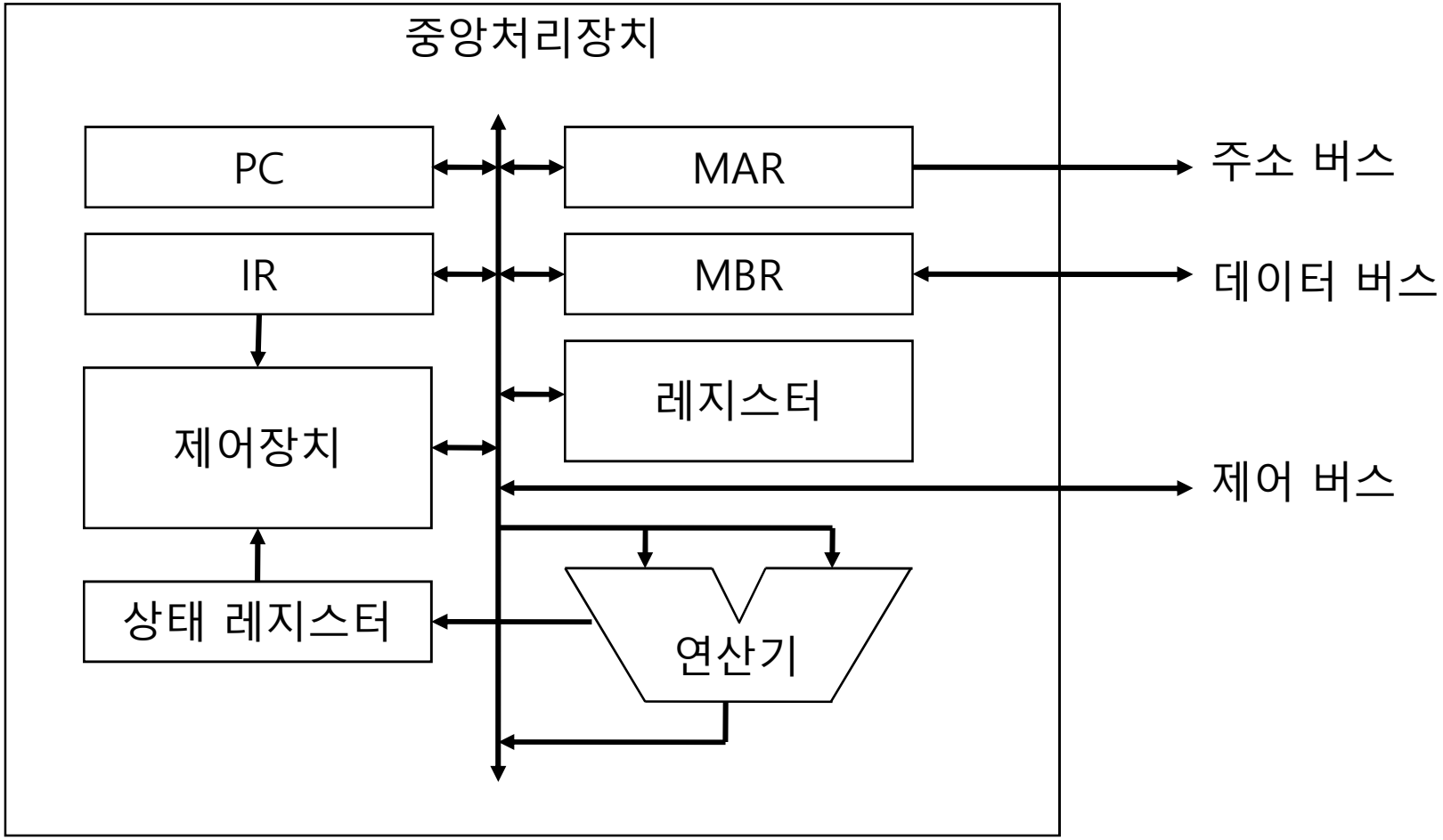
중앙처리장치

- 중앙처리장치의 구성 요소
- 레지스터의 종류
- 인터럽트
- 명령어 사이클

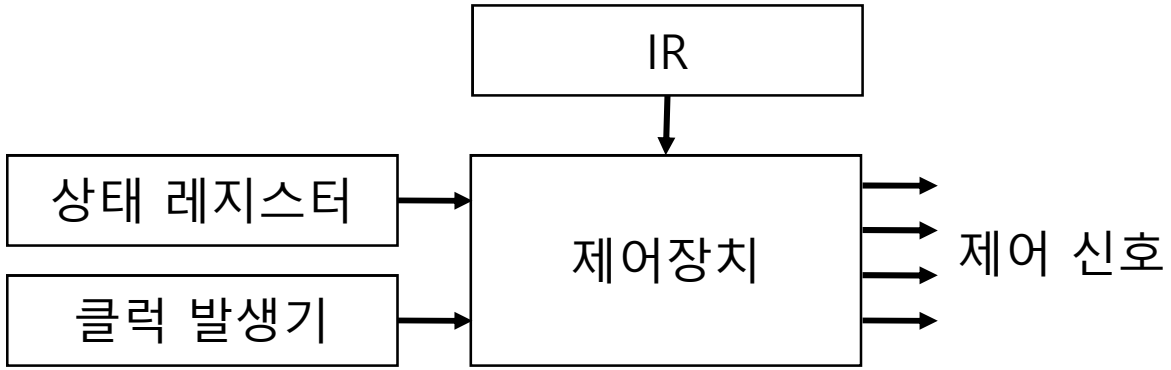
중앙처리장치 구성 요소



제어장치

- 제어장치의 기능
  - □ 순서 제어(sequence control)
    - 명령어를 순차적으로 실행한다.
    - 계속해서 명령어 사이클을 진행시킨다.
  - □ 동작 제어(operation control)
    - 각 명령어를 실행한다.

