메모리와 캐시 메모리

6-1 RAM의 특징과 종류

주기억장치에는 RAM, ROM

통상 우리가 말하는 메모리는 RAM을 지칭하는 경우가 많다

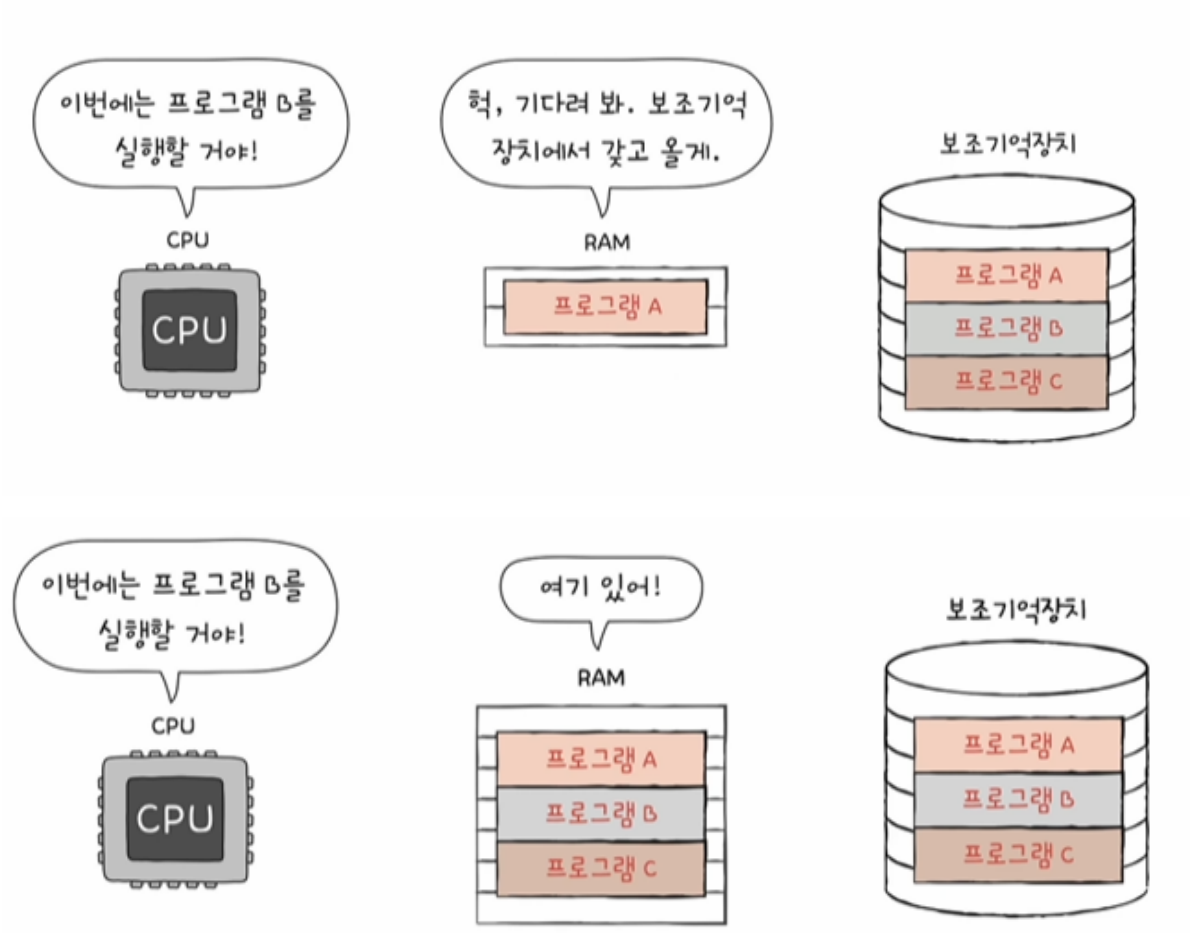
RAM의 특징

실행할 대상을 저장. -> 저장한 내용을 전원이 꺼지면 삭제가 된다. 그래서 보조기억장치에 보관할 대상을 저장

따라서 램은 **휘발성**, 보조기억장치은 **비휘발성**

RAM의 용량과 성능

용량이 크다면, 많은 프로그램들을 동시에 실행하는데에 유리



RAM의 종류

- DRAM (Dynamic RAM)
  - 저장된 데이터가 동적으로 사라지는 RAM (데이터 소멸을 막기 위해 주기적으로 재활성화 필요)
  - 상대적으로 소비전력이 낮고 저렴하고 집적도가 높아 대용량으로 설계하기 용이하여 일반적으로 사용된다.
- SRAM (Static RAM)
  - (전원이 켜져있는 동안) 저장된 데이터가 정적인 (사라지지않는) RAM
  - DRAM보다 일반적으로 더 빠름
  - 일반적으로 캐시메모리에서 사용되는 RAM
  - 상대적으로 소비전력이 높고 가격이 높고 집적도가 낮아 대용량으로 설계할 필요는 없으나 빨라야하는 장치에 사용

	DRAM	SRAM
재충전	필요함	필요 없음
속도	느림	빠름
