.:: RESUMEN ORGANIZACIÓN ::.

Índice

LA COMPUTADORA . . . 2
- La máquina de Von Neumann . . . 2
- Estructura y funcionamiento . . . 2
  Componentes de una computadora . . . 2
  Funciones básicas de una computadora . . . 2
  Flags . . . 2
- Formato,ciclo de instrucción y MDDs . . . 3
  Formato de la instrucción . . . 3
  Ciclo de instrucción . . . 3
  Modos de direccionamiento (MDDs) . . . 3

REPRESENTACIÓN DE DATOS . . . 3
- Tipos de datos . . . 3
  Representación de enteros . . . 4
  Punto flotante . . . 6

LÓGICA DIGITAL . . . 7
- Compuertas básicas . . . 7
- Leyes de De Morgan . . . 7
- Suma de productos . . . 7
- Máscaras . . . 7
- Circuitos combinacionales . . . 7
  Semi-sumador y sumador completo . . . 7
  Multiplexores . . . 8
  Decodificadores . . . 8
- Circuitos secuenciales . . . 8
  Biestables (Flip-Flops) . . . 8
  Reloj (Clock-CLK) . . . 8
  Tipos de Flip-Flops . . . 8

MEMORIAS . . . 10
- Jerarquía de memoria . . . 10
- Clasificación por características . . . 10
- Tipos de acceso . . . 10
- Memoria principal . . . 10
  Memorias semiconductoras . . . 10
    RAM . . . 11
    ROM . . . 11
    Organización 2D y 21/2D . . . 11
    Implementando una memoria . . . 13
    Memoria caché . . . 13
- Memoria secundaria . . . 14
  Disco magnético . . . 14
    RAID . . . 15
  CD-ROM . . . 15
  Cinta magnética . . . 15

PERIFÉRICOS . . . 16
- Teclado . . . 16
- Monitor . . . 16
- Impresora . . . 16
- Modem . . . 17

$> LA COMPUTADORA

- LA MÁQUINA DE VON NEUMANN

Fue diseñada en 1945 por John Von Neumann y estableció la arquitectura de las computadoras que, con algunas modificaciones, es empleada hoy en día por la mayoría de los fabricantes.

Arquitectura constituida por:
- Una **Unidad Aritmética Lógica**: realiza operaciones elementales como suma, comparaciones
- Una **Unidad de Control**: encargada de interpretar las instrucciones almacenadas en memoria para su ejecución
- **Dispositivos de E/S**: permiten la comunicación con el mundo exterior
- Una **Memoria Principal**: donde se almacenan los datos y las instrucciones
- Un **Bus de Comunicación**: permite el paso de información de direcciones, datos y control

- ESTRUCTURA Y FUNCIONAMIENTO

La ***estructura*** es el modo en que los componentes están relacionados
El ***funcionamiento*** es la operación de cada componente individual como parte de la estructura

- **Componentes de una computadora*:***

- ***CPU***: controla el funcionamiento del computador y lleva a cabo el procesamiento de datos
  # UC: controla el funcionamiento de la CPU y del computador
  # ALU: lleva a cabo las operaciones aritméticas y lógicas
  # REGS: proporcionan almacenamiento interno y temporal a la CPU
  # Interconexiones del CPU: comunica la UC, la ALU y los REGS
- ***Memoria principal***: almacena datos
- ***E/S***: transfiere datos entre el computador y el entorno externo
- ***Sistema de interconexiones (buses)***: proporciona comunicación entre la CPU, la mem. y E/S

REGS: son memorias rápidas y chicas, que almacenan información para el CPU temporalmente
  - ♦ ***Accesibles***:
  - *IR*: almacena temporalmente una instrucción
  - *PC*: contador de programa que apunta a la instrucción que debe ejecutarse
  - *De uso gral*. (Ax,Bx..): se utilizan como almacenamiento temporal
  - ♦ ***Ocultos*** [conectados a los buses]:
  - *MAR*: registro de dirección de memoria
  - *MBR*: registro de datos de memoria

Buses**:** un bus es un camino de comunicación que conecta dos o más dispositivos
  - ♦ *Bus de datos:* transporta datos
  - ♦ *Bus de direcciones:* identifica el origen y/o destino de los datos
  - ♦ *Bus de control:* brinda información de control y temporizado

- **Funciones básicas de una computadora*:***
- Procesamiento de datos
- Almacenamiento de datos
- Transferencia de datos
- Control

- **Flags**: son bits que el procesador establece de modo automático acorde al resultado de cada operación realizada

- *Z* (cero) = '1' si el resultado es todo 0
- *C* (carry) = '1' si hay carry en la suma, o borrow en la resta
- *N* (negativo) = '1' si el resultado es negativo (el bit más significativo es '1')
- *V* (overflow) = '1' si hay una condición fuera de rango en Ca2

- FORMATO, CICLO DE INSTRUCCIÓN Y MDDs

Dentro de la computadora cada instrucción está representada mediante una secuencia de bits. La secuencia se divide en campos en correspondencia a los elementos que la componen. A este esquema se lo conoce como *formato de la instrucción*

- **Formato de la instrucción**

- *Código de operación*: especifica la operación a realizar
- *Referencia del operando fuente*: establece dónde se encuentre el/los operando/s de entrada
- *Referencia del operando resultado*: establece dónde almacenar el resultado
- *Referencia de la siguiente instrucción*: le dice a la CPU donde buscar la siguiente instrucción

- **Ciclo de instrucción**

1. Cálculo dirección instrucción **(IAC)**: determina la dirección de la siguiente instrucción a ejecutar. Se copia la dirección del PC en MAR y es puesta en el bus de direcciones
2. Captación de la instrucción **(IF)**: la CPU lee la instrucción desde memoria, se pone en el bus de datos, se copia en MBR y después se lleva a IR *[Se incrementa el PC para captar la siguiente instrucción]*
3. Decodificación de la operación en la instrucción **(IOD)**: analiza la instrucción para determinar el tipo de operación a realizar y los operandos a utilizar
4. Cálculo de la dirección del operando **(OAC)**: si la operación implica la referencia a un operando en la memoria o E/S, entonces se determina la dirección
5. Captación de operando **(OF)**: capta el operando desde memoria o E/S
6. Operación con los datos **(DO)**: realiza la operación indicada en la instrucción
7. Cálculo dirección resultado **(OAC)**
8. Almacenamiento de operando **(OS)**: escribe el resultado en memoria o lo saca por E/S

- **Modos de direccionamiento (MDDs)**

Objetivos:
- disminuir la cantidad de bits x instrucción
- lograr un manejo más eficiente de los datos (por ej. con arreglos)
- tienen la ventaja de que las direcciones pueden no conocerse hasta que se ejecuta el programa

MDDs

- **Inmediato**: el operando está presente en la propia instrucción. No requiere acceso a memoria
- **Directo**: el campo de direcciones contiene la dirección efectiva del operando. 1 acceso a mem.
- **Indirecto**: el campo de direcciones referencia la dirección de una palabra de memoria que contiene la dirección efectiva del operando. 2 accesos a mem. (1 para captar la dirección y otra para captar el valor)
- **x Reg**: similar al directo, pero se referencia un registro. No requiere acceso a memoria
- **Indirecto x Reg**: similar al indirecto, pero el campo de direcciones referencia a un registro. 1 acceso a mem.
- **x desplazamiento (OFFSET)**: requiere 2 campos de direcciones, al menos uno explícito. Usado para desplazamiento relativo, desplazamiento con registro base, e indexado
- **Desde la Pila (stack)**: el puntero de pila se mantiene en un registro, así las referencias a la pila son direcciones de acceso indirecto x Reg.

