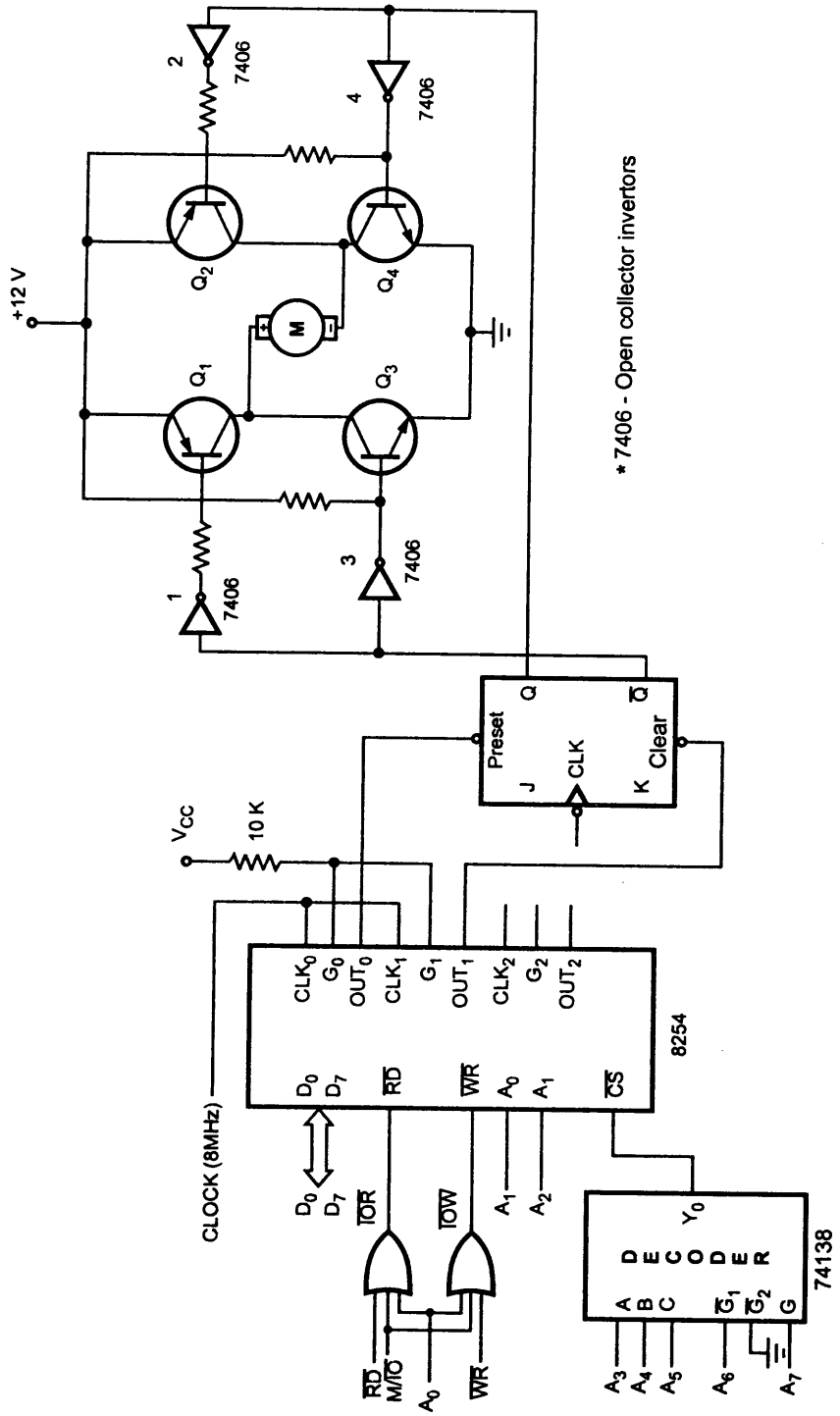
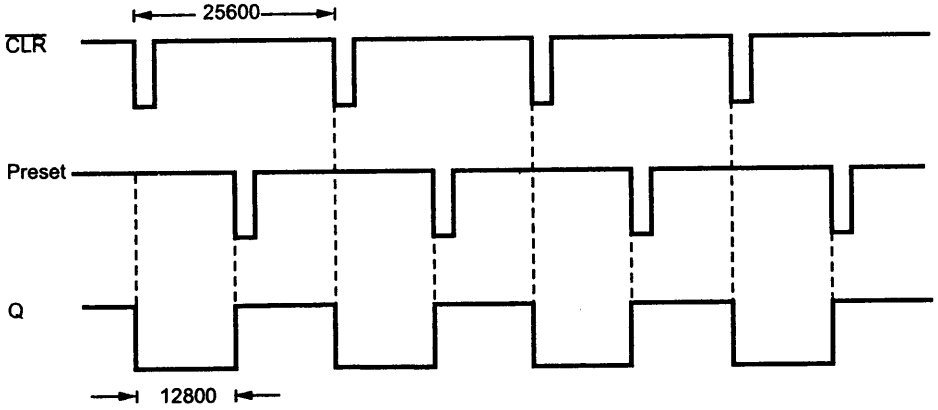
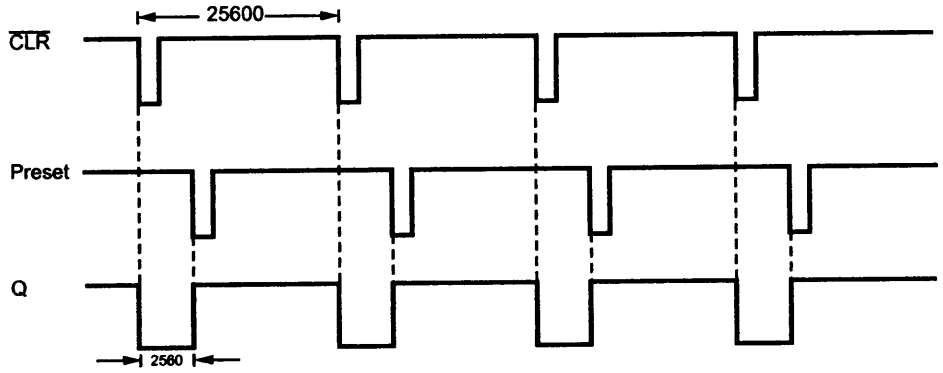


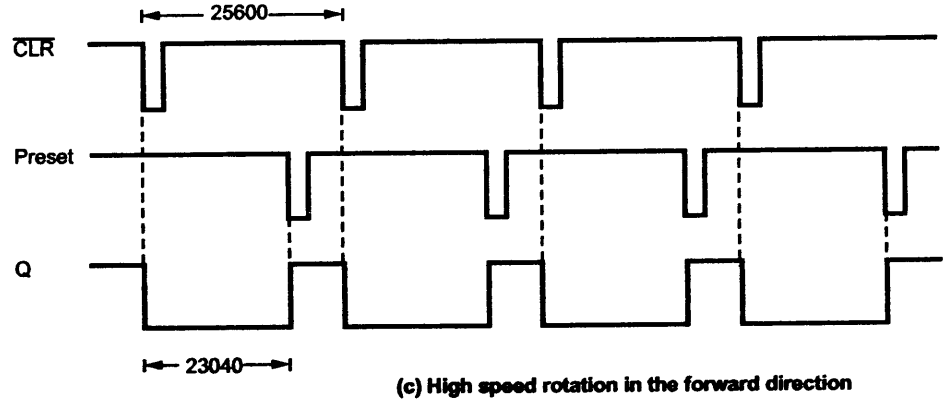
Fig. 3 DC motor speed and direction control using the 8254 timer



(a) No rotation



(b) High speed rotation in the reverse direction



(c) High speed rotation in the forward direction

Fig. 4

torque because of the current flowing through it. Fig. 4 shows some timing diagrams and their effects on the speed and direction of the motor.

The variable duty cycle is achieved by programming two counters, counter 0 and counter 1 of 8254. The counter 0 and counter 1 are both programmed to divide the input clock by 25,600. The duty cycle now can be changed by changing the point at which counter 0 is started in relationship to counter 1. The clock input is divided by each counter by 25,600. This produces the basic operating frequency for the motor and as 25,600 is divisible by 256 a short program given below allows 256 different speeds.

```
;  Procedure to control speed and direction of DC motor
;  Addresses for 8254 are :
;  Counter 0 : 80H
;  Counter 1 : 82H
;  Counter 2 : 84H
;  Counter 3 : 86H
;  Control word for counter 0 : 00110100 B,
;                   counter 1 : 01110100 B
   COUNT EQU 25600
      PUSH AX           ; save register
      PUSH BX
      MOV BL,100
      MUL BL            ; Multiply by 100
      MOV BX,AX
      MOV AX,COUNT      ; Subtract result of
      SUB AX,BX         ; multiplication from COUNT
      MOV BX,AX
      MOV AL,01110100B  ;  [ Load control word
      OUT 86H,AL        ;     for counter 1 ]
      MOV AX,COUNT      ;  [ Load counter 1
      OUT 82H,AL        ;     register with
      MOV AL,AH         ;     Count =
      OUT 82H,AL        ;     25600 ]
      MOV AL,11000010B  ;  [ Send the command word
      OUT 82H,AL        ;     to latch the count using
                        ;     Read back command ]
AGAIN:IN AL,82H         ;  [ Read the
      MOV AH,AL         ;  Count
      IN AL,82H         ;  of
      XCHG AL,AH        ;  counter 1 ]
      CMP AX,BX         ;  Compare it with calculated
                        ;  count
      JB AGAIN          ; If not equal read again
      MOV AL,00110100B  ;  [ Load control word for
      OUT 86H,AL        ;     Counter 1 ]
      MOV AX,COUNT      ;  [ Load counter 0
      OUT 80H,AL        ;     register with
      MOV AL,AH         ;     count =
      OUT 80H,AL        ;     25600 ]
```

```
        POP  BX              ·  Restore registers
        POP  AX
        RET
```

As shown in the program, value passed in AL register decides the direction and speed of the DC motor. If value in AL is 80H then motor does not rotate. However, when value in AL approaches 00H, the motor rotates with increase in speed in forward direction. On the other hand, when value in AL approaches FFH, the motor rotates with increase in speed in reverse direction.

The time interval between counter 1 and 0 is calculated interms of count. This is accomplished by multiplying value in AL by 100 and then subtracting it from 25600. This is required because the counters are down-counters that count from the programmed count to 0, before restarting.

Initially, counter 1 is started with count 25600. It is read and compared with the calculated cound until it is equal. Once it reaches to the calculated count, counter 0 is started with count 25600. From this point forward, both counters continue generating clear and preset pulses until the procedure is called again to adjust the speed and direction of the DC motor.

7. Program Statement : Interfacing Printer : Refer page no. 9-25.

8. Program Statement : Serial data transfer from PC to PC through COM port.

Program a) : PC to PC communication to send/receiver data between two PCs serial link is used. COM2 of one PC is connected to the COM2 of another PC.

