컴퓨터 구조

기억장치 (2)

(제 10주 차)

서울사이버대학교

오 창 환

학습 목표

- 캐쉬 기억장치, 캐쉬 인출 방식 등을 설명할 수 있다.
- 캐쉬 사상 방식을 설명할 수 있다.
- 캐쉬 교체 알고리즘, 캐쉬 쓰기 정책 등을 설명할 수 있다.

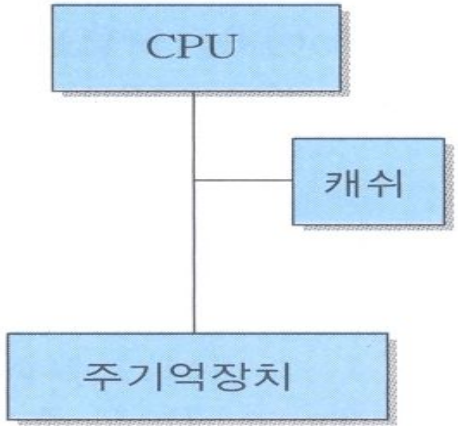
학습 내용

- 캐쉬 기억장치, 캐쉬 인출 방식
- 캐쉬 사상 방식
- 캐쉬 교체 알고리즘, 캐쉬 쓰기 정책

캐쉬 기억장치 (1)

- 캐쉬는 CPU 칩과 인접한 곳에 위치하며, CPU 칩 내부에 포함되기도 함.
- 또한 캐쉬로는 고속 기억장치 칩들을 사용하기 때문에, 캐쉬 액세스 시간이 주기억장치 액세스 시간보다 더 짧지만, 칩의 가격이 그만큼 더 높아지고 공간도 제한되기 때문에 캐쉬의 용량은 일반적으로 주기억장치의 용량에 비해 훨씬 더 작음.
- CPU가 기억장치로부터 어떤 데이터를 읽으려고 할 때는 먼저 그 데이터가 캐쉬에 있는 지를 검사하는데, 만약 있다면 데이터는 즉시 CPU로 전달되어 액세스 시간이 크게 단축됨.

  그러나, 만약 없다면 그 데이터는 주기억장치로부터 인출되어야 하기 때문에 더 오랜 시간이 걸림.



• 캐쉬의 위치

캐쉬 기억장치 (2)

- CPU가 원하는 데이터가 이미 캐쉬에 있는 상태를 캐쉬 적중(cache hit)이라고 하고, 반대로 CPU가 원하는 데이터가 캐쉬에 없는 상태를 캐쉬 미스(cache miss)라고 함. 캐쉬에 적중되는 정도를 나타내는 적중률(hit rate) H는 아래와 같이 정의됨.

$$H = \frac{\text{캐쉬에 적중되는 횟수}}{\text{전체 기억장치 액세스 횟수}}$$

- 캐쉬 적중률은 원하는 데이터가 캐쉬에 있을 확률이라고 말할 수 있으며, 캐쉬에 없을 확률인 미스율(miss ratio)은 (1-H)가 됨.
  캐쉬가 사용된 시스템에서 평균 기억장치 액세스 시간, T_a는 아래 식과 같음.

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

  여기에서 T_c는 캐쉬 액세스 시간이고, T_m은 주기억장치 액세스 시간을 나타냄.

캐쉬 기억장치 (3)

- 캐쉬 적중률이 높아질수록 평균 액세스 시간은 캐쉬의 액세스 시간에 접근하게 되는데 캐쉬 적중률은 프로그램과 데이터의 지역성(locality)에 크게 의존함.
  지역성이란 CPU가 주기억장치의 특정 부분에 위치한 프로그램 코드나 데이터를 빈번히 혹은 집중적으로 액세스 하는 현상을 말하며 그 특성에 따라 다음과 같이 분류됨.
  * 시간적 지역성 : 최근에 액세스된 프로그램 코드나 데이터가 가까운 미래에 다시 액세스될 가능성이 높음.
  * 공간적 지역성 : 기억장치 내에 서로 인접하여 저장되어 있는 데이터들이 연속적으로 액세스될 가능성이 높음.
  * 순차적 지역성 : 분기(branch)가 발생하지 않는 한, 명령어들은 기억장치에 저장된 순서대로 인출되어 실행됨.

캐쉬 기억장치 (4)

- 캐쉬 설계에 있어서의 공통적인 목표는 아래와 같음.
  * 캐쉬 적중률의 극대화 : CPU가 원하는 정보가 캐쉬에 있을 확률을 높여야 함.
  * 캐쉬 액세스 시간의 최소화 : 캐쉬 적중 시에 캐쉬로부터 CPU로 정보를 인출해오는데 걸리는 시간을 가능한 한 단축시켜야 함.
  * 캐쉬 실패에 따른 지연시간의 최소화 : 캐쉬 미스가 발생한 경우에 주기억장치로부터 캐쉬로 정보를 읽어오는 데 걸리는 시간을 최소화해야 함.
  * 주기억장치와 캐쉬 간의 데이터 일관성 유지 및 그에 따른 오버헤드의 최소화 : CPU가 캐쉬의 내용을 변경하였을 때, 주기억장치에 그 내용을 갱신하는 절차 때문에 발생하는 지연시간을 최소화 해야 함.
- 캐쉬의 크기, 즉 용량이 커질수록 적중률이 높아지지만, 비용도 동일한 비율로 상승하므로 캐쉬의 크기는 비용과의 상호조정을 통하여 적절하게 결정되어야 함. 또한 캐쉬의 용량이 커질수록 주소 해독 및 정보 인출을 위한 주변 회로가 더 복잡해지기 때문에 액세스 시간이 다소 더 길어짐.

캐쉬 인출 방식(1)

- 주기억장치로부터 캐쉬로 정보를 인출해오는 방식도 캐쉬 적중률에 많은 영향을 주는데 이러한 방식에는 요구 인출(demand fetch) 방식과 선인출(prefetch) 방식이 있음.

  * 요구 인출 방식 : 필요한 정보만 인출해 오는 방법

  * 선인출 방식 : 필요한 정보 외에 앞으로 필요할 것으로 예측되는 정보도 미리 인출해오는 방법임.

    즉 CPU가 원하는 정보를 인출할 때 그 정보와 근접한 위치에 있는 정보들을 함께 인출하여 캐쉬에 적재함.

