[JO2122] 컴퓨터구조

2022년 1학기

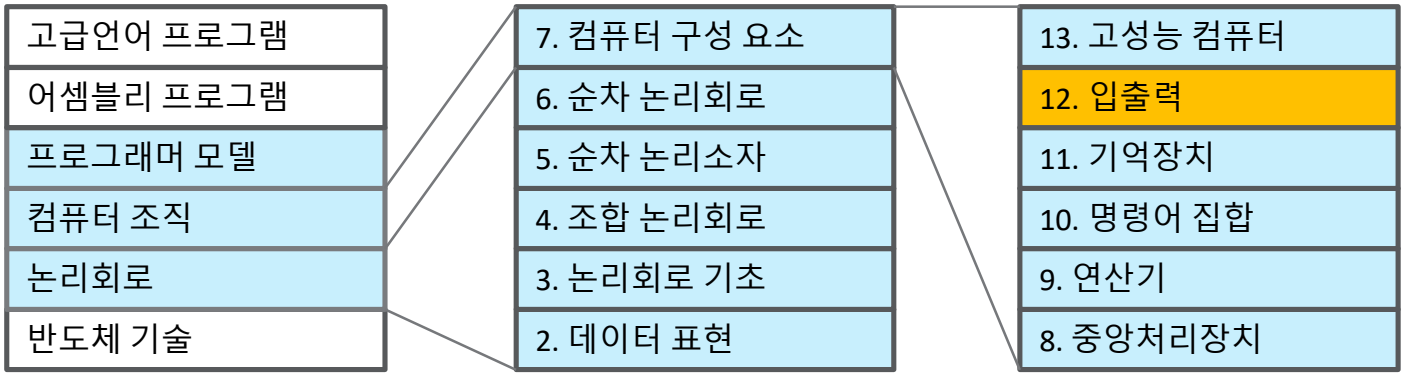
상명대학교 소프트웨어학과 박희민

12.1 입출력장치 개요
12.2 프로그램 구동 입출력
12.3 인터럽트 구동 입출력
12.4 직접 기억장치 액세스
12.5 요약

2022-06-01

CHAP12 입출력

제12장 입출력



- 학습 목표
  - 컴퓨터와 입출력장치를 연결하는 구조를 표현할 수 있다.
  - 컴퓨터와 입출력장치 간에 데이터를 전송하는 방법을 설명할 수 있다.
- 내용

12.1 입출력장치 개요
12.2 프로그램 구동 입출력
12.3 인터럽트 구동 입출력
12.4 직접 기억장치 액세스
12.5 요약

12.1 입출력장치 개요

- 입출력 장치 = 주변 장치(peripheral)
  - 종류가 많고 다양하다.
  - 중앙처리장치나 기억장치와 동작 방법이 다르다.
  - 입출력 포트(I/O ports)를 통해 데이터를 교환한다.
- 학습 목표
  - 여러 가지 입출력장치를 분류할 수 있다.
  - 중앙처리장치와 입출력장치를 연결하는 방법을 설명할 수 있다.
- 내용

12.1.1 입출력장치 종류

12.1.2 입출력 모듈

12.1.3 입출력 포트

12.1.4 입출력 방법

12.1.1 입출력장치 종류

< 표 12-1> 입출력장치의 분류

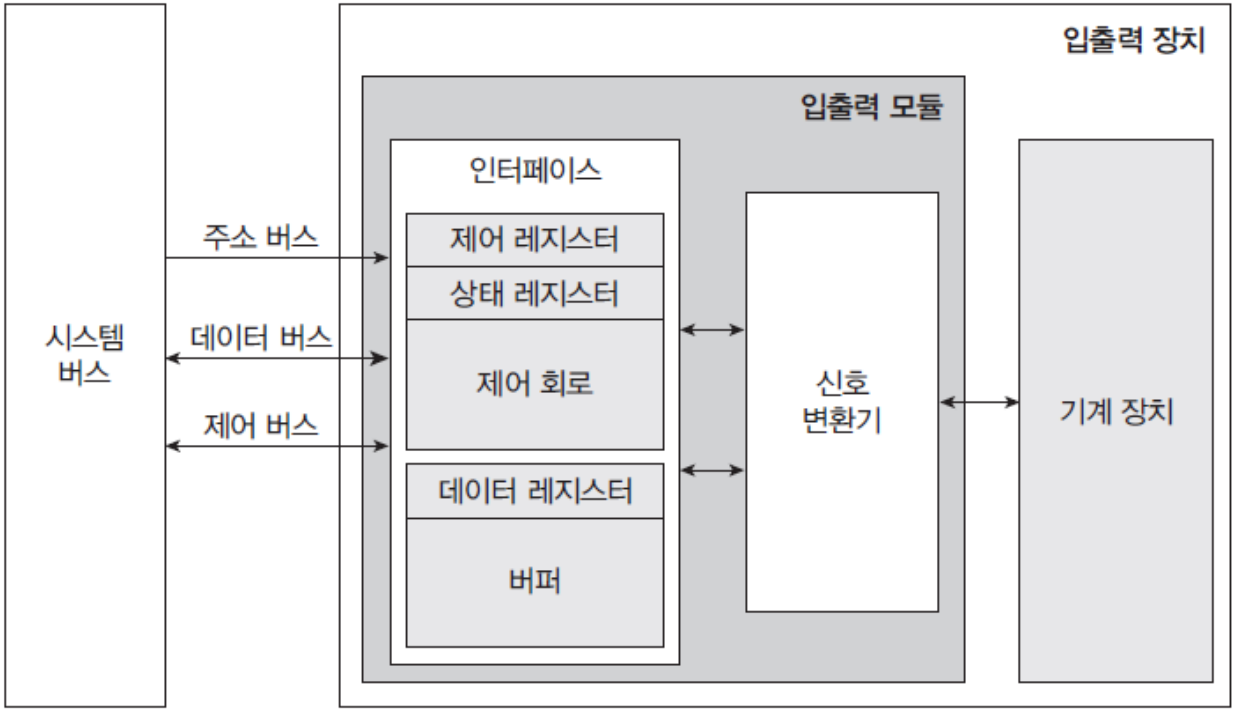
주변장치	데이터 전달 방향			사용자				전송 방식		통신 방식	
	입력	출력	입출력	사람	기계	통신	컴퓨터	문자	블록	직렬	병렬
마우스	√			√				√		√	
모니터		√		√					√		√
센서	√				√			√		√	√
모터		√			√			√		√	√
모뎀			√			√		√		√	
네트워크 카드			√			√			√		√
하드 디스크			√				√		√		√
USB 기억장치			√				√		√	√	

12.1.2 입출력 모듈

- 중앙처리장치와 입출력장치의 차이점
  - 동작 속도: 느리다.
  - 동작 방식: 전자 또는 기계식
  - 데이터 형식: 문자(바이트) 단위
  - 오류 발생 가능성: 크다.

