MICROPROCESSOR AND COMPUTER ARCHITECTURE

UNIT-3

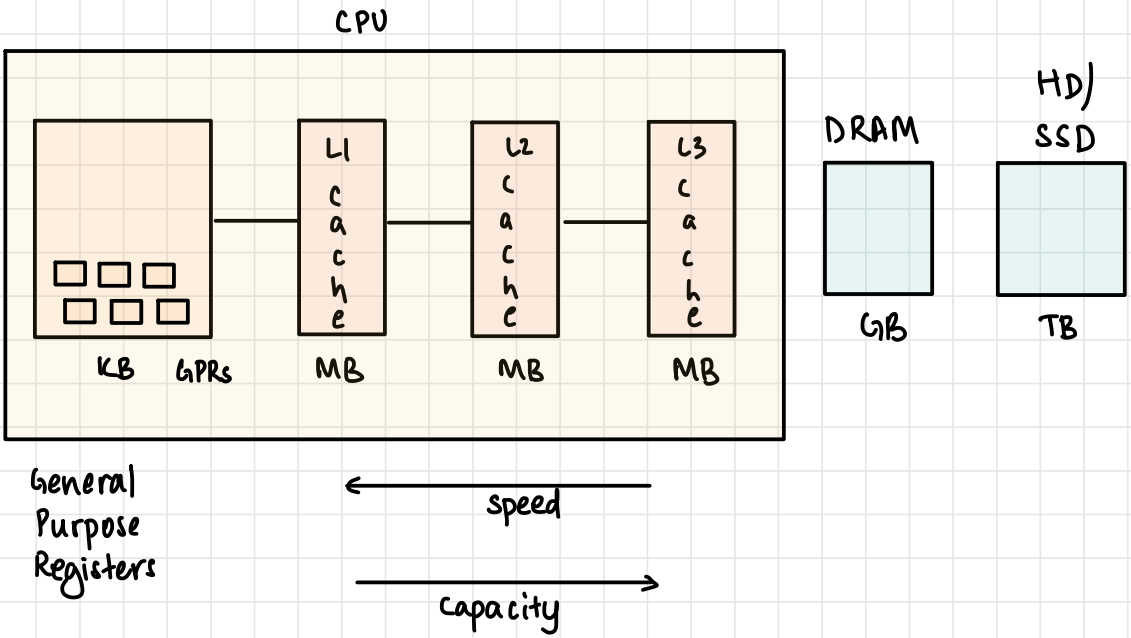
memory

feedback/corrections: vibha@pesu.pes.edu

VIBHA MASTI

MEMORY

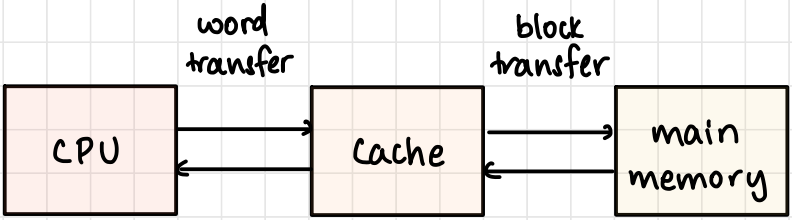
- Processor speed vs memory – gap
- Registers: reside in CPU, faster to fetch from / write to; limited in number



Locality of Reference

1) Temporal Locality
   - set of instr likely to be referenced soon
   - eg: looping

