NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

Department of Computer Engineering

II Year / IV Semester

20CEL47A – MICROPROCESSORS AND INTERFACING LABORATORY

LABORATORY MANUAL

Academic Year : 2021 - 2022
Semester : EVEN

NEW HORIZON COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

Department of Computer Engineering

MICROPROCESSORS AND INTERFACING LAB

Course Code	**: 20CEL47A**	**Credits**	**: 2**
L:T:P	**: 0:0:2**	**CIE Marks**	**: 25**
Exam Hours	**: 03**	**SEE Marks**	**: 25**

LIST OF EXPERIMENTS

1. Byte and word data transfer in different addressing modes.

2. Write an assembly level programs for basic arithmetic operations using 8086
   (i) Signed and Unsigned Addition
   (ii) Subtraction
   (iii) Signed and Unsigned Multiplication
   (iv) Signed and Unsigned Division

3. Write an assembly level programs assembly level programs for basic logical operation using 8086
   (i) To check number is positive or negative
   (ii) To count number of one's & zero's

4. Write an assembly Level programs for code conversion of 8086

(i) ASCII to binary
(ii) Decimal to Hex
(iii) ASCII to Decimal
(iv) Binary to BCD and vice versa

5. Write an assembly level programs for String operations using 8086
   (i) Reverse the string
   (ii) To check whether the string is palindrome or not

6. Write an assembly level program using 8086 for sorting operations like ascending, descending, largest and smallest in microprocessor

7. Interfacing of 8086 with (Assembly Level Programming)
   (i) Stepper motor
   (ii) Seven segment Display
   (iii) Logic controller (BCD up counter and Down counter)
   (iv) Keyboard Display Interface

LAB-IN-CHARGE **HOD-CE**

MICROPROCESSORS AND INTERFACING LAB

Course Code	**: 20CEL47A**	**Credits**	**: 2**
L: T:P	**: 0:0:2**	**CIE Marks**	**: 25**
Exam Hours	**: 03**	**SEE Marks**	**: 25**

Expt. No	Topics	Course Outcomes
1	1. Byte and word data transfer in different addressing modes. 2.Write an assembly level programs for basic arithmetic operations using 8086 (i) Signed and Unsigned Addition (ii) Subtraction (iii) Signed and Unsigned Multiplication (iv) Signed and Unsigned Division 3. Write an assembly level programs assembly level programs for basic logical operation using 8086 (i) To check number is positive or negative (ii) To count number of one's & zero's	CO1
2	4.Write an assembly Level programs for code conversion of 8086 (i) ASCII to binary; (ii) Decimal to Hex; (iii) ASCII to Decimal; (iv) Binary to BCD and vice versa	CO2
3	5.Write an assembly level program for String operations using 8086 (i) Reverse the string (ii) To check whether the string is palindrome or not. 6.Write an assembly level program using 8086 for sorting operations like ascending, descending, largest and smallest in microprocessor	CO3
4	7.Interfacing of 8086 with (Assembly Level Programming) (i) Stepper motor (ii) Seven segment Display (iii)Logic controller (BCD up counter and Down counter) (iv)Keyboard Display Interface	CO4

CO1	Write assembly level programs using 8086 to perform arithmetic and logical operations.
CO2	Apply the knowledge of computer number system to write code conversion programs in 8086.
CO3	Develop assembly code for string operations, sorting of numbers and branch instructions of 8086.
CO4	Demonstrate the I/O interfacing of 8086 with peripheral devices.

INDEX

5.2	Write an ALP to check whether the given string is Palindrome or not		
6	**PROGRAMS INVOLVING SORTING OPERATIONS**		
6.1	Write an ALP to find the largest number in an array		
6.2	Write an ALP to find the Smallest number in an array		
6.3	Write an ALP to sort the given array in an Ascending order.		
6.4	Write an ALP to sort the given array in an Descending order.		
7	**INTERFACING WITH 8086 MICROPROCESSOR**		
7.1	Write an ALP to illustrate Logic Controller		
7.2	Write an ALP to illustrate Seven Segment Display		
7.3	Write an ALP to illustrate Keyboard Interface		

Scheme of Evaluation

Evaluation Criteria	**Weightage**
CIE – I	15 Marks
CIE – II (2 – Lab Internals – 25 Marks Each)	10 Marks
SEE	25 Marks

Prepared By	**Verified By**	**Approved By**
DR.C.R.RATISH		
LAB-IN-CHARGE		HOD-CE

INTRODUCTION TO 8086 MICROPROCESSOR

8086 Internal Block diagram

8086 is a 16-bit processor having 16-bit data bus and 20-bit address bus. The block diagram of 8086 is as shown. (Refer figures 1A & 1B). This can be subdivided into two parts; the Bus Interface Unit (BIU) and Execution Unit (EU).

Bus Interface Unit:

The BIU consists of segment registers, an adder to generate 20 bit address and instruction pre fetch queue. It is responsible for all the external bus operations like opcode fetch, mem read, mem write, I/O read/write etc. Once this address is sent OUT of BIU, the instruction and data bytes are fetched from memory and they fill a 6-byte First in First out (FIFO) queue.

Execution Unit:

The execution unit consists of: General purpose (scratch pad) registers AX, BX, CX and DX; Pointer registers SP (Stack Pointer) and BP (Base Pointer); index registers source index (SI) & destination index (DI) registers; the Flag register, the ALU to perform operations and a control unit with associated internal bus. The 16-bit scratch pad registers can be split into two 8-bit registers. AX □ AL, AH ; BX □ BL, BH; CX □ CL, CH; DX □ DL, DH.

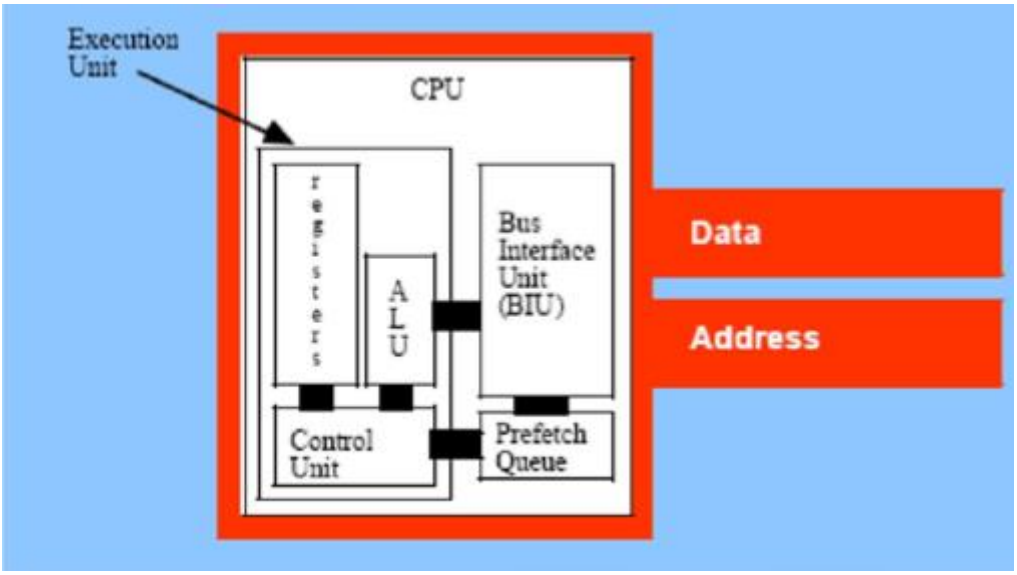


Figure 1A

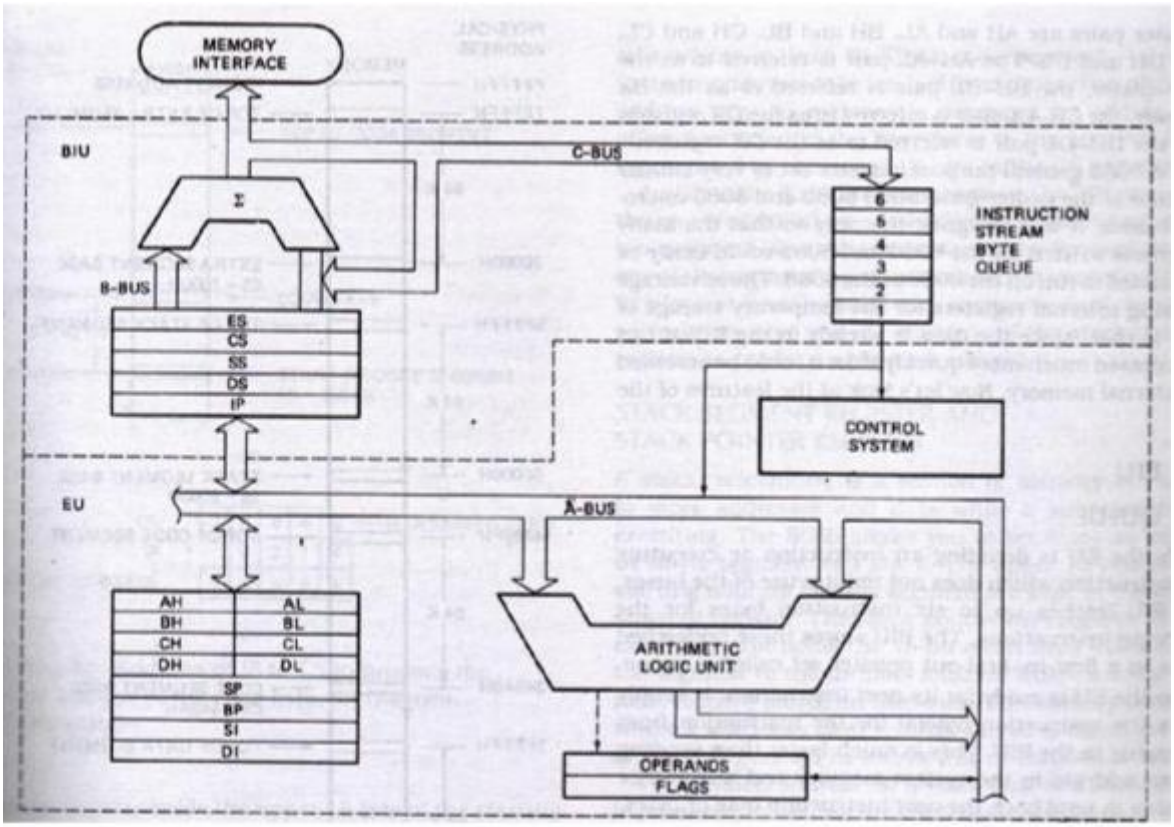
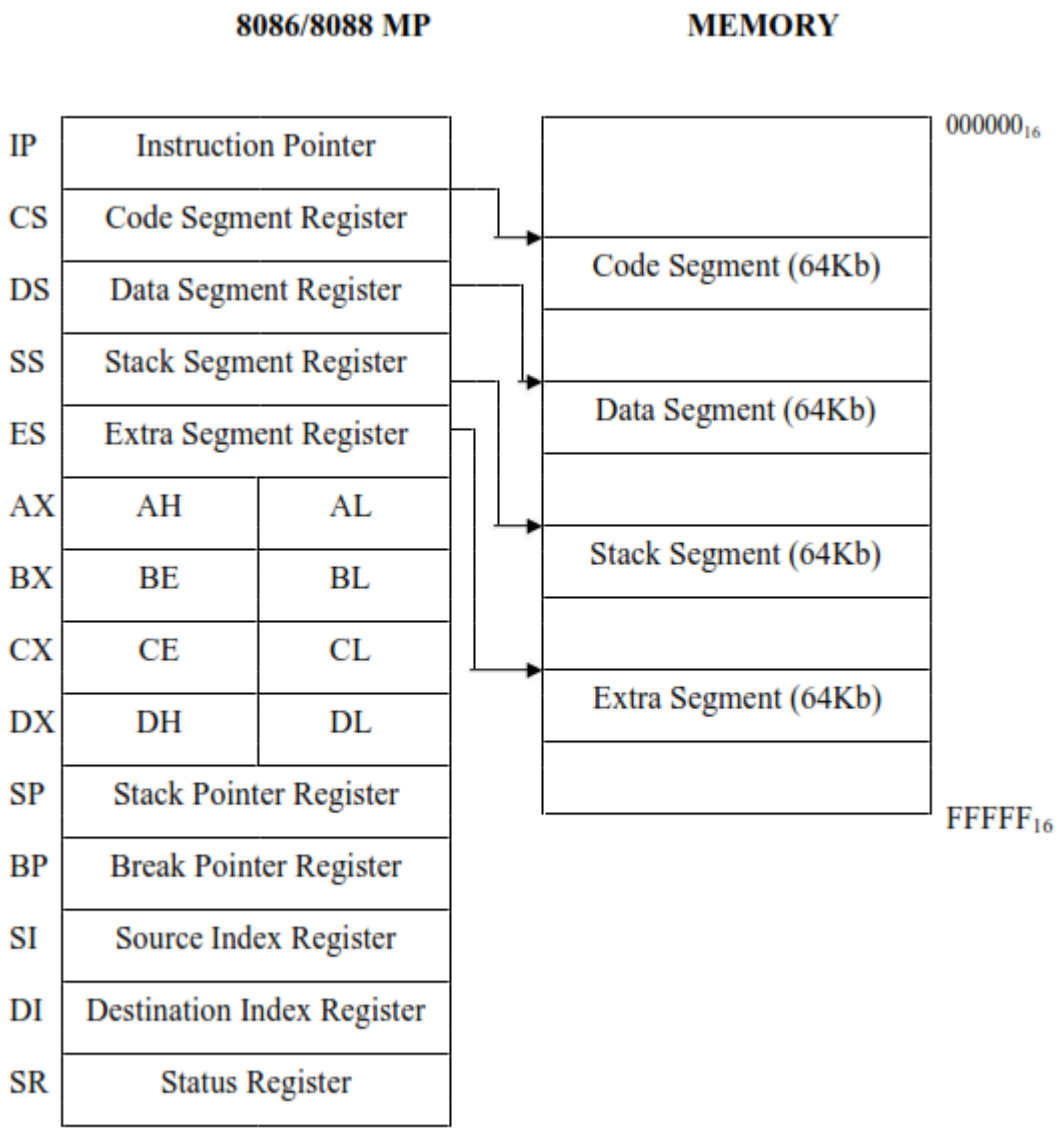


