메모리 계층 구조

배경

먼저, 메모리 계층 구조가 어느 흐름에서 나왔는지 확인한다.

메모리와 CPU 연결 구조

구성

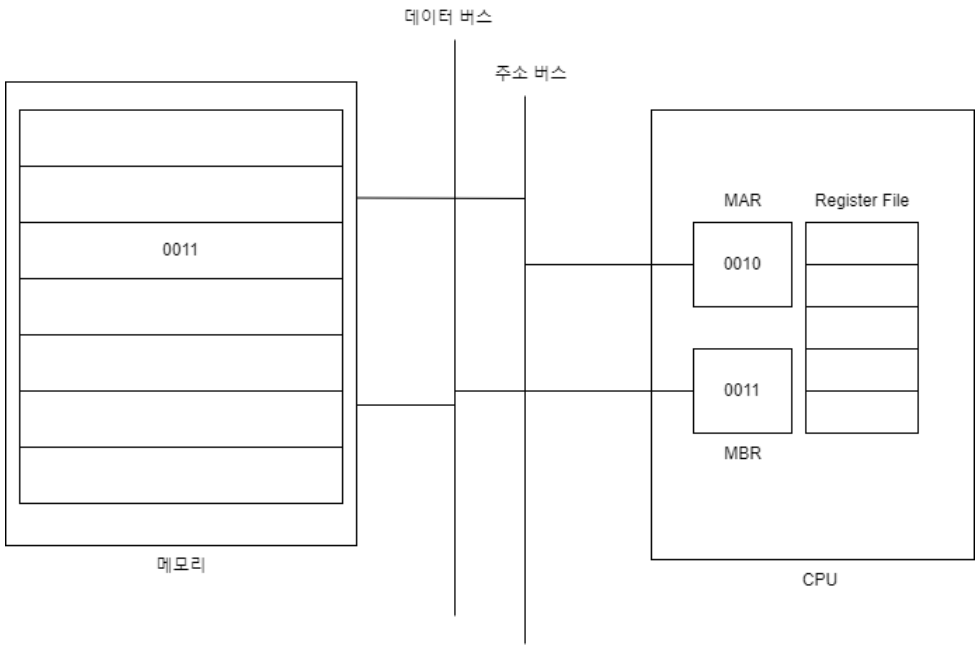
연결 구조 구성은 다음과 같다.

- CPU 의 레지스터
  - 메모리 주소 레지스터 (MAR) : 메모리 관리자가 접근할 메모리 주소 저장소
  - 메모리 버퍼 레지스터 (MBR) : 메모리 관리자가 접근할 데이터 저장소
  - 범용 레지스터 파일 (register file) : 프로세서 내부에 저장된 고속 기억 장치의 집합
- 메모리
- 버스 : 컴퓨터 구성 요소들이 통신하기 위한 배선 집합
  - 데이터 버스 : 데이터 전용 버스
  - 주소 버스 : 주소 전용 버스

과정

위의 구성을 통해, 메모리에서 읽기 요청 (Reg[3] ← M[2]) 은 다음과 같이 수행된다.

1. MAR = 0010 저장 후, 주소 버스로 메모리에 보낸다.
2. 메모리 읽기 전용 제어 신호를 보낸다
3. 메모리 0010 번지에 0011 값을 읽어서 MBR = 0011 저장한다.
4. 레지스터 파일에 저장한다.



성능 개선

메모리는 **속도, 용량, 가격** 면에서 모두 충족시켜야 한다.

특히, 속도가 CPU보다 메모리가 느리므로, 메모리의 속도 향상이 컴퓨터 성능에 큰 영향을 끼친다.

이를 위해, 다음과 같은 해결책들이 있다.

1. **[대역폭 개선]** 광폭 버스 메모리

   데이터 버스의 폭을 넓혀서 한번 접근할 때 많은 데이터를 전송할 수 있도록 한다.

2. **[병행 처리]** 교차 메모리

   메모리를 다수의 메모리 뱅크로 분할하여, 병행 접근을 가능하게 한다.

3. **[특수 메모리 개발]** 연관 메모리

   고속 탐색 분야에서 사용되는 특별한 메모리를 개발한다.

