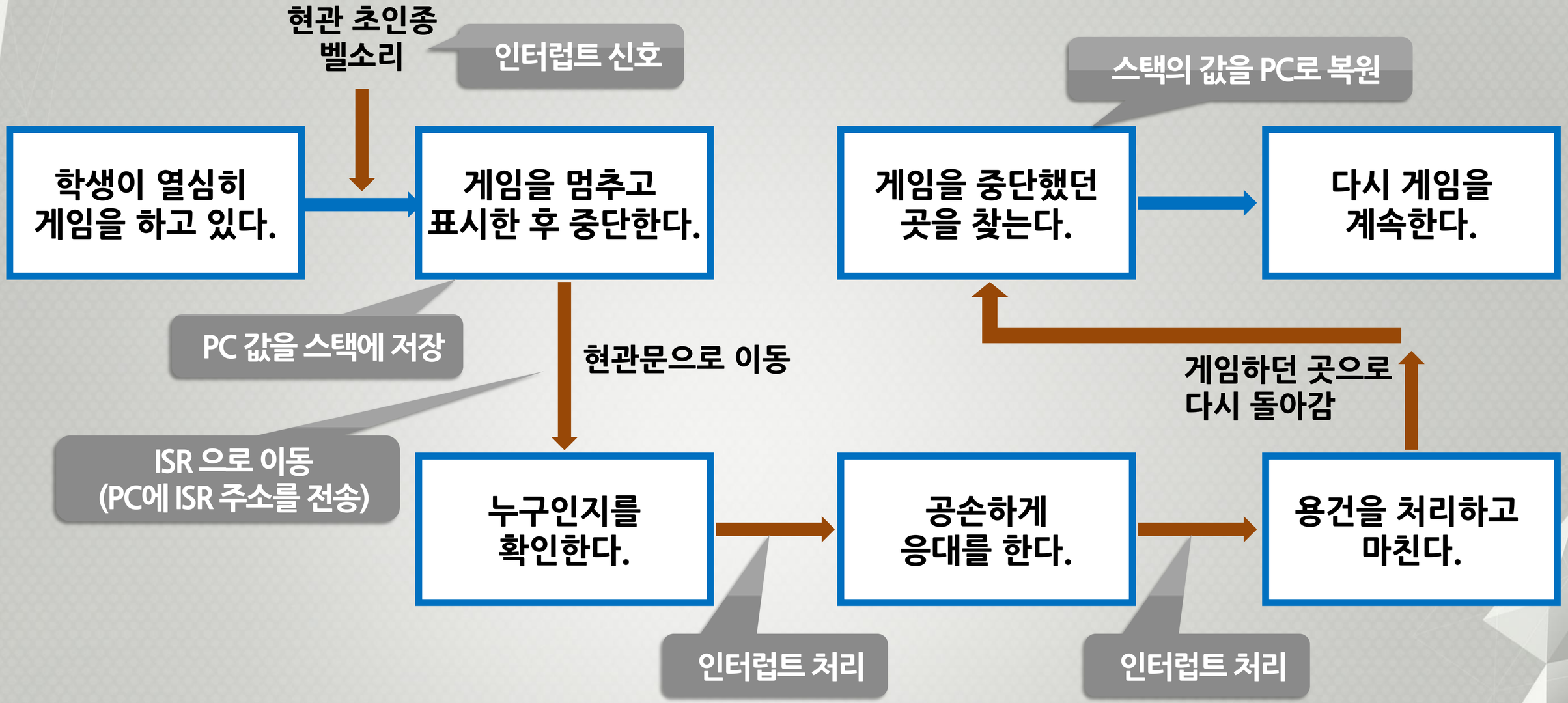
학습목표

- 인터럽트 개념 및 처리 방식을 설명할 수 있다.
- 서브루틴 처리 과정(CALL & RETURN)을 설명할 수 있다.
- 명령어 파이프라이닝 기법을 설명할 수 있다.

