컴퓨터 구조

4-3. 명령어 사이클과 인터럽트
5-1. 빠른 CPU 설계 기법

발표자 : 정승원 | 원터디 Lab | jswfs1284@naver.com | 2024.01.06

4장-3 명령어 사이클과 인터럽트

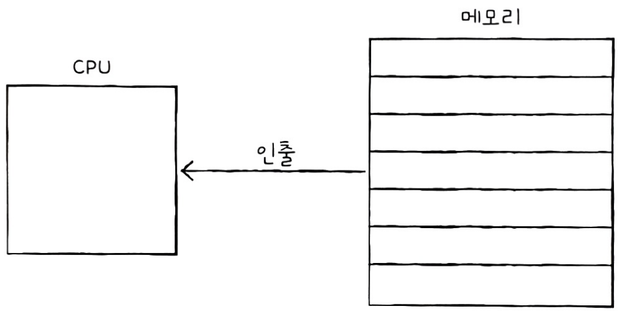
KEY WORD 명령어 사이클, 인터럽트, 예외, 하드웨어 인터럽트, 인터럽트 서비스 루틴

01

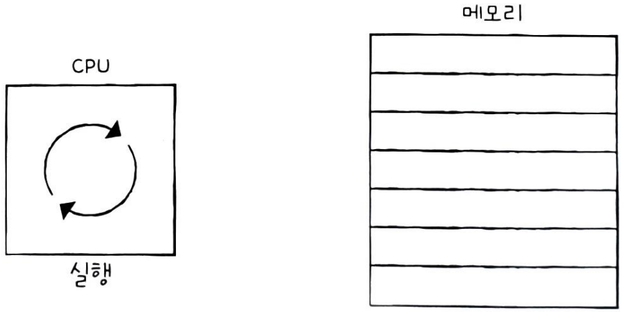
명령어 사이클

Instruction cycle

: 하나의 명령어를 실행하는 주기



① 인출 사이클 fetch cycle : 메모리에 있는 명령어를 인출하는 과정

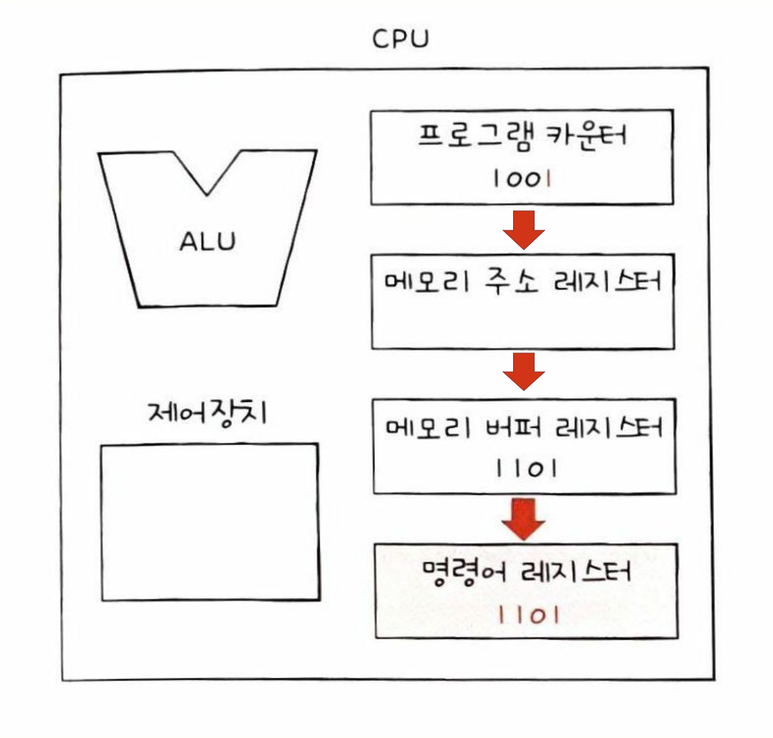


② 실행 사이클 execution cycle : 명령어를 실행하는 과정

- CPU로 가져온 명령어, 즉 **명령어 레지스터**의 값을 실행