$> REPRESENTACIÓN DE DATOS

Las computadoras almacenan datos e instrucciones en memoria. Para esto, utilizan el sistema binario. La razón por la cual se utiliza binario es porque el dispositivo se encuentra en uno de dos estados posibles (0 o 1), con lo cual es más fácil identificar el estado del mismo.

- TIPOS DE DATOS: las computadoras manejan 4 tipos básicos de datos binarios:

- Números enteros con/sin signo
- Números decimales codificados en binario (BCD) + hexadecimales codificados en binario (BCH)
- Números reales(|R) con signo
- Caracteres

TEOREMA FUNDAMENTAL DE LA NUMERACIÓN

NÚMERO = $x_2.b^2 + x_1.b^1 + x_0.b^0$…

x = dígito del nº
b = base (decimal, binario, etc.)

RANGO: diferencia entre el número mayor y el menor

RESOLUCIÓN: diferencia entre dos números consecutivos

ERROR ABSOLUTO (EA): diferencia entre el valor representado y el valor a representar
{***EA Máximo*** ≤ Resolución/2}

ERROR RELATIVO: EA/Nº a representar

- **Representación de números enteros:**

- Sin signo (BSS)
- Módulo y signo (BCS)
- Complemento a 1 (Ca1)
- Complemento a 2 (Ca2)
- Exceso

- **BSS**

- Con "n" bits puedo representar 2^n números distintos
- Rango = {0 ➔ (2^n-1)}

Ej: 8 bits ➔ Rango = {0 ➔ 2^8 -1}={0➔255}={0000 0000 ➔ 1111 1111}

- **BCS**

- 2^n números distintos
- 1 bit de signo (el más significativo) y n-1 bits de magnitud [0=positivo,1=negativo]
- Rango = {-(2^{n-1} -1)➔+(2^{n-1} -1)} [Con dos '0']

Ej: 8 bits ➔ Rango = {-127➔+127}={1111 1111➔0111 1111}

- **Ca1**

- 2^n números distintos
- Números positivos empiezan con '0', negativos con '1'
- Si el número es positivo, tiene su representación binaria (como siempre)
  Si es negativo se invierten todos los bits
- Rango = {-(2^{n-1} -1)➔+(2^{n-1} -1)} [Con dos '0']

Ej: 8 bits ➔ Rango = {-127➔+127}={1000 0000➔0111 1111}

- **Ca2**

- 2^n números distintos
- Números positivos empiezan con '0', negativos con '1'
- Si el número es positivo, se representa como siempre
  Si es negativo, "mirando" desde la derecha se escribe el número igual hasta el primer '1' inclusive; el resto de los bits se invierten
- Rango = {-(2^{n-1})➔+(2^{n-1} -1)} [Con un solo '0']

Ej: 8 bits ➔ Rango = {-128➔+127}={1000 0000➔0111 1111}

- **Exceso 2 $^{n-1}$**

- 2^n números distintos
- Para **obtener** el valor real del número, se le resta el exceso
  Para **escribir** un número, se lo escribe en binario y se le suma el exceso
- Rango = {($-2^{(n-1)}$)➔+($2^{(n-1)}$-1)}

---------- ***EXCESO 2 $^{n-1}$*** = valor del bit más significativo (ej: en 8 bits, **1**000 0000 = 128) ----------

Ej: 8 bits ➔ Rango = {-128➔+127}={0000 0000➔1111 1111}

Si tengo el número 0111 0000, para ***obtener el valor real*** se le resta el exceso (en este caso 128):
0111 0000 = 102 ➔ 102-128=-26
Si quisiera ***representar*** el número 13, se escribe en binario y se suma el exceso:
13 = 0000 1101 ➔ 0000 1101 + 1000 0000 (128)=1000 1101

- **BCD + BCH**

- BCH

Cada dígito hexadecimal se convierte uno a uno en binario
Un dígito hexadecimal = 4 bits

Ej: A2B = A: 1010 , 2: 0010 , B: 1011 ➔ 1010 0010 1011
1101 1111= DF

- BCD

Cada dígito decimal se convierte uno a uno en binario
Un dígito decimal = 4 bits
En representaciones con signo: ***1100 (C) = POSITIVO*** <> ***1101 (D)= NEGATIVO***
Dos ámbitos de aplicación:
- E/S y periféricos, se usa 1 byte x dígito (***desempaquetado***)
- Cálculo, 4 bits x dígito (***empaquetado***)

==> Desempaquetado

♦ *Sin signo:*
- 8 bits x dígito = 4 para el número y 4 (a la izquierda) se completan con ‘1’ (relleno)

Ej: 185 = 11110001 11111000 11110101

♦ *Con signo:*
- 8 bits x dígito = 4 para el número y 4 ‘1’ de relleno, el último dígito se rellena con el signo

Ej: - 357 = 11110011 11110101 11010111

==> Empaquetado

♦ *Sin signo:*
- 4 bits x dígito, se rellena el byte con ‘0’ al principio

Ej: 82 = 10000010 [NO SE RELLENA]
143 = 00000001 01000011 [SE RELLENA CON ‘0’]

♦ *Con signo:*
- 4 bits x dígito, se rellena con ‘0’ de ser necesario
- El signo ocupa el lugar de los 4 bits menos significativos

Ej: +82 = 00001000 00101100 (+)
- 143 = 00010100 00111101 (-)

==> *Suma en BCD*

- Suma de 2 dígitos ≤ 9

Ej: 41+24=65
0100 0001+0010 0100 = 0110 0101 = 65 [Bien]

- Suma de 2 dígitos > 9

Ej: 45+26=71
0100 0101+0010 0110 = 0110 1011 = 611 [MAL!!]

