SSN College of Engineering Department of Computer Science and Engineering

III year - UCS1512 – Microprocessors Lab

Floating point operations

Exp No: 09

Name: Rahul Ram M

Register Number: 185001121

Date: 13/10/2020

9a) Floating point addition:

Aim:

To design 8086-program for floating point addition.

Procedure for executing MASM:

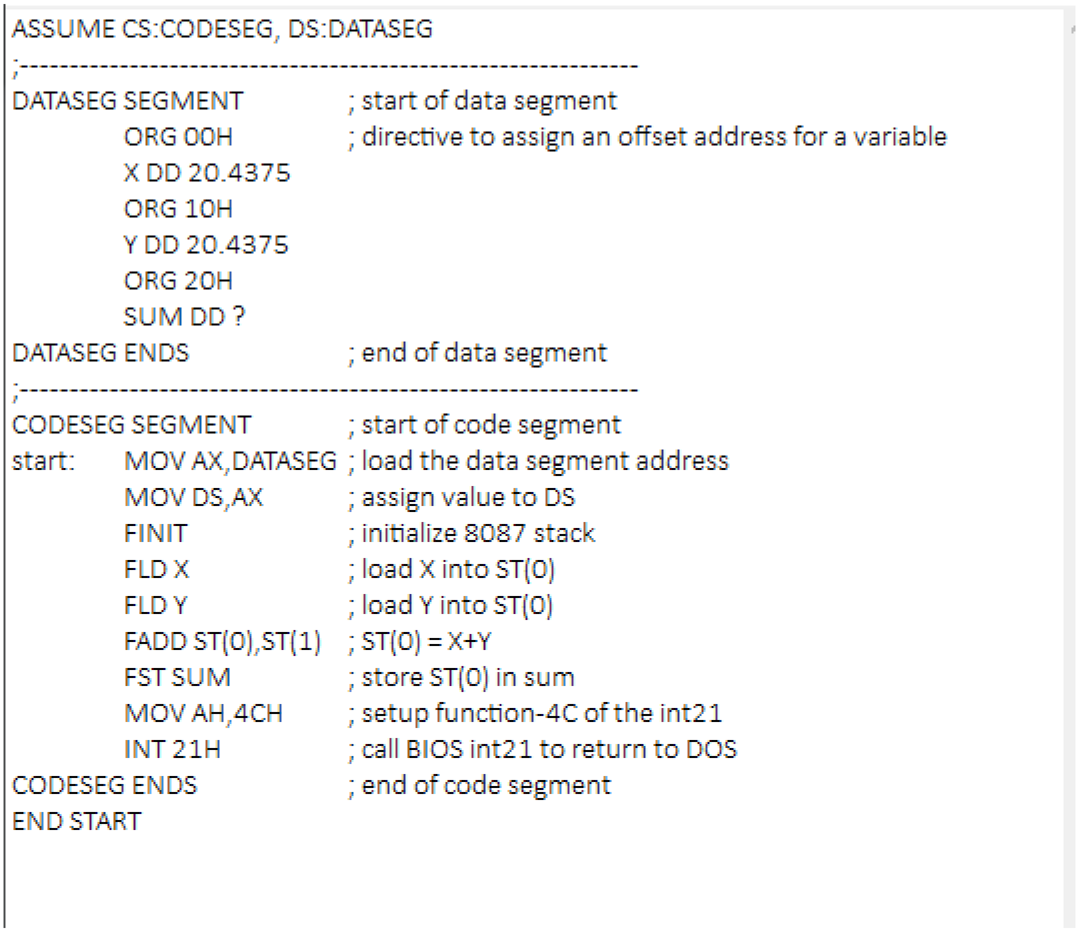
1. Run Dosbox and mount your masm folder to a drive in dosbox.
2. Goto the mounted drive.
3. Save the 8086 program with extension .asm in the same folder using command “edit”
4. After creating the file, assemble it using the command “masm filename.asm”
5. Link the file using the command “link filename.obj;”
6. Use debug command with filename.exe to execute and analyse the memory contents, “debug filename.exe”.
7. In debug, command “u” will display the unassembled code.
8. Use command “d segment:offset” to see the content of memory locations starting from segment:offset address.
9. To change the value in memory, use the command “e segment:offset”
10. Verify the memory contents to ensure the updates (using command “d”).
11. . Execute using the command “g” and check the outputs.
12. “q” to exit from debug and “exit” to exit from command prompt and to close the Dosbox.

