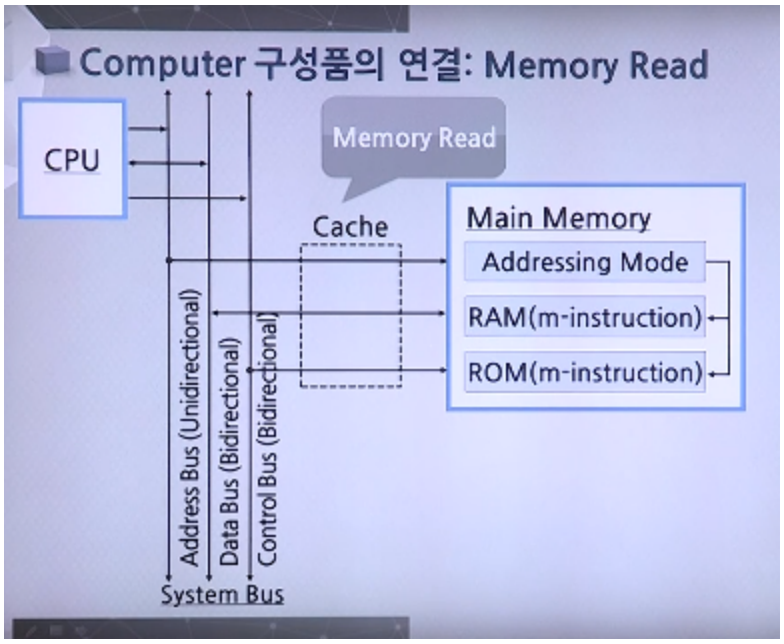
2강_컴퓨터 구성품의 연결



크게 네 가지 : 메모리 리딩, 메모리 라이팅(I/O reading), ssd에서 데이터 읽어오는 케이스, ssd에 데이터 쓰는 케이스(I/O write)

