RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL

New Scheme Based On AICTE Flexible Curricula

Electronics & Communication Engineering V-Semester

EC501 MICROPROCESSOR AND ITS APPLICATIONS

UNIT I Salient features of advanced microprocessors. RISC & CISC processors. Review and evolution of advanced microprocessors:8086,8088, 80186/286/386/486/Pentium, introduction to 8086 processor: Register organization of 8086,Architecture,signal description of 8086,minimum mode 8086 systems and timings and maximum mode 8086 systems and timings

UNIT II Intel 8086 microprocessor programming: 8086 Instruction Set, Addressing modes, Assembly Language Programming with Intel 8086 microprocessor

UNIT III Introduction to the various interfacings chips like 8155, 8255, Interfacings key boards, LEDs , ADC, DAC and memory Interfacing.

UNIT IV General purposes programmable peripheral devices (8253), 8254 programmable interval timer, 8259A programmable interrupt controller & 8257 DMA controller, USART, serial I/O & data Communication .

UNIT V Introduction to microcontrollers (8051) and embedded systems: 8051 architecture, pin description , I/O configuration , interrupts, addressing modes, an overview of 8051 instruction set, embedded system, use of microcontrollers in embedded systems

Reference Books:

1. Advance microprocessor and peripheral –A.K. Ray and K. M. Bhurchandi, Tata Mcgraw Hill
2. Microprocessor and Interfacing – D.V.Hall, McGraw Hill.
3. The Intel microprocessor - Barry B. Brey, Pearson
4. The 8086 & 8088 Microprocessor- LIU and Gibson, Tata McGraw Hill
5. The 8051 microcontroller and embedded systems-M.A. Mazidi, Janice GillispieMazidi, Pearson Prentice Hall

UNIT I: Salient features of advanced microprocessors. RISC & CISC processors. Review and evolution of advanced microprocessors:8086,8088, 80186/286/386/486/Pentium, introduction to 8086 processor: Register organization of 8086, Architecture, signal description of 8086, minimum mode 8086 systems and timings and maximum mode 8086 systems and timings

---

RISC & CISC processors:

Sl. No.	RISC Architecture	CISC Architecture
i	RISC stands for Reduced Instruction Set Computer.	CISC stands for Complex Instruction Set Computer.
ii	RISC processors have simple instructions taking about one clock cycle.	CSIC processor has complex instructions that take up multiple clocks for execution.
iii	Performance is optimized with more focus on software	Performance is optimized with more focus on hardware.
iv	It has no memory unit and uses a separate hardware to implement instructions.	It has a memory unit to implement complex instructions.
v	It has a hard-wired unit of programming.	It has a microprogramming unit.

Architecture of 8086 Microprocessor: The microprocessor 8086 architecture is divided into two parts, bus interface unit (BIU) and execution unit (EU). The basic principle in this architecture is instruction pipelining and memory segmentation which increases the processing capability.

Bus Interface Unit (BIU) : This unit establish the communication between the external devices and the processor. The BIU consists of address conversion block which converts 16-bit offset and 16-bit segment address in to 20-bit physical address. It includes the 6-bytes instruction queue for storing the instruction to be executed. The segment registers hold the address of the respective segments and instruction pointer register holds the address of the next instruction to be executed.

Execution Unit (EU) : The execution unit (EU) of the 8086 tells the BIU where to fetch instructions or data from, decodes instructions and executes instruction. The EU contains control circuitry which directs internal operations. A decoder in the EU translates instructions fetched from memory into a series of actions which the EU carries out. The EU is has a 16-bit ALU which can add, subtract, AND, OR, XOR, increment, decrement, complement or shift binary numbers. The EU is decoding an instruction or executing an instruction which does not require use of the buses.

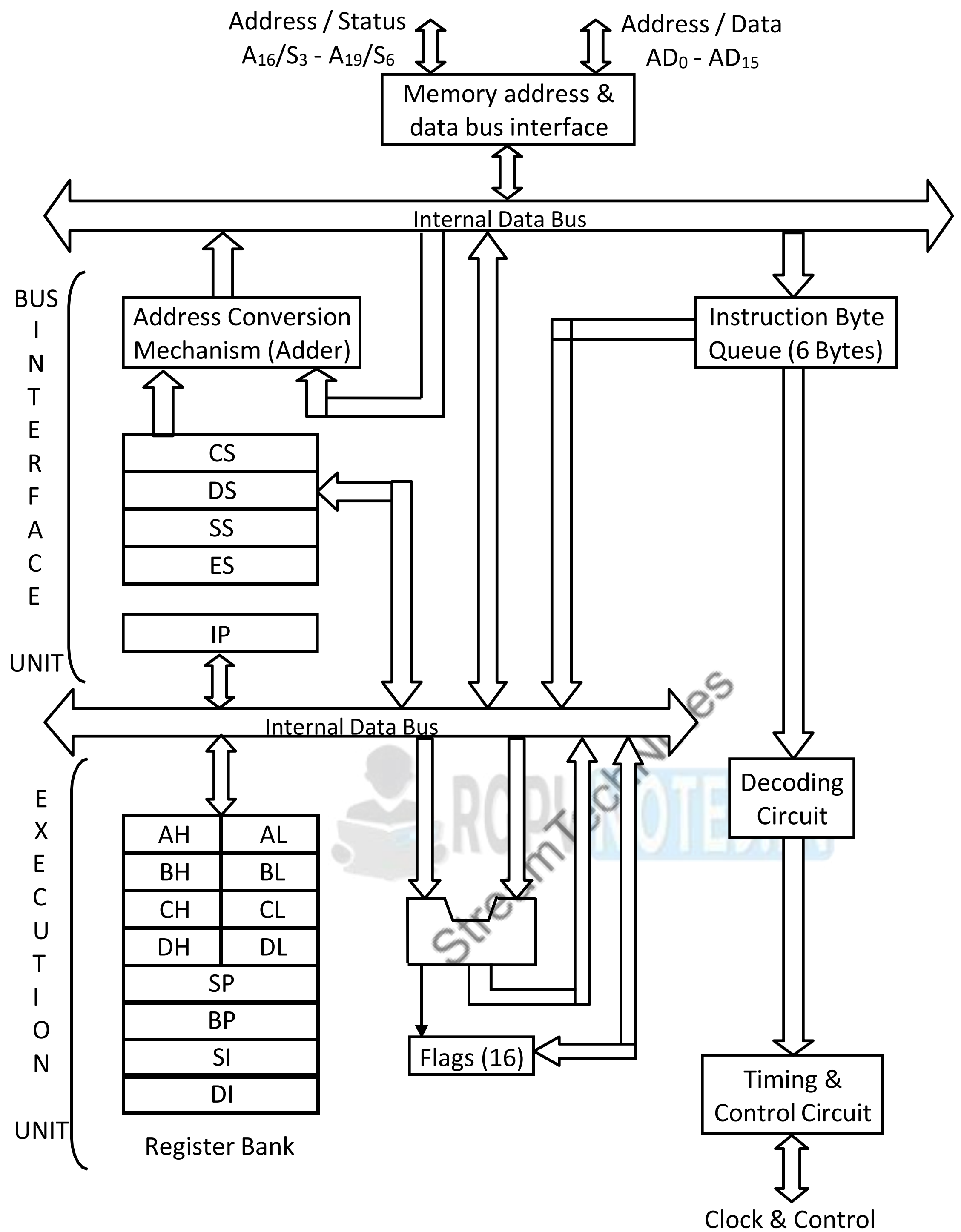


Fig.1.1 : Internal Architecture of 8086 Microprocessor

Register Organisation:

General Purpose Registers:

The registers AX,BX,CX and DX are the four general purpose 16-bit registers. These registers can also be used as 8-bit register of higher (H) and lower order(L). The general purpose registers have some special functions as mentioned below.

AX - used as 16-bit accumulator, with the lower 8-bits designated as AL and higher 8-bits as AH.

BX - used as an offset storage for forming physical addresses in case of certain addressing modes.

CX – used as a default counter in case of string and loop instruction.

DX – used as implicit operand or destination in case of a few instruction.

Segment Registers:
There are four 16-bit segment registers CS,DS,ES and SS used to point to the memory segments. Instead of using 20-bit register for a physical address, the processor just maintain the two 16-bit register of segment address and offset address.
CS – Code segment register used to hold the address of the code segment of the memory space.
DS – Data segment register used to hold the address of the current data segment of the memory space.
ES – Extra segment register which also pointing to the data segment
SS – Stack segment register is pointing to the top of the stack space in the memory.

Pointers and index registers.
The pointers contain within the particular segments. The pointers IP, BP, SP usually contain offsets within the code, data and stack segments respectively
SP -- Stack Pointer is a 16-bit register pointing to program stack in stack segment.
BP -- Base Pointer is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.
SI -- Source Index is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.
DI -- Destination Index is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

Flag Register:
The 8086 flag register as shown in the fig 1.6. 8086 has 9 active flags and they are divided into two categories:
1. Conditional Flags
2. Control Flags

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	O	D	I	T	S	Z	X	AC	X	P	X	CY

O-Overflow Flag D- Direction Flag
I- Interrupt Flag T- Trap Flag
S-Sign Flag Z-Zero Flag
AC-Auxiliary Carry Flag P –Parity Flag
CY-Carry Flag X- Not Used

Conditional Flags : Conditional flags are as follows:
Carry Flag (CY): This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.
Auxiliary Flag (AC): If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AC flag is set i.e. carry given by D3 bit to D4 is AC flag. This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.
Parity Flag (PF): This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity flag is reset.
Zero Flag (ZF): It is set; if the result of arithmetic or logical operation is zero else it is reset.
Sign Flag (SF): In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

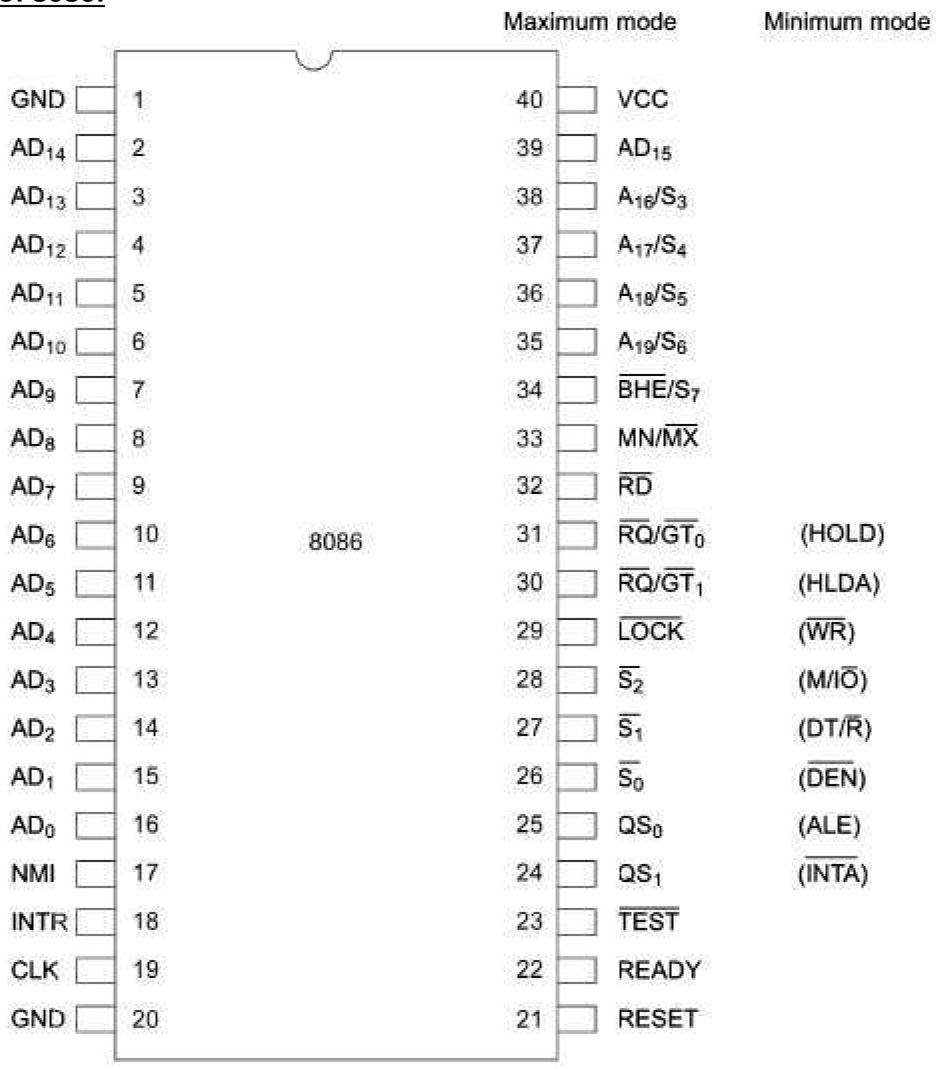
Control Flags: Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows

Trap Flag (TF): It is used for single step control. It allows user to execute one instruction of a program at a time for debugging. When trap flag is set, program can be run in single step mode.

Interrupt Flag (IF): It is an interrupt enable/disable flag. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled. It can be set by executing instruction sit and can be cleared by executing CLI instruction.

Direction Flag (DF): It is used in string operation. If it is set, string bytes are accessed from higher memory address to lower memory address. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

Signal Description of 8086:



AD_{15}-AD_0: These are the time multiplexed memory I/O address and data lines. Address remains on the lines during T_1 state, while the data is available on the data bus during T_2, T_3, T_W and T_4. Here T_1, T_2, T_3, T_4 and T_W are the clock states of a machine cycle. T_W is a wait state. These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3: These are the time multiplexed address and status lines. During T_1, these are the most significant address lines or memory operations. During I/O operations, these lines are low. During memory or I/O operations, status information is available on those lines for T_2, T_3, T_W and T_4 .The status of the interrupt enable flag bit(displayed on S_5) is updated at the beginning of each clock cycle. The S_4 and S_3

combinedly indicate which segment register is presently being used for memory accesses as shown in Table. These lines float to tri-state off (tristated) during the local bus hold acknowledge. The status line S6 is always low(logical). The address bits are separated from the status bits using latches controlled by the ALE signal.

BHE/S_7-Bus High Enable/Status: The bus high enable signal is used to indicate the transfer of data over the higher order (D_{15}-D_8) data bus as shown in Table 1.2. It goes low for the data transfers over D_{15}-D_8 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T_1 for read, write and interrupt acknowledge cycles, when- ever a byte is to be transferred on the higher byte of the data bus. The status information is available during T_2, T_3 and T_4. The signal is active low and is tristated during 'hold'. It is low during T_1 for the first pulse of the interrupt acknowledge cycle.

RD-Read: Read signal, when low, indicates the peripherals that the processor is performing a memory or I/O read operation. RD is active low and shows the state for T2, T3, TW of any read cycle. The signal remains tristated during the 'hold acknowledge'.
READY: This is the acknowledgement from the slow devices or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086. The signal is active high.
INTR-lnterrupt Request: This is a level triggered input. This is sampled during the last clock cycle of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle. This can be internally masked by resetting the interrupt enable flag. This signal is active high and internally synchronized.
TEST: This input is examined by a 'WAIT' instruction. If the TEST input goes low, execution will continue, else, the processor remains in an idle state. The input is synchronized internally during each clock cycle on leading edge of clock.
NMI-Non-maskable Interrupt: This is an edge-triggered input which causes a Type2 interrrupt. The NMI is not maskable internally by software. A transition from low to high initiates the interrupt response at the end of the current instruction. This input is internally synchronized.
RESET: This input causes the processor to terminate the current activity and start execution from FFFF0H. The signal is active high and must be active for at least four clock cycles. It restarts execution when the RESET returns low. RESET is also internally synchronized.
CLK-Clock Input: The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle. The range of frequency for different 8086 versions is from 5MHz to 10MHz.
VCC : +5V power supply for the operation of the internal circuit. GND ground for the internal circuit.

MN/MX :The logic level at this pin decides whether the processor is to operate in either minimum (single processor) or maximum (multiprocessor) mode.

The following pin functions are for the minimum mode operation of 8086.
M/IO -Memory/IO: This is a status line logically equivalent to S_2 in maximum mode. When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation. This line becomes active in the previous T_4 and remains active till final T_4 of the current cycle. It is tristated during local bus "hold acknowledge".
INTA -Interrupt Acknowledge: This signal is used as a read strobe for interrupt acknowledge cycles. In other words, when it goes low, it means that the processor has accepted the interrupt. It is active low during T_2, T_3 and T_W of each interrupt acknowledge cycle.

ALE-Address latch Enable: This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high and is never tristated.
DT /R -Data Transmit/Receive: This output is used to decide the direction of data flow through the transreceivers (bidirectional buffers). When the processor sends out data, this signal is high and when the processor is receiving data, this signal is low. Logically, this is equivalent to S_1 in maximum mode. Its timing is the same as M/I/O. This is tristated during 'hold acknowledge'.
DEN-Data Enable This signal indicates the availability of valid data over the address/data lines. It is used to enable the transreceivers (bidirectional buffers) to separate the data from the multiplexed address/data signal. It is active from the middle of T_2 until the middle of T_4 DEN is tristated during 'hold acknowledge' cycle.
HOLD, HLDA-Hold/Hold Acknowledge: When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access. The processor, after receiving the HOLD request, issues the hold acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus (instruction) cycle. At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLDA signal. HOLD is an asynchronous input, and it should be externally synchronized.

