CS61C Fall 2012 Midterm

Your Name: _____ **SID:** _____

Your Lab Time (Circle): W12-2 W1-3 W2-4 W3-5 W4-6
W6-8 W8-10 W9-11 T8-10 T10-12
T12-2 T2-4 T4-6 T6-8 T8-10 F9-11

This exam is closed book, closed notes, open two-sided crib sheet, and no calculator is needed. It is worth **60** points and represents 10% of your course grade. It contains **8** questions on **nine** numbered pages, including the cover page. The MIPS Green Card is appended to the examination at the end. Write all of your work and answers on these pages. Do not remove the staple and don't hand in stray pieces of paper.

Question 0: You will receive 1 point for properly filling out this page as well your login on every page of the exam.

Question	Points	Score
0	1	
1	4	
2	5	
3	5	
4	5	
5	10	
6	5	
7	10	
8	15	
Total	60	

All the work included on these sheets is my own and my own alone. 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.

Signature: _____

Question 1: *Potpourri (0.5 points per answer, 4 points total)*

a) **Circle the single correct answer or fill in the blanks**

i. The dominant form of parallelism in Graphics Processing Unit (GPU) is

SISD MISD SIMD MIMD

ii. The dominant form of parallelism in Warehouse-Scale Computers is

SISD MISD SIMD MIMD

which in class we have used via MapReduce.

iii. Git will add all if you add the -A flag to the 'git add' command.

iv. If you run a debugger, it will pause execution if you set a breakpoint.

v. The maximum number of instructions you can jump past in a j-type instruction is $2^{26} - 1$(or -2).

vi. The maximum number of bytes you can branch past is 2^{17}(or -4).

b) **Truth or False**

i. Most of the power of a Warehouse-Scale Computer (WSC) is consumed by servers and networking equipment.

True False

ii. Increasing the utilization of information technology equipment increases the WSC's Power Usage Effectiveness (PUE)

True False

Question 2: *Pointers (1 point per answer, 5 points total)*

Given below is a main function that prints five statements. What are the five things that are printed? If it is undefined what would be printed, please write UNDEFINED for the print.
Assume that the numbers are stored in big-endian format!
Assume that ints are 32 bits and chars are 8 bits!

```
int main(int argc, char** argv) {
      unsigned int data[] = {0x01234567, 0x89ABCDEF};
      unsigned char* charFront = (char *) data;
      unsigned char* charMid = (char *) &(data[1]);
      unsigned char* charEnd = (char *) &(*(data+2));

      // %x means print the variable in hex.
      printf("The first print: %x\n", (int) *(charFront+1));
      printf("The second print: %x\n", (int) *(charFront+4));
      printf("The third print: %x\n", (int) charMid[1]);
      printf("The fourth print: %x\n", (int) *(charMid-4));
      printf("The fifth print: %x\n", (int) charEnd[-1]);
}
```

The printed statements are:

a) `The first print: 23`

b) `The second print: 89`

c) `The third print: AB`

d) `The fourth print: 01`

e) `The fifth print: EF`

Half credit: Solving the correct 8 bits, then extending those in some way.
Ex: *FFFFFF***EF EF***000000*
Half credit: Solving in little-endian instead of big endian
Half credit: Writing correct answer OR wrong answer
(As in explicitly had two answers and used "or")
Half credit: Writing correct answer in binary

Little endian answer:
a) 45
b) ef
c) cd
d) 67
e) 89

Question 3: *Project (1 point for first two answers, 1.5 for the last two, 5 points total)*

a) Fill in the blank with the Hadoop interface input/output type used by the mapper and reducer implementation:

writable

b) Fill in the blank with what happens if the --auto-shutdown=n flag is not set:

Processes could accidentally run indefinitely.

c) You have booted a large EC2 instance using 10 machines. After you run for 2 hours and 6 minutes, a bug forces you to terminate the instance. So, you then reboot a large EC2 instance using 20 machines for 1 hour and 12 minutes to make up for lost time. Assuming a large instance has a rate of 1 dollar per hour per machine, how much did you spend?

70 dollars

d) You have to solve a problem using Amazon EC2 servers. You know the server will finish the problem in an hour using 10 machines, but the deadline for your solution is just over 1 minute away. You attempt to solve the problem quickly by running an instance with 600 machines.
However, even though the cluster magically booted up instantaneously, you were late turning in your project and wasted a lot of money in the endeavor. This scenario indicates your solution lacked which kind of scaling? Circle one:

Weak scaling Strong scaling

Question 4: *Numbers (1 point per answer, 5 points total)*

Given the indicated number representation, fill in the blank with exactly one of LESS THAN (<), GREATER THAN (<), or EQUAL TO (=). E.g., 1 < 2 and 1 > 0!

