CMPE 180-90

Lecture – Caches
2

Haluk Katircioglu
Computer Engineering Department
SJSU

Adapted from Chapter 8 slides, Digital Design and Computer Architecture, Second Edition, David Money Harris and Sarah L. Harris, MK

REVIEW

The Principle of Locality – why does a cache work?

Exploit locality to make memory accesses fast

- **Temporal Locality:**
  - Locality in time
  - If data used recently, likely to use it again soon
  - **How to exploit:** keep recently accessed data in higher levels of memory hierarchy
- **Spatial Locality:**
  - Locality in space
  - If data used recently, likely to use nearby data soon
  - **How to exploit:** when access data, bring nearby data into higher levels of memory hierarchy too

REVIEW

Cache - Memory Performance

- **Hit:** data found in that level of memory hierarchy
- **Miss:** data not found (must go to next level)

Hit Rate = # hits / # memory accesses
= 1 – Miss Rate

Miss Rate = # misses / # memory accesses
= 1 – Hit Rate

- **Average memory access time (AMAT):** average time for processor to access data

AMAT = $t_{\text{cache}} + MR_{\text{cache}}[t_{MM} + MR_{MM}(t_{VM})]$

Cache - Memory Performance – example 1

- A program has 2,000 loads and stores
- 1,250 of these data values in cache
- Rest supplied by other levels of memory hierarchy
- **What are the hit and miss rates for the cache?**

Hit Rate = 1250/2000 **= 0.625**

Miss Rate = 750/2000 = **0.375** = 1 – Hit Rate

Cache - Memory Performance – example 2

- Suppose processor has 2 levels of hierarchy: cache and main memory
- t_{cache} = 1 cycle, t_{MM} = 100 cycles
- **What is the AMAT of the program from Example 1?**

$$\textbf{AMAT} = t_{cache} + MR_{cache}(t_{MM})$$
$$= [1 + 0.375(100)] \text{ cycles}$$
$$= \textbf{38.5 cycles}$$

REVIEW

Cache

- Highest level in memory hierarchy
- Fast (typically ~ 1 cycle access time)
- Ideally supplies most data to processor
- Usually holds most recently accessed data

Cache Terminology

- **Capacity (*C*):**
  - number of data bytes in cache
- **Block size (*b*):**
  - bytes of data brought into cache at once
- **Number of blocks (*B* = *C*/*b*):**
  - number of blocks in cache: *B* = *C*/*b*
- **Degree of associativity (*N*):**
  - number of blocks in a set
- **Number of sets (*S* = *B*/*N*):**
  - each memory address maps to exactly one cache set

REVIEW

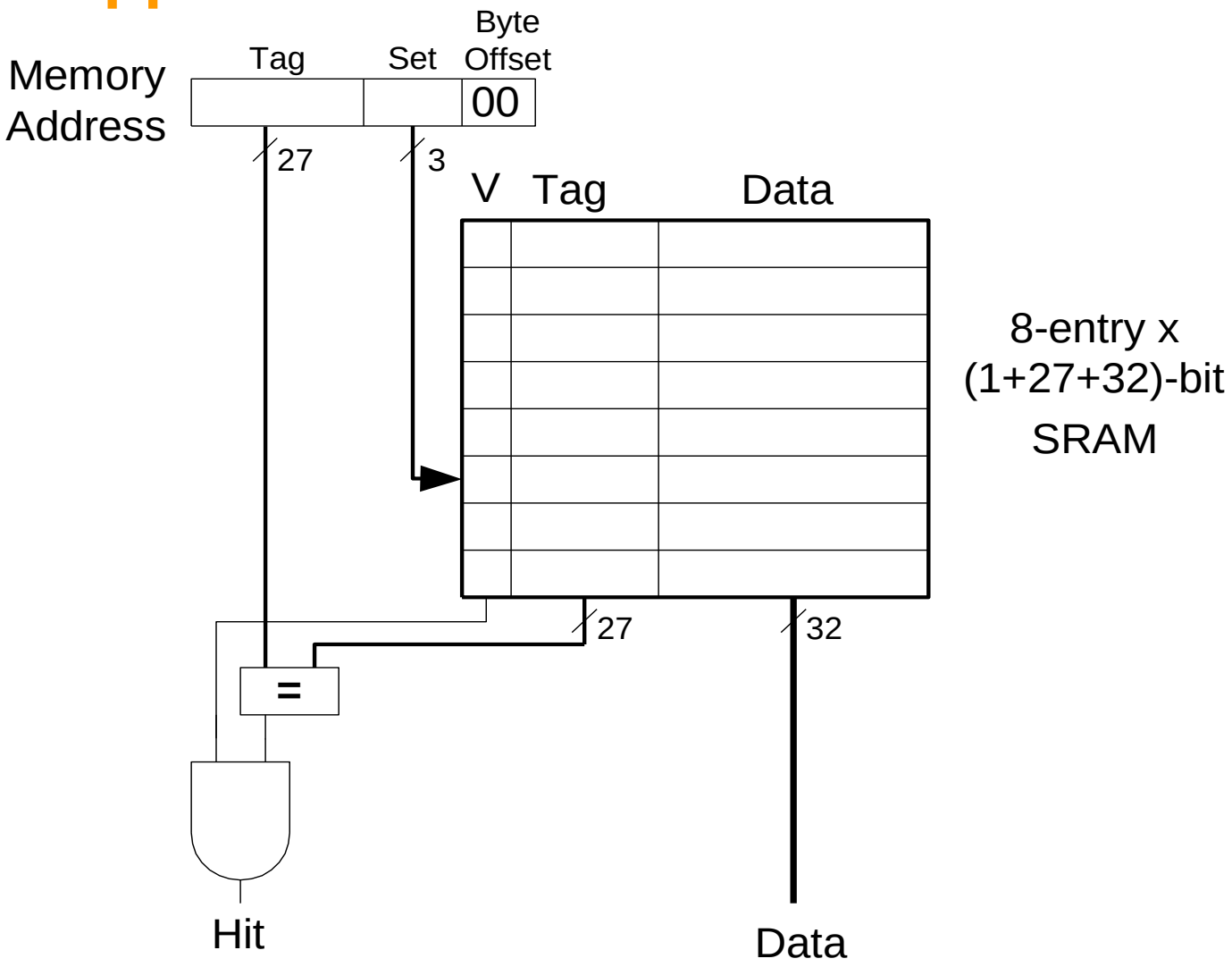
How is data found?