가격	저렴함	비쌈
집적도	높음	낮음
소비 전력	적음	높음
사용 용도	주기억장치(RAM)	캐시 메모리

- SDRAM (Synchronous DRAM) - 디램의 발전된 형태
  - 클럭신호와 동기화된 DRAM
- DDR SDRAM (Double Data Rate SDRAM) - SDRAM의 두배 넓은 대역폭
  - 최근 가장 대중적으로 사용하는 RAM
  - 대역폭을 넓혀 속도를 빠르게 만든 SDRAM
  - DDR2 SDRAM - SDRAM의 네배 넓은 대역폭 (DDR SDRAM의 두배)

6-2 메모리의 주소 공간

논리주소: CPU와 실행 프로그램이 이해하는 주소

물리주소: 실제 하드웨어의 주소

CPU에는 메모리에 무엇이 저장되어있는지는 저장되어있지 않다

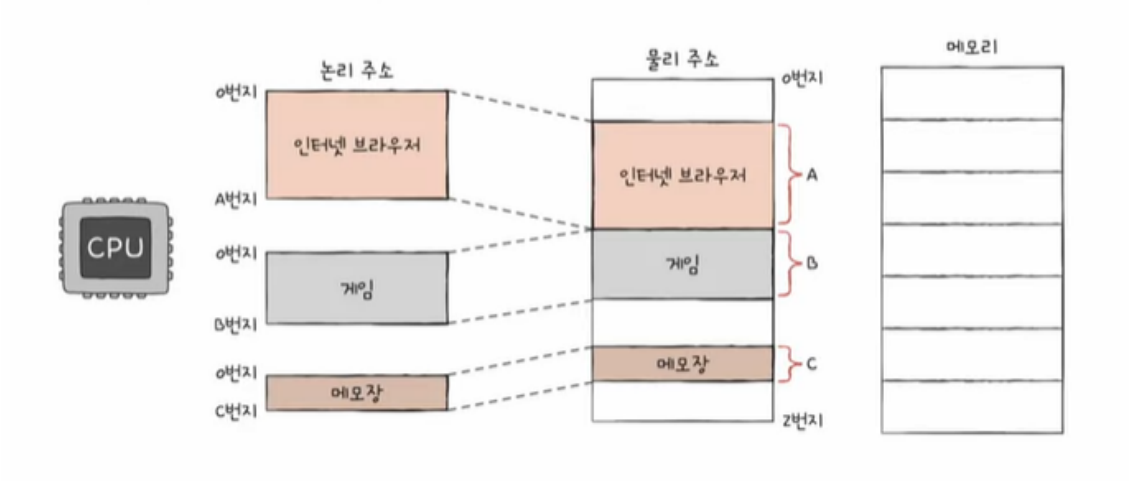
메모리에 저장된 값은 계속 변하기 때문: 실행할때마다 적제되는 주소는 다르다

물리 주소

- 메모리 입장에서 바라본 주소
- 하드웨어상의 실제 주소
- 하드웨어상의 주소이기때문에 0번지가 하나밖에 없

논리 주소

- CPU와 실행중인 프로그램의 각각의 입장에서 바라본 주소
- 실행중인 프로그램마다 0번지가 있다



물리주소와 논리주소의 변환

CPU가 메모리와 상호작용을 하려면 통역이 필요함!

