Resumen de Organización de Computadoras (OC)

Clase 4

Circuitos Lógicos: Los circuitos lógicos digitales son construidos a partir de puertas lógicas, Las funciones lógicas se implementan interconectando puertas lógicas.

Puertas Lógicas: son dispositivos electrónicos que producen como señal de salida una operación booleana a partir de las señales de entrada. Las puertas Básicas son: And,Or y Not, a partir de ellas se crearon las puertas Nand, Nor, Xor y Xnor, Cada puerta puede definirse de 3 maneras:

- **Ecuaciones Lógicas**: Cada señal de salida se expresa como una función booleana de las señales de entrada.

Ejemplo: Una compuerta And puede expresarse como: (con 2 entradas: “A y B”) F(resultado)= A*B o AB.

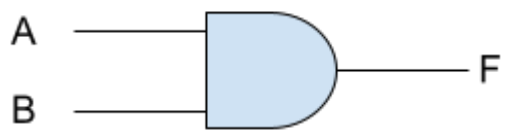
- **Tablas de Verdad:** Para cada una de las 2^**N** combinaciones posibles de las **N** señales de entrada, se enumera el valor binario de cada una de las **M** señales de salida, es decir, a cada combinación de las entradas le corresponde una señal de salida.

Ejemplo: Compuerta And

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

- **Símbolo Gráfico**: El circuito es representado a partir de las puertas lógicas base, interconectadas.

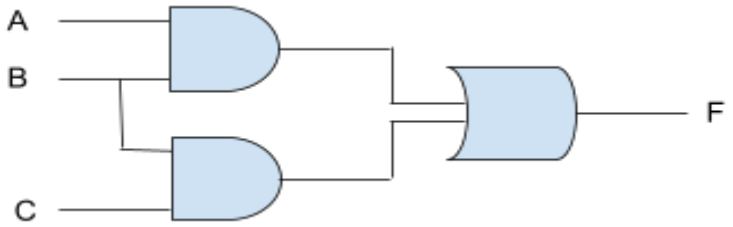
Ejemplo: Compuerta And.



Cada puerta tiene una o 2 entradas y una salida, Cuando los valores de entrada cambian, casi inmediatamente cambian los valores de salidas, con un pequeño retraso en la propagación de la señal a través de las compuertas (Retardo De Puerta).

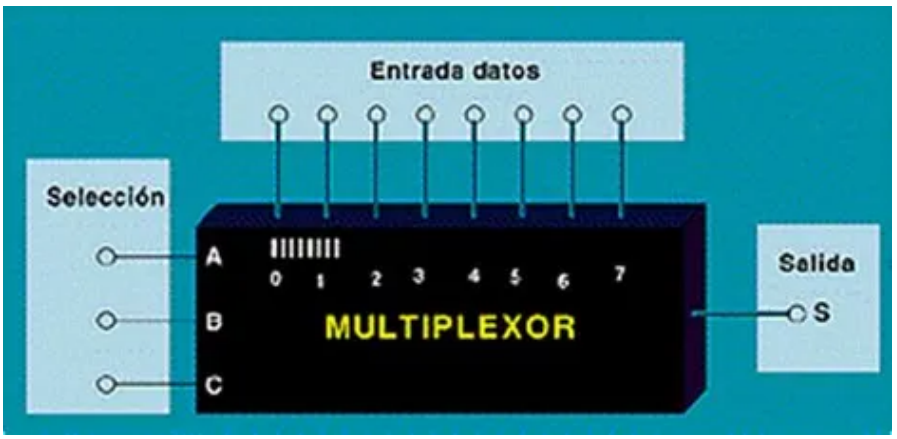
¿Por qué las computadoras utilizan el sistema binario?
Porque para las computadoras es mucho más fácil procesar los datos con el código binario ya que solo posee 2 estados.

Circuitos Combinacionales: Un circuito combinacional es un conjunto de puertas lógicas interconectadas, cuya salida, depende del valor de sus entradas en ese instante, como con las puertas lógicas básicas. Estos circuitos implementan las funciones esenciales de un computador digital. aunque, a excepción de ROM,no proporcionan memoria o información de estado.
Ejemplo Gráfico : Como vemos, este circuito está compuesto por 2 compuertas AND y una OR.

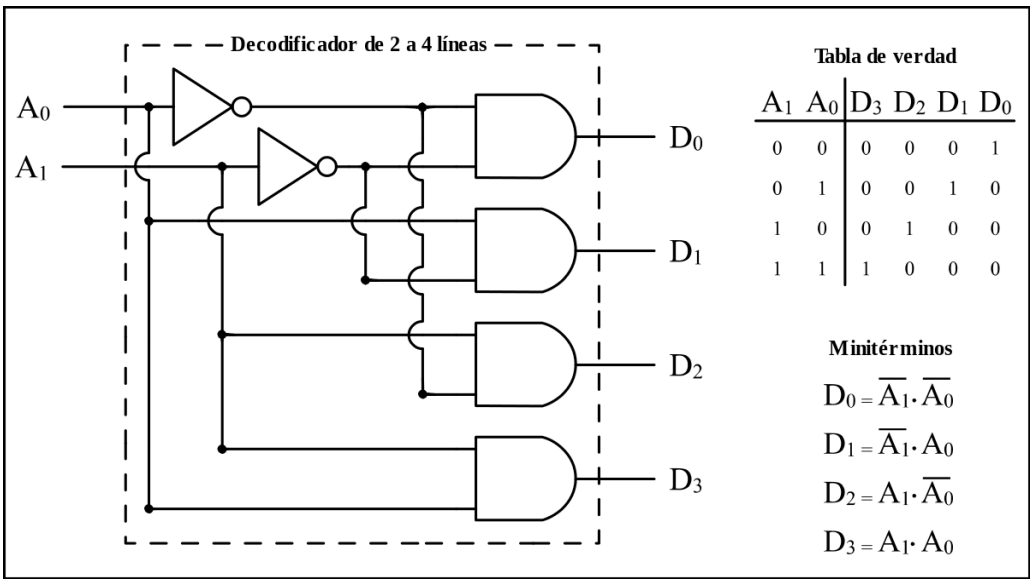


- **Multiplexores:** Este circuito conecta varias entradas a una única salida, cuenta con entradas de control capaces de seleccionar una de las entradas de datos para que pase por la salida. Los multiplexores son utilizados en circuitos digitales para controlar el enrutamiento de señales de datos. Ejemplo: la carga del PC (Program Counter). El valor a cargar al PC puede venir de una o varias fuentes:
- **De un contador Binario**. Si el PC se va a incrementar para la siguiente instrucción.
- **Del registro de instrucción**, Si se acaba de ejecutar una instrucción de salto usando direccionamiento directo.
- **De la salida de la ALU,** si la instrucción de salto específica la dirección usando modo de desplazamiento.

Las distintas entradas pueden estar conectadas a un multiplexor con el PC conectado en salida, estas líneas deben seleccionar cual es el valor a cargar al PC, como este cuenta con varios bits, deben utilizarse varios multiplexores, uno por bit.



- **Decodificadores:** Un decodificador es un circuito combinacional con varias líneas de salida, con una sola de ellas seleccionada en un instante dado, dependiendo del patrón de líneas de entrada, generalmente un decodificador tiene **N** entradas y 2^**N** salidas.

