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. (30) The following code searches through an array pointed to by ECX for a word whose contents are equal to those of EAX. Such a word is assumed to exist, and once it is found, its address will be in EBX. Fill in the blanks (you do not have to put an instruction at **done**):

```
      movl ________________________________________
top:  _____________________________________________
      je done
      addl ________________________________________
      _____________________________________________
done:
```

2. (30) The function **snc()** is intended as a replacement for the library function **strncpy()**. The latter has arguments **dst**, **src** and **n**, and copies **n** bytes at the string pointed to by **src** to the string pointed to by **dst**. However, **snc()** may do it faster, since it uses an instruction from the MOVS family. Fill in the blanks.

```
snc:
   pushl %ecx
   pushl %esi
   pushl %edi
_________________________________________________
_________________________________________________
_________________________________________________
_________________________________________________
   popl %edi
   popl %esi
   popl %ecx
_________________________________________________
```

3. Consider the ISR, top of p.168. Consider parts (a) and (b) below to be independent of each other.

(a) (15) State code, to be inserted after line 5, that will put 1 or 0 into EDX, depending on whether the user struck the A key or not. (Assume that whatever registers you use have previously been saved.)

(b) (15) State code, to be inserted after line 3, that will put 1 or 0 into EDX, depending on whether the Carry Flag had been set just before the interrupt occurred. If you use any registers, push their old values first.

x. (10) Suppose **m** is declared as

```
int m[8][20];
```

How many words apart in memory will be **m[5][5]** and **m[7][7]**?

Solutions:

1.

```
      movl %ecx, %ebx
top:  cmpl (%ebx), %eax
      je done
      addl $4, %ebx
      jmp top
done:
```

2.

```
snc:
   pushl %ecx
   pushl %esi
   pushl %edi
   movl 24(%esp), %ecx
   movl 20(%esp), %esi
   movl 16(%esp), %edi
   rep movsb
   popl %edi
   popl %esi
   popl %ecx
   ret
```

3a. The key point was that the machine uses a scan code rather than ASCII.

```
     movl $0,%edx
     cmpb $0x1e, %al
     jnz w
     movl $1,%edx
  w: addl $0,%edx  # dummy instruction
```

3b.

```
movl 16(%esp), %edx  # get saved EFLAGS value from stack
andl $1, %edx
```

Note that solutions based on JC/JNC instead of using the saved EFLAGS value on the stack received only 5 points. Given the information in our course, you cannot be sure that the EFLAGS register is still intact after the interrupt action, subsequent PUSH instructions, etc.

Similarly, you cannot tell from our course material whether instructions like

```
movl %eflags, %eax
```

are legal or not. (They aren't.)

4. C uses row-major order. The element **m[5][5]** will be 14 words from the end of row 5, thus 15 words from the beginning of row 6; then there will be 20 words to the beginning of row 5; then 7 more words to **m[7][7]**. That's a total distance of 42 words.

In general, the element **m[i][j]** is at the word 20i+j words past the beginning of **m**. So, just subtract, getting a distance of 42 words.