VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Experiment 8:
Basic Programming with 8087

Programme	:	**BTech. CSE Core**	Semester	:	**Win 2021-22**
Course	:	**Microprocessor and Interfacing**	Code	:	**CSE2006**
Faculty	:	**Dr. Florence Gnana Poovathy J**	Slot	:	**L15+L16**
Name	:	**Hariket Sukesh Kumar Sheth**	Register No.	:	**20BCE1975**

Basic Programming with 8087

Aim: To calculate the area of the square by taking the side as the input

Tool Used: Assembler – MASM611

Algorithm:

Step 1: First of all, mount the c drive using the command: **mount c c:\masm611\bin**

Step 2: After pressing **enter,** type **c:** and press enter.

Step 3: Now give a command, **<filename>.asm** for writing/editing the code and the write the code.

Step 4: A pop window appears; there we have to write out code(instructions) following the logic given below.

- Two variables are defined, one to hold the value of side, and the other will be used to store the value of area after calculation
- The data is moved to DS register, via AX register.
- FINIT signals the initialization of 8087 commands and registers.
- The variable SIDE is loaded onto the stack top ST(0) via FLD.
- The value is also stored in ST(4) via FST.
- The values at ST(0) and ST(4) are multiplied via FMUL, and the product is stored at ST(0).
- The product is then copied to the variable SQAR, via FST.

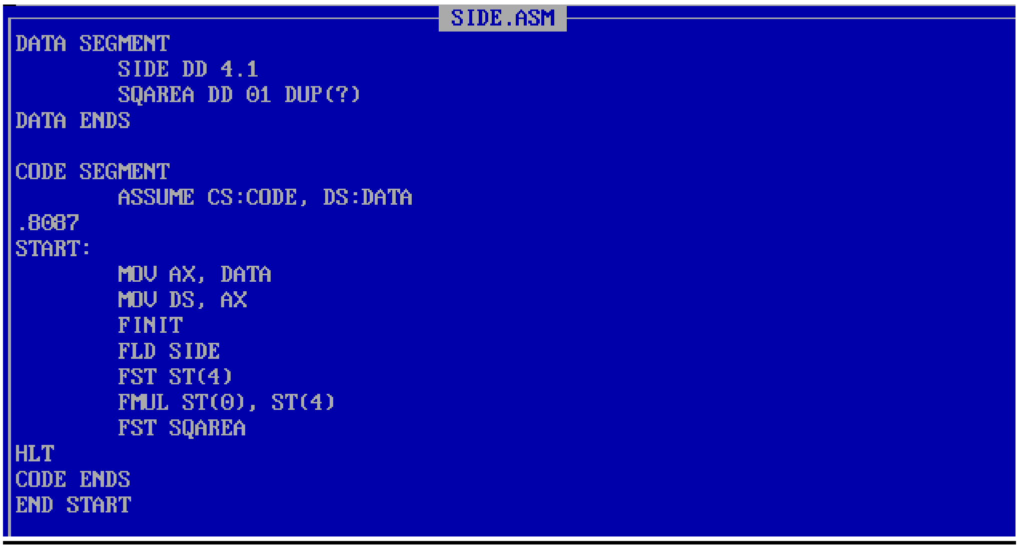
Step 5: Now give a command, **masm <filename>.asm** for running the code. The object file is created.

Step 6: Now give a command, **link <filename>.obj** to link the object file to library file present in the bin folder.

Step 7: Press **ENTER** four times.

