CS 61C Midterm #1 — July 9th, 1998

Your name _____

login cs61c–_____

This exam is worth 30 points, or 15% of your total course grade. The exam contains six 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 six 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.

When writing procedures, don't put in error checks. Assume that you will be given arguments of the correct type and specified format.

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. We will use truncate 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 late, I certify that I have not discussed the exam questions or answers with anyone who has knowlegde of the exam.

_____

0	/1
1	/5
2	/6
3	/6
4	/7
5	/5
total	/30

Your name _____ login cs61c–_____

Question 1 (5 points):

Consider the following 32 bit binary value `11111111111111111111111111111110`.

(a) Write this value out in hexadecimal.

(b) decimal, interpreting it as an unsigned value. Write this as the neares power of 2 and add or subtract the approprate offset. (EG, if you want to write 9, write $2^3 + 1$.)

(c) decimal, interpreting it as a sign/magnitude value. Write this as the nearest power of 2 and add or subtract the appropriate offset. (EG, if you want to write -9, write $-(2^3 + 1)$.)

(d) decimal, interpreting it as a ones complement signed value. Write this as the nearest power of 2 and add or subtract the appropriate offset.

(e) decimal, interpreting it as a twos complement signed value. Once again, write this as the nearset power of 2 and add or subtract the appropriate offset.

(f) What is this value if you interpret it as IEEE single precision floating point?

Your name _____ login cs61c–_____

Question 2 (6 points):

Something on MIPS encoding. A couple of parts.

(a) Encode the following MIPS instruction in its binary representation: `lui $20 0xf00d`

(b) Decode the following binary number as a MIPS instruction and give the equivelent MIPS assembly language statement:

`0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0`

(c) `li $Rd` *imm* is a pseudo instruction with a 32 bit immediate. Convert it to a series of actual MIPS instructions. For credit, you need to use the exact minimum of MIPS instructions. (You can use `high` to signify the upper 16 bits of the immediate, and `low` to signify the lower 16 bits.)

(d) The `addi` instruction uses a 16 bit immediate. What is the largest constant which can be added with the `addi` instruction. (HINT: The immediate is sign extended).

Your name _____ login cs61c–_____

Question 3 (6 points):

Answer the following short questions. Be sure to read the questions carefully before answering:

(a) True or false: For every 32 bit signed, two's complement number there exists a corresponding IEEE **double precision** floating point number.

(b) What is the difference between the `add` and `addu` instructions in MIPS?

(c) True or false: Two's complement integer addition is not associative.

(d) Using your 1st grade math, add the following pairs of 8 bit unsigned numbers together

```
  10110101          10111111
+ 00101111        + 01101011
----------        ----------
```

(e) Using **saturating** arithmatic, add the following 8 bit signed, twos complement numbers together.

```
  00110101          00111111
+ 00101111        + 01101011
----------        ----------
```

Your name _____ login cs61c–_____

Question 4 (7 points):

Consider this C program definition:

```
int foo(int a){
  int i;
  int result = 1;
  for(i = 0; i < a; ++i){
    result = result + bar(i);
  }
  return result;
}
```

The arguments are passed using the register calling convention. Convert this C function into MIPS assembly language. You can assume that `bar` already exists, accepts a single integer as an argument, and returns an integer. All you need to do is call it properly in order to use the function `bar`. Please tell us which registers you use for storing the program variables, to make grading easier.

```
        # My implementation uses $___ for storing a
        # My implementation uses $___ for storing i
        # My implementation uses $___ for storing result
  foo:
```

Your name _____ login cs61c–_____

Question 5 (5 points):

Write a MIPS function `div4` which accepts a single argument which is an IEEE double precision floating point number in `$a0` and `$a1` (with the most significant bits in `$a0`), divides it by 4, and returns that value **without using any floating point instructions**. You do not need to and **should not include** code to handle underflow, subnormal values, $\pm\infty$, or NaN. (Remember, IEEE double precision floating point has an 11 bit exponent with a bias of 1023 and a 52 bit significand).

`div4`: