Topic 11

Memory Hierarchy

- Cache (3)

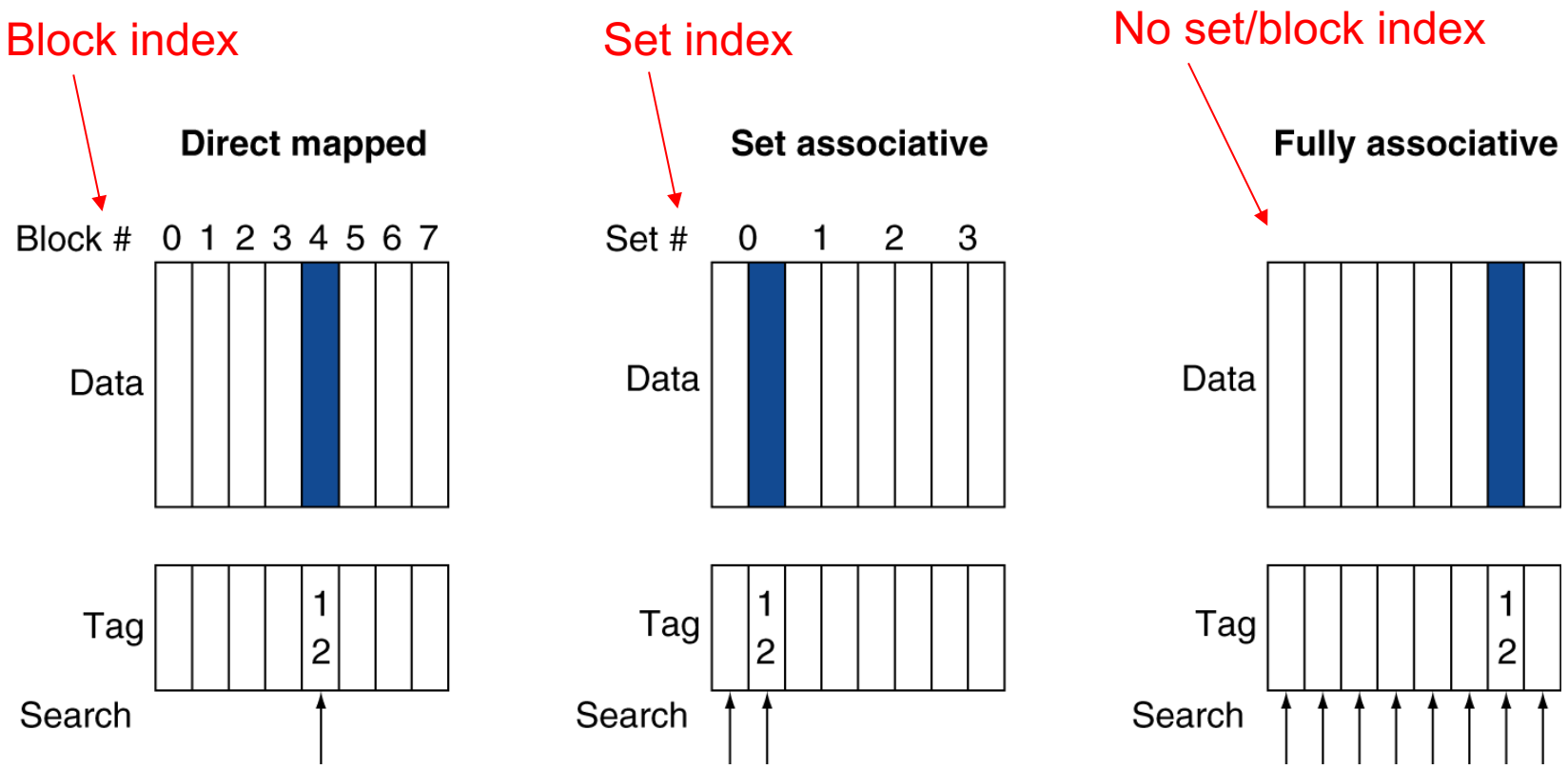
Improve Performance – Associative Caches

- *n*-way **set associative** cache
  - Each set contains *n* blocks
  - A main memory block can use any block within the corresponding set
  - Each address maps to a unique **set** (not block)
    - Set index = (Block address) % (number of sets in cache)
  - However, to locate a block in a set, need to compare n times
    - all *n* tags in a set must be checked and compared
    - *n* comparators (more effective - faster)

Associative Caches

- Fully associative – opposite extreme of direct mapped
  - Allow a given block to go in any cache entry
  - Must search all entries to find a hit
  - One comparator each block (expensive)
    - # of comparator = cache size (block number)

Associative Cache Example



Spectrum of Associativity

- For a cache with 8 blocks

One-way set associative (direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Associativity Example

- Compare caches of 4 two-word blocks
  - Direct mapped, 2-way set associative, fully associative
  - Block access sequence: 0, 8, 0, 12, 8

Associativity Example

Direct mapped (1-way associative)

Associativity Example

- Direct mapped (1-way associative)

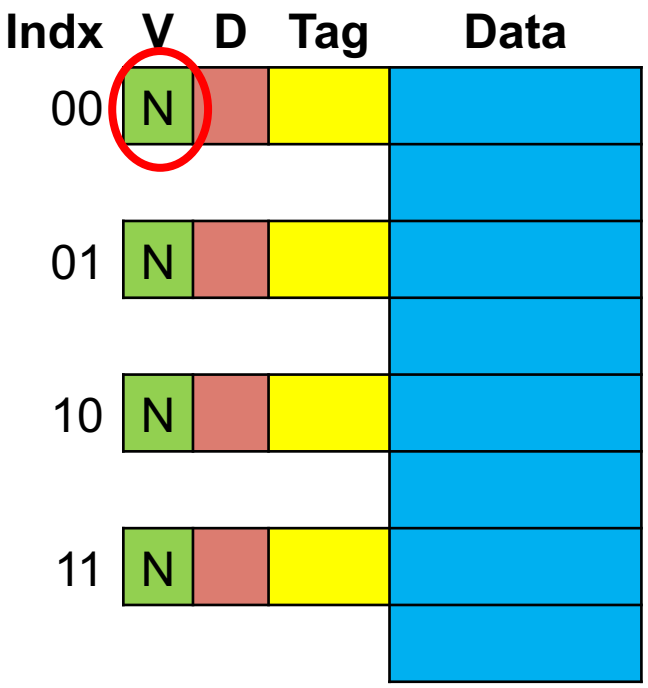
Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	00 00 0	miss	00

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	23
R4	87
R5	62
R6	99
R7	135
…	…

CPU



Miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	00 00 0	miss	00

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	23
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	0	00	110
				120
01	N			
10	N			
11	N			

Fetch

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	00 00 0	hit	00

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	110
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	0	00	110
				120
01	N			
10	N			
11	N			

Load again

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	01 00 0	miss	00

m
m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	110
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	0	00	110
				120
01	N			
10	N			
11	N			

miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	01 00 0	miss	00

m `lw R3←mem[0]`
m **`lw R4←mem[8]`**
`sw R5→mem[0]`
`lw R6←mem[12]`
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	0	01	110→3
				120→300
01	N			
10	N			
11	N			

Replace

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	01 00 0	hit	00

m
m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	0	01	3
				300
01	N			
10	N			
11	N			

Load again

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	00 00 0	miss	00

m `lw R3←mem[0]`
m `lw R4←mem[8]`
m **`sw R5→mem[0]`**
`lw R6←mem[12]`
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	0	01	3
				300
01	N			
10	N			
11	N			

Miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	00 00 0	miss	00

m `lw R3←mem[0]`
m `lw R4←mem[8]`
m **`sw R5→mem[0]`**
`lw R6←mem[12]`
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	0	00	3→110
				300→120
01	N			
10	N			
11	N			

Replace

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	00 00 0	hit	00

m `lw R3←mem[0]`
m `lw R4←mem[8]`
m **`sw R5→mem[0]`**
`lw R6←mem[12]`
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	1	00	110→62
				120
01	N			
10	N			
11	N			

Write, set dirty

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	01 10 0	miss	10

m `lw R3←mem[0]`
m `lw R4←mem[8]`
m `sw R5→mem[0]`
m **`lw R6←mem[12]`**
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	1	00	62
				120
01	N			
10	N			
11	N			

Miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	01 10 0	miss	10

m `lw R3←mem[0]`
m `lw R4←mem[8]`
m `sw R5→mem[0]`
m **`lw R6←mem[12]`**
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	1	00	62
				120
01	N			
10	Y	0	01	234
				912
11	N			

