Name: _____

CS 281 — Computer Organization — Fall 2017

Exam #1

This exam should have five pages, including the MIPS instruction reference at the end of the exam. Closed book and notes. Calculators allowed for base-10 work only.

```
mystery:
        li   $s0, 1           #
here:   slt  $s1, $a0, $s0    #
        beq  $s1, $s0, there  #
        sub  $a0, $a0, $a1    #
        j here               #
there:  li   $v0, 1           #
        beq  $a0, $zero, gone #
        li   $v0, 0           #
gone:   jr   $ra             #
```

Problem 1: [15 points]

Below, describe in *English* what the MIPS assembly-language subroutine above does. (That's *what*, not *how*.) To get in the right frame of mind, think about what you would write as a descriptive header comment for the procedure. Feel free to add comments to each line to aid in your comprehension, but they aren't required for full credit. Be as brief and concise as you can.

Problem 2: [20 points]

The MIPS procedure in Problem 1 doesn't use the stack properly. Please fill in the additional code necessary to properly maintain the stack. (Write it in on the previous page, making it clear where your new additions belong with respect to the existing code.)

Problem 3: [24 points]

The questions below refer to the following bit sequences:

```
A:   10111101          B:   01100111
```

a) Assuming that A and B represent unsigned 8-bit integers, what are their respective base-10 values?

b) Assuming that A and B represent 8-bit two's complement signed integers, what are their respective base-10 values?

c) What is the 8-bit sequence corresponding to the sum of these two integers?

d) What is the 8-bit sequence corresponding to the sum of A and A? Does overflow occur?

Problem 4: [20 points]

The company you work for is redesigning their market-leading processor. It currently runs at 2.5 GHz, and they're considering a variety of architectural changes and compiler optimizations to get better performance. The marketing folks' favorite benchmark consists of 750 billion instructions (750,000,000,000). Most of them (40%) are Class A, each of which take 2 cycles to execute. The rest of the instructions are split evenly between Class B, which take 3 cycles, and Class C, which take 5.

    a) What's the average CPI for the benchmark?

    b) How long does the benchmark take to execute the on the 2.5GHz processor?

    c) The marketing folks are hoping that the benchmark will run in 800 seconds on the new machine. What would the clock speed have to be to achieve this goal if nothing else changed?

    d) The compiler team claims they can optimize the generated code so that it uses fewer Class C instructions. (They wouldn't be replaced by other instructions — they'd just disappear.) How many Class C instructions would have to be eliminated from the benchmark program to meet the 800 second target on the original 2.5GHz processor?

Problem 5: [20 points]

a) What is the base-10 value of the BradFloat™ `0 1010 010`?

b) What is the largest possible value (in base 10) that can be represented in a BradFloat™? Explain.

c) One of my many happy customers has requested a BradFloat™ implementation that uses an unsigned exponent field — all three bits would be used as positive contributions. What are the implications of this on the values that could be represented?

MIPS Instruction Set Summary

Category	Instruction	Example
Arithmetic	add	`add  $1,$2,$3`
	subtract	`sub  $1,$2,$3`
	add immediate	`addi $1,$2,100`
	add unsigned	`addu $1,$2,$3`
	subtract unsigned	`subu $1,$2,$3`
	add immediate unsigned	`addiu $1,$2,100`
	multiply	`mul  $1,$2,$3`
Logical	and	`and  $1,$2,$3`
	or	`or   $1,$2,$3`
	and immediate	`andi $1,$2,100`
	or immediate	`ori  $1,$2,100`
	shift left logical	`sll  $1,$2,10`
	shift right logical	`srl  $1,$2,10`
Data transfer	load word	`lw $1, 100($2)`
	store word	`sw $1, 100($2)`
	move	`move $1,$2`
	load immediate	`li   $1,100`
	load upper immediate (high 16 bits)	`lui  $1,100`
Conditional branch	branch on equal	`beq  $1,$2,label`
	branch on not equal	`bne  $1,$2,label`
	set on less than	`slt  $1,$2,$3`
	set less than immediate	`slti $1,$2,100`
	set less than unsigned	`sltu $1,$2,$3`
	set less than immediate unsigned	`sltiu $1,$2,100`
Unconditional jump	jump	`j label`
	jump register	`jr $31`
	jump and link	`jal label`