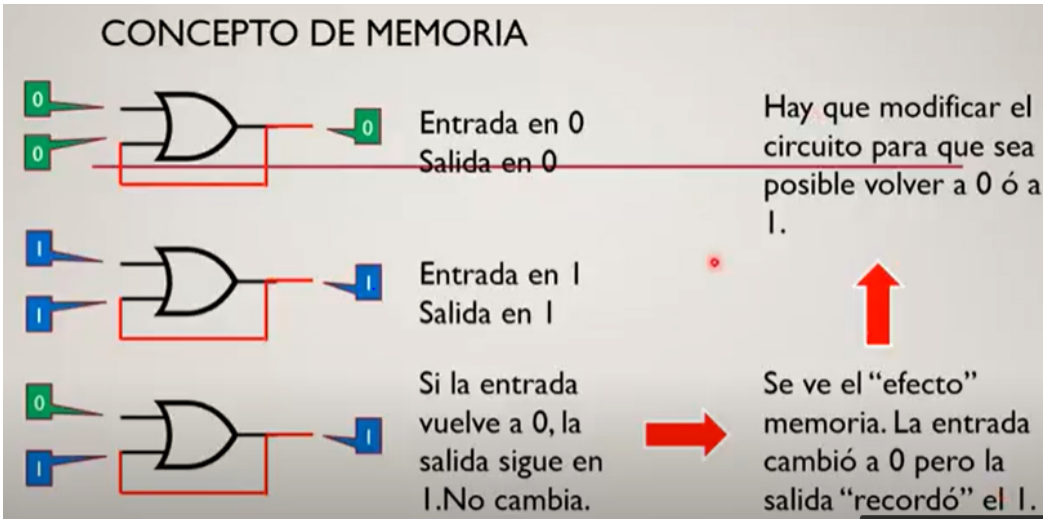


- Los decodificadores tienen varios usos en computadoras digitales. Ejemplo: La decodificación de direcciones.

Circuitos Secuenciales: Estos circuitos digitales son más complejos que los combinatorios, La salida actual de un circuito secuencial depende, no solo de la entrada actual, sino también de la historia pasada de las entradas.

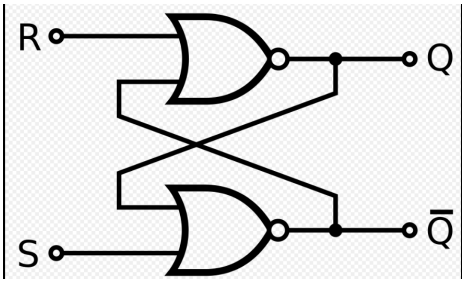
A diferencia de los circuitos combinacionales, en estos circuitos aparecen lazos de retroalimentación. **Las salidas del circuito pueden actuar como valores de entrada**.

Concepto de memoria:



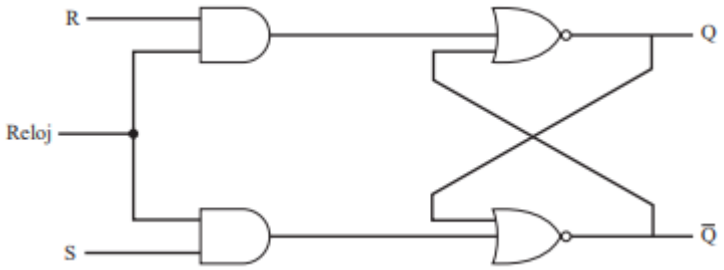
Tipos De circuitos secuenciales:

- **Biestables**: La forma más sencilla de representar un circuito secuencial, existen varios tipos y todos comparten 2 propiedades:
- El biestable es un dispositivo con 2 estados. Está en uno de dos estados, en ausencia de entrada, recordando el último estado. Entonces, el biestable puede funcionar como una memoria de 1 bit.
- El biestable tiene 2 salidas, una complementaria a la otra, normalmente denominadas como Q y ~Q.
- **El cerrojo(latch) S-R:** Este circuito tiene 2 entradas, S(set) y R(reset), y 2 salidas, Q y ~Q, y consiste en dos puertas NOR conectadas por retroalimentación. Este circuito cambia, tras un breve retardo de tiempo, en respuesta a un cambio en la entrada. Esto se denomina operación asíncrona.

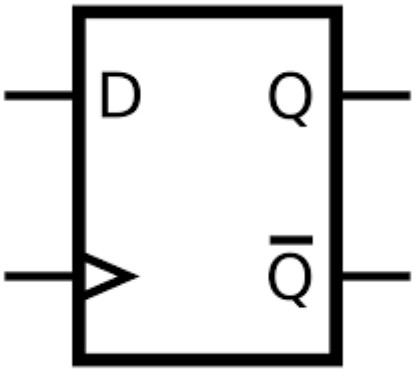


- **S-R Síncrono:** La mayoría de los cambios producidos en una computadora digital están sincronizados por un pulso de reloj. Este

dispositivo denominado biestable RS síncrono, cuenta con 2 entradas que solo aplican los cambios a las entradas de las compuertas NOR solo durante el pulso del reloj.



- **Biestable D**: Los biestables S-R tienen el problema de la condición R=1, S=1 que debe ser evitada. Este biestable soluciona este problema limitando el circuito a una sola entrada(D). Utiliza un inversor para que la “segunda entrada” sea distinta a D. La salida del biestable D es siempre igual al valor más reciente aplicado a la última entrada.



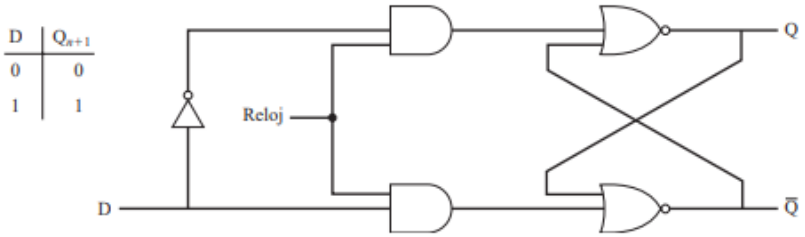


Figura B.27. Biestable D.

- **Biestable J-K:** Cuenta con 2 entradas como el S-R, sin embargo, todas las combinaciones posibles de los valores de entrada son válidos. Las primeras 3 combinaciones son las mismas que el biestable S-R. Sin entrada la salida es estable. J hace que la salida sea 1 y K 0, cuando ambas entradas son 1, la función realizada se denomina conmutación: la salida se invierte.
- Estos Biestables, pueden ser utilizados como contadores.

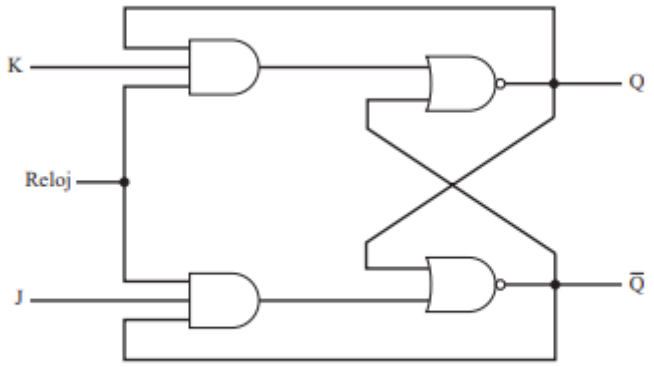
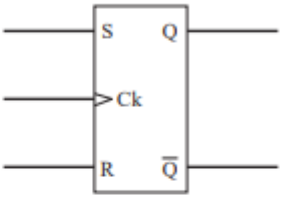
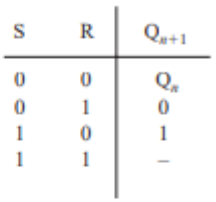
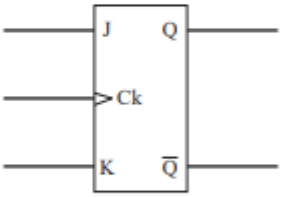
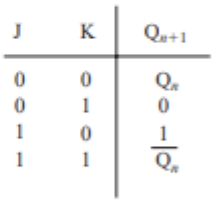
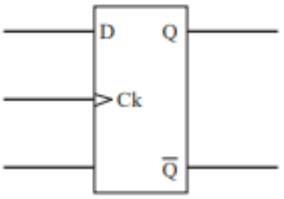


Figura B.28. Biestable J-K.