- 입출력 모듈(I/O module)
  - 컴퓨터와 인터페이스 제공
  - 기능: 신호 변환, 버퍼링, 오류 검출

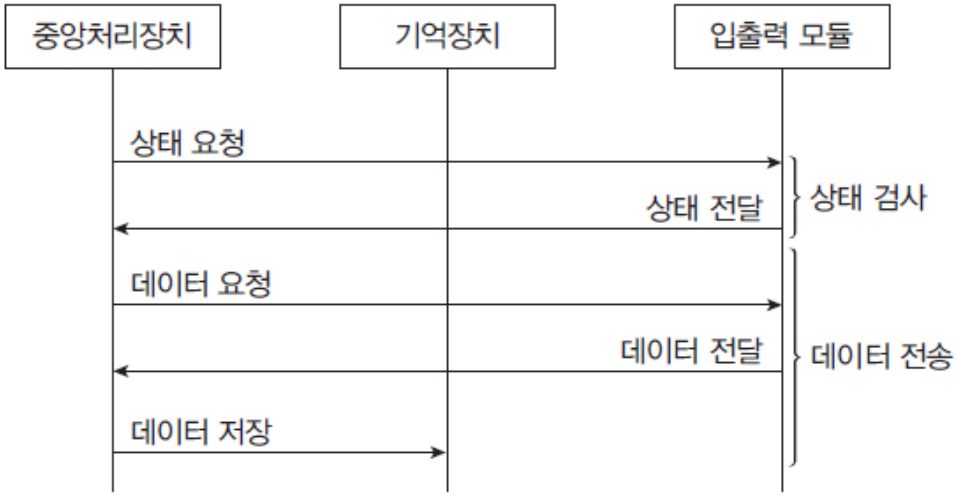
입출력장치의 구조



[그림 12-1] 입출력장치의 구조

- 제어 레지스터: 입출력장치의 기능과 동작 방식 결정
- 상태 레지스터: 입출력장치의 내부 상태를 중앙처리장치로 전달
- 데이터 레지스터: 데이터 형식 변환, 중앙처리장치와 데이터 교환
- 버퍼: 전송 속도 차이 문제 해결

핸드쉐이킹 (handshaking)



[그림 12-2] 간단한 핸드쉐이킹

- 핸드쉐이킹 = 프로토콜
  - 중앙처리장치와 입출력장치간의 통신 흐름을 제어하는 절차 또는 규약
- 처리 과정
  - 상태 검사: 입출력장치가 데이터 전송 준비를 마쳤는지 확인
  - 데이터 전송: 실제 데이터 전달

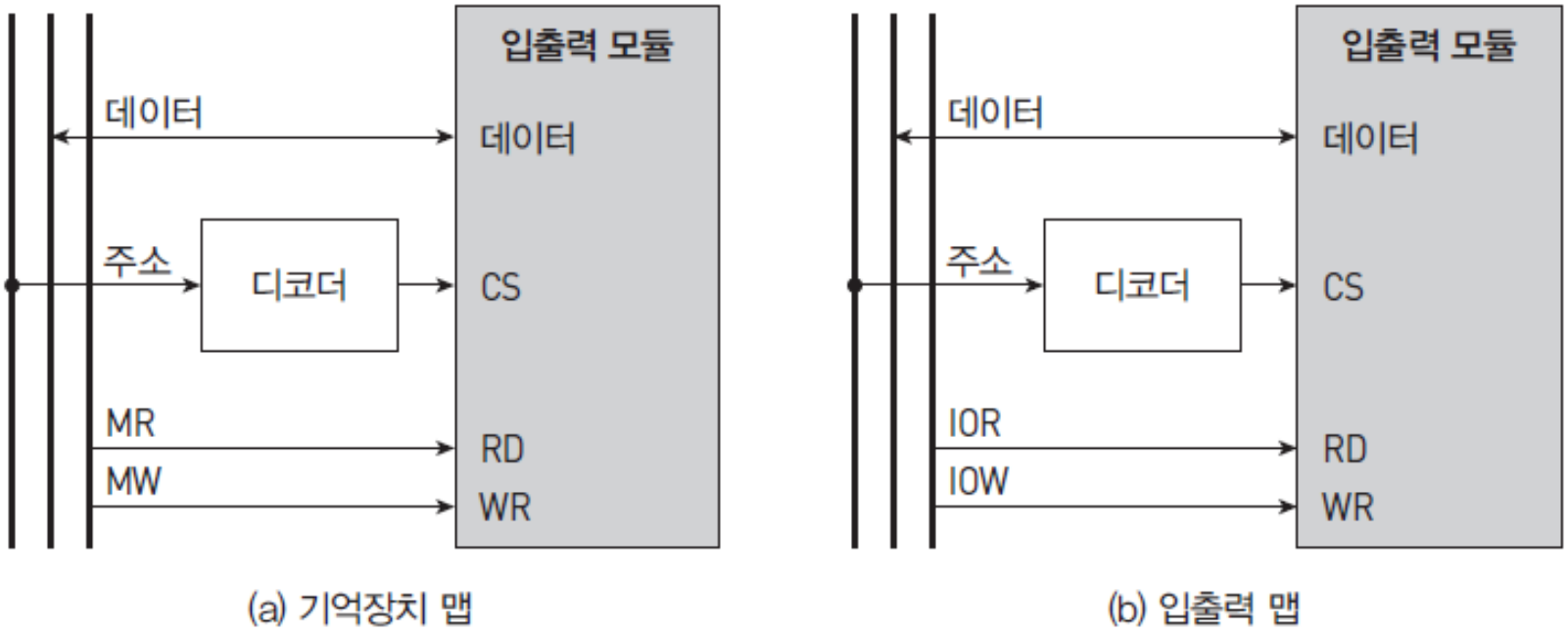
12.1.3 입출력 포트

- 입출력 포트 = 입출력 주소
  - 기억장치 맵(memory map): load, store 명령어
  - 독립 입출력 맵 (isolated I/O map): input, output 명령어
- 명령어(instruction)와 명령(command)
  - 명령어: 중앙처리장치가 실행하는 기본 동작을 나타내는 코드
  - 명령: 입출력장치가 의미 있는 동작을 수행하도록 명령어를 실행함으로써 전달하는 데이터 혹은 데이터를 전달하는 행위

