04. CPU workflow

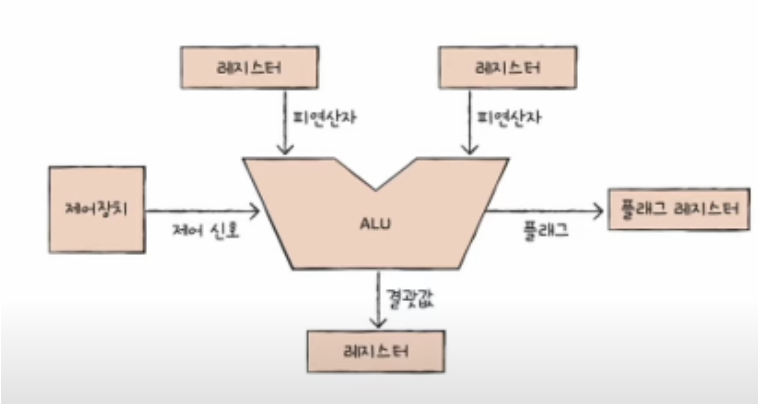
yongzzai

Summary of Chap.04 of book [1]

1 ALU와 제어장치

1.1 ALU가 받아들이는 정보

* ALU는 아래와 같은 정보를 받아들임.



* 계산을 하기위해선 피연산자와 수행할 연산이 필요함.
* 2+1을 수행한다고 하면, **피연산자**인 2와 1을 레지스터로부터 받아들이고, 제어장치로부터 더하기라는 **제어신호**를 받아들여서 2+1을 수행함.
* 결과값을 메모리가 아닌 레지스터에 담는 이유는 CPU가 레지스터에 접근하는 속도가 램에 접근하는 속도보다 훨씬 빠르기 때문임.

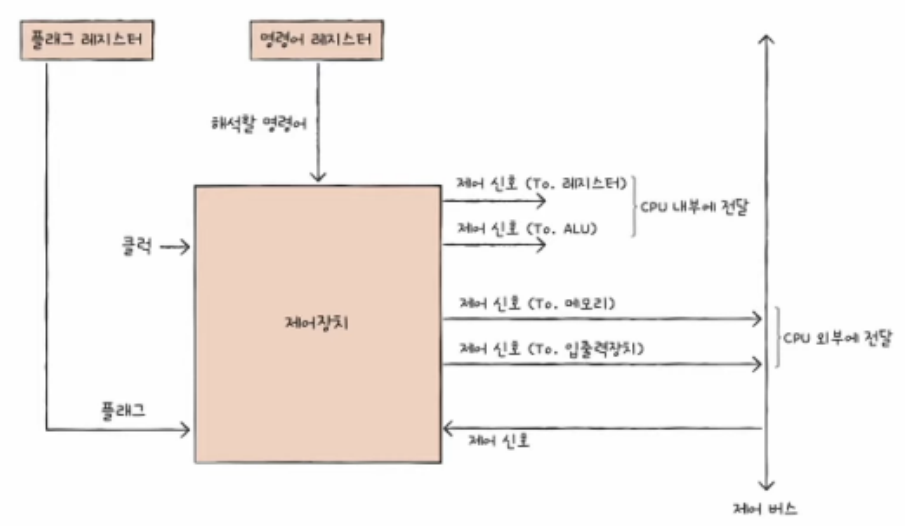
1.2 플래그

* **플래그 레지스터**는 ALU가 내보내는 연산 결과에 대한 부가정보를 담게됨. (참고. 플래그는 이진수 $0101_{(2)}$가 양수인지 음수인지를 구분할 때 사용함 (Chap 02))

플래그 종류	의미	사용 예시
부호 플래그	연산한 결과의 부호를 나타낸다.	부호 플래그가 1일 경우 계산 결과는 음수, 0일 경우 계산 결과는 양수를 의미한다.
제로 플래그	연산 결과가 0인지 여부를 나타낸다.	제로 플래그가 1일 경우 연산 결과는 0, 0일 경우 연산 결과는 0이 아님을 의미한다.
캐리 플래그	연산 결과 올림수나 빌림수가 발생했는지를 나타낸다.	캐리 플래그가 1일 경우 올림수나 빌림수가 발생했음을 의미하고, 0일 경우 발생하지 않았음을 의미한다.
오버플로우 플래그	오버플로우가 발생했는지를 나타낸다.	오버플로우 플래그가 1일 경우 오버플로우가 발생했음을 의미하고, 0일 경우 발생하지 않았음을 의미한다.
인터럽트 플래그	인터럽트가 가능한지를 나타낸다. 인터럽트는 04-3절에서 설명한다.	인터럽트 플래그가 1일 경우 인터럽트가 가능함을 의미하고, 0일 경우 인터럽트가 불가능함을 의미한다.
슈퍼바이저 플래그	커널 모드로 실행 중인지, 사용자 모드로 실행 중인지를 나타낸다. 커널 모드와 사용자 모드는 09장에서 설명한다.	슈퍼바이저 플래그가 1일 경우 커널 모드로 실행 중임을 의미하고, 0일 경우 사용자 모드로 실행 중임을 의미한다.

