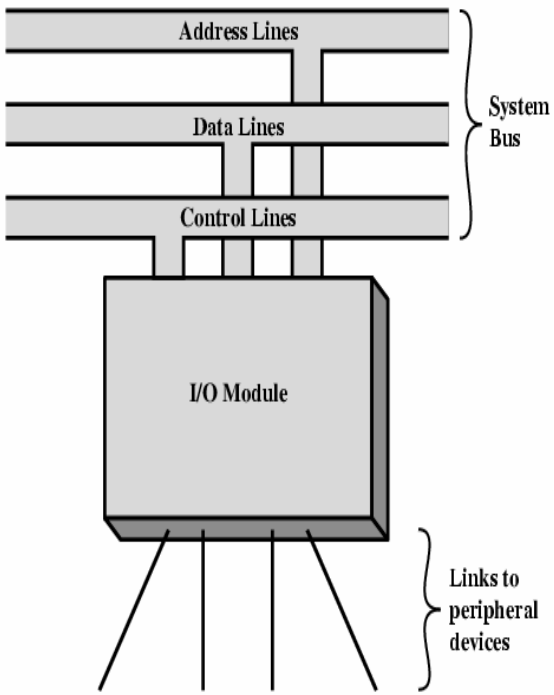
7.1 외부 장치들



I/O module의 배치

I/O Module

주변장치와 컴퓨터간의 데이터를 교환하는 수단을 제공하는 장치 (시스템 버스에 연결되고, 버스와의 통신기능을 수행한다.

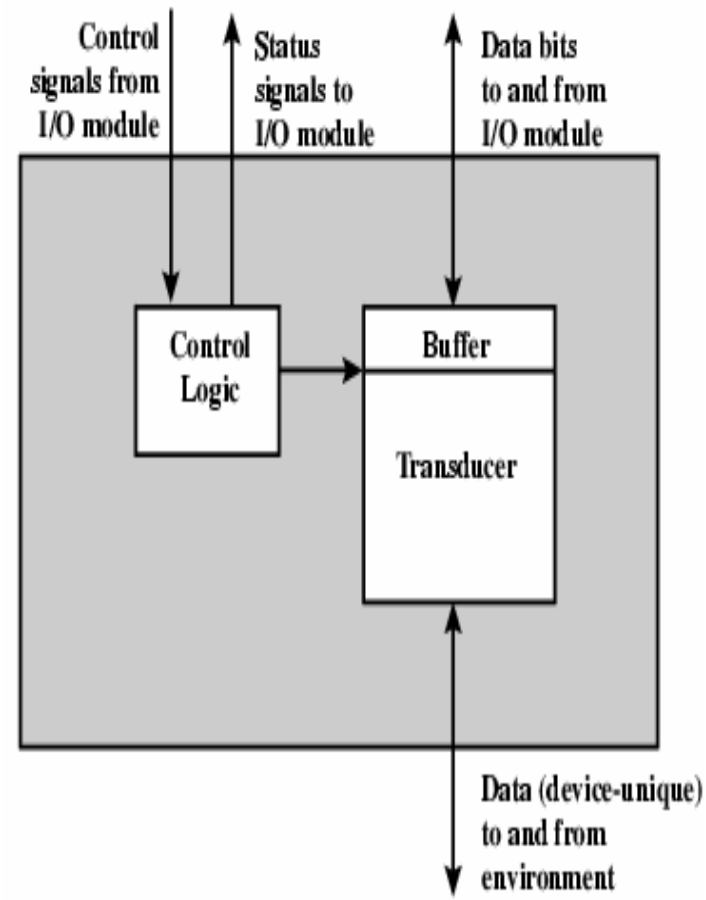
* 외부 장치를 시스템 버스에 직접 연결하지 않는 이유

- 주변장치의 다양성 :
  주변장치 제어회로가 다양하며 이를 프로세서 내부에 두는 것은 비효율적이다.

- 주변장치 데이터 전송율의 다양성 : 속도의 차이

- 프로세서와는 다른 데이터 형식을 사용 :
  메모리-워드, 외부장치-바이트를 사용.

7.1 외부 장치와의 연결



외부장치 블럭도

- 외부(주변)장치와의 연결

- 데이터신호 : bits 들의 전송

- 제어신호 : 외부장치가 수행할 동작지시
  (input, output, read, write)
  I/O module -> 각 장치의 제어회로

- 상태신호 : 주변장치의 상태보고
- Transducer : 전기적신호를 데이터신호로 변환하는 역할.
  (전압, 전류 값--- 비트열로 변환)

* 데이터신호만 쌍방향성을 가진다.

