UNIT 3

Peripheral Devices and their characteristics: Introduction, Input-Output Interface, Modes of Transfer-Programmed I/O, Priority Interrupt, Direct memory Access, Input – Output Processor (IOP), Intel 8089 IOP, Standard I/O interfaces – PCI, USB, SCSI.

Introduction

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

Input-Output Interface

Input-Output 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 peripheral 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 :

I/O Bus

Data lines

Address lines

Control lines

Interface Modules : VLSI Chip

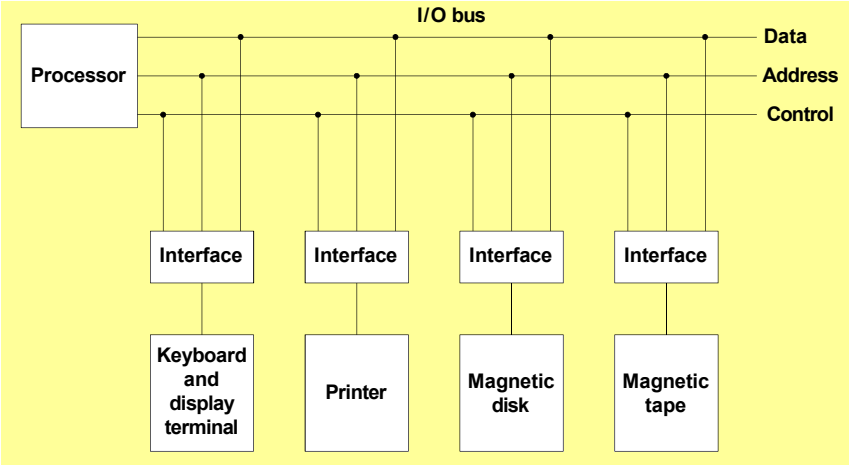
SCSI(Small Computer System Interface)

IDE (Integrated Device Electronics)

Centronics

RS-232

IEEE-488 (GPIB)



Connection of I/O bus to input-output devices

I/O command :

1. Control Command
2. Status Command
3. Input Command
4. Output Command

I/O Bus versus Memory Bus

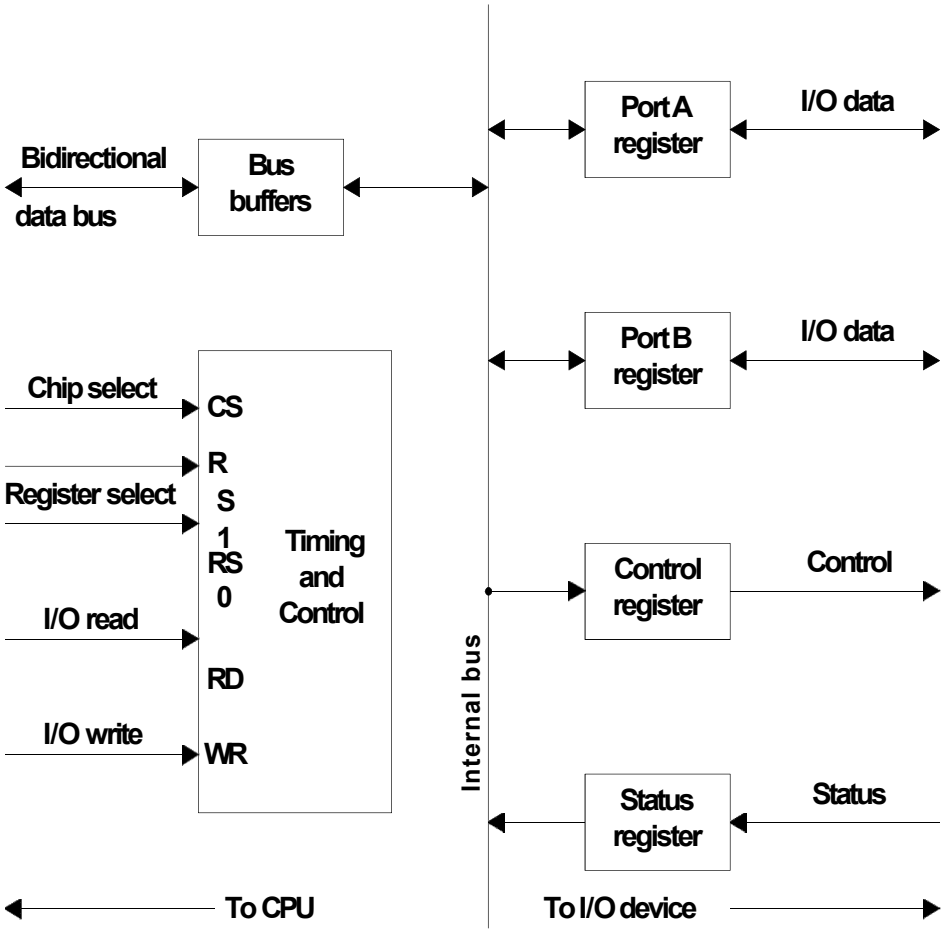
Computer buses can be used to communicate with memory and I/O

