

# Rendimiento del caché

# Temario

- Rendimiento del caché.
- Cachés multinivel.

# Rendimiento del caché

- Dos métodos para medir el rendimiento de una memoria caché.
  1. Medir el tiempo de ejecución contando los ciclos que se pierden por accesos a la memoria.
  2. Medir el tiempo promedio de acceso a la memoria (AMAT).

# Ciclos perdidos

- El caché puede ser 10 veces más lento que el procesador.
- Si se produce una falla en el caché, se tiene que acceder la memoria principal.
- Los ciclos utilizados en acceder la memoria se deben contar para calcular el tiempo de sistema.

# Ecuación del tiempo

- Ecuación del tiempo original:
- $T = C \times P$
- Donde:
- $T$  es el tiempo en segundos.
- $C$  es el número total de ciclos de reloj utilizados por el programa.
- $P$  es el período del reloj en ciclos por segundo.

# Ecuación del tiempo

- Los ciclos  $C$  que hace el programa incluyen:
  1. Ciclos de reloj usados en ejecutar el programa.
  2. Ciclos de reloj perdidos en detenciones por peticiones al caché.

# Ecuación del tiempo

- La ecuación del tiempo se convierte en:
- $T = (C_x + C_s) \times P$
- Donde:
- $T$  es el tiempo en segundos.
- $C_x$  es el número de ciclos de reloj utilizados por el programa.
- $C_s$  es el número de ciclos perdidos por detenciones debido a peticiones al caché.
- $P$  es el período del reloj en ciclos por segundo.

# Ciclos perdidos por detenciones

- Los ciclos perdidos por detenciones vienen principalmente de fallas de caché.
- El total de ciclos perdidos es la suma de los ciclos perdidos por lecturas más los ciclos perdidos por escrituras.
- $C_s = C_{load} + C_{store}$

# Ciclos perdidos por lecturas

- Los ciclos perdidos por lecturas dependen de:
  - Número de lecturas por programa.
  - El castigo por falla debido a lecturas en ciclos.
  - La tasa de fallas de lecturas.

$$\text{Ciclos detención lectura} = \frac{\text{lecturas}}{\text{programa}} \times \text{tasa de fallas por lectura} \times \text{castigo por falla de lectura}$$

# Ciclos perdidos por escrituras

- Los ciclos perdidos por escrituras dependen de la estrategia (write-through o write-back).
- En write-through hay 2 fuentes de detenciones:
  - Fallas de escritura, donde hay que cargar el bloque antes de continuar la escritura.
  - Detenciones por el buffer de escritura cuando el buffer está lleno.

$$\text{Ciclos detención escritura} = \left( \frac{\text{escrituras}}{\text{programa}} \times \text{tasa de falla escritura} \times \text{castigo por falla escritura} \right) + \text{detenciones por buffer escritura}$$

# Ciclos perdidos por escrituras

- En sistemas bien diseñados las detenciones por el buffer de escritura se pueden ignorar.
- En write-back hay posibles detenciones adicionales causadas por escribir un bloque del caché sucio en la memoria. Se ignoran.

# Ciclos perdidos en total

- Por lo general, los castigos por falla de lectura y escritura son iguales.

$$\text{Ciclos de reloj detención memoria} = \frac{\text{accesos memoria}}{\text{programa}} \times \text{tasa de falla} \times \text{castigo por falla}$$

$$\text{Ciclos de reloj detención memoria} = \frac{\text{instrucciones}}{\text{programa}} \times \frac{\text{fallas}}{\text{instrucción}} \times \text{castigo por falla}$$

# Ejemplo 1

- Un sistema con un caché dividido (split).
- Tasa de fallas del caché de instrucciones: 2%.
- Tasa de fallas del caché de datos: 4%.
- Castigo por falla: 100 ciclos.
- El CPI es 2 sin detenciones de memoria.
- ¿Qué tanto mas rápido correría el procesador con un caché perfecto (sin fallas)?
- Suponer las siguientes frecuencias de instrucciones de SPECint2000.

