CE/CZ 3001:
Advanced Computer Architecture

Module 5: Memory Systems

- cache design (cont.)

Asst Prof Liu Weichen

School of Computer Science and Engineering

Nanyang Technological University, Singapore

Summary of pre-recorded videos

- Memory hierarchy
- Cache design principles
- General organization of cache
- Cache placement policy
- Direct-mapped cache
- Fully associative cache
- Set associative cache
- Cache replacement policies
- Cache write policies
- Instruction cache vs data cache

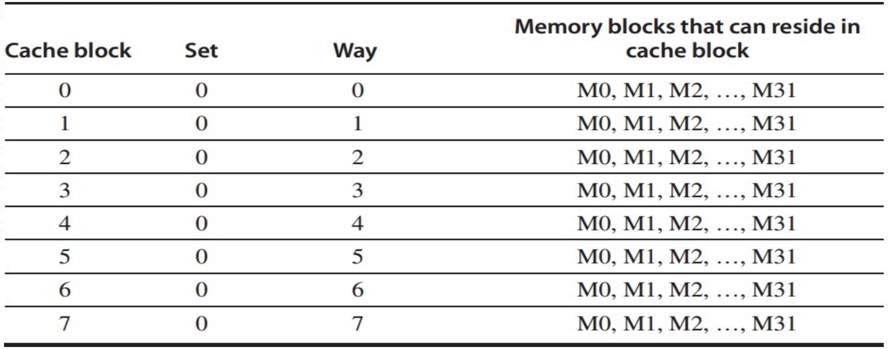
Summary: Cache Schemes

- Direct Mapped Cache
  - a block can only be in one place
- Set Associative Cache
  - a block can be in any entry of a set
    - tags of all the entries in the set must be searched
- Fully Associative Cache
  - blocks can go anywhere
  - tags of all the blocks in the cache must be searched
    - many parallel comparisons (a comparator for each cache entry)
    - practical only for caches with small number of blocks
- Increasing degree of associativity
  - typically decreases miss rate
  - increases hit time (due to extra comparison/selection)
    - design trade-off between miss penalty and area/time overhead

Excise:

We assume that we have a 512-byte cache with 64-byte blocks.
We assume that the main memory is 2 KB.
We can regard the memory as an array of 64-byte blocks: M0, M1, …, M31.

If the cache is fully associative:



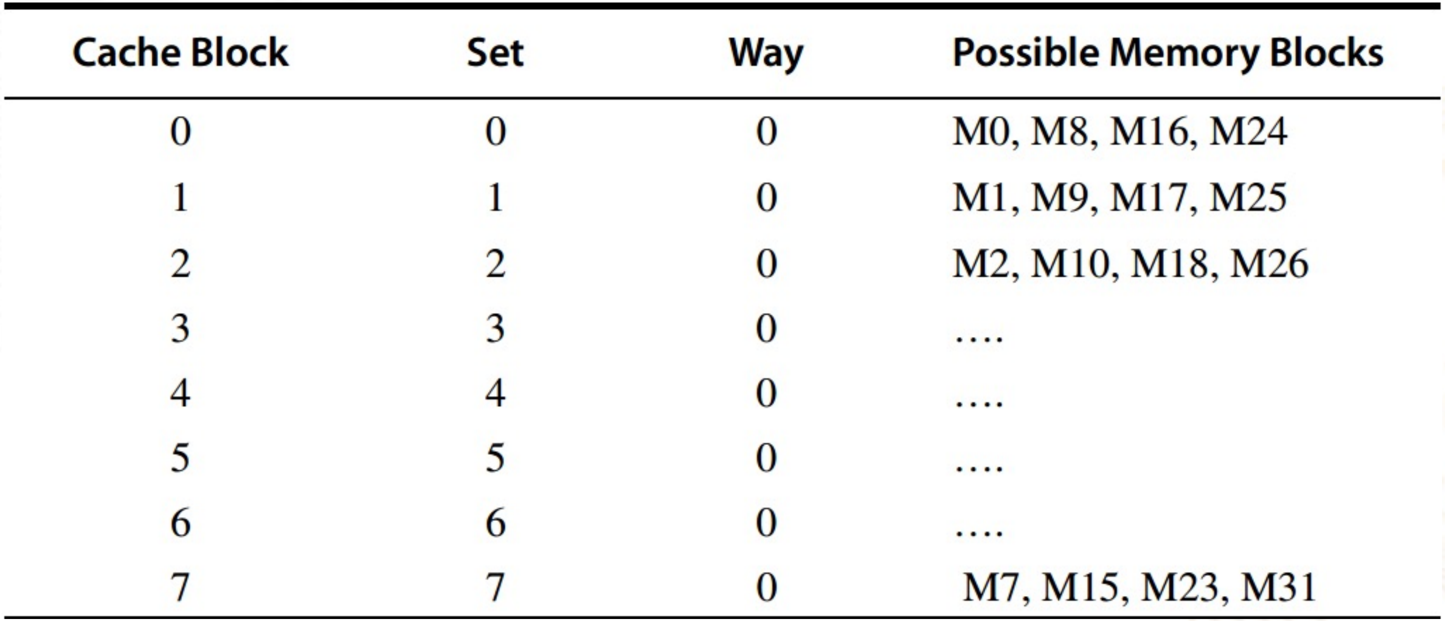
Cache block	Set	Way	Memory blocks that can reside in cache block
0	0	0	M0, M1, M2, …, M31
1	0	1	M0, M1, M2, …, M31
2	0	2	M0, M1, M2, …, M31
3	0	3	M0, M1, M2, …, M31
4	0	4	M0, M1, M2, …, M31
5	0	5	M0, M1, M2, …, M31
6	0	6	M0, M1, M2, …, M31
7	0	7	M0, M1, M2, …, M31

a. Show the content of the table if cache is organized as a direct mapped cache.

b. Repeat part (a) with the cache organized as a four-way set associative cache.

a. Show the content of the table if cache is organized as a direct mapped cache.

