

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.
Seoul National University

Fall 2020

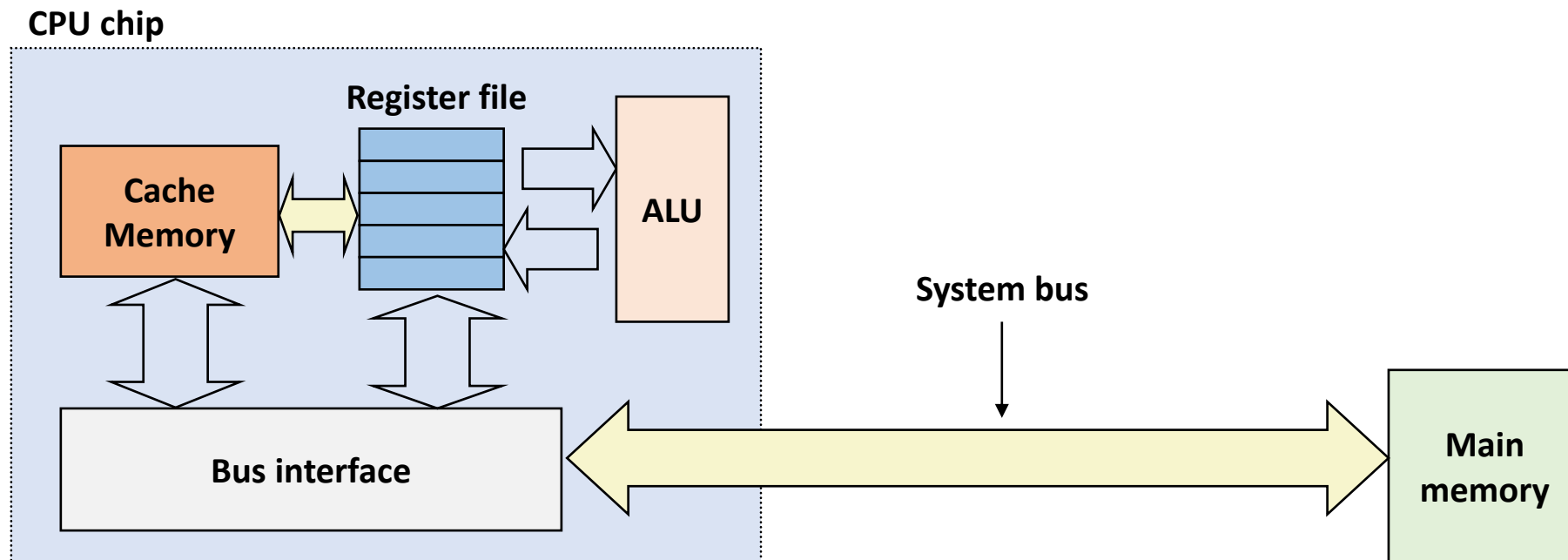
Cache

Chap. 5.3



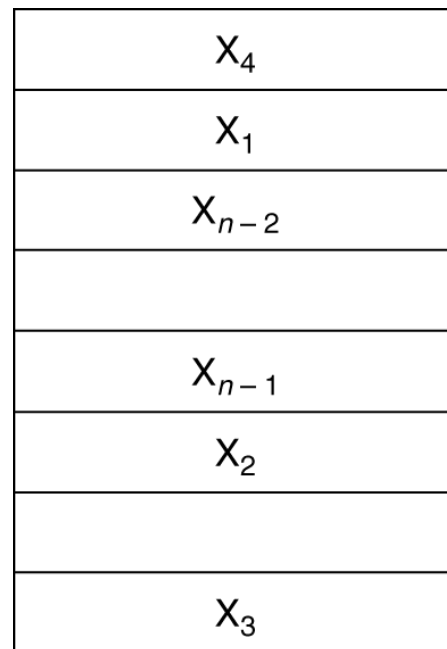
Cache Memories

- Small, fast SRAM-based memories managed automatically in hardware
 - Hold frequently accessed blocks of main memory
- CPU looks first for data in cache

