

Organización del Computador

Unidad 1: Introducción

Computadora: máquina (conjunto de elementos mecánicos, eléctricos y electrónicos) capaz de procesar gran cantidad de información a alta velocidad.

Clasificación de las computadoras:

1. Según el tipo de información que manejan:

- ◆ Analógicas: operan con voltajes o intensidad de corriente.
- ◆ Digitales: operan con información binaria. Tienen mayor precisión.

2. Según las generaciones:

- ◆ 1° generación: constituida por válvulas (dispositivos electrónicos consistentes en tubos que requerían mucha energía).
- ◆ 2° generación: aparición del transistor (miniaturización). Aparición de los compiladores.
- ◆ 3° generación: circuitos integrados (muchos componentes en una misma pastilla). Aparición de los Sistemas Operativos.
- ◆ 4° generación: circuitos con alto grado de integración. Aparición de los lenguajes de 4° generación.
- ◆ 5° generación: permiten procesamiento en paralelo.

Arquitectura: determinan una familia de computadoras. Son las características computacionales visibles al programador, tienen un impacto directo en la ejecución de un programa.

- Microarquitectura (microprogramada – *hardwired*)
- Formato de instrucciones
- Repertorio de instrucciones
- Tipos de datos
- Direccionamiento de memoria
- Tamaño de la palabra
- Registros
- Interrupciones
- Formato de memoria

Organización: implementación de la arquitectura. Ejemplos: interfaces entre CPU/memoria, tecnología de memoria. Está relacionada con las unidades operativas y sus interconexiones.

Modelo de capas:

Software	Nivel 6	Lenguaje orientado a problema
	Nivel 5	Lenguaje ensamblador
	Nivel 4	Sistema operativo
	Nivel 3	ISA (<i>Instruction Set Architecture</i>)

Hardware	Nivel 2	Microarquitectura
	Nivel 1	Lógica digital (circuitos)
	Nivel 0	Dispositivos (transistores)

Tipos de computadoras:

- Computadoras de mano: *smartphones*, PDAs.
- Computadoras portátiles: *notebook*, *netbook*, *tablet PC*.
- Estaciones de trabajo: PC, consolas de juego.
- Mini computadores: servidores.
- Macro computadoras: mainframes.
- Súper computadoras: *Deep Blue*.

Unidad 2: Máquina elemental

Conceptos preliminares

Registro: memoria muy rápida que almacena una cierta cantidad de bits.

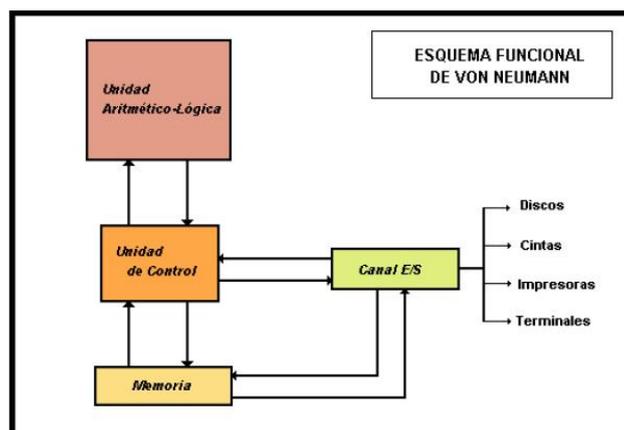
Compuerta: circuitos electrónicos bi-estables unidireccionales (sólo permiten dos estados, abierto / cerrado). Son la base del hardware para las computadoras. Para intercambiar información entre dos registros la misma debe viajar por el bus, y esto se logra a través de las compuertas. Cuando las mismas se abren la información está disponible en el bus, cuando se cierra desaparece. No pueden existir dos compuertas abiertas al mismo tiempo.

Arquitectura Von Neumann

Es del año 1945. Promueve la mecanización del tratamiento digital de la información. Tiene dos conceptos fundamentales:

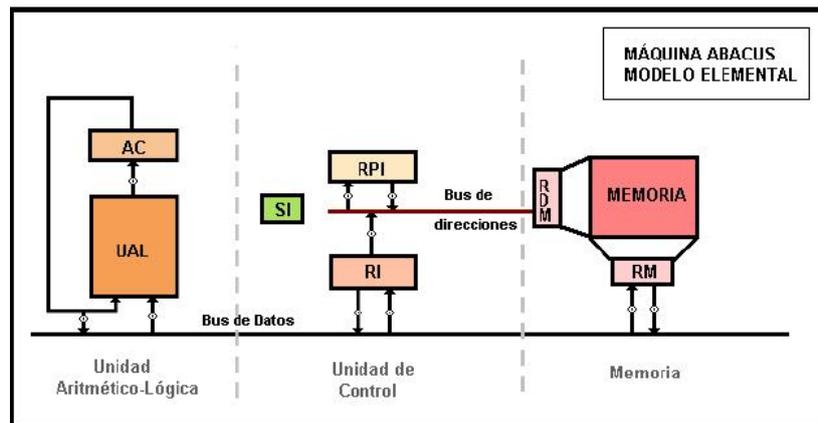
- Principio del programa almacenado: el computador debe tener el programa almacenado en su propia memoria. El uso de la memoria es para el almacenamiento de datos y las instrucciones del programa.
- Principio de ruptura de secuencia: las operaciones de decisión lógica deben ser automáticas, dotando a la máquina de una instrucción llamada “salto condicional”.

La computadora con arquitectura Von Neumann tiene cinco partes:



- **Memoria:** está dividida en celdas cuyo contenido es variable y son identificadas con un número fijo llamado “dirección de memoria”. La capacidad total de la memoria está dada por la cantidad de celdas disponibles. Las celdas tienen datos e instrucciones (a diferencia de la arquitectura Harvard que tiene una memoria para datos y otra para instrucciones).
- **UAL:** unidad encargada de realizar operaciones elementales de tipo aritmético (sumas y restas) y lógicas (comparaciones, NOT, XOR, etc.)
- **UC:** controla y gobierna todas las operaciones (búsqueda, decodificación y ejecución de instrucciones).
- **Dispositivo de E/S:** gestiona la transferencia de información entre los periféricos y la memoria central.
- **Bus de datos:** sistema digital que transporta datos entre las distintas partes (no la almacena, solo la transmite).
 - Bus de datos: mueve la información por los componentes del hardware.
 - Bus de direcciones: ubica los datos en la memoria.
 - Bus de control: marca el estado de una instrucción.

Abacus



- Es una máquina de una sola dirección.
- El acumulador es un registro particular que alberga el primer operando y el resultado de las operaciones. Todas las operaciones se realizan “contra” el acumulador. El acumulador realiza las operaciones aritméticas, lógicas y de comparación.
- El ciclo de memoria equivale a dos impulsos de reloj.
- La UC contiene tres registros:
 - RPI: contiene la dirección de la próxima instrucción a ejecutar.
 - RI: contiene la instrucción extraída de la memoria.

CO (Código de Operación)	OP (Operando)
--------------------------	---------------

- SI: administra la apertura y cierre de compuertas.
- La memoria tiene dos registros:
 - RDM: contiene la dirección de la celda de memoria.
 - RM: contiene el dato de la celda de memoria.
- Tamaño RPI = tamaño RDM = tamaño OP = cantidad de celdas direccionables
- Tamaño AC = tamaño RI = tamaño RM = longitud de instrucción = longitud de celda
- Maneja una aritmética de Binario Punto Fijo con signo de 16 bits.

- Las operaciones pueden ser entre registros, entre registro y dato inmediato, o entre registro y dato en memoria.

Unidad 3: Componentes de un computador

Microprocesador

	RISC (Reduced Instruction Set Computer)	CISC (Complex Instruction Set Computer)
Instrucciones	Simples, pocas, de igual tamaño, formatos simples	Complejas, muchas, de tamaño variable, formatos complejos
Registros	Muchos, de uso general	Pocos, de uso específico
Direccionamiento	Pocos mecanismos	Muchos mecanismos
Tipos de datos	Pocos, simples	Muchos, complejos
Micro arquitectura	Por cables (<i>hardwired</i>)	Microprogramada
Compiladores	Complejos	Simples
Ejemplos	PowerPC, ARM, SPARC, MIPS	IBM Mainframe 360, Intel x86, arquitectura Z, Motorola 68k

Multiprocesamiento: se procesan varias instrucciones simultáneamente, utilizando más de un procesador.

Memoria

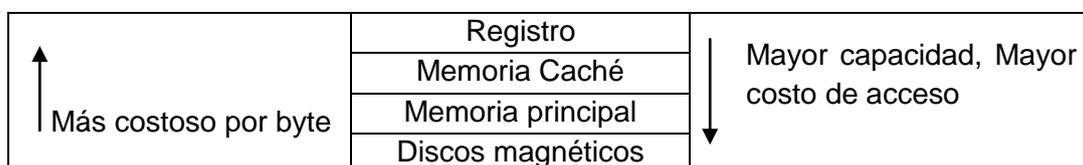
Memoria: Parte de la computadora que permite almacenar los datos y los programas. La memoria está dividida en celdas, la unidad direccionable más pequeña, cada una de las cuales tiene 2 atributos: contenido (valor) y dirección (invariable, identificada con un número). Si una memoria tiene n celdas, tendrá las direcciones 0 a $n-1$. Si una celda tiene k bits, podrá contener cualquiera de 2^k combinaciones de bits distintas.

Clasificación de la memoria:

- Volátiles: cuando se desconecta se pierde la información.
- No volátiles: cuando se desconecta la información se mantiene.

Memoria caché: Es una memoria pequeña y rápida, que se encuentra dentro del chip de la CPU que contiene las palabras de memoria de mayor uso. Cuando la CPU necesita una palabra, primero la busca en el caché. Si no está ahí, recurre a la memoria principal. Por lo general las palabras más usadas es conveniente tenerlas en el caché.

Jerarquía de memoria:



Segmentación y paginación

Los **segmentos** son bloques de memoria de tamaño variable, que contienen información de la misma clase y constituyen el objeto principal sobre el que se basa el mecanismo de protección.

La segmentación es eficaz para organizar la memoria en módulos lógicos de similares características, que son el soporte de la programación estructurada. Es posible compartir recursos entre todas las tareas si se sitúan en el espacio global, o bien, ser exclusivos de una tarea concreta si están ubicados en el área local de la misma. Se puede asignar a cada segmento un determinado nivel de privilegio.

Intel basa el control de la memoria en la segmentación.

La **paginación** divide el espacio de memoria en trozos de longitud fija, llamados páginas, que, en el caso del Pentium, tienen un tamaño de 4 KB ó 4 MB. La paginación simplifica la labor del programador de sistemas al simplificar los algoritmos de intercambio entre objetos de la memoria física y la memoria virtual, al manejar elementos del mismo tamaño lo que supone un incremento de la velocidad en la localización y relocalización de páginas. La paginación es optativa y sólo funciona cuando la activa el programador.

La paginación y la segmentación son técnicas complementarias y ambas introducen ventajas particulares en la gestión de la memoria virtual. Los sistemas operativos UNIX y DOS son, respectivamente, ejemplos representativos de dichas técnicas.

Cuando sólo se precisa trabajar con la paginación, al estar activa siempre la segmentación, es preciso que toda la memoria física se considere como un único segmento, que ocupa un espacio continuo o lineal. El modelo "plano" permite esta posibilidad.

Direccionamiento

Direccionamiento: "función" que recibe un campo operando de una instrucción y devuelve la dirección real del dato en memoria.

- Directo y absoluto: en el campo operando se especifica la dirección real del dato. Los programas no pueden ser trasladados a otra parte de la memoria, sin cambiar todos los operandos. Los operandos deben ser grandes, para poder acceder a toda la memoria. Abacus, ibm, intel.
- Inmediato: en el campo operando se especifica el dato. Abacus, ibm, intel.
- Indirecto: el campo operando contiene la dirección en memoria de una celda X. Esa celda X contiene la dirección Y real del dato en memoria. Ibm, intel
- Relativo: la dirección del dato se obtiene sumando una constante al contenido del campo operando. Ibm, intel
 - Por base y desplazamiento: se especifica un registro base (que contiene una dirección) y un desplazamiento desde esa dirección (la constante). Ibm
 - Por base, índice y desplazamiento: se especifica un registro base, un registro índice y el desplazamiento. Ibm

- Por referencia al programa: la dirección del dato se obtiene sumando el contenido del RPI a un desplazamiento indicado.
- Por yuxtaposición (direccionamiento paginado): la memoria se encuentra dividida en 2^p páginas (bloques de longitud 2^m). Para obtener las direcciones necesitamos:
 - Indicador de página (IP): en un registro específico o de propósito general de la máquina.
 - Dirección de la palabra dentro de la página (DP): en el campo de la instrucción.

Así, concatenando ambas partes obtenemos la dirección completa.

Administración de memoria

Multiprogramación: tratamiento en forma concurrente de más de un programa en tiempo real. Elimina o reduce el tiempo ocioso de la CPU.

Problemas asociados a la multiprogramación

- Problemas de asignación de memoria (se resuelven utilizando memoria virtual)
- Protección de los programas y los datos en memoria.
- Preservación del estado de la máquina, en los cambios de programa.

La administración de memoria es una función del sistema operativo que consiste en decidir a qué programas se le da un espacio en la memoria del sistema.

