# 16.482 / 16.561 Computer Architecture and Design

Instructor: Dr. Michael Geiger

Spring 2014

**Lecture 8:**

Memory hierarchies (continued)

# Lecture outline

- Announcements/reminders
  - HW 5 due today
  - HW 6 to be posted; due 4/17
- Review
  - Multiple issue and multithreading
  - Memory hierarchy design
- Today's lecture
  - Continue with memory hierarchy design

# Review: TLP, multithreading

- ILP (which is implicit) limited on realistic hardware for single thread of execution
- Could look at explicit parallelism: Thread-Level Parallelism (TLP) or Data-Level Parallelism (DLP)
- Focus on TLP and multithreading
  - Coarse-grained: switch threads on a stall
  - Fine-grained: switch threads every cycle
  - Simultaneous: allow multiple threads to execute in same cycle

# Review: memory hierarchies

- We want a large, fast, low-cost memory
  - Can't get that with a single memory
- Solution:  use a little bit of everything!
  - Small SRAM array → *cache*
    - Small means fast and cheap
    - More available die area → multiple cache levels on chip
  - Larger DRAM array → *main memory*
    - Hope you rarely have to use it
  - Extremely large *hard disk*
    - Costs are decreasing at a faster rate than we fill them

# Review: Cache operation & terminology

- Accessing data (and instructions!)
  - Check the top level of the hierarchy
    - If data is present, **hit**, if not, **miss**
    - On a miss, check the next lowest level
      - With 1 cache level, you check main memory, then disk
      - With multiple levels, check L2, then L3
- Average memory access time gives overall view of memory performance

  AMAT = (hit time) + (miss rate) x (miss penalty)
  - Miss penalty = AMAT for next level
- Caches work because of locality
  - Spatial vs. temporal

# 4 Questions for Memory Hierarchy

- Q1: Where can a block be placed in the upper level?
  (Block placement)
  - *Fully associative, set associative, direct-mapped*
- Q2: How is a block found if it is in the upper level?
  (Block identification)
- Q3: Which block should be replaced on a miss?
  (Block replacement)
- Q4: What happens on a write?
  (Write strategy)

# Q2: Block identification

- Tag on each block
  - No need to check index or block offset
- Increasing associativity shrinks index, expands tag

# Address breakdown

- Can address single byte (smallest addressable unit)
  - If addressing multiple bytes, address is lowest in word/half-word
  - E.g., word containing bytes 0-3 accessed by address 0
- **Block offset**: byte address within block
  - # block offset bits = $\log_2$(block size)
- **Index**: line (or set) number within cache
  - # index bits = $\log_2$(# of cache lines or sets)
  - # of cache sets = $$\frac{\text{total cache size}}{(\text{block size}) \times (\text{associativity})}$$
- **Tag**: remaining bits

# Address breakdown example

- Given the following:
  - 32-bit address
  - 32 KB direct-mapped cache
  - 64 bytes per block

What are the sizes for the tag, index, and block offset fields?

