B.E.

Fifth Semester Examination, 2009-2010

Microprocessors and Inter-facing (EE-309-E)

Note : Attempt any five questions. All questions carry equal marks.

Q. 1. (a) Write the purpose and working of following components in 8085 :

(i) Instruction register

(ii) ALU and flag register

(iii) Serial input/output control

(iv) Stack pointer

(v) Program Counter.

Ans. (i) Instruction Register : In a typical processor operation, the processor first fetches the opcodes of instruction from memory (i.e, it places an address on the address bus and memory responds by placing the data stored at the specified on the data bus). The CPU stores this opcode in a register called the instruction register. This opcode is further sent to the instruction decoder to select one of the 256 alternatives.

(ii) ALU and Flag Register : Arithmetic Logic Unit perform arithmetic and logical function on eight bit variables. The arithmetic unit performs bitwise fundamental arithmetic operations such as addition and subtraction. The logic unit performs logical operation such as complement AND, OR and EXOR as well as rotate and clear. The ALU can perform 20 arithmetical and 14 logical operations. Different operations are selected by providing different control signal to the ALU. Arithmetic operations are done by using 2's complement addition/substraction in circuits in the ALU.

Flag Register : It is an 8-bit register in which five of the bits carry significant information in the form of flags-S (sign flags), Z (Zero), AC (Auxillary carry), P (Parity flag) CY (carry flag).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	CY

(iii) Serial Input/Output Control : During data transmission over long distances, it is necessary to transmit data bit by bit, to reduce the cost of cabling. In serial communication one bit is transferred at a time over a single line. The 8085 serial input/output port of provides two lines are SOD and SID for serial communication. The serial output data (SOD) line is used to send data serially and serial inputs data (SID) line is used to receive data serially.

(iv) Stack Pointer : The stack is a reserved area of the memory in the RAM where temporary informations may be stored. A 16-bit stack pointers is used to hold the address of the most recent stack entry (address of the last entry). It is user accessable register and stack normally operated in the last in first out (LIFO).

(v) Program Counter (PC) : Program is a sequence of instructions. Microprocessor fetches three instructions from the memory and executes then sequentially. The program counter in a special purpose register which at a given time, stores the address of the next instruction to be fetched. Program counter acts on a pointer to the next instruction. How processor increments program counter depends on the nature of the instructions. For one byte instruction it increments program counter by one, for two byte instruction it increments program counter by two such that program counter always points to the address to the next instruction.

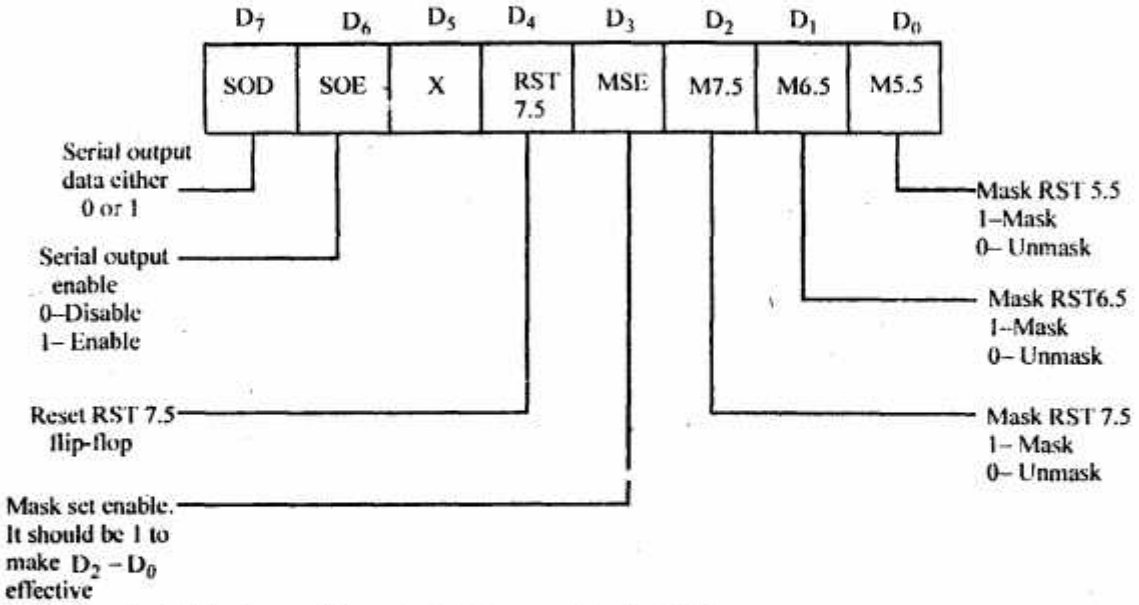
Q. 1. (b) Describe the use of SIM instruction in 8085.

Ans. SIM (Set Interrupt Mask) :

Opcode	Operand	Byte	M-cycle	T-states	Addressing mode
SIM	None	1	1	4	None

Description : This instruction make the interrupts are desired. It also sends out serial data through the SOD pin. For this instruction command byte must be loaded in the accumulator.

The pattern for command byte is,



Example : SIM : Assume the contents of a commulator in OEH.

SOD	SOE	X	RST 7.5	MSE	M7.5	M6.5	M5.5
0	0	0	0	1	1	1	0

→ Accumulator → OE

This instruction will mesh RST 7.5 and RST 6.5 interrupts whereas RST 5.5 interrupt will be unmarkcd. It will also desirable serial output.

Q. 2. (a) Two 8 bit numbers are stored in the location 2401H and 2402 H. Multiply them and store the result in the location 2403H and 2404 H (LS Byte at 2403 and MS Byte at 2404 H). Program to be written in 8085 assembly language.

