제7장 시스템 버스, I/O 및 인터럽트

7.1 시스템 버스

7.2 버스 중재

7.3 I/O 장치의 접속

7.4 인터럽트를 이용한 I/O

7.5 DMA를 이용한 I/O

7.1 시스템 버스 (system bus)

- 컴퓨터시스템의 구성 요소들(CPU, 기억장치, I/O 장치들)을 상호 연결해주는 중심 통로

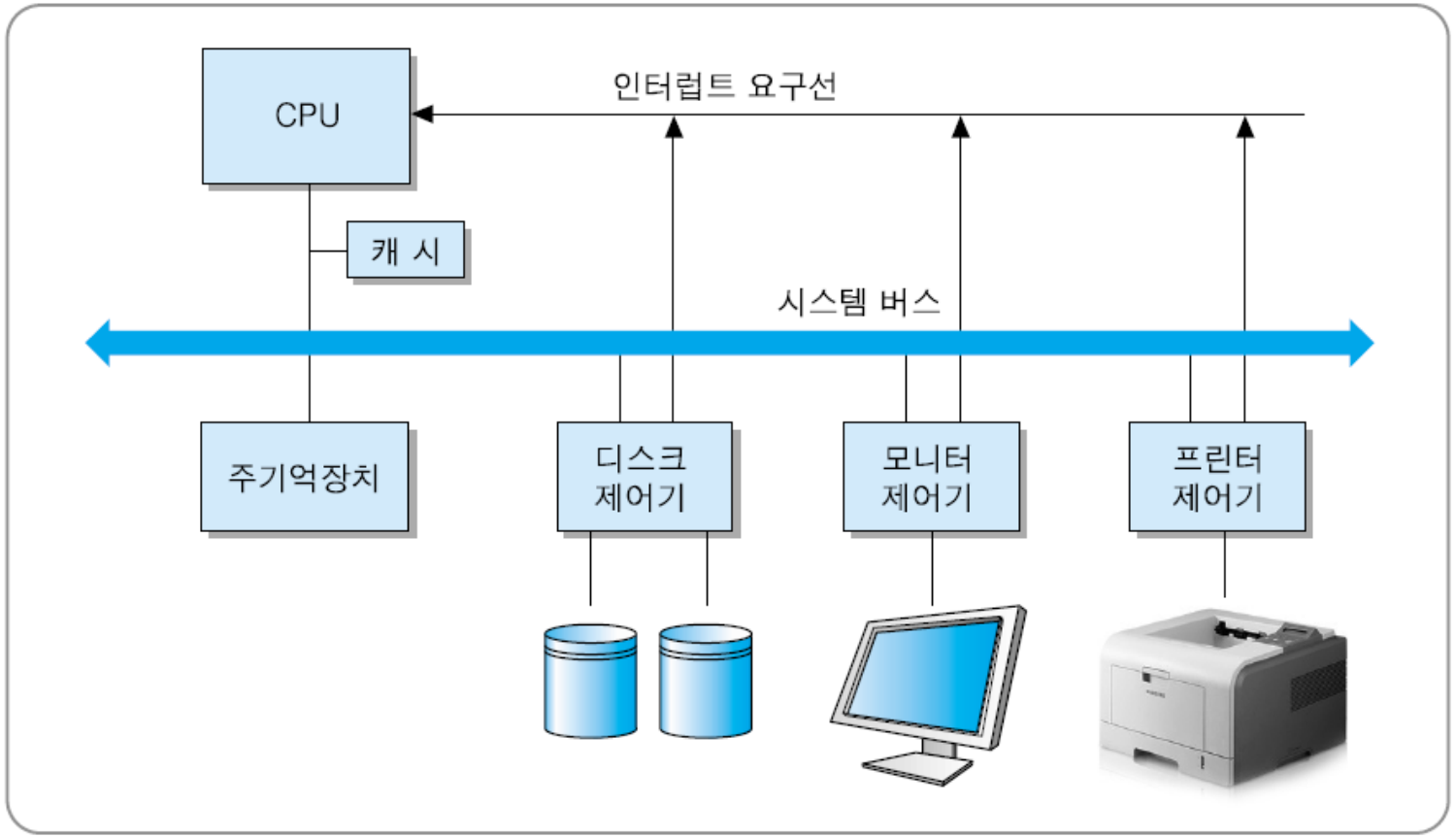


그림 7-1 시스템 버스를 이용한 컴퓨터시스템의 구성도

7.1.1 시스템 버스의 조직

- 버스(bus) : 구성 요소들간에 교환할 각종 정보들을 전송하는 선(line) 들로 구성
- 버스 선의 수
  - 한 번에 전송하는 데이터 비트들의 수, 기억장치 주소 비트들의 수 및 제어 신호들의 수에 따라 결정됨
  - 소형 컴퓨터 : 50 ~ 100 개의 신호선들 사용
  - 중대형급 시스템 : 100 개 이상의 신호선들 사용

기능에 따른 버스의 종류

- 데이터 버스(data bus)
  - 시스템 요소들 사이에 데이터를 전송하는 데 사용되는 선들의 집합
  - 양방향 전송(bidirectional transfer)
  - 버스 폭(선들의 수) = CPU와 기억장치 사이에 한 번에 전송되는 비트 수
- 주소 버스(address bus)
  - CPU가 기억장치로 (또는 기억장치로부터) 데이터 쓰기 (또는 읽기) 동작을 할 때, 해당 기억장소를 지정하는 주소를 전송하기 위한 선들의 집합
  - 단방향 전송(unidirectional transfer) : CPU → 기억장치 및 I/O 제어기
  - 주소 버스의 비트 수에 의해 시스템에 접속될 수 있는 전체 기억장치 용량이 결정됨
  - 직접 주소지정 할 수 있는 기억장소의 단위 : 바이트(byte) 혹은 단어(word)
- 제어 버스(control bus)
  - CPU와 기억장치 및 I/O 장치 사이에 제어 신호들을 전송하는 선들의 집합

주소 버스의 폭(비트 수)에 따른 기억장치 용량 예

- 주소 버스 = 16비트 인 경우
  - 주소지정 가능한 최대 기억장소들의 수 = 2^{16} = 65,536(64K) 개
  - 바이트 단위 주소지정일 경우 최대 기억장치 용량 = 64 Kbyte
  - 32-비트 단어 단위 주소지정일 경우, 최대 기억장치 용량 = 256 Kbyte
- 주소 버스 = 30비트 인 경우
  - 최대 2^{30} = 1G (≈ 10억) 개
  - 바이트 단위 주소지정일 경우 최대 기억장치 용량 = 1 Gbyte
  - 32-비트 단어 단위 주소지정일 경우,
    최대 기억장치 용량 = 4 Gbyte

제어 버스

- 기억장치 및 I/O 장치와의 데이터 교환을 위한 제어신호들
  - 기억장치 쓰기(memory write) 신호 : 버스에 실린 데이터를 주소가 지정하는 기억장소에 저장되도록 하는 제어 신호
  - 기억장치 읽기(memory read) 신호 : 주소가 지정하는 기억장소의 내용을 읽어서 버스에 실리게 하는 제어 신호
  - I/O 쓰기(I/O write) 신호 : 버스에 실린 데이터를 지정된 I/O 장치로 출력되게 하는 제어 신호
  - I/O 읽기(I/O read) 신호 : 지정된 I/O 장치로부터 데이터를 읽어서 데이터 버스에 실리게 하는 제어 신호
- 버스 중재를 위한 제어신호들(이 신호들의 집합을 중재 버스라고도 부름)
- 인터럽트 메커니즘을 위한 제어신호들(이 신호들의 집합을 인터럽트 버스라고도 부름)

중재 버스(arbitration bus)

- 버스 마스터(bus master): 시스템 버스에 접속되는 요소들 중에서 버스 사용의 주체가 되는 요소들 (예: CPU, 기억장치 모듈, I/O 제어기, 등)
- 버스 중재 (bus arbitration): 시스템 버스에 접속된 두 개 또는 그 이상의 버스 마스터들이 동시에 버스를 사용하고자 할 때 순서대로 한 개의 마스터씩 버스를 사용할 수 있게 해주는 동작
- 중재 버스 : 버스 중재를 위한 신호 선들의 집합
  - 버스 요구(bus request) 신호: 버스 마스터가 버스 사용을 요구했음을 알리는 신호
  - 버스 승인(bus grant) 신호 : 버스 사용을 요구한 마스터에게 사용을 허가하는 신호
  - 버스 사용중(bus busy) 신호 : 현재 버스가 사용되고 있는 중임을 나타내는 신호

인터럽트 버스(interrupt bus)

- 인터럽트 메커니즘을 위한 제어 신호선들의 집합
  - 인터럽트 요구(interrupt request) 신호 : I/O 장치가 인터럽트를 요구했음을 알리는 신호
  - 인터럽트 확인(interrupt acknowledge) 신호 : CPU가 인터럽트 요구를 인식했음을 알리는 신호
- 그 이외의 제어신호들
  - 버스 클록(bus clock) 신호 : 동기식 버스에서 버스 동작들의 시작 시간을 일치시키기 위하여 제공되는 공통 클록 신호
  - 리셋(reset) 신호 : 모든 시스템 요소들의 동작을 초기화시키는 신호

버스 대역폭(bus bandwidth)

- 버스의 속도를 나타내는 척도로서, 단위 시간당 전송할 수 있는 데이터 양을 나타내며, 버스 클록의 주기에 의해 결정

[예] 버스 클록의 주기: **50 *ns*** (클럭 주파수: **20 MHz**)

데이터 버스의 폭: **64** 비트(**8** 바이트)

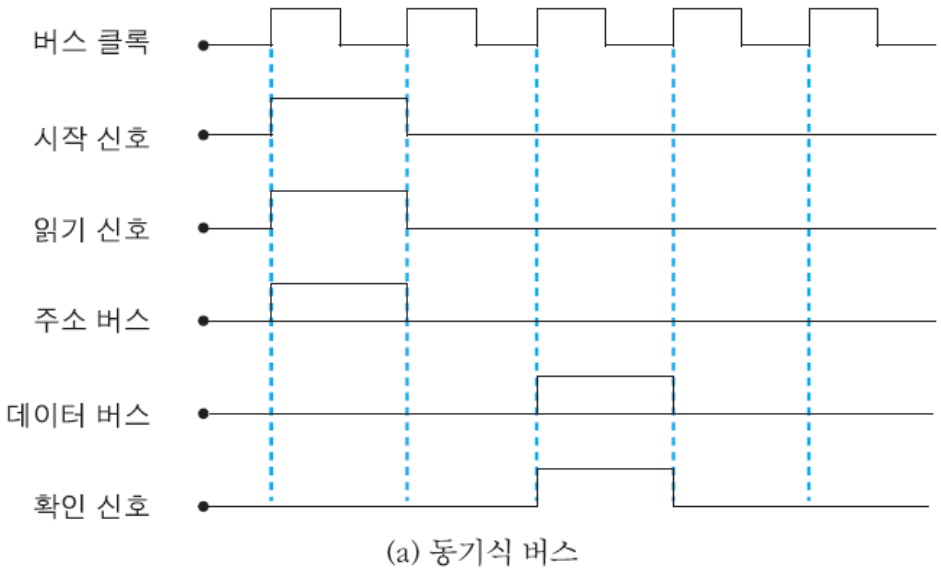
➔ 버스 대역폭 = **8 byte / (50 × 10^{-9} sec) = 160 [Mbytes/sec]**

(즉, 이 버스를 통하여 초당 **1**억 **6**천 바이트의 데이터 전송 가능)

7.1.2 시스템 버스의 기본 동작

- 쓰기 동작(write operation) 순서
  - ① 버스 마스터가 버스 사용권 획득
  - ② 버스를 통하여 주소와 데이터 및 쓰기 신호 전송
- 읽기 동작(read operation) 순서
  - ① 버스 마스터가 버스 사용권 획득
  - ② 주소와 읽기 신호를 보내고, 데이터가 전송되어 올 때까지 대기
- 버스 동작의 타이밍에 따른 버스의 분류
  - 동기식 버스(synchronous bus) : 시스템 버스에서 모든 버스 동작들이 공통의 버스 클럭을 기준으로 발생
  - 비동기식 버스(asynchronous bus) : 버스 동작들의 발생 시간이 관련된 다른 버스 동작의 발생 여부에 따라 결정

동기식 버스와 비동기식 버스의 시간 흐름도



동기식 / 비동기식 버스의 비교

- 동기식 버스

  [장점] 인터페이스 회로 간단

  [단점] 버스 클럭의 주기가 가장 오래 걸리는 버스 동작의 소요 시간을 기준으로 결정되므로, 클럭 주기보다 더 짧은 시간이 걸리는 버스 동작의 경우에는 동작이 완료된 후에도 다음 주기가 시작될 때까지 대기

- 비동기식 버스

  [장점] 각 버스 동작이 완료 즉시 연관된 다음 동작이 발생하기 때문에 동기식 버스에서와 같이 낭비되는 시간이 없음

  [단점] 연속적 동작을 처리하기 위한 인터페이스 회로가 복잡

- 소규모 컴퓨터 : 비동기식 버스 사용

  일반적인 컴퓨터시스템 : 동기식 버스 사용

7.2 버스 중재 (bus arbitration)

- 버스 경합(bus contention) : 한 개의 시스템 버스에 접속된 여러 개의 버스 마스터들이 동시에 버스 사용을 요구하는 현상
- 버스 중재(bus arbitration) : 버스 경합이 발생하는 경우, 어떤 기준에 따라 버스 마스터들 중에서 한 개씩만 선택하여 순서대로 버스를 사용할 수 있게 해주는 동작
- 버스 중재기(bus arbiter) : 버스를 중재하는 하드웨어 모듈

버스 중재 방식의 분류

- 제어 신호들의 연결 구조에 따른 중재 방식의 분류
  - 병렬 중재 방식(parallel arbitration scheme)
    - 각 버스 마스터들이 독립적인 버스 요구 신호를 발생하고, 별도의 버스 승인 신호를 받음
      → 버스 마스터들의 수와 같은 개수의 버스 요구 선 및 승인 신호 선 필요
  - 직렬 중재 방식(serial arbitration scheme)
    - 버스 요구와 승인 신호 선이 각각 한 개씩만 존재하며, 각 신호 선을 버스 마스터들 간에 직렬로 접속하는 방식

