COL216
Computer Architecture

Memory Organization:

Cache performance

10th March 2022

Cache Policies

- Placement — direct / associative / set assoc
- Read — sequential / concurrent, with/without forwarding
- Load — with/without wrap around
- Replacement — LRU / LFU / FIFO / Random
- Fetch — Demand fetch / Pre-fetch
- Write — ??

Write Hit

- Write in cache ?
  - Obviously yes
- Write in main memory ?
  - May be
    - Writing in memory has an overhead
    - Writing will keep cache and memory consistent
    - Can we handle inconsistent data?

Write Miss

- Write in cache ?
  - May be
    - Where to write?
- Write in main memory ?
  - Definitely yes, if answer above is no
    - Writing in memory has an overhead
    - Writing will keep cache and memory consistent
    - Can we handle inconsistent data?

Write Policies

- Write Back
  - Do not write in memory immediately
  - Write a block in memory when it is getting evicted from cache
- Write Through
  - Write word in memory immediately
  - Two choices in case of a write miss
    - Write Allocate
    - No Write Allocate

Write Policies - Comparison

	WB	WTWA	WTNWA
wr hit	-	word write	word write
wr miss	block rd*	block rd word write	word write
rd miss	block rd*	block rd	block rd
Miss rate	lower	lower	higher

* Write back displaced block if “dirty bit” set

Write Policies : Timings

- Write time depends upon
  - number of blocks transferred
  - block transfer time
  - number of words transferred
  - word transfer time
- Timings as seen by the CPU vs timings as seen by BUS
  - CPU time can be minimized by using a write buffer

Write Buffers

- Processor writes into a buffer and proceeds without waiting
- Transfer from buffer to memory takes place in background
- Possible with WT as well as WB
- What happens if a read comes up for data still in buffer?