Nombre	Símbolo gráfico	Tabla característica
S–R		
J–K		
D		

La salida Q_{n+1} se refiere al valor de la salida resultante de las nuevas entradas.

Señal de Reloj (Clock): Indica cuando se va a producir un cambio.

Suma de Productos: Cuando se tiene un circuito, puede implementarse a partir de una tabla de verdad, se toman las combinaciones

de las entradas que generan un 1 en su salida y se hace una suma de sus productos.

:

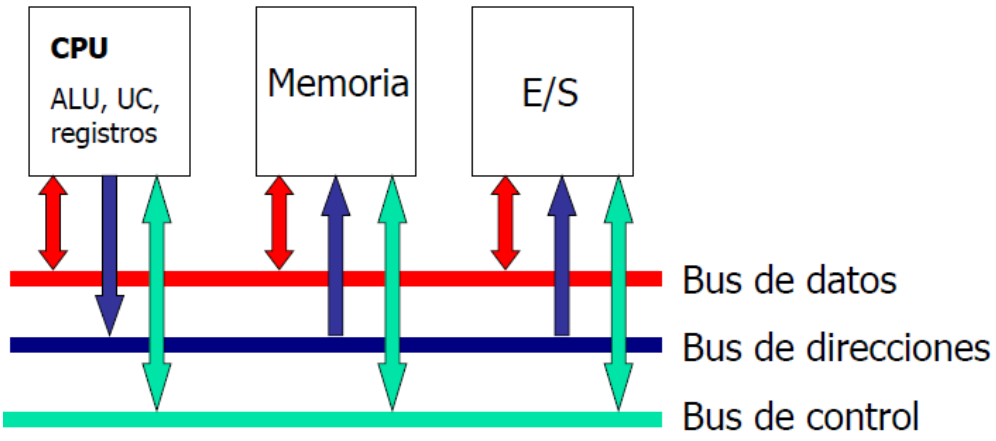
A	B	F
1	0	1
0	1	1
0	0	0

~AB + A.~B

Clase 5

Estructura de una computadora: Una computadora está compuesta por la CPU(unidad central de procesamiento), Memoria y los dispositivos de entrada/ salida, conectados por un sistema de interconexión (los buses).
La **CPU** está compuesta por registros, una **ALU** y una unidad de control, conectadas mediante una interconexión interna(buses) de la **CPU**.
La **Unidad De Control** está compuesta por decodificadores y registros, una lógica de Secuenciamiento y una memoria de control. Esta se encarga de decodificar instrucciones e interpretarlas para luego tomar acciones para llevarlas a cabo.

Interconexión de un sistema de cómputo: los componentes deben estar conectados entre sí.
Definición de Bus: Es un medio de comunicación compartido, es decir, van a todos los dispositivos, cada uno identificado por una dirección única, por el bus de direcciones viajan los unos y ceros que identifican cada dispositivo.
Bus de direcciones: Es un conjunto de cables, donde viajan unos y ceros para identificar a un dispositivo particular.
La CPU, a través de la unidad de control escribe sobre el bus de direcciones la dirección a la que se quiere conectar, la memoria no cambia las direcciones y los periféricos tampoco.
Bus de Datos(o de información): Las instrucciones y datos del programa que se encuentran en memoria, viajan a la CPU a través del bus de datos.
Bus de Control: Viajan unos y ceros que son señales,utilizadas para organizar toda la tarea. Este bus es controlado por la UC.



Aclaración:Todo pasa dentro de la CPU,en una suma, se llevan los operandos(números sobre los cuales se lleva a cabo la suma) a los registros de la CPU, y la ALU efectúa la suma, guardando el resultado donde lo indique la instrucción del programa.

Arquitectura Von Neumann:

Se le atribuye al matemático John Von Neumann, la idea del concepto de “programa autocontenido” ya que antes, cada vez que se quería hacer un programa había que cambiar todo el Hardware, por lo que pensó en que sería más sencillo si el programa se guardara en la memoria, junto con los datos. Entonces, un ordenador podría conseguir sus instrucciones leyendo de la memoria, y se podría hacer o modificar un programa colocando los valores en una zona de memoria.

Se comenzó el diseño de un nuevo computador de programa-almacenado en 1946, y se terminó en 1952, su estructura general era:

- **Una memoria principal**, que almacena datos e instrucciones.
- **Una unidad aritmético-Lógica**, capaz de hacer operaciones con datos binarios.
- **Una unidad de control**, que interpreta las instrucciones en memoria y provoca su ejecución.
- **Un equipo de entrada-salida** (E/S) dirigido por la unidad de control.

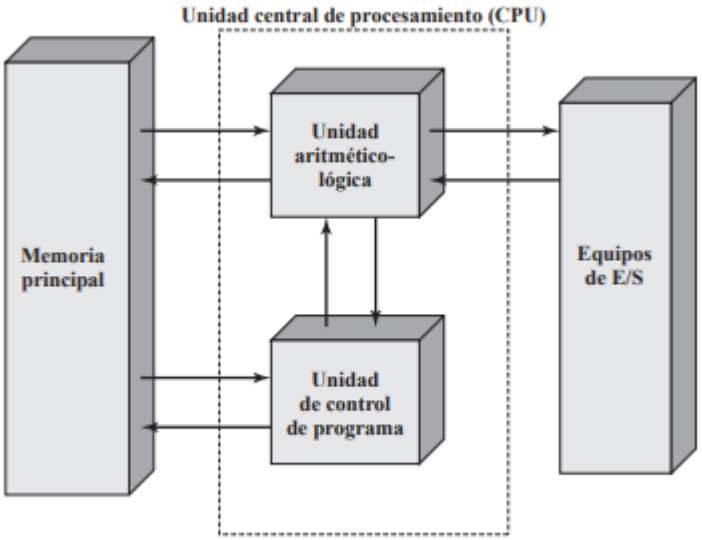


Figura 2.1. Estructura del computador IAS.

La CPU cuenta con posiciones de almacenamiento llamadas registros para realizar sus operaciones:

- **Registro temporal de memoria(MBR):** Contiene una palabra que debe ser almacenada en la memoria, o es usado para recibir una palabra procedente de la memoria. Es el intermediario entre el Bus de Datos y la CPU.
- **Registro de dirección de memoria(MAR):** Especifica la dirección en memoria de la palabra que va a ser escrita o leída en MBR. Es el intermediario entre el Bus de Direcciones y la CPU.

- **PC(program Counter):** Cada vez que se ejecuta un programa, se carga en memoria, y el sistema operativo llena el PC con la dirección de la primera instrucción a ejecutar.
- **IR(Registro de instrucciones):** Se guarda la instrucción a ejecutar, es interpretada por la unidad de control y maneja el bus de control, de acuerdo a las tareas que se deben realizar.

Clase 6

Concepto de Programa: Es una secuencia de pasos, donde se realiza una operación aritmético/Lógica o un movimiento de datos por cada paso, para cada operación, son necesarias diferentes señales de control, la unidad de control saca toda la información necesaria de cada instrucción.
Un programa está compuesto por instrucciones almacenadas en memoria, procesadas por la CPU, que trae de una a la vez desde memoria, cumpliendo cada operación de manera ordenada.

