



Resumen de Arquitectura del Comp

Arquitectura De Computadores (Universidad Católica de Salta)

Unidad I: Circuitos de Lógica Digital

• Computadoras Digitales

Es un **Sistema Digital** que ejecuta tareas computacionales

- *Sistema Digital: por que maneja elementos discretos de información (Binarios) → pueden ser impulsos eléctricos*

Es una **interconexión de Módulos Digitales**

Un Sistema de Computadora se subdivide a veces en dos entidades funcionales

- Hardware

Consta de todos los componentes electrónicos y dispositivos electromecánicos que comprenden la entidad física del dispositivo. Se divide por lo general en tres grandes partes:

1- **La Unidad Central de Procesamiento (CPU)**, que esta a su vez contiene:

- **Unidad Aritmética y Lógica** → para manipular datos

- **Varios Registros** → para almacenar los datos

- **Circuitos de Control** → para leer de la memoria y ejecutar instrucciones.

2- **Memoria Principal de la Computadora (RAM)**, que almacena las instrucciones y los datos, por lo que la CPU puede acceder a cualquier parte de la memoria en forma aleatoria y recuperar la información binaria dentro de un intervalo fijo.

3- **El Procesador de E/S**, contiene circuitos electrónicos para comunicarse y controlar la transferencia de información entre la computadora y el mundo exterior.





- Software



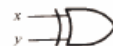
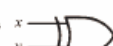
Consiste en las instrucciones y los datos que la computadora manipula para ejecutar las diversas tareas de procesamiento de datos. En tanto a los programas incluidos en un paquete de Software de sistema se les conoce como **Sistema Operativo**. Se distinguen de los programas de aplicación escritos por el usuario con el propósito de resolver problemas particulares.

• Compuertas Lógicas

Son Bloques de Hardware que producen señales binarias (variables 0 y 1) → cuando el requisito de entrada son satisfechos.

“n” variables tienen 2^n combinaciones posibles.

Nombre	Símbolo gráfico	Función algebraica	Tabla de verdad															
AND		$F = xy$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = x + y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inversor		$F = x'$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	F	0	1	1	0									
x	F																	
0	1																	
1	0																	
Separador		$F = x$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	x	F	0	0	1	1									
x	F																	
0	0																	
1	1																	

Nombre	Símbolo gráfico	Función algebraica	Tabla de verdad															
NAND		$F = (xy)'$	<table> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = (x + y)'$	<table> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
OR-exclusiva (XOR)		$F = xy' + x'y$ $= x \oplus y$	<table> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
NOR-exclusiva o equivalencia		$F = xy + x'y'$ $= x \odot y$	<table> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

• Algebra Booleana

El algebra de Boole como cualquier otro sistema matemático deductivo puede ser definida por un conjunto de elementos, un conjunto de operadores, un numero de axiomas o postulados.

• Circuitos Combinatorios

Definición

Un Circuito Combinatorio consiste en variables de entrada, compuestas lógicas y variables de salida.

El análisis de un circuito combinatorio comienza con un diagrama de circuito lógico determinado y culmina con un conjunto de de funciones booleanas o una tabla de verdad.

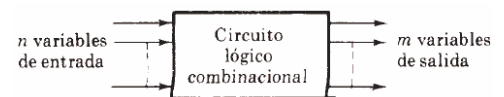


Diagrama de bloque de un circuito combinatorial

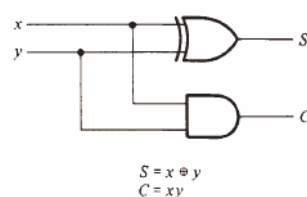
Sumadores

- Semisumador

Es un Circuito Combinatorio que ejecuta la suma de 2 bits (suma de dos dígitos binario)

S: representa el bit menos significativo de la suma

C: es el bit de acarreo, es necesaria esta salida por que la suma de 1+1 (ambas entradas = a 1) es el binario 10, que tiene dos dígitos



x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = (x + y)(x' + y')$$

$$C = xy$$

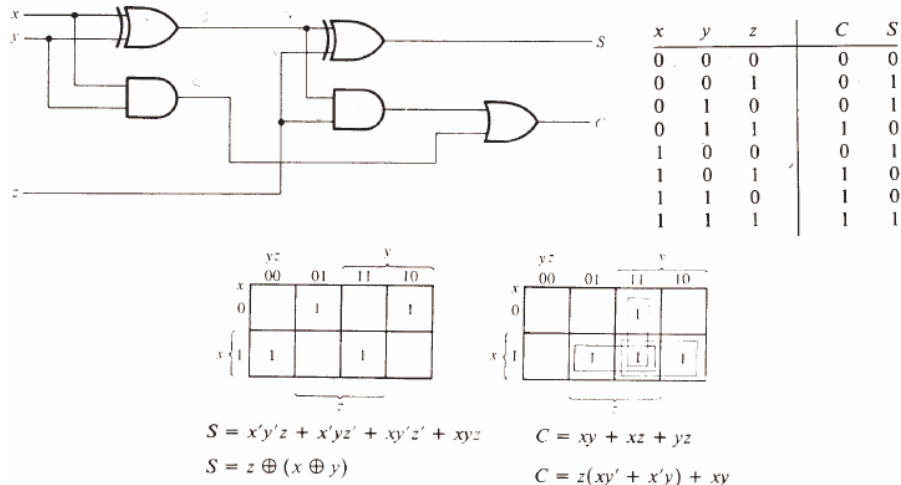
- Sumador Completo

Es un Circuito Combinatorio que ejecuta la suma de tres bits de entrada

Consiste en tres entradas y dos salidas

x e y representan los dos bits significativos a sumarse

z representa el acarreo de la posición menos significativa previa



Configuración de un sumador completo con dos sumadores medios y una compuerta OR

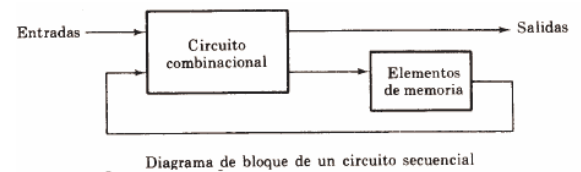
• Circuitos Secuenciales

Definición

Es una interconexión de **Flip-Flops** y **compuertas**.

Son **Circuitos Combinatorios** al cual se les conectan **Elementos de Memoria** para formar un camino de realimentación.

Los elementos de memoria son capaces de almacenar información binaria dentro de ellos. Los elementos de memoria usados en estos circuitos se llaman Flip-Flops (.)



Flip-Flops

Definición: Son elementos de almacenamientos empleados en los circuitos secuenciales con reloj (celdas binarias capaces de almacenar un bit de información).

Un circuito Flip-Flop tiene dos entradas, una para el valor normal y uno para el valor complemento del bit almacenado en él. La información binaria puede entrar a un F-F en una variedad de formas, éste hecho determina diferentes tipos de F-Fs.

- Circuitos Básicos

Un circuito F-F puede construirse con dos compuertas NAND o dos compuertas NOR. La conexión de acoplamiento intercrucado de la salida de una compuerta a la entrada de la otra constituye un camino de realimentación.

Cada F-F tiene dos salidas Q y Q' y dos entradas S (set) y R (reset).

En particular este circuito Recibe el nombre de Flip-Flop RS acoplado directamente o bloqueador SR

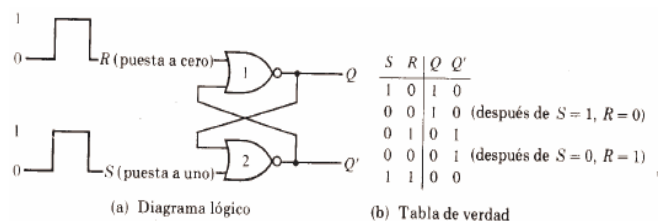
Circuito Básico con compuertas NOR (es un circuito asincrónico)

- Partiendo de: entr de puesta a 1 (**set**) es **1** y la entr de puesta a cero (**reset**) sea **0**...

- Compuerta2=1, su salida $Q'=0 \rightarrow$ ambas entradas de

Compuerta1=0, su salida $Q=1$

- De la misma manera un 1 en la entrada de puesta a cero (reset) cambia la salida Q a 0 y Q' a 1



Circuito flip-flop básico con compuertas NOR

- Flip-Flop RS Temporizado

Consiste en un **F-F básico NOR** y dos compuertas **AND**

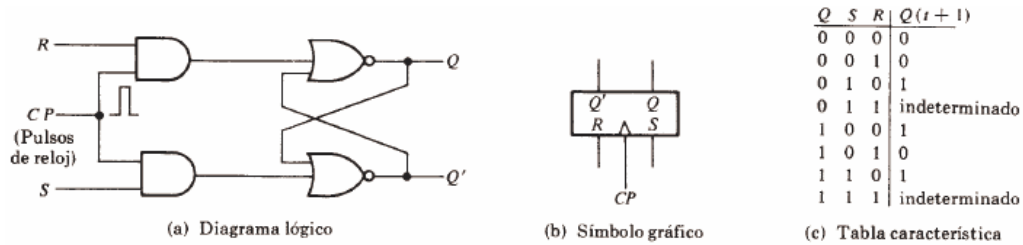
- Las Salidas de las compuertas **AND** = 0 si \rightarrow pulso de reloj (c) = 0, independientemente de los valores de entrada de S y R

- Cuando C vaya a 1, la información de las entradas S y R llegan al **F-F**

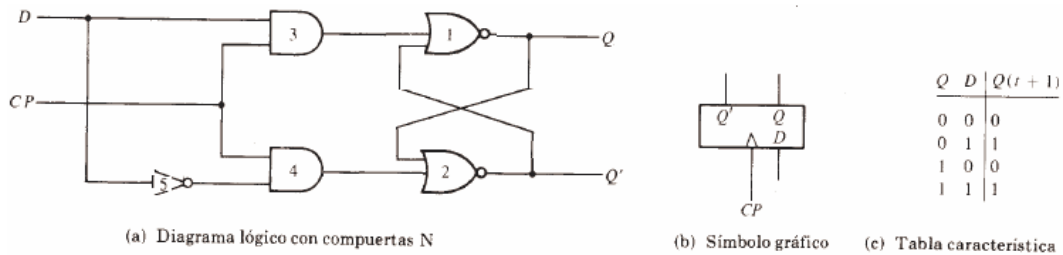
- El estado de de **puesta a uno** se logra con: $S=1, R=0$ y $C=1$

- Para cambiar el estado de **puesta a cero (o borrado)** las entradas deben ser $S=0, R=1$ y $C=1$

- **Indeterminación:** con $S=1$ y $R=1$ causará que ambas salidas vayan momentáneamente a 0 \rightarrow Cuando se quite el pulso, el estado del F-F será **indeterminado (podría resultar cualquier estado)** \rightarrow dependiendo de si la entrada de puesta a uno o la puesta a cero, permanezca el mayor tiempo, antes de la transición a 0 al final del pulso.

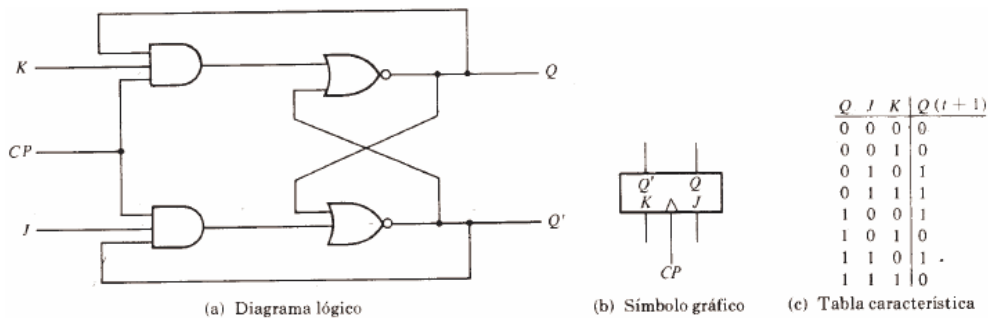


- Flip-Flop D



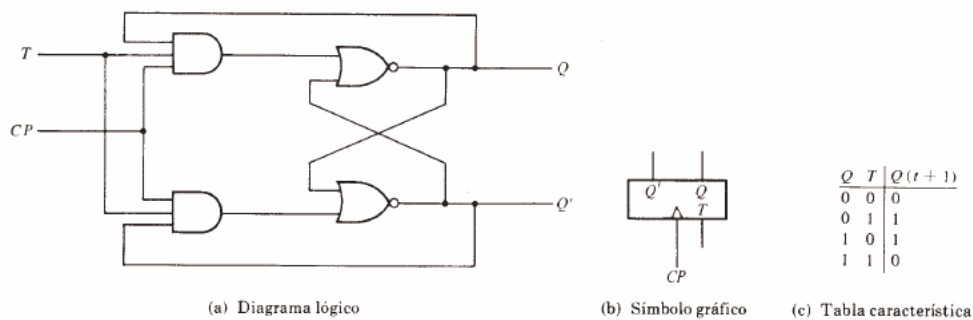
Flip-flop D temporizado

- Flip-Flop JK



Flip-flop JK temporizado

- Flip-Flop T



Flip-flop T temporizado

- Flip-Flop Disparado por Flancos

- Flip-Flop Maestro Esclavo

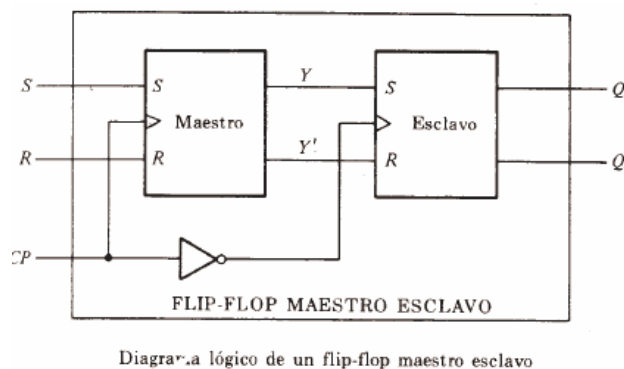


Tabla de Exitacion

Tabla que enlista las combinaciones de entrada requeridas para un cambio de estado dado. Son la tablas mostradas junto a cada uno de los diagramas de los F-F's

Unidad 3: transferencia de Registro y microoperaciones

• Lenguaje de Transferencia de Registros

Def. Sist. Digital

- Es una interconexión de módulos de hardware que realizan una tarea específica de procesamiento de información. Construidos a partir de componentes digitales.

