Chapter.7 Input / Output

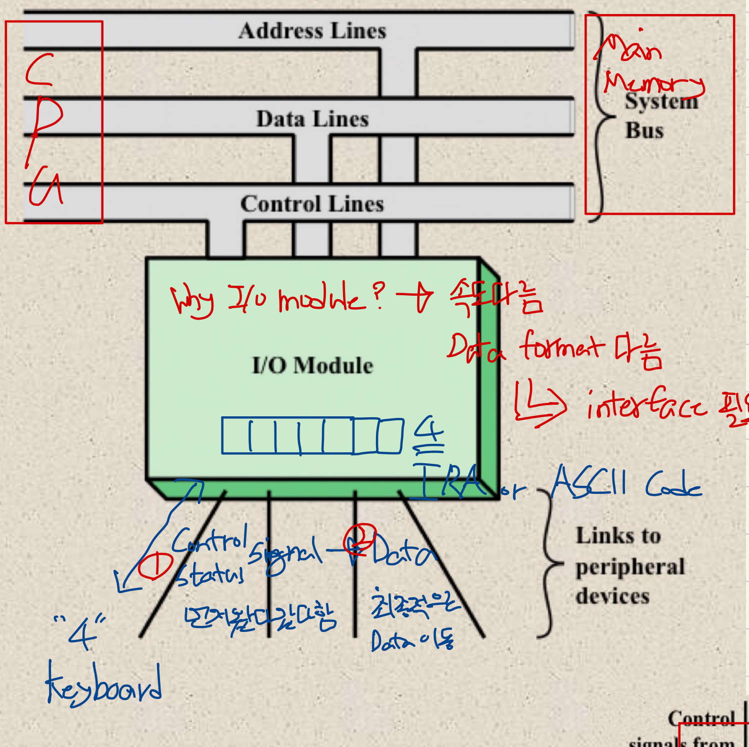
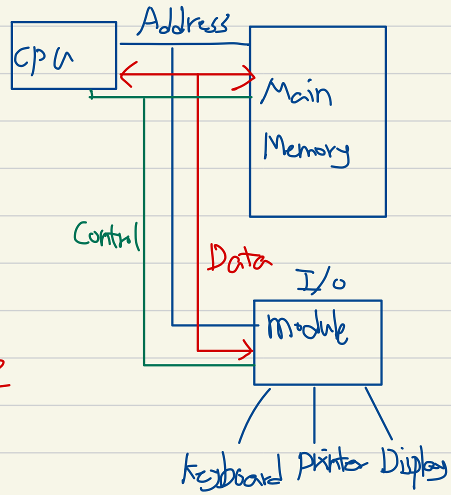

Address Lines
Data Lines
Control Lines
CPU
Main Memory
System Bus
I/O Module
Why I/O module? → 속도다름
Data format 다름
↳ interface 필요
4
IRA or ASCII Code
Control
Status
signal → Data
①
②
먼저왔다갔다함
최종적으로 Data 이동
Links to peripheral devices
"4"
keyboard

Figure 7.1 Generic Model of an I/O Module


CPU
Address
Main Memory
Control
Data
I/O module
Keyboard
Printer
Display

External Devices
- Human readable
  keyboard, Display
- Machine readable
  외장 하드, SSD, Sensor
- Communication
  wifi, 이더넷

IRA (International Reference Alphabet)
- Printable : 문자, 숫자, 특수기호
- Control : 저장, 뒤로가기, 리턴