Ans. (2401H) = 03H (as a multiplexer)

n/

(2402H) = B2H (as a multiplicand)

Result = B2 + B2 + B2 = 216H
(2403 H) = 16H
(2404 H) = 02H

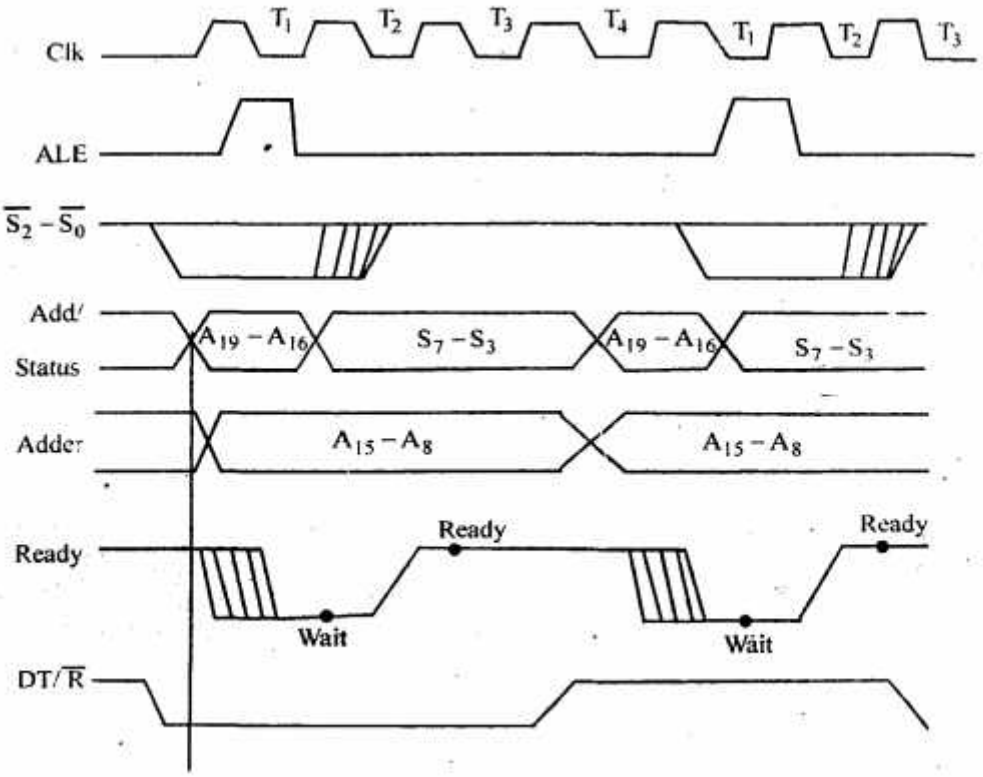
Source Program ;

```
    LDA 2401 H;          Initialize multiplier
    MOVE, A
    MVI D, 00;           Move immediate 00 in DE register.
    LDA 2402 H;          Initialize multiplicand
    MOV C, A;            Initialize multiplier
    LXI H, 0000H;        Result = 0
Back : DAD D;            Decrement multiplier
    JNZ Back;            If multiplier ≠ 0 repeat
    SHLD 2403H;          Store result.
    HLT;                 Terminate program
```

Q. 2. (b) Make the timing diagram of the instruction MVI A, 4FH. The corresponding machine code is

2000 H 3EH
2001 H 4FH

Ans.



Q. 3. (a) Write the purpose of following flags in 8086 :

(i) Trap flag

(ii) Interrupt-flag

(iii) Direction-Flag.

Ans. (i) Trap Flag : One way to debug a program is to run the program one instruction at a time and see the contents of used registers and variables after execution of every instructions. This process is called single stepping through a program. Trap flag is used to single stepping through a program. Trap flag is used for single stepping through a program. If set, a trap is executed after execution of each instruction. i.e., interrupt service routine is executed which displays various registers and memory variable contents on the display after execution of each instructions. This programmer can easily trace and correct errors in the program.

(ii) Interrupt Flag : It is used to allow prohibit the interruptions of a program. If set, a certain type of interrupt (a maskable interrupt) can be recognized by the 8086, otherwise there interrupt are ignored.

(iii) Direction Flag : It is used with string instruction. If DF = 0, the string is processed for its beginning with the first elements having lowest address otherwise, the string is process from the high address towards the low address.

Q. 3. (b) Explain the following addressing modes in context with 8086 :

(i) Direct **(ii) Register**

(iii) Register indirect **(iv) Based Index**

(v) Immediate.

Ans. (i) Direct Addressing : In direct addressing an unsigned 16-bit displacement, will be specified in the instruction. The displacement is the effective address (EA) or offset. In case of 8-bit displacement the effective address is obtained by sign extending the 8-bit displacement to 16 bit.

Example : MOV DX, [08H]

EA = 0008H (Sign extended 8-bit displacement)

$BA = (DS) \times 16_{10}$; MA = BA + EA

(DX) ← (MA) or (OL) ← (MA)

(OH) ← (MA + 1)

MA = Memory address

BA = Base Address

(ii) Register Addressing : In register addressing the instruction will specify the name of the register which holds the data to be operated by the instruction :

Example : MOV CL, DH (CL) ← (DH)

MOV BX, DX (BX) ← (DX)

(iii) Register Indirect Addressing : In this addressing mode the name of register in which holds the effective address (EA) will specified in the instruction. The register used to hold the effective address are BX, SI and DI. The content of DS is used for segment base address calculation. When segment override prefix is employed, the content-register specified in the override prefix will be used for base address calculations instead of DS register.

Example : MOV. CX, [BX]

$EA = (BX) : BA = DS \times 16_{10} ; MA = BA + EA$

$(CX) \leftarrow (MA)$ or $(CL) \leftarrow (MA)$

$(CH) \leftarrow (MA + 1)$

(iv) Based Index Addressing : In this addressing mode the effective address is given by sum of base value, index value and an 8 bit or 16-bit displacement specified in the instruction. The base value is stored in BX or BP register. The index value is stored in SI and DI register. In case of 8-bit displacement, it is extended to 16-bit before addressing to base value. This type of addressing will be useful in addressing two dimensional arrays where we require no modifies.

Example: MOV DX, [BX + SI + OAH]

$OOOAH \xleftarrow{\text{Signed extend}} OAH ; EA = (BX) + (SJ) + OOOAH$

$BA = (DS) \times 16_{10} ; MA = BA + EA$

$(DX) \leftarrow (MA)$ or $(DL) \leftarrow (MA)$

$(DH) \leftarrow (MA + 1)$

(v) Immediate Addressing : In immediate addressing mode an 8-bit or 16-bit data is specified as part of the instruction.

Example:

1. MOV DL, 8OH $(DL) \leftarrow O8H$

The 8-bit data (O8H) given in the instruction is moved to DL register.

2. MOV AX, OA8FH $(AX) \leftarrow OA8FH$

16-bit data (OA8F) given in the instruction is moved to AX register.

Q. 3. (c) Differentiate between ADD AX, [BX] and ADD AX, BX.

Ans. ADD AX, [BX]. In this instruction,

$[AX] \leftarrow (AX) + [BX]$

The content of register [BX] location which address is in [BX] register added and the result is stored in register.

$\Rightarrow$ ADD AX, BX

$(AX) \leftarrow (AX) + (BX)$

The content of two register and added and the result is stored in register AX.