Program to transmit one character

```
CODE  SEGMENT
ASSUME   CS : CODE, DS : CODE
ORG   0100H
START :    MOV    AH,00H     ; Initialize serial port COM2 with
           MOV    AL,03H     ; 8-data bits, 1 stop bit, parity
           MOV    DX,01H     ; none and baud rate 110 bps.
           INT    14H
           MOV    AH,08H     ; Read character from keyboard
           INT    21H

           MOV    AH,01H     ; Transmit character in AL to
           MOV    DX,01H     ; COM2 serial port
           INT    14H

           MOV    AH,4CH     ; Terminate program and
           INT    21H        ; return to DOS
CODE ENDS
END START
```

Program to receive one character

```
CODE SEGMENT
ASSUME CS : CODE , DS :
ORG 0100H
START :    MOV AH,00H    ; Initialize serial port COM2 with
           MOV AL,03H    ; 8-data bits, 1 stop bit, parity
           MOV DX,01H    ; none and baud rate 110 bps.
           INT 14H

AGAIN :    MOV AH,03H    ; Read the status of  COM2
           MOV DX,01H
           INT 14H

           AND AH,01H    ; check COM2, if it is ready
           CMP AH,01H    ; to receive data
           JNE AGAIN     ; If not check status again

           MOV AH,02H    ; Receive data from serial port
           MOV DX,01H    ; COM2
           INT 14H

           MOV DL,AL     ; Load data in DL
           MOV AH,02H    ; Display the received data
           INT 21H

           MOV AH,4CH    ; Terminate program and
           INT 21H       ; return to DOS
CODE ENDS
END START
```

Program b) : Program to transfer one file from one PC to another PC

Program to transmit file

```
CODE SEGMENT
ASSUME CS : CODE , DS : CODE
ORG 0100H
START :    JMP SKIP_DATA     ; INITIALIZE VARIOUS
FILE_NAME DB 'C: \MASM\file 1. dat',0
                             ; PARAMETERS
FILE_HANDLE DW (?)
BUFF DB (?)
SIZ DW (?)

SKIP_DATA :    MOV AH,3DH          ; OPEN FILE
               MOV AL,2
```

```
        MOV  DX,OFFSET  FILE_NAME
        INT  21H
        MOV  FILE_HANDLE,  AX

        MOV  AH,42H             ; Set the file pointer at
        MOV  AL,02              ; the end and use
        MOV  BX,FILE_HANDLE     ; end address to get
        MOV  CX,  0             ; the file size
        MOV  DX,  0
        INT  21H
        MOV  SIZ,AX

        MOV  AH,42H             ; SET file   POINTER AT
        MOV  AL,00              ; STARTING POSITION
        MOV  BX,FILE_HANDLE
        MOV  CX,0
        MOV  DX,0
        INT  21H

BACK :  MOV  AH,  3FH           ; Read file one chacacter
        MOV  BX,  FILE_HANDLE   ; at a time
        MOV  CX,  1
        MOV  DX,OFFSET  BUFF
        INT  21H

        MOV  SI,OFFSET  BUFF

        MOV  AH,00H             ; Initialize COM 1
        MOV  AL,03H
        MOV  DX,0
        INT  14H

        MOV  AH,01H             ; Transmit character read
        MOV  AL,[SI]            ; from file to COM1
        MOV  DX,0               ; Display the same
        INT  14H                ; Character on the monitor
        MOV  AH,02H
        MOV  DL,[SI]

        INT  21H

        DEC  WORD  PTR  SIZ     ; Decrement size pointer
        CMP  SIZ,0              ; Check if end of file
        JNZ  BACK
```

```
            MOV  AH,  4CH          ; Terminate program and
            INT  21  H             ; return to DOS
 CODE    ENDS
 END     START
Program to Receive file
CODE SEGMENT
ASSUME CS:CODE,  DS:CODE
ORG 0100H
START :     MOV  AH,00             ; INITIALIZE   COM1  PORT
            MOV  AL,  03
            MOV  DX,00
            INT  14H

AGAIN :     MOV  AH,03             ; Read status of COM1
            MOV  DX,00
            INT  14H

            AND  AH,01             ; Check COM1, if it is ready
                                   ; to receive
            CMP  AH,01             ; data
            JNE  AGAIN             ; if not check status again

NEXT :      MOV  AH,02             ; Receive data from serial
            MOV  DX,00             ; Port COM1
            INT  14H

            CMP  AL,1AH            ; Check for end of
            JE  STOP               ; file character if yes   stop

            MOV  DL,AL             ; Display the received
            MOV  AH,02             ; character
            INT  21H
            JMP  NEXT              ; Goto receive next character

STOP :      MOV  AH,4CH            ; TERMINATION
            INT  21  H
CODE ENDS
END START
```

□□□

Microprocessor

✎ **Chapterwise University Questions with Answer**

Jan./Feb.-2005
July/Aug.-2005
Jan./Feb.-2006
July/Aug.-2006
Jan./Feb.-2007
July/Aug.-2007
Jan./Feb.-2008
July/Aug.-2008

1 Introduction to Microprocessor Based Computer System

Q.1 *Trace the development of Intel 86 family of microprocessors briefly indicating the additional features introduced at each stage of development from 8086 to Pentium IV.* **(Jan./Feb.-2005, 10 Marks)**

Ans. : Refer section 1.1.5.

Q.2 *Give a general picture of the evolution of the Intel microprocessors from 4004 to pentium series, giving some idea of the time frame of introduction of the additional features in different processors.* **(Jan./Feb.-2006, 10 Marks)**

Ans. : Refer section 1.1.5.

Q.3 *Write an explanatory note on the development of Intel microprocessor from 4004 onwards. Give the important features of each of the processors.* **(July/Aug.-2007, 10 Marks)**

Ans. : Refer section 1.1.5.

□□□

2 8086 / 8088 CPU

Q.1 *Enumerate the user programmable registers of the 8086 processor. Describe their general and special uses.* **(Jan./Feb.-2005, 6 Marks)**

Ans. : Refer section 2.3.

Q.2 *State the segments used in calculating the physical memory address in the following instructions :*

i) MOV AX, [BP + 6] *ii) MOV AX, [DI + 6]* **(Jan./Feb.-2005, 4 Marks)**

Ans. : **i) MOV AX, [BP + 6]** : Stack segment. **ii) MOV AX, [DI + 6]** : Data segment.

Q.3 *Indicate the different registers available in 8086 and explain their operations.* **(July/Aug.-2005, 6 Marks)**

Ans. : Refer section 2.3.

Q.4 *Explain the advantages of dividing memory into segments.* **(July/Aug.-2005, 8 Marks)**

Ans. : Refer section 2.5.

Q.5 *What is meant by segment override prefix ? Explain with example.* **(July/Aug.-2005, 4 Marks)**

Ans. : Refer section 2.5.

Q.6 *What is the purpose of the segment registers in the 8086?* **(Jan./Feb.-2006, 6 Marks)**

Ans. : Refer section 2.3.

Q.7 *Indicate generally how the different segments of 8086 are used by default, like for example, code segment is used for instruction fetch with IP register.* **(Jan./Feb.-2006, 4 Marks)**

Ans. : Refer section 2.5.

Q.8 *Describe the i) features of 8086 ii) registers of 8086 and iii) differences between 8086 and 8088.* **(July/Aug.-2006, 12 Marks)**

Ans. : Refer sections 2.1, 2.3 and 2.6.

Q.9 *Briefly explain the advantages of the instruction queue in 8086.* **(July/Aug.-2006, 3 Marks)**

Ans. : Refer section 2.2.1.

Q.10 *Sketch and briefly explain the 8086.* **(July/Aug.-2006, 5 Marks)**

Ans. : Refer section 2.2.

Q.11 *Write the programming model of 8086 microprocessor and explain.* **(Jan./Feb.-2007, 8 Marks)**

Ans. : Refer section 2.3.

Q.12 *Explain with suitable example working of all segment and offset registers in 8086 μp.* **(Jan./Feb.-2007, 8 Marks)**

Ans. : Refer sections 2.3.2 and 2.3.3.

Q.13 *Explain the functions of the following registers in 8086 CPU :*

i) The segment registers ii) The instructions queue iii) The flag register. **(July/Aug.-2007, 10 Marks)**

Ans. : Refer sections 2.3.2, 2.2.1 and 2.3.4.

Q.14 *Write the programming model of Intel 8086. Explain their general and special uses.* **(Jan./Feb.-2008, 10 Marks)**

Ans. : Refer section 2.3.

Q.15 *Explain memory segmentation. What are advantages of using segment registers in memory segmentation?* **(Jan./Feb.-2008, 6 Marks)**

Ans. : Refer section 2.5.

Q.16 *How does the pipelining in 8086 increases its through put? Explain* **(Jan./Feb.-2008, 4 Marks)**

Ans. : Refer section 2.2.

Q.17 *Write the programming model of 8086 and explain the flag register format in detail.* **(July/Aug.-2008, 12 Marks)**

Ans. : Refer section 2.3.

Q.18 *Explain the advantages of segmentation.* **(July/Aug.-2008, 4 Marks)**

Ans. : Refer section 2.5.

❑❑❑

3 Instruction Set of 8086/8088 and Assembly Language Programming

Q.1 *Explain with examples, different addressing modes of the 8086 processor.*
(Jan./Feb.-2005, 10 Marks)

Ans. : Refer section 3.2.

Q.2 *Explain the (MOD-REG-R/M) byte of an 8086 instruction, with its interpretations.*
(Jan./Feb.-2005, 5 Marks)

Ans. : Refer section 3.20.

Q.3 *What do the following instructions do ?*

i) ROL *ii) RCL*

iii) SAR *iv) STD*

v) XCHG AX, [BX] **(Jan./Feb.-2005, 5 Marks)**

Ans. : Refer sections 3.5.8.2, 3.5.8.1, 3.9 and 3.4.

Q.4 *Explain the assembler directives PUBLIC and EXTRN with a simple example.*
(Jan./Feb.-2005, 6 Marks)

Ans. : Refer section 3.13.

Q.5 *Distinguish between macro and a procedure.* **(Jan./Feb.-2005, 4 Marks)**

Ans. : Refer section 3.19.

Q.6 *Write short note on ASCII arithmetic in the 86 family of microprocessors.*
(Jan./Feb.-2005, 6 Marks)

Ans. : Refer section 3.5.6.2.

Q.7 *Eventhough interrupt service routine is similar to any procedure routine, the last instruction of an interrupt routine is IRET which is coded differently from the RET instruction of the subroutine return. Explain the reasons for this separate IRET instruction.*
(Jan./Feb.-2005, 4 Marks)

Ans. : Refer section 3.11.