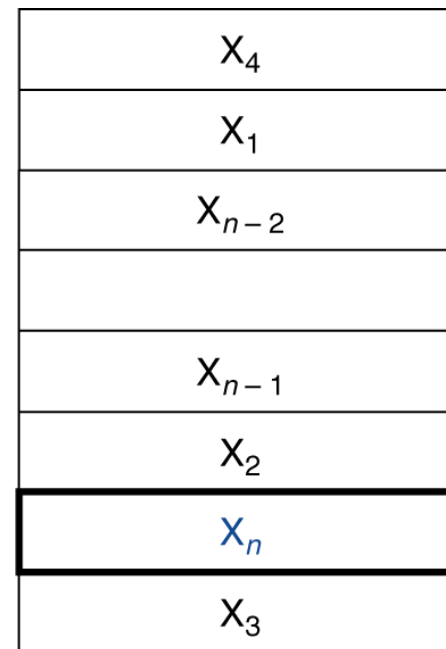


Issues

- Given memory accesses X_1, \dots, X_{n-1}, X_n
 - How do we know if the data is present in the cache?
 - Where do we look?



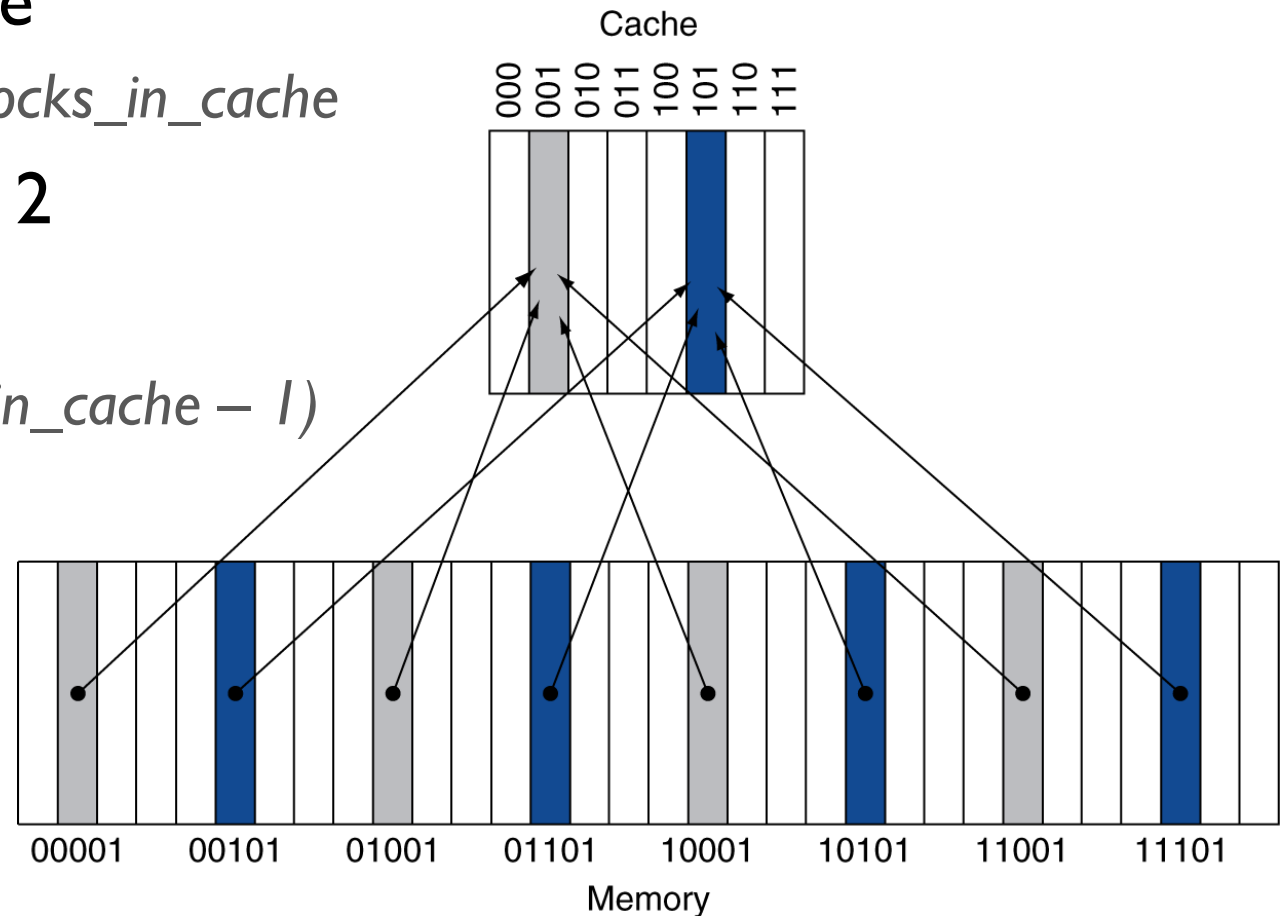
a. Before the reference to X_n



b. After the reference to X_n

Direct Mapped Cache

- Location determined by address
- Direct mapped: only one choice
 - $index = \text{Block_address} \bmod \# \text{Blocks_in_cache}$
- $\# \text{Blocks_in_cache}$ is a power of 2
