Name: _____

Directions: Work only on this sheet (on both sides, if needed); do not turn in any supplementary sheets of paper. There is actually plenty of room for your answers, as long as you organize yourself BEFORE starting writing. In order to get full credit, SHOW YOUR WORK.

1. (10) Find the 8-bit 2s complement representation of -18. (Give your answer in hex.)

2. (10) Consider the assembler output on p.3 of our PLN unit on machine language. Suppose the program were to have additional instructions beyond the label **done**, and suppose at that label there were a JNZ instruction whose target (i.e. jump destination) were in offset 0x0025 of the **.text** segment. What would the machine code be for this instruction?

3. (10) Write at most two lines of assembly code which will copy the memory word pointed to by EAX to the memory word pointed to by EBX. You may assume all other registers are available to you.

4. (10) State the contents of the Mantissa Field for the decimal number 1.75, under the IEEE standard.

5. (10) How would a compiler likely translate

```
x[12] += 15;
```

Here **x** is a global **int** array. Give your answer in assembly language. Any code that works will get full credit, but no credit will be given to any code containing more than seven instructions, with five being "normal" and two being possible.

6. Suppose the code on p.30 of our PLN unit on Linux assembly language had been

```
.data
x: .string "abcde"  # 5 characters plus a null
   .space 3

.text
.globl _start
_start:
    movl $x, %esi
    movl %esi, %edi
    addl $3, %edi
    movl $6, %ecx
    rep movsb
done:
    movl $0,%eax
```

Suppose that we execute this in GDB, and pause at **done**.

(a) (10) What GDB command could we use to examine the 9 bytes beginning at **x**?

(b) (10) What would the contents of those 9 bytes be? (In your answer, state the content of each individual byte, either in character, decimal or hex form.)

7. The POP instruction on line 25 of p.5 of the PLN unit on subroutines is there to clean up the stack. But it only removes one element from the stack, when it should be two.

(a) (5) Replace the POP by an ADD instruction which will properly clean up the stack.

(b) (5) Suppose instead of the ADD instruction we simply were to add a second POP. For which instructions, if any, would this cause a change in machine code? (Note that I am asking whether machine code itself would change, not whether offsets would change.) List the new machine code for all such instructions.

8. (20) Look at the assembler language output for the full example in Sec. 7 of the PLN unit on subroutines. Suppose the **.data** and **.text** segments begin at 2008 and 80888, respectively. (All addresses in this problem will be in decimal unless otherwise stated. Note that it may not be necessary to use all given information in this problem.) Then the address of the 0x48 for the DECL instruction is _____. Also, suppose that just before the CALL is executed, c(ESP) = 36000. Then right after the CALL instruction is executed, the pushed value 3 for **n** will be in the word with address _____ and the contents of byte 35998 will be _____.

Solutions:

1. 0xee

2. 75 0c

3.

```
movl (%eax), %ecx
movl %ecx, (%ebx)
```

4. $1.75 = 1 + 2^{-1} + 2^{-2}$, so the base-2 represention is 1.11, very similar to the example of $1.625 = 1.101_2$ in the PLN. So, the Mantissa Field is $11\underbrace{0...0}_{21\ 0s}$.

5.

```
movl $48, %ebx
addl $15, x(%ebx)
```

or

```
addl $15, x+48
```

or ...

6.a `x/9c &x`

6.b As mentioned in the PLN, MOVS does the same thing a loop would do, only faster. In this case, in which the destination and source arrays overlap each other, this implies that some characters will be overwritten before it reaches their turns to be copied. The final contents

starting at **x+3** will be 'a','b','c','a','b','c','a','b','c', NOT 'a','b','c','d','e',NULL.

7.a `addl $8, %esp`

7.b The extra POP adds one byte of code. So, the machine code for the CALL instruction changes, since the distance to **init** from (the instruction after) the CALL will increase by one byte. The new code will be e8 10 00 00 00.

8.a The offset of 0x48 is 0x0016 = 22, so the address is 80888+22 = 80910.

8.b The value 3 is in word 36000.

8.c The CALL will push the return address, so that value will be in 36000-4 = 35996. Byte 35998 will be the third-least significant byte in that word. The contents of the whole word will be 80888+0x10 = 0x00013c08, and the third-least significant byte will be 01.