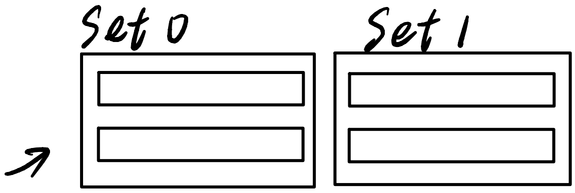
Final RC Part 3

VE370FA22 TA Jiajun Gu

N-way Set Associative Cache



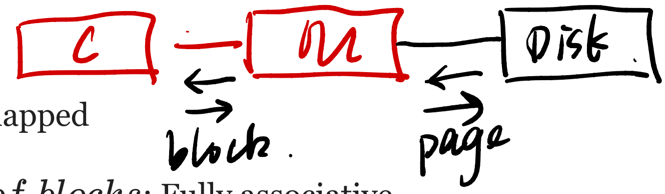
Motivation: Reduce miss rate

- Each set contains n blocks
- A main memory block can use *any of the blocks* within the corresponding set
- $Set\ Index = (Block\ address)\%(number\ of\ sets\ in\ cache)$
- $Block\ Offset = word\ offset + byte\ offset$

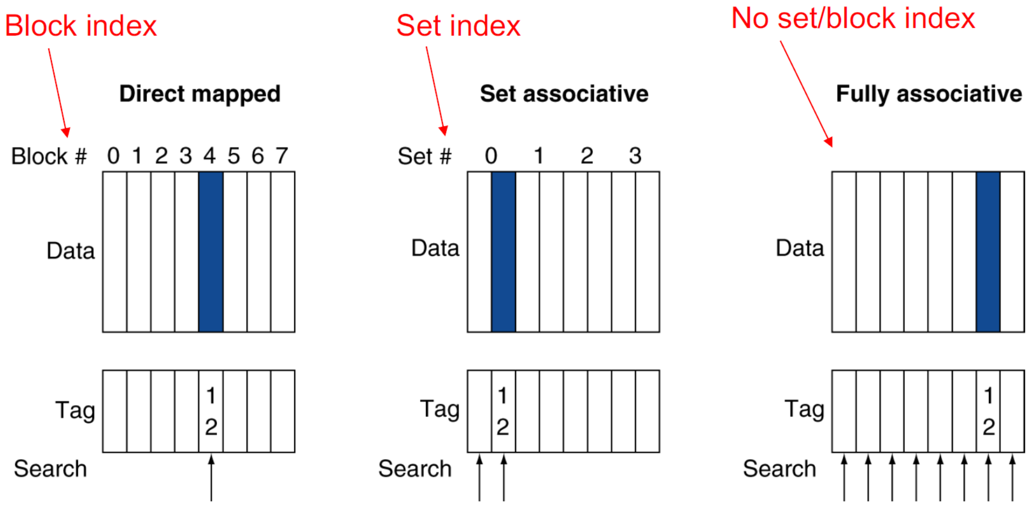
CAUTION

- n-way associative means *each set contains n blocks*, **not** *the cache contains n sets*!
- The basic element for operation is still **block** (compare with **page** in VM)

