왜 우리는 많은 양의 데이터를 사용하지 못하는 것일까?

데이터 이동 과정은 다음과 같다.

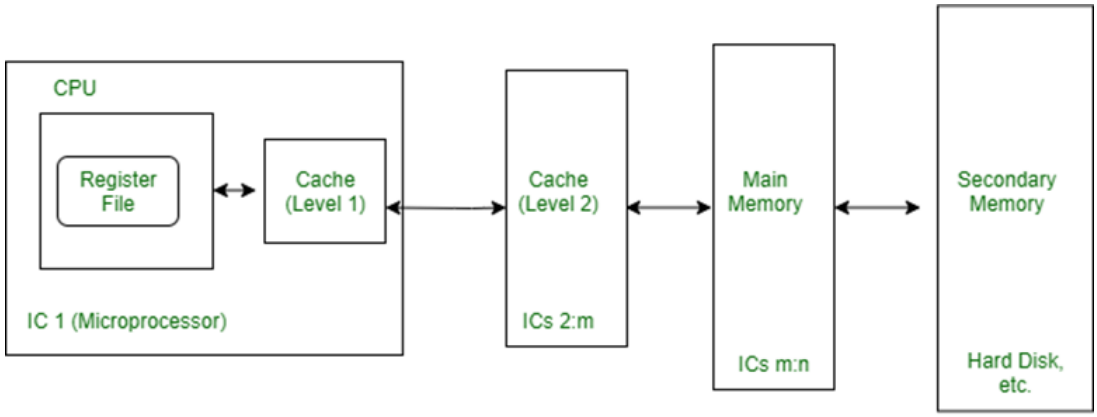
디스크에 저장되어 있는 코드와 데이터를 Memory의에 load 한다.

코드에서 할당하거나 선언한 변수들을 스택과 힙에 저장하고,

디스크에 가져온 코드와 데이터 그리고 스택과 힙을 통해서 CPU를 통해 연산한다. CPU는 변수의 주소를 통해서 Memory값에 접근하며 이를 Random access라고 한다.

연산 결과를 Memory에 저장, 프로그램 종료 이후 혹은 저장 명령이 입력 시에 Memory에 있는 값들을 Disk에 저장한다.

즉 우리가 코드를 작성하는데 있어서 Disk→ memory→ cpu의 이동 과정이 필요하다.



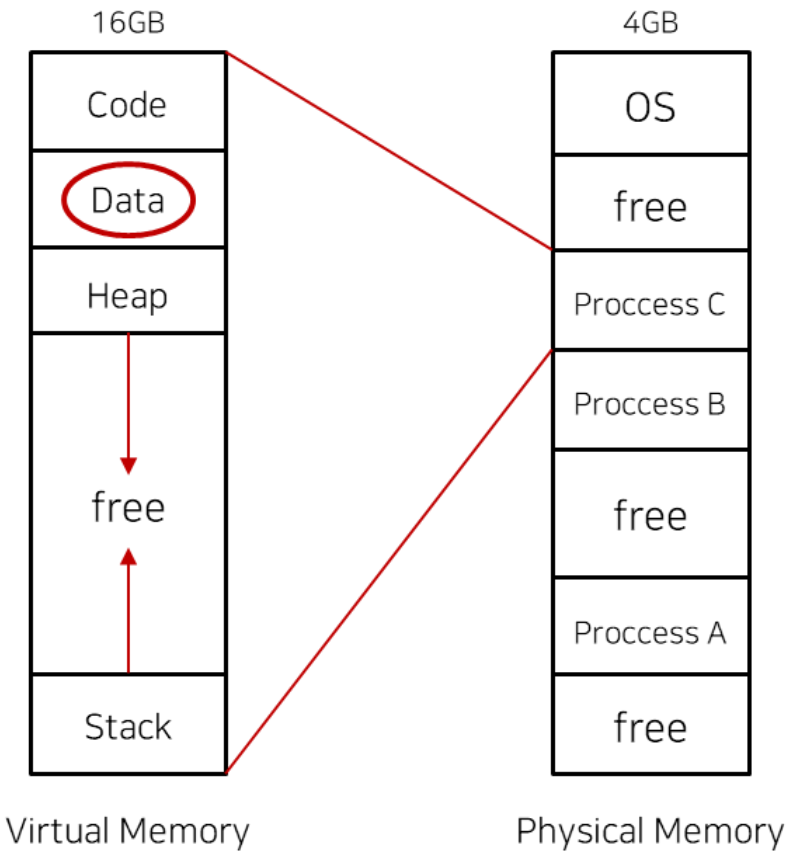
Block Diagram Memory Hierarchy

```
with open('data.txt', 'r') as f: ## 데이터를 읽어온다.
    data = f.readlines()

data = [int(x.strip()) for x in data]  # 파일에서 읽어온 데이터를 정수형으로 변환

data.sort()  # 오름차순으로 정렬

with open('sorted.txt', 'w') as f: ## 데이터를 저장한다.
    for d in data:
        f.write(str(d) + '\n')
```

sort 코드를 작성해봤다. 이 코드들은 어디서 작동이 되는 것일까?

데이터가 저장되어 있는 disk가 아닌, 메모리 단계까지 disk에 저장되어있는 데이터를 load 한 이후에 이를 메모리에서 sort 그리고 sort된 값들을 다시 disk에 저장한다.

즉 많은 데이터라고 한다면 이를 한번에 memory에서 처리하지 못하기 때문에 데이터를 따로따로 쪼개서 가져오게 되는데 여기에서 분산 컴퓨팅이 사용될 수 있다.



Virtual 메모리가 있기 때문에 우리는 우리가 생각한 양에 비해서 많은 양의 데이터를 사용할 수 있다. Process가 각자 자신의 memory를 가지고 있다고 생각하고 virtual memory를 생성, Virtual Memory는 os가 정해준 크기의 값으로 가상 공간을 형성하게 된다. 이 크기는 실제 physical 메모리보다 커질 수 있고 대부분 크다.

virtual 메모리는 필요에 따라 disk에서 데이터를 가져와서 physical 메모리에 load하고 필요하지 않은 데이터는 다시 disk에 저장하는 과정을 거친다. 즉 memory보다 큰 값을 한번에 memory에 넣지 않지만, disk에서 데이터를 load 하고 다시 save 하는 과정을 통해서 memory보다 큰 사이즈의 공간을 사용 가능한 것이다.

하지만 동시에 처리할 수 있는 양도 늘어나는 것일까?

정확히는 동시에 처리할 수 있는 데이터의 양은 virtual memory와 상관이 없다. physical 메모리에 올라갈 수 있어야지, process가 동시에 사용할 수 잇기 때문이다. 다음과 같은 그림 예시에서는 4GB를 사용한다.

이유는 다음과 같다,

Data 말고도 다양한 정보를 저장하고 처리하는 것 뿐만이 아닌, 다양한 작업이 공유하면서 메모리를 사용하기 때문에(page기법) 한 작업이 4GB 메모리를 모두 사용하고 싶다고 해서 모두 사용할 수 없으며, 즉 한번에 4GB 의 데이터를 사용할 수 없다.

우리가 사용하는 다양한 R이나 python에서 데이터를 많이 불러올 경우에 멈추는 경우가 있는데, 그 경우에 동시에 메모리에서 데이터가 할당받을 수 있는 양에 비해서, 큰 데이터를 가져왔기 때문일 수 있다.