Cache Performance

Average memory access time =

Hit time + Miss rate * Miss penalty

Mem stalls / Instr =

Miss rate * Miss Penalty * Mem accesses / Instr

Performance Improvement

- Reducing miss penalty
- Reducing miss rate
- Reducing hit time

Performance Improvement

- Reducing miss penalty
- Reducing miss rate
- Reducing hit time

Miss rate depends on

- cache size, block size, associativity, policies

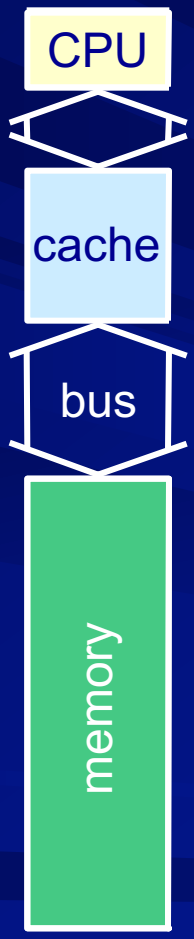
Miss penalty depends on

- Memory access time
- cache – memory interface

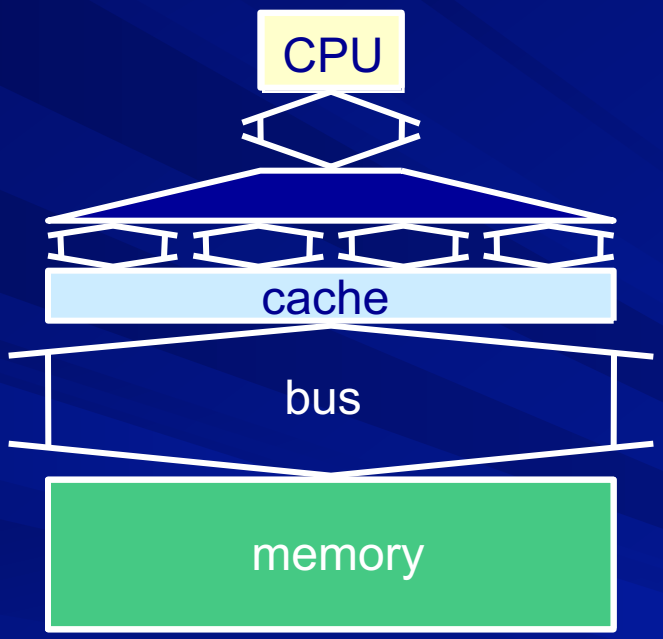
Types of misses

- compulsory miss
- capacity miss
- conflict miss

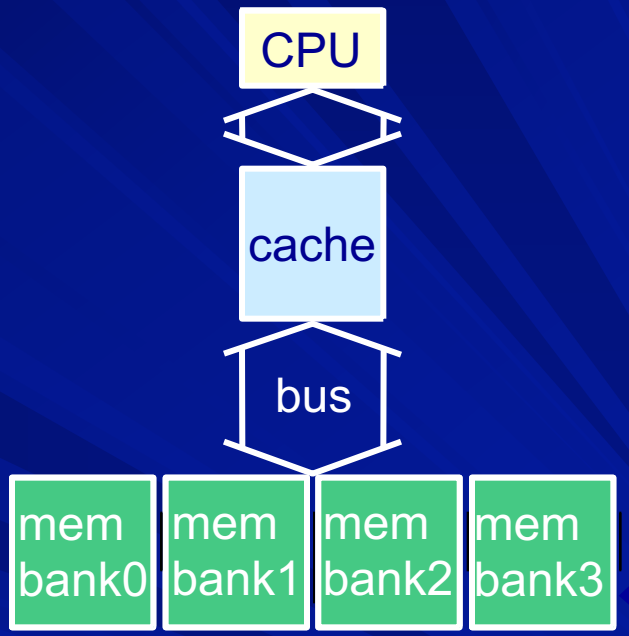
Transferring blocks to/from memory



a. one word wide memory

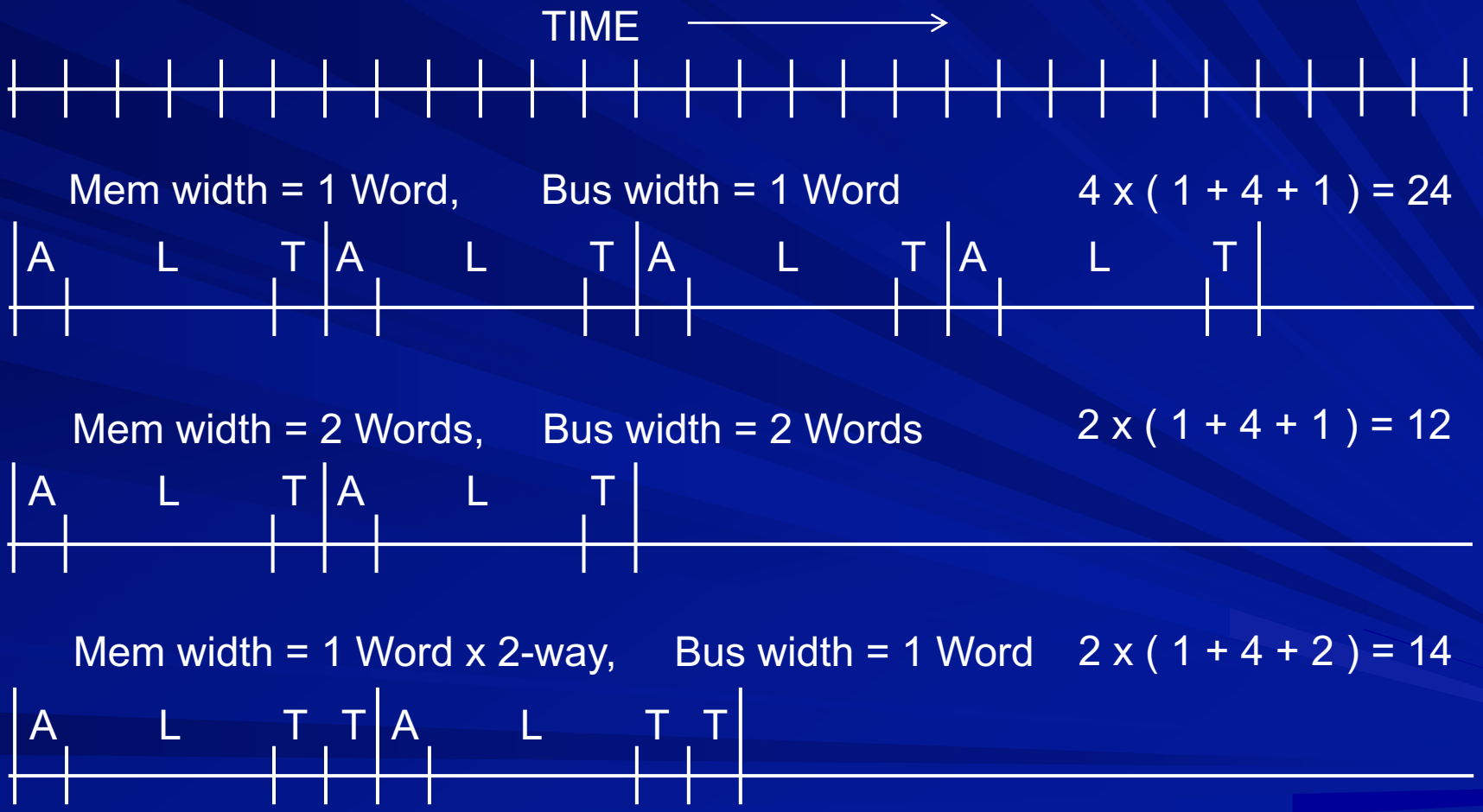


b. four word wide memory



c. interleaved memory

Miss penalty



Miss penalty example

- 1 clock cycle to send address
- 15 cycles for RAM access
- 1 cycle for sending data
- block size = 4 words

Compute miss penalty

Answer:

case (a): 4 (1 + 15 + 1) = 68 or 1 + 4 (15 + 1) = 65

case (b): 1 + (15 + 1) = 17

case (c): 1 + 15 + 4 x 1 = 20

DRAM with page mode

- Memory cells are organized as a 2-D structure
- Entire row is accessed at a time internally and kept in a buffer
- Reading multiple bits from a row can be done very fast
