2학기

인하공업전문대학
INHA TECHNICAL COLLEGE

운영체제

(Operating System)

조규철교수

입출력 제어 및 인터럽트

CONTENTS

입출력 제어 및 인터럽트

CONTENTS

입출력 제어 및 인터럽트

01 5.1 입출력 장치의 구조 및 종류

운영체제의 주요 기능

- ✓ 자료 입출력(input/output)을 위한 모든 컴퓨터의 입출력 장치 제어
- ✓ 입출력 장치와 시스템의 다른 부분과의 인터페이스 기능을 제공해야 한다

【표 2-1】 **운영체제의** 입출력 프로그램 기능

기본기능

- ✓ 블록 단위의 데이터 전송 기능
- ✓ 컴퓨터 본체나 보조기억 장치와 외부 환경과의 인터페이스 기능
- ✓ 하드웨어와는 독립적인 관리가 가능한 자원과의 독립성 유지 기능
- ✓ 프로세스 개념을 사용한 쉽고 편리한 입출력 수행 기능

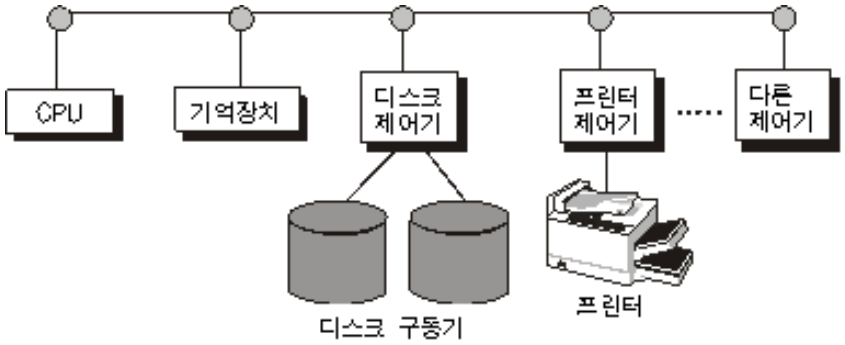
주요기능

- ✓ 주변장치와 주기억장치 간의 블록 단위 데이터 전송 기능
- ✓ 주변장치와 CPU간의 논리적 특성과 물리적 특성 일치 기능
- ✓ 주변장치와 주기억장치 간의 자원 공유 기능

01 5.1 입출력 장치의 구조 및 종류

- 입출력 하드웨어 장치 : 블록 장치(block device)/문자 장치(character device)로 분류
  - ✓ 블록장치: 자신의 주소를 가지고 고정된 크기의 블록으로 정보를 저장하는 장치
  - ✓ 문자장치는 어떤 블록 구조와도 상관없이 문자 스트림(stream)으로 전송하거나 수신가능
- 장치 제어기(device controller) : 입출력 장치의 전기적인 구성요소
- I/O장치는 CPU와 장치제어기 사이의 통신을 위해 단일버스를 사용

- 입출력 프로그래밍의 4가지 목적
  1. 장치와의 독립성 유지
  2. 입출력 시 오류 처리
  3. 입출력 시 동기와 비동기 전송
  4. 입출력 시 장치의 공유와 전용



【그림 2-21】 장치 제어기 연결 구조

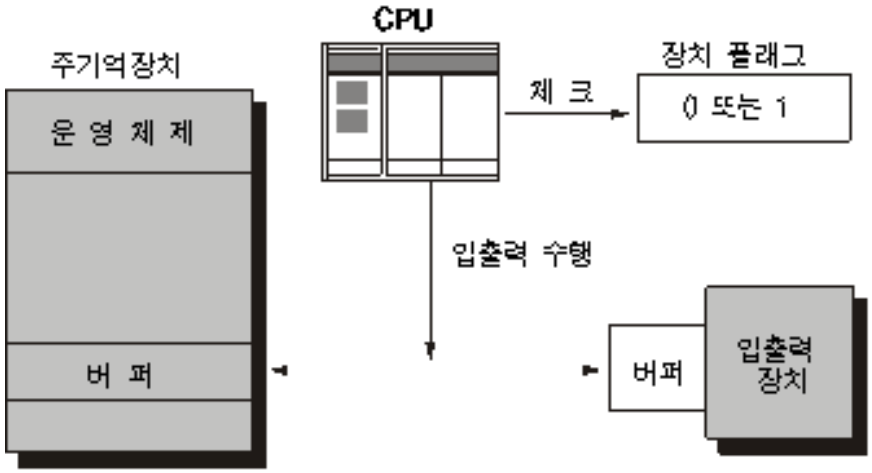
01 5.2 입출력 방식의 종류

- CPU와 입출력 장치와의 처리 속도 수십~ 수십만 배 차이가 남
- 컴퓨터 시스템에서 입출력 하는 방법
  - ✓ CPU가 직접 담당하여 입출력 하는 직접 입출력 방법
  - ✓ 입출력 채널(I/O channel) 또는 직접 기억장치 접근(DMA : Direct Memory Access)에 의해 수행되는 간접 입출력 방법

5.2 입출력 방식의 종류

(1) 직접 입출력 방식

- 초기의 컴퓨터에서 사용된 방식
- CPU가 프로그램 수행 중 입출력 명령을 만나면 직접 입출력을 수행
- 전체 프로그램의 실행 시간은 프로그램의 입출력 시간에 실행시간의 합
- CPU가 장치 플래그(flag)의 상태를 검사하여 상태 플래그가 0일 때만 입출력을 수행



