[JO2122] 컴퓨터구조

2022년 1학기

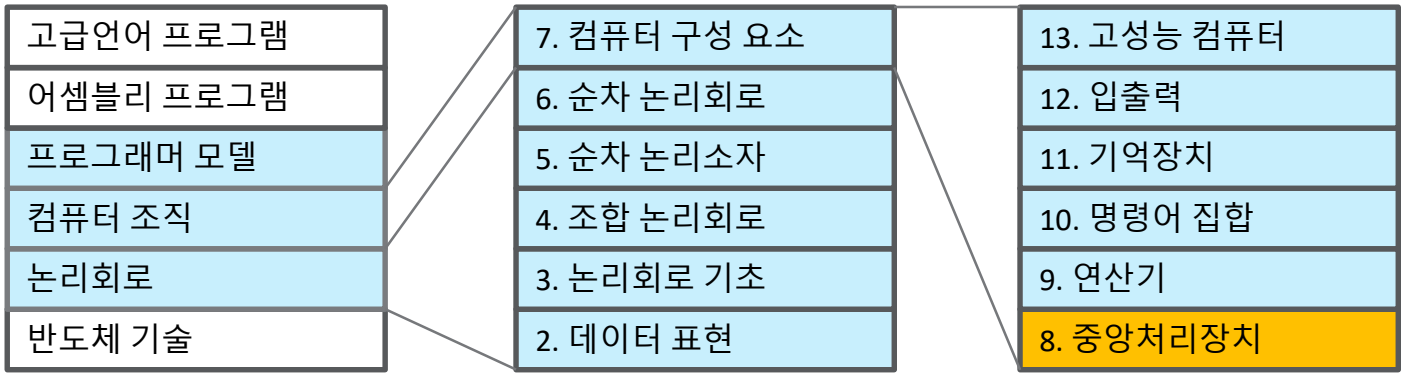
상명대학교 소프트웨어학과 박희민

8.1 중앙처리장치 구조
8.2 레지스터
8.3 인터럽트
8.4 제어장치
8.5 요약

2022-05-04

CHAP08 중앙처리장치

제8장 중앙처리장치

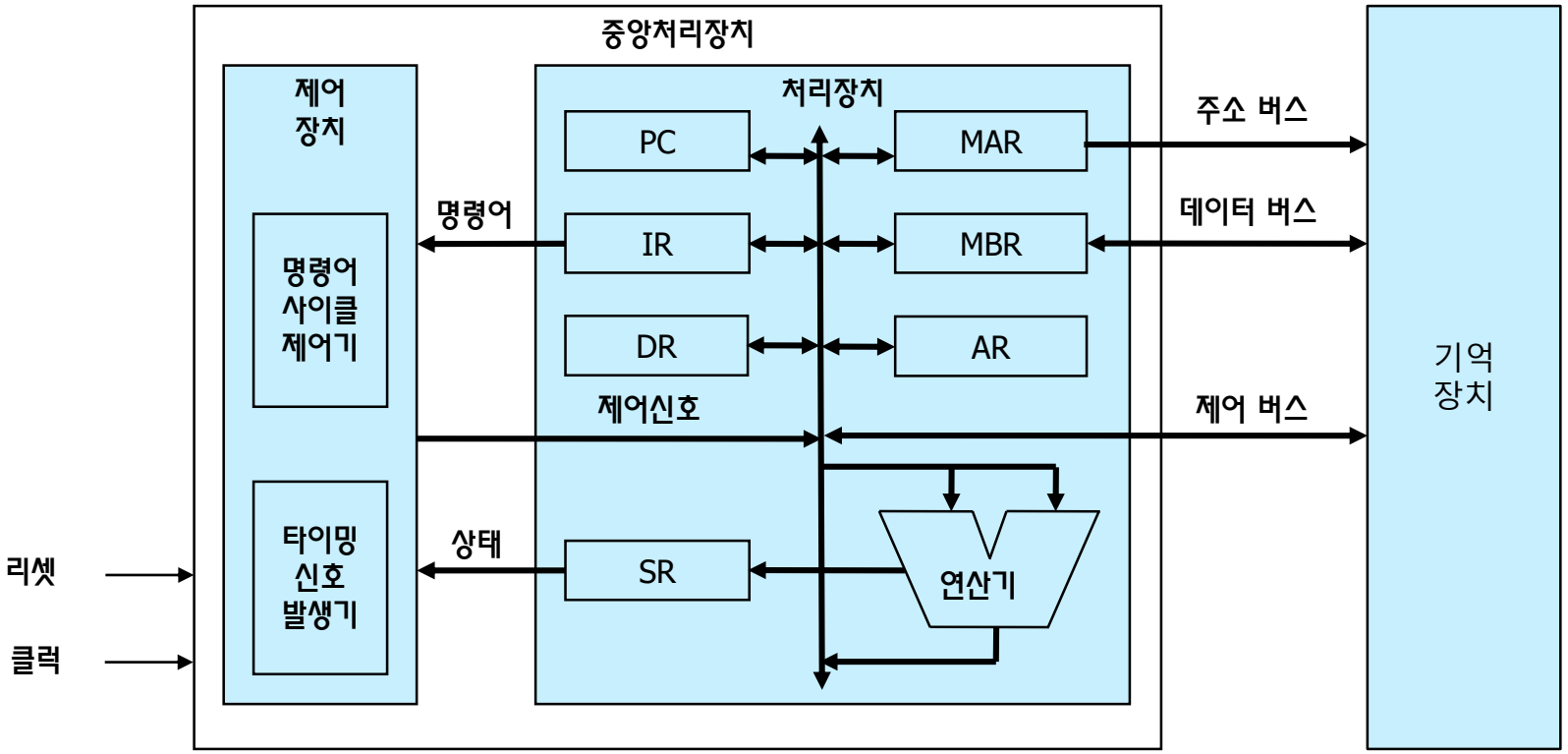


- 학습 목표
  - 레지스터 종류를 제시하고 활용 방법을 설명할 수 있다.
  - 인터럽트 개념을 이해하고 처리 과정을 설명할 수 있다.
  - 제어장치의 구조를 제시하고 기능을 설명할 수 있다.
- 내용
  - 8.1 중앙처리장치 구조
  - 8.2 레지스터
  - 8.3 인터럽트
  - 8.4 제어장치
  - 8.5 요약

8.1 중앙처리장치 구조

- 컴퓨터 기능 요약
  - 프로그램 실행: 프로그램 = sequence of instructions
  - 명령어 사이클: {인출 단계, 실행 단계}
  - 인출 단계: IR ← Mem(PC), PC ← PC + 명령어(단어) 크기
  - 실행 단계: 제어장치에서 명령어 해석하고, 처리장치에서 실행
- 학습 목표
  - 중앙처리장치의 구조를 제시하고,
  - 기능을 제어장치와 처리장치로 나누어 설명할 수 있다.
- 구성
  - 8.1.1 제어장치
  - 8.1.2 처리장치

중앙처리장치 구조



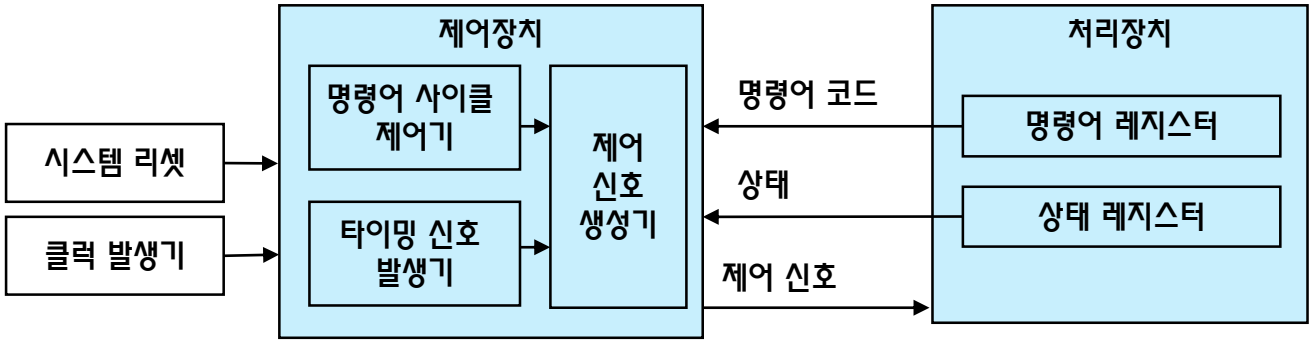
PC: Program Counter
IR: Instruction Register

