Resumen de organización de computadoras

Organización de computadoras:

Los elementos principales de un computador son la unidad de procesamiento central **(CPU),** la memoria principal, el subsistema de **E/S** y algunos medios de interconexión de todos los componentes. La CPU por su parte consta de una unidad de control **(UC),** unidad aritmético-lógica **(ALU)**, registros internos e interconexiones.

Las computadoras ***almacenan datos e instrucciones***, los dos aspectos fundamentales de la aritmética del computador son la forma de representar los números (sistema binario) y los algoritmos utilizados para realizar operaciones aritméticas básicas (resta, suma, etc).

SISTEMA DE NUMERACIÓN

Es un sistema de un conjunto de símbolos y reglas, que nos permiten construir todos los números validos para el sistema, clasificados en POSICIONALES Y NO POSICIONALES.

POSICIONALES: depende del símbolo y también de la posición que ocupa dentro del número (tiene base B, sólo hay B símbolos distintos para formar todos los números del sistema).

¿CÓMO?.. Para cada número se genera usando los distintos dígitos, y cada uno de ellos resulta multiplicado por una potencia de la base, de acuerdo a la posición que ocupan.

Teorema fundamental de la numeración

Este teorema establece la forma general de construir números en un sistema de numeración posicional. Primero establecemos unas definiciones básicas:

N es el valor de una cantidad expresada en **base B** y con (**n+1+d**) dígitos en posiciones **i**.

La fórmula general para construir un número N, con un número finito de decimales, en un sistema de numeración posicional de base b es la siguiente:

$$N = \sum_{i=-d}^{i=n} dígito_i \times base^i$$

Esta representación posibilita la realización de sencillos algoritmos para la ejecución de operaciones aritméticas.

TIPOS DE DATOS

Números enteros sin signo (BSS)

Si el número tiene n bits:

2^n = N°s distintos. (Depende del número de dígitos).

RANGO: $0 \rightarrow (2^n-1)$.

Representación en BCS (Binario con signo).

Con N bits, un bit representa el signo (es el más significativo) y n-1 bits a la magnitud.

S= Representa solo el signo y NO forma parte del número

Magnitud= representa en BSS.

0 INDICA POSITIVO (+) y 1 INDICA NEGATIVO (-).

RANGO: [$-(2^{n-1}-1) \rightarrow +(2^{n-1}-1)$]. Con dos ceros (+0→ 00000000, -0 → 10000000).

Números negativos: 111111111→ $-(2^{n-1}-1) = -127$.

Numero positivos: 01111111→ $(2^{n-1}-1)= 127$.

Por ejemplo: +18= 00010010; -18= 10010010 (signo-magnitud). El rango de BCS va desde (-2^n-1) a (2^n-1).

BCS y BSS tienen la misma cantidad de combinaciones pero lo que cambia necesariamente es el rango de cada sistema.

Punto fijo

Todos los números a representar tienen exactamente la misma cantidad de dígitos y la coma fraccionaria se ubica siempre en el mismo lugar.

Su almacenamiento en la computadora no se guarda coma alguna, se supone que está en un lugar determinado.

RANGO Y RESOLUCIÓN.

Rango: es la diferencia entre el número a representar más chico y el número representable más grande.

Resolución: es la diferencia entre dos números consecutivos.

ERROR

El máximo error en pto. fijo cometido puede considerarse como la mitad de la diferencia (resolución) entre dos números.

Los **FLAGS** son bits que establece el procesador automáticamente de acuerdo con los resultados de las operaciones realizadas.

¿Para QUÉ?

*Determinar la relación entre dos números (mayor, menor, igual).

*Realizar o NO una transferencia de control.

¿Cuáles son?

- **Z** (cero): se pone en 1 cuando el resultado da 0.
- **C** (carry): vale 1 en la suma si hay un acarreo del bit más significativo; en la resta vale 1 si hay borrow hacia el bit más significativo. Sirve para indicar una condición de fuera de rango.
- **N** (negativo): Si empieza con 1 entonces N=1 y el número es negativo (Ca2).
- **V** (oVerflow): cuando la resta o suma involucra a números con signo (Ca2), V=1 indica una condición fuera de rango, significa cantidad de bits para expresar que el resultado no es suficiente.

Hexadecimal codificado en binario (BCH)

- Los dígitos se convierten uno a uno en binario.
- Siempre se necesitan 4 bits para representar un dígito hexadecimal.

Decimal codificado en binario (BCD)

- Los dígitos se convierten uno a uno en binario.
- Para representar un dígito decimal se requerirán 4 bits.

Tiene dos ámbitos de aplicación:

El BCD puede ser empaquetado o desempaquetado:

- ***Desempaquetado:*** E/S y periféricos, los números se codifican usando un byte por dígito.
- ***Empaquetado***: se utiliza en cálculo, se reservan 4 bits por dígito.

En el desempaquetado:

Si se tiene que representar un número sin signo, utilizo 4 bits para el binario puro y los restantes relleno con bits a la izquierda con "1". Por ejemplo: 834: 11111000 11110011 11110100= F8 F3 F4.

Si es con signo, relleno con C_{16}=1100 ($12_{(10)}$) para los positivos y D_{16}= 1101 ($13_{(10)}$) para los negativos en el último digito solamente, los otros con 1111. Por ejemplo: +834: 11111000 11110011 11000100 = F8 F3 C4 y -834: 11111000 11110011 11010100 = F8 F3 D4

En el empaquetado:

+834: 10000011 01001100= 83 4C

-34: 00000011 01001101= 03 4D

SUMA en BCD

Usamos 10 dígitos, del 0 al 9. Nos sobran 6 combinaciones de 4 bits.

Al sumar dos dígitos BCD, se nos presentan dos casos:

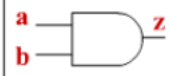
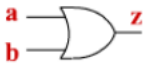
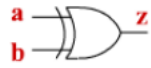
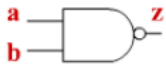
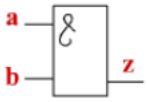
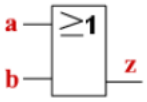
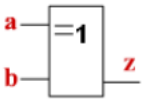
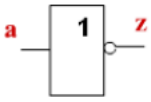
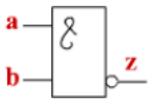
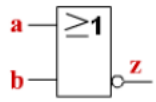
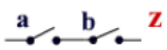
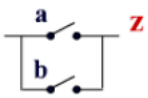
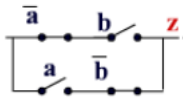
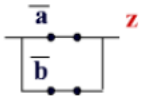
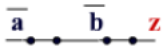
- La suma es ≤ 9
- La suma es > 9

Cuando la suma de los dos dígitos da > 9 hay que generar el "acarreo" porque hay seis combinaciones no usadas. Entonces es ahí cuando hay que sumar 6 en ese dígito.