【그림 2-2】 직접 입출력 방식

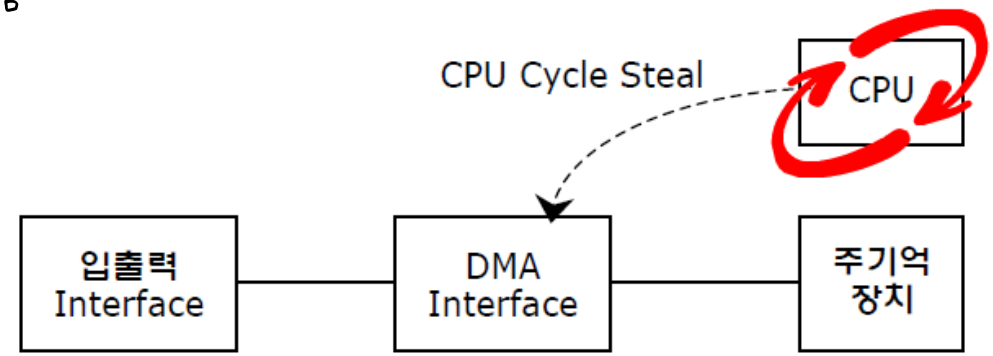
5.2 입출력 방식의 종류

(2) 간접 입출력 방식

- CPU가 직접 입출력을 담당하지 않고 전용 입출력 프로세서인 DMA나 채널(channel)을 사용하는 방식
- 프로그램이 수행되는 도중에 입출력 명령이 발생하면,
  CPU에게 입출력 명령의 종류, 장치 이름, 입출력 프로그램의 주소 등과 같은 정보가 전달되며,
  CPU는 입출력 프로세서에게 입출력 명령을 주고, 입출력 프로세서로부터 입출력 완성신호가 올 때까지 작업을 중단하지 않고 다른 프로그램을 수행

1) 직접 메모리 접근(DMA, direct memory access) 방식

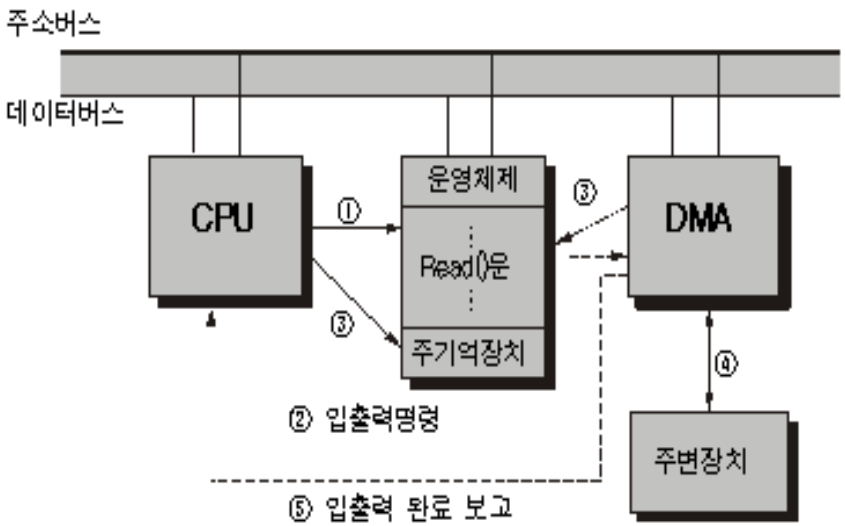
- 주변장치들(하드디스크, 그래픽 카드, 네트워크 카드, 사운드 카드)이 메모리에 직접 접근하여 읽거나 쓸 수 있도록 하는 기능
- 직접 입출력과의 차이는 CPU 레지스터를 이용하지 않고 CPU의 사이클을 훔쳐(cycle steal) 입출력을 수행한다는 점



【그림 2-3】 CPU cycle steal(가로채기)방식

- CPU와 DMA가 동시에 기억장치를 접근하는 경우,
  CPU는 기억장치 참조사이클을 중지하고 DMA가 우선적으로 기억장치 참조사이클을 수행하여 입출력을 수행
- DMA에서의 데이터 전송은 기억장치 버스를 통하여 수행하고 빠르게 데이터 전송을 수행

❖ DMA 방식 입출력 과정



【그림 2-4】 DMA 방식

- ① 프로그램 수행 도중 CPU가 입출력 명령을 만난다.
- ② CPU는 DMA에게 입출력 명령을 내린다.
- ③ CPU와 독립적으로 DMA는 기억장치의 데이터를 직접 입출력 한다.
- ④ CPU는 입출력 명령을 내린 후 다른 프로그램을 수행한다.
- ⑤ DMA는 입출력 완성 시 CPU에게 인터럽트로 완료 사실을 보고한다.