Q.8 *Write an 8086 ALP using DOS interrupts to read a two digit hexadecimal number and display the same on the console.* **(Jan./Feb.-2005, 10 Marks)**

Ans. : Refer sections 3.17.3 and 3.17.4

Q.9 *Identify the addressing modes of the instructions given below and justify the answer with clear explanation*

i) MOV WORD PTR [SI], 5

ii) MOV DS : [1000H], 10H

iii) MOV AX, Num [BX + DI] **(July/Aug.-2005, 6 Marks; July/Aug.-2007, 8 Marks)**

Ans. : **i) MOV WORD PTR [SI], 50** : This instruction uses register indirect addressing mode to specify destination location. Because the contents of SI register are used to generate effective address.

ii) MOV DS : [1000H], 10H : This instruction uses direct addressing mode. Because the effective address is specified in the instruction.

iii) MOV AX, Num [BX + DI] : This instruction uses base relative plus index addressing mode. Because base is provided by array name Num, BX register provides first index to array and the contents of DI acts as an offset.

Q.10 *Explain in detail, various fields of machine language instructions for 16-bit instruction format.* **(July/Aug.-2005, 8 Marks)**

Ans. : Refer section 3.20.

Q.11 *Using related string instructions, write an 8086 assembly language program to search for the given element in the location SER of byte type in an array of 10 bytes and to indicate the result by storing the number of occurrence of appearance of search element at the memory location identified by RES.* **(July/Aug.-2005, 8 Marks)**

Ans. : Refer program 4 on page 3-71.

Q.12 *What is meant by modular programming ? Explain the differences between a procedure and a macro.* **(July/Aug.-2005, 6 Marks)**

Ans. : Usually, programming task is divided into subtasks. Separate modules (program) are written for performing subtasks. Each module is tested separately. The common routines required in modules are written in separate module and they are 'called' from individual modules as per requirements. Programming done in this fashion is called **'Modular Programming'**. Refer Table 3.8.

Q.13 *Without using push and POP instructions, write a procedure with name, FLAGTEST in 8086 assembly language to check the condition of the auxiliary carry flag and to set parity, sign and zero flags to 1 if auxiliary carry flag is 1 else reset the flags without affecting all other flags.* **(July/Aug.-2005, 6 Marks)**

Ans. :

```
PROC   FLAGTEST  NEAR
       LAHF              ; Get the lower byte of flag in AH
       MOV AL,AH         ; Save flag contents
       AND AL,10H        ; Mask all flags except Aux.Carry flag
       JNZ SKIP
```

```
                AND AH,3BH     ; [Reset Sign, Parity and
                SAHF           ;  zero flags]
                JMP LAST
SKIP :          OR AH,0C4H     ; [Set sign, parity and
                SAHF           ;  zero flags]
LAST :          RET
FLAGTEST        ENDP
```

Q.14 *Write short notes on :*

a) Assembler directives. b) String instructions of 8086. **(July/Aug.-2005, 10 Marks)**

Ans. : Refer sections 3.13 and 3.6.

Q.15 *At a certain instant during the execution of a program the 8086 processor has the following data in the registers :* **(Jan./Feb.-2006, 15 Marks)**

AX = 1234H; BX = 5678H; SI= ABCDH, DI = CDEFH

CS = 3456H and IP = 789AH; DS = ES = 4567H

State the addressing modes used and work out the addresses of source and destination of data, when each of the following each of the following instructions are executed:

i) MOV AX, BX

ii) MOV [BX], AX

iii) MOV word ptr [BX + DI + 3456H], 9ABCH

iv) MOV AX, [9ABCH]

v) LODSW

Ans. : i) MOV AX, BX - Register addressing mode

This instruction will copy contents of BX register to AX register.

AX will contain 5678H.

ii) MOV [BX], AX - Register indirect addressing mode

This instruction will copy 16-bit contents from AX into a memory location offset by the value of EA specified in BX.

Contents of AX - 1234H is copied to memory locations 4ACE8H and 4ACE9H.

4	5	6	7	0	H	DS
+	5	6	7	8	H	BX
4	A	C	E	8	H	Physical address

iii) MOV word ptr [BX + DI + 3456H], 9ABCH - Relative based indexed addressing mode.

This instruction will copy contents 9ABCH to the calculated physical address as,

		1	1	1			
		5	6	7	8	H	BX
	+	C	D	E	F	H	DI
		3	4	5	6	H	16-bit displacement
Effective address	1	5	8	B	D	H	

		1				
4	5	6	7	0	H	DS
+ 1	5	8	B	D	H	
5	A	F	2	D	H	Physical address

iv) MOV AX, [9ABCH] - Direct addressing mode. This instruction will copy the contents of memory location, at a displacement of 9ABCH from data segment base into AX register. Effective address is 9ABCH, directly written in instruction.

4	5	6	7	0	H	DS
+	9	A	B	C	H	
4	F	1	2	C	H	Physical address

v) LODSW - String instruction

This instruction copies a word from a string location pointed to by SI to AX

SI will point to source string and copies word to AX.

Q.16 *i) State clearly the various assembler directives used and their meaning in the assembly language segment defined below :*

OUR_DATA SEGMENT

NUMB1 DW 1234H

NUMB2 DB 12H, 34H, 3DUP (0)

OUR_DATA ENDS

ii) Show the string of data making up the segment when the above segment part is assembled. **(Jan./Feb.-2006, 5 Marks)**

Ans. : i)

- SEGMENT - It is used to indicate start of logical segment. e.g. In above program OUR_DATA SEGMENT indicates the start of logical segment OUR_DATA.

- DW - It tells the assembler to define a variable of type word or to reserve storage locations of type word in memory. In above program NUMB1 DW 1234H, declares a variable of type word named NUMB1. Statement also tells to initialize NUMB1 with value 1234H.
- DB - It is used to declare a byte type variable or to set aside one or more storage locations of type byte in memory. e.g. NUMB2 DB 12H, declares a variable NUMB2 of type byte and initializes it with value 12H.
- DUP - The statement NUMB2 DB 12H, 34H, 3DUP(0) Reserves an storage of byte for NUMB2 with two values 12H, 34H and 3 numbers initialized to 0.
- ENDS - End segment is used with the name of a segment to indicate the end of logical segment. ENDS is used with SEGMENT directive to "bracket" a logical segment containing instruction or data. OUR_DATA ENDS - ends the logical segment OUR_DATA.

ii) When above segment part is assembled, there is a creation of OUR_DATA logical segment. In this segment variable NUMB1 is created of type word and with value 1234H. Second variable NUMB2 is created of type byte. i.e. NUMB2 having two values 12H, 34H and 3 numbers initialized to 0.

Q.17 *State if the following statements or statement pairs are meaningful and valid in an 8086 assembly language program. If valid indicate what action they would perform, else give reasons as to why they are invalid.* **(Jan./Feb.-2006, 10 Marks)**

i) ADD AX, 3 *ii) ADD AX, BL*

iii) SUB BL, AL *iv) PUSH AX*

DAA *POP BX*

v) ADD AX_1 [SI]

INC SI

Ans. : i) ADD AX, 3 - Valid

It will add value 3 to the contents of AX register and result of addition will get stored in AX.

ii) ADD AX, BL - Invalid

For adding contents the source and destination must be of same type i.e. either byte or word. Here BL is 8-bit register and AX is 16-bit register. Therefore invalid.

iii) SUB BL, AL

DAA **Invalid**

It will subtract contents of AL from contents of BL and result will get stored in BL. After that it will check result of BL, whether it is in correct BCD format i.e. the upper and lower nibble of BL are less than or equal to 9, using DAA instruction.

But DAA instruction works after an addition. Here subtraction is performed hence even though instructions are valid statement pair is not meaningful.

iv) PUSH AX

POP BX - valid

PUSH AX - will decrement SP by 2 and copy AX to stack.

POP BX - will copy word from top of stack to BX and increment SP by 2.

These two instructions copy the contents of AX register to BX register

v) ADD AX, [SI] - valid

INC SI

ADD AX, [SI] - ADD word from memory at OFFSET [SI] in DS to contents of AX.
INC SI - Increment location of SI by one where initially SI is pointing.

Q.18 *Distinguish between macros and procedures.*
(Jan./Feb.-2006, 5 Marks; Jan./Feb.-2007, 6 Marks; July/Aug.-2007, 4 Marks; Jan./Feb.-2008, 4 Marks; July/Aug.-2008, 5 Marks)

Ans. : Refer table 3.8.

Q.19 *Briefly explain the assembler directives EXTRN and PUBLIC, indicating their use.*
(Jan./Feb.-2006, 5 Marks)

Ans. : Refer section 3.13.

Q.20 *Write an ALP that uses the procedure to read a 4 digit hexadecimal number from keyboard using DOS interrupt and stores it in memory location HEX. Program should take care of i) Both lower and upper case alphabets used with Hexadecimal numbers. ii) Block all other characters.* **(Jan./Feb.-2007, 10 Marks)**

Ans. : *Refer section 3.17.3.*

Q.21 *Write an 8086 ALP using DOS function calls, to read a single digit Hex number from the keyboard. Any non-hex key input should be ignored.* **(Jan./Feb.-2006, 10 Marks)**

Ans. : Refer section 3.17.3.

Q.22 *What is the function of LOCK prefix in 8086 instructions ?* **(July/Aug.-2006, 2 Marks)**

Ans. : Refer section 3.10.

Q.23 *Describe the program memory and stack memory addressing modes of 8086.*
(July/Aug.-2006, 10 Marks)

Ans. : Refer sections 3.2.2 and 3.2.3.

Q.24 *Indicate the addressing modes of the destination operand and calculate its real mode address for the following instructions*

i) ADD [DI], AL *ii) SUB BLOCK, AX*

iii) AND [BX + DI], CL *iv) ADDC [BP+100H], AX*

Assume CS = 1000H, DS = 2000H, SS = 4000H,

BX = 3FFFH, DI = 5A00H, SP = 64A0H,

IP = A000H, BP = 69B0H, BLOCK = 4000H. **(July/Aug.-2006, 8 Marks)**

Ans. : i) ADD [DI] , AL - Register indirect

Real mode address = DI + DS =	2	0	0	0	0	H	(20 bit)
	+	5	A	0	0	H	effective address
	2	5	A	0	0	H	

ii) SUB BLOCK, AX - Immediate addressing

Substract contents of AX from BLOCK contents and store result in BLOCK.

Real mode address = BLOCK + DS =	2	0	0	0	0	H
	+	4	0	0	0	H
	2	4	0	0	0	H

iii) AND [BX + DI], CL - Based indexed

Real mode address = DS + BX + DI

	2	0	0	0	0	H		3	F	F	F	H
+		9	9	F	F	H	+	5	A	0	0	H
Real mode address =	2	9	9	F	F	H		9	9	F	F	H	effective address

iv) ADDC [BP + 100H], AX - Register relative

Real mode address = SS + BP + Displacement

4	0	0	0	0	H
+	6	9	B	0	H
		1	0	0	H
4	6	A	B	0	H

Q.25 *Is the instruction XCHG [SI] [DI] correct ? If not what is wrong in it ?*
(July/Aug.-2006, 2 Marks)

Ans. : XCHG [SI] [DI] : XCHG instruction is used to exchange contents of source and destination. In above instruction both the operands source and destination are memory locations and it is not possible to directly exchange the contents of two memory locations. Hence above instruction is wrong.

