Microprocessor

Unit 5:
Basic I/O, Memory R/W and Interrupt Operations

Course Outline

- Memory mapped I/O, I/O Mapped I/O and Hybrid I/O
- Direct Memory Access (DMA)
  - Introduction, Advantage and Application
  - 8237 DMA Controller and Interfacing
- Interrupt
  - 8085 Interrupt Pins and Priority
  - Maskable and Non-maskable Interrupts
  - RST Instructions
  - Vector and Polled Interrupt
- 8259 Interrupt Controller
  - Block Diagram and Explanation
  - Priority Modes and Additional Features

Interrupts

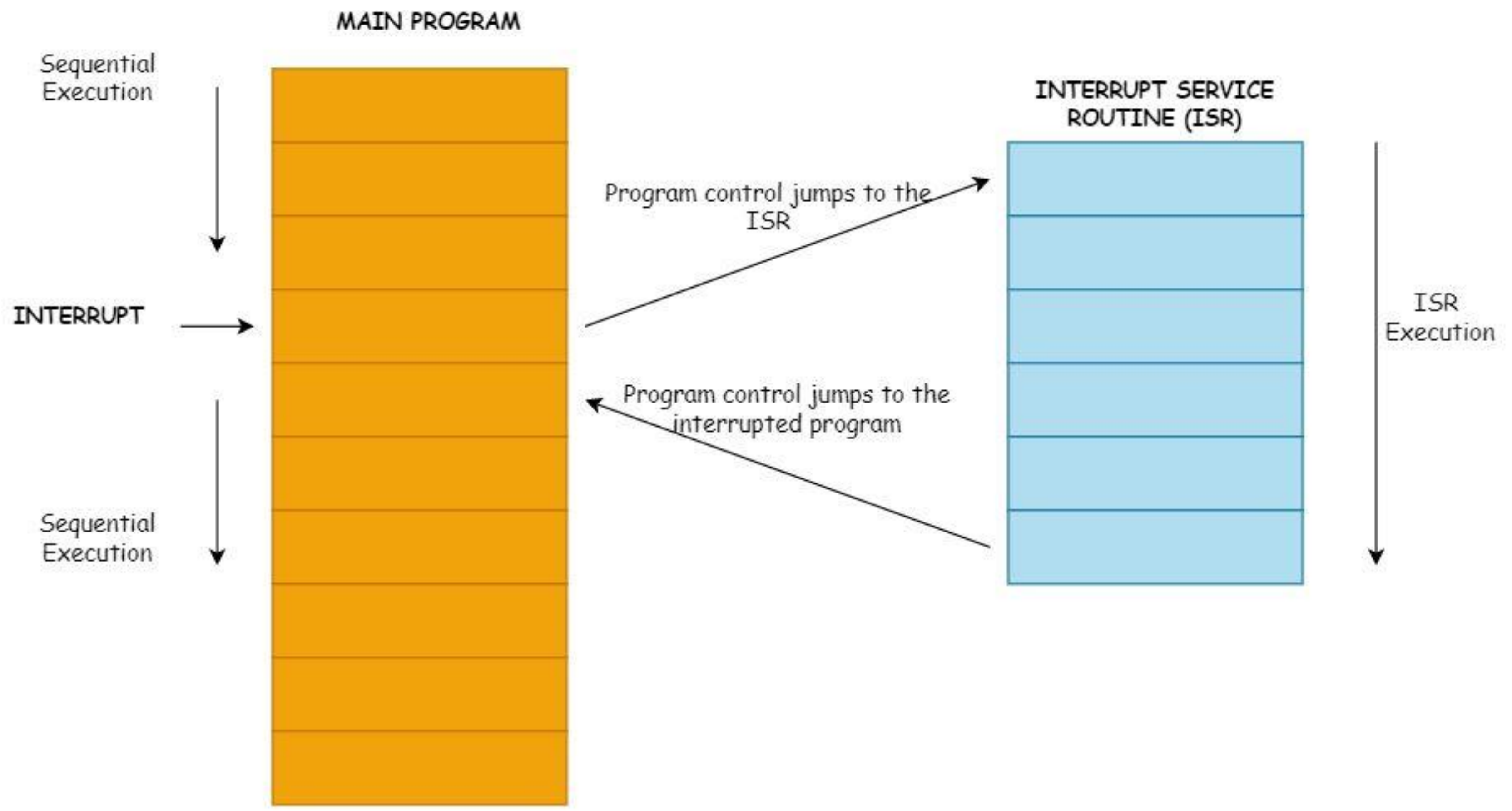
Interrupts

- Interrupts are the signals generated by the external devices to request the microprocessor to perform a task.
- Whenever an interrupt request is generated in the system then it must not be neglected and be acknowledged as soon as possible.
- When a microprocessor is interrupted, it stops executing its current program and calls a special routine which services the interrupt.
  - The event that causes the interruption is called **Interrupt**.
