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) Fill in the blank: The name of the UNIX program which in class we've been referring to as the "clerk" is _____.

2. Look at the example assembly language program on p.4 of the unit on Linux assembly language.

(a) (15) State the contents of the registers EAX, EBX and ECX after the third execution of the **decl** instruction. Assume that x starts at location 0x500.

(b) (10) Suppose we had accidentally written **incl** instead of **decl**. Would the program still work correctly? If so, clearly state why; if not, then very specifically state what would happen.

3. (10) In class we discussed an alternative hardware design for Intel in which conditional jumps would have two operands instead of one. For example, the instruction

```
jnz top
```

in our example program would become something like

```
jnz %eax,top
```

Fill in the blank: The advantage we discussed was that this design would eliminate _____.

4. (10) In class it was stated that Intel CPUs do not allow two operands of the same instruction to be in memory, but an exception was cited. State which specific counterexample was given.

5. (10) In our floating-point system at the bottom of p.8 and top of p.9 of Chapter 1, what is the largest possible number which can be represented, expressed in $u \times 2^v$ form? What is the smallest possible positive number?

6. (10) For some positive number x, say the last few bits in the 2s comp. rep. of -x are ...101101. (We are NOT given the more-significant bits here.) Find the corresponding bits for -(x+3).

7. (10) Suppose we are writing a **vi**-like text editor which is capable of processing documents which consist of a mixture of English and Chinese, as in our discussion on this in lecture.

Say our editor reads the file specified by the user into a global array of **char** named TheFile. We have a global **int** variable named Cursor which points to the current position of the cursor within the file. In other words, the character currently highlighted by the editor cursor begins at TheFile[Cursor]. That character also ends at that same position in the array if it is an English character; if it is a Chinese character, the character consists of TheFile[Cursor] and TheFile[Cursor+1]. There is a null byte immediately following the last character in TheFile. As a check on your understanding, note that if there are, for example, 3 English and 2 Chinese characters before the character currently being highlighted by the cursor, then Cursor will equal 7.

Our editor will have a **void** function named DoLeftArrow(), which takes the proper actions when the user hits $\leftarrow$, i.e. the function will move the cursor one (English or Chinese) character to the left. Write this function. Your function will call another one, ScreenUpdate(), which physically moves the cursor image on the screen; assume that is given, i.e. you will not write that function yourself here.

8. (15) The following code assumes that the .data segment includes the following:

```
x:
   .word 0
y:
   .byte 0
```

We will be interpreting the quantities in x and y as 16-bit and 8-bit signed integers, respectively.

In the text segment, y is changed (code NOT shown below), and then we wish to copy y to x, accounting for the fact that we are using them to store quantities of different sizes. To do so, we have the following code:

```
   ... # y is changed in the "..."
   movl $x,%eax
   movb y,%bl
   movb %bl,(%eax)
   addl _______,%eax
   cmpb $0,y
   js a
   movb _______,(%eax)
   jmp b
a:
   movb _______,(%eax)
b:
```

Fill in the three blanks in the code above.

Solutions:

1. as

2.a. 1; 8; 0x50c

2.b. Eventually ECX would exceed the range of memory allocated to the program, and a seg fault would occur.

3. The EFLAGS register (thus saving space on the CPU chip).

4. decl (%eax)

5. 511×2^{31}; 1×2^{-32}

6. To get -(x+3) from -x, just "wind back" three more times, yielding ...101010.

7.

```
void DoLeftArrow()

{  if (Cursor > 1)  // Cursor = 1 is an exceptional case
      // check for Chinese, i.e. high bit set
      if (TheFile[Cursor-2] < 0) Cursor -= 2;  // yes, Chinese
      else Cursor--;  // no, English
   else Cursor--;  // if Cursor = 1, no room for a Chinese character
   ScreenUpdate();
}
```

8.

```
   ... # y is changed in the "..."

   #copy y to low byte of x
   movl $x,%eax
   movb y,%bl
   movb %bl,(%eax)

   # point EAX to high byte of x, to prepare for later movb
   addl $1,%eax

   # if y >= 0, need to put 0s in the high byte of x; if y < 0, need to
   # put 1s there
   cmpb $0,y
   js a
   movb 0x00,(%eax)
   jmp b
a:
```

```
   movb 0xff,(%eax)
b:
```