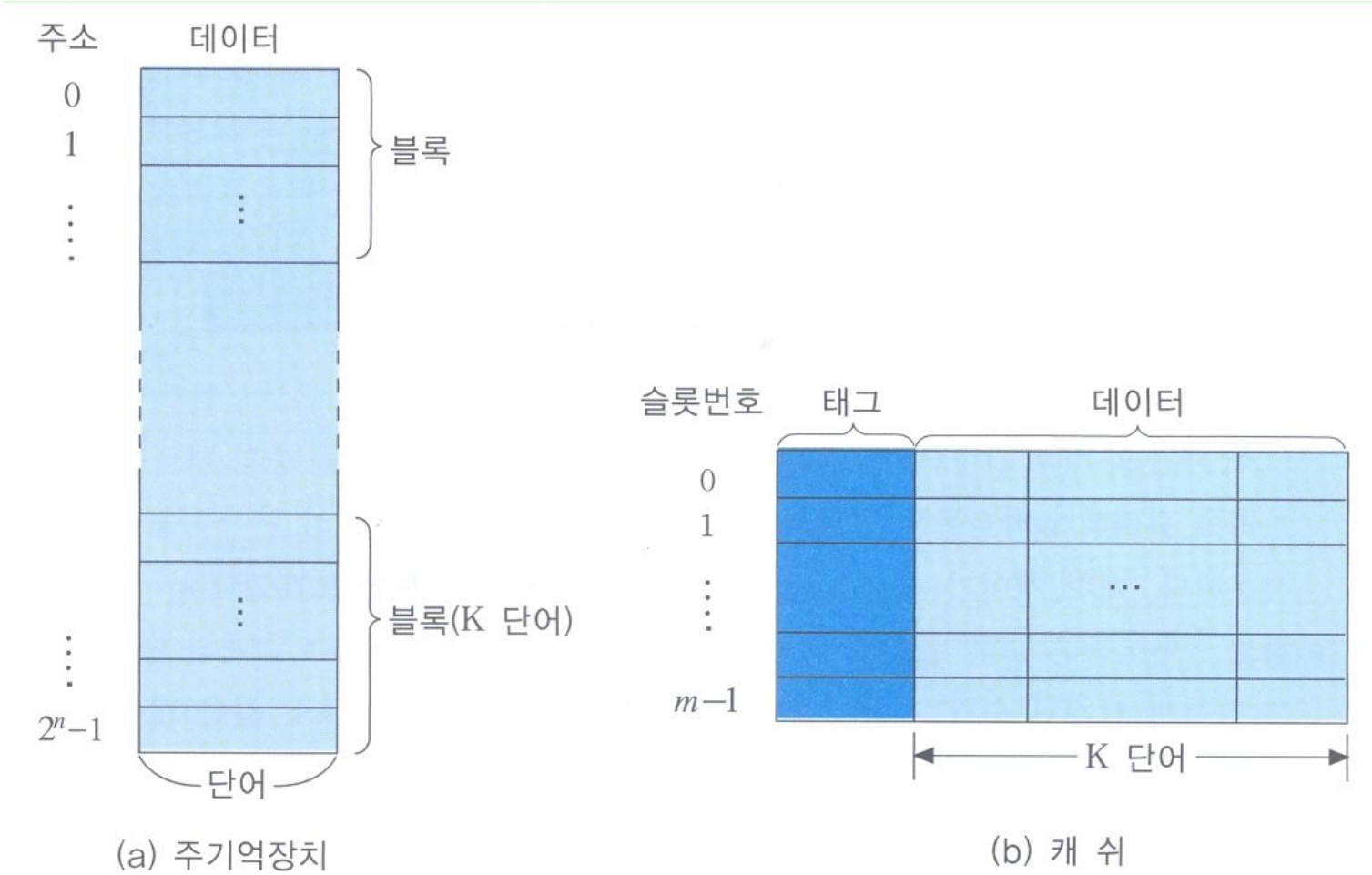
    주기억장치를 액세스할 때 함께 인출되는 정보들의 그룹을 블록(block)이라고 함.

    블록이 커지면 더 많은 정보들을 한 번에 읽어올 수 있지만, 인출 시간이 그만큼 길어짐.

캐쉬 인출 방식(2)

- 아래 그림은 블록으로 나누어진 주기억장치와 그에 따른 캐쉬의 조직을 나타냄.
  주기억장치는 2^n 개의 단어들로 구성되며, 각 단어는 n비트의 주소에 의해 저장됨.
  선인출을 위하여 주기억장치를 K개의 단어들로 이루어진 블록으로 나눌 경우에
  전체 블록들의 수는 2^n /K개가 됨.
- 캐쉬는 m개의 슬롯들로 구성되는데, 각 슬롯에는 주기억장치 블록과 같은 크기인
  K개의 단어들이 저장됨. 즉 슬롯의 크기와 주기억장치 블록의 크기가 같아야 함.
  선인출 방식을 적용하는 경우에 CPU가 주기억장치로부터 어떤 단어를 읽으면,
  그 단어를 포함하고 있는 블록 전체가 캐쉬 슬롯들 중의 어느 하나로 적재됨.
- 캐쉬의 용량이 작기 때문에 캐쉬 슬롯의 수는 주기억장치 블록의 수보다 훨씬 더
  적으므로 어느 순간에든 기억장치 블록들 중의 일부분만이 캐쉬에 적재될 수 있음.
- 결과적으로 캐쉬의 각 슬롯은 여러 개의 블록들에 의해 공유되므로 각 캐쉬 슬롯은
  데이터 블록 외에도 현재 자신을 공유하는 블록들 중의 어느 것이 적재되어 있는 지를
  구분해 주는 태그(tag)를 포함하고 있어야 함.

캐쉬 인출 방식(3)



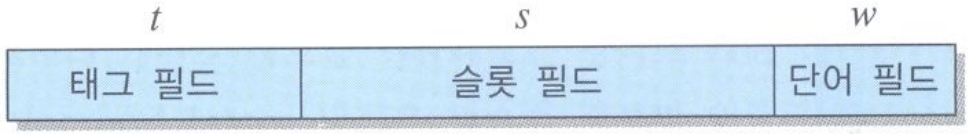
• 주기억장치와 캐쉬의 조직

2교시

캐쉬 사상 방식(1)

(1) 직접 사상

- 직접 사상 방식에서는 주기억장치의 블록들이 지정된 어느 한 캐쉬 슬롯으로만 사상 될 수 있는데 이 방식을 구현하기 위해서는 주기억장치 주소는 다음과 같이 세 개의 필드들로 구성됨.



- 이 주소의 상위 (t+s) 비트들은 주기억장치의 2^{t+s} 개의 블록들 중의 하나를 지정함. 그리고 최하위 w비트들은 2^{w} 개의 단어들로 구성된 각 블록 내의 단어들을 구분하는 데 사용됨.

  캐쉬 제어기(cache controller)는 최상위 t 비트들을 태그(tag) 번호로 해석하고, s 비트들은 슬롯 번호로 해석함.

  슬롯 번호는 캐쉬의 $m = 2^{s}$ 개의 슬롯들 중에서 그 블록이 적재될 수 있는 슬롯을 지정해 줌. 그리고 태그 번호는 동일한 캐쉬 슬롯을 공유하는 주기억장치 블록들을 서로 구분하는 데 사용됨.

캐쉬 사상 방식(2)

- 직접 사상 방식에서 주기억장치의 블록 j가 적재될 수 있는 캐쉬 슬롯의 번호 i는 다음과 같은 모듈로(modulo) 함수에 의해 결정됨.

  i = j mod m

  예를 들어 캐쉬 슬롯의 수 m = 4라면, 주기억장치의 여섯 번째 블록(j=6)은 6 mod 4 = 2이므로 캐쉬의 2번 슬롯에 적재될 수 있음.

  즉, 각 캐쉬 슬롯은 2^t 개의 블록들에 의해 공유됨.

