

# preguntas teóricas posibles-final

## CONSULTAS RUNCO

### ▼ LA NORMALIZACIÓN EN PFloatante. Cómo se normaliza, para qué sirve. Qué normalizaciones conoce?

Para escribir nros reales con coma.

**Tenemos 1 problema:** En una mantisa fraccionaria se puede tener múltiples representaciones de un mismo n°. Si hablamos de normalización ya sabemos que hablamos de mantisa fraccionaria, es decir, de pf.

la **normalización** sirve para obtener un único par de exponente y mantisa para cada representación de numero.

Una de las primeras normalizaciones que vimos es la de **0,1** donde todas las mantisas comenzaban así → hacer que el bit más significativo (luego de la coma) sea 1.

**#Dato** → Si la mantisa es BCS el primer bit va a ser para el signo y la normalización se verá con el segundo bit más significativo

**:( Desventaja:** Al arrancar todas las mantisas con 0,1 el sistema no puede representar el n° 1

\*bit implícito

¿Cómo se escribe un Nro en punto flotante normalizado?

1. Se escribe el Nro en el sistema propuesto para la mantisa.
2. Se desplaza la coma y se cambia el exponente hasta obtener la forma normalizada.
3. Se convierte el exponente al sistema propuesto para él

Luego vimos de la de IEEE 754, donde la normalización de la mantisa era **1,** **Ese primer 1 no se almacena- bit implícit vol II.** Pero la idea era la misma (especificaba una forma de operar con números en pfloatante)

Mantisa : en representación Signo Magnitud o BCS. Exponente en Exceso raro.  
(127/1023 bss →  $2^{n-1}$  -1) 011111

→ Tenemos dos tipos de IEEE754

## DOBLE PRECISIÓN Y SIMPLE PRECISIÓN

Doble precisión: Cadena de 64 bits → 1 bit de signo + 11 de exponente + 52 de mantisa.

Rango del exponente [2^-126 a 2 ^128]

Simple Precisión: Cadena de 32 bits → 1 bit de signo ,8 para el exponente, 23 para la mantisa.

Rango del exponente [ 2^-1022 a 2^1024]

### ▼ qué me garantiza una mantisa normalizada?

la representación única de un numero en pf

### ▼ Bit implícito.

Como todos los números de la normalización comienzan igual, ya sea la ieee754 o la otra, para qué almacenarlo? entonces no lo almaceno, Pero como igual en la memoria ese espacio estaba disponible , me permite tener un bit más al final disponible para representar un número aún mas preciso.

### ▼ Para qué sistemas de los que vimos sirve la normalización que nosotros le explicamos ?

Sirve para **BSS y BCS** (ya que el signo está en otro bit y no pasa nada con el 0,1) porque si yo digo que todos los números empiezan con 0,1 y ustedes saben que los positivos empiezan con 0 y los negativos con 1, ahí yo no podría escribir números negativos. Entonces necesito un sistema o que no tenga signo, o que el signo esté aparte del valor del modulo del numero. **Una mantisa en ca2 se puede normalizar? no, no nos serviría.**

### ▼ Formato ieee754. situaciones especiales, xq es así. Como funciona el rango, la resolución.. Por que el rango es así... ( trampa por la normalizacion)

Simple y doble precisión

El exponente en exceso= 011111 (127) en simple precisión

La mantisa en BCS. normalizada en 1,

Los casos especiales para representar números fuera de lo que es el formato normalizado.

### ▼ Casos especiales del IEEE754

Hay casos especiales que representan números que no tengo que hacer la cuenta , como si fueran normalizados.



### Casos especiales:

- $E = 255/2047, M \neq 0 \Rightarrow \text{NaN -Not a Number-}$
- $E = 255/2047, M = 0 \Rightarrow \text{Infinito}$
- $E = 0, M = 0 \Rightarrow \text{Cero}$
- $E = 0, M \neq 0 \Rightarrow \text{Denormalizado}$ 
  - $\pm 0, \text{mantisa\_s-p } 2^{-126}$
  - $\pm 0, \text{mantisa\_d-p } 2^{-1022}$

Notas de clase 3

43

- donde el exponente tiene todo 1(LOS DOS PRIMEROS DE LA DIAPO)
- o el exponente tiene todo cero.(LOS DOS ULTIMOS DE LA DIAPO)

En esos casos tengo que venir a ver esta tablita de la diapo

### ▼ Cómo se escribe un numero en Ca2

Vamos en camino a solucionar el problema de tener dos 0 y tratar de solucionar el primer dígito donde indica solo el signo, queremos que tmb forme parte del número. Ca2 soluciona ambas. **Los N bits representan al numero.**

Primero en sí tener un numero en complemento es

- *Definición: "Se define el complemento de un número A a un número N (con A menor que N) como la cantidad que le falta a A para llegar a N".*

Los positivos se escriben igual que en BSS.

Los negativos → Tiene el Ca2 del valor deseado. (tener sumado un exceso)

cheat es sacar el Ca1 y sumarle 1. Copio desde la der t...

Ya no tengo el 0 negativo , recuperé ese valor y ahora tengo el -128 (rango asimetrico ) .[-128 → 127]

▼ Como se escribe un numero en Ca1

▼ Como se escribe un numero en BCS

▼ Como se representa un numero en BCD.

▼ el exceso

tengo un nro- le sumo una cantidad fija(el exceso)

▼ Como se calcula la resolución en coma flotante

▼ Sistema de numeración

▼ para qué esta el formato en punto flotante?

**para representar numeros en un rango muy grande o muy chico. se usan menos bits para almacenarlo y se agilizan las cuentas.** si bien en los numeros grandes la resolucion es pesima, quizas es porque no necesito tanta precision (ejemplo de distancias del radio de la tierra, o distancias entre planetas, no me sirve hablar de cm) la mayoria de las veces que lo uso es porque necesito mayor precision de la que me ofrece punto fijo.

▼ Definición de complemento a la base general.

gral para cualquier base. Como se obtiene el complemento de un nro? **el complemento de n nro = la base elevada a la cantidad de bits - el numero que quiero complementar.** Cuando la base es 2 se llama complemento a 2

▼ para qué sistema sirve la suma de la ALU?

**bss y ca2.** LOS UNICOS DOS SISTEMAS QUE SIGUEN LAS REGLAS COMO EL BINARIO CRUDO SON CA2 Y EL BSS. por eso tenemos el carry para el BSS Y EL OVERFLOW PARA CA2. por eso hacemos una unica cuenta y hacemos dos interpretaciones del resultado pero la cuenta es unica, es la misma. La alu suma con binario crudo

▼ Completando la anterior..**Para qué servían los flags?**

El bit de carry indicaba una condición fuera de rango en BSS . El bit de overflow indica una condición fuera de rango en números CA2. No pude representar el numero con la cantidad de bits que tengo .

▼ **LEYES DE DEMORGAN. APLICACIONES.**

(son Identidades del algebra booleana)

La de De Morgan es importante porque **une producto con suma**. me da la formula de conversion entre NAND Y NOR.

<b>De Morgan</b>	$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$
------------------	--	--

→

Yo con NAND y NOR puedo hacer todas las demás funciones. son completas.

▼ **Instrucción de suma ADD. habia add con signo o sin signo?**

no, porque la alu no suma con signo,suma binario crudo.

▼ **Error absoluto y relativo**

$$EA(X) = |x_{\text{realidad}} - X_{\text{real}}|$$

$$ER(x) = EA(X) / X$$

▼ **Cual es el circuito combinatorio mas importante de la cpu?**

La ALU.

▼ **qué tipo de circuito es la ALU?**

es un **combinacional, porque es un sumador**. otros ejemplos de combinacionales son el el multiplexor, el decodificador .

▼ **qué circuito combinacional existe ya armado en la cpu o que se pueda armar**

La alu por ejemplo es un circuito combinacional.

\*el sumador )

hay multiplexores,  
decodificadores para direccionar memoria .

▼ qué maneras hay de representar un circuito combinatorio hay?

tablas de verdad, forma grafica - cada compuerta o cada funcion tiene un simbolo distinto, ecuación .

▼ EL método de suma de producto.

Como pasar una vez que yo tengo la tabla de verdad a la ecuación o al circuito lógico?. una vez armada la tabla de verdad miro los unos de las salidas - **los 1 porque hay una combinación única de las entradas que genera ese 1,** haciendo cumplir los 1 me aseguro que el resto va a valer 0 - (**la tabla de verdad esta definida con todas las combinaciones posibles de entradas y una columna de salida que responde a la función lógica que corresponda**) es la que voy a escribir como termino de la ecuación . Si vale 0 la pongo negada, si b vale 1 la pongo sin negar y asi. Dependiendo si las entradas valen 0 y 1. están todas unidas con un and porque necesito que esas entradas sean SIMULTANEAS. que se den todos esos valores a la vez. Cada termino lo uno con un ORG, con un + (es suma es una función logica, es un +) porque es uno u el otro u el otro.

## Función M

$$M = \bar{A}\bar{B}C + A\bar{B}C + AB\bar{C} + ABC$$

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

## ▼ Qué diferencia hay entre un circuito secuencial y uno combinacional ?

Ambos responden a los valores lógicos en las entradas PERO

La diferencia está en las salidas. En uno secuencial las salidas no solo dependen de las entradas en ese instante, dependen tambien de la historia pasada, de las entradas anteriores. La salida vuelve a conectarse a la entrada. Los primeros que definimos de estos son los **flip flops**. Vimos dos aplicaciones con flipflop: en memoria, registros , registros de desplazamiento y contadores.

el secuencial va a tener la característica que una de las salidas propias de la compuerta vuelve para ser entrada.

En el combinacional, por otro lado, la salida está determinada exclusivamente por los valores de las entradas en ese instante.

- Si cambia la entrada, cambia la salida.
- Los valores pasados de las entradas no influyen en los valores de las salidas.

## ▼ Con qué flip flops se crean los contadores?

se hacen con flip flop JK trabajando como T. O flip flop T. y los registros y las memorias se construyen con flipflop D.

## ▼ combinacionales: diferencia entre el semisumador y el sumador completo.

con el semisumador puedo sumar 2 digitos binarios. el completo lo mismo con la posibilidad de sumar el acarreo.

## ▼ como puedo armar un elemento de hardware que sea capaz de almacenar un bit? y que tenga la capacidad de recordar.

con flipflops SR (secuenciales)- D - El JK trabajando como D.

## ▼ qué tipo de flipflops conoces ? aplicaciones de los flip flops.

El JK puede emular a cualquier de ellos por su tabla de verdad. con las dos entradas en 1 se utiliza como contador, trabaja como toggle.

los registros de desplazamiento y las memorias se pueden hacer con el **flip flop D**. El JK tambien puede trabajar como D, hay que conectarlo de una manera especial pero trabaja como D. La entrada K es la J invertida, asi trabaja como D.

▼ Cómo hago que un flipflop JK que tiene dos entradas trabaje como D que tiene una sola?

hago una única entrada J y la K es la J negada.

▼ MULTIPLEXOR →

- El multiplexor conecta **muchas entradas** → **una sola salida**
- Esto depende de los canales de selección (qué entrada selecciona)

Tenemos

- entradas de datos Di
- canales de selección Si
- salida F

Cada entrada de datos puede ser direccionada a la salida dependiendo del valor que se ingresa en la entrada de selección → **Una sola entrada es redireccionada a la salida.**

Un ejemplo es la carga del **contador de programa (PC)** El valor a cargar puede venir de:

- De un contador binario, si el PC se va a incrementar para la siguiente instrucción.
- Del IR, si se acaba de ejecutar una instrucción de salto usando direccionamiento directo.
- De la salida de la ALU, si la instrucción de salto especifica la dirección usando modo de desplazamiento.

▼ DECODIFICADOR

El **DECODIFICADOR** por ejemplo es un dispositivo con 3 entradas y 8 salidas- con 3 bits

Dependiendo la cantidad de bits del bus de direcciones , vas a direccionar a memoria . x ej, si tenes 3 bits, tenes posibilidad de 8 cajitas de memoria( $2^3$ ), entonces en el deco entran los 3 bits y selecciona 1 de las 8 lineas, esa linea habilitaría la cajita de memoria(dependiendo del aprotón de lineas de entrada).

Un decodificador tiene N entradas y 2N salidas.

Los decodificadores tienen muchos usos en computadores digitales. Un ejemplo es la decodificación de direcciones.

#### ▼ CODIFICADOR

Un codificador es un circuito combinacional con  $2^N$  entradas y N salidas, cuya misión es presentar en la salida el código binario correspondiente a la entrada activada.

#### ▼ SUMADOR

#### ▼ Ejemplos de aplicación ROM Combinacional

A los circuitos combinacionales se les llama a veces circuitos «sin memoria», ya que su salida depende sólo de la entrada actual y no retiene la historia de las entradas anteriores.

Sin embargo, hay un tipo de memoria que se implementa con circuitos combinacionales, llamada memoria de solo lectura (**ROM**).

#### ▼ Porqué a los flipflops tambien se los llama biestables?

**Se llama bi-estable porque el circuito posee sólo 2 estados posibles de funcionamiento, se queda en cada uno de ellos, salvo que las entradas provoquen un cambio.**

#### ▼ Qué problema Soluciona el flipflop JK?

Qué Problema arregla el jk?

Repara la situación de indefinido del RS → Lo hace dando como resultado el complemento del estado anterior

#### ▼ Secuenciales → Asincronicos y Sincrónicos.

Los circuitos secuenciales pueden ser

- **Asincrónicos:** las salidas cambian cuando cambian las entradas (si corresponde). Los que veníamos viendo.
- **Sincrónicos:** las salidas cambian cuando cambian las entradas Y **una señal (de sincronización) lo habilita.**
  - La señal de sincronización es una referencia temporal llamada “reloj”.
  - Permite definir el instante en el que ocurren los sucesos.
  - En especial si los sucesos deben ocurrir simultáneamente.

Reloj (Clock o CK) → Señal que cambia periódicamente entre 1 y 0 a intervalos regulares de tiempo.

Es una entrada de control: El flip flop va a tomar en cuenta esta entrada para determinar en qué instante va a mirar sus entradas para procesarlas y largar la salida

Lo más común es que respondan a flancos crecientes pero también pueden responder a decrecientes.

#### ▼ Registros y Contadores.( Secuenciales)

##### **Registro**

Los flip flops almacenan un bit —> Los registros se crean a partir de los flip flop D

Ante una determinada señal del clock, este memoriza y genera una salida

Contador

#### ▼ VON NEUMAN

- Von neuman fue el que introdujo la memoria como tal, la idea de escribir un programa y de programa almacenado. Introdujo el **sistema binario**, el direccionamiento de la memoria, el concepto de tener una unidad de control, una alu que haga cuentas... 5 UNIDADES. **DATOS Y PROGRAMAS RESIDEN EN MEMORIA.**

Von neuman propuso la **maquina de 3 direcciones**. LAS MAQUINAS DE HOY EN DIA SON DE DOS DIRECCIONES.

#### ▼ Cómo está organizado el sistema de cómputos?

cpu..memoria,etc

#### ▼ **porqué podemos decir que el procesador es un módulo que sabe interpretar funciones básicas?**



## Tipos de instrucciones

- En lenguajes de alto nivel escribimos:

$$X := X + Y$$

- Esta instrucción suma los valores almacenados en las posiciones de memoria X e Y.
- Esto puede implicar cargar registros, sumarlos y luego almacenar el resultado en memoria.
  
- Una instrucción de alto nivel puede requerir varias instrucciones de máquina.
- El lenguaje de alto nivel expresa operaciones en forma “concisa” usando variables.
- El lenguaje de máquina expresa las operaciones en forma “básica” involucrando movimiento de datos y uso de registros.

qué tiene que pasar con un programa en lenguaje de alto nivel para ser ejecutado?

- Cualquier programa escrito en lenguaje de alto nivel se debe convertir a un lenguaje de máquina para ser ejecutado.
- El conjunto de instrucciones de máquina debe ser capaz de expresar cualquiera de las instrucciones de un lenguaje de alto nivel.

▼ Qué registros necesito, qué lugares de almacenamiento temporario necesito adentro de la cpu?

▼ que utiliza la cpu para comunicarse con los dispositivos o con al memoria?

Buses.

▼ EL BUS DE DATOS ES **BIDIRECCIONAL**. QUIEN DETERMINA EN QUE SENTIDO VA A IR?

▼ Como seria la mecánica a grandes rasgos de ejecutar un programa?

▼ Ciclo de instrucción

precuela, a grandes rasgos →

Vengo por el bus de datos con una instrucción, llega al MBR, queda almacenado ahí, así puedo **liberar el bus de datos, la instrucción queda en el MBR**. ahora el bus de datos se puede ir a hacer otra cosa, CC tenia que esperar hasta que esté almacenado en el IR para poder desocuparse.

Por el camino negro llega al IR ahí el bloque verde toma la instrucción, la interpreta y si sabe que tiene buscar un dato en memoria, por el camino negro escribe en el MAR una dirección , y sale para el bus de direcciones. Al rato, por el bus de datos llegó el dato, se guarda en MBR, pero como es un dato no va a IR, va a unos de los registros D0, D1...ETC.

