RUTGERS

THE STATE UNIVERSITY
OF NEW JERSEY

Computer Architecture (CS-211)
Recitation-10

Song Wen

Topics

- Cache
- PA5

* Some materials are collected and compiled from previous year's CS 211 lectures and TAs

Cache Terminology

- *block* (*cache line*): minimum unit that may be cached
- *frame*: cache storage location to hold one block
- *hit*: block is found in the cache
- *miss*: block is not found in the cache
- *miss ratio*: fraction of references that miss
- *hit time*: time to access the cache
- *miss penalty*: time to replace block on a miss
- Temporal locality: Recently accessed locations will likely be accessed again in near future
- Spatial locality: Will likely access locations close to ones recently accessed in near future

Average Memory Access Time (AMAT)

- L1 cache hit is 5 cycles (core to L1 and back)
- L2 cache hit is 20 cycles (core to L2 and back)
- memory access is 100 cycles (core to mem and back)
- 20% miss ratio in L1 and 40% miss ratio in L2
- Calculate AMAT?

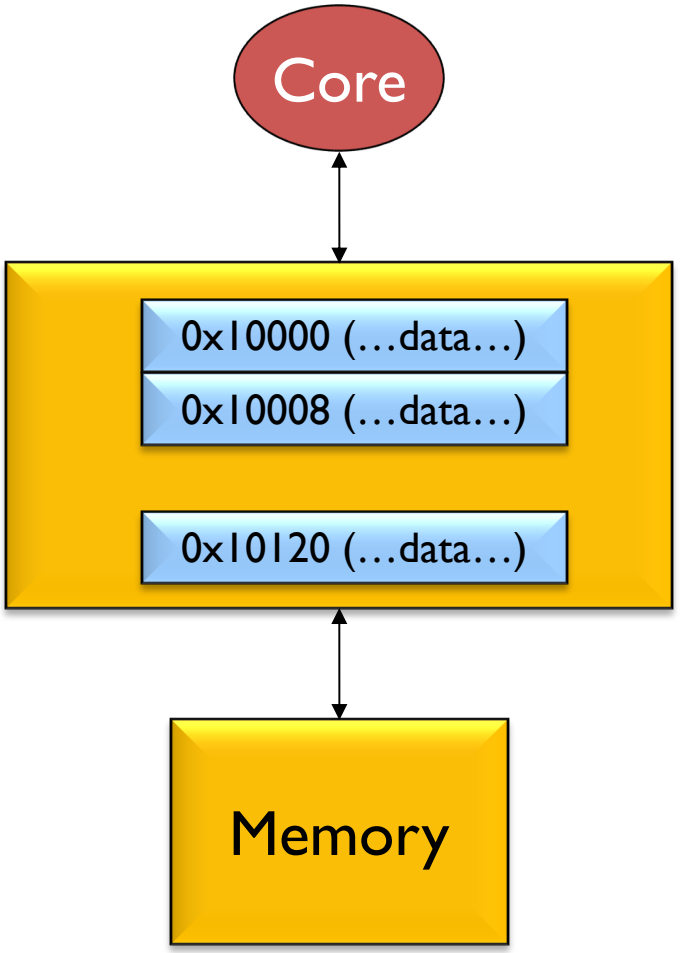
AMAT = Hit time + (Miss rate x Miss penalty)

- AMAT Calculation:
  - $0.8 \times 5 + 0.2 \times (0.6 \times 20 + 0.4 \times 100) \approx 14$

Miss Ratio Example

Address sequence from core:
(assume 8-byte lines)

- 0x10000 Miss
- 0x10004 Hit
- 0x10120 Miss
- 0x10008 Miss
- 0x10123 Hit
- 0x10004 Hit



Final *miss ratio* is 50%

Writes and Cache

Reading information from a cache is straight forward

What about writing?

- What if you're writing data that is already cached (write-hit)?
- What if the data is not in the cache (write-miss)?

Dealing with a write-hit

- Write-through - immediately write data back to memory
- Write-back - defer the write to memory for as long as possible

Dealing with a write-miss

- write-allocate - load the block into memory and update
- no-write-allocate - writes directly to memory

Write-through caches are typically no-write-allocate

Write-back caches are typically write-allocate

Programming Assignment 5

- Implement a cache simulator!
- Input parameters: **cache size, associativity, block size**
  - Cache size: power of 2
  - Block size: power of 2
- Replacement policy: **FIFO, LRU**
- Write policy: **Write through**

Program should follow

$ ./first <cache size> <associativity> <cache policy> <block size> <trace file>

Cache size: total size of the cache (power of 2)
Trace file: name of the trace file
Associativity

