señales de control, en diferentes instantes de tiempo, dentro de un ciclo de instrucción. Por tanto será necesario un contador y un circuito combinacional para permitir la presencia de estas señales.

Este circuito es conocido con el nombre de circuito de tiempo. Con estos dos refinamientos, el circuito simplificado de la Unidad de Control, queda como el indicado en la siguiente figura:



Supongamos una sencilla UCP (CPU), que posee un solo acumulador (AC), e indiquemos los caminos entre los distintos elementos. Estos serán los caminos de datos. Asimismo, tal como se indica en la figura, proveamos de compuertas a los caminos, compuertas (indicadas mediante círculos) que serán habilitadas o abiertas desde la unidad de control, mediante las señales  $C_1, \dots, C_{13}$ , y también se tienen otras señales, que son las que predisponen o conectan los circuitos internos de la ULA para que resuelva la operación indicada por la instrucción.

## Rutas De Datos Y Señales De Control



Sub-Ciclo	Micro-operaciones Timing	Señales de Control activas
Búsqueda:	$t_1 : \text{MAR} \leftarrow (\text{PC})$ $t_2 : \text{MBR} \leftarrow \text{Memoria}$ $\text{PC} \leftarrow (\text{PC}) + 1$ $t_3 : \text{IR} \leftarrow (\text{MBR})$	$C_2$ $C_5; C_R$ $C_4$
Indirecto:	$t_1 : \text{MAR} \leftarrow (\text{IR(Direcc)})$ $t_2 : \text{MBR} \leftarrow \text{Memoria}$ $t_3 : \text{IR(Dirección)} \leftarrow (\text{MBR(Dirección)})$	$C_8$ $C_5; C_R$ $C_4$
Interrupción:	$t_1 : \text{MBR} \leftarrow (\text{PC})$ $t_2 : \text{MAR} \leftarrow \text{Guarda-Direcc}$ $\text{PC} \leftarrow \text{Dir. Rutina}$ $t_3 : \text{Memoria} \leftarrow (\text{MBR})$	$C_1$ $C_{12}; C_W$

$C_R$  : Señal de control de lectura.

$C_W$  : Señal de control de escritura para el bus del sistema.

$C_5$ : Provoca que los datos sean leídos desde el bus externo y pasen al MBR.

Dos nuevas señales de control, P y Q, con la siguiente interpretación:

- $PQ = 00$ : Sub-ciclo de Búsqueda
- $PQ = 01$ : Sub-ciclo indirecto
- $PQ = 10$ : Sub-ciclo de ejecución
- $PQ = 11$ : Sub-ciclo de Interrupción

La expresión booleana que define a  $C_5$  es:

$$C_5 = \overline{P} \overline{Q} T_2 + \overline{P} Q T_2$$

Lo cual indica que la señal debe ser aplicada en el segundo período de tiempo, o sea en  $t_2$ , en ambos sub-ciclos, el de búsqueda y el indirecto. Esta expresión no está completa, dado que  $C_5$  también es necesaria durante el sub-ciclo de ejecución. Para nuestro ejemplo, supongamos que hay solo tres instrucciones que permiten la lectura desde la memoria, LDA, ADD y AND, por lo que podemos redefinir nuestra  $C_5$  como:

$$C_5 = \bar{P}\bar{Q}T_2 + \bar{P}QT_2 + P\bar{Q}(LDA + ADD + AND)T_2$$

El mismo proceso puede ser repetido para cada señal de control a generar, y entonces el resultado será un conjunto de ecuaciones booleanas a resolver para implementar el combinacional de control.

Para los actuales procesadores, con un complejo y extenso conjunto de instrucciones, la cantidad de ecuaciones booleanas a resolver se vuelve muy grande, de varios cientos de ellas, con lo cual el problema resulta altamente dificultoso.

### Control Microprogramado

En el caso de las unidades de control microprogramadas se utiliza un procesador que se encarga de realizar todas las funciones de la unidad de control, esto permite optimizar la forma en la que las instrucciones se van a ejecutar e incluso añadir nuevas instrucciones.

Tenemos el lenguaje de Microprogramación que hace referencia a las micro-operaciones. Cada línea describe un conjunto de micro-operaciones que debe ser llevada a cabo en un cierto instante de tiempo, y que es conocida como "Microinstrucción".

La secuencia de estas microinstrucciones es conocida con el nombre de "Microprograma", o también "Firmware". Un microprograma es la vía intermedia entre el Software y el Hardware. ¿Cómo podemos utilizar esto para diseñar unidades de control?

Consideremos que por cada micro-operación todo lo que tiene que hacer es generar un conjunto de señales de control.

De aquí que por cada micro-operación, cada línea de control emergente está en "1" o en "0". Esta condición puede ser representada por una palabra binaria, que tiene un dígito para cada una de estas líneas.

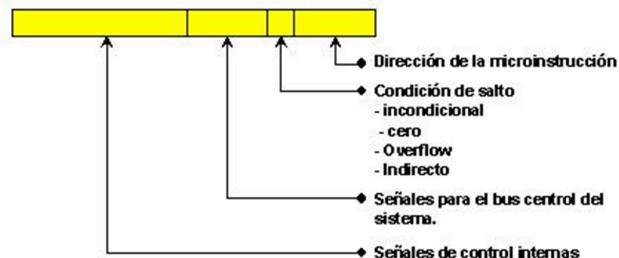
En consecuencia, podemos construir la llamada "palabra de control", en la cual cada bit representa una línea de control, por tanto cada micro-operación puede ser representada por una combinación diferente de unos y ceros en la palabra de control.

A continuación, podemos pensar en un listado de palabras de control, que lógicamente representará una secuencia de micro-operaciones llevadas a cabo por la unidad de control. Por otra parte, debemos reconocer que la secuencia de micro-operaciones no es fija, dado que a veces podemos tener un ciclo indirecto y a veces no.

Pongamos ahora la secuencia de palabras de control en una memoria, con cada palabra teniendo una sola y única dirección.

Luego, agreguemos un campo de dirección a cada una, indicando la dirección de la próxima palabra de control a ser ejecutada si se cumple con cierta condición, para lo cual deberán agregarse algunos bits para indicar ésta condición.

### Microinstrucción Horizontal



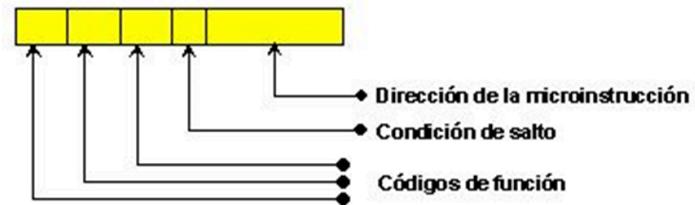
El resultado es conocido como "microinstrucción horizontal", y es mostrado en figura ANTERIOR, de donde se desprende el formato siguiente:

Hay un bit por cada línea de control interna y otro por cada línea de control del bus de control. También hay un campo de condición y hay un campo con la dirección de la microinstrucción que debe ejecutarse a continuación. Tal microinstrucción debe interpretarse como sigue:

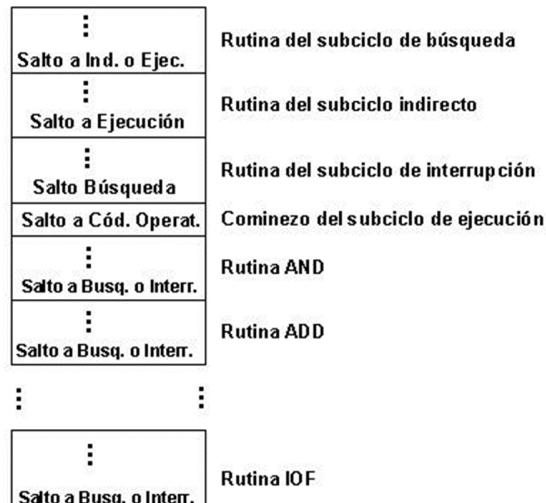
- Para ejecutar esta microinstrucción, active todas las líneas de control correspondientes al BIT "1" y desactive todas las líneas de control correspondientes al BIT "0".
- Si la condición indicada por los BITS de condición es falsa, ejecute la siguiente instrucción en la secuencia.
- Si la condición indicada por los BITS de condición es verdadera, la próxima microinstrucción a ser ejecutada es la indicada en el campo de dirección.

### Microinstrucción Vertical

El sistema de microinstrucción horizontal precisa muchas posiciones binarias, por tanto conviene emplear un sistema de codificación en los campos de las micro-operaciones, lo cual nos lleva a una "Microinstrucción vertical".



### Organización De La Memoria De Control



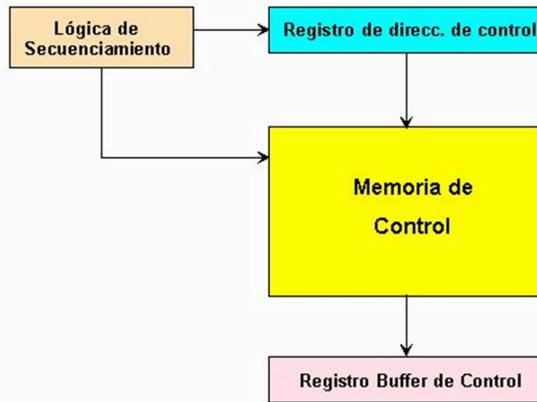
En la figura anterior, se muestra como las palabras de control pueden ser dispuestas en una "memoria de control".

Las microinstrucciones de cada rutina deben ser ejecutadas secuencialmente.

Cada rutina termina con un una bifurcación o un salto, indicando donde se debe buscar la próxima.

La Memoria de control puede tomarse como una concisa descripción de la operación completa de la Unidad de Control, dado que describe la secuencia de micro-operaciones a ser llevada a cabo durante cada sub-ciclo (búsqueda, indirecto, ejecución e interrupción), y además indica el secuenciamiento de estos.

## Microarquitectura De La Unidad De Control Microprogramada



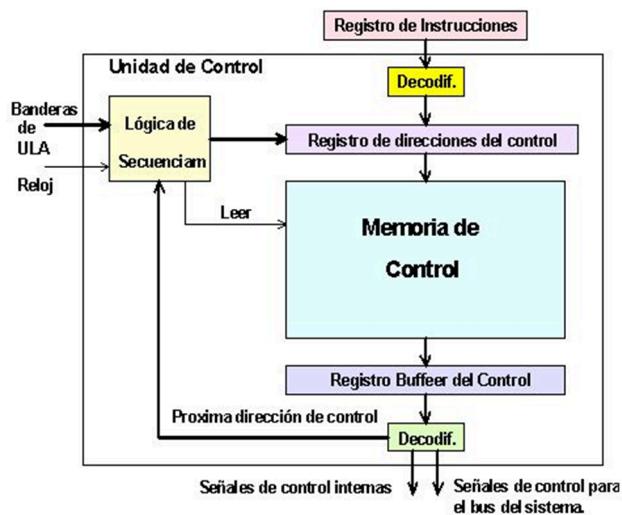
En la figura anterior se tienen los elementos clave de la implementación microprogramada.

El conjunto de microinstrucciones es almacenado en la memoria de control (Control Memory), el registro de direcciones de control (Control Address Register) contiene la dirección de la próxima microinstrucción que debe ser leída.

Cuando es leída una microinstrucción en la memoria de control, es transferida al registro buffer del control(Control Buffer Register).

La lógica de secuenciamento determinará cuál será la próxima microinstrucción a leer de la memoria de control.

## Funcionamiento De La Unidad De Control Microprogramada



1. Para ejecutar una instrucción, la lógica de secuenciamento entrega un comando de lectura a la memoria de control.
2. La palabra cuya dirección es especificada por el registro de direcciones del control, es pasada al registro buffer del control.

3. El contenido del registro buffer de control genera las señales de control y la información de próxima dirección que necesita la unidad lógica de secuenciamiento
4. La unidad lógica de secuenciamiento carga una nueva dirección en el registro de direcciones del control, en base a lo indicado por la información de nueva dirección proveniente del registro buffer del control y de las banderas de la ULA.

Todo esto ocurre durante un pulso de reloj.

#### **Secuenciamiento De Las Microinstrucciones (Opciones Punto 4)**

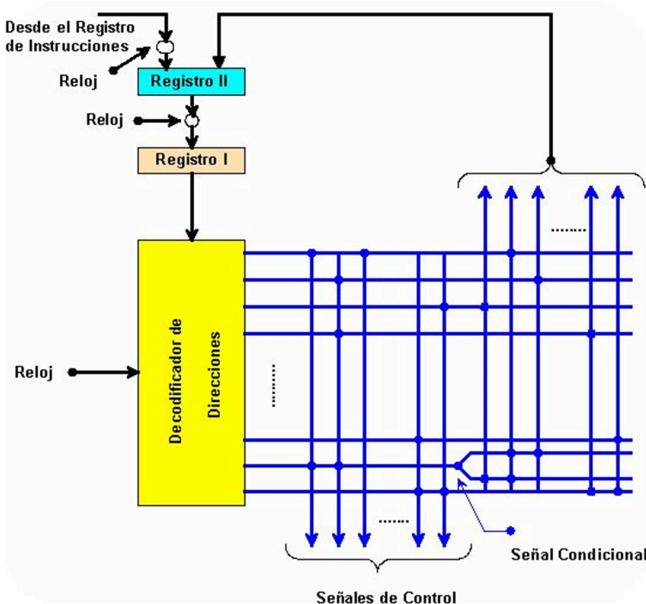
- Obtener la siguiente microinstrucción: Sumar 1 al registro de direcciones del control.
- Saltar a una nueva rutina en base a una microinstrucción de salto: Cargar el campo de dirección del registro buffer de control en el registro de direcciones del control.
- Saltar a una rutina de instrucción de máquina (macroinstrucción): Carga el registro de direcciones del control de acuerdo al código operativo contenido en el IR (el decodificador superior traslada o mapea el código operativo a la dirección de comienzo del microprograma).

#### Funcionamiento

El decodificador inferior es usado para las microinstrucciones verticales, traslada los códigos en señales de control individuales .

La ventaja de las microinstrucciones verticales es que son más compactas (pocos bits) que las microinstrucciones horizontales a expensas de una pequeña cantidad adicional de lógica y tiempo de retardo

#### Control De Wilkes



El corazón del control microprogramado es una matriz de diodos, parcialmente llena. Durante un ciclo de máquina, una fila de la matriz es activada mediante un pulso, lo que genera señales en los puntos donde están conectados los diodos (marcados con puntos en el diagrama).

La primera parte de la fila genera señales de control, que son las enviadas a controlar la UCP (CPU), mientras que la segunda parte genera la dirección de la fila que debe ser pulsada en el próximo ciclo de máquina. En consecuencia, cada línea de la matriz es una microinstrucción, y la matriz en sí misma es la memoria de control.

En el inicio de un ciclo, la dirección de la fila a ser pulsada es contenida en el Registro I. Esta dirección es la entrada del decodificador, el que, cuando es activado por el pulso de reloj, activa esa fila de la matriz.

