**Chapter 4 CPU의 작동 원리**

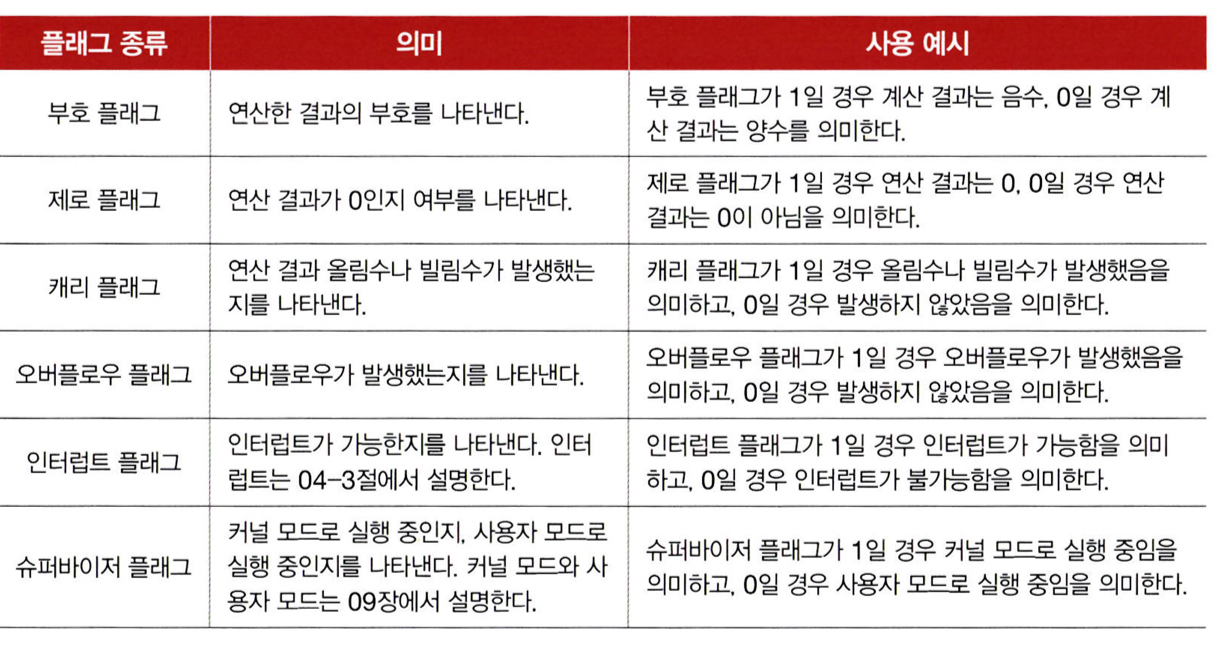
***4-1*** **ALU와 제어장치**

CPU 내부에는 계산을 담당하는 **ALU**, 명령어를 읽어 들이고 해석하는 **제어장치**, 작은 임시 저장 장치인 **레지스터**라는 구성 요소 있음

**<ALU>**

* ALU가 계산을 하기 위해서는 피연산자와 수행할 연산이 필요
* 레지스터를 통해 피연산자를 받아들이고, 제어장치로부터 수행할 연산을 알려주는 제어 신호를 받아들임
* 연산 수행 후 결괏값은 일시적으로 레지스터에 저장 -> 메모리에 접근하는 것은 프로그램 속도를 늦출 수 있기 때문
* 계산 결과와 더불어 **플래그**를 내보냄

