6.Memory_&_Cache_Memory

6.Memory_&_Cache_Memory

6.1. RAM의 특징과 종류

6.1.1. 특징

- 실행할 프로그램의 명령어와 데이터를 저장
- CPU가 직접 접근해 복사/저장 할 수 있다.
- volatile memory로, 전원이 꺼지면 저장 내용이 날아간다.
- 이와 다르게, non-volatile memory는 비휘발성 저장 장치(보조기억장치)로, HDD, SSD, CD-ROM, USB가 대표적이다.
- 보조기억장치는 보관할 대상을 저장한다.

6.1.2. RAM의 용량과 성능

- 용량이 크다면 여러 프로그램을 동시에 실행하는 데 유리하다.
- 단, 용량과 프로그램의 실행 속도가 비례하지는 않는데, 많은 프로그램을 미리 올려놓더라도 보조기억장치에 접근하는 시간을 줄이는 데는 한계가 있기 때문이다.

6.1.3. 종류

6.1.4. DRAM

- Dynamic RAM의 준말.
- 저장된 데이터가 동적으로 변하는(사라지는) RAM.
- 즉, 시간이 지나면 저장된 데이터가 점차 사라지며, 주기적으로 데이터를 재활성화 (재저장) 해야한다.
- 소비전력이 비교적 낮고, 저렴하며, 집적도가 높기 때문에 대용량으로 설계하기 용이하다.

6.1.5. SRAM

- Static RAM.
- 저장된 데이터가 변하지 않는다. ~~전원 공급이 없다면 데이터가 휘발된다.~~
- 일반적으로 DRAM 보다 빠르다.
- 집적도가 낮고, 소비 전력이 크며, 가격이 비싸기 때문에 일반적으로 DRAM이 더 많이 쓰인다.
- 단, 특수한 경우, 캐시 메모리 처럼 대용량일 필요는 없지만 속도가 빨라야 하는 저장장치에 사용된다.

6.1.6. SDRAM

- Synchronous Dynamic RAM
- 클럭 신호와 동기화 된 DRAM.
- 클럭에 맞춰 CPU와 정보를 주고받을 수 있다.

6.1.7. DDR SDRAM

- Double Data Rate SDRAM
- 최근 가장 흔히 사용된다.
- 대역폭을 높여 속도를 향상시킨다.
- 두 배의 대역폭으로 한 클럭 당 두 번씩 CPU와 데이터를 주고받을 수 있다. 때문에 속도도 두 배다.
- 한 클럭당 하나씩 데이터를 주고받는 SDRAM은 Single Data Rate SDRAM이라 한다.
- DDR2, DDR3, DDR4 등은 대역폭이 2배씩 증가한다.

6.2. 메모리 주소 공간

6.2.1. 물리 주소와 논리 주소

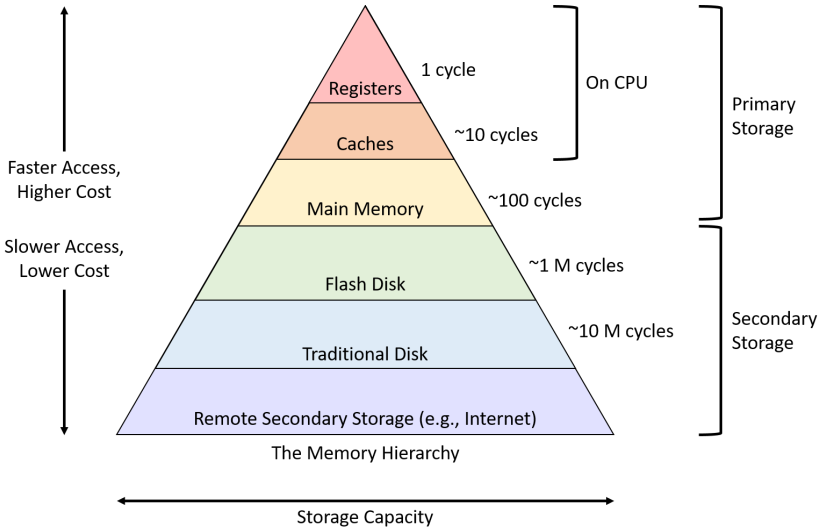
- 메모리에 저장된 정보는 계속 변한다. 프로그램이 적재되고 삭제되는 과정에서 사용하는 주소는 매번 달라진다.
- 물리 주소 Physical address:
  - 메모리 하드웨어가 사용하는 주소
  -
- 논리 주소 Logical address:
  - CPU와 실행 중인 프로그램이 사용하는 주소
  - 실행 중인 프로그램 각각에 부여된, 0번지부터 시작하는 주소
- Memory Management Unit[1]
  - 논리-물리 주소 간 변환을 수행하는 하드웨어
  - CPU가 발생시킨 논리 주소에 베이스 레지스터 값을 더해 논리 주소를 물리 주소로 변환
  - 현재 베이스 레지스터에 15000이 저장되어있고 CPU가 발생시킨 논리 주소가 100번지라면 메모리의 15100번지에 적힌.
  - 베이스 레지스터는 프로그램의 첫 물리 주소. 논리 주소는 프로그램 시작점에서의 offset.