- Cache organized into *S* sets
- Each memory address maps to exactly one set
- Caches categorized by # of blocks in a set:
  - **Direct mapped:** 1 block per set
  - ***N*-way set associative:** *N* blocks per set
  - **Fully associative:** all cache blocks in 1 set

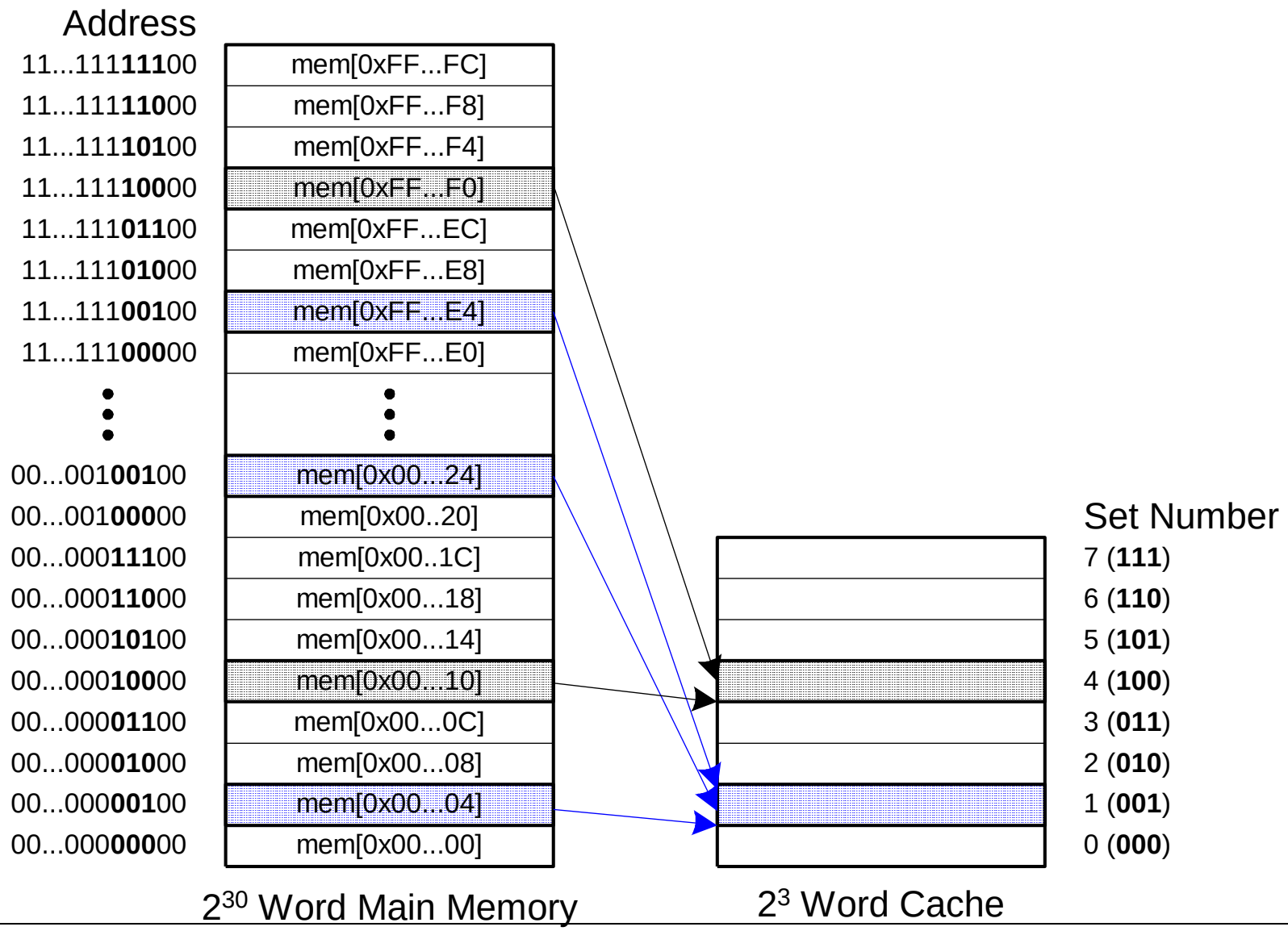
- Examine each organization for a cache with:
  - Capacity (*C* = 8 words)
  - Block size (*b* = 1 word)
  - So, number of blocks (*B* = 8)

