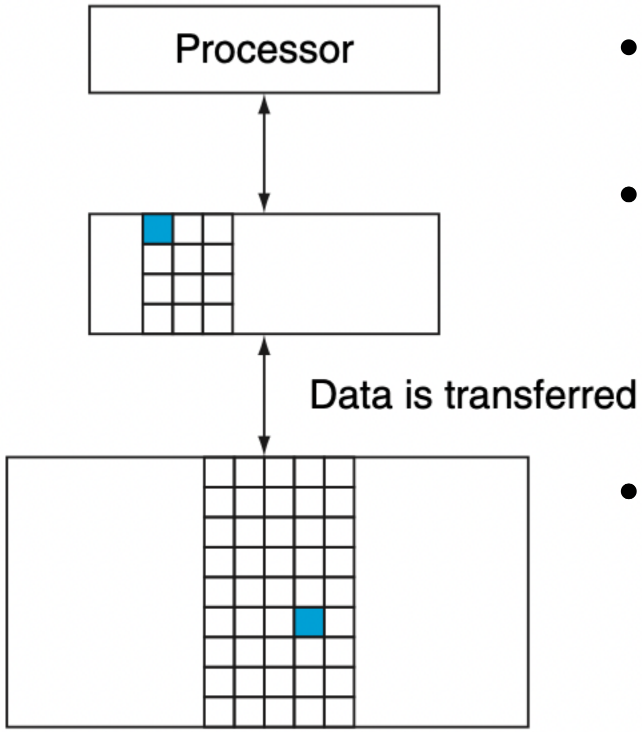
Computer Architecture
(ENE1004)

Lec – 20: Large and Fast: Exploiting Memory Hierarchy (Chapter 5) – 2

Schedule

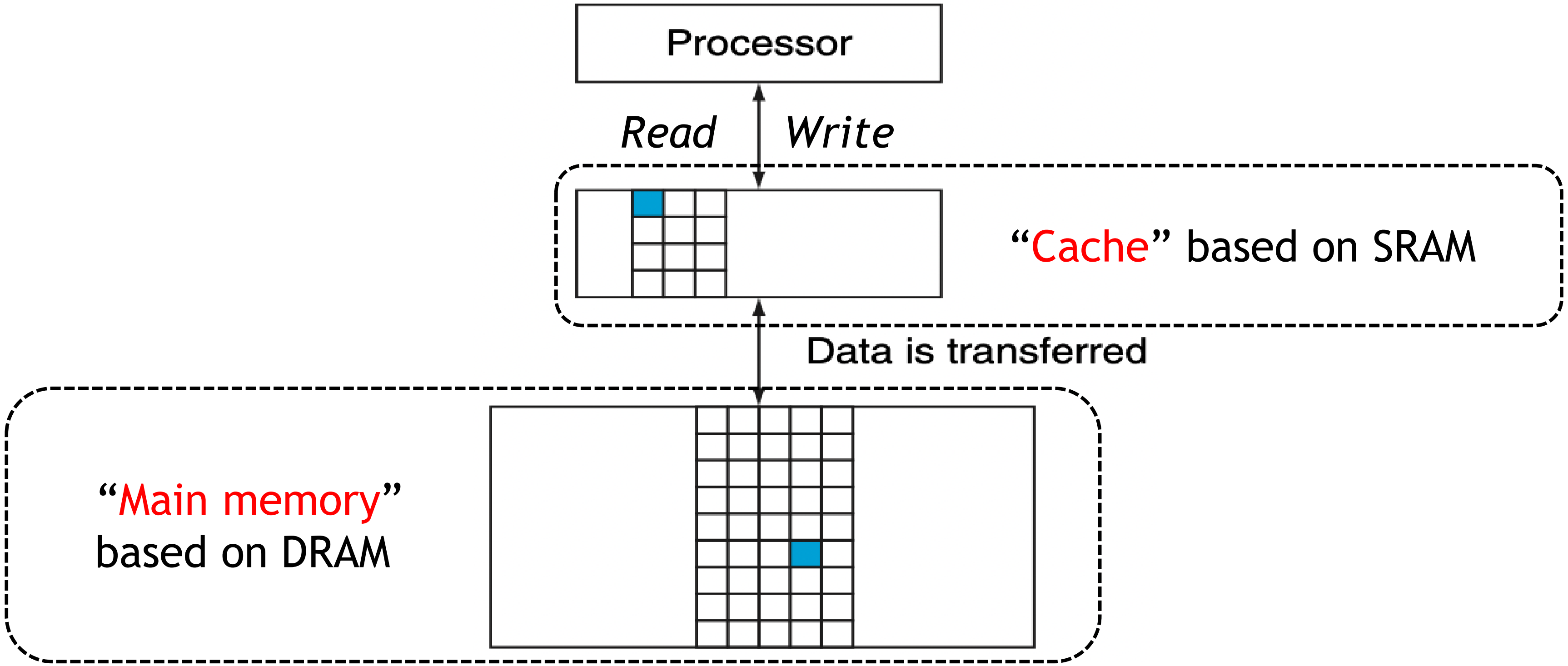
- Final exam: Jun. 19, Monday
    - (24334)1:25~2:25pm
    - (23978) 2:30~3:30pm
    - Sample questions will be provided by Jun. 17 (Saturday)
- Assignment #2: Jun. 20, Tue. by midnight
    - It is put back as much as possible
- Remaining class days
    - 1(Thur), 5(Mon), 8(Thur), 12(Mon), 15 (Thur)

