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. (15) Look at the output of **gcc -S** at the top of p.10 of our PLN unit on subroutines. Suppose the author of the last two **movl** instructions had chosen to use ESP instead of EBP. Show what those two instructions would be now.

2. Look at the second GDB session printout covered by May in the discussion section. (It consists of an assembly language program which sums elements in an array, adapted from the example in Sec. 7 of our PLN unit on subroutines; output from **as -a**; and a GDB session.)

(a) (15) When the **ret** is executed, what address will we return to?

(b) (10) When the **ret** is executed, the circuitry for that instruction will look in a certain memory location to determine where to return to (i.e. the value asked for in part (x) above). What will be the address of that location?

(c) (15) In the GDB session, suppose we had issued the **c** ("continue") command, instead of **q** ("quit"). Then what would be the address of the first instruction fetch to occur outside our **.text** segment?

3. (10) Recall that **strace** is used like this:

```
% strace a.out
```

Based on knowledge from our course, give an example of a line of C code which is likely to be in the source code for **strace**. It must be a system call, and for full credit it must be complete.

4. (35) The assembly language code below searches a string for the first occurrence of 'a' (guaranteed to exist), and returns the position of that occurrence (0 for the first character in the string, 1 for the second, 2 for the third, etc.). The function **firsta()** is to be callable from any C program, with this prototype:

```
int firsta(char *s);
```

Fill in the blanks:

```
.text
.globl firsta
firsta:
   movl ________, %eax
top:
   cmpb $'a', (%eax)
   jz foundit
   --------
   jmp top
foundit:
   --------
   ret
```

Solutions:

1.

```
movl $0, (%esp)
movl $4, (%esp)
```

2.a This is what is on the top of the stack at that point, 0x8048084.

2.b This is the address of the top of the stack at that point, 0bfffcec4.

2.c We would continue, executing the two MOV instructions and then the RET in line 37. That would pop the stack for the "return address," when in fact there is no return address on the stack. What would be on the stack instead? Unfortunately, we erred in line 27, where we should have added 8 to the stack pointer, so the value $x+4 is still there. So, that is where we will "return" to–the second word in the **.data** segment, which is at 0x80490a8.

3.

```
execve(argv[1]);
```

4.

```
4(%esp)
incl %eax
subl 4(%esp), %eax
```