CS61C Fall 2013 Midterm Instructions

This exam is closed book, closed notes, open two-sided crib sheet. Please put away all phones and calculators -- you won't need them. The exam is worth **50** points and represents 10% of your course grade. It contains **50** multiple choice questions on **14** numbered pages, including the cover page. One and only one answer is correct for each question. There is no penalty for incorrect answers. The MIPS Green Card is appended at the end. Be sure to write your name, course login, lab section, and lab section meeting time on the separately distributed answer form.

Don't cheat by looking at other students' answer sheets. There is no guarantee they know what they are doing either!

PSSST! WHAT'S 7+6?
THREE HUNDRED BILLION GAZILLION.
OH, THANKS FOR THE BIG HELP!
THAT'S A THREE, FOLLOWED BY 85 ZEROES.
AH! I KNEW THAT.

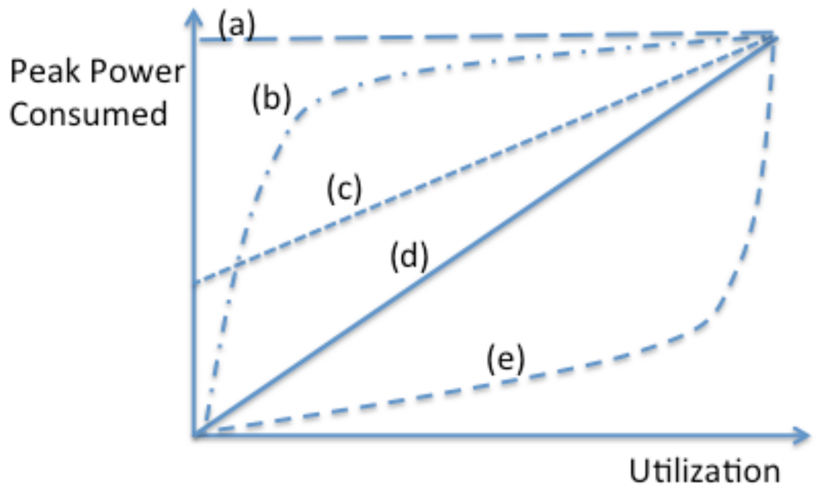
<u>Section I</u>: *Cloud Computing and Warehouse Scale Computers*

1. Consider what happens to Power Usage Effectiveness (PUE) when the total energy consumed by a datacenter stays the same but its IT equipment becomes more energy efficient by cutting its energy consumed in half? Which answer best describes the effect on PUE?
   (a) PUE is halved
   (b) PUE doubles
   (c) PUE stays the same
   (d) PUE approaches 1
   (e) PUE approaches 0

2. Consider the following elements of the datacenter memory hierarchy: memory and disk on a local processing node, a node in the same rack, or a node in a different rack. Which one of the following statements is false?
   (a) Local disk is about 1000 times slower than local memory.
   (b) Memory on another processor in another rack is three times slower than memory on another processor in the same rack.
   (c) Latency to another disk in the same rack is 10% slower than latency to a local disk.
   (d) Bandwidth to memory in the same rack is 10x that of memory in another rack.
   (e) Local disk bandwidth is 10x the bandwidth to a disk on another node in the same rack.

3. The following strategies may affect datacenter performance and availability. Which one is true?
   (a) Data replication improves availability but not performance.
   (b) Partitioning data into smaller fragments and distributing them across numerous machines improves performance and availability.
   (c) Load balancing of requests across numerous servers only improves availability.
   (d) It is not difficult to distinguish between slow servers and broken servers in the datacenter.
   (e) Compression can improve performance and availability.

For the next two questions, consider the following figure, showing five possible relationships between processor utilization (x-axis) and peak power consumed (y-axis):



4. Which line best describes the current state of processor power consumption as a function of utilization?
5. Which line represents the closest to the ideal relationship between utilization and peak power?

