Program No*: 1* *INTRODUCTION to MASM*

EDITOR:

An editor is a program, which allows you to create a file containing the assembly language statements for your program. As you type in your program the editor stores the ASCII codes for the letters and numbers in successive RAM locations. When you have typed in all of your programs you then save the file on a floppy of hard disk. This file is called source file. The next step is to process the source file with an assembler. In the MASM/TASM assembler you should give your source file name the extension .ASM

ASSEMBLER:

*A*n assembler program is used to translate the assembly language mnemonics for instructions to the corresponding binary codes. When you run the assembler it reads the source file of your program the disk where you saved it after editing on the first pass through the source program the assembler determines the displacement of named data items the offset of labels and this information in a symbol table. On the second pass through the source program the assembler produces the binary code for each instruction and inserts the offset etc that is calculated during the first pass. The assembler generates two files on floppy or hard disk. The first file called the object file is given the extension. OBJ. The object file contains the binary codes for the instructions and information about the addresses of the instructions. The second file generated by the assembler is called assembler list file. The list file contains your assembly language statements the binary codes for each instructions and the offset for each instruction. In MASM/TASM assembler MASM source file name ASM is used to assemble the file. Edit source file name LST is used to view the list file which is generated, when you assemble the file.

LINKER :

A linker is a program used to join several object files into one large object file and convert to an **exe** file. The linker produces a link file, which contains the binary codes for all the combined modules. The linker however doesn't assign absolute addresses to the program, it assigns is said to be re-locatable because it can be put anywhere in memory to be run. In TASM/MASM LINK source filename is used to link the file.

DEBUGGER:

A debugger is a program which allows you to load your object code program into system memory, execute the program and troubleshoot are debug it the debugger allows you to look at the contents of registers and memory locations after your program runs. It allows you to change the contents of register and memory locations after your program runs. It allows you to change the contents of register and memory locations and return the program. A debugger also allows you to set a break point at any point in the program. If you inset a breakpoint the debugger will run the program up to the instruction where the breakpoint is set and stop execution. You can then examine register and memory contents to see whether the results are correct at that point. In MASM, MD filename is used to debug the file.

DEBUGGER FUNCTIONS:

1. Debugger allows to look at the contents of registers and memory locations.
2. We can extend 8-bit register to 16-bit register which the help of extended register option.
3. Debugger allows to set breakpoints at any point with the program.
4. The debugger will run the program up to the instruction where the breakpoint is set and then stop execution of program. At this point, we can examine registry and memory contents at that point.
5. With the help of dump we can view register contents.
6. we can trace the program step by step with the help of T.
7. We can execute the program completely at a time using G.

DEBUGGER COMMANDS

ASSEMBLE: To write assembly language program from the given address.

A starting address <cr>

Eg: a 4000 <cr>

Starts program at an offset of 4000.

DUMP: To see the specified memory contents

D memory location first address last address

(While displays the set of values stored in the specified range, which is given above)

Eg: d 2000, 2010 <cr>

Display the contents of memory locations from 2000 to 2010 (including).

GO: To execute the program

G: one instruction executes (address specified by IP)

G address <cr>: executes from current IP to the address specified

G first address last addresses <cr>: executes a set of instructions specified between the given addresses.

QUIT: To exit from the debugger.

Q < cr >

REGISTER: Shows the contents of Registers

R register name

Eg: r ax

Shows the contents of register.

TRACE: To trace the program instruction by instruction.

T = 2000 <cr>: traces only the current instruction. (Instruction specified by IP)

T = 1000 02 <cr>: Traces instructions from 100 to 101, here the second argument specifies the number of instructions to be traced.

UNASSEMBLE*:* To unassembled the program.

Shows the op-codes along with the assembly language program.

U 4000 <cr>: unassembled from 4000 onwards instructions.

U 5000 6000 <cr>: unassembles the lines from 5000 to 6000

Program No: 2 (a) *MULTY BYTE ADDITION*

```
DATA SEGMENT
N1    DB    55H, 66H, 77H
N2    DB    11H, 22H, 33H
RESULT     DB   3H  DUP (00)
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA

        START:    MOV AX, DATA
                  MOV DS, AX
                  MOV SI, OFFSET N1
                  MOV DI, OFFSET N2
                  MOV BX, OFFSET RESULT
                  CLC
                  MOV CX, 0003H
                  MOV AX, 0000H
        BACK:     MOV AL, [SI]
                  MOV DL, [DI]
                  ADC AL, DL
                  MOV [BX], AL
                  INC SI
                  INC DI
                  INC BX
                  DEC CX
                  JNZ   BACK
                  MOV AH, 4CH
                  INT 21H
                  INT  3H
            CODE ENDS
            END START
```

RESULT:

```
  55H   66H   44H
  11H   22H   33H
  66H   88H   77H
```

Viva Questions:_

1. What is the Function of ADC ?
2. What is the purpose of BX register?
3. What is the Function of CLC?
4. What is the other instruction which can be used instead of MOV SI offset N1?
5. What is the function of MOV AH, 4CH & INT 21H?
6. What is the purpose of INT 3H?

Program No: 2(b) *MULTY BYTE SUBTRACTION*

```
DATA SEGMENT
N1    DB    55H, 66H, 77H, 88H
N2    DB    11H, 22H, 33H, 44H
RESULT     DB 4H  DUP(00)
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA

        START:    MOV AX, DATA
                  MOV DS, AX
                  MOV SI, OFFSET N1
                  MOV DI, OFFSET N2
                  MOV BX, OFFSET RESULT
                  CLC
                  MOV CX, 0004H
                  MOV AX, 0000H
        BACK:     MOV AL, [SI]
                  MOV DL, [DI]
                  SBB AL, DL
                  MOV [BX], AL
                  INC SI
                  INC DI
                  INC BX
                  LOOP BACK
                  MOV AH, 4CH
                  INT 21H
                  INT 3H
            CODE ENDS
            END START
```

RESULT:

55H	66H	77H	88H
11H	22H	33H	44H
44H	44H	44H	44H

Viva Questions:_

1. Why subtract with carry instruction is used in the loop?
2. What is the purpose served by BX register?
3. Why subtraction is done with AL register why not with AX ?
4. What is the other instruction which can be used instead of MOV DI, offset N2 ?

Program No: 2(c) *MULTY BYTE MULTIPLICATION*

```
DATA SEGMENT
N1    DB    05H, 04H, 02H
N2    DB    01H, 02H, 03H
RESULT     DB   4H  DUP (00)
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA

        START:    MOV AX, DATA
                  MOV DS, AX
                  MOV SI, OFFSET N1
                  MOV DI, OFFSET N2
                  MOV BX, OFFSET RESULT
                  MOV CL, 03H
                  MOV AX, 0000H
                  MOV DX, 0000H
        BACK:     MOV AL, [SI]
                  MOV CH, [DI]
                  MUL  DH
                  MOV [BX], AL
                  INC SI
                  INC DI
                  INC BX
                  LOOP BACK
                  MOV AH, 4CH
                  INT 21H
                  INT 3H
            CODE ENDS
            END START
```

RESULT:

05H	04H	02H
01H	02H	03H
05H	08H	06H

Viva Questions:_

1. What is the use of stack pointer?
2. How many model assignment are the name them?
3. What is a directive?
4. What is a pseudo operation?
5. ORG 2000H implies what?
6. Al register is used why not AX?
7. What is the purpose of INT3 in the program?
8. What is the purpose of MOV AH, 4CH and INT 21H in the program?.