Lógica digital

Un circuito digital es aquel en el que están presentes dos valores lógicos, hay compuertas (dispositivos electrónicos) encargadas de realizar distintas funciones con estos valores lógicos, y estas compuertas son:

FUNCIONES LÓGICAS BÁSICAS

NOMRE	AND - Y	OR - O	XOR O-exclusiva	NOT Inversor	NAND	NOR
SÍMBOLO						
SÍMBOLO						
TABLA DE VERDAD	a b z 0 0 0 0 1 0 1 0 0 1 1 1	a b z 0 0 0 0 1 1 1 0 1 1 1 1	a b z 0 0 0 0 1 1 1 0 1 1 1 0	a z 0 1 1 0	a b z 0 0 1 0 1 1 1 0 1 1 1 0	a b z 0 0 1 0 1 0 1 0 0 1 1 0
EQUIVALENTE EN CONTACTOS						
AXIOMA	$z = a \cdot b$	$z = a + b$	$z = \bar{a} \cdot b + a \cdot \bar{b}$	$z = \bar{a}$	$z = \overline{a \cdot b}$	$z = \overline{a + b}$

Para describir estos circuitos se utiliza el álgebra booleana, donde las variables y funciones solo puedan adoptar 1 o 0.

La tabla de verdad consta de 2^n combinaciones de valores de entrada que además describe los renglones de dicha tabla, y permite describir la función en cada combinación de 1 o 0 distinto.

TABLA DE IDENTIDADES DEL ÁLGEBRA DE BOOLE:

Identidad	$1.A=A$	$0+A=A$
Nula	$0.A=0$	$1+A=1$
Idempotencia	$A.A=A$	$A+A=A$
Inversa	$A.\overline{A}=0$	$A+\overline{A}=1$
Conmutativa	$A.B=B.A$	$A+B=B+A$
Asociativa	$(AB).C=A(BC)$	$(A+B)+C=A+(B+C)$
Distributiva	$A+B.C=(A+B).(A+C)$	$A.(B+C)=AB+AC$
Absorción	$A.(A+B)=A$	$A+A.B=A$
De Morgan	$\overline{A.B}=\overline{A}+\overline{B}$	$\overline{A+B}=\overline{A}.\overline{B}$

SUMA DE PRODUCTO Y PRODUCTO DE SUMAS.

Suma de productos

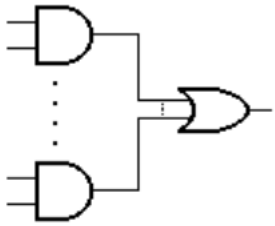
Es la suma de dos o más productos mediante la adición (suma) booleana.

Una barra no puede extenderse a sobre más de una variable:

Válido: $\overline{A}\overline{B}\overline{C}$ No válido: $\overline{ABC}$

La implementación de una suma de productos simplemente requiere aplicar la operación OR a las salidas de dos o más puertas AND.

El circuito combinatorio de suma de productos debe de tener el siguiente patrón:



EJ:..

M=~ABC + A~BC + AB~C + ABC

- Hay tantos términos como 1s en la tabla.
- Cada término vale 1 para una única combinación de A, B y C.
- Las variables que valen 0 en la tabla aparecen aquí negadas.

Producto de sumas

Productos (AND) de términos sumas (OR) formados por varias variables complementadas o no.

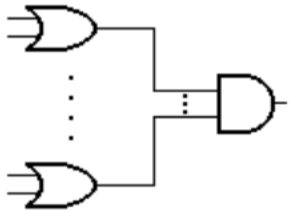
Cuando dos o más términos de suma se multiplican.

Una barra no puede extenderse a sobre más de una variable:

Válido: $\overline{A}\overline{B}\overline{C}$ No válido: ~~$\overline{ABC}$~~

La implementación de un producto de sumas requiere simplemente la aplicación de la operación AND a las salidas de dos o más puertas OR.

El circuito combinatorio de un producto de sumas debe de tener el siguiente patrón:



TECNICAS DE COMPLEMENTOS

El complemento a un número N de un número A (A < N) es igual a la cantidad que le falta a A para ser N.

Complemento a N de A = N - A

El complemento a un número N del número (N - A) es igual a A.

Complemento a N de (N-A) = N - (N-A) = A

Complemento a 1

Si el número es positivo, se representa igual que siempre.

Si el número es negativo, se invierten todos los bits, o el peso que tiene el primer dígito es $-(2^{n-1}-1)$ y el resto de los dígitos está como siempre, o por definición del complemento a la base disminuida: $Ca1 = (b^n - 1) - N°$.

El **rango** va $-(2^{n-1}-1)$ a $(2^{n-1}-1)$. Hay dos ceros, tiene un intervalo simétrico.

Numero negativos: 11111111 -0 → 10000000 $-(2^{n-1} - 1) = -127$.

Números positivos: 01111111 + $(2^{n-1} - 1) = +127$ → 00000000 +0.

Complemento a 2

Si el número es positivo, se representa igual que siempre.

Si es negativo: se obtiene invirtiendo todos los bits y luego sumarle uno(1), o "mirando" desde la derecha se escribe un número igual hasta el primer "1" inclusive y luego invierten los demás dígitos, o por definición de complemento a la base: $Ca2 = (b^n - N°)$.

El **rango** va desde $-(2^{n-1})$ a $(2^{n-1}-1)$. Hay un solo cero, con intervalo asimétrico (hay un negativo más).

Numero negativos: 11111111 -1 →10000000 - (2n-1)= -128.

Numero positivos: 01111111 + (2n-1 –1)=+127 →00000000 +0.

Exceso

La representación de un número A es la que corresponde a la suma del mismo y un valor constante E (exceso).

Exceso E de A = A + E + (2^{n-1}-1).

Dado un valor, el número se obtiene restando el valor del exceso.

A = (exceso E de A) –E. (EL SIGNO DEL NUMERO A RESULTA DE UNA RESTA.

El **rango** va desde $-2^{(n-1)}$ <=x <= $2^{(n-1)}$-1.

• Representar el N° en Ca2 y luego sumarle cantidad fija (Exceso). El exceso se forma con 1 en el bit más significativo (más a la izquierda seguido de todos ceros).3

Punto flotante

El punto fijo tiene limitaciones al representar números muy grandes y muy pequeños; esto deriva en el punto flotante; el cual, mediante el desplazamiento de la coma N veces y la multiplicación por su base N veces, permite representar números muy grandes y muy chicos con la misma cantidad de bits, la resolución va cambiando a lo largo del intervalo. Esto quiere decir que tendremos un rango mayor, por eso se paga el precio de que esten los N°s más separados o espaciados.

Ahora los números se representan mediante una mantisa (M) y un exponente (E). La base no necesita almacenarse ya que va a ser siempre la misma. $\pm M \times B^{\pm E}$.

MANTISA

La mantisa normalizada surge con el objetivo de tener un único par de valores de mantisa y exponente para un número. Todas las mantisas fraccionarias normalizadas se definen como 0,1…

BIT IMPLÍCITO

Ya que todos los números normalizados empiezan con 0,1…, este 1 siempre está presente, por ese motivo no lo almacenamos y podemos “adicionar” un bit más a la mantisa, por lo tanto ganamos mayor precisión al momento de representar determinado número. → a esto se lo conoce como bit implícito (bit NO almacenado).

Construir un numero en punto flotante normalizado:

1- Se escribe en el sistema propuesto para la mantisa
2- Se corre la coma y se cambia el exponente hasta normalizar
3- Se convierte el exponente al sistema propuesto

Error absoluto

Es la diferencia entre el valor representado (el que se pudo) y el valor a representar.