Memory Read

CPU, 중간에 system bus, cache, main memory 네 가지 블록

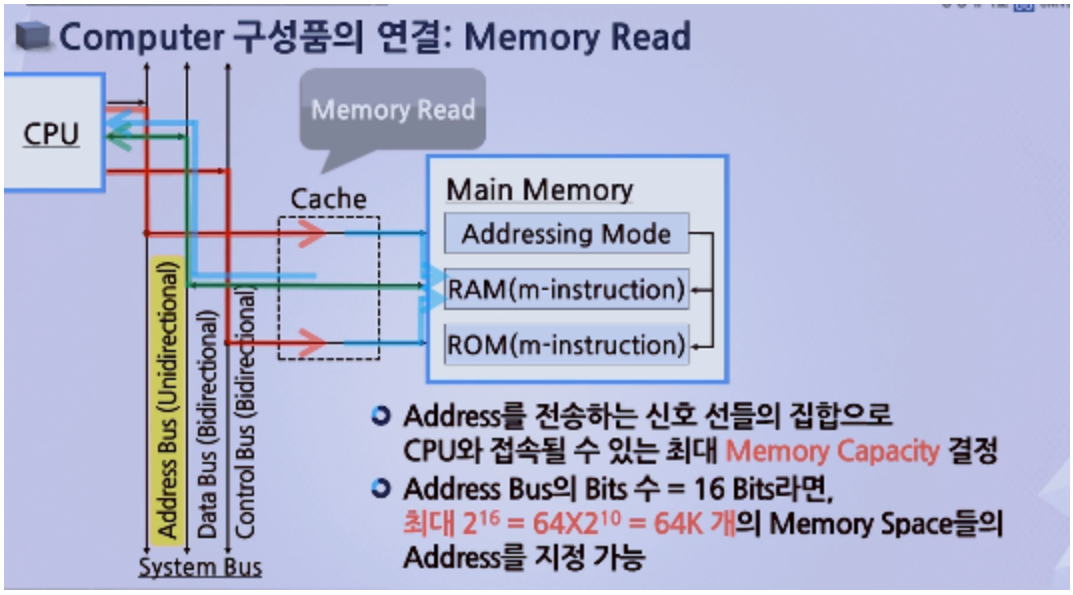
memory의 어디에 데이터가 있는가 ~ 주소를 보내줌

주소버스를 통해서 주소가 이동이 됨

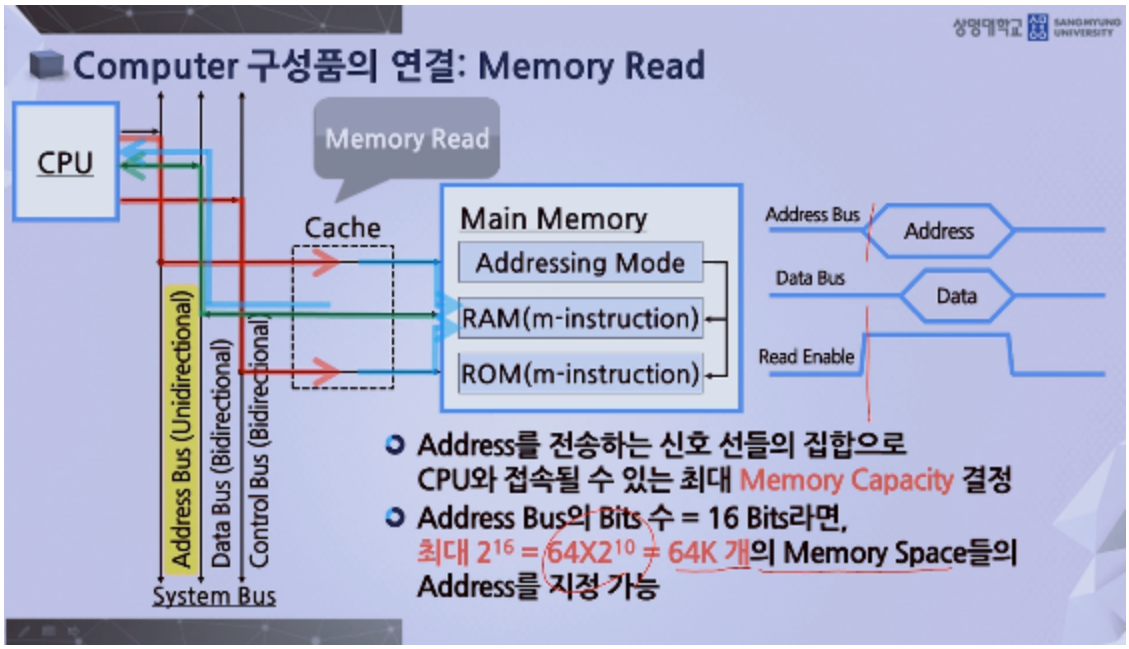
메모리 입장에서 어떤 주소가 입력 되는지, 데이터를 주는지 저장인지 컨트롤 버스 통해서 CPU가 신호 줌. 두 개의 신호를 동시에 줘야 한다

이 신호는 캐시로 먼저 저장이 된다. 캐시 안에 해당 데이터가 있으면, CPU로 데이터 버스를 통해서 전송하면 메모리 리드가 끝남

없으면, 메인 메모리로 그대로 신호가 전송됨 → 메인 메모리의 데이터가 데이터 버스를 통해서 cpu로 전송됨



Address Bus는 '주소를 전송하는 신호 선들의 집합이다'