<표 12-2> 입출력장치에 대한 제어신호

제어 신호	의미	명령어	비고
MR(Memory Read)	기억장치 주소에서 데이터를 읽는다.	Load	모든 프로세서가 제공한다.
MW(Memory Write)	기억장치 주소에 데이터를 쓴다.	Store	
IOR(I/O Read)	입출력 포트에서 데이터를 읽는다.	Input	제공하지 않는 프로세서도 존재한다.
IOW(I/O Write)	입출력 포트에 데이트를 쓴다.	Output	

입출력 포트 할당

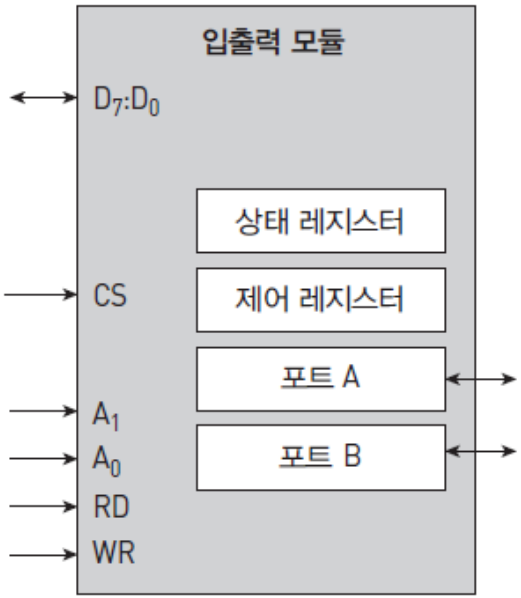


(a) 기억장치 맵 (b) 입출력 맵

[그림 12-3] 입출력 포트 할당

입출력 모듈: GPIO 장치

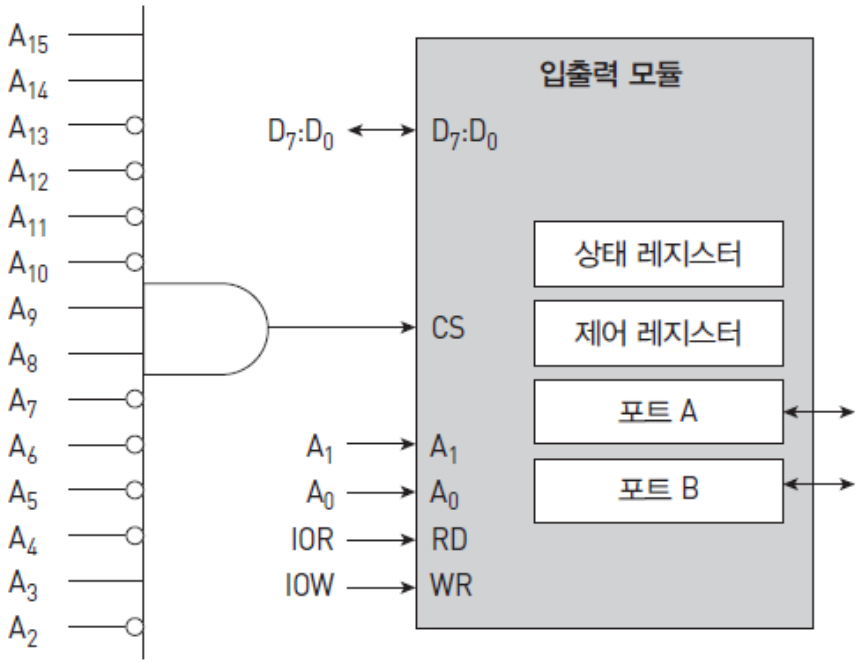
- GPIO (General Purpose Input/Output) 장치
  - 설정에 따라 입력과 출력 중 하나로 사용할 수 있는 입출력장치
- GPIO 장치와 동작 특성표의 예



[그림 12-4] GPIO 장치

CS	A_1	A_0	RD	WR	동작
0	X	X	X	X	동작하지 않는다.
1	0	0	1	0	상태 레지스터를 데이터로 출력한다.
1	0	1	0	1	제어 레지스터에 데이터를 기록한다.
1	1	0	1	0	포트 A의 값을 데이터로 출력한다.
1	1	0	0	1	포트 A에 데이터를 기록한다.
1	1	1	1	0	포트 B의 값을 데이터로 출력한다.
1	1	1	0	1	포트 B에 데이터를 기록한다.

GPIO 장치 연결



[그림 12-5] GPIO 장치 연결

- 입출력 주소 공간
  - 0000h ~ FFFFh
- 입출력 포트
  - C308h ~ C30Bh
  - C308h: 상태 레지스터
  - C309h: 제어 레지스터
  - C30Ah: 포트 A
  - C30Bh: 포트 B
- 예제
  - INPUT R0, C308h
  - OUTPUT C309h, R0
  - INPUT R1, C30Ah
  - OUTPUT C30Bh, R1

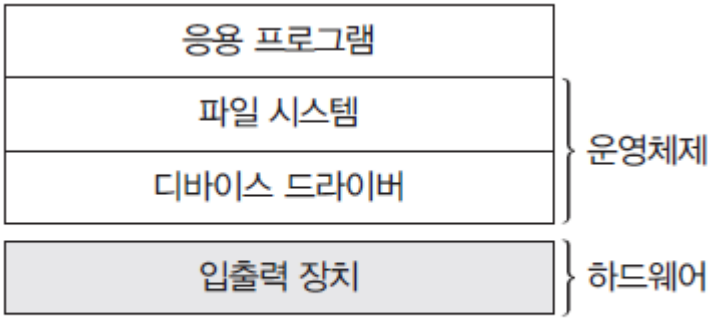
12.1.4 입출력 방법

<표 12-4> **입출력 방법**

입출력 방법	상태 검사	입출력 동작	기억장치 전송	준비 사항
프로그램에 의한 입출력	중앙처리장치	중앙처리장치	중앙처리장치	없음
인터럽트 구동 입출력	없음	중앙처리장치	중앙처리장치	인터럽트 벡터 인터럽트 서비스 루틴
직접 기억장치 액세스	없음	DMA 제어기	DMA 제어기	DMA 제어기 초기화