  - The special routine executed to service the interrupt is called **ISR – Interrupt Service Routine**.

Interrupts

- When microprocessor receives any interrupt signal from peripheral(s) which are requesting its services, it stops its current execution and program control is transferred to a sub-routine by generating **CALL** signal and after executing sub-routine by generating **RET** signal again program control is transferred to main program from where it had stopped.
- When microprocessor receives interrupt signals, it sends an acknowledgement (INTA) to the peripheral which is requesting for its service.

Interrupt


MAIN PROGRAM
Sequential Execution
INTERRUPT
Sequential Execution
Program control jumps to the ISR
Program control jumps to the interrupted program
INTERRUPT SERVICE ROUTINE (ISR)
ISR Execution

Interrupt

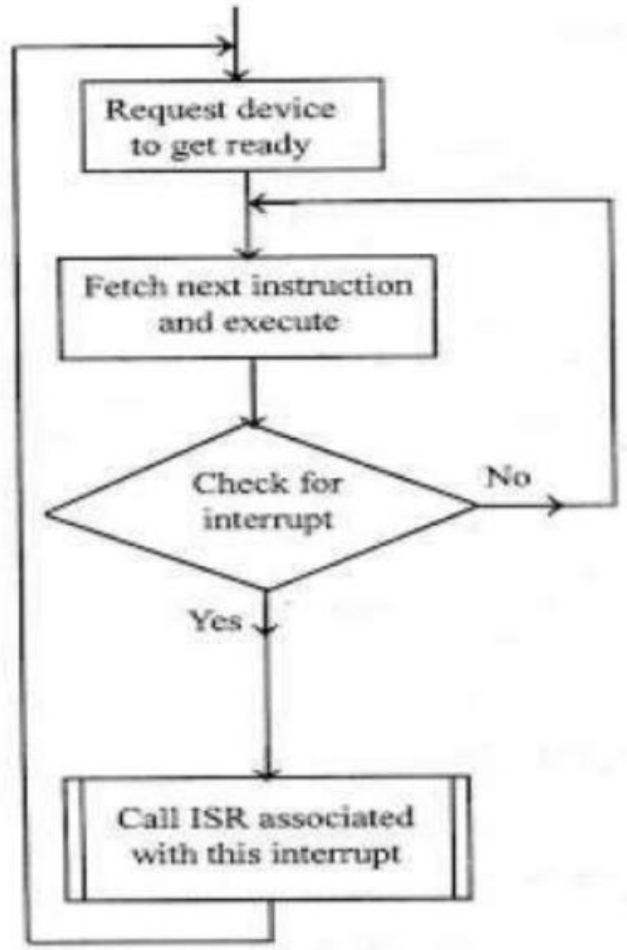

Request device to get ready
Fetch next instruction and execute
Check for interrupt
No
Yes
Call ISR associated with this interrupt

