L6 Cache Part II

Zonghua Gu, 2018

Acknowledgement: some slides taken from UC Berkeley CS61C

Cache (*Performance)* Terms

- Hit rate: fraction of accesses that hit in the cache
- Miss rate: 1 – Hit rate
- Miss penalty: time to access a block from lower level in memory hierarchy
- Hit time: time to access cache memory (including tag comparison)

Average Memory Access Time (AMAT)

- Average Memory Access Time (AMAT) is the average time to access memory considering both hits and misses in the cache

AMAT = Hit rate * Hit time + Miss rate * Miss time

= (1 – Miss rate)*Hit time + Miss rate * (Hit time + Miss penalty)

= Hit time + Miss rate * Miss penalty

- For single-level cache, Miss penalty = Memory access time, since Miss time = Memory access time + Hit time

AMAT Example

AMAT = Hit time + Miss rate * Miss penalty

- Given a 200 psec clock, a miss penalty of 50 clock cycles, a miss rate of 2%, and a cache hit time of 1 clock cycle, what is AMAT?

  A : ≤200 psec

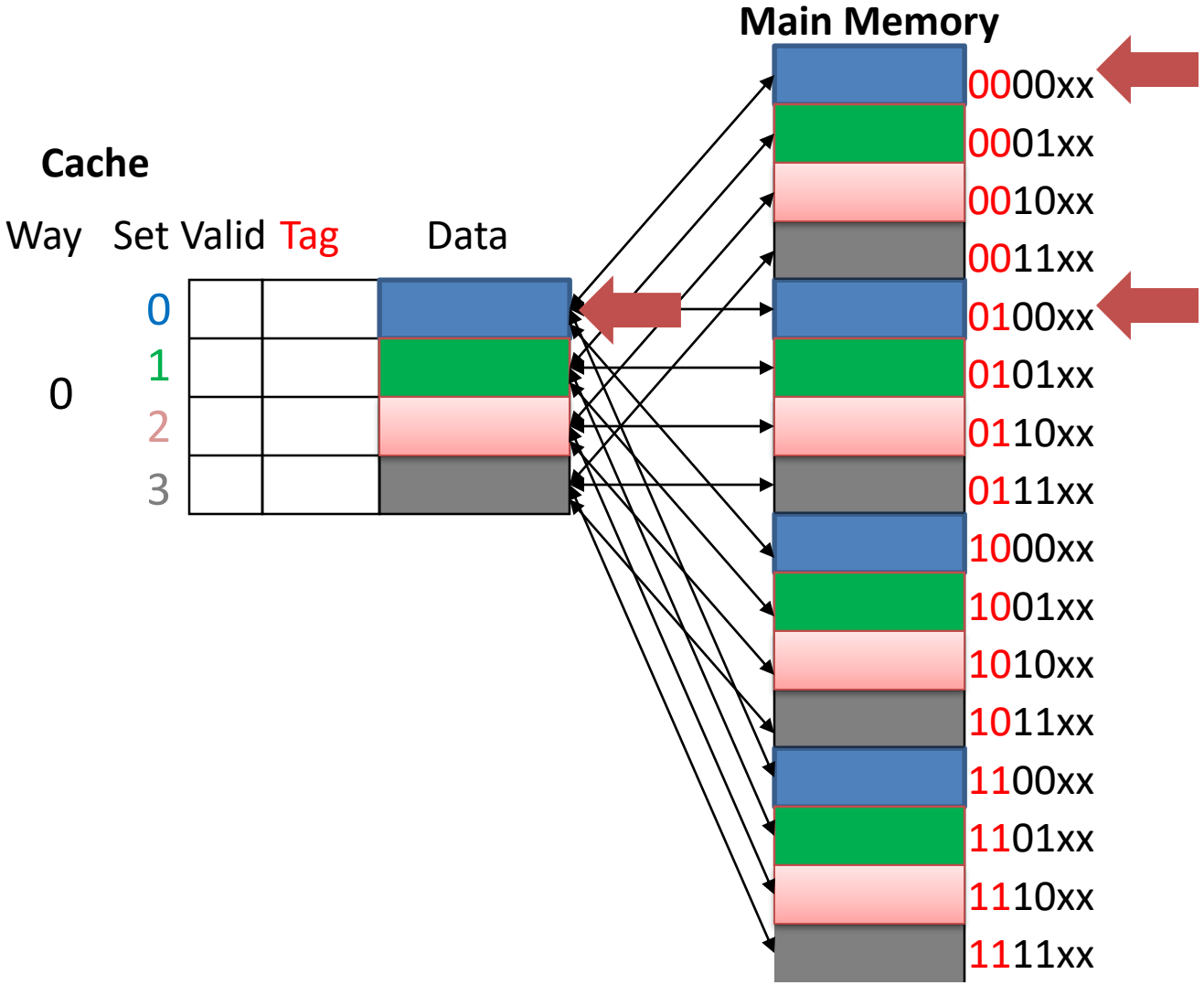
  B : 400 psec

  C : 600 psec

  D : ≥ 800 psec

AMAT=(1+.02*50)*200=400 psec

Ping Pong Cache Example: DM Cache w/ 4 Blocks

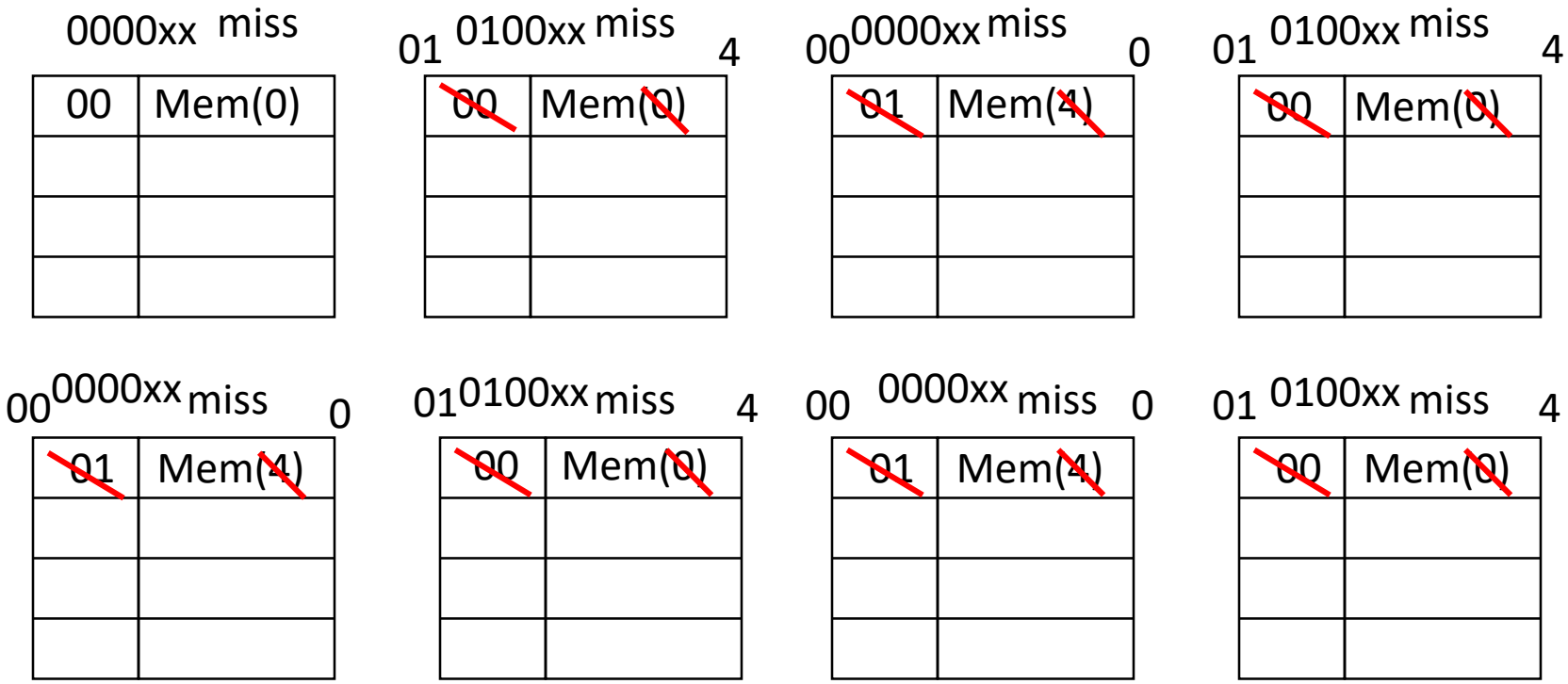


- Consider the sequence of memory addresses referenced at runtime :
  - 0000xx, 0100xx, 0000xx, 0100xx, 0000xx, 0100xx, 0000xx, 0100xx
  - They all map to Set 0, which contains 1 cache block

Ping Pong Cache Example: DM Cache w/ 4 Blocks

- Consider the sequence of memory addresses referenced at runtime: 0000xx, 0100xx, 0000xx, 0100xx, 0000xx, 0100xx, 0000xx, 0100xx. All mapped to Set 0.