Program No: 2 (d) ***MULTY BYTE DIVISION***

```
DATA SEGMENT
N1      DB    55H, 66H, 99H
N2      DB    11H, 22H, 33H
RESULT  DB  3H  DUP(00)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA

        START:  MOV AX, DATA
                MOV DS, AX
                MOV SI, OFFSET N1
                MOV DI, OFFSET N2
                MOV BX, OFFSET RESULT
                MOV CL, 03H
                MOV AX, 0000H
                MOV DX, 0000H
        BACK:   MOV AL, [SI]
                MOV CH, [DI]
                DIV CH
                MOV [BX], AL
                INC SI
                INC DI
                INC BX
                LOOP BACK
                MOV AH, 4CH
                INT 21H
                INT 3H
            CODE ENDS
            END START
```

RESULT: 55H 66H 99H
11H 02H 03H
05H 33H 33H

Viva Questions:_

1. What AL has been used and not AX ?
2. What happens if num1 contains 0AAH and num2 contains 0FFH. ?
3. How do you account for the difference obtained in previous question?
4. Why should AX be used not AL. ?
5. What happens if num1 and num2 values are interchanged?
6. If carry is set to 1 before subtraction what is the instruction to be used?
7. What is an extended accumulator?
8. AL and BL are used for multiplying why not AX & BX?
9. Instead of using MOV BL is it not possible to MUL num2?
10. What is the instruction used for signed multiplication?

Program No: 3(a) ***SIGNED MULTIPLICATION***

```
DATA SEGMENT
        N1 DB 09H
        N2 DB 02H
        N3 DB 02H DUP(00)
DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

   START: MOV AX, DATA
          MOV DS, AX
          SUB AX, AX
          MOV AL, N1
          MOV CH, N2
          MOV BX, OFFSET N3
          IMUL CH
          MOV [BX], AL
          MOV AH, 4CH
          INT 21H
          INT 3H
          CODE ENDS
          END START
```

RESULT:

```
  09H
  02H
  18H
```

Viva Questions:

1. What is the difference between IMUL and MUL?
2. What is the use of instruction CBW?
3. What is the use of instruction CWD?
4. What is the use of instruction pointer?
5. What is the use of index pointers?

Program No: 3(b) *SIGNED DIVISION*

```
DATA SEGMENT
        N1   DB   40H
        N2   DB   20H
        N3   DB   02H DUP (00H)
DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

            START:   MOV AX, DATA
                     MOV DS, AX
                     XOR AX, AX
                     MOV AL, N1
                     CBW
                     MOV CH, N2
                     MOV BX, OFFSET N3
                     IDIV CH
                     MOV [BX], AL
                     MOV AH, 4CH
                     INT 21H
                     INT 3H
                CODE ENDS
                END START
```

RESULT:

```
  40H
 20H
800H
```

Viva Questions:

1. What is the purpose of SUB AX,AX?
2. What is the difference between IDIV and DIV?
3. What is the use of instruction CBW & CWD?
4. What is the importance of segmentation?
5. What is the difference between signed & unsigned numbers?

Program No: 4(a) *ASCII ADDITION*

```
DATA SEGMENT
        N1 DB 14H
        N2 DB 04H
        N3 DB  2H DUP (00)
DATA ENDS

CODE SEGMENT
                ASSUME CS:CODE, DS:DATA

                START:      MOV AX, DATA
                            MOV DS, AX
                            XOR AX, AX
                            MOV AL, N1
                            MOV CL, N2
                            MOV BX, OFFSET N3
                            ADD AL, CL
                            AAA
                            MOV [BX], AL
                            MOV AH, 4CH
                            INT 21H
                            INT 3H
                CODE ENDS
                END START
```

Viva Questions:

1. What is the purpose of AAA instruction?
2. What is the importance of ASCII addition?
3. What is the difference between addition and ASCII addition?
4. What is the importance of INT 21H?
5. What is meant by pipe lining?

Program No: 4(b) *ASCII SUBTRACTION*

```
DATA SEGMENT
        N1  DB 04H
        N2  DB  02H
        N3  DB 02H DUP (00)
DATA ENDS

    CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

        START:  MOV AX, DATA
                MOV DS, AX
                XOR AX, AX
                MOV AL, N1
                MOV  BX, OFFSET N3
                SUB AL,   N2
                AAS
                MOV [DL], AL
                MOV AH, 4CH
                INT 21H
                INT 3H

            CODE ENDS
            END START
```

Viva Questions:

1. What is the purpose of ASCII subtraction?
2. What is the instruction used for ASCII subtraction? _
3. What is the purpose of AAS?
4. What is difference between AAS and AAA?
5. What is the importance of ASCII No's?

Program No: 4(c) *ASCII* MULTIPLICATION

```
DATA SEGMENT
        N1 DB 04H
        N2 DB  03H
        N3  DB 02H DUP(00)
DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

                START:      MOV AX, DATA
                            MOV DS, AX
                            XOR AX, AX
                            MOV AL, N1
                            MOV BX,  OFFSET N3
                            MUL N2
                            AAM
                            MOV [BL], AL
                            MOV AH, 4CH
                            INT 21H
                            INT 3H
                        CODE ENDS
                        END START
```

Viva Questions:

1. What is the purpose of XCHG instruction is ASCII adjust after multiplication. ?
2. Why is ASCII adjust after multiply cab be called as 1 byte binary to BCD conversion?
3. What does AL & AH contains?

Program No: 4(d) ***ASCII DIVISION***

```
DATA SEGMENT
        N1 DB  06H
        N2 DB  04H
        N3 DB 02H DUP (00)
DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

              START:    MOV AX, DATA
                        MOV DS, AX
                        XOR AX, AX
                        MOV AL, N1
                        MOV BX, N3
                        AAD
                        DIV N2
                        MOV [BX], AL
                        MOV AH, 4CH
                        INT 21H
                        INT 3H
                  CODE ENDS
                  END START
```

Viva Questions:

1. What is the ASCII instruction, which is used before the arithmetic operation?
2. Why is ASCII adjust before division is done before actual division?
3. What does AL & AH contains?

