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.

1. Consider the C and assembly code in Section 6.8.1, pp.134-136.

(a) (20) Based on the information here, how many arguments does **exit()** have? (i) None. (ii) One. (iii) Two. (iv) Three. (v) It would have the number of arguments given in c(EAX).

(b) (25) Suppose that in running this code under GDB, I set a breakpoint at line 35, p.136, and run the program. When it stopped at the breakpoint, I submitted these commands, with the results shown:

```
35          pop %ebx
(gdb) info registers
eax            0xbfaaaad4       -1079334188
ecx            0xbfaaaa50       -1079334320
edx            0x1      1
ebx            0x804a020        134520864
esp            0xbfaaaa18       0xbfaaaa18
ebp            0xbfaaaa38       0xbfaaaa38
esi            0x8048460        134513760
edi            0x8048340        134513472
eip            0x8048443        0x8048443 <addone+7>
eflags         0x200202 [ IF ID ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x33     51
(gdb) x/5x 0xbfaaaa18
0xbfaaaa18:     0xb8074ff4      0x0804841b      0x0804a020      0x08049ff4
0xbfaaaa28:     0xbfaaaa48
```

State the addresses of **x** and the instruction on line 26, p.135.

(c) (10) It would have been better to use ECX instead of EBX on p.136. Very briefly explain what advantage would accrue from using ECX here, citing a specific passage in the textbook.

2. (25) Suppose we wish to store data on a stack that grows *away* from 0. Thus we cannot use the **pushl** instruction, and of course subroutines calls will use the "normal" stack, not this one; we will just use this one to store data. Give two lines of assembly code that will push the number 88 onto this new stack. Assume that EDX will serve as our stack pointer.

3. (20) Suppose we're writing an assembly language program whose **.data** segment is

```
.data
x:     .long 0
fmt:   .string "%d"
```

We wish to read in the value of **x** from the keyboard, by calling the C library function **scanf()**. Give assembly code (no more than five lines) that will do this.

Solutions:

1a. (ii)

1b. **x** is at 0x804a020, and the instruction is at 0x0804841b.

1c. Page 141, bottom says we are guaranteed there will be no "live" values in ECX. *Thus we need not save it on the stack, and later pop it off, thus saving time.*

2.

```
addl $4,%edx
movl $88,(%edx)
```

3.

```
pushl $x
pushl $fmt
call scanf
```