<u>Section II:</u> *C Programming*

6. What is the value of **s** after the following code?

```
unsigned int t[] = {0,1,2,3,4,5,6,7};
unsigned int s = sizeof(t);
```

(a) The length of the t array.
(b) The number of bytes in the t array.
(c) The number of bytes in one unsigned int.
(d) The number of bytes in an unsigned int pointer.
(e) Nothing; you cannot find the size of an array.

7. The **%s** format specifier takes in a **char*** and prints until it finds a null character. What do the following two lines of code print?

```
char *s = "uncharacteristic";
printf("%s",s+s[7]-s[6]);
```

(a) "uncharacteristic"
(b) "ncharacteristic"
(c) "characteristic"
(d) "haracteristic"
(e) Nothing; the address is not valid

8. What does the following method do?

```
char * bizarre(char *f,char y) {
   char *h = f;
   for (h=f; *h!=y&&*h;)
       h++;
       if(*h) {
         *h = 0;
          h++;
       }
   return h;
 }
```

(a) It returns a pointer to the first location of y in f.
(b) It splits f at the first occurrence of y and then returns a pointer for the remaining string.
(c) It finds the last location of y in f, zeros out that location, and then returns a pointer to the next location.
(d) It zeros out most of f until it finds y and then returns a pointer to the location of y in f.
(e) Nothing; it cannot be compiled.

9. What does this code print?

```
unsigned char a = 0b01110110;
char b = 0b10011010;
b >>= 1;
b ^= 0b00001101;
unsigned char c =  a & b;
printf("0x%02x",c);
```

(a) `0x40`
(b) `0x36`
(c) `0x44`
(d) `0x01`
(e) `0x00`

10. What is the FIRST line wrong with this code?

```
Line    Code
0       typedef char * yarn;
1       char * get_word() {
2          return "secret";
3       }
4       void print_word(yarn thread) {
5          printf("%s\n",thread);
6       }
7       int main() {
8          print_word(get_word());
9          return '\0';
10      }
```

(a) Line 0: A typedef is being used on a pointer.
(b) Line 2: It returns a pointer to local data that will go out of scope.
(c) Line 8: A char* is given where a yarn was expected.
(d) Line 9: A null terminator is returned instead of an int.
(e) Nothing is wrong with this code.

<u>**Section III**</u>: *C-to-MIPS*

A Stanford student has attempted to convert some C code into MIPS. The original C code was:

```
int add_all_the_nums(int* first_num, int num_count){
  int sum = 0;
  for(int i=0; i < num_count; i++){
      sum += first_num[i];
  }
  return sum;
}
```

The student's solution was:

```
Line    Code
        ADD_ALL_NUMS:
1         add  $t0,$0,$0

        LOOP:
2         beq  $a1,$0,END
3         lw   $t1,0($a0)
4         add  $t0,$t0,$t1
5         addi $a1,$a1,-1
6         addi $a0,$a0,1
7         j    LOOP

        END:
8         add  $v0,$t0,$0
9         jr $ra
```

As you can probably guess, this MIPS code has problems.

11. Which of the following changes will fix a problem with the code, assuming that the input is always valid?

    (a) change line 3 to `lw   $t1,4($a0)`
    (b) change line 7 to `jal  LOOP`
    (c) change line 6 to `addi $a0,$a0,4`
    (d) change line 8 to `or   $v0,$t0,$0`
    (e) change line 5 to `slt  $a1,$a0,$a1`

    For the following two questions, we have partially encoded the instruction at line 2, **`beq $a1,$0,END`** into hex:

opcode	rs	rt	immediate
000100	?????	?????	???? ???? ???? ????

