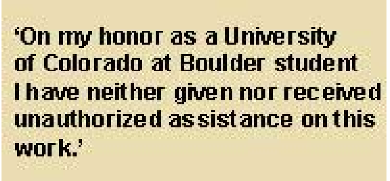
Full Name ____

CSCI 2400, Fall 2014

Second Midterm Exam
SOLUTIONS



'On my honor as a University of Colorado at Boulder student I have neither given nor received unauthorized assistance on this work.'

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.

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

2. **[20 Points]**

(a) 0100111110.

Rubric: Break this as $+1$ **for sign bit,** $+2$ **for exp field and** $+2$ **for mantissa.**

(b) Decimal value is 30 since we do not have enough bits to store the 0.125.

Mr. Manning does not accomplish the feat since we have rounding off error. We need more bits for mantissa.

Rubric: $+3$ **for decimal value and** $+2$ **for remainder of the question.**

(c) $32.125_{10} = 1.00000001 \times 2^5$.

Minimum bits for $exp = 4$.

Since range with $exp = 4$ is $[-6, 7]$.

Minimum bits for frac = 8.

Since the fractional piece requires 8 bits for precise storage.

Rubric: $+2$ **for converting from decimal to binary,** $+4$ **for exp and** $+4$ **for mantissa**

Rubric: -4 **for not showing the work**.

3. **[20 Points]**

```
A1: 63 6f 75 6e 74 65 72 6d 65 61 73 75 72 65 21 00
A2: 0x00216572 aka the beginning of the get_buffer function
A3: 0x40
A4: 'easu' block = 65 61 73 75, but in reverse order if read out,
                0x 75 73 61 65, maybe accept either order for part (d)

0x34 0x35 0x36 0x37 0x38 0x39 0x3a 0x3b 0x3c 0x3d 0x3e 0x3f 0x40 0x41 0x42 0x43
63   6f   75   6e   74   65   72   6d   65   61   73   75   72   65   21   00
c    o    u    n    t    e    r    m    e    a    s    u    r    e    !    0
```

Rubric:

a **+11**

b **+3**

c **+3**

d **+3**

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, input3, out1, and out2 can be stored in registers.

(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?
Answer: 2 (Issue time of the multiply).
(+5 for 2) (+1 for 4)

(b) **[5 Points]**

```
int input1, input2, input3;
    for (i=0; i< length; i++){
        input1 = input1 + array1[i];
        input2 = input2 + array2[i];
        input3 = input2 * input1;
    }
```

What is the CPE of this loop?
Answer: 2 (see timing diagram)
(+5 for 2) (+2.5 for 1) (+1 for 3)

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) __TRUE__ A NOP sled is used by an attacker to overcome stack randomization.

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

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

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

(e) __TRUE __ 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.

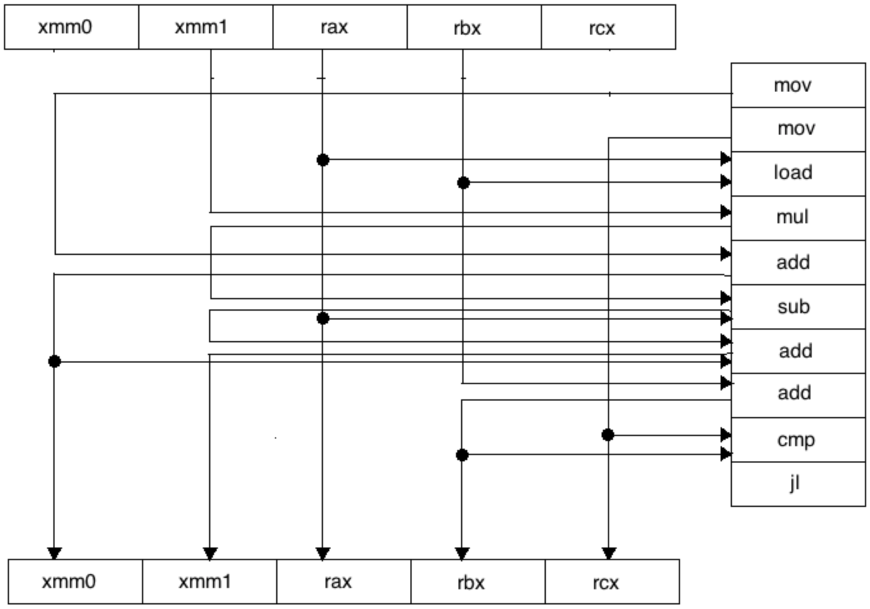


Figure 1: Dataflow diagram

(b) **[4 Points]** Which two registers are on the Critical Path for this loop? xmm1 and rbx