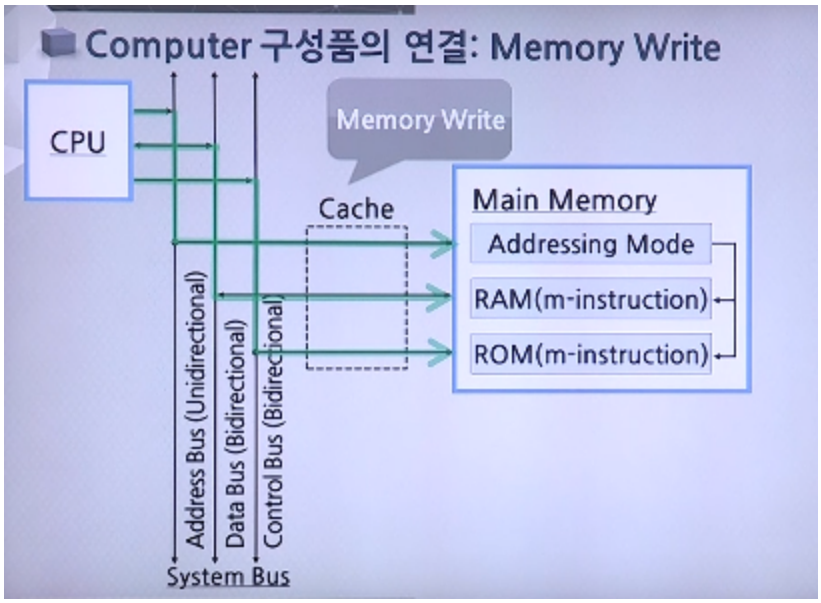
타이밍도

Read Enable 신호는 한 비트인 신호, Data Bus와 Address Bus는 한 비트가 아니라 n 비트란 의미. address하고 데이터가 한 비트만 전송되는 것이 아니라, n 비트가 저 마름모 안에 전송되는 것

x축은 시간의 흐름, address bus와 read enable의 시작지점이 같음 : 동시에 신호가 뜬다는 이야기 (내가 read할 memory에 있는 주소 지정을 CPU가 해줘야 하고, 동시에 control bus를 통해서 읽고싶다~는 read enable신호를 동시에 보내주게 되는 것 → 시간 지연 후 Data Bus에 데이터가 실리게 됨(CPU에서 메인메모리로 주소버스나 리드가능 신호가 전송되는 시간)

시간 딜레이, 메모리가 데이터를 data bus에 싣게 됨. 끝나는 지점이 CPU에서 데이터를 받아들이게 되는 시점

Memory Write



CPU에서 주소를 보내야 하고, CPU에서 연산할 데이터를 메모리로 보내야 함. 데이터 버스에서 데이터가 동시에 실려야 한다

Wiriting Enable신호를 동시에 전송해줘야만 메인 메모리가 그 신호들을 받아서, 메모리를 쓴다는 것을 알게 됨. 써야 될 데이터는 데이터 버스를 향해서 오고 있음

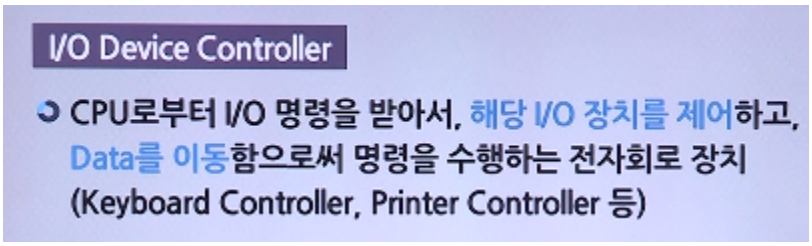
이 데이터를 어디다가 써야하는지는 address bus로부터 온 address가 지정을 해줘야 함

데이터버스의 라인이 32개==32비트다 == 동시에 병렬로 32비트를 쭉 보낼 수가 있다 : CPU와 Memory 간의 데이터 전송을 한 번에 1 Clock에 32비트가 한꺼번에 전송하게 된다

Control Bus : 시스템 내의 각종 요소들의 동작을 제어하기 위한 신호 선들의 집합
(memory read/write 말고도 여러 신호들 전송