| Core MIPS                  | Name  | Integer | Fl. pt. | Arithmetic core + MIPS-32    | Name  | Integer | Fl. pt. |
|----------------------------|-------|---------|---------|------------------------------|-------|---------|---------|
| add                        | add   | 0%      | 0%      | FP add double                | add.d | 0%      | 8%      |
| add immediate              | addi  | 0%      | 0%      | FP subtract double           | sub.d | 0%      | 3%      |
| add unsigned               | addu  | 7%      | 21%     | FP multiply double           | mul.d | 0%      | 8%      |
| add immediate unsigned     | addiu | 12%     | 2%      | FP divide double             | div.d | 0%      | 0%      |
| subtract unsigned          | subu  | 3%      | 2%      | load word to FP double       | l.d   | 0%      | 15%     |
| and                        | and   | 1%      | 0%      | store word to FP double      | s.d   | 0%      | 7%      |
| and immediate              | andi  | 3%      | 0%      | shift right arithmetic       | sra   | 1%      | 0%      |
| or                         | or    | 7%      | 2%      | load half                    | lhu   | 1%      | 0%      |
| or immediate               | ori   | 2%      | 0%      | branch less than zero        | bltz  | 1%      | 0%      |
| nor                        | nor   | 3%      | 1%      | branch greater or equal zero | bgez  | 1%      | 0%      |
| shift left logical         | sll   | 1%      | 1%      | branch less or equal zero    | blez  | 0%      | 1%      |
| shift right logical        | srl   | 0%      | 0%      | multiply                     | mul   | 0%      | 1%      |
| load upper immediate       | lui   | 2%      | 5%      |                              |       |         |         |
| load word                  | lw    | 24%     | 15%     |                              |       |         |         |
| store word                 | sw    | 9%      | 2%      |                              |       |         |         |
| load byte                  | lbu   | 1%      | 0%      |                              |       |         |         |
| store byte                 | sb    | 1%      | 0%      |                              |       |         |         |
| branch on equal (zero)     | beq   | 6%      | 2%      |                              |       |         |         |
| branch on not equal (zero) | bne   | 5%      | 1%      |                              |       |         |         |
| jump and link              | jal   | 1%      | 0%      |                              |       |         |         |
| jump register              | jr    | 1%      | 0%      |                              |       |         |         |
| set less than              | slt   | 2%      | 0%      |                              |       |         |         |
| set less than immediate    | slti  | 1%      | 0%      |                              |       |         |         |
| set less than unsigned     | sltu  | 1%      | 0%      |                              |       |         |         |
| set less than imm. uns.    | sltiu | 1%      | 0%      |                              |       |         |         |

# Ejemplo 1

- Para el caché de instrucciones:

$$\text{Ciclos de detención} = I \times 0.02 \times 100 = 2 \times I$$

- Para el caché de datos (loads y stores son 35%):

$$\text{Ciclos de detención} = I \times 0.35 \times 0.04 \times 100 = 1.44 \times I$$

- Número total de ciclos de detención de memoria es  $2 \times I + 1.44 \times I = 3.44 \times I$ .
- CPI con detenciones de memoria es  $2 + 3.44 = 5.44$ .
- El porcentaje de tiempo de ejecución perdido por detenciones de memoria es  $3.44/5.44 = 63\%$ .

# Ejemplo 1

- El rendimiento con y sin detenciones se compara:

$$\frac{\text{Rendimiento1}}{\text{Rendimiento2}} = \frac{\text{Tiempo ejecución2}}{\text{Tiempo ejecución1}}$$

- Tiempo de ejecución =  $I \times \text{CPI} \times \text{Período de reloj}$ .

$$\frac{\text{Tiempo CPU con detenciones}}{\text{Tiempo CPU con caché perfecto}} = \frac{I \times \text{CPI}_{\text{detenciones}} \times \text{Período reloj}}{I \times \text{CPI}_{\text{perfecto}} \times \text{Período reloj}}$$

$$= \frac{\text{CPI}_{\text{detenciones}}}{\text{CPI}_{\text{perfecto}}}$$

# Ejemplo 1

- El rendimiento de la CPU con caché perfecto es mejor por  $5.44/2 = 2.77$  veces.

