

■ 7-1 Peripheral Devices

◆ I/O Subsystem

- Provides an efficient mode of communication between the central system and the outside environment

◆ Peripheral (or I/O Device)

- Input or Output devices attached to the computer
 - » Monitor (*Visual Output Device*) : CRT, LCD
 - » KBD (*Input Device*) : light pen, mouse, touch screen, joy stick, digitizer
 - » Printer (*Hard Copy Device*) : Dot matrix (*impact*), thermal, ink jet, laser (*non-impact*)
 - » Storage Device : Magnetic tape, magnetic disk, optical disk

◆ ASCII (*American Standard Code for Information Interchange*) Alphanumeric Characters

- I/O communications are usually involved in the transfer of ASCII information
- ASCII Code : Tab. 11-1

■ 7-2 Input-Output Interface

◆ Interface

- 1) A conversion of signal values may be required

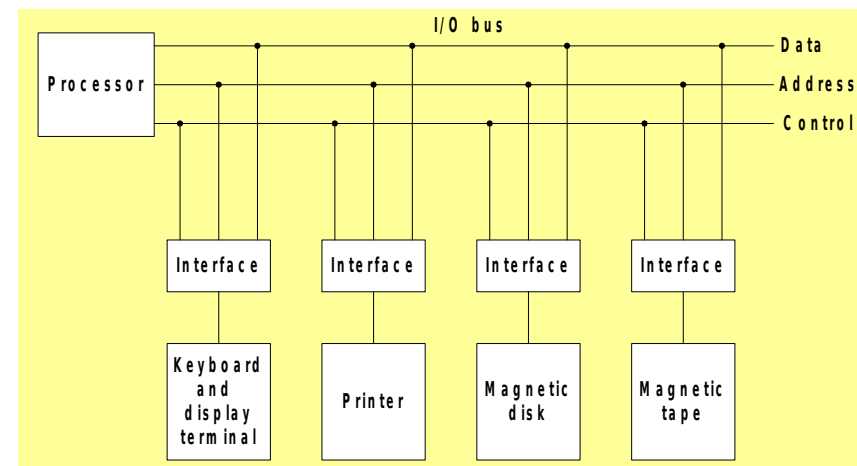
- 2) A synchronization mechanism may be needed
 - » The data transfer rate of peripherals is usually slower than the transfer rate of the CPU
- 3) Data codes and formats in peripherals differ from the word format in the CPU and Memory
- 4) The operating modes of peripherals are different from each other
 - » Each peripherals must be controlled so as not to disturb the operation of other peripherals connected to the CPU

◆ Interface

- Special hardware components between the CPU and peripherals
- Supervise and Synchronize all input and output transfers

◆ I/O Bus and Interface Modules : *Fig. 11-1*

- I/O Bus
 - » Data lines
 - » Address lines
 - » Control lines
- Interface Modules :
 - » SCSI (Small Computer System Interface)
 - » IDE (Integrated Device Electronics)
 - » Centronics
 - » RS-232
 - » IEEE-488 (GPIB)



- I/O command : 8251 SIO

- » Control Command
- » Status Command
- » Input Command
- » Output Command

- ◆ I/O Bus versus Memory Bus

- Computer buses can be used to communicate with memory and I/O

- » 1) Use two separate buses, one for memory and the other for I/O : Fig. 11-19, p. 421

- I/O Processor

- » 2) Use one common bus for both memory and I/O but have separate control lines for each : **Isolated I/O** or **I/O Mapped I/O**

- **IN, OUT** : I/O Instruction
- **MOV** or **LD** : Memory read/write Instruction

*** Control Lines**

I/O Request, Mem Request, Read/Write

- » 3) Use one common bus for memory and I/O with common control lines : **Memory Mapped I/O**