MAR: Memory Address Register
MBR: Memory Buffer Register

AR: Address Register
DR: Data Register
SR: Status Register

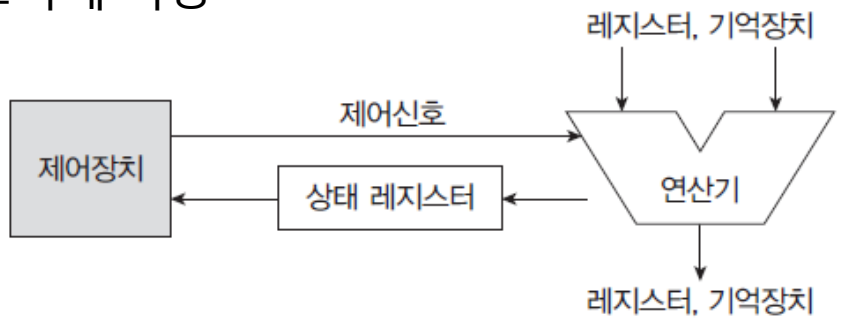
8.1.1 제어장치

- 제어장치 기능
  - 순서 제어(sequence control): 명령어 사이클 운영
    - 명령어를 차례대로 실행한다.
  - 동작 제어(operation control): 제어신호 생성
    - 각 명령어를 실행한다.
- 제어장치 구조
  - 명령어 사이클 제어기: 상태도 운영, 명령어 사이클 구분
  - 타이밍 신호 발생기: 클럭 구분
  - 제어신호 생성기: (명령어 코드, 상태)를 받아 제어 신호 생성



8.1.2 처리장치

- 레지스터 (8.2절)
  - 중앙처리장치 내부 기억장치: 주소와 데이터 임시 저장
  - 제어용 레지스터: PC, SR, MAR, MBR
  - 명령어 실행용 레지스터
    - 데이터 레지스터 (DR), 주소 레지스터 (AR), 범용 레지스터 (GPR)
- 연산기 (9장)
  - 데이터 처리 담당
  - 제어신호는 연산기의 동작 결정
  - 연산 후 상태를 상태 레지스터에 저장



[그림 8-2] 연산기 주변 회로

8.1 중앙처리장치 구조 요약

- 제어장치
  - 순서제어
    - 명령어 사이클을 운영하여 프로그램 진행
  - 동작제어
    - 제어신호를 생성하여 명령어 실행

- 처리장치
  - 레지스터
    - 제어용 레지스터: 프로세서가 자체적으로 프로그램 진행을 제어하기 위해 사용
    - 명령어 실행용 레지스터: 프로그램 실행에 필요한 주소와 데이터를 임시 저장
  - 연산기
    - 데이터 처리 담당

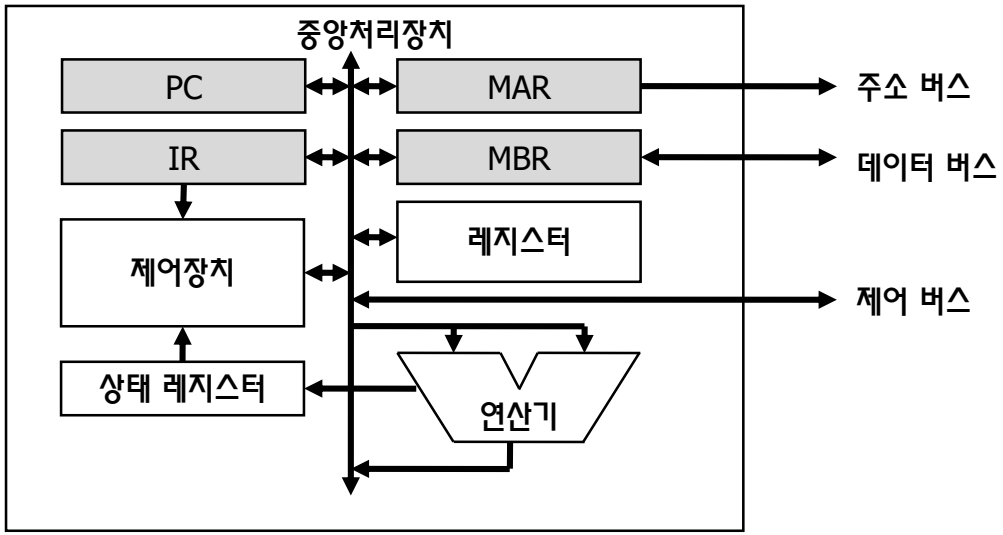
8.2 레지스터

- 학습 목표
  - 레지스터 종류와 기능을 설명할 수 있다.
  - 상태 레지스터 플래그의 기능을 설명할 수 있다.
  - 시스템 스택 구조를 제시하고 스택 포인터를 활용 방법을 설명할 수 있다.

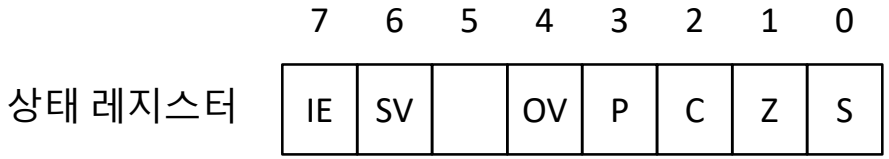
- 내용
  - 8.2.1 제어용 레지스터
  - 8.2.2 상태 레지스터
  - 8.2.3 명령어 실행용 레지스터
  - 8.2.4 주소 레지스터
  - 8.2.5 스택 포인터

8.2.1 제어용 레지스터

- 기억장치 주소 레지스터(MAR, Memory Address Register)
  - 기억장치 주소 저장
  - 주소 버스에 연결
- 기억장치 버퍼 레지스터(MBR, Memory Buffer Register)
  - 기억장치 데이터 중계
  - 데이터 버스에 연결
- 데이터 읽기
  1. MAR ← 주소 레지스터
  2. MBR ← Mem[MAR]
  3. 데이터 레지스터 ← MBR
- 데이터 쓰기
  1. MAR ← 주소 레지스터
  2. MBR ← 데이터 레지스터
  3. Mem[MAR] ← MBR



8.2.2 상태 레지스터



- 상태 레지스터
  - 플래그(flag)의 모임
- 플래그 종류
  - 조건 플래그(conditional flags)
    - 연산 결과 반영
  - 제어 플래그(control flags)
    - 중앙처리장치 동작 제어

- 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)

조건 플래그

- [예제 8-1] 덧셈 후 조건 플래그의 값을 구하라.

$$\begin{array}{r} 0110\_0101 \\ +0101\_0100 \\ \hline 1011\_1001 \end{array}$$

1) S = _____: sign flag
2) Z = _____: zero flag
3) C = _____: carry flag
4) P = _____: even parity
5) OV = _____: overflow flag

8.2.3 명령어 실행용 레지스터

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

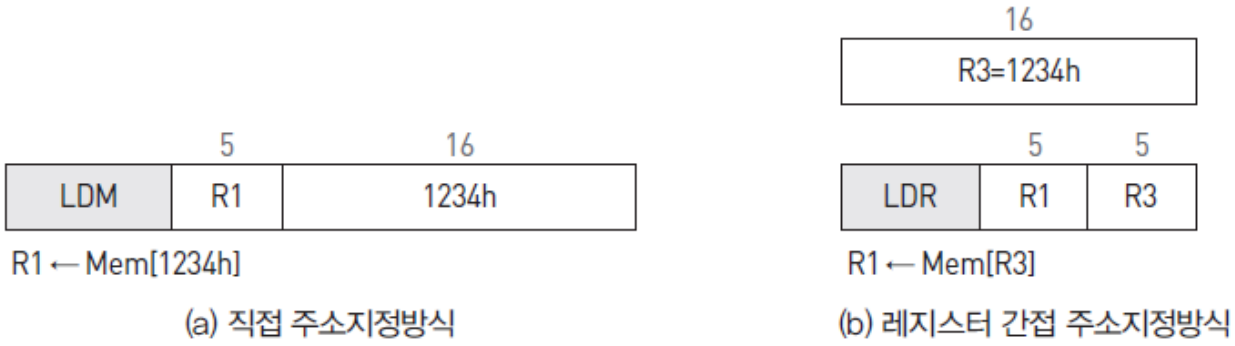
- 주소 레지스터(Address Register)
  - 기억장치 주소 저장
  - 다양한 주소지정방식