2) 입출력 채널 방식 (I/O channel)

- 가장 완벽한 입출력 방식
- 독립된 입출력 프로세서인 채널이 입출력을 대신하고,
  CPU는 다른 프로그램을 실행하여CPU의 효율을 향상시킬 수 있어
  다중 프로그래밍이 가능
- 입출력 프로세서는 채널이라 불리는 프로세서로써,
  그 자체가 하나의 CPU와 비슷한 성능을 가지는 보조 CPU
- CPU와 기억 장치를 공유하고 내부에 몇 개의 레지스터를 포함
- 대부분의 시스템은 여러 개의 채널이 있으며 주변 장치의 특성에 따라 주변 장치를 연결하는 방식도 다름

2-1) 입출력 채널 방식의 종류

①선택(selector) 채널

✓한 순간에 하나의 주변 장치만을 선택하여 연결하고 처리

✓주로 디스크나 드럼과 같이 고속인 장치들을 연결

② 멀티플렉서(multiplexer) 채널

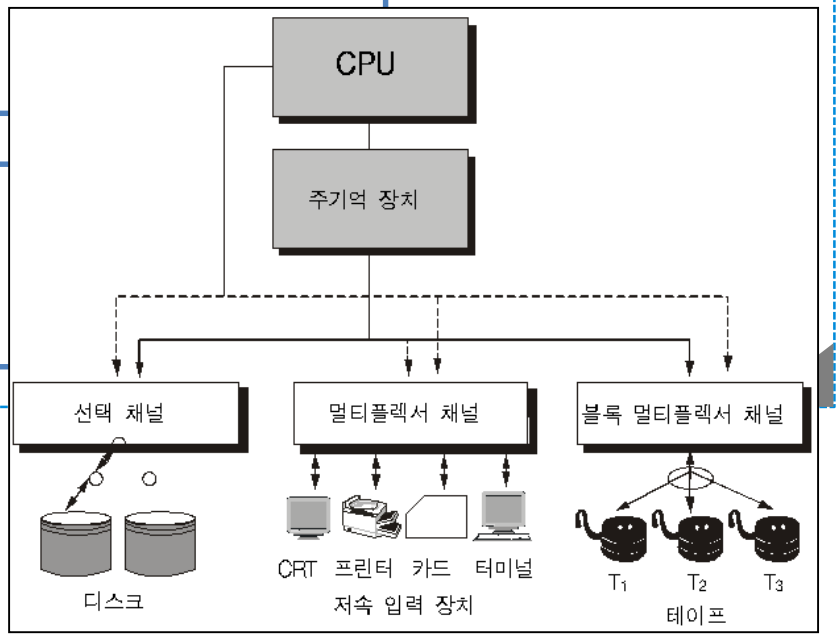
✓일시에 여러 장치들을 연결하여 처리

✓주로 터미널, 카드 판독기, 프린터 같은 저속의 장치들을 연결

③ 블록 멀티플렉서(block multiplexer) 채널

✓블록단위로 테이프와 같은 장치와 연결

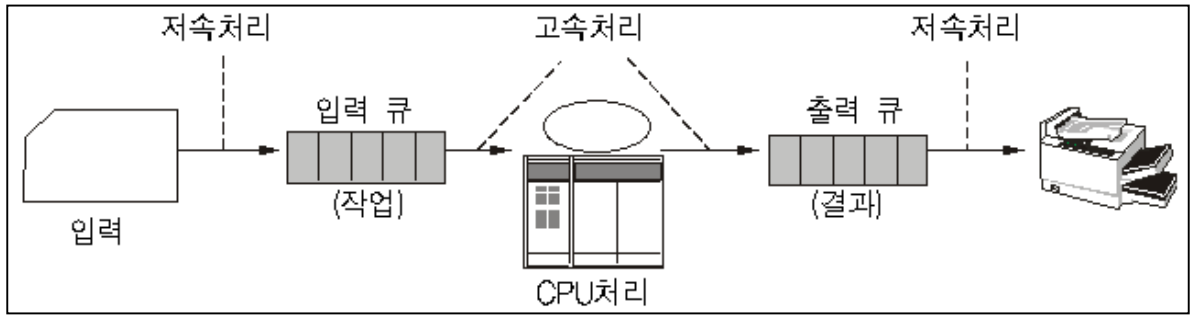
✓다수의 주변장치와 다중화하며 동시에 처리



【그림 2-5】 다양한 입출력장치와 채널 연결

(1) 버퍼링(buffering)

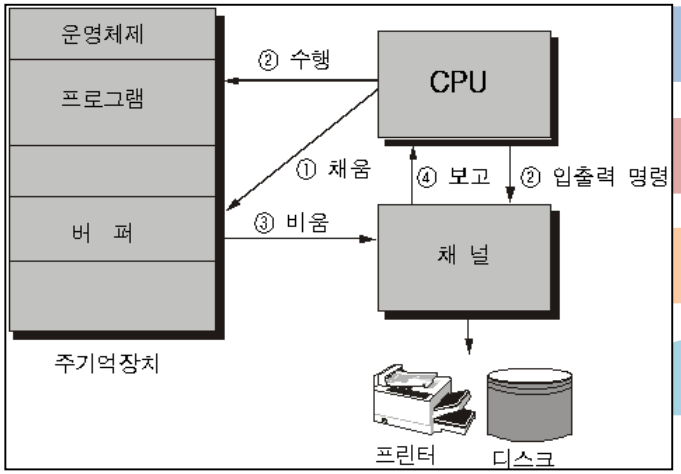
- CPU와 I/O 장치간의 속도차이를 극복할 수 있는 방법
- 주기억장치의 일부를 버퍼(임시 저장장소)로 사용함
- 한 작업의 연산과 함께 입출력을 동시에 수행하는 방법으로 데이터를 입력하고 CPU가 이를 연산하려고 하는 순간에 입력 장치가 즉시 다음 입력을 시작함
- CPU가 문자나 데이터 하나하나를 읽을 때마다 채널에게 명령을 수행한다면 매우 복잡하고 많은 작업량이 필요하다.
- 버퍼를 사용하여 문자나 데이터를 미리 읽어 온 후 필요할 때마다 사용함