버스 중재 방식의 분류

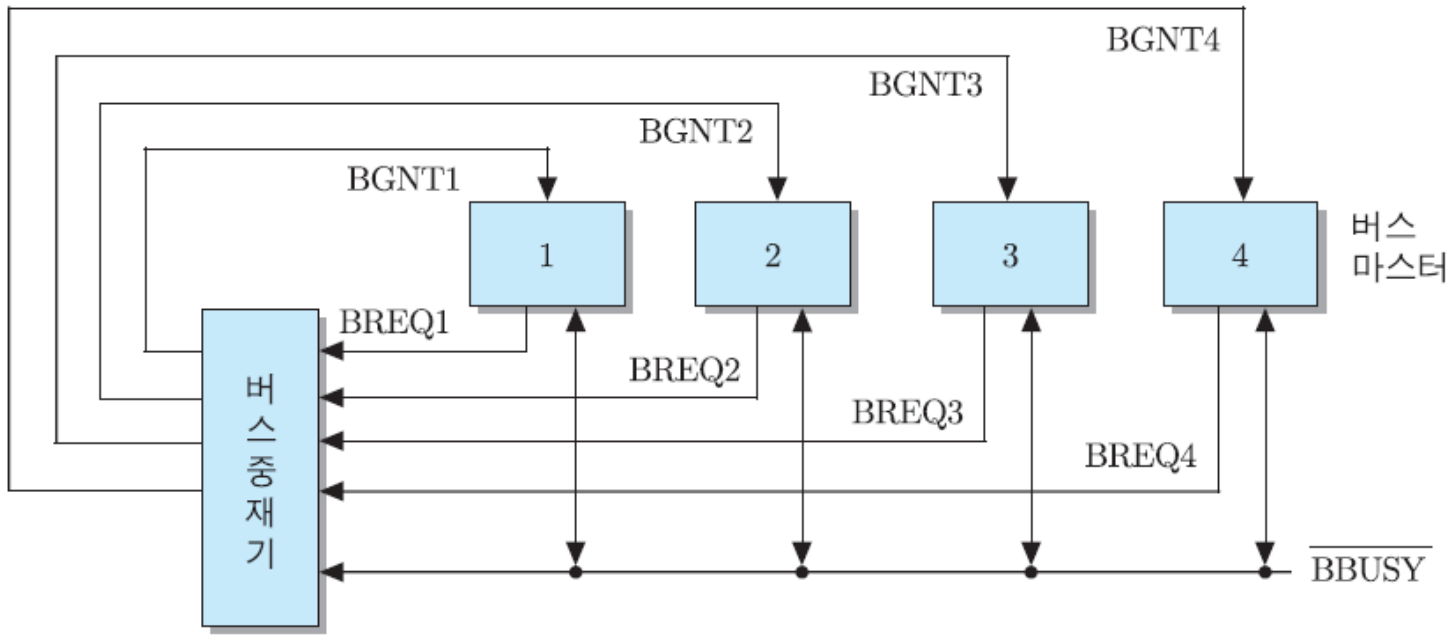
- 버스 중재기의 위치에 따른 분류
  - 중앙집중식 중재 방식(centralized arbitration scheme)
    - 시스템 내에 버스 중재기가 한 개만 존재하는 방식
    - 버스 마스터들이 발생하는 버스 요구 신호들은 하나의 중재기로 보내지고, 중재기는 정해진 중재 원칙에 따라 선택한 버스 마스터에게 승인 신호를 발생
  - 분산식 중재 방식(decentralized arbitration scheme)
    - 여러 개의 버스 중재기들이 존재하며(일반적으로 각 버스 마스터가 중재기를 한 개씩 가짐), 버스 중재 동작이 각 마스터의 중재기에 의하여 이루어지는 방식

7.2.1 병렬 중재 방식

- 우선순위의 결정 방식에 따른 분류
  - 고정-우선순위 방식(fixed-priority scheme) : 각 버스 마스터에 지정된 우선순위가 고정되어 있는 방식
  - 가변-우선순위 방식(dynamic-priority scheme) : 우선순위를 변경할 수 있는 방식

1) 중앙집중식 고정-우선순위 중재방식

- 모든 버스 마스터들이 하나의 버스 중재기에 접속
- 중재기와 가장 가까이 위치한 버스 마스터 1이 가장 높은 우선순위, 버스 마스터 4가 가장 낮은 우선순위를 가지는 것으로 가정



중앙집중식 고정 우선순위 중재 방식 (계속)

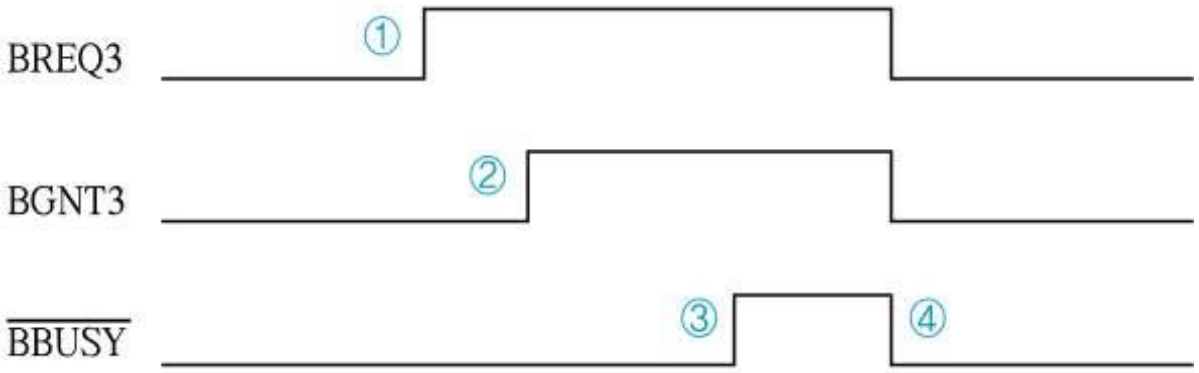
[예] '버스마스터 1'이 버스를 사용중일 때, '버스마스터 3'이 버스 사용을 요구한 경우:

① '마스터 3'이 BREQ3 신호를 세트

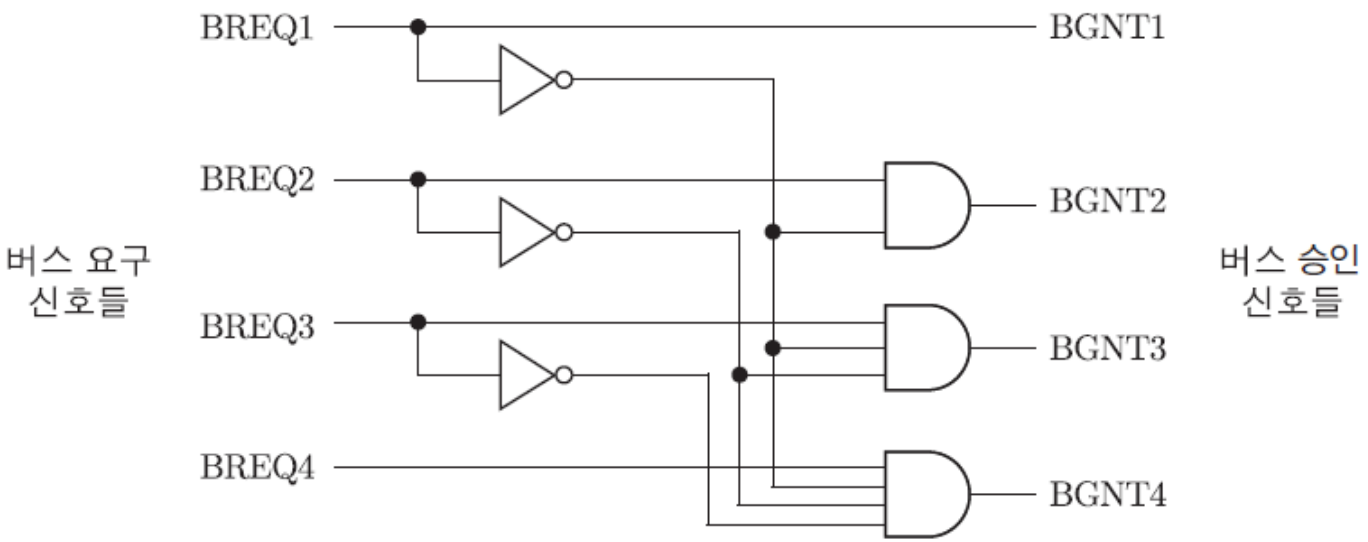
② 버스 중재기가 '마스터 3'에게 BGNT3 신호를 세트 함으로써 버스 사용을 허가

③ '마스터 1'이 버스 사용을 끝내고 BBUSY 신호를 해제

④ '마스터 3'이 BBUSY 신호를 다시 세트하고, 버스 사용을 시작 (이때 BREQ3와 BGNT3는 제거)



병렬 중재기의 내부 회로도



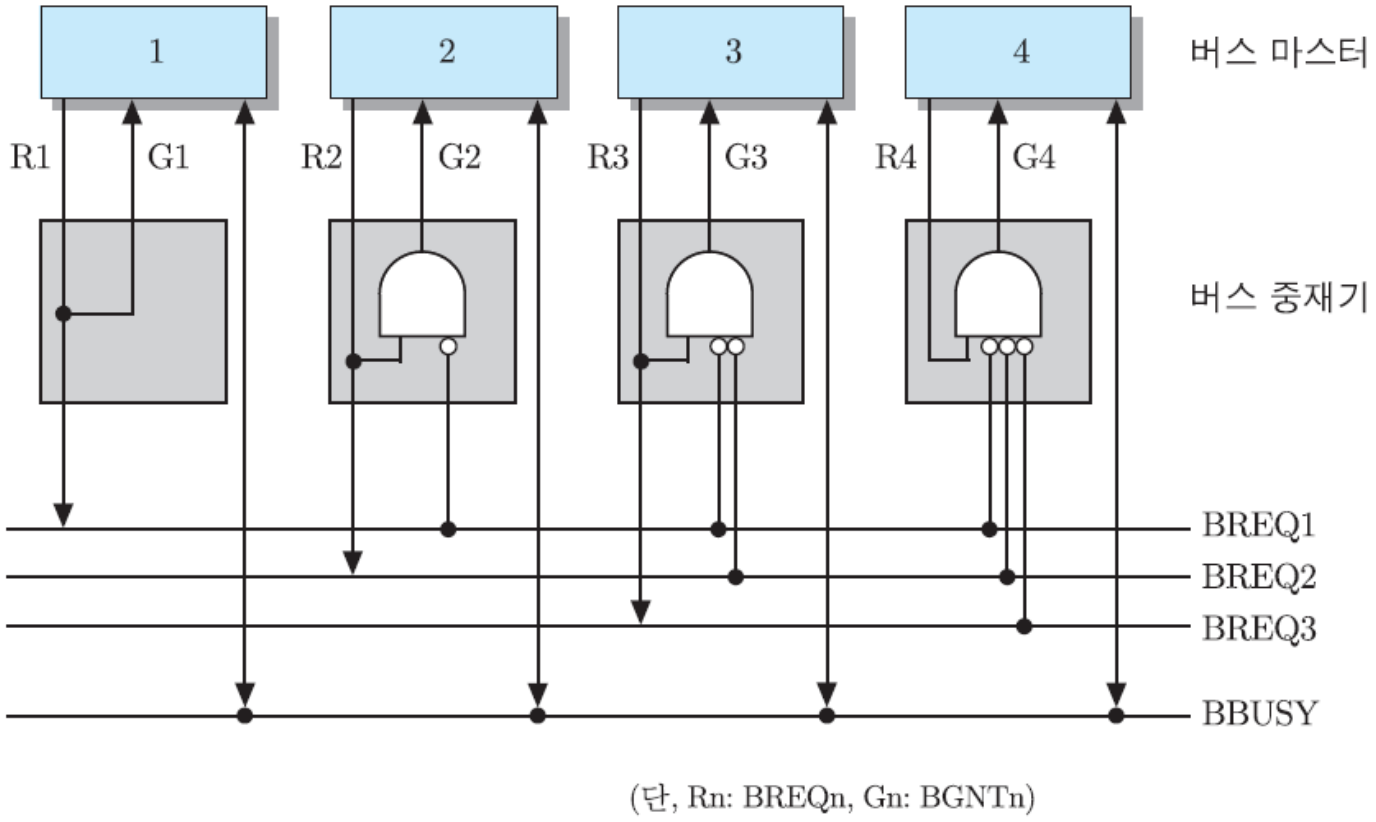
2) 분산식 고정-우선순위 방식

- 모든 버스 마스터들이 중재기를 한 개씩 보유
- 중재 동작
  - 각 중재기는 자신보다 더 높은 우선순위를 가진 마스터들의 버스 요구 신호들을 받아서 검사하여, 그들이 버스 사용 요구를 하지 않은 경우에만 자신의 버스 마스터로 버스 승인 신호 발생
  - 승인 신호를 받은 버스 마스터는 BBUSY 신호를 검사하여, 비활성화 상태(다른 마스터가 버스를 사용하지 않는 상태)일 때 버스 사용을 시작
- 분산식 중재 방식의 장단점

