



12. 메모리 구조

목표

- 메모리 계층을 이해하고, 캐시 메모리 및 가장 메모리의 구성을 알고 동작 방법 이해



메모리 계층

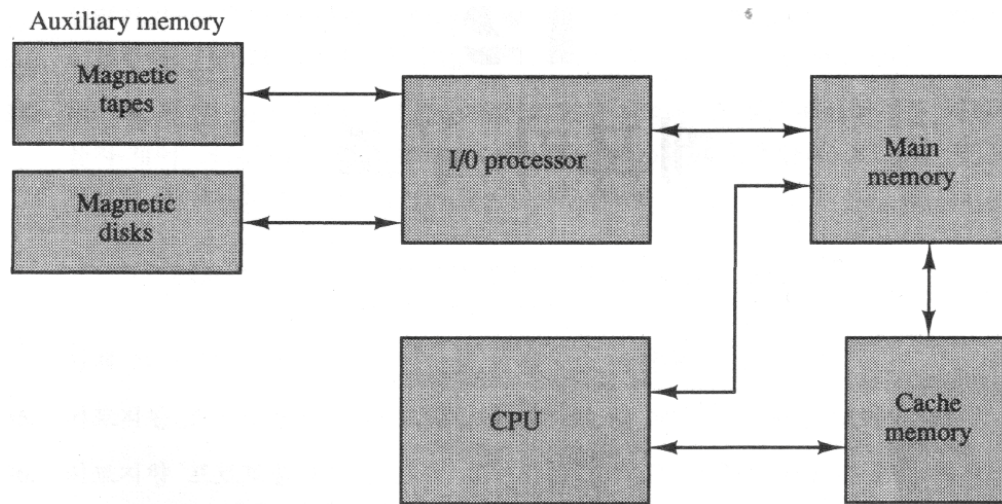


기억장치

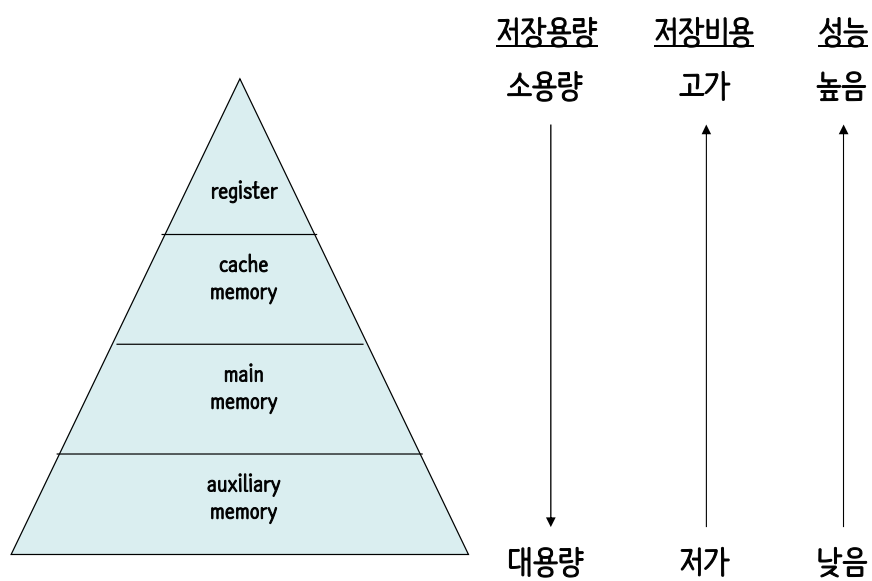
- 프로그램 및 데이터 저장
- 종류
 - 주기억장치
 - 보조기억장치



메모리 계층



메모리 계층

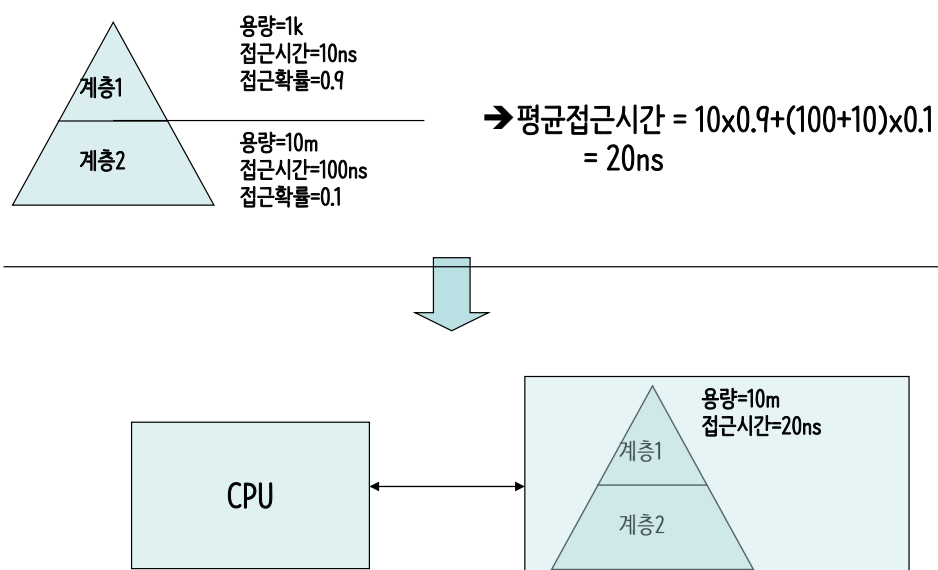


메모리 계층 목적

- 전체 메모리 시스템의 가격 최소화 및
- 평균 접근 시간 최소화



메모리 계층 목적



주기억 장치

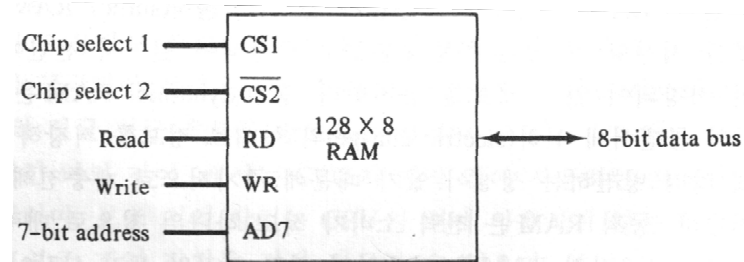


주기억장치

- 프로그램 및 데이터 저장