【그림 2-6】 버퍼링의 구조

1) 단일 버퍼링(single buffering)

- 한 개의 버퍼를 사용하는 경우로 CPU는 채널이 버퍼를 채울 동안 기다리거나 다른 프로그램을 실행함
- CPU가 직접 입출력을 담당하지 않고 전용 입출력 프로세서인 DMA나 채널(channel)을 사용하는 방식

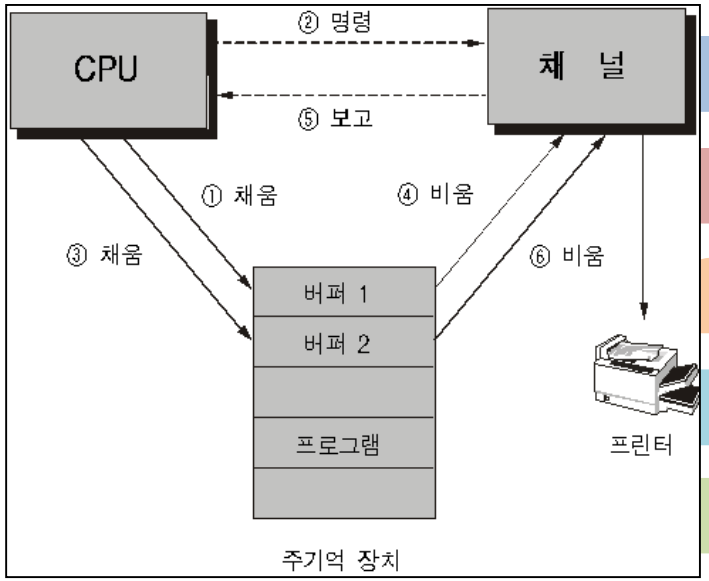


【그림 2-7】 단일 버퍼링의 구조

- ① CPU는 버퍼에 출력 데이터를 채운다. (채널은 버퍼가 채워질 때까지 대기)
- ② 버퍼가 채워지면 채널에게 명령하고, CPU 자신은 다른 일을 한다.
- ③ 채널은 CPU와 독립적으로 버퍼를 비운다.
- ④ 채널은 버퍼를 다 비운 후에 CPU에 보고한다.

2) 이중 버퍼링(dual buffering)

- 두 개의 버퍼를 이용하여 단일 버퍼링의 단점을 보완하고, 입출력 과 CPU의 처리 성능을 높이는 방법
- 입출력 작업과 처리 작업이 동시에 진행될 수 있으나 기억 장치의 낭비가 있을 수도 있다.
- CPU는 채널에게 하나의 버퍼에 대한 입출력을 명령하고, 채널은 CPU의 명령을 받고 독자적인 입출력을 수행하므로써 CPU는 더 이상 입출력에 관여하지 않으므로 CPU의 효율이 향상된다.



【그림 2-8】 이중 버퍼링의 구조

➔ ① CPU가 버퍼1을 채우고, 채널에게 입출력을 명령한다.

➔ ② 동시에 CPU는 즉시 버퍼 2를 채운다.

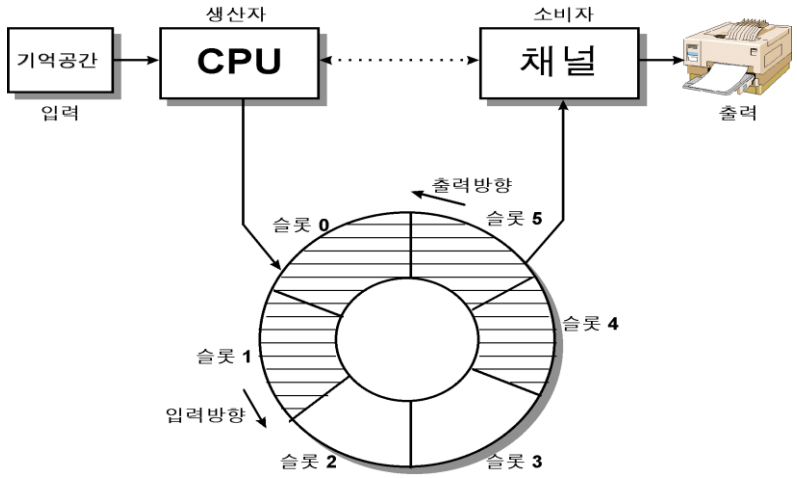
➔ ③ 채널은 버퍼 1을 비운다.

➔ ④ 채널은 버퍼 1을 다 비운 사실을 CPU에게 인터럽트를 걸어 보고한다.

➔ ⑤ ⑥ CPU는 버퍼 1을 채우고, 채널에 입출력을 명령하면 채널은 버퍼 2를 비운다

3) 환형 버퍼링(circular buffering)

- 환형 큐를 사용하여 여러 개의 버퍼를 원형으로 구성한 입출력 방식으로 CPU와 채널은 동시에 자신의 일을 독립적으로 수행한다.
- 환형 버퍼링은 여러 개의 버퍼를 사용하므로 사용되지 않을 경우에는 기억장치의 낭비를 초래하게 된다.



