

13강_명령어 사이클 및 인출 사이클

🕒 Created	@Aug 11, 2020 5:22 PM
🏷 Tags	

명령어 실행에 대하여

학습목표

- » CPU가 기억장치에 저장되어 있는 다양한 명령어들에 대해 인출하고 실행하는 과정에 대해 설명할 수 있다.
- » “컴퓨터 처리 (Computer Processing)” 라는 작업을 단계별로 세분화하여 그 기능들을 학습함으로써 CPU의 전체적인 처리과정을 설명할 수 있다.

학습내용

- » 명령어 사이클 및 인출 사이클
- » 명령어 종류 및 실행 사이클
- » 간접사이클 및 종합실행 예제

CPU가 기억장치에 저장되어 있는 다양한 명령어들을 실행함으로써 이루어지는 “컴퓨터 처리(Computer Processing)” 라는 종합적인 작업들을 여기서는 매우 기본적인 기능으로 분해하여 단계별로 처리함으로써 CPU의 전반적인 동작을 이해할 수 있도록 한다.

명령어 인출(Instruction Fetch)

- 주기억장치로부터 지정된 주소에 있는 명령어를 IR로 가져온다.

명령어 해독(Instruction Decode)

- 실행해야 할 동작을 결정하기 위하여 인출된 명령어를 해독한다.

데이터 인출(Data Fetch)

- 명령어 실행을 위하여 데이터가 필요한 경우에는 주기억장치 또는 입출력장치로부터 데이터를 가져온다.

IR(instruction Register)

데이터에 대한 산술, 논리 연산들은 ALI에서 한다. 이 연산들을 수행하는 과정이 데이터 처리.

처리를 통해 얻은 결과 저장(데이터 저장)

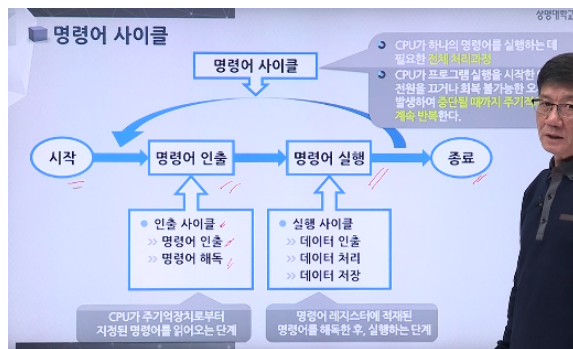
CPU가 기억장치에 저장되어 있는 다양한 명령어들을 실행함으로써 이루어지는 "컴퓨터 처리(Computer Processing)"라는 종합적인 작업들을 여기서는 매우 기본적인 기능으로 분해하여 단계별로 처리함으로써 CPU의 전반적인 동작을 이해할 수 있도록 한다.

데이터 처리(Data Process)

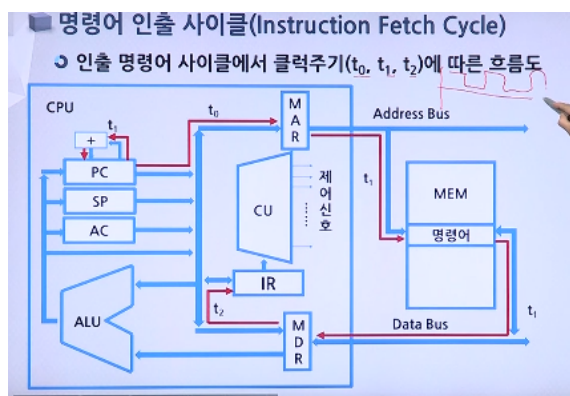
➤ 데이터에 대한 산술 혹은 논리적 연산을 수행한다.

데이터 저장(Data Store)

➤ 수행한 결과를 저장장치에 저장한다.



(전원을 끄거나 회복 불가능한 오류가 발생하여 중단될때까지 주기적으로 계속 반복한다.)



인출 명령어 클럭주기 한 주기 동안에 무엇을 하나. t_0, t_1, t_2

빨간색으로 표시된 부분이 시간마다 흐름.

PC에서 데이터를 MAR로 가져오게됨 - 첫 번째 클럭

MAR의 주정보는 메모리 어딘가에 지정 - 두 번째 클럭

그와 동시에 프로그램 카운터는 자동적으로 증가(워드길이만큼) - 두 t_1 은 동시에 이루어지는 마이크로 오퍼레이션이 된다
마지막 t_2 동안 MDR에 있는 것을 IR로 옮기기

이 과정을 표현한 것 :: **마이크로 오퍼레이션**

CPU 클럭주기가 빠르면 이 시간도 빨라지고, 느려지면 이 시간도 느려진다. 같이 연동

명령어 인출 사이클(Instruction Fetch Cycle)

인출 사이클의 마이크로 연산(Micro-operation)

- 이것은 모든 명령어가 공통적으로 수행된다.

t_0 : $MAR \leftarrow PC$

t_1 : $MDR \leftarrow M[MAR], PC \leftarrow PC+1$

t_2 : $IR \leftarrow MDR$

여기서, t_0 , t_1 , 및 t_2 는 CPU 클럭주기

명령어 인출 사이클(Instruction Fetch Cycle)

인출 사이클의 마이크로 연산(Micro-operation)

클럭 t_0	현재의 PC의 주소 값을 CPU 내부 버스를 통하여 MAR로 전송한다.
클럭 t_1	그 주소 값이 지정하는 기억장치 주소로부터 읽혀진 명령어를 데이터 버스를 통하여 MDR에 저장하고, PC의 값에 워드의 길이 만큼을 더한다.
클럭 t_2	MDR에 있는 명령어가 명령어 레지스터인 IR로 전송된다.

예) CPU 클럭이 2GHz 인 경우 클럭 주기 및 인출 사이클 시간

클럭 주기 = $1 \text{ sec} \div 2 \times 10^9 = 0.5 \text{ ns}$

인출 사이클 시간 = $0.5 \text{ ns} \times 3 = 1.5 \text{ ns}$