▼ qué se guarda en el registro IR?

la instrucción encontrada en memoria gracias a la dirección que tenía el program counter. A este registro **solo llegan instrucciones, nunca datos**. Si llegara a haber un dato se cuelga la compu.

▼ quién hace precisamente dentro de la cpu la interpretación de los 1 y 0 de la instrucción?

La unidad de control.

▼ el registro contador de la ALU

un registro formado por flipflops que cuentan la cantidad de pulsos de reloj que entra en el primer flipflop, se hace con flipflops jk trabajando como t el jk con las dos entradas en 1.- (en lógica secuencial. ) nosotros vimos de lógica secuencial nuevos bloques básicos que son los flipflops con los que armamos cosas como registro contador, registro común, registro desplazamiento.

▼ qué cambia en la alu cuando usaban instrucciones de un operando

va a sumar dos números uno que va a venir en la instrucción y otro dato que va a estar en el registro acumulador. previamente tenes que tener cargado el acumulador con el nro que querés sumar. **FORMATO DE INSTRUCCIÓN.**

▼ **Cómo está compuesta una instrucción, qué información lleva? qué información es necesaria en una instrucción? El ciclo de instrucción.**

Todas las instrucciones son **AUTOCONTENIDAS**. En la instrucción misma tiene que estar todo lo que hay que hacer, toda la información. Si es una suma dice que hay que sumar, con quien, donde están esos datos,dónde guardar el resultado. La Cpu a través de la UC se va enterando de lo que tiene que hacer cada vez que analiza instrucción por instrucción. Cada instrucción tiene que viajar desde la memoria a la CPU , ahí la analiza la UC y toma las acciones necesarias cuando sabe qué hay que hacer , para ir a buscar los datos, qué operaciones hay que hacer si es de alu, solo mover datos, si es operación lógica u aritmética,etc. Ese conjunto de pasos para llevar a cabo la ejecución de una instrucción es lo que conocemos como **conjunto de instrucción**. algunos **pasos son comunes a todos ellos**, todas las instrucciones hay que buscarlas a memoria porque allí residen por ejemplo. En cuanto a la **información que debe llevar: tiene que tener un conjunto de 1 y 0 que represente al código de operación, otro conjunto de 1 y 0 que represente la dirección de los operandos, otra del resultado y otro de la dirección de la próxima instrucción**. Eso es lo que conocemos como la **maquina de 4 direcciones**. → **formato de la instrucción**. Así son muy largas, conviene que sean más cortas → cuantos mas bits tenga la instrucción mas tardará en procesarse. → evolucion a la maquina de tres direcciones → maquina de dos direcciones → máquinas de una dirección : donde lo único que está especificado es una sola dirección - mas allá de la especificación de operación.

- El contador de programa siempre tiene la dirección de la próxima instrucción
- cuando la UC termina decodificar la instrucción sabe cuánto ocupa en memoria y entonces a partir de eso incrementa el contador de programa.

▼ **Cómo sabe la CPU a dónde ir a buscar la primera instrucción del programa?**

Hay un registro dentro de la CPU que se llama PC (program counter, contador de programa) que es el que lleva la cuenta de LA DIRECCIÓN de la próxima

instrucción que hay que ejecutar. Lo que tiene el pc es LA DIRECCIÓN de la instrucción, no la instrucción en sí. Después incrementa el valor del PC y sigue en secuencia hasta una instrucción que le diga que terminó el programa. Una vez que el PC . Cuando tiene la dirección para ir a buscarla a memoria la trae por el BUS DE DATOS.

- cuando hablamos del formato no hay que olvidar mencionar que todas las instrucciones tienen código de operación porque es lo que nos dice qué hay que hacer pero eso no es una dirección.

- ▼ cual es la información que tiene que llevar una instrucción? después vemos como podemos ahorrarnos bits, primero la información completa.

codigo de operacion, direccion de los dos operandos, direccion del resultado y direccion de la proxima instruccion. son 5 campos, 4 direcciones. el formato de la instruccion va a depender despues del tipo de máquina que estemos utilizando.

- ▼ **por qué el stack se dice que tiene modo de direccionamiento implicito/indirecto?**

porque las únicas dos instrucciones PUSH Y POP vos no indicas con que registro usas, ya sabes que es el SP. ese registro.

- ▼ Formato → las maquinas de varias direcciones.
- ▼ El conjunto de instrucciones que tenemos particular
- ▼ **Modos de direccionamiento**

Las distintas maneras de explicar en la instrucción dónde se encuentran los operandos,dónde voy a guardar los datos, dónde los muevo. La fuente y el destino de alguna operación (los nombres que le dan INTEL).

- ▼ **Por qué hay distintos? para qué tengo tantos modos de direccionamiento? Es decir, cual es su objetivo? SON 3.**

**uno de los objetivos principales es disminuir la cantidad de bits en una instrucción.** Podríamos decir que una instrucción por ejemplo ADD AX BX ahi estamos diciendo que los datos están en los registros AX Y BX, pero no decimos cuales son esos datos, sólo donde están . Como los registros son pocos, con 4 bits me alcanza para especificar los 16 registros y ahi disminuyo la cantidad de bits en una instrucción, si yo pongo una dirección eso ocupa mas bits.

**-Manejo mas eficiente de los datos (a los arreglos, a las tablas, para recorrerlo) —> por eso existe el direccionamiento indirecto.** a parte de los corchetes de poner el registro BX podemos poner un registro indice, por eso en pascal podemos poner el nombre del arreglo y entre [] un índice. El nombre del arreglo está asociado con la dirección del primer elemento que va cargando en BX y despues el indice iria en otro registro, entonces nosotros podriamos poner BX + SI, x ej . que tengamos ese direccionamiento indirecto nos permite recorrer tablas o arreglos. LOS ARREGLOS EN CADP.

**Otra razón importante es que yo muchas veces no conozco las direcciones del programa hasta que se carga, entonces tengo que tener alguna manera de decir desde dónde se cargó el programa, o que a partir de tal valor guardo las variables y eso.** RAZÓN IMPORTANTE POR LA CUAL TENGO TANTOS MODOS DE DIRECCIONAMIENTO.

▼ Dar algún ejemplo de instrucción.

▼ **si el código de operación tiene solamente 2 bits cuantas instrucciones distintas va a poder tener la maquina?**

igual que los bits,  $2^2 \rightarrow 4$  instrucciones. Por eso es importantes saber cuantas instrucciones voy a tener, para saber cuantos bits voy a necesitar, porque tengo que nombrar cada operación con un conjunto de 1 y 0 distinto.

▼ Cómo sabe la memoria que la cpu está buscando un dato?

la unidad de control por el bus de direcciones pone la dirección a donde se quiere conectar- a la cajita de memoria. todas las memorias reciben esa dirección pero electrónica de por medio solo una cajita se conecta, la que tiene esa dirección. A través del bus de control yo envío señales para saber si voy a leer o escribir y la electrónica se acomoda para que el dato vaya o venga a la CPU correspondientemente. a través del bus de datos va viajar cualquier cosa ya sea instrucciones o datos.

▼ Como se diferencia entre un dato y una instrucción?

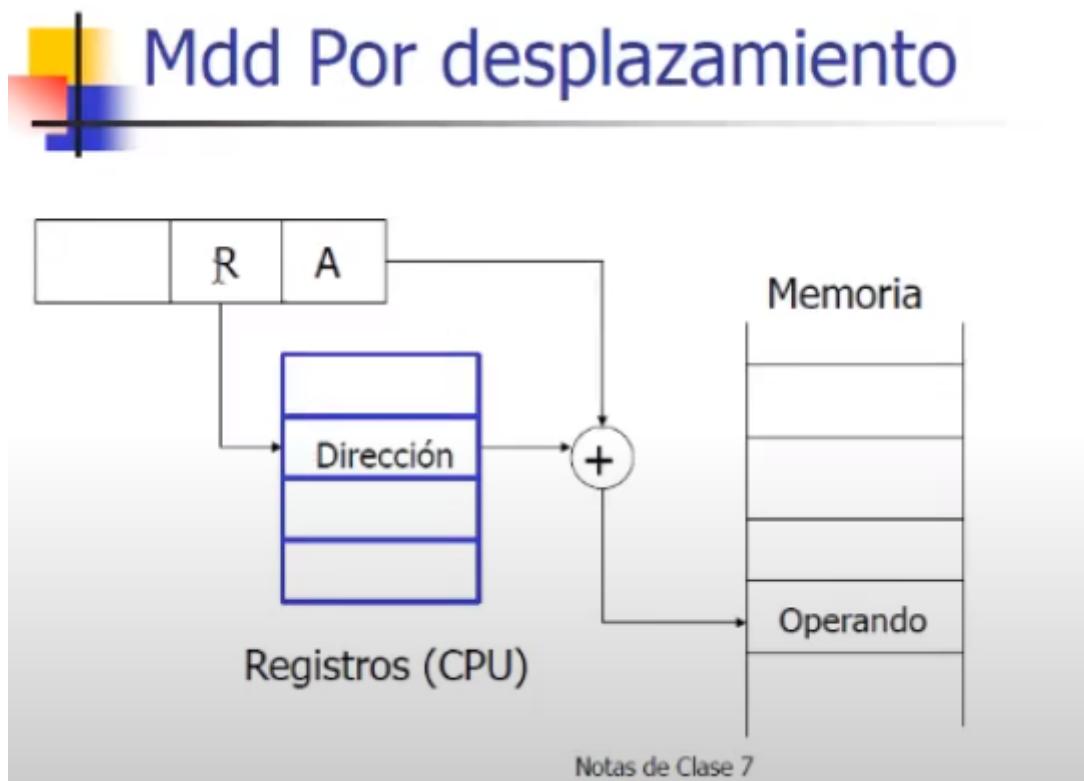
PARA ELSIMULADOR QUE VEMOS EN ESTA MATERIA enceuntra la primera instrucción en 2000h , a partir de ahí mientras va decodificando sabe lo que ocupa una instrucción y donde está la próxima, y lo que es dato y lo que es instrucción, y lo que es dato dependiendo del modo de direccionamiento y lo que es instrucción. pero ambos son 1 y 0 que viajan por el bus de datos.

▼ Explicar detalladamente la lectura o la escritura en memoria.

- cuando nosotros obtenemos el dato de memoria (estoy leyendo un dato) primero pasa por el bus de datos nuestra información , viene por el bus de datos y ahí se carga en el MBR, es la ruta por donde viene. y luego se va a al IR, donde se hace la decodificación.
- La maquina busca una instrucción, como sabe que es una instrucción → va al IR, ah le falta otra parte del código de operación, lo trae, va al IR, ya sabe que si ocupa 4 bytes y esos anteriores eran 2 , sabe que los dos que siguen son datos entonces ya no va al IR, porque sabe con qué formato trabaja.

▼ Modo de direccionamiento con desplazamiento → requiere dos campos de dirección. al menos uno tiene que ser explícito.

hay distintos tipos de desplazamiento. LO IMPORTANTE QUE TIENEN QUE SABER es que vos podés tener un registro + un número y en función de eso hay distintas opciones: pdf 42 de la clase. hay variantes pero tiene dos componentes la dirección por lo menos.



vemos que tiene dos componentes el registro (R) que tiene una dirección, + A



## Por desplazamiento (2)

- Combina capacidades de indirecto y directo. Requiere que la instrucción tenga dos campos de dirección. Estos dos campos se suman para producir la dirección efectiva. Los más comunes:
  - Relativo
  - De registro base
  - Indexado



## Relativo

- El registro referenciado de manera implícita es el contador de programa PC.
- La dirección de la instrucción actual se suma al campo de dirección para producir la dirección efectiva.
  - El campo de dirección se trata como un número en Ca2.



## De registro base

- El registro referenciado contiene una dirección de memoria y el campo de dirección tiene un desplazamiento.

Este tendría la dirección de memoria en un registro y un numero despues, lo que es desplazamiento es como si fuer aun numero,, esta en un registro base como BX. Bx + un numero.



## Indexado

- Se direcciona la memoria con un registro más un desplazamiento.
  - Es "igual" al anterior pero se intercambian los papeles del registro y del desplazamiento.
  - La Indexación proporciona un mecanismo eficiente para realizar operaciones iterativas.
  - Se utiliza un registro llamado **índice**
    - algunas máquinas incrementan ó decrementan este registro como parte de la instrucción (Autoindexación)

Notas de Clase 7

45

- ▼ en forma implícita relativo al contador de programa quien tiene esos direccionamientos?

los saltos , porque vos en los saltos podes especificar qué numero hay que saltar respecto a donde está el PC para llegar a la dirección correcta. Vos en los saltos podes poner dos cosas: la dirección a la que hay que saltar o un numero que hay que sumar al PC para llegar a ese dirección. es lo mismo, solo que uno es "implícito". Las instrucciones push y pop tienen de alguna manera direccionamiento implícito, porque usan el SP sin decirlo, porque sabemos que es donde está apuntando el SP que se hace el push y pop, vos no tenes explicitamente escrito eso.

▼ Qué elementos contienen los 1 y 0 de una instrucción?

es como preguntar sobre el formato de una instrucción. = qué información tiene que tener una instrucción. → **la instrucción es autocontenido, tiene que estar toda la información de lo que hay que hacer. si voy a hacer una suma tiene que decir donde están los operandos, donde guardo el resultado, que es una suma(la tarea a realizar y con qué modo de direccionamiento- el código de operación), donde esta la dirección de la próxima instrucción.**

▼ como se da cuenta la CPU si lo que se busca en memoria es un dato o una instrucción?

una vez que encontraste la primera instrucción , sabe si ocupa 2, 4 6 u 8 bytes. EL contador del programa se incrementa acorde a la instrucción que acaba de decodificar, y ahí sigue encontrando la instrucción. Una a continuación. de la otra pero no quiere decir que todas tengan que tener el mismo tamaño. Si cuando va a buscar la instrucción no hay un código de operación valido, no sabe que hacer, se pierde → > es crucial encontrar la primera. A partir de ahí, decodificando instrucciones va a encontrando las demás.

▼ modo de direccionamiento por registro - entre registro y registro

# MOV registro,REGISTRO

## Modo de direccionamiento por registro

1000101W 11rrrRRR

Registros de 16 bits

W  
1= 16 bits  
0= 8 bits

10001011 11rrrRRR

Registros de 8 bits

10001010 11rrrRRR



Según como tenga el ultimo bit ese que llamo W, dice si es un 1 es de 16 bits el movimiento y si es un 0 es de 8 bits. Despues tiene 11 fijo, y luego 6 bits para representar los dos registros. los rrr con minúscula son los 3 primeros bits, y los otros 3 R con mayúsculas el otro.

### ▼ Qué son los modos de direccionamiento? Cuales son?

son las distintas manera de especificar en una instrucción donde están los operandos , o donde guardar el resultado. etc. Porque no poner la dirección física y listo?

1. para acortar la cantidad de bits. el direccionamiento por registro ocupa menos bits, es mucho mas corto que hacer un acceso a memoria.
2. manejo mas eficiente de datos—> si no tengo direccionamiento indirecto no puedo recorrer una tabla, un arreglo,
3. no conozco las direcciones físicas hasta el momento de ejecutar el programa. necesito un modo de direccionamiento que me diga a partir de donde guardé el programa.

- ▼ CONSIDERACIONES QUE TIENE EN CUENTA EL DISEÑADOR DEL CONJUNTO DE INSTRUCCIONES.
- ▼ qué característica especial tiene la ALU en caso de maquinas de 1 dirección?
- ▼ Registro contador de la ALU. La ALU en sí

▼ Jerarquía de Memoria

GRACIAS A LOS PRINCIPIOS O PROPIEDADES QUE CUMPLEN LOS PROGRAMAS YO PUEDO ARMAR LA JERARQUIA DE MEMORIA . PRINCIPIOS DE LOCALIDAD Y..

- ▼ Porqué funciona la jerarquía de memoria? por qué me sirve?
 

hay mas frecuencia de accesos a medida que subo en la pirámide, etc.  
**PRINCIPIOS DE LOCALIDAD. TEMPORAL Y ESPACIO DE REFERENCIA**  
 .Esos principios los cumplen LOS PROGRAMAS (los principios los cumplen los programas y por eso pueden armar la jerarquía), no la memoria.

Dice que si accedí a una zona de memoria es probable que vuelva a acceder a esa zona o cercana, el temporal dice que si accedí a una posición de memoria dentro de un corto tiempo voy a acceder nuevamente, entonces lo que se hace es elaborar esa pirámide,  
 toda la manera en la que nos enseñaron a programar cumple con estos principios.

