**Estructura del Computador - 66.70**
**MEMORIA**

**RAM:** memoria de acceso aleatorio
**ROM:** memoria de lectura

**SRAM (Static RAM)**
- Usa flip flops.
- Más veloz pero gasta más recursos.
- Aplica en registros y memoria caché.

**DRAM (Dynamic RAM)**
- Implementada con capacitores.
- Más lenta por el refresco de los capacitores que se descargan.
- Ocupa menos espacio y es más barata.

**CRITERIOS PARA REDUCIR EL TIEMPO DE ACCESO:**
- **Bancos entrelazados:**
  - Accede a un banco mientras en el otro se refresca la información.
  - Enmascara el tiempo de refresco.
  - Mejora el rendimiento si las posiciones sucesivas están en bancos diferentes.
- **Mayor ancho del bus.**
- **Mayor velocidad del clock (SDRAM).**
- **Con igual velocidad de clock se lee durante el flanco ascendente y descendente.**

**CONEXIÓN MEMORIA PROCESADOR**

**MICROPROCESADOR (bus sistema) CONTROLADOR DE MEMORIA (bus interno) RAM**

El controlador de memoria o **MMU** se encarga de manejar el acceso a la memoria física, por ejemplo se encarga de enviar las señales RAS y CAS para determinar de qué columna y fila se leerá y cómo.
Además se encarga de:
- **Traducir las direcciones**, pasando de direcciones virtuales a direcciones físicas de la RAM o disco rígido.
- **Control de caché:** almacena los datos de la memoria física en una caché para que no se tenga que volver a leer de la memoria principal.

- **Control de acceso a memoria:** para que los procesos no accedan a una dirección que no les pertenece.

**TIEMPOS DE ACCESO**

Hay grandes brechas entre los tiempos de acceso entre el disco y la DRAM y entre la DRAM y el CPU.
Como una computadora requiere de mucha memoria y rápida y es difícil de conseguir, se le hace creer que la tiene.

**Brecha DISCO - DRAM -> Memoria virtual**
**Brecha DRAM - CPU -> Memoria caché**

**MEMORIA CACHÉ (la maleducada)**

A partir del principio de localidad:
- **Localidad temporal:** si accedí a una dirección de memoria, volveré a acceder a ella en poco tiempo (iteraciones, procedimientos recursivos).
- **Localidad espacial:** si accedí a una dirección de memoria, hay más probabilidades de que acceda a sus direcciones cercanas. (datos almacenados en posiciones contiguas).

**La memoria caché es una memoria muy rápida, de poco almacenamiento y es cercana físicamente al CPU.**

**Funciona dividida en bloques, donde al acceder a un dato de memoria principal bajo su bloque completo al caché y en el próximo acceso se verifica si la posición buscada está en caché, si no lo leo de la memoria principal y cargo ese bloque.**

**VISIBILIDAD DESDE EL PUNTO DE VISTA DEL PROGRAMADOR**

El programador puede “beneficiar” a la caché si a la hora de programar usa el principio de localidad.

**Memoria Principal vs Caché**

La caché esta construida con electrónica más rápida (SRAM), tiene un árbol de decodificación más pequeño y tiene cercanía física al CPU, no se comunica por bus.

**FALLO DE LA CACHÉ**

Ocurre cuando el procesador busca un dato o una instrucción en su caché y no encuentra una copia válida, así que debe buscar el dato en RAM, lo cual provoca un menor rendimiento.

Luego del fallo, se produce lo siguiente:

Una vez obtenido el bloque de memoria de la RAM se colocará una copia en la caché, pisando algún bloque cargado anteriormente según su política de reemplazo o si hay espacio, añadiéndolo directamente para futuras referencias.

Para optimizar la caché en la programación en alto nivel se deben tener en cuenta 2 cosas: el principio de localidad y que almacena en bloques.

Se propone:

- Según localidad espacial, acceder a las estructuras de datos de forma ordenada, secuencial y predecible.
- Según localidad temporal, almacenar de manera empaquetada las funciones y variables más utilizadas.

**CACHÉ ESPECIALIZADO vs CACHÉ UNIFICADO**

Caché Unificada: almacena juntos datos e instrucciones. Es más fácil de gestionar pero hay menos grado de optimización.

Caché Especializada: separamos los datos de las instrucciones. Se podría generar una caché más sencilla para las instrucciones ya que la mayoría de operaciones son de lectura. Y para los datos una más optimizada para lectura/escritura.

Elegir entre una u otra depende además de la arquitectura, la organización de los datos y la política de reemplazo de la caché.

**CACHÉ MULTINIVEL**

Se utilizan múltiples niveles de caché (2 o 3), la idea es que los datos se almacenen en el nivel más cercano al CPU, si es que allí no se encuentran, se van recorriendo los niveles, de esta manera retrasamos el fallo de la caché, el acceso a la memoria principal.

Una desventaja es que a medida que van aumentando los niveles, estos ocupan mayor espacio físico.