CS/COE1541: Introduction to Computer Architecture

Dept. of Computer Science
University of Pittsburgh

http://www.cs.pitt.edu/~melhem/courses/1541p/index.html

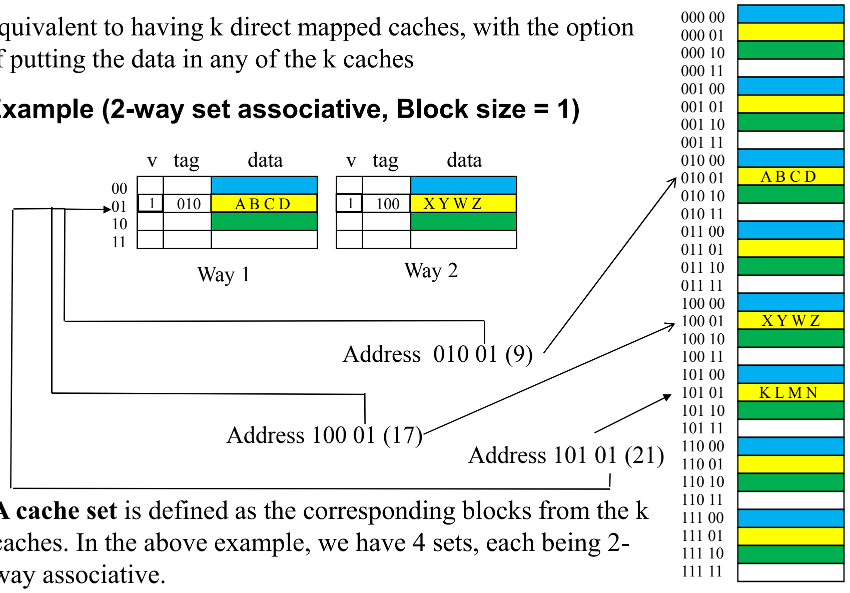
Chapter 5: Exploiting the Memory Hierarchy
Lecture 3

Lecturer: Rami Melhem

Decreasing miss ratio with k-way associativity

Equivalent to having k direct mapped caches, with the option of putting the data in any of the k caches

Example (2-way set associative, Block size = 1)

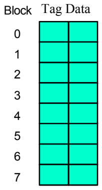


A cache set is defined as the corresponding blocks from the k caches. In the above example, we have 4 sets, each being 2-way associative.

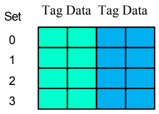
Degree of associativity can vary from 1-way to full

Example: a cache with 8 data blocks

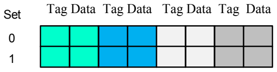
One-way set associative
(direct mapped)



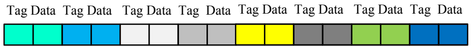
two-way set
associative



4-way set associative



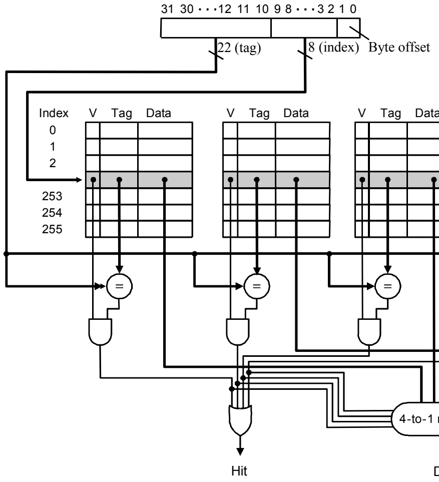
8-way set associative
(fully associative)


Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data

Need a block replacement policy to determine which block to evict in case a new block is to be cached into a full set.

- *LRU replacement → evict the least recently used block*

Example: 4KB cache, 4-way associative, block size = 1 word

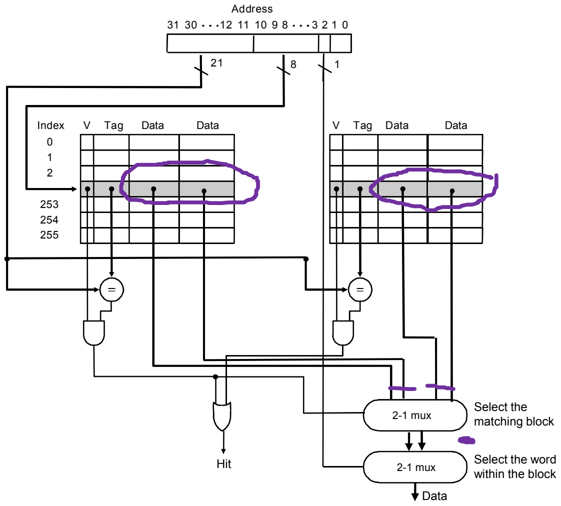


Equivalent to four, 1KB, direct mapped caches.