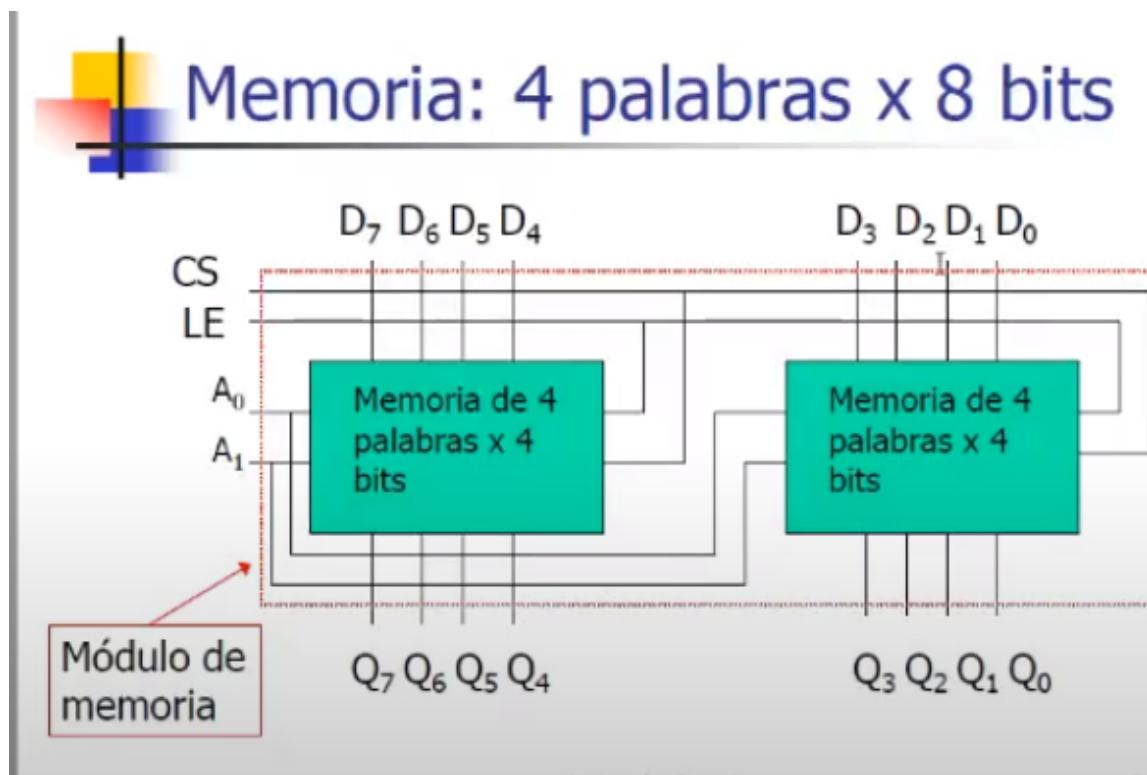
Tipos de memoria, ORGANIZACIONES 2D 2YMEDIO D.

HAY QUE EXPLICAR LAS DOS CARILLAS DEL LIBRO DE STALLING.

**Memoria estática/memoria dinámica.** En una de ellas, la 2d, todo esta organizado en un chip . y en el 2 y medio d la palabra de memoria esta repartida entre varios chips. El libro da un ejemplo de que sale un bit por chip pero no es necesariamente siempre así, podría tener 1 byte provisto por cada chip. La ventaja y desventajas de uno y otro: Si hay W lineas de direcciones voy a tener  $2^W$  cajitas de memoria para direcccionar —> la logica que decodifica todo esto es mas compleja si tengo menos cajitas de memoria que aporten mas bits como es la 2 y medio...etc. Estan organizadas presencialmente . habla tmb el libro de la corrección de errores porque los bits estan separados en distintos chips y eso se puede corregir. Cambia la organización de memoria , tengo un unico chip como puede ser el bios (el bios no está repartido en 2ymedioD, está en uno solo). Cual ocupa menos lugar? cual es mas rápida? dependiendo de si esta

hecha con flip flops o con la simulación de la pila cargada es un 1 lapila descargada es un 0(capacitores). Hay que entender que una vez que yo tengo separado en distintos chips puedo tener por ahí mas bits del bus de direcciones dependiendo de la cantidad de memoria, obviamente tengo mas bits del bus de datos...a eso apunta. qué cambia en el bus de direcciones, en el bus de datos. una puede tener una lógica más compleja que otra... LEER Y ENTENDER ESAS DOS HOJAS EN EL LIBRO DE STALLING.

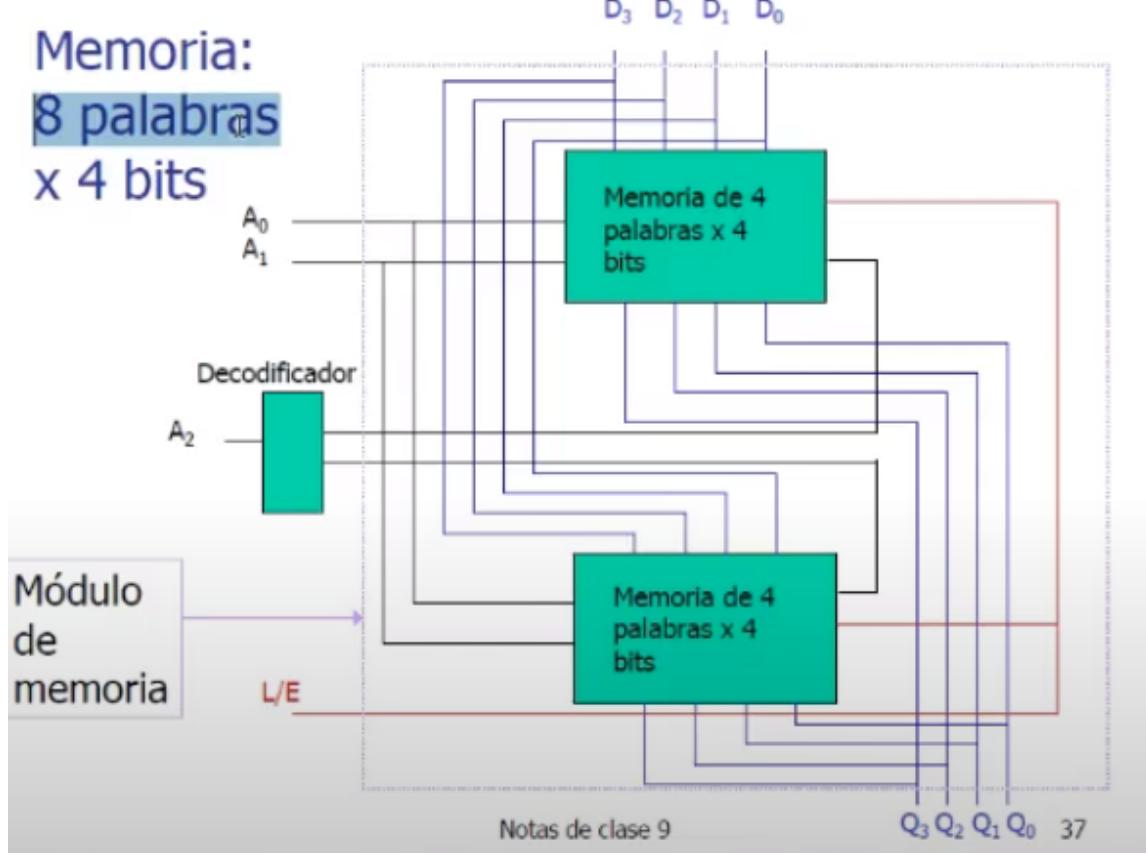
- ▼ cada módulo de memoria cubre el espacio de direccionamiento requerido pero solo cubre una parte de la palabra, o sea **quiero ampliar los bits del bus de datos** → Solución: usar varios módulos en paralelo. ver figura adentro.



Los conectamos de una manera que ampliamos el bus de datos porque tenemos 8 bus de datos en vez de 4, los bits del bus de dirección de memoria son los mismos porque van los mismos a cada chip porque lo que busco es direccionar no mas palabras, sino una palabra mas grande.

- ▼ La longitud de la palabra es la deseada pero los módulos no tienen la capacidad deseada. Solución: cubrir un cierto rango de direcciones con módulos de memoria en serie. Cada modulo estará en direcciones distintas. Ver figura dentro

## Memoria: 8 palabras x 4 bits



▼ yo tengo chips que tienen 1024 posiciones de memoria de 1byte cada una. y despues le piden armar donde uno tiene que expandir el bus de datos y en algunos casos expandir el bus de direcciones. Si yo tengo un chip que puedo sacar a la vez un byte y le digo ahora quiero tener 1024 cajitas pero sacar 4 bytes, voy a tener que sacar cuatro chips donde amplio el bus de datos. La cantidad de direcciones no, porque sigo direccionando 1024 cajitas solo que ahora en vez de un byte van a ser de 4 bytes. **ESTO ES LO DEL ANTERIOR**

- a fin de que todo trabaje de una manera mancomunada yo tengo distintos tipos de memoria(distinta tecnología desde memoria que tengo gran cantidad pero mas lentas, a memorias mas pequeñas con menor cantidad pero mas rápidas)si bien no voy a trabajar a la velocidad de la mas rápida siempre tampoco voy a trabajar de la mas lenta siempre, voy a trabajar un promedio de todas las velocidades. Cual es la idea? a mi me gustaría trabajar mas a la velocidad de la memoria cache xq es la mas rápida. entonces uno podría pensar bueno pongo todo en memoria cache, seria muy caro. entonces pongo un poquito de memoria cache. y después mas principal. Entonces por qué funciona la memoria? porque en realidad el programa que yo ejecuto lo cargo en la memoria principal y ahí lo ejecuto en la memoria principal, entonces la CPU habla con la memoria ppal,

entonces encuentra todo ahí , trabajar a la velocidad de esa memoria, pero cuando hay algo que no encuentra en la memoria ppal - porque todo el disco no entra en la ppal'— vuelve otra vez al disco. Pero al disco no voy muy seguido, voy cada tanto , entonces alguien pensando en eso dijo " vamos a poner en una memoria entre el procesador y la ppal donde vamos a pensar las mismas ideas entre el disco y la memoria principal entre las ppal y la cache. → la cache es mas rápida. la mayoría de las veces el procesador encuentra los datos que quiere en la memoria cache. muy de vez en cuando va a la memoria principal y ahí baja un poco el promedio de tiempo. **Tenemos distintos tipos de memoria: la idea fue la siguiente—> la mayoría de las veces cuando la CPU busca una instrucción la encuentra en la memoria caché.** de vez en cuando tiene que ir a la memoria principal para buscar lo que no encuentra en la cache. Y aun mucho mas de vez en cuando va al disco cuando no encuentra algo. Por qué? Hubo alguien que estudió y encontró que los programas tienen propiedades por como están programados que hace que cumplan esos principios **TEMPORALES Y ESPACIALES** que dicen que si yo accedo una posición de memoria en un tiempo cercano voy a volver a esa posición de memoria, y que accedo a posiciones vecinas de una posición de memoria. Definiendo arreglos , procedimientos que se llaman múltiples veces. **LA JERARQUÍA DE MEMORIA FUNCIONA PORQUE LOS PROGRAMAS TIENEN DETERMINADAS PROPIEDADES.** Si un programa arranca en la posición de memoria 0' y la segunda instrucción la tiene 10 gb mas lejos, no me sirve de nada tener un cache de 8mb, nunca voy a encontrar lo que quiero ahí.

- LA MEMORIA ESTÁTICA ES LA QUE SE UTILIZA PARA CACHE, LA DINÁMICA ES LA QUE SE UTILIZA PARA LA PRINCIPAL, LA RAM.
- ▼ Qué significa que la memoria RAM sea de acceso aleatorio?

que el acceso a cualquier posición de memoria no depende de donde estas parado, tarda lo mismo en acceder. podes acceder a cualquier byte que tardas lo mismo. qué seria algo que no es de acceso aleatorio? una cinta de un cassette, para ir al final tenes que pasar por toda la cinta anterior. La cache también es de acceso aleatorio, la ROM también pero de escritura.
- Los discos que se usan ahora los ssd de estado sólido, tenes todo grabado,es como una memoria.
- Memoria cache- yo tengo que buscar si determinado contenido de la memoria ram está en la cache,por eso se llama ASOCIATIVO, no sé si está ,tengo que

buscarlo, no voy directamente a buscar algo.

▼ Cual es la unidad minima de memoria que se puede tener?

Lo mas chico es un bit. pero la unidad minima direccionable son 8 bits, no hemos trabajado con menos.

▼ Tipos de memorias

Tenemos distintos tipos de memoria: los registros (hechos con flipflops, generalmente los mas rápidos), memoria cache (en general es estática, tamb es mas rapida, hecha con flipflop), memoria dinamica- la ram donde uno carga los programas (en general hecha con el modelo de pilita cargada es un 1 pilita descargad es un 0, no est ahecha con flipflops, tengo que dinámicamente cada tanto recargarlo, reforzar el 1), memoria secundaria como el disco(que puede ser magnético o de estado sólido) y desp los almacenamientos secundarios.

Lo que tiene de piola la diá amica es que ocupa menos lugar, entonces donde hay mayor cantidad de ram la hago con dinámica, la cache como hay menos la hago estática yes mas rápida porque hay menos cantidad. **Hay que tener bien en claro por qué funciona la jerarquía de memoria .**

Tenes chips, quiero armar un banco de memoria mas grande. cada uno tiene una cajita de memoria que almacena 8 bits pero vos necesitas 32 bits, entonces necesitas 4 de esos chips y ahi tenes los 32 bits. Decodificas la misma posición de memoria en los 4 bits a la vez y ahi tenes los 32 bits. Ahora, en general se prefiere hacer eso. Yo prefiero repartirlo en los distintos chips para mejor corrección de errores, si hay errores solo afecta algunos bits y no a todos los 32, etc.

▼ La cache está hecha con flipflop, que son mas rápidos. entonces **por que la memoria dinamica(o la ram) no está hecha con flipflops si son mas rápidos?** esta hecha con el modelo de la bateria o el capacitor en vez de con flipflops.

porque ocupa menos lugar, entonces en un chip podes poner mas memoria, podes poner mas ram que caché- y bueno la memoria dinámica paga el precio. Tengo una bateria por cada bit, entonces que pasa, el 1 se va descargando por la imperfección del mecanismo entonces cada tanto hay que estar refrescnadola para que sigua siendo 1 la pila- por eso se llama dinámica. se prefiere hacer eso solo porque ocupa menos lugar y entonces puedo poner mas memoria. pago el precio de tener que refrescarlo.

▼ jerarquia de memoria solo se aplica a instrucciones o datos e instrucciones y porque

- ▼ qué es el cilindro
- ▼ nombrar las diferentes pistas .
- ▼ Cuantas caras tiene una pista?
- ▼ **Formula de la capacidad de discos**

$$\text{Capacidad} = \frac{\text{bytes}}{\text{sector}} \times \frac{\text{sectores}}{\text{pista}} \times \frac{\text{pistas}}{\text{superficie}} \times \# \text{ de superficies}$$

Se desperdicia espacio en pistas externas.

- Hoy en día se usan zonas para incrementar la capacidad
- c/zona tiene fija la cantidad de bits/pista.
- requieren circuitos más complejos.

- ▼ Formato de discos

Define cantidad, tamaño y función de distintos campos en cada pista

- Hardware: tamaño de sector fijo por marcas físicas.
- Software: tamaño de sector determinado por S.O.

- ▼ Tiempos de discos

Tiempos

- Tiempo de seek (búsqueda)
- Mover al cilindro (o pista) correcto
- Tiempo de latencia (por rotación)
- Esperar que el sector “pase” por debajo de la cabeza

**Tiempo de Acceso:** T.seek + T.latencia

**Tiempo Total:**

T. de Acceso + T. de Transferencia de datos

#### EXAMEN FLOR

- ▼ En qué consiste la normalización, para qué se hace? qué es el bit implícito?ya que estoy hacer una intro breve de Punto flotante.

- ▼ Cómo es el rango en la normalización (pregunta trampa)

▼ Representación de la normalización en el estándar IEE754.

Mantisa, exponente.

### Casos especiales

$E = (255/2047) M = 0 \Rightarrow \text{NAN} (\text{not a number})$

$E = (255/2047) M \neq 0 \Rightarrow \text{infinito}$

$E = 0 M = 0 \Rightarrow \text{Cero}$

$E = 0 M \neq 0 \Rightarrow \text{Denormalizado- en el sentido de fuera de norma- . contempla todos los números entre 0 y 1}$

- 0, mantisa  $2 < -126$  sp
- 0, mantisa  $2^{1022}$  dp

▼ Circuito Combinatorio y Circuito Secuencial - codificador, decodificador ,multiplexor - nombrar los que me sean mas fáciles.

Definiciones:

Ejemplo de circuito secuencial

Aparece en la búsqueda de almacenar un valor lógico el...

**FLIP FLOP SR** → es de alguna manera el elemento de memoria más simple

-2 entradas Set y Reset

-y dos salidas complementarias  $q$  y  $\neg q$

Responde a una nueva tabla de verdad en la que:

- Si  $S=1$  y  $R=0$  La salida será 1 porque la setea.
- Si  $S=0$  y  $R=0$  La salida será la que había en el instante anterior, el valor anterior. La "memoria"
- Si  $S=0$  y  $R=1$  La salida será 0 porque fue reseteada.
- Si  $S=1$  y  $R=1$  nos da un estado prohibido/indefinido, no se puede predecir qué comportamiento tendrá.

**FLIP FLOP D** →

Se basa en el SR Sincronico— pero va a tener una sola entrada y la segunda entrada va a ser la inversa de la anterior. Soluciona el estado prohibido

## ▼ Ciclo de Instrucción

### ▼ Cómo se indica dónde están los operandos en una instrucción? Mencionar la del stack.

Mediante los modos de direccionamiento.