1. Asignación contigua simple: consiste en darle a un solo programa la totalidad de la memoria que no está ocupada por el sistema operativo, aunque en realidad ella requiera solo una mínima parte de ese total. La tarea tiene control absoluto de la CPU, hasta su finalización o hasta que ocurra un error.

Ventajas: es una forma sencilla; para el programa es la situación ideal ya que tiene la totalidad de la memoria y los recursos disponibles.

Desventajas: se desperdicia tiempo de procesamiento y el tiempo del usuario.



2. Asignación particional: consiste en dividir la memoria en particiones (hasta 16) de tamaño prefijado por el programador. En cada partición se carga un programa. El programador debe saber en qué partición correrá el programa y cuál es el tamaño de la misma.

Ventaja: aumenta la cantidad de trabajo realizado por unidad de tiempo (*throughput*).

Desventaja: se produce fragmentación. Cuando termina la ejecución de los programas chicos, no hay lugar para poner a los más grandes, porque las particiones son fijas.

3. Asignación particional reasignable: consiste en particionar la memoria organizarla para que puedan entrar más programas, y tener un registro de reubicación que apunta al inicio del programa próximo a ejecutarse. Cuando se quiere cargar una celda, su dirección "real" en memoria se obtiene sumado la dirección "ficticia" (provista en la instrucción) y el contenido del registro de reubicación.

Ventajas: se puede reorganizar la memoria; se elimina la fragmentación; la memoria está más compactada; hace al programa independiente de su ubicación física en la memoria.

Desventajas: cada acceso a memoria requiere una cuenta; tiempo insumido en reorganizar la memoria; hay todavía desaprovechamiento de memoria.

4. Asignación paginada: consiste en dividir la memoria física en sectores de igual tamaño (*frames*), y dividir a los programas en páginas que tienen el mismo tamaño de los *frames* (generalmente son 4k). Entonces, cada página de los programas se almacenan en un *frame* distinto, y se tiene un "mapa de páginas" para cada programa. Con la ayuda de este mapa se obtienen las direcciones reales de los datos en memoria.

Ventajas: no es necesario reorganizar la memoria ya que los espacios libres son rellenados sin tener que compactar (solo hay que actualizar el mapa); minimiza la fragmentación.

Desventajas: la paginación de los programas desperdicia espacio (en promedio se desperdicia media página por programa); tiempo insumido al consultar el mapa de páginas; espacio ocupado por los mapas de páginas; este método requiere que todas las páginas del programa (todo el programa) estén en memoria antes de poder ejecutarse.

5. Asignación paginada por demanda: el espacio de direcciones en el que opera una tarea en un sistema de paginación por demanda se conoce con el nombre de memoria virtual. La memoria virtual puede, en principio, ser mayor que la memoria física disponible. La memoria se divide en *frames*, y los programas se dividen en páginas. Se tiene un mapa de páginas para cada programa, que indica el estado de cada página (si está cargada en memoria, la fecha de último acceso, *frame* en el que se encuentra). En memoria se cargan sólo las páginas que son necesarias. Si se necesita una página que no está almacenada y no hay espacio disponible, se quita la página que menos se usó y se carga la página solicitada. Se guarda en el disco una copia espejo a la página saliente.

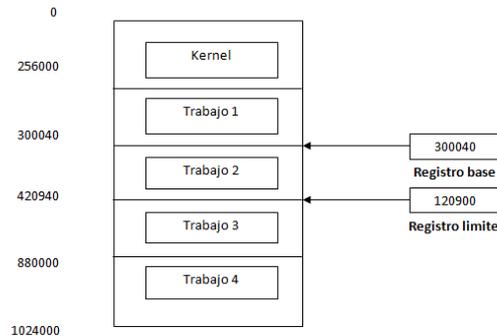
Ventajas: se tienen más programas en memoria; memoria "infinita" para el programador (permite ejecutar programas que no entran completamente en la memoria).

Desventajas: puede pasarse más tiempo cargando y descargando páginas que procesando (*thrashing*); mantenimiento del mapa de páginas; muchos accesos a disco (*trade off*) por cada *page fault*; muchos accesos a mapa de páginas.

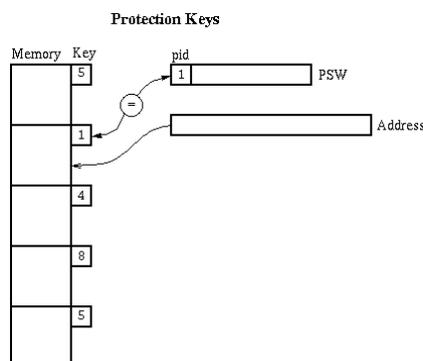
Protección de memoria

El modo de protección de memoria es una forma de controlar los permisos de acceso en una computadora; lo elige el fabricante de la misma. Tiene sentido en un entorno de multiprogramación, y se utiliza para prevenir que un programa acceda a una parte de la memoria que no le corresponde.

- Por registros límites: se tienen 2 registros que apuntan al comienzo y al final del programa que se está ejecutando. Solo se puede acceder a la partición que está entre los límites. En el caso de la memoria particionada reubicable, el registro límite inferior coincide con el registro de reubicación, y el registro límite superior es reg de reubicación + longitud del programa.



- Por llave y cerradura: la memoria se divide en bloques de un tamaño determinado (ej.: 2 kb) y cada programa tiene asignado un conjunto de 4 bits (la cerradura) único. En la PSW hay otro conjunto de bits (la llave). Cuando se quiere acceder a un programa en particular, la llave y la cerradura deben coincidir, caso contrario hay "error de protección". Este sistema se utiliza en la arquitectura IBM System/370. Para cambiar las llaves y cerraduras hay instrucciones especiales: SSK (Set Storage Key, poner una cerradura a un bloque), ISK (Insert System Key, averiguar una cerradura), SPKA (Set PSW Key from Address, pone una llave), IPK (Insert PSW Key, averigua el valor de una llave).



- Por bit asociado: cada byte tiene un bit asociado, si este está en "0" su acceso está permitido, si está en "1" su acceso está prohibido. Cuando el SO le da el control a un programa, pone en 0 todos los bits asociados a esa partición y en 1 a todos los bits de las otras partes. Para acceder a un byte en particular se debe verificar que su bit asociado esté en "0". Si no, el hardware lanza un error. La desventaja de este método es que mantener actualizados los bits es costoso.

Canales de E/S

Canal: procesador que se encarga de trasladar información entre el computador y los periféricos.

- Tiene capacidad lógica (por eso se llama procesador).
- Está conectado a los periféricos a través de la UC de periféricos. Cada UC está especializada en un determinado periférico. Algunos canales de E/S están permanentemente conectados a un único dispositivo mientras que otros pueden estar conectados con varios a

la vez, a los que puede accionar uno por vez (canales selectores) o con simultaneidad (multiplexores).

- Tienen acceso directo a la memoria y tienen prioridad con respecto a la CPU para entrar a ella. Accede para leer o dejar datos o para buscar su comando (instrucción del canal).
- Nunca funcionan en forma independiente; existe una relación amo/esclavo con el computador. El canal sólo funciona cuando recibe una orden del computador, que además podrá interrumpir el funcionamiento del mismo.
- Los canales son necesarios ya que cuando el computador manda a grabar algo, transcurre un cierto tiempo mecánico entre la orden y la finalización de la tarea, y de esto se ocupa el controlador. Si no existiera el controlador, la CPU se encargaría de eso. Para bajar costos, se pone un procesador más barato en el canal, ya que hace una tarea fácil.
- Los canales permiten el multiprocesamiento, ya que en los tiempos de entrada/salida de un programa, el CPU puede encargarse de ejecutar otro programa. Esto se llama *solapamiento de E/S* entre programas, pero también se puede llevar a cabo poniendo varios buffers para el mismo programa.
- El canal permite multiprogramación, aumenta el *throughput* (cantidad de trabajo total por unidad de tiempo realizado por el sistema).
- Homogeneiza la velocidad de transferencia de datos a la CPU (siempre es la misma, independientemente de que los periféricos tienen distinta velocidad de transferencia).
- Es la función del SO actuar como supervisor para lograr la coordinación de todos los procesadores.
- El canal es una unidad programable, capaz de leer, decodificar y ejecutar un programa de canal, registrado en memoria central.

Funcionamiento del Canal

Los canales deben ser activados con los siguientes datos:

- periférico a conectar
- dirección del buffer
- cantidad de bytes a trasladar

El canal tiene dos registros propios:

- registro de cuenta: cantidad de bytes a leer (-1 en cada lectura)
- registro de memoria: dirección donde guardar / guardar (+1 en cada lectura / escritura)

Cuando comienza una operación de E/S, el canal recibe datos y guarda en el registro de cuenta la cantidad de bytes que debe transmitir, y en el registro de dirección la dirección de memoria desde o hacia donde transmitir. Cada byte que pasa disminuye en uno el registro de cuenta y aumenta en uno el de dirección.

La operación de E/S termina cuando:

- registro cuenta = 0
- cancelación de CPU
- se terminaron los datos de la cinta
- error de máquina (ejemplo: error de paridad)

El canal avisa al CPU mediante interrupciones.

Para que el canal comience una operación de E/S, la CPU manda una instrucción:

SIO 132 (canal 1, dispositivo 32). Devuelve en la Channel Status Word (CSW):

- CC=0 → todo bien
- CC=1 → hay problemas
- CC=2 → canal ocupado,
- CC=3 → canal no existe

Otras instrucciones pueden ser:

- TIO (*Test I/O*) → averigua el estado del canal y dispositivo
 - TCH (*Test Channel*) → averigua el estado del canal
 - HIO (*Halt I/O*) → cancela la operación de E/S
1. Durante la ejecución del programa en la unidad central, se genera en memoria una lista de instrucciones de canal, que es el programa de canal.
 2. La unidad central ejecuta una instrucción que proporciona al canal la dirección del periférico interesado y la dirección de la primera instrucción del programa de canal. Si el periférico está libre, se pone en actividad, sino se avisa al CPU. La dirección de la primer instrucción de canal es enviada y almacenada en el contador de instrucciones del canal. A partir de este momento, el programa de canal se ejecuta sin intervención de la CPU.
 3. El programa va a buscar a memoria la primer instrucción de canal, que consta de Cod. Operación de Periférico, Dir. Primer palabra, Cuanta de Palabras y finalmente Indicadores.
 4. Las demandas de transferencia elemental emanan del controlador del periférico que trabaja a su propio ritmo. El canal efectúa la transferencia pedida utilizando y actualizando la dirección en curso y el contador.
 5. La transferencia termina cuando el contador es nulo o hay una señal de fin de transferencia proveniente del controlador.
 6. El canal comprueba los indicadores (Encadenamiento de datos – Encadenamiento de Gobierno – Interrupción)

Programación E/S

El procesador de E/S puede observar el estado de la CPU inspeccionando un registro de estado. La comunicación entre el canal de E/S y la CPU tiene lugar casi siempre por medio de interrupciones, la cual es una disposición mecánica que hace que la CPU suspenda la ejecución, reserve su estado presente y transfiera el control a una posición determinada. En esa posición se especifica la dirección de un programa que ha de entrar en proceso como respuesta a la interrupción (programa de procesamiento de interrupciones).

Estructura del procesador E/S

Hay tres agrupamientos básicos de instrucciones de E/S (comandos):

- Transferencia de datos: lectura, escritura o verificación de estado de dispositivo.
- Control de dispositivos: control (descarga de páginas, rebobinado de cintas, etc.)

- Bifurcación: transferencia de control dentro del programa de canal.

El canal busca las instrucciones (CCW) en la memoria y las descifra según el formato.

Programas de canal: está formado por los comandos de canal.

Channel Command Word (CCW): posición fija en memoria, que contiene una instrucción del programa canal y operandos.

- **Command Code:** consta de 2 a 4 bits de operación (estándares) y de 4 a 6 bits modificadores (dependen del dispositivo). Dice lo que hay que hacer.
- **Data Address:** especifica la posición de un byte en la memoria principal (inicio del campo al que se refiere la instrucción).
- **Byte Count:** cantidad de bytes a transmitir.
- **Flag: Bit 32-37:**
 - 32: encadenamiento de datos. Si es 1 indica que debe usarse el área de almacenamiento de la próxima CCW con la operación en curso.
 - 33: encadenamiento de comandos. Cuando está en 1 indica que debe seguir leyendo, en 0 indica último comando, terminó el programa del canal.
 - 34: si está en 1 suprime el error de longitud incorrecta. Útil cuando se acaban los bytes al leer y el registro de cuenta no llegó a 0.
 - 35: si está en 1 suprime transferencia de datos (el canal hace toda la operación pero no transfiere). Útil para probar programas de canal.
 - 36: si está en 1 causa interrupción de E/S, devuelve el control a la CPU (finalización provocada por el programa).

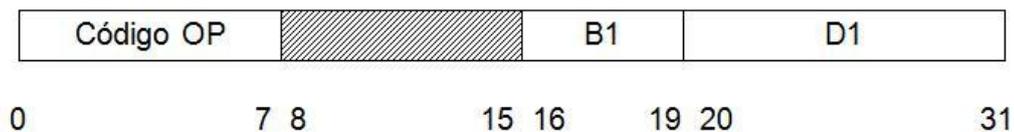
Channel Address Word (CAW): (pag. 15 cartilla): posición fija en memoria, que indica la dirección de la primera instrucción del programa de canal. Se accede a ella por cada SIO y cambia por cada SIO.