1) Use two separate buses, one for memory and the other for I/O (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*

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

Example of I/O Interface:



CS	RS1	RS0	Register selected
0	×	×	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

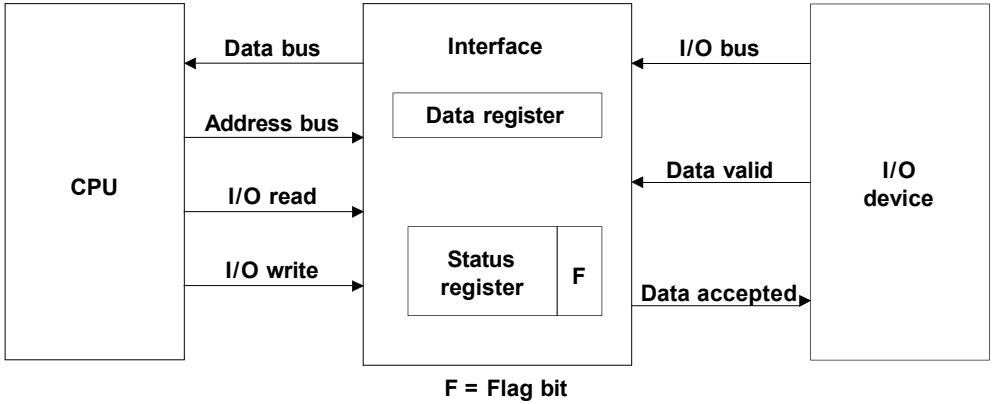
Modes of Transfer

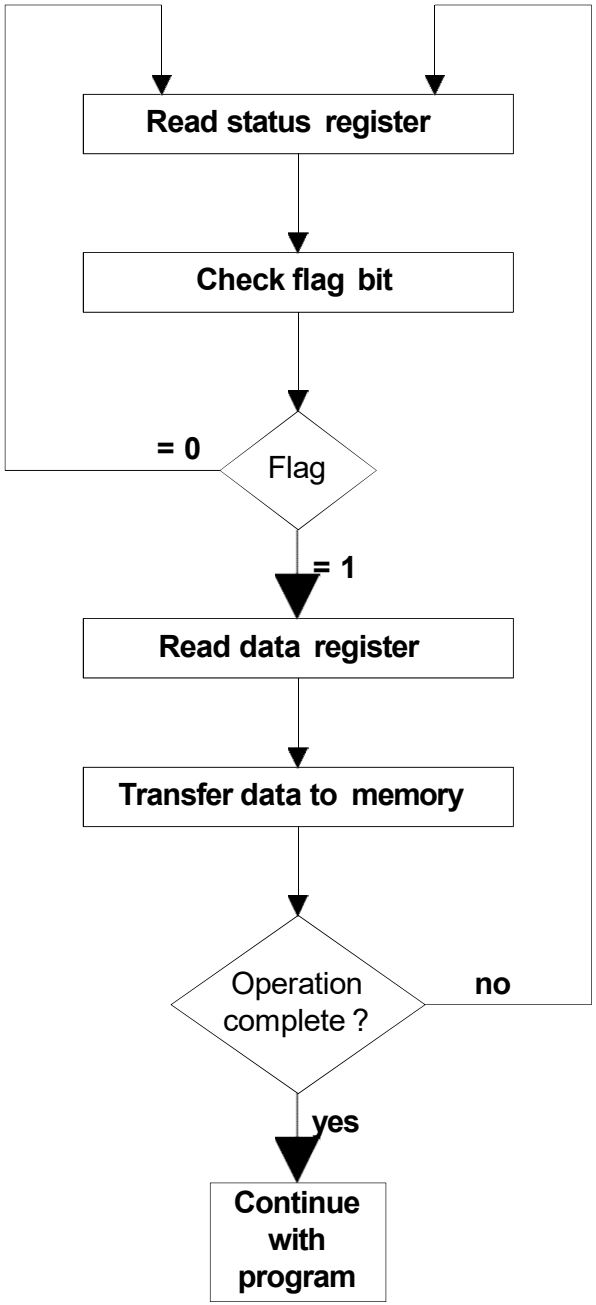
Data transfer to and from peripherals

1) *Programmed I/O*
2) *Interrupt-initiated I/O*