El del stack → Push Pop Stack pointer

tiene asociado un registro apuntador o puntero de pila cuyo valor es la dirección tope de pila o stack.

### ▼ Jerarquía de Memoria. Qué es ? porqué funciona?. Ejemplos

Funciona por los principios que cumplen los programas de **Localidad y Espacialidad**

La cache está situada entre el procesador y la memoria principal, entonces conviene almacenar los datos que se usaron recientemente en la cache. El procesador necesita un dato, primero se fija si está en la cache y si no está lo busca en memoria, lo almacena en la caché,etc.

### ▼ Diferencia entre Memoria RAM Dinámica y Estática

Por ahí puedo empezar por decir que RAM es por Random Access memory osea, memoria de acceso aleatorio. Es aleatoria porque se puede acceder a cualquier celda de memoria en el mismo tiempo (es decir , lleva el mismo tiempo) independientemente de la posición en la estructura en la memoria en la que se esté. *Memoria Volátil*.

Tenemos la estática y la dinámica.

→ La estática o (SRAM) es la basada en Flip flops. *Esta que es super rápida la usamos en la caché.*

→ La dinámica o (DRAM) es la basada en transistores (capacitores) . La ventaja de estas es que almacenan más información que la Estática en la misma superficie porque los capacitores son mas chicos que los flipflops. La desventaja es que hay que refrescarlo dado que los capacitores se descargan(se descarga la información) . *Esta la usamos como memoria principal.*

### ▼ Qué organizaciones de memoria conoces?

La 2D y la 2 1/2 D D PORQUETIENE DECODIFICADORES.

**En la 2D todos los bits (de una misma palabra) están en un mismo chip.**

Esto mismo la hace muy **Larga y Estrecha**

Cada línea de Selección de palabras tiene que tener un manejador y conectarse al decodificador → **Desventaja ocupa mucha superficie.**

#### **Dificulta el uso eficaz de los circuitos correctores de Error**

Cada línea horizontal se conecta a cada posición de memoria, seleccionando un renglón o fila. Las líneas verticales conectan cada bit a la salida.

Tiene **un decodificador** que está en el chip y tiene  $2^n$  salidas para n entradas. (bits del bus de direcciones) Crece junto con el tamaño de la memoria.

**En la 2 y media D** Funciona igual que la 2d PERO **Los bits de una misma palabra están dispersos en distintos bits.**

En la 2 y media al estar los bits dispersos en distintos chips hay **menor probabilidad de error.**

**La dirección se divide en dos partes: una de renglón y otra de columna (fila y columna)** es decir, **hay dos decodificadores.** Al usar decodificación separada de filas y columnas, reduce la complejidad de cada decodificador.

#### ▼ **Estructura de los discos rígidos → llenar con lo del pdf**

Platos

- Superficies cubiertos , material magnético.

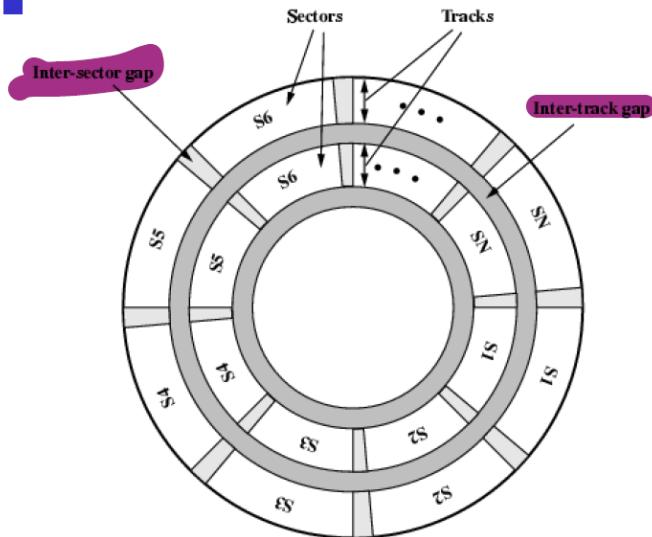
## Mecánica de lectura y escritura

- ❖ Lectura y escritura es a través de una cabeza transductora (bobina).
- ❖ Durante lectura/escritura, la cabeza es estacionaria y el plato gira.
- ❖ Se almacenan ceros y unos por medio de la magnetización de pequeñas áreas de material.

## Organización de los datos

- ✓ Anillos concéntricos: **pistas** ó **tracks**.
  - Espacios (gaps) entre pistas
  - Reducir gaps para aumentar capacidad
  - Mismo N° de bits por pista
  - Velocidad angular constante
- ✓ Pistas divididas en sectores
- ✓ Mínimo tamaño de bloque: **sector**
- ✓ Bloque: más de un sector (**cluster**)

## Pista y Sector



- Un número entero de sectores se graban en una pista.
- El sector es la unidad de transferencia de/hacia el disco.

Nº 10 de Clase 10

25

## Sector típico

Un sector

10 bytes	512 bytes	12 bytes
Encabezado	Datos	Código para errores

✓ Sucesión o serie de bits divididos en campos

- **Encabezado** con información para sincronizar la lectura e identificar el sector.
- **Datos** con longitud en bytes expresada usualmente como potencia de 2.
- **Código para errores** con información para detectar y/o corregir posibles errores.

▼ Qué ventaja tiene trabajar con cilindros? tiene que ver con cómo trabajan las cabezas del disco.

Cuando están apilados voy a tener un cabezal por cada uno de las caras del palto platos y si se usan cilindros es mas rápida la recuperación de los datos porque se puede hacer en simultaneo.

**Las cabezas se mueven todas juntas y por eso con el cilindro se acceda mas rápidamente , se lee mas rápidamente si se escribe en una cilindro.**

▼ **En qué afecta la velocidad de giro del disco? es bueno que gire muy rápido? En cuanto al tiempo de acceso.**

Si gira mas rápido el tiempo de acceso va a ser menor, el tiempo de latencia que tengo que esperar a que se posicione va a ser menor.

▼ **Concepto de RAID(?) del disco**

Los raid son **múltiples discos físicos** que el procesador va a utilizar como si fuesen 1. Tienen niveles, hay diferentes formas de utilizarlos. niveles, no jerarquía.

**.Puedo tener accesos múltiples dividiendo la info, puedo tener copias espejadas. Y que pasa si yo pierdo un disco porque dividí la información? y no tenía espejado, tengo la información de paridad(un disco donde tengo todas las paridades con los bloques distribuidos???) que me permite recuperar la info que perdí.**

▼ **Modem MODEM (MOdulador, DEModulador) AD/DA**

El modem es un **dispositivo que permite conectar dos computadores remotos** utilizando la **Línea telefónica** de forma que puedan intercambiar información entre sí.

Cumple la función de **modularizar y demodularizar la información.**

**Convierte señales '0' y '1' en tonos de audio. Los módem envían datos como una serie de tonos a través de la Línea telefónica.** Los tonos se "encienden" (ON) o apagan (OFF) para indicar un 1 o un 0 digital.

Tasa Bits/seg (bps) es el número de bits enviados por segundo.

Tasa Baudio (baud rate) es el número de cambios de señal por segundo . El baudio es el número de veces que esos tonos se ponen a ON o a OFF.

Es posible enviar varios bits por baudio, señalando en frecuencias diferentes.

La información que maneja el computador es digital.Por las limitaciones físicas de las líneas de transmisión, **no es posible enviar información digital a través de un circuito telefónico(señales analógicas).**Es necesario un

proceso de transformación de la información. Durante este proceso la información se adecua para ser transportada por el canal de comunicación. —> **modulación-demodulación y es el que se realiza en el modem a través de conversores.**

Existen distintos **sistemas de modulación de una señal analógica para que transporte información digital**, siendo los métodos más sencillos **la modulación de amplitud y la modulación de frecuencia**. Otros mecanismos como la modulación de fase o los métodos combinados permiten transportar más información por el mismo canal.

### **Smart" Modems**

→ A veces llamados "Hayes compatible"

Computadora controla:

- discado
- establece la tasa de bit (bit rate)
- programa contestador, re-discado, etc.
- capaz de compresión de datos

### **▼ Periféricos de salidas...video. Rom de caracteres? Para qué se usaba?**

→ Memoria de visualización

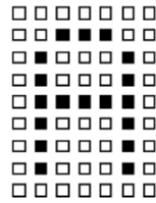
#### **En los Monitores alfanuméricicos**

(En memoria se almacenan sólo códigos de carácter)

La ROM de caracteres lo que hace es **convertir los códigos de carácter en pixels**. Por carácter se generan varios pixels sucesivos en varias líneas sucesivas.

#### **Los bits de una línea son leídos serialmente**

- Se accesa 9 veces a la misma posición horizontal y sucesivas posiciones verticales



**(a) Character matrix**

▼ Cómo funciona un monitor alfanumerico? y uno grafico?

El monitor alfanumerico es uno de los dos tipos de Monitores de video. (el otro es el GRÁFICO)

**Monitores alfanuméricos**

- En memoria se almacenan sólo códigos de carácter
- Los códigos de carácter se convierten en pixels por una **ROM de caracteres**
- Por carácter se generan varios pixels sucesivos en varias líneas sucesivas

**Controlador de Video (alfanumérico)**

Contadores cuentan

- los 7 puntos en un carácter,
- los 80 caracteres a lo ancho de la pantalla,
- las 9 líneas en un carácter, y
- las 64 filas de caracteres desde arriba hacia abajo

**Monitores gráficos (bit mapped)**

- Cada pixel es representado por bits en memoria
- Los visualizadores B/N pueden usar un bit por pixel
- En gama de grises/color requerirán varios bits por pixel

▼ Impresoras inject INK JET. Como forman el carácter?

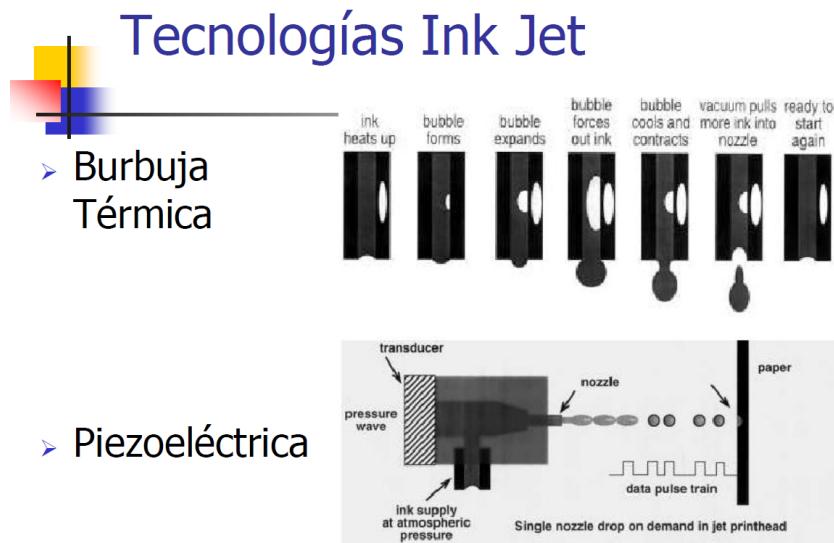
Trasductor ultrasónico (otro nombre??) lanza pequeños chorros de tinta a los puntos correctos con la cabeza moviéndose sobre el papel.

→ Blanco/negro y Color

## Tecnologías:

→  Burbuja térmica : se calentaba la tinta, se formaban unas burbujas, las burbujas se expanden y empujan fuera la tinta. La burbuja se enfria y se contrae, se empuja mas tinta y se puede empezar de nuevo.

→ Piezoeléctrica



Traditionally, inkjets have had one massive attraction over laser printers; their ability to produce color. The down side is that although inkjets are generally cheaper to buy than lasers, they are more expensive to maintain. Cartridges need to be changed more frequently

### ▼ Si querés generar un original y copias con carbónico.. sirve ese tipo de impresora?

Si, En las aplicaciones que no requerían impacto (por ejemplo, la impresión de copias de carbono), la inyección de tinta era superior en casi todos los aspectos: funcionamiento comparativamente silencioso, mayor velocidad de impresión y calidad de salida casi tan buena como la de una impresora láser. A mediados del decenio de 1990, la tecnología de inyección de tinta había superado a la de matriz de puntos en el mercado general.

**Dot-Matrix Printers (matriz de puntos)** → They are relatively expensive and do not produce high-quality output. However, they can print **to continuous stationery multi-page forms**, something laser and inkjet printers cannot do.

Matriz de Puntos:

- Arma los caracteres
- Punzón golpea una cinta entintada y marca el papel
- **Tantos punzones como alto de la matriz de caracteres**
- Baja resolución

Imprime una columna por vez

- Puede usar una ROM de caracteres
- La ROM se lee en paralelo por columna, en vez de serie por fila como en el video alfanumérico

La tecnología de impacto de matriz de puntos sigue siendo utilizada en dispositivos y aplicaciones como:

- Caja registradora
- Cajeros automáticos
- Sistemas de detección y alarma de incendios
- Las impresoras de impacto de matriz de puntos como la Epson (que no necesitan **papel térmico**) son más tolerantes a las condiciones de funcionamiento calientes y sucias que se encuentran en muchos entornos industriales, lo que permite que se utilicen incluso en entornos como las cocinas de restaurantes o cafeterías.

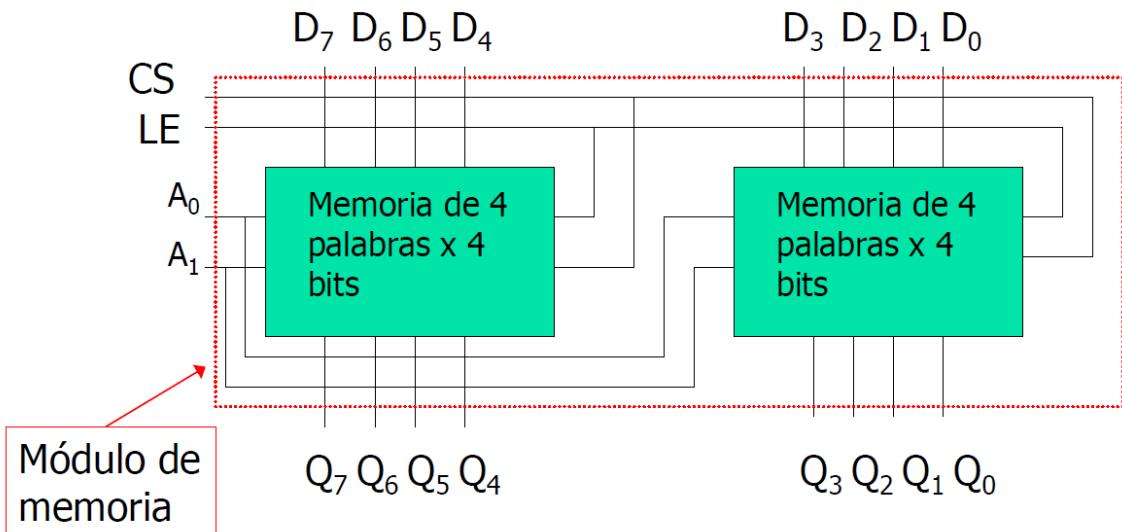
#### PROBLEMAS MEMORIAS → No entiendo nada este tema

- ▼ Cada módulo de memoria cubre el espacio de direccionamiento requerido, pero sólo cubre una parte de la palabra.

**Solución: usar varios módulos en paralelo**



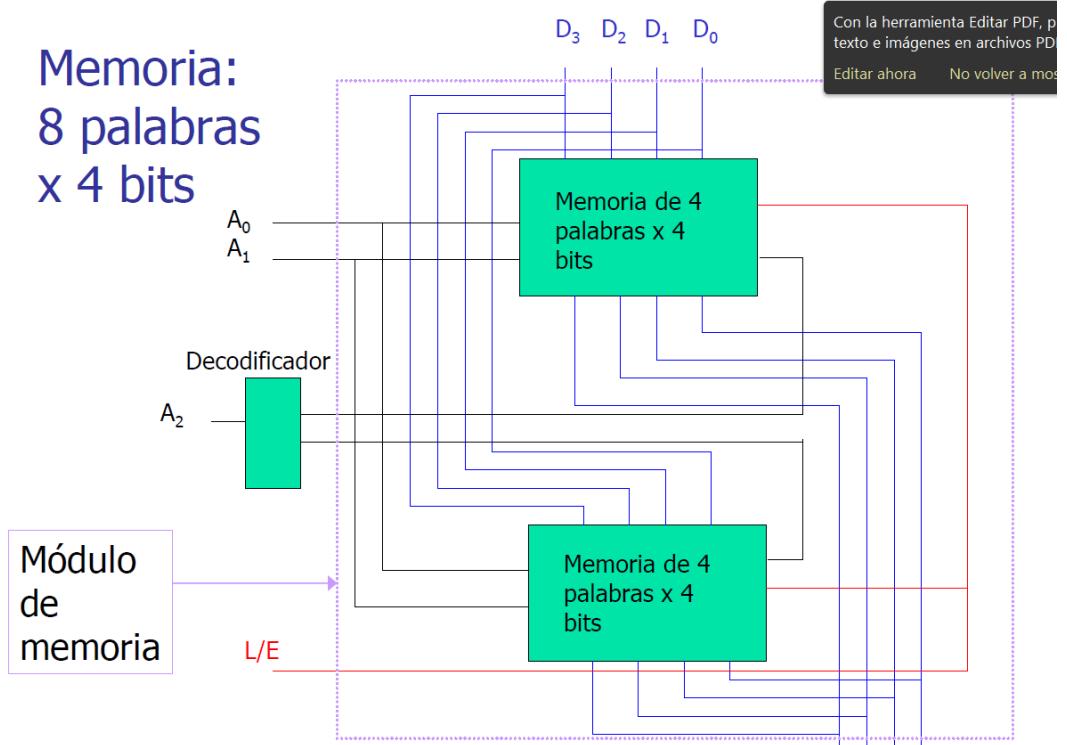
# Memoria: 4 palabras x 8 bits



- ▼ como trabaja la Memoria, como sabe q es un dato y como sabe q es una instrucción..
  - ▼ La longitud de la palabra es la deseada, pero los módulos no tienen la capacidad deseada.