Memory Terminology



- Every pair of levels in the hierarchy can be thought of as having an upper and lower level
  - Data is copied between only two adjacent levels at a time
- "Block (line)" - The minimum unit of data item that can be either present or not present in the two levels is called "block"
- "Hit" - If the data requested by the processor appears in some block in the upper level, the request is called a "hit"
  - It can be serviced quickly from the upper-level faster memory
  - "Hit time" - The time to access the upper level memory
- "Miss" - If the data is not found in the upper level (instead, it can be found in the lower level), it is called "miss"
  - The data is copied from the lower level to the upper level; then is read
  - The total time is "miss penalty"
- "Hit ratio" is the fraction of accesses found in the upper level
  - The higher the hit ratio, the higher the performance
  - "Miss ratio" = 1 - "hit ratio"

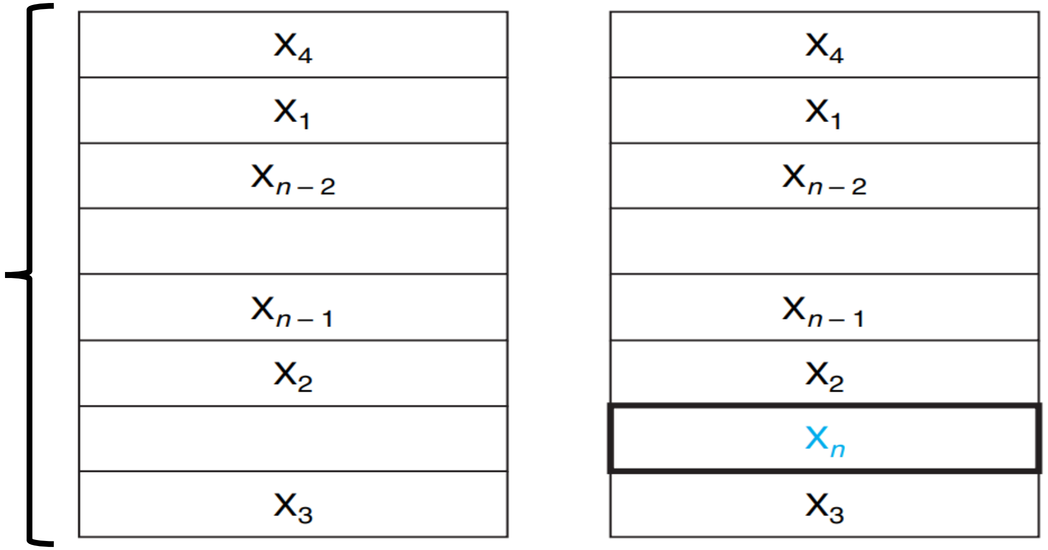
Basics of Caches (1)



- In this course, we’ll focus on the most upper level two memory technologies
  - SRAM in upper level is called “cache”
  - DRAM in lower level is called “main memory”
  - Processor always works with the cache
  - Data set in the cache changes by transferring data between the cache and the main memory
- Specifically, we’ll see how a cache is designed and managed