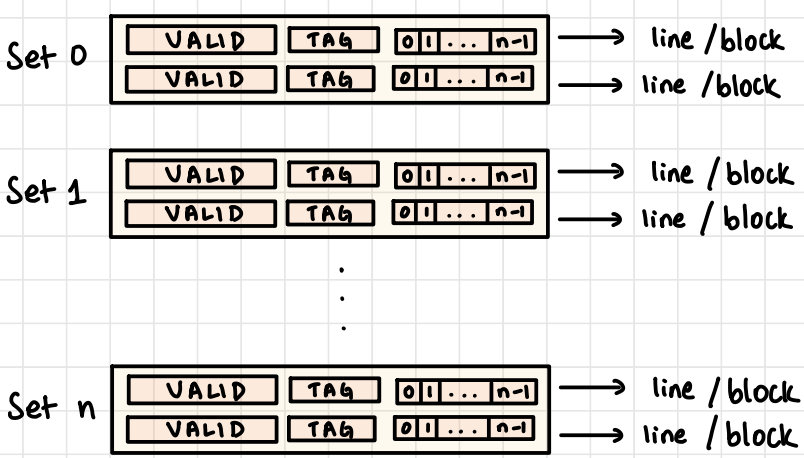
2) Spatial locality
   - if fetching from memory, fetch from other nearby locations
   - eg: array elements



Cache mapping technique

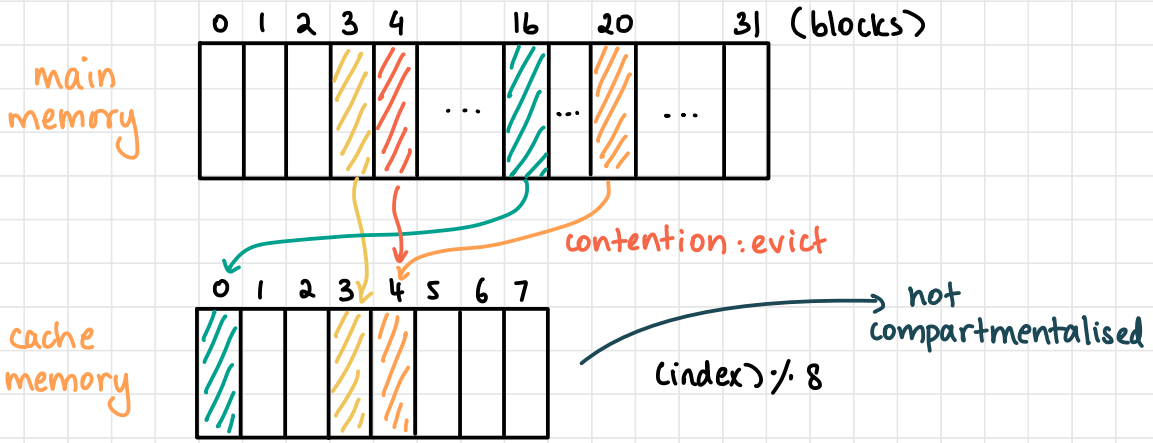
1) Direct Mapped Cache
2) Set Associative Mapping
3) Fully Associative Mapping

GENERAL ORGANISATION of CACHE



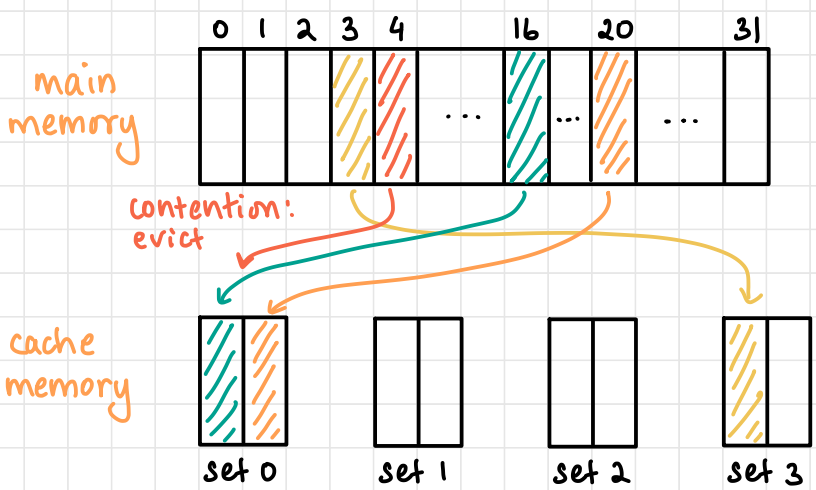
- compartmentalised into sets of 2 (rows)