Solución: cubrir un cierto rango de direcciones con módulos de memoria 'en serie'.  
-Cada módulo 'estará' en direcciones distintas.

## Memoria: 8 palabras x 4 bits



24/2 10:14] Letizia ✨: Te menciono un toque todo.

Arrancmo con normalizacion, cómo es el rango (pregunta trampa) IEE. Preguntas trampas de normalizacion y casos especiales.

Después paso a circuitos secuenciales y combinacionales. Cuales sln los combinacionales, multiplexor, decodificador(que lo dije mal) desplazador. Secuenciales, registros y contadores, estos coko son y que hacen, como estan formados

[24/2 10:14] Letizia ✨: De decodifcador me preguntaron sobre memoria 2D y 2D1/2

[24/2 10:14] Letizia ✨: No la sabia 😅

[24/2 10:16] Letizia ✨: Después seguimos con ciclo de instrucciones, aproveche y le mande sobre maquina de N instrucciones, me preguntaron que característica especial tiene la ALU en caso de maquinas de 1 dirección (eso no sabía)

Alu, registros. Registro contador de la ALU (que no sabía)

Jerarquia de memoria (esta tenes que saberla bien HASTA CON EJEMPLOS me preguntaron ejemplso) y si la jerarquia de memoria solo se aplica a instrucciones o

datos e instrucciones y porque (en esa la pilotee bien)

Y desoues periféricos. Ahí me mataron

Disco, como es, como funciona, velocidades

Raid

Impresora

Pantalla

Video

Rom de caracteres en cada una

[24/2 10:17] Letizia ✨: Perdón por el mensaje medio así de la nada, pero eso es todo.

Ah y definieron la nota con Flip Flops. Y tuve que hablar TOOOODO de flip flops.

[24/2 10:34] Letizia ✨: Ah, y SRAM y DRAM

Ah y les tuve que dibujar un disco y nomrar las partes

- -----

Buenas gente.. Me tomé un rato para descansar pero lesuento, me tocó con villagarcia y otro más (q no recuerdo el apellido) particularmente repasan la materia como "hilando" los conceptos y luego saltando entre los temas..

Quiero decir comienza uno, listo vamos al otro y así... Estuve. Más de una hora y aún así me dijeron q podríamos seguir más, pero había otro alumno 🤓 esperando..

- Arrancó con von neuman, que es q hace como está,
- como trabaja la Memoria, como sabe q es un dato y como sabe q es una instrucción..
- Sistema de numeración.. Todos, bss bcs ca1 ca2 ( q significa complementar un número, x ej.. Podes complementar en base 10?)
- ieee q es como es, situaciones especiales, xq es así.. Como funciona el rango, la resolución.. Por que el rango es así... ( trampa por la normalizacion)...
- Tocando la parte digital, explicar tipos de circuitos.. Cuales conoces, como son.. Para q son.. Ahí me pusieron dibujos de unos circuitos (me mató)

te ayudan bastante pero no hacía mucha agua..

- Me preguntó cual es el circuito combinatoria más importante del cpu... Y era la alu..

- Te dan dibujos y los tenes q explicar decir q son y cómo son y si podes, decir q hacen..
- Hablamos por último de periféricos.. Impresora.. Me tocó térmica..

Charlar un poco.. Afine guitarra.. Me parece q lo dije un poco como el toor.. Pero fue suficiente.. Fue un poco de todo eso, te acompañan bastante..

## Examen Sofía

- ▼ Representación en punto flotante: Utiliza la normalización de los nros. ¿En qué consiste la normalización y para qué se usa?

La Normalización viene a solucionar un problema de punto flotante donde un mismo nro puede tener diferentes representaciones, cambiando el exponente y la mantisa. La primera que vimos era la que la mantisa empieza con 0,1 (sirve para BSS y BCS). Como todas empiezan con 0,1, se dieron cuenta que podían obviarla y así ganar un bit para aumentar la precisión del nro representado, mejora la resolución. Cuando escribo el nro sí incluyo ese bit, pero no en la ocupación en memoria, se almacena a partir del bit que le sigue al 0,1.

La normalización implica que la mantisa es fraccionaria (RELLENAR CON LO DE RUNCO)

Luego vimos la normalización que utiliza el standar IEEE754, donde cambia y pasa a ser 1, en la mantisa. Nos damos cuenta que hay nros que no vamos a poder representar por su propia naturaleza (los que vayan del 0 al 1) esto provoca que el rango esté compuesto de dos intervalos que se cortan alrededor del cero. Nunca lo tocan, no llego al 0.

Los casos especiales son 4 y se reconocen por el exponente:

- E todos bit en 1 y M = 0 —> Infinito
- E todo 1 y M <> 0 —> Nan
- E todo 0 y M = 0 —> Cero
- E todo 0 y M <> 0 —> Denormalizado (fuera de la norma)

En este caso se mira la tablita y dependiendo si es simple o doble precisión, se pone

+/- 0,mantisa y exp 2 a la -126 si es simple precisión

+/- 0, mantisa y exp 2 a la - 1022 si es doble

El 1, no se almacena, funciona similar al bit implícito. La mantisa está en BCS o ???????? y el exponente en exceso.

Tiene dos subformatos, el de simple precisión y el de doble precisión.

#### ▼ Diferencias entre circuito combinatorios y circuitos secuenciales

Combinatorio: sus valores de salida dependen de los valores de entrada en el momento actual. Ejemplo: ALU, **multiplexor** (tiene varias entradas y una sola salida, que según unos selectores que tiene puedo seleccionar una entrada particular), decodificador (por ejemplo si tengo 3 bits que entran al **decodificador**, como trabaja como potencia de 2, va a poder direccionar 2 a la 3 cajitas de memoria)

Secuenciales: sus valores de salida dependen de sus valores de entrada pero también del valor anterior inmediato.

**Ejemplos:** Memoria?. Flip Flops: SR (que tiene dos entradas Set pone estado en 1 y Reset pone estado en 0, y dos salidas Q y Qnot que son complementarias, si ambos están en 0, recuerda el estado anterior y si ambos están en 1, es el estado prohibido).

Esto del estado prohibido se soluciona con el **flip flop JK** (sincrónico), que ahora devuelve ese estado prohibido como el estado anterior negado.

**Sincrónico:** Significa que la habilitación de cambios de estados se da por el compás de un reloj. Hasta que el reloj no da la señal, no suceden cambios, no importa si las entradas cambian, ni nada.

#### ▼ Ciclo de instrucción

Se divide a grandes rasgos en dos etapas

- Búsqueda de la instrucción
  - Se va a buscar la instrucción a la dirección de memoria que está almacenada en el PC, que es justamente la dir. donde se encuentra la próxima instrucción a ejecutar.

- Esa dirección va al MAR (qué es?) y queda a la espera en ese espacio intermedio entre la CPU y el bus de direcciones que me lleva a la memoria dónde está esa instrucción.
- La encuentra en memoria y la manda a la CPU mediante el bus de datos. Llega al MBR (qué es???) y libera al bus de datos.
- Del MBR llega al IR
- La UC interpreta esa instrucción, que al ser autocontenido, tiene toda la información necesaria para su ejecución.
- En caso de necesitar otro operando por ejemplo, el ciclo de instrucciones se repite pero en vez de guardarse en el IR que es sólo para las instrucciones, se guarda en uno de los registros temporales.

¿Cómo encuentra los operandos?

A través de los modos de direccionamiento **RELEENAR con breve definición**

- Directo
- Indirecto
- Indirecto por registro
- Stack —> ¿con qué instrucciones se usa? con push y pop, y el registro SP (apunta al tope de la pila). Cuando hago Push, el SP ¿qué hace?

El objetivo de esto es simplificar/acortar las instrucciones y tener lugares de almacenamientos más grandes.

#### ▼ Jerarquías de memoria

En un computador hay distintas memorias con distintas características y velocidades: algunas muy grandes pero muy lentas, y otras más rápidas pero muy chicas. Entonces se trata de buscar un equilibrio entre la cantidad que ponemos de cada una y cómo las estructuramos entre sí para llegar a funcionamiento más eficiente posible a partir de un costo razonable de recursos.

Pirámide: La memoria más costosa y rápida son los registros, luego la caché, la memoria principal, luego los discos y luego los periféricos.

Todo esto funciona a partir de los principios de localidad que cumplen los programas.

- Principio de localidad espacial: si accedés a una posición de memoria, es muy probable que luego accedas a una cercana.

El principio de localidad espacial me dice que conviene trabajar copiando de memoria principal a caché, copiando de bloques digamos, no de a una palabra por vez por ejemplo. Dado que vamos a tener en la memoria principal más cantidad de bloques que en la caché, **¿cuál es la metodología que usamos para decidir dónde poner los bloques desde la memoria principal a la caché? MAPEO??**

- Principio de localidad temporal: si accedés a una posición de memoria, es muy probable que en un corto periodo de tiempo, vuelvas a acceder a la misma, es por eso que esas posiciones que acabo de usar, las guardamos en la caché para que las vaya a buscar ahí y no a otra memoria que tarda más.

▼ ¿Qué se entiende por memoria RAM dinámica y estática?

Estática: **RELENAR**. Flip flops. Para la memoria caché se usa este tipo de memoria.

Dinámica: Está basada en transistores, que ocupan menos espacio que los flip flops, por lo que puedo poner más cantidad. Sin embargo los transistores se cargan / descargan, por lo que para su correcto funcionamiento necesitan ser refrescados continuamente. Cosa que no pasa con los flip flops que ocupan más espacio pero son más rápidos. Para la principal se usa este tipo de memoria.

▼ Dentro de la memoria principal hay dos organizaciones:

2D —> Tiene un solo decodificador, el mismo para líneas horizontales y verticales. Con el decodificador 2D encontrás una palabra ponele.

2 1/2 D —> Tiene dos decodificadores (esto es una ventaja porque tengo un decodificador para las líneas horizontales y otro para las verticales, lo que hace que el decodificador sea más simple) Con el decodificador 2 1/2 D encuentro un bit. **ES ASÍ???**

▼ Pista, sector, cilindro. ¿A qué hacen referencia esos conceptos?

Se refiere a la estructura de organización del disco. **BUSCAR**

▼ RAID **BUSCAR**

Tienen 3 funcionalidades que están buenas

- Sirve para hacer una copia espejada de un disco
- Paridad?
- Separa los bloques en distintos discos (para qué sirve?)

Todo esto es para tener estructuras que permitan un almacenamiento seguro.

▼ Periféricos: Concepto de Módem

Modularizar y demodularizar. Su función es comunicar computadoras de manera remota a través de una línea telefónica? El trabajo de modularizar y demodularizar es justamente convertir esas señales analógicas en digitales, que son las que usan los computadores. **BUSCAR**

▼ ROM de caracteres, ¿en qué periférico la utilizamos?

-En el monitor alfanumérico. En la memoria está almacenado el código de los caracteres y el ROM de caracteres convierte ese código en píxeles y pinta los caracteres según corresponda. A GRANDES RASGOS.

-Impresora

---

▼ Representación de datos

Las computadoras almacenan datos e instrucciones en memoria (esto lo introdujo VN)

Para ésto utilizan el sistema binario (1 y 0)

Este sistema puede almacenar 4 tipos básicos de datos binarios

- Números enteros con y sin signo
- Números reales con signo
- Números decimales codificados en binario (BCD)
- Caracteres

Para representar números enteros tengo los siguientes sistemas:

- BSS —> Represento 2 a la N nros distintos. (N=cant de bits) y el Rango va de 0 a 2 a la n -1

- BCS —> Con  $n$  bits. 1 representa al signo y  $n-1$  a la magnitud. 0 en el bit de signo representa positivo y 1 representa negativo. Rango va de  $-(2^{n-1})$  a  $(2^{n-1}-1)$ . Tiene dos representaciones posibles del cero. El signo no forma parte del nro, representa sólo al signo, el resto a la magnitud. Intervalo es simétrico (el más negativo que puedo escribir y el más positivo es el mismo valor)
- **Ca1**—> Todos los bits forman parte del número -soluciona el problema de la pérdida del bit con el signo- Si el nro es positivo se representa igual que en binario sin signo (BSS), va a comenzar con 0. Si es negativo, se tiene que aplicar el complemento a 1 para representarlo (se representa en positivo y luego se invierten todos los nros). El rango es el mismo que en BSS pero un negativo se representa distinto.
- Ca2 —> Si el nro es positivo se representa igual que en binario sin signo (BSS), va a comenzar con 0. Con el negativo, copio (mirando desde la derecha) todos los bits hasta el primer 1, y luego invierto los que siguen. Rango va de  $-(2^{n-1})$  a  $(2^{n-1}-1)$ . Ya no tengo el 0 negativo, lo recuperé al tener un negativo más, por esto tengo un **intervalo asimétrico**.
- Exceso —> Representar un exceso es tomar la representación de un numero en Ca2 y sumarle una cantidad fija (un 1 adelante y 0 atrás). El rango en exceso y en Ca2 es el mismo, puedo representar la misma cantidad de nros y los mismos nros, pero las combinaciones de 1 y 0 que representan a esos nros, cambia dependiendo el sistema.

La cantidad de representaciones que puedo hacer de nros distintos, depende siempre de la cantidad de bits, no del sistema.

Los números positivos se representan como siempre, BSS, BCS, CA1 y CA2 los positivos se representan igual.

- Rango: desde el nro menor al mayor que puedo representar
- Resolución: diferencia entre dos nros consecutivos

### Teorema fundamental de la numeración

Este teorema nos dice que cualquier sistema de numeración posicional está representado por un dígito  $x$  la base elevada a una posición. A la izq, el número

que da es siempre el numero en base 10. En los sistemas de representación posicional, cada posición tiene un peso distinto.

Los sistemas de representación que nosotros vemos, se basan en este teorema.

### Teorema Fundamental de la Numeración

$$N^o = \sum_{i=-m}^n (dígito)_i \times (base)^i$$

**Representación en punto fijo**—> todos los nros a representar tienen la misma cantidad de dígitos y la coma fraccionaria está siempre en el mismo lugar. Es posible representar un rango centrado en el 0.

**El primer problema que solucionamos fue la representación de nros negativos, ahí es cuando ideamos sistemas como Ca2. Luego el tema de la coma, ¿por qué tengo que ver un formato nuevo si con los que tengo puedo representar nros con coma? para tener un rango de números muy pequeños a muy grandes representados en muy pocos dígitos. ¿por qué? porque tener números con muchos dígitos hace mas lenta las cuentas, ralentiza la maquina.**

Es por esto que se idea la coma o punto flotante.

**Representación en punto flotante** —> En vez de representar el nro completo, representamos sólo la **mantisa y el exponente**, me hacen ahorrar un montón de bits, y que la aritmética, o sea sacar cuentas, sea más rápido. **La idea es que en punto flotante, necesito muchos menos dígitos para representar un nro (menos bits). Ocupa menos lugar, es más eficiente, mucho más rápido.**

En punto flotante podemos representar un rango mayor de nros, con menos cantidad de bits. Sin embargo pagamos el precio que la resolución no es constante, cambia a lo largo del exponente. Esto se soluciona agregando más bits a la mantisa y más bits al exponente.

### Formato de un nro en punto flotante

S	Exponente	Mantisa
---	-----------	---------

**El problema con punto flotante es que puedo tener varias representaciones de mantisa y exponente para un mismo nro.**

### Normalización -siempre para la mantisa-

Nace para tener una única representación de mantisa y exponente para representar un nro en coma flotante.

- La mantisa está normalizada cuando la podemos escribir como 0,1
- La mantisa normalizada nunca se hace 0 (doble intervalo de rango porque no toca el cero de ninguno de los dos lados - y +)
- Un exponente negativo no hace que el nro sea negativo, eso lo hace la mantisa.
- Como el mínimo que puedo escribir siempre partirá del 0,1 en un sistema normalizado no podemos representar el 0 en la mantisa.
- Que todas las mantisas normalizadas empiecen con 0,1 me va a asegurar que haya una sola representación para el mismo número.
- Trabajamos con mantisas fraccionarias y también sirve para mantisas en BSS y BCS (ya que el signo está en otro bit y no pasa nada con el 0,1). No en Ca1 ni Ca2.

