Microprocessor (W12)
- Memory 2 (Cache) -

Dong Min Kim
Department of IoT
Soonchunhyang University
dmk@sch.ac.kr

Contents

03 캐시 기억 장치

❖ 캐시의 사용 목적

- CPU와 주기억 장치의 속도 차이로 인한 CPU 대기 시간을 최소화 시키기 위하여 CPU와 주기억 장치 사이에 설치하는 고속 반도체 기억장치

❖ 캐시의 특징

- 주기억 장치보다 액세스 속도가 높은 칩 사용
- 가격 및 제한된 공간 때문에 용량이 적다.

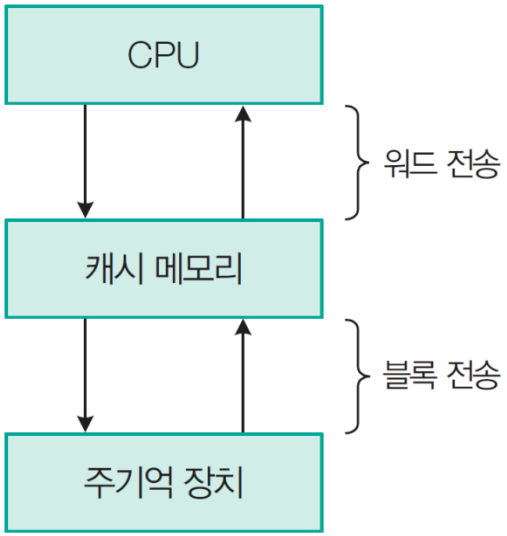
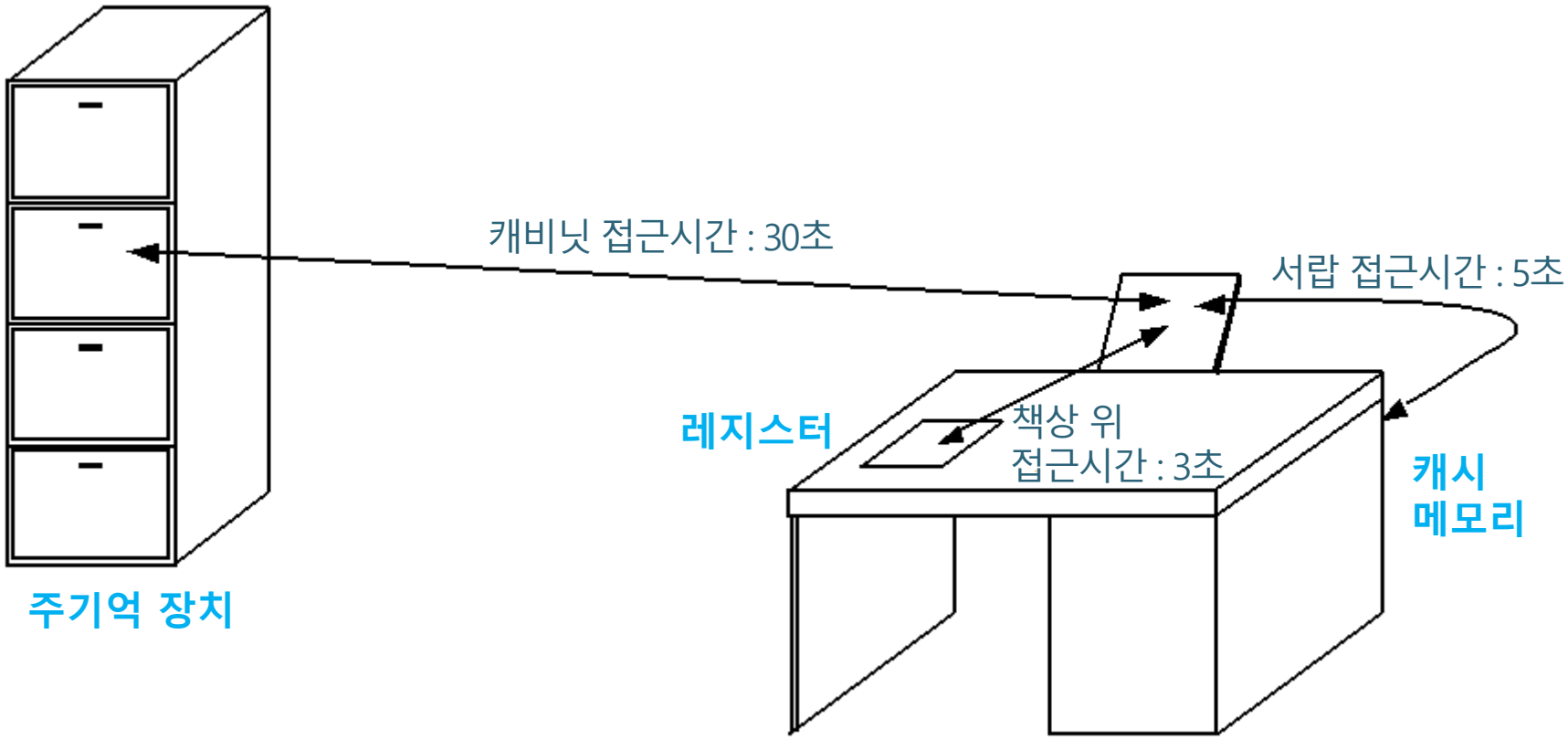


그림 6-20 캐시 위치

03 캐시 기억 장치

❖ **책상 위**(레지스터), **서랍**(캐시), **파일 캐비닛**(주기억 장치)**의 비교**



03 캐시 기억 장치

❖ 캐시의 기본적인 동작 흐름도

- CPU가 기억 장치에서 어떤 정보(명령어 또는 데이터)를 읽으려는 경우 먼저 해당 정보가 캐시에 있는지 검사한다.
- 있다면 해당 정보를 즉시 읽어 들이고, 없다면 해당 정보를 주기억 장치에서 캐시로 적재한 후 읽어 들인다.

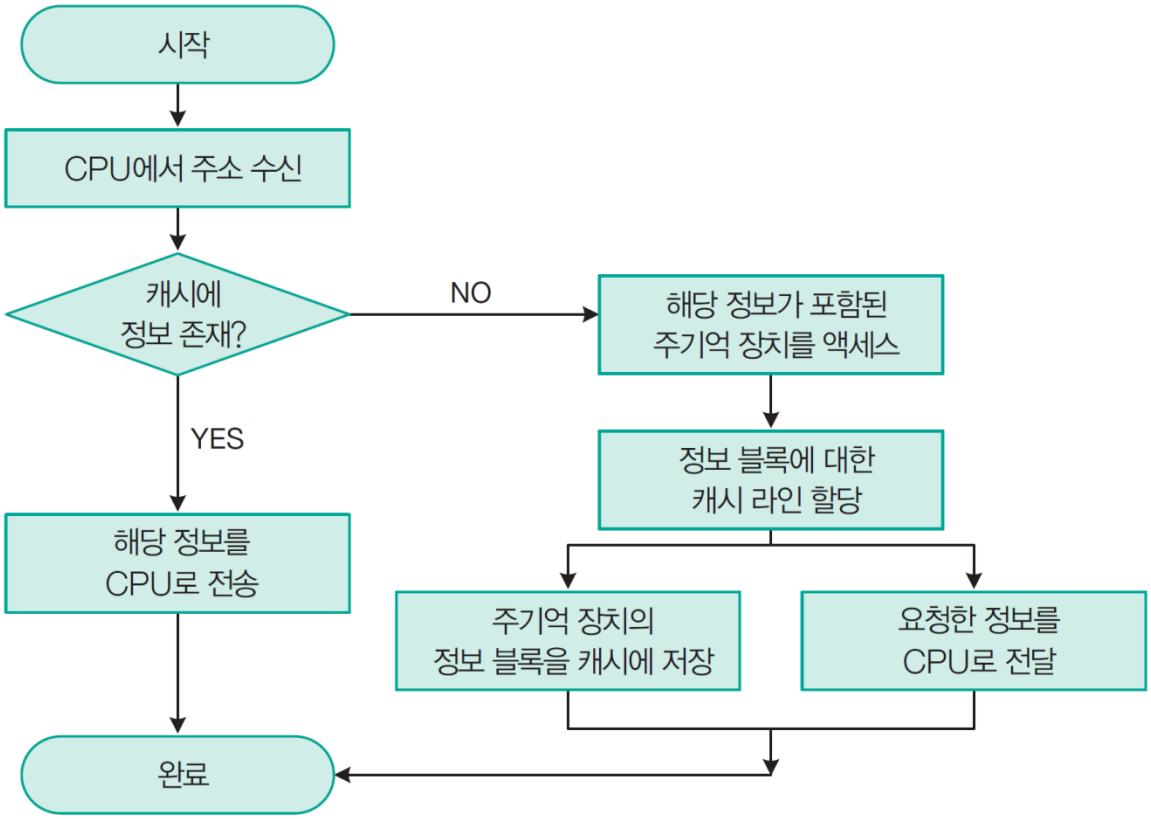


그림 6-21 캐시 읽기 동작 순서

03 캐시 기억 장치

❖ 캐시의 히트율 및 평균 액세스 시간

- 캐시 히트(cache hit) : CPU가 원하는 데이터가 캐시에 있는 상태
- 캐시 미스(cache miss) : CPU가 원하는 데이터가 캐시에 없는 상태. 이 경우에는 주기억 장치로부터 데이터를 읽어온다.
- 히트률(hit ratio) : 캐시에 히트되는 정도(H)