Figure 1 B

Different registers and their operations are listed below:

Register	Uses/Operations
AX	As accumulator in Word multiply & Word divide operations, Word I/O operations
AL	As accumulator in Byte Multiply, Byte Divide, Byte I/O, translate, Decimal Arithmetic
AH	Byte Multiply, Byte Divide
BX	As Base register to hold the address of memory
CX	String Operations, as counter in Loops
CL	As counter in Variable Shift and Rotate operations
DX	Word Multiply, word Divide, Indirect I/O



Execution of Instructions in 8086:

The microprocessor sends OUT a 20-bit physical address to the memory and fetches the first instruction of a program from the memory. Subsequent addresses are sent OUT and the queue is filled up to 6 bytes. The instructions are decoded and further data (if necessary) are fetched from memory. After the execution of the instruction, the results may go back to memory or to the output peripheral devices as the case may be.

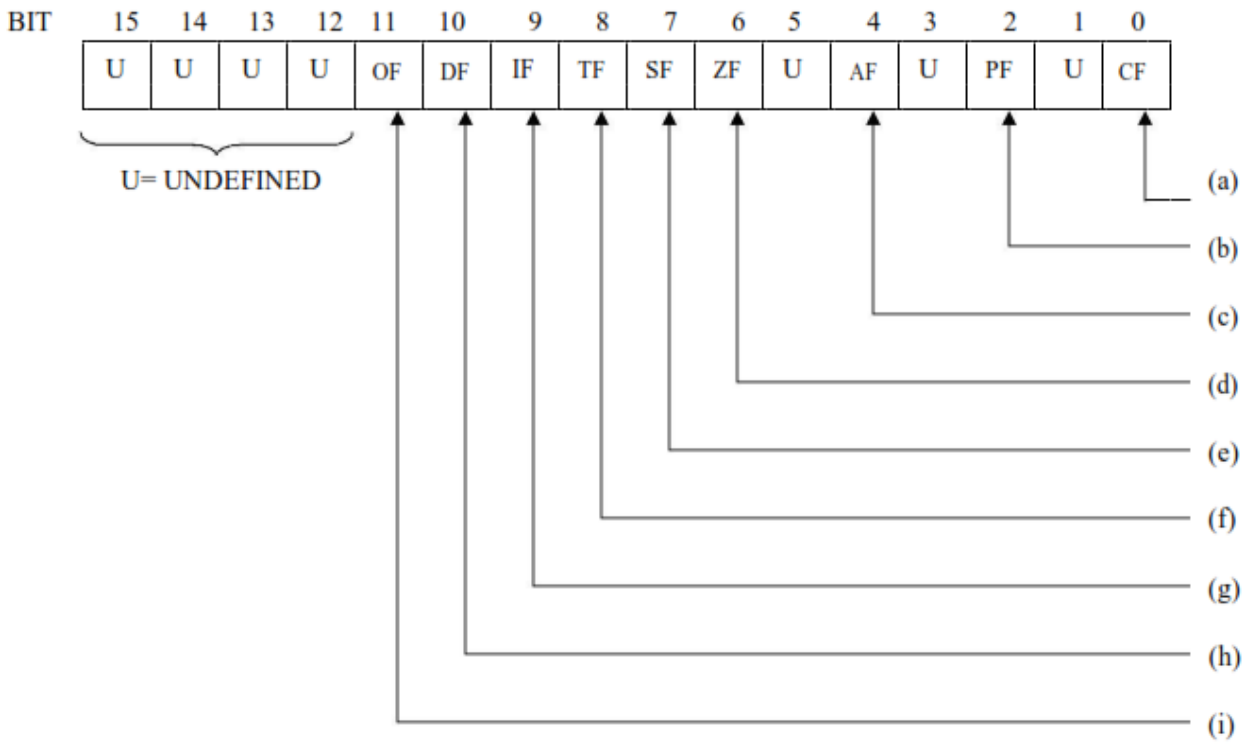


Fig C: 8086 Flag Register Format

(a) : CARRY FLAG – SET BY CARRY OUT OF MSB
(b) : PARITY FLAG – SET IF RESULT HAS EVEN PARITY
(c) : AUXILIARY CARRY FLAG FOR BCD
(d) : ZERO FLAG – SET IF RESULT = 0
(e) : SIGN FLAG = MSB OF RESULT
(f) : SINGLE STEP TRAP FLAG
(g) : INTERRUPT ENABLE FLAG
(h) : STRING DIRECTION FLAG
(i) : OVERFLOW FLAG

TUTORIALS - Creating source code

The source code consists of 8086/8088 program memories, appropriate pseudo-Opcodes and assembler directives. The first is created with a text editor and is given an extension ASM. The text editor may be any word processor that can produce standard ASCII code.

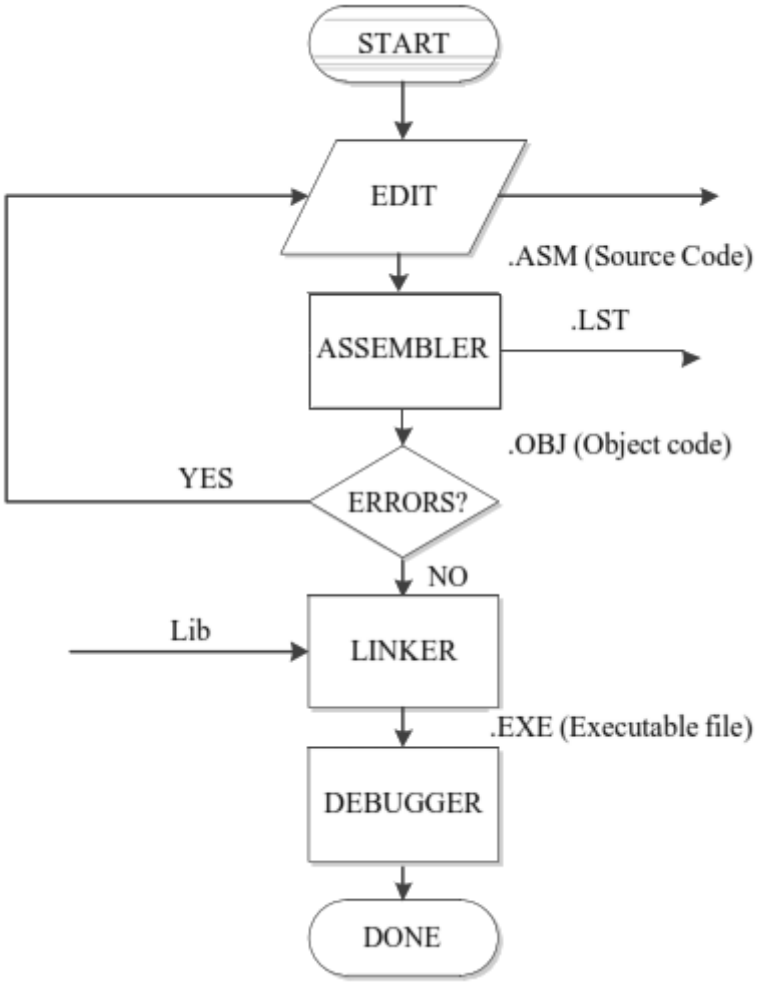


Fig Flow Chart of Creation and Execution of Program in MASM

Assembling the program

To assemble the program two assemblers are available. They are:

- **Microsoft Macro Assembler (MASM) and**
- **Borland Turbo Assembler (TASM).**

Besides doing the tedious task of producing the binary codes for the instruction statements, an assembler also allows the user to refer to data items by name rather by numerical addresses. This makes the program much more readable. In addition to program instructions, the source program contains directives to the assembler. Pseudo instructions are assembler directives entered into the source code along with the assembly language. Once the program written completely, it can be assembled to obtain the OBJ file by executing MASM. The assembly language program file name should be mentioned along with the command.

MASM<file name.ASM>

- The <file name.ASM> file that contains the assembly language program is assembled.
- The assembler generates error messages if there are any error (Syntax errors).
- These errors are listed along with the line number. If there are no errors then .OBJ file is created. To obtain the .EXE file the user has to LINK the .OBJ file.

LINK <file name>; or TLINK <file name>;

If a file is smaller than 64K bytes it, can be converted from an execution file to a command file (.COM). The command file is slightly different from an execution file (.EXE). In a command file the program must be originated at location 100H before it can execute. This means that the program must be no longer than (64K-100H) in length. The command file requires less space in memory than the equivalent execution file. The system loads .COM file off the disk into the computer memory more quickly than the execution file. To create a .COM file from a .EXE file, we need the EXE2BIN converter EXE2BIN converts .EXE file to .COM or binary file.

Example: EXE2BIN <filename><file name.com>

The <filename> with an EXE extension is converted to <filename> with .com extension with the above command.

Test and Debug

The executable program can be run under DOS or DUBUG. As a thumb rule a program under DOS only when there is no error or it produces some not visible or audible result. If the

program result is stored in registers or in memory, the result is visible. Hence it should be run using DEBUG or TD (Turbo Debugger) or code-view only. .EXE file can be loaded into memory using DEBUG.

Example: DEBUG<filename.EXE>

Using DEBUG, it is possible to find the bugs in the program. After loading it into the memory it is possible to check and correct the errors using different commands in DEBUG. Some of the commands are as follows:

G-GO

Format: G[offset][, offset]

Action: Executes a program starting at the current location offset values are temporary breakpoints. Upon encounter of a breakpoint instruction the processor stops and displays registers and flag contents.

T – TRACE

Format: T [Instruction count]

Action: Executes one or more instructions and displays register and flag values for each of them.

Example: T: Executes only the next instructions

T5: Executes the next 5 instructions

P- PTRACE

Format: P [instruction count]

Action: Same as Trace, but treats subroutine calls, interrupts, loop instructions, and repeat

String instructions as a single instruction

Q-QUIT

Format: Q

Action: Exists to dos.

A-Assemble

Format: A<CS: offset>

Action: This command allows us to enter the assembler mnemonics directly.

U- Unassemble

Format: U<CS: offset>

Action: This command lists a program from the memory. The memory start location is specified by CS: offset.

User can use code view to debug the program by following the steps given below:

Write the program in a file with .ASM extension using an editor [PRETEXT Editor which saves it in ASCII].

Ex: EDIT TEST1.ASM

Assemble the program using the command MASM/ZI file name;

Ex: MASM TEST1.ASM

Link the program using the command LINK/CO file name;

Ex: LINK TEST1.OBJ

To debug use

DEBUG FILENAME.EXE

EXPERIMENT: 1

BYTE AND WORD DATA TRANSFER IN DIFFERENT ADDRESSING MODES

EXPERIMENT: 1

AIM:

To transfer bytes and word data different addressing modes in a 8086 Microprocessor using Assembly Level language.

