**Chapter 6 메모리와 캐시 메모리**

***6-1*** **RAM의 특징과 종류**

**<RAM의 특징>**

* 휘발성 저장 장치이다.
  + 휘발성 저장 장치: 전원을 끄면 저장된 내용이 사라지는 저장 장치
  + 비휘발성 저장 장치: 전원이 꺼져도 저장된 내용이 유지되는 저장 장치

ex) 하드디스크, SSD, CD-ROM, USB 메모리

* 비휘발성 저장 장치인 보조기억장치에는 보관할 대상을 저장하고 휘발성 저장 장치인 RAM에는 실행할 대상을 저장
* CPU가 실행하고 싶은 프로그램이 보조기억장치에 있다면 이를 RAM으로 복사하여 저장한 뒤 실행

**<RAM의 용량과 성능>**

* RAM의 용량이 작다면 보조기억장치에서 실행할 프로그램을 가져오는 일이 잦아 실행 시간이 길어짐
* RAM 용량이 충분히 크다면 보조기억장치에서 많은 데이터를 가져와 미리 RAM에 저장할 수 있어 많은 프로그램을 동시에 실행하는데 유리함
* RAM 용량이 커지면 프로그램 실행 속도가 어느 정도 증가하는 것은 맞지만, 용량이 필요 이상으로 커졌을 때 속도가 그에 비례하여 증가하지는 않음

**<RAM의 종류>**

**DRAM**

* Dynamic RAM의 준말 -> 저장된 데이터가 동적으로 변하는(사라지는) RAM
* 데이터의 소멸을 막기 위해 일정 주기로 데이터를 재활성화(다시 저장)해야 함
* 일반적으로 메모리로써 사용하는 RAM -> 소비 전력이 비교적 낮고, 집적도가 높기 때문에 대용량으로 설계하기 용이

**SRAM**

* Static RAM의 준말 -> 저장된 데이터가 변하지 않는 RAM
* 데이터를 재활성화할 필요가 없고 속도도 DRAM보다 빠름
* DRAM보다 집적도가 낮고, 소비 전력도 크며, 가격도 더 비싸기 때문에 메모리로 잘 사용되지는 않는다.
* 대용량으로 만들어질 필요는 없지만 속도가 빨라야 하는 저장 장치에서 사용됨

ex) 캐시메모리

**SDRAM**

* Synchronous Dynamic RAM의 준 말로 DRAM의 발전된 형태
* 클럭 신호와 동기화 됨 -> 클럭 타이밍에 맞춰 CPU와 정보를 주고 받음

**DDR SDRAM**

* Double Data Rate SDRAM
* 최근 가장 흔히 사용되는 RAM
* 대역폭을 넓혀 속도를 빠르게 만든 SDRAM
  + 대역폭: 데이터를 주고받는 길의 너비
* SDRAM에 비해 두 배의 대역폭으로 한 클럭당 두 번씩 CPU와 데이터를 주고 받을 수 있음 -> 전송 속도가 두 배가량 빠름
* SDR SDRAM(Single Data Rate SDRAM)이라고 부르기도 함

**DDR2 SDRAM**

* DDR SDRAM보다 대역폭이 두배 넓은 SDRAM

**DDR3 SDRAM**

* DDR2 SDRAM보다 대역폭이 두 배 넓고, SDR SDRAM보다 대역폭이 여덟 배 넒음

**DDR4 SDRAM**

* 최근에 흔히 사용하는 메모리로 SDR SDRAM보다 열여섯 배 넓은 대역폭을 가짐

***6-2*** **메모리의 주소 공간**

**<물리 주소와 논리 주소>**

메모리에 저장된 정보는 시시각각 변함 -> 새롭게 실행되는 프로그램이 적재되고 실행이 끝난 프로그램은 삭제 됨, 같은 프로그램을 실행하더라도 실행할 때마다 적재되는 주소가 달라질 수 있음