Channel Status Word (CSW): (pag. 14 cartilla): posición fija en memoria, en donde el canal deja información sobre cómo termino la operación; por ejemplo: si luego de una SIO devuelve CC=0 entonces la CPU va al CSW para ver que pasó.

Comunicaciones entre la CPU y el Canal:

Hay dos tipos de comunicaciones entre la CPU y el Canal:

1. Instrucciones de E/S iniciadas por la CPU (de CPU a Canal): todas las instrucciones tienen el siguiente formato SI modificado:



El número del dispositivo y el canal quedan especificados por la suma del contenido del registro B1 y el del campo D1.

Hay cuatro instrucciones CPU – E/S ejecutadas por la CPU:

- SIO: (*Start Input Output*) Comienzo de la operación. Se necesitan dos datos: número del canal y del dispositivo, y dirección del inicio del programa del canal. Esta instrucción especifica el número de canal y el dispositivo y el área 72-75₁₆ en la memoria es la CAW, la cual señala el principio del programa del canal. Puede ser ejecutada solo cuando el sistema está en sistema operativo (sino daría error), no la pueden usar los usuarios. Cambia el código de condición: si es 0 entonces está todo bien, si es 1 hay algún problema (la información sobre qué tipo de problema es está en el CSW (pág. 14), si es 2 el canal está ocupado, si es 3 significa canal inexistente.
- TIO: (*Test Input Output*) averigua el estado del canal y del dispositivo para ver si está ocupado o no.
- TCH: (*Test Channel*) averigua el estado del canal.
- HIO: (*Halt Input/Output*) permite detener el funcionamiento del canal.

2. Interrupciones de la CPU iniciadas por el Canal (de canal a CPU)

Tipos de canal

- ✓ Selectores: operan en la modalidad *bursts* (ráfagas) o bloques completos de información. Se conectan a dispositivos rápidos (cintas y discos) y no se desafectan de ellos hasta que termina la operación (cuando se asigna a un periférico se dedica totalmente a él).
- ✓ Multiplexores: operan con la modalidad byte a byte, son conectados a dispositivos lentos (teclado, impresora, lectoras de tarjetas). Pueden ser conectados a más de un periférico a la vez. Por cada periférico abre un subcanal, todos estos se juntan en el canal, por el cual viajan mezclados los bytes de cada periférico. Al llegar a memoria van a su respectivo buffer.
- ✓ Block multiplexor: similar a multiplexor, pero se maneja con bloques de bytes. Si el bloque es muy grande se puede pensar como un selector.

Sub-canal: es una capacidad administrativa del multiplexor de conectarse a varios periféricos. Tiene una capacidad máxima de sub-canales.

Sistema operativo

Sistema operativo: conjunto de programas residentes en memoria cuya función es administrar los recursos del sistema (memoria, CPU, canales y periféricos) para optimizar su utilización.

- *Núcleo, supervisor o monitor*: parte del SO que está siempre en la memoria (en la parte más baja).
- El resto del sistema operativo queda en periféricos (discos), y existe un área donde se cargan las partes del SO que se necesitan y no están en memoria.

Funciones del SO:

1. Manejar operaciones de E/S
2. Atiende las interrupciones
3. Administra el uso de la memoria, a qué sector va cada programa
4. Cargar los programas (los busca)

5. Manejar la protección de memoria (que un programa no invada el área de otro)
6. Manejar los mensajes con el operador
7. Manejar los OPEN y CLOSE de los archivos
8. Llevar la contabilidad de los sistemas
9. Administrar el uso de la CPU
10. Manejar los errores
11. Administrar los periféricos

Carga inicial del SO:

Es el proceso de arranque del computador y carga del sistema operativo para poder utilizarlo.

IPL (*Initial Program Loader*): cuando se ejecuta, carga el SO en la posición inicial.

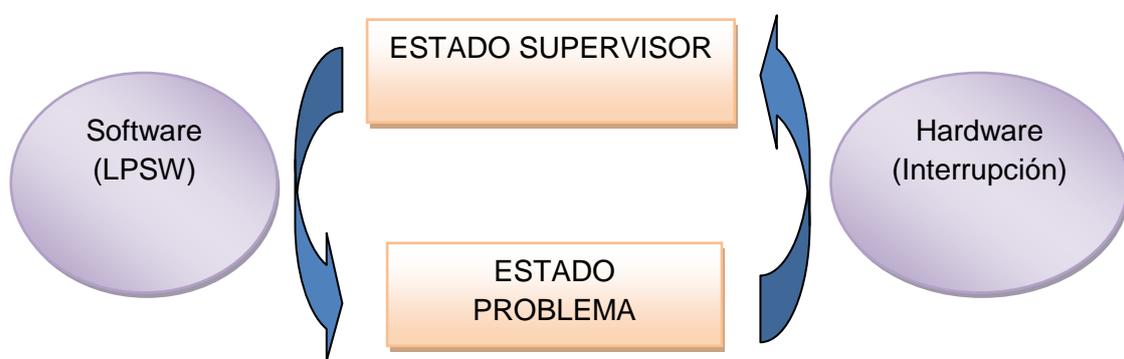
Programa Stand-Alone: programa que no necesita de un SO para funcionar. Debe ser capaz de manejar las interrupciones. Ejemplos: SO, IPL.

Hay tres métodos de carga:

- Por hardware (*bootstrap*): se aprieta una tecla de carga inicial y se pone el RPI en 0, en el RI una instrucción de lectura, en el registro de dirección del canal se pone 0 y en el registro de cuenta del canal 1. Comienza a funcionar la máquina y se ejecuta la instrucción del RI (lectura); empieza a trabajar el canal, conectado siempre a un dispositivo que tiene el IPL en el registro 0. La máquina busca el RPI que es 0, (comienzo del IPL) y lo ejecuta, cargando el supervisor o parte del SO en memoria.
- Método IBM: elegimos el periférico en el cual debe estar cargado el programa de IPL. Se presiona la tecla de carga y por hardware el canal carga un registro de 24 bytes en la memoria (una PSW y dos CCW). El canal actúa como si lo que hizo hubiese sido ocasionado por una CCW. Como consecuencia carga por cada CCW 80 bytes en la memoria (160 bytes = IPL). Cuando termina con la CCW2 provoca una interrupción y la CPU ejecuta LOAD PSW (cargada por hardware) y carga la PSW de dirección 0. Esta PSW apunta a la primera instrucción del IPL que al ejecutarse carga el supervisor. El SO se carga encima del programa de IPL.
- Por memoria ROM (*Read Only Memory*): al apretar la tecla de carga del SO el sistema busca en ROM el IPL (grabado por el fabricante) y lo transfiere a la memoria RAM desde donde se ejecuta. Si un programa no usa SO hay que cargarlo con alguna maniobra del tipo IPL. Si no alcanza la memoria se desaloja el SO.

El computador tiene 2 estados posibles:

- Estado supervisor: la PSW tiene el bit 15 en 0. El SO siempre se ejecuta en este estado. Mientras el sistema está en modo supervisor, las protecciones de memoria son inhibidas.
- Estado problema: la PSW tiene el bit 15 en 1. No se pueden ejecutar instrucciones privilegiadas en este estado.



Ejemplos de instrucciones privilegiadas:

- De E/S: SIO / TIO / HIO / TCM
- De timer o reloj: STCK (*Store Clock*) averigua la hora, SCK (*Set Clock*) pone el reloj en hora.
- De protección de memoria SSK (*Set Storage Key*) cambia una cerradura de un bloque, ISK (*Insert Storage Key*) pregunta por la cerradura de un bloque, SPKA (*Set PSW Key From Address*) cambia 1 llave en la PSW, IPK (*Insert PSW Key*) averigua la llave de la PSW actual
- Enmascara interrupciones: SSM (*Set System Mask*)
- Cargar PSW: LPSW (*Load PSW*)

Interrupciones

Interrupción: mecanismo de asincrónico con la ejecución del programa, mediante el cual el computador deja de ejecutar el programa en curso y pasa a ejecutar otro programa llamado Rutina de Atención a la Interrupción (RAI).

Quando ocurre una interrupción ...:

- 1) Se termina de ejecutar la instrucción en curso,
- 2) Se guarda (por hardware) la PSW en un lugar fijo de la memoria, llamado PSW "vieja". Se carga la PSW "nueva" sobre la PSW en curso. La PSW nueva, apunta a la RAI.
- 3) Se ejecuta la RAI;
- 4) La última instrucción de la RAI es LPSW (*Load PSW*), que restaura la PSW "vieja" y continua con la ejecución del programa interrumpido (el retorno es por software).

El estado actual de la CPU puede modificarse cargando una nueva PSW o modificando una parte de la PSW en curso.

Algunos de los bits de la PSW se prestan para enmascarar ciertas interrupciones. Una vez enmascaradas, las interrupciones de E/S, externas o de máquina deben ser inhibidas temporariamente y quedar pendientes. Dicho enmascaramiento afecta solo a 4 de las 15 interrupciones del programa. Algunas de las instrucciones de conmutación de estado son: LPSW (*Load PSW*), SPM (*Set Program Mask*), SSM (*Set System Mask*), SVC (*Supervisor Call*), SSK (*Set Storage Key*) e ISK (*Insert Storage Key*).

Si mientras se está ejecutando una interrupción aparece otra, puede llamarse a una rutina de cola de interrupciones. Si mientras se está ejecutando una cola de interrupciones ocurre una interrupción más, se volverá a llamar a la rutina de cola, lo que puede ocasionar la pérdida de una interrupción. Para evitar esto, se deben enmascarar todas interrupciones mientras la CPU procesa una cola.

Tipos de interrupciones:

- ◆ *De E/S*: se produce cada vez que un canal “finaliza” una operación de E/S. En la CSW queda el resultado de la operación. La PSW es un registro de la CPU, mientras que la CSW es un área de memoria, que lleva ese nombre por la función que cumple.
- ◆ *Por programa*: provocadas por acontecimientos que suceden durante la ejecución del programa y que no fueron programadas.
 1. Operación: si se intenta ejecutar una instrucción cuyo CO no existe.
 2. Instrucción privilegiada: cuando un usuario intenta ejecutar una instrucción privilegiada estando en modo problema.
 3. *Execute*: le dice al sistema que ejecute una instrucción que está en otro lugar de la memoria. La interrupción se produce cuando se hace un execute de un execute.
 4. Protección de memoria: cuando un programa quiere acceder a una zona de memoria que no tiene permitida.
 5. Direccionamiento: cuando se quiere acceder a un área de memoria no existente.
 6. Especificación: cuando, por ejemplo, uso una instrucción que precisa entorno a palabra con una dirección sin entornar.
 7. Datos (*data exception*): cuando los datos no respetan el formato.
 8. *Overflow*: puede ser de punto fijo, de flotante o de empaquetado. Si no está inhibida la condición de overflow, cancela el programa.
 9. División por cero: puede ser de punto fijo, flotante o empaquetado.
 10. *Underflow* de punto flotante: no cancela, sino que pone 0 y sigue adelante.
 11. Significación: cuando en punto flotante se pierden dígitos en la mantisa.
- ◆ *Por llamada a Supervisor*: cada vez que se ejecuta una instrucción SVC (*Supervisor Call*). Esta interrupción no es asincrónica, está escrita en el programa (autointerrupción). Se utiliza para pasar de estado “problema” a estado “supervisor”. Está incluida en las macroinstrucciones, pues estas requieren del uso del SO (OPEN; CLOSE, etc.). Tiene un código que va de 0 a 255, según el cual se sabe que rutina debe ver.

Rutinas del SO: Luego de intercalar las instrucciones de, por ejemplo, GET, el ensamblador se fija en el código de la última instrucción (SVC) que le dice que era una GET y va a las rutinas que se ocupan del bloqueo y desbloqueo. Cada vez que se hace una GET un registro lógico pasa al área de E/S (en memoria). El SO se da cuenta que tiene que ir a buscar un nuevo registro físico por:

IOCS (*Input Output Control System*): consta de dos partes.

- LIOCS (*Logical IOCS*): contiene las rutinas de bloqueo y desbloqueo. Se fija si hay un nuevo registro lógico, si hay lo pasa y corre el puntero, sino le pasa el control a la PIOCS.
 - PIOCS (*Physical IOCS*): maneja el canal para pasar un nuevo registro físico. Hace la SIO, etc. Cuando el canal termina de trabajar hace una interrupción de E/S.
- ◆ *Externa*: puede deberse a tres causas: el operador pulsa la *tecla externa* (en ese momento el SO interactúa con el operador), reloj de la máquina (*timer*) (estrategia de asignación del procesador por un tiempo determinado, el *timer* le avisa al SO entonces hay una interrupción), o comunicación entre computadoras (retorna mediante una interrupción externa).
 - ◆ *Por error de máquina*: se divide en dos tipos.

- Recuperable: el SO lo atiende, guarda todo el programa en un archivo y luego devuelve su control. Por ejemplo: falla en cinta.
- No recuperable: no retorna nunca.

Prioridad de atención de interrupciones

No se puede atender a más de una interrupción del mismo tipo al mismo tiempo, por eso existen prioridades de ejecución de interrupciones. Las prioridades permiten que una interrupción interrumpa una RAI en proceso.

Si se está procesando	Entonces está inhibida...				
	E/S	PROG	SVC	EXT	MAQ
E/S	0	1	1	1	1
PROG	0	0	1	1	1
SVC	0	0	0	1	1
EXT	0	0	0	0	1
MAQ	0	0	0	0	0

0 : inhibida, 1 : desinhibida

La más prioritaria es la de error de máquina.