- 범용 레지스터(General Purpose Register)
  - 데이터 레지스터 + 주소 레지스터
  - 통상 8개 ~ 32개

8.2.4 주소 레지스터

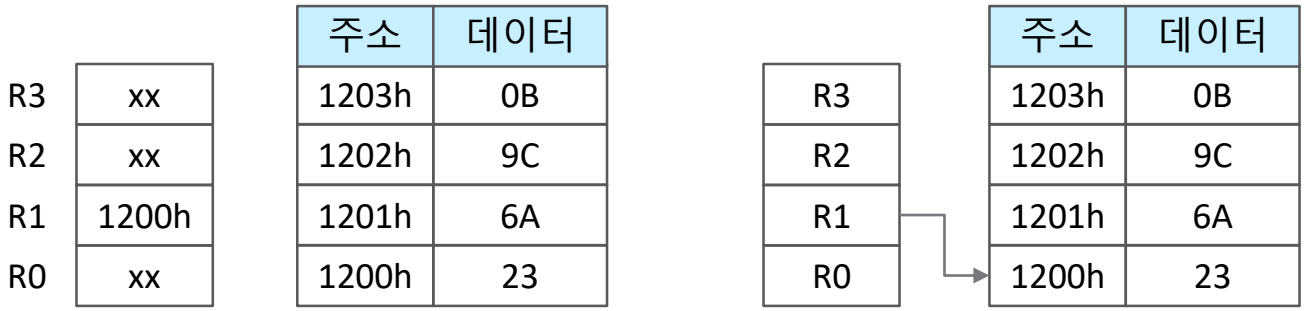
- 주소 레지스터
  - 기억장치를 액세스할 주소를 저장하는 레지스터
  - 포인터(pointer): 주소 레지스터와 같은 의미
  - 레지스터에 기억장치 주소를 저장하면, 명령어 길이가 짧아진다.
- [예제 8-2] 기억장치 용량이 64K바이트이고, 범용 레지스터가 32개이다.

1) 명령어에 기억장치 주소를 표현하기 위한 비트 수는?
2) 명령어에 레지스터 중 하나를 표현하기 위한 비트 수는?



[그림 8-4] 기억장치 액세스 방법

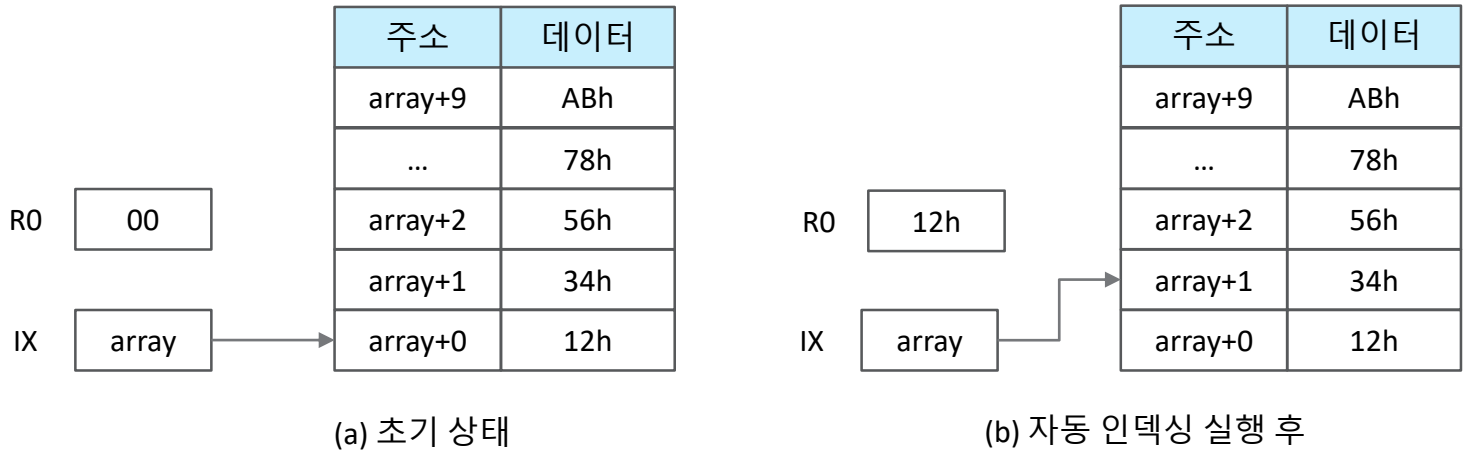
포인터 개념



[그림 8-5] 포인터 개념

- [그림 8-5(a)] R1에 1200h이 저장되어 있다.
- [그림 8-5(b)] R1이 1200h를 가리킨다. (R1의 값이 1200h임을 암시적으로 표현한다.)

인덱스 레지스터



[그림 8-6 인덱스 레지스터

- 배열 액세스
- 자동 인덱싱: R0 ← R0 + Mem(IX), IX ← IX + 1

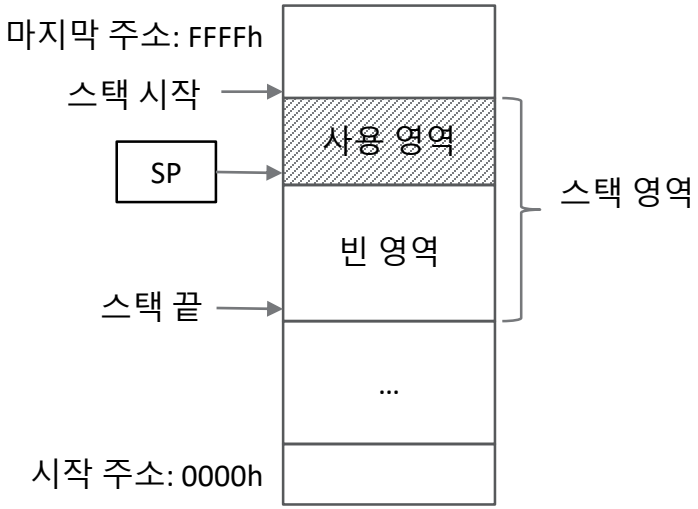
8.2.5 스택 포인터

- 스택(stack)
  - Last-In-First-Out: 데이터를 쌓아 두었다가 하나씩 꺼내서 사용
  - 한 곳에서 액세스: stack top ← stack pointer

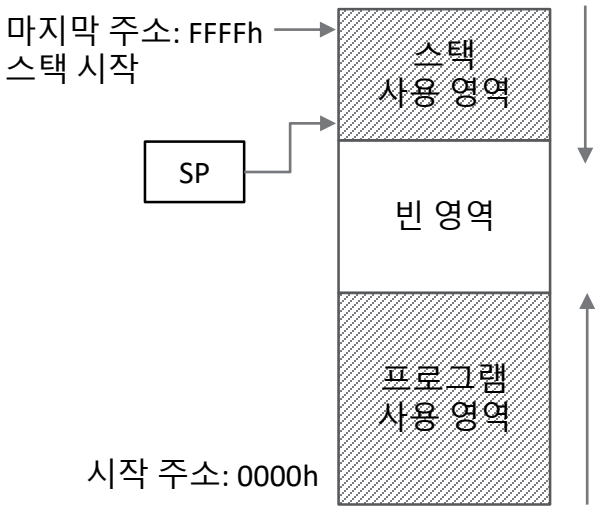
- 시스템 스택(system stack)
  - 주기억장치의 일부를 스택 영역으로 활용
  - 시스템 스택의 활용
    - 함수 호출시 리턴 주소 저장, 지역 변수 저장, 파라미터 전달

