**EECS 213 Fall 2007**

**Midterm Exam**

**Instructions:**

- Make sure that your exam is not missing any sheets, then write your full name 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 CLOSED BOOK. You may use one sheet of notes (both sides). Good luck!

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (6):</td>
<td></td>
</tr>
<tr>
<td>2 (12):</td>
<td></td>
</tr>
<tr>
<td>3 (14):</td>
<td></td>
</tr>
<tr>
<td>4 (7):</td>
<td></td>
</tr>
<tr>
<td>5 (10):</td>
<td></td>
</tr>
<tr>
<td>6 (8):</td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL (55):</strong></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1. (6 points):
Consider the following datatype definitions on an IA32 (x86) machine.

```c
typedef struct {
    char c;
    double *p;
    int i;
    double d;
    short s;
} struct1;

typedef union {
    char c;
    double *p;
    int i;
    double d;
    short s;
} union1;
```

A. Using the template below (allowing a maximum of 32 bytes), indicate the allocation of data for a structure of type `struct1`. Mark off and label the areas for each individual element (there are 5 of them). Cross hatch the parts that are allocated, but not used (to satisfy alignment).

Assume the alignment rules discussed in lecture: data types of size $x$ must be aligned on $x$-byte boundaries. Clearly indicate the right hand boundary of the data structure with a vertical line.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
|-----------------------------------------------|
```

B. How many bytes are allocated for an object of type `struct1`?

C. What alignment is required for an object of type `struct1`? (If an object must be aligned on an $x$-byte boundary, then your answer should be $x$.)

D. If we define the fields of `struct1` in a different order, we can reduce the number of bytes wasted by each variable of type `struct1`. What is the number of unused, allocated bytes in the best case?

E. How many bytes are allocated for an object of type `union1`?

F. What alignment is required for an object of type `union1`? (If an object must be aligned on an $x$-byte boundary, then your answer should be $x$.)
Problem 2. (12 points):

In the following questions assume the variables $a$ and $b$ are signed integers and that the machine uses two’s complement representation. Also assume that $\text{MAX}_\text{INT}$ is the maximum integer, $\text{MIN}_\text{INT}$ is the minimum integer, and $W$ is one less than the word length (e.g., $W = 31$ for 32-bit integers).

Match each of the descriptions on the left with a line of code on the right (write in the letter). You will be given 2 points for each correct match.

<table>
<thead>
<tr>
<th>1. One’s complement of $a$</th>
<th>a. $\tilde{(\tilde{a} \mid (b \uparrow (\text{MIN}<em>\text{INT} + \text{MAX}</em>\text{INT})))}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. $a$</td>
<td>b. $((a \uparrow b) &amp; \tilde{b}) \mid (\tilde{a} \uparrow b) &amp; b$</td>
</tr>
<tr>
<td></td>
<td>c. $1 + (a \ll 3) + \tilde{a}$</td>
</tr>
<tr>
<td>3. $a &amp; b$</td>
<td>d. $(a \ll 4) + (a \ll 2) + (a \ll 1)$</td>
</tr>
<tr>
<td></td>
<td>e. $((a &lt; 0) ? (a + 3) : a) \gg 2$</td>
</tr>
<tr>
<td>4. $a \ast 7$</td>
<td>f. $a \uparrow (\text{MIN}<em>\text{INT} + \text{MAX}</em>\text{INT})$</td>
</tr>
<tr>
<td></td>
<td>g. $\tilde{(a \mid (\tilde{a} + 1))} \gg W) &amp; 1$</td>
</tr>
<tr>
<td>5. $a \div 4$</td>
<td>h. $\tilde{(a \gg W)} \ll 1)$</td>
</tr>
<tr>
<td></td>
<td>i. $a \gg 2$</td>
</tr>
<tr>
<td>6. $(a &lt; 0) ? 1 : -1$</td>
<td></td>
</tr>
</tbody>
</table>
Problem 3. (14 points):
Consider the following 8-bit floating point representation based on the IEEE floating point format:

- There is a sign bit in the most significant bit.
- The next 3 bits are the exponent. The exponent bias is $2^{3-1} - 1 = 3$.
- The last 4 bits are the fraction.
- The representation encodes numbers of the form: $V = (-1)^s \times M \times 2^E$, where $M$ is the significand and $E$ is the biased exponent.

The rules are like those in the IEEE standard (normalized, denormalized, representation of 0, infinity, and NAN). FILL in the table below. Here are the instructions for each field:

- **Binary:** The 8 bit binary representation.
- **M:** The value of the significand. This should be a number of the form $x \div 2^y$, where $x$ is an integer, and $y$ is an integral power of 2. Examples include 0, $\frac{3}{4}$.
- **E:** The integer value of the exponent.
- **Value:** The numeric value represented.

Note: you need not fill in entries marked with "—".

<table>
<thead>
<tr>
<th>Description</th>
<th>Binary</th>
<th>M</th>
<th>E</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minus zero</td>
<td></td>
<td></td>
<td></td>
<td>$-0.0$</td>
</tr>
<tr>
<td>—</td>
<td>0 100 0101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smallest denormalized (negative)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Largest normalized (positive)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One</td>
<td></td>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td>5.5</td>
</tr>
<tr>
<td>Positive infinity</td>
<td></td>
<td>—</td>
<td>—</td>
<td>$+\infty$</td>
</tr>
</tbody>
</table>
Problem 4. (7 points):
Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    sall $4,%eax
    subl 8(%ebp),%eax
    movl %ebp,%esp
    popl %ebp
    ret

foo2:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    testl %eax,%eax
    jge .L4
    addl $15,%eax
.L4:
    sarl $4,%eax
    movl %ebp,%esp
    popl %ebp
    ret

foo3:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    shrl $31,%eax
    movl %ebp,%esp
    popl %ebp
    ret

int choice1(int x)
{
    return (x < 0);
}

int choice2(int x)
{
    return (x << 31) & 1;
}

int choice3(int x)
{
    return 15 * x;
}

int choice4(int x)
{
    return (x + 15) / 4;
}

int choice5(int x)
{
    return x / 16;
}

int choice6(int x)
{
    return (x >> 31);
}
```

Fill in your answers here:
foo1 corresponds to choice ________
foo2 corresponds to choice ________
foo3 corresponds to choice ________
Problem 5. (10 points):
Consider a 6-bit two’s complement representation. Fill in the empty boxes in the following table:

<table>
<thead>
<tr>
<th>Number</th>
<th>Decimal Representation</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td></td>
<td>01 1010</td>
</tr>
<tr>
<td>n/a</td>
<td></td>
<td>10 0110</td>
</tr>
<tr>
<td>Tmax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmax+TMax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin+TMin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmax+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>
</tbody>
</table>
Problem 6. (8 points):
Consider the following assembly representation of a function foo containing a for loop:

```
foo:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
    movl 8(%ebp),%ebx
    leal 2(%ebx),%edx
    xorl %ecx,%ecx
    cmpl %ebx,%ecx
    jge .L4
.L6:
    leal 5(%ecx,%edx),%edx
    leal 3(%ecx),%eax
    imull %eax,%edx
    incl %ecx
    cmpl %ebx,%ecx
    jl .L6
.L4:
    movl %edx,%eax
    popl %ebx
    movl %ebp,%esp
    popl %ebp
    ret
```

Fill in the blanks to provide the functionality of the loop:

```
int foo(int a)
{
    int i;
    int result = _____________;

    for( _______; _______; i++ ) {
        _________________;
        _________________;
    }
    return result;
}
```