Q. 4. (a) Write a program in assembly language of 8086 to compare two strings and print appropriate message i.e., "strings are same" or "strings are different".

Ans.

ASSUME CS : CODE ;	ASSEMBLER Directive
ORG 1000H;	Assembler directive
START : MOV SI, 1100H;	Set SI register as a pointer for array
MOV D1; 1200;	Load the address of data to search DI register
MOV DL, [DI];	Get the data to search is DL register

```
        MOV BL, 01H;              Set BL reg. on position count.
        MOV AL, [SI];             Get first element of array in AL register
        AGAIN : CMPAL, DL:        Compare an element of array with the data to search
        JZ AVAIL;                 If data are equal then jump to AVAIL.
        INC SJ;                   If data are not equal, increment address pointed
        INC BL;                   Increment position count
        MOV AL, [SI];             Get the next element of array in AL register
        CMPAL, 20H;               Check for end of array.
        JNZ AGAIN;                If not end repeat search otherwise go to NOTAVA
NOTAVA : MoV CX, 0000H;           If search data is not found others store Zero
        MOV [DI + 1], CX
        MOV [DI + 3], CX
        JMP                       OVER
AVAIL : MOV BH, OFFH
        MOV [DI+1], BH;           Store FFH to indicate availability of data
        MOV [DI +2], BL;          Store the position of data
        MOV [DI + 3], SI;         Store the address of data
        OVER : HLT
        CODE : ENDS;              Assembler directive
        END;                      Assembler direction
```

Q. 4. (b) Write a program in 8086 assembly language which reads a character. If it is small alphabet converts it into capital alphabet and vice-versa. Otherwise it prints the message "Not an alphabet."

Ans.

```
        START : MOV AH            07TH : Read the character from keyboard.
          INT 21H
        * IFAL > = 'A' && ALO < = 'Z';   If characters in lower cases
          ADD AL, 20H;            Add 20H
        * ENDIF
          MOV AH, 02H; Display character on video screen where
          MOV DL, A               ASCII code is stored in DL
          JNT 21H
          END START
          END.
```

Q. 5. (a) The CS = 429EH and IP = 4ABCH. Find the corresponding absolute physical address.

Ans. CS = 429EH

IP = 4ABC H

Absolute Physical Address :

= IP – CS

= 4ABCH – 429EH

= 081EH

Absolute physical address = 081EH.

Q. 5. (b) Describe the purpose of model directive in 8086 assembly language.

Ans. Model Directive in 8086 Assembly Language : Assembler directives are defined in form of words b and these are directions to the assembler. These are not the instruction for 8086. These directives are used to unsight and execute programmes either turbo assembler (TASM) or macro-assembler (MASM). In additions to instruction, programmer contains assembler directives which are directions to assembler, called assembler directives for pseudo-instruction these are various directives used for 8086 assembly language programming.

Example : DB, DW, DQ and DT directions.

These directives is used to tell the assembler to deflective a variable of in program as per their length. The DB directive specifies the data is of type byte and stands for define type.

DW Define Wound : The DW directive is used to tell the assembler to define a variable type word 'or' reverse storage locations of type word is memory location.

DD Define Double Word : The DD directive is used to tell the assembler to define a variable of type double word 'or' to reverse memory location which can be access as type double word 'or' four bytes.

Q. 5. (c) Write the purpose of the following instruction in 8086 assembly with example :

(i) Loop **(ii) Rep**

(iii) Aam **(iv) Movsb**

(v) Xlat.

Ans. (i) Loop : Loop instruction are used to execute a group of instructions, a number of times as specified by a count value stored in CX register. The number of instructions to be looped will be specified directly is the instruction as or signed eight bit number (displacement or disk 8). For positive displacement the instruction below the loop instuction one executed and for negative displacement the instruction above the loop instructions are executed. The :ontent of CX register is decremented by one after reach execution of looked instructions. The effective address of first instruction of the loop is obtained by sign extending the disk 8 to 916 bit and adding to JP.

Example : LOOPZ disk P8

Loop if (CX) $\neq 0$ and ZF = 1

(CX) $\leftarrow$ (CX)–1

(ii) Rep Instruction : REPZ/REPE

While CX $\neq 0$ and ZF = 1, repeat execution of string instruction and (CX) $\leftarrow$ (CX)–1.

It is a prefix used for compare or scan string instruction when or string instruction is prefixed with RED2/ REPT, the instruction execution is repeated if CX $\neq 0$ and ZF =1 . After each execution of string instructions, the output of CX is decremented by 1. The repeat operation is terminated if CX = 0 or ZF = 0.

(iii) Aam Instruction : BCD adjust after multiply. After the two unpacked BCD digits are multiplied, the AAM instruction adjust the product to two unpacked BCD digits in AX.

Example :

```
AL = 0000 0100 = Unpacked BCD 4
CL = 0000 0110 = Unpacked BCD 6
MUL CI        ; ALXCL, Result in AR
              ; AX = 0000 0010 0000 0100 = 0204H
              ; Which is unpacked BCD for 24.
```

(iv) Movsb :