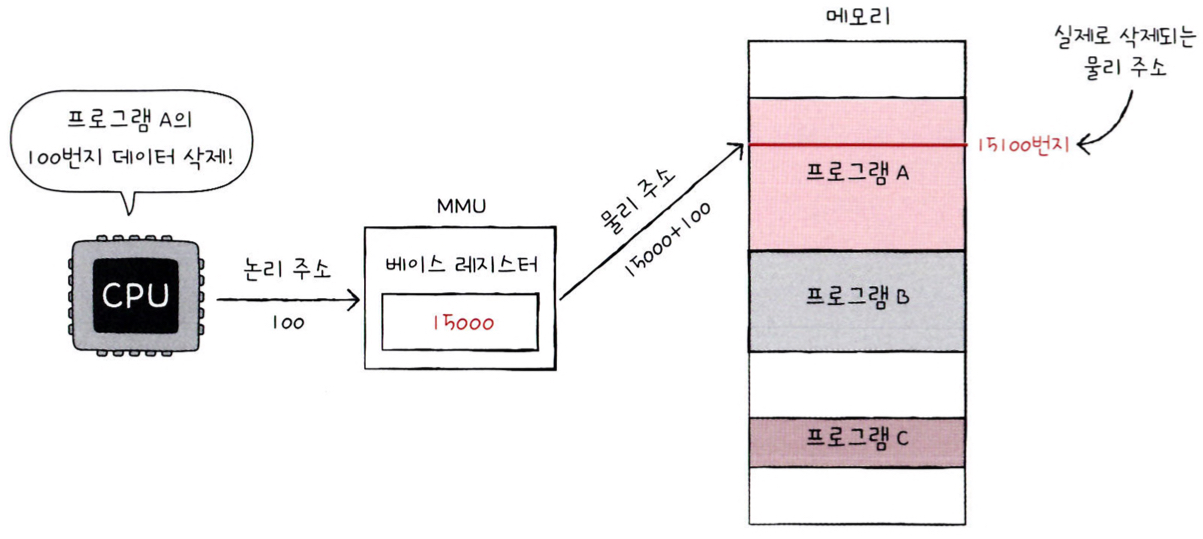
**물리 주소**: 정보가 실제로 저장된 하드웨어상의 주소

**논리 주소**: CPU와 실행 중인 프로그램이 사용하는 주소로 실행 중인 프로그램 각각에게 부여된 0번지부터 시작되는 주소

모든 프로그램은 각자를 위한 논리 주소를 가짐 -> 프로그램마다 같은 논리 주소가 있을 수 있음

CPU는 논리 주소를 받아들이고, 해석하고, 연산함 -> CPU가 메모리와 상호작용하려면 논리 주소와 물리 주소 간의 변환이 이루어져야 함

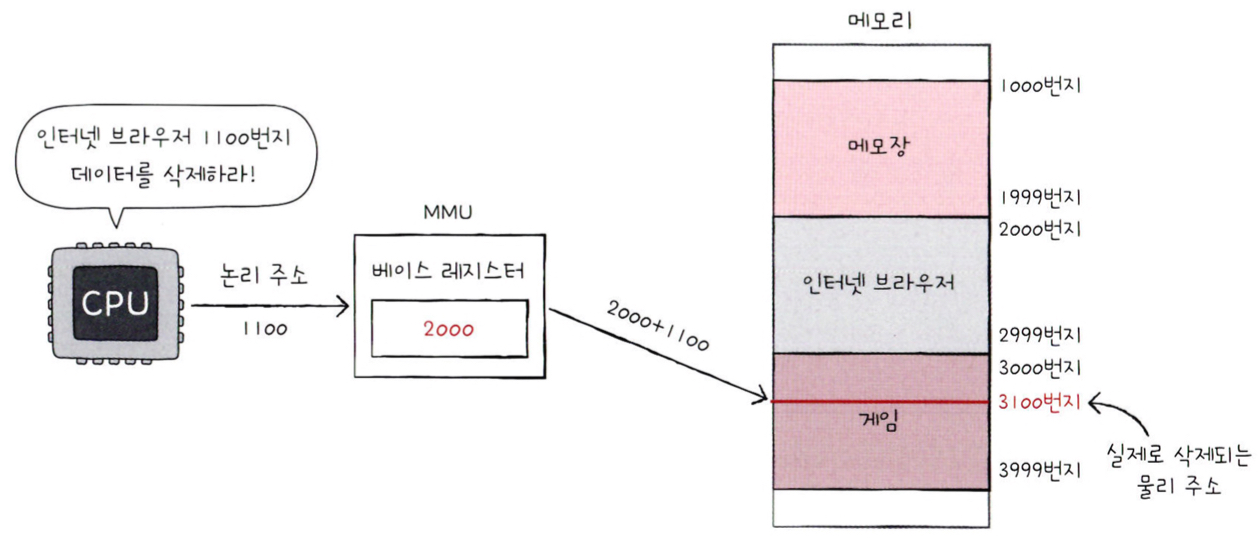
**메모리 관리 장치(MMU)**: CPU와 주소 버스 사이에 위치하여 CPU가 발생시킨 논리 주소에 베이스 레지스터 값을 더하여 논리 주소를 물리 주소로 변환