ERROR ABSOLUTO MÁXIMO <= Resolución/2.

ERROR RELATIVO = EA/Número a representar.

Error relativo

Es el cociente de la división entre el error absoluto y el valor exacto. Si se multiplica por 100, se obtiene el tanto por ciento (%) de error

ESTANDAR IEEE 754

Este estándar se desarrolló para facilitar la portabilidad de los programas de un procesador a otro y para alentar el desarrollo de programas numéricos sofisticados. Este estándar ha sido ampliamente adoptado y se utiliza prácticamente en todos los procesadores y coprocesadores aritméticos actuales.

Surge para poner un estándar en la forma de escribir un número en punto flotante.

	Simple	Doble precisión
Bits en signo	1	1
Bits en exponente	8	11
Bits en fracción	23	52
Total de bits	32	64
Exponente en exceso	127	1023
Rango de exponente	-126 a +127	–1022 a +1023
Rango de números	2^{-126} a $\sim 2^{128}$	2^{-1022} a $\sim 2^{1024}$

Sumar o resta

- Comprobar valores de ceros.
- Ajustar mantisas (ajustar exponentes).
- Sumar o restar mantisas.
- (si es normalizada) normalizar el resultado.

Multiplicar o dividir

Mucho más sencillo que la suma y resta.

- Sumar exponentes.
- Multiplicas o dividís mantisas (tener en cuenta el signo),
- (si es normalizado) normalizar el resultado.
- Redondeo.

Circuitos combinatorios

Los circuitos combinatorios es un conjunto de puertas interconectadas que responden a los valores lógicos de las entradas, es decir, que la salida depende de las entradas en ese momento y, por lo tanto, si cambia la entrada, cambia la salida. Por esto se dice que los sistemas combinacionales no cuentan con memoria. Un circuito combinacional es un sistema que contiene operaciones booleanas básicas (AND, OR, NOT).

Consiste en ***n*** entradas binarias y ***m*** salidas binarias.

•**Tabla de verdad:** para cada una de las 2^n combinaciones posibles ***n*** señales de entrada, se enumera el valor binario de cada una de las ***m*** señales de salida.

•**Símbolo gráfico:** describe la organización de las interconexiones entre puertas.

Circuitos secuenciales

Se usa una forma de circuitos lógicos más digitales más compleja: los circuitos secuenciales. La salida actual de un circuito secuencial depende no sólo de la entrada actual, sino también de la historia pasada de las entradas. Esto se debe a que tienen memoria y son capaces de almacenar información a través de sus estados internos. En estos circuitos, las salidas son también entradas, es decir, que una salida transporta información hacia la entrada.

Otra manera de ver estos circuitos es que la salida actual de un circuito secuencial depende de la entrada actual y del estado actual del circuito.

Biestables

La forma más sencilla de un circuito secuencial es un biestable. Hay varios tipos, entre todos ellos comparten dos propiedades:

- El biestable es un dispositivo con dos estados '0' y '1'. Está en uno de dos estados de ausencia de entrada, recordando el último estado. Entonces, el biestable puede funcionar como una memoria, capaces de almacenar un bit de información.
- Tiene dos salidas, que son siempre complementarias. Normalmente se denominan Q y Q (Q negada)
  Estos pueden tener dos salidas Q y Q negada, las cuales son siempre complementarias.
- Clasificación:
  1- Según utilicen o no una señal de reloj:

  -Síncronos

  -Asíncronos

  2- Según se activen por flanco o por nivel:

  -Latches (cerrojos), activos por flanco

  -Flip-flops, activos por nivel

Los Flip-flops son ampliamente usados para el almacenamiento y transferencia de datos digitales y se usan normalmente en unidades llamadas "registros", para el almacenamiento de datos numéricos binarios.

Los flip-flops cambian sus salidas si sus entradas cambian, pero depende de si son sincrónicos o asincrónicos cuando es que esto sucede:

- Asincrónicos: cuando en la entrada se establece alguna combinación, las salidas cambian. Tras un breve tiempo de retardo, en respuesta a un cambio en la entrada. En el más empleado es el biestable RS.
- Sincrónicos: hay una entrada especial que determina cuando cambian las salidas. Además de las entradas de control posee una entrada de sincronismo o de reloj, este es una señal de tiempo precisa que determina cuando ocurren los eventos.

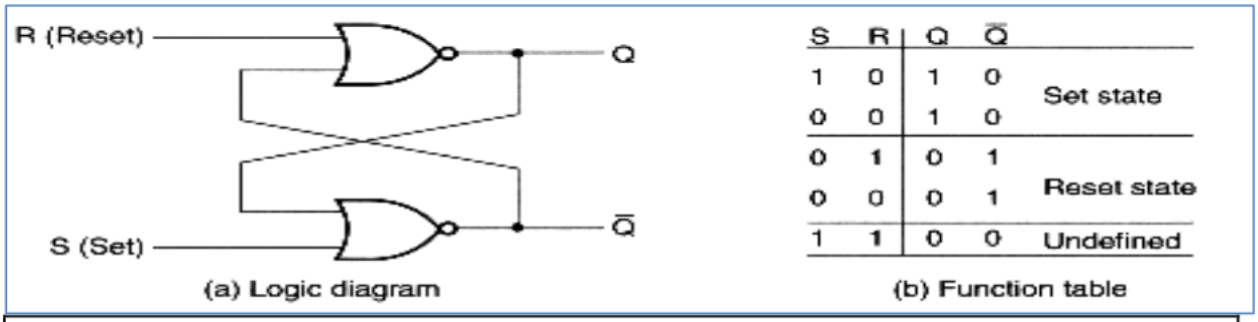
Estos flip flops biestables se clasifican según la manera en que responden a las señales de entrada:

S-R asincrónico: Dispositivo de almacenamiento temporal de 2 estados (alto y bajo), cuyas entradas principales permiten al ser activadas:

- R: el borrado (*reset* en inglés), puesta a 0 o nivel bajo de la salida.
- S: el grabado (*set* en inglés), puesta a 1 o nivel alto de la salida

Si no se activa ninguna de las entradas, el biestable permanece en el estado que poseía tras la última operación de borrado o grabado. En ningún caso deberían activarse ambas entradas a la vez, ya que esto provoca que las salidas

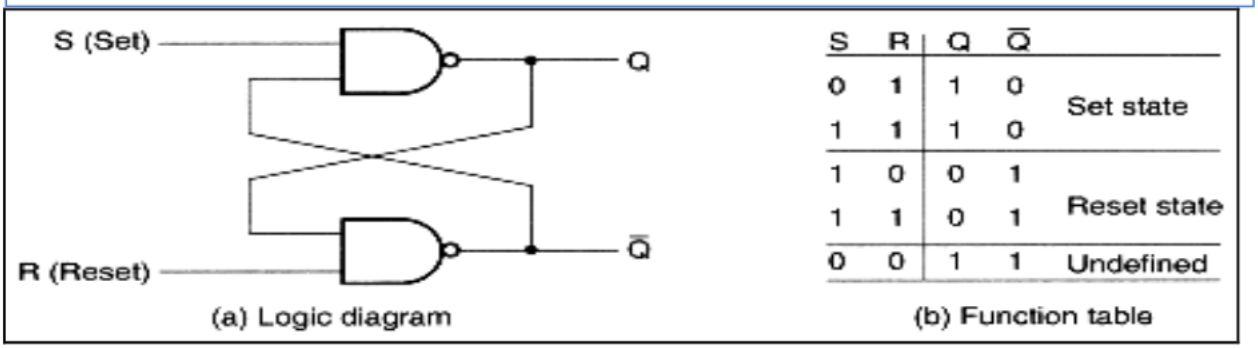
directas (Q) y negada (‘Q) queden con el mismo valor: ha bajo, si el flip-flop está construido con puertas NOR, o a alto, si está construido con puertas NAND.