Fig (a) : Main program execution sequence

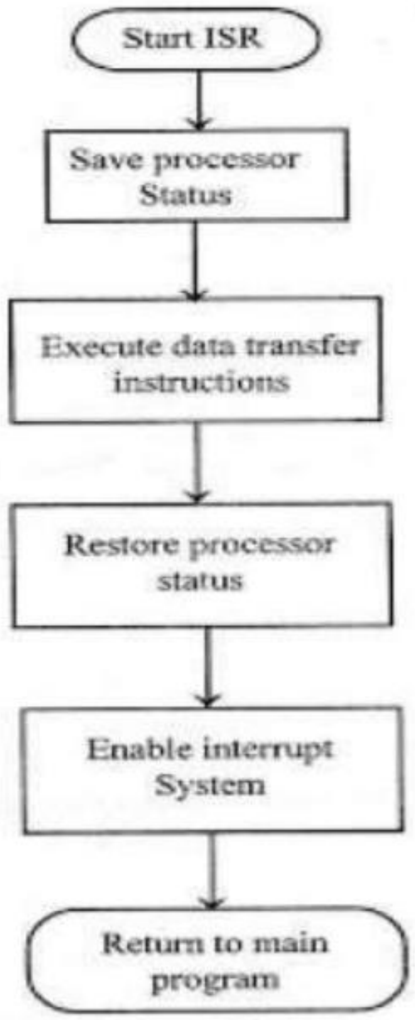


Fig (b) : ISR execution sequence

Types of Interrupts

- Hardware and Software Interrupts
  - From where it is being generated
- Vectored and Non-Vectored Interrupts
  - Based upon vector address (fixed or not)
- Maskable and Non-Maskable Interrupts
  - If it can be neglected or not

Hardware and Software Interrupts

- When microprocessors receive interrupt signals through pins (hardware) of microprocessor, they are known as Hardware Interrupts.
    - There are 5 Hardware Interrupts in 8085 microprocessor. They are – INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.
- Software Interrupts are those which are inserted in between the program which means these are mnemonics of microprocessor.
    - There are 8 software interrupts in 8085 microprocessor. They are – RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6, RST 7.

Maskable and Non-Maskable Interrupts

- **Maskable Interrupts** are those which can be disabled or ignored by the microprocessor.
  - These interrupts are either edge-triggered or level-triggered, so they can be disabled.
  - INTR, RST 7.5, RST 6.5, RST 5.5 are maskable interrupts in 8085 microprocessor.
- **Non-Maskable Interrupts** are those which cannot be disabled or ignored by microprocessor.
  - TRAP is a non-maskable interrupt.
  - It consists of both level as well as edge triggering and is used in critical power failure conditions.

Vectored and Non-Vectored Interrupts

- **Vectored Interrupts** are those which have fixed vector address (starting address of sub-routine) and after executing these, program control is transferred to that address.
- **Non-Vectored Interrupts (Scalar Interrupt)** are those in which vector address is not predefined.
  - The interrupting device gives the address of sub-routine for these interrupts. INTR is the only non-vectored interrupt in 8085 microprocessor.

Interrupt Vector

- Interrupt vectors are **addresses that inform the interrupt handler as to where to find the ISR.**
- All interrupt are mapped onto a memory area called the Interrupt Vector Table (IVT).
- The IVT is usually located in memory page 00 (0000H – 00FFH).
- Each of interrupts are associated with a specific interrupt vector.

Interrupt	**Interrupt Vector**
TRAP	0024H
RST 7.5	003CH
RST 6.5	0034H
RST 5.5	002C
INTR	Decided by hardware

Vector Interrupts

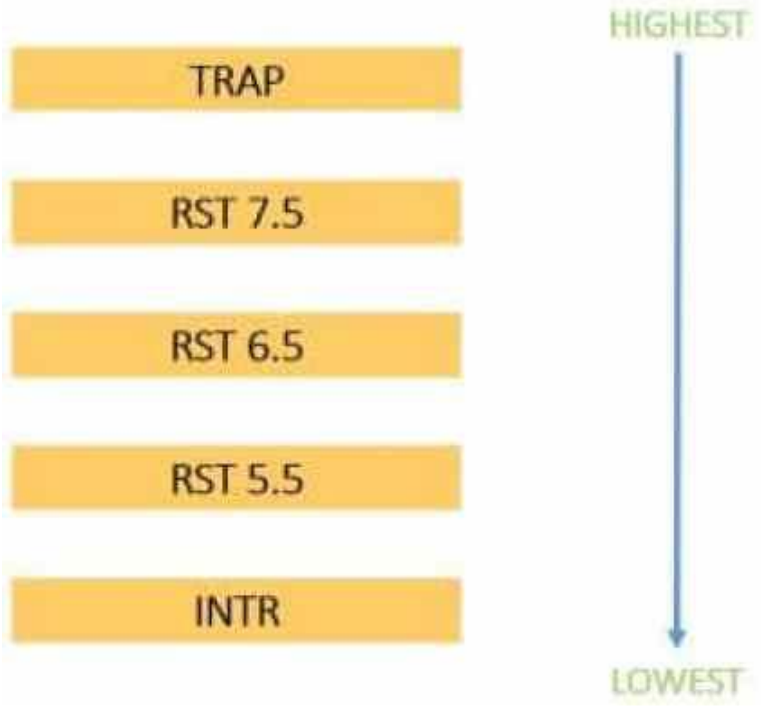
- Finding the address of these vectored interrupts are very easy. Just multiply 8 with the RST value.
- For RST 7.5,