[장점] 중앙집중식에 비하여 중재 회로가 간단하므로 동작 속도가 빨라진다

[단점] 고장을 일으킨 중재기를 찾아내는 방법이 복잡하고, 한 중재기의 고장이 전체 시스템의 동작에 영향을 미칠 수가 있다

분산식 고정-우선순위 방식의 구성도



3) 가변 우선순위 방식

- 시스템의 상태(또는 조건)에 따라 각 버스 마스터들의 우선순위를 계속 변화시키는 방식

  [단점] 중재 회로 복잡

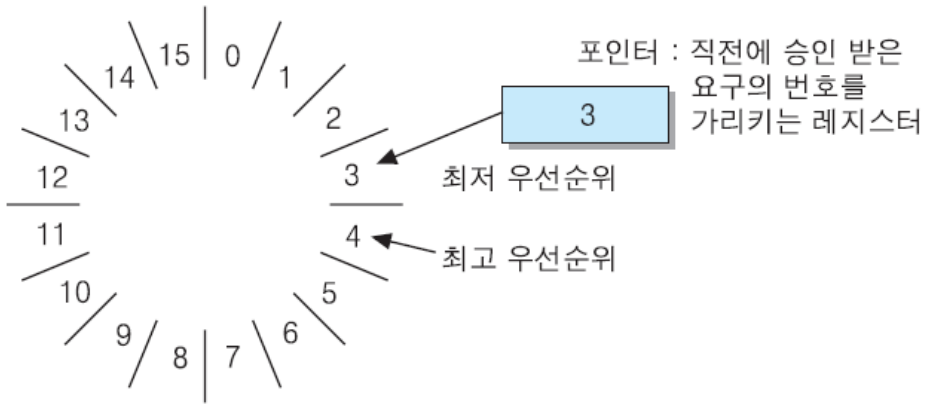
  [장점] 모든 마스터들이 공정하게 버스를 사용할 수 있게 해준다

- 회전 우선순위(rotating priority) 방식

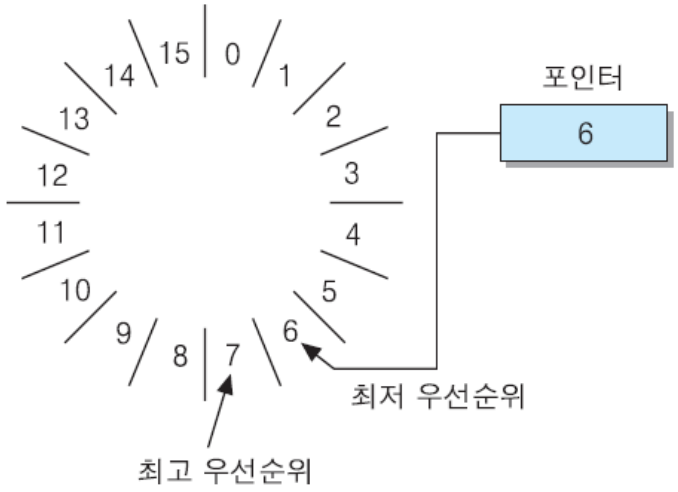
  [방법1] 중재 동작이 끝날 때마다 모든 마스터들의 우선순위가 한 단계씩 낮아지고, 가장 우선순위가 낮았던 마스터가 최상위 우선순위를 가지도록 하는 방법

  [방법2] 일단 버스 사용 승인을 받은 마스터는 최하위 우선순위를 가지며, 바로 다음에 위치한 마스터가 최상위 우선순위를 가지도록 하는 방법 → Acceptance-dependent식 회전 우선순위 방식

Acceptance-dependent식 회전 우선순위 방식의 예



(a) 마스터 3의 요구가 승인된 후



(b) 마스터 6의 요구가 승인된 후

가변 우선순위 방식 (계속)

- 동등 우선순위 방식 : 모든 마스터들이 동등한 우선순위를 가지며, FIFO(First-In First-Out) 알고리즘 사용
- 임의 우선순위 방식 : 각 중재 동작이 끝날 때마다 우선순위를 임의로 결정
- 최소-최근 사용(Least-Recently Used: LRU) 방식 : 최근 가장 오랫동안 버스를 사용하지 않은 버스 마스터에게 최상위 우선순위 할당

  [단점] 회로가 매우 복잡

7.2.2 직렬 중재 방식

1) 중앙집중식 직렬 중재 방식

- 하나의 버스 사용승인 신호선(BGNT)이 데이지-체인(daisy-chain) 형태로 모든 버스 마스터들을 직렬로 연결
- 우선순위는 버스 승인 신호선이 연결된 순서대로 결정

❑ 동작 원리

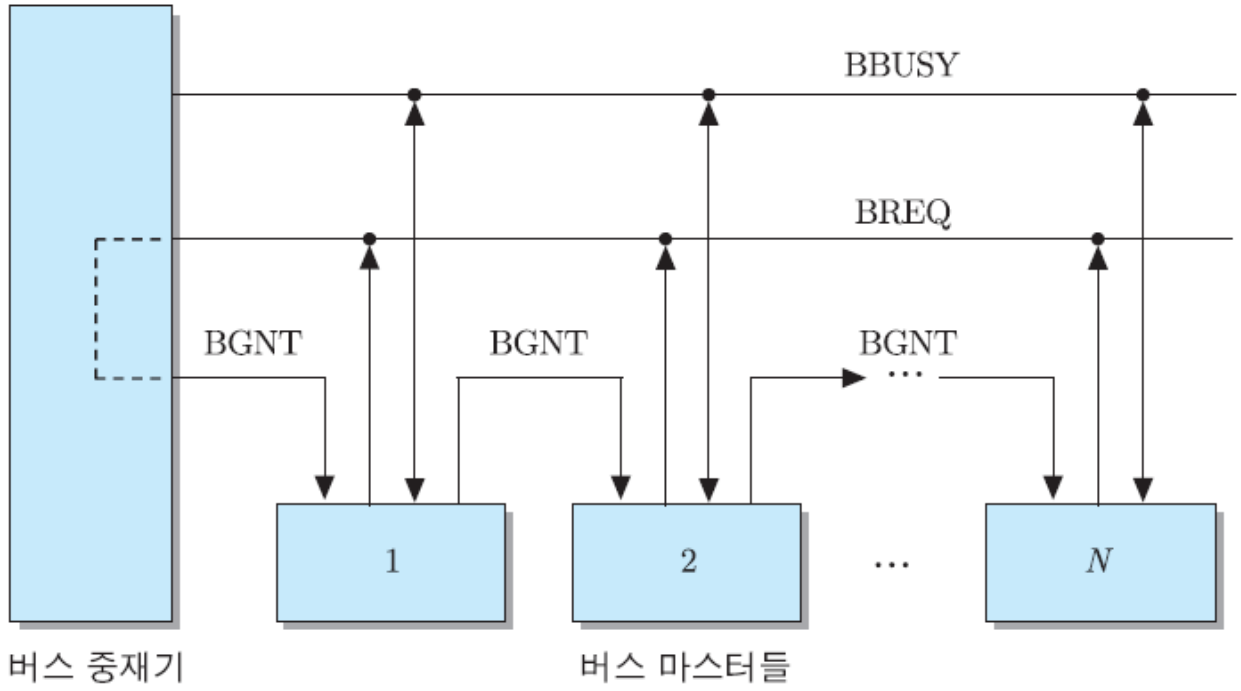
- 한 개 또는 그 이상의 버스 마스터가 버스 사용을 요구하면 공통의 BREQ 신호가 세트됨
- 버스 중재기 : 데이지 체인의 첫 번째에 접속된 마스터로 승인 신호(BGNT) 전송
- BGNT 신호를 받은 마스터는 만약 버스 사용을 요구한 상태라면, 버스 사용권을 가짐

중앙집중식 직렬 중재 방식 (계속)

- 만약 버스 사용을 요구하지 않은 상태라면, 승인 신호를 다음에 연결된 마스터로 통과
- 승인 신호는 버스를 요구한 마스터에게 도달할 때까지 계속 통과

  → 버스 요구를 보낸 마스터들 중에서 중재기에 가장 가까이 위치한(우선순위가 가장 높은) 마스터에게 승인 신호가 전달되면 그 마스터가 버스 사용권을 획득

중앙집중식 직렬 중재 방식의 구성도 (데이지 체인)



2) 분산식 직렬 중재 방식

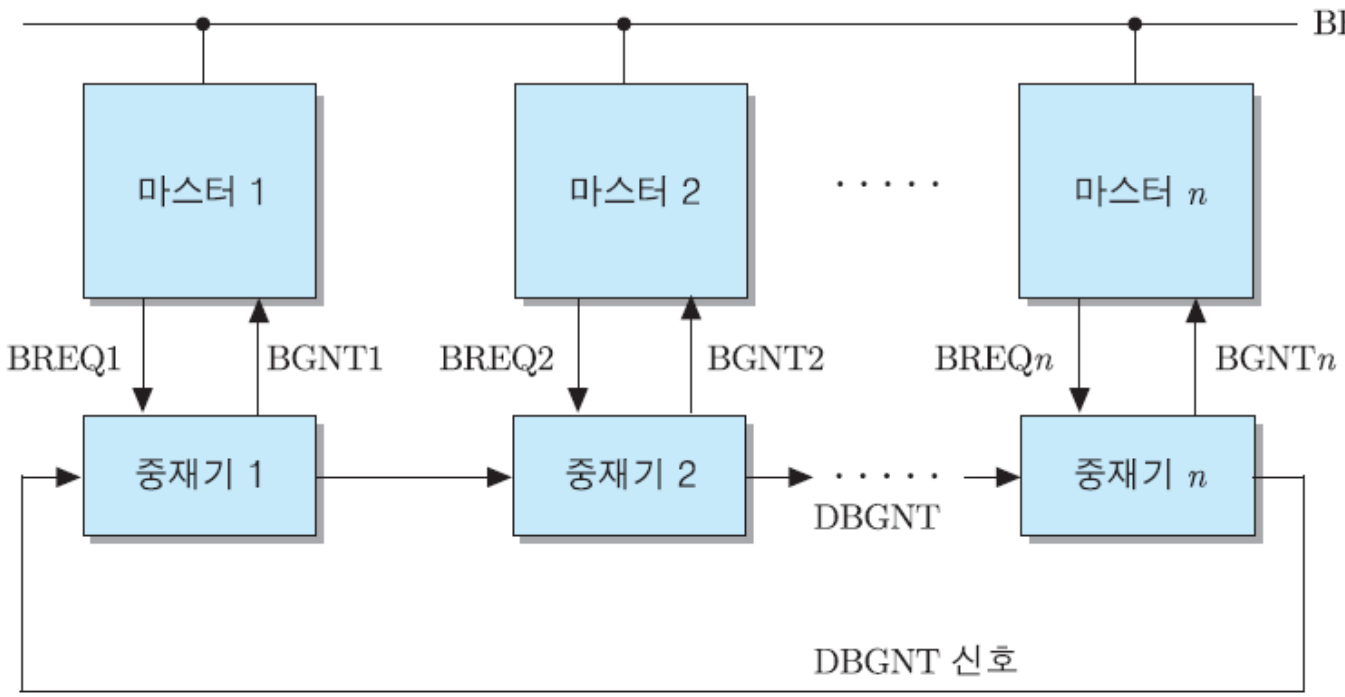
❑ 구성

- 데이지-체인 버스 승인 신호(daisy-chained bus grant signal: DBGNT)선으로 버스 중재기들을 순환형(circular)으로 접속

❑ 동작 원리

- 버스 사용권을 부여 받은 마스터가 버스 사용을 시작하는 순간에 (그 마스터의 중재기는) 자신의 우측에 위치한 마스터의 중재기로 접속된 DBGNT 신호를 세트
- 만약 그 마스터가 버스 사용을 신청하고 기다리던 중이었다면, 중재기는 즉시 DBGNT 신호를 받아들여서 BGNT 신호를 발생시켜 마스터로 전송
- DBGNT 신호를 받은 마스터가 버스 요구를 하지 않은 상태라면, 그 신호를 우측의 다음 중재기로 통과시킴. 그러한 과정은 버스를 요구한 마스터에 도달할 때까지 반복됨

분산식 직렬 중재 방식의 구성도



분산식 직렬 중재 방식 (계속)

- 특징 : 각 마스터의 우선순위가 계속 변화
  - 버스 사용 승인을 받으면 다음 중재 동작에서는 최하위 우선순위를 가짐
  - 버스를 사용한 마스터의 바로 우측에 위치한 마스터가 최상위 우선순위를 가짐
  - 순환형 구조에서 DBGNT 신호가 연결된 순서대로 우선순위가 결정됨
- 단점 : 어느 한 지점에만 결함이 발생해도 전체 시스템의 동작 중단

7.2.3 폴링 방식

❑ 폴링 방식(polling scheme)의 원리

- 버스 사용을 원하는 마스터가 있는지를 버스 중재기가 주기적으로 검사하여 사용 승인 여부를 결정

1) 하드웨어 폴링 방식

- 버스 중재기와 각 버스 마스터 간에 별도의 폴링 선(polling line)이 존재
- 2진 코드화된 폴링 주소(binary encoded polling address)를 이용하면, 폴링 선의 수가 $\log_2 N$ 개로 감소
- 공통의 BREQ 선과 BBUSY 선이 각각 한 개씩 존재

하드웨어 폴링 방식 (계속)

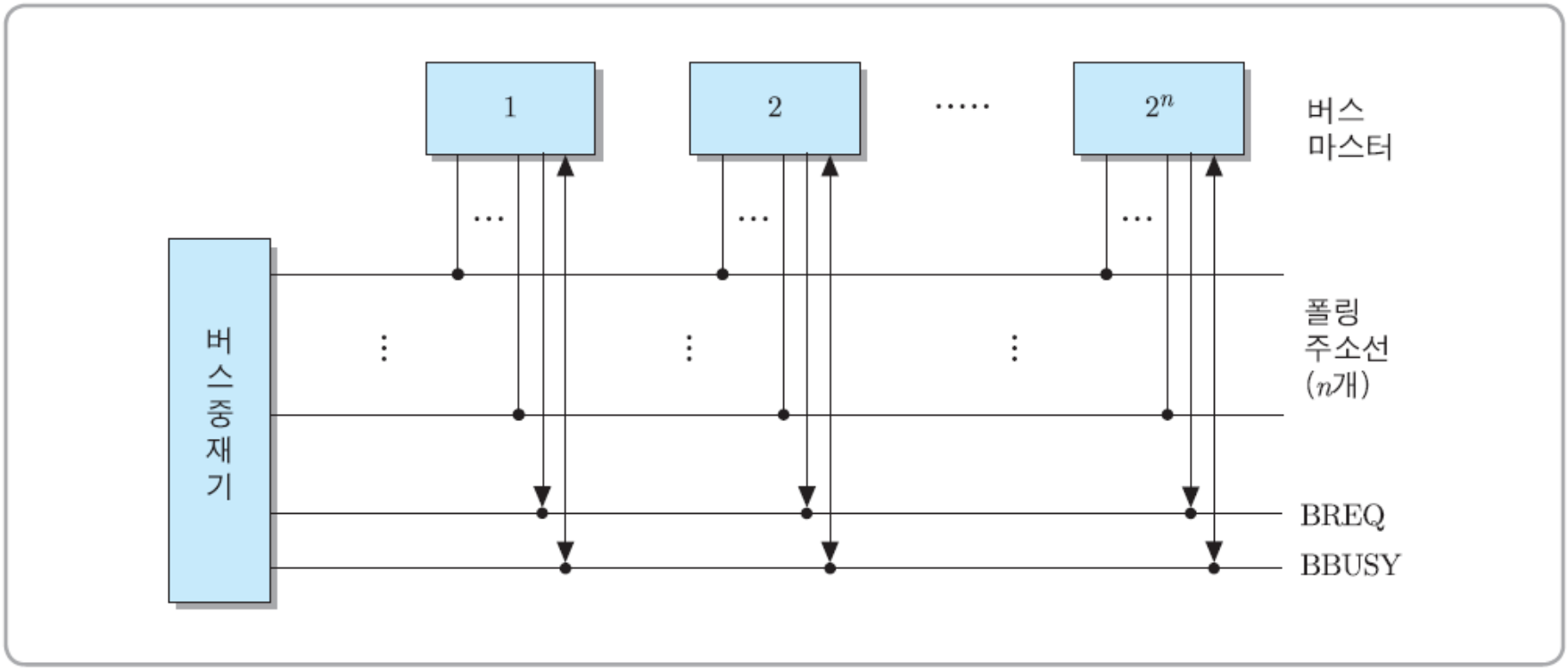
❑ 동작 순서

- 중재기는 폴링 주소를 발생하여 검사할 마스터를 지정한 다음에, 그 마스터가 버스 사용을 원하는지 묻는다
- 지정된 마스터가 버스 사용을 원하면 BREQ 신호를 세트
- BREQ 신호가 세트되면, 중재기는 현재 검사 중인 마스터에게 버스 사용을 허가하고, 그렇지 않으면(지정된 마스터가 버스 사용을 원하지 않으면) 다음 마스터들에 대한 검사를 순서대로 진행