**플래그**: 연산 결과에 대한 추가적인 상태 정보. 플래그들은 플래그 레지스터라는 레지스터에 저장



**<제어장치>**

**제어장치**: 제어 신호를 내보내고, 명령어를 해석하는 부품

**제어 신호**: 컴퓨터 부품들을 관리하고 작동시키기 위한 일종의 전기 신호

1. 제어장치는 클럭 신호를 받아들임

* **클럭**: 컴퓨터의 모든 부품을 일사불란하게 움직일 수 있게 하는 시간 단위
* 하나의 명령어가 여러 클럭에 걸쳐 실행될 수 있음

1. 제어장치는 해석해야 할 명령어를 받아들임

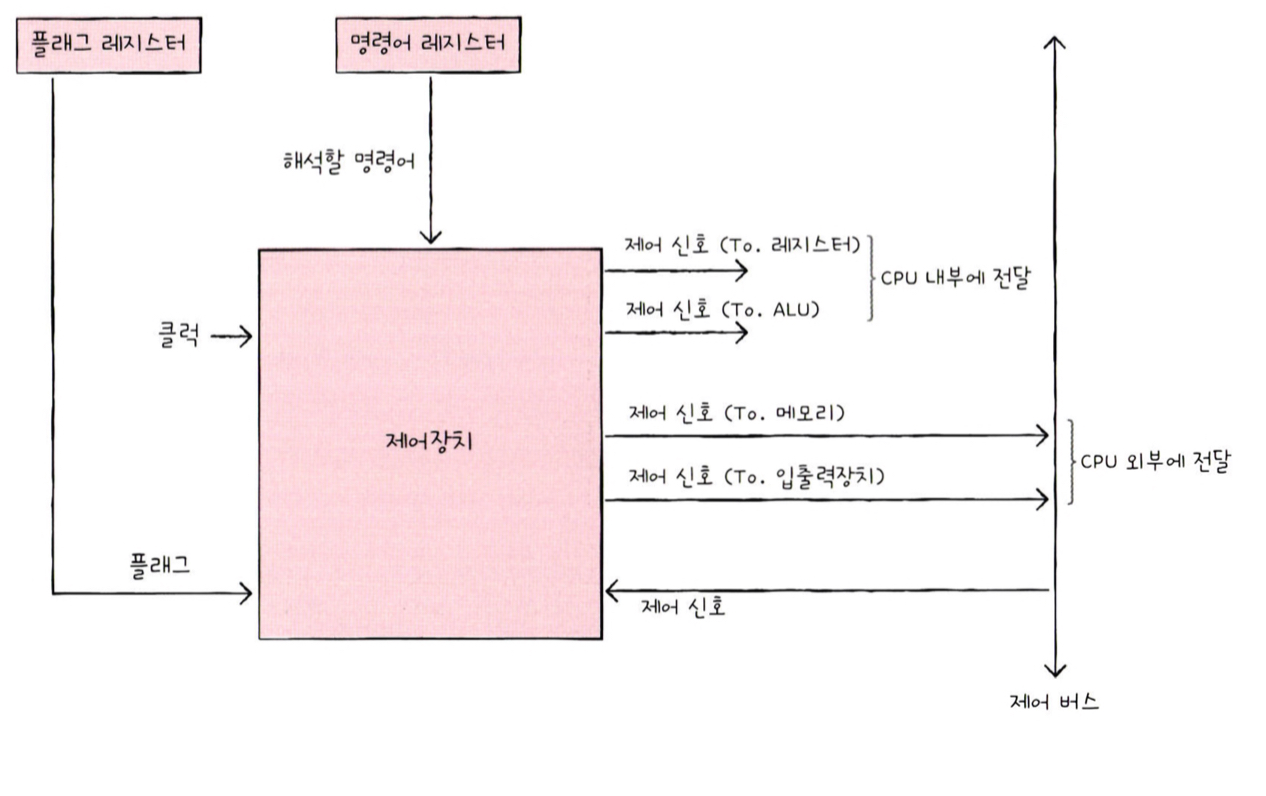
* CPU가 해석해야할 명령어는 명령어 레지스터에 저장
* 명령어 레지스터로부터 명령어를 받아들이고 해석한 뒤 제어신호를 발생시킴

1. 제어장치는 플래그 레지스터 속 플래그 값을 받아들임

* ALU 연산에 대한 추가적인 상태 정보인 플래그 값을 참고하여 제어 신호를 발생시킴

1. 제어장치는 시스템 버스, 그 중에서 제어 버스로 전달된 제어 신호를 받아들임

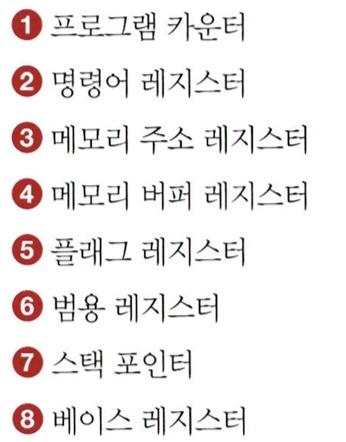
* 제어 신호는 CPU뿐만 아니라 입출력 장치를 비롯한 CPU 외부 장치도 발생시킴
* 제어장치가 CPU 외부에 제어 신호를 전달한다는 말은 곧, 제어 버스로 제어 신호를 내보낸다는 말
* 외부로 보내는 제어 신호에는 크게 메모리에 전달하는 제어 신호와 입출력장치에 전달하는 제어 신호가 있음
* CPU 내부에 전달하는 제어 신호는 수행할 연산을 지시하기 위해 ALU에 전달하는 신호와 레지스터 간에 데이터를 이동시키거나 레지스터에 저장된 명령어를 해석하기 위한 제어 신호가 있음