Each 1KB direct mapped cache has 256 = 2^8 blocks.
(each block = 2^0 = 1 word)
→
Byte offset = 2 bits
Block offset = 0 bits
Index = 8 bits
Tag = remaining 22 bits

- Need to compare tags in all four ways
- If one matches, return the corresponding stored data,
- May use block size larger than one word.

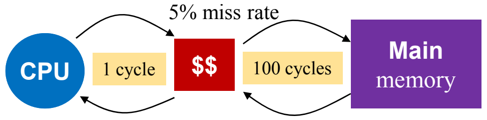
Example: 4KB cache, 2-way associative, block size = 2 words



Equivalent to two, 2KB, direct mapped caches.

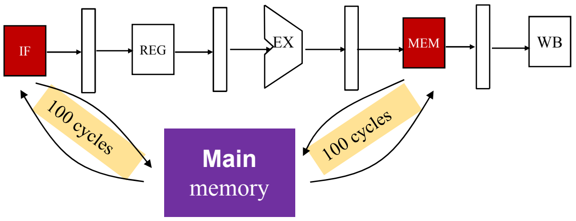
Each 2KB direct mapped cache has 256 = 2^8 blocks. (each block = 2^1 = 2 word)
→
Byte offset = 2 bits
Block offset = 1 bits
Index = 8 bits
Tag = remaining 21 bits

Performance analysis



Cache access time = 1 cycle
Memory access time = 100 cycles
Average memory access time
= 1 + miss rate * miss penalty
= 1 + 0.05 * 100 = 6 cycles

Effect of cache misses on pipelined execution: Pipeline will stall on a miss to the Instruction or Data caches – thus increasing the CPI.

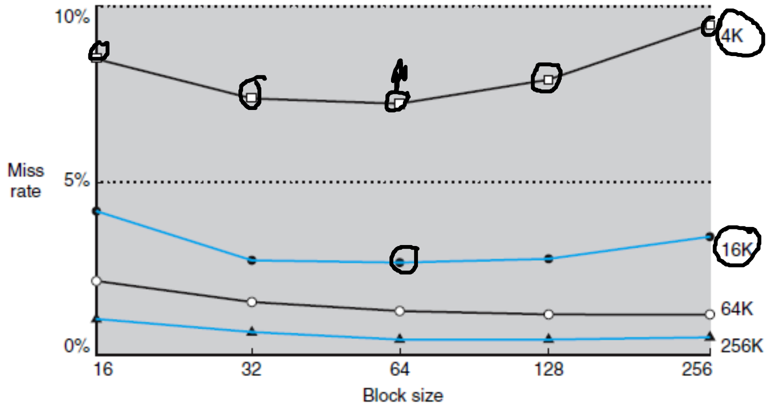


Example: Assume a pipeline where 36% of the instructions are lw or sw
Assume the instruction cache miss rate = 2% and data cache miss rate = 4%

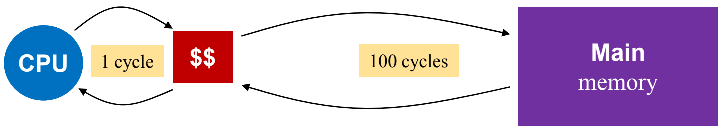
Effective miss rate = 0.02 + 0.36 * 0.04 = 0.0344 misses per instruction
CPI = CPI ignoring cache misses + 0.0344 * 100 = CPI no misses + 3.376 cycle

Effect of block size on miss rate

- Because of access locality, larger blocks decreases the miss rate.
- For random accesses, larger blocks increases miss rate
- Memory accesses are a mix of sequential and non-sequential access,
- Hence, larger blocks, up to a given optimum size, decreases the miss rate.



