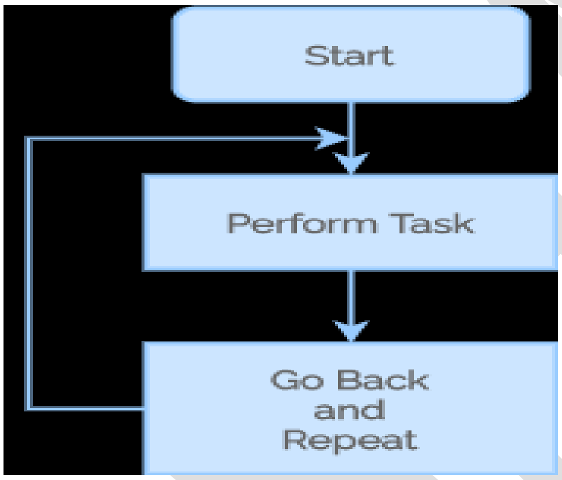
Programming Techniques

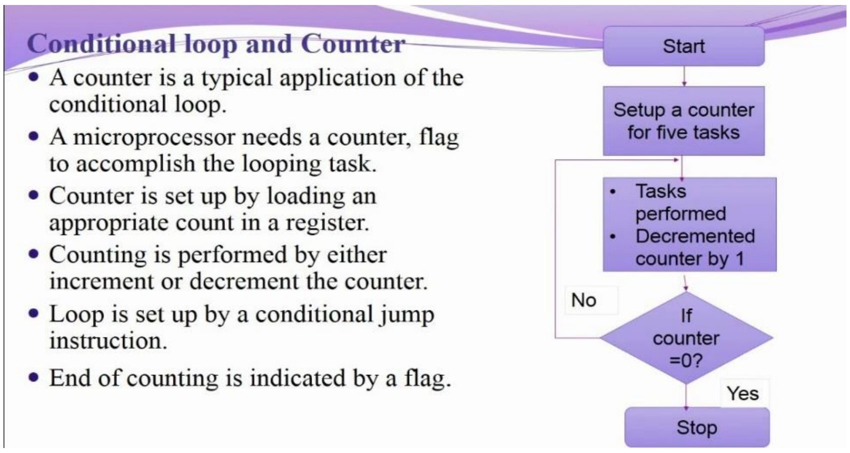
Looping

- The Programming technique used to instruct the microprocessor to repeat task is called looping.
- This process is accomplished by using jump instructions.
- A loop can be classified into two groups:
- Continuous loop – repeats a task continuously
- Conditional loop- repeats a task until certain data condition are met.
- A continuous loop is set up by using the unconditional jump instruction shown in the flowchart.
- Program with Continuous loop does not stop repeating the tasks until the system is reset.



- A Conditional loop is setup by the conditional jump instructions.
- These instructions Checks flags (zero, carry, etc) and repeat the specified task if the conditions are satisfied.

These loops usually include counting and indexing



Conditional loop and Counter

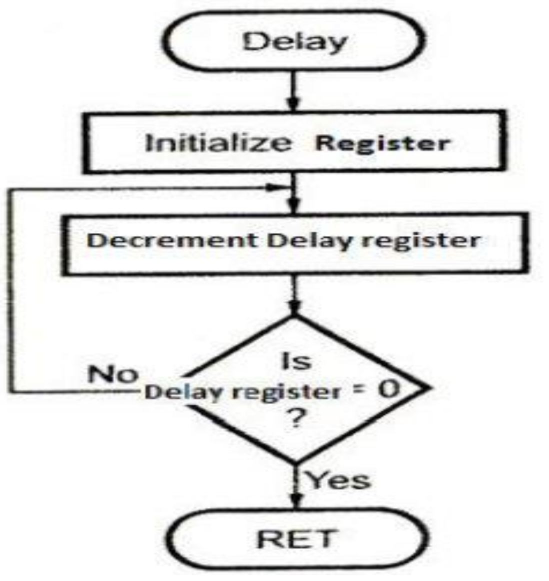
- A counter is a typical application of the conditional loop.
- A microprocessor needs a counter, flag to accomplish the looping task.
- Counter is set up by loading an appropriate count in a register.
- Counting is performed by either increment or decrement the counter.
- Loop is set up by a conditional jump instruction.
- End of counting is indicated by a flag.

Indexing:- This programming Techniques in Microprocessor 8085 allows programmer to point or refer the data stored in sequential memory locations one by one.

Counting:- This technique allows programmer to count how many times the instruction/set of instructions are executed.

Delay Generation in 8085

- It is a procedure used to design a specific delay.
- When the delay subroutine is executed, the microprocessor does not execute other tasks.
- For the delay we are using the instruction execution times.



- A register is loaded with a number, depending on the time delay required and then the register is decremented until it reaches zero by setting up a loop with conditional jump

```
MVI   B, FFH
LOOP:     DCR  B
JNZ       LOOP
```

The first instruction will be executed once, it will take 7 T-states.

