# Topic 11

#### **Memory Hierarchy**

- **Cache (3)** 

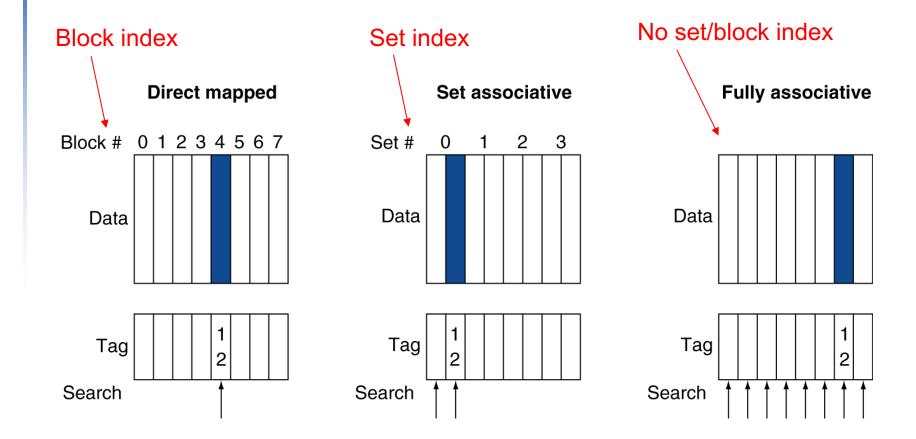
#### **Improve Performance – Associative Caches**

- n-way set associative cache
  - Each set contains n blocks
  - A main memory block can use any of the blocks 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, we need to search n times in the n blocks
    - 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
  - Entire cache is just one set
  - A block can go in any of the cache blocks
  - Must search all entries to find a hit
  - One comparator each block
    - # of comparator = cache size (block number)

# **Associative Cache Example**

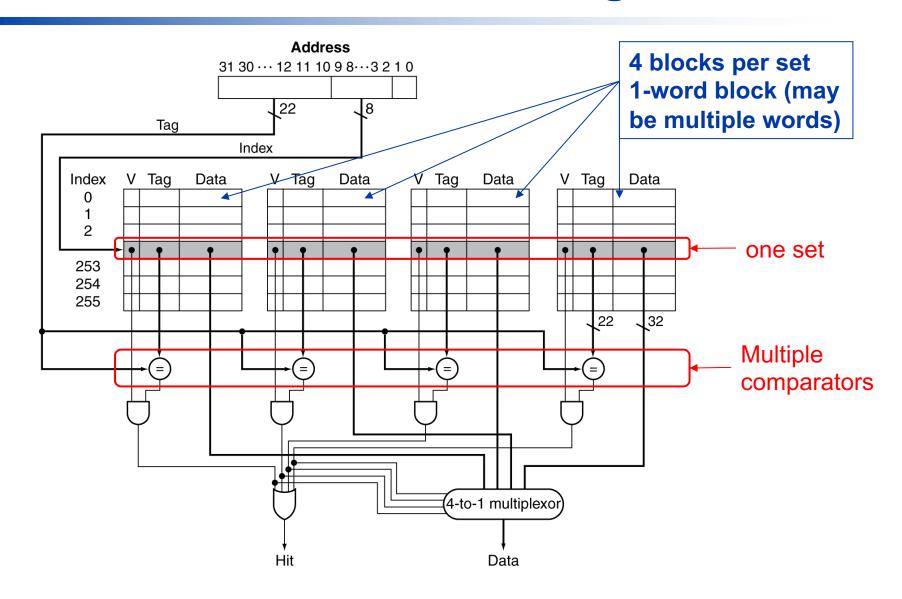


# Locating a Block

Memory address Tag Index Word & Byte offset

- Memory address decomposition
  - Index locate a set in cache
  - Tag upper address bits to locate block
  - Word and 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**



# **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														

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

Direct mapped (1-way associative)

**Word Addr** 

Data

Direct mapped (1-way associative)

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

lw R3←mem[0] lw R4 $\leftarrow$ mem[8] R0 sw R5 $\rightarrow$ mem[0] **R1** lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] R3 R4 **R5** R6 R7 

Indx	V	D	Tag	Data
00	N			
01	N			
10	N			
11	N			
		· · · · · ·		
		M	liss	

Direct mapped (1-way associative)

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

lw	R3←mem[0]		
lw	R4←mem[8]	R0	20
SW	R5→mem[0]	R1	23
lw sw	R6←mem[12] R7→mem[8]	R2	36
<i>2</i>		R3	23
		R4	87
		R5	62
		R6	99
		R7	135

Indx	V	D	Tag	Data	
00	Υ	0	00	110	
				120	
01	N				
10	N				
11	N				
		F	etch	า	

**Word Addr** 

**Data** 

Direct mapped (1-way associative)

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

lw R3←mem[0] lw R4←mem[8] R0 sw R5 $\rightarrow$ mem[0] **R1** lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] R3 R4 **R5** R6 R7 

Indx	V	D	Tag	Data	
00	Y	0	00	110	
				120	
01	N				
10	Ν				
11	N				
	L	.08	ad a	again	

**Word Addr** 

Data

Direct mapped (1-way associative)

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

lw	R3 <b>←</b> mem[0]		
lw		R0	20
SW	R5→mem[0]	R1	23
lw sw	R6←mem[12] R7→mem[8]	R2	36
DW		R3	110
		R4	87
		R5	62
		R6	99
		R7	135
		•••	•••

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

**Word Addr** 

**Data** 

m m

Direct mapped (1-way associative)

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

 $lw R3 \leftarrow mem[0]$ 1w R4 \( \tag{mem[8]} R<sub>0</sub> 20 sw R5 $\rightarrow$ mem[0] **R1** 23 lw R6←mem[12] 36 sw R7 $\rightarrow$ mem[8] R2 **R3** 110 R4 87 **R5** 62 R6 99 135 R7

Indx	V	n	Tag	Data	5	23
	V				6	615
00	Υ	0	01	110→3	7	712
ı				120→300	8	3
01	N					<u> </u>
					9	300
10	N				10	62
10	IN				11	99
11	N				12	234
11	IN				13	912
			_		14	0
		R	epla	ace	15	10

**Word Addr** 

Data

110

120

133

233

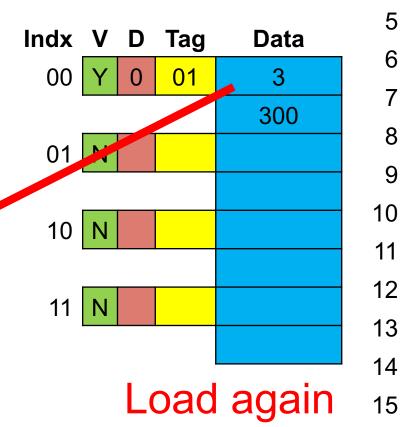
36

4

Direct mapped (1-way associative)

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

lw R3←mem[0] **R4**←mem[8] 20 R0  $R5 \rightarrow mem [0]$ **R1** 23 R6←mem[12] 36 R2  $R7 \rightarrow mem [8]$ 110 R3 3 R4 **R5** 62 R6 99 135 R7



**Word Addr** 

**Data** 

110

120

133

233

36

23

615

712

3

300

62

99

234

912

0

10

4

6

Direct mapped (1-way associative)

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

lw			
lw	R4 ← mem[8]	R0	20
sw lw	<b>R5→mem[0]</b> R6←mem[12]	R1	23
SW		R2	36
		R3	110
		R4	3
		R5	62
		R6	99
		R7	135

m

m

m

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

**Word Addr** 

**Data** 

CPU

Direct mapped (1-way associative)

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] **R1** lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] **R3** R4 **R5** R6 R7 

m

m

Indx	V	D	Tag	Data	
00	Υ	0	00	3→110	
				300→12	
01	Ζ				
10	N				
11	N				
Replace					

Data

**Word Addr** 

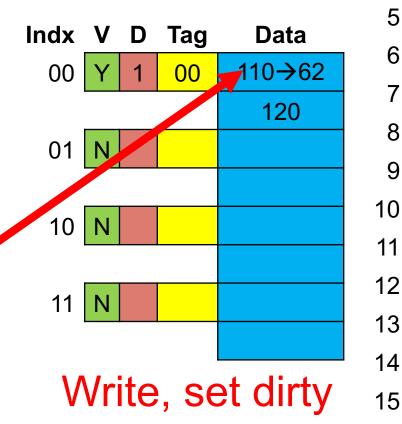
Direct mapped (1-way associative)