Ciclo de Instrucción:

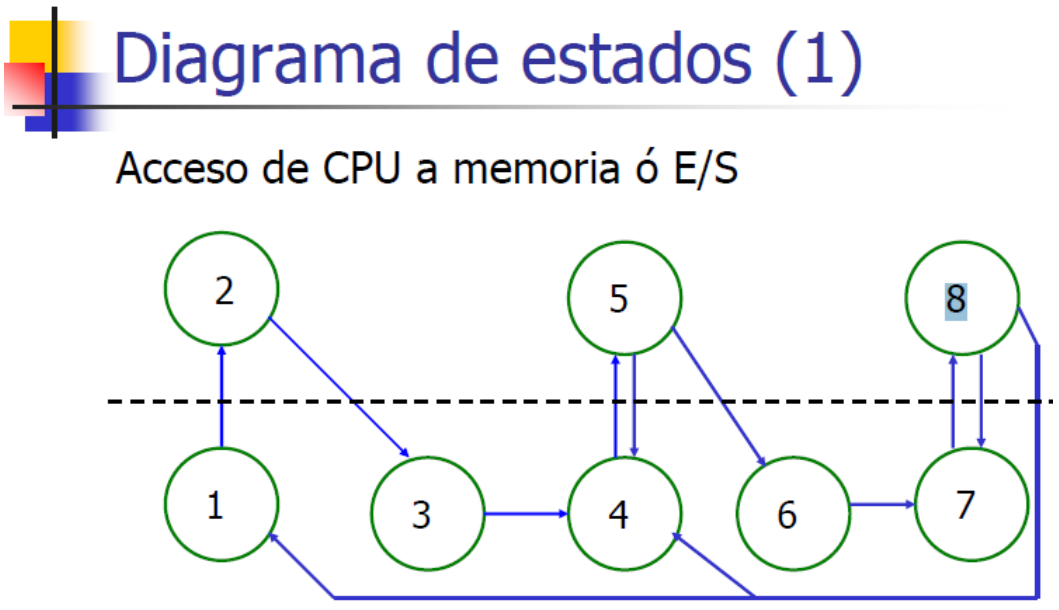
Este ciclo descompone el procesamiento de una instrucción en 2 etapas:

- **Ciclo de Búsqueda**: Se lee la instrucción desde la memoria y se la lleva al registro IR, y la UC decodifica la instrucción.
- **Ciclo De Ejecución**: Ejecuta la instrucción e incrementa el valor del PC.

La CPU va a buscar una instrucción a memoria, la carga en el PC, que siempre apunta a la dirección de la siguiente instrucción a ejecutar, se carga el IR con la instrucción, el registro de instrucciones se la manda a la unidad de control para que la intérprete y lleva a cabo las acciones requeridas, que pueden ser:

- Una transferencia de datos entre CPU y memoria.
- Una transferencia de datos entre CPU y E/S.
- Procesamiento de datos: operaciones aritmético-Lógicas en datos.
- Control: alterar la secuencia de ejecución de instrucciones, ej: llamar a una subrutina.

Luego se incrementa el valor de PC para que apunte a la siguiente instrucción a ejecutar.



Explicación del diagrama de estados: Se calcula la dirección de la instrucción[1], se la va a buscar a memoria[2], se la decodifica[3], se calcula la dirección del/los operando/s[4], se lo/s va a buscar a memoria o a los dispositivos E/S[5], se opera sobre los datos[6], se calcula la dirección del operando destino[7] y se lo almacena en memoria o en los dispositivos E/S[8].

Clase 7

- Las instrucciones son autocontenidas, porque cada instrucción debe tener la información de, que se debe hacer, sobre qué datos y donde almacenar el resultado.

Formato de Instrucción: elementos que debe tener una instrucción:

- **Código de operación**: un conjunto de unos y ceros que indican lo que debe hacer la instrucción.
- **Operando/s Fuente**: Indica en donde se encuentra el/los operandos con los cuales se va a trabajar según la instrucción dada.
- **Operando Destino**: Indica el lugar en el que se debe guardar el resultado de la operación.
- **Referencia a la siguiente instrucción**: Esto le dice a la CPU donde debe buscar la siguiente instrucción a ejecutar, luego de la ejecución de la instrucción anterior. En la mayoría de los casos, una instrucción se encuentra a continuación de la otra, pero en caso de que se provocase un llamado de subrutina podría encontrarse en otra dirección.
- Tanto el operando fuente como el operando destino, pueden encontrarse en la memoria, los registros de la CPU o en dispositivos de E/S.

Tipos de Instrucciones:

- **Procesamiento de datos:** Operaciones aritméticas y lógicas sobre los datos.
- **Almacenamiento de datos:** Transferencia de datos dentro del sistema. ej: almacenar un dato en los registros de la CPU o la memoria.
- **Instrucciones de E/S:** Transferencia de datos entre la computadora y dispositivos externos.

ej: comunicación o intercambio de datos de la CPU con un disco, o una placa de red.

- **Control:** Instrucciones de comprobación y de bifurcación.

Diseño del conjunto de instrucciones: El aspecto de diseño más básico a considerar en el formato es la **longitud o tamaño de la instrucción. Esta decisión**

afecta, y se ve afectada por, el tamaño de la memoria, su organización, la estructura de buses, la complejidad del procesador y la velocidad del procesador. Esta decisión define la riqueza y flexibilidad de la máquina desde el punto de vista del programador en lenguaje ensamblador. se debe tener en cuenta:

- **El tipo de operaciones que se podrán realizar:** Se debe saber cuantas y cuales se sumarán al repertorio de instrucciones de máquina ya que es el medio que tiene el programador para controlar el procesador, esto define la cantidad de bits que va a tener el código de operación de una instrucción.
- **Tipos de datos:** Los distintos tipos de datos con los que se efectúan operaciones.
- **Los Formatos de instrucciones: Longitud de la instrucción (en bits), número de direcciones,** tamaño de los distintos campos, etc.
- **Los registros:** Número de registros del procesador que pueden ser referenciados por las instrucciones, y su uso.
- **Direccionamiento:** El modo o modos de direccionamiento mediante los cuales puede especificarse la dirección de un operando.

Longitud de la instrucción: El programador desea más codops, ya que podría resolver las mismas tareas con programas más cortos, mayor cantidad de MDD porque dan más flexibilidad para implementar funciones como gestionar tablas, más memoria permitiría el direccionamiento de rangos de memoria más grandes. Si se implementaran estas reformas, la longitud de las instrucciones sería mucho mayor y tal vez no aumentaría tanto la utilidad.
Una cuestión relacionada es la velocidad de transferencia de la memoria. Esta velocidad no es comparable con el aumento de la velocidad de los procesadores. Por ello, la memoria puede convertirse en un cuello de botella si el procesador puede ejecutar las instrucciones más rápido que lo que tarda en captarlas. Una solución a este problema es el uso de memoria caché, otra es utilizar instrucciones más cortas.

Número de Direcciones:

- **Si tengo una máquina de 4 direcciones,** la instrucción cuenta con la dirección de 2 operandos fuente, la dirección del operando destino y la dirección de la próxima instrucción, las instrucciones ocupan muchos bits en referencias a memoria.(si cada dirección ocupará 24 bits, la instrucción tiene 96 bits de referencias).
- **Una máquina de 3 direcciones**, cuenta con la dirección de 2 operandos fuente y un operando destino. en este caso, la referencia a la siguiente instrucción a ejecutar se guarda en un registro especial de la CPU (El contador del programa) y cada vez que se termina de ejecutar una instrucción, se incrementa el PC para que apunte a la dirección de la siguiente instrucción a ejecutar, las instrucciones siguen ocupando muchos

bits en referencias a memoria(siguiendo el ejemplo anterior, la instrucción tiene 72 bits de referencias).