a) Unsigned 32-bit numbers

A. $0111\ 0000\ 1111\ 0101\ 0111\ 0000\ 1111\ 0101_{two}$
B. $0111\ 0000\ 1011\ 0101\ 0111\ 0000\ 1111\ 0101_{two}$

A ___ **>** ___ B

b) One's-Complement 32-bit numbers

A. $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two}$
B. $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two}$

A ___ **=** ___ B

c) Two's-Complement 32-bit numbers

A. $1111\ 0000\ 1111\ 0101\ 0111\ 0000\ 1111\ 0101_{two}$
B. $1111\ 0000\ 1011\ 0101\ 0111\ 0000\ 1111\ 0101_{two}$

A ___ **>** ___ B

d) IEEE Standard Single-Precision Floating-Point Numbers

A. $1111\ 1111\ 0111\ 0101\ 0111\ 0000\ 1111\ 0101_{two}$
B. $1111\ 1110\ 0111\ 0101\ 0111\ 0000\ 1111\ 0101_{two}$

A ___ **<** ___ B

e) IEEE Standard Single-Precision Floating-Point Numbers

A. $1000\ 0000\ 1111\ 0101\ 0111\ 0000\ 1111\ 0101_{two}$
B. $1000\ 0001\ 0111\ 0101\ 0111\ 0000\ 1111\ 0101_{two}$

A ___ **>** ___ B

Question 5: *C->Assembly->MIPS (1 point per instruction fill-in, 10 points total)*

Given below is a C code fragment from a binary search tree routine (don't worry if you don't know exactly what that is). It returns the array index of an integer that matches the given search element, and for simplicity we assume the element is always present. Convert the C code into the minimum number of MIPS instructions necessary to implement the C functionality. Answer by filling in the blanks with MIPS assembly language in the code template below. **Note that the minimum code necessary may use fewer blanks than the number shown.**

```
C:
int helperPBST (int *arr, int pos, int elem) {
      int temp = pos - 1;
      if (elem == *(arr + temp)) {
            return pos;
      }
      else if (elem > *(arr + temp)) {
            return helperPBST(arr, pos*2+1, elem);
      }
      else {
            return helperPBST(arr, pos*2, elem);
      }
}

MIPS:
# $a0 is arr
# $a1 is pos
# $a2 is elem
pbst:
addi $t0, $a1, -1
sll    $t0, $t0, 2      #$t0 is being used as a pointer offset
add    $t0, $t0, $a0    #calculate arr + temp
lw     $t1, 0($t0)      #dereference (arr + temp)
beq $a2, $t1, success
slt $t3, $a2, $t1
beq $t3, $0, right
    sll $a1, $a1, 1     #pos = pos*2
    j pbst              #recurse. Use j instead of jal because
                  #this is tail recursive

_____________________
right:
    sll $a1, $a1, 1     #pos = pos*2
    addiu $a1, $a1, 1   #pos = pos + 1
    j pbst              #recurse.
success:
    addiu $v0, $a1, 0   #return pos
jr $ra
```

This problem was graded in blocks. After the first mistake in a block the remainder of a block was marked as incorrect. For example, many people failed to shift $t0 to the left by 2, and so all of the first three blanks were marked as wrong. Similar logic was applied to blanks 4-6, 7-9, and 10.

There were two situations in which we showed leniency. If a student mixed up which part of the MIPS code corresponded to the else if and which corresponded to the else then 3 points were deducted, as opposed to 6. Additionally, if a student performed the pos*2 multiplication in the first sll slot, but managed to turn in a solution that would other wise be correct they were only docked for the first three blanks.

Question 6: *Memory Hierarchy (1 point for each letter, 5 points total)*

a) Your are given two programs that read in n files (where n is very large), perform some small operation on each file (like counting the occurrence of a word, etc.), and then generate some output files. Listed below are the high-level pseudo-code implementations of each program:

Program A:	**Program B:**
Read file1 *Do-something* *Output out1* *Read file2* *Do-something* *Output out2* … (and so on)	*Read file1* *Read file2 …* (read all the files) *Do-something* *Do-something …* (do something to all the files) *Output out1* *Output out2 …* (output all the results)

i. Which program would you expect to better utilize a data cache?
(circle one) A B

ii. Which program would you expect to better utilize an instruction cache?
(circle one) A B

b) Rank the following types of memories in terms of average memory access time:

Disk storage, Register, Main Memory, L1 Cache, L2 Cache

Fastest *Register , L1 Cache, L2 Cache, Main Memory, Disk storage* Slowest

c) The ratio of L1 cache access time to register access time
(time_L1_cache / time_register) is closest to (circle one):

0.1 1 10 100 10,000

d) The ratio of main memory (DRAM) access time to register access time
(time_main_memory / time_register) is closest to (circle one):