  - sequentially, without giving address again
  - randomly, giving only the column addresses

Performance analysis example

CPI_{eff} = CPI + Miss rate * Miss Penalty * Mem accesses / Instr

CPI = 1.2 Miss rate = 0.5%

Block size = 16 w

Miss penalty??

Assume mem access / Instr = 1

Miss penalty calculation

Data / address transfer time = 1 cycle

Memory latency = 10 cycles

a) Miss penalty = 16*(1+10+1) = 192

b) Miss penalty = 4*(1+10+1) = 48

c) Miss penalty = 4*(1+10+4*1) = 60

Back to CPI calculation

CPI_{eff} = 1.2 + .005 * miss penalty * 1.0

a) 1.2 + .005 * 192 * 1.0 = 1.2 + .96
= 2.16

b) 1.2 + .005 * 48 * 1.0 = 1.2 + .24
= 1.44

c) 1.2 + .005 * 60 * 1.0 = 1.2 + .30
= 1.50

What makes cache work?

- Temporal Locality
  - references repeated in time
- Spatial Locality
  - references repeated in space
  - Special case: Sequential Locality

Locality Example

Code 1

```
for ( i = 0; i < 8000; i++)
    for ( j = 0; j < 8; j++)
        A [ i ] [ j ] = B [ j ] [ 0 ] + A [ j ] [ i ];
```

Code 2

```
for ( j = 0; j < 8; j++)
    for ( i = 0; i < 8000; i++)
        A [ i ] [ j ] = B [ j ] [ 0 ] + A [ j ] [ i ];
```

Arrays are stored row-wise in C, column-wise in Matlab

Identify spatial locality and temporal locality

Is cache managed by HW or SW?