$$H = \frac{\text{캐시에 히트되는 횟수}}{\text{전체 기억장치 액세스 횟수}}$$

- 캐시의 미스율(miss ratio) = (1 - H)
- 평균 기억장치 액세스 시간(T_a)

$$T_a = H \times T_c + (1-H) \times (T_c + T_m)$$
$$\approx H \times T_c + (1-H) \times T_m$$

$$T_c + T_m \approx T_m$$

단, T_c 는 캐시 액세스 시간, T_m 은 주기억 장치 액세스 시간

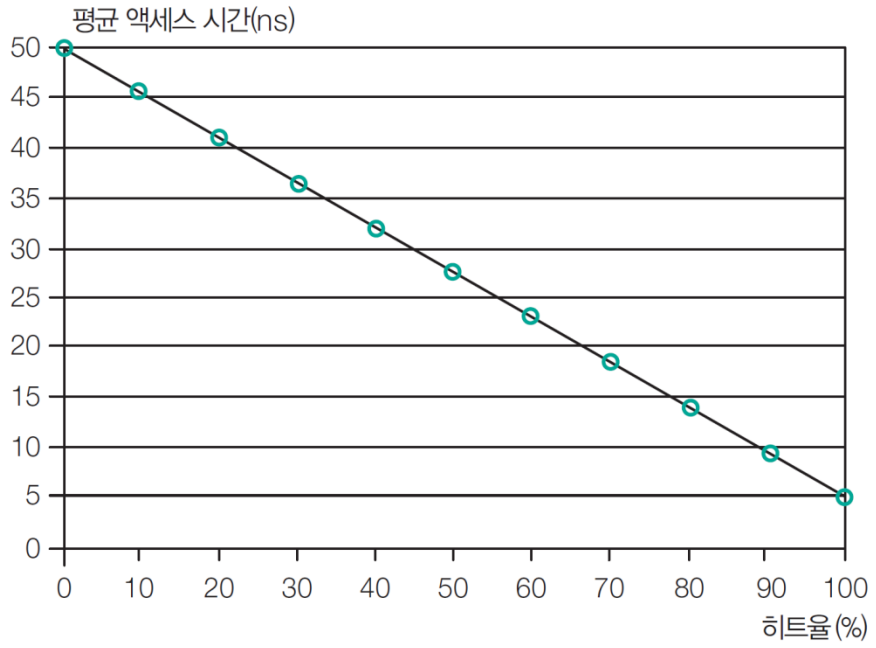
예제 6-5

캐시의 액세스 시간이 T_c=5ns이고, 주기억 장치의 액세스 시간이 T_m=50ns인 기억 장치 시스템에서 캐시 히트율이 0%부터 100%까지 10% 간격으로 변할 때 평균 액세스 시간 T_a를 구하고 이를 그래프로 나타내어라. 결과도 설명하여라. 단, 그래프의 x축은 히트율, y축은 평균 액세스 시간으로 한다.

풀이

히트율(%)	평균 액세스 시간(ns)
0	50.0
10	45.5
20	41.0
30	36.5
40	32.0
50	27.5
60	23.0
70	18.5
80	14.0
90	9.5
100	5.0

캐시의 히트률이 높아질수록 평균 기억장치 액세스시간은 캐시 액세스 시간에 접근



End of Example

❖ 참조 지역성(locality of reference)

- **공간적 지역성**(spatial locality) : 기억장치 내에 인접하여 저장되어 있는 데이터들이 연속적으로 액세스될 가능성이 높다. 예를 들어 표나 배열의 데이터들이 저장되어 있는 기억 장치 영역은 관련 연산이 수행되는 동안 자주 액세스된다.
- **시간적 지역성**(temporal locality) : 최근에 액세스된 프로그램이나 데이터가 가까운 미래에 다시 액세스될 가능성이 높다. 예를 들어 서브루틴이나 루프 프로그램들은 반복적으로 호출되며, 공통 변수들도 자주 액세스된다.

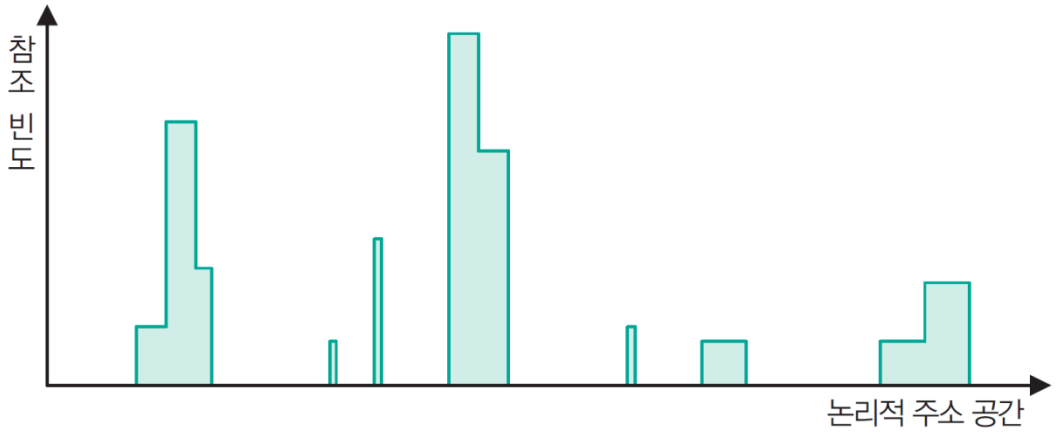


그림 6-22 주소 참조의 전형적인 비균등 분포

03 캐시 기억 장치

❖ 캐시 설계에 있어서의 공통적인 목표

- 캐시의 히트율을 극대화해야 한다.
- 캐시 히트인 경우 캐시에서 정보를 읽어 오는 시간을 최소화해야 한다.
- 캐시 미스인 경우 주기억 장치에서 캐시로 정보를 가져오는 데 걸리는 시간을 최소화해야 한다.
- 캐시의 내용이 변경되었을 경우 주기억 장치에 해당 내용을 갱신하는 데 소요되는 시간을 최소화해야 한다.

❖ 캐시 설계의 설계 요소

표 6-6 캐시의 설계 요소

캐시 용량		쓰기 정책	write-through
사상 방식	직접 사상		write-back
	연관 사상	라인 크기	
	세트-연관 사상	캐시 수	단일 레벨 또는 다중 레벨
교체 알고리즘	Least Recently Used(LRU)		단일 캐시 또는 분리 캐시
	First In First Out(FIFO)		
	Least Frequently Used(LFU)		
	Random		

03 캐시 기억 장치

1 캐시 용량

- 용량이 커질수록 히트율은 높아지지만, 비용이 증가
- 용량이 커질수록 주소 해독 및 정보 인출을 위한 주변 회로가 복잡해지므로 액세스 시간이 다소 더 길어진다.
- CPU 칩 또는 메인보드의 공간에도 제한을 받는다.