- Use low-order address bits
 - $index = \text{Block_address} \& (\# \text{Blocks_in_cache} - 1)$
- Why low-order bits?



Tags and Valid Bits

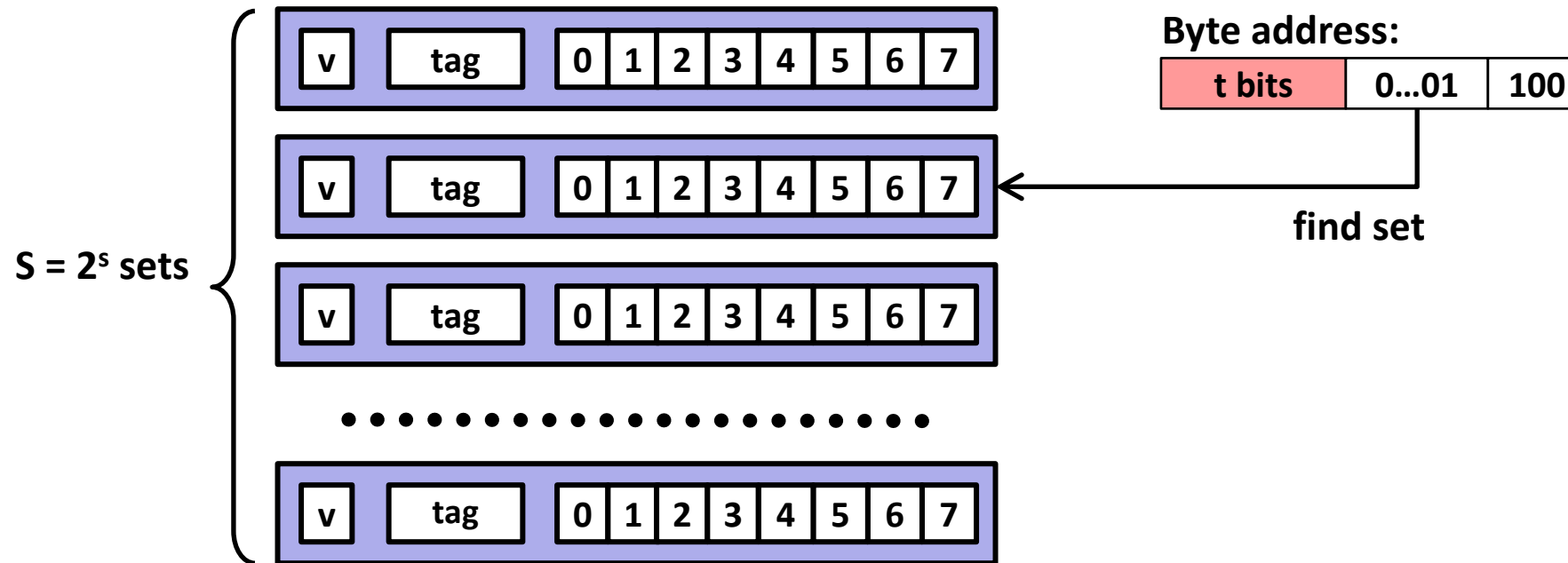
- How do we know which particular block is stored in a cache location?
 - Store block address as well as the data
 - Actually, only need the high-order bits
 - Called the tag



- What if there is no data in a location?
 - Valid bit: 1 = present, 0 = not present
 - Initially 0