DCR B instruction takes 4 T-states.

This will b executed 255(FF) times.

The JNZ instruction takes 10 T-states when it jumps (It jumps 254 times), otherwise it will take 7 T-states.

Suppose clock frequency is 2 MHz

Clock period , T = 1/f =1/(2*10^6) = 0.5 micro second

Time to execute MVI = 7*0.5=3.5 micro seconds

Time delay in loop

(T*T states * N10)=0.5*10^-6*14*255

= 1785 micro seconds

= 1.8ms

Adjusted loop delay

If JNZ instruction is executed in 7 T states,

TLA = TL – (3 T states clock period)

= 1785 – (3*.5)

= 1785-1.5 =1783.5 microseconds

Total delay

Time to execute instruction outside loop + time to execute loop instruction

=(3.5+1783.5)=1787 microseconds = 1.8 ms

STACK IN 8085

- The stack is a LIFO (last in first out) data structure.
- The Stack Pointer register will hold the address of the top location of the stack.
- On a stack, we can perform two operations: PUSH and POP.
- In case of PUSH operation, the SP register gets decreased and new data item used to insert on to the top of the stack.

- On the other hand, in case of POP operation, the data item will have to be deleted from the top of the stack and SP register will get increased.

LXI SP ,8000H

LXI H, 2000H

PUSH H

POP D

HLT

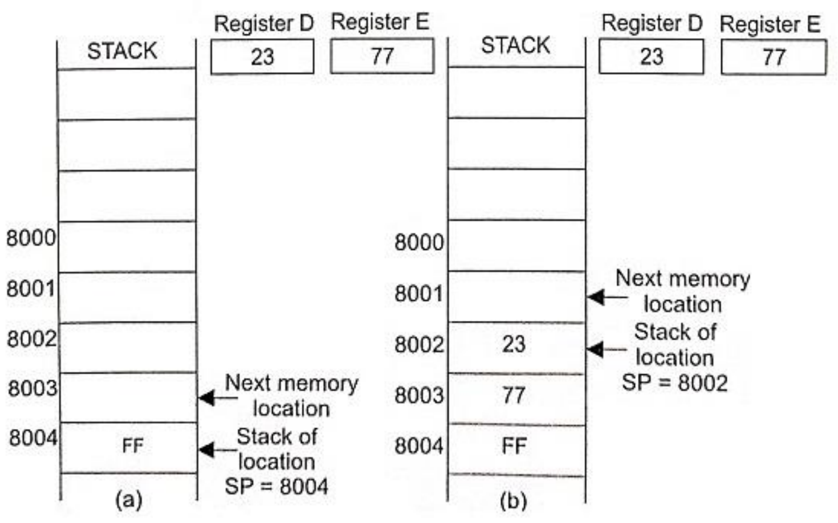


Fig. 4.4 *(a) Stack before PUSH operation (b) Stack after PUSH operation*

- LXI SP,8000H – The address of the stack pointer is set to 8000H
- LXI H,2000H – Next, we add a number to the HL pair.
- The most significant two bits will enter the H register. The least significant two bits will enter the L register.
- PUSH H- The PUSH command will push the contents of the H register first to the stack. Then the contents of the L register will sent to the stack. So the new stack top will hold 00H.
- POP D – The POP command will remove the contents of the stack and store them to the DE register pair.
- The top of the stack clears first and enters the E register.
- The new top of the stack is 20H now.
- This one clears last and enters the D register.
- The contents of the DE register pair is now 2000H.

HLT – HLT indicates that the program execution needs to stop.

Subroutine, call and return instructions and delay programs

- Subroutine is a sequence of program instructions that perform a specific task, packaged as a unit.
- This unit can then be used in programs wherever that particular task have to be performed.
- Subroutine is often coded so that it can be started (called) several times and from several places during one execution of the program, including from other subroutines, and then branch back(return) to the next instruction after the call, once the subroutine's task is done.
- It is implemented by using Call and Return instructions.
- Whenever the instructions in a subroutine are required to be executed, we branch program control to the subroutine using the CALL instruction.
- CALL is a 3-Byte instruction, with 1 Byte for the opcode, and 2 bytes for the address of the subroutine.
- CALL mnemonics stands for "call a subroutine".
- After executing the instructions written in the subroutine we shall want to return control to the next instruction written after the CALL instruction then we shall use mnemonic RET.
- Here RET stands for RETurn from the subroutine.
- RET is a 1-Byte instruction.
- CALL : To redirect program execution to the subroutine.
- RET : Return execution to calling routine.
- Subroutine also known as procedure, subprogram, function, routine, method.
- Also used for delaying the program execution for a specific time – Delay subroutines or delay program