- **MOV** or **LD** : I/O and Memory read/write Instruction

*** Control Lines**

Read/Write

Intel, Zilog

Motorola

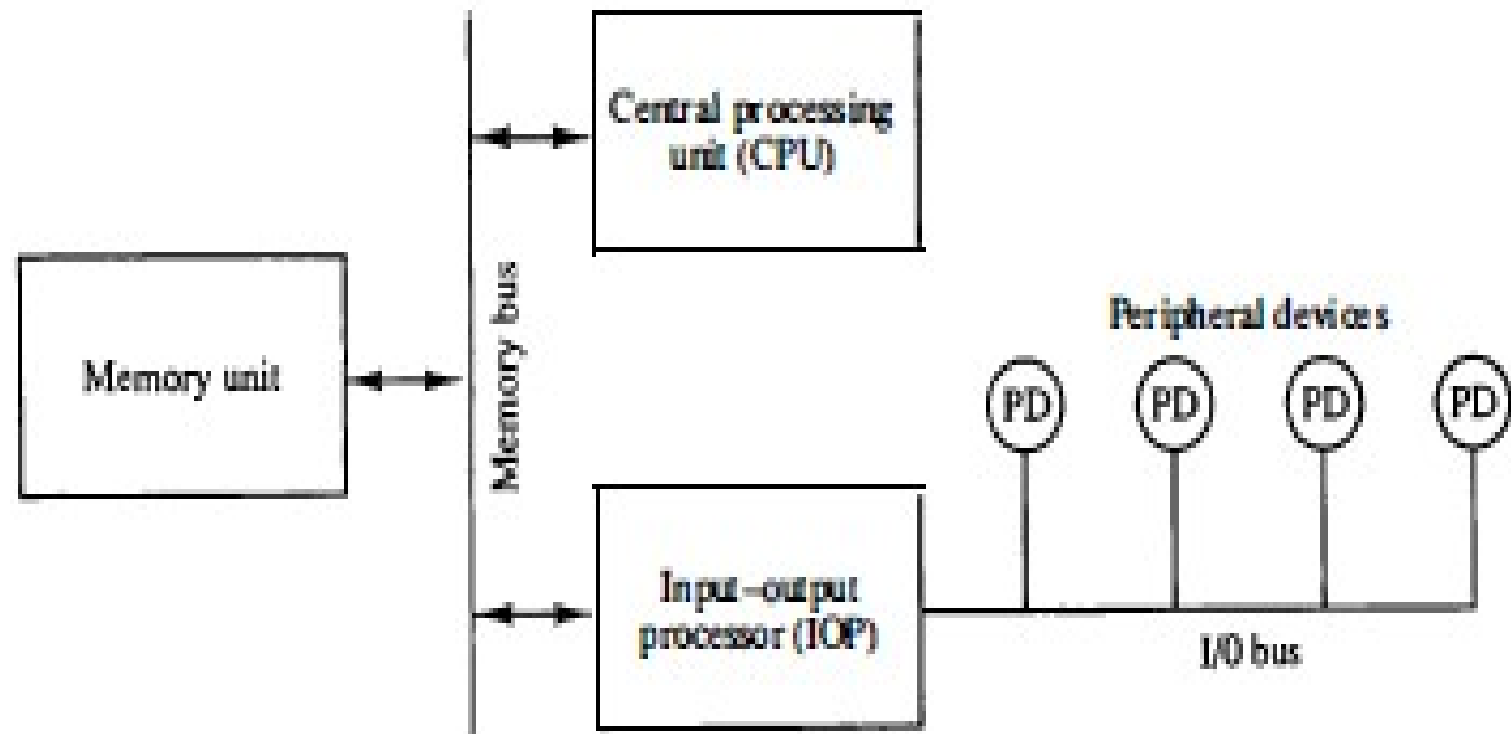
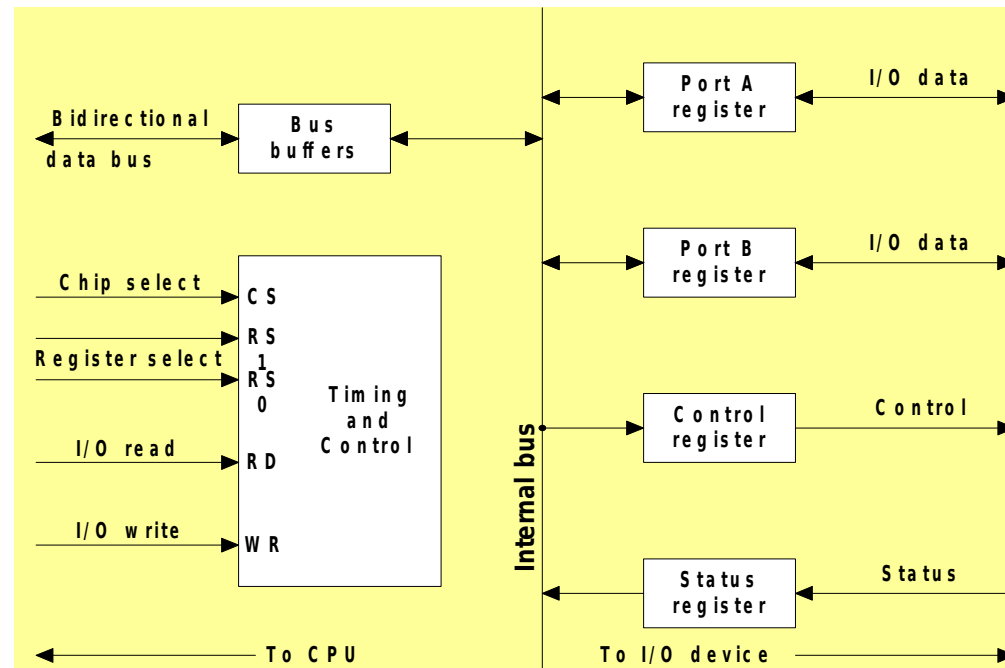