## Ejemplo 2

- Suponer que la velocidad de reloj de la computadora del ejemplo anterior se duplica, pero el tiempo de acceso a la memoria, las fallas de caché y la tasa de fallas son los mismos. ¿Cuánto más rápida será la computadora con el reloj más rápido?
- Si la velocidad de reloj es el doble, el castigo por falla es de  $2 \times 100 = 200$  ciclos.
- Razón: castigo por falla = tiempo de acceso a la memoria x velocidad de reloj.

## Ejemplo 2

- Número total de ciclos por detenciones de memoria:  
 $1 \times 0.02 \times 200 + 1 \times 0.35 \times 0.04 \times 200 = 6.8 \times 1$ .
- CPI con detenciones de memoria:  $2 + 6.8 = 8.8$ .
- Porcentaje de tiempo de ejecución perdido por detenciones de memoria:  $6.8 / 8.8 = 77\%$ .
- Se comparan la CPU del ejemplo 1 con la CPU del ejemplo 2 (tiene el doble de velocidad de reloj).
- $$\frac{T_{CPU1}}{T_{CPU2}} = \frac{1 \times 5.44 \times P}{1 \times 8.8 \times (P/2)} = \frac{5.44 \times 2}{8.8} = 1.24$$
- La CPU2 es 24% más rápida que la CPU1.

## Ejemplo 3

- Suponer lo siguiente:
- Velocidad del reloj = 200 MHz (5 ns por ciclo).
- CPI = 1.1 con caché perfecto.
- 50% instrucciones aritmético/lógicas, 30% cargas y stores, 20% brincos.
- 10% de las instrucciones de acceso a la memoria tienen un castigo por falla de 50 ciclos.
- 1% de los accesos al caché de instrucciones tienen un castigo por falla de 50 ciclos.

## Ejemplo 3

- Calcular el CPI real.

## Ejemplo 3

- $CPI = CPI_{ideal} + CPI_{con\ detenciones\ de\ memoria}$ .
- $CPI = 1.1 +$  // **CPI ideal**  
 $(0.3 \times 0.1 \times 50) +$  // **Falla de datos**  
 $(1 \times 0.01 \times 50) =$  // **Falla de instrucciones**  
 $1.1 + 1.5 + 0.5 = 3.1$
- El procesador es 3.1 veces más lento que si tuviera caché perfecto.
- El 64.5% ( $2 / 3.1$ ) del tiempo el procesador está detenido por la memoria.

# AMAT

- Average Memory Access Time (tiempo promedio de acceso a memoria).
- $AMAT = \text{Tiempo de éxito} + \text{tasa de fallas} \times \text{castigo por falla}$ .
- El AMAT indica que el rendimiento es afectado por el tiempo de acceso a la memoria, incluyendo éxitos y fallas.

# Ejemplo

- Encontrar el AMAT para una CPU con los siguientes datos:
- Velocidad de reloj = 2 GHz
- Castigo por falla = 20 ciclos
- Tasa de fallas del caché = 5%
- Tiempo de acceso al caché = 1 ciclo.
- $AMAT = 1 + 0.05 \times 20 = 2$  ciclos
- $Periodo = 1 / 2e9 = 0.5$  ns
- $AMAT = 1$  ns

# Mejorar el rendimiento

- Opción 1: reducir el número de ciclos perdidos por detenciones de memoria.

$$\text{Ciclos de reloj detención memoria} = \frac{\text{accesos memoria}}{\text{programa}} \times \text{tasa de falla} \times \text{castigo por falla}$$

- castigo por falla = tiempo de acceso a la memoria x velocidad de reloj
- Opción 2: reducir el AMAT.
- AMAT = Tiempo de éxito + tasa de fallas x castigo por falla

# Mejorar el rendimiento

- Factores del rendimiento: tasa de fallas del caché, castigo por falla, tiempo de éxito.
- La tasa de fallas se reduce seleccionando un cache n-way con n mayor a la actual y/o un cache multinivel.
- El castigo por falla se reduce seleccionando una memoria con acceso más rápido a la actual.

# Mejorar el rendimiento

- El tiempo de éxito se puede reducir mediante un caché con capacidad para buscar en paralelo.

