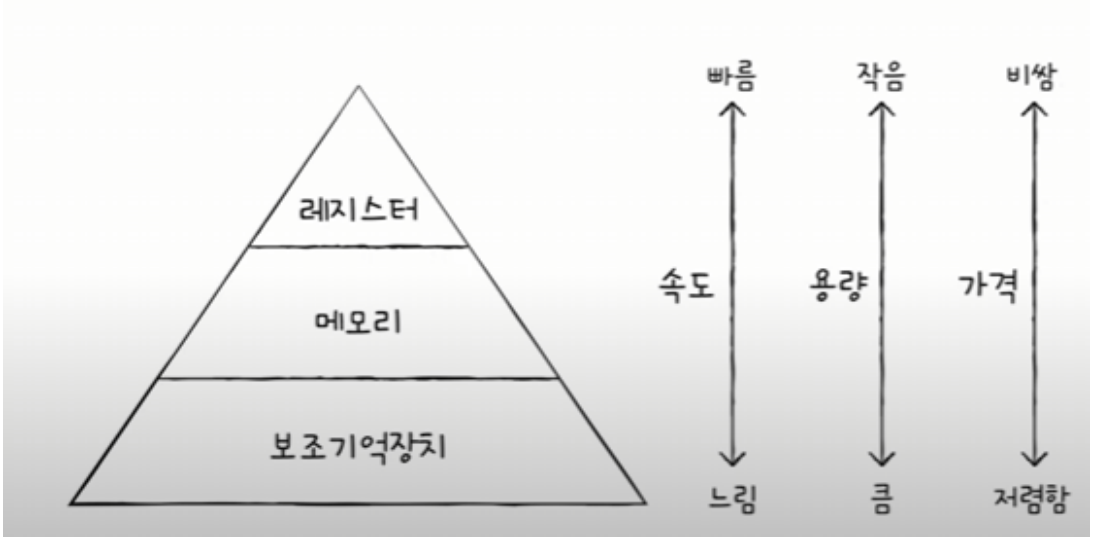
17강. 캐시 메모리

저장 장치 계층 구조 (Memory hierarchy)

컴퓨터의 저장 장치는 속도와 용량에 따라 여러 계층으로 나뉘어 사용됩니다. 이를 저장 장치 계층 구조라고 부르며, 여기서 “memory” 라는 단어는 RAM만을 뜻하지 않고, 레지스터에서 시작해 캐시, 메인 매모리 RAM, 보조기억장치(SSD, HDD)까지 포함하는 넓은 개념입니다.

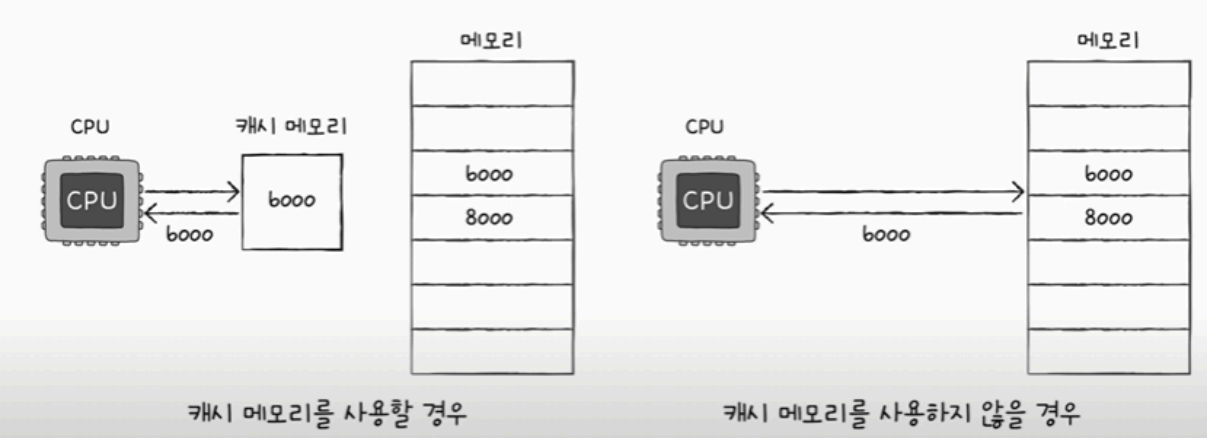


각 계층은 빠르지만 비싸거나, 느리지만 싸고 용량이 큰 등 서로 다른 특성을 가지기 때문에 이들을 적절히 조합하여 전체적인 성능과 비용의 균형을 맞추는 방식으로 설계됩니다.

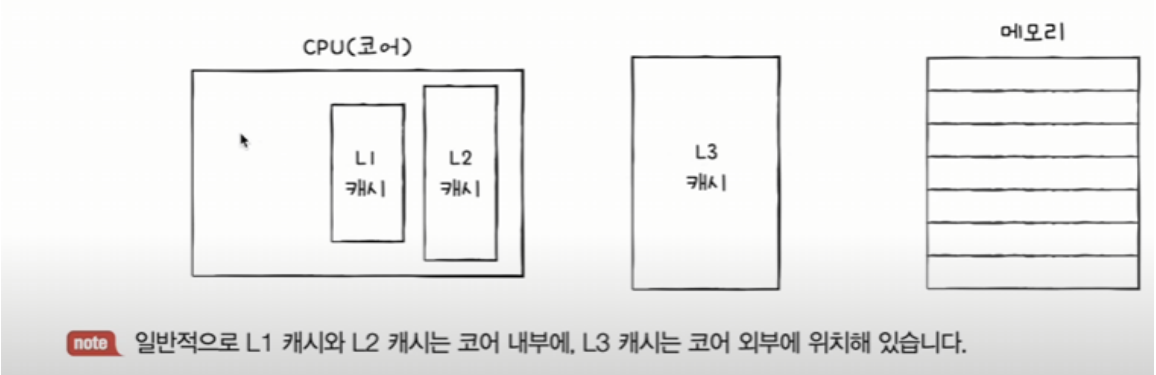
캐시 메모리

이 계층 구조 속에서 중요한 역할을 하는 것이 바로 캐시 메모리(Cache Memory) 입니다.