Figure 11-19 Block diagram of a computer with I/O processor.

◆ Example of I/O Interface : *Fig. 11-2*

- 4 I/O port : Data port A, Data port B, Control, Status
 - » Compare : 8255 PIO (port A, B, C, Control/Status)
- Address Decode : CS, RS1, RS0



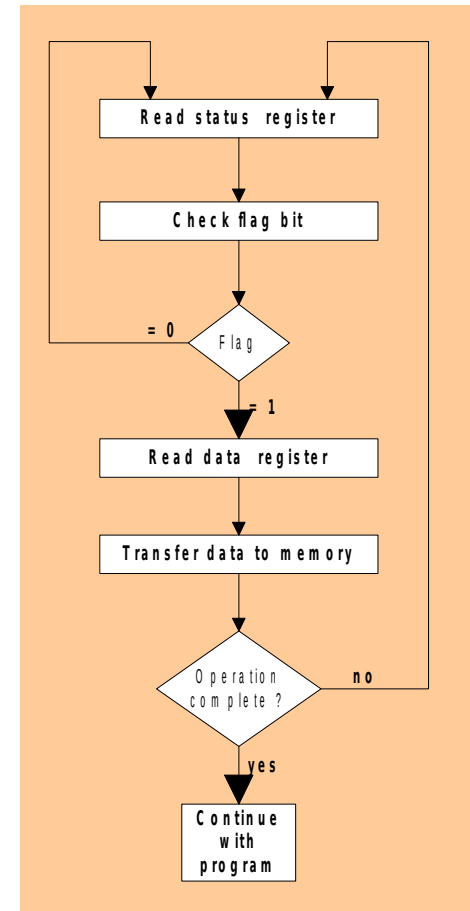
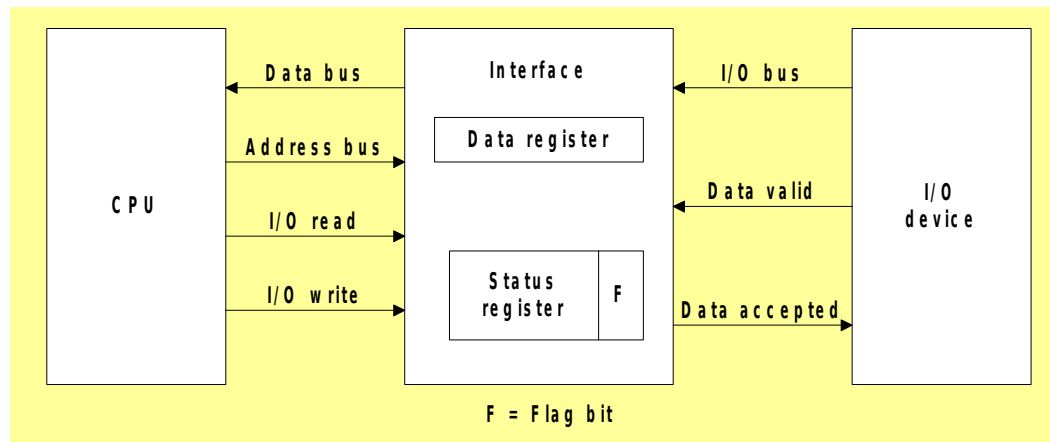
CS	RS	RS	Register selected
0	x	x	None : data bus in high- impedance
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

