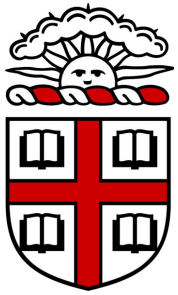
EN164: Design of Computing Systems
Lecture 26: Memory Systems 2

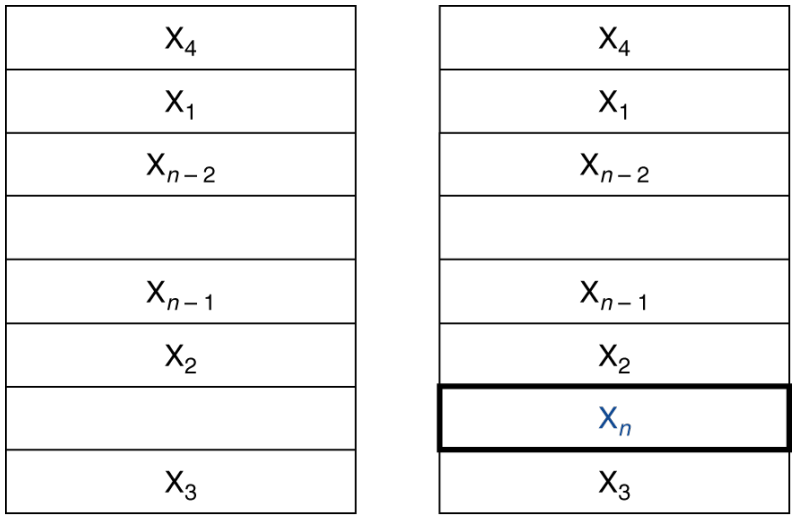
Professor Sherief Reda
http://scale.engin.brown.edu
Electrical Sciences and Computer Engineering
School of Engineering
Brown University
Spring 2011



[material from Patterson & Hennessy, 4th ed and Harris 1st ed]

Cache memory

- Cache memory
  - The level of the memory hierarchy closest to the CPU
  - Fast (typically ~ 1 cycle access time)
  - Made out of SRAM cells
- Given accesses X_1, …, X_{n-1}, X_n

