CSM 61C Spring 2020

Number Representation

Exam Question Compilation

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

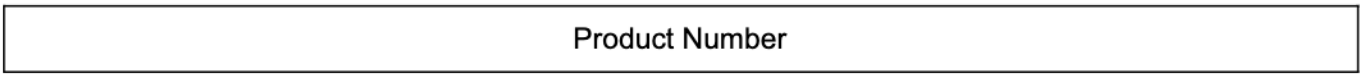
Questions:

- Summer 2019 Midterm 1 Q1
- Fall 2019 Quest Q1, Q2, Q3
- Spring 2015 Final M1-1AB
- Fall 2018 Quest Q1
- Spring 2018 Midterm 1 Q1
- Spring 2018 Final Q1A, Q1B
- Summer 2018 Midterm 1 Q1
- Fall 2018 Midterm Q1 A-D
- Fall 2015 Final MT2-5
- Fall 2019 Final Q2

Question 1: We're bored of Euclid. (Number Representation) - 17 pts

Morgan, Nick, and Branden are disappointed in the selection of food available near Soda, so they open a store selling many different kinds of products. They need YOUR help to come up with a barcode scheme for everything they sell.

1. Branden wants to assign each product in the store its own unique number. This will be encoded as the barcode. If they sell 29 unique items, what is the smallest number of bits they can use to encode the barcode?



`Barcode = _____ bits`

2. When the store runs out of a particular item, it would be helpful to see what other kinds of that item there are in stock. Morgan proposes adding a "product group" field to the barcode in addition to the existing product number. Note that now each product number does not need to be globally unique and instead just needs to be unique within its product group. If there are 5 unique product groups, what is the smallest number of bits they can use for the product group field?

`Product Group = _____ bits`

3. We expand to have 12 product groups. The largest has 15 items in it while the smallest has one item. Nick argues the entire barcode can now be condensed to only 6 bits without losing product grouping or unique identifiers. Is he correct? If yes, explain why, if no, what is the actual minimum size?

   [ ] Yes, he is correct [ ] No, he is incorrect

   _____

   _____

   _____

4. The team decides on the following barcode field sizes (which may or may not reflect your answers above).

Product Group (4 bits)	Product Number (5 bits)

   Morgan loads all the barcodes into the database but runs into a problem; she'd like to reserve the barcode of all zeros (so, product group = 00...0, product number = 00...0) for products that are out of stock. Assuming there are 8 product groups holding between 1 and 31 products each, can she implement the all-zero barcode without adding bits to the scheme? Explain.

   [ ] Yes, she can [ ] No, she can't

   _____

   _____

   _____

5. Business is booming and the team has the opportunity to expand! They purchase a new store and modify the barcode to keep track of products sold at store 0 and store 1 separately.

Store Code (1 bit)	Product Group (4 bits)	Product Number (5 bits)

   Assuming the same item sold across stores differs ONLY in the top bit (that is, they have the same product group and product number regardless of the store they're sold at) what is the maximum number of items Morgan, Nick, and Branden can uniquely identify with this barcode scheme? You may leave your answer as an unsimplified equation.

   _____ Unique Items

_____ *Your Name (first last)*

_____ *SID*

UC Berkeley
Fall 2019
CS61C Quest

_____ *← Name of person on left (or aisle)*

_____ *Name of person on right (or aisle) →*

Q1) [10 Points] **Negate** the following **nibble *binary/hex*** numbers, or write N/A if not possible. Remember to write your answer in the appropriate base. (A nibble is 4 bits)

(Unsigned) **0b0101**	*(Bias = -7)* **0b0100**	*(Bias = -7)* **0xF**	*(Two's Comp)* **0b1100**	*(Two's Comp)* **0xA**
0b	**0b**	**0x**	**0b**	**0x**

...scratch space below...

Q2) [6 Points] Which of the following sums will yield an **arithmetically incorrect result** when computed with **two's complement nibbles**?

Correct ◯ Incorrect ◯	Correct ◯ Incorrect ◯	Correct ◯ Incorrect ◯
0xD + 0xE + 0xF	**0x7 + 0x8**	**0x3 + 0x5**

...scratch space below...

Q3) [12 Points] For each of the following representations, what is the *fewest number of bits* needed to cover the given range, which is inclusive of the endpoints (e.g., [1, 4] is the numbers 1, 2, 3 and 4). Write "N/A" if it is impossible. For the **Bias *Value*** (final value = unsigned + bias value), we'll let YOU specify whatever offset you wish to minimize the total number of bits needed for the Bias encoding.

Range	Unsigned	One's Comp	Two's Comp	Sign&Mag	Bias	Bias *Value*
[0, 10]		5	5			0
[-4, -1]		4				
[1, 4]				4	2	

...scratch space below...

