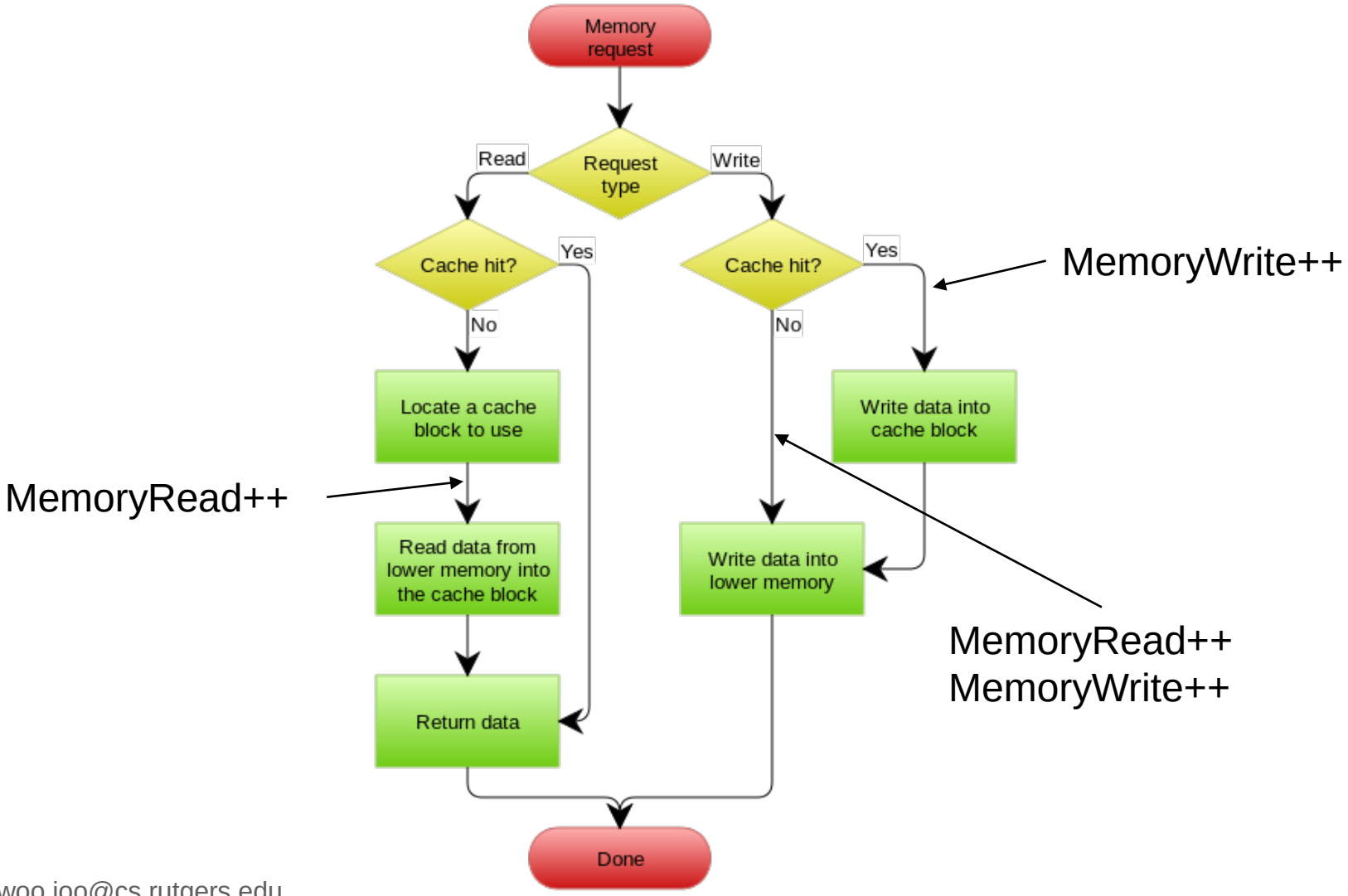
RUTGERS

THE STATE UNIVERSITY
OF NEW JERSEY

Recitation 11

Jae Woo Joo

Programming Assignment 4



Programming Assignment 4

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

```
$./first 32 assoc:2  4 trace1.txt
cache A
Memory reads: 173
Memory writes: 334
Cache hits: 827
Cache misses: 173
cache B
Memory reads: 667
Memory writes: 334
Cache hits: 333
Cache misses: 667
```

C = S * E * B

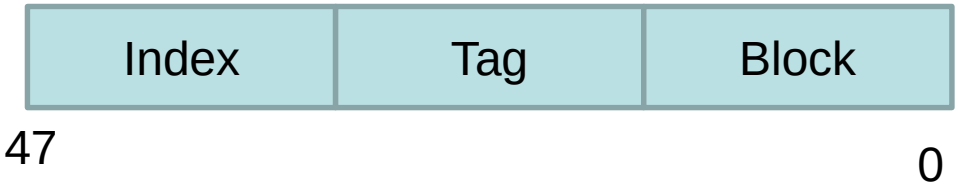
32 = S * 2 * 4
S = 4 Bytes

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

Programming Assignment 4

- You should implement two types of caches
- **Type A**: uses the indexing schemes discussed in class

- **Type B**: the order of index and tag bits are reversed



Programming Assignment 4

- $ ./first 32 assoc:2 4 trace1.txt

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

- trace1.txt
  - 0x804ae19: W 0x9cb32e0
  - 0x9cb32e0

    = 0000 0000 0000 0000 0000 1001 1100 1011 0011 0010 1110 0000

  - Type A
    - 0000 0000 0000 0000 0000 1001 1100 1011 0011 0010 1110 00 00
  - Type B
    - 00 00 0000 0000 0000 0000 1001 1100 1011 0011 0010 1110 00 00