【그림 2-9】 환형 버퍼링의 구조

01 5.3 버퍼링과 스풀링

단일버퍼

- ✓ 버퍼가 하나
- ✓ CPU는 버퍼가 다 채워질 때까지 대기상태로 기다려야 함
- ✓ CPU의 효율성이 떨어짐

→ 메모리 공간을 적게 차지함.

→ 비효율적임

이중버퍼

- ✓ 버퍼를 채우고 비우는 작업이 중복되어 병렬로 수행
- ✓ 단일 버퍼보다 메모리 공간을 많이 차지함(버퍼가 2개임)

→ CPU의 대기 시간 감소함.

환형버퍼

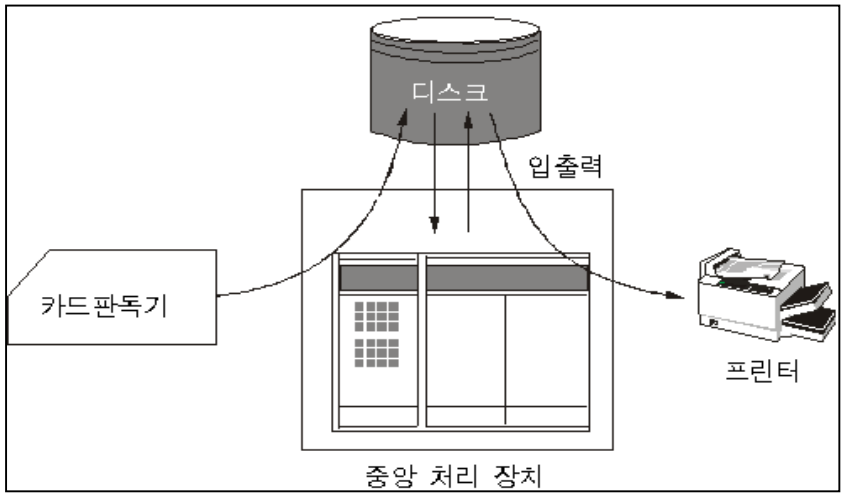
- ✓ 여러 개의 버퍼
- ✓ 기억장치 낭비를 초래함
- ✓ 버퍼가 많이 필요하지 않은 환경에서는 비효율적임.

→ 다양한 I/O 처리를 독립적으로 수행이 가능함

【표 2-2】 단일/이중/환형 버퍼링의 장단점 비교

(2) 스풀링(SPOOL : simultaneous peripheral operation on-line)

- 디스크의 일부를 스풀 공간(매우 큰 버퍼)처럼 사용하는 방식
- 입출력 장치와 CPU의 속도 차이를 해소하여, 미리 입력 장치에서 블록을 읽어 들여 디스크 상에 출력 파일로 저장한 후 나중에 출력할 수 있도록 한다.
- 디스크를 스풀 공간으로 사용하는 이유는 입출력 장치와 CPU의 속도 차이를 최소화하기 위함이다. 즉 미리 입력 장치에서 블록을 읽어 디스크 상에 출력 파일로 저장한 후 나중에 출력할 수 있도록 한다.



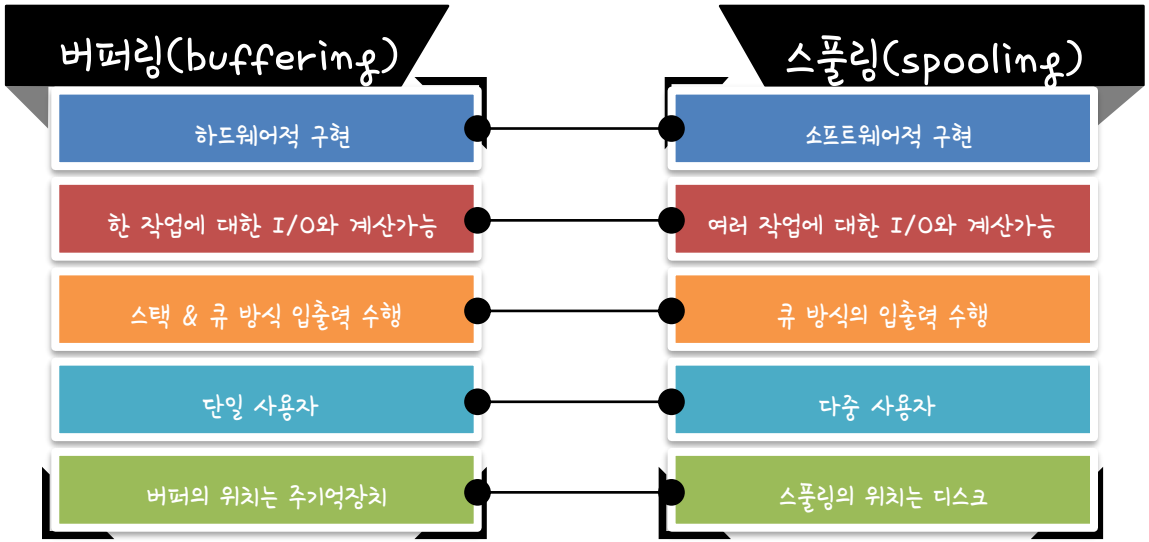
【그림 2-10】 스풀링의 구조

버퍼링과 스풀링의 차이점

✓ 버퍼링은 한 작업의 연산 뿐만 아니라 입출력을 동시에 수행

✓ 스풀링은 한 작업의 입출력 뿐만 아니라 다른 작업의 연산을 중복할 수 있다. 간단한 시스템에서도 스풀러는 한 작업의 출력을 인쇄하는 동안 동시에 다른 작업의 입력을 수행할 수 있다.