Basics of Caches (2)

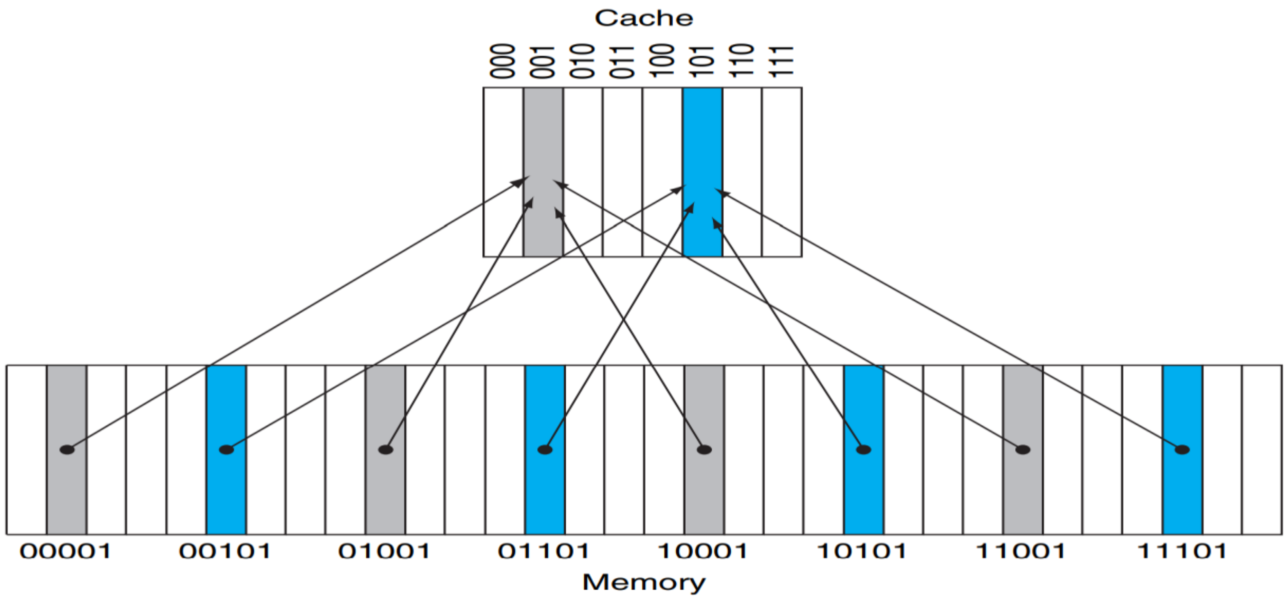
"Cache" with 8 blocks, which can hold up to 8 data items at a time



a. Before the reference to X_n b. After the reference to X_n

- An example of simple cache
  - The cache consists of 8 blocks (8 lines); currently, it holds 6 data items and two blocks are empty
  - Given a request to one of the 6 data items, it is "hit" and the item is provided from the cache
  - Given a request to data item (X_n), which does not exist in the cache, it is "miss" and the data item is brought from the main memory into the cache
- Here, one may have the following important questions
  - (1) How do we know if a data item is in the cache?
  - (2) If it is, how do we find it?