❑ 우선순위 결정 방법

- 중재기가 마스터를 검사하는 순서에 의하여 결정되며, 검사할 마스터의 번호는 2진 카운터(binary counter)를 이용하여 발생

하드웨어 폴링 방식의 구성도



2) 소프트웨어 폴링 방식

- 동작 원리
  - 폴링의 순서와 과정을 버스 중재기내의 프로세서가 관장하는 방식

- 단점: 프로그램을 실행해야 하므로 하드웨어 방식에 비하여 속도가 더 느림

- 장점: 우선순위(폴링 순서)의 변경이 용이

7.3 I/O 장치의 접속

7.3.1 I/O 제어

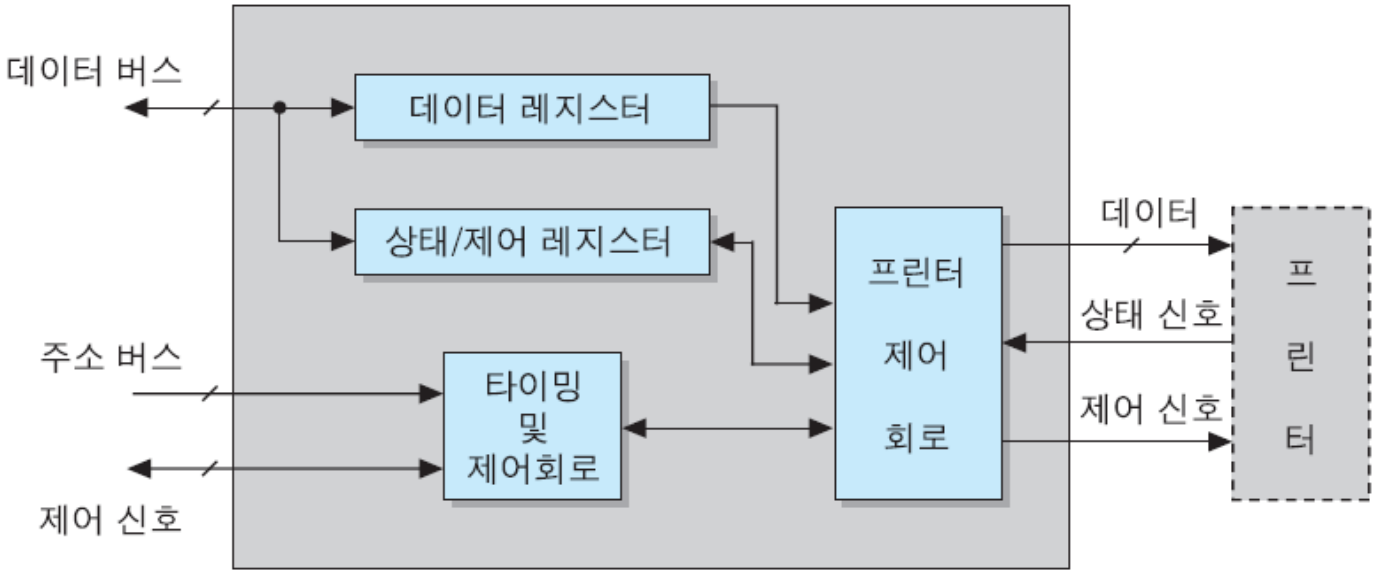
- I/O 장치가 시스템 버스에 직접 접속되지 못하는 이유
  - I/O 장치들은 종류에 따라 제어 방법이 서로 다르며, 그러한 제어 회로들을 CPU 내부에 모두 포함시키는 것이 불가능하기 때문에 CPU가 그들을 직접 제어할 수가 없음
  - I/O 장치들의 데이터 전송 속도가 CPU의 데이터 처리 속도에 비하여 훨씬 더 느리고 속도가 서로 다르기 때문에, 고속의 시스템 버스와 I/O 장치들 사이에 직접 데이터를 교환하는 것은 불가능
  - I/O 장치들과 CPU가 사용하는 데이터 형식의 길이가 서로 다른 경우가 많음

➔ 인터페이스 장치인 I/O 제어기(I/O controller)를 사용

I/O 제어기의 주요 기능

- I/O 장치의 제어와 타이밍 조정
- CPU와의 통신 담당
- I/O 장치와의 통신 담당
- 데이터 버퍼링(data buffering) 기능 수행
- 오류 검출

프린터 제어기의 내부 구성도



상태/제어 레지스터 (status/control register)

- 내부적으로 두 개의 레지스터로 구성되지만 주소는 하나만 지정 되는 레지스터들
- 상태 레지스터 : I/O 장치의 상태와 오류 검사 결과 등을 나타내는 비트들로 구성 (CPU에 의한 읽기 동작 시 선택됨)
- 제어 레지스터 : CPU가 보낸 I/O 명령 단어(I/O command word)를 저장 (CPU에 의한 쓰기 동작 시 선택됨)
- CPU가 프린터로 데이터를 출력하는 과정
  1. CPU가 프린터 제어기에게 프린터의 상태를 검사하도록 요청
  2. 제어기는 프린터의 상태를 검사하여 준비되었는지를 가리키는 비트(RDY 비트)를 세트

CPU가 프린터로 데이터를 출력하는 과정 (계속)

③ CPU는 RDY 비트를 (반복) 검사하고, 프린터가 다음 프린트를 시작할 준비가 된 상태라면 제어기로 출력 명령과 데이터를 전송

④ 제어기는 프린트 동작을 제어하기 위한 신호들과 데이터를 프린터로 전송

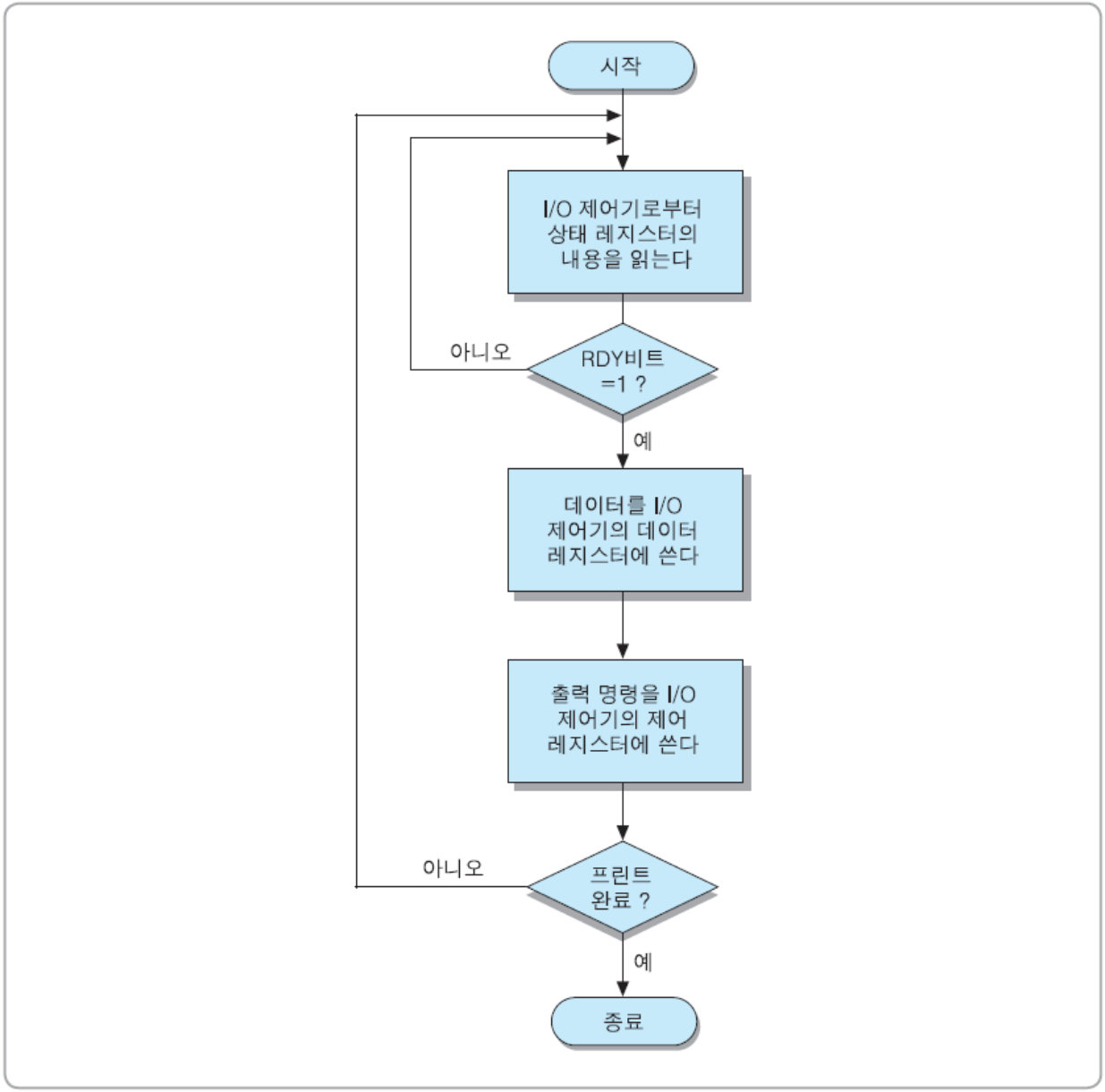
* 데이터 레지스터를 큰 용량의 반도체 기억장치로 이루어진 데이터 버퍼(data buffer)로 대체한다면, 데이터 블록 전체를 한꺼번에 전송받아 프린트 가능

프로그램을 이용한 I/O (programmed I/O)

- CPU가 반복적으로 I/O 장치의 상태를 검사하면서 I/O 동작을 처리하는 방식

- 장점
  - 간단하며, 별도의 하드웨어가 필요하지 않음
- 단점
  - CPU가 I/O 동작에 직접 관여해야 하므로, 그 동안에 다른 일을 하지 못함

프로그램을 이용한 I/O의 흐름도



7.3.2 I/O 주소지정(I/O addressing)

- 각 I/O 장치당 두 개씩의 주소 할당
  - 데이터 레지스터 주소, 상태/제어 레지스터 주소
- I/O 주소 지정 방법
  - 기억장치-사상 I/O(memory-mapped I/O)
  - 분리형 I/O(isolated-I/O)

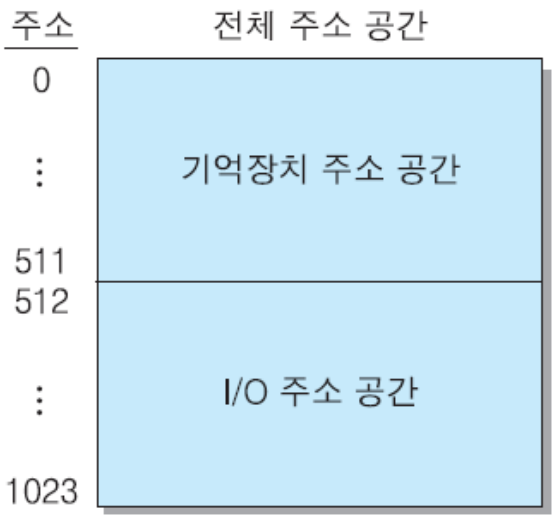
1) 기억장치-사상 I/O

- 기억장치 주소 영역의 일부분을 I/O 제어기 내의 레지스터들의 주소로 할당하는 방식
- 프로그래밍에서 기억장치 관련 명령어들을 I/O 장치 제어에도 사용 가능

  [예] LOAD 명령어, STORE 명령어, 등
- 기억장치 읽기/쓰기 신호를 I/O 읽기/쓰기 신호로 사용 (별도의 I/O 제어 신호가 필요하지 않음)

기억장치-사상 I/O의 주소 공간 할당의 예

- 주소 비트들이 10 비트인 경우 → 전체 기억 장소들의 수 = 1024
- 0 번지~511 번지(상위 512 개 주소) : 기억장치에 할당
- 512 번지~1023 번지(하위 512 개 주소) : I/O 장치들에 할당



기억장치-사상 I/O의 예

❑ 기억장치-사상 I/O 방식을 이용한 경우의 프린터 출력 프로그램

- 데이터 레지스터 주소 : 512 번지
- 상태/제어 레지스터 주소 : 513 번지
- 상태 레지스터 최하위 비트(b_0) : RDY 비트로 사용
- 제어 레지스터 최상위 비트(b_7) : 프린트 시작(start) 비트로 사용

```
TEST :  LOAD    513     ; 상태 레지스터의 내용을 읽는다.
        ANI     01      ; RDY 비트를 제외한 모든 비트들을 '0'으로 리셋 시킨다.
        JZ      TEST    ; 만약 RDY 비트가 '0'이라면, TEST로 점프한다.
                        ;
        LOAD    M[a]    ; 프린트할 데이터를 기억장치 a번지로부터 읽어온다.
        STOR    512     ; 프린트할 데이터를 데이터 레지스터에 쓴다.
        LOAD    80H     ; AC에 2진수 '10000000'을 적재한다(START 비트 ← '1')
        STOR    513     ; 프린트 시작 명령을 보낸다.
```