1) Direct Mapped Cache



- (block no.) mode (no. of sets/ blocks)
- leads to contention
- each line treated as block / set
- aka line mapping technique
- more than one block of memory mapped to one block in cache

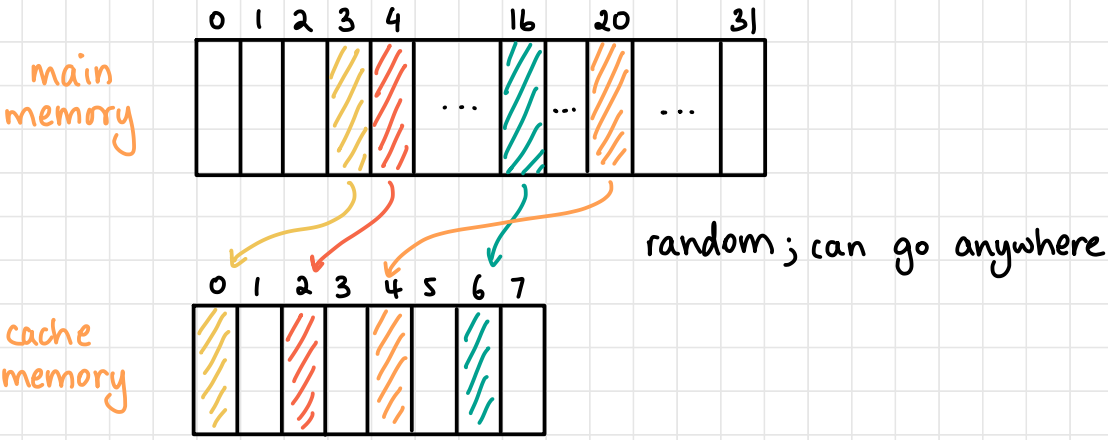
2) Set Associative Mapping



- improvement over direct
- flexibility to place block anywhere in set
- better utilisation of space

(block no.) mode (no. of sets)

3) Fully Associative Mapping

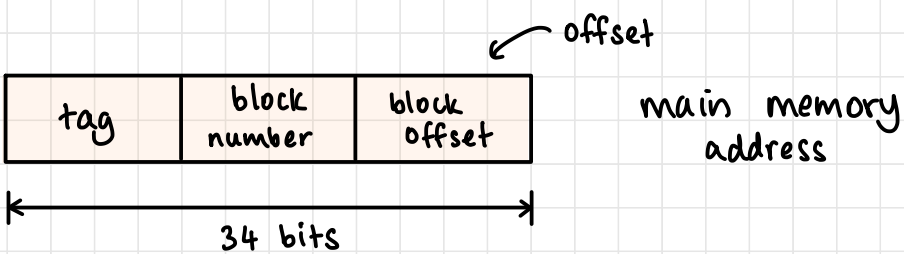


- any block of main memory mapped anywhere on cache
- search is slower

BITS NEEDED TO ADDRESS MAIN MEMORY

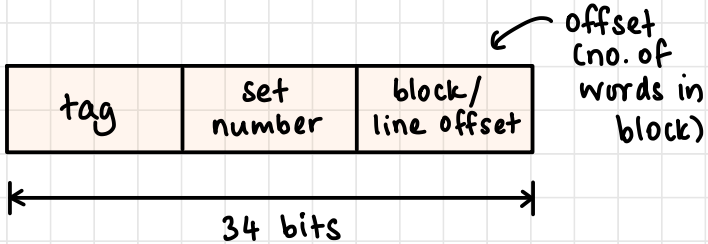
- eg: 16 GB RAM : $2^{4} \times 2^{30}$ = 34 bits

