Clase 7:
Temas: **Formatos de instrucción**, **modos de direccionamiento**.
Formatos de instrucción:
Una **instrucción** posee **4 elementos**, **referencia** a la **siguiente instrucción**, **referencia** al operando **resultado**, **referencia** al operando/s **fuente**, **referencia** al código de **operación**.
Al **esquema** que **determina** cómo está **codificado** los mismos se le conoce como **formato de instrucción**, que suele ser **representado** por un **Mnémonico** (ADD, SUB), donde **cada campo** del código **es un elemento**.

Existen **2** tipos de **instrucciones**: De **alto nivel**, que serían lenguajes comunes de programación, y **de lenguaje de máquina**, que serían más concisas. Se tendrían, **con todas** las **instrucciones** de lenguaje **de máquina**, poder **hacer todas** aquellas de **alto nivel**, ya que cuando uno compila las de alto nivel las pasa a lenguaje de máquina.
Podemos **categorizar** las **instrucciones** en:
Procesamiento de **datos** (aritméticas lógicas), **Almacenamiento** de **datos** (transferencias), **Instrucciones E/S** (entre cpu y externos), y **control**.

Máquinas: Una máquina se puede **definir** por la **cantidad** de **referencias** explícitas a **direcciones** de **memoria** que poseen en sus instrucciones:
4 direcciones tienen: Instrucción, dirección resultado, (dirección) operando 1,(dirección) operando 2, dirección próxima instrucción.
3 direcciones tienen: Instrucción, dirección resultado, (dirección) operando 1, (dirección) operando 2 (La **próxima instrucción** se pasa a guardar en el **PC** de la **CPU**).
2 direcciones tienen: Instrucción, (dirección) operando 1, (dirección) operando 2. (**Operando 1** actúa como **dirección de resultado**, usa **registros temporales** donde se envía el **operando 1** para hacer las acciones).
1 dirección tienen: Instrucción, (dirección) operando 1. (Nuevo registro especial, el **acumulador**, usa **load** y **store**).

En resumen:
El conjunto de instrucciones es el medio que tiene el programador para controlar la CPU.
Hay que **tener en cuenta**:
Tipos de operaciones: cuántas y cuáles.
Tipos de datos: cuáles son.
Formato de instrucciones: longitud (bits), No de direcciones, tamaño de cada campo.
Registros: cantidad que se pueden referenciar mediante instrucciones y su uso.
Direccionamiento: la manera de especificar la dirección de un operando o una instrucción (la próxima).

Modo de **direccionamiento**:
Para **reducir** la cantidad de **bits** a **especificar** dónde están los datos hacemos esto usamos **registros**, por ejemplo de haber 32 registros vs 64k espacios de memoria, se necesitan **menos bits** para **direccionar 32** que 64k.
Otra forma es especificando operandos implícitamente, corte reg2+reg2+fuente1; siendo fuente 1 el acumulador.
En resumen estos modos están para **reducir** la cantidad de **bits** de **instrucción**, contener direcciones que no se sepan hasta la ejecución, que sea más **eficiente** el **manejo** de **datos**, por ejemplo con arreglos.
Modos:
Inmediato: **Operando** se encuentra **en** la **memoria** de la **instrucción**, limitado por tamaño de campo, y generalmente no se usa porque se consideran “Números mágicos”, es decir no tienen explicaciones de para qué sirven, en cambio si nombramos variable si tendrían.

Directo (por registro o memoria): Contiene **en memoria de instrucción** la **dirección** del **operando**, de ser memoria sería el nombre de una variable, o sino un registro, que sería más rápido pero hay pocos.
Indirecto (por memoria o registro): En **memoria** de **instrucción estaría** la **dirección del registro o dirección** de **memoria contendrá** la **dirección** de donde está el **valor**, es decir actuaría por puntero, esto sirve para aumentar el espacio de direccionamiento porque una dirección de menos bits apunta a una mayor.
Desplazamiento/relativo: Requiere que la instrucción tenga **dos campos de direccionamiento** menos cuando uno está implícito, el **primero** contendrá una **dirección**, y el **segundo** un **desplazamiento**. Sumando la dirección de memoria y el desplazamiento conseguimos la dirección de memoria efectiva (donde está el dato). Es potente porque no te limita con más accesos a memoria como al indirecto, sino que sumamos y eso no retrasa. Para el programador sirve para acceder a vectores, matrices, o listas. Tipos más usados:

