

# CS61C FINAL EXAM: EARLY VERSION, 5/11/99

Last name \_\_\_\_\_ First name \_\_\_\_\_  
 Student ID number \_\_\_\_\_ Login: cs61c-\_\_\_\_\_

**Please circle the last two letters of your login name.**

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
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

Discussion section meeting time \_\_\_\_\_ TA's name \_\_\_\_\_  
 The student on my left is \_\_\_\_\_ login cs61c-\_\_\_\_\_  
 The student on my right is \_\_\_\_\_ login cs61c-\_\_\_\_\_

You are allowed to use two 8.5" x 11" double-sided handwritten pages of notes. **No calculators.** This booklet contains 11 numbered pages including the cover page, plus photocopied pages from COD and an ASCII table. Put all answers on these pages, please; don't hand in stray pieces of paper. The exam contains 9 substantive questions, plus the dreaded question 0 and the extra credit question. You have three hours, so relax – this exam isn't worth having a heart failure over. Good luck!

**I certify that my answers to this exam are all my own work. If I am taking this exam early, I certify that I shall not discuss the exam questions or answers with anyone until after the scheduled exam time.**

Signature \_\_\_\_\_

Question	Max Points	Your Points
<b>The Dreaded Question 0</b>	<b>(-1 to 0)</b>	
<b>Performance</b>	<b>7</b>	
<b>Pliable Data, The Revenge</b>	<b>6</b>	
<b>Theme and Variations</b>	<b>3</b>	
<b>Cache and VM</b>	<b>6</b>	
<b>The Newsgroup Question</b>	<b>11</b>	
<b>HLL to Asm</b>	<b>12</b>	
<b>Dave's Dirty Clothes</b>	<b>6</b>	
<b>Interrupt Questions</b>	<b>4</b>	
<b>Tree Monkey Question</b>	<b>15</b>	
<b>Extra Credit</b>	<b>(+1)</b>	
<b>Total</b>	<b>70</b>	

(If you are feeling down, starting the test by reading the Extra Credit question might help to inspire you ☺)

**Question 0 (-1 point if not followed):** Fill out the front page correctly and write your login at the top of each of the pages. Circle your login initials on the cover page so that we can read them.

**Performance Question (7 points):**

You are running a benchmark on your company's processor, **Mbase**, which runs at 400 MHz and has the following characteristics:

**Mbase:**

Instruction Type	Frequency	Cycles
A	40%	2
B	30%	3
C	20%	3
D	10%	5

a) (2 points) What is the CPI rating for **Mbase**?

You ask the hardware team if they can improve the processor design. They tell you that they could make this processor run at 500 MHz, however they would have to increase the number of cycles for instruction type C to 4. (All the other instruction types still take the same number of cycles). Call this machine **Mopt**.

**Mopt:**

Instruction Type	Frequency	Cycles
A	40%	2
B	30%	3
C	20%	4
D	10%	5

b) (1 point) What is the CPI rating for **Mopt**?

c) (2 points) How much faster is **Mopt** than **Mbase** (you may leave the answer as an improper fraction)?

d) (2 points) Is there an instruction mix that makes **Mbase** faster than **Mopt**? If so, suggest such a mix. (Note: The mix doesn't have to contain all the instruction types.)

**Pliable Data, The Revenge (6 points):** (use the sheets appended to the final)

Dave has to attend a "conference" in the Bahamas following the 61C final, so all the TA's take him to the airport to send him off. As he steps on the plane, he tells the TA's that he's left some work in his office, and he asks them to finish it for him.

Kelvin thinks to himself, "I bet if I raced back and finished all of Dave's work by myself, I'd feel cocky enough to start answering people's questions on the newsgroup." He jets back to Dave's office and finds a single paper on his desk with the following line written at the top of the page:

**0011 0010 0011 0010 0011 0010 0000 0000 = ?**

Kelvin notices that Dave has scrawled some notes on the bottom of the paper:

0011 0010 (base 2) = 50 (base 10)  
 0110 0100 (base 2) = 100 (base 10)  
 $2^7 = 128$   
 $2^8 = 256$

a.) **(2 points)** Kelvin thinks, "Well Dave was always telling us how he wants to become a walking disassembler. He must want the MIPS instruction corresponding to this bit pattern." Which MIPS instruction should Kelvin write?

b.) **0011 0010 0011 0010 0011 0010 0000 0000 = ?**

**(2 points)** Mark comes in just before Kelvin starts writing and snatches the paper from him. He looks at the page and says, "Oh, Dave has written down the elements of a C-style string in binary. He must want to know what string it represents." Which string should Mark write?