기억장치-사상 I/O (계속)

[장점] 프로그래밍이 용이 (사용 가능한 명령어들이 다양)

[단점] 기억장치 주소 공간이 감소 (절반)

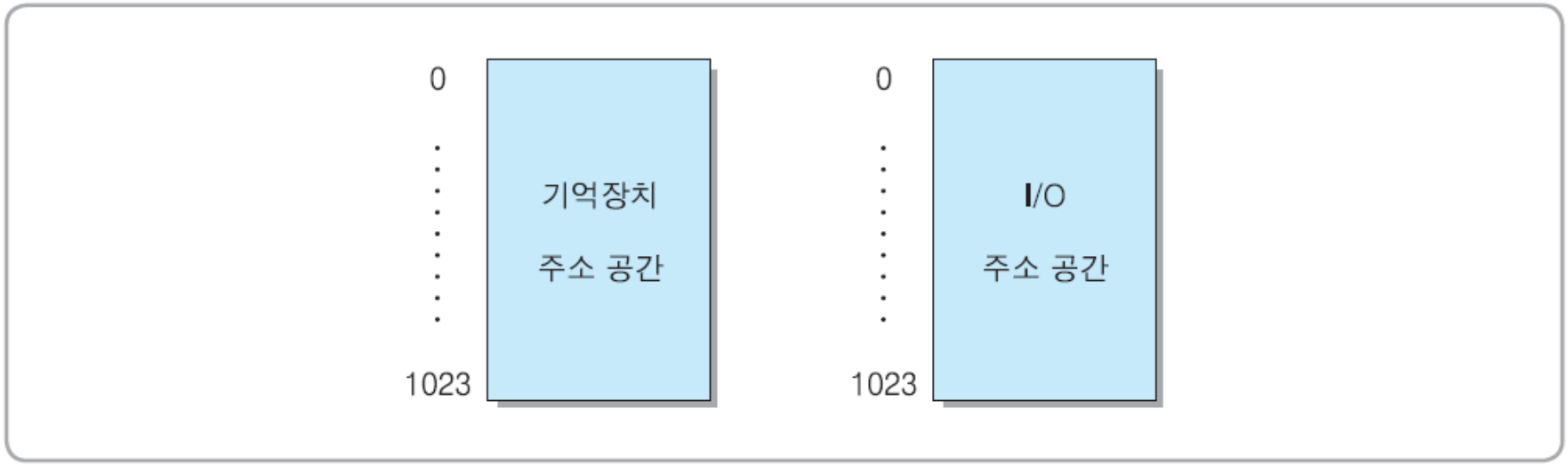
2) 분리형 I/O (isolated-I/O)

- I/O 장치 주소공간을 기억장치 주소 공간과는 별도로 할당하는 방식
- I/O 제어를 위해서 별도의 I/O 명령어 사용
  (예: IN, OUT 명령어)
- 별도의 I/O 읽기/쓰기 신호 필요

분리형 I/O의 주소 공간 할당의 예

- 주소 비트의 수가 10 개일 때,

  기억장치 주소와 I/O 주소를 각각 아래와 같이 1024 개씩 할당 가능



[단점] I/O 제어를 위해 I/O 명령어들만 이용할 수 있으므로, 프로그래밍이 불편

[장점] I/O 주소공간으로 인하여 기억장치 주소 공간이 줄어들지 않음

분리형 I/O의 예

- 분리형 I/O 방식을 이용한 경우의 프린터 출력 프로그램
  - I/O 장치의 데이터 레지스터 주소 : 0 번지
  - 상태/제어 레지스터 주소 : 1 번지

```
TEST :  IN      1       ; 상태 레지스터의 내용을 읽는다.
        ANI     01      ; RDY비트를 제외한 모든 비트들을 '0'으로 리셋 시킨다.
        JZ      TEST    ; 만약 RDY비트가 '0'이라면, TEST로 점프한다.
                        ;
        LOAD    M[a]    ; 프린트할 데이터를 기억장치 a번지로부터 읽어온다.
        OUT     0       ; 프린트할 데이터를 데이터 레지스터에 쓴다.
        LOAD    80H     ; AC에 2진수 '10000000'을 적재한다(START 비트 ← '1').
        OUT     1       ; 프린트 시작 명령을 보낸다.
```

7.4 인터럽트를 이용한 I/O

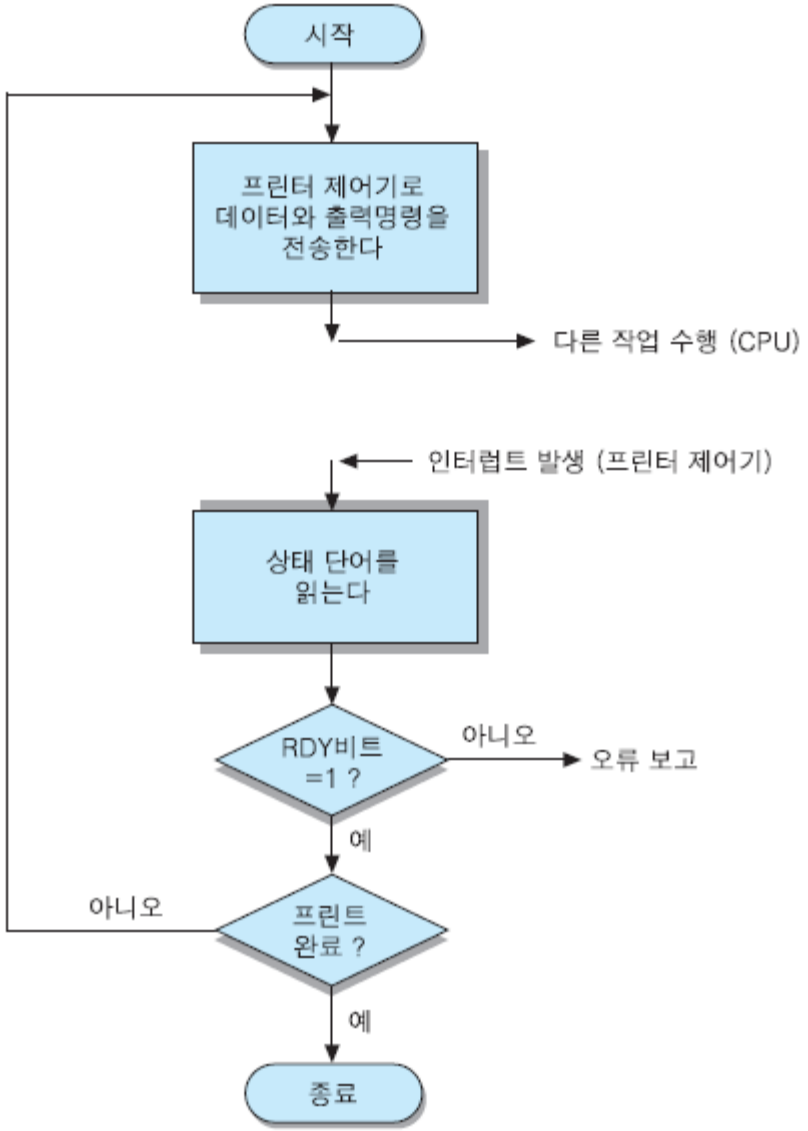
- 인터럽트-구동 I/O(interrupt-driven I/O) : 인터럽트 메커니즘을 이용함으로써, I/O 동작이 진행되는 동안에 CPU가 다른 작업을 처리할 수 있도록 하는 방식
- 동작 순서
  - CPU가 I/O 제어기에게 명령을 전송하고, CPU는 다른 작업 수행
  - 제어기는 I/O 장치를 제어하여 I/O 명령을 수행
  - I/O 명령 수행이 완료되면, 제어기는 CPU로 인터럽트 신호를 전송
  - CPU는 인터럽트 신호를 받는 즉시 원래의 프로그램으로 복귀하여 수행을 계속

인터럽트-구동 I/O 방식에서의 프린터 출력 흐름도

① CPU는 데이터와 프린트 명령을 프린터 제어기로 전송하고, 다른 작업을 수행

② 그 데이터의 프린트가 종료되면, 제어기가 CPU로 인터럽트 요구 신호를 전송

③ 프린트할 내용이 남아 있다면, CPU는 다음에 프린트할 데이터를 준비하여 위의 과정을 반복

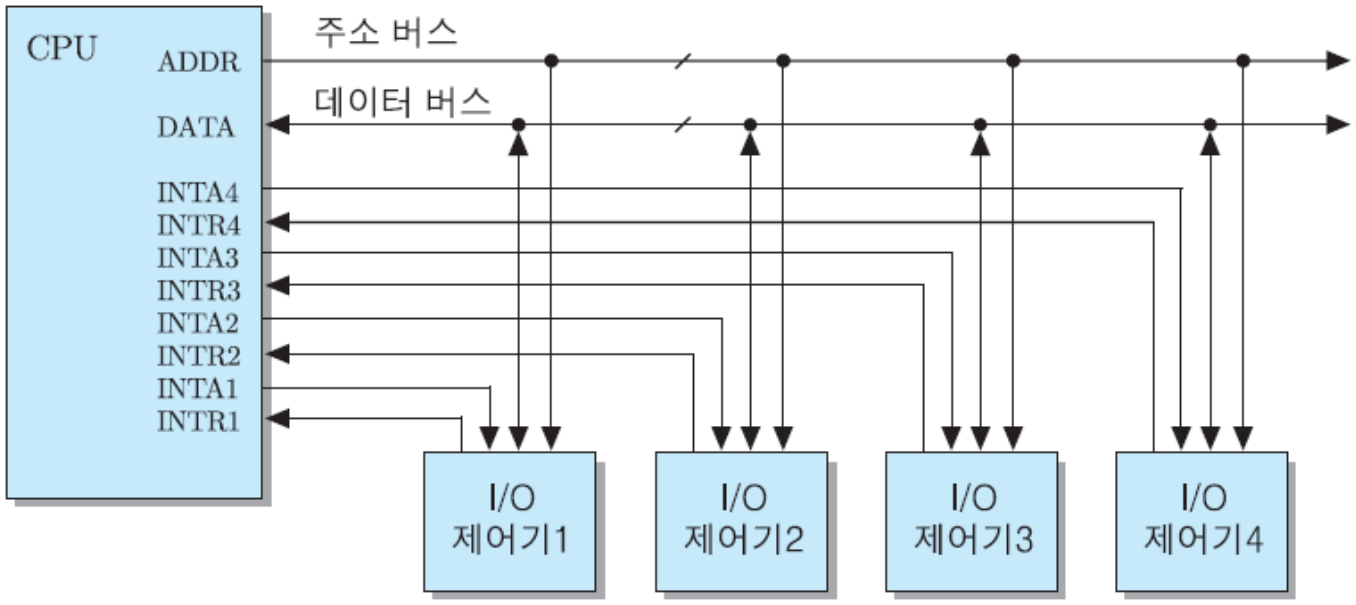


인터럽트-구동 I/O의 구현 방법

- 다중-인터럽트(multiple interrupt) 방식
- 데이지-체인(daisy-chain) 방식
- 소프트웨어 폴링(software polling) 방식

7.4.1 다중-인터럽트 방식

- 각 I/O 제어기와 CPU 사이에 별도의 인터럽트 요구 (interrupt request: INTR) 선과 인터럽트 확인(interrupt acknowledge: INTA) 선을 접속하는 방법



다중-인터럽트 방식의 예

- I/O 제어기 #2가 인터럽트를 요구하는 경우의 동작 순서

1. I/O 제어기#2가 INTR2 신호를 세트
2. CPU는 INTA2 신호를 세트 함으로써 그 제어기에게 인터럽트 요구를 인식하였음을 알리고, 인터럽트를 위한 서비스를 시작
3. I/O 제어기2는 INTR2 신호를 해제(0으로 리셋)
4. CPU도 INTA2 신호를 해제

다중-인터럽트 방식 (계속)

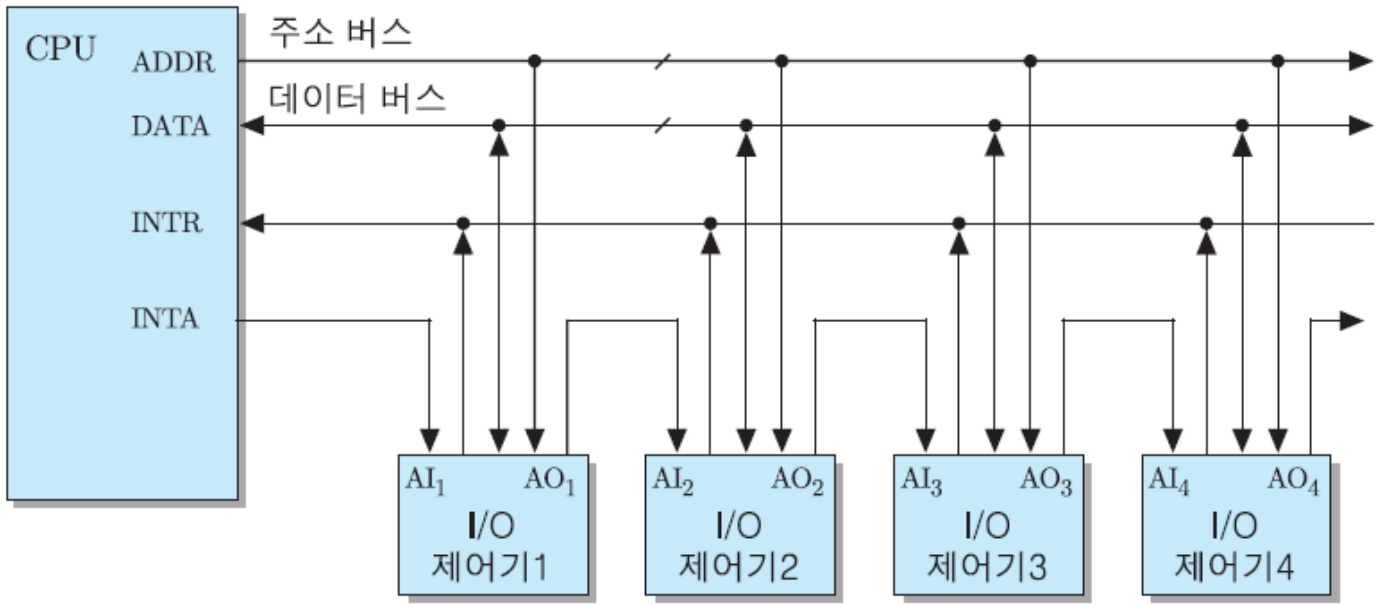
[장점] CPU가 인터럽트를 요구한 장치를 쉽게 찾아낼 수 있다

[단점] 하드웨어가 복잡하고, 접속 가능한 I/O 장치들의 수가 CPU의 인터럽트 요구 입력 핀의 수에 의해 제한된다

7.4.2 데이지-체인 방식

- CPU로부터 발생되는 **INTA** 출력 선을 I/O 제어기들에 직렬로 접속하는 방식
- 인터럽트를 요구한 I/O 장치는 **AIn** 입력을 받는 즉시 자신의 고유(ID) 번호, 즉 인터럽트 벡터(interrupt vector)를 데이터 버스를 통하여 CPU로 전송
  - 인터럽트 벡터는 해당 I/O 장치를 위한 인터럽트 서비스 루틴의 시작 주소를 결정하는데 사용
- 만약 **AIn** 입력을 받은 I/O 장치가 인터럽트를 요구하지 않은 상태라면, 그 입력을 다음 장치로 통과(pass)시킴