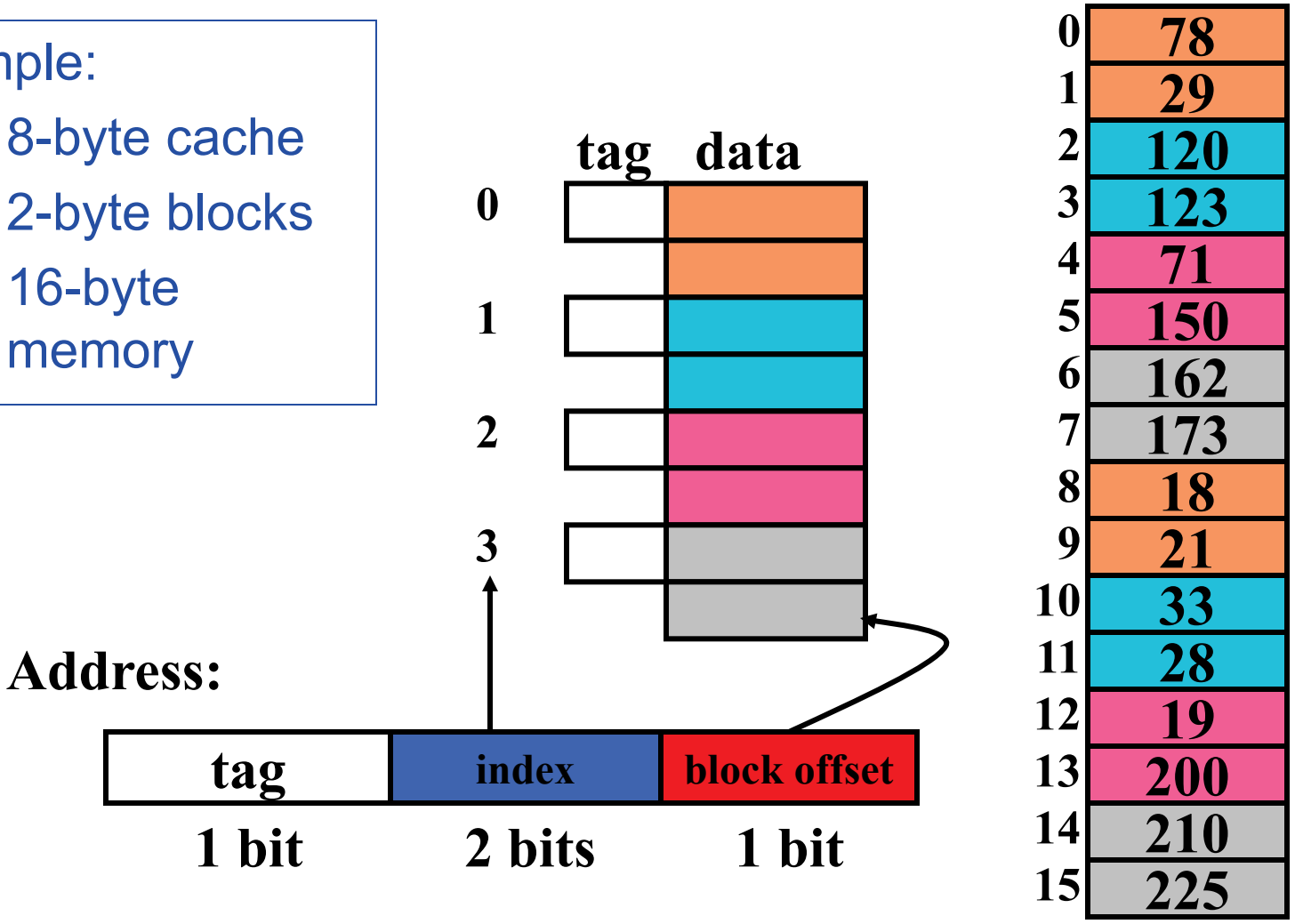
# Solution

- We know:
  - Size of each block: 64 bytes = $2^6$ bytes
    → Takes 6 bits to address each byte within a block
    → Offset size = 6 bits
  - Cache is direct-mapped: associativity = 1
  - Total cache size = 32 KB = $2^{15}$ bytes (1 K = $2^{10}$)
    → # of cache lines = $2^{15}$ / ($2^6$ x $2^0$) = $2^9$
    → Index size = 9 bits
  - Address size = 32 bits
    → Already use 6 + 9 = 15 bits for offset and index
    → Tag size = 32 – 15 = 17 bits

# Mapping memory to cache

Example:

- 8-byte cache
- 2-byte blocks
- 16-byte memory



# Mapping memory example

- Say we have the following cache organization
  - Direct-mapped
  - 4-byte blocks
  - 32 B cache
  - 6-bit memory addresses
- Given the addresses below, answer the following questions
  - Which addresses belong to the exact same block?
    - i.e., both tag and index are equal
  - Which addresses map to the same cache line, but are part of different blocks?
    - i.e., index numbers are equal, but tags are different
  - Addresses: 3, 4, 6, 11, 37, 43

# Solution

- Figure out size of each field
  - 4 = $2^2$ bytes per block
    → 2-bit offset
  - 32 / (4 x 1) = 8 = $2^3$ cache lines → 3-bit index
  - 6-bit address, 5 bits for offset & index → 1-bit tag
- Convert addresses into binary
  - 3 = **0 000 11**
  - 4 = **0 001 00**
  - 6 = **0 001 10**
  - 11 = **0 010 11**
  - 37 = **1 001 01**
  - 43 = **1 010 11**
- Addresses 4 & 6 are part of same block
  - Index = 001, tag = 0
- Address 37 maps to same cache line as 4 & 6, but different block
  - Index = 001, tag = 1
- Addresses 11 & 43 map to same cache line, but different blocks
  - Index = 010 for both
  - Tag = 0 for 11, 1 for 43

