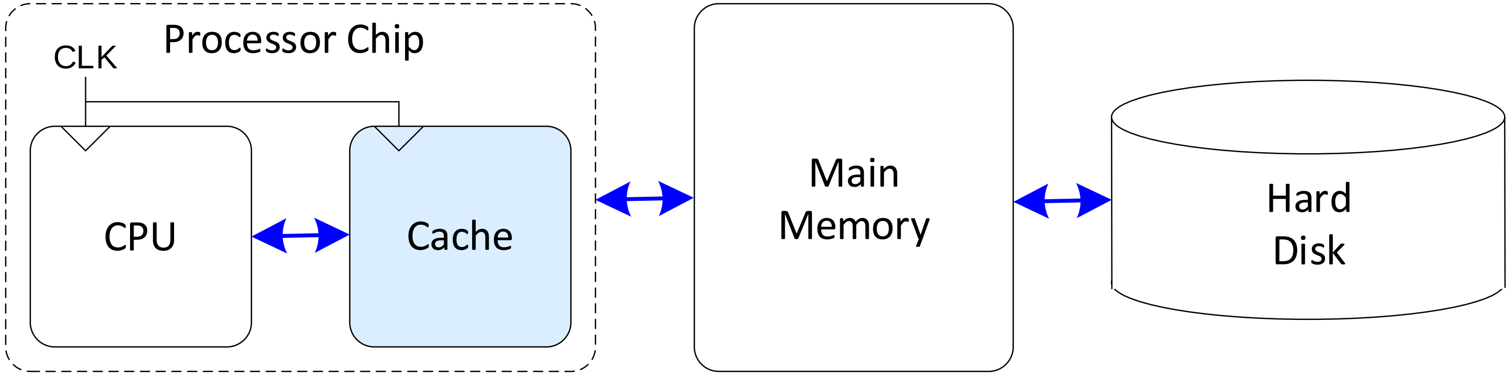
Chapter 7: Microarchitecture

Caches

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 Design Questions

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

We focus on data loads, but stores follow the same principles.

What data is held in the cache?

- Ideally, cache anticipates needed data and puts it in cache
- But impossible to predict future
- Use past to predict future – temporal and spatial locality:
  - **Temporal locality:** copy newly accessed data into cache
  - **Spatial locality:** copy neighboring data into cache too

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

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)

Example Cache Parameters

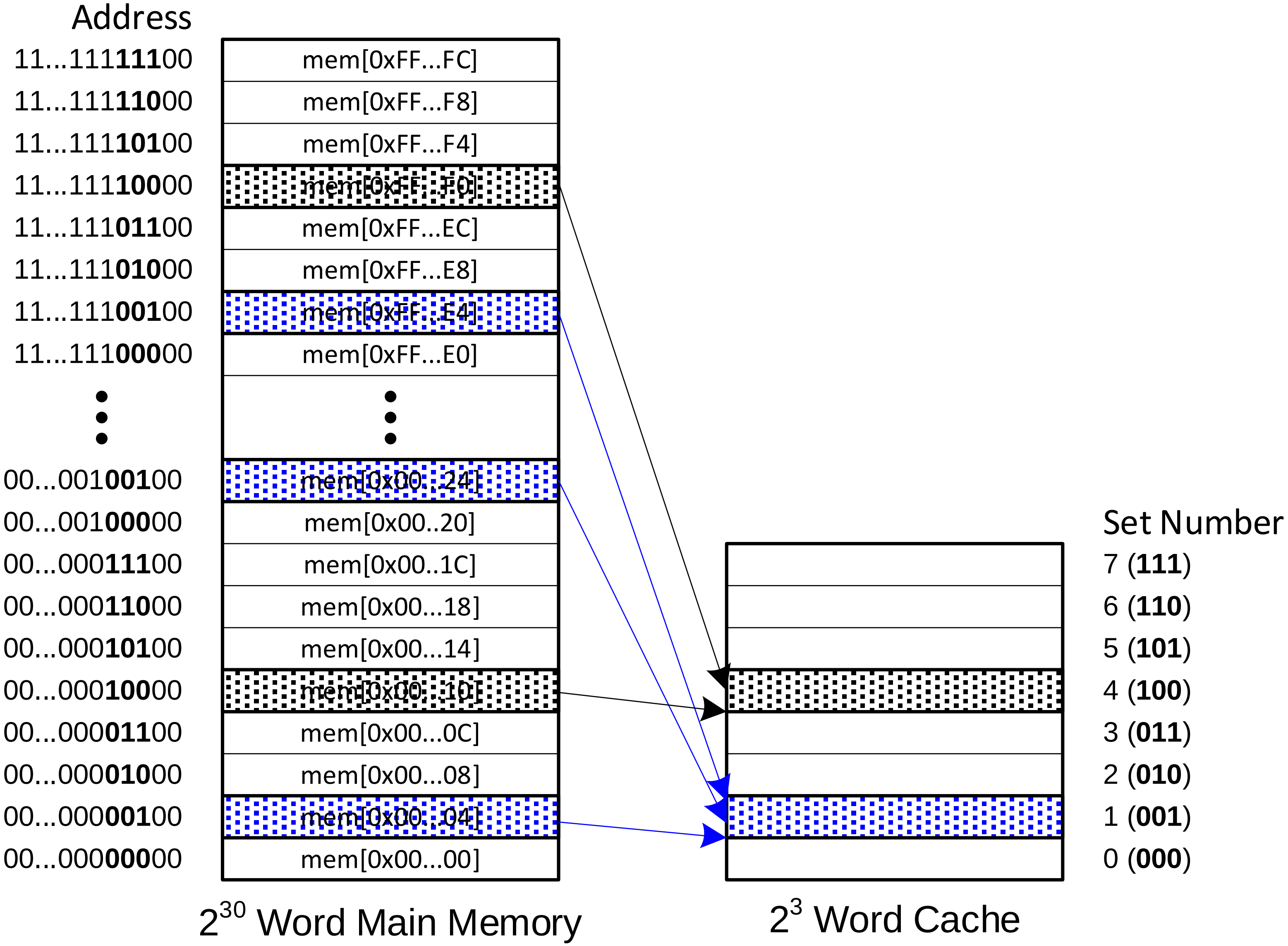
- ***C*** **= 8** words (capacity)
- ***b*** **= 1** word (block size)
- So, ***B*** **= 8** (# of blocks)