(a) Logic diagram

S	R	Q	$\overline{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

(b) Function table



(a) Logic diagram

S	R	Q	$\overline{Q}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

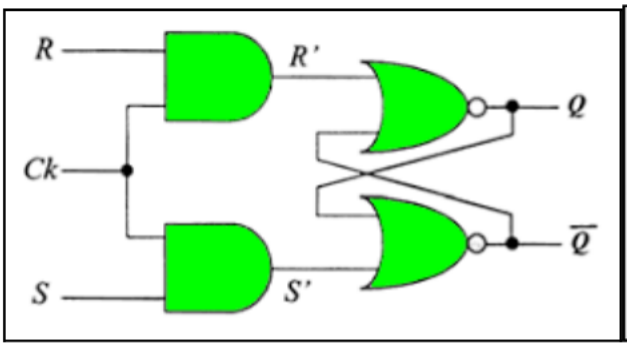
(b) Function table

El biestable S-R se puede definir con una tabla parecida a la tabla de verdad, llamada *tabla característica*, que muestra el siguiente estado o estados de un circuito secuencial en función de los estados y entradas actuales.

Se muestran que las entradas para las (NOR) S= 1 y R=1 no están permitidas, ya que producirían una salida inconsistente (Q y $\underline{Q}$ igual a 0). Caso contrario con las (NAND) cuando S=0 y R=0; (Q y $\underline{Q}$ igual a 1).

Biestable S-R sincrónico

Además de las entradas R y S, posee una entrada Clock de sincronismo cuya misión es la de permitir o no el cambio de estado del biestable. En la siguiente figura se muestra un ejemplo de un biestable síncrono a partir de una asíncrona. (a) Circuito biestable s-r sincrónico y (b) esquema normalizado.



CK	S	R	Q_{n+1}
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Prohibido
0	x	x	Q_n

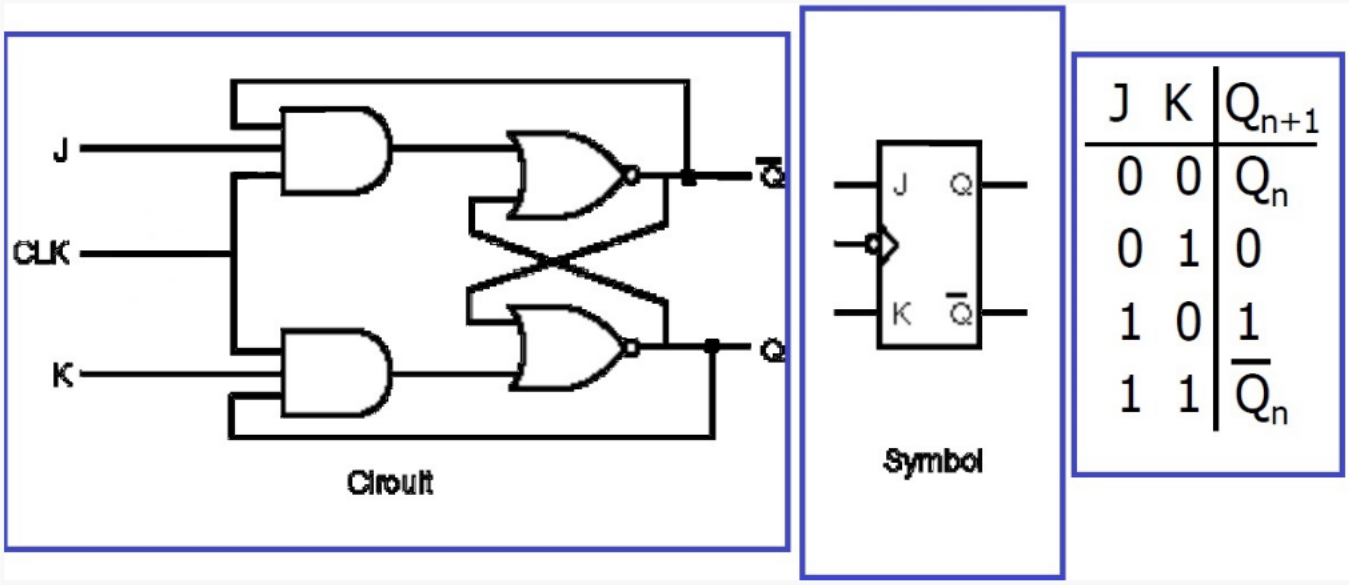
Biestable J-K

Es versátil y es uno de los tipos de flip-flop más usados. Su funcionamiento es idéntico al del flip-flop S-R en las condiciones SET, RESET y de permanencia de estado. La diferencia está en que el flip-flop J-K, todas las combinaciones posibles de los valores de entrada son válidos.

Este dispositivo de almacenamiento es temporal que se encuentra dos estados (alto y bajo), cuyas entradas principales, J y K, a las que debe el nombre, permiten al ser activadas:

- J: (Jump en inglés), puesta a 1 ó nivel alto de la salida.
- K: (Keep en inglés), puesta a 0 ó nivel bajo de la salida.

Si no se activa ninguna de las entradas, el biestable permanece en el estado que poseía tras la última operación. A diferencia del biestable RS, en el caso de activarse ambas entradas a la vez (son ambos high; alto;), la salida adquirirá el estado contrario al que tenía. Si J y K son diferentes, la salida Q toma el valor de J durante la subida del siguiente pulso de sincronismo.

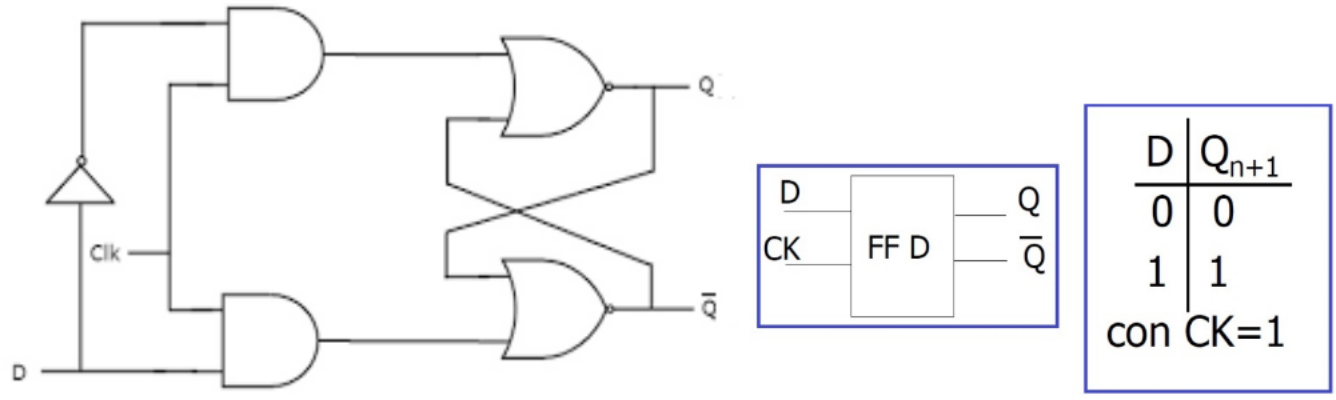


.