7.1 외부 장치와의 연결

Table 7.1 The International Reference Alphabet (IRA)

bit position															
b_7				0	0	0	0	1	1	1	1				
b_6				0	0	1	1	0	0	1	1				
b_5				0	1	0	1	0	1	0	1				
b_4	b_3	b_2	b_1												
0	0	0	0	NUL	DLE	SP	0	@	P	‘	p				
0	0	0	1	SOH	DC1	!	1	A	Q	a	q				
0	0	1	0	STX	DC2	”	2	B	R	b	r				
0	0	1	1	ETX	DC3	#	3	C	S	c	s				
0	1	0	0	EOT	DC4	$	4	D	T	d	t				
0	1	0	1	ENQ	NAK	%	5	E	U	e	u				
0	1	1	0	ACK	SYN	&	6	F	V	f	v				
0	1	1	1	BEL	ETB	’	7	G	W	g	w				
1	0	0	0	BS	CAN	(	8	H	X	h	x				
1	0	0	1	HT	EM	)	9	I	Y	i	y				
1	0	1	0	LF	SUB	*	:	J	Z	j	z				
1	0	1	1	VT	ESC	+	;	K	[	k	{				
1	1	0	0	FF	FS	,	<	L	\	l	|				
1	1	0	1	CR	GS	-	=	M	]	m	}				
1	1	1	0	SO	RS	.	>	N	^	n	~				
1	1	1	1	SI	US	/	?	O	_	o	DEL				

IRA 문자표

- 키보드 연결의 경우

- 키보드터치 – 컴퓨터연결 - 모니터표시
- 데이터 교환의 기본 단위 : 문자
- 문자는 7-8 bit로 표시 (8bit 는 패리티)
  IRA (international reference Alphabet)
  ASCII 는 IRA의 미국 버전 (128문자)

- 동작 :
  키 입력 – 전기신호전송 – transducer해석
  - bit pattern으로 변화 – I/O module로 전송

7.2 I/O module

- 외부(주변)장치에서 프로세서로 데이터가 전송

① 프로세서는 I/O module 에 장치의 상태를 검사하도록 요청.

② I/O module은 장치의 상태를 점검하고 이를 보고.

③ 외부장치의 작동/전송이 가능하면, 프로세서는 I/O module에 데이터 전송을 요구.

④ I/O module 은 데이터 전송을 외부장치로 요구하고 데이터를 받는다.

⑤ 데이터가 I/O module에서 프로세서로 이동

7.2 I/O module

- I/O module의 기능

① **명령해독(command decoding)** : 제어버스상의 신호
프로세서로부터 주어진 명령어를 해석하고 동작지시

② **데이터 교환** : 외부장치로부터 메인 메모리로 혹은 외부장치간 통신

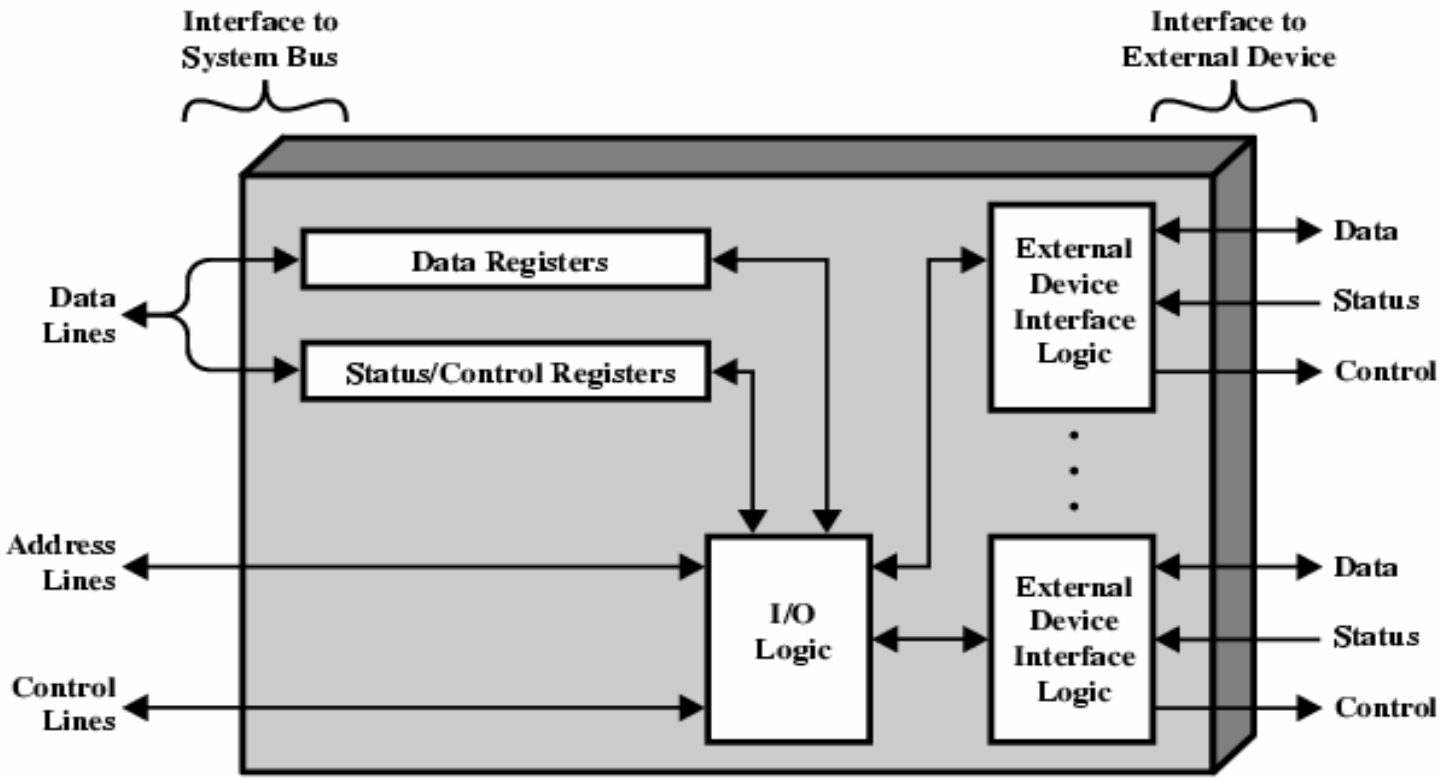
③ **상태보고** (status report) : Busy/Ready상태 및 장치결함상태를 보고