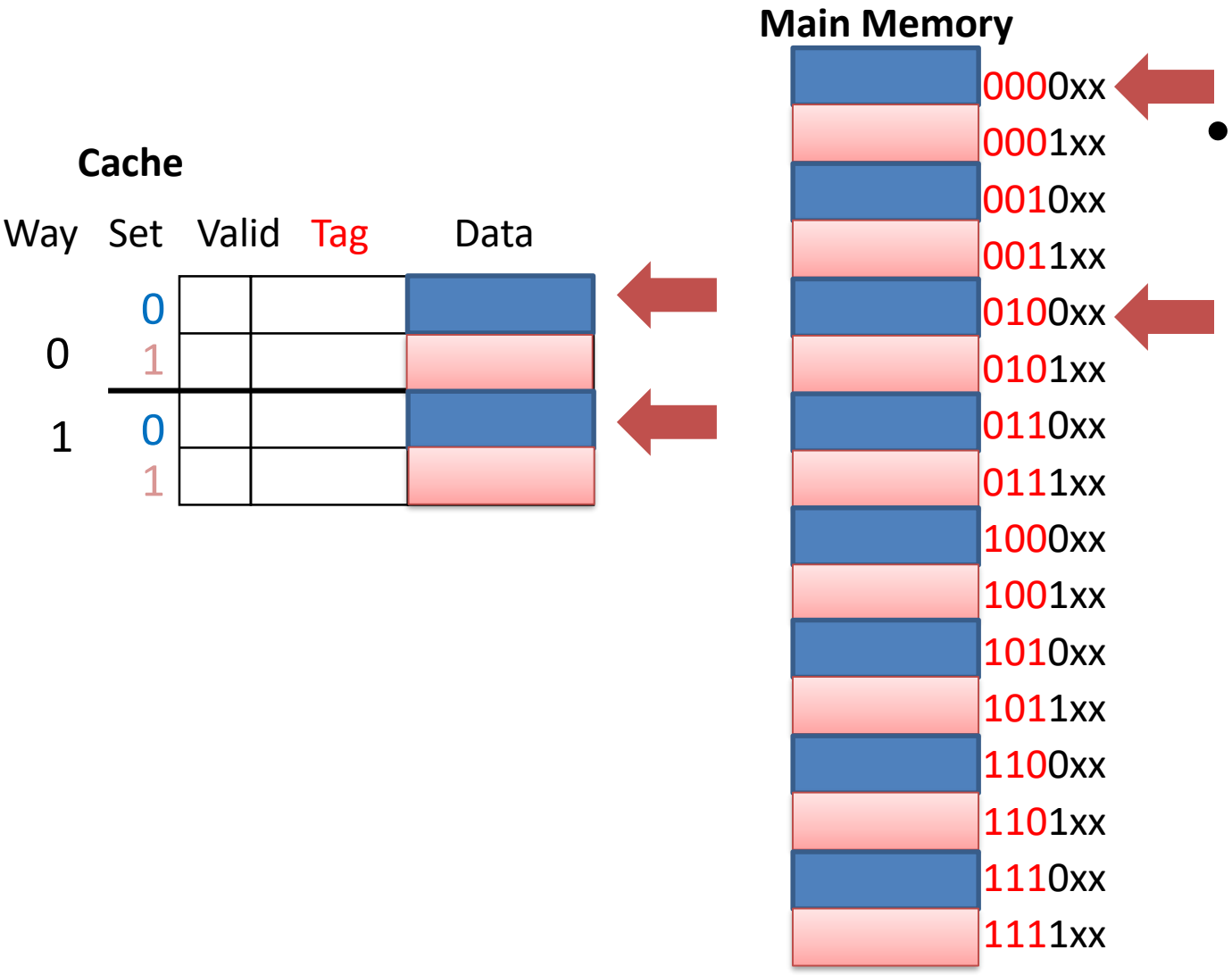
  Start with an empty cache - all blocks initially marked as not valid



  - 8 requests, 8 misses

Ping-pong effect due to conflict misses - two memory addresses that map into the same cache block