- 제어장치가 **명령어 레지스터**에 담긴 값을 해석하고, 제어신호 발생

01

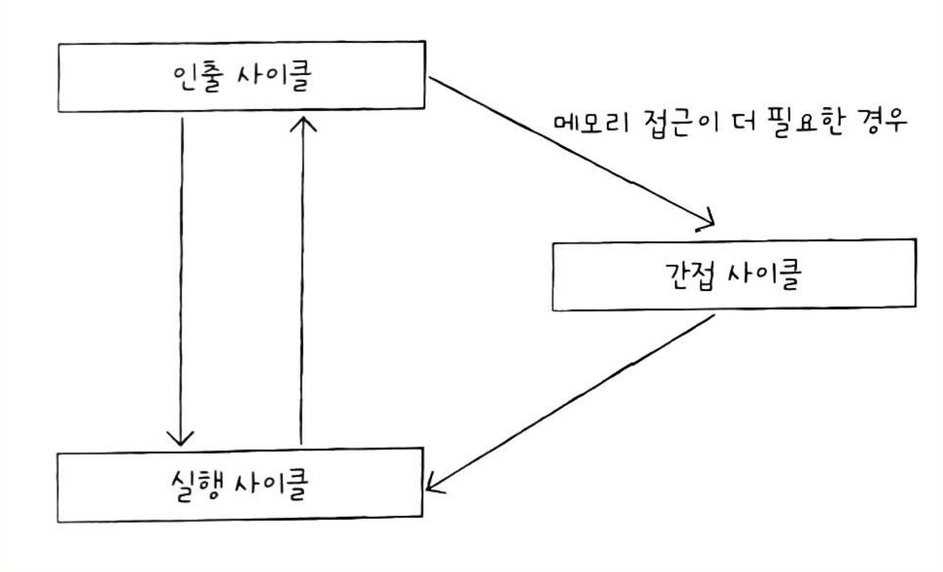
명령어 사이클



[참고] 레지스터의 작동 과정 (p.115) == 인출 과정

01

명령어 사이클



예) 피연산자operand 필드에 유효주소의 주소가 적힌 경우는, 메모리 접근을 한 번 더 해야 한다.

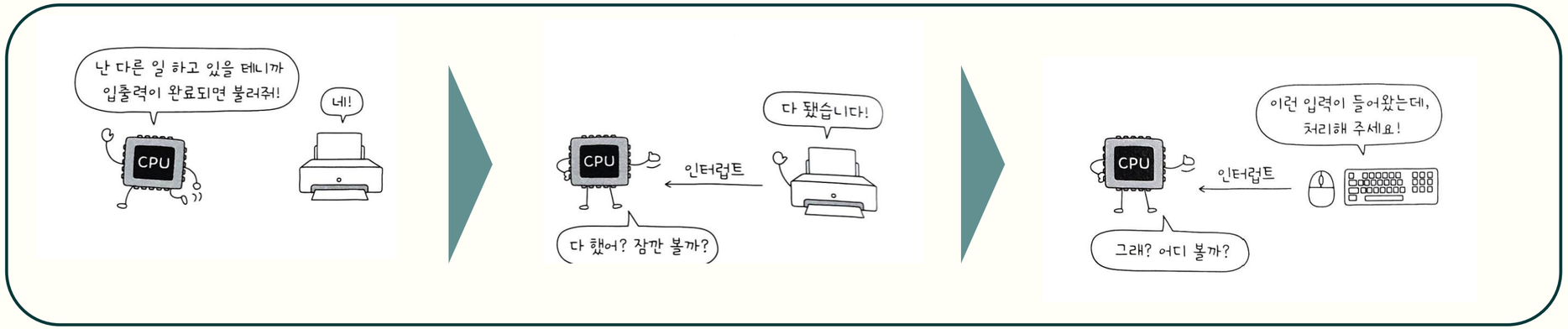
➔ 간접 사이클까지 추가됨

02

인터럽트

interrupt; 중단하다/방해하다

- **CPU 작업을 방해하는 신호**
- **종류:**
  - ✓ 동기 인터럽트 synchronous interrupts (예외): CPU에 의해 발생하는 인터럽트.
  - ✓ 비동기 인터럽트 asynchronous interrupts (**하드웨어 인터럽트**): 주로 입출력장치에 의해 발생하는 인터럽트.