c.) **0011 0010 0011 0010 0011 0010 0000 0000 = ?**

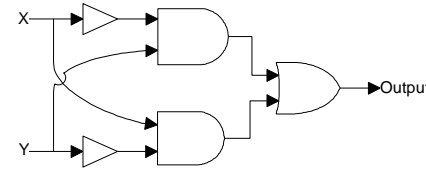
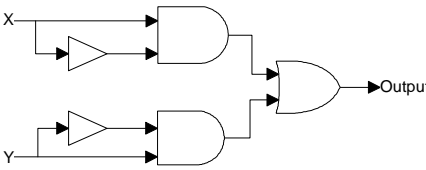
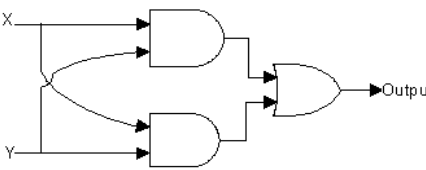
**(2 points)** Gek comes in just before Mark starts writing and snatches the paper from him. He looks at the page and says, "Oh, Dave has written down a single precision floating point number. I don't have time to figure out the significand, but I'll write down the sign and the exponent in decimal." He writes (+/-) ??? x 2^\_\_\_\_\_. Which sign and what value for the exponent should Gek write?

Sign: \_\_\_\_\_ Exponent: \_\_\_\_\_

d.) [Bonus question worth 0 points]  
 The rest of the TA's all come piling into the office just before Gek starts writing and they say, "After you left, Dave clarified that the work he wanted us to do was finish painting M'Piero on his wall." What should the TA's draw on the wall? [Use the back of the sheet if necessary.]

**Theme and Variations (3 points):**

Match the C expressions on the left with the logic circuits on the right.

<p>Question 1</p> <p>Output = 0</p>	 <p>A</p>
<p>Question 2</p> <p>Output = (X &amp;&amp; !Y)    (Y &amp;&amp; !X)</p>	 <p>B</p>
<p>Question 3</p> <p>Output = (X &amp;&amp; Y)</p>	 <p>C</p>

Write your answers here:

1. \_\_\_\_\_ 2. \_\_\_\_\_ 3. \_\_\_\_\_

**Cache and VM Question (6 points):**

The page size of a computer is 16 **Kbytes**; the block size is 32 **words** and the machine is **byte**-addressable. The cache size is 1 **Kbyte**, and it is 4-way set associative. The virtual addresses are 42 bits and physical addresses are 36 bits long. Calculate the sizes (number of bits) of the following fields:

- a) block offset \_\_\_\_\_
- b) set index \_\_\_\_\_
- c) tag \_\_\_\_\_
- d) page offset \_\_\_\_\_
- e) virtual page number \_\_\_\_\_
- f) physical page number \_\_\_\_\_

**The Newsgroup Question (11 points):**

Although the CS61C review lecture on variable arguments was great, Joe Computer is very puzzled. What is this M'Piero thing? What does it have to do with Kelvin? "I don't get it!" Determined to find the answer to his confusion, he decides to check the newsgroup by starting the program "trn."

**In what order do things happen when trn is run? Part 1 lists a set of things that occur when trn is run. Please time-order the steps from 1 to 13. The odd numbered steps are given to you. Fill in the rest with even numbers.**

Please assume:

- No part of the program has been loaded into memory yet.
- Page size is 4KB and there is only one cache.
- The page table entry loaded from the memory for page 0x00040 maps to physical page 0x14329.
- The TLB is between the CPU and the cache, as in class (the cache uses physical addresses).
- Block size for the cache is 8 words (32 bytes).
- **In part 1 all of the actions occur. In part 2, some of them are incorrect and do not occur.**

**Part 1 (6 points):**

Given steps:

- \_\_1\_\_ Joe Computer types "trn" at the command line.
- \_\_3\_\_ The CPU attempts to fetch the first instruction, 0x00040000 (pointed to by the pc).
- \_\_5\_\_ The page table for this process is accessed to find the entry for address 0x00040000, which has the invalid bit set (not loaded from disk yet).
- \_\_7\_\_ The TLB is updated with an entry mapping virtual page 0x00040 to physical page 0x14329, with the valid bit set.
- \_\_9\_\_ The cache misses for the block containing 0x14329000 and attempts to load the block from memory.
- \_\_11\_\_ The instruction at virtual address 0x00040000 is successfully loaded from the cache, completing the instruction fetch phase.
- \_\_13\_\_ The CPU attempts to fetch the second instruction, 0x00040004.

