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) For each of the following instructions, state the number of bus transactions which would occur, including the instruction fetch. (Ignore cache and prefetch effects.)

(a) **movl %ebx,(%ecx)**

(b) **movl %ebx,(%esp)**

(c) **movl %ebx,12(%ecx)**

(d) **movl %ebx,x**

(e) **movl %ebx,x(%ecx)**

(f) **jnz top**

2. Look at the output of **as -a** on p.15 of our unit on subroutines.

In Parts (a) and (b) of this question, suppose that the **.data** and **.text** segments begin at 0x4000 and 0x6000, respectively. Suppose also that just before Step A for Line 24, c(ESP) = 0x1000. Use hex notation in your answers.

(a) (10) List the values, if any, which we can be sure that will be in the PC at some point during the processing of the instruction in Line 24. ("Processing" includes Steps A-C, but you should NOT break your list down according to step.)

(b) (10) Just prior to Step A of Line 35, state what c(ESP) and c(ECX) will be.

In Part (c), do NOT make the assumptions used in Parts (a) and (b).

(c) (15) Suppose the **call** in Line 24 and the **ret** in Line 35 were both to be replaced by **jmp** instructions (with Line 26 having a label, say **top2**). Show the machine code which would be produced by the assembler for these two instructions.

3. (15) As discussed in our unit on machine language and in class, in some cases the assembler can only "partially" assemble an instruction, and the linker must later adjust the code produced by the assembler. State which <u>instructions</u> in this **as -a** output on pp.3-4 of the unit on machine language fall into this category. Your answer will consist of a list of line numbers.

4. (5) In the function **sum()** on p.10 of the unit on subroutines, suppose the local variables had been:

```
int i=12,s=5,z[10];
```

Show the assembly code that **gcc** might produce from this code fragment, in place of what we see at the top of p.11. Make sure that you only modify the code on p.11, not adding any new lines.

5. (10) In Homework III.A, we were finally able to take a program consisting at least partly of assembly language and run it by itself, i.e. not within GDB, without causing a seg fault. Identify the line on pp.7-8 of the unit on subroutines which made this possible.

6. (15) Say we have C code

```
int g(int x)
{  int q;
...
}
...
int y[20],z;  // global variables
...
z = g(y[4]);
```

Show the assembly code that **gcc** might produce from this code fragment, up to and including a **call** instruction. (More than one answer is possible.)

Solutions:

1. 2, 2, 2, 2, 1

2.a 0x6000 (just prior to instruction fetch), 0x06005 (after instruction decode), 0x6013 (after Step C)

2.b 0x1000 - 4 = 0xffc

2.c The first JMP will be only 2 bytes, rather than the 5 occupied by the CALL, to **top** will be at 0x0002 and **init** will be at 0x0010. Thus the distance operand of the first JMP will be 0x0010 - 0x0002 = 0x0e, and since the op code is 0xeb, the JMP instruction will be 0xeb0e.

Again because everything is moved forward 3 bytes due to having the CALL replaced by the shorter JMP, the second JMP, which replaces the RET, would be at 0x0022 - 3 = 0x001f. That JMP instruction occupies 2 bytes, i.e. 0x001f and 0x0020, so the jump distance is measured from 0x0021 to **top**, which is at 0x0002. Thus the distance is 0x0002 - 0x0021 = -0x1f = 0xe1 (in 8-bit arithmetic, since the distance operand is just 1 byte). So, the second JMP instruction will be 0xebe1.

3. 27, 33

4.

```
subl $48, %esp
movl $5, -8(%ebp)
movl $12, -4(%ebp)
```

5. call exit

6. For example:

```
movl $16, %eax
pushl y(%eax)
```