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

lw R3←mem[0] R4 ← mem [8] R0  $R5 \rightarrow mem[0]$ **R1** R6←mem[12] R7→mem[8] R2 R3 R4 **R5** R6 R7 

m

m



**Word Addr** 

**Data** 

**CPU** 

Direct mapped (1-way associative)

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

lw	R3←mem[0]		
lw	R4←mem[8]	R0	20
SW <b>lw</b>		R1	23
	R7→mem[8]	R2	36
		R3	110
		R4	3
		R5	62
		R6	99
		R7	135

m

m

m

m

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

**Data** 

Word Addr

Direct mapped (1-way associative)

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] 20 R<sub>0</sub> sw R5 $\rightarrow$ mem[0] **R1** 23 lw R6←mem[12] 36  $R7 \rightarrow mem [8]$ R2 **R3** 110 R4 3 **R5** 62 R6 99 135 R7

m

m

m

Indx	V	D	Tag	Data	5	23
1	V				6	615
00	Y	1	00	62	7	712
ı				120	8	3
01	N					
					9	300
10	V	0	01	224	10	62
10	I	U	UI	234	11	99
11	N			912	12	234
11	IN				13	912
		-	14	0		
		F	15	10		

**Word Addr** 

**Data** 

110

120

133

233

36

4

CPU

Direct mapped (1-way associative)

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

 $lw R3 \leftarrow mem[0]$ 1w R4←mem[8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] **R1** lw R6←mem[12] R2  $R7 \rightarrow mem [8]$ **R3** R4 **R5** R6 R7

m

m

m

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

**Word Addr** 

Data

CPU

Direct mapped (1-way associative)

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

lw R3←mem[0]		
lw R4←mem[8]	R0	20
sw R5 $\rightarrow$ mem[0] lw R6 $\leftarrow$ mem[12]	R1	23
sw R7→mem[8]	R2	36
	R3	110
	R4	3
	R5	62
	R6	234
	R7	135

m

m

m

m

m

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

Data

Word Addr

Direct mapped (1-way associative)

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

lw	R3 <b>←</b> mem[0]		
lw	R4←mem[8]	R0	20
SW		R1	23
lw sw	R6←mem[12]  R7→mem[8]	R2	36
<b>5</b> W	it / / inclin [ 0 ]	R3	110
		R4	3
		R5	62
		R6	234
		R7	135

m

m

m

m

m

Indx	V	D	Tag	Data		
00	Υ	1	00	62		
				120		
01	Ν					
10	Υ	0	01	234		
				912		
11	N					
Write back						

Data

110<del>→</del>62

**Word Addr** 

Direct mapped (1-way associative)

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

lw R3←mem[0]		
lw R4←mem[8]	R0	20
sw R5 $\rightarrow$ mem[0]	R1	23
lw R6←mem[12]	R2	36
sw R7→mem[8]		
	R3	110
	R4	3
	R5	62
	R6	234
	R7	135

m

m

m

m

m

lndx	V	D	Tag	Data	5	23
ĺ	V				6	615
00	Υ	0	01	62→3	7	712
				120→300		
01	N				8	3
O I	IN				9	300
10	V	0	01	224	10	62
10	Y	U	01	234	11	99
44	NI			912	12	234
11	N				13	912
					14	0
	F	Re	pla	ce	15	10

**Word Addr** 

**Data** 

62

120

133

233

36

**CPU** 

Direct mapped (1-way associative)

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

 $lw R3 \leftarrow mem[0]$ 1w R4←mem[8] R<sub>0</sub> 20 sw R5 $\rightarrow$ mem[0] **R1** 23 lw R6←mem[12] 36 R2 sw  $R7 \rightarrow mem[8]$ **R3** 110 R4 3 **R5** 62 234 R6 135 R7

m

m

m

m

Indx	V	D	Tag	Data
00	Y	1	01	3→135
				300
01	Z			
10	Y	0	01	234
				912
11	N			
Write, set dirty				

**Word Addr** 

Data

62

120

133

233

36

23

615

712

300

62

99

234

912

0

10

10

11

12

13

14

15

CPU

25

2-way associative

Word Addr D

Data

110

120

133

615

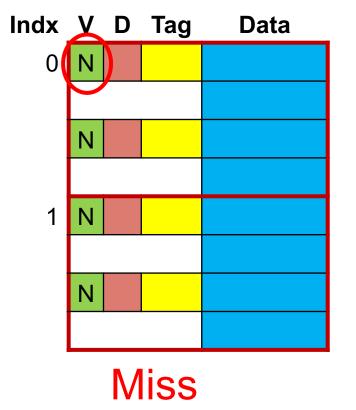
Requested mem addr	Word addr	Hit/miss	Cache set
00000 00	000 0 0	miss	0

3 233 4 36 5 23

6

lw	R3←mem[0]		
lw	R4←mem[8]	R0	20
SW	$R5 \rightarrow mem[0]$	R1	23
lw	R6←mem[12]	ΙΧΙ	
SW	R7→mem[8]	R2	36
		R3	23
		R4	87
		R5	62
		R6	99
		R7	135

m



**CPU** 

2-way associative cache

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

lw R3←mem[0] lw R4 $\leftarrow$ mem[8] R0 sw R5 $\rightarrow$ mem[0] **R1** lw R6←mem[12] sw R7 $\rightarrow$ mem[8] R2 R3 R4 R5 R6 R7 

Indx	V	D	Tag	Data	
0	Υ	0	000	110	
				120	
	N				
1	N				
	N				
'		F	etcl	า	

**Word Addr** 

Data

2-way associative cache

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

lw R3←mem[0] lw R4 $\leftarrow$ mem[8] R0 sw R5 $\rightarrow$ mem[0] **R1** lw R6←mem[12] sw R7 $\rightarrow$ mem[8] R2 R3 R4 **R5** R6 R7 

**CPU** 

Indx	V	D	Tag	Data	
0	Υ	0	000	110	
				120	
	N				
1	N				
	N				
	L	.08	ad a	again	

**Word Addr** 

Data

#### 2-way associative cache

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R0 sw R5 $\rightarrow$ mem[0] R1 lw R6←mem[12] sw R7 $\rightarrow$ mem[8] R2 R3 R4 **R5** R6 R7 

Indx		D	Tag	Data
0	Υ	0	000	110
				120
	N			
1	N			
	Z			
miss				

**Word Addr** 

Data

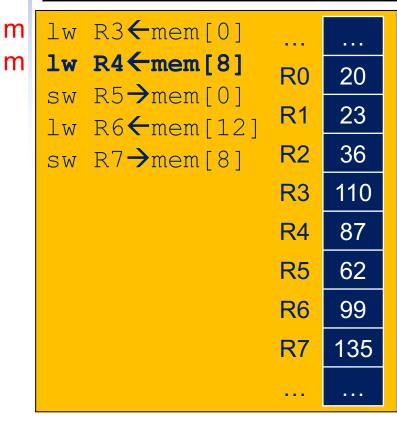
m m

#### **Word Addr Data**

2-way associative cache

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

Tag Indx V Data D Q N N Fetch, not replace

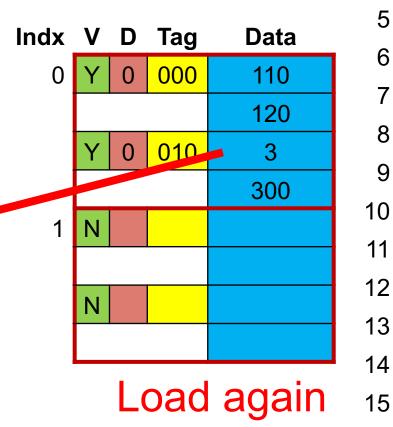


#### 2-way associative cache

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

lw R3←mem[0] **R4**←mem[8] R<sub>0</sub> 20  $R5 \rightarrow mem [0]$ R1 23 R6←mem[12] 36 R2  $R7 \rightarrow mem [8]$ R3 110 3 R4 **R5** 62 R6 99 R7 135

m



**Word Addr** 

**Data** 

110

120

133

233

36

23

615

712

3

300

62

99

234

912

0

10

4

5

6

**CPU** 

#### Word Addr

110

120

133

233

36

23

615

712

3

300

62

99

234

912

0

10

15

0

#### 2-way associative cache

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

Write, set dirty

 $1 \text{w} R3 \leftarrow \text{mem}[0]$ R4 ← mem [8] 20 R0  $R5 \rightarrow mem[0]$ **R1** 23 R6←mem[12] 36 R2  $R7 \rightarrow mem [8]$ R3 110 3 R4 **R5** 62 R6 99 R7 135