디바이스 드라이버

- 디바이스 드라이버
  - 응용 프로그램 개발자를 위하여 입출력장치를 제어하는 기능을 제공하는 프로그램
  - 프로그래머에게 입출력장치를 감춘다.
  - 프로그래머는 입출력장치를 파일과 같이 사용할 수 있다.
  - 운영체제의 일부



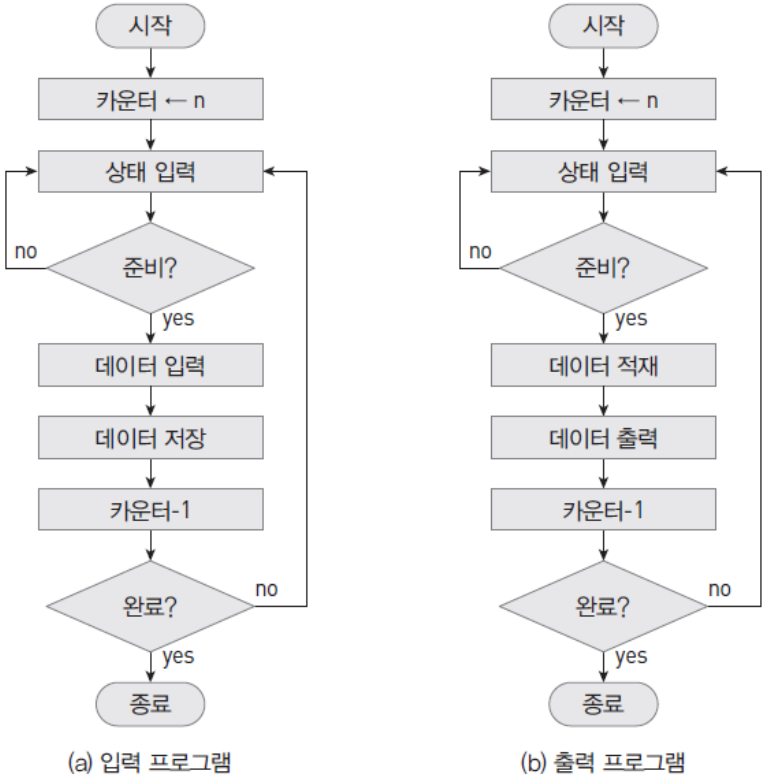
[그림 12-6] 소프트웨어 계층

12.1 입출력장치 개요 요약

- 입출력장치 특성
  - 기계장치 동작을 포함한다.
  - 컴퓨터 시스템의 동작보다 느리다.
  - 입출력 모듈을 통해 시스템 버스에 연결된다.
- 입출력 포트
  - 입출력장치에게 할당되는 주소
- 입출력 방법
  - 프로그램 구동 입출력
  - 인터럽트 구동 입출력
  - 직접 기억장치 액세스

12.2 프로그램 구동 입출력

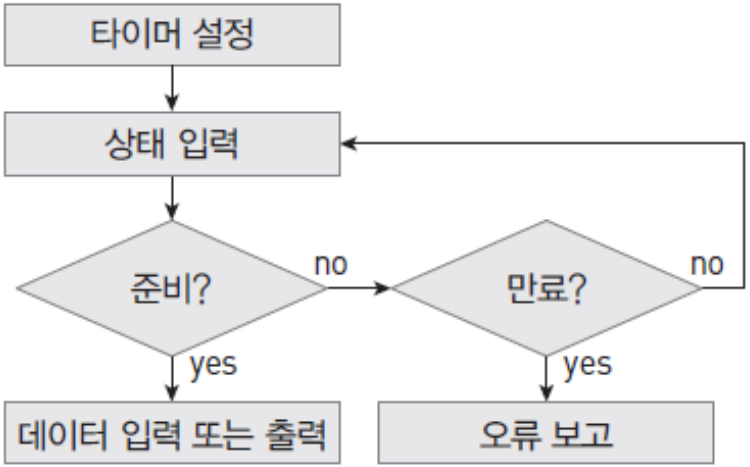
- 프로그램 구동 입출력(Programmed I/O)
  - 중앙처리장치가 프로그램을 실행함으로써 입출력 과정 통제
  - 상태 검사
  - 데이터 입력/저장
  - 데이터 적재/출력
- 단점
  - 클럭 사이클 낭비
- 예
  - 명령어 실행 시간 1μsec
  - 입출력장치 동작 시간 1msec
  - 상태 검사에 명령어 2개 실행
  - 상태 검사 횟수: 약 500번



[그림 12-7] 입출력 프로그램의 흐름도

워치독 타이머

- 워치독 타이머(Watch-dog timer)
  - 무한 루프 방지
  - 입출력장치가 동작하는 최대 시간 설정
  - 대기 과정에 시간 검사
  - 타이머가 만료되면 오류 보고