- SP (Stack Pointer) 레지스터
  - Point to stack top

스택 구조



(a) 기억장치 중간 부분을 스택으로 사용



(b) 기억장치 마지막 부분을 스택으로 사용

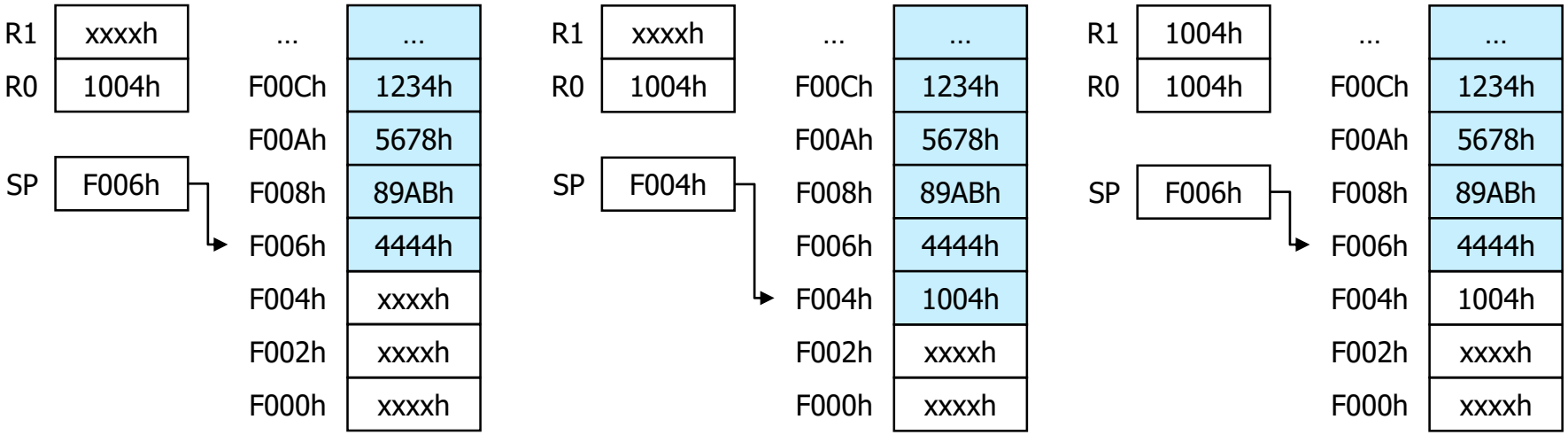
- 스택 동작
  - PUSH: 스택에 데이터 추가. SP 값 감소
  - POP: 스택에서 데이터 제거. SP 값 증가

스택 동작

PUSH 오퍼랜드:
SP ← SP - [단어 크기]
Mem[SP] ← **오퍼랜드**

POP 오퍼랜드:
오퍼랜드 ← Mem[SP]
SP ← SP + [단어 크기]

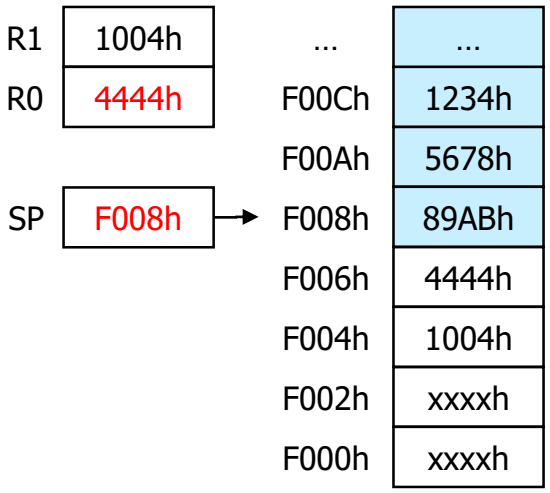
- 스택의 데이터 크기 = 레지스터 크기 = 단어 크기
- 예: 기억장치 바이트 단위로 구성, 데이터 16 비트



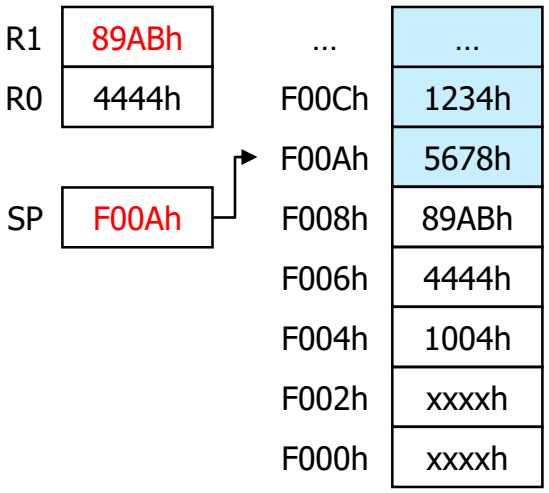
(a) **초기 상태** (b) PUSH R0 **실행 후** (c) POP R1 **실행 후**

[예제 8-3] 스택 상태 변화

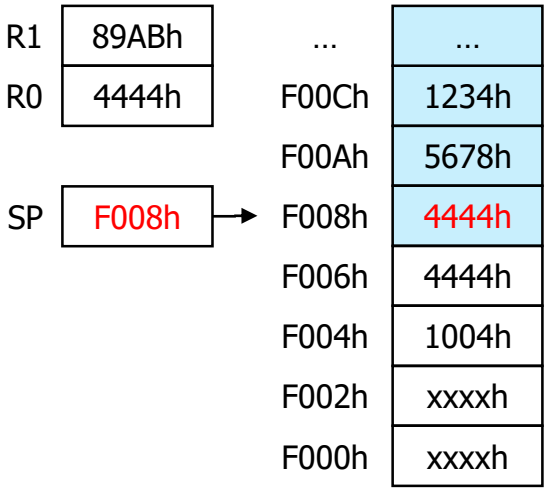
- [그림 8-8(c)] 상태에서 다음 명령어 연속 실행할 때 스택의 변화
  - POP R0
  - POP R1
  - PUSH R0