The following pin functions are for the maximum mode operation of 8086.
$\overline{QS_1}$ and $\overline{QS_0}$ – Queue status: These status signals indicate the code pre-fetch queue. These are active during the CLK cycle after the queue operation is performed. The 6-bytes instruction queue which store the instruction called as pre-fetch queue provides the pipelined processing feature in 8086. The queue status indicates as mentioned

QS_1	QS_0	Indication
0	0	No operation
0	1	First byte of opcode from the queue
1	0	Empty queue
1	1	Subsequent byte from the queue

S_2, S_1, S_0 – Status Lines:

S_2	S_1	S_0	Indication
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

$\overline{\text{LOCK}}$: This output pin indicates the use of system bus by the other master when this signal is low.
RQ / GT_0 , RQ / GT_1: These pins are used by other local bus master, in maximum mode operation . The local bus are release at the end of the processor's current bus cycle. Each of these pins are bidirectional with RQ/GT_0 having highest priority than RQ/GT_1

Memory Organisation:

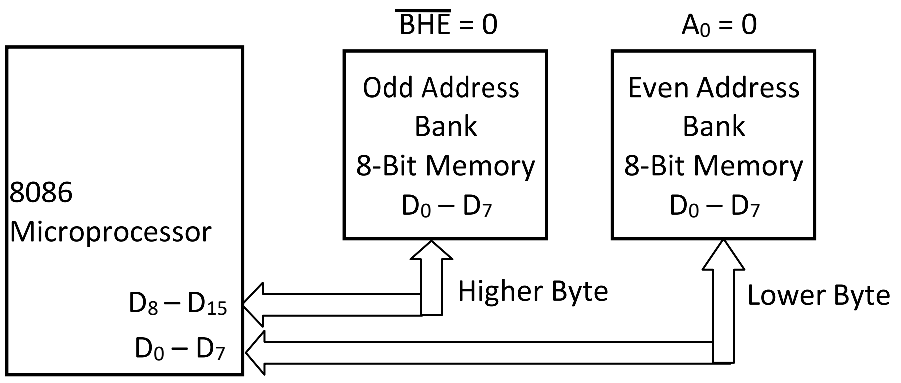


Fig.1.2: Physical Memory Organisation

The physical memory in 8086 is organised as odd and even bank. The total 1Mbytes of memory is divided into two 512Mbytes memory bank each. The arrangements of these memory bank is as shown in Fig.2.b. The lower byte data lines are connected with even memory bank and higher byte data lines are connected with odd memory bank. A_0 is used to select the even memory bank and the BHE line is used to select the odd memory bank. The read and write cycle will operate to activate both the banks or only one bank based on the length of the data. Since the 8086 is capable of operating on 16-bit data, with this type of configuration it is possible to operate on 16-bit at a time.

Clock Generator 8284: The 8284A is an 18-pin IC designed for use with the 8086 microprocessors. The following is a list of each pin and its function. The pin details are as shown in Fig. .

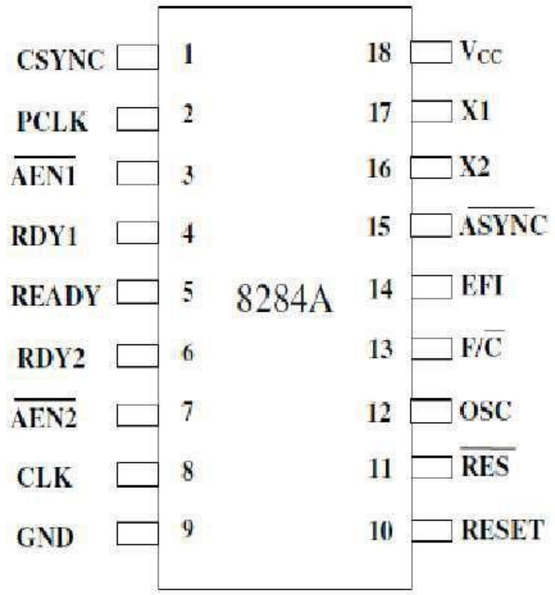


Fig. 1.3 Pin Diagram of Clock Generator

$\overline{AEN1}$ and $\overline{AEN2}$: The address enable pins are provided to qualify the ready signals. RDY1 and RDY2, respectively. Which are used to cause wait states, along with the RDY1 and RDY2 inputs. Wait states are generated by the READY pin of the 8086microprocessor. This is controlled by these two inputs.

RDY1 and RDY2: The bus ready inputs are provided in conjunction with the $\overline{AEN1}$ and $\overline{AEN2}$ pins to cause wait states in an 8086 microprocessor based system.

ASYNC: The ready synchronization selection input selects either one or two stages of synchronization for the RDY1 and RDY2 inputs.
READY: Ready is an output pin that connects to the 8086 microprocessor READY input. This signal is synchronized with the RDY1 and RDY2 inputs.
X1 and X2: The Crystal Oscillator pins connect to an external crystal used as the timing source for the clock generator and all its functions.
F/C*: The Frequency/Crystal select input results the clocking source for the 8284A. If this pin is held high, an external clock is provided to the EFI input pin, and if it is held low, the internal crystal oscillator provides the timing signal.
EFI: The External Frequency input is used when the F/C is pulled high. EFI supplies the timing whenever the F/C* pin is high.
CLK: The clock output pin provides CLK input signal to the 8086 microprocessors and other components in the system. The CLK pin has an output signal that is one-third of the crystal or EFI input frequency and has a 33% duty cycle, which is required by the 8086 microprocessors.
PCLK: The Peripheral Clock signal is one-sixth the crystal or EFI input frequency and has a 50 percent duty cycle. The PCLK output provides a clock signal to the peripheral equipment in the system.
OSC: The Oscillator output is a TTL level signal that is at the same frequency as the crystal or EFI input.
RES*: The reset input is an active-low input to the 8284A. The RES* pin is often connected an RC network that provides power-on resetting.
RESET: The Reset output is connected to the 8086/8088 microprocessors RESET input pin.
CSYNC: The clock synchronization pin is used whenever the EFI input provides synchronization in systems with multiple processors. When the internal crystal oscillator is used, this pin must be grounded.
GND: The ground pin is connects to ground.
Vcc: This power supply pin connects to + 5.0V with a tolerance of ± 10 percent

Minimum Mode:

In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX* pin to logic1. In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system. The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system. The latches are generally buffered output D-type flip-flops, like, 74LS373 or 8282. They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086. Transreceivers are the bidirectional buffers and sometimes they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signal. They are controlled by two signals, namely, DEN* and DT/R*. The DEN* signal indicates that the valid data is available on the data bus, while DT/R indicates the direction of data, i.e. from or to the processor. The system contains memory for the monitor and users program storage.

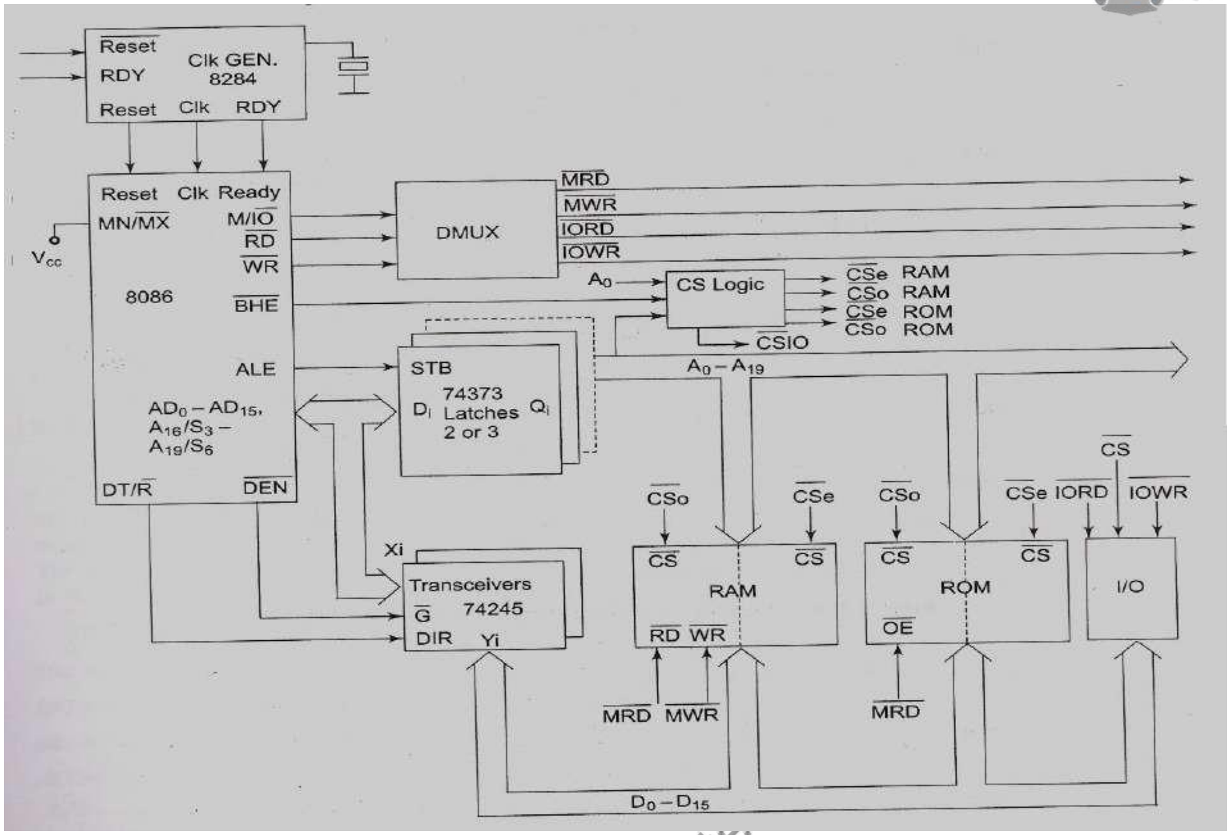


Fig.1.4: Minimum Mode Configuration

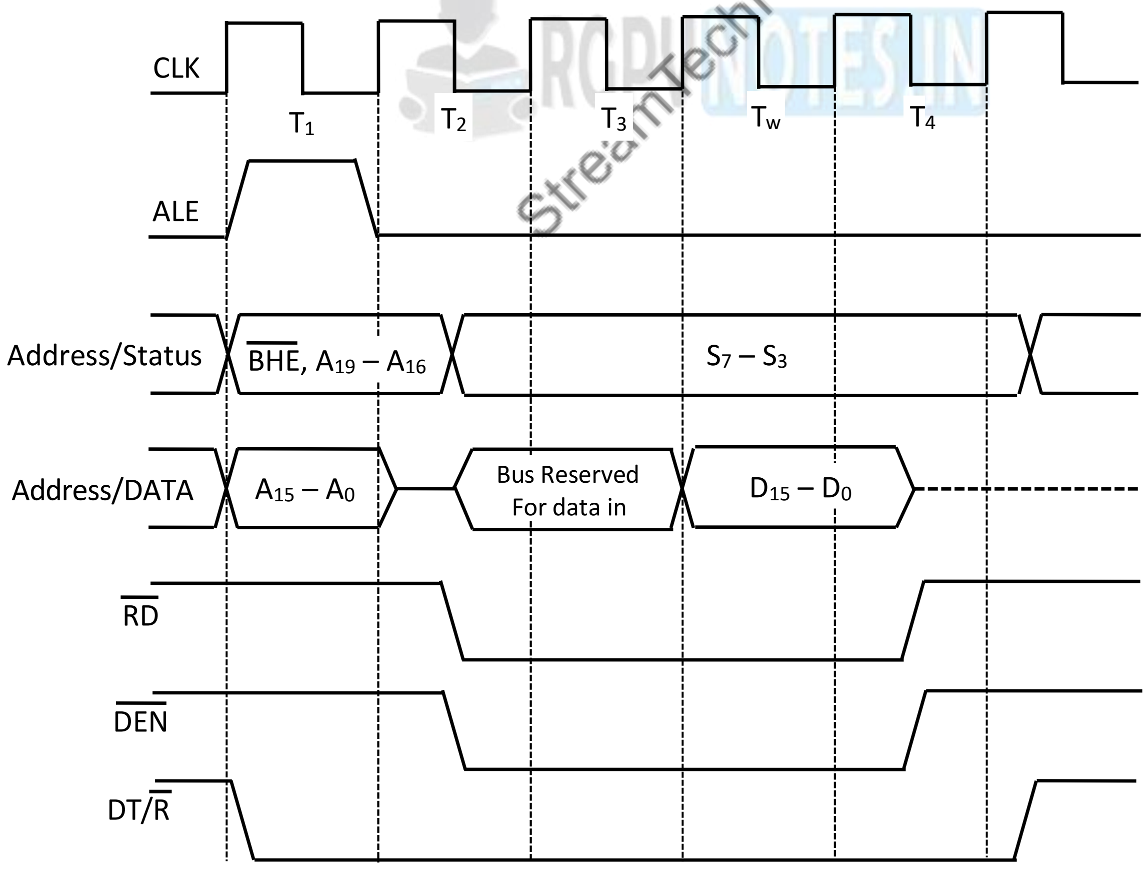


Fig.1.5 : Read cycle timing diagram for Minimum mode

- The read cycle begins with T_1 state by assertion of ALE signal. The address is latched during this state.
- At T_2 the address is removed from the local bus and the lines are reserved fro data. the RD signal is activated to read during this state.
- The data is made available by the enable device on the line and a valid data is available once the RD signal goes low and
- The process after finding the data on the lines make the RD signal high and tristate the data lines. Thus the process needs T_1 to T_4 state to complete one machine cycle.

Maximum Mode:

In the maximum mode, the 8086 is operated by strapping the MN/MX* pin to ground. In this mode, the processor derives the status signals S2*, S1* and S0*. Another chip called bus controller derives the control signals using this status information. In the maximum mode, there may be more than one microprocessor in the system configuration. The other components in the system are the same as in the minimum mode system. The general system organization is as shown in the Fig1.6 The basic functions of the bus controller chip IC8288, is to derive control signals like RD* and WR* (for memory and I/O devices), DEN*, DT/R*, ALE, etc. using the information made available by the processor on the status lines. The bus controller chip has input lines S2*, S1* and S0* and CLK. These inputs to 8288 are driven by the CPU. It derives the outputs ALE, DEN*, DT/R*, MWTC*, AMWC*, IORC*, IOWC* and AIOWC*. The AEN*, IOB and CEN pins are specially useful for multiprocessor systems. AEN* and IOB are generally grounded. CEN pin is usually tied to +5V.

