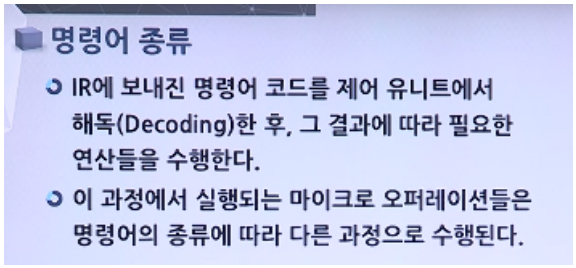
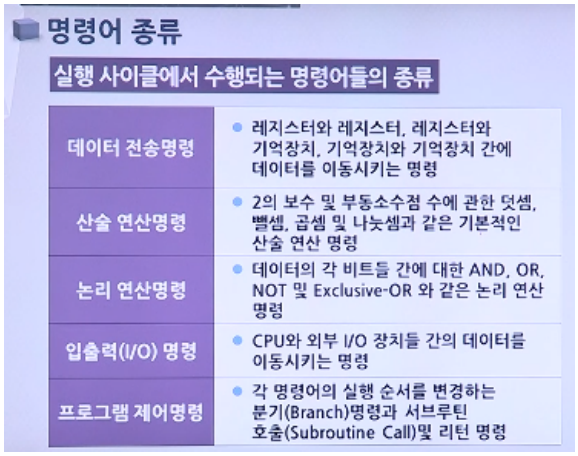
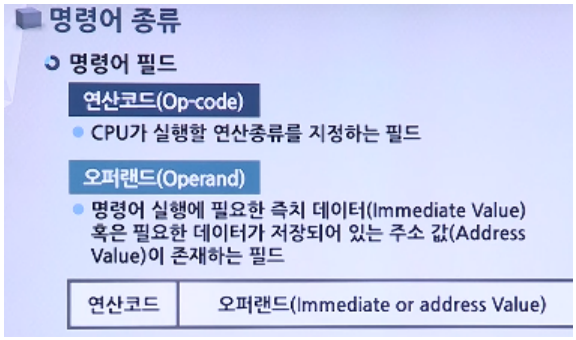
14강_명령어 종류 및 실행 사이클

Created	@Aug 11, 2020 5:23 PM
Tags	



여기서 보내진 명령어 코드, 즉 실행 명령어는 전부 다르기 때문에 내용이 다르다 마이크로 오퍼레이션들은 명령어의 종류에 따라 다른 과정으로 수행된다

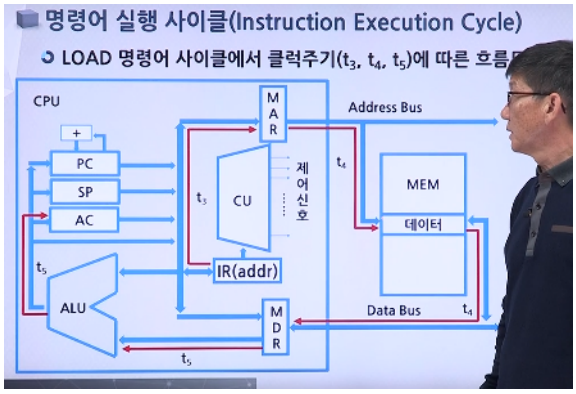




오퍼랜드는 한개 이상일 수 있다

LOAD 명령어 실행 사이클

LOAD 명령 : 메모리로부터 CPU로 필요



한 것을 가져오는 명령
* 클럭주기가 t3, t4, t5 인것은 이미 인출해서 ZERO, 1, 2, 3라는 클럭을 이미 사용했기 때문에 그 후에 사용되는 클럭 표시함

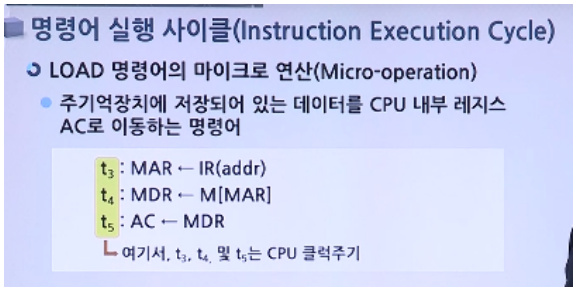
IR애 LOAD라는 명령을 가지고 있다는 것을~

그 LOAD라는 명령을 보면 OP코드가 LOAD이고, 그 뒤에 오퍼랜드에 보니까 address라는 정보가 있습니다. 그래서 address 정보는 메모리에 필요한 address 정보를 줌으로써 거기에 데이터를 CPU로 가져와라 이런 뜻이니까 이 addr 값을 t3라는 클럭 동안에 MAR로 옮기게 되죠.