Unordered steps: (Assign the even step numbers 2, 4, 6, 8, 10, and 12 to the six options below)

- \_\_\_\_\_ The TLB hits for virtual page number 0x00040, the physical address 0x14329000 is sent to the cache.
- \_\_\_\_\_ The TLB misses while attempting to find an entry for the virtual page number 0x00040.
- \_\_\_\_\_ Physical page number 0x14329 is loaded into memory from disk, and the page table is updated.
- \_\_\_\_\_ The instruction at virtual address 0x00040000 is successfully fetched, and on the next clock tick will move on to its decode stage.
- \_\_\_\_\_ A page table for the process is created by the operating system. Static memory area is created, space is allocated for the static parts (i.e. arrays) of the program, heap and stack are initialized. All the TLB entries from the previous process are marked invalid.
- \_\_\_\_\_ The block containing 0x14329000 is loaded into the cache from memory.

## [Newsgroup Question continued]

### Part 2 (5 points):

Now that you have ordered what happens for the first instruction, what will happen for the second instruction? (Assume that this question starts where Part 1 left off. **Remember, some of these may NOT occur.** Please order the \*correct\* actions [starting with the number 1], and put an "X" in front of incorrect actions):

- \_\_\_\_\_ The TLB misses for the virtual page corresponding to address 0x00040004, and the previous procedures are used to load the right page into memory and update the page table.
- \_\_\_\_\_ The cache misses for the block containing 0x14329004 and attempts to load the block from memory.
- \_\_\_\_\_ The TLB hits for virtual page number 0x00040, the physical address 0x14329004 is sent to the cache.
- \_\_\_\_\_ (after many more instructions are executed)... The newsgroup article is read, M'Piero is displayed on the screen, Joe Computer finally gets the joke and posts a message praising the fact that CS61C has such a nifty teaching staff this semester ☺.
- \_\_\_\_\_ The instruction at virtual address 0x00040004 is successfully loaded from the cache, completing its instruction fetch phase.
- \_\_\_\_\_ The block containing 0x14329000 is loaded into the cache from memory.

**HLL to Asm Question (12 points):**

Below are four pieces of C code and four pieces of MIPS code. In the blank provided next to each piece of C code, enter the letter corresponding to the MIPS code that accurately performs EXACTLY the same function as the C code. **Each piece of MIPS code matches to exactly one and only one piece of C code.** There are no repeats. There are no “none of the above” answers.

Assume that, at the beginning of each piece of C code, the integer variable *i* contains as its value a non-negative integer. Also assume that both arrays are integer arrays of size 20 (so there will be no addressing problems in any of the code).

Let the following be true of the registers prior to execution of the MIPS code:

- \$t0 contains the value the integer variable *i* would contain in the corresponding C code
- \$s0 contains the address of A[0]
- \$s1 contains the address of B[0]

C code:

<p>1. _____</p> <pre>do {     A[i] = B[i] + i;     i++; } while (i &lt; 10);</pre>	<p>2. _____</p> <pre>while (i &lt; 10) {     A[i] = B[i];     A[i] += i; }</pre>	<p>3. _____</p> <pre>if (i &lt; 10) {     A[i] = B[i];     A[i] += i;     i++; }</pre>	<p>4. _____</p> <pre>for (; i &lt;= 9; i++) {     A[i] = B[i] + i; }</pre>
--	--	--	--

MIPS code:

A.

```
Begin: slti    $t1, $t0, 10
       beq     $t1, $0, Fin
       sll    $t1, $t0, 2
       add   $t2, $s1, $t1
       lw   $t3, 0($t2)
       add  $t3, $t3, $t0
       add  $t2, $s0, $t1
       sw   $t3, 0($t2)
       addi $t0, $t0, 1
       j    Begin
Fin:
```

B.

```
Begin: slti    $t1, $t0, 10
       beq     $t1, $0, Fin
       sll    $t1, $t0, 2
       add   $t2, $s1, $t1
       lw   $t3, 0($t2)
       add  $t3, $t3, $t0
       add  $t2, $s0, $t1
       sw   $t3, 0($t2)
       j    Begin
Fin:
```

C.

```
Begin: sll    $t1, $t0, 2
       add   $t2, $s1, $t1
       lw   $t3, 0($t2)
       add  $t3, $t3, $t0
       add  $t2, $s0, $t1
       sw   $t3, 0($t2)
       addi $t0, $t0, 1
       slti $t1, $t0, 10
       beq  $t1, $0, Fin
       j    Begin
Fin:
```

D.