Program No: 5(a) PACKED TO UNPACKED BCD

```
DATA SEGMENT
        N1    DB   56H, 49H, 33H
        N2    DB    06H  DUP(00)
DATA ENDS
      CODE SEGMENT
      ASSUME CS:CODE, DS:DATA

            START:   MOV AX, DATA
                     MOV DS, AX
                     XOR AX, AX
                     MOV SI, OFFSET N1
                     MOV DI, OFFSET N2
                     MOV CX, 0003H
            BACK:    MOV AL, [SI]
                     MOV BL, AL
                     AND AL, 0F0H
                     MOV CL, 04H
                     ROR BL, CL
                     AND BL, 0FH
                     MOV [DI], AL
                     INC DI
                     MOV [DI], BL
                     INC SI
                     INC DI
                     DEC CX
                     JNZ   BACK
                     MOV AH, 4CH
                     INT 21H
                     INT 3H
                CODE ENDS
                END START
```

RESULT:

I/P :	56H	49H	33H
O/P :	05H,06H,	04H,09H,	03H,03H

Viva Questions:

1. What is the purpose of the instruction ROR AL, CL?
2. What is the purpose of the instruction AND AL, 0FH .?
3. What is the expansion of UPBCD?
4. What is the use of DAA instruction?
5. What is the reason for packing unpacked BCD?
6. What is common between unpacked BCD and ASCII?

Program No: 5(b) UNPACKED BCD TO PACKED

```
DATA SEGMENT
        N1      DB   05H, 06H
DATA ENDS

        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

                START:      MOV AX, DATA
                            MOV DS, AX
                            XOR AX, AX
                            MOV AL, N1
                            MOV BL, AL
                            MOV CL, 04H
                            ROR BL, CL
                            OR AL, BL
                            MOV AH, 4CH
                            INT 21H
                            INT 3H
                CODE ENDS
                END START
```

RESULT:

I/P : 05H 06H

O/P : 56H

Viva Questions:

1. What are the flags effects in this program?
2. What is the purpose of XOR AX, AX ?
3. What is the use of INT 21H?
4. Why AX is called as accumulator?
5. How many segments & what is the use of Extra Segment?

Program No: 5(C) BCD to ASCII CONVERSION

```
DATA SEGMENT
        N1    DB   56H
        N2    DB   02H DUP (00)
DATA ENDS
    CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

            START:    MOV AX, DATA
                      MOV DS, AX
                      XOR AX, AX
                      MOV AL, N1
                     MOV SI,  OFFSET N2
                      MOV BL, AL
                      AND AL, 0F0H
                      ADD AL, 30H
                      MOV CL, 4H
                      ROR  BL, CL
                      AND BL, 0FH
                     ADD BL, 30H
                      MOV [SI] , BL
                      MOV AH, 4CH
                      INT 21H
                      INT 3H
                CODE ENDS
                END START
```

RESULT:

I/P : 56H
O/P: 35H,36H

Viva Questions:

1. What is the difference between adding 30h and OR 30H to a BCD number to conversion to ASCII?
2. Why unpacking is necessary during the conversion?
3. What is the ASCII character for symbol A?
4. What is the ASCII character for symbol zero '0'?
5. What is ROR instruction will do?

Program No: 5(d) ASCII to BCD CONVERSION

```
DATA SEGMENT
        N1      DB   35H
        N2      DB   02H DUP (00)
DATA ENDS
        CODE SEGMENT
        ASSUME CS: CODE, DS: DATA

              START:    MOV AX, DATA
                        MOV DS, AX
                        XOR AX, AX
                        MOV AL, N1
                        AND AL, 0FH
                        MOV AH, 4CH
                        INT 21H
                        INT 3H
                  CODE ENDS
                  END START
```

RESULT:

I/P : 35H
O/P : 05H

Viva Questions:

1. What is the use of DAA instruction?
2. What is the reason for packing & Un-Packed BCD?
3. What is the ASCII instruction, which is used before the airthematic operation?
4. What is the Expansion of BCDIP?
5. What is the need for segmentation?

Application Programs

Program No: 6 (a) EVEN AND ODD NUMBERS

```
DATA SEGMENT
        N1      DB   56H, 49H, 33H

DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

                START:  MOV AX, DATA
                        MOV DS, AX
                        XOR AX, AX
                        MOV SI, OFFSET N1
                        MOV DX, 0000H
                        MOV BX, 0000H
                        MOV CX, 0003H
                BACK:   MOV AL, [SI]
                        ROR AL, 01H
                        JC      X
                        INC BX
                        JMP Y
                X:      INC DX
                Y:      INC SI
                        DEC CX
                        JNZ   BACK
                        MOV AH, 4CH
                        INT 21H
                        INT 3H
                CODE ENDS
                END START
```

RESULT:

I/P : 56H 49H 33H
O/P : BX=0001H,DX=0002H

Viva Questions:

1. What is the function of Bx?
2. What are the branch instructions?
3. What is the difference between conditional and unconditional jump instructions?
4. What is the function of Ax in the program?
5. What is significance of accumulator?

Programm:6 (b) POSITIVE AND NEGATIVE NUMBER

```
DATA SEGMENT
        N1      DB   51H, 20H, 33H,80H,19H

DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

                START:  MOV AX, DATA
                        MOV DS, AX
                        XOR AX, AX
                        MOV SI, OFFSET N1
                        MOV DX, 0000H
                        MOV BX, 0000H
                        MOV CX, 0005H
                BACK:   MOV AL, [SI]
                        ROL AL, 01H
                        JC       X
                        INC BX
                        JMP Y
                X:      INC DX
                Y:      INC SI
                        DEC CX
                        JNZ    BACK
                        MOV AH, 4CH
                        INT 21H
                        INT 3H
                CODE ENDS
                END START
```

RESULT:

I/P : 56H 49H 33H

O/P : BX=0004H,DX=0001H

Viva Questions:

1. What is the function of ROL?
2. What are the branch instructions?
3. What is the difference between conditional and unconditional jump instructions?
4. What is the function of JC in the program?
5. What is significance of accumulator?

Program No: 7(a) ASCENDING ORDER

```
DATA SEGMENT
        N1      DB   56H, 49H, 33H,05H,12H,17H,08H

DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

                START:      MOV AX, DATA
                            MOV DS, AX
                            XOR AX, AX
                            MOV BX, 0006H
                Z:          MOV SI, OFFSET N1
                            MOV CX, 0006H
                BACK:       MOV AL, [SI]
                            INC SI
                            CMP AL,[SI]
                            JBE     Y
                            XCHG AL,[SI]
                            DEC SI
                            MOV [SI], AL
                            INC SI
                    Y:      DEC CX
                            JNZ   BACK
                            DEC BX
                            JNZ   Z
                            MOV AH, 4CH
                            INT 21H
                            INT 3H
                    CODE ENDS
                    END START
```

RESULT:

I/P : 56H, 49H, 33H,05H,12H,17H,08H
O/P : 05H, 08H,12H,17H, 33H, 49H, 56H,

Viva Questions:

1. What is the function of JBE ?
2. What is the need of CMP instructions?
3. What is the difference between conditional and unconditional jump instructions?
4. What is the function of XCHG in the program?
5. What is significance of accumulator?