Algorithm:

1. START: Move the starting address of data segment to AX register and move the data from AX register to DS register.
2. Initialize the 8087-stack using FINIT command. This stack will be used for floating point operations.
3. Load the floating-point number from variable X to the top of the stack i.e. ST (0) using FLD command.

4. Now again load the floating-point number from variable Y to the top of the stack i.e. ST (0) using FLD command. The previous stack top contents will be pushed into the stack.
5. Using FADD add ST(0) and ST(1) which stores the result in ST(0).
6. Using FST store the resulting floating-point number from the top of the stack to the variable SUM.
7. Move the hexadecimal value 4C into AH register. INT 21H means invoke the interrupt identified by the hexadecimal number 21. In MS-DOS, invoking interrupt 21h while AH = 4Ch causes the current process to terminate and uses the value of register AL as the exit code of the process.

Program:

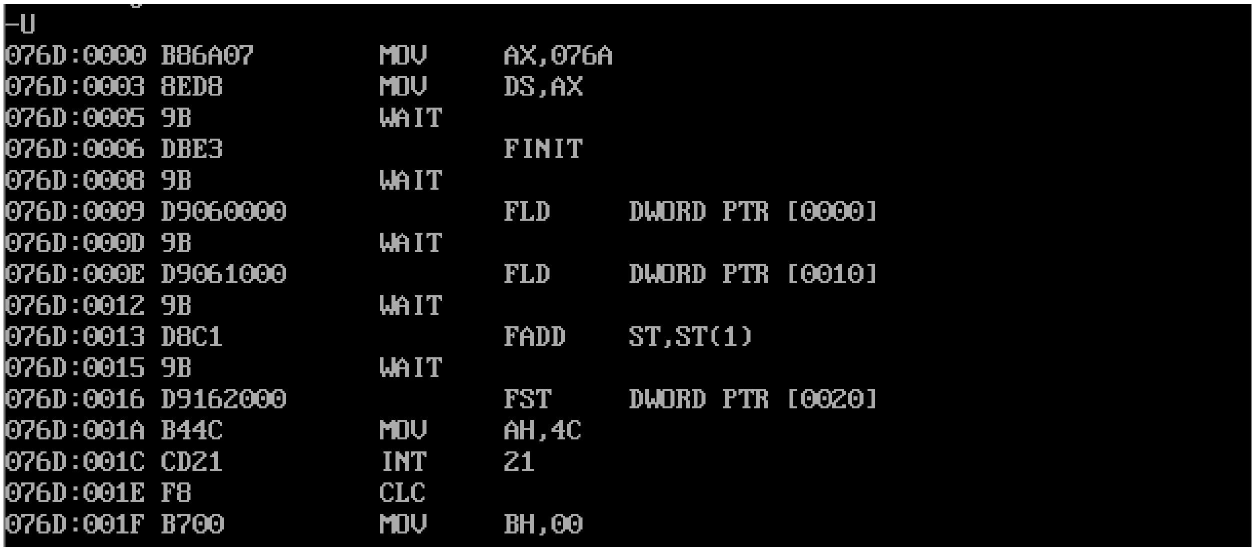


```
ASSUME CS:CODESEG, DS:DATASEG
;-----------------------------------------------------------
DATASEG SEGMENT             ; start of data segment
        ORG 00H             ; directive to assign an offset address for a variable
        X DD 20.4375
        ORG 10H
        Y DD 20.4375
        ORG 20H
        SUM DD ?
DATASEG ENDS                ; end of data segment
;-----------------------------------------------------------
CODESEG SEGMENT             ; start of code segment
start:  MOV AX,DATASEG      ; load the data segment address
        MOV DS,AX           ; assign value to DS
        FINIT               ; initialize 8087 stack
        FLD X               ; load X into ST(0)
        FLD Y               ; load Y into ST(0)
        FADD ST(0),ST(1)    ; ST(0) = X+Y
        FST SUM             ; store ST(0) in sum
        MOV AH,4CH          ; setup function-4C of the int21
        INT 21H             ; call BIOS int21 to return to DOS
CODESEG ENDS                ; end of code segment
END START
```

	Program	Comments
START:	MOV AX, DATA MOV DS, AX	Transferring the data from DATA to AX register and from AX register to DS register.
	FINIT	Initialize 8087 stack.
	FLD X	load X into ST(0)
	FLD Y	load Y into ST(0)
	FADD ST(0), ST(1)	ST(0) = X+Y
	FST SUM	Store ST(0) in sum.
	MOV AH,4CH	Setup function-4C of the int21.