- **Una máquina de 2 direcciones** cuenta con 2 operandos, por lo que, siguiendo los ejemplos anteriores, la instrucción tiene 48 bits de referencias. Esta máquina cuenta con 2 operandos Fuente y el primero de estos operandos, es la dirección en la que se va guardar el resultado de la operación, modificando al primer operando, por lo que, es importante guardar el op 1 en otro lugar. Al igual que la máquina de 3 direcciones, el contador del programa guarda la dirección a la siguiente instrucción a ejecutar.
- **Máquina de una dirección:** Se cuenta con la dirección de un solo operando, la CPU cuenta con un registro especial llamado Acumulador, y se suman 2 instrucciones para cargar y descargar este registro. El contador del programa guarda la dirección a la siguiente instrucción a ejecutar, la instrucción tiene 24 bits de referencias.Ej: si se quiere realizar una suma, se debe captar el primer operando, cargarlo en el acumulador, pasar a la siguiente instrucción, captar el segundo número de la suma y sumar el operando con lo guardado en el acumulador.

Modos de direccionamiento: Son las distintas maneras de especificar donde se encuentran los datos de una instrucción.Los modos de direccionamiento tienen como objetivo reducir el tamaño de bits de las direcciones referenciadas por los operandos. **Hay 2 métodos generales para reducir esta cantidad de bits:**

- **Utilizar los registros como lugar de almacenamiento temporal:** Si un operando va a utilizarse varias veces, puede colocarse en un registro para conseguir un **rápido acceso** al operando(porque **me ahorro ir a buscar el dato a través de los buses externos**) y se necesitan **menos bits**(por que los registros siempre son menos a las posiciones de la memoria).

Modos de direccionamiento:

- **Direccionamiento por registro:** ambos operandos se encuentran almacenados en registros, este mdd tiene la desventaja de que la cantidad de registros es baja y la ventaja de que no requiere accesos a memoria, por lo que es muy rápido.
- **Direccionamiento indirecto por registro:** uno de los operandos es un registro y el otro es una dirección de memoria almacenada dentro de un registro. Este mdd requiere un acceso a memoria.
- **Direccionamiento inmediato:** uno de los operandos es un registro o una posición de memoria y el otro un dato inmediato. ej; (MSX88) add BX, FFh.

- **Direccionamiento directo:** uno de los operandos es un registro y el otro una variable o posición de memoria, este mdd requiere un acceso a memoria.
- **Direccionamiento indirecto por memoria:** En la instrucción está la dirección de la dirección del operando. Trata de solucionar el problema del MDD directo. Así, con una dirección de menos bits en la instrucción, se apunta a una dirección de más bits**.** Este MDD es más lento debido a la cantidad de accesos a memoria.
- **Del stack:** El modo de direccionamiento de pila es una forma de direccionamiento implícito. Las instrucciones máquina no necesitan incluir una referencia a memoria sino que operan implícitamente con la cabecera de la pila.
- **Mdd Por desplazamiento:**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:** La dirección de la instrucción actual se suma al campo de dirección para producir la dirección efectiva.
    - **De registro base:**El registro referenciado contiene una dirección de memoria y el campo de dirección tiene un desplazamiento.
    - **indexado:** Se direcciona la memoria con un registro más un desplazamiento,se utiliza un registro llamado índice.

Clase 8

Organización de registros:

Registros que son visibles al programador, es decir, que pueden ser nombrados al escribir una instrucción y se pueden clasificar en:

- **Propósito general**: Pueden contener el operando para cualquier operación.
- **Datos:** solo se les podrá cargar datos.
- **direcciones:**solo se les podrá cargar direcciones.
- **Códigos de condición:** toman valores dependiendo de la última operación aritmética realizada(como los flags de zero, negativo,overflow o carry).

No visibles al programador, es decir, que no pueden ser nombrados ni modificados por el usuario, como los registros de estado y control que son utilizados por la Unidad de control para controlar la operación de la CPU, los 4 esenciales para la ejecución de las instrucciones son:

- **Contador del programa(Program Counter, PC).**
- **Registro de instrucción(IR).**
- **Registro de dirección de memoria(MAR).**
- **Registro Buffer de memoria(MBR).**

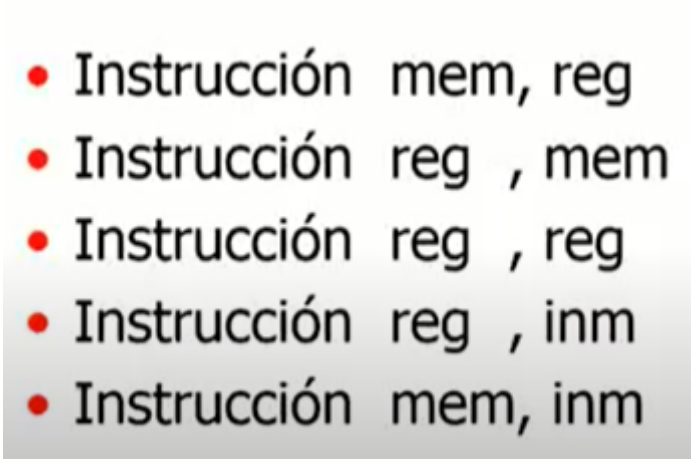
Aclaración: todos estos registros están definidos al final del capítulo 5.

Estos registros se utilizan para el movimiento de datos entre la CPU y la memoria.

Número de registros: la cantidad de registros afecta el tamaño de las instrucciones(**cuantos más registros, más bits deben utilizarse para especificarlos en la instrucción**). Si se tienen **pocos registros**, las instrucciones que los utilizan ocupan menos bits, pero **crecen las referencias a memoria.**
Longitud de los registros: Los registros de direcciones deben tener la capacidad para almacenar la dirección más grande, los de datos deben estar habilitados para almacenar la mayoría de los tipos de datos.

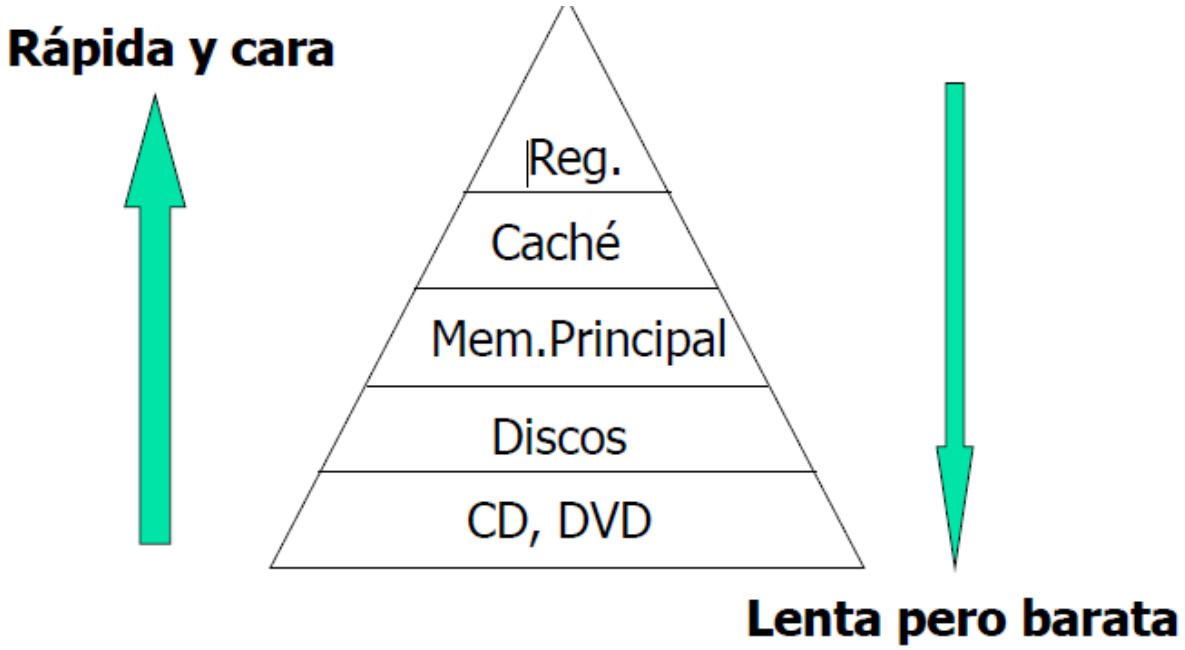
Bits de condición(banderas, Flags): son **bits establecidos por la CPU como resultado de operaciones**, no son modificados directamente por el programador. **Ejemplo**: Cuando utilizo un condicional If y comparo 2 números para ver si son iguales, se realiza una operación aritmética y, en vez de guardar el resultado, se visualiza el estado de los Flags, si la bandera da 1, ya sé que la condición se cumplió.