2 사상 방식

- **블록**(block) : 주기억 장치와 캐시 사이에 이동되는 정보 단위
- 주기억 장치 용량 = 2^n 워드, 블록 = K개 워드 → 블록의 수 = $2^n/K$ 개
- **라인**(line) : 캐시에서 각 블록이 저장되는 장소. 라인 수 m개, 각 라인에는 워드 K개가 저장된다.
- **태그**(tag) : 라인에 적재된 블록을 구분해주는 정보

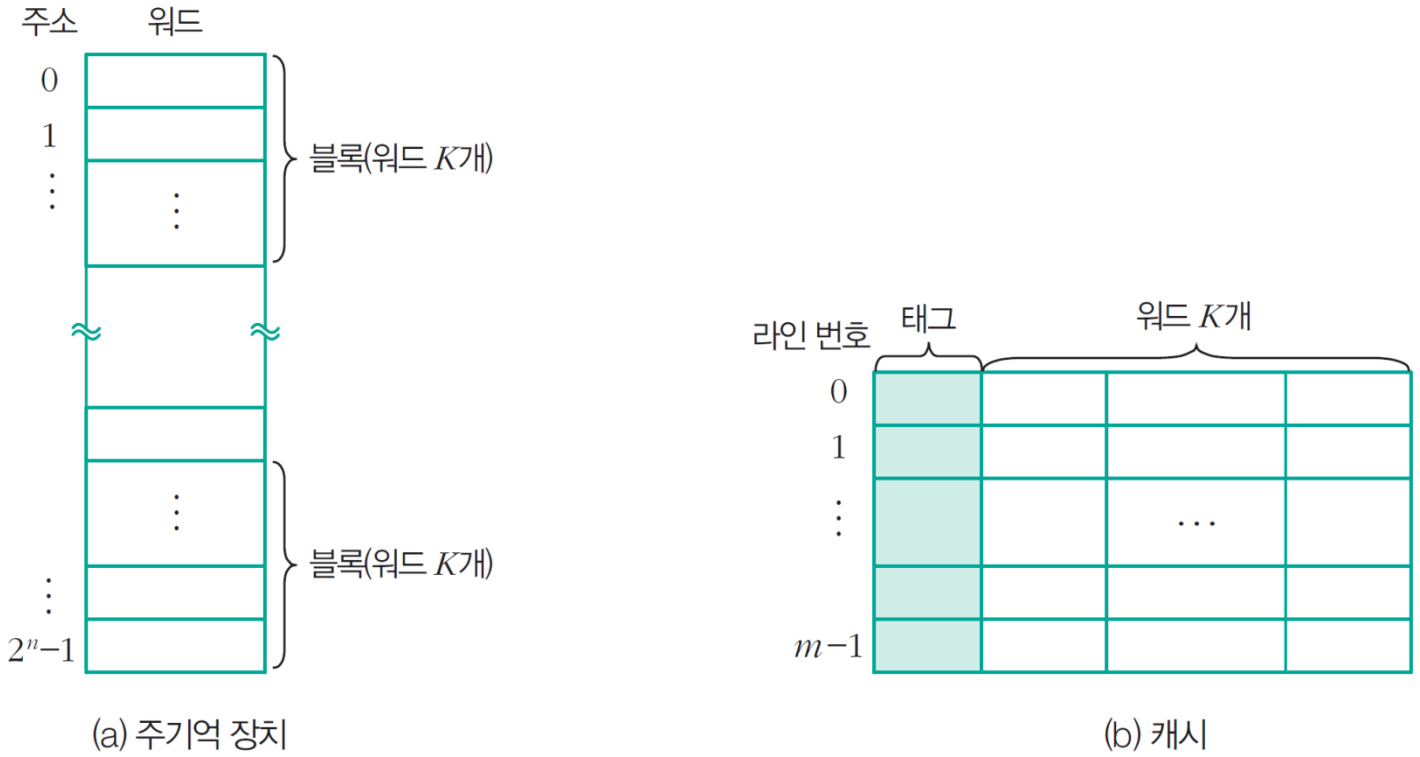


그림 6-23 주기억 장치와 캐시의 구성

03 캐시 기억 장치

❖ 사상 방식(mapping scheme)

사상 방식이란 주기억 장치의 블록이 어느 캐시 라인에 들어갈 것인지 결정하는 방법

- 직접 사상(direct mapping)
- 완전-연관 사상(fully-associative mapping)
- 세트-연관 사상(set-associative mapping)

❖ 사상 방식을 설명하기 위한 모델 예

- 주기억 장치의 용량은 128(=2^7)바이트
- 주기억 장치 주소 = 7비트 (바이트 단위로 주소가 지정, 워드 길이는 1바이트)
- 블록 크기 = 4바이트 → 주기억 장치에는 128/4=32개의 블록이 있다.
- 캐시 크기 = 32 바이트, 캐시 라인의 크기 = 4바이트(블록 크기와 동일)
- 전체 캐시 라인의 수, m = 32/4 = 8개

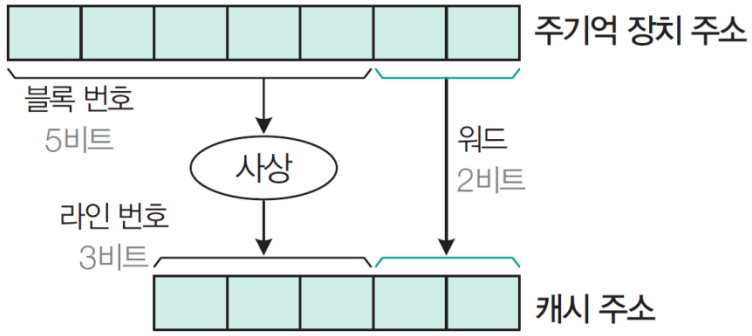


그림 6-24 주기억 장치와 캐시의 주소 사상

❑ 직접 사상 (direct mapping)

- 주기억 장치의 블록들이 지정된 하나의 캐시 라인으로만 적재됨
- 주기억 장치 주소는 필드 3개로 구성된다. 주기억 장치의 주소 지정에는 7비트가 필요하므로 $t+s+w=7$이다.
- 한 블록 내에는 워드 4개가 들어가므로 각 워드를 구별하기 위해 $w=2$비트가 필요하고, 캐시 라인의 수가 8개이므로 각각을 구별하기 위해 $s=3$비트, 그리고 태그 필드는 나머지 $t=2(=7-3-2)$비트가 된다.
- 태그 필드 값은 캐시 라인을 공유하는 주기억 장치 블록들을 서로 구분하는 데 사용된다.

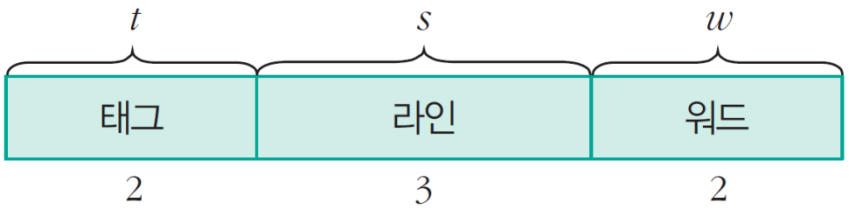


그림 6-25 직접 사상에서 주소 필드의 구성

03 캐시 기억 장치

- 주기억 장치의 블록 j(=0, 1, …, 31)가 적재될 수 있는 캐시 라인 번호는 다음 연산으로 결정된다.

$$i = j \bmod m$$

- [표 6-7]에는 각 캐시 라인을 공유하는 블록들의 번호 4(=2^t=2^2)개가 나열되어 있다. 여기서 블록 번호는 주기억 장치 주소의 상위 5비트에 해당한다.