Check for hit or miss

Hit: Read/write cache

Miss: Possible actions

1. write back a block in memory
2. read a block from memory
3. wite a word in memory

Are these actions performed by hardware or by software?

Hit time, miss rate, miss penalty

How do these parameters vary with

1. cache size?
2. degree of associativity?
3. block size?
4. memory access time?
5. degree of memory interleaving?

Indicate the type of miss that is affected in each case.

Cache comparison example

Cache	mapping	block size	I-miss	D-miss	CPI
1	direct	1 word	4%	8%	2.0
2	direct	4 word	2%	5%	??
3	2-way s.a.	4 word	2%	4%	??

Miss penalty = 6 + block size

50% instructions have a data reference

Solution:

Stall cycles: cache1: 7*(.04+.08*.5)=.56

cache2: 10*(.02+.05*.5) = .45

cache3: 10*(.02+.04*.5) = .40

Solution continued

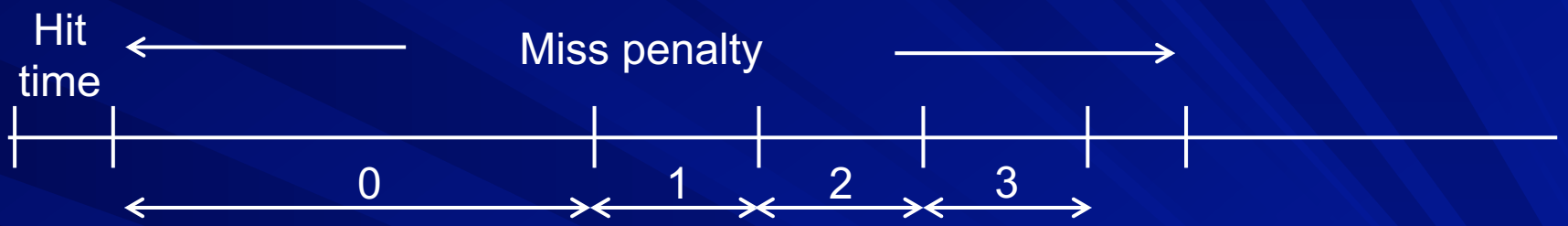
Cache	CPI	clock period	time/instr
1	2.0	2.0	4.0
2	2.0 - .56 + .45 = 1.89	2.0	3.78
3	2.0 - .56 + .40 = 1.84	2.4	4.416

Hit time, miss rate, miss penalty

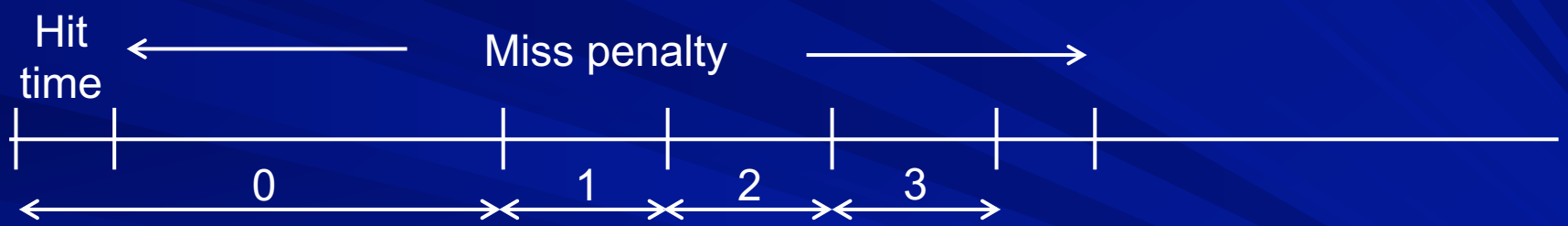
How do these parameters vary with

1. read policy?
2. load policy?
3. replacement policy?
4. fetch policy?
5. write policy?