m

m

h

**CPU** 

33

Word Addr Data

6

8

9

10

11

**110** 120

2-way associative cache

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

3 233 4 36 5 23

615

712

3

300

62

99

0

133

lw R3←mem[0]
lw R4←mem[8]
sw R5→mem[0]
lw R6←mem[12]
sw R7→mem[8]
R1
R2
R3
R3
R3
R1
R3
R1
R1
R2
R3
R3
R3
R3
R3
R3
R3

m

m

h

m

 Indx
 V
 D
 Tag
 Data

 0
 Y
 1
 000
 62

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

120
Y 0 010 3
300
N 0 010

**Miss** 

12 234 13 912

14

15 10

CPU

34

# Replacement Policy

- Direct mapped: no other choices
- Set associative
  - Prefer non-valid entry, if there is one
  - Otherwise, choose to replace a block 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

Word Addr Data

110 120

133

233

36

2-way associative cache

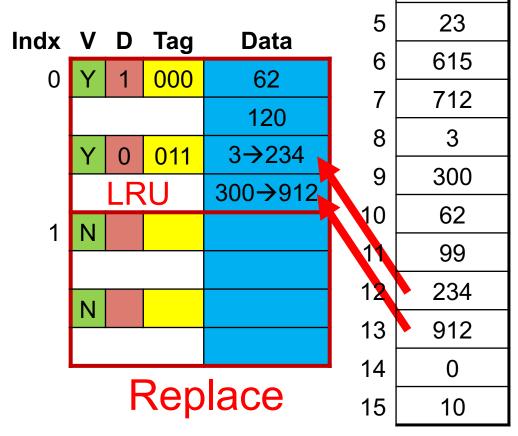
Requested mem addr	Word addr	Hit/miss	Cache set
01100 00	011 0 0	miss	0

 $lw R3 \leftarrow mem[0]$ R4←mem[8] 20 R0  $R5 \rightarrow mem[0]$ **R1** 23  $R6\leftarrow mem[12]$ 36 R2  $R7 \rightarrow mem [8]$ R3 110 R4 3 **R5** 62 R6 99 R7 135

m

h

m



**CPU** 

2-way associative cache

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] R1 lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] R3 R4 R5 R6 R7

m

h

m

Indx	V	D	Tag	Data
0	Υ	1	000	62
				120
	Υ	0	011	<b>234</b>
				912
1	17			
	Z			
Load again				

Data

**Word Addr** 

2-way associative cache

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

lw	R3 <b>←</b> mem[0]		
lw	R4←mem[8]	R0	20
SW	2 3	R1	23
lw sw	R6←mem[12]  R7→mem[8]	R2	36
<b>5</b> W	it / / mem [0]	R3	110
		R4	3
		R5	62
		R6	234
		R7	135
		Κ/	133

m

m

h

m

m

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

**Data** 

**Word Addr** 

Word Addr Data

110<del>→</del>62 

2-way associative cache

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

miss 0

lw R3←mem		
lw R4←mem	RU	20
sw R5→mem	<b>P</b> 1	23
lw R6←mem		36
sw R7→mem	[0]	
	R3	110
	R4	3
	R5	62
	R6	234
	R7	135

m

m

h

m

m

inax	V	ט	ıag	Data		/
0	Υ	1	000	62		,
		LR	C	120		
	Υ	0	011	234		
				912		
1	Ν					
	N					
Write back						

Write back

**CPU** 

#### 2-way associative cache

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

lw R3←mem[0]		
lw R4←mem[8]	R0	20
sw R5 $\rightarrow$ mem[0] lw R6 $\leftarrow$ mem[12]	R1	23
sw R7→mem[8]	R2	36
	R3	110
	R4	3
	R5	62
	R6	234
	R7	135

m

m

m

m

Indx	V	D	Tag	Data	5	23
	V			62 <del>→</del> 3	6	615
0	Y	0	010		7	712
		LR	U	120→300	8	3
	Υ	0	011	234	9	_
				912		300
1	Ν				10	62
·					11	99
	N				12	234
	IN				13	912
			_		14	0
Replace			15	10		