**베이스 레지스터**: 프로그램의 가장 작은 물리 주소 (프로그램의 첫 물리 주소)를 저장

**논리 주소**: 프로그램의 시작점으로부터 떨어진 거리

**<메모리 보호 기법>**

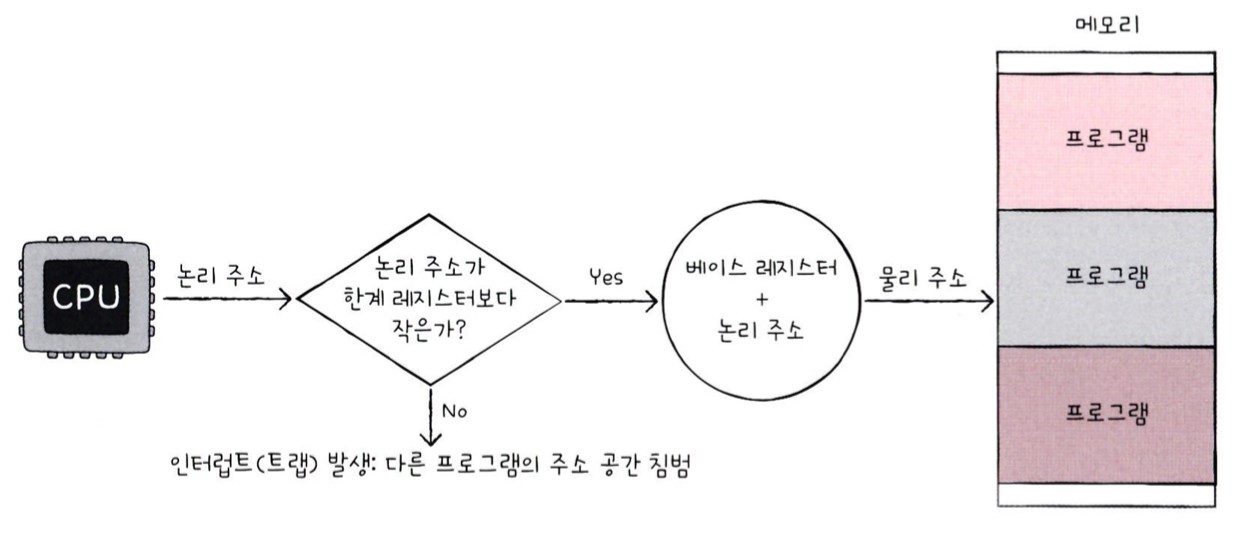


논리 주소 범위를 벗어난 명령어 실행을 방지하고 실행 중인 프로그램이 다른 프로그램에 영향을 받지 않도록 보호할 방법이 필요

-> **한계 레지스터**라는 레지스터가 담당

**한계 레지스터**: 논리 주소의 최대 크기를 저장

프로그램의 물리 주소 범위: 베이스 레지스터 값 이상, 베이스 레지스터 값 + 한계 레지스터 값 미만



-> CPU는 메모리에 접근하기 전에 접근하고자 하는 논리 주소가 한계 레지스터보다

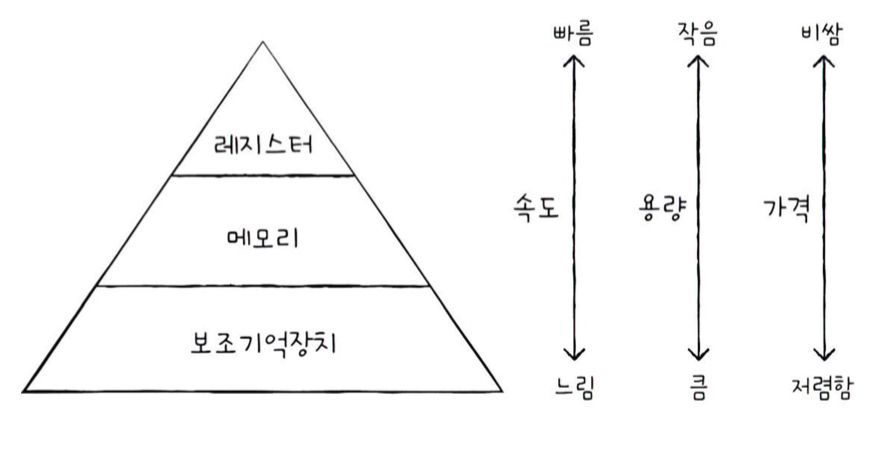
작은지를 항상 검사

-> 만약 한계 레지스터보다 높은 논리 주소에 접근하려고 하면 인터럽트(트랩)를 발생시켜 실행을 중단

***6-3*** **캐시 메모리**

**<저장 장치 계층 구조>**

컴퓨터가 사용하는 저장 장치들은 CPU에 얼마나 가까운가를 기준으로 계층적으로 나타낼 수 있음



-> 낮은 가격대의 대용량 저장장치를 원한다면 느린 속도를 감수해야 함

-> 빠른 메모리를 원한다면 작은 용량과 비싼 가격을 감수해야 함

**<캐시 메모리>**

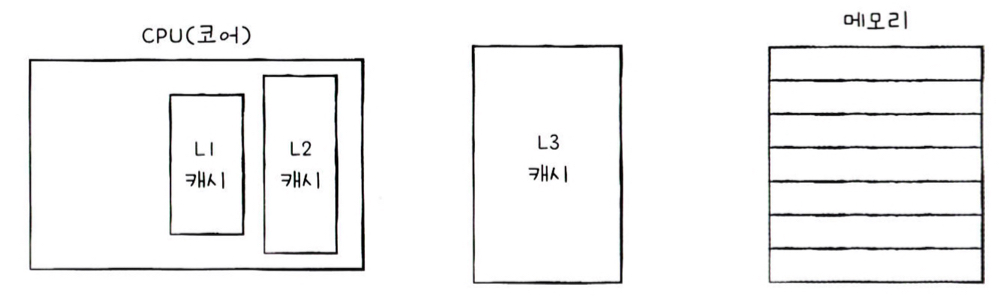
CPU가 메모리에 접근하는 속도는 레지스터에 접근하는 속도보다 느림 -> 그럼에도 CPU는 프로그램을 실행하는 과정에서 메모리에 빈번히 접근해야 함 -> 그래서 등장한 저장 장치가 **캐시 메모리**

**캐시 메모리**

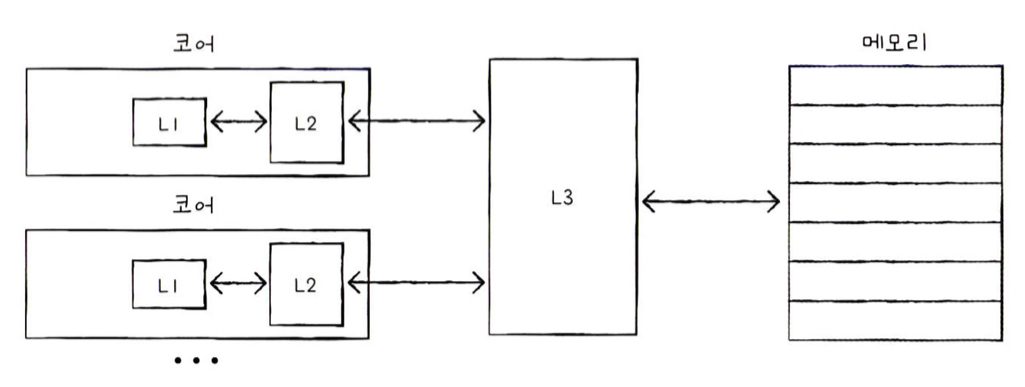
* CPU와 메모리 사이에 위치함
* 레지스터보다 용량이 크고 메모리보다 빠른 SRAM 기반의 저장 장치