En estos casos se le suma 6 a cada dígito con acarreo, entonces:

0110 1011 + 0000 0110 = 0111 0001 = 71 [Bien]

- **Punto Flotante**

- 2^n números distintos para 'n' bits
- ***Rango mayor*** a punto fijo para igual cantidad de bits
- ***Resolución variable*** a lo largo del rango

==> *Formato*

+/- M.B $^{+/- E}$ [signo][exponente(E)][mantisa(M)]
Ej: *Mantisa*: BCS, 8 bits, fraccionaria,1 bit de signo, *Exponente*: Ca2, 4 bits, entero

Máximo positivo = 0(+) $0{,}11111111 \times 2^{0111} = +(1-2^{-8}) \times 2^{+7}$
Mínimo positivo = 0 $0{,}00000001 \times 2^{1000} = +(2^{-8}) \times 2^{-8}$

Máximo negativo = 1(-) $0{,}000000001 \times 2^{1000} = -(2^{-8}) \times 2^{-8}$
Mínimo negativo = 1 $0{,}11111111 \times 2^{0111} = -(1-2^{-8}) \times 2^{+7}$

==> *Normalización*
Se introduce con el objetivo de tener un único par de valores de mantisa y exponente para un nº

- La mantisa fraccionaria se define de la forma: **0,1**xxxxxxx, donde las 'x' corresponden al nº. Todas las mantisas deben empezar con 0,1
- Dado el caso de querer *interpretar* un nº con mantisa normalizada, éste debe comenzar con '1', por ej: 1001; no es posible interpretarlo si esto no se cumple, ej: 0101

 # Bit implícito
Con bit implícito no es necesario que el nº contenga la normalización en sí mismo. Simplemente se le agrega '**,1**' adelante. El bit implícito cuenta a la hora de interpretar el nº!

Ej: Nº = 0(+)00101 ➔ + 0,100101

==> *Estándar IEEE 754*

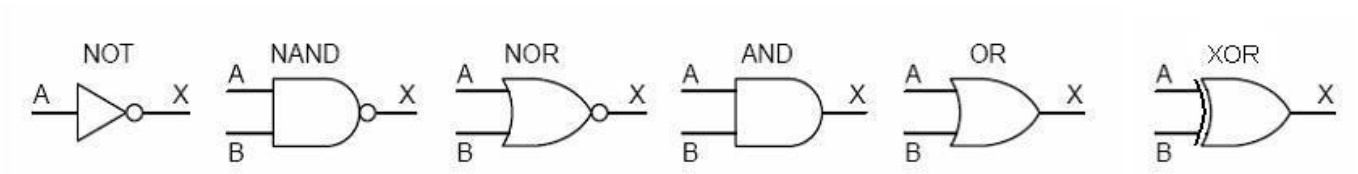
 # Formato:
- Mantisa fraccionaria normalizada como '**1,**xxxxxxxx'
- Exponente representado en exceso $2^{n-1}-1$

 # Simple precisión: 1b signo + 8b exp [exceso 127] + 23b mantisa = 32b
 # Doble precisión: 1b signo + 11b exp [exceso 1023] + 52b mantisa = 64b

 # Casos especiales:
- E = 255/2047 & M <> 0 ➔ NaN
- E = 255/2047 & M = 0 ➔ ∞
- E = 0 & M = 0 ➔ 0
- E = 0 & M <> 0 ➔ Denormalizado

$> LÓGICA DIGITAL

- Un circuito digital presenta 2 valores lógicos (1,0 :: true, false)
- Una ***compuerta*** es un circuito electrónico que produce como señal de salida el resultado de una operación booleana sencilla entre las señales de entrada
- ***COMPUERTAS BÁSICAS***: NOT, NAND, NOR, AND, OR, XOR



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

- **Leyes de De Morgan**

$$\overline{A.B}=\overline{A}+\overline{B} \qquad \overline{A+B}=\overline{A}.\overline{B}$$

- **Suma de productos**

Dada una tabla correspondiente a un circuito 'X':

A	B	X
0	0	1
0	1	1
1	0	0
1	1	1

Para averiguar la lógica del circuito, basta emplear el siguiente método: cada '0' representa una entrada negada; cada '1' una entrada sin negar. Entonces se escriben los productos de cada '1' correspondiente a la salida 'X' y se los suma

==> $\bar{A}.\bar{B} + \bar{A}.B + A.B = F$

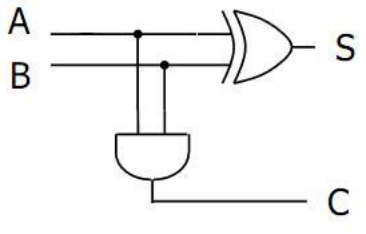
- **Máscaras**

- *AND* ➔ con una entrada que sea '0', SALIDA = 0
- *OR* ➔ con una entrada que sea '1', SALIDA = 1
- *XOR* ➔ entrada '1' = invierte el valor de la entrada
  entrada '0' = deja la entrada como estaba

- CIRCUITOS COMBINACIONALES

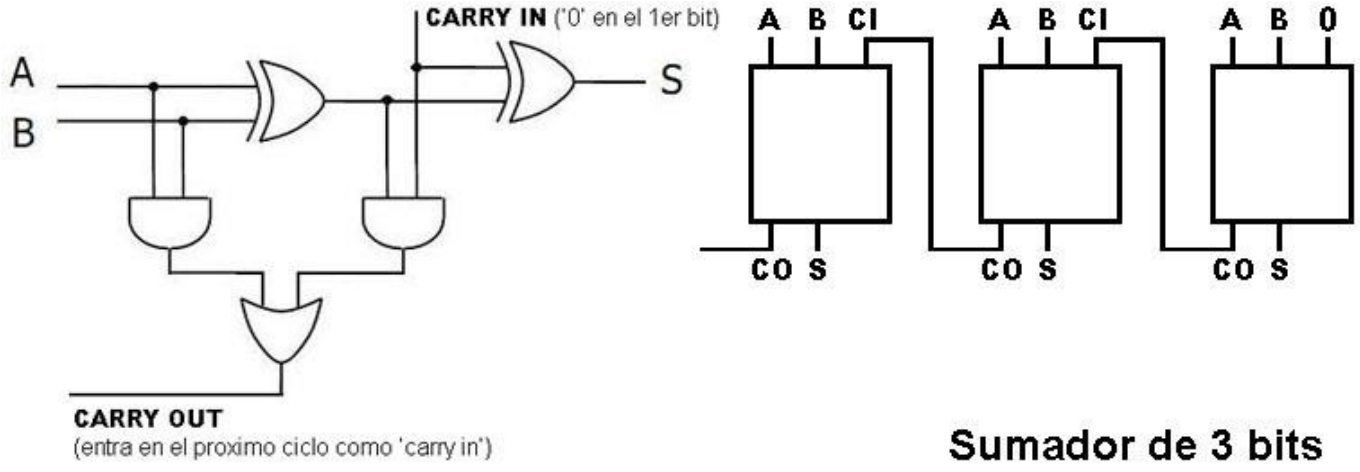
- Un circuito combinacional es un conjunto de compuertas interconectadas, cuya salida, en un momento dado, es función solamente de la entrada en ese instante
- Si cambia la entrada, cambia la salida
- Los valores pasados de las entradas no influyen en los valores de la salida

- **Semi-sumador y sumador completo**



Semi-Sumador (Half Adder)

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



- **Multiplexores**: un multiplexor conecta varias entradas a una única salida. En un momento dado, se selecciona una de las entradas para que pase a la salida. Se usan en circuitos digitales para controlar el enrutamiento de señales y datos, por ejemplo en el PC

- **Decodificadores**: un decodificador contiene varias líneas de salida, con una sola de ellas seleccionada en un instante dado, dependiendo del patrón de líneas de entrada. En general tienen 'n' entradas y '2^n' salidas. Se usan por ejemplo en la decodificación de direcciones.