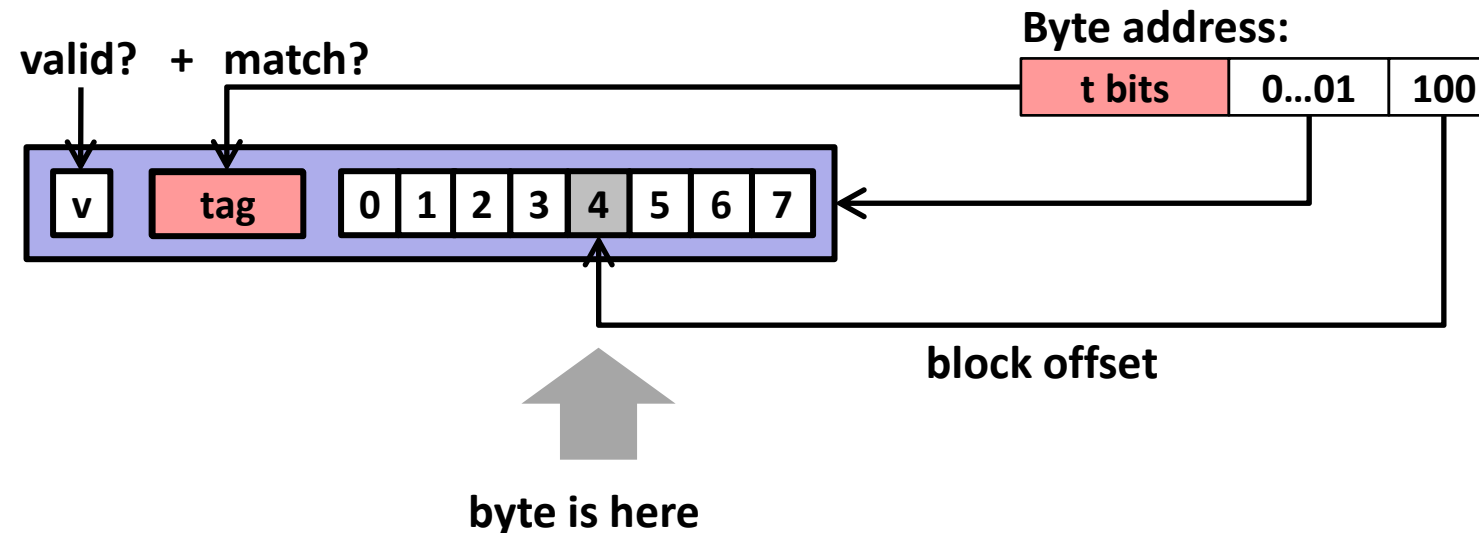
Direct Mapped Cache: Lookup

- Assume: cache block size 8 bytes



Direct Mapped Cache: Lookup

- Assume: cache block size 8 bytes



No match: old line is evicted and replaced

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Initial state

Word address	Binary address	Cache block	Hit / Miss

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 22 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 26 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 22 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss
22	10 110	110	Hit

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 26 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss
22	10 110	110	Hit
26	11 010	010	Hit

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 16 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss
22	10 110	110	Hit
26	11 010	010	Hit
16	10 000	000	Miss

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 3 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss
22	10 110	110	Hit
26	11 010	010	Hit
16	10 000	000	Miss
3	00 011	011	Miss

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 16 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss
22	10 110	110	Hit
26	11 010	010	Hit
16	10 000	000	Miss
3	00 011	011	Miss
16	10 000	000	Hit

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 18 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss
22	10 110	110	Hit
26	11 010	010	Hit
16	10 000	000	Miss
3	00 011	011	Miss
16	10 000	000	Hit
18	10 010	010	Miss

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	10	Mem[10010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 16 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss
22	10 110	110	Hit
26	11 010	010	Hit
16	10 000	000	Miss
3	00 011	011	Miss
16	10 000	000	Hit
18	10 010	010	Miss
16	10 000	000	Hit

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	10	Mem[10010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