Instrucciones en Intel: las instrucciones tienen la forma: **Instrucción Op Destino, Op Fuente.**
No se permite que los 2 operandos de instrucción accedan a memoria,se deberá pasar la información de al menos un operando a un registro. El operando a la izquierda de la coma siempre será el que guarde el resultado de la operación, o sobre el que se hará la transferencia de datos.



Clase 9

Jerarquía De Memoria:



¿Por qué funciona? Funciona por el **principio de localidad de las referencias** que básicamente dice que cuando un programa hace una referencia a memoria, es muy probable que acceda de nuevo a esa referencia o a una cercana y también es probable que se acceda a la palabra de nuevo en un "corto" lapso de tiempo.
El funcionamiento de la jerarquía de memoria se encuentra sustentado por 2 principios que exhiben los programas, ósea, que cumplen los programas, **NO LA MEMORIA.**
Principio de localidad espacial de referencia: Cuando se accede a una palabra de memoria, es muy probable que el próximo acceso sea cercano a la que accedió recientemente.
Ejemplo de localidad espacial: En la ejecución secuencial de un código, una instrucción está debajo de la otra, cuando se hace uso de una pila, los datos guardados en la misma son cercanos.
Principio de localidad temporal de referencia: Cuando se accede a una posición de memoria es muy probable que en un lapso "corto" de tiempo, se acceda nuevamente.
Ejemplos de localidad temporal: Formación de bucles o ciclos,**llamados a procedimientos o funciones.**
Gracias a la jerarquía de memoria se puede conseguir una reducción de accesos a memoria por parte del procesador mientras se desciende por la misma, por lo que la ejecución de los programas es más rápida.

¿Cómo está compuesta una jerarquía de memoria?

En un sistema de cómputo, no se maneja un solo tipo de memoria, sino que hay **varios tipos de memorias y tecnologías de distintas velocidades** que trabajan de manera coordinada para tener un tiempo de respuesta frente a la CPU tal que,

no sea tan lenta como los discos magnéticos, ni tan rapida como los registros(por que seria muy costoso).
A medida que se asciende en la jerarquía de memoria, aumenta el coste por bit, el rendimiento y la frecuencia de accesos a ese tipo de memoria por parte del procesador.

Objetivo
El objetivo de la jerarquía de memoria es conseguir una **gran capacidad de almacenamiento y una bajo tiempo de acceso**,teniendo en cuenta el coste.

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

Métodos de acceso a memoria:

- **Memoria de acceso aleatorio:** Cada posición direccionable de memoria tiene un único mecanismo de acceso cableado físicamente, por lo que el tiempo es independiente a la zona de memoria a la que se quiere acceder, ósea, acceder a cualquier lugar de la memoria tarda el mismo tiempo.
- **Memoria de acceso secuencial:** El acceso debe realizarse con una secuencia lineal específica. Por ejemplo, si tengo una cinta, y estoy en el principio y el dato que busco está al final de la cinta, va a tardar más tiempo en acceder porque se debe recorrer toda la cinta.
- **Memoria de acceso directo:** Tiene asociado un mecanismo de lectura/escritura,los bloques individuales tienen una dirección única basada en su dirección física. Discos
- **Memoria de acceso asociativo:**Una palabra es recuperada basándose en una porción de su contenido en lugar de su dirección. El tiempo de recuperación de un dato es una constante independiente de la posición o de los patrones de acceso anteriores. Se asocia con la memoria caché.

Organización: El elemento básico de una memoria de semiconductor es la celda de memoria. Las celdas de memoria deben estar conectadas **al bus de direcciones** porque con los unos y ceros que viajan por el bus de direcciones se va a conectar o elegir una celda determinada de memoria, a**l bus de datos** porque es

por donde se intercambia la información, ya sea de lectura o escritura y por último con **el bus de contro**l porque debe indicar si se debe leer o escribir sobre la misma.

Organización del chip:

- **Organización 2D**: Hay w líneas de dirección(significa que hay 2^w palabras dentro del chip para acceder) entran al chip, y con la electrónica de decodificación, se permite elegir una de estas cajitas, luego hay una lógica que decide si se lee o escribe, esto se encuentra conectado al bus de datos, todas las palabras están en el mismo chip, ejemplo: el BIOS.
- Los valores binarios se almacenan utilizando **configuraciones de puertas que forman biestables(Flip-Flops)**. **No requieren refrescos**, porque mientras se mantengan alimentados mediante una tensión continua de corriente retendrá sus datos.
- **Organización 2 1⁄2 D:** Está hecha con celdas que **almacenan los datos como cargas eléctricas en condensadores. Requieren refrescos periódicos** para mantener memorizados los datos debido a que los condensadores tienden a descargarse, por más que sean alimentados constantemente.
  No todos los bits se encuentran en el mismo chip, 2 1⁄2 D tiene mayor capacidad de detección de errores.

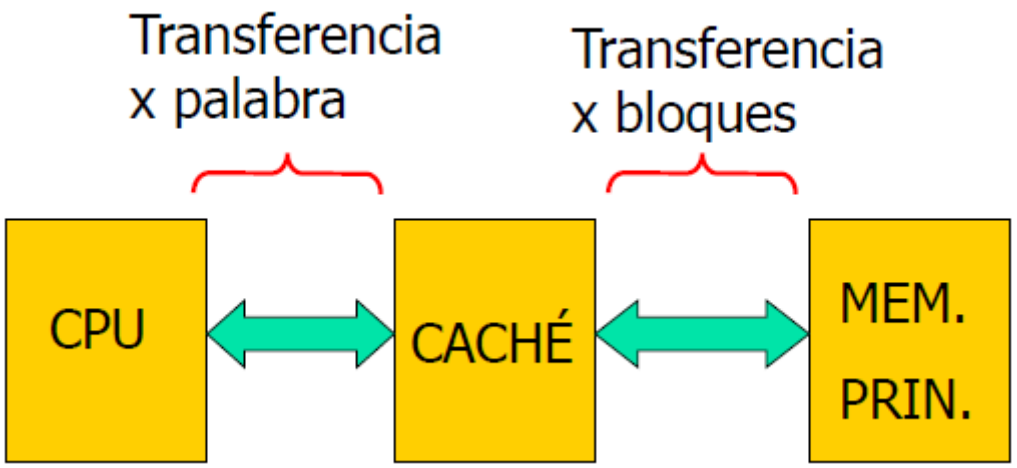
Una celda **2 1⁄2 D** es más **simple** que una **estática** y en consecuencia, **son más densas**, por lo que entran más celdas por unidad de superficie, además, son **más económicas**, esto lleva a que estas memorias sean preferidas en memorias más grandes, como la **memoria principal**, en cambio, **las 2D**, son un poco más rápidas y son utilizadas como **memorias caché**.