3) *Direct Memory Access (DMA)*

Example of Programmed I/O

Data transfer from I/O device to CPU.





Flowchart for CPU program to input data.

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)

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*

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

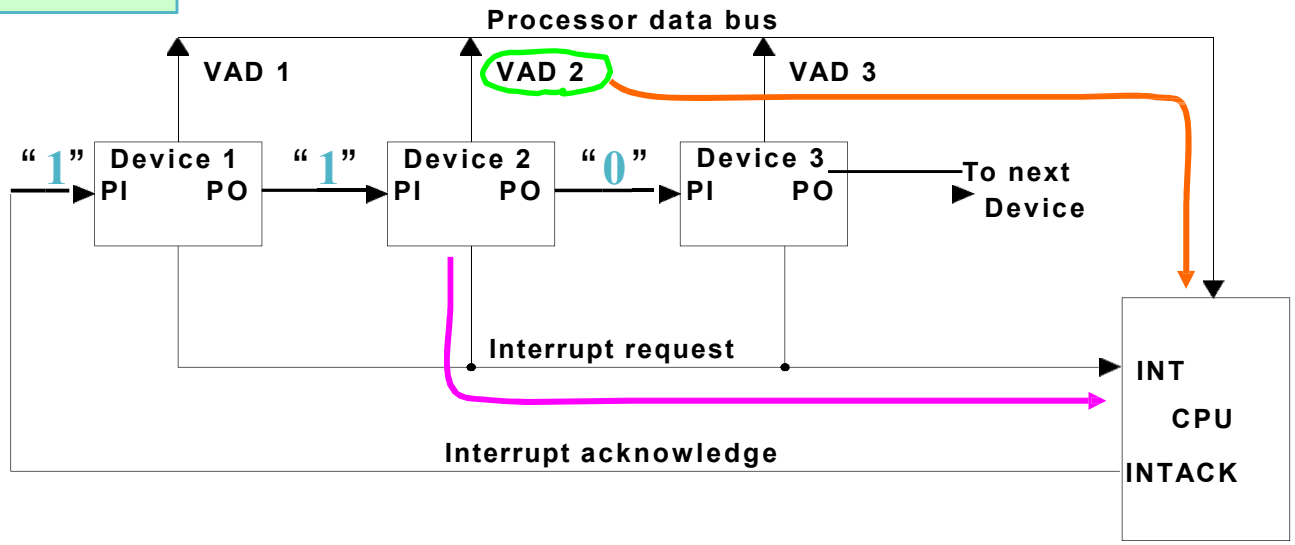
Polling priority interrupt

If there are many interrupt sources, the time required to poll them can exceed the time available to service the I/O device