Biestable D

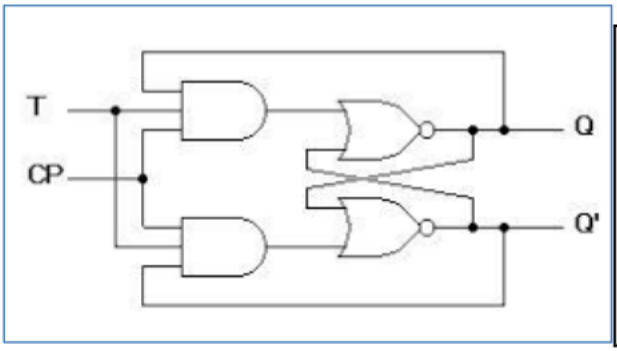
Es uno de los flip-flops más sencillos, se denomina biestable de datos porque es, en efecto, un almacén para un bit de datos (1 o 0). La salida es siempre igual al valor más reciente aplicado a la entrada. Por lo tanto, recuerda y produce la última entrada. También se le llama biestable de retardado porque retrasa un 0 o un 1 aplicado a la entrada durante un pulso del clock.

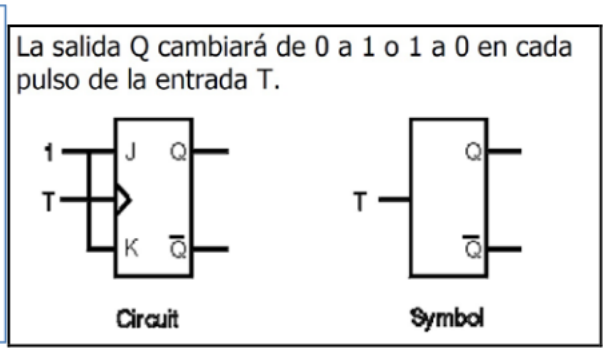
Su función es dejar pasar lo que entra por D, a la salida Q, después de un pulso del reloj.



Biestable T

Dispositivo de almacenamiento temporal de 2 estados (alto y bajo). El biestable T cambia de estado cada vez que la entrada de sincronismo o de reloj se dispara mientras la entrada T está a nivel alto. Si la entrada T está a nivel bajo, el biestable retiene el nivel previo.

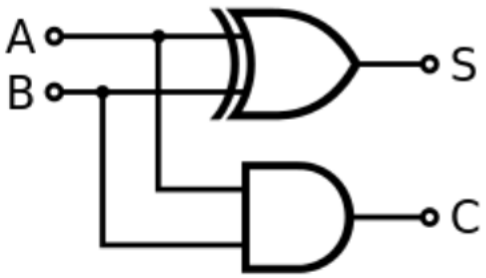




Semi sumador

El semisumador suma dos dígitos binarios simples A y B, denominados sumandos, y sus salidas son Suma (S) y Acarreo (C). La señal de acarreo representa un desbordamiento en el siguiente dígito en una adición de varios dígitos. El diseño más simple de semisumador, representado a la derecha, incorpora una puerta XOR para S y una puerta AND para C. Dos semisumadores pueden ser combinados para hacer un sumador completo, añadiendo una puerta OR para combinar sus salidas de acarreo. La tabla de verdad para el semisumador se detalla seguidamente:

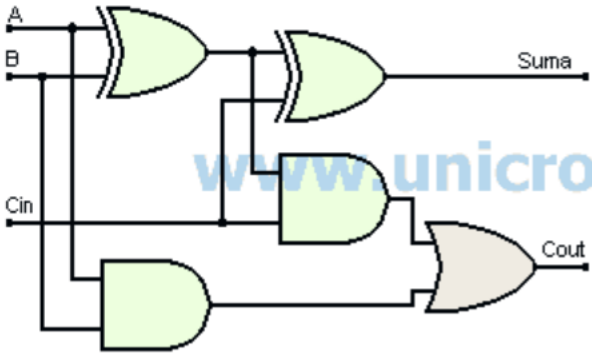
Entradas		Salidas	
A	**B**	**Acarreo**	**Suma**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Sumador completo

Un sumador completo suma números binarios junto con las cantidades de acarreo. Un sumador completo de un bit añade tres bits, a menudo escritos como A, B y C_{in} siendo A y B son los sumandos y C_{in} es el acarreo que proviene de la anterior etapa menos significativa. El circuito produce una salida de dos bits, al igual que el semisumador, denominadas acarreo de salida (C_{out}) y suma S.

Si queremos sumar dos números de n bits, esto se puede hacer poniendo juntos un conjunto de sumadores de forma tal que el acarreo de un sumador sea la entrada del siguiente (digamos cuando en el sistema decimal hay un "llevo" debido a la suma de las unidades y hay que pasarlas a las decenas).



Entradas			Salidas	
A	B	Ci	Cout	Suma
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Modelo de Von Neuman

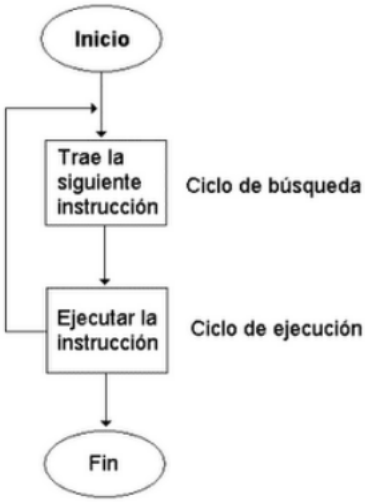
Consta de cinco componentes principales:

1- (UE) Unidad de entrada: provee instrucciones y datos.
2- (UM) Unidad de memoria: se almacena datos e instrucciones.
3- (ALU) Unidad aritmética-lógico: procesa los datos.
4- (UC) Unidad de control: dirige la operación.
5- (US) Unidad de salida: se envían los resultados.

Ciclo de instrucción (función de la computadora)

Consiste en ejecutar un programa, trayendo desde la memoria las instrucciones que lo componen una por una y cumplirlas de forma ordenada.

El procesamiento de una instrucción se pude dividir en su búsqueda (lectura desde memoria) y ejecución. Solo se interrumpe si ocurre algún error.



La CPU cuenta con un registro llamado Program Counter (CP), el cual almacena la dirección de memoria de las próximas instrucciones que debe buscar. Cuando va a buscar una instrucción el cp se incrementa y así apunta a la siguiente.