Direct Mapped Cache

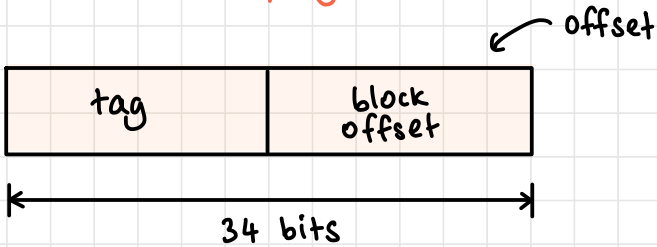


- valid 0: stale/junk data

Set Associative Mapping



Fully Associative Mapping



Q: A 4-way set associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4GB. How many bits are required for the tag field?

main mem = 4 GB
cache size = 16 KB = 2^{14} bytes
block size = 8 x 32 bits = 32 bytes = 8 words

size of mem = $4 \times 2^{30} = 2^{32}$

$\therefore$ 32-bit address for main mem

size of set = $4 \times 8 \times 4 = 2^7$ bytes

$\therefore$ no of sets = $\frac{16\ KB}{4 \times 8 \times 4} = \frac{16 \times 2^{10}}{16 \times 2^3} = 2^7$

↳ block

block size = $8 \times 4 = 32$ bytes $= 2^5$ bytes

block offset = word = 5 bits

$\therefore$ 7 bits for sets

5 bits for word

$\therefore$ 20 bits for tag

Q: Size of MM = 64 K words, cache = 128 blocks, 16 words/block, 1 word = 4 bits, 2-way set associative mapping. Find no. of bits for tag, set, word.

memory = $2^6 \times 2^{10} = 2^{16}$ words

1 set = 2 blocks = 2 × 16 words = 32 words

1 block = 16 words ⇒ 4 bits offset

main memory = 64×2^{10} words = 2^{16} words

= 16 bit addressing

cache → 128 blocks = 64 sets = 2^6 sets

∴ 6 bits for set

Q: Consider direct mapped cache 512 kB, block size 1 kB, 7 bits in tag. Find size of main memory. Word = 1 byte.

cache size = 512 KB = 2^9 bytes

block size = 2^{10} bytes ⇒ 10 bits for offset

no of blocks = 512 = 2^9 blocks

⇒ 9 bits for block

main memory address = 7 + 10 + 9 = 26 bits

∴ size of main memory = $2^{20} \times 2^6$ = 64 MB

Q: Consider a 4-way SAC with block size = 4 KB. Size of MM = 16 GB. 10 tag bits. Find cache size.

size of MM = $16 \times 2^{30} = 2^{34}$ bytes

$\therefore$ address = 34 bit

tag bits = 10 bits

size of block = 4 kB = $2^2 \times 2^{10} = 2^{12}$ bytes

$\therefore$ block offset = 12 bits

$\therefore$ Set no. bits = $34 - (10 + 12)$

$= 34 - (22) = 12$ bits

$\therefore$ no of sets = 2^{12}

$\therefore$ size of cache = $2^{12} \times 4 \times 4$ kB

$= 2^{12} \times 2^2 \times 2^2 \times 2^{10}$

$= 2^{26}$ bytes

$= 64$ MB

size of cache = 64 MB

Q: Cache = 512 KB, tag = 10, set = 8 blocks / set
Main memory = ?

Cache size = no. of sets x 8 x size of block

no of sets = 2^x

size of block = 2^y

x: set bits
y: offset bits

$2^9 \times 2^{10} = 2^{x+y+3}$

$19 = x+y+3 \Rightarrow x+y = 16$

size of memory address = $10 + x + y$

$= 26$ bits

$\therefore$ size of main memory = $2^{26} = 64$ MB

Q: A computer system uses 16 bit memory, direct mapped 2KB cache, 64 bytes / block. Assume word is 1 byte.

Calculate tag, block, word bits

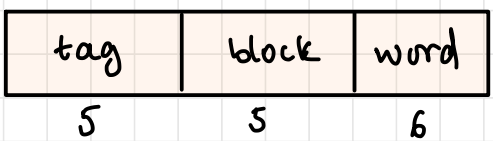
block size = 64 bytes = 2^6 bytes

$\therefore$ block offset = 6 bits = word

no. of blocks = $\frac{2 \times 2^{10}}{2^6} = 2^5$

$\therefore$ block = 5 bits

tag = 16 − (5 + 6) = 5 bits tag



Q: A computer system uses 16 bit memory, 2-way set associative, 2KB cache, 64 bytes / block. Assume word is 1 byte.