그래서 MMU(메모리관리장치)라는 하드웨어에 의해 변환

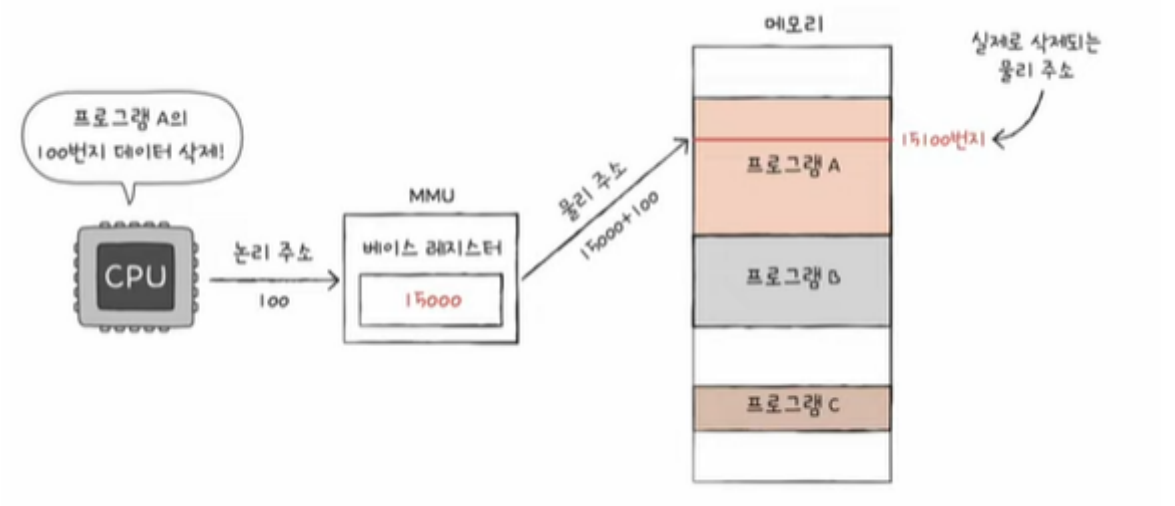
CPU -> 논리주소 -> MMU -> 물리주소 -> 메모리

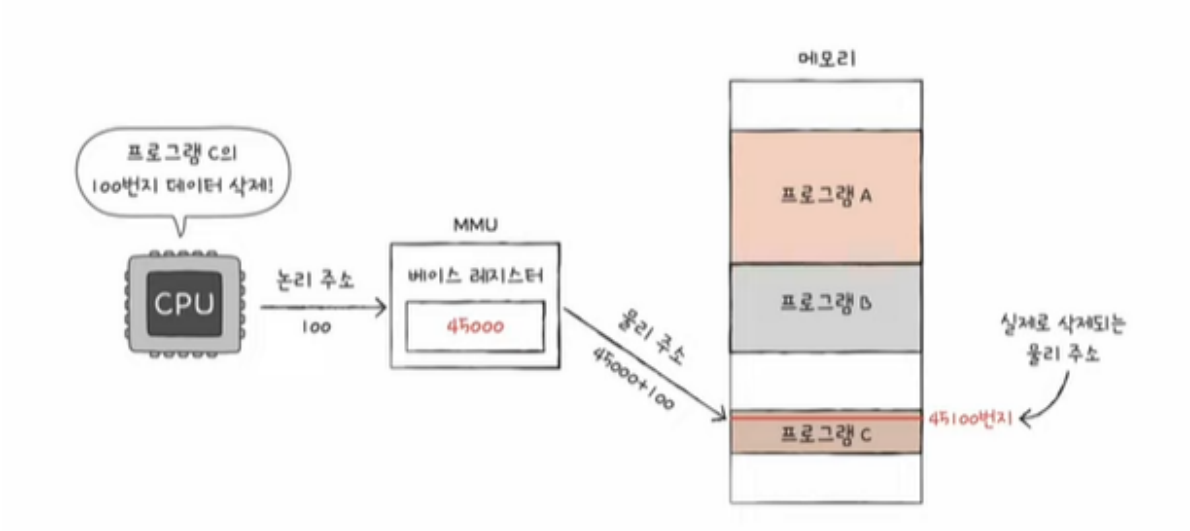
논리주소와 베이스 레지스터 값을 더하여 논리 주소를 물리 주소로 변환

베이스 레지스터: 프로그램의 기준 주소