- 비교적 대용량/고속 메모리
- 종류
 - RAM
 - ROM

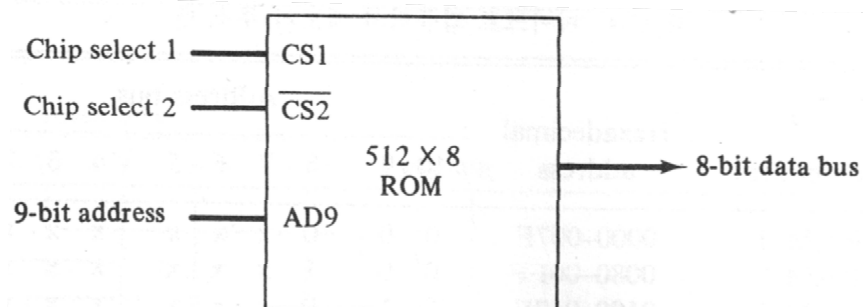


RAM



CS1	$\overline{\text{CS2}}$	RD	WR	Memory function	State of data bus
0	0	×	×	Inhibit	High-impedance
0	1	×	×	Inhibit	High-impedance
1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High-impedance

ROM



보조 기억 장치



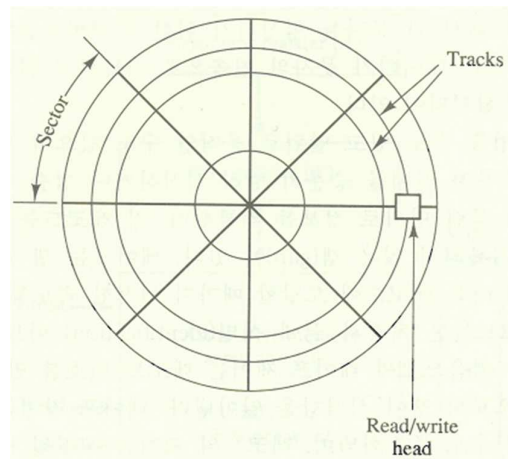
보조기억장치

- 대용량의 데이터 저장.
- 종류
 - 자기 디스크
 - 자기 테이프 등...



