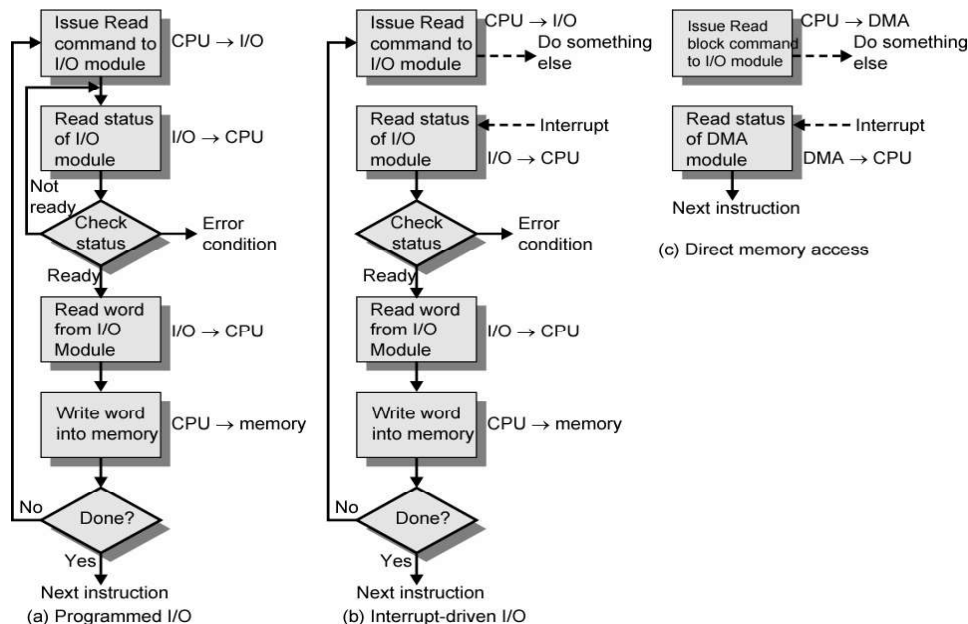


Input Output Techniques

I/O Techniques

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA)

Input Output Techniques

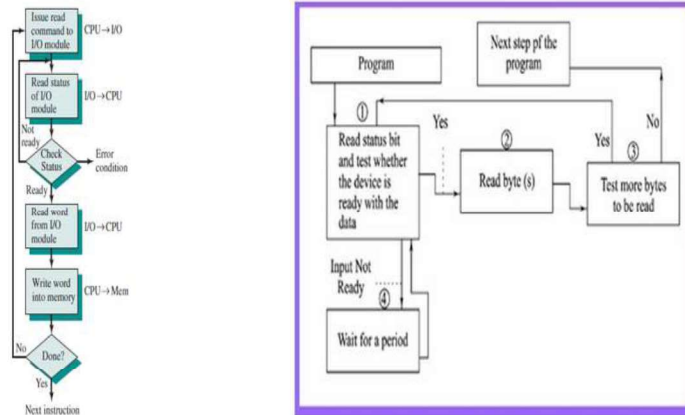


Programmed I/O

- CPU has direct control over I/O
 - Sensing status
 - Read/write commands
 - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time

Programmed I/O

Programmed I/O Mode Input Data Transfer



- CPU requests I/O operation.
- I/O module performs operation and sets status bits after completion.
- CPU checks status bits periodically.
- I/O module does not inform CPU directly that it does not interrupt CPU.
- CPU must wait.

Interrupt Driven I/O

- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready

Interrupt

- An interrupt is a signal from a device attached to a computer or from a program within the computer that causes the CPU to stop its normal program execution and perform service related to the event.
- Examples of interrupts :I/O completion, divide-by-0, etc.
- **Maskable Interrupt:** It is a hardware interrupt that may be ignored by setting a bit in an interrupt mask register's (IMR) bit-mask.
- **Non-maskable Interrupt:** is a hardware interrupt that does not have a bit-mask associated with it - meaning that it can never be ignored. NMIs are often used for timers, especially watchdog timers