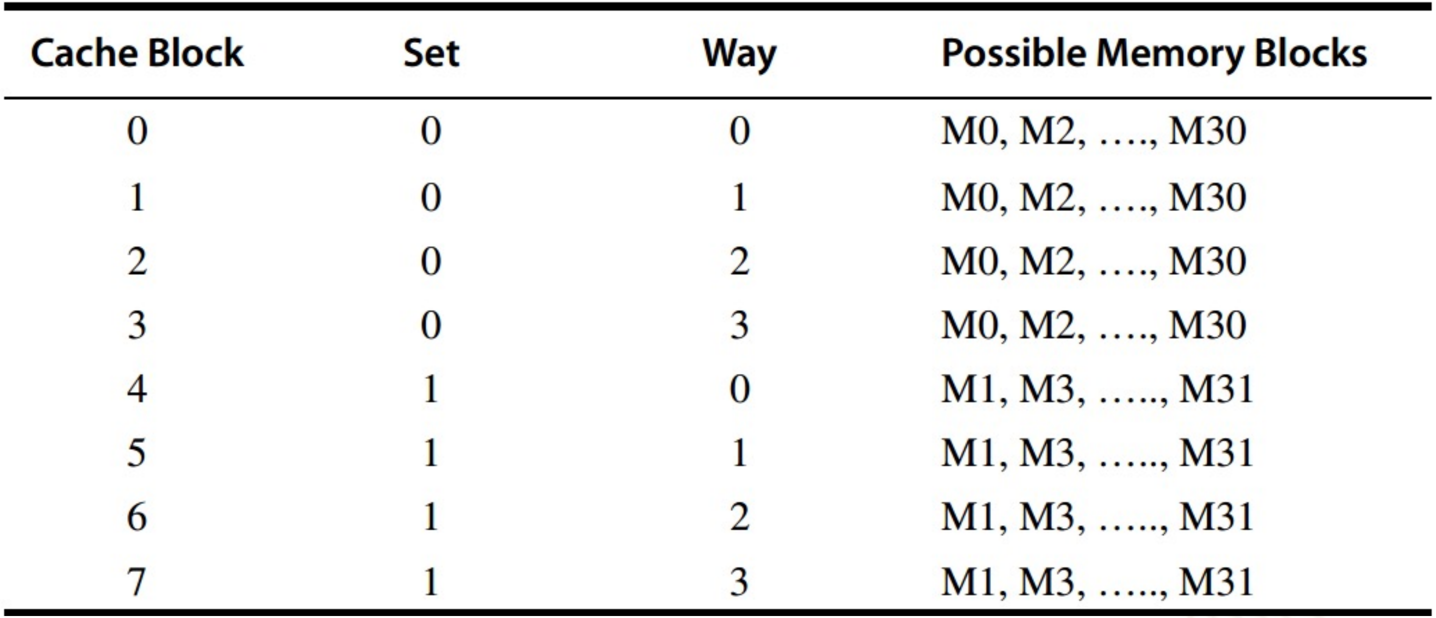
(Block address) MOD (Number of sets in cache)



Cache Block	Set	Way	Possible Memory Blocks
0	0	0	M0, M8, M16, M24
1	1	0	M1, M9, M17, M25
2	2	0	M2, M10, M18, M26
3	3	0	….
4	4	0	….
5	5	0	….
6	6	0	….
7	7	0	M7, M15, M23, M31

b. Repeat part (a) with the cache organized as a four-way set associative cache.

(Block address) MOD (Number of sets in cache)



Cache Block	Set	Way	Possible Memory Blocks
0	0	0	M0, M2, …., M30
1	0	1	M0, M2, …., M30
2	0	2	M0, M2, …., M30
3	0	3	M0, M2, …., M30
4	1	0	M1, M3, ….., M31
5	1	1	M1, M3, ….., M31
6	1	2	M1, M3, ….., M31
7	1	3	M1, M3, ….., M31

Block Replacement Policies on a Cache Miss

- Direct mapped cache
  - No choice in picking the “victim entry”: only one block frame is checked and if necessary replaced
- Set-associative or fully associative cache
  1. Random
     - to spread allocation uniformly
  2. LRU (Least-Recently Used)
     - application of principle of (temporal) locality
  3. FIFO (round-robin)
     - similar idea as LRU, but simpler/cheaper to implement
  4. NRU (not recently used)
     - approximation of LRU, but simpler/cheaper to implement

Cache Write Policies: Discussion

- Advantages of Write-Back Policy

  + CPU writes occur at the speed of the cache memory

  + multiple writes within a block require only one write to the lower-level memory

  + some writes don’t go to memory

  + write back uses less memory bandwidth (good for multiprocessors) and dissipate less power (good for embedded applications)

- Advantages of Write-Through Policy

  + easier to implement than write back

  + cache is always clean, so read misses are faster as they never result in writes to lower level