12. What should go in the **`rs`** field?
    (a) 00101
    (b) 01001
    (c) 00100
    (d) 00001
    (e) 00000

13. What should go in the **`immediate`** field (assuming no delayed branching)?
    (a) 1111 1111 1111 1011
    (b) 0000 0000 0000 1100
    (c) 0000 0000 0000 0110
    (d) 0000 0000 0000 0101
    (e) 0000 0000 0001 0100

14. Assume that the code was loaded into the 0x0 segment (the upper four bits of PC are all 0's), and that line 7, **`j LOOP`**, was encoded as follows:

opcode	address
000010	0b 000 0010 000 0000 0000 0000 0010 (0x100002)

What is the address of line 6, **`addi $a0,$a0,1`**?
(a) 0x00100002
(b) 0x00400008
(c) 0x0040000C
(d) 0x00400018
(e) 0x00100012

15. In line 5, **`addi $a1,$a1,-1`** there is a negative immediate. Immediates are embedded as 16-bit numbers, but this instruction is trying to add it to a 32-bit number in **`$a1`**. What happens to the 16-bit immediate during the execution of this instruction?
(a) 16 0's get added in front of it.
(b) It gets shifted to the left by 16 bits.
(c) 16 1's get added in front of it.
(d) It gets shifted to the right by 16 bits.
(e) It gets multiplied by 4.

Section IV: *Compilers and Loaders*

For the next three questions, match the phrase that best defines the concept:

(a) Translates assembly language into binary instructions.
(b) Translates source code into intermediates and immediately executes it.
(c) Combines independent programs and resolves labels into an executable file.
(d) Related collection of subroutines and data structures.
(e) Translate a high-level language into assembly language.

16. Compiler

17. Assembler

18. Module

For the next two questions, match the term that best defines the definition:

(a) Assembly Language
(b) Machine Language
(c) Source Language
(d) Target Language
(e) Object Language

19. Symbolic representation of a computer's binary encoding

20. High-Level Languages such as C or Java

Section V: *Number Representations*

21. Given the following numbers and the representations they should be interpreted as, which of the following is the correct ordering of *least* to *greatest*?
    i . 0xC0700000 in *floating point*
    ii. 0xFFFFFFFC in *two's complement*
    iii. 0xFF800000 in *floating point*
    (a) i, ii, iii
    (b) iii, ii, i
    (c) iii, i, ii
    (d) i, iii, ii
    (e) ii, i, iii

22. Flipping all of the bits in ones complement is the same as:
    (a) Adding one to the number
    (b) Subtracting one from the number
    (c) Subtracting the number from 2^n-1, where n is the number of bits
    (d) Dividing the number by negative one
    (e) Multiplying the number by negative one, then subtracting one

23. What is the smallest positive integer 32 bit IEEE floating point *cannot* represent?
    (a) 1
    (b) $2^{24} + 1$
    (c) $2^{128} - 1$
    (d) $2^{127} + 2^{126} + 2^{125} + \ldots + 2^{104} + 1$
    (e) $2^{128} + 2^{127} + 2^{126} + \ldots + 2^{105} + 1$

24. Give the result of adding the following binary numbers in **two's complement.** Assume we are working with 4 bit binary numbers only: **1010 + 1110**
    (a) -8_{ten}
    (b) 4_{ten}
    (c) -4_{ten}
    (d) Overflow
    (e) Underflow

25. Executing the following C code:
```
#include <stdio.h>
int a = 11;
if (0x2a == 42 && a) {
   printf("The cat is a cat cat.")
}
```
    will:
    (a) Print the sentence
    (b) Error when it gets to the third line
    (c) It doesn't actually compile
    (d) Error when it gets to the fourth line
    (e) None of the above

Section VI: Memory Hierarchy

26. Which of the following best describes an action likely to require fewer clock cycles than loading a whole cache line from memory.
    (a) Loading a subset of the cache line from memory.
    (b) Writing to a subset of the cache line with a write-through policy.
    (c) Writing to a subset of the cache line with a write-back policy.
    (d) Accessing a very small portion of the disk.
    (e) None of the above.