Effect of cache size on miss rate and hit time



CPU

1 cycle

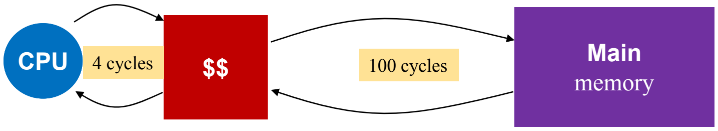
$$

100 cycles

Main memory

Small on-chip cache is fast

- If miss rate = 10%,
  average memory access time = 1 + 0.1 * 100 = 11 cycles.



CPU

4 cycles

$$

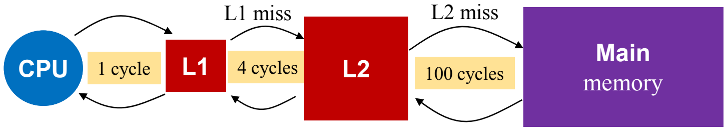
100 cycles

Main memory

With better integration, may have larger on-chip caches

- Larger cache leads to **lower miss rate** but **larger hit time**
- If miss rate = 8% and hit time = 4 cycles, then
  average memory access time = 4 + 0.08 * 100 = 12 cycles.

Improving performance with multilevel caches


L1 miss

L2 miss

CPU

1 cycle

L1

4 cycles

L2

100 cycles

Main memory

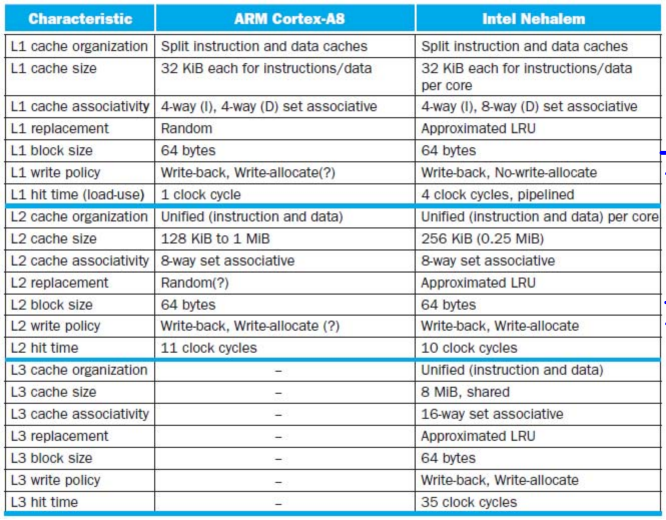
Example (a 1GHz machine → 1ns cycles)

- L1 hit time = 1 cycle,
- L2 hit time = 4 cycles
- DRAM access time = 100ns (100 cycles)
- L1 miss rate = 10%
- 70% of L1 misses are hits in L2 (hence 30% are misses)

- Average memory access time = $1 + 0.1 * (\underline{4} + \underline{0.3} * \underline{100}) = 4.4$ cycles.

Note that all L1 misses are looked up in L2 and only the L2 misses are sent to memory

Multilevel On-Chip Caches for Cortex-A8 and Core-i7



Characteristic	ARM Cortex-A8	Intel Nehalem
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KiB each for instructions/data	32 KiB each for instructions/data per core
L1 cache associativity	4-way (I), 4-way (D) set associative	4-way (I), 8-way (D) set associative
L1 replacement	Random	Approximated LRU
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate(?)	Write-back, No-write-allocate
L1 hit time (load-use)	1 clock cycle	4 clock cycles, pipelined
L2 cache organization	Unified (instruction and data)	Unified (instruction and data) per core
L2 cache size	128 KiB to 1 MiB	256 KiB (0.25 MiB)
L2 cache associativity	8-way set associative	8-way set associative
L2 replacement	Random(?)	Approximated LRU
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate (?)	Write-back, Write-allocate
L2 hit time	11 clock cycles	10 clock cycles
L3 cache organization	–	Unified (instruction and data)
L3 cache size	–	8 MiB, shared
L3 cache associativity	–	16-way set associative
L3 replacement	–	Approximated LRU
L3 block size	–	64 bytes
L3 write policy	–	Write-back, Write-allocate
L3 hit time	–	35 clock cycles

The 3C cache miss model

Cache misses can be classified into one of the following:

1) Compulsory misses

- Result from referencing a block for the first time
- Unavoidable

2) Capacity misses

- Result from the limited cache size (assuming a fully associative cache)
- May be avoided if a larger cache is used

3) Conflict misses

- Result from hash collisions if cache is not fully associative

Remember to send any question that you had while listening to this presentation to me before class time. I will answer the question that I receive in class.