표 6-7 직접 사상에서 각 캐시 라인을 공유하는 주기억 장치 블록들

캐시 라인	주기억 장치 블록 번호			
0(000)	00000	01000	10000	11000
1(001)	00001	01001	10001	11001
2(010)	00010	01010	10010	11010
3(011)	00011	01011	10011	11011
4(100)	00100	01100	10100	11100
5(101)	00101	01101	10101	11101
6(110)	00110	01110	10110	11110
7(111)	00111	01111	10111	11111

03 캐시 기억 장치

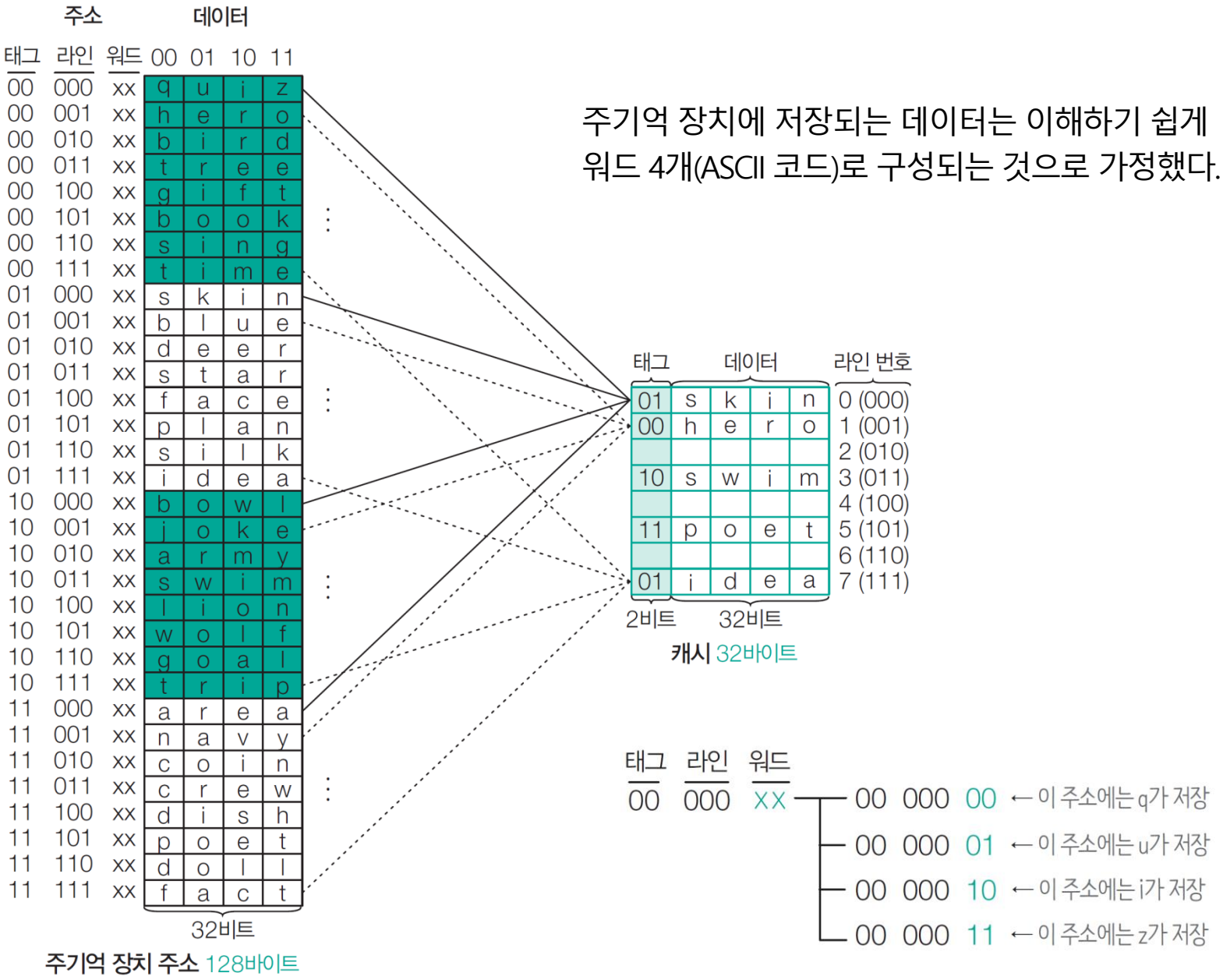


그림 6-26 직접 사상의 예

03 캐시 기억 장치

❖ 직접 사상에서 캐시 내부 구성 및 읽기 동작

❶ 캐시로 주기억 장치 주소 11 101 01이 보내진다(태그 필드=11, 라인 필드=101, 워드 필드=01).

❷ 라인 필드가 101이므로 5번 캐시 라인이 선택된다.

❸ 선택된 5번 라인의 태그 비트 11을 읽어서, ❹ 주기억 장치 주소의 태그 필드인 11과 비교한다.

❺ 두 태그 비트가 일치하므로 캐시가 히트된 것이다.

❻ 다음에는 주소의 워드 필드가 01이므로 poet 중에서 o(1110 1111)가 인출되어 CPU로 전송된다.

❼ 그러나 태그 비트가 일치하지 않으면 캐시가 미스된 것이므로 주소 전체가 주기억 장치로 보내져서 해당 블록을 인출해 온다. 인출된 블록은 지정된 캐시 라인에 저장된다. 그런데 그 라인을 공유하는 다른 블록이 이미 저장되어 있는 상태라면 원래 블록은 지워지고 새로 인출된 블록이 저장된다.

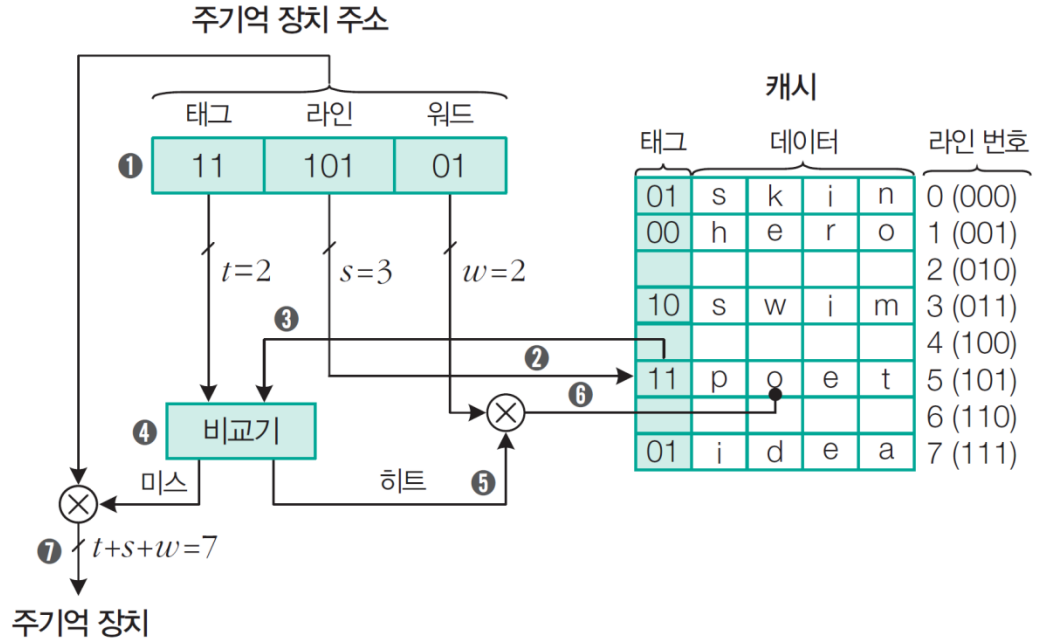


그림 6-27 직접 사상에서 캐시 내부 구성 및 동작

예제 6-6

직접 사상 캐시의 라인들이 [그림 6-26]과 같이 저장되어 있다고 가정하자. 이때 CPU에서 발생한 주소들이 다음과 같은 경우 히트인지 또는 미스인지 구별하여라. 미스인 경우 주기억 장치에서 데이터를 인출하여 해당 라인에 적재된 후의 결과를 설명하라.

(a) 00 001 01 (b) 01 010 11 (c) 10 011 10 (d) 11 111 00

풀이

(a) **히트** : 이 블록은 1번 라인에 적재될 수 있는데, 현재 태그가 00이므로 일치한다. 인출되는 데이터는 hero 중에서 e다.

(b) **미스** : 이 블록이 적재될 수 있는 2번 라인이 비어 2번 라인의 데이터 필드에는 deer가 적재되고, 태그는 01로 세트된다. 최종적으로 인출되는 데이터는 deer 중에서 r이다.

(c) **히트** : 이 블록은 3번 라인에 적재될 수 있는데, 현재의 태그가 10이므로 일치한다. 인출되는 데이터는 swim 중에서 i다.

(d) **미스** : 주소의 태그 비트는 11이고, 이 블록이 적재될 수 있는 7번 라인의 현재 태그는 01이므로 일치하지 않는다. 따라서 7번 라인의 데이터 필드는 fact로 대체되고, 태그는 11로 변경된다. 최종적으로 인출되는 데이터는 fact 중에서 f다.

