University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Spring 2014 Instructor: Dr. Dan Garcia 2014-03-12

After the exam, indicate on the line above where you fall in the emotion spectrum between "sad" & "smiley"...

Last Name	**Answer Key**
First Name	
Student ID Number	
Login	**cs61c-**
Login First Letter (please circle)	**a b c d e f g h i j k l m** **n o p q r s t u v w x y z**
Login Second Letter (please circle)	**a b c d e f g h i j k l m** **n o p q r s t u v w x y z**
*The name of your **LAB** TA (please circle)*	Alan | Jeffrey | Kevin | Roger | Sagar | Shreyas | Sung Roa | William
Name of the person to your Left	
Name of the person to your Right	
All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet. ***(please sign)***	

Instructions (Read Me!)

- Don't Panic!
- This booklet contains 6 numbered pages including the cover page. Put all answers on these pages; don't hand in any stray pieces of paper.
- Please turn off all pagers, cell phones & beepers. Remove all hats & headphones. Place your backpacks, laptops and jackets at the front. Sit in every other seat. Nothing may be placed in the "no fly zone" spare seat/desk between students. No computing devices allowed.

Question	**1**	**2**	**3**	**4**	**Total**
Minutes	48	24	24	24	120
Points	**30**	**15**	**15**	**15**	**75**
Score	**30**	**15**	**15**	**15**	**75**

Question 1: ***What's that funky smell?! Oh yeah, it's potpourri...*** (48 min, 30 pts)

a) The *unsigned distance* between two bit patterns is the absolute value of the difference of their values, interpreted as unsigned numbers. Rank the following according to the unsigned distance between -1 and 0 (+0 if a representation has multiple zeros) in that representation. You should assign a rank of 1 to the representation with the ***smallest*** unsigned distance between -1 and 0.

`int64_t`	5
64-bit One's Complement	4
64-bit Sign and Magnitude	2
64-bit Bias notation	1
`double`	3

Show all your work here!

b) As defined in IEEE 754-2008 standard, half-precision floating point (FP) is a 16-bit FP representation: 1 sign bit, 5 exponent bits, and 10 significand bits. The exponent bias of 15. What is the binary representation of the *smallest half-precision float which is strictly larger than 1*? What is its value? Leave your answer in terms of powers of two.

0b`0 01111 0000000001` = $1 + 2^{-10}$

Show all your work here!

c) How would J-type instructions be affected (in terms of their "reach") if we relaxed the requirement that instructions be placed on word boundaries, and instead required them to be placed on *half-word* boundaries.

`The range over which we can jump would be cut in half.`

d) Building on the idea from the previous question, give a minor tweak to the MIPS ISA to allow us to use *true absolute addressing* (i.e., maximal "reach") for all J-type instructions.

`Only allow jumps to addresses which are multiples of 2^6.`

e) Assume a request for 100 Bytes on either the **Stack** and **Heap** would succeed.
In the worst case, *how many clock cycles* would it take to **allocate** 100 Bytes in each area *& why*?
("How many" should be answered with one of: {1, tens, thousands+})

Stack: `1` because `it's just addiu $sp $sp -100`

Heap: `Thousands+` because `the freelist might contain lots of slivers to check`

Question 1: ***What's that funky smell?! Oh yeah, it's potpourri... (continued)*** (48 min, 30 pts)

f) You have a program that can achieve almost a 20x speedup with millions of processors, so what is the percent of the parallel portion of its code? ___95___

Show all your work here!

g) Suppose the assembler knew the file line numbers of all labels before it began its first pass over a file, and that every line in the file contains an instruction. Then the assembler would need __2__ pass(es) to translate a MAL file, and __1__ pass(es) to translate a TAL file. These numbers differ because of _____pseudoinstructions_____ (write n/a if they don't differ).

h) Complete the code below, using *only one TAL instruction*, so that it returns true iff `$a0` is an I-type instruction or a J-type instruction, and then translate the instruction into binary then hexadecimal.

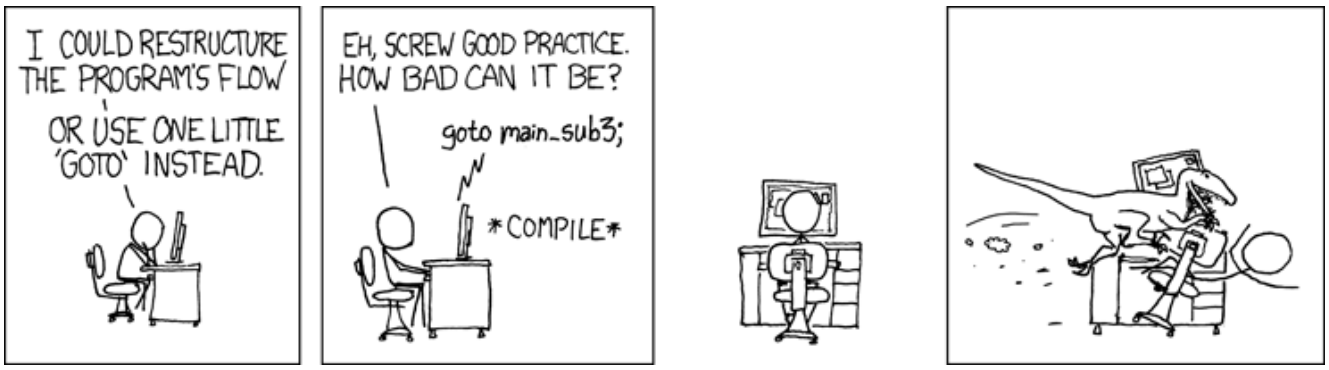
```
                srl $v0 $a0 26    0000|00 00|000 0|0100 0001|0 110|10 00|0010  0004 1682
IJ-instr:  _____________________ ➔ 0b_____________________________________ ➔ 0x__________
           jr $ra
```

Show all your work here!

i) What is one thing Google did to increase their Power Usage Efficiency (PUE)?

