CSM 61C Spring 2020 — Floating Point

Exam Question Compilation Solutions

This document is a PDF version of old exam questions by topic, ordered from least difficulty to greatest difficulty.

Questions:

- Fall 2019 Midterm Q1
- Spring 2018 Final Q9
- Summer 2019 Midterm 2 Q1
- Summer 2018 Final Q4
- Summer 2019 Final Q1.7-Q1.11
- Fall 2018 Final M2 Part 1
- Fall 2017 Final Q9
- Summer 2018 Midterm 1 Q1

Fall 2019 Midterm Solution

Q1) *Float, float on...* **(7 pts = 2 + 3 + 2)**

You notice that floats can generally represent much larger numbers than integers, and decide to make a modified RISC instruction format in which all immediates for jump instructions are treated as 12-bit floating point numbers with a mantissa of 7 bits and with a standard exponent bias of 7. *Hint: Refer to reference sheet for the floating point formula if you've forgotten it...the same ideas hold even though this is only a 12-bit float…*

a) To jump the farthest, you set the float to be the most positive (not ∞) integer representable. *What are those 12 bits* (in hex)?	b) What is the *value* of that float (in decimal)?	c) Between 0 and (b)'s answer (inclusive), *how many integers are not representable*?
0x77F	255	0

Part A

Since you want the most positive float, the **sign bit** should be 0. For the **exponent**, you want the second biggest possible exponent, as the biggest possible exponent is always used for NaN and infinity. With 4 exponent bits, the largest possible number is 1111 or 15, so the second largest is 1110, or 14 (as an unsigned number; with the bias, this becomes 2^7).

With an exponent of 2^7 and 7 mantissa bits, when you multiply 1.<mantissa> by the exponent, the decimal point will end up right after the last mantissa bit. This means *every* number representable with an exponent of 2^7 is an integer, so you just need the largest one (which will be when you have all 1's). Putting the three parts together, 0 -- 1110 --1111111 is 0x77F.

Part B

The value of the float above will be $(-1)^0 * 2^7 * 1.1111111 = 11111111.0$ in binary, or 255.

Part C

For an exponent of 2^7 with 7 mantissa bits you have a step size of $2^{-7} * 2^7 = 2^0 = 1$. This means every integer between 0 and 255 is representable.

Problem 9 ***[F-1] Floating Point*** **(13 points)**

IEEE 754-2008 introduces half precision, which is a binary floating-point representation that uses 16 bits: 1 sign bit, 5 exponent bits (with a bias of 15) and 10 significand bits. This format uses the same rules for special numbers that IEEE754 uses. Considering this half-precision floating point format, answer the following questions:

(a) For 16-bit half-precision floating point, how many different valid representations are there for NaN?

> **Solution:** $2^{11} - 2$

(b) What is the smallest positive non-zero number it can represent? You can leave your answer as an expression.

> **Solution:** bias $= 2^{5-1} - 1 = 2^4 - 1 = 15$
> Binary representation is: 0 00000 0000000001
> $= 2^{-14} * 2^{-10} = 2^{-24}$

(c) What is the largest non-infinite number it can represent? You can leave your answer as an expression.

> **Solution:** Binary representation is: 0 11110 1111111111
> $= 2^{16} - 2^5 = 65504$

(d) How many floating point numbers are in the interval [1, 2) (including 1 but excluding 2)?

> **Solution:** 2^{10}
> 0b0 01111 0000000000 = 1
> 0b0 10000 0000000000 = 2
> There are 2^{10} numbers within the interval.

Summer 2019 Midterm 2 Solution

Question 1: Floating *Points to your Cheat Sheets - 10 pts

For all of the following questions we are using the IEEE 754 single precision floating point from lecture. If you do not remember the details, some can be found on the back side of the green sheet.

1. Represent **14.75** in its floating point representation. Put your answer in hexadecimal.

sign = 0
14.75 = 1110.11 = 1.11011 * 2^3
significand = 1101100....0
exp - bias = 3 -> exp - 127 = 3 - > exp = 130 = 0b10000010

0x416C0000

2. Represent **-2^{-147}** in its floating point representation. Put your answer in hexadecimal.

sign = 1
$2^{-147} < 1 * 2^{1 - 127}$, so we are working with a denormalized number
exp = 0
denorm fomula = -1^{sign} * 0.significand * 2^{-126}
$2^{-147} = 2^{-21} * 2^{-126}$
significand = 2^{-21} = 0b000..0100

0x80000004

3. What value is represented by 0xFF800001?

NaN

For the remaining questions we are going to consider 2 possible changes:

- **Option 1**: Adding a bit to signficand and removing a bit from the exponent
- **Option 2**: Adding a bit to the exponent and removing a bit from the significand

For each of the following questions select whether **option 1, option 2, neither,** or **both** will accomplish the presented task. Assume that the bias also shifts to be $\mathbf{2^{exp\_bits - 1} - 1}$.

4. Represent pi more accurately than our IEEE 754 single precision floating point.

[✓ Ⓐ Option 1] Ⓑ Option 2 Ⓒ Neither Ⓓ Both

More significand bits leads to more digits of pi.