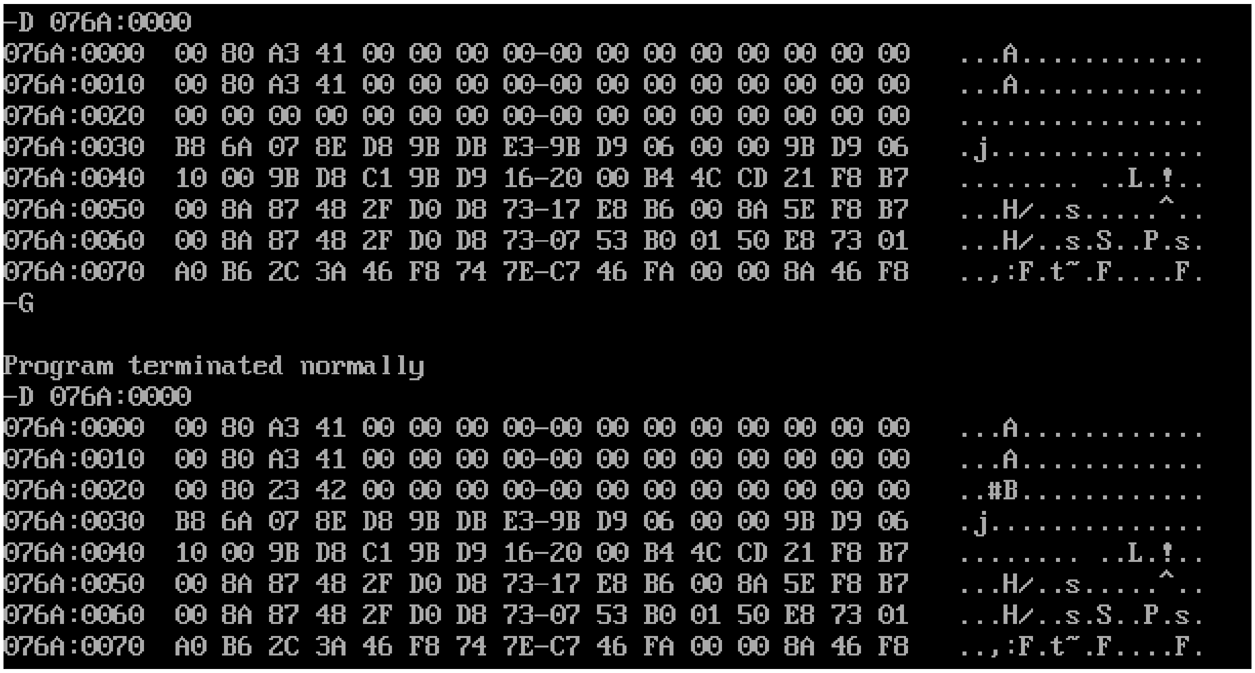
	INT 21H	Call BIOS int21 to return to DOS.

Unassembled Code:



```
-U
076D:0000 B86A07        MOV     AX,076A
076D:0003 8ED8          MOV     DS,AX
076D:0005 9B            WAIT
076D:0006 DBE3                  FINIT
076D:0008 9B            WAIT
076D:0009 D9060000              FLD     DWORD PTR [0000]
076D:000D 9B            WAIT
076D:000E D9061000              FLD     DWORD PTR [0010]
076D:0012 9B            WAIT
076D:0013 D8C1                  FADD    ST,ST(1)
076D:0015 9B            WAIT
076D:0016 D9162000              FST     DWORD PTR [0020]
076D:001A B44C          MOV     AH,4C
076D:001C CD21          INT     21
076D:001E F8            CLC
076D:001F B700          MOV     BH,00
```