Fetch

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	01 10 0	hit	10

m `lw R3←mem[0]`
m `lw R4←mem[8]`
m `sw R5→mem[0]`
m **`lw R6←mem[12]`**
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	234
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	1	00	62
				120
01	N			
10	Y	0	01	234
				912
11	N			

Load again

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	01 00 0	miss	00

m `lw R3←mem[0]`
m `lw R4←mem[8]`
m `sw R5→mem[0]`
m `lw R6←mem[12]`
m **`sw R7→mem[8]`**

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	234
R7	135
…	…

CPU

Indx	V	D	Tag	Data
00	Y	1	00	62
				120
01	N			
10	Y	0	01	234
				912
11	N			

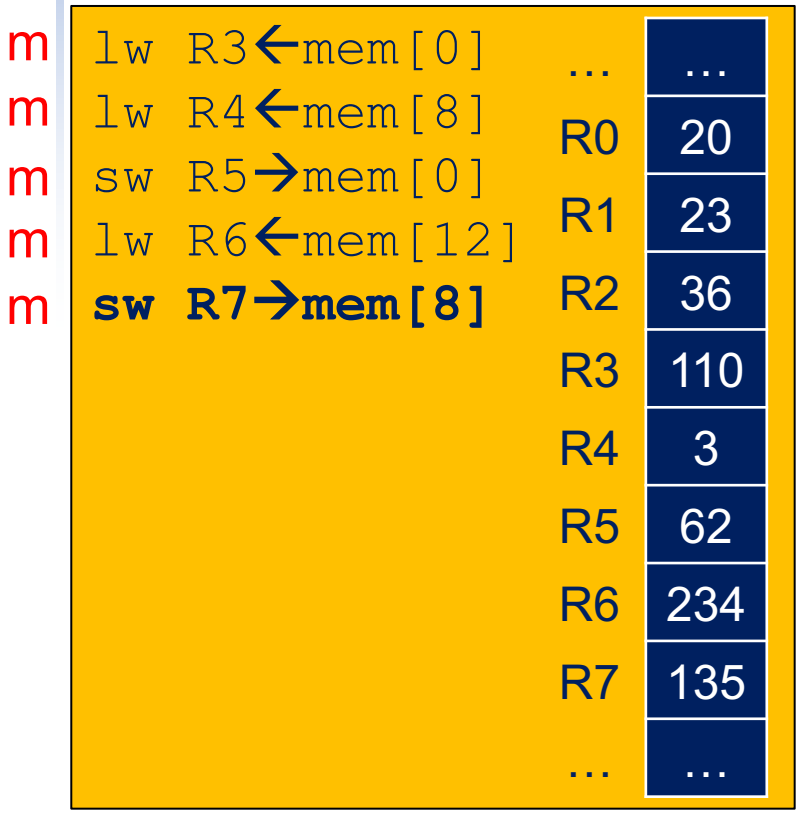
Miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

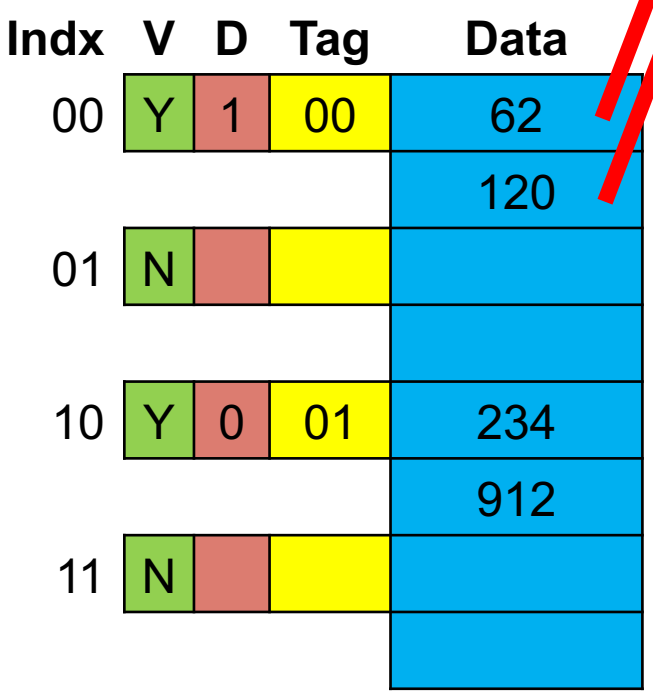
Associativity Example

- Direct mapped (1-way associative)

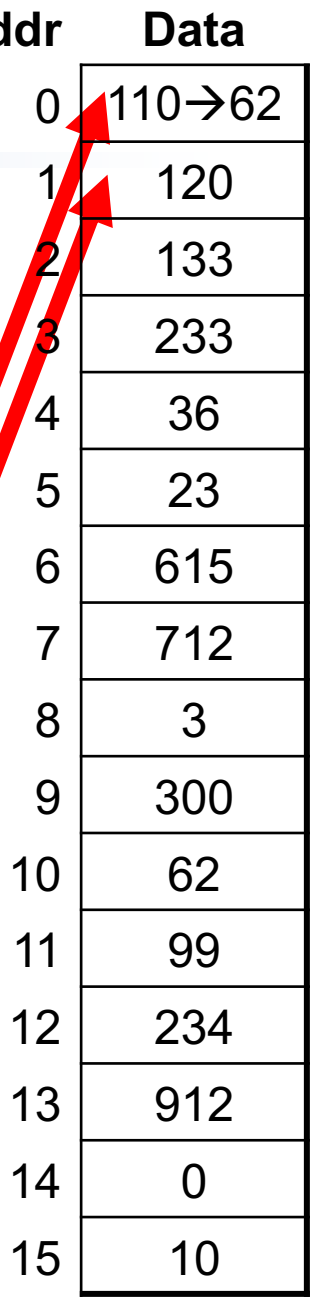
Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	01 00 0	miss	00