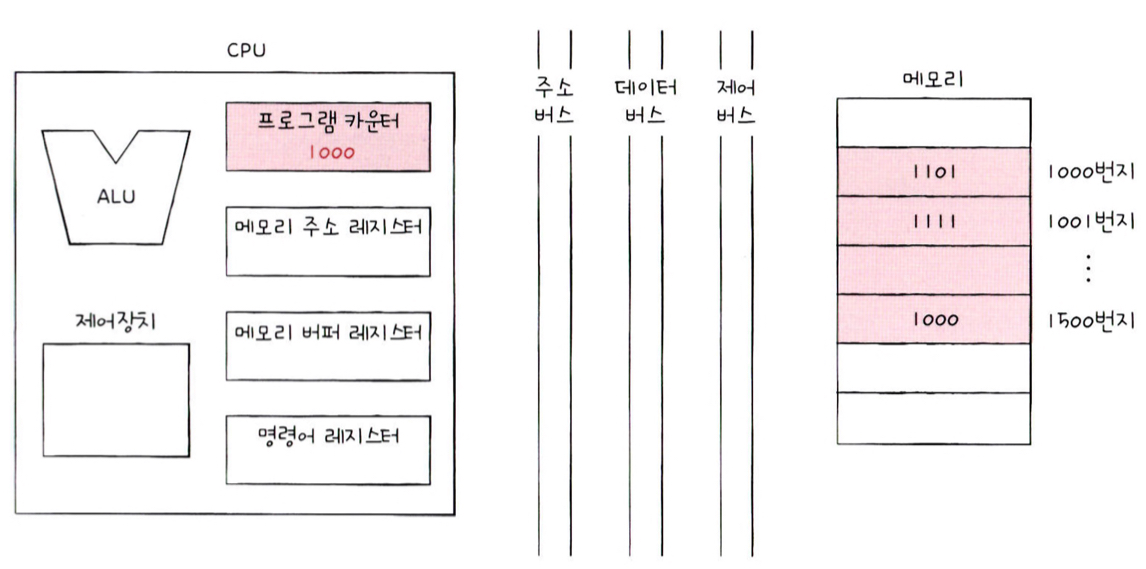


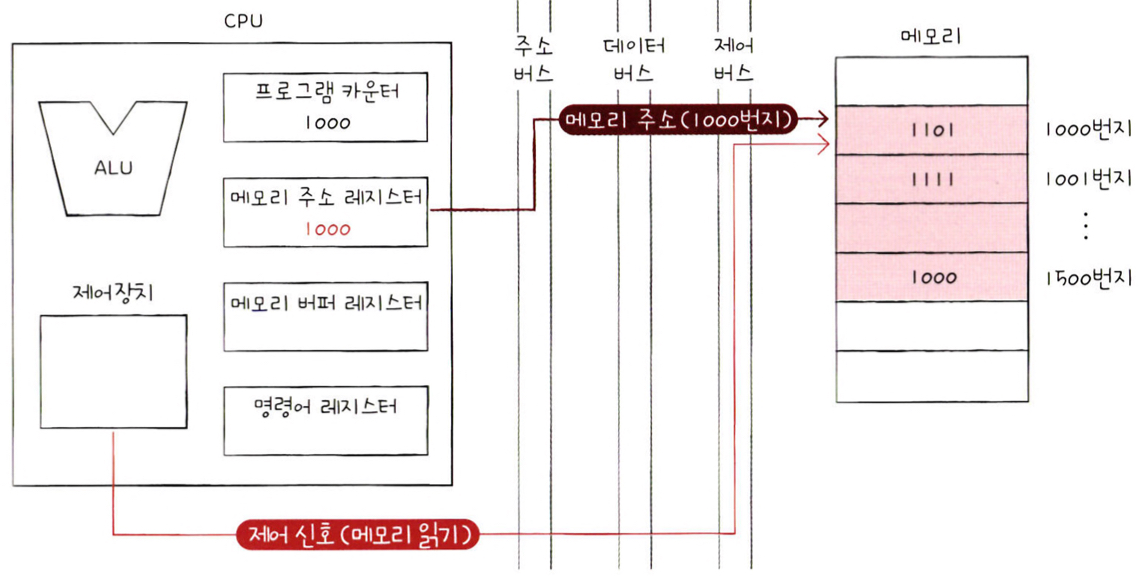