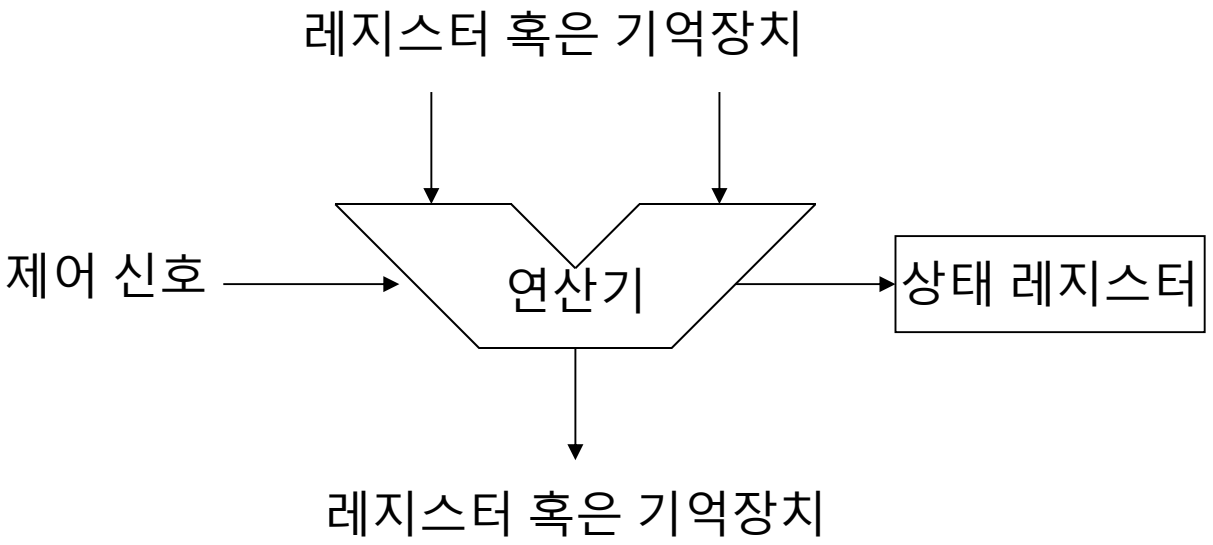
- 제어장치의 입력과 출력



연산기

- 연산기의 기능
  - 모든 데이터 처리 담당
  - 제어 신호에 의하여 동작 결정
  - 연산 수행 결과를 상태 레지스터에 저장

- 연산기의 구조



상태 레지스터 사용 예

```
1.          cmp    X, Y        // compare X and Y
2.          brg    lable1      // branch to lable1 if greater than 0
3.          inc    X           // increment x
4.   lable1:   …                // branch to here
```

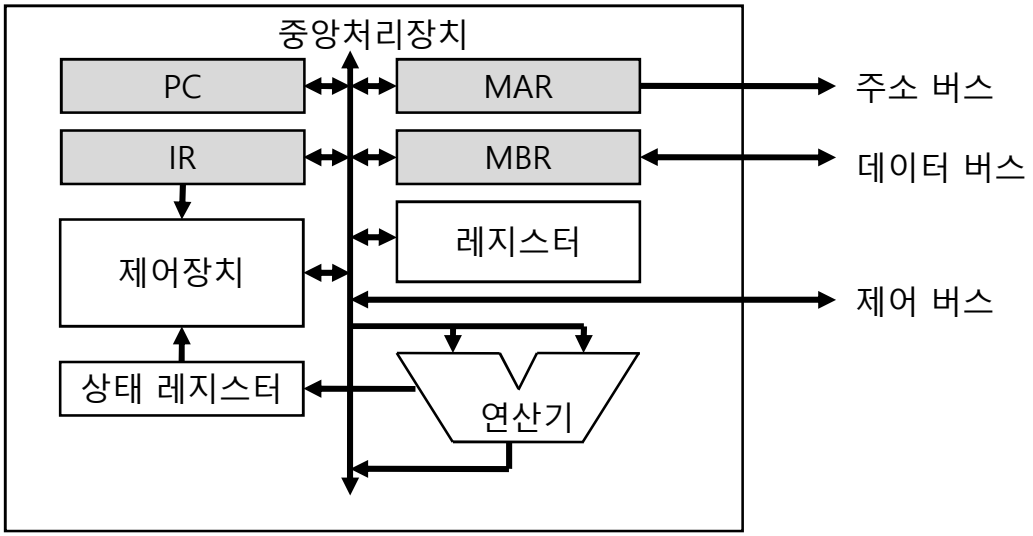
- Line 1: 비교 결과를 상태 레지스터에 저장
- Line 2: 상태 레지스터를 참조하여 분기 여부 결정

레지스터

- 레지스터
  - 중앙처리장치의 내부 기억장치
  - 기억장치 중 속도가 가장 빠르다.

- 레지스터 종류
  - 제어용: PC, IR, MAR, MBR
  - 명령어 실행용: 데이터 레지스터, 주소 레지스터, 범용 레지스터

제어용 레지스터



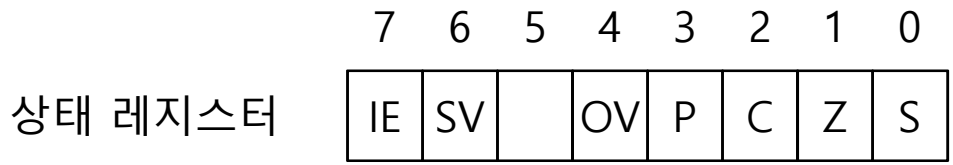
데이터 읽기

1. MAR ← 주소 레지스터
2. MBR ← Mem[MAR]
3. 데이터 레지스터 ← MBR

데이터 쓰기

1. MAR ← 주소 레지스터,
   MBR ← 데이터 레지스터
2. Mem[MAR] ← MBR

상태 레지스터



상태 레지스터

- 플래그(flag)의 모임
- 조건 플래그
  - 연산 결과 반영
- 제어 플래그

플래그 종류

- 조건 플래그
  - 부호 (S, sign flag)
  - 제로 (Z, zero flag)
  - 자리올림수 (C, carry flag)
  - 패리티 (P, parity flag)
  - 오버플로우 (OV, overflag flag)
- 제어 플래그
  - 인터럽트 (IE, Interrupt Enable flag)
  - 운영체제 (SV, supervisor mode flag)

ex 4-1 조건 플래그

```
  0110_0101
+ 0101_0100
-----------
  1011_1001
```

- S = 1: sign flag
- Z = 0: non zero
- C = 0: no carry
- P = 1: number of 1= 5, even parity
- OV = 1: positive addition → negative result

명령어 실행용 레지스터

- 데이터 레지스터 (Data Register)
  - 중앙처리장치가 처리하는 데이터 임시 저장
  - 누산기 (accumulator)

- 주소 레지스터 (Address Register)
  - 기억장치 주소 저장
  - 주소지정방식(6.3절): 중앙처리장치는 기억장치를 다양한 방법으로 액세스 한다.
  - 스택 포인터, 베이스 레지스터, 인덱스 레지스터

