

# Organización de computadoras

Preparándome para el final.

*"Todos los estudiantes de informática deben, en cierta medida, comprender y valorar los componentes funcionales de un computador, sus características, su funcionamiento y sus interacciones"*

## ▼ Clases

### ▼ Clase 1 y 2: Representación de datos

Las computadoras almacenan datos e instrucciones en memoria, y para ello utilizan el sistema binario.

Razones:

1. el dispositivo se encuentra en uno de dos estados posibles (0 o 1)
2. identificar el estado es más fácil si solo hay dos

### ▼ Tipos de datos

Las computadoras manejan 4 tipos básicos de datos binarios

- Números enteros con/sin signo
  1. Sin signo

La cantidad de representaciones depende del número de bits. Osea, si el número tiene N bits, puedo representar  $2^N$  números distintos. El rango va desde (0 a  $2^N - 1$ )

2. Módulo y signo

3. Complemento a uno (Ca1)

4. Complemento a dos (Ca2)

5. Exceso

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

#### ▼ Sistema Posicional

Sistema Posicional: es un sistema que si quiero formar un número nuevo sólo tengo que desplazar los dígitos, ejemplo: decimal.  
contraejemplo: romano.

La sumatoria siempre da un número en base 10.



Teorema Fundamental de la Numeración

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

$$... + x_4 \times B^4 + x_3 \times B^3 + x_2 \times B^2 + x_1 \times B^1 + x_0 \times B^0 + x_{-1} \times B^{-1} + x_{-2} \times B^{-2} + ...$$

$N^o$  es el valor decimal de una cantidad expresada en base B y con (n+1+m) dígitos en posiciones i.

#### ▼ Sistemas Decimal, Binario y Hexadecimal

- Sistema Decimal: base 10
- Sistema Binario: base 2
- Sistema Hexadecimal: base 16

#### ▼ Conversiones entre los sistemas de numeración

##### ▼ Decimal-Binario

### Conversión decimal-binario

El método de conversión de un número decimal a un número binario consiste en efectuar, sobre la parte entera del número decimal, divisiones sucesivas de los cocientes por el número 2, hasta que el cociente tome el valor 0. La unión de todos los restos obtenidos, escritos en orden inverso, nos proporciona ahora el número decimal inicial expresado en sistema binario.

Ejemplo. Convertir el número decimal 15 a binario.

$$\begin{array}{r} 15 \div 2 \\ 1 \text{ } 7 \div 2 \\ 1 \text{ } 3 \div 2 \\ 1 \text{ } 1 \div 2 \\ 1 \text{ } 0 \end{array}$$

Leyendo los restos, del último obtenido al primero de ellos, tenemos:  $1111_2 = 15_{10}$

### ▼ binario-hexa y viceversa

Dígito hexadecimal	Dígito binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

### ▼ cualquier base - decimal

#### Teorema Fundamental de la Numeración

### ▼ Números en punto fijo

Misma cantidad de dígitos y la coma en el mismo lugar.

Rango: diferencia entre el número mayor y el menor

### ▼ BSS

$(0 \text{ a } (2^N - 1))$

### ▼ BCS

$-(2^{(N-1)} - 1) \text{ a } +(2^{(N-1)} - 1)$

### ▼ Ca1

$$-(2^{(N-1)}-1) \text{ a } +(2^{(N-1)}-1)$$

Se obtiene

-Si empieza con 0, como BCS

-Si empieza con 1, proceso de BCS e invierto los bits

#### ▼ Ca2

$$-(2^N-1) \text{ a } +(2^{(N-1)}-1)$$

Se obtiene

-Si comienza con 0, es como BCS

-Si comienza con 1, luego de invertir sumo 1. O desde el primer bit de 0 todo igual hasta el primer 1, el cual copio pero después invierto

#### ▼ Exceso

$$-(2^N-1) \text{ a } +(2^{(N-1)}-1)$$

Se obtiene

Obtengo el exceso el cual es  $2^{(N-1)}$

-Si quiero pasar de decimal a binario se lo suma y luego lo paso a binario.

-Si quiero pasar de binario a decimal, paso el binario a decimal (con el teorema) y le resto el exceso.

Resolución: diferencia entre dos números consecutivos (en punto fijo es siempre la misma)

#### ▼ Representación y error

Al convertir un número decimal a sistema binario tendremos 2 casos.

Sin restricción de cantidad de bits a usar:

$$3,125 \text{ (base 10)} = 11,001 \text{ (base 2)}$$

Con restricción de cantidad de bits a usar, por ejemplo 3 en la parte entera y 4 en la fraccionaria:

$$3,125 \text{ (base 10)} = 011,0010 \text{ (base 2)}$$

Error: Hay números que no se pueden representar exactamente por lo que obtenemos el error más cercano al número que utilizamos.

Error relativo:  $EA(x)/X$

Error absoluto:  $X'-X$

máximo: Resolución de los dos números dividido por dos

▼ BCD (sistema decimal codificado en binario)

Representa cada dígito decimal por su equivalente en binario.



Dígito decimal	Código BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

▼ Desempaquetado sin signo

Completa los 8 bits con unos

ej: 834 = 1111000 1110011 1110100

= F8 F3 F4

▼ Desempaquetado con signo

C (base 16) = 1100 representa el signo +

D (base 16) = 1101 representa el signo -

Lo agrego en el último dígito y adelante

ej: +834 = 1111000 1110011 11000100

= F8 F3 C4

-834 = 1111000 1110011 11010100

= F8 F3 D4

▼ Empaquetado sin signo

Puede usar los 8 bits para dos números. Si sobra espacio, se completa con ceros.

ej: 834 = 00001000 00110100

= 08 34

▼ Empaquetado con signo

agrego el signo al final, y sigo completando con ceros adelante si es necesario.

ej +834 = 10000011 01001100  
= 83 4C

▼ Suma

Si el resultado de un número en BCD me da mayor a 9, tengo que sumarle 6

▼ Representación de datos alfanuméricos

Los caracteres en un sistema de cómputo se representan internamente como un conjunto de unos y ceros. El ASCII es el estándar.

- Falta operaciones aritméticas y flags (parece que no lo toman) y el nivel de lógica digital

▼ Clase 3: Números en punto flotante

La idea de punto flotante es almacenar números muy grandes y muy pequeños. Se dice que el punto o coma es flotante porque la posición de la coma depende del exponente. De esta manera, se almacenan muchos menos bits que si almaceno el número original.

Los números en punto flotante se almacenan en el sistema de cómputo de la siguiente manera:

$$\pm M \times B^{\pm E}$$

M = mantisa, B = base, E = exponente

La mantisa y el exponente pueden estar en cualquiera de los sistemas vistos (Ca1, Ca2, Exceso, BCS, BSS), la base va a ser siempre 2 y no se almacena pq estamos trabajando en un sistema de cómputo.

La mantisa puede ser fraccionaria y el exponente no.

▼ Su resolución varía a lo largo del intervalo del rango

Inferior negativo

Superior negativo

Inferior positivo

Superior positivo

#### ▼ Normalización

Todas las mantizas FRACCIONARIAS empiezan con 0,1. Si tiene signo es BCS y sino es BSS

#### ▼ Normalización con bit implícito

El 1 despues de la coma lo tengo en cuenta pero no lo escribo

#### ▼ Rango

- Numeros en BSS
  - Mínimo: MinMantisaPositiva y MinExponenteNegativo
  - MáximoPositivo: MaxMantisaPositiva y MaxExponentePositivo
- Números en BCS
  - Minimo negativo: MinMantisaNegativa y MaxExponentePositivo
  - Máximo positivo: MaxMantisaPositiva y MaxExponentePositivo

#### ▼ Representación de números normalizados

1. Se escribe el número en binario con la mantisa.
2. Corro la coma para que quede normalizado actualizando el exponente.
3. Paso el exponente a binario en el sistema que me dieron.

#### ▼ Errores

- ERROR ABSOLUTO: La diferencia entre el número que quiero representar y el número más próximo en el sistema.
- ERROR RELATIVO:  $EA / \text{NUMERO QUE QUISE REPRESENTAR}$
- ERROR ABSOLUTO MÁXIMO: Resolución / 2.
  - Resolución:  $2^a$  a la posición del último dígito de la mantisa.

#### ▼ Estándar IEEE 754

El IEEE 754 es un estándar de aritmética en coma flotante. Este estándar especifica como deben representarse los números en coma flotante con simple precisión (32 bits) o doble precisión (64 bits), y también cómo deben realizarse las operaciones aritméticas con ellos.

Emplea mantisa fraccionaria, normalizada y en representación signo magnitud (M y S), sin almacenar el primer dígito, que es igual a 1. El exponente se representa en exceso, que en este caso no se toma como  $2^{(n-1)}$ , sino como  $2^{(n-1)} - 1$

#### ▼ Simple precisión

El estándar IEEE-754 para la representación en simple precisión de números en coma flotante exige una cadena de 32 bits. El primer bit es el bit de signo (S), los siguientes 8 son los bits del exponente (E) y los restantes 23 son la mantisa (M):

La mantisa es fraccionaria normalizada, con la coma después del primer bit que es siempre uno (1,), en M y S.

El exponente se representa en exceso.

#### ▼ Tabla machete

Signo	Exponente	Mantisa	Valor
1	$0 < E < 255$	Cualquiera	$-1,M * 2^{E-127}$ donde M es la mantisa
0	$0 < E < 255$	Cualquiera	$1,M * 2^{E-127}$ donde M es la mantisa
Cualq.	255	No nulo	NaN ("Not a number"). Se aplica para señalar varias condiciones de error.
1	255	0	-Infinito
0	255	0	Infinito
1	0	No nulo	$-0,M * 2^{-126}$ (Números sin normalizar)
0	0	No nulo	$0,M * 2^{-126}$ (Números sin normalizar)
1	0	0	-0
0	0	0	0

#### ▼ Representación



## Punto Flotante

*Ejemplo de representación en Simple Precisión*

Representar el número 5,6875 en el estándar IEEE 754:

1. Representamos en BSS  $\Rightarrow 5,6875 = 101,1011 \cdot 2^{-0}$
2. Movemos la coma para que quede normalizada en 1,  $\Rightarrow 1,011011 \cdot 2^{-2}$
3. Represento mi Exponente (2) en Exceso  $2^{(n-1)} - 1$ 
  - a. Exceso =  $2 \cdot (8 - 1) - 1 = 127$
  - b. Sumo el Exceso y represento  $\Rightarrow 2 + 127 = 129 = 10000001$

Resultado  $\Rightarrow 0 \ 10000001 \ 011011000000000000000000$

### ▼ Doble precisión

El estándar IEEE-754 para la representación en doble precisión de números en coma flotante exige una cadena de 64 bits. El primer bit es el bit de signo (S), los siguientes 11 son los bits del exponente (E) y los restantes 52 son la mantisa (M).

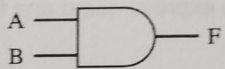
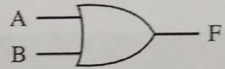
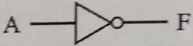
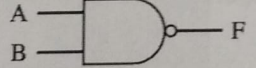
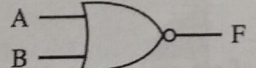
### ▼ Tabla machete

Signo	Exponente	Mantisa	Valor
1	$0 < E < 2047$	Cualquiera	$-1, M \cdot 2^{E-1023}$ donde M es la mantisa
0	$0 < E < 2047$	Cualquiera	$1, M \cdot 2^{E-1023}$ donde M es la mantisa
Cualq.	2047	No nulo	NaN ("Not a number"). Se aplica para señalar varias condiciones de error.
1	2047	0	-Infinito
0	2047	0	Infinito
1	0	No nulo	$-0, M \cdot 2^{-1022}$ (Números sin normalizar)
0	0	No nulo	$0, M \cdot 2^{-1022}$ (Números sin normalizar)
1	0	0	-0
0	0	0	0

### ▼ Clase 4: Circuitos lógicos

#### ▼ Puerta lógica

Una puerta lógica es un circuito electrónico que produce como señal de salida una operación booleana sencilla de las señales de entrada. Las puertas básicas usadas en lógica digital son AND, OR, NOT, NAND y NOR. Cada puerta se define de tres formas: símbolo gráfico, notación algebraica y tabla de verdad.

Nombre	Símbolo gráfico	Función algebraica	Tabla verdad															
AND		$F = A \cdot B$ or $F = AB$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \overline{A}$ or $F = A'$	<table><tr><th>A</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
NAND		$F = \overline{(AB)}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{(A + B)}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																

#### ▼ Circuito lógico combinacional

Un circuito lógico combinatorio o combinacional es un conjunto de puertas interconectadas cuya salida, en un momento dado, es función solamente de la entrada en ese instante. Como ocurre con una puerta sencilla, la aparición de la entrada viene seguida casi inmediatamente por la aparición de la salida, con solo retardos de puerta.

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

En general, un circuito combinacional consiste en  $n$  entradas binarias y  $m$  salidas binarias. Un circuito combinacional puede definirse de tres formas:

1. Tabla de verdad: para cada una de las posibles combinaciones de las  $n$  señales de entrada, se enumera el valor binario de cada

una de las  $m$  señales de salida.

2. Símbolo gráfico: Describe la organización de las interconexiones entre puertas.
3. Ecuaciones booleanas: Cada señal de salida se expresa como una función booleana de las señales de entrada.

#### ▼ Circuito lógico secuencial

Circuito lógico cuya salida depende de los valores actuales y pasados de las señales de entrada.

##### ▼ Está formado por:

1. Señales de entrada y salida
2. Señal de reloj (clock)
3. Lógica combinacional
4. Almacenamiento

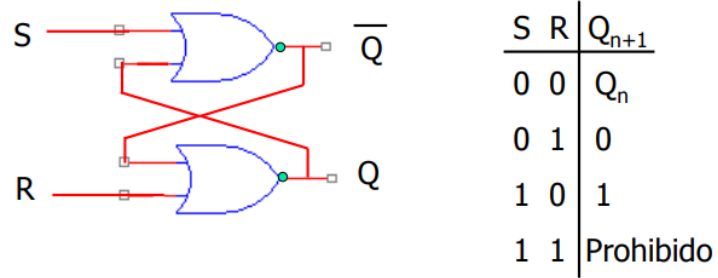
#### ▼ Circuitos Secuenciales BIESTABLES (Flip-Flop)

La forma más sencilla de un circuito secuencial es un biestable. Hay varios tipos de biestables y todos ellos comparten dos propiedades

- El biestable es un dispositivo con dos estados. Está en uno de dos estados, en ausencia de entrada, recordando el último estado. Entonces el biestable puede funcionar como memoria de un bit.
- El biestable tiene dos salidas, que son siempre complementarias. Normalmente se denominan  $Q$  y  $\sim Q$  (negado)

##### ▼ S-R

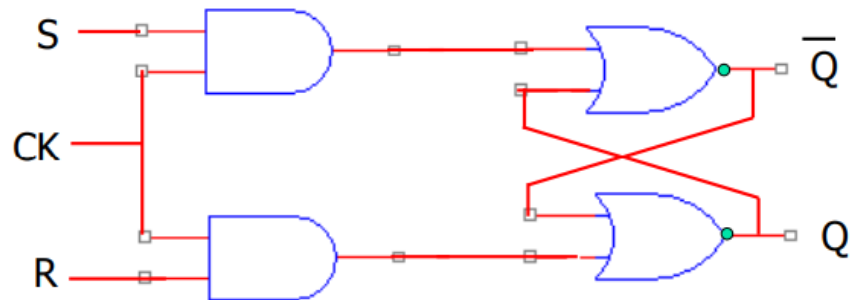
Tiene dos entradas,  $S$  (Set) y  $R$  (Reset), y dos salidas,  $Q$  y  $\sim Q$ , y consiste en dos puertas NOR conectadas por realimentación



### ▼ S-R Sincrónico

Reloj: Es una señal de tiempo precisa que determina cuando se producen los eventos.

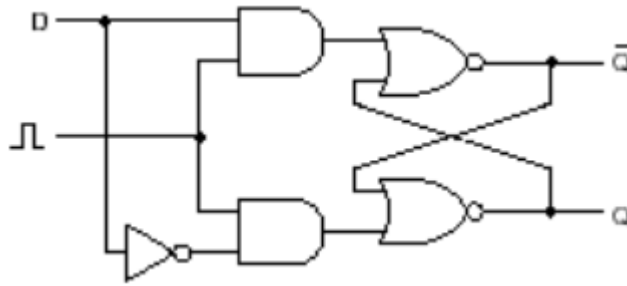
S y R son las entradas que tendrán efecto cuando CK tome valor 1



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

### ▼ D

Un problema con los S-R es que la condición R 1, S 1 debe ser evitada. Una manera de hacerlo es permitir solo una entrada. El biestable D lo cumple, usando un inversor.



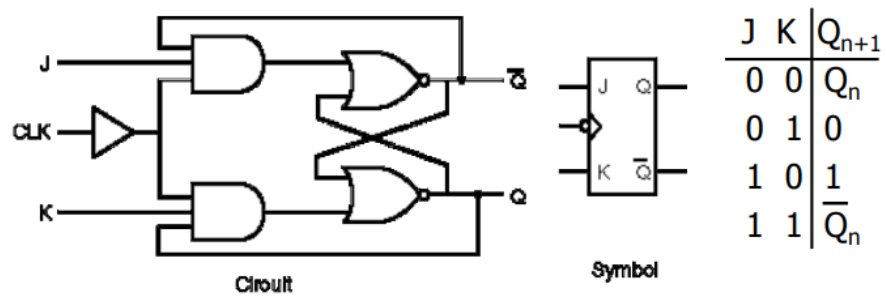
Elimina 2 estados del SR

D	$Q_{n+1}$
0	0
1	1

con CK=1

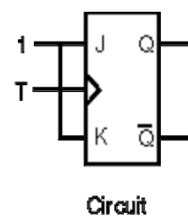
#### ▼ J-K

Todas las combinaciones posibles de los valores de entrada son válidos



#### ▼ T

T = Toggle. Invierte su estado cada vez que CK=1



#### ▼ Construir circuitos solo con NAND y NOR

- Ley de morgan, y ley de involución

- Para reemplazar un not NOT = NAND Y NOR con la misma entrada

#### ▼ Clase 5: Modelo Von Neumann

El matemático John Von Neumann propuso la idea del programa-almacenado en memoria.

##### ▼ Este modelo consta de 5 componentes principales

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

##### ▼ Aspectos más importantes

- Utilización del sistema binario
- Instrucciones y datos residen en memoria
- La memoria es direccionable por localidad sin importar el dato almacenado

##### ▼ Interconexión de un sistema de cómputo

Un sistema de cómputo está constituido por 3 subsistemas (CPU, Memoria, E/S)

##### ▼ Se debe poder comunicar entre sí

Un bus es un medio de comunicación compartido, constituido por líneas (camino de comunicación) capaces de transmitir señales binarias.

##### ▼ Bus de datos

Transporta datos e instrucciones. Número de líneas = número de bits que se pueden transferir al mismo tiempo (8,16,32,64)

##### ▼ Bus de direcciones

- Identifica el origen o el destino de los datos

- El ancho del bus determina la capacidad máxima de memoria del sistema

#### ▼ Bus de control

- Información de control y temporizado
  - Señales de lectura/escritura de memoria y E/S
  - Señales de selección o habilitación
  - Señales de reloj (clock)
  - Señales de pedido de interrupción

#### ▼ Clase 6: Ciclo de instrucción

La función básica que realiza un computador es la ejecución de un programa, constituido por un conjunto de instrucciones almacenadas en memoria. El procesador (CPU) es el encargado de ejecutar estas instrucciones especificadas en el programa. El procesador lee (busca) la instrucción en memoria, y la ejecuta. La ejecución del programa consiste en la repetición del proceso de búsqueda de instrucción y ejecución de instrucción.

El procesamiento que requiere una instrucción se denomina ciclo de instrucción, este tiene dos pasos: ciclo de búsqueda y ciclo de ejecución.

La ejecución del programa se interrumpe sólo si la máquina se apaga, error, o instrucción de interrupción.

#### ▼ Los ciclos de búsqueda y ejecución

Al comienzo de cada instrucción, la CPU busca una instrucción en memoria. Para seguir la pista de la instrucción que debe captarse a continuación, se utiliza un registro llamado contador de programa (PC, Program Counter). A no ser que se indique otra cosa, la CPU siempre incrementa el PC después de captar cada instrucción, de forma que captará la siguiente instrucción de la secuencia (la instrucción situada en la siguiente dirección de memoria).

La instrucción buscada se almacena en un registro de la CPU conocido como registro de instrucción (IR, Instruction Register). La instrucción se escribe utilizando un código binario que especifica la

acción de debe realiza la CPU. La CPU interpreta la instrucción y lleva a cabo la acción requerida.

En general, la acción puede ser de cuatro tipos:

- Procesador-Memoria: deben transferirse datos desde la CPU a la memoria, o desde la memoria a la CPU.
- Procesador-E/S: deben transferirse los datos a o desde el exterior mediante transferencias entre la CPU y un módulo de E/S.
- Procesamiento de datos: la CPU ha de realizar alguna operación aritmética o lógica con los datos.
- Control: alterar la secuencia de ejecución de instrucciones.

▼ Pasos uno por uno

PREGUNTAR SI HAY QUE NUMERARLOS EN EL FINAL

▼ Operación aritmética

1. **Cálculo dirección instrucción**: determina la dirección de la siguiente instrucción a ejecutarse.
2. **Búsqueda instrucción**: lee la instrucción de su posición de memoria a la cpu.
3. **Decodificación de la instrucción**: analiza la instrucción para determinar el tipo de operación a realizar y los operandos que se usarán.
4. **Cálculo de dirección los operandos**: si la operación implica la referencia a operandos en la memoria o e/s, entonces se determina la dirección.
5. **Búsqueda de los operandos**: busca el operando en la memoria o e/s.
6. **Operación sobre los datos**: ejecuta la instrucción.
7. **Cálculo de la dirección resultado**.
8. **Almaceno resultado**.
9. **Actualización de banderas y del contador de programa**.

▼ Salto incondicional



1. **Cálculo dirección instrucción:** determina la dirección de la siguiente instrucción a ejecutarse.
2. **Búsqueda de instrucción:** lee la instrucción de su posición de memoria a la cpu.
3. **Decodificación de instrucción:** La unidad de control decodifica la instrucción obtenida para identificar que se trata de un salto incondicional y determina la dirección de destino.
4. **Calculo de la dirección de salto:** La dirección a la que se debe saltar es calculada. Esta puede estar en la instrucción misma (dirección absoluta) o puede ser relativa al contador de programa (PC).
5. **Actualización del contador de programa:** La unidad de control carga el valor de la dirección de salto calculada en el contador de programa (PC). Esto indica la nueva dirección desde la que se obtendrá la próxima instrucción.
6. **Ejecución del salto:** Debido a que el salto es incondicional, el procesador no evalúa ninguna condición y simplemente continúa la ejecución desde la dirección de salto, pasando a buscar la siguiente instrucción a partir de la nueva posición del contador de programa.

#### ▼ Clase 7: Formatos y MDD

##### ▼ Elementos de una instrucción de máquina

- Código de operación
  - especifica la operación a realizar (ej. suma)
  - es un código binario
- Referencia a operandos fuente
  - establece dónde se encuentran los datos con los que va a trabajar el código de operación
- Referencia del operando resultado
  - establece dónde almacenar el resultado
- Referencia a la siguiente instrucción

- dice al procesador de dónde captar la siguiente instrucción

▼ Los operandos fuente y resultado pueden estar en tres lugares

1. Memoria
2. Registro de la CPU
3. Dispositivo de E/S

▼ Formato de instrucción

- Dentro de la computadora cada instrucción está representada mediante una secuencia de bits.
- La secuencia se divide en campos en correspondencia a los elementos que la componen.
- Este esquema se conoce como formato de instrucción. Osea, un formato de instrucción define como está organizada la instrucción.
- Un formato de instrucción debe incluir un código de operación e, implícita o explícitamente, cero o más operandos.
- El diseño de formato de instrucciones debe considerar la longitud de las instrucciones, fija o variables, los números de bits asignados al código de operación y a cada referencia a operando, y la forma en que se determina el modo de direccionamiento.

▼ Tipos de instrucciones

Cualquier programa escrito en alto nivel debe traducirse a lenguaje máquina para ser ejecutado. Por lo tanto, el repertorio de instrucciones de máquina debe ser suficientemente amplio como para expresar cualquiera de las instrucciones de un lenguaje de alto nivel. Teniendo esto presente, los tipos de instrucciones se pueden clasificar de la siguiente manera:

- De procesamiento de datos: instrucciones aritméticas y lógicas.
- De almacenamiento de datos: instrucciones de memoria.
- De transferencia de datos: instrucciones de E/S.
- De control: instrucciones de comprobación y bifurcación.

▼ Máquinas de N direcciones

Una de las formas de describir la arquitectura de un procesador es en términos del número de direcciones contenidas en cada instrucción.

▼ Máquina de 4 direcciones

- la instrucción contiene direcciones explícitas para operandos, resultado y próxima instrucción.
- Son "raras" (96 bits de referencias)

▼ Máquina de 3 direcciones

- La instrucción contiene la dirección de los operandos y del resultado
- La dirección de la próxima instrucción se guarda en un registro del procesador que se llama PC
- Ejemplo de instrucción: ADD A, B, C. Se guarda en A la suma de B y C.
- Instrucción larga (72 bits referencias)

▼ Máquina de 2 direcciones

- La instrucción contiene la dirección de los operandos. El resultado de la operación se guarda en uno de los operandos.
- Hay que mover el op1 a un registro temporal
- Menos elección donde guardar el resultado
- Ejemplo de instrucción: ADD A,B. Se guarda en A la suma de A y B
- Reduce el tamaño de la instrucción (48 bits referencias)

▼ Máquina de 1 dirección

- Un operando y resultado en un lugar predefinido
- Usan un registro especial (acumulador)
- Instrucciones para cargar y descargar el acumulador
- Instrucción más corta (24 bits de referencias)
- Necesitan las instrucciones especiales LOAD y STORE

#### ▼ Diseño del conjunto de instrucciones

- El conjunto de instrucciones es el medio que tiene el programador para controlar la CPU.
- Hay que tener en cuenta:
  - Tipos de operaciones
    - cuántas y cuáles
  - Tipos de datos
    - cuáles
  - Formato de instrucciones
    - longitud (bits), N de direcciones, tamaño de cada campo
  - Registros
    - cantidad que se pueden referenciar mediante instrucciones y su uso
  - Direccionamiento
    - la manera de especificar la dirección de un operando o una instrucción (la próxima)

#### ▼ Tipos de operaciones

- Transfencia de datos: mov, load, store
- Aritméticas: add, sub, inc, dec mul
- Lógicas: and, or, xor, not
- Conversión
- E/S: in/out
- Transferencia de control: salto, bifurcación
- Control de sistema: usadas por el s/o

#### ▼ Tipos de datos

- Direcciones
- Números: enteros, punto fijo, punto flotante
- Carácteres: ASCII, BCD

- Datos lógicos

#### ▼ Modos de direccionamiento

El modo de direccionamiento de un operando se refiere a la forma en que una instrucción accede a los datos que necesita para operar. Define cómo se especifica la ubicación del operando dentro de la instrucción, y dónde guardar el resultado.

Los MDD tienen como objetivo:

- disminuir la cantidad de bits de la instrucción.
- manejo más eficiente de datos (arreglos).
- la dirección puede que no se conozca hasta el momento de ejecutar el programa.

#### ▼ 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 utiliza para definir constantes y para inicializar variables.
- Desventaja: tamaño del operando limitado por el tamaño del campo de direccionamiento.

#### ▼ Directo

- El campo de dirección tiene la dirección efectiva del operando.
- Espacio limitado de direcciones por cantidad de bits del campo.
- Uso: acceder a variables globales.

#### ▼ Indirecto

- En la instrucción está la dirección de memoria que contiene la dirección del operando.
- Ventaja: espacio de direccionamiento mayor.
- Desventaja: múltiples accesos a memoria

#### ▼ Por registro

- Es igual directo, pero se especifica un registro en lugar de una posición de memoria.
- Menos bits.
- No requiere acceso a memoria de datos.
- Desventaja: los registros no son muchos.

#### ▼ Indirecto por registro

- En la instrucción se especifica el registro que tiene almacenada la dirección.
- Tiene un espacio de direccionamiento grande.
- Accede una vez menos a memoria que el indirecto.
- La dirección se llama apuntador.

#### ▼ Por desplazamiento

- Combina capacidades mdd indirecto y directo
- Requiere que la instrucción tenga dos campos de dirección (se suman para tener la dirección efectiva)
- Los más comunes son:
  - Relativo: el registro referenciado de manera implícita es el PC.
  - De registro base: el registro referenciado contiene una dirección de memoria y el campo de dirección tiene un desplazamiento.
  - Indexado: se direcciona con un registro + un desplazamiento, sirve para operaciones iterativas.

#### ▼ Pila

- Arreglo lineal de localidades de memoria.
- Zona de memoria reservada.
- Last in, first out.
- Asociado con la pila hay un registro puntero de pila.

### ▼ Clase 8: Registros e instrucciones

#### ▼ Organización de registros

## ▼ Tipos

### 1. Visibles al usuario

- Utilizados por el programador
- Pueden ser asignados a una variedad de funciones
- se pueden utilizar para direccionamiento (ej. indirecto de registro)
- sólo para datos ó sólo para direcciones
- los registros de dirección pueden ser asignados para un mdd (ej. reg índice para direccionamiento autoindexado)

### 2. De control y estado

- Utilizados por la UC para controlar la operación de la CPU
- No son visibles para el programador
- Los 4 esenciales
  1. PC: contador de programa
  2. IR: registro de instrucción
  3. MAR: de dirección de memoria
  4. MBR: buffer de memoria

## ▼ Número de registros

1. Afecta tamaño de la instrucción
2. Más bits para especificar la instrucción
3. Más referencias a memoria
4. Número óptimo: 8-32 registros

## ▼ Longitud de registros

- De direccionamiento: deben ser capaces de almacenar la dirección más grande.
- De datos: deben estar habilitados para almacenar la mayoría de los tipos de datos.

## ▼ Bits de condición

- Bits establecidos por la CPU como resultado de operaciones.
- Pueden ser utilizados por las instrucciones de bifurcación condicional.
- Generalmente no son alterados por el programador.

#### ▼ Para CPU PII intel

- De uso general  
AX : acumulador  
BX : puntero base  
CX : contador  
DX : datos
- Cada uno con su parte alta y baja
- Cualquier registro de uso general puede contener el operando para cualquier código de operación, sin embargo puede haber restricciones. Puede haber registros para operaciones en coma flotante y para operaciones con la pila
- En algunos casos pueden utilizarse para funciones de direccionamiento, por ejemplo, en direccionamientos indirectos por registro o con desplazamiento. En otros hay una separación parcial o total entre los registros de datos y direcciones

#### ▼ Assembler

- De bajo nivel
- Arquitectura Von Neumann
- Máquinas de 1,2,3 direcciones
- Tamaño de programa en memoria = código de operación + operandos
- Cantidad de accesos a memoria = instrucciones + operandos
- Variables pueden ser de 1 o 2 bytes
- Registros AX, BX, CX, DX, cada uno con su parte alta H y baja L
- Memoria celdas de 2 bytes



- 4 modos de direccionamiento: directo, indirecto, directo por registro, indirecto por registro

#### ▼ Operaciones

1. ADD O ADC
2. SUB O SBB
3. INC
4. DEC
5. CMP
6. AND
7. OR
8. XOR
9. NOT
10. NEG

#### ▼ Saltos

JMP etiqueta = Salta incondicionalmente

JS = Salta si es negativo

JNS = Salta si es positivo

JZ = Salta si es cero

JNZ = Salta si no es cero

JC = Salta si hay carry

JNC = Salta si no hay carry

JO = Salta si hay overflow

JNO = Salta si no hay overflow

#### ▼ Pilas

- Sector de la memoria, last in , first out
- Requiere registro puntero: SP
- Arranca en 8000h, pero no se utiliza
- Instrucciones: PUSH o POP

#### ▼ PUSH

1. DEC SP
2. Inserta parte alta del dato
3. DEC SP
4. Inserta parte baja del dato

▼ POP

1. Inserta parte baja del dato
2. INC SP
3. Inserta parte alta del dato
4. INC SP

▼ Subrutinas

- Se declaran en 3000h
- Se llaman con CALL y se vuelve con RET
- Al hacer call se hace IMPLICITAMENTE un push IP
- Por lo tanto, si hago PUSH en una subrutina tengo que hacer si o si un POP en la misma.

▼ Clase 9: Subsistema de memoria

▼ Jerarquía de memoria

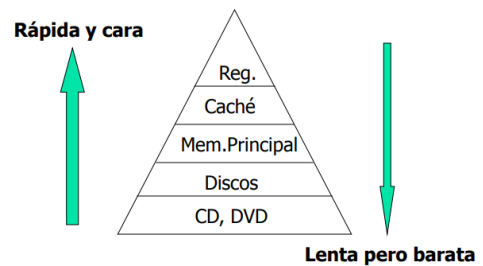
La forma en la que se organizan los distintos tipos de memoria es lo que se conoce como jerarquía de memoria.

En la cima de la jerarquía están los registros, seguidos por la memoria caché y luego la memoria principal. En la base, están las memorias secundarias (discos magnéticos) y de almacenamiento "off-line" (CD,DVD,cintas).

Hay tres características clave de la memoria: costo por bit, capacidad y tiempo de acceso:

- A menor tiempo de acceso, mayor costo por bit
- A mayor capacidad, menor costo por bit
- A mayor capacidad, mayor tiempo de acceso

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



## ▼ Principios

La memoria caché es una memoria pequeña y rápida que se encuentra entre la CPU y la memoria principal. El uso de la memoria caché (y la jerarquía de memoria en general) se sustenta en dos principios o propiedades que exhiben los programas:

- Principio de localidad espacial de referencia: cuando se accede a una palabra de memoria, es muy probable que próximo acceso sea en la vecindad de la palabra anterior.
  - Se sustenta en:
    - Ejecución secuencial del código.
    - Tendencia de los programadores a hacer próximas entre sí las variables relacionadas.
    - Acceso a estructuras tipo matriz o pila.
- Principio de localidad temporal de referencia: cuando se accede a una posición de memoria, es muy probable que en un lapso de tiempo corto, dicha posición de memoria sea accedida nuevamente.
  - Se sustenta en:
    - Formación de ciclos o bucles.
    - Subrutinas.
    - Pilas.

## ▼ Métodos de acceso

- Acceso aleatorio (RAM): el tiempo para acceder a una locación dada es independiente de la secuencia de accesos anteriores y es constante. Ejemplo: la memoria principal.

- Acceso secuencial: el acceso debe hacerse en una secuencia lineal específica. Ejemplo las unidades de cinta.
- Acceso directo: discos magneticos.
- Acceso asociativo: memoria caché

#### ▼ Organización

El elemento básico de una memoria de semiconductor es la celda de memoria.

##### ▼ Las celdas comparten 3 propiedades.

1. Dos estados posibles (1 y 0)
2. Se puede escribir en ellas
3. Se pueden leer para conocer el estado

##### ▼ Las celdas tiene 3 terminales funcionales capaces de llevar una señal eléctrica.

1. Selección: selecciona una celda de memoria
2. Control: especifica lectura o escritura
3. escritura/lectura de datos

##### ▼ Organización del chip

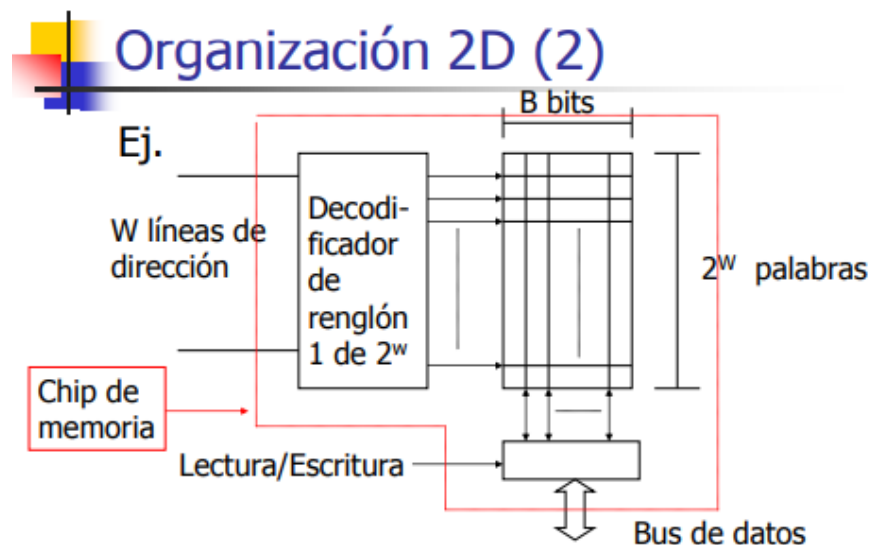
- Cada chip contiene un arreglo de celdas de memoria
- En las memorias de semiconductor hay dos organizaciones 2D y 2½D

##### ▼ ORGANIZACIÓN 2D (SRAM)

- La organización 2D de memoria semiconductora no requiere "refresco" cuando se usa en memorias **SRAM (Static Random Access Memory)** porque almacena datos de forma estática en lugar de dinámica.
- A diferencia de la DRAM, una SRAM es un dispositivo digital basado en los mismos elementos que se usan en el procesador. Los valores binarios se almacenan utilizando configuraciones de puertas que forman biestables (flip-flops). Una RAM retendrá sus datos en tanto se mantenga alimentada.

Cada bit de información en una celda de SRAM está almacenado en un circuito estable compuesto por transistores, que pueden mantener su estado (0 o 1) indefinidamente mientras haya energía suministrada. Este diseño permite que los datos permanezcan almacenados de manera estática, sin necesidad de refrescar constantemente la carga eléctrica, como ocurre en la DRAM.