De acuerdo con lo que indiquen las señales de control, durante el mismo ciclo, se pasa al Registro II ya sea el código operativo contenido en el registro de instrucciones, o la segunda parte de la fila pulsada, que contiene la dirección de la próxima microinstrucción a ser ejecutada.

El contenido del Registro II es pasado al Registro I por un pulso de reloj. En consecuencia, se utilizan pulsos de reloj alternados para activar una fila de la matriz, y para transferir el contenido del Registro II al Registro I.

Esta disposición de dos registros es necesaria por cuanto el decodificador es simplemente un circuito combinacional, y con un solo registro, la salida se vuelve entrada durante el mismo ciclo, provocando una condición inestable.

Este esquema es muy similar a la microprogramación horizontal descrita anteriormente. La principal diferencia es que en la descripción previa, el registro de direcciones del control puede ser incrementado en una unidad para entregar la

nueva dirección. En el esquema de Wilkes, la siguiente dirección está contenida en la misma microinstrucción.

Para permitir la bifurcación, una fila debe contener dos partes de dirección, controladas por una señal de condición, tal como se muestra en la figura.

#### Ventajas De La Microprogramación

Implementar una unidad de control en forma sencilla.

El microprograma es almacenado en una rom, si se cambia la misma se cambia el programa de microcontrol. Esto cambia la actuación de la máquina.

Flexibilidad.

Emulación de otras máquinas.

#### Desventajas De La Microprogramación

La máquina es más lenta que la de la unidad de control cableada.

## FORMATO DE INSTRUCCIONES (B)

### Instrucciones

- Las instrucciones son todas “INSTRUCCIONES DE MÁQUINA”
- Toda máquina tiene su “CONJUNTO DE INSTRUCCIONES”
- El conjunto de instrucciones “ES PROPIO DE CADA MÁQUINA”

### Elementos De Una Instrucción

Por cada instrucción, la unidad de control debe tener:

- **EL CÓDIGO DE OPERACIÓN:** Especifica la operación a ser llevada a cabo en código binario, también conocido como “Código Operativo”.
- **REFERENCIA AL OPERANDO FUENTE:** Indica donde buscar los datos que son las entradas de la operación.
- **REFERENCIA AL OPERANDO RESULTADO:** Donde hay que almacenar el resultado de la operación.
- **REFERENCIA A LA PRÓXIMA INSTRUCCIÓN:** Lo que indica a la UCP donde buscar la próxima instrucción, luego de haber concluido la ejecución de la actual.

El Operando Fuente y Operando Resultado se encuentra en → Memoria principal o virtual, Registros de la CPU y Periféricos de Entrada y Salida

### Representación De Instrucciones

Las instrucciones son representadas por una secuencia de bits divididas en campos específicos. La forma de la instrucción es conocida como “formato” y tiene 3 campos: Código operativo, referencia al primer operando y referencia al segundo operando.

## Formato De Instrucción Simple De Dos Direcciones

CODIGO OPERATIVO	REFERENCIA OPERANDO 1	REFERENCIA OPERANDO 2
---------------------	--------------------------	--------------------------

### Tipos de instrucciones

- **De procesamiento de datos (Instrucciones lógicas y aritméticas)**

Lógicas → AND, OR, NOT

Aritmética → La suma y resta con o sin acarreo, incremento y decremento de un registro, comparaciones, ajuste decimal, complemento y negación.

- **De almacenamiento de datos (Instrucciones con referencia a memoria)**

→ Movimiento de datos entre la memoria y los registros

- **De movimiento de datos (Instrucciones de Entrada y Salida)** → Transferir datos hacia la memoria o al usuario

- **De control (Instrucciones de Prueba y Bifurcación)**

Prueba → Comprobar el contenido de una palabra

Bifurcación → Transferir el control del programa a otro punto

### Abreviaturas Nemotécnicas

Para interpretar las expresiones en código binario se ha utilizado lo siguiente

**ADD** = suma  
**SUB** = sustracción  
**MPY** = multiplicación  
**DIV** = división  
**LOAD** = carga de datos desde la memoria  
**STOR** = almacenamiento de datos en memoria

### Cantidad de direcciones

Para realizar una operación aritmética necesitamos

1. DIRECCIÓN PRIMER OPERANDO
2. DIRECCIÓN SEGUNDO OPERANDO
3. DIRECCIÓN PARA EL RESULTADO
4. DIRECCIÓN PRÓXIMA INSTRUCCIÓN

La de cuatro es la menos utilizada, porque la referencia a la próxima instrucción, normalmente se encuentra en el contador de programa (PC)

## EJEMPLO INSTRUCCIÓN CON UNA DIRECCIÓN

$$Y = (A-B)/(DE+C)$$

LOAD D	AC <-- D
MPY E	AC <-- ACxE
ADD C	AC <-- AC+C
STOR Y	Y <-- AC
LOAD A	AC <-- A
SUB B	AC <-- AC-B
DIV Y	AC <-- AC÷Y
STOR Y	Y <-- AC

El registro acumulador (AC) es una dirección implícita.

### EJEMPLO INSTRUCCIÓN CON DOS DIRECCIONES

$$Y = (A-B)/(DE+C)$$

MOVE Y,A	Y <-- A
SUB Y,B	Y <-- Y-B
MOVE T,D	T <-- D
MPY T,E	T <-- TxE
ADD T,C	T <-- T+C
DIV Y,T	Y <-- Y÷T

### EJEMPLO INSTRUCCIÓN CON TRES DIRECCIONES

$$Y = (A-B)/(DE+C)$$

SUB Y,A,B	Y <-- A-B
MPY T,D,E	T <-- Dx E
ADD T,T,C	T <-- T+C
DIV Y,Y,T	Y <-- Y÷T

### Instrucciones De Cero Direcciones

Es posible tener máquinas con instrucciones sin ninguna dirección , las que son aplicables a la conformación de memoria especial, denominada Memoria de Pila (Stack Memory), en la cual la referencia es siempre a la primera casilla de la pila, que es de tipo LIFO (Last In First Out -Último que entra primero que sale).

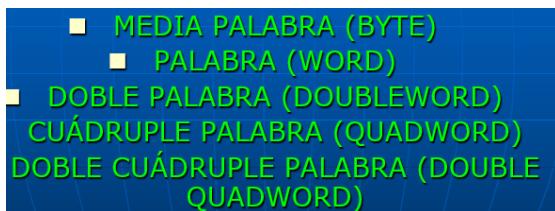
### Tipos de Operandos

- Direcciones → Escrita en hexadecimal, operable en la unidad lógica y aritmética
- Números → Coma fija o enteros, coma flotante y decimales (BCD)

- Caracteres → ASCII (American Standard for Information Interchange) de 7 u 8 BITS Y EBCDIC (Extended Binary Coded Decimal Interchange Code) de 8 BITS.
- Datos Lógicos → Palabra como serie de bits

## Tipos de datos en intel x86

Tratan con datos de 8,16,32,64 o 128 bits de longitud



La dirección puede comenzar en cualquier byte, que es la unidad mínima de información.

Cuando los datos son accedidos a través de un bus de 32 bits, la transferencia tiene lugar en unidades de doble palabra y la dirección comienza en un valor divisible por cuatro.

Utiliza el estilo Little-endian para almacenar valores, es decir, el byte menos significativo es almacenado en la dirección más baja.

Los tipos de datos anteriores son denominados generales, Intel x86 soporta un impresionante arreglo de tipos de datos específicos (once) reconocidos y operados por instrucciones particulares.

Los enteros con signo son representados en complemento a dos y pueden ser de 16, 32 o 64 bits de longitud.

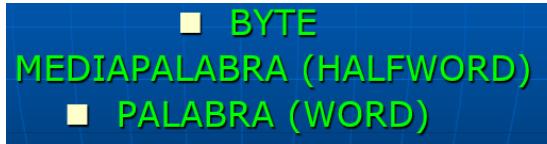
Los tipo punto flotante se refiere a un conjunto de tipo de datos que son usados por la unidad de punto flotante(fpu-floating-point unit) para operar con ellos al ejecutar las instrucciones de punto flotante. Incluye tres representaciones de punto flotante de acuerdo con el estándar IEEE 754.

Los tipos de datos SIMD(single-instruction-multiple-data) empaquetados fueron introducidos en la arquitectura x86 como parte de las extensiones del conjunto de instrucciones para mejorar el desempeño de las aplicaciones multimediales. Estas incluyen a las extensiones MMX(multimedia extensions) y SSE (streaming SIMD extensions).

El concepto básico es que múltiples operandos son empaquetados dentro de una sola ítem de referencia a memoria y que estos múltiples operandos son operados todos en paralelo.

## Tipos de datos ARM

Pueden tratar con datos de 8, 16 o 32 bits de longitud



Todos los tipos de datos soportan representación de enteros sin signo y enteros con signo en complemento a dos.

La mayoría de las implementaciones de procesadores arm no proporcionan hardware para punto flotante, para reducir consumo y superficie del circuito.

Si es necesaria la aritmética en punto flotante debe implementarse por software.

También soporta un coprocesador en punto flotante opcional que utiliza las representaciones definidas en el estándar IEEE 754

En el registro de control de programa del sistema tiene el bit de estado “E” que puede ser colocado en cero o en uno con la instrucción “SETEND” y seleccionar cuál sistema endian de carga y almacenamiento de valores de registro se utilizará (little-endian o big-endian)

La dirección de cada uno de los bytes está fija en la memoria, sin embargo la ubicación en los registros es diferente:

- Si E-BIT=0 responde a little-endian
- Si E-BIT =1 responde a big-endian

## Tipos De Operaciones

- Transferencia de datos (MOVE, SET, PUSH, POP)
- De transferencia del control (JUMP, SKIP)
- Aritméticas (ADD, SUB, INCREMENT)
- Lógicas (AND, OR, NOT)
- De conversión (TRANSLATE, CONVERT)
- De E/S (INPUT, OUTPUT)

## TRANSFERENCIA DE DATOS

<b>MOVE</b>	Transfiere una palabra o un bloque desde la fuente al destino.
<b>STORE</b>	Transfiere una palabra desde la UCP (CPU) a la memoria.
<b>LOAD (Fetch)</b>	Transfiere una palabra de la memoria a la UCP (CPU).
<b>EXCHANGE</b>	Intercambia los contenidos de la fuente y el destino.
<b>CLEAR (Reset)</b>	Transfiere al destino una palabra conteniendo todos ceros.
<b>SET</b>	Transfiere al destino una palabra conteniendo todos unos.
<b>PUSH</b>	Transfiere una palabra de la fuente al acceso a la pila.
<b>POP</b>	Transfiere una palabra del acceso a la pila al destino.

33

## ARITMÉTICAS

<b>ADD</b>	Computa la suma de dos operandos
<b>SUBTRACT</b>	Computa la resta de dos operandos
<b>MULTIPLY</b>	Computa el producto de dos operandos
<b>DIVIDE</b>	Computa el cociente de dos operandos
<b>ABSOLUTE</b>	Reemplaza al operando por su valor absoluto
<b>NEGATE</b>	Cambia el signo del operando
<b>INCREMENT</b>	Suma 1 al operando
<b>DECREMENT</b>	Resta 1 al operando

## LÓGICAS

<b>AND</b>	Conforma la operación indicada entre bits.
<b>OR</b>	Conforma la operación indicada entre bits.
<b>NOT (Complement)</b>	Conforma la operación indicada entre bits.
<b>EXCLUSIVE-OR</b>	Conforma la operación indicada entre bits.
<b>TEST</b>	Prueba la condición especificada.
<b>COMPARE</b>	Hace la comparación lógica o aritmética de dos o más operandos.
<b>SET CONTROL VARIABLES</b>	Conjunto de instrucciones que ubica controles para protección, manejo de interrupciones, timers, etc.
<b>SHIFT</b>	Desplaza a derecha o izquierda al operando, introduciendo constantes en los extremos.
<b>ROTATE</b>	Desplaza a derecha o izquierda al operando, en forma circular.

35

## TRANSFERENCIA DE CONTROL

<b>JUMP (Branch)</b>	Transferencia incondicional, cargando al contador de programa con un nuevo contenido.
<b>JUMP CONDITIONAL</b>	Prueba la condición indicada y carga o <u>nó</u> al contador de programa con un nuevo contenido.
<b>JUMP TO SUBROUTINE</b>	Guarda la información de control del programa en ejecución y salta a la nueva dirección especificada.
<b>RETURN</b>	Reemplaza el contenido de los registros con datos conocidos.
<b>EXECUTE</b>	Busca los operandos indicados y ejecuta la instrucción sin alterar al contador de programa.
<b>SKIP</b>	Incrementa al contador de programa para pasar a la próxima instrucción.
<b>SKIP CONDITIONAL</b>	Prueba la condición especificada y cambia o <u>nó</u> el contenido del contador de programa.
<b>HALT</b>	Detiene la ejecución del programa.
<b>WAIT (Hold)</b>	Suspende la ejecución del programa hasta que se cumpla una condición dada.
<b>NO OPERATION</b>	No se realiza ninguna operación, pero el programa <u>sigue</u> ejecutando.

## ENTRADA/SALIDA

<b>INPUT (Read)</b>	Transfiere datos del dispositivo indicado o puerto de E/S al destino.
<b>OUTPUT (Write)</b>	Transfiere datos de la fuente especificada al puerto de E/S o dispositivo indicado.
<b>START I/O</b>	Transfiere instrucciones al procesador de E/S para iniciar la operación de E/S
<b>TEST I/O</b>	Transfiere información de estado del sistema de E/S al destino indicado.

## CONVERSIÓN

<b>TRANSLATE</b>	Traduce los valores de una sección de la memoria, en base a una tabla de correspondencias.
<b>CONVERT</b>	Convierte el contenido de una palabra de una forma a otra.

## Transferencia De Datos +++++++

En la instrucción debe especificarse :

- Localización de los operandos fuente y destino: cada locación puede estar en la memoria, en un registro, o en el acceso a una pila.
- Longitud de los datos a transferir.

- Modo de direccionamiento para cada operando.

Datos En Memoria +++++++

1. Calcular la dirección de memoria.
2. Si la dirección se refiere en memoria virtual, se debe trasladar el dato a la zona de memoria real.
3. Determinar si el ítem referenciado está en la caché.
4. Si no está, buscarlo en el módulo de memoria correspondiente.

Transferencia Del Control

Sirve para modificar la secuencia del programa.

#### **Motivos para modificar secuencia:**

- Cuando es necesario ejecutar una instrucción más de una vez, para evitarlo, es posible escribir en forma separada el conjunto de instrucciones que será repetido, y acceder a él toda vez que sea necesario.
- Todo programa implica el tomar algún tipo de decisión. En estos casos la ejecución del programa puede realizarse por diversos caminos según cual sea esa decisión.
- Para confeccionar correctamente un programa relativamente largo, es conveniente poder separarlo en tareas y programar y correr cada una de ellas por separado.

#### **Operaciones:**

- Bifurcación (BRANCH)

- BRP X: Saltó a la dirección X si el resultado es positivo.
- BRN X: Saltó a la dirección X si el resultado es negativo.
- BRZ X: Saltó a la dirección X si el resultado es cero.
- BRO X: Saltó a la dirección X si hay un rebose (Overflow).
- BRE R1,R2,X: Salta a X si  $<R1> = <R2>$ .