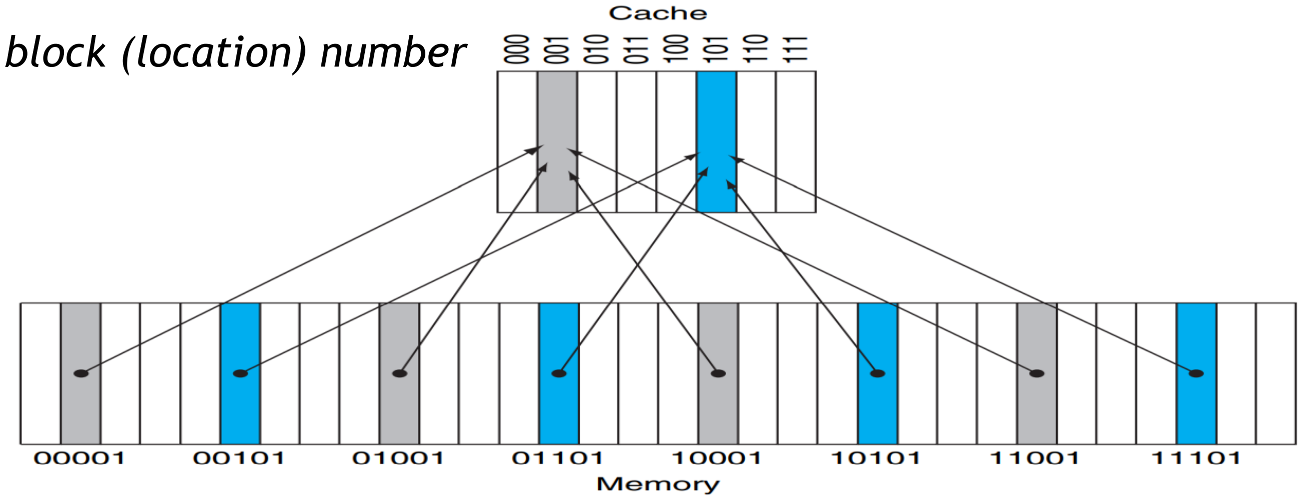
Direct Mapped Cache (1)



- One answer: If each data item can go in exactly one place in the cache, then it is straightforward to find the data item if it is in the cache
  - Given a request, we can just check the designated single place instead of scanning all the places
- Direct-mapped cache: each memory location is mapped directly to one location in the cache
  - It assigns a location in the cache to each data item
  - The assignment of a cache location is based on "the address of the data item" in memory
- Note that multiple data items should share a cache location
  - 8 cache blocks are assigned to 32 data items; 4 different data items should share a cache block
  - If a data item uses the block, the other three items cannot use the same block