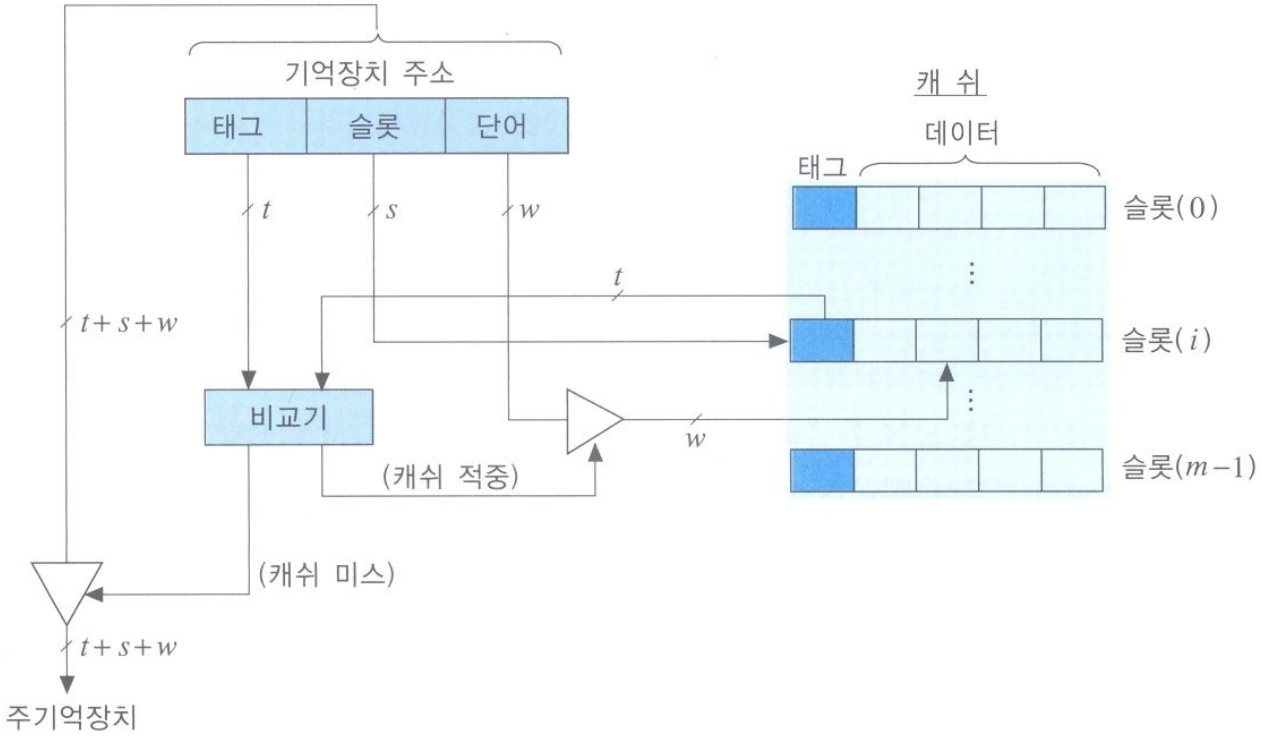
캐쉬 슬롯	주기억장치 블록 번호들
0	$0, \quad m, \quad \ldots, 2^{t+s}-m$
1	$1, \quad m+1, \quad \ldots, 2^{t+s}-m+1$
⋮	⋮
$m-1$	$m-1, \quad 2m-1, \quad \ldots, 2^{t+s}-1$

- 각 캐쉬 슬롯을 공유하는 주기억장치 블록들

캐쉬 사상 방식(3)

• 어떤 블록이 지정된 슬롯으로 적재되었을 때, 그 슬롯을 공유하는 다른 블록들과 구분될 수 있도록 하기 위하여 그 블록에 태그를 붙여야 함.

결과적으로 캐쉬의 각 슬롯에는 태그와 데이터 블록이 함께 저장되며, 캐쉬 적중 여부는 주소의 태그 비트들과 슬롯에 저장된 태그 비트들을 비교함으로써 결정됨.



• 직접 사상 캐쉬의 조직

캐쉬 사상 방식(4)

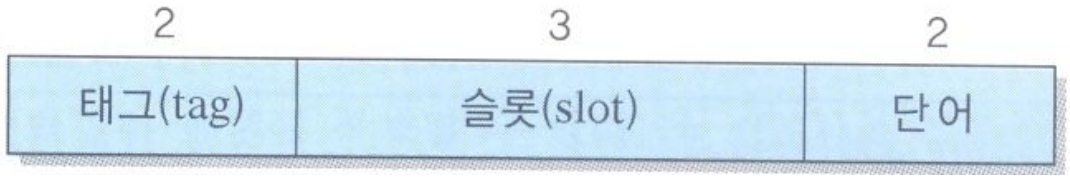
- 캐쉬로 기억장치 주소가 보내지면, 그 중 슬롯 필드의 s 비트들에 의해 캐쉬의 해당 슬롯이 선택되고 선택된 슬롯의 태그 비트들이 읽혀져서 주소의 태그 비트들과 비교됨.
  만약 그 두 태그 값이 일치하면 캐쉬가 적중된 것이며 그 경우에는 주소의 최하위에 있는 w개의 비트들을 이용하여 슬롯 내의 단어들 중에서 하나가 선택되고, 그 단어는 인출되어 CPU로 보내짐.
  그러나, 만약 태그 값이 일치하지 않는다면, 캐쉬가 미스되었음을 의미하므로 기억장치 주소 전체가 주기억장치로 보내져서 한 블록을 읽어오게 됨.
  인출된 블록은 지정된 캐쉬 슬롯에 적재되며, 주소의 태그 비트들이 그 슬롯의 태그 필드에 저장됨.
  만약 그 슬롯을 공유하는 다른 블록이 이미 적재되어 있는 상태라면 그 내용은 지워지고 새로이 인출된 블록이 적재됨.

캐쉬 사상 방식(5)

• 주기억장치와 캐쉬가 아래와 같은 크기를 가진 경우에 대하여 지금까지의 설명을 실제 적용해 보기로 함.

* 주기억장치의 용량은 128바이트임. 따라서 주기억장치의 주소는 7비트이며, 바이트 단위로 주소가 지정됨.
* 단어의 길이는 한 바이트인 것으로 가정함.
* 주기억장치의 블록 크기를 4단어(즉, 4바이트)로 함. 따라서 블록 수는 128/4 = 32개임.
* 캐쉬의 크기는 32바이트 임.
* 주기억장치의 블록 크기가 4바이트이므로 캐쉬 슬롯의 크기도 4바이트가 되어야 하며, 결과적으로 슬롯의 수 m = 32/4 = 8개가 됨.

이 예의 경우에 블록 내의 단어 수가 네 개이므로 w는 2비트이며, 슬롯의 수(m)가 8개이므로 s=3비트, 태그 필드(t)는 나머지 2비트가 됨.