④ **주소인식** : 각 I/O 장치들의 주소를 인식

⑤ **데이터버퍼링** : 주기억장치-외부장치간의 전송속도 차이를 조절

⑥ **오류보고 및 검출** : 장치의 기계적/전기적 오 동작, 비트 오류 검출/보고

7.2 I/O module 의 조직



데이터 레지스터 : 데이터를 임시저장 / 데이터 버퍼
상태/제어 레지스터 : 현재 외부장치 상태 저장 / 제어정보수신을 위해 사용
입출력논리 : 프로세서의 명령에 따라 입출력 모듈을 제어
외부장치 연결을 위한 주소 인식
장치인터페이스 : 연결되어있는 각 장치와의 연결 요소

7.2 I/O module

- I/O module의 기능

I/O 모듈은 프로세서를 대신하여 입출력과 관련된 일을 수행한다.

I/O 가 대신 처리하는 일의 분량에 따라 프로세서의 효율성은 높아 진다.
이때 프로세서는 간단한 명령만을 내리고 명령에 따른 세부적인 수행은
I/O 모듈이 처리 할 수 있다.

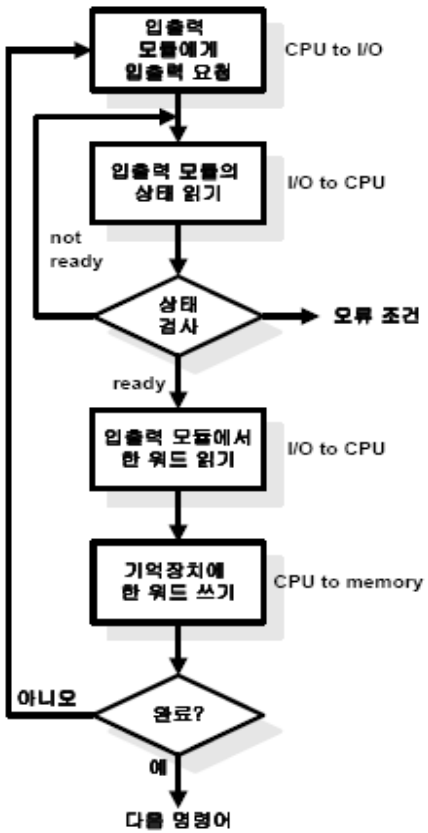
- I/O 모듈이 대부분의 일을 처리하는 경우 I/O module 을 IOP (I/O processor, 혹은 I/O channel 이라고 한다)

- 매우 기본적인 동작만을 I/O module 이 처리하고 프로세서가 대부분의 작업을 수행하는 경우, I/O module은 장치제어기로 불린다.