(a) Calculate tag, block, word bits

(b) Processor reads data sequentially from the following addresses: 128, 144, 2176, 2180, 128, 2176. Indicate hits and misses

(a) cache = $2 \times 2^{10} = 2^{11}$ bytes

set size = 128 bytes = 2^7

no. of sets = $\frac{2^{11}}{2^7} = 2^4 = 16$ sets

$\therefore$ set bits = 4

no. of words / block = 64 words = 2^6

$\therefore$ word bits = 6

$\therefore$ tag bits = 6

(b) 128, 144, 2176, 2180, 128, 2176 in base -10

		tag	set	offset / word
1.	$(128)_{10}$ =	000000	0010	000000
2.	$(144)_{10}$ =	000000	0010	010000
3.	$(2176)_{10}$ =	000010	0010	000000
4.	$(2180)_{10}$ =	000010	0010	000100
5.	$(128)_{10}$ =	000000	0010	000000
6.	$(2176)_{10}$ =	000010	0010	000000

1. will be miss; tag for first block of set 0010 is set to 000000

2. hit; set 0010, block 1 tag is 000000

3. miss; set 0010, block 2 tag = 000010

4. hit

5. hit

6. hit

VALID BIT

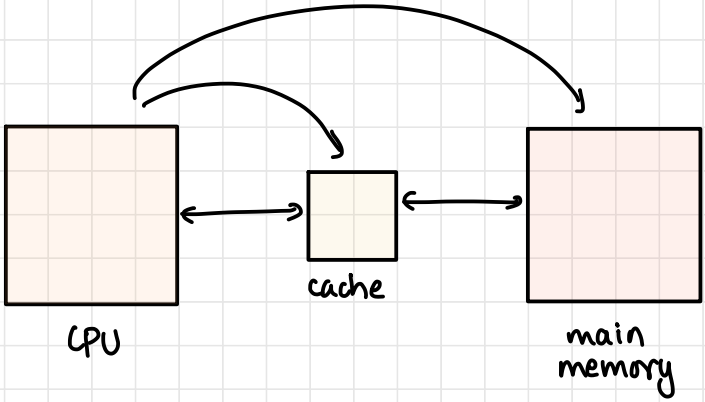
- Provided for each block
- Valid = 0 when power turned on (stale/invalid memory)
- Valid = 1 when block loaded
- If data on disk changes, main memory updated, cache valid bit set to 0

write hit

(1) Write Through Protocol
(2) Write Back Protocol

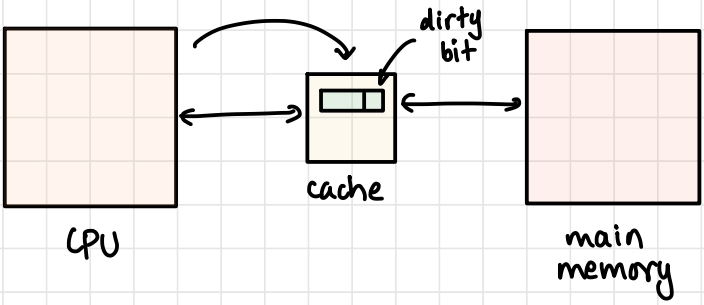
(1) Write Through Protocol

- both main memory & cache updated simultaneously
- ensures consistency
- increased latency



(2) Write Back Protocol

- change only cache
- dirty bit / modified bit flag in cache set to 1
- while sending out dirty block (victim block), main memory is updated



ADVANTAGE OF WRITE BACK

- Not all write operations need to access memory; lower latency
- Several writes in same cache block: force memory write only once at writeback time