■ 7-3 Modes of Transfer

◆ Data transfer to and from peripherals

- 1) Programmed I/O : *in this section*
- 2) Interrupt-initiated I/O : *in this section and sec. 7-5*
- 3) Direct Memory Access (**DMA**) : *sec. 7-6*
- 4) I/O Processor (**IOP**)

◆ Example of Programmed I/O : *Fig. 11-10, 11-11*



◆ Interrupt-initiated I/O

- 1) Non-vectored : fixed branch address
- 2) Vectored : interrupt source supplies the branch address (**interrupt vector**)

◆ Software Considerations

- I/O routines
 - » software routines for controlling peripherals and for transfer of data between the processor and peripherals
- I/O routines for standard peripherals are provided by the manufacturer (**Device driver, OS or BIOS**)
- I/O routines are usually included within the operating system
- I/O routines are usually available as operating system procedures (**OS or BIOS function call**)

■ 7-4 Priority Interrupt

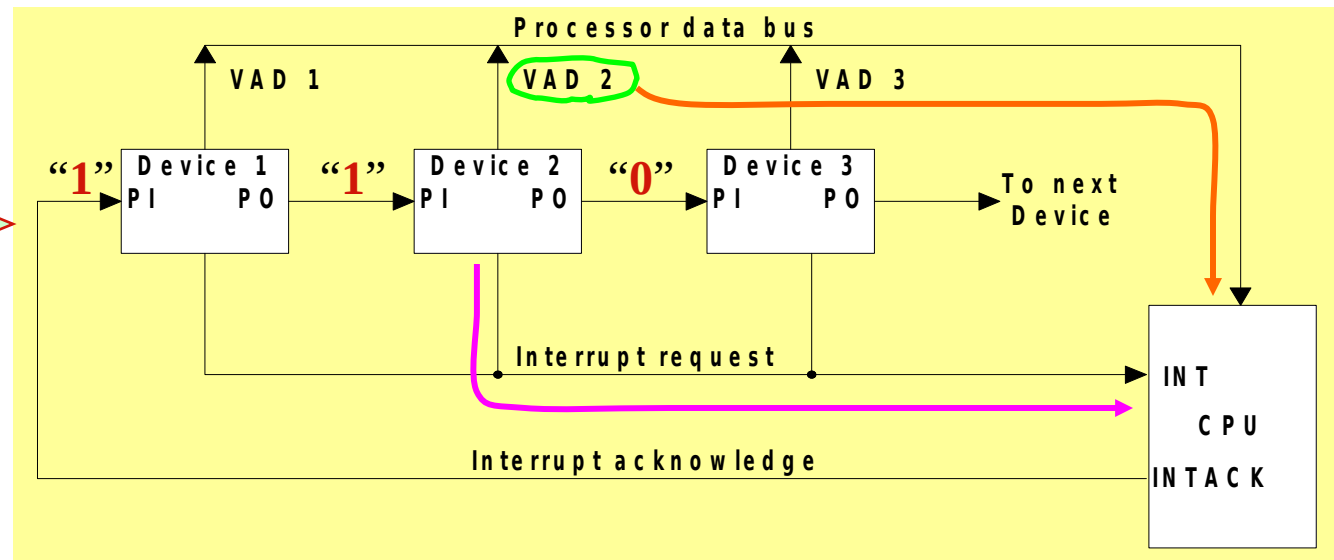
◆ Priority Interrupt

- Identify the source of the interrupt when several sources will request service simultaneously
