CS 61C Final — August 13th, 1998

Your name ____

login cs61c–_______

This exam is worth 60 points, or 30% of your total course grade. The exam contains eight substantive questions, plus the following:

Question 0 (1 point): Fill out this front page correctly and put your name and login correctly at the top of each of the following pages.

This booklet contains xx numbered pages including the cover page, plus a copy of the back inside cover of Patterson & Henessey. Put all answers on these pages, please; don't hand in stray pieces of paper. This is a closed book exam, calucaltors are allowed.

When writing procedures, write straightforward code. Do not try to make your program slightly more efficient at the cost of making it impossible to read and understand. You do not need to include error checks.

If you find one question especially difficult, leave it for later; start with the ones you find easier. We will use round to even as our rounding mode to round all fractional points to integer values.

READ AND SIGN THIS:

I certify that my answers to this exam are all my own work, and that I have not discussed the exam questions or answers with anyone prior to taking this exam.

If I am taking this exam early, I certify that I will not discuss this exam until everyone has completed the exam at the normal time.

____

0	/1
1	/8
2	/8
3	/8
4	/8
5	/9
6	/9
7	/9
total	/60

This page left blank

Your name ____ login cs61c–_______

Question 1, ***Vocabulary and TLAs*** **(8 points):**

(1 point): What does the MEM stage of the 5 stage pipeline do?

Issues reads and writes to memory

(1 point): What is a packet?

A block of data sent in the network

(1 point): Where is Nick Weaver's Office?

615 Soda Hall

(2 point): What part of the 5 stage pipeline performs address calculations?

The alu, or EX phase

(1 point): What is a branch delay slot?

The instruction immediatly after a branch, which is executed whether or not the branch is taken.

(2 point): What does the TLB do?

It translates a virtual address to a physical address, acting as a cache for the page table.

(.1 point): What does TLA mean?

Three Letter Acronym

Your name ____ login cs61c–_______

Question 2 (8 points) review:

(2 points): Add the following 8 bit, **sign/magnitude** numbers.

```
   10001101           10001011
  +01010000          +10001011
 -----------        -----------
   01000011           10010110
```

(2 point): Write out the hexidecmal representation of the double precision IEEE floating number of the largest magnitude. (IEEE double has an 11 bit exponent with a bias of 1023)

```
0 11111111110 1111....
0111 1111 1110 1111....
7FEF FFFF FFFF FFFF
```

(1 point): True or false: A logical right shift of N is equivelent to division by 2^N for an **unsigned** number.

true

(1 point): What is the hexadecemal representation of the number 42?

2A

(2 point): Which registers are not saved across function calls?

\$a0-a3, \$v0-v1, \$t0-t10, \$ra (not including \$ra is OK)

Your name ____ login cs61c–_______

Question 3, structures, arrays, pointers (8 points):

Consider the following declaration and function:

```
struct bar{
  struct bar *next;
  int i;
  void (*fn)(struct bar *);
  char c[4];
};

struct bar *b; int i;
```

Assume that `i` is in `$s0` and that `b` is in `$s1`. You can use temporary registers, but not saved registers. Translate the following C expressions into MIPS assembly.

```
____ b = b->next;
   lw $s1 0($s1)

____ b->c[i] = b->c[b->i];
  lw $t0 4($s1)
  add $t0 $s1 $t0
  lw $t1 12($t0)
  add $t0 $s1 $s0
  sw $t1 12($t0)

____ b->next = b[i].next->next;
  sll $t0 $s0 4 # Mult by 16 for indexing
  add $t0 $t0 $s1
  lw $t0 0($t0) # b[i].next
  lw $t0 0($t0) # b[i].next->next
  sw $t0 0($s1) # write out

____ (b[i].fn)(b);
  sll $t0 $s0 4 # Mult by 16 for index
  add $t0 $t0 $s1
  lw $t0 8($t0) # function
  move $a0 $s1
  jalr $t0
```

Your name ____ login cs61c–_______

Question 4, Caches 1 (8 points):

(2 point): A 32kB cache has a linesize of 16 bytes and is 4 way set-associative. How many bits of an address will be the tag? Index? Offset?

19 bits are required for the tag

9 bits are required for the index

4 bits are required for the offset

(2 point): In a 2 way set associative cache, three addresses, **A**, **B** and **C** all map to the same cache line but have separate tags. What is a minimum sequence of accesses which, if repeated, will maximize the miss rate in the cache?

A B C

(1 point): If the above sequence is repeated for a long period of time, what will the miss rate be if the cache uses an LRU replacement policy?

1 (all misses)

(1 points): If the hit time is 1 cycle, and the miss **penalty** is 3 cycles, what will the average memory access time (in clock cycles) for the LRU replacement policy using the above sequence?

4 cycles

(1 point): If the sequence is repeated for a long period of time, will the miss rate be improved if random is used as the replacement policy?

Miss rate will be improved by random

(1 point): Which of LRU or random will have a lower miss rate if the sequence is **A B C C B A A B C ...**?

LRU will have a lower miss rate.