  the subroutine (ISR) address = 8 * 7.5 = 60 = (**3C**)H
- For RST 6.5,

  the subroutine (ISR) address = 8 * 6.5 = 52 = (34)H
- Similarly for RST 5.5 and TRAP (RST 4.5)
- Memory page for all interrupts are (**00**).

Priority of Interrupts

- When microprocessor receives multiple interrupt requests simultaneously, it will execute the interrupt service request (ISR) according to the priority of the interrupts.



Interrupt: Priority, Mask, Vector

Interrupt	Priority	Trigger	Mask	Vector
TRAP (RST 4.5)	1st	Edge and Level	Non-maskable	Vectored
RST 7.5	2nd	Edge	Maskable	Vectored
RST 6.5	3rd	Level	Maskable	Vectored
RST 5.5	4th	Level	Maskable	Vectored
INTR	5th	Level	Maskable	Non-vectored

Interrupt Service Routine (ISR)

- A small program or a routine that when executed, services the corresponding interrupting source is called an ISR.

Interrupts

- **TRAP**
  - It is a non-maskable interrupt, having the highest priority among all interrupts.
  - By default, it is enabled until it gets acknowledged.
  - In case of failure, it executes as ISR and sends the data to backup memory.
  - This interrupt transfers the control to the location 0024H.
- **RST7.5**
  - It is a maskable interrupt, having the second highest priority among all interrupts.
  - When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 003CH address
- **RST 6.5**
  - It is a maskable interrupt, having the third highest priority among all interrupts.
  - When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 0034H address.

Interrupts

- **RST 5.5**
  - It is a maskable interrupt. When this interrupt is executed, the processor saves the content of the PC register into the stack and branches to 002CH address.
- **INTR**
  - It is a maskable interrupt, having the lowest priority among all interrupts. It can be disabled by resetting the microprocessor.
  - When **INTR signal goes high**, the following events can occur −
  - The microprocessor checks the status of INTR signal during the execution of each instruction.
  - When the INTR signal is high, then the microprocessor completes its current instruction and sends active low interrupt acknowledge signal.
  - When instructions are received, then the microprocessor saves the address of the next instruction on stack and executes the received instruction.