02

인터럽트

하드웨어 인터럽트

- 하드웨어 인터럽트 == 알림
- CPU는 입출력 작업 도중에도 효율적으로 명령어를 처리하기 위해 하드웨어 인터럽트 사용.
- 예) CPU가 프린터에 출력 명령 → but, 프린터는 CPU보다 느림

  → CPU는 프린트 작업이 끝날 때까지 기다리거나, 주기적으로 완료 여부를 확인해야 함

  → 해결책: 프린트 완료 인터럽트 발생 시키기

02

인터럽트

하드웨어 인터럽트

- 인터럽트 처리 순서

지금 끼어들어도 되나요?

어, 인터럽트다

인터럽트 받아도 돼? ➔ ㅇㅇ

기다려봐 백업좀,

B번 매뉴얼을 따라 처리해볼게

끝. 하던거 마저 해야지.

① 입출력장치가 CPU에 **인터럽트 요청 신호**를 보냄.

② CPU는 실행 사이클이 끝나고, 새로운 명령어 인출 전에 인터럽트 여부 확인.

③ CPU는 인터럽트 요청을 확인하고 **인터럽트 플래그**를 통해 현재 인터럽트 받아들일 수 있는지의 여부 확인.

④ 인터럽트 받아들일 수 있다면 CPU는 지금까지의 작업을 백업.

⑤ CPU는 **인터럽트 벡터**를 참조하여 **인터럽트 서비스 루틴**을 실행.

⑥ 인터럽트 서비스 루틴 실행이 끝나면, 백업해둔 작업을 복구하여 마저 실행

서비스 루틴 식별 위한 정보
(서비스 루틴 시작 주소)

인터럽트를 처리하는
프로그램

02

인터럽트

Q. 모든 인터럽트가 CPU가 하던 일을 멈추게 하나요?

A. 아니요. CPU가 막을 수 있는 인터럽트maskable interrupt와, 막을 수 없는 인터럽트non maskable interrupt가 있습니다.

지금 끼어들어도 되나요?

어, 인터럽트다

인터럽트 받아도 돼? ➔ 안돼

➔ 단, 인터럽트 플래그가 안된다고 해도, 막을 수 없는 인터럽트 존재. (예: 정전, 하드웨어 고장 인터럽트)