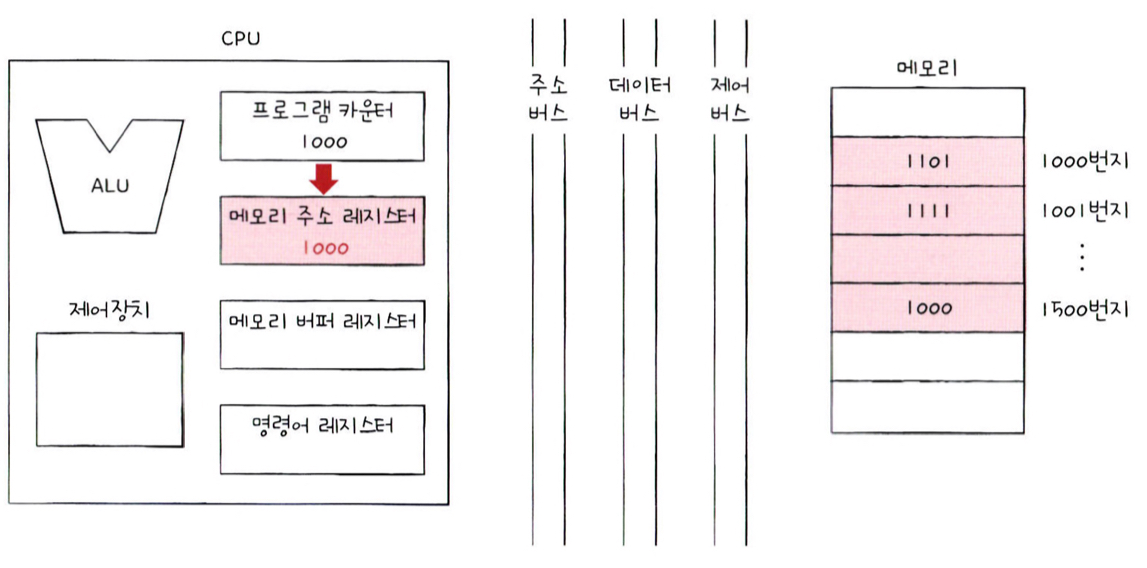
***4-2*** **레지스터**