Las bifurcaciones o saltos pueden ser hacia adelante o hacia atrás.

- Salto (SKIP)

Pasar por alto una instrucción, lo cual significa que la dirección de la próxima instrucción es el contenido del contador de programa, más una unidad.

Como no hace falta el campo dirección para el salto, se pueden incorporar otras condiciones.

- Llamado a subrutina (CALL SUB)

Programa autosuficiente incorporable a un programa mayor. El programa mayor invoca (llama) a la subrutina y le transfiere el control, y se lo devuelve al programa mayor cuando se concluya su ejecución.

Implica dos instrucciones básicas, el llamado a la subrutina y el retorno al programa principal.

Una instrucción de llamado o de retorno, es una instrucción de bifurcación o de salto que transfiere el control a otra dirección.

Características:

- Economía: ahorro de espacio en memoria y al menor trabajo de programación
- Modularidad: un gran programa puede ser subdividido en pequeñas unidades, lo cual también facilita la tarea de programación

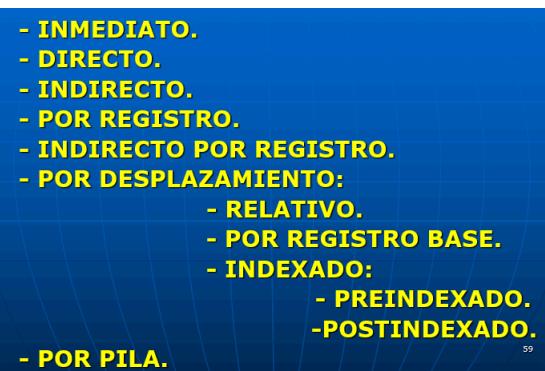
CALL SUB # (Llamado a subrutina #)

RETURN (Retorno al programa que invocó a la subrutina)

### Tipos de Direccionamiento

Los campos de direccionamiento son de longitud limitada, por lo tanto hay que implementar algunas técnicas que relacionan la capacidad disponible con la longitud de los campos

#### Diferentes modos



#### Convenciones

<b>M</b>	<b>= Parte mando de la Instrucción</b>
<b>A</b>	<b>= Contenido del campo de dirección</b>
<b>DE</b>	<b>= Dirección Efectiva</b>
<b>(X)</b>	<b>= Contenido de la locación X.</b>

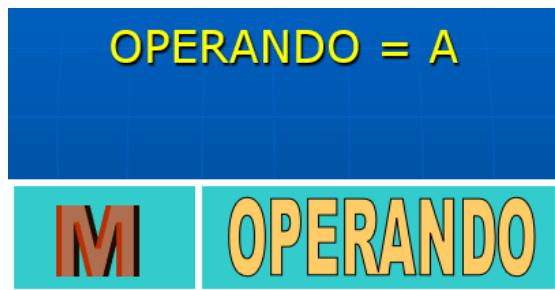
### Indicar El Modo De Direccionamiento A La Unidad De Control

- Agregar al mando un campo de modo de direccionamiento
- Diferentes códigos operativos, distintos modos de direccionamiento

### Instrucción Típica



Direccionamiento inmediato



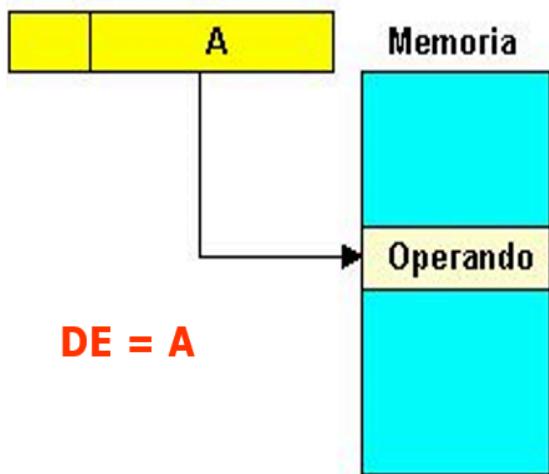
Es utilizado principalmente para definir constantes, o determinar valores iniciales para las variables.

Ventaja → Se acelera el cálculo donde esa constante o valor inicial son utilizados reiterativamente.

Desventaja → El tamaño del número es restringido

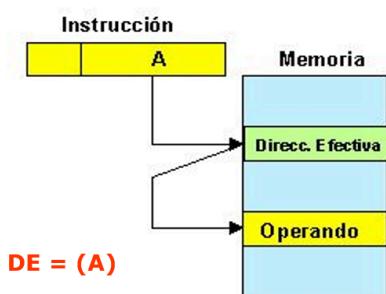
Direccionamiento Directo

### Instrucción



Su desventaja, es que solo se puede direccionar memorias de poca capacidad.

### Direccionamiento Indirecto



Ventaja → Con una longitud de n bits se puede direccionar  $2^n$  posiciones de memoria

Desventaja → Requerir 2 accesos a memoria para obtener el operando

### Direccionamiento Por Registro

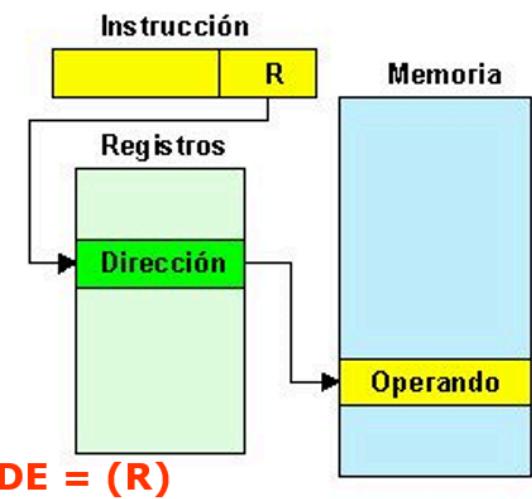


Parecido al directo solo que hace referencia a un registro en vez de a una posición de memoria

Ventaja → No hace referencia a memoria para buscar un operando y solo se necesita un pequeño campo de dirección

Desventaja → El espacio de direcciones es MUY limitado

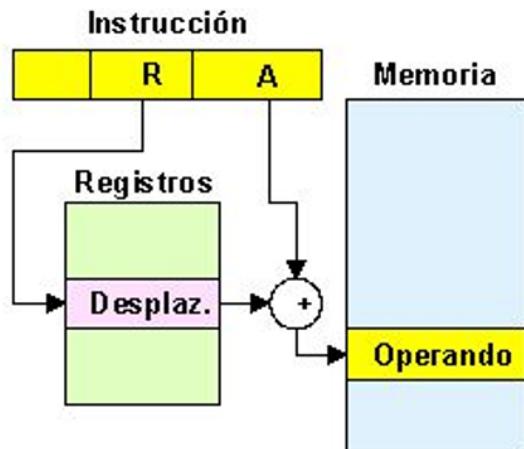
### Direccionamiento Indirecto Por Registro



Es completamente análogo al direccionamiento indirecto, solo que en vez de acceder a memoria para hallar la dirección, la misma se encuentra en un registro de la UCP (CPU).

Las ventajas y limitaciones son análogas a las del direccionamiento indirecto.

#### Direccionamiento Por Desplazamiento

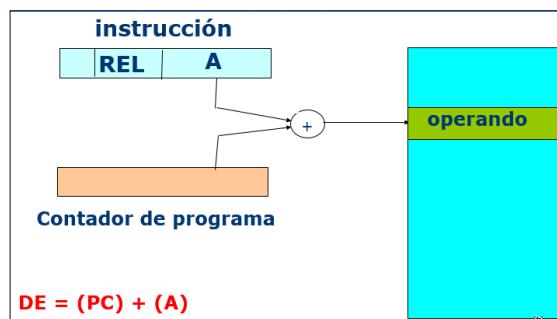


Combina las ventajas del direccionamiento directo con el direccionamiento indirecto por registros.

Requiere que haya dos campos de direccionamiento, uno referente al registro a utilizar y otro al desplazamiento a agregar al contenido de ese registro.

Los tres usos más comunes de este tipo de direccionamiento:

## Direccionamiento Relativo

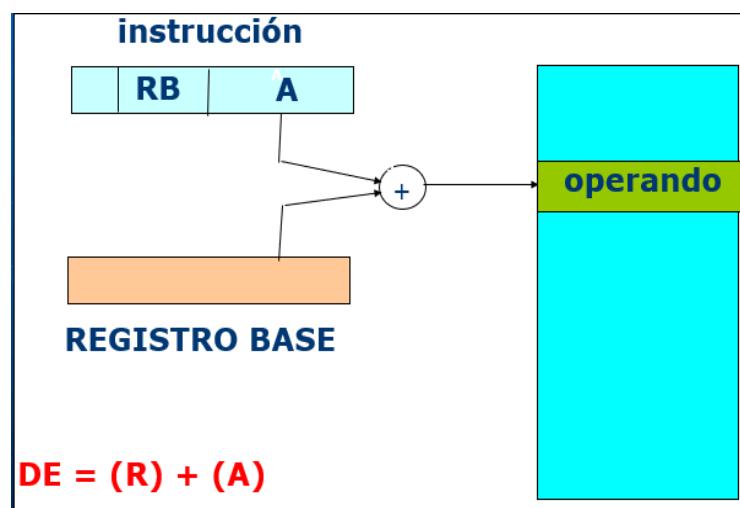


La referencia implícita es el contador de programa (PC), a cuyo contenido es sumado el contenido del campo de dirección para producir la DIRECCIÓN EFECTIVA.

Normalmente el contenido del campo de dirección es tratado como un número en complemento a dos, por lo que el desplazamiento puede ser hacia adelante o hacia atrás.

La dirección efectiva es función de la dirección de la instrucción, dada por el PC, por tanto esto permite una efectiva economía de bits en el direccionamiento.

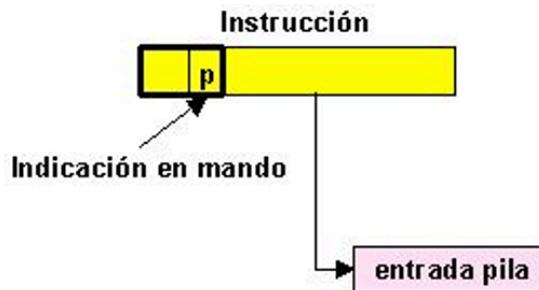
- **Direccionamiento Por Registro Base**



El registro referenciado contiene la dirección de memoria, y el campo de dirección de la instrucción, contiene el desplazamiento.

Reúne las ventajas del modo anterior y además permite direccionar todo un programa a partir de una cierta dirección de base, de aquí el nombre de registro base, lo cual permite la carga simultánea de varios programas, cada uno de los cuales es referenciado mediante un registro.

- **Direccionamiento A Pila**



La pila es una matriz lineal de posiciones de memoria.

Tomar una cierta cantidad de posiciones de memoria y un puntero de pila, el que apunta siempre a la última casilla ocupada.

- Cada vez que se introduce un dato el puntero cambia a la casilla recién ocupada, y cada vez que se extrae uno, el puntero vuelve atrás.
- El valor del puntero de la pila se mantiene en un registro, por lo que el direccionamiento de la pila es en realidad un direccionamiento relativo al contenido del mismo.

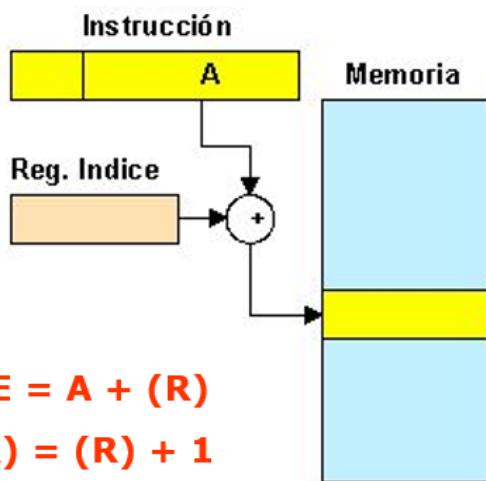
No es necesario hacer referencia a memoria en el campo de direcciones, sino que siempre se opera en el registro de puntero de pila (SPR).

- **Direccionamiento Indexado**

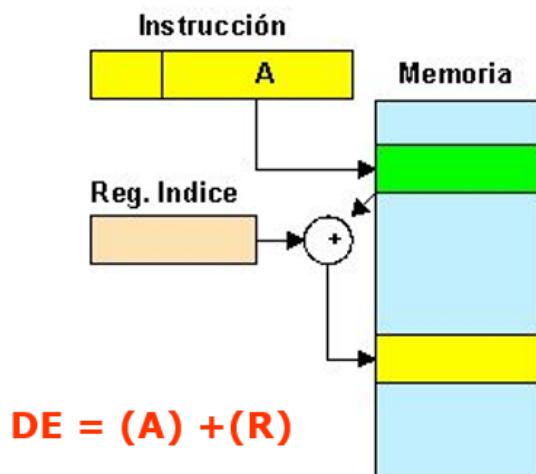
El campo de dirección hace referencia a una dirección de memoria, y el registro referenciado contiene un desplazamiento positivo a partir de ésta dirección.

Registro de índice incrementa en una unidad cada vez que es referenciado.

**Preindexado** → Indexación es efectuada antes de calcular la dirección

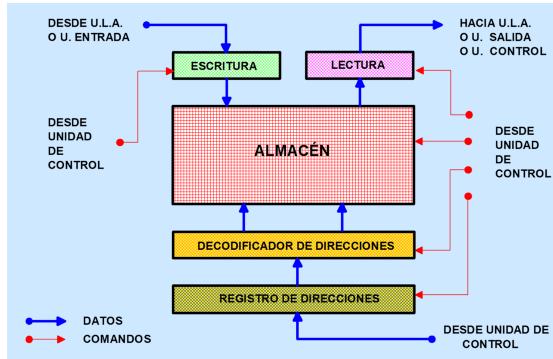


**PostIndexado** → Emplear conjuntamente el direccionamiento indirecto y el indexado



## MEMORIA

### UNIDAD DE MEMORIA (A)



**PUNTO DE MEMORIA:** Se denomina así a la celda capaz de almacenar un bit, o sea, la mínima unidad de almacenamiento que puede existir. También se la conoce como “celda de bit”.

**CELDA DE MEMORIA:** Grupo de celdas de bit capaces de almacenar una palabra, siendo la palabra una tira de bits de una longitud prefijada, que puede manejar la máquina como entidad única. En general corresponde a la longitud, medida en bits, de los registros de la unidad aritmética y lógica.

**DIRECCIONAMIENTO:** Es el procedimiento por el cual es posible ubicar las posiciones de memoria para efectuar su lectura o su escritura.

**LECTURA/ESCRITURA:** Lectura → Transferir el contenido de una celda de memoria a un registro cualquiera de la máquina, generalmente el registro de salida de la memoria. Escritura → Almacenar información en una celda.

En ambos casos es necesario primero direccionar la celda y luego de ubicada la misma, producir la transferencia de la información en el sentido deseado.