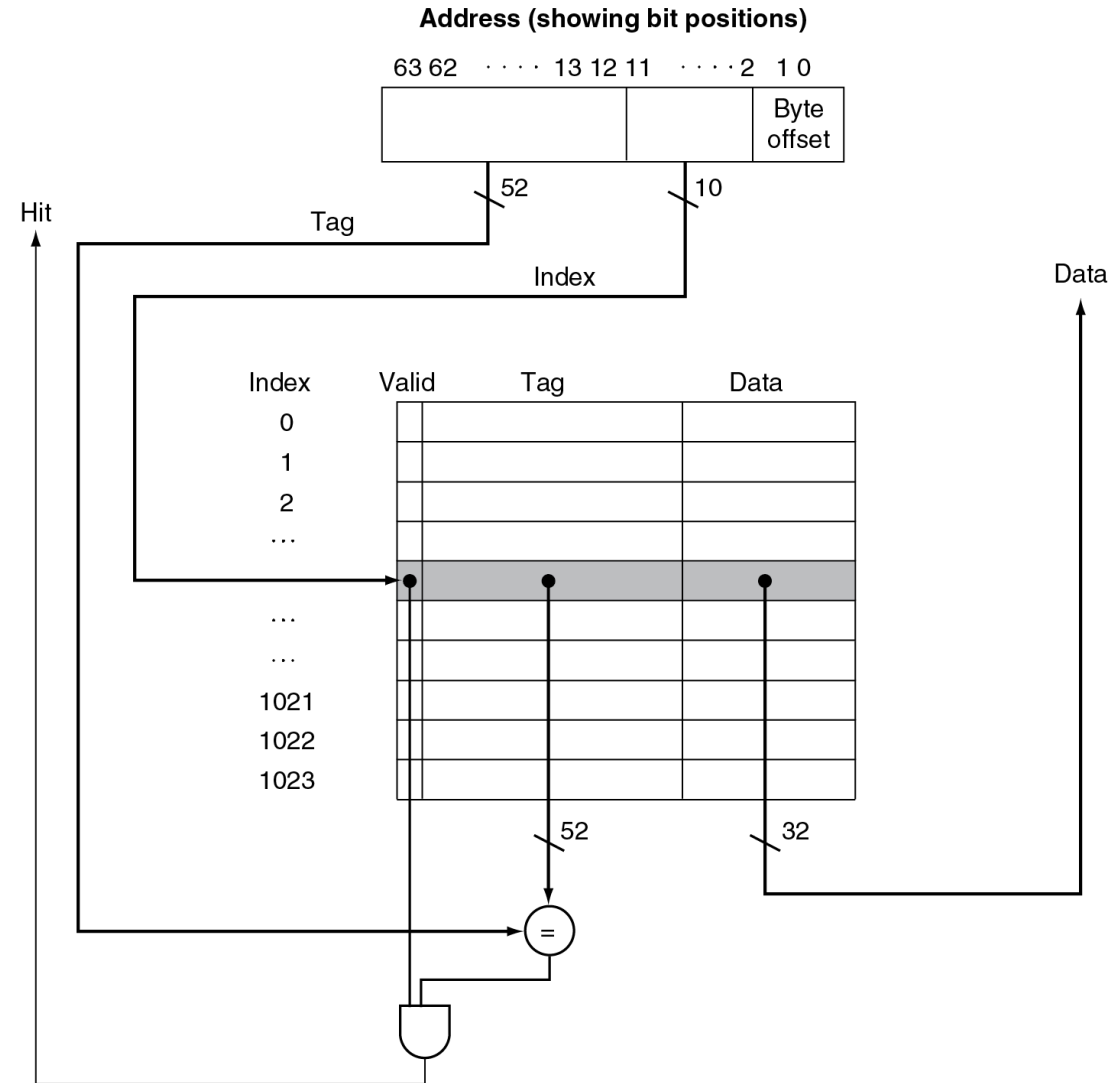
Direct Mapped Cache Example

- 8 blocks, 1 word / block, direct mapped
- Address 26 accessed

Word address	Binary address	Cache block	Hit / Miss
22	10 110	110	Miss
26	11 010	010	Miss
22	10 110	110	Hit
26	11 010	010	Hit
16	10 000	000	Miss
3	00 011	011	Miss
16	10 000	000	Hit
18	10 010	010	Miss
16	10 000	000	Hit
26	11 010	010	Miss