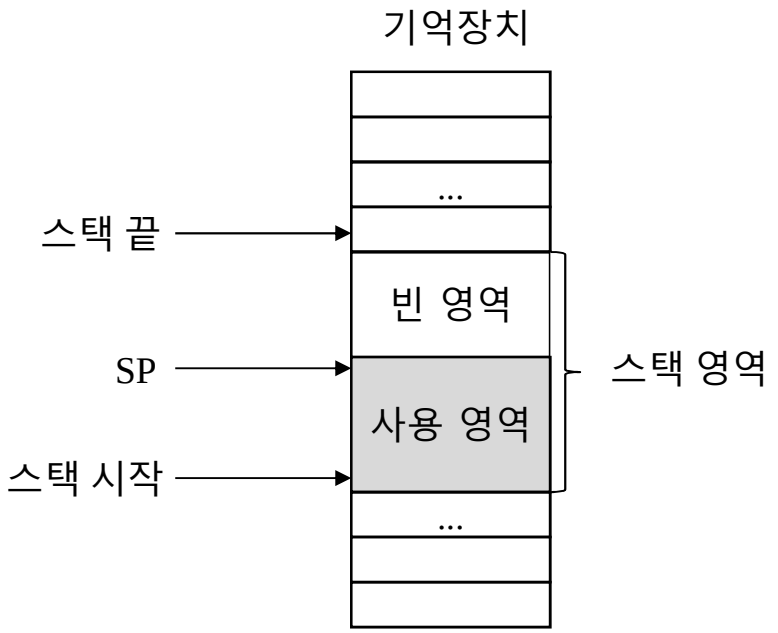
- 범용 레지스터 (General Purpose Register)
  - 데이터 저장용 , 또는 주소 저장용으로 사용 가능
  - 범용 레지스터 중 일부를 특별한 용도로 지정하는 경우도 있다.
    - 베이스 레지스터, 또는 인덱스 레지스터
  - 통상 8 개 ~ 32 개

스택 포인터

- 스택
  - Last-In-First-Out
  - 기억장치의 일부를 스택 영역으로 활용

- SP (Stack Pointer)
  - Point to stack top

- 스택 동작
  - PUSH: 스택에 데이터 추가
  - POP: 스택에서 데이터 제거



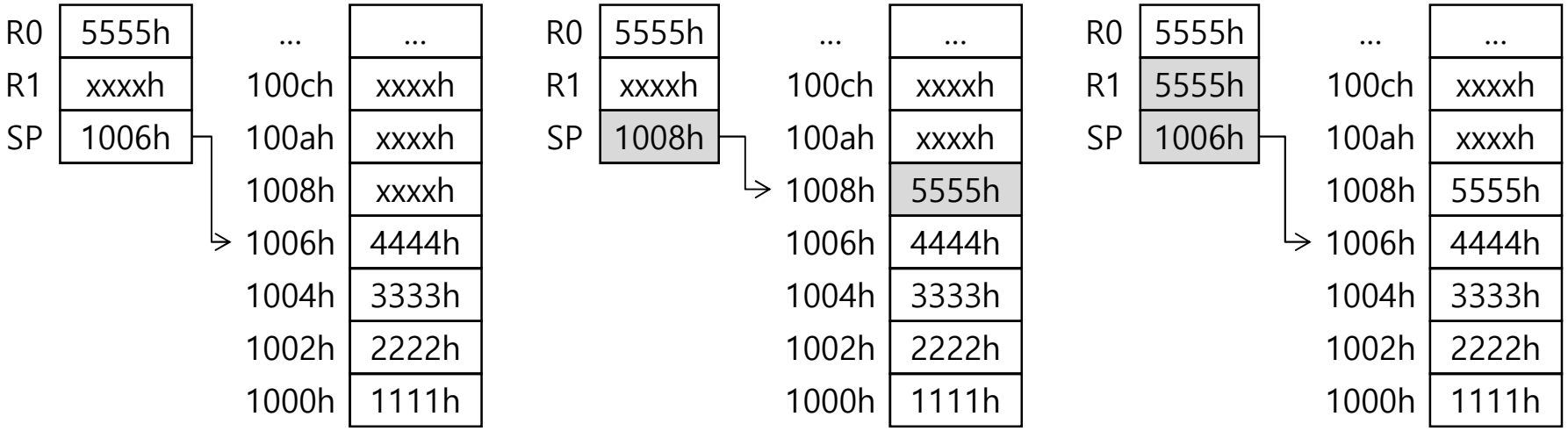
스택 동작

- 스택의 데이터 크기 = 레지스터 크기 = 단어 크기

PUSH 오퍼런드:
SP ← SP + [단어 크기]
Mem[SP] ← 오퍼런드

POP 오퍼런드:
오퍼런드 ← Mem[SP]
SP ← SP - [단어 크기]

