Experiment No-01

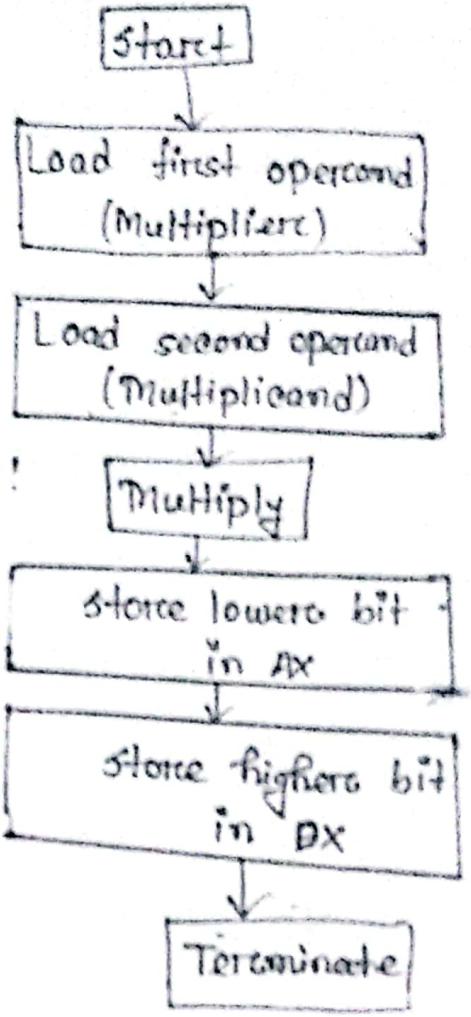
Experiment Name:- Write an assembly language Program for Performing 16-bit multiplication

Apparatus required: Computer with EMU8086 software

Algorithm:

1. Define the data segment and stack
2. Declare the variable to store the operand and result
3. Load the first operand into the AX register
4. Load the second operand into the BX register
5. Multiply the values in AX and BX using the MUL instruction
6. Store the lower 16-bits of the result in the result variable (AX).
7. Store higher 16-bits in BX
8. Terminate the Program using INT and End Program.

Flowchart:-



Program:-

```
.MODEL SMALL
.STACK
.DATA
        MULTIPLICAND  DW  00FFH;
        MULTIPLIER    DW  00FFH;
        PRODUCT DW 2(DUP(0);

.CODE
START MAIN PROC
```

```
MOV  AX, @DATA
MOV  DS , AX
MOV  AX, MULTIPLICAND
MUL  MULTIPLIER
MOV  PRODUCT, AX
MOV  PRODUCT+2 , DX
MOV   AH, 4CH
INT   21H

MAIN  ENDP
END MAIN
```

Result :-

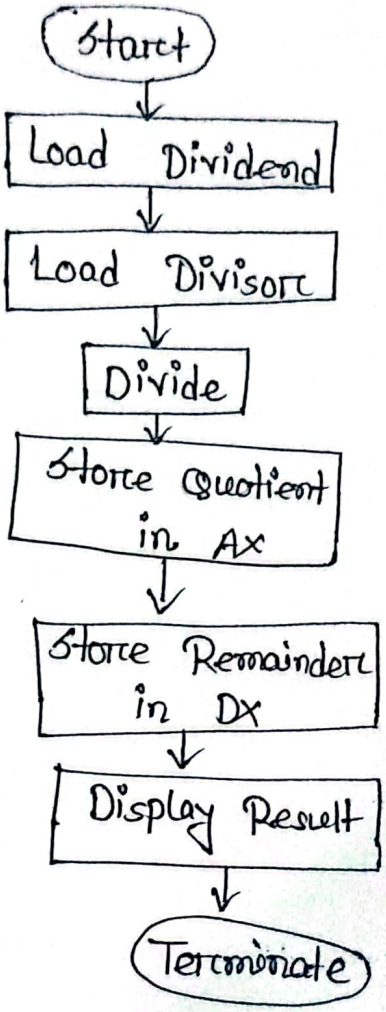
Experiment NO-02

Experiment Name:- Write an assembly language Program to divide two numbers

Apparatus required:- Computer with EMU 8086 Software

Algorithm:- 1. Define data segment and stack
2. Declare the variable to store the dividend, divisor, Quotient and Remainder
3. Load the dividend into Ax register
4. Load the divisor into Bx register
5. Divide the value into Ax and Bx using the DIV instruction
6. Store the quotient in the variable designed for it
7. Store the remainder in the variable designated for it
8. Display the Quotient and Remainder
9. Terminate the Program

Flowchart :-



Program :-

```
.MODEL SMALL
.DATA
        W1  DW  02222H
        W2  DW
        Q   DW (0)
        R   DW (0)
```

```
.CODE
MAIN  PROC
      MOV  AX, @DATA
      MOV  DS, AX
      MOV  AX, W1
      MOV  BX, W2
      DIV  BX
      MOV  Q, AX
      MOV  R, DX
      MOV  AH, 4CH
      INT  21H
MAIN  ENDP
 END  MAIN
```

Result:-

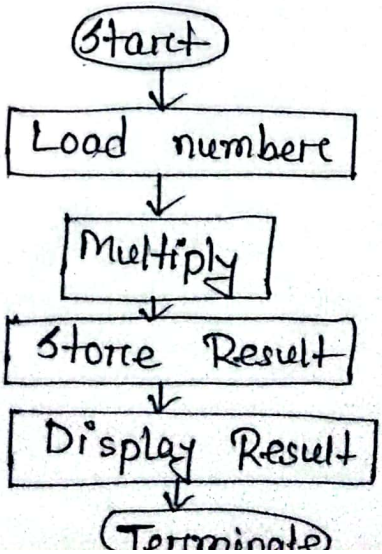
Experiment No-03

Experiment Name:- Write an assembly language Program to find square of a number

Apparatus Required:- Computer with EMU 8086 Software

Algorithm:
1. Define the data segment and stack
2. Declare the variable to store the number
3. Load the number into the AX register
4. Multiply the value of AX by itself using the 'MUL' instruction
5. Store the result in the designated variable
6. Display the result
7. Terminate the Program

Flowchart:



Program:

```
.MODEL SMALL
.STACK
.DATA
        X DB 08H
        SQR DW (0)
.CODE
MAIN PROC
        MOV AX, @DATA
        MOV DS, AX
        MOV AL, X
        MUL AL
        MOV SQR, AX
        MOV AH, 4CH
        INT 21H
MAIN ENDP
    END MAIN
```

