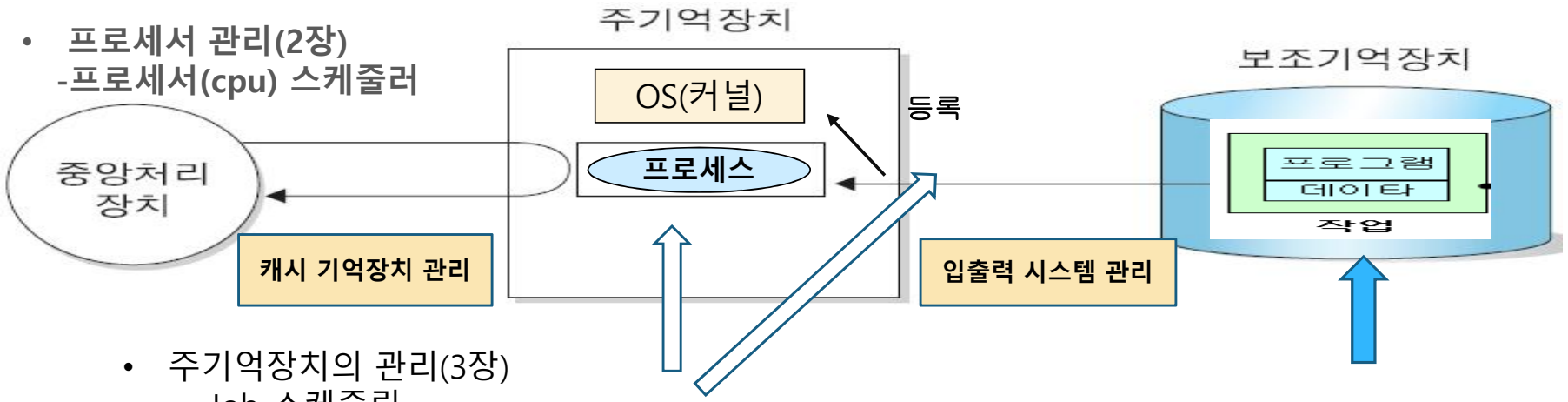
운 영 체 제

캐시 기억장치 관리

5장 디스크 스케줄링과 파일 시스템



- 프로세서 관리(2장)
  -프로세서(cpu) 스케줄러

- 주기억장치의 관리(3장)
  -Job 스케줄링
  -주소 바인딩
  -주기억장치 관리 기법
  =>인출 기법, 배치 기법,교체 기법, 할당 기법(연속)
- 가상 메모리 관리(4장)
  -분산 할당 기법
  =>페이징 기법, 세그먼테이션 기법
  -페이지 교체 기법
  => FIFO(First In First Out),
  OPT(Optimal Replacement), LRU, LFU, NUR, SCR

- 디스크 스케줄링 (5장)
  -FCFS, SSTF, SCAN, C-SCAN
  => 탐색시간 최소화
- 파일 시스템(5장)
  -파일 구조
  -디렉토리 구조

학습 내용

캐시 기억장치 관리

- 캐시기억장치의 개념
  - 기억장치들 간의 상호 연관성
- 캐시기억장치의 원리
- 캐시기억장치와 주기억장치 사이의 정보교환 방법
- 캐시기억장치의 교체 알고리즘

캐시기억장치의 개념

- 주기억장치에 저장되어 있는 명령어와 데이터 중의 일부를 임시적으로 복사해서 저장하는 장치로 데이터를 저장하고 인출하는 속도가 주기억장치보다 빠름

- 중앙처리장치가 캐시기억장치에 저장된 명령어와 데이터를 처리할 경우, 주기억장치보다 더 빠르게 처리할 수 있음

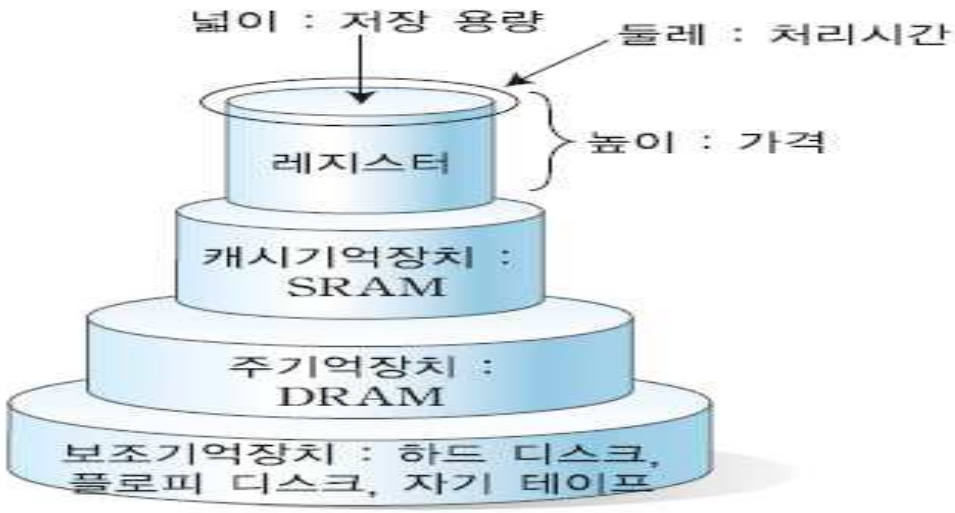
- 결과적으로 캐시기억장치는 느리게 동작하는 주기억장치와 빠르게 동작하는 중앙처리장치 사이에서 속도차이를 줄여줘서 중앙처리장치에서의 데이터와 명령어 처리속도를 향상시킴