캐쉬 사상 방식(6)

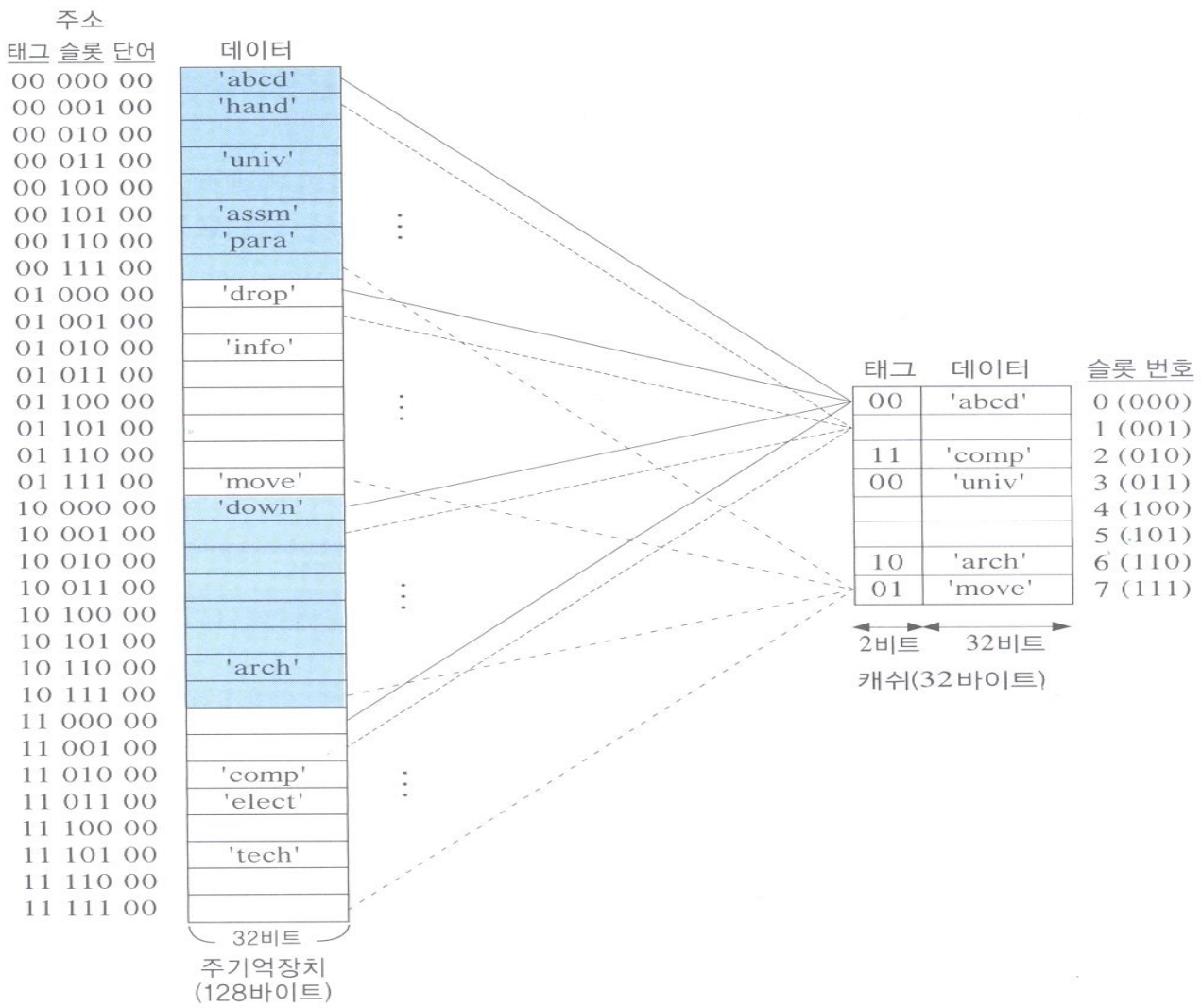
- 상기 예에서 캐쉬 슬롯의 수 m=8이므로, 각 기억장치 블록이 공유하게 될 캐쉬 슬롯의 번호 i = j mod 8이 됨. 따라서 직접 사상 방식을 적용하면 각 캐쉬 슬롯에 적재될 수 있는 주기억장치 블록들은 아래 표와 같음.

  각 캐쉬 슬롯을 공유하는 네 개의 블록들의 번호가 나열되어 있는데 끝의 세 자리 비트는 해당 캐쉬 슬롯 번호와 동일함.
- 직접 사상 방식은 간단하고, 구현하는 비용이 적게 든다는 장점이 있지만, 각 주기억장치 블록이 적재될 수 있는 캐쉬 슬롯이 한 개뿐이라는 단점이 있음.

캐쉬 슬롯	주기억장치 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

• 각 캐쉬 슬롯을 공유하는 주기억장치 블록들

캐쉬 사상 방식(7)

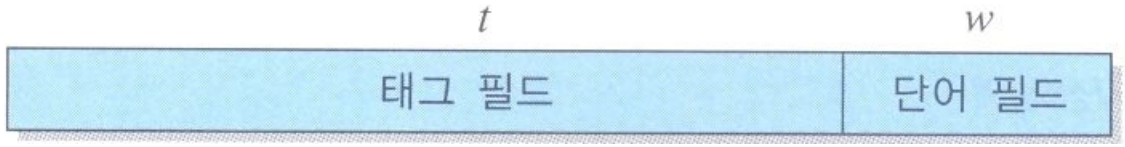


- 직접 사상의 예

캐쉬 사상 방식(8)

