CS 61C Final — December 6, 1994

Your name _____

login cs61c–_____

This exam is worth 40 points, or 30% of your total course grade. The exam contains eight questions.

This booklet contains ten numbered pages including the cover page. Put all answers on these pages, please; don't hand in stray pieces of paper. This is an open book exam.

Our expectation is that many of you will not complete one or two of these questions. If you find one question especially difficult, leave it for later; start with the ones you find easier.

1	/4
2	/6
3	/6
4	/5
5	/4
6	/6
7	/5
8	/4
total	/40

Question 1 (4 points):

A. The hexadecimal representation for the binary number

```
01110010100101110010 10
```

is:

```
a. 0x729728
b. 0x729722
c. 0x1ca5ca
d. 0x1ba5ba
```

B. All of the following numbers have the same value *except*:

```
a. Two's complement: 11110111
b. One's complement: 11110110
c. Sign magnitude:   10001000
d. Bias-127:         01110110
```

C. An ASCII keyboard has 47 keys that produce graphic characters. Each of those can be used alone, or along with the CONTROL or the SHIFT key, but not both. (This is a simplification of the real story, but close.) For example, you can type the `A` key (producing `a`), `shift-A` (producing `A`), or `control-A` (producing an invisible character).

How many codes can this keyboard produce? How many bits does it take to represent one of them?

D. Translate the MAL instruction

```
addiu $29, -12
```

into hexadecimal machine language.

Your name _____ login cs61c–_____

Question 2 (6 points):

This two-part question asks you to figure out why two parts of the design of a computer you've never seen (one that was very successful in its day) were not used in the MIPS design. You're not expected to have seen these features before; you should be able to deduce whatever you need to know from the descriptions here.

A. The PDP-10 had a Block Transfer instruction (`blt`) for copying a consecutive chunk of memory (such as an array) from one place to another. Its operands were the beginning source address, the beginning destination address, and the length of the transfer. This instruction allows a high level language to have assignments from one array variable to another, instead of a loop to copy the array element by element. Why wouldn't this instruction make sense in the MIPS architecture? (One or two sentences should be enough.)

B. The PDP-10 had four different procedure-calling instructions! (One of them, called `jsp` for Jump and Save PC, was much like the `jal` instruction on the MIPS machines.) We'd like you to comment on another of them, the `jsr` (Jump to SubRoutine). It was designed to allow procedure calling without a stack. Its operand was a memory address. The instruction stored the PC at that address, then started executing instructions at the following word. So a subroutine would start this way:

```
subr:   .word  0            ; reserve space for PC
        addi   ...          ; first instruction of subroutine
```

The subroutine would return with a jump-indirect instruction whose operand was the address `subr`, i.e., the address containing the return address. Give *two* reasons (no more than two sentences each) why this procedure calling mechanism wouldn't be a good idea in the MIPS architecture.

Question 3 (6 points):

Translate the following C function into a MAL procedure. Use `$16` for local variable `i`. Parameters are passed in registers `$4`, `$5`, and `$6`. Use the register convention discussed in class. Notice that `a` and `b` are arrays.

```
void mapArray(int a[], int b[], int dim)
{
  int i;
  extern int fun(int);

  for (i = 0; i < dim; i++)
    a[i] = fun(b[i]);
}
```

Your name _____ login cs61c–_____

Question 4 (5 points):

Write a MIPS assembly procedure procedure, `numBitsSet`, which returns as its integer result the number of `1` bits in its integer argument. For example, the integer `9` has the binary representation `0...01001` and so it has two `1` bits, so `numBitsSet(9)` returns the value 2.

Question 5 (4 points):

A. Consider the following cache geometry:

```
Data size: 4096 bytes
Width: 16 bytes
Type: Direct-mapped
```

The address `0x7f55a356` will be mapped to:

```
a. Set number: 0xa35, tag:   0x7f55
b. Set number:  0x35, tag:  0x7f55a
c. Set number: 0x356, tag:  0x7f55a
d. Set number:  0x56, tag: 0x7f55a3
```

B. What is the number of sets in an 80 Kbyte, 5-way set associative cache with a block size of 32 bytes? (1 Kbyte = 1024 bytes.)

C. Suppose that a page contains 4 Kbytes. A virtual address and a physical address are both 32 bits wide. Here is the TLB:

	Virtual page #	Physical page #
0	0x00678	0x27
1	0x01234	0x543
2	0x12345	0x200
3	0x45678	0x5

Convert the virtual address `0x12345678` into a physical address.

D. An operating system could allow any of the following to be paged out *except*:

```
a. page tables
b. disk I/O handler
c. system call handler
d. user programs
```

Your name _____ login cs61c–_____

Question 6 (6 points):

Running the `cache` program from homework 8 on the Wazcog Mark IV computer, you get the following results:

```
stride->          4       8      16      32      64     128     256

size
----
  64:           109     126     122     121     101     115     116
 128:           120     118     116     121     117     105     112
 256:           119     116     102     117     113     113     110
 512:           466     966     949     949     983     308     311
1024:           458     949     983     966     949     983     307
2048:           449     992     954     966     979     979     979
```

(Stride and size in bytes, times in nanoseconds. Yes, this is an artificially small example, to make the numbers fit easily.)

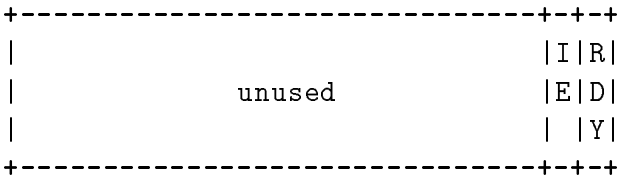
What is the cache size in bytes?

What is the block size in bytes?

What is the allocation policy? (Direct mapped, fully associative, or set associative? If N-way set associative, what is N?)

Question 7 (5 points):

Suppose that we attach to our MIPS computer a timer device. It has one register, the control register, at location `0xffff0010`. The control register is similar to those of the other devices we've seen:



Every sixtieth of a second (a *tick*), the timer turns on its ready bit. If its interrupt enable bit is also set, it interrupts. As soon as the central processor reads the control register, the ready bit is turned off. (The processor sees the old value of the bit. Therefore, each tick will be seen by the processor exactly once.)

We want a user program to be able to suspend itself for a chosen number of ticks this way:

```
        li   $4,number-of-ticks
        jal  sleep
```

Here is a simplified version of the `sleep` procedure. As in the I/O lab projects, we'll let the procedure loop while waiting, although in a real system it would instead switch control to a different user program.

```
sleep:  sw   $4, SleepCount
        lui  $8, 0xffff
        li   $9, 2
        sw   $9, 0x10($8)       ; turn on timer interrupt enable
loop:   lw   $8, SleepCount
        bgtz $8, loop
        jr   $31
```

Your job is to write the exception handler that checks if the timer is ready, and if so, subtract one from the value in `SleepCount`. When that value becomes zero, turn off timer interrupts.

Write your answer on the next page.

Your name _____ login cs61c–_____

Question 7 continued:

```
        .ktext  0x80000080

intrp:
```

Question 8 (4 points):

A. When an exception occurs, the MIPS processor does all of the following, *except*:

```
a. reads the Cause register.
b. runs the code starting at location 0x80000080.
c. switches to kernel mode and disables interrupts.
d. saves the address of the instruction that raised the exception.
```

B. The main advantage of using interrupts is:

```
a. allows the processor to do other useful tasks
      while waiting for slow I/O.
b. allows centralized error handling.
c. allows the processor to switch to kernel mode.
d. allows a user program to have access to I/O devices.
```

C. In two sentences or less, explain the differences between kernel mode and user mode.