- 캐시기억장치는 고속 완충 기억 장치라고 함

캐시기억장치의 개념

❖ 컴퓨터의 기억장치 구성

- 기억장치의 계층적으로 분류

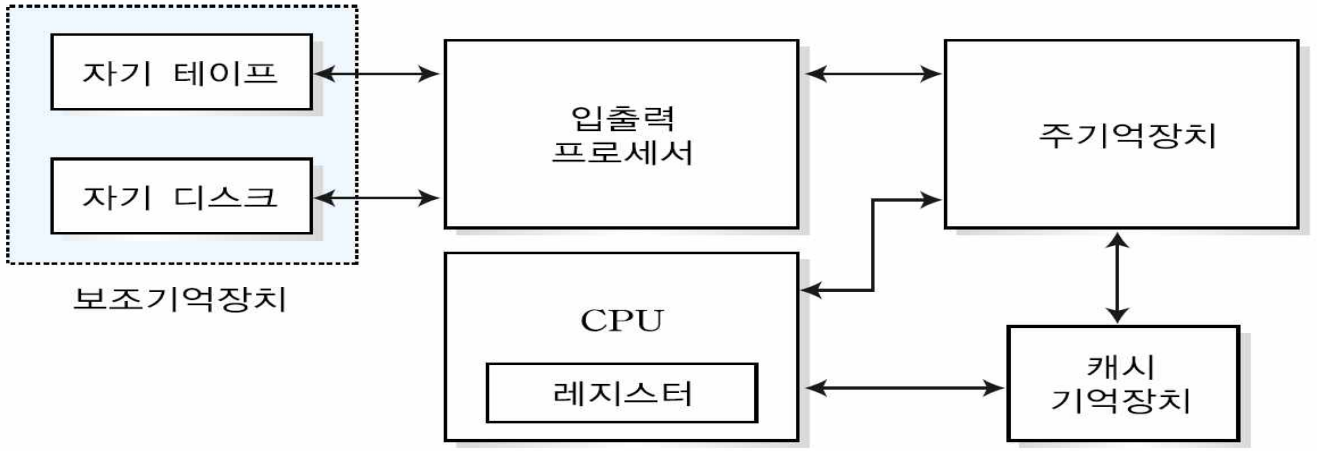


- 원의 넓이 : 저장 용량을 나타냄.
- 원의 둘레 : 기억장치가 데이터를 저장하거나 인출하는데 걸리는 시간. 처리속도를 말함.
- 원통의 높이 : 기억장치의 가격을 나타냄.

캐시기억장치의 개념

- 중앙처리장치 내부
  - CPU의 처리속도와 비슷한 접근속도를 가진 레지스터들이 포함한다.
  - 이러한 기억장치는 높은 가격 때문에 많은 용량으로 구성하기 어렵다.

- 캐시(Cache)기억장치
  - 주기억장치에 비해 5~10배 정도 접근속도가 빠르다.
  - 자주 사용되는 명령들을 저장하고 있다가 중앙처리장치에 빠른 속도로 제공한다.
  - 캐시기억장치의 용량에 의해 CPU의 가격이 결정된다.

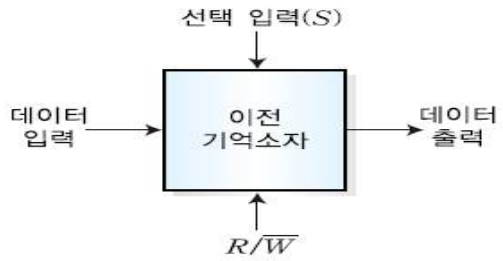


캐시기억장치의 개념

❖ SRAM의 기억소자 구조

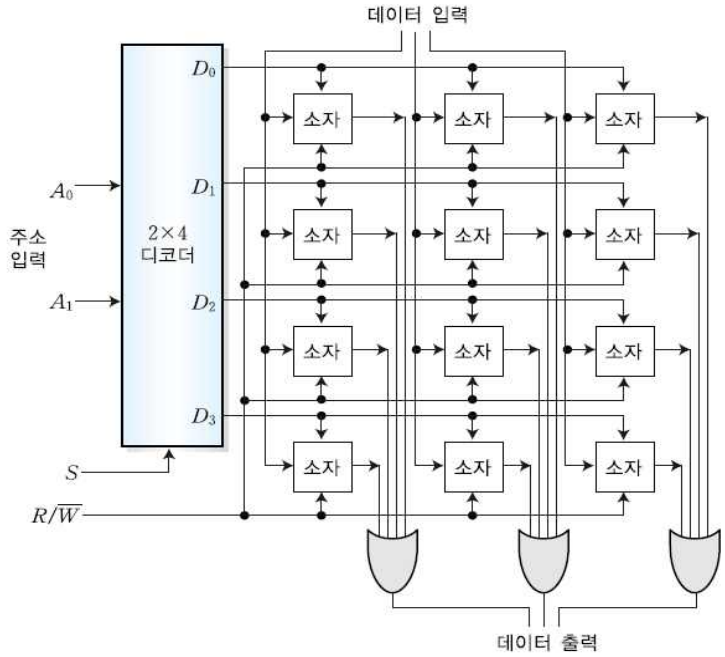
- SRAM의 기억소자들을 격자 구조로 배열하면 많은 비트를 저장할 수 있는 진정한 SRAM이 됨
-