End of Example

❖ 직접 사상 캐시의 장단점

장점	• 하드웨어가 간단하고, 구현 비용이 적게 든다.
단점	• 각 주기억 장치 블록이 적재될 수 있는 캐시 라인이 한 개뿐이기 때문에, 그 라인을 공유하는 다른 블록이 적재되는 경우에는 반복적으로 미스되어 히트율이 떨어짐(swap-out)

❑ 완전-연관 사상(fully-associative mapping)

- 주기억 장치 블록이 캐시의 어떤 라인으로든 적재 가능
- 태그 필드 = 주기억 장치 블록 번호

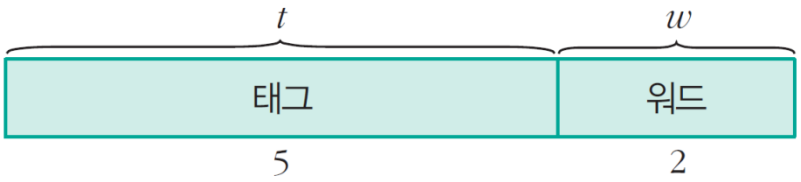
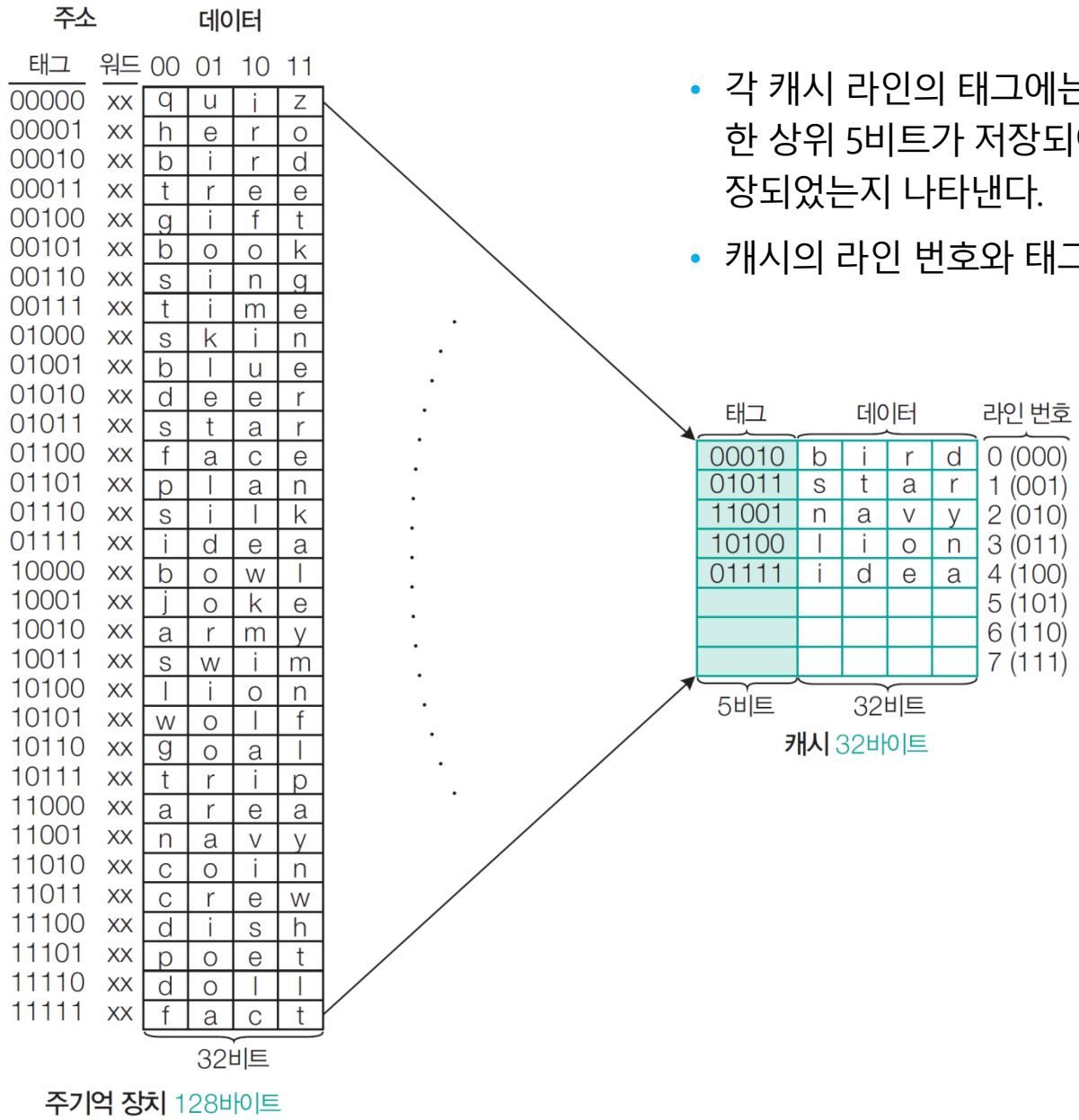


그림 6-28 완전-연관 사상에서 주소 필드의 구성

03 캐시 기억 장치



- 각 캐시 라인의 태그에는 주소의 워드 필드(2비트)를 제외한 상위 5비트가 저장되어 주기억 장치의 어느 블록이 저장되었는지 나타낸다.
- 캐시의 라인 번호와 태그 필드는 관련이 없다.

그림 6-29 완전-연관 사상의 예

❖ 완전-연관 사상에서 캐시 내부 구성 및 읽기 동작

❶ 캐시로 주기억 장치 주소 11001 11이 보내진다. 태그 필드는 11001이고, 워드 필드는 11이다.

❷ 캐시의 모든 라인의 태그들과 주기억 장치 주소의 태그 필드 내용을 비교한다.

❸ 2번 캐시 라인의 11001과 주소의 태그 필드 11001이 일치하므로 캐시가 히트된 것이다.

❹ 다음에는 주소의 워드 필드가 11이므로 navy 중에서 y(0111 1001)가 인출되어 CPU로 전송된다.

❺ 그러나 태그 비트가 일치하지 않으면 캐시가 미스된 것이므로 주소 전체가 주기억 장치로 보내져서 해당 블록을 인출해 온다. 인출된 블록은 5번 캐시 라인에 저장된다.

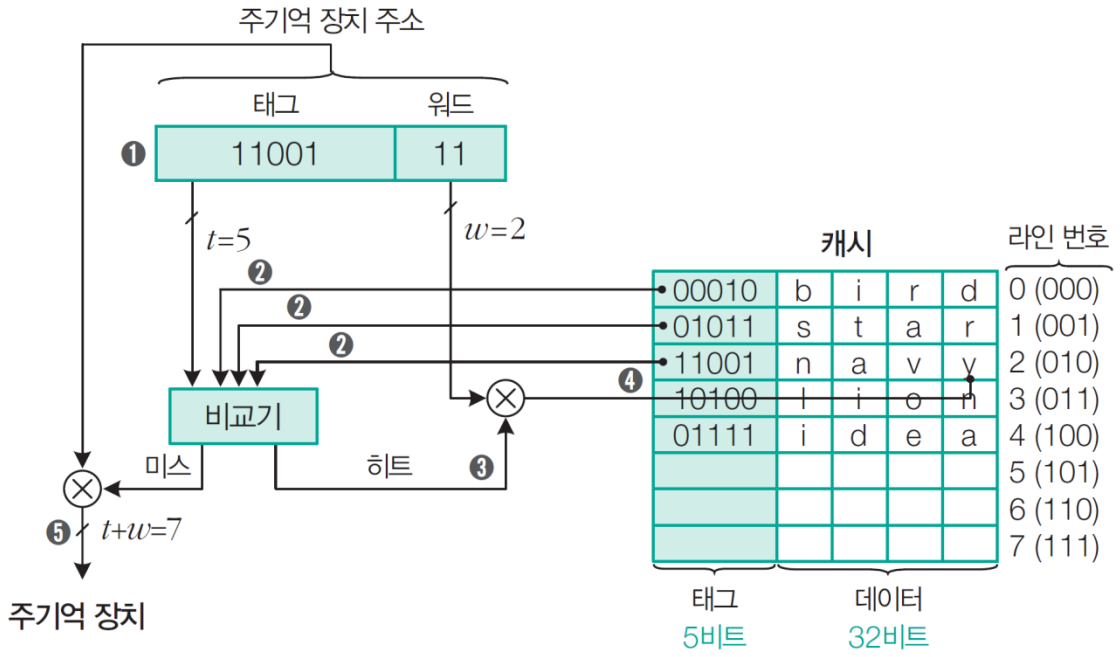


그림 6-30 완전-연관 사상에서 캐시 내부 구성 및 동작

예제 6-7

완전-연관 사상 캐시의 라인들이 [그림 6-29]와 같이 저장되어 있다고 가정하자. 이때 CPU에서 발생한 주소가 다음과 같은 경우 히트인지 미스인지 구별하여라. 미스인 경우 주기억 장치에서 데이터를 인출하여 해당 라인에 적재된 후의 결과에 대해 설명하라.