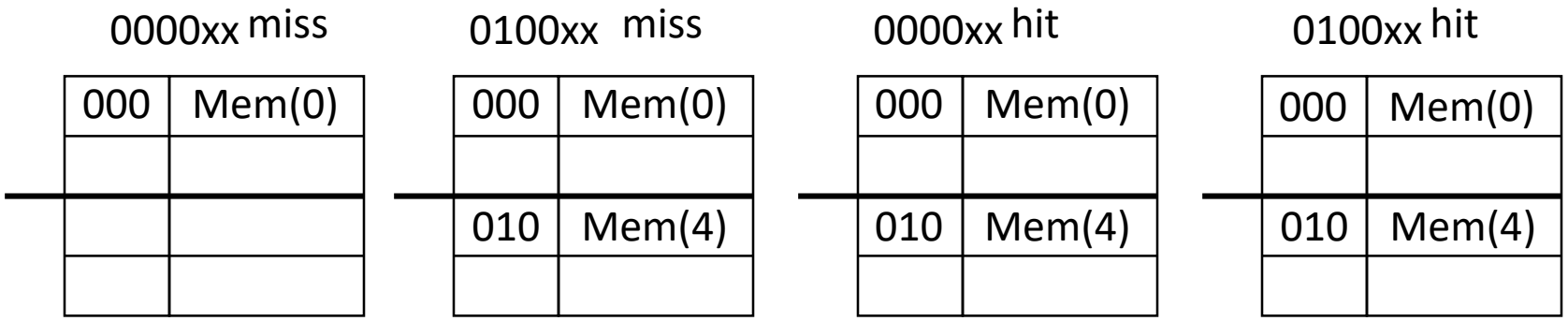
Ping Pong Cache Example: 2-Way SA Cache w/ 4 Blocks



- Consider the sequence of memory addresses referenced at runtime :
  - 0000xx, 0100xx, 0000xx, 0100xx, 0000xx, 0100xx, 0000xx, 0100xx
  - They all map to Set 0, which contains 2 cache blocks

Ping Pong Cache Example: 2-Way SA Cache w/ 4 Blocks

- Consider the sequence of memory addresses referenced at runtime: 0000xx, 0100xx, 0000xx, 0100xx, 0000xx, 0100xx, 0000xx, 0100xx. All mapped to Set 0.
  Start with an empty cache - all blocks initially marked as not valid

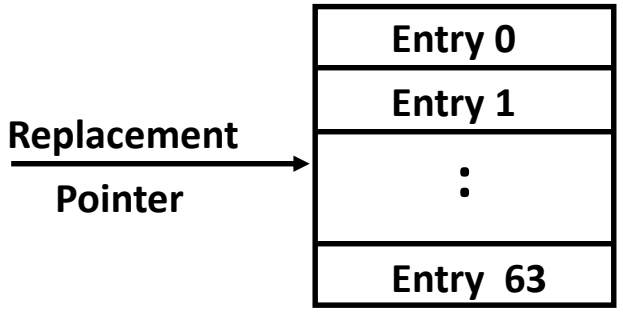


… …

  - 8 requests, 2 misses
- Two memory addresses that map into the same cache set can co-exist in the 2-way SA cache, removes the Ping-Pong effect of DM cache

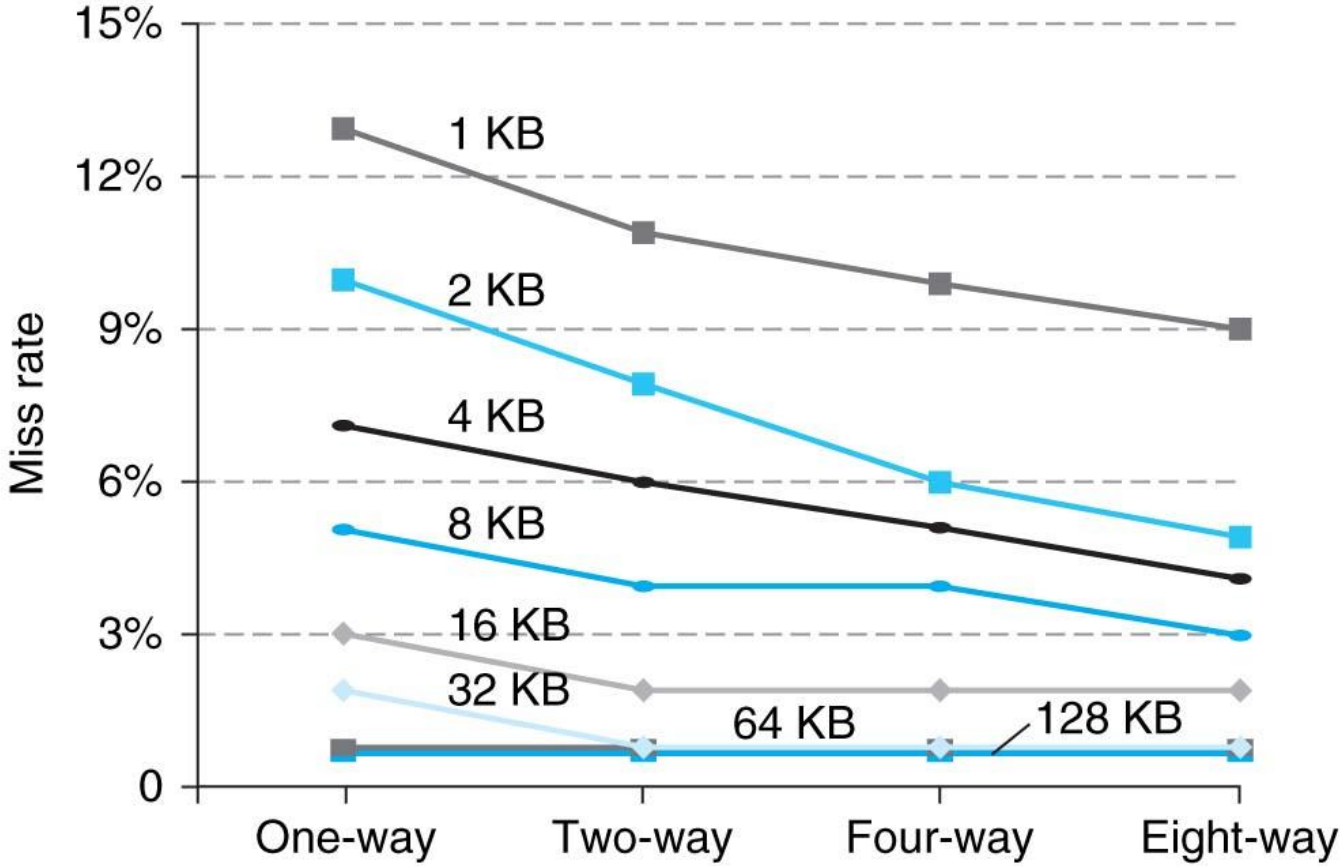
Cache Replacement Policies

- Random Replacement
  - A cache block is randomly selected to evict
- Least-Recently Used
  - Hardware keeps track of access history
  - Replace the entry that has not been used for the longest time
  - For 2-way SA cache, one bit per set → set to 1 when a block is referenced; reset the other way's bit to 0; always replace the block with bit=0.
  - For N-way SA cache, can be expensive to implement
- Example of a simple "Pseudo" LRU Implementation for 64-way SA cache
  - Replacement pointer points to one cache entry
  - Whenever access is made to the entry the pointer points to:
    - Move the pointer to the next entry
  - Otherwise: do not move the pointer
  - (example of "not-most-recently used" replacement policy)



Benefits of Set-Associative Caches

- Choice of DM $ versus SA $ depends on the cost of a miss versus the cost of implementation



- Largest gains are in going from direct mapped to 2-way (20%+ reduction in miss rate)

Reduce AMAT

- Reduce hit time
  - e.g., smaller cache, lower associativity
- Reduce miss rate
  - e.g., larger cache, higher associativity
- Reduce miss penalty
  - e.g., multiple-level cache hierarchy
- Need to balance cache parameters (Capacity, associativity, block size)

Sources of Cache Misses (3 C's)

- *Compulsory*: cold start, first access to a block
  - Misses that would occur even with infinite cache
  - Can be reduced by increasing block size
- *Capacity*: cache is too small to hold all data needed by the program
  - Misses that would occur even under perfect replacement policy
  - Can be reduced by increasing cache capacity
- *Conflict*: collisions due to multiple memory addresses mapped to same cache set
  - Recall the ping-pong cache example
  - Can be reduced by increasing associativity and/or increasing cache capacity

Review

Answer: # Cache Misses for DM Cache

- Consider the following program that repeatedly accesses an array of words A[4]

  for (int i=0; i++, i<10000) {sum += A[0]+A[1]+A[2]+A[3];}

- Suppose A[0] starts at memory address 000000, how many cache misses due to reading array A[] for a DM cache with either 1) the Tag-Set Index-Offset model; or 2) the Set Index-Tag-Offset model? How to classify the cache misses?

For Tag-Set Index-Offset model: 4 misses

- 1st cache miss brings into cache a block with address 0000xx into Set 0, which contains A[0]
- 2nd cache miss brings into cache a block with address 0001xx into Set 1, which contains A[1]
- 3rd cache miss brings into cache a block with address 0010xx into Set 2, which contains A[2]
- 4th cache miss brings into cache a block with address 0011xx into Set 3, which contains A[3]
- 9996 cache hits.

4 misses are all compulsory

For Set Index-Tag-Offset model: 10000 misses

- 1st cache miss brings into cache a block with address 0000xx into Set 0, which contains A[0]
- 2nd cache miss brings into cache a block with address 0001xx into Set 0, which contains A[1], and replaces A[0]
- 3rd cache miss brings into cache a block with address 0010xx into Set 0, which contains A[2], and replaces A[1]
- 4th cache miss brings into cache a block with address 0011xx into Set 0, which contains A[3], and replaces A[2]
- 5th cache miss ress 0000xx into Se ces A[3]
-

4 misses are compulsory
9996 misses are conflict

Review

Answer: # Cache Misses for 2-Way SA Cache

- Consider the following program that repeatedly accesses an array of words A[4]

  for (int i=0; i++, i<10000) {sum += A[0]+A[1]+A[2]+A[3];}

- Suppose A[0] starts at memory address 000000, how many cache misses due to reading array A[] for a 2-way SA cache with either 1) the Tag-Set Index-Offset model; or 2) the Set Index-Tag-Offset model? How to classify the cache misses?

For Tag-Set Index-Offset model: 4 misses

- 1st cache miss brings into cache a block with address 0000xx into Set 0, which contains A[0]
- 2nd cache miss brings into cache a block with address 0001xx into Set 1, which contains A[1]
- 3rd cache miss brings into cache a block with address 0010xx into Set 0, which contains A[2]
- 4th cache miss brings into cache a block with address 0011xx into Set 1, which contains A[3]
- 9996 cache hits.

4 misses are all compulsory

For Set Index-Tag-Offset model: 10000 misses

- 1st cache miss brings into cache a block with address 0000xx into Set 0, which contains A[0]
- 2nd cache miss brings into cache a block with address 0001xx into Set 0, which contains A[1]
- 3rd cache miss brings into cache a block with address 0010xx into Set 0, which contains A[2], and replaces one of A[0] or A[1]. Suppose A[0] is replaced.
- 4th cache miss brings into cache a block with address 0011xx into Set 0, which contains A[3], and replaces one of A[1] or A[2]. Suppose A[1] is replaced
- 5th cache miss brings into cache a block with address 0000xx into Se… 3]
- …

4 misses are compulsory
9996 misses are conflict

Review

Answer: # Cache Misses for FA Cache

- Consider the following program that repeatedly accesses an array of words A[4]

  for (int i=0; i++, i<10000) {sum += A[0]+A[1]+A[2]+A[3];}

- Suppose A[0] starts at memory address 000000, how many cache misses due to reading array A[] for a FA cache with either 1) the Tag-Set Index-Offset model; or 2) the Set Index-Tag-Offset model? How to classify the cache misses?

For either model: 4 misses

- 1st cache miss brings into cache a block with address 0000xx into Set 0, which contains A[0]
- 2nd cache miss brings into cache a block with address 0001xx into Set 0, which contains A[1]
- 3rd cache miss brings into cache a block with address 0010xx into Set 0, which contains A[2]
- 4th cache miss brings into cache a block with address 0011xx into Set 0, which contains A[3]
- 9996 cache hits.

4 misses are compulsory

Effect of Cache Parameters on Performance

- Larger cache size
  + reduces capacity and conflict misses
  - Increases hit time

- Higher associativity
  + reduces conflict misses
  - increases hit time

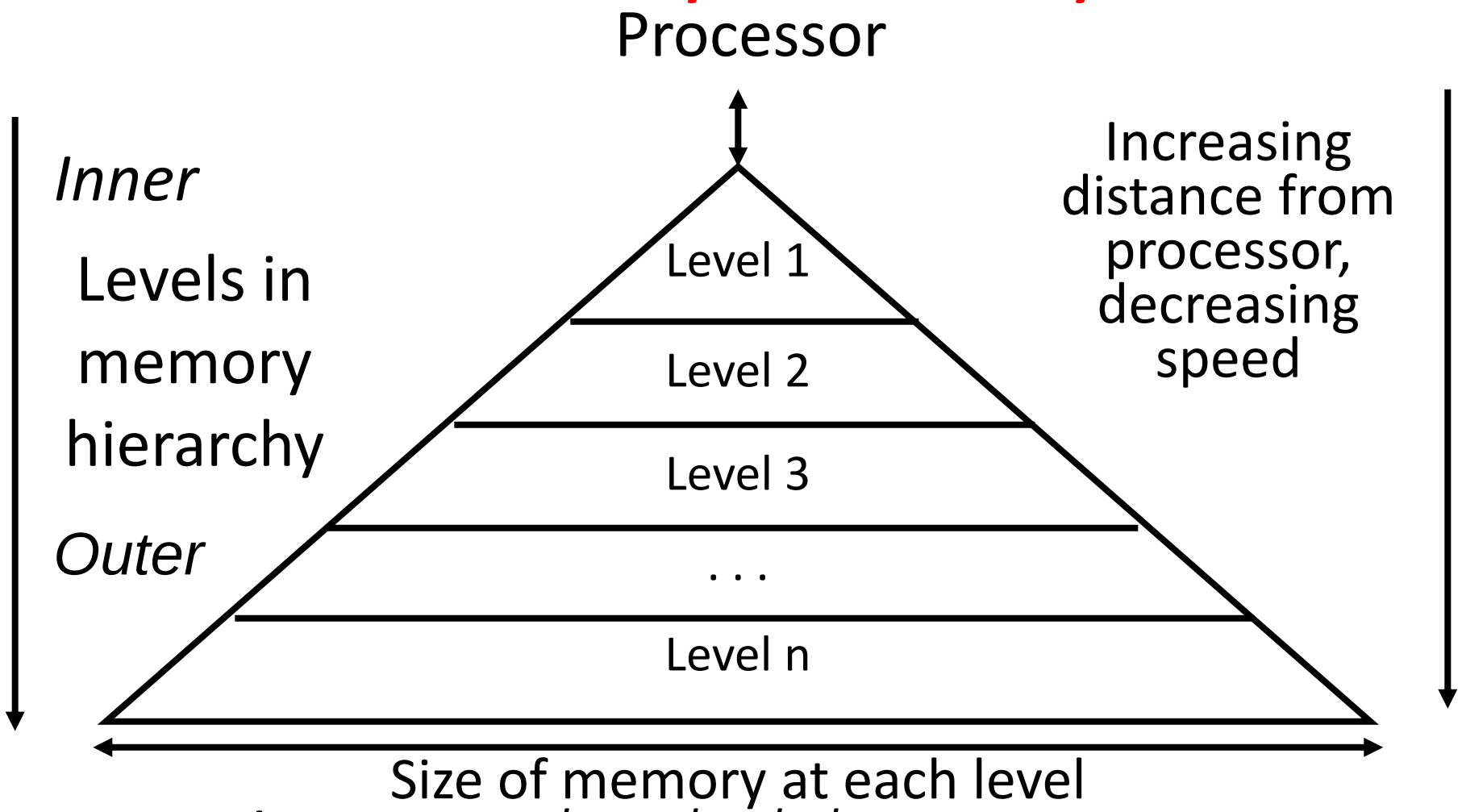
- Larger block size
  + reduces compulsory misses
  - increases conflict misses and miss penalty

Improving Cache Performance

AMAT = Hit time + Miss rate x Miss penalty

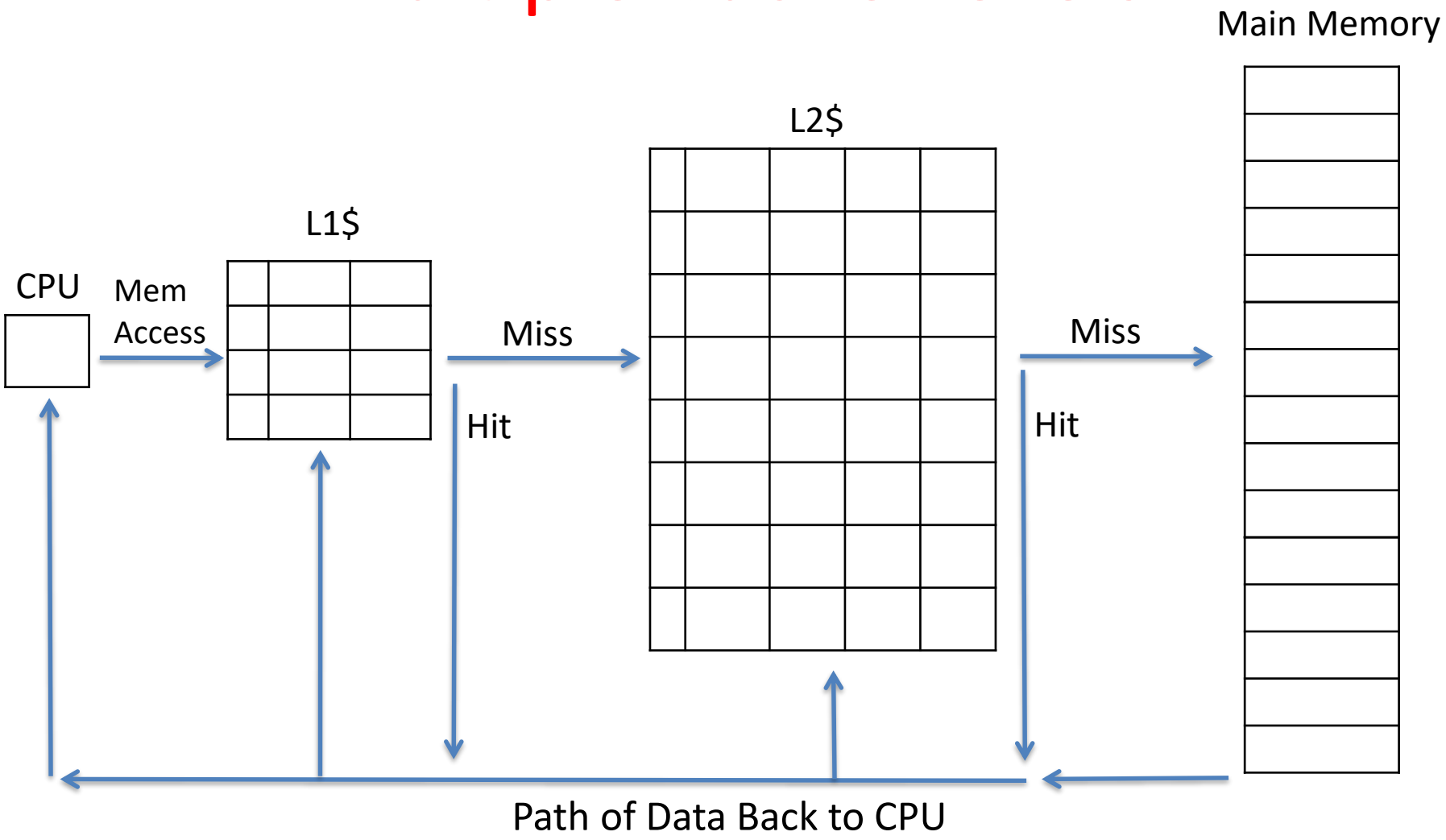
- Reduce hit time
  - e.g., smaller cache, lower associativity
- Reduce miss rate
  - e.g., larger cache, higher associativity
- Reduce miss penalty
  - e.g., multiple-level cache hierarchy

Memory Hierarchy



As we move to lower levels, latency goes up and price per bit goes down.

Multiple Cache Levels



L1 cache size < L2 cache size << memory size

Local vs. Global Miss Rates

- *Local miss rate* – the fraction of references to one level of a cache that miss
  - L2 Local Miss Rate = L2 Misses / L1 Misses
- *Global miss rate* – the fraction of references that miss in all levels of caches and must go to memory
  - Global Miss rate = L2 Misses / Total Accesses
  - = (L2 Misses / L1 Misses) × (L1 Misses / Total Accesses)
  - = L2 Local Miss Rate × L1 Local Miss Rate
- L1 Miss Penalty = L2 AMAT; L2 Miss Penalty = Memory access time
- L1 cache only: AMAT = Hit Time + Miss rate × Miss penalty
- L1+L2 caches: AMAT = L1 Hit Time + L1 Local Miss rate × (L2 Hit Time + L2 Local Miss rate × L2 Miss penalty)

AMAT Example

- L1 Hit Time: 1 cycle, L1 Miss Rate: 2%
- L2 Hit Time: 5 cycle, L2 Miss Rate: 5%.
- Main Memory access time: 100 cycles
- No L2 Cache:
  - AMAT = 1 + .02*100 = 3
- With L2 Cache:
  - AMAT = 1 + .02*(5 + .05*100) = 1.2

Multilevel Cache Considerations

- Different design considerations for L1 Cache and LLC (Last Level Cache)
  - L1 Cache design should focus on fast access: minimize hit time to achieve shorter clock cycle, e.g., with smaller size, lower associativity; miss penalty is small thanks to L2 and lower caches, so higher miss rate is OK
  - LLC design should focus on low miss rate: miss penalty due to main memory access is very large, e.g., with larger size, higher associativity

Summary

- Calculation of AMAT
- Three major categories of cache misses:
  - Compulsory Misses; Conflict Misses; Capacity Misses
- Multi-level caches
  - Optimize 1st level to minimize hit time
  - Optimize last level to minimize miss rate
- Lots of cache parameters (large design space)
  - Block size, cache size, associativity, write-back vs. write through, etc.