SOFTWARE REQUIRED:

MASM Assembler

PROGRAM

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    ORG 1000H
    MSGS DB 02H,04H,05H,0AH
    ORG 2000H
    MSGR DB " "
DATA ENDS
CODE SEGMENT
    MAIN:MOV AX,DATA
    MOV DS,AX
    MOV DX,077AH
    MOV SI,OFFSET MSGS
    MOV DI,OFFSET MSGR
    MOV CX,04H
  L: MOV AL,[SI]
    MOV [DI],AL
    INC SI
    INC DI
    DEC CX
    JNZ L
```

```
    INT 03H
CODE ENDS
    END MAIN
```

Input: **1000H: 02H,04H,05H,0AH**

Output: **2000H: 02H,04H,05H,0AH**

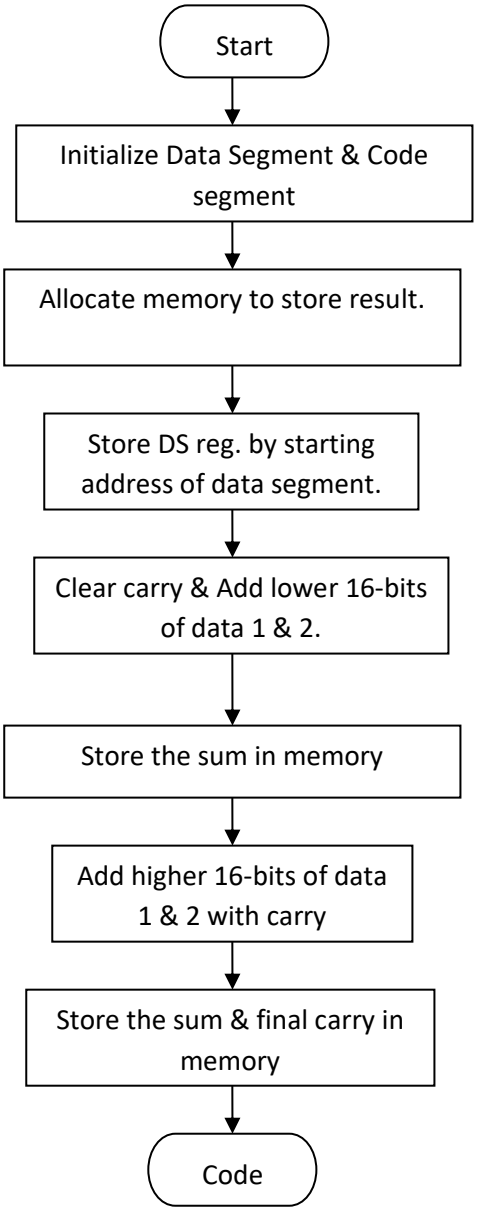
RESULT: The program for byte and word data transfer in different addressing modes is executed and the output is verified.

EXPERIMENT: 2

ARITHEMETIC OPERATIONS USING 8086

2.1 Addition of two 32-bit numbers

Flow Chart:



EXPERIMENT: 2

2.1 Addition of two 32-bit numbers

AIM:

To write a ALP to add two 32 bit numbers and verify the same using MASM

SOFTWARE REQUIRED:

MASM Assembler

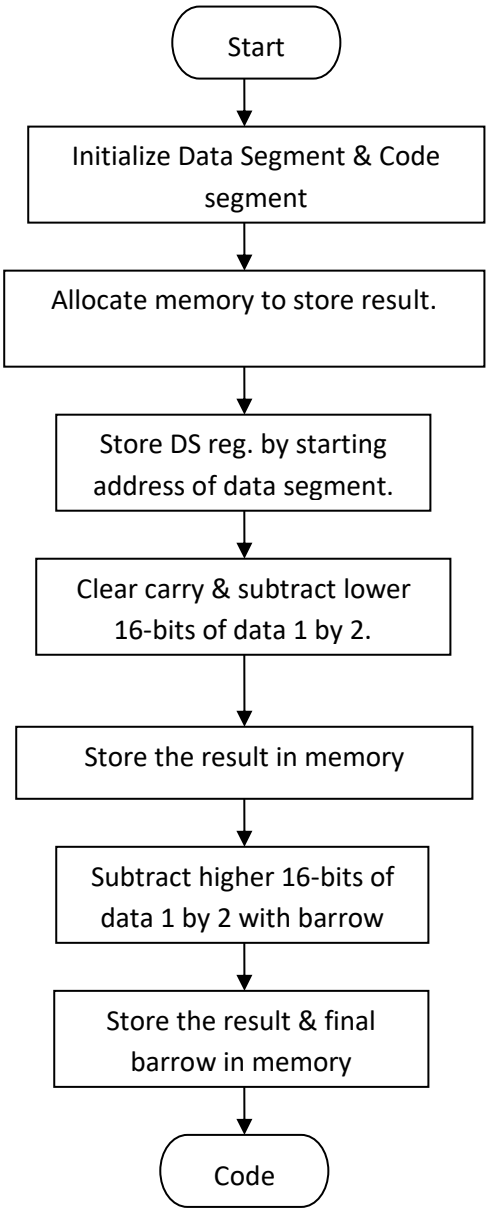
PROGRAM

```
data segment
num1 dw 1234h, 5678h
 num2 dw 0ab12h, 0cdefh
result dw 3 dup(?)
data ends

code segment
        assume cs:code,ds:data
        main:mov ax,data
        mov ds,ax
        mov dx,00h
        mov ax, num1
        mov bx, num2
clc
        add ax, bx
        mov result, ax
mov ax, num1+2
        mov bx, num2+2
adc  ax, bx
        mov result+2, ax
        adc  dx, 00h
        mov result+4, dx
        mov ah, 4ch
        int 21h
        code ends
        end main
```

Input:	**num1: 5678 1234H**	**num1: 1122 3344**
	num2: CDEF AB12H	**num2: 55667788**
Output:	**0001 2467 BD46H**	**0000 6688AACC**

RESULT: The program for addition of two 32-bit numbers is executed and the output is verified.

2.2. Subtraction of two 32-bit numbers.



2.2. Write an ALP for Subtraction of two 32-bit numbers.

Software Required: MASM Assembler

```
data segment
n1      dw 0ba98h, 0fedch
 n2      dw 5678h, 1234h
result dw 3 dup(?)
data ends

code segment
      assume cs:code,ds:data
      main:mov ax,data
      mov ds,ax
      mov dx, 00h
      mov ax, n1
      mov bx, n2
clc
      sub  ax, bx
      mov result, ax
mov ax, n1+2
      mov bx, n2+2
sbb  ax, bx
      mov result+2, ax
      sbb  dx, 00h
      mov result+4, dx
      mov ah, 4ch
      int 21h
      code ends
      end main
```

Input: **n1: FEDC BA98H**
n2:1234 5678H

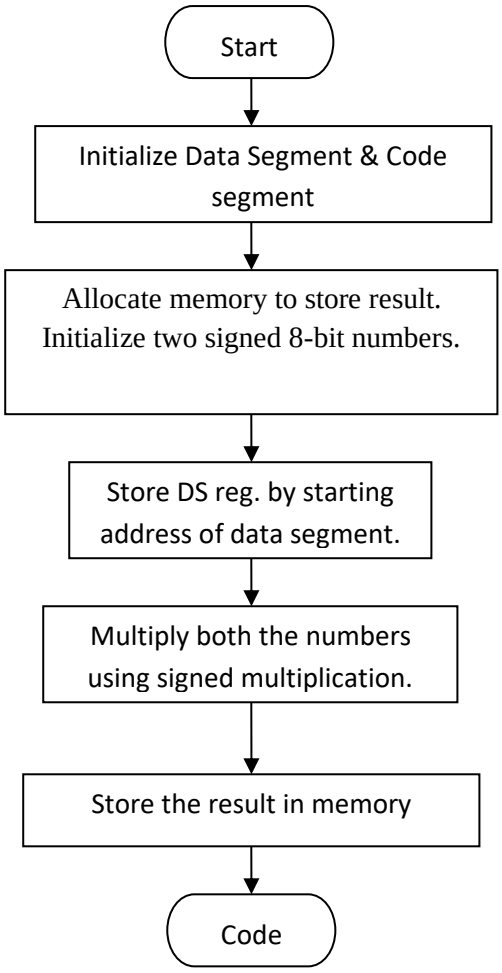
Output: **0000 ECA8 6420H**

Input: **n1:F234 6678H**
n2:F780 5384H

Output: **FFFF FAB4 12F4H**

RESULT: The program for Subtraction of two 32-bit numbers is executed and the output is verified.

2.3.1. Multiplication of two signed 8-bit numbers.



Multiplication of two 8-bit signed numbers

2.3.1. Aim: Write an ALP for Multiplication of two signed 8-bit numbers.

Software Required: MASM Assembler

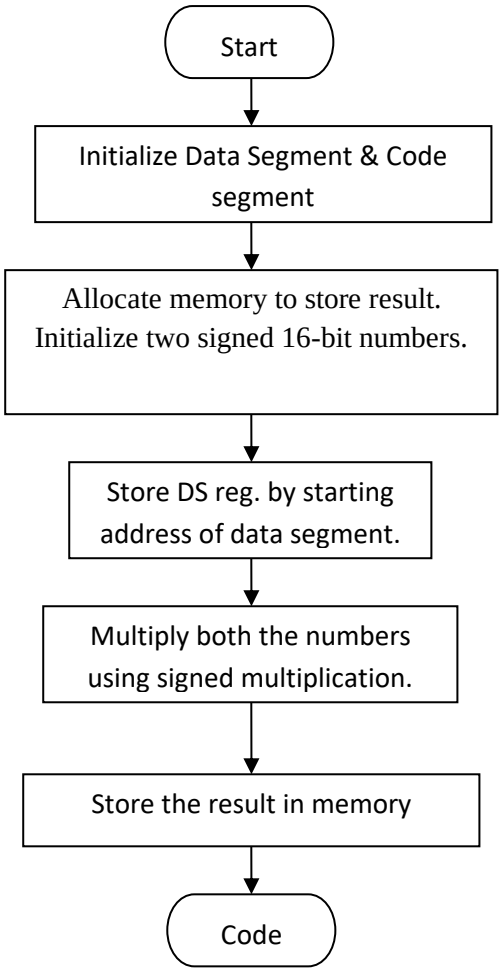
```
data segment
n1      db   0f4h
n2      db   12H
result dw  2dup(?)
data ends

code segment
      assume cs:code,ds:data
      main:mov ax,data
      mov ds,ax
mov al, n1
      mov bl, n2
imul bl
      mov result, ax
mov ah, 4ch
      int 21h
      code ends
      end main
```

Input: n1=F4H n1= D2H
n2=12H n2= 46H

Output: 1128H 396CH

RESULT: The program for multiplication of two signed 8 bit numbers is executed and the output is verified.

2.3.2. Multiplication of two signed 16-bit numbers.



Multiplication of two 16-bit signed numbers

2.3.2. Aim: Write an ALP for Multiplication of two signed 16-bit numbers.

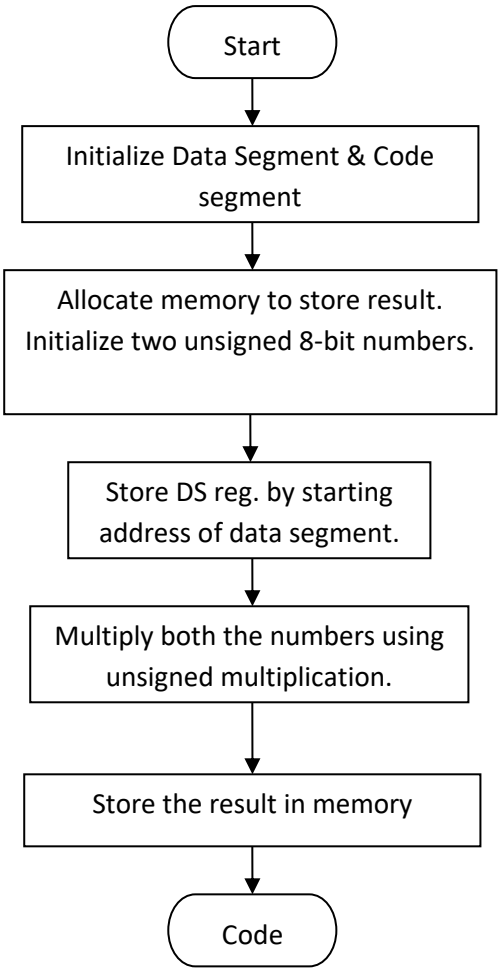
Software Required: MASM Assembler

```
Data segment
n1      dw  1234h
 n2      dw  ff40h
       result  dw (?)
data ends

code segment

       assume cs:code,ds:data
       main:mov ax,data
       mov ds,ax
mov ax, n1
       mov bx, n2
imul bx
       mov result, ax
       mov result+2, dx
mov ah, 4ch
        int 21h
       code ends
       end main
```

Input: n1=1234h n1= 1122 H
n2=ff40H n2= 3344H
Output: 12265900 36E5308H

RESULT: The program for multiplication of two signed 16-bit numbers is executed and the output is verified.

2.3.3. Multiplication of two unsigned 8-bit numbers.



2.3.3. Aim: Write an ALP for Multiplication of two unsigned 8-bit numbers.

Software Required: MASM Assembler

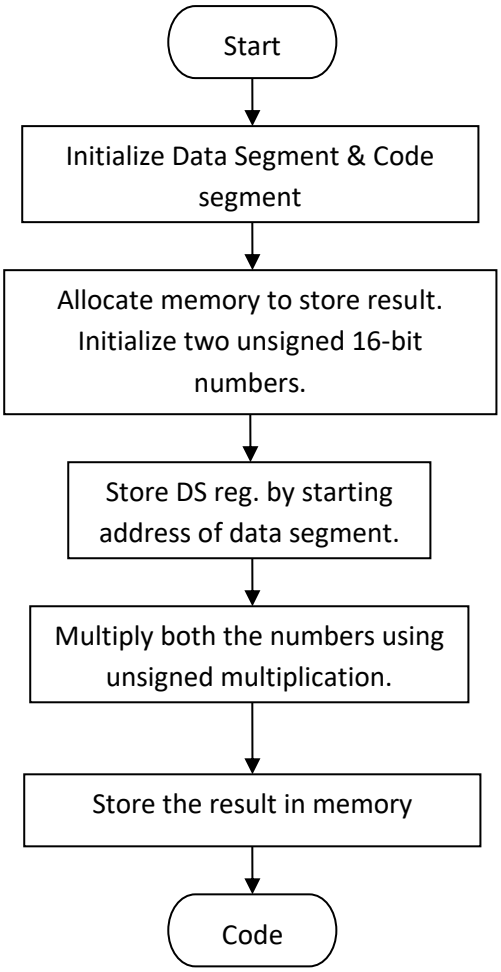
PROGRAM:

```
data segment
n1      db   0ffh
 n2      db   0fh
result  dw  ?
data ends

code segment
      assume cs:code,ds:data
      main:mov ax,data
      mov ds,ax
mov al, n1
      mov bl, n2
mul bl
      mov result, ax
mov ah, 4ch
      int 21h
      code ends
      end main
```

Input: n1=0ffH n1= 12H
n2=0Fh n2= 34H
Output: 0EF1H 3A8H

RESULT: The program for multiplication of two unsigned 8-bit numbers is executed and the output is verified.

2.3.4 Multiplication of two unsigned 16-bit numbers.



Multiplication of two 16-bit unsigned numbers

2.3.4 Aim: Write an ALP for Multiplication of two unsigned 16-bit numbers

Software Required: MASM Assembler

PROGRAM

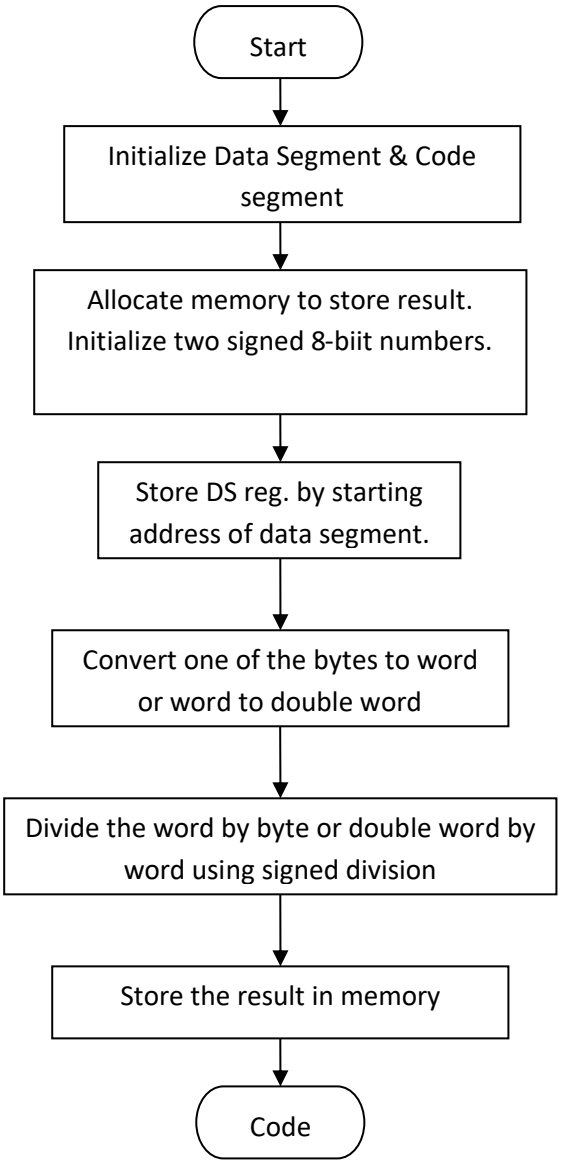
```
data segment
 n1      dw  1234h
 n2      dw  0ff40h
  result  dw ?, ?
data ends
code segment
      assume cs:code,ds:data
      main:mov ax,data
      mov ds,ax
mov ax, n1
      mov bx, n2
mul bx
      mov result, ax
      mov result+2, dx
mov ah, 4ch
      int 21h
      code ends
      end main
```

Input: n1=1234H;n2=0FF40H; Output:12265900

n1= 1234H n2=5678H Output: 6260060

RESULT: The program for multiplication of two unsigned 16-bit numbers is executed and the output is verified.

2.4.1 8-bit signed division.



2.4.1. Aim: Write an ALP for 8-bit signed division.

Software Required: MASM Assembler

PROGRAM

8-bit signed division

```
data segment
n1      db 0afh
        n2      db 0feh
result db ?
data ends
code segment
        assume cs:code,ds:data
        main:mov ax,data
        mov ds,ax
mov al, n1
        cbw
        mov bl, n2
idiv bl
        mov result, al
        mov ah, 4ch
         int 21h
        code ends
       end main
```

Input: n1=E102H
n2=78H

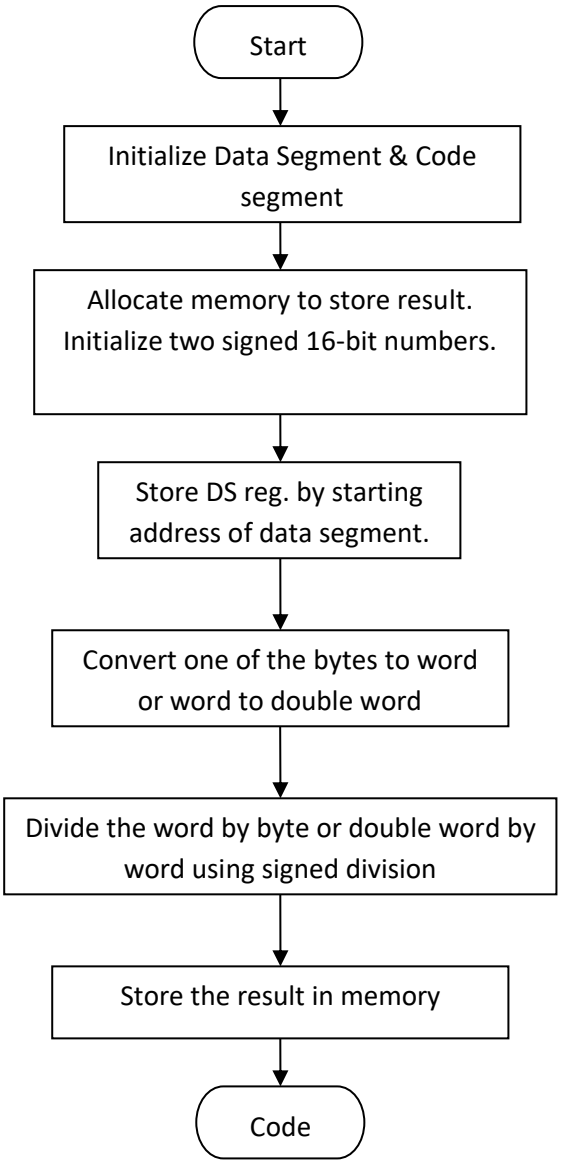
Output: Quo = 1E0H
Rem = 02H

Input: n1=1122H
n2= 33H

Output: 56H

RESULT: The program for division of two signed 8-bit numbers is executed and the output is verified.

2.4.2. 16-bit signed division



2.4.2. Aim: Write an ALP for 16-bit signed division

Software Required: MASM Assembler

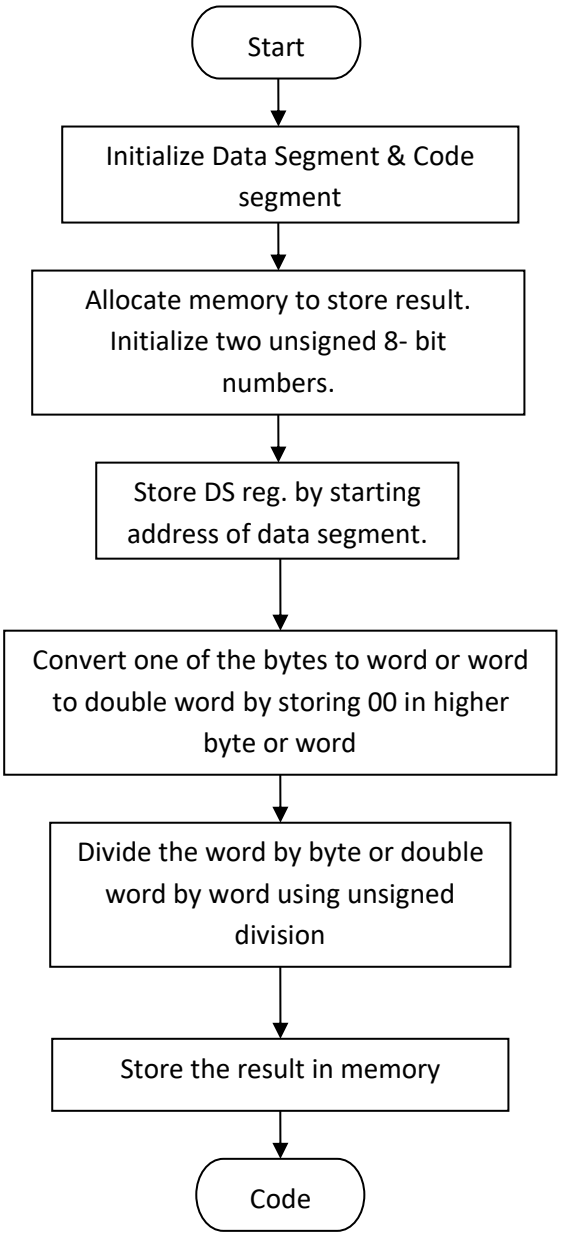
PROGRAM

16-bit signed division

```
                                                        Input:
n1=FAEBH
data segment                                                n2=1234H
n1      dw FAEBh                                        Output:Quo=DH
        n2      dw 1234h                                        Rem=00H
result dw ?
data ends
code segment                                            Input: n1=  FFCC
        assume cs:code,ds:data                                  n2 = 1234
        main:mov ax,data
         mov ds,ax                                              Output:
Quo: E H
mov ax, n1
        cwd                                                     Rem:  0H
        mov bx, n2
idiv bx
        mov result, ax
        mov ah, 4ch
        int 21h
        code ends
        end main
```

RESULT: The program for division of two signed 16-bit numbers is executed and the output is verified

2.4.3. 8-bit Unsigned division.



2.4.3. Write an ALP for 8- or 16-bit Unsigned division.

Software Required: MASM Assembler

8-bit Unsigned division

```
data segment
n1      db 40h
        n2      db 02h
result db  2dup(?)
data ends

code segment
        assume cs:code,ds:data
        main:mov ax,data
        mov ds,ax
mov al, n1
        cbw
        mov bl, n2
div bl
        mov result, al
        mov result+1,ah
        mov ah, 4ch
        int 21h
        code ends
        end main
```

Input: n1=40H
n2=02H

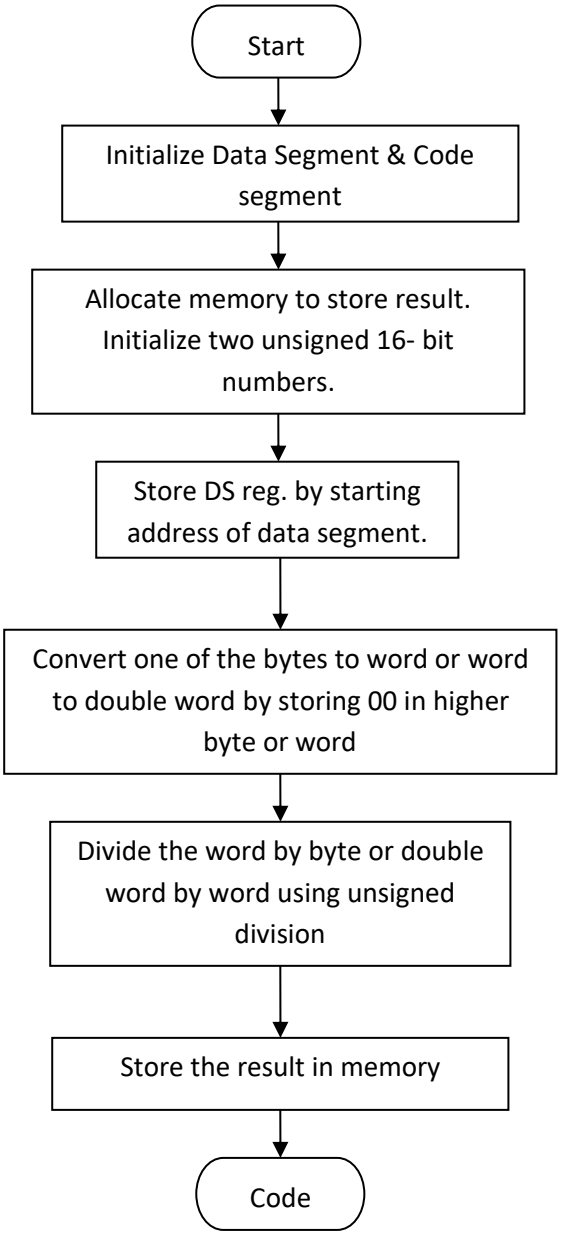
Output: Quo=20H
Rem=00H

Input: n1 = 7FH
n2 = 0EH

Output: Quo = H
Rem=H

RESULT: The program for division of two unsigned 8-bit numbers is executed and the output is verified.

2.4.4. 16-bit Unsigned division.



2.4.4. Write an ALP for 8- or 16-bit Unsigned division

Software Required: MASM Assembler

PROGRAM

data segment