(a) 01011 01 (b) 10010 11
(c) 00010 10 (d) 11100 00

풀이

(a) **히트** : 1번 라인에 적재되어 있으며, 인출 데이터는 star에서 t다.

(b) **미스** : 비어 있는 첫 번째 라인인 5번 라인에 적재되므로 데이터 필드에는 army가 적재되고, 태그는 10010으로 변경된다. 최종 인출 데이터는 army에서 y다.

(c) **히트** : 0번 라인에 적재되며, 인출 데이터는 bird에서 r이다.

(d) **미스** : 라인 순으로 6번 라인에 적재되므로 데이터 필드에는 dish가 적재되고, 태그는 11100으로 변경된다. 최종 인출 데이터는 dish에서 d다.

End of Example

❑ 세트-연관 사상(set-associative mapping)

- 직접 사상과 완전-연관 사상의 조합
- 주기억 장치 블록 그룹이 하나의 캐시 세트를 공유하며, 그 세트에는 두 개 이상의 라인들이 적재될 수 있음
- 전체 캐시 라인(m)은 v개의 세트들로 나누어지며, 각 세트들은 k개의 라인들로 구성(k-way)
- 주기억 장치 블록(j)이 적재될 수 있는 캐시 세트의 번호 i :

$$m = v \times k$$
$$i = j \bmod v$$

- 앞의 예에서 캐시 라인의 수는 m=8이다. 세트당 캐시 라인의 수가 k=2라면 세트 수는 v=8/2=4개다. 따라서 캐시는 세트 4개로 구성되고, 각 세트에 캐시 라인이 2개 있다.

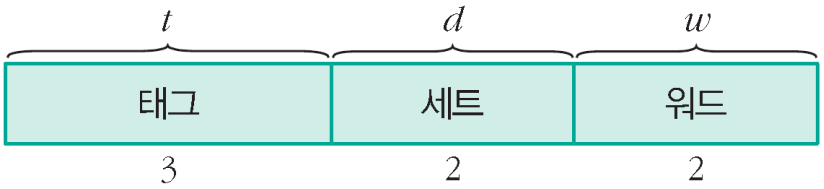


그림 6-31 세트-연관 사상에서 주소 필드의 구성

03 캐시 기억 장치

- 주기억 장치의 블록 j(=0, 1, ..., 31)가 적재될 수 있는 세트 번호는 j mod 4로 결정된다.
- 세트-연관 사상 방식에서 세트 4개, 세트당 캐시 라인 2개에 들어갈 수 있는 주기억 장치 블록 32개는 다음과 같다.

표 6-8 세트-연관 사상에서 각 세트를 공유하는 주기억 장치 블록

세트 번호	주기억 장치 블록 번호							
0(00)	00000	00100	01000	01100	10000	10100	11000	11100
1(01)	00001	00101	01001	01101	10001	10101	11001	11101
2(10)	00010	00110	01010	01110	10010	10110	11010	11110
3(11)	00011	00111	01011	01111	10011	10111	11011	11111

03 캐시 기억 장치

- 2-way 세트-연관 사상 방식

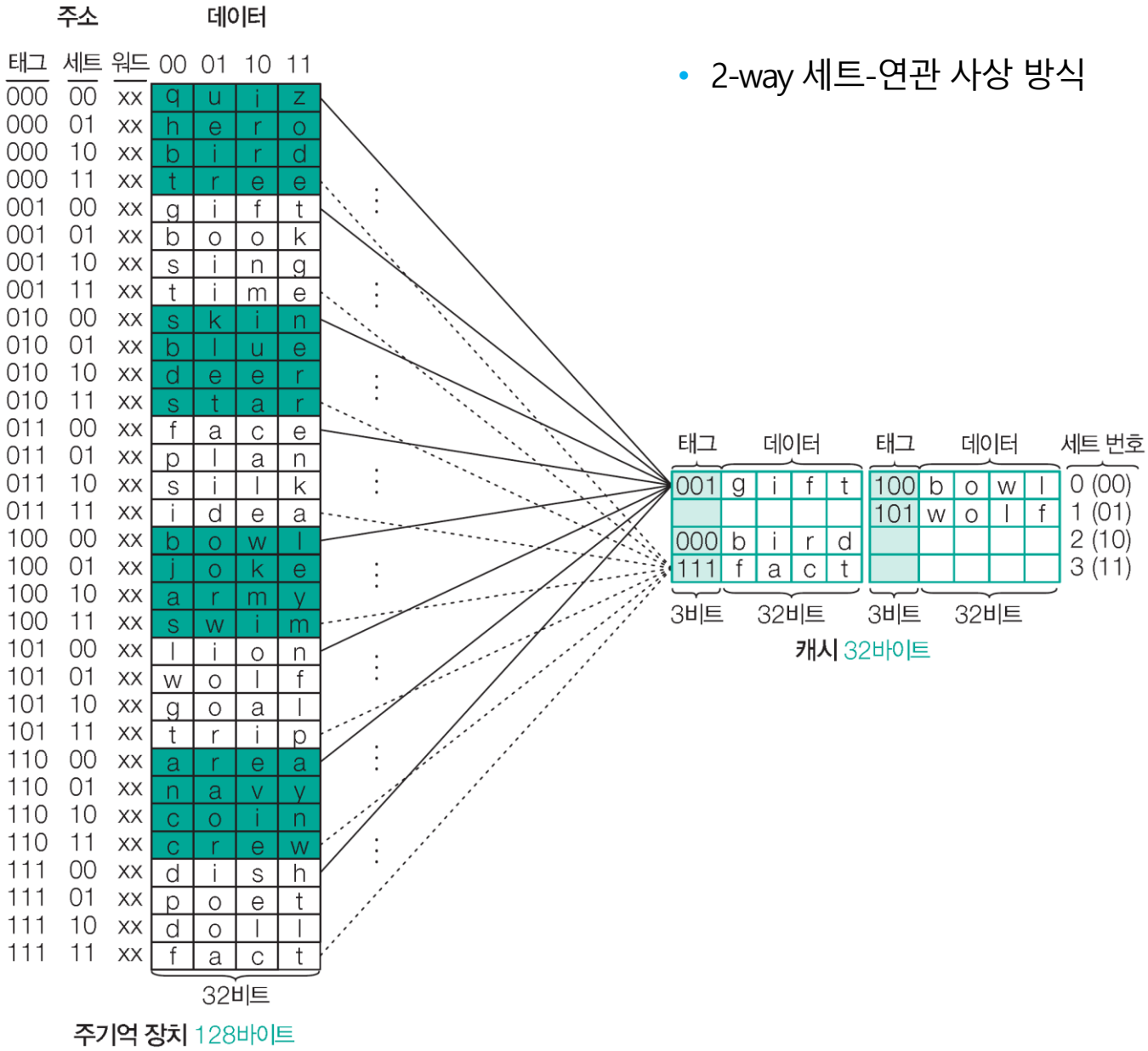


그림 6-32 세트-연관 사상의 예

❖ 세트-연관 사상에서 캐시 내부 구성 및 읽기 동작

❶ 주기억 장치 주소 001 00 10이 캐시로 보내진다(태그 필드=001, 세트 필드=00, 워드 필드= 10).

❷ 00 세트 번호를 이용해 캐시의 0번 세트를 선택한다.

❸ 0 번 세트 라인의 태그에 주기억 장치 주소의 태그 비트0 01과 일치하는 것이 있는지 검사한다.

❹ 0번 세트 내 첫 번째 라인의 태그 비트가 001이므로 히트되었다.

❺ 다음에는 주소의 워드 필드가 10이므로 gift 중에서 f(1110 0110)가 인출되어 CPU로 전송된다.