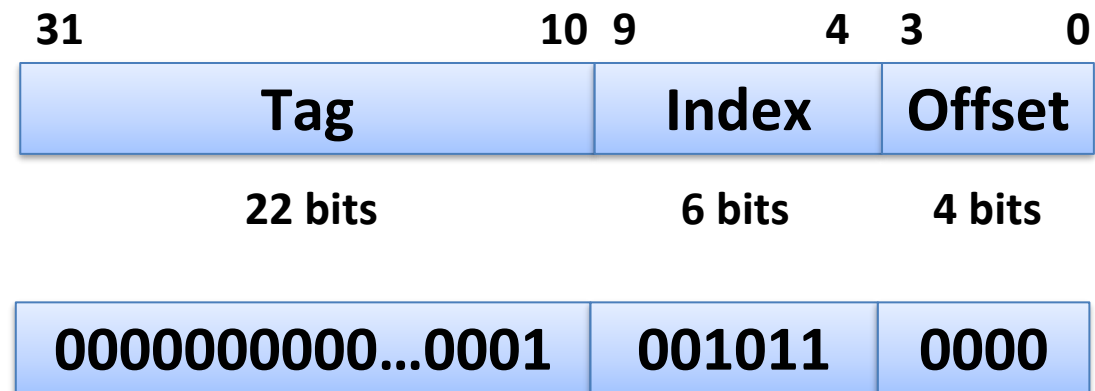
Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Address Subdivision



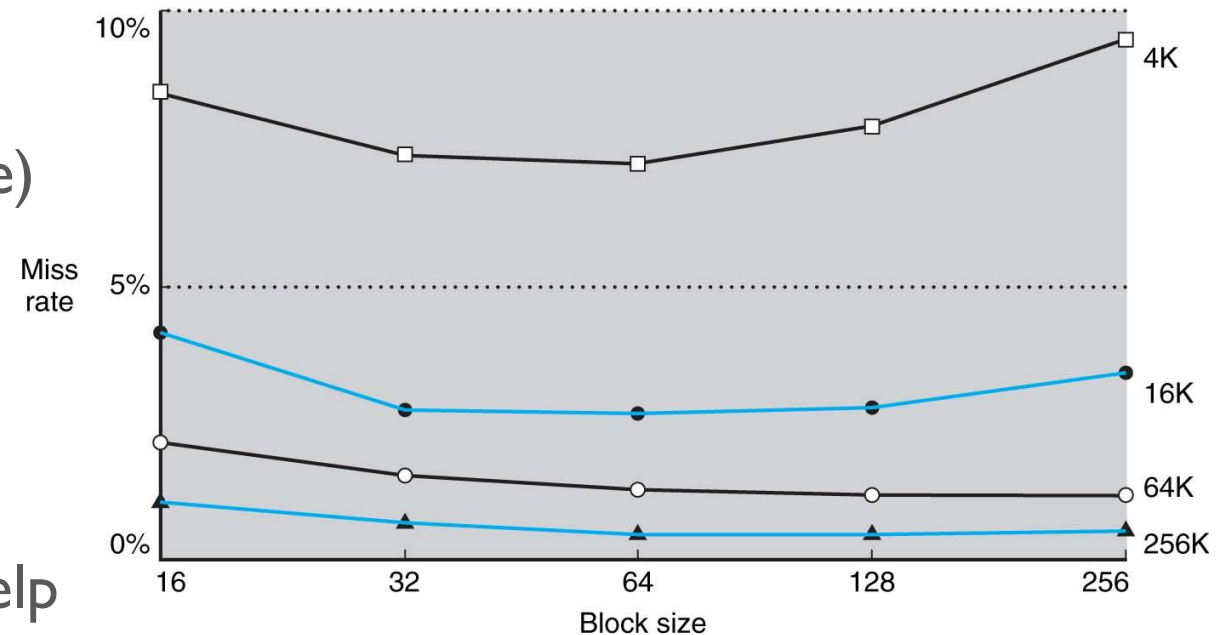
Example: Larger Block Size

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



Block Size Considerations

- Larger blocks should reduce miss rate
 - Due to spatial locality
- But in a fixed-sized cache
 - Larger blocks \Rightarrow fewer of them (more competition \Rightarrow increased miss rate)
 - Larger blocks \Rightarrow pollution
- Larger miss penalty
 - Can override benefit of reduce miss rate
 - Early restart and critical-word-first can help



Cache Misses

- On cache hit, CPU proceeds normally
- On cache miss
 - Stall the CPU pipeline
 - Fetch block from next level of hierarchy
 - Instruction cache miss
 - Restart instruction fetch
 - Data cache miss
 - Complete data access