(a) POP R0 **실행 후**

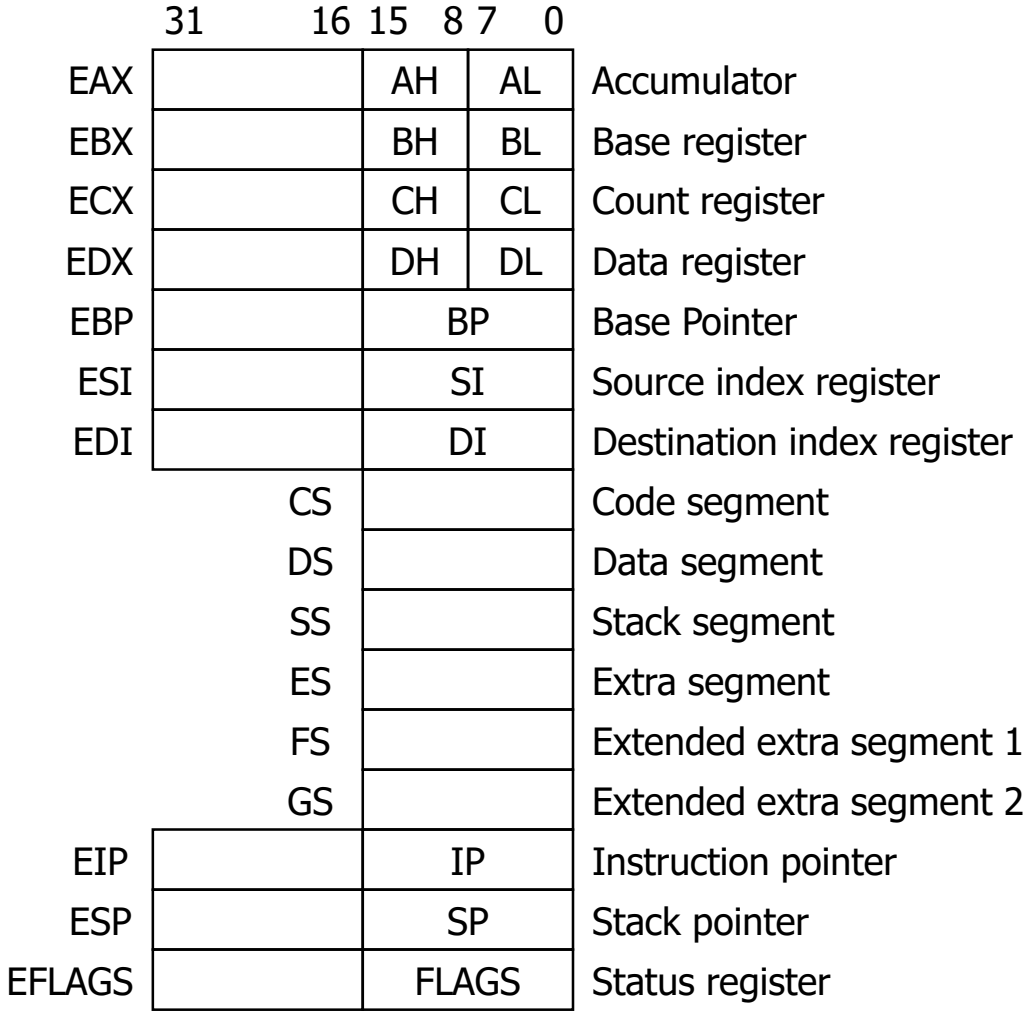


(b) POP R1 **실행 후**

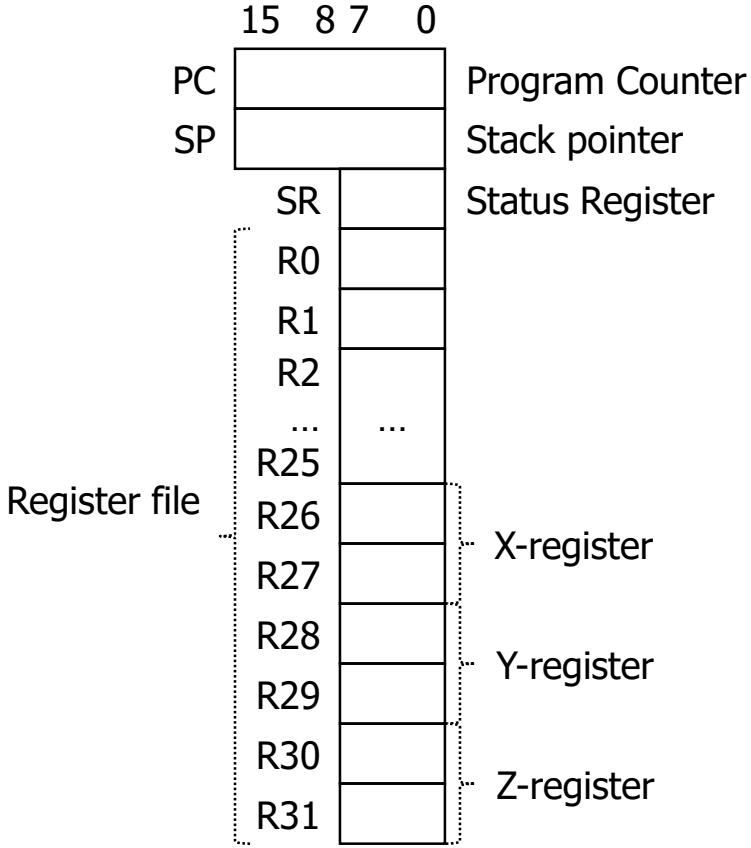


(c) PUSH R0 **실행 후**

프로세서 레지스터



(a) Intel Pentium 레지스터



(b) AVR Atmega128 레지스터

8.2 레지스터 요약

- 제어용 레지스터
  - PC, IR: 명령어 인출
  - MAR, MBR: 주기억장치 인터페이스
  - SR(상태 레지스터): 조건 플래그와 제어 플래그의 모임
- 명령어 실행용 레지스터
  - 데이터 레지스터
  - 주소 레지스터(포인터)
    - 인덱스 레지스터: 배열 액세스
    - 스택 포인터(SP): 시스템 스택 운영. PUSH, POP
  - 범용 레지스터

8.3 인터럽트

- 인터럽트
  - 컴퓨터 내부 또는 외부에서 발생하는 갑작스러운 사건에 대응하는 기능
- 학습 목표
  - 인터럽트 처리를 위하여 중앙처리장치가 제공하는 기능을 이해하고
  - 인터럽트 요청부터 서비스까지 처리 과정을 설명할 수 있다.

- 내용

8.3.1 인터럽트 개념

8.3.2 인터럽트 처리 기능

8.3.3 인터럽트 서비스 루틴

8.3.1 인터럽트 개념

- 인터럽트(interrupt)
  - 방해하다.
  - 처리 과정
    - 인터럽트 요청
    - 상태 저장
    - 인터럽트 서비스
    - 복귀

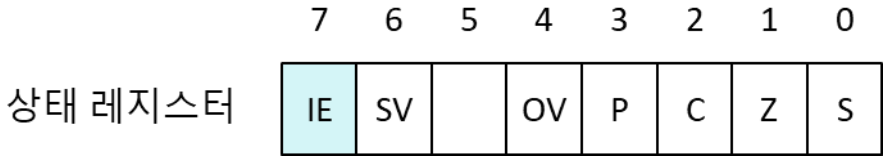
- 인터럽트 종류
  - 내부 인터럽트
    - 하드웨어 고장
    - 실행할 수 없는 명령어
    - 명령어 실행 오류: 나누기 0
    - 사용 권한 위배
  - 외부 인터럽트
    - 타이머 인터럽트
    - 입출력 인터럽트

8.3.2 인터럽트 처리 기능

- 인터럽트 처리를 위해 필요한 사항
  - 인터럽트 가능 플래그 (Interrupt Enable flag)
  - 인터럽트 요청과 확인
  - 인터럽트 단계
  - 인터럽트 서비스 루틴 – 8.3.3절

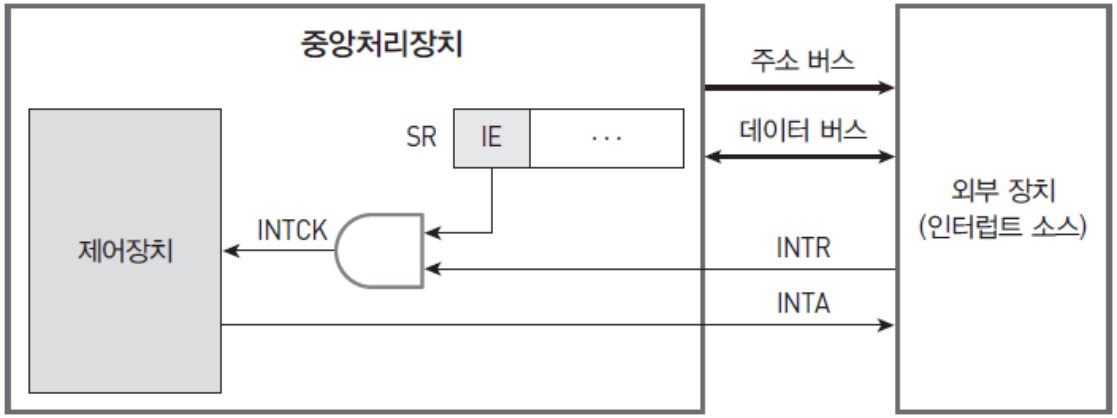
인터럽트 가능 플래그

- 인터럽트 가능 플래그 (IE flag, Interrupt Enable flag)
  - 상태 레지스터에 있는 제어 플래그 중 하나
  - 중앙처리장치가 인터럽트를 허용할지 제어