```
n1      dw  2244h
 n2      dw 0d6h
result dw  2 dup(?)
data ends
```

Input: n1=2244H
n2=D6H

code segment

```
      assume cs:code,ds:data
      main:mov ax,data
      mov ds,ax
mov ax, n1
      cwd
      mov bx, n2
div bx
      mov result, ax
      mov result+2, dx
      mov ah, 4ch
      int 21h
      code ends
      end main
```

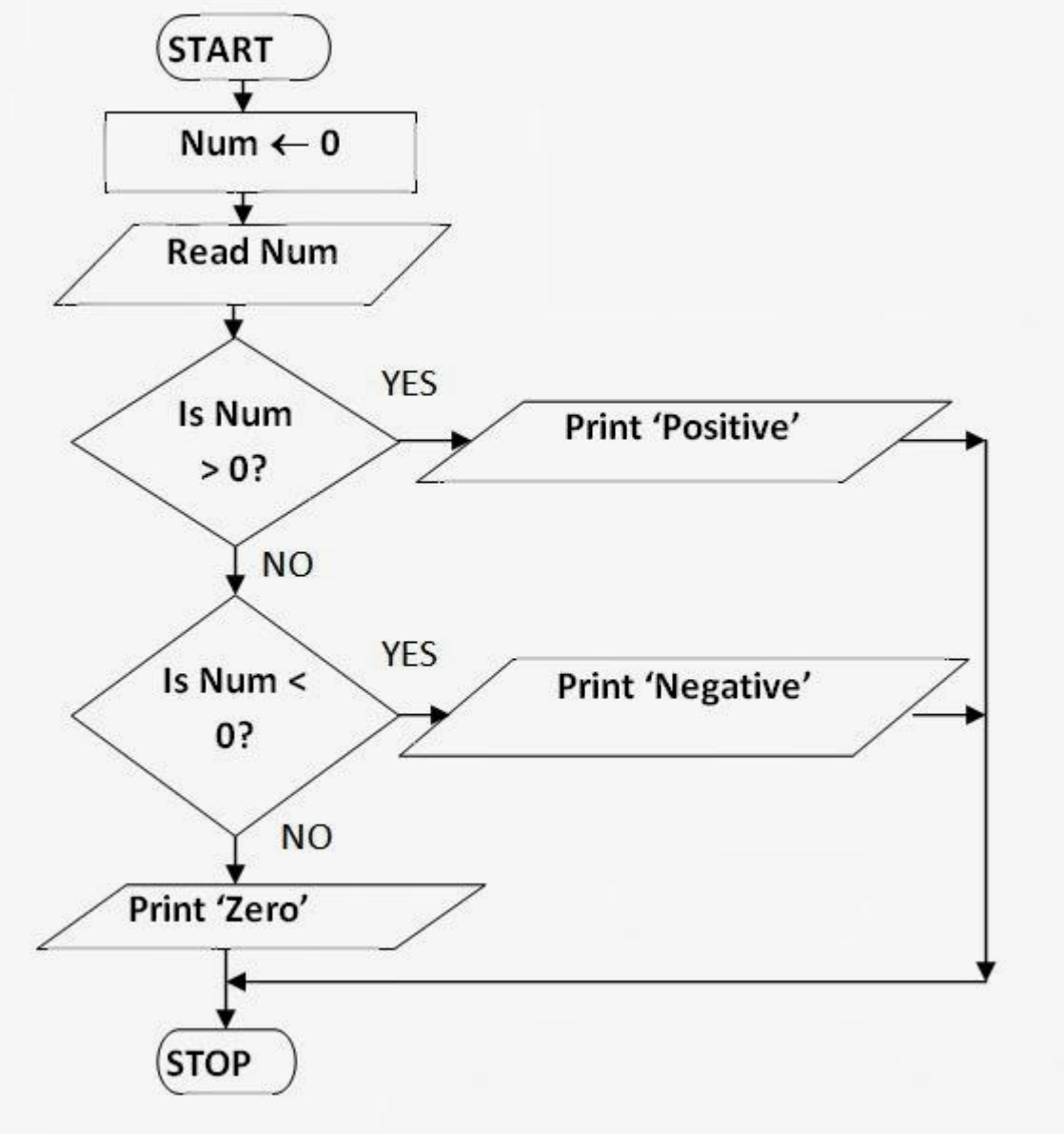
Output:Quo=28H
Rem=D4H

RESULT: The program for division of two unsigned 16-bit numbers is executed and the output is verified.

EXPERIMENT NO: 3

LOGICAL OPERATIONS USING 8086

3.1. To check whether the given number is Positive or Negative.



EXPERIMENT NO: 3

LOGICAL OPERATIONS

3.1 *To check whether the given number is Positive or Negative.*

Aim: To write a ALP to perform logical operations using 8086 microprocessor.

Software Required: MASM Assembler

PROGRAM

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
ORG 1000H
NUM DB " "
DISPLAY1 DB "POSITIVE NUMBER $"
DISPLAY2 DB "NEGATIVE NUMBER $"
DATA ENDS
CODE SEGMENT
START: ORG 2000H
    MOV AX,DATA
    MOV DS,AX
    MOV SI,OFFSET NUM
    MOV AL,[SI]
    MOV BL,AL
    SHL AL,01H
    JC NG
    MOV AH,09H
    MOV DX,OFFSET DISPLAY1
    INT 21H
    JMP STOP
NG:  MOV AH,09H
    MOV DX,OFFSET DISPLAY2
    INT 21H
```

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

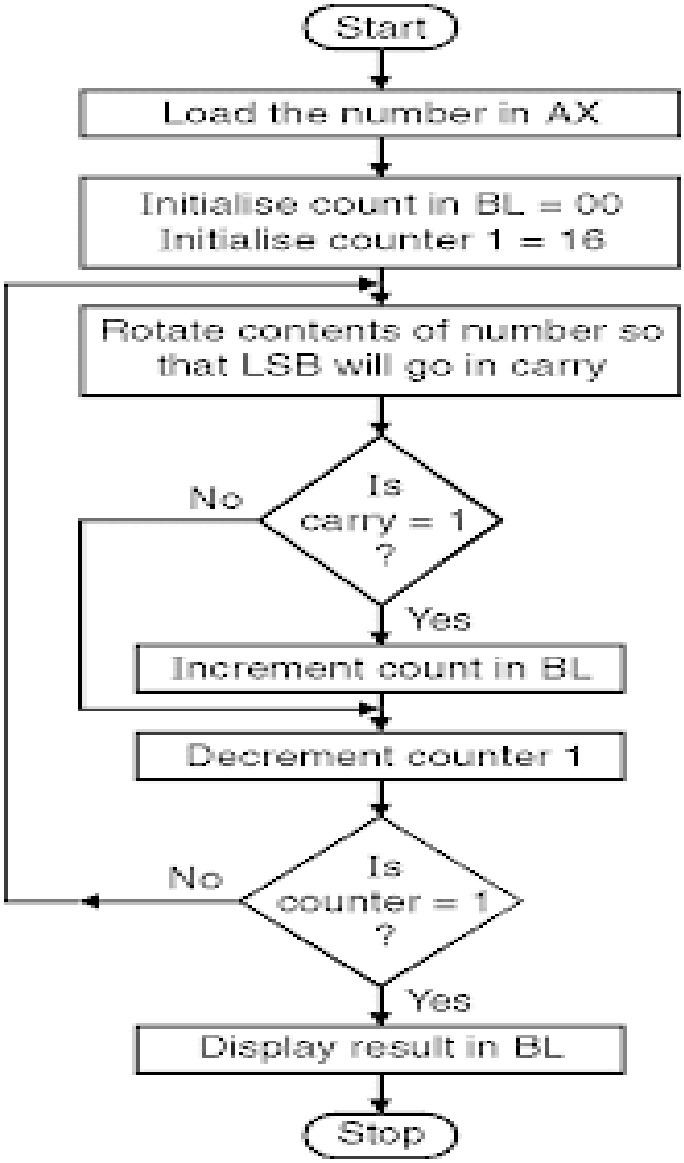
Input: 98H
Output: Negative

Input: 41H
Output:Positive

RESULT:

The program to check whether the given number is Positive or Negative is executed and the output is verified

3.2. To find the number of 1s and 0s in a given number.



3.2. Write an ALP to find the number of 1s and 0s in a given number.

AIM: To write a microprocessor 8086 program to find the number of 1s and 0s in a given number

Software Required: MASM Assembler

PROGRAM

```
data segment
n1     db 82h
ones  db 1dup(?)
zeros db 1dup(?)
data ends

code segment
       assume cs:code,ds:data
       main:mov ax,data
       mov ds,ax
mov cx, 08h
       mov al, n1
       mov bl,00h
       mov dl,00h
back: shr al, 01
         jc l1
         inc bl
         jmp next
l1:  inc dl
next:   loop back
         mov ones, dl
         mov zeros,bl
         mov ah, 4ch
         int 21h
         code ends
         end main
```

Input: 48H 55H

Output: Ones=2 (dl=2) Ones=4 (dl=4)

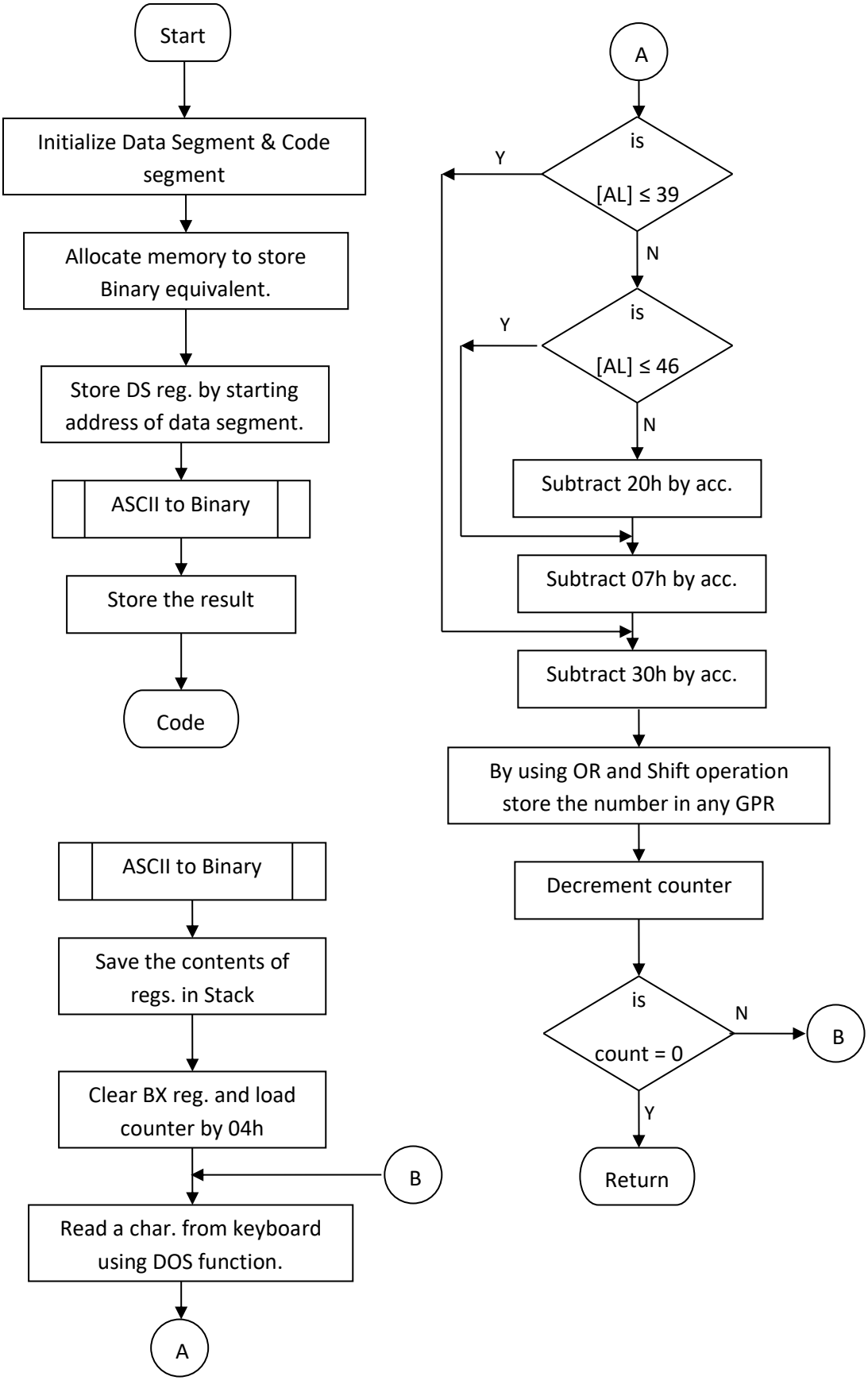
Zeros=6 (bl=6) Zeros=4 (bl=4)

RESULT: The program *to find the number of 1s and 0s in a given number* is executed and the output is verified

EXPERIMENT: 4

CODE CONVERSIONS

4.1 ASCII to Binary



EXPERIMENT -4

CODE CONVERSIONS

4.1 ASCII to Binary

Aim:

To write an ALP program to convert ASCII to Binary

Software Used: MASM

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA  SEGMENT
A DB 0AHD
DATA ENDS
CODE  SEGMENT
        MAIN: MOV     AX,DATA
              MOV     DS,AX
              MOV     AL,A
              MOV     CL,08H
              MOV     AH,00H
         UP : SHL     AL,01H
              MOV     BL,AL
              MOV     AL,00H
              ADC     AL,30H
              MOV     DL,AL
              MOV     AH,02H
              INT     21H
              MOV     AL,BL
              DEC     CL
              JNZ     UP
              MOV     AH,4CH
              INT     21H
        CODE ENDS
        END MAIN
```

INPUT 1 : 37 H
OUTPUT1 :00110111

INPUT2 : 'K'
OUTPUT2: 01001011

RESULT:

The program to convert o *ASCII code to Binary code* is executed and the output is verified

4.2 Decimal to Hexadecimal conversion

Aim:

To write an ALP program to convert Decimal number to Hexadecimal number

Software Used: MASM

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
NUM DB " "
CO DB " "
DATA ENDS
CODE SEGMENT
START: ORG 1000H
MOV AX,DATA
MOV DS,AX
MOV SI,OFFSET NUM
MOV BL,[SI]
MOV AL,BL
AND BL,0FH
AND AL,0F0H
MOV CL,04H
SHR AL,CL
MOV DL,10
MUL DL
ADD AL,BL
MOV DI,OFFSET CO
MOV [DI],AX
INT 03H
CODE ENDS
END START
```

INPUT1 : 66
OUTPUT1: 42H

INPUT2: 47
OUTPUT:2E

RESULT:

The program to convert Decimal to Hexadecimal is executed and the output is verified.

4.3. ASCII to Decimal

Aim: To write a ALP program to convert ASCII to Decimal number and execute using MASM

Software: MASM

Program:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
ORG 1000H
ASCII DB 48,49,50,51,52,53,54,55,56,57
ORG 2000H
BCD DB 10 DUP(?)
DATA ENDS
CODE SEGMENT
START:  MOV AX,DATA
        MOV DS,AX
        MOV SI,OFFSET ASCII
        MOV DI,OFFSET BCD
        MOV CL,0AH
    L1:CALL ATOB
        DEC CL
        JNZ L1
        INT 03H
   ATOB PROC
        MOV AL,[SI]
        SUB AL,30H
        MOV [DI],SI
        INC SI
        INC DI
        RET
CODE ENDS
END START
```

INPUT: 48,49,50,51,52,53,54,55,56,57

OUTPUT: 0,1,2,3,4,5,6,7,8,9

Result: Hence ALP program for converting ASCII to Decimal is written and executed.

4.4 Binary to BCD

Aim: To write a ALP to convert Binary number to BCD using 8086 microprocessor and execute using MASM tool

Software: MASM

Program:

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
NO1 DB "1001000000110110"
ORG 1000H
D1 DW 4 DUP (?)
DATA ENDS
CODE SEGMENT
START:
MOV AX, DATA
MOV DS, AX
LEA SI, NO1
LEA DI, D1
MOV CX, 04H
TOP:
MOV BX, 00H
MOV AX, [SI]
ROR AX, 1
JNC P2
ADD BX, 08H
P2:
INC SI
MOV AX, [SI]
ROR AX, 1
JNC P3
ADD BX, 04H
P3:
INC SI
MOV AX, [SI]
ROR AX, 1
JNC P4
```

```
ADD BX, 02H
P4:
INC SI
MOV AX, [SI]
ROR AX, 1
JNC P5
ADD BX, 01H
P5:
MOV [DI], BX
INC DI
INC SI
DEC CX
JNZ TOP
INT 3
CODE ENDS
END START
```

INPUT: 1001000000110110

OUTPUT: 9036

INPUT:0011100001110000

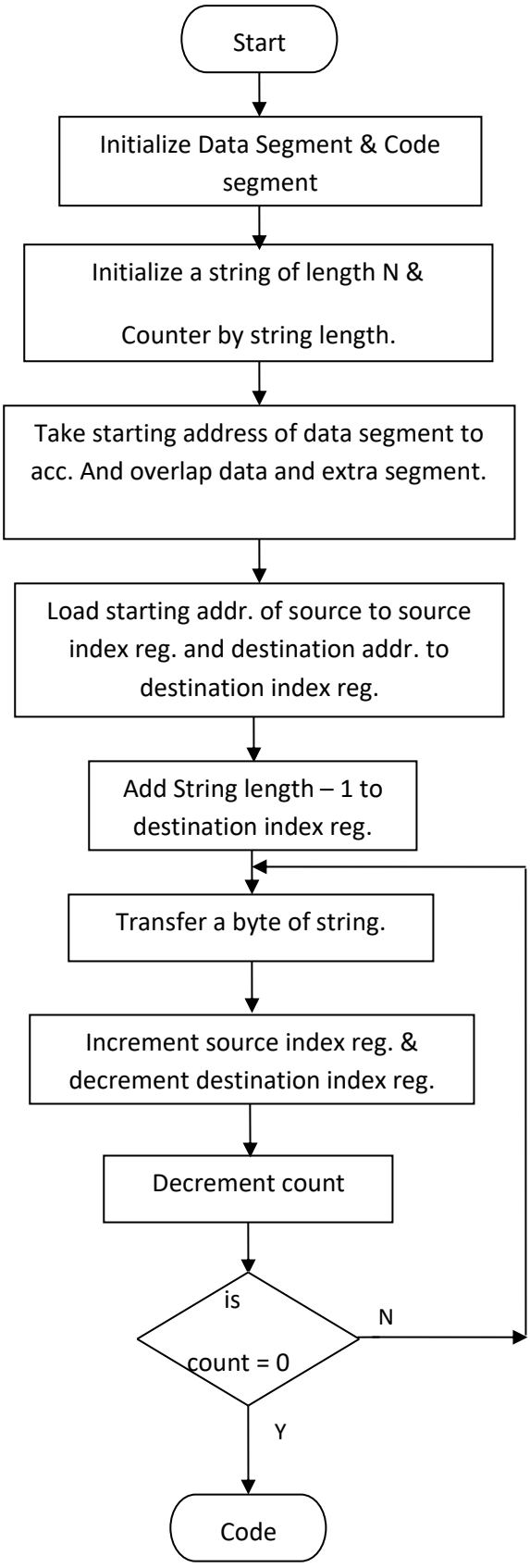
OUTPUT: 3870

Result: **Thus, ALP program for converting Binary to BCD is executed and verified using MASM tool.**

EXPERIMENT: 5

STRING OPERATIONS

5.1 Reverse the String



EXPERIMENT: 5

STRING OPERATIONS

5.1 Reverse the given string.

Aim: To Write an ALP to reverse the given string.

Software Required: MASM Assembler

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
    MSGS DB "WELCOME$"
    LEN EQU ($-MSGS)
    MSGR DB 25 DUP(?)
DATA ENDS
CODE SEGMENT
    MAIN:MOV AX,DATA
    MOV DS,AX
    MOV ES,AX
    MOV SI,OFFSET MSGS
    MOV DI,OFFSET MSGR
    MOV CX,LEN-1
    ADD DI,LEN-2
    CLD
REVERSE:  MOVSB
      DEC DI
      DEC DI
      LOOP REVERSE
      MOV BYTE PTR[DI+LEN],'$'
      MOV DX,OFFSET MSGR
      MOV AH,09H
      INT 21H
      MOV AH, 4CH
      INT 21H
```

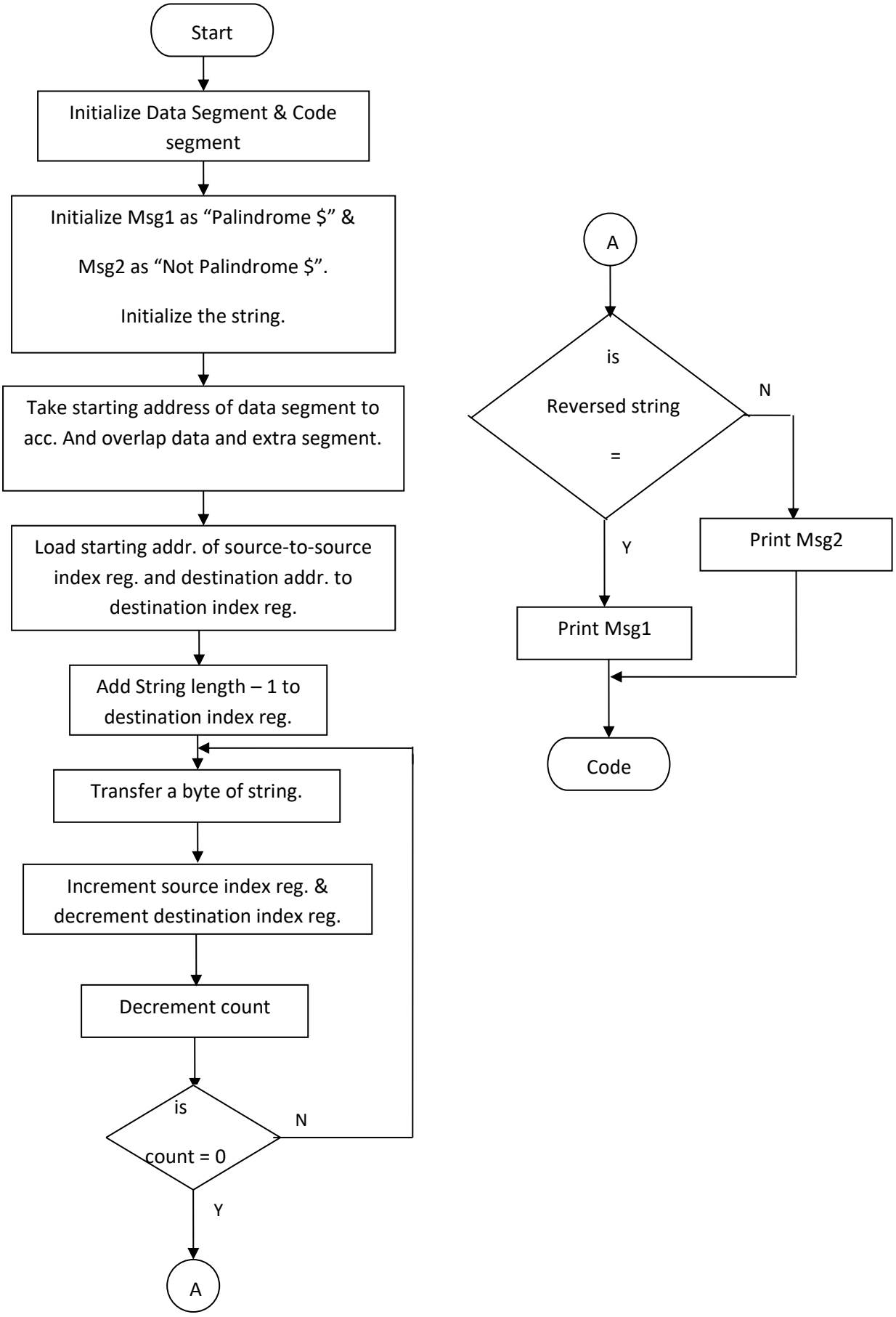
```
CODE ENDS
END MAIN
```

Input: 0001→CODE ENDS Input: END MAIN

Output: 0006→ONE Output: 0006→ONE

RESULT: The program to reverse a String is executed and the output is verified

5.2. Check whether the given string is Palindrome or not



5.2. Aim: To Write an ALP to check whether the given string is Palindrome or not

Software Required: MASM Assembler

PROGRAM :