# Q3: Block replacement

- On cache miss, bring requested data into cache
  - If line contains valid data, that data is evicted
- When we need to evict a line, what do we choose?
  - Easy choice for direct-mapped—only one possibility!
  - For set-associative or fully-associative, choose least recently used (LRU) line
    - Want to choose data that is least likely to be used next
    - Temporal locality suggests that's the line that was accessed farthest in the past

# Direct-mapped cache example

- Assume the following simple setup
  - Only 2 levels to hierarchy
  - 16-byte memory → 4-bit addresses
  - Cache organization
    - Direct-mapped
    - 8 total bytes
    - 2 bytes per block → 4 lines
    - Write-back cache
  - Leads to the following address breakdown:
    - Offset: 1 bit
    - Index: 2 bits
    - Tag: 1 bit

# Direct-mapped example (cont.)

- We'll use the following sequence of accesses

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

# Direct-mapped cache example: initial state

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Registers:**
**$t0 = ?, $t1 = ?**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #1

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 1 = $0001_2$**
**→ Tag = 0**
**→ Index = 00**
**→ Offset = 1**

**Registers:**
**$t0 = ?, $t1 = ?**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: 0

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | **29** |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #1

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 1 = $0001_2$**
**→ Tag = 0**
**→ Index = 00**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = ?**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 0 | 78 | 29 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: 1

**Memory**

| | |
|---|---|
| 0 | 78 |
| 1 | 29 |
| 2 | 120 |
| 3 | 123 |
| 4 | 71 |
| 5 | 150 |
| 6 | 162 |
| 7 | 173 |
| 8 | 18 |
| 9 | 21 |
| 10 | 33 |
| 11 | 28 |
| 12 | 19 |
| 13 | 200 |
| 14 | 210 |
| 15 | 225 |

# Direct-mapped cache example: access #2

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 8 = $1000_2$**
**→ Tag = 1**
**→ Index = 00**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = ?**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 0 | 78 | 29 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: 1

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #2

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 8 = $1000_2$**
**Tag = 1**
**Index = 00**
**Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| **1** | **0** | **1** | **18** | **21** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: **2**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #3

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 4 = $0100_2$**
**→ Tag = 0**
**→ Index = 10**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: 2

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Q4: What happens on a write?

| | Write-Through | Write-Back |
|---|---|---|
| Policy | Data written to cache block<br>also written to lower-level memory | Write data only to the cache<br>Update lower level when a block falls out of the cache |
| Debug | Easy | Hard |
| Do read misses produce writes? | No | Yes |
| Do repeated writes make it to lower level? | Yes | No |

# Write policy example

- Say we have the following code sequence:

```
lw    $t0, 0($t1)
addi  $t2, $t0, 5
sw    $t2, 4($t1)
addi  $t2, $t2, 10
sw    $t2, 8($t1)
addi  $t2, $t2, 20
sw    $t2, 12($t1)
```

- Assume all addresses map to the same cache block
- Assume this initial cache line state (each block entry is 1 word, not 1 byte, shown in decimal)

| Valid | Dirty | Tag | Block | | | |
|---|---|---|---|---|---|---|
| 1 | 0 | 1000 | 100 | 0 | 0 | 0 |

# Write policy example (cont.)

- In write-through cache, no need for dirty bit
- Every single write (store instruction) causes entire block to be written to memory
  - After first store, cache line:

| Valid | Tag | Block | | | |
|---|---|---|---|---|---|
| 1 | 1000 | 100 | 105 | 0 | 0 |

  - After second store:

| Valid | Tag | Block | | | |
|---|---|---|---|---|---|
| 1 | 1000 | 100 | 105 | 115 | 0 |

  - After third store

| Valid | Tag | Block | | | |
|---|---|---|---|---|---|
| 1 | 1000 | 100 | 105 | 115 | 135 |