Módulos Digitales

- Se definen mejor por los registros que contienen y las operaciones que realizan sobre los datos que almacenan.

Microoperación

- Es una operación ejecutada sobre los datos almacenados en los registros. De otra manera se dice que es una operación ejecutable sobre uno o más registros.

Lenguaje de Transferencia de Registros

- Se denomina a la notación que se utiliza para describir las transferencias de microoperaciones entre registros.

• Transferencia de Registros

Representación de Transf. de Reg.

- La transferencia de información de un registro a otro se representa en forma simbólica mediante un operador de sustitución ($R2 \leftarrow R1$)

- Se diferencian: Reg. Fuente

Reg. Destino: debe tener capacidad de carga paralela

Función de control

- Es una variable booleana (0-1)

- Es conveniente separar las variables de control de la operación de transferencia de registros al especificar una función de control. Ej (P: $R2 \leftarrow R1$) en sincronía con el reloj aplicado al registro.

• Transferencia de Canal y de Memoria

Bus Común (o canal común)

- Es un conjunto de líneas comunes, una para cada bit de un registro, mediante las cuales se transfiere información binaria una a la vez.

- Posee \rightarrow un Multiplexor para cada bit (significativo) del Registro fuente

- En general un Sist. de Bus de "n" líneas \rightarrow "n" Multiplexores (n x 1)

\rightarrow "k" Reg. de "n" bits

- La constitución de un sistema de bus para cuatro registros se muestra a continuación:

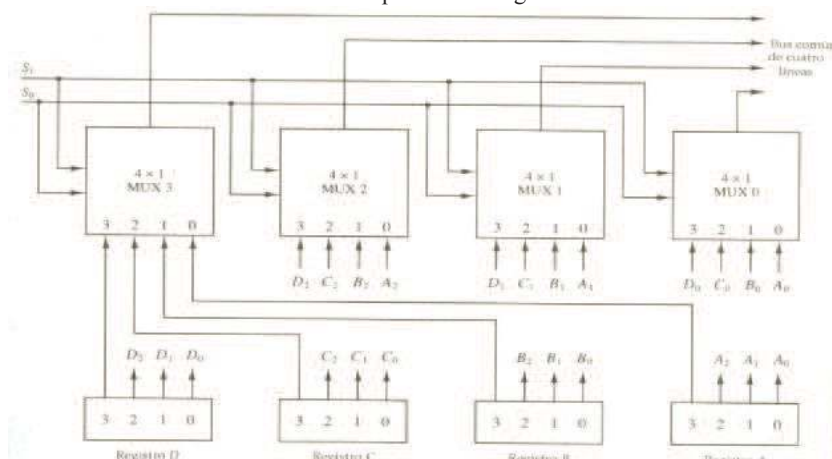


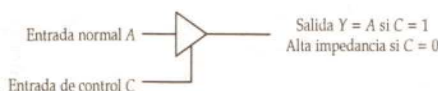
Tabla de función

S1	S2	Registro Selecc.
0	0	A
0	1	B
1	0	C
1	1	D

Canal de Bus de 3 Estados

- Compuerta de Tres Estados

Es un circuito digital que presenta tres estados (0-1-Alta impedancia), el estado de alta impedancia actúa como un circuito abierto, lo que significa que la salida está desconectada y no tiene importancia lógica.



Transferencia de Memoria

- Operación de Lectura:

Se denomina a la operación que realiza transferencia de información de una palabra de memoria al ambiente externo.

$Leer DR \leftarrow M[AR]$

- Operación de Escritura:

Se denomina a la transferencia de nueva información para almacenarse en memoria.

$Escribir M[AR] \leftarrow R1$

• Microoperaciones

Las microoperaciones que se encuentran con mayor frecuencia en las computadoras digitales se clasifican en cuatro categorías:

- **Microoperaciones de transferencia de registro:** que transfieren de información binaria de un registro a otro.
- **Microoperaciones aritméticas:** que ejecutan operaciones sobre datos numéricos almacenados en los registros.
- **Microoperaciones lógicas:** ejecutan operaciones de manipulación de bits sobre datos no numéricos almacenados en reg.
- **Microoperaciones de corrimiento:** que ejecutan operaciones de corrimiento sobre los datos almacenados en registros.

Microoperaciones Aritméticas

Ejecutan operaciones sobre datos numéricos almacenados en los registros.

Las microoperaciones básicas son: **Suma, Resta, incremento, decremento y corrimiento.**

- **Microoperación de sumar:** $R3 \leftarrow R1 + R2$ (la suma se transfiere al registro R3).

- **Microoperación de restar:** $R3 \leftarrow R1 + R2' + 1$ (en vez de $R1 - R2$ la resta binaria realiza con complemento a 1 y compl. a 2).

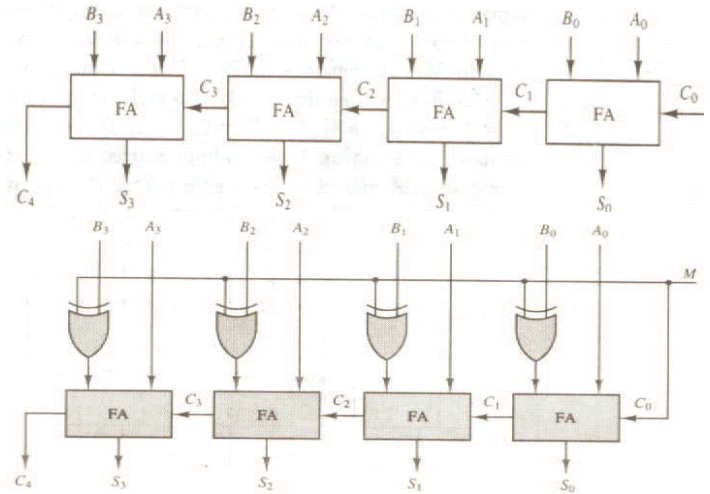
- **Microoperaciones de Incremento y Decremento:** se representan mediante las operaciones de +1 y -1, respectivamente.

Seleccionar			Entrada	Salida	Microoperaciones
S_1	S_0	C_m	Y	$D = A + Y + C_m$	
0	0	0	B	$D = A + B$	Sumar
0	0	1	B	$D = A + B + 1$	Sumar con acarreo
0	1	0	\bar{B}	$D = A + \bar{B}$	Restar con préstamo
0	1	1	\bar{B}	$D = A + \bar{B} + 1$	Restar
1	0	0	0	$D = A$	Transferir A
1	0	1	0	$D = A + 1$	Incrementar A
1	1	0	1	$D = A - 1$	Decrementar A
1	1	1	1	$D = A$	Transferir A

- Sumador Binario

Es un circuito digital que suma números binarios de cualquier longitud. Se construye con circuitos sumadores completos conectados en cascada, con el acarreo de salida de un sumador completo conectado en acarreo de entrada del siguiente sumador completo.

Ej Sumador Binario de 4bit.

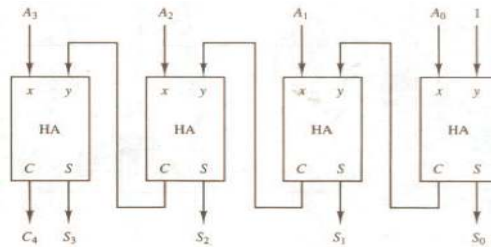


- Sumador Sustractor

Las operaciones de suma y resta se pueden combinar en un circuito común al incluir una compuerta OR exclusiva con cada sumador completo. Ej Sum Sust de 4 bit

Funcionamiento: entrada de modo M controla la operación $\rightarrow M=0$ (suma) - $M=1$ (resta)

Ej de compl. a 2: $B_0' + 1 + A$



- Incrementador Binario

Es un conjunto de semisumadores conectados en cascada (Serie). Ej Increment Bin de 4 bit.

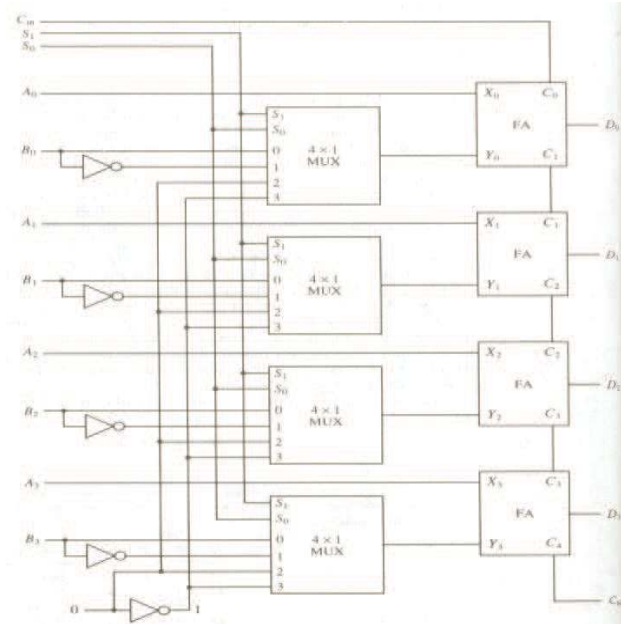
- Circuito Aritmético

Las microoperaciones aritméticas descritas pueden implantarse en un circuito aritmético compuesto, cuyo componentes básicos son:

- * Sumadores completos en paralelo
- * Multiplexores para elegir las diferentes operaciones aritméticas a realizar.

Características:

- * Se lo controla con un Bus de entrada, para obtener las diferentes operaciones.
- * Usa las entradas del FA conectada a un MUX (para las diferentes operaciones)

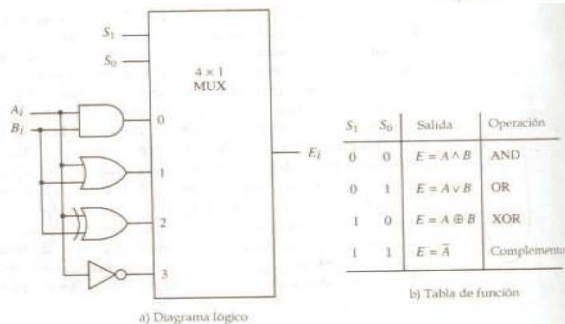


Microoperaciones Lógicas

- Especifican operaciones binarias para arreglos de bits almacenados en registros. Estas operaciones consideran cada bit de registro en forma separada y lo tratan con variables binarias.

- En otras palabras una Microoperación lógica considera a cada bit del registro en forma separada.

- Son muy útiles para la manipulación de bits de
- datos binarios y para tomar
- decisiones lógicas



Microoperaciones de Corrimiento

- Se utilizan para transferencia serial de datos

- También se utilizan junto con operaciones aritméticas y lógicas y de procesamiento de datos.

- Existen 3 tipos de corrimiento:

Corrimiento Lógico: $R \leftarrow Shl R(izq)$ $R \leftarrow Shr R(der)$

Corrimiento Circular: $R \leftarrow Cil R$ $R \leftarrow Cir R$

Corrimiento Aritmético: $R \leftarrow Sha R$ $R \leftarrow Shra R$

Unidad 4: Organización y diseño básico de computadoras

• Códigos de Instrucción

Definición

Es un conjunto de bits que le dice a la computadora como ejecutar una operación específica.

Por lo general se divide en **partes** y cada una tiene una interpretación propia. **La parte más básica** de un código de instrucción es su parte *de operación*.

El código de operación de una instrucción **es un grupo de bits que definen operaciones como sumar, restar, multiplicar, desplazar y complementar**.

Un **código de operación** se denomina **Macrooperación**, porque especifica un conjunto de microoperaciones.

Organización de un programa almacenado

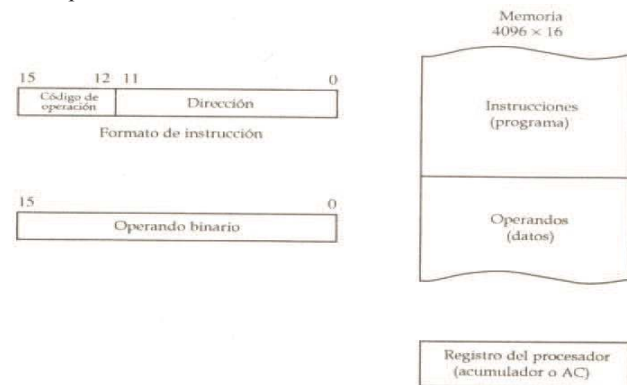
- La manera más simple de organizar una computadora es tener un registro de procesador y un formato de código de instrucción con dos partes:

- * La primera especifica **la operación** que se va a ejecutar ($n \text{ bits} - 2^n \text{ op}$)
- * La segunda especifica **una dirección** (registro o palabra de memoria donde se encontrará el operando)

- **Las instrucciones están almacenadas en una sección de la memoria y los datos en otro.**

- Las computadoras que tienen un registro de procesador único, por lo general lo nombran **Acumulador** y lo etiquetan como **AC**.

- La operación se ejecuta con el operando de la memoria y el contenido del AC. (Si una operación de un código de instrucción no necesita un operando de a memoria, puede usarse el resto de los bits de la instrucción para otros propósitos. Ej.: limpiar, complementar, incrementar AC).



Direccionamiento

- **Operando Inmediato (del bit 0 al 11 es un op):** cuando la 2 parte de un cód de instrucción especifica un operando (bit de direc como operando real)

- **Direccionamiento Directo (I=0):** la dirección especifica un operador inmediato (se hace una referencia a memoria).

