Andrew login ID: _____

Full Name: _____

Section: _____

15-213/18-243, Spring 2011

Exam 1

Thursday, March 3, 2011 (v1)

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!

1 (12):	
2 (17):	
3 (13):	
4 (11):	
5 (20):	
6 (12):	
7 (15):	
TOTAL (100):	

Problem 1. (12 points):

Multiple choice.

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

1	2	3	4	5	6	7	8	9	10
11	12								

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.

```
/*
 * 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.

Bits	Value	Bits	Value
011 000	1		5
	17	111 010	
110 001			3/32
	9 1/2		8 1/2

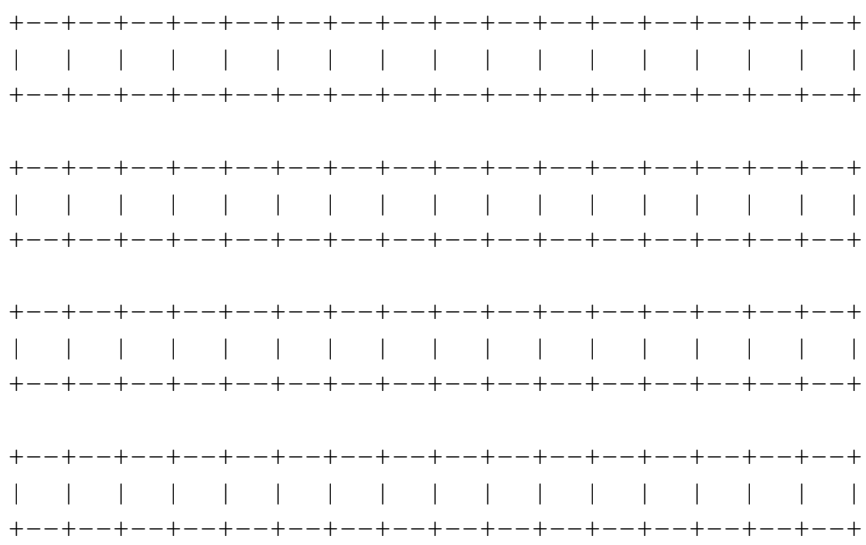
Problem 4. (11 points):

Structs.

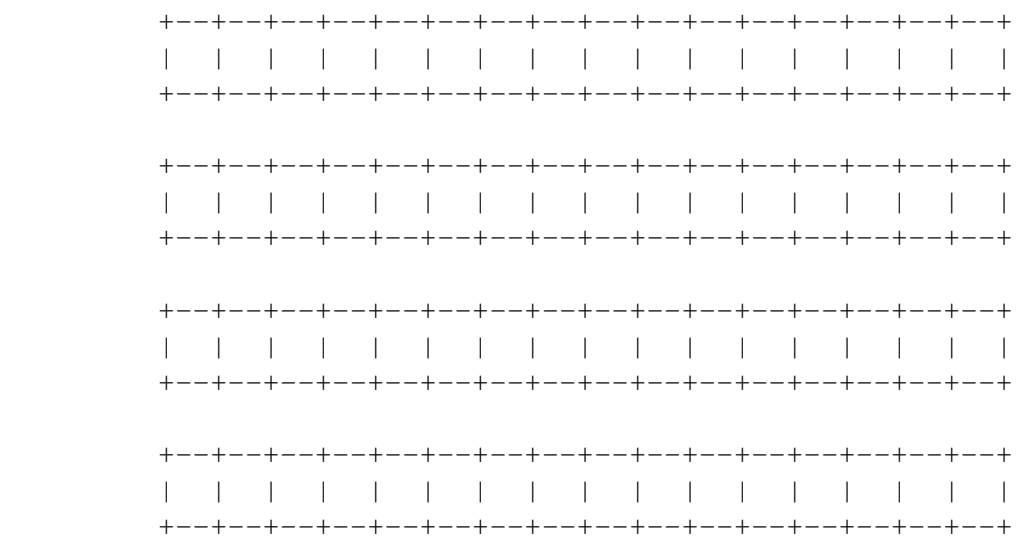
Consider the following struct:

```
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 x86 assembly dump, please fill in the blank lines for the function in the provided C code:

```
int lolwut(char *s)
{
    int i, n;

    n = 0;

    for(i = 0; ____________________; i++)
    {
        if(_____________________________________)
        {
            return -1;
        }
        n = ____________________;
    }
    return _________;
}
```

```
080483a4 <lolwut>:
 80483a4:   55                  push   %ebp
 80483a5:   89 e5               mov    %esp,%ebp
 80483a7:   53                  push   %ebx
 80483a8:   8b 5d 08            mov    0x8(%ebp),%ebx
 80483ab:   0f b6 0b            movzbl (%ebx),%ecx
 80483ae:   ba 00 00 00 00      mov    $0x0,%edx
 80483b3:   84 c9               test   %cl,%cl
 80483b5:   74 31               je     80483e8 <lolwut+0x44>
 80483b7:   8d 41 d0            lea    -48(%ecx),%eax
 80483ba:   ba 00 00 00 00      mov    $0x0,%edx
 80483bf:   3c 09               cmp    $0x9,%al
 80483c1:   76 0c               jbe    80483cf <lolwut+0x2b>
 80483c3:   eb 1e               jmp    80483e3 <lolwut+0x3f>
 80483c5:   83 c3 01            add    $0x1,%ebx
 80483c8:   8d 41 d0            lea    -48(%ecx),%eax
 80483cb:   3c 09               cmp    $0x9,%al
 80483cd:   77 14               ja     80483e3 <lolwut+0x3f>
 80483cf:   8d 14 92            lea    (%edx,%edx,4),%edx
 80483d2:   0f be c1            movsbl %cl,%eax
 80483d5:   8d 54 50 d0         lea    -48(%eax,%edx,2),%edx
 80483d9:   0f b6 4b 01         movzbl 0x1(%ebx),%ecx
 80483dd:   84 c9               test   %cl,%cl
 80483df:   75 e4               jne    80483c5 <lolwut+0x21>
 80483e1:   eb 05               jmp    80483e8 <lolwut+0x44>
 80483e3:   ba ff ff ff ff      mov    -1,%edx
 80483e8:   89 d0               mov    %edx,%eax
 80483ea:   5b                  pop    %ebx
 80483eb:   5d                  pop    %ebp
 80483ec:   c3                  ret
```

A. At address `0x080483a7` 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.

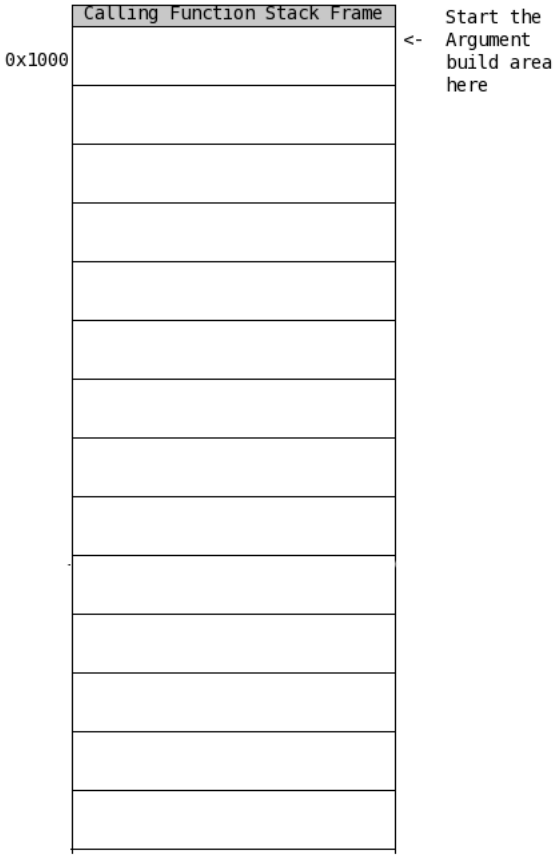
B. Assume that immediately after executing the instruction at address `0x080483a7` (`push %ebx`), the value of `%esp` is `0xffff0000`. If that is the case, at which address would one find the argument `s`?

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 exection of the `sub` instruction.




```
int foo(int a, int b, int c, int d);
push %ebp
mov  %esp,%ebp
push %ebx
sub $0x10,$esp
```

Problem 7. (15 points):

The Hit or Miss Question

Given a 32-bit Linux system that has a 2-way associative cache of size 128 bytes with 32 bytes per block. Long longs are 8 bytes. For all parts, assume that `table` starts at address 0x0.

```
int i;
int j;
long long table[4][8];
for (j = 0; j < 8; j++) {
  for (i = 0; i < 4; i++) {
    table[i][j] = i + j;
  }
}
```

A. This problem refers to code sample 1. In the table below write down in each space whether that element's access will be a hit or a miss. Indicate hits with a 'H' and misses with a 'M'

	0	1	2	3	4	5	6	7
0								
1								
2								
3								

What is the miss rate of this code sample?

```
int i;
int j;
int table[4][8];
for (j = 0; j < 8; j++) {
  for (i = 0; i < 4; i++) {
    table[i][j] = i + j;
  }
}
```

B. This problem refers to code sample above. In the table below write down in each space whether that element's access will be a hit or a miss. Indicate hits with a 'H' and misses with a 'M'

	0	1	2	3	4	5	6	7
0								
1								
2								
3								

What is the miss rate of this code sample?

C. One code sample performs better than the other. Why is this?