15-213/18-243, Spring 2011

Exam 1

Thursday, March 3, 2011 (v2)

Instructions:

- Make sure that your exam is not missing any sheets, then write your Andrew login ID, full name, and section on the front.

- This exam is closed book, closed notes. You may not use any electronic devices.

- 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 100 points.

- The problems are of varying difficulty. The point value of each problem is indicated. Good luck!

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(12):</td>
</tr>
<tr>
<td>2</td>
<td>(17):</td>
</tr>
<tr>
<td>3</td>
<td>(13):</td>
</tr>
<tr>
<td>4</td>
<td>(11):</td>
</tr>
<tr>
<td>5</td>
<td>(20):</td>
</tr>
<tr>
<td>6</td>
<td>(12):</td>
</tr>
<tr>
<td>7</td>
<td>(15):</td>
</tr>
<tr>
<td>TOTAL</td>
<td>(100):</td>
</tr>
</tbody>
</table>
Problem 1. (12 points):

Multiple choice.

Write the correct answer for each question in the following table:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Consider an int *a and an int n. If the value of %ecx is a and the value of %edx is n, which of the following assembly snippets best corresponds to the C statement return a[n]?

(a) ret (%ecx,%edx,4)
(b) leal (%ecx,%edx,4),%eax
   ret
(c) mov (%ecx,%edx,4),%eax
   ret
(d) mov (%ecx,%edx,1),%eax
   ret

2. Which of the following 8 bit floating point numbers (1 sign, 3 exponent, 4 fraction) represent NaN?

(a) 1 000 1111
(b) 0 111 1111
(c) 0 100 0000
(d) 1 111 0000

3. %rsp is 0xdeadbeefdeadd0d0. What is the value in %rsp after the following instruction executes?

pushq %rbx

(a) 0xdeadbeefdeadd0d4
(b) 0xdeadbeefdeadd0d8
(c) 0xdeadbeefdeadd0cc
(d) 0xdeadbeefdeadd0c8

4. How many lines does a direct-mapped cache have in a set?

(a) 0
(b) 1
(c) 2
(d) 4
5. On an x86_64 Linux system, which of these takes up the most bytes in memory?
   (a) char a[7]
   (b) short b[3]
   (c) int *c
   (d) float d

6. Two-dimensional arrays are stored in ______ order, to help with cache performance.
   (a) column-major
   (b) row-major
   (c) diagonal-major
   (d) Art-major

7. Which register holds the first argument when an argument is called in IA32 (32 bit) architecture?
   (a) edi
   (b) esi
   (c) eax
   (d) None of the above

8. What is the C equivalent of mov 0x10(%rax,%rcx,4),%rdx
   (a) rdx = rax + rcx + 4 + 10
   (b) *(rax + rcx + 4 + 10) = rdx
   (c) rdx = *(rax + rcx*4 + 0x10)
   (d) rdx = *(rax + rcx + 4 + 0x10)

9. What is the C equivalent of leal 0x10(%rax,%rcx,4),%rdx
   (a) rdx = 10 + rax + rcx + 4
   (b) rdx = 0x10 + rax + rcx*4
   (c) rdx = *(0x10 + rax + rcx*4)
   (d) *(0x10 + rax + rcx + 4) = rdx

10. What is the C equivalent of mov %rax,%rcx
    (a) rcx = rax
    (b) rax = rcx
    (c) rax = *rcx
    (d) rcx = *rax
11. In x86 (IA32) an application’s stack grows from
   (a) High memory addresses to low memory addresses
   (b) Low memory addresses to high memory addresses
   (c) Both towards higher and lower addresses depending on the action
   (d) Stacks are a fixed size and do not grow.

12. True or False: In x86_64 the %rbp register can be used as a general purpose register.
   • True
   • False
Problem 2. (17 points):

Bits.

A. Convert the following from decimal to 8-bit two’s complement.

\[
67 = \\
-35 = 
\]

B. Please solve the following are datalab-style puzzle. Please write brief and clear comments. You may use large constants. eg. instead of saying \((1 << 16)\), you may use 0x10000.

```c
/*
 * reverseBytes - reverse bytes
 * Example: reverseBytes(0x12345678) = 0x78563412
 * Legal ops: ! ~ & ^ | + << >>
 */
int reverseBytes(int x)
{
}
```

C. Assume x and y are of type int. For each expression below, give values for x and y which make the expression false, or write "none" if the expression is always true.

- \(((x \ ^ \ y) < 0)\)
- \(((\~(x \ | \ (\~x + 1))) \ >> \ 31) \ & \ 0x1) == !x\)
- \((x \ ^ \ (x>>31)) - (x>>31) > 0\)
- \((x \ >> \ 31) + 1) >= 0\)
- \((!x \ | \ !!y) == 1\)
Problem 3. (13 points):

*Floats.*

Consider a 6-bit floating point data type with 3 exponent bits and 3 fraction bits (there is no sign bit, so the data type can only represent positive numbers). Assume that this data type uses the conventions presented in class, including representations on NaN, infinity, and denormalized values.

