|  |  |  |
| --- | --- | --- |
|  |  |  |

**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:

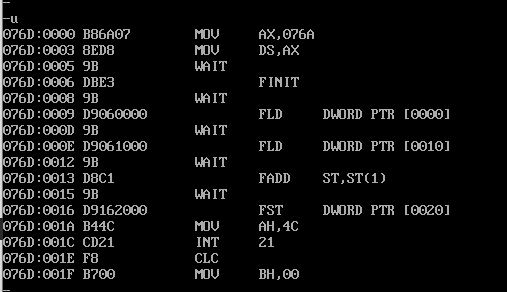
### (i) FLOATING POINT ADDITION

## 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.

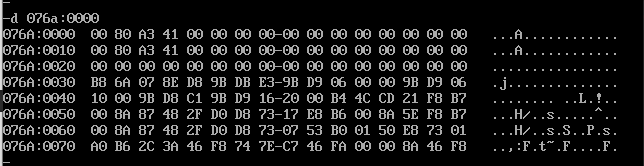
|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| **assume** cs:code, ds:data | Declare code and data segment. |
|  |  |
| **data segment** | Initialize data segment with values. |
| org 00h | Directive to assign an offset address for a variable. |
| x dd 20.4375 | Stores the first number. |
| org 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:

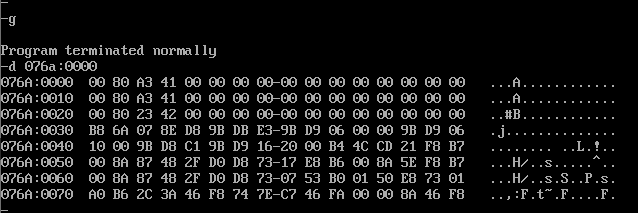


## Snapshot of sample input and output:

**INPUT:**



**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.

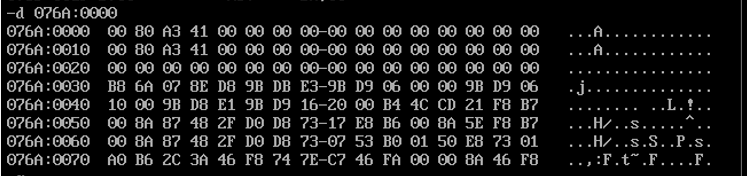
|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| **assume** cs:code, ds:data | Declare code and data segment. |
|  |  |
| **data segment** | Initialize data segment with values. |
| org 00h | Directive to assign an offset address for a variable. |
| x dd 20.4375 | Stores the first number. |
| org 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:

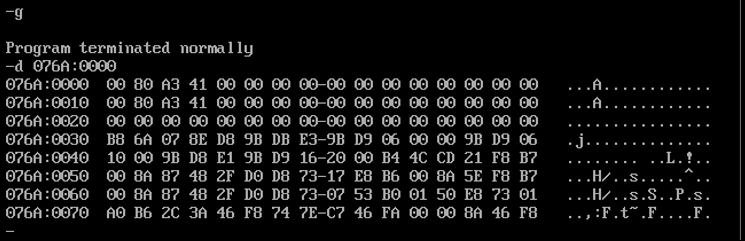


## Snapshot of sample input and output:

**INPUT:**



**OUTPUT:**



## Result:

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