Snapshot of sample input and output:



```
-D 076A:0000
076A:0000  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0010  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  B8 6A 07 8E D8 9B DB E3-9B D9 06 00 00 9B D9 06   .j..............
076A:0040  10 00 9B D8 C1 9B D9 16-20 00 B4 4C CD 21 F8 B7   ........ ..L.!..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-G

Program terminated normally
-D 076A:0000
076A:0000  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0010  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0020  00 80 23 42 00 00 00 00-00 00 00 00 00 00 00 00   ..#B............
076A:0030  B8 6A 07 8E D8 9B DB E3-9B D9 06 00 00 9B D9 06   .j..............
076A:0040  10 00 9B D8 C1 9B D9 16-20 00 B4 4C CD 21 F8 B7   ........ ..L.!..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Result:

Thus the 8086 program for floating-point addition is executed successfully in DOS-BOX

9b) Floating point subtraction:

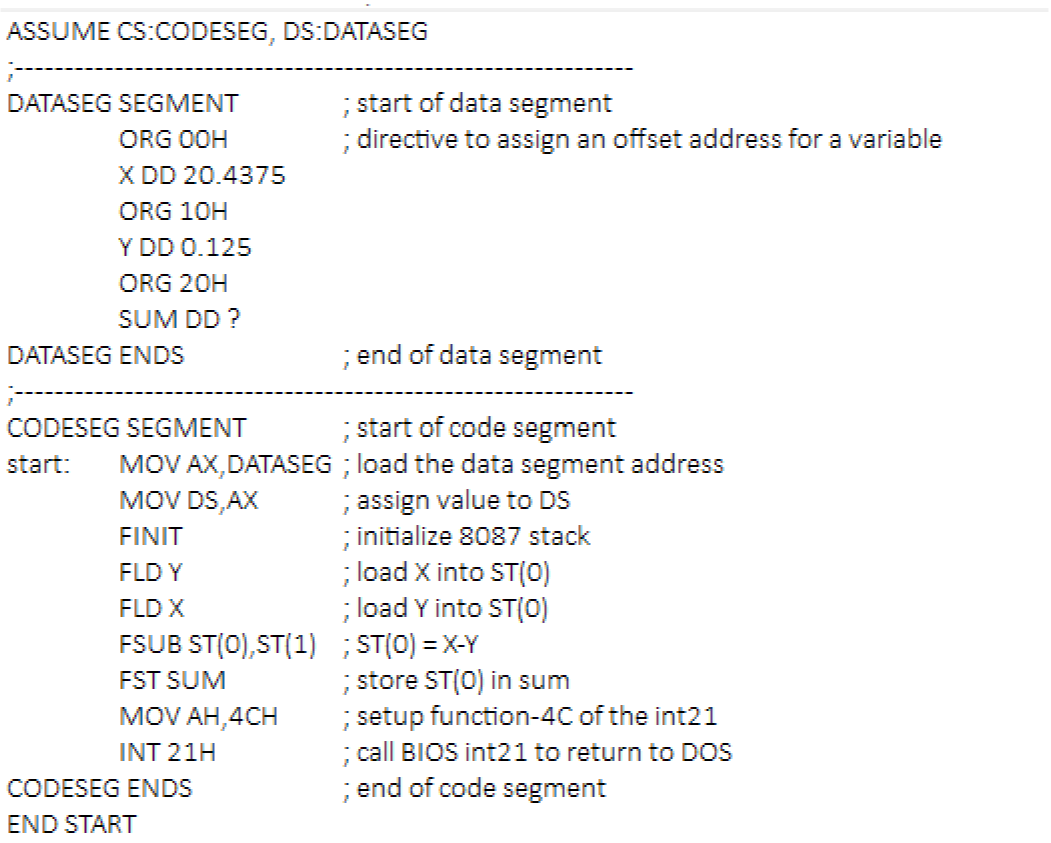
Aim:

To design 8086-program for floating point subtraction.

Algorithm:

8. START: Move the starting address of data segment to AX register and move the data from AX register to DS register.
9. Initialize the 8087-stack using FINIT command. This stack will be used for floating point operations.
10. Load the floating-point number from variable X to the top of the stack i.e. ST (0) using FLD command.
11. Now again load the floating-point number from variable Y to the top of the stack i.e. ST (0) using FLD command. The previous stack top contents will be pushed into the stack.
12. Using FSUB sub ST(0) from ST(1) and store the result in ST(0).
13. Using FST store the resulting floating-point number from the top of the stack to the variable SUM.
14. Move the hexadecimal value 4C into AH register. INT 21H means invoke the interrupt identified by the hexadecimal number 21. In MS-DOS, invoking interrupt 21h while AH = 4Ch causes the current process to terminate and uses the value of register AL as the exit code of the process.

Program:

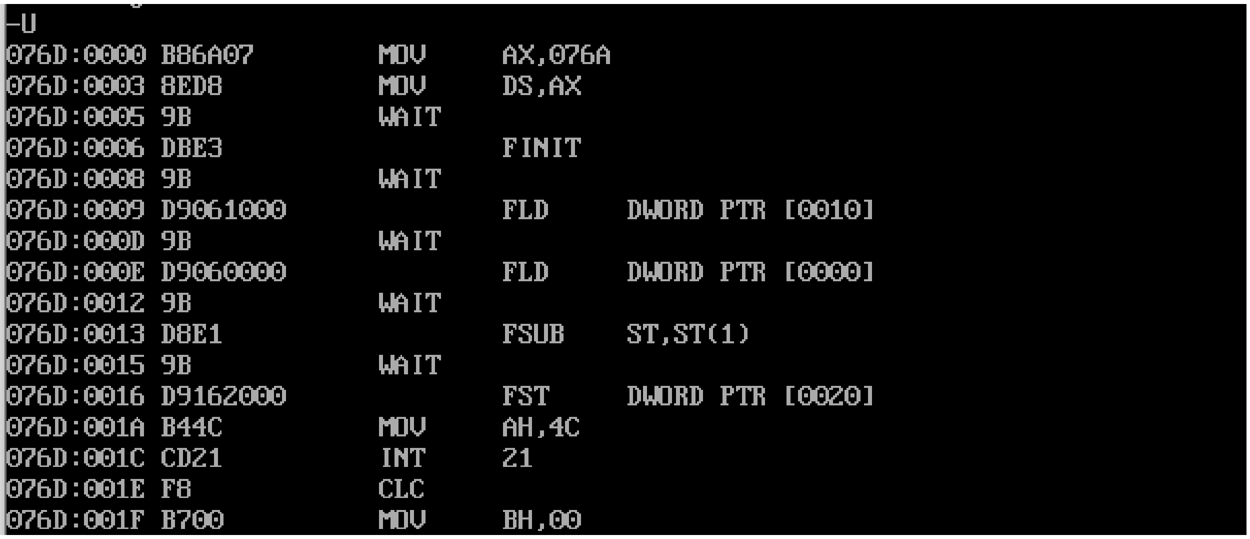