CPU



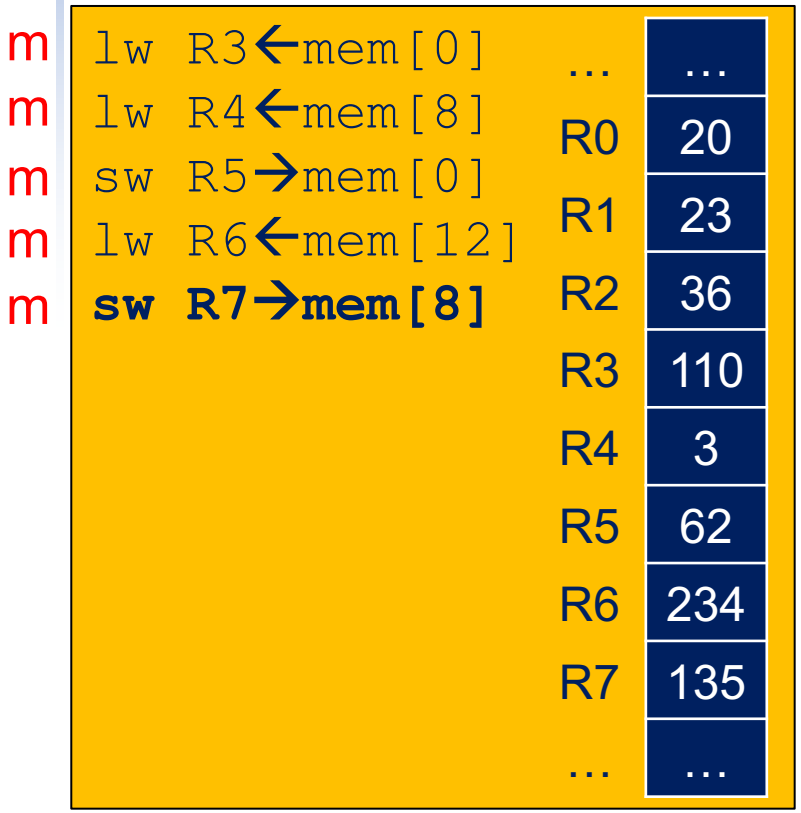
Write back

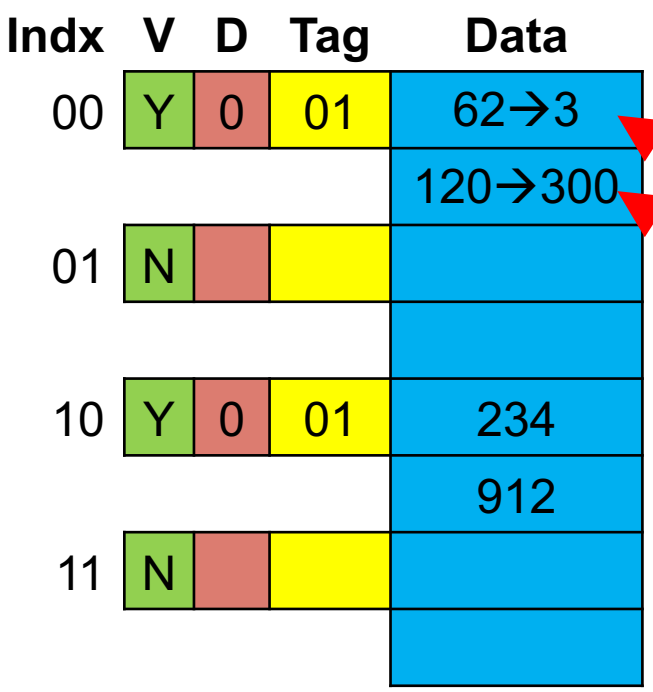


Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	01 00 0	miss	00





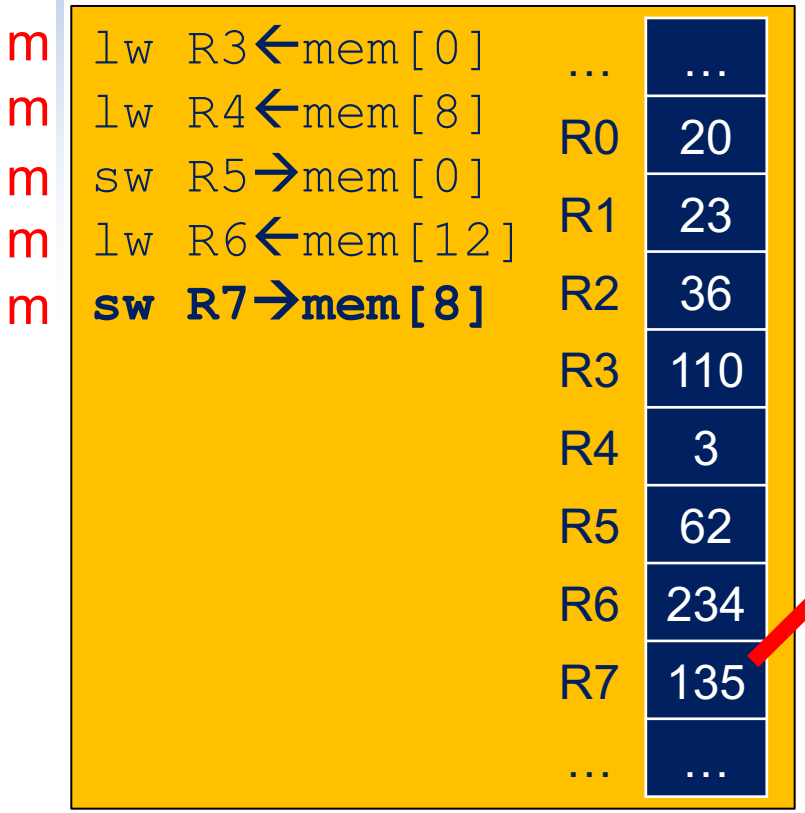
Replace

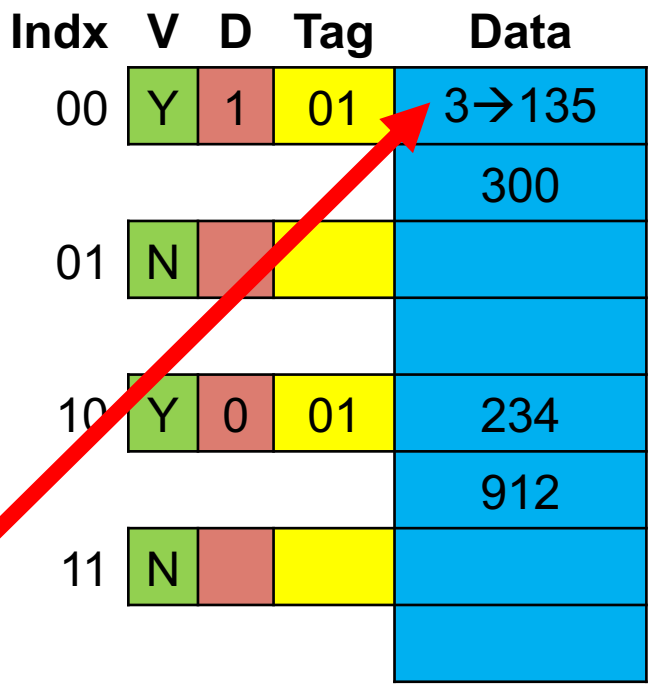
Word Addr	Data
0	62
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- Direct mapped (1-way associative)

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	01 00 0	miss	00





Write, set dirty

Word Addr	Data
0	62
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

2-way associative

Associativity Example

- 2-way associative cache

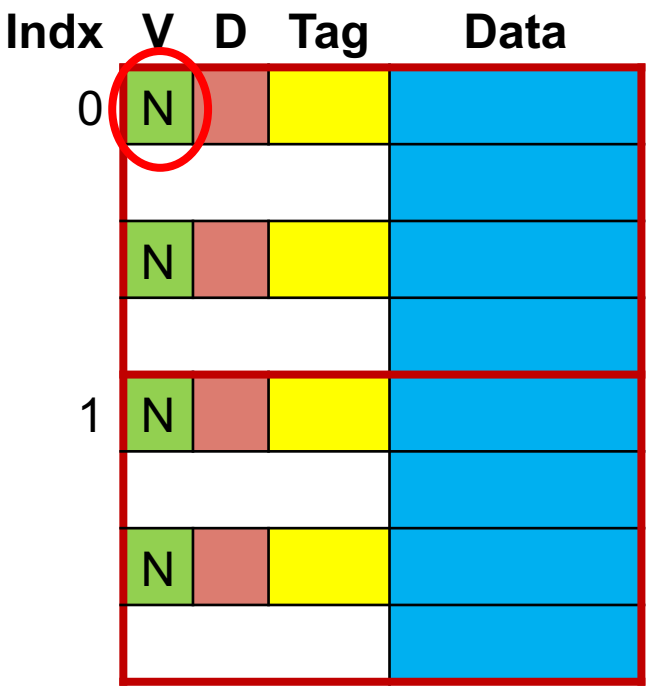
Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	000 0 0	miss	0

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	23
R4	87
R5	62
R6	99
R7	135
…	…

CPU



Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	000 0 0	miss	0

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	23
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
0	Y	0	000	110
				120
	N			
1	N			
	N			

Fetch

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	000 0 0	hit	0

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	110
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
0	Y	0	000	110
				120
	N			
1	N			
	N			

Load again

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	010 0 0	miss	0

m `lw R3←mem[0]`
m **`lw R4←mem[8]`**
`sw R5→mem[0]`
`lw R6←mem[12]`
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
0	Y	0	000	110
				120
	N			
1	N			
	N			

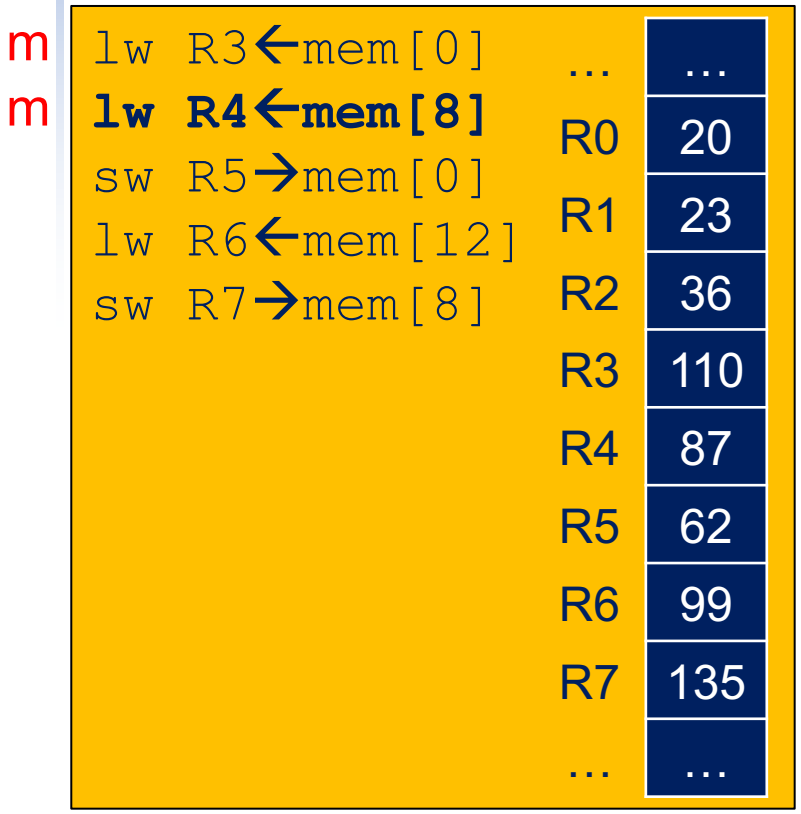
miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