- IE flag 제어 명령어
  - STI (Set Interrupt Flag)
    - IE ← 1
    - 인터럽트 요청을 인식함
  - CLI (Clear Interrupt Flag)
    - IE ← 0
    - 인터럽트 요청을 인식하지 않음

인터럽트 요청과 확인

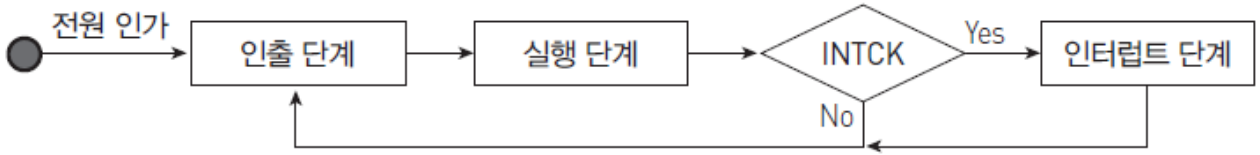


[그림 8-10] 인터럽트 요청과 확인

- INTR (Int. Request)
  - 외부 장치가 중앙처리장치로 인터럽트 요청
- INTA (Int. Acknowledge)
  - 중앙처리장치가 외부 장치에게 인터럽트 요청을 받았음을 확인
- INTCK = IE·INTR
  - 제어장치는 INTCK를 확인.
  - IE=0일 때, INTR 무시

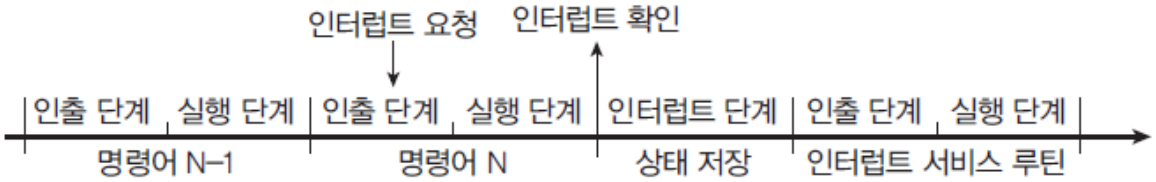
인터럽트 단계

- 명령어 사이클



[그림 8-11] 인터럽트를 고려한 명령어 사이클

  - 각 단계를 상태로 취급할 수 있다.
  - 동기 순차 논리회로로 구현된다.
- 인터럽트 요청과 승인



[그림 8-12] 인터럽트 요청과 확인

  - 인터럽트 요청은 아무 때나 (중앙처리장치 동작과 비동기적으로 발생)
  - 인터럽트 확인은 실행 단계가 끝날 때

인터럽트 단계의 동작

- 인터럽트 단계

  SP ← SP – [단어 크기] // PUSH PC, 콘텍스트 저장

  Mem[SP] ← PC

  SP ← SP – [단어 크기] // PUSH SR

  Mem[SP] ← SR

  PC ← ISR 시작 주소 // PC 갱신

- 인터럽트 서비스 루틴 (ISR)
  - 인터럽트 단계 후 실행하는 프로그램
  - 마지막에 인터럽트 리턴 명령어(IRET) 실행
- 인터럽트 리턴 명령어 (Return from Interrupt)

  SR ← Mem[SP] // POP SR, 콘텍스트 복구

  SP ← SP + [단어 크기]

  PC ← Mem[SP] // POP PC

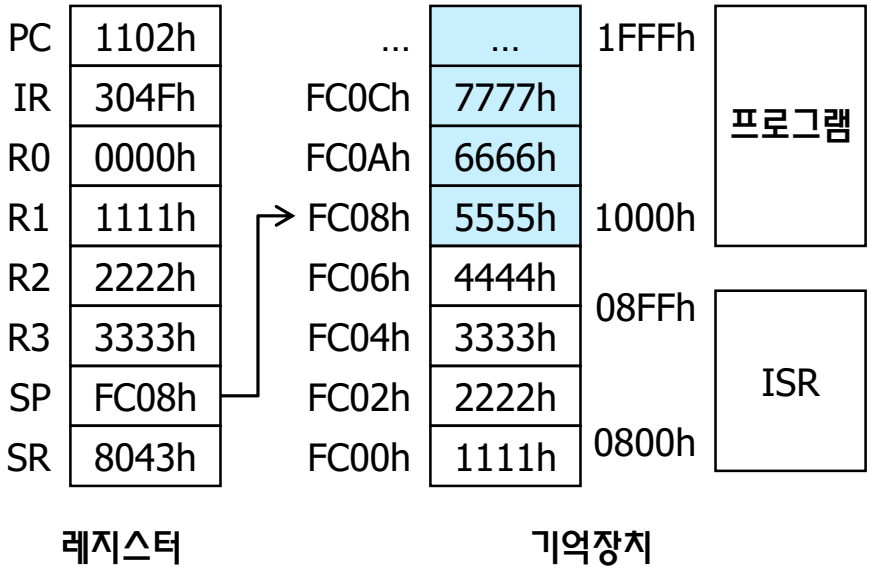
  SP ← SP + [단어 크기]

[예제 8-4] 인터럽트 요청 전후

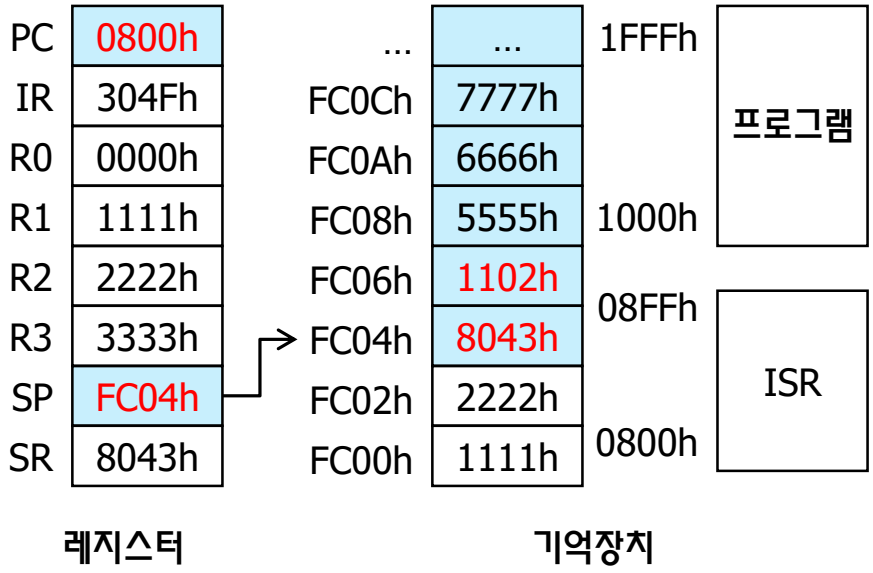
- 프로세서 상태
  - 다음에 실행할 명령어 주소
    PC=1102h
  - 현재 실행 중인 명령어 코드 IR=304Fh
  - 스택 탑: SP=FC08h
  - ISR 시작 주소 = 0800h

- 인터럽트 단계 실행 후

PUSH PC // SP ← FC06, Mem[SP] ← 1102h
PUSH SR // SP ← FC04 Mem[SP] ← 8043h
PC ← ISR 주소 // PC ← 0800



(a) 인터럽트 단계 실행 전



(b) 인터럽트 단계 실행 후

8.3.3 인터럽트 서비스 루틴

- 인터럽트 서비스 루틴(ISR, Interrupt Service Routine)
  - 인터럽트를 처리하는 프로그램
  - Interrupt Handler
- 인터럽트 처리 순서