5. Represent smaller positive numbers than IEEE 754 single precision floating point.

Ⓐ Option 1 [✓ Ⓑ Option 2] Ⓒ Neither Ⓓ Both

smallest denorm number is always $(2^{-num\_significand\_bits}) * 2^{-bias + 1}$

Because the bias of option 2 is roughly twice as big as IEEE 754 our smallest number is much smaller.

6. Represent more numbers in the range [1, 2) than IEEE 754 single precision floating point.

[✓ Ⓐ Option 1] Ⓑ Option 2 Ⓒ Neither Ⓓ Both

There are $2^{num\_significand\_bits}$ numbers in the range [1, 2). As a result option 1 has the most numbers.

7. Represent more numbers than IEEE 754 single precision floating point.

Ⓐ Option 1 [✓ Ⓑ Option 2] Ⓒ Neither Ⓓ Both

This one is tricky. All values represented are numbers except for NaN. With 32 bits we represent 2^{32} values, so now we just need to remove the NaN. For every scheme this is the number of values at the largest exponent minus positive and negative infinity or:

$2^{num\_significand\_bits} - 2$

So we can represent $2^{32} - 2^{num\_significand\_bits} + 2$ numbers in any floating point scheme. Since Option 2 has the fewest exponent bits it represents the most numbers.

Summer 2018 Final Solution

SID: ________________

Question 4: loating Point (9 pts)

Being the feisty floating point fanatic you are, you devise a new scheme to interpret 32-bit floats. You keep the breakdown of the Sign/Exponent/Significand fields the same. However, there is no implicit 1. or 0. before the significand bits; also, you evaluate the 23 significand bits as an **unsigned number**, which you then multiply with $2^{Exp - Bias}$ and $(-1)^{Sign}$ as you normally would.

$$(-1)^{Sign} \times Significand_{unsigned} \times 2^{Exp - Bias}$$

where $Bias$ = 127.
There are no denormalized numbers.

Exponent	Significand	Value
Largest	Zero	±Infinity
Largest	Nonzero	NaN

We walk through one way of representing 3 in this scheme, using a Significand of 0b00...011 = 3
$(-1)^0 \times 3 \times 2^{127 - Bias} = 1 \times 3 \times 2^0 = 3$ **Sign**: 0, **Exponent**: 127 = 0x7F, **Significand**: 3 = 0x000003

Answer the following questions comparing the standard IEEE Floating Point and your new scheme (let's call it Bloating Point).

1) Bloating Point represents a larger amount of unique numeric values than Floating Point. T [✓ F]

2) 0x3D80001E is a valid representation of 1.875 in Bloating Point. (Hint: 1.875 = 15/8 [✓ T] F

3) There exists a 32-bit number whose Floating Point value and Bloating Point value are the same. [✓ T] F

4) How many Bloating Point numbers evaluate to $+2^{-124}$?

$1 * 2^{-124}$	$4 * 2^{-126}$
$2 * 2^{-125}$	$8 * 2^{-127}$

4

5) What is the smallest positive number divisible by 5 that Bloating Point cannot represent? You may leave all or part of your answer as a power of 2.

The smallest positive Bloating Point number divisible by 5 involves a significand of ~ all 1s, since that is when we lose precision. We start our search with a significand of all 1s, which is 2^{23}- 1. The closest number above this number that is divisible by 5 is 2^{23}+2 (using modular arithmetic / finding a pattern). However, 2^{23}+2 can be represented in Bloating Point: $(2^{22} + 1) * 2^1$. Thus, we find the next number divisible by 5 is 2^{23}+7, which can't be represented in Bloating Point since it's not even.

$2^{23}+7$

You propose a new 16 bit floating point number. It has:

- 1 sign bit
- 11 exponent bits
- 4 significand bits
- A bias of 1023
- All other rules consistent with IEEE 754 floating point.

7. Represent 4.75 in our new floating point scheme

sign = 0
exp - bias = 2
exp = 2 + 1023 = 1025
significand = 0011

Sign:0b0

Exponent: 0b10000000001

Significand:0b 0011

8. How many numbers does our floating point scheme represent in the range [0, 1) (the range 0 to 1, where 0 is included and 1 is not)? For this question assume -0 is not in this interval. You may leave your answer unsimplified.

1 has an exponent value of 1023 because $2^0 = 1$. This means alls the values with a positive sign and exponent less than 1023 are in this range. Because there are 4 significand bits there are 16 numbers per exponent.

= 1023 * 16 = 16368

16368 numbers

Now let's compare to a 16 bit two's complement number.

9. Which can represent a larger number (ignore infinities)?

[✓ Ⓐ Our Floating Point Scheme] Ⓑ Two's Complement

Our largest number in two's compliment is $2^{15} - 1$. The largest number in floating point is 2^{1023} * (1.1111), which is clearly much larger.

10. Which scheme represents more numbers in the range [1, 64)?

[✓ Ⓐ Our Floating Point Scheme] Ⓑ Two's Complement

In two's complement there are exactly 64 numbers. For two's compliment we look at each power of 2 (an exponent value) and because there are 4 significand bits there will be 16 per power of 2. Between 1 and 64 there are 6 powers of 2. 6 * 16 = 96.