- CIRCUITOS SECUENCIALES

- Tienen la característica de "almacenar" valores lógicos internamente
- Estos valores se almacenan aunque las entradas no estén
- Las salidas dependen tanto de las entradas como del estado interno del circuito

- **Biestables (FLIP-FLOP)**

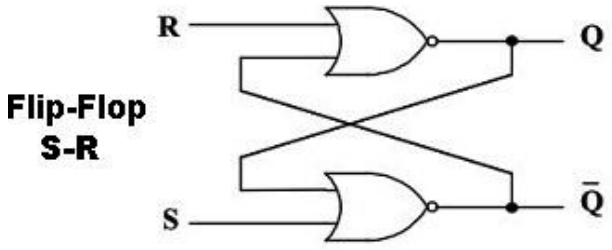
- Es un dispositivo con 2 estados. En ausencia de entrada "recuerda" el último estado
- El biestable tiene dos salidas que se complementan (Q y ~Q)

- *CLASIFICACIÓN:*
- Según la manera en que las salidas responden a las entradas: SR, J-K, D, T
- Respecto del instante en que pueden cambiar las salidas***:***
  - ⇨ *Asincrónicos* = cuando en la entrada se establece una combinación, las salidas cambian
  - ⇨ *Sincrónicos* = la presencia de un CLK determina cuando cambian las salidas

- **Reloj (Clock-CLK):** es una señal de tiempo precisa que determina cuando se producen eventos. A veces es necesario que los procesos se ejecuten simultáneamente o con cierto orden.

- **Tipos de Flip-Flops**

- ***S-R***
- El circuito consta de dos entradas, S (set) y R (reset), y dos salidas, Q y ~Q, y consiste en dos puertas NOR conectadas en retroalimentación
- 'S' determina el valor de la salida Q_{n+1}

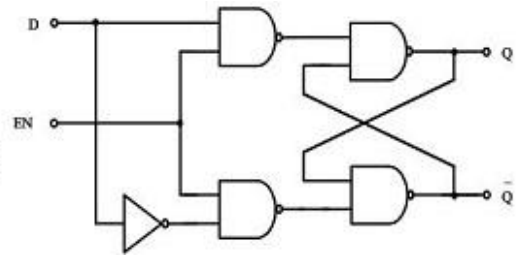


S	R	Q_{n+1}	
0	0	Q_n	==> mantiene estado
0	1	0	
1	0	1	
1	1	Error!	

- ***D***

- El biestable D permite solucionar el problema de los biestables S-R usando un inversor, y garantizando que las entradas que no son del reloj sean opuestas
- La salida del biestable D es siempre igual al valor más reciente aplicado a la entrada. Por lo tanto, recuerda y produce la última entrada

Flip-Flop D
[tipo de S-R]
REGISTRO DE 1 BIT

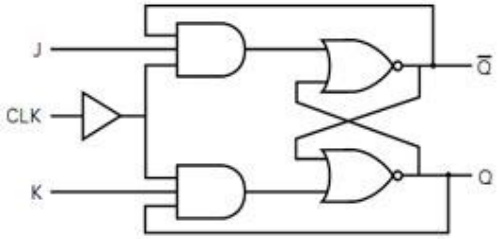


D	Q_{n+1}
0	0
1	1

- ***J-K***

- En el biestable J-K todos los valores de entrada son válidos
- 'J' determina el valor de la salida Q_{n+1}

Flip-Flop J-K



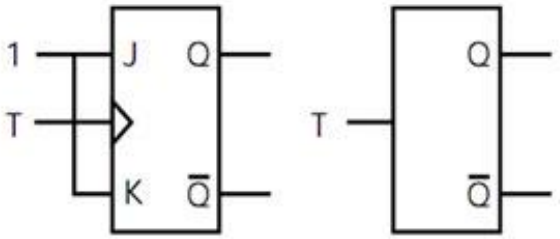
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

==> niega estado anterior

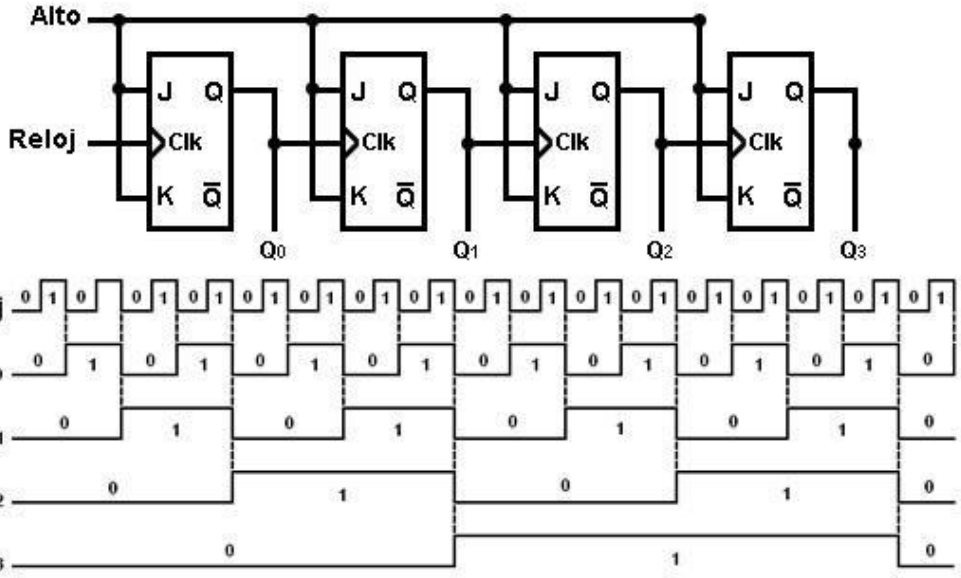
- ***T***

- Mantiene su estado o lo cambia dependiendo del valor de T cada vez que se activa
- Se pueden enlazar 'n' biestables T para crear un contador de hasta 2^n-1

Flip-Flop T
[tipo de J-K]
CONTADOR



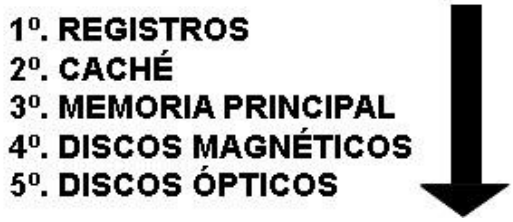
T	Q_{n+1}
0	Q_n
1	$\overline{Q}_n$



Contador asíncrono de 4 bits con Flip-Flops J-K

$> MEMORIAS

- JERARQUÍA DE MEMORIA



- CLASIFICACIÓN POR CARACTERISTICAS

⇒ **Método de almacenamiento**
- Semiconductoras: RAM, ROM
- Magnéticas: discos mag.
- Ópticas: CD, DVD