10 100 1,000 10,000 1,000,000

e) The ratio of disk storage access time to register access time
(time_disk / time_register) is closest to (circle one):

10 100 1,000 10,000 1,000,000

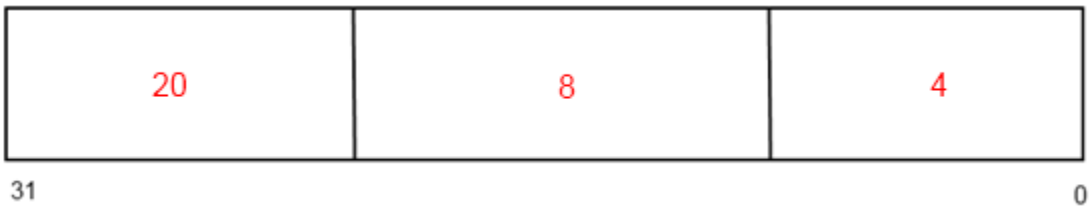
Question 7: *Caches (2 points for each letter, 10 points)*

a) Given a data cache with 4 blocks each of 4x32-bit words, what would the hit rate be for the cache when the following array accesses are made? Assume the cache is initially empty, that the processor operates on 32-bit integers and has a byte-addressed memory, and that the first element of array a is at address 0.

```
int a[20];
a[0];
a[4];
a[12];
a[1];
a[0];
a[5];
a[13];
a[17];
a[0];
a[1];
```

Hit Rate = _____50_____%

b) For a different data cache, also byte-addressed with 32-bit addresses, with a total data capacity of 4KB and 4x32b word blocks, label the indicated fields as tag, index, and offset, and indicate the number of bits for each.



In addition to data storage, each block of cache requires some additional state storage to enable the hardware to function correctly.

c) If this cache were write-through, each cache block would require ____149_____ bits of storage total.

d) If this cache were write-back, each cache block would require ____150____ bits of storage total.

e) In an attempt to improve cache performance, you try a new strategy. You think that because your L1 cache has a hit rate of 90%, you could improve the AMAT by randomly going straight to memory 10% of the time instead of first checking the L1 cache. Assuming the cache has a hit time of 5 cycles, and memory has an access time of 100 cycles, calculate the AMAT for this "improved" system.

Average Memory Access Time = _____23.5_____ cycles

Question 7:

a) 2 points : 2 if correct, 0 if not.

b) 2 points : 2 for all correct,
1 for at least one field correct or correct fields but wrong order

c) 2 points : 2 for correct answer or correct answer using tag size from (b),
1 for correct answer * 2^8,
0 otherwise

d) 2 points : same as c

e) 2 points : 2 for correct answer,
1 for correct formula but simple math mistake (ie .1*100 = 1 or .9*15 = 12)

Question 8: *MIPS ISA (3 points for each blank, 15 points total)*

Given below is a mystery function. Determine what the values in memory, starting at address 0x10010000, will be when execution reaches line #7. Execution starts from line 0 ("main"), and we have shown the values in memory at several other points in execution to aid you.

```
0  main:
1      lui $a0 0x1001
2      ori $a0 0x0000
3      la $a1, f1
4      jal mystery
5      la $a1, f2
6      jal mystery
7      la $a1, f3
8      jal mystery
9      addi $v0 $0 10
10     syscall
11 mystery:
12     addi $sp, $sp, -12
13     sw $ra, 8($sp)
14     add $s0, $a0, $0
15     add $t0, $a0, $0
16     add $t1, $a1, $0
17 loop:
18     lw $a0, 0($t0)
19     beq $a0, $0, end
20     sw $t0, 4($sp)
21     sw $t1, 0($sp)
22     jalr $t1
23     lw $t1, 0($sp)
24     lw $t0, 4($sp)
25     sw $v0, 0($t0)
26     addi $t0, $t0, 4
27     j loop
28 end:
29     add $a0 $s0 $0
30     lw $ra 8($sp)
31     addi $sp, $sp, 12
32     jr $ra
33 f1:
34     addi $v0, $a0,1
35     jr $ra
36 f2:
37     addi $t0 $0 2
38     loop2: slt $t1 $a0 $t0
39     bne $t1 $0 end2
40     sub $a0 $a0 $t0
41     j loop2
42 end2: add $v0 $0 $a0
43     jr $ra
44 f3:
45     sll $v0, $a0, 1
46     jr $ra
```

Line #	0x1001 0000	0x1001 0004	0x1001 0008	0x1001 000c	0x1001 0010
0	4	8	11	14	0
5	5	9	12	15	0
7	_1_	_1_	_0_	_1_	_0_
9	2	2	0	1	0

There were a lot of questions about the fourth element of line 9, and that is a one instead of two, because of the beq check at line 19.