Clase 15 - Jerarquía de Memoria y Memoria Caché

IIC2343 - Arquitectura de Computadores

Profesor:
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl

Arquitectura de Computadores: Mejoras y extensiones

En lo que hemos visto hasta ahora

- Una máquina programable que ejecuta programas
- Interacción con dispositivos de entrada y salida

Arquitectura de Computadores: Mejoras y extensiones

En lo que hemos visto hasta ahora

- Una máquina programable que ejecuta programas
- Interacción con dispositivos de entrada y salida

Lo que nos queda ver **mejoras:**

- Mejora los accesos a memoria
- Mejora en lo que respecta a la ejecución de más de un programa
- Mejora en lo que respecta a la ejecución de instrucciones en la CPU.

Acceso a Memoria

Accesos a memoria en el computador básico

Pasos de la ejecución de ADD A,(dir)

1. Obtención de la instrucción de la memoria. **Fetch**
2. Decodificación de la instrucción en señales de control. **Decode**
3. Obtención del dato ubicado en dir de la memoria. **Memory**
4. Ejecución de la operación ADD en la ALU. **Execute**
5. Escritura del resultado en el registro A. **Write back**

Generalmente, el código que escribimos cumple dos principios de localidad

1. Principio de localidad temporal

Es probable que un dato obtenido de memoria sea usado posteriormente en otra operación

2. Principio de localidad espacial

Es probable que datos cercanos al buscado también sean usados (bloque)

Principio de Localidad Temporal

Es probable que un dato obtenido de memoria sea usado posteriormente en otra operación

```
.data
var1: .word 3
var2: .word 2
res:  .word 0
.text
  main:
    lw t0, var1
    lw t2, res
    while:
      lw t1,var2
      add t2, t2, t1
      addi t0, t0, -1
      beqz t0, end
      j while
    end:
      la t0, res
      sw t2, 0(t0)
```

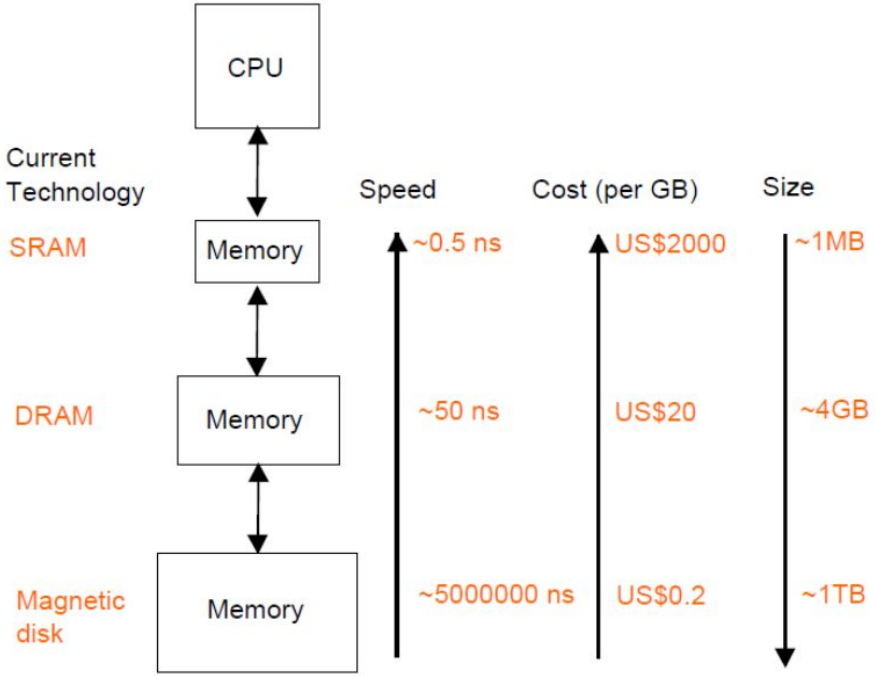
Principio de Localidad Espacial

Es probable que datos cercanos al buscado también sean usados (bloque)

```
.data
arr: .word 1, 0, 6, 12, 1
len: .word 5
index: .word 0
avg: .word 0
.text
  main:
    lw t0, index
    lw t1, len
    lw t2, avg
    la t3, arr
    while:
      beq t0, t1, end
      lw t4, 0(t3)
      add t2, t2, t4
      addi t0, t0, 1
      addi t3, t3, 4
      j while
   end:
      div t2, t2, t1
      la t5, avg
      sw t2, (t5)
```

¿Como podemos aprovechar estos principios para acelerar la ejecución de una instrucción?

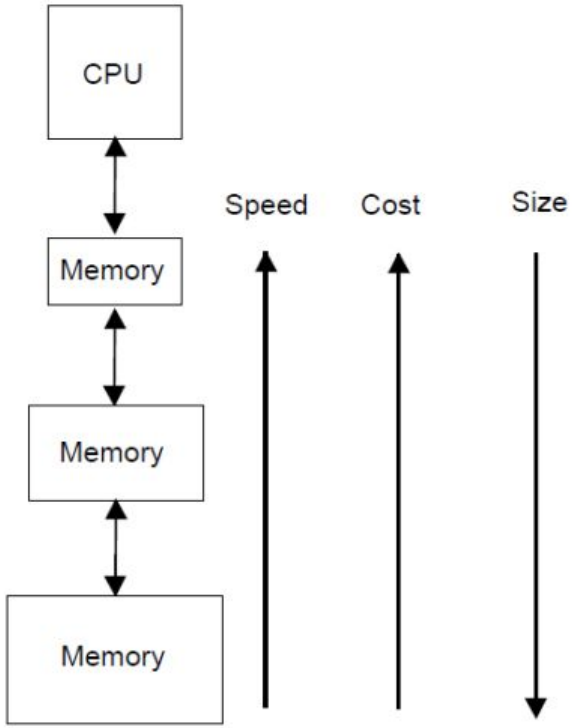
Jerarquía de Memoria es una buena opción para mejorar el rendimiento



¿Cómo evaluamos una jerarquía de memoria?