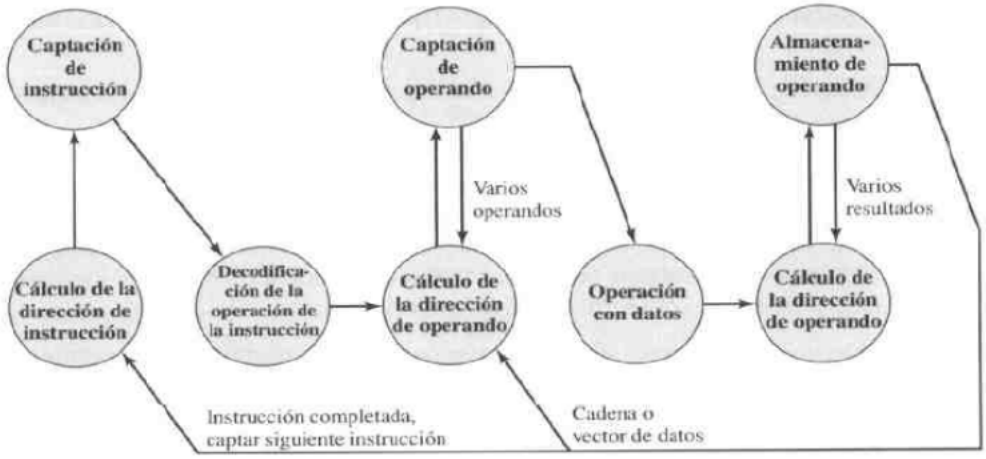
La instrucción buscada se guarda en un registro de CPU llamado Instruction Registrer (IR) en un código binario que permite a la CPU entender la instrucción y llevar a cabo las acciones.

Las acciones caen en cuatro tipos:

- CPU-memoria, datos pueden transferirse entre memoria y CPU.
- CPU-entrada/salida, datos pueden transferirse entre CPU y E/S.
- Procesamiento de datos: CPU ejecuta aritméticas o lógicas en los datos.
- Control: alterar la secuencia de ejecución de instrucciones.

Los pasos detallados serian:

1- Determino la dirección de la siguiente dirección.
2- Leo la instrucción en su posición.
3- Incremento PC.
4- Decodifico la instrucción y determino su tipo y sus operandos.
5- Si necesito operando de memoria o de E/S.
6- Busco los operandos en la memoria o E/S.
7- Incremento PC.
8- Ejecuto la instrucción.
9- Calculo la dirección del resultado.
10- Almaceno el resultado.



Repertorio de instrucciones

El diseño de instrucciones es el medio que tiene el programador para controlar la CPU. Hay que tener en cuenta cuantos y cuales tipos de operaciones y cuales tipos de datos.

- Formato de instrucciones: longitud de la instrucción (en bits, número de direcciones, tamaño de los distintos campos).
- Registros: cantidad que se pueden referenciar mediante instrucciones y su uso.
- Direccionamiento: manera de referenciar la dirección de un operando o instrucción.
- Repertorio de operaciones: cuantas y que operaciones considerar, y cuan complejas deben ser. Transferencia de datos, aritméticas y lógicas.
- Tipos de datos: los distintos tipos de datos con los que se realizan las operaciones. Números (enteros, punto fijo y flotante), carácter y datos lógicos.

Formato de instrucción

Los elementos de una instrucción son:

- Código de operación: código binario que especifica la operación a realizar.
- Referencia del operando fuente: establece donde almacena el resultado.
- Referencia del operado resultado: establece donde almacena el resultado.
- Referencia de la siguiente instrucción: le dice a la CPU, donde buscar la siguiente instrucción previo a la ejecución anterior. Los operandos fuentes pueden estar en: memoria, registro de la CPU, dispositivos de E/S.

Instrucciones de maquina:

- Procesamiento de datos
- Almacenamiento de datos
- Instrucciones de E/S
- Control

Máquina de 4 direcciones: ADD DirRes, DirOp1, DirOp2, DirProxInstr. Tiene direcciones explicitas para operandos, resultado y próxima instrucción. Son raras, cada campo de dirección tiene que tener bits para acomodar una dirección completa.

Máquina de 3 direcciones: ADD DirRes, DirOp1, DirOp2. La dirección de la próxima instrucción esta almacenada en un registro de la CPU, llamado contador de programas (PC). ADD, MUL, SUB.

Máquina de 2 direcciones: ADD DirOp1, DirOp2. Reduce el tamaño de la instrucción. Hay que mover el operando 1 a un registro temporal. Menos elección donde guardar el resultado. MOV, ADD, MUL, SUB.

Máquina de 1 dirección: ADD DirOp1. Tiene un registro especial (acumulador). Instrucciones para cargar y descargar el acumulador. LOAD, ADD, MUL, SUB, STORE.

Organización de los Registros

Dentro de la CPU hay un conjunto de registros que funciona como un nivel de memoria, por encima de la memoria principal y de la cache en la jerarquía. Los registros de la CPU son de dos tipos:

1- Registros visibles para el usuario: permiten minimizar las referencias a memoria principal cuando se optimiza el uso de registros.
2- Registros de control y de estado: son utilizados por la UC para controlar el funcionamiento de la CPU y por programas privilegiados del sistema operativo para controlar la ejecución de programas.

Registros visibles para el usuario

Se clasifican de la siguiente manera:

- Uso general: pueden ser asignados por el programador a diversas funciones.
- Datos: se usan únicamente para contener datos y no se pueden emplear en el cálculo de una dirección de operando.
- Direcciones: pueden ser de uso más o menos general, o pueden estar dedicados a un modo de direccionamiento particular.
- Códigos de condición (flags): son bits fijados por hardware de la CPU como resultado de una operación. Forma parte de un registro de control. Son parcialmente visibles al usuario y no pueden ser alterados.

Registros de control y de estado

Son esenciales cuatro registros para la ejecución de una instrucción:

- Contador de programa (PC): contiene la dirección de la instrucción a captar.
- Registro de instrucción (IR): contiene la instrucción captada más recientemente.
- Registro de dirección de memoria (MAR): contiene la dirección de una posición de memoria.

- Registro de intermedio de memoria (MBR): contiene la palabra de datos a escribir en memoria, o la palabra leída más recientemente.

Los cuatros registros se usan para la transferencia de datos entre la CPU y la memoria. Dentro de la CPU, los datos tienen que ofrecerse a la ALU para su procesamiento. La ALU puede tener acceso directo a MBR y a los registros visibles para el usuario. Como alternativa, puede haber registros intermedios adicionales entorno a la ALU que sirven como registros de E/S de la ALU, e intercambian datos con MBR y los registros visibles para el usuario.

Modos de direccionamiento

Los modos de direccionamiento tienen como objetivo reducir los bits de la instrucción, manejo más eficiente de datos y que la dirección no se conozca hasta ejecutarse. Hay dos métodos generales:

1- Si un operando va a usarse varias veces, puede colocarse en un registro, para acceder más rápido y usar menos bits.
2- Especificar uno o más operandos en forma implícita.

