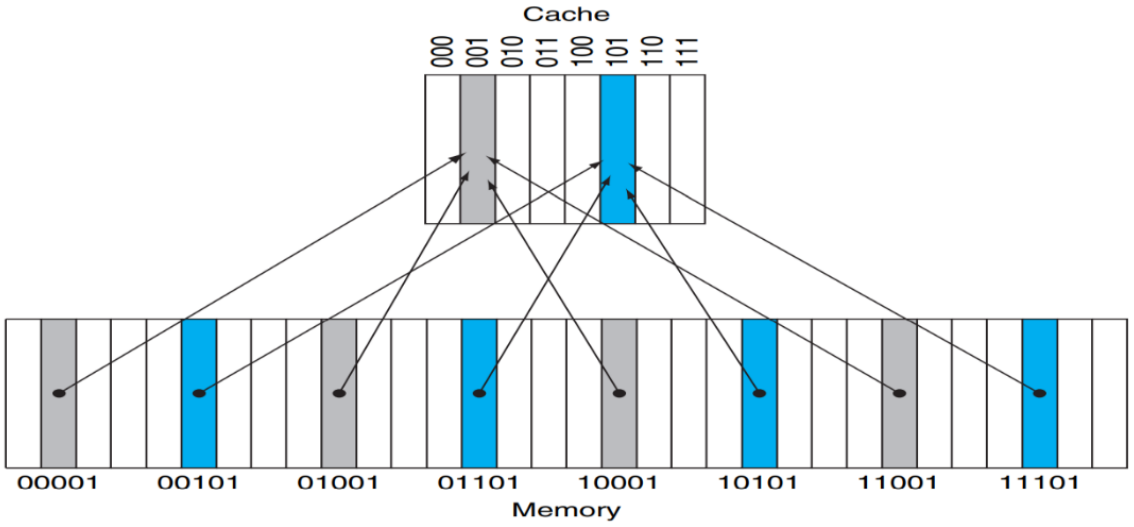
Computer Architecture (ENE1004)

▼ Lec 18

Lec 18: Large and Fast : Exploting Memory Hierarchy 2 (메모리 계층 구조 살펴보기 2)

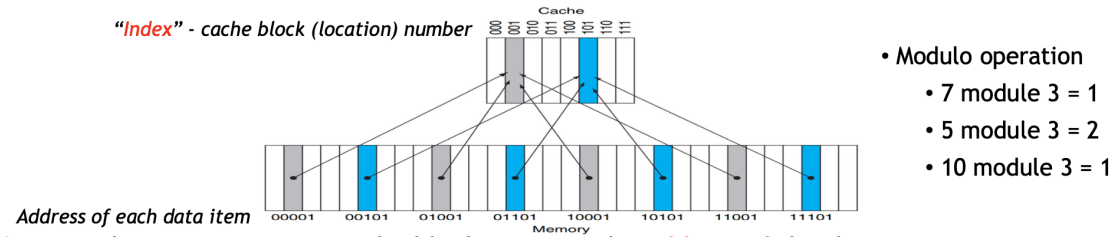
Direct Mapped Cache (1)



- 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 (한 가지 답변 : 각 데이터 항목이 캐시에서 정확히 한 곳에 들어갈 수 있다면, 데이터 항목이 캐시에 있으면 해당 데이터 항목을 찾는 것이 간단하다.)
    - Given a request, we can just check the designated single place instead of scanning all the places (요청이 주어지면 모든 위치를 스캔하는 대신 지정된 단일 위치만 확인하면 된다.)
- Direct-mapped cache: each memory location is mapped directly to one location in the cache (직접 매핑 캐시 : 각 메모리 위치가 캐시의 한 위치에 직접 매핑된다.)
    - It assigns a location in the cache to each data item (캐시 내 위치를 각 데이터 항목에 할당한다.)