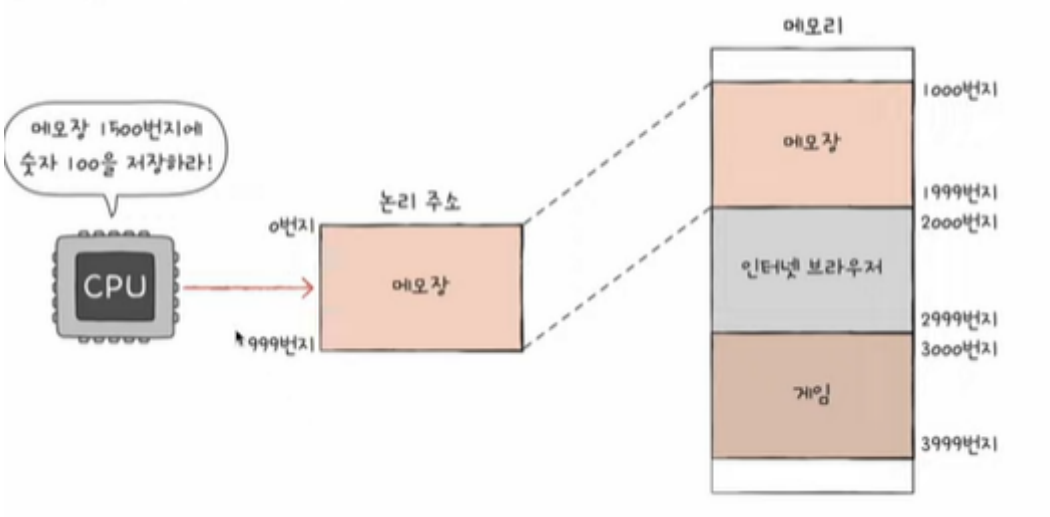
ㄴ 프로그램의 시작 주소가 담김

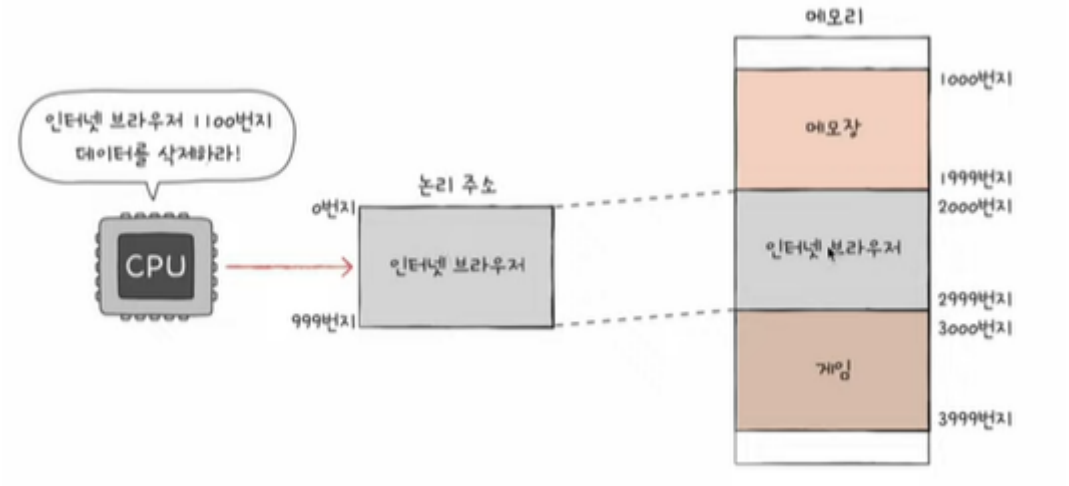
ㄴ 프로그램의 가장 작은 물리주소(프로그램의 첫 물리주소)를 저장





위험한 경우: 논리주소의 범위를 벗어난 주소를 명령하면, 엄한 데이터에 실행이 된다

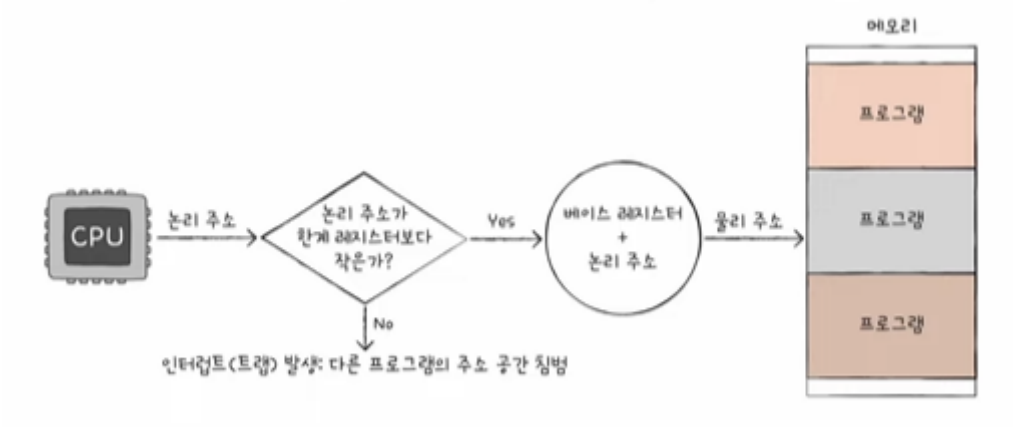




