CS 61C Summer 2015
Guerrilla Section 1: Number Representation & MIPS

Question 0: *Silly Rabbit, Trits Are for Kids*

A new memory technology with three distinct states is exploding into the technology industry! Let's see if we can't develop some new number representations to take advantage of this new development.

(a) First, define a rule, analogous to what we use for binary numerals, for determining the unsigned value of a ternary numeral d_n, d_{n-1} . . . d_0, and use this rule to convert 2102_3 into decimal:

unsigned(d_n, d_{n-1} . . . d_0) = _____

unsigned(2102_3) = _____

(b) Next we'd like to define an analogue to two's complement for ternary numerals, which we'll call three's complement. Three's complement numbers should be as evenly distributed between positive and negative as possible (favor negative if necessary), should have a zero at 0_3, and should increase in value when incremented as an unsigned value (except in the case of overflow). Define a rule for negating a three's complement number.

(c) What is the most positive possible three's complement 8-trit number? Using this result, specify a rule for determining if a three's complement number is positive or negative.

(d) There are two different two's complement numbers who are their own inverse. Specify these numbers.

(e) Which numbers in three's complement are their own inverse?

(f) What arithmetic operation is a shift left logical equivalent to with three's complement numbers?

(g) What arithmetic operation is a shift right arithmetic equivalent to with two's complement numbers?

Question 1: *Number Representation*

1) Convert the following 8-bit two's complement numbers from hexadecimal to decimal:

0x80 = _____
0xF4 = _____
0x0E = _____

2) What's the biggest change to the PC as the result of a jump on a 32-bit MIPS system?

3) Assume that the most significant bit (MSB) of x is a 0. We store the result of flipping x's bits into y. Interpreted in the following number representations, how large is the magnitude of y relative to the magnitude of x? Circle ONE choice per row.

Unsigned	|y| < |x|	|y| = |x|	|y| >|x|	Can't Tell
One's Complement	|y| < |x|	|y| = |x|	|y| >|x|	Can't Tell
Two's Complement	|y| < |x|	|y| = |x|	|y| >|x|	Can't Tell
Sign and Magnitude	|y| < |x|	|y| = |x|	|y| >|x|	Can't Tell

Question 2: *MIPS*

a) Write the MAL MIPS function reverse_str(char *string, int string_length), that can reverse strings (with an even length) in-place. The MIPS should be non-delayed branch, and you will probably not use all the lines. In your solution, register $a0 should signify the parameter char *string and register $a1 should signify the parameter int string length.

```
reverse_str:  beq $a1 $0 done
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              _____________________________
              j reverse_str
done:             jr $ra
```

b) Complete the code below, using at most two TAL MIPS instructions, so that the function returns false if $a0 contains an R-type instruction and true otherwise.

```
NotRType: _____________________________
          _____________________________
          _____________________________
```

Question 3: *It's a Bloody MIPStery*

(HINT for students in the Guerilla Section: for part b, try running through your MIPS code with a simple example based off of your response for part a)

Consider the following MIPS function mystery:

```
1  mystery :
2     beq $a1 , $0 , L3
3     sll $a1 , $a1 , 2
4     addu $a1 , $a0 , $a1
5     move $t8 , $a0
6  L1 :
7     addiu $t8 , $t8 , 4
8     beq $t8 , $a1 , L3
9     move $t9 , $t8
10 L2 :
11    beq $t9 , $a0 , L1
12    lw $t0 , -4( $t9 )
13    lw $t1 , 0( $t9 )
14    slt $t2 , $t1 , $t0
15    beq $t2 , $0 , L1
16    sw $t1 , -4( $t9 )
17    sw $t0 , 0( $t9 )
18    addiu $t9 , $t9 , -4
19    # PRINTF HERE
20    j L2
21 L3 :
22    jr $ra
```

(a) What would the type signature or function header of mystery be in a corresponding C program?

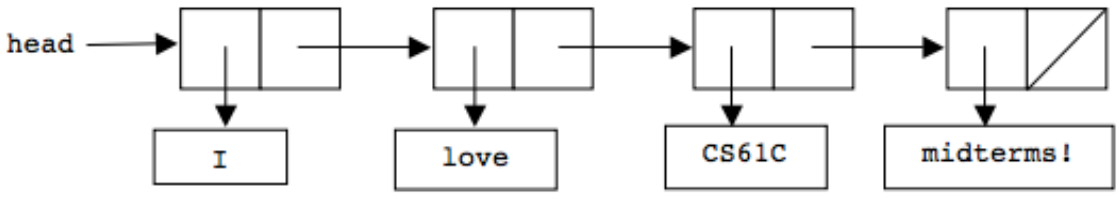
(b) Explain what mystery does. You may not know the specific type, but you should be able to respond with a single word.

(c) Suppose we wanted to modify the code to include an invocation of the library function printf at the line marked #PRINTF HERE, but not modify any other lines of code in the function. List the registers which would need to be saved onto the stack before calling printf and restored afterwards. If no registers need to be saved/restored, write "none".

(d) Suppose we wanted to modify the code so that the type of its first argument was (char*). Specify, in ascending order, the line numbers which would need to be changed to do so.

Question 4: *"free at last, thank gosh we are free at last…"*

We wish to free a linked list of strings (example below) whose nodes are made up of this struct. Complete the code below; we have started you off with some filled in. You may use fewer lines, but do not add any.

```
FreeLL:   beq ____, _____, NULL_CASE
          ____________________________
          ____________________________
          ____________________________
          ____________________________
          ____________________________
          jal FreeLL
          lw $a0 0($sp)
          ____________________________
          ____________________________
          ____________________________
          ____________________________
          jal free
          ____________________________
          ____________________________
          ____________________________
NULL_CASE:    jr $ra
```

```
// Assume compiler packs tightly
struct node {
       char *string;
       struct node *next;
};

void FreeLL(struct node *ptr) {
       if (ptr == NULL) return;
       else {
              FreeLL(ptr->next);
              free(ptr->string);
              free(ptr);
       }
}
```