$$MA = (DS) \times 16_{10} + (SI)$$

$$MA_E = (ES) \times 16_{10} + (DI)$$

$$(MA)_E \leftarrow (MA)$$

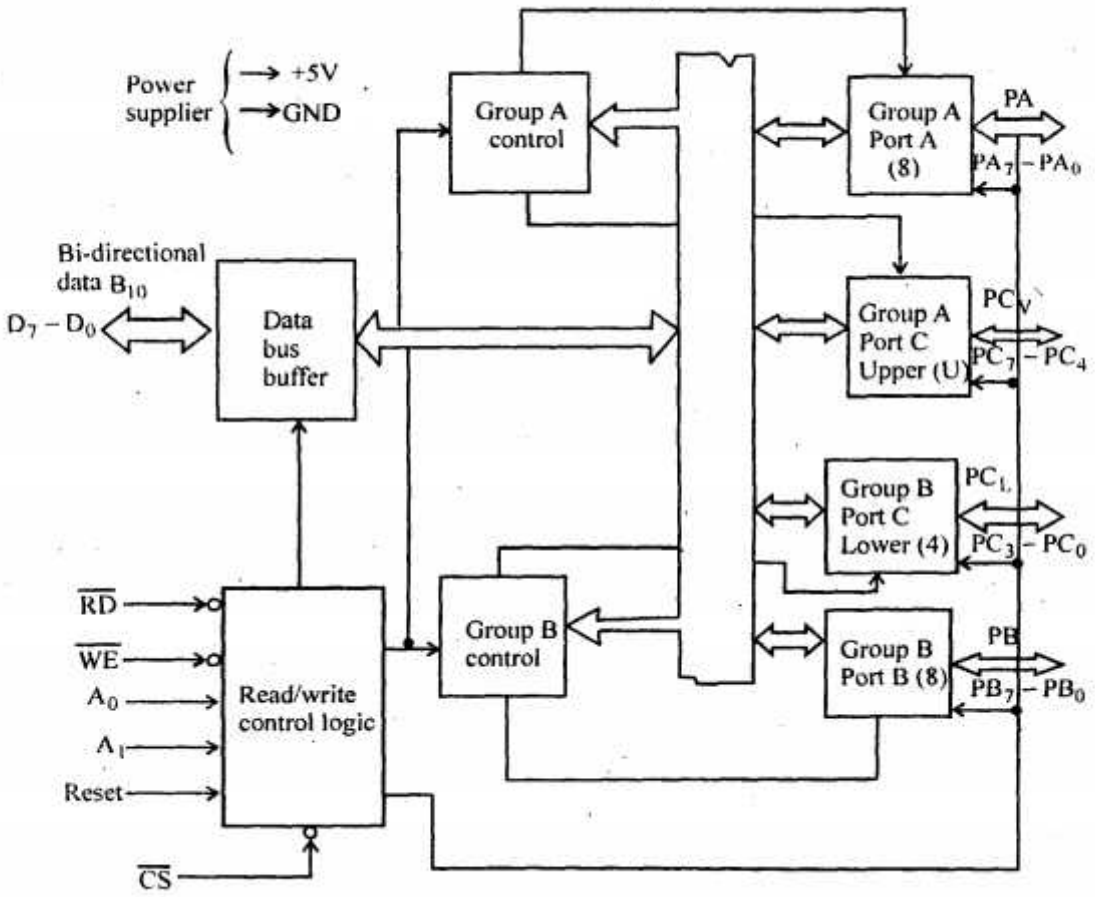
IF DF = 0, then $(DI) \leftarrow (DI) + 1$

$$(SI) \leftarrow (SI) - 1$$

One byte of a string data stored in data segment is copied into extra segment, and SJ and DI one automatically (incremented) decremented by 1 depending on direction flag (DF).

Q. 6. Explain the working of 8255 in mode 2 and BSR mode. Also explain how the contents of control register are interpreted in BSR and input/output mode.

Ans. The Internal Block Diagram of 8255 : It consists of data bus buffer, control logic and group A and group B control.



Block Diagram of 8255

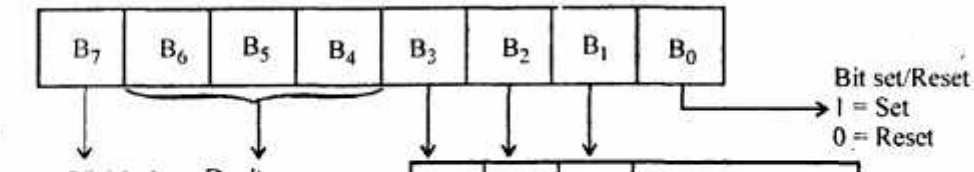
Operations Modes :

Bit Set-Reset (BSR) Modes : The individual bits of port-C can be set or reset by sending out a single OUT instruction to the control register. When PORT C is used for control/status operations, this features can be used to set or reset individual bits.

MODE 2 : Bidirectional Input/Output Data Transfer : This mode allows bidirectional data transfer over a single 8-bit data bus using handshaking signals. This features is available only in Group A with port A as the 8-bit bidirectional data bus and $PC_3 - PC_7$ d are used for handshaking purpose. In this mode, both input and outputs are latched. Due to use of a single 8-bit data bus connecting it to the peripheral, only when the peripheral request it. The remaining lines of port C is $PC_0 \cdot PC_2$ can be used for simple input/output function. The port-B can be programmed in mode –0 or in mode–1. When port-B is programmed is mode –1, $PC_0 - PC_2$ lines of Port-C are used as handshaking signals.

The 8255 has two control words.

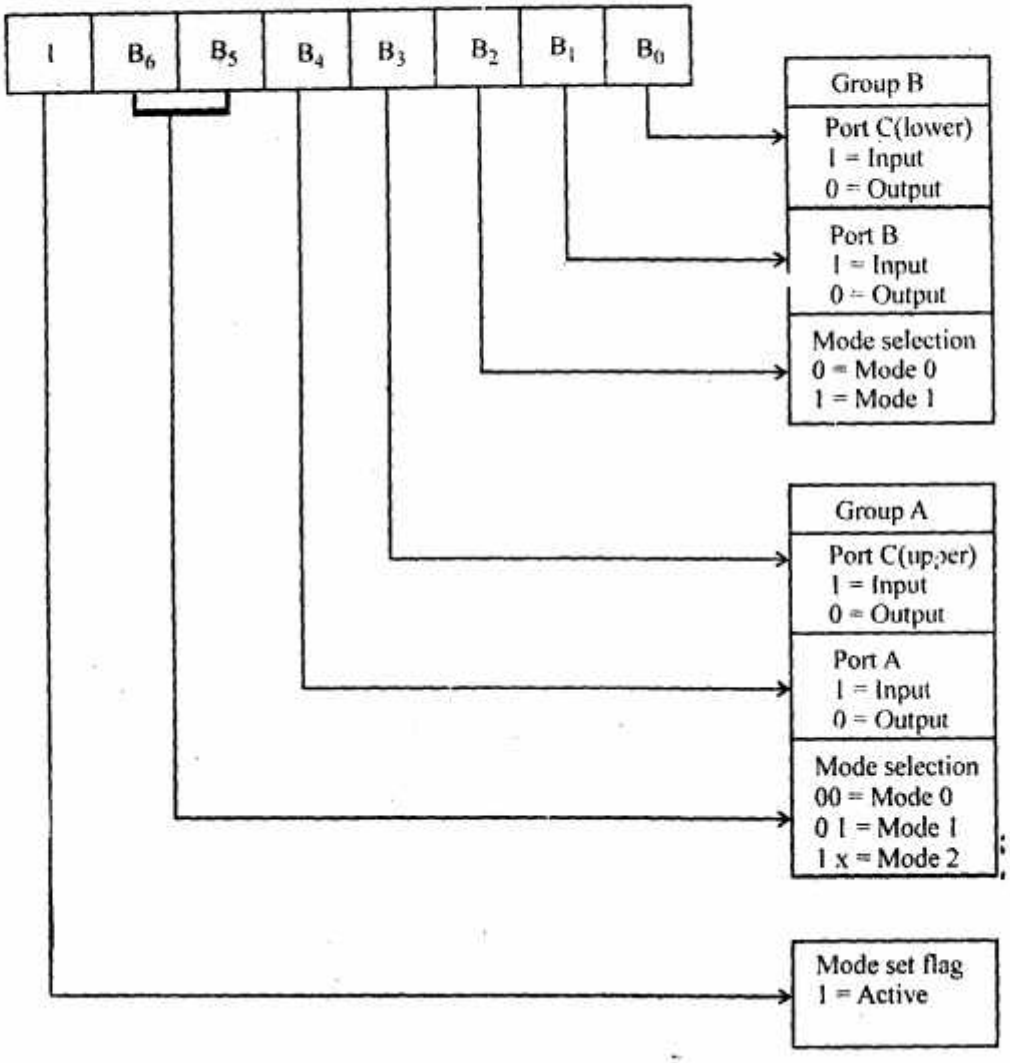
1. For Bit Set/Reset Mode : Below shows bit set/reset control word format.