**Word Addr** 

**Data** 

62

120

133

233

36

**CPU** 

2-way associative cache

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] **R1** lw R6←mem[12] R2 sw  $R7 \rightarrow mem[8]$ R3 R4 **R5** R6 R7

m

m

m

Indx	V	D	Tag	Data	
0	Υ	1	010	<b>→</b> 3→135	
				300	
	Υ	0	011	234	
				912	
1	N				
	Z				
Write, set dirty					

Data

**Word Addr** 

Fully associative (4-way associative)

Word Addr Data

110 120

133

4-way (fully) associative cache

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

36

1w R3←mem[0] R4←mem[8] R<sub>0</sub> 20  $R5 \rightarrow mem[0]$ R1 23 lw R6←mem[12] 36 R2  $R7 \rightarrow mem [8]$ R3 23 R4 87 **R5** 62

m

Tag Indx V D Data N N N Miss

**CPU** 

R6

R7

99

135

4-way (fully) associative cache

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

1w R3←mem[0] lw R4 $\leftarrow$ mem[8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] R1 lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] R3 R4 **R5** R6 R7 

ndx	V	D	Tag	Data	
	Υ	0	0000	110	
				120	
	N				
	N				
	N				
'		F	etch		•

**Word Addr** 

Data

m

4-way (fully) associative cache

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

1w R3←mem[0] lw R4←mem[8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] R1 lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] R3 R4 **R5** R6 R7 

Indx	V	D	Tag	Data	
	Υ	0	0000	110	
				120	
	N				
	N				
	Z				
•	L	.08	ad a	gain	

**Word Addr** 

Data

m

4-way (fully) associative cache

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

 $lw R3 \leftarrow mem[0]$ 1w R4 - mem [8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] R1 lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] R3 R4 **R5** R6 R7 

Indx	V	D	Tag	Data
	Υ	0	0000	110
				120
	N			
	N			
	N			
,		m	iss	

**Word Addr** 

Data

m m



#### 4-way (fully) associative cache

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R<sub>0</sub> 20 sw R5 $\rightarrow$ mem[0] R1 23 lw R6←mem[12] 36 sw R7 $\rightarrow$ mem[8] R2 R3 110 R4 87 **R5** 62 R6 99 R7 135

m

Indx	V	D	Tag	Data	5	23
IIIUX	v				6	615
	Υ	0	0000	110	7	712
				120		
	Υ	0	0100	3	8	<b>3</b>
	•		0100		9	<b>3</b> 00
				300	10	62
	N					
					11	99
	N				12	234
	IN				13	912
	<u> </u>				14	0
Fet	ch	۱, ۱	not r	eplace	<b>e</b> 15	10

**Word Addr** 

**Data** 

110

120

133

233

36

CPU

47

4-way (fully) associative cache

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] R1 lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] R3 R4 **R5** R6 R7 

m

Indx	V	D	Tag	Data
	Υ	0	0000	110
				120
	Υ	0	0100	3
				300
	N			
	N			
		1	oad	again

Data

**Word Addr** 

CPU

4-way (fully) associative cache

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R0 sw R5 $\rightarrow$ mem[0] R1 lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] R3 R4 **R5** R6 R7 

m

m

h

Indx	V	D	Tag	Data
	Υ	1	0000	<b>√</b> 110 <del>&gt;</del> 62
				120
	Y	0	0100	3
				300
	N			
	N			
Write, set dirty				

**Word Addr** 

Data

CPU

R7

Word Addr

**Data** 

<ul><li>4-way (fully)</li></ul>	associative cache
---------------------------------	-------------------

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R0 sw R5 $\rightarrow$ mem[0] R1 lw R6←mem[12] sw R7 $\rightarrow$ mem[8] R2 R3 R4 **R5** R6 

m

m

h

m

Indx	V	D	Tag	Data	
	Υ	1	0000	62	
				120	
	Υ	0	0100	3	
				300	
	N				
	N				
'	Miss				

