CSE2003: System Programming
Midterm Exam. (Spring 2009)

3:00PM - 4:20PM, April 22, 2009.
Instructor: Jin-Soo Kim

Student ID: _____________
Name: _____________
Codename: _____________

<table>
<thead>
<tr>
<th>Q1 (20)</th>
<th>Q5 (40)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2 (40)</td>
<td>Q6 (60)</td>
</tr>
<tr>
<td>Q3 (40)</td>
<td>Q7 (30)</td>
</tr>
<tr>
<td>Q4 (20)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total (250)</td>
</tr>
</tbody>
</table>

1. Write the full name of the following acronym. (20 points)
(1) BIT ( )
(2) ALU ( )
(3) CPU ( )
(4) CISC ( )
(5) EIP ( )

2. Which of the following IA-32 instructions requires a memory access? Write "Yes" if the corresponding instruction requires a memory access to complete. Otherwise, write "No". (40 points. Do not take guesses. Each wrong answer will be penalized with -4 points.)

(1) movl %eax, %edx ( )
(2) movl % (%ebp), %eax ( )
(3) leal 16(%ebp, %eax, 4), %ecx ( )
(4) call mult32bits ( )
(5) imull %edx ( )
(6) addl (%ebp, %ecx), %ebx ( )
(7) retr ( )
(8) pushl %eax ( )
(9) jl L57 ( )
(10) jmp *%eax ( )
3. What is the output of the following program when it is executed on Linux/IA32? Fill in each blank with a correct value. (40 points)

```c
#include <stdio.h>
union {  
    int i;
    short s[2];
    unsigned char c[4];
} u;

int main()
{
    int a0, a1;
    u.i = 0xbadbeef;
    a0 = (int) u.s[0];
    a1 = (int) u.s[1];
    printf("sizeof(int)=%d, sizeof(short)=%d, sizeof(char)=%d\n",  
           sizeof(int), sizeof(short), sizeof(char));
    printf("sizeof(u.i)=%d\n", sizeof(u.i));
    printf("sizeof(u.s)=%d, sizeof(u.s[0])=%d\n",  
           sizeof(u.s), sizeof(u.s[0]));
    printf("sizeof(u.c)=%d, sizeof(u.c[0])=%d\n",  
           sizeof(u.c), sizeof(u.c[0]));
    printf("%08x, %08x\n", a0, a1);
    printf("u.c=0x\%02x 0x\%02x 0x\%02x 0x\%02x", u.c[0], a.c[1], u.c[2], u.c[3]);
}
```

$ ./a.out

```
sizeof(int)=4, sizeof(short)=2, sizeof(char)=1
sizeof(u.i)=
sizeof(u.s)=, sizeof(u.s[0])=
sizeof(u.c)=, sizeof(u.c[0])=
sizeof(u)=
```

4. After compiling the following C program on Linux/IA32, we obtained the assembly codes shown on the right. What are real values of A and B in the original C program? (20 points)

```c
#define A 42  /* ??? */
#define B 67  /* ??? */

int amxb (int x)
{
    return A*x + B;
}
```

```asm
pushl %ebp
movl %esp, %ebp
movl 8(%ebp), %eax
lea 4(%eax, %eax, 4), %eax
```

```
```
5. Consider a 6-bit two's complement integer representation. Fill in the empty boxes in the following table. (40 points)

<table>
<thead>
<tr>
<th>Number</th>
<th>Decimal Representation</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6(_{10})</td>
<td>6</td>
<td>000110</td>
</tr>
<tr>
<td>29(_{10})</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>-1(_{10})</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>-24(_{10})</td>
<td>-24</td>
<td></td>
</tr>
<tr>
<td>1100 (_2)</td>
<td></td>
<td>001100</td>
</tr>
<tr>
<td>111000 (_2)</td>
<td></td>
<td>111000</td>
</tr>
<tr>
<td>Zero</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>The largest number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The smallest number</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6. Consider the following 5-bit floating point representation based on the IEEE floating point format. There is a sign bit in the most significant bit. The next three bits are the exponent, with an exponent bias of 3. The last bit is the fraction. The rules are like those in the IEEE standard (normalized, denormalized, representation of zero, infinity, and NaN).

<table>
<thead>
<tr>
<th>Sign (1 bit)</th>
<th>Exponent (3 bits)</th>
<th>Fraction (1 bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Fill in the empty boxes in the following table. (50 points)

<table>
<thead>
<tr>
<th>Number</th>
<th>Decimal Representation</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Zero</td>
<td>+0.0</td>
<td>00000</td>
</tr>
<tr>
<td>Negative Zero</td>
<td>-0.0</td>
<td></td>
</tr>
<tr>
<td>0.75 (_{10})</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>One</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Positive Infinity</td>
<td>+(\infty)</td>
<td></td>
</tr>
<tr>
<td>Negative Infinity</td>
<td>-(\infty)</td>
<td></td>
</tr>
<tr>
<td>Not-a-Number</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>The largest number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The smallest positive number</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(2) Show all the possible non-zero values that are represented in denormalized form. (10 points)
7. Consider the following IA-32 assembly code fragment obtained by the `objdump` utility. As can be seen in the code, the function $g()$ is called by $f()$.

```
08048384 <g>:
80483841  55          push %ebp
80483842  89 e5       mov %esp,%ebp
80483844  8b 45 0c    mov 0xc(%esp),%eax
80483847  c3 43 08    add 0x8(%ebp),%eax
8048384a  5d          pop %ebp
8048384b  c3          ret

0804838f <f>:
804838f1  55          push %ebp
804838f2  89 e5       mov %esp,%ebp
804838f4  83 ec 08    sub $0x8,%esp
804838f7  c7 44 24 04 18 00 00    movl 0x18,0x4(%esp)
804838fa  00          
80483901  c7 04 24 00 00 00 00    movl $0x00,(%esp)
8048390e  89 e5       mov %esp,%ebp
80483911  89 e5       mov %ebp,%esp
80483914  5d          pop %ebp
80483915  c3          ret
```

(1) How many arguments does the function $f()$ take? (5 points)

(2) How many arguments does the function $g()$ take? (5 points)

(3) What does the function $g()$ compute? Write the equivalent C program. (5 points)

(4) When the CPU is about to execute the `add` instruction at 0x0804838a in $g()$, show the values stored in the following locations of the stack. (15 points)