Ridiculously small, but will illustrate organizations

Chapter 7: Microarchitecture

Direct-Mapped Caches

Direct Mapped Cache



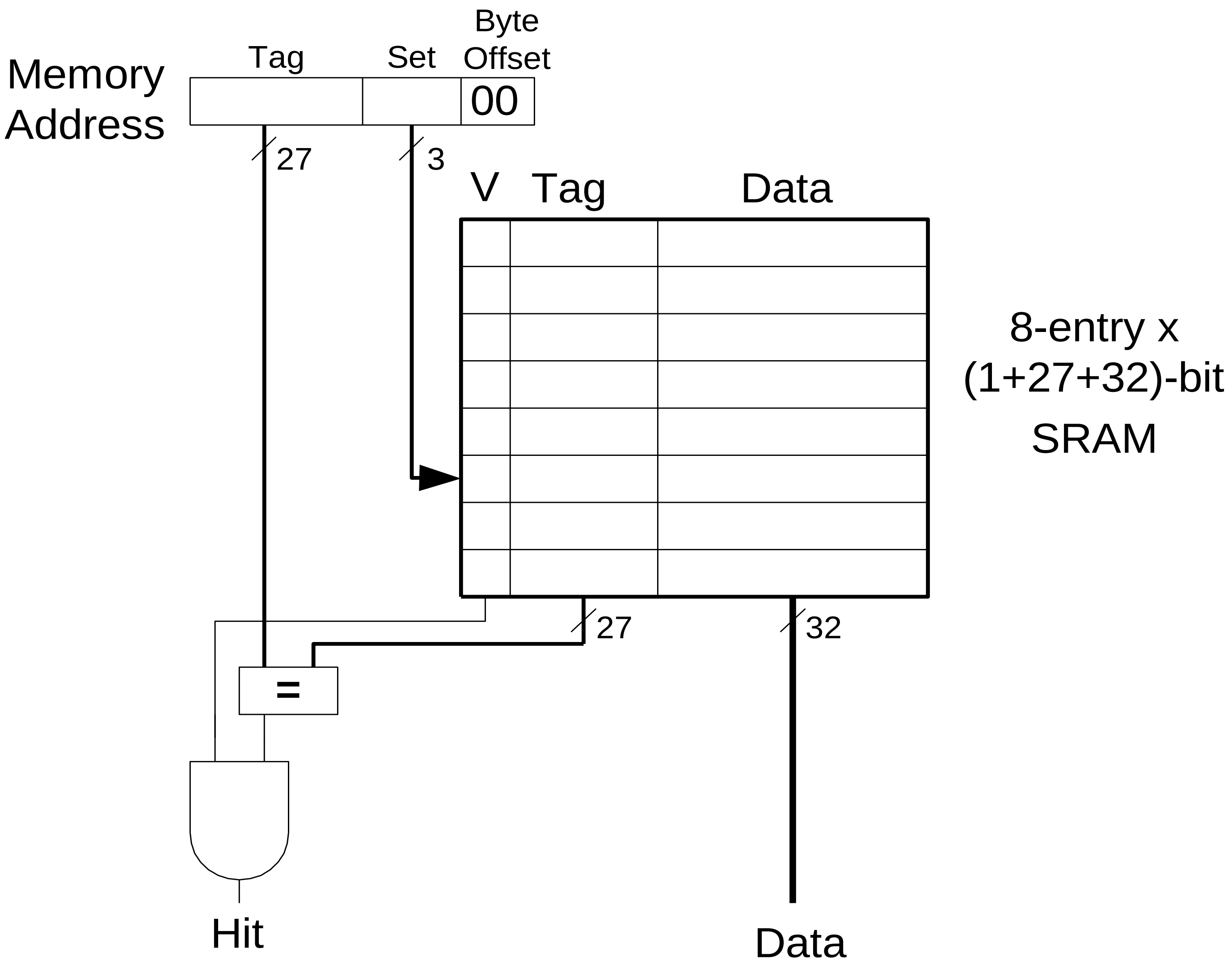
Address	
11...111**111**00	mem[0xFF...FC]
11...111**110**00	mem[0xFF...F8]
11...111**101**00	mem[0xFF...F4]
11...111**100**00	mem[0xFF...F0]
11...111**011**00	mem[0xFF...EC]
11...111**010**00	mem[0xFF...E8]
11...111**001**00	mem[0xFF...E4]
11...111**000**00	mem[0xFF...E0]
⋮	⋮
00...001**001**00	mem[0x00...24]
00...001**000**00	mem[0x00..20]
00...000**111**00	mem[0x00..1C]
00...000**110**00	mem[0x00...18]
00...000**101**00	mem[0x00...14]
00...000**100**00	mem[0x00...10]
00...000**011**00	mem[0x00...0C]
00...000**010**00	mem[0x00...08]
00...000**001**00	mem[0x00...04]
00...000**000**00	mem[0x00...00]