Write-Through

- On data-write hit, could just update the block in cache
 - But then cache and memory would be inconsistent
- Write-through: also update memory
- But makes writes take longer
 - e.g., if base CPI = 1, 10% of instructions are stores, write to memory takes 100 cycles:
Effective CPU = $1 + 0.1 \times 100 = 11$
- Solution: write buffer
 - Holds data waiting to be written to memory
 - CPU continues immediately
 - Only stalls on write if write buffer is already full

Write-Back

- **Alternative: On data-write hit, just update the block in cache**
 - Keep track of whether each block is dirty (“dirty bit”)
- **When a dirty block is replaced**
 - Write it back to memory
 - Can use a write buffer to allow replacing block to be read first

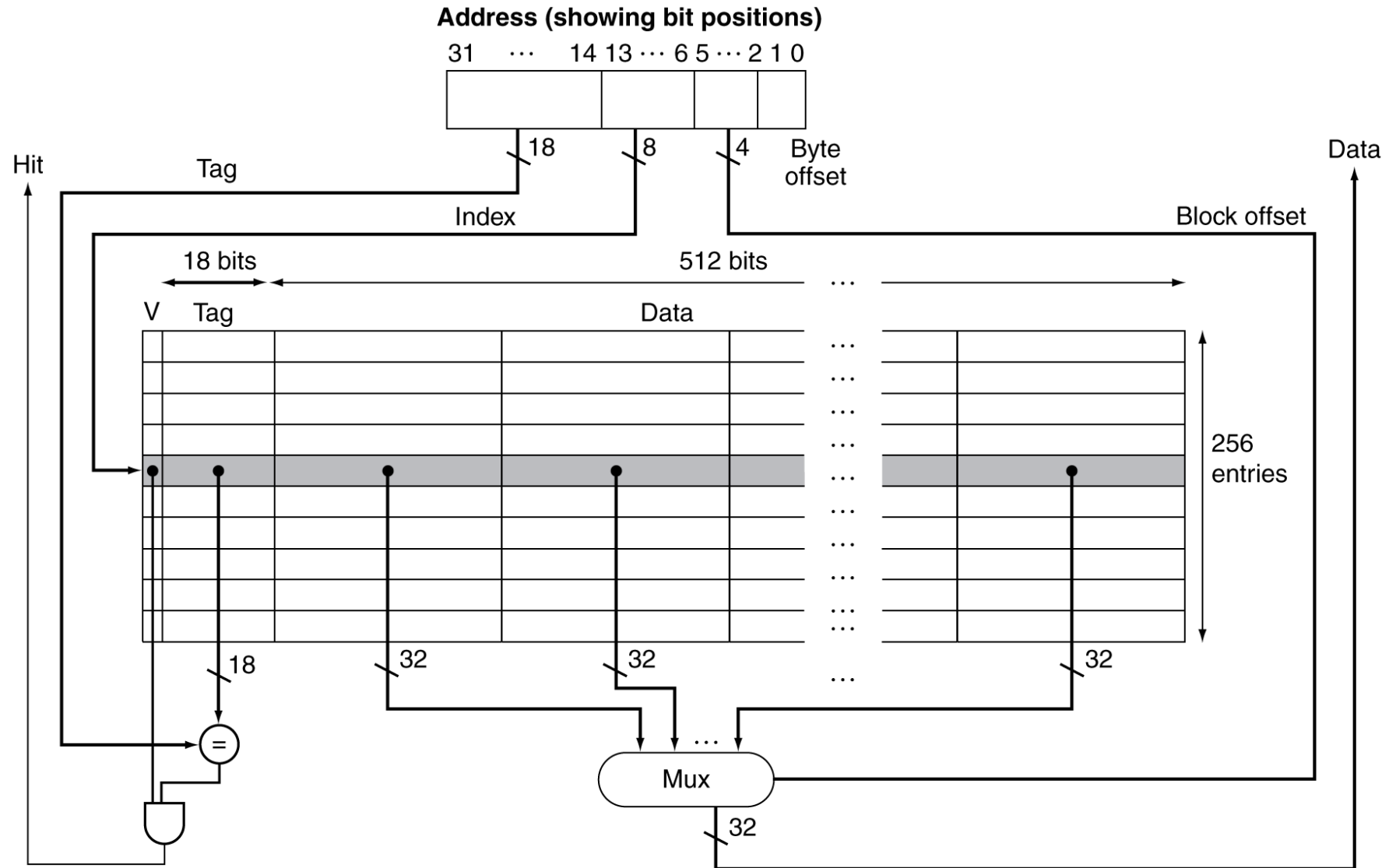
Write Allocation

- What should happen on a write miss?
- Alternatives for write-through
 - Allocate on miss: fetch the block
 - Write around: don't fetch the block
 - Since programs often write a whole block before reading it (e.g., initialization)
- For write-back:
 - Usually fetch the block