데이지-체인 방식의 구성도

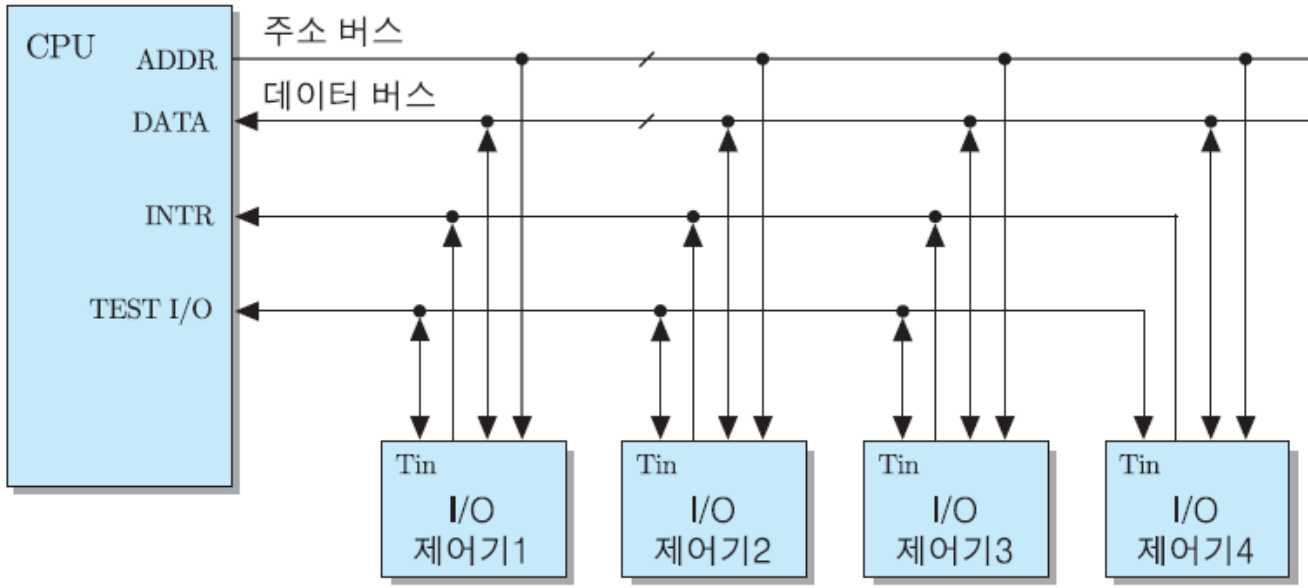


[장점] 하드웨어가 간단

[단점] 우선순위가 낮은 장치들이 서비스를 받지 못하고
매우 오랫동안 기다려야 하는 경우가 발생(starvation)

7.4.3 소프트웨어 폴링 방식

- CPU가 모든 I/O 제어기들에 접속된 TEST I/O 선을 이용하여 인터럽트를 요구한 장치를 검사하는 방식
  - Test I/O 신호를 이용하여 각 I/O 장치의 인터럽트 플래그가 세트되어 있는 지를 검사 (검사 순서가 우선순위를 결정)



소프트웨어 폴링 방식 (계속)

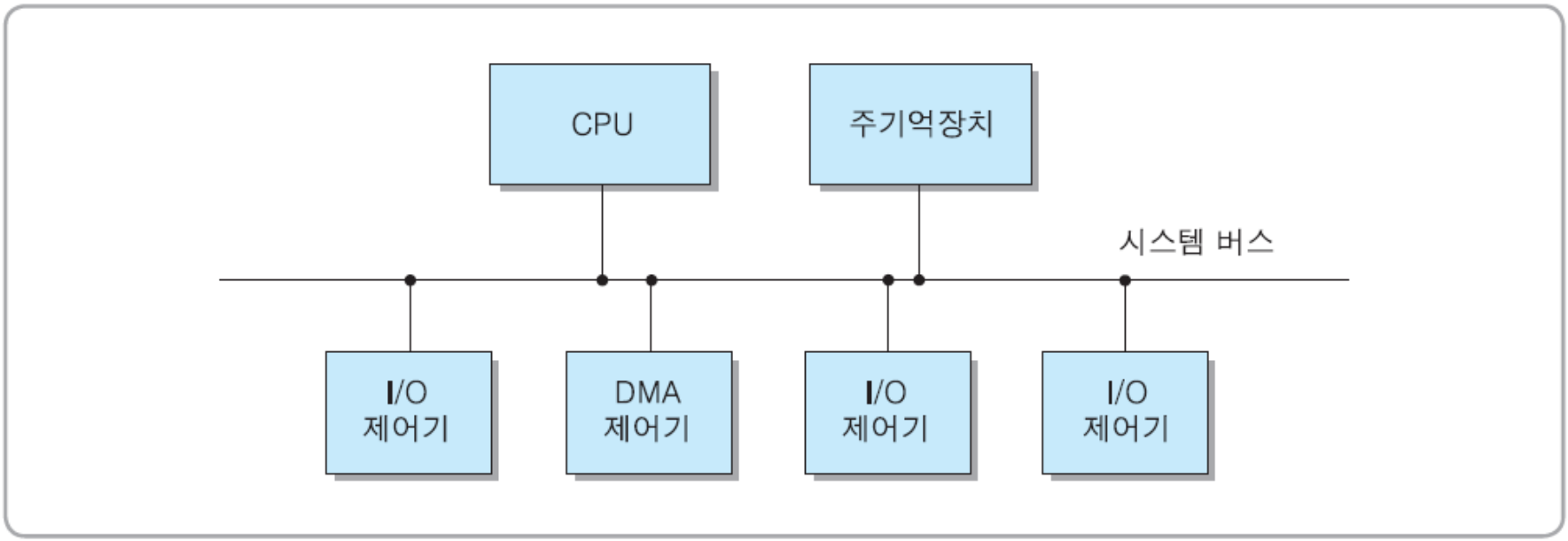
[장점] 우선순위의 변경이 용이

[단점] 처리 시간이 오래 걸림

7.5 직접기억장치액세스

- Direct Memory Access(DMA) : CPU의 개입 없이 I/O 장치와 기억장치 사이에 데이터를 전송하는 방식 (사이클 스틸링(cycle stealing)이라고도 함)
- 방법 : CPU가 주기억장치를 액세스하지 않는 시간(CPU가 내부적으로 명령어를 해독하거나 ALU 연산을 수행하는 시간) 동안에 시스템 버스를 사용하여 주기억장치와 I/O 장치 간에 데이터 전송

DMA 제어기가 포함된 시스템 구성도



DMA 처리 순서

① CPU가 DMA 제어기(DMA controller)로 아래 정보를 포함한 명령을 전송

- I/O 장치의 주소 (예: 디스크 드라이브 번호, 실린더 번호, 섹터 번호)
- 연산(쓰기 혹은 읽기) 지정자
- 데이터가 쓰여지거나 읽혀질 주기억장치 영역의 시작 주소
- 전송될 데이터 단어들의 수

② DMA 제어기는 CPU로 버스 요구(BUS REQ) 신호를 전송

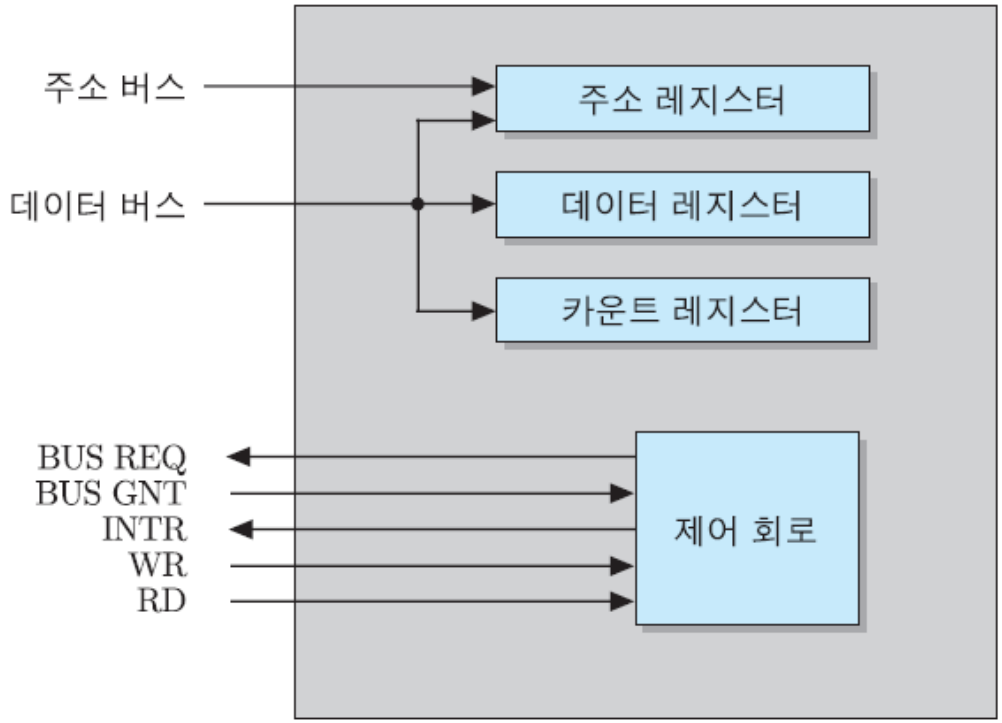
③ CPU가 DMA 제어기로 버스 승인(BUS GRANT) 신호를 전송

④ DMA 제어기가 주기억장치로부터 데이터를 읽어서, 디스크 제어기로 전송.
각 데이터에 대하여 ②,③,④번을 두 번 반복 (주기억장치 → DMA 제어기 & DMA 제어기 → 디스크 제어기 → 데이터 저장) : 시스템 버스 두 번 사용

⑤ 저장할 데이터들이 남아있다면, 그 수만큼 ②~④번 동작을 반복

⑥ 모든 데이터들의 저장이 완료되면, CPU로 INTR 신호를 전송

DMA 제어기의 내부 구조



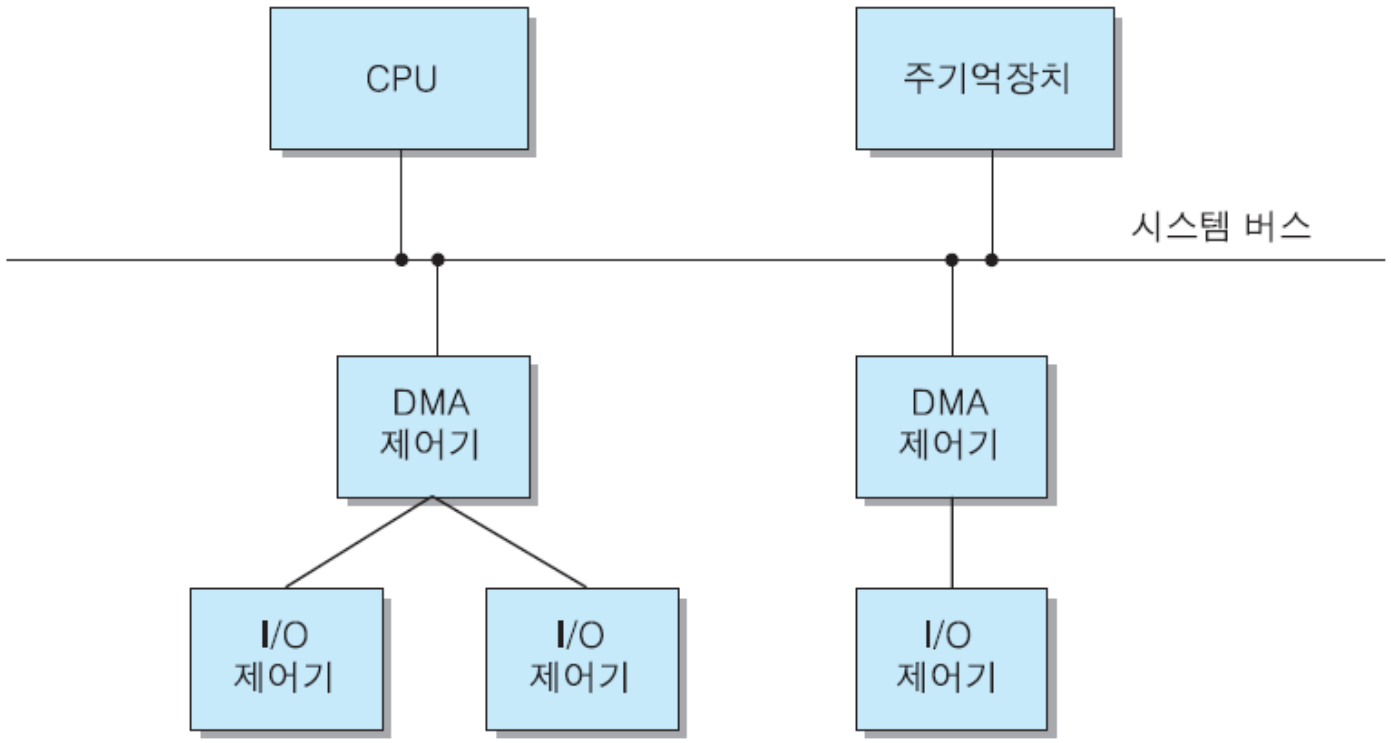
DMA의 문제점 [I]

