|  |  |  |
| --- | --- | --- |
|  | FLOATING POINT OPERATIONS |  |
| Exp No.: 9 |  | **Name:** S Vishakan |
| Date: 14-10-2020 |  | **Reg. No:** 18 5001 196 |

**AIM:**

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

1. Floating point Addition.
2. Floating point Subtraction.

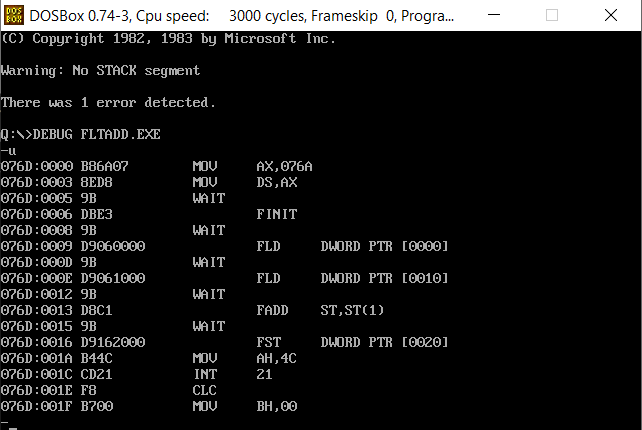
**PROGRAM – 1: FLOATING POINT ADDITION:**

**ALGORITHM:**

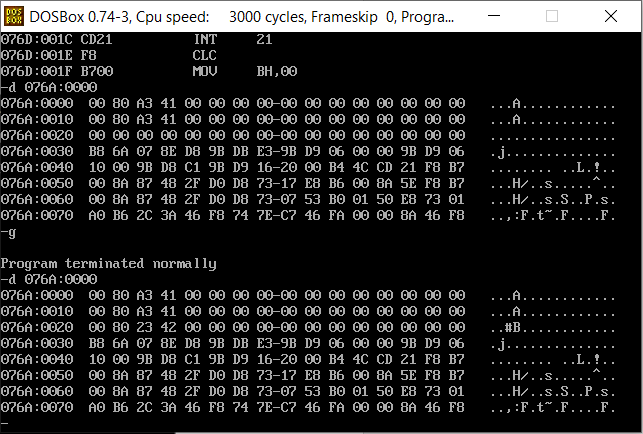
1. Begin.
2. Declare the data segment.
3. Initialize data segment with the 2 floating point numbers and a variable for storing their sum.
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Initialize Floating point operation using FINIT.
10. Move the contents of the two numbers into the stack ST.
11. Add them and store the value in top of the stack.
12. Move the content in top of the stack to variable ‘sum’.
13. Introduce an interrupt for safe exit. (INT 21h)
14. Close the code segment.
15. End.

|  |  |
| --- | --- |
| **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 |  |
| int 21h | Interrupt the process with return code and exit. |
| code ends |  |
| end start |  |

**UNASSEMBLED CODE:**



**SAMPLE I/O SNAPSHOT:**



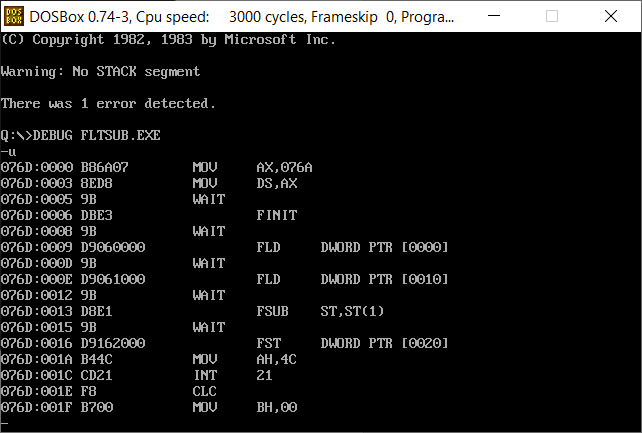
**PROGRAM – 2: FLOATING POINT SUBTRACTION:**

**ALGORITHM:**

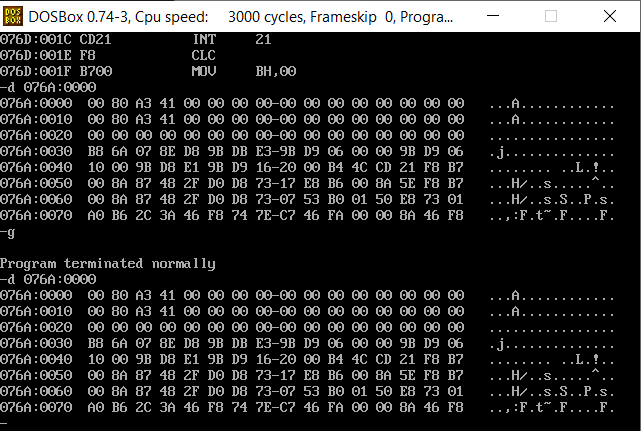
1. Begin.
2. Declare the data segment.
3. Initialize data segment with the 2 floating point numbers and variables for storing their difference diff.
4. Close the data segment.
5. Declare the code segment.
6. Set a preferred offset (preferably 100h)
7. Load the data segment content into AX register.
8. Transfer the contents of AX register to DS register.
9. Initialize Floating point operation using FINIT.
10. Move the contents of the two numbers into the stack ST.
11. Subtract them and store the value in top of the stack.
12. Move the content in top of the stack to variable ‘diff’.
13. Introduce an interrupt for safe exit. (INT 21h)
14. Close the code segment.
15. End.

|  |  |
| --- | --- |
| **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 |  |
| int 21h | Interrupt the process with return code and exit. |
| code ends |  |
| end start |  |

**UNASSEMBLED CODE:**



**SAMPLE I/O SNAPSHOT:**



**RESULT:**

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