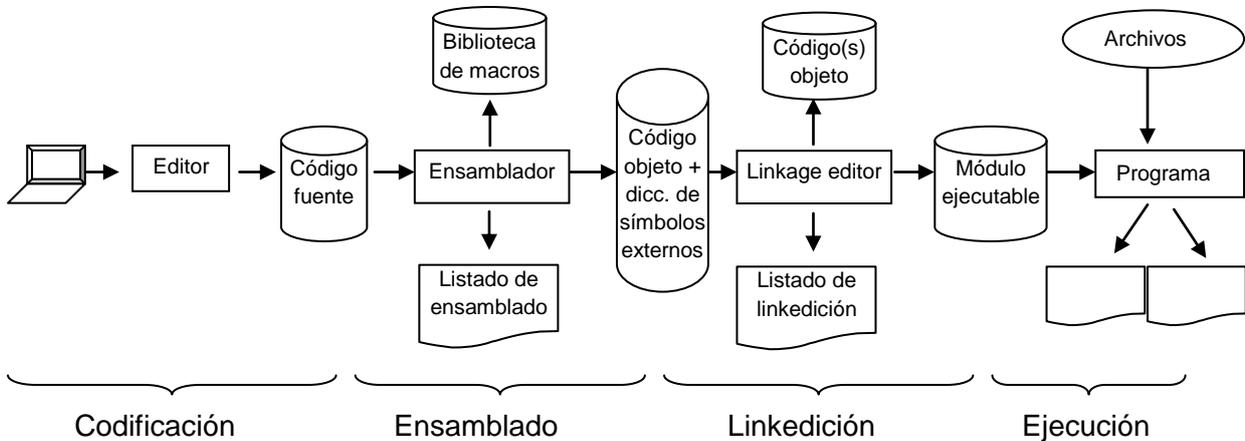
Unidad 4: Assembler IBM /370

- **Compilador:** programa que acepta como entrada un programa fuente redactado en un lenguaje de alto nivel y produce como salida un programa objeto.
- **Intérprete:** programa que aparentemente ejecuta un programa fuente como si estuviera redactado en lenguaje de máquina. Generalmente interpretan línea por línea los comandos ingresados por teclado. Son más lentos que los compiladores.
- **Ensamblador:** programa que lee instrucciones en lenguaje Assembler y las traduce al lenguaje máquina. Sus ventajas son:
 - Permite escribir instrucciones en forma simple y clara, utilizando códigos nemotécnicos.
 - Permite definir áreas de memoria (DS) y áreas con valor (DC).
 - Permite operar con rótulos. Los rótulos tienen de 1 a 8 caracteres alfanuméricos con la condición que el primero sea alfabético.

Proceso de ensamblado

1. Escribimos el programa en pantalla y lo guardamos en un archivo (código fuente).
2. El ensamblador lee el archivo y genera el programa objeto, para lo cual consulta la biblioteca de macros (SUBENTRY, GET, etc.), los intercala y produce la salida: listado de ensamblado. Para generar el código objeto el ensamblador realiza dos pasadas:

- a. Crear una tabla con todos los símbolos y sus valores. Determinar la longitud de cada instrucción para poder calcular las direcciones de los rútilos.
 - b. Usar la tabla para generar el código objeto.
3. El programa objeto es tomado por el *linkeditor* que resuelve las direcciones externas, busca las rutinas externas, las intercala y genera un módulo ejecutable.



Arquitectura

Almacenamiento

Capacidad de memoria : 2^{24} bytes (16 MB)

Bit	Byte	Halfword	Fullword	Doubleword
1 bit	8 bits. <i>Zone (4 bits) y Numeric (4 bits)</i> HIGH ORDER NIBBLE LOW ORDER NIBBLE	2 bytes. Frontera divisible por 2.	4 bytes. Frontera divisible por 4.	8 bytes. Frontera divisible por 8.

Registros

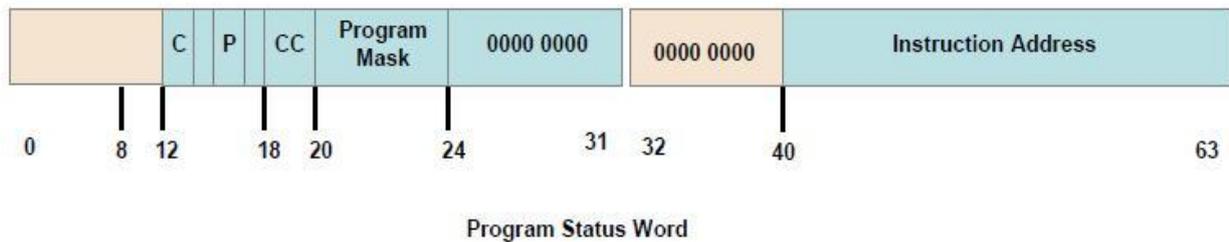
Hay 16 registros generales de 32 bits (1 Fullword) que se usan para la aritmética binaria, para crear direcciones de memoria, para contar en los bucles, etc. Van del 0 al 15. Hay 4 registros de punto flotante de 64 bits (0,2,4,6). Todos los registros se manejan con BPF con signo.

Direccionamiento

Un byte en memoria se indica especificando la dirección de inicio del byte (base) y un desplazamiento desde él. La dirección base se encuentra en un registro, y el desplazamiento se indica en una instrucción. En notación explícita una dirección se ve como $D(X,B)$, donde D es el desplazamiento, X es el registro índice y B es el registro base. La dirección se calcula como

D+(X)+(B). Un registro base puede direccionar hasta 4096 bytes. El registro base se carga en tiempo de ejecución, pero se define en tiempo de ensamblado.

Program Status Word



Es una colección de información que indica el estado actual de la máquina. Contiene dos campos importantes:

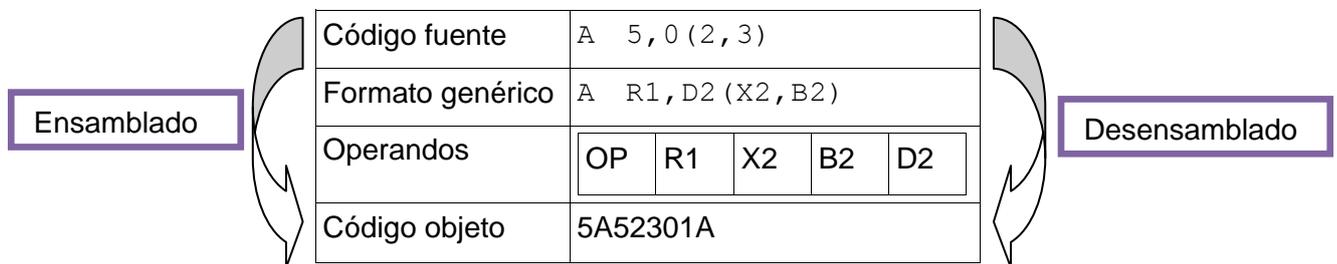
- *Condition Code (CC)*: bits que cambian luego de ciertas instrucciones y sirven para bifurcar. Está formado por los bits 18 y 19. La máscara consiste en 4 bits que apuntan a (0, 1, 2, 3) del código de condición y pone un 1 en la opción que le interesa que se cumpla y un 0 en la que no.

00	Equal
01	Low
10	High
11	Overflow

- *Instruction Address (24 o 31 bits)*: Contiene la dirección de la próxima instrucción a ejecutarse. S/360 tenía direcciones de 24 bits, en 1970 se expandió a 31 bits. 2^{31} bits son 2 gigabytes de memoria.

Formatos de instrucciones

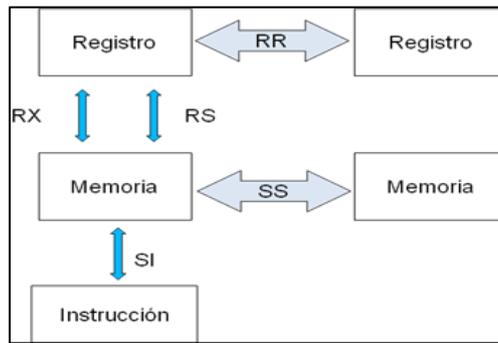
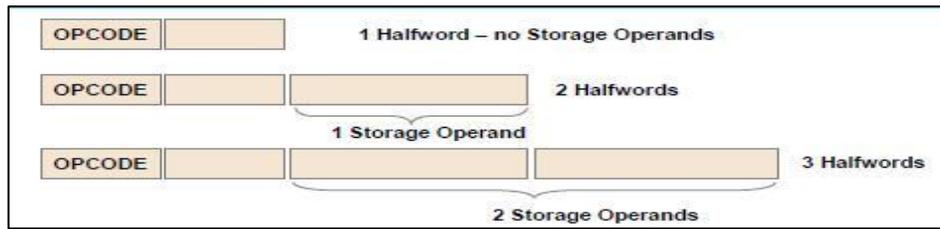
Ejemplo:



Clasificadas por el lugar en donde residen los operandos:

SS1	<i>Storage to Storage</i>	OP D1(L1,B1),D2(B2)	3 halfwords
SS2	<i>Storage to Storage</i>	OP D1(L1,B1),D2(L2,B2)	3 halfwords
RR	<i>Register to Register</i>	OP R1,R2	1 halfword
SI	<i>Storage Immediate</i>	OP D1(B1),I2	2 halfwords

RX	<i>Register to Indexed Storage</i>	OP R1,D2(X2,B2)	2 halfwords
RS	<i>Register to Storage</i>	OP R1,R3,D2(B2)	2 halfwords



Clasificadas por su función:

Aritméticas	AR, A, AP, SR, S, SP, MR, M, MP, DR, D, DP
Comparaciones	CR, C, CLC, CLI, CP
Lógicas	OR, O, OC, OI, NR, N, NC, NI, XR, X, XC, XI
Corrimientos	SLA, SLL, SRA, SRL, SLDA, SLDL, SRDA, SRDL
Conversiones	PACK, UNPK, CVB, CVD
Manejo de archivos	OPEN, GET, PUT, CLOSE
Carga y transferencia	L, LR, LM, LA, ST, STM, MVC, MVN, MVZ, MVI, ZAP, STC, STCM, IC, ICM
Control	BC, BCR, BCT, BAL, BALR
Entrada / salida	WTO, WTOR

Clasificadas por su tipo:

- **Instrucciones propiamente dichas:** Son aquellas que el ensamblador transforma uno a uno en lenguaje máquina (LA, PACK, ST, etc.).
- **Macro-instrucciones:** instrucciones pre-escritas, guardadas en una biblioteca de macroinstrucciones, y que son invocadas por su nombre (CLOSE, OPEN, GET, PUT, etc.).
- **Pseudo-instrucciones:** Comandos para el ensamblador que no son transformados en lenguaje máquina (END, USING, START, etc.).

Operandos:

1. Registros generales.
2. Explícitos: dados como base, índice y desplazamiento.
3. Simbólicos: apuntan a zonas de memoria (rótulos).
4. Literales: definen área de memoria anónimas.
5. Inmediatos: ocupan 1 byte.
6. Máscaras: especifican la condición de bifurcación.

Definición de áreas y datos

ROTULO1 DS rTLn

ROTULO2 DC rTLn'constante'

r	Factor de repetición: se utiliza para el solapamiento de campos. El factor 0 existe y se utiliza para forzar la alineación y para el solapamiento de campos.
T	Tipo de dato: - Z: <i>Zoned</i> (1 dígito por byte) - P: <i>Packed</i> (2 dígitos por byte) - X: Hexadecimal - C: <i>Character</i> (1 byte, EBCDIC) - H: <i>Halfword</i> - F: <i>Word</i> - D: <i>Doubleword</i> - B: Binario (BPF c/signo)
L	Largo del campo
n	Número que indica el largo del campo

Ejemplos:

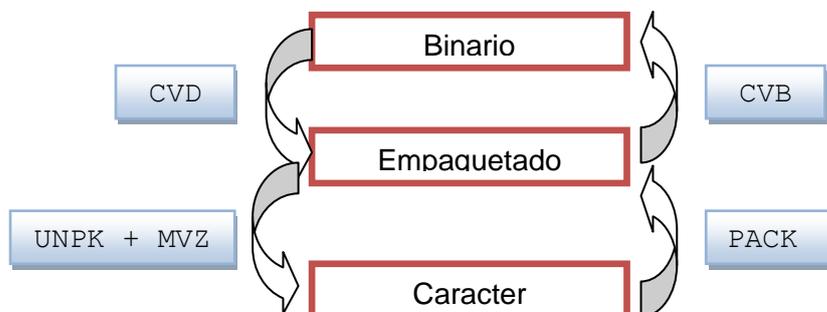
Definición de datos	En memoria
X DC CL3'ABC'	C1C2C3
PACK DC PL3'54'	00054C
ZONE DC ZL5'-354'	F0F0F3F5D4
BINA DC F'1'	1

El tamaño máximo de DC es de 256 bytes.

El tamaño máximo de DS es de 65.535 bytes.

Cuando se usa DC, se debe especificar un valor inicial. Cuando se usa DS no.

El espacio asignado en memoria es " $r \leq \lceil n \rceil$ ".