```
ASSUME CS:CODESEG, DS:DATASEG
;------------------------------------------------------------
DATASEG SEGMENT             ; start of data segment
        ORG 00H             ; directive to assign an offset address for a variable
        X DD 20.4375
        ORG 10H
        Y DD 0.125
        ORG 20H
        SUM DD ?
DATASEG ENDS                ; end of data segment
;------------------------------------------------------------
CODESEG SEGMENT             ; start of code segment
start:  MOV AX,DATASEG      ; load the data segment address
        MOV DS,AX           ; assign value to DS
        FINIT               ; initialize 8087 stack
        FLD Y               ; load X into ST(0)
        FLD X               ; load Y into ST(0)
        FSUB ST(0),ST(1)    ; ST(0) = X-Y
        FST SUM             ; store ST(0) in sum
        MOV AH,4CH          ; setup function-4C of the int21
        INT 21H             ; call BIOS int21 to return to DOS
CODESEG ENDS                ; end of code segment
END START
```

	Program	Comments
START:	MOV AX, DATA	Transferring the data from DATA to AX register and

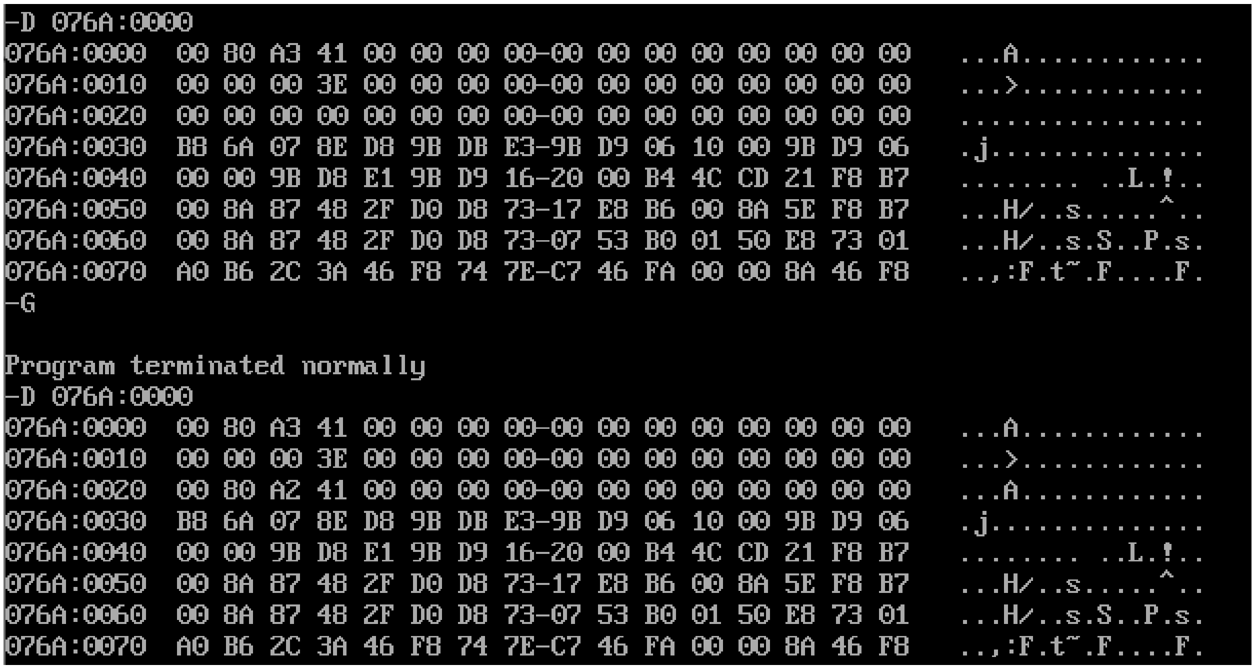
	MOV DS, AX	from AX register to DS register.
	FINIT	Initialize 8087 stack.
	FLD X	load X into ST(0)
	FLD Y	load Y into ST(0)
	FSUB ST(0), ST(1)	ST(0) = X-Y
	FST SUM	Store ST(0) in sum.
	MOV AH,4CH	Setup function-4C of the int21.
	INT 21H	Call BIOS int21 to return to DOS.

Unassembled Code:



```
-U
076D:0000 B86A07        MOV     AX,076A
076D:0003 8ED8          MOV     DS,AX
076D:0005 9B            WAIT
076D:0006 DBE3                  FINIT
076D:0008 9B            WAIT
076D:0009 D9061000              FLD     DWORD PTR [0010]
076D:000D 9B            WAIT
076D:000E D9060000              FLD     DWORD PTR [0000]
076D:0012 9B            WAIT
076D:0013 D8E1                  FSUB    ST,ST(1)
076D:0015 9B            WAIT
076D:0016 D9162000              FST     DWORD PTR [0020]
076D:001A B44C          MOV     AH,4C
076D:001C CD21          INT     21
076D:001E F8            CLC
076D:001F B700          MOV     BH,00
```

Snapshot of sample input and output:



```
-D 076A:0000
076A:0000  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0010  00 00 00 3E 00 00 00 00-00 00 00 00 00 00 00 00   ...>............
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  B8 6A 07 8E D8 9B DB E3-9B D9 06 10 00 9B D9 06   .j..............
076A:0040  00 00 9B D8 E1 9B D9 16-20 00 B4 4C CD 21 F8 B7   ........ ..L.!..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
-G

Program terminated normally
-D 076A:0000
076A:0000  00 80 A3 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0010  00 00 00 3E 00 00 00 00-00 00 00 00 00 00 00 00   ...>............
076A:0020  00 80 A2 41 00 00 00 00-00 00 00 00 00 00 00 00   ...A............
076A:0030  B8 6A 07 8E D8 9B DB E3-9B D9 06 10 00 9B D9 06   .j..............
076A:0040  00 00 9B D8 E1 9B D9 16-20 00 B4 4C CD 21 F8 B7   ........ ..L.!..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Result:

Thus the 8086-program for floating-point subtraction is executed successfully in DOS-BOX