**ACCESO A MEMORIA:** Proceso de lectura/escritura en memoria.

**TIEMPO DE ACCESO:** Para efectuar una lectura o una escritura en memoria, se demora un cierto tiempo, pudiendo no ser ambos iguales. Se acostumbra a definir el tiempo de acceso, como el promedio de ambos.

$$t_a = \frac{1}{2} (t_l + t_e)$$

Donde:  $t_a$  = tiempo de acceso.

$t_l$  = tiempo de lectura.

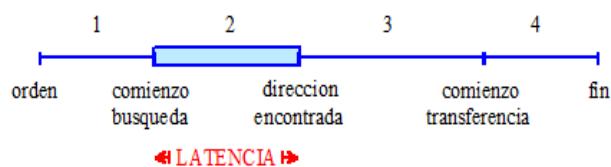
$t_e$  = tiempo de escritura.

**CICLO DE MEMORIA:** Cada vez que se produce un acceso a memoria. Parte de la orden de efectuar el proceso, hasta que él mismo ha sido completado.

A partir del momento que se aplica, desde la unidad de control, la señal de efectuar una búsqueda, hay un breve retardo. Solo después de éste breve lapso, comienza efectivamente la búsqueda del lugar donde está guardada la información, o sea de la posición de memoria que debe ser accedida.

Este tiempo, que es llamado “latencia”, depende de varios factores, el principal es el tipo de memoria.

- Si es de acceso aleatorio, estrictamente la latencia no existe, pero en la actualidad se utiliza en las memorias RAM como sinónimo de ciclo de memoria.
- Si es de cinta magnética, la latencia media puede llegar a valer tanto como la mitad del tiempo total de lectura de toda la cinta.
- Si es de un disco, se trata de disponer la cabeza en la pista indicada y luego esperar como promedio media vuelta del disco.



## Dimensiones De Las Memorias

**CAPACIDAD:** Es la cantidad de información que puede almacenar, y se mide en bits, bytes o palabras, y por supuesto mediante algún múltiplo kilo, mega o giga. Ejemplo: 64 Megabytes (64MB).

**CAUDAL:** Es la cantidad de información que puede transferir por unidad de tiempo, medida en bits o bytes. Ejemplo: 134 Gigabytes/segundo (134 GB/s).

**DENSIDAD:** Puede ser lineal, superficial o volumétrica, y permite determinar la cantidad de información, en bits o bytes que puede ser almacenada por unidad de longitud, de superficie, o de volumen. Ejemplos: 82 bits/cm, 300 kilobytes/cm<sup>2</sup>, 250 Megabytes/cm<sup>3</sup>

## Rendimiento De Una Memoria +++++++

El tiempo medio de acceso a memoria es una medida del rendimiento de la misma. Tiene en cuenta los posibles errores que se producen en los procedimientos.

- Tiempo de acierto: Es el tiempo necesario para un acceso correcto.

- Demora por fallas: Es el tiempo necesario para buscar la información en otro bloque de memoria o en el almacén externo, y que se debe agregar cuando el dato no se encuentra en la memoria principal.
- Frecuencia de fallas: Es la cantidad de veces por unidad de tiempo, que ocurre una falla de acceso a la memoria.

Tiempo medio de acceso = Tiempo de acierto + (Frecuencia de falla x Demora por falla)

## Clasificación De Las Memorias

- **Por su aplicación:**

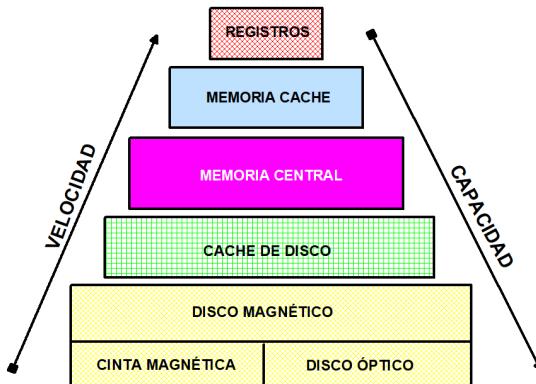
**MEMORIA CENTRAL**: Es utilizada exclusivamente para el almacenamiento de programas y datos que se están ejecutando en ese momento. También se denomina Memoria Interna o Memoria Principal.

**MEMORIA PERIFÉRICA**: Almacena también programas y datos, que para su utilización, la unidad de control deberá proceder a transferirlos a la memoria central. También se denomina Memoria Externa o Memoria de Masa.

- **Por su jerarquía** (Grado de proximidad de la misma con la unidad lógica aritmética):

Tiende al equilibrado de los anchos de banda, denominando así a la capacidad de transferencia de información, resultando en un computador que el ancho de banda de la unidad lógica y aritmética es el mayor de todos, debido al caudal de información que puede manejar en la unidad de tiempo, siguiendo en orden de méritos la memoria central y luego las periféricas.

Para tratar que todo funcione con caudales parecidos, se usan diversos tipos de memorias, ubicados en cascada entre los circuitos de cálculo y el almacén externo.



La memoria CACHÉ, es una memoria intermedia entre la memoria central y la unidad de cálculo, utilizada para almacenar la información de próxima utilización. Normalmente la transferencia de la caché a las unidades de control y de cálculo, es por palabras, mientras que entre la memoria central y la caché lo es por bloques o conjuntos de palabras.

A veces también se emplea una CACHÉ para el disco, la cual se ubica entre la memoria central y los discos magnéticos, ópticos y cinta magnética, cuya finalidad es la de permitir efectuar transferencias en modo multiprocesamiento, o sea solapando con otras funciones de la UCP (CPU).

Los registros utilizados por la UCP (CPU), a veces se agrupan en la denominada memoria borrador, o "scratch pad", denominada así por cuanto soporta los datos intermedios de las operaciones. Por otra parte, cuando la memoria externa o periférica es de gran capacidad y por lo tanto de transferencia lenta, es conocida como memoria fichero.

- **Por su tipo de acceso:**

**SECUENCIAL:** Cuando la lectura se realiza recorriendo todas las posiciones en forma secuencial, tal como sucede en las cintas magnéticas. CINTAS, CD-ROM

**CÍCLICO:** Cuando la información es accesible en intervalos regulares de tiempo. DISCOS

**COORDINADO:** El acceso es independiente de cada una de las posiciones, un sistema de direccionamiento por coordenadas. ROM, RAM, DRAM

- **Por su funcionamiento:**

**ESTÁTICAS:** Cuando no es necesario ningún tipo de movimiento, ni de cargas eléctricas, ni de campos magnéticos. SRAM, ROM

**DINÁMICAS:** Cuando es necesario el continuo refresco para reponer cargas perdidas.DRAM

- **Por la permanencia de la información:**

**VOLÁTILES:** Cuando la información almacenada se pierde al cabo de cierto tiempo. DRAM

**NO VOLÁTILES:** Cuando la información perdura indefinidamente, o por lo menos un tiempo muy largo. DISCOS

- **Por su forma de lectura:**

**LECTURA DESTRUCTIVA:** Cuando es necesario modificar la información almacenada para leerla. DRAM

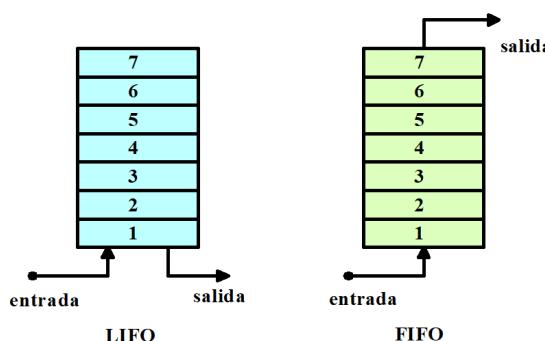
**LECTURA NO DESTRUCTIVA:** Cuando no es necesario alterar su contenido para leerlo. RAM, ROM

## Memorias Especializadas

- Memorias de pilas:

FIFO (first in first out)

LIFO (last in first out)



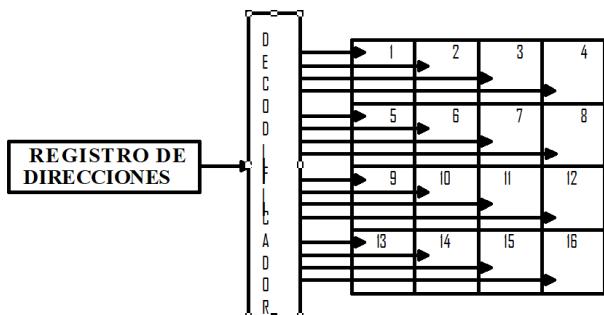
- Memorias de lectura exclusiva: ROM (READ ONLY MEMORY)
- Memorias asociativas: Direccionadas por contenido

## Direccionamiento

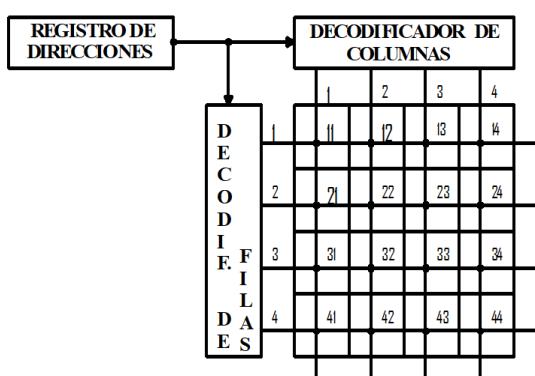
Proceso por el cual se accede a una celda (a veces mal llamada dirección).

Depende de la tecnología de fabricación de la memoria.

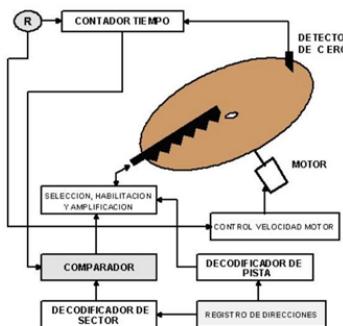
### Direccionamiento por número de celda



### DIRECCIONAMIENTO POR FILAS Y COLUMNAS (COORDENADO)



### Direccionamiento por pista y tiempo de un disco rígido con cabezas múltiples



## Memorias De Semiconductor

Por su funcionamiento, existen dos tipos de almacenes de semiconductor, los de lectura exclusiva o ROM (Read Only Memory) y los de lectura y escritura, genéricamente denominados RAM (Random Access Memory o Memoria de Acceso Aleatorio).

- ROM
  - Memorias muertas, por cuanto no son alterables ni tampoco Volátiles.
  - Se usan para almacenar programas específicos que se deben realizar siempre por una computadoras o instrumentos.
  - Por ejemplo los programas de arranque, o datos de puesta a punto, condiciones iniciales o de prueba.
  - Se utiliza para la BIOS.
- PROM (ROM PROGRAMABLE)
  - Pueden ser programadas por el usuario.
  - Se graban una sola vez.
  - Se utilizan para el desarrollo de prototipos.
  - Luego se puede pasar a ROM (solo el fabricante)
- EPROM (ROM BORRABLE Y REPROG. O REPROM)
  - Significa erasable prom, o sea prom borrable.
  - Tienen una ventana superior para borrarlas por luz ultravioleta.
  - También son conocidas como reprom (reprogrammable rom)
  - Se usan para el desarrollo de prototipos y modelos.
- EAROM (ROM ELECTRICAMENTE ALTERABLE, O EEROM)
  - Electrically alterable rom
  - Se puede variar el contenido celda por celda y aún bit por bit.
  - Se usan como memorias permanentes en odómetros (cuentakilómetros), cajas de supermercado, y otras aplicaciones donde el contenido de la memoria solo varía lentamente.
- SRAM (RAM ESTÁTICA)
  - Generalmente consisten en un multivibrador set-reset.
  - Las de mejor comportamiento son las de tecnología nmos.
  - Las de mayor velocidad de acceso son las pmos.

- Las de menor consumo las cmos.
  - Se las utiliza como memoria central o como memoria caché de la computadora.
- DRAM (RAM DINÁMICA)
    - Son memorias ram cuyo contenido debe ser actualizado permanentemente.
    - Estructura simple y de alta densidad.
    - Se utilizan en computadoras personales por su bajo costo y por su rendimiento.
    - El refresco consiste en una lectura ordenada y su correspondiente reescritura.
    - Se utilizan como memoria central de la computadora.

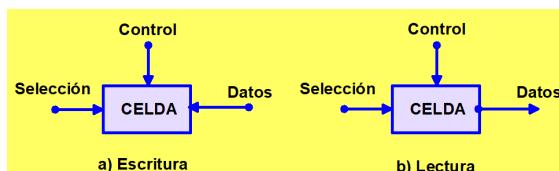
## Organización De Las Memorias De Semiconductor

Los multivibradores o las celdas dram se disponen según matrices.

Se agregan en el chip los circuitos de selección, de lectura y de escritura.

En el caso de las dram se agregan los circuitos de refresco.

## Operación De Las Celdas De Memoria



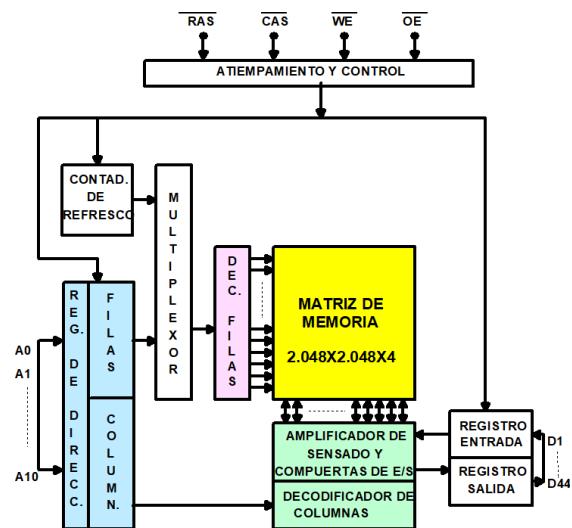
La línea de selección procede del decodificador de direcciones y es la encargada de habilitar la célula para ejecutar la operación indicada por la línea de control. La línea de datos es la que llevará o traerá la información binaria.

## Memoria Dram De 16 MBITS

Las celdas se disponen en cuatro módulos de 4 megabits cada uno.

Cada módulo posee 2.048 x 2.048 celdas.

El sistema de acceso permite leer palabras de cuatro bits, uno por cada módulo.



### Señales De Control Aplicadas Al Chip

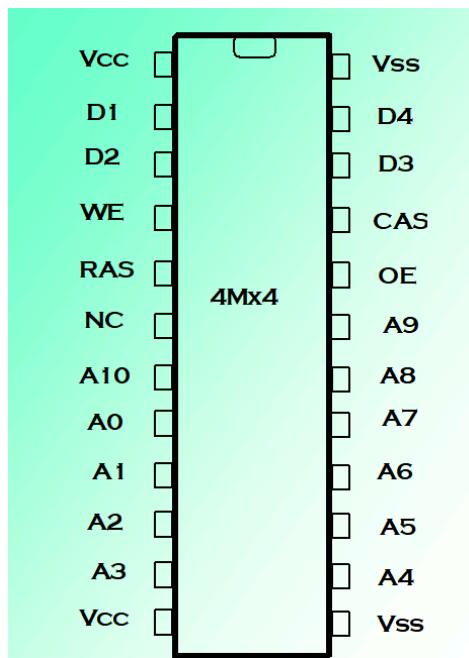
- RAS: row address select
- CAS: column address select
- WE: write enable
- OE: output enable

### Otros Terminales Del Chip

- Once líneas de direcciones
- Cuatro líneas de datos
- Cuatro líneas de control.
- Se debe agregar alimentación y tierra

En total 24 terminales, de los cuales uno no se usa.