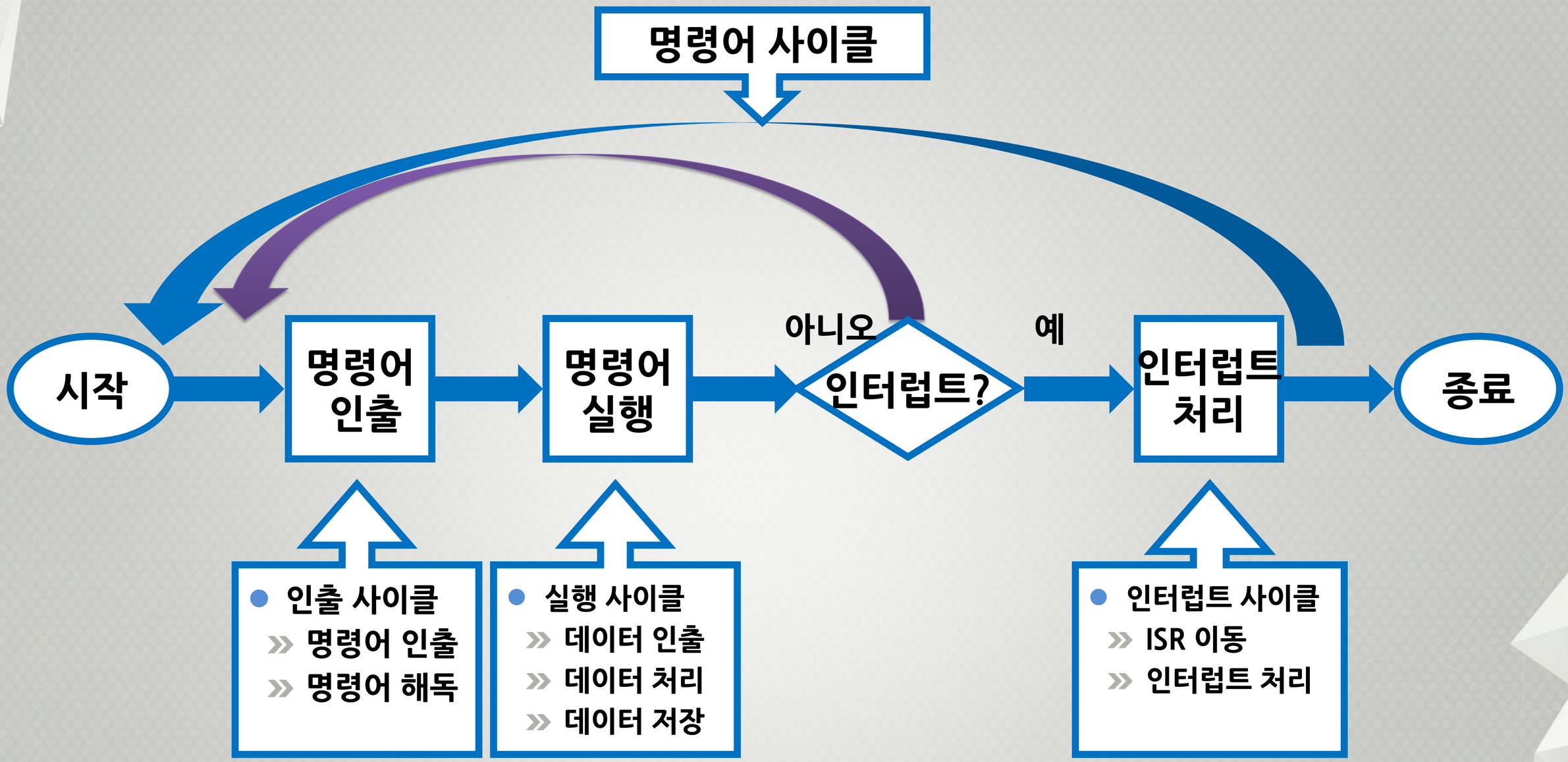
학습내용

- 인터럽트
- 서브루틴
- 명령어 파이프라이닝

인터럽트 신호처리의 기본적 개념



명령어 및 인터럽트 사이클



인터럽트(Interrupt)

CPU가 정상적인 프로그램 실행 중에 또 다른 프로그램의 실행요구로 현재 실행 중인 프로그램을 중단시키고 요구된 프로그램을 실행하는 것을 **인터럽트**라고 한다.