- 예: 기억장치 바이트 단위로 구성, 데이터 16 비트



(a) 초기 상태 (b) PUSH R0 실행 후 (c) POP R1 실행 후

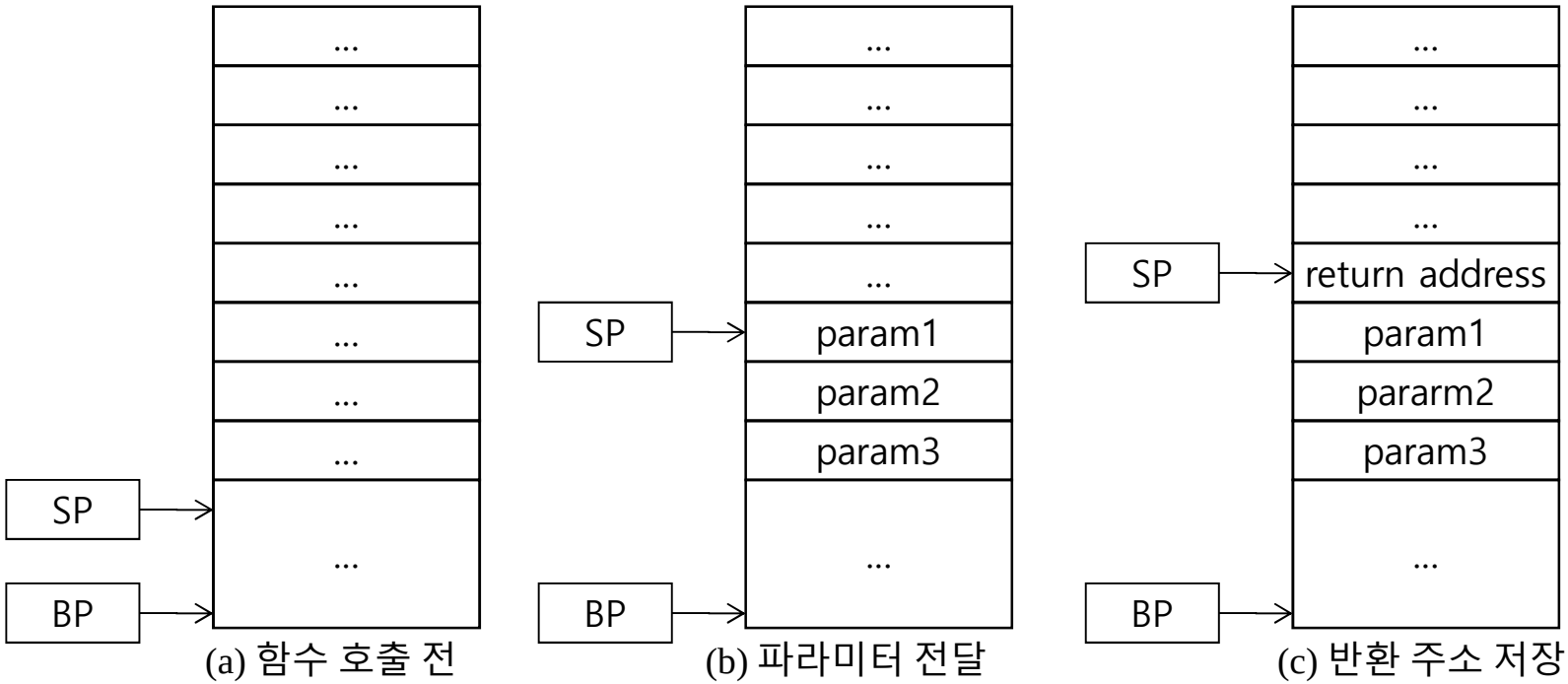
스택 구현

SP 갱신 시기	SP 증가		SP 감소	
PUSH 명령어 실행 시 저장 전에 SP 갱신	PUSH	SP ← SP + [단어의 크기] Mem[SP] ← 오퍼런드	PUSH	SP ← SP - [단어의 크기] Mem[SP] ← 오퍼런드
	POP	오퍼런드 ← Mem[SP] SP ← SP - [단어의 크기]	POP	오퍼런드 ← Mem[SP] SP ← SP + [단어의 크기]
PUSH 명령어 실행 시 저장 후에 SP 갱신	PUSH	Mem[SP] ← 오퍼런드 SP ← SP + [단어의 크기]	PUSH	Mem[SP] ← 오퍼런드 SP ← SP - [단어의 크기]
	POP	SP ← SP - [단어의 크기] 오퍼런드 ← Mem[SP]	POP	SP ← SP + [단어의 크기] 오퍼런드 ← Mem[SP]

베이스 Pointer

- Base Pointer (Frame Pointer): 스택의 기준점
  - □ 사용 예

```
void function(int param1, param2, param3)
{
    int local1, local2;
    …
}
```

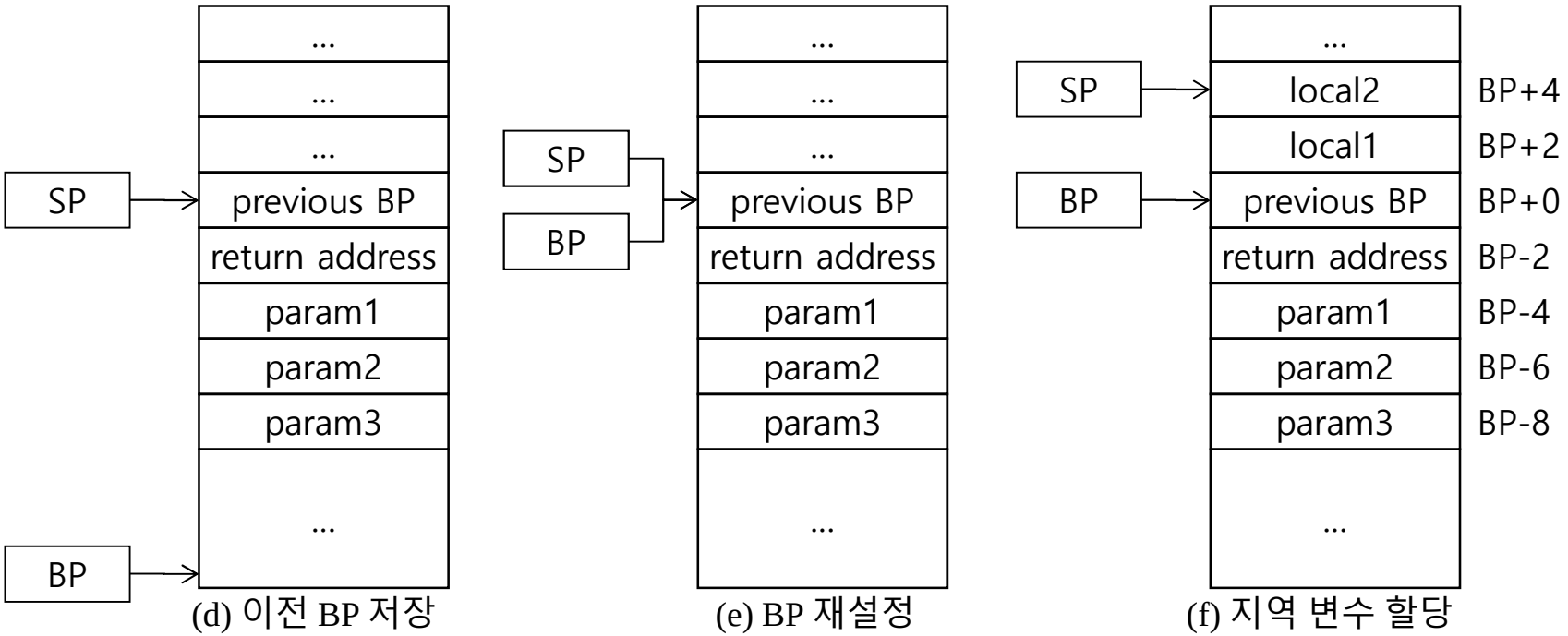


(a) 함수 호출 전 (b) 파라미터 전달 (c) 반환 주소 저장

베이스 Pointer(frame pointer) 사용 예

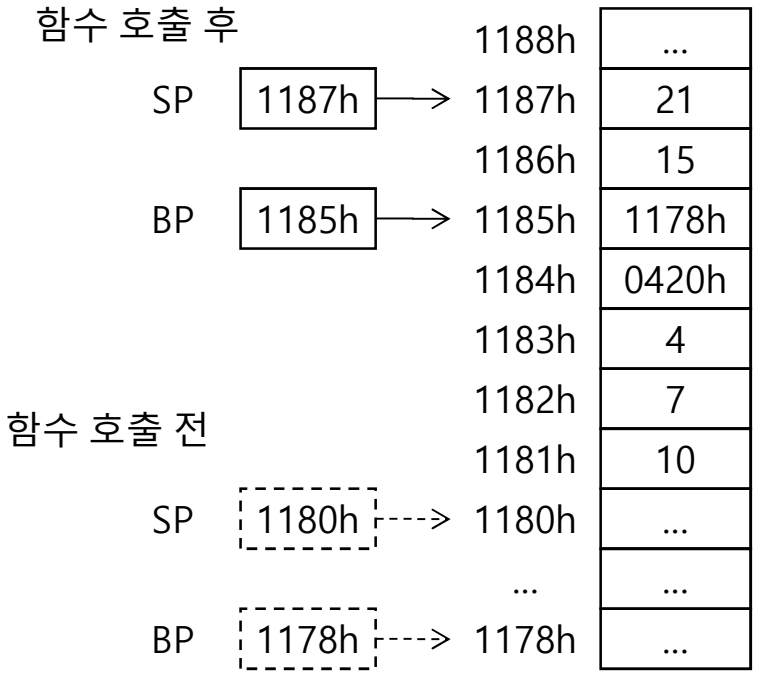
- 사용 예

```
void function(int param1, param2, param3)
{
    int local1, local2;
    ...
}
```