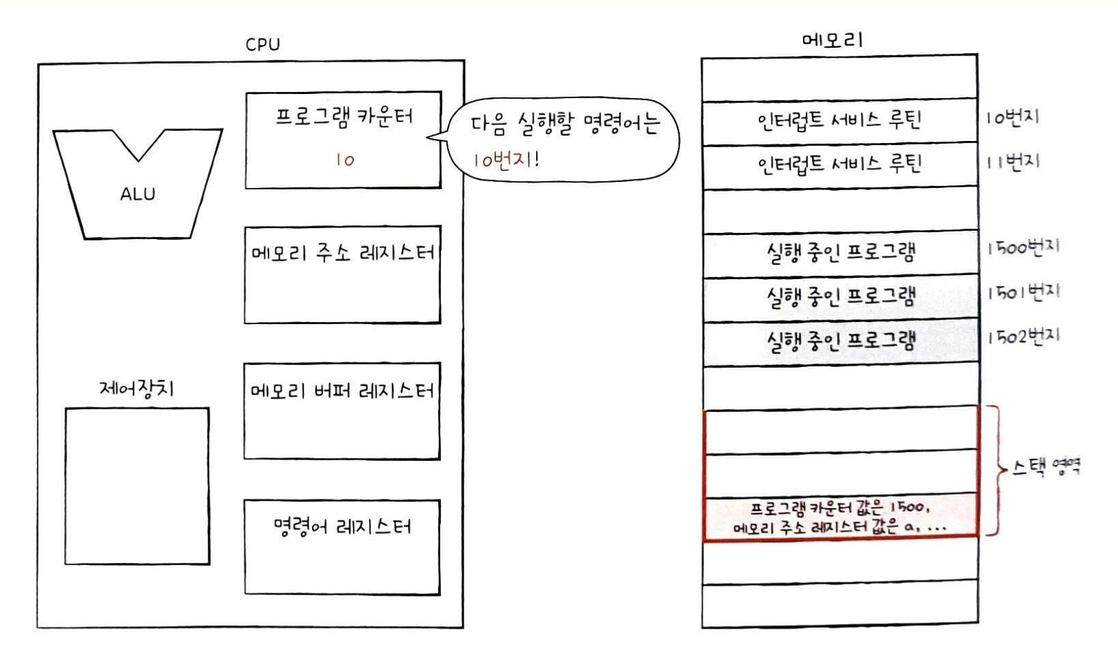
02

인터럽트

Q. 인터럽트 서비스 루틴을 실행하게 되면, 프로그램 카운터에 기존에 있던 주소는 덮어씌워지나요?

A. 네. 하지만 기존 작업은 전부 스택에 백업해둡니다.

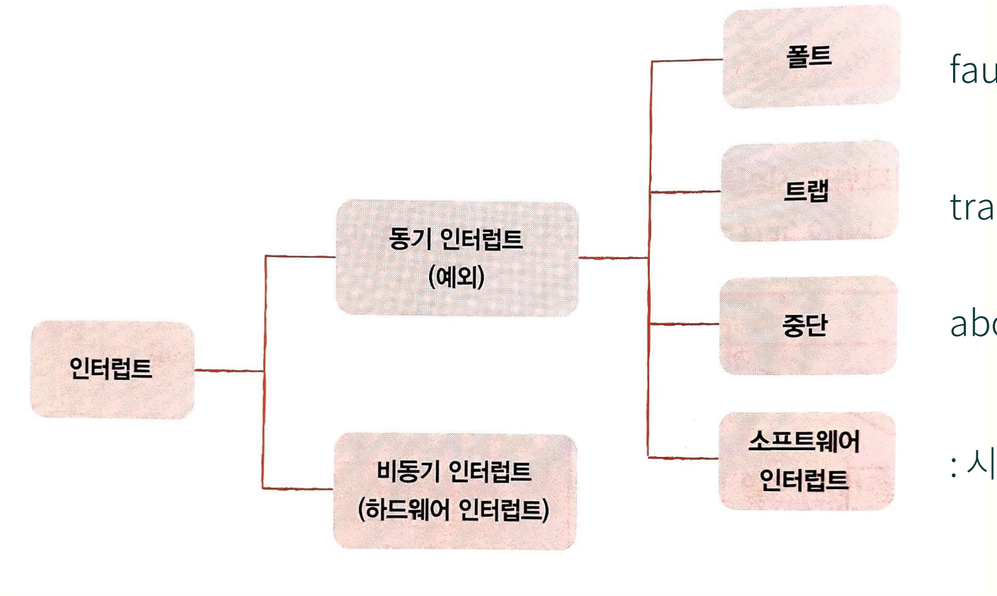
인터럽트 서비스 루틴을 실행한 후 스택에 저장해둔 값을 불러와 재개합니다.



예외의 종류

예외가 발생하면 CPU는 하던 일을 중단하고 예외를 처리한다.

그리고 기존 작업으로 돌아와 실행을 재개한다.



fault: 예외가 발생한 명령어부터 실행 재개

trap: 예외가 발생한 명령어의 다음 명령어부터 재개 (ex. 디버깅)

abort: 강제 중단할 수밖에 없는 심각한 오류

: 시스템 호출이 발생했을 때 (9장에서 다룸)

5장-1 빠른 CPU를 위한 설계

KEY WORD 클럭, 코어, 멀티코어, 스레드, 멀티스레드

01

클럭

clock

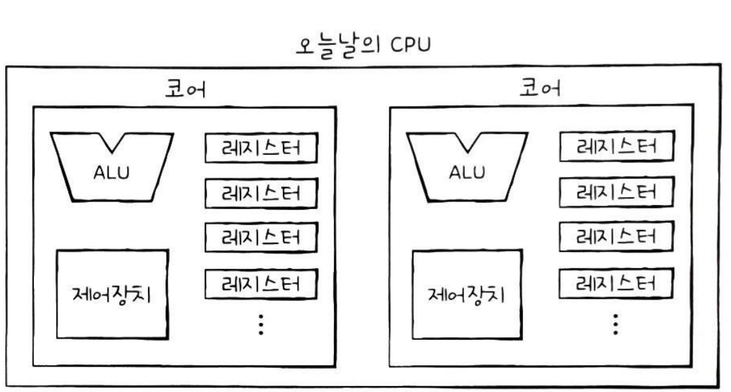
앞 장 내용: 컴퓨터 부품들은 '클럭 신호'에 맞춰 움직인다.