- **Direccionamiento Indirecto (I=1):** los bit de la 2da parte de la instrucción representan la **dirección de una palabra de memoria**, en el cual se encuentra la direc del operando. Se necesita dos referencias a memoria para buscar un operador.

- **Dirección Efectiva:** es la dirección real del operando

• Registros de Computadora

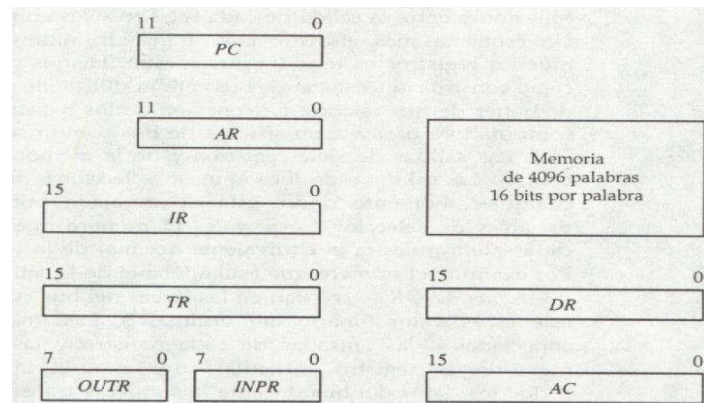
- Por lo general las instrucciones de computadora se almacenan en posiciones de memoria consecutivas y se ejecutan de manera secuencial, una a la vez.

- El control lee las instrucciones de una dirección específica de la memoria y la ejecuta.

- Después continúa leyendo la siguiente y así sucesivamente.

Lista de registros para una computadora básica:

Símbolo	Nombre	# Bits	Función
DR	(Registro de datos)	16	Contiene el operando en la memoria
AR	(Registro de dirección)	12	Contiene la dirección para la memoria
AC	(Acumulador)	16	Contiene el registro del procesador
IR	(Registro de instrucción)	16	Contiene el código de instrucción
PC	(Contador de programa)	12	Contiene el código de la siguiente instrucción
TR	(Registro temporal)	16	Contiene datos temporales
INPR	(Registro de entrada)	8	Contiene el carácter de entrada
OUPR	(Registro de salida)	8	Contiene el carácter de salida.



- También es necesario proporcionar un registro en la unidad de control para almacenar el código de instrucción después de que se lee de la memoria. La computadora necesita también los registros del procesador para manipular datos y un registro para contener una dirección de memoria.

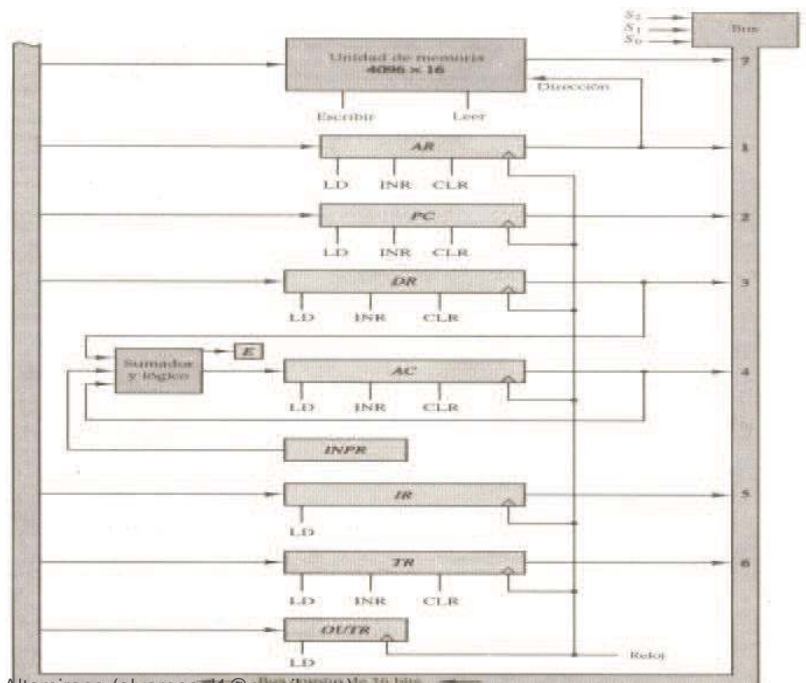
- Para una unidad de memoria que tiene una capacidad de 4096 palabras y cada palabra contiene 16 bits. Se necesitan :
 12 bits para especificar la dirección del operando (por cada palabra)
 3 bits de operación de la instrucción
 1 bit para especificar un direccionamiento (directo o indirecto)

Bus del Sistema

- **La computadora básica contiene 8 registros, una unidad de memoria y una unidad de control.** Deben proporcionarse trayectorias para transferir información de un registro a otro y entre la memoria y el registro. Un esquema eficiente para transferir información en un sistema con muchos registros es usar un bus común.

Características

Salidas de 7 registros conectadas al Bus Común (nro de referencia es el equivalente decimal de la selección bin)
 Selección de a través de S0, S1 y S2



• Instrucciones de Computadora

Formato de Código de instrucción

Cada formato tiene 16 bits. La parte del código de operación de la instrucción contiene tres bits y el significado de los 13 bits restantes depende del código de operación que se encuentre.

La computadora tiene tres formatos de código de instrucción:

- ***Instrucción de referencia a memoria:** utiliza 12 bits para especificar una dirección y un bit para especificar el direccionamiento (0 directa y 1 indirecta)
- ***Instrucción de referencia a registro:** se reconocen mediante el código 111 con un 0 en el extremo izquierdo de la instrucción. No se necesita un operando ya que los otros 12 bits se utilizan para especificar la operación o prueba que se va a ejecutar.
- ***Instrucción de entrada-salida:** no necesitan una referencia a memoria y se reconocen con el código de operación 111 con 1 en el bit del extremo izquierdo de la operación y los 12 bits restantes se utilizan para especificar el tipo de operación de E/S o la prueba que se va a ejecutar.

Conjunto de Instrucciones

Una computadora debe tener un conjunto mínimo de instrucciones para que el usuario pueda construir programas de lenguaje de computadora. El conjunto de instrucciones está completo si la computadora incluye:

- Instrucciones aritméticas, lógicas y de corrimiento.
- Instrucciones para mover información hacia y desde la memoria y los registros del procesador.
- Instrucciones de control de programa.
- Instrucciones de entrada y salida.

• Temporización y Control

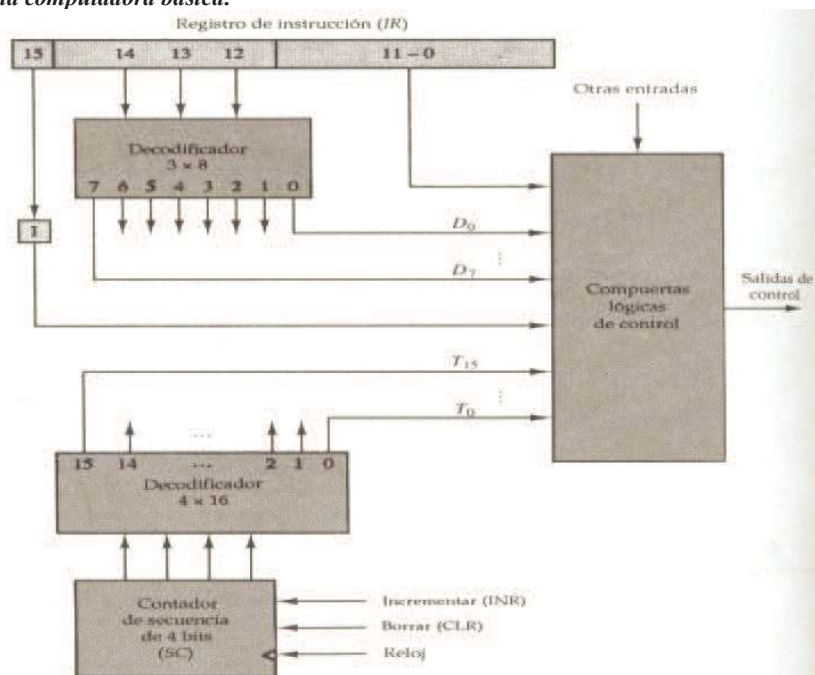
Temporización

Todo reloj está manejado por un generador de reloj maestro

Tipos de organización de control

- **Control por cableado:** se logra mediante compuertas, F-F's, decodificadores.
Ventaja: modo de reparación rápido.
Desventaja: para hacer cambios, se cambia la instalación del cableado.
- **Control Microprogramado:** la información de control está almacenada en una memoria de control.
Ventaja: los cambios se hacen en la memoria de control.
Desventaja: lenta.

Unidad de control de una computadora básica:



• Ciclo de Instrucción

- Un programa residente en la unidad de memoria de la computadora está formado por una secuencia de instrucciones. El programa se ejecuta en la computadora recorriendo un ciclo para cada instrucción. A su vez cada ciclo de instrucción se divide en una secuencia de subciclos o fases. Cada ciclo de instrucción consiste en las siguientes fases:

- 1- **Buscar una instrucción de la memoria.** (PC tiene las direcciones de las instrucciones secuenciales)
- 2- **Decodificar la instrucción.** (la unidad de control determina el tipo de instrucción -de referencia a memoria, de referencia a registros o de I/O- que se acaba de leer después de la decodificación)
- 3- **Leer la dirección efectiva de la memoria si la instrucción tiene una dirección indirecta.**
- 4- **Ejecutar la instrucción.**

- Cuando se termina el paso 4, el control regresa al paso 1 para buscar, decodificar y ejecutar la siguiente instrucción. Este proceso continua en forma indefinida a menos que se encuentre una instrucción de alto (HALT).

- Las microoperaciones para las fases de búsqueda y de codificación pueden especificarse mediante los siguientes enunciados:

T0: AR

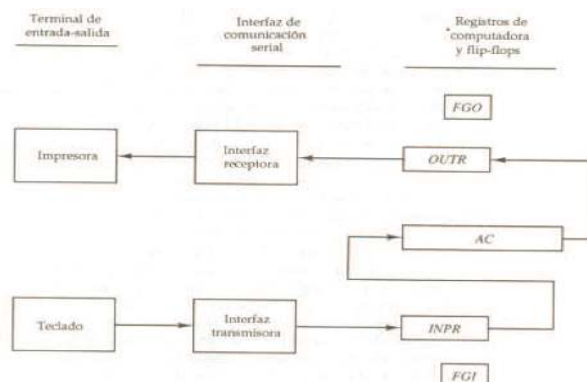
- Para T0 la dirección de PC se transfiere a AR.
- Para T1 la palabra de memoria leer dada por la dirección AR, se transfiere al registro de instrucciones, al mismo tiempo PC se incrementa en 1 a fin de reparar la dirección de la siguiente instrucción.
- Para T2 se codifica el código de instrucción se transfiere a AR.

• Instrucciones de Referencia a Memoria

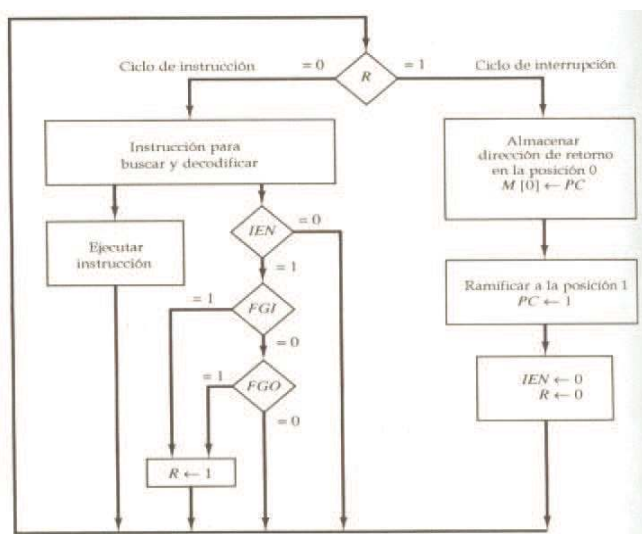
- **LDA (Cargar a AC):** Esta instrucción transfiere la palabra de memoria específica por la dirección efectiva de AC.
- **STA (Almacenar AC):** Esta instrucción almacena el contenido de AC en la palabra de memoria especificada por la dirección efectiva.
- **BUN (Brinco incondicional):** Esta instrucción transfiere el control del programa a la instrucción específica por cada *dirección efectiva*. (Ejecuta la función llamada *retorno de la subrutina*). Permite que el programador especifique una instrucción fuera de secuencia y se dice que el programa se brinca de manera incondicional.
- **BSA (Brincar y guardar la dirección de retorno):** Esta instrucción es útil para brincar hacia una porción del programa llamada subrutina o procedimiento (Esta instrucción ejecuta la función que se llama *llamada a subrutina*).
- **ISZ (Incrementa y brinca si es cero):** Incrementa la palabra especificada por la dirección efectiva y, si el valor incrementado es 0, PC se incrementa en 1 para saltar a la siguiente instrucción en el programa.