Cache Miss

- Data not present in cache
- Miss penalty
- Latency: Time req to retrieve first word of block
- Bandwith: time req to retrieve rest of block

read miss

- Load through / early restart - do not wait for entire block to be transferred

write miss

- Fetch and then overwrite in cache

(1) Write No Allocate
- write directly to memory without affecting cache
- good if same location not needed soon
- valid bit set to 0

(2) Write Allocate
- load newly written data into cache
- easily accessible in cache
- same data needed again

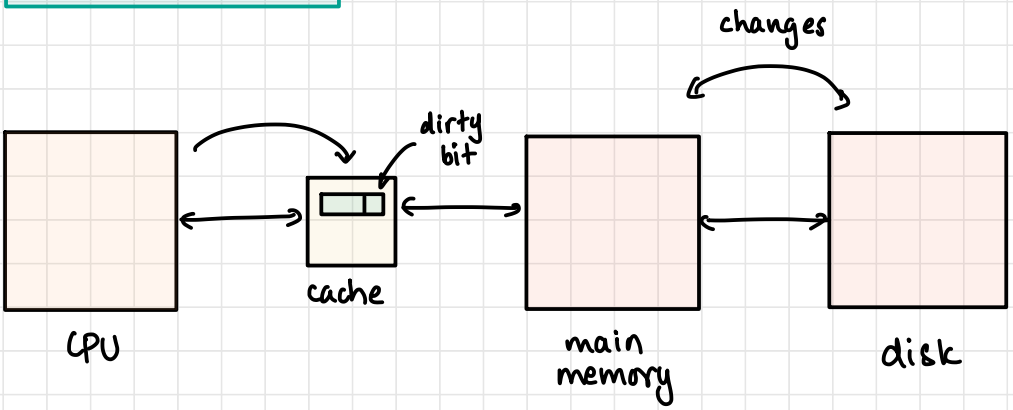
Types of Misses

1) Compulsory Miss
- When cache initially empty, compulsory miss

2) Capacity Miss
-

3) Conflicting Miss
- Set associative mapping
- Due to constraints, even if blocks empty

CACHE COHERENCE



- disk and memory data changes and writeback protocol used
- cache data might have also changed (dirty bit)
- copies of data different
- option: force writeback before main memory is updated from disk

Write Buffer

- processor waiting for writing into main memory: time consuming
- processor places write request into buffer, continues execution
- future access to data: access from buffer
- write through

Q: Consider DMC with 8 cache blocks. Memory block requests are 4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7, which memory blocks will be present in the cache at the end of the sequence? Also calculate hit/miss ratio.

0	~~8~~ ~~16~~ 8 16	8 miss → 8 hit → 16 miss → 8 miss → 16 miss
1	25	25 miss → 25 hit → 25 hit
2		
3	~~3~~ ~~19~~ ~~35~~ 3	3 miss → 19 miss → 35 miss → 3 miss
4	~~4~~ 45	4 miss → 45 miss
5		
6	~~6~~ 22	6 miss → 22 miss
7	7	7 miss

hits = 3 misses = 14 hit ratio = 3/17

Q: Consider 2WSAM with 8 cache blocks. Memory block requests are 4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7, which memory blocks will be present in the cache at the end of the sequence? Also calculate hit/miss ratio. (LRU)

0	~~4~~ 16 8	4 miss → 16 miss → 16 hit 8 miss → 8 hit → 8 hit
1	25 45	25 miss → 25 hit → 25 hit 45 miss
2	6 22	6 miss 22 miss
3	~~3~~ ~~35~~ 7 ~~19~~ 3	3 miss → 35 miss → 7 miss 19 miss → 3 miss

hits = 5 misses = 12 hit ratio = 5/17

Replacement Algorithms

- Direct mapped cache: memory block occupies fixed spot; replacement strategy trivial
- Fully associative, set associative need strategies