▪ SRAM의 기억소자 블록



▪ 격자의 배열구조로 형성된 SRAM의 기본 구조

- 기억소자들이 4x3의 배열이므로 12비트를 저장, 워드의 길이는 3비트가 된다.
- 4개의 워드가 존재하므로 주소는 0, 1, 2, 3의 4개가 필요. 2x4 디코더는 2비트를 가지고 4개 중에서 하나를 선택한다.

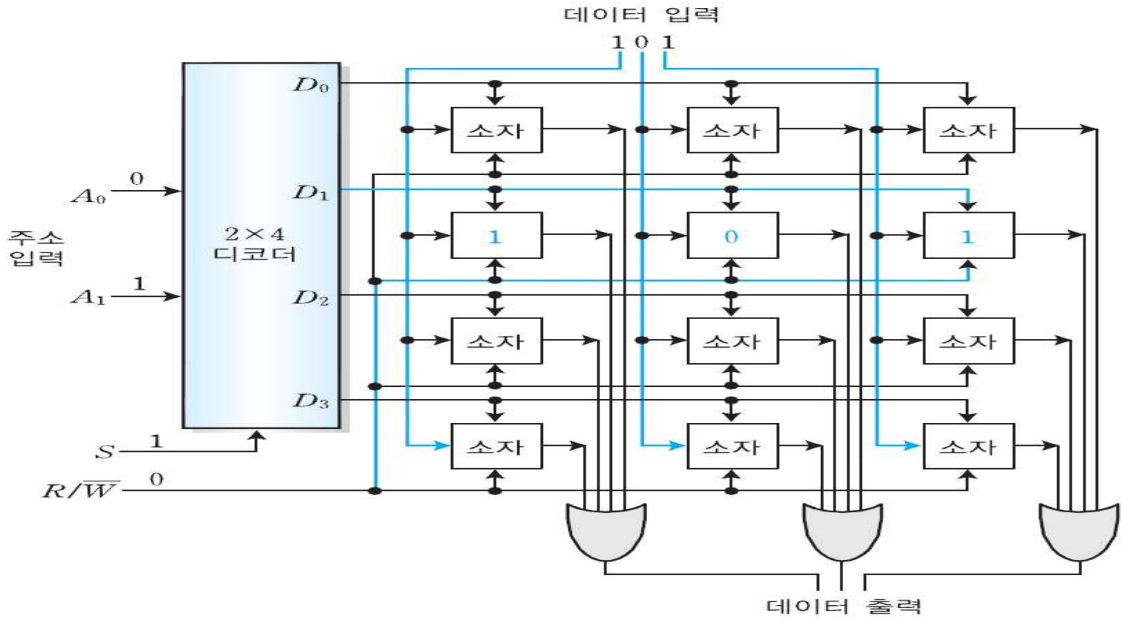


주소	SRAM(4X3=12bit)		
00			
01			
10			
11			

캐시기억장치의 개념

❖ SRAM의 쓰기 동작 과정

- 주소 선택선을 통해 입력신호 A_0A_1=01이 입력되면 디코더에서 주소 D_1이 선택된다.
- 그리고 제어 단은 $R/\overline{W}$=0 이 되어 쓰기동작을 수행한다. 따라서 주소 D_1의 기억 소자 3개에 입력 데이터 101이 저장되게 된다.
- 여기서 모든 기억소자에 입력 값이 전달되지만, D_1을 제외한 다른 주소는 쓰기 제어 입력을 받지 못하므로 어떤 값도 저장되지 않는다.