예제 4-4 함수 호출 예

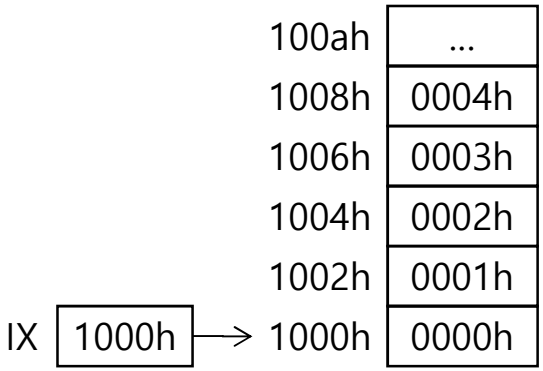
- 단어 크기는 기억장치 장소 한 개의 크기와 같고, 16 비트이다.
- 함수 호출하기 전
  - SP = 1180h, BP = 1178h이다.
- 함수호출
  - function(4, 7, 10)으로 호출
- 리턴할 주소
  - 0420h
- 함수의 지역 변수의 값은
  - local1 = 15, local2 = 21이다.



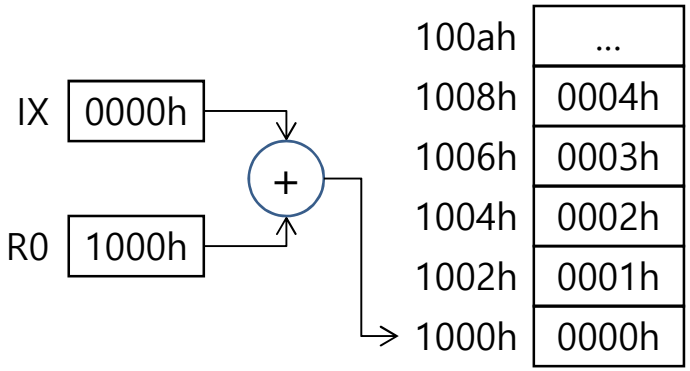
인덱스 레지스터

- 인덱스 레지스터 (Index Register)
  - 연속적으로 저장되어 있는 데이터(배열) 액세스
  - 자동 인덱싱 (automatic indexing)