2^{30} Word Main Memory

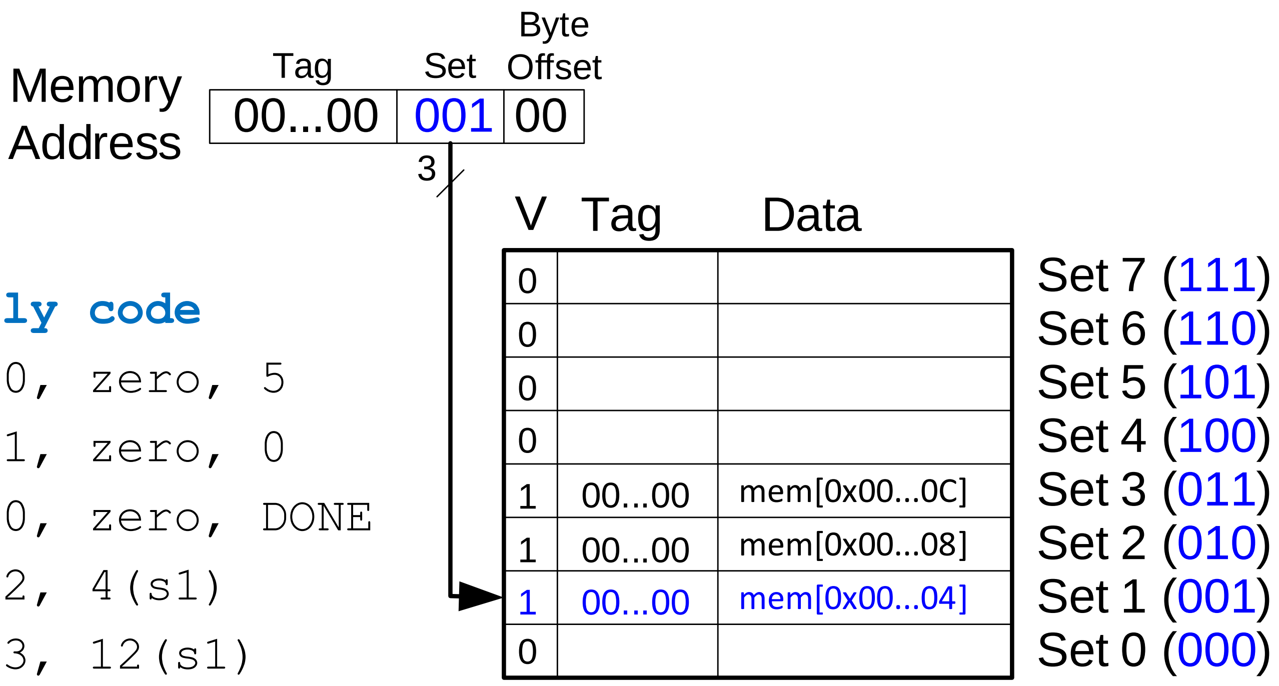
	Set Number
	7 (**111**)
	6 (**110**)
	5 (**101**)
▨	4 (**100**)
	3 (**011**)
	2 (**010**)
▨	1 (**001**)
	0 (**000**)