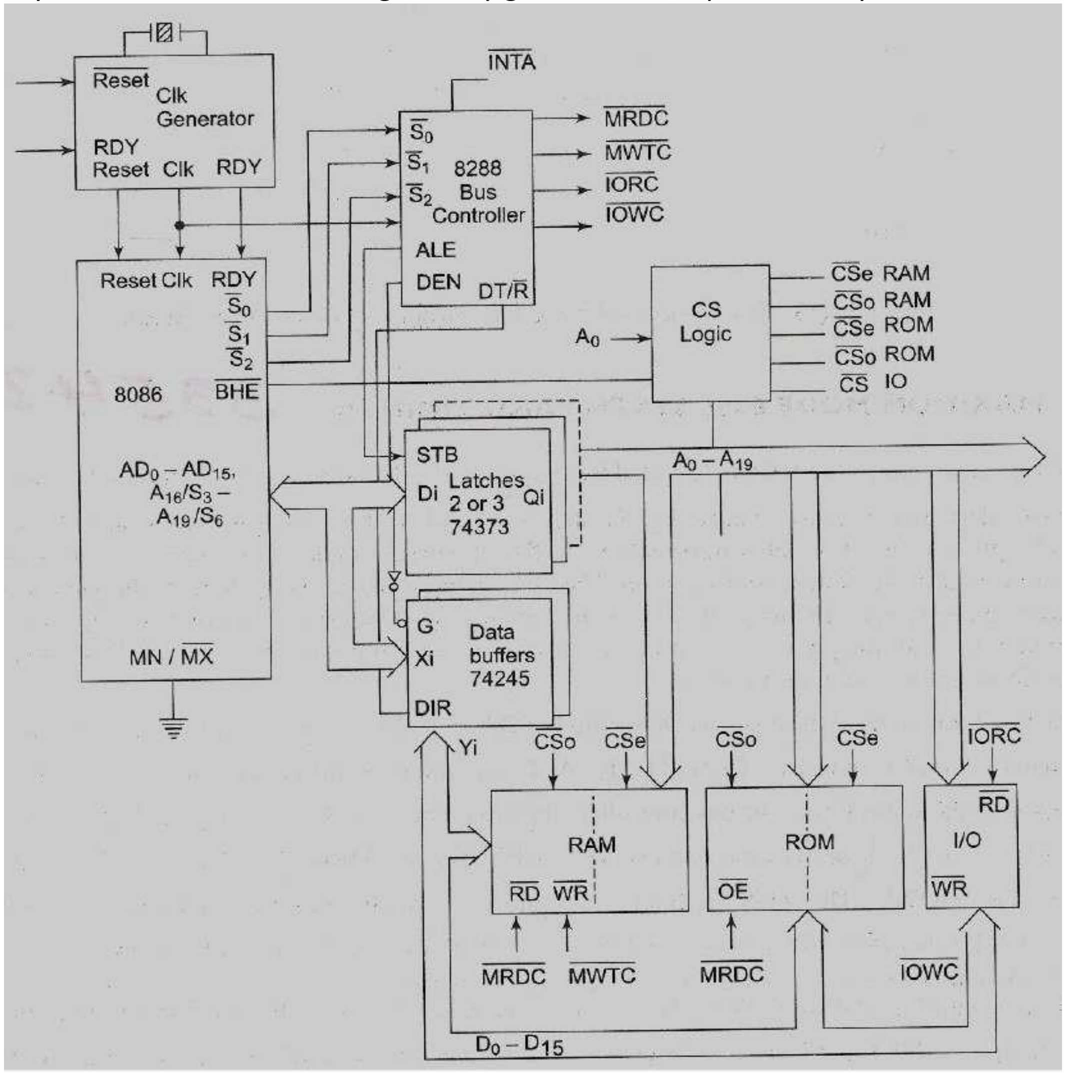


Fig.1.6: Maximum Mode Configuration

Unit III : Introduction to the various interfacings chips like 8155, 8255, Interfacings key boards, LEDs , ADC, DAC and memory Interfacing

---

Peripheral Devices And Their Interfacing: The general purpose computer system consists of the CPU, memory, keyboard, display unit and some special purpose input / output (I/O) devices. The central processing unit (CPU) is referred as the microprocessor. This microprocessor is connected with I/O devices and are called as peripheral devices. The microprocessor is heart of the computing system while all the peripheral devices are built around the microprocessor. Since the processor can not work without memory, the memory (primary) is also considered as an integral part of a microprocessor system.

The peripheral devices are connected with CPU to enable it to communicate with the user, so that the microprocessor works efficiently and effectively. The special purpose peripheral devices such as programmable interrupt controller (PIC), direct memory access (DMA), etc. simplifies both the hardware and software considerably. The peripheral devices need initialisation control signal to communicate with CPU & user. The memory does not need any special kind of initialisation signal. The peripheral device may be treated as memory location by the CPU. The configuration methods of peripheral devices with CPU are discussed in details.

PIO 8255 (Programmable Input Output Port): The IC 8255 is programmable peripheral input output port through which the peripheral devices can be connected to communicate with 8-bit or 16-bit Intel microprocessor. It has 24 input/output lines. These lines are individually programmed in two groups of 12-lines each named as group-A and group-B. Each of these groups consists of subgroups of 8-bit port and a 4-bit port. The group-A consist of 8-bit port-A (PA_7-PA_0) and 4-bit upper bits of port-C (PC_7-PC_4). The group-B consist of 8-bit port-B (PB_7-PB_0) and 4-bit lower bits of port-C (PC_3-PC_0). All the three ports can be used as individually either as input or as output ports. These ports can be programmed using its internal register called control word register (CWR). The internal architecture of IC 8255 is as shown in Fig.3.1 below.

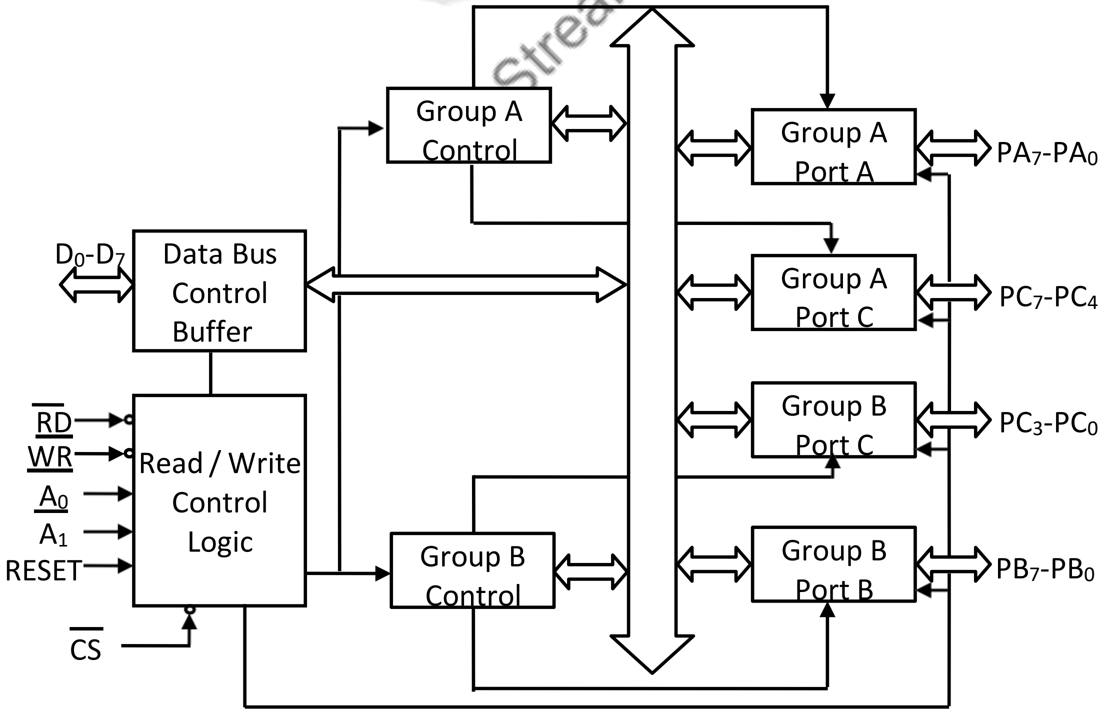


Fig.3.1 : PIO 8255 Internal Architecture

Pin Configuration of 8255 IC:

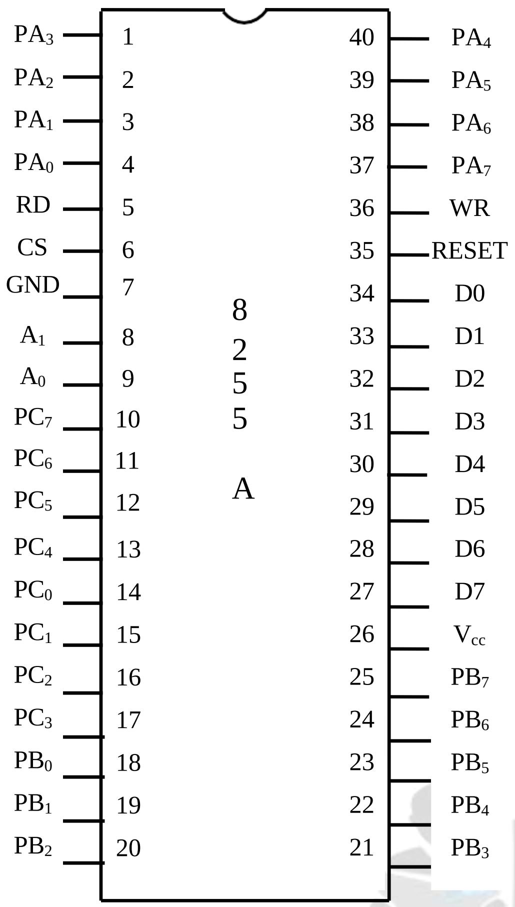


Fig.3.2 : Pin Configuration of 8255A

A1	A0	Port Selection
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control Word Register (CWR)

The pin configuration of 8255A is as shown in Fig.3.2 It is 40 pin IC and the pin details are as described below. The 3 ports can be configured as buffered input or latched output based on control word and mode of operation

(i) $PA_7 - PA_0$: Port A has 8 lines which can be configured as either output or input depending on the control word register value.

(ii) $PB_7 - PB_0$: Port B has 8 lines which can be configured as either output or input depending on the control word register value.

(iii) $PC_7 - PC_4$: Port C has 8 lines which can be configured as either output or input depending on the control word register value. Port C can also be used as handshake signal lines in mode 1 and mode 2. Upper nibble of Port C are used as handshake signal with Port A .

(iv) $PC_3 – PC_0$: In mode 1 & 2, the lower nibble of Port C lines are used as handshake lines with Port B.

(v) RD : It is active low input signal driven by the microprocessor to indicate the read operation to 8255.

(vi) WR : It is active low input signal driven by the microprocessor to indicate the write operation.

(vii) CS : It is a chip select line which enables the 8255 to respond to read & write operation.

(viii) $A_1 – A_0$: These two lines are address lines to select the appropriate port or control word register.

(ix) D_7-D_0: The 8-bit data bus which carry data or control word to / from the microprocessor.

(x) RESET : It is a logic high signal which clears the CWR of 8255.

Modes of Operation of 8255: The PIO 8255 operates basically in two modes, i.e., I/O mode and Bit Set Reset (BSR) mode. In the BSR mode only port C can be used to set or reset its individual bits. The input/output ports can be programmed in I/O mode of operation of 8255. Further there are 3 modes of I/O mode of operations namely, mode 0, mode 1 and mode2.

BSR Mode: In this mode, any of the 8-bits of port C can be set or reset depending on B_0 of the control word register(CWR). The bit to be set or reset is selected by bit select flags B_3, B_2, and B_1 of the CWR as given in Table. The control word format is shown in Fig.3.3.

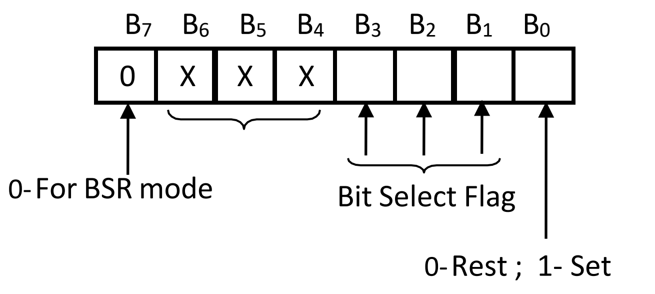


Fig.3.3 : BSR mode control word register format

Table : Bit Select Flag configuration

B3	B2	B1	Selected Bits of Port C
0	0	0	PC_0
0	0	1	PC_1
0	1	0	PC_2
0	1	1	PC_3
1	0	0	PC_4
1	0	1	PC_5
1	1	0	PC_6
1	1	1	PC_7

<u>I/O Modes :</u> There are three modes of I/O mode of operation. They are mode 0, mode 1, mode 2.

<u>Mode 0 (Basic I/O Mode) :</u> This mode is also called as Input / Output mode. This mode provides input and output capability using each of the three ports. The features of this mode are as follows

- Two 8-bit ports (Port A & B) and two 4-bit ports of Port C are available. The two 4-bit ports can be combinedly used as a third 8-bit port.
- Any port can be used as input or as latched output.
- A maximum of four ports are available so that overall 16 I/O configurations are possible.

The modes of each of the port can be selected by programming a control word register. The B7 bit of the CWR validates the I/O modes of operation and then reset of the bits shown the different modes of operation as shown in the Fig.3.4 .

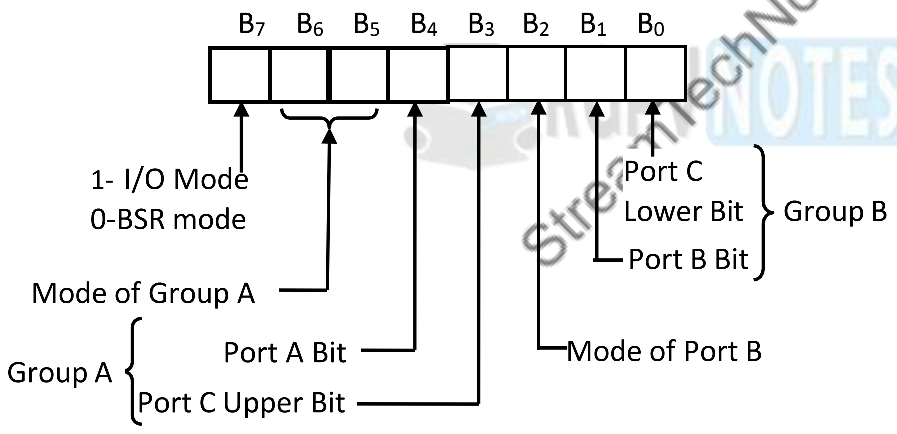


Fig.3.4 : I/O mode control word register format

(i) Group A Modes

B_6	B_5	**Mode**
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	X

(ii) Port B mode is dependent upon B_2 bit. A port is an output port if the bit is '0' else it is input port

<u>Mode 1 (Strobed I/O Mode) :</u> This mode is known as strobed input/output mode. In this mode the handshanking signals control the input / output action of the specified port. The features of this mode are as follows

- Two groups A & B are available for data transfer with handshanking
- Group A includes 8-bit Port A and as I/O port along with 4-bit of Port C as control/data port. Group B includes 8-bit Port B and as I/O port along with 4-bit of Port C as control/data port.
- PC_0 – PC_2 are used to generate control signals for port B and PC_3-PC_5 are used to generate control signals for port A. PC_6 & PC_7 may be used as independent data lines.

<u>Input Control Signal Definitions (Mode 1):</u>

- $\overline{STB}$ (Strobe Input): When this signal is low, the 8-bit data available in input is loaded into input latches. The pin PC_4 in group A and PC_2 in group B generates this handshake signal.

- IBF (Input buffer full): Once the data is loaded into the latches this signal rises to logic '1'. PC_5 in group A and PC_1 in group B this acts output acknowledge signal.
- INTR (Interrupt request): This is active high output signal which can be used to interrupt the CPU whenever an input device requests the service. PC_3 for group A and PC_0 for group B is used.

Output Control Signal Definitions (Mode 1):

- $\overline{OBF}$ (Output buffer full): When the data is written by the CPU to the specified output port, then this signal falls to logic low. PC7 for group A and PC1 for group B are used.
- $\overline{ACK}$ (Acknowledge input): This signal acts as an acknowledgement to be given by an output device. When the data received by the output device which is send by the CPU, then this signal falls to logic low.
- INTR (Interrupt request): This is active high output signal which can be used to interrupt the CPU whenever an input device requests the service. PC_3 for group A and PC_0 for group B is used.

Mode 2 (Strobed bidirectional I/O) : This is strobed bidirectional I/O mode.
This mode is used primarily in applications such as data transfer between the two computers. Port A can be configured as the bidirectional port and Port B either in mode 0 or mode 1.
Features

- The single 8-bit port in group A is available with both input and output latched.
- The 8-bit port is bidirectional and additionally a 5-bit control port is available.
- Three I/O lines are available at Port C (PC_2-PC_0).
- The 5-bit control port C (PC3-PC7) is used for handshake signals.

Problem Statement:
Interface an 8255 with 8086 to work as an I/O port. Initialize port A as output port, port B as input port and port C as output port. Port A address should be 0740H. Write a program to sense switch positions SW0-SW7 connected at port B. The sensed pattern is to be displayed on port A, to which 8 LEDs are connected, while the port C lower displays number of on switches out of the total eight switches.