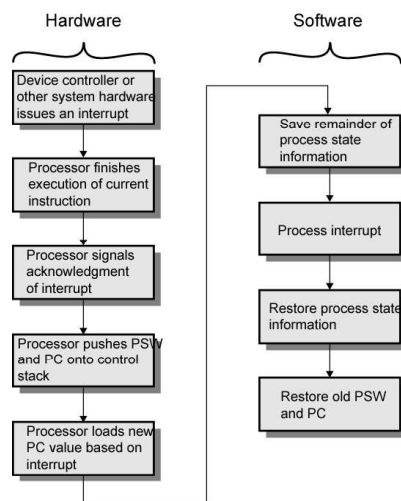
Interrupt-Initiated I/O

- When the interface determines that the I/O device is ready for data transfer, it generates an *Interrupt Request* to the CPU .
- Upon detecting an interrupt, CPU stops momentarily the task it is doing, branches to the service routine to process the data transfer, and then returns to the task it was performing

Interrupt Driven I/O Basic Operation

- CPU issues read command
- I/O module gets data from peripheral while CPU does other work
- I/O module interrupts CPU after completion
- CPU requests data
- I/O module transfers data

Simple Interrupt Processing



CPU Viewpoint

- • Issue read command
- • Do other work
- • Check for interrupt at end of each instruction cycle
- • If interrupted:-
 - – Save context (registers)
 - – Process interrupt
- • Fetch data & store

DMA

	INTERRUPT DRIVEN I/O	POLLING (PROGRAMMED I/O)
1	I/O device interrupts the processor whenever it wants to perform a data Transfer.	Processor periodically checks (polls) the status of every I/O device to know if it wants to perform a data Transfer.
2	Processor is free to carry on its own operations, hence saves system time .	Processor is busy in constantly checking all I/O devices, hence system time wasted .
3	Additional hardware required to handle interrupts. E.g.: 8259 Programmable Interrupt controller.	Additional hardware not required .
4	Increases cost and complexity of the system.	System is cheaper and less complex .
5	Interrupt priority has to be managed through software or through hardware.	No such issue.
6	Interrupt vector addresses (ISR Addresses) need to be stored in an Interrupt Vector table - IVT .	No such issue.

For **input**, the device interrupts the CPU when new data has arrived and is ready to be retrieved by the system processor. The actual actions to perform depend on whether the device uses I/O ports or memory mapping.

For **output**, the device delivers an interrupt either when it is ready to accept new data or to acknowledge a successful data transfer. Memory-mapped and DMA-capable devices usually generate interrupts to tell the system they are done with the buffer.

- 2 methods in which the CPU chooses the branch address of the service routine
 - Non-vectored interrupt
 - Branch address is assigned to a fixed location
 - User has to provide address of subroutine using a CALL instruction
 - Vectored interrupt
 - Interrupt supplies the branch info (direct or indirect address of ISR) to the CPU
 - Processor automatically generates the subroutine address

Priority Interrupt

- Establishes **priority** over the various sources to determine **which condition is to be serviced first** when two or more requests arrive simultaneously.
- The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced.
- Priority can be established by **software or hardware**

Polling-S/W

- **Software Priority – polling** – common branch address for all interrupts. Program to perform the test will be stored in this **common branch address**.
- Highest priority source is tested first and if its interrupt signal is on, control branches to a service routine for this source.
- Otherwise, the next-lower priority source is tested and so on.
- **Disadvantage – if many interrupts**, time required to poll them may exceed the time available to service the I/O device.

