**Simber Homework Ch12**

**Short Answer**

3. 11011.01011 = 1.101101011 x

4. 0000100111101.1 = 1.001111011 x

5. two types of NaNs

1) quiet NaN – propagate through most arithmetic operations without causing an exception

2) signaling NaN – generate a floating-point invalid operation exception

7. **FST** – copies a floating-point from the top of the FPU

stack into memory

**FSTP** – copies the value in ST(0) to memory and pop

ST(0) off the stack

\*FST does not pop the stack.

8. **FCHS**(change sign) instruction reverses the sign of the floating-point value in ST(0).

11. **FCOM**(compare floating-point value) instruction compares

ST(0) to its source operand.

\* F6 family processors introduced the FCOMI instruction.

13. **RC field** specifies which rounding method to use

\* field values

00 binary: round to nearest even(default)

01 binary: round down toward negative infinity

10 binary: round up toward positive infinity

11 binary: round toward zero(truncate)

**Algorithm Workbench**

1. **0 10000010 11001100000000000000000**

+1110.011 normalized 1.110011 x

sign bit: 0 ; positive

exponent: 10000010 ; 3 + 127 = 130 = 126 + 4 = 10000010b

fraction: 11001100000000000000000

2. **0.101** binary

binary long division method

5 = 0101 8 = 1000

.101

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1000 | 0101.0

-100 0

\_\_\_\_\_\_\_\_\_

1 000

1 000

\_\_\_\_\_\_\_\_

0

9.

double B = 7.8;

double M = 3.6;

double N = 7.1;

double P = - M \* (N + B);

influx: - M \* (N + B)

postfix: M – N B + \*

.data

B REAL8 7.8

M REAL8 3.6

N REAL8 7.1

P REAL8 ?

.code

fld M ; ST(0) = M

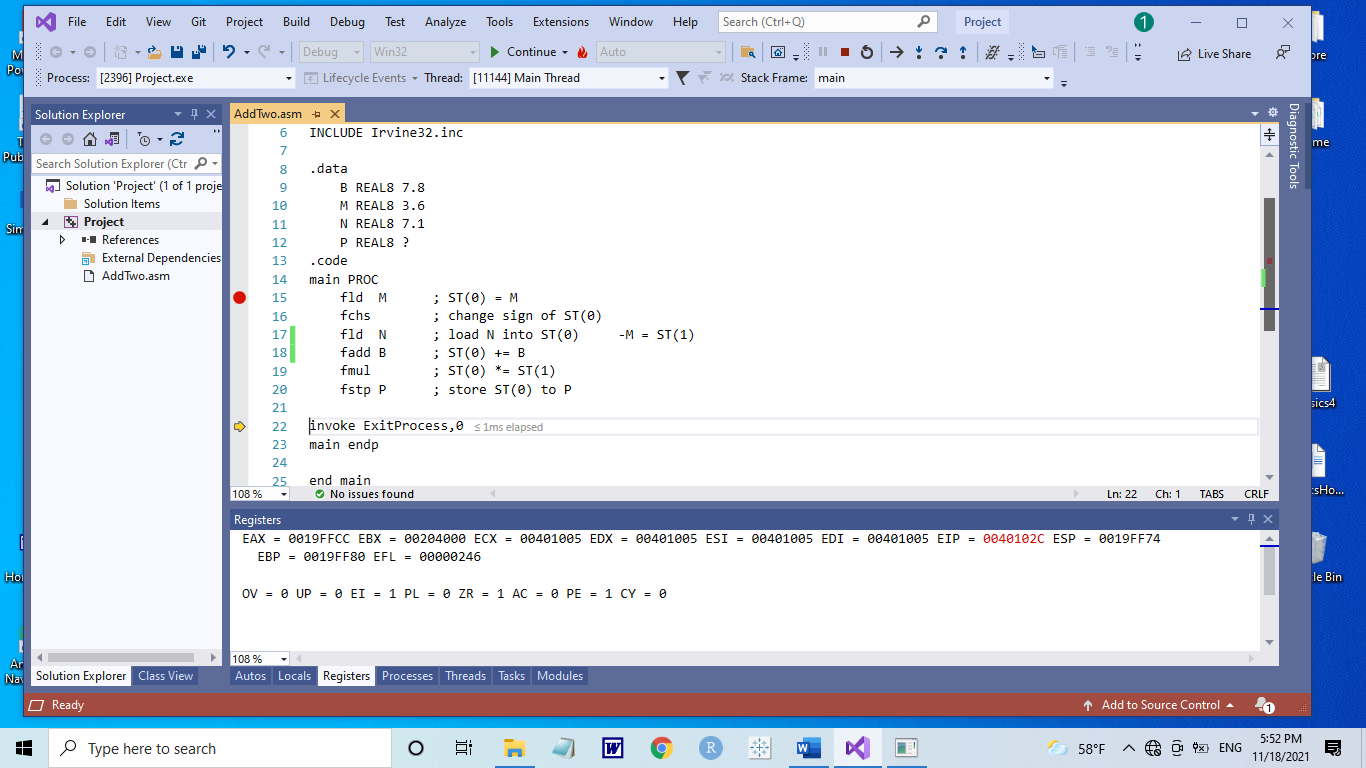
fchs ; change sign of ST(0)

fld N ; load N into ST(0) -M = ST(1)

fadd B ; ST(0) += B

fmul ; ST(0) \*= ST(1)

fstp P ; store ST(0) to P



\* Just curious?

How can see the result?