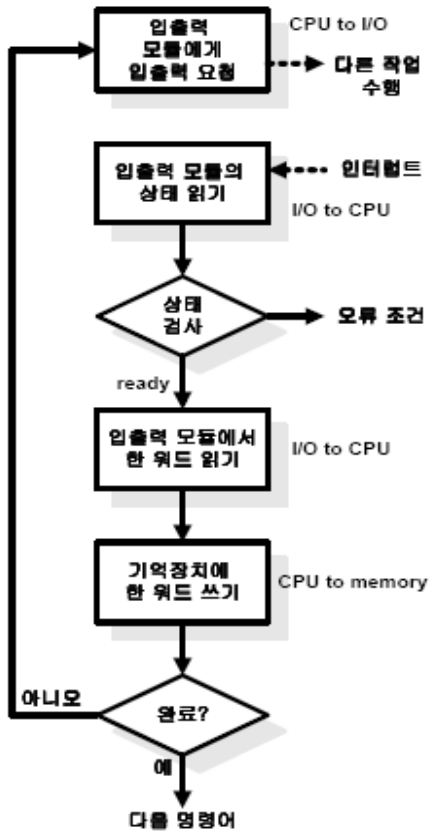
7.3 Program I/O

- CPU 제어 입출력

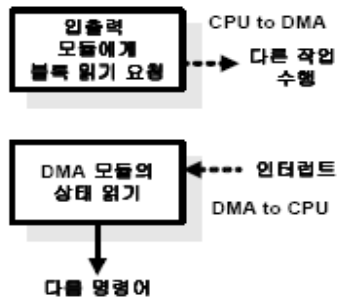
프로세서가 입출력 명령어를 처리하는 방식에는 Program I/O, 인터럽트 구동I/O, DMA I/O 이 있다.



(a) 프로그램된 입출력



(b) 인터럽트 기반 입출력



(b) DMA 방식

7.3 Program I/O

- Program I/O

- 프로세서가 프로그램 수행 중 I/O 명령어를 만나면, 해당 I/O에게 명령을 보내 이를 수행하게 한다.

- I/O module은 요구된 동작을 수행 후 그 결과를 상태 및 데이터 레지스터에 저장한다.

- I/O module은 이때 독립적으로 동작하고 프로세서에는 영향을 주지 않는다. (인터럽트를 발생시키지 않는다)

- 프로세서는 언제 I/O동작이 끝나는지 알지 못하므로, 주기적으로 I/O 모듈의 상태를 점검한다.

7.3 Program I/O

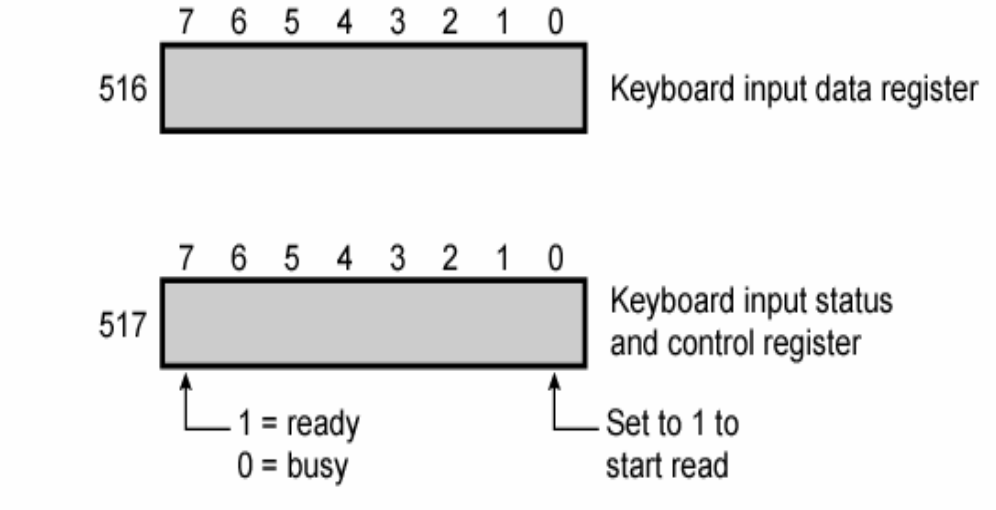
- 주소 지정 방식

- 프로세서가 입출력 명령어를 수행할 때 주소 정보도 I/O 모듈에 전송
  각 I/O 모듈은 주소정보를 분석하여 자신에 해당되는지를 파악.
  I/O 장치의 주소 지정방식은 다음의 두 가지가 있다.

①. 기억장치 사상방식 (Memory-mapped I/O)
  - 주기억장치와 입출력장치는 모두 단일 주소 공간을 사용
    (물리적으로는 떨어져 있는 장치지만 주소상으론 같은 공간)
  - 프로세서는 I/O module 레지스터를 주기억장치 위치와 동일하게 취급
    (I/O module 레지스터 내용을 메인 메모리의 워드로 취급)
  - 따라서 동일한 명령어 (store/load)로 I/O 쓰기, 읽기 처리

②. 고립형 (Isolated)
  - 주기억장치와 I/O module은 서로 다른 주소 공간을 사용.
  - 주소버스는 공유하지만, 다른 명령어 사용으로 별도의 제어버스 사용.

7.3 Program I/O



ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

7.4 인터럽트 구동 I/O

- 인터럽트 구동 I/O

  프로그램 I/O의 문제점 : I/O module이 데이터를 송수신 할 준비가 될 때 까지 프로세서는 기다려야 한다는 것.
  이 문제점을 개선 한 것이 인터럽트 구동 I/O 이다.