Clase 10

Memoria Caché: Esta memoria se utiliza por una cuestión de costo y no de tecnología.

¿Cómo trabaja la caché?

Es una memoria mucho más pequeña (en capacidad) y rápida que se encuentra **ubicada entre la CPU y la memoria principal**, cuando la CPU quiere captar una instrucción de la memoria principal, la caché no copia solo esa instrucción de la memoria principal, sino que copia un bloque, esto es por el principio de localidad espacial de referencia, **toma un bloque de instrucciones porque lo más probable es que la siguiente instrucción a ejecutar sea cercana a la última**, y la CPU toma de a una instrucción de la memoria caché, hasta que se termine el bloque. Esta memoria sirve para aumentar la velocidad de ejecución de un programa, aprovechando los principios que exhiben los mismos.

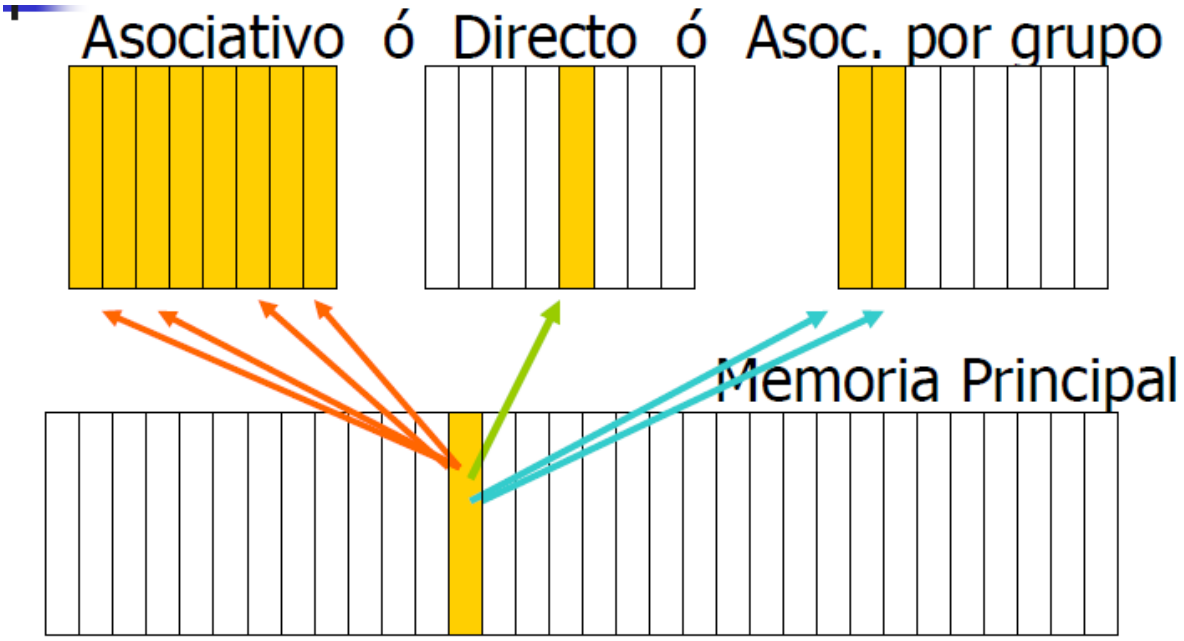


Mapeo de la memoria:

Directo: Es un método bastante simple y de bajo costo, **a cada bloque de datos o instrucciones se le asigna una única línea en la memoria caché**, la **ventaja** de este método, es que a la hora de buscar un dato, **solo se puede buscar en un lugar**, lo que acelera la búsqueda. La principal **desventaja** es que si se les **asigna la misma línea de caché a 2 bloques distintos,** y la CPU solicita repetidamente datos o instrucciones de estos 2 bloques, se están intercambiando constantemente en la caché y **la tasa de aciertos sería baja**.

Asociativo: Este método, permite que **los bloques de datos o instrucciones, se guarden en cualquier lugar de la caché**, supera la desventaja del mapeo directo, pero, cada vez que se copia un bloque en la caché, **además de los datos/instrucciones también se debe guardar información que diga, a qué bloque de la memoria principal pertenece**.

Asociativo Por Grupos: La memoria caché se divide en conjuntos, cada conjunto se divide en k líneas y a **cada bloque de la memoria principal se le asigna un conjunto y una de esas k líneas**, este método mejora los 2 anteriores porque, por un lado, achica la etiqueta de las direcciones de memoria, por lo que **la búsqueda se facilita y no se tiene una única línea en la cual copiar un bloque**, por lo que es poco probable que a 2 bloques que son requeridos constantemente por la CPU se les asigne la misma línea de caché.



Fallos y Aciertos: La efectividad de una caché se mide a través de la frecuencia de aciertos, **un acierto de caché sucede cuando los datos que necesita el procesador se encuentran almacenados en la caché**, lo que permite que la CPU obtenga datos a gran velocidad.
Un fallo de caché se produce cuando los datos solicitados por la CPU no se encuentran en la caché, por lo que la CPU debe ir a buscarlos a memoria principal, lo que produce que los datos se obtengan a menor velocidad.
Caché L1 y L2: Una caché tiene una tasa de aciertos de más del 90%, se utiliza una segunda caché porque se decidió mejorar un 90% con el mismo razonamiento que enfocarse la falla que es menor que el 10%.

Datos

Representación hexadecimal

Cada dígito hexadecimal se representa con 4 dígitos binarios, porque la base 16 y la base 2 están relacionadas por 2^4, esto puede hacerse porque ambas bases están relacionadas por una potencia exacta.

Máscaras con puertas lógicas

Al aplicar una máscara, el or puede ser utilizado para poner un bit en 1, la and para poner en 0 y la xor para invertir el bit superior.

Overflow

Overflow: En una resta cuando el resultado de 0 - 1 = 1 y cuando 1 - 0 = 0, hay overflow, cuando los 2 son de igual signo, no hay overflow.
En una suma cuando 0 + 0 = 1 y cuando 1 + 1 = 0 hay overflow, cuando los 2 son de distinto signo no hay overflow.

Sistemas de numeración
Sistema posicional: desplazo los dígitos de acuerdo al número que quiero formar.

Punto Flotante: Surge de la necesidad de representar números reales y enteros con un rango de representación muy grande o muy pequeño.

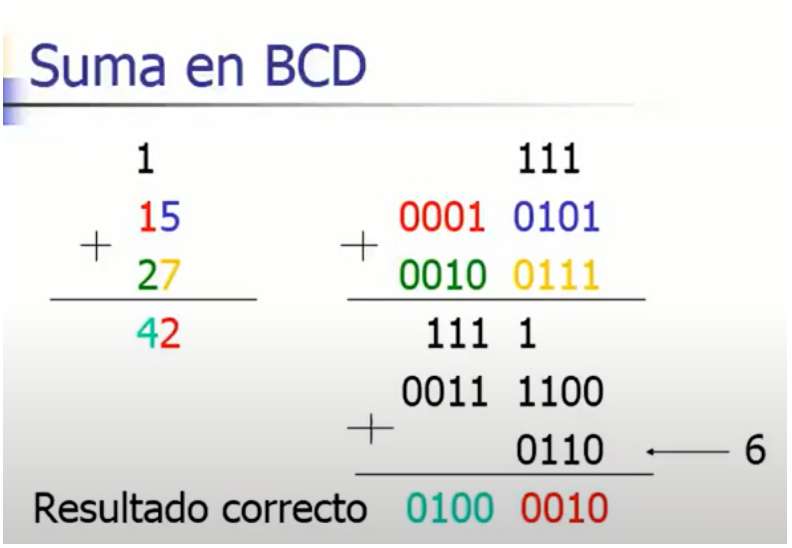
BCD Desempaquetado sin signo
Para pasar un número decimal a BCD desempaquetado sin signo se toma cada número decimal y se lo pasa a binario, cada número tiene 4 bits en 1 por delante para rellenar. ej: 843 = 11111000 11110100 11110011. y si se pasa a Hexadecimal: F8 F4 F3.

