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.

Remember: $2^{10} = 1024$ and $2^{16} = 65536$. This may help your calculations.

1. For each of the following operations, give a single assembly language instruction which will accomplish the task:

(a) (5) Copy the number 12 to EDX.

(b) (5) Copy the number 12 to the word in memory pointed to by EDX.

(c) (5) Copy the number 12 to the byte in memory pointed to by EDX.

(d) (5) Copy the number 12 to the word in memory EDX bytes past location 200.

2. (10) There are two systems described in Sec. 3.2 of our notes on Data Storage and Representation for storing floating-point numbers, one a simple system and the other being the IEEE standard. For each of the following, state whether storage in the given system is unique or allows multiple representations of a number:

(a) The simple system.

(b) The IEEE system.

3. (10) Suppose the source code for a program contains the instruction **mov 4(%esp), %ecx**. When we run **as -a**, we find that the corresponding machine code is 8B4C2404. Suppose we accidentally run **cat** on the executable program file. Give a complete list, if any, of the printable characters we can be sure to see on the screen.

4. (10) Consider the instruction **shll $23, %eax**, with c(EAX) = x being considered a signed number. Fill in the blank: If x happens to be nnonegative, then the use of this instruction will be safe as long as x $\leq$ _____.

5. (25) The code below copies a string which starts at **x** to **y**, in reverse order. Fill in the blanks.

```
    movl n, ___________
    movl $x, %eax
    movl $y, %ebx
    addl n, %ebx
    decl %ebx
top:
    movb (%eax), %dl
    movb %dl, __________
    incl %eax
    ------------------
    decl %ecx
    jns top
```

6. (15) Consider the code

```
x: .long 65574
y: .string "cumulus"
```

Find base-10 values m and n such that

```
x: .byte u
   .byte v
   .word n
y: .string "cumulus"
```

produces the same effect, i.e. the same set of bytes when the program is loaded into memory to be run, as the original code.

7. (10) A program includes the line

```
s: .string "\0+\0\0rst"
```

where the first, third and fourth characters are null. During debugging, if we type

```
(gdb) p s
```

in GDB, what will be printed out?

1.

```
movl $12, %edx
movl $12, (%edx)
movb $12, (%edx)
movl $12, 200(%edx)
```

2. (a) multiple; (b) unique

3. L,$

4. We need to avoid changing a positive to a negative, i.e. letting a 1 bit get to the MSB position. Since we are shifting left 23 bit positions, this implies that our number must be confined to Bits 7-0 (where Bit 0 is the LSB), which in turn means a number at most $2^8 - 1 = 511$.

5.

```
%ecx
(%ebx)
decl %ebx
```

6. The number 65574 has the hex form 0x00010026. Now, to put this in byte/byte/word form, note that that first byte will have the lowest address, and thus due to little-endianness is the least significant byte of the number, i.e. 0x26, i.e. 38. Continuing this reasoning, we get u = 38, v = 0 and w = 1.

7. The print (**p**) command prints out one 32-bit quantity, i.e. one word. Thus it will print out the first 4 bytes starting at **s**. The fact that we originally filled those bytes

using the **.string** directive is irrelevant; remember, there are no data types at the machine level. So, GDB treats the first 4 bytes as an integer, printing them out in base-10 (not hex, since there was no `/x` in the command). That number is $43 \times 256 = 11008$.