동작 >

- 프로세서는 I/O module 에 입출력 관련 명령을 내리고 다른 일 수행.
- I/O module은 I/O 동작이 끝나면 인터럽트를 프로세서에 보낸다.
- 프로세서는 현재 수행 중인 명령을 끝내고 인터럽트 확인 신호를 보낸다.
- 프로세서는 현재 수행 중인 명령어의 위치와 상태 및 레지스터 내용을 시스템스텍에 보관한다.
- 프로세서는 처리할 첫째 인터럽트 명령어 주소를 프로그램 카운터 레지스터에 적재하고 인터럽트 루틴을 처리한다.
- 인터럽트 처리 완료 후 프로세서는 시스템스텍의 내용을 복구하고 다음의 명령어들을 처리 한다.

7.4 설계 이슈

- 인터럽트를 발생시킨 장치의 구분
  - 어떤 입출력 모듈에서 인터럽트를 발생시켰는지를 알아야 한다.

방식>

①. 다중인터럽트 선 : 모듈별로 전송선을 설정-> 쉬운 해결/비실용적

②.소프트웨어 폴 : 범용인터럽트 처리루틴을 실행-> 어떤장치가 인터럽트를 요구했는지 질의 하고 I/O module 은 이에 응답.

③. 하드웨어 폴 : 각 입출력 모듈은 공통된 인터럽트 요청/확인선으로 연결
  - 인터럽트 발생모듈은 데이터버스에 특정워드 적재
  - 이 특정워드(모듈의 고유번호나 주소)를 벡터라고 함.

④. 버스 중재 방식 : 입출력 모듈은 인터럽트 전달 전에 버스의 사용권한을 얻어야 한다.
  - 이 사용권한을 요구하는 신호로부터 모듈을 식별가능

7.4 설계 이슈

- 다중 인터럽트의 처리
  - 동시에 여러 인터럽트가 요구될 때 우선 순위를 결정해야 한다.

방식>

①. 다중인터럽트 선 : 가장 우선 순위가 높은 I/O 장치를 선택

②.소프트웨어 폴 : 우선 순위가 높은 순부터 인터럽트 요구했는지를 문의

③. 하드웨어 폴 : I/O module 의 직렬연결 된 순서가 순위를 결정
(데이지체인)

④. 버스 중재 방식 : 버스 중재기가 우선 순위를 결정한다.

7.5 직접기억장치 억세스(DMA)

- 인터럽트 기반 방식의 문제점
  - 입출력 전송률이 프로세서가 장치를 검사하고 서비스하는 속도에 영향을 받는다. 또한 프로세서가 입출력 전송을 위하여 많은 시간을 소비한다. (프로세서의 데이터 전송 시 직간접적인 간섭에 의함)

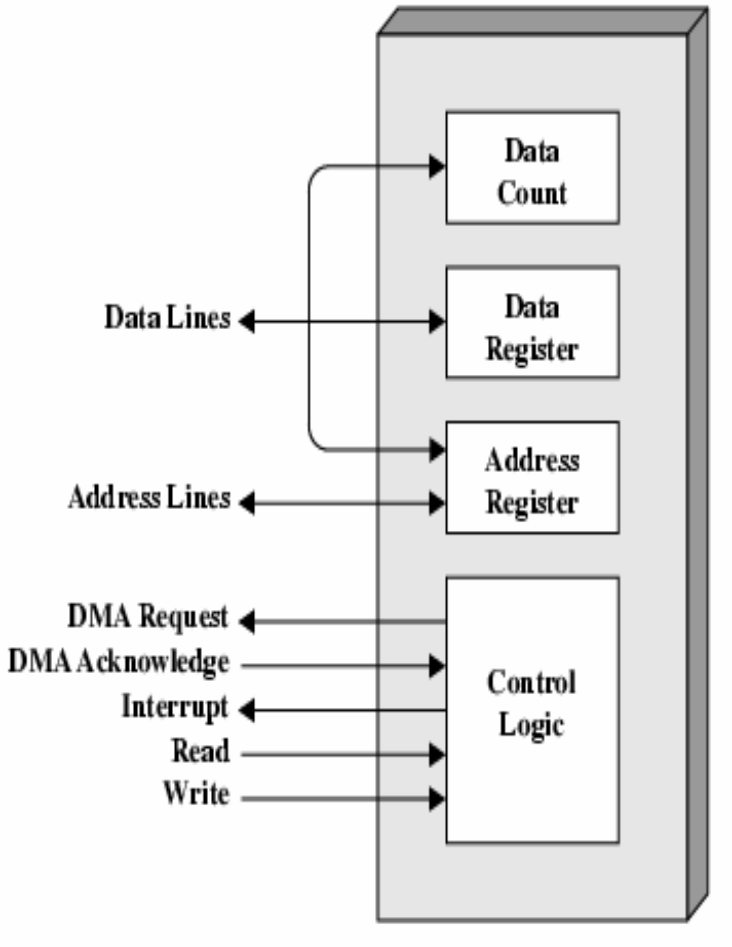
- DMA 방식

  프로세서의 관여 없이 입출력 모듈이 직접 주 기억장치와 데이터를 교환

  - DMA 를 사용하기 위해서는 시스템 버스에 DMA 모듈을 추가적으로 설치해야 한다.