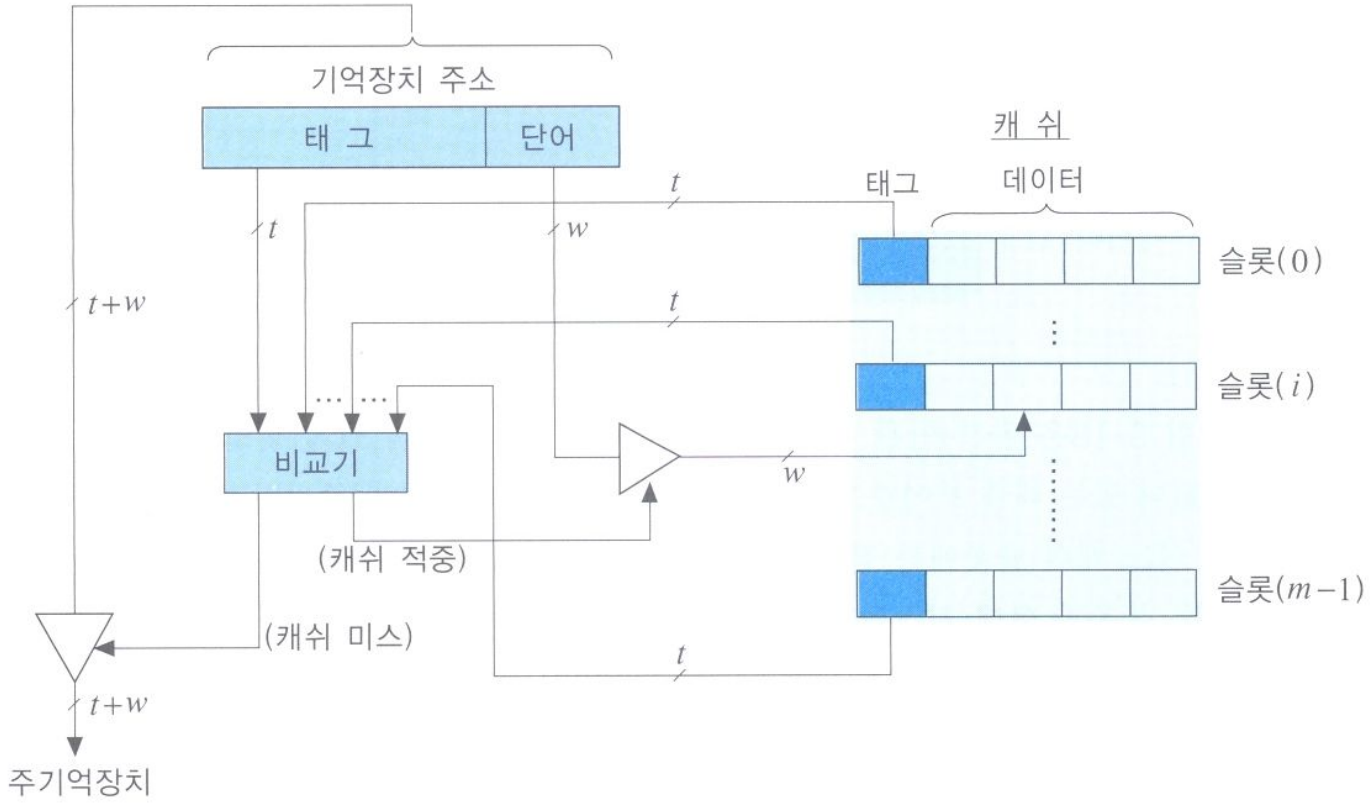
(2) 완전-연관 사상

- 완전-연관 사상(fully-associative mapping) 방식은 주기억장치 블록이 캐쉬의 어떤 슬롯으로든 적재될 수 있도록 허용함으로써 직접 사상 방식의 단점을 보완하고 있음.
  이 방식에서는 캐쉬 액세스 과정에서 주기억장치 주소가 아래와 같이 태그 필드와 단어 필드로만 구성됨.
  결과적으로 태그 값이 주기억장치 블록 번호와 동일함. 따라서 어떤 블록이 캐쉬로 적재되면, 그 주소의 태그 필드의 모든 비트들이 그 슬롯의 태그 부분에 저장됨.
- 주기억장치 블록은 캐쉬의 어떤 슬롯에든 적재될 수 있으므로, 캐쉬 적중 여부를 검사할 때는 캐쉬의 모든 슬롯들의 태그들과 주기억장치 주소의 태그 필드 내용을 비교하여 일치하는 것이 있는 지 확인해야 함. 만약 일치하는 슬롯이 있다면 캐쉬가 적중된 것이므로 단어 필드 값을 이용하여그 슬롯에 저장되어 있는 단어들 중의 하나를 인출하여 CPU로 전송함. 만약 없다면 캐쉬 미스이므로 주소를 주기억장치로 보내어 데이터를 액세스 함.



캐쉬 사상 방식(9)

• 아래 그림에서 기억장치 주소의 태그 비트들은 모든 캐쉬 슬롯의 태그들과 비교되고 있음. 이 비교를 순차적으로 수행한다면 많은 시간이 걸릴 것이므로 연관 기억장치 등을 이용하여 병렬로 신속히 이루어질 수 있도록 하드웨어가 구성되어야 함.

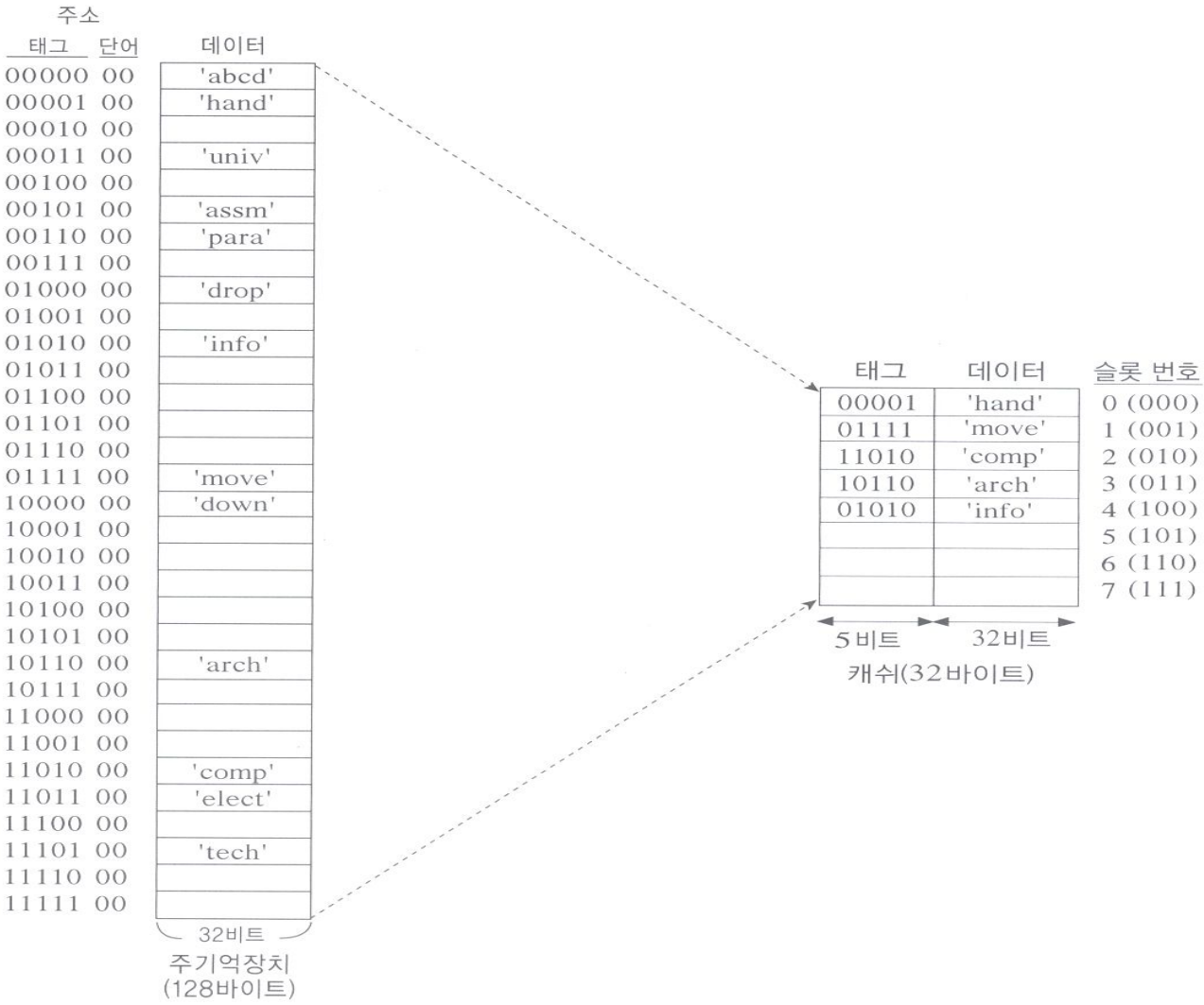


• 완전-연관 사상 캐쉬의 조직

캐쉬 사상 방식(10)