(1) Random
- replace random block from cache

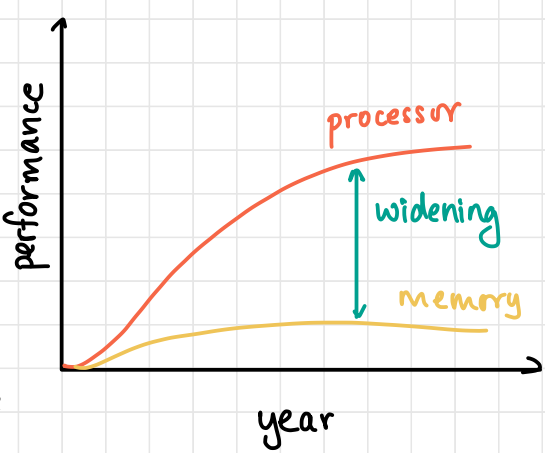
(2) Least Recently Used (LRU)
- replace block that has not been used for longest time
- expensive
- slight randomness introduced for better performance

(3) First In First Out (FIFO)
- evict block which has been in the cache longest

(4) Least Frequently Used (LFU)
- evict block that has been used least frequently

CACHE PERFORMANCE

- Ideal: all requests are hits
- Realistic: not 100%
- How helpful cache is
- Memory wall
- Locality of reference: 10% of code takes up 90% of CPU time



- temporal (looping)
- spatial (arrays)

cache performance - ideal

$$CPU_{time} = CPU_{clock\ cycles} \times CPU_{clock\ cycle\ time}$$

perfect cache

cache performance - realistic

$$CPU_{time} = (CPU_{clock\ cycles} + \text{memory stall cycles}) \times CPU_{clock\ cycle\ time}$$

depends on no. of misses x miss penalty

mem stall cycles

$$\text{mem stall cycles} = IC \times \frac{\text{misses}}{\text{instruction}} \times \text{miss penalty}$$

IC → instruction count

$$\text{mem stall cycles} = IC \times \frac{\text{mem accesses}}{\text{instruction}} \times \frac{\text{misses}}{\text{memory accesses}} \times \text{miss penalty}$$

IF → ID → IE → MEM → WB

IF → instruction cache

MEM → data cache

$$\text{mem stall cycles} = IC \times \frac{\text{mem accesses}}{\text{instruction}} \times \text{miss rate} \times \text{miss penalty}$$

CPI - Cycles per Instruction

$$CPI = \frac{\text{CPU clock cycles for program}}{\text{instruction count}}$$

Q: Assume we have a computer where CPI = 1.0 when all memory accesses hit the cache. Only data accesses are load and store, total about 50% of instructions. Miss penalty is 25 CC, miss rate 2%, how much faster would comp be if all instructions were cache hits?

$$\text{time CPU}_{ideal} = (\text{CPU}_{cc} + \text{mem stall}) \times \text{clock cycle time}$$

$$= (CPI \times IC) \times \text{clock cycle time}$$

$$= IC \times \text{clock cycle time}$$

$$\text{time CPU}_{real} = (\text{CPU}_{cc} + \text{mem stall}) \times \text{clock cycle time}$$

$$\text{mem stall} = IC \times \frac{\text{mem access}}{\text{instr}} \times \text{miss penalty} \times \text{miss rate}$$

$$= IC \times 1.5 \times 25 \times 0.02 = 0.75\ IC$$

$$\text{time CPU}_{real} = (1.75\ IC) \times \text{clock cycle time}$$

$\therefore$ Speedup = 1.75 times

Q: Assume that the CPI for a computer is 1.0 and all memory accesses hit in the cache. If 30% of instructions are load/stores, miss penalty is 100 cycles, miss rate 5%, how much faster would the computer be if all instructions were cache hits?