=> 이를 보호하기 위해서

메모리 보호

한계 레지스터



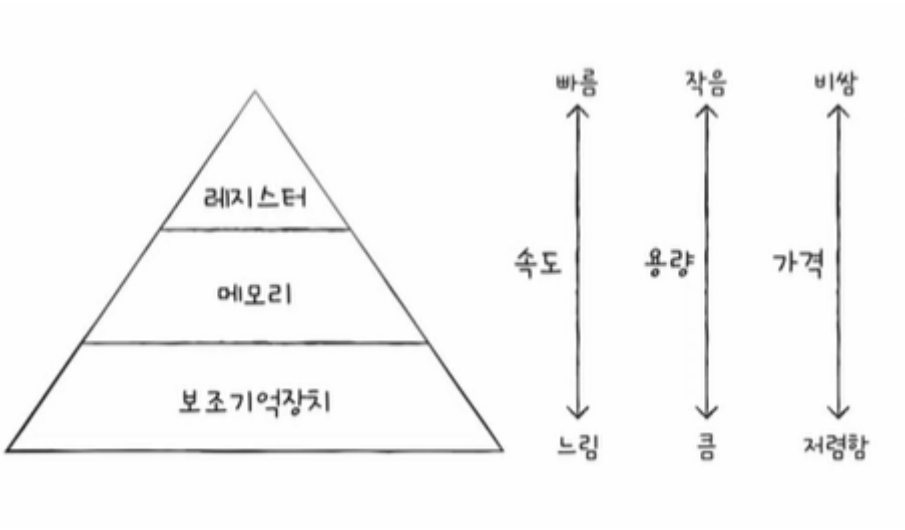
- 프로그램의 영역을 침범할 수 있는 명령어의 실행을 막음
- 베이스 레지스터가 실행중인 프로그램의 가장 작은 물리주소를 저장한다면, 한계 레지스터는 논리 주소의 최대 크기를 저장
- 베이스 레지스터 값 <= 프로그램의 물리 주소 범위 < 베이스 레지스터 + 한계 레지스터 값