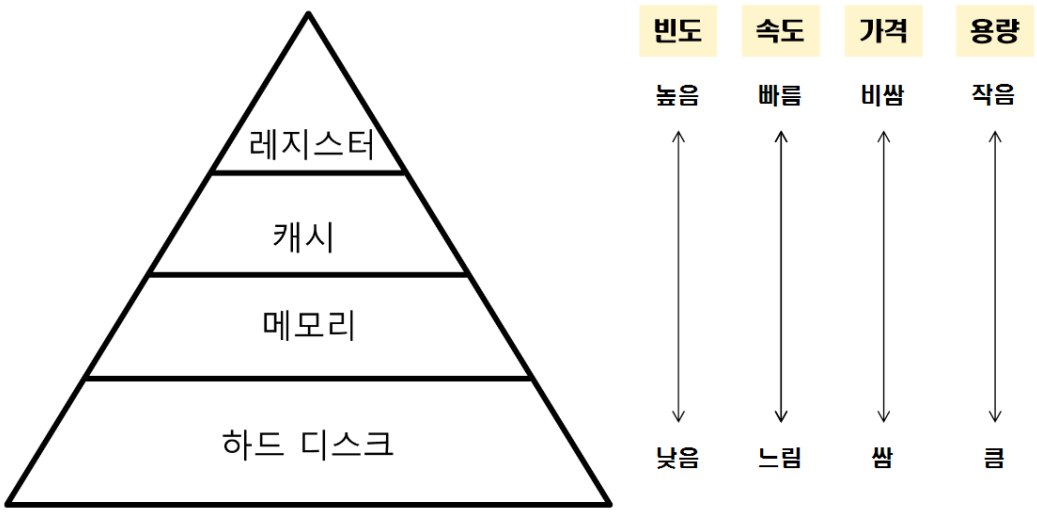
4. **[메모리 설계]** 메모리 계층 구조

   저렴하고 효율적인 기억 장치를 구성

메모리 계층 구조

메모리 계층 구조란 CPU와 메모리 속도 차이를 극복하기 위해 속도, 용량이 다른 다수의 계층으로 구성된 메모리 구조이다.

즉, 소용량이고 빠른 메모리는 CPU와 가깝게 배치하고, 대용량과 느린 메모리는 CPU와 멀리 배치한다.



각 계층의 특징

	용량	접근 시간	관리자	하위 계층과 전송 단위
레지스터	< 1KB	< 0.3ns	컴파일러	워드
캐시	< 16MB	<15ns	하드웨어	블록
메인 메모리	< 16GB	< 200ns	운영체제	페이지
보조 기억 장치	>100GB	< 5,000,000ns	운영체제, 사용자	

과정

만약 CPU로부터 데이터 요청 시, 데이터를 찾아가는 과정은 다음과 같다.

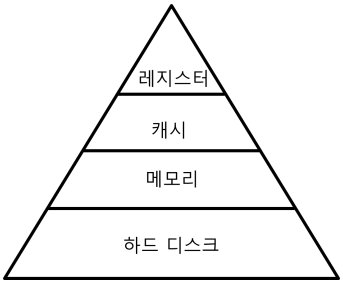
1. CPU의 데이터 요청
2. 레지스터에서 데이터 확인

3. 데이터 부재시, 메모리에서 데이터 확인
4. 데이터 부재시, 하드디스크에서 데이터 확인

- 만약 데이터가 있다면, 아래 계층의 데이터를 복제하여 상위 계층에게 전달한다.

설계 결정

메모리 계층 구조를 설계 시, 다음 3가지를 결정해야 한다.



1. 배치 방법 : 하위에서 상위로 데이터 전송 시, 상위 계층의 저장 위치 결정
    - direct mapping : 정해진 곳에만 배치한다.
    - fully associated mapping : 어디에나 배치한다.
    - set associative mapping : 정해진 집합 내에 어디에나 배치한다.

2. 교체 방법 : 상위 계층에서 하위 계층으로 추출할 데이터 결정
    - 무작위 알고리즘 : 무작위로 추출할 데이터를 선정한다.
    - FIFO 알고리즘 : 처음 올라온 데이터를 선정한다.
    - LRU 알고리즘 : 가장 오랫동안 사용되지 않은 데이터를 선정한다.
        - 단점 : 시간을 기록하는 데이터 공간 필요
    - LFU 알고리즘 : 가장 적게 사용되었던 데이터를 선정한다.
        - 단점 : 호출 횟수를 기록하는 데이터 공간 필요
    - NUR 알고리즘 : LRU + LFU. 단, 시간과 빈도의 유무만 비트로 표현하여 데이터 공간 확보

3. 쓰기 방법 : 상위 계층에서 데이터 변경 시, 하위 계층의 데이터 갱신 시기
    - 즉시 쓰기 : 하위 계층에 즉시 반영한다.
        - 장점 : 데이터 일관성 유지가 쉽다
        - 단점 : 트래픽이 있다. 속도가 느리다.
    - 나중 쓰기 : 버퍼에 모아두었다가 한번에 하위 계층에 반영한다.
        - 즉시 쓰기의 장단점과 반대

캐시 메모리와 메인 메모리의 차이

	캐시 메모리	메인 메모리
목적	cpu와 메모리의 속도 차를 줄이기 위해, **속도 향상**	작업의 필수 장치로서, **프로그래머에게 확장된 메모리 공간, 메모리 보호, 공유 등을 제공**
관리자	하드웨어	운영체제
구성	1. 데이터 메모리 2. 태그 메모리 3. 태그 비교기	1. 사용자 영역 - 물리 주소 공간 2. 운영체제 영역 - 주소 변환 테이블
동작	1. cpu의 명령어와 데이터 존재 여부를 요청 2.	3. cache miss 시, 데이터 요청 4. **page hit OR page**

	cache hit OR cache miss (2-1. TLB hit OR TLB miss)	**fault**
배치	“블록 배치 방법”	“페이지 사상 방법”
교체	“블록 교체 방식”	“페이지 교체 알고리즘”
갱신	“블록 갱신 방식” 즉시 쓰기 나중 쓰기	“페이지 쓰기” **나중 쓰기** - 보조 기억 장치는 많이 느리기 때문
성능 개선	적중률 = cache hit 수 / 전체 메모리 참조 수	1. 테이블 크기가 크다 ⇒ 페이지 테이블의 개선 2. 메모리 2번 접근 ⇒ 페이지 테이블의 캐시화(TLB)