    - The assignment of a cache location is based on “the address of the data item” in memory (캐시 위치 할당은 메모리의 “데이터 항목 주소”를 기반으로 한다.)
- Note that multiple data items should share a cache location (여러 데이터 항목이 캐시 위치를 공유해야 한다.)
    - 8 cache blocks are assigned to 32 data items; 4 different data items should share a cache block (32개 데이터 항목에 8개 캐시 블록이 할당되며, 4개의 다른 데이터 항목이 캐시 블록을 공유해야 한다.)
    - If a data item uses the block, the other three items cannot use the same block (한 데이터 항목이 블록을 사용하는 경우 다른 세 항목은 동일한 블록을 사용할 수 없다.)

Direct Mapped Cache (2) - Index



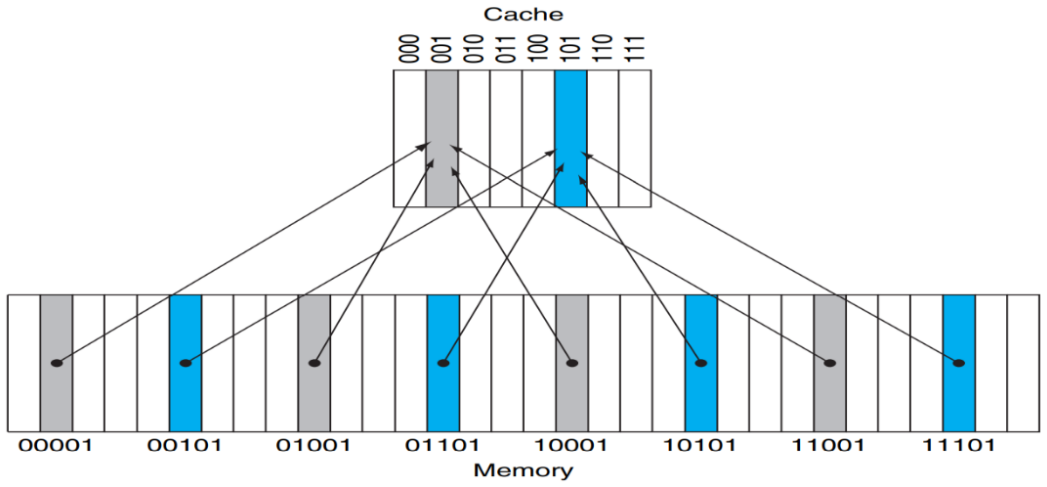
- One simple way to assign a cache block is to use the address of the data item (캐시 블록을 할당하는 간단한 방법 중 하나는 데이터 항목의 주소를 사용하는 것이다.)
    - We call the cache block number “index” (캐시 블록 번호를 “인덱스”라고 부른다.)
    - Index = (Address) modulo (Number of blocks in the cache) (인덱스 = (주소) 모듈로 (캐시의 블록 수))
- In the example, the number of blocks in the cache is 8 (2^3) (이 예제에서 캐시의 블록 수는 8(2^3)이다.)
    - Memory address 1_ten (00001_two) is mapped to cache block 1_ten (001_two); 1 module 8 = 1 (메모리 주소 1_ten(00001_two)은 캐시 블록 1_ten(001_two)에 매핑됨; 1 모듈 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 (메모리 주소

9_ten(01001_two), 17_ten(10001_two) 및 25_ten(11001_two)도 캐시 블록 1_ten(001_two)에 매핑된다; 9 모듈 8 = 1, 17 모듈 8 = 1 및 25 모듈 8 = 1이다.)

    - 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) (메모리 주소 5_ten(00101_2), 13_ten(01101_2), 21_ten(10101_2), 29_ten(11101_2)은 모두 캐시 블록 5_ten(101_2)에 매핑된다.)

- Simply, check the last 3 bits of the memory address; the size of cache (2^3)! (간단히 메모리 주소의 마지막 3비트, 즉 캐시 크기(2^3)를 확인하면 된다!)
    - 00001_two, 01001_two, 10001_two, 11001_two ; the index of these four data items is 001_two (00001_2, 01001_2, 10001_2, 11001_2; 이 네 데이터 항목의 인덱스는 001_2이다.)
    - 00101_two, 01101_two, 10101_two, 11101_two ; the index of these four data items is 101_two (00101_2, 01101_2, 10101_2, 11101_2 ; 이 네 데이터 항목의 인덱스는 101_2이다.)