설명 잘 모르겠네

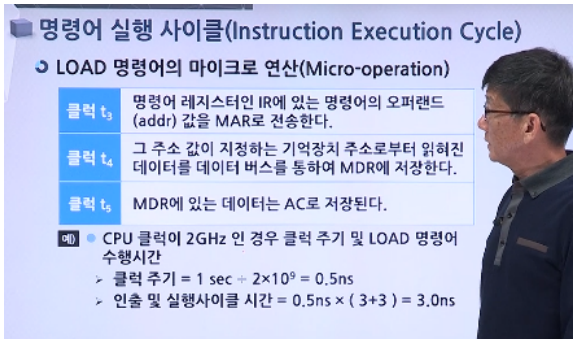
그 다음 클럭에서는 MAR에 이제 주소정보가 메모리를 지정함으로써 필요한 데이터가 이 데이터버스를 통해서 MDR까지 오는 클럭. 그게 이제 두 번째 클럭이 되겠구요. 그 다음에 LOAD. 명령어의 뜻에 따라 MDR이 과연 이제 어디로 가느냐. 마지막 클럭에는 어큐뮬레이터(accumulator) 가야 되니까 여기 지금 화살표 되어 있는데로 t5라는 클럭 동안에 이렇게 통해서 AC 로 이동하는 이 과정이 바로 LOAD 명령어의 사이클 주기가 되겠죠. 그러면 LOAD 명령어가 이렇게 되면 끝나는 겁니다.

인출할 때는 그 자체의 명령어가 뭔지 모르지만, 인출한 다음에 IR의 해독을 통해서 LOAD라는 것을 알고, 그 명령을 마이크로 오퍼레이션으로 이제 표현할 수 있는 것



첫 번째 클럭에서는 그 IR에 원래 있떤 IR 명령어에 오퍼랜드 필드의 addr 값을 MAR로 보내고, 또 MAR에 저장되어 있는 그 내용을 메모리 지정을 하여 해당 주소 값에 있는 내용을 MDR로 보내게 된다

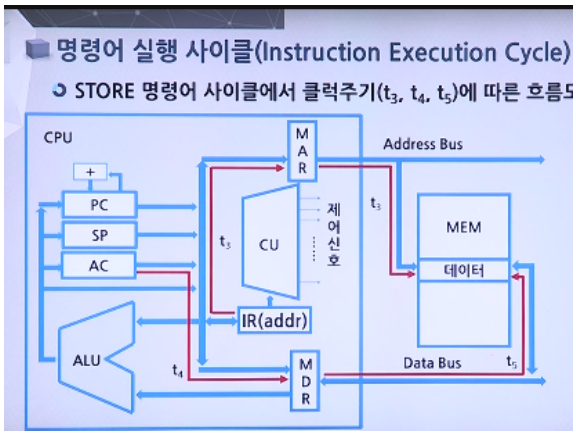
MDR은 ALU를 통해서 최종 목적지인 AC(accumulator)로 보내게 됨.



이것이 세 개의 클럭에서 이뤄지는 마이크로 오페레이션(LOAD 명령이 인출에서부터 실행까지 끝나는 과정이 됨)

메모리 지연이 없다고 가정했을 때임!

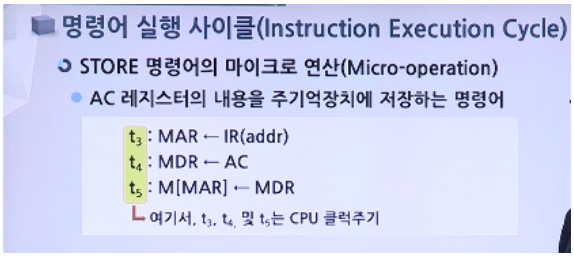
STORE 명령어 실행 사이클

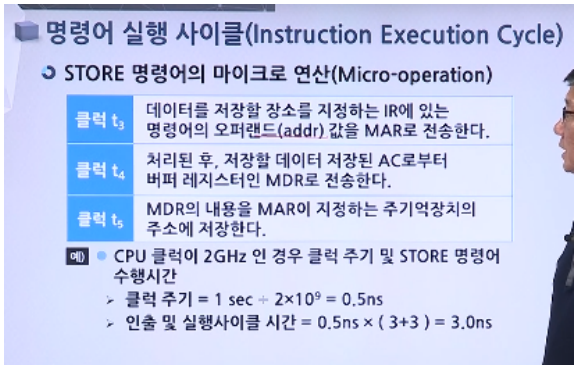


STORE은 ALU에서 연산된 결과를 또는 어떤 형태든지 필요한 데이터를 메모리로 보내는 과정
메모리로 보낼 때 어디에 저장되어 있는가? : CPU 내에서 AC에 그 값을 가지고 있다

