NEW YORK CITY COLLEGE OF TECHNOLOGY

THE CITY UNIVERSITY OF NEW YORK

Department of Computer Engineering Technology

300 Jay Street, Brooklyn, NY 11201-1909

LAB REPORT

CET 3510 – OL71

(MICROCOMPUTER SYSTEMS TECHNOLOGY LABORATORY)

LAB #9

Floating Point Arithmetic Operations

Name: Puja Roy

Date: 12/5/21

Due Date: 12/12/21

Table of Contents

Objective:

The objective of this lab is to write a C/C++ program to examine three floating-point formats, single precision, double precision, and double extended precision which can be used to handle a wide range of performance and accuracy requirements. Most importantly, the purpose of this lab is to perform floating-point instructions for data transfer, data store, and floating-point arithmetic.

Materials:

- Microsoft Visual Studio C++ Community Edition 2019

Procedure:

1. First, open Microsoft Visual Studio C++ Community Edition 2019
2. Then, type program#1, compile and run the program.
3. Then modify the code.
4. Lastly, analyze the output.

Code:

```
3   #include<stdlib.h>
4   #include <iostream>
5   #include <time.h>
6   using namespace std;
7
8   //addition for single precision floating-point numbers
9   float faddition(float x, float y);
10
11  //subtraction for single precision floating-point numbers
12  float fsubtraction(float x, float y);
13
14  //addition for double precision floating-point numbers
15  double daddition(double x, double y);
16
17  //Subtraction for double precision floating-point numbers
18  double dsubtraction(double x, double y);
19
20  int main()
21  {
22      //Declare variables here
23      char ch, ch1, ch2, ch3;
24      //Single  precision floating-Point variables
25      float f1, f2, fsum, fsub;
26      //double precision floating-Point variables
```

```
53              std::cin >> ch;
54              ch3 = ch;
55              switch (ch3)
56              {
57              case 'a':
58              {
59                  cout << "Input two floating point operands in decimal  format\n";
60                  cin >> f1;
61                  cin >> f2;
62                  cout << "The first floating point value is " << f1 << endl;
63                  cout << "The second floating point value is " << f2 << endl;
64                  fsum = faddition(f1, f2);
65                  cout << "The sum of" << f1 << " and " << f2 << " is " << fsum << endl;
66                  printf("============================================================\n");
67                  break;
68              }
69              case 'b':
70              {
71                  printf("Input two floating point operands in decimal format\n");
72                  cin >> f1;
73                  cin >> f2;
74                  cout << "The first floating point value is " << f1 << endl;
75                  cout << "The second floating point value is " << f2 << endl;
76                  fsub = fsubtraction(f1, f2);
77                  cout << "The difference of" << f1 << " minus " << f2 << " is " << fsub << endl;
78                  printf("============================================================\n");
79                  break;
80              }
81              default: goto QuitLable;
82              }
83              goto Submenu1;
84          }
85          else if (ch2 == '2')
86          {
87          Submenu2:
88              cout << "Submenu - input your choice\n";
89              cout << "a, Inout two floating point oprands for addition, and display "
90                  << " the sum in the format of a decimal number.\n";
91              cout << "b, Input two floating point operands for subtraction, and display "
92                  << " the difference in the format of a decimal number.\n";
93              cout << "q,Quit\n";
94
95              std::cin >> ch;
96              ch3 = ch;
97              switch (ch3)
98              {
99              case 'a':
100             {
101                 cout << "Input two floating point operands in decimal format\n";
102                 cin >> d1;
103                 cin >> d2;
104                 cout << "The first floating point operant is " << d1 << endl;
```

```
105                     cout << "The second floating point operant is " << d2 << endl;
106                     dsum = daddition(d1, d2);
107                     cout << "The sum of" << d1 << " and " << d2 << " is " << dsum << endl;
108                     printf("============================================================\n");
109                     break;
110                 }
111                 case 'b':
112                 {
113                     printf("Input two floating point operands in decimal format\n");
114                     cin >> d1;
115                     cin >> d2;
116                     cout << "The first floating point operant is " << d1 << endl;
117                     cout << "The second floating point operant is " << d2 << endl;
118                     dsub = dsubtraction(d1, d2);
119                     cout << "The difference of" << d1 << " minus " << d2 << " is " << dsub << endl;
120                     printf("============================================================\n");