Direct Mapped Cache (3) - Tag

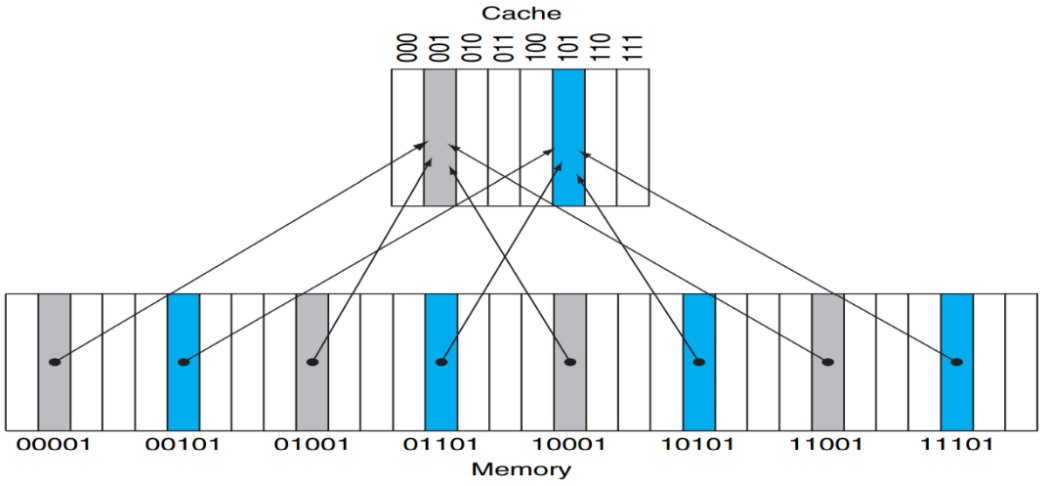


Data whose addresses are 00001, 01001, 10001, and 11001 can be placed in cache block 001; they have the same index! (주소가 00001, 01001, 10001, 11001인 데이터는 캐시 블록 001에 배치할 수 있으며 인덱스가 동일하다.)

To identify one of the four, tags (00, 01, 10, 11) can be used; tags are all different! (네 가지 중 하나를 식별하기 위해 태그(00, 01, 10, 11)를 사용할 수 있으며, 태그는 모두 다르다!)

- A cache block can contain the different data items (캐시 블록에는 다양한 데이터 항목이 포함될 수 있다.)
    - Question: How do we know whether the requested data corresponds to the data in the cache? (질문 : 요청된 데이터가 캐시에 있는 데이터와 일치하는지 어떻게 알 수 있나요?)
    - We can answer the question by adding “tag” to each cache block (각 캐시 블록에 “태그”를 추가하면 이 질문에 답할 수 있다.)
- The address of data is divided and used for two different purposes (데이터의 주소는 두 가지 용도로 나뉘어 사용된다.)
    - The lower bits are used for “index” (to identify its cache block) (아래쪽 비트는 “index”(캐시 블록 식별)에 사용된다.)
    - The upper bits are used for “tag” (to identify the specific data) (상위 비트는 “tag”(특정 데이터를 식별하기 위해)에 사용된다.)
    - (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
      ((
      b4 b3 b2 b1 b0) 분할; 하위 3 비트는 캐시 블럭에 사용되는 “인덱스”이고, 상위 2 비트는 특정 데이터를 나타내는 데 사용되는 “태그”이다.)

Direct Mapped Cache (4) - Valid Bit



- A cache block does not always hold a valid data (캐시 블록이 항상 유효한 데이터를 보유하는 것은 아니다.)

    - The cache block can be empty, if none of the four data exist in the cache (네 가지 데이터 중 하나라도 캐시에 존재하지 않으면 캐시 블록이 비어 있을 수 있다.)
    - When a processor starts up, the cache does not have any data (프로세서가 시작될 때 캐시에 데이터가 없다.)
- We can address the above by adding a “valid bit” to each cache block (각 캐시 블록에 “유효한 비트”를 추가하여 위의 문제를 해결할 수 있다.)
    - 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 (이 비트가 설정되어 있으면 캐시 블록에 유효한 데이터가 있는 것이고, 유효한 데이터가 원하는 데이터인지 아닌지 태그를 확인해야 한다.)
    - If it is not set, you do not have to check the tag, since any information in the cache block is not valid (설정되지 않은 경우 캐시 블록의 모든 정보가 유효하지 않으므로 태그를 확인할 필요가 없다.)