Evaluaremos una jerarquía de memoria con las siguientes métricas:

- Hit: Búsqueda exitosa de un dato en la memoria caché.
- Miss: Búsqueda fallida de un dato en la memoria caché.
- Hit-rate (HR): #Hits ÷ #Accesos a memoria
- Miss-rate (MR): 1 - HR
- Hit-time (HT): Tiempo que toma buscar un dato en memoria caché, independiente de si se encuentra o no.
- Miss-penalty (MP): Tiempo que toma, luego de un miss en caché, copiar un bloque de memoria del siguiente nivel en la jerarquía en la caché y luego acceder al dato buscado desde esta.



Evaluacion Tiempo promedio

Tiempo promedio en caso de *hit*

Tiempo promedio en caso de *miss* *

$$TP = HR * HT + (1 - HR) * (HT + MP)$$
$$= HT + (1 - HR) * MP$$

Imagen Referencia: https://github.com/IIC2343/Syllabus-anteriores/blob/main/Apuntes%20(viejos)/8%20-%20Comunicaci%C3%B3n%20de%20CPU%20y%20Memoria%20con%20IO.pdf

Memoria caché ocupa el primer nivel de la jerarquía

La memoria caché se encuentra en la CPU...

- ...pero la CPU no tiene idea que existe.
- Memoria principal y caché se dividen en bloques y líneas, respectivamente (localidad espacial).
- Para describir el funcionamiento, sólo es necesario entender su comunicación con la CPU y con el siguiente nivel de la jerarquía (memoria principal) .

Memoria caché ocupa el primer nivel de la jerarquía

CPU sólo conoce, con respecto a la jerarquía de

memoria, el tamaño de la memoria principal.

• Controlador de caché es encargado de realizar la comunicación y coordinación.

• Las funciones que debe implementar el controlador pueden definirse a grandes rasgos en:

1. Mecanismo de acceso a datos

2. Política de escritura

Acceso a Memoria

Mapeo directo (directly mapped) es la función de correspondencia más simple

Con este esquema, cada bloque de la memoria principal se asocia a solo una línea de la caché.

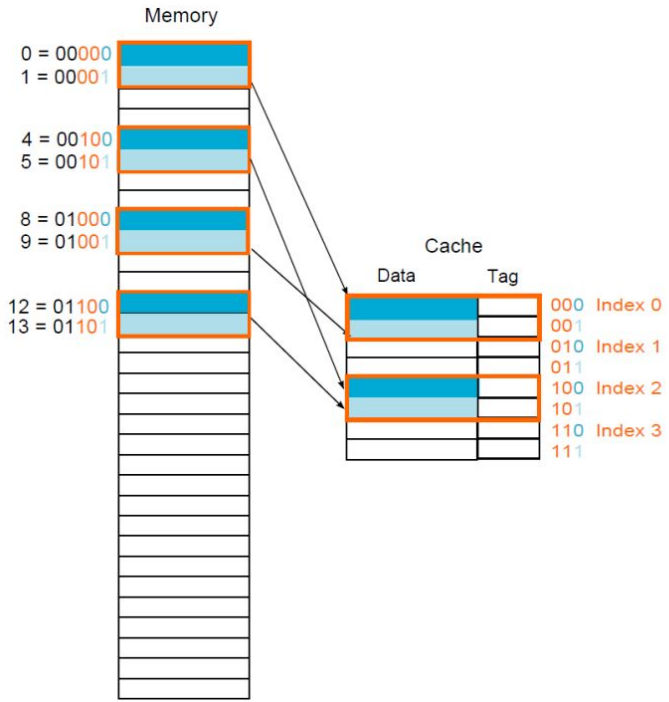
- Cada línea en la caché es identificada con un índice.
- Tamaño de líneas y cantidad de estas siempre serán potencias de 2 (por simplicidad).
- La dirección del bloque de memoria será la dirección de la primera palabra de éste.
- Mapeo de bloque a línea se basa en el operador módulo:

línea = bloque mod num_líneas

Mapeo directo (directly mapped) es la función de correspondencia más simple

Revisemos un ejemplo con los siguientes datos:

- Memoria principal: 32 bytes (32 palabras)
- Caché de 8 bytes y 4 líneas



Mapeo directo (directly mapped) es la función de correspondencia más simple

Además de almacenar las palabras y el tag, cada línea debe tener un bit de validez.

• Este bit se utiliza cuando la caché comienza a ser llenada => palabras son no válidas.

• Finalicemos el mapeo directo con un ejemplo de su funcionamiento:

- Caché de 8 bytes y 4 líneas

- Acceso a direcciones de memoria: 12, 13, 14, 4, 12, 0

Clase 15 - Jerarquía de Memoria y Memoria Caché

IIC2343 - Arquitectura de Computadores

Profesor:
- Felipe Valenzuela González
Correo:
frvalenzuela@alumni.uc.cl