Andrew login ID: __________________________
Full Name: __________________________

CS 15-213, Fall 2003

Exam 1

October 7, 2003

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name and Andrew login ID on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maximum score of 55 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like. You may use a calculator, but no laptops or other wireless devices. Good luck!

<table>
<thead>
<tr>
<th>1 (09):</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (10):</td>
</tr>
<tr>
<td>3 (04):</td>
</tr>
<tr>
<td>4 (08):</td>
</tr>
<tr>
<td>5 (06):</td>
</tr>
<tr>
<td>6 (06):</td>
</tr>
<tr>
<td>7 (08):</td>
</tr>
<tr>
<td>8 (04):</td>
</tr>
</tbody>
</table>

TOTAL (55):
**Problem 1. (9 points):**
For this problem, assume the following:

- We are running code on a 6-bit machine using two’s complement arithmetic for signed integers.
- short integers are encoded using 3 bits.
- Sign extension is performed whenever a short is casted to an int.
- Right shifts ints are arithmetic.

Fill in the empty boxes in the table below. The following definitions are used in the table:

```c
short sy = -3;
int y = sy;
int x = -17;
unsigned ux = x;
```

Note: You need not fill in entries marked with “–”.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Decimal Representation</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>–</td>
<td>–6</td>
<td></td>
</tr>
<tr>
<td>–</td>
<td>–</td>
<td>01 0010</td>
</tr>
<tr>
<td>ux</td>
<td></td>
<td></td>
</tr>
<tr>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x &gt;&gt; 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>–TMin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin + Tmin</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 2. (10 points):

Consider the following 12-bit floating point representation based on the IEEE floating point format:

- There is a sign bit in the most significant bit.
- The next $k = 4$ bits are the exponent. The exponent bias is 7.
- The last $n = 7$ bits are the significand.

Numeric values are encoded in this format as a value of the form $V = (-1)^s \times M \times 2^E$, where $s$ is the sign bit, $E$ is exponent after biasing, and $M$ is the significand.

Part I

How many FP numbers are in the following intervals $[a, b)$? For each interval $[a, b)$, count the number of $x$ such that $a \leq x < b$.

A. Interval $[1, 2)$: _________

B. Interval $[2, 3)$: _________

Part II

Answer the following problems using either decimal (e.g., 1.375) or fractional (e.g., $11/8$) representations for numbers that are not integers.

A. For denormalized numbers:
   (a) What is the value $E$ of the exponent after biasing? _________
   (b) What is the largest value $M$ of the significand? _________

B. For normalized numbers:
   (a) What is the smallest value $E$ of the exponent after biasing? _________
   (b) What is the largest value $E$ of the exponent after biasing? _________
   (c) What is the largest value $M$ of the significand? _________
Part II

Fill in the blank entries in the following table giving the encodings for some interesting numbers.

<table>
<thead>
<tr>
<th>Description</th>
<th>$E$</th>
<th>$M$</th>
<th>$V$</th>
<th>Binary Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0 0000 0000000</td>
</tr>
<tr>
<td>Smallest Positive (nonzero)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Largest denormalized</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smallest positive normalized</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 3. (4 points):

Consider the following C functions and assembly code:

```c
int fun4(int *ap, int *bp)
{
    int a = *ap;
    int b = *bp;
    return a + b;
}

int fun5(int *ap, int *bp)
{
    int b = *bp;
    *bp += *ap;
    return b;
}

int fun6(int *ap, int *bp)
{
    int a = *ap;
    *bp += *ap;
    return a;
}
```

Which of the functions compiled into the assembly code shown? __________
Problem 4. (8 points):
Consider the following four C and IA32 functions. Next to each of the four IA32 functions, write the name of the C function that it implements. If the assembly routine doesn’t match any of the above functions, write NONE. To save space, the startup code for each IA32 function is omitted:

```c
int winter(int foo[8][12],
           int i, int j)
{
    return foo[i][j];
}

int *spring(int foo[8][12],
            int i, int j)
{
    return foo[i];
}

int summer(int** foo,
           int i, int j)
{
    return foo[i][j];
}

int *fall(int** foo,
          int i, int j)
{
    return foo[i];
}
```

```assembly
pushl %ebp
movl %esp,%ebp

movl 8(%ebp),%edx
movl 12(%ebp),%eax
movl %ebp,%esp
popl %ebp
movl (%edx,%eax,4),%eax
ret

movl 8(%ebp),%edx
movl 12(%ebp),%eax
movl %ebp,%esp
movl (%edx,%eax,4),%eax
movl %ebp,%esp
popl %ebp
movl (%eax,%edx,4),%eax
ret

movl 8(%ebp),%ecx
movl 12(%ebp),%eax
movl 16(%ebp),%edx
movl (%ecx,%eax,4),%eax
movl %ebp,%esp
popl %ebp
movl (%eax,%edx,4),%eax
ret

movl 8(%ebp),%eax
leal (%eax,%eax,2),%eax
sall $4,%eax
addl 8(%ebp),%eax
movl %ebp,%esp
popl %ebp
ret

movl 8(%ebp),%eax
movl 12(%ebp),%edx
movl 16(%ebp),%ecx
sall $2,%ecx
leal (%edx,%edx,2),%edx
sall $4,%edx
addl %edx,%ecx
movl %ebp,%esp
popl %ebp
movl (%eax,%ecx),%eax
ret
```
Problem 5. (6 points):