Los ***modos*** de direccionamiento son:

- **Inmediato**: el operando se obtiene automáticamente de la memoria al mismo tiempo que la instrucción. No requiere una referencia extra a memoria. Se usa para definir constantes e inicializar variables. Como ventaja, no se requiere acceso adicional a memoria para obtener el operando, ahorrándose pues un ciclo de memoria o de caché en el ciclo de instrucción, la desventaja es que el tamaño del número está restringido a la longitud del campo de direcciones que, en la mayoría de los repertorios de instrucciones, es pequeño comparado con la longitud de la palabra, también son que el valor del dato es constante y el rango de valores que se pueden representar está limitado por el tamaño de este operando.
- Normalmente , el número se almacena en complemento a dos.
- Ejemplo: MOV AX,12

- **Directo**: En este modo la dirección efectiva es igual a la parte de dirección de la instrucción. El operando reside en la memoria y su dirección es dada directamente por el campo de dirección de la instrucción, por lo que no necesita ningún cálculo especial. Se usa para acceder a variables globales, cuya dirección se conoce al compilar. Ejemplo: MOV AX, 17H.

- Por **registro**: como el directo solo que se apunta a un registro, no a una posición de memoria. Al hacer esto se requiere de menos bits y no se accede a memoria de datos. Lo malo es que no hay muchos registros y estos son muy preciados.

- **Indirecto por memoria**: en la instrucción se encuentra la dirección de la dirección del operando. Sirve para apuntar a una dirección de más bits de los que tiene el campo, y asi se consigue un espacio de direccionamiento mayor, con la desventaja que se accede más veces a memoria. Ejemplo: MOV A,@17H

- **Indirecto por registro:** en la instrucción se especifica el registro que tiene almacenado la dirección. La ventaja es que para direcciones a registros se usan menos bits que para memoria.

Con desplazamiento: requiere que la instrucción tenga dos campos de dirección, al menos uno de los cuales explícito. Estos dos campos se suman para obtener la dirección deseada. Combina el modo directo e indirecto mediante registros. Los más comunes son: relativo, indexado y de registro base.

- Relativo: el registro implícito es el PC. La dirección de la instrucción actual se suma al campo de dirección para producir la dirección definitiva. El campo se trata como un número en CA2.
- Indexado: se direcciona la memoria con un registro y un desplazamiento. Se intercambian los papeles del registro y el desplazamiento. Se utiliza un registro llamado índice.
- De registro base: el registro referenciado contiene una dirección de memoria y el campo tiene un desplazamiento.

Del stack: el stack es un arreglo lineal de localidades de memoria, una zona de memoria reservada. Es una lista donde el último en entrar es el primero en salir. Hay un registro apuntador “puntero” cuyo valor es la dirección tope de la pila. El modo de direccionamiento de pila es una forma de direccionamiento implícito, las instrucciones máquina no necesitan incluir una referencia a memoria sino que operan implícitamente con la cabecera de la pila.

Tipos de operaciones

Los códigos de operación varían de una maquina a otra, pero las operaciones son las mismas. Los tipos de operaciones son: transferencias de datos, aritméticas, lógicas, conversión, entrada/salida, control del sistema, control de flujo.

Transferencia de datos: la instrucción de transferencia debe especificar varias cosas: posiciones de los operandos fuente y destino, longitud de los datos a transferir, modo de direccionamiento para cada operando.

Aritméticas: las operaciones aritméticas básicas son: suma, resta, multiplicación y división; pero hay operaciones que requieren un solo operando: absolute, negative, increment, decrement.

Lógicas: se basan en operaciones booleanas. En este caso se aplican operando básicos como AND, OR, XOR, EQUAL. Además se incluyen operaciones de desplazamiento y rotación.

Tipos de datos

Los operandos de las instrucciones sirven para expresar el lugar donde están los datos que hemos de utilizar. Estos datos se almacenan como una secuencia de bits y, según la interpretación de los valores almacenados, podemos tener tipos de datos diferentes que, generalmente, también nos determinarán el tamaño de los operandos.

A continuación presentamos los tipos generales de datos más habituales:

1- Dirección: Tipo de dato que expresa una dirección de memoria
2- Número: Tipo de dato que expresa un valor numérico. Habitualmente distinguimos tres tipos de datos numéricos (y cada uno de estos tipos se puede considerar con signo o sin signo): Números enteros, Números en punto fijo, Números en punto flotante.
3- Carácter: Tipo de dato que expresa un carácter. Habitualmente se utiliza para formar cadenas de caracteres que representarán un texto..
4- Dato lógico: Tipo de dato que expresa un conjunto de valores binarios o booleanos;

Memoria

Clasificación según las características:

Ubicación: - CPU (registros), interna (memoria principal), externa (memoria secundaria, dispositivos de almacenamiento)

Capacidad: para memorias internas se expresa normalmente en bytes o palabras de longitud de 8, 16 y 32 bits (generalmente). Para memorias externas se expresa en bytes

- Registros: 1KB
- Memoria principal: 128 MB
- Cache: 1 MB
- Discos: 40 GB

Duración de la información:

- Volátiles: la información se pierde o desaparece cuando se desconecta la alimentación. Ej: RAM.
- No volátiles: la información permanece grabada sin modificaciones, a menos que se realicen intencionalmente. Ej: discos, cintas.
- Permanentes: no pueden ser modificadas a menos que se destruya la unidad de almacenamiento. Ej: ROM, EPROM.

Unidad de transferencia:

- Palabra (para la memoria interna)
- Bloques, unidades más grandes que la palabra (para la memoria externa).

Métodos de acceso

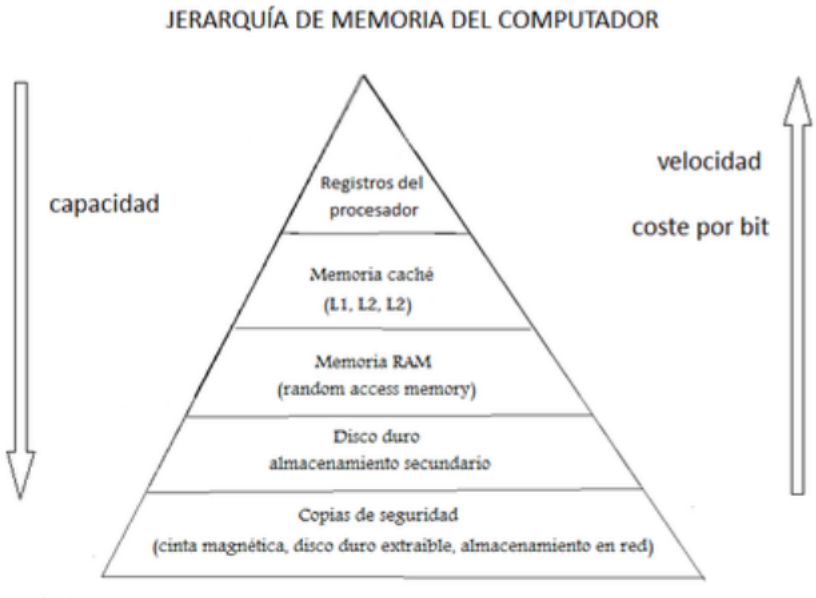
- Acceso aleatorio: el tiempo para acceder a una variante dada es independiente de la secuencia de accesos anteriores y es constante.
- Acceso secuencial: el acceso debe hacerse en una secuencia lineal específica. Variable.
- Acceso directo: los bloques o registros tienen una dirección única que se basa en la localización física.
- Acceso asociativo: es una memoria de tipo aleatorio, que permite comparar ciertas posiciones de bits dentro de una palabra buscando que coincidan con unos valores dados. Memoria cache.