❻ 그러나 태그 비트가 일치하지 않으면 캐시가 미스된 것이므로 주소 전체가 주기억 장치로 보내져서 해당 블록을 인출해 온다. 적절한 교체 알고리즘에 의하여 2개 중 하나를 선택하여 그 라인에 새로운 블록을 적재해야 한다.

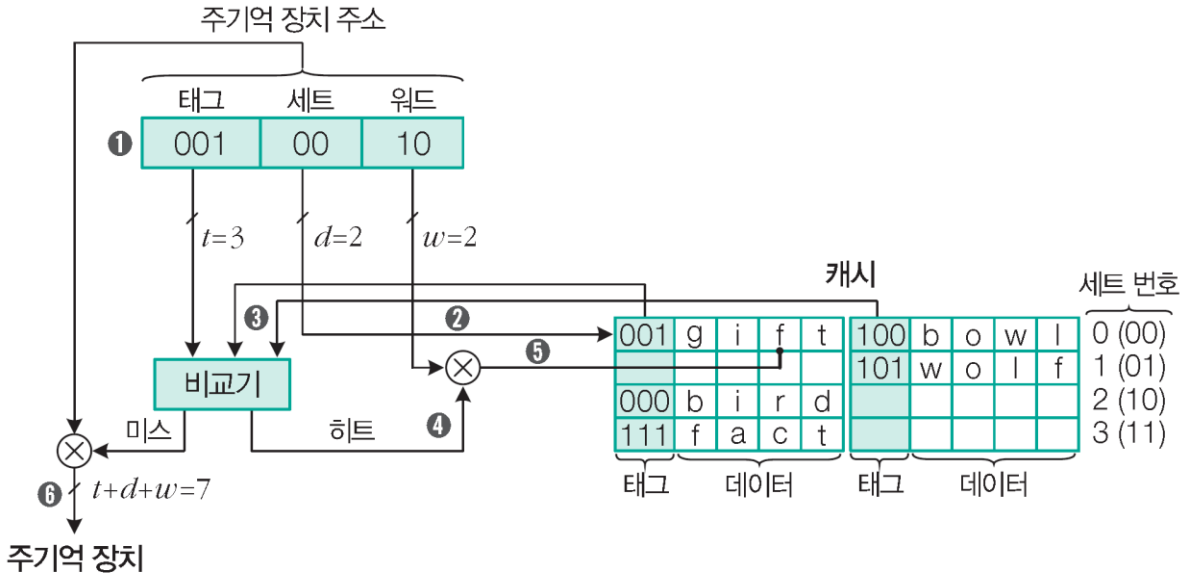


그림 6-33 세트-연관 사상에서 캐시 내부 구성 및 동작

예제 6-8

세트-연관 사상 캐시의 라인들이 [그림 6-32]와 같이 저장되어 있다고 가정하자. 이때 CPU에서 발생한 주소들이 다음과 같은 경우 히트인지 또는 미스인지 구별하여라. 그리고 미스인 경우 주기억 장치에서 데이터를 인출하여 해당 라인에 적재된 후 결과를 설명하여라.

(a) 100 00 01 (b) 011 10 11

(c) 111 11 10 (d) 010 00 00

풀이

(a) **히트** : 0번 세트의 두 번째 라인에 적재되어 있으며, 인출 데이터는bowl에서 o이다.

(b) **미스** : 적재될 수 있는 2번 세트의 첫 번째 라인의 태그 000과 주소의 태그 011이 일치하지 않는다. 따라서 빈 두 번째 라인에 silk가 적재되고, 태그는 011로 변경된다. 최종적으로 인출 데이터는 silk에서 k다.

(c) **히트** : 3번 세트의 첫 번째 라인에 적재되어 있으며, 인출 데이터는fact에서 c다.

(d) **미스** : 이 블록이 적재될 수 있는 0번 세트의 두 라인의 태그(001, 100)가 주소의 태그 010과 일치하지 않는다. 편의상 새로운 블록이 두 번째 라인에 적재된다고 가정하면 skin이 적재되고, 태그는 010으로 변경된다. 최종 인출 데이터는 skin에서 s다.

End of Example

❖ 사상 방식의 비교

표 6-9 사상 방식의 비교

사상 기법	단순성	태그 연관 검색	캐시 효율	교체 기법
직접	단순	없음	낮음	불필요
완전-연관	복잡	연관	높음	필요
세트-연관	중간	중간	중간	필요

3 교체 알고리즘

- 세트-연관 사상에서 주기억 장치로부터 새로운 블록이 캐시로 적재될 때, 만약 세트 내 모든 라인들이 다른 블록들로 채워져 있다면, 그들 중의 하나를 선택하여 새로운 블록으로 교체
- 교체 알고리즘 : 캐시 히트율을 극대화할 수 있도록 교체할 블록을 선택하기 위한 알고리즘
  - **LRU**(Least Recently Used) : 사용되지 않은 채로 가장 오래 있었던 블록을 교체하는 방식
  - **FIFO**(First-In-First-Out) 알고리즘 : 캐시에 적재된 지 가장 오래된 블록을 교체하는 방식
  - **LFU**(Least Frequently Used) 알고리즘 : 참조되었던 횟수가 가장 적은 블록을 교체하는 방식
  - **Random** : 사용 횟수를 고려하지 않고 후보 캐시 라인 중 임의로 선택하여 교체하는 방식

예제 6-9

FIFO 교체 알고리즘을 사용하는 세트-연관 사상 캐시로 다음 블록을 연속으로 액세스하는 경우 각 캐시 라인에 적재되는 블록을 표시하고 히트율을 구하여라. 단, 각 세트의 라인 수는 (a) 2개, (b) 4개라고 가정한다.

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

풀이

(a) 캐시 라인 수가 2개인 경우 : 히트율=6/20

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	1	1	0	0	0	4	4	3	3	3	3	1	1	1	1	7	7	7
	0	0	2	2	3	3	3	2	0	0	0	2	2	2	0	0	0	0	1
						hit				hit	hit			hit		hit		hit	

(b) 캐시 라인 수가 4개인 경우 : 히트율=10/20

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	7	7	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2
	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	7	7	7
		1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
			2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
				hit		hit		hit	hit		hit	hit			hit	hit		hit	hit

End of Example

예제 6-10

LRU 교체 알고리즘을 사용하는 세트-연관 사상 캐시로 다음 블록을 연속으로 액세스하는 경우 각 캐시 라인에 적재되는 블록을 표시하고 히트율을 구하여라. 단, 각 세트의 라인 수는 (a) 2개, (b) 4개라고 가정한다.

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

풀이

(a) 캐시 라인 수가 2개인 경우 : 히트율=3/20

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
	7	7	1	1	0	0	0	0	2	2	0	0	2	2	2	2	1	1	0	0
		0	0	2	2	3	3	4	4	3	3	3	3	1	1	0	0	7	7	1
							hit					hit			hit					

(b) 캐시 라인 수가 4개인 경우 : 히트율=12/20

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	3	7	7	7
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			1	1	1	1	1	4	4	4	4	4	4	1	1	1	1	1	1	1
				2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
					hit		hit		hit	hit	hit	hit	hit		hit	hit	hit		hit	hit

End of Example

4 쓰기 정책

- 캐시의 블록이 변경되었을 때 그 내용을 주기억 장치에 갱신하는 시기와 방법의 결정

❖ Write-through

- 모든 쓰기 동작들이 캐시로 뿐만 아니라 주기억 장치로도 동시에 수행되는 방식

❖ Write-back

- 캐시에서 데이터가 변경되어도 주기억 장치에는 갱신되지 않는 방식

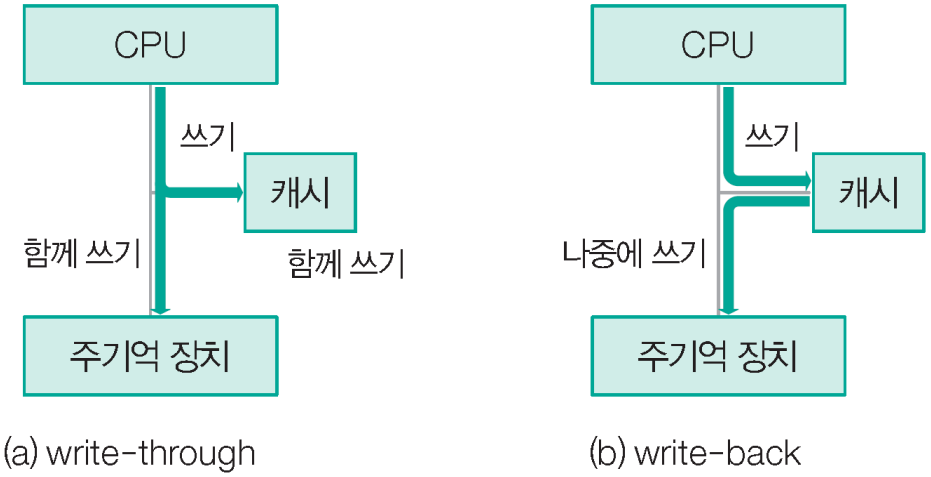


그림 6-34 쓰기 정책

03 캐시 기억 장치

❖ 각 방식의 장단점

Write-through

장점	• 캐시에 적재된 블록의 내용과 주기억 장치에 있는 그 블록의 내용이 항상 같다.
단점	• 모든 쓰기 동작이 주기억 장치 쓰기를 포함하므로, 쓰기 시간이 길어진다.