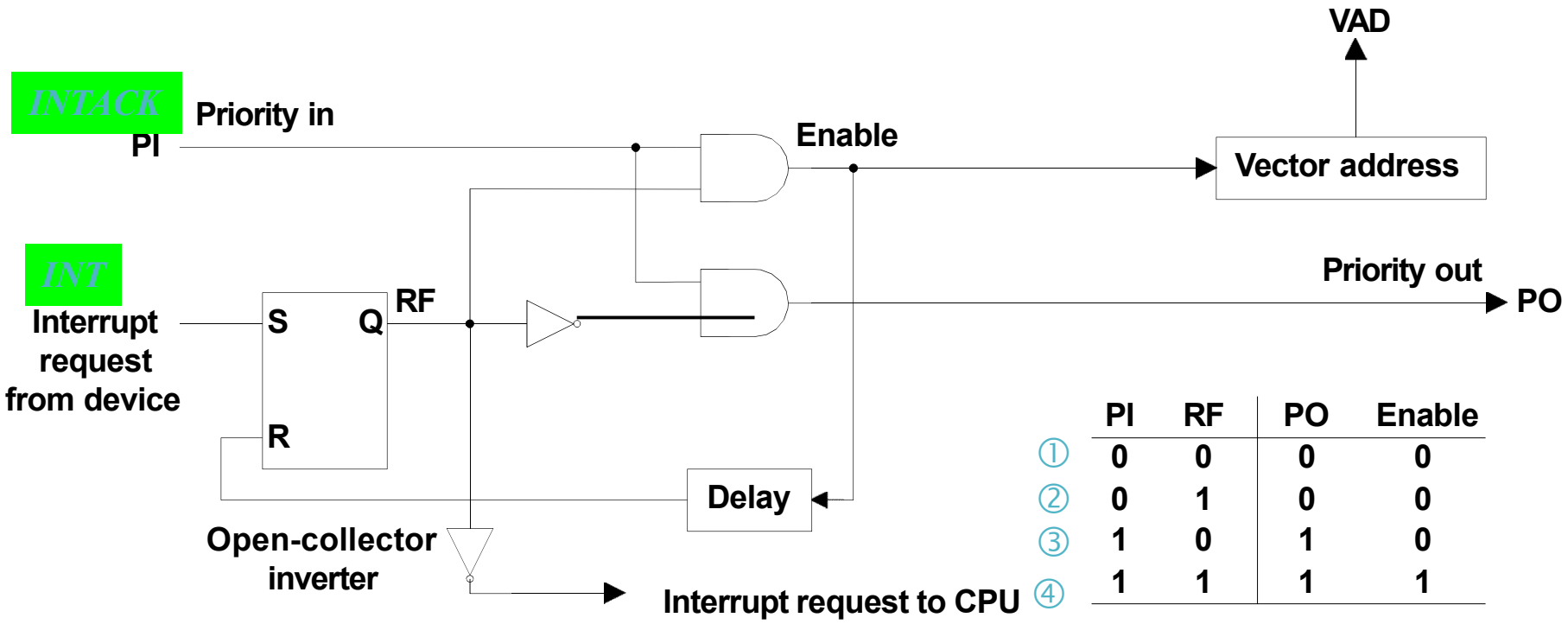
Hardware priority interrupt

Daisy Chaining

Device 2
Interrupt Request



One stage of the daisy-chain priority arrangement.



	PI	RF	PO	Enable
①	0	0	0	0
②	0	1	0	0
③	1	0	1	0
④	1	1	1	1

One stage of the daisy-chain priority arrangement :

① *No interrupt request*
② *Invalid : interrupt request, but no acknowledge*
③ *No interrupt request : Pass to other device (other device requested interrupt)*
④ *Interrupt request*

Parallel Priority

Priority Encoder Parallel Priority :

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 :

Inputs				Outputs			Boolean functions
I_0	I_1	I_2	I_3	x	y	IST	
1	×	×	×	0	0	1	
0	1	×	×	0	1	1	$x = I_0' I_1'$
0	0	1	×	1	0	1	$y = I_0' I_1 + I_0' I_2'$
0	0	0	1	1	1	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	0	×	×	0	