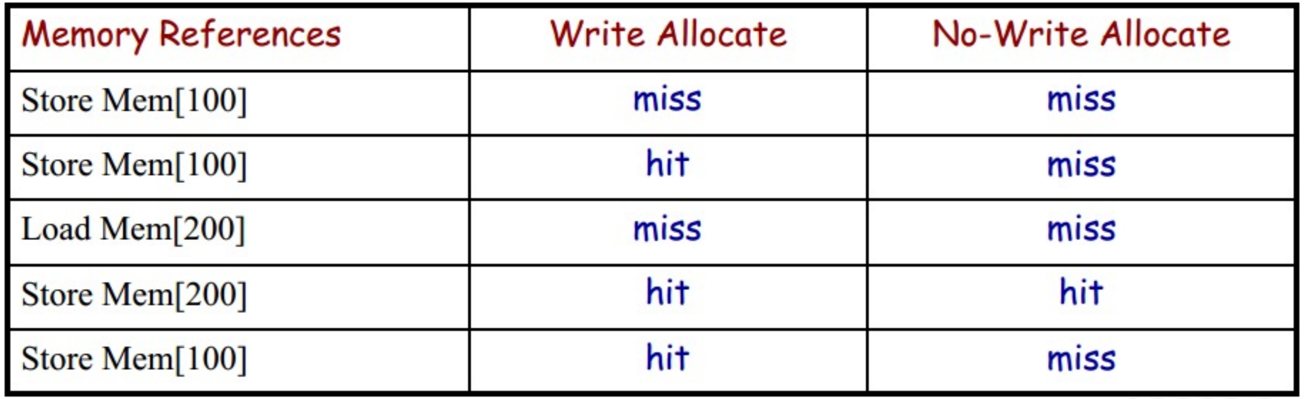
  + next lower level has the most current copy of the data

  + simplifies data coherency (good for multiprocessors and I/O)

What Happens on a Write Miss

- Since data are “not needed” on a write there are two options:
  - – Write Allocate (typically used by write-back caches)
    - the block is first allocated and then the same policy that is used for a write hit is executed
  - – No-Write Allocate (typically used by write-through caches)
    - the cache is not affected by a write miss while the block is modified only in the lower-level memory
      - – thus a block stays out of the cache until the program tries to read it

Example

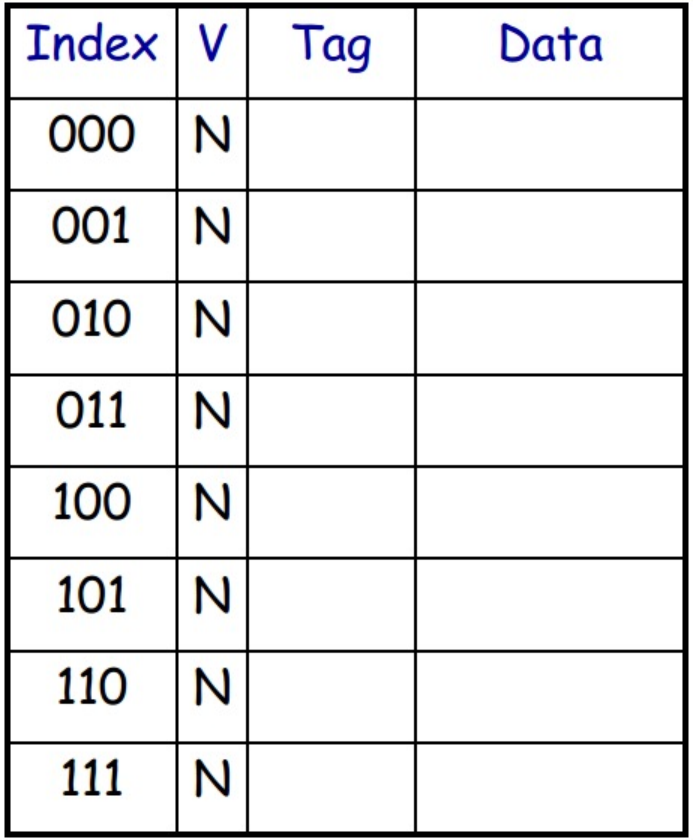


Memory References	Write Allocate	No-Write Allocate
Store Mem[100]	miss	miss
Store Mem[100]	hit	miss
Load Mem[200]	miss	miss
Store Mem[200]	hit	hit
Store Mem[100]	hit	miss

Summary: Cache Design

- Cache
  - Organization schemes
    - direct mapped
    - set associative
    - fully associative
  - Replacement policies
    - random, FIFO, LRU, etc.
  - Write policies
    - write back (with write allocate for write miss)
    - write through (with no-write allocate)