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” (캐시의 각 항목은 “인덱스” + “유효 비트” + “태그” + “데이터”로 구성된다.)
    - There are 8 entries (blocks) in the cache, each of which is identified based on the 3 bit “index” (캐시에는 8개의 항목(블록)이 있으며, 각 항목은 3비트 “인덱스”를 기준으로 식별된다.)
    - If the 1-bit “valid bit” tells whether the information in the entry is valid or not (1비트 “유효 비트”는 항목의 정보가 유효한지 여부를 알려준다.)

- If a data is placed in its corresponding entry, its 2-bit “tag” is set to indicate the specific data (데이터가 해당 항목에 배치되면 2비트 “태그”가 특정 데이터를 나타내도록 설정된다.)

- At first, the cache is empty (it does not hold any data) right after power-on (처음에는 전원을 켠 직후 캐시가 비어있다. (데이터가 저장되어 있지 않음))
  - The valid bit of each entry is set to “N” (각 항목의 유효 비트는 “N”으로 설정된다.)
  - The tag and data are also empty (태그와 데이터도 비어 있다.)

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} mod 8) = 110_{two}
26	11010_{two}	miss (5.6c)	(11010_{two} mod 8) = 010_{two}
22	10110_{two}	hit	(10110_{two} mod 8) = 110_{two}
26	11010_{two}	hit	(11010_{two} mod 8) = 010_{two}
16	10000_{two}	miss (5.6d)	(10000_{two} mod 8) = 000_{two}
3	00011_{two}	miss (5.6e)	(00011_{two} mod 8) = 011_{two}
16	10000_{two}	hit	(10000_{two} mod 8) = 000_{two}
18	10010_{two}	miss (5.6f)	(10010_{two} mod 8) = 010_{two}
16	10000_{two}	hit	(10000_{two} mod 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 (캐시에 일련의 요청(참조)이 주어지는 시나리오 예시)
  - If the referenced data is in the cache, it is “hit” and the data is serviced from the cache (참조된 데이터가 캐시에 있으면 “히트”가 되고 캐시에서 데이터가 서비스 된다.)
  - 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 (참조된 데이터가 캐시에 없으면 “미스”이며 기존 데이터는 메인 메모리에서 복사된 참조된 데이터로 대체된다.)
- Data 22 (10110_two) is requested (데이터 22 (10110_two)가 요청되었다.)
  - It can be placed in the cache block whose “index” is 110_two (the lower 3 bits 10110_two) (”인덱스”가 110_two(하위 3비트 10110_two)인 캐시 블록에 배치될 수 있다.)
  - It is “miss” because the valid bit is set to N (Data 22 does not exist in the cache block) (유효한 비트가 N으로 설정되어 있으므로 “미스”이다. (데이터 22가 캐시 블록에 존재하지 않는다.))
  - Data 22 is brought from the main memory and stored in the cache block whose index is 110_two (데이터 22는 메인 메모리에서 가져와서 인

덱스가 110_two인 캐시 블록에 저장된다.)

- Tag is set to 10_two to indicate data 22 (the upper 2 bits of 10110_two); valid bit is set to Y (태그는 데이터 22(10110_two의 상위 2비트)를 나타내기 위해 10_two로 설정되고, 유효 비트는 Y로 설정된다.)