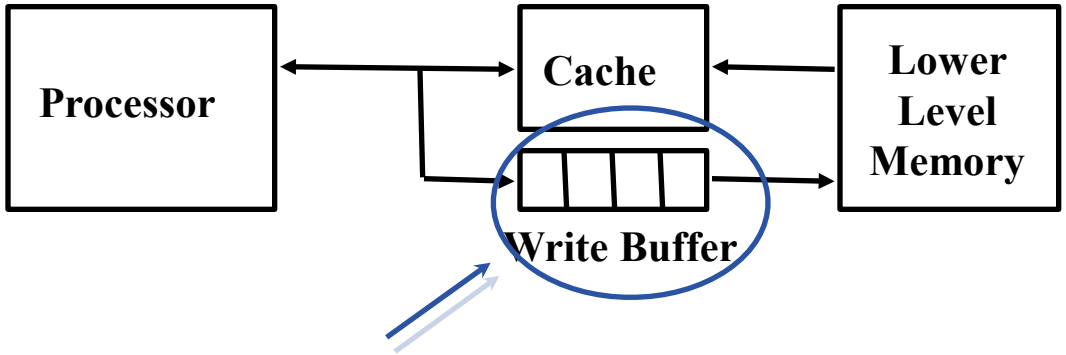
# Write policy example (cont.)

- Write-through cache produces 3 memory writes
  - Not strictly necessary—as long as data remains in cache, most up-to-date value is available at top of hierarchy
  - Accesses never go to memory until block is evicted
- In write-back cache, only write data to memory on eviction
  - Check dirty bit to see if write back is needed
    - Bit is set on every write to block
    - Bit is cleared when block is evicted and new data brought into line
  - After first store:

| Valid | Dirty | Tag | Block | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 | 100 | 105 | 0 | 0 |

  - Second, third store proceed in same way
  - Difference is that no writes to memory occur in this sequence!

# Write Buffers for Write-Through Caches




**Holds data awaiting write-through to lower level memory**

**Q. Why a write buffer ?** **A. So CPU doesn't stall**

**Q. Why a buffer, why not just one register ?** **A. Bursts of writes are common.**

**Q. Are Read After Write (RAW) hazards an issue for write buffer?** **A. Yes! Drain buffer before next read, or send read 1st after check write buffers.**

# Direct-mapped cache example: access #3

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 4 = $0100_2$**
**→ Tag = 0**
**→ Index = 10**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 0 | 0 | 0 | 0 | 0 |
| **1** | **1** | **0** | **18** | **150** |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: **3**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #4

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 13 = $1101_2$**
**→ Tag = 1**
**→ Index = 10**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 18 | 150 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: 3

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #4

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

Address = 13 = $1101_2$
Tag = 1
Index = 10
Offset = 1

Registers:
$t0 = 29, $t1 = 18

Must write back dirty block

Cache

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 18 | 150 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: 4

Memory

| | |
|---|---|
| 0 | 78 |
| 1 | 29 |
| 2 | 120 |
| 3 | 123 |
| 4 | 18 |
| 5 | 150 |
| 6 | 162 |
| 7 | 173 |
| 8 | 18 |
| 9 | 21 |
| 10 | 33 |
| 11 | 28 |
| 12 | 19 |
| 13 | 200 |
| 14 | 210 |
| 15 | 225 |

# Direct-mapped cache example: access #4

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 13 = $1101_2$**
**Tag = 1**
**Index = 10**
**Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 0 | 0 | 0 | 0 | 0 |
| **1** | **1** | **1** | **19** | **29** |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: **4**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #5

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 9 = $1001_2$**
**→ Tag = 1**
**→ Index = 00**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 0
Misses: 4

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #5

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 9 = $1001_2$**
**→ Tag = 1**
**→ Index = 00**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 21**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 1
Misses: 4

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Additional examples

- Given the final cache state above, determine the new cache state after the following three accesses:
  - `lb $t1, 3($zero)`
  - `sb $t0, 2($zero)`
  - `lb $t0, 11($zero)`

# Direct-mapped cache example: access #6

Instructions:

```
lb $t1, 3($zero)
sb $t0, 2($zero)
lb $t1, 11($zero)
```

**Address = 3 = $0011_2$**
**→ Tag = 0**
**→ Index = 01**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 1
Misses: 4