⇒ **Método de acceso**
- Acceso aleatorio: RAM, ROM
- Acceso secuencial: cintas
- Acceso directo: discos (mag. y opt.)
- Acceso asociativo: caché

⇒ **Duración de la información**
- Volátiles: RAM
- No volátiles: discos, cintas
- Permanentes: ROM, EPROM

⇒ **Modo de acceso**
- Por palabra: memoria principal
- Por bloque: discos, caché

- TIPOS DE ACCESO

⇒ **Acceso secuencial**

La memoria se organiza en unidades de datos llamadas “registros”. El acceso se realiza con una secuencia lineal específica. Se utiliza un mecanismo de lectura/escritura compartida, que debe ir trasladándose desde su posición actual a la deseada, pasando y obviando cada registro intermedio. El tiempo de acceso es muy variable

⇒ **Acceso directo**

El método de acceso directo también tiene asociado un mecanismo de lectura/escritura. Sin embargo, los bloques o registros tienen una dirección única basada en su dirección física. El acceso se lleva a cabo mediante un acceso directo a una vecindad dada, seguido de una búsqueda secuencial hasta alcanzar la posición buscada. El tiempo de acceso también es muy variable

⇒ **Acceso aleatorio**

Cada posición direccionable de memoria tiene un único mecanismo de acceso, cableado físicamente. El tiempo para acceder a una posición dada es constante e independiente de la secuencia de accesos previos. Cualquier posición puede seleccionarse aleatoriamente y puede ser direccionada y accedida directamente

⇒ **Acceso asociativo**

Es una memoria del tipo de acceso aleatorio, que permite hacer una comparación de ciertas posiciones de bits dentro de una palabra buscando que coincidan con ciertos valores dados, y hacer esto para todas las palabras simultáneamente. Entonces, una palabra es recuperada basándose en una porción de su contenido, en lugar de su dirección. Cada posición tiene su propio mecanismo de direccionamiento y el tiempo de recuperación de un dato es constante.

- MEMORIA PRINCIPAL

- **Memorias Semiconductoras**

En computadores antiguos, la forma más común de almacenamiento de acceso aleatorio para la memoria principal consistía en una matriz de pequeños anillos ferromagnéticos denominados *núcleos*. Hoy en día es casi universal el uso de memorias semiconductoras para la memoria principal.

➢ **RAM** (Random-Access Memory)

- **Propiedades**:
  - ♦ se puede escribir y leer datos rápidamente
  - ♦ Read/Write se ejecutan mediante señales eléctricas
  - ♦ es volátil, debe estar continuamente alimentada o se pierden los datos
  - ♦ almacenamiento temporal

- **Tipos:**
  - ♦ *Dinámicas* ***(DRAM)***: es un tipo de memoria cuyas celdas se implementan utilizando un transistor y un condensador. El condensador mantiene el bit de información como cargas (0 o 1), y el transistor actúa como un conmutador que permite a los circuitos del chip leer el condensador o cambiar su estado. Requiere refrescos periódicos para mantener los datos
  - ♦ *Estáticas* ***(SRAM)***: se implementa con biestables (flip-flops). Una RAM estática mantendrá sus datos mientras se le suministre corriente; no necesitan refresco. Son menos densas y algo más rápidas que las dinámicas. Se utilizan, por ejemplo, en registros y caché

- **Otros tipos de RAM**:
  - ♦ ***EDRAM***: igual que DRAM, pero con una pequeña caché SRAM
  - ♦ ***CDRAM***: igual que EDRAM, pero con una caché más grande que también puede usarse de buffer
  - ♦ ***SDRAM***: RAM síncrona. Emplea un modo de ráfagas para eliminar los tiempos de establecimiento de dirección y de precarga de las líneas de fila y de columna posteriores al primer acceso. Se puede secuenciar la salida de una serie de bits de datos. Utiliza un registro de modo para especificar la longitud de la ráfaga
  - ♦ ***RDRAM***: utiliza un bus especial que entrega direcciones e información de control, usando un protocolo asíncrono orientado a bloques. Se obtienen velocidades de hasta 500mb/s
  - ♦ ***RAMLINK***: representa el cambio más radical respecto de la DRAM convencional. Se centra en la interfaz procesador/memoria, en lugar de en la arquitectura interna de los chips. RamLink es una interfaz de memoria con conexiones punto a punto dispuestas en un anillo. Intercambia datos en forma de paquetes. Proporciona una arquitectura expandible

➢ **ROM** (Read-Only Memory)

- **Propiedades**:
  - ♦ se puede escribir solo una vez
  - ♦ presenta la ventaja de que los datos están permanentemente en memoria principal

- **Aplicaciones**:
  - ♦ subrutinas de biblioteca para funciones de uso frecuente
  - ♦ programas del sistema
  - ♦ tablas de funciones

- **Otros tipos de ROM**
  - ♦ ***PROM***: igual que una ROM convencional, pero se puede escribir, posteriormente a la fabricación, eléctricamente
  - ♦ ***EPROM***: igual que PROM, pero se puede borrar, por radiación UV, y escribir varias veces
  - ♦ ***EEPROM***: se puede escribir sin borrar todo el contenido anterior
  - ♦ ***FLASH***: utiliza borrado eléctrico rápido, que permite borrar por bloques

⇒ **Organización**

- ***Todas las celdas de memoria comparten 3 propiedades***:
  - ♦ 2 estados estables (1, 0)
  - ♦ se puede escribir en ellas, al menos una vez
  - ♦ se pueden leer para conocer el estado

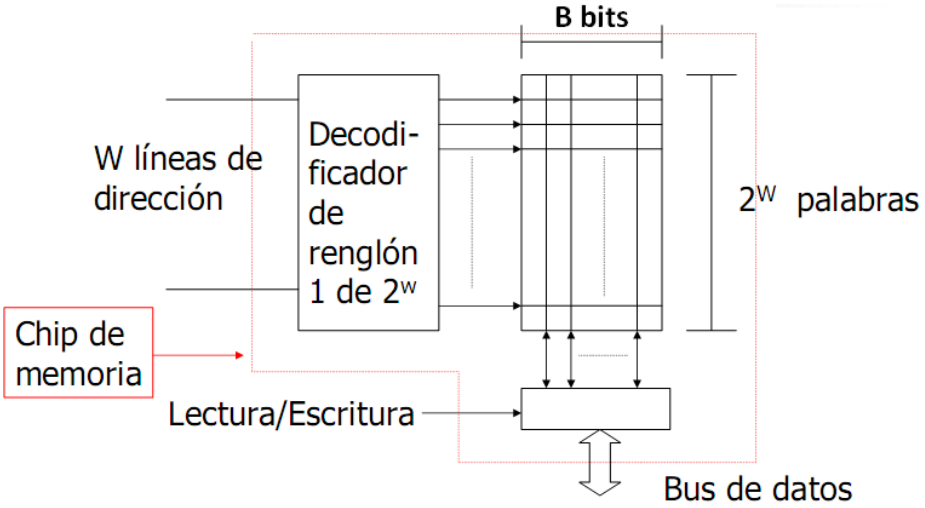
- ***Memorias más grandes que un registro:***
  - ♦ direccionan palabras individuales
  - ♦ cada chip contiene un arreglo de celdas de memoria

- ***Enfoque 2D y 2½D***

⇒ **2D**

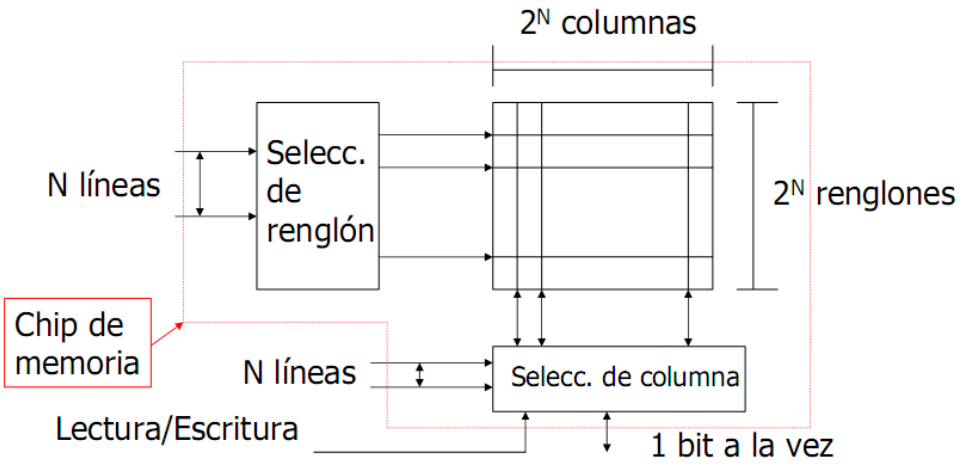
El arreglo está organizado en 2^W palabras de B bits cada una. Cada línea horizontal (una de 2^W) se conecta a cada posición de memoria, seleccionando un renglón. Las líneas verticales conectan cada bit a la salida. El decodificador que está en el chip, tiene 2^W salidas para W entradas (bits del bus de direcciones)

La problemática que tiene este tipo de disposición de memoria es que es físicamente muy difícil de implementar con una cantidad de registros grandes. Esto se debe a que, si pensamos en este tipo de organización como una lista de celdas, esta estructura es muy frágil y requiere mucho espacio a lo largo del soporte de la memoria. Hay que mencionar que, si bien puede haber muchas celdas, el tiempo de acceso a ellas con esta disposición es casi constante

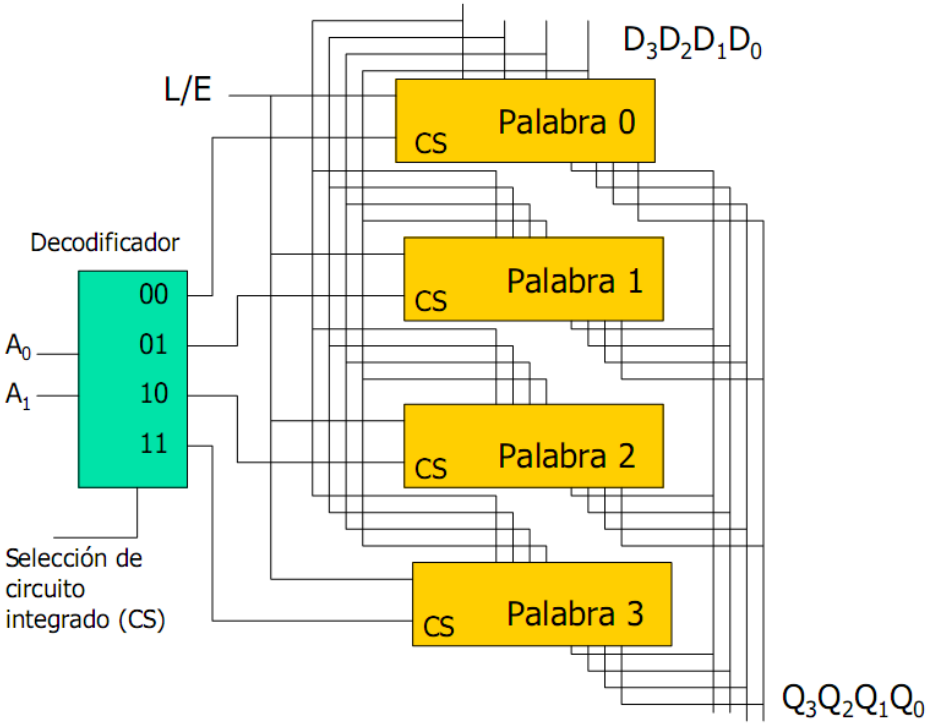


⇒ **2½D**

Dadas las dificultades a las que nos enfrentamos en el modo de direccionamiento 2D, y teniendo en cuenta el coste (económico, y técnico) que tiene su implementación, se ha optado por otro tipo de modo de direccionamiento que satisfagan requerimientos diferentes a los previamente satisfechos. Así surge el modo de direccionamiento 2½D, el cual se organiza de la siguiente manera: una memoria que utilice este tipo de direccionamiento, tendría una estructura compuesta por varios chips, los cuales tendrían la responsabilidad de retornar el valor de un bit, el cual se utilizaría para formar el valor de la palabra de N bits que se quiera leer, o escribir en la memoria.
Para ello, sería necesario proveerle al chip la posición a la que quiero acceder. En consecuencia, cuando a la memoria se le pide acceder a una dirección, la misma se divide en dos partes (a través de decodificadores), una que le indica a la memoria, que “renglón” se quiere leer (con la cual se ubica a todos los chips necesarios para la reconstrucción de ese “renglón”) y otra, que indica en que parte de ese chip se debe buscar el bit correspondiente a una fracción del “renglón” a devolver (“columna”). Una vez identificado cada bit de la palabra, se reconstruye la misma, y se la ubica en la interfaz de salida de la estructura.
A pesar de que obtener una palabra de esta memoria requiere más procesamiento, puede extender su capacidad inmensamente con respecto al direccionamiento 2D. Ahora bien, a consecuencia de este gran incremento en la capacidad de almacenamiento de una memoria con este modo de direccionamiento, y debido al hecho de que por una cuestión de costes, estas se construyen con transistores, estas memorias, si bien son mucho mas baratas que las 2D, también son considerablemente mas lentas. Sobre todo teniendo en cuenta el procesamiento extra que conlleva decodificar una dirección de memoria en particular



⇒ **Implementando una memoria**



⇒ **<u>Memoria caché</u>**

Es una memoria construida en base a memorias SRAM de alta velocidad, que está conectada al procesador y a la memoria principal del sistema.
El objetivo de la caché es lograr que la velocidad de acceso a memoria sea lo más rápida posible. Para esto, la caché contiene una copia de bloques de la memoria principal. Así, cuando el procesador intenta leer una palabra de memoria, se hace una comprobación para determinar si la palabra está en caché. Si es así, se entrega dicha palabra al procesador a gran velocidad; si no, un bloque de memoria principal se transfiere a la caché, y una palabra es entregada al procesador

EFECTIVIDAD = frecuencia de aciertos = número de veces que la caché acierta direcciones
- *<u>Acierto</u>*: sucede cuando los datos que necesita el procesador están en la caché
- *<u>Fallo</u>*: ocurre cuando los datos no están en caché y se deben obtener de la memoria principal

==========> **<u>PRINCIPIO DE LOCALIDAD</u>** <==========

El principio de localidad asegura que los programas acceden únicamente a una porción relativamente pequeña de su espacio de direccionamiento durante un corto lapso de tiempo. Por lo tanto, existe una fuerte tendencia a que los accesos futuros sean en el mismo bloque, o uno contiguo.