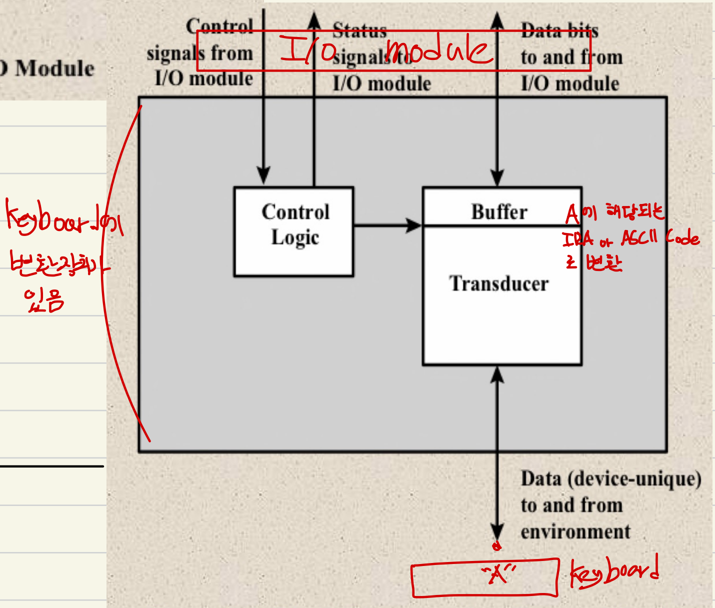

Control signals from I/O module
Status signals to I/O module
Data bits to and from I/O module
I/O module
Control Logic
Buffer
Transducer
A에 해당되는 IRA or ASCII Code로 변환
keyboard에 변환장치가 있음
Data (device-unique) to and from environment
"A"
keyboard

Figure 7.2 Block Diagram of an External Device

I/O module의 기능들

- Control & timing
- processor communication
- Device communication
- Data buffering
- Error detection

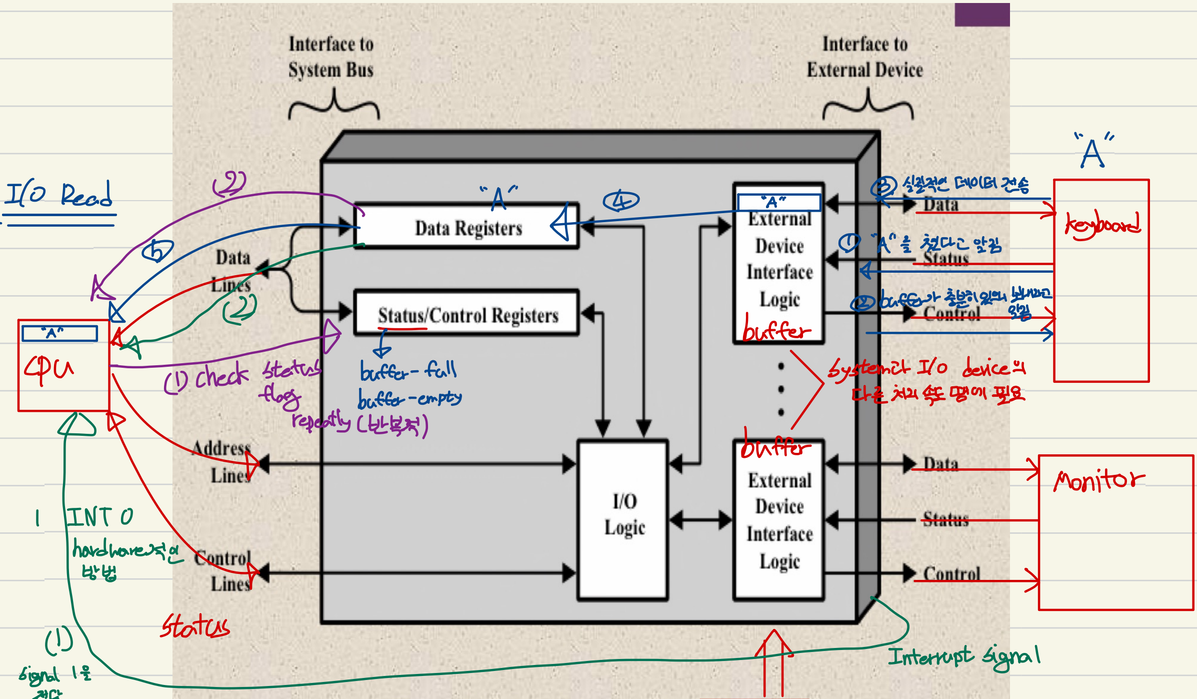
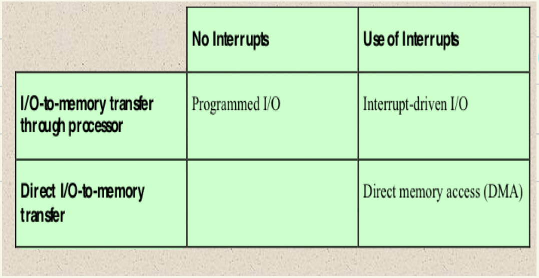

