e) I’m getting the message “cannot execute binary file”. In one or two sentences, explain what the problem is and how to fix it.

f) In our 32-bit single-precision floating point representation, we decide to convert one significand bit to an exponent bit. How many denormalized numbers do we have relative to before? (Circle one)

More
Fewer

Rounded to the nearest power of 2, how many denorm numbers are there in our new format? (Answer in IEC format)

_____________

Question 2: Flippin’ Fo’ Fun (10 points, 14 minutes)

Assume that the most significant bit (MSB) of \( x \) is a 0. We store the result of flipping \( x \)’s bits into \( y \). Interpreted in the following number representations, how large is the magnitude of \( y \) relative to the magnitude of \( x \)? Circle ONE choice per row.

|                      | \(| y | < | x |\) | \(| y | = | x |\) | \(| y | > | x |\) | Can’t Tell |
|----------------------|-----------------|-----------------|-----------------|-------------|
| Unsigned             |                 |                 |                 |             |
| One’s Complement     | \(| y | < | x |\) | \(| y | = | x |\) | \(| y | > | x |\) | Can’t Tell |
| Two’s Complement     | \(| y | < | x |\) | \(| y | = | x |\) | \(| y | > | x |\) | Can’t Tell |
| Sign and Magnitude   | \(| y | < | x |\) | \(| y | = | x |\) | \(| y | > | x |\) | Can’t Tell |
| Biased Notation      | \(| y | < | x |\) | \(| y | = | x |\) | \(| y | > | x |\) | Can’t Tell |
| (e.g. FP exponent)   |                 |                 |                 |             |
1) What is that funky smell? Oh, it’s just Potpourri... (48 min, 30 pts)

Suppose for questions (a)-(d) that we were to modify the MIPS ISA so that it exposed 64 registers instead of 32, and adjusted the field widths of our R, I and J instruction formats to be able to address all the registers, but did not change the size of the opcode or shamt fields. Registers and instructions will remain 4 bytes wide.

a) At most how many instructions can a single beq instruction now reach? ____________________

b) How many more addresses can now be reached from a jal instruction? ____________________

c) How many different R-type instructions can we now have? ________________________________

d) If it costs 256 cents per bit of register memory (yikes!), how much would these 64 registers cost, in cents? Answer using IEC notation (e.g., 4 mebicents). _______________

Questions (e)-(g) concern the IEEE floating point standard.

e) What float is encoded by the following bits: 0xc14c0000? ______________ (show all work here)

f) What is the smallest positive normalized number? Number: _______ encoded as 0x________ (show all work here)

g) Write the MAL MIPS function IsNotInfinity to return non-0 if the input is NOT ±∞, 0 if it is ±∞.

<table>
<thead>
<tr>
<th>IsNotInfinity: movl %edi, %eax</th>
</tr>
</thead>
<tbody>
<tr>
<td>_____ $1, %eax # make +/- Inf look the same</td>
</tr>
<tr>
<td>_____ __________________________</td>
</tr>
<tr>
<td>ret</td>
</tr>
</tbody>
</table>
M3) What is that Funky Smell? Oh, it’s just Potpourri… (10 pts, 20 mins)

a) How many non-negative floats are < 2 ? _______ (you must show your work above for credit)

b) What’s the biggest change to the PC as the result of a jump on a 32-bit MIPS system? ________________
(answer in IEC format, like 16 kibibytes or 128 gibibytes)

c) Fellow 61C student Ben Bitdiddle was told to write a function count_az that takes an input string of lower-case letters (only 'a' through 'z') and returns an array of the number of occurrences of all letters; a histogram if you will. The returned array will be zero-indexed and the indices will correspond to their respective order in the alphabet (i.e. a = 0, b = 1, ..., z = 25). E.g., if the input str is “baaad”, the output array will look like the right column of the table on the right. Fix all the errors; we should be able to call it like this: myAZ = count_az(str); yourAZ = count_az(str);

```c
1 int count_az(char *str) {
2     int count[26];        // Create the count array
3     while(*str) {        // Go through the whole string
4         int index = &str - 0x97;    // The 97 is from the MIPS green sheet...
5         count[index]++;    // Increment the appropriate bucket
6         str++;           // Go to the next character
7     }
8  }
9 free(str);                // Free the string storage
```