Localidad TEMPORAL: si se hace referencia a un objeto, existe una cierta tendencia a volver a referenciarlo en un corto espacio de tiempo (bucles en un programa o llamadas a subrutinas)
Localidad ESPACIAL: si se hace referencia a un objeto, también tenderán a ser referenciados los demás objetos que están ubicados en direcciones próximas a éste (acceso a vectores de datos o a la memoria de instrucciones)

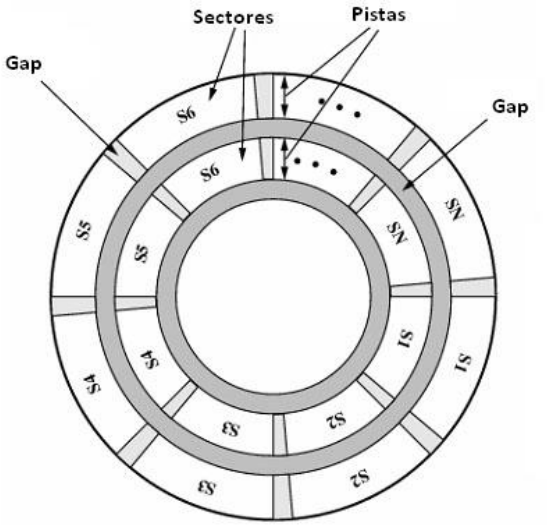
==========

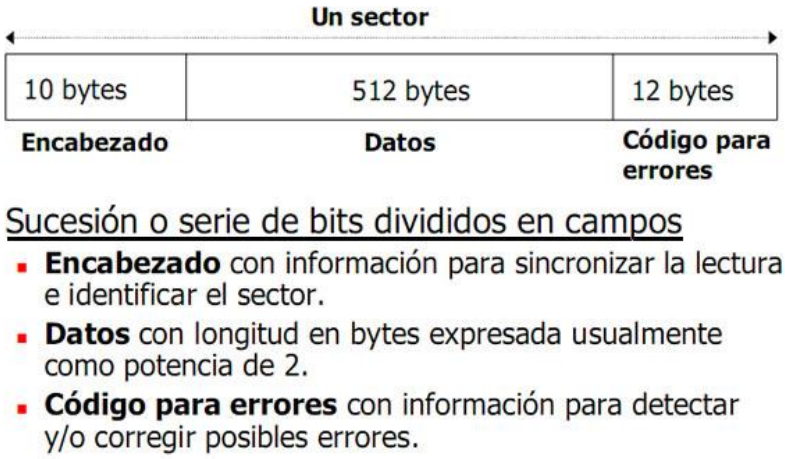
- MEMORIA SECUNDARIA

- **Disco magnético (DM)**

Un DM es un plato circular construido con metal o plástico, cubierto por un material magnetizable. Los datos se graban y se recuperan a través de una bobina llamada ***cabezal***.
Los DM son los dispositivos más comúnmente usados para almacenamiento secundario. Se caracterizan por tener un ***acceso directo*** a la información, tiempo de acceso relativamente pequeño, una gran capacidad de almacenamiento y un bajo coste

Un DM está formado por uno o varios ***platos*** que giran simultáneamente, teniendo cada plato sus propios ***cabezales de L/E***. La información se almacena distribuída en ***pistas*** concéntricas. Cada pista, a su vez, se encuentra dividida en ***sectores*** que forman ***bloques***. Un bloque es la unidad de transferencia desde y hacia el disco. Tanto las pistas como los sectores se separan entre si por ***intrapistas vacías (gaps)***, para evitar imposiciones de precisión ilógicas.





- *Características posibles*:
  - ♦ Disco removible o fijo
  - ♦ Uno o más platos
  - ♦ Platos de simple o doble lado
  - ♦ Cabeza fija o móvil
  - ♦ Mecanismos de cabeza: contacto (floppy), distancia fija, aerodinámica (Winchester)

- *Estructura de un DISCO DURO*:
  - ♦ Múltiples platos
  - ♦ Platos giran constantemente
  - ♦ Una cabeza por cara
  - ♦ Todas las cabezas se mueven a la vez
  - ♦ Las pistas alineadas en cada plato forman cilindros
  - ♦ Los datos se almacenan por cilindros

- *Tiempos*

Tiempo de seek (búsqueda) = mover al cilindro (o pista) correcto
Tiempo de latencia (por rotación) = esperar que el sector "pase" por debajo de la cabeza
Tiempo de acceso = T. seek + T. latencia
TIEMPO TOTAL = T. de acceso + T. de transferencia de datos

- *Cálculo de la capacidad del disco*

$$\text{CAPACIDAD} = \frac{\text{bytes}}{\text{sector}} \times \frac{\text{sectores}}{\text{pista}} \times \frac{\text{pistas}}{\text{superficie}} \times \text{nº de superficies}$$

- ***RAID***

⇒ *Características*:
- ♦ NO es una jerarquía
- ♦ Es un conjunto de unidades físicas de disco, vistas por el SO como una única unidad lógica
- ♦ Los datos se distribuyen a través de las distintas unidades físicas
- ♦ La capacidad de los discos redundantes se usa para almacenar información de paridad que garantice la recuperación de los datos en caso de fallo de disco.

⇒ *RAID 0:* los datos del usuario y del sistema están distribuidos a lo largo de todos los discos del conjunto. La ventaja es que si hay pendientes dos peticiones diferentes de E/S para dos bloques de datos distintos, entonces es muy probable que estos últimos estén en distintos discos, por lo que ambas peticiones se pueden emitir en paralelo, reduciendo el tiempo de cola de E/S. Los datos se organizan en forma de ***tiras*** a través de los discos disponibles, formando ***"franjas"***.

⇒ *RAID 1:* se duplican los datos (cada disco tiene un disco espejo [***MIRRORING***]). Aspectos positivos: una petición de lectura puede ser servida por cualquiera de los discos que contienen los datos pedidos; una petición de escritura requiere que las dos tiras correspondientes se actualicen, cosa que se puede hacer en paralelo; la recuperación tras un fallo es sencilla, se acceden a los datos desde la segunda unidad.

⇒ *RAID 2:* usa ***acceso paralelo***. En un conjunto de acceso paralelo, todos los discos miembros participan en la ejecución de cada petición de E/S. Descompone los datos en tiras, a menudo muy pequeñas. Utiliza discos de paridad para almacenar códigos de corrección de errores.