Consider the following data type definition:

typedef struct {
    char c;
    double d;
    short s;
    double *pd;
    float f;
    char *pc;
} struct1;

Using the template below (allowing a maximum of 40 bytes), indicate the allocation of data for a structure of type struct1. Mark off and label the areas for each individual element (there are 6 of them). Cross hatch the parts that are allocated, but not used (to satisfy alignment). **Clearly indicate the right hand (end) boundary of the data structure with a vertical line.**
Problem 6. (6 points):

Consider the following IA32 code for a procedure called mystery:

```assembly
mystery:
    pushl %ebp
    movl %esp,%ebp
    movl 12(%ebp),%edx
    movl 16(%ebp),%eax
    addl 8(%ebp),%edx
    testl %edx,%edx
    jle .L4
    .L6:
        incl %eax
        decl %edx
        testl %edx,%edx
        jg .L6
    .L4:
        movl %ebp,%esp
        popl %ebp
        ret
```

Based on the assembly code, fill in the blanks below in mystery’s C source code. (Note: you may only use symbolic variables from the source code in your expressions below - do not use register names.)

```c
int mystery(int a, int b, int c) {
    int x, y;

    y = _____________;

    for (____________; ____________; _____________) {

        _____________;
    }

    return __________;
}
```
The next problem concerns the following C code:

```c
/* copy string x to buf */
void foo(char *x) {
    int buf[1];
    strcpy((char *)buf, x);
}

void callfoo() {
    foo("abcdefghi");
}
```

Here is the corresponding machine code on a Linux/x86 machine:

```
080484f4 <foo>:
  080484f4: 55 pushl %ebp
  080484f5: 89 e5 movl %esp,%ebp
  080484f7: 83 ec 18 subl $0x18,%esp
  080484fa: 8b 45 08 movl 0x8(%ebp),%eax
  080484fd: 83 c4 f8 addl $0xffffffff8,%esp
  08048500: 50 pushl %eax
  08048501: 8d 45 fc leal 0xfffffffc(%ebp),%eax
  08048504: 50 pushl %eax
  08048505: e8 ba fe ff ff call 80483c4 <strcpy>
  0804850a: 89 ec movl %ebp,%esp
  0804850c: 5d popl %ebp
  0804850d: c3 ret

08048510 <callfoo>:
  08048510: 55 pushl %ebp
  08048511: 89 e5 movl %esp,%ebp
  08048513: 83 ec 08 subl $0x8,%esp
  08048516: 83 c4 f4 addl $0xffffffff4,%esp
  08048519: 68 9c 85 04 08 pushl $0x804859c  # push string address
  0804851e: e8 d1 ff ff ff call 80484f4 <foo>
  08048523: 89 ec movl %ebp,%esp
  08048525: 5d popl %ebp
  08048526: c3 ret
```
Problem 7. (8 points):
This problem tests your understanding of the stack discipline and byte ordering. Here are some notes to help you work the problem:

- `strcpy(char *dst, char *src)` copies the string at address `src` (including the terminating `'\0'` character) to address `dst`. It does **not** check the size of the destination buffer.

- Recall that Linux/x86 machines are Little Endian.

- You will need to know the hex values of the following characters:

<table>
<thead>
<tr>
<th>Character</th>
<th>Hex value</th>
<th>Character</th>
<th>Hex value</th>
</tr>
</thead>
<tbody>
<tr>
<td>'a'</td>
<td>0x61</td>
<td>'f'</td>
<td>0x66</td>
</tr>
<tr>
<td>'b'</td>
<td>0x62</td>
<td>'g'</td>
<td>0x67</td>
</tr>
<tr>
<td>'c'</td>
<td>0x63</td>
<td>'h'</td>
<td>0x68</td>
</tr>
<tr>
<td>'d'</td>
<td>0x64</td>
<td>'i'</td>
<td>0x69</td>
</tr>
<tr>
<td>'e'</td>
<td>0x65</td>
<td>`'\0'</td>
<td>0x00</td>
</tr>
</tbody>
</table>

Now consider what happens on a Linux/x86 machine when `call foo` calls `foo` with the input string “abcdefgghi”.

A. List the contents of the following memory locations immediately after `strcpy` returns to `foo`. Each answer should be an unsigned 4-byte integer expressed as 8 hex digits.

- `buf[0] = 0x_________`
- `buf[1] = 0x_________`
- `buf[2] = 0x_________`

B. Immediately **before** the `ret` instruction at address `0x0804850d` executes, what is the value of the frame pointer register `%ebp`?

- `%ebp = 0x_________`

C. Immediately **after** the `ret` instruction at address `0x0804850d` executes, what is the value of the program counter register `%eip`?

- `%eip = 0x_________`
Problem 8. (4 points):
Consider the following fragment of IA32 code taken directly from the C standard library:

0x400446e3: call 0x400446e8
0x400446e8: popl %eax

After the popl instruction completes, what hex value does register %eax contain?