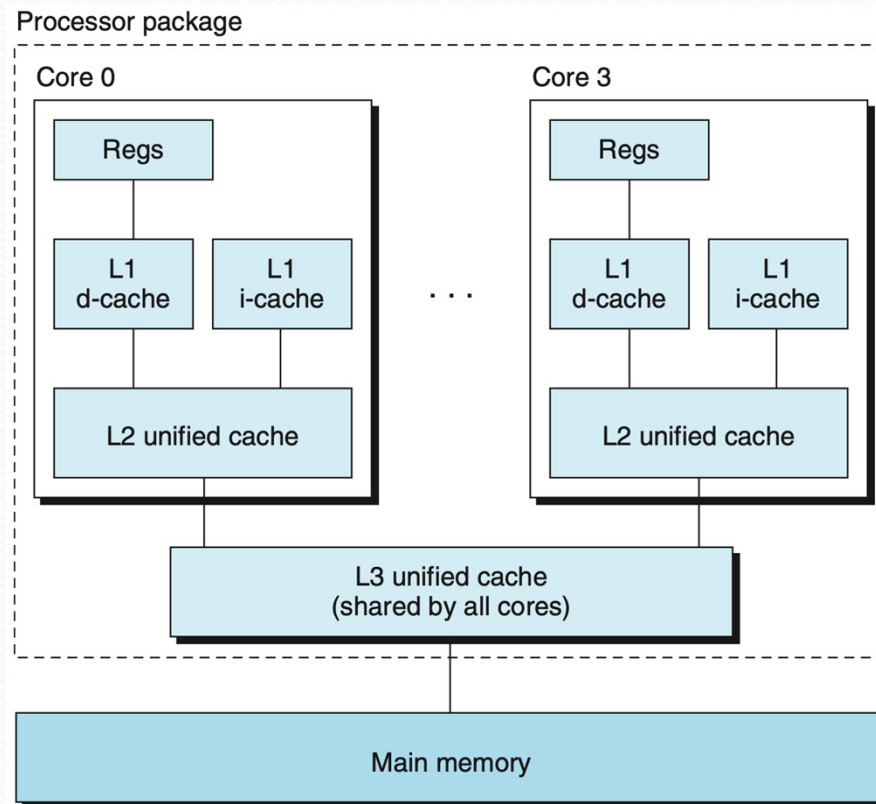
# AMAT vs asociatividad

| Tamaño<br>caché<br>(KB) | Asociatividad |       |       |       |
|-------------------------|---------------|-------|-------|-------|
|                         | 1-way         | 2-way | 4-way | 8-way |
| 1                       | 7.65          | 6.60  | 6.22  | 5.44  |
| 2                       | 5.90          | 4.90  | 4.62  | 4.09  |
| 4                       | 4.60          | 3.95  | 3.57  | 3.19  |
| 8                       | 3.30          | 3.00  | 2.87  | 2.59  |
| 16                      | 2.45          | 2.20  | 2.12  | 2.04  |
| 32                      | 2.00          | 1.80  | 1.77  | 1.79  |
| 64                      | 1.70          | 1.60  | 1.57  | 1.59  |
| 128                     | 1.50          | 1.45  | 1.42  | 1.44  |

# Cachés multinivel

- Los procesadores modernos tienen por lo menos dos niveles de caché (L1 y L2).
- El caché L2 se accesa cuando hay una falla en el caché L1.
- Si el caché L2 contiene el dato, el castigo por falla de L1 es el tiempo de acceso de L2 que es mucho menor que el tiempo de acceso a la memoria
- Si el dato no está en L1 ni en L2, se accesa la memoria y el castigo por falla es mayor.

# Intel Core i7



Fuente: CS-APP, p. 668

# Intel Core i7

| Cache type       | Access time (cycles) | Cache size ( $C$ ) | Assoc. ( $E$ ) | Block size ( $B$ ) | Sets ( $S$ ) |
|------------------|----------------------|--------------------|----------------|--------------------|--------------|
| L1 i-cache       | 4                    | 32 KB              | 8              | 64 B               | 64           |
| L1 d-cache       | 4                    | 32 KB              | 8              | 64 B               | 64           |
| L2 unified cache | 10                   | 256 KB             | 8              | 64 B               | 512          |
| L3 unified cache | 40–75                | 8 MB               | 16             | 64 B               | 8,192        |

**Fuente:** CS-APP, p. 668

# Ejemplo

- Se tiene una CPU con:
  - $\text{CPI} = 1$  con caché perfecto (sin fallas).
  - Velocidad de reloj = 5 GHz.
- Tiempo de acceso de la memoria = 100 ns.
- Tasa de fallas del caché primario = 2%.
- ¿Qué tan rápido es el sistema si se agrega otro nivel de caché con tiempo de acceso de 5 ns y capaz de reducir la tasa de fallas a la memoria a 0.5%?