11. Which scheme represents more numbers in the range [64, 128)?

Ⓐ Our Floating Point Scheme [✓ Ⓑ Two's Complement]

Our floating point scheme only represents 16 numbers (each power of 2 in the range has 16 significand values) while two's complement represents 64.

Fall 2018 Final Solution

M2) *Floating down the C...* [this is a 2-page question] (8 points = 1,1,2,1,1,1,1, 20 minutes)

Consider an 8-bit "minifloat" S**E**EEMMMM (1 sign bit, 3 exponent bits, 4 mantissa bits). All other properties of IEEE754 apply (bias, denormalized numbers, ∞, NaNs, etc). The bias is -3.

a) How many minifloats are there in the range [1, 4)? (i.e., 1 ≤ `f` < 4) **32**

Bias of -3 means the exponent can go from -3 to 4 → to 2^3 so we are in range. 1 and 4 are powers of 2, so that's two "ranges", and with MMMM = 16 mantissa values, that's **32** mantissa values.

b) What is the number line distance between 1 and the smallest minifloat bigger than 1? **1/16**

1 is a special number since the exponent is 0 (after the bias is applied), thus it's 2^0 * 1.MMMM → 1.MMMM (the binary shift left/right by $2^{\text{EXP-Bias}}$ goes away) → the least M is counting by **1/16** so the next number after 1.0000_2 is 1.0001_2 which is 1+1/16.

c) Write **`times2`** in one line using integer operations. Assume the leftmost "E" bit of `f` (bolded above) is 0.

```
minifloat times2(minifloat f) { return f * 2.0; }
```

```
times2: addi a0, a0, 0b00010000 = 0x10 ## Assume f is in the lowest byte of the register

    Ret
```

We have to add one to the exponent to make it work,cool!

Q9: Floating Point

Some of the 61C TAs get tired of having to convert floating-point values into 32 bits. As a result they propose the following smaller floating-point representation.
It consists of a total of 12 bits as shown below:

Sign (1)	Exponent (4)	significand (7 bits)
0	1 4	5 11

- The largest exp remains reserved as in traditional floating point
- The smallest exp follows the same denormalized formula as traditional floating point
- **Numbers are rounded to the closest representable value. Any numbers that have 2 equidistant representations are rounded down towards zero.**

All of the parts of this question refer to this floating-point scheme.

1. What is the smallest nonzero positive value that can be represented? Write your answer as a numerical expression in the answer packet.

$$2^{-13}$$

2. Consider some **positive normalized** floating-point number p where p is described as p = 2^y * 1.significand. What is the distance (i.e. the difference) between p and the next-largest number after p that can be represented? Write your answer as a numerical expression.

$$2^{y-7}$$

3. Now instead let p be a positive denormalized number described as p = 2^y * 0.significand. What is the distance between p and the next-largest number after p that can be represented?

$$2^{-13}$$

Summer 2018 Midterm 1 Solution

Question 1: Number Representation and Floating Point (12 pts)

Given the following bit string 0b1111 1100, answer the following questions:

1) What is this bitstring's value if it was interpreted as an **unsigned number**?

(2^8 - 1) - 3 = 252

2) What is this bitstring's value if it was interpreted in **two's complement**?

(Flip all the bits and add 1) * -1 = (0b0000 0011 + 1) * -1 = -4

3) Suppose the bit string was represented as **fixed point** where the bits following the dot (.) represent the base (2) to the power of a negative exponent. What number does the bitstring 0b1111.1100 represent?

0b1111 = 15; 0b.1100 = 0.5 + 0.25 = 0.75 -> 15.75

4) Now let's devise a scheme for interpreting fixed point numbers as positive or negative values. Complete the following sentence:

Given an 8-bit fixed point bitstring 0bXXXX.XXXX with a value of Y, in order to compute -Y, we must flip all the bits of Y and add:

The idea in two's complement is that you add 0b00...001 to the flipped bitstring.
0b0000.0001 in fixed point has a value of 1/16. 1/16

5) What is the value of 0b1111.1100 given the **two's complement fixed point** representation described above?

. (Flip all the bits and add 1/16) * -1 = (0b0000.0011 + 1/16) * -1 = -0.25

6) You are given the following field breakdown and specifications of an 8-bit floating point, which **follows the same rules** as standard 32-bit IEEE floats, except with different field lengths:

Sign: 1 bit
Exponent: 3 bits
Significand: 4 bits

Exponent Value	Significand Value	Floating Point Value
Smallest	Zero, Non-Zero	±0, Denormalized
Largest	Zero, Non-zero	±Infinity, NaN

What is the floating point value of 0b1111 1100:

Exponent field is the largest exponent (0b111) and the significant is non-zero-> NaN

7) We now modify the floating point description in part 6 so that the exponent field is now in **two's complement** instead of in bias notation. Compute the floating point value of 0b1111 1100.

Exponent: 0b111 = -1, Signif: 0.75 = (-1) x 2^-1 x 1.75 = -0.875