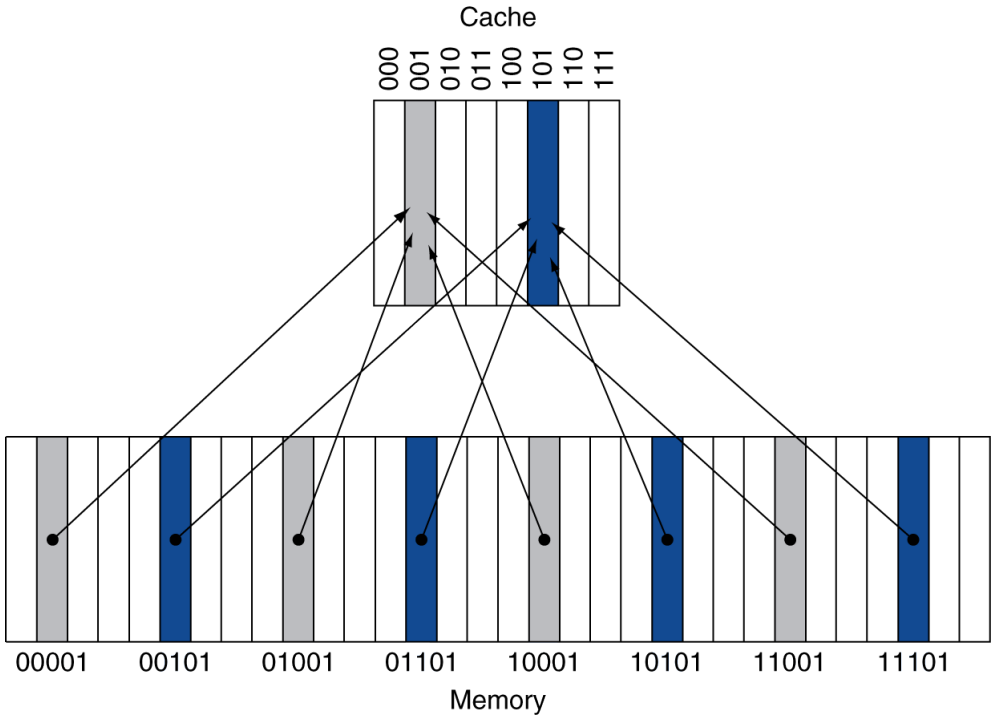


a. Before the reference to X_n b. After the reference to X_n

- What data is held in the cache?
- How is data found?
- What data is replaced?