Program No: 7(b) DESCENDING ORDER

```
DATA SEGMENT
        N1      DB   56H, 49H, 33H,05H,12H,17H,08H

DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

                START:      MOV AX, DATA
                            MOV DS, AX
                            XOR AX, AX
                            MOV BX, 0006H
                Z:          MOV SI, OFFSET N1
                            MOV CX, 0006H
                BACK:       MOV AL, [SI]
                            INC SI
                            CMP AL,[SI]
                            JAE         Y
                            XCHG AL,[SI]
                            DEC  SI
                            MOV  [SI],AL
                            INC SI
                Y:          DEC CX
                            JNZ    BACK
                            DEC BX
                            JNZ    Z
                            MOV AH, 4CH
                            INT 21H
                            INT 3H
                    CODE ENDS
                    END START
```

RESULT:

I/P : 56H, 49H, 33H,05H,12H,17H,08H
O/P : 56H, 49H,33H,17H,12H,08H,05H

Viva Questions:

1. What is the function of JAE?
2. What IS the need of CMP instructions?
3. What is the difference between conditional and unconditional jump instructions?
4. What is the function of XCHG in the program?
5. What is significance of accumulator?

Program No: 8 BLOCK TRANSFER

```
DATA SEGMENT
        N1  DB 01H,02H,03H
DATA ENDS
        EXTRA SEGMENT
        N2 DB 03H DUP (00)
        EXTRA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA, ES:EXTRA

              START:    MOV AX, DATA
                        MOV DS, AX
                        MOV AX, EXTRA
                        MOV ES, AX
                        MOV SI, OFFSET N1
                        MOV DI, OFFSET N2
                        CLD
                        MOV CX, 0003H
              REP       MOVSB
                        MOV AH, 4CH
                        INT 21H
                        INT 3H
                    CODE ENDS
                    END START
```

RESULT:

I/P : 01H, 02H, 03H
O/P : 01H, 02H, 03H

Viva Questions:

1. If the DF=1, will the SI and DI register decremented?
2. The destination memory is pointed by which register combination?
3. The source is pointed to by which register combination?
4. What is the purpose of instruction pointer?
5. What is the purpose of stack pointer?

Program No: 9(a) READ A CHARACTER WITH & WITH OUT ECHO

```
 ASSUME CS:CODE,

START :     MOV AH, 01H
            INT 21H
            MOV AH, 4CH
            INT 21H
            INT 3H
            CODE END
            END START
```

RESULT:

INPUT: ‘a’
OUPUT:’a’

```
ASSUME CS:CODE,

START :     MOV AH, 07H
            INT 21H
            MOV AH, 4CH
            INT 21H
            INT 3H
            CODE END
            END START
```

RESULT:

INPUT: ‘a’
OUPUT:-

Viva Questions:

1. What is the difference between a character with & without echo?
2. What is the use of MOV AH, 01H and MOV AH, 07H?
3. What are the general purposes registers?
4. What is the use of Extra segment?
5. What is the purpose of Instruction pointer?

Program No: 9(b) DISPLAY CHARACTER

ASSUME CS CODE:

```
START:  MOV AX,DATA
        MOV  DS, AX
        MOV CX, 0005H
   X :  MOV AH, 02H
        MOV DL, 'Z'
        INT 21H
        MOV AH, 02H
        MOV DL, 20H
        INT 21H
        DEC CX
        JNZ  X
        INT3H
        CODE END
        END START
```

RESULT:

INPUT : 'Z'
OUTPUT : Z Z Z Z Z

Viva Questions:

1. What is the need of Segments?
2. What is the difference between INT 21H and INT 3H?
3. What is the use of Base Pointer?
4. What is the use of Index Register?
5. What is the use of addressing modes?

Program No: 10 STRING REVERSAL

```
DATA SEGMENT
        N1   DB  'MLRIT$'
        N2   DB   05H DUP (00)
DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

              START:    MOV AX, DATA
                        MOV DS, AX
                        MOV SI, OFFSET N1
                        MOV DI, OFFSET N2
                        ADD  DI, 0005H
                        CLD
                         MOV  AH,  '$'
                    X:   CMP AH,  [SI]
                         JE      Y
                        MOV AL, [SI]
                        MOV [DI], AL
                        INC  SI
                        DEC DI
                        LOOP   X
                        MOV AH, 4CH
                        INT 21H
                      Y: INT 3H
                    CODE ENDS
                    END START
```

Viva Questions:

1. Why BX register is added with '5'?
2. Why MOVS instruction is not used?
3. What is the function of LODS and STOS instructions?
4. What is the use of segmentation?
5. What is the length of instruction pointer?

Program No: 11 STRING INSERTION

```
DATA SEGMENT
STRING1 DB 'EMPTY VESSELS MORE NOISE$'
STRLEN EQU ($-STRING1)
DATA ENDS

EXTRA SEGMENT
STRING2 DB STRLEN+5 DUP (0)
EXTRA ENDS

        CODE SEGMENT
        ASSUME CS: CODE, DS: DATA, ES: EXTRA

            START:  MOV AX, DATA
                    MOV DS, AX
                    MOV SI, OFFSET STRING1
                    MOV DI, OFFSET STRING2
                    CLD
                    MOV CX, 14
                    REP MOVSB
                    MOV DL, 5
            BACK:    MOV AH, 01
                    INT 21H
                    STOS STRING2
                    DEC DL
                    JNZ BACK
                    MOV CX, 11
                    REP MOVSB
                    NOP
                    MOV AH, 4CH
                    INT 21H
                    INT 3H
                CODE ENDS
                END START
```

Viva Questions:_

1. Why register 'DI' is loaded with 5?
2. What is the function of rep movsb?
3. What is the purpose of mov ah, 01h / int 21h?
4. What is meant flag register?
5. What is the use of SI and DI registers?

Program No: 12 STRING DELETION

```
.MODEL TINY

DATA SEGMENT
        STRING1 DB 'EMPTY VESSELS MAKE MORE NOISE$'
        STRLEN EQU ($-STRING1)
DATA ENDS

        EXTRA SEGMENT
        STRING2 DB STRLEN-5 DUP (0)
        EXTRA ENDS

        CODE SEGMENT
        ASSUME CS: CODE, DS: DATA, ES: EXTRA

                START:  MOV AX, DATA
                        MOV DS, AX
                        MOV AX, EXTRA
                        MOV ES, AX
                        MOV SI, OFFSET STRING1
                        MOV DI, OFFSET STRING2
                        CLD
                        MOV CX, 13
                        REP MOVSB
                        CLD
                        MOV SI, 18
                        MOV CX, 12
                        REP MOVSB
                        MOV AH, 4CH
                        INT 21H
                        INT 3H
                CODE ENDS
                END START
```

Viva Questions:_

1. What is the purpose of string length?
2. What does ‘equ’ stands for?
3. What is the purpose of label start after the end directive?