자기 디스크

- 저장 구조
 - 자화된 평판 이용
- 저장 단위
 - 섹터(들)
- 접근 시간
 - 탐색시간(seek time)
 - 회전지연시간(rotational delay)
 - 전송시간(transfer time)

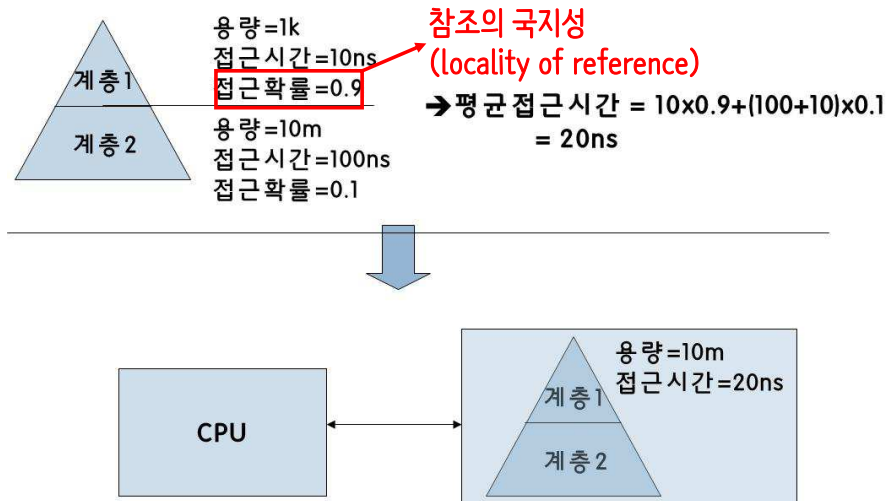


캐시 메모리



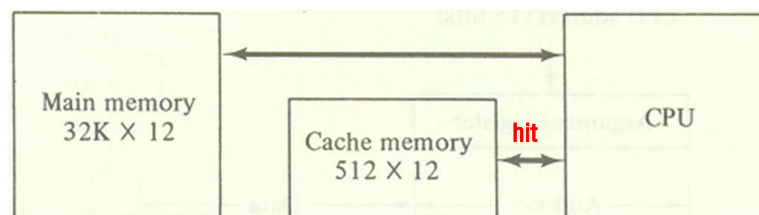
캐시 메모리

- 자주 참조되는 워드들을 고속의 메모리에 저장
- ➔ 평균 메모리 접근 시간 감소.



캐시 메모리



- 구성



- 동작 원리

- CPU의 메모리 워드 주소 생성.
- 해당 주소를 캐시 메모리에서 탐색.
- 캐시 메모리 내 존재하면 CPU에게 워드 전송
- ➔ Hit
- 존재하지 않으면 메모리가 CPU에게 전송.
- ➔ Miss

캐시 적중율

- hit ratio  
- 메모리 참조의 총 수 대비 캐시 메모리 존재의 수.
- > 0.9
- 평균 접근 시간에 영향을 줌.



캐시 메모리 사상(mapping)