Solution:
The control word is decided upon as follows:

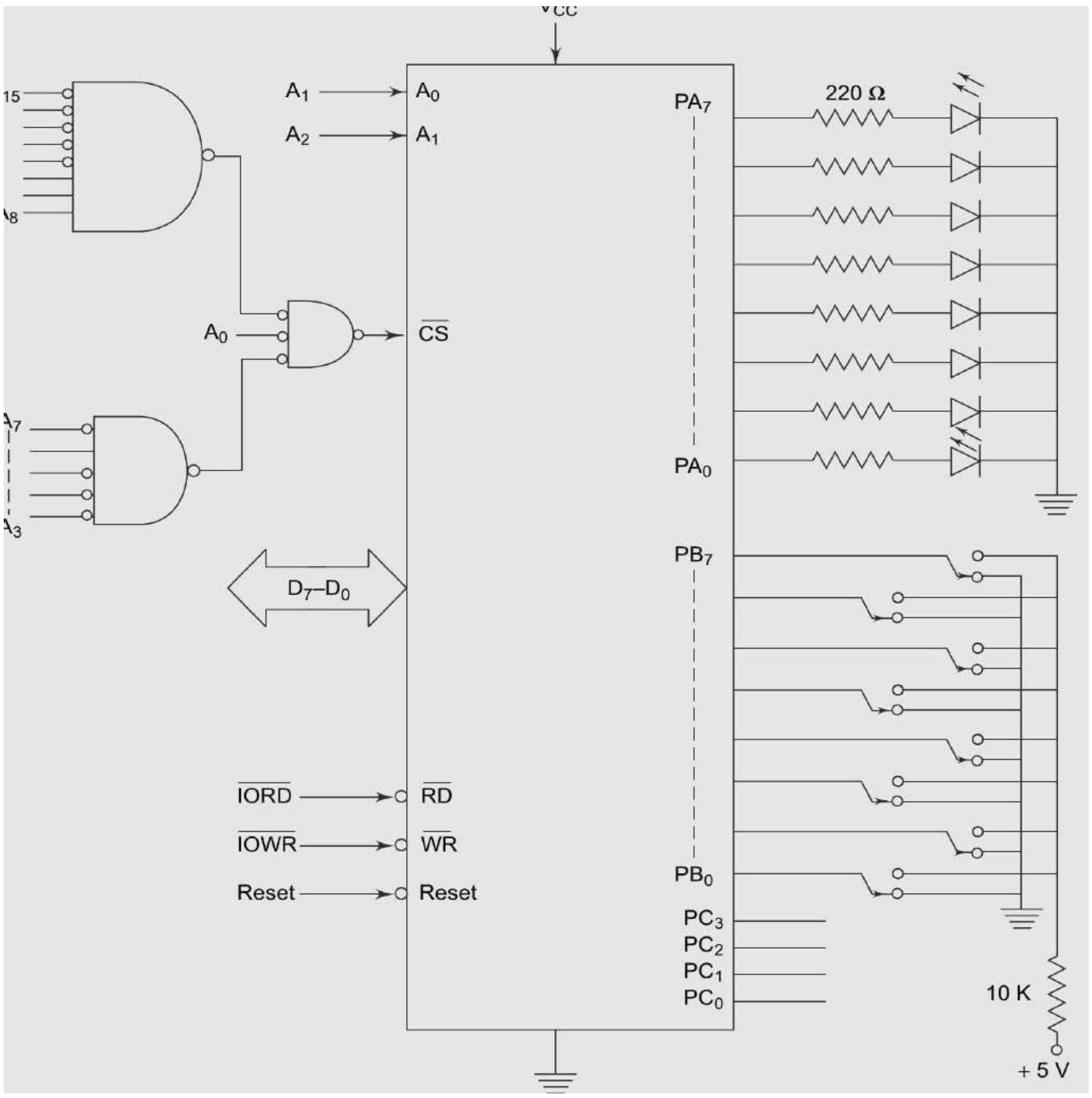
B7	B6	B5	B4	B3	B2	B1	B0
1	**0**	**0**	**0**	**0**	**0**	**1**	**0**
I/O Mode	Port-A in Mode 0		Port-A, O/P	Port-C, O/P	Port-B Mode 0	Port-B, I/P	Port-C, O/P

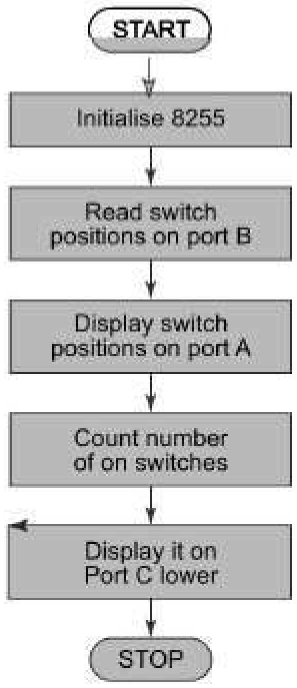
Control word = 82 H

Address line bit A_0 of the microprocessor is used to select the lower order data bus. Hence any changes in this bit will not affect the selection of the ports. Therefore A_2 and A_1 of the microprocessor address bits are used to select the ports and connected to A_1 and A_0 of 8255 IC. The address mapping table is as shown in below.

The 8255 port addresses are tabulated as shown below.

8255 Ports	**I/O Address Lines**																**Hex. Port Addresses**
	A15	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	
Port-A	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	**0740 H**
Port-B	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	**0742 H**
Port-C	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0	0	**0744 H**
CWR	0	0	0	0	0	1	1	1	0	1	0	0	0	1	1	0	**0746 H**




START
Initialise 8255
Read switch positions on port B
Display switch positions on port A
Count number of on switches
Display it on Port C lower
STOP

	MOV DX, 0786H	Initialize CWR with control word 82H
	MOV AL, 82H	
	OUT DX, AL	
	SUB DX, 04H	Get address of port B in DX
	IN AL, DX	Read port B for switch positions in to AL
	SUB DX, 02H	Get port A address in DX.
	OUT DX, AL	Display switch positions on port A
	MOV BL, 00H	Initialize BL for switch count
	MOV CH, 08H	Initialize CH for total switch number.
above:	ROL AL	Rotate AL through carry to check whether the switches are ON or OFF i.e., either 1 or 0.
	JNC below	
	INC BL	
Below:	DEC CH	Check for next switch. If all switch are checked, the number of ON switches are in BL.
	JNZ above	
	MOV AL, BL	Display it on port C,
	ADD DX, 04H	
	OUT DX, AL	
	HLT	Stop

UNIT II: Intel 8086 microprocessor programming: 8086 Instruction Set, Addressing modes, Assembly Language Programming with Intel 8086 microprocessor

---

<u>**Addressing Modes:**</u> The addressing modes are specifying the method of data access. It is indicating the flow of instruction execution and also the size of machine code. Depending on the type device communication, different addressing modes can be used.

(i) **Immediate Addressing Mode:** In this addressing mode the source operand is the data which can be stored into the destination operand.
e.g., MOV AX , 0140H ; after execution of this instruction AX=0140H

(ii) **Direct Addressing Mode:** One of the operand in the instruction is the address of the memory location from/to which the can read/write into the specified register.
e.g., MOV BH, [0100H] ; after execution of this instruction the content of the memory location pointed by memory location whose offset address is 0100H is copied into BH register.

(iii) **Register Addressing Mode:** Both the operand in this mode of address are the register of equal size.
e.g. MOV AL,CL ; after execution of this instruction the content of CL is copied into AL

(iv) **Register Indirect Addressing Mode:** The source or the destination operand register is holding the address of the memory location from/to which the data can be read/written into destination or source operand. The offset address is in either BX, SI or DI registers.
e.g., MOV [BX],DL ; after execution of this instruction the content of the memory location pointer by the offset address which is stored in BX register is copied into DL register.

(v) **Indexed Addressing Mode:** In this addressing mode the offset address is in the index register SI or DI. It one of the special case of register indirect addressing mode.
e.g., MOV BX , [SI] ; MOV [DI] , AX

(vi) **Register Relative Addressing Mode:** In this mode the required data location address is formed by adding an 8-bit or 16-bit displacement with the content of the register specified in the instruction. BX,BP,SI and DI are used.
e.g., MOV AX, 30H[BX]

(vii) **Based Indexed Addressing Mode:** The effective address of the data is formed, by adding content of base register to the content of an index register. The default segment register is DS.
e.g., MOV AX, [BX][SI]

(viii) **Relative Based Indexed Addressing Mode:** The effective address of the data is formed, by adding an 8-bit or 16-bit displacement with the sum of contents of base register to the content of an index register in a default segment register.
e.g., MOV AX, 30H [BX][SI]

<u>**Addressing Modes for Control Transfer Instruction:**</u>

(ix) **Intrasegment Direct Mode:** The address to which the control is to be transferred lies in the same segment in which the control transfer instruction lies. The value of the address is the immediate data in the

instruction. If it is 8-bit displacement, then called as short jump and if it 16-bit then called as long jump.
e.g., JMP Label ; were label is the 8-bit or 16-bit displacement

(x) **Intrasegment Indirect Mode:** The address to which the control is to be transferred lies in the same segment in which the control transfer instruction lies. The value of the address is passed indirectly through base registers.
e.g., JMP [BX] ; were BX is holding the displacement

(xi) **Intersegment Direct Mode:** The address to which the control is to be transferred is in a different segment. It is branching from one code segment to another code segment. The destination address are specified in the instruction directly.
e.g., JMP 5000H : 3000H ; were effective address 3000H in segment address 5000H.

(xii)**Intersegment Direct Mode:** The address to which the control is to be transferred is in a different segment. It is branching from one code segment to another code segment and its address is passed indirectly.
e.g., JMP [2000H] ; were four memory block contain the address as IP(LSB), IP(MSB), CS(LSB) and CS(MSB) sequentially.

Instruction Set:

The 8086 instruction sets are classified into 8 main categories.
(i) Data Transfer Instructions
(ii) + Arithmetic & Logical Instructions
(iii) Branch Instructions
(iv) Loop Instructions
(v) Machine Control Instructions
(vi) Flag Manipulation Instructions
(vii) Shift & Rotate Instructions
(viii) String Instructions

(i) Data Transfer Instructions

These instructions are used to transfer data from some source to a destination. Some of the instruction is used to copy the data. Following are the instruction under this class of instruction set.
(MOV, PUSH, POP, XCHG, XLAT, IN, OUT, LEA, LDS, LES, LAHF, SAHF, PUSHF, POPF).
The most commonly used instruction set is discussed below.

MOV: Copies a word or a byte of data from source to a destination. Direct loading of the segment registers with immediate data is not permitted.
e.g., MOV AX,BX ; MOV AH,08H ; MOV AH,[SI] ; MOV [0300H],BX ; MOV DS,CX ;

PUSH: It pushes the content of the specified register/memory location on to the stack. The stack pointer is decremented by 2, after each execution of this instruction.
e.g., PUSH AX ; PUSH DS ; PUSH [0400H], PUSHF

POP: It loads the specified register / memory location with the contents of the memory location pointed by current stack segment and stack pointer. The stack pointer is incremented by 2.
e.g., POP AX; POP DS, POP [0400H], POPF

XCHG: It exchanges the content of specified source and destination operands.
e.g., XCHG [0400H], AX; XCHG BX,AX

XLAT: Translate byte using look-up table.

Input & Output port transfer instructions.
IN: Copy a byte or word from specified port to an accumulator (AL or AX)
e.g., IN AX, 06H; IN AX,DX

OUT: Copy a byte or word from accumulator (AL or AX) to a specified port
e.g., OUT 06H, AL; OUT DX, AX

LAHF: Load AH with the low byte of the flag register
SAHF: Store content of AH register to low byte of flag register.

(ii) Arithmetic & Logical Instructions

Th e instructions which are used to perform arithmetic and logical operations are grouped into this category. Following are the instruction under this class of instruction set.
(ADD , ADC, SUB, SBB, INC, DEC, NEG, CMP, MUL, IMUL, DIV, AAA, AAS, AAM, AAD, DAA, DAS, AND, OR, NOT, XOR, TEST,)
The most commonly used instruction set is discussed below.

ADD (Addition) : It is used add two 8-bit or 16-bit numbers. The source and destination operands may be any register or memory location. Both operands should not be the memory location. The segment registers cannot be used for addition operation. All the condition flag bits are affected.
e.g., ADD AX, 0140H ; ADD BX, CX ; ADD [2040H],AX ; ADD [0300H], 0100H

ADC (Add with Carry) : It is used add two 8-bit or 16-bit numbers along with carry bit. The rest of the rules are as applicable to ADD instruction. All the condition flag bits are affected.
e.g., ADC AX, 0140H ; ADC BX, CX ; ADC [2040H],AX ; ADC [0300H], 0100H

SUB (Subtraction) : It is used subtract the 8-bit/16-bit source operand from destination operand. The source and destination operands may be any register or memory location. Both operands should not be the memory location. The segment registers cannot be used for subtraction operation. All the condition flag bits are affected.
e.g., SUB AX, 0140H ; SUB BX, CX ; SUB [2040H],AX ; SUB [0300H], 0100H

SBB (Subtract with Borrow): It is used subtract the 8-bit/16-bit source operand and the borrow bit (CF) from destination operand. Subtraction with borrow is subtraction of 1 from result obtained from SUB. Both operands should not be the memory location. The segment registers cannot be used for subtraction operation. All the condition flag bits are affected.
e.g., SBB AX, 0140H ; SBB BX, CX ; SBB [2040H],AX ; SBB [0300H], 0100H

INC (Increment): The content of the specified operand is incremented by 1. The conditional flag bits are affected except CF.
e.g., INC AX ; INC [BX] ; INC [5000H]

DEC (Decrement): This instruction decrements the content of the specified operand by 1. The conditional flag bits are affected except CF.
e.g., DEC AX ; DEC [BX] ; DEC [5000H]

CMP (Compare) : It compares the two numbers. The source operand is compared with the destination operand . The source operand is subtracted from the destination operand but the result is not stored. The flag bits are affected depending on the result.
e.g., CMP AX, 0200H ; CMP [5000H], 0200H

MUL (Unsigned Multiplication): It multiplies an unsigned byte with the content of AL and the result is stored in AX. The 16-bit number is multiplied with the content of AX and stored in DX and AX. Immediate operand is not allowed in this instruction.
e.g., MUL BH ; (AX) ← (AL) x (BH)
MUL CX ; (DX) (AX) ← (AX) x (CX)

DIV (Unsigned Division): It performs the unsigned division operation. It divides an unsigned 32-bit number by 16-bit or 8-bit operand. The dividend must be in AX for 16-bit operation and divisor may be specified from any of the addressing mode except immediate addressing mode. The result is in AL(quotient) and the remainder will be in AH. For 32-bit operation the higher word should be in DX and lower in AX. The divisor may be specified from any of the addressing mode except immediate mode. The quotient will be in AX and remainder will be in DX.
e.g., DIV BH ; (AL) ← (AX) / (BH) , AL – quotient , AH -- remainder
DIV CX ; (AX) ← (DX AX) / (CX) , AX – quotient , DX -- remainder

__Logical Instructions__

AND : Logical AND : It is performs the AND operation on every bit of the destination and source operand. The source may any register or immediate data and destination may register or memory location. The result is stored in the destination operand.
e.g., AND AX, 0008H ; AND AX, BX

OR : Logical OR : It is performs the OR operation on every bit of the destination and source operand. The source may any register or immediate data and destination may register or memory location. The result is stored in the destination operand.
e.g., OR AX, 0008H ; OR AX, BX

NOT : Logical Invert : This instruction complements the contents of an operand register or a memory location, bit by bit.
e.g., NOT AX ; NOT [5000H]

XOR : Logical Exclusive OR: It is performs the XOR operation on every bit of the destination and source operand. The source may any register or immediate data and destination may register or memory location. The result is stored in the destination operand.
e.g., XOR AX, 0008H ; XOR AX, BX

(iii) __Branch Instructions:__

These set of instructions are used to transfer the flow of execution of program to a new address specified in the instruction. The new values of the CS & IP registers are updated with address to which the control has to be transferred. The branch instructions are classified into two types
(a) Unconditional Branch Instructions (b) Conditional Branch Instructions

JMP: Unconditional Jump
This instruction unconditionally transfers the control of execution to the specified address using an 8-bit or 16-bit displacement. No Flags are affected by this instruction.

Conditional Branch Instructions

JZ/JE Label	;Transfer execution control to address 'Label', if ZF=1.
JNZ/JNE Label	; Transfer execution control to address 'Label', if ZF=0
JS Label	;Transfer execution control to address 'Label', if SF=1.
JNS Label	;Transfer execution control to address 'Label', if SF=0.
JO Label	;Transfer execution control to address 'Label', if OF=1.
JNO Label	;Transfer execution control to address 'Label', if OF=0.
JNP Label	;Transfer execution control to address 'Label', if PF=0.
JP Label	;Transfer execution control to address 'Label', if PF=1.
JB Label	;Transfer execution control to address 'Label', if CF=1.
JNB Label	;Transfer execution control to address 'Label', if CF=0.

(iv) Loop Instructions:
These set of instructions are used to execute the part of the program to be repeated from the specified address. There are two types of loop instructions used (i) unconditional & (ii) Conditional

LOOP Label : LOOP Unconditionally
This instruction repeats some of the instructions from the specified address. At each iteration, content of CX register is decremented automatically, if the value of CX is not zero then it repeats.

```
Example:    MOV CX, 0004H
            MOV BX, 7526H
    Label 1 MOV AX, CODE
            OR BX, AX
            LOOP Label 1
```

Conditional Branch Instructions
LOOPZ / LOOPE Label ; It repeats the set of instruction from the specified label if ZF = 1 and CX = 0
LOOPNZ / LOOPNE Label ; It repeats the set of instruction from the specified label if ZF = 0

(v) Machine Control instructions
These instructions control the bus usage and execution of instructions.
WAIT – Wait for Test input pin to go low.
HLT – Halt the process.
NOP – No operation.
ESC – Escape to external device like NDP
LOCK – Bus lock instruction prefix.

(vi) Flag Manipulation Instructions

The flag bits of 8086 can be modified to control the functioning of processor. Following are the instructions used to modify.

CLC – Clear Carry Flag.
CMC – Complement Carry Flag.
STC – Set Carry Flag.
CLD – Clear Direction Flag.
STD – Set Direction Flag.
CLI – Clear Interrupt Flag.
STI – Set Interrupt Flag.

(vii) Shift and Rotate Instructions

SAL/SHL : SAL / SHL destination, count.
SAL (Shift Arithmetic Left) and SHL(Shift Logical Left) are two mnemonics for the same instruction. It shifts each bit in the specified destination to the left and '0' is stored at the vacated bit (LSB position). The MSB bit is shifted into the carry flag. The no. of shift is specified by the count value.
e.g., SHL BX ; SAL AX, CL ; default value of count is '1' and if CL register is mentioned then it contain the count value.