Program No: 13 *LENGTH OF THE STRING*

```
DATA SEGMENT
        STRING1 DB 'EMPTY VESSELS MAKE MORE NOISE$'
        STRLEN EQU ($-STRING1)
        RES DB 0
        CORT DB 'STRLENGTH FOUND CORRECT$'
        INCORT DB 'STRLENGTH FOUND INCORRECT$'
DATA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA

                START:  MOV AX, DATA
                        MOV DS, AX
                        SUB CL, CL
                        MOV BL, STRLEN
                        MOV SI, OFFSET STRING1
                BACK:   LODSB
                        INC CL
                        CMP AL,'$'
                        JNZ BACK
                        MOV RES, CL
                        CMP CL, BL
                        JZ CORRECT
                        MOV DX, OFFSET INCORT
                        MOV AH, 09
                        INT 21H
                CORRECT: MOV DX, OFFSET CORT
                        MOV AH, 09
                        INT 21H
                        MOV AH, 4CH
                        INT 21H
                        INT 3H
                CODE ENDS
                END START
```

Viva Questions:

1. What is the operation performed by the instruction cmp al,$?
2. What is function 09h / int 21h performed?
3. Why SI is not been incremented is the program?

***Program No: 14 STRING** COMPARISION*

```
DATA SEGMENT
        STRING1 DB 'EMPTY'
        STRLEN EQU ($-STRING1)
        NOTSFUL DB 'STRINGS ARE UNEQUAL$'
        SFUL DB 'STRINGS ARE EQUAL$'
DATA ENDS
        EXTRA SEGMENT
        STRING2 DB 'EMPTY'
        EXTRA ENDS
        CODE SEGMENT
        ASSUME CS:CODE, DS:DATA, ES:EXTRA

                START:      MOV AX, DATA
                            MOV DS, AX
                            MOV AX, EXTRA
                            MOV ES, AX
                            MOV SI, OFFSET STRING1
                            MOV DI, OFFSET STRING2
                            CLD
                            MOV CX,LENGTH STRING1
                            MOV CX, STRLEN
                            REP CMPSB
                            JZ FORW
                            MOV AH, 09H
                            MOV DX, OFFSET NOTSFUL
                            INT 21H
                            JMP EXITP
                            FORW:MOV AH,09H
                            MOV DX, OFFSET SFUL
                            INT 21H
                            EXITP:
                            NOP
                            MOV AH, 4CH
                            INT 21H
                            INT 3H
                        CODE ENDS
                        END START
```

Viva Questions:

1. What is the significance of CLD?
2. How does CMPSB perform the comparison?

Program No: 15 LEFT ENTRY DECODED MODE

```
DATA SEGMENT
CTRL      EQU   3002H     ;control word
DAT        EQU   3000H     ;data word
DATA    SEGMENT
TBL:  DB       3FH, 06H, 5BH,4FH
        DB       66H, 6DH, 7DH, 07H
DATA    ENDS
CODE SEGMENT
    ORG  0000: 4000H
    ASSUME CS: CODE  , DS:DATA
START:
    MOV      AL, 09H    ;LE 8 bit char. display
    MOV      DX, CTRL
    OUT      DX, AL
    MOV      AL, 31H     ;clock dividing factor
    OUT      DX, AL

LE:   MOV      AL, D0H     ; clear display
    OUT      DX, AL
    MOV      CX, 0FFFFH  ;wait till display is cleared
L1:   LOOP     L1
    MOV      AL, 01H     ;control word for 8 chars
    OUT      DX, AL      ;left entry mode. 8 no. of chars
    MOV      AH, 08      ;will be displayed
    MOV      BX, OFFSET TBL  ;char displayed are stored
    MOV      AL, 90H     ;control word for writing to
    OUT      DX, AL       ; display auto increment
RPT:
    MOV      AL  , [BX]
    MOV      DX , DAT
    OUT      DX, AL
    MOV      CX, 0FFFFH  ; delay
L2: LOOP     L2
    INC      BX
    DEC      AH
    JNZ      RPT
    MOV      CX, 0FFFFH
L3:   LOOP     L3
       INT      3
CODE    ENDS
END
```

Program No: 16 RIGHT ENTRY ENCODED MODE

```
DATA SEGMENT
CTRL      EQU   3002H     ;control word
DAT        EQU   3000H     ;data word
DATA     SEGMENT
      ORG    0000H:3000H
TBL:   DB       3FH,06H,5BH,4FH
         DB       66H,6DH,7DH,07H

DATA    ENDS
CODE   SEGMENT
    ORG 0000:4000H
    ASSUME CS:CODE ,DS:DATA

START:
    MOV      AL, 08H   ;RE 8  bit char. display
    MOV      DX, CTRL
    OUT      DX, AL
    MOV      AL, 31H   ;clock  dividing factor
    OUT      DX, AL

RE:  MOV      AL, D0H   ; display clear
    OUT      DX, AL
    MOV      CX, 0FFFFH
L1:  LOOP     L1
    MOV      AL, 10H    ; control word for rt entry mode
    OUT      DX, AL
    MOV      AH, 08
    MOV      BX, OFFSET TBL
    MOV      AL, 90H
    OUT      DX, AL
RPT:
    MOV      AL, [BX]
    MOV      DX, DAT
    OUT      DX, AL
    MOV      CX, 0FFFFH
L2:  LOOP     L2
    INC      BX
    DEC      AH
    JNZ      RPT
    MOV      CX, 0FFFFH
L3:  LOOP     L3
      INT    3
CODE    ENDS
END START
```

Program No: 17 8255 OPERATING IN STROBED OUTPUT MODE (MODE 1)

```
PORTA     EQU  3000H
PORTB     EQU  3002H
PORTC     EQU  3004H
CTLP_55   EQU  3006H
CMD59     EQU  FFD8H
DATA59    EQU  FFDAH
DATA            SEGMENT
       ORG     0:3000H
IBYTE:  DB      0
DATA            ENDS

CODE   SEGMENT
    ORG 0000:4000H

    ASSUME CS:CODE  ,DS:DATA

START:
     MOV AX,00H
     MOV SS,AX        ;Stack segment initialisation
     MOV SP,2000H

     CLI
     CLD

     MOV AX,00H       ;data segment initialisation
     MOV DS,AX

     MOV BX,202H
     PUSH CS
     POP AX           ;initialisation of interrupt vector
     MOV [BX],AX
     MOV BX,200H
     LEA AX,CS:SERVICE
     MOV [BX],AX

     MOV DX,CMD59    ;ICW1
     MOV AL,13H
     OUT DX,AL

     MOV DX,DATA59    ;ICW2(interrupt vector address)
     MOV AL,80H       ;initialisation of PIC 8259A
     OUT DX,AL
     MOV AL,0FH
     OUT DX,AL
```

```
    MOV AL,FEH
    OUT DX,AL

    MOV DX,CTLP_55    ;initialise 8255 in mode1.
    MOV AL,B4H        ;portA i/p and portB o/p
    OUT DX,AL

    MOV AL,05H        ;enable INTE flip flop which is
    MOV DX,CTLP_55    ;controlled by bit set/reset of PC2
    OUT DX,AL

    MOV AL,0AH
    OUT DX,AL         ;IBF(PC5) taken low
    MOV AL,0FH
    OUT DX,AL         ;OBF2*(PC7) taken high

    MOV   AL,IBYTE
    MOV      CL,AL
    MOV DX,PORTB      ;output data on portB
    OUT DX,AL

    STI
BACK1: MOV     DX,PORTC
       IN      AL,DX
       AND     AL,01
       CMP     AL,01
       JNZ     BACK1

BACK:  JMP BACK

SERVICE:
       MOV     AL,CL
      NOT AL           ;Complement the value and
      MOV CL,AL        ;output it to portB
      MOV DX,PORTB
      OUT DX,AL
RETURN:STI
      IRET

CODE   ENDS
END
```