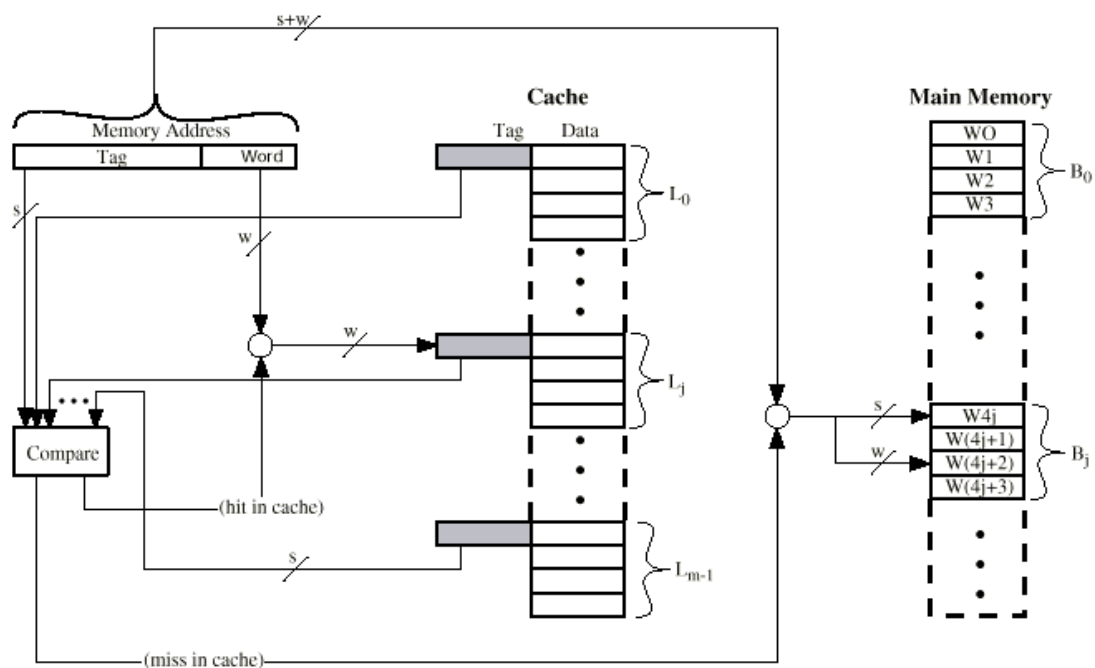
- WHY mapping?
 - 메모리 블록의 수 \gg 캐시 메모리 블록의 수
- 종류
 - associative mapping
 - direct mapping
 - set-associative mapping



Associative Mapping

- 메모리 블록은 임의의 캐시 블록에 매핑

Associative Mapping



Associative Mapping

- 특징
 - 매핑의 유연성.
 - 하드웨어 복잡도 증가



Direct Mapping

- 메모리 블록은 정해진 캐시 블록에 매핑.
 - 한 캐시 블록에 여러 메모리 블록들이 매핑
- ➔ 매핑된 메모리 블록의 구분 : **태그** 이용



Direct Mapping

- 매핑의 일반화

- 캐시 메모리 2^k 워드
- 메모리 2^n 워드
- ➔ 태그 : $(n-k)$ 비트, 인덱스 : k 비트



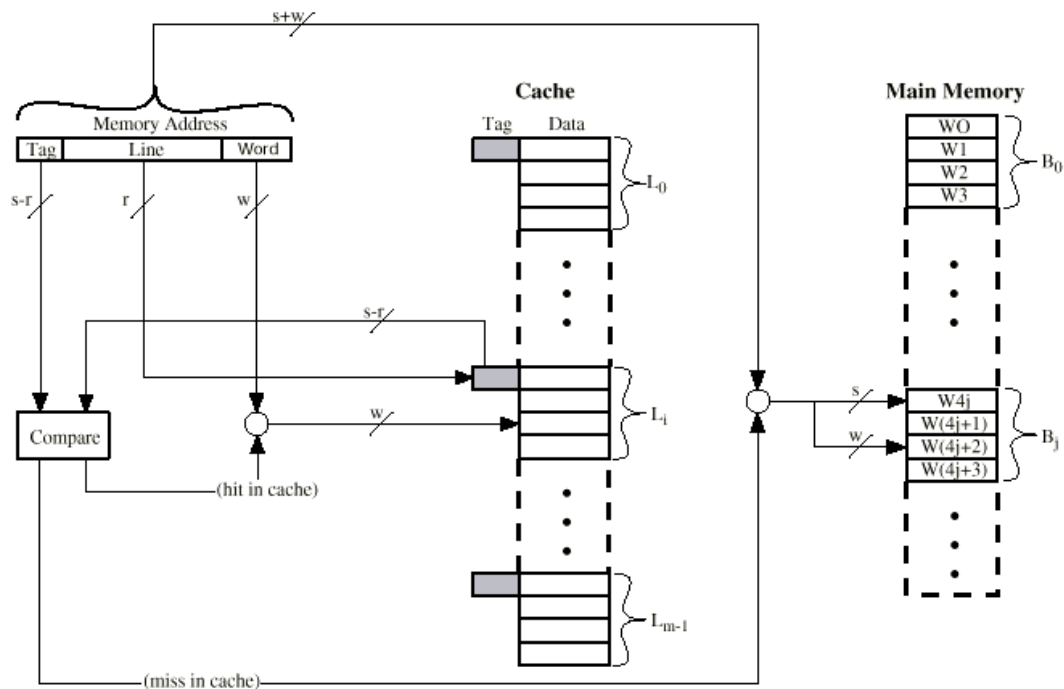
Direct Mapping

- 특징

- 간단한 매핑 방법.
- 같은 캐시 블록으로 매핑되는 메모리 블록들이 반복적으로 사용될 경우 성능 저하.