SHR : SHR destination, count
It shifts each bit in the specified destination to the right and '0' is stored at the vacated bit (MSB position). The LSB bit is shifted into the carry flag. The no. of shift is specified by the count value.
e.g., SHR BX ; SHR AX, CL ; default value of count is '1' and if CL register is mentioned then it contain the count value.

SAR : SAR destination, count
It shifts each bit in the specified destination to the right and the vacated bit is stored with old MSB bit itself (MSB position). The LSB bit is shifted into the carry flag. The no. of shift is specified by the count value.
e.g., SAR BX ; SAR AX, CL ; default value of count is '1' and if CL register is mentioned then it contain the count value.

ROL Instruction : ROL destination, count
This instruction rotates all bits in a specified byte or word to the left some number of bit positions. MSB is placed as a new LSB and a new CF.
e.g., ROL CX, 1 ; MOV CL, 03H ; ROL BL, CL

ROR Instruction : ROR destination, count
This inst ruction rotates all bits in a specified byte or word to the right some number of bit positions. LSB is placed as a new MSB and a new CF.
e.g., ROR CX, 1 ; MOV CL, 03H ; ROR BL, CL

RCL Instruction : RCL destination, count
This instruction rotates all bits in a specified byte or word some number of bit positions to the left along with the carry flag. MSB is placed as a new carry and previous carry is place as new LSB.
e.g., RCL CX, 1 ; MOV CL, 04H ; RCL AL, CL
RCR Instruction : RCR destination, count
This instruction rotates all bits in a specified byte or word some number of bit positions to the right along with the carry flag. LSB is placed as a new carry and previous carry is place as new MSB.

e.g., RCR CX, 1 ; MOV CL, 04H ; RCR AL, CL

ROR Instruction : ROR destination, count
This instruction rotates all bits in a specified byte or word to the right some number of bit positions. LSB is placed as a new MSB and a new CF.
e.g., ROR CX, 1 ; MOV CL, 03H ; ROR BL, CL

RCL Instruction : RCL destination, count
This instruction rotates all bits in a specified byte or word some number of bit positions to the left along with the carry flag. MSB is placed as a new carry and previous carry is place as new LSB.
e.g., RCL CX, 1 ; MOV CL, 04H ; RCL AL, CL

RCR Instruction : RCR destination, count
This instruction rotates all bits in a specified byate or word some number of bit positions to the right along with the carry flag. LSB is placed as a new carry and previous carry is place as new MSB.
e.g., RCR CX, 1 ; MOV CL, 04H ; RCR AL, CL

(viii) String Manipulation Instructions

String is a collection of characters stored in the consecutive memory locations. This data is represented in its ASCII equivalent of character. The 8086 has a set of instructions which can be used to manipulate the string. In these instructions two parameters are required, starting and end address of the string and length of the string. The register CX hold the length of the string and it is decremented after every byte of operation. The pointer register which holds the starting address of the string is incremented or decremented depending on the direction flag (DF). If the byte string operation is used, then the index registers are updated by one. If the word string operation is used, then the index registers are updated by two.

Assembly Language Programming :

There are 3 different ways of writing the program by using high level language, assembly language and machine language. The microprocessor or microcontroller is programmed using machine language programming. The main advantage of machine language programming is that the memory control is directly in the hands of the programmer. The disadvantages are tedious to write and communicate in machine language. The assembly language programming is simpler as compared to the machine language programming. The instruction mnemonics are directly written in the assembly language programs. These programs are more readable by other programmers. There are some tools used to convert these assembly language programs into machine codes.

(i) Assembler : It is a program used to convert an assembly language program into the machine code. It decides the address of each label and assigns the value to each of the variables, then forms the machine code. It also find the syntax error and not the logical errors. There are two assemblers used, Microsoft Macro Assembler (MASM) and Turbo Assembler (TSAM).
(ii) Linker : The DOS linking program LINK.EXE links the different object modules of a source program and function library to generate an integrated source program. The input to the linker is the .OBJ file that contains the object modules of the source program. It performs the three tasks

- Searches the program to find library routines used by program
- Determines the memory locations that code from each module will occupy and relocates its instructions by adjusting absolute references
- Resolves references among files

(iii) Loader : The loader is a part of the operating system and places codes into the memory after reading the '.exe' file. This step is necessary because the available memory addresses may not start from 0x0000, and binary codes have to be loaded at the different addresses during the run. The loader finds the appropriate start address.

1. Write an assemble language program (ALP) to add two 16-bit numbers

Solution:

```
DATA SEGMENT
        var1    DW      2136H
        var2    DW      4321H
        res     DW      ?
DATA ENDS
ASSUME          CS:CODE,        DS:DATA
CODE SEGMENT
START :         MOV     AX, DATA
                MOV     DS, AX
                MOV     AX, var1
                CLC
                MOV     BX, 0000H
                ADD     AX, var2
                JNC     below
                INC     BX
        below:  MOV   res, AX
                MOV     res+2,BX
                MOV   AH,   4CH
                INT 21H
        CODE ENDS
        END START
```

2. Write an assemble language program (ALP) to multiply two 8-bit numbers

Solution:

```
DATA SEGMENT
        var1    DB      36H
        var2    DB      43H
        res     DW      ?
DATA ENDS
ASSUME          CS:CODE,        DS:DATA
CODE SEGMENT
START :         MOV     AX, DATA
                MOV     DS, AX
                MOV     AL, var1
                MOV     BL, var2
                MUL     BL
                MOV     res, AX
                MOV     AH, 4CH
                INT     21H
        CODE ENDS
        END START
```

Unit IV : General purposes programmable peripheral devices (8253), 8254 programmable interval timer, 8259A programmable interrupt controller & 8257 DMA controller, USART, serial I/O & data Communication .

Programmable Interval Timer 8253/8254: The generation of arbitrary time delay precisely using software routine programs is difficult and becomes the software overhead. The Intel's programmable counter/timer device (8253) generates accurate time delays and the microprocessor becomes free from this task and minimises the software overhead. The internal architecture of 8253 is shown in Fig 4.1. It has the following features.

(i) It has 3 independent 16-bit presettable down BCD or hexadecimal counters, with a maximum counter rate of 2.6MHz.

(ii) All the 3 counter can be independently programmed to maintain them simultaneously and controlled by control word register.

(iii) The 8-bit data bus buffer is connected with microprocessor to transmit and receive the data, through the execution of IN or OUT instruction.

(iv) The read/write logic controls the direction of the operation and selection of the counter or control word register.

(v) It is possible to read easily on line without disturbing the clock input to the counter. This facility is called as "on the fly" reading of counter.

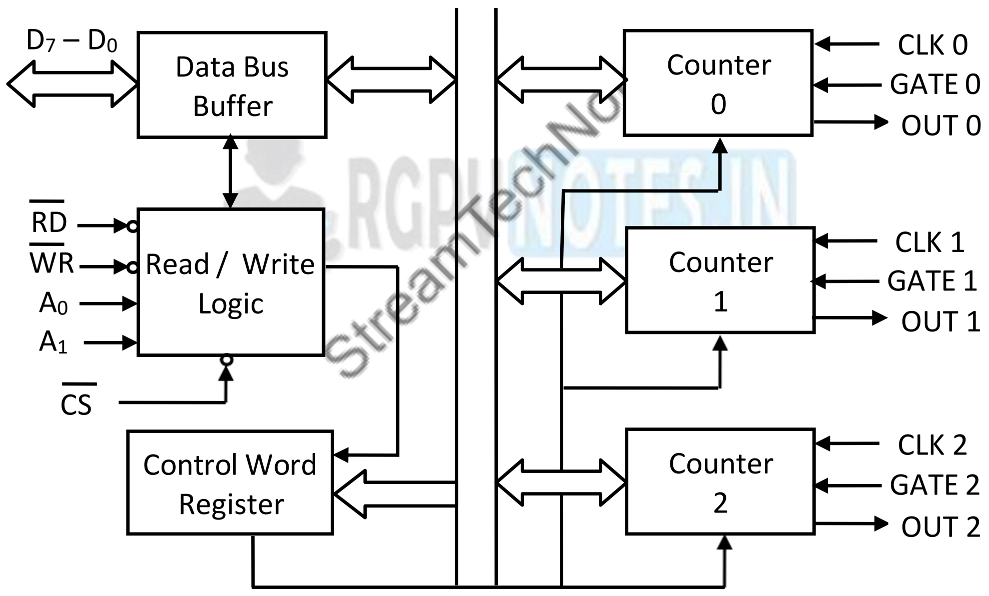


Fig.4.1 : Internal Block Diagram of 8253/8254 Programmable Interval Timer

The A_0 and A_1 pins are the address pins which are used to address internally the mode control word register and three counters. The 8-bit control word data is written by microprocessor into the control word register and control the specific counter. The CLK, GATE and OUT pins are available for each of the three timer channels. A control word must be written into respective control word register by the microprocessor to initialise each of the counters of 8253 to decide its operating mode. The 8253 can be operated in six different modes. All the three counters can be operated in any one mode or they may be in different modes of operation at a time.

A_1	A_0	Selection Operation
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word

Operating Modes of 8253:

Its three counters can operate in one of the six different modes of operation.

(i) Mode0 (Interrupt on terminal count)
(ii) Mode1 (Programmable monoshot)
(iii) Mode2 (Rate generator)
(iv) Mode3 (Square wave generator)
(v) Mode4 (Software triggered strobe)
(vi) Mode5 (Hardware triggered strobe)

The control word register bit format and its value will decide the mode of operation of 8253. The bit definition of control word is as shown in Fig.4.2.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SC_1	SC_0	RL_1	RL_0	M_2	M_1	M_0	BCD

SC_1	SC_0	Operation
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Illegal

RL_1	RL_0	Operation
0	0	Latch Counter for 'On The Fly' reading
0	1	Read/Load Least Significant Byte only
1	0	Read/Load MSB only
1	1	Read/Load LSB first then MSB

M_2	M_1	M_0	Selected Mode
0	0	0	Mode 0
0	0	1	Mode 1
0	1	0	Mode 2
0	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

D_0 – 0 – Hexadecimal Count
D_0 – 1 – BCD Count

Fig.4.2 : Control Word Register Format and Bit Definitions

Mode 0: This mode is called as interrupt on terminal count. In this mode the output is initially low after the mode is set. The count value is loaded into the counter and then starts decrementing the count value after the falling edge of the clock. Once it reaches to the count value i.e., count value zero, the output goes high and remains high until the new count or CWR is reloaded.

Mode 1: This mode is called as programmable one-shot mode. It can be used as a monostable multivibrator. The duration of the oscillations can be decided by the count loaded in the count register.

Mode 2: This mode is called as rate generator or divide by N counter. The value of N is loaded as the count value, after N clock pulses the output becomes low only for one clock cycle. The count N is reloaded again the output becomes high and remains so for N clock pulse.

Mode 3: This mode can be used as a square wave rate generator. When the count N loaded with even, then half of the count the output remains high and for the remaining half it remains low. If the count loaded is odd, the first pulse remains high and then half the value high and then low.

Mode 4: This mode is named as software triggered strobe. When the count is loaded into the count register, the output goes high. On terminal count, the output goes low for one clock cycle, and then it again goes high.

Mode 5: This mode is called as hardware triggered strobe. The strobe signal is generated in response to an externally generated signal. Once the counter is loaded, the output goes high, after the terminal count is reached the output goes low.

Interrupts & Interrupt Service Routine:

The word 'interrupt' means break the normal sequence of operation, perform the important other sequence of operation and then continue with the previous operation. Interrupts in 8086 microprocessor breaks the normal execution of sequence of instructions, serve the requesting device. The program which is used to serve the requesting device is called Interrupt Service Routine (ISR). Once the execution of ISR is completed, the control is transferred back again to the main program which was being executed at the time of interruption.

In any computing system, there are many devices connected to monitor some of the physical parameters. 8086 microprocessor has two pins to handle the interrupts, i.e., NMI and INTR. The NMI is a non-maskable interrupt input pin, which cannot be disabled by any means. The INTR interrupt is a maskable interrupt input pin, which be masked using the Interrupt Flag (IF). There are 256 types of interrupts handled by 8086 microprocessor. When more than one interrupt occurs at a time, then an external chip called programmable interrupt controller is required to handle them.

Interrupts of 8086: The 8086 microprocessor interrupts are classified into two types:

(i) Software Interrupts (ii) Hardware Interrupts

(i) Software Interrupts: there are 256 software interrupt instructions INT N, were N has a value from 0 to 255. (INT 0 to INT 255). These are further classified as

(a) INT 0 to INT 4 are reserved for dedicated operation.

(b) INT 5 to INT 31 are reserved for the particular operation of the system.

(c) INT 32 to INT 255 are used by the user.

(ii) Hardware Interrupts: The two pins NMI and INTR are used to connect the I/O devices to communicate with the microprocessor.

(a) NMI : It is a non-maskable vectored interrupt pin with positive edge triggered pin. This can not be disabled by any means. It can be used in emergency situations of system design. The ISR for this interrupt is stored in the interrupt vector table. This has highest priority.

(b) INTR : It is a maskable interrupt through which the I/O device is connected. Whenever an external device send the request,

Interrupt Cycle: The interrupts are classified into two types, one is external interrupt and the other is internal interrupt. An external device interrupts the processor from outside is called external interrupt. The internal interrupt is generated internally by the processor circuit, or by the execution of an interrupt instruction.

In case of external device interrupting the CPU at either NMI or INTR pin of 8086, while the CPU is executing the instructions. The Fig.4.3 shows the interrupt sequence of 8086 and Fig.4.4 shows the structure of interrupt vector table. Following are the steps during the interrupt cycle.

(i) The CPU first completes the execution of the current instruction. The IP is then incremented to point to the next instruction.

(ii) The CPU then acknowledges the requesting device on its INTA pin if it is non-maskable interrupt. otherwise it checks the interrupt flag (IF), if the IF is set, the request is acknowledged otherwise ignored.

(iii) After acknowledgement, the CPU computes the vector address based on the type of interrupt which is passed to the interrupt structure of the CPU.

(iv) The contents of PSW is pushed to the stack & then contents of IP & CS are pushed to the stack which are required to continue the execution after completion of ISR execution.

(v) The IF flag is cleared (reset) & also trap flag is cleared. The control is then transferred to the ISR for serving the interrupting device.

(vi) The new address of ISR is found from the interrupt vector table and execution of the ISR starts.
(vii) After completion of ISR, the main IP, CS & PSW are restored and the main program execution continues.