【표 2-3】 버퍼링과 스풀링의 차이점 비교



CONTENTS

입출력 제어 및 인터럽트

02 6.1 인터럽트 정의 및 처리 과정

- 정상적인 프로그램을 수행하는 컴퓨터 시스템에 예기치 않은 상황이 발생할 경우 수행중인 프로그램을 중단하거나 비정상적인 종료를 하고 운영체제에 의해 적절한 조치를 취할 수 있도록 하는 기법
- 인터럽트의 예
  - ✓ 전화를 받는 있는 도중, 초인종을 누른다면 일단 전화를 끊고 손님을 맞이한 다음, 다시 전화를 걸어서 통화하는 경우 : 초인종을 누르는 행위가 전화를 일시 중지시키는 인터럽트가 됨
  - ✓ 프린팅 하는 도중에 더 이상 자료가 필요하지 않은 경우 스위치를 끄는 경우

【그림 2-11】 일상생활 속에서의 인터럽트 예

02 6.1 인터럽트 정의 및 처리 과정

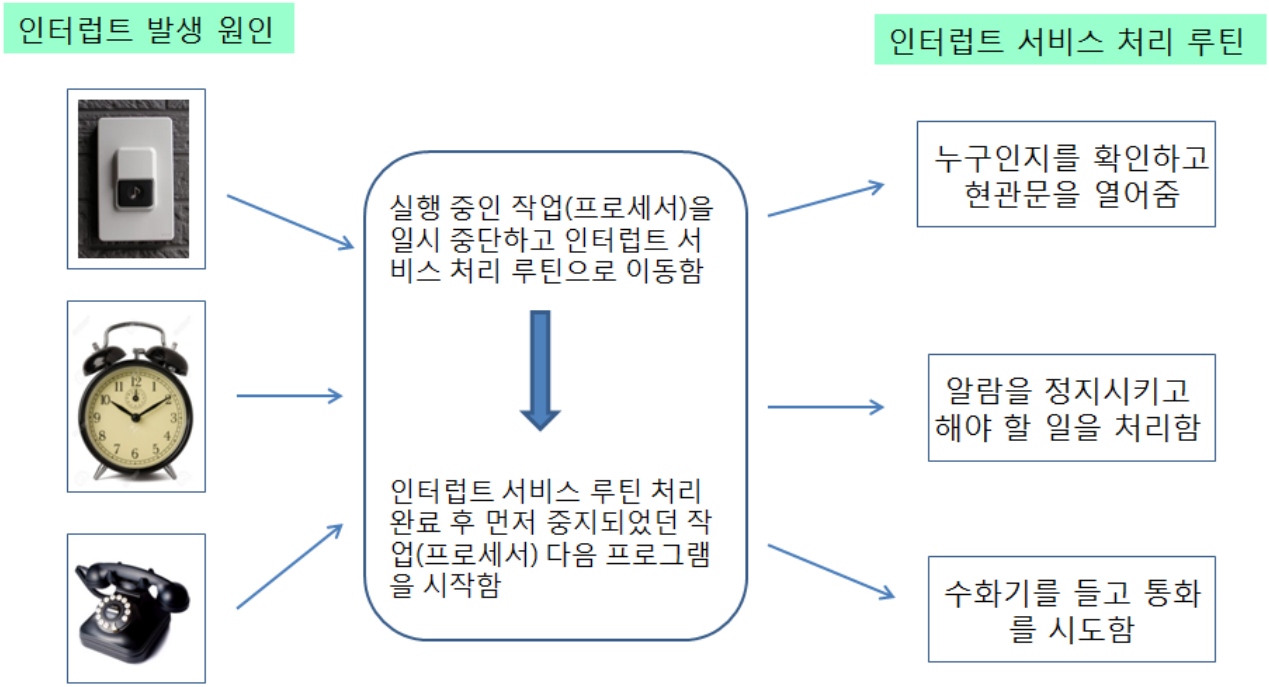
[인터럽트 처리 과정]

- ✓ 인터럽트가 발생하게 되면 운영체제가 제어권을 가지게 된다
- ✓ 제어권이 제어 프로그램에게 주어지면 제어 프로그램(control program)의 인터럽트 처리루틴(interrupt handing routine)이 이를 처리하게 된다
- ✓ 인터럽트 처리 루틴에서 처리가 끝나면, 운영체제 관리자에게 인터럽트 처리 완료를 알리게 됨
- ✓ 시스템은 인터럽트가 발생하기 이전의 상태로 돌아가서 먼저 실행되던 처리 프로그램(processing program)의 실행을 다시 시작하게 된다