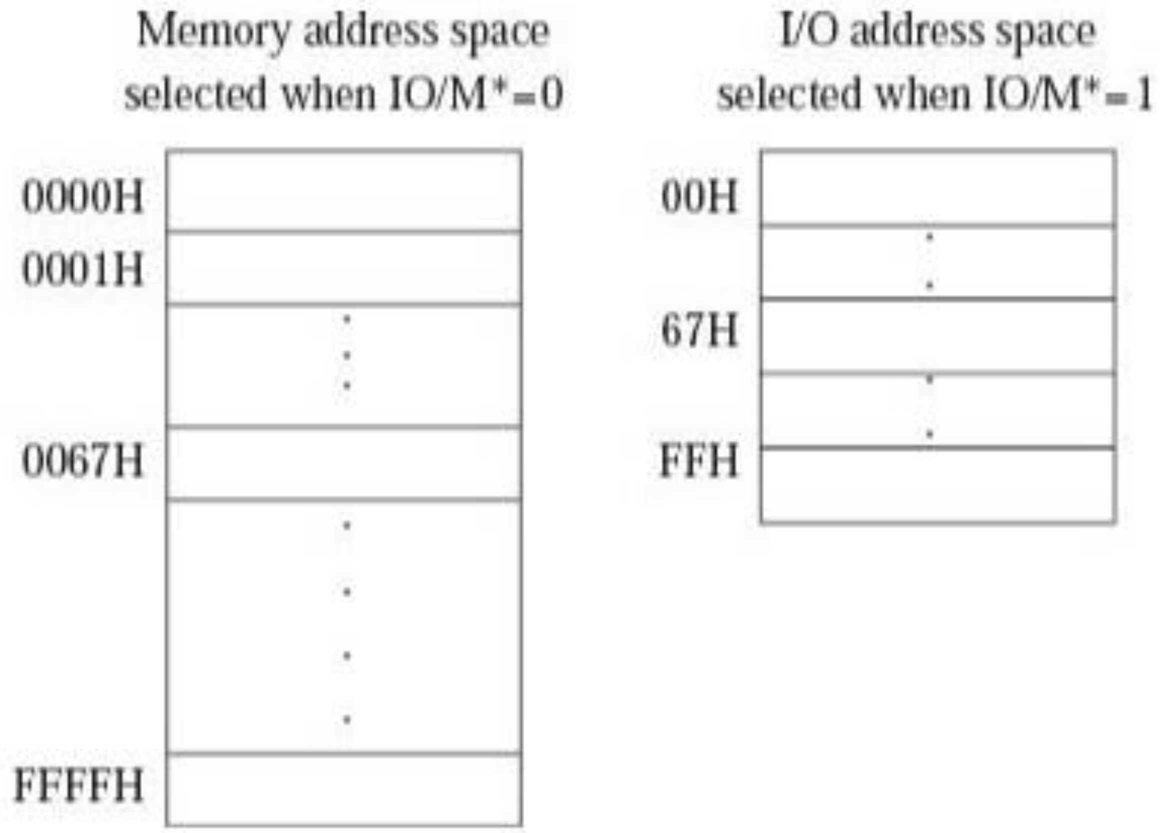
Restart instructions (RSTn) in 8085

- In 8085 Instruction set, RSTn is actually standing for “Restart n”.
- And in this case, n has a value from 0 to 7 only.
- Thus the eight possible RST instructions are there, e.g. RST 0, RST 1, …, RST 7. They are 1-Byte call instructions.
- Functionally RST n instruction is similar with:

  RST n = CALL n*8
- For example, let us consider RST 4 is functionally equivalent to CALL 4*8, i.e. CALL 32 = CALL 0020H.
- The advantage of RST 2 is that it is only 1 Byte, whereas CALL 0010H is 3-Byte long.
- Thus RST instructions are useful for branching to frequently used subroutines.

Restart instructions (RSTn) in 8085

Mnemonics, Operand	Opcode(in HEX)	In Binary	Bytes	Target Address (n*8)
RST	C7	1100 0111	1	0000H
RST 1	CF	1100 1111	1	0008H
RST 2	D7	1101 0111	1	0010H
RST 3	DF	1101 1111	1	0018H
RST 4	E7	1110 0111	1	0020H
RST 5	EF	1110 1111	1	0028H
RST 6	F7	1111 0111	1	0030H
RST 7	FF	1111 1111	1	0038H


Memory address space selected when IO/M*=0
0000H
0001H
0067H
FFFFH
I/O address space selected when IO/M*=1
00H
67H
FFH

Memory Vs IO mapped I/O

Basis for Comparison	Memory mapped I/O	I/O mapped I/O
Basic	I/O devices are treated as memory.	I/O devices are treated as I/O devices.
Allotted address size	16-bit ($A_0 – A_{15}$)	8-bit ($A_0 – A_7$)
Data transfer instructions	Same for memory and I/O devices.	Different for memory and I/O devices.
Cycles involved	Memory read and memory write	I/O read and I/O write
Interfacing of I/O ports	Large (around 64K)	Comparatively small (around 256)
Control signal	No separate control signal is needed for I/O devices.	Special control signals are used for I/O devices.
Efficiency	Less	Comparatively more

Memory Vs IO mapped I/O

Basis for Comparison	Memory mapped I/O	I/O mapped I/O
Decoder hardware	More decoder hardware required.	Less decoder hardware required.
IO/M’	During memory read or memory write operations, IO/M’ is kept low.	During I/O read and I/O write operation, IO/M’ is kept high.
Data movement	Between registers and ports.	Between accumulator and ports.
Usability	In small systems where memory requirement is less.	In systems that need large memory space.
Speed of operation	Slow	Comparatively fast
Example of instruction	LDA ****H STA ****H MOV A, M	IN **H OUT **H

I/O interface

- The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface.
- The CPU is interfaced using special communication links by the peripherals connected to any computer system.
- These communication links are used to resolve the differences between CPU and peripheral.
- There exists special hardware components between CPU and peripherals to supervise and synchronize all the input and output transfers that are called interface units.