Question 5, Caches 2 (9 points):

(4 points) Gill Bates was lazy when completing project 5, so instead of running the cache.c program, he decided to fake the numbers, but got stuck. He knew that the machine in question had a 4 way set associative cache with LRU replacement policy, used 16 byte cache lines, was 8kB in size, and required 40 ns to do a read+write on a cache hit, and 160 ns to do a read+write on a cache miss. Complete the fake output which will lead one to believe that the output reflects a cache of the stated parameters. Do NOT introduce any noise into your figures. You should easily be able to exactly calculate the numbers for each entry you need to complete. (You can write " (ditto) if an entry should be the same as the entry above it).

```
Size:   4096 Stride:      4 read+write:     40 ns
Size:   4096 Stride:      8 read+write:     40 ns
                        ...
Size:   4096 Stride:   1024 read+write:     40 ns
Size:   4096 Stride:   2048 read+write:     40 ns
Size:   8192 Stride:      4 read+write:     40 ns
Size:   8192 Stride:      8 read+write:     40 ns
Size:   8192 Stride:     16 read+write:     40 ns
Size:   8192 Stride:     32 read+write:     40 ns
Size:   8192 Stride:     64 read+write:     40 ns
Size:   8192 Stride:    128 read+write:     40 ns
Size:   8192 Stride:    256 read+write:     40 ns
Size:   8192 Stride:    512 read+write:     40 ns
Size:   8192 Stride:   1024 read+write:     40 ns
Size:   8192 Stride:   2048 read+write:     40 ns
Size:   8192 Stride:   4096 read+write:     40 ns
Size:  16384 Stride:      4 read+write:     70 ns
Size:  16384 Stride:      8 read+write:    100 ns
Size:  16384 Stride:     16 read+write:    160 ns
Size:  16384 Stride:     32 read+write:    160 ns
Size:  16384 Stride:     64 read+write:    160 ns
Size:  16384 Stride:    128 read+write:    160 ns
Size:  16384 Stride:    256 read+write:    160 ns
Size:  16384 Stride:    512 read+write:    160 ns
Size:  16384 Stride:   1024 read+write:    160 ns
Size:  16384 Stride:   2048 read+write:    160 ns
Size:  16384 Stride:   4096 read+write:     40 ns
Size:  16384 Stride:   8192 read+write:     40 ns
```

(2 points) Writeback caches have a mechanism for "flushing" the cache, which will cause all dirty lines to be written out to main memory. Similarly, all caches have a mechanism to invalidate all cached data (set the valid bit to 0) for all cache lines. A processor has a writeback L1 data cache and a separate L1 instruction cache and has no coherancy between the two. If a program wishes to modify its own code, what will the program need to do to the instruction and data caches before the modified code can be correctly executed? (Hint: The code will be written into the Dcache, but not into main memory. Also, the ICache will have an old copy of the code in question).

Flush the Dcache, invalidate the Icache

(3 points) If the page size is suitably large, the cache can be "virtually addressed", using the virtual address instead of the physical address to access and fetch data. This allows translation to occur after the cache determines if a hit or miss occurs, and only if there is a miss in the cache. If the cache has a line size of 2^L bytes, an associativity of 2^K and a cache size of 2^N bytes. What is the minimum page size in order for the cache to be virtually addressed? (Hint: Only the tag portion of the address can undergo translation if a cache is to be virtually addressed)

$N - K$ bits are needed for the index and offset. So the page size must be at least 2^{N-K}.

Your name ____ login cs61c–_______

Question 6, Pipelining and Dependencies (9 points):

Given the following MIPS code:

```
1)  addi $t0 $t1 100
2)  lw $t2 4($t0)
3)  add $t3 $t1 $t2
4)  sw $t3 8($t0)
5)  lw $t5 0($t6)
```

(4 points) For each instruction, list the instructions which are read-after-write dependant on that instruction.

Instruction 2 and 4 are dependant on instruction 1

Instruction 3 is dependant on instruction 2

Instruction 4 is dependant on instruction 3

Instruction 5 is dependant on instruction 4

(5 points) For the code, diagram what is happening in each pipeline stage as the instructions are executed. Show where forwarding occurs in the pipeline.

```
                       +---+---+---+---+---+
1)  addi $t0 $t1 100   |IF |ID |EX |MEM|WB |
                       +---+---+---+---+---+---+
2)  lw $t2 4($t0)          |IF |ID |EX |MEM|WB | (Gets $t0 from above)
                           +---+---+---+---+---+---+---+
3)  add $t3 $t1 $t2            |IF |ID |xxx|EX |MEM|WB | (Gets $t1)
                               +---+---+---+---+---+---+---+
4)  sw $t3 8($t0)                  |IF |xxx|ID |EX |MEM|WB | (Get $t3)
                                   +---+---+---+---+---+---+---+
5)  lw $t5 0($t6)                          |IF |ID |EX |MEM|WB |
                                           +---+---+---+---+---+
```

Your name ____ login cs61c–_______

Question 7 (9 points) Network and System call stuff:

(2 points): A network device receives a packet by having a small buffer capable of holding a single packet. When a packet is received completely it sends an interrupt, allowing the operating system to copy the packet out of the network device's buffer. If the minimum packet size is 1kB, and the maximum packet size is 8kB, and the network bandwidth is 1MB/s, what is the maximum number of interrupts/second for the network which the operating system needs to handle?

1K (1000 or 1024 depending on your definition) interrupts, as you can get 1k packets of size 1k in a second.

(2 points): A network has a latency of 100ms, and a bandwidth of 10MB/s. How l ong will it take to transmit 1 byte across the network?

100 ms

(2 points): How long will it take to transmit 100 MB of data?

10 seconds

(1 points): What sort of protocol is TCP?

Reliable, in order connection based. (the big ones to get are reliable and in order)

(2 points): For what sort of application would you rather use an **unreliable** protocol like UDP instead of a reliable protocol like TCP?

Games, video, or other applications where you want current data, and would rather just not see dropped data than wait for it to arrive before moving on.