① 프로그램 실행 중

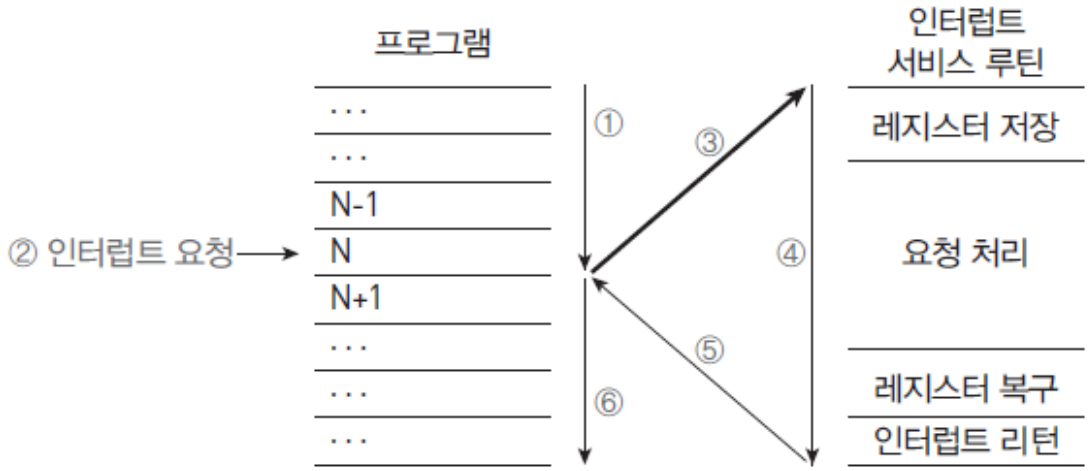
② 인터럽트 요청
- 프로그램 실행과 비동기

③ 인터럽트 단계
- 인터럽트 확인
- 콘텍스트 저장
- PC ← ISR 주소

④ 인터럽트 서비스 루틴 실행

⑤ 인터럽트 리턴
- 콘텍스트 복구

⑥ 중단된 프로그램 다시 시작



[그림 8-14] 인터럽트 처리 과정

8.3 인터럽트 요약

- 인터럽트
  - 중앙처리장치가 갑작스러운 사건에 대처하는 기능
- 인터럽트 가능 플래그(IE flag)
  - 생태 레지스터의 제어 플래그 중 하나
  - 인터럽트 허용 여부 결정
- 인터럽트 단계
  - 명령어 사이클의 한 단계
  - PC와 SR 저장, PC ← ISR 시작 주소
- 인터럽트 서비스 루틴(ISR)
  - 인터럽트를 처리하는 프로그램
  - 실행 후 원래 실행 중이던 프로그램으로 복귀

8.4 제어장치

- 학습 목표
  - 제어장치의 구조를 제시하고, 제어장치가 명령어 코드를 받아 제어 신호를 생성하는 과정을 설명할 수 있다.
  - 하드와이어드 제어 방식과 마이크로프로그램 제어 방식의 특징을 구별할 수 있다.
- 내용

8.4.1 제어장치 기능

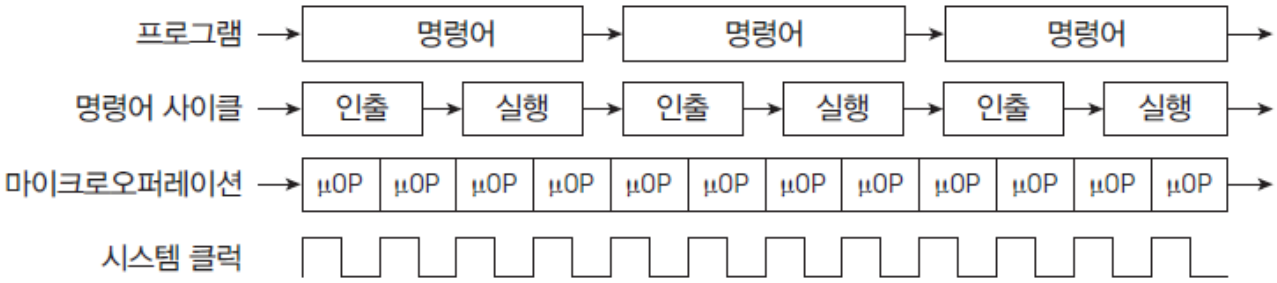
8.4.2 제어장치 구조

8.4.3 명령어 사이클

8.4.1 제어장치 기능

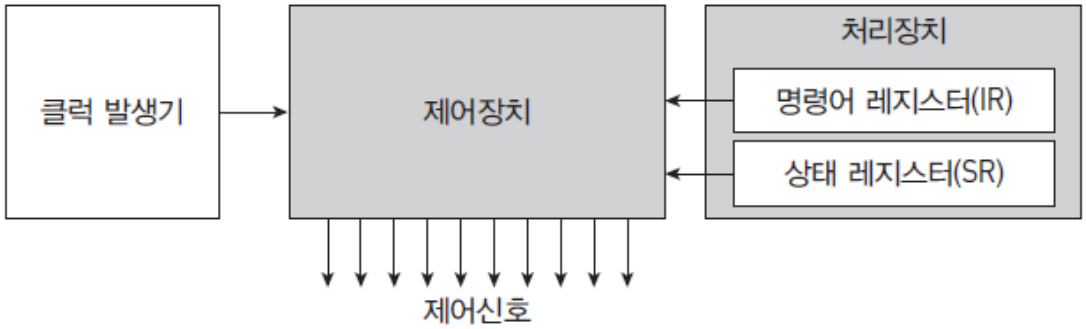
- 제어장치의 기능
  - 순서 제어
    - 명령어 사이클을 진행시킨다.
  - 동작 제어
    - 각 단계에서 필요로 하는 제어 신호를 생성한다.

- 마이크로오퍼레이션
  - 한 개의 클럭 구간에서 실행하는 레지스터 동작



[그림 8-15] 프로그램 실행 과정

제어장치 입력과 출력



[그림 8-16] 제어장치의 입력과 출력

- 입력
  - 명령어 레지스터(IR): 동작 코드(opcode) 해독
  - 상태 레지스터(SR): 조건 실행 여부 결정
  - 클럭 발생기: 동기 순차논리회로
- 출력
  - 제어신호
    - 레지스터 적재와 버스 출력 제어
    - 연산기가 수행할 동작 결정
    - 제어 버스: 기억장치 읽기/쓰기, 인터럽트 확인, DMA 허용 등

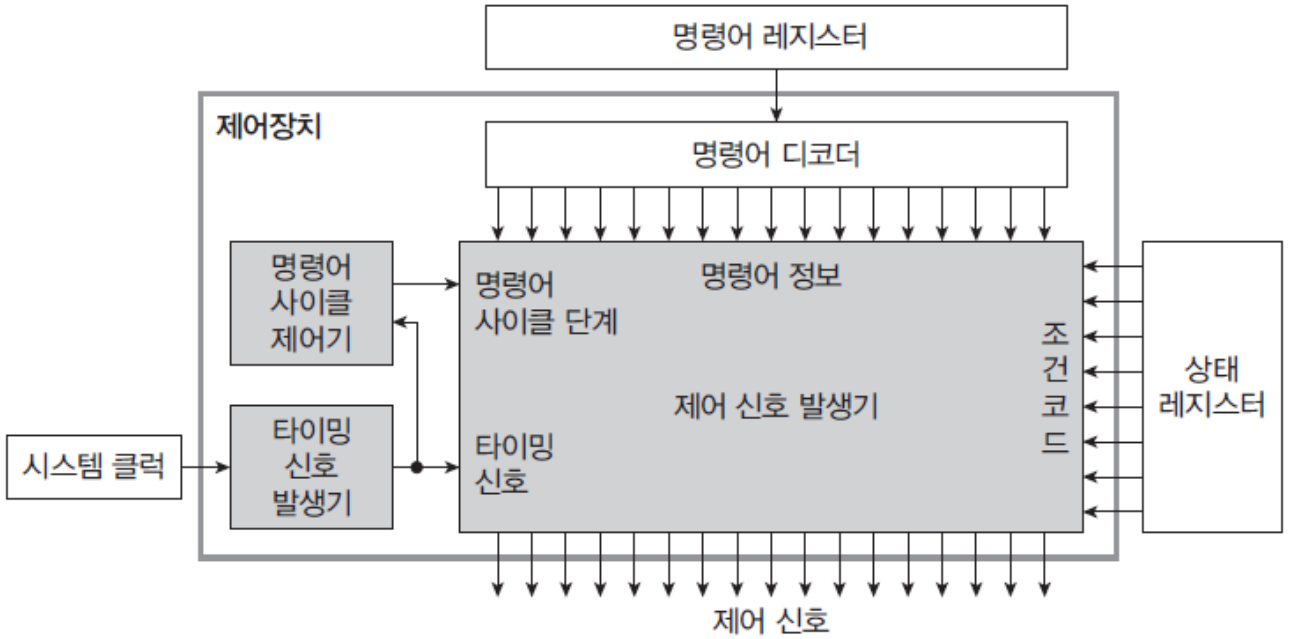
8.3.2 제어장치 구조

제어장치 구현 방법

구분	하드와어어드 제어장치	마이크로프로그램 제어장치
구현 방법	순서 제어: 상태 머신 동작 제어: 조합 회로	순서 제어: 순서제어기(sequencer) 동작 제어: 제어 기억장치
구현 과정	간소화, 상태도. 복잡하고 어렵다.	마이크로프로그래밍. 간단하고 쉽다.
동작 속도	빠르다.	느리다.
회로 수정	회로를 다시 연결한다. 복잡하다.	하드웨어는 항상 동일하다. 간단하다.
적용 분야	RISC	CISC