* 프로그램 속 명령어와 데이터는 실행 전후로 반드시 레지스터에 저장
* CPU 안에는 다양한 레지스터들이 있고, 각기 다른 역할을 가짐

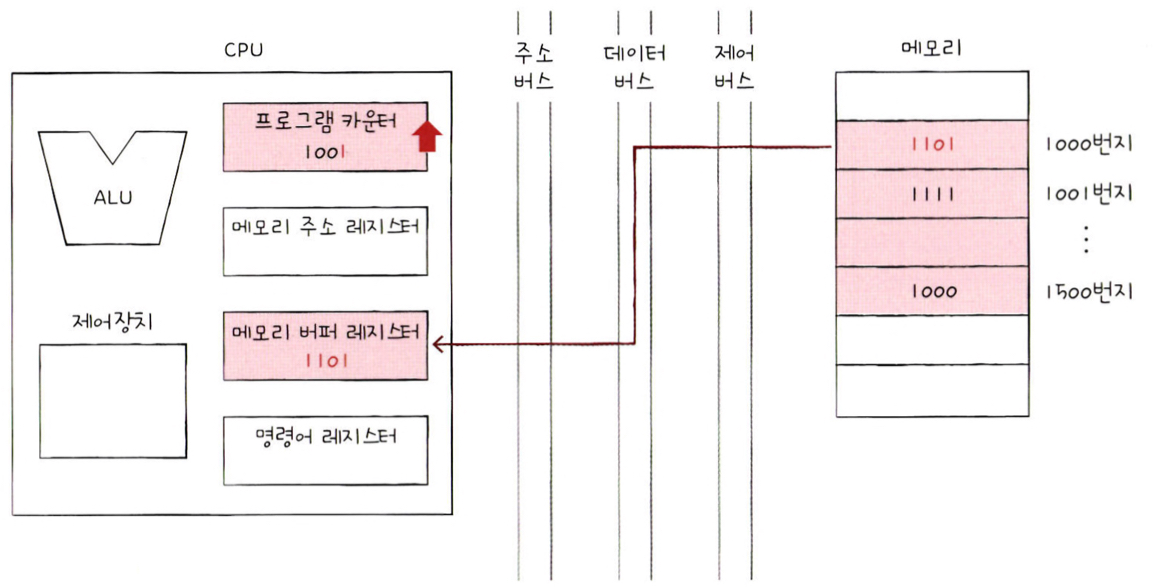
**<반드시 알아야 할 레지스터>**



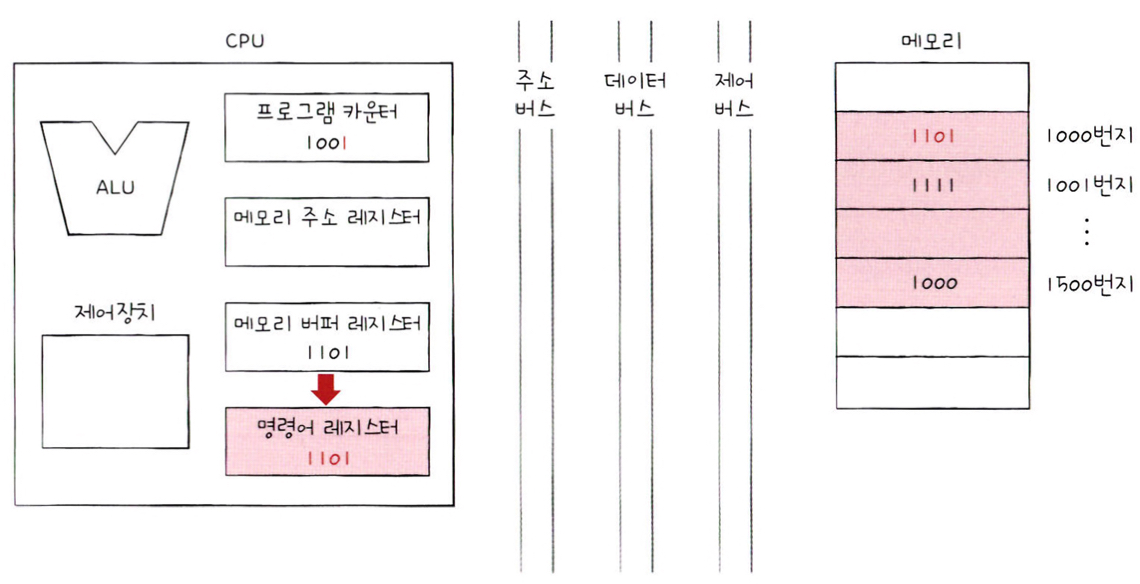
**프로그램 카운터**(명령어 포인터): 메모리에서 가져올 명령어의 주소를 저장 

