CS 61CL (Clancy/Culler) **Exam 2**
Fall 2008

Read and fill in this page now.
Do NOT turn the page until you are told to do so.

Your name: _____

Your login name: _____

Your lab section day and time: _____

Your lab t.a.: _____

Name of the closest person on your right (possibly "aisle" or "wall") _____

Name of the closest person on your left (possibly "aisle" or "wall") _____

Problem 0	_____			Total:		_____	/60
Problem 1	_____	_____		Problem 5	_____		
	_____	_____		Problem 6	_____		
Problem 2	_____			Problem 7	_____	_____	
Problem 3	_____	_____		Problem 8	_____		
	_____	_____		Problem 9	_____	_____	
Problem 4	_____			Problem 10	_____		

This is an open-book test. You have approximately two hours to complete it. You may consult any books, notes, or other paper-based inanimate objects available to you. You may not use any electronic devices.

To avoid confusion, read the problems carefully. If you find it hard to understand a problem, ask us to explain it. If you have a question during the test, please come to the front or the side of the room to ask it.

This exam comprises 15% of the points on which your final grade will be based. Partial credit may be given for wrong answers. Your exam should contain eleven problems (numbered 0 through 10) on thirteen pages. Please write your answers in the spaces provided in the test; in particular, we will not grade anything on the back of an exam page unless we are clearly told on the front of the page to look there.

Some students are taking this exam late. Please do not talk to them, mail them information, or post anything about the exam to news groups or discussion forums until after Thursday.

Relax—this exam is not worth having heart failure about.

Problem 0 (2 points)

Put your login name on each page. Also make sure you have provided the information requested on the first page.

Problem 1 (10 points)

Part a

Complete the C function below, intended to return the Rs field of its MIPS machine language instruction argument.

```
unsigned int getRs (unsigned int inst) {
```

Part b

Write the same function in MIPS assembly language. Follow all relevant conventions for register use.

Problem 1, continued

Part c

Translate the following C code to MIPS assembly language. Assume that ch is stored in register $t0. You may use other $t registers.

```
if (ch >= 'a' && ch <= 'z') {
   ch = ch – 'a' + 'A';
}
```

Part d

Do the same for the following. Assume that ch is stored in $t0 and answer is to be stored in $v0. You may use other $t registers.

```
switch (ch) {
   case 'y':
      answer = 1;
      break;
   case 'n':
      answer = 0;
      break;
   default:
      answer = -1;
      break;
}
```

Problem 2 (4 points)

For each MIPS machine language instruction below, circle its assembly language counterpart.

8D28FFF8	`lw $9,-32($8)`	`lw $2,-32($8)`
	`lw $9,-8($8)`	`lw $2,-8($8)`
	`lw $9,-2($8)`	`lw $2,-2($8)`
	`lw $8,-32($9)`	`lw $8,-32($2)`
	`lw $8,-8($9)`	`lw $8,-8($2)`
	`lw $8,-2($9)`	`lw $8,-2($2)`
	`subu $9,$8,$31`	`subu $31,$9,$8`
01022020	`addi $8,$2,8224`	`add $8,$2,$4`
	`addi $2,$8,8224`	`add $4,$8,$2`
	`lb $8,8224($2)`	`add $2,$4,$8`
	`lb $2,8224($8)`	`add $2,$2,$0`

Problem 3 (4 points)

Most people count on their hands in unary, holding up as many fingers as the value they want to represent. Consider a Boolean circuit that converts a 3-bit binary number N in [0,5] to a 5-bit unary number U, with N 1 bits padded on the left by $5-N$ 0 bits. (For inputs larger than 5, it doesn't matter what it does.) Here is its truth table.

N_2	N_1	N_0		U_4	U_3	U_2	U_1	U_0
0	0	0		0	0	0	0	0
0	0	1		0	0	0	0	1
0	1	0		0	0	0	1	1
0	1	1		0	0	1	1	1
1	0	0		0	1	1	1	1
1	0	1		1	1	1	1	1

Part a

Give a circuit that, given N_2, N_1, and N_0 as inputs, outputs U_0.

Part b

Give a circuit that, given N_2, N_1, and N_0 as inputs, outputs U_4.

Part c

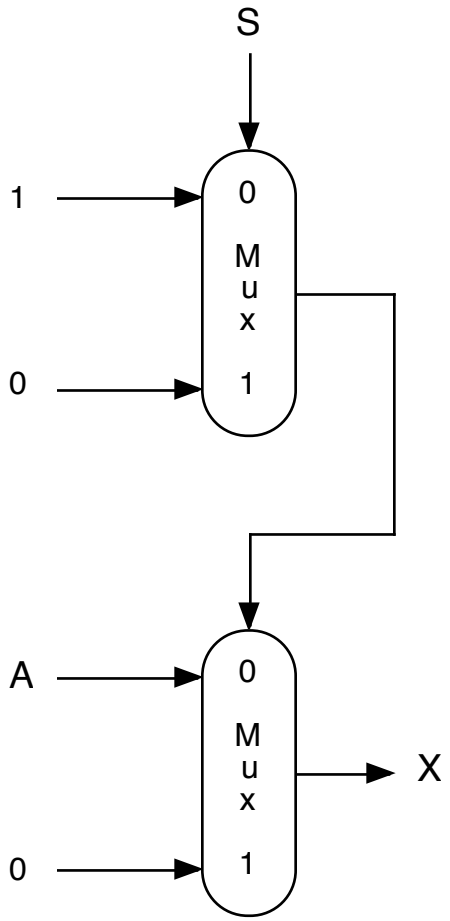
Give a sum-of-products expression for U_2 in terms of N_2, N_1, and N_0.