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 Interrupt Cycle

Sequence of microoperations during Interrupt Cycle

$SP \leftarrow SP - 1$: Decrement stack pointer

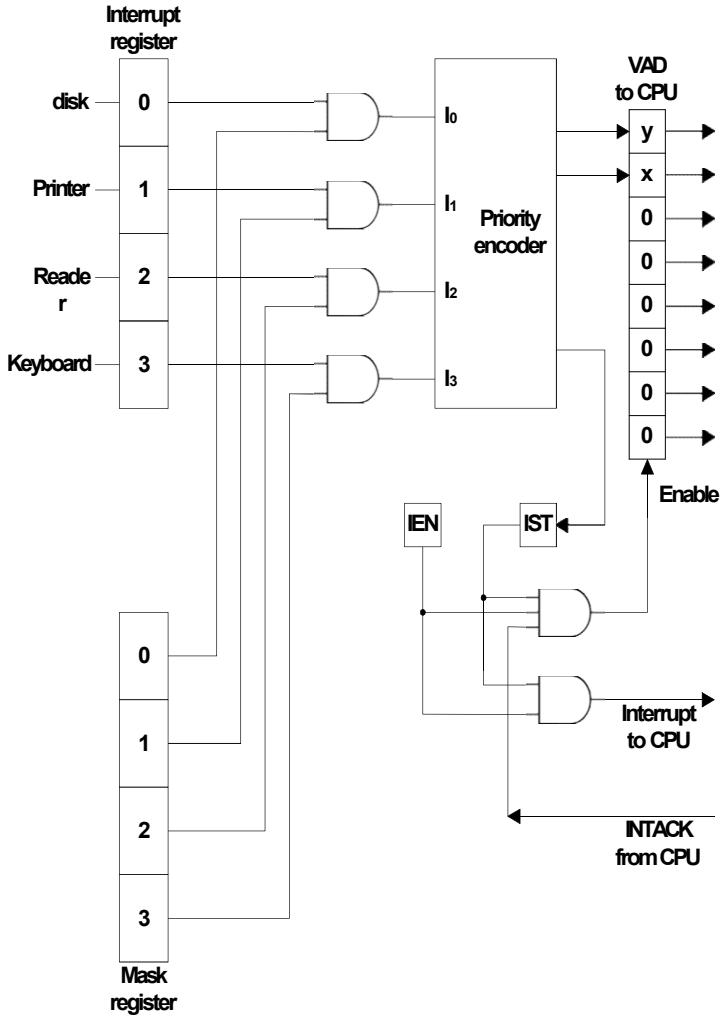
$M[SP] \leftarrow PC$: Push PC into stack