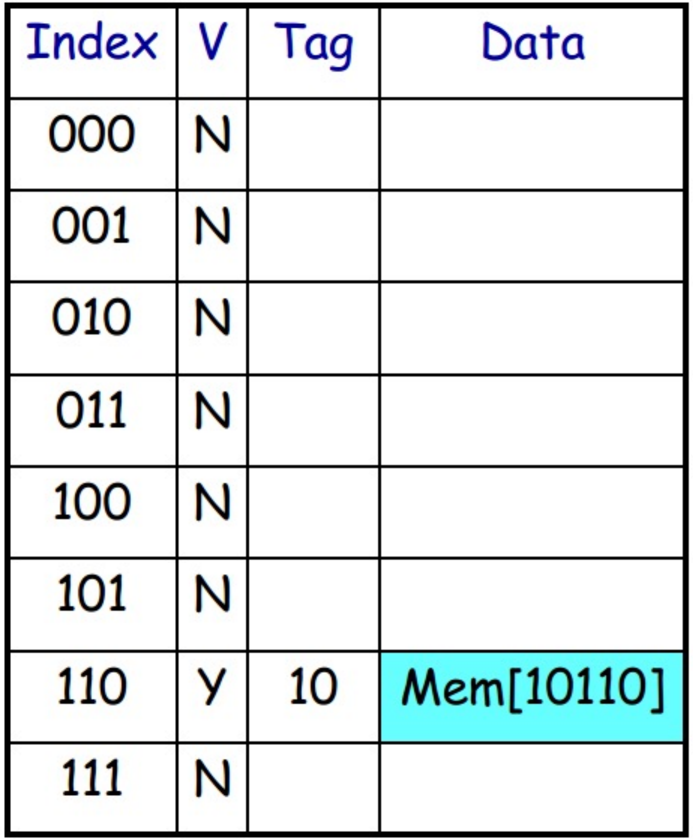
Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word



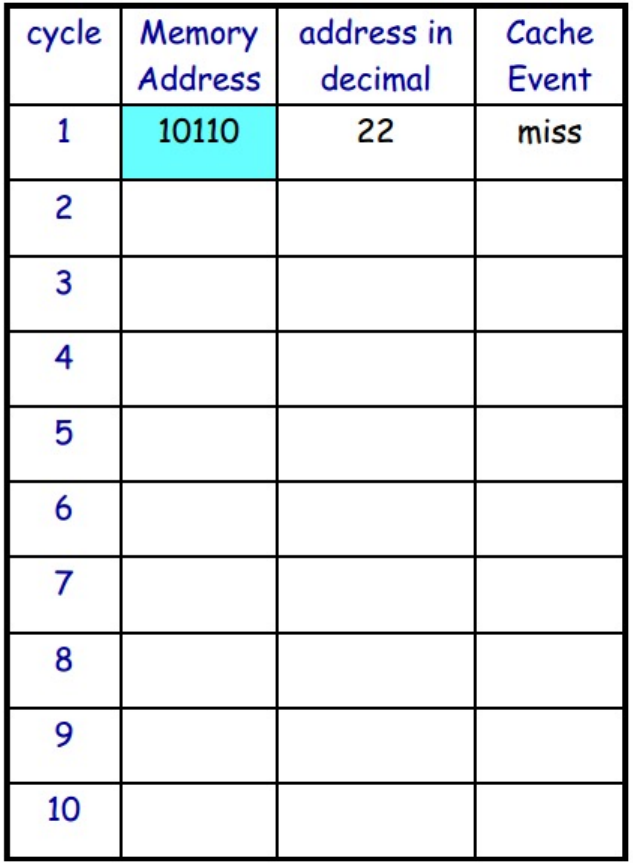
Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

- Assumption
  - Word addressing
  - –8 block frames
  - – block size = 1 word
  - – main memory of 32 words
  - – we consider ten subsequent accesses to memory