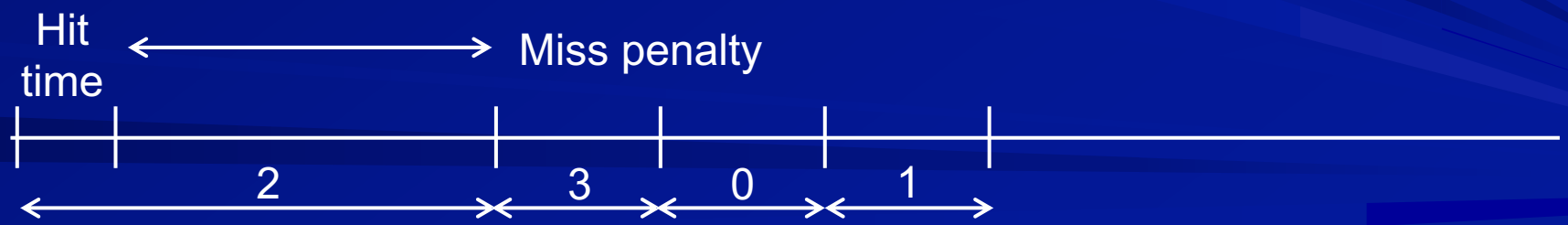
Miss penalty



Concurrent read



Wrap around load + forwarding



Cache plus various buffers

- Buffer for block(s)/word(s) waiting to be written into memory
  - write buffer
- Buffer for block(s) fetched in anticipation
  - prefetch buffer
- Buffer for displaced blocks waiting for re-admission or permanent disposal
  - victim cache

Which write policy works better?

- Clustered and frequent writes
  - get the block into cache
  - do many writes into it
  - write back
- Sparse writes
  - just write into main memory
- Sparse writes followed by reads
  - get the block
  - write into both

Improved buffer for write through

- Effective for Write Through cache when there are multiple writes to same block
- Buffer size = 1 block, rather than 1 word
- Writes belonging to same block are collected in buffer
- Write from buffer to memory when
  - Buffer gets full or
  - Write for a different block arrives

Split cache vs. Unified, cache

Compare

1. access conflicts
2. space utilization
3. cost.

On-chip cache vs. off-chip cache

Compare

1. access time
2. capacity.

2-Level Cache Performance

Average memory access time =

Hit time + Miss rate * Miss penalty

Mem stalls / Instr =

Miss rate * Miss Penalty * Mem accesses / Instr

Average memory access time = Hit time +

Miss rate1 * Miss penalty1 +

Miss rate2 * Miss penalty2

Mem stalls / Instr = (Miss rate1 * Miss penalty1 +

Miss rate2 * Miss penalty2) *

Mem accesses / Instr

Miss rates in 2 level cache

Consider 2 level cache

L1 miss rate = no. of L1 misses/ no. of requests at L1

L2 miss rate = no. of L2 misses/ no. of requests at L2?

- Global Miss rate
  - no. of misses / no. of requests, on the whole
- Solo Miss rate
  - miss rate if only L2 cache was present
- Relationship among these?

Inclusive & Exclusive caches

- Inclusive:

  Every information that is there in L1 cache is also in L2 cache

  Not vice versa (of course)

- Exclusive:

  No information is duplicated

- Neither inclusive, nor exclusive

What is required to ensure the above properties?

2-level cache example

CPI with no miss = 1.0 Clock = 500 MHz

Main mem access time = 200 ns

Miss rate = 5%

Adding L2 cache (20 ns) reduces miss to 2%. Find performance improvement.

Solution:

Miss penalty (mem) = 200/2 = 100 cycles

Effective CPI with L1 = 1+5%*100 = 6

Solution continued

Miss penalty (L2) = 20/2 = 10 cycles

Total CPI = Base CPI + stalls due to L1 miss
+ stalls due to L2 miss

= 1.0 + 5% * 10 + 2% * 100

= 1.0 + 0.5 + 2.0 = 3.5

Performance ratio = 6.0/3.5 = 1.7

Question

Why many modern processors have 3 levels of cache, while a decade ago it was common to have only one level?