- CPU가 프로그램 실행 중에 인터럽트 신호 처리를 요청
  - CPU는 원래의 프로그램 수행을 중단한다.
  - 요구된 인터럽트를 위한 서비스 프로그램을 먼저 수행한다.
    - 어떤 장치가 인터럽트를 요구했는지 확인한다.
    - 해당 인터럽트 서비스 루틴을 호출한다.
  - 인터럽트 처리가 끝나면 본 프로그램으로 복귀한다.

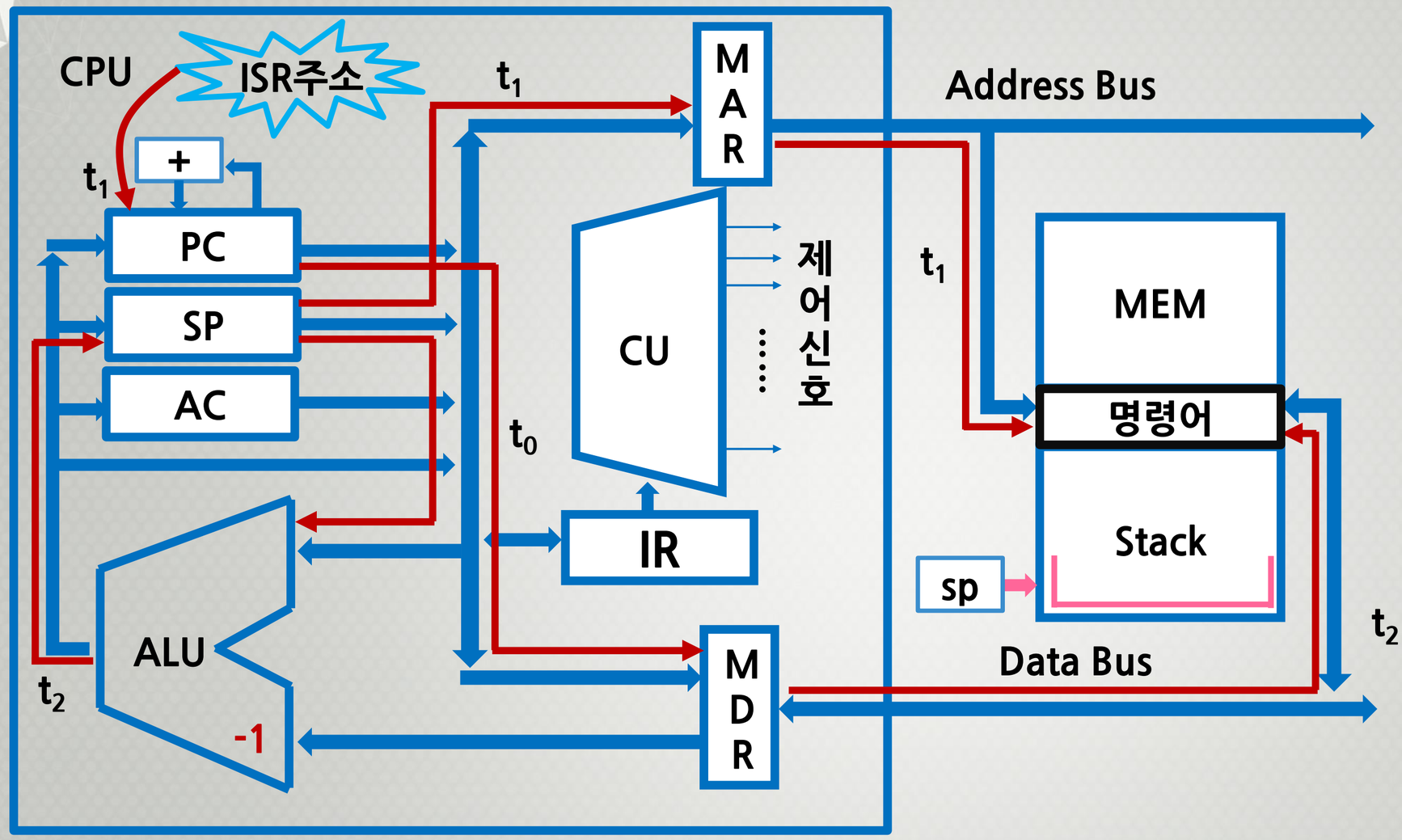
인터럽트(Interrupt)