REVIEW

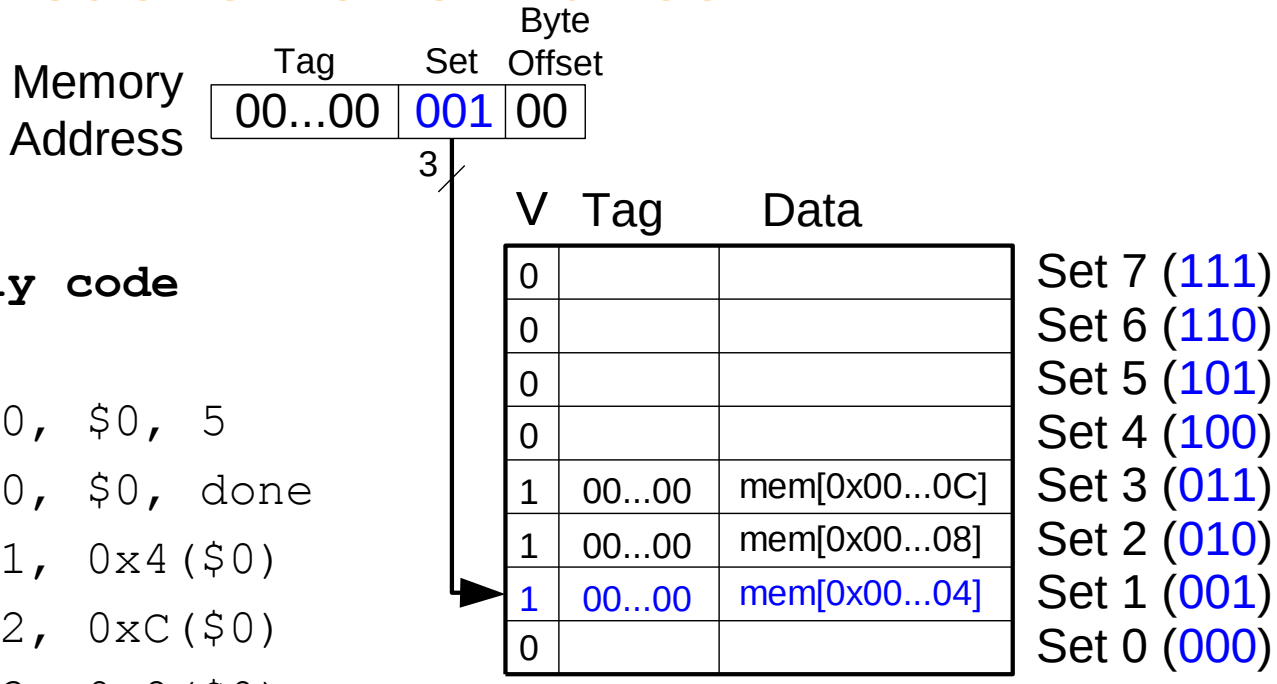
Direct Mapped Cache



Cache Memory 2



Direct Mapped Cache Performance




```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```

Miss Rate = 3/15

= 20%

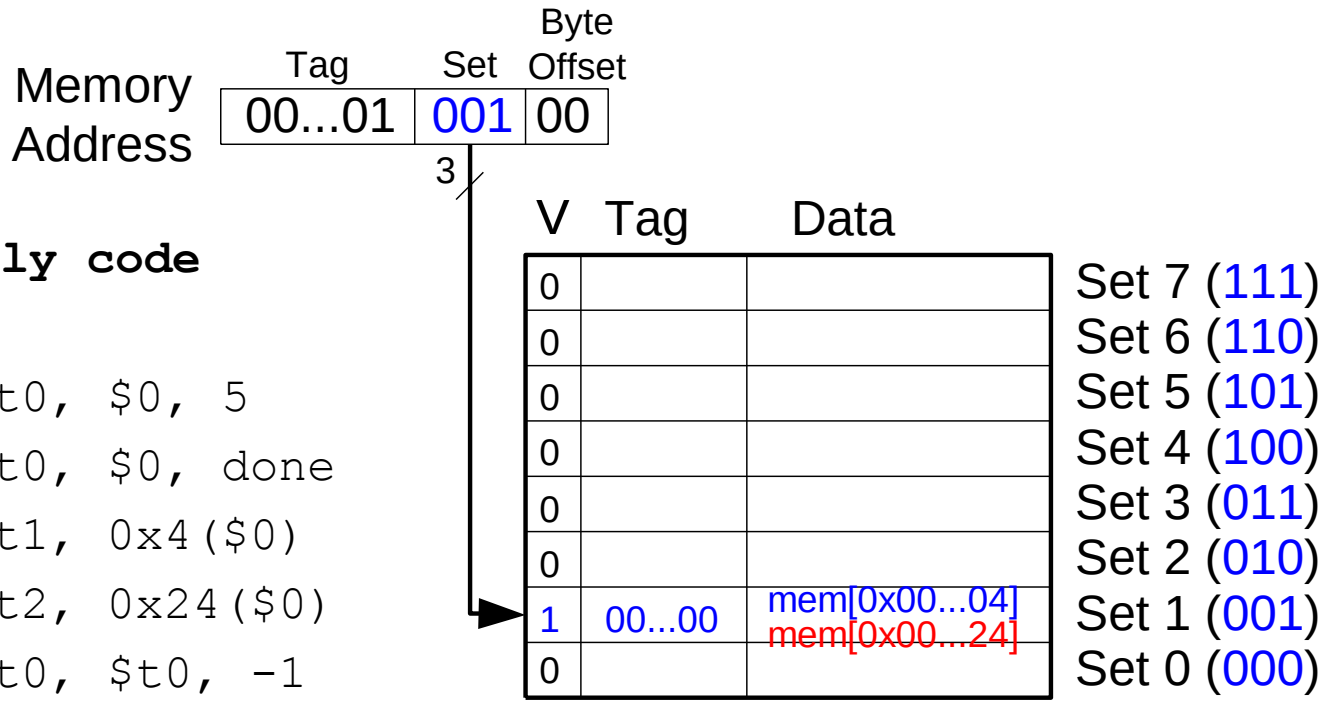
Temporal Locality

Compulsory Misses

Direct Mapped Cache Performance