• Entrada/Salida e Interrupción

- Una computadora no puede tener un propósito útil a menos que se comunique con un ambiente externo. Las instrucciones y los datos almacenados en la memoria deben provenir de algún dispositivo de entrada. Los resultados deben transmitirse al usuario a través de algún dispositivo de salida. Ej.: Una unidad terminal con un teclado y una impresora.
- 2 F-F's de control: $\rightarrow FGI = 1$ (en este caso no puede cambiar el contenido de INPR)
 $\rightarrow FGO = 0$ (no puede cambiar el contenido de OUTR)



• Interrupción del programa



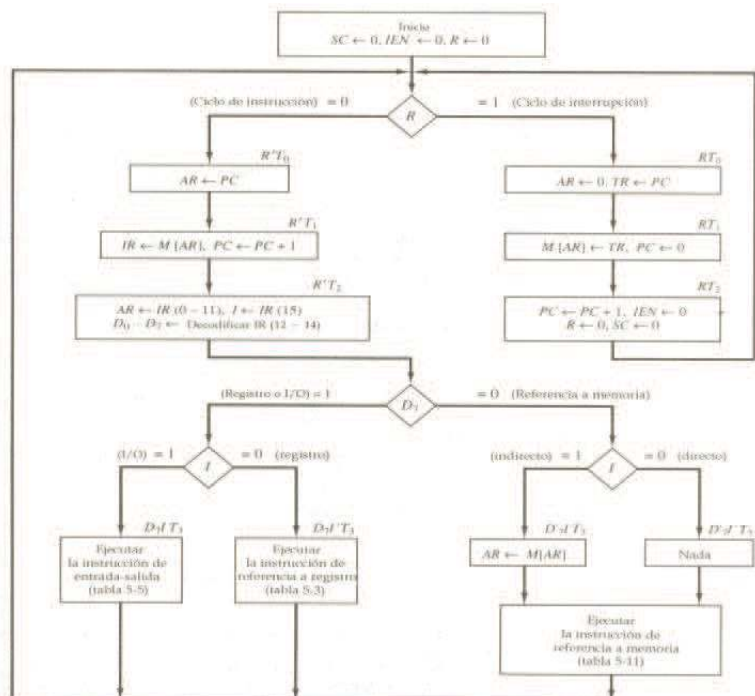
El Flip-Flop de habilitación de interrupción **IEN** puede habilitarse con dos instrucciones. Cuando **IEN** se borra a cero (con la instrucción **IOF**) las banderas no pueden interrumpir a la computadora. Cuando **IEN** se activa en 1 (con la instrucción **ION**) la computadora esta vez si puede ser interrumpida. Estas dos instrucciones le proporcionan al programador la opción de decidir si utiliza o no la opción de interrupción.

• Ciclo de interrupción:

El ciclo de interrupción se inicia después de la última fase de ejecución si el flip-flop R de interrupción es igual a 1. Este flip-flop se activa en 1 si IEN y si FGI o FGO son iguales a 1. Esto puede suceder con cualquier transición de reloj excepto cuando las señales de temporización T0, T1, T2 están activas. La condición para activar el flip-flop R en 1 pueden expresarse mediante el siguiente enunciado de transferencia de registro: $T_0'T_1'T_2'(IEN)(FGI + FGO): R \leftarrow 1$.

• Descripción Completa de una Computadora

En la figura se muestra el ultimo diagrama de flujo del ciclo de instrucción, incluyendo un ciclo de interrupción para la computadora basica.



• Diseño de una Computadora Básica

La computadora básica consta de los siguientes componentes de hardware:

- Unidad de Memoria 4096 x 16
- 9 Registros (**AR, PC, DR, AC, IR, TR, OUTR, INPR y SC**)
- 7 F-F's (**I, S, E, R, IEN, FGI, FGO**)
- 2 Decodificadores (3x8 y 4x 16) p/ Unid. de Ctrl.
- Bus Común de 16 bits
- Compuertas Lógicas de Control
- Circuito Sumador y Lógico conectado a la entrada de AC

• Diseño de un acumulador lógico

- **Memoria de Control**

- Palabra de control*

- ### Unidad de control microprogramada

- Microinstrucción (p/c palabra en la memoria de ctrl)*

- ### *Microprograma*

- | | | |
|-----------------|--------------------|--------------------|
| Palabra de Ctrl | → Microinstrucción | → Microoperaciones |
| | { | |
| | Microprograma | |

Memoria de Control

- ## Secuenciador de Microprograma

- ```

graph LR
 Entrada[Entrada externa] --> Generador[Generador de la dirección siguiente (secuenciador)]
 Generador --> RegistroControl[Registro de direccionamiento de control]
 RegistroControl --> Memoria[Memoria de control (ROM)]
 Memoria --> RegistroDatos[Registro de datos de control]
 RegistroDatos --> Salida[Palabra de control]
 RegistroDatos -- Información de la dirección siguiente --> Generador

```

**Figura 7-1** Organización de control microprogramada.

- **Secuencia de dirección**

## Rutina

- Es un grupo de microinstrucciones almacenadas en la memoria de control.

## Mapeo

- Es un proceso que transforma el código de instrucción en dirección de memoria de control

La transformación de bits del código de instrucción a una dirección en la memoria de control donde se localiza la rutina se denomina proceso de *mapeo*. Este proceso transforma el código de instrucción en una dirección de memoria de control.

En resumen, las posibilidades de secuencia de dirección que se requieren en la memoria de control son:

1. Incrementar el registro de direccionamiento de control
2. Transferencia de control del programa condicional o incondicional, dependiendo de las condiciones de los bits de estado
3. Un proceso de mapeo de los bits de la instrucción a una dirección para una memoria de control
4. Una operación para llamar y regresar la solicitud de una subrutina y retorno

El incrementador incrementa el contenido del registro de direccionamiento de control en uno, para seleccionar la microinstrucción siguiente en secuencia.

## Transferencia condicional

- **Depende de la condición especificada por bits especiales (acarreo de sumador – bit de signo – bit de modo).**
- Las condiciones de estado son bits especiales del sistema que proporcionan información de parámetros, como el acarreo de un sumador, el bit de signo, de un número, etc. La información en estos bits puede probarse y se pueden iniciar acciones con base en su condición si su valor es 1 o 0.

## Subrutinas

- Son programas que utilizan otras rutinas para realizar una tarea particular.
- Similar a la Modularidad en programación → *objetivo: evitar repetir microcodigo.*

## Memoria de Control

- Puede ser: → RAM → ventaja: cambios fáciles, sin recurrir a procedimientos de fabricación  
→ ROM → ventaja: barato, rápido, previene que el usuario cambie la arq del sistema.

- **Diseño de la Unidad de Control**

Los bits de la microinstrucción por lo general se dividen en campos y cada campo define una función separada y distinta. *Hay tes decodificadores, porque cada campo requiere un decodificador para producir las señales de control correspondientes.* Este método reduce el tamaño de los bits de microinstrucción pero necesita hardware adicional externo a la memoria de control.

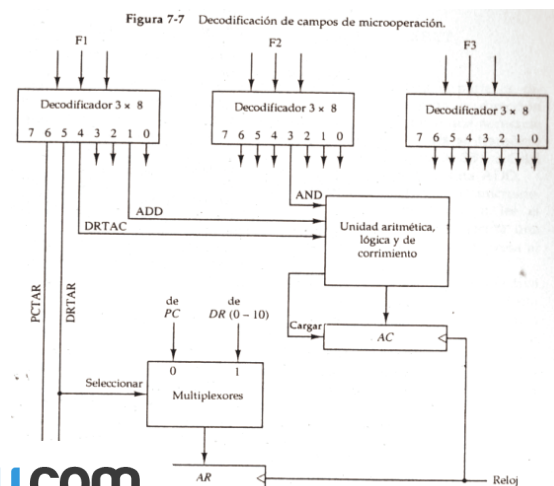
Los 9 bits del campo de la microoperación se dividen en 3 campos de 3 bits cada uno. La salida de la memoria de control de cada

microoperaciones. *Ver figura*

This document is available free of charge on

StuDocu.com

Descargado por Alvaro Altamirano (alvaroesal1@gmail.com)

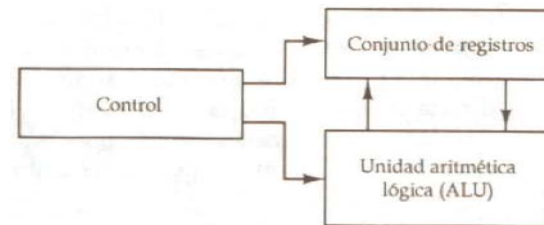


## • **Introducción**

### **Componentes principales del CPU**

- La CPU esta formada de tres partes principales:

- \* El **conjunto de registro**: almacena datos intermedios que se usan durante la ejecución de las instrucciones.
- \* La **unidad aritmética-lógica (ALU)**: lleva a cabo las microoperaciones requeridas para ejecutar las instrucciones.
- \* La **unidad de control**: supervisa la transferencia de información entre los registros e instruye a la ALU sobre cual operación ejecutar.



- En este capitulo se describe la organización y arquitectura de la CPU con énfasis en el punto de vista del usuario de la computadora. De manera breve se describe como los registros de comunican con la ALU mediante buses y explica la operación de la pila (Stack) de memoria y las instrucciones típicas incorporadas por lo regular en las computadoras.

## • **Organización General de los Registros**

- Hacer referencia a localidades de memoria para aplicaciones representa una respetable inversión de tiempo porque el acceso a memoria es la operación que consume más tiempo en una computadora.
- Es más conveniente y eficiente almacenar estos valores intermedios en **registros de procesador**. Cuando se incluye una gran cantidad de registros en la CPU es más eficiente conectarlos mediante un canal de sistema común.
- **Los Registros** se comunican uno con el otro no solo para transferencia directa de datos, sino también mientras ejecutan diversas microoperaciones.
- **La Unidad de Control** que opera el canal del sistema de la CPU elige el flujo de información a través de los registros y la ALU al seleccionar los diversos componentes del sistema.

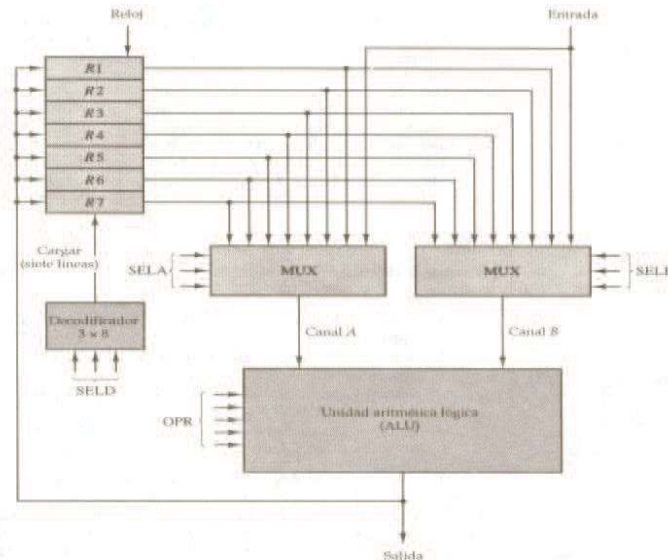
### **Palabra de Control**

Existen 14 entradas de selección binaria en la unidad y su valor combinado especifica una **Palabra de Control**. (Ej de ALU común – Ver pag 258)

## • **Organización de una Pila**

### **Pila de Registro**

- Una Pila de Registro es un dispositivo de almacenamiento.
- La pila es esencialmente una unidad de memoria que tiene:
  - \* **1 Apuntador de Pila (SP)**: Registro que apunta la parte superior de la pila (dirección de SP = palabra que esta en la cima de la Pila).
  - \* **2 F-F's**: (Lleno - Vacío)



a) Diagrama de bloque



b) Palabra de control

### **Notación Polaca Inversa**

- Una organización de pila es muy eficiente para evaluar expresiones aritméticas. Para evaluar expresiones aritméticas en notación interna fija es necesario analizar la expresión en todas sus partes para determinar cuál operación será la siguiente en ejecutarse. Las expresiones aritméticas pueden representarse en **notación fija previa** o **notación polaca**.
- En otras palabras, es un método aritmético que representa las operaciones de un modo mas fácil para la computadora.
- Esta notación usa Pila de Registros, que logra mayor eficiencia de evaluación de expresiones algebraicas
- Ej.  $A*B+C*D \rightarrow$  en notación polaca inversa es:  $AB*CD*+$

## • **Formatos de las Instrucciones**

- Por lo general, una computadora tendrá varios formatos de códigos de instrucciones. Una función de la unidad de control dentro de la CPU es interpretar cada código de instrucción y proporcionar las funciones de control necesarias para procesar la instrucción.
- **Normalmente el formato de una instrucción** se muestra en una caja rectangular. Los bits de la instrucción se dividen en grupos llamados campos. Los campos más comunes son:
  - \* **Código de operación**: especifica la operación que se va a ejecutar.
  - \* **Dirección**: representa una dirección de memoria o un registro de procesador.
  - \* **Modo**: especifica la manera en que se determina el operando o la dirección efectiva.

### **Instrucciones de tres direcciones:**

- Las computadoras con formato de instrucción de tres direcciones pueden utilizar cada campo de dirección para especificar un registro de procesador o un operando de memoria.
  - \* **Ventaja**: produce a la brevedad programa cuando evalúa expresiones aritméticas
  - \* **Desventaja**: las instrucciones de código binario requieren demasiados bits para especificar tres direcciones.

### **Instrucciones de dos direcciones**

Son las más comunes en las computadoras comerciales. En ellas, también cada campo de dirección puede especificar un registro de procesador o una palabra de memoria.

### **Instrucciones de una dirección**

Utilizan un registro acumulador (AC) implícito para toda la manipulación de datos. Para la manipulación y división se necesita un segundo registro.

### **Instrucción de cero direcciones**

Una computadora organizada con una pila no utiliza un campo de dirección para las instrucciones ADD y MUL. sin embargo, las instrucciones PUSH y POP necesitan un campo de dirección para especificar el operando que comunica con la pila.

### **Instrucciones RISC**

El conjunto de instrucciones de un procesador RISC típico esta limitado a usar las instrucciones cargar y almacenar cuando hay comunicación entre la memoria y la CPU sin transferirse a la memoria.



## • Modos de Direccionamiento

*La manera en que eligen los operandos durante la ejecución del programa depende del modo de direccionamiento de la instrucción.* El modo de direccionamiento especifica una regla para interpretar o modificar el campo de dirección de la instrucción antes de que se haga la referencia real al operando. La disponibilidad de modos de direccionamiento proporciona al programador la flexibilidad para escribir programas más eficientes. (Es imperativo recordar el ciclo de operación básico). La instrucción puede tener más de un campo de dirección, y cada campo de dirección puede estar asociado con su propio modo de direccionamiento particular. Hay dos modos que no necesitan el campo de dirección: los modos implícito e inmediato.