- **Relativo/Relativo a PC**: Completa **mentira** de que es de los **más usados** porque para encontrar ejemplos estas minimo 10 minutos. Equivalente a registro base, con la diferencia que **utiliza** el **PC** de manera **implícita** en la **instrucción** y solo hay que **expresar con** una **etiqueta** el **desplazamiento para calcular** la **dirección** de **memoria efectiva**. Ejemplo: **Opcode R, A. R** contendría el **PC** y **A** el **desplazamiento**, y eso te daría el lugar del operando
- **Relativo a Registro base/Registro base**: La **dirección** de **memoria** se almacena **en** el **registro base**, y **desplazamiento** está **solamente** en la **instrucción**. El **RB** a veces está **implícito**, y solo hay que definir el desplazamiento. **MOV AL, [BX+1]**
- **Indexado**: La **dirección** de **memoria** se encuentra explícitamente **en** la **instrucción** y el **desplazamiento** se almacena **en** el **registro índice** (**RI**). Indexado directo de forma: MOV AL, Vector[SI] Ejemplo: **Mov ax, x[1].**

Stack: El stack que es un arreglo lineal de espacios de memoria se usa como una lista donde se van “apilando” los nuevos datos arriba del otro, así el primero en entrar sale el último. Tiene un SP para apuntar al tope del stack.

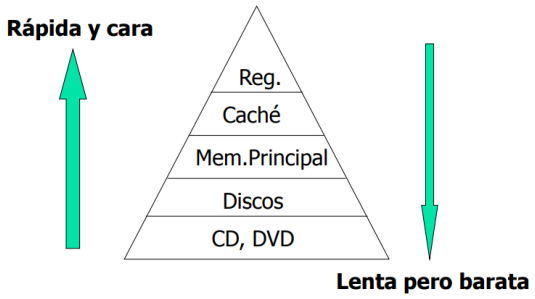
Clase 8:
Temas: Organización de **registros**, **instrucciones**.
Organización de **registros**:
2 tipos:
-**Visibles** al **usuario** y **usados** por el **programador**.
-Registros de control y estado, **usados** por **unidad control** para controlar la operación de CPU, **invisibles** al **programador**.
4 tipos de registros **visibles** al **usuario**:
Propósito **general**, **datos**, **dirección**, códigos de **condición**.
Los **registros generales** son **flexibles**:
-Cualquier registro general **puede contener** el **operando** para **cualquier** operación.
-**Algunos** pueden estar **limitados** a **solo punto flotante** o **algo así**.
-Se **pueden** usar para **direccionamiento**.
-**Sólo** para **datos o** sólo para **direcciones**.
-Los **registros** de dirección **pueden ser asignados** para un **mdd**.
Conviene tener de **8 a 32 registros**, ya que la cantidad aumenta los bits para referenciarlos, **además tener** algunos **especializados** que no se pueden usar son útiles para ahorrar bits. (Acumulador).
Si el **registro** fuese **de direcciones** tiene que ser **capaz** de **almacenar** la **dirección más grande**. Si fuese **de datos** debe estar **habilitado** para la **mayoría** de tipos de **datos**.
Algunas de estas **máquinas permiten usar 2** registros **contiguos como** uno **solo**, para almacenar doble longitud.

Las **flags** son bits **establecidos por** la **cpu** como **resultados** de las **operaciones**, que pueden ser **usados** para **definir bifurcaciones** en la programación, aunque los mismos no se cambian directamente por el programador (generalmente).
Registros de **control y estado**:
-Se usan para **controlar** las **operaciones** de la **cpu** y **no** son generalmente **visibles**.
-**4** esenciales: (Program counter, apunta a la siguiente instrucción) **PC**, (Instruction register, guarda la instrucción actual) **IR**, (Memory access register, guarda la dirección de memoria de lo leído) **MAR**, (memory buffer register, guarda la información que va hacia o desde la cpu) **MBR**.

Acá hablaría de las **instrucciones** específicas de intel, sin embargo ya sabemos casi todo porque usamos MSX88, peros solo voy a poner **ejemplos** que hay allí:
MOV [BX+D], AL (Direccionamiento base+índice)
MOV [BX+2Ah], AX (Relativo por registro *Base* {el base no estaba, pero asumo que es es})
MOV [BX+DI+2Ah], AX (Relativo base+indice {DI es el registro índice destino, si aparece SI, es el registro índice fuente})

Cosas a **tener en cuenta** cuando **diseñando** una **instrucción** (su formato):
Longitud, **números** de **bits** (ancho de banda de la memoria), **velocidad** de **memoria/procesador**, (instrucciones más cortas hacen parecer más rápido al procesador).
Se **necesitan**:
Suficientes **bits para** expresar **todas** las **operaciones deseadas**.
Dejar **bits libres** para el futuro.
Cantidad de **bits** de **datos**.