하드와이어드 제어 장치

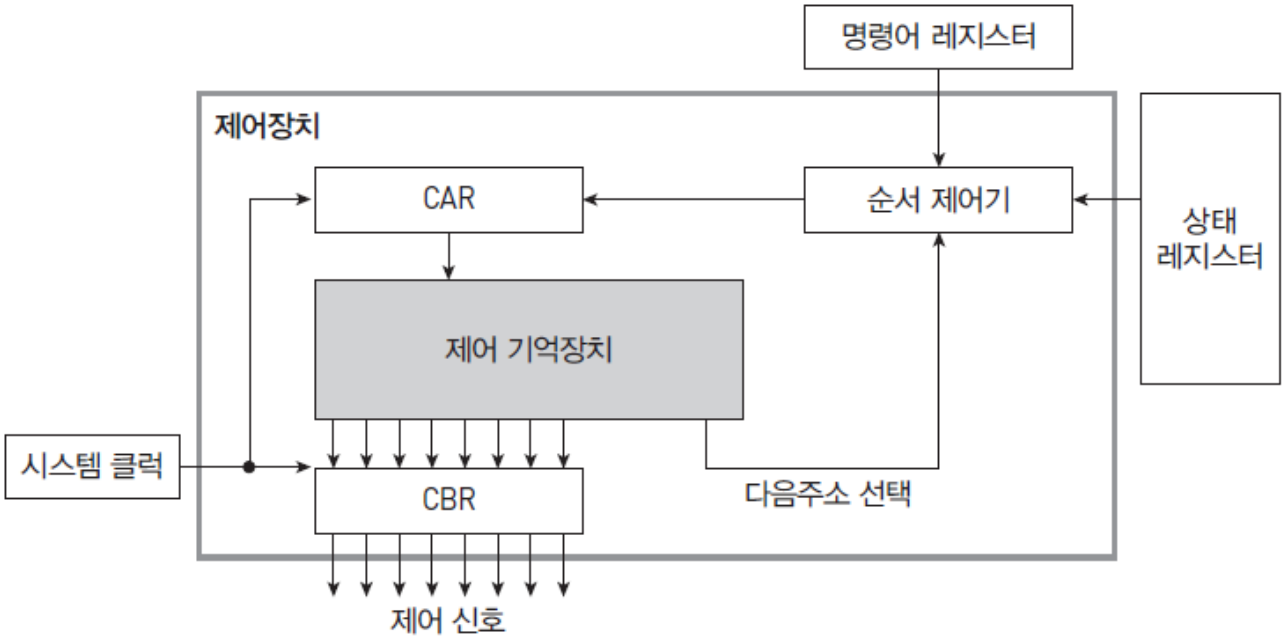
- 명령어 사이클 제어기: 명령어 사이클 단계 구별
- 타이밍 신호 발생기: 클럭 펄스 구분
- 명령어 디코더: 명령어 코드를 해석하여 명령어 종류 구별
- 제어신호 발생기: 제어 신호 생성



[그림 8-17] 하드와이어드 제어장치

마이크로프로그램 제어장치

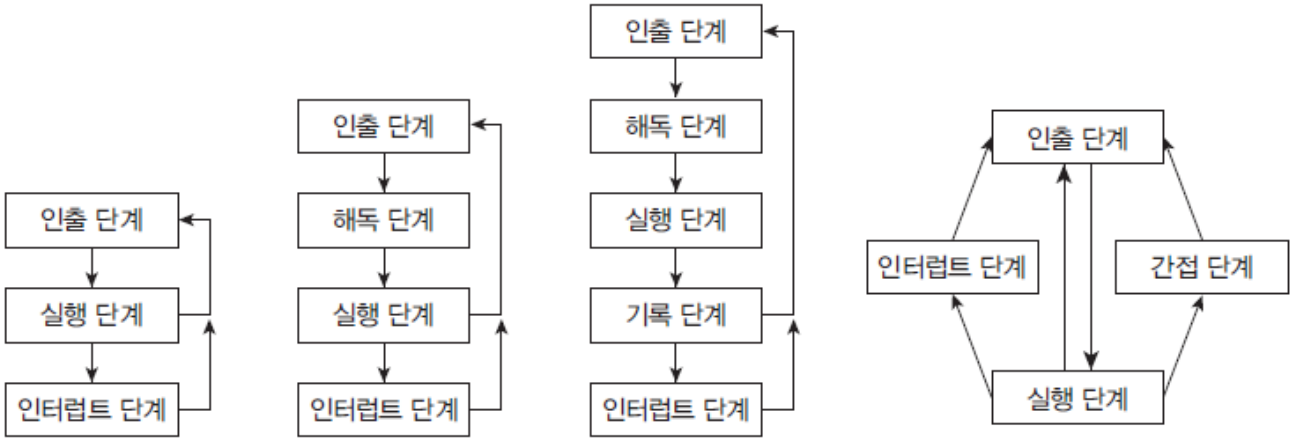
- 제어 기억장치(Control Memory): 제어 신호 저장
- 제어주소 레지스터(CAR, Control Address Register): 제어 기억장치 주소 공급
- 제어 버퍼 레지스터(CBR, Control Buffer Register): 제어 신호를 외부로 제공
- 순서제어기(microprogram sequencer): CAR에 적재할 다음주소 결정



[그림 8-18] 마이크로프로그램 제어장치

8.3.3 명령어 사이클

- 명령어 사이클의 기본 구성: 인출 단계/실행 단계/인터럽트 단계
- 프로세서마다 명령어 사이클의 실행 단계를 세분화
  - 해독 단계(decode stage): 명령어 코드를 해석
  - 기록 단계(write stage): 명령어 실행 결과를 레지스터나 기억장치로 저장
  - 간접 단계(indirect stage): 기억장치에서 데이터 인출



[그림 8-19] 명령어 사이클

8.4 제어장치 요약

- 명령어 실행 과정의 계층적 분해
  - 명령어 사이클
  - 명령어 사이클 단계
  - 마이크로오퍼레이션 (클럭 단위 동작)
- 제어장치 구현 방법
  - 하드와이어드 제어장치: 순차 조합 논리회로로 구현
  - 마이크로프로그램 제어장치: 기억장치로 구현
- 명령어 사이클 세분화

요약

- 8.1 중앙처리장치 구조
  - 제어장치
  - 처리장치: 연산기, 레지스터
- 8.2 레지스터
  - 제어용: PC, IR, MAR, MBR
  - 명령어 실행용: SR, AR, DR, SP, IX(인덱스 레지스터)
- 8.3 인터럽트
  - 외부 인터럽트, 내부 인터럽트
  - 인터럽트 단계, 인터럽트 가능 플래그, 인터럽트 서비스 루틴
- 8.4 제어장치
  - 제어장치 기능: 순서제어, 동작 제어
  - 제어장치 구조: 하드와이어드 제어장치, 마이크로프로그램 제어장치
  - 명령어 사이클: 인출, 해독, 간접, 실행, 기록, 인터럽트 단계