CC	**Call subroutine if Carry flag is set (CY=1)**
CNC	**Call subroutine if Carry flag is reset (CY=0)**
CZ	**Call subroutine if Zero flag is set(Z=1)**
CNZ	**Call subroutine if Zero flag is reset (Z=0)**
CM	**Call subroutine if Sign flag is set (S=1, negative number)**
CP	**Call subroutine if Sign flag is reset (S=0, positive number)**
RC	**Return if Carry flag is set (CY =1)**

RNC	**Return if Carry flag is reset(CY=0)**
RZ	**Return if Zero flag is set (Z=1)**
RNZ	**Return id Zero flag is reset(Z=0)**
RM	**Return if Sign flag is set(S=1, negative number)**
RP	**Return if Sign flag is reset (S=0, positive number)**
RPE	**Return if Parity flag is set (P=1, even parity)**
RPO	**Return if Parity flag is reset(P=0, odd parity)**

Interrupts in 8085

- Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor and get itself serviced.
- Generally, a particular task is assigned to that interrupt signal.
- It is a signal to the processor initiated by hardware or software indicating an event that need immediate attention.
- A small program or a routine that when executed services the corresponding interrupting source is called as an ISR(Interrupt service routine).
- ISR tells the processor what to do when interrupt occurs.
- There are 4 Maskable Interrupt and 1 Non-Maskable Interrupt.

There are two types of interrupts used in 8085 Microprocessor:

- **Hardware Interrupts**
- **Software Interrupts**

There are 6 pins available in 8085 for hardware interrupt.

- **TRAP**
- **RST 7.5**
- **RST 6.5**
- **RST 5.5**
- **INTR**
- **INTA**

Software interrupts is a particular instructions that can be inserted into the desired location in the program. There are eight software interrupts in 8085 Microprocessor. From RST0 to RST7.

- **RST0**
- **RST1**
- **RST2**
- **RST3**
- **RST4**
- **RST5**
- **RST6**
- **RST7**

Vectored Interrupts: Fixed vector address (starting address of subroutine)

Non vectored interrupt: Vector address is not predefined.

- Interrupting device gives the address of subroutine

Instruction for Interrupts-

1.Enable Interrupt (EI) – The interrupt enable flip-flop is set and all interrupts are enabled following the execution of next instruction followed by EI.

2.Disable Interrupt(DI)- This instruction is used to reset the value of enable flip-flop hence disabling all the interrupts.

3. SIM : Set interrupt mask

4. RIM : read interrupt mask

Interrupt process

1.When the microprocessor is executing a program, it checks the INTR line during execution of each instruction.

2.If INTR is high, and the interrupt is enabled, microprocessor completes the current instruction, disables interrupt enable flip-flop. Sends a signal is called INTA(Interrupt acknowledged)

- Processor can't accept any interrupt until interrupt flip-flop is enabled.

3. INTA is used to insert a restart(RST) instruction through external hardware. RST instruction is a 1 byte call instruction that transfers program control to a specific memory location and restart the execution at that memory location after executing step 4.

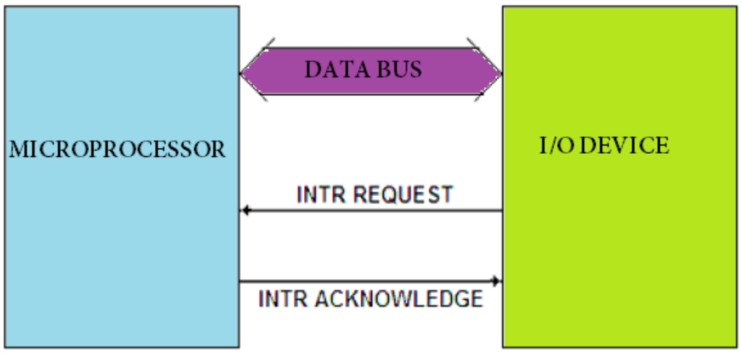
4. When the microprocessor receives an RST(CALL) instruction, it saves the memory address of next instruction on the stack. Program is transferred to CALL location.