Programming Assignment 4 (Example)

- $ ./first 4 direct 1 trace1.txt

Trace1.txt
R 1100
R 0100
W 0100
W 1100
W 1110
R 0100

C = S * E * B
4 = S * 1 * 1
S = 4

Block bit : 0
Set index : 2
Tag bit : 2 (4 – 2 – 0)

Type A
- Read: 5
- Write: 3
- Hit: 1
- Miss: 5

R: 1 1 1 1 1
W: 1 1 1
H: 1
M: 1 1 1 1 1

Writing to Cache

- Assume that the address we want to write to is already loaded in the cache.

Index	V	Tag	Block
110	1	11010	42803

Address	Data
1101 0110	42803

- Then we write a new value to that address
  - (Mem[214] = 21763)

Index	V	Tag	Block
110	1	11010	21763

Address	Data
1101 0110	42803

Inconsistent Memory

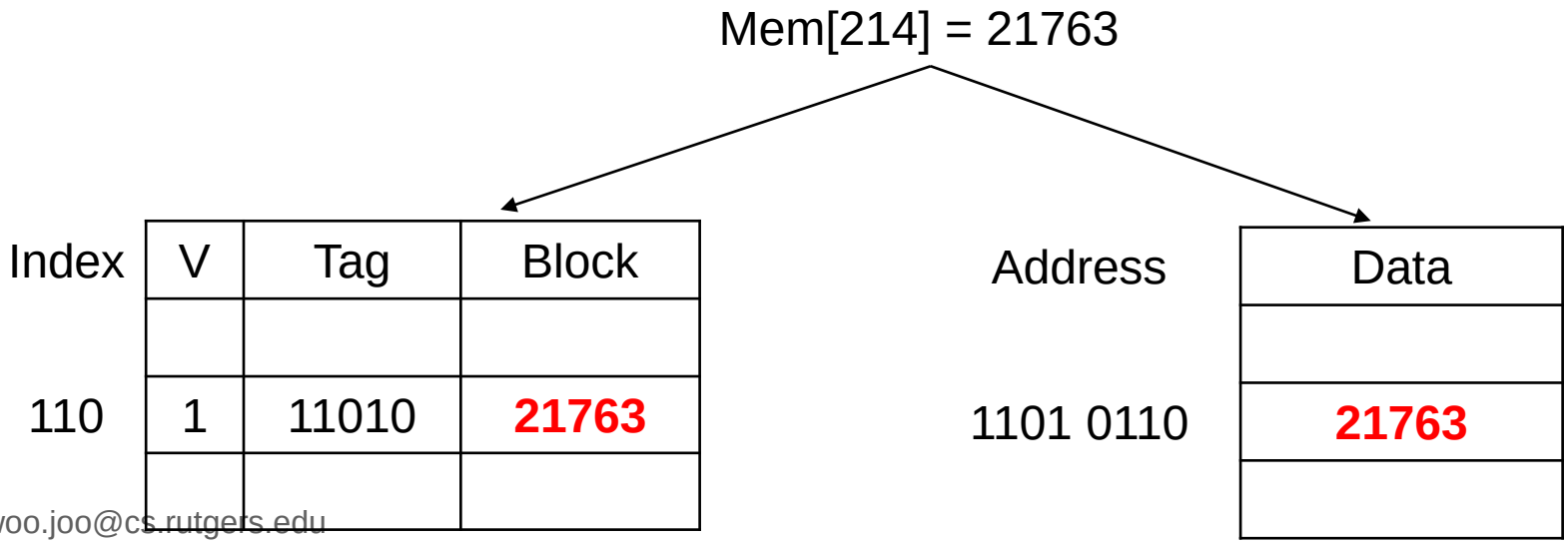
- Now the cache and the memory contain different data
  - Inconsistent data!

- How can we solve these kind of problem?

Write-through Cache

- Write-through cache solves the inconsistency problem by forcing all writes to update **both cache** and **main memory**.

- This is simple to implement and keeps the cache and memory consistent
  - Disadvantage: For every write to main memory, we use up bandwidth between the cache and the memory



Write-back Caches

- In write-back cache, the memory is not updated until the cache block needs to be replaced.

- We might write some data to the cache at first, leaving it inconsistent with the main memory
  - This point, the cache block is marked “dirty” to indicate inconsistency

Mem[214] = 21763

Index	V	**Dirty**	Tag	Block
110	1	**1**	11010	**21763**

Address	Data
1000 1110	1225
1101 0110	**42803**

Write-back Caches

- We don't need to store the new value back to main memory until the cache block gets replaced
- If we read from Mem[142], which maps to same cache block, the modified cache contents will first be written to main memory

Index	V	**Dirty**	Tag	Block
110	1	**1**	11010	**21763**

Address	Data
1000 1110	1225
1101 0110	**21763**

- Then the cache block will be replaced with address 142

Index	V	**Dirty**	Tag	Block
110	1	**0**	11010	**1225**

Address	Data
1000 1110	**1225**
1101 0110	21763

Write-back Caches

- Dirty blocks are written back to file when evicted
- Advantage
  - Not all write operations need to access main memory (Cost efficient)
- Disadvantage
  - Complex to implement
  - There exists data inconsistency issue if program crashes before writing dirty blocks

Q & A

- Any questions?