6-3 캐시 메모리

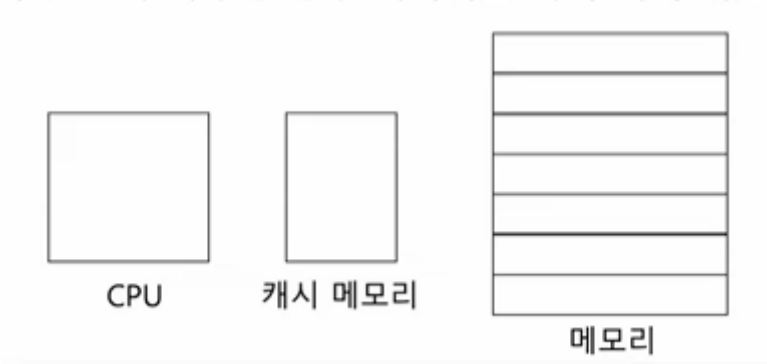
CPU가 메모리에 접근하는 시간은 CPU 연산 속도보다 느리다

- CPU와 가까운 저장장치는 빠르고, 멀리있는 저장 장치는 느리다.
- 속도가 빠른 저장 장치는 저장 용량이 작고, 가격이 비싸다.

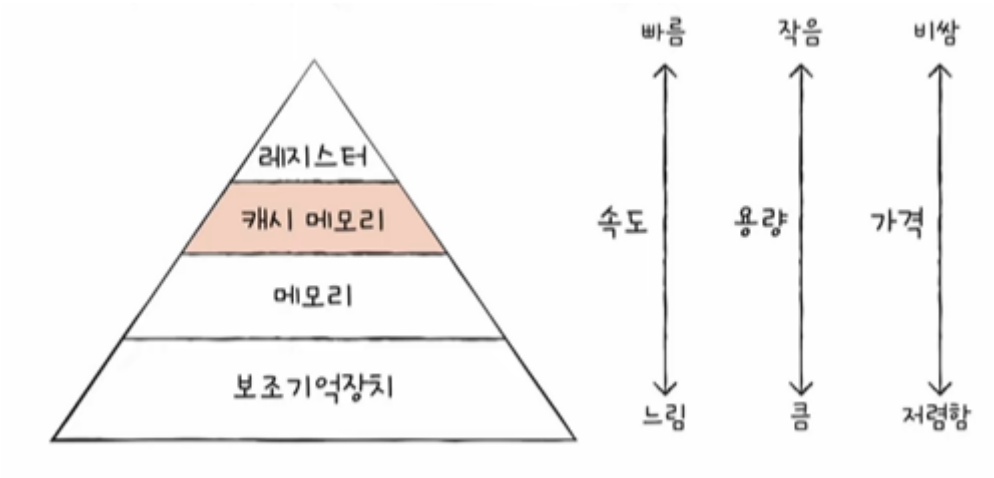
저장 장치 계층 구조: CPU에 얼마나 가까운가를 기준으로 계층적으로 나타냄