0	0	0	Set/Reset	PC_0
0	0	1	Set/Reset	PC_1
0	1	0	Set/Reset	PC_2
0	1	1	Set/Reset	PC_3
1	0	0	Set/Reset	PC_4
1	0	1	Set/Reset	PC_5
1	1	0	Set/Reset	PC_6
1	1	1	Set/Reset	PC_7

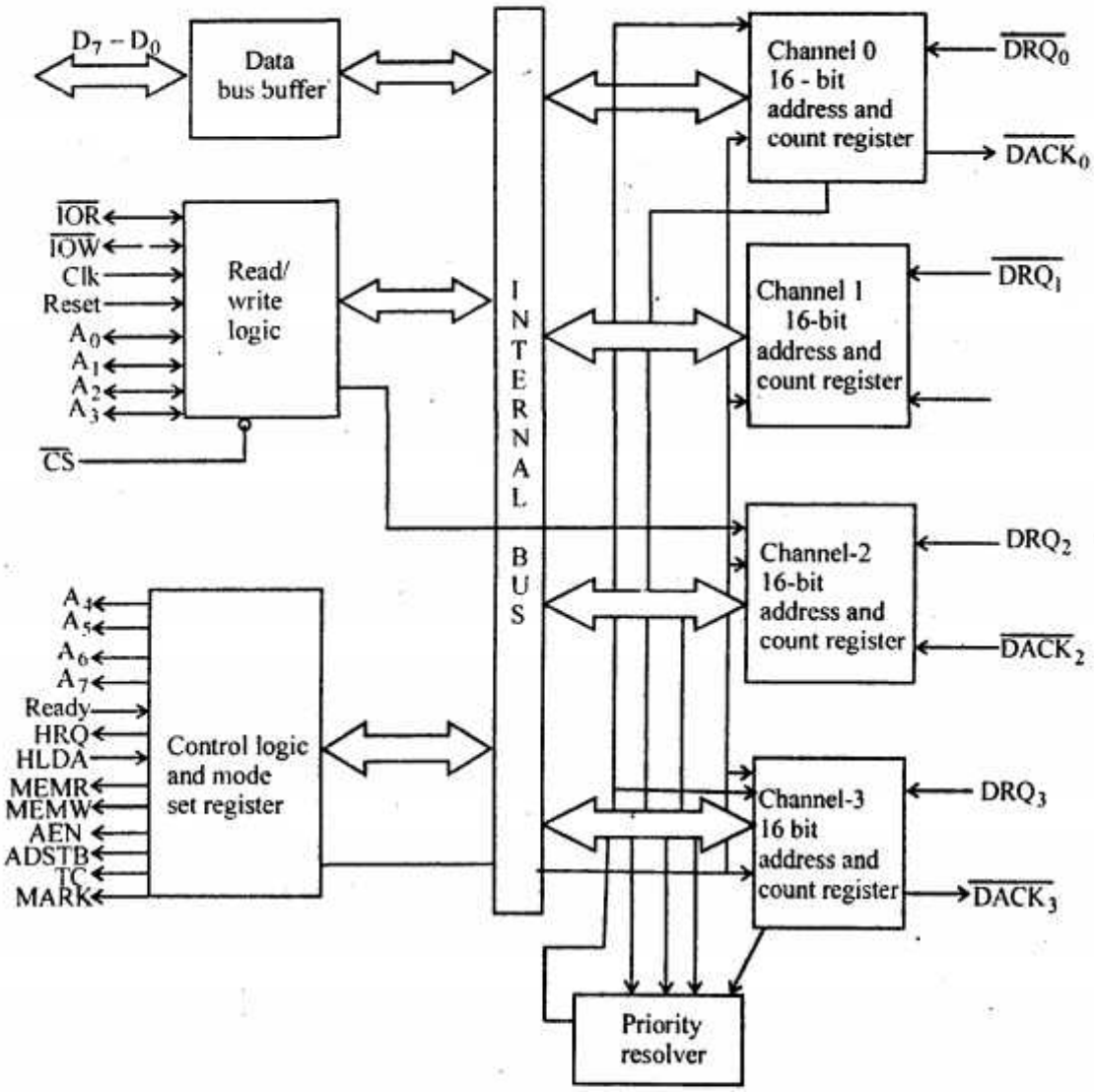
The eight possible combination of the set bits $B_3 - B_1$ in the bit-set reset format determine particular bit in $PC_0 - PC_7$ being set or reset as per the status of bit B_0. A BSR word is written for each bit that is to be set or reset. For example, if bit PC_4 is to be set and bit PC_5 is to be next, the appropriate words that will have to be loaded into the control register will be $0 \times X \times 1001$ and $0 \times X \times 1010$ respectively where X is don't care.

For Input/Output Mode : The controls word for both mode definition and Bit set–reset are loaded into the same control register, with bit B_7 is used for specifying whether the word loaded into the control register is a mode definition word or bit set reset word. If B_7 is high the word is taken as a mode definition word and if it is slow, it is takes as a bit set-reset word. The appropriate bits are set or reset depending on the type of operation desired and loaded into control register.



Q. 7. (a) Explain the working of 8237 DMA controller.

Ans. Block Diagram :



Each channel has two programmable 16-bit register. One register is used to program the starting address of memory location for DMA data transfer and another register is used to program a 14 bit count value and a 2-bit code for type DMA transfer. The address in the address register is automatically incrmented after every read/write/verify transfer. The formal of court register is shown below.