Mode of Transfer

- The binary information that is received from an external device is usually stored in the memory unit.
- The information that is transferred from the CPU to the external device is originated from the memory unit.
- CPU merely processes the information but the source and target is always the memory unit.
- Data transfer to and from the peripherals may be done in any of the three possible ways
    - Programmed I/O.
    - Interrupt- initiated I/O.
    - Direct memory access(DMA).

Programmed I/O

- It is due to the result of the I/O instructions that are written in the computer program.
- Each data item transfer is initiated by an instruction in the program.
- Usually the transfer is from a CPU register and memory.
- In this case it requires constant monitoring by the CPU of the peripheral devices.
- **Example of Programmed I/O:** In this case, the I/O device does not have direct access to the memory unit. A transfer from I/O device to memory requires the execution of several instructions by the CPU, including an input instruction to transfer the data from device to the CPU and store instruction to transfer the data from CPU to memory.
- In programmed I/O, the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer.
- This is a time consuming process since it needlessly keeps the CPU busy.
- This situation can be avoided by using an interrupt facility. This is discussed below.

Interrupt- initiated I/O

- Since in the above case we saw the CPU is kept busy unnecessarily.
- This situation can very well be avoided by using an interrupt driven method for data transfer.
- By using interrupt facility and special commands to inform the interface to issue an interrupt request signal whenever data is available from any device.
- In the meantime the CPU can proceed for any other program execution.
- The interface meanwhile keeps monitoring the device. Whenever it is determined that the device is ready for data transfer it initiates an interrupt request signal to the computer.
- Upon detection of an external interrupt signal the CPU stops momentarily the task that it was already performing, branches to the service program to process the I/O transfer, and then return to the task it was originally performing.

