# 과 제 제 출

**일 자 : 2016. 4. 25**
**제 출 자 : 황 원 주**

## RAM은 어떤 단어들의 약자이며 왜 RAM 이라고 부르게 되었는가?

RAM은 Random Access Memory의 약자로 직역하면 랜덤으로 접근하는 메모리, 즉 '임의 접근 메모리'를 뜻하는 말이다. 이는 메모리에 접근하는 방식에 따른 분류로 다른 방식으론 SAM(Sequential Access Memory)이 있다.

메모리에 접근하는 방식에는 크게 두가지가 있다. 하나는 순차적으로 접근하는 방식이고, 다른 하나는 임의로 접근하는 방식이다. 전자에 해당하는것이 SAM(Sequential Access Memory) 이고 후자에 해당하는 것이 RAM(Random Access Memory)이다.
이를 쉽게 설명하자면 우리가 일기를 쓴다고 가정을 하자. 그동안 기록했던 일기에는 일정 기간 동안 자신의 내용이 담겨 있을 것이다. 그중 자신이 원하는 날짜의 내용을 찾아보는 방법은 앞의 날짜부터 한장한장 넘기며 찾는 방법 혹은 오늘부터 역으로 거꾸로 한장한장 넘기며 그 페이지를 찾은 방법이 있을것이다. 이렇게 한장한장 순차적으로 넘겨가며 찾는 방법이 SAM 방식과 유사 하다. 그 중 앞에서 차근차근 찾아가는 방법을 FIFO(Fisrt-In-Fisrt-Out : 첫번째로 들어간 것이 첫번째로 나온다.) 접근 메모리라 하고 오늘부터 거꾸로 찾아가는 방법을 LIFO(Last-In-First_Out : 마지막으로 들어간 것이 첫번째로 나온다.) 접근 메모리라 한다.
이와 다른 방법으로 특별한 사건이나 날씨별로 분류하기 위해 포스트 잇 등을 이용해 일정 표식을 남겨 두었다고 가정해 보자. 그렇다면 한장한장 넘기지 않고도 그날의 일기를 바로 찾을 수 있을 것이다. 즉 내가 찾고싶은 날짜의 위치를 기억하고 그 곳을 바로 찾을 수 있다면 한장한장 넘길 필요가 없이 바로 갈 수 있는 것이다. 이렇게 순차적이 아닌 표시에 의해 임의로 찾는 방법이 RAM과 유사하다 할 것이다.
이런 RAM(임의 접근 메모리)도 읽기만 가능한 메모리와 읽기 쓰기 모두 가는한 메모리 두가지로 나눌 수 있는데, 읽기만 가능한 메모리를 ROM(Read Only Memory)이라 하고 읽고 쓰기가 모두 가능한 메모리를 RWM(Read Write Memory) 이라 부른다. ROM은 대표적으로 하드 디스크가 있고 우리가 현재 사용하고 있는 RAM은 엄밀하게 RWM이라고 할 수 있다.
하지만 RWM은 SAM의 세분화된 메모리 방식을 일컫는 용어로 선점을 했기 때문에 용어의 혼란을 막고자 RAM이라고 부르게 되었고 현재까지 사용되고 있다.

## 하버드 구조와 폰 노이만 구조의 한계는 각각 무엇이며 왜 폰 노이만 구조가 주로 사용되는 것일까?

하버드 구조와 폰 노이만 구조에 관한 자료들을 읽은 후 다음과 같은 생각이 들었다.

'고속도로가 있다. 이 고속도로는 서울과 인근 도시를 연결 해 주는 고속도로이다. 이 고속도로는

평소에는 문제가 없다. 차들이 시속 100km로 빠르게 달린다. 문제는 출퇴근 시간이다. 출근시간에는 상행선이 막히고, 퇴근시간에는 하행선이 막힌다. 고속도로 관리 사무소는 이런 문제를 예상했기 때문에 중앙에 가변 차로를 많이 만들어 놓았다. 이로인해 출근시간엔 상행선을 넓게 하고, 퇴근시간엔 하행선을 넓게 했다. 그래서 출퇴근 시간에도 큰 문제 없이 잘 운영되었다.
그런데 문제가 생겼다. 차량이 많아진 것이다. 출근시간에 상행선을 넓혔지만 톨게이트에서 병목현상이 생긴다. 퇴근시간에도 마찬가지 였다.
이를 해결하기 위해서는 차선을 넓히고 버스 전용차로를 시행해야 한다. 하지만 거기에 들어가는 비용이 만만치 않아 계속 고민 중이다. ‘

위 상황은 현재 폰 노이만 구조이다. 가변차로는 상행과 하행 두가지를 동시에 할 수 없다. 과학이 발달해 차량도 빨라지고 하이패스도 생겼지만 제한된 차선은 병목현상과 속박을 일으켜 그 기능을 다 발휘하지 못하게 한다. 이것을 해결하기 위해 차선을 분리하는 방법이 있다. 그것이 하버드 구조다. 도로를 넓히고 버스전용차로를 실행한다. 이것으로 명령어 메모리와 데이터 메모리가 분리되어 다닐 수 있다. 그에따라 속도가 빨라진다. 하지만 이 방법엔 돈이 많이든다. 제반 비용이 문제다.

정리하면, 폰 노이만 구조는 메모리에서 CPU까지 가는 길이 하나다. 그 하나의 길로 CPU는 메모리로 부터 명령어도 받고, 데이터도 받는다. 그리고 연산된 결과 데이터를 다시 메모리로 가져다 놓는 것도 그 버스에서 일어난다. 과학의 발전으로 CPU의 처리 속도는 빨라 졌지만 메모리 속박과 버스 병목 문제로 그 기능을 다 하기가 힘들다. 이것이 노이만 구조의 한계이다.
이것의 대안이 하버드 구조이다. 명령어 메모리와 데이터 메모리를 분리하여 CPU에 도달하게 한다. 그렇기 때문에 명령과 데이터 메모리를 읽는것을 동시에 할 수 있다. 하지만 이것도 구조적 한계가 있다. 데이터가 혼합된 경우 명령어 메모리가 데이터로 사용되는 경우가 있는데 이럴경우 명령어 메모리로 부터 데이터를 읽을 수 있는 특수 상황의 명령어가 필요하고, 주소 변환등의 과정을 거쳐야 한다. 또한 구성에 비용이 많이 들고, 복잡한 프로세스를 요구한다.

그렇다면 왜 폰 노이만 구조가 주로 사용되는 것일까?
첫번째는 단순하기 때문 일 것이다. 데이터를 분리 하지 않는 것은 하버드 구조 보다 단순 한 것이고, 이것은 오류등의 문제를 보다 적게 가져갈 수 있고, 안정성을 높일 수 있다.
두번째는 경제적인 문제 때문 일 것이다. 구조를 바꾸는데 들어가는 비용으로 얻는 만족도가 현재의 경제가치 보다 위에 있지 않을 것 이기 때문일 것이다.