Como todas las mantisas normalizadas empiezan igual (0,1) me doy cuenta que podría no escribirlo y que esté implícito:

### Bit implícito

Se trata de dar por descontado ese 0,1 y así ganar un bit más para la representación del nro. Esto me da la ventaja de tener más precisión del nro representado, **mejora la resolución**. Cuando escribo el nro sí incluyo ese bit, pero no en la ocupación en memoria, se almacena a partir del bit que le sigue al 0,1. El bit implícito no modifica el exponente, pero agrega un bit más a la

mantisa, la vuelve más precisa porque tengo más posibilidades de representación.

Rango en mantisa normalizada —> Siempre que tenga que calcular rangos en un sistema normalizado voy a tener dos "pedazos" de cada lado, tengo dos intervalos, la unión entre dos intervalos porque hay una zona cercana al 0 que no está representada, entonces se corta y luego retoma, por eso tengo dos intervalos de representación posible.

Resolución en punto flotante —> es la diferencia entre dos representaciones sucesivas y varía a lo largo del rango, no es constante como en punto fijo.

Error absoluto —> es la diferencia entre el valor representado y el valor a representar. Si pude representar el valor exacto que quería representar, el error absoluto es 0. Siempre el peor error, el error máximo, es la mitad de la resolución, el peor caso.

## Estándar IEEE754

Formato estándar de escribir números en punto flotante. Se escribe para que todas las maquinas utilicen el mismo formato, si todas las maquinas hicieran cálculos con distintos formatos de números sería casi imposible.

Hay dos tipos

- Simple precisión —> 1 bit signo, 8 de exponente y 23 de mantisa
- Doble precisión —> 1 bit de signo, 8 de exponente y 52 de mantisa

## Normalización en IEEE754

- La mantisa es siempre 1, (UNO COMA)
- Acá el exponente está representado en exceso pero un exceso un tanto distinto. La lógica es similar, porque podemos partir de  $Ca^2$ , PERO el exceso que sumaremos va a ser lo distinto. en vez de 10000 ahora es 00001.

- Como es un nro normalizado, el 0 y las zonas cercanas a él, no están representadas.
- Cada vez que el exponente sea todo 1 o todo 0, es un **caso especial**, esto quiere decir que no voy a buscar representar el nro de la forma normalizada, sino que es una **representación particular**.
- En el rango de exponente no tenés el -128 ni el -127 porque sirven para representar los casos especiales de la tablita.

#### 4 casos especiales

- E todos bit en 1 y M = 0 —> Infinito
- E todo 1 y M <> 0 —> Nan
- E todo 0 y M = 0 —> Cero
- E todo 0 y M <> 0 —> Denormalizado (fuera de la norma)

En este caso se mira la tablita y dependiendo si es simple o doble precisión, se pone

+/- 0, mantisa y exp 2 a la -126 si es simple precisión

+/- 0, mantisa y exp 2 a la - 1022 si es doble

#### Banderas aritméticas

Bits de condición que el procesador cambia automáticamente en respuesta a una operación aritmética. Hay 4: overflow, carry, negativo y 0.

¿Para qué nos sirven estos flags? Por eje en la ejecución de estructuras de control, la máquina chequea las banderas. Es más rápido que estar analizando el resultado, son para tomar decisiones más veloces.

Me dan información para tomar decisiones.

La idea es que cada vez que el procesador ejecuta una operación aritmética (tomamos sólo sumas y restas nosotros) acomoda estos bits bandera en forma automática en respuesta al resultado que dio dicha operación aritmética.

**El sistema de cómputo solo puede hacer cuentas en binario sin signo y en complemento a 2.** Por eso las banderas solo responden a esos dos tipos de cuentas.

**Z (cero)** —> Vale uno si el resultado de la operación dio todos los bits en cero.

**C (carry)** —> En la suma vale 1 si hay acarreo del bit más significativo. En la resta vale 1 si hay borrow hacia el bit más significativo. Cuando la operación involucra nros sin signo, indica condición FUERA DE RANGO. Está asociada a BSS

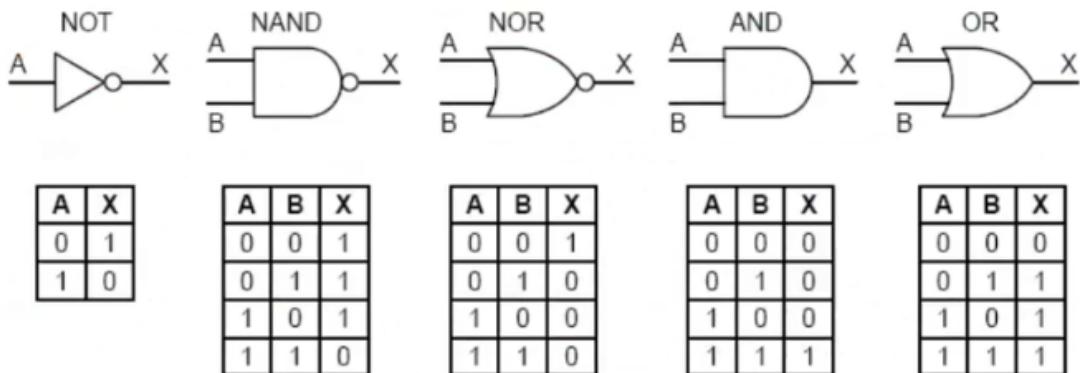
**V** —> El bit de overflow trabaja para Ca2. En 1 indica condición fuera de rango (desborde)

**N** —> Es 1 si el resultado es negativo. Copia al bit más significativo.

## ▼ Lógica digital

Un circuito digital es en el que están presentes dos valores lógicos

Las compuertas lógicas son dispositivos electrónicos que realizan operaciones con estos dos valores.



Combinando estos dispositivos armo distintos circuitos donde las variables y funciones sólo pueden tomar los valores binarios: 1 y 0. Se llama álgebra booleana.

Las tres maneras que tengo de representar estas compuertas es mediante **ecuación, símbolo gráfico y tablas de verdad**. La tabla de verdad tiene todas las entradas posibles (para nosotros A y B) y todas las salidas posibles.

La **ley De Morgan** es importante porque une producto con suma, porque me da la fórmula de conversión entre NAND Y NOR, que son compuertas universales que me permiten emular a todo el resto y construir circuitos usando sólo este tipo de dispositivo. Morgan me hace conseguir esas compuertas a partir de cualquier otra. Esto simplifica mucho la construcción de circuitos y abarata los costos.

### Tipos de circuitos

#### Combinacionales

## Circuitos Combinacionales o Combinatorios

- Responden a los valores lógicos en las entradas, la salida está determinada exclusivamente por los valores de las entradas en ese instante.
- Si cambia la entrada, cambia la salida.
- Los valores pasados de las entradas no influyen en los valores de las salidas.

Ejemplos de circuitos combinacionales son

- ALU —> El más importante de todos. Se trata de un circuito integrado con la capacidad de realizar diferentes operaciones aritméticas y lógicas (es decir, del álgebra de Boole), con dos palabras de n bits. Como salidas tiene los 4 bits del resultado, más una salida comparador ( $A = B$ ) y salidas de acarreo.
- Multiplexor—> **conecta varias entradas con una sola salida.** Se usan en circuitos digitales para controlar el enrutamiento de señales de datos. Un ejemplo es la carga del conectador de programa (PC).

- Codificador—>Circuito combinacional con  $2N$  entradas y  $N$  salidas, cuya misión es presentar en la salida, el código binario correspondiente a la entrada activada.
- Decodificador —> Circuito combinacional con varias líneas de salida, con una sola de ellas seleccionada en un instante dado, dependiendo del patrón de líneas de entrada. Su función principal es la de redireccionar espacios de memoria. Un decodificador de  $n$  entradas puede direccionar  $2^n$  espacios de memoria.
- Comparador —> donde si todos los bits A son iguales a los bits B, la salida es 1.
- Desplazador —> según el valor de la entrada C, se correrán un lugar a derecha o izquierda.

## **Secuenciales**

Aparecen por la necesidad de realizar circuitos más complejos, como memorias.

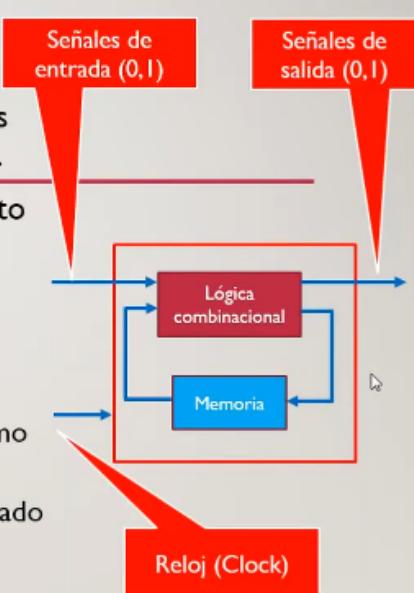
Los circuitos secuenciales son circuitos donde las salidas dependen tanto de las entradas como del “estado interno” del circuito.

El estado interno del circuito:

- Los circuitos secuenciales tienen la característica de retener (“almacenar”) internamente valores.
- Estos valores internos pueden mantenerse aunque las entradas se hayan modificado: concepto de “almacenamiento”.

## CIRCUITO LÓGICO SECUENCIAL

- Definición: circuito lógico cuya salida depende de los valores actuales y pasados de las señales de entrada.
- Aparecen lazos de realimentación. Salidas del circuito pueden actuar como valores de entrada.
- Un circuito lógico secuencial está formado por:
  - Señales de entrada y salida (señales binarias).
  - Señal de reloj (señal binaria con forma periódica).
  - Lógica combinacional (determina la salida y el próximo estado).
  - Almacenamiento (mantiene información sobre el estado actual).



La salida va a depender ya no sólo de los valores de entrada actuales (como en los circuitos combinacionales) sino también del estado anterior inmediato. Sería como conectar la salida a la entrada. Tiene una memoria muy rudimentaria que sólo me permite almacenar un 1, no tengo manera de volver a cero.

Para solucionar esto, se crean **circuitos secuenciales biestables (o flip flops)**. Es un nuevo dispositivo que tiene dos entradas y dos salidas.

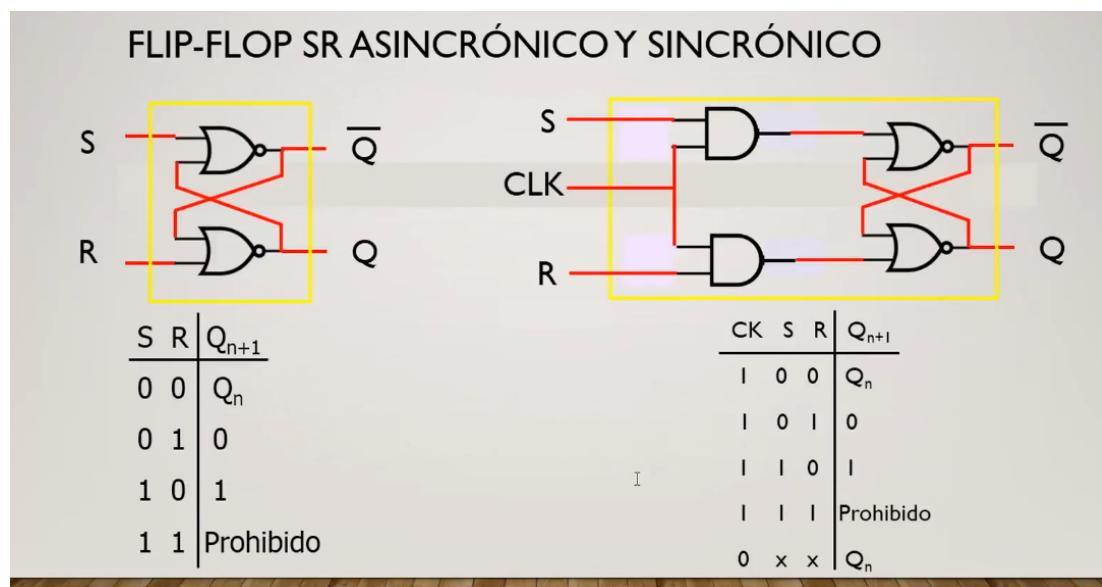
Los llamamos biestables porque **porque son estados estables, lo que ubico a la entrada queda indefinidamente estable hasta que yo lo haga cambiar**.

Tipos de flip flops

- SR - set, reset.
  - Tengo dos entradas —> Set, que pone el estado en 1 y Reset, que pone el estado en 0
  - Tengo dos salidas —> Q y Q negado, que son complementarias.
  - Q<sub>n+1</sub> será el estado a la salida después que yo puse las entradas.
  - Q<sub>n</sub> era el que estaba antes de las entradas.
  - Puede ser un flip flop asincrónico/sincrónico
  - Hay dos maneras de esquematizar este circuito, por medio de compuertas AND y por medio de NOR



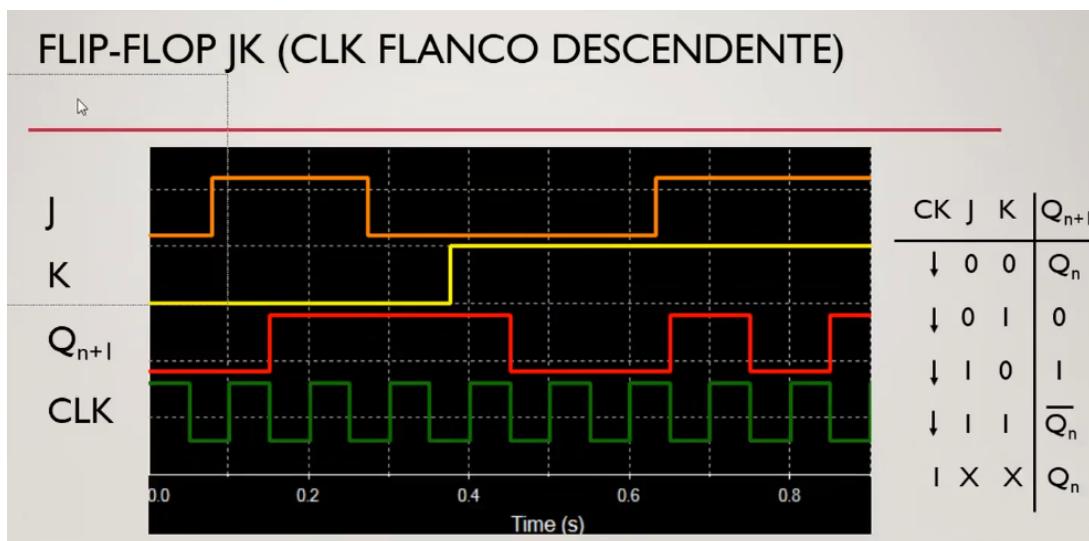
Los flip flop sincrónicos son los flip flops que incluyen un reloj (clock) que les marca el ritmo con la señal de habilitación, les dice cuándo pueden actualizarse los valores en caso de haber cambios. Los estados actuales van a seguir dependiendo de los valores de entradas, lo que el reloj marca es el momento en que va a leerse esa señal de entrada.



Notamos que la tabla de verdad en el SR sincrónico tiene un valor de salida llamado prohibido que es cuando todos los valores de entrada y el del clock están en 1.

Esto se soluciona con el flip flop sincrónico JK

- JK
  - Es igual que el SR sincrónico con la única variante que se soluciona el estado prohibido, y ahora el 11 no es prohibido sino que es el estado anterior negado



- Flip-Flop D ( $D = \text{data}$ )
  - Se basa en el SR Sincronico— pero va a tener una sola entrada y la segunda entrada va a ser la inversa de la anterior. Soluciona el estado prohibido
  - El FF tipo D es un FF con una sola entrada de datos.
  - Se puede pensar como un FF tipo S-R en el que la segunda entrada es la invertida de la primera.
  - La ventaja que tiene es que se requiere una sola señal para cambiar la salida (y por lo tanto para almacenar 1 bit).
  - Como se puede apreciar de la tabla de la verdad del FF D, dado que tiene una sola entrada, no es posible que ocurra una situación de estado prohibido.