Manejo de archivos

Parámetros de la DCB (Data Control Block)

Parámetro	Descripción	Opciones posibles
DDNAME	Rótulo que contiene la dirección del archivo.	
DSORG	Especifica la organización del archivo.	PS: "Physical Sequential"
MACRF	Especifica el formato de las macros utilizadas para acceder a los registros del archivo.	GM: "Get Move", para input. PM: "Put Move", para output.
RECFM	Especifica el formato de los registros.	FB: "Fixed Blocked" VB: "Variable Blocked"
EODAD	Dirección a la que bifurca cuando se produce un EOF.	
LRECL	Longitud del registro lógico (en bytes).	80 para input y 133 para output.
BLKSIZE	Longitud del registro físico o bloque (en bytes).	Se omite para input, 6650 para output.

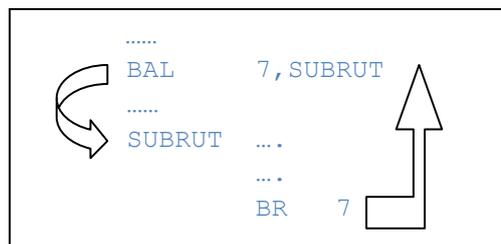
Abrir y cerrar archivos

Ejemplo:

```
OPEN (FILEIN, (INPUT), FILEOUT, (OUTPUT))  
.....  
CLOSE (FILEIN, , FILEOUT)
```

Rutinas internas

Si en la rutina principal utilizamos un registro X, en la subrutina no debemos utilizar ese registro.
Ejemplo:



Rutinas externas

Cuando cualquier programa o subrutina devuelve el control al programa que lo llamó, los contenidos de los registros generales 2 a 14 deben ser los mismos que cuando se dio entrada a la rutina.

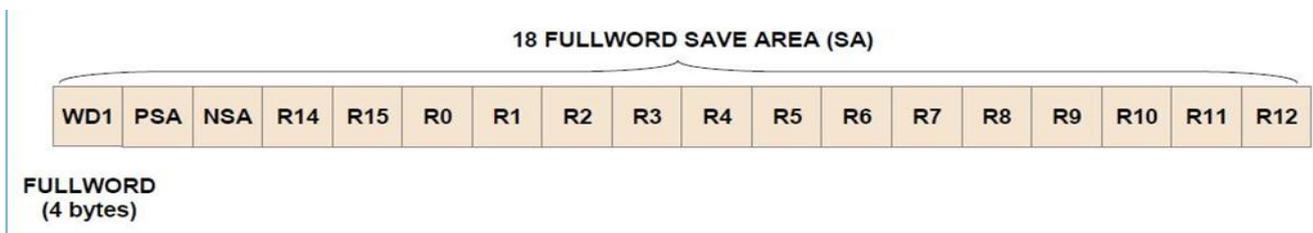
Macros SUBENTRY y SUBEXIT

START	X'00'	Pseudoinstrucción que indica que ahí comienza el programa. '00' es el contenido inicial del contador de posiciones del ensamblador. Reporta el rótulo identificatorio del programa.
STM	14, 12, 12 (13)	Guarda los registros 14 a 12 en la SA (del programa llamador).
BALR	3, 0	Carga en el registro base la dirección de la próxima instrucción.
USING	*, 3	Pseudoinstrucción que avisa al ensamblador cuál es el registro base (no se da cuenta con BALR porque es una instrucción más). El op1 indica cual será el valor del registro base (* es el valor del contador de posiciones en ese momento).
LR	15, 13	Preservar la dirección de la SA superior en el registro 15.
LA	13, SA	En el registro 13 se carga la dirección de la SA propia.
ST	13, 8 (15)	Dir de la SA actual en la 3° palabra de la SA superior.
ST	15, 4 (13)	Dir de la SA superior en la SA propia.
B	SIGUE	
SA	DS 18F	Definición de la <i>Save Area</i> .
SIGO	EQU *	Pseudoinstrucción que asigna al rótulo SIGUE la dirección de *

L	13, 4 (13)	En el registro 13 se recupera la dirección de la SA superior.
LM	14, 12, 12 (13)	Recupera los registros del programa llamador desde su SA.
BR	14	Retorno a la dirección contenida en el registro 14.

Save Area

Es un espacio de 18 Fullwords que se utiliza para resguardar el contenido de los registros al momento de transferir el control de un módulo a otro.



Condiciones de link

El programa llamador debe:

1. Si la tabla de parámetros (que contiene las direcciones en memoria de los parámetros a utilizar) es de longitud variable, guardar un '1' en el primer bit del último elemento de la tabla.
2. R1 = dirección de la tabla de parámetros.
3. R15 = dirección del módulo a llamar.
4. R13 = dirección de la SA.
5. R14 = dirección de la próxima instrucción a ejecutarse.

El programa llamado debe:

1. Guardar los registros del programa llamado

2. Establecer los valores iniciales de los registros base
3. Guardar su propia SA y linkearla con la del programa llamador

Unidad 5: Almacenamiento secundario

Cintas magnéticas

Cinta magnética: fue el primer tipo de memoria secundaria; se almacenan en un carrete (*reel*) cuya longitud se mide en pies. Es un tipo de almacenamiento no volátil que consiste de una cubierta magnética sobre una tira delgada de plástico. Las cintas son dispositivos de acceso secuencial, lo cual implica que para leer el registro n se deben leer necesariamente los $n-1$ registros anteriores.



En general las cintas graban 9 bits: 1 byte con información y un bit adicional. El valor del **bit de paridad** se fija de forma tal que la cantidad de bits en 1 de un número par (si se trabaja con paridad par), o un valor impar (si se trabaja con paridad impar).

La lectura de los unos y ceros se hacía midiendo cambios de polaridad en la superficie magnética

Densidad de grabación (δ): cantidad de bytes grabados por unidad de longitud de cinta (pulgada).

Registro: conjunto de información correspondiente a una entidad.

Cuando se graba o lee información en cinta la transferencia se puede realizar sólo si la cinta pasa bajo la cabeza lectora / grabadora a una velocidad determinada. La unidad necesita cierto tiempo para alcanzar esta velocidad.

Con cada registro la unidad de cinta se pone en funcionamiento, alcanza la velocidad requerida para la transferencia, graba y frena hasta detenerse.

La cantidad desperdiciada por un registro es siempre la misma. Esta depende de la unidad y no de la forma en que se grabe.

Inter-Record Gap (IRG) o Inter-Block Gap (IBG): espacio de cinta desperdiciado entre dos registros (el desperdicio en detenerse luego de grabar el primero y arrancar para grabar el segundo).

Para disminuir la cantidad de cinta desperdiciada en IRG se suele agrupar varios registros y grabarlos en bloques.

Cada registro para ser grabado, es almacenado en una porción de memoria central llamada **buffer** de entrada/salida. Cuando aquellos llegan a una cantidad previamente establecida, se graba el bloque en la cinta.

De esta manera, el tamaño del grupo de registros que se graba es mayor que el de un registro solo y, aunque el tamaño de cada IRG es siempre el mismo, la cantidad de IRG es menor, disminuyendo el espacio desperdiciado en estos.

Registro lógico (RL): registro que contiene la información con la que trabaja la aplicación.

Registro físico (RF): bloque de registros lógicos que se graba en el dispositivo de almacenamiento.

Factor de bloqueo (FB): relación entre registro físico y registros lógicos. El FB indica la cantidad de registros lógicos que contiene cada registro físico. Los dispositivos de almacenamiento sólo operan con registros físicos.

Como hemos visto, el FB indica la cantidad de registros lógicos por cada registro físico.

$$FB = \frac{QRL}{QRF} = \frac{LRF}{LRL}$$

$$LRF = LRL * FB$$

La LRF puede expresarse en bytes (LRFb) o en pulgadas (LRFi):

$$LRFb = LRFi * \delta$$

$$QRF = \frac{QRL}{FB}$$

Fragmentación: efecto producido por el desperdicio de espacio. En algunos casos, los porcentajes elevados de fragmentación se deben a la elección de un FB no adecuado al sistema con que se cuenta. Reconocemos tres tipos de fragmentación.

1. Fragmentación interna: se produce en los casos en que el último registro físico del archivo no está completamente usado por registros lógicos.

$$FRI = QRF * LRF - QRL * LRL$$

2. Fragmentación externa: está originada por porciones de cintas no utilizadas por registros físicos ni IRG's que no pueden ser utilizadas para otro archivo.

$$FRE = L_{cinta} - QRF_{x\ cinta} * (LIRG + LRF) \text{ para un solo carrete.}$$
$$FRE = FRE_{x\ cinta} * (Qcintas - 1)$$

3. Fragmentación del sistema: se debe al espacio en cinta desperdiciado entre bloques, producto de la aceleración y desaceleración de la unidad de cinta para alcanzar la velocidad de arrastre.

$$FRS = QRF * LIRG$$

El **porcentaje de fragmentación interna** advierte el grado de importancia del posible desperdicio producido en el último registro físico del archivo.

$$FRI = \frac{FRI}{Total\ del\ archivo} * 100$$

El **porcentaje de fragmentación total** se calcula como la suma de los tres porcentajes antes mencionados.

El total de espacio que ocupa el archivo se calcula de la siguiente manera:

$$Total\ del\ archivo = (Q_{Carretes} - 1) * L_{Carrete} + QRF_{Ultimo\ carrete} * (LRF + IRG)$$

Velocidad de arrastre (Va): velocidad a la que pasa la cinta por la cabeza lectora grabadora cuando se realiza la transferencia de datos.

Velocidad de transferencia (Vt): es una característica de cada unidad e indica cuanta información puede leerse o grabarse por unidad de tiempo.

$$Vt[b/seg] = Va[i/seg] * \delta[b/i]$$

El **tiempo de grabación** de un registro aislado se calcula sumando el tiempo consumido en la transferencia de los datos y el tiempo utilizado por el dispositivo en arrancar y frenar, llamado tiempo de IRG.

$$T_{1RF} = T_{transf} + TIRG$$

$$T_{transf} = \frac{LRF}{Vt}$$

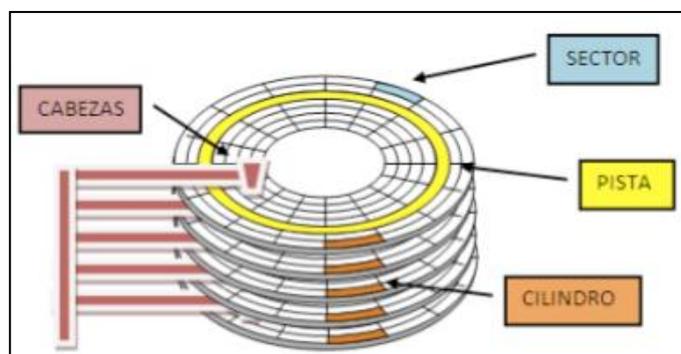
El TIRG es el tiempo empleado por el dispositivo para frenar y acelerar entre cada bloque de información que graba. No es un dato fácilmente calculable debido a que la aceleración de la unidad no es constante.

$$T_{ARCH} = T_{1RF} * QRF = \left(\frac{LRF}{Vt} + TIRG \right) * QRF$$

Discos magnéticos

Discos magnéticos: están compuestos por una serie de platos recubiertos con material magnetizable (óxido de hierro) y un armazón que sirve de soporte al brazo que porta las cabezas lectoras-grabadoras, cada uno de estos platos será denominado en este apunte como un disco, si el dispositivo tiene más de un disco (plato) se lo llamará *Disc-Pack*. Los discos están girando todo el tiempo.

Se dice que los discos son magnéticos por el modo en el cual se lee y se graba la información en el disco, lo cual se hace mediante la polarización de la superficie magnética; la cabeza lectora grabadora detecta estos cambios de polarización y los traduce en unos y ceros. Antes de ser utilizado el disco debe ser inicializado de forma tal que se conozca su organización y la controladora pueda leer correctamente la información, a este proceso se lo llama formateo de bajo nivel (*low level format*).



Cuando se requiere seguridad existen en algunos países reglamentaciones especiales de tipo gubernamental que regulan como debe borrarse la información de un disco de modo tal que los datos no puedan ser recuperados.

Cada **disco** (plato) se divide en un número entero de **pistas** (*tracks*).

Cilindro: conjunto de pistas equidistantes del centro que son accedidas simultáneamente por las cabezas lectoras-grabadoras. Un cilindro se compone de n pistas.

Geometría de un disco: descripción de la estructura física del mismo. Comprende número de cilindros, número de cabezas, número de sectores por cilindro, número de pistas por cilindro, y número de sectores por pista.

Discos no sectorizados

En los **discos no sectorizados** la pista se divide en bloques cuya longitud puede ser definida por el usuario.

Al igual que en cintas existen en discos los registros lógicos y los registros físicos, la relación entre el tamaño del registro físico y el tamaño del registro lógico está dada por el factor de bloqueo. De acuerdo al factor de bloqueo establecido el S.O. fija un área de memoria llamada *buffer* de entrada-salida en la cual se irán formando los registros físicos para luego ser transferidos desde o hacia el periférico.

El registro físico es en discos no sectorizados la unidad mínima de transferencia.

El factor de bloqueo debe ser un número entero. Si la cuenta diera como resultado un número con decimales habrá que redondear.

Al igual que en cintas existe en discos el IRG (*Inter Record Gap*). En este caso el IRG se considerara como un espacio de la pista que se desperdicia. Sin embargo el IRG no es un espacio realmente desperdiciado pues contiene información del sistema. Al igual que en cintas hay un IRG por cada registro físico.