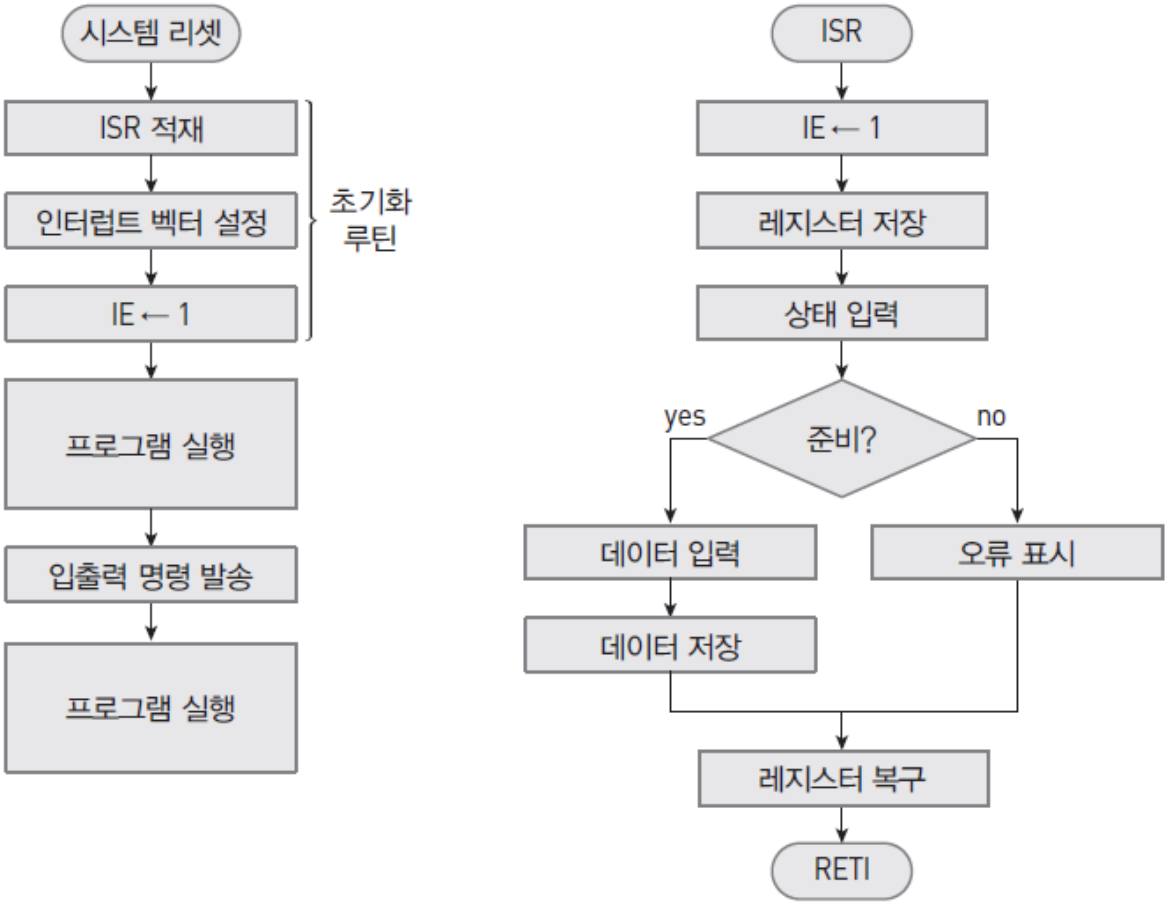
[그림 12-8] 워치독 타이머 활용

12.3 인터럽트 구동 입출력

- 인터럽트 구동 입출력
  - 중앙처리장치가 입출력 장치 준비 사항을 검사하지 않는다.
  - 인터럽트 서비스 루틴(인터럽트 핸들러)에서 입출력 처리
- 학습 목표
  - 다중 인터럽트 환경에서 인터럽트를 요청한 장치를 구별할 수 있다.
  - 인터럽트 서비스 루틴을 찾는 방법을 설명할 수 있다.
- 내용

12.3.1 인터럽트 서비스 루틴
12.3.2 다중 인터럽트 처리
12.3.3 소프트웨어 폴링
12.3.4 다중 인터럽트 요청선
12.3.5 데이지 체인
12.3.6 우선순위 인코더

12.3.1 인터럽트 서비스 루틴

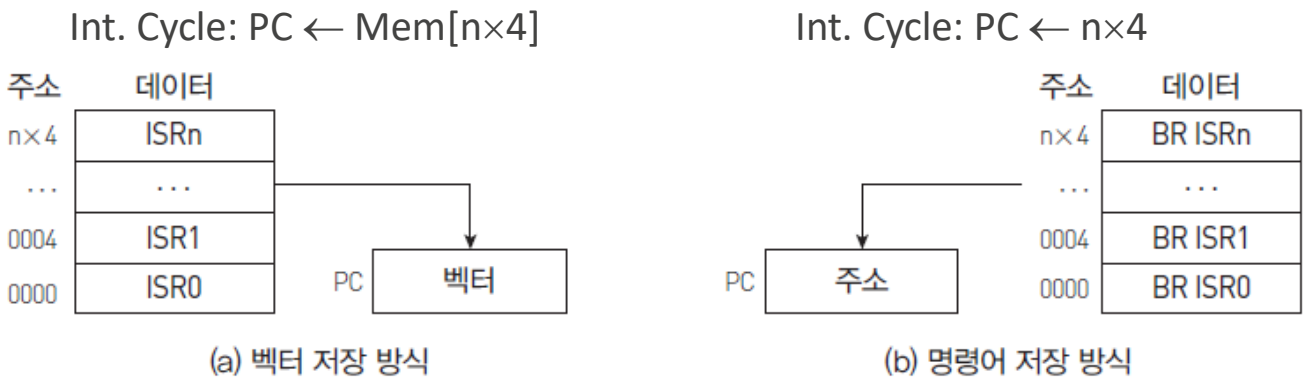


(a) 시스템 초기화 (b) 인터럽트 서비스 루틴

[그림 12-9] 인터럽트에 의한 데이터 입력 처리 과정

12.3.2 다중 인터럽트 처리

- 다중 인터럽트(multiple interrupt)
  - 여러 개의 인터럽트 소스
  - 인터럽트 소스(인터럽트를 요청한 장치) 구별 필요
- 인터럽트 벡터 테이블
  - 인터럽트 서비스 루틴의 시작 주소(ISR 주소)를 모아 놓은 기억장치 영역
  - 인터럽트 사이클에서 인터럽트 요청 장치 번호(n)를 확인하고
  - 인터럽트 벡터 테이블에서 ISR 주소를 구한다.