2^3 Word Cache

Direct Mapped Cache Hardware



Direct Mapped Cache Performance

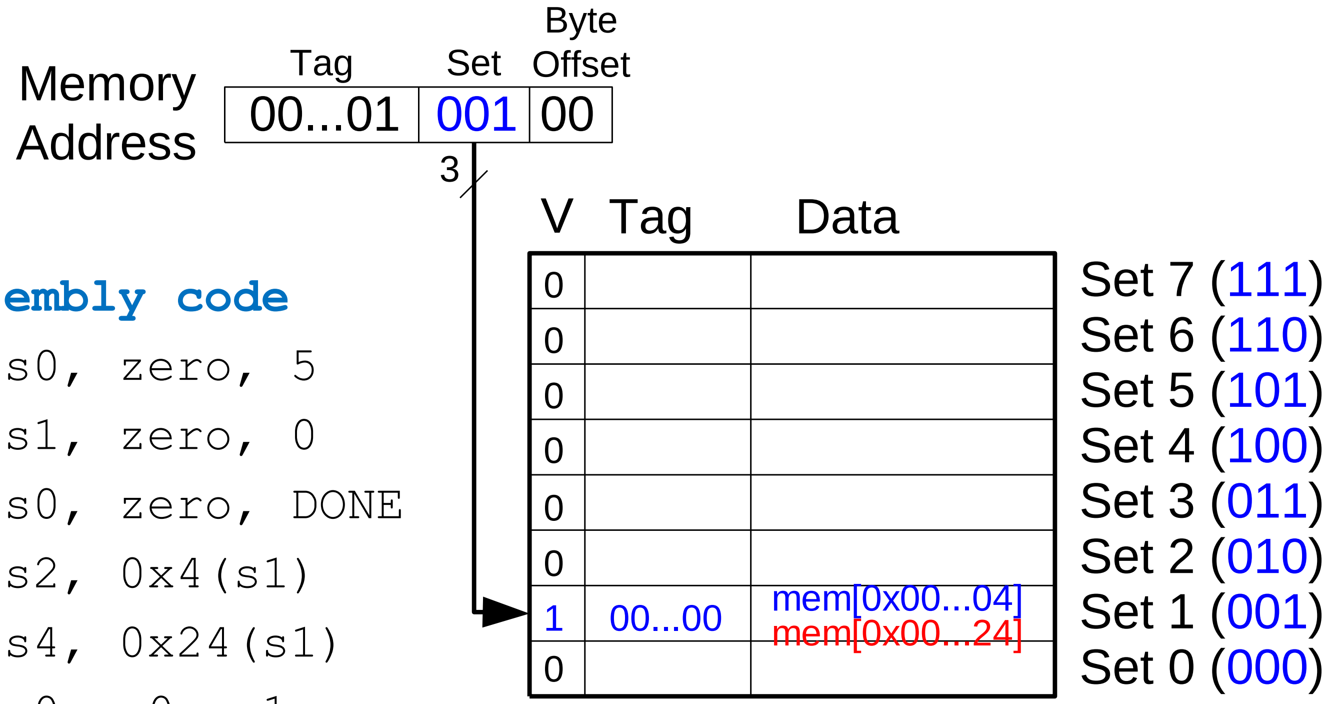


V	Tag	Data	
0			Set 7 (111)
0			Set 6 (110)
0			Set 5 (101)
0			Set 4 (100)
1	00...00	mem[0x00...0C]	Set 3 (011)
1	00...00	mem[0x00...08]	Set 2 (010)
1	00...00	mem[0x00...04]	Set 1 (001)
0			Set 0 (000)

```
# RISC-V assembly code
          addi s0, zero, 5
          addi s1, zero, 0
LOOP:     beq  s0, zero, DONE
          lw   s2, 4(s1)
          lw   s3, 12(s1)
          lw   s4, 8(s1)
          addi s0, s0, -1
          j    LOOP
DONE:
```

Miss Rate

Direct Mapped Cache: Conflict Miss



V	Tag	Data	
0			Set 7 (111)
0			Set 6 (110)
0			Set 5 (101)
0			Set 4 (100)
0			Set 3 (011)
0			Set 2 (010)
1	00...00	mem[0x00...04] mem[0x00...24]	Set 1 (001)
0			Set 0 (000)

```
# RISC-V assembly code
        addi s0, zero, 5
        addi s1, zero, 0
LOOP:   beq  s0, zero, DONE
        lw   s2, 0x4(s1)
        lw   s4, 0x24(s1)
        addi s0, s0, -1
        j    LOOP
DONE:
```

Miss Rate