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #6

Instructions:

```
lb $t1, 3($zero)
sb $t0, 2($zero)
lb $t1, 11($zero)
```

**Address = 3 = $0011_2$**
**→ Tag = 0**
**→ Index = 01**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 123**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| **1** | **0** | **0** | **120** | **123** |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 1
Misses: **5**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #7

Instructions:

```
lb $t1, 3($zero)
sb $t0, 2($zero)
lb $t1, 11($zero)
```

**Address = 2 = $0010_2$**
**→ Tag = 0**
**→ Index = 01**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 123**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 1 | 0 | 0 | 120 | 123 |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 1
Misses: 5

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #7

**Instructions:**

```
lb $t1, 3($zero)
sb $t0, 2($zero)
lb $t1, 11($zero)
```

**Address = 2 = $0010_2$**
**→ Tag = 0**
**→ Index = 01**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 123**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| **1** | **1** | **0** | **29** | **123** |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 2
Misses: 5

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #8

Instructions:

```
lb $t1, 3($zero)
sb $t0, 2($zero)
lb $t1, 11($zero)
```

**Address = 11 = $1011_2$**
**→ Tag = 1**
**→ Index = 01**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 123**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| 1 | 1 | 0 | 29 | 123 |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 2
Misses: 5

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #8

**Instructions:**

```
lb $t1, 3($zero)
sb $t0, 2($zero)
lb $t1, 11($zero)
```

**Address = 11 = $1011_2$**
**→ Tag = 1**
**→ Index = 01**
**→ Offset = 1**

**Must write back dirty block**

**Registers:**
**$t0 = 29, $t1 = 123**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| **1** | **1** | **0** | **29** | **123** |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 2
Misses: 5

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 29 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Direct-mapped cache example: access #8

Instructions:

```
lb $t1, 3($zero)
sb $t0, 2($zero)
lb $t1, 11($zero)
```

**Address = 11 = $1011_2$**
**→ Tag = 1**
**→ Index = 01**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 28**

**Cache**

| V | D | Tag | Data | |
|---|---|---|---|---|
| 1 | 0 | 1 | 18 | 21 |
| **1** | **0** | **1** | **33** | **28** |
| 1 | 1 | 1 | 19 | 29 |
| 0 | 0 | 0 | 0 | 0 |

Hits: 2
Misses: **6**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 29 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Replacement policies: review

- On cache miss, bring requested data into cache
  - If line contains valid data, that data is evicted
- When we need to evict a line, what do we choose?
  - Easy choice for direct-mapped—only one possibility!
  - For set-associative or fully-associative, choose least recently used (LRU) line
    - Want to choose data that is least likely to be used next
    - Temporal locality suggests that's the line that was accessed farthest in the past

# LRU example

- Given:
  - 4-way set associative cache
  - Five blocks (A, B, C, D, E) that all map to the same set
- In each sequence below, access to block E is a miss that causes another block to be evicted from the set.
- If we use LRU replacement, which block is evicted?
  - A, B, C, D, **E**
  - A, B, C, D, B, C, A, D, A, C, D, B, A, **E**
  - A, B, C, D, C, B, A, C, A, C, B, **E**

# LRU example solution

- In each case, determine which of the four accessed blocks is least recently used
- Note that you will frequently have to look at more than the last four accesses
  - **A**, B, C, D, **E** → evict A
  - A, B, C, D, B, C, A, D, A, **C**, D, B, A, **E** → evict C
  - A, B, C, **D**, C, B, A, C, A, C, B, **E** → evict D

# Set associative cache example

- Use similar setup to direct-mapped example
  - 2-level hierarchy
  - 16-byte memory
  - Cache organization
    - 8 total bytes
    - 2 bytes per block
    - Write-back cache
    - **One change:** 2-way set associative
- Leads to the following address breakdown
  - Offset: 1 bit
  - Index: 1 bit
  - Tag: 2 bits

# Set associative cache example (cont.)

- Use same access sequence as before

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

# Set associative cache example: initial state

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Registers:**
**$t0 = ?, $t1 = ?**

**MRU = most recently used**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| **Set 0** | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |
| **Set 1** | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #1

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 1 = $0001_2$**
**→ Tag = 00**
**→ Index = 0**
**→ Offset = 1**

