Name: ______________________________

Directions: **Work only on this sheet** (on both sides, if needed). MAKE SURE TO COPY YOUR ANSWERS TO A SEPARATE SHEET FOR SENDING ME AN ELECTRONIC COPY LATER.

1. (20) Fill in the blanks: The analog of a compiler at the assembly language level is called a ________________. In our class, we refer to it metaphorically as a human ___________.

2. As you know, when writing functions/subroutines, we may originally write one for use in one program but then find we can use it—unchanged—in another program. Suppose that is the case for the subroutine **findmin** in Section 3.6, pp.72ff; in other words, we simply copy lines 54-95 to the source file of another program, and use the subroutine there. Say the array we'll be using it on begins at a label **z** and has 12 elements. The relevant section of code will be

```
movl  ------------------------ # blank (a)
------------------------ # blank (b)
call  findmin
movl  $168,w
```

(a) (15) State what should go in blank (a).

(b) (15) State what should go in blank (b).

(c) (15) Suppose the address of the memory location labeled **z** turns out to be 0x100c. What value will be in EDX when the subroutine returns (i.e. just before we execute the instruction involving 168)?

3. (15) Consider an instruction

```
addl (%ecx), (%edx)
```

Assume that the two registers point to memory locations that the program has permission to read from and write to. Which one of the following is true? (Note that this is NOT a multipart problem, and will occupy only one line in your electronic file.)

(i) There would be no assembler error message, and execution would work as intended.

(ii) There would be no assembler error, but execution may produce incorrect results.

(iii) There would be no assembler error, but execution may produce a seg fault or other execution error.

(iv) There would be an assembler error, even though the assembler could have translated the line to machine code that would work as intended.

(v) There would be an assembler error, because no such machine instruction exists.

4. (20) Consider the program in Section 3.6, pp.72ff. Suppose we assemble and link, creating an executable file **badsort**. We then try to execute it "normally" from the shell command line:

```
% badsort
```

Give the line number of the instruction in this source code that will likely be the last one to execute before the program encounters an execution error, e.g. a seg fault.

Solutions:

1. assembler, clerk

2.a movl $z, %eax

2.b movl $11, %ebx

3. (v)

4. 52