Interface to System Bus
Interface to External Device
I/O Read
CPU
"A"
Data Registers
"A"
Status/Control Registers
Data Lines
Address Lines
Control Lines
I/O Logic
External Device Interface Logic
External Device Interface Logic
Data
Status
Control
Data
Status
Control
"A"
keyboard
Monitor
(2)
⑤
④
(2)
① "A"를 쳤다고 알림
② buffer가 충분히 있으니 보내라고 알림
③ 실질적인 데이터 전송
(1) check status flag repeatly (반복적)
buffer-full
buffer-empty
buffer
buffer
system과 I/O device의 다른 처리 속도 땜에 필요
1
INT O
hardware적인 방법
Status
(1)
signal 1을 전달
Interrupt signal

Figure 7.3 Block Diagram of an I/O Module

CPU는 Data Register에 값이 있는지 어떻게 아는가?

↳ Status가 buffer-full or buffer-empty로 구분 → I/O module의 Status를 CPU가 어떻게 인지?



	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

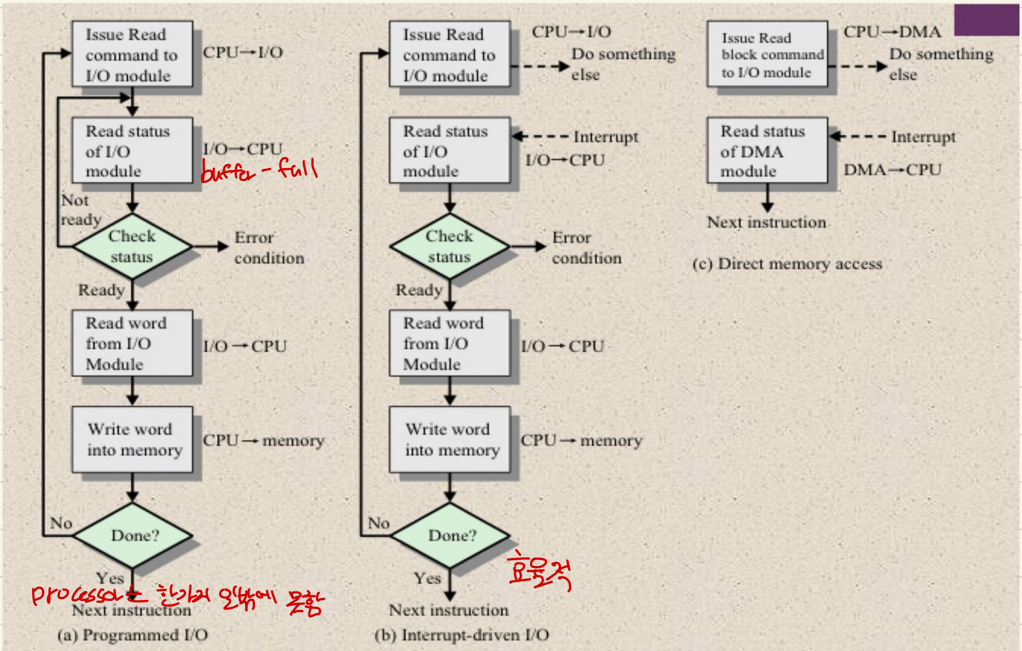
① Programmed I/O (SW) - CPU가 직접 상태를 읽는다.