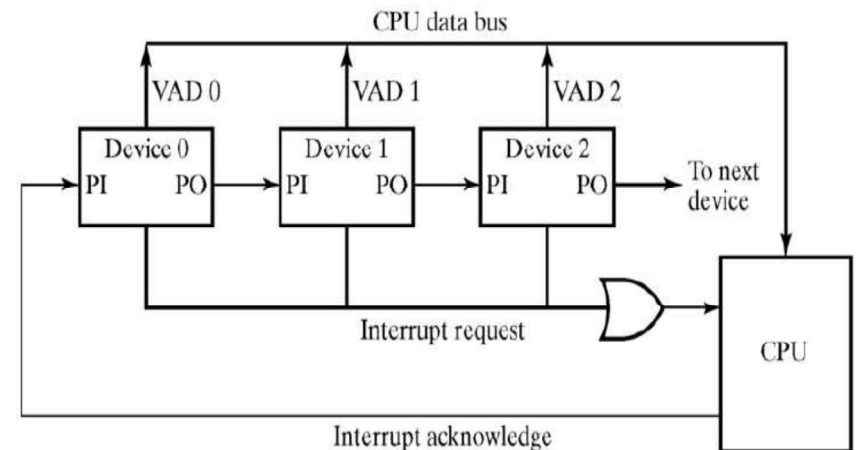
Vectored Interrupts-H/W

- Then, hardware priority interrupt unit can be used **to speed up the operation**.
- Hardware priority interrupts – accepts interrupt request from many sources, determines which of the incoming requests has the highest priority and issues an interrupt request to the computer.
- Here, each interrupt source has its own **interrupt vector** to access its own service routine directly.
- Hardware priority interrupts – **Serial or Parallel** connection of interrupt lines.

Serial - Daisy chaining

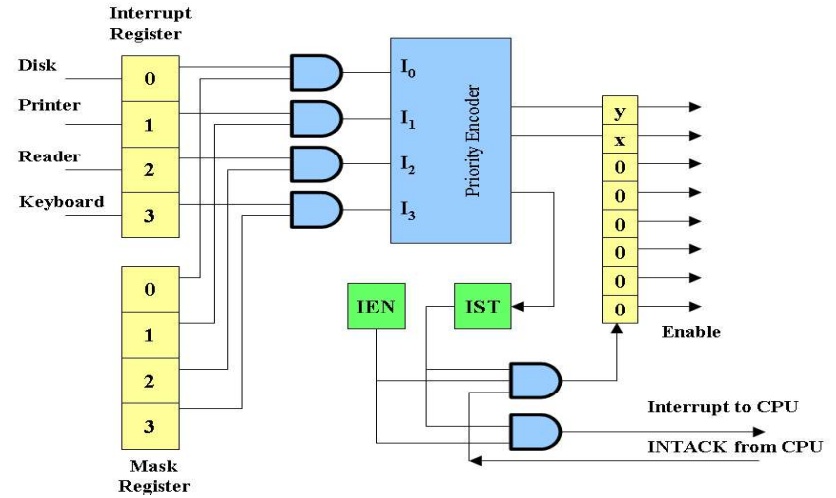
Serial hardware priority function

- * Interrupt Request Line
 - Single common line
- * Interrupt Acknowledge Line
 - Daisy-Chain



Parallel Priority Interrupt

- Uses a **register** whose bits are set separately by the interrupt signal from each device.
- Priority is established according to the **position of the bits in the register**.
- Mask register is used to disable lower priority interrupts while a higher priority device is being serviced.
- It can also provide a facility that allows a high-priority device to interrupt the CPU while a lower priority device is being serviced



- **Mask register** has a same number of bits as the interrupt register.
- By means of program instructions, it is possible to set or reset any bit in the mask register.

- The priority encoder generates two bits of the vector address, which is transferred to the CPU.
- Another output from the encoder sets an interrupt status flip-flop IST when an interrupt that is not masked occurs.
- The interrupt enable flip-flop IEN can be set or cleared by the program to provide an overall control over the interrupt system.
- IST ANDed with IEN provide a common interrupt signal for the CPU.
- The INTACK signal from the CPU enables the bus buffers in the output register and a vector address VAD is placed into the data bus.

Priority encoder

Circuit that implements the priority function.

Logic – if two or more inputs arrive at the same time, the input having the highest priority will take precedence.

Boolean functions

$$X = I_0 I_1 \quad Y = I_0 I_1 + I_0 I_2 \quad IST = I_0 + I_1 + I_2 + I_3$$

Inputs			
I ₀	I ₁	I ₂	I ₃
1	d	d	d
0	1	d	d
0	0	1	d
0	0	0	1
0	0	0	0

Outputs		
d	Y	IST
0	0	1
0	1	1
1	0	1
1	1	1
d	d	0

Interrupt

- An interrupt is a signal from a device attached to a computer or from a program within the computer that causes the CPU to stop its normal program execution and perform service related to the event.
- **Maskable Interrupt:** It is a hardware interrupt that may be ignored by set/reset a bit in an interrupt mask register's (IMR) bit-mask.
- **Non-maskable Interrupt:** is a hardware interrupt that does not have a bit-mask associated with it - meaning that it can never be ignored. NMIs are often used for timers, especially watchdog timers

Interrupt Service Routine

An interrupt handler, also known as an interrupt service routine (ISR), is a callback subroutine in an operating system whose execution is triggered by the reception of an interrupt. Interrupt handlers have a multitude of functions, which vary based on the reason the interrupt was generated.