Clase 9:
Temas: Subsistema de **Memoria**, **Organización** de **Memoria** Principal.
Se dice que la **velocidad** del **procesador** se **duplica** cada **18 meses** sin variar su precio. Mientras que la **capacidad** de **memoria** se **cuadruplica** cada **36 meses**, la **velocidad** de memoria es de **10% anual**.
Hay **brecha** entre las **velocidades** del **procesador** y de la **memoria**, así que para que las arquitecturas pueden trabajar con estas diferencias usan distintos tipos de memoria en diferentes ratios para simular una de velocidad aceptable, por ejemplo registros y discos en conjunto.
Jerarquía de **memoria**:



Los discos duros y ssd irían entre discos y memoria principal.
La memoria de la parte superior se accede más (es un tema de estadístico).

En resumen: La **memoria** de la **computadora** tiene **tecnologías diferentes**, basadas en **fundamentos físicos distintos**, **localizadas** en **lugares distintos**. Y el objetivo de esto es tener **capacidad** de almacenamiento **alta**, y peor tiempo de **acceso corto**.

Tipos de memoria	Tiempo de acceso	Tamaño típico
Registros	1 ns	1 KB
Caché	5-20 ns	1 MB
Mem. Principal	60-80 ns	1 GB
Discos	10 ms	160 GB

Características de las memorias:
Duración:

- **Volátil**: Ram
- No **volátil**: discos, cintas
- **Permanentes**: ROM, EPROM.

Modo de **acceso**:

- Por **palabra**: Memoria principal (ram).
- Por **bloque**: Discos, caché.

Velocidad:

- **Semiconductoras**: **Tiempo** de **acceso** (tiempo máximo que transcurre desde que se inicia la operación de lectura/escritura hasta que se almacena/lee el dato). **Tiempo** de **ciclo** (tiempo mínimo que tiene que haber entre dos operaciones sucesivas sobre la memoria). El tiempo de ciclo es mayor al de acceso.
- **Magnéticas**: **Tiempo** de **acceso** (Tiempo de posicionar el cabezal + tiempo de latencia + tiempo de lectura). **Velocidad** de **transferencias**: Bytes/seg.

Métodos de **acceso**:

- Acceso **aleatorio**: **Tiempo** de **acceso independiente** de accesos anteriores, ejemplo ram.
- Acceso **secuencial**: El **acceso** se hace en **secuencia lineal** específica. Variable. Ejemplo cintas.
- Acceso **directo**: Se **acceden** por **bloques** que **tienen direcciones únicas** basadas en su posición física, variable, ejemplo discos magnéticos.
- Acceso **asociativo**: Memoria **caché**.

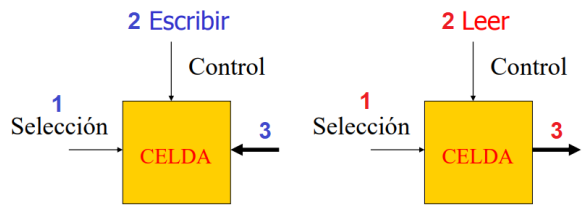
Memoria de **acceso aleatorio**:
RAM, se puede **acceder** a **cualquier celda** de memoria en un **mismo tiempo**. Tiene **una parte basada** en **flip flops** (**SRAM (STATIC RANDOM ACCESS MEMORY)**) (mas cara pero mas rapida) y **otra basada** en **transistores** (**DRAM (DYNAMIC RANDOM ACCESS MEMORY**) (Mas barata).
La **ROM** es del **mismo tipo**.
Los **transistores** de la **dram** se **deben refrescar** porque pierden carga, pero son más chicos, así que **almacenan más**, la **SRAM** se **usa como** memoria **caché**.
3 **propiedades** de todas las memorias **semiconductoras**:

1. **Solo** puede ser **1 o 0**.
2. Se **pueden escribir** en ellas.
3. Se **pueden leer**.

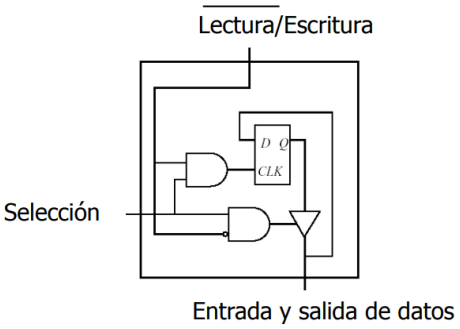
La **celda** de **memoria** tiene **3 terminales** para llevar una señal eléctrica:

- **Selección**
- **Control**: Decide si lectura o escritura.
- **E/L** de datos

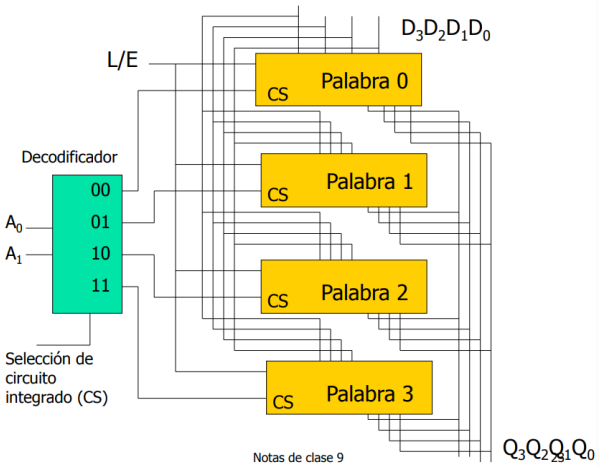
Visión gráfica: Llega la señal que la selecciona, otra que le dice que hacer, y después le llega o manda la información.



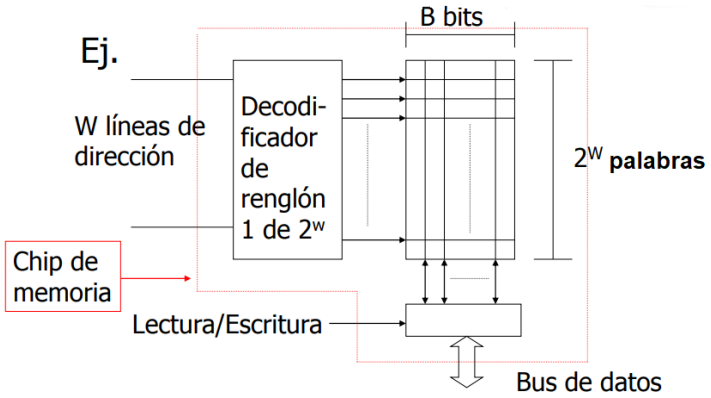
Celda de memoria flip flop D:

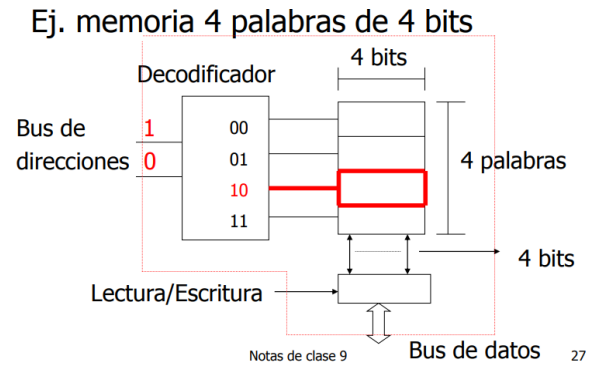


Guarda 1 bit de datos. Para memorias más grande s se deben organizar para direccionar palabras individuales:

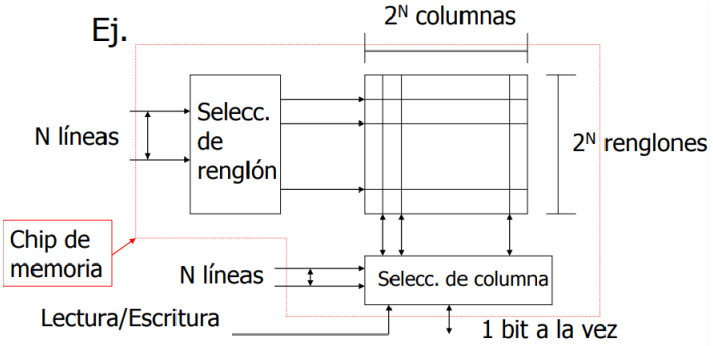


El anterior gráfico muestra un decodificador de 2 bits para seleccionar 4 palabras, después selecciona si L/E, y después se manda una señal de escritura a través de Dx, o de lectura a través de Qx. Cada chip contiene un arreglo de celdas de memoria.
Hay **dos tipos** de **organizaciones** de memorias semiconductoras: **2D, y 2½D**.
2D: Está **organizada** en **2^W palabras** de **B bits** cada una. **Cada línea** horizontal **de 2^W** palabras **se conecta** con una posición de memoria **seleccionando un renglón**.
Las **líneas verticales se conectan con cada bit a la salida**, y el **decodificador** del mismo **tendría 2^w salidas para w entradas**. Básicamente el ejemplo anterior. Ejemplo determinando las partes:

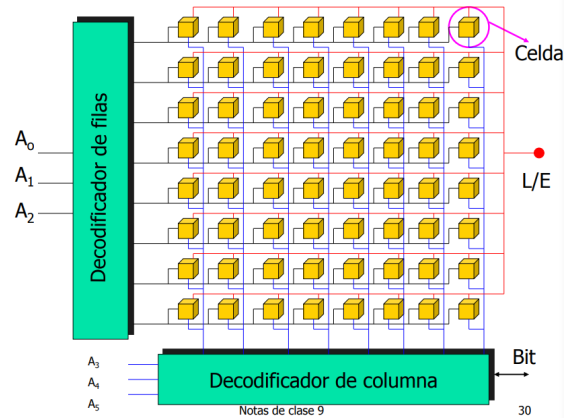


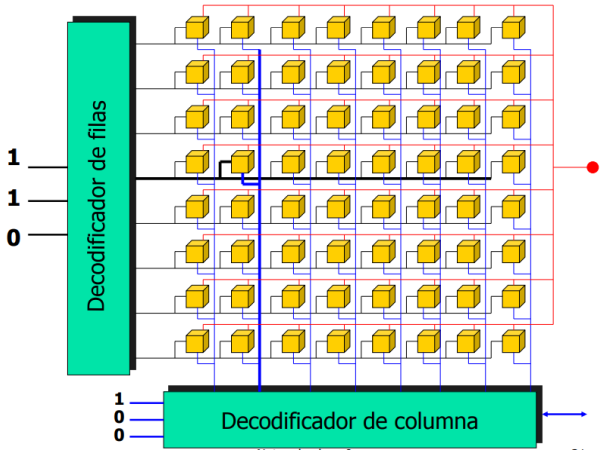


El **problema** principal de esto es que **se hace** un **rectángulo** muy **fino** debido a la disposición de las palabras, para esto **surge 2½D**:
Es **cuadrado**, los **bits** de una misma palabra están **dispersos** por chips.
La **dirección** tiene una **parte** para seleccionar **renglón y** otra **columna**, como direccionar una posición de una ciudad, y hay **2 decodificadores**.

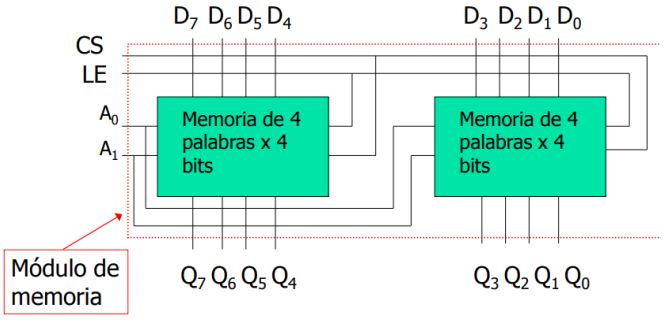


Como vemos, **entra** al **decodificador** de **renglón** la **cantidad** de **líneas**, y lo **mismo** con el de **columna**, por ejemplo entrarían 3 líneas para cada una si fuera 8x8, a la vez le llega la señal de lectura y escritura para saber qué hacer. Aparentemente el bit se manda o escribe 1 a la vez.

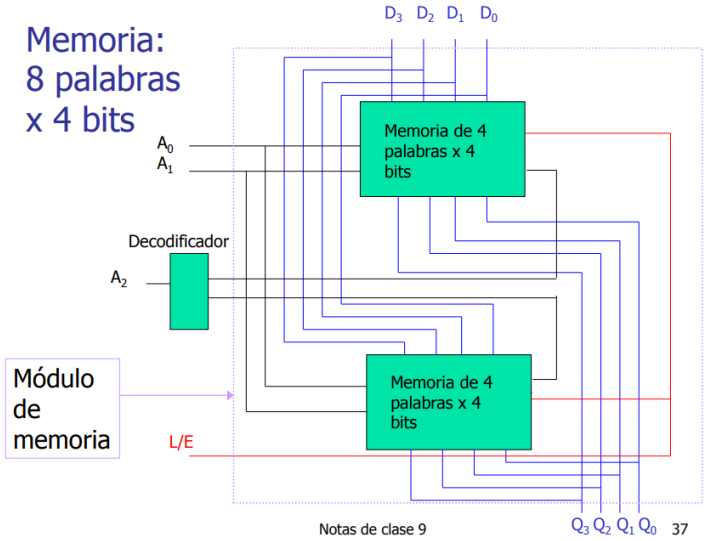




En **2D** los **bits** están en un **mismo chip**, en **2½D** la **misma palabra** estaría en **distintos chips**. **2D** ocupan **mucha superficie**, tendría **muchas palabras** y **pocos bits** en ellas, cada línea tiene que tener un manejador y conectarse al decodificador.
2½D tiene **menos** probabilidad de **error** porque los bits están dispersos, y **2D dificulta** el **uso** de circuitos **correctores**. También los **decodificadores** particulares de **2½D** son más **simples**, pero hay más.
Ahora, si queremos aumentar el tamaño de las palabras con 2½D ponemos chips en horizontal:

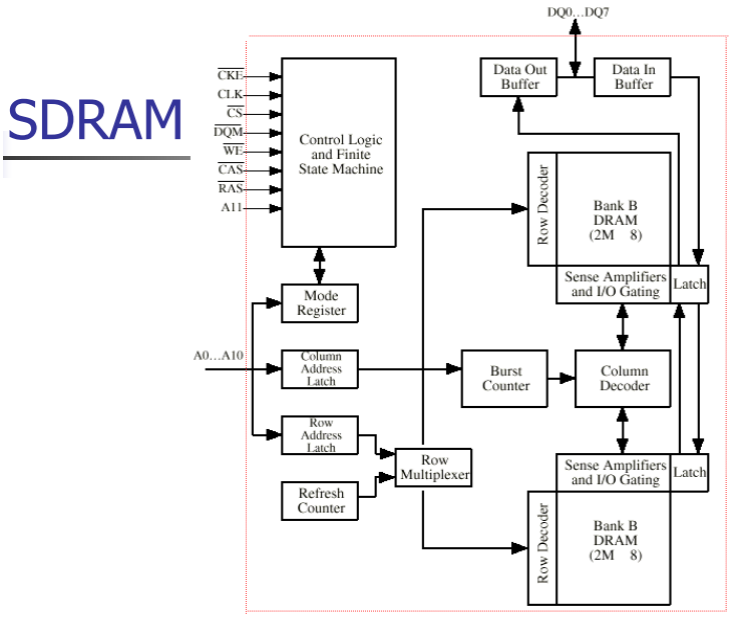


4 palabras de 8 bits.
Si queremos aumentar la capacidad en cambio sería así:



Se ve más complicado de lo que es por las líneas poronga pero imaginate que: **Selecciona con 2 bits que renglón direccionar**, luego **con otro decodificador selecciona** qué **memoria** específica elegiría.
Tecnologías ram:

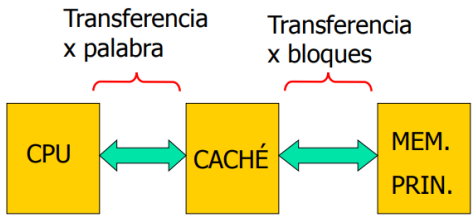
-DRAM básica es la **misma desde** los **primeros** chips.
-Enhanced DRAM, **contiene** un **SRAM pequeño**, **guarda** lo **último leído** como una caché.
-Cache DRAM, **contiene** una **SRAM grande**, **se usa** esa SRAM **como caché** o buffer serial.
-**Synchronous DRAM** (**SDRAM**), se **usa en DIMMs**, **sincronizada por clock** externo, **permite** mientras que la ram mueve los datos para que **CPU sepa cuándo** los va a **recibir** y **para** que **haga otras cosas**. Tiene modo Burst para trabajar en bloques.



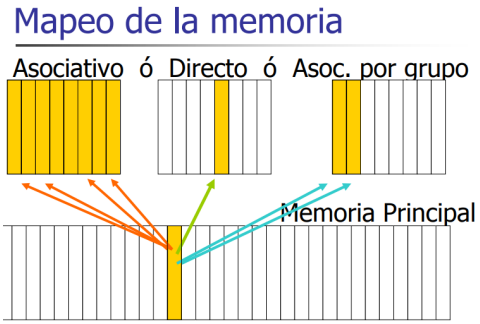
Clase 10:
Temas: Memoria **Cache**, **Memoria Externa**.
Históricamente a medida que **aumentaba** la cantidad de **circuitos** que pueden ir en un chip los diseñadores de **cpu** decidieron aumentar la **velocidad** y los de **memoria** la **capacidad**.
Esto se traduce en que una solicitud de **lectura tarda múltiples ciclos** de clock. En cada ciclo la CPU accede al menos una vez a la memoria a buscar la instrucción y o operandos. Por lo que la velocidad de la CPU se ve limitada por esto.
Para que la **memoria** sea igual de **rápida** que la **CPU** esta debe **estar dentro** de la misma, que es costoso. La solución es tener una memoria rápida de poca capacidad dentro, y afuera una de mucho más pero lenta.
El uso de la **caché** se basa en **2 principios**:
1-Localidad **espacial de referencia**: Es muy probable que el **próximo acceso** de memoria esté **cerca** del anterior.
2-Localidad t**emporal de referencia**: Es muy probable que se **vuelva** a **acceder** a un **mismo** espacio de memoria.
La localidad espacial sucede porque: El código se ejecuta secuencialmente (con excepciones), los programadores ponen las variables juntas, y existe la pila y las matrices.
La localidad temporal sucede porque: Hay bucles o ciclos, hay subrutinas, y hay pilas.
Ejemplo:

```
for i:= 1 to 10 do
        A[i]:=0;
```