6.2.2. 메모리 보호 기법

- 프로그램의 논리 주소 영역을 벗어난 명령어들은 실행될 수 없다.
  - 예로, 한 프로그램의 영역이 1000 번지에서 1999 번지라면, 해당 프로그램에서 1001 번지의 데이터를 수정하거나 삭제할 수 없는 것.
- 한계 레지스터 limit reg:
  - 논리 주소의 최대 크기를 저장
  - 하나의 프로그램에서 다른 프로그램의 접근을 차단
  - 즉, 프로그램의 물리 주소 범위는 베이스 레지스터 ~ 베이스 레지스터 + 한계 레지스터 미만이 된다.
  - 과정은 다음과 같다.
    1. 논리 주소 요청
    2. 논리 주소 범위 확인
       1. 범위 벗어나는 요청 ⇒ trap (software interrupt)
          - 시스템 제어권을 OS에 넘기고 interrupt 원인(addressing error)을 파악
          - 프로그램을 Abort 하거나 다른 제제를 걺
       2. 범위 내 요청 ⇒ base reg + offset
       3. 물리적 메모리 내용 읽기 후 전달

6.3. 캐시 메모리

6.3.1. 저장 장치 계층 구조



- Memory hierarchy
- CPU에 가까운 저장장치일 수록 빠르고, 용량이 작고, 비싸다.
- 아래 계층으로 내려갈 수록 용량은 크지만 처리가 느리다.

6.3.2. Cache memory

- 느린 메모리에 빈번히 접근할 때 발생하는 자원 낭비를 완화하기 위해 cache memory를 사용
- CPU의 연산 속도와 메모리 접근 속도의 차이를 완화.
- 사용할 일부 데이터를 미리 가져옴으로써 CPU가 메모리에 자주 접근하는 것을 막는다.
- L1, L2, L3 캐시가 있으며, 숫자가 낮을 수록 CPU와 가깝다. 일반적으로 L1, L2는 CPU 코어 내부에, L3는 외부에 존재한다.
- 멀티 코어 프러세서에서, L1, L2 캐시는 각 코어마다 고유한 캐시 메모리로, L3 캐시는 여러 코어가 공유하는 형태로 사용한다.
- Split cache도 있으며, 명령어만 저장하는 L1I cache, 데이터만 저장하는 L1D cache로 분리하기도 한다.

6.3.3. 참조 지역성 원리

- CPU가 사용할 법한 대상을 예측하여 저장
- 예측한 데이터가 실제 활용될 경우 cache hit레 한다. 그 반대는 cache miss.
- cache hit ratio (H)는 "캐시 적중 횟수 / 전체 기억장치 참조 횟수"로 구한다.
- 일반적으로 적중률은 85 ~ 95%이며, locality of reference, principle of locality (참조 지역성 원리)에 따라 가져올 데이터를 결정한다.
  - 참조 지역성 원리는 CPU의 다음 두 가지 경향을 이용한다.
    1. 최근 접근했던 메모리 공간에 다시 접근하려는 경향.
    2. 점근한 메모리 공간 근처를 접근하려는 경향.
  - `1.`의 경우, 변수로 저장한 값은 프로그램이 실행되는 동안 자주 사용되기 때문에 발생한다. 이렇게 최근 접근했던 메모리 공간에 재접근 하는 경향을 **시간 지역성(temporal locality)**이라 한다.
  - `2.`의 경우, 프로그램을 적재할 때 관련 데이터들을 주변에 모아 적재하기 때문에 발생한다. 이처럼 접근한 메모리 공간 근처를 접근하려는 경향을 **공간 지역성 (spatial locality)**이라 한다.