- **Modo implícito:** se especifican los operandos en forma implícita en la definición de la instrucción (instrucciones de dirección cero en una computadora organizada con pila).
- **Modo inmediato:** se especifica el operando en la instrucción misma. Tiene un campo de operando en lugar de un campo de dirección.

Recordar que el campo de dirección de una instrucción puede especificar una palabra de memoria o un registro de procesador. La unidad de control en la CPU utiliza el campo de dirección para obtener el operando de la memoria. La **dirección efectiva** se define como la dirección de memoria obtenida del cálculo, fijado mediante el modo de direccionamiento proporcionado. Es la dirección en que el control se transfiere en respuesta a una dirección de tipo brinco o salto. Cuando especifica un registro se dice que la instrucción está en modo de registro.

- **Modo de registro:** los operandos están en registros que residen dentro de la CPU.
- **Modo indirecto por registro:** la instrucción especifica un registro en la CPU cuyo contenido proporciona la dirección del operando en la memoria. El registro seleccionado contiene la dirección del operando en lugar del operando mismo.
- **Modo de autoincremento o autodecremento:** similar al modo de registro indirecto, excepto en que el registro se incrementa o decrementa después (a antes) de que su valor se use para acceder la memoria.
- **Modo de direccionamiento directo:** la dirección efectiva es igual a la parte de dirección de la instrucción.
- **Modo de direccionamiento indirecto:** el campo de dirección de la instrucción proporciona la dirección en la que se almacena la dirección efectiva en la memoria. *El control utiliza su parte de dirección para acceder la memoria una vez más con el fin de leer la dirección efectiva.*

Unos cuantos modos de direccionamiento requieren que el campo de dirección de la instrucción se sume al contenido de un registro específico en la CPU. En estos modos, la dirección efectiva se obtiene del cálculo siguiente:

**Dirección efectiva** = *parte de dirección de la instrucción + el contenido de registro del CPU*

- **Modo de direccionamiento relativo:** el contenido del contador de programa se suma a la parte de dirección de la instrucción para obtener la dirección efectiva, cuya posición en la memoria es relativa a la dirección de la siguiente instrucción.
- **Modo de direccionamiento indexado:** el contenido de un registro índice se suma a la parte de dirección de la instrucción para obtener la dirección efectiva. Cada operando del arreglo se almacena en la memoria en relación con la dirección inicial.
- **Modo de direccionamiento de registro base:** el contenido de registro base se suma a la parte de dirección de la instrucción para obtener la dirección efectiva. Se considera que un registro índice contiene un número de índice que se relaciona con la parte de dirección de la instrucción. Se considera que un registro base contiene una dirección base y que el campo de dirección de la instrucción proporciona un desplazamiento en relación con esta dirección base.

## • Transferencia y Manipulación de Datos

- La mayor parte de las instrucciones de computadora pueden clasificarse en tres categorías: **instrucciones de transferencia de datos, instrucciones de manipulación de datos e instrucciones de control de programa.**

## • Computadoras de Conjunto de Instrucciones Reducido

Un aspecto importante de la arquitectura de computadoras es el diseño de conjunto de instrucciones para el procesador. El conjunto de instrucciones elegido para una computadora particular determina la manera en el que se construyen los programas de lenguaje de máquina. Conforme la circuitería digital se hizo más barata con la aparición de los circuitos integrados, las instrucciones de computadora tendieron a aumentar, tanto en cantidad como en complejidad. Una computadora con gran cantidad de instrucciones se clasifica como una **computadora con conjunto de instrucciones complejo (CISC: Complex Instruction Set Computer)**. Muchos diseñadores de computadora recomendaron que las máquinas utilizaran menos instrucciones con fórmulas sencillas para que pusieran ejecutarse con mucha mayor rapidez dentro de la CPU, sin tener que utilizar la memoria con tanta frecuencia. Este tipo de computadoras se clasifica como **computadoras de conjunto de instrucciones reducido (RISC: Reduced Instruction Set Computer)**.

### Características CISC

Una razón para la tendencia a proporcionar conjuntos de instrucciones complejos es el deseo de simplificar la compilación y mejorar el desempeño general de la computadora. La tarea se simplifica si existen instrucciones de máquina que integren los enunciados de forma directa. El propósito esencial de una arquitectura CISC es intentar proporcionar una única instrucción de máquina para cada enunciado que esté escrita en un lenguaje de alto nivel. Otra característica de las CISC es la incorporación de formatos de instrucciones de tamaño variable. Para guardar estos formatos en una palabra de memoria de longitud fija se necesitan circuitos especiales de decodificación que cuenten los bytes dentro de la palabra y separen las instrucciones de acuerdo con la longitud de sus bytes. Las instrucciones proporcionan la manipulación directa de los operandos que residen en la memoria. En resumen, las principales características de la arquitectura CISC:

- Una gran cantidad de instrucciones (de 100 a 250).
- Algunas instrucciones que ejecuten tareas especializadas y que no se usen con frecuencia.
- Una gran cantidad de modos de direccionamiento (de 5 a 20 modos diferentes).
- Formatos de instrucciones de extensión variable.
- Instrucciones que manipulen operandos en la memoria.

### Características RISC

El concepto de la arquitectura RISC significa un intento para reducir el tiempo de ejecución al simplificar el conjunto de instrucciones de la computadora. Sus principales características son:

- Relativamente pocas instrucciones.
- Relativamente pocos modos de direccionamiento.
- El acceso a memoria limitado a instrucciones de carga y almacenamiento.
- Todas las operaciones realizadas dentro de los registros de la CPU.
- Formatos de instrucciones decodificados con facilidad, de longitud fija.
- Ejecución del ciclo de instrucción única.
- Control por circuitería en lugar de microprogramado.

Todos los cálculos se hacen entre los datos almacenados en los registros del procesador. Los resultados se transfieren a la memoria mediante instrucciones almacenar. Casi todas las instrucciones tienen registros de direccionamiento simple. Un formato de instrucción resulta fácil de decodificar. También puede hacerse más sencilla la lógica de control. Para operaciones más rápidas es preferible un control de circuitería sobre uno microprogramado.

*Una característica de los procesadores RISC es su capacidad para ejecutar una instrucción por ciclo de reloj. Esto se logra al hacer simultáneamente las fases de recuperación, decodificación y ejecución de dos o tres instrucciones, usando un procedimiento denominado paralelismo. Una instrucción cargar o almacenar puede requerir dos ciclos de reloj porque el acceso a memoria toma más tiempo que las operaciones de registro.*

Otras características de la arquitectura RISC son:

- Una cantidad de registros en el procesador relativamente grande.
- Uso de ventanas de registros traslapados (superpuestos) para acelerar la llamada y retorno de procedimientos.
- Paralelismo de instrucciones eficiente.
- Soporte de compilador para traducción eficiente de programas de lenguaje de alto nivel a programas de lenguaje máquina.

*La ventaja del almacenamiento en registros, a diferencia del almacenamiento en memoria, es que los registros pueden transferir información a otros registros mucho más rápido que la transferencia de información hacia y desde la memoria. Se pueden minimizar las operaciones. Se puede minimizar las operaciones al conservar en registros los operandos a los que se accede con mayor frecuencia.*

### **Ventana de registros traslapados o superpuestos**

En los lenguajes de programación de alto nivel ocurren con frecuencia llamadas y retornos de procedimientos. Después de un retorno de procedimiento, el programa reestablece los valores de registro anteriores, transmite los resultados al programa solicitante y retorna de la subrutina. Guardar y reestablecer registros, y pasar parámetros y resultados son operaciones que consumen tiempo. *Una característica de algunos procesadores RISC es que utilizan ventanas de registros traslapados para ofrecer el paso de parámetros y evitar la necesidad de guardar y reestablecer valores de registros.* Cada solicitud de procedimiento produce la asignación de una nueva ventana, que consiste en un conjunto de registros del archivo de registros, para que la use el nuevo procedimiento. *Las ventanas para procedimientos adyacentes tienen registros superpuestos que comparten para proporcionar el paso de parámetros y resultados.* Los registros altos del procedimiento llamado se superponen a los registros bajos del procedimiento llamado y, por lo tanto, los parámetros se transfieren en forma automática del procedimiento que llama al llamado. Las ventanas tienen una organización circular, de manera que las ventanas de los extremos son adyacentes y por lo tanto también se traslapan. *(ver libro pág 305)*

En general, la organización de las ventanas de registros tienen las siguientes relaciones:

- *G = Cantidad de registros globales* (para todos los procedimientos).
- *L = Cantidad de registros locales* en cada ventana.
- *C = Cantidad de registros comunes a las dos ventanas.*
- *W = Cantidad de ventanas.*

### **En Resumen**

**Características RISC** (intento de reducir tiempo de ejecución, al especificar el conjunto de instrucciones)

1. Relativamente pocas instrucciones.
2. Relativamente pocos modos de direccionamiento.
3. Acceso a memoria limitado (carga y almacenamiento)
4. Operaciones realizadas en registros → n° de registros relativamente grande.
5. Formato de Instrucción de longitud fija
6. Control por circuitería.

**Características CISC** (posee un conjunto de instrucciones complejas, para simplificar la compilación y mejorar el desempeño gral de la maquina)

1. Gran cantidad de instrucciones (algunas ejecutan tareas especializadas, que no se usan con frecuencia)
2. Gran cantidad de modos de direccionamiento
3. Formatos de instrucción de extensión variable
4. Instrucciones que manipulan operándos en memoria.



### • Procesamiento Paralelo

#### **Definición**

Es un término que se usa para denotar un grupo de técnicas significativas que se usan para proporcionar tareas simultáneas de procesamiento de datos con el fin de aumentar la velocidad computacional de un sistema de computadora.

- **Objetivo:** Aumentar cantidad de procesamiento concurrente (tareas simultáneas), con el fin de disminuir el tiempo de ejecución (mejorar la eficiencia del equipo).

- **El procesamiento paralelo** se establece al distribuir los datos entre las unidades funcionales múltiples.

- **Desventaja:** Aumenta la cantidad de circuitería y por ende el costo del sistema (hoy en día son económicamente factibles)

#### **Arquitectura Paralela**

Es una técnica que descompone un proceso secuencial en “Suboperaciones” → c/u se ejecuta en un segmento dedicado (son concurrentes).

Es eficiente para aplicaciones que repiten la misma tarea muchas veces.

- **Línea:** es un conjunto de segmentos de procesamiento por el cual fluye información binaria.

(Los registros proporcionan aislamiento entre segmentos)

- **Tarea:** operación total ejecutada cuando se recorren todos los segmentos de la línea

- **Circuito Verificador:** Detecta instrucciones cuyos operadores fuentes son destino de instrucciones que se encuentran mas adelante.

La **arquitectura paralela o de líneas paralelas (pipe-line)**. Es una técnica en la que se descompone un proceso secuencial en suboperaciones, y cada subproceso se ejecuta en un segmento dedicado especial que opera en forma concurrente con los otros segmentos.

El resultado obtenido del cálculo en cada segmento se transfiere al siguiente segmento en la línea. El nombre línea implica un flujo de información. La simultaneidad de los cálculos es posible al asociar un registro con cada segmento en la línea. Se aplica un reloj a todos los registros después de que ha transcurrido un tiempo suficiente para ejecutar toda la actividad del segmento. Cada segmento tiene unos cuantos registros y circuitos combinatorios. Cada ciclo de reloj produce una nueva salida y mueve los datos un paso adelante en la línea. El reloj debe continuar hasta que se emerge la última salida de la línea. **Definimos una tarea como la operación total ejecutada cuando se recorren todos los segmentos en la línea. (ver libro pág 323 y 325)**

Existen dos áreas de diseño de computadoras en las que es aplicable la organización paralela: **una línea aritmética** y una **línea de instrucciones**.

### • Arquitectura Paralela RISC

- **La arquitectura RISC puede tener segmentos paralelos (al requerir un ciclo de reloj).**

- **Se basa en el compilador para minimizar los retrasos.**

Puede usarse la sencillez del conjunto de instrucciones para implantar una línea paralela de instrucciones con un número pequeño de operaciones y que cada una se ejecute en un ciclo de reloj. La decodificación de la operación puede ocurrir al mismo tiempo que la selección de registro. Como todos los operandos están en registros, no es necesario calcular una dirección efectiva o recuperar operandos de la memoria. La línea paralela de instrucciones puede implantarse con dos o tres segmentos. Un segmento recupera la instrucción de la memoria del programa y el otro segmento ejecuta la instrucción en la ALU. Puede usarse un tercer segmento para almacenar el resultado de la operación de la ALU en un registro destino. Para evitar conflictos, las máquinas RISC utilizan dos canales separados con dos memorias: una para almacenar las instrucciones y otra para almacenar los datos. **Una de las principales ventajas de la RISC es su capacidad para ejecutar instrucciones a una velocidad de una instrucción por ciclo de reloj.** Mediante su arquitectura paralela, el procesador logra el propósito de una ejecución de instrucción en un solo ciclo. Otra característica de la RISC es el soporte que proporciona el compilador que traduce el programa de lenguaje de alto nivel en un programa de lenguaje de máquina. En lugar de diseñar hardware para manejar las dificultades asociadas con conflictos de datos y pérdidas por transferencias de control, los procesadores se basan en la eficiencia del compilador para detectar y minimizar los retrasos que se encuentran con estos problemas.