Cada ciclo chequea el valor de i (espacial de referencia), y cada asignación almacena un 0 en el siguiente elemento del arreglo (temporal de referencia).
La **cache** entonces se **basa** en la idea de que cuando se referencia una palabra, se **copia** en esta **memoria** chiquitita, pero no solo la palabra, sino **su vecindario**. Así cuando la **CPU hace** otro **pedido**, la **caché intercepta** la **señal** y si tiene lo que pide se lo manda.

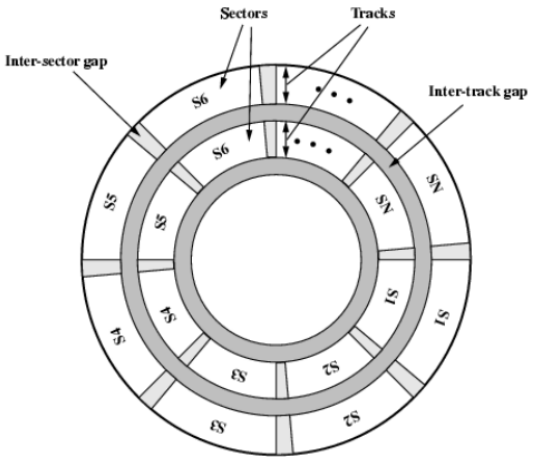


Formas de mapearla memoria:

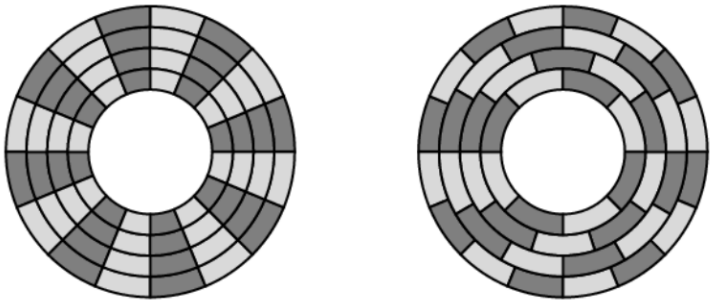


Directo: El bloque de memoria se divide en **etiqueta, índice, y palabra**. **Asigna posiciones** en base al **índice** asignado. (Cada **sector** de la **caché corresponde** a **varios** específicos de la **principal**, de ahí saca el índice, pero si les corresponde el mismo sector no pueden estar a la vez), **busca** el **índice** donde estaría espacio de memoria, y ahí **comprueba** la **etiqueta**.
Asociativo: Permite **cargar** el **bloque** de memoria en **cualquier línea** de la caché, el control sobre la misma se interpreta como una etiqueta y una palabra, la **etiqueta** lo **identifica** de manera única, **pero hay** que **chequear todas** las etiquetas simultáneamente para determinar si la caché lo tiene o no.
Asociativo por **grupo**: Este si tiene **etiqueta**, **índice**, y **palabra**. En el caso de ser **2 vías** (que tiene 2 memorias ram al mismo tiempo), **se busca en las 2** al mismo tiempo y ahí se **comparan** las **etiquetas** con la comparación simultánea.
Sobre los aciertos y fallos: Una **caché se mide** en la **frecuencia** de **aciertos**, es decir la cantidad de veces que si tiene lo que la cpu pide.
La **caché** tiene **2 niveles**: **L1 y L2**. Estos son **subdivisiones** dentro de la caché que también **varían** en **velocidad** y **almacenamiento**. Optimizan los aciertos, mejoran el porcentaje, en vez de bajar el porcentaje de fallos individuales.

Memoria externa:
Discos **magnéticos**, **ópticos** y **cintas magnéticas**.
Los discos **magnéticos** están **cubiertos** de **óxido de hierro** que es altamente **magnético**, (también se usa vidrio, que se dilata menos, es más uniforme y tiene menos defectos en su superficie). Estos funcionan **teniendo áreas** de discos **magnetizadas** en diferentes condiciones por un transductor, este **transductor lee/escribe**, y se mueve relativamente al disco. La cabeza transductora es la que hace la operación (bobina), el plato gira y la cabeza se mantiene estacionaria. Los 0 y 1 se guardan por medio de magnetización de las áreas.
Se organiza en: **Pistas** o tracks con **espaciadas** entre sí, todas las **pistas** tienen **misma cantidad** de **bits**, y tienen **velocidad angular constantes**. Las pistas se dividen en sectores, y un conjunto de estos es un bloque (cluster).