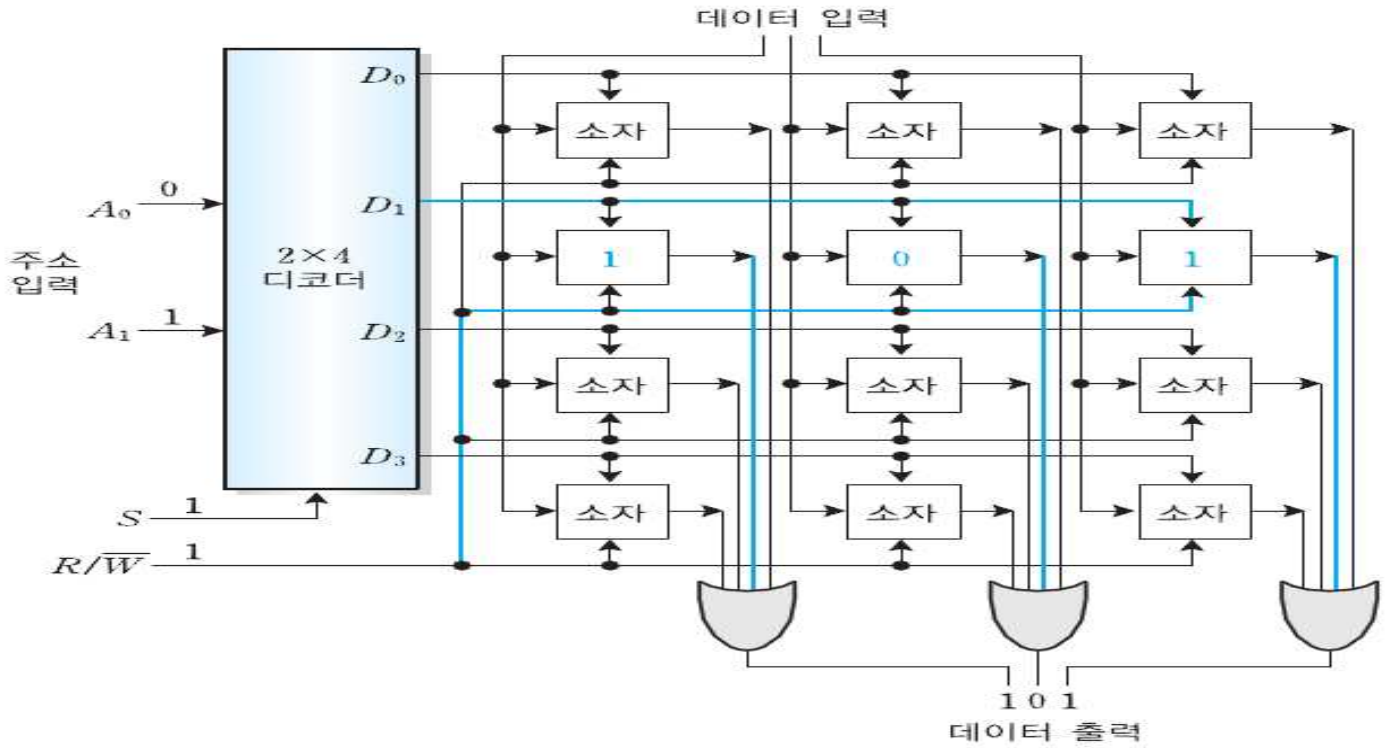


주소	SRAM(4X3=12bit)		
00			
01	1	0	1
10			
11			

캐시기억장치의 개념

❖ SRAM의 읽기 동작 과정

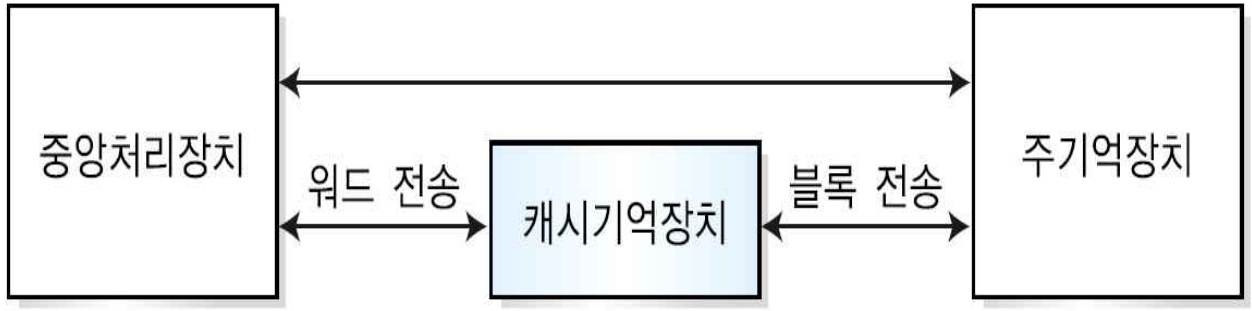
- 주소 선택선을 통해 들어온 입력신호 A_0A_1는 01의 값에 해당되는 주소가 디코더에서 선택된다.
- 이때 제어 단은 $R/\overline{W}$=1 이 되어, 선택된 각각의 기억소자는 출력단자를 통하여 출력을 내보내게 된다.



주소	SRAM(4X3=12bit)		
00			
01	1	0	1
10			
11			

캐시기억장치의 원리

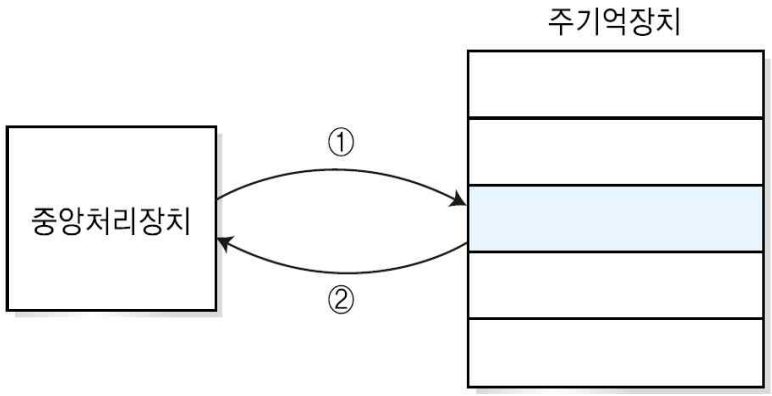
- 중앙처리장치는 그 속도가 저장장치에 비해서 고속이므로 저장장치의 읽기와 쓰기 동작과정 동안 기다려야 하며, 이런 문제를 극복하기 위해서는 중앙처리장치의 처리속도만큼 빠른 저장장치가 필요
- 주기억장치는 보조기억장치보다 처리속도가 빠르지만 중앙처리장치의 처리속도와 비교하면 그 차이는 크다. 그래서 주기억장치보다 빠른 저장장치를 생각하게 되었고 이에 따라 **캐시기억장치가 등장**
- 캐시기억장치는 5~100ns 정도의 빠른 접근시간을 제공하는 기억장치로 수행할 명령어나 피연산자를 주기억장치로부터 가져와 저장하고 있다가 빠른 속도로 중앙처리장치에 제공함