**Ejemplo: línea paralela de instrucciones de tres segmentos**

El ciclo de instrucción puede dividirse en tres suboperaciones e implantarse en tres segmentos. *Recuperación de instrucción, operación de la ALU y ejecución de la instrucción.*

### • Procesamiento Vectorial

#### **Definición**

- **Usa vectores para evitar hacer gran cantidad de cálculos.**

- Existe un tipo de problema computacional que está más allá de la capacidad de una computadora convencional, se caracteriza por el hecho de que requiere una gran cantidad de cálculos que le tomaría a una computadora convencional días o semanas para terminar. **Las computadoras con posibilidades de procesamiento de vectores pueden ejecutar aplicaciones especializadas, ya que permiten realizar muchos cálculos.** Estas computadoras con procesamiento vectorial tienen muchas áreas de aplicación, como *predicciones climatológicas a largo plazo, exploraciones petrolíferas, análisis de datos sísmicos, diagnósticos médicos, simulaciones de aerodinámica y vuelo espacial, inteligencia artificial, mapeo de genes humanos, etc.* **Para cumplir con estas necesidades y un buen desempeño es necesario usar circuitos rápidos y confiables, aplicando técnicas de procesamiento paralelo y procesamiento vectorial.**

### • Arreglo de Procesador SIMD

- Un arreglo de procesador es un procesador que ejecuta cálculos sobre arreglos de datos grandes. El término hace referencia a dos tipos diferentes de procesadores: **arreglo de procesador conectado y arreglo de procesador SIMD.**

## Unidad 9: Hardware para aritmética de computadoras

### Introducción

Las instrucciones aritméticas en las computadoras digitales manipulan los datos para producir los resultados necesarios para la solución de problemas computacionales. *Estas instrucciones son responsables de la mayor parte de la actividad para el procesamiento de datos en una computadora.* Un procesador aritmético es la parte de una unidad de procesador que ejecuta operaciones aritméticas (suma, resta, multiplicación y división). *Una instrucción aritmética puede especificar datos binarios o decimales*, y en cada caso, los datos pueden estar en forma de punto fijo o punto flotante. Existen tres maneras de representar números binarios negativos de punto fijo: *magnitud con signo, complemento a 1 con signo o complemento a 2 con signo.*

En esta unidad se desarrollan los diferentes algoritmos aritméticos y se muestran los procedimientos para obtenerlos con hardware digital. Se considera la suma, resta multiplicación y división para los siguientes tipos de datos:

1. Datos binarios de punto fijo, en representación de magnitud con signo.
2. Datos binarios de punto fijo, en representación de complemento a dos con signo.
3. Datos binarios de punto flotante
4. Datos decimales en código binario (BCD)

### Suma y Resta

Existen 3 maneras de representar números binarios de punto fijo:

- Magnitud con signo
- Complemento a 1 con signo o
- Complemento a 2 con signo

| Paralelismo                                                                                                              | Serie    |
|--------------------------------------------------------------------------------------------------------------------------|----------|
| Operaciones muy rápidas.<br>Desperdicio de tiempo entre algunos segmentos.<br>Nunca llega a la máxima velocidad teórica. | No tanta |

### Hardware para suma y resta de magnitudes con signo

**Sobreflujo:** ocurre cuando se suman dos números de  $n$  bits c/u y la suma ocupa  $n+1$  dígitos.

### Unidad aritmética decimal

El usuario de una computadora prepara datos con número decimales u recibe resultados en forma decimal. *Para ejecutar operaciones aritméticas con datos decimales, es necesario convertir los números decimales de entrada en binarios, para ejecutar todos los cálculos con números binarios y convertir el resultado en decimal.* Cuando la aplicación solicita una gran cantidad de entradas y salidas y un número relativamente más pequeño de cálculos aritméticos es más conveniente hacer la aritmética interna de manera directa con los números decimales.

Una unidad aritmética decimal es una función digital que ejecuta microoperaciones decimales. Puede sumar o restar números decimales al formar el complemento a 9 o 10 del sustraendo. Una unidad aritmética decimal de una sola etapa consta de 9 variables binarias de entrada y 5 variables binarias de salida, porque se necesita un mínimo de 4 bits para representar cada dígito decimal codificado. Cada etapa debe tener 4 entradas para un dígito sumando, 4 entradas para el otro sumando y un acarreo de entrada ( $4+4+1 = 9$  variables binarias de entrada). Las salidas incluyen 4 terminales para el dígito de suma y uno para el acarreo de salida ( $4+1 = 5$  variables binarias de salida).

### SUMADOR BCD

Consideremos la suma aritmética de 2 dígitos decimales en BCD, junto con un posible acarreo de una etapa anterior. Como cada dígito de entrada no excede de 9, la suma de salida no puede ser mayor que  $9 + 9 + 1 = 19$ , el 1 en la suma es un acarreo de entrada. Supongamos que *se aplican los dígitos BCD a un sumador binario de 4 bits*. El sumador formará la suma de binario y producirá un resultado que variará entre de 0 a 19. Estos números binarios se listan en la tabla y se etiquetan mediante los símbolos **K**, **Z<sub>8</sub>**, **Z<sub>4</sub>**, **Z<sub>2</sub>**, **Z<sub>1</sub>**. **K** es el acarreo y los subíndices bajo la letra **Z** representan las ponderaciones 8, 4, 2, 1 que pueden asignarse a los 4 bits en el código BCD. La primera columna en la tabla lista las sumas binarias conforme aparecen en las salidas del sumador binario de 4 bits. La suma de salida de 2 números decimales debe representarse en BCD y debe aparecer en la forma que se lista en la segunda columna de la tabla. *El problema es encontrar una regla para convertir el número binario de la primera columna en la representación correcta de dígito BCD del número de la segunda columna. (ver libro pág 390-392)*

Al examinar el contenido de la tabla, es evidente que cuando la suma binaria es igual que o menos que 1001, el número BCD correspondiente es idéntico y, por lo tanto, no es necesaria la conversión. Cuando la suma binaria es mayor que 1001, obtenemos una representación BCD no válida. *La suma de binario 6 (0110) a la suma binaria la convierte en representación BCD correcta y también produce un acarreo de salida, conforme se requiere.*

- Un método para sumar números decimales en BCD sería emplear un sumador binario de 4 bits y ejecutar la operación aritmética de un dígito a la vez. El par de dígitos de orden inferior se suma primero para producir una suma binaria.
- Si el resultado es igual que o mayor que 1010, se corrige al sumar 0110 a la suma binaria.* Esta segunda operación producirá en forma automática un acarreo de salida para el siguiente par de dígitos significativos.

El siguiente par de dígitos de orden superior junto con el acarreo de entrada, se suma después para producir una suma binaria. Si el resultado es igual que o mayor que 1010, se corrige la suma a 0110. El procedimiento se repite hasta que se suman todos los dígitos decimales.

El circuito lógico que detecta la corrección necesaria puede derivarse de las entradas de tabla. Es obvio que *se necesita una corrección cuando la suma binaria tiene un acarreo de salida K = 1. La condición para una corrección y un acarreo de salida puede expresarse mediante la expresión booleana  $C = K + Z_8Z_4 + Z_8Z_2$  cuando  $C = 1$  es necesario sumar 0110 a la suma binaria y proporcionar un acarreo de salida para la siguiente etapa.*

Un sumador BCD es un circuito que suma dos dígitos BCD en paralelo y produce un dígito de suma también en BCD. Un sumador BCD debe incluir la lógica de corrección en su construcción interna. Para sumar 0110 a la suma binaria utilizamos un segundo sumador binario de 4 bits. Los dígitos decimales, junto con el acarreo de entrada se suman primero en el sumador binario de 4 bits superior, para producir una suma binaria. Cuando el acarreo de salida es igual a cero, no se suma nada a la suma binaria. Cuando es igual a 1, se suma el binario 0110 a la suma binaria mediante el sumador binario de 4 bits inferior. (Puede ignorarse el acarreo de salida generado por el sumador binario inferior, porque proporciona información ya disponible en la terminal de acarreo de salida).

Un sumador paralelo decimal que suma  $n$  dígitos decimales necesita  $n$  etapas de sumador BCD, con el acarreo de salida de una etapa conectado al acarreo de entrada de la siguiente etapa de orden superior.

### RESTA BCD

Una resta directa de dos números decimales requerirá un circuito sustractor diferente de alguna manera de un sumador binario. *Es más práctico ejecutar la resta al tomar al complemento a 9 o 10 del sustraendo y sumarlo al minuendo ( $A - B = A + B' + 1$ ).*

El complemento a 9 de un dígito decimal representado en BCD puede obtenerse al complementar los bits de la representación codificada del dígito, siempre y cuando se incluya una corrección. Existen dos métodos de corrección posibles:

1. *Se suma el binario 1010 (decimal 10) a cada dígito complementado y al acarreo descartado después de cada suma.*
2. *Se suma el número binario 0110 (decimal 6) antes de que se complemente el dígito.* Es el método más conveniente.

Una etapa de una unidad aritmética decimal que puede sumar o restar dos dígitos BCD consiste en un sumador BCD y un complementador a 9. El modo **M** controla la operación de la unidad. Con  $M = 0$  las salidas  $s$  forman la suma  $A + B$ . con  $M = 1$ , las salidas  $s$  forman la suma de  $A +$  el complemento a 9 de  $b$ . Para números con  $n$  dígitos decimales necesitamos  $n$  etapas como esa. El acarreo de salida  $c_{i+1}$  de una etapa debe conectarse al acarreo de entrada  $c_i$  de la siguiente etapa de orden superior. *La mejor manera de restar los dos números decimales es dejar  $m = 1$  y aplicar un 1 al acarreo de entrada  $c_i$  de la primera etapa. Las salidas formarán la suma de  $A +$  el complemento a 10 de  $b$ , lo cual es equivalente a una operación de resta si se descarta el acarreo de la última etapa. (ver libro pág 393)*

*Luego de que queda la resta representada  $A - B = A + B' + 1$ , esta expresión es una suma de  $A$  más el complemento a 10 de  $b$ . que se resolverá como una suma BCD, aplicando los pasos ya explicados.*

### Operaciones aritméticas decimales

Los algoritmos para las operaciones aritméticas con datos decimales son similares a los algoritmos para las operaciones correspondientes de datos binarios. Los números decimales en BCD se almacenan en los registros de la computadora en grupos de cuatro bits. Cada grupo de cuatro bits representa un dígito decimal y debe tomarse como una unidad cuando se ejecutan microoperaciones decimales. *Una barra en el símbolo de la letra del registro representa el complemento a 9 del número decimal almacenado en el registro. Al sumar 1 al complemento a 9 se produce el complemento a 10.*

### Sumador BCD

Es un circuito que suma dos dígitos BCD en paralelo y produce un dígito de suma también en paralelo. Debe incluir la lógica de corrección en su construcción interna.

## Unidad 10: Organización de Entrada-Salida

- **Dispositivos Periféricos**

Un subsistema de E/S proporciona un modo de comunicación entre el sistema central y el ambiente externo. Una computadora no tiene ninguna capacidad útil si sin la capacidad de recibir información de una fuente externa y de transmitir los resultados de manera comprensible. El medio más familiar de introducir información alfanumérica de manera directa es el teclado.

**Dispositivos Periféricos:** dispositivos conectados a la computadora. Estos dispositivos están diseñados para leer información hacia adentro o afuera de la unidad de memoria ante un comando de la CPU y se considera que son parte del sistema total de la computadora.

- **Interfase de Entrada y Salida**

**Interfase de entrada-salida**

Proporciona un método para transferir información entre dispositivos de almacenamiento interno y de E/S externas. Los periféricos conectados necesitan un enlace de comunicación especial para funcionar como una interfase con la unidad de procesamiento central. El propósito es resolver las diferencias entre la computadora central y cada periférico.

1. Los periféricos son dispositivos electromecánicos y electromagnéticos y su manera de operar es diferente al de la CPU y la memoria que son dispositivos electrónicos. Por lo tanto se requiere una conversión de valores de señal.
2. LA velocidad de transferencia de datos por lo general es menor que la velocidad de transferencia de la CPU, puede necesitarse un mecanismo de sincronización.
3. Los códigos de datos y los formatos en los periféricos son diferentes del formato de la palabra de la CPU y en la memoria.
4. Los modos de operación de los periféricos son diferentes uno de otros y cada uno debe estar controlado para no perturbar la operación de otros periféricos conectados a la CPU.

**Canal de E/S y módulos de interfase**

El canal de E/S, consta de de datos, líneas de dirección y líneas de control.

Cada dispositivo periférico tiene asociada una unidad de interfase.

Cada interfase decodifica la dirección y el control que se recibe del canal de E/S, y las interpreta para el periférico y proporciona señales para el controlador del periférico.

El canal de E/S del procesador se conecta a todas las interfaces del periférico.

Para comunicarse con un dispositivo particular, el procesador coloca una dirección de dispositivo en las líneas de direccionamiento.

Cada línea conectada al canal de E/S contiene un decodificador de dirección que monitorea las líneas de direccionamiento.

Al mismo tiempo que queda disponible la dirección en las líneas de direccionamiento, el procesador proporciona un código de función en las líneas de control.

- **Comando de E/S:** se denomina al Código de Función y, es en esencia, una instrucción que se ejecuta en la interfaz y esta conectada a la unidad periférica.

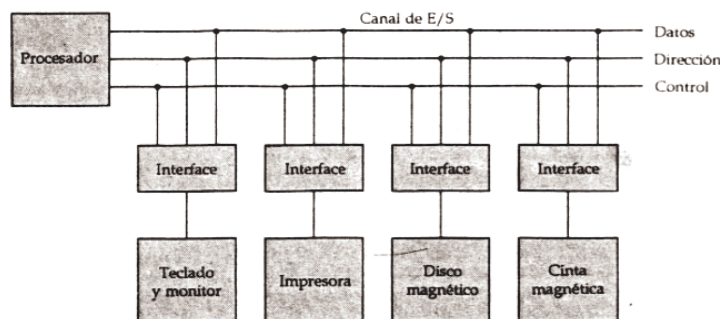
- **Comando de Control:** se emite para activar el periférico e informarle que hacer. Depende particularmente del periférico, este recibe su propia secuencia distinta de comandos de control.

- **Comando de Control de Estado:** se utiliza para probar diversas condiciones de estado en la interfaz y un periférico. Ej. Probar el estado de un periférico antes de que se inicie una transferencia.

- **Comando de Salida de Datos:** hace que la interfaz responda transfiriendo datos del canal a uno de sus registros, como por ejemplo, la computadora comienza a mover la cinta al emitir un comando de control. Después el procesador monitorea el estado de la cinta mediante un comando de estado.