- 각 데이터 전송 때마다 시스템 버스를 두 번씩 사용

  ➔ 버스 사용량 증가로 인한 시스템 성능 저하

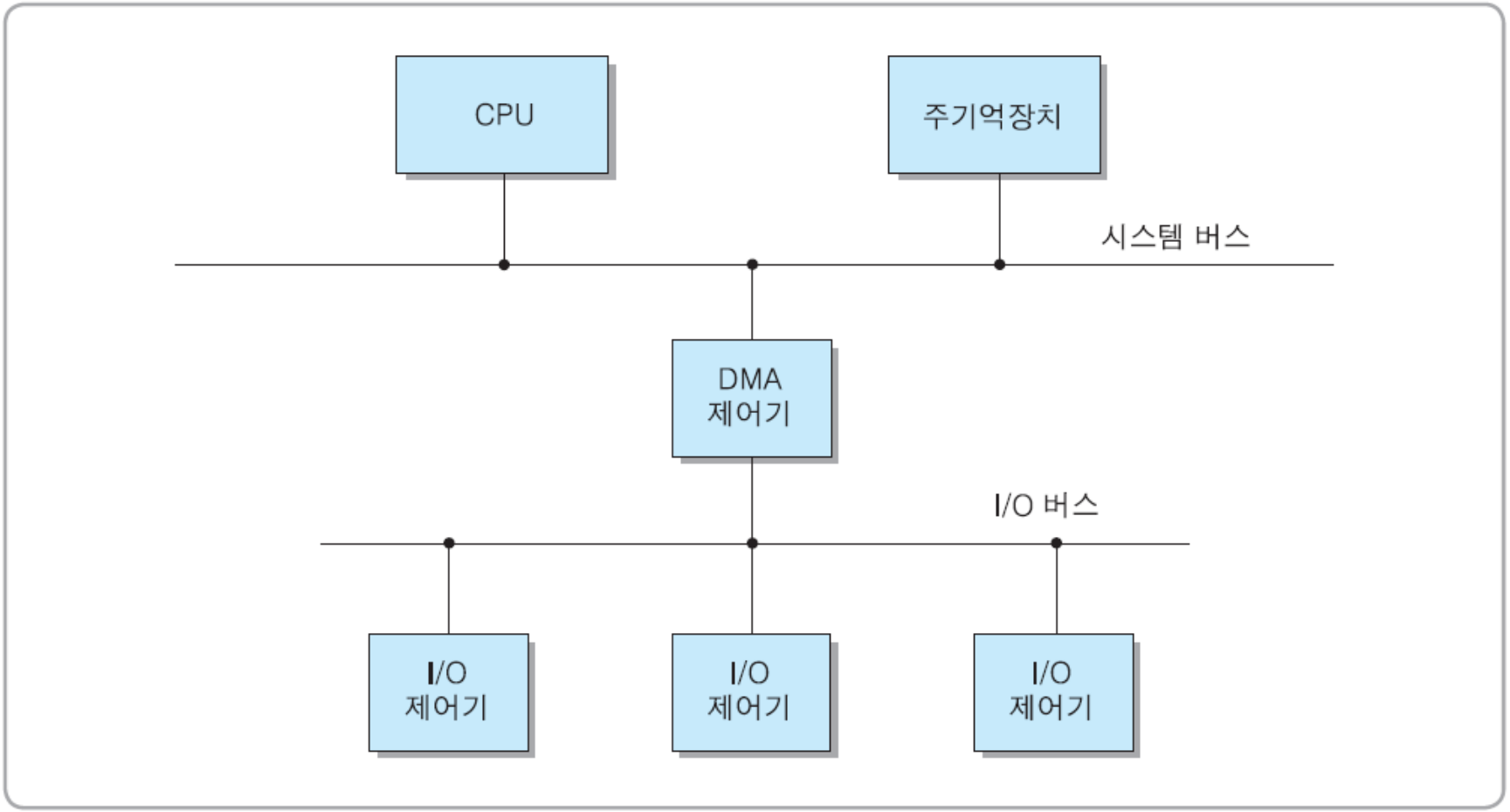
- 해결책
  - I/O 장치들을 DMA 제어기에 접속 ➔ DMA 제어기가 주 기억장치를 액세스 할 때만 시스템 버스 사용

DMA 제어기를 이용한 I/O 접속 방법



I/O 버스를 이용한 DMA 구성도

- I/O 버스 사용 ➔ 한 개의 DMA 제어기가 다수의 I/O 장치들을 지원



DMA의 문제점 [II]

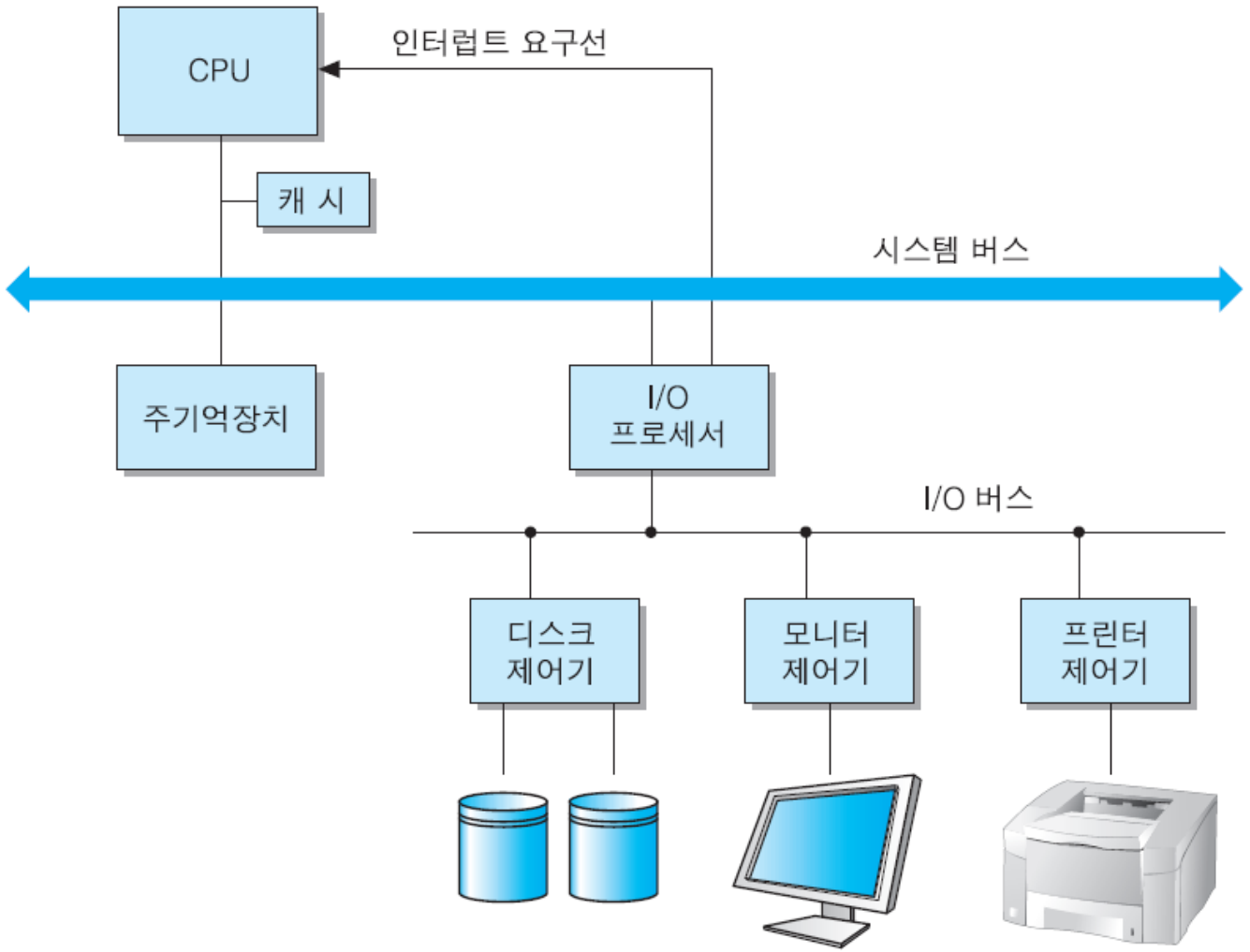
- I/O 장치들은 종류와 속도가 다양하고 제어 방법도 복잡하기 때문에, 간단한 구조를 가진 DMA 제어기로 지원하는 데는 한계가 있음
- 디스크 쓰기 혹은 읽기 동작의 경우에는 데이터 블록의 크기가 512 바이트 이상이기 때문에 그 데이터들을 버퍼링(임시 저장)하기 위한 내부 기억장치가 필요

- 해결책 : I/O 프로세서(I/O processor: IOP) 사용 ---
  I/O 채널(I/O channel)이라고도 함

IOP

- IOP 보드에 포함될 요소들
  - I/O 제어 프로그램(OS의 I/O device driver 프로그램)을 실행할 수 있는 프로세서
  - 데이터 블록을 임시 저장(버퍼링) 할 수 있는 용량의 지역 기억장치(local memory)
  - 시스템 버스 인터페이스 및 버스 마스터 회로
  - I/O 버스 중재 회로

I/O 프로세서가 사용된 시스템의 구성도



기 본 문 제

7.1 다음 버스들 중에서 양방향 전송 기능을 가져야 하는 버스는 어느 것인가?

가. 주소 버스
나. 데이터 버스
다. 제어 버스
라. 인터럽트 버스

7.2 펜티엄 4 마이크로프로세서의 주소 버스 폭은 36비트이다. 주소지정이 가능한 최대 기억장치 용량은?

가. 16MByte
나. 64MByte
다. 32GByte
라. 64GByte

7.3 다음 중에서 기본적인 시스템 버스에 속하지 않는 것은?

가. 제어 버스
나. 데이터 버스
다. 주소 버스
라. I/O 버스

7.4 버스 클록의 주파수가 100MHz이고, 데이터 버스 폭이 64비트라면, 이 버스의 대역폭은 몇 [Mbytes/sec]가 되는가?

가. 800
나. 64
다. 640
라. 80

7.5 다음 중에서 각 버스 마스터가 버스 요구 선과 승인 선을 별도로 가지고 있는 버스 중재 방식이 아닌 것은?

가. 데이지-체인 방식
나. 중앙집중식 고정 우선순위 방식
다. 분산식 고정 우선순위 방식
라. 분산식 직렬 중재 방식

7.6 다음 중에서 모든 버스 마스터들이 균등하게 버스를 사용할 수 있게 해주는 버스 중재 방식은?

가. 고정 우선순위 방식
나. 분산식 직렬 중재 방식
다. 회전 우선순위 방식
라. 하드웨어 폴링 방식

7.7 다음 중에서 버스 중재기가 여러 개 필요한 중재 방식은 ?

가. 데이지-체인 방식
나. 중앙집중식 고정 우선순위 방식
다. 분산식 고정 우선순위 방식
라. 소프트웨어 폴링 방식

7.8 CPU가 I/O 장치를 직접 접속하여 제어하지 못하는 이유가 아닌 것은?

가. I/O 장치마다 제어 방식이 서로 다르다.
나. I/O 장치의 데이터 전송 속도가 시스템 버스의 속도보다 더 느리다.
다. I/O 장치의 데이터 형식이 다양하다.
라. 공급되는 전원이 다르기 때문이다.

7.9 다음 중에서 I/O 제어기의 구성 요소가 아닌 것은?

가. 데이터 레지스터
나. 산술논리연산장치
다. 상태 레지스터
라. 장치 제어회로

7.10 기억장치-사상 I/O 방식을 사용하는 시스템에서 주소가 12비트이면, I/O 장치를 최대 몇 개까지 접속할 수 있는가?

가. 12개
나. 2048개
다. 4096개
라. 1024개

7.11 다음 중에서 분리형 I/O 방식의 가장 큰 장점에 해당하는 것은?

가. 프로그래밍이 용이하다.
나. 기억장치 공간이 줄어들지 않는다.
다. 제어 신호의 수가 줄어든다.
라. I/O 인터페이스가 간단하다.

7.12 데이지-체인 방식으로 인터럽트를 요구한 장치를 찾아내는 방법에 관한 설명으로 잘못된 것은?

가. 모든 장치들이 직렬로 연결된다.
나. 장치들의 우선순위가 고정된다.
다. 접속할 수 있는 장치의 수가 CPU의 인터럽트 입력 핀의 수에 의해 제한된다.
라. 인터럽트 벡터를 이용하여 장치의 ID를 CPU로 알려준다.

7.13 소프트웨어 폴링 방식을 이용한 인터럽트-구동 I/O의 단점은 다음 중 어느 것인가?

가. 처리 시간이 오래 걸린다.
나. 하드웨어가 복잡하다.
다. 우선순위가 고정된다.
라. 연결될 수 있는 장치의 수가 제한된다.

7.14 인터럽트 처리에서 I/O 장치들의 우선순위를 지정하는 이유는 무엇인가?

가. 인터럽트 발생 빈도를 확인하기 위하여
나. CPU가 하나 이상의 인터럽트를 처리하지 못하게 하기 위하여
다. 중요도가 더 높은 장치에 대한 서비스를 먼저 처리하기 위하여
라. 인터럽트 처리 루틴의 주소를 알기 위하여

7.15 CPU가 DMA 제어기로 보내는 정보가 아닌 것은?

가. 전송될 데이터의 수
나. 연산 지정자
다. I/O 장치의 주소
라. CPU 레지스터 번호

7.16 DMA에 관한 설명으로 옳은 것은?

가. 저속 I/O 장치를 위하여 사용된다.
나. 대형 컴퓨터에서만 사용된다.
다. CPU 개입 없이 I/O 장치와 기억장치 간에 데이터를 전송한다.
라. 수행되는 동안 CPU는 상태 검사를 반복한다.

7.17 DMA 제어기의 한계를 극복하기 위하여 사용하는 방식은 어느 것인가?

가. I/O 프로세서
나. 다중 인터럽트
다. 프로그램을 이용한 I/O
라. 멀티플렉싱

연 습 문 제

7.1 버스 클록이 300MHz이고 데이터 버스의 폭이 64비트인 시스템 버스의 대역폭은 몇 [GBytes/sec]가 되는지 구하라.

7.2 동기식 버스와 비동기식 버스의 상대적 장점을 각각 설명하라.

7.3 병렬 중재 방식과 직렬 중재 방식의 상대적 장점을 각각 설명하라.

7.4 중앙집중식 중재 방식과 분산식 중재 방식의 상대적 장점을 각각 설명하라.

7.5 네 개의 버스 마스터들이 한 개의 버스 중재기에 의해 중재되는 중앙집중식 고정 우선순위 방식에 대하여 다음 물음에 답하라. 단, 버스 마스터1이 가장 높은 우선순위를 가지며, 버스 마스터4가 가장 낮은 우선순위를 가지는 것으로 가정한다.

(1) 전체 구성도를 그려라.
(2) 버스 마스터3이 버스를 사용하고 있는 중에 버스 마스터1이 버스 요구를 발생한 경우에 대한 시간 흐름도를 그려라.
(3) (2)번의 경우에 만약 BBUSY 신호가 없다면 어떤 문제점이 발생할 수 있는지 설명하라.

7.6 가변 우선순위 중재 방식들 중에서 회전 우선순위 방식을 구현하는 두 가지 방법을 설명하고, 장단점을 비교하라. 또한, 버스를 빈번히 사용하는 버스 마스터가 존재하는 경우에 어느 방식이 더 효율적인가?

7.7 주소 버스가 12비트인 시스템에서 I/O 주소지정을 위하여 아래의 방식들이 사용되는 경우에, 최대 몇 개의 I/O 장치들을 접속할 수 있는가?

(a) 분리형 I/O
(b) 기억장치-사상 I/O

7.8 기억장치-사상 I/O 방식과 분리형 I/O 방식의 장단점을 비교하라.

7.9 인터럽트를 처리하기 위한 다음 방법들에 대하여 아래 물음에 답하라.

(a) 데이지-체인 방식
(b) 소프트웨어 폴링 방식
(c) 다중 인터럽트 방식

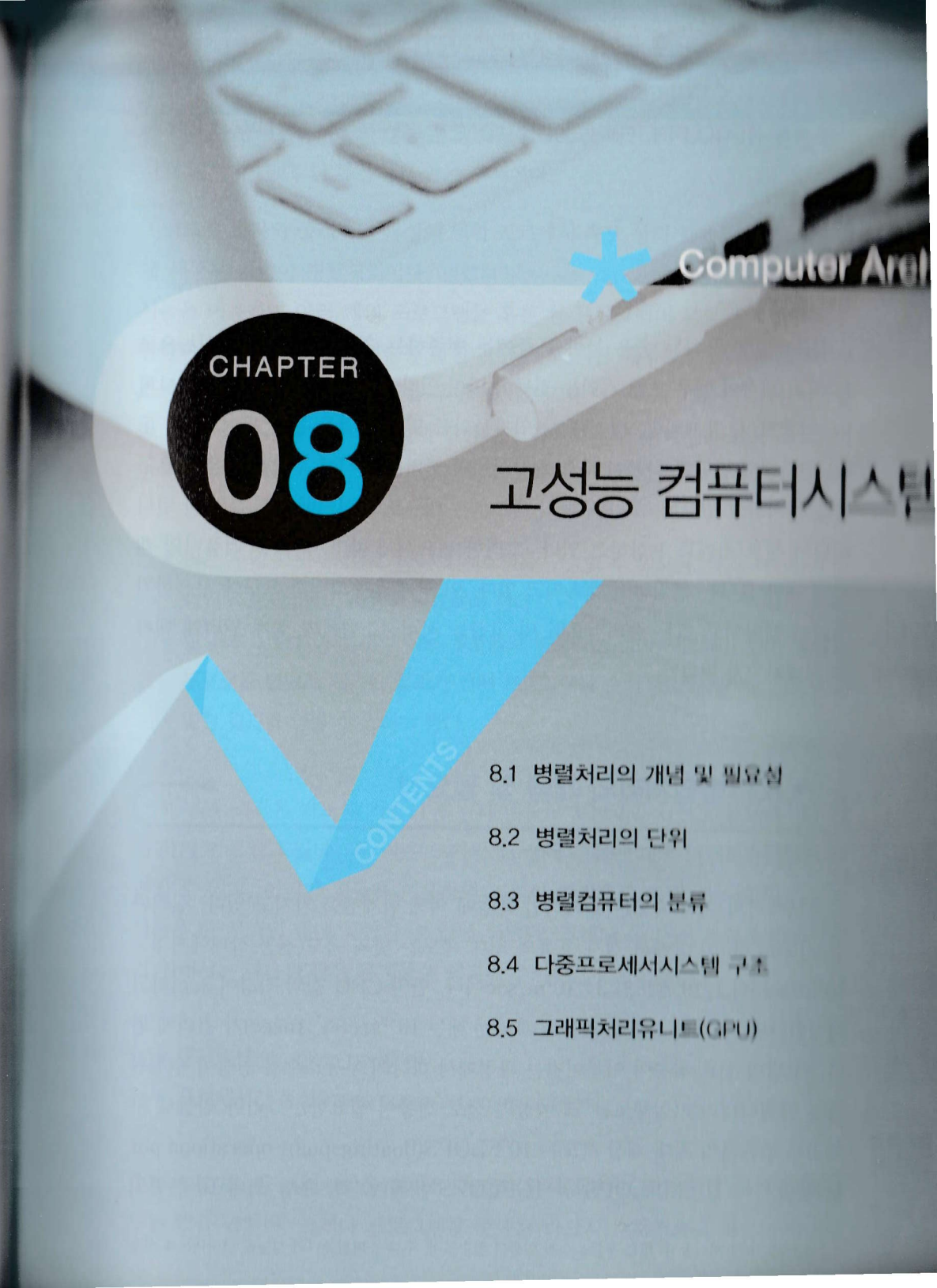
(1) 하드웨어가 가장 간단한 방식은 어떤 것인가?
(2) 우선순위를 쉽게 변경할 수 있는 방식은 어떤 것인가?
(3) 처리 속도가 가장 빠른 것은 어떤 방식인가?

7.10 데이지-체인 I/O 접속 방식에서 어느 한 I/O 장치가 기근(starvation)될 가능성은 없는가? 만약 있다면, 그 이유를 설명하라.

7.11 시스템 버스에 네 개의 I/O 제어기들과 한 개의 DMA 제어기가 접속되어 있다. 이 구성의 문제점을 들라.

7.12 문제 7.11과 같은 시스템 구성에 대한 개선 방안으로서, 모든 I/O 제어기들을 DMA 제어기에 접속할 수 있다.

(1) 이와 같은 구성은 어떤 문제점을 가지는가?
(2) 만약 I/O 장치들이 모두 서로 다른 유형이라면, 추가적으로 어떤 어려움이 발생하는가?



Computer Arch

CHAPTER 08

고성능 컴퓨터시스템

CONTENTS

8.1 병렬처리의 개념 및 필요성
8.2 병렬처리의 단위
8.3 병렬컴퓨터의 분류
8.4 다중프로세서시스템 구조
8.5 그래픽처리유니트(GPU)

제7장

7.1 8 bytes × 300 MHz = 2400 [Mbytes/sec]

7.2 동기식 버스는 인터페이스 회로가 간단하다는 장점이 있다. 그러나 가장 오래 걸리는 버스 동작의 소요시간을 기준으로 하여 버스 클럭의 주파수를 정해야 하기 때문에, 클럭 주기보다 더 짧은 시간이 걸리는 버스 동작의 경우에는 동작이 완료된 때부터 다음 주기가 시작될 때까지의 시간이 낭비되는 단점이 있다.

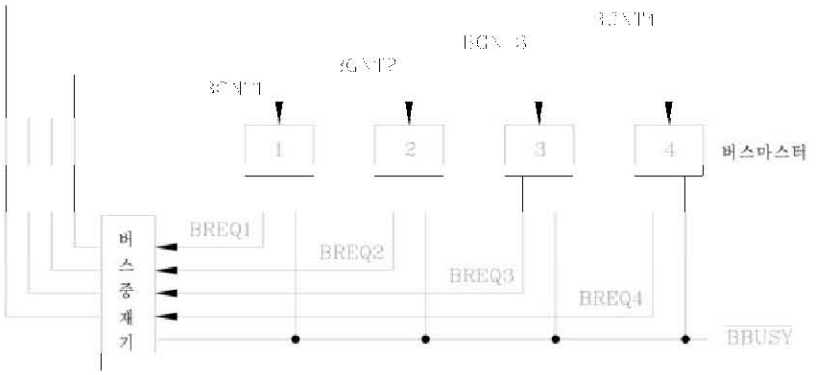
비동기식 버스에서는 각 버스 동작이 완료된 즉시, 연관된 다음 동작이 일어나므로 동기식 버스에서와 같이 낭비되는 시간은 없다. 그러나 이러한 연속적 동작을 처리하기 위한 인터페이스 회로가 복잡해지는 단점이 있다.

7.3 분산 중재 방식은 중앙집중식에 비하여 중재회로가 간단하기 때문에 동작 속도가 더 빠르고 신뢰도가 높은 장점이 있다. 신뢰도가 높아지는 이유는 어떤 중재기가 고장나더라도 해당 마스터에만 영향을 미치기 때문이다. 그러나 고장을 일으킨 중재기를 찾아내는 방법이 복잡해지는 점과 한 중재기의 고장이 전체 시스템의 동작에 영향을 줄 수도 있다. 예를 들면, 고장난 중재기가 버스 승인 신호를 잘못 발생시키는 경우에는 두 개의 마스터들이 동시에 버스를 사용하게 될 수도 있다.

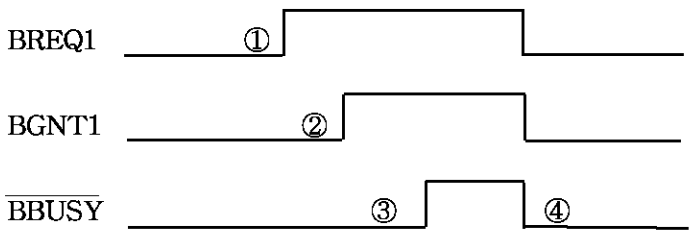
7.4 병렬 중재 방식은 버스 마스터와 같은 수의 버스 요구선과 승인 신호선을 필요로 하기 때문에 회로가 더 복잡해진다.

직렬 중재방식은 병렬 중재방식에 비하여 하드웨어가 더 간단하다는 장점이 있지만, 우선순위를 변경하기 어렵기 때문에 우선순위가 낮은 장치가 오랫동안 버스 사용을 할 수 없게 되는 단점이 있다.

7.5 (1)



(2)



(3) BBUSY 신호는 어떤 버스 마스터가 버스를 사용하고 있는 중임을 나타내므로, 버스 승인을 받은 마스터는 그 신호가 활성화되어 있는 동안에는 기다렸다가 비활성화되면 버스 사용을 시작한다. BBUSY 신호가 없다면 버스의 사용 허가를 받은 마스터1이 즉시 버스 사용을 시작하게 되고, 따라서 현재 버스를 사용하고 있는 마스터3의 데이터와 충돌이 일어나게 된다.

7.6 회전 우선순위 중재방식에는 (1) 현 중재 동작이 끝난 후에는 모든 마스터들의 우선순위가 한 단계씩 낮아지고 가장 우선순위가 낮았던 마스터가 최상위 우선순위를 가지도록 하는 방법과 (2) 일단 버스 사용 승인을 받은 마스터는 최하위 우선순위를 가지며 바로 다음에 위치한 마스터가 최상위 우선 순위를 가지도록 하는 방법이 있다. 두번째 방법을 acceptance-dependent식 회전 우선 순위 중재 방식이라고 한다.
(1)의 방법에서 버스를 빈번히 사용해야 하는 마스터가 존재하면 한번 버스를 사용한 후에는 모든 마스터들이 버스 사용을 마칠 때까지 기다려야 하므로 비효율적이다. 따라서 (2)의 방법을 사용하는 것이 더 효율적이다.

7.7 기억장치-사상 I/O 방식의 경우 : 주소가 12비트인 시스템에서 주소지정이 가능한 전체 기억 장소들의 수는 4096개이므로 I/O장치들에 할당되는 주소는 그 절반인 2048개이지만, 각 장치 당 두 개씩의 주소가 할당되므로 1024개의 I/O장치들을 접속할 수 있다.

분리형 I/O 방식 : I/O 주소 공간이 기억장치 주소 공간과는 별도로 지정되기 때문에 4096개의 주소가 할당되며, 각 장치 당 두 개씩의 주소가 할당되므로 2048개의 I/O장치들을 접속할 수 있다.

7.8 기억장치-사상 I/O 방식은 관련 레지스터들에 대하여 읽기 및 쓰기를 할 때도 그들을 기억장치와 동일하게 취급하여, 기억장치 액세스에 사용하는 명령어들을 사용할 수 있다는 장점과 I/O장치들에 대한 주소로서 기억장치 주소 공간의 일부를 할당하기 때문에 기억장치를 위한 주소 공간이 그만큼 감소하게 된다는 단점이 있다.

분리형 I/O 방식에서는 기억장치 주소 공간이 I/O 때문에 줄어들지 않는 다는 장점이 있지만, I/O 관련 명령어들만 이용해야 하기 때문에 프로그래밍이 불편해지는 단점이 있다.

7.9 (1) 데이지-체인 방식
(2) 소프트웨어 폴링 방식
(3) 다중-인터럽트 선들을 사용하는 방식

7.10 우선순위가 높은 I/O 장치들이 인터럽트를 빈번히 요구하는 경우에는 우선순위가 아주 낮은 I/O 장치에게는 인터럽트 승인 신호가 전달되지 않게 되므로 기근(starvation)이 발생할 가능성이 있다.

7.11 각 I/O 동작을 위해서 시스템 버스를 두 번씩 사용해야 한다. 따라서 I/O 동작을 위한 시스템 버스 사용 빈도가 너무 높아져서 전체 시스템 성능이 저하되는 문제점이 있다.

7.12
(1) 하나의 DMA 제어기가 모든 I/O 동작을 관장해야 하므로 병목 현상이 발생할 수 있다.
(2) I/O 장치들의 속도와 제어 방법이 서로 다르면 DMA 제어기의 내부 구조가 매우 복잡해진다. 또한 여러 개의 I/O 버스 및 인터페이스 회로들이 필요하게 된다.