27. In the worst case scenario, cache performance can be:
    (a) Worse than series of direct access to memory
    (b) Still better than series of direct access to memory
    (c) Equivalent to series of direct access to memory
    (d) No relevance to series of direct access to memory
    (e) Equivalent to not having the cache there

28. The proportional difference in access time from CPU to L1 cache is ________ that from cache to memory.
    (a) Greater than
    (b) Less than
    (c) The same as
    (d) Undefined in respect to
    (e) Not enough information

29. The write policy of the cache:
    (a) Changes the memory address to which the data is written
    (b) Causes a memory access to fail
    (c) Affects the hit rate of read accesses on the cache
    (d) Must be declared by software
    (e) None of the above

30. If instructions without memory-related functions are called:
    (a) There is no way the caches would get accessed
    (b) There is no way the memory will be accessed
    (c) There is no way the disk will be accessed
    (d) All of the above
    (e) None of the above

Section VII: Cache Performance

Answer the following questions assuming the following memory/cache organization:

4 GiB Byte-Addressed Memory
256 KiB Direct-Mapped Cache
8 Byte Words
4 Word Blocks

31. Give the Tag : Index : Offset breakdown for the above cache:
    (a) 16 : 15 : 4
    (b) 14 : 13 : 3
    (c) 16 : 13 : 3
    (d) 14: 13 : 5
    (e) None of the above

32. Suppose we switch to a word-addressed memory. Give the Tag : Index : Offset breakdown:
    (a) 16 : 15 : 4
    (b) 14 : 13 : 3
    (c) 16 : 13 : 3
    (d) 14: 13 : 5
    (e) None of the above

33. Using the following code to answer the question. You may assume **`sizeof(int) = 4`**. Also assume that **`arr`** is block aligned.

```
#define WIDTH _?_
#define STRIDE _?_
void printer(int* arr) {
   // assume arr is of length WIDTH
   for (int i = STRIDE; i < WIDTH; i += STRIDE) {
         printf("%d\n", *(arr+i));
         printf("%d\n", *(arr+i-STRIDE));
   }
}
```

    Suppose WIDTH = 2^{20}. What is the smallest stride that produces the worst possible hit rate and what is that hit rate? (assume a cold cache, answers are STRIDE, Hit Rate).
    (a) 8, 0%
    (b) 16, 0%
    (c) 2^{16}, 0%
    (d) 1, 50%
    (e) 2^{20}, 10%

34. Calculate AMAT for a machine with the following specs: L1 hits take 3 cycles, L1 local miss rate is 25%. L2 hits take 10 cycles, L2 local hit rate is 60%. L3 hits take 100 cycles, L3 global miss rate is 9%. Main memory accesses take 1000 cycles and all data is available in memory.
    (a) 105.5 cycles
    (b) 1000 cycles
    (c) 24.5 cycles
    (d) 25.5 cycles
    (e) 278.25 cycles

35. Suppose we call **`printer()`** repeatedly with the same arr. How does increasing **`STRIDE`** change the "amount" of spatial and temporal locality with respect to **`arr`**?
    (a) Increasing STRIDE decreases temporal locality and increases spatial locality.
    (b) Increasing STRIDE increases temporal locality and increases spatial locality.
    (c) Increasing STRIDE does not change temporal locality and decreases spatial locality.
    (d) Increasing STRIDE increases temporal locality and decreases spatial locality.
    (e) Increasing STRIDE increases temporal locality and does not change spatial locality.