```
data segment
        Msgs db  "APPA$"
        len     equ ($ - Msgs)
Msgr db  20 dup(0)
        Msgp db  "Palindrome $"
Msgn db  "Not Palindrome $"
Data ends
code segment
assume cs:code,ds:data
main : mov ax,data
        mov ds,ax
        mov es, ax
        mov si, offset Msgs
        mov di, offset Msgr
        mov cx, len-1
        add di, len-2
        cld
reverse:    movsb
            dec di
            dec di
            loop reverse
            mov byte ptr[di+len],'$'
            mov si, offset Msgs
            mov di, offset Msgr
            mov cx, len
            repe cmpsb
             jz  pal
            mov dx, offset Msgn
            jmp display
pal:        mov dx, offset Msgp
display:    mov ah, 09
            int 21h
            mov ah, 4ch
            int 21h
            code ends
            end main
```

Input: MALAYALAM

Output:Palindrome

Input: MICRO

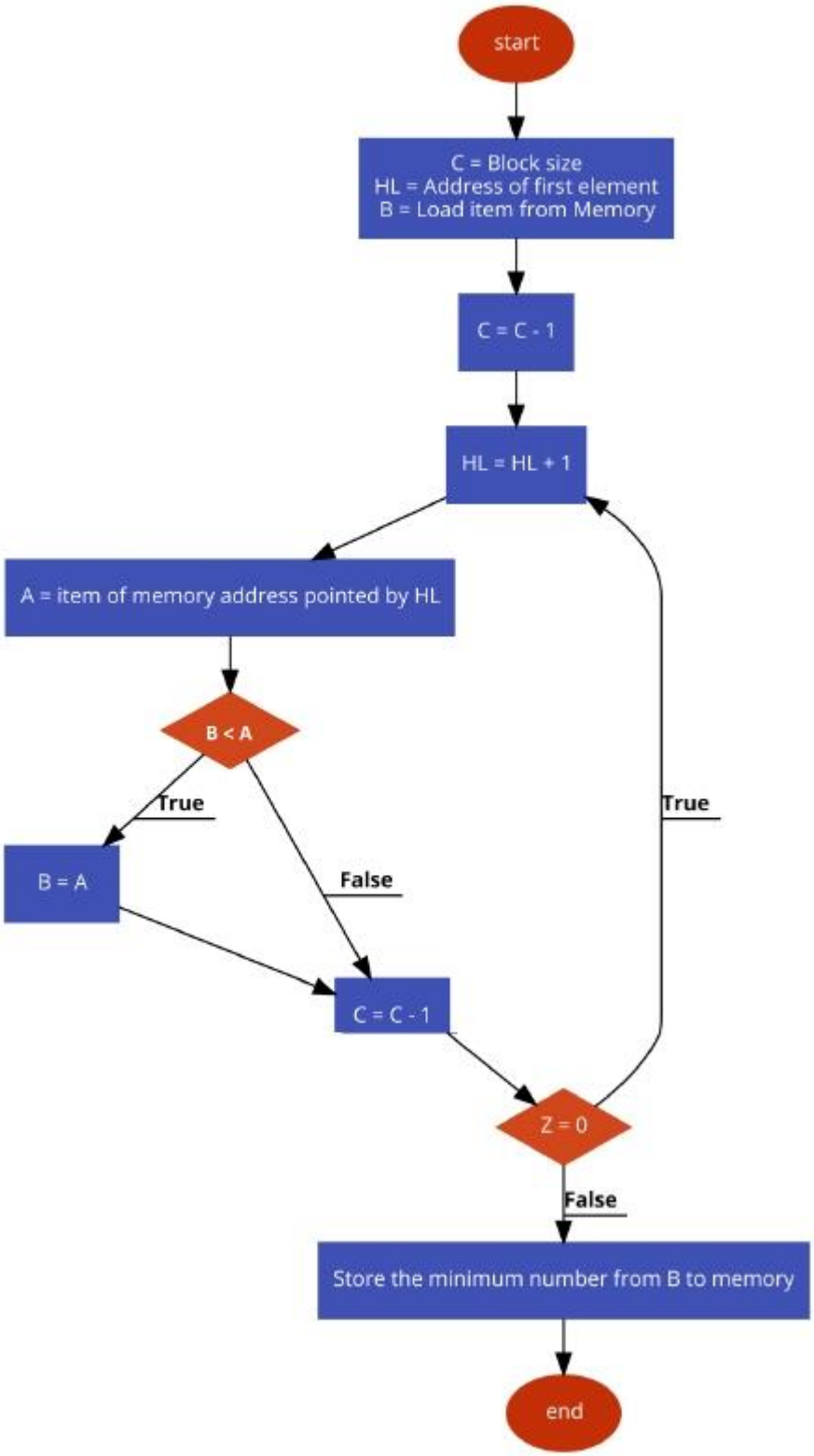
Output: Not Palindrome

RESULT: The program to check whether the given string is Palindrome or not is executed and the output is verified.

EXPERIMENT: 6

SORTING OPERATIONS

6.1 Finding Largest Number



EXPERIMENT – 6

SORTING OPERATIONS

6.1 Finding Largest Number

Aim: ***To Write an ALP to find the largest number in an array***

Software Required : MASM Assembler

PROGRAM:

```
ASSUME CS: CODE,DS:DATA
DATA SEGMENT
ORG 1000H
LIST DB 05H,06H,03H,02H,09H
DATA ENDS
CODE SEGMENT
START: ORG 2000H
      MOV AX,DATA
      MOV DS,AX
      MOV CL,04H
      MOV SI,OFFSET LIST
      MOV AL,[SI]
  L1: CMP AL,[SI+1]
      JNC L
      XCHG AL,[SI+1]
  L:  INC SI
      DEC CL
      JNZ L1
      MOV SI,3000H
      MOV [SI],AL
      INT 03H
CODE ENDS
END START
```

Input: 76H,F2H,46H,18H,24H

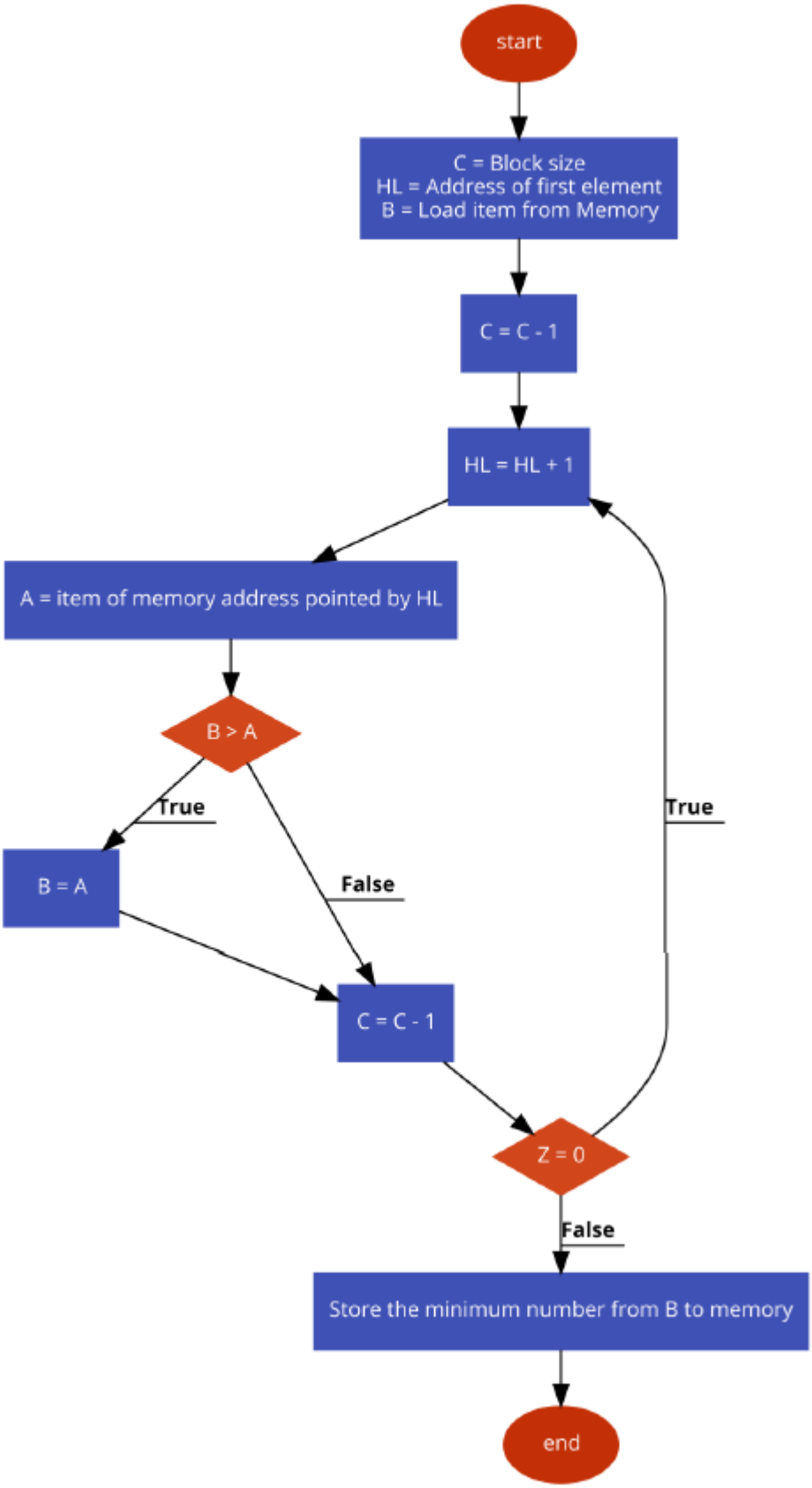
Output: F2H

Input: 01H 05H 02H 04H 03H

Output: 05H

RESULT: The program to *find the largest number in an array* is executed and the output is verified.

6.2 Finding Smallest Number



6.2. Aim: To Write an ALP to find smallest number in an array

Software Required: MASM Assembler

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
ORG 1000H
LIST DB 05H,06H,03H,02H,09H
DATA ENDS
CODE SEGMENT
START: ORG 2000H
    MOV AX,DATA
    MOV DS,AX
    MOV CL,04H
    MOV SI,OFFSET LIST
    MOV AL,[SI]
  L1: CMP AL,[SI+1]
    JC L
    XCHG AL,[SI+1]
  L:  INC SI
    DEC CL
    JNZ L1
    MOV SI,3000H
    MOV [SI],AL
    INT 03H
CODE ENDS
END START
```

Input: 76H,F2H,34H,19H,01H

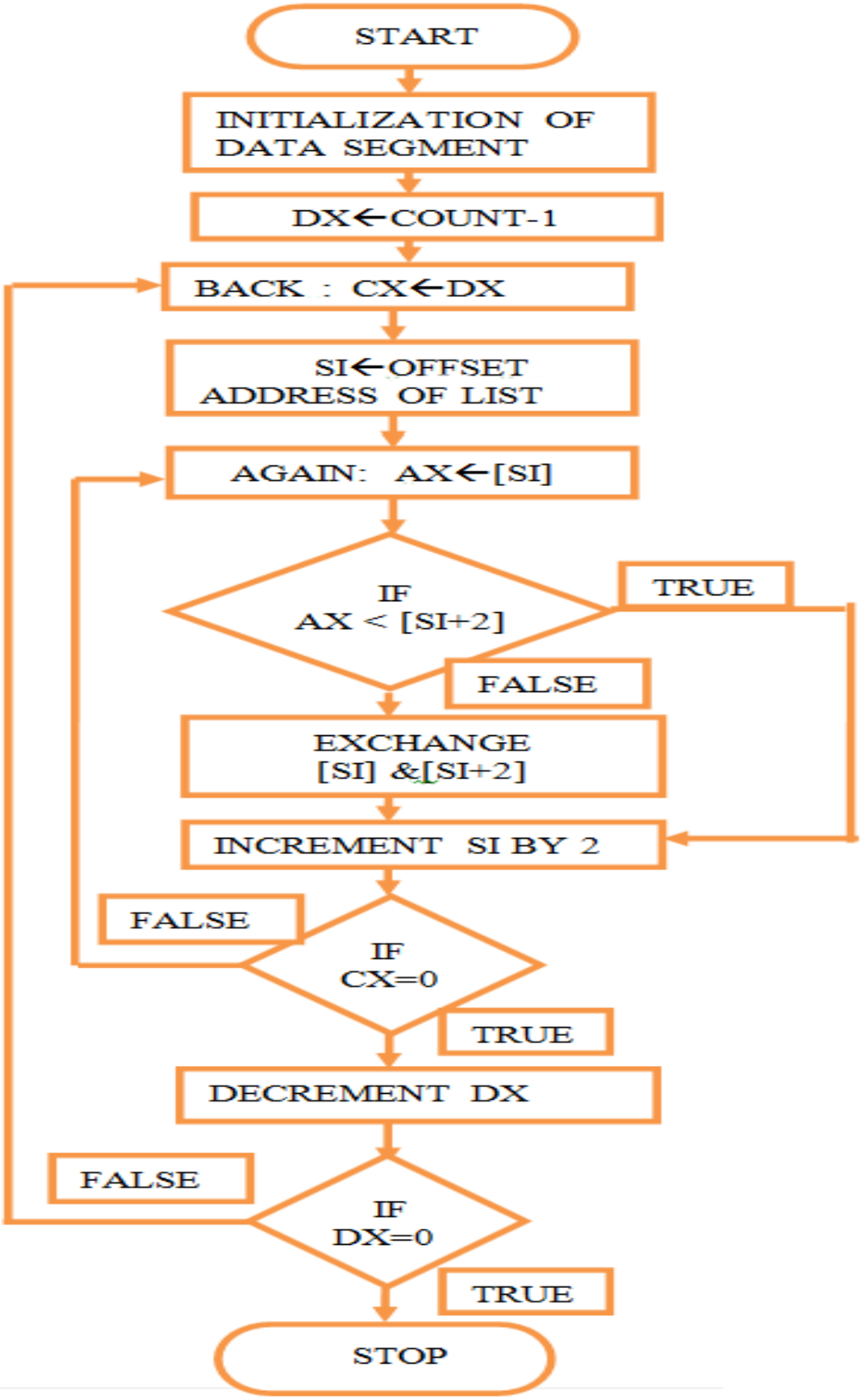
Output: 01H

Input: 01H 05H 02H 04H 03H

Output: 01H

RESULT: The program to *find the smallest number in an array* is executed and the output is verified.