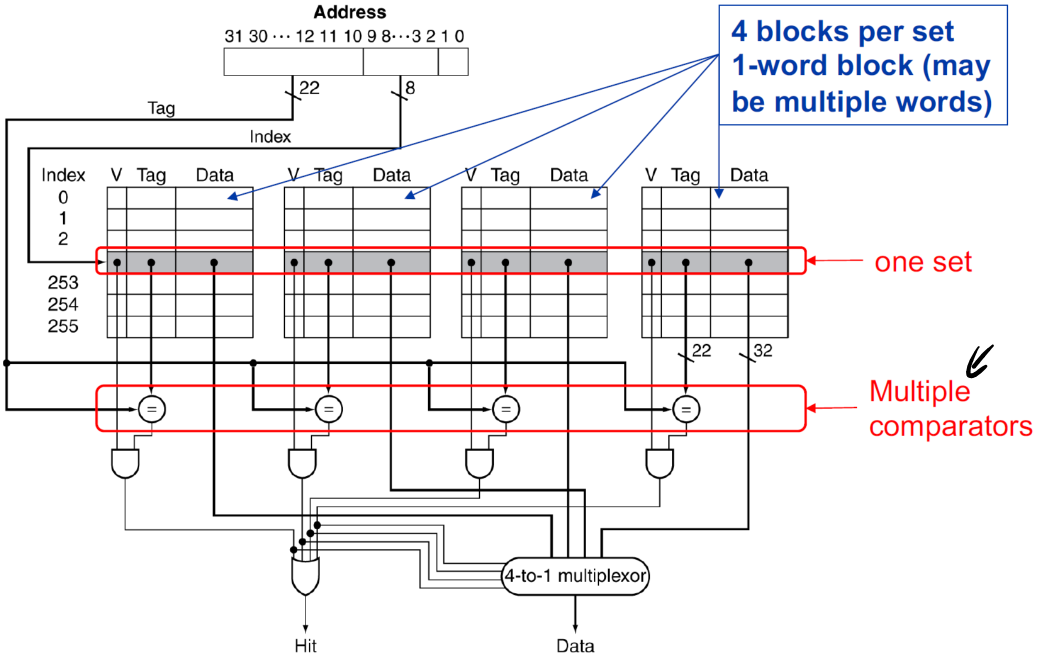
Special case



- $n = 1$: Directed mapped
- $n = total\ num\ of\ blocks$: Fully associative



Tradeoff: We may need to search n times in a set / use specific circuit structure (comparators) to locate a block in it. -> ***Extra hit time / hardware***



Exercise:

1. Suppose memory size is 64 words, use 4-way associative cache of 16 words size, 2 words per block.

- Memory
  - memory word address: $\log_2(64) = 6$ bits
  - memory byte address: $6 + 2 = 8$ bits
- Cache
  - byte offset: 2 bits
  - word offset: 1 bit
  - set index: 1 bit
    - number of blocks: $16/2 = 8$
    - number of sets: $8/4 = 2$
  - tag: $8 - 2 - 1 - 1 = 4$ bits

2. Given a 4-way associative cache with 2K 8-word blocks, and a 32-bit byte address 0x810023FE, show the set index and tag for this byte address. (2K=2^{11})

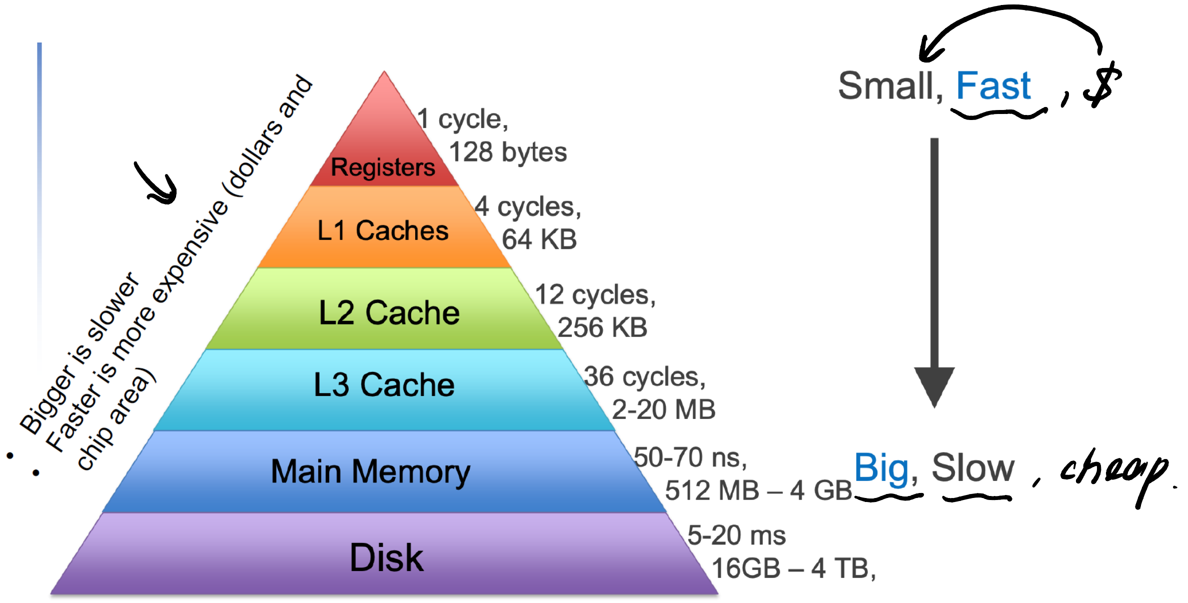
Tag	Set Index	Word Offset	Byte Offset
31:14	13:5	4:2	1:0