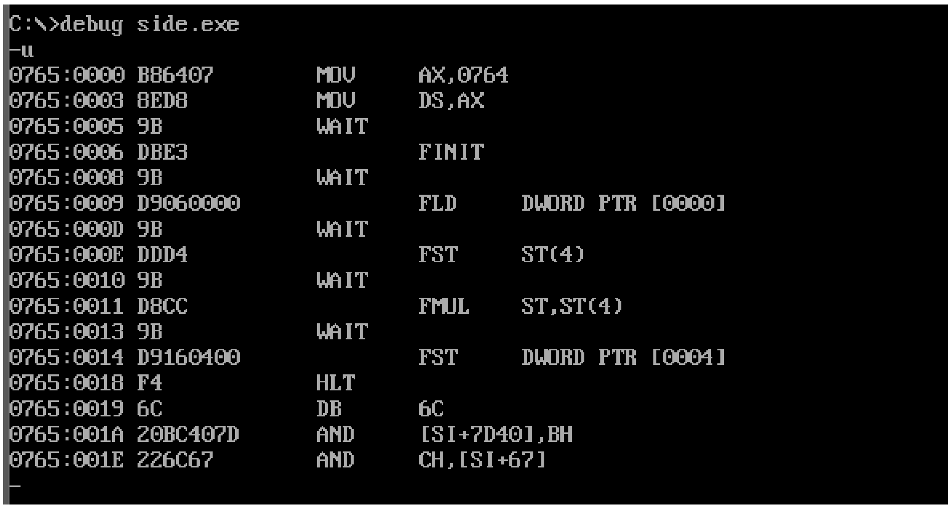
Step 8: Write **<filename>.exe**

PROGRAM:



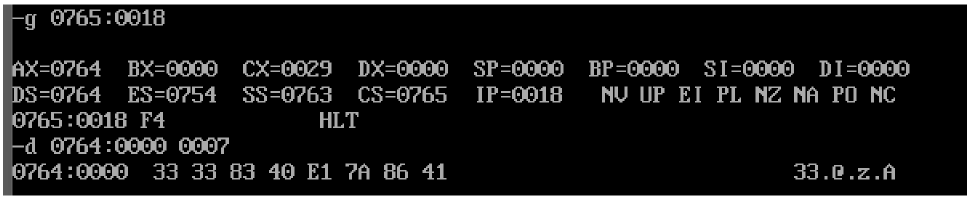
```
DATA SEGMENT
        SIDE DD 4.1
        SQAREA DD 01 DUP(?)
DATA ENDS

CODE SEGMENT
        ASSUME CS:CODE, DS:DATA
.8087
START:
        MOV AX, DATA
        MOV DS, AX
        FINIT
        FLD SIDE
        FST ST(4)
        FMUL ST(0), ST(4)
        FST SQAREA
HLT
CODE ENDS
END START
```

OUTPUT:



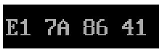
```
C:\>debug side.exe
-u
0765:0000 B86407        MOV     AX,0764
0765:0003 8ED8          MOV     DS,AX
0765:0005 9B            WAIT
0765:0006 DBE3                  FINIT
0765:0008 9B            WAIT
0765:0009 D9060000              FLD     DWORD PTR [0000]
0765:000D 9B            WAIT
0765:000E DDD4                  FST     ST(4)
0765:0010 9B            WAIT
0765:0011 D8CC                  FMUL    ST,ST(4)
0765:0013 9B            WAIT
0765:0014 D9160400              FST     DWORD PTR [0004]
0765:0018 F4            HLT
0765:0019 6C            DB      6C
0765:001A 20BC407D      AND     [SI+7D40],BH
0765:001E 226C67        AND     CH,[SI+67]
-
```

REGISTER / MEMORY CONTENTS



```
-g 0765:0018

AX=0764  BX=0000  CX=0029  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0764  ES=0754  SS=0763  CS=0765  IP=0018   NV UP EI PL NZ NA PO NC
0765:0018 F4            HLT
-d 0764:0000 0007
0764:0000  33 33 83 40 E1 7A 86 41                           33.@.z.A
```

We would get the result by reversing this



```
E1 7A 86 41
```

Actual = 41 86 7A E1

Sample Input	**Sample Output**
Side = 4.1	Hexadecimal Result = E1 7A 86 41 Actual Result = 41 86 7A E1 **Result = 16.81**
Side = 2.9061	Hexadecimal Result = 6E 20 07 41 Actual Result = 41 07 20 6E **Result = 8.44542**