For this page, assume all `mallocs` are successful, all necessary libraries are `#included`, and any heap accesses outside what the program allocates is a segmentation fault.

Q4) [12 Points] Which of the following are possible, if perhaps unlikely, results of attempting to compile and run this code? (select ALL that apply)

```
int main() {
    int32_t *str = (int32_t *) malloc(sizeof(int32_t) * 3);
    printf("%s", (char *) str); // A char is 8 bits.
    return 0;
}
```

- ▢ Compilation error due to invalid typecast
- ▢ Runtime typecasting error
- ▢ A segmentation fault
- ▢ The program prints the empty string
- ▢ The program prints **`CS61C`**
- ▢ The program prints **`CS61C rocks!`**

Q5) [10 Points] Each of the following evaluate to an address in memory. In other words, they "point" somewhere. Where in memory do they **point**?

	Code	*Static*	*Stack*	*Heap*
`arr`	○	○	○	○
`arr[0]`	○	○	○	○
`dest`	○	○	○	○
`dest[0]`	○	○	○	○
`&arrPtr`	○	○	○	○

Q6) [10 Points] The program below runs through the array of strings, doing something to each of the characters and putting the results in the `dest` array.

What are the first 8 characters the program prints? (Note: The program DOES compile and run without error.)

_ _ _ _ _ _ _ _

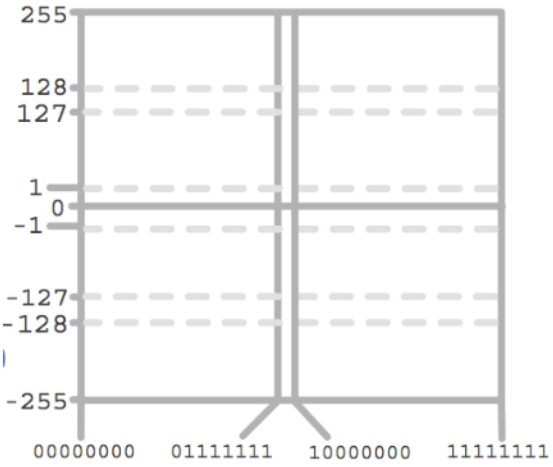
```
// The ASCII values for 'A', 'B', etc. are 65, 66, ... ~~!!!!!~~ Important
// The ASCII values for 'a', 'b', etc. are 97, 98, ... ~~!!!!!~~ Important

char *arr[] = {"Go", "Bears"};

int main() {
   char **arrPtr = arr;
   char *dest[2];
   int j;

   for (int i = 0; i < 2; i++) {
       char *currString = *arrPtr;
       dest[i] = (char *) malloc(strlen(currString) + 1);
       for (j = 0; j < strlen(currString); j++) {
           dest[i][j] = currString[j] & ~(1 << 5); // Hint: Focus on this line!
       }
       dest[i][j] = '\0';
       arrPtr++;
   }
   printf("%s %s", dest[0], dest[1]);
}
```

M1-1: *I smell a potpourri section covering midterm one…* (9 points)

a) Which of the following number representations give **`0xFFFFFFFE`** the **most positive** value when converted to decimal?

A) Bias (with standard bias) B) Unsigned C) Two's complement D) Sign and Magnitude

b) Consider a plot that shows the mapping between 8-bit two's complement binary numbers and their decimal equivalents (i.e. binary is on the x-axis and decimal is on the y-axis). Fill in the plot to the left and answer the following questions.



i) Fill in the plot to the left.
ii) Describe (in binary) where discontinuities occur in the plot, if any:

iii) What are the most positive and most negative decimal values that this representation can store?

Q1a) With **3 bits**, how do we represent **-2**? If it can't be done, select "**N/A**". (Select ONE per row)

	000	001	010	011	100	101	110	111	N/A
Unsigned	○	○	○	○	○	○	○	○	○
Sign/Magnitude	○	○	○	○	○	○	○	○	○
One's Complement	○	○	○	○	○	○	○	○	○
Two's Complement	○	○	○	○	○	○	○	○	○
Bias; use bias of $-(2^{N-1}-1)$ from lecture	○	○	○	○	○	○	○	○	○

...scratch space below...

Q1b) Convert 26_{10} to the following bases (and remove any leading zeros)

Binary	*Hex*
0b	**0x**

Q1c) Add these *Two's Complement* nibbles:

1001 + 1011	Does it overflow a nibble? (Select ONE) ○ Yes ○ No

Problem 1 ***Number Representation***

(a) Translate the following decimal numbers into 8-bit two's complement and unsigned binary representation in the table below. If a translation is not possible, please write "N/A". **Write your final answer in hexadecimal format.**

Decimal Number	Two's Complement	Unsigned Number
10	0x	0x
129	0x	0x
-12	0x	0x

(b) Suppose that we define the negative of x to be just $\overline{x}$. We will call this new number representation scheme **one's complement**. Note that the top bit of a one's complement number still denotes the number's sign (0 for positive, 1 for negative).