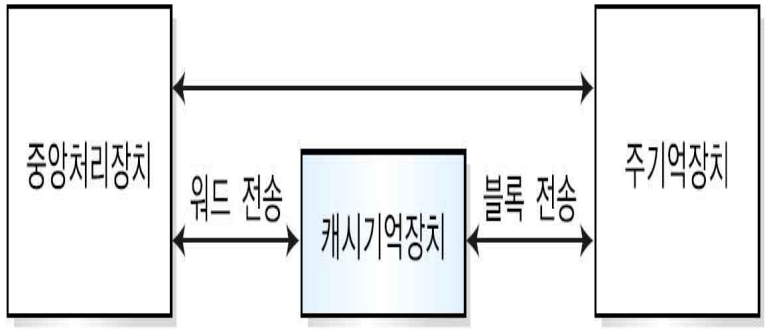
캐시기억장치의 원리

❖ 캐시기억장치의 동작

- 캐시기억장치가 없는 컴퓨터 시스템의 기억장치 접근
  - ① 단계 : CPU가 명령어와 데이터를 인출하기 위해서 주기억장치에 접근한다.
  - ② 단계 : 주기억장치에서 명령어나 필요한 정보를 획득하여 CPU내의 명령어 레지스터 등에 저장한다.



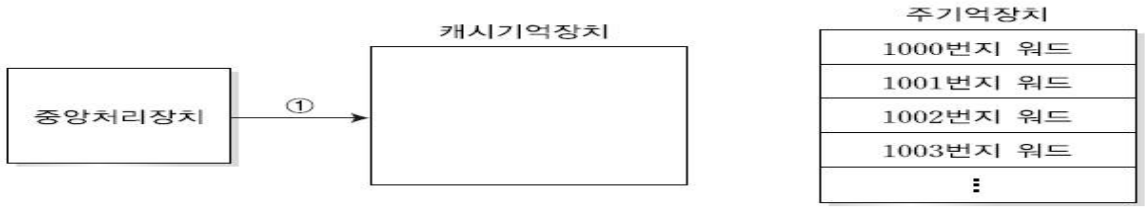
- 캐시기억장치를 포함하고 있는 컴퓨터 시스템
  - CPU가 명령어 또는 데이터를 인출하기 위해 주기억장치보다 캐시기억장치를 먼저 조사한다.
  - CPU가 명령어를 인출하기 위해 캐시기억장치에 접근하여 그 명령어를 찾았을 때를 적중(hit)이라고 하고, 명령어가 존재하지 않아 찾지 못하였을 경우를 실패(miss)라고 한다.



캐시기억장치의 원리

❖ 캐시기억장치의 실패(miss)

- 중앙처리장치가 1000번지의 워드가 필요하고 캐시기억장치에 존재하지 않을 경우
  - 1단계에서 캐시기억장치가 1000번지의 워드를 저장하고 있는지를 검사하고 1000번지 워드가 캐시기억장치 내에 존재하지 않는다면 실패상태가 된다



- 실패인 경우에 원하는 정보를 찾아 CPU로 전달하는 과정
  - ① 단계 : 주기억장치에서 필요한 정보를 획득하여 캐시기억장치에 전송한다.
  - ② 단계 : 캐시기억장치는 얻어진 정보를 다시 중앙처리장치로 전송한다.

캐시기억장치의 원리

❖ 캐시기억장치의 적중(hit)

- CPU가 1002번지의 워드를 필요로 하고 이것이 캐시기억장치에 존재할 경우

- ① 단계 : 캐시기억장치가 1002번지의 워드를 저장하고 있는지를 검사하고, 1002번지 워드가 캐시기억장치 내에 존재 한다면 적중이 된다.

- ② 단계 : 캐시기억장치에서 얻어진 정보를 중앙처리장치로 직접으로 전송한다. 주기억장치를 거치는 것보다 훨씬 빠른 속도로 원하는 정보를 획득하게 된다.

캐시기억장치의 원리

❖ 적중률(Hit Ratio)

- 적중률은 캐시기억장치를 가진 컴퓨터의 성능을 나타내는 척도로 적중률이 높을수록 속도가 향상

$$\text{적중률} = \frac{\text{적중 수}}{\text{전체 메모리 참조 횟수}}$$

- 주기억장치와 캐시기억장치에서 데이터를 인출하는데 소요되는

평균 기억장치 접근시간 $T_{average}$ = 캐시기억장치 접근시간 평균+ 주기억장치 접근시간 평균

$$T_{average} = H_{hit\_ratio} \times T_{cache} + (1 - H_{hit\_ratio}) \times T_{main}$$

$T_{average}$ = 평균 기억장치 접근 시간	T_{main} = 주기억장치 접근 시간
T_{cache} = 캐시기억장치 접근 시간	H_{hit_ratio} = 적중률

- .
- 평균 캐시기억장치 접근시간은 캐시기억장치 접근시간 T_{cache}와 적중률 H_{hit_ratio}와의 곱으로 얻어진다.
- 평균 주기억장치 접근시간은 주기억장치 접근시간 T_{main}과 실패율 $(1-H_{hit_ratio})$와의 곱으로 얻어진다. 여기서 실패율은 곧 주기억장치에 접근하는 율을 나타낸다.