Velocidad:

- Memoria semiconductora: tiempo de acceso (tiempo máximo desde que se inicia la operación de lectura/escritura hasta obtener/almacenar el dato). Los tiempos de acceso de un registro es 1 ns, de la cache 1-20 ns, de la memoria principal 60-80 ns y de un disco 10 ms. Tiempo de ciclo (tiempo minimo que debe haber entre dos operaciones sucesivas. T ciclo menor t acceso).
- Memorias magnéticas: tiempo de acceso (tiempo de posicionar el cabezal + tiempo de latencia (+ tiempo de lectura)). Velocidad de transferencia (bytes/seg).

Jerarquía de memoria

La jerarquía de memoria es la organización piramidal de la memoria en niveles que tienen las computadoras. El objetivo es conseguir el rendimiento de una memoria de gran velocidad al coste de una memoria de baja velocidad, basándose en el principio de cercanía de referencias. La memoria debe seguir al procesador.



Los puntos básicos relacionados con la memoria pueden resumirse en:

- Capacidad
- Velocidad
- Coste por bit

La cuestión de la capacidad es simple, cuanto más memoria haya disponible, más podrá utilizarse.

La velocidad óptima para la memoria es la velocidad a la que el microprocesador puede trabajar, de modo que no haya tiempos de espera entre cálculo y cálculo, utilizados para traer operandos o guardar resultados.

En suma, el coste de la memoria no debe ser excesivo, para que sea factible construir un equipo accesible.

Los tres factores compiten entre sí, por lo que hay que encontrar un equilibrio.

Las siguientes afirmaciones son válidas:

- A menor tiempo de acceso, mayor coste por bit.
- A mayor capacidad, menor coste por bit.
- A mayor capacidad, mayor tiempo de acceso.

Memoria principal: memoria semiconductora

RAM (random access memory): es aleatoria ya que puede acceder a cualquier celda de memoria en el mismo tiempo, independientemente de la posición en la estructura. Se divide en:

- Dinámica: hecha con celdas, que almacenan datos como cargas en transistores, que al descargarse requieren refrescos periódicos para mantener memorizados los datos; llamada DRAM (almacenan más información que SRAM en la misma superficie, ya que los transistores son más chicos que los flip-flops) y se utiliza como memoria principal.
- Estática: los valores binarios se almacenan utilizando configuraciones de puertas que forman biestables; más rápidas, complejas y caras que las dinámicas, y no necesitan refresco; llamada SRAM, es más rápida que la DRAM y es utilizada como memoria cache.

El elemento básico de una memoria semiconductora es la celda de memoria. Estos comparten 3 propiedades:

1- Dos estables: para representar el 0 y 1.
2- Se puede escribir en ellas (al menos 1 vez).
3- Se pueden leer para conocer el estado.

La celda cuenta con 3 terminales capaces de llevar una señal eléctrica:

- Selección: selecciona la celda.
- Control: especifica lectura o escritura.
- Escritura/lectura de datos.

Organización de memoria

Una memoria de 1 bit la implementamos con flip- flops y armamos registros sencillos de N bits con flip-flop. Para una memoria más grande se necesita una organización diferente en la cual sea posible direccionar palabras individuales: chip. Cada chip contiene un arreglo de celdas de memoria. Se han empleado dos enfoques organizadores:

- Organización 2D: el arreglo está organizado en 2^w palabras de B bits cada una. Cada una de las líneas horizontales se conecta a cada posición de memoria, seleccionando un renglón. Las líneas verticales conectan a cada bit a la salida. El decodificador que está en el chip, tiene 2^w salidas para W entradas (bits del bus de dirección).
- Organización 2 1/2D: el arreglo es cuadrado. Los bits de una misma palabra están dispersos en distintos chips. La dirección ahora consta del renglón y la columna, por lo que hay 2 decodificadores

Comparación:

- En 2D todos los bits están en el mismo chip. En 2 1/2D están en distintos.
- 2D es muy y estrecho, numero grande de palabras de pocos bits
- 2D dificulta el uso eficaz de los circuitos correctores de error, mientras que en 2 1/2D al estar los bits dispersos en cada chip, hay menos probabilidad de error.
- En 2 1/2D al usar decodificación separa de filas y alumnos, reduce la complejidad de los decodificadores.

Memoria caché

El uso de esta memoria se sustenta de dos principios:

- Principio de Localidad Temporal: si una dirección es referenciada es muy probable que la misma dirección vuelva a ser referenciadas de nuevo muy pronto. Ejecución secuencial del código, tendencia de los programadores a hacer próximos entre si variables relacionadas, acceso a estructuras tipo matriz o pila.
- Principio de Localidad Espacial: si una dirección es referenciada es muy probable que las direcciones próximas a esta serán referenciadas de nuevo muy pronto. Formación de ciclos o bucles, subrutinas, pilas, etc.

La efectividad del cache se expresa a través de la frecuencia de aciertos (números de veces que el cache acierta direcciones, es decir, cuando los datos que necesita el CPU están en la cache). Un fallo de cache ocurre cuando los datos buscados no se encuentran en el cache.

La idea del cache es que cuando se usa una palabra, ella y alguna de la vecinas. Se traen de la memoria grande y lenta a la cache, para que la próxima vez que se necesite, el acceso se haga al cache y sea más rápido.

Tiempo de acceso de un disco magnético

Los componentes que definen el tiempo de acceso son: Por un lado, la cabeza de lectura, que tiene una cierta velocidad para alcanzar el cilindro buscado (tiempo de seek).

También los discos en si, que tienen una determinada velocidad de giro. Esto determina el tiempo de latencia, que ocurre desde que la cabeza de lectura se posiciona sobre el cilindro, hasta que el sector buscado pasa por debajo de la cabeza.

Para calcular el tiempo de acceso promedio debemos sumar el tiempo de seek y el tiempo de latencia promedio. Este último se puede calcular como el tiempo de giro de una pista dividido dos.

Impresoras

Se pueden encontrar las siguientes técnicas de impresión:

- De caracteres: Imprimen carácter por carácter de manera unidireccional o bidireccionalmente, por lo que son muy lentas
- De línea: Imprimen una línea de caracteres simultáneamente, por lo que son más rápidas.
- De página: Imprimen toda una página en un sola pasada, aunque internamente imprimen por línea.

A su vez hay distintos mecanismos de impresión:

De impacto: Son aquellas que imprimen mediante el impacto del cabezal sobre la hoja.

Matriz de puntos: Es un tipo de impresora muy lenta, que utiliza un conjunto de agujas que golpean una cinta entintada sobre el papel, imprimiendo punto a punto.

Inyección de tinta: Son aquellas que imprimen mediante la carga de gotas de tinta, por medio de electricidad estática.

Láser: Son aquellas que imprimen mediante la radiación de un láser sobre una superficie con propiedades electrostáticas, desde un tambor que tiene la imagen impregnada en un tóner.

Modem

MODEM: MOdulador, DEModulador. Convierte señales '0' y'1' en tono de audio y viceversa. La tasa de Bits/seg (bps) es el número de bits enviados por segundo. Tasa de baudio es el número de cambio de señal por segundo.

Su principal uso es para telecomunicaciones: convierte señales analógicas provenientes de un sistema telefónico a cadenas binarias. Es una de las tecnologías más usadas para conectarse a internet.