121                     break;
122                 }
123 
124                 default: goto QuitLable;
125                 }
126                 goto Submenu2;
127             }
128             else
129             {
130                 goto EndLable;
131             }
132         QuitLable:
133             cout << "Do you like to continue the floating point arithmetic operations (Y/N)?"
134                 << "Enter Y(y) or N(n)" << endl;
135             cin >> ch;
136             ch1 = ch;
137         }
138 
139     EndLable:
140         cout << "Exit program" << endl;
141 
142         system("pause");
143         exit(0);
144 
145         return 0;
146     }
147 
148     // addition for single precision floating-point numbers
149     float faddition(float x, float y)
150     {
151         float f;
152         _asm
153         {
154             // push a single precision floating point number x
155             //onto  the top of the stack ST(0)
156             FLD x;
```

```
157             /* push a single preciosion floating point number y
158             onto the top of the stack ST(0), move x down to the stack ST(1)*/
159             FLD y;
160             //add ST(0) with ST(1), store the sum into ST(0)
161             FADD;
162             //copy ST(0) to memory variable f
163             FST f
164         }
165         return f;
166     }
167
168     //subtraction for single precision floating-point numbers
169     float fsubtraction(float x, float y)
170     {
171         float f;
172         _asm
173         {
174             // push a single precision floating point number x
175             //onto  the top of the stack ST(0)
176             FLD x;
177             /* push a single preciosion floating point number y
178             onto the top of the stack ST(0), move x down to the stack ST(1)*/
179             FLD y;
180             //store the difference of ST(1)-ST(0) = x-y into ST(1)
181             FSUB ST(1), ST(0);
182             //exchange the top of the stack with register ST(1)
```

```
183             FXCH ST(1);
184             //copy ST(0) to memory variable f
185             FST f;
186         }
187         return f;
188     }
189
190     //addition for double precision float point numbers
191     double daddition(double x, double y)
192     {
193         double d;
194         _asm
195         {
196             // push a single precision floaitng point number x
197             //onto the top of the stack ST(0)
198             FLD x;
199             // push a single precision floating point number y
200             //onto the top of the stack ST(0), move x down to the stack ST(1)
201             FLD y;
202             //FADD ST(0), ST(1), store the sum into ST(0)
203             FADD;
204             // copy ST(0) to memory varaible d
205             FST d;
206         }
207         return d;
208     }
```

```
210     //subtraction for double precision float point numbers
211     double dsubtraction(double x, double y)
212     {
213         double d;
214         _asm
215         {
216             // push a single precision floaitng point number x
217             //onto the top of the stack ST(0)
218             FLD x;
219             // push a single precision floating point number y
220             //onto the top of the stack ST(0), move x down to the stack ST(1)
221             FLD y;
222             //FADD ST(0), ST(1), store the sum into ST(0)
223             FSUB ST(1), ST(0);
224             // copy ST(0) to memory varaible d
225             FXCH ST(1);
226             //copy ST(0) to memory variable d
227             FST d;
228         }
229         return d;
230     }
231
```

Output:

```
Start the floating point calculator Y/N, enter Y(y) or N(n)
y
Menu:
1, Single precision floating point arithmetic operation (32-bit)
2, Double precision floating point arithmetic operation (64-bit)
3, Exit
Menu Options:
 1
Submenu - input your chopice
a, Input two floating point operands for addition, and display the sum in the format of a decimal number.
b, Input two floating point operands for subtraction, and display the difference in the format of a decimal number.
q,Quit
a
Input two floating point operands in decimal  format
2.1 3.2
The first floating point value is 2.1
The second floating point value is 3.2
The sum of2.1 and 3.2 is 5.3
============================================================
Submenu - input your chopice
a, Input two floating point operands for addition, and display the sum in the format of a decimal number.
b, Input two floating point operands for subtraction, and display the difference in the format of a decimal number.
q,Quit
```

Conclusion:

Throughout this experiment, there were three floating-point formulas including single precision, double precision, and double extended precision which were used to handle a wide range of

performance and accuracy requirements. From this lab, I examined that these 3 floating-point formulas have binary storage formats. To use floating points, we must use FSTP, FMUL, FDIV, FDIVR, FSQRT etc. Floating point numbers are different than integer numbers which is why we're using these floating point formats.