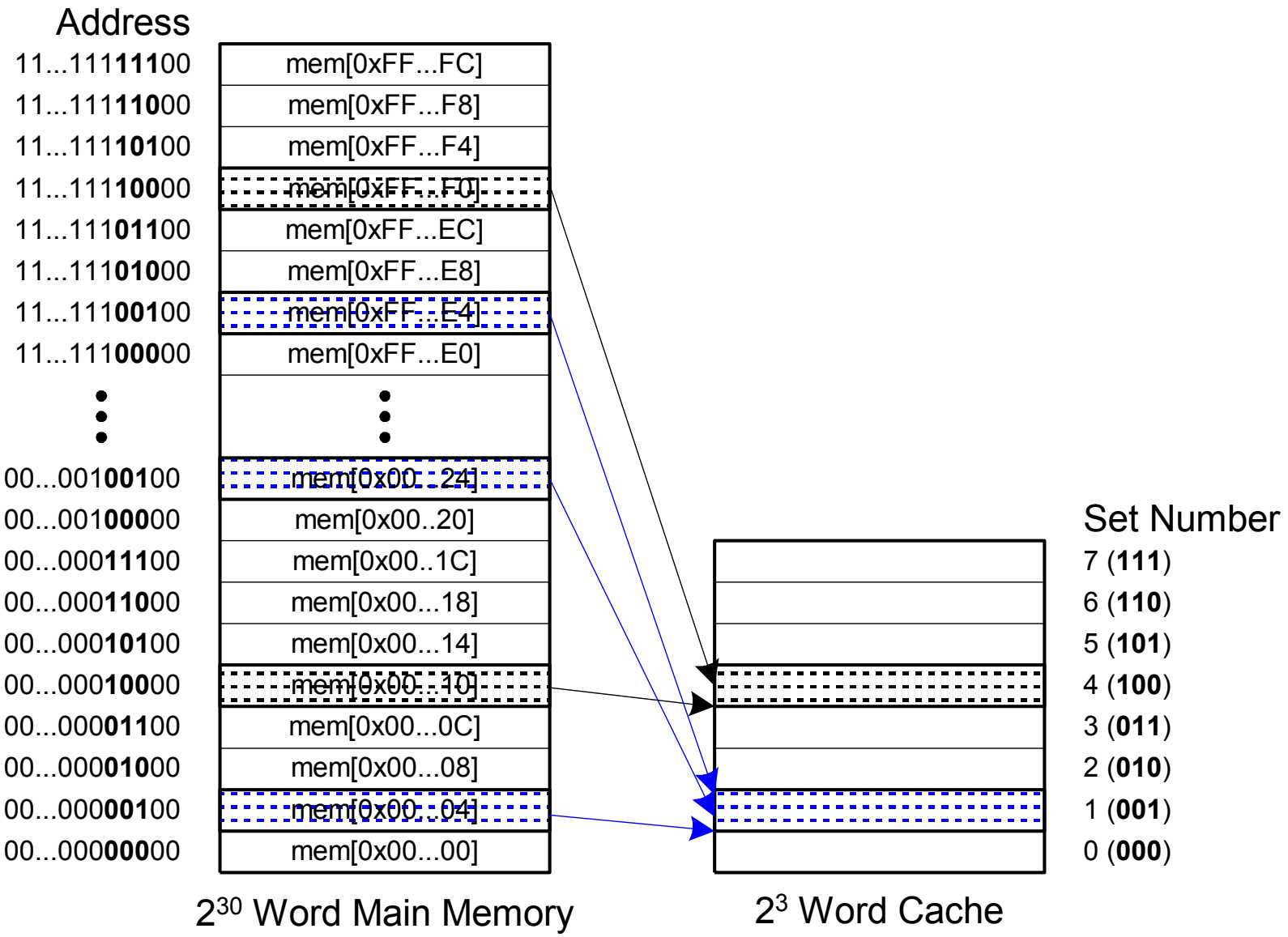
Direct mapped cache

- Location determined by address
- Direct mapped: only one choice
  - (Block address) modulo (#Blocks in cache)