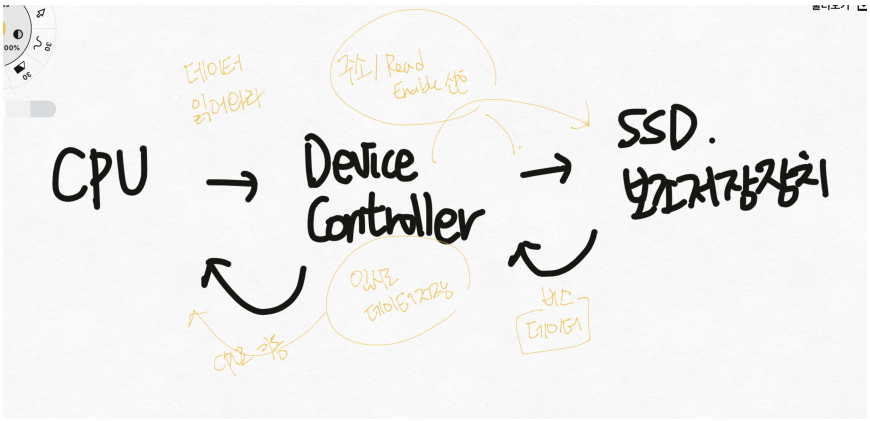
타이밍도

I/O Device Controller

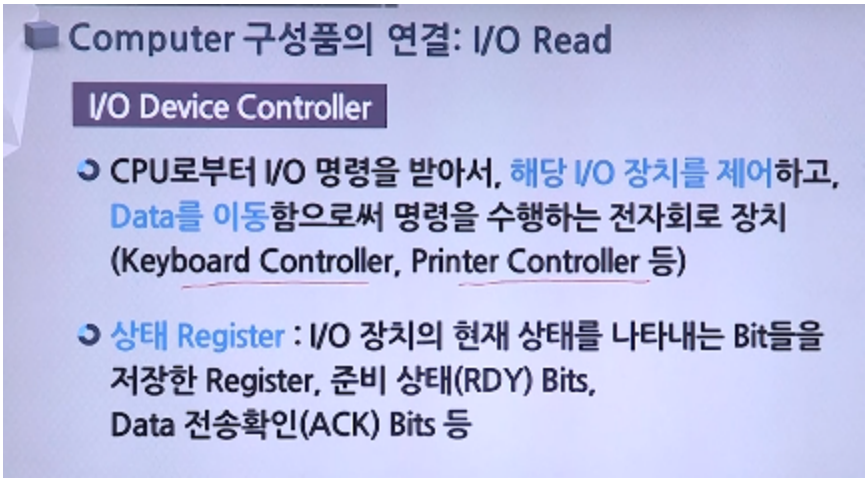


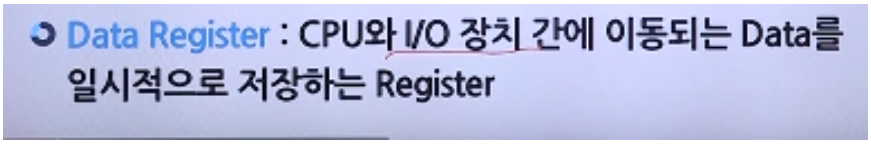
매개체 역할

CPU로부터 I/O 명령을 Device Controller가 받음(CPU와 직접적으로 연결됨) → 해당 I/O 장치를 제어하고(주소 전달), 데이터를 이동시킴

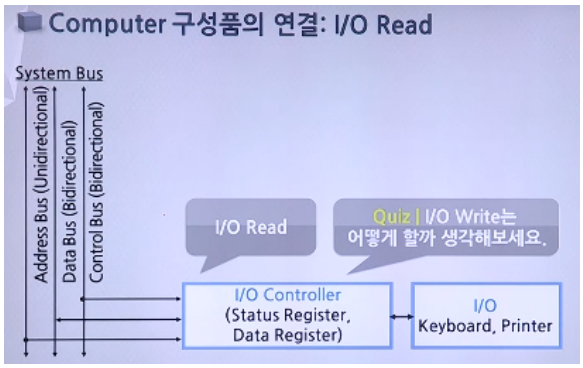


디바이스 컨트롤러에는 두 개의 레지스터가 존재한다

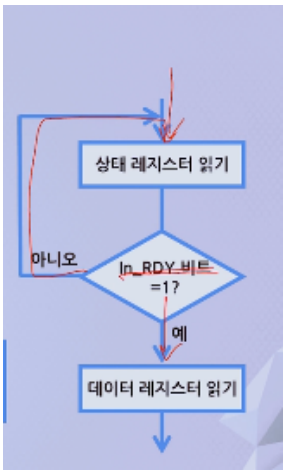




새로운 데이터가 차면, status register의 특정 비트를 0에서 1로 바꾼다 ('ssd에서 저 디바이스 컨트롤러 데이터가 이동해 있구나, 그럼 cpu가 그 데이터를 가져오면 되겠구나')