**메모리 주소 레지스터**: 메모리의 주소를 저장하는 레지스터

**메모리 버퍼 레지스터**: 메모리와 주고받을 값(데이터, 명령어)을 저장하는 레지스터



**명령어 레지스터**: 방금 메모리에서 읽어 들인 명령어를 저장하는 레지스터



**범용 레지스터**: 다양하고 일반적인 상황에서 자유롭게 사용할 수 있는 레지스터. CPU 안에는 여러 개의 범용 레지스터들이 있음

**플래그 레지스터**: 연산 결과 또는 CPU 상태에 대한 부가적인 정보를 저장하는 레지스터

**<특정 레지스터를 이용한 주소 지정 방식(1): 스택 주소 지정 방식>**

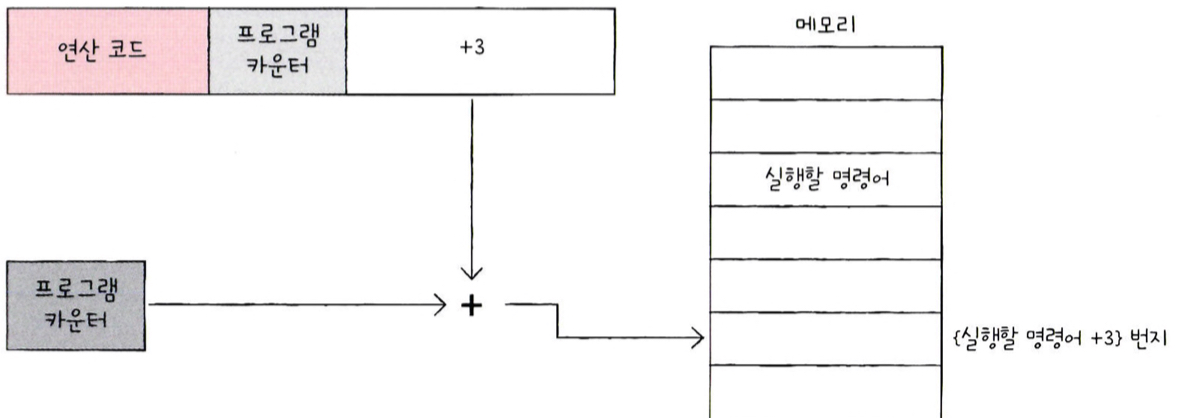
**스택 주소 지정 방식**: 스택과 스택 포인터를 이용한 주소 지정 방식

* **스택**: 한쪽 끝이 막혀있는 통과 같은 저장 공간. 가장 최근에 저장하는 값부터 꺼냄
* **스택 포인터**: 스택의 꼭대기를 가리키는 레지스터
* **스택 영역**: 메모리안에서 다른 주소 공간과는 다르게 스택처럼 사용하기로 암묵적으로 약속된 영역

**<특정 레지스터를 이용한 주소 지정 방식(2): 변위 주소 지정 방식>**

**변위 주소 지정 방식**: 오퍼랜드 필드의 값과 특정 레지스터의 값을 더하여 유효 주소를 얻어내는 주소 지정 방식

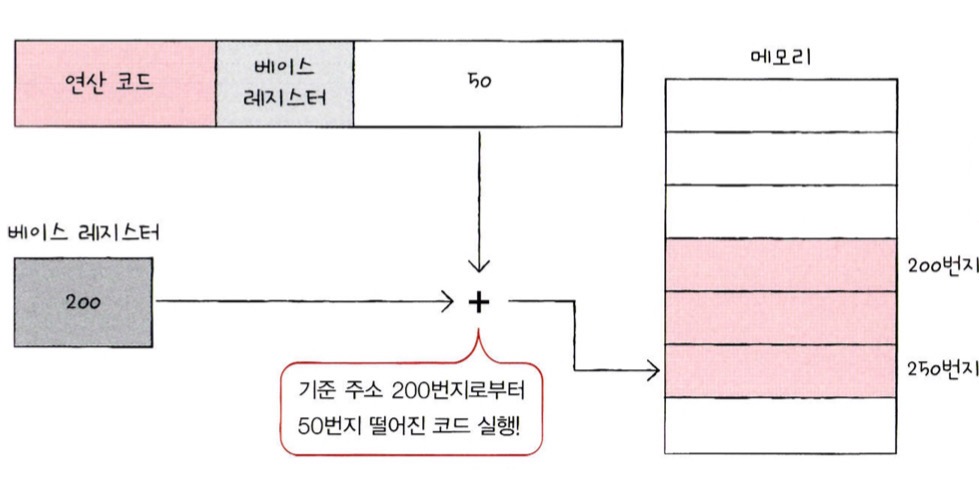
* **상대 주소 지정 방식**: 오퍼랜드와 프로그램 카운터의 값을 더하여 유효 주소를 얻는 방식. if문과 유사하게 분기하여 특정 주소의 코드를 실행할 때 사용



* **베이스 레지스터 주소 지정 방식**: 오퍼랜드와 베이스 레지스터의 값을 더하여 유효 주소를 얻는 방식
  + 베이스 레지스터: 기준 주소
  + 오퍼랜드: 기준 주소로부터 떨어진 거리