- #Blocks is a power of 2
- Use low-order address bits

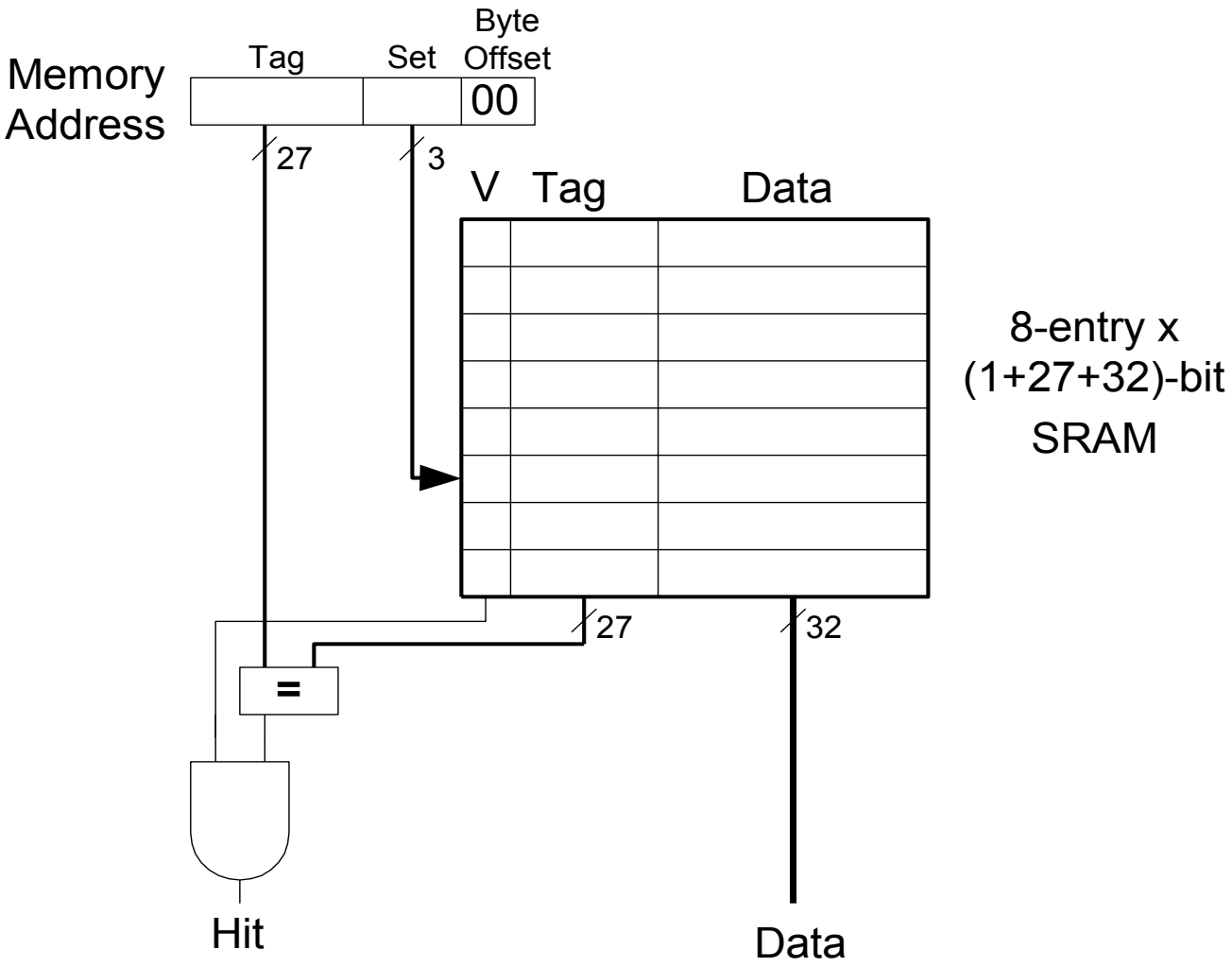
Example


Address
11...111**11**100
11...111**11**000
11...111**10**100
11...111**10**000
11...111**01**100
11...111**01**000
11...111**00**100
11...111**00**000
00...001**00**100
00...001**00**000
00...000**11**100
00...000**11**000
00...000**10**100
00...000**10**000
00...000**01**100
00...000**01**000
00...000**00**100
00...000**00**000
mem[0xFF...FC]
mem[0xFF...F8]
mem[0xFF...F4]
mem[0xFF...F0]
mem[0xFF...EC]
mem[0xFF...E8]
mem[0xFF...E4]
mem[0xFF...E0]
mem[0x00..24]
mem[0x00..20]
mem[0x00..1C]
mem[0x00...18]
mem[0x00...14]
mem[0x00...10]
mem[0x00...0C]
mem[0x00...08]
mem[0x00...04]
mem[0x00...00]
Set Number
7 (**111**)
6 (**110**)
5 (**101**)
4 (**100**)
3 (**011**)
2 (**010**)
1 (**001**)
0 (**000**)
2^{30} Word Main Memory
2^{3} Word Cache