Example: Intrinsity FastMATH

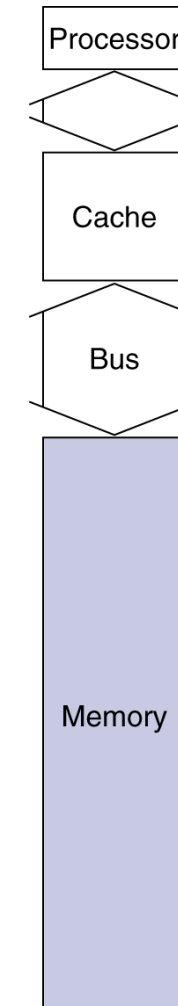
- **Embedded MIPS processor**
 - 12-stage pipeline
 - Instruction and data access on each cycle
- **Split cache: separate I-cache and D-cache**
 - Each 16KB: 256 blocks x 16 words / block (4 bytes / word)
 - D-cache: write-through or write-back
- **SPEC2000 miss rates**
 - I-cache: 0.4%
 - D-cache: 11.4%
 - Weighted average: 3.2%

Example: Intrinsicity FastMATH



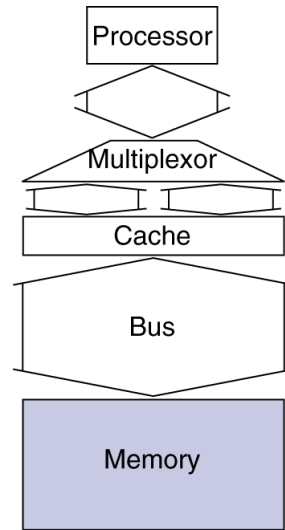
Main Memory and Cache

- Use DRAMs for main memory
 - Fixed width (e.g., 1 word)
 - Connected by fixed-width clocked bus
 - Bus clock is typically slower than CPU clock
- Example cache block read
 - 1 bus cycle for address transfer
 - 15 bus cycles per DRAM access
 - 1 bus cycle per data transfer
- For 4-word block, 1-word-wide DRAM
 - Miss penalty = $1 + 4 \times 15 + 4 \times 1 = 65$ bus cycles
 - Bandwidth = $16 \text{ bytes} / 65 \text{ cycles} = 0.25 \text{ bytes/cycle}$

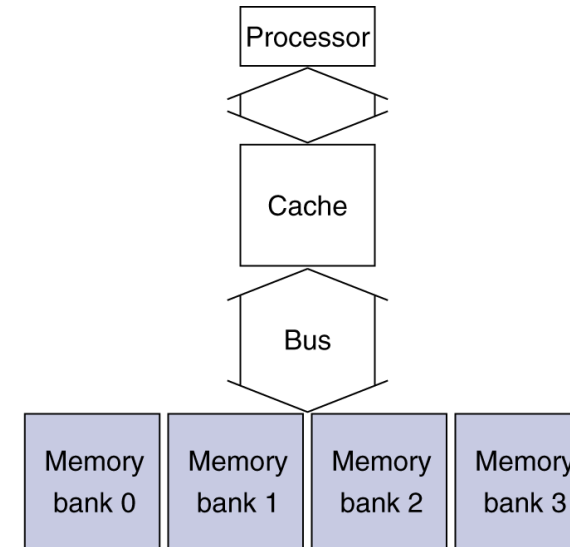


a. One-word-wide memory organization

Increasing Memory Bandwidth



b. Wider memory organization



c. Interleaved memory organization

■ 4-word wide memory

- Miss penalty = $1 + 15 + 1$
= 17 bus cycles
- Bandwidth = $16 \text{ bytes} / 17 \text{ cycles}$
= 0.94 bytes/cycle

■ 4-bank interleaved memory

- Miss penalty = $1 + 15 + 4 \times 1$
= 20 bus cycles
- Bandwidth = $16 \text{ bytes} / 20 \text{ cycles}$
= 0.8 bytes/cycle