  - DMA 모듈은 프로세서가 버스를 사용하지 않을 때 또는 버스를 잠시 사용하지 못하도록 하고 대신 버스를 사용한다.
    (보통은 후자의 방법을 사용하며 이를 사이클 훔치기라고 한다.

7.5 직접기억장치 억세스(DMA)



DMA 모듈의 구조

7.5 직접기억장치 억세스(DMA)

- DMA 방식에서 데이터 읽기/쓰기

  - 읽기/쓰기 동작을 위한 프로세서의 명령 (to DMA)

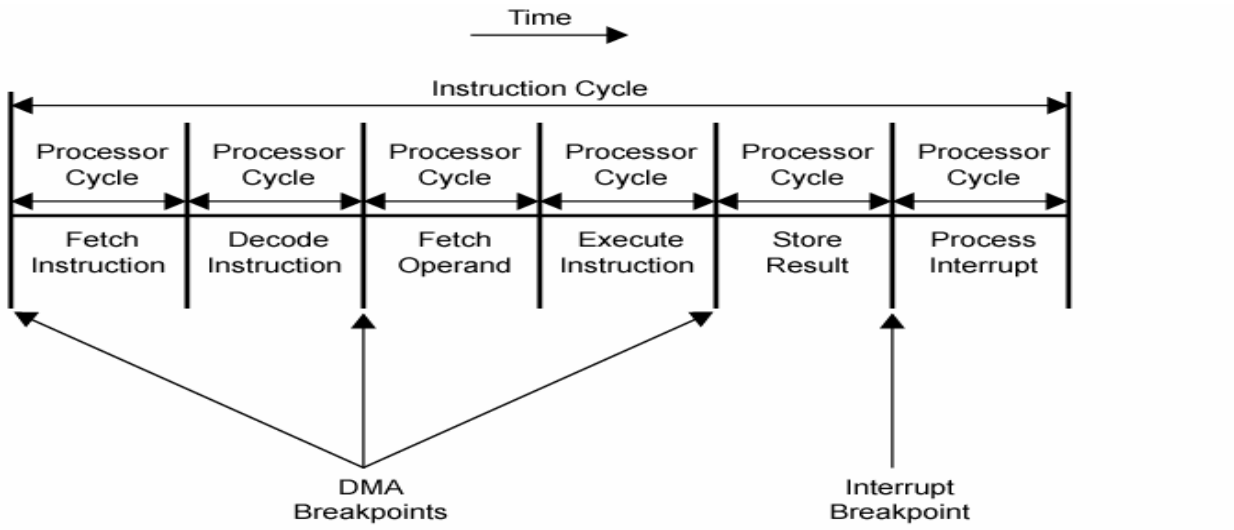
  ①. 프로세서와 DMA 간에 제어선 을 통해 읽기/쓰기 요청
  ②. 입출력 장치를 식별하는 주소를 데이터 버스선을 통해 전달한다.
  ③. 데이터가 읽혀지거나 쓰여질 기억장치 부분의 시작 주소를
  데이터 버스를 통해 DMA 모듈내의 주소레지스터에 전달한다.
  ④. 읽을 또는 쓸 워드의 수도 데이터 버스선을 통하여 보내서
  데이터 계수 레지스트에 저장된다.

  - 위의 명령을 전달 후 프로세서는 다른 작업을 수행
  - DMA 는 독자적으로 명령을 수행 후 데이터를 이동시킨다.
  - 이동이 끝나면 인터럽트를 프로세서에 보낸다.

  결과적으로 프로세서는 동작의 초기와 끝에만 관여

7.5 직접기억장치 억세스(DMA)

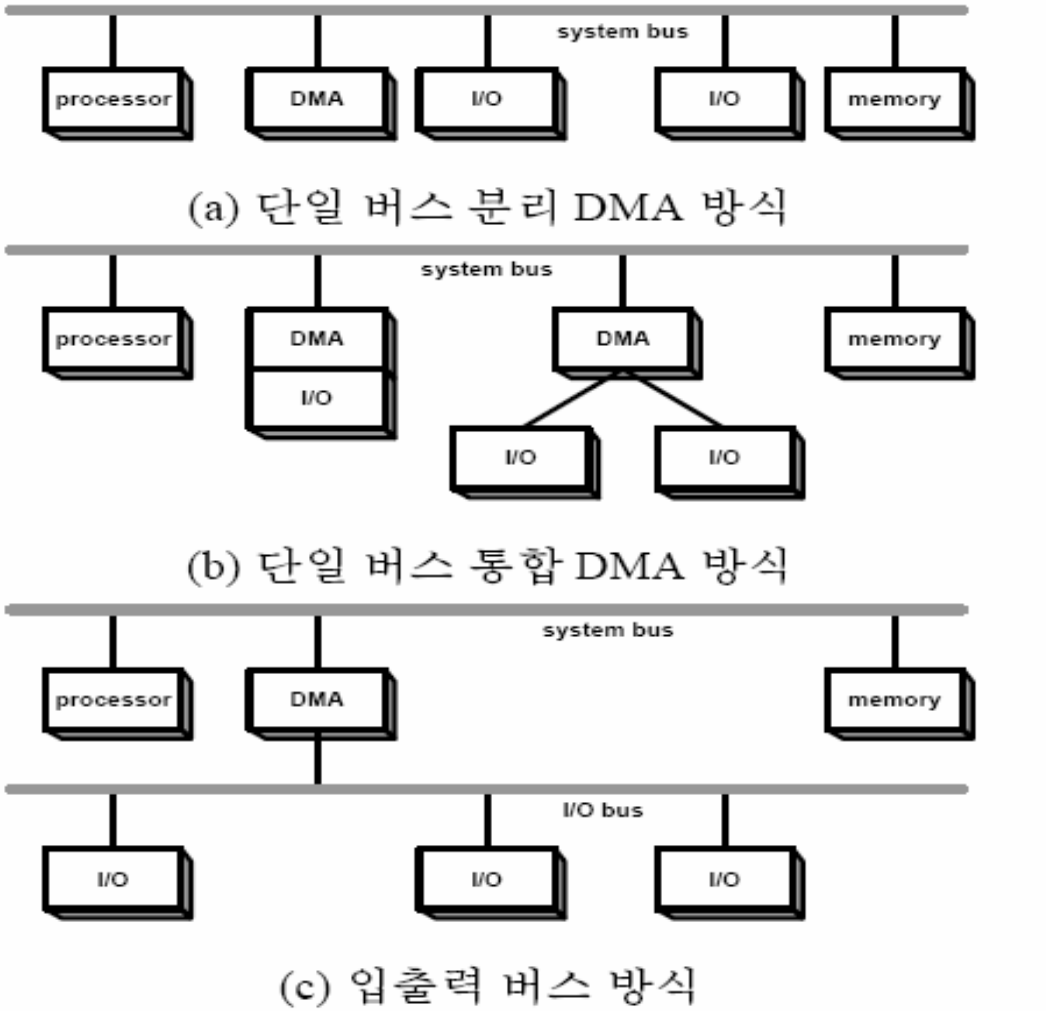
- DMA가 프로세서를 잠시 중단 시킬 수 있는 위치.



- DMA가 프로세서를 잠시 중단시킬 때 프로세서는 상태정보 보관을 필요로 하지 않는다.
- 프로세서는 단순히 한 클럭 주기동안 쉰다. 이 클럭 주기 동안 DMA는 한 워드를 전송한다. 따라서 프로세서의 처리속도는 느려지지만 프로그램 I/O 나 인터럽트 방식에 비해서는 효율적이다.

7.5 직접기억장치 억세스(DMA)

- DMA 구성 방식의 종류



7.5 직접기억장치 억세스(DMA)

- DMA 구성 방식의 종류

(a) 단일버스분리 방식
- 모든 모듈이 공통된 시스템 버스를 공유
- 프로그램 I/O 사용
- 비용은 저렴/효율성은 감소
  (한 워드전송에 두 번의 버스사용이 필요)

(b) 단일버스통합 방식
- DMA 모듈과 몇 개의 I/O 모듈 간에 전용선으로 통신하는 방식
- DMA 모듈을 입출력 모듈과 결합하여 사용할 수 있다.
- DMA 모듈은 I/O 모듈의 일부분으로 혹은 독립적으로 사용.
  (한 워드전송에 한 번의 버스사용이 필요)

(c) 입출력 버스 방식
- DMA 모듈에 포함해야 하는 인터페이스 수를 하나로 줄일 수 있으며 확장이 용이하다.
- I/O 버스를 이용하여 DMA와 I/O 간의 데이터 교환이 이루어진다.

7.6 I/O 채널과 프로세서

- 입출력 기능의 발전 단계

단계-1. CPU가 직접 주변장치를 제어

단계-2. 입출력 모듈을 이용한 프로그램 I/O 방식 사용.

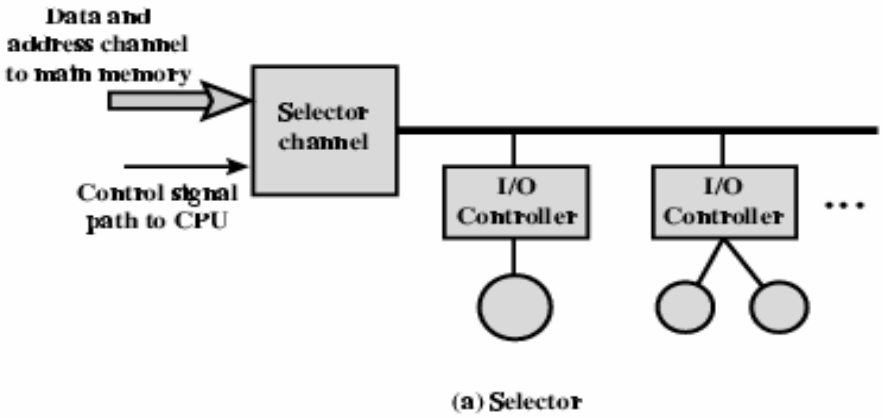
단계-3. 입출력 모듈을 이용한 인터럽트 기반 I/O 방식 사용.

단계-4. DMA 방식 사용.

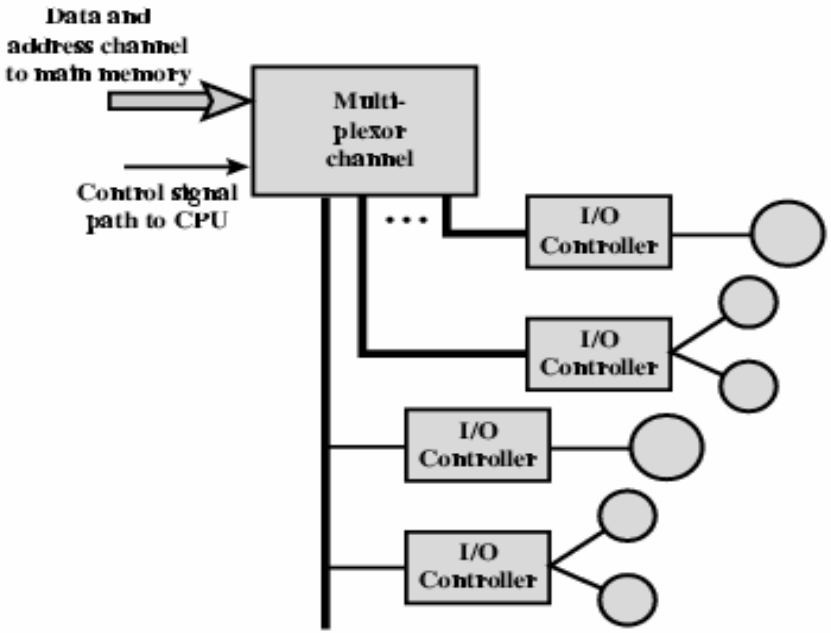
단계-5. 입출력 모듈에 강력한 프로세서 탑재. → I/O 채널로 불림.
-주 기억장치로부터 명령어를 직접 인출하여 실행.
(프로세서는 I/O 프로그램 수행만 지시)

단계-6. 입출력 모듈에 내부 기억장치를 포함. (단일 컴퓨터로써 동작)
-> I/O 프로세서로 통칭.

7.6 I/O 채널과 프로세서



(a) Selector



(b) Multiplexor

- 입출력 채널의 구성 방법
  - I/O 채널은 I/O 동작에 대하여 독립적인 제어권을 가진다.
  - I/O 명령어 들은 주 기억장치에 보관되며, I/O 채널에 있는 프로세서에 의하여 동작된다.

(a) 선택기 채널 방식
- 다중 I/O 제어기와 직렬연결
한번에 한 개의 I/O 제어기와 통신.

(b) 멀티플렉서 채널방식
- 다중 I/O 제어기와 병렬연결
여러장치를 동시에 처리하여 준다.

7.7 외부 인터페이스

- 입출력 모듈과 주변장치간의 인터페이스

- 병렬인터페이스 : 여러선을 이용하여 데이터를 교환 (프린터)
- 직렬인터페이스 : 단일선을 이용하여 데이터를 교환 (마우스)

- 전통적으로 병렬인터페이스는 고속의 주변장치와 연결
  직렬인터페이스는 저속의 주변장치와 연결

- 최근에는 USB, fire-wire 와 같은 고속의 직렬 인터페이스가 등장

- 입출력 모듈은 그 내부의 버퍼를 이용하여 시스템과 주변장치간의 속도 차이를 극복하여 준다.

- 입출력 모듈과 주변장치를 연결하는 방법
  일대일 : 입출력 모듈과 주변장치간의 전용선을 이용하는 방식
  일대다 : 버스와 같은 형태로 여러주변장치가 하나의 모듈에 연결
  (USB, fire-wire)