Section VIII: Parallelism

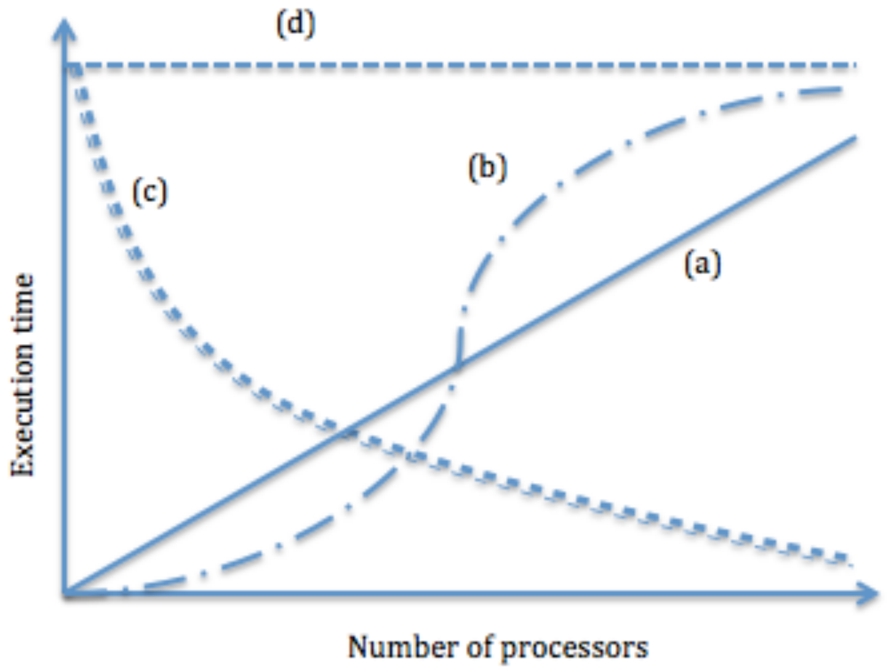
36. Which type of parallel computing architecture is no longer commonly encountered in machines today?
    (a) MIMD
    (b) MISD
    (c) SIMD
    (d) SISD
    (e) SNSD

37. When thousands of users are browsing (but maybe not buying) items on Amazon.com, what kind of parallelism are their servers handling?
    (a) EC2
    (b) Data-level
    (c) Hadoop
    (d) Request-level
    (e) Strong scaling

38. A given program runs x = 100 floating point operations, 10 of which cannot be parallelized. Assuming the other operations can be perfectly parallelized, what is the speed-up for 9 processors?
    (a) 10
    (b) 5
    (c) 9
    (d) 11.1
    (e) 0.9

39. You've run two MapReduce jobs using the same map and reduce functions. In the first, you processed 10GB in 7min 22sec, using a 5-machine cluster. In the second, you processed 30GB in approximately the same amount of time, using a 15-machine cluster. What kind of scaling did you observe, if any?
    (a) Weak scaling
    (b) Strong scaling
    (c) Weak and strong scaling
    (d) Inverse scaling
    (e) No scaling

40. Assuming a fixed data set, which lines on the graph below best represents strong and weak scaling?
    (a) Strong: A, Weak: C
    (b) Strong: A, Weak: D
    (c) Strong: C, Weak: A
    (d) Strong: C, Weak: D
    (e) Strong: D, Weak: B



<u>Section IX</u>: Lab and Project Knowledge

41. Suppose your Git repository has two branches, `branch1` and `branch2`. `branch1` contains only files `a.txt` and `b.txt`. `branch2` contains only a modified version of `a.txt`, as well as a new file, `c.txt`. If you are in `branch2`, what will happen after you run the command `git merge branch1`?
    (a) Nothing, because there's a syntax error with the merge command you just typed.
    (b) `Branch1` now contains `a.txt` and `b.txt`. `Branch2` now contains `a.txt`, `b.txt`, and `c.txt.`
    (c) `Branch1` now contains `a.txt`, `b.txt`, and `c.txt`. `Branch2` now contains `a.txt` and `b.txt`.
    (d) The two branches are merged one branch containing `a.txt`, `b.txt`, and `c.txt`.
    (e) The merge does not take place due to the conflicts between the two versions of `a.txt`.

42. Which one of the following statements regarding Hadoop is ***false***?
    (a) A combiner in Hadoop must override the reduce() function.
    (b) Data types that are used as keys in Hadoop implement the Writable interface.
    (c) Multiple copies of a map task may run at the same time.
    (d) Hadoop can handle multiple worker failures, but is unable to handle master failure.
    (e) A reduce task cannot start until all of its input data has been moved onto the worker.

