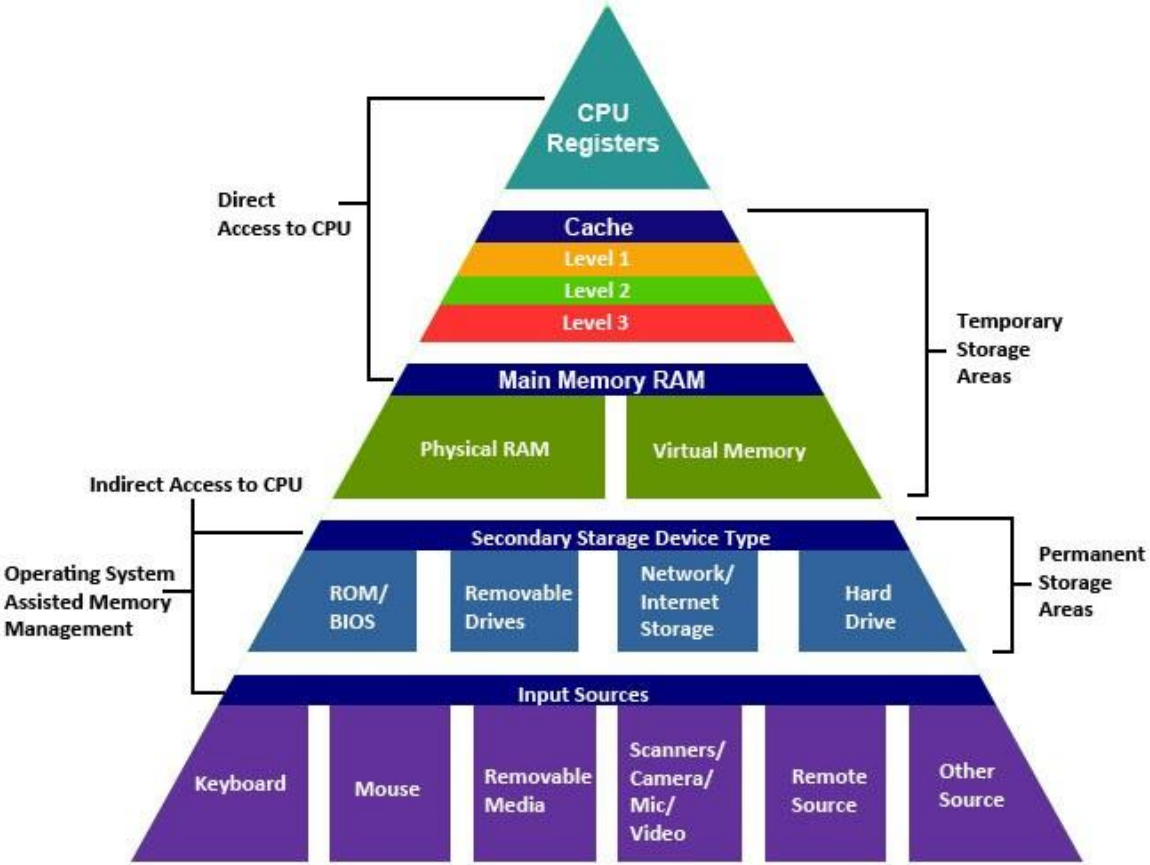
Memory Hierarchy (메모리 계층 구조)



1. register (32 bit , 64 bit)

레지스터는 CPU 안에 존재하는 저장 공간을 의미한다. 레지스터는 1bit의 정보를 저장할 수 있는 flip-flop의 집합이다. 레지스터는 데이터와 명령어를 저장하는 역할을 하며, 가장 빠른 속도로 접근 가능한 메모리이다.

레지스터는 CPU 코어 내부에도 몇 개 존재하지 않는 귀한 장치이며(100개 들어있으면 많이 들어있는 것이다), 대부분의 현대 프로세서는 메인 메모리에서 레지스터로 데이터를 옮겨와 데이터를 처리한 후 그 내용을 다시 레지스터에서 메인 메모리로 저장하는 로드-스토어 설계를 사용하고 있다.

최신 프로세서에서 레지스터는 대개 레지스터 파일로 구현되지만, 과거에는 플립플롭, 마그네틱 코어, 박막 필름 메모리 등으로 구현되기도 했다.

레지스터의 최대 처리 용량은 CPU의 처리 용량과 같은데, 32bit 컴퓨터는 2^32 까지, 64bit 컴퓨터는 2^64까지 인식이 가능하다.

2. Cache

캐시 메모리는 CPU 내부에 존재하는 저장공간을 의미한다. 보통 명령어를 저장하는 Instruction Cache와 데이터를 저장하는 Data Cache로 구분되어 있으며, 때로는 캐시 메모리도 L1 Cache, L2 Cache 등 여러 레벨으로 분할해서 세부 계층화 하기도 한다.