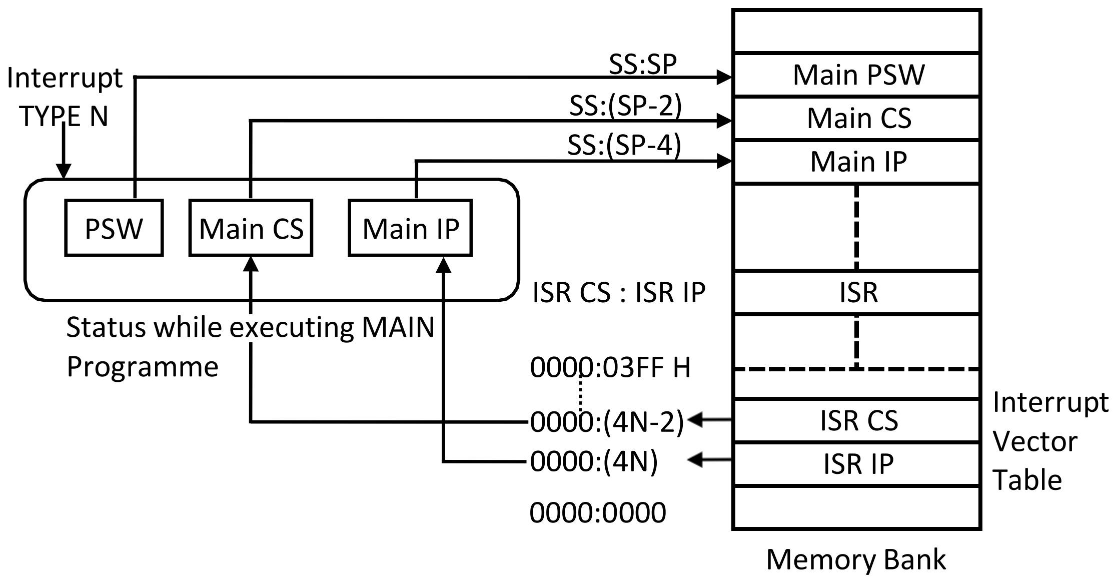
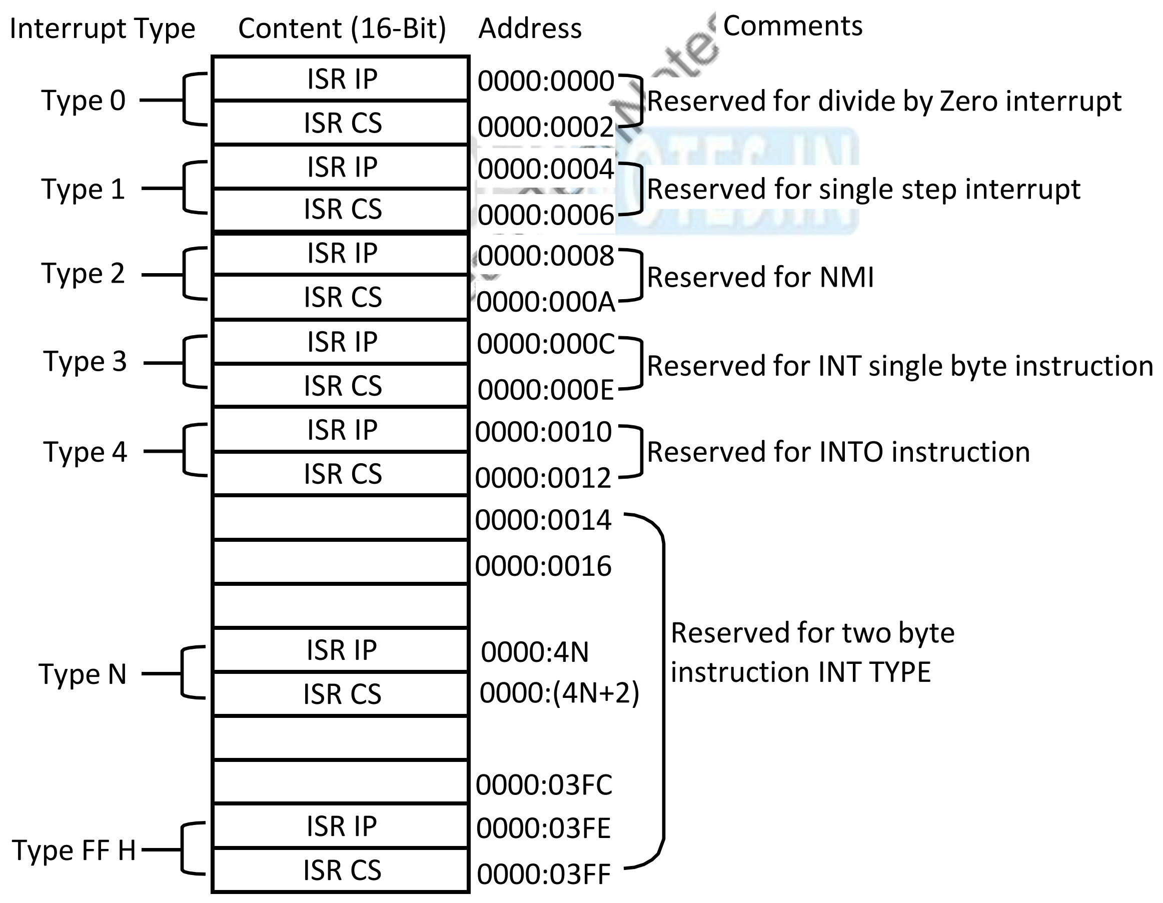


Fig.4.3 : Interrupt Response Sequence



ISR : Interrupt Service Routine

Fig.4.4: Structure of Interrupt Vector Table of 8086

Programmable Interrupt Controller 8259A: A programmable interrupt controller is able to handle a number of interrupts simultaneously. It takes care of the type of interrupt and its priorities. The 8259 was designed to operate only with 8-bit processors, 8259A is compatible with both 8-bit and 16-bit processors. The internal block diagram of 8259A is shown in Fig.4.5 below. The function of each of the block is described as follows:

(i) **Interrupt Request Register (IRR):** The interrupts at IR_0 to IR_7 input lines are stored in the IRR register in order to serve them on the priority basis. It can handle 8 interrupts at a time.

(ii) **In-Service Register (ISR):** This register is used to store the interrupts which are being served and keeps track of the requests being served.

(iii) **Priority Resolver:** The priority of the interrupt requests appearing at a time is determined by this unit. The IR_0 has the highest priority and IR_7 has the lowest priority in the fixed priority mode, however it can be altered by programming in rotating priority mode. The highest priority interrupt request is served first.

(iv) **Interrupt Mask Register (IMR):** This register stores the bits required to mask the interrupt inputs. IMR operates on IRR to resolve the priority.

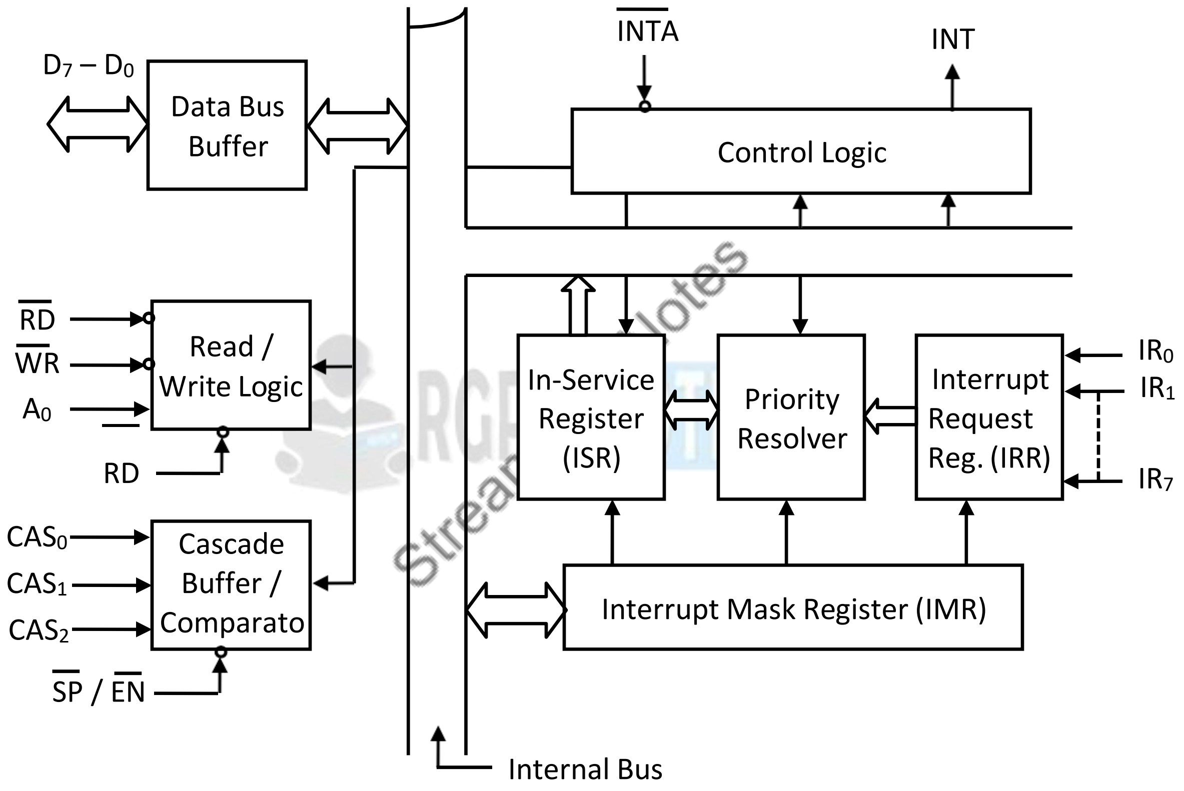


Fig.4.5: 8259 Interrupt Controller Block Diagram

(v) **Interrupt Control Logic:** This block communicates with microprocessor and manages the acknowledge signals to serve one of the eight interrupt requests. It accepts the interrupt acknowledge (INTA) signal and release the vector address on to the data bus to serve.

(vi) **Data Bus Buffer:** This is a tristate bidirectional buffer, which interface internal bus of 8259A to the processor system data bus. Control words, status and vector information is passed through data buffer during read or write operation.

(vii) **Cascade Buffer / Comparator:** This block stores the ID of all 8259A when more than one 8259A is used in the system. The I/O pins CAS0-CAS2 acts as output pins when 8259A is used as master, the same pins acts as input pins when the 8259A is used as slave.

(viii) **Read / Write Control Logic:** This circuit accepts & decodes commands from processor. This also allows the status of the 8259A to be transferred on to the data bus.

<u>Interrupt Sequence in an 8086 System:</u> The interrupt sequence in 8086 microprocessor when used with interrupt controller 8259A

(i) The IR lines which are requesting will set the corresponding IRR bits.

(ii) The priority resolver of 8259A resolves the priority and sends an INT signal to processor.

(iii) The processor acknowledges with INTA pulse.

(iv) After receiving an INTA signal, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive data bus during this period.

(v) The 8086 will initialise a second INTA pulse. During this period 8259A release an 8-bit pointer on to data bus from where it is read by the processor.

(vi) This completes the interrupt cycle. The ISR bit is reset at the end of second INTA pulse if automatic end of interrupt (AEOI) mode is programmed, otherwise ISR bit remains set until an EOI command is executed at the end of interrupt subroutine.

<u>DMA Controller 8257:</u> The internal architecture of 8257 DMA controller is as shown in the Fig. 4.6 below. This controller has four DMA channels, which can be used independently for DMA data transfer through these channels at a time. The DMA controller has 8-bit internal data buffer, a read/write logic unit, a control unit, a priority resolving unit along with a set of registers.

(i) Register Organisation of 8257: The DMA controller performs its operations over four independent DMA channels. These four channels of 8257 has pair of two 16-bit registers, DMA address register & terminal count register. Also, there are two common registers for all the channels, mode set register & status register. These ten registers are selected using A_0-A_3 address lines.

(a) DMA Address Register: Each of the channel has one DMA address register, which stores the starting address of memory location from where the DMA operation has to start. The device access this address to transfer the block of data.

(b) Terminal Count Register: It a 16-bit register used to ascertain that the data transfer through a DMA channel stops after the required number of DMA cycles. This register should be appropriately written before the actual DMA operation starts. The lower order 14-bits of the terminal count register are initialised with binary equivalent number. The bits higher order two bits indicate the type of DMA operation.

(c) Mode Set Register: This register is used to enable the DMA channels individually and to set the various modes of operation. The channel should not be enabled till the DMA address & terminal count register contains valid data.

(d) Status Register: The status register of 8257 indicate the status of each channel. The lower order 4-bits contain the terminal count status for the four channels. The bit set, indicates that the specific channel has reached the terminal count condition.

(ii) Data Bus Buffer : The 8-bit, tristate, bidirectional buffer interfaces the internal bus of 8257 with external system bus.

(iii) Read/Write Logic : In the slave mode, this block accepts the I/O read or I/O write signal, decodes the address A_0-A_3 and accordingly selects the register. In master mode, this block generates the IOR and IOW signals to control the data flow.

(iv) Control Unit : This unit generates the required control signals to control the sequence of operations. AEN, ADSTB, MEMR, MEMW, TC & MARK along with address lines A_4-A_7 in master mode are generated.

(v) Priority Resolver: The priority resolver resolves the priority of the four DMA channels depending on the mode of operation set during programming.

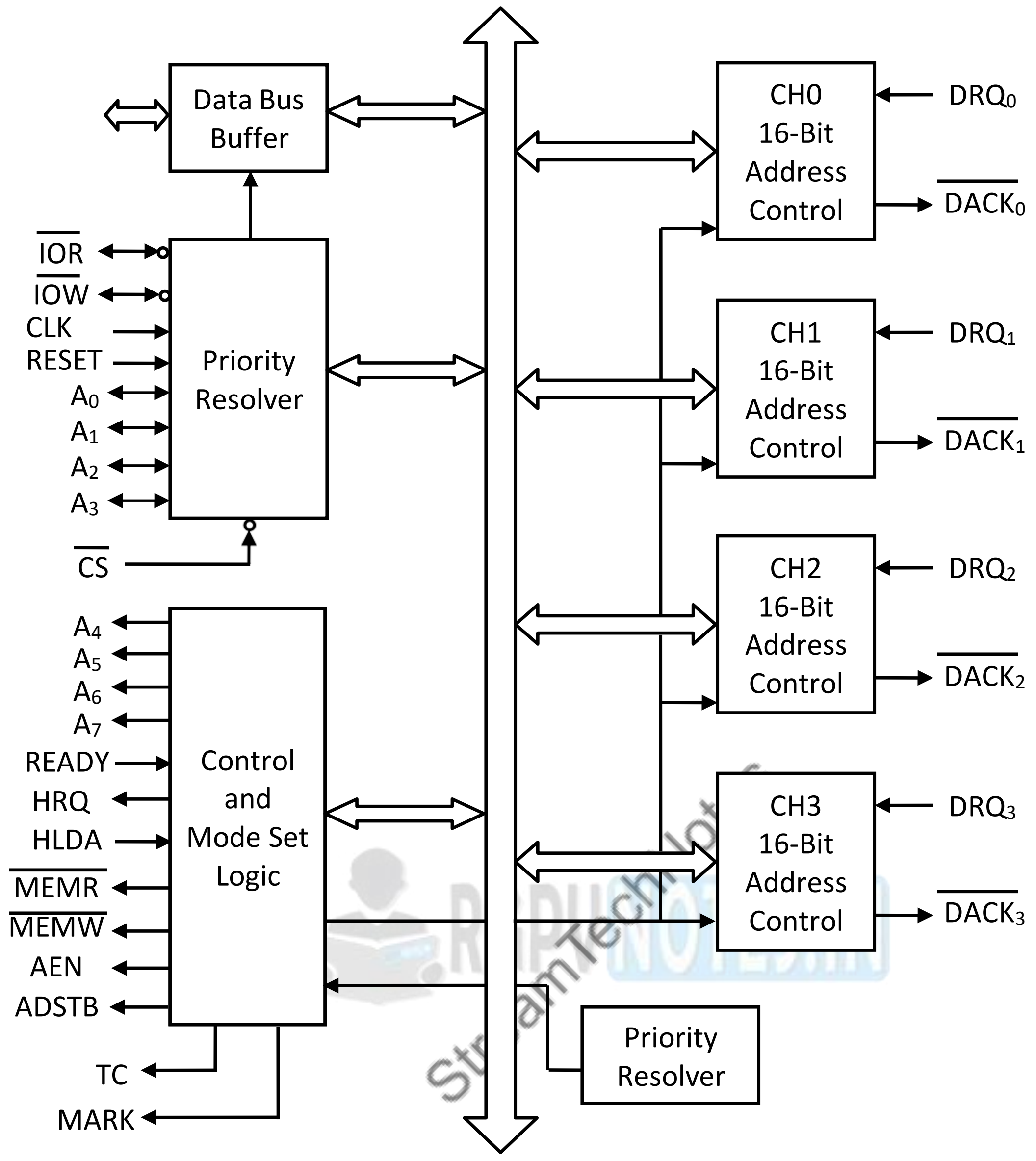


Fig. 4.6 : Internal Architecture of 8257

<u>Programmable Communication Interface 8251 USART</u>:

8251 is a Universal Synchronous and Asynchronous Receiver and Transmitter compatible with Intel's processors. This chip converts the parallel data into a serial stream of bits suitable for serial transmission. It is also able to receive a serial stream of bits and convert it into parallel data bytes to be read by a microprocessor.

Basic Modes of data transmission

a) Simplex

b) Duplex

c) Half Duplex

a) Simplex mode : Data is transmitted only in one direction over a single communication channel. For example, the processor may transmit data for a CRT display unit in this mode.

b) Duplex Mode : In duplex mode, data may be transferred between two transreceivers in both directions simultaneously.

c) Half Duplex mode : In this mode, data transmission may take place in either direction, but at a time data may be transmitted only in one direction. A computer may communicate with a terminal in this mode. It is not possible to transmit data from the computer to the terminal and terminal to computer simultaneously.

Architecture of 8251 USART: The data buffer interfaces the internal bus of the circuit with the system bus. The read / write control logic controls the operation of the peripheral depending upon the operations initiated by the CPU. C / D decides whether the address on internal data bus is control address / data address. The modem control unit handles the modem handshake signals to coordinate the communication between modem and USART. The transmit control unit transmits the data byte received by the data buffer from the CPU for serial communication. The transmission rate is controlled by the input frequency. Transmit control unit also derives two transmitter status signals namely TXRDY and TXEMPTY which may be used by the CPU for handshaking. The transmit buffer is a parallel to serial converter that receives a parallel byte for conversion into a serial signal for further transmission.