```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```



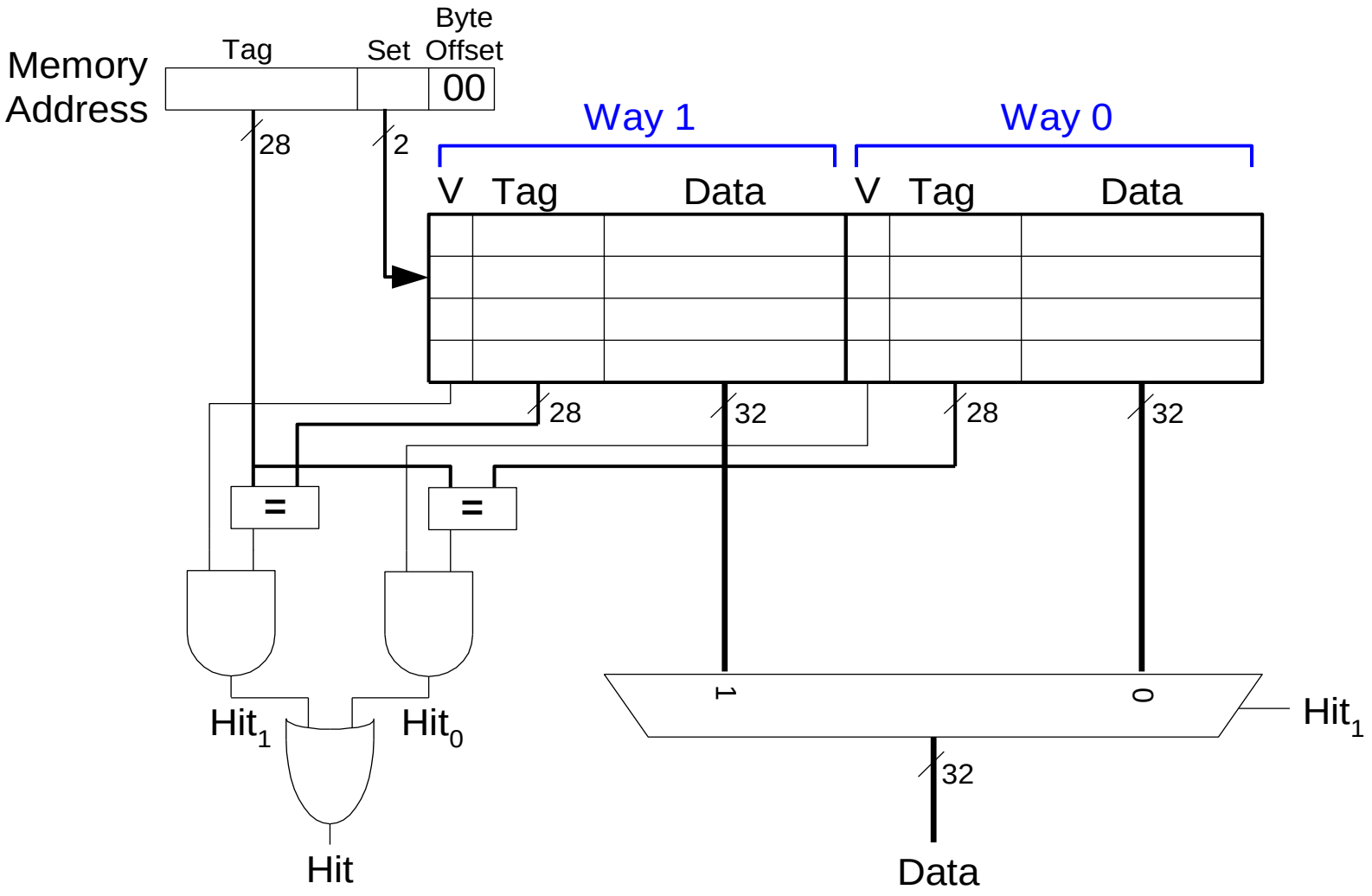
Miss Rate = 10/10

= 100%

Conflict Misses

REVIEW

N-Way Set Associative Cache



N-Way Set Associative Cache Performance

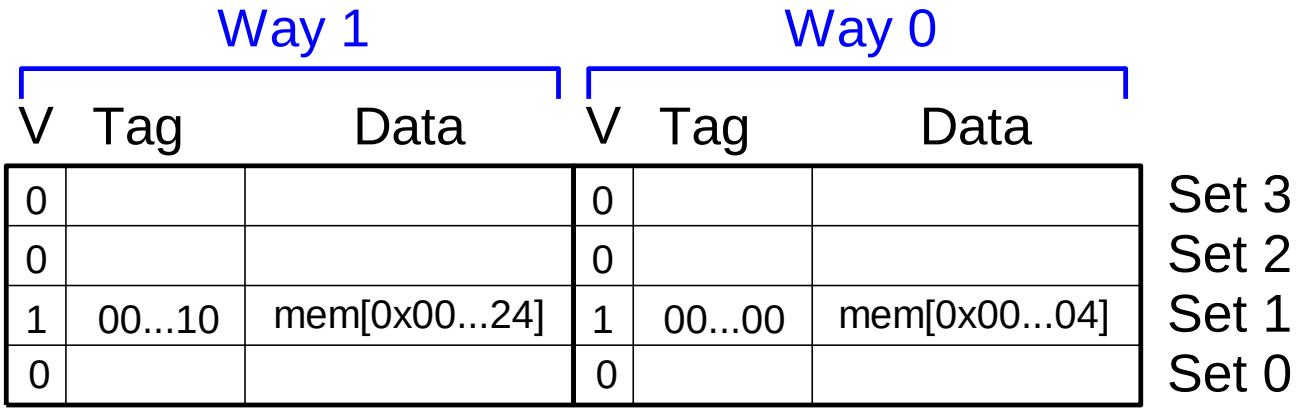
```