즉 베이스 레지스터 속 기준 주소로부터 얼마나 떨어져 있는 주소에 접근할 것인지를

연산하여 유효 주소를 얻어내는 방식



***4-3*** **명령어 사이클과 인터럽트**

**<명령어 사이클>**

**명령어 사이클**: 프로그램 속 각각의 명령어들은 일정한 주기가 반복되며 실행되는데 이 주기를 명령어 사이클이라고 함

* **인출 사이클**: 메모리에 있는 명령어를 CPU로 가지고 오는 단계
* **실행 사이클**: CPU로 가져온 명령어를 실행하는 단계 (제어장치가 명령어 레지스터에 담긴 값을 해석하고 제어 신호를 발생시킴)
* **간접 사이클**: 간접 주소 지정 방식일 경우 명령어를 CPU로 가져왔다 하더라도 명령어를 실행하기 위해 메모리 접근을 한번 더 해야 하는 단계
* **인터럽트 사이클**: 인터럽트가 발생한 경우 인터럽트를 처리하는 단계

**<인터럽트>**

**인터럽트**: CPU의 작업을 방해하는 신호

* **동기 인터럽트(예외)**: CPU에 의해 발생하는 인터럽트 ex) 프로그래밍상의 오류
* **비동기 인터럽트(하드웨어 인터럽트)**: 주로 입출력장치에 의해 발생하는 인터럽트. 인터럽트 받을 때까지 다른 작업 처리 가능 ex) 세탁기 완료 알림, 전자레인지 조리 완료 알림

**하드웨어 인터럽트** **처리 순서**

1. 입출력장치는 CPU에 **인터럽트 요청 신호**를 보냄
2. CPU는 실행 사이클이 끝나고 명령어를 인출하기 전 항상 인터럽트 여부를 확인
3. CPU는 인터럽트 요청을 확인하고 **인터럽트 플래그**를 통해 현재 인터럽트를 받아들일 수 있는지 여부를 확인
4. 인터럽트를 받아들일 수 있다면 CPU는 지금까지의 작업을 백업
5. CPU는 **인터럽트 벡터**를 참조하여 **인터럽트 서비스 루틴**을 실행
6. 인터럽트 서비스 루틴 실행이 끝나면 4에서 백업해 둔 작업을 복구하여 실행을 재개

**인터럽트 요청 신호**: CPU에게 지금 끼어들어도 되는지 묻는 것

**인터럽트 플래그**: 플래그가 가능으로 설정되어있으면 인터럽트 요청 신호를 받아들이고 인터럽트를 처리, 불가능으로 설정되어 있다면 해당 요청 무시

* 무시할 수 없는 인터럽트 존재 ex) 정전, 하드웨어 고장

**인터럽트 서비스 루틴(인터럽트 핸들러)**

* 인터럽트를 처리하기 위한 프로그램
* 인터럽트가 발생했을 때 해당 인터럽트를 어떻게 처리하고 작동해야 할지에 대한 정보로 이루어진 프로그램
* 인터럽트를 처리하는 방법은 입출력장치마다 다르므로 각기 다른 인터럽트 서비스 루틴을 가짐
* 메모리에 여러 개의 인터럽트 서비스 루틴이 저장
* CPU는 수많은 인터럽트 서비스 루틴을 구분하기 위해 **인터럽트 벡터**를 이용

**인터럽트 벡터**

* 인터럽트 서비스 루틴을 식별하기 위한 정보
* 인터럽트 벡터를 알면 인터럽트 서비스 루틴의 시작 주소를 알 수 있음

인터럽트 서비스 루틴 실행 전 **백업**: 프로그램 카운터 값 등 현재 프로그램을 재개하기 위해 필요한 모든 내용을 스택에 백업

인터럽트 서비스 루틴 **실행**: 그리고 나서 인터럽트 서비스 루틴의 시작 주소가 위치한 곳으로 프로그램 카운터 값을 갱신하고 인터럽트 서비스 루틴을 실행

이전 작업 **재개**: 인터럽트를 처리하고 나면 스택에 저장해 둔 값을 다시 불러온 뒤 이전까지 수행하던 작업을 재개

**예외의 종류**

* **폴트**: 예외를 처리한 직후 예외가 발생한 명령어부터 실행을 재개하는 예외
* **트랩**: 예외를 처리한 직후 예외가 발생한 다음 명령어부터 실행을 재개하는 예외
* **중단**: CPU가 실행 중인 프로그램을 강제로 중단시킬 수밖에 없는 심각한 오류를 발견했을 때 발생하는 예외
* **소프트웨어 인터럽트**: 시스템 호출이 발생했을 때 나타남