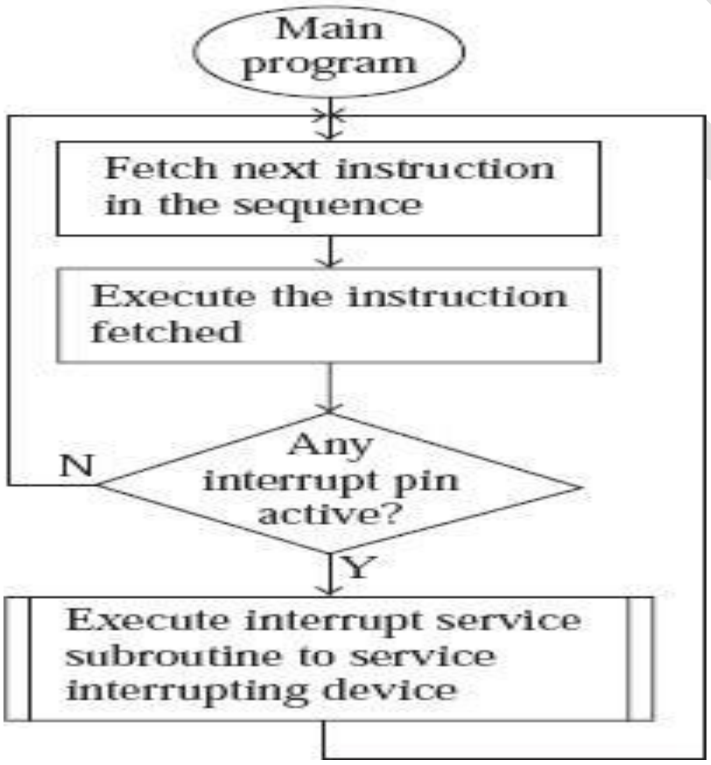
5. Executing ISR.

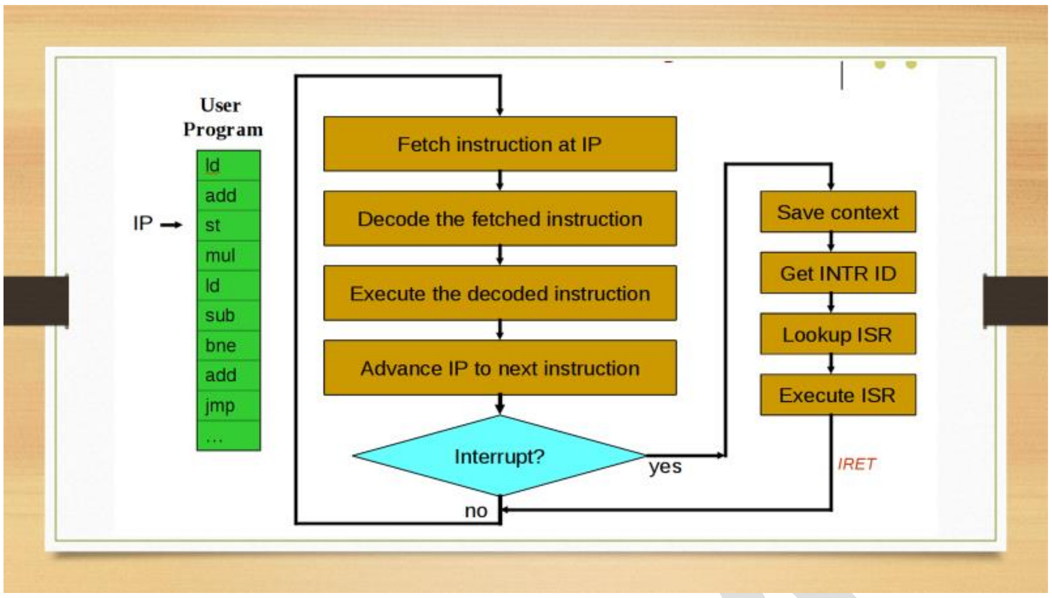
- ISR includes EI to enable the interrupt again.
- At the end of subroutine, RET instruction retrieves Memory address where the program was interrupted and Continues execution.

Interrupt driven programs

- Interrupt I/O is a way of controlling input/output activity whereby a peripheral or terminal that needs to make or receive a data transfer sends a signal.
- This will cause a program interrupt to be set.
- Relative to the total interrupt system, the processors enter an interrupt service routine.




Main program
Fetch next instruction in the sequence
Execute the instruction fetched
Any interrupt pin active?
N
Y
Execute interrupt service subroutine to service interrupting device



Memory Interfacing in 8085

- The Memory Interfacing in 8085 is used to access memory quite frequently to read instruction codes and data stored in memory.
- This read/write operations are monitored by control signals.
- The microprocessor activates these signals when it wants to read from and write into memory.

