Chapter 2
(Starting with Appendix B)

Instructor: Josep Torrellas

CS433

Memory Hierarchy Design

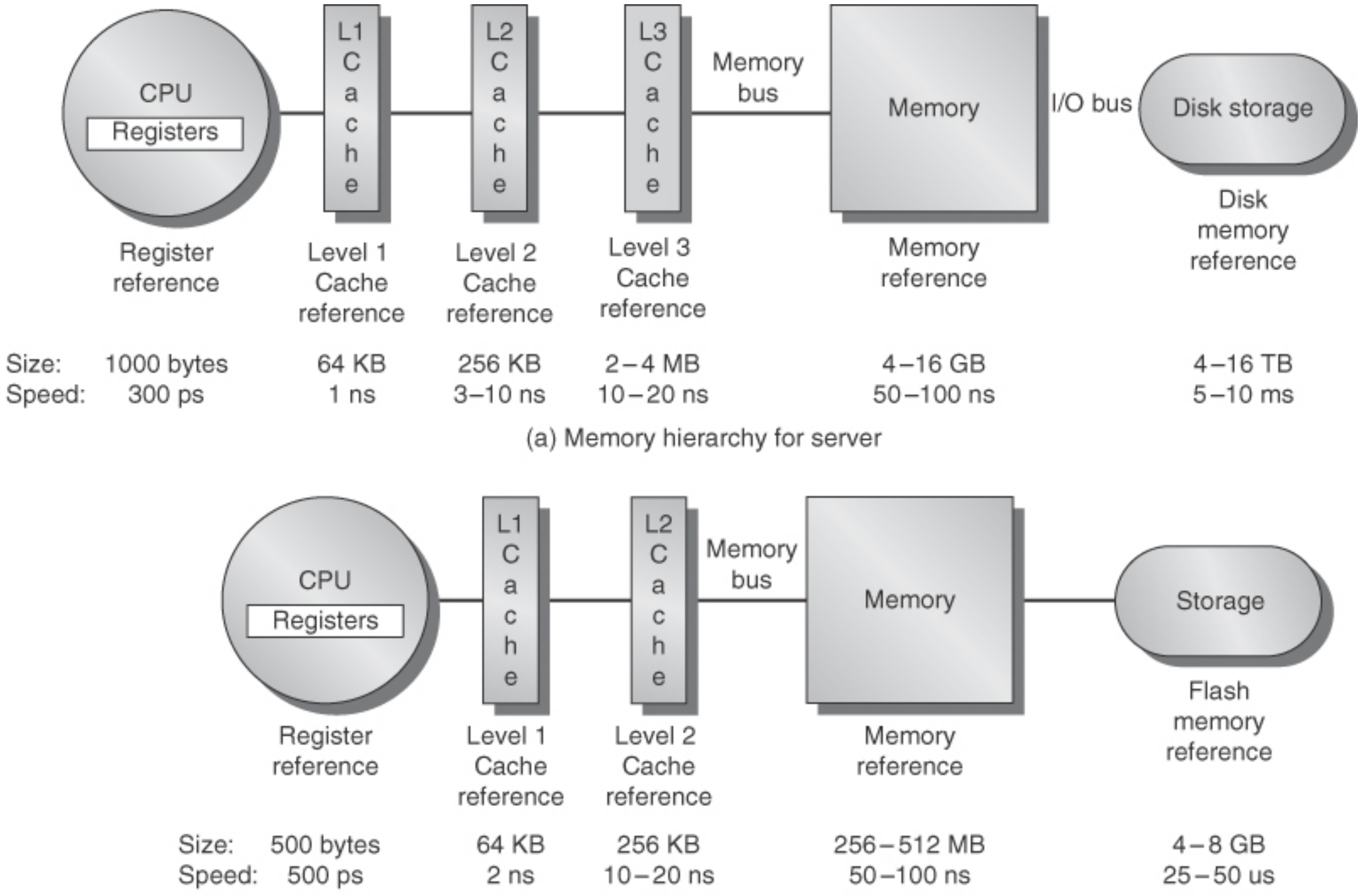
1980 microprocessor → no cache

1995 → 2 level of caches

Why ?

- Programmers want more memory
- Principle of locality
- Processor - memory performance gap
- Smaller hardware is faster
- faster ⇒ more expensive

Memory Hierarchy



(a) Memory hierarchy for server

(b) Memory hierarchy for a personal mobile device

- Figure 2.1

Cache Blocks (Lines)

Block → minimum unit of information that can be present in the cache

Hit / Miss

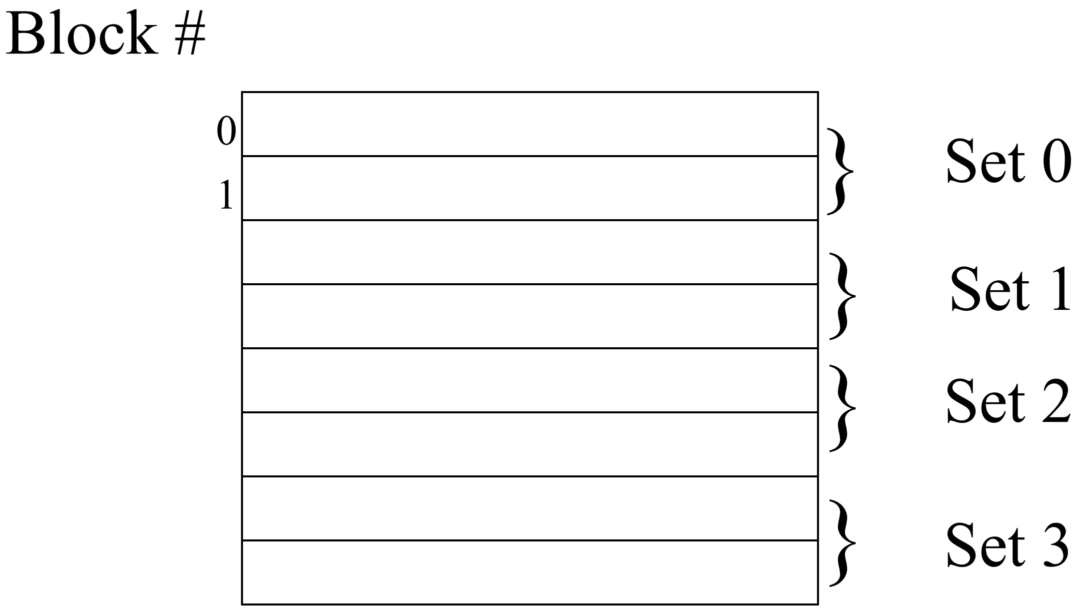
Block placement

Block identification

Block replacement

Write strategy

Block Placement



Block address in memory : 10
Direct mapped : (Block addr) Mod (# blocks in cache)
Block # : 10 MOD 8 = 2
Set Associative : (Block Addr) Mod (# sets in cache)
Set # : 10 MOD 4 = 2
Fully Associative : Anywhere

Block Placement

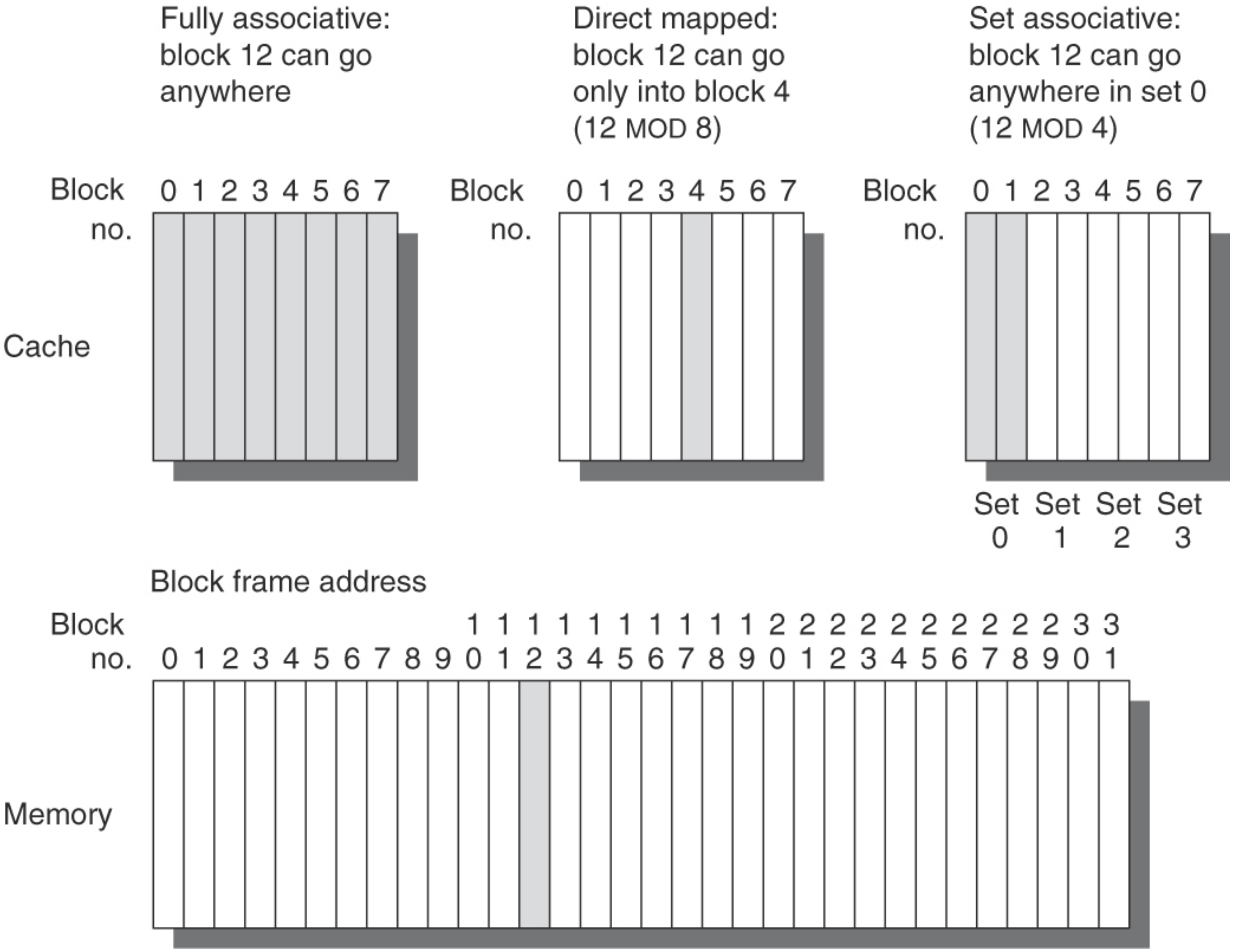


Figure B.2

Block Identification

<table>
<tr><td colspan="2">Block Address</td><td rowspan="2">Block
Offset</td></tr>
<tr><td>Tag</td><td>Index</td></tr>
</table>

- Index and block offset not stored
- Tag
- Valid bit
- Associativity ↑ ⇒ Size of tag ↑

Block Replacement

- Random
- Least recently used (LRU)

Write Policy

Writes form 25% of data cache traffic

7% of total memory traffic

- Make common case fast ⇒ optimize for reads
- Also processor waits for reads to complete; need not wait for writes
- Easy to make reads fast :

  read data in parallel with reading and comparing tag
- Writes : Block cannot be modified until tag matches
- Size of write

Two Basic Write Policies

- Write through
- Write back
- Dirty bit saves writes
- Write back ⇒ less memory bandwidth
- Write through ⇒ read miss never results in write to lower level. Easier to implement
- Write stall during write through
  - use a write buffer

Write Miss Policies

On a write miss :

- Write allocate (fetch on write)
- No-write allocate (write around)

- Write miss policy and write policy are independent
- Write back caches generally use write allocate
- Write through caches often use no write allocate

Cache Performance

Avg Memory Access Time = Hit time + Miss rate * Miss Penalty

Example : Compare a 32-KB unified cache with a 16KB I-cache and 16 KB D-cache.

Given : Miss rates are 1.99% , 0.64% and 6.47% respectively . 75% of memory accesses are instruction references.

Hit 1 clock cycle

Miss 50 clock cycles

Load/Store access in unified cache : adds 1 extra clock cycle due to structural hazard

For the split cache,overall miss rate is

(75% * 0.64%) + (25% * 6.47%) = 2.10%

Average memory access time split

= 75% (1+ 0.64% * 50) + 25% * (1+6.47% *50)

= 2.05

Avg memory access time unified

= 75% * (1+1.99% * 50) + 25% *(1+**1**+1.99% * 50)

= 2.24

$$\begin{matrix}\text{CPU}\\ \text{time}\end{matrix} = \left[\begin{matrix}\text{CPU Execution}\\ \text{Clock cycles}\end{matrix} + \begin{matrix}\text{Memory stall}\\ \text{Clock cycles}\end{matrix}\right] * \begin{matrix}\text{clock cycle}\\ \text{time}\end{matrix}$$

Assumption : All memory stalls are due to cache misses.

$$\begin{aligned}\left[\begin{matrix}\text{Memory stall}\\ \text{clock cycles}\end{matrix}\right] &= \text{Reads} * \text{Read miss rate} * \text{Read miss Penalty}\\ &\quad + \text{Writes} + \text{Write miss Rate} * \text{Write miss Penalty}\\ &= \text{Memory accesses} * \text{Miss rate} * \text{Miss Penalty}\end{aligned}$$

Consider a machine with :

Cache miss penalty = 50 cycles

$\text{CPI}_{\text{execution}} = 2.0$

Miss Rate = 2%

Memory References per inst = 1.33

$$\text{CPU time} = \text{IC} * (\text{CPI}_{\text{execution}} + \frac{\text{Memory stall clock cycles}}{\text{Instruction}}) * \text{Clock cycle time}$$

$$= \text{IC} * (2.0 + (1.33 * 2\% * 50)) * \text{Clock cycle time}$$

$$= \text{IC} * 3.33 * \text{Clock Cycle time}$$

Impact of cache: 1.67x

If no cache: 68.5 30x

If perfect CPIexecution: 2.33x

Cache misses have double - barreled impact on a CPU with low CPI and fast clock

1. Lower CPI $_{\text{execution}}$ → relative impact of FIXED number of cache miss cycles increases

2. Identical memory hierarchy → CPU with faster clock sees more stall cycles

Note: in ooo execution processors: part of the memory access latency is overlapped with computation!