- 클럭 속도 : 단위는 헤르츠(Hz), 1초에 몇 번 클럭이 반복되는지를 나타냄.
  ex) 1초간 한 번 똑-딱 하면 클럭속도는 1 Hz
- 클럭 속도가 높을 수록 CPU는 1초에 명령어 사이클을 더 빠르게 반복함.
- 클럭 속도는 일정하지 않음. 고성능이 필요할 때는 클럭 속도를 높임.
  최대 클럭 속도를 강제로 끌어 올리는 방법을 **오버클럭킹**이라고 함.

➔ 단, 클럭 속도를 높이는 것은 CPU를 빠르게 만들지만 이것만으로는 한계가 있으며, 발열문제도 있음.

01

코어, 멀티코어



- 코어: 명령어를 실행하는 부품
- 멀티 코어 (멀티 프로세서): 일꾼이 여러 명 있는 것
- 재밌는 사실: 2.4GHz의 단일코어 CPU 보다 1.9GHz의 멀티코어 CPU가 성능이 좋음.

➔ **단, 처리량이 적은 경우엔 많은 코어 수가 도움이 되지 않는다.**
또한 코어 수에 비례하게 성능이 증가하지 않고,
코어에 연산이 적절히 분배되지 않으면 효율이 높지 않다.

01

스레드, 멀티 스레드

thread; 실행 흐름 단위

소프트웨어적 스레드

- 워드 프로세서 실행시,

  - 사용자로부터 내용 입력받기

  - 맞춤법 검사하기

  - 수시로 저장하기

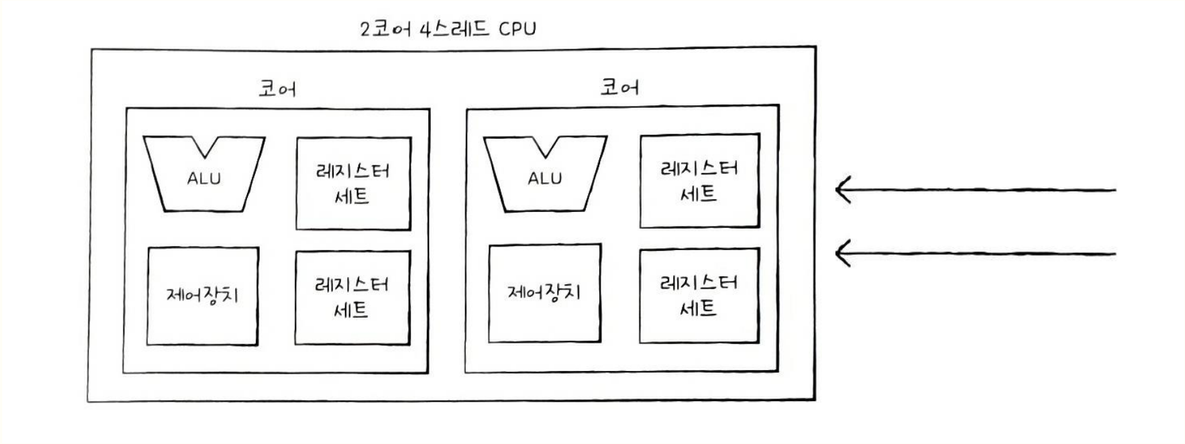
  를 동시에 해냄

01

스레드, 멀티 스레드

하드웨어적 스레드 (논리 프로세서)

- 하나의 코어가 동시에 처리하는 명령어 단위
- 마치 프로세서(코어)가 여러 개인 듯한 효과를 냄
- 핵심은 레지스터: 명령어를 처리하기 위한 레지스터를 여러 개 가지면 됨



02

오늘 배운 것

01. ‘명령어 사이클’이 CPU가 하나의 명령어를 처리하는 흐름이라는 것을 배움.

02. 그 흐름을 방해하는 것이 ‘인터럽트’라는 것을 배움.

03. 클럭, 코어, 스레드 개념을 배움.

04. 빠른 CPU를 만드는 설계 기법인 멀티코어와 멀티스레드를 이해함.

감사감사감자
감사감자감사

다음 발표 일정 : 5장 나머지, 내일 주히방