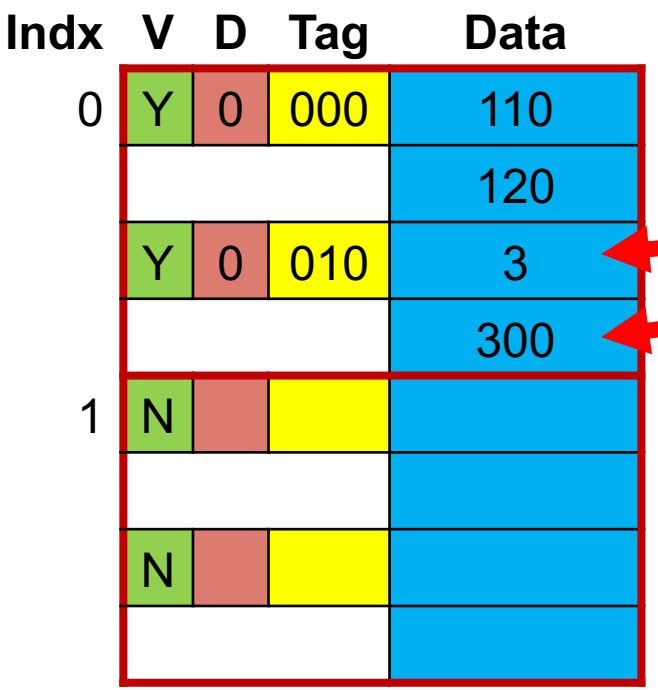
Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	010 0 0	miss	0



CPU



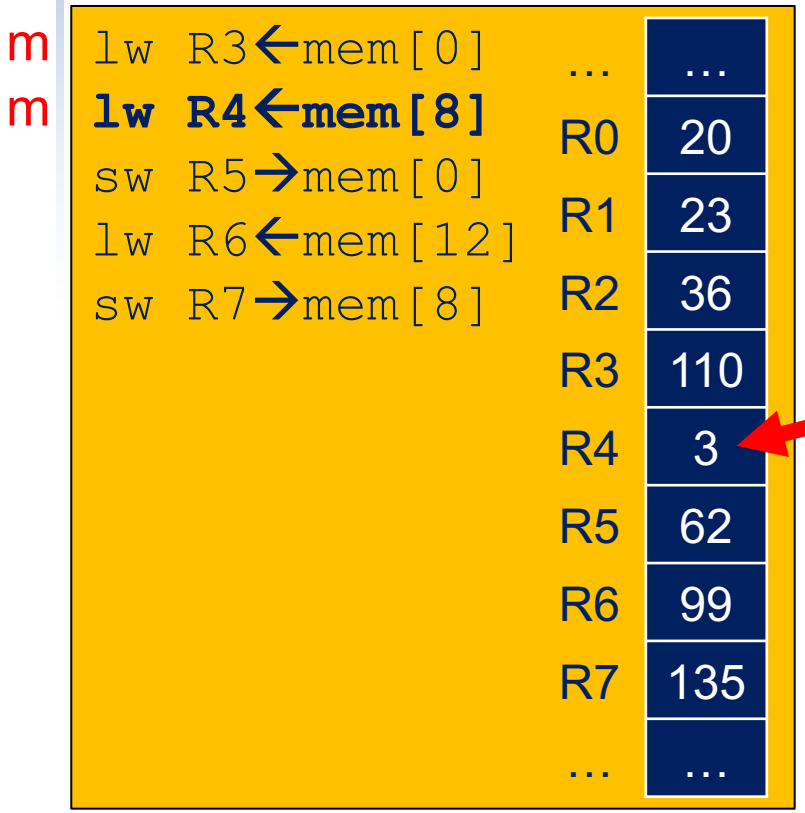
Fetch, not replace

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	010 0 0	hit	0



CPU

Indx	V	D	Tag	Data
0	Y	0	000	110
				120
	Y	0	010	3
				300
1	N			
	N			

Load again

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	000 0 0	hit	0

m lw R3←mem[0]
m lw R4←mem[8]
h **sw R5→mem[0]**
lw R6←mem[12]
sw R7→mem[8]

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
0	Y	1	000	110→62
				120
	Y	0	010	3
				300
1	N			
	N			

Write, set dirty

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	011 0 0	miss	0

m `lw R3←mem[0]`
m `lw R4←mem[8]`
h `sw R5→mem[0]`
m **`lw R6←mem[12]`**
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
0	Y	1	000	62
				120
	Y	0	010	3
				300
1	N			
	N			

Miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Replacement Policy

- Direct mapped: no choice
- Set associative
  - Prefer non-valid entry, if there is one
  - Otherwise, choose among entries in the set
- Choosing policy
  - ***Least-recently used (LRU)***
    - Choose the one unused for the longest time
    - Need a tracking mechanism for usage
      - Simple for 2-way, manageable for 4-way, too hard beyond that
  - Random
    - Gives approximately the same performance as LRU for high associativity

Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	011 0 0	miss	0

```
m  lw R3←mem[0]
m  lw R4←mem[8]
h  sw R5→mem[0]
m  lw R6←mem[12]
   sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
0	Y	1	000	62
				120
	Y	0	011	3→234
	LRU			300→912
1	N			
	N			

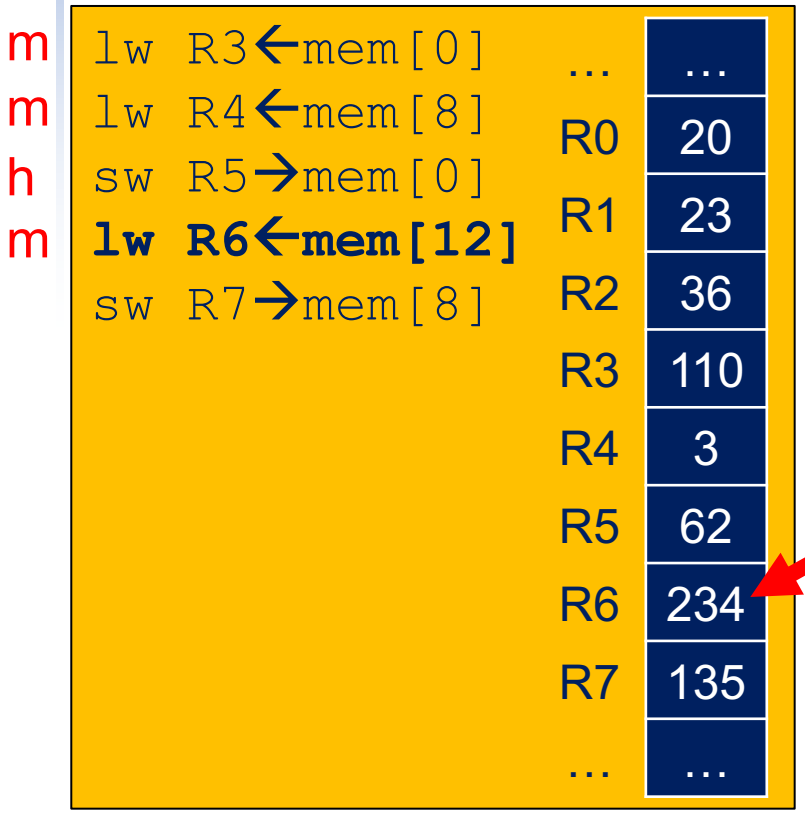
Replace

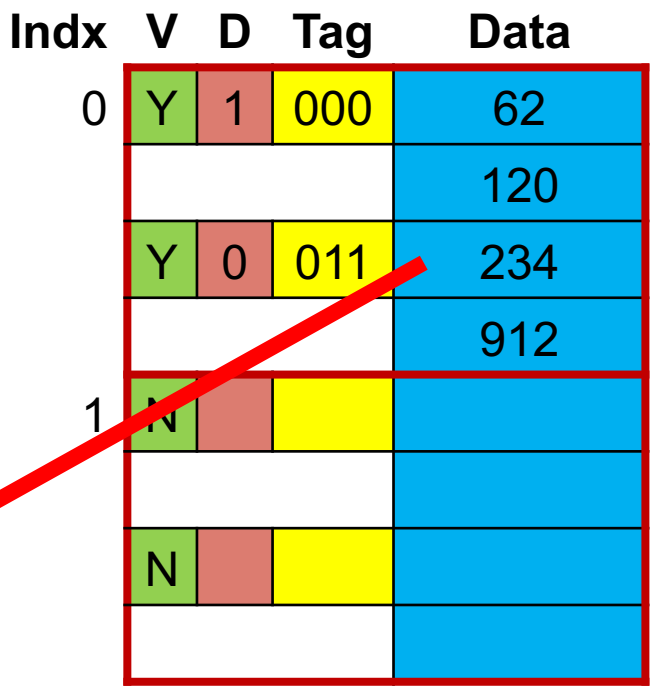
Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

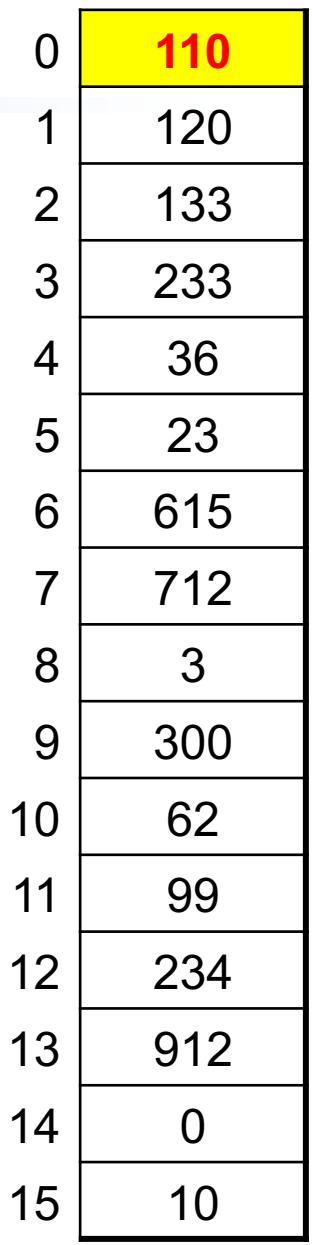
Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	011 0 0	hit	0







Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	010 0 0	miss	0

m `lw R3←mem[0]`
m `lw R4←mem[8]`
h `sw R5→mem[0]`
m `lw R6←mem[12]`
m **`sw R7→mem[8]`**

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	234
R7	135
…	…

CPU

Indx	V	D	Tag	Data
0	Y	1	000	62
				120
	Y	0	011	234
				912
1	N			
	N			

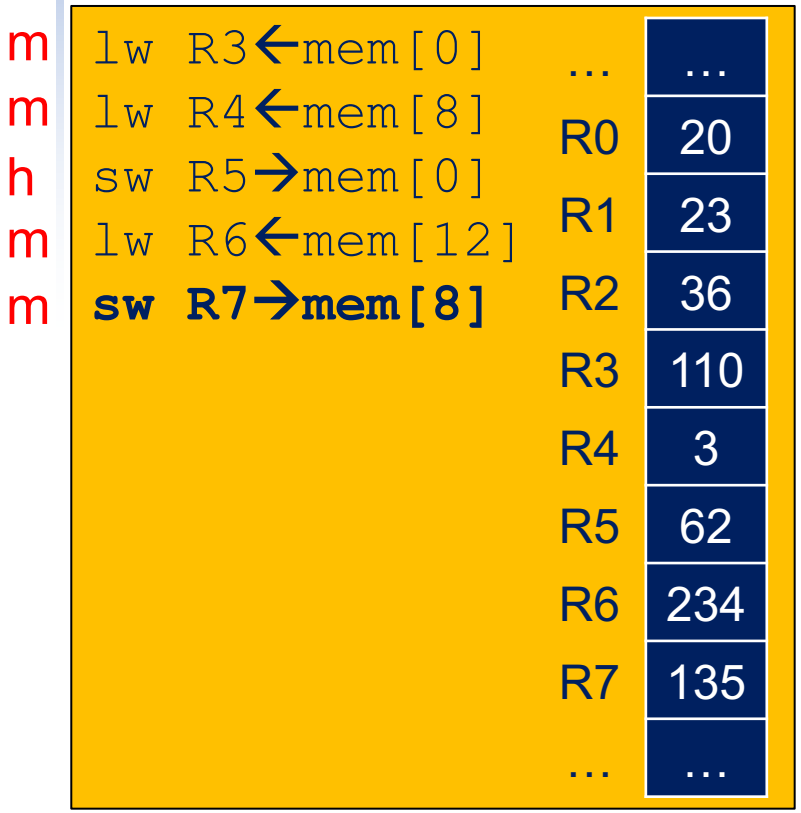
Miss

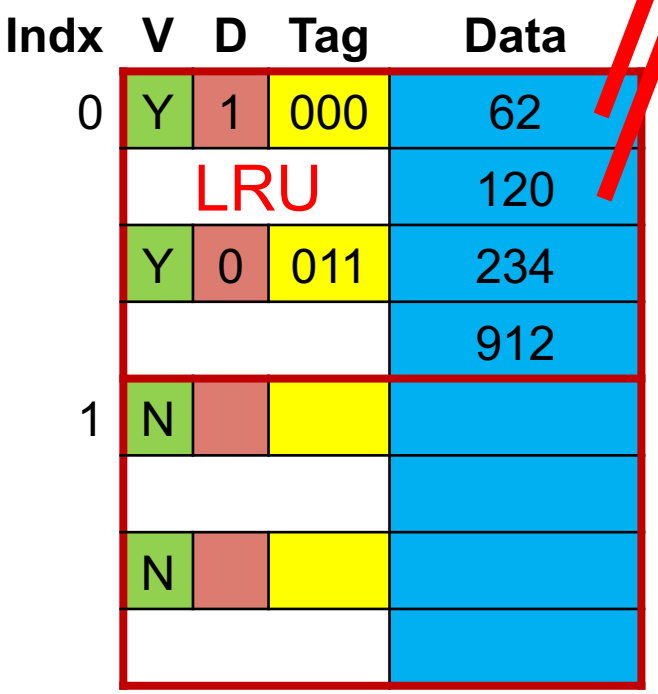
Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

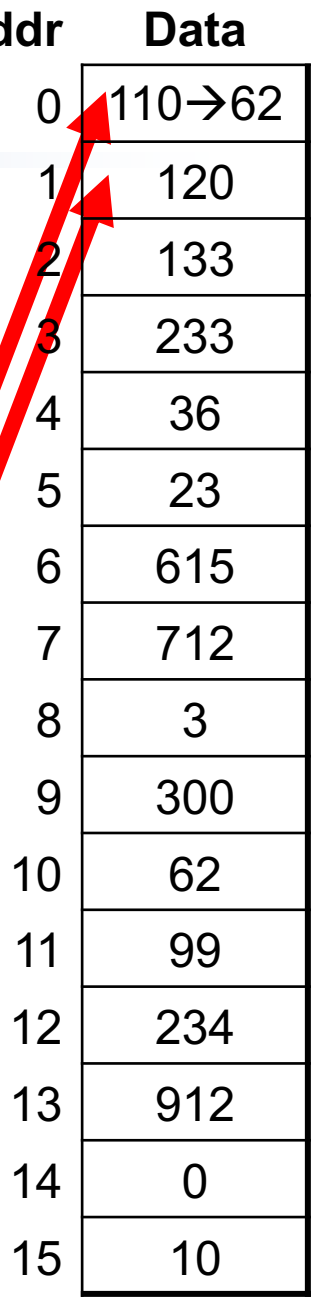
Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	010 0 0	miss	0







Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	010 0 0	miss	0

m `lw R3←mem[0]`