43. You want to change the Map output value type in your Hadoop project from **`IntWritable`** to **`DoubleWritable`**. You've already made all the changes to your **`map()`** and **`reduce()`** functions, including the function signatures, but nothing else. Assume that you are not using a combiner, and that you've included the statement **`import org.apache.hadoop.io.*`** What is the minimum number of additional lines you'll need to change to get your code to compile?
    (a) 0
    (b) 1
    (c) 2
    (d) 3
    (e) 4

44. You are debugging the following snippet of code with GDB:

```
Line   Code
20     int main(void) {
21         int x = 2, *y = x;
22         *y = square(x);  //square() is defined elsewhere
```

You set a breakpoint by typing the command `break 22`, and then you type run. You then type `step`. What line gets printed by GDB when you hit the breakpoint, and what happens after you type `step`?

(a) Line 21, and the program continues execution when you type step
(b) Line 21, and the program crashes when you type step since line 22 is buggy
(c) Line 22, and the program continues execution when you type step
(d) Line 22, and the program crashes when you type step since line 22 is buggy
(e) Neither lines, the program crashes before GDB can pause at the breakpoint

45. Fill in the blank: When MARS encounters a syscall, it checks the _____ register to see which command it should execute.
    (a) **`$at`**
    (b) **`$a0`**
    (c) **`$s0`**
    (d) **`$t0`**
    (e) **`$v0`**

Section X: *MIPS Conventions*

46. The following is pseudocode for a recursive function that finds the total sum of the sizes of each element of a hailstone sequence (you don't need to know what these are).

```
hailstone(int n)
   if(n==1), return n
   else if (n is even), return n + hailstone(n/2)
   else return n + hailstone(3*n+1)
```

Using proper, standard MIPS convention, how many registers should we store to the stack if we were to implement this function with this exact algorithm in MIPS? Assume we use no **$s** registers.

(a) 0
(b) 1
(c) 2
(d) 4
(e) None of the above

47. The following is some messy code written by an angsty Stanford student who doesn't want to play by the rules.

```
repeat:
   addi $sp, $sp, -4
   sw $ra, 0($sp)
   beq $a0, $zero, bob
   addi $a0, $a0, -1
   jal task #task is a function written by a Cal student
   lw $ra, 0($sp)
   addi $sp, $sp, 4
   j mary
bob:
   jr $ra
mary:
   addi $s1,$ra,-4
   jr $s1
```

Assuming I have a program that uses proper MIPS conventions, and assuming task uses proper MIPS conventions, which of the following options could go wrong if I do the following in my program?

```
   addi $a0,$0,8
   jal repeat
```

(i) Memory accesses from the stack in my program will be off.
(ii) Repeat might run task 1 time instead of 8.
(iii) If repeat returns, it might not return to the correct address,

(a) The program will run fine
(b) (i) and (ii)
(c) (ii) and (iii)
(d) (i) and (iii)
(e) (i), (ii), and (iii)

48. The following is a small snippet of code from a function. What is the maximum number of *load* or *store* instructions that can be eliminated from this clip of code so it still works

as desired for an arbitrary function passed in **$a1**? Assume the function in **$a1** obeys proper MIPS conventions, and that the clip of code works properly with none of the loads or stores eliminated.

```
# if f is the function stored in $a1,
# then this is supposed to return f($a0) + f($a0)
# code before
   move  $s0,$a1
   addi  $sp,$sp,-16
   sw    $ra,0($sp)
   sw    $a0,4($sp)
   sw    $s0,8($sp)
   jalr  $s0
   sw    $v0,12($sp)
   lw    $a0,4($sp)
   lw    $s0,8($sp)
   jalr  $s0
   lw    $t0,12($sp)
   add   $v0,$v0,$t0
   lw    $ra,0($sp)
   addi  $sp,$sp,16
#code after
```

(a) less than 2
(b) 2
(c) 3
(d) 4
(e) 5 or more

49. Which of the following is guaranteed to stay unchanged after you call a function that uses good MIPS conventions?
(a) **$ra**
(b) **$a2**
(c) **$sp**
(d) **$v1**
(e) More than one of the above

50. Which of the following is true?
(a) MIPS can store words to memory locations occupied by the program (e.g., **sw $s0, 0($ra))**
(b) addiu will error if overflow or underflow occur
(c) MIPS is primarily little-endian
(d) MIPS uses the CISC architecture paradigm
(e) More than one of the above