Interrupt Overhead

The interrupt overhead is caused by **context switching** (storing and restoring the state of CPU)

On interrupt handler entry, the context of the **current process and its thread must be saved**. On exit, it must be restored.

On handler entry, memory locations different from the memory locations in the cache are used, and therefore **cache updates are required**.

Interrupt Cycle

The Interrupt enable flip-flop (IEN) can be set or cleared by program instructions.

A programmer can therefore allow interrupts (clear IEN) or disallow interrupts (set IEN)

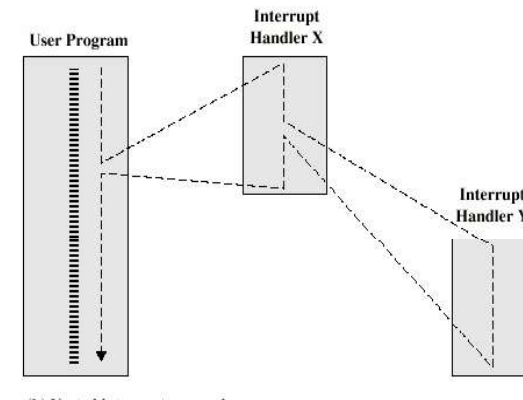
At the end of each instruction cycle the CPU checks IEN and IST. If either is equal to zero, control continues with the next instruction. If both = 1, the interrupt is handled.

Interrupt micro-operations:

$SP \leftarrow SP - 1$ (Decrement stack pointer)
 $M[SP] \leftarrow PC$ Push PC onto stack
 $INTACK \leftarrow 1$ Enable interrupt acknowledge
 $PC \leftarrow VAD$ Transfer vector address to PC
 $IEN \leftarrow 0$ Disable further interrupts
Go to fetch next instruction

Interrupt nesting

- An interrupt can happen while executing an ISR. This is called *interrupt nesting*.



Role of CPU in transfer of information

peripheral device -> CPU -> memory

⇒ CPU limits the speed of transfer

Peripheral device -> memory

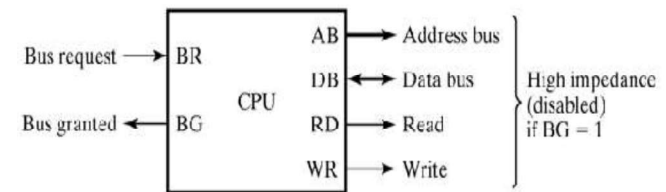
⇒ Transfer speed increases

⇒ Peripheral device manage the memory bus directly.

⇒ DMA (Direct memory Access)

⇒ CPU is idle

CPU Bus Signals for DMA Transfer

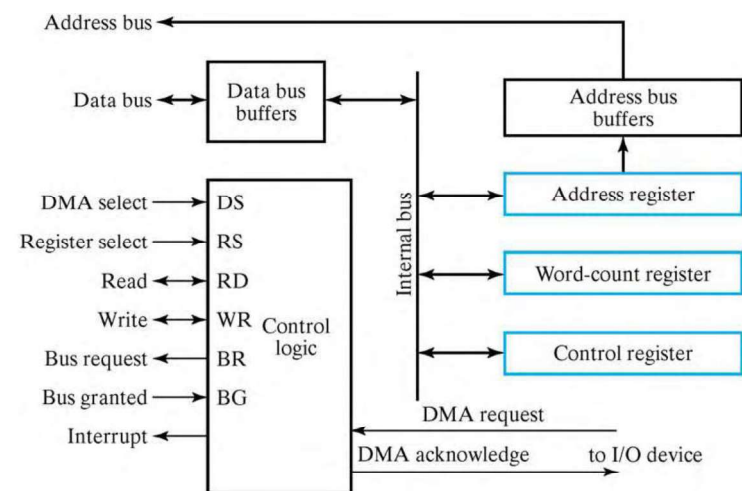


- DMA Controller sends **B**us **R**quest to CPU
- CPU stops execution of current instruction and places address bus, data bus, read and write lines into high impedance state.
- CPU activates **B**us **G**rant
- After transfer DMA disables BR
- CPU continues normal operation