Translate the following decimal numbers into 8-bit one's complement binary representation. If the translation is not possible, please write "N/A". **Write your final answer in hexadecimal format.**

Decimal Number	One's Complement
13	0x
-6	0x

(c) What is the range of integers (in decimal format) that we can represent with an n-bit one's complement binary number?

_____

Problem 1 ***[MT1-1] Number Rep***

Answer the following questions about number representation:

(a) Unsigned Base 4

(i) What is the range that a 4 digit unsigned base 4 number can represent? Write the bounds in decimal.

[_____, _____]

(ii) Convert 107_{10} to unsigned base 4.

_____

(b) Signed Base 4

(i) Suppose we wanted to use a bias in order to represent negative numbers in base4. If we are working with a 4 digit base 4 number, what should we choose as our bias? (Our bias should create equal amounts of negative and positive numbers for our range. If this is not possible, select a bias that will result in 1 more negative number than positive numbers). Express your answer in decimal.

Bias = **-** _____

(ii) Suppose rather than using a bias notation, we decide to do the following.

For each base 4 number, we will reserve the most significant digit to strictly be used as a sign bit. A digit value of 1 will indicate a negative number, and a digit value of 0 will indicate a positive number. Any other values will result in an invalid number. For instance:

$$0003_4 = +3 \qquad 1003_4 = -3 \qquad 2003_4 = Invalid$$

How many valid numbers can we represent with a 4 digit base 4 number using this scheme?

_____

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

_____

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?

_____

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:

_____

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

_____

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:

_____

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.

_____

Q1) *Float, float on…* (6 points; a,b 1pt; c,d 2pts)

Consider an 8-bit "minifloat" SEEEEEMM (1 sign bit, 5 exponent bits, 2 mantissa bits). All other properties of IEEE754 apply (bias, denormalized numbers, ∞, NaNs, etc). The bias is -15.

a) How many NaNs do we have? _____

b) What is the bit representation (in hex) of the next minifloat bigger than the minifloat represented by the hexadecimal value is **`0x3F`**? _____

c) What is the bit representation (in hex) of the encoding of -2.5? _____

d) What does **`should_be_a_billion()`** return? *(assume we always round down to 0)* _____

```
minifloat should_be_a_billion() {
      minifloat sum = 0.0;
      for (unsigned int i = 0; i < 1000000000; i++) { sum = sum + 1.0; }
      return sum;
}
```

MT2-5: What is the *floating point* of *complex* numbers? (5 points)

Your friend needs your help to make a new complex number representation. As a refresher, complex numbers are in the form $a + bi$, where a is the real component, b is the imaginary component, and the magnitude is $\sqrt{a^2 + b^2}$.

You decide to create a 16-bit representation for storing both the real and imaginary components as floating point numbers with the following form: The first 8 bits will represent the real component, and the latter 8 bits will represent the complex component. Your new representation looks like this:

Sign	Exponent	Significand	Sign	Exponent	Significand
15	14-12	11-8	7	6-4	3-0

Bits per field:
Sign: 1
Exponent: 3
Significand: 4
Everything else follows the IEEE standard 754 for floating point, except in 16 bits
Bias: 3

a. Convert 0xB248 into a complex number.

b. What is the smallest positive number you can represent with a nonzero real component and zero complex component.

Recall the following floating point representation from the midterm:

Sign	Exponent	Significand
15	14-9	8-0

Bits per field:
Sign: 1
Exponent: 6
Significand: 9
Everything else follows the IEEE standard 754 for floating point, except in 16 bits

c. Ignoring infinities, which of the two representations presented above can represent a number with the larger magnitude.

Complex Number Representation Midterm Representation

Q2) Open to Interpretation **(11 pts = 2 + 3 + 4 + 2)**

Let's consider the hexadecimal value **`0xFF000003`**. How is this data interpreted, if we treat this number as…

a) an array `A` of unsigned, 8-bit numbers? Please write each number in **decimal,** assume the machine is **big endian**, and write **`A[0]`** on the left, **`A[3]`** on the right.

_____, _____, _____, _____

SHOW YOUR WORK HERE

b) an IEEE-754 single-precision floating point number?

SHOW YOUR WORK

c) a RISC-V instruction? If there's an immediate, write it in decimal.

_____

SHOW YOUR WORK

d) a **`(uint32_t *)`** variable **`c`** in **little-endian** format, and we call **`printf((char *) &c)`**? If an error or undefined behavior occurs, write "Error". If nothing is printed, write "Blank". Please refer to the ASCII table provided on your reference sheet. For non-printable characters, please write the value in the Char column from the table. For example, for a single backspace character, you would write "**BS**".

_____

SHOW YOUR WORK