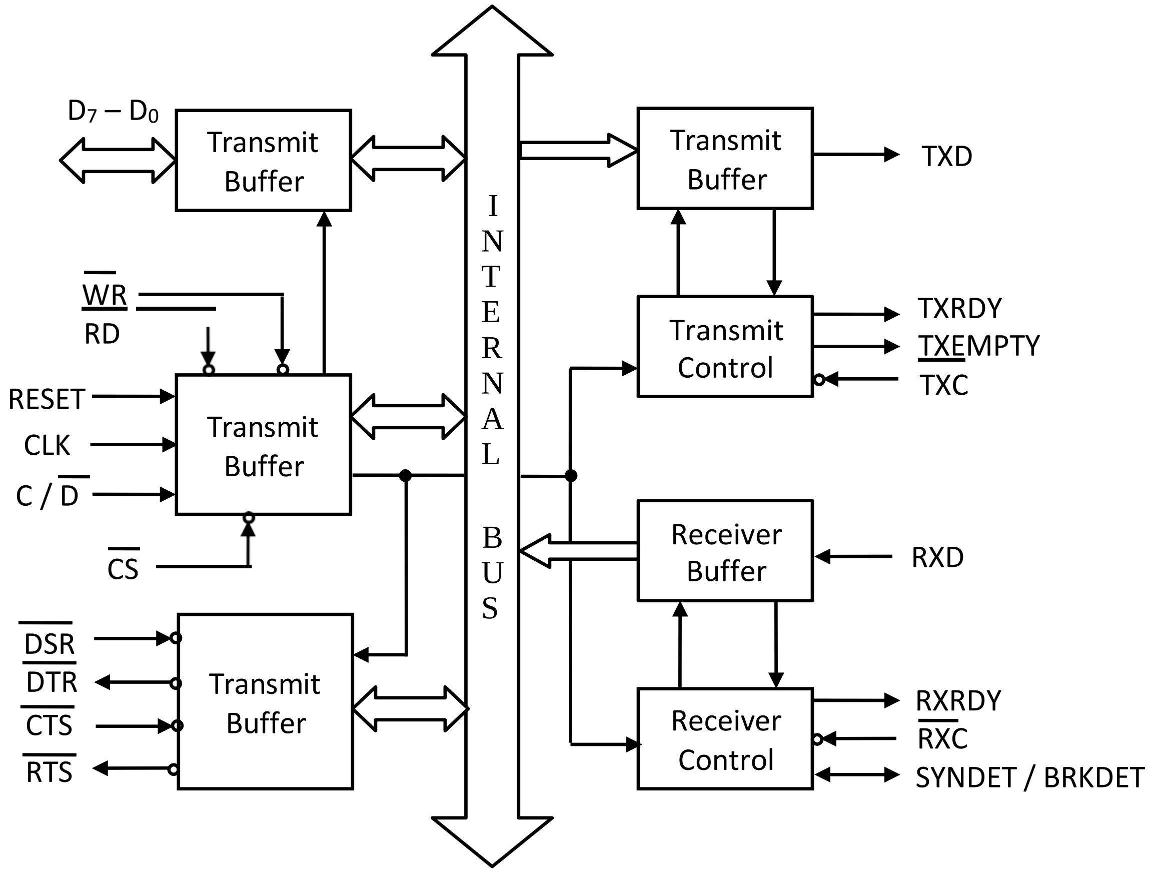


Fig.4.7 : 8251 USART Internal Architecture

The receive control unit decides the receiver frequency as controlled by the RXC input frequency. The receive control unit generates a receiver ready (RXRDY) signal that may be used by the CPU for handshaking. This unit also detects a break in the data string while the 8251 is in asynchronous mode. In synchronous mode, the 8251 detects SYNC characters using SYNDET/BD pin.

UNIT V: Introduction to microcontrollers (8051) and embedded systems: 8051 architecture, pin description, I/O configuration, interrupts, addressing modes, an overview of 8051 instruction set, embedded system, use of microcontrollers in embedded systems.

Features of 8051 Microcontroller:

The 8051-microcontroller architecture consists of the following features

(i) 8-bit CPU with accumulator register A and register B
(ii) 16-bit program counter (PC) and data pointer register (DPTR)
(iii) 8-bit Program status word register (PSW)
(iv) 8-bit stack pointer register (SP)
(v) Internal 128 bytes RAM and 4K bytes ROM
(vi) 32 I/O pins arranged as four 8-bit ports ($P_0 – P_3$)
(vii) Two 16-bit timer / counters (T_0 and T_1)
(viii) Full duplex serial data communication
(ix) Two external and three internal interrupt sources
(x) Oscillator and clock circuits

Comparison between Microprocessor & Microcontroller:

Sl. No.	Microprocessor	Microcontroller
1	Microprocessor contains ALU, General purpose registers, stack pointer, program counter, clock timing circuit, interrupt circuit	Microcontroller contains the circuitry of microprocessor, and in addition it has built in ROM, RAM, I/O Devices, Timers/Counters etc.
2	There are few bit handling instructions	There are many bit handling instructions
3	Few no. of pins are multifunctional	More no. of pins are multifunctional
4	This system required additional hardware	This system requires less additional hardware
5	Large no. of instructions with flexible addressing modes	Limited no. of instructions with few addressing modes
6	More access time to memory and IO devices, as they are connected externally	Less access time for built in memory and IO devices.

8051 architecture:

The register organisation in 8051 microcontrollers is as listed below

(i) Accumulator is an 8 bit register widely used for all arithmetic and logical operations. Accumulator is also used to transfer data between external memory.
(ii) B register is used along with Accumulator for multiplication and division. A and B registers together is also called MATH registers.
(iii) PSW (Program Status Word). This is an 8 bit register which contains the arithmetic status of ALU and the bank select bits of register banks.
(iv) Stack Pointer (SP) – it contains the address of the data item on the top of the stack. Stack may reside anywhere on the internal RAM. On reset, SP is initialized to 07 so that the default stack will start from address 08 onwards.
(v) Data Pointer (DPTR) – DPH (Data pointer higher byte), DPL (Data pointer lower byte). This is a 16 bit register which is used to furnish address information for internal and external program memory and for external data memory.
(vi) Program Counter (PC) – 16 bit PC contains the address of next instruction to be executed. On reset PC will set to 0000. After fetching every instruction PC will increment by one.

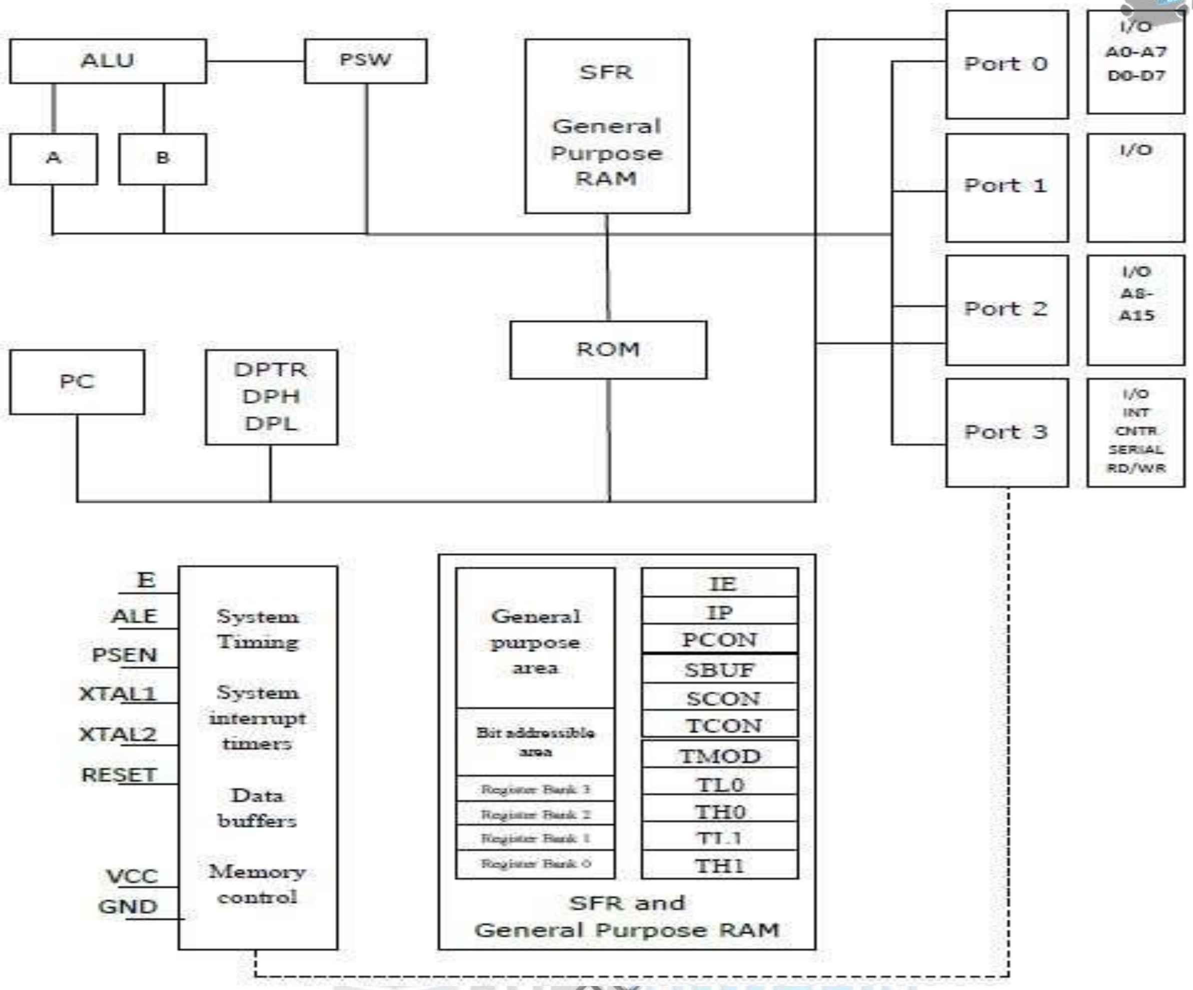


Fig. : Internal Architecture of 8051 Microcontroller

Program Status Word (PSW) Register:

This register indicates the status of the result of some instructions. Each of the bit is called as a flag. These flag bits are tested by some of the conditional branch instructions to make decisions based on the flag states. The flags are grouped into a program status register.

The 8051 has four arithmetic flags which are modified by the arithmetic operations and three general purpose user flags which can be modified by user as desired. The fig. below shows the flag bits and its function.

7	6	5	4	3	2	1	0
CY	AC	F_0	RS_1	RS_0	OF	-	P

Bit	Symbol	Description
7	CY	Carry flag used in arithmetic, logical and jump instructions
6	AC	Auxiliary carry used in BCD operation
5	F0	User flag 0
4	RS_1	Register bank select bit 1
3	RS_0	Register bank select bit 0
2	OF	Overflow flag used in arithmetic operations
1	-	Reserved for future use
0	P	Parity flag, shows the parity of register A ; 1 if odd parity

RS_1	RS_0	Function
0	0	Select register bank 0
0	1	Select register bank 1
1	0	Select register bank 2
1	1	Select register bank 3

The internal 128 bytes RAM structure is as shown in the Fig. below

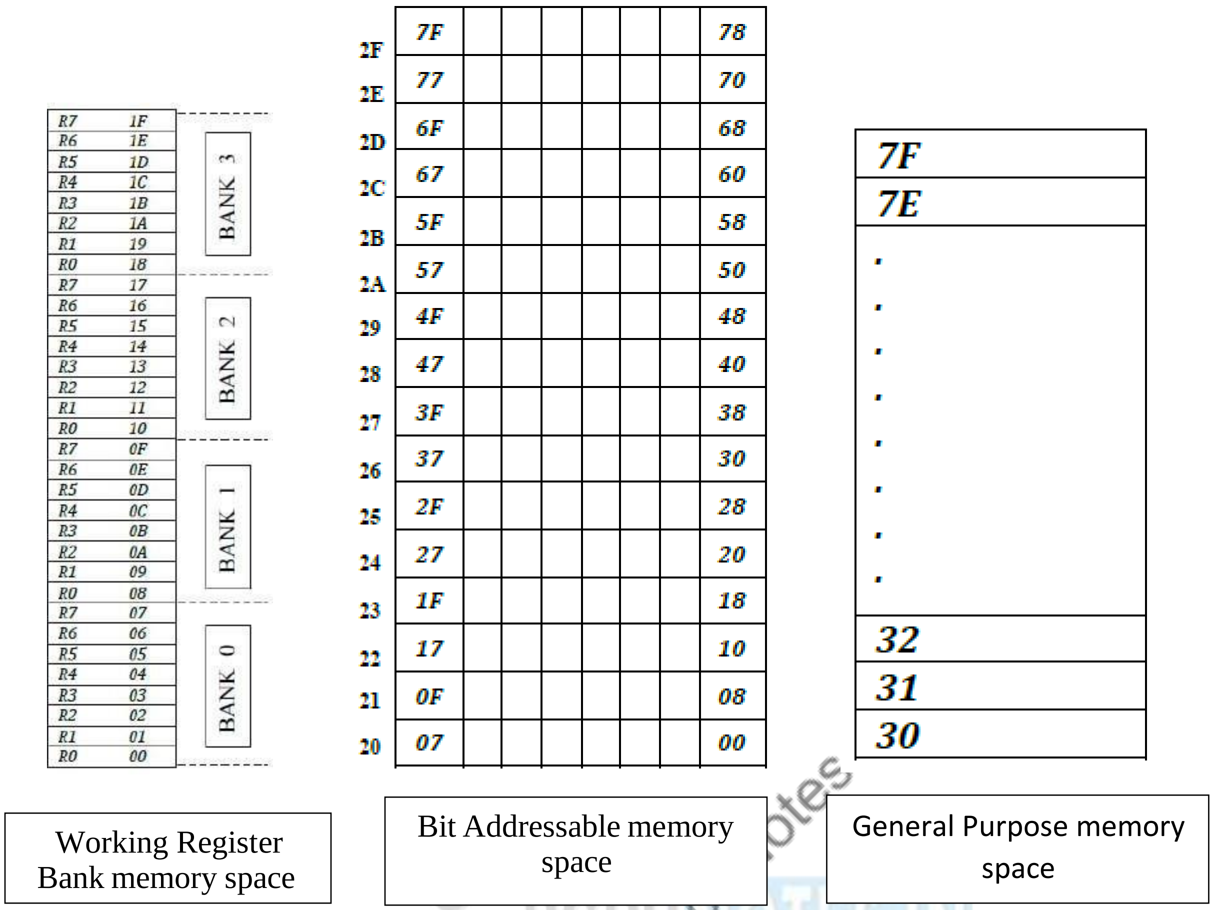


Fig. : Internal Memory organisation of 8051 microcontroller

Pin Description:

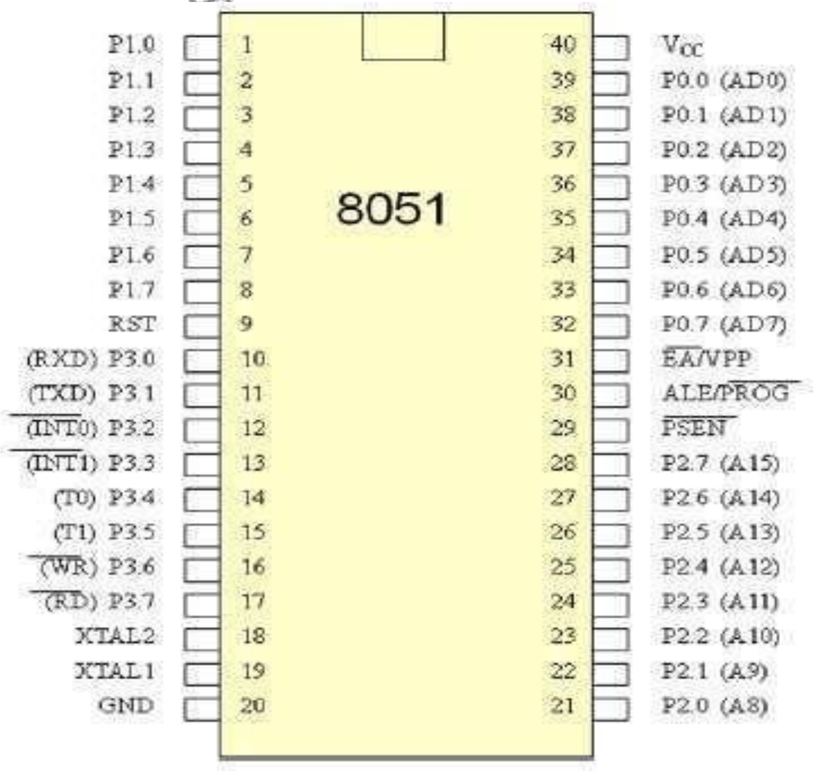


Fig. : Pin diagram of 8051 Microcontroller

Pin No.	Description
Pins 1-8	**PORT 1**. Each of these pins can be configured as an input or an output.
Pin 9	**RESET**. A logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning.
Pins10-17	**PORT 3**. Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions
Pin 10	**RXD.** Serial asynchronous communication input or Serial synchronous communication output.
Pin 11	**TXD.** Serial asynchronous communication output or Serial synchronous communication clock output
Pin 12	**INT0.**External Interrupt 0 input
Pin 13	**INT1.** External Interrupt 1 input
Pin 14	**T0.** Counter 0 clock input
Pin 15	**T1.** Counter 1 clock input
Pin 16	**WR.** Write to external (additional) RAM
Pin 17	**RD.** Read from external RAM
Pin 18, 19	**XTAL2, XTAL1.** Internal oscillator input and output. A quartz crystal which specifies operating frequency is usually connected to these pins.
Pin 20	**GND. Ground.**
Pin 21-28	**Port 2**. If there is no intention to use external memory, then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64Kb is not used, which means that not all eight port bits are used for its addressing, the rest of them are not available as inputs/outputs.
Pin 29	**PSEN.** If external ROM is used for storing program, then a logic zero (0) appears on it every time the microcontroller reads a byte from memory.
Pin 30	**ALE.** Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output. After receiving signal from the ALE pin, the external latch latches the state of P0 and uses it as a memory chip address. Immediately after that, the ALE pin is returned its previous logic state and P0 is now used as a Data Bus.
Pin 31	**EA**. By applying logic zero to this pin, P2 and P3 are used for data and address transmission with no regard to whether there is internal memory or not. It means that even there is a program written to the microcontroller, it will not be executed. Instead, the program written to external ROM will be executed. By applying logic one to the EA pin, the microcontroller will use both memories, first internal then external (if exists).
Pin 32-39	**PORT 0**. Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).
Pin 40	**VCC**. +5V power supply.