- Determine which condition is to be serviced first when two or more requests arrive simultaneously
- Establishing the priority of simultaneous interrupts can be done by:
 - » 1) Software : **Polling**
 - » 2) Hardware : **Daisy chain, Parallel priority**

◆ Polling

- Identify the highest-priority source by software means
 - » One common branch address is used for all interrupts
 - » Program polls the interrupt sources in sequence
 - » The highest-priority source is tested first
- Polling priority interrupt
 - » If there are many interrupt sources, the time required to poll them can exceed the time available to service the I/O device
 - » Hardware priority interrupt

◆ Daisy-Chaining : *Fig. 11-12*



◆ Parallel Priority

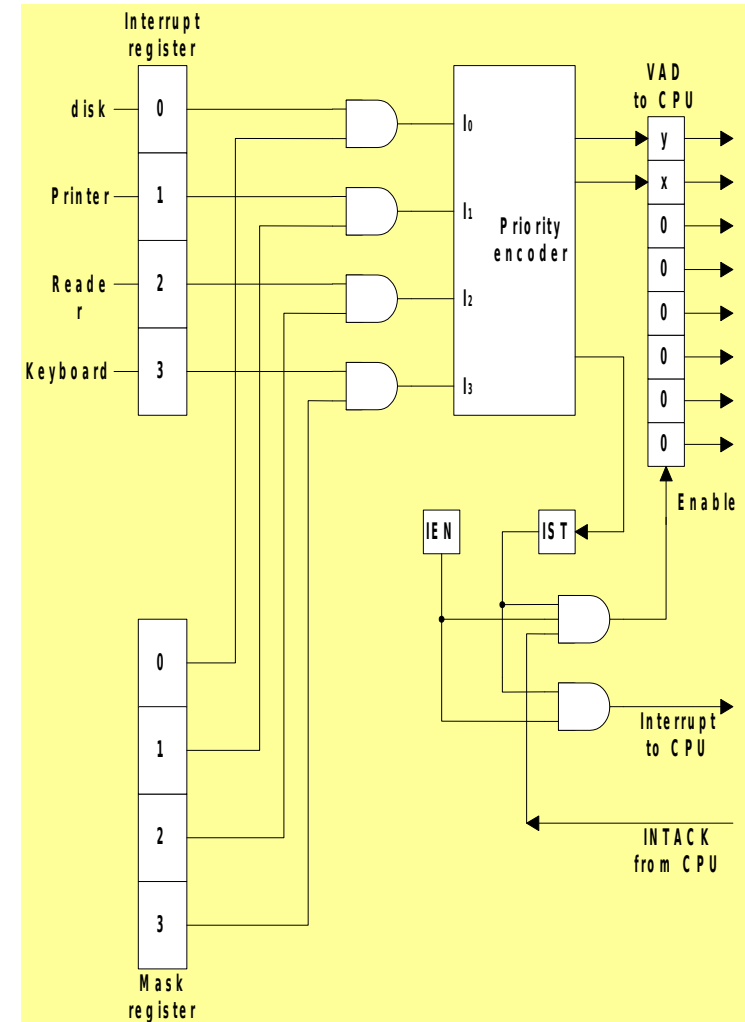
- Priority Encoder Parallel Priority : **Fig. 11-14**
 - » Interrupt Enable F/F (**IEN**) : set or cleared by the program
 - » Interrupt Status F/F (**IST**) : set or cleared by the encoder output
- Priority Encoder Truth Table : **Tab. 11-2**