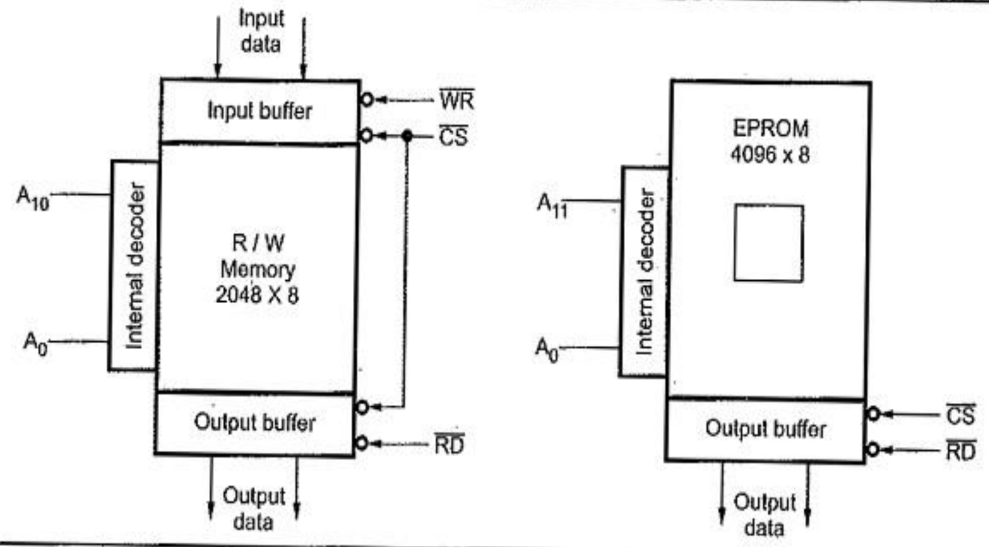


Fig. 4.13 (a) Logic diagram for RAM Fig. 4.13 (b) Logic diagram for EPROM

- For Memory Interfacing in 8085, following important points are to be kept in mind.
- Microprocessor 8085 can access 64Kbytes memory since address bus is 16-bit. But it is not always necessary to use full 64Kbytes address space. The total memory size depends upon the application.

- Generally EPROM (or EPROMs) is used as a program memory and RAM (or RAMs) as a data memory. When both, EPROM and RAM are used, the total address space 64Kbytes is shared by them.
- The capacity of program memory and data memory depends on the application.
- It is not always necessary to select 1 EPROM and 1 RAM. We can have multiple EPROMs and multiple RAMs as per the requirement of application.
- We can place EPROM/RAM anywhere in full 64 Kbytes address space. But program memory (EPROM) should be located from address 0000H since reset address of 8085 microprocessor is 0000H.
- It is not always necessary to locate EPROM and RAM in consecutive memory For example : If the mapping of EPROM is from 0000H to OFFFH, it is not must to locate RAM from 1000H. We can locate it anywhere between 1000H and FFFFH. Where to locate memory component totally depends on the application.

The memory interfacing requires to **:**

Select the chip

Identify the register

Enable the appropriate buffer.

- Microprocessor system includes memory devices and I/O devices.
- It is important to note that microprocessor can communicate (read/write) with only one device at a time, since the data, address and control buses are common for all the devices.
- In order to communicate with memory or I/O devices, it is necessary to decode the address from the microprocessor.
- Due to this each device (memory or I/O) can be accessed independently.

Programmable Peripheral Interface 8255A

- The 8255A is a general purpose programmable I/O device designed to transfer the data from I/O to interrupt I/O under certain conditions as required.
- It can be used with almost any microprocessor.
- It consists of three 8-bit bidirectional I/O ports (24I/O lines) which can be configured as per the requirement.

Ports of 8255A

8255A has three ports, i.e., PORT A, PORT B, and PORT C.

- **Port A** contains one 8-bit output latch/buffer and one 8-bit input buffer.

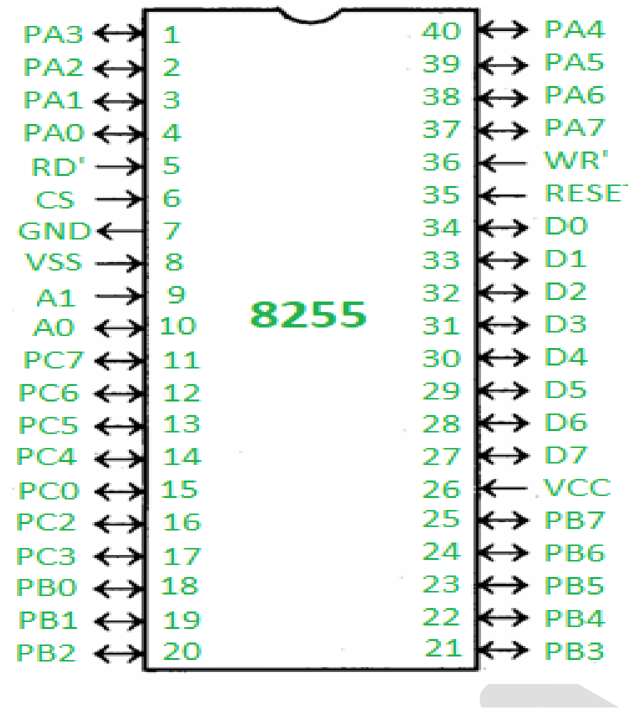
- **Port B** is similar to PORT A.
- **Port C** can be split into two parts, i.e. PORT C lower (PC0-PC3) and PORT C upper (PC7-PC4) by the control word
- These three ports are further divided into two groups, i.e. Group A includes PORT A and upper PORT C.
- Group B includes PORT B and lower PORT C.
- These two groups can be programmed in three different modes, i.e. the first mode is named as mode 0, the second mode is named as Mode 1 and the third mode is named as Mode 2.