Write-back

장점	• 기억장치에 대한 쓰기 동작의 횟수가 최소화되고, 쓰기 시간이 짧아진다.
단점	• 캐시의 내용과 주기억 장치의 해당 내용이 서로 다르다.

⇨ 블록을 교체할 때는 캐시의 상태를 확인하여 주기억 장치에 갱신하는 동작이 선행되어야 하며, 그를 위하여 각 캐시 라인이 상태 비트(status bit)를 가지고 있어야 한다.

예제 6-11

주기억 장치 액세스 시간이 50ns, 캐시 액세스 시간이 5ns인 시스템이 있다. 전체 액세스 요구 중에서 70%는 읽기 동작이고, 30%는 쓰기 동작이었으며, 캐시 히트율은 90%였다. 캐시 쓰기 정책이 (a) write-through일 때와 (b) write-back일 때 평균 기억 장치 액세스 시간을 각각 구하여라. 단, write-back에서 미스가 발생한 경우 새로운 블록을 적재하기 위해 교체할 라인들의 20%는 이미 수정된 상태에 있었다고 가정한다.

풀이

(a) Write-through 정책인 경우

- 읽기동작의 평균시간 = 0.9×5ns+0.1×50ns = 9.5ns
- 쓰기동작의 평균시간 = 50ns (모든 쓰기 동작은 주기억 장치를 액세스)
- 평균 기억장치 액세스 시간 = 0.7×9.5ns+0.3×50ns = 21.65ns

(b) Write-back 정책인 경우 읽기동작과 쓰기동작에 같은 시간이 걸리므로,
평균 기억장치 액세스 시간 = 0.9×5ns+0.1×(50ns+0.2×50ns) = 10.5ns

End of Example

5 라인 크기

❖ 블록 크기에 따른 특성

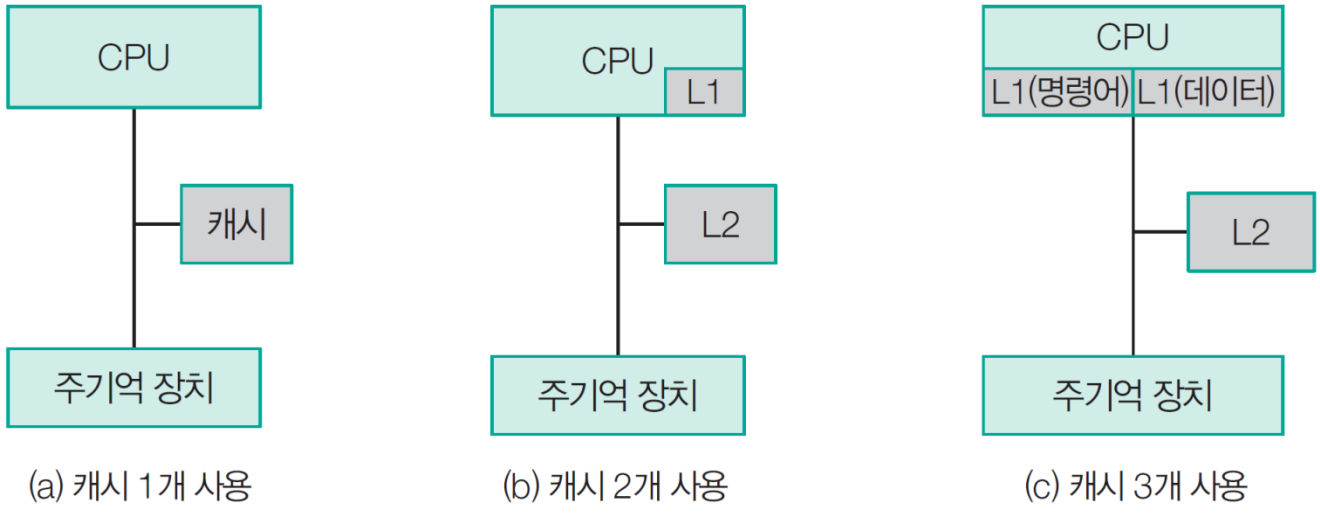
- 캐시 용량은 한정되어 있기 때문에 블록 크기가 커질수록 캐시에 들어올 수 있는 블록의 수는 줄어든다. 새로운 블록을 읽어 올 때마다 원래 캐시 내용은 교체되기 때문에 블록 수가 적으면 인출된 직후에 다른 블록과 다시 교체된다.
- 블록 크기가 커질수록 원하는 워드에서 멀리 떨어져 있는 워드들도 같이 읽혀 오며, 그들은 가까운 미래에 사용될 가능성은 낮다.

⇨ 대략적으로 블록 크기가 8~32바이트 정도가 최적에 가까운 것으로 알려져 있다.

6 캐시 수

❑ 계층적 캐시

- **온-칩 캐시**(on-chip cache) : 캐시 액세스 시간을 단축시키기 위하여 CPU 칩 내에 포함시킨 캐시 (그림의 L1)
- **계층적 캐시**(hierarchical cache) : 온-칩 캐시를 1차(L1) 캐시로 사용하고, CPU 외부에 더 큰 용량의 2차(L2) 캐시를 설치하는 방식
- **분리 캐시**(split cache) : 캐시를 명령어 캐시와 데이터 캐시로 분리



(a) 캐시 1개 사용 (b) 캐시 2개 사용 (c) 캐시 3개 사용

그림 6-35 다양한 계층적 캐시 구조

❖ 계층적 캐시에서의 히트율

- L2의 용량이 L1보다 크며, L1의 모든 내용이 L2에도 존재
- 먼저 L1을 검사하고, 만약 원하는 정보가 L1에 없다면 L2를 검사하며, L2에도 없는 경우에는 주기억 장치를 액세스
- L1은 속도가 빠르지만, 용량이 작기 때문에 L2 보다 히트율은 더 낮다.
- 평균 기억 장치 액세스 시간(T_a)

$$T_a = H_1 \times T_{L1} + (H_2 - H_1) \times T_{L2} + (1 - H_2) T_m$$

L1 캐시의 액세스 시간과 히트율 : T_{L1}, H_1
L2 캐시의 액세스 시간과 히트율 : T_{L2}, H_2
주기억 장치의 액세스 시간 : T_m
H_2 는 전체 기억 장치 액세스에 대한 L2의 히트율을 의미한다.

예제 6-12 L1 캐시, L2 캐시, 주기억 장치의 액세스 시간이 각각 2ns, 5ns, 50ns이고, H_1=0.7, H_2=0.9라고 할 때, 평균 기억 장치 액세스 시간을 구하여라.

풀이

$$T_a = 0.7 \times 2ns + (0.9 - 0.7) \times 5ns + (1 - 0.9) \times 50ns$$
$$= 1.4ns + 1ns + 5ns = 7.4ns$$

End of Example

❑ 통합 캐시와 분리 캐시

❖ 통합 캐시(unified cache)

- 온-칩 캐시가 처음 출현했을 때, 대부분은 명령어와 데이터를 하나의 캐시에 저장

❖ 분리 캐시(split cache)

- 캐시를 명령어 캐시와 데이터 캐시로 분리
- 명령어 인출 유니트와 실행 유니트 간의 캐시 액세스 충돌 제거
- 고성능 파이프 라인 프로세서들에서 사용

Summary

- 캐시 기억 장치의 세 가지 매핑 방법과 교체 알고리즘 이해