Un sector tipo suele tener: Encabezado (sincroniza la lectura e identifica al sector), datos, y código para errores (detecta e intenta corregir errores).
Ahora: Un disco puede tener cabeza fija o móvil, disco removible o fijo, simple o doble lado.Uno o múltiples platos, y la cabeza puede ser contacto (floppy, se raya más fácil), distancia fija o aerodinámica (Winchester).
El disco tiene una cabeza por cara. Todas se mueven solidariamente. Y las pistas alineadas forman cilindros, que son como aglomeraciones de datos relacionados (o que se intentan que estén) para que se reduzcan los movimiento de cabezas y aumente la velocidad de respuesta.
El disco está girando constantemente como ya dijimos (CAV rpm), mientras que los bits están más dispersos en la periferia que en el centro del mismo, por un tema de cómo están construidos, allí giran más rápido, para leer o escribir solo hay que mover el cabezal a la pista, y allí espera que pase por arriba, este tiempo determine la velocidad de transferencia.
Tiempos:
Seek (ir con el cabezal cilindro o pista correcta).
Latencia (Esperar que pase debajo del disco).
Acceso (Seek+Latencia)
Tiempo total: Acceso+Transferencia de datos.
La capacidad del disco es igual a: (bytes por sector)*(sectores por pista)*(pistas por superficie)*cantidad de superficies, es decir todo por todo:
Bytes*sectores*pistas*superficies.



Grabación en CAV **Grabación en zonas**

La grabación en zonas aumenta el espacio productivo.
El formato define las características y función de los distintos campos en cada pista.
Hardware es: Tamaño de sector fijo por marcas físicas.
Software: Tamaño de sector determinado por S.O.

Clase 11:
Temas clase: Almacenamiento óptico, monitores, impresoras.

RAID: Redundant array of independent/inexpensive disk, de nivel 0 a 6 son formas de organizar discos físicos como uno solo.
CD-ROM: Disco para audio hecho de policarbonato que almacena datos en pozos 'Pits' y se lee por láser. Capacidad de 650 mb. De velocidad lineal de 1,2m/seg. Velocidad angular de 200 a 530 rpm. Y velocidad de reproducción de 75 sectores/1 segundo. Es complicado acceder a él. Era fácil de producir en masa, caro en pequeñas cantidades, lento, solo lectura. También existieron CD de lectura y escritura. Y uno recordable que tiene 'Worm', aunque el profesor no sabía que era.
DVD: 2 tipos, en el que la V video y era solo para películas, y otro donde la V era versatile (se explica solo).
Tenía alta capacidad, de hasta 17 gb con doble lado y doble capa. Y bueno después viene el blu-ray que por el tipo de longitud de onda que usa es mejor.
Cinta magnética: ¿Lo tengo que explicar? nadie lo usa. Lenta y barata.
Modem (audio): Convierte señales ceros y unos en audio. Se introduce la tasa de bits/seg (bps) y la tasa de baudio (cambios de señal por segundo) con un baud rate máximo de 2400.
AM (amplitud modulada): Cambia la
amplitud de señal para representar las cosas.
FM (frecuencia modulada): Cambia la frecuencia de la señal.
Fase modulada: Cambia la forma.
Es posible enviar varios bits a la vez con diferentes frecuencias, ejemplo 4 señales diferentes 2400 veces por segundo, cada una representa una combinación de 2 bits. 2 bits por baudio. Tasa bps=baudaje x log2 (n).
Modems inteligentes: Establece bitrate, capaz de compresión, mayor bitrate (56k), 2400 baudios máximo.
Debido a que no hay reloj común el tiempo debe ser inferido por los datos.
Para evitar errores se usa un protocolo donde se hace que la señal tenga misma cantidad de ceros y unos, si está mal el ratio se pudo haber mandado mal.
Entrada de datos: Teclado, mouse.
Salida de datos: Monitores, impresoras.
Un monitor presenta algo como 60 frames por segundo, y tiene 500 píxeles verticalmente, y en cada línea hay 700 puntos, entonces sería: 60x500x700= 21M de puntos/segundo.
El monitor puede ser terminal, orientado a presentar caracteres ascii, o mapeado en memoria orientado a pixel.
Las impresoras, bueno depende de su tecnología, algunas escriben por una matriz de puntos, y usan un ROM de caracteres para leer y escribir a donde van. Otras como la láser tienen mayor dpi. Y la inkjet lanza chorros de teína a los puntos correctos.