BCD Empaquetado con signo
En BCD empaquetado sin signo sería: 843= 00001000 01000011.
Los números en binario se leen de atrás para adelante, por lo que los primeros 4 ceros se agregan para completar el byte y que el número valga lo que debe valer.
la C hexadecimal indica el signo + en el BCD y la D Hexadecimal indica el signo -.

- \+ 834 en BCD desempaquetado: 11111000 11110011 11000100.
- \- 834 en BCD desempaquetado: 11111000 11110011 11010100.

BCD Empaquetado con signo: 1100 = + y 1101 = -, el signo se coloca al final de la cadena del número binario.
ej: +834= 10000011 01001100.
-834= 10000011 01001101.

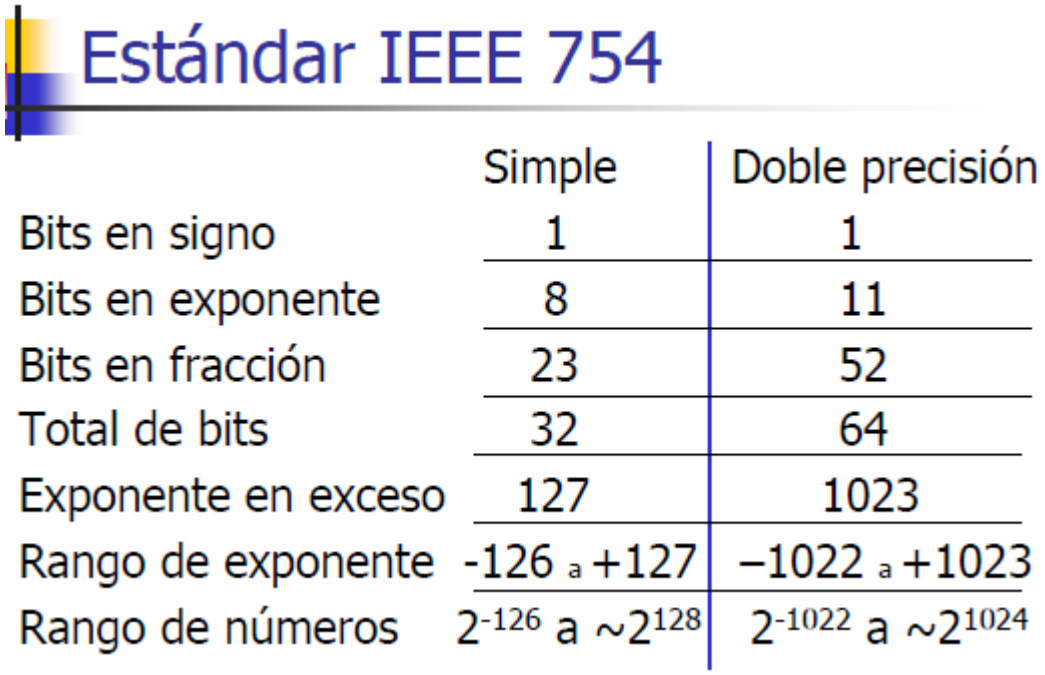
Suma en BCD
Cuando quiero realizar una suma en BCD si alguna de las cadenas de 4 bits da como resultado una combinación mayor a 1001, se le deben sumar las 6 combinaciones que no utiliza el BCD para que se genere el BCD correcto:



De Morgan
Las leyes de De Morgan son importantes porque permiten transformar compuertas Nor en nand y viceversa.

Resolución: Distancia entre dos representaciones consecutivas.

IEEE 754:



	Simple	Doble precisión
Bits en signo	1	1
Bits en exponente	8	11
Bits en fracción	23	52
Total de bits	32	64
Exponente en exceso	127	1023
Rango de exponente	-126 a +127	−1022 a +1023
Rango de números	2^{-126} a $\sim 2^{128}$	2^{-1022} a $\sim 2^{1024}$

Banderas(Flags)

Para asegurarse de que en una cuenta entre 2 cadenas de bits en BSS es correcta, el flag de carry debe estar en 0, si la cuenta se hace en CA2, el flag de Overflow debe estar en 0.

ej: 11110000 + ZCVN: 0001
00001111 BSS: 240 + 15 = 255
11111111 = 255

ej: CA2 11110000 + ZCVN= 0001
00001111 -16 + 15= -1
11111111

Definición de computadora: Una computadora es una máquina digital, sincrónica con la capacidad de realizar cálculos numéricos y lógicos mediante el control de programas, esta máquina permite la comunicación con el mundo exterior.

Si nos preguntan, ¿cómo está compuesta una jerarquía de memoria? tengo que responder que se compone de distintas memorias,o por donde va la pregunta? Puedes construir la pirámide y explicar cada memoria e irte para el lado de porqué funciona.

y en el biestable SR, no se permite que las 2 entradas sean 1 porque las salidas son complementarias y no pueden tener salidas iguales?
Efectivamente, este resultado causa que el biestable se vuelva inestable.

Si tengo que explicar el formato de instrucción,¿describir sus elementos? (cod op, operandos,referencia a la próxima instrucción)
Podes explicar la maquina de 4 direcciones y decir que después baja la cantidad para hacer las instrucciones más chicas.

tengo una duda sobre una cuenta de perifericos

una imagen en una pantalla de 100 cm por 50 cm
posee una resolucion de 100 puntos por cm

Facundo Feliu 17:27
cuantos bytes de memoria se necesitan para almacenar una imagen true color

seria multipliciar

100x50x100x24?

Diego MONTEZANTI 17:29
50 x 100 x 100^2 pixels x 3Bytes/pixel

Calculo de memoria de video de una pantalla alfanumérica: N° Filas x N° de columnas x (8 bits para ascii + bits de color + bits de atributos)
calcular bits de color: si te dicen que hay 256 colores, usas 8 bits.
atributos: subrayado, negrita.

Ejemplo: N° Filas= 80, N°Columnas = 100, 1024 colores == bits de color=10.
atributos = 3
80 x 100 x (8 + 3 + 10) = 168000 bits.
21000 bytes
20KB
Calculo de memoria de video de una pantalla Gráfica:
Alto x Ancho x bits de color.

Error Absoluto Máximo: resolución/2
Error relativo: Error Absoluto/ num a representar.
Error Absoluto: /número a representar- número representado/

Ciclo de instrucción de un salto incondicional: Se calcula la dirección en la memoria, se va a buscar la instrucción mediante el bus de direcciones y se le pasa al PC, la instrucción se guarda en el IR y la uc la decodifica, se calcula la dirección a la que se debe saltar, la ALU calcula cuánto le tiene que sumar al PC mediante una resta entre la dirección de destino y la actual, luego se carga el PC para ejecutar la siguiente instrucción. Esta instrucción utiliza el MDD por desplazamiento relativo al PC**.**

Pasos para representar un número en un sistema cualquiera:
Paso 1: representarlo sin restricciones de bits.
Paso 2:adaptarlo al formato de mantisa y exponente.
Paso 3: adaptarlo al formato del sistema.
Paso 4: Solución.

Alejandro Acosta 2/2/2023.