Different ways of DMA transfer

- **Burst transfer** – a block sequence consisting of a number of memory words is transferred in a continuous burst => need for fast devices
- **Cycle stealing** – transfer one word at a time, after which it must return control of the buses to the CPU.
 - CPU delays 1 cycle to allow Direct Memory I/O transfer

DMA Controller



DMA Controller contd.,

- Three registers
 - **Address registers** – to specify the desired location in memory
 - Incremented after each word transfer
 - **Word count registers** – numbers of words to be transferred
 - Decmented after each word transfer and tested for 0.
 - **Control register** – specifies mode of transfer

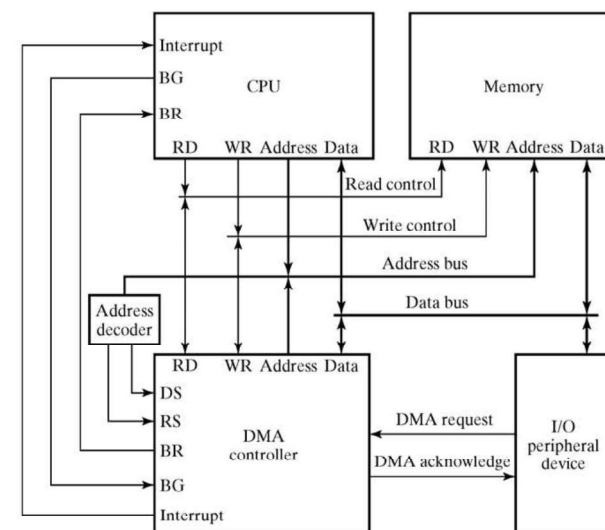
Initialization of DMA

- **CPU initializes DMA** by sending the following information through data bus
 - Starting address of the memory block where data are available (for read) or where data are to be stored (for write)
 - The word count
 - Control – to read or write
 - A control to start the DMA transfer
- After initialization, **CPU stops communicating** with DMA unless it receives **an interrupt signal** or if it wants to check how many words have been transferred.

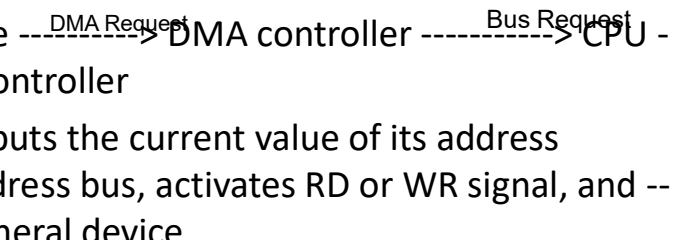
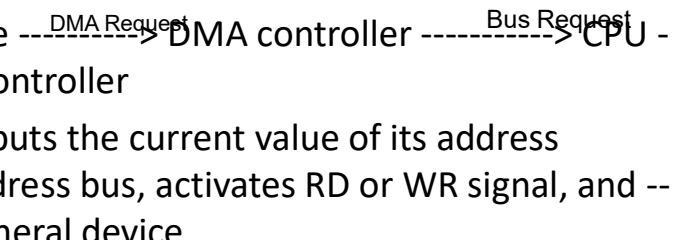
DMA Controller

- Data bus and control lines => used to communicate with CPU
- Registers in DMA are selected by CPU through address bus by enabling DS and RS inputs.
- BG = 0 => CPU reads from or writes to the DMA registers
- BG = 1 => DMA directly communicates with Memory by specifying address in address bus and activates the RD or WR control.
- Request and acknowledge signals are used to communicate with peripheral devices

DMA Transfer



DMA Transfer

- Peripheral device  DMA controller  CPU
-----> DMA controller
- DMA controller puts the current value of its address register onto address bus, activates RD or WR signal, and --
-----> peripheral device
- RD and WR are bidirectional
 - BG = 0 => CPU communicates with the internal DMA registers
 - BG = 1 => RD and WR are output lines from DMA controller to the RAM to specify read or write operation for data

DMA contd.,

- When the peripheral device receives a DMA acknowledge, it puts a word onto data bus or receives a word from data bus
- For each word transfer, DMA increments address register and decrements word count register.
- If WC = 0, DMA disables Bus request.
- Termination is informed to CPU by means of interrupt signal.
- When CPU receives interrupt signal, it checks value of word count register.

Application

- Fast transfer of information between magnetic disk and memory
- Updating the display in an interactive terminal