(1) 인터럽트의 발생원인

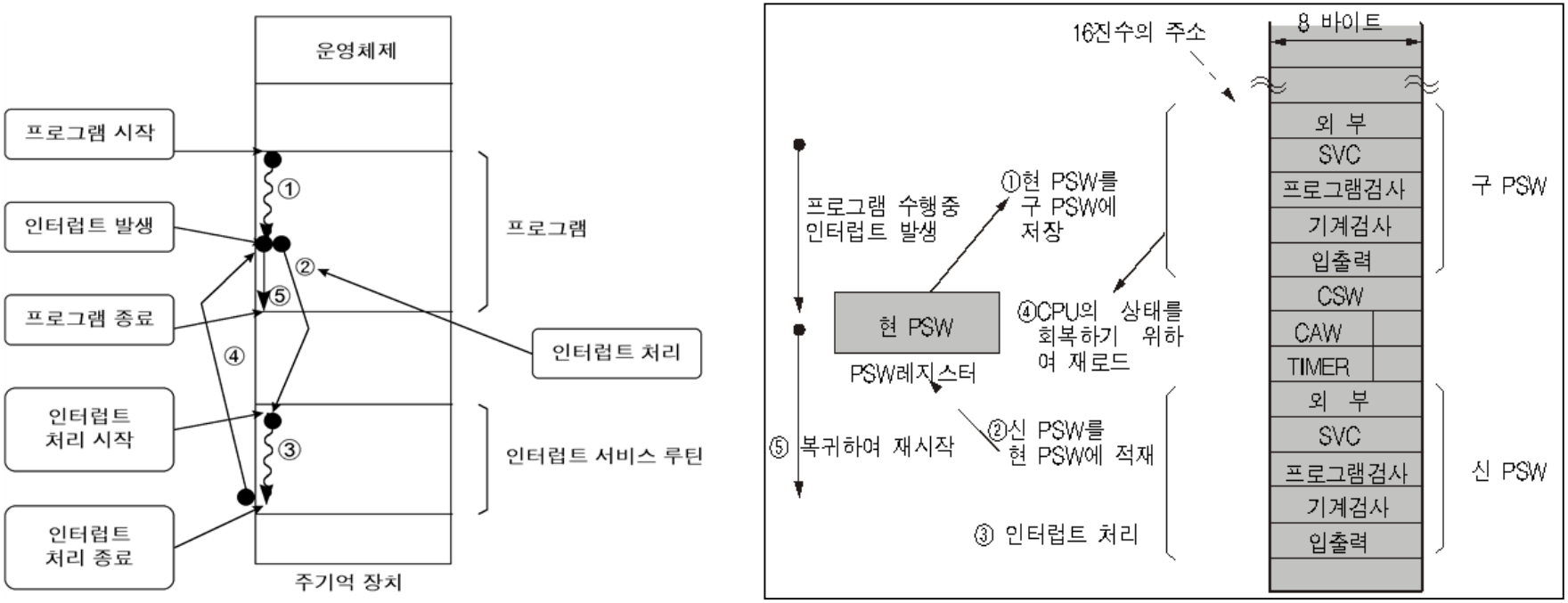
- ❖ 프로그램 수행 중에 컴퓨터 내부 및 외부에서 예상치 못한 상태 발생.
- ❖ 정전 혹은 데이터 전송 과정에서 오류 발생 등 컴퓨터 자체의 기계적인 문제 발생.
- ❖ 보호된 기억 공간에 접근 혹은 불법적인 명령어의 수행 등과 같은 프로그램상의 문제 발생.
- ❖ 컴퓨터 조작자가 의도적으로 조작에 의해 중단시키는 경우.
- ❖ 입출력 장치들의 동작에 CPU의 기능이 요청되는 경우.

02 6.1 인터럽트 정의 및 처리 과정



【그림 2-12】인터럽트 발생원인 및 인터럽트 처리 과정 예

(2) 인터럽트 처리 과정



【그림 2-13】인터럽트 처리 과정

02 6.1 인터럽트 정의 및 처리 과정

① 프로세서 수행 도중에 인터럽트 발생 장치로부터 인터럽트가 발생 된다

② 중앙처리장치가 인터럽트 요청을 받으면,
현재 수행 중인 프로그램의 수행에 관련된 H/W 및 S/W 상태를 보존
(현 PSW → 구 PSW)

③ 인터럽트 처리 루틴에서 인터럽트의 원인을 찾고,
이에 대한 인터럽트 서비스 루틴을 수행(인터럽트 처리)

④ 인터럽트 서비스 루틴의 수행을 통하여 실질적인 인터럽트에 대한 조치가 완료되면 인터럽트 처리를 종료(인터럽트 처리 완료)

⑤ 다시 인터럽트 처리 기능을 이용해서 미리 보존한 프로그램의 상태를 복구하고 인터럽트 당한 프로그램을 중단된 곳에서부터 계속할 수 있도록 조치(구 PSW → 신 PSW)

⑥ 중앙처리장치는 조금 전 발생하였던 인터럽트 다음의 프로그램을 수행(복귀하여 재시작함)

⑦ 인터럽트 처리 루틴이란 인터럽트 후에 인터럽트를 위한 주소로 분기하기까지의 처리 프로그램 루틴이며, 인터럽트 요청에 의해서 수행되어야 하는 프로그램 루틴이다.