- 예
  - int array[5] = { 0, 1, 2, 3, 4 }; // 1000h 번지부터 할당

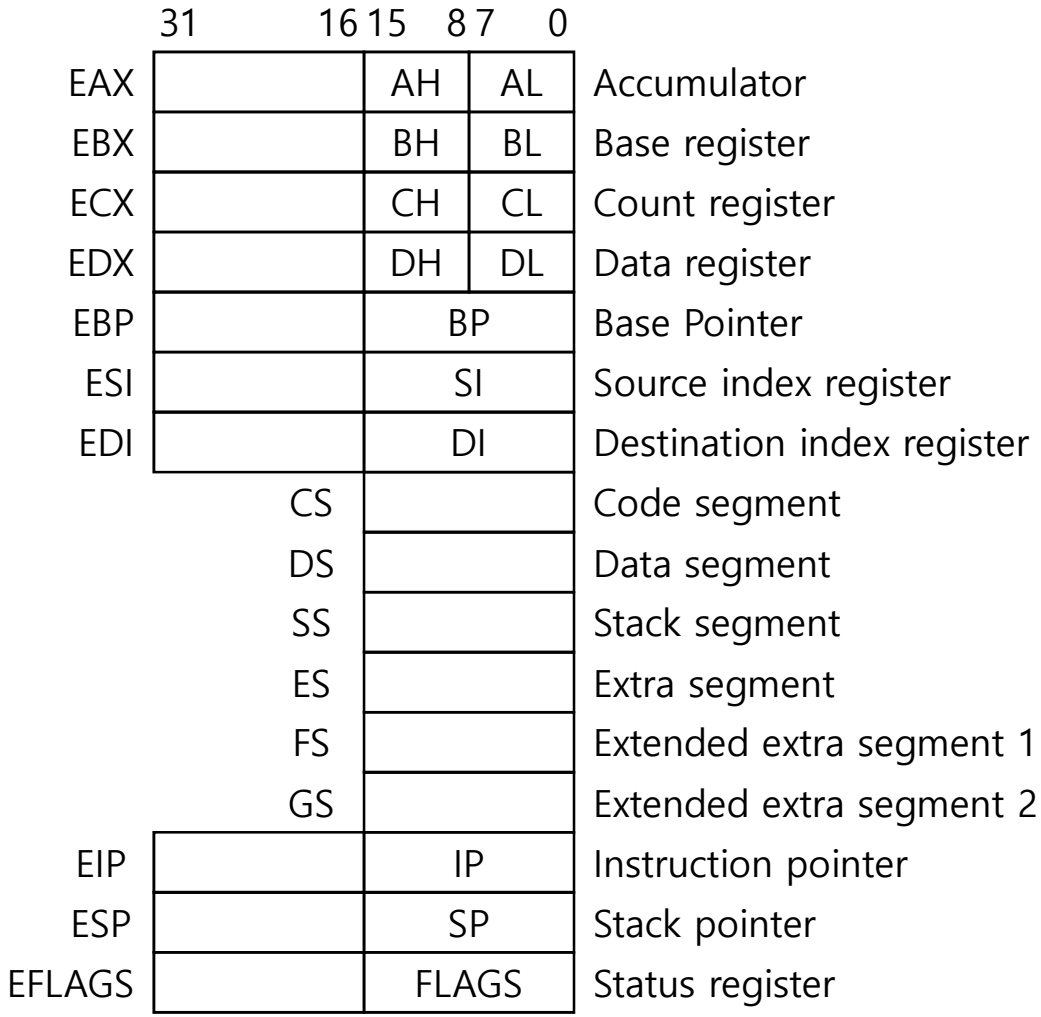


(a) 사용 예 1: R1 ← Mem[IX], IX ← IX + 2

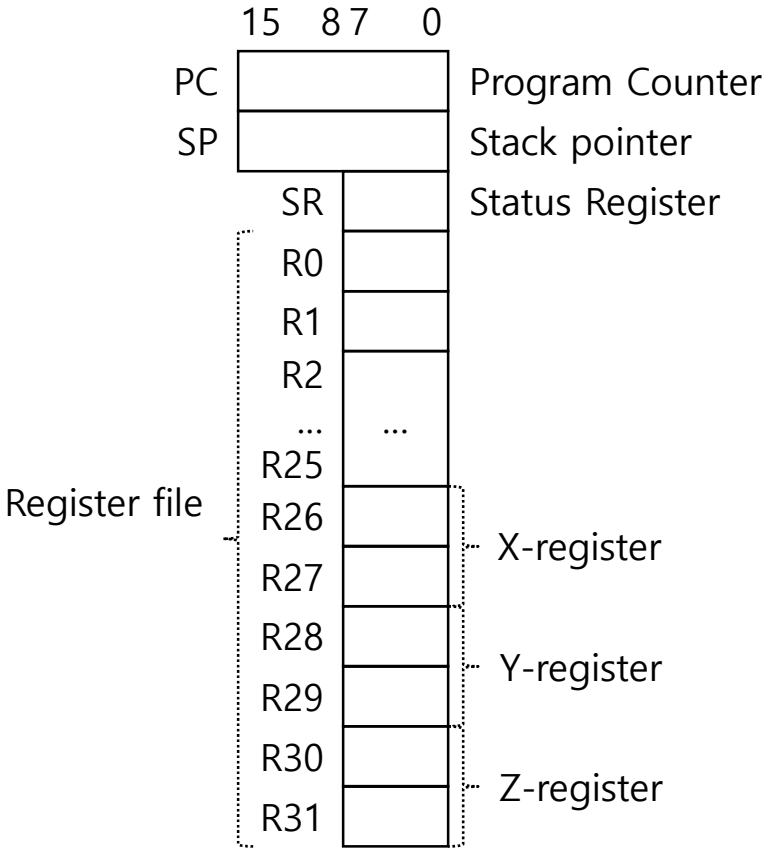


(b) 사용 예 2: R1 ← Mem[R0+IX], IX ← IX + 2

Intel Pentium 레지스터



AVR Atmega128 레지스터

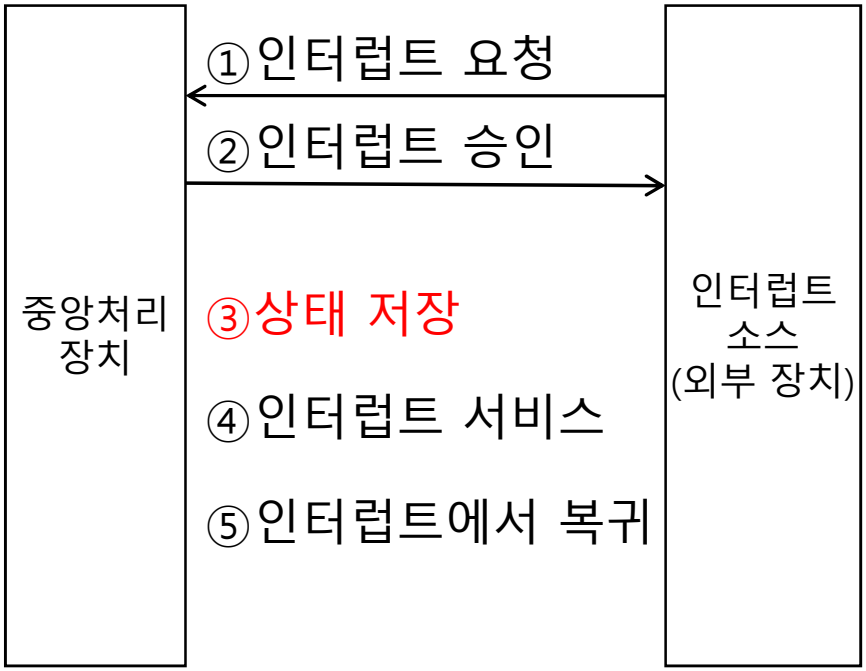


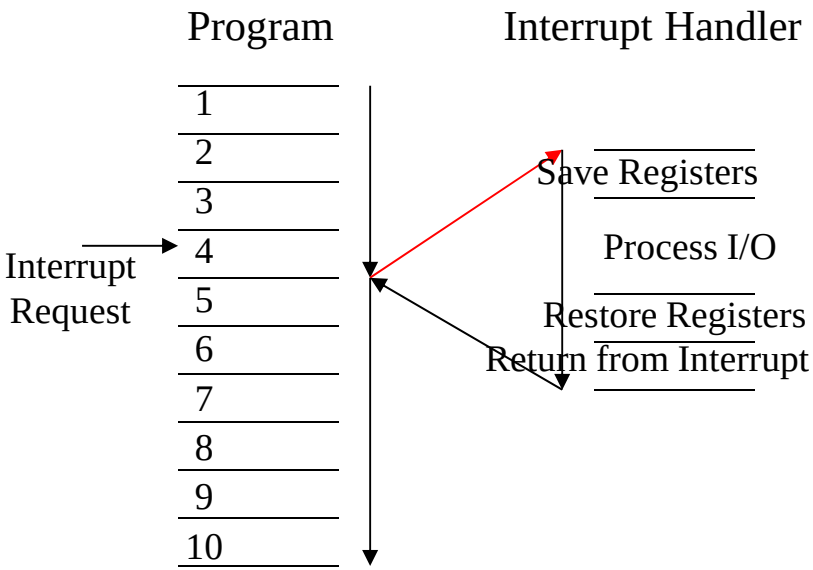
인터럽트

- 인터럽트
  - 컴퓨터 내부 혹은 외부에서 발생할 수 있는 갑작스러운 사건에 대응하는 기능

- 인터럽트 종류
  - 내부 인터럽트
    - 하드웨어 고장
    - 실행할 수 없는 명령어
    - 명령어 실행 오류: 나누기 0
    - 사용 권한 위배
  - 외부 인터럽트
    - 타이머 인터럽트
    - 입출력 인터럽트

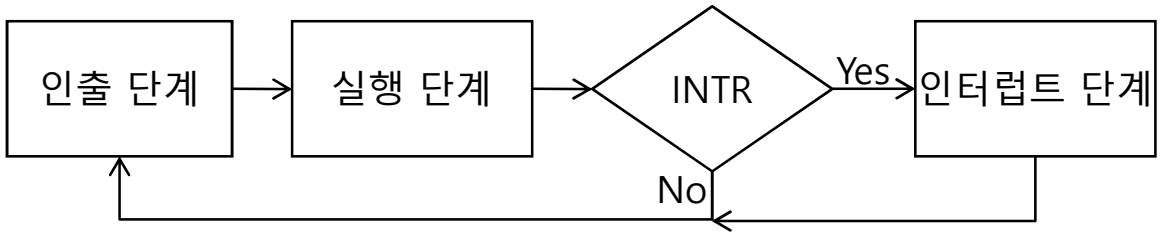
인터럽트 처리 과정


중앙처리
장치
①인터럽트 요청
②인터럽트 승인
③상태 저장
④인터럽트 서비스
⑤인터럽트에서 복귀
인터럽트
소스
(외부 장치)