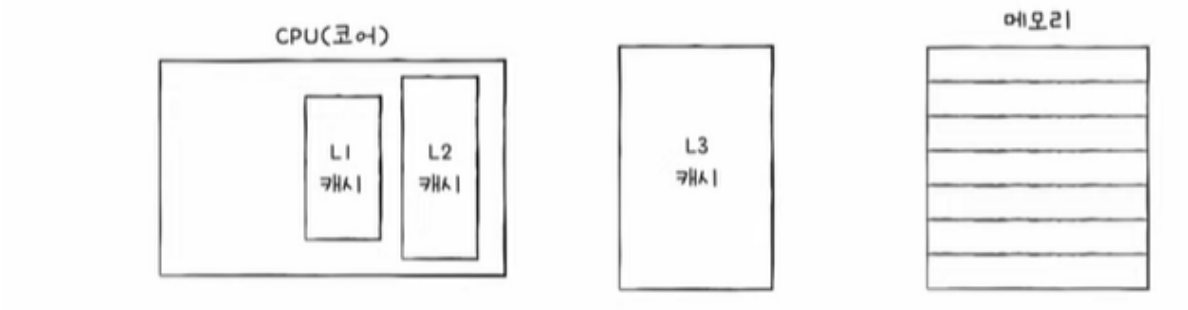
캐시메모리



- CPU와 메모리 사이에 위치
- SRAM 기반의 저장 장치
- CPU의 연산속도가 너무 빨라서, 메모리보다 조금 빠른 메모리를 만들어서 속도의 차이를 완화시키기 위해
- 물건을 사러가는 것에 비유
    - 대형마트는 내가 사고싶은 거의 모든 물건들을 가지고 있지만 왕복시간이 길다.
    - 편의점은 내가 당장 필요한 물건들만 있지만 왕복시간이 짧다.



계층적 캐시 메모리

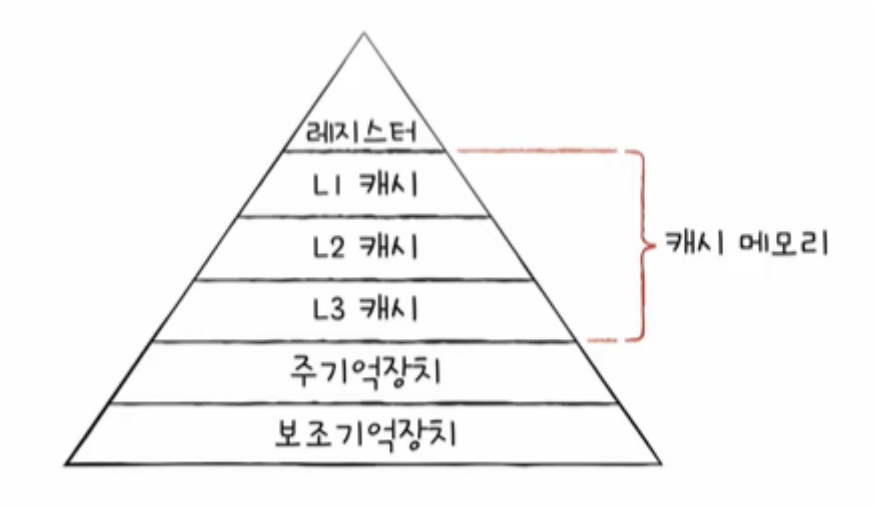


CPU 내/외부에 있는 캐시 메모리 계층적 구조: L1-L2-L3

그래도 레지스터가 제일 빠르다

분리형 캐시

L1캐시를 L1D (data), L1I (instruction) 두가지로 분리하여 관리 (더 빨리 하려고)



캐시 메모리에는 무엇을 저장하나?

CPU가 자주 사용할 법한 내용을 예측하여 저장

예측이 맞았을 경우 = 캐시 히트

예측이 틀렸을 경우 = 캐시 미스 (메모리에 접근해야겠죠?)

캐시 적중률 = 캐시 히트 횟수 / (캐시 히트 횟수 + 캐시 미스 횟수)

ㄴ 높으면 높을수록 성능이 높다~ 요즘은 못해도 80%

예측하는 방법...? 참조 지역성의 원리

- CPU가 메모리에 접근할 때의 주된 경향을 바탕으로 만들어진 원리
    1. CPU는 최근에 접근했던 메모리 공간에 다시 접근하려는 경향이 있다. - 시간 지역성
    2. CPU는 접근한 메모리 공간 근처를 접근하려는 경향이 있다. - 공간 지역성