m `lw R4←mem[8]`
h `sw R5→mem[0]`
m `lw R6←mem[12]`
m **`sw R7→mem[8]`**

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	234
R7	135
…	…

CPU

Indx	V	D	Tag	Data
0	Y	0	010	62→3
	LRU			120→300
	Y	0	011	234
				912
1	N			
	N			

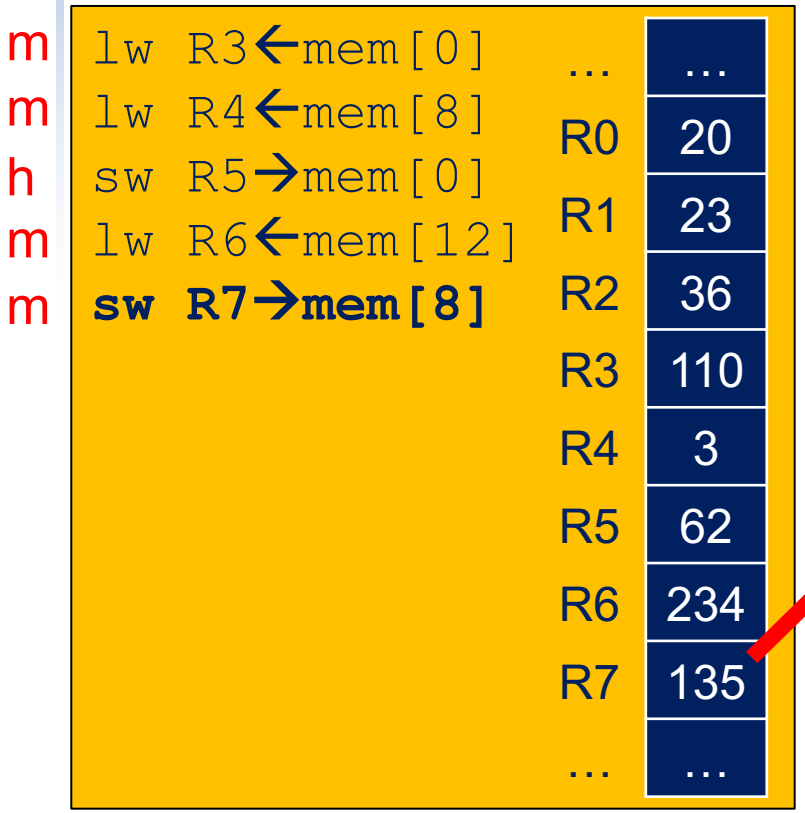
Replace

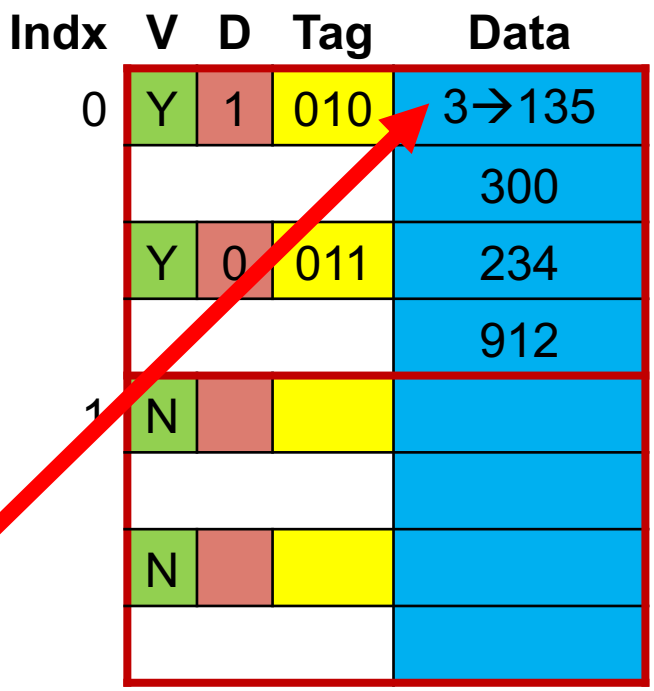
Word Addr	Data
0	62
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 2-way associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	010 0 0	miss	0





Write, set dirty

Word Addr	Data
0	62
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

Fully associative (4-way associative)

Associativity Example

- 4-way (fully) associative cache

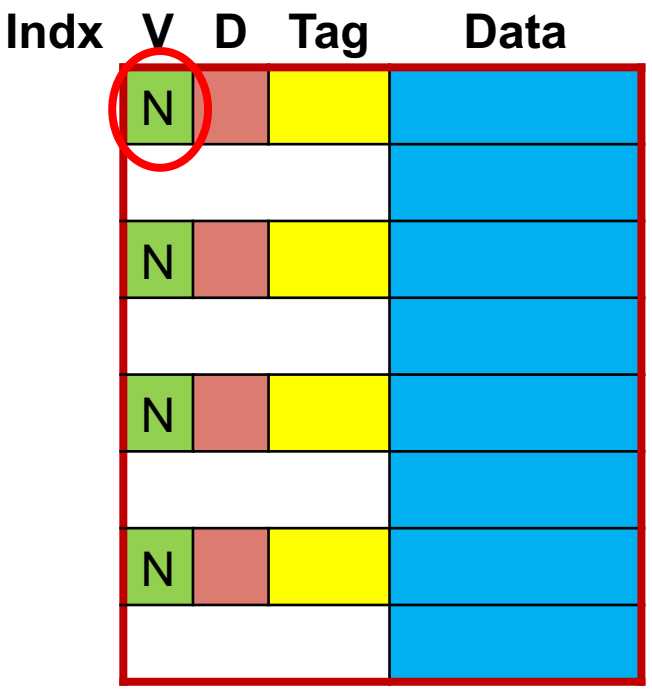
Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	0000 0	miss	-

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	23
R4	87
R5	62
R6	99
R7	135
…	…

CPU



Miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	0000 0	miss	-

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	23
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	0	0000	110
				120
	N			
	N			
	N			

Fetch

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	0000 0	hit	-

m