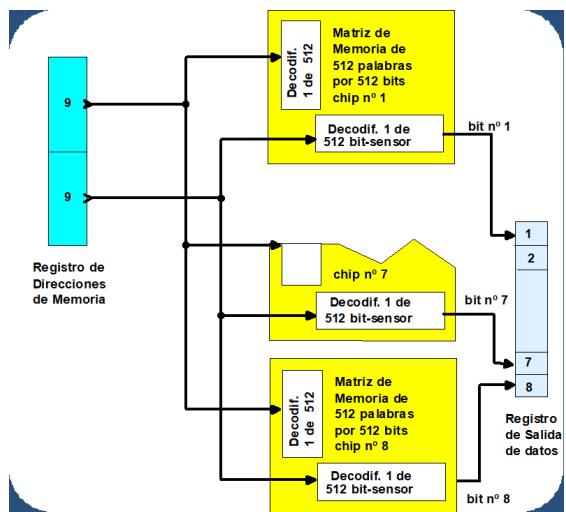
## Disposición De Terminales Del Chip



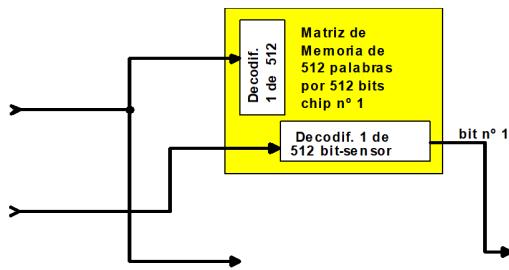
## Lógica De Chips

- Cada chip integrado tiene una cierta capacidad.
- Para aumentarla se combinan diversos chips.
- Para ello tienen todos una entrada denominada chip select. (selección del chip)

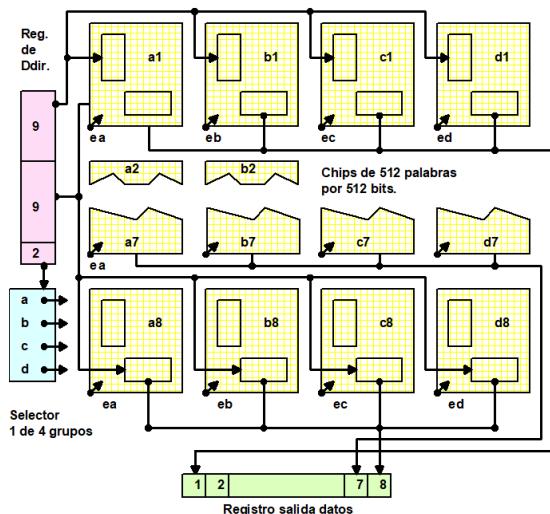
## Utilización De Varios Chips



## Disposición De Cada Chip



## Organización Por Módulos Para Un Mbyte



- Son chips donde es direccionable cada bit ( $512 \text{ palabras} \times 512 \text{ bits} = 262.144 \text{ bits}$ ,  $262.144 / 1.024 = 256 \text{ kbits}$ )
- Cada módulo es de 256k palabras de 8 bits (256 kBytes)
- Cada módulo precisa 18 bits de dirección ( $2^{18} = 262.144$  direcciones,  $262.144 / 1.024 = 256 \text{ k}$ )
- Con 8 chips se llega a leer 256k palabras de ocho bits
- Con cuatro conjuntos iguales se llega a 1 m palabras de 8 bits (1MByte = 8 Mbits).
- Se usan cuatro columnas de chips, cada columna tiene una matriz de 256 k palabras de 8 bits
- Se usan 20 líneas de dirección ( $2^{20} = 1.048.576$  direcciones,  $1.048.576 / 1.024 = 1.024 \text{ k}$ ,  $1.024 \text{ k} / 1.024 = 1 \text{ M}$ )
- Los 18 bits menos significativos se alimentan a los 32 módulos (8 chips/grupo x 4 grupos = 32 chips)
- Los dos bits de mayor orden son la entrada de una lógica de selección de grupo, que envía el "chip select" a una de las cuatro columnas de módulos

## Organización De Las Memorias Para Computadores Personales

En una pc hay tres tipos de memorias:

- La memoria del bios, que es ROM.
- La central, que es tipo DRAM.
- La memoria caché que generalmente es una SRAM CMOS.

Conformación de la memoria central:

Antes → Chips soldados a la motherboard.

Ahora → Pequeñas placas de circuito impreso que se colocan en ranuras especiales.

Disposición:

Existen dos formas de conectar los circuitos en las placas:

- SIMM (single in line memory module)
  - Se fabrican en módulos con 30 y con 72 terminales.
  - 72 terminales permiten direcciones de 32 bits por cada placa.
  - Se necesitan dos placas de 30 terminales para alcanzar la misma capacidad
- DIMM (dual in line memory module)
  - La capacidad de direccionamiento se lleva a 64 bits. para ello se usa impresión de doble capa
  - En total hay 162 contactos, 168 contactos (sdr-sdram), 184 contactos (ddr-sdram), 240 contactos (ddr2 y ddr3-sdram) y 288 contactos (ddr4-sdram) por placa.

SODIMM(small outline dimm)

Son placas DIMM de bajo perfil para uso en portátiles.

DRAM SINCRÓNICA (SDRAM = Synchronous DRAM)

DRAM → Su carga y su lectura pueden hacerse solo a la velocidad del refresco.

SDRAM → Lectura se hace sincrónicamente con el procesador sin tener en cuenta la velocidad de refresco y sin imponer tiempos de espera.

DDR SDRAM

- DDR (DOUBLE DATA RATE)
- Transfiere información dos veces en cada ciclo de reloj, una vez en el flanco de subida y otra vez en el flanco de bajada

- Frecuencia de memoria principal hasta 400 MHz
- Tensión de alimentación de 2.5V

## DDR2 SDRAM

- Frecuencia de memoria principal hasta 600 MHz
- Tensión de alimentación a 1.8V
- DDR3: Frecuencia de memoria principal hasta 1100 MHz y tensión de alimentación 1.5V
- DDR4: Frecuencia de memoria principal hasta 2666 MHz y tensión de alimentación 1.2V

## COMPARACIÓN DDR1 DDR2 DDR3

	DDR1	DDR2	DDR3
Prefetch buffer (bits)	2	4	8
Voltage level (V)	2.5	1.8	1.5
Front side bus data rates (Mbps)	200, 266, 333, 400	400, 533, 677, 800	800, 1066, 1330, 1600

DDR1 usa 2 bits en el buffer de prebúsqueda de manera que puede operar al doble de velocidad de datos que una SDR (Single Data Rate) SDRAM usando la misma frecuencia de chip.

DDR2 dobla a 4 bits en el buffer de prebúsqueda y también dobla la frecuencia de reloj para transferir desde el buffer al bus de datos. Esto le permite doblar la velocidad de datos con respecto a DDR1.

DDR3 dobla a 8 bits en el buffer de prebúsqueda y aumenta la frecuencia de reloj para alcanzar una mayor velocidad de datos con respecto a DDR2.

## Velocidad De Las Memorias

Millones de transferencias por segundo se refiere al número de transferencias de datos que pueden darse en un segundo.

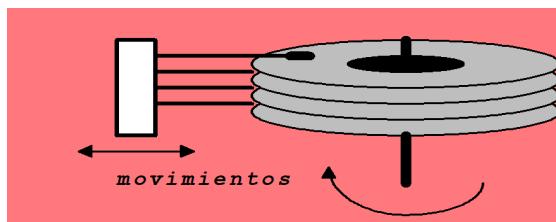
Se expresa en millones (megas) de transferencias por segundo (MT/s) o en GT/s (Gigas). Usualmente se aplica al número de transferencias efectivas, que pueden darse en un bus.

Por ejemplo, en sistemas de memoria DDR SDRAM, si la frecuencia de trabajo es de 100 MHz, como se transfieren datos tanto en los flancos de subida como de bajada de la señal de reloj, la velocidad de transferencia de datos será de 200 MT/s.

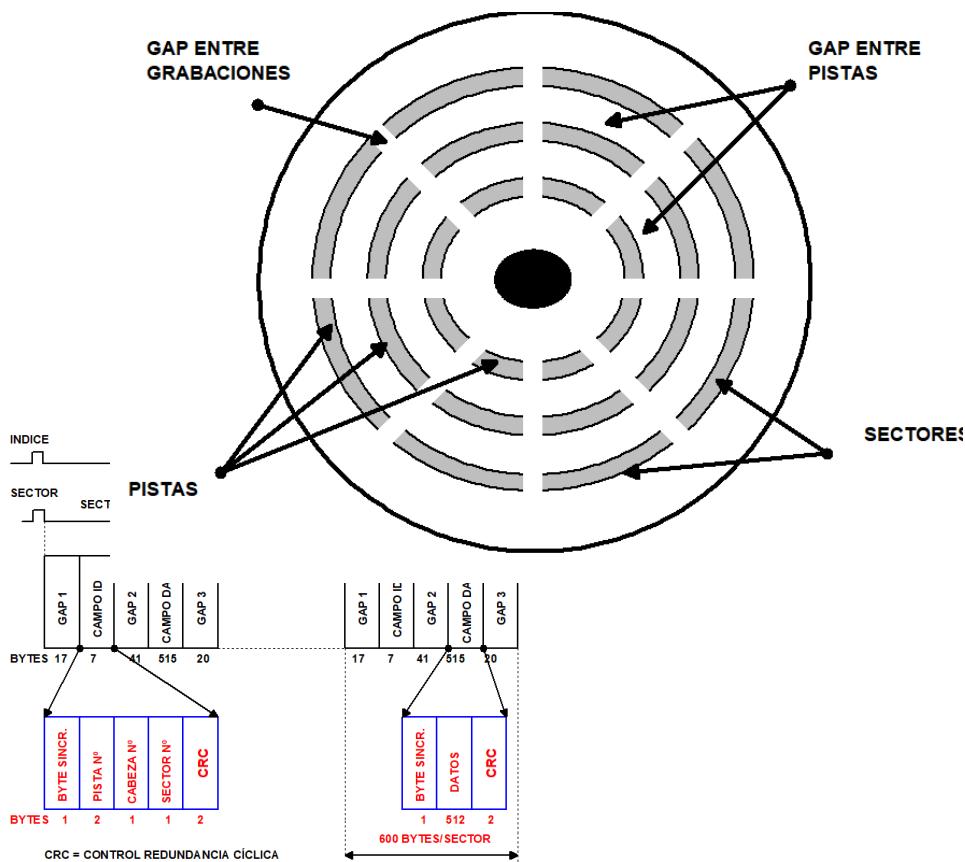
## Memorias Externas

Así se designan todos los almacenes conectados a la computadora mediante los terminales de e/s.

### Discos Rígidos (HDD)



- Formato del disco:



### HDD (Unidad De Estado Sólido) y SSD (Solid State Drive)

- Un HDD o SSD es un dispositivo de almacenamiento de datos que usa una memoria no volátil, como la memoria flash.
- Los SSD son menos sensibles a los golpes, son prácticamente inaudibles, tienen un menor tiempo de acceso y de latencia.
- Hacen uso de la misma interfaz que los discos duros (HDD) por lo que son fácilmente intercambiables.

### Discos Duros Híbridos (HHD)

Combinan ambas tecnologías, es decir, discos duros y memorias flash, que intentan aunar capacidad y velocidad a un precio inferior a un SSD.

### Cinta Magnética

- La grabación se hace hasta 9 bits en paralelo, 8 del dato y uno de paridad.
- La grabación del paquete de datos es precedida por un gap y un byte de sincronismo.
- Actualmente se usan magazines o cassettes especiales, con hasta 36 pistas paralelas en el ancho.
- Se sigue utilizando por economía para almacenes históricos.

### Disco Óptico

- Inicio de los sistemas digitales de audio.
- De allí nació el disco óptico de datos.

#### Variantes del cd para datos:

- CD-ROM (Read Only Memory Compact Disc)
- WORM (Writable Once - Read Many)
- EOD (Erasable Optic Disk)

### CD-ROM

- La grabación consiste en una serie de puntos deprimidos.
- La lectura se hace por medio de un haz de láser, el cual al reflejarse en la superficie normal tiene una polarización.
- Cuando se refleja en una depresión la polarización cambia, lo cual es detectado por un fotodiodo.
- En consecuencia, la presencia o ausencia de luz es la forma de leer los datos.
- Técnicas de grabación:
  - CAV (CONSTANT ANGULAR VELOCITY)
    - Se varía el espacio ocupado por cada bit = desperdicio de superficie pero se puede implementar un direccionamiento por pista y sector como en los discos magnéticos.

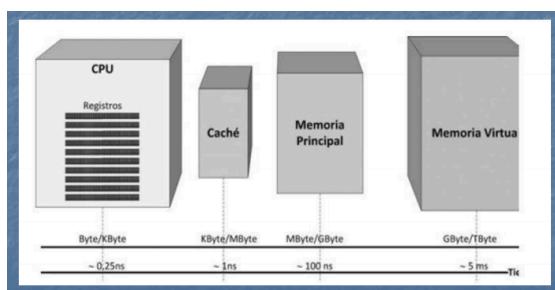
- CLV (CONSTANT LINEAR VELOCITY)
  - Se varía la velocidad de giro, disminuyendo a medida que nos alejamos del centro y nos permite aprovechar mejor la superficie disponible, pero la grabación es en espiral como en un disco de pasta o en una cinta magnética.

## DVD(Digital Versatile Disk)

Esencialmente es igual al cd, pero tiene mayor capacidad, lograda por un mayor acercamiento de las pistas y de los puntos de quemado, así como por la inclusión de una capa más de material grabable.

## **MEMORIA FÍSICA Y MEMORIA VIRTUAL (B)**

Jerarquía de las memorias



**Memoria Caché:** fabricada con memoria SRAM y controlada por el controlador de caché

**Memoria Principal:** Fabricada con memoria DRAM y controlada por el controlador de memoria principal.

**Memoria Virtual:** Fabricada con tecnología magnética y se controla desde el sistema operativo a través del controlador del disco duro.