◆ Interrupt Cycle

- At the end of each instruction cycle, CPU checks IEN and IST
- if both IEN and IST equal to “1”
- CPU goes to an Instruction Cycle
 - » Sequence of microoperation during Instruction Cycle

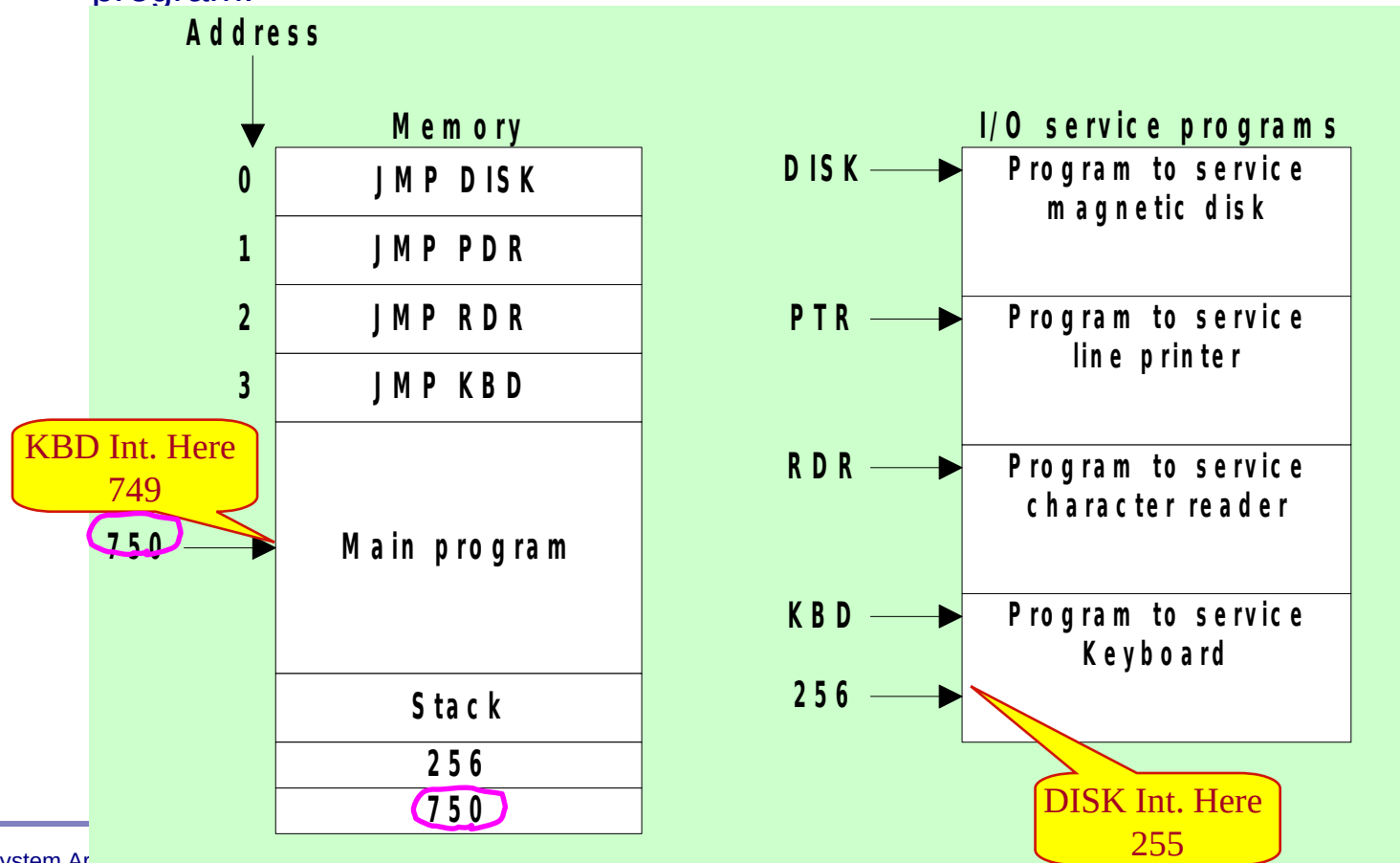
$SP \leftarrow SP - 1$	: Decrement stack point
$M[SP] \leftarrow PC$	: Push PC into stack
$INTACK \leftarrow 1$	: Enable INTACK
$PC \leftarrow VAD$	: Transfer VAD to PC
$IEN \leftarrow 0$	: Disable further interrupts
Go to Fetch next instruction	