Program No:18 *PROGRAM TO TEST 8255 IN MODE 2 (BIDIRECTIONAL INPUT MODE)*

```
PORTA     EQU  3000H        ;8255 PORTA address
PORTB     EQU  3002H        ;8255 PORTB address
PORTC     EQU  3004H        ;8255 PORTC address
CTLP_55   EQU  3006H        ;8255 CONTROL PORT address
DBDT      EQU  0F800:4F1FH   ;routine to display
CMD59     EQU  FFD8H
DATA59    EQU  FFDAH

DATA           SEGMENT
          ORG 0000:3000H
MSG   DB      '8255 in md2 I/P ',0h
DATA           ENDS

CODE   SEGMENT
     ORG 0000:4000H
     ASSUME CS:CODE ,DS:DATA

START:
      MOV AX,00H
      MOV SS,AX           ;stack segment initialisation
      MOV SP,2000H


      CLI
      CLD

      MOV AX,00H          ;data segment initialisation
      MOV DS,AX

      MOV BX,202H
      PUSH CS
      POP AX
      MOV [BX],AX         ;interrupt vector initialisation
      MOV BX,200H
      LEA AX,CS:SERVICE
      MOV [BX],AX

      MOV DX,CMD59    ;ICW1 H
      MOV AL,13H
      OUT DX,AL
      MOV DX,DATA59    ;ICW2(interrupt vector address)
      MOV AL,80H          ;initialisation of PIC 8259A
      OUT DX,AL
      MOV AL,0FH
```

```
    OUT DX,AL
    MOV AL,FEH
    OUT DX,AL

     MOV AL,C1H          ;initialise 8255 in mode 2
     MOV DX,CTLP_55      ;portA i/p & o/p,portC lower i/p.
     OUT DX,AL

     MOV AL,09H          ;enable INTE2 flip flop which is
     MOV DX,CTLP_55      ;controlled by bit set/reset of pc4
     OUT DX,AL

     STI                 ;enable interrupt

BACK: JMP BACK

;When STB* pulse is given by pressing the switch,STB* line
;goes low & IBF goes high & INTR line goes high and data is
;read by 8255.Then control jumps to interrupt service routine
;to perform read operation.O/P the same on portB and display.

SERVICE:
    MOV DX,PORTA
    IN AL,DX
    MOV CL,AL
    MOV DX,PORTB
    OUT DX,AL

    MOV CH,00H
    MOV SI,CX

    STI
    IRET

CODE   ENDS
END
```

Program No: 19 ***PROGRAM TO TEST 8255 IN MODE 2 (BIDIRECTIONAL OUTPUT MODE)***

```
PORTA    EQU  3000H
PORTB    EQU  3002H
PORTC    EQU  3004H
CTLP_55  EQU  3006H
CMD59    EQU  FFD8H
DATA59   EQU  FFDAH

DATA          SEGMENT
        ORG 0000:3000H
IBYTE   DB     0
DATA          ENDS

CODE  SEGMENT
   ORG 0000:4000H
   ASSUME CS:CODE  ,DS:DATA

START:
    MOV AX,00H
    MOV SS,AX        ;stack segment initialisation
    MOV SP,2000H


    CLI
    CLD

    MOV AX,00H       ;data segment initialisation
    MOV DS,AX

    MOV BX,202H
    PUSH CS
    POP AX
    MOV [BX],AX      ;interrupt vector initialisation
    MOV BX,200H
    LEA AX,CS:SERVICE
    MOV [BX],AX

    MOV DX,CMD59    ;ICW1
    MOV AL,13H
    OUT DX,AL
    MOV DX,DATA59    ;ICW2(interrupt vector address)
    MOV AL,80H       ;initialisation of PIC 8259A
    OUT DX,AL
    MOV AL,0FH
    OUT DX,AL
    MOV AL,FEH
```

```
    OUT DX,AL

    MOV AL,C1H       ;initialise 8255 in mode 2
    MOV DX,CTLP_55   ;port A i/p & o/p,port C lower i/p
    OUT DX,AL

    MOV AL,0DH       ;enable INTE flip flop
    MOV DX,CTLP_55
    OUT DX,AL

    MOV AL,03H       ;OBF1* taken high
    OUT DX,AL

    STI              ;enable interrupt

BACK:  JMP BACK

 SERVICE:

    MOV AL,IBYTE
    MOV DX,PORTA
    OUT DX,AL
    STI
    IRET

CODE  ENDS
END
```

Program No:20 *PROGRAM TO TEST 8255 IN MODE 1 (STROBED I/O MODE)*

```
PORTA     EQU  3000H
PORTB     EQU  3002H
PORTC     EQU  3004H
CTLP_55   EQU  3006H
CMD59     EQU  FFD8H
DATA59    EQU  FFDAH

DATA          SEGMENT
DATA          ENDS

CODE   SEGMENT

    ORG 0000:4000H

    ASSUME CS:CODE ,DS:DATA


START:
     MOV AX,00H      ;initialisation of stack pointer
     MOV SS,AX
     MOV SP,2000H

     CLI
     CLD

     MOV AX,00H      ;data segment initialisation
     MOV DS,AX

     MOV BX,202H     ;initalisation of interrupt vector
     PUSH CS
     POP AX
     MOV [BX],AX
     MOV BX,200H
     LEA AX,CS:SERVICE
     MOV [BX],AX

     MOV DX,CMD59    ;ICW1
     MOV AL,13H
     OUT DX,AL

     MOV DX,DATA59    ;ICW2(interrupt vector address)
     MOV AL,80H
     OUT DX,AL
```

```
    MOV AL,0FH
    OUT DX,AL       ;ICW4

    MOV AL,0FEH
    OUT DX,AL       ;OCW1(IR0 mask reset)

    MOV AL,0B4H     ;initialisation of 8255 for mode1
    MOV DX,CTLP_55  ;input operation
    OUT DX,AL

    MOV AL,09H      ;to enable INTE flip flop which is
    OUT DX,AL       ;controlled by bit set/reset of pc4

     MOV AL,03H     ;OBF1(PC1) is set
     OUT DX,AL

     MOV AL,0AH     ;IBF(PC5) is reset
     OUT DX,AL

     MOV AL,0FH     ;OBF2(PC7) is set
     OUT DX,AL
      MOV    DX,PORTC
BACK1:
      IN     AL,DX
      AND    AL,08
      CMP    AL,08
      JNZ    BACK1


      STI          ;enable interrupt bit

BACK:  JMP BACK

SERVICE:

     MOV DX,PORTA
     IN AL,DX
     MOV DX,PORTB
     OUT DX,AL
     STI
     IRET

CODE  ENDS
END
```

Program No:21 ***PROGRAM TO TEST 8255 IN MODE 0 (BASIC I/O)***

```
PORTA     EQU  3000H     ;8255 PORT A address
PORTB     EQU  3002H     ;8255 PORT B address
PORTC     EQU  3004H     ;8255 PORT C address
CTLP_55   EQU  3006H     ;8255 control port address

DATA          SEGMENT
DATA          ENDS

CODE   SEGMENT

    ORG 0000:4000H

    ASSUME CS:CSEG,DS:DSEG

    ; displaying message on LCD

START:
     MOV AL,90H       ;control word to initialise 8255
     MOV DX,CTLP_55   ;portA as i/p, portB and portC as o/p
     OUT DX,AL

LOOP1:
     MOV DX,PORTA
     IN AL,DX
     NOT AL
     MOV DX,PORTB
     OUT DX,AL

     MOV AL,02H       ;reset pc1 bit
     MOV DX,CTLP_55
     OUT DX,AL
     CALL DELAY

     MOV AL,03H       ;set pc1 bit
     MOV DX,CTLP_55
     OUT DX,AL
     CALL DELAY

     MOV AL,0AH       ;reset pc5 bit
     MOV DX,CTLP_55
     OUT DX,AL
     CALL DELAY

     MOV AL,0BH       ;set pc5 bit
```

```
    MOV DX,CTLP_55
    OUT DX,AL
    CALL DELAY

    MOV AL,0EH      ;reset pc7 bit
    MOV DX,CTLP_55
    OUT DX,AL
    CALL DELAY

    MOV AL,0FH      ;set pc7 bit
    MOV DX,CTLP_55
    OUT DX,AL
    CALL DELAY
;       MOV  DX,PORTA
;       IN   AL,DX
;       NOT  AL
                ;This program can be executed for
    JMP LOOP1       ;different switch settings

 DELAY:MOV CX,7FFFH
 NEXT:LOOP NEXT       ;loop for delay
     RET

CODE  ENDS
END
```

Program No:22 ***PROGRAM TO TEST 8251 RECEIVING PART***

```
CLOCK_FREQ EQU  1536000     ;8253 clock frequency
CTL_8251   EQU  3402H       ;8251 control port address
DATA_8251  EQU  3400H       ;8251 data port address
TMR1_8253  EQU  3002H       ;8253 timer1 address
CTL_8253   EQU  3006H       ;8253 control port address
EXT_RAM_LC EQU  0:FF00H     ;RAM location
DBDT       EQU  F800:4F1FH  ;routine for display on data field

CNT_BAUD_9600_MODE16 EQU 000AH
CNT_BAUD_4000_MODE01 EQU 0140H
CNT_BAUD_2400_MODE16 EQU 0028H
CNT_BAUD_1200_MODE64 EQU 0014H
CNT_BAUD_0300_MODE64 EQU 0050H

MODE_WORD16 EQU CEH
MODE_WORD1  EQU CDH
MODE_WORD64 EQU CFH

DATA   SEGMENT
   ORG 0000:3000H
DATA  ENDS

CODE   SEGMENT
    ORG 0000:4000H
    ASSUME CS:CODE ,DS:DATA

START:
    MOV AX,00H      ;initialisation of stack pointer
    MOV SS,AX
    MOV SP,2000H
    MOV DS,AX
    CLI
    CLD
    MOV BX,0202H     ;initalisation of interrupt vector
    PUSH CS
    POP AX
    MOV [BX],AX
    MOV BX,200H
    LEA AX,CS:SRVC2
    MOV [BX],AX

    MOV DX,FFD8H    ;ICW1
    MOV AL,13H
    OUT DX,AL
```

```
MOV DX,FFDAH    ;ICW2(interrupt vector address)
MOV AL,80H
OUT DX,AL

MOV AL,0FH
OUT DX,AL       ;ICW4

MOV AL,0FEH
OUT DX,AL       ;OCW1(IR0 mask reset)

MOV BX,EXT_RAM_LC   ;BX points to RAM location where the
                    ; Character read from 8251 is stored

MOV DX,CTL_8253     ;initialize timer1 in mode2
MOV AL,76H
OUT DX,AL

MOV DX,TMR1_8253
MOV AL,<CNT_BAUD_9600_MODE16    ;load LSB in count reg
OUT DX,AL
MOV AL,>CNT_BAUD_9600_MODE16    ;load MSB in count reg
OUT DX,AL

STI             ;enable interrupt

MOV DX,CTL_8251   ;send 0's to guarantee,device is in
MOV AL,00H        ;command instruction format before
OUT DX,AL         ;the RESET command is issued.

NOP
NOP
NOP
NOP
OUT DX,AL

NOP
NOP
NOP
NOP
OUT DX,AL

MOV DX,CTL_8251   ;send internal RESET command to
MOV AL,40H        ;return device to idle state
OUT DX,AL
NOP
NOP
NOP
NOP
```

```
    MOV DX,CTL_8251    ;load mode control word
    MOV AL,MODE_WORD16
    OUT DX,AL
    NOP
    NOP
    NOP
    NOP

    MOV DX,CTL_8251    ;load command word
    MOV AL,36H
    OUT DX,AL

BACK1: NOP
    JMP BACK1


SRVC2:
    MOV DX,DATA_8251
    IN AL,DX          ;In the service routine data is
    IN AL,DX          ;read from 8251,check whether typed
    NOP               ;character is 0DH.If yes it indicates
    NOP               ;it is the last char and is displayed
    NOP               ;in the data field of the display and
    NOP               ;reinitialise pointer to the starting
    CMP AL,0DH        ;address of RAM location.
    JNZ AHEAD2

    MOV AH,00
    MOV SI,AX
    CALL FAR DBDT
    MOV BX,EXT_RAM_LC
    JMP TERM

AHEAD2:MOV [BX],AL      ;If typed char is other than 0DH
     INC BX           ;then store the char in RAM loc.
TERM:  STI
     IRET
CODE   ENDS
END
```