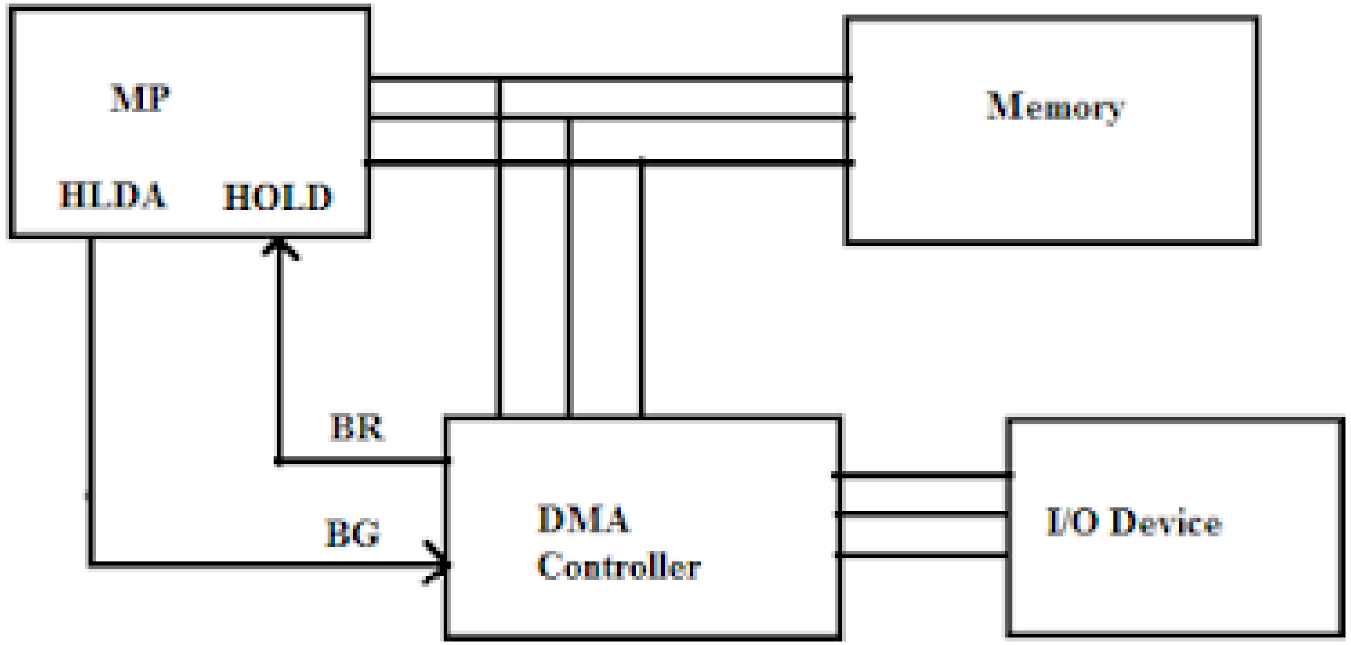
Direct Memory Access

- Direct Memory Access is a technique for transferring data within main memory and external device without passing it through the CPU.
- DMA is a way to improve processor activity and I/O transfer rate by taking-over the job of transferring data from processor, and letting the processor to do other tasks.
- This technique overcomes the drawbacks of other two I/O techniques which are the time consuming process when issuing command for data transfer and tie-up the processor in data transfer while the data processing is neglected.
- It is more efficient to use DMA method when large volume of data has to be transferred.
- For DMA to be implemented, processor has to share its' system bus with the DMA module.
- Therefore, the DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily. The latter technique is more common to be used and it is referred to as cycle stealing.

Direct Memory Access (DMA)

- DMA is a process of communication for data transfer between memory and input/output, controlled by an external circuit called DMA controller, without involvement of CPU.
- 8085 MP has two pins HOLD and HLDA which are used for DMA operation.
- First, DMA controller sends a request by making Bus Request (BR) control line high. When MP receives high signal to HOLD pin, it first completes the execution of current machine cycle, it takes few clocks and sends HLDA signal to the DMA controller.
- After receiving HLDA through Bus Grant (BG) pin of DMA controller, the DMA controller takes control over system bus and transfers data directly between memory and I/O without involvement of CPU. During DMA operation, the processor is free to perform next job which does not need system bus.
- At the end of data transfer, the DMA controller terminates the request by sending low signal to HOLD pin and MP regains control of system bus by making HLDA low.

DMA



Bus Request : It is used by the DMA controller to request the CPU to relinquish the control of the buses.
Bus Grant : It is activated by the CPU to Inform the external DMA controller that the buses are in high impedance state and the requesting DMA can take control of the buses. Once the DMA has taken the control of the buses it transfers the data. This transfer can take place in many ways.
Microprocessor relinquish the system bus and DMA takes control.

Types of DMA transfer using DMAC

- **Burst Transfer**
  - DMA returns the bus after complete data transfer.
- **Cyclic Stealing**
  - An alternative method in which DMA controller transfers one word at a time after which it must return the control of the buses to the CPU.
  - The CPU delays its operation only for one memory cycle to allow the direct memory I/O transfer to “steal” one memory cycle.
- **Interleaved mode**
  - In this technique , the DMA controller takes over the system bus when the microprocessor is not using it. An alternate half cycle i.e. half cycle DMA + half cycle processor.

Advantages and Disadvantages of DMA

- Advantages
  - allows a peripheral device to read from/write to memory without going through the CPU
  - allows for faster processing since the processor can be working on something else while the peripheral can be populating memory
- Disadvantages
  - requires a DMA controller to carry out the operation, which increases the cost of the system

HOLD

- It is active high signal.
- It is requesting the use of the address and data bus.
- After HOLD signal received by the microprocessor, it relinquishes the buses.
- All buses are tri-stated and a HOLD Acknowledgement (HLDA) signal is sent out.
- MPU regains the control of buses after HOLD goes low.

HLDA

- It is Hold Acknowledgement Signal
- This is an active high signal
- It indicates that the MPU is relinquishing the control of the buses