3. RAM

CPU와 별개인 메모리 중 최상위 메모리이자 '주기억장치'의 최하위 메모리이다. SSD, HHD와 Cache의 속도차이를 메꾸기 위해 만들어진 제품이다. PC당 수~수십GB정도로 이루어져있으며, 가정용에는 대부분 DRAM으로 구성된다. 32bit 컴퓨터의 최대 메모리 용량은 4GB이고, 64bit 컴퓨터의 경우 이론상 16EB(엑사바이트)까지 램 용량을 추가할 수 있다.

4. SSD

Solid State Drive의 줄임말. CPU, 그래픽카드, Ram은 전기적으로 동작하는 반면 하드디스크는 물리적으로 동작하기 때문에 병목현상이 걸리는 경우가 많았다. 하지만 SSD가 보급되고 가격이 낮아지면서 HDD와 가성비 경쟁이 될 정도가 되자 많은 소비자들이 SSD를 사용해 성능향상을 누리게 되었다.

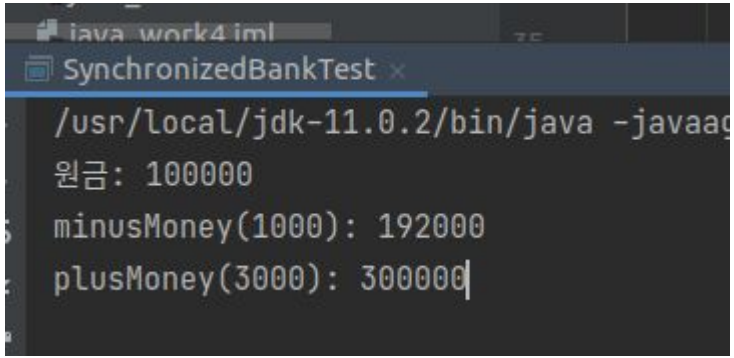
5. HDD

HDD는 비휘발성 데이터 저장소를 의미하며, 위의 기기와는 불허하는 엄청난 가성비를 가지고 있다. 1TB HDD가 5만원이 되지 않는다! 하지만 속도는 최대 200MB/s, 평균 60~150MB로 상당히 느리다.

```java
class C extends Thread {
    public void run() {
        for(int i = 0; i < 100; i++) {
            SynchronizedBankTest.sb.plusMoney(3000);
        }

        //SynchronizedBankTest.sb.plusMoney(3000);

        System.out.println("plusMoney(3000): " +
                SynchronizedBankTest.sb.getMoney());
    }
}

class D extends Thread {
    public void run() {
        for(int i = 0; i < 100; i++) {
            SynchronizedBankTest.sb.minusMoney(1000);
        }

        //SynchronizedBankTest.sb.minusMoney(1000);

        System.out.println("minusMoney(1000): " +
                SynchronizedBankTest.sb.getMoney());
    }
}
```

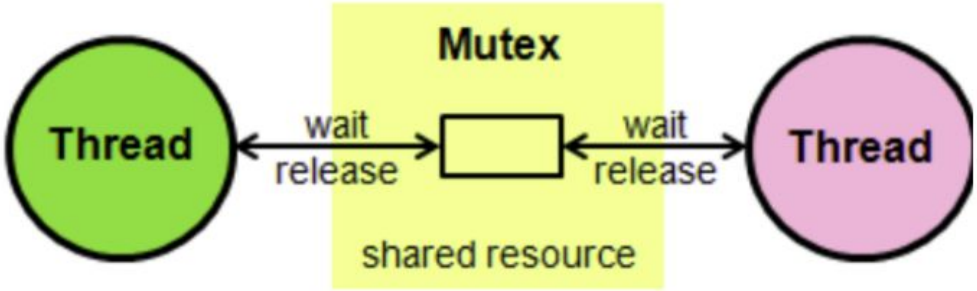
매서드 전체에 Synchronized 한 결과값이 간헐적으로 아래와 같이 나오는데 minusmoney 쓰레드가 처리도중에 cpu 경쟁권을 뺏겨서 저런값이 나오게 된건가요?

아직 Synchronized개념을 정확하게 인지하지 못하겠습니다...



```
for(int i = 0; i < 1000; i++) {
    // 현재 케이스는 실제 Critical Section에만 강제 동기화를 걸었다.
    // 그러므로 여러 태스크들이 동시에 접근할 수 있는 영역을
    // 부분적으로 안전하게 보호한 반면
    // 이전의 예제는 매서드 전체를 보호했다.
    // 그러므로 당연히 Critical Section만 방어할때에 비해 성능이 저하된다.
    synchronized (PerfSyncBankTest.psb) {
        PerfSyncBankTest.psb.plusMoney(3000);
    }
}
```

매서드 전체보단 C.S 부분만 정확히 찾아서 방어하는게 효과적이다?? 라고 해석하는게 맞나요?



-Mutex(상호배제)
Critical Section을 가진 Thread들의 running time이 서로 겹치지 않게,단독으로 실행되게 하는 기술, 혹은 스레드가 공유하는 영역을 공유 사용하지 못하도록 데이터의 영역을 보호해 주는 기능을 하는 도구.