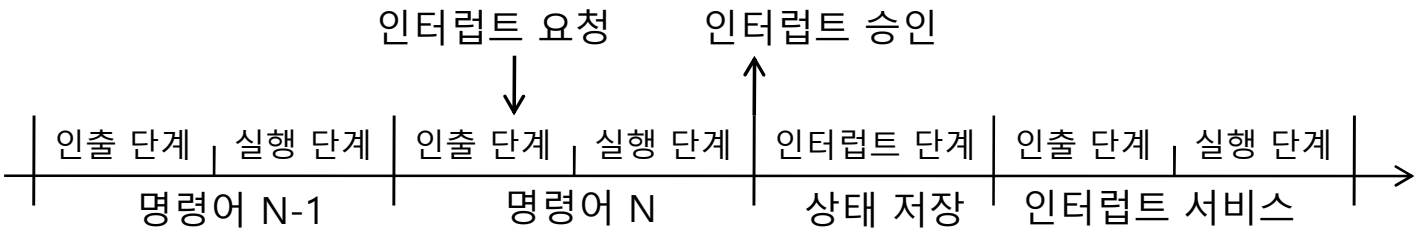

Program
Interrupt Handler
Interrupt
Request
1
2
3
4
5
6
7
8
9
10
Save Registers
Process I/O
Restore Registers
Return from Interrupt

명령어 사이클 수정

- 명령어 사이클



- 인터럽트 요청과 승인



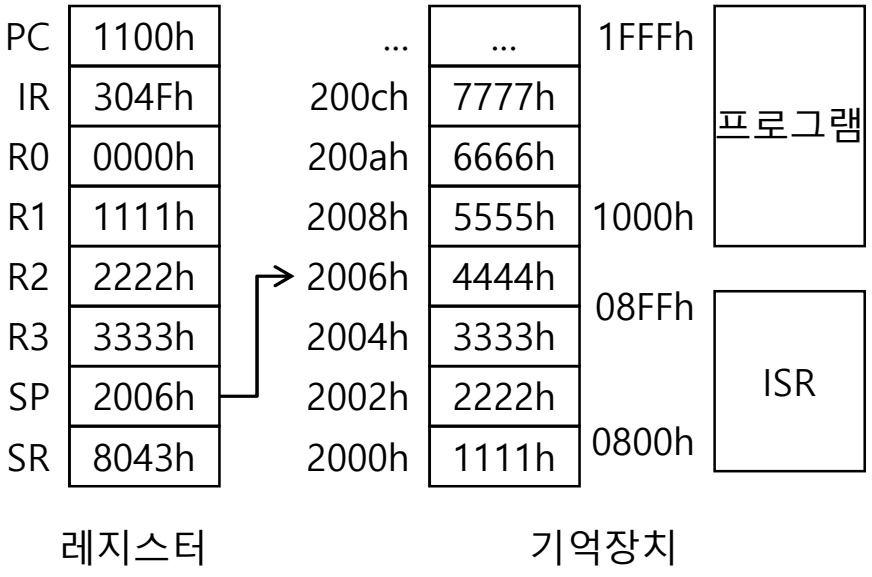
인터럽트 단계

- 인터럽트 단계
  - Context(PC와 SR)을 스택에 저장
  - Interrupt source를 확인하고 ISR의 시작 주소 확인
  - PC에 ISR 시작 주소 적재

- 인터럽트 서비스 루틴 (ISR)
  - 일반 프로그램

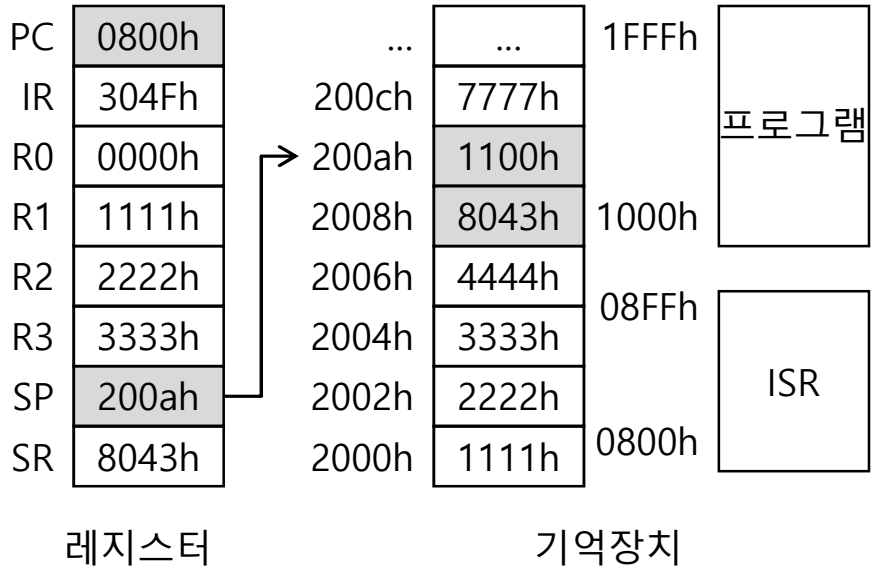
- 인터럽트 리턴 명령어 (Return from Interrupt)
  - Context 복구

예제 4-5 인터럽트 단계 전/후 상태



(a) 인터럽트 단계 실행 전

PUSH SR
PUSH PC
PC ← ISR



(b) 인터럽트 단계 실행 후

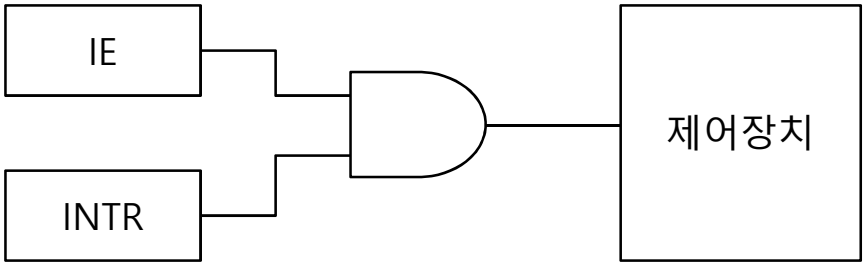
인터럽트 가능 플래그

- IE flag
  - 상태 레지스터 안의 한 개의 flag
  - 중앙처리장치가 인터럽트를 허용할지 제어

- 컴퓨터의 상태의 변화
  - 시스템 리셋 후: 인터럽트 불가능 상태
  - 초기화 과정 (부트로더)
    - 시스템 스택 설정
    - 인터럽트 서비스 루틴 적재
    - 인터럽트 벡터 설정
  - 인터럽트 허용 상태로 전환