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word

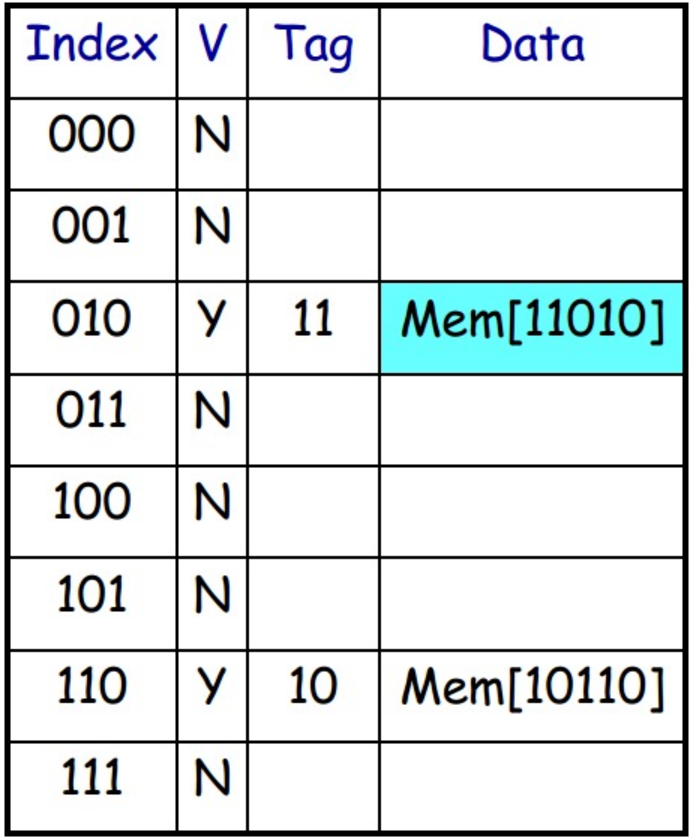


Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

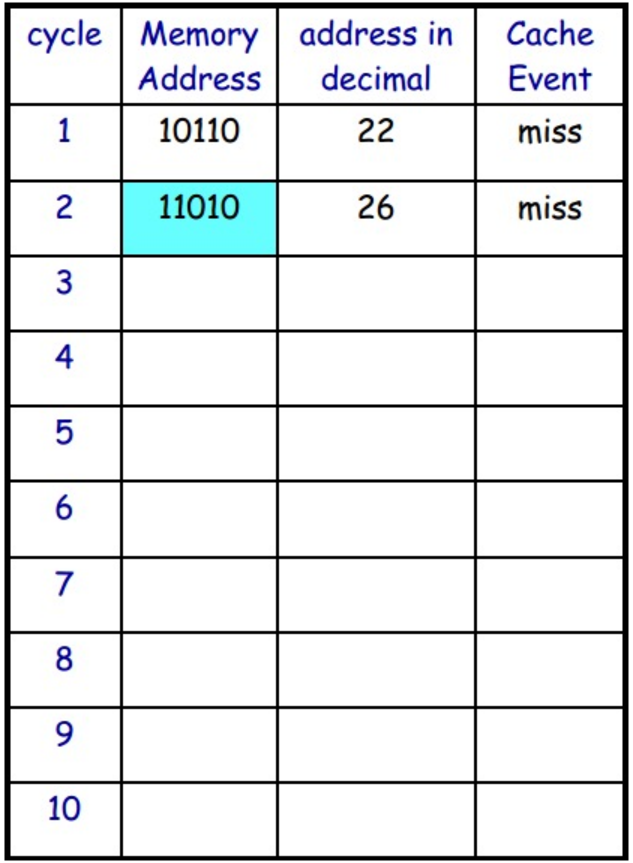


cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2			
3			
4			
5			
6			
7			
8			
9			
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word



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		



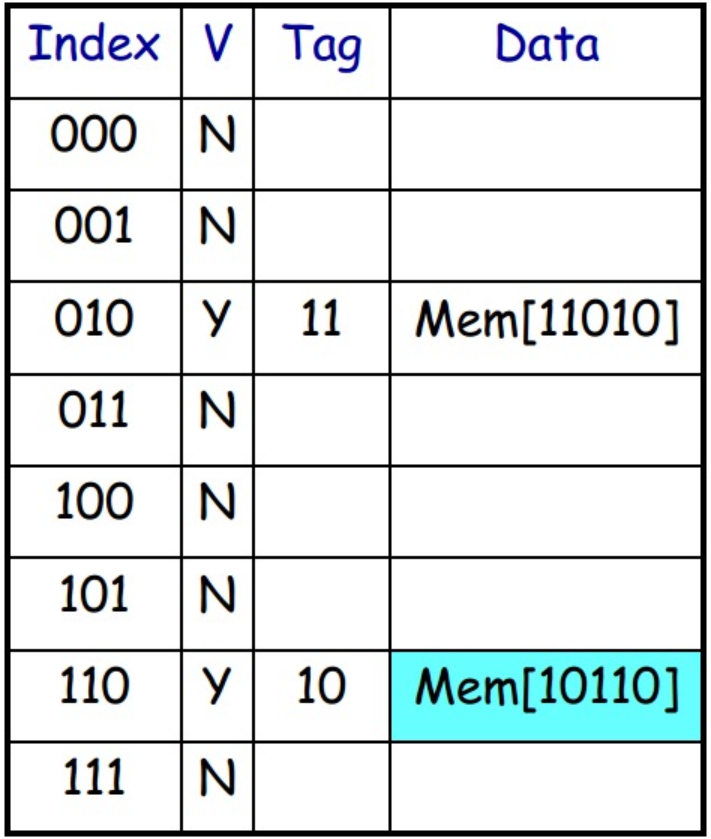
cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3			
4			
5			
6			
7			
8			
9			
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word

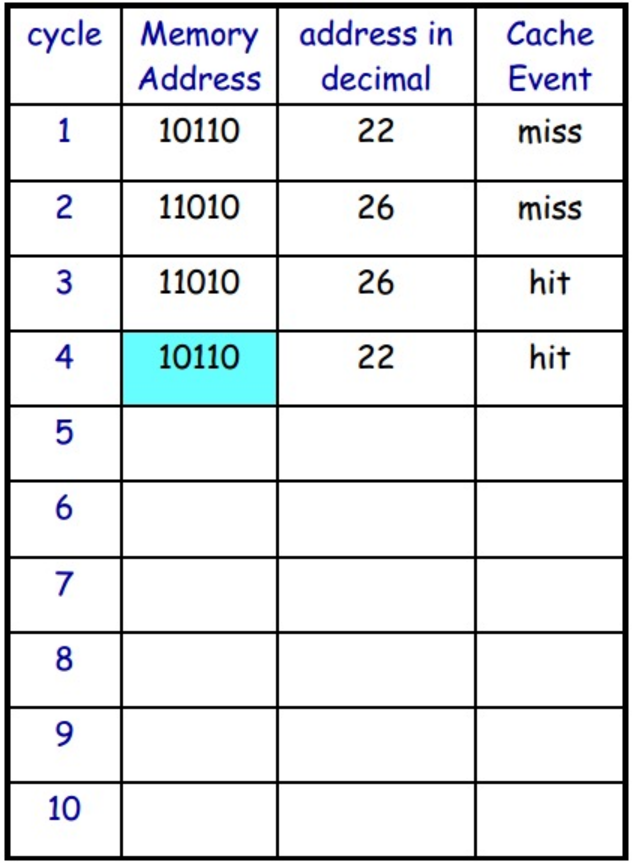
Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3	11010	26	hit
4			
5			
6			
7			
8			
9			
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word