# MIPS assembly code

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```

Miss Rate = 2/10

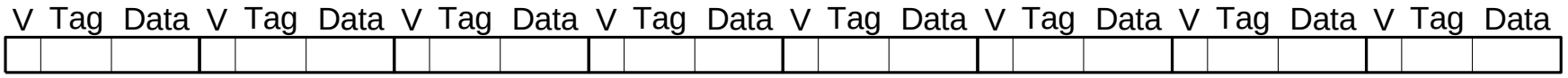
= 20%

Associativity reduces conflict misses



REVIEW

Fully Associative Cache

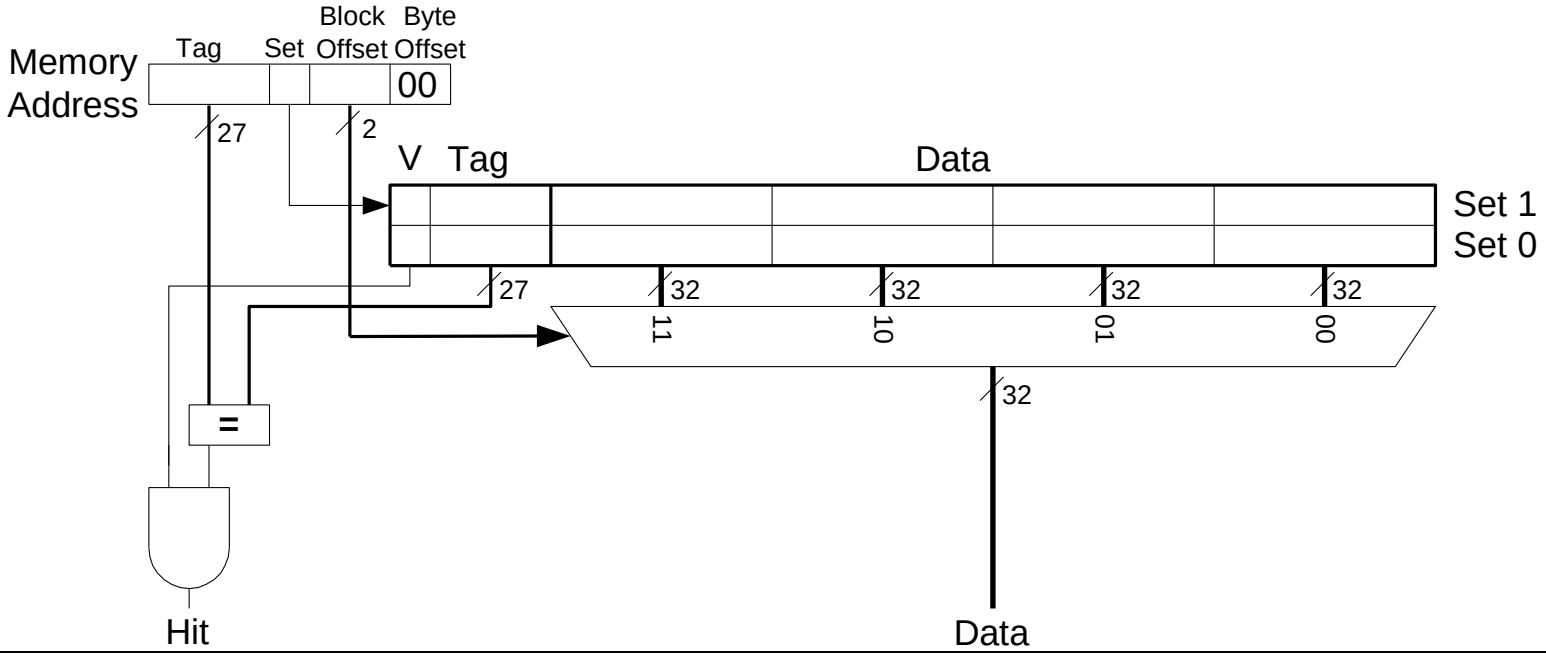


Reduces conflict misses

Expensive to build

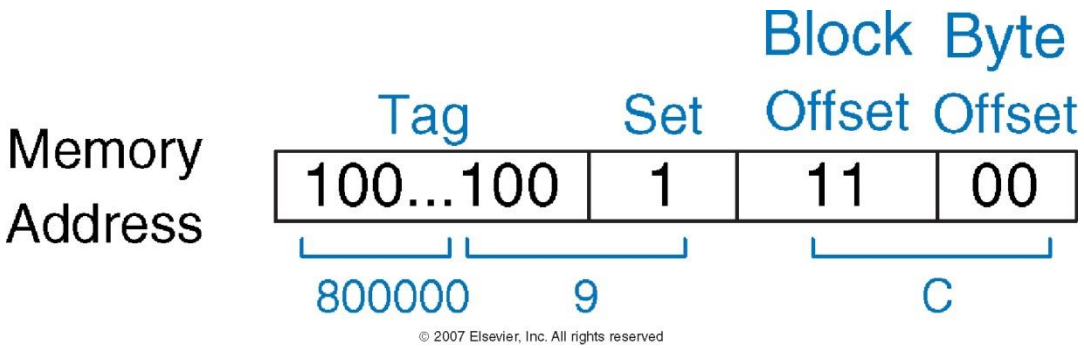