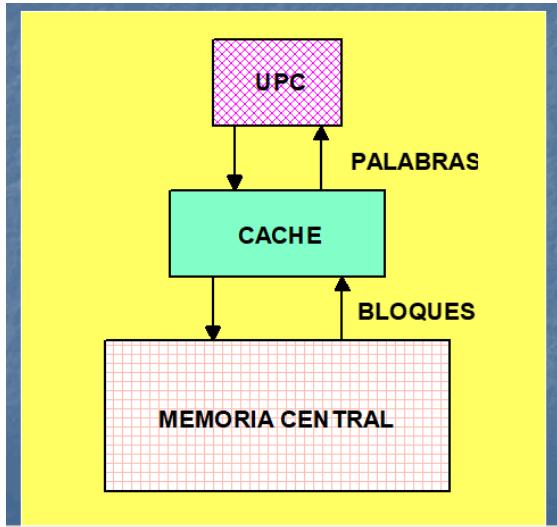
### **Memoria Virtual**

Conjunto de programas que dispone el sistema operativo que hace creer a la CPU que puede manejar directamente la memoria virtual. La CPU sólo puede acceder a una memoria física cuya capacidad está limitada por el tamaño del bus de direcciones

Para usar una memoria Virtual el procesador tiene que seguir una serie de pasos:

- Mecanismo de gestión de memoria llamado Unidad de Gestión de Memoria.
- Si está en memoria principal accede normalmente

### **Memoria Caché**



### Funcionamiento:

- 1\_ Verifica que el dato esté en la caché
- 2\_ Si esto ocurre, es entregado inmediatamente
- 3\_ Si no está, se debe cargar un bloque de la memoria principal

### Direccionamiento de la caché

Debe usarse la misma dirección que la de una palabra en la memoria central y debe hacerse un mapeo correcto

#### *Tipos de mapeo*

Mapeo directo: Cada línea de la caché ocupa varios bloques de la memoria central

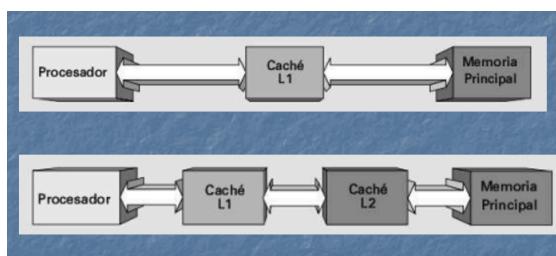
Mapeo asociativo total: Cada línea de la caché ocupa todos los bloques de la memoria

Mapeo asociativo por conjuntos: La caché es dividida en una serie de conjuntos

### Memoria Caché Niveles:

Nivel 1 (L1): Este nivel es el más cercano al procesador, por lo tanto, esta memoria caché es pequeña y rápida.

Nivel 2 (L2): Es el siguiente nivel de la jerarquía, de mayor tamaño y por lo tanto, más lento aunque con menos fallos de capacidad.



## **Tipos de conexión de memoria caché**

Serie → Todas las peticiones a memoria se realizan a través de la caché.

Paralelo → Todas las peticiones llegan simultáneamente a la caché y a la memoria principal.

## **Elementos para el diseño de caches.**

1- Capacidad de la caché → 256 kb - 512 kb

2- Tamaño de los bloques

- Bloques grandes ya que mejora la probabilidad de encontrar lo buscado en él
- Bloques chicos ya que mejora la velocidad de transferencia

## **Direccionamiento físico y lógico**

Memoria lineal → Dirección lógica es igual a la dirección física

Memoria segmentada → Memoria se almacena en segmentos

Memoria paginada → Las páginas tienen la misma cantidad de bytes

# **ENTRADA Y SALIDA**

## **UNIDAD DE ENTRADA/SALIDA**

Operación de Entrada y Salida +



Figura de camino que debe seguir el procesador para hacer una operación de E/S, el sistema operativo y los drivers son muy importantes en las transferencias de información.

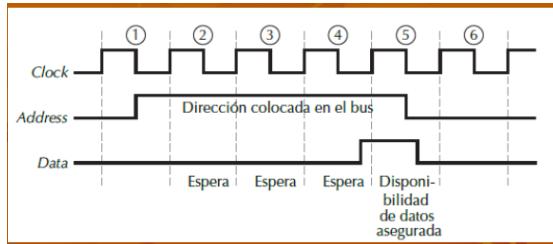
- + Existen varios dispositivos: de E/S básica (teclado, ratón, monitor), de almacenamiento (disco duro, CD, DVD) de impresión (impresora, escáner), etc.
- + Todos se conectan al sistema a través de una interfaz (unidad hardware/software) que suele estar ubicada en una tarjeta o adaptador y cuyas funciones principales son:
  - Interpretar las órdenes que recibe del procesador y transmitirlas al periférico.
  - Controlar la transferencia de datos entre el procesador y el dispositivo.
  - Informar al procesador el estado del dispositivo.

Las interfaces suelen clasificarse como serie o paralelo según cómo manejen la información o en interfaces generales o específicos según estén diseñados para dar soporte a un gran conjunto de dispositivos o a un tipo de dispositivo concreto.

La unidad física que permite que el dispositivo se conecte a su interfaz suele denominarse puerto.

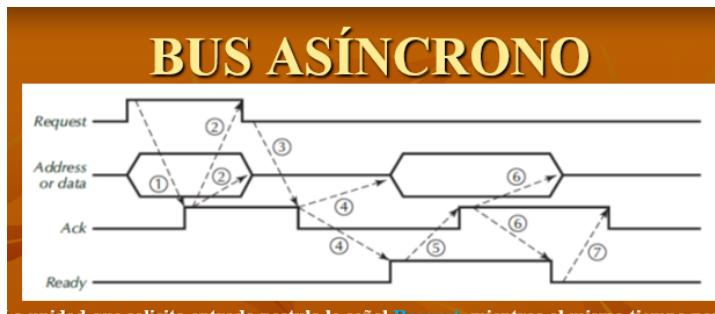
El interfaz se conecta mediante un bus de E/S con un controlador que suele encontrarse en un chipset que gestiona la transferencia de información con el procesador o con la memoria según los mecanismos de gestión de E/S disponibles en la arquitectura.

## Bus Síncrono



Una señal de reloj es parte del bus y los eventos ocurren en ciclos de reloj específicos, de acuerdo con un calendario acordado (protocolo de bus).

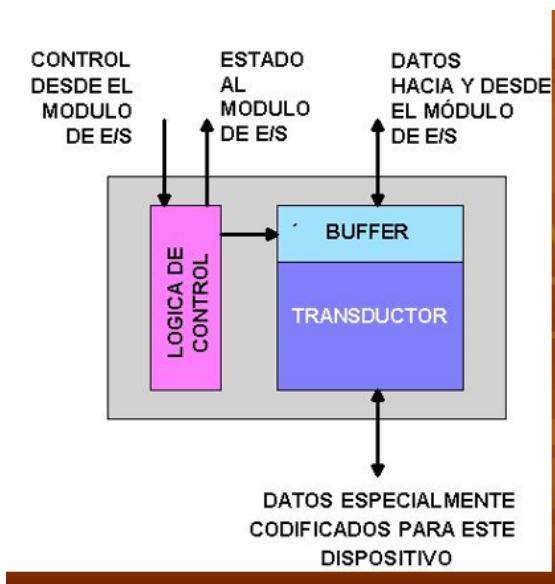
## Bus Asíncrono



Pueden acomodar dispositivos de distintas velocidades que se comuniquen a través de distancias más largas, la temporización fija del bus síncrono se sustituye con un protocolo de handshaking.

Bus asíncrono diferencia en la velocidad de procesamiento de los periféricos con respecto a la UC, si hay mucha dif de tiempo entonces es necesario el diálogo que se establece en una comunicación asincrónica ++++++++

## Periférico genérico



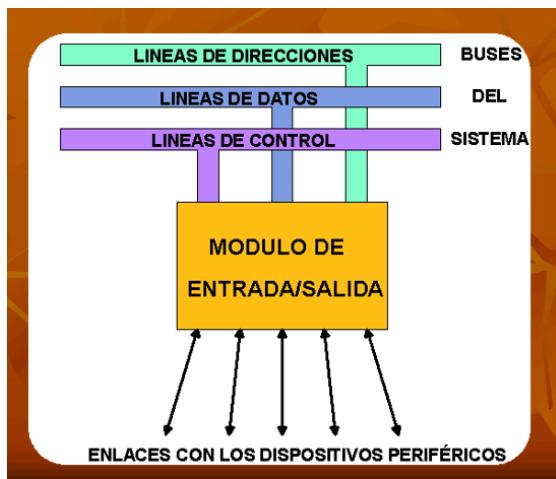
Señales de estado: Indican el estado del dispositivo → Listo, Ocupado o Desocupado

Señales de Control: Función del dispositivo → Leer, Transmitir o Recibir

Lógica de Control: Controla la actuación del dispositivo de acuerdo a las indicaciones dadas.

Transductor: Convierte los datos tanto a la forma de energía necesaria para el dispositivo, como en tamaño y/o formato. Lleva asociado el buffer para poder almacenar los datos durante el tiempo necesario para realizar su función.

## Módulo de E/S



- Encargados del intercambio de información de la CPU con el medio

### Funciones de los módulos de E/S

- Comunicación con la CPU
- Comunicación con los periféricos
- Almacenamiento de datos
- Detección de errores

### Procedimiento para entrada de datos

- 1- La CPU le pregunta al módulo para conocer el estado del periférico
  - 2- El Módulo responde dando su estado
  - 3- Si el dispositivo está operacional y preparado, la CPU solicita la transferencia mediante un comando.
  - 4- El módulo obtiene una unidad de datos del periférico.
  - 5- Esta unidad es transferida por el módulo a la CPU
- A veces se transfieren bloques de información

### Comunicaciones de la CPU

- **Decodificación de comandos** → El bus de control envía comandos que deben ser decodificados por el módulo de e/s.
- **Datos** → Los datos son intercambiados por la CPU y el módulo de E/S mediante el bus de datos
- **Reporte de estados** → Se debe conocer el estado del buffer del módulo (Ocupado o Listo)

- **Reconocimiento de la dirección** → El módulo de E/S deberá reconocer una única dirección por cada periférico que controla

### Tamponamiento de datos

El módulo de E/S debe ser capaz de proveer al tamponamiento de los datos para adecuar las distintas velocidades de E/S.

### Detección de errores

El módulo de E/S debe ser capaz de detectar errores en la transferencia de datos, dando aviso de ello a la CPU.

Estos errores pueden ser debidos a fallas mecánicas ,eléctricas o también pueden deberse a la modificación del patrón binario del código.

### Técnicas para E/S

- **Programada** → La CPU ejecuta un programa para hacer la transferencia y esto implica control total sobre la operación

#### Instrucciones de E/S

Estas dependen → Forma de direccionamiento del periférico, cantidad de periféricos y la dirección dada al periférico.

#### Direccionamiento de periféricos

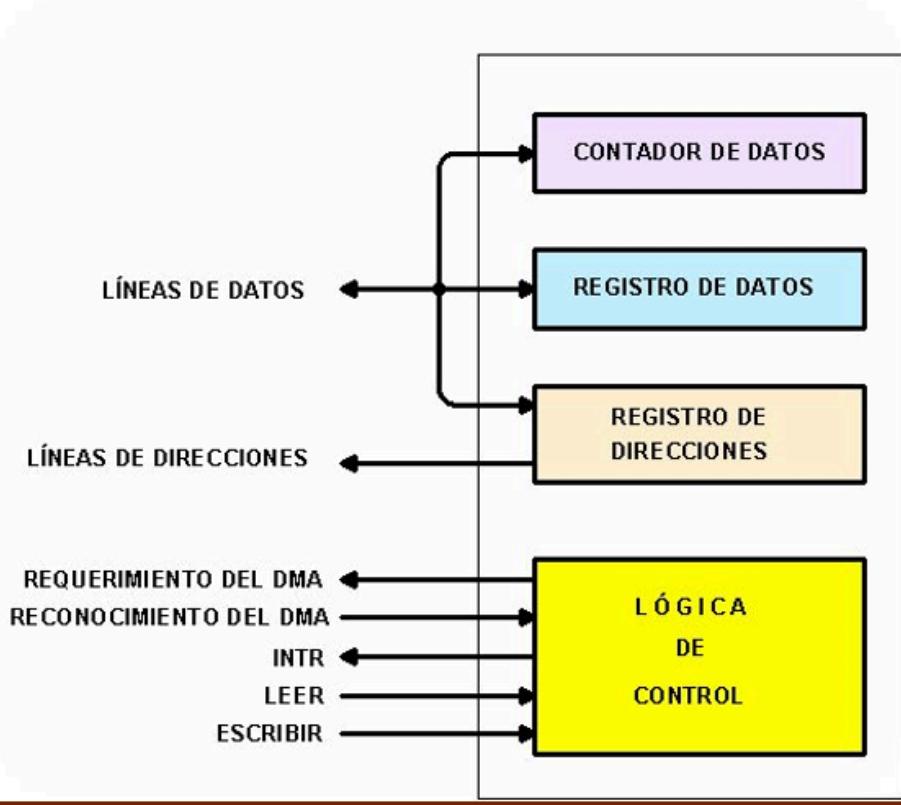
Por mapeo de memoria → 1 espacio para las direcciones de datos como de periféricos, con instrucciones de carga y almacenamiento

Aislado → 1 línea para direccionar al periférico, con instrucciones IN y OUT

- **E/S por interrupciones** → La CPU se hace cargo de entregar los datos o de recibirlos y almacenarlos
- **E/S por Acceso a Memoria** → La memoria y el módulo de E/S intercambian datos sin interrupción de la CPU

### Acceso directo a memoria (DMA)

# ACCESO DIRECTO A MEMORIA



65

## Funcionamiento del DMA

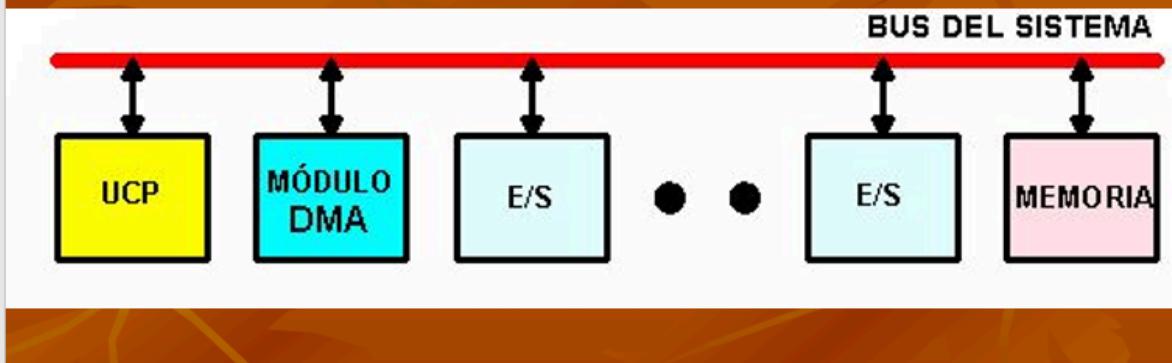
El módulo DMA simula el funcionamiento de la CPU para E/S, la CPU sigue con su tarea, el DMA comanda toda la transferencia.