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		



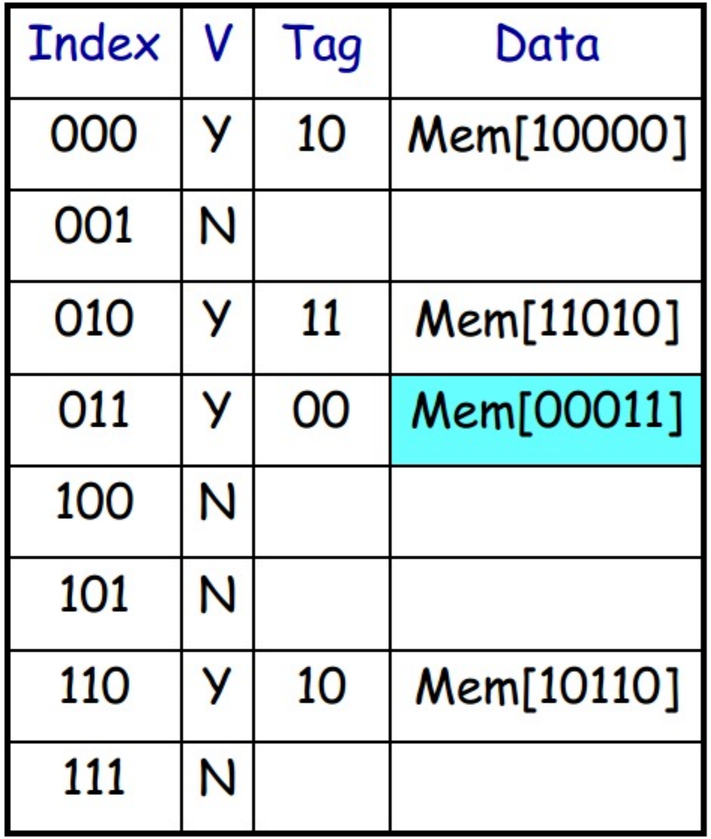
cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3	11010	26	hit
4	10110	22	hit
5			
6			
7			
8			
9			
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word

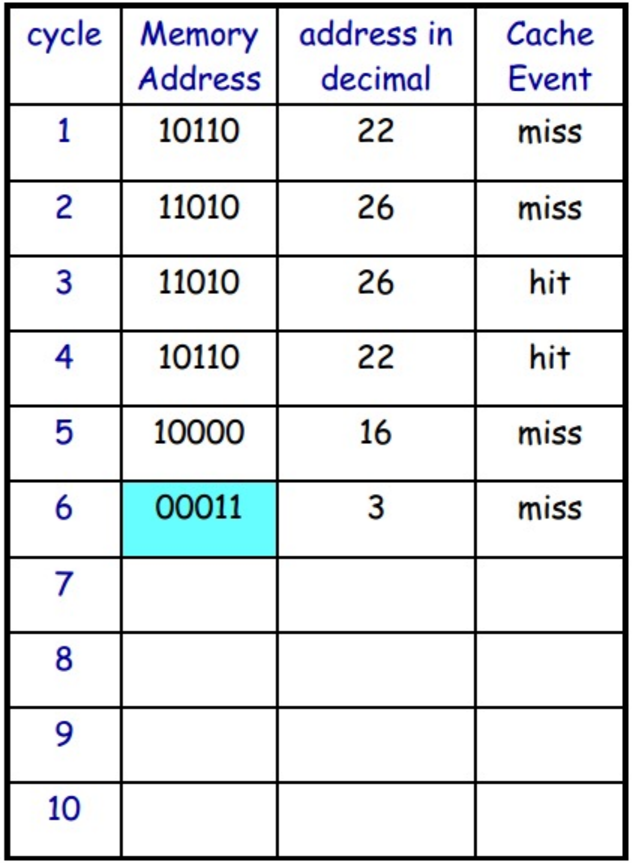
Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3	11010	26	hit
4	10110	22	hit
5	10000	16	miss
6			
7			
8			
9			
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word



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		



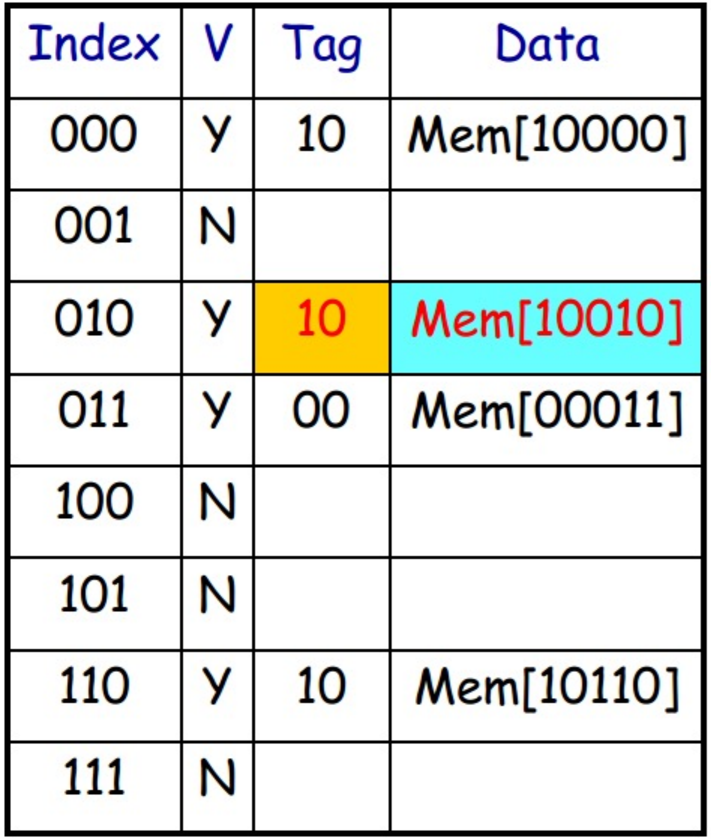
cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3	11010	26	hit
4	10110	22	hit
5	10000	16	miss
6	00011	3	miss
7			
8			
9			
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word

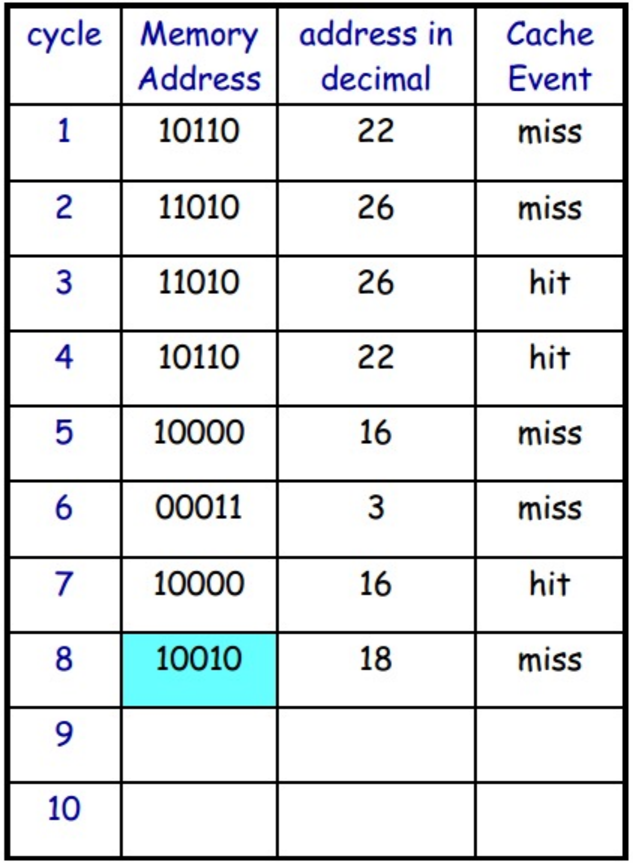
Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3	11010	26	hit
4	10110	22	hit
5	10000	16	miss
6	00011	3	miss
7	10000	16	hit
8			
9			
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word

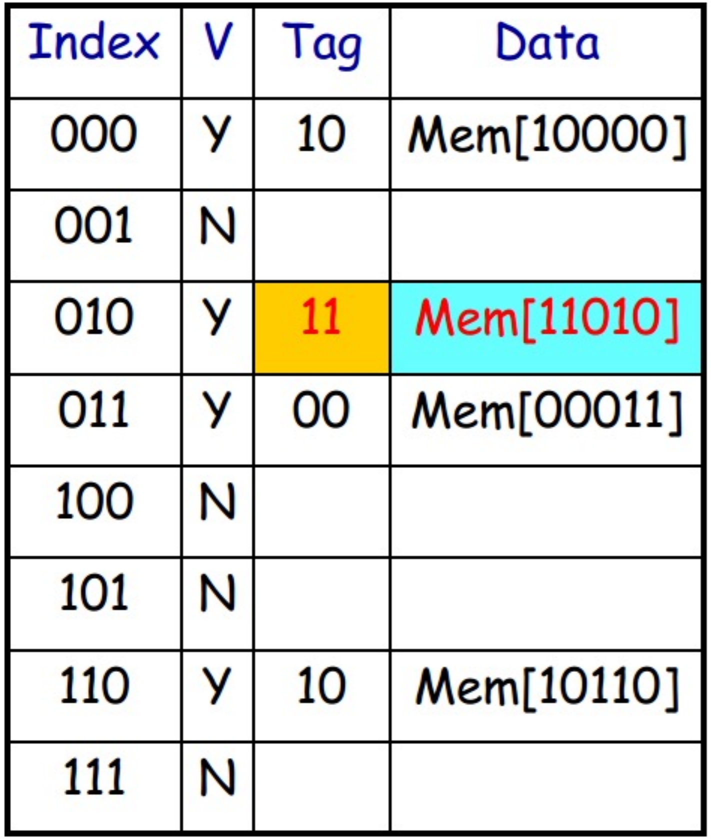


Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	10	Mem[10010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

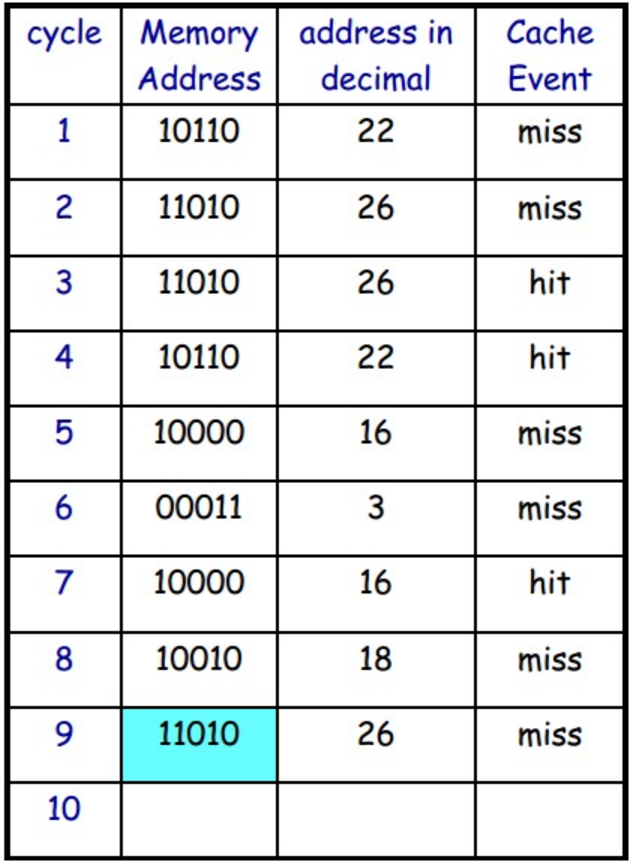


cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3	11010	26	hit
4	10110	22	hit
5	10000	16	miss
6	00011	3	miss
7	10000	16	hit
8	10010	18	miss
9			
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word

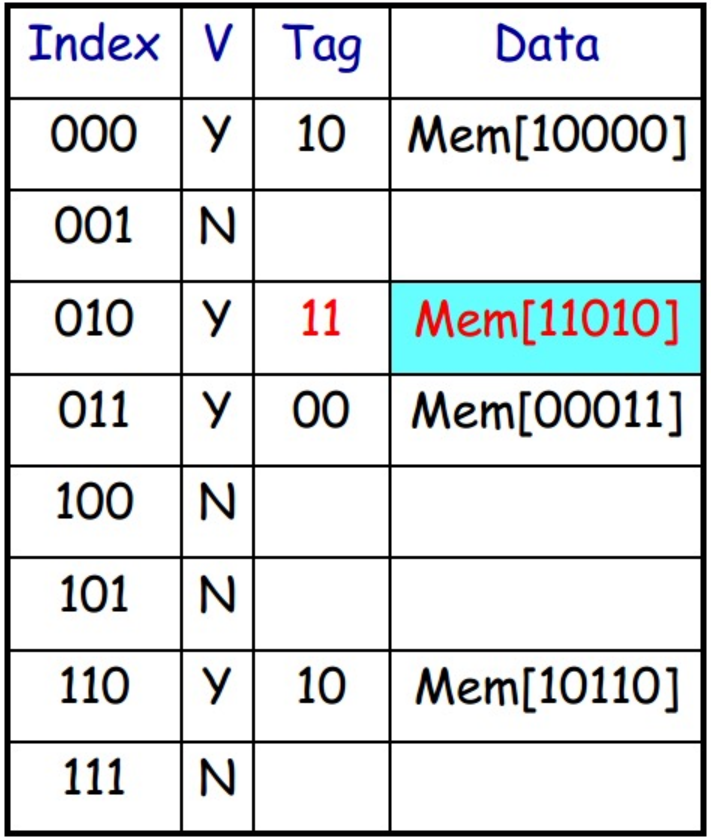


Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

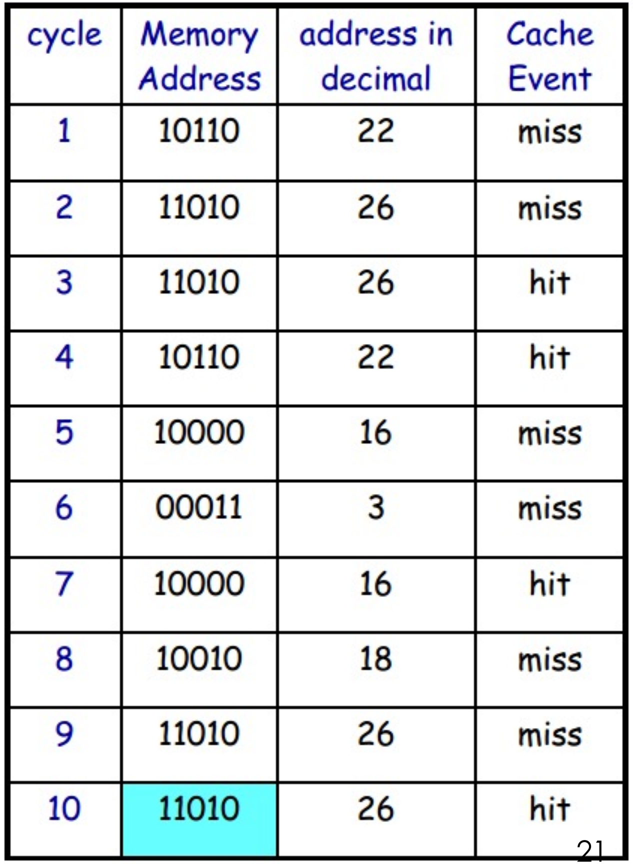


cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3	11010	26	hit
4	10110	22	hit
5	10000	16	miss
6	00011	3	miss
7	10000	16	hit
8	10010	18	miss
9	11010	26	miss
10			

Example: Accessing a Direct Mapped Cache with 8 Blocks and Block Size of 1 Word



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		



cycle	Memory Address	address in decimal	Cache Event
1	10110	22	miss
2	11010	26	miss
3	11010	26	hit
4	10110	22	hit
5	10000	16	miss
6	00011	3	miss
7	10000	16	hit
8	10010	18	miss
9	11010	26	miss
10	11010	26	hit