Direct Mapped Cache (2) - Index

“Index” - cache block (location) number

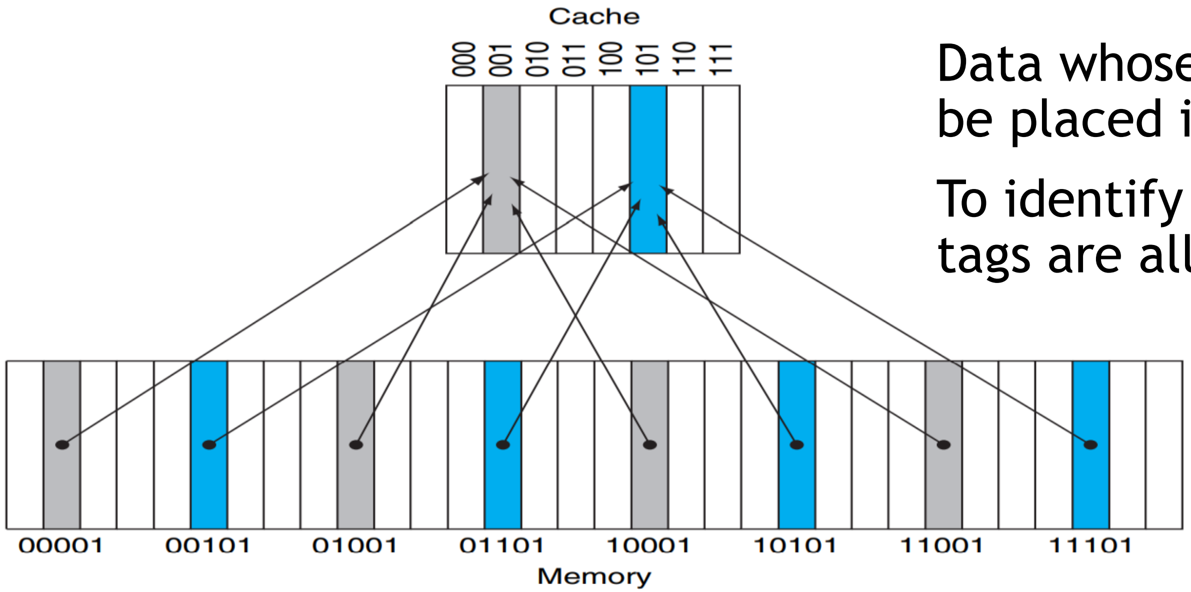


Address of each data item

- Modulo operation
  - 7 module 3 = 1
  - 5 module 3 = 2
  - 10 module 3 = 1

- One simple way to assign a cache block is to use the address of the data item
  - We call the cache block number “index”
  - Index = (Address) modulo (Number of blocks in the cache)
- In the example, the number of blocks in the cache is 8 (2^3)
  - Memory address 1_{ten} (00001_{two}) is mapped to cache block 1_{ten} (001_{two}); 1 module 8 = 1
  - Memory addresses 9_{ten} (01001_{two}), 17_{ten} (10001_{two}), and 25_{ten} (11001_{two}) are also mapped to cache block 1_{ten} (001_{two}); 9 module 8 = 1, 17 module 8 = 1, and 25 module 8 = 1
  - Memory addresses 5_{ten} (00101_{two}), 13_{ten} (01101_{two}), 21_{ten} (10101_{two}), and 29_{ten} (11101_{two}) are all mapped to cache block 5_{ten} (101_{two})
- Simply, check the last 3 bits of the memory address; the size of cache (2^3)!
  - 00001_{two}, 01001_{two}, 10001_{two}, 11001_{two} ; the index of these four data items is 001_{two}
  - 00101_{two}, 01101_{two}, 10101_{two}, 11101_{two} ; the index of these four data items is 101_{two}

Direct Mapped Cache (3) - Tag

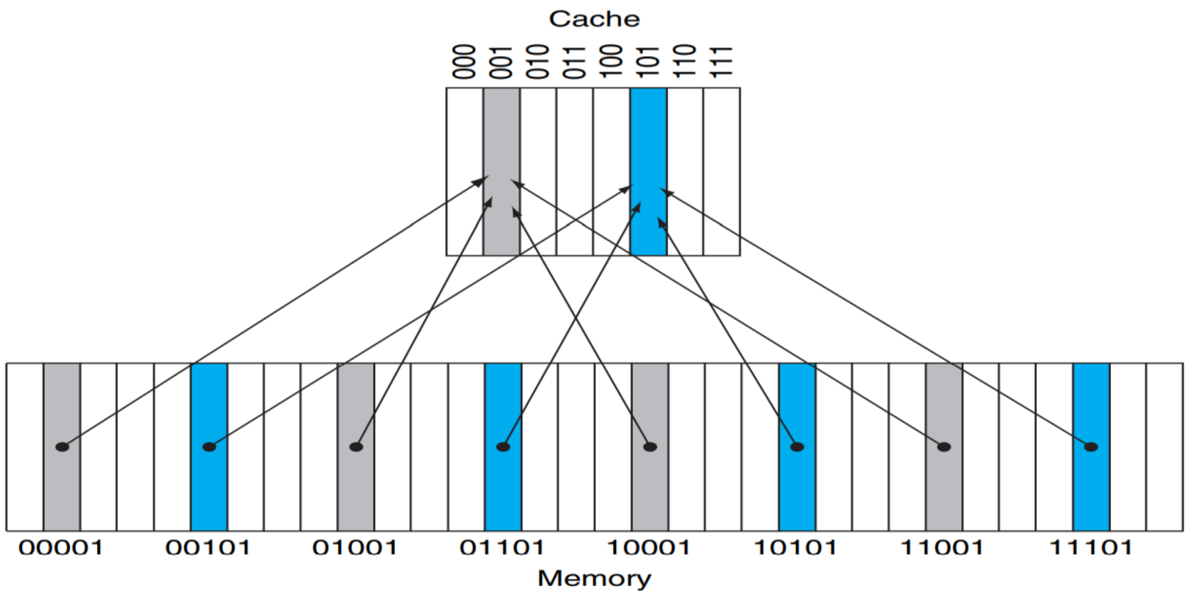


Data whose addresses are 00001, 01001, 10001, and 11001 can be placed in cache block 001; they have the same index!

To identify one of the four, tags (00, 01, 10, 11) can be used; tags are all different!

- A cache block can contain the different data items
  - Question: How do we know whether the requested data corresponds to the data in the cache?
  - We can answer the question by adding "tag" to each cache block
- The address of data is divided and used for two different purposes
  - The lower bits are used for "index" (to identify its cache block)
  - The upper bits are used for "tag" (to identify the specific data)
  - (b4 b3 b2 b1 b0) is divided; the lower 3 bits are "index", which is used for a cache blcok; the upper 2 bits are "tag", which is used to indicate the specific data

Direct Mapped Cache (4) – Valid Bit



- A cache block does not always hold a valid data
  - The cache block can be empty, if none of the four data exist in the cache
  - When a processor starts up, the cache does not have any data
- We can address the above by adding a “valid bit” to each cache block
  - If it is set, the cache block has a valid data; then, you need to check the tag whether the valid data is what you want or not
  - If it is not set, you do not have to check the tag, since any information in the cache block is not valid

Access a Direct Mapped Cache (1)

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

a. The initial state of the cache after power-on

- Each entry of the cache consists of “index” + “valid bit” + “tag” + “data”
  - There are 8 entries (blocks) in the cache, each of which is identified based on the 3-bit “index”
  - If the 1-bit “valid bit” tells whether the information in the entry is valid or not
  - If a data is placed in its corresponding entry, its 2-bit “tag” is set to indicate the specific data
- At first, the cache is empty (it does not hold any data) right after power-on
  - The valid bit of each entry is set to “N”
  - The tag and data are also empty

Access a Direct Mapped Cache (2)

Decimal address of reference	Binary address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
22	10110_{two}	miss (5.6b)	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	miss (5.6c)	$(11010_{two} \bmod 8) = 010_{two}$
22	10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	miss (5.6d)	$(10000_{two} \bmod 8) = 000_{two}$
3	00011_{two}	miss (5.6e)	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
18	10010_{two}	miss (5.6f)	$(10010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

b. After handling a miss of address (10110_{two})

- An example scenario where a series of requests (references) are given to the cache
  - If the referenced data is in the cache, it is "hit" and the data is serviced from the cache
  - If the referenced data is not in the cache, it is "miss" and the existing data is replaced with the referenced data, which is copied from the main memory
- Data 22 (10110_{two}) is requested
  - It can be placed in the cache block whose "index" is 110_{two} (the lower 3 bits 10110_{two})
  - It is "miss" because the valid bit is set to N (Data 22 does not exist in the cache block)
  - Data 22 is brought from the main memory and stored in the cache block whose index is 110_{two}
  - Tag is set to 10_{two} to indicate data 22 (the upper 2 bits of 10110_{two}); valid bit is set to Y

Access a Direct Mapped Cache (3)

Decimal address of reference	Binary address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
22	10110_{two}	miss (5.6b)	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	miss (5.6c)	$(11010_{two} \bmod 8) = 010_{two}$
22	10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	miss (5.6d)	$(10000_{two} \bmod 8) = 000_{two}$
3	00011_{two}	miss (5.6e)	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
18	10010_{two}	miss (5.6f)	$(10010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

c. After handling a miss of address (11010_{two})

- Data 26 (11010_{two}) is requested
  - It can be placed in the cache block whose "index" is 010_{two} (the lower 3 bits 11010_{two})
  - It is "miss" because the valid bit is set to N (Data 26 does not exist in the cache block)
  - Data 26 is brought from the main memory and stored in the cache block whose index is 010_{two}
  - Tag is set to 11_{two} to indicate data 26 (the upper 2 bits of 11010_{two}); valid bit is set to Y
- Data 22 (10110_{two}) is requested again
  - Go to the index 110_{two}; first, check its valid bit Y; then, check the tag 10_{two} ; it is "hit"
- Data 26 (11010_{two}) is requested again
  - Go to the index 010_{two}; first, check its valid bit Y; then, check the tag 11_{two} ; it is "hit"

Access a Direct Mapped Cache (4)

Decimal address of reference	Binary address of reference	Hit or miss in cache	Assigned cache block (where found or placed)
22	10110_{two}	miss (5.6b)	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	miss (5.6c)	$(11010_{two} \bmod 8) = 010_{two}$
22	10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	miss (5.6d)	$(10000_{two} \bmod 8) = 000_{two}$
3	00011_{two}	miss (5.6e)	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
18	10010_{two}	miss (5.6f)	$(10010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

d. After handling a miss of address (10000_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

e. After handling a miss of address (00011_{two})

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

f. After handling a miss of address (10010_{two})