```
lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
```

…	…
R0	20
R1	23
R2	36
R3	110
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	0	0000	110
				120
	N			
	N			
	N			

Load again

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	0100 0	miss	-

m `lw R3←mem[0]`
m **`lw R4←mem[8]`**
`sw R5→mem[0]`
`lw R6←mem[12]`
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	0	0000	110
				120
	N			
	N			
	N			

miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	0100 0	miss	-

m `lw R3←mem[0]`
m **`lw R4←mem[8]`**
`sw R5→mem[0]`
`lw R6←mem[12]`
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	87
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	0	0000	110
				120
	Y	0	0100	3
				300
	N			
	N			

Fetch, not replace

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	0100 0	hit	-

m `lw R3←mem[0]`
m **`lw R4←mem[8]`**
`sw R5→mem[0]`
`lw R6←mem[12]`
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	0	0000	110
				120
	Y	0	0100	3
				300
	N			
	N			

Load again

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	0000 0	hit	-

m lw R3←mem[0]
m lw R4←mem[8]
h **sw R5→mem[0]**
lw R6←mem[12]
sw R7→mem[8]

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	1	0000	110→62
				120
	Y	0	0100	3
				300
	N			
	N			

Write, set dirty

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	0110 0	miss	-

m `lw R3←mem[0]`
m `lw R4←mem[8]`
h `sw R5→mem[0]`
m **`lw R6←mem[12]`**
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	1	0000	62
				120
	Y	0	0100	3
				300
	N			
	N			

Miss

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	0110 0	miss	-

m `lw R3←mem[0]`
m `lw R4←mem[8]`
h `sw R5→mem[0]`
m **`lw R6←mem[12]`**
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	99
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	1	0000	62
				120
	Y	0	0100	3
				300
	Y	0	0110	234
				912
	N			

Fetch

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	0110 0	hit	-

m `lw R3←mem[0]`
m `lw R4←mem[8]`
h `sw R5→mem[0]`
m **`lw R6←mem[12]`**
`sw R7→mem[8]`

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	234
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	1	0000	62
				120
	Y	0	0100	3
				300
	Y	0	0110	234
				912
	N			

Load again

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

Associativity Example

- 4-way (fully) associative cache

Requested mem addr	Word addr	Hit/miss	Cache set
01000 00	0100 0	hit	-

m	lw R3←mem[0]	
m	lw R4←mem[8]	
h	sw R5→mem[0]	
m	lw R6←mem[12]	
h	**sw R7→mem[8]**	

…	…
R0	20
R1	23
R2	36
R3	110
R4	3
R5	62
R6	234
R7	135
…	…

CPU

Indx	V	D	Tag	Data
	Y	1	0000	62
				120
	Y	1	0100	3→135
				300
	Y	0	0110	234
				912
	N			

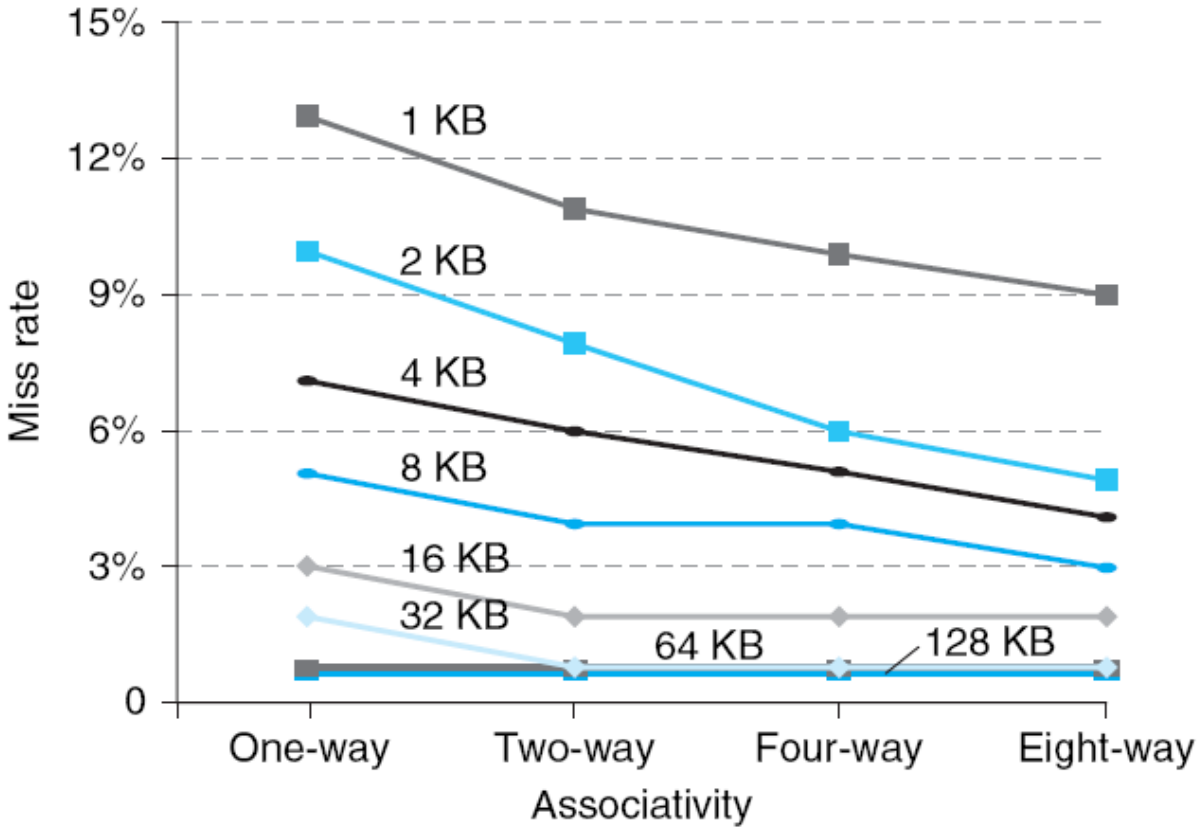
Write, set dirty

Word Addr	Data
0	110
1	120
2	133
3	233
4	36
5	23
6	615
7	712
8	3
9	300
10	62
11	99
12	234
13	912
14	0
15	10

How Much Associativity

- ***Increased associativity decreases miss rate***
  - But with diminishing improvement
- Simulation of a system with 64KB D-cache, 16-word blocks, SPEC2000
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

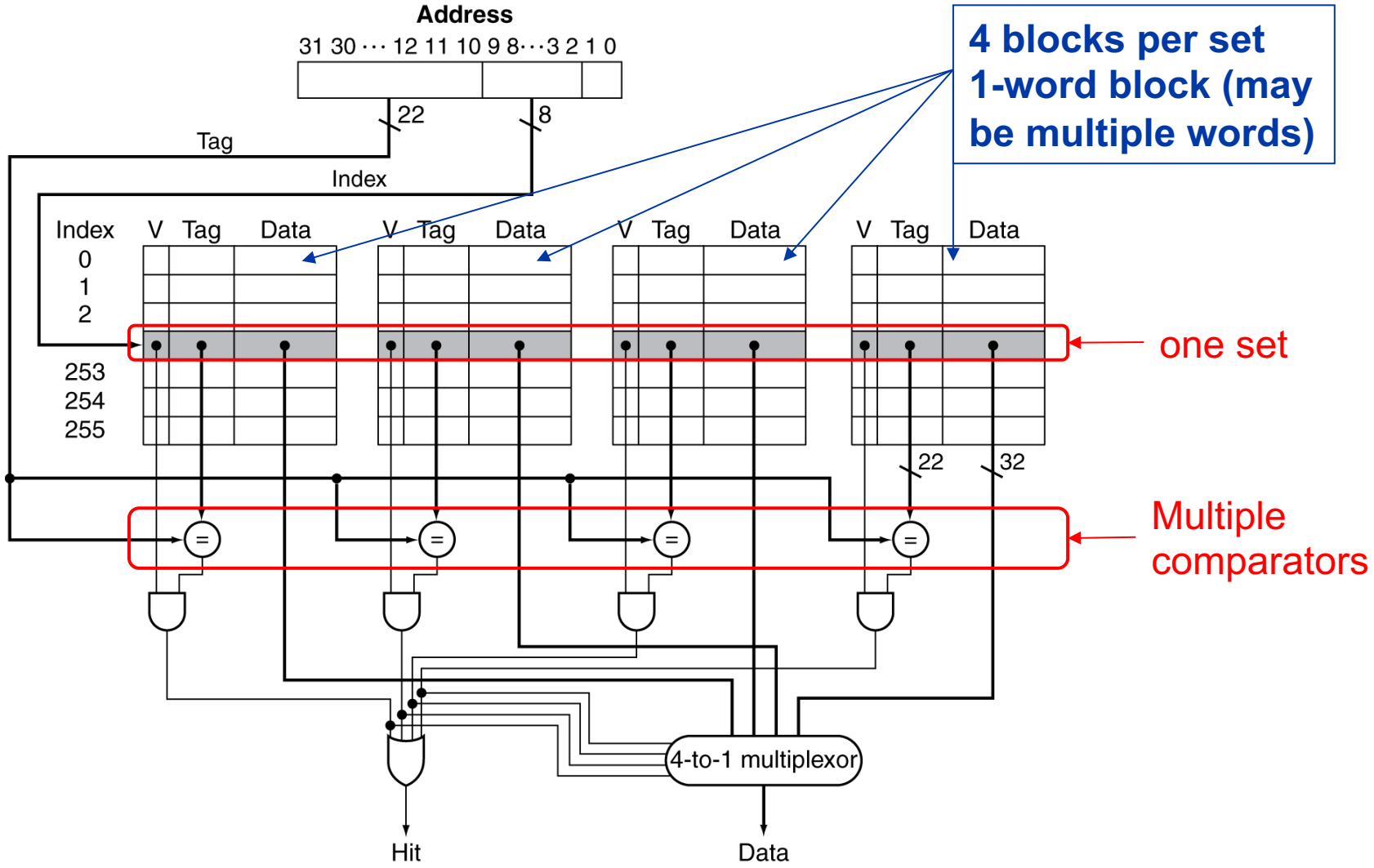
How Much Associativity



Locating a Block

- Memory address decomposition
  - Index – locate a set in cache
  - Tag – upper address bits to locate block
  - Word/Byte offset – to locate a word/byte in a block
- Size of index field
  - Increasing degree of associativity decreases the number of sets, decreases number of bits for index, increases tag field
    - Doubling # of blocks by 2 halves # of set by 2
    - Reduce index bits by 1
    - Increase tag bits by 1
- All blocks in a set must be searched
  - Tag field compared in parallel
  - Extra hardware and **extra access (hit) time**

Set Associative Cache Organization



Exercise

- 2K blocks in cache
- 4-way associative
- 8 words in each block
- 32-bit byte address 0x810023FE requested by CPU, for example