En la gran mayoría de los problemas de almacenamiento en discos es fundamental conocer cuántos registros físicos entran en una pista.

$$QRP = \frac{LP}{LRF + IRG}$$

Si QRP no fuera un número entero, entonces tendríamos que el último registro físico de la pista ocupa una parte de la primera pista y una parte de la segunda pista.

Spanning: un mismo registro físico puede ocupar dos pistas.

En este apunte solo consideraremos unidades de disco que NO admitan *spanning* de pistas, por lo tanto QRP deberá ser redondeado siempre hacia abajo para que entre un número entero de registros físicos por pista.

$$\text{Si LRF es fijo} \rightarrow QRP = \left\lfloor \frac{LP}{LRF + IRG} \right\rfloor$$

$$QP = \left\lfloor \frac{QRF}{QRP} \right\rfloor$$

Si se quisiera calcular para un disc-pack el número de cilindros que ocupa un archivo la cuenta es:

$$Q_{cilindros} = \left\lceil \frac{QP}{QP_x \text{ Cilindro}} \right\rceil$$

Buffer: porción de memoria que se destina a la entrada y salida para un archivo determinado. El periférico lee y escribe los datos desde y hacia el *buffer*, el cual es luego manejado por el S.O. trasladando los datos desde y hacia la memoria del usuario. Existe un *buffer* por cada archivo abierto y cada *buffer* tiene la longitud necesaria para almacenar un registro físico del archivo.

En cintas el registro físico podía ser tan grande como se quisiera, dependiendo de la memoria disponible. En discos esto no ocurre, pues además del buffer existe una limitación muy importante en cuanto a la longitud del registro físico: la longitud de la pista. Al trabajar con discos que no admiten *spanning* un registro físico no puede ocupar más de una pista.

$$LRF_{max} = LP - IRG \text{ (sin considerar el buffer)}$$

El otro factor que limita la longitud del registro físico es la disponibilidad de espacio en memoria para el buffer. Si la longitud máxima del buffer es menor que LRF_{max} , decimos que el buffer "restringe" al problema.

Factor de bloqueo óptimo en discos: es aquel que mejor cumple con el criterio de optimización fijado. El FB óptimo depende de qué es lo que se quiere optimizar. Dos criterios de optimización son:

- Minimizar la cantidad de accesos al disco. Cada registro físico exige un acceso para ser leído o grabado. Minimizar accesos implica minimizar la cantidad de registros físicos del disco. Minimizar QRF es lo mismo que minimizar QRP. Para minimizar QRP hay que lograr que la longitud del registro físico sea máxima, para ello hay que maximizar el factor de bloqueo.
- Minimizar la cantidad de bytes desperdiciados (por archivo o por pista). Para un archivo $QBD = (QP * LP) - (QRL * LRL)$. Si suponemos que $FRI=0$, para una pista $QBD = FRE + FRS$.

FRI = Fragmentación interna, bytes desperdiciados por una mala utilización del ULTIMO registro físico. Esto ocurre solo en la última pista del archivo. En discos la FRI no es un desperdicio real pues un RF puede sobre-escribirse, luego es posible agregar RLs a un RF ya grabado y de esta forma disminuir FRI. FRI en discos no sectorizados es un dato de poca relevancia.

FRS = Fragmentación del sistema, bytes desperdiciados por IRGs, esto ocurre en todas las pistas. No es realmente una "fragmentación" pero en este apunte lo vamos a considerar así para simplificar los conceptos. El concepto es análogo al de cintas.

FRE = Fragmentación Externa, bytes "desperdiciados" por pista y que no son IRGs. Bytes que "sobran" por pista. Esto ocurre por cada pista que ocupe el archivo. Debido a la FRE puede darse que habiendo muchos bytes libres en un disco no pueda grabarse un registro de un determinado archivo pues no existe ninguna pista con suficiente espacio disponible.

En cintas todos los carretes podían ser tratados de la misma forma excepto el último, en discos ocurre lo mismo, todas las pistas se pueden tratar de igual forma excepto la última pista del archivo.

$$\begin{aligned}
 FRE &= QP - QRF * (LRF + IRG) \\
 FRS &= QRF * IRG \\
 FRI &= QRF * LRF - QRL * LRL
 \end{aligned}$$

$$\% FR_{xpista} = \frac{FRE+FRS}{LP} * 100 = \frac{LP-QRP*LRF}{LP} * 100$$

Teorema: Si el tamaño del *buffer* no es limitante entonces el mayor Factor de bloqueo posible minimiza %FR.

El tiempo necesario para leer o escribir un RF se calcula de la siguiente manera:

$$T_{1RF} = T_{seek} + T_{search} + T_{transferencia}$$

Tiempo de Seek: tiempo necesario para que la cabeza lectora-grabadora se desplace desde la pista en la cual se encuentra hasta la pista del dato. Como no es posible saber dónde se encuentra la cabeza en cada momento hay que utilizar un tiempo de *seek* promedio.

$$\begin{aligned}
 T_{seek \text{ promedio}} &= \frac{1}{3} * T_{seekmax} \\
 T_{seekmax} &= QP * T_{seek1pista}
 \end{aligned}$$

A todos los tiempos de *seek* hay que agregarles también un pequeño tiempo, ϵ , que es el tiempo que tarda en "arrancar" el brazo. En nuestras formulaciones no consideraremos a este tiempo.

Tiempo de Search / demora rotacional / período de latencia: tiempo necesario para que el dato llegue hasta la cabeza lectora grabadora. También debe usarse un tiempo promedio que se calcula como 1/2 del tiempo de rotación.

Tiempo de rotación: tiempo que tarda el disco en dar una vuelta.

Tiempo de transferencia: tiempo necesario para leer o escribir los datos. Se calcula como:

$$\begin{aligned}
 TT_{1RF} &= \frac{LRF}{Vt} \\
 Vt &= \frac{LP}{T_{rotación}} = \frac{LP}{2T_{search}}
 \end{aligned}$$

La mayoría de los discos rígidos tienen actualmente una velocidad de 7200 revoluciones por minuto y los diskettes 360 revoluciones por minuto.

Discos sectorizados

Discos Sectorizados: son aquellos en los cuales cada pista del disco se divide en un cierto número de sectores de n bytes cada uno. Los sectores están predefinidos por el fabricante del disco.

$$Capacidad = Q_{sectores} \cdot L_{sector}$$

En discos sectorizados los IRGS existen como separación entre sectores, pero no son considerados en absoluto por el programado. En este caso TODOS los bytes de la pista están disponibles para almacenar datos.

$$LP = L_{sector} * QSects_{xPista}$$

Intuitivamente una forma lógica de ubicar los sectores en la pista es numerarlos en forma correlativa. Sin embargo, esta disposición tiene un problema: era frecuente que una vez leído un sector el disco ya hubiera girado lo suficiente como para que el sector dos ya hubiera pasado por delante de la cabeza lectora-grabadora. Luego era necesario esperar casi una vuelta entera del disco para poder acceder al sector dos.

Para solucionar este problema se recurrió al *Interleave factor*. Lo que se hizo fue separar los sectores de forma tal de que dos sectores correlativos pudieran leerse uno a continuación de otro, el espacio entre sectores se llenaba precisamente con otros sectores, la distancia entre dos sectores se llama *Interleave factor*.

En general el tamaño de un sector es fijo para una unidad de discos, y suele ser un tamaño no muy grande, por ejemplo 512 bytes. Si se tuviera que acceder al disco cada 512 bytes, el número de accesos sería muy elevado y los tiempos de lectura y grabación muy altos. La mayoría de los sistemas operativos arreglan este problema mediante el uso de *clusters*.

Cluster: conjunto de sectores que pueden ser leídos con un único acceso. Es la unidad mínima de transferencia. Todo archivo ocupa por lo menos 1 *cluster* aunque en realidad sólo tenga un byte. Un cluster de mayor tamaño tiende a generar una mayor fragmentación, pero permite mejorar los tiempos por ser necesarios menos accesos al disco. Un cluster de menor tamaño reduce la fragmentación pero requiere muchos accesos al disco. En general para archivos grandes convienen clusters grandes y para archivos chicos convienen clusters chicos. En general el tamaño del cluster es fijo y no se puede variar.

Extent: espacio contiguo de almacenamiento dentro del disco. Cuando un proceso requiere escribir en un archivo, se aloca un extent. Cuando se vuelve a escribir en el archivo (luego de otras operaciones), la información se escribe a continuación de la anterior. Esto reduce la fragmentación y posiblemente también el esparcimiento de los archivos.

A la hora de almacenar los registros que conforman un archivo dentro del disco sectorizado se presentan dos alternativas.

1) *No spanned storage*: consiste en almacenar el siguiente registro en un nuevo sector. La información almacenada en el disco no se esparce entre sectores.

Ventaja: para acceder a un registro determinado solo necesitaríamos levantar del disco un sector, ya que estamos seguros de que el registro no se “esparcirá” entre sectores

Desventaja: desperdiciamos bytes.

2) *Spanned storage*: consiste en almacenar el siguiente registro a continuación del primero.

Ventaja: no desperdicia espacio ya que aprovechamos el 100% del tamaño de cada sector.

Desventaja: para acceder a la información del segundo registro deberíamos leer tanto el primer sector como el segundo.

Es esta última la forma en que el sistema operativo MS-DOS y las diversas versiones de Windows gestionan el almacenamiento en disco.

Organización D.O.S.: en MS-DOS se trabaja con discos sectorizados en los cuales se aplican todos los conceptos vistos, a excepción de:

- Los archivos se organizan como una cadena de bytes. Un archivo se compone de uno o más clusters que no tienen necesidad de ser contiguos. Todo archivo ocupa una cantidad entera de clusters, y al menos uno.
- El encadenamiento entre los clústeres de los archivos y el control de clústeres libres se registra en la **FAT** (*File Allocation Table*).
- Un archivo puede leerse en forma *random* o secuencial. Hay un acceso por clúster en lectura *random*, y un acceso por pista en lectura secuencial.
- Solo hay FRI en el último cluster del archivo (*slack*), y la FRE deja de ser relevante pues todo archivo se puede distribuir por los distintos clusters libres del disco.
- El DOS no asegura que si se graban 20 clusters en forma secuencial de un cierto archivo estos sean grabados en forma consecutiva en el disco. En general los archivos se encuentran fragmentados; los clusters que componen el archivo se encuentran distribuidos por distintas partes del disco.

Ventaja: se reduce la FRI.

Desventaja: se complica el acceso a un registro lógico en particular.

La cantidad de clusters necesarios para un archivo es: $Q_{clusters} = \frac{QRL * LRL}{L_{cluster}}$

El espacio alocado del archivo (longitud física del archivo) es: $EA = L_{cluster} * Q_{clusters}$

El *slack* es: $FRI = L_{cluster} * L_{clusters} - LRL * QRL$

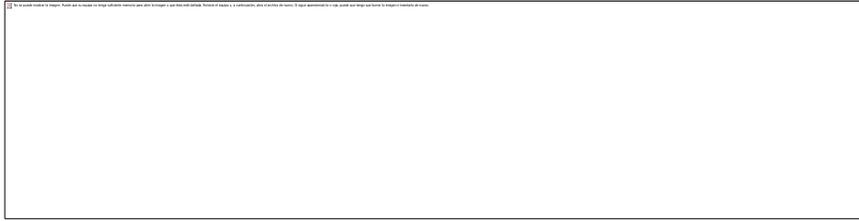
La FRI se produce sólo en el último cluster del archivo. La misma se puede estimar mediante

$$FRI \approx \frac{3}{4} * L_{cluster}$$

En los discos sectorizados los conceptos de t_{seek} y t_{search} son los mismos que en discos no sectorizados. La única diferencia es que en discos sectorizados se hace un acceso por cada cluster en lectura *random* y un acceso por cada pista del archivo en lectura secuencial. La lectura secuencial es notablemente más veloz que la lectura *random*.

CD-ROMs

En discos ópticos la lectura se hace de acuerdo a la reflexión de un rayo láser sobre una superficie tratada en la cual se graban los datos mediante marcas llamadas *pits*. A las superficies que quedan entre dos *pits* se las denomina *lands*. La transición de *land* a *pit* y vuelta a *land* representa un uno. Un cierto período de tiempo sin transiciones representa un cero.



Debido a las características de los lectores no es posible mediante este esquema que existan dos unos seguidos en la superficie del disco. El problema se corrige empleando un sistema de traducción de los bits leídos desde el disco a bytes 'comunes'. Este sistema traduce 14 bits leídos del CD en un byte. El sistema se conoce como *eight to fourteen modulation* (EFM).

Cada CD tiene una única pista en espiral sobre la cual se graban los datos. Sobre esta pista se estructuran sectores todos ellos de la misma longitud lo cual permite grabar todos los sectores con la misma densidad de grabación. Este tipo de organización se denomina *CLV (Constant Linear Velocity)* pues la velocidad lineal de lectura es constante (espero hayan notado porque es que esto debe ser así) y se diferencia de la organización *CAV (Constant Angular Velocity)* empleada en los discos en donde la velocidad de rotación es constante.

El esquema CLV perjudica el acceso *random* a los datos ya que para "saltar" desde una posición del disco hasta otra la unidad debe recalcular la velocidad a la cual deberá girar el disco. Esto se hace utilizando prueba y error. Los *seeks* en un CD son muy costosos, esto se debe a que los CDs de datos se basan en la industria ya existente para los CDs de música, donde la operación más frecuente es la lectura secuencial.