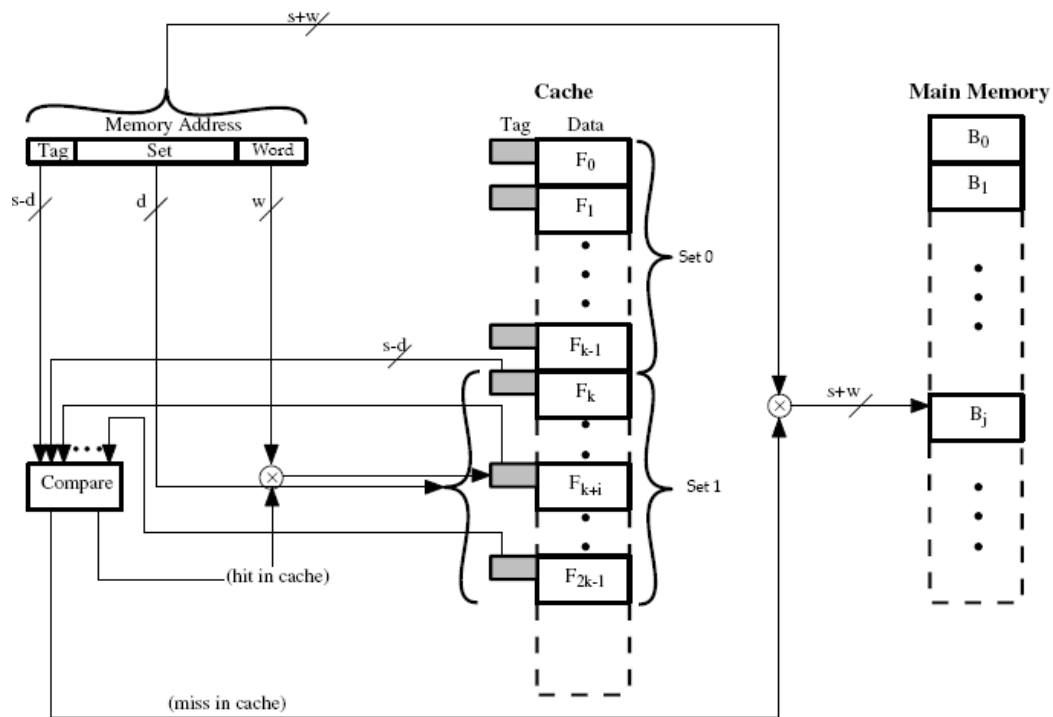
Direct Mapping



Set-Associative Mapping

- 직접 매핑과 연관 매핑을 조합.
- 다수의 캐시 블록을 세트(set)으로 묶음.
 - 세트 크기 = 1 → 직접 매핑
 - 세트 크기 = 캐시 블록의 수 → 연관 매핑
- 메모리 블록은 정해진 set으로 매핑.
- set 내에서는 임의의 캐시 블록에 매핑.

k-way Set-Associative Mapping



교체 알고리즘

- 모든 캐시 블록이 사용 중 & 캐시 미스
➔ 새로운 메모리 블록이 매핑되어야 함.
- 어떤 캐시 블록을 교체?
 - LRU (Least Recently Used)
 - FIFO (First In First Out)
 - LFU (Least Frequently Used)
 - Random

쓰기 정책

- 캐시 블록이 수정된 경우 언제 메모리 블록을 수정할 것인가?
- 정책
 - Write-through
 - Write-back



쓰기 정책 – Write-Through

- cache와 메모리를 동시에 update.
- 메모리 블록은 항상 유효함(valid).
- 메모리 접근 증가.
- 간단 & 많이 사용.



쓰기 정책 – Write-Back

- cache만 update.
- 메모리 블록은 무효화함(invalidate).
- 교체될 때 메모리 블록 update.
- 메모리 접근의 최소화.



Q&A