* 위의 표는 플래그의 종류를 나타냄.

1.3 제어장치



* **클럭**은 컴퓨터의 모든 부품을 일사분란하게 움직일 있게 하는 시간단위 (시계가 똑딱똑딱하듯이 박자를 만들어냄)

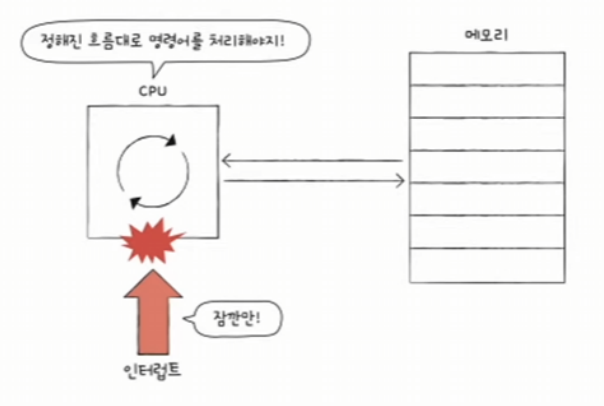
* **해석할 명령어**는 **명령어 레지스터**를 통해 출력됨.
* 제어장치가 내보내는 **제어신호**는 CPU내부에 전달하는 제어신호와 CPU외부에 전달하는 제어신호로 구분됨.

2 레지스터

* 레지스터는 CPU내부의 작은 임시저장장치임. 프로그램 속 명령어와 데이터는 실행 전후로 레지스터에 저장됨.
* 레지스터는 CPU마다 다르고, 레지스터마다 종류가 다양하지만 공통적으로 포함되는 필수적인 레지스터의 종류가 있음.

1. **프로그램 카운터**
메모리에서 가져올 명령어의 주소
2. **명령어 레지스터**
해석할 명령어 (방금 메모리에서 읽어들인)
3. **메모리 주소 레지스터**
메모리의 주소
4. **메모리 버퍼 레지스터**
메모리와 주고받을 값(데이터와 명령어)
5. **플래그 레지스터**
연산 결과 또는 CPU상태에 대한 부가적인 정보
6. **범용 레지스터**
다양하고 일반적인 상황에서 자유롭게 사용함. 여러 종류가 있음.
7. **스택 포인터**
자료구조의 스택(Stack)에서 가장 위의 주소를 가리킴. (스택이 어디까지 찼는지)
8. **베이스 레지스터**
변위 주소 지정 방식 즉, 오퍼랜드 필드의 값(변위)과 특정 레지스터의 값을 더하여 유효주소를 얻음.

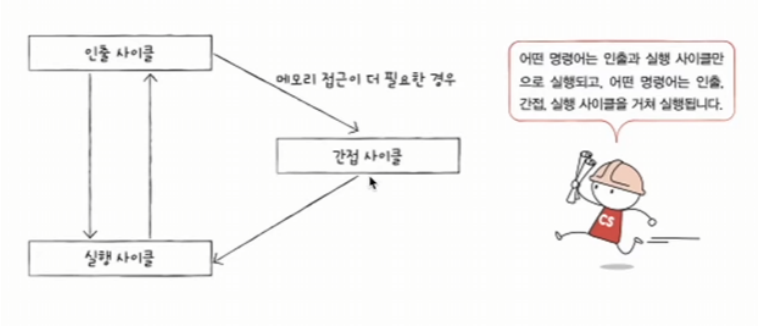
3 명령어 사이클과 인터럽트



* CPU는 메모리로부터 명령어나 데이터를 가져와서 처리함. 이때 명령을 처리하는 과정에는 일정한 주기가 존재함. CPU는 메모리안의 프로그램을 정해진 흐름대로 처리하고 이 흐름 또는 주기를 **명령어 사이클**이라고 함. 이때 정해진 흐름을 방해하는 신호를 **인터럽트**라고 함.