Result:-

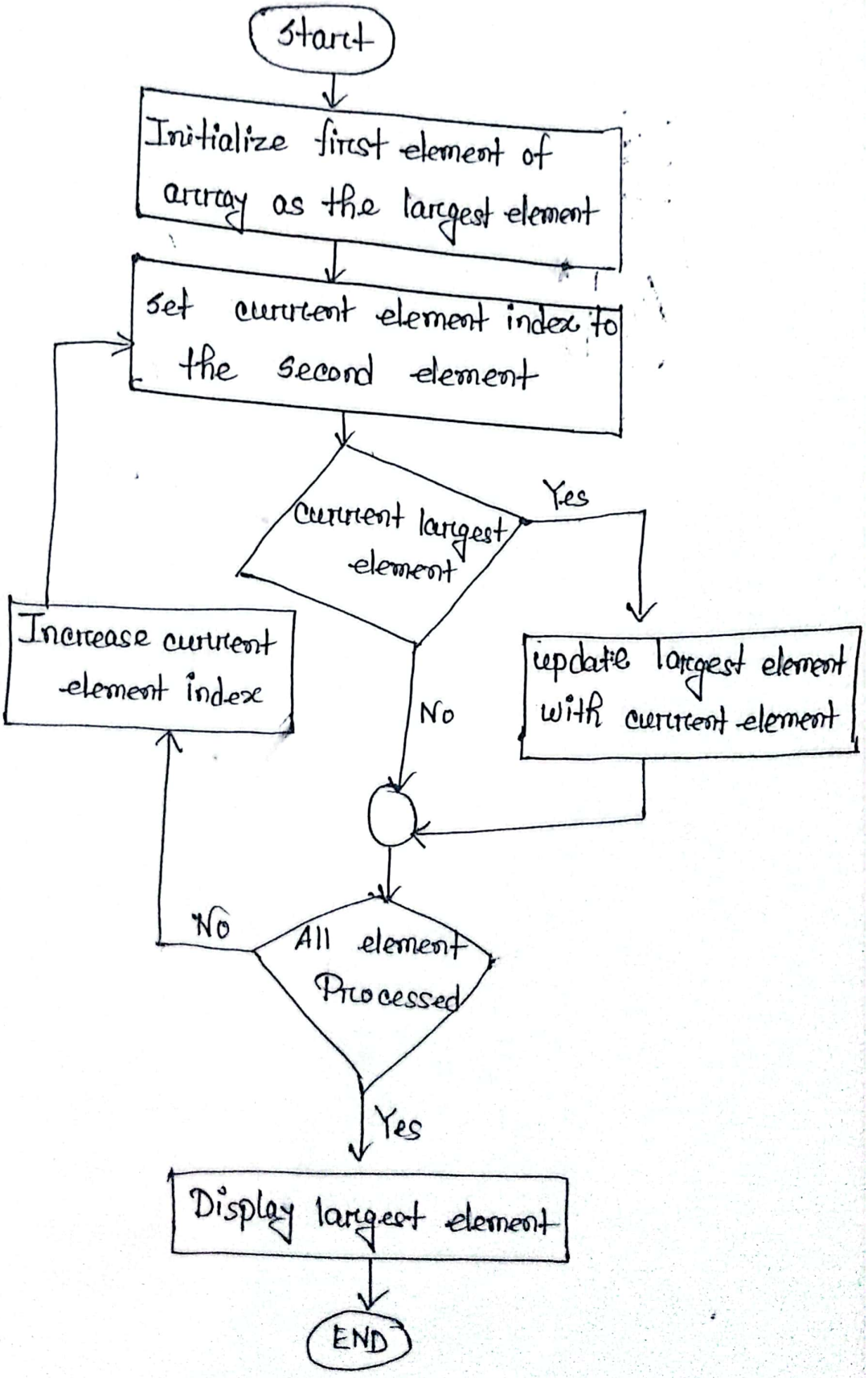
Experiment No-04

Experiment Name: Write an assembly language Program to find the largest number in an array of data

Apparatus: Computer with EMU 8086 Software

Algorithm:
1. Initialize the largest number variable to the first element of the array
2. Iterate through the remaining elements of the array
3. Compare each element with the current largest number
4. If the current element is larger than the largest number, update the largest number
5. Repeat step 3-4 for all remaining elements
6. The largest element is the final value stored in the largest number variable

Flowchart :-


Start
Initialize first element of array as the largest element
Set current element index to the second element
Current largest element
Yes
No
Increase current element index
update largest element with current element
All element Processed
No
Yes
Display largest element
END

Program:

```
.MODEL SMALL
.STACK
.DATA
    NUM DB 12H, 37H, 01H, 36H, 76H
    SMALL DB (0)
.CODE
    MOV AX, @DATA
    MOV DS, AX
    MOV CL, 05H
    MOV AL, 00H
    LEA SI, NUM
LOOP1:
    CMP AL, [SI]
    JNC SKIP
    MOV AL, [SI]
SKIP:
    INC SI
    DEC CL
    JNZ LOOP1
    MOV SMALL, AL
    MOV AH, 4CH
    INT 21H
END
```

Result:

Experiment No-05

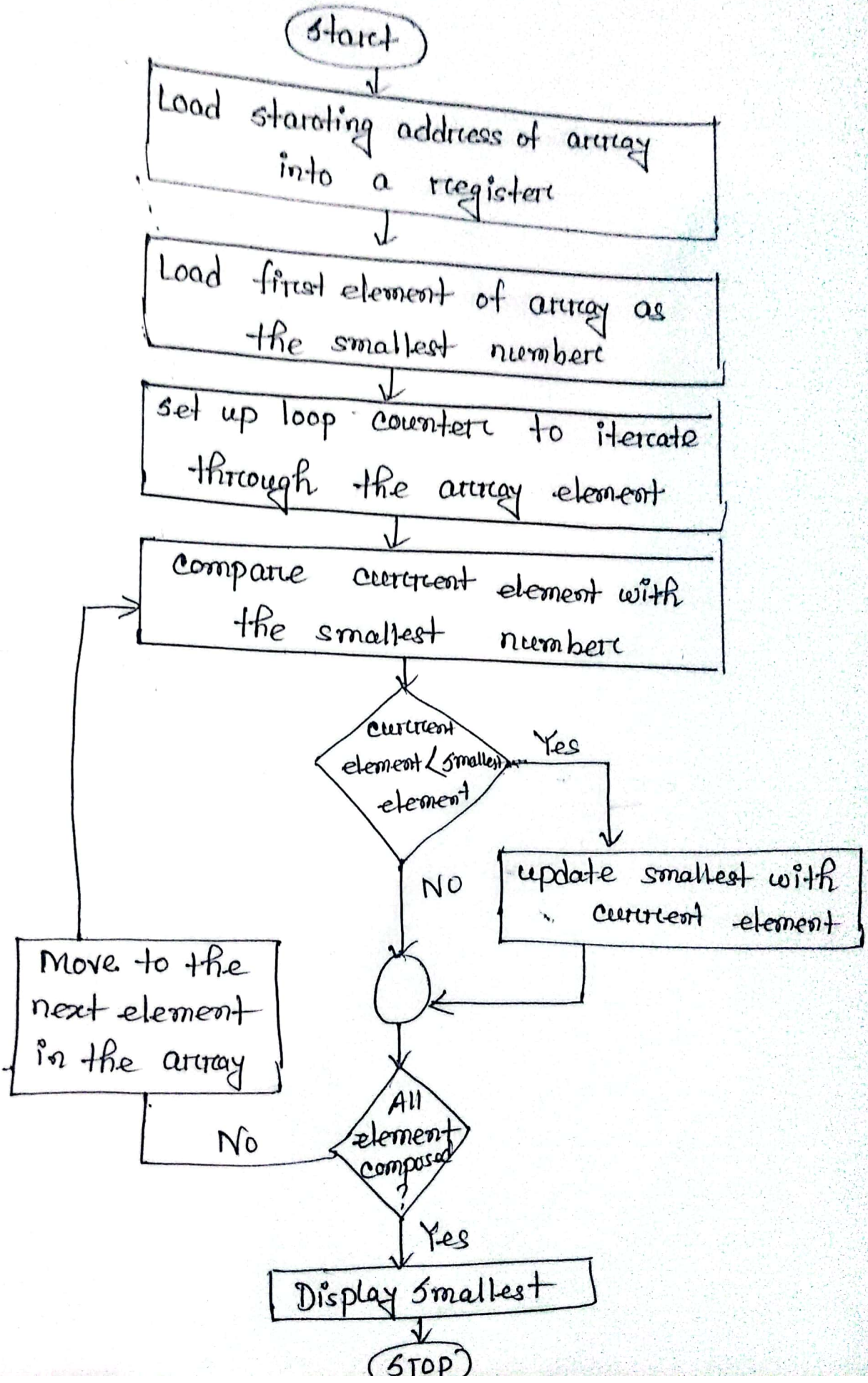
Experiment Name: Write an assembly language Program to find the smallest number in an array of data

Apparatus: Computer with EMU 8086

Algorithm:

1. Load the starting address of the array into a register
2. Load the first element of the array as the initial smallest number
3. set up a loop counter to iterate through array element
4. Compare each element with the current smallest number
5. If the current element is smaller than the smallest number, update the smallest number
6. Move to the next element of the array
7. Repeat steps 4-6 until all elements have been compared
8. Display the smallest number

Flowchart:


Start
Load starting address of array into a register
Load first element of array as the smallest number
Set up loop counter to iterate through the array element
Compare current element with the smallest number
Current element < Smallest element
Yes
NO
update smallest with current element
Move to the next element in the array
All element compared ?
No
Yes
Display Smallest
STOP

Program:-

```
.MODEL SMALL
.STACK
.DATA
   NUM DB 12H, 37H, 01H, 36H, 76H
   SMALL DB (0)
.CODE
     MOV AX, @DATA
     MOV DS, AX
     MOV CL, 05H
     MOV AL, 0FFH
     LEA SI , NUM
LOOP1: CMP AL , [SI]
       JC SKIP
       MOV AL, [SI]
SKIP:
       INC SI
       DEC CL
       JNZ LOOP1
       MOV SMALL , AL
       MOV AH, 4CH
       INI 21H
END
```

Result:

Experiment-06

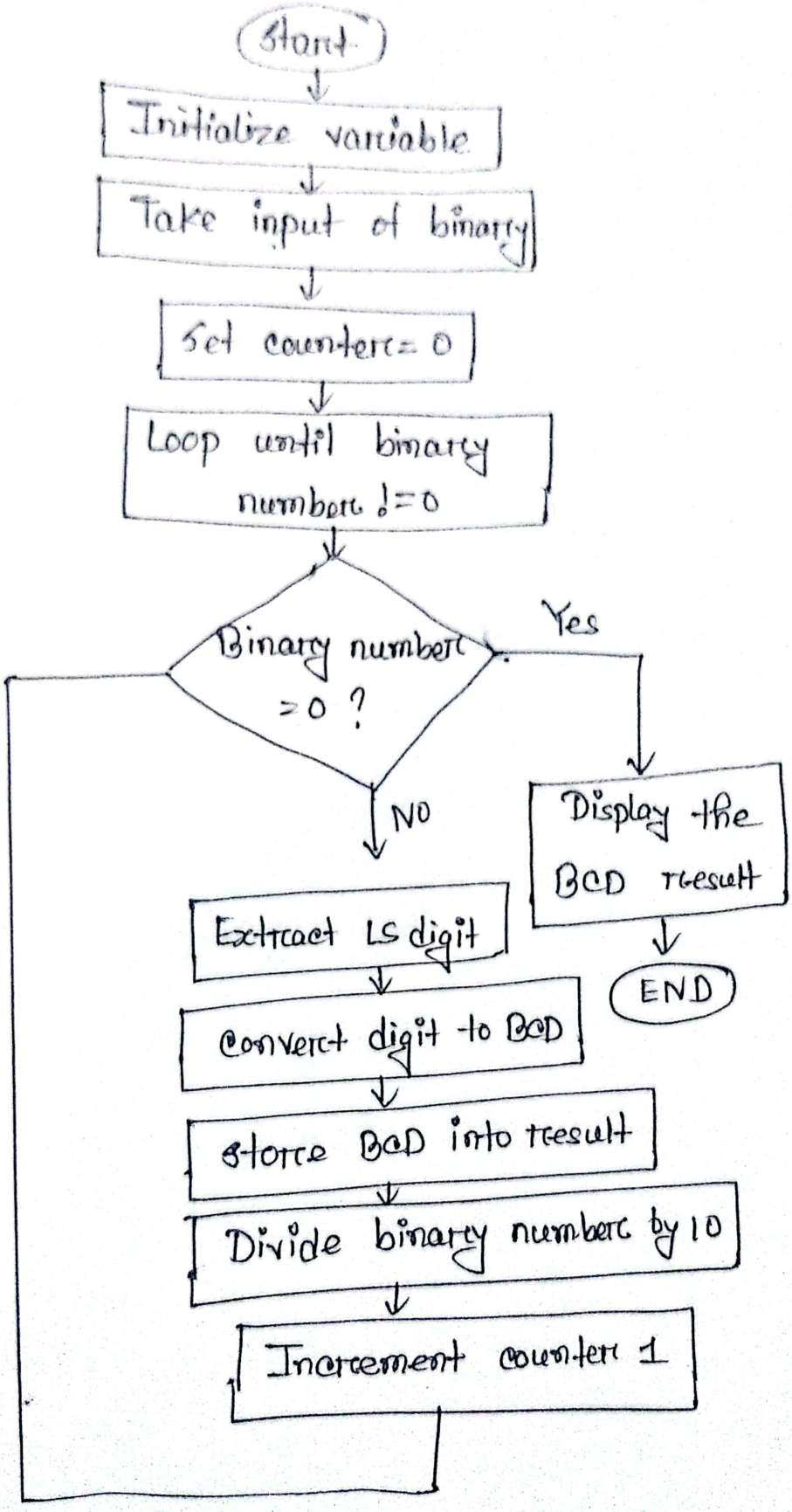
Experiment Name: Write an assembly language Program to convert a given binary to BCD