Program No:23 TEST PROGRAM TO TEST THE 8259 ON THE 8086 UNI KIT

```
CLRDSP   EQU    0F800:4BB1H
OP       EQU    0F000:1000H
DATA   SEGMENT

     ORG    0000:3000H

MSG    DB     ' HELLO .....',0H
MSG1   DB     '  I GOT INTR 0  ',0H
MSG2   DB     '  I GOT INTR 1  ',0H
MSG3   DB     '  I GOT INTR 2  ',0H
MSG4   DB     '  I GOT INTR 3  ',0H
MSG5   DB     '  I GOT INTR 4  ',0H
MSG6   DB     '  I GOT INTR 5  ',0H
MSG7   DB     '  I GOT INTR 6  ',0H
MSG8   DB     '  I GOT INTR 7  ',0H

DSEG   ENDS

CODE   SEGMENT
     ASSUME  CS:CODE  ,DS:DATA

     ORG    0:4000H
START:
     PUSH   CS
     POP    ES
     CALL   FAR CLRDSP     ;
     MOV    AX,0000H
     MOV    DS,AX
     MOV    BX,0202H
     MOV    CX,08H
FILL_CS:
     MOV    WORD PTR [BX],00H
     ADD    BX,4
     LOOP   FILL_CS
     MOV    BX,0200H
     LEA    AX,CS:SERV1
     MOV    [BX],AX
     ADD    BX,4
     LEA    AX,CS:SERV2
     MOV    [BX],AX
     ADD    BX,4
     LEA    AX,CS:SERV3
     MOV    [BX],AX
     ADD    BX,4
     LEA    AX,CS:SERV4
```

```
        MOV     [BX],AX
        ADD     BX,4
        LEA     AX,CS:SERV5
        MOV     [BX],AX
        ADD     BX,4
        LEA     AX,CS:SERV6
        MOV     [BX],AX
        ADD     BX,4
        LEA     AX,CS:SERV7
        MOV     [BX],AX
        ADD     BX,4
        LEA     AX,CS:SERV8
        MOV     [BX],AX
        ADD     BX,4
        MOV     DX,03000H
        MOV     AL,13H
        OUT     DX,AL
        MOV     DX,03002H
        MOV     AL,80H
        OUT     DX,AL
        MOV     AL,0FH
        OUT     DX,AL
        MOV     AL,00H
        OUT     DX,AL
        MOV     DI,80H
        MOV     SI,OFFSET MSG
        CALL    FAR OP
        STI
BCK:    JMP     BCK

        ORG     0:6000H

SERV1:  MOV     DI,0C0H
        MOV     SI,OFFSET MSG1
        CALL    FAR OP
        STI
        IRET

        ORG     0:600DH
SERV2:  MOV     DI,0C0H
        MOV     SI,OFFSET MSG2
        CALL    FAR OP
        STI
        IRET

        ORG     0:601AH
SERV3:  MOV     DI,0C0H
        MOV     SI,OFFSET MSG3
```

```
        CALL    FAR OP
        STI
        IRET

        ORG     0:6027H
SERV4:  MOV     DI,0C0H
        MOV     SI,OFFSET MSG4
        CALL    FAR OP
        STI
        IRET

        ORG     0:6034H
SERV5:  MOV     DI,0C0H
        MOV     SI,OFFSET MSG5
        CALL    FAR OP
        STI
        IRET


        ORG     0:6041H

SERV6:  MOV     DI,0C0H
        MOV     SI,OFFSET MSG6
        CALL    FAR OP
        STI
        IRET

        ORG     0:604EH
SERV7:  MOV     DI,0C0H
        MOV     SI,OFFSET MSG7
        CALL    FAR OP
        STI
        IRET

        ORG     0:605BH
SERV8:  MOV     DI,0C0H
        MOV     SI,OFFSET MSG8
        CALL    FAR OP
        STI
        IRET

CSEG    ENDS
        ENDS
```

COMMAND LINE (DOS):

1. **Edit the program using 'EDIT' editor**
2. **Save as name .A51**
3. **Assemble using A51.EXE**
   C: A51 name .A51
   Result: name .LST, name .OBJ
4. **LINK & Convert to Hex format using OH51.EXE**
   C: OH51 name .OBJ
   Result: name .HEX
5. **Simulate Using SIM31.EXE**

C: SIM31

Using Integrated Design Environment (IDE)
KEIL µVision – (Windows)

1. **Select START\PROGRAMS\KEIL <Left Click>**
2. **Create New project**
   Select PROJECT\NEW\<Left Click> "enter name'<Left Click>
   Choose TARGET\INTEL\8051 <Left Click>
   Choose BUILD OOPTIONS
   Check Box HEX <OK><Left Click>
3. **Create File <ALT><F><N> 'name of file .ASM' <Left Click>**
   Edit File & Save
4. **SELECT PROJECT WINDOW – ADD FILE – SELECT - <Left Click>**
5. **PROJECT\BUILD\ <Left Click>**
   If no errors go to 7 else 3
6. **DEBUG\START\ <Left Click>**

PROGRAM (a): TRANSFERRING A BLOCK OF DATA FROM INTERNAL ROM TO INTERNAL RAM

```
                ORG 0000H
                LJMP START
                ORG 000BH
                RETI
                ORG 0013H
                RETI
                ORG 001BH
                RETI
                ORG 0023H
                RETI
                ORG 0023H
                RETI
                ORG 0026H
                RETI
                ORG 200H
START:          MOV DPTR,#MESSAGE
                MOV R0,#20H
                MOV R7,#11
REP:            MOV A,#00H
                MOVC A,@A+DPTR
                MOV @R0,A
                INC R0
                INC DPTR
                DJNZ R7,REP

MESSAGE: DB "HELLO WORLD"
                END
```

Viva Questions:_

1. What are the two registers in 8051, which are used for indirect addressing?
2. What are the 16 bit registers available in 8051?
3. Which register cannot be decremented?
4. 8051 is a ------- bit microcontroller?

PROGRAM: (b) ***INTERNAL ROM TO EXTERNAL RAM***

```
        ORG 0000H

START:  MOV A,#00
        MOV SP,#07H
        MOV R3,#11
        MOV DPTR,#200H
        PUSH DPL
        PUSH DPH
REP:    MOV DPTR,#MESSAGE
        MOV R2,A
        MOVC A,@A+DPTR
        POP DPH
        POP DPL
        MOVX @DPTR,A
        INC DPTR
        PUSH DPL
        PUSH DPH
        INC R2
        MOV A,R2
        DJNZ R3,REP
MESSAGE: DB "HELLO WORD"
        END
```

Viva Questions:_

1. What does “jnz” instruction do?
2. What is the purpose of DPTR ?
3. How many 8 bit registers are available in 8051?
4. What is the purpose of R2?
5. What is the purpose of R3?

PROGRAM:(c) ***INTERNAL ROM TO EXTERNAL RAM***

```
                ORG 200H

START:          MOV R0,#40H
                MOV A,#30H
                MOV R6,#10

REP:            MOV @R0,A
                INC A
                INC R0
                DJNZ R6,REP
                MOV R1,#40H
                MOV DPTR,#400H
                MOV R7,#00H

REP1:           MOV A,@R0
                MOVX @DPTR,A
                INC DPTR
                INC R1
                INC R7
                CJNE R7,#10,REP1

                END
```

Viva Questions:_

1. What is DPTR relative addressing?
2. What is the purpose of R7?
3. What is the purpose of R0?
4. What does "jnz" instruction do?
5. The stack pointer of 8051 grows upwards or downwards?

PROGRAM: (d) ***UNDERSTANDING THREE MEMORY AREAS OF 00-FF***

```
        ORG 0000H
        LJMP START

        ORG 200H

START:  MOV A,#41H
        MOV 20H,A
        MOV A,#42H
        MOV R0,#20H
        MOV @R0,A
        MOV A,#43H
        MOV 80H,A
        MOV A,#44H
        MOV R0,#80H
        MOV @R0,A

        END
```

Viva Questions:_

1. What is the instruction used for loading accumulator from code memory?
2. What is the purpose of register R0?
3. What are the two registers in 8051 which are used for indirect addressing?