#### ▼ Von Neumann / Estructura de una PC

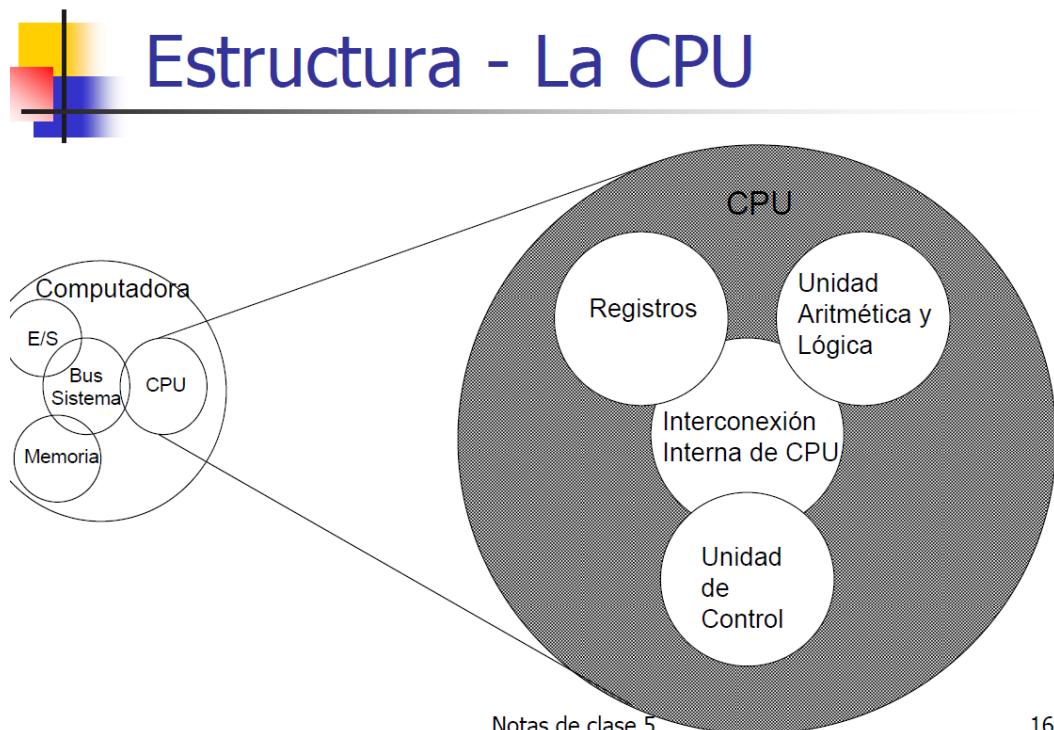
Una computadora es una máquina digital, sincrónica, que realiza cálculos numéricos y lógicos, controlada por un programa y con comunicación al exterior (por ejemplo a partir de una pantalla).

Cualquier sistema de cómputos tiene

- una unidad central de procesamiento
- memoria
- periféricos de entrada y salida —> para ingresar datos (ej teclado) para ver datos (ej monitor)
- sistema de interconexión para que trabajen estos tres componentes —> **Buses**: medio de comunicación compartido entre todos los dispositivos.

La CPU es lo que llamamos hoy en día microprocesador y se compone de:

- **Registros**: lugar de almacenamiento temporal. El objetivo es el mismo, almacenamiento de datos. pero temporal. (datos, instrucciones, lo que sea, todos los 1 y 0 que llegan a la cpu, los almacena en registros).
- **ALU**: la que hace las cuentas. trabaja en punto fijo. HOY EN DÍA tienen agregados unidades de procesamiento en punto flotante.
- **Unidad de Control**: es el verdadero computador dentro del computador, el corazón de la cpu. Es el que lleva la batuta. Es la que sabe lo que hay que hacer.

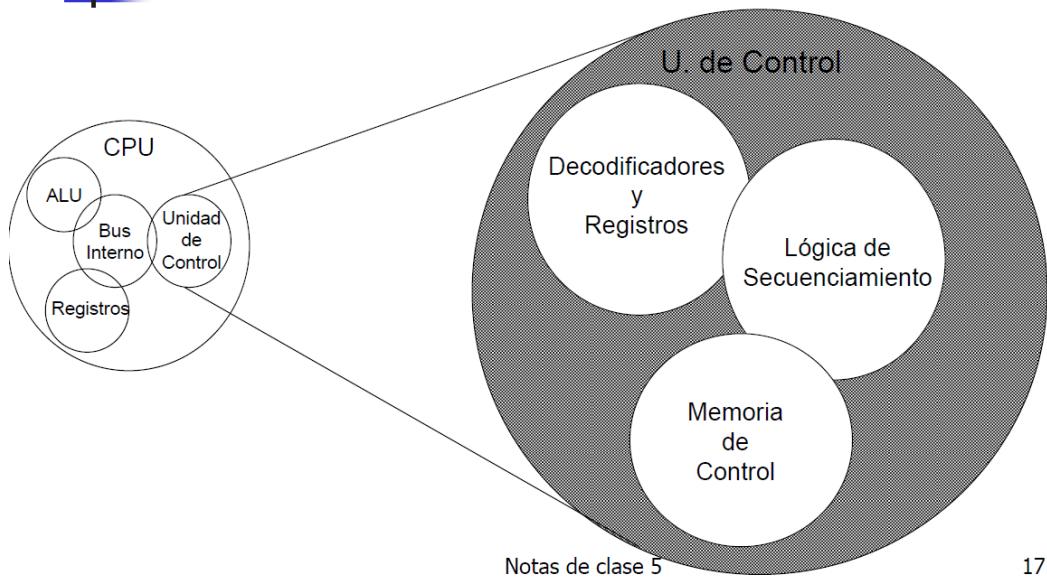


Notas de clase 5

16



# Estructura - Unidad de Control



## Von Neumann

Introdujo la idea de computadora con

- Unidad de control con una ALU que haga las operaciones
- Utilizar un sistema binario —> disminuye la probabilidad de fallos y simplifica las funciones
- Direccionamiento de memoria
- Programa almacenado: instrucciones y datos.

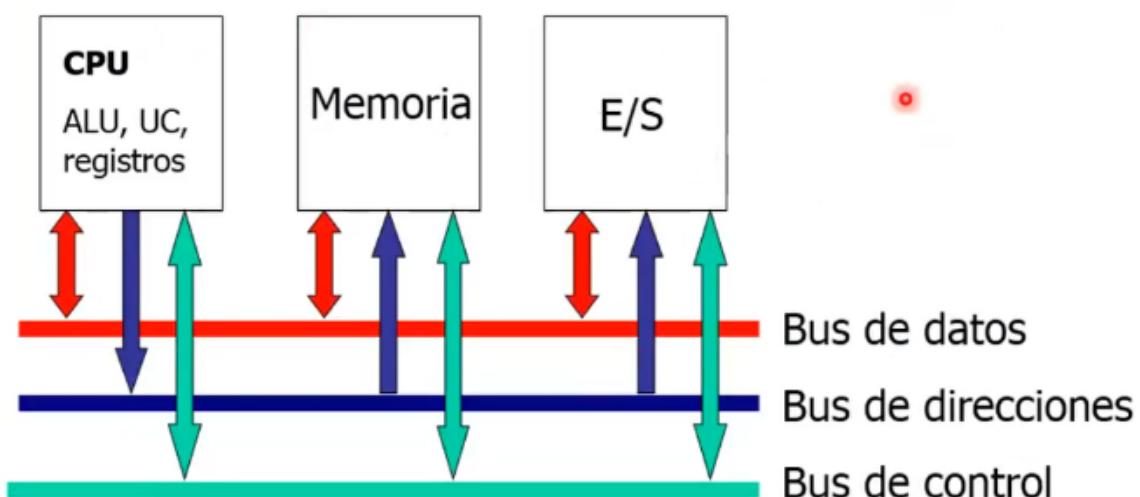
Propuso la **maquina de 3 direcciones**. LAS MAQUINAS DE HOY EN DIA SON DE DOS DIRECCIONES.

Antes de la aparición del microprocesador, donde todo esto lo tenemos dentro de la CPU, eran todos bloques individuales. La programación era mecánica: para cambiar un programa había que cambiar hardware.

El sistema de cómputos basado en arquitectura VN se compone de:

- Unidad de entrada: provee las instrucciones y los datos
- Unidad de memoria: donde se almacenan datos e instrucciones
- Unidad aritmético-lógica: procesa los datos
- Unidad de control: dirige la operación
- Unidad de salida: se envían los resultados

## Buses



Tengo 3 tipos de Buses

- **Bus de direcciones:** Como todos los dispositivos están conectados al mismo lugar, la CPU necesita del sistema de direcciones (conjunto de 1 y 0 que identifica a un dispositivo en particular) para identificar a cada dispositivo. TODOS los dispositivos que se conectan a la CPU, tienen direcciones. Como la CPU va a elegir con quién se comunica, el bus de direcciones lo maneja la CPU (a través de la UC , en definitiva). **NADIE va a escribir una dirección en el bus de direcciones que no sea la UC.**
- **Bus de datos:** Ahora me falta un mecanismo para **intercambiar información**. Para eso tengo el caminito rojo, similar en estructura al bus de

direcciones pero en este se llama **bus de datos**. Ahí viajan también las instrucciones, además de los datos, toda la información viaja por el bus de datos. Es bidireccional.

- **Bus de control:** Ahora me falta algún mecanismo que controle la operación y que diga de alguna manera si voy a leer, escribir, almacenar el dato en memoria. Eso lo dice la unidad de control a través de lo que llamamos **bus de control**. Son también unos y ceros pero en realidad son señales que controlan la operación.

Para ir a una dirección escribo en el bus de direcciones (azul) - identifica a cada dispositivo La información viaja por el bus de datos (rojo) - intercambia información (instrucciones + datos)

Las operaciones se controlan por el bus de control (verde)

Dispositivos de hardware:

- Entrada: Teclado / Mouse / Joystick
- Salida: Monitor / Impresora
- Para procesamiento: CPU / Memoria
- Para almacenamiento: Memoria / Discos / CD / DVD

Lo más importante para la materia son la cpu, la memoria y los buses, porque son los dispositivos que realmente mejoran la performance de la computadora.

#### ▼ Ciclo de instrucción

Si yo quiero ejecutar una instrucción, esa instrucción hay que traerla desde memoria y almacenarla adentro de la CPU . La CPU tiene que entender qué es lo que tiene que hacer con esa instrucción y luego cumplir con esa operación. Ejecutar un programa implica traer de memoria a las instrucciones una a una e interpretarlas.

La CPU se da cuenta de lo que tiene que hacer una instrucción a la vez. Las trae de memoria, las interpreta y toma las acciones de acuerdo a lo que interpretó de esa instrucción. **Todo pasa adentro de la CPU** → tengo que sumar 2 números, esos dos números tienen que viajar a la ALU, ahí se hace la cuenta y el resultado habrá que devolverlo o viajar otra vez hacia memoria.

A grandes rasgos se compone de dos etapas

- Búsqueda de la instrucción
  - Traer la instrucción desde memoria a la CPU a través del bus de datos.
  - Es común a todas las instrucciones
- Ejecución de la instrucción
  - Depende de la instrucción, las operaciones que requiera

1- Búsqueda de la instrucción en memoria: ¿Cómo sabe a dónde ir a buscarla? Hay un registro dentro de la CPU que se llama PC (program counter, contador de programa) que es el que guarda la dirección de memoria de la próxima instrucción a ejecutar (no tiene la instrucción en sí). Copia esa dirección en el MAR y de ahí al bus de direcciones. A través del bus de direcciones la va a buscar a esa dirección. Después incrementa el valor del PC para apuntar a la próxima instrucción a ejecutarse y sigue en secuencia hasta una instrucción que le diga que terminó el programa. Esta búsqueda es secuencial, instrucción por instrucción.

2- Llegada de la instrucción al registro IR: Una vez que el PC envía la dirección por el bus de direcciones y la CPU busca la instrucción en memoria, llega a través del bus de datos al MBR y luego se guarda en un **registro IR (registro de instrucciones)**. A este registro SÓLO viene instrucciones, nunca datos. Si hay algo ahí que no es una instrucción se nos cuelga la maquina, se nos pierde el programa. ¿Cómo sabe que es una instrucción? porque su dirección estaba en el PC (en el pc siempre está la dirección de memoria de la próxima instrucción a ejecutar)

3- Cuando la instrucción ya está en el registro IR, el próximo paso es la interpretación de la misma, acción que se lleva a cabo por la UC. La interpretación íntegra puede realizarse por **cada instrucción es autocontenida**, quiere decir que en ella está toda la información necesaria para su ejecución: qué operación es, qué operandos involucra, dónde se coloca el resultado, etc.

4- Luego de la interpretación se realizan las acciones requeridas/procedimientos. Por lo general son alguno de estas 4:

- Los datos van de CPU a Memoria

- CPU a E/S
- Procesamiento de datos: operaciones aritmético lógicas.
- Control: alterar la secuencia de ejecución de instrucciones.

En las acciones requeridas por ejemplo se va a buscar los datos que necesita la ALU, los operandos. El primer operando lo guarda en el RT1 (registro temporal 1) en la entrada de la ALU. ¿Como sabe a dónde ir a buscarlo? se lo dijo la instrucción porque es AUTOCONTENIDA. Va a buscar el segundo dato y lo lleva al siguiente registro temporal de la ALU. Hace la operación y lo guarda en otro registro temporal.

5- Direcciona a través del bus de direcciones una zona de memoria y a través del bus de datos almacena el resultado en memoria.

**Todo esto se repite instrucción por instrucción al compás de una señal de reloj.**

**MAR**(Memory address register) es intermediario entre el bus de direcciones y la CPU. Registro donde quedan almacenadas las direcciones.

**MBR** (memory buffer register) es intermediario entre el bus de datos y la CPU.

#### ▼ Formato de instrucción

Elementos de una instrucción de máquina

- **Código de operación:** Es un código binario que especifica la operación a realizar (ej suma). Los más comunes son: ADD, SUB, MOV, AND, OR, XOR, etc.
- **Referencia del operando fuente:** Establece dónde se encuentra el operando. La operación puede involucrar a más de un operando fuente (o de entrada)
- **Referencia del operando resultado:** establece dónde almacenar el resultado

*Los operando fuente/resultado pueden estar en memoria, registro de la CPU o dispositivos de E/S*

- **Referencia de la siguiente instrucción:** Le dice a la CPU dónde buscar la siguiente instrucción luego de ejecutarse la instrucción anterior. En la mayoría de los casos se ubica a continuación de la instrucción actual.

- ¿En qué casos no se daría eso de que la próxima instrucción se ubica a continuación de la actual? si se interpone alguna estructura de control o llamadas a un procedimiento que me haga saltar a otra instrucción.

El conjunto de instrucciones de máquina debe ser capaz de expresar cualquiera de las instrucciones de lenguajes de alto nivel.

Las instrucciones de máquina se dividen en:

- Procesamiento de datos: operaciones aritmético lógicas.
- Almacenamiento de datos: transferencias dentro del sistema
- Instrucciones de E/S: transferencia de datos entre la computadora y mecanismos externos
- Control: son las que nos van a dejar hacer en lenguaje de máquina lo equivalente a los if, while, for, es decir los jump o saltos.

¿Cuál es la información que necesita una instrucción?

Código de operación, dirección de los dos operandos, dirección del resultado y dirección de la próxima instrucción. Son 5 campos, 4 direcciones. El formato de la instrucción va a depender después del tipo de máquina que estemos utilizando.

Tipos de máquinas:

Clasifican a las instrucciones dependiendo el número de operandos que tiene(direcciones). Una de las formas tradicionales de describir la arquitectura de un procesador es en términos del número de direcciones contenidas en cada instrucción → *¿Cuál es el número máximo de direcciones que serían necesarias en una instrucción?*

1. De 4 direcciones:

- La instrucción especifica 4 direcciones
- Direcciones explícitas para operandos, resultado y próxima instrucción
- Cada campo tiene que tener los bits suficientes para acomodar la dirección completa
- 96 bits de referencia

Si tenemos instrucciones tan largas el programa tarda mucho en ejecutarse porque hay que leer todos los bits. En afán de acortarlo, aparecen las máquinas

de 3 direcciones. Acá es cuando apareció por primera vez el PC para acomodar la dirección de la próxima instrucción a ejecutar.

2. De 3 direcciones:

- No contiene la dir de la próxima instrucción, está almacenada en el PC.
- Todavía sigue siendo larga (72 bits de referencia)

3. De 2 direcciones:

- Se mueve el operador 1 a un registro temporal
- Reduce considerablemente el tamaño de la instrucción (48 bits de referencia)
- El resultado es almacenado en una de las direcciones de los operandos

4. De 1 dirección:

- Implementa el registro Acumulador (AC)
- Instrucciones para cargar y descargar el acumulador
- Un operando y resultado en lugar predefinido
- Instrucción más corta (24 bits de referencia)
- Tiene como desventaja que se necesita una instrucción para enviar el dato del acumulador a la memoria y viceversa

¿Cómo se diseñan las instrucciones?

El conjunto de instrucciones es el medio que tiene el programador para controlar a la CPU.

Para su diseño tiene en cuenta

- Tipos de operaciones
  - Transferencia de datos: MOV (load/store)
  - Artiméticas: add, sub, inc, etc
  - Lógicas: and, nor, not, etc
  - Conversión
  - E/S
  - Transferencia de control: salto, bifurcación

- Control del sistema. usada por OS
- Tipos de datos
- Formato de la instrucción
- Registros: cantidad que se pueden referenciar
- Direccionamiento: La manera de especificar la dirección de un operando o la próxima instrucción

## Modos de direccionamiento

Son las distintas maneras de explicar en la instrucción dónde se encuentran los operandos, dónde voy a guardar los datos, dónde los muevo. La fuente y el destino de alguna operación (los nombres que le dan INTEL).

Sus principales objetivos son

- Disminuir la cantidad de bits en una instrucción.