Apparatus: Computer with EMU8086 Software

Algorithm:

1. Initialize variable to store the binary number, BCD result and counter
2. Take input of binary number
3. Set counter 0
4. Perform the following steps until the binary number is not equal to zero
   - (a) Extract the least significant digit of binary number
   - (b) Convert the digit to BCD by Performing necessary calculation
   - (c) store the BCD digit into BCD result
   - (d) Divide the binary number by 10 to shift the digit
   - (e) Increment the counter
5. Display the result

Flowchart:


Start
Initialize variable
Take input of binary
Set counter= 0
Loop until binary number != 0
Binary number = 0 ?
Yes
NO
Display the BCD result
END
Extract LS digit
Convert digit to BCD
Store BCD into result
Divide binary number by 10
Increment counter 1

Program:-

```
.MODEL SMALL
.DATA
        BIN  DB  0FFH
        BCD  DB  2 DUP(0)
.CODE
        MOV  AX, @DATA
        MOV  DS, AX
        MOV  AL, BIN
        MOV  BL, AL
        MOV  CX, 0000H
CONTENT CMP AL, CL
        JE   NEXTI
        MOV  AL, 00H
BACK:   INC  CL
CONTENT ADD  AL, 01H
        DAA
        JNC  NEXT2
        PUSH AX
        MOV  AL, 00H
        ADC  AL, 00H
        DAA
        ADD  CH, AL
        POP  AX
NEXT2:  CMP  BL, CL
        JNZ  BACK
```

```
NEXT1:
        MOV  BCD, AL
        MOV  BCD+1, CH
        MOV  AH, 4CH
        INT  21H
        END
```