Careful airflow handling, elevated cold aisle temp, use free cooling, per-server ups

_____.



Question 2: *Running in circles* (24 min, 15 pts)

a) Recall the exercise `ll_cycle` from lab 2, in which we checked if a linked list contained a cycle using the *tortoise and hare* algorithm. We've provided you with most of a simple *recursive* implementation in the space below. Fill in the base cases.

```
typedef struct node {
      int value;
      struct node *next;
} node;
```

```
int ll_has_cycle(node *ptr) {
   if (!ptr)
         return 0;
   return has_cycle(ptr, ptr->next);
}

int has_cycle(node *tortoise, node *hare) {
                      hare == tortoise
    if (_____________________________________________)
        return 1;
                   !hare || !hare->next
    if (_____________________________________________)
        return 0;

    return has_cycle(tortoise->next, hare->next->next)
}
```

b) Now that you've warmed up on the C version of this code, let's convert `has_cycle` into ***recursive*** MAL MIPS. Assume that the fields of the `structs` are not permuted from the `struct` definition. You may use fewer lines than we provide you, but do not add any more than the space provided.

```
# $a0 contains the pointer to the tortoise, $a1 contains the pointer to the hare.

has_cycle:  li $v0 1

            beq $a0 $a1 done

            li $v0 0
                  $a1     $0
            beq _______ ______ done
            lw $a1 4($a1)
            ___________________________
                  $a1     $0
            beq _______ ______ done
            lw $a0 4($a0)
            ___________________________
            lw $a1 4($a1)
            ___________________________
                   $sp   $sp   -4
            addiu _____ _____ _____     ← circled, changed to nop
            sw $ra 0($sp)
            ___________________________  ← circled, changed to nop
            jal has_cycle
            ___________________________  ← change to j has_cycle
            lw $ra 0($sp)
            ___________________________
                   $sp   $sp   4
            addiu _____ _____ _____

done:       jr $ra
```

c) You want to change this to be an *iterative* MIPS solution, but *you want to change the fewest lines* you can. Circle those lines you would change to make to the program work *iteratively*.

Question 3: ***Our band is called 1023MiB... We haven't had any gigs yet.*** (24 min, 15 pts)

We have a standard 32-bit byte-addressed MIPS machine with a 1KiB direct-mapped write-back cache and 16B block size.

a) How many bits are used for the Tag? ____22____ …Index? ____6____ …Offset? ____4____

Consider the following C code, and answer the questions below. `d` and `s` are pointers to 8-bit unsigned integer arrays, of the same size (a multiple of the cache size) that are aligned on 16-byte boundaries. The arrays contain only one `0x00`, in the last byte. `d` and `s` are not necessarily distinct.

```
void our_strcpy(uint8_t *d, uint8_t *s) {
    char c;
    do {
         c = *s;
        *d = c;
        s++; d++;
    } while (c);
}
```

b) What is the *lowest* possible cache hit rate for `our_strcpy`? _____0_____

c) What *types* of misses are there? Compulsory and conflict

d) What is the *smallest possible value* of `(d - s)` that would get this hit rate? _____1 KiB_____

e) What is the *highest* possible cache hit rate for `our_strcpy`? ____31/32_____

f) What is one possible value of `(d - s)` where we would get this hit rate? _____0_____

2 misses per block * 2^6 blocks/cache * 2^13 caches = 2^20 misses

g) If we ran `our_strcpy` with a 4-way set-associative LRU cache, and the size of both `d` and `s` is 8MiB, what is the *most # of misses* possible? _____1 Mebi_____

Show all your work here!

Question 4: *A bad case of Not Invented Here Syndrome...* (24 min, 15 pts)

a) A colleague of yours has implemented some homebrew C99 string manipulation functions, while steadfastly refusing to use any standard libraries, but they're buggy! We've marked each potentially problematic line with `// <number>`. Your job is to fill in a correct replacement line in the corresponding row of the following table, *or write 'OK' if there is nothing wrong*. DO NOT LEAVE ANY FIELDS BLANK, or we will assume you just didn't get to this part of the exam.

Line number	Replacement Code
1	`for(char c = *s; (c = *s) != '\0'; s++) {`
2	`*s += 'a' – 'A';`
3	`OK`
4	`OK`
5	`odds += numbers[i] & 1` … `odds += *(numbers+i) & 1`

```
/** Converts the string S to lowercase */
void string_to_lowercase(char *s) {
  for(char c = *s; c != '\0'; s++) {          // 1
    if(c >= 'A' && c <= 'Z') {
      s += 'a' - 'A';                         // 2
    }
  }
}

/** Returns the number of bytes in S before, but not counting, the null terminator. */
size_t string_length(char *s) {
  char *s2 = s;
  while(*s2++);                               // 3
  return s2 - s - 1;                          // 4
}

/** Return the number of odd numbers in a number array */
uint32_t number_odds(uint32_t *numbers, uint32_t size) {
  uint32_t odds = 0;
  for (uint32_t i = 0; i < size; i++)
    odds += *numbers+i &&  1;                 // 5
  return odds;
}
```