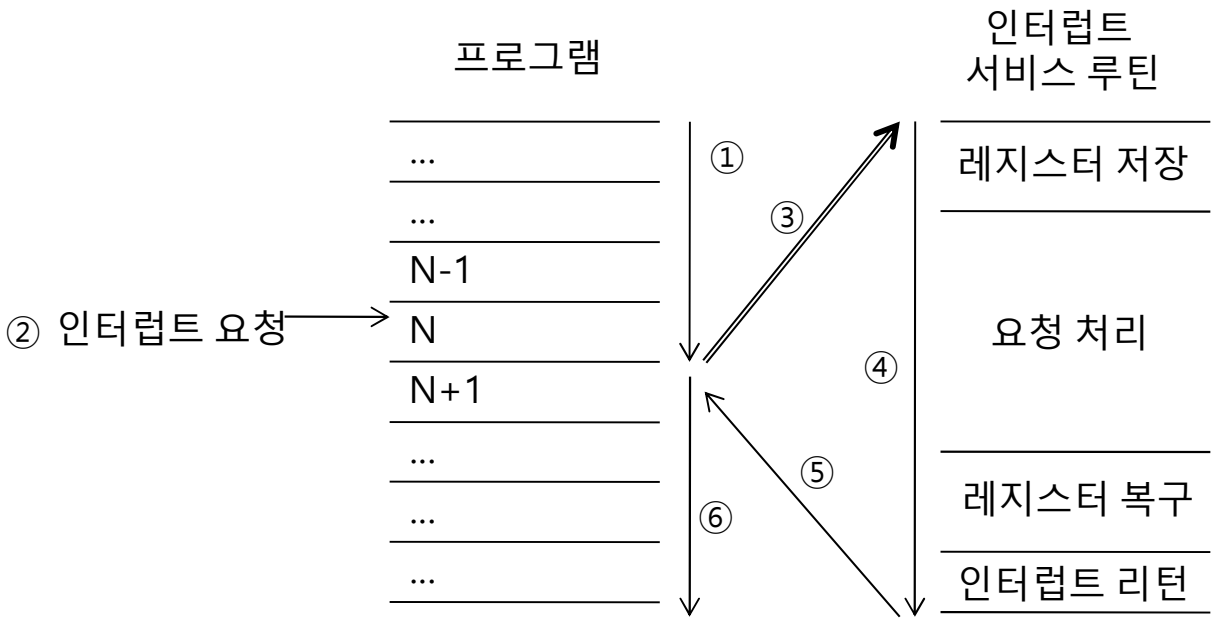
인터럽트 제어

- IE 제어 명령어
  - STI (Set Interrupt Flag)
  - CLI (Clear Interrupt Flag)

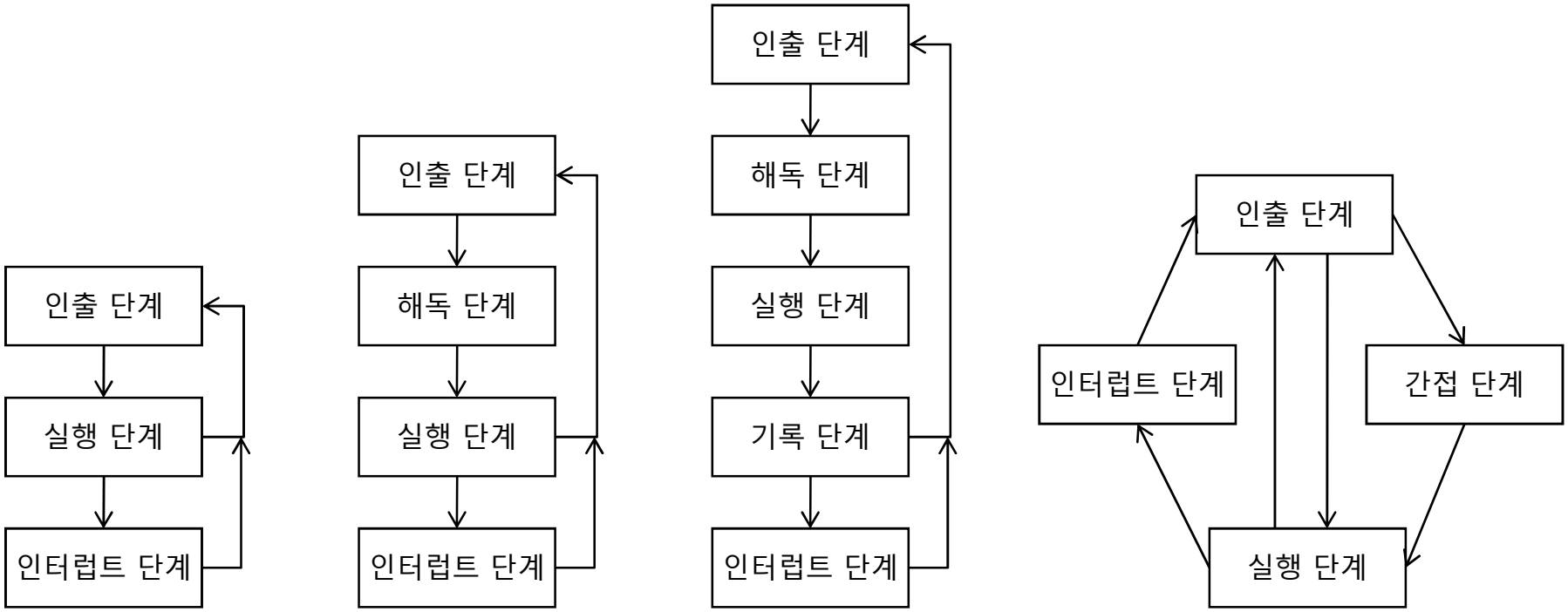


- 임계 영역 (critical section)
  - 프로그램 중 실행이 중단되었다가 다시 시작되면 심각한 문제가 발생하는 영역
  - IE 제어 명령어로 보호한다.

인터럽트 서비스 루틴


프로그램
인터럽트
서비스 루틴
...
...
N-1
N
N+1
...
...
...
①
② 인터럽트 요청
③
④
⑤
⑥
레지스터 저장
요청 처리
레지스터 복구
인터럽트 리턴

명령어 사이클



(a) 3-단계 명령어 사이클 (b) 4-단계 명령어 사이클 (c) 5-단계 명령어 사이클 (d) 4-단계 명령어 사이클