Full Name ________________________________

CSCI 2400, Fall 2014
Second Midterm Exam

Instructions:

- Check that your exam has all 5 pages, and write your full name clearly on the front.
- Write your answers in the space provided for each problem. Feel free to use the back of each page to help you determine the answer, but make sure your answer is entered in the space provided on the front of the page.
- This exam is CLOSED BOOK and no electronics are allowed. You can use one page of personal notes and the printed midterm packet of tables. Good luck!

Problem	Page	Possible	Score
1	1	8	
2	2	20	
3	3	20	
4	4	10	
5	4	10	
6	5	22	
Total		90	

1. **[8 Points]** Look at the assembly instructions below. These instructions will be pipelined to speed up the code execution. Unfortunately the code cannot be pipelined as is due to data dependencies. Add 'NOP' instruction in the code below at the appropriate locations to ensure that pipelining can be accomplished. (Hint: draw out a timing diagram of the different pipeline stages)

```
mov $1, %ebx
mov $2, %eax
sub $1, %ebx
mov $5, %edx
mov $3, %ecx
add %eax, %ebx
sub $2, %edx
sub %ecx, %eax
add $1, %ebx
sub $1, %ecx
```

2. **[20 Points]**

Mr. Manning needs to make his 500th career touchdown pass to Mr. Thomas. For the force with which Mr. Manning throws the ball, the projectile motion required to make the pass requires an angle of 30.125 units **precisely**. You are Mr. Manning's subconscious. You need to tell him the angle at which he is supposed to throw the projectile. Unfortunately, he can process only binary with IEEE floating point format. This includes a sign bit, 5-bit exponent field with bias 15 and a 4 bit mantissa / fractional field. Please help Mr. Manning to achieve this feat!

Feel free to use the back of the page to work out your answers below, but let us know if you did.

(a) **[5 Points]** First, convert 30.125 to binary using the given floating point format.

(b) **[5 Points]** What decimal value is represented by the IEEE formatted binary solution found for part 2a?

$value =$ ________________________

Using the current IEEE format and considering that the required angle of the projectile is **precisely** 30.125, can Mr. Manning accomplish his feat? Please justify your answer.

(c) **[10 Points]** If the pass required the angle to be **precisely** 32.125 units, what is the **minimum number of bits** we need for the exponent and the mantissa/fractional field so that Mr. Manning can accomplish the feat? (Hint: the optimal design for storing 32.125 might require an increase or decrease in the number of bits from the IEEE format of the previous parts 2a and 2b of this question, and this could be true for either/both of the exponent and mantissa fields.) Please justify your answer and show your work.

3. **[20 Points]**

```
char *atk = "countermeasure!"

int get_buffer(){
  char buf[8];
  fake_gets(buf);
  return 1;
}
void fake_gets(char* buf){
  int i;
  for(i=0; atk[i]!=0; i++){
    buf[i] = atk[i];
  }
  buf[i]=0;
}
```

```
00216572 <get_buffer>:
  216572: pushl %ebp
  216573: movl %esp,%ebp
  21675: subl $0x14,%esp
  216578: leal -0x8(%ebp),%eax
  21657b: movl %eax,(%esp)
  21657e: call 0x00216214 <fake_gets>
  216583: movl $1,%eax
  216587: movl %ebp, %esp
  216589: popl %ebp
  21658a: ret
```

Similarly to `Gets` from your buffer lab, `fake_gets` writes data in to the array, buf, that is its only argument. To simplify things `fake_gets`' input comes from a null-terminated string stored in memory, called `atk`. (Remember that C stores strings as null-terminated character arrays).

The diagram below represents a section of the stack, with memory addresses and `buf` labeled. The following table of characters' hex values may also be helpful.

u	0x75
c	0x63
t	0x74
!	0x21

e	0x65
o	0x6f
r	0x72
m	0x6d

n	0x6e
s	0x73
u	0x75
a	0x61

Using the code above answer the following questions:

(a) Suppose we have just executed the popl instruction (at 0x00216589) within get_buffer. Fill in the following diagram of the stack with the correct hex values at that point. The value of buf and a few memory addresses have been filled in to get you started.

0x34	0x35	0x36	0x37	0x38	0x39	0x3a	0x3b	0x3c	0x3d	0x3e	0x3f	0x40	0x41	0x42	0x43
63	6f	75													

(b) What is the address of the next instruction that executes after the ret instruction at 0x0021658a?

(c) What is the location of %esp after the execution of the popl instruction at 0x00216589?

