Exp No: 9 **Date:** 19/10/2020

FLOATING POINT OPERATIONS

Name: Swetha Saseendran

Reg No: 185001183

Aim:

To write assembly language programs to perform the following floating-point arithmetic:

1. Floating point Addition.
2. Floating point Subtraction

Programs:

(i) FLOATING POINT ADDITION

Algorithm:

- Declare the data segment.
- Initialize data segment with the 2 floating point numbers and a variable for storing their sum.
- Close the data segment.

- Declare the code segment.
- Set a preferred offset (preferably 100h)
- Load the data segment content into AX register.
- Transfer the contents of AX register to DS register.
- Initialize Floating point operation using FINIT.
- Move the contents of the two numbers into the stack ST.
- Add them and store the value in top of the stack.
- Move the content in top of the stack to variable ‘sum’.
- Introduce an interrupt for safe exit. (INT 21h)
- Close the code segment.

PROGRAM	COMMENTS
assume *cs:code, ds:data*	Declare code and data segment.
data segment	Initialize data segment with values.
org 00h	Directive to assign an offset address for a variable.
x dd 20.4375	Stores the first number.
org 10h	
y dd 20.4375	Stores the second number.
org 20h	
sum dd ?	Variable to store the value of the sum.
data ends	End of data segment.
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: *mov ax, data*	Transfer data from "data" to AX.
mov ds, ax	Transfer data from memory location AX to DS.
finit	Initialize 8087's stack.
fld x	Load 'x' into ST(0).
fld y	Load 'y' into ST(0).
fadd ST(0), ST(1)	ST(0) = ST(0) + ST(1)
fst sum	Store the value of sum in the variable 'sum'.
break*: mov ah, 4ch*	Moves the hexadecimal value 4c to ah.
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).
code ends	
end start	

Unassembled Code:

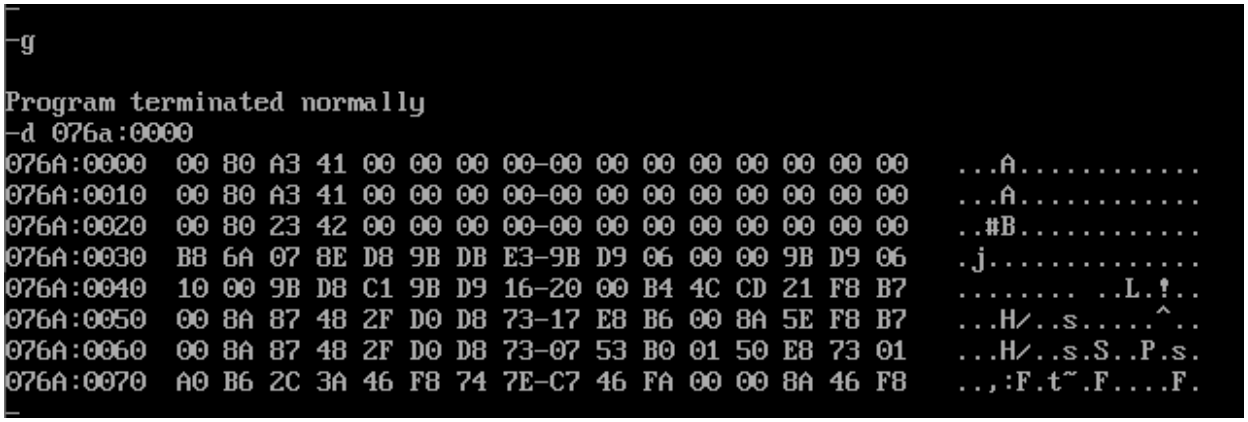
```
-u
076D:0000 B86A07        MOV     AX,076A
076D:0003 8ED8          MOV     DS,AX
076D:0005 9B            WAIT
076D:0006 DBE3                  FINIT
076D:0008 9B            WAIT
076D:0009 D9060000              FLD     DWORD PTR [0000]
076D:000D 9B            WAIT
076D:000E D9061000              FLD     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:

INPUT:

```
-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.
-
```

OUTPUT:



(ii) FLOATING POINT SUBTRACTION:

Algorithm:

- Declare the data segment.
- Initialize data segment with the 2 floating point numbers and variables for storing their difference diff.
- Close the data segment.
- Declare the code segment.
- Set a preferred offset (preferably 100h)
- Load the data segment content into AX register.
- Transfer the contents of AX register to DS register.
- Initialize Floating point operation using FINIT.
- Move the contents of the two numbers into the stack ST.
- Subtract them and store the value in top of the stack.
- Move the content in top of the stack to variable ‘diff’.
- Introduce an interrupt for safe exit. (INT 21h)
- Close the code segment.

PROGRAM	COMMENTS
assume *cs:code, ds:data*	Declare code and data segment.
data segment	Initialize data segment with values.
org 00h	Directive to assign an offset address for a variable.
x dd 20.4375	Stores the first number.
org 10h	
y dd 20.4375	Stores the second number.
org 20h	
diff dd ?	Variable to store the value of the difference.
data ends	End of data segment.
code segment	Start the code segment.
org 0100h	Initialize an offset address.
start: *mov ax, data*	Transfer data from "data" to AX.
mov ds, ax	Transfer data from memory location AX to DS.
finit	Initialize 8087's stack.
fld x	Load 'x' into ST(0).
fld y	Load 'y' into ST(0).
fsub ST(0), ST(1)	ST(0) = ST(0) - ST(1)
fst diff	Store the value of sum in the variable 'diff'.
break: *mov ah, 4ch*	Moves the hexadecimal value 4c to ah.
int 21h	When Software interrupt 21 is called with AH=4C, then current process terminates. (i.e., These two instructions are used for the termination of the process).
code ends	
end start	

Unassembled Code:

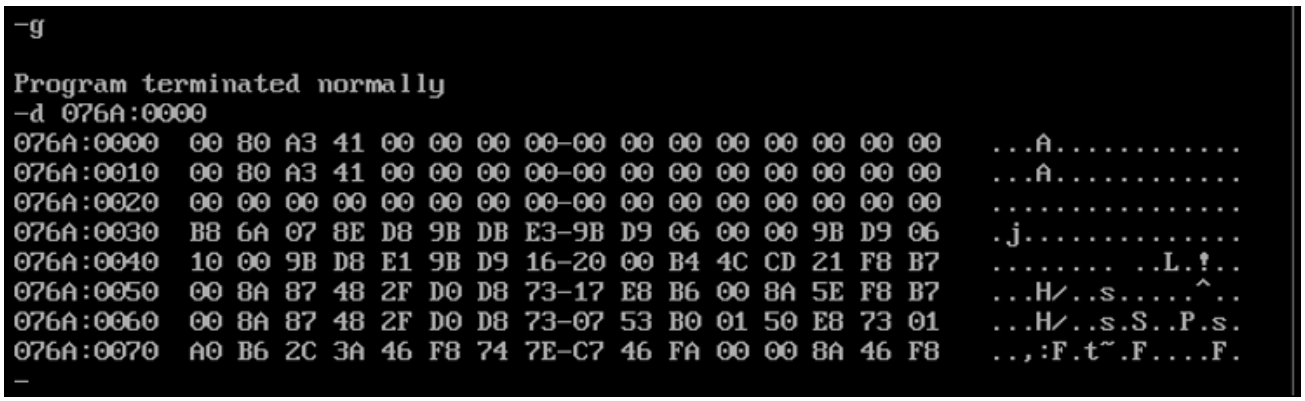
```
-u
076D:0000 B86A07        MOV     AX,076A
076D:0003 8ED8          MOV     DS,AX
076D:0005 9B            WAIT
076D:0006 DBE3                  FINIT
076D:0008 9B            WAIT
076D:0009 D9061000              FLD     DWORD PTR [0010]
076D:000D 9B            WAIT
076D:000E D9060000              FLD     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:

INPUT:

```
-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 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.
```

OUTPUT:



Result:

The assembly level programs were written to perform the above specified floating-point arithmetic operations and their output was verified.