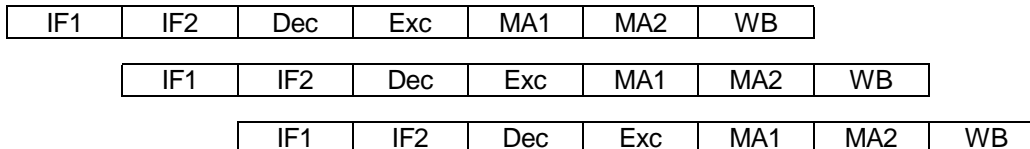
```
Begin: slti    $t1, $t0, 10
       beq     $t1, $0, Fin
       sll    $t1, $t0, 2
       add   $t2, $s1, $t1
       lw   $t3, 0($t2)
       add  $t3, $t3, $t0
       add  $t2, $s0, $t1
       sw   $t3, 0($t2)
       addi $t0, $t0, 1
Fin:
```

**Dave's Dirty Clothes Question (6 points):**

The pipeline example in the book and in class used 5 stages (instruction fetch, instruction decode/register read, execute, data memory, and write back), and the time per stage was 2 ns for memory access, 2 ns for ALU operation, and 1 ns for register file read or write. Suppose we now replace the ALU with a faster version that runs in 1 ns, and we change the instruction cache and data cache so that they can accept a new address every 1 ns, but it still takes 2 ns to access the instruction or data. (The latency is still 2 ns, but throughput is one word every 1 ns.) The time for the rest of the stages is unchanged. Our new pipeline has 7 stages, as follows:

- 1) Instruction Fetch Part A (start fetching next instruction)
- 2) Instruction Fetch Part B (finish fetching next instruction)
- 3) Instruction decode/register read/perform branch
- 4) Execute
- 5) Data Access Part A (start reading or writing desired data)
- 6) Data Access Part B (finish reading or writing desired data)
- 7) Write Back

This would look something like:



Select **all** of the following statements that are true about these two pipeline organizations. (Circle the letter if the statement is true, leave it unmarked otherwise).

- a. The original pipeline could have a clock rate of 500 MHz, and the new pipeline could have a clock rate of 1000 MHz.
- b. The peak instruction throughput is much improved in the new pipeline, but the instruction latency is unchanged.
- c. If the old pipeline stalled for one clock cycle on an instruction dependent on a load, the new pipeline would have to stall for up to two clock cycles on an instruction dependent on a load.
- d. If the old pipeline used a delayed branch mechanism (with one branch delay slot), the new pipeline would have a delayed branch mechanism with two branch delay slots.

**Interrupt Questions (4 points):**

Circle T or F:

- 1) T or F: When an exception or I/O interrupt occurs, interrupts are disabled while vital information is being saved.
- 2) T or F: Both add and addu can overflow. The only difference is that addu doesn't trigger an exception.
- 3) T or F: The instructions, lw and sw, do not trigger exceptions or I/O interrupts.
- 4) T or F: In order for an I/O interrupt to occur (for example when a key on the keyboard is pressed), the only bit that needs to be set is the device's I.E., found in one of its registers.



**Tree Monkey Question (15 points):**

You are a tree monkey who owns a banana tree. Foreseeing the long winter coming, you wish to count the number of ripe banana's currently in the tree so you can ration them over the winter.

Being a smart monkey, you will write a computer program to count the bananas in the tree. You represent the tree as a binary tree in which each node has a number representing the bananas in the node and maximally two branches to other nodes in the tree. If the node is a leaf then both branches will point to NULL. Here is the C structure to represent a node in the tree:

```
struct TreeNode {
    int numBananas;

    struct TreeNode* left;
    struct TreeNode* right;
}
```

Being a smart monkey in a matter of seconds you come up with the following C function which, handily enough, is recursive. In other words, this function calls itself.

```
int CountBananas(struct TreeNode *pBananaTree) {
    if(pBananaTree == NULL)
        return 0;

    return CountBananas(pBananaTree->right) +
        CountBananas(pBananaTree->left) +
        pBananas->numBananas;
}
```

Unfortunately, your monkey compiler broke, so you will have to compile this function by hand. Being a good monkey you will write comments as needed. Below is a main procedure which will call your function. You are to fill in the code for the CountBananas procedure, which begins on the next page. (Note: **Solutions that are not recursive will not get credit.** Also, you must use the TreeNode struct as is. Try to stick as close to the C function given.)

```
__start:      .text
              la    $a0, ROOT          #put the root in arg0
              jal   CountBananas

              add   $t0, $0, $v0       #put the value into temp
              puti  $t0                 #print it out
              done                    #end of program
```

[The CountBananas procedure is to be written on the next page.]

**[Tree Monkey Question continued]**

(Fill in the code for CountBananas here.)

CountBananas :

**Extra Credit Question (+1 point):**

“Nothing in this world can take the place of persistence.  
Talent will not; nothing is more common than unsuccessful men with talent.  
Genius will not; unrewarded genius is almost a proverb.  
Education alone will not; the world is full of educated derelicts.  
Persistence and determination alone are omnipotent.”

Question: Who is credited with this quote?