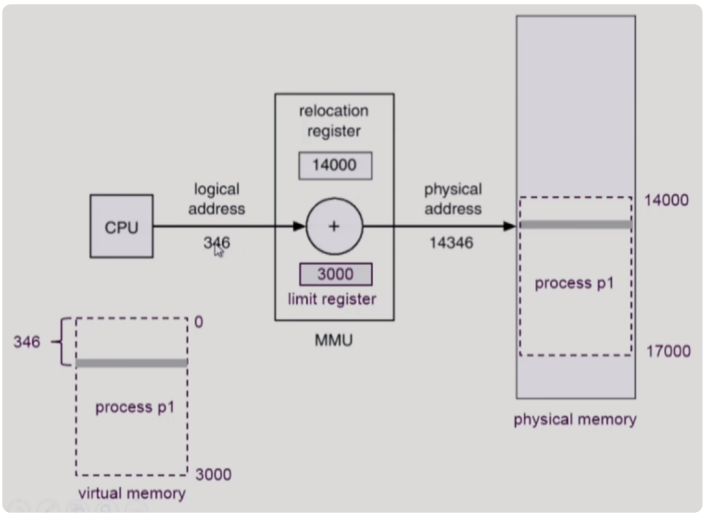
---

1. Operating_System #9.2.1. MMU

9.2.1. MMU

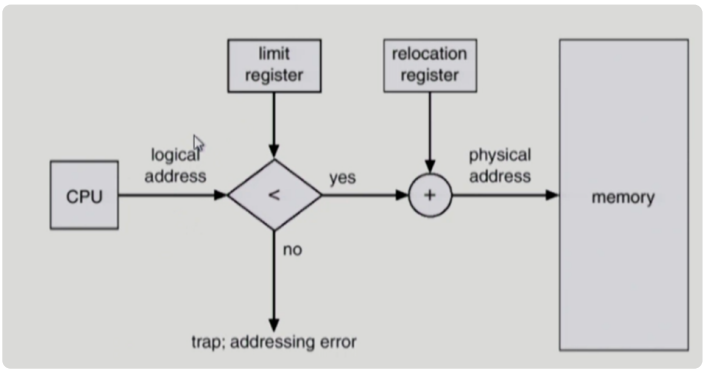
- Memory-Management Unit
  - logical address를 physical address로 매핑해 주는 **Hardware device**hardware-memory-management
- MMU scheme
  - 사용자 프로세스가 CPU에서 수행되며 생성해내는 모든 주소값에 대해 base register (=relocation register)의 값을 더 한다.
- user program
  - logical address만을 다룬다
  - 실제 physical address를 볼 수 없으며 알 필요가 없다.

9.2.1.1. Dynamic Relocation



- logical address(346)은 MMU를 지나 physical address(14346)로 변환된다.
- MMU는 relocation register와 limit register를 이용해 주소를 변환한다.
- 기본적으로는 logical address가 들어오면 relocation register의 값을 더해 physical address를 확인한다.

9.2.1.2. Hardware Support for Address Translation



- 운영체제 및 사용자 프로세스 간 메모리 보호를 위해 사용하는 레지스터
- Relocation register (= base register): 접근할 수 있는 물리적 메모리 주소의 최소값
- Limit register: 논리적 주소의 범위

1. 논리주소 요청
2. 논리주소의 범위 확인
   1. 범위를 벗어나는 요청 ⇒ trap (software interrupt)
      - 시스템 제어권을 운영체제에 넘기고 interrupt 원인(addressing error)을 파악
      - 프로그램을 abort 하거나 다른 제제를 걺.
   2. 범위 내의 요청 ⇒ base register의 값을 더함
   3. 물리적 메모리 내용 읽기 후 전달