CPU

**CPU** 

Data

110

120

133

233

36

23

615

712

3

300

62

99

234

912

0

10

14

15

**Fetch** 

4-way (fully) associative cache

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R<sub>0</sub> sw R5 $\rightarrow$ mem[0] **R1** lw R6←mem[12] R2 sw R7 $\rightarrow$ mem[8] **R3** R4 **R5** R6 R7

m

h

m

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

Data

**Word Addr** 

**CPU** 

4-way (fully) associative cache

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

 $lw R3 \leftarrow mem[0]$ lw R4←mem[8] R<sub>0</sub> 20 sw R5 $\rightarrow$ mem[0] **R1** 23 lw R6←mem[12] 36 R2  $R7 \rightarrow mem[8]$ R3 110 R4 3 **R5** 62 234 R6 135 R7

m

h

m

h

Indx	V	D	Tag	Data
	Υ	1	0000	62
				120
	Υ	~	0100	3→135
				300
	Y	7	0110	234
				912
	Ν			
Write, set dirty				

Data

110

**Word Addr** 

**CPU** 

# **How Much Associativity**

- Increased associativity decreases miss rate
  - But with diminishing improvement
- Simulation of a system with 64KB
   D-cache, 16-word blocks, SPEC2000

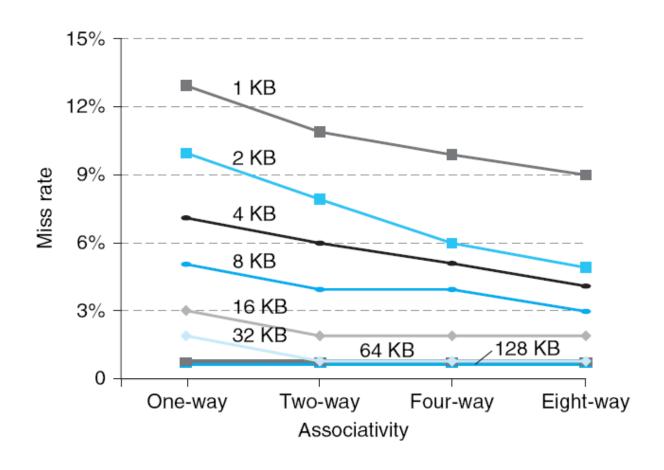
1-way: 10.3%

2-way: 8.6%

4-way: 8.3%

8-way: 8.1%

# **How Much Associativity**



### **Exercise**

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

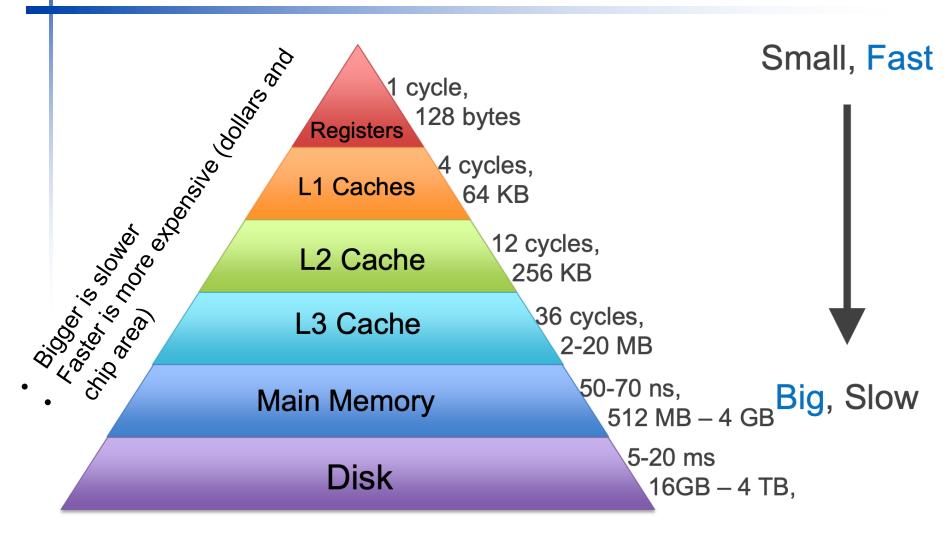
```
lui x10, 0x81002
addi x10, x10, 0x3FE //x10=0x810023FE
lb x5, 0(x10)
```

 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

## **Multi-level Cache**



Intel Haswell Processor, 2013

Image: cs.cornell.edu/courses/cs3410/

# 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 =  $1 + 0.02 \times 400 = 9$

# **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