Michael McNichol
247345 (2021SP Microprocessor Assembly Lang (CIS-2420-NET01))

1.) The portability of source code written in C++ is written specifically towards the underlying facilities of the operating system. The toolchain (C++ Compiler, assembler, etc) is built to convert all instructions into machine code understood by that computers' instruction set often times attempting to make optimizations where possible. This could be an instruction set for ARM, x86, or various chips where the type of toolchain used would be included or guided by the Operating Systems needs.

    The portability of the x86 assembly language will only work with an x86 processor. These are the specific instructions that the CPU is capable of executing and are not one for one with other CPU instruction sets.

2.) I'm not sure if this is asking for all of the instructions necessary to move the strings into the registers that are used to display characters and point the stream to the terminal output. I'll give it a shot but assuming it's just the math operations I'll leave that as a second answer.

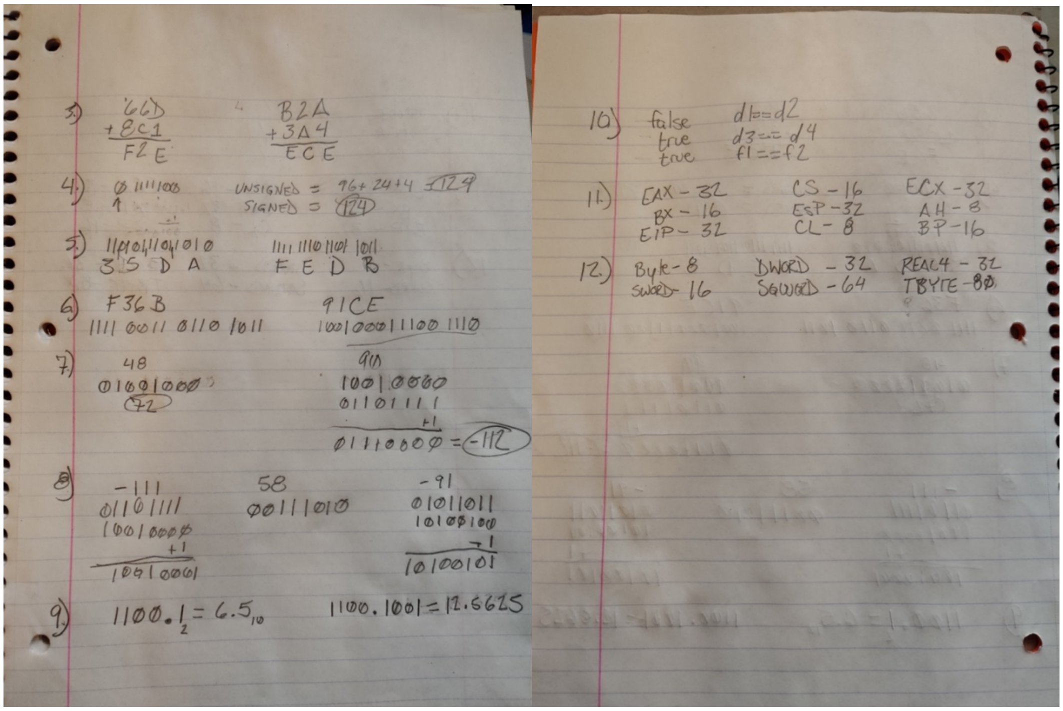
    Assembly Instructions: Moving X into the eax register, divide by a floating point, mov result into a register for printing, interrupt to screen, mov an @data section pointer of the "is the decimal percent value" to the register for printing to screen, interrupt to screen, mov x to the register for printing to screen, interrupt to screen.

    8 Assembly Instructions

    For machine instructions I feel like you are asking me how many 1's and 0's. Double and floating point both being DWORDs, string being 33 characters of (assuming) ASCII 8x1 Byte. (33 * 8 * 4) + (3 * 2 * 8) = 312

3 – 12: **Attached Photo of Paper**

Note:* ***On question 9* *for the Floating Point I was a little confused because you didn't say the mantissa was 3 bits etc. like you had done in the Assigned work previously. You used the term "Pure Binary" which I'm guessing you just meant place the decimal where it would go and show the bits as the ½, ¼, etc. I get wrapped around the axle fairly easy so may have misread this.*



3.) 66D + 8C1 = F2E   B2A + 3A4 = ECE

4.) Ø1111100   UNSIGNED = 96+24+4 = 124
SIGNED = 124

5.) 11Ø1Ø111Ø11Ø1Ø   1111 111Ø 11Ø1 1Ø11
3 5 D A   F E D B

6.) F36B   91CE
1111 ØØ11 Ø11Ø 1Ø11   1ØØ1 ØØØ1 11ØØ 111Ø

7.) 48   9Ø
Ø1ØØ1ØØØ   1ØØ1 ØØØØ
72   Ø11Ø1111
+1
Ø111ØØØØ = -112

8.) -111   58   -91
Ø11Ø1111   ØØ111Ø1Ø   Ø1Ø11Ø11
1ØØ1ØØØØ   1Ø1ØØ1ØØ
+1   +1
1ØØ1ØØØ1   1Ø1ØØ1Ø1

9.) $1100.1_2 = 6.5_{10}$ 1100.1001 = 12.5625

10.) false d1==d2
true d3==d4
true f1==f2

11.) EAX - 32   CS - 16   ECX - 32
BX - 16   ESP - 32   AH - 8
EIP - 32   CL - 8   BP - 16

12.) Byte - 8   DWORD - 32   REAL4 - 32
SWORD - 16   SQWORD - 64   TBYTE - 80

13.) String Haiku

```
firstFiveSyllable     BYTE   "DANGEROUS ALONE", 0
secondSevenSyllable   BYTE   "HERE TAKE THIS TO GO WITH YOU", 0
thirdFiveSyllable     BYTE   "YOU ARE NOT ALONE", 0
```