In read transfer the data is transferred from memory to input/output device. In write transfer the data is transferred from input/output device to memory. Verification operations generate the DMA address without generating the DMA memory and input/output control signals. A port from the address and count register of each channel, the 8257 has a mode set registers and status register. The mode set register is used to program

various features of 8257 and the status register can be read to know the terminal count status of the channels, the register of 8257 are selected for read/write operation during slave/programming mode by sending a 4-bit address to 8257 although $BA_0 - A_3$ lines.

While programming 16 bit register the low byte has to be send first and then high byte. Internally the loading of low byte and high byte into 16 bit register one taken can by a first/last flip-flop.

The mode set registers is used to program the following features :

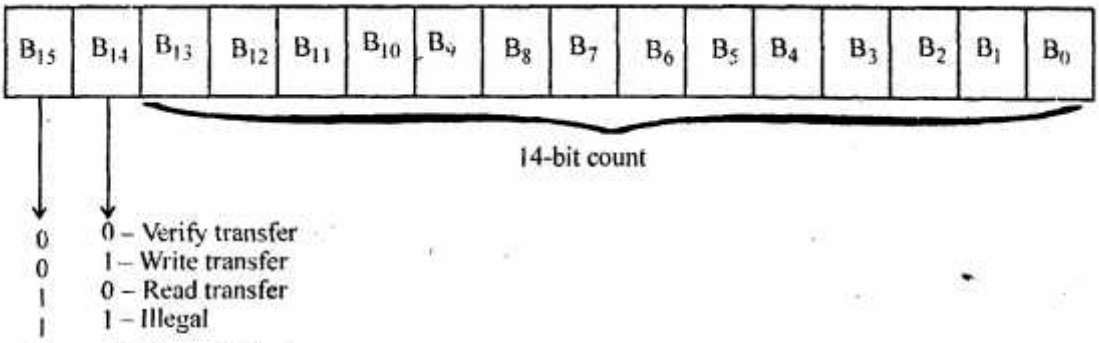
→ Enable/disable a channel

→ Fixed/rotating priority

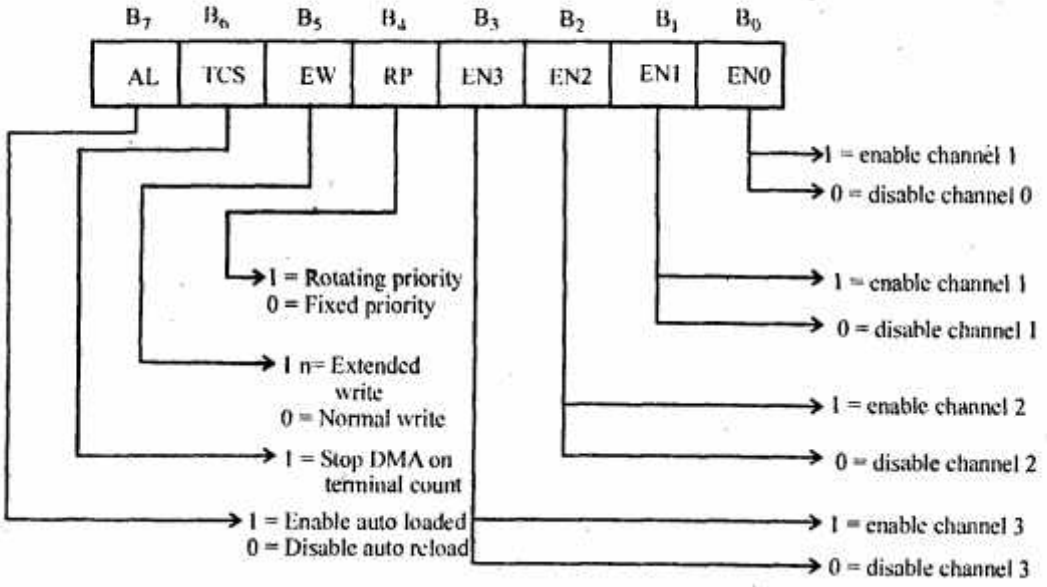
→ Stop DMA on terminal count

→ Extended/normal write time

→ Auto reading of channel-2

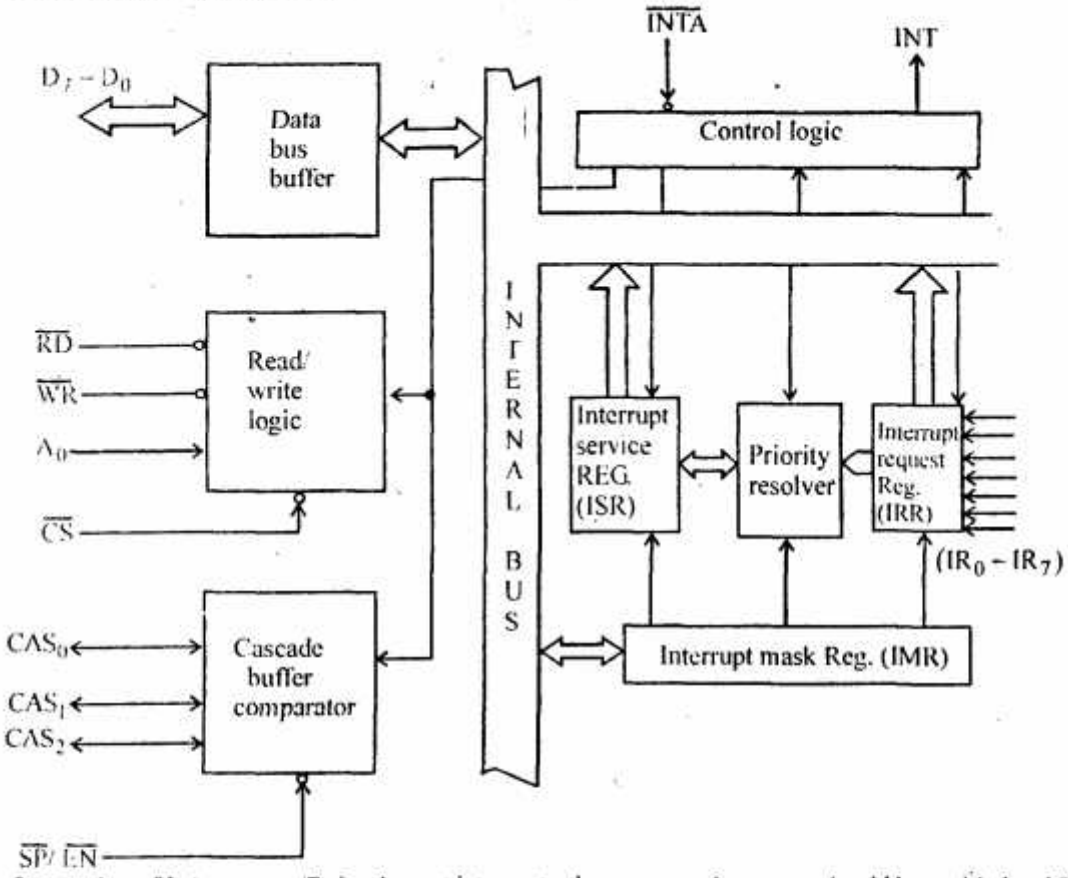


Format of Control Word :



