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. Earlier problems tend to be easier.

1. Fill in the blanks with either "increase," "decrease" or "not affect" (you can use the same answer multiple times):

- (a) (5) A PUSH instruction will _____ the value of ESP.
- (b) (5) A CALL instruction will _____ the value of ESP.
- (c) (5) A JMP instruction will _____ the value of ESP.
- (d) (5) A RET instruction will _____ the value of ESP.

2. (10) Fill in the blank with a UNIX system call: The system call made by **tcsh** when you run **as** is _____

3. (10) Fill in the blank with information obtained during our discussion in lecture: If we wished to know how the instruction labeled INL on p.108 of Neveln is expressed in AT&T syntax, we would use _____.

4. Assume that both TryAddOne.c and AddOne.s are exactly as on our Web pages, with no changes whatsoever. (Note: This is the revised version, with an **incl** instruction in AddOne.s.) The compiler will generate a **pushl**, a **call** and an **addl** instruction from this line of C code. Suppose these three instructions are located at 0x500, 0x505 and 0x50a, respectively. Suppose also that addone() begins at 0x600 and x is at 0x700, and prior to the execution of the **pushl**, EAX, EBX and ESP contain 0x168, 0x1088 and 0x102c, respectively.

- (a) (15) At the very end of the execution of the **call** instruction, what values will be in the PC, MAR and MDR?
- (b) (10) What value will be in ESP immediately prior to the fetch of the **addl** at 0x50a?
- (c) (15) Give a complete list, in chronological order, of the values which will go into the MDR during the 5 instructions of addone(). Exclude instruction fetches.

5. (20) Write AT&T-syntax assembly code which will be callable from C using the function prototype

```
int sumarray(int *, int, int *);
```

For convenience, let us refer here to the arguments as "p," "n" and "q."This function will return the sum of n consecutive words, the first of which is pointed to by p. It will also store in the word pointed to by q an indicator as to whether any carries resulted from the summing (1 means yes, 0 means no). Do not include a .data segment. Note: The C compiler will produce code which pushes the rightmost argument first (in this case, the place for the carry), then the others moving right to left. **Please write a draft version on scratch paper first, and then copy it neatly to this sheet here.**

Solutions:

1. decrease, decrease, not affect, increase

2. execve()

3. intel2gas

4.(a) PC: 0x600; MAR: 0x1024; MDR: 0x50a

4.(b) 0x1028

4.(c) 0x1088; 0x700; 7; 8; 0x1088; 0x50a

5.

```
.text

.globl sumarray

sumarray:

        # save old values of registers we'll use
        pushl %ebx
        pushl %ecx
        pushl %edx

        # get parameters from stack
        movl 16(%esp), %ebx   # p
        movl 20(%esp), %ecx   # n
        movl 24(%esp), %edx   # q

        # use EAX for sum, since must return in EAX anyway
        movl $0, %eax

        # set carry parameter to 0 for now,  then to 1 later maybe
        movl $0, (%edx)  # note parentheses

top:    addl (%ebx), %eax
        jnc nc
        # carry found, so set carry parameter to 1
        movl $1, (%edx)
nc:     addl $4, %ebx   # point EBX to next word to be summed
        decl %ecx       # decrement loop count
        jnz top

        # restore old register values
        popl %edx
        popl %ecx
        popl %ebx

        # EAX already set for return, so nothing left to do
        ret
```