- **Comando de entrada de datos:** es lo opuesto al de la salida de datos, la interfaz recibe datos del periférico y lo coloca en su registro intermedio. El procesador verifica si los datos están disponibles mediante un comando de estado y después envía un comando de entrada de datos. La interfase coloca los datos sobre las líneas de datos, donde el procesador los acepta.

Figura 11-1 Conexión de canal de E/S a dispositivos de entrada-salida.



- **Transferencia Asíncrona de Datos**

- **Modos de Transferencia**

- **Prioridad de Interrupción**

- **Acceso Directo a Memoria (DMA)**

- **Procesador de Entrada-Salida (IOP)**

- **Comunicación Serial**



## Unidad 11: Organización de la Memoria

### • Jerarquía de la Memoria

- La **unidad de memoria** es un componente esencial en cada computadora ya que sirve para almacenar programas y datos.

- **Memoria Principal:** unidad de memoria que se comunica directamente con la CPU, y contiene solo los programas y datos que necesita en ese momento el procesador

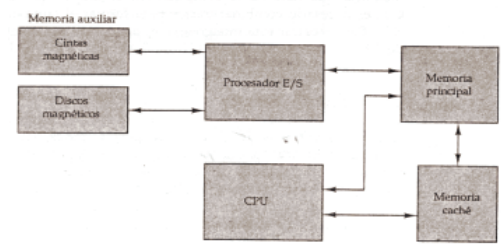
- **Memoria Auxiliar:** Se usa para almacenar programas del sistema, archivos y otra información de respaldo que es transferida a la memoria principal cuando se necesita.

- **Memoria Caché:** es una memoria especial de muy alta velocidad, utilizada para aumentar la capacidad de procesamiento, al poner disponible para la CPU los programas y datos actuales a una velocidad rápida.

- El propósito general de utilizar una jerarquía de memorias es obtener la velocidad de acceso promedio más alta posible, en tanto se minimiza el costo total de todo el sistema de memoria. El sistema de jerarquía de memoria consiste en todos los sistemas de almacenamiento que se emplean en un sistema de computadoras.

En la parte mas baja de la jerarquía están las **cintas magnéticas**, que son relativamente lentas y que se usan para almacenar archivos removibles. Después, están los **discos magnéticos** que se utilizan como almacenamiento de respaldos. La **memoria principal** ocupa una posición central al poder comunicarse directamente con la CPU y con los dispositivos de memoria auxiliar. Luego está la **memoria caché** que es un tipo muy especial de memoria de alta velocidad.

Figura 12-1 Jerarquía de memoria en un sistema de computadora.



### • Memoria Principal

La **memoria principal** es la unidad de almacenamiento central en un sistema de computadora. La tecnología que se utiliza para este tipo de memoria se basa en los circuitos integrados semiconductores.

**Memoria de Acceso Aleatorio (RAM)** → para almacenar la > parte de los procesos y datos que están sujetos a cambio... Es volátil

Se dispone de la RAM en circuitos integrados en dos modos de operación posibles, estático y dinámico:

- **RAM estática:** consiste esencialmente en flip-flops internos que almacenan información binaria, dicha información almacenada es valida mientras la unidad está encendida.

- **RAM dinámica:** almacena la información binaria en forma de cargas eléctricas que se aplican a capacitores. Unos transistores MOS son los capacitores en el CI. La carga almacenada y los capacitores tienden a perderse con el tiempo y los capacitores deben cargarse en forma periódica al refrescar la memoria dinámica.

Gran parte de la memoria principal, en una computadora de propósito general, está formada por CI de RAM, pero una parte de la memoria está formada por la ROM

### Memoria de Solo Lectura (ROM) no es volátil

También es de acceso aleatorio, se utiliza para almacenar programas que residen en forma permanente en la computadora y para tablas de constantes que no cambian de valor una vez que se termina la programación de la computadora.

Entre otras cosas, la parte ROM de la memoria se necesita para almacenar un:

- Cargador de inicialización: es un programa cuya función es iniciar la operación de la programación de la computadora cuando se enciende la unidad

- Programa de inicialización: se encarga de cargar una parte del sistema operativo del disco a la memoria principal y después se transfiere el control al sistema operativo, el cual prepara la computadora para su uso general.

### Mapa de Dirección de Memoria

El mapa de dirección de memoria es una tabla que especifica la dirección de memoria asignada a cada CI. Es una representación grafica del espacio de dirección asignado a cada CI en el sistema.

Ej. Un sistema de computadora necesita: 512 byte de RAM, 512 byte de ROM.

La configuración se muestra en la tabla, donde:

- **Componente:** especifica si se usa un CI de RAM o ROM
- **Dirección Hexadecimal:** asigna un rango de direcciones hexadecimales equivalentes para cada CI
- **Canal de direcciones:** la diferencia entre una dirección RAM y ROM se establece con otra línea del canal (Línea 10 para este proposito, cuando L10=0 la CPU selecciona una RAM – L10=1 Selecciona la ROM)

TABLA 12-1 Mapa de dirección de memoria para microcomputadora

| Componente | Dirección hexadecimal | Canal de direcciones |   |   |   |   |   |   |   |   |   |
|------------|-----------------------|----------------------|---|---|---|---|---|---|---|---|---|
|            |                       | 10                   | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| RAM 1      | 0000-007F             | 0                    | 0 | 0 | x | x | x | x | x | x | x |
| RAM 2      | 0080-00FF             | 0                    | 0 | 1 | x | x | x | x | x | x | x |
| RAM 3      | 0100-017F             | 0                    | 1 | 0 | x | x | x | x | x | x | x |
| RAM 4      | 0180-01FF             | 0                    | 1 | 1 | x | x | x | x | x | x | x |
| ROM        | 0200-03FF             | 1                    | x | x | x | x | x | x | x | x | x |

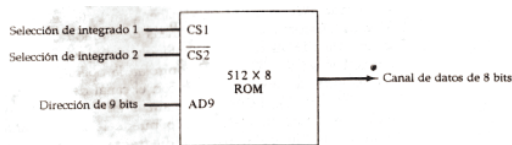
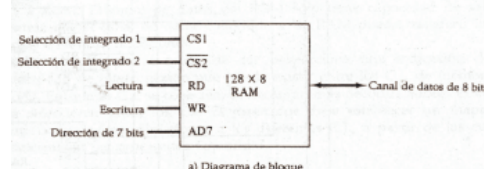


Figura 12-3 C. I. de ROM típico.

Figura 12-2 Integrado de RAM típico.



a) Diagrama de bloque

| CS1 | CS2 | RD | WR | Función de memoria | Estado de canal de datos |
|-----|-----|----|----|--------------------|--------------------------|
| 0   | 0   | x  | x  | Inhibición         | Alta impedancia          |
| 0   | 1   | x  | x  | Inhibición         | Alta impedancia          |
| 1   | 0   | 0  | 0  | Inhibición         | Alta impedancia          |
| 1   | 0   | 0  | 1  | Escritura          | Datos de entrada a RAM   |
| 1   | 0   | 1  | x  | Lectura            | Datos de salida de RAM   |
| 1   | 1   | x  | x  | Inhibición         | Alta impedancia          |

b) Tabla de funciones

- Los **CI de RAM** tienen 128 bytes y necesitan 7 líneas de direccionamiento
- El **CI de ROM** tiene 512 bytes y necesita 9 líneas de direccionamiento

Por lo tanto las x se asignan siempre a las líneas de canal de orden inferior: las líneas del 1 al 7 para la RAM y las líneas del 1 al 9 la ROM

En el ejemplo se escoge las líneas 8 y 9 para que representen cuatro combinaciones binarias diferentes.

### Conexión de la Memoria a la CPU

Los CI de RAM y ROM están conectados a la CPU por medio de los canales de datos y de dirección:

- **Canal de Dirección:** las líneas de orden inferior en dicho canal seleccionan el byte dentro de los CI y las otras líneas seleccionan un CI particular por medio de sus entradas de selección de integrado

- Ej. Es una configuración que proporciona una capacidad de memoria de 512 byte de RAM y 512 bytes de ROM

- cada RAM recibe los siete bits de orden inferior para seleccionar uno de los 128 byte posibles.
- El CI RAM particular seleccionado se determina de las líneas < 8 y 9 del canal de direcciones (a través de un DECO 2x4)
- Las salidas RD y WR del microprocesador se aplican a la entrada de cada CI RAM.
- La selección entre RAM y ROM se hace por medio de la línea 10 del canal (valor 0 para RAM – 1 para ROM)

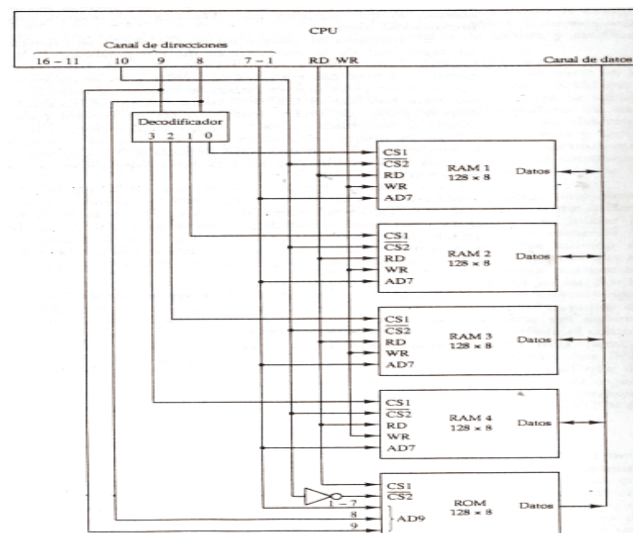
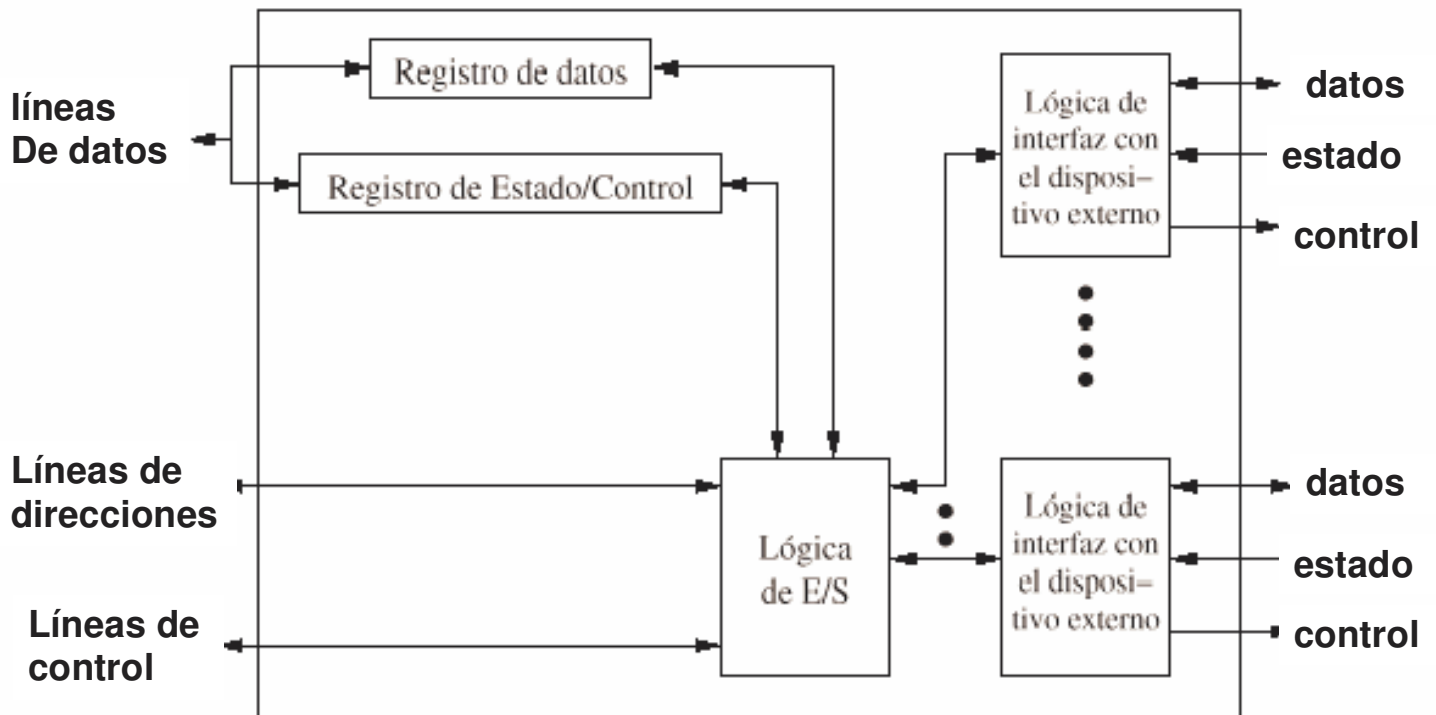


Figura 12-4 Conexión de memoria a la CPU.