② Interrupt I/O (HW)

③ Direct memory access (DMA)

I/O Commands (processor에 의해 받는 명령어)

- Control : CPU가 I/O module에게 명령을 주는 것
- Test : I/O module의 상태를 체크하는 것
- Read : I/O module의 data를 가져오는 것
- Write : I/O module에 data를 쓰는 것


Issue Read command to I/O module
CPU→I/O
Read status of I/O module
I/O→CPU
Not ready
Check status
Error condition
Ready
Read word from I/O Module
I/O→CPU
Write word into memory
CPU→memory
No
Done?
Yes
Next instruction
(a) Programmed I/O
Issue Read command to I/O module
CPU→I/O
Do something else
Read status of I/O module
Interrupt
I/O→CPU
Check status
Error condition
Ready
Read word from I/O Module
I/O→CPU
Write word into memory
CPU→memory
No
Done?
Yes
Next instruction
(b) Interrupt-driven I/O
Issue Read block command to I/O module
CPU→DMA
Do something else
Read status of DMA module
Interrupt
DMA→CPU
Next instruction
(c) Direct memory access

buffer - full

processor는 한가지 일밖에 못함

효율적

Figure 7.4 Three Techniques for Input of a Block of Data

I/O mapping Summary

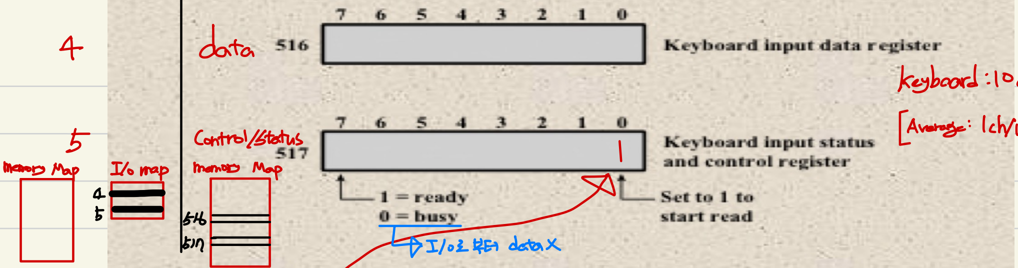
- Memory mapped I/O

메모리 번지수 처럼 쓰는 것

- Isolated I/O

I/O.1, I/O.2, I/O3

Isolated I/O | memory mapped I/O


4
5
memory Map
I/o map
4
5
data
516
Control/Status
517
memory Map
516
517
7 6 5 4 3 2 1 0
Keyboard input data register
7 6 5 4 3 2 1 0
1
Keyboard input status and control register
1 = ready
0 = busy
Set to 1 to start read
I/O로 부터 data X

Keyboard : 10 chars/second
[Average : 1ch/100ms] (1s = 1000ms)

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

control

Programmed I/O

상태를 점검 (S/w적으로)

(a) Memory-mapped I/O

다음

(memory 번지)

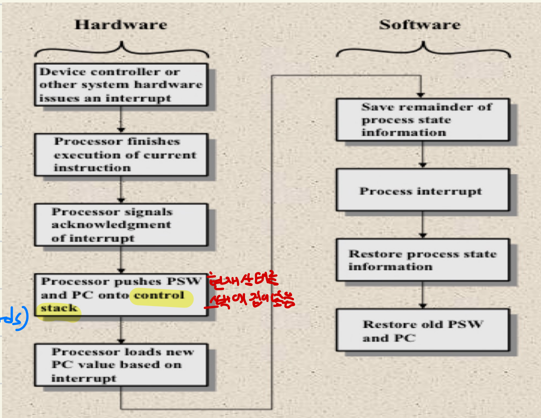
ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201 memory 번지	Loop until complete
	In	4	Load data byte

(b) Isolated I/O

Figure 7.5 Memory-Mapped and Isolated I/O

Interrupt I/O

- I/o module이 processor한테 Interrupt를 걸음 (H/w적)
- data를 가져온 후에는 processor는 그 전에 하던일을 계속함


Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
현재상태를 스택에 집어넣음
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC

PSW (program state words)

Figure 7.6 Simple Interrupt Processing

Stack [Last In-First-Out]