Branch to ISR



◆ Software Routines

- for servicing the interrupt requests and for controlling the interrupt hardware registers
- Each device has its own service program that can be reached through a jump (JMP) instruction stored at the assigned vector address.
- The symbolic name of each routine represents the starting address of the service program.

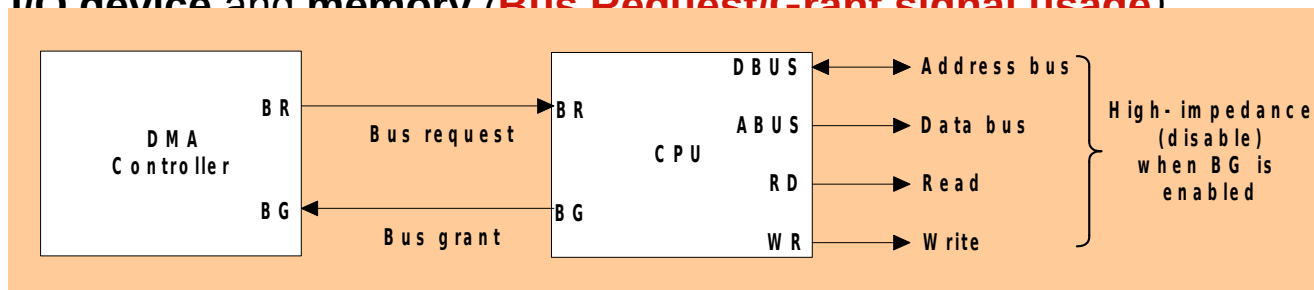


- Initial Operation of ISR
 - » 1) Clear lower-level mask register bit
 - » 2) Clear interrupt status bit IST
 - » 3) Save contents of processor registers
 - » 4) Set interrupt enable bit IEN
 - » 5) Proceed with service routine
- Final Operation of ISR
 - » 1) Clear interrupt enable bit IEN
 - » 2) Restore contents of processor registers Fig. 11-14
 - » 3) Clear the bit in the **interrupt register** belonging to the source that has been serviced
 - » 4) Set lower-level priority bits in the mask register
 - » 5) Restore return address into PC and set IEN

■ 7-5 Direct Memory Access (DMA)

◆ DMA

- DMA controller takes over the buses to manage the transfer **directly** between the I/O device and memory (**Bus Request/Grant signal usage**)