$INTACK \leftarrow 1$: Enable INTACK

$PC \leftarrow VAD$: Transfer VAD to PC

$IEN \leftarrow 0$: Disable further interrupts

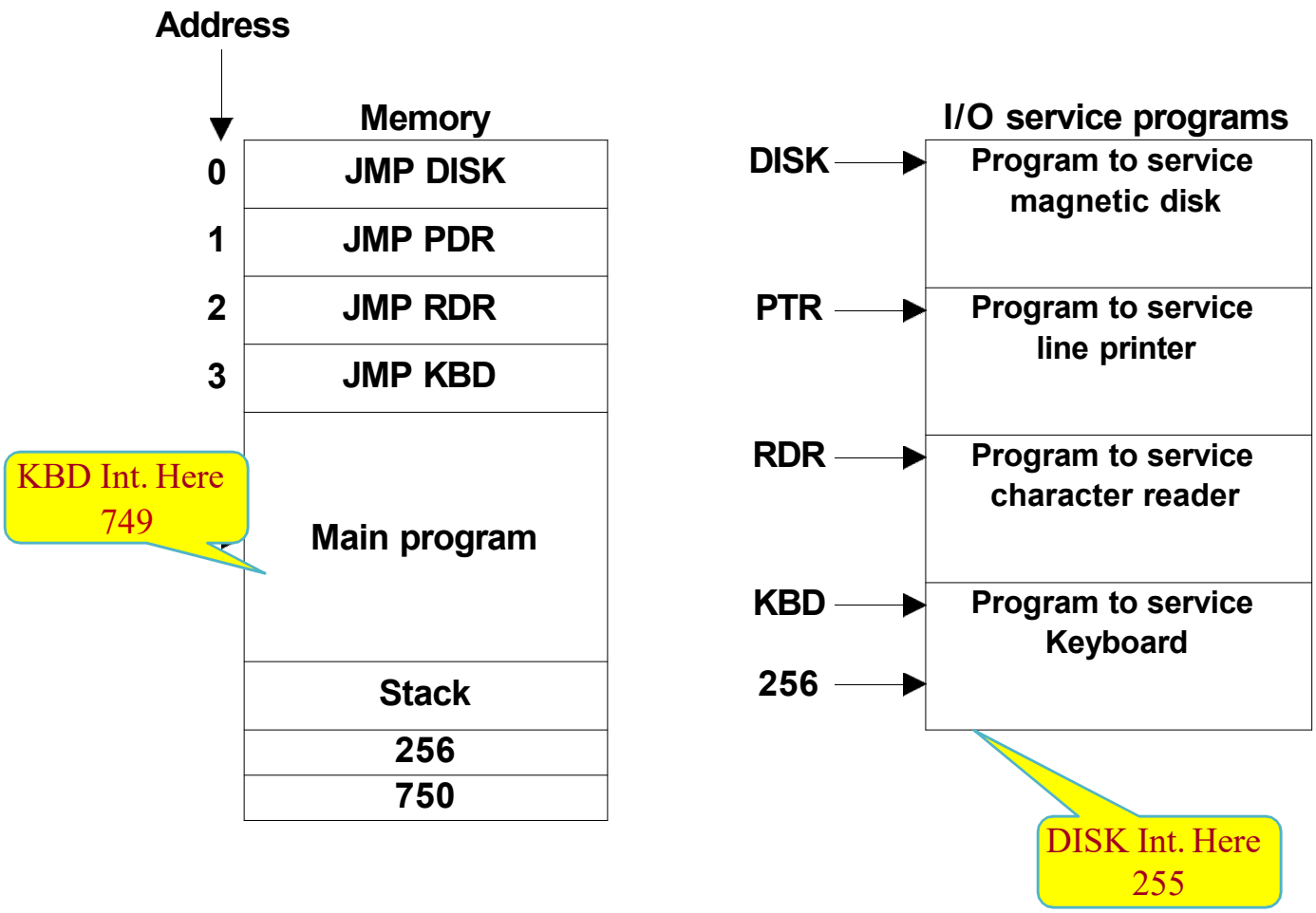
Go to Fetch next instruction



Software Routines

CPU main program 749 KBD interrupt

KBD service program 255 DISK interrupt



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

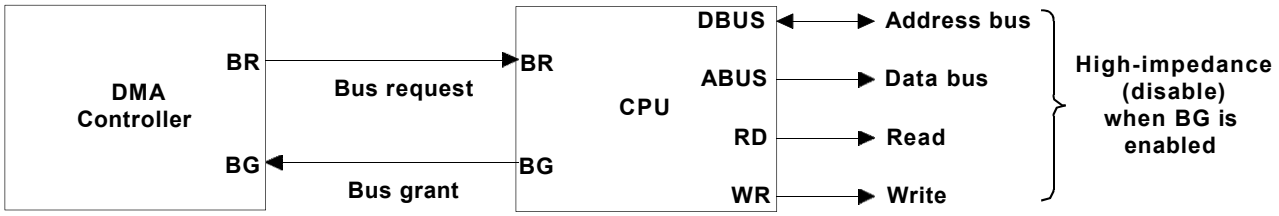
3) Clear the bit in the interrupt register belonging to the source that has been serviced

4) Set lower-level priority bits in the mask register

5) Restore return address into PC and set IEN

Direct Memory Access (DMA)

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



CPU bus signals for DMA transfer.