Result

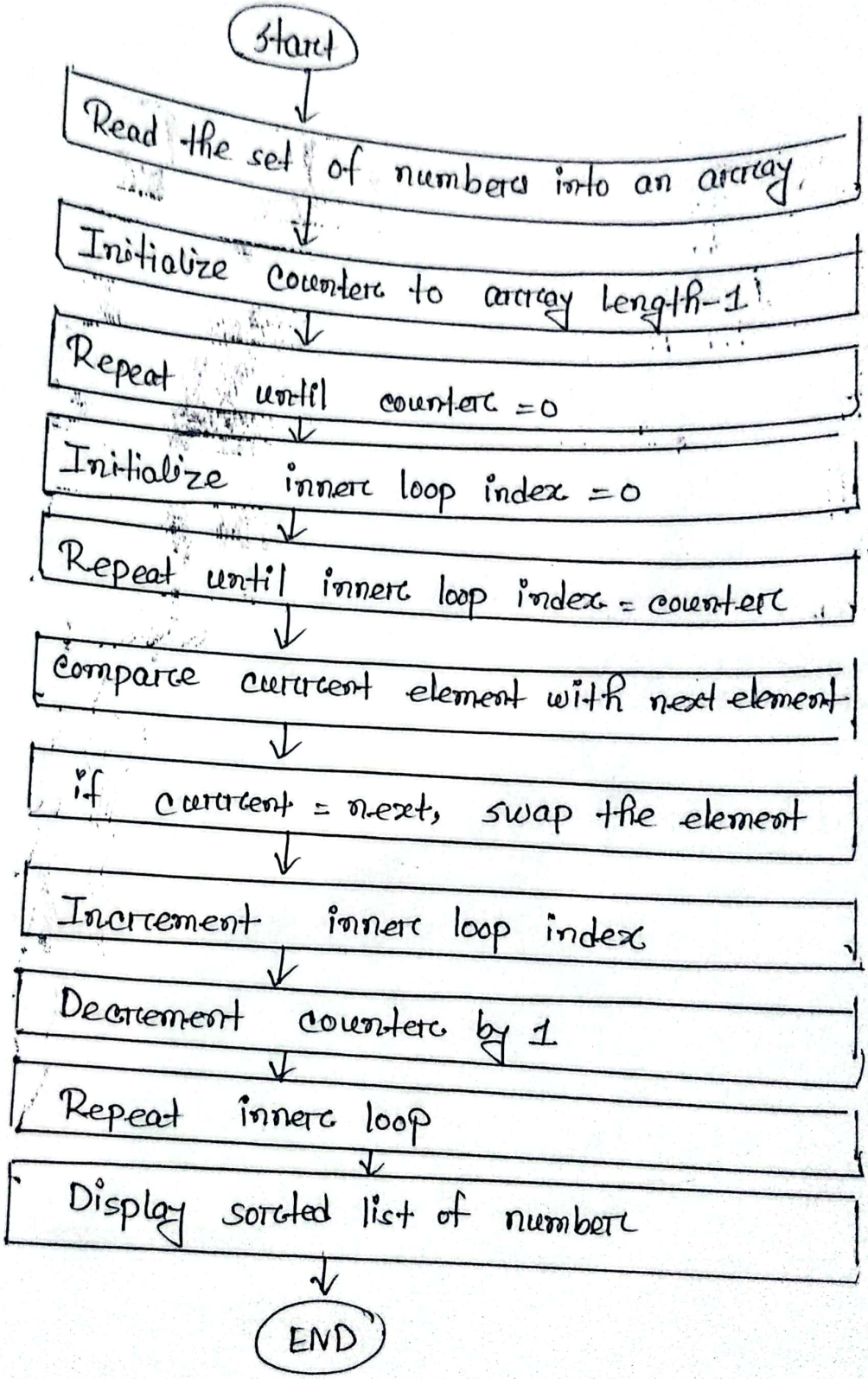
Experiment No-07

Experiment Name: Write an assembly language program to sort the set numbers in ascending order

Apparatus: Computers with EMU 8086

Algorithm:
1. Read the set of numbers into an array
2. Initialize a counter to the length of the array minus 1.
3. Repeat step 4 to 7 until the counter becomes zero
4. Initialize an inner loop index to zero
5. Repeat step 6 to 10 until the inner loop index reaches the counter
6. Compare the current element with the next element in the array
7. If the current element is greater than the next element, swap them
8. Decrement the counter by 1
9. Repeat step 5
10. Increment the inner loop index 1
11. Display the sorted set of numbers

Flowchart:


Start
Read the set of numbers into an array.
Initialize counter to array length-1
Repeat until counter =0
Initialize inner loop index =0
Repeat until inner loop index = counter
Compare current element with next element
if current > next, swap the element
Increment inner loop index
Decrement counter by 1
Repeat inner loop
Display sorted list of number
END

Programs:

```
.MODEL SMALL
.STACK
.DATA
    NUM DB 12H, 37H, 01H, 36H, 76H
    SIZE DB 5
.CODE
    MOV AX, @DATA
    MOV DS, AX
    LEA SI, NUM
    MOV CL, SIZE
    DEC CL
OUTER_LOOP:
        MOV CH, CL
        LEA DI, NUM
INNER_LOOP:
        MOV AL, [DI]
        MOV BL, [DI+1]
        CMP AL, BL
        JBE SKIP_SWAP

       MOV [DI], BL
       MOV [DI+1], AL

SKIP_SWAP:
          INC DI
         DEC CH
         JNZ INNER_LOOP
         DEC CL
         JNZ OUTER_LOOP
        MOV AH, 4CH
        INT 21H
END
```

Experiment No-08

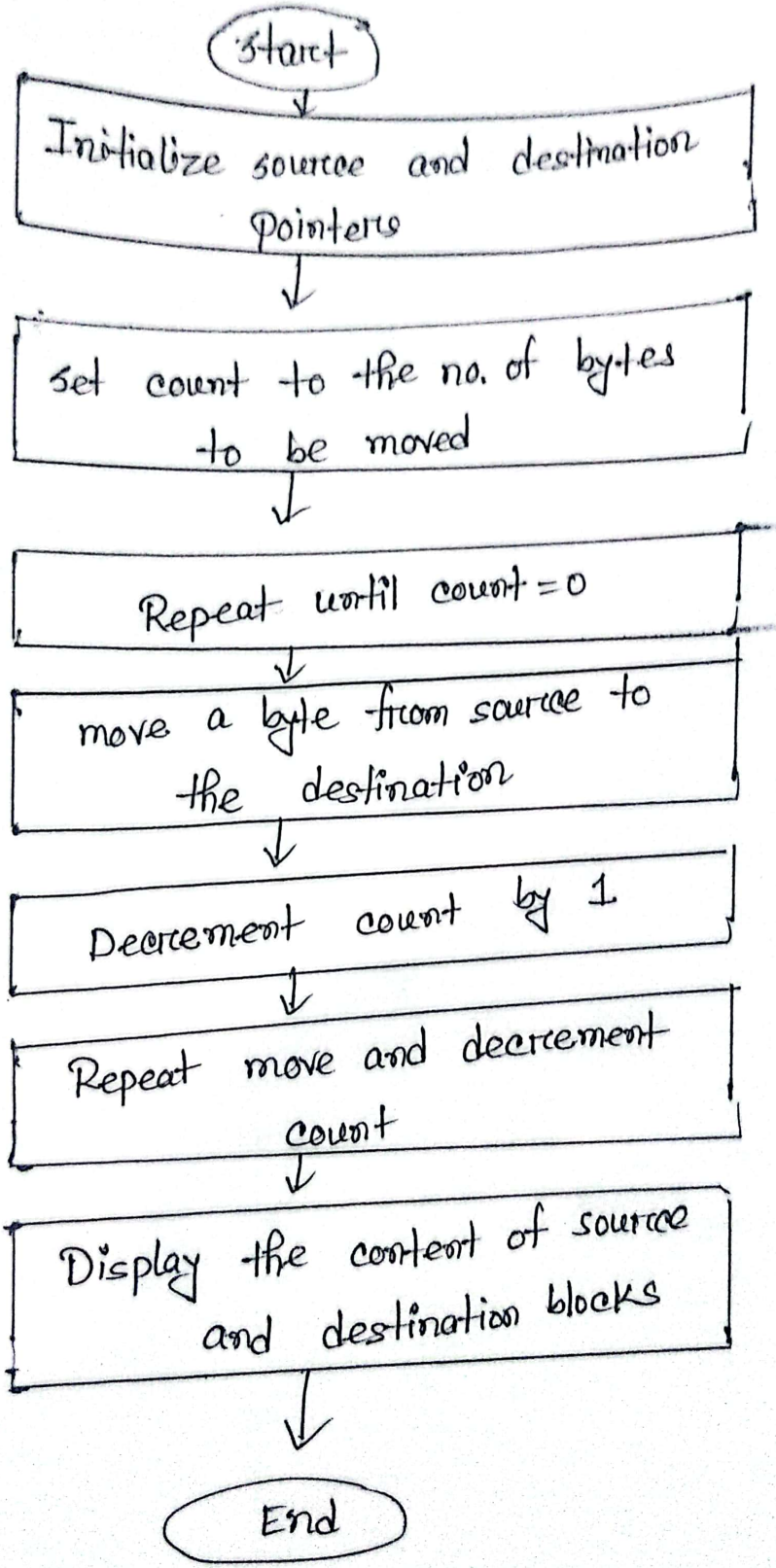
Experiment Name: Write an assembly language Program to move a block of data without overlap