[그림 12-10] 인터럽트 벡터 테이블

다중 인터럽트 처리 방법

- 인터럽트 구별 방법

12.3.3 소프트웨어 폴링

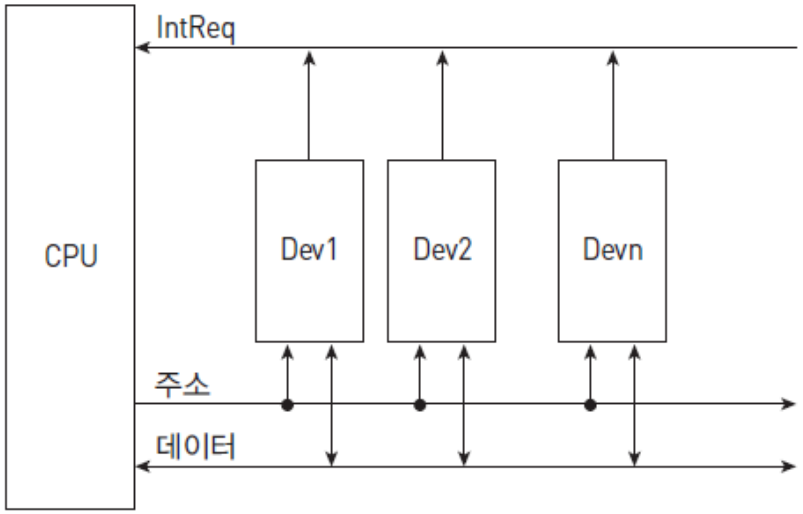
12.3.4 다중 인터럽트 요청선

12.3.5 데이지 체인

12.3.6 우선순위 인코더

- 관련 주제
  - 하드웨어 구조
  - 인터럽트 요청 장치 구별 방법
  - ISR 결정 방법
  - 우선순위 결정 방법

12.3.3 소프트웨어 폴링



(a) 회로도

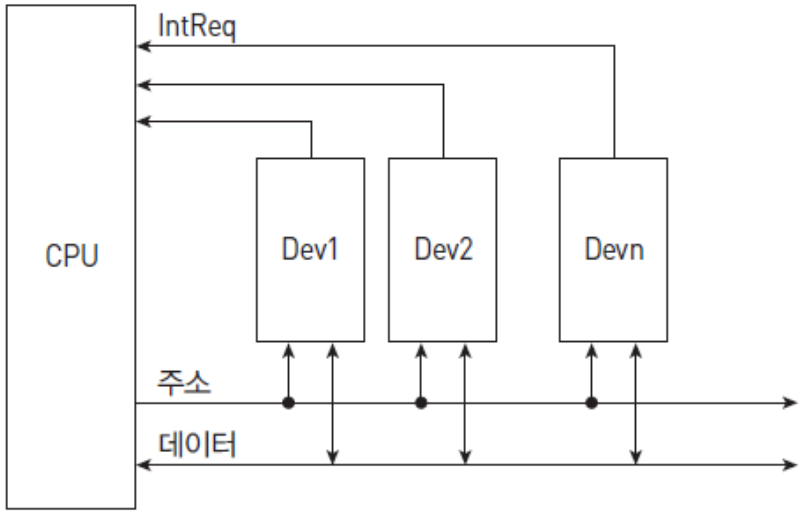
```
Interrupt ISR( )
{
   if (Dev1 requests an interrupt)
      call int-handler-1( );
   else if (Dev2 requests an interrupt)
      call int-handler-2( );
   . . .
   else if (Devn requests an interrupt)
      call int-handler-n( );
   return from interrupt;
}

int-handler-1( ){ . . . }
int-handler-2( ){ . . . }
int-handler-n( ){ . . . }
```

(b) 인터럽트 서비스 루틴

- 폴링: 프로그램으로 물어보는 동작
  - 인터럽트 요청 장치: 대표 인터럽트 서비스 루틴에서 프로그램으로 확인
  - 인터럽트 서비스 루틴: 대표 인터럽트 서비스 루틴에서 핸들러 함수 호출
  - 우선순위: 소프트웨어적으로 인터럽트를 요청하였는지 물어보는 순서

12.3.4 다중 인터럽트 요청선



(a) 회로도

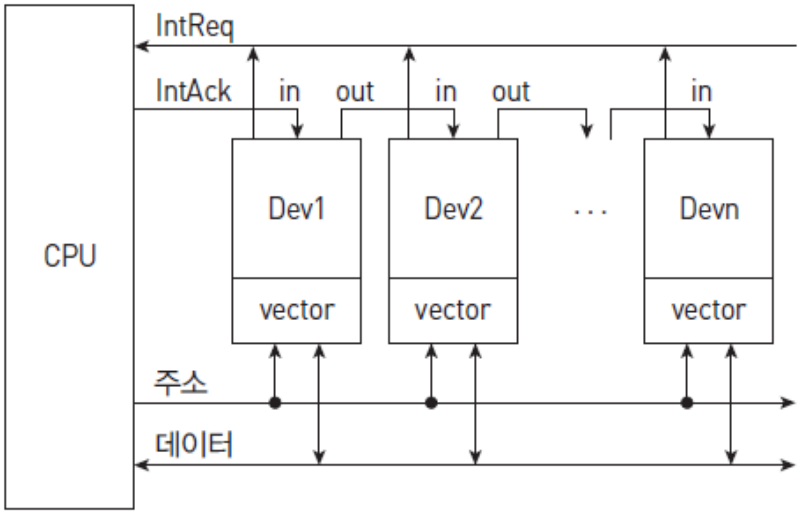
```
int-handler-1( )
{
   save registers;
   process I/O;
   restore registers;
   return from interrupt
}

int-handler-2( ){ · · · }
 · · ·
 · · ·
int-handler-n( ){ · · · }
```

(b) 인터럽트 서비스 루틴

- 다중 인터럽트 요청선: 장치마다 별도의 인터럽트 요청선 사용
  - 인터럽트 요청 장치: 인터럽트 요청선으로 구별
  - 인터럽트 서비스 루틴: 인터럽트 요청선마다 별도의 서비스 루틴
  - 우선순위: 하드웨어로 IntReq에 우선순위 부여