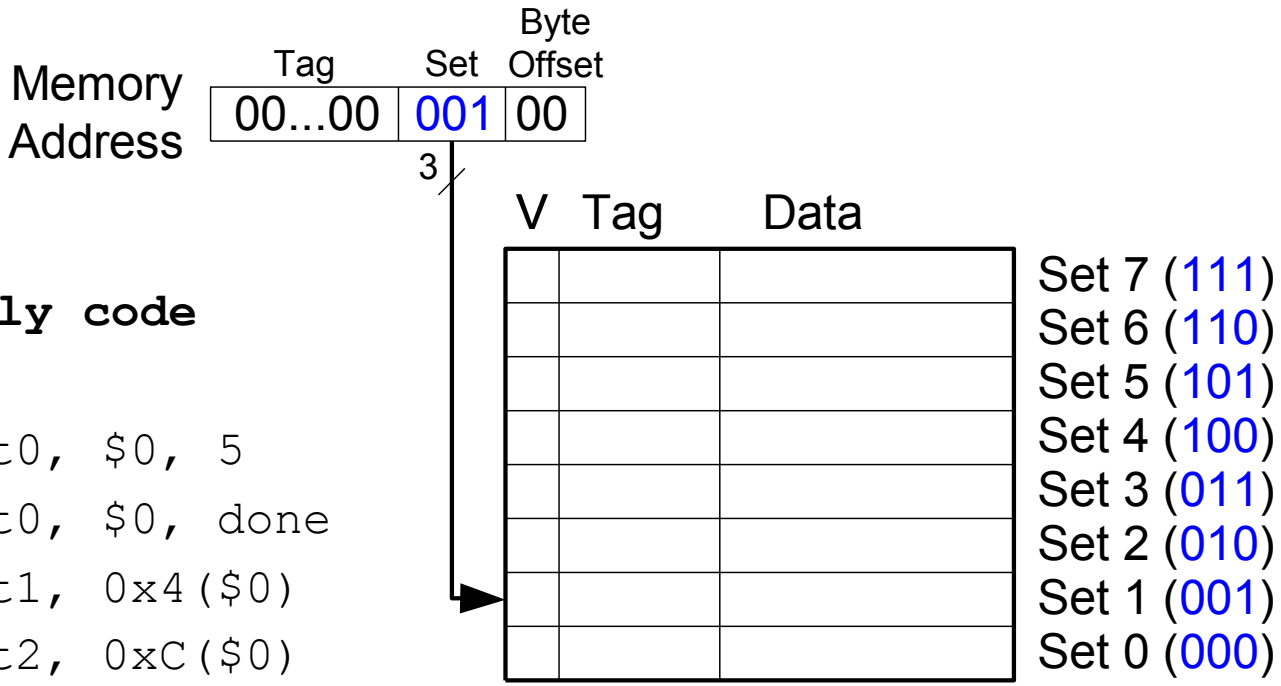
Tag and valid bits

- How do we know which particular block is stored in a cache location?
  - Store block address as well as the data
  - Actually, only need the high-order bits
  - Called the tag
- What if there is no data in a location?
  - Valid bit: 1 = present, 0 = not present
  - Initially 0

Cache hardware design



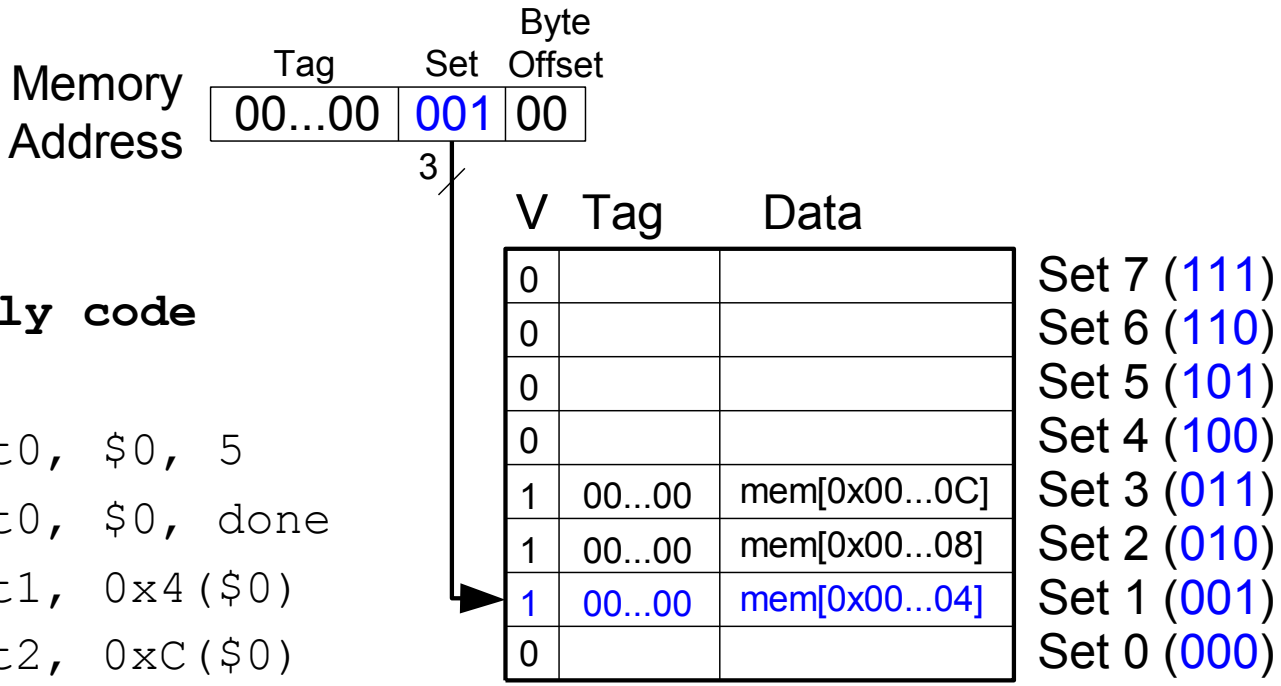
Direct cache performance example




```
# 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:
```