인터럽트 벡터 테이블 (Interrupt Vector Table)	• 다양한 인터럽트 신호를 처리하는 인터럽트 서비스 루틴의 시작 주소를 포함
인터럽트 서비스 루틴 (Interrupt Service Routine)	• 인터럽트를 처리하기 위하여 수행하는 프로그램 루틴

- CPU의 인터럽트 처리 동작
  - 현재 실행중인 명령어 실행을 끝낸 즉시, 다음에 실행할 명령어의 주소(PC의 내용)를 스택(Stack)에 저장한다.
  - 일반적으로 스택은 주기억장치의 특정 부분을 지정하여 사용한다.
  - 인터럽트 서비스 루틴을 호출하기 위하여 그 루틴의 시작 주소를 PC에 저장한다.
    - 시작 주소는 인터럽트를 요구한 장치로부터 전송되거나 미리 결정된 주소 값으로 결정한다.

인터럽트 사이클(Interrupt Cycle)

- 인터럽트 사이클에서 클럭주기(t_0, t_1, t_2) 에 따른 주소값 및 명령어의 흐름도



인터럽트 사이클(Interrupt Cycle)

인터럽트 사이클의 마이크로 연산(Micro-operation)

t_0 : MDR ← PC

t_1 : MAR ← SP, PC ← ISR시작주소

t_2 : M[MAR] ← MDR, SP ← SP - 1

↳ 여기서, t_0, t_1, 및 t_2는 CPU 클럭주기

인터럽트 사이클(Interrupt Cycle)

인터럽트 사이클의 마이크로 연산(Micro-operation)

클럭 t_0	현재 PC의 주소정보를 MDR로 전송한다.
클럭 t_1	SP의 내용이 MAR로 전송되고, PC의 내용은 인터럽트 서비스 루틴의 시작 주소로 변경된다.
클럭 t_2	MDR에 저장되어 있던 원래 PC의 내용이 스택에 저장된다.

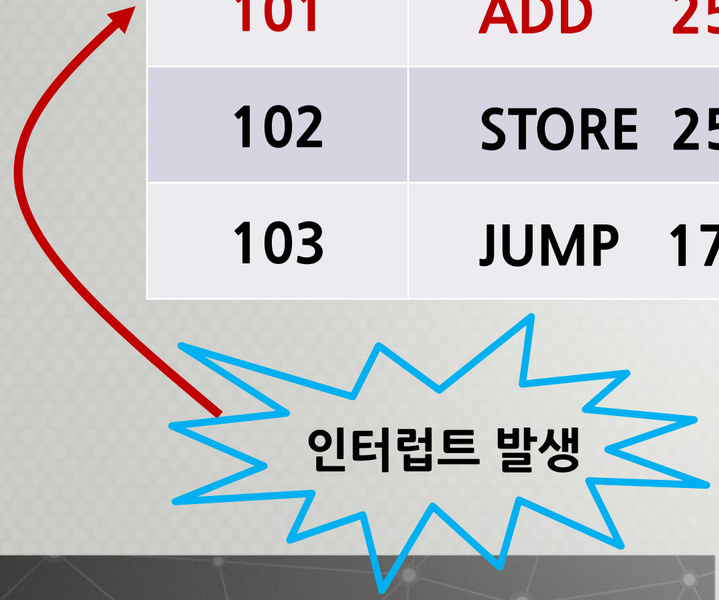
예)