Un sector de un disco CLV se referencia mediante un número de la forma MM:SS:XX donde MM es el minuto, SS es el segundo, y XX el sector.

Cada CD puede almacenar aproximadamente 74 minutos de audio, y sabiendo que cada segundo se divide en 75 sectores nos da un total de 333000 sectores. Cada sector a su vez puede almacenar 2352 bytes, lo cual representa una capacidad de 747 megabytes.

CDs de Audio

- Capacidad: la capacidad estándar de un CD de audio es de 74 minutos.
- En un CD de audio el promedio de fallas es de un byte cada dos CDs.

CDs de Datos

- Capacidad: por cada sector solamente 2048 bytes pueden contener datos. La capacidad de un CD de datos es entonces de 650 Mb.
- Con el agregado de los códigos correctores el promedio de fallas es de un byte equivocado cada 20.000 discos.

La organización de un sector de 2352 bytes para grabar datos es la siguiente:



SYNC	12 bytes destinados a sincronización.
SECTOR ID	4 bytes, cada sector se identifica por número de minuto, número de segundo y número de sector. (Recordar 75 sectores por segundo)
DATOS	2048 bytes.
ED	4 bytes para detección de errores.
null	8 bytes que no se utilizan.
Error correction	276 bytes con códigos auto-correctores.

DVD-ROMs

Los **DVD** (*Digital Versatile Disc*) fueron creados en 1996, y surgieron como una necesidad de incrementar el mercado del video, ya que en un DVD la calidad de audio e imagen es alta. Actualmente también se utiliza para el almacenamiento de software en general (*office suites*, video juegos, etc.).

Similitudes con los CDs	Diferencias con los CDs
<ul style="list-style-type: none"> - ambos son discos de plástico de 120 mm de diámetro, 1.2 mm de espesor - ambos necesitan un láser para leer la información almacenada - similar tiempo de acceso 	<ul style="list-style-type: none"> - DVD provee más capacidad - mayor coeficiente de transferencia que el CD - las pistas están más juntas, por lo tanto permiten más pistas por disco - los pits son mucho más chicos, por lo tanto permite más pits por pista - el DVD ha hecho más eficiente la estructura de la información grabada en el disco

Hay cinco formatos de DVD:

1. DVD-ROM: medio de alta capacidad de almacenamiento de datos
2. DVD-Video: medio de almacenamiento digital de películas
3. DVD-Audio: formato de almacenamiento de audio únicamente, similar al CD de audio
4. DVD-R: ofrece un formato de almacenamiento de una sola grabación y múltiples lecturas
5. DVD-RAM: primer formato regrabable de DVD; luego compitió con DVD-RW y DVD+RW

Existen cuatro versiones del DVD:

1. DVD-5: de un solo lado y una sola capa, con capacidad de 4,7 GB
2. DVD-9: de un solo lado y dos capas, con capacidad de 8,5 GB
3. DVD-10: de dos lados y una capa, con capacidad de 9,4 GB
4. DVD-18: de dos lados y dos capas, con capacidad de 17 GB

La tecnología de capa dual tiene una capa reflectante abajo, cubierta por una capa semirreflejante. Dependiendo de la distancia focal del láser, el haz se refleja en una capa o en la otra. La capa inferior necesita fosos y *lands* un poco mayores para que su lectura sea confiable, por lo que su capacidad es menor que la de la capa superior.

Además de sus cinco formatos físicos, el DVD también tiene varios formatos de aplicación como DVD de Video y DVD de Audio. La consola de juegos PlayStation 2 de Sony es un ejemplo de un formato especial de aplicación.

RAIDs

RAID (*Redundant Array of Inexpensive Disks*, luego sería *Redundant Array of Independent Disks*): sistema de almacenamiento que usa múltiples discos duros entre los que distribuye o replica los datos. Según la configuración (nivel) pueden brindar: mayor integridad, mayor tolerancia a fallos, mayor *throughput* y mayor capacidad. La idea es que combinando varios dispositivos de bajo costo en un conjunto ofrecen mayor capacidad, fiabilidad, velocidad o una combinación de éstas, que un solo dispositivo de última generación y coste más alto.

Los más comúnmente usados son:

- RAID 0: Conjunto dividido
- RAID 1: Conjunto en espejo
- RAID 5: Conjunto dividido con paridad distribuida.

Nivel	Imagen	Descripción	Ventajas	Desventajas
0	<p>RAID 0</p>	<p>Distribuye los datos equitativamente entre dos o más discos sin información de paridad o redundancia.</p> <p>Una buena implementación de un RAID 0 dividirá las operaciones de lectura y escritura en bloques de igual tamaño y los distribuirá equitativamente entre los dos discos.</p>	<p>Aumenta la velocidad de acceso a los discos.</p>	<p>No ofrece tolerancia al fallo.</p>
1	<p>RAID 1</p>	<p>Crea una copia exacta (o espejo) de un conjunto de datos en dos o más discos.</p>	<p>El rendimiento de lectura se incrementa aproximadamente como múltiplo lineal del número de copias. Es tolerante a los fallos.</p>	
2	<p>RAID 2</p>	<p>Divide los datos a nivel de bits. Utiliza el código de Hamming. Cada byte de datos almacenado está "repartido" entre los discos del RAID.</p>	<p>Permite tasas de transferencias extremadamente altas.</p>	

3	<p>RAID 3</p>	<p>Usa división a nivel de bytes con un disco de paridad dedicado.</p>		<p>Cualquier operación de lectura o escritura exige activar todos los discos del conjunto.</p>
4	<p>RAID 4</p>	<p>Usa división a nivel de bloques con un disco de paridad dedicado.</p>	<p>Puede servir varias peticiones de lectura simultáneamente.</p>	
5	<p>RAID 5</p>	<p>Usa división de datos a nivel de bloques distribuyendo la información de paridad entre todos los discos miembros del conjunto.</p>	<p>Coste de redundancia.</p>	<p>Las escrituras son costosas en términos de operaciones de disco y tráfico entre los discos y la controladora.</p>

Corrección de errores

Hay códigos que sirven para protegerse contra los errores ocasionales que pueden cometer las computadoras. Para eso, se añaden bits extra de una forma especial a cada palabra de la memoria. Cuando se lee una palabra de la memoria, se verifican los bits adicionales para ver si ha ocurrido algún error.

m bits de datos

r bits de verificación

n longitud total ($n = m + r$)

Distancia de Hamming: número de posiciones de bits en las que dos palabras de código difieren.

En una palabra de memoria de m bits, están permitidos los 2^m patrones de bits, pero por cómo se calculan los bits de verificación, sólo 2^m de las 2^n palabras de código son válidas.

Códigos auto correctores de Hamming

$$1 + m + r \leq 2^r$$

Para saber dónde está el bit corrupto, se suman los bits de paridad cuya suma resulte incorrecto.

Ejemplo:

Bit 1) Suma ok

Bit 2) Error

Bit 3) Suma ok

Bit 4) Error

Bit 2 + Bit 4 --> $2 + 8 = 10$ (Posición 10 está el bit corrupto a ser corregido).

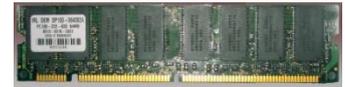
Unidad 6: Periféricos

Periférico: dispositivo accesorio a una computadora, que no es parte de ella, aunque sí depende de la misma. Estos se conectan al computador mediante puertos.

Periféricos de almacenamiento

Unidad de estado sólido (Solid-State Drive o SSD): dispositivo de almacenamiento de datos que usa una memoria no volátil, como la memoria flash, o una memoria volátil como la SDRAM, para almacenar datos, en lugar de los platos giratorios magnéticos encontrados en los discos duros convencionales. Hay de dos tipos:

- **SDRAM (Synchronous Dynamic Random Access Memory):** memoria dinámica de acceso aleatorio (DRAM) que está sincronizada con el bus del sistema. Este tipo tiene una batería incorporada.



- **Memoria flash:** memoria no volátil que no requiere baterías, es más lento que la SDRAM, son resistentes a los golpes. Se utilizan en tarjetas de memoria, memorias USB, reproductores de MP3, etc.

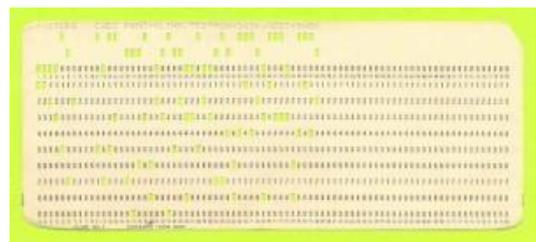


Ventajas de SSDs sobre Discos Duros	Desventajas de SSDs sobre Discos Duros
<ol style="list-style-type: none">1. Velocidad de lectura 10x, velocidad de escritura 100x2. Consumen menos energía3. Son silenciosos4. Son más seguros (en promedio duran 2 millones de horas contra el millón de los discos duros)5. El tiempo de acceso es constante6. Son livianos7. Son menos vulnerables a los golpes	<ol style="list-style-type: none">1. Son más costosos por gigabyte2. Tienen menor recuperación ante fallas3. Son más vulnerables a campos eléctricos o magnéticos4. Tienen menor capacidad5. Tienen menor vida útil

Almacenamiento offline: es un procedimiento que consiste en almacenar información en un medio que no está bajo el control de la CPU. La información se guarda en un dispositivo auxiliar hasta que la persona conecta el dispositivo y se guarda en el mismo. Es decir que este proceso requiere de la intervención humana.

Periféricos de entrada

1. **Tarjetas perforadas:** es un sistema de entrada, procesamiento y almacenamiento de datos utilizado en el siglo 20. Consiste en un pedazo de papel que contiene información representada por la presencia o ausencia de agujeros en posiciones fijas. La tarjeta tiene 12 filas y 80 columnas. Este sistema fue desarrollado por Herman Hollerith para ser utilizado en las oficinas de censos de Estados Unidos. Ventajas: baratas. Desventajas: es muy lento, son incómodas, si se desordenan puede haber problemas, se gastan con el uso.



2. **Teclados:** es un mecanismo que posee botones con letras o números. Los hay de dos tipos: electromecánicos y con membranas, que son flexibles. Una de sus desventajas es la lentitud, ya que sólo se pueden ingresar, en promedio, hasta 85 caracteres por minuto.

3. **Punteros:** son dispositivos que permiten el ingreso de datos espaciales a la computadora; esto se logra mediante la detección de movimiento a lo largo de una superficie o bien mediante el cálculo de ángulos de reflexión. Los movimientos de estos dispositivos se reflejan en una pantalla con movimientos de un “cursor”.

- Tecnología *Touch*: los dispositivos como las tabletas digitalizadoras, los lápices ópticos y las pantallas táctiles son ejemplos de estos dispositivos.

- Pantallas táctiles: son pantallas de vidrio que están cubiertas por materiales que detectan la posición del toque del dedo midiendo los cambios en la corriente eléctrica.

- Lápices ópticos: son lápices que permiten al usuario apuntar a objetos o dibujar en pantallas, de forma muy similar a la pantalla táctil pero con más precisión. Estos dispositivos detectan pequeños cambios en el brillo de un punto de la pantalla. Entonces la computadora puede resolver la posición (X,Y) del lápiz.



- *Mouse*: es un dispositivo puntero que detecta movimientos en dos dimensiones relativos a su superficie de apoyo y los traduce en el movimiento de un cursor en una pantalla.

- *Joystick*: es un dispositivo de entrada que consiste en una vara en una base que direcciona un ángulo, y puede tener botones. Se utilizan muy frecuentemente en video juegos.

4. **Reconocimiento de escritura:** es la capacidad de un ordenador de recibir entradas manuscritas. Una pantalla táctil detecta los movimientos que el usuario realiza mediante una pluma especial, quien debe escribir de una forma determinada.

5. **Scanner:** es un dispositivo que escanea imágenes, texto, e incluso objetos, y lo convierte en una imagen digital. Estos dispositivos tienen resoluciones que se miden en “ppi” (pixels per inch); cuando se duplica la resolución se cuadruplica el tamaño del archivo creado. A modo de referencia, una hoja de 25x30 genera un archivo de 1 megabyte. También existe la posibilidad de reducir la resolución comprimiendo el archivo, lo que se traduce en pérdida de calidad.

6. **Caracteres magnéticos** (*Magnetic Ink Character Recognition* o MICR): se utilizan mucho en la industria bancaria para el procesamiento de cheques. Estos caracteres son fácilmente reconocibles por personas; se imprimen en tipografías especiales con tinta magnética, que generalmente contienen óxido de hierro. Las tipografías más comúnmente utilizadas son:

- CMC-7: se utiliza en Argentina y Francia.

1 2 3 4 5 6 7 8 9 0

- E-13B: se utiliza en India, Estados Unidos, Canadá y el Reino Unido.

1 2 3 4 5 6 7 8 9 0

7. Reconocimiento de caracteres (Optical Character Recognition u OCR): es la traducción mecánica o electrónica de imágenes que contienen texto (escrito a mano o con computadora) a archivos de texto de máquina. Se utiliza para convertir libros a formato digital, digitalizar bases de datos, etc. Este sistema requiere de una calibración para leer una tipografía específica.