12.3.5 데이지 체인



(a) 회로도

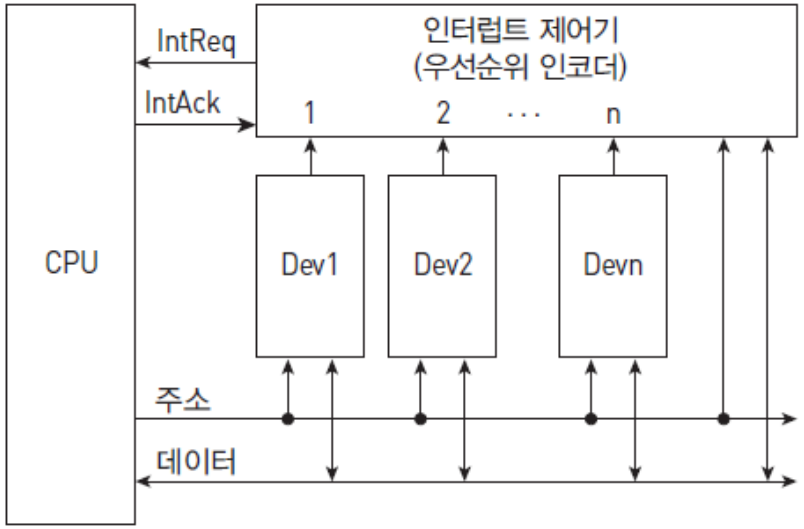
```
Boot( )
{
   . . .
   write int.vector into each I/O module;
   . . .
}

int-handler-1( ){ . . . }
int-handler-2( ){ . . . }
int-handler-n( ){ . . . }
```

(b) 인터럽트 서비스 루틴

- Daisy chain = hardware polling
  - 인터럽트 요청 장치: IntAck를 체인 형태로 연결
  - 인터럽트 서비스 루틴: IntAck를 보내고 데이터 버스를 읽어 ISR 결정
  - 우선순위: 체인 연결 순서

12.3.6 우선순위 인코더



(a) 회로도

```
Boot( )
{
  . . .
  initialize interrupt vector table
  . . .
}

int-handler-1( ){ · · · }
int-handler-2( ){ · · · }
int-handler-n( ){ · · · }
```

(b) 인터럽트 서비스 루틴

- 인터럽트 제어기: 우선순위 인코더 내장
  - 인터럽트 요청 장치: 우선순위 인코더가 장치 구별
  - 인터럽트 서비스 루틴: IntAck를 보내고 데이터 버스를 읽어 인터럽트 번호 결정
  - 우선순위: 인터럽트 제어기에 있는 우선순위 인코더

다중 인터럽트 종합

- 다중 인터럽트 처리 방법 중복 사용
  - 다중 인터럽트 요청선
  - {데이지 체인, 우선순위인코더} 중 하나
  - 한 개의 인터럽트 요청선을 소프트웨어 폴링으로 공유

- 예: 2개 이상의 인터럽트 요청선 제공
  - NMI(Non-maskable interrupt)
  - I/O interrupts: 공유 가능: 소프트웨어 폴링으로 구별

12.3 인터럽트 구동 입출력 요약

- 인터럽트 구동 입출력
  - 인터럽트 서비스 루틴에서 입출력 동작 수행
  - 상태 검사 시간 절약
- 인터럽트 벡터 테이블
  - 인터럽트 서비스 루틴의 시작 주소를 모아 놓은 주기억장치 영역
- 다중 인터럽트 처리 방법
  - 소프트웨어 폴링: 프로그램을 수행하여 인터럽트 요청 여부를 확인
  - 다중 인터럽트 요청선: 인터럽트 요청선을 분리
  - 데이지 체인: 인터럽트 확인선을 체인 형태로 연결
  - 우선순위 인코더: 인터럽트 제어기 사용

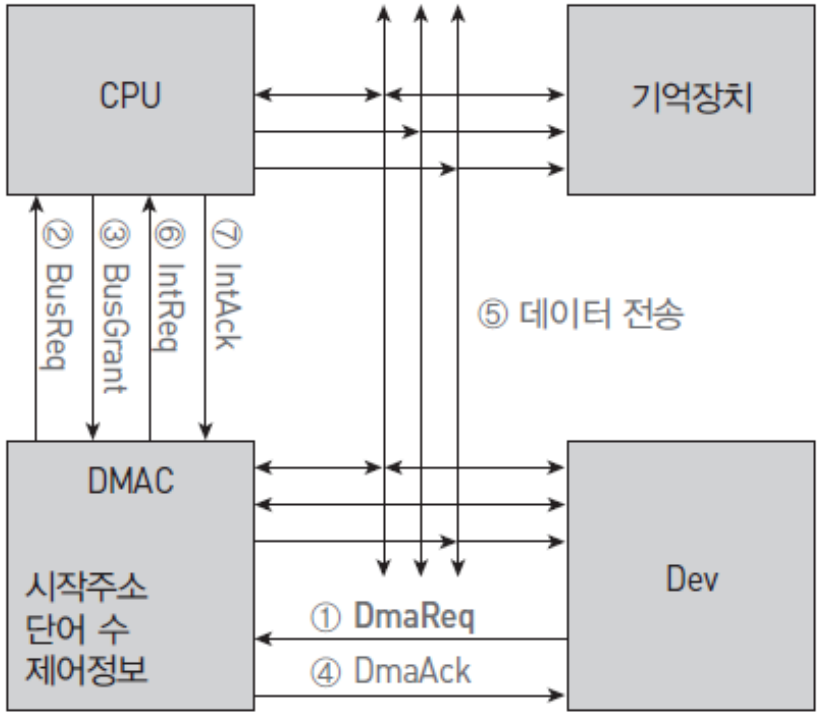
12.4 직접 기억장치 액세스

- DMA
  - 입출력장치와 기억장치간 직접 데이터 전송
    - 프로그램이 DMA 제어기에게 데이터 전송 명령 하달
    - 중앙처리장치는 다른 작업 실행
    - 입출력장치가 준비 되면 버스 사용 요청
    - DMA 제어기 지휘하에 데이터 전송
  - 한 번에 블록(512 바이트 ~ 4 K바이트) 전송
  - 컨텍스트 스위치 불필요
  - DMA 제어기(DMA controller) 필요
  - 한 개의 DMA 제어기는 여러 개의 채널 관리
  - Cycle stealing