Apparatus: Computer with EMU8086

Algorithm:

1. Initialize the source pointer to the starting address of the source block
2. Initialize the destination pointer to the starting address of the destination block
3. Set the count to the number of bytes to be moved
4. clear the direction flag using the CLD instruction to ensure forward movement
5. Use the MOVS instructions to move a byte from the source block to the destination block
6. Decrement the count by 1
7. Repeat step 5 and 6 until the count becomes zero
8. Display the counters of the source and destination block

Flowchart:


Start
Initialize source and destination Pointers
Set count to the no. of bytes to be moved
Repeat until count = 0
move a byte from source to the destination
Decrement count by 1
Repeat move and decrement count
Display the content of source and destination blocks
End

Program:-

```
.MODEL SMALL
.DATA
    BLK1 DB 01,02,03,04,05,06,07,08,09,0AH BLK2 DB 10
    DUP(0)
    COUNT DW 0AH
.CODE
    MOV Ax
    @DATA MOV
    DS, AX MOVES, Ax
    MOV SI, OFFSET BLK1
    MOV DI, OFFSET BLK2
   MOV CX, COUNT

AGAIN: CLD
       REP MOVSB
       MOV
```

Result:-

Experiment No-09

Experiment Name: Write an assembly language program to transfer a string in forward direction

Apparatus: Computer with EMU8086

Algorithm:-

1. Initialize the source pointer to the starting address of the source string
2. Initialize the destination pointer to the starting address of the destination string
3. Repeat the following steps until the end of the string reached:

a) Load a character from the source string using 'mov al, [SI]'

b) store the character in the destination string using 'mov [di] al'.

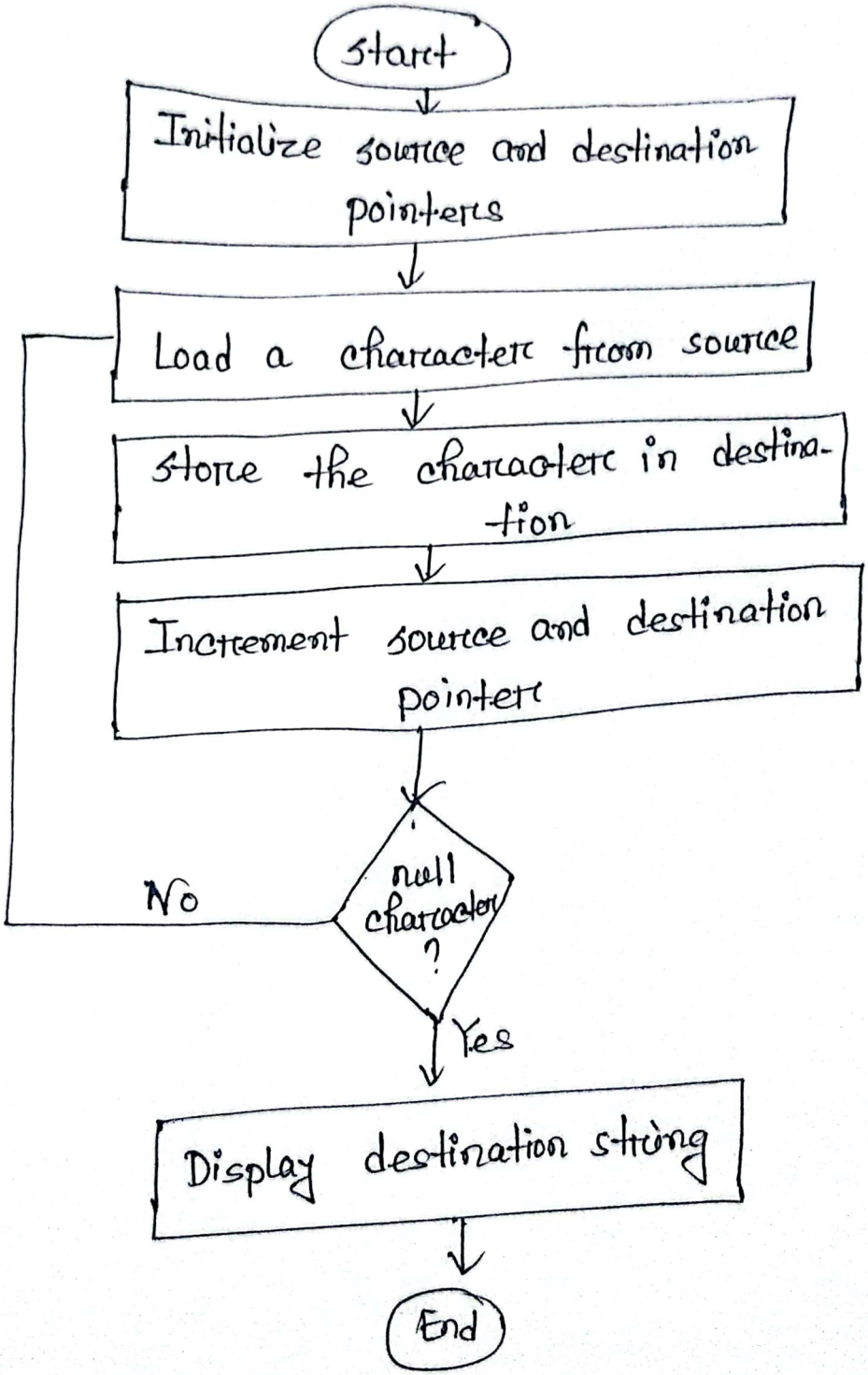
c) Increment the source pointer to point to the next character in the source string

d) Increment the destination pointer to point to the next position in the destination string

e) check if the loaded character is null character if not repeat the loop

4. Display the updated destination string

Flowchart:



Program:

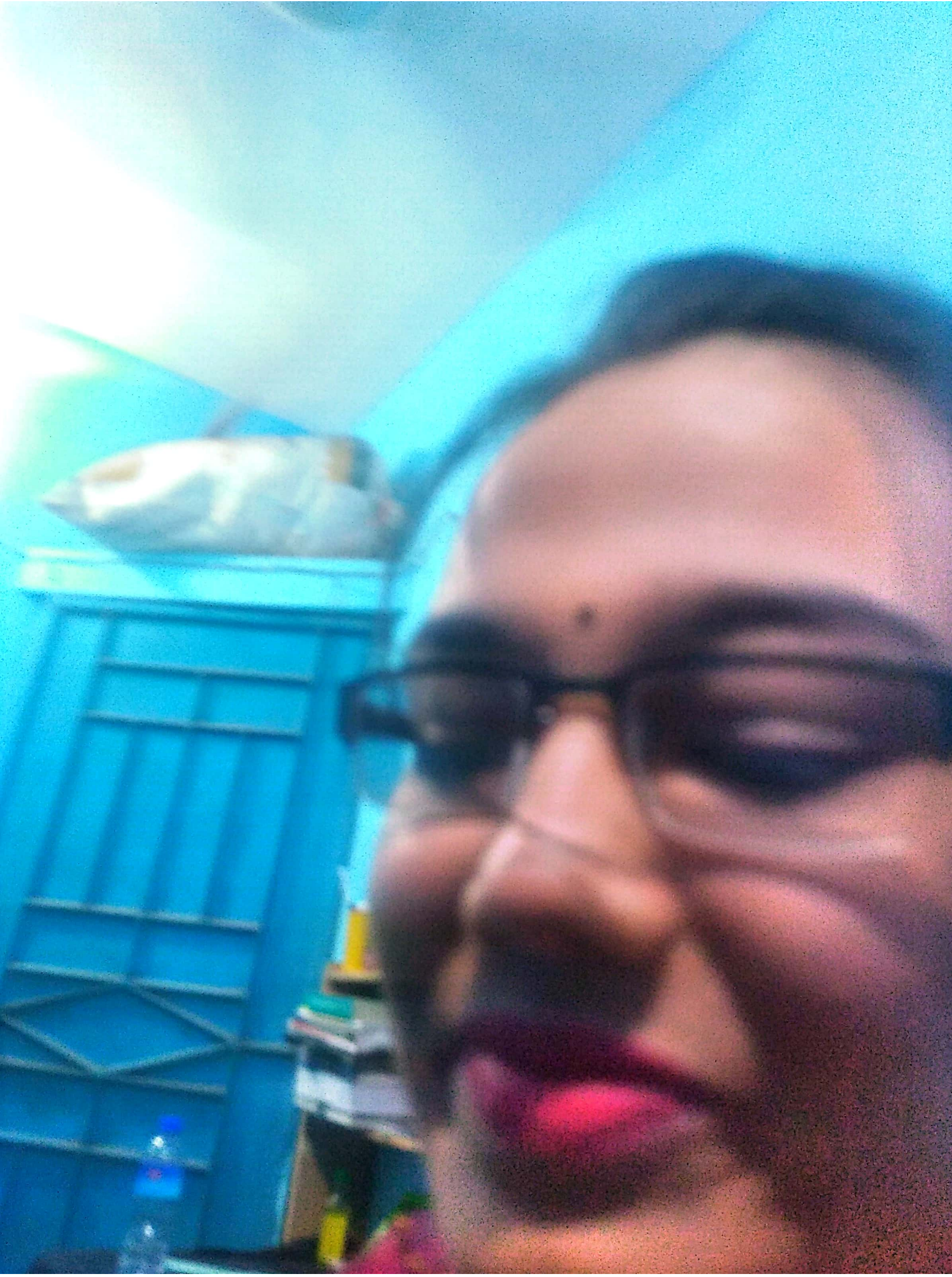
```
.MODEL SMALL
.DATA
  STR1 DB 09H, 'BANGLADESH', '$'
  STR2 DB ?
  STR1 DB 09H, 'STRING1: $'
  ST2 DB 09H, 'STRING2: $'
  LEN  DB 0FH
.CODE
MAIN PROC
     MOV AX, @DATA
     MOV DS, AX
     MOV ES, AX
     LEA SI, STR1
     LEA DI, STR2

  LEA DX, ST1
  MOV AH, 09H
  INT 21H

  LEA DX, STR1
  MOV AH, 09H
  INT 21H

  LEA DX, ST2
  MOV AH, 09H
  INT 21H

  CLD
  MOV CH, 00H
  MOV CL, LEN
  REP. MOVSB
```



```
LEA DX, STR2
MOV AH, 09H
INT 21H

MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN
```

```
LEA DX, STR2
MOV AH, 00H
INT 21H

MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN
```

Experiment No-10

Experiment Name: Write an assembly language program to multiply two 16-bit binary number to give a 32-bit result

Apparatus: Computer with EMU8086

Algorithm:

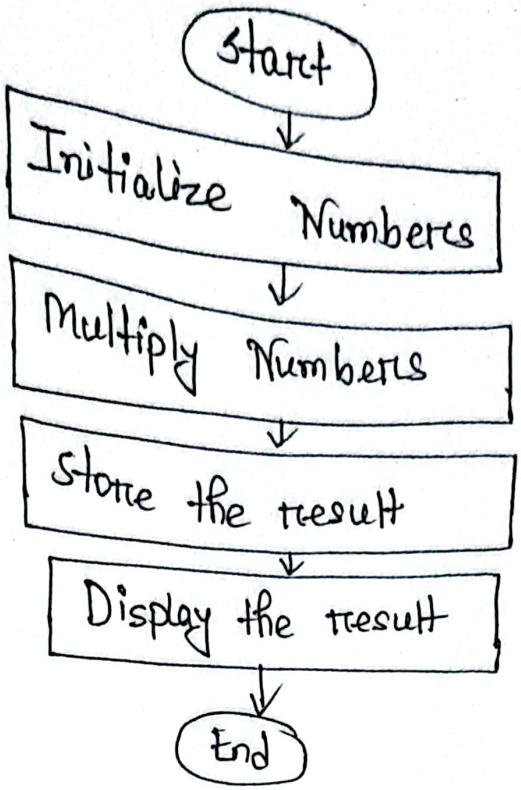
1. Initialize the first 16-bit number and the second 16-bit binary number

2. Multiply the two numbers using the mul instruction Load the first number into the AX register and use the second number as the operand for multiplication

3. Store the result in a 32-bit variable. Store the lower 16-bit of the result in the memory location pointed to by 'result' and store upper 16-bits in the memory location 'result+2'