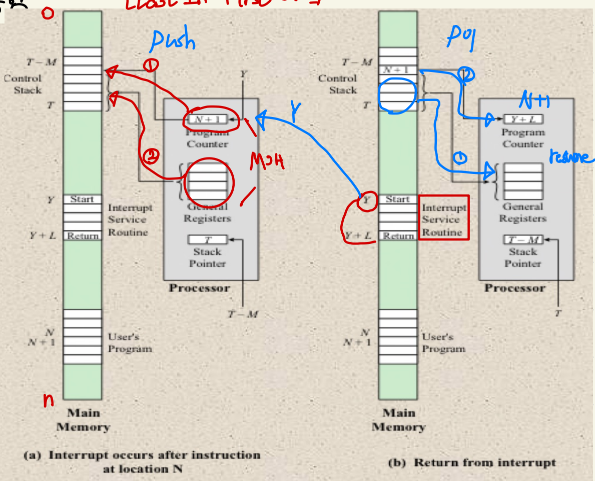
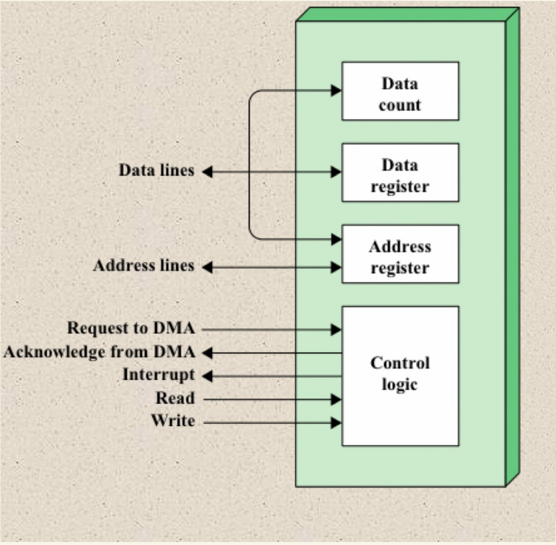

0
Push
T − M
Control Stack
T
①
N + 1
Program Counter
Y
②
M개
General Registers
T
Stack Pointer
Processor
T − M
Y
Start
Y + L
Return
Interrupt Service Routine
N
N + 1
User's Program
n
Main Memory
(a) Interrupt occurs after instruction at location N
Pop
T − M
N + 1
Control Stack
T
②
N+1
Y + L
Program Counter
①
return
Y
Start
Y + L
Return
Interrupt Service Routine
General Registers
T − M
Stack Pointer
Processor
T
N
N + 1
User's Program
Main Memory
(b) Return from interrupt

Figure 7.7 Changes in Memory and Registers for an Interrupt

Direct Memory Access (DMA)

CPU는 참여시키지 X


Data count
Data register
Address register
Control logic
Data lines
Address lines
Request to DMA
Acknowledge from DMA
Interrupt
Read
Write

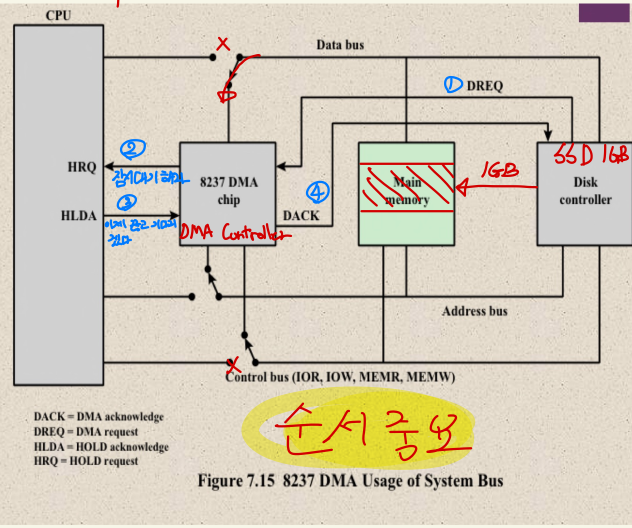

CPU
HRQ
HLDA
Data bus
① DREQ
② 잠시대기 해라
③ 이제 끊고 기다리겠다
④
8237 DMA chip
DMA Controller
DACK
Main memory
Disk controller
SSD 1GB
1GB
Address bus
Control bus (IOR, IOW, MEMR, MEMW)
DACK = DMA acknowledge
DREQ = DMA request
HLDA = HOLD acknowledge
HRQ = HOLD request
순서 중요

Figure 7.15 8237 DMA Usage of System Bus