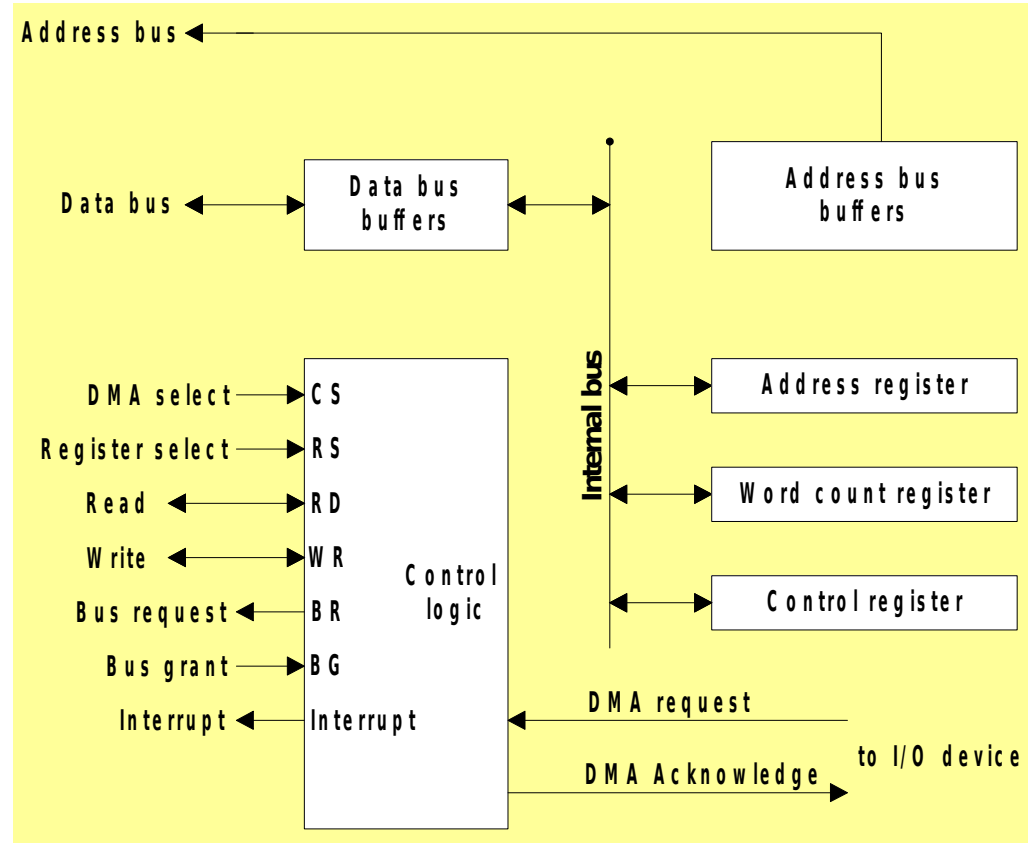


◆ Transfer Modes

- 1) Burst transfer : Block
- 2) Cycle stealing transfer : Byte

◆ DMA Controller (Intel 8237 DMAC) : Fig. 11-17

- DMA Initialization Process
 - » 1) Set Address register :
 - memory address for read/write
 - » 2) Set Word count register :
 - the number of words to transfer
 - » 3) Set transfer mode :
 - read/write,
 - burst/cycle stealing,
 - I/O to I/O,
 - I/O to Memory,
 - Memory to Memory
 - Memory search
 - I/O search
 - » 4) DMA transfer start : *next section*
 - » 5) EOT (End of Transfer) :
 - Interrupt



◆ DMA Transfer (I/O to Memory)

- 1) I/O Device sends a DMA request
- 2) DMAC activates the **BR** line
- 3) CPU responds with **BG** line
- 4) DMAC sends a DMA acknowledge to the I/O device
- 5) I/O device puts a word in the data bus (*for memory write*)
- 6) DMAC write a data to the address specified by **Address register**
- 7) Decrement **Word count register**
- 8) **Word count register** = 0
EOT interrupt CPU
- 9) **Word count register** $\neq$ 0
DMAC checks the DMA request from I/O device