캐시기억장치의 원리

❖ 평균 기억장치 접근시간 $T_{average}$ 계산 예

- T_{cache} = 50ns, T_{main} = 400ns일 때, 적중률을 증가시키면서 기억장치 접근시간을 계산하면
  - 적중률 70%의 경우: $T_{average}$ = 0.7 x 50ns + 0.3 x 400ns = 155ns
  - 적중률 80%의 경우: $T_{average}$ = 0.8 x 50ns + 0.2 x 400ns = 120ns
  - 적중률 90%의 경우: $T_{average}$ = 0.9 x 50ns + 0.1 x 400ns = 85ns
  - 적중률 95%의 경우: $T_{average}$ = 0.95 x 50ns + 0.05 x 400ns = 67.5ns
  - 적중률 99%의 경우: $T_{average}$ = 0.99 x 50ns + 0.01 x 400ns = 53.5ns

  - 캐시기억장치의 적중률이 높아질수록 평균 기억장치 접근시간은 캐시기억장치 접근시간에 근접하게 되어 컴퓨터의 처리 속도의 성능 향상을 가져온다.

  - **평균 기억장치 접근시간 $T_{average}$ = 캐시기억장치 접근시간 평균+ 주기억장치 접근시간 평균**
    = 적중률 X 캐시기억장치 접근시간 평균+ 실패율 X주기억장치 접근시간 평균

캐시기억장치 설계 시 고려할 요소

❖ 캐시기억장치 설계 시 고려할 요소

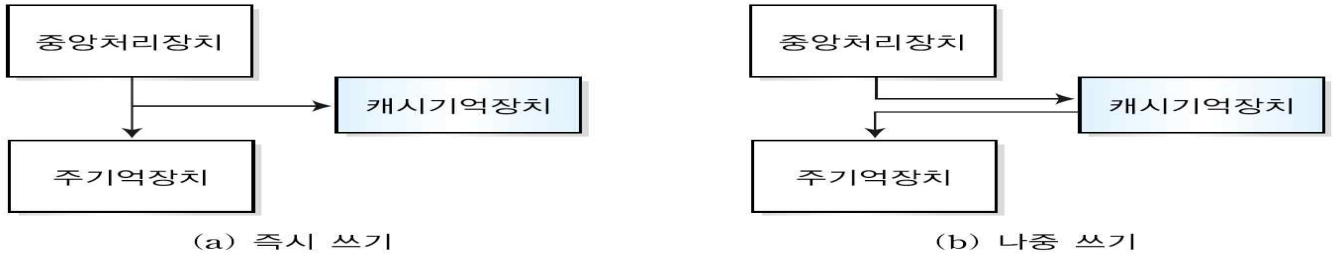
- 캐시기억장치의 크기(Size)
  - 캐시기억장치의 크기와 비용 간의 조정을 통해 적절한 용량과 비용을 결정해야 한다. 연구 결과에 의하면 1 k~128 k 단어(word)가 최적이라고 알려져 있음
- 인출 방식(fetch algorithm)
  - 요구인출(Demand Fetch) 방식
  - 선인출(Prefetch) 방식
- 쓰기 정책(Write policy)
- 블록 크기(Block size)
- 캐시기억장치의 수(Number of caches)
- 사상 함수(Mapping function)
- 교체 알고리즘(Replacement algorithm)

캐시기억장치 설계 시 고려할 요소

❖ 쓰기 정책(Write Policy)

- CPU가 프로그램을 실행하는 동안 연산 결과를 캐시기억장치에 기록하는 경우가 발생한다. 이때 캐시기억장치와 주기억장치에 저장된 데이터가 상이하게 존재하므로 주기억장치의 데이터를 갱신하는 절차가 필요하다.

❖ 쓰기 정책의 종류



- 즉시 쓰기(Write-though) 방식
  - CPU의 연산 결과가 기억장치에 저장하는 쓰기 동작은 캐시기억장치뿐만 아니라 주기억장치에서도 동시에 발생, 데이터의 일관성을 쉽게 보장할 수 있다.
- 나중 쓰기(Write-back) 방식
  - 새롭게 생성된 중앙처리장치의 데이터를 캐시기억장치에만 기록하고 기록된 블록에 대한 교체가 일어날 때 주기억장치에 기록하는 방식