3.1 명령어 사이클

* 프로그램 속 명령어들은 일정한 주기가 반복되며 CPU에 의해 실행되고, 이 주기를 **명령어 사이클**이라고 함.



1. 인출 사이클: 메모리에 있는 값을 CPU로 가져오는 작업의 주기
2. 실행 사이클: 가져온 값을 실행하는 주기
3. 간접 사이클: 인출-실행 사이클 외에 메모리에 추가적인 접근이 필요한 경우 (ex. 주소필드 등이 필요함.)

3.2 인터럽트

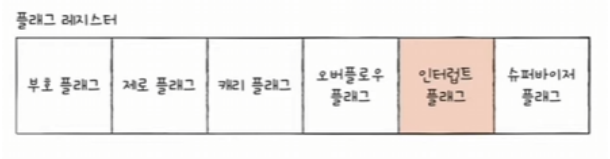
* 일반적으로 CPU의 작업은 **인출-실행-간접**으로 실행이 되지만 인터럽트에 의해 사이클이 방해받을 수 있음.
* **인터럽트**는 CPU가 주목할 필요가 있을 때, CPU가 빠르게 처리해야할 다른 작업이 생겼을 때 발생됨.
* 인터럽트는 두가지 종류가 있음 – > 동기 인터럽트와 비동기 인터럽트

1. **동기 인터럽트**
CPU가 예기치 못한 상황을 접했을 때 발생함. 예를 들어, 실행할 수 없는 명령어가 있거나, 디버깅, 주소에 데이터가 없는 상황 등에 발생됨.
2. **비동기 인터럽트 (하드웨어 인터럽트)**
입출력장치에 의해 발생됨. 알림과 같은 역할을 함 (ex. 세탁, 조리완료됨 등)
* 마우스 클릭, 키보드 입력 등도 인터럽트의 한 종류이며 CPU가 입출력 작업

도중에도 효율적으로 명령어를 처리하기 위해 사용함.
* 하드웨어 인터럽트는 아래와 같은 순서를 통해 처리됨. 인터럽트의 종류에 상관 없이 인터럽트 처리 순서는 아래의 순서를 거의 따름.

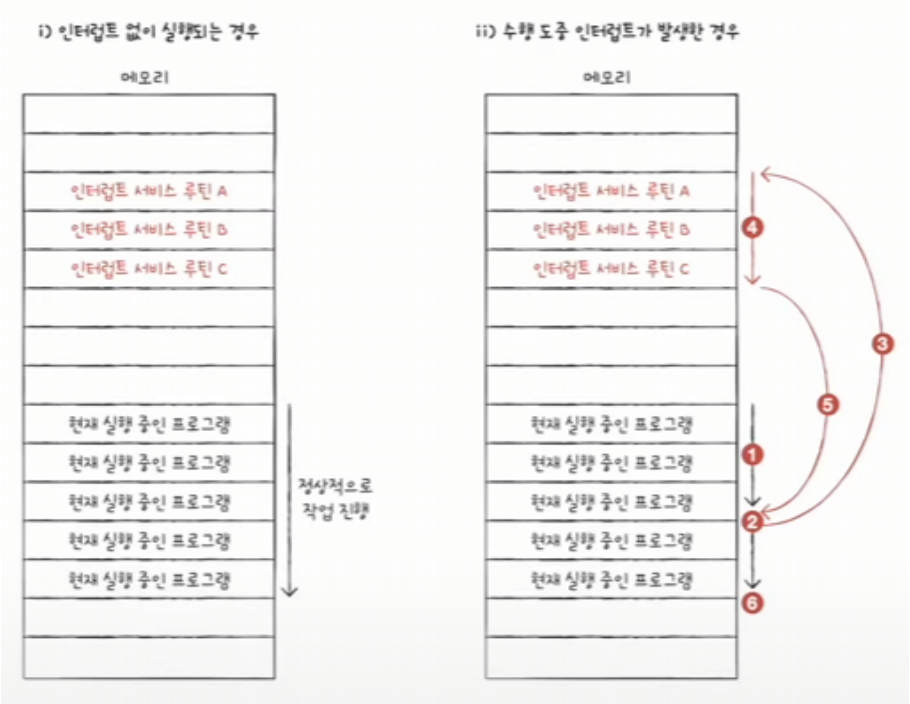
❶ 입출력장치는 CPU에 **인터럽트 요청 신호**를 보냅니다.
❷ CPU는 실행 사이클이 끝나고 명령어를 인출하기 전 항상 인터럽트 여부를 확인합니다.
❸ CPU는 인터럽트 요청을 확인하고 **인터럽트 플래그**를 통해 현재 인터럽트를 받아들일 수 있는지 여부를 확인합니다.
❹ 인터럽트를 받아들일 수 있다면 CPU는 지금까지의 작업을 백업합니다.
❺ CPU는 **인터럽트 벡터**를 참조하여 **인터럽트 서비스 루틴**을 실행합니다.
❻ 인터럽트 서비스 루틴 실행이 끝나면 ❹에서 백업해 둔 작업을 복구하여 실행을 재개합니다.