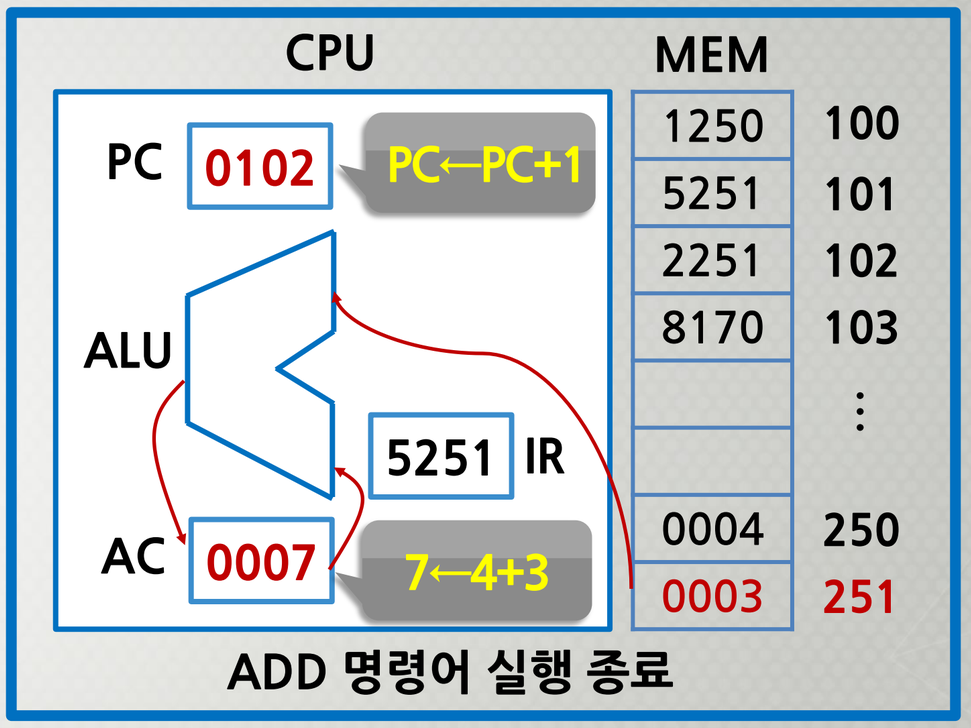
- CPU 클럭이 2GHz 인 경우 클럭 주기 및 인터럽트 사이클 수행시간
  - 클럭 주기 = 1 sec ÷ 2×10^9 = 0.5ns
  - 실행사이클 시간 = 0.5ns × 3 = 1.5ns

인터럽트 사이클(Interrupt Cycle)

- 예) 아래 프로그램의 두 번째 명령어인 ADD 251 명령어가 실행되는 동안에 인터럽트가 발생하였다. SP=900, 인터럽트 서비스 루틴의 시작 주소=500번지라고 가정했을 때의 처리과정