캐시는 CPU와 메인 메모리 사이에 위치하며, 레지스터보다는 용량이 크고 메모리보다는 훨씬 빠른 SRAM 기반의 저장 장치입니다. CPU는 매운 빠른 속도로 연산을 수행하지만 메모리에 직접 접근하는 속도는 상대적으로 느립니다. 이 속도 차이를 줄이기 위해, CPU가 앞으로 사용할 가능성이 높은 데이터를 미리 캐시 메모리에 담아두는 것입니다.



캐시 메모리는 한 단계로만 구성되지 않고, L1, L2, L3와 같이 계층적 구조를 가집니다.



가장 빠른 것은 L1 캐시로, CPU 코어 내부에 있습니다. 속도를 높이기 위해 다시 명령어 캐시 (L1I)와 데이터 캐시 (L1D)로 나누어 관리되기도 합니다.

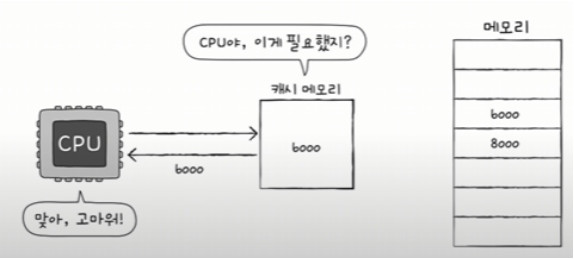
L2 캐시 역시 코어 내부에 존재하며 L1보다 용량이 큽니다. 반면 L3 캐시는 보통 CPU 코어 외부에 존재하며, 여러 코어가 공유하는 형태로 동작합니다.

정리하면 속도는 L1 > L2 > L3 순서로 빠르며, 용량은 반대입니다.

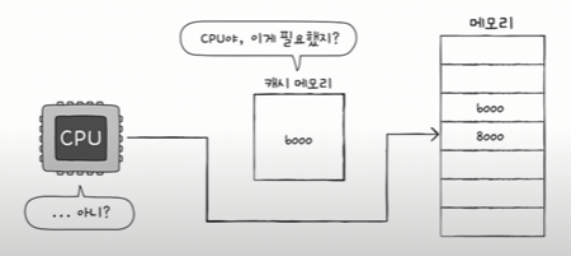
참조 지역성의 원리 (Locality of Reference)

그렇다면 캐시는 어떤 데이터를 저장해야 효율적일까요?

캐시의 용량은 메인 메모리에 비해 훨씬 작기 때문에 모든 데이터를 다음 수 없습니다. 따라서 CPU가 앞으로 자주 사용할 법한 데이터를 예측해서 담아야 합니다. 이때 예측이 맞으면 캐시 히트(hit) 라고 하며, 잘못되면 캐시 미스(miss)가 발생해 CPU는 다시 메모리에 접근해야 합니다.



캐시 히트



캐시 미스

캐시의 성능은 흔히 캐시 적중률 (캐시 히트 ÷ 전체 접근 횟수) 로 평가됩니다. 적중률이 높을수록 CPU의 효율이 올라갑니다.

이 예측 방법의 기반이 되는 개념이 참조 지역성의 원리입니다.

참조 지역성은 크게 두 가지로 나뉩니다. 첫 번째는 시간 지역성 입니다. CPU는 최근에 접근한 데이터를 곧 다시 접근할 가능성이 크다는 것입니다.

예를 들어, 반복문에서 같은 변수를 여러 번 읽는 경우가 대표적인 예시입니다.

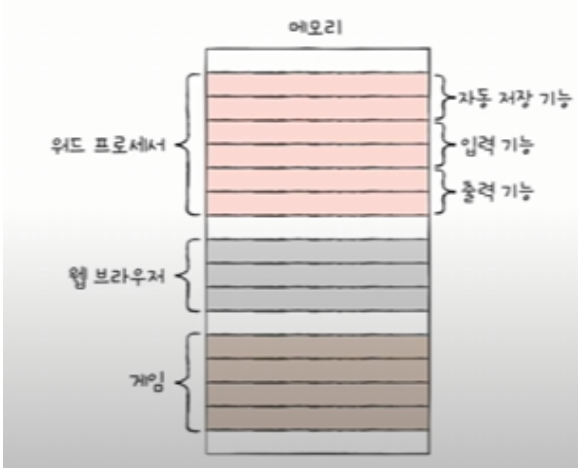
```c
#include <stdio.h>

int main(void) {
    int num = 2;

    for (int i = 1; i <= 9; i++)
        printf("%d X %d = %d\n", num, i, num * i);
    return 0;
}
```

두 번째는 공간 지역성입니다. CPU는 특정 메모리 주소를 사용하면 그 근처의 주소도 함께 사용할 가능성이 높다는 성질입니다.

예를 들어, 배열이나 구조체처럼 관련된 데이터가 인접해 저장된 경우가 있습니다.



이처럼 캐시 메모리는 CPU의 빠른 연산 속도와 메모리의 느린 접근 속도 사이의 간극을 줄이기 위해 고안된 장치로, 계층적 구조와 참조 지역성의 원리를 바탕으로 컴퓨터 성능 향상에 핵심적인 역할을 하고 있습니다.