**Registers:**
**$t0 = ?, $t1 = ?**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 0

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #1

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 1 = $0001_2$**
**→ Tag = 00**
**→ Index = 0**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = ?**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| **Set 0** | **1** | **0** | **1** | **00** | **78** | **29** |
| | 0 | 0 | 0 | 00 | 0 | 0 |
| **Set 1** | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: **1**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #2

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 8 = $1000_2$**
**→ Tag = 10**
**→ Index = 0**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = ?**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 0 | 1 | 00 | 78 | 29 |
| | 0 | 0 | 0 | 00 | 0 | 0 |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 1

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | **18** |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #2

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 8 = $1000_2$**
**→ Tag = 10**
**→ Index = 0**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 0 | 0 | 00 | 78 | 29 |
| | 1 | 0 | 1 | 10 | 18 | 21 |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 2

**Memory**

| | |
|---|---|
| 0 | 78 |
| 1 | 29 |
| 2 | 120 |
| 3 | 123 |
| 4 | 71 |
| 5 | 150 |
| 6 | 162 |
| 7 | 173 |
| 8 | 18 |
| 9 | 21 |
| 10 | 33 |
| 11 | 28 |
| 12 | 19 |
| 13 | 200 |
| 14 | 210 |
| 15 | 225 |

# Set associative cache example: access #3

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 4 = $0100_2$**
**→ Tag = 01**
**→ Index = 0**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 0 | 0 | 00 | 78 | 29 |
| | 1 | 0 | 1 | 10 | 18 | 21 |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 2

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #3

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 4 = $0100_2$**
**→ Tag = 01**
**→ Index = 0**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 18**

**Evict non-MRU block**
**Not dirty, so no write back**

Hits: 0
Misses: 2

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| **Set 0** | **1** | **0** | **0** | **00** | **78** | **29** |
| | 1 | 0 | 1 | 10 | 18 | 21 |
| **Set 1** | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #3

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 4 = $0100_2$**
**→ Tag = 01**
**→ Index = 0**
**→ Offset = 0**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| **Set 0** | **1** | **1** | **1** | **01** | **18** | **150** |
| | 1 | 0 | **0** | 10 | 18 | 21 |
| **Set 1** | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: **3**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #4

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 13 = $1101_2$**
**→ Tag = 11**
**→ Index = 0**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 1 | 1 | 01 | 18 | 150 |
| | 1 | 0 | 0 | 10 | 18 | 21 |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 3

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #4

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 13 = $1101_2$**
**→ Tag = 11**
**→ Index = 0**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Evict non-MRU block**
**Not dirty, so no write back**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 1 | 1 | 01 | 18 | 150 |
| | **1** | **0** | **0** | **10** | **18** | **21** |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 3

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #4

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 13 = $1101_2$**
**→ Tag = 11**
**→ Index = 0**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

Hits: 0
Misses: **4**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 1 | **0** | 01 | 18 | 150 |
| | **1** | **1** | **1** | **11** | **19** | **29** |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #5

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 9 = $1001_2$**
**→ Tag = 10**
**→ Index = 0**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 1 | 0 | 01 | 18 | 150 |
| | 1 | 1 | 1 | 11 | 19 | 29 |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 4

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 71 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #5

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 9 = $1001_2$**
**→ Tag = 10**
**→ Index = 0**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 18**

**Evict non-MRU block**
**Dirty, so write back**

Hits: 0
Misses: 4

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| **Set 0** | **1** | **1** | **0** | **01** | **18** | **150** |
| | 1 | 1 | 1 | 11 | 19 | 29 |
| **Set 1** | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #5

Instructions:

```
lb   $t0, 1($zero)
lb   $t1, 8($zero)
sb   $t1, 4($zero)
sb   $t0, 13($zero)
lb   $t1, 9($zero)
```

**Address = 9 = $1001_2$**
**→ Tag = 10**
**→ Index = 0**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 21**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | **1** | **0** | **1** | **10** | **18** | **21** |
| | 1 | 1 | **0** | 11 | 19 | 29 |
| Set 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: **5**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Additional examples

- Given the final cache state above, determine the new cache state after the following three accesses:
  - `lb $t1, 3($zero)`
  - `lb $t0, 11($zero)`
  - `sb $t0, 2($zero)`