Q.26 *Explain the general format of 8086 instruction depicting the details of each field.*
(July/Aug.-2006, 8 Marks)

Ans. : Refer section 3.20.

Q.27 *What is the function of assembler directives ? Briefly explain any four such directives.*
(July/Aug.-2006, 6 Marks)

Ans. : Refer section 3.13.

Q.28 *Assume AL = B4H, BL = 11H. What is the result of executing the instructions MUL BL and IMUL BL? Also indicate the status of CF and OF in each case.*
(July/Aug.-2006, 4 Marks)

Ans. : AL = B4H, BL = 11H

MUL BL : This is byte to byte multiplication. It will perform multiplication of AL contents with BL contents and result will present in AX register.

Hence AX = 0BF4

CF = 1 and OF = 1 because most significant byte of result contains part of result.

IMUL BL : This instruction perform multiplication of signed byte in AL with signed byte in BL and result will get stored in AX.

Hence AX = FAF4

CF = 1 and OF = 1 same as above.

Q.29 *What initialization is necessary to use string instructions ? Explain the five string operations of 8086. What are its advantages ?* **(July/Aug.-2006, 10 Marks)**

Ans. : Refer section 3.6.

Q.30 *Differentiate between the instructions*

MOV AX, BLOCK and MOV AX, OFFSET BLOCK. **(July/Aug.-2006, 2 Marks)**

Ans. : 1) MOV AX, BLOCK

This instruction will load AX register with the contents of BLOCK variable.

i.e. if BLOCK variable contains 1234H then AX will contain 1234H.

2) MOV AX, OFFSET BLOCK

This instruction will determine offset of variable BLOCK from the start of segment in which BLOCK is defined and code this displacement in as part of instruction. When instruction executes, this computed displacement will be loaded in AX.

Q.31 *What is the advantages of using a macro over a procedure ? Write a macro that exchanges the contents of two word locations whose addresses are passed to it as parameters.* **(July/Aug.-2006, 2 Marks)**

Ans. : Refer section 3.19.

Macro example -

```
data segment
        P1 DW 1234H
        P2 DW 5678H
data ends
        SIM macro S1, S2
            MOV SI, S1
            MOV DI, S2
            MOV CX, [SI]
            MOV AX, [DI]
            MOV [SI], AX
            MOV [DI], CX
        ENDM
Code segment
assume CS:code, ds:data
start : MOV AX, data
            MOV DS,AX
            LEA DX,P1
            LEA BP,P2
            SIM DX,BP
        Code ends
        end start
```

Q.32 *What is wrong with following instructions :*

i) POP CS *ii) MOV [AX], 20H*

iii) MOV SS, DS *iv) MOV BL, SI* **(Jan./Feb.-2007, 4 Marks)**

Ans. : **i) POP CS :** It is an illegal instruction

ii) MOV [AX], 20H : We cannot use AX register for indirect addressing, we have to use BX register. It is also necessary to specify type override. In this case it is a byte ptr.

iii) MOV SS, DS : In the MOV instruction, both source and destinations registers cannot be segment registers.

iv) MOV BL, SI : The source and destination in the MOV instruction must both be of type byte, or they must be of type word. Here, source is of type word and destination is of type byte. Hence the instruction is wrong.

Q.33 *Explain with example following addressing modes in 8086.*

i) Register addressing *ii) Base plus index addressing*

iii) Relative program memory addressing iv) Stack memory addressing.

(Jan./Feb-2007, 10 Marks)

Ans. : Refer section 3.2.

Q.34 *Find the physical 20-bit address of source operand for (i) to (iii) and the result after execution of (iv) and (v) of following instructions. Given DS = 0200H ES = 0400H SS = 0300H AX = 050AH BX = 0500H SP = 0100H BP = 0050H SI = 0250H DI = 0150H DF = 0*

i) MOV AL, [1234H] *ii) MOV DI, [BX + 100H]*

iii) MOV BH, [BP + SI + 200H] *iv) STOSW*

v) PUSH BX

(Jan./Feb-2007, 10 Marks)

Ans. : i) MOV AL, [1234H]

Physical address =	0	2	0	0	[0]	H	Shifted contents of DS
+		1	2	3	4	H	offset
	0	3	2	3	4	H	

ii) MOV DI, [BX + 100H]

Physical address =	0	2	0	0	[0]	H	Shifted contents of DS
+		0	5	0	0	H	contents of BX
		0	1	0	0	H	displacement
	0	2	6	0	0	H	

iii) MOV BH, [BP + SI + 200H]

Physical address =	0	3	0	0	[0]	H	Shifted contents of SS
+		0	0	5	0	H	contents of BP
		0	2	5	0	H	contents of SI
		0	2	0	0	H	displacement
	0	3	4	A	0	H	

iv) STOSW : This instruction copies a word from AX to a memory location in the extra segment pointed by DI. Since DF is 0, and it is a word string DI will automatically be incremented by 2 after the copy operation.

∴ Physical address =	0	4	0	0	0	H	Shifted contents of ES
+		0	1	5	0	H	contents DI
	0	4	1	5	0	H	

Therefore, the contents of AL are copied at 04150H memory location and the contents of AH are copied at 04151H memory location. After copy contents of DI will be 0152H. Flags are not affected by this instruction.

v) PUSH BX : This instruction decrements the stack pointer by 2 and copies a word form BX register to the location in the stack segment where the stack pointer then points.

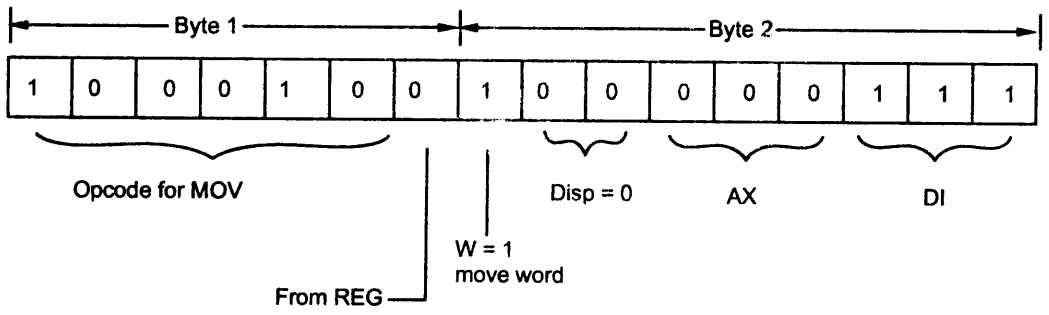
SP ← SP – 2 = 0 1 0 0 H – 2 = 0 0 F E H

Physical address =	0	3	0	0	0	H	Shifted contents of ES
+	0	0	0	F	E	H	contents SP
	0	3	0	F	E	H	

Therefore, the contents of BL are copied of 030FEH memory location and the contents of BH are copied at 030FDH memory location. Flags are not affected by this instruction.

Q.35 *Given the opcode 8907H, explain how these two bytes are interpreted in machine language. What is the resulting instruction?* **(Jan./Feb.-2007, 7 Marks)**

Ans. :



Instruction :

MOV AX, DI

Q.36 *Given the product after multiplication in AXreg as 0058H.*

i) Get its BCD equivalent using single instruction.

ii) ASCII equivalent of above result using a logic instruction.

iii) Double the above result using rotate instruction. **(Jan./Feb.-2007, 5 Marks)**

Ans. : i) AAM

ii) OR AX, 3030 H

iii) SAL AX, 1

Q.37 *Use appropriate logic instructions that do following.*

i) Set (1) right most four bits of AX.

ii) Clear (0) left most three bits of AX.

iii) Invert 7, 8 and 9^{th} bit of AX.

iv) Test whether right most bit is set.

v) Clear the register AX. **(Jan./Feb.-2007, 5 Marks)**

Ans. : i) OR AX, F000H

ii) AND AX, 1FFFH

iii) XOR AX, 0380H

iv) TEST AX, 8000H

v) XOR AX, AX

Q.38 *Explain how JMP [DI] and JMP FAR PTR [DI] instructions work.* **(Jan./Feb.-2007, 4 Marks)**

Ans. : Refer section 3.7.2.

Q.39 *Explain i) LOOP instruction ii) PUBLIC AND EXTRN directive* **(Jan./Feb.-2007, 6 Marks)**

Ans. : Refer sections 3.8 and 3.13.

Q.40 *Write a display macro using for statement that is used to display 'VTU' on the screen.* **(Jan./Feb.-2007, 5 Marks)**

Ans. : Refer section 3.19.

Q.41 *Explain the working of interrupt flag (IF) and trap flag (TF) in 8086. Write a procedure that sets TF to enable trap and a procedure to clear TF to disable trap.* **(Jan./Feb.-2007, 8 Marks)**

Ans. : Refer section 3.4 and Q.5 in Chapter 5.

Q.42 *Give the salient features of string manipulation instructions in 8086. Explain with suitable examples.* **(July/Aug.-2007, 8 Marks)**

Ans. : Refer section 3.6.

Q.43 *Give any four instruction formats of 8086 microprocessor.* **(July/Aug.-2007, 4 Marks)**

Ans. : Refer section 3.20.

Q.44 *What are assembler directives? Explain the significance of the following assembler directives:*

i) Assume ii) Number db 12H iii) EXTRN iv) PUBLIC. **((July/Aug.-2007, 8 Marks)**

Ans. : Refer section 3.13.

Q.45 *Write a note on the interrupt instructions in 8086.* **(July/Aug.-2007, 4 Marks)**

Ans. : Refer section 3.11.

Q.46 *Write note on Machine control instruction.* **(July/Aug.-2007, 5 Marks)**

Ans. : Refer section 3.9.

Q.47 *Describe the 7 data related addressing modes and program memory addressing modes of 8086 with an example each.* **(Jan./Feb.-2008, 12 Marks)**

Ans. : Refer section 3.2.

Q.48 *Given that*

BX = 63A9H, SI = 1C57H, DS = 4500H, displacement = 2B86H

Determine the effective and physical address resulting from these registers and the addressing modes.

i) Register indirect using BX, iii) Based indexed

ii) Register relative using BX, iv) Based indexed relative. **(Jan./Feb.-2008, 8 Marks)**

Ans. : i) Register indirect using BX

Effective address = 63A9H

$\therefore$ Physical address =

	4	5	0	0	[0]	H	Shifted contents of ES
+		6	3	A	9	H	contents BX
	4	B	3	A	9	H	

ii) Register relative using BX

Effective address = BX + Displacement

	6	3	A	9	H	BX
+	2	B	8	6	H	Displacement
	8	F	2	F	H	

∴ Physical address =	4	5	0	0	0	H	Shifted contents of DS
+		8	F	2	F	H	Effective address
	4	D	F	2	F	H	

iii) Based indexed

Effective address = BX + SI

	6	3	A	9	H	BX
+	1	C	5	7	H	SI
	8	0	0	0	H	

∴ Physical address =	4	5	0	0	0	H	Shifted contents of DS
+		8	0	0	0	H	Effective address
	4	D	0	0	0	H	

iv) Based indexed relative

Effective address = BX + SI + Displacement

	6	3	A	9	H	BX
+	1	C	5	7	H	SI
	2	B	8	6	H	Displacement
	A	B	8	6	H	

∴ Physical address =	4	5	0	0	0	H	Shifted contents of DS
+		A	B	8	6	H	Effective address
	4	F	B	8	6	H	

Q.49 *What are assembler directives? Explain the following assembler directives.*
i) EQU, ii) ORG, iii) EVEN iv) DB. **(Jan./Feb.-2008, 10 Marks)**

Ans. : Refer section 3.13.

Q.50 *Explain the working of the following instructions.*

i) CALL WORDPTR [BX], ii) CALL DWORDPTR [BX].

(Jan./Feb.-2008, 6 Marks)

Ans. :

i) **CALL WORDPTR [BX] :** This instruction is an indirect CALL within segment near or intrasegment CALL. It replaces contents of IP with contents of word memory location in DS pointed to by BX.

ii) **CALL DWORDPTR [BX] :** This instruction is an indiret call to another segment - far or intersegment CALL. New values for CS and IP are fetched four memory locations in DS. The new value for CS is fetched from [BX] and [BX + 1]; the new IP is fetched from [BX +2] and [BX + 3].

Q.51 *Generate the opcode for the instruction MOV AX, BX.* **(July/Aug.-2008, 4 Marks)**

Ans. : Instruction MOV AX, BX

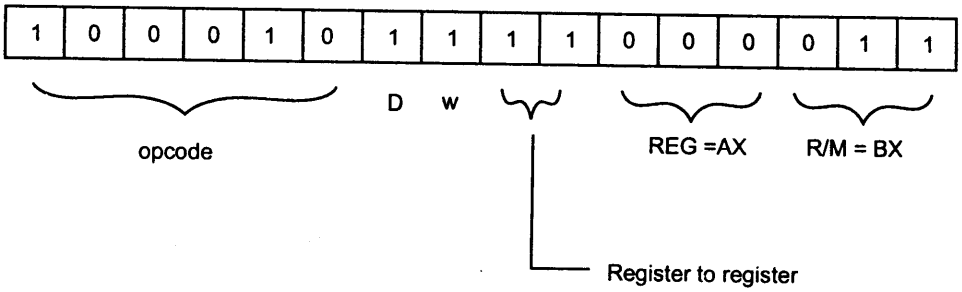
opcode = 100010

W = 1 Since we are moving word

D = 1 Moving word to AX Destination

REG = 000 AX register

R/M = 011 BX register



∴ opcode = 8BC3H

Q.52 *Given that*

DS = D470H	CS = 2123H	SS = 2091H
ES = 1ABCH	BP = 0030H	SP = 0123H
ST = 0A12H	DI 1234H	BX = 3F00H

Determine the physical address resulting from the following instructions.

i) MOV, DL, [BP + SI] ii) MOV AX, [BX] [DI – 04H] iii) MOV [BX], CX iv) MOV CL, [8734H]

(July/Aug. -2008, 8 Marks)

Ans. : i) MOV, DL, [BP + SI]

Physical address =		2	0	9	1	[0]	H	Shifted contents of SS
	+		0	0	3	0	H	Contents of BP
	+		0	A	1	2	H	Contents of SI
		2	1	3	5	2	H	

ii) MOV AX, [BX] [DI – 04H]

Physical address =		D	4	7	1	[0]	H	Shifted contents of DS
	+	0	3	F	0	0	H	Contents of BX
	+	0	1	2	3	0	H	DI-04 H
		D	C	8	3	0	H	

iii) MOV [BX], CX

Physical address =		D	4	7	0	[0]	H	Shifted contents of DS
	+	0	3	F	0	0	H	Contents of BX
		D	8	6	0	0	H	

iv) MOV CL, [8734H]

Physical address =		D	4	7	0	[0]	H	Shifted contents of DS
	+		8	7	3	4	H	Offset
		D	C	E	3	4	H	

Q.53 *Explain the following addressing modes with example. i) Direct addressing ii) Register relative addressing iii) Port addressing iv) Relative program memory addressing.* **(July/Aug.-2008, 8 Marks)**

Ans. : Refer section 3.2.

Q.54 *Identify the addressing modes of the following instructions.* **(July/Aug.-2008, 4 Marks)**

i) MOV CL, LIST [SI + 2] ii) PUSH AX iii) MOV DX, [DI] iv) JMP BX.

Ans. : i) MOV CL, LIST [SI + 2] : Register relative addressing mode

ii) PUSH AX : Stack memory addressing mode

iii) MOV DX, [DI] : Register indirect addressing mode

iv) JMP BX : Indirect program memory addressing mode.

Q.55 *Explain the following instructions with examples.*

i) LDS ii) SAHF iii) CWD iv) SAL v) JBE. **(July/Aug.-2008, 10 Marks)**

Ans. : Refer sections 3.4.3, 3.4.5, 3.12, 3.5.8.1 and 3.7.2.

Q.56 *Explain the difference between the following 2 instructions.*

i) LEA BX, ARRAY ii) MOV BX, offset ARRAY. **(July/Aug.-2008, 4 Marks)**

Ans. : These two instructions are equivalent. However, MOV BX, offset ARRAY instruction generates slightly shorter machine code than LEA BX, ARRAY instruction.

Q.57 *Explain the following directives i) PUBLIC ii) EXTRN.* **(July/Aug.-2008, 4 Marks)**

Ans. : Refer section 3.13.

Q.58 *Write an assembly language program to convert the BCD No. in AL to 7-segment code, using lookup table concept.* **(July/Aug.-2008, 6 Marks)**

Ans. : Refer section 3.17.5.

Q.59 *Write short notes on :*

a. Modular programming b. String instructions in 8086.

Ans. : Refer Q. 10 and section 3.6.

❑❑❑

4 BIOS and DOS Interrupts

Q.1 *Write a note on DOS and BIOS interrupt.* **(July/Aug.-2007, 5 Marks)**

Ans. : Refer sections 4.1 and 4.2.

Q.2 *Write the DOS function CALL NOS. to achieve the following :*

i) Reading a key with an echo ii) Reading a key without an echo iii) Reading an entire line with echo iv) To display and ASCII character v) To display a string of characters. **(July/Aug.-2008, 5 Marks)**

Ans. : Refer section 4.5.

Q.27 *Write an ALP to display a string on console using DOS interrupts.* **(July/Aug.-2008, 4 Marks)**

Ans. : Refer section 4.5.2.

❑❑❑

5 Assembly Language Programs

Q.1 *An unsorted 16-bit integer word array starts at address ARRAY-ST and has in all n words. Write an 8086 ALP to generate two different arrays from it - one of positive numbers, starting at memory address ARRAY_POS and the other of negative numbers starting at address ARRAY_NEG. Use string instructions as far as possible.*

(Jan./Feb.-2005, 10 Marks)

Ans. : The specified 8086 ALP routine is as follows :

```
; Assume :    Both DS and ES are initialized at same data segment
              MOV CX, n              ; Load word count in CX
              LEA BP, ARRAY_ST       ; Initialize array base pointer
              LEA DI, ARRAY_POS      ; Initialize positive pointer
              LEA SI, ARRAY_NEG      ; initialize negative pointer
    BACK :    MOV AX, DS:[BP]        ; Get the word
              MOV BX, AX             ; Save word
              AND AX, 8000H          ; Mask all bits except MSB
              JZ NEXT                ; If MSB = 0 go to next
              MOV [SI], BX           ; [otherwise save number in
                                     ;  ARRAY_NEG]
              INC SI                 ; [Increment
              INC SI                 ;  pointer]
              JMP SKIP               ; go to SKIP
    NEXT :    MOV [DI], BX           ; Save number in ARRAY_POS
              INC DI                 ; [Increment
              INC DI                 ;  Pointer]
    SKIP :    INC BP                 ; [Increment base
              INC BP                 ;  Pointer]
              LOOP BACK              ; loop though sequence of
                                     ; instructions until CX = 0
              RET
```

Q.2 *Write an 8086 ALP to multiply a 2-digit BCD number by a single digit BCD number by repeated addition, using DAA instruction.* **(Jan./Feb.-2005, 5 Marks)**

Ans. :

```
.mode small
. stack 100
.data
    NO1       DB   32 H      ; First number
    NO2       DB   24H       ; Second number
    Result    DW   ?         ; Result of multiplication
.Code
 Start : MOV AX, @ data      ; [Initialize
```

```
        MOV DS, AX          ; data segment]
        MOV CH, 00          ; [Initialize counter
        MOV CL, NO2         ; for add operations]
        MOV DX, 0000H       ; Result = 0
        MOV BL, NO1         ; Get first number
BACK :  MOV AX, DX          ; Get the result
        ADD AL, BL          ; [Add BCD
        DAA                 ;  number]
        MOV DL, AL          ; Save lower byte of result

        JNC SKIP
        MOV AL, DH          ; [if carry add 01 in higher
        ADD AL, 01          ;  byte]
        DAA
        MOV DH, AL          ; save higher byte of result
SKIP :  LOOP BACK           ; Repeat addition until CX = 0
        MOV Result, DX      ; save result
        END start
```

Q.3 *Convert a 2-digit BCD number in AL, to a 2-digit Hex number in AL, using the AAD instruction.* **(Jan./Feb.-2005, 5 Marks)**

Ans. : AAD instruction needs BCD digits in AH and AL in the unpacked form. So we must have uppack digits in AL first and then use AAD instruction.

Routine :

```
MOV AH, AL          ; Save digit in AH
MOV CL, 04          ; Initialize rotation count
ROL AH, CL          ; Rotate AH to exchange digits
AND AH, 0FH         ; Mask upper nibble
AND AL, 0FH         ; Mask lower nibble
AAD
RET
```

Q.4 *The 8086 processor has no instruction to manipulate the trap flag. Consider while debugging a program, a user wants the trap flag to remain set, over a portion of the program so that in that region of the program, single stepping is automatically introduced. Write a user generated software interrupt routine, that toggles the trap flag when invoked, so that the user invokes this interrupt at the start of the specific region in the program where he desires to single step, and invokes the same interruption again at the end of the region (where he wants to stop single stepping).*

(Jan./Feb.-2005, 10 Marks)

Ans. : Routine to Toggle T flag :

```
PUSH F          ; [Get the flag contents
POP AX          ;  in AX register]
MOV BH, AH      ; Save higher byte of flag
AND AH, 01H     ; Mask all bits, except TF bit
JZ NEXT         ; if zero goto NEXT
AND BH, 0FEH    ; [otherwise make it
```

```
        MOV AH, BH    ; zero]
        JMP LAST      ; goto LAST
NEXT :  OR BH, 01H    ; [Make IF = 1]
        MOV AH, BH
LAST :  PUSH AX       ; [Save AX contents in
        POP F         ;  the flag]
        RET           ; Return to main program
```

Q.5 *Write a program using DOS INT 21H function call to read a key from keyboard. Display message GOOD if the key pressed is 'G' otherwise, do not display any message.*

(July/Aug.-2005, 8 Marks)

Ans. :

```
. model small
. data MES DB 10,13 'GOOD $'
. code
Start :  MOV ax,@data ; [Initialize
         MOV ds,ax    ;  data segment]
         MOV ah,01    ; [Read
         INT 21h      ;  keyboard]
         CMP al,'G'   ; Check if key is G
         JNZ LAST     ; if not 'G' goto LAST
         LEA dx,MES   ; [Display
         MOV AH,09H   ;  message
         INT 21H      ;  MES]
LAST :   MOV AH,4CH   ; [Terminate and
         INT 21H      ;  exit to DOS]
         END Start
```

Q.6 *An 8086 register stores two 32-bit numbers : number 1 in AX; BX, number 2 in SI : DI. Write an assembly language procedure to compare the two numbers and return with the carry set if number 1 is greater than number 2. The number themselves should be returned unaltered.*

(Jan./Feb.-2006, 10 Marks)

Ans. :

```
Data segment
        M1 DB 10, 13, "Number 1 is greater $"
        M2 DB 10, 13, "Number 2 is greater $"
Data ends
Code segment
assume CS:code,ds:data
        Start :  MOV ax, data
                 MOV DS,ax
                 MOV AX, 6666H
                 MOV BX, 1111H
                 MOV SI, 4444H
                 MOV DI, 1111H
        Call CMPROC
        MOV AH, 4CH
        INT 21H
        code ends
```

```
        end  Start
        PROC CMPROC  NEAR
        CMP  AX,  SI
        JC   L1
        JNZ  L2
        CMP  BX,  DI
        JC   L1
        JZ   L1
L2:     STC
        JMP  LAST
L1 :    STC
        CMC
LAST :  RET
```

Q.7 *Explain how the DO-WHILE loop can be programmed in an 8086 processor.*

(Jan./Feb.-2006, 5 Marks]

Ans. : DO-WHILE loop can be programmed using conditional jump instruction i.e. it will repeat a particular group of instruction until condition is true.

Following example illustrates it. When user press any key from keyboard it is accepted, until user press 'ENTER' key.

```
eg  Data segment
        m1   DB 10,13, "Press key $"
        m2   DB 10,13, "Pressed key $"
    Data ends
    code segment
    assume CS:code,ds:data
    start:      MOV AX,data
                MOV DS,AX
    L1 :        LEA DX,M1;DO
                MOV AH,09H
                INT 21H
                MOV AH,01H
                INT 21H
                MOV BL,AL
                LEA DX,M2
                MOV AH,09H
                INT 21H
                MOV DL,BL
                MOV AH,02H
                INT 21H
                CMP DL, 0DH
                JNZ L1 ; WHILE ZF = 0
                MOV AH, 4CH
```

```
    INT 21H
CODE ENDS
END START
```

Q.8 *An 8086 program in block moved from the code segment starting with address 10000H to another code segment starting with address 23A00H. Is it necessary to change the address of the jump instructions in the new code segment ? If no, illustrate with an example program.* **(July/Aug.-2006, 5 Marks)**

Ans : Yes it is necessary to change the address of the jump instruction in the new code segment. Because if jump instruction is from one segment to another segment (for jump) then values of CS and IP are loaded on stack, if jump is within same segment, then only IP is loaded on stack. i.e. it is near jump.

Q.9 *Write a 8086 program segment to exchange the contents of two non overlapping memory blocks each containing ten words and starting with the addresses SOURCE and DSTN respectively.* **(July/Aug.-2006, 6 Marks)**

Ans. :

```
        .model small
        .stack 100
        .data
        .code
         MOV AX,@data;initialize  DS
         MOV DS, AX ; initialize  DS
         MOV ES, AX ; initialize  ES
         MOV CX, 0014H; initialize counter
         MOV SI,SOURCE; initialize offset in DS
         MOV DI, DSTN; initialize offset in ES
BACK:    MOV AL, DS:[SI];get byte from 1st block
         MOV AH, ES:[DI];get byte from 2nd block
         MOV ES:[DI],AL ; store byte in 2nd block
         MOV DS:[SI],AH ; store byte in 1st block
            INC SI ; increment source pointer
            INC DI ; increment destination pointer
            LOOP BACK; until CX=0
          END
```

Q.10 *Write a full segment 8086 program to check whether the string stored at location starting with the address STRNG is a palindrome. If so, store value 01 in the location PAL. Otherwise store 00 there.*

Use atleast one string instruction. **(July/Aug.-2006, 10 Marks)**

Ans. : Refer program 14.

Q.11 *Write a DO-WHILE constructs to read a key and echo it using DOS INT21 H interrupt until enter key is pressed in the keyboard.* **(July/Aug.-2006, 4 Marks)**

Ans. : Refer answer of Q. 8.

Q.12 *Write a program code using REP MOV SW instruction to transfer 20 words of data from a memory block in segment pointed by DS to another memory block in segment pointed by ES. Explain the role of SI, DI register DF and REP prefix in the program.*

(Jan./Feb.-2007, 8 Marks)

Ans. : Refer Program 17 on page 5-18.

Q.13 *Write a program that uses XLAT instruction to convert a BCD number (0-9) into ASCII number (30h to 39h) store result in location ASCDAT.* **(Jan./Feb.-2007, 5 Marks)**

Ans. :

```
• MODEL SMALL
• DATA
  TABLE  DB   30H
         DB   31H
         DB   32H
         DB   33H
         DB   34H
         DB   35H
         DB   36H
         DB   37H
         DB   38H
         DB   39H
ASCDAT
• CODE
START : MOV AX,@DATA   ; [Initialize
        MOV DS, AX     ;   Data Segment]
        MOV AL, 08H    ;   Loads AL with any BCD
                       ;   for example 8, to be
                       ;   converted to ASCII code
        MOV BX, OFFSET TABLE  ;   Load BX with the
                              ;   offset of starting address
                              ;   of lookup table
        XLAT           ;   Copy byte from address
                           pointed by [BX + AL] back
                           into AL
        MOV AH, 4CH    ;   [EXIT to
        INT 21H        ;   DOS]
        END START
        END
```

Q.14 *Write an assembly language program to add 8 digit BCD number in AX-BX register to 8 digit BCD number in CX-DX register. AX-CX are most significant registers and result is found in CX-DX registers after addition.* **(Jan./Feb.-2007, 10 Marks)**

Ans. :

```
PUSH    CX        ;   save cx register
XCHG    AX, BX    ;   Get lower word in AX
ADD     AL, DL    ;   Add first byte
DAA               ;   Adjust AL to BCD
```

```
MOV     DL, AL  ;   save result of addition of first byte
PUSHF           ;   save Flags
MOV     CL, 04H ;   Exchange
ROL     AX, CL  ;   AH and AL
POPF            ;   Restore flags
ADC     AL, DH  ;   Add second byte with carry
DAA             ;   adjust AL to BCD
MOV     DH, AL  ;   save result of addition of second byte
POP     CX      ;   Restore CX
XCHG    AX, BX  ;   Get the higher word in Ax
ADC     AL, CL  ;   Add third byte with carry
DAA             ;   adjust AL to BCD
MOV     CL, AL  ;   save result of addition of third byte
PUSH    CX      ;   save CX register
PUSHF           ;   save flags
MOV     CL, 04H ;   Exchange
ROL     AX, CL  ;   AH and AL
POPF            ;   Restore Flags
POP     CX      ;   Restore CX
ADC     AL, CH  ;   Add fourth byte with carry
DAA             ;   adjust AL to BCD
MOV     CH, AL  ;   save result of addition of fourth byte.
```

Q.15 *Write an ALP that reverses the given string stored at STRNG save the result at REVSTG.* **(Jan./Feb.-2007, 6 Marks)**

Ans. : Refer program 12 on page 5-7.

Q.16 *Write an assembly language program to arrange 'N' bytes of data is ascending order. Give the comments for each of the instructions used in developing your program.* **(July/Aug.-2007, 8 Marks)**

Ans. : Refer program 25 on page 5-62.

Q.17 *Write an ALP for addition of two 3×3 matrices. The matrices are stored in the form of lists (row wise). Store the result of addition in the third list.* **(Jan./Feb.-2008, 10 Marks)**

Ans. : Refer program 9 on page 5-2.

Q.18 *Write an ALP to read a string of characters from keyboard without echo and display.* **(Jan./Feb.-2008, 6 Marks)**

Ans. :

```
• MODEL SMALL
• STACK 100
• DATA
STR   DB 80 DUP('$')
MES1 DB 10, 13, 'ENTER THE STRING $'

START :  MOV AX, @DATA     ;    [Initialize
         MOV DS, AX        ;    Data segment]
         MOV AH, 09H       ;    Display message
```

```
            LEA DX, MES1
            INT 21H
            LEA BX, STR         ;   Load address of STR in BX
BACK :      MOV AH, 08H         ;   Read key without echo
            INT 21H
            CMP AL, 13          ;   check for Enter key
            JZ DISP
            MOV [BX], AL        ;   Save key
            INC BX              ;   Increment pointer
            JMP BACK
DISP :      MOV AH, 09H         ;   Display string
            LEA DX, STR
            INT 21H
            MOV AH, 4CH         ;   Return to DOS
            INT 21H
            END START
            END)
```

Q.19 *Write a procedure to set and a procedure reset the TRAP flag to enable and disable. Single stepping of the program.* **(Jan./Feb.-2008, 6 Marks)**

Ans. : Refer similar program from Q.5.

Q.21 *Write subroutines to i) Set the TRAP flag ii) Reset the TRAP flag.* **(July/Aug.-2008, 4 Marks)**

Ans. : Refer similar program from Q.5.

❑❑❑

6 8086 Interrupts

Q.1 *On receiving a hardware interrupt, the 8086 processor pushes the flag to the stack and clears the TF and the IF before doing any further operation. Explain why this is required.* **(Jan./Feb.-2005, 6 Marks)**

Ans. : The 8086 processor pushes the flags to the stack and clears the TF and the IF before doing any further operation for following reasons :

- After execution of interrupt service routine 8086 resumes execution of main program from where it has left. 8086 processor pushes the flags to the stack to store the status of all flags so it can be restored when 8086 resumes the execution of main program.
- 8086 clears the TF and the IF flags to prevent any further interrupts from INTR interrupt input and as a trap.

Q.2 *What is an interrupt vector ? Explain, in detail, the events that occur when a real mode interrupt becomes active.* **(July/Aug.-2005, 6 Marks)**

Ans. : Refer section 6.2.3.

Q.3 *Explain the software interrupt operation of 8086.* **(Jan./Feb.-2006, 5 Marks)**

Ans. : Refer section 6.3.6.

Q.4 *Explain the hardware interrupt operation of the 8086 processor from the time the interrupt pin is activated to the time the interrupt routine starts executing.* **(Jan./Feb.-2006, 10 Marks)**

Ans. : Refer section 6.2.3.

Q.5 *Describe the reserved interrupts of 8086.* **(July/Aug.-2006, 5 Marks)**

Ans. : Refer section 6.2.3.

Q.6 *Describe a scheme to generate an NMI interrupt on power failure.* **(July/Aug.-2006, Jan./Feb.-2007, July/Aug.-2007, 5 Marks)**

Ans. : The 8086 will automatically do a type 2 interrupt response when it receives a low-to-high transition on its NMI input pin. Nonmaskable input pin means the type 2 interrupt response cannot be disabled by any program instructions.

Use of NMI in case of power failure is to save program data. Some external circuitry detects when the a.c. power to the system fails and sends an interrupt signal to the NMI input. Because of the large filter capacitors in most power supplies. The d.c. system power

will remain for perhaps 50 ms after the a.c. power is gone. This is more than enough time for a type 2 interrupt service procedure to copy program data to some RAM which has a battery backup power supply. When the a.c. power returns, program data can be restored from the battery backed RAM, and the program can resume execution where it left off.

Q.7 *Explain the Software Interrupts.* **(Jan./Feb.-2007, 3 Marks)**

Ans. : Refer section 6.3.6.

Q.8 *Explain the interrupt structure in 8086. Write the functions of atleast five dedicated software interrupts in 8086.* **(July/Aug.-2007 8 Marks)**

Ans. : Refer section 6.3.

Q.9 *Write the note on 8259 interrupt controller.* **(Jan./Feb.-2008, July/Aug.-2008, 6 Marks)**

Ans. : Refer section 6.5.

Q.10 *What is an interrupt vector table? Explain the events that occur when INTR interrupt is activated.* **(July/Aug.-2008, 10 Marks)**

Ans. : Refer section 6.2.3.

Q.11 *Write down the interrupt priority of 8086.* **(July/Aug.-2008, 2 Marks)**

Ans. : Refer section 6.4.

❑❑❑

7 8086/8088 Configurations

Q.1 *With appropriate circuit diagrams, explain how you would generate, data, address and control buses for memory and I/O interfacing from an 8086 processor in the MAX mode of operation.* **(Jan./Feb.-2005, 10 Marks)**

Ans. : Refer section 7.7.1.

Q.2 *Indicate the action of the $\overline{LOCK}$ pin using a timing diagram, when executing the instruction LOCK : X CHG AX, [SI], in an 8086 system.* **(Jan./Feb.-2005, 4 Marks)**

Ans. : The LOCK prefix causes the $\overline{LOCK}$ pin to activate for the duration of a locked instruction. Thus, during the execution of LOCK XCHG AX, [SI] instruction $\overline{LOCK}$ pin will be low. The $\overline{LOCK}$ pin will remain low for two memory read cycles if the word is accessed from odd address; otherwise it will remain low for one memory read cycle.

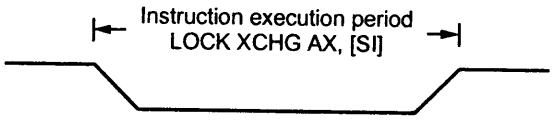


Fig. 1

Q.3 *Draw the timing diagram showing all relevant signals for a memory write cycle in an 8086 system in the MAX mode with no WAIT state.* **(Jan./Feb.-2005, 6 Marks)**

Ans. : Refer section 7.7.3.1

Q.4 *Sketch the read bus cycle of 8086 and explain briefly.* **(July/Aug.-2005, 8 Marks)**

Ans. : Refer section 7.6.3.1

Q.5 *Why address demultiplexing is required in 8086 processor ? Explain how address demultiplexing is done.* **(July/Aug.-2005, 6 Marks)**

Ans. : Refer section 7.6.1.

Q.6 *What is instruction queue ? Explain the use of instruction queue and its relation with the pins QS0 and QS1 of 8086 processor.* **(July/Aug.-2005, 6 Marks)**

Ans. : Refer page 2-3 and page 7-4.

Q.7 *With the help of block diagram and with the clear indication of signals, explain how control signals for memory and I/O read/write operations can be generated when 8086 processor is used in maximum mode.* **(July/Aug.-2005, 8 Marks)**

Ans. : Refer section 7.7.1.

Q.8 *Write short note on minimum mode of operation of 8086 processor.* **(July/Aug.-2005, 5 Marks)**

Ans. : Refer section 7.6.1.

Q.9 *Show the timing diagram to execute a memory read operation in an Intel 8086 processor.* **(Jan./Feb.-2006, 10 Marks)**

Ans. : Refer section 7.6.3.1.

Q.10 *What are the different status that are given out on the bus $\overline{S}_2$, $\overline{S}_1$ and $\overline{S}_0$ in maximum mode of 8086 ? How different control signals are generated from this bus ? Explain briefly each of these control signals.* **(July/Aug.-2006, 10 Marks)**

Ans. : Refer section 7.2.3.

Q.11 *Write bus cycle in minimum mode. How to introduce wait states in this bus cycle by slow memory or I/O devices ?* **(July/Aug.-2006, 10 Marks)**

Ans. : Refer section 7.6.3.1.

Q.12 *Show the timing diagram to execute a memory write operation in 8086 in minimum mode.* **(Jan./Feb.-2007, 6 Marks)**

Ans. : Refer section 7.6.3.1.

Q.13 *Explain 8284A clock generator.* **(Jan./Feb.-2007, 4 Marks; July/Aug.-2008, 5 Marks)**

Ans. : Refer section 7.6.1.

Q.14 *Why address demultiplexing is required in 8086? Explain how it is implemented.* **(July/Aug.-2007, 8 Marks)**

Ans. : Refer section 7.5.

Q.15 *Explain the functions of the following pins of 8086.*

i) RESET, ii) $\overline{BHE}$ / S_7, iii) $\overline{LOCK}$ iv) MN/$\overline{MX}$. **(Jan./Feb.-2008, 4 Marks)**

Ans. : Refer section 7.2.

Q.16 *Explain the MAX mode of 8086 with block diagram.* **(Jan./Feb.-2008, 12 Marks)**

Ans. : Refer section 7.7.1.

Q.17 *Explain the concept of bus buffering and latching with a neat diagram.* **(Jan./Feb.-2008, 8 Marks)**

Ans. : Refer section 7.6.1.

Q.18 *Explain in brief the minimum mode and maximum mode operation of 8086.*
(July/Aug.-2008, 6 Marks)

Ans. : Refer sections 7.6.1 and 7.7.1.

Q.19 *Draw the timing diagram for a memory read cycle of 8086 in the MAX mode.*
(July/Aug.-2008, 6 Marks)

Ans. : Refer section 7.7.3.1.

❑❑❑

11 8086/8088 Based Multiprocessing Systems

Q.1 *Write an 8086/8087 system ALP to find the frequency response characteristics of a series L-C-R circuit in the form of a table. Assume the response to correspond to the power expended in the L-C-R at constant input voltage with varying frequency.*

(Jan./Feb.-2005, 10 Marks)

Ans. : In series LCR circuit power dissipated is given by I^2R. L and C do no dissipate any power. The current I is given by V/Z, where Z is equal to $\sqrt{R^2+(X_L-X_C)^2}$. We know that, $X_C = 1/2\ \pi\ fC$ and $X_L = 2\ \pi\ fL$. To make a table, frequency Vs dissipated power we have calculate I^2R for each frequency. We start from 100 Hz and take thousand readings with the interval of 100 Hz.

```
     .model small
     .data
     R1 DD 100
     C          DD 0.000001
     L          DD 0.0001
     FRE        DD 100              ; Minimum frequency
     POWER      DD ?
     V          DD 5.0              ; Input voltage
     X_L        DD ?
     X_C        DD ?
     A_FRE      DD 1000 DUP (?)
     A_POW      DD 1000 DUP (?)
          TWO   DD 2.0
          HUN   DD 100.0
.Code
Start :         MOVE AX, @ data     ; [Initialize
                MOV DS, AX          ; data segment]
                XOR DI              ; Clear DI
                MOV CX, 1000        ; Initialize couter
                FINIT               ; Initialize 8087
BACK :          FLD L
                FMUL TWO            ; 2L
                FLDPI
                FMUL                ; 2πL
                FLD FRE
                FMUL                ; 2πfL
                FSTP X_L            ; Save X_L
                FLD C
                FMUL TWO            ; 2C
                FLDPI
                FMUL                ; 2πfC
```

```
FLD FRE
FMUL                ; 2πfC
FLD 1
FDIVR               ; X_C = 1/2 πfC
FSTP X_C            ; Save X_C
FLD X_L
FUSB X_C            ; X_L - X_C
FMUL ST, ST(0)      ; (X_L - X_C)^2
FLD R1
FMUL ST, ST(0)      ; R^2
FADD ST, ST(1)      ; R^2 + (X_L - X_C)^2
FSQRT               ; √(R^2 + (X_L - X_C)^2) = Z
FLD V
FDIVR               ; V/Z = I
FMUL ST, ST(0)      ; I^2
FDIV R1             ; I^2/R = Power dissipated
FSTP POWER          ; Save power
MOV A_FRE [DI] , FRE
MOV A_POW [DI], POWER
INC DI
INC DI
INC DI
INC DI
FLD FRE
FADD HUN                ; FRE = FRE + 100.0
FSTP FRE                ; Save FRE
LOOP BACK
END start
```

Q.2 *Explain floating point formats for 80 × 87 co-processors and illustrate, with example, the steps for converting decimal number to single precision floating point number and vice versa.* **(July/Aug.-2005, 10 Marks)**

Ans. : Refer section 11.4.6.

Steps to convert single precision floating point number to decimal number are as follows :

- Separate sign bit, next 8-bit as exponent and remaining bits as a fraction.
- Find the actual exponent by removing bias exponent from exponent field.
- Denormalize the number.

Q.3 *Explain numeric execution unit of 80×87 co-processor.* **(July/Aug.-2005, 4 Marks)**

Ans. : The numeric execution unit of 8087 executes all instructions that involve the register stack; these include arithmetic, logical, transcendental, constant and data transfer instructions. The data path in the numeric execution unit is 84-bits wide (68 fractions bits, 15 exponent bits and a sign bit) which allow internal operand transfers to be performed at very high speed.

Q.4 *What is the difference between*

i) Forward and reverse division?

ii) FADD and FADDP?

iii) FSTSW and FNSTSW instructions?

iv) FTST and FXAM instructions? **(July/Aug.-2005, 4 Marks)**

Ans. : i) Forward and reverse division : Refer page 11-23.

ii) FADD and FADDP : Refer section 11.4.8.2.

iii) FSTSW and FNSTSW instructions : The FSTSW instruction copies the status register into destination register. The FNSTSW instruction also copies the status register into destination register, but without a wait.

iv) FTST and FXAM instructions : Refer section 11.4.8.3.

Q.5 *Distinguish between the following pairs of pentium floating point instructions :*

i) FADD and FADDP M32; (double word or 32-bit memory)

ii) FADD and FADD ST (3), ST

iii) FADD and FADD M32; (32 bit-memory) **(Jan./Feb.-2006, 6 Marks)**

Ans. :

i) FADD - FADD Destination, source

Adds real number from specified source to real number at specified destination

FADDP M32 - Adds ST to specified stack element and increments stack pointer by one.

ii) FADD - will add real number from specified source to real number at specified destination.

FADD ST (3), ST - It will add ST to ST (3) and result in ST (3).

iii) FADD will add real number from specified source to real number at specified location.

FADD M32 - Add real number from memory and ST.

Q.6 *Work out the floating point number representation in the IEEE 754, single and double precision standard formats for the number 3.625 decimal.* **(Jan./Feb.-2006, 10 Marks)**

Ans. : 1) Convert decimal number to binary format

$(3.625)_{10}$ **Integer part :** 3 = 0011

Functional part :

$0.625 \times 2 = 1.250$ 1

$0.250 \times 2 = 0.500$ 0

$0.500 \times 2 = 1.00 \quad 1$

Binary number = 0011.101

= 11.101

2) Normalize the number

$$11.101 = 1.1101 \times 2^1$$

Representation of number in different formats :

i) Short real

For given number

S = 0, E = 1, F = 1101

Bias for short real format = 127

Biased exponent = 1 + 127 = 128

= 10000000

Number in short real format

0 10000000 1101 00...0 = 3.625_{10}

S Exponent Fraction

ii) Long real

Bias for long real format = 1023

Biased exponent = 1 + 1023 = 1024

= 10000000000

Number in long real format

0 10000000000 1101 00...0 = 3.625_{10}

S Exponent Fraction

iii) Biased for Temporary real format = 16383

Biased exponent = 1 + 16383 = 16384 = 100000000000000

Number in temporary real format

0 100000000000000 1101 00...0 = 3.625_{10}

S Exponent Fraction

Q.7 *Give the tag bit interpretation for the tag word in the tag register of 80×87 floating point processor.* **(Jan./Feb.-2006, 4 Marks)**

Ans. : Refer section 11.4.5.4.

Q.8 *With a block schematic, explain the architecture of arithmetic processor 8087.* **(July/Aug.-2006, 10 Marks)**

Ans. : Refer section 11.4.5.

Q.9 *Explain with a neat block diagram the architecture of arithmetic processor 8087.* **(Jan./Feb.-2007, 8 Marks)**

Ans. : Refer section 11.4.5.

Q.10 *Describe the operations performed by each of the following 8087 instructions:*

i) FLD TAX-RATE ii) FSQRT iii) FLDPI iv) FPTAN. **(July/Aug.-2007, 4 Marks)**

Ans. : **i) FLD TAX-RATE** : Refer section 11.4.8.1.

ii) FSQRT : Refer section 11.4.8.2.

iii) FLDPI : Refer section 11.4.8.5.

iv) FPTAN : Refer section 11.4.8.4.

Q.11 *Write a program to compute the volume of a sphere using 8087 instruction. (Use formula* $V = 4\ \pi R^3/3$*).* **(July/Aug.-2007, 8 Marks)**

Ans. :

```
        • MODEL SMALL
        • DATA
          RADIUS    DD   3.4
          VOLUME    DD   ?
          THREE     DD   3.0  ;    constant
          FOUR      DD   4.0  ;    constant
      • CODE
START     :    MOV AX, @DATA      ;[Initialize
               MOV DS, AX         ; DATA Segment]
               FINIT              ;   Initialize 8087
               FLD RADIUS         ;   Load radius (R) in ST
               FMUL ST, ST(0)     ;   R^2
               FMUL RADIUS        ;   R^3
               FLDPI              ;   Load PI
               FMUL  ST, ST(1)    ;   π R^3
               FMUL FOUR          ;   4π R^3
               FDIV THREE         ;   4π R^3/3
               FSTP VOLUME        ;   store result
               END START
```

Q.12 *What is a coprocessor? Explain the features of an 8087 NDP (Numeric Data Processor).* **(July/Aug.-2007, 8 Marks)**

Ans. : Refer section 11.4.

Q.13 *Write note on : Data types used in numeric coprocessor.* **(July/Aug.-2007, 5 Marks)**

Ans. : Refer section 11.4.6.

Q.14 *Explain the different 8087 data types.* **(Jan./Feb.-2008, 7 Marks)**

Ans. : Refer section 11.4.6.

Q.15 *Write an ALP using 8086 and 8087 instructions to find the area of a circle given the radius.* **(Jan./Feb.-2008, 7 Marks)**

Ans. : Refer section 11.4.9.

Q.16 *With a neat block diagram, explain the architecture of 8087 arithmetic coprocessor.* **(July/Aug.-2008, 10 Marks)**

Ans. : Refer sections 11.2 and 11.4.5.

❑❑❑

12 Bus Interface

Q.1 *Indicate the basic features of the PCI bus or the USB.*
(Jan./Feb.-2005, 5 Marks; Jan./Feb.-2006, 10 Marks)

Ans. : Refer sections 12.2.1 and 12.4.1.

Q.2 *Write short note on parallel printer interface.* **(Jan./Feb.-2005, 7 Marks)**

Ans. : Refer section 12.3.

Q.3 *Write short note on PCI bus.* **(July/Aug.-2005, 5 Marks)**

Ans. : Refer section 12.2.

Q.4 *Explain the features of USB, its pin details, data format and stop and wait control mechanism.* **(July/Aug.-2006, 8 Marks)**

Ans. : Refer section 12.14.

Q.5 *Explain USB.* **(Jan./Feb.-2007, 4 Marks)**

Ans. : Refer section 12.14.

Q.6 *What are bus standards? Mention any three PC bus architectures. Give the salient features of PCI bus.* **(July/Aug.-2007, 8 Marks)**

Ans. : Refer sections 12.1 and 12.2.1.

Q.7 *Explain the features of USB and LPT interface.* **(July/Aug.-2007, 8 Marks)**

Ans. : Refer sections 12.2.1 and 12.3.

Q.8 *Write the features of USB, its pin details. Explain USB data formats and commands.*
(Jan./Feb.-2008, 8 Marks)

Ans. : Refer section 12.14.

Q.9 *Explain the features of the following in brief : i) PCI bus ii) USB bus.*
(July/Aug.-2008, 10 Marks)

Ans. : Refer sections 12.2.1 and 12.14.1.

❑❑❑

13 The 80386, 80486 and Pentium Processor

Q.1 *Describe the basic 486 architecture.* **(Jan./Feb.-2005, 5 Marks)**

Ans. : Refer section 13.6.2.

Q.2 *Write short note on Pentium microprocessor.* **(Jan./Feb.-2005, 7 Marks; Jan./Feb.-2007, 4 Makrs)**

Ans. : Refer section 13.7.

Q.3 *Explain control, debug and test registers of 80386 processor.* **(July/Aug.-2005, 12 Marks)**

Ans. : Refer section 13.1.3.

Q.4 *What is cache memory ? Explain briefly taking 80486 processor as reference.* **(July/Aug.-2005, 8 Marks)**

Ans. : In a computer system the program which is to be executed is loaded in the main memory (DRAM). Processor then fetches the code and data from the main memory to execute the program. The DRAMs which form the main memory are slower devices. So it is necessary to insert wait states in memory read/write cycles. This reduces the speed of execution. To speed up the process, high speed memories such as SRAMs must be used. But considering the cost and space required for SRAMs, it is not desirable to use SRAMs to form the main memory. The solution for this problem is come out with the fact that most of the microcomputer programs work with only small sections of code and data at a particular time. In the memory system small section of SRAM is added along with main memory, referred to as **cache memory**.

Refer section 13.6.2.

Q.5 *Bring out the main additional features of the Pentium processor in comparison to the 80386 processor.* **(Jan./Feb.-2006, 10 Marks)**

Ans. : Refer section 13.7.

Q.6 *Explain where the based scaled indexed addressing with displacement will be useful.* **(Jan./Feb.-2006, 5 Marks)**

Ans. : Refer section 13.25. This addressing mode is useful to access particular field in the record.

Q.7 *Describe the features of (i) 80386 and 80486 and (ii) Memory sytem of 80386.* **(July/Aug.-2006, 8 Marks)**

Ans. : Refer sections 13.1.1, 13.6.1 and 13.4.

Q.8 *List the extended registers found in 80386 microprocessor.* **(July/Aug.-2007, 4 Marks)**

Ans. : Refer sections 13.1.2 and 13.1.3.

Q.9 *Write a note on the memory organization in advanced microprocessors.* **(July/Aug.-2007, 4 Marks)**

Ans. : Refer section 13.4.

Q.10 *What are the unique features of a Pentium Processor?* **(July/Aug.-2007, 8 Marks)**

Ans. : Refer section 13.7.

Q.11 *Explain the operation of the 80486 cache memory.* **(July/Aug.-2007, 8 Marks)**

Ans. : Refer section 13.6.6.

Q.12 *Write the structure of control register and debug and test register of 80386 and explain.* **(Jan./Feb.-2008, 6 Marks)**

Ans. : Refer section 13.1.3.

❑❑❑

Q.1 *Write an ALP to check whether the 32-bit data stored at location starting with address VALUE is NIBBLE-wise palindrome. If true store FF in 'AL'else store 00 in 'AL'.*

(Jan./Feb.-2008, 10 Marks)

Ans. : Refer program 5 on page L-6.

Q.2 *Write a program to open a new file EC. DAT in the current directory and drive. If it is successfully opened, write 200 H bytes of data into it from a data block named BLOCK.*

(Jan./Feb.-2008, 10 Marks)

Ans. : Refer similar program 7 on page L-22.

Q.3 *Write an 8086 assembly language program to check if the given data byte is a valid 2 out of 5 code. If valid display the message 'A VALID CODE' else display the message 'NOT A VALID CODE'.* **(July/Aug.-2008, 6 Marks)**

Ans. : Refer program 4 on page L-6.

❑❑❑