Direct cache performance example




```
# 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:
```

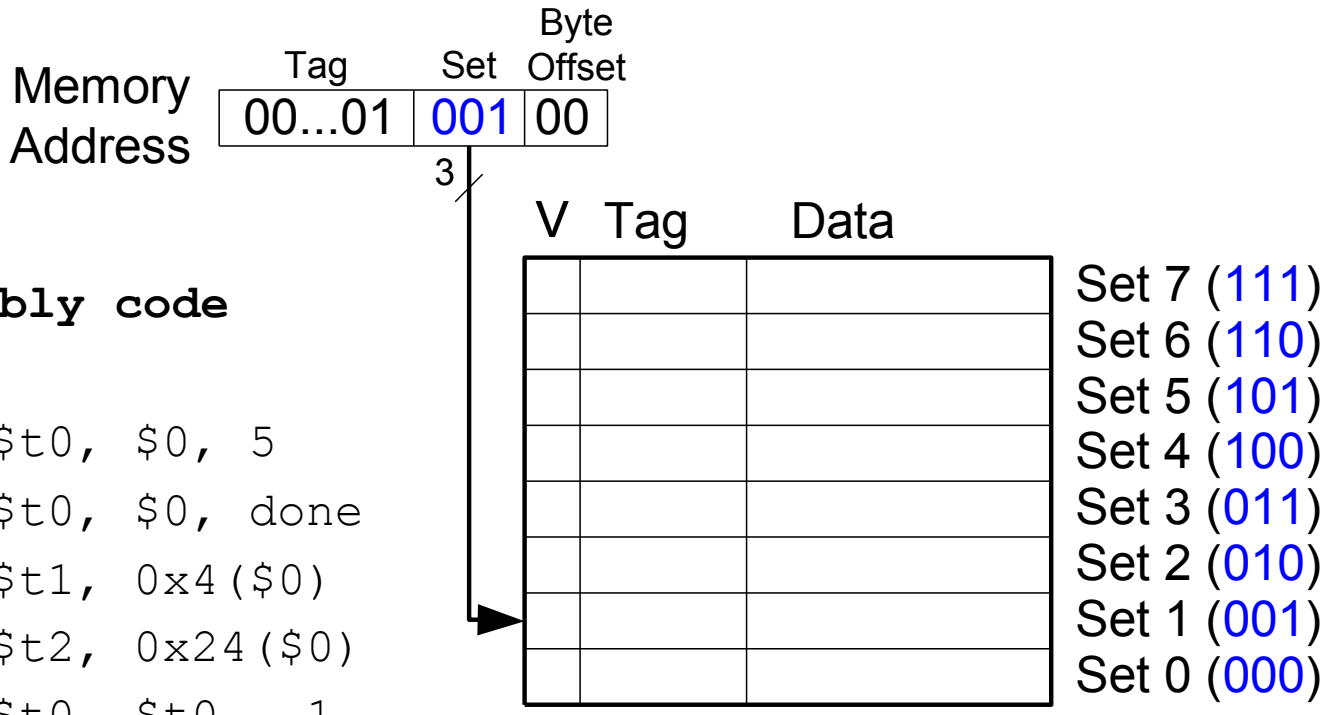
Miss Rate = 3/15

= 20%

Temporal Locality

Compulsory Misses

Direct mapped cache: conflict



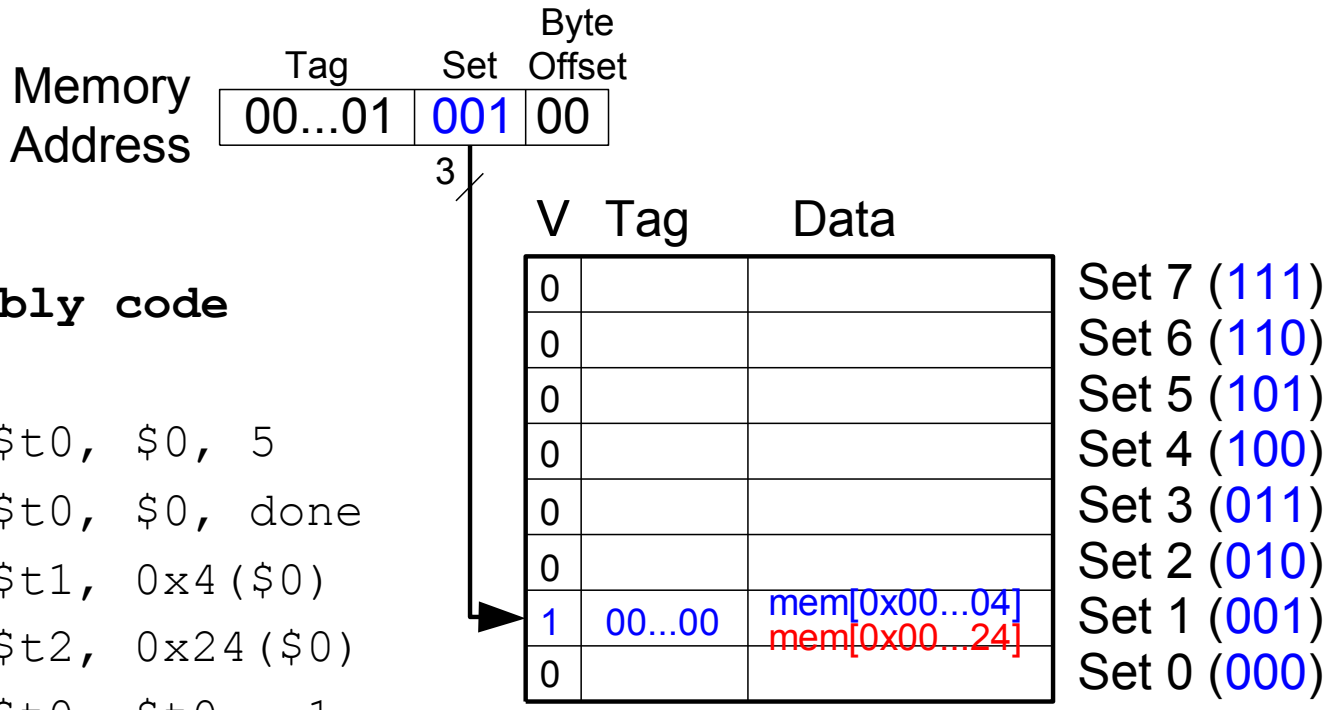

```
# MIPS assembly code

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

Direct cache map: conflict

```
# 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:
```