- **direct** – simulate a direct mapped cache
- **assoc** – simulate a fully associative cache
- **assoc:n** – simulate an n-way associative cache (n should be power of 2)

Programming Assignment 5

- Your program should print out
  - Number of memory reads
  - Number of memory writes
  - Cache hits
  - Cache misses

```
$./first 32 assoc:2  fifo 4 trace2.txt
no-prefetch
Memory reads: 3499
Memory writes: 2861
Cache hits: 6501
Cache misses: 3499
with-prefetch
Memory reads: 3521
Memory writes: 2861
Cache hits: 8124
Cache misses: 1876
```

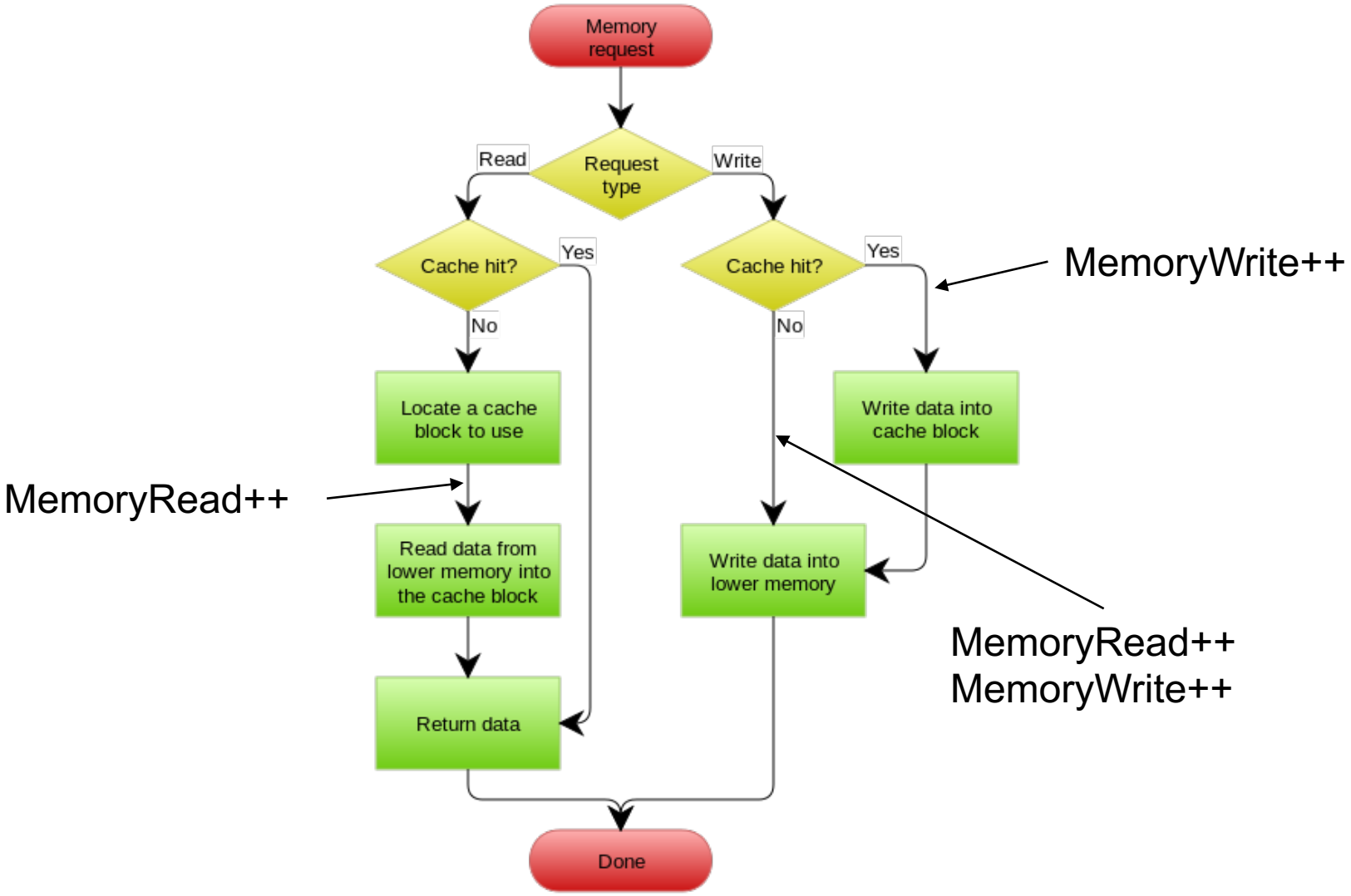
C = S * E * B

32 = S * 2 * 4

S = 4 Bytes

Set bits : 2
Block bits : 2
Tag bits : 48 - 2 – 2 = 44

Programming Assignment 5



Q&A

Thanks!