DMA 전송 과정

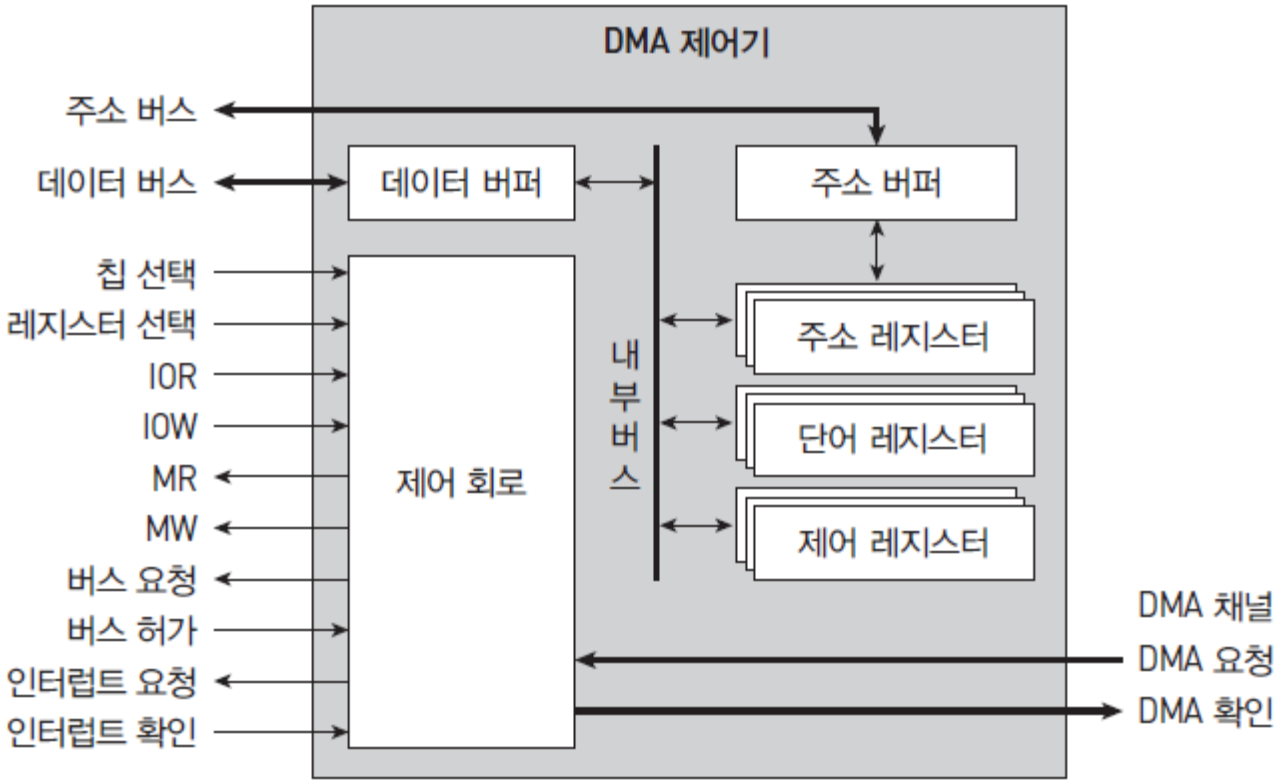
- 준비
  - 기억장치 주소
  - 단어 수
  - 데이터 전송 방향
  - 입출력 채널 번호
- 처리 과정

① DMA request
② Bus request
③ Bus grant
④ DMA acknowledge
⑤ Memory transfer by DMAC
⑥ Interrupt request
⑦ Interrupt acknowledge



[그림 12-15] DMA 제어기 연결과 DMA 처리 과정

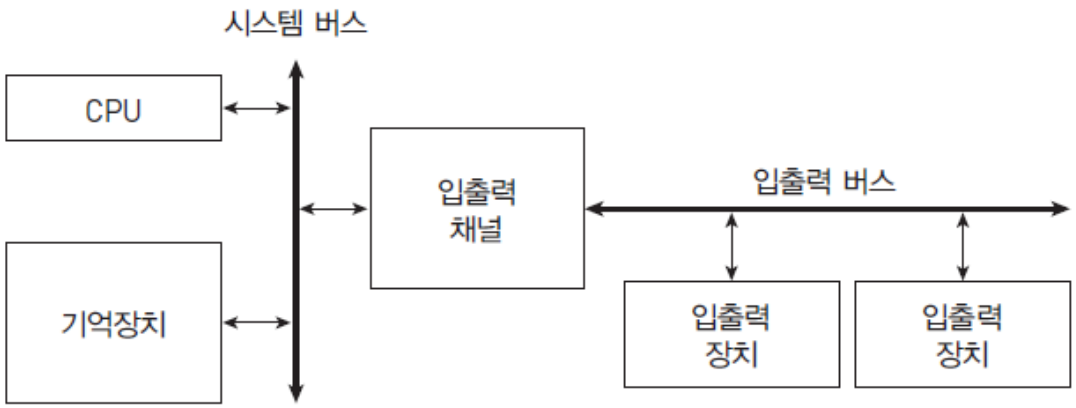
DMA 제어기 구조



[그림 12-16] DMA 제어기의 내부 구조

입출력 채널

- 입출력 채널 = 입출력 프로세서
  - 일종의 버스 제어기
  - 중앙처리장치는 입출력 채널에게 명령 하달
  - 입출력 채널이 입출력 장치와 데이터 송수신
  - 입출력 채널은 DMA 방식으로 기억장치 액세스



[그림 12-17] 입출력 채널

12.4 직접 기억장치 액세스 요약

- DMA 입출력
  - DMA 제어기가 데이터 전송 과정 관리
- 데이터 전송 준비
  - 기억장치 주소
  - 단어 수
  - 데이터 전송 방향
  - 입출력 채널 번호
- DMA 과정
  - 입출력장치가 DMA 요청
  - DMA 제어기가 시스템 버스 구동
  - 블록 단위 데이터 전송

12.5 요약

- 입출력장치 개요
  - 종류가 많고 기계적인 요소 포함
  - 입출력 포트: 기억장치 맵, 입출력 맵
- 프로그램 구동 입출력
  - 중앙처리장치가 프로그램을 수행하여 상태 검사 및 데이터 전송
  - 상태 검사에 시간 낭비
  - 워치독 타이머: 무한 루프 방지
- 인터럽트 구동 입출력
  - 인터럽트 벡터
  - 소프트웨어 폴링, 다중 인터럽트 요청선, 데이지 체인, 우선순위 인코더
- 직접 기억장치 액세스(DMA)
  - DMA 제어기에 의한 블록 단위 전송
  - 사전 준비 사항: 기억장치 주소, 단어 수, 전송 방향, 입출력 채널 번호