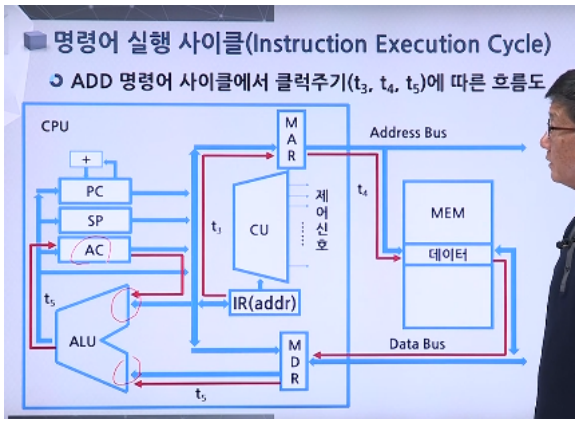
AC에서 메모리로 저장을 하기 위해서, AC가 MDR로 보내진다(메모리를 과연 어디에 지정/저장 할 것인지, 누가 지정 할 것인지의 문제)

STORE 명령 인출 → STORE 뒤에 오퍼랜드 명령 : address 정보~ IR에 있다 → 이 IR의 address 정보를 MAR로 보내, MAR이 지정하게 됨 → MAR로 지정한 곳을 MDR이 데이터 저장함



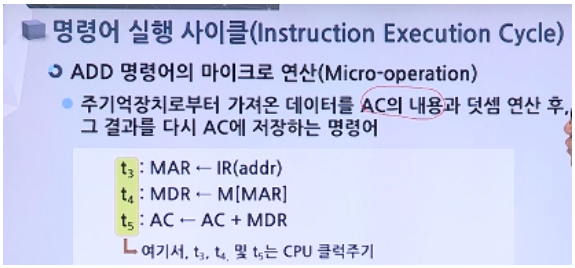


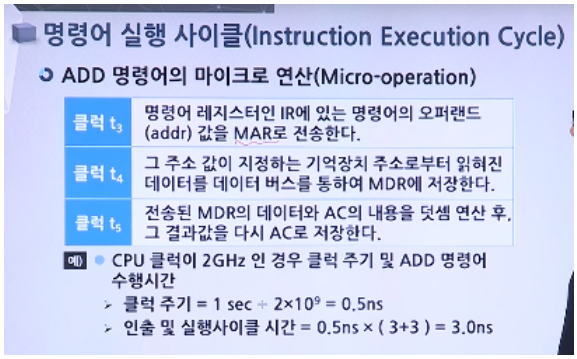
ADD 명령어 실행 사이클



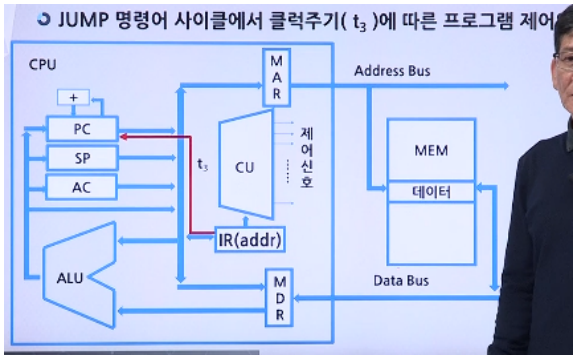
더하는 것이니 ALU 이용. ALU에 인풋되는 두 개를 더해서 AC에 보내는 과정

ADD 명령어 인출→ 데이터가 필요하다 → 데이터가 어디있나 ? : 오퍼랜드에 addr 정보를 지정하고 있음 → addr을 MAR로 보내기 → MAR로 지정된 데이터가 데이터 버스를 통해서 그 다음 클럭에 MDR로 나오게 됨 → MDR의 내용과 AC 원래 있던 데이터가 ALU로. 더해지고 다시 AC에 저장(업데이트)





JUMP 명령어 실행 사이클



프로그램 카운터가 프로그램을 수행하는 데 주소정보를 가지고 있음. JUMP를 위해서는 이 프로그램 카운터의 내용을 바꿔줘야 ← JUMP 명령어의 목적

JUMP 명령어다! → 어디로 JUMP? → 오퍼랜드 보니 ADDR 정보가 있다. 이걸 Program Counter(PC)에 준다 → PC의 내용이 MAR로 전해져서 하는 **인스트럭션 패치** 과정으로 연결

JUMP는 PC로 주소정보를 전달하는 한 단계만 하면 되어서, 앞의 명령어들하고 시간차이가 난다