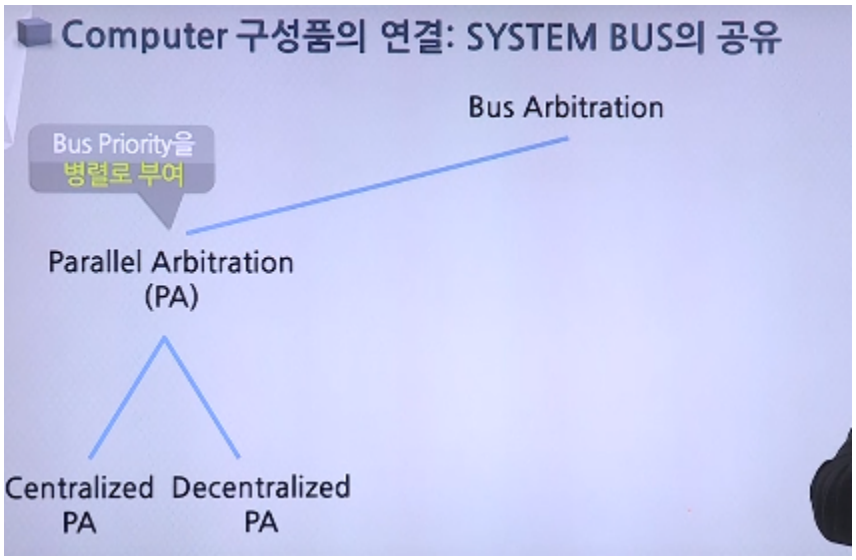


블록도



I/O 컨트롤러가 시스템버스에 직접적으로 연결됨, 실제 I/O 디바이스들을 시스템 버스와 직접적으로 연결 되어 있지 않고, I/O 컨트롤러를 통해서 연결이 되게 됨

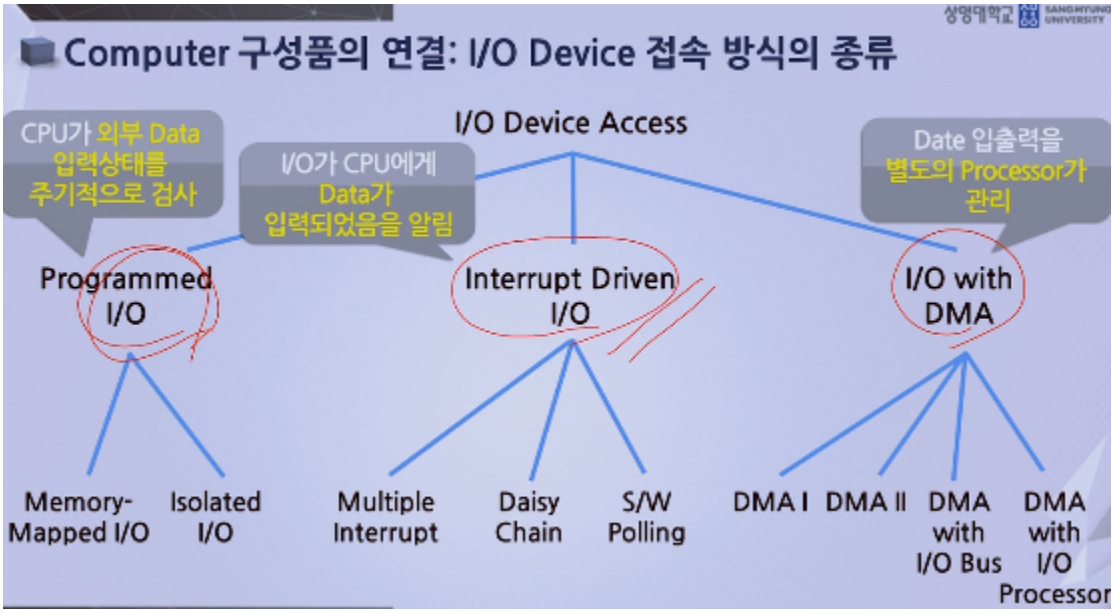
I/O Controller 디바이스 데이터 레지스터, 저장, 저장할 때마다 status Register는 기존의 0값을 1로 바꿈, 이걸 보고 CPU가 I/O controller에 있는 데이터 레지스터의 데이터를 가지고 오게됨