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 (데이터 26(11010_two)이 요청된다.)
  - It can be placed in the cache block whose “index” is 010_two (the lower 3 bits 11010_two) (“인덱스”가 010_two(하단 3비트 11010_two)인 캐시 블록에 배치할 수 있다.)
  - It is “miss” because the valid bit is set to N (Data 26 does not exist in the cache block) (유효한 비트가 N으로 설정되어 있으므로 “미스”이다 (데이터 26이 캐시 블록에 존재하지 않음).)
  - Data 26 is brought from the main memory and stored in the cache block whose index is 010_two (데이터 26은 메인 메모리에서 가져와 인덱스가 010_two인 캐시 블록에 저장된다.)
  - Tag is set to 11_two to indicate data 26 (the upper 2 bits of 11010_two); valid bit is set to Y (태그는 데이터 26(11010_two의 상위 2비트)을 나타내기 위해 11_two로 설정되고 유효 비트는 Y로 설정된다.)
- Data 22 (10110_two) is requested again (데이터 22(10110_two)가 다시 요청된다.)
  - Go to the index 110_two; first, check its valid bit Y; then, check the tag 10_two ; it is “hit” (인덱스 110_two로 이동하여 먼저 유효한 비트 Y를 확인한 다음 태그 10_two를 확인하면 “히트”이다.)
- Data 26 (11010_two) is requested again (데이터 26(11010_two)이 다시 요청된다.)
  - Go to the index 010_two; first, check its valid bit Y; then, check the tag 11_two ; it is “hit” (인덱스 010_two로 이동하여 먼저 유효한 비트 Y를

확인한 다음, 태그 11_two를 확인한다.)

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} mod 8) = 110_{two}
26	11010_{two}	miss (5.6c)	(11010_{two} mod 8) = 010_{two}
22	10110_{two}	hit	(10110_{two} mod 8) = 110_{two}
26	11010_{two}	hit	(11010_{two} mod 8) = 010_{two}
16	10000_{two}	miss (5.6d)	(10000_{two} mod 8) = 000_{two}
3	00011_{two}	miss (5.6e)	(00011_{two} mod 8) = 011_{two}
16	10000_{two}	hit	(10000_{two} mod 8) = 000_{two}
18	10010_{two}	miss (5.6f)	(10010_{two} mod 8) = 010_{two}
16	10000_{two}	hit	(10000_{two} mod 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})

- 26(11010_two) in cache block (index 010) is replaced by 18(10010_two) (캐시 블록(인덱스 010)의 26(11010_two)은 18(10010_two)로 대체된다.)
  - They are mapped to the same cache block (index 010) (동일한 캐시 블록 (인덱스 010)에 매핑된다.)
  - The recently referenced data item replaces less recently referenced data (최근에 참조된 데이터 항목이 최근에 참조되지 않은 데이터를 대체한다.)
  - This behavior allows a cache to take advantage of "temporal locality" (이 동작을 통해 캐시는 "시간적 지역성"을 활용할 수 있다.)

Access a Direct Mapped Cache (5)

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} mod 8) = 110_{two}
26	11010_{two}	miss (5.6c)	(11010_{two} mod 8) = 010_{two}
22	10110_{two}	hit	(10110_{two} mod 8) = 110_{two}
26	11010_{two}	hit	(11010_{two} mod 8) = 010_{two}
16	10000_{two}	miss (5.6d)	(10000_{two} mod 8) = 000_{two}
3	00011_{two}	miss (5.6e)	(00011_{two} mod 8) = 011_{two}
16	10000_{two}	hit	(10000_{two} mod 8) = 000_{two}
18	10010_{two}	miss (5.6f)	(10010_{two} mod 8) = 010_{two}
16	10000_{two}	hit	(10000_{two} mod 8) = 000_{two}

- In this example, how many cache accesses are there? (이 예제에서는 캐시 엑세스가 몇 개인가?)
  - 9 cache accesses (캐시 엑세스 9회)
- Among them, how many hits and misses are there? (그 중 히트와 미스는 몇 회인가요?)
  - 4 hits and 5 misses (히트 4회, 미스 5회)
- What is the cache hit ratio (miss ratio)? (캐시 적중률(미스 비율)은 얼마인가?)
  - 4/9 (5/9)
  - "Hit ratio" is the fraction of accesses found in the upper level ("적중률"은 상위 수준에서 발견된 엑세스의 비율이다.)
    - The higher the hit ratio, the higher the performance (적중률이 높을수록 성능이 높다.)
    - "Miss ratio" = 1 - "hit ratio" ("미스 비율" = 1 - "적중률")
- An ideal cache achieves the hit ratio of "1" (이상적인 캐시는 적중률 "1"을 달성한다.)