(d) To what value is %ebp set after the execution of the popl instruction at 0x00216589?

4. **[10 Points]** The following problem concerns optimizing a procedure for maximum performance on an Intel Pentium IV with the following characteristics of the functional units:

Operation	Latency	Issue Time/Rate
Integer Add	1	1
Integer Multiply	3	1
Floating Point Add	2	1
Floating Point Multiply	4	2
Load or Store (Cache Hit)	1	1

Assume there is one of each functional unit, array1 and array2 have the correct types, e.g. int or floating point. Assume input1, input2, out1, out2 and out3 can be stored in registers. You may use the back of the page.

(a) **[5 Points]**

```
float out1, out2, input1, input2;
    for (i=0; i< length; i++){
        out1 = input1 + array1[i];
        out2 = out1 * array2[i];
    }
```

What is the CPE of this loop?

(b) **[5 Points]**

```
int input1, input2, out3;
    for (i=0; i< length; i++){
        input1 = input1 + array1[i];
        input2 = input2 + array2[i];
        out3 = input2 * input1;
    }
```

What is the CPE of this loop? (Hint: draw out the timing diagram)

5. **[10 Points]** In the following, state whether the statement is true or false. An incorrect answer will cancel a correct answer. You may leave a question blank. The lowest possible score is zero on this question.

(a) ______ A NOP sled is used by an attacker to overcome stack randomization.

(b) ______ Increasing the amount of loop unrolling always improves performance.

(c) ______ Denormalized floating point values achieve worse precision around zero than normalized values.

(d) ______ Always-taken branch prediction exploits looping behavior.

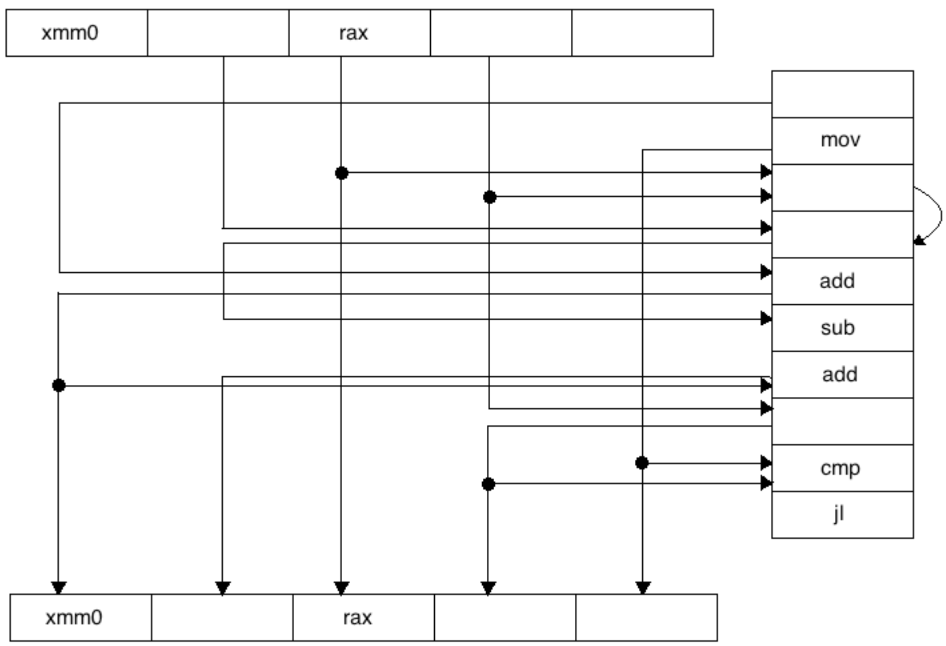
(e) ______ Amdahl's Law says that the speedup in a program is limited by the sequential component that you cannot parallelize.

6. **[22 Points]**

Consider the following x86-64 assembly code for an inner loop:

```
L1:
   movs    $1, %xmm0
   movs    $3, %rcx
   mulss    (%rax,%rbx,4), %xmm1
   adds  $2, %xmm0
   subs    %rax, %xmm1
   adds    %xmm0, %xmm1
   addq    $1, %rbx
   cmpq    %rcx, %rbx
   jl      .L1
```

(a) **[18 Points]** Complete the dataflow diagram below by filling in the blank cells and by drawing two missing arrows (Hint: look at the inputs and outputs of the instructions).



(b) **[4 Points]** Which two registers are on the Critical Path for this loop?