Replacement Policy

- Direct mapped: no other choices
- Set associative:
  - Prefer non-valid (empty) entry, if there is one
  - Otherwise, choose to replace a block in the set
- Policy:
  - **Least-recently used (*LRU*)** ←
    - Choose the one unused for the longest time
    - Simple for 2-way, manageable for 4-way, too hard beyond that
  - Random ←
    - Gives approximately the same performance as LRU **for high associativity**

Example: *(Check by yourself)* T11 p35-41

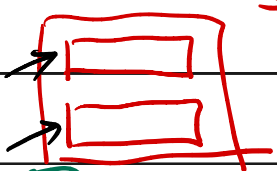
Multilevel Caches



- Primary (L-1) cache attached to CPU
  - Small, but fast
  - Focus on **minimal hit time** because miss penalty is smaller
  - Smaller block size L0 Total cache size is smaller.
    - Reduce search (hit) time
    - Reduce miss penalty (less time to fetch)

- Level-2 (secondary) cache services misses from primary cache
  - Larger, slower, but still faster than main memory
  - Focus on **low miss rate** to avoid main memory access
  - Higher associativity and Larger. block size

Memory Unit	**Function (likely)**	**SRAM/DRAM**
L1 cache	A cache for a cache	SRAM
L2 cache	A cache for main memory	SRAM
Main Memory	A cache for disks	DRAM
TLB	A cache for a page table	SRAM

Modification	**Effect on miss rate**	**Negative Effect**
1 Increase cache size	Decrease *capacity misses* 	Increase cost/access time (if use other materials)/structures
2 Increase associativity	Decrease *conflict misses*	Increase complexity/hit time
3 Increase block size	Decrease *compulsory misses*; May increase *conflict misses* (e.g., cache size fixed)	Increase miss penalty

More on miss types: Types of Cache Misses - GeeksforGeeks

Performance Evaluation (multilevel cache):

CPI

If not specified, base CPI should contain the CPI used for hit.

$$Miss\ penalty\ (\text{main memory}) = \left\lceil \frac{Main\ Memory\ Access\ Time}{Clock\ Cycle\ Time} \right\rceil$$

With L-1 cache:

$$Effective\ CPI = Base\ CPI + Miss\ penalty\ (\text{main memory}) \times Miss\ Rate$$

With L-2 cache:

$$Access\ Time\ (\text{L-1 to L-2}) = \text{L-1}\ Miss\ penalty$$

$$Miss\ penalty\ (\text{L-1 to L-2}) = \left\lceil \frac{Access\ Time\ (\text{L-1 to L-2})}{Clock\ Cycle\ Time} \right\rceil$$

$$Miss\ penalty\ (\text{L-1 miss \&\& L-2 miss}) = Miss\ penalty\ (\text{L-1 to L-2}) + Miss\ penalty\ (\text{main memory})$$

$$CPI = Base\ CPI$$

$$+\text{L-1}\ Miss\ rate \times Miss\ penalty\ (\text{L-1 to L-2}) \times \text{L-2}\ hit\ rate$$

$$+\text{L-1}\ Miss\ rate \times Miss\ penalty\ (\text{L-1 miss \&\& L-2 miss}) \times \text{L-2}\ miss\ rate$$

Performance ratio = CPI (with L-1 cache) / CPI (with L-1 & L-2)