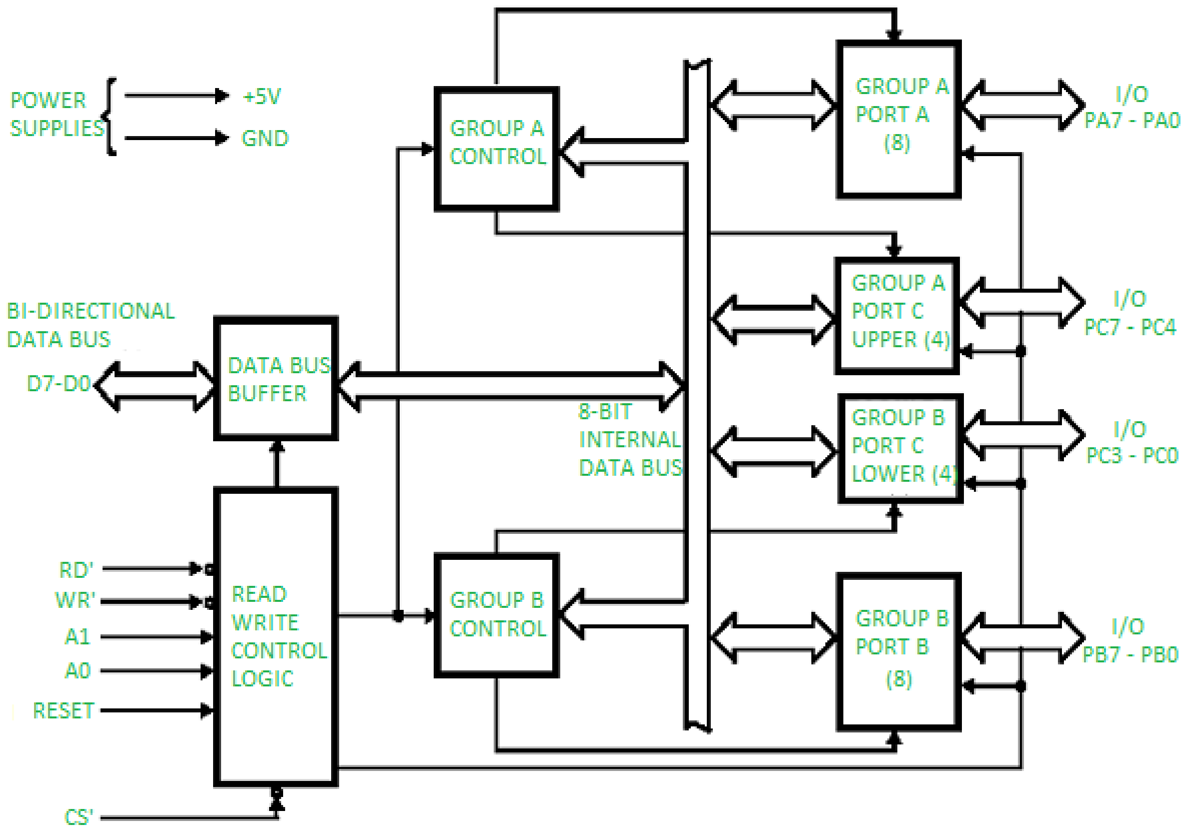
Operating Modes

- 8255A has three different operating modes −
- **Mode 0** − In this mode, Port A and B is used as two 8-bit ports and Port C as two 4-bit ports. Each port can be programmed in either input mode or output mode where outputs are latched and inputs are not latched. Ports do not have interrupt capability.
- **Mode 1** − In this mode, Port A and B is used as 8-bit I/O ports. They can be configured as either input or output ports. Each port uses three lines from port C as handshake signals. Inputs and outputs are latched.
- **Mode 2** − In this mode, Port A can be configured as the bidirectional port and Port B either in Mode 0 or Mode 1. Port A uses five signals from Port C as handshake signals for data transfer. The remaining three signals from Port C can be used either as simple I/O or as handshake for port B.

Features of 8255A

- The prominent features of 8255A are as follows −
- It consists of 3 8-bit IO ports i.e. PA, PB, and PC.
- Address/data bus must be externally demux'd.
- It is TTL compatible.
- It has improved DC driving capability





Data Bus Buffer

- It is a tri-state 8-bit buffer, which is used to interface the microprocessor to the system data bus. Data is transmitted or received by the buffer as per the instructions by the CPU. Control words and status information is also transferred using this bus.

Read/Write Control Logic

- This block is responsible for controlling the internal/external transfer of data/control/status word. It accepts the input from the CPU address and control buses, and in turn issues command to both the control groups.