8. **Código de barra:** es una representación de datos que puede ser leída por computadoras, que contiene información sobre el objeto que tiene dicho código. Estos son escaneados por *scanners* ópticos, y más recientemente también por teléfonos inteligentes. Su uso es particularmente famoso en los supermercados. *Universal Product Code* (UPC) es una simbología especial que se usa para seguir la ruta de, por ejemplo, productos en un comercio. La simbología UPC-A está formada por 95 bits o 30 barras verticales.



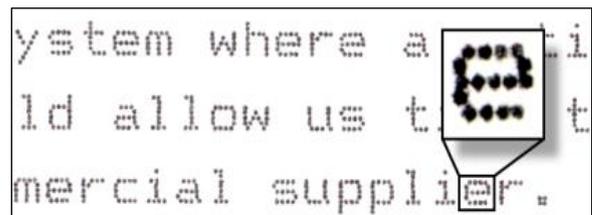
9. **Dispositivos de video:** se utilizan para digitalizar imágenes o video del mundo exterior a un formato que puede interpretar una computadora. Algunos ejemplos son: cámaras digitales, *webcams*, lectores de barras, lectores de huellas dactilares, etc.

10. **Dispositivos por voz:** se utilizan para capturar (actualmente también para crear) sonidos. Los dispositivos inteligentes que reconocen el habla convierten palabras habladas por personas en texto. Estos dispositivos deben ser “entrenados” para reconocer la voz del hablante. Son especialmente útiles para: personas con discapacidades, en lo militar, en la telefonía, etc. Estos dispositivos aprenden con el tiempo, es decir, van actualizando su diccionario de palabras interno, y así reducen el margen de error.

Periféricos de salida

1. **Impresoras:** son periféricos que producen texto y/o gráficos de documentos en forma de papel o transparencias que estaban almacenados en forma electrónica. En general, las impresoras son lentas (30 páginas por minuto es considerado “rápido”) y el costo por hoja es relativamente alto. Las hay de dos tipos: de impacto y sin impacto. La diferencia entre ambos es simplemente si el papel entra o no en contacto con la cinta entintada. Las siguientes son impresoras sin impacto:

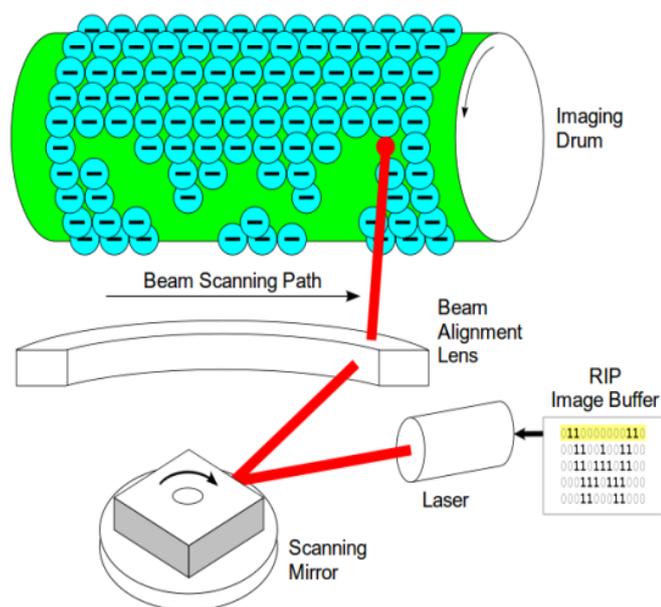
1. **Matriz de puntos** (*Dot matrix*): posee un cabezal que se desplaza de izquierda a derecha sobre la página, e imprime por impacto de tinta. Estas máquinas son baratas, muy duraderas y pueden imprimir copias carbón, pero con el tiempo los bastones pierden precisión, suelen ser ruidosas y lentas, y sólo pueden imprimir texto y gráficos simples.



2. **De chorro de tinta** (*Inkjet*): funcionan expulsando gotas de tinta de diferentes tamaños sobre el papel. Son muy populares hoy en día. Entre sus ventajas podemos mencionar su bajo costo, que son silenciosas y fáciles de implementar, y su tamaño pequeño comparado con otros tipos de impresoras como las láser. Como contrapartida mencionamos que son lentas, tienen un alto costo en concepto de cartuchos (se consumen rápido y son costosos), etc.

3. **Láser** (*Laserjet*): su principal ventaja radica en que produce documentos de alta calidad a una velocidad notable. Su uso es recomendado a usuarios con una alta cantidad de impresiones y con un uso intermitente. El dispositivo de impresión consta de un tambor fotoconductor unido a un depósito de tóner y un haz láser que es modulado y proyectado a través de un disco especular hacia el tambor fotoconductor. El giro del disco provoca un barrido del haz sobre la generatriz del tambor. Las zonas del tambor sobre las que incide el

haz quedan ionizadas y, cuando esas zonas (mediante el giro del tambor) pasan por el depósito del tóner atraen el polvo ionizado de éste. Posteriormente el tambor entra en contacto con el papel, impregnando de polvo las zonas correspondientes. Para finalizar se fija la tinta al papel mediante una doble acción de presión y calor.



4. **Térmicas:** estas impresoras producen imágenes mediante el calentamiento selectivo de papel sensible al calor. Éste es un sistema muy empleado en terminales de venta, cajeros automáticos, para imprimir *tickets* o recibos, o para crear etiquetas. Su costo es muy bajo, pero con este mecanismo sólo puede imprimirse el color negro, y son lentas.

2. **Microfilm:** son películas o papeles que contienen micro-reproducciones de documentos. Las imágenes de microfilm suelen ser 25 veces más pequeñas que el documento original. Sus ventajas: son compactas, baratas, duraderas (una película puede durar 500 años si se guarda en las condiciones adecuadas), y fácilmente visibles (basta con una lupa o un proyector). Sus desventajas: son muy pequeñas para ver naturalmente.



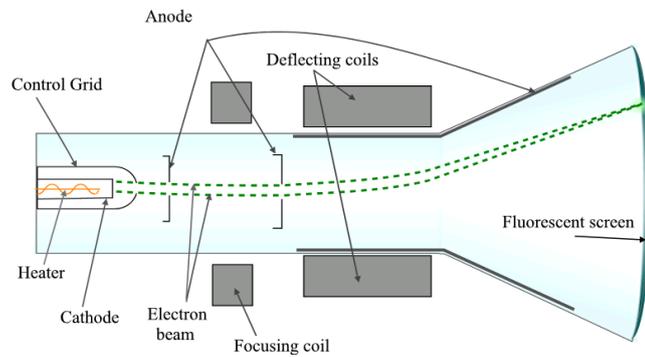
3. **Plotters:** son periféricos que imprimen gráficos vectoriales de alta resolución.

1. De mesa: consiste en un lápiz que se mueve a lo largo y a lo ancho de una superficie. Dado que requieren muchos movimientos mecánicos, son lentas.

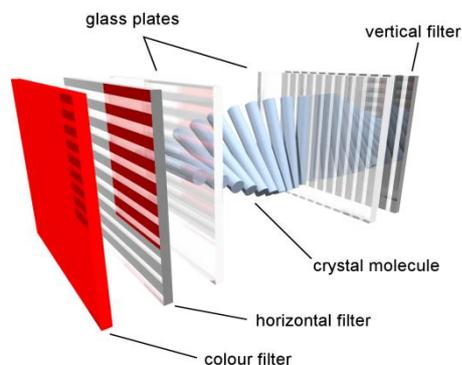
2. Cilindros: son similares a los plotters de mesa, con la diferencia de que la superficie donde se dibuja es un cilindro, y el cabezal se mueve sólo en una dirección (hacia arriba o hacia abajo).

4. **Pantallas:** son equipos eléctricos que representan imágenes transmitidas electrónicamente para su recepción visual, sin producir un archivo permanente.

1. **Tubos de rayos catódicos** (*Cathodic Ray Tubes* o CRT): es un tubo vacío que contiene una pistola de electrones y una pantalla fluorescente, con aceleradores internos o externos que desvían el haz de electrones, y así crea imágenes en forma de luz en la pantalla.



2. **De cristal líquido** (*Liquid Crystal Display* o LCD): es una pantalla plana y delgada formada por píxeles rellenos con cristales líquidos y colocados en frente a una fuente de luz para producir imágenes color o monocromadas. Sus ventajas: son compactas, livianas, más baratas, más duraderas, y más eficientes energéticamente que los CRTs. Su funcionamiento fue descubierto en 1888.



5. **Voz** (*Text to Speech* o TTS): su principal ventaja radica en la rapidez; un humano responde más rápidamente a estímulos auditivos que visuales. Su complejidad varía desde palabras pregrabadas, frases pregrabadas, o fonemas.

6. **Entorno virtual**: son entornos simulados por computadora que simulan la presencia física de una persona en lugares del mundo, reales o imaginarios. Se utilizan, por ejemplo, en el pilotaje de transportes, en simulaciones militares, etc.

Códigos de caracteres

Código de caracteres: sistema que empareja caracteres de un repertorio dado con números naturales, octetos, pulsos eléctricos, etc., para facilitar la transmisión de datos. Algunos de los ejemplos más comunes son:

- **ASCII** (*American Standard Code for Information Interchange*): cada carácter ocupa 7 bits, con lo cual están definidos 128 caracteres.
- **EBCDIC** (*Extended Binary Coded Decimal Interchange Code*): cada carácter ocupa 8 bits, con lo cual están definidos 256 caracteres. Se utiliza principalmente en máquinas IBM. No posee ventajas técnicas por sobre ASCII o Unicode.
- **Unicode**: es el formato más utilizado hoy en día. Su última versión contiene un repertorio de más de 109.000 caracteres, cubriendo 93 lenguajes distintos.

Módem

Módem (*modulador-demodulador*): es un dispositivo que modula señales para codificar información digital, y que demodula dichas señales para decodificar la información.

El modulador emite una señal denominada portadora. Generalmente, se trata de una simple señal eléctrica sinusoidal de mucha mayor frecuencia que la señal moduladora. La señal moduladora constituye la información que se prepara para una transmisión (un módem prepara la información para ser transmitida, pero no realiza la transmisión). La moduladora modifica alguna característica de la portadora (que es la acción de modular), de manera que se obtiene una señal, que incluye la información de la moduladora. Así el demodulador puede recuperar la señal moduladora original, quitando la portadora. Las características que se pueden modificar de la señal portadora son:

- Amplitud, dando lugar a una modulación de amplitud (AM/ASK).
- Frecuencia, dando lugar a una modulación de frecuencia (FM/FSK).
- Fase, dando lugar a una modulación de fase (PM/PSK)

También es posible una combinación de modulaciones o modulaciones más complejas como la modulación de amplitud en cuadratura.

Hoja de fórmulas

Unidades de medida

1 pie = 12 pulgadas = 20,48 cm.

1 pulgada = 2,54 cm

Cintas

$$FB = \frac{QRL}{QRF} = \frac{LRF}{LRL}$$

$$LRF_{bytes} = LRF_{pulgadas} * \delta$$

$$FRI = QRF * LRF - QRL * LRL$$

$$FRE = L_{cinta} - QRF_{x\ cinta} * (IBG + LRF)$$

$$FRS = QRF * IBG$$

$$Total\ archivo = (Q_{carretes} - 1) * L_{carrete} + QRF_{ult\ carrete} * (LRF + IBG)$$

$$Vt = Va * \delta$$

$$T_{1RF} = T_{IBG} + T_{transf}$$

$$T_{transf} = \frac{LRF}{Vt}$$

$$T_{archivo} = T_{1RF} * QRF$$

Discos no sectorizados

$$FB = \frac{LRF}{LRL} = \frac{QRL}{QRF}$$

$$QRF_{x\ pista} = \left\lfloor \frac{LP[bytes]}{LRF + IBG} \right\rfloor$$

$$QP = \left\lfloor \frac{QRF}{QRP} \right\rfloor$$

$$Q_{cilindros} = \left\lfloor \frac{QP}{QP_{x\ cilindro}} \right\rfloor$$

Max LRF = LP - IBG (si el buffer no restringe)

$$QBD_{x\text{ pista}} = FRE + FRS$$

$$QBD_{x\text{ archivo}} = (LP * QP) - (LRL * QRL)$$

$$FRI = (QRF * FB - QRL) * LRL$$

$$FRS = QRP * IBG$$

$$FRE = \begin{cases} LP - QRP(LRF + IBG) \\ LP - QRP_{ultima}(LRF + IBG) \end{cases}$$

$$\%FR = 100 * \frac{FRE + FRS}{LP} = 100 * \frac{LP - QRP * LRP}{LP}$$

$$T_{1RF} = T_{seek} + T_{search} + T_{transf}$$

$$T_{seek\ prom} = \frac{1}{3} * T_{seek\ max}$$

$$T_{seek\ max} = QP * T_{seek\ 1\ pista}$$

$$T_{search} = \frac{1}{2} * T_{1\ vuelta}$$

$$T_{transf} = \frac{LRF}{Vt}$$

$$Vt = \frac{LP}{T_{1\ vuelta}}$$

Discos sectorizados

$$Capacidad = Q_{sectores} * L_{sector}$$

$$LP = L_{sector} * Q_{sectores\ x\ pista}$$

$$Q_{clusters\ necesarios} = \left\lceil \frac{QRL * LRL}{L_{cluster}} \right\rceil$$

$$Espacio\ alocado = Q_{clusters} * L_{cluster}$$

$$FRI = Slack = Q_{clusters} * L_{cluster} - LRL * QRL$$

$$FRI \approx \frac{3}{4} * L_{cluster}$$