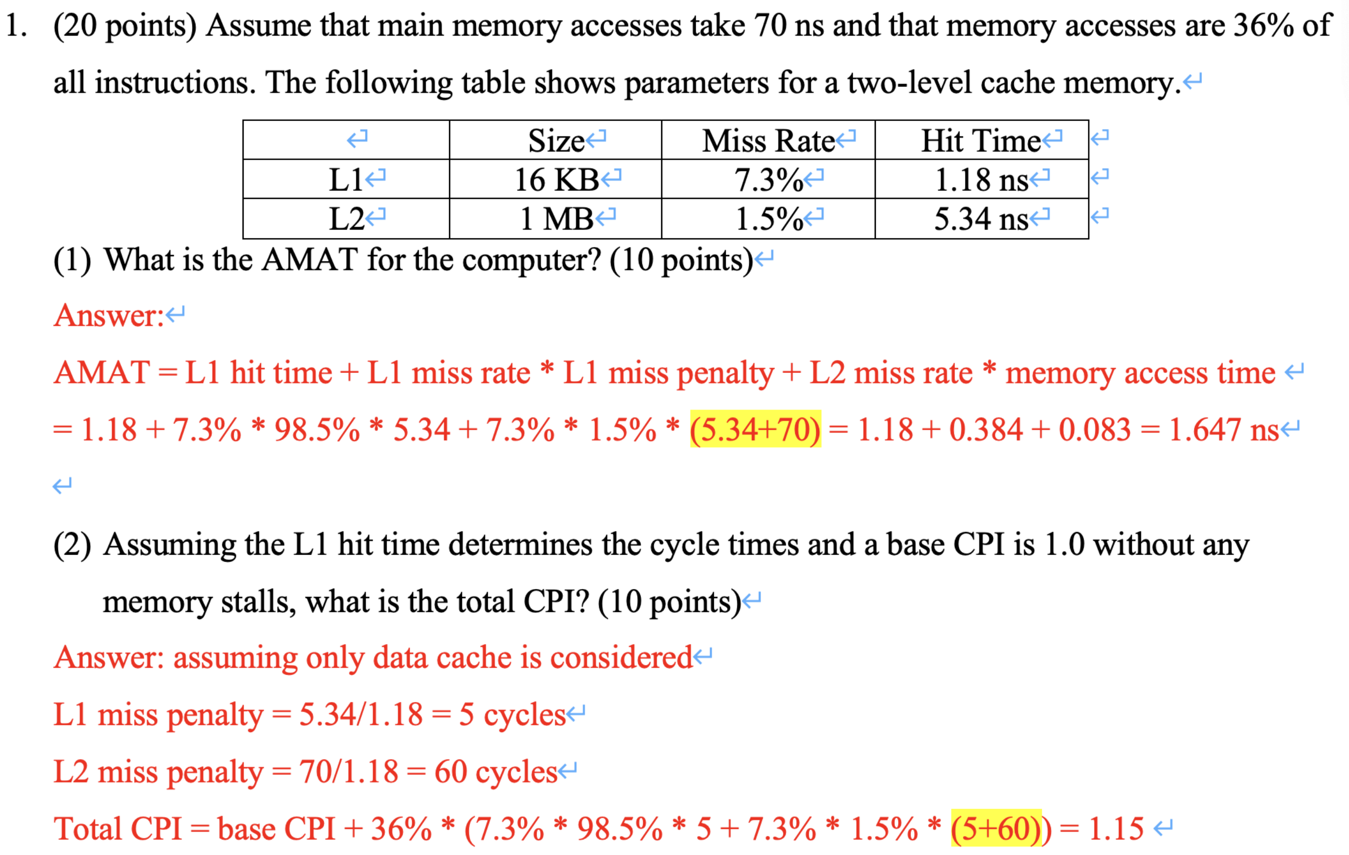
AMAT

$$AMAT\ (one\ level) = Hit\ Time + Miss\ Rate \times Miss\ Penalty$$

*If not specified, hit time should also be spent when a miss occurs. (Think about this: why **Hit Time** is not multiplied by **Hit Rate**?)*

$$AMAT\ (two\ levels) = L1\ Hit\ Time$$

$$+L1\ Miss\ Rate \times (L1\ Miss\ Penalty + L2\ Miss\ Rate \times L2\ Miss\ Penalty)$$



1. (20 points) Assume that main memory accesses take 70 ns and that memory accesses are 36% of all instructions. The following table shows parameters for a two-level cache memory.

	Size	Miss Rate	Hit Time
L1	16 KB	7.3%	1.18 ns
L2	1 MB	1.5%	5.34 ns

(1) What is the AMAT for the computer? (10 points)

Answer:

AMAT = L1 hit time + L1 miss rate * L1 miss penalty + L2 miss rate * memory access time

= 1.18 + 7.3% * 98.5% * 5.34 + 7.3% * 1.5% * (5.34+70) = 1.18 + 0.384 + 0.083 = 1.647 ns

(2) Assuming the L1 hit time determines the cycle times and a base CPI is 1.0 without any memory stalls, what is the total CPI? (10 points)

Answer: assuming only data cache is considered

L1 miss penalty = 5.34/1.18 = 5 cycles

L2 miss penalty = 70/1.18 = 60 cycles

Total CPI = base CPI + 36% * (7.3% * 98.5% * 5 + 7.3% * 1.5% * (5+60)) = 1.15

Other notes (may not be useful)

What does it mean for an architecture to be called a load/store architecture?

A. Load and Store instructions are supported by the ISA.

B. Load and Store instructions can also perform arithmetic instructions on data in memory.

C. Data must first be loaded into a register before it can be operated on.

D. Every load must have an accompanying store at some later point in the program.

Reference

1. VE370 2021FA, Final RC III
2. ECE3700J 2022FA, Lecture Slides