# Ejemplo

- El castigo por falla es:

$$\text{castigo por falla} = \frac{\text{tiempo de acceso}}{\text{periodo de reloj}}$$

- Para la memoria principal el castigo es:

$$100 \text{ ns} / 0.2 \text{ ns} = 500 \text{ ciclos.}$$

- El CPI efectivo con un nivel de caché es:

$$\text{CPI total} = \text{CPI base} + \text{ciclos de detención por instrucción}$$

- Para la CPU con un nivel de caché:

$$\text{CPI} = 1 + 0.02 \times 500 = 11.0$$

# Ejemplo

- Con dos niveles de caché, una falla en el caché L1 se puede resolver por L2 o por la memoria.
- Si se resuelve en L2, el tiempo de acceso a L2 es el castigo por falla.
- En otro caso, el castigo por falla es la suma de los tiempos de acceso a L2 y a la memoria.
- El castigo por falla por un acceso a L2 es:

$$5 \text{ ns} / 0.2 \text{ ns} = 25 \text{ ciclos}$$

# Ejemplo

- El CPI efectivo para un caché de dos niveles:  
$$\text{CPI total} = \text{CPI base} + \text{detenciones primarias por instrucción} + \text{detenciones secundarias por instrucción}$$
- Para la CPU con dos niveles de caché:  
$$1.0 + 0.02 \times 25 + 0.005 \times 500 = 1 + 0.5 + 2.5 = 4$$
- Por lo tanto, la CPU con dos niveles de caché es más rápida que la CPU con un nivel en:  
$$11 / 4 = 2.75$$

# Tasa de fallas global / local

- Se introducen dos nuevos conceptos:
- Tasa de fallas global (global miss rate) es la fracción de referencias que fallan en todos los niveles.
- Tasa de fallas local (local miss rate) es la fracción de referencias que fallan en un nivel.
- El caché primario es mas pequeño y con menor tiempo de acceso que los secundarios.

# Ejemplo

- Suponer que de 1000 referencias a la memoria, hay 40 fallas en L1 y 20 fallas en L2.
- Calcular las tasa de fallas.
- Para L1: tasa de fallas =  $40 / 1000 = 4\%$ .
- Para L2:
  - Tasa de fallas local =  $20 / 40 = 50\%$ .
  - Tasa de fallas global =  $20 / 1000 = 2\%$ .

# Ejemplo

- Con los datos anteriores, suponer lo siguiente:
  - El castigo por falla de L2 a la memoria es de 200 ciclos.
  - El tiempo de acceso (hit time) de L2 es de 10 ciclos.
  - El tiempo de acceso de L1 es de 1 ciclo.
  - Hay 1.5 referencias a la memoria por instrucción.
- Calcular el AMAT.

# Ejemplo

- $AMAT = Hit\ time_{L1} + Miss\ rate_{L1} \times (Hit\ time_{L2} + Miss\ rate_{L2} \times Miss\ penalty_{L2})$
- $AMAT = 1 + 0.04 \times (10 + 0.5 \times 200) =$
- $1 + 0.04 \times 110 = 5.4\text{ ciclos}$

# Resumen

- El rendimiento de un sistema de cachés se puede medir de dos formas:
  - Midiendo el tiempo de ejecución contando los ciclos que se pierden por accesos a la memoria.
  - Midiendo el tiempo promedio de acceso a la memoria (AMAT).
- El rendimiento se puede mejorar afectando los factores del rendimiento: la tasa de fallas, el castigo por falla, el tiempo de éxito.

# Resumen

- Para reducir la tasa de fallas, los procesadores modernos utilizan memorias caché multinivel.