A. What is the bias?

B. What is the largest value, other than infinity, that can be represented?

C. What is the smallest value, other than zero, that can be represented?

D. Fill in the following table. Use round-to-even. If a number is too big to represent, use the representation of infinity, and if it is too small to represent, use the representation of 0. Value should be written in decimal.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Value</th>
<th>Bits</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>011 000</td>
<td>1</td>
<td>011 010</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>111 010</td>
<td></td>
</tr>
<tr>
<td>110 001</td>
<td>9 1/2</td>
<td></td>
<td>3/32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110 001</td>
<td>8 1/2</td>
</tr>
</tbody>
</table>
Problem 4. (11 points):

**Structs.**

Consider the following struct:

```c
typedef struct
{
    char a[3];
    short b[3];
    double c;
    long double d;
    int* e;
    int f;
} JBOB;
```

A. Show how the struct above would appear on a 64-bit (“x86_64”) Linux machine. Label the bytes that belong to the various fields with their names and clearly mark the end of the struct. Use x’s to indicate bytes that are allocated in the struct but are not used. A long double is 16 bytes long.
B. Rearrange the above fields in foo to conserve the most space in the memory below. Label the bytes that belong to the various fields with their names and clearly mark the end of the struct. Use hatch marks or x’s to indicate bytes that are allocated in the struct but are not used.

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| | | | | | | | | | | | | | | | |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| | | | | | | | | | | | | | | | |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| | | | | | | | | | | | | | | | |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| | | | | | | | | | | | | | | | |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| | | | | | | | | | | | | | | | |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

C. How many bytes are wasted in part A, inside and after the struct, if the next memory value is a pointer?

D. How many bytes are wasted in part B, inside and after the struct, if the next memory value is a pointer?
Problem 5. (20 points):

Assembly/C translation.

Given the following x86 assembly dump, please reconstruct the function in the provided C code.

```c
int mystery(int (*f)(int, int), int* arr, int c)
{
    int i, x;
    if(_________________)
        return ______________;
    x = _________________;
    for(i = ______________; ______________; __________)
        x = ________________________________;
    return x;
}
```

(gdb) disas mystery
Dump of assembler code for function mystery:
0x080483a4 <mystery+0>:push %ebp
0x080483a5 <mystery+1>:mov %esp,%ebp
0x080483a7 <mystery+3>:push %edi
0x080483a8 <mystery+4>:push %esi
0x080483a9 <mystery+5>:push %ebx
0x080483aa <mystery+6>:sub $0xc,%esp
0x080483ab <mystery+9>:mov 0xc(%ebp),%edi
0x080483ac <mystery+12>:mov 0x10(%ebp),%esi
0x080483ad <mystery+15>:test %esi,%esi
0x080483b0 <mystery+17>:jle 0x80483db <mystery+55>
0x080483b2 <mystery+19>:mov (%edi),%edx
0x080483b3 <mystery+21>:cmp $0x1,%esi
0x080483b5 <mystery+23>:jle 0x80483d9 <mystery+53>
0x080483b8 <mystery+24>:mov 0x80483d9 <mystery+53> %edi,%edx
0x080483ba <mystery+26>:mov $0xa,%eax
0x080483b9 <mystery+31>:mov (%edi,%ebx,4),%eax
0x080483be <mystery+34>:mov %eax,0x4(%esp)
0x080483ca <mystery+38>:mov %edx,(%esp)
0x080483cd <mystery+41>:call *0x8(%ebp)
0x080483d0 <mystery+44>:mov %eax,%edx
0x080483d2 <mystery+46>:add $0x1,%ebx
0x080483d5 <mystery+49>:cmp %ebx,%esi
0x080483d7 <mystery+51>:jg 0x80483c3 <mystery+31>
0x080483d9 <mystery+53>:mov %edx,%esi
0x080483db <mystery+55>:mov %esi,%eax
0x080483dd <mystery+57>:add $0xc,%esp
0x080483e0 <mystery+60>:pop %ebx
0x080483e1 <mystery+61>:pop %esi
0x080483e2 <mystery+62>:pop %edi
0x080483e3 <mystery+63>:pop %ebp
0x080483e4 <mystery+64>:ret
End of assembler dump.

A. At address 0x080483a9 we see the instruction push %ebx. Name two things that happen as a result of executing that instruction, and explain why the instruction is necessary.

B. Assume that immediately after executing the instruction at address 0x080483a9 (push %ebx), the value of %esp is 0xffff0000. If that is the case, at which address would one find the argument f?
Problem 6. (12 points):

Stacks.

Given the following function prototypes, and initial lines of IA32 assembly for each function, fill in the stack frame diagram with

- any arguments to the function foo
- the return address
- Any registers stored on the stack by the asm fragment (register names not values)
- The location on the stack pointed to by %esp and %ebp after the execution of the sub instruction.

```c
void foo(char *a, int b);
push %ebp
mov %esp, %ebp
sub $0x10, %esp
```
Problem 7. (15 points):

We will consider performance issues associated with caching the reads from array A. Assume other variables are stored in registers. Also assume A is cache-aligned, and that the cache is cold before running the code.
Consider the following code:

```c
#include N 128

int myst(int[] A)
{
    int i, result;

    for (i = 0; i < N; i++)
        result += A[i]*A[n-i-1];

    return result;
}
```

A. Consider a 64-byte, direct mapped cache with 4 sets. Fill in the values that will be stored in this cache when the code reaches the point `return result;`
B. Consider a 64-byte, two-way set associative cache with 4 sets. Fill in the values that will be stored in this cache when the code reaches the point `return result;` Each rectangle in the table represents 4 bytes.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>