Spatial Locality

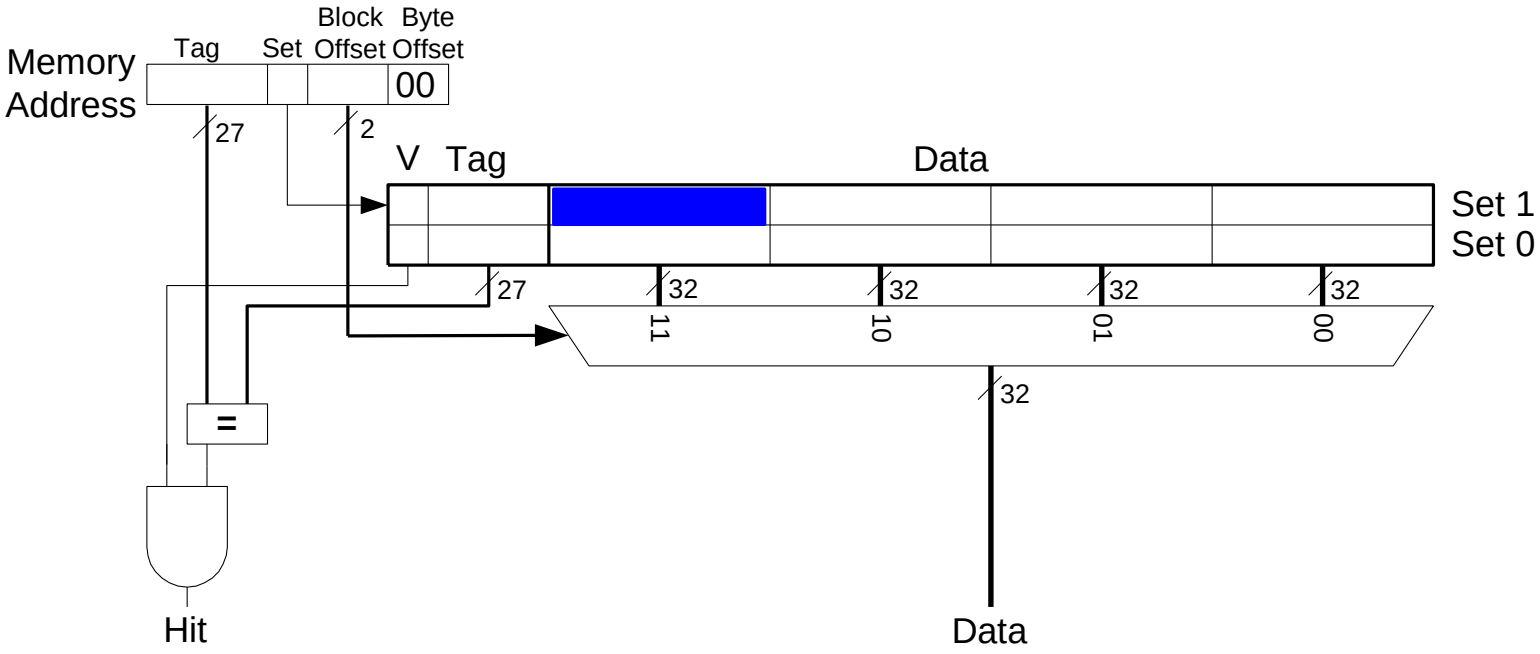
- Increase block size:
  - Block size, ***b* = 4 words**
  - *C* = 8 words
  - Direct mapped (1 block per set)
  - Number of blocks, ***B* = 2** (*C*/*b* = 8/4 = 2)



REVIEW

Cache with a Larger Block Size





Direct Mapped Cache Performance