<table>
<thead>
<tr>
<th>index</th>
<th>count[index]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 for 'a'</td>
<td>3</td>
</tr>
<tr>
<td>1 for 'b'</td>
<td>1</td>
</tr>
<tr>
<td>2 for 'c'</td>
<td>0</td>
</tr>
<tr>
<td>3 for 'd'</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>25 for 'z'</td>
<td>0</td>
</tr>
</tbody>
</table>

Line # | Add Change Remove | Additions / Changes / Removals |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Question 4: Let Me Float This Idea By You** (9 Points, 16 Minutes)

For a very simple household appliance like a thermostat, a more minimalistic microprocessor is desired to reduce power consumption and hardware costs. We have selected a **16-bit** microprocessor that does not have a floating-point unit, so there is no native support for floating point operations (no `float/double`). However, we’d still like to represent decimals for our temperature reading so we’re going to implement floating point operations in software (in C).

a) Define a new variable type called `fp`:

```plaintext
_______________________________________
```

We have decided to use a representation with a **5-bit exponent field** while following all of the representation conventions from the MIPS 32-bit floating point numbers **except denoms**.

Fill in the following functions. Not all blanks need to be used. You can call these functions and assume proper behavior regardless of your implementation. Assume our hardware implements the C operator “>>” as *shift right arithmetic*.

b)

```plaintext
/* returns -num */
fp negateFP(fp num) {
    return _____________________________________________________________;
}
```

c)

```plaintext
/* returns the signed value of the exponent */
int getExp(fp num) {
    _____________________________________________________________
    return _____________________________________________________________;
}
```

d)

```plaintext
/* multiplies floating point num by 2^n, while detecting over/underflow */
/* remember, there are no denoms */
fp multPow2(fp num, int n) {
    _____________________________________________________________
    if(____________________________________________) exit(1);  #overflow
    if(____________________________________________) exit(-1); #underflow
    _____________________________________________________________
    return _____________________________________________________________;
}
```
Question 5: Floating Point (10 pts)

Assume integers and IEEE 754 single precision floating point are 32 bits wide.

a) Convert from IEEE 754 to decimal: 0xC0900000

b) What is the smallest positive integer that is a power of 2 that can be represented in IEEE 754 but not as a signed int? You may leave your answer as a power of 2.

c) What is the smallest positive integer x such that x + 0.25 can’t be represented? You may leave your answer as a power of 2.

d) We have the following word of data: 0xFFC00000. Circle the number representation below that results in the most negative number.

<table>
<thead>
<tr>
<th>Unsigned Integer</th>
<th>Two’s Complement</th>
<th>Floating Point</th>
</tr>
</thead>
</table>


e) If we decide to stray away from IEEE 754 format by making our Exponent field 10 bits wide and our Mantissa field 21 bits wide. This gives us (circle one):

MORE PRECISION // LESS PRECISION

Question 6: Performance (4 pts)

We are using a processor with a clock period of 1 ns.

a) Program A contains 1000 instructions with a CPI of 1.2. What is the CPU time spent executing program A?

b) Program B contains 500 instructions but accesses memory more frequently, what is the maximum CPI that program B can have without executing slower than program A?
Question 1: Number Representation (8 pts)

a) Convert $0x1A$ into base 6. Don’t forget to indicate what base your answer is in!

b) In IEEE 754 floating point, how many numbers can we represent in the interval $[10,16)$? You may leave your answer in powers of 2.

c) If we use 7 Exponent bits, a denorm exponent of -62, and 24 Mantissa bits in floating point, what is the largest positive power of 2 that we can multiply with 1 to get underflow?

Local phone numbers in the USA typically have 7 decimal digits, which use the symbols 0 to 9. For example, Jenny Tutone’s phone number is:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Line Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>867</td>
<td>5309</td>
</tr>
</tbody>
</table>

d) How many unique phone numbers can be encoded by this scheme?

e) How many bits would we need to represent a phone number if we treated it as a single 7-digit decimal? You may use $\log()$ and $\text{ceil}()$ in your answer and the variable $E$ to represent the correct answer to part (d).