Información para el DMA → Tipo de Operación, Dirección del dispositivo, localización de memoria de arranque para la lectura o escritura, cantidad de palabras que deben ser escritas o leídas

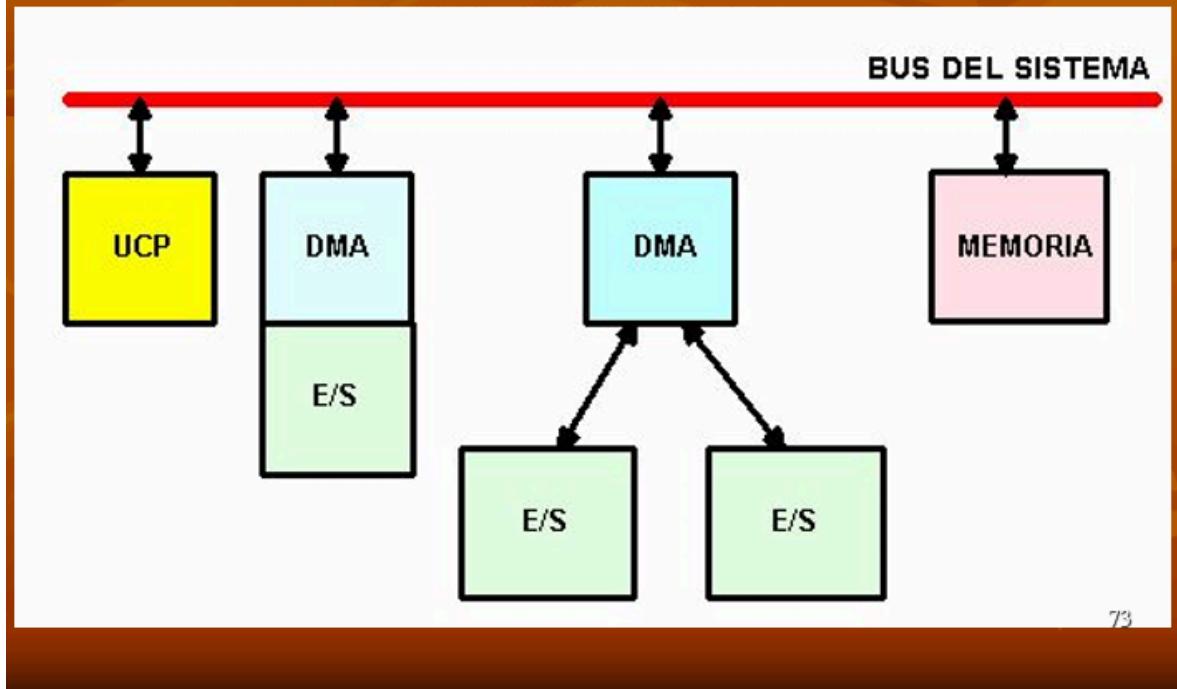
# **CONFIGURACIONES POSIBLES CON DMA**

- CON BUS ÚNICO Y DMA SEPARADA
- CON BUS ÚNICO Y DMA CON E/S INTEGRADAS
- CON BUS DE E/S

## **CON BUS ÚNICO Y DMA SEPARADA**

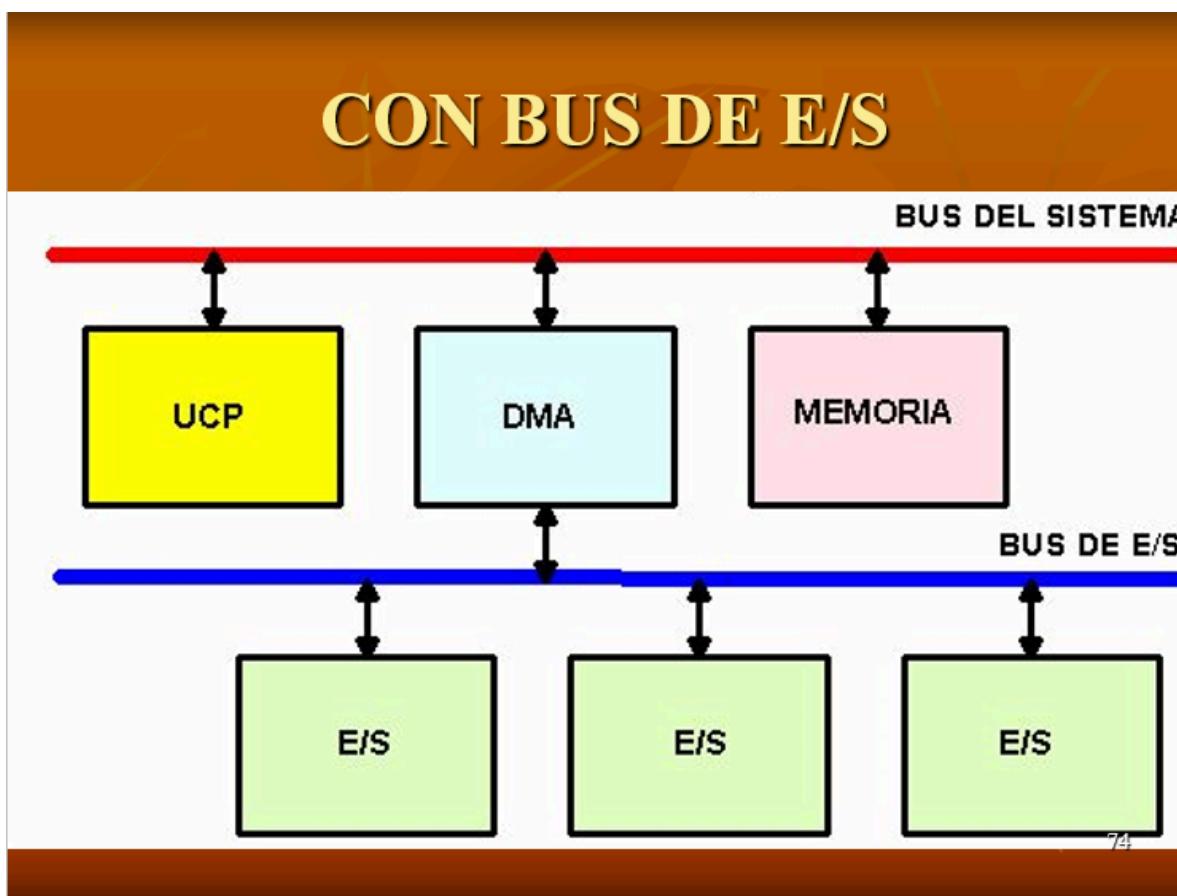


## CON BUS ÚNICO Y DMA-E/S INTEGRADAS



73

## CON BUS DE E/S



74

## Canales de E/S

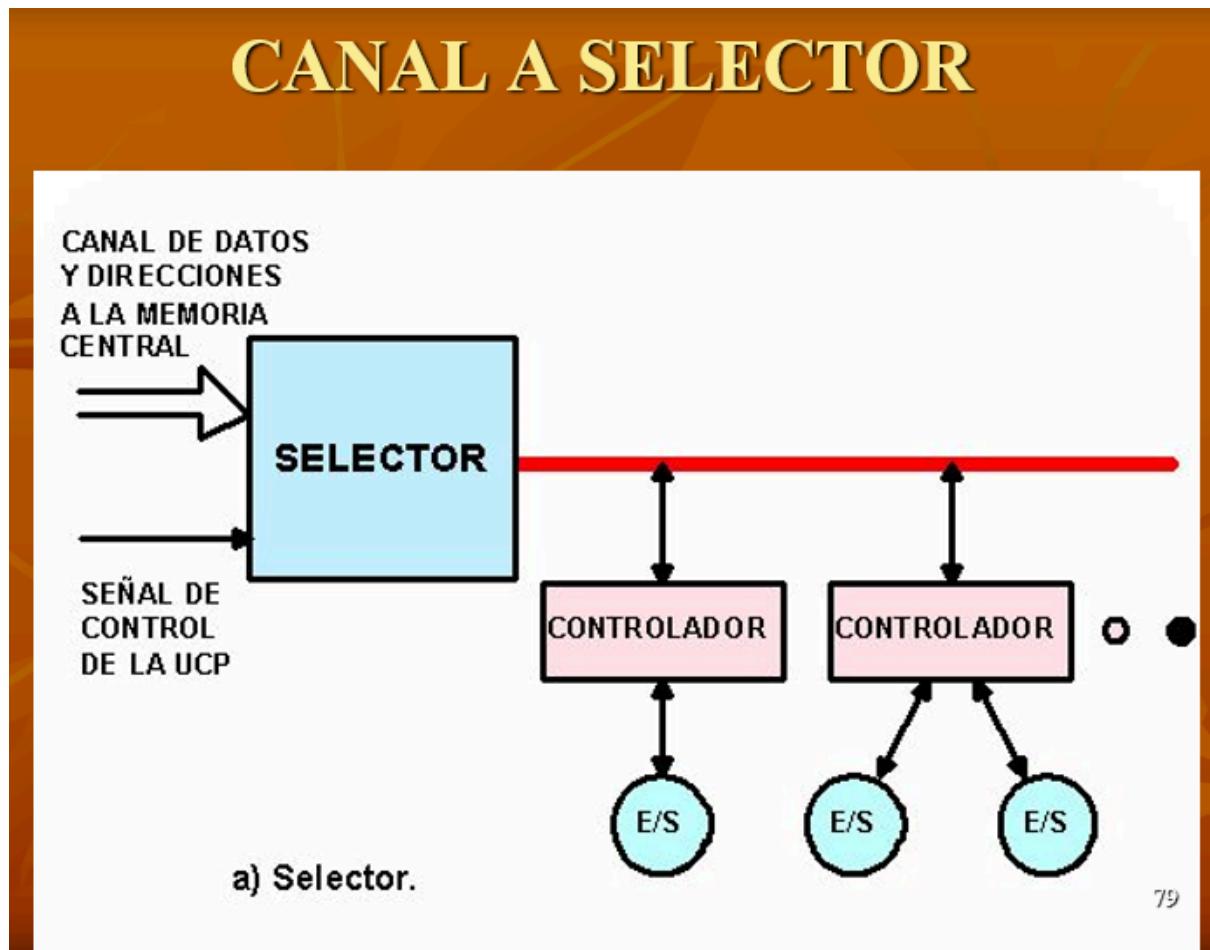
### Evolución de las funciones de E/S

CPU controla las E/S, luego se agrega un módulo de E/S y por último el módulo de E/S accede directamente a la memoria mediante el DMA

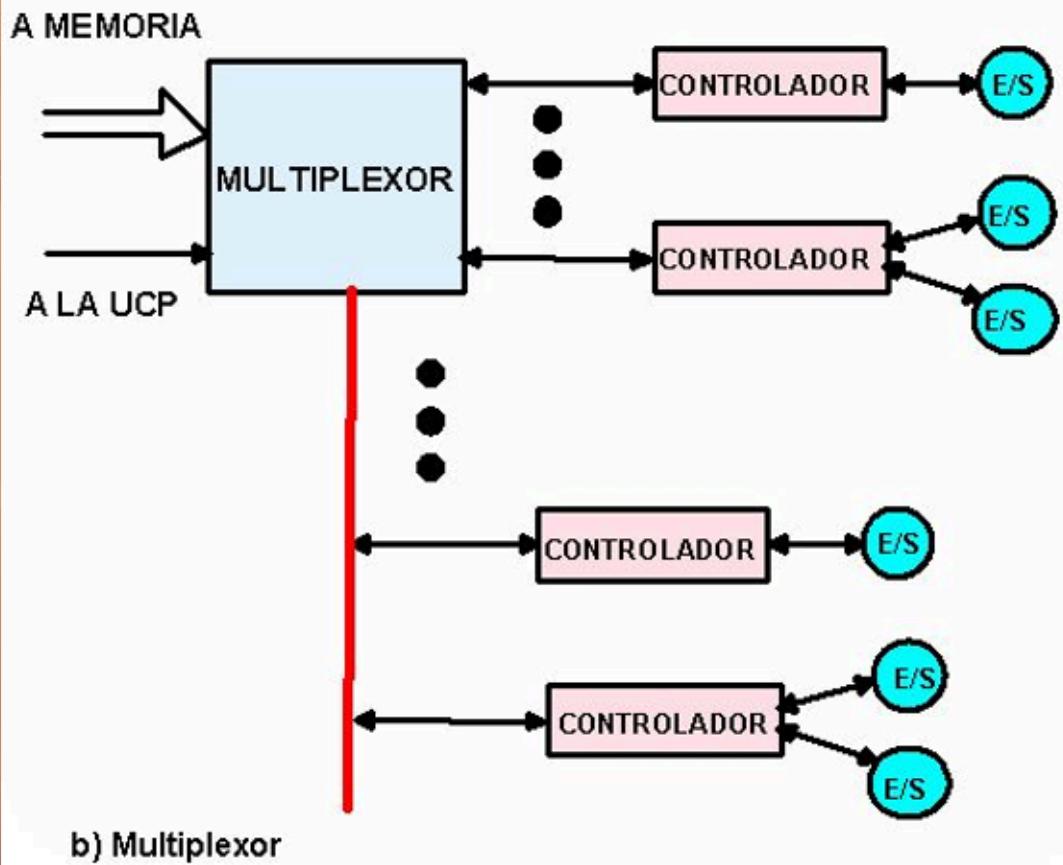
### Características de los canales de E/S

Son una extensión del concepto de DMA, la CPU no se encarga de la E/S sino que un procesador lo hace.

2 tipos de canales → Selector y Multiplexor



# CANAL A MULTIPLEXOR



## Interfase externa

Encargada de conectar el módulo de E/S con el periférico

Existen 2 tipos

Serie → Se utiliza para dispositivos más lentos (impresoras)

Paralelo → Se utiliza para dispositivos más rápidos (discos)

## BUSES DE ENTRADA Y SALIDA (B)

Un bus es un medio de transmisión compartido que interconecta dos o más dispositivos y permite que se establezca entre ellos una correcta comunicación.

Se compone de diferentes conductores eléctricos denominados líneas

Se pueden encontrar diferentes tipos de líneas en un bus:

- **Según su función:** Líneas de datos, líneas de direcciones o líneas de control.
- **Según su uso:** Líneas dedicadas o líneas multiplexadas.
- **Según sus características eléctricas:** Líneas unidireccionales o bidireccionales.

### Niveles de buses

- Nivel físico:
  - Capacidad de conexión (número máximo de dispositivos que pueden conectarse al bus)
  - Longitud máxima del bus
  - Soporte físico
  - Niveles de tensión
  - Frecuencia de funcionamiento
  - Tipos de conectores, etc.
- Líneas:
  - Ancho del bus (número total de líneas)
  - Ancho de datos
  - Líneas unidireccionales o bidireccionales, etc.
- Modo de operación.
  - Protocolos de transferencia (que determina el tipo de comunicación que se establece entre los dispositivos conectados al bus)
  - Protocolos de sincronización (que determina el inicio y el fin de cada transferencia de información)
  - Protocolos de arbitraje (que controla el acceso al bus).

### Jerarquía de buses

- Bus del sistema y bus de memoria:
  - Son los que conectan el procesador con el resto del sistema y la memoria principal con el controlador de memoria respectivamente. Se trata de buses rápidos, cortos y propietarios.
- Buses de expansión.
  - Se trata de buses más largos, lentos, abiertos y accesibles por el usuario.

