19 May 2020 12:02

datum

DATUM

3.3 Main memory updating

→ Cache coherency

→ reads / writes

→ Statistics: reads are dominating
- Instructions are read.
- Data are read / written
- Writes represent ≤ 10% of mem. accesses.

Amdahl's Law: make the common case faster.

We cannot neglect writes!

a) Write Through: write in the cache and MM at the same time

b) Write Back: write in the cache only
- Update in MM when the block is replaced
- 1 bit dirty / clean

What happens in case of a Write Miss?

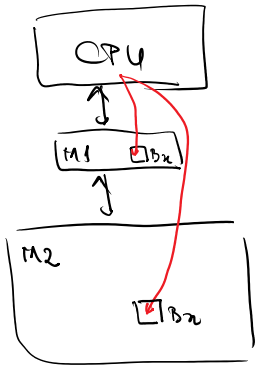
In the case of read misses we always allocate!

α) Write Allocate

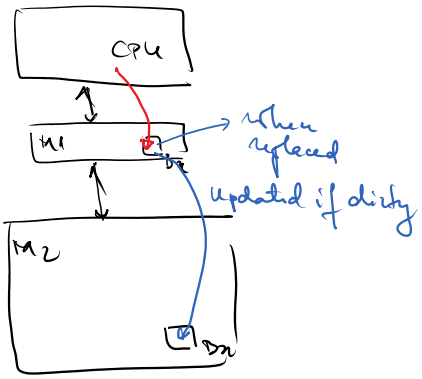
β) Write No-Allocate

2 Natural policies → WT + WNA + Write in MM
→ WB + WA + Write in Cache

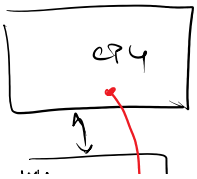
WT hit



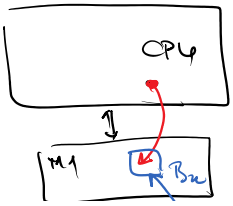
WB hit



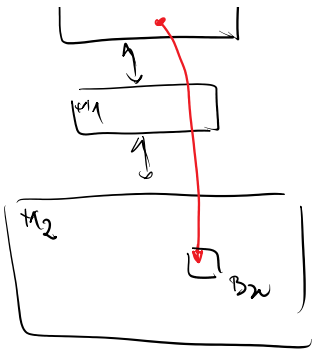
WT + WNA (Miss)

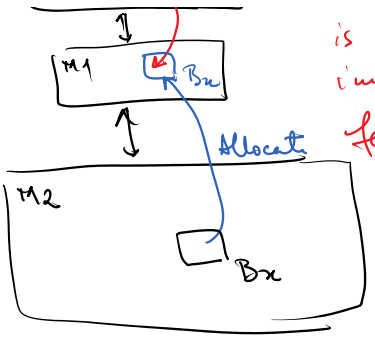


WB + WA



This policy is always implemented





is always implemented for reads!

3.4 Memory hierarchy performance

$$\text{CPU time} = (\text{CPU execution clock cycles} + \text{Memory stall clock cycles}) \times \text{Clock cycle time}$$

$$\text{Memory stall clock cycles} = \frac{\text{Reads}}{\text{Program}} \times \text{Read Miss Rate} \times \text{Read Penalty} + \frac{\text{Writes}}{\text{Program}} \times \text{Write Miss Rate} \times \text{Write Miss Penalty}$$

$$\text{Memory stall clock cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss Rate} \times \text{Miss Penalty}$$

$$\text{CPU time} = IC \times \left(\text{CPI}_{\text{execution}} + \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss Rate} \times \text{Miss Penalty}\right) \times \text{Clock cycle time}$$

Misses / Instruction

$\text{CPI}_{\text{execution}}$: supplementary relative to AMAT, ≥ 1

Example 1 $\text{CPI}_{\text{execution}} = 8.5$ c.c

Memory references / Instruction = 3

Miss Rate = 11% Miss Penalty = 6cc.

Influence of cache imperfection as %?

$$\text{CPU time}_{\text{perfect}} = IC \times 8.5 \times \text{Clock cycle time}$$

$$\text{CPU time} = IC \times (8.5 + 3 \times 0.11 \times 6) \times \text{Clock cycle time} \simeq IC \times 10.5 \times \text{Clock cycle time}$$

200 cc. gen. ... 8.5 c.c. perf.

x ... 100

$$x = \frac{200}{8.5} = 23.5\%$$

Example 2

- 64KiB cache
- Clock cycle time = 20ns
- $\frac{\text{Memory accesses}}{\text{Instruction}} = 1.3$
- Clock cycles / Instruction for a perfect cache = 1.5 c.c.
- For S.A. => clock frequency degradation with 8.5%

(A) DM cache Miss Rate = 3.5%

(B) 2-way SA Miss Rate = 3%

AMAT = Average Memory Access Time

Miss Penalty for both solutions = 200 ns

(B) 2-way SA Miss Rate = 3%

Which solution is better ?