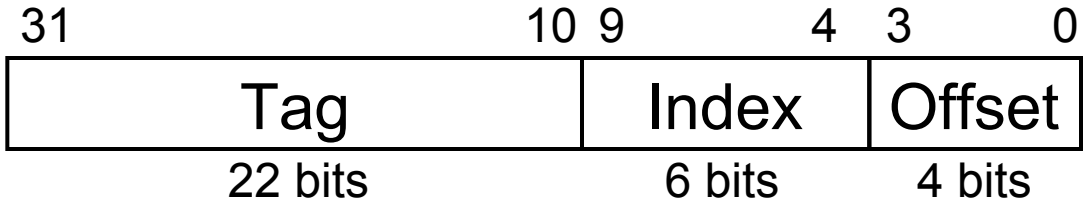
Miss Rate = 10/10

= 100%

Conflict Misses

Impact of larger block sizes

- 64 blocks, 16 bytes/block
  - To what block number does address 1200 map?
- Block address = $\lfloor 1200/16 \rfloor = 75$
- Block number = 75 modulo 64 = 11



Block Size Considerations

- Larger blocks should reduce miss rate
  - Due to spatial locality
- But in a fixed-sized cache
  - Larger blocks ⇒ fewer of them
    - More competition ⇒ increased miss rate