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

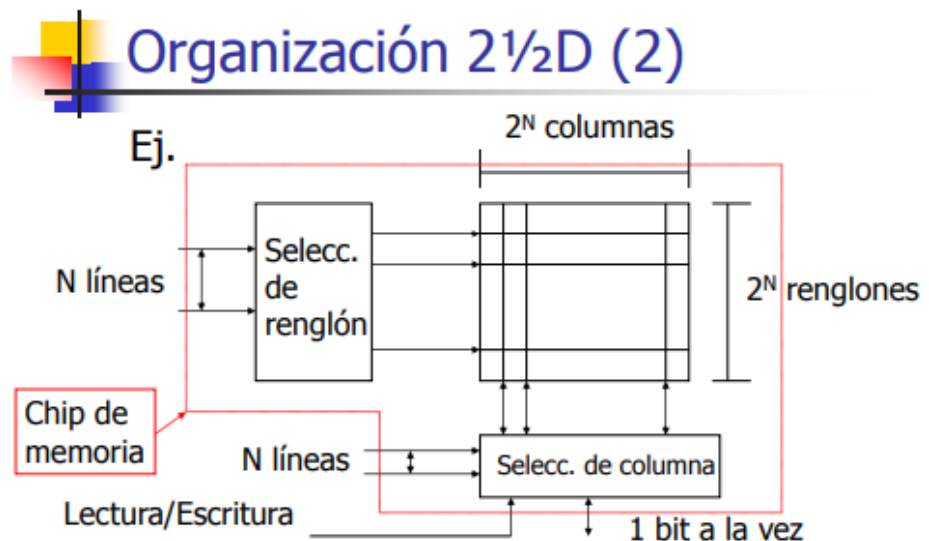


#### ▼ ORGANIZACIÓN 2½D (DRAM)

- La organización **2½D de memoria semiconductora** requiere "refresco" cuando se utiliza en tipos de memoria como la **DRAM**.
- La DRAM está hecha con celdas que almacenan los datos como cargas eléctricas en capacitores. La presencia o ausencia de carga en un condensador se interpretan como el uno o el cero binarios. Ya que los

capacitores tienen una tendencia natural a descargarse (incluso manteniéndola alimentada), requieren refrescos periódicos para mantener memorizados los datos.

- DRAM = MEMORIA PRINCIPAL.
- El arreglo es 'cuadrado' y funciona igual que 2D.
- Los bits de una misma palabra están dispersos en distintos chips.
- La dirección se divide en dos partes (a la mitad): una selección de renglón y una selección de columna. Hay 2 decodificadores. Para seleccionar una celda de memoria necesita la selección de los dos decodificadores.



#### ▼ COMPARACIÓN

- En 2D todos los bits estarán en un mismo chip, en 2½D en distintos.
- La 2D es muy larga y estrecha.
- 2D dificulta el uso de los circuitos correctores de error. En 2½D al estar los bits dispersos en distintos chips hay menos probabilidad de error.
- En 2½D al usar decodificación separada de filas y columnas, reduce la complejidad de los decodificadores.

## ▼ Clase 10: Caché y memoria externa

### ▼ Caché

- Memoria pequeña que se encuentra entre la CPU y la memoria principal.
- Después que la CPU emite una solicitud de lectura a la memoria pasan muchos ciclos de reloj antes que reciba la palabra que necesita.
- La velocidad a la cual la CPU ejecuta instrucciones está limitada por el tiempo del ciclo de memoria.
- La efectividad de la caché se expresa a través de la frecuencia de aciertos.
- Un acierto en la caché sucede cuando los datos que necesita el procesador están almacenados en la caché (la CPU los obtiene a alta velocidad).
- Un fallo en la caché ocurre cuando los datos buscados no están en la caché (la CPU tiene que ir a la memoria principal = más lento).
- Tiene dos niveles L1 y L2. L1 es una memoria caché que se encuentra interna al procesador y trabaja a su misma frecuencia (por eso la rapidez de acceso) y la L2 que es ligeramente más lenta que la L1 pero tiene mayor capacidad.

### ▼ Memoria externa

#### ▼ Discos magnéticos

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

- **Medio:** Platos recubiertos de material magnético, como óxido de hierro o vidrio.
- **Tipos:**
  - **Disco Duro (HDD):** Almacena datos en platos giratorios mediante cabezales que escriben y leen información por magnetización.

- **Discos Floppy (Disquetes):** Discos de baja capacidad y portátiles, usados antiguamente.
- **Estructura:**
  - **Pistas y sectores:** Los datos se organizan en anillos concéntricos (pistas) que se dividen en sectores; estos sectores son las unidades básicas de almacenamiento.
  - **Cilindros:** Estructura formada por las pistas alineadas en los platos.
- **Capacidad:** Se mide en función de los bytes por sector, sectores por pista, pistas por superficie y número de superficies.
- **Tiempo de acceso:** Depende del tiempo de búsqueda (seek) y la latencia de rotación.
- **Formato:** 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 sistema operativo.

#### ▼ Discos ópticos

- **Medio:** Policarbonato recubierto con una capa reflectiva (generalmente de aluminio) que almacena datos en "pits" y "lands" leídos mediante láser.
- **Tipos:**
  - **CD-ROM:** Capacidad de alrededor de 650-700 MB, lectura de datos mediante láser, basado en tecnología de audio.
  - **DVD:** Mayor capacidad que el CD (hasta 17 GB en discos de doble cara y doble capa), con tecnología multicapa y compresión MPEG para video.
  - **Blu-ray:** Capacidad mucho mayor (hasta 50 GB en doble capa) y utiliza un láser azul de menor longitud de onda para almacenar más datos.



- **Ventajas y Desventajas:**

- Los discos ópticos son duraderos, removibles, y fáciles de producir en masa, aunque su velocidad de lectura/escritura es inferior a los discos magnéticos.

- ▼ Cintas magnéticas

- ▼ Clase 11: Periféricos

- ▼ RAID

Los **RAID** son configuraciones de múltiples discos duros que funcionan como una unidad lógica para mejorar el rendimiento, la capacidad o la seguridad de los datos. Esta tecnología distribuye y organiza los datos entre los discos físicos, y en algunos niveles de RAID incluye redundancia mediante el almacenamiento de información de paridad. RAID tiene varios niveles, cada uno diseñado para diferentes necesidades de rendimiento y protección de datos.