Transfer Modes

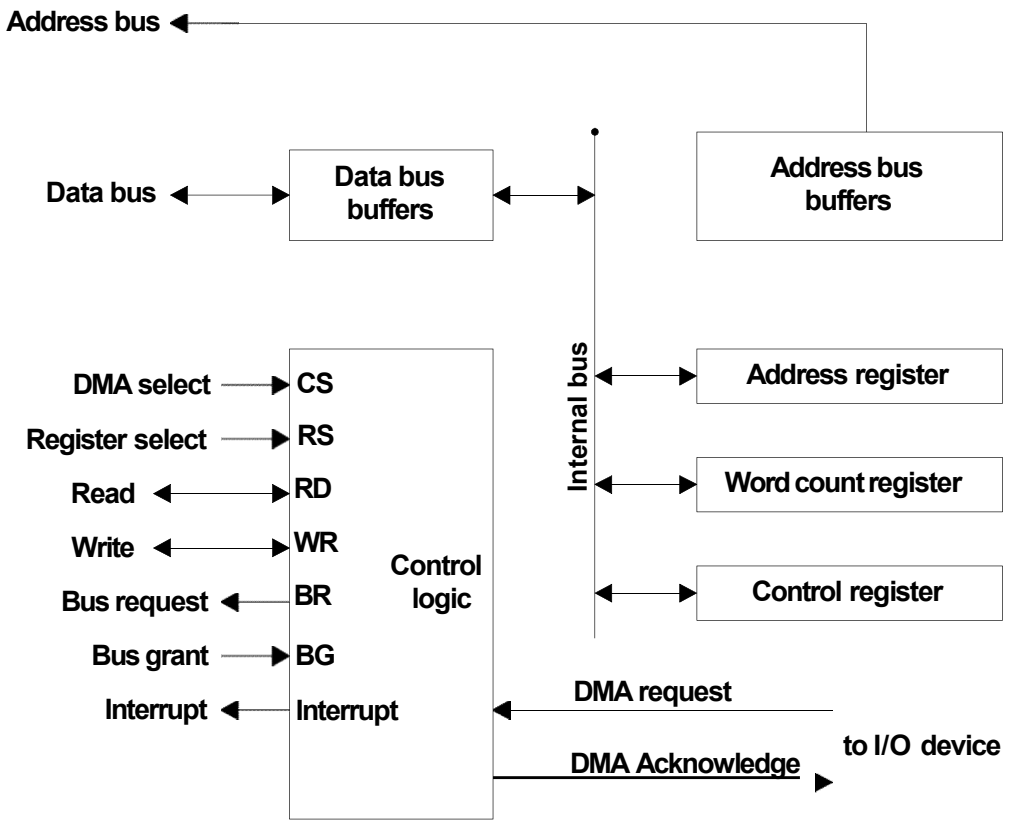
1) Burst transfer : Block

2) Cycle stealing transfer : Byte

DMA Controller (Intel 8237 DMAC) :

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:
5) EOT(End of Transfer) :
   Interrupt CPU



Block diagram of DMA controller.

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

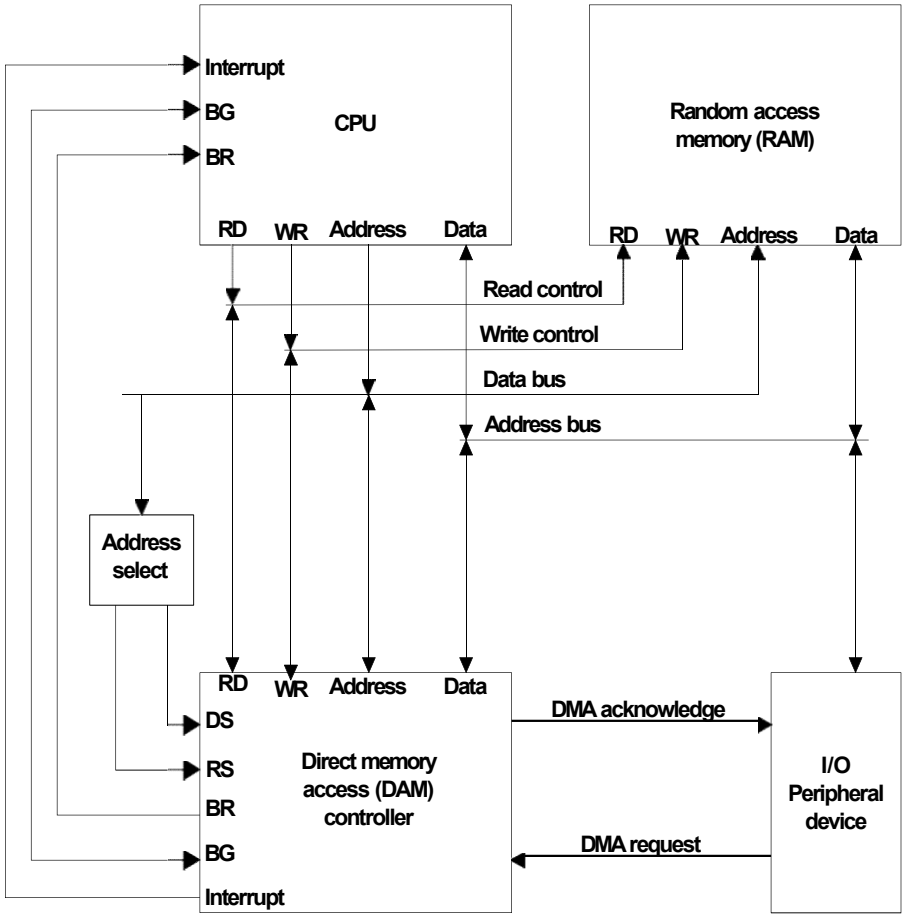
7)Decrement Word count register

8) Word count register = 0

EOT (End of Transfer), interrupt CPU

9) Word count register ≠ 0

DMAC checks the DMA request from I/O device



DMA transfer in a computer system.

- DMA transfer is very useful in many applications.

- It is used for fast transfer of information between magnetic disks and memory.

- It is also useful for updating the display in an interactive terminal. Typically, an image of the screen display of the terminal is kept in memory which can be updated under program control. The contents of the memory can be transferred to the screen periodically by means of DMA transfer.