## Ancho de Banda

El ancho de banda (bandwidth) del bus, se refiere a la cantidad de información que es capaz de transferir por unidad de tiempo.

$$BW = \text{Ancho de datos} \times f \times N^{\circ} \text{ transferencias/ciclo}$$

- Ancho de datos: n° de líneas de datos del bus.
- f: frecuencia de funcionamiento de esas líneas.
- N° transferencias/ciclo: transferencias de información que realiza el bus por cada ciclo de reloj.

## Optimización De Un Bus

Cualquier optimización que pretenda mejorar el ancho de banda de un bus debe ir orientada a incrementar alguno de estos tres aspectos:

- El ancho de datos.
- La frecuencia de operación del bus.
- El número de transferencias por ciclo.

## Ancho De Datos Y Frecuencia De Operación

El ancho está limitado por las interferencias que se producen entre líneas y otras, especialmente a frecuencias de operación altas.

Es difícil encontrar buses con un ancho de datos por encima de 64 o 128 bits.

La tendencia de optimización actual es la utilización de buses con un ancho cada vez menor, casi serie en lugar de paralelo, y que utilicen señalización diferencial para soportar frecuencias de funcionamiento mayores.

## Número de transferencias por ciclo

Mejorar en todo lo posible el modo de operación del bus y sus protocolos de transferencia, sincronización y arbitraje para que le diera tiempo a realizar más de una transferencia de información por ciclo.

Se aprovechan los flancos de subida y de bajada de las señales de reloj, o se combinan varias señales de reloj desfasadas.

Casi siempre se realiza un número de transferencias por ciclo que sea potencia de dos.

## Utilización de protocolos de comunicaciones de alto rendimiento

Los buses de E/S dentro de una computadora cada vez se parecen más a las redes de comunicaciones y se maneja la misma terminología: conexión punto a punto , paquete de datos, switch.

La idea principal es: evitar la señalización de control , y utilizar protocolos de red dentro de las arquitecturas, codificando la información de control junto con los datos, enrutando los paquetes resultantes mediante switches desde el origen hasta el destino.

## MICROPROCESADORES

### EVOLUCIÓN DE LOS MICROPROCESADORES (A)

#### Arquitectura interna

La arquitectura interna del microprocesador es la distribución física de sus componentes

Los primeros se crearon con la arquitectura de Von Neumann:

Memoria Principal, ULA, UC, U E/S

**Arquitectura 8088** → Intel divide al procesador en 2 partes:

Unidad de Ejecución: Realiza todas las operaciones

Unidad de Interfaz con el BUS: Controla las transferencias con el mundo exterior

#### Evolución en las Técnicas de Diseño de los Procesadores

- **Procesador Secuencial** → Se ejecuta una instrucción luego de otra
- **Procesador Segmentado** → Permite superponer en el tiempo la ejecución de varias instrucciones y aprovechar el paralelismo a nivel de instrucción dentro del procesador.

La ejecución de una instrucción completa en un procesador segmentado se divide en 5 etapas, cada instrucción pasará por aquéllas que necesite para completar su ejecución:

**Fetch:** Se busca en la memoria la instrucción que está almacenada en la dirección que indica el contador de programa

**Decode:** Se decodifica la instrucción separando sus diferentes campos. El código operativo indica que tipo de instrucción es y qué tipo de operación se debe realizar en la ruta de datos. Si es necesario, se leen 1 o 2 operandos de los registros del procesador.

**Execution:** Se ejecuta la operación utilizando para ello algún tipo de ALU

**Memory Access:** Acceder a memoria para leer o escribir

**WriteBack:** Pasar el resultado a un registro

- **Procesador Segmentado Multifuncional** → Cuatro etapas: ALU de enteros, sumador/restador en coma flotante segmentado, multiplicador/divisor en coma flotante segmentado y traducción de dirección virtual a física

❖ **Procesador Segmentado Multifuncional Superescalar Estático  
Con Emisión De Dos Instrucciones Por Ciclo**

Si la planificación es estática, la emisión de instrucciones debe realizarse en orden y los riesgos se detectan y resuelven antes de la emisión. Por lo tanto, las instrucciones que se emiten simultáneamente deben ser independientes y satisfacer una serie de restricciones.

Como la emisión debe realizarse en orden, si en un bloque de instrucciones se encuentran dependencias, sólo se podrán emitir las instrucciones anteriores a la primera que presenta una dependencia. La ordenación que permite aprovechar el paralelismo entre instrucciones debe ser realizada por el compilador, ya que el procesador emite y ejecuta las instrucciones en orden, simplemente comprueba que las instrucciones que se van a emitir en paralelo pueden ejecutarse en paralelo porque no hay riesgos que lo impidan.

❖ **Procesador Segmentado Multifuncional Superescalar Dinámico  
con Emisión de dos instrucciones por ciclo**

La planificación dinámica de instrucciones intenta reducir al máximo el número de paradas del procesador modificando el orden de las instrucciones en tiempo de ejecución para aprovechar al máximo el paralelismo a nivel de instrucción (ILP) y que los recursos del procesador estén desocupados el menor tiempo posible.

Se basa en buscar y decodificar las instrucciones en el orden secuencial en el que aparecen en el código, pero en permitir que la lectura de operandos y las etapas posteriores se realicen fuera de orden. Es decir, la emisión de instrucciones , que no es más que comprobar que éstas tienen disponibles sus operandos y las unidades funcionales que necesitan para ejecutarse, puede desordenarse para que los riesgos de datos no obliguen a parar el procesador.

## Arquitectura Interna. Ley De Moore

El número de transistores que incorpora un microprocesador se duplicaría cada 18/24 meses.

## ARQUITECTURA 80386

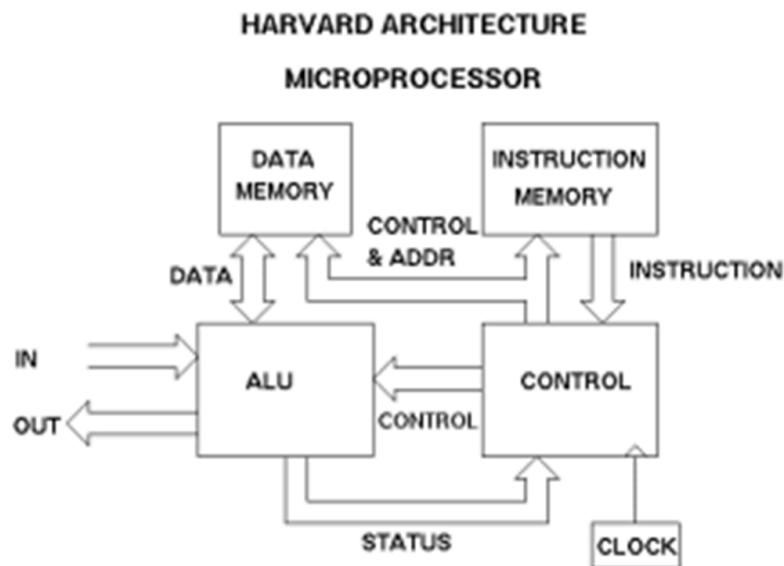
- Primer arquitectura de Intel de 32 bits
- Tiene una CPU compuesta por tres unidades: la unidad de prebúsqueda de instrucción, la unidad de decodificación y la unidad de ejecución.
- Posibilidad de conexión a memoria caché.
- Primer procesador de Intel en soportar multitarea, es decir, permite correr múltiples programas al mismo tiempo.

## ARQUITECTURA 80486

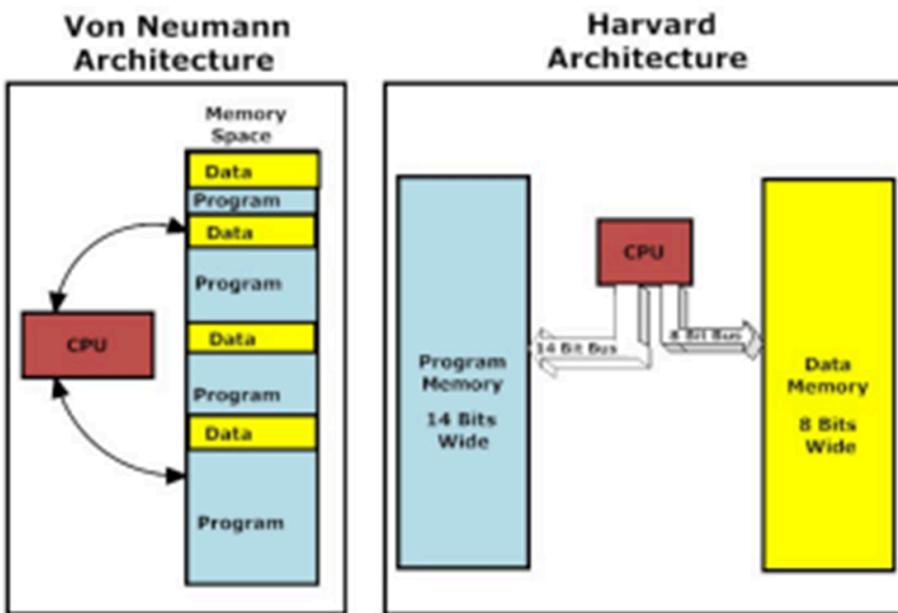
Agrega:

- Unidad de Control microprogramada
- Unidad de caché
- Unidad de coma flotante
- Bus optimizado

## Arquitectura Harvard



# Comparación



La arquitectura Harvard está diseñada para resolver las debilidades de la arquitectura Von Neumann. El procesador dispone de dos memorias independientes con las que se comunica con buses propios: la memoria de instrucciones y la memoria de datos.

El tamaño de las instrucciones no está relacionado con el de los datos, pudiéndose así optimizar para que cualquier instrucción ocupe una única posición de memoria de programa.

En la actualidad la memoria caché de nivel uno (L1) es la que está dividida en dos memorias independientes una para instrucciones y otra para datos.

## TIPOS DE MICROPROCESADORES (B)

### Arquitectura RISC (Reduced Instruction Set Computers)

Son máquinas de estructura y software muy simples, mientras más simple es la estructura, más rápido es el funcionamiento.

#### Motivos

Habían cosas en hardware que el software no podía resolver

#### Características RISC

- Conjunto de instrucciones de máquina simple y limitado
- Unidad de control cableada
- Maximización del uso de registro
- Operaciones de ciclo único

#### Uso de registros

Cuantos son utilizados, cuál es su necesidad, el tiempo que pasa entre que es cargado y que cambia su contenido

#### Operaciones de ciclo único

Son operaciones que se realizan en un solo ciclo de reloj, lo que las hace más rápidas y sencillas.

#### Instrucciones y modos de direccionamiento

El RISC tiene un limitado conjunto de instrucciones, generalmente no más de 50, por lo tanto posee pocos modos de direccionamiento (sólo los más simples)

Las instrucciones deben tener un formato fijo y sencillo

#### Mediciones sobre programas

Mediciones sobre operandos → Son constantes o variables, son enteros o caracteres, etc

Mediciones sobre operaciones → Con qué frecuencia aparecen en los distintos programas, especialmente en la transferencia de control

#### Tipos de Mediciones

*Estáticas* → Se refieren al texto del programa, nos indicarán la cantidad de memoria usada para almacenar el programa

*Dinámicas* → Se refiere a la ejecución de programas (códigos de máquina)

#### Encauzamiento

Es una forma de mejorar el rendimiento de un computador, se utilizan cauces en la UC y ULA

## **Controversia RISC-CISC(COMPLEX INSTRUCTION SET COMPUTERS)**

Durante años se construyeron equipos más complejos, ayudados por el costo de hardware

El RISC rompe con esta tendencia, provocando ciertas reacciones a favor y en contra.

## **ARQUITECTURA EN PARALELO**

### **Computadoras en paralelo**

Se conoce como “Cuello de botella de Von Neumann” a la estructura de una computadora con una sola CPU, que realiza solicitudes en secuencia sobre un bus hacia la memoria.

#### Cuestiones para un sistema en paralelo:

- 1- La naturaleza, tamaño y número de los elementos procesadores
- 2- La naturaleza , tamaño y número de los módulos de memoria
- 3- La estrategia de interconexión entre procesadores y memorias

#### Esquemas de interconexión

Estáticos → Se conectan los componentes de manera fija, en forma de estrellas

Dinámicos → Se conectan los componentes a una red de interruptores que puede en forma dinámica dirigir mensajes entre los componentes

#### Paralelismo

Paralelismo ordinario → Correr grandes porciones de software en paralelo con poca o ninguna comunicación entre las partes

Paralelismo fino → Muchas operaciones de ULA operando al mismo tiempo

#### Acoplamiento

Bajo acoplamiento → Sistema que las CPU interactúan entre sí con bajo ancho de banda

Alto acoplamiento → Sistema que los componentes interactúan entre sí con un alto ancho de banda

#### Granularidad

La granularidad del cálculo es la medida de la cantidad de operaciones asignadas a cada elemento procesador

- Granularidad fina: pocas operaciones en cada elemento procesador.
- Granularidad gruesa: muchas operaciones en cada elemento procesador.

#### **Clasificación de las arquitecturas → (Flynn)**

### Simple flujo de Instrucciones y Simple flujo de Datos

SISD: Tiene un flujo de datos, uno de instrucciones y realiza una operación a la vez

### Simple flujo de Instrucciones y Múltiples flujo de Datos

SIMD: Tienen varias ULA para llevar a cabo una instrucción con diferentes conjuntos de datos

### Múltiple flujo de Instrucciones y Simple flujo de Datos

MISD: Tienen muchas instrucciones operando sobre los mismos datos

### Múltiple flujo de Instrucciones y Múltiple flujo de Datos

MIMD: Son varias CPU que operan como parte de un sistema más grande

## **Ejecución de instrucciones en paralelo**

- SISD con múltiples unidades funcionales
- SISD con segmentación de instrucciones
- SIMD procesamiento vectorial
- SIMD con procesador de arreglos

## **Interruptor de barras cruzadas**

En cada intersección de líneas horizontales y verticales existe un punto de cruce que es un interruptor que se puede abrir o cerrar dependiendo de si se van o no a conectar las líneas verticales y horizontales

## **Modos de expresar el rendimiento**

Tiempo paralelo: Tiempo total que requiere un programa para su ejecución en un procesador paralelo

Speedup: La relación entre el tiempo de ejecución de un programa en un procesador secuencial y el tiempo paralelo

Ley de Amdahl: Aumento de velocidad en función de la cantidad de procesadores y la cantidad de operaciones que deben ejecutarse de manera secuencial

## **Multiprocesadores De Memoria Compartida En Base A Buses**

- Si la memoria es tan rápida como las CPU, esta distribución está bien.
- Si la memoria es más lenta que la CPU, sería mejor dividir la memoria en varios módulos independientes

## **Multiprocesadores De Memoria Compartida Con Etapas Múltiples**

- Los buses no se adaptan a los grandes sistemas
- La forma de implementar una memoria compartida con un número importante de CPU, es dividir la memoria en módulos

## **Paralelización de un algoritmo**

Convertir un algoritmo en paralelo puede mejorar el rendimiento hasta un límite que está determinado por la cantidad de operaciones secuenciales