# Set associative cache example: access #6

**Instructions:**

```
lb $t1, 3($zero)
lb $t0, 11($zero)
sb $t0, 2($zero)
```

**Address = 3 = $0011_2$**
**→ Tag = 00**
**→ Index = 1**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 21**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| **Set 0** | 1 | 0 | 1 | 10 | 18 | 21 |
| | 1 | 1 | 0 | 11 | 19 | 29 |
| **Set 1** | 0 | 0 | 0 | 00 | 0 | 0 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 5

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #6

**Instructions:**

```
lb $t1, 3($zero)
lb $t0, 11($zero)
sb $t0, 2($zero)
```

**Address = 3 = $0011_2$**
**→ Tag = 00**
**→ Index = 1**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 123**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 0 | 1 | 10 | 18 | 21 |
| | 1 | 1 | 0 | 11 | 19 | 29 |
| Set 1 | **1** | **0** | **1** | **00** | **120** | **123** |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: **6**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #7

**Instructions:**

```
lb $t1, 3($zero)
lb $t0, 11($zero)
sb $t0, 2($zero)
```

**Address = 11 = $1011_2$**
**→ Tag = 10**
**→ Index = 1**
**→ Offset = 1**

**Registers:**
**$t0 = 29, $t1 = 123**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 0 | 1 | 10 | 18 | 21 |
| | 1 | 1 | 0 | 11 | 19 | 29 |
| Set 1 | 1 | 0 | 1 | 00 | 120 | 123 |
| | 0 | 0 | 0 | 00 | 0 | 0 |

Hits: 0
Misses: 6

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #7

**Instructions:**

```
lb $t1, 3($zero)
lb $t0, 11($zero)
sb $t0, 2($zero)
```

**Address = 11 = $1011_2$**
**→ Tag = 10**
**→ Index = 1**
**→ Offset = 1**

**Registers:**
**$t0 = 28, $t1 = 123**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| Set 0 | 1 | 0 | 1 | 10 | 18 | 21 |
| | 1 | 1 | 0 | 11 | 19 | 29 |
| Set 1 | 1 | 0 | **0** | 00 | 120 | 123 |
| | **1** | **0** | **1** | **10** | **33** | **28** |

Hits: 0
Misses: **7**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #8

**Instructions:**

```
lb $t1, 3($zero)
lb $t0, 11($zero)
sb $t0, 2($zero)
```

**Address = 2 = $0010_2$**
**→ Tag = 00**
**→ Index = 1**
**→ Offset = 0**

**Registers:**
**$t0 = 28, $t1 = 123**

**Cache**

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| **Set 0** | 1 | 0 | 1 | 10 | 18 | 21 |
| | 1 | 1 | 0 | 11 | 19 | 29 |
| **Set 1** | 1 | 0 | 0 | 00 | 120 | 123 |
| | 1 | 0 | 1 | 10 | 33 | 28 |

Hits: 0
Misses: 7

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

# Set associative cache example: access #8

**Instructions:**

```
lb $t1, 3($zero)
lb $t0, 11($zero)
sb $t0, 2($zero)
```

**Address = 2 = $0010_2$**
**→ Tag = 00**
**→ Index = 1**
**→ Offset = 0**

**Registers:**
**$t0 = 28, $t1 = 123**

**Memory**

| | |
|---|---|
| **0** | 78 |
| **1** | 29 |
| **2** | 120 |
| **3** | 123 |
| **4** | 18 |
| **5** | 150 |
| **6** | 162 |
| **7** | 173 |
| **8** | 18 |
| **9** | 21 |
| **10** | 33 |
| **11** | 28 |
| **12** | 19 |
| **13** | 200 |
| **14** | 210 |
| **15** | 225 |

**Cache**

Hits: **1**
Misses: 7

| | V | D | MRU | Tag | Data | |
|---|---|---|---|---|---|---|
| **Set 0** | 1 | 0 | 1 | 10 | 18 | 21 |
| | 1 | 1 | 0 | 11 | 19 | 29 |
| **Set 1** | **1** | **1** | **1** | **00** | **28** | **123** |
| | 1 | 0 | **0** | 10 | 33 | 28 |

# Final notes

- Next time
  - Virtual memory
  - Cache optimizations
- Reminders
  - HW 5 due today
  - HW 6 to be posted; due 4/17