- 이 방식에서 유의할 점은 주기억장치로부터 인출된 데이터 블록을 캐쉬의 어느 슬롯에 적재할 것인가 하는 것임.
  캐쉬에 빈 슬롯들이 있다면, 슬롯 번호에 따라 차례대로 적재하면 되지만, 만약 비어 있는 슬롯 없이 캐쉬가 완전히 채워진 상태일 때 새로운 블록이 인출되어 온다면, 적절한 교체 알고리즘을 이용하여 슬롯들 중의 하나를 선택하고 새로운 블록을 적재해야 함.
  이때 그 슬롯에 먼저 적재되어 있던 블록은 지워지게 되므로 만약 그 블록이 캐쉬에 적재되어 있던 동안에 새로운 내용으로 변경된 적이 있었다면, 그 블록은 주기억장치에 갱신한 다음에 새로운 블록을 그 슬롯에 적재해야 함.
- 완전-연관 사상에서는 새로운 블록이 캐쉬로 적재될 때 슬롯의 선택이 자유롭기 때문에 프로그램 세그먼트나 데이터 배열 전체가 캐쉬로 적재될 수도 있음.
  이 경우에 만약 지역성이 높다면 적중률이 매우 높아질 것임.
  그러나 완전-연관 사상 방식은 캐쉬 슬롯들의 태그들을 병렬로 검사하기 위하여 복잡하고 비용이 높은 하드웨어를 포함해야 한다는 결정적인 단점이 있기 때문에,
  실제 시스템에서는 거의 사용되지 않고 있음.

캐쉬 사상 방식(11)



서울사이버대학교 SEOUL CYBER UNIVERSITY

- 완전-연관 사상의 예

3 교시

캐쉬 사상 방식(12)

(3) 세트-연관 사상

- 세트-연관 사상(set-associative mapping) 방식은 직접 사상 방식과 완전-연관 사상 방식의 장점만을 취하기 위한 절충안임.
- 캐쉬는 먼저 v개의 세트(set)들로 나누어지며, 각 세트는 k개의 슬롯들로 구성됨.
  따라서 캐쉬 슬롯의 수 m은 아래와 같으며, 각 주기억장치 블록이 적재될 수 있는 캐쉬 세트 번호의 i는 아래 식에 의해 결정됨.

  $m = v \times k$

  $i = j \bmod v$

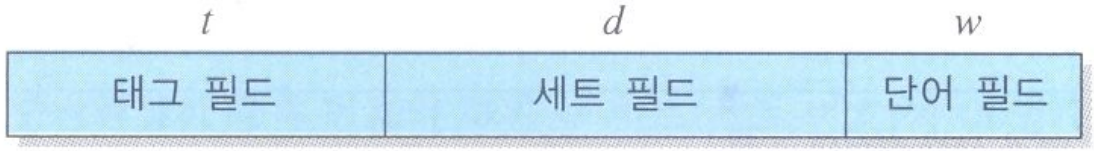
  단, m : 캐쉬 슬롯의 전체 개수, i : 캐쉬 세트의 번호 j : 주기억장치의 블록 번호

캐쉬 사상 방식(13)

- 이해를 돕기 위하여 먼저 이 식들을 앞에서 사용한 예에 적용해 보면, 전체 캐쉬 용량이 32바이트이고 슬롯의 크기는 4바이트이므로, 슬롯의 수 m = 32/4 = 8개임.
  만약 세트 당 두 개씩의 슬롯 (k=2)을 둔다고 가정한다면, 세트의 수 v = 8/2 =4개가 됨.
  결과적으로 캐쉬는 네 개의 세트들로 구성되고,
  각 세트에는 슬롯이 두 개씩 있는 구조가 됨.
  주기억장치의 7번 블록은 7 mod 4 = 3번 세트에 적재될 수 있는데
  직접 사상 방식과 다른 점은 7번 블록이 적재될 3번 세트에는 두 개의 슬롯들이 있으므로,
  그 중 한 개가 채워져 있더라도 다른 슬롯에 적재될 수 있다는 것임.
  이와 같이 각 세트에 슬롯이 두 개가 있는 경우를 2-way 세트-연관 사상이라고 말하며
  네 개 있는 경우를 4-way 세트-연관 사상이라고 말함.

캐쉬 사상 방식(14)

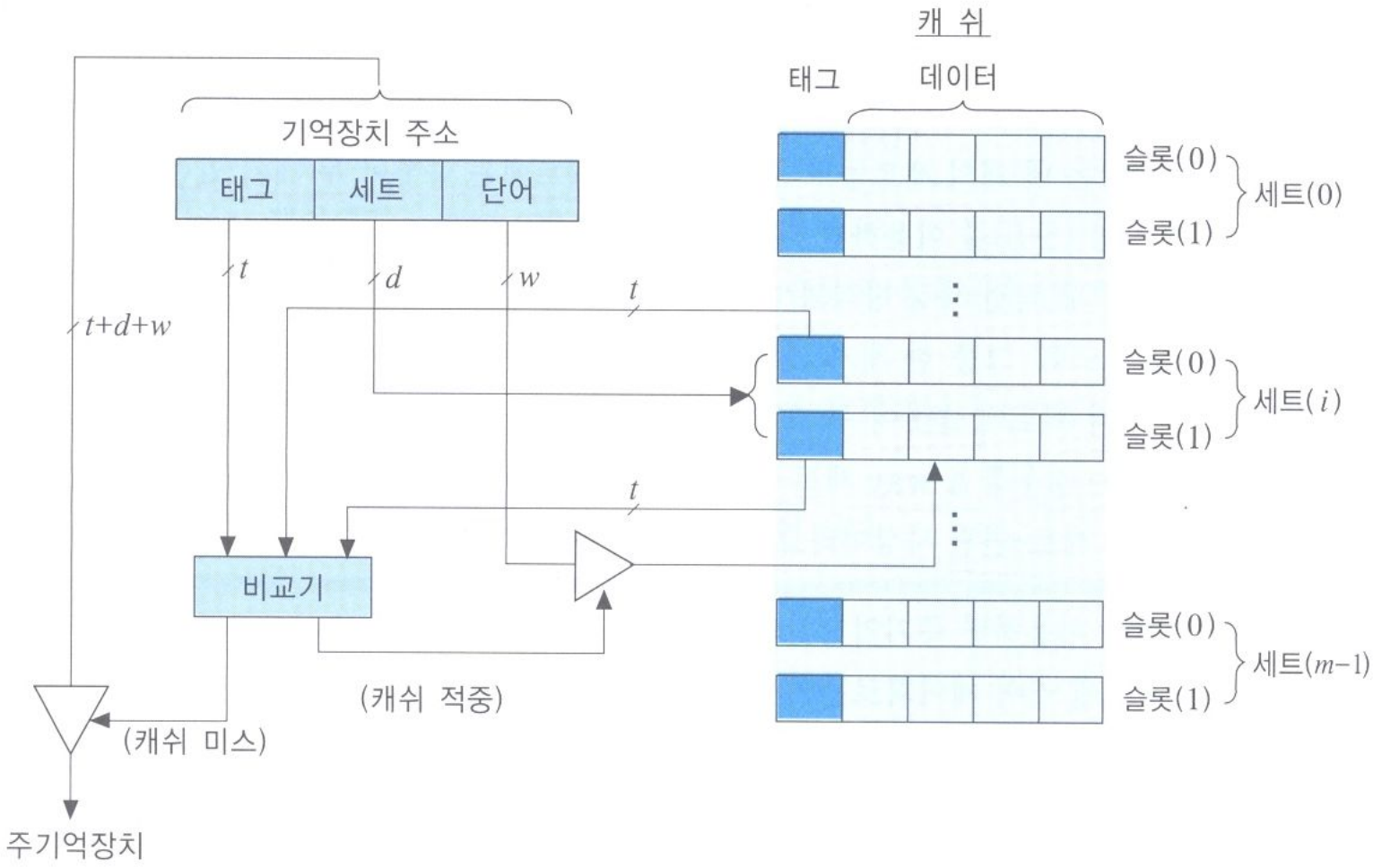
- 세트-연관 사상을 사용하면 주기억장치의 블록 Bj는 세트 i내의 어떤 슬롯이로든 사상될 수 있으며, 이 경우에 캐쉬 제어회로는 주기억장치 주소를 아래와 같은 세 개의 필드로 나누어 해석함.



여기서 d개의 세트 비트들은 캐쉬의 $v = 2^d$ 개의 세트들 중에서 해당 주기억장치 블록이 적재될 수 있는 세트를 선택하는 데 사용되고, 태그 필드의 t 비트들은 그 세트 내에 있는 슬롯들의 태그들과 비교되어 캐쉬 적중 여부를 결정하는 데 사용됨.

- 아래 그림은 2-way 세트-연관 사상 캐쉬의 조직을 보여주고 있음.
  지정된 세트에 포함된 슬롯의 태그들이 기억장치 주소의 태그와 비교됨.
  만약 일치되는 태그가 있다면, 캐쉬가 적중된 것이므로 w개의 비트에 의해 그 슬롯 내 2^w 개의 단어들 중 하나가 선택되어 인출됨.
  만약 그 세트 내 어느 슬롯의 태그와도 일치하지 않는다면, 캐쉬가 미스 된 것이므로 주기억장치 액세스가 시작됨.

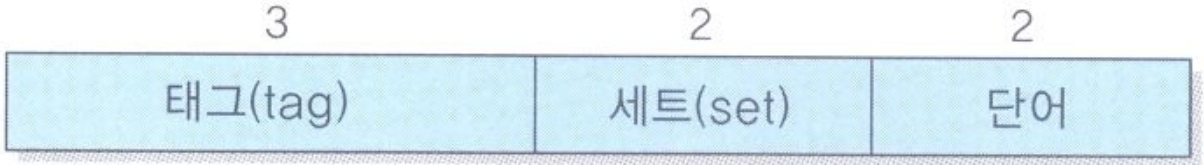
캐쉬 사상 방식(15)



• 세트-연관 사상 캐쉬의 조직

캐쉬 사상 방식(16)

- 이 방식을 사용하면 주기억장치 블록들은 지정된 어느 한 세트를 공유하는데, 각 세트에는 다수의 슬롯들이 있으므로, 직접 사상 방식에 비해 기억장치 블록이 적재될 수 있는 공간이 더 많아짐.
  그러나 캐쉬의 전체 크기가 고정되어 있기 때문에 세트의 수는 그만큼 감소함.
  예를 들어, 8개의 슬롯들로 이루어진 캐쉬에서 각 세트에 2개씩의 슬롯을 둔다면 세트 수는 4개가 됨. 그러나 각 세트에 슬롯을 4개씩 둔다면 세트 수는 2개가 될 것임.
- 앞에서 사용하였던 예에 2-way 세트-연관 사상 방식을 적용한다면, 기억장치 주소는 아래와 같이 세 개의 필드로 나누어진 것으로 해석됨.



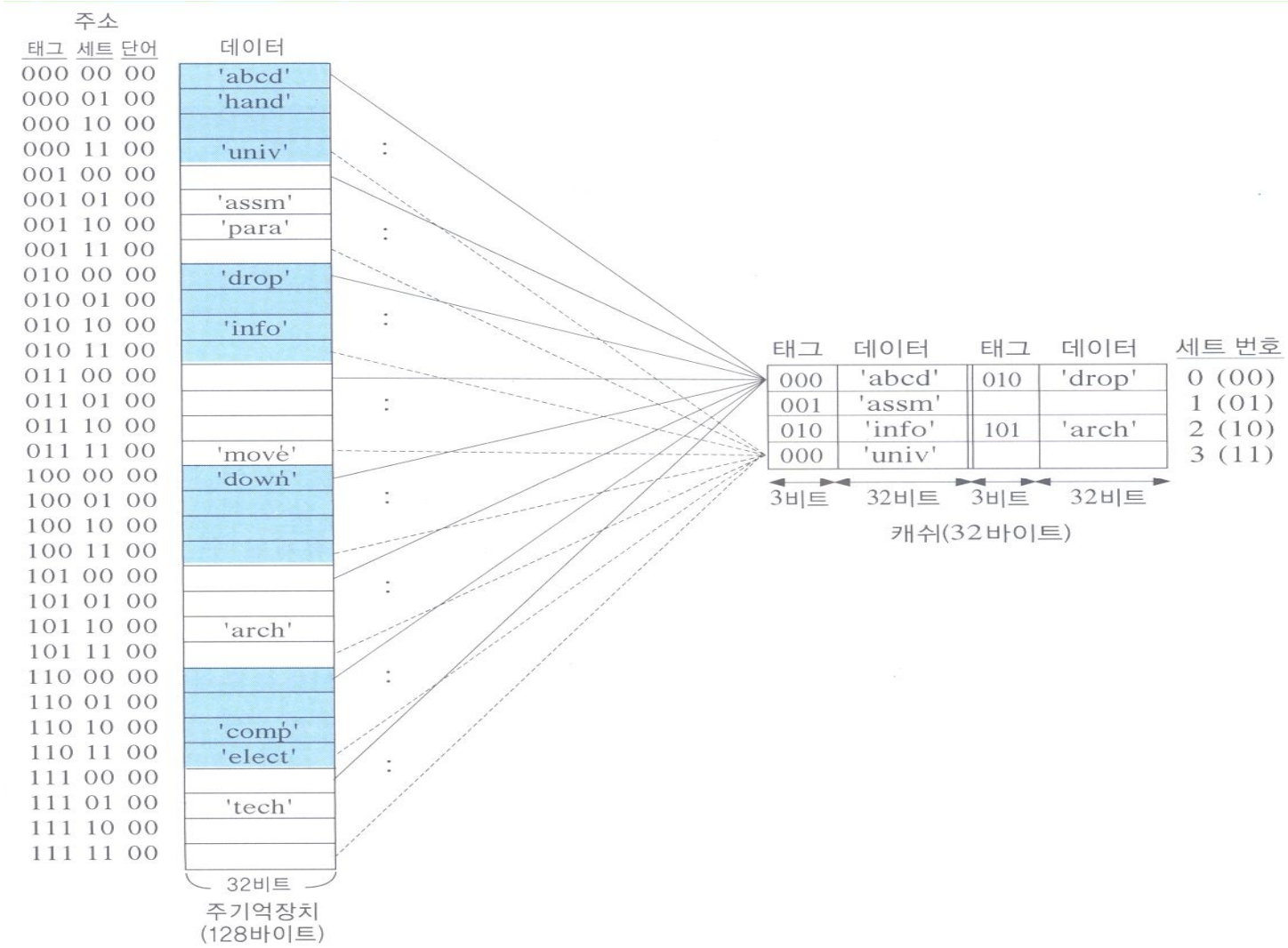
캐쉬 사상 방식(17)

- 이 예에서 캐쉬에는 전체적으로 8개의 슬롯들이 있고 그들이 2^2=4개의 세트들로 나누어지므로, 각 세트에는 슬롯이 두 개씩 있게 됨.
  세트 필드의 2비트는 4개의 캐쉬 세트들 중 하나를 선택하는 데 사용됨.
  그리고 태그 필드가 3비트 이므로 각 캐쉬 세트는 8개의 주기억장치 블록들에 의해 공유되며, 그들 중 두 개가 동시에 그 세트 내에 적재될 수 있음.