AMAT = Average Memory Access Time

- Miss Penalty for both solutions = 200 ns (usually is expressed as c.c.)

$AMAT = \underbrace{t_{ac}}_{1\,c.c.} \times \text{Hit Rate} + (t_{ac} + \underbrace{\text{Miss Penalty}}_{\text{(time miss penalty)}}) \times \text{Miss Rate} =$ (Time!)

$= t_{ac} + \text{Miss Penalty} \times \text{Miss Rate}$

$AMAT_A = 20\,ns + 0.039 \times 200\,ns = 27.8\,ns$

$AMAT_B = 21.7\,ns + 0.03 \times 200\,ns = 27.7\,ns$

2-way SA is better

$\text{Clock cycle time}_B = 1.085 \times 20\,ns = 21.7\,ns$

Miss Penalty (time)

$CPU\,time_A = IC \times (CPI_{ideal} + \frac{\text{Memory acc}}{\text{Instr}} \times \text{Miss Rate} \times \text{Miss Penalty}) \times \text{Clock cycle time} =$

$= IC \times (CPI_{ideal} \times \text{Clock cycle time} + \frac{\text{Memory acc}}{\text{Instr}} \times \text{Miss Rate} \times \text{Time Miss Penalty}) =$

$= IC \times (1.5 \times 20\,ns + 1.3 \times 0.039 \times 200\,ns) = 40.14\,ns \times IC$

$CPU\,time_B = IC \times (1.5 \times 21.7\,ns + 1.3 \times 0.03 \times 200\,ns) = (7.8 + 32.55)\,ns \times IC$

$= 40.35\,ns \times IC$

DM is better!!

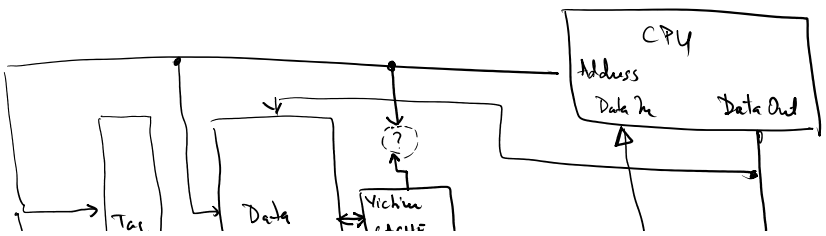
- Cache performance is decisively influenced by misses!

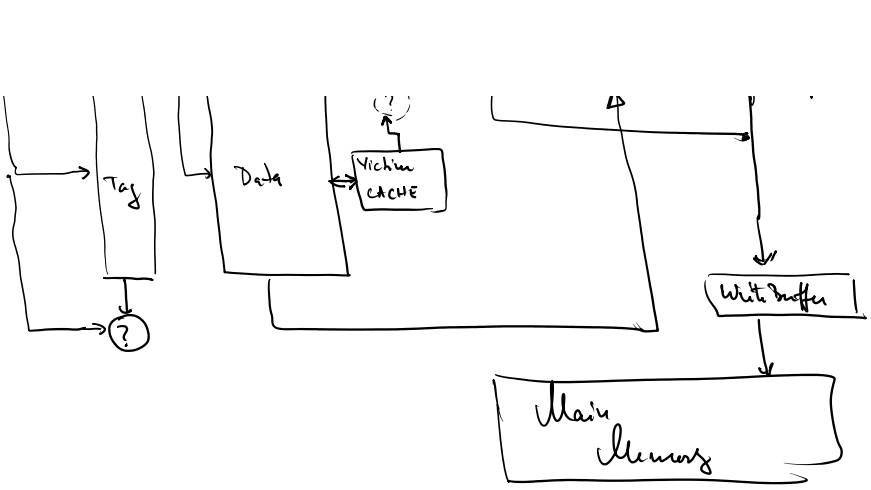
What are the sources of misses ?

- Compulsory → empty (cold start) misses ⟼ bigger block size
- Conflict → DM (collision misses) ⟼ higher SA (maybe FA) ⟼ degrades the t_{ac}
- Capacity → Too many references ⟼ Compiler optimization (loop tiling)

3.5 Cache performance optimization

- Compiler optimization (software)
- Hardware ① – bigger block size
  ② – higher associativity
  ③ – Victim caches
  ④ – Pseudo-associative caches




Tag
Data
Victim CACHE
?
?
Write Buffer
Main Memory