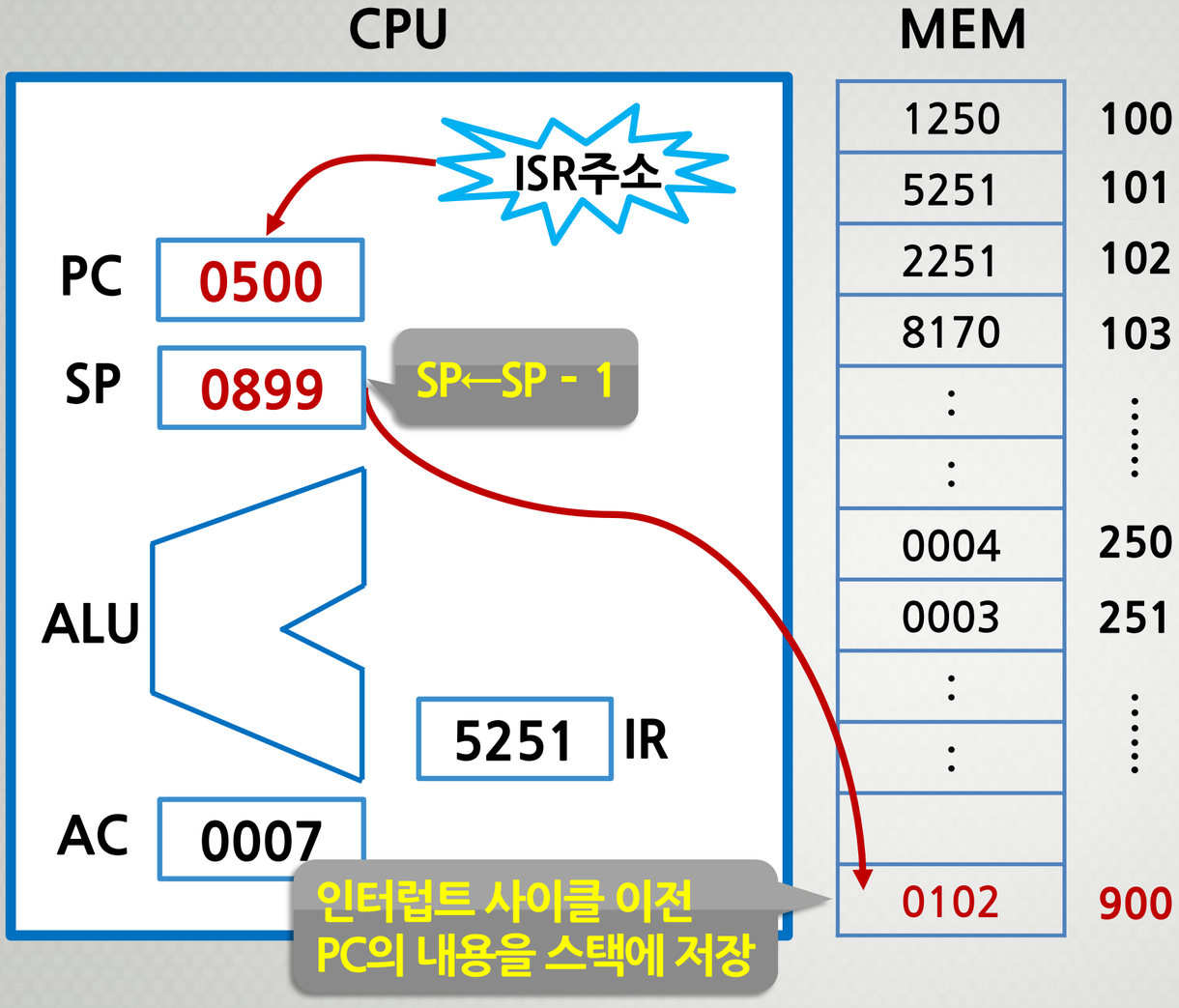
주소	명령어	기계 코드
100	LOAD 250	1250
101	ADD 251	5251
102	STORE 251	2251
103	JUMP 170	8170





인터럽트 사이클(Interrupt Cycle)

인터럽트 사이클 종료 상태



다중 인터럽트(Multiple Interrupt)

인터럽트 서비스 루틴을 수행하는 동안에 **또 다른 인터럽트가 발생**하는 것을 말한다.