Q. 7. (b) Make the block diagram of 8259 P/C and explain how it helps in managing multiple number of interrupts.

Ans. Block Diagram of 8259 :



Processing of Interrupts : To implement interrupts, the processor interrupts should be enabled and 8259 is initialized.

The 8259 is initialized by sending ICWs and OCWs. The ICWs are used to set up the proper conditions and to specify interrupt type number. The OCWs are used to perform function such as masking interrupts, setting up status, read operations etc. After the 8259 is initialized, the following sequence of events occur when one or more interrupt request lines go high.

(i) The IRR store request.

(ii) The priority resolver checks three register. The IRR is checked for interrupt request. The JMR is checked for marking bits and the ISR for the interrupt request being served. It resolves the priority and sets the INT high when appropriate.

(iii) The processor acknowledges the interrupt by sending to $\overline{INTA}$ signals one by one.

(iv) When the first $\overline{INTA}$ is received, the appropriate priority bit in the ISR is set to indicate which interrupt level is being served and the corresponding bit is the IRR is reset to indicate that the request is accepted.

(v) When 8259 receives the second $\overline{INTA}$, it places the type number on the data bus.

(vi) The processor multipliers the type number by four to generate a vector table address and from vector table address and from vector table the processes read the vector address of the interrupt type and load in IP and CS register. Then the processes start exciting ISR.

Q. 8. Write short notes on following :

(i) Memory mapped input/output vs/s input/output mapped input/output

(ii) 8254 programmable internal time.

Ans.

Memory Mapped Input/Output	Input/Output Mapped Input/Output
1. In this device address is 16-bit. Thus, A_0 to A_{15} lines are used to generate device address.	1. In this input/output device address is 8-bit. Thus, $A_0 - A_7$ or $A_8 - A_{15}$ lines are used to generate device address.
2. $\overline{MEMR}$ and $\overline{MEMW}$ control signals are used to control read and write input/output operations.	2. $\overline{IOR}$ and $\overline{IOW}$ control signals are used to control read and write operations.
3. Instruction available are LDA adder, STA adder LOAX up MOV, M, R, etc.	3. Instruction available are IN and OUT.
4. Data transfer to between any register and input/output device.	4. Data transfer between accumulators and input/output device.
5. Maximum number of input/output devices are 65536.	5. Maximum number of input/output devices are 256.
6. Decoding 16-bit address may require more hardware.	6. Decoding 8-bit address will require less hardware.

(ii) 8254 Programmable Internal Time :

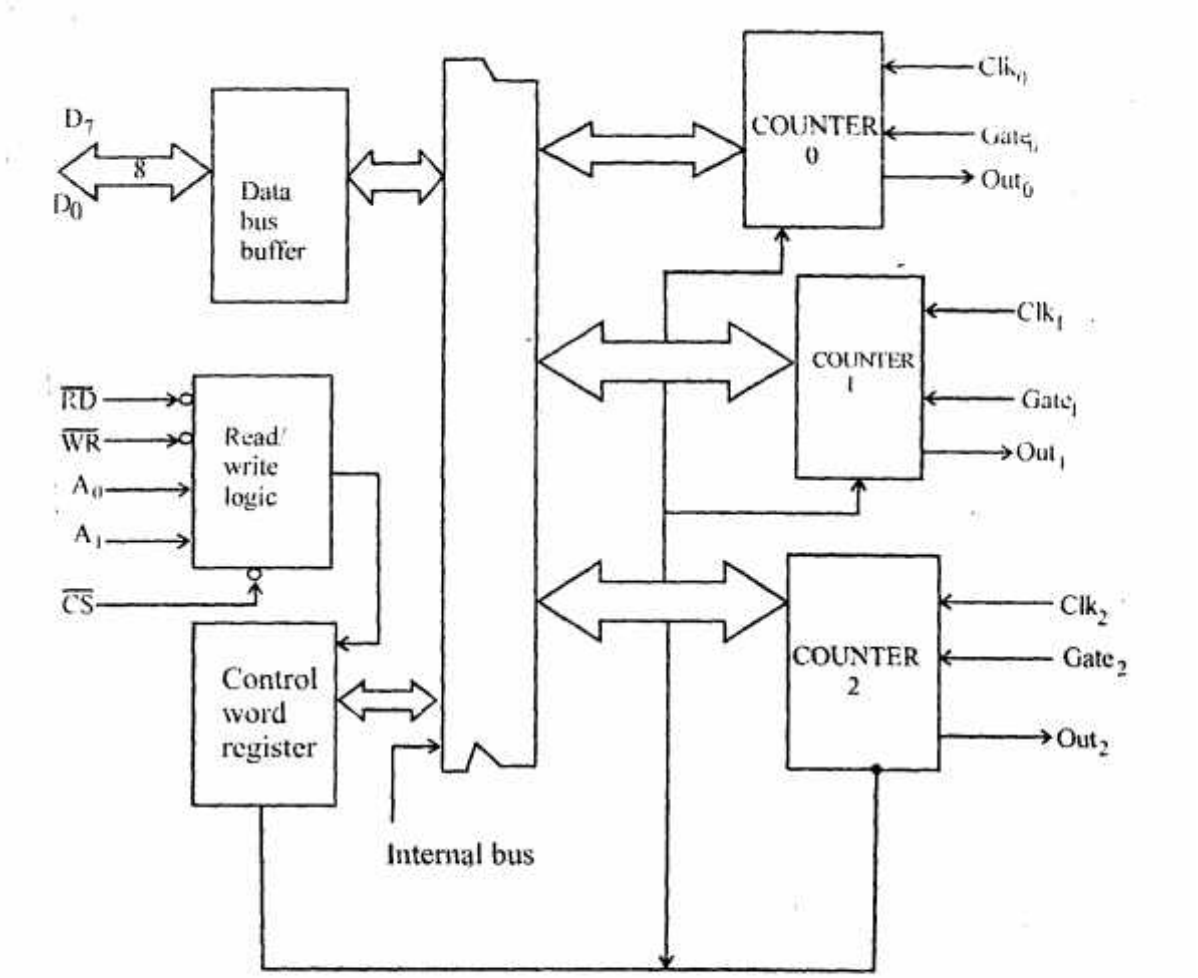
Features :