6.3 Ascending Order



6.3. Aim: To Write an ALP to sort the given array in an Ascending order.

Software Required: MASM Assembler

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
ORG 1000H
LIST DB 05H,06H,03H,02H,09H
DATA ENDS
CODE SEGMENT
START: ORG 2000H
    MOV AX,DATA
    MOV DS,AX
    MOV CL,04H
    MOV DL,04H
 L1: MOV DL,CL
    MOV SI,OFFSET LIST
 L2: MOV AL,[SI]
    CMP AL,[SI+1]
    JC L
    XCHG AL,[SI+1]
    MOV [SI],AL
 L:  INC SI
    DEC DL
    JNZ L2
    DEC CL
    JNZ L1
    INT 03H
CODE ENDS
END START
```

Input: 45H,ADH,81H,76H,FEH

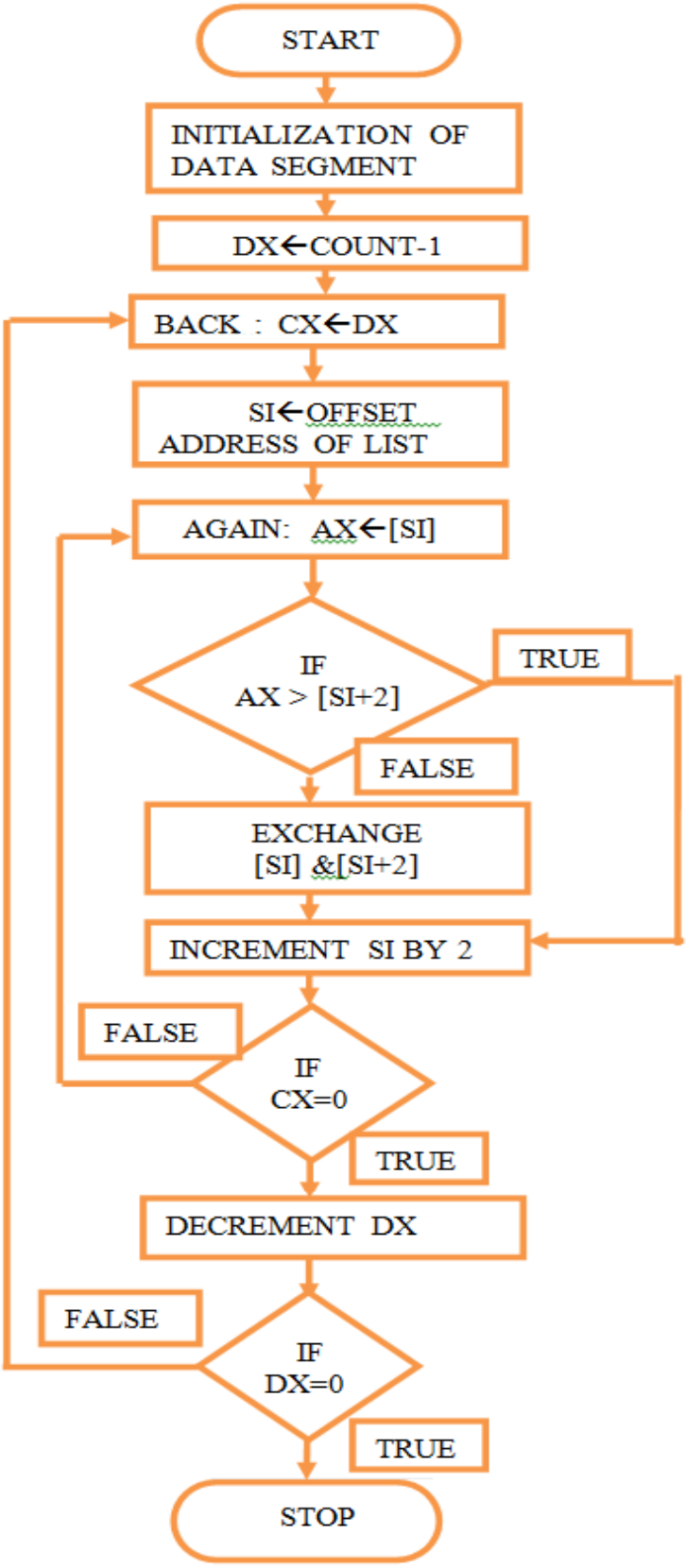
Output: 45H,76H,81H,ADH.FEH

Input: 05H 04H 03H 02H 01H

Output: 01H 02H 03H 04H 05H

RESULT: The program to sort the given array in an Ascending order.is executed and the output is verified

6.4 Descending Order



6.4. Aim: To Write an ALP to sort the given number in Descending order

Software Required: MASM Assembler

PROGRAM:

```
ASSUME CS:CODE,DS:DATA
DATA SEGMENT
ORG 1000H
LIST DB 05H,06H,03H,02H,09H
DATA ENDS
CODE SEGMENT
START: ORG 2000H
    MOV AX,DATA
    MOV DS,AX
    MOV CL,04H
    MOV DL,04H
 L1: MOV DL,CL
    MOV SI,OFFSET LIST
 L2: MOV AL,[SI]
    CMP AL,[SI+1]
    JNC L
    XCHG AL,[SI+1]
    MOV [SI],AL
 L:  INC SI
    DEC DL
    JNZ L2
    DEC CL
    JNZ L1
    INT 03H
CODE ENDS
END START
```

Input: 45H,ADH,81H,76H,FEH Intput: 01H 02H 03H 04H 05H

Output: FEH,ADH,81H,76H.45H Output: 05H 04H 03H 02H 01H

RESULT: The program to sort the given array in descending order is executed and the output is verified.

EXPERIMENT -7

INTERFACING WITH 8086 MICROPROCESSOR

EXPERIMENT-7

INTERFACING WITH 8086 MICROPROCESSOR

7.1 Write an ALP to illustrate Logic Controller

Aim: ***To write an ALP to illustrate Logic Controller***

Program:

```
///logic controller interface

data segment


pa equ 0xef70h
pb equ 0xef71h
pc equ 0xef72h
ctrl equ 0ef73h
data ends
code segment
assume cs:code,ds:data
main:mov ax,data
mov ds,ax
mov al,80h
mov dx,0ef73h
out dx,al
mov al,00
mov cx,10
repup:mov dx,0ef70h
out dx,al
call delay
add al,01
loop repup
mov al,09
mov cx,10
repdown:mov dx,0ef70h
out dx,al
call delay
sub al,01
loop repdown
mov ah,4ch
```

```
int 21h
delay proc
push cx
push bx
mov cx,0ffffh
loop1:mov bx,0ffffh
in1:dec bx
jnz in1
loop loop1
pop bx
pop cx
ret
delay endp
code ends
end main
```

Result : **Thus interfacing of Logic Controller using 8086 is demonstrated .**

7.2 To write an ALP to interface Seven Segment Display using 8086.

Program:

```
//seven segment display

data segment
fcode db 8eh,0f9h,88h,86h
scode db 89h,86h,0c7h,8ch
pa equ 0ef70h
pb equ 0ef71h
pc equ 0ef72h
cw equ 0ef73h
data ends
code segment
assume cs:code,ds:data
main:mov ax,data
     mov ds,ax
     mov dx,cw
     mov al,80h
     out dx,al
again:mov si,offset fcode
      call display1
      call delay
      mov si,offset scode
      call display1
      call delay
      mov ah,06h
      mov dl,0ffh
      int 21h
      cmp al,'q'
      jne again
      mov ah,4ch
      int 21h

display1 proc
      mov cx,04h
loop2:mov bl,08h
      mov al,[si]
```

```
next: rol al,01h
      mov dx,pb
      out dx,al
      push ax
      mov bh,al
      mov al,01h
      mov dx,pc
      out dx,al
      mov al,00h
      out dx,al
      dec bl
      pop ax
      jz next1
      mov al,bh
      jmp next

next1:inc si
      loop loop2
      ret
      display1 endp

delay proc
      push ax
      push cx
      mov cx,0faa0h

loop1: mov ax,0ffffh
loop3: dec ax
       jnz loop3
       loop loop1
       pop cx
       pop ax
       ret
       delay endp
       code ends
       end main
```

Result: **Thus ALP for seven segment display interfacing is demonstrated.**

7.3 To write an ALP program for interfacing Keyboard using 8086 microprocessor.

Program:

```
//switch keypad

data segment
pa equ 0ef70h
pb equ 0ef71h
pc equ 0ef72h
cw equ 0ef73h
prom db "press any key$"
msg1 db "the row is"
res1 db?
msg2 db"the column is"
res2 db?,13,10,'$'
data ends
code segment
assume cs:code,ds:data
main:mov ax,data
     mov ds,ax
     mov ah,09h
     lea dx,prom
     int 81h
     mov al,90h
     mov dx,cw
     out dx,al
again:mov ah,06
      mov dl,0ffh
      int21h
      jnz quit
      mov si,00
      call scan
      mov al,16h
      add al,31h
      mov res1,al
      mov al,ah
      add al,31h
      mov res2,al
      cmp si,00
      je again
      mov ah,09h
      lea dx,'msg'
```

```
    int 21h
quit:mov ah,40h
     int 21h
     scan proc
     mov cx,3
     mov bh,00
     mov al,80h
natrow:rol al,1
     mov bl,al
     mov dx,pc
     out dx,al
     mov dx,pa
     in al,dx,
     cmp al,00
     jne keyid
     mov al,bl
     inc bm
     loop natrow
     ret
keyid:mov si,1
   mov cx,8
   mov ah,00h
 agn:rol al.1
     jc skip
     inc ah
     loop agn
  skip:ret
  scan endp
  code ends
  end main
```

Result: **Thus, ALP program was written for interfacing keyboard using 8086 microprocessor.**

VIVA QUESTIONS

1.What is a Microprocessor?

2. What are the 4 Segments?

3. What is Bandwidth?

4. What is Clock Speed?

5. What are the features of Intel 8086?

6. What is Logical Address?

7. What is The Effective Address:

8. What is Physical Address?

9.What are the flags in 8086?

10.Why crystal is a preferred clock source?

11.What is Tri-state logic?

12.What happens when HLT instruction is executed in processor?

13.What is Program counter?

14.What is 1st / 2nd / 3rd / 4th generation processor?

15.Name the processor lines of two major manufacturers?

16.How many bit combinations are there in a byte?

17.Have you studied buses? What types?

18.What is the Maximum clock frequency in 8086?

19.What is meant by Maskable interrupts?

20.What is Non-Maskable interrupts?

21.What are the different functional units in 8086?

22.What are the various segment registers in 8086?

23.What does EU do?

24.Which Stack is used in 8086? k is used in 8086?

25.What are the flags in 8086?

26.What is SIM and RIM instructions?

27.What are the different types of Addressing Modes?

28.What are the General Data Registers & their uses?

29.What are Segment Registers & their uses?

30.What are Flag registers?

31.What does the 8086 Architecture contain?

32. What are Data Copy/Transfer Instructions?

33. What are Machine Control Instructions?
34. What are Flag Manipulation Instructions?
35. What are String Instructions?
36. What are different parts for 8086 architecture?
37.What is an Interrupts
38. What is an Opcode?
39.What is an Operand?
40.Explain the difference between a JMP and CALL instruction?
41.What is meant by Polling?
42.What is meant by Interrupt?
43.What is an Instruction?
44.What is Microcontroller and Microcomputer?
45.What is Assembler?
46.Define Variable?
47.Explain Dup?
48.Define Pipelining?
49.What is the use of HLDA?
50. Explain about "LEA"?