캐시기억장치 설계 시 고려할 요소

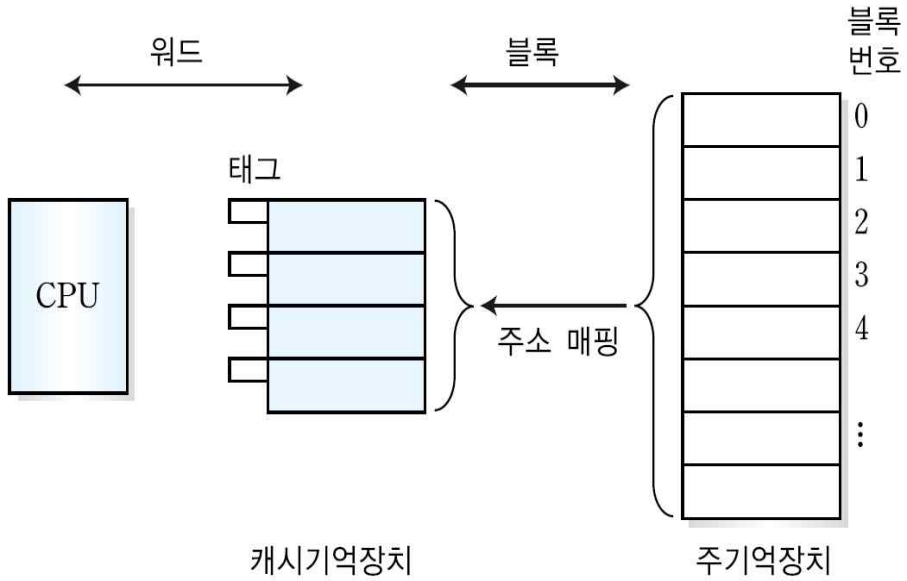
❖ 블록 크기와 캐시의 수

- 블록 크기(Block size)
  - 동시에 인출되는 정보들의 블록이 커지면 한꺼번에 많은 정보를 읽어 올 수 있지만 블록 인출시간이 길어지게 된다.
  - 블록이 커질수록 캐시기억장치에 적재할 수 있는 블록의 수가 감소하기 때문에 블록들이 더 빈번히 교체되며, 블록이 커질수록 멀리 떨어진 단어들도 같이 읽혀오기 때문에 가까운 미래에 사용될 가능성이 낮다.
  - 일반적인 블록의 크기는 4 ~ 8단어가 적당하다.
- 캐시의 수(Number of Caches)
  - 일반적인 시스템은 오직 하나의 캐시기억장치를 가지고 있었다.
  - 최근에는 캐시기억장치들이 계층적 구조로 설치되거나 기능별로 분리된 다수의 캐시기억장치를 사용하는 것이 보편화 되었다.
  - 캐시기억장치를 설계할 때에는 몇 계층으로 할 것인지를 결정하여야 하며, 통합 형태와 분리 형태 중에서 어떤 형태로 구성할 것인지를 결정해야 한다.
  - 설계과정에서 사용할 캐시의 수가 결정된다.

캐시기억장치와 주기억장치 사이의 정보교환 방법

❖ 주기억장치와 캐시기억장치 간의 정보 공유

- 주기억장치에 저장된 데이터의 일부가 블록단위로 캐시기억장치에 복사되고,
  중앙처리장치가 적중된 데이터들을 워드 단위로 캐시기억장치에서 읽어온다.



▪ 주기억장치와 캐시기억장치 간의 정보 공유

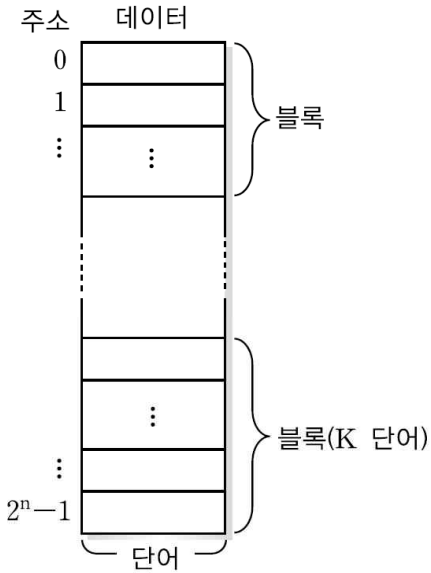
캐시기억장치와 주기억장치 사이의 정보교환 방법

❖ 사상 함수

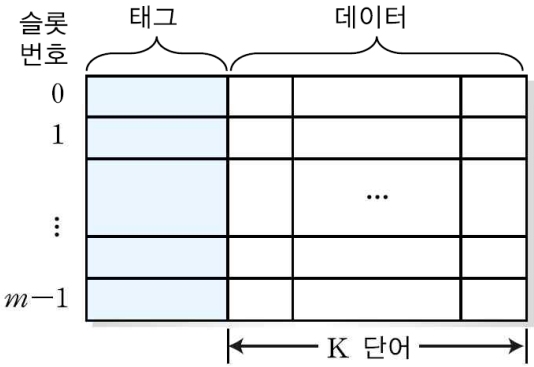
- 주기억장치에서 하나의 번지에 저장되는 데이터의 단위는 단어가 된다.
- 단어가 여러 개 모여서 하나의 블록(Block)이 되며, 캐시기억장치로 인출되는 정보의 그룹이다.
- 만약 주기억장치의 용량이 2^n 개의 단어라고 하고 블록이 K개의 단어로 구성된다면, 블록의 개수는 블록의 수 = $2^n/K$ 개다.

▪ 주기억장치와 캐시기억장치의 구조

- 캐시기억장치에서 슬롯(Slot)은 한 블록이 저장되는 장소다. 따라서 블록은 캐시기억장치 각 슬롯에 저장되는 데이터의 길이가 된다.
- 태그(tag)는 슬롯에 적재된 블록을 구분해주는 정보다.