- 다중 인터럽트의 처리방식
  - 인터럽트 플래그 (Interrupt Flag) 를 이용하는 방식
    - CPU가 인터럽트 요청을 처리하는 도중에는 새로운 인터럽트 요청이 발생하더라도 이를 수행하지 않도록 하는 방식이다.
    - 이 때 새로운 인터럽트 요청은 플래그를 불가능(Disabled) 상태로 설정하고, 대기 상태로 설정한다.
    - 이것은 가능(Enable) 상태로 변경되면 그 때 인식된다.
    - 시스템 운영상 중요한 프로그램 수행이나, 도중에 중단할 수 없는 데이터 입출력 동작 등을 위한 인터럽트를 처리하는데 주로 사용한다.

다중 인터럽트(Multiple Interrupt)

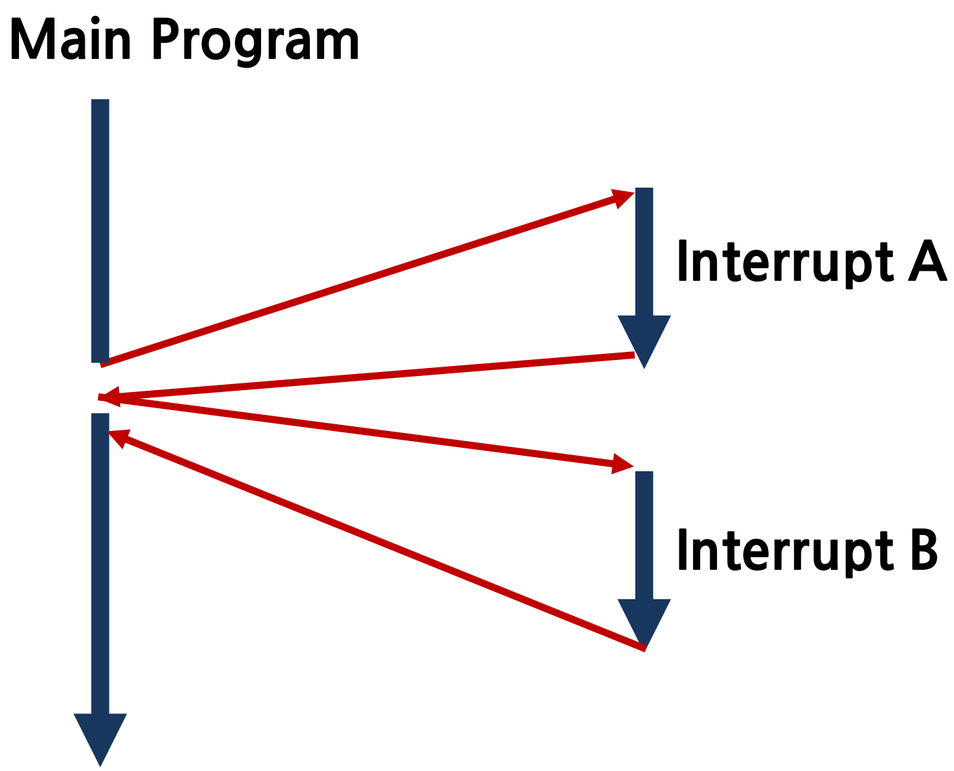
인터럽트 서비스 루틴을 수행하는 동안에 **또 다른 인터럽트가 발생**하는 것을 말한다.

- 다중 인터럽트의 처리방식
  - 우선순위(priority)를 정하는 방식
    - 우선순위가 낮은 인터럽트 요청을 처리하는 동안에 우선순위가 더 높은 인터럽트 요청이 들어오면, 현재 진행되는 낮은 순위의 요청은 중단되고 높은 순위의 요청이 처리되도록 하는 방식이다.

다중 인터럽트(Multiple Interrupt)

- 다중 인터럽트 처리방식에 따른 제어의 흐름도

인터럽트 플래그(Interrupt Flag) 이용



다중 인터럽트(Multiple Interrupt)

- 다중 인터럽트 처리방식에 따른 제어의 흐름도