02 6.1 인터럽트 정의 및 처리 과정

- 운영체제는 인터럽트가 발생하는 원인에 따라 적절한 인터럽트 처리 루틴 에 의해 최소한의 조치를 취하고 인터럽트 서비스 루틴에 의해 발생한 상황을 해결한다.
- 운영체제의 커널은 인터럽트 발생 신호를 감지하면 프로그램 카운터 (PC) 의 내용과 CPU에 의해 수행한 현재까지의 모든 프로그램 상태 정보를 저장한 후에(현 PSW → 구 PSW) 인터럽트 서비스 루틴을 수행시키고 인터럽트 처리가 완료 되면 발생하기 이전의 상태로 복귀하여 중단되었던 원래의 프로그램 수행을 재개한다.

문맥 교환(context switching)

- ✓ 인터럽트가 발생 할 때 운영체제가 인터럽트를 당한 실행중인 프로그램의 상태를 저장한 후 제어권을 인터럽트 서비스 루틴으로 넘겨주는 작업
- ✓ 문맥 교환의 과정에서 PSW(program status word)는 명령문 수행의 순서를 조절하며, 실행 중이던 프로그램의 상태에 대한 여러 가지 정보를 보관하고 있다.

(1) 인터럽트 종류

- 기계 착오 인터럽트(machine check interrupt)
- 프로그램 검사 인터럽트(program check interrupt)
- 재시작 인터럽트(restart interrupt)
- 입출력 인터럽트(I/O interrupt) : 내부(internal) 인터럽트
- 외부 인터럽트(external interrupt)
- 감시자(제어자) 호출 인터럽트(SVC, supervisor call interrupt)

기계 착오 인터럽트(machine check interrupt)

- ✓ 프로그램을 수행하는 도중에 기계의 착오로 인하여 생기는 인터럽트
- ✓ 기계에 이상이 생겨 발생하는 인터럽트

재시작 인터럽트(restart interrupt)

- ✓ 조작자(operator)가 재시작 버튼을 누른 경우
- ✓ 다중처리 시스템에서 다른 프로세서로부터 재시작 명령문이 도착한 경우

외부 인터럽트(external interrupt)

- ✓ 외부의 신호에 따라 발생하는 인터럽트
- ✓ 조작자가 시스템의 요구에 따라 인터럽트 키를 누른 경우
- ✓ 타이머에 의해 일정한 시간이 경과한 경우

02 6.2 대표적인 인터럽트 종류

입출력 인터럽트(I/O interrupt) : 내부(internal) 인터럽트

- ✓ 입출력 요구가 발생하였을 때 발생하는 인터럽트
- ✓ CPU에서 채널이나 입출력장치의 변화를 알리기 위해 발생
- ✓ 입출력 종료 및 입출력 오류에 의해 생기는 인터럽트

프로그램 검사 인터럽트(program check interrupt)

- ✓ 프로그램 명령법이나 지정법에 잘못이 있을 때 발생하는 인터럽트
- ✓ divide by zero(0)로 나누었을 경우 발생
- ✓ 프로그램상의 착오나 예외 상황이 발생하였을 경우

감시자(제어자) 호출 인터럽트(SVC, supervisor call interrupt)

- ✓ 사용자가 프로그램에서 SVC 명령을 호출 하였을 경우
- ✓ I/O 수행 및 기억장치 할당 등의 역할을 하는 인터럽트
- ✓ SVC call 명령을 호출하였을 때 발생
- ✓ 조작자(operator)와의 대화를 위해 발생

02 6.2 대표적인 인터럽트 종류

【표 2-4】 인터럽트 종류와 의미

종 류	의미
입출력 인터럽트	해당 입출력 하드웨어가 주어진 입출력 동작을 완료하였거나, 입출력 도중 오류 등이 발생하였을 경우 CPU에 대하여 요청하는 인터럽트이다.
외부 인터럽트	CPU의 하드웨어 신호에 의해서 발생하고 프로그램의 외적인 상황에서 일어나며, 프로그램과 비동기적인 인터럽트로서, 입출력 장치, 타이밍 장치, 전원 등의 요인으로 발생하는 것을 말한다.
슈퍼바이저 호출 인터럽트	프로그램 명령어 상에서 신호가 발생하고, 명령어의 수행에 의해 발생하는 인터럽트로서 특수한 호출 명령으로 프로그래머에 의해 프로그램 상의 원하는 위치에서 인터럽트를 일으키게 된다.
프로그램 검사 인터럽트	프로그램 상의 불법적인 명령이나, 데이터의 잘못 사용으로 발생하는 인터럽트로 흔히 트랩(trap)이라고 한다.
기계 검사 인터럽트	컴퓨터 자체 내의 기계적인 장애나 에러로 인하여 발생한 오류이다. 즉 CPU, 메모리 및 각종 하드웨어 고장 시 발생한다.
재시작 인터럽트	조작자(operator)가 콘솔 상에서 재시작(RESET) 버튼이나, CTRL+ALT+DEL 키를 누른 경우에 발생한다.

(2) 인터럽트 처리 우선순위

❖ 여러 장치에서 동시에 하나 이상의 인터럽트가 발생했을 때 먼저 서비스할 장치를 결정하기 위해서 필요하다.

❖ 여러 장치에서 여러 가지의 인터럽트가 동시에 시스템 내에서 발생하였을 때 운영체제 관리자는 다음과 같은 우선순위로 인터럽트를 처리하게 된다.

02 6.2 대표적인 인터럽트 종류

** 인터럽트 처리 우선 순위

【그림 2-14】 인터럽트 처리 우선순위

Q & A

열공합시다!!