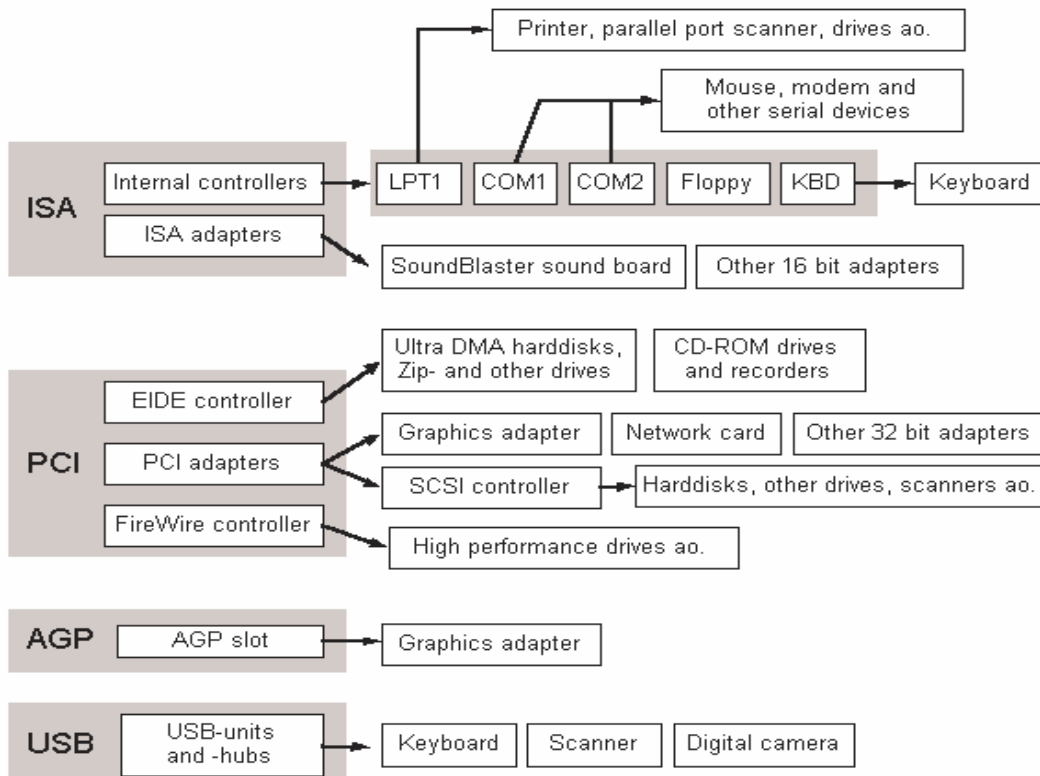
# Diagrama de bloques de un módulo de E/S

Interfaz con el bus del sistema


Interfaz con el (los) dispositivo externo





## BUSES EN LA PC




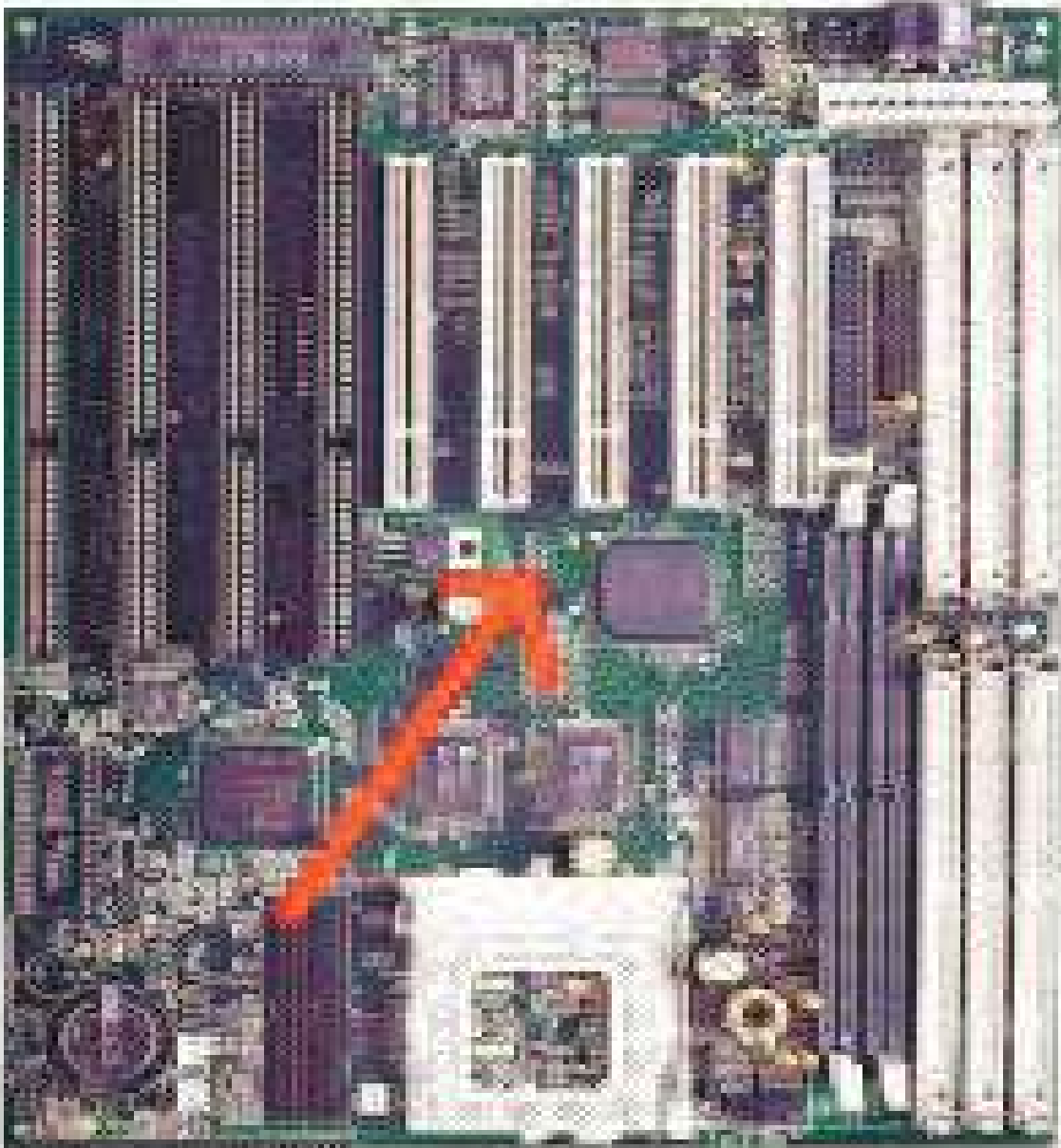


|                                                                                    |                                         |             |               |            |
|------------------------------------------------------------------------------------|-----------------------------------------|-------------|---------------|------------|
| ISA                                                                                | Industry Standard Architecture (AT) Bus |             |               |            |
|  |                                         |             |               |            |
| Pins                                                                               | Data Bus                                | Address Bus | Transfer Rate | Speed      |
| 98                                                                                 | 8/16 bits                               | 24 bits     | 5.5MB/s       | 6-8.33 MHz |

|                                                                                    |                                       |             |               |        |
|------------------------------------------------------------------------------------|---------------------------------------|-------------|---------------|--------|
| PCI                                                                                | Peripheral Component Interconnect Bus |             |               |        |
|  |                                       |             |               |        |
| Pins                                                                               | Data Bus                              | Address Bus | Transfer Rate | Speed  |
| 124                                                                                | 32 bits                               | 32 bits     | 132MB/s       | 33 MHz |
| 188                                                                                | 64 bits                               | 64 bits     | 264MB/s       | 33 MHz |

| AGP                                                                                 | Accelerated Graphics Port |             |               |        |
|-------------------------------------------------------------------------------------|---------------------------|-------------|---------------|--------|
|  |                           |             |               |        |
| Pins                                                                                | Data Bus                  | Address Bus | Transfer Rate | Speed  |
| 188                                                                                 | 32 bits                   | 32 bits     | 264MBs        | 66 MHz |

| FSB                                                                                  | Front Side (System Memory) BUS |             |               |         |
|--------------------------------------------------------------------------------------|--------------------------------|-------------|---------------|---------|
|  |                                |             |               |         |
| Pins                                                                                 | Data Bus                       | Address Bus | Transfer Rate | Speed   |
| 168                                                                                  | 64 bits                        | 64 bits     | 528 MBs       | 66 MHz  |
| 168                                                                                  | 64 bits                        | 64 bits     | 800 MBs       | 100 MHz |
| 168                                                                                  | 64 bits                        | 64 bits     | 1064 MBs      | 133 MHz |

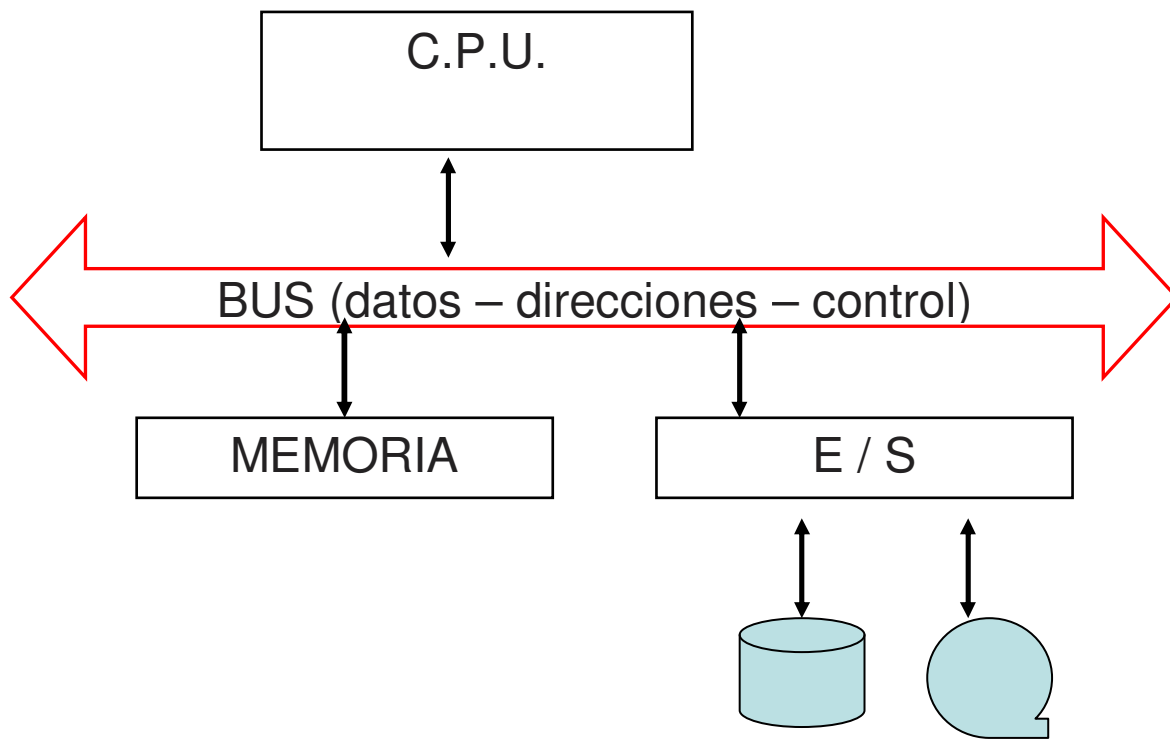


### 3 TIPOS DE CONFIGURACION

#### A) BUS UNICO

♣ Bajo costo

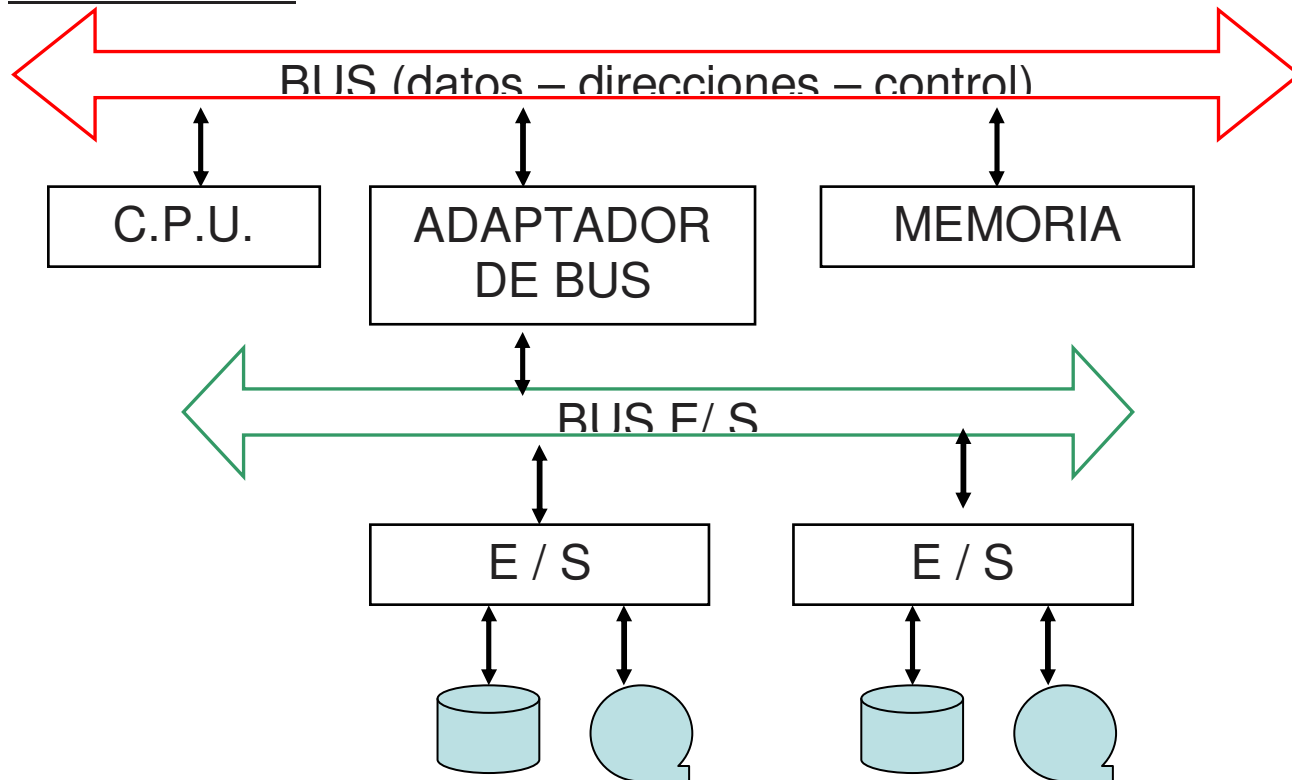
♣ Bajo Rendimiento



#### B) DOS BUSES

♣ Memoria – CPU

♣ Entrada / Salida



#### C) TRES BUSES

♣ El bus Memoria – CPU, solo para el trafico con memoria

♣ Diferentes buses se conectan con el bus de backplane