Part d

Simplify your answer to part c.

Problem 4 (4 points)

The circuit below consists of two multiplexors, some of whose inputs are "hard-wired" to 1 or 0. Give a simplified boolean expression for the output X in terms of the other inputs A and S. Show your work.



Problem 5 (6 points)

This problem involves setting up the stack for a call to snprintf.

Suppose that the data segment contains the following declarations:

```
chars:   .asciiz  "ABCDMN = "
more:    .asciiz  "p112A;"        # 'p' has ASCII code 112
format:  .asciiz  "%s %d %c"
buffer:  .space   200
```

Given below is the first few instructions of a call to snprintf, similar to what was provided in the file snprintf.s. Complete the sequence of assembly language statements that sets up a call to snprintf. Upon return from snprintf, the memory labeled by buffer should contain the string

```
"N = 112 ;"
```

Your code must use the format string and buffer specified above. Every argument you supply to snprintf should be retrieved from the data segment (the areas labeled by chars and more) via lw, la, or lb instructions.

```
_start:
        addi    $sp,$sp,-8      # reserve stack space for arguments
        la      $a0,buffer      # arg 0 <- buffer
        li      $a1,200         # arg 1 <- size of buffer
        la      $a2,format      # arg 2 <- format

# You fill in.
```

```
        jal     snprintf
```

Problem 6 (5 points)

Consider an assembly language translation of the following declarations.

```
struct node {
   int values[5];
   struct node * next;
};
struct node * lists[10];
```

Suppose that $s0 contains a pointer to lists[0].

Each assembly language segment shown below corresponds to a *single* C assignment statement. Supply that statement. Hint: draw diagrams.

`addi $t0,$s0,4` `sw $0,0($t0)`	
`lw $t0,8($s0)` `sw $t0,24($s0)`	

`lw $t0,20($s0)`
`lw $t0,20($t0)`
`sw $0,20($t0)`	

Problem 7 (6 points)

Part a

Provide (in hexadecimal) the IEEE floating point representation of 4.5 and –0.625, and indicate how you found them.

decimal	IEEE floating point
`4.5`	
`–0.625`	

Part b

Show in detail the steps involved in adding these two values.

Problem 8 (4 points)

In lab, you found the smallest value `x` for which `x = x+1`. Suppose we increased the number of bits in the exponent by 1 and corresponding reduced the number of bits in the fraction by 1. Would the smallest `x` for which `x = x+1` increase, stay the same, or decrease as a result of this change in format? Briefly defend your choice. If the desired `x` changes, indicate by how much.

Problem 9 (7 points)

Translate the following C function to MIPS assembly language. Follow all relevant register conventions.

```
int answer (char *prompt) {
   char ch;
   printf ("%s", prompt);
   ch = getchar ( );
   if (ch == 'y') {
      return 1;
   } else {
      return 0;
   }
}
```

Problem 10 (8 points)

The MIPS assembler, when run with the code on the left below as input, produced the machine code on the right.

Assembly language, .text section	Relocatable binary, .text section	
	Address	Contents

`# Argument is the number of bytes`
`# the caller wants to allocate.`
`# Address of the requested storage`
`# is returned, or 0 if request`
`# can't be satisfied.`		

`stackalloc:`		
`lw $v0,nextfree`	00	3c010000
	04	8c220064
`add $t0,$a0,$v0`	08	00824020
`la $t1,nextfree`	0c	3c010000
	10	34290064
`ble $t0,$t1,ok`	14	0128082a
	18	10200003
`add $v0,$0,$0`	1c	00001020
`j return`	20	0800000c
`ok:`		
`sw $t0,nextfree`	24	3c010000
	28	ac280064
`return:`		
`jr $ra`	2c	03e00008
Assembly language, .data section	Relocatable binary, .data section	
`stg:`	00	00000000
`.space 100`	…	...
	60	00000000
`nextfree:`		
`.word stg`	64	00000000

Part a

In the "Relocatable binary: Contents" column, circle each instruction that contributes an entry to the relocation table.

Problem 10, continued

Part b

Here is the code from the previous page. Assume that the .text segment is loaded starting at location 0x00400000 and the .data segment is loaded starting at 0x10010000. Make whatever fixes to the .text and .data segments that are necessary to turn relocatable addresses into absolute addresses.

.text segment		.data segment	
Address	Contents	Address	Contents
00	3c010000	00	00000000
04	8c220064	…	...
08	00824020	60	00000000
0c	3c010000	64	00000000
10	34290064		
14	0128082a		
18	10200003		
1c	00001020		
20	0800000c		
24	3c010000		
28	ac280064		
2c	03e00008		