1. **인터럽트 요청 신호**
입출력장치가 CPU에게 지금 인터럽트를 보내도되는지 요청하는 신호



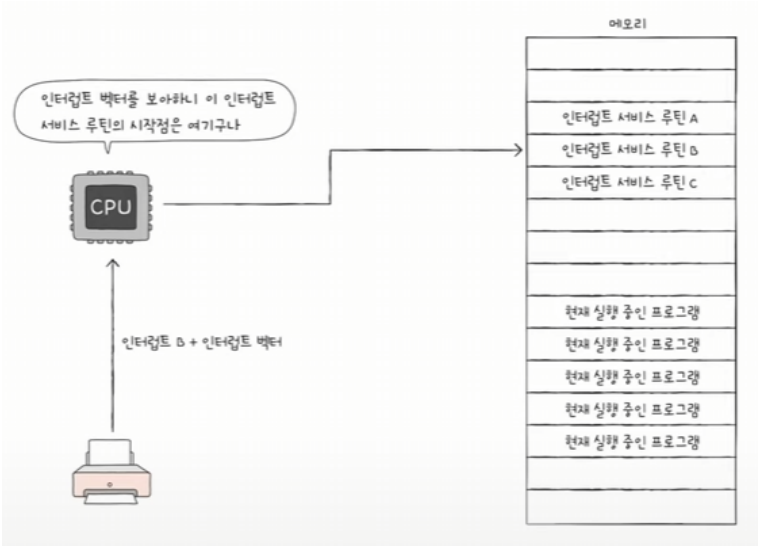
요청 신호를 받은 CPU는 플래그 레지스터의 인터럽트 플래그를 확인함. 현재 인터럽트를 받아들일 수 있는지 없는지를 확인.

2. **인터럽트 서비스 루틴**
CPU가 인터럽트를 수용하기로 하면 인터럽트 서비스 루틴을 실행함. 인터럽트 서비스 루틴은 프로그램이기 때문에 메모리에 저장되어있음.
아래의 그림은 인터럽트가 발생되지 않을때와 발생될 때의 작업 흐름을 설명함.



3. **인터럽트 벡터**

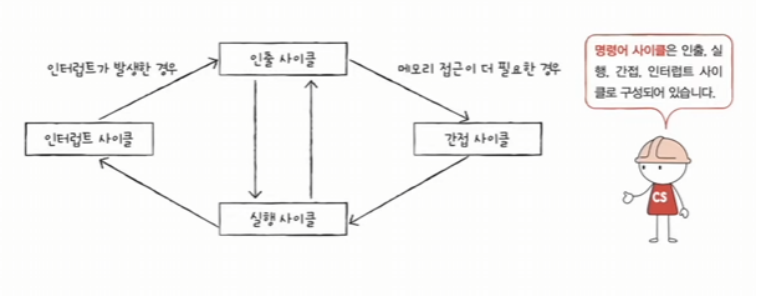
키보드, 마우스, 등등 각각 독립적인 인터럽트 서비스 루틴을 가지는데 이것을 구분하기 위한 정보가 **인터럽트 벡터**임.



4. **백업**
인터럽트 서비스 루틴을 실행할 때, 기존에 진행중이던 작업을 **백업** 해두어야함. 이때, 메모리에 Stack 영역에 기존 작업을 백업해두고 인터럽트 서비스 루틴을 수행한 이후에 다시 Stack 영역에 있는 기존 작업을 불러옴.

3.3 명령어 사이클 요약

* CPU는 결국 메모리에 있는 프로그램을 아래와 같이 정형화된 흐름에 따라서 처리함.



References

[1] Minchul Kang. 혼자 공부하는 컴퓨터 구조+운영체제. Hanbit Media, 2022.