```
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```

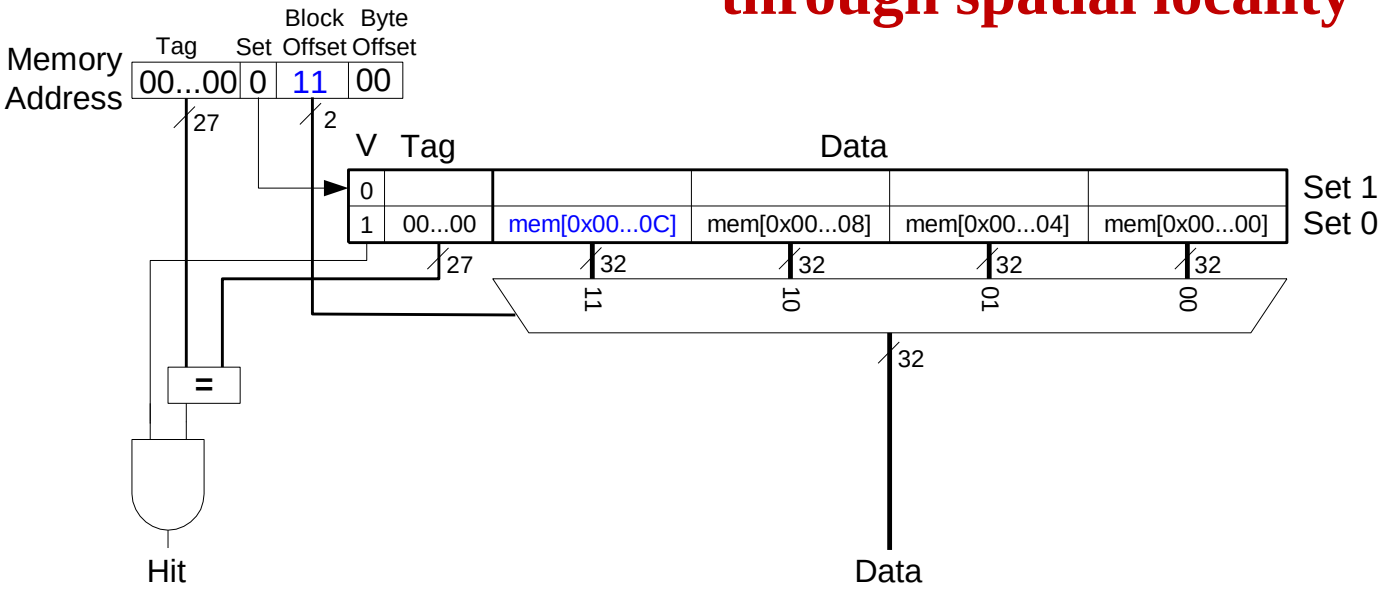
Miss Rate = 1/15

= 6.67%

Larger blocks

reduce compulsory misses

through spatial locality



REVIEW

Capacity Misses

- Cache is too small to hold all data of interest at once
- If cache full: program accesses data X & evicts data Y
- ***Capacity miss*** when access Y again
- How to choose Y to minimize chance of needing it again?
- **Least recently used (LRU) replacement**: the least recently used block in a set evicted

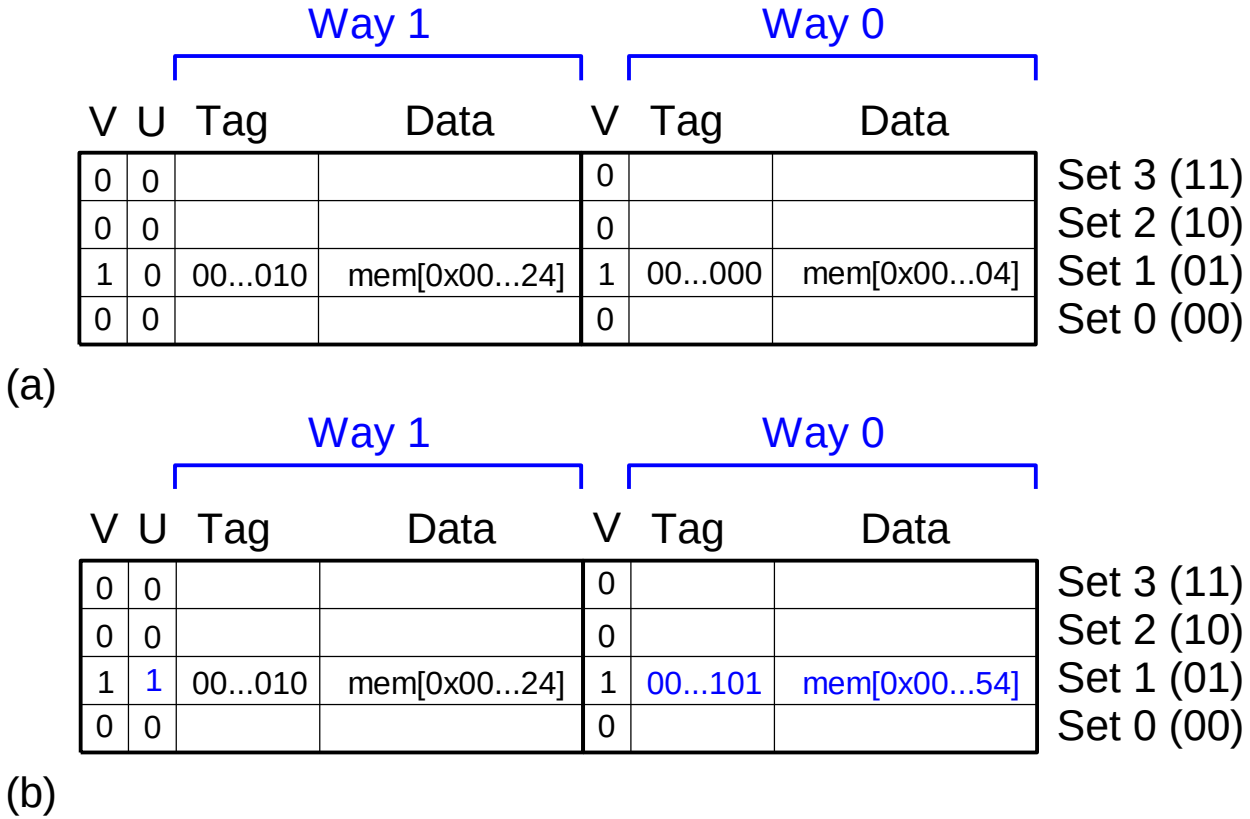
Types of Misses:

- **Compulsory:** first time data accessed
- **Capacity:** cache too small to hold all data of interest
- **Conflict:** data of interest maps to same location in cache

Miss penalty: time it takes to retrieve a block from lower level of hierarchy

LRU Replacement

```
# MIPS assembly
lw $t0, 0x04($0)
lw $t1, 0x24($0)
lw $t2, 0x54($0)
```



REVIEW

SUMMARY

- **What data is held in the cache?**
  - Recently used data (temporal locality)
  - Nearby data (spatial locality)
- **How is data found?**
  - Set is determined by address of data
  - Word within block also determined by address
  - In associative caches, data could be in one of several ways
- **What data is replaced?**
  - Least-recently used way in the set