CPU time with ideal cache = ($CPU_{clock\ cycles}$ + mem stall cycles) x clock cycle time

= (IC x CPI + 0) x clock cycle time

= IC x clock cycle time

CPU time with imperfect cache = ($CPU_{clock\ cycles}$ + mem stall cycles) x clock cycle time

mem stall cycles = $IC \times \frac{\text{mem accesses}}{\text{instructions}} \times \text{miss rate} \times \text{miss penalty}$

= IC x (1 + 0.3) x 0.05 x 100

= IC x 6.5

CPU time = (IC x 1 + IC x 6.5) x clock cycle time

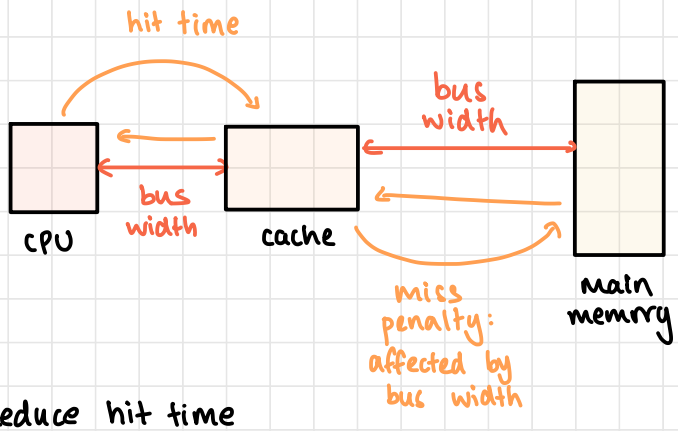
= 7.5 x IC x clock cycle time

Speedup = 7.5 times

AVERAGE MEMORY ACCESS TIME

AMAT = Hit time + miss rate × miss penalty

(Hit time: constant; miss penalty: additional cc if miss incurred)



- Reduce hit time
- Reduce miss rate
- Reduce miss penalty

- Miss penalty depends on bus width

Q: A certain processor uses fully associative cache of size 16 KB. Cache block size is 16 bytes. Assume byte addressible main memory, 32-bit addressing. Tag=? Index=?

$$\text{no. of blocks} = \frac{16 \times 2^{10}}{16} = 2^{10} \text{ blocks}$$

$$\text{words/block} = 16 \text{ words} = 2^4 \text{ words/block}$$

tag = 28 bits

index (block) = 0 bits

Q: The width of physical address is 40 bits. Width of tag field in 512 KB 8-way associative cache is?

Assume block size: 32 bytes

$$\text{no. of sets} = \frac{512 \times 2^{10}}{32 \times 8} = \frac{2^{19}}{2^8} = 2^{11} \text{ sets}$$

$$\text{size of block} = 32 \text{ bytes} = 2^5 \text{ bytes}$$

$$\therefore \text{tag bits} = 40 - (11 + 5) = 24 \text{ bits}$$

Q: Consider a 4WSA cache, 128 lines, line size 64 words, mem address 20 bits. Tag=? Line=? Word=?

$$\text{no. of sets} = \frac{128}{4} = 2^5$$

$$\text{set no} = 5 \text{ bits}$$

$$\text{line size} = 64 \text{ words} = 2^6 \text{ bytes}$$

$$\text{line bits} = 6$$

$$\therefore \text{tag bits} = 9$$

Q: Consider a fully associative mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find Number of bits in tag

cache size: 16 KB = $2^4 \times 2^{10} = 2^{14}$ bytes

block size = 256 bytes = 2^8 bytes

main memory = 128 KB = $2^7 \times 2^{10} = 2^{17}$ bytes

$\therefore$ addressing = 17 bits

fully associative

block offset = 8 bits

$\therefore$ tag = 9 bits

Q. Consider a 2-way set associative cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find Number of bits in tag

cache size = 16 KB = 2^{14} bytes

block size = 2^8 bytes

set size = 2^9 bytes

$$\text{no of sets} = \frac{2^{14}}{2^9} = 2^5 \text{ bytes}$$

$\therefore$ set bits $= 5$ bits

block offset = 8 bits (size of block)

main memory = 128 KB = $2^7 \times 2^{10} = 2^{17}$ bytes

= 17 bit addressing

$\therefore$ tag = $17 - (8+5) = 4$ bits