세트 번호	주기억장치 블록 번호들							
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

• 각 세트를 공유하는 주기억장치 블록들

캐쉬 사상 방식(18)



• 세트-연관 사상의 예

캐쉬 교체 알고리즘 (1)

- 캐쉬 미스가 발생하여 새로운 블록이 주기억장치로부터 캐쉬로 올라왔을 때, 그 블록이 적재될 수 있는 슬롯들이 이미 다른 블록들로 채워져 있다면 그 블록들 중의 하나가 교체(replace)되어야 함.
- 직접 사상에서는 새로운 블록이 적재될 수 있는 슬롯이 하나뿐이기 때문에 선택의 여지가 없으나, 완전-연관 사상과 세트-연관 사상에서는 적절한 교체 알고리즘이 필요함.
- 교체 알고리즘 들 중에서 최소 최근 사용(Least Recently Used : LRU) 알고리즘과 FIFO(First-In-First-Out : FIFO) 알고리즘이 있음.

  * LRU 알고리즘 : 사용되지 않은 채로 가장 오래 있었던 블록을 교체하는 것임.

    2-way 세트-연관 방식에서는 각 슬롯이 USE 비트를 가지고 있도록 하여, 어떤 슬롯이 액세스 되면, 그 슬롯의 USE 비트는 1로 세트 되고, 그 세트 내 다른 슬롯의 USE 비트는 0으로 세트 됨.

    새로운 블록이 그 세트로 적재될 때는 USE 비트가 0인 슬롯이 교체됨.

    더 최근에 사용되었던 데이터들이 앞으로도 다시 사용될 가능성이 높다는 가정이 맞는 경우가 실제로 많기 때문에 LRU 알고리즘은 높은 적중률을 보여 줌.

캐쉬 교체 알고리즘 (2)

* FIFO 알고리즘 : 캐쉬 내에 가장 오랫동안 있었던 블록을 교체하는 것임.

- 다른 알고리즘으로는 최소 사용 빈도(Least Frequently Used : LFU)가 있는데, 이 알고리즘은 참조되었던 횟수가 가장 적은 블록을 교체하는 것임. LFU는 각 슬롯에 계수기(counter)를 설치하여 구현할 수 있음.
- 사용 횟수에 근거하지 않는 기법으로는 후보 슬롯들 중에서 한 슬롯을 임의(random)로 선택하는 것임.

캐쉬 쓰기 정책 (1)

• 캐쉬의 블록이 변경되었을 때 그 내용을 주기억장치에 갱신하는 시기와 방법을 결정하는 것을 쓰기 정책(write policy)이라고 함.

여기에는 write through와 write back이 있음

* write through : 모든 쓰기 동작들이 캐쉬로 뿐만 아니라 주기억장치로도 동시에 행해짐.

따라서, 주기억장치의 내용들은 항상 유효함.

이 방식의 단점은 쓰기 동작에 걸리는 시간이 길어진다는 것임.

* write back : 캐쉬에서 데이터가 변경되어도 주기억장치에는 갱신되지 않음.

만약 캐쉬에서 데이터가 변경되면, 그 블록의 M(modified) 비트가 세트 되고 나중에 그 블록이 교체될 때, M 비트가 세트 된 경우에는 주기억장치에 갱신된 후에 교체됨.

결과적으로 write back 정책을 사용하면 기억장치에 대한 쓰기 동작의 횟수가 최소화 됨.

또한 쓰기 동작이 캐쉬까지만 이루어지므로 쓰기 시간이 짧아짐.

문제점으로는 주기억장치의 일부 블록들이 새로운 내용으로 갱신될 때까지는 무효 상태에 있게 된다는 점임.

또한 제어 회로가 복잡해지고, 각 캐쉬 슬롯이 상태 비트를 가지고 있어야 하는 단점이 있음.

캐쉬 쓰기 정책 (2)

- 쓰기 정책에 따른 쓰기 동작의 비교

셀프 테스트

• 주기억장치가 1,024 단어이고, 캐쉬가 32 단어이며, 하나의 슬롯은 4 단어로 이루어진다면, 슬롯의 개수는 몇 개인가?

* 8개

해설) 캐쉬가 32단어의 용량을 가지는데 하나의 슬롯이 4단어라면 슬롯의 개수는 32 나누기 4 =8이 됨.

• 직접 사상 방식에서 슬롯의 수가 8일 때 주기억장치의 20번째 블록은 캐쉬의 몇 번 슬롯에 적재될 수 있는가?

* 4번 슬롯

해설) 20 mod 8인데 이것은 20을 8로 나눈 나머지라는 의미이므로 20나누기 8 = 몫 2, 나머지 4, 즉 4번 슬롯

• 캐쉬 교체 알고리즘에서 캐쉬 내에 가장 오랫동안 있었던 블록을 교체하는 알고리즘은 무엇인가?

* FIFO 알고리즘

해설) FIFO는 First In First Out로서 먼저 입력된, 즉 오래된 블록이 교체 대상인 알고리즘임.

요점 정리

- CPU가 원하는 데이터가 이미 캐쉬에 있는 상태를 캐쉬 적중(cache hit)이라고 하고, 반대로 CPU가 원하는 데이터가 캐쉬에 없는 상태를 캐쉬 미스(cache miss)라고 함.
- 캐쉬 설계에 있어서의 공통적인 목표는 캐쉬 적중률의 극대화, 캐쉬 액세스 시간의 최소화, 캐쉬 실패에 따른 지연시간의 최소화, 주기억장치로부터 캐쉬로 정보를 읽어오는 데 걸리는 시간의 최소화 등임.
- 직접 사상 방식에서는 주기억장치의 블록들이 지정된 어느 한 캐쉬 슬롯으로만 사상될 수 있는데 이 방식을 구현하기 위해서는 주기억장치 주소는 태그필드, 슬롯 필드, 단어 필드 등으로 구성됨.
- 완전-연관 사상(fully-associative mapping) 방식은 주기억장치 블록이 캐쉬의 어떤 슬롯으로든 적재될 수 있도록 허용함.
- 세트-연관 사상(set-associative mapping) 방식은 직접 사상 방식과 완전-연관 사상 방식의 장점만을 취하기 위한 절충안임.
- 교체 알고리즘 들 중에서 최소 최근 사용(Least Recently Used : LRU) 알고리즘과 FIFO(First-In-First-Out : FIFO) 알고리즘이 있음.