→ Operating frequency 0–10 MHz

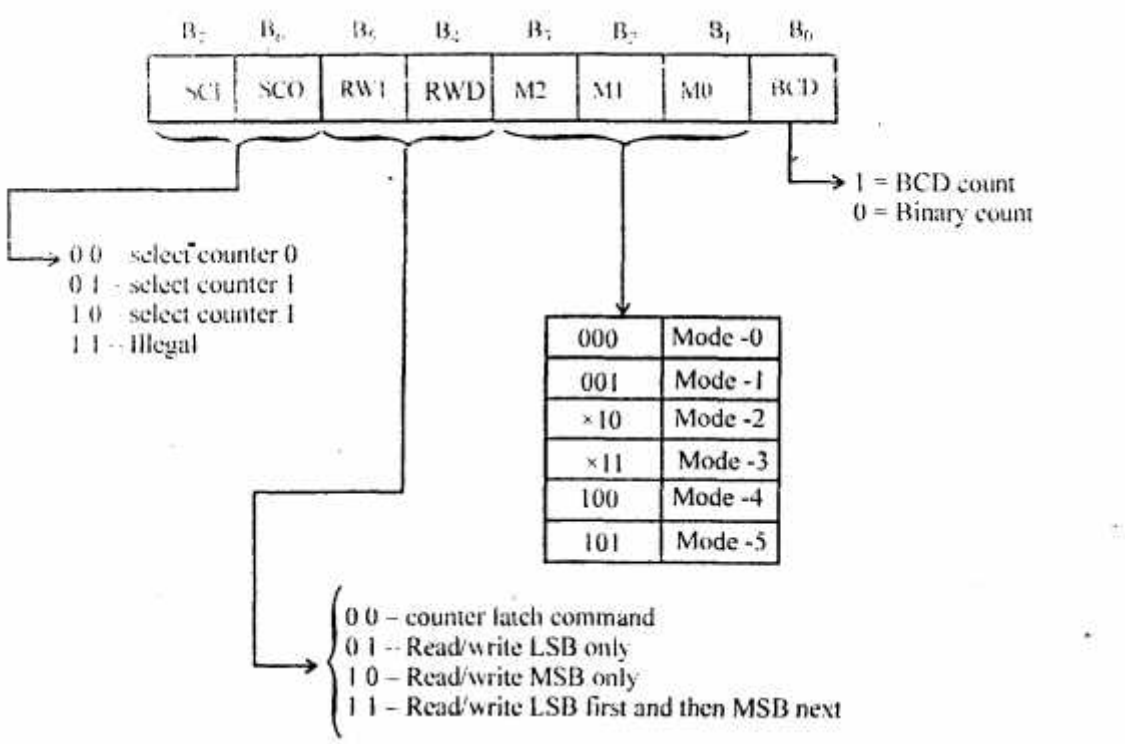
→ Uses H-MOS technology

→ Read-back command available

→ Read and write of the same counter can be interleaved can't be interleaved.

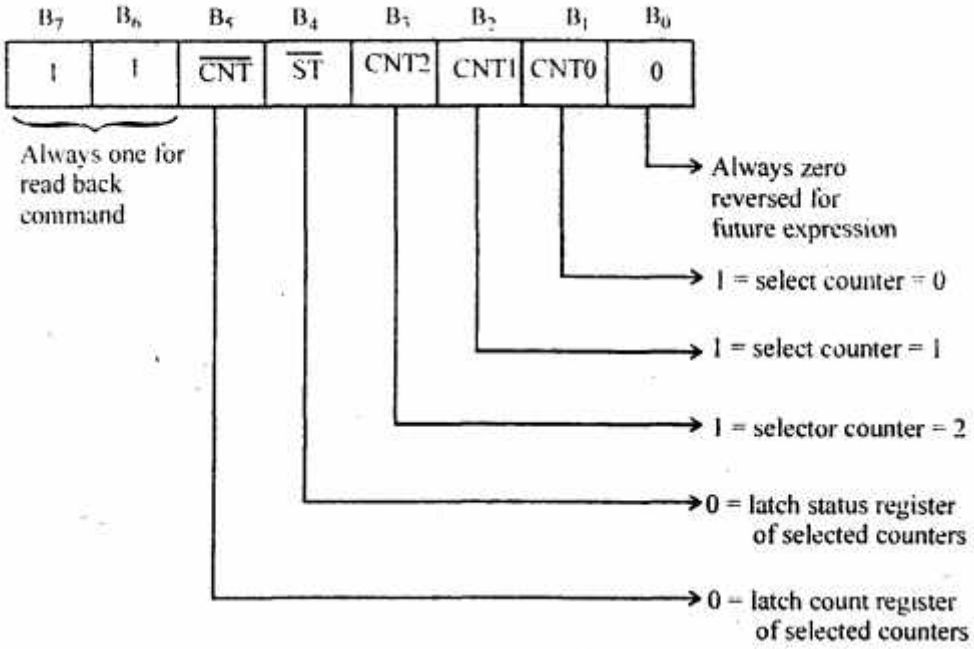


Programming 8254 : Each counter of 8254 can be individually programmed by writing a control word followed by the count value. The format of control word is shown below :



These bits are programmed for reading/writing the 16-bit count value is proper order. If the count value is read without stopping the counter, then the count value may change between reading the LSB and MSB. To avoid this the counter latch command can be used to latch the count value to an internal latch available at the output of each counter before read operations.

Alternatively, a separate react back control word is available for latching the count value in 8254. The format of read-back control word of 8254 is shown below. This control word has to be send to same control register address before read operations to latch the count values. The control register indentifiers this control word from the value of bits B_6 and B_7.



The read-back control word can be used to latch one or all the counter by sending a single control word. This control word is also used to latch the status register to the output latch of the counters, so that the status register can be read by using the respective counter address. At any one time we can latch either the count value by programming the B_5 as zero or latch the status registers by programming the bit B_4 as zero.

The format of status register of each counter is shown below. The status word can be read to check the programmed stations of the counter and also to verify whether the count value has reached terminal count i.e., zero or not.