CS

- It stands for Chip Select. A LOW on this input selects the chip and enables the communication between the 8255A and the CPU. It is connected to the decoded address, and A_0 & A_1 are connected to the microprocessor address lines.

CS	**A_1**	**A_0**	**Result**
0	0	0	PORT A
0	0	1	PORT B
0	1	0	PORT C
0	1	1	Control Register
1	X	X	No Selection

WR

- It stands for write. This control signal enables the write operation. When this signal goes low, the microprocessor writes into a selected I/O port or control register.

RESET

- This is an active high signal. It clears the control register and sets all ports in the input mode.

RD

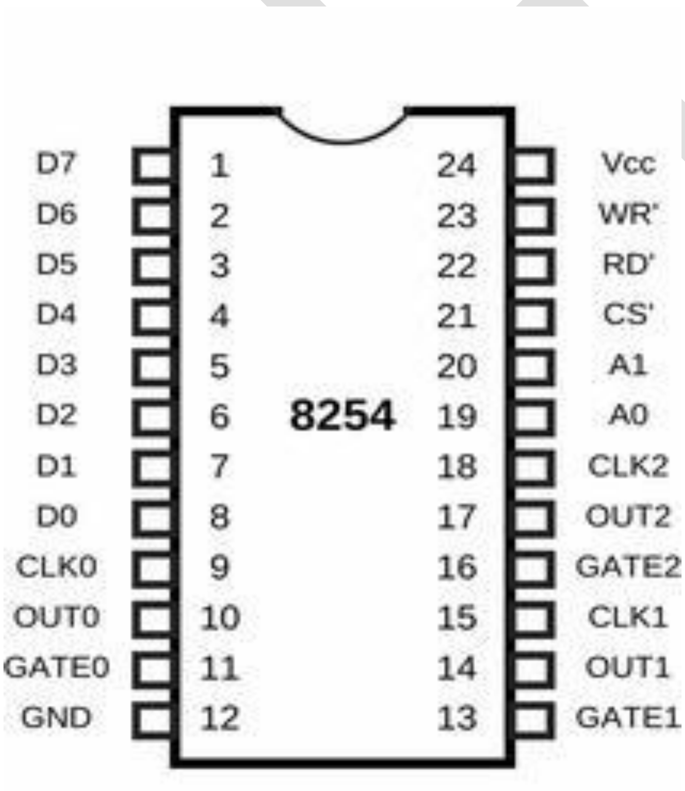
- It stands for Read. This control signal enables the Read operation. When the signal is low, the microprocessor reads the data from the selected I/O port of the 8255.
- A_0 and A_1
- These input signals work with RD, WR, and one of the control signal. Following is the table showing their various signals with their result.

A_1	A_0	RD	WR	CS	Result
0	0	0	1	0	**Input Operation** PORT A → Data Bus
0	1	0	1	0	PORT B → Data Bus
1	0	0	1	0	PORT C → Data Bus
0	0	1	0	0	**Output Operation** Data Bus → PORT A
0	1	1	0	0	Data Bus → PORT A
1	0	1	0	0	Data Bus → PORT B
1	1	1	0	0	Data Bus → PORT D

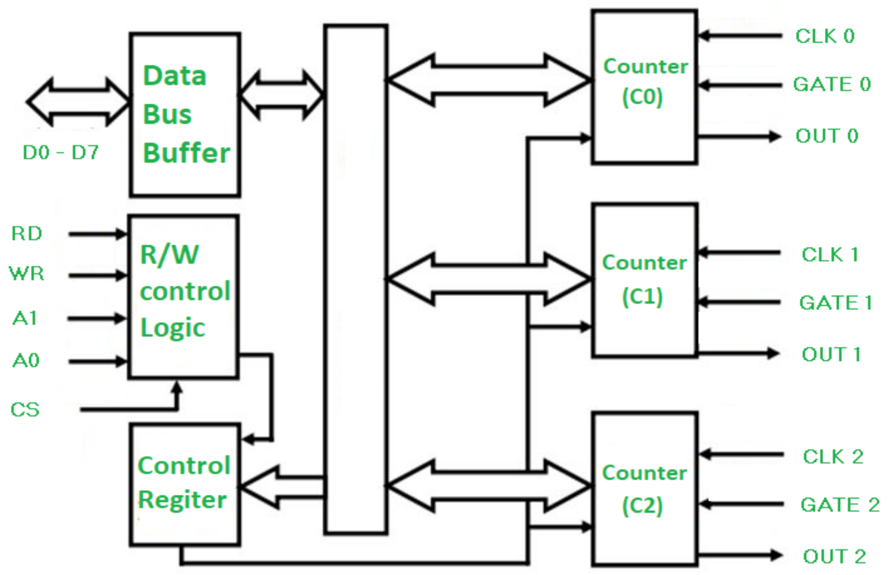
8254 Programmable Interval Timer

- 8254 is a device designed to solve the timing control problems in a microprocessor.
- It has 3 independent counters, each capable of handling clock inputs up to 10 MHz, and size of each counter is 16 bit.
- It operates in +5V regulated power supply and has 24 pin signals

The pin diagram for 8254 is



The basic block diagram of 8254 is:



- It has 3 counters each with two inputs (Clock and Gate) and one output. Gate is used to enable or disable counting. When any value of count is loaded and value of gate is set(1), after every step value of count is decremented by 1 until it becomes zero.
- Depending upon the value of CS, A1, and A0 we can determine the addresses of the selected counter.