우선권 문제. CPU가 동시에 Memory에도 데이터를 요구, I/O(SSD 보조저장장치)에도 데이터 요구하는 경우 발생 가능 : 우선권 어디에?

Bus Arbitration

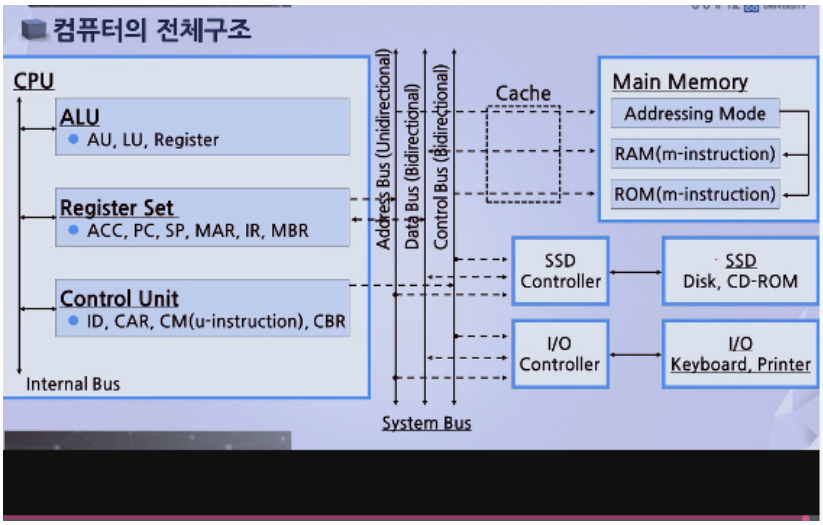
Bus Contention : 두 개 이상의 Device가 동시에 System Bus에 사용을 요구할 때 그럴 경우에 생기는 충돌 → 어디에 우선권을 줄건지 결정하는 방식을 Bus Arbitration



Programmed I/O(Polling) : 주기적으로 확인하는 방식

Interrupt Driven I/O : 초인종 누르면 나가는 타입 → 현재 대부분의 방식은 interrupt를 쓴다

I/O with DMA : 대리인이 데리고 와주는 타입



인 DMA Processer라는 것이 있는데 걔가 알아서 다 해주는 거에요.

그러니까 이것은 별도로 이야기할 필요는 없고, 나중에 이것이 어떻게 동작하는지, 자세히.. 나중에 한 10주차 이후에 그때 아마 확인하실 수 있을거에요.

자 ,그래서 여러가지 이야기를 했는데요. 컴퓨터 구조의 전체적인 모습을 보면 앞서 맨 처음에 보여드렸던 그림과 차이가 없는데 ,이런 형태로 구성이 되있고요.

이 그림은 머릿속에 넣어 두시는 게 좋을 것 같습니다. 앞으로도 반복적으로 이 그림을 보여드릴 것이고, 이 내부에서 어떤 식으로 데이터가 흘러가는지, 어떻게 저장하는지, 어떻게 데이터를 읽어오는지를 이 블록도를 기본으로 해서 여러분들한테 앞으로 설명을 드리도록 하겠습니다.