* CPU의 연산 속도와 메모리 접근 속도의 차이를 조금이나마 줄이기 위해 탄생
* 메모리에서 CPU가 사용할 일부 데이터를 미리 캐시 메모리로 가지고 와서 활용

-> 데이터로의 접근 시간을 줄일 수 있음

* 컴퓨터 내부에는 여러 개의 캐시 메모리가 있음 -> L1캐시, L2캐시, L3캐시



CPU가 메모리 내에 데이터가 필요하다고 판단하면 우선 L1 캐시에 해당 데이터가 있는지를 알아보고, 없다면 L2, L3 캐시 순으로 데이터를 검색



멀티 코어 프로세서에서 일반적으로 L1,L2 캐시는 코어마다 고유한 캐시 메모리로 할당되고, L3 캐시는 여러 코어가 공유하는 형태로 사용

**<참조 지역성 원리>**

캐시 메모리는 CPU가 사용할 법한 대상을 예측하여 저장

**캐시 히트**: 자주 사용될 것으로 예측한 데이터가 실제로 들어맞아 캐시 메모리 내 데이터가 CPU에서 활용될 경우

**캐시 미스**: 자주 사용될 것으로 예측하여 캐시 메모리에 저장했지만, 예측이 틀려 메모리에서 필요한 데이터를 직접 가져와야 하는 경우. 캐시미스가 자주 발생하면 성능이 떨어짐

**캐시 적중률**: 캐시 히트 횟수 / (캐시 히트 횟수 + 캐시 미스 횟수)

캐시 메모리는 **참조 지역성의 원리**에 따라 메모리로부터 가져올 데이터를 결정

* **참조 지역성의 원리**: CPU가 메모리에 접근할 때의 주된 경향을 바탕으로 만들어진 원리
  + **시간 지역성**: CPU는 최근에 접근했던 메모리 공간에 다시 접근하려는 경향이 있다

ex) 변수를 여러 번 사용

* + **공간 지역성**: CPU는 접근한 메모리 공간 근처를 접근하려는 경향이 있다

ex) 워드 프로세서 프로그램을 실행할 때 워드 프로세서 프로그램이 모여 있는 공간 근처를 접근, 사용자가 입력 시에는 그 중에서도 입력 기능이 모여 있는 공간 근처를 집중적으로 접근