Addressing Modes:

8051 Microcontroller supports six addressing modes
(i) Direct Addressing Mode (ii) Indirect Addressing Mode (iii) Register Addressing Mode (iv) Register Specific (v) Immediate addressing mode (vi) Indexed Addressing Mode

(i) Direct Addressing Mode: In this mode, the operands are specified using the 8-bit address field in the instruction format.
e.g., MOV R_0 , 89H

(ii) Indirect Addressing Mode: In this mode the 8-bit address of an operand is stored in a register. The 16-bit address in stored in DPTR register only.

e.g., MOV A, @R_0

(iii) Register Addressing Mode: In this mode the operands are stored in the registers $R_0 – R_7$ of the selected bank.

e.g., ADD A, R_2

(iv) Register Specific: In this mode the operand is implicitly specified using one of the registers.

e.g. RLA ; rotates the accumulator data left.

(v) Immediate addressing mode: In this mode, an immediate data, is specified in the instruction.

e.g., ADD A, # 20H

(vi) Indexed Addressing Mode: Only program memory can be accessed using this addressing mode. The program counter or the data pointer register are the allowed 16-bi address storage registers.

```
e.g.,  MOVC  A,  @A+DPTR
       JMP  @A+DPTR
```

8051 Instructions Set:

The instructions of 8051 can be broadly classified under the following headings.

I. Data transfer instructions
II. Arithmetic instructions
III. Logical instructions
IV. Branch instructions
V. Subroutine instructions
VI. Bit manipulation instructions

I. Data transfer instructions:

In this group, the instructions perform data transfer operations of the following types.

- ✓ Move the contents of a register Rn to A

```
MOV A,R2
MOV A,R7
```

- ✓ Move the contents of an external memory to A or A to an external memory

```
MOVX A,@R1
MOVX @R0,A
```

- ✓ Move the contents of program memory to A

```
MOVC A, @A+PC
MOVC A, @A+DPTR
```

- ✓ Push and Pop instructions

- ✓ Exchange instructions
The content of source ie., register, direct memory or indirect memory will be exchanged with the contents of destination ie., accumulator.

```
XCH A,R3
XCH A,@R1
```

- ✓ Exchange digit. Exchange the lower order nibble of Accumulator (A0-A3) with lower order nibble of the internal RAM location which is indirectly addressed by the register.

```
XCHD A,@R1
XCHD A,@R0
```

II. Arithmetic instructions:

The 8051 can perform addition, subtraction. Multiplication and division operations on 8 bit numbers.

- ✓ Addition

  In this group, we have instructions to add the contents of A with source data with or without carry.

  ```
  ADD A, #45H
  ADDC A, #OB4H
  ```

  CY AC and OV flags will be affected by this operation.

- ✓ ***Subtraction***

  In this group, we have instructions to subtract the contents of A with source data with or without carry.

  ```
  SUBB A, #45H
  SUBB A, #OB4H
  ```

- ✓ ***Multiplication***

  MUL AB. This instruction multiplies two 8 bit unsigned numbers which are stored in A and B register. After multiplication the lower byte of the result will be stored in accumulator and higher byte of result will be stored in B register.

  ```
  MOV A,#45H              ;[A]=45H
  MOV B,#0F5H       ;[B]=F5H
  MUL AB                  ;[A] x [B] = 45 x F5 = 4209
                    ;[A]=09H, [B]=42H
  ```

- ✓ ***Division***

  DIV AB. This instruction divides the 8 bit unsigned number which is stored in A by the 8 bit unsigned number which is stored in B register. After division the result will be stored in accumulator and remainder will be stored in B register.

  ```
  MOV A,#45H  ;[A]=0E8H
  MOV B,#0F5H ;[B]=1BH
  DIV AB        ;[A] / [B] = E8 /1B = 08 H with remainder 10H
               ;[A] = 08H, [B]=10H
  ```

- ✓ **Increment:** increments the operand by one. **INC A**
- ✓ **Decrement:** decrements the operand by one. **DEC A**

III. Logical instructions:

- ✓ **Logical AND**

  ANL destination, source: ANL does a bitwise "AND" operation between source and destination, leaving the resulting value in destination. The value in source is not affected. "AND" instruction logically AND the bits of source and destination.

  ANL A,#DATA ANL A, Rn
  ANL A,DIRECT ANL A,@Ri

- ✓ **Logical OR**

  ORL destination, source: ORL does a bitwise "OR" operation between source and destination leaving the resulting value in destination. The value in source is not affected. " OR " instruction logically OR the bits of source and destination.

  ORL A,#DATA ORL A, Rn
  ORL A,DIRECT ORL A,@Ri

- ✓ **Logical Ex-OR**
  XRL destination, source: XRL does a bitwise "EX-OR" operation between source and destination, leaving the resulting value in destination. The value in source is not affected. " XRL " instruction logically EX-OR the bits of source and destination.
  XRL A,#DATA XRL A,Rn
  XRL A,DIRECT XRL A,@Ri

- ✓ **Logical NOT**
  CPL complements operand, leaving the result in operand. If operand is a single bit, then the state of the bit will be reversed. If operand is the Accumulator, then all the bits in the Accumulator will be reversed.
  CPL A, CPL C, CPL bit address
- ✓ **SWAP A** – Swap the upper nibble and lower nibble of A.
- ✓ **Rotate Instructions**
  RR A
  This instruction is rotate right the accumulator. Its operation is illustrated below. Each bit is shifted one location to the right, with bit 0 going to bit 7.
  RL A
  Rotate left the accumulator. Each bit is shifted one location to the left, with bit 7 going to bit 0
  RRC A
  Rotate right through the carry. Each bit is shifted one location to the right, with bit 0 going into the carry bit in the PSW, while the carry was at goes into bit 7
  RLC A
  Rotate left through the carry. Each bit is shifted one location to the left, with bit 7 going into the carry bit in the PSW, while the carry goes into bit 0.

IV. **Branch instructions:**

- ✓ **Jump and Call Program Range**
  - There are 3 types of jump instructions. They are:-
    - ➢ Relative Jump : Only 1 byte of jump address needs to be specified in the 2's complement form, ie. For jumping ahead, the range is 0 to 127 and for jumping back, the range is -1 to -128.

    - ➢ Short Absolute Jump: In this case only 11bits of the absolute jump address are needed. In 8051, 64 kbyte of program memory space is divided into 32 pages of 2 kbyte each. The instruction length becomes 2 bytes.

    - ➢ Long Absolute Jump: Applications that need to access the entire program memory from 0000H to FFFFH use long absolute jump. Since the absolute address has to be specified in the op-code, the instruction length is 3 bytes (except for JMP @ A+DPTR). This jump is not re-locatable.

  - Another classification of jump instructions is
    - ➢ Unconditional Jump is a jump in which control is transferred unconditionally to the target location.
      L JMP (long jump). This is a 3-byte instruction. First byte is the op-code and second and third bytes represent the 16-bit target address which is any memory location from 0000 to FFFFH

      AJMP: this causes unconditional branch to the indicated address, by loading the 11 bit address to 0 -10 bits of the program counter. The destination must be therefore within the same 2K blocks.

      SJMP (short jump). This is a 2-byte instruction. First byte is the op-code and second byte is the relative target address, 00 to FFH (forward +127 and backward -128 bytes from the current PC value). To calculate the target address of a short jump, the second byte is added to the PC value which is address of the instruction immediately below the jump.

- Conditional Jump

  Conditional Jump instructions.

JBC	Jump if bit = 1 and clear bit
JNB	Jump if bit = 0
JB	Jump if bit = 1
JNC	Jump if CY = 0
JC	Jump if CY = 1
CJNE reg,#data	Jump if byte ≠ #data
CJNE A,byte	Jump if A ≠ byte
DJNZ	Decrement and Jump if A ≠ 0
JNZ	Jump if A ≠ 0
JZ	Jump if A = 0

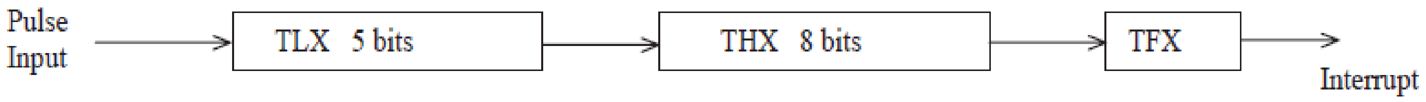
V. **Subroutine instructions:** Subroutines are handled by CALL and RET instructions.

VI. **Bit manipulation instructions:** 8051 has 128 bit addressable memory. Bit addressable SFRs and bit addressable PORT pins. It is possible to perform following bit wise operations for these bit addressable locations.

Timer Operation:

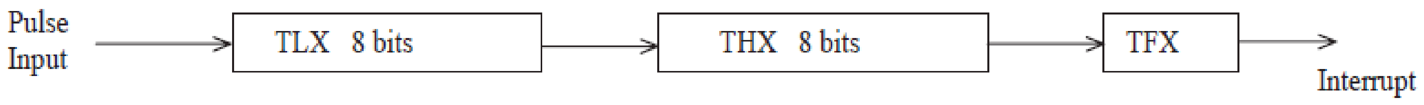
Timers can operate in four different modes. They are as follows

(i) ***Timer Mode-0:*** In this mode, the timer is used as a 13-bit UP counter as follows.

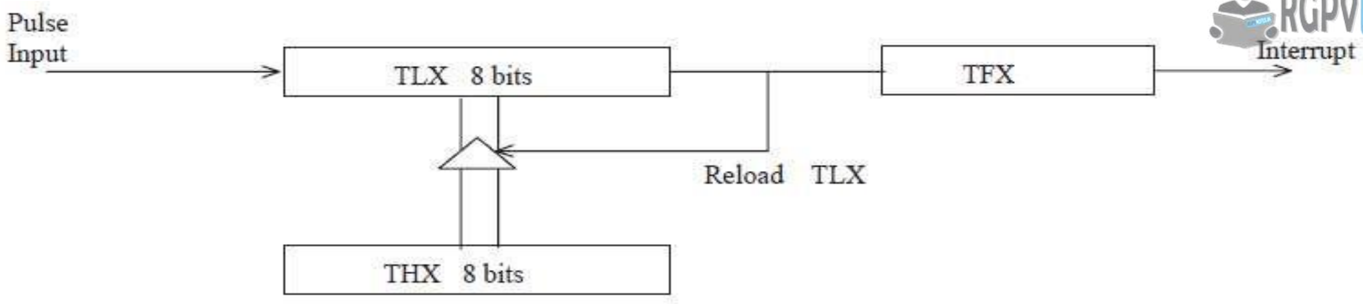


The lower 5 bits of TLX and 8 bits of THX are used for the 13 bit count. Upper 3 bits of TLX are ignored. When the counter rolls over from all 0's to all 1's, TFX flag is set and an interrupt is generated. The input pulse is obtained from the previous stage. If TR1/0 bit is 1 and Gate bit is 0, the counter continues counting up. If TR1/0 bit is 1 and Gate bit is 1, then the operation of the counter is controlled by input. This mode is useful to measure the width of a given pulse fed to input.

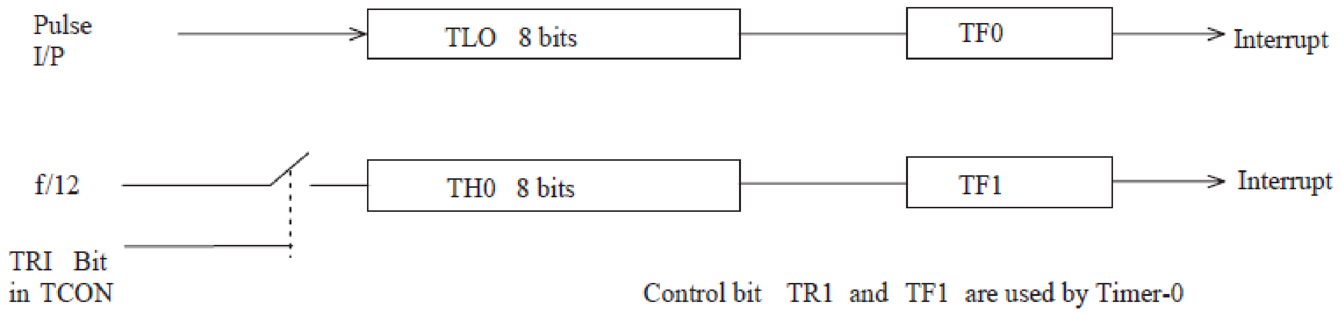
(ii) ***Timer Mode-1:*** This mode is similar to mode-0 except for the fact that the Timer operates in 16-bit mode.



(iii) ***Timer Mode-2: (Auto-Reload Mode):*** This is a 8 bit counter/timer operation. Counting is performed in TLX while THX stores a constant value. In this mode when the timer overflows i.e. TLX becomes FFH, it is fed with the value stored in THX. For example if we load THX with 50H then the timer in mode 2 will count from 50H to FFH. After that 50H is again reloaded. This mode is useful in applications like fixed time sampling.



(iv) Timer Mode-3: Timer 1 in mode-3 simply holds its count. The effect is same as setting TR1=0. Timer0 in mode-3 establishes TL0 and TH0 as two separate counters. Control bits TR1 and TF1 are used by Timer-0 (higher 8 bits) (TH0) in Mode-3 while TR0 and TF0 are available to Timer-0 lower 8 bits(TL0).



1. Write a program to subtract a 16 bit number stored at locations 51H-52H from 55H-56H and store the result in locations 40H and 41H. Assume that the least significant byte of data or the result is stored in low address. If the result is positive, then store 00H, else store 01H in 42H.

Solution:

```
ORG 0000H         ; Set program counter 0000H
MOV A, 55H        ; Load the contents of memory location 55 into A
CLR C             ; Clear the borrow flag
SUBB A,51H        ; Sub the contents of memory 51H from contents of A
MOV 40H, A        ; Save the LSByte of the result in location 40H
MOV A, 56H        ; Load the contents of memory location 56H into A
SUBB A, 52H       ; Subtract the content of memory 52H from the content A
MOV 41H, A        ; Save the MSbyte of the result in location 415.
MOV A, #00        ; Load 005 into A
ADDC A, #00       ; Add the immediate data and the carry flag to A
MOV 42H, A        ; If result is positive, store00H, else store 0lH in 42H
END
```

2. Write a program to add two Binary Coded Decimal (BCD) numbers stored at locations 60H and 61H and store the result in BCD at memory locations 52H and 53H. Assume that the least significant byte of the result is stored in low address.

Solution:

```
ORG 0000H         ; Set program counter 00004
MOV A,60H         ; Load the contents of memory location 6.0.H into A
ADD A,61H         ; Add the contents of memory location 61H with contents of A
DA A              ; Decimal adjustment of the sum in A
MOV 52H, A        ; Save the least significant byte of the result in location 52H
MOV A,#00         ; Load 00H into .A
ADDC A,#00H        ; Add the immediate data and the contents of carry flag to A
MOV 53H,A         ; Save the most significant byte of the result in location 53:,
END
```

3. Write a program to store data FFH into RAM memory locations 50H to 58H using indirect addressing mode

Solution:

```
        ORG 0000H           ; Set program counter 0000H
        MOV A, #0FFH        ; Load FFH into A
        MOV RO, #50H        ; Load pointer, R0-50H
        MOV R5, #08H        ; Load counter, R5-08H
Start:  MOV @RO, A          ; Copy contents of A to RAM pointed by R0
        INC RO              ; Increment pointer
        DJNZ R5, start      ; Repeat until R5 is zero
        END
```

4. Write a program to compute 1 + 2 + 3 + N (say N=15) and save the sum at70H

Solution:

```
        ORG 0000H           ; Set program counter 0000H
        N EQU 15
        MOV R0,#00          ; Clear R0
        CLR A               ; Clear A
again:  INC R0              ; Increment R0
        ADD A, R0           ; Add the contents of R0 with A
        CJNE R0,#N,again    ; Loop until counter, R0, N
        MOV 70H,A           ; Save the result in location 70H
        END
```