⇒ *RAID 3:* similar a RAID 2, también usa acceso paralelo. Utiliza solo un disco redundante. Distribuye los datos en pequeñas tiras. Además, no usa código de corrección de errores, calcula un bit de paridad para cada conjunto de bits.

⇒ *RAID 4:* usa ***acceso independiente***. En un conjunto de acceso independiente, cada disco opera independientemente, con lo que peticiones de E/S separadas se atienden en paralelo. Usa tiras de datos relativamente grandes. Calcula una tira de paridad bit a bit a partir de las tiras de cada disco, y se almacena en la tira del disco de paridad.

⇒ *RAID 5:* similar a RAID 4, también usa acceso independiente. Distribuye las tiras de paridad a lo largo de todos los discos, evitando un potencial cuello de botella.

⇒ *RAID 6:* usa doble paridad, almacenada en bloques separados en distintos discos. Proporciona una disponibilidad de los datos extremadamente alta.

- **CD-ROM**

⇒ *Características:*
- ♦ Almacena los datos como 'pits'
- ♦ L/E por láser (L=reflejado)
- ♦ Capacidad = 650mb
- ♦ Acceso directo y lento
- ♦ Removible y robusto

- **Cinta magnética**

⇒ *Características:*
- ♦ Cinta de plástico flexible cubierta por óxido magnético
- ♦ Acceso en serie
- ♦ Lenta y barata
- ♦ Se usa en backups y archivo

$> PERIFÉRICOS

Se denominan periféricos a las unidades o dispositivos a través de los cuales el computador se comunica con el mundo exterior.
Los dispositivos de E/S transforman la información externa en señales codificadas, permitiendo su transmisión, detección, interpretación, procesamiento y almacenamiento de forma automática. Los ***dispositivos de Entrada*** transforman la información externa según alguno de los códigos de E/S. En un ***dispositivo de Salida*** se efectúa el proceso inverso.

- **Teclado**

Se utiliza para introducir órdenes e información al computador. Cuando una tecla es pulsada, se realiza la conversión de su posición, en una matriz de contactos, a un código alfanumérico. Seguidamente se envía el código al computador.

- **Monitor**

Es un dispositivo de salida que, mediante una interfaz, muestra los resultados del procesamiento de una computadora. El más difundido es el de Tubo de Rayos Catódicos (TRC) que puede ser monocromático o color, existen también de Cristal Liquido o Plasma.

➢ ***Clasificación según:***

⇒ *Los colores que muestran:*

- **Monocromáticos**: Pueden mostrar la información en blanco y negro, en ámbar o en verde.
- **A color**: Pueden mostrar la información con las combinaciones del rojo, del azul y del verde (RGB).

⇒ *Como representan la información:*

- **De Caracteres**: Ya fuera de uso. Son aquellos monitores que poseen dos memorias para representar únicamente caracteres: una del tipo RAM donde se guarda la información de cada carácter, con su atributo; y una memoria del tipo ROM, donde se almacena los patrones de cada carácter almacenado
- **Gráficos**: Son aquellos monitores que utilizan un trazado de líneas para mostrar la información. Poseen solo una memoria del tipo ROM, donde se almacenan la intensidad, el color, y otros atributos posible para cada píxel representando en la resolución de pantalla. Cada píxel es representado por bits en memoria

⇒ *Tamaño de pantalla* (en pulgadas)
⇒ *Resolución de pantalla* (en pixeles): número de puntos de imagen en pantalla

- **CGA** (640 * 200)
- **VGA** (640 * 480)
- **SVGA** (1024 * 768)

➢ ***Valores***

- **PUNTOS POR SEGUNDO** = *resoluciónVertical x resoluciónHorizontal x cuadros/segundo*
- **Monitor alfanumérico** = 7bits x carácter + 3bits de propiedades (opcional)
- **Monitor gráfico** = 1bit x píxel en *MONOCROMO* :: 3bytes x píxel en *TRUE COLOR*

- **Impresora**

Las impresoras son periféricos que escriben la información de salida sobre papel.

➢ ***Clasificación según:***

⇒ *La forma de imprimir*:

- **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 muy rápidas.
- **De página**: Imprimen toda una pagina en una sola pasada, aunque internamente imprimen línea por línea.

⇒ *El mecanismo 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.
    - *Margarita, bola, cilindro, etc.*
    - *Cinta*
    - *Tambor*

- **Térmicas**: Son aquellas que imprimen mediante la transferencia de calor al cabezal, a través de una matriz de pequeñas resistencia, a las que al pasar corriente eléctrica por ellas, se calientan formando los puntos. Estas impresoras pueden ser de caracter (las líneas se imprimen con un cabezal móvil) o de línea (contiene tantas cabezas como caracteres a imprimir por línea, son más rápidas).
- **Chorro a tinta**: Son aquellas que imprimen mediante la carga de gotas de tinta, por medio de electricidad estática. El caracter se forma con la tinta que cae al papel según la carga estática que posea la misma. Cuando finaliza, las gotas que sobran se desvían hacia un camino de retorno al cartucho.
Son modelos bidireccionales y muy utilizados por su velocidad y gran calidad de impresión.
- **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 tóner.
Poseen un sistema similar a las fotocopiadoras, con una elevada velocidad y calidad de impresión

- **Modem (MOdulador - DEModulador)**

Es un dispositivo, externo o interno, utilizado para la comunicación entre computadoras a través de líneas telefónicas.
El módem convierte las señales digitales (‘0’ y ‘1’) en analógicas (tonos de audio) (*MODULARIZACIÓN*), y las envía por la línea de teléfono a la que deben estar conectados el emisor y el receptor. Cuando la señal llega a su destino, otro módem se encarga de reconstruir la señal digital primitiva (*DEMODULARIZACIÓN*), de cuyo proceso se encarga la computadora receptora. Además, un módem está programado para ser tolerantes a errores de transmisión

Tasa Bits/seg (bps) = número de bits enviados por segundo.
Tasa Baudio (baud rate) = número de cambios de señal por segundo

$> OTROS CONCEPTOS

- **Assembly**

Assembly, o lenguaje de ensamble, es un lenguaje que, en principio, define un mapeo directo de códigos (codops, que en realidad son una secuencia de bits con un significado en particular para la máquina) a instrucciones comprensibles por un humano.
El programa encargado de tomar programas escritos en assembly y generar los códigos que la computadora entenderá se denomina assembler o ensamblador

- **Una familia de computadoras**

Rompiendo con el método de la industria, IBM creó una serie de ordenadores de pequeños a grandes y de alto a bajo rendimiento, todos ellos usando el mismo conjunto de comandos. Esto *permitía a los clientes usar modelos más baratos y después ampliarlos a sistemas más potentes conforme se incrementaban sus necesidades sin pasar por el gasto excesivo de reescribir su software*. IBM hizo el primer uso comercial de la tecnología de microcódigo para lograr esta compatibilidad, empleándola en todos sus modelos menos los modelos más potentes