(a) 주기억장치

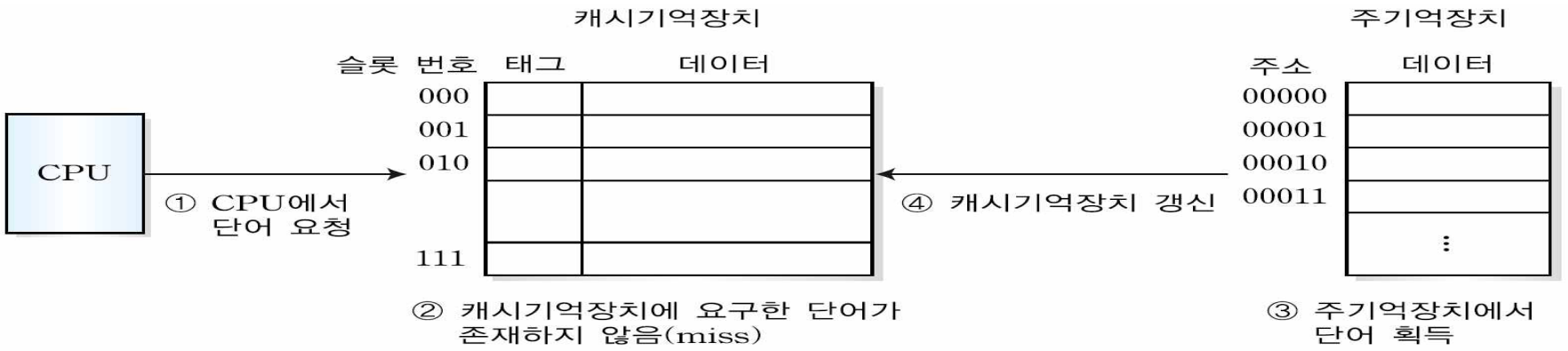


(b) 캐시기억장치

캐시기억장치와 주기억장치 사이의 정보교환 방법

❖ 사상의 개념

- 캐시기억장치에서의 인출이 실패하게 되면 캐시기억장치의 일부분은 주기억장치로 옮기고 주기억장치에서 필요한 정보를 캐시기억장치에 기억시키는 정보 교환이 이루어진다.
- 이렇게 주기억장치와 캐시기억장치 사이에서 정보를 옮기는 것을 사상(mapping)이라고 한다.



- **캐시기억장치의 사상 방법**에는 다음과 같이 대표적인 세 가지 방법이 존재
  - **직접 사상(direct mapping)**
  - **연관 사상(associative mapping)**
  - **집합 연관 사상(set-associative mapping)**

캐시기억장치의 교체 알고리즘

❖ 교체 알고리즘(Replacement Algorithms)

캐시기억장치에서 슬롯(Slot)은 한 블록이 저장되는 장소

- 캐시기억장치의 모든 슬롯이 데이터로 채워져 있는 상태에서 실패일 때
  - 캐시기억장치의 특정 슬롯에서 데이터를 제거하고 주기억장치에서 새로운 데이터 블록을 가져와야 한다.
  - 캐시기억장치의 어느 슬롯 데이터를 제거하는가를 결정하는 방식이 교체 알고리즘이다.

- **교체 알고리즘에는 대표적인 종류**
  - **LRU : 최소 최근 사용 알고리즘**
  - **LFU : 최소 사용 빈도 알고리즘**
  - **FIFO : 선입력 선출력 알고리즘**
  - **RANDOM : 랜덤**
    - 캐시기억장치에서 임의의 블록을 선택하여 삭제하고 새로운 블록으로 교체하는 방식이다. 그러나 효율성을 보장하기가 어렵다.

캐시기억장치의 교체 알고리즘

❖ 교체 알고리즘의 종류

- 최소 최근 사용(LRU, Least Recently Used) 알고리즘
  - 현재까지 알려진 교체 알고리즘 중에서 가장 효과적인 교체 알고리즘으로 집합 연관 사상에서 사용되는 방식이다.
  - CPU로의 인출이 없는 가장 오래 저장되어 있던 블록을 교체하는 방식이다.
- 최소 사용 빈도(LFU, Least Frequently Used) 알고리즘
  - 적재된 블록들 중에서 인출 횟수가 가장 적은 블록을 교체하는 방식이다.
  - 최소 최근 사용 알고리즘이 시간적으로 오랫동안 사용되지 않은 블록을 교체, 최소 사용 빈도 알고리즘은 사용된 횟수가 적은 블록을 교체하는 방식이다
- 선입력 선출력(FIFO, First In First Out) 알고리즘
  - 가장 먼저 적재된 블록을 우선적으로 캐시기억장치에서 삭제하는 교체 방식
  - 캐시기억장치에 적재된 가장 오래된 블록이 삭제되고 새로운 블록이 적재된다.
  - 구현이 용이, 시간적으로 오래된 블록을 교체하여 효율성을 보장하지 못한다.
- 랜덤(Random)
  - 캐시기억장치에서 임의의 블록을 선택하여 삭제하고 새로운 블록으로 교체하는 방식이다. 그러나 효율성을 보장하기가 어렵다.