Por ejemplo: ADD AX BX —> ahí estamos diciendo que los datos están en los registros AX y BX, pero no decimos cuales son esos datos, sólo donde están . Como los registros son pocos, con 4 bits me alcanza para especificar los 16 registros y así disminuyo la cantidad de bits en una instrucción, si yo pongo una dirección eso ocupa mas bits.

- Manejo mas eficiente de los datos (a los arreglos, a las tablas, para recorrerlo)—> Me permite recorrer tablas o arreglos.
- La dirección puede que no se conozca hasta que el programa se ejecuta —>Tengo que tener alguna manera de decir desde dónde se cargó el programa, o que a partir de tal valor guardo las variables.

## Tipos de MDD

- **Inmediato**—> el operando se obtiene automáticamente de la memoria, al mismo tiempo que la instrucción. No requiere una referencia extra a memoria de datos. Se usa para definir constantes e inicializar variables. La desventaja es que el tamaño del operando está limitado por el tamaño del campo de direccionamiento.
- **Directo** —> El campo de dirección tiene la dirección efectiva del operando. Tiene espacio limitado de direcciones por la cantidad de

bits del campo. Se usa para acceder a variables globales cuya dirección se conoce en el momento de compilación.

- **Por registro** —> Igual al directo pero se especifica un registro en vez de una dirección de memoria, ya que la referencia a memoria ocupa menos bits que la dir. en sí y no requiere acceso a memoria de datos. La desventaja es que los registros son limitados.
- Indirecto por memoria —> En la instrucción está la dirección de la dirección del operando. Trata de solucionar el problema del directo: con una dirección de menos bits, se apunta a una de más bits.
- Indirecto por registro —> En la instrucción se especifica el registro que tiene almacenada la dirección. La ventaja es que usa menos bits para especificar el registro que la posición en memoria.
- Por desplazamiento—> Combina capacidades de directo e indirecto y requiere que la instrucción tenga dos campos de dirección, al menos uno tiene que ser explícito. Estos dos campos se suman para formar la dirección efectiva. Los más comunes son: Indexado, de registro base y relativo.
- Del stack—> arreglo lineal de direcciones de memoria. Es una pila donde el último en entrar es el primero en salir. Es una zona de memoria reservada. Asociado al stack hay un registro apuntador (SP) cuyo valor es la dirección del tope de la pila. Sólo utilizo las instrucciones PUSH Y POP, el SP se encarga de saber a dónde ir.

### Registros de uso general

Estos son los principales registros que como programadores podemos usar para cualquier cosa.

Tienen la forma INSTRUCCIÓN, DESTINO, FUENTE.

Destino y fuente son 2 operandos especificados por algunos de los mdd vistos. El de instrucción es un operando registro de la CPU.

- AX : acumulador, es el principal en las operaciones aritméticas
- BX : puntero base (dir de memoria)
- CX : contador, interviene en instrucciones de ciclo
- DX : datos, participa en multiplicación y división

## ▼ Memoria

La memoria del sistema es el lugar donde la computadora almacena los programas y datos actuales que están en uso. Hay varios niveles de memoria de computadora, incluyendo ROM, RAM, caché, página y gráficos, cada uno con objetivos específicos para el funcionamiento del sistema.

♦ **Ubicación:** Indica si la memoria es interna o externa a la computadora.

- La memoria interna:

- Suele identificarse con la **memoria principal**.

Sin embargo hay además otras formas de memoria interna.

- El **procesador** necesita su propia **memoria local en forma de registros**.
- La **unidad de control** del procesador también **puede necesitar su propia memoria interna**.
- La **memoria caché** es también otro tipo de memoria interna.

- La memoria externa:

- Consta de dispositivos periféricos de almacenamiento:

- **Discos y cintas**, que son accesibles por el procesador a través de controladores de E/S

### Algunos tipos de memoria interna

#### ♦RAM:

La más común es la denominada memoria de acceso aleatorio (**RAM**, Random-Access Memory)

Características:, las RAM pueden utilizarse solo como almacenamiento temporal.

- Es posible tanto leer datos como escribir rápidamente nuevos datos en ellas.
- La otra característica distintiva de una RAM es que es **volátil**
  - Una RAM debe estar siempre alimentada. Si se interrumpe la alimentación se pierden los datos.

Las dos formas tradicionales de RAM utilizadas en los computadores son la DRAM y la SRAM.

**RAM dinámica (DRAM): 2D** → LA PALABRA ESTÁ ORGANIZADA EN 1 CHIP

- Está hecha con celdas que almacenan los datos como **cargas eléctricas** en **CONDENSADORES**.
- La presencia o ausencia de carga en un condensador se interpretan como el uno o el cero binarios.
- A diferencia de los flip-flops, los condensadores tienen una tendencia natural a descargarse, las RAM dinámicas requieren refrescos periódicos para mantener memorizados los datos.
- El término dinámica hace referencia a esta tendencia a que la **carga almacenada se pierda**, incluso manteniéndola siempre alimentada.

**RAM estática (SRAM): 2 ½ D** → La palabra esta dispersa en chips distintos

- Es un dispositivo digital, basado en los mismos elementos que se usan en el procesador.
- En una RAM estática, los valores binarios se almacenan utilizando configuraciones **flip-flops** y mantiene la información mientras el circuito de memoria recibe alimentación eléctrica.
- Para implementar cada celda de memoria son necesarios varios **transistores**, por lo que la memoria tiene una **capacidad de**

**integración limitada** y su costo es elevado en relación con otros tipos de memoria RAM, como la DRAM;

- Es el tipo de memoria RAM más rápido.
- En 2 1/2 D al estar los bits dispersos en distintos chips hay menor probabilidad de error.
- En 2 1/2 D al usar decodificación separada de filas y columnas, reduce la complejidad de los decodificadores.

#### ♦ROM (ROM, Read-Only Memory)

Características:

- Una ROM es **no-volátil**; es decir, no se requiere fuente de alimentación para mantener memorizados los valores de los bits.
- Memoria sólo de **lectura**
- Contiene un patrón permanente de datos que **no puede alterarse**.
- Aunque es posible leer de una ROM, no se pueden escribir nuevos datos en ella

#### ♦Memoria Caché

El objetivo de la memoria caché es lograr que la velocidad de la memoria sea lo más rápida posible, consiguiendo al mismo tiempo un tamaño grande al precio de memorias semiconductoras menos costosas.

- La Memoria Caché se sitúa entre la memoria principal y el procesador
- Puede estar formada por uno o varios niveles.
- Tiene un menor tiempo de acceso que el de la memoria principal con el objetivo de reducir el tiempo de acceso medio a los datos
- Tiene una capacidad de almacenamiento más reducida que la memoria principal.

Funcionamiento:

- Si un dato que se requiere está en la memoria caché, se lo proporciona al procesador sin acceder a la memoria principal
- Sino, se lleva el dato de la memoria principal a la memoria caché y después se proporciona el dato al procesador.

### Algunos tipos de memoria externa:

Como la memoria principal de los computadores no tiene capacidad infinita y es volátil. Se requiere disponer de almacenamientos externos.

Mientras los dispositivos de **memoria primaria** permiten un acceso **inmediato** del programa a la información que contienen

Los dispositivos de almacenamiento secundario guardan la información en un soporte que no permite el acceso inmediato desde el programa y se requiere un paso **previo de lectura** (o entrada) que recupera dicha información desde el almacenamiento y lo coloca en la memoria.

Sacrificando la rapidez del acceso se obtienen capacidades muchísimo mayores a precios muy inferiores y con tiempos de respuesta soportables para cada tipo de aplicación.

Los soportes magnéticos son el medio más usual de almacenar la información en un sistema informático. Entre la variedad existente, podemos destacar: cinta, disco y disquete.

Las principales características de estos soportes son:

- Reutilizables (salvo en los más antiguos).
- Elevada capacidad de almacenamiento.
- No volátiles.
- Más económicos que la memoria principal (RAM).

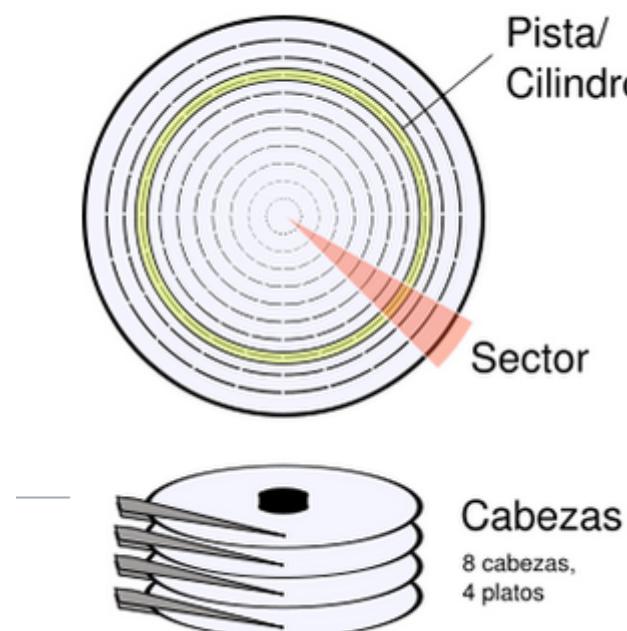
### **Disco magnético**

Son sistemas de almacenamiento de información que en la actualidad tienen una gran importancia, ya que constituyen el principal soporte utilizado como memoria masiva auxiliar.

La superficie del disco magnético se divide en las siguientes partes:

- Pista: Zona a la que accede el cabezal si este se queda fijo en una posición y el disco sigue girando. Si el cabezal se tratara de un lápiz, la pista sería la zona que el cabezal dibuja sobre el disco (que se trataría de una circunferencia). Hay que tener en cuenta que las pistas más cercanas al centro del disco son de menor tamaño al tener menor radio la circunferencia.

- Sector: Es una subdivisión de una pista (track) en un disco magnético. Cada sector almacena una cantidad fija de datos.
- Cilindro: Conjunto de pistas a las que el conjunto de cabezales puede acceder desde una posición. Un cilindro está compuesto por un conjunto de sectores.



- En el acceso, hay que considerar dos tiempos:
  - Tiempo de **búsqueda de la pista** (T seek).
  - Tiempo de **espera al sector** (T latencia).
- Tiempo de acceso será: **Tiempo de acceso = T seek + T latencia**

## Disco óptico

Dependiendo de como se refleja la luz es como se va a interpretar la información

### DISCOS COMPACTOS CD-ROM.

- Tanto el CD de audio como el CD-ROM (compact diskread-only memory, memoria de disco compacto de solo-lectura) comparten una tecnología similar.

## Múltiples unidades de discos (RAID)

Ya que el ritmo de mejora de prestaciones en memoria secundaria ha sido menor que en procesadores y en memoria principal, se desarrollaron conjuntos de discos que operan independientemente y en paralelo.

Puntos importantes:

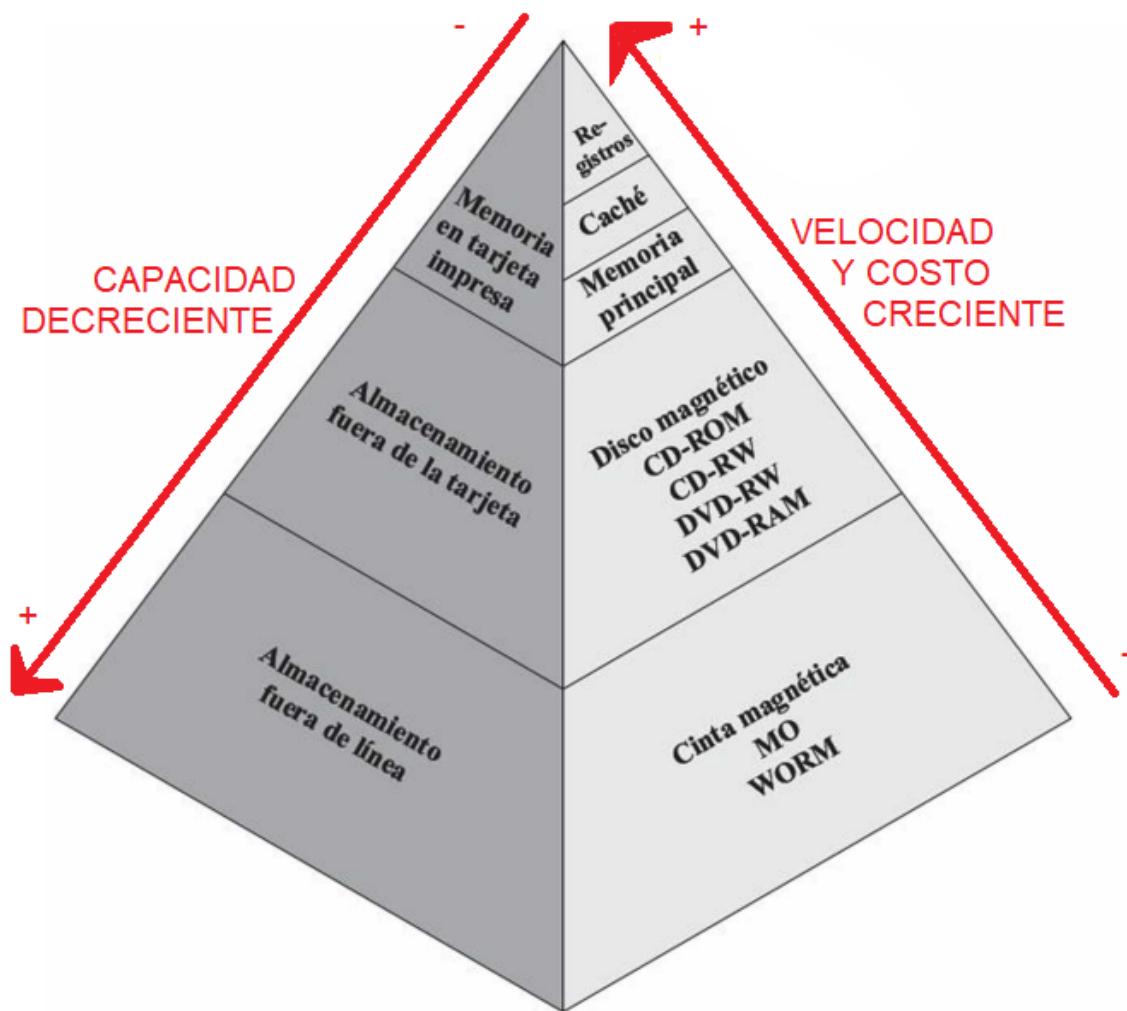
- Con varios discos, las **peticiones separadas de E/S se pueden gestionar en paralelo**, siempre que los datos requeridos residan en discos separados.
- Además, se puede ejecutar en paralelo una única petición de E/S si el bloque de datos al que se va a acceder está distribuido a lo largo de varios discos
- Con el uso de varios discos, hay una **amplia variedad de formas** en las que se pueden organizar los datos
- **Se puede añadir redundancia para mejorar la seguridad.**

La estrategia RAID

- Reemplaza una unidad de disco de gran capacidad por **unidades múltiples de menor capacidad**
- distribuye los datos de forma que se puedan **habilitar accesos simultáneos** a los datos de varias unidades mejorando las prestaciones de E/S
- permitiendo más fácilmente **aumentos en la capacidad.**

## Jerarquías de memoria

El objetivo en el diseño del sistema de memoria, es obtener gran capacidad y un tiempo de acceso reducido con el precio más bajo posible. Como no existe ninguna tecnología que cumple simultáneamente estos requisitos, la memoria de una computadora se estructura en varios niveles con el objetivo de conseguir mejores prestaciones, y forma lo que se denomina Jerarquía de Memoria.



Se trata de buscar un equilibrio entre la cantidad que ponemos de cada una y cómo las estructuramos entre sí para llegar a funcionamiento más eficiente posible a partir de un costo razonable de recursos.

Pirámide: La memoria más costosa y rápida son los registros, luego la caché, la memoria principal, luego los discos y luego los periféricos.

Todo esto funciona a partir de los principios de localidad que cumplen los programas.

- Principio de localidad espacial: si accedés a una posición de memoria, es muy probable que luego accedas a una cercana.
- El principio de localidad espacial me dice que conviene trabajar copiando de memoria principal a caché, copiando de bloques digamos, no de a una palabra por vez por ejemplo. Dado que vamos a tener en la memoria principal más cantidad de bloques que en la caché, **¿cuál es la metodología que usamos para decidir dónde poner los bloques desde la memoria principal a la caché? MAPEO??**

- Principio de localidad temporal: si accedés a una posición de memoria, es muy probable que en un corto periodo de tiempo, vuelvas a acceder a la misma, es por eso que esas posiciones que acabo de usar, las guardamos en la caché para que las vaya a buscar ahí y no a otra memoria que tarda más.

La jerarquía garantiza el funcionamiento en una velocidad "promedio"

Funciona ya que los programas cumplen con ciertos principios que nos lo permiten:

- Principio de localidad espacial de referencia: Cuando se accede a una palabra de memoria, es 'muy probable' que el próximo acceso sea en la **vecindad** de la palabra anterior.
- Principio de localidad temporal de referencia: Cuando se accede a una posición de memoria, es 'muy probable' que un lapso de 'tiempo corto', dicha posición de memoria sea **accedida nuevamente**.