CS	A1	A0	SELECTION
0	0	0	C0
0	0	1	C1
0	1	0	C2
0	1	1	Control Register

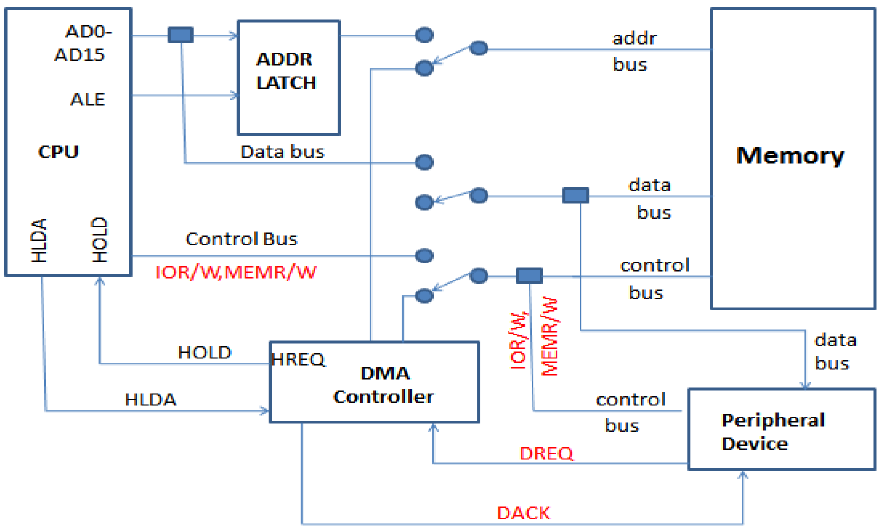
Applications –

- To generate an accurate time delay
- As an event counter
- Square wave generator
- Rate generator
- Digital one shot

8237: DMA CONTROLLER

- Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU).
- It is also a fast way of transferring data within (and sometimes between) computer.
- The DMA I/O technique provides direct access to the memory while the microprocessor is temporarily disabled.
- The DMA controller temporarily borrows the address bus, data bus and control bus from the microprocessor and transfers the data directly from the external devices to a series of memory locations (and vice versa).
- Two control signals are used to request and acknowledge a direct memory access (DMA) transfer in the microprocessor-based system.
    - The HOLD signal as an input(to the processor) is used to request a DMA action.
    - The HLDA signal as an output that acknowledges the DMA action.
- When the processor recognizes the hold, it stops its execution and enters hold cycles.
- HOLD input has higher priority than INTR or NMI.
- The only microprocessor pin that has a higher priority than a HOLD is the RESET pin.
- HLDA becomes active to indicate that the processor has placed its buses at high-impedance state.
- Direct memory accesses normally occur between an I/O device and memory without the use of the microprocessor.
    - A DMA read transfers data from the memory to the I/O device.
    - A DMA write transfers data from an I/O device to memory.

- The system contains separate memory and I/O control signals.
- Hence the Memory & the I/O are controlled simultaneously
- The DMA controller provides memory with its address, and the controller signal selects the I/O device during the transfer.
- Data transfer speed is determined by speed of the memory device or a DMA controller.
- In many cases, the DMA controller slows the speed of the system when transfers occur.
- The serial PCI (Peripheral Component Interface) Express bus transfers data at rates exceeding DMA transfers.
- This in modern systems has made DMA is less important.



- When the DMA controller sees a DMA request, it responds by performing one or many data transfers from that I/O device into system memory or vice versa.
- Channels must be enabled by the processor for the DMA controller to respond to DMA requests.
- The number of transfers performed, transfer modes used, and memory locations accessed depends on how the DMA channel is programmed.

A DMA controller typically shares the system memory and I/O bus with the CPU and has both bus master and slave capability

- In bus master mode, the DMA controller acquires the system bus (address, data, and control lines) from the CPU to perform the DMA transfers.
- Because the CPU releases the system bus for the duration of the transfer, the process is sometimes referred to as cycle stealing.

- –In bus slave mode, the DMA controller is accessed by the CPU, which programs the DMA controller's internal registers to set up DMA transfers.

The internal registers consist of source and destination address registers and transfer count registers for each DMA channel, as well as control and status registers for initiating, monitoring, and sustaining the operation of the DMA controller

****