```
lui x10, 0x81002
addi x10, x10, 0x3FE //x10=0x810023FE
lb x5, 0(x10)
```

- Show address and organization of the target cache block, and locate the requested data

Improve Performance – Multilevel Caches

- ***Multilevel cache decreases miss penalty***
- Primary (L-1) cache attached to CPU
  - Small, but fast
- Level-2 (secondary) cache services misses from primary cache
  - Larger, slower, but still faster than main memory
- Main memory services L-2 cache misses
- Some high-end systems include L-3 cache

Multilevel Cache Example

- Given
  - CPU base CPI = 1, clock rate = 4GHz
  - Miss rate (misses/instruction) = 2%
  - Main memory access time = 100ns
    - As miss penalty, ignoring other times
- With one-level cache
  - Miss penalty = 100ns/0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = 9

Example (cont.)

- Now add L-2 cache
  - Access time = 5ns (L-1 miss penalty)
  - Miss rate for L-2 = 25% of L1 misses (have to access main memory)
    - L-1 cache miss have a miss on L-2
- Primary (L-1) cache miss with L-2 hit
  - Miss penalty = 5ns/0.25ns = 20 cycles
- Primary cache miss with L-2 miss main memory hit
  - Extra penalty = 400 cycles
- CPI = base CPI + L-1 miss L-2 hit (cycles per instruction)
  + L-1 miss L-2 miss (cycles per instruction)
  - CPI = 1 + 0.02 × 75% × 20 + 0.02 × 25% × (20+400) = 3.4
- Performance ratio = 9/3.4 = 2.6

Multilevel Cache Considerations

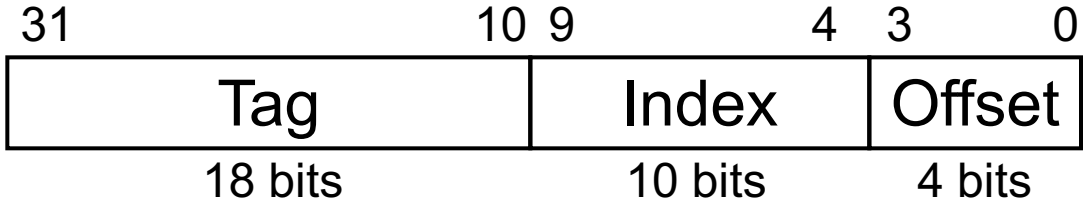
- Primary cache
  - Focus on minimal hit time because miss penalty is smaller
  - And to reduce CPU clock cycle
- Secondary cache
  - Focus on low miss rate to avoid main memory access
  - Hit time has less overall impact

Multilevel Cache Considerations

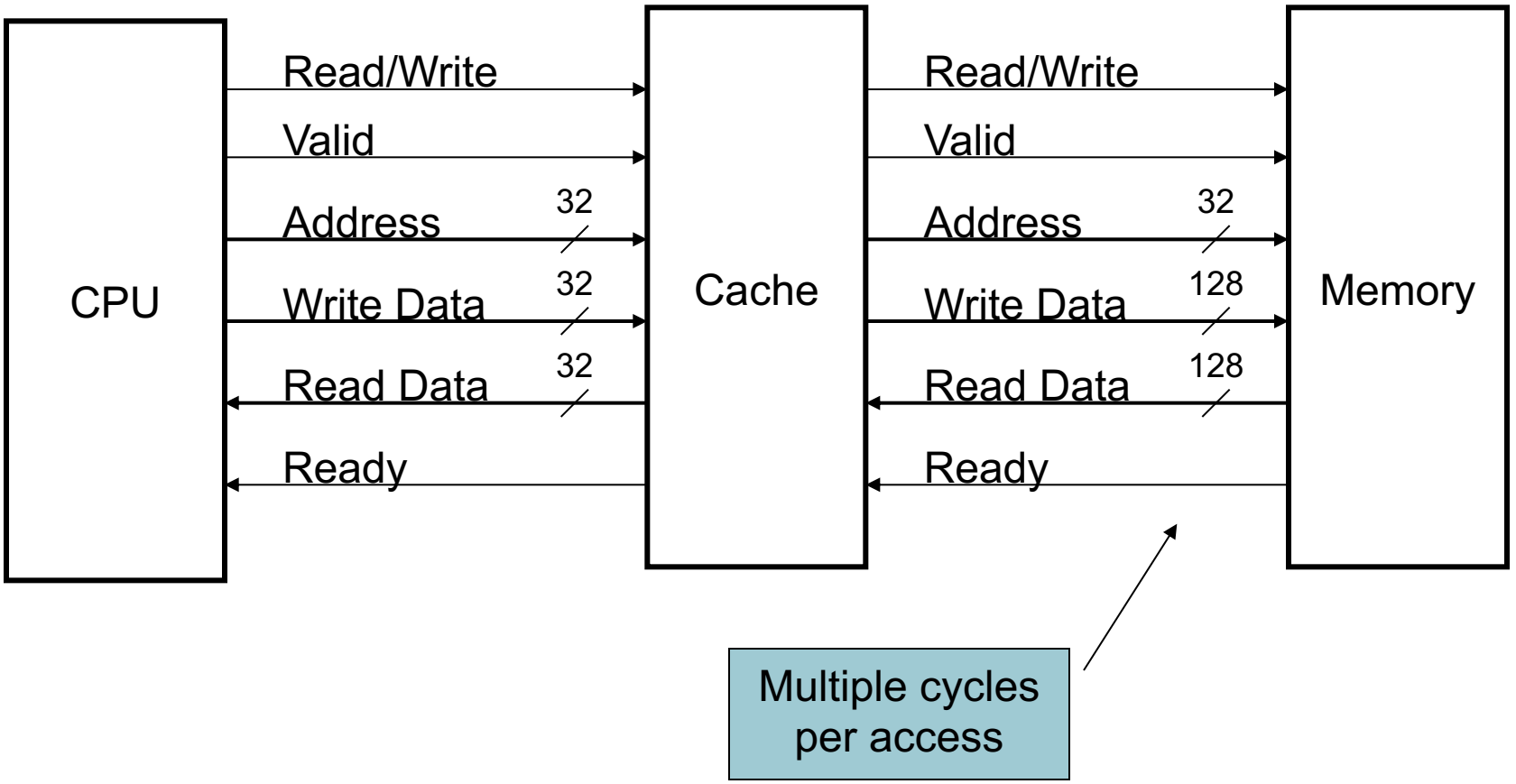
- Comparison with single level cache
  - L-1
    - Smaller cache size
    - Smaller block size, because of
      - Smaller total cache size
      - Reduced search time -> reduced hit time
      - Reduced miss penalty -> less time to fetch
  - L-2
    - Cache and block size much larger
      - because of less critical hit time
    - Higher associativity and block size to reduce miss rate
      - Because miss penalty is more severe

Cache Controller

- Example cache characteristics
  - Direct-mapped, write-back, write allocate
  - Block size: 4 words (16 bytes)
  - Cache size: 16 KB (1024 blocks)
  - 32-bit byte addresses
  - Valid bit and dirty bit per block
  - Blocking cache
    - CPU waits until access is complete

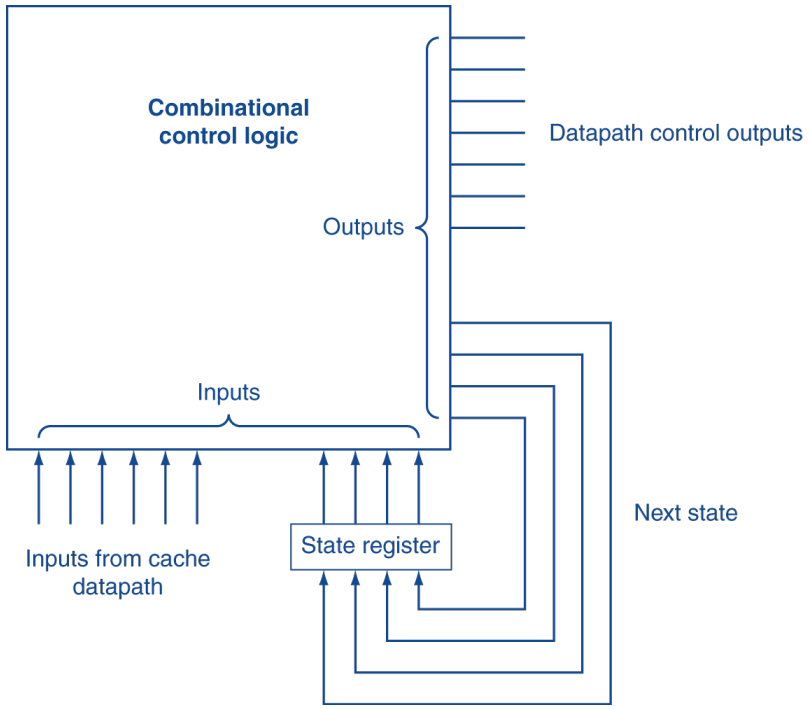


Interface Signals



Finite State Machines

- Use an FSM for sequence control steps
- Set of states, transition on each clock edge
  - State values are binary encoded
  - Current state stored in a register
  - Next state = f_n (current state, current inputs)
- Control output signals = f_o (current state)



Cache Controller FSM