4. Display the result

Flowchart:-



Program:-

```
.MODEL SMALL
.STACK
.DATA
    NUM1 DW
    NUM2 DW
    RESULT_HIGH DW ?
    RESULT_LOW DW ?

.CODE
    MOV AX,@DATA
    MOV DS, AX
    MOV AX, NUM1
    MOV CX, NUM2
    MUL CX
```

```
MOV  RESULT_LOW, AX
MOV  RESULT_HIGH, DX

MOV  AH, 4CH
INT  21H
END
```

Result:-

Experiment No-11

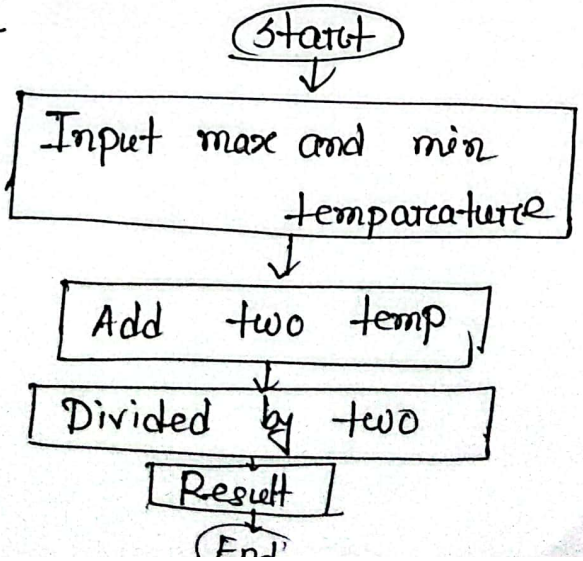
Experiment Name:- Write an assembly language program to find out the average two (max and low) temparatures

Apparatus:- Computer with EMU 8086

Algorithms:

1. Initialize maximum temparature, minimum temparature and average temparature
2. Calculate the average temparature by adding the maximum and minimum temparature and then dividing by 2
3. Display the average temparature

Flowchart:-



Program :-

```
.MODEL SMALL
.STACK
.DATA
    TEMP_MAX  DW
    TEMP_MIN  DW
    TEMP_AVG  DW ?
.CODE
    MOV AX,@DATA
    MOV DS,AX
    MOV AX, TEMP_MAX
    ADD AX, TEMP_MIN
    MOV CX, 2
    MOV DX, 0
    DIV CX
    MOV TEMP_AVG, AX
    MOV AH, 4CH
    INT 21H
END
```

Result :

Experiment No-12

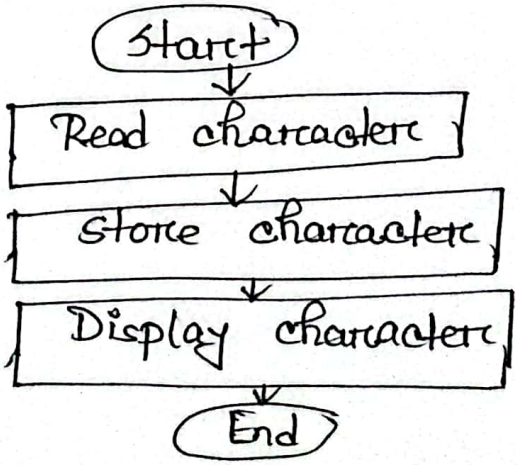
Experiment Name:- Write an assembly language program to read a character from keyboard

Apparatus:- Computer with EMU8086

Algorithm:-

1. Initialize a variable to store the input character
2. Read a character from the keyboard using the DOS interrupt 'int 21H' with function '01H'
3. Store the input character in the designated variable
4. Display the input character using the DOS interrupt 'int 21H' with function '02H'
5. Terminate the Program

Flowchart:-



Program:

```
.MODEL SMALL
.STACK
.CODE

MAIN PROC

    PRINT 'ENTER A CHARACTER: '
    MOV AH, 1
    INT 21H
    MOV BL, AL

    MOV AH, 2
    MOV DL, 10
    INT 21H
    MOV DL, 13
    INT 21H

PRINT 'OUTPUT:'
MOV AH, 2
MOV DL, BL
INT 21H

exit: MOV AH, 4CH
      INT 21H

   MAIN ENDP
END MAIN
```

Experiment No-13

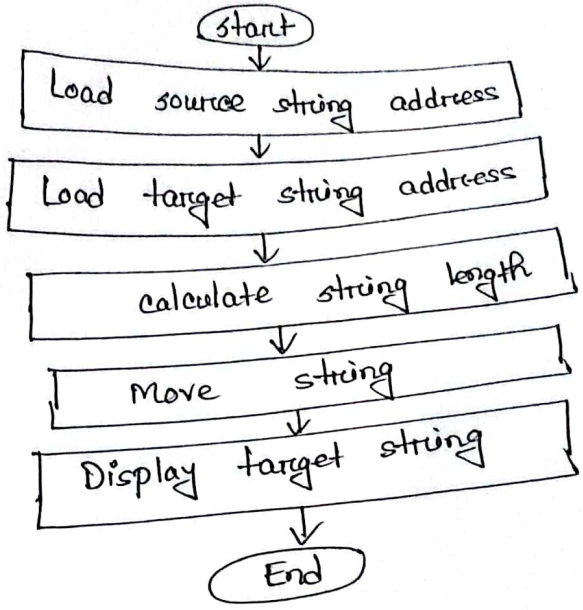
Experiment Name:- Write an assembly language program for moving string from one location to another in memory

Apparatus:- Computer with EMU8086

Algorithm:-

1. Initialize the source string and data string in the data segment
2. Load the address of the source string into the SI register using the 'lea' instruction
3. Load the address of the target string into the DI register using the 'lea' instruction
4. Calculate the length of the target string into the DI register using the 'lea' instruction
5. use 'rep movsb' instruction to move the string byte by byte from the source to the target
6. Display the target string using DOS interrupt 'int 21h' with function '09h'

Flowchart:-



Program:-

```
.MODEL SMALL
.STACK
.DATA
    SOURCE DB
    DESTINATION DB 20 DUP( )

.CODE
    MOV AX, @DATA
    MOV DS, AX
    MOV ES, AX
```

```
LEA SI, SOURCE
LEA DI, DESTINATION

MOV CX, 13
CLD
REP MOVSB

MOV AH, 4CH
INT 21H

END
```

Result: