


You are viewing this thread in readonly mode.

## [Midterm] Past Exams - 2022 #468

 **Eric Che** STAFF 9 months ago in [Exam - Midterm](#) 967  
VIEWS




You can find the past exams here: <https://cs61c.org/sp24/resources/exams/>. Please check the [linked past Piazza/Ed Q&A PDFs](#) first before asking here. Many of the questions are already answered in those! [Video walkthroughs](#) are also available!

When posting questions, please reference the semester, exam, and question in this format so it's easier for students and staff to search for similar questions:

### Semester-Exam-Question Number

For example: **SP22-Final-Q1, SU22-MT-Q3, FA23-MT-Q1**

 **Anonymous Newt** 7mth #468baa ✓ Resolved

Sp24-mt-q4:

I'm wondering, when computing the significand, why they needed to left shift off all the bits to the left of the most significant 1 and then right shift back. Couldn't they have computed the correct number of bits first and then just left-shifted once? like, for the first three lines:

```
sub t1 t1 a0
```

```
sub t1 s4 9
```

```
sll s4 s4 t1
```

or is there a need to get rid of everything to the left? (i thought everything to the left was 0)

Implement `convert`, a RISC-V function, as follows:

- Input `a0`: a **nonzero** unsigned integer
- Returns in `a0`: the IEEE-754 single-precision floating point representation of `a0`, rounded towards zero if there is no exact representation.

For example, the integer 2 (0x00000002) should be converted to its corresponding IEEE-754 single-precision floating point representation 0x40000000. The integer 268435471 (0x1000000F) has no exact floating point representation; instead, the representation 0x4d800000 (rounded towards zero) should be returned.

You may assume that `magnitude` is implemented correctly and behaves as specified in the previous question, regardless of your implementation above. You may not assume any specific implementation of `magnitude` and you may not modify any `s` registers except for `s4`.

```
1 convert:
2     # prologue omitted

3     mv s4 a0
4     jal ra magnitude

5     addi t1 x0 32                                # set significand
6     sub t1 t1 a0
7     sll s4 s4 t1
8     srli s4 s4 9
9     addi a0 a0 127                                # set exponent
10    slli a0 a0 23
11    add a0 a0 s4
12    # epilogue omitted
13    jr ra
```

♡ ...



Justin Yokota STAFF 7mth #468bab

You don't necessarily know from the start if you need to left-shift the data, or right-shift the data; 0xC000 0000 would need a right-shift from its initial position, for example. Note that you can't shift by a negative shift amount, as the shift gets casted to an unsigned value.

By left-shifting, then right-shifting, we account for both cases without needing a branch.

♡ ...



Anonymous Stingray 8mth #468afa

✓ Resolved

Sp22 Q4.1:

Can we also use `self.x` and `self.y` instead of `->`? more importantly, what is the significant difference between the two, and under which context should we be using which of them? Thanks!

**Solution:**

```
7  Vector *transform(Vector *self, int (*f)(int)) {
8      Vector* newVector = malloc(sizeof(Vector));
9      newVector -> x = f(self->x);
10     newVector -> y = f(self->y);
11     return newVector;
12 }
```

♡ ...



E Eddy Byun STAFF 8mth #468afd

`self.x` and `self.y` do not work here. `self->x` is similar to `(*self).x`. Since `self` is a pointer to a `Vector` struct, we first need to dereference the pointer and then we can access the `x` field within that struct.

♡ ...



Anonymous Spoonbill 8mth #468aed

✓ Resolved

Sp22-Midterm-Q3.1

Why we are not converting the answer to two complements over here?

Note: we think this is the trickiest question on the exam.

Define statements can be useful, but it's important to be careful when using them.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define abs(x) ((x) < 0 ? -(x) : (x))
4 #define f(a,b) a*b/4
5 int main() {
6     int a = 10;
7     printf("Question 3.1: %d\n", a^2);
8     int i = 0xA6004F4E;
9
10    printf("Question 3.2: 0x%X\n", i|(i<<4));
11    printf("Question 3.3: 0x%X\n", abs(i));
12
13    int b = 10;
14    printf("Question 3.4: %d\n", f(0+1, b));
15    printf("Question 3.5: %d\n", f(1+0, b));
16
17    int k = 100;
18    int* kptr = &k;
19    printf("Question 3.6: %d\n", f(k+,kptr));
20    return 0;
21 }
```

♡ ...

M **Minh Nguyen** STAFF 8mth #468afe

We don't have to manually convert to two complements in C because the language automatically handles negative numbers with its operators :)

♡ ...

**Anonymous Seal** 8mth #468aea

✓ Resolved

Su22 Q4.1

How does `andi` check if the number is odd? If I put 0 instead of 1, does that check if it's even?

♡ ...

A **Andy Chen** STAFF 8mth #468aef

The `and` of a number with 1 is equal to that number's last bit. If a number's last bit is 1, then it is odd, and vice versa (if a number's last bit is 0, then it is even). The `and` of any number with 0 is always 0 (you can think about why this is the case), so replacing `andi t2 t1 1` with `andi t2 t1 0` would end up always putting 0 into `t2`. Here's a couple examples:

```
10 (even): 0000 0000 0000 0000 0000 0000 0000 1010
           1: 0000 0000 0000 0000 0000 0000 0000 0001
10 & 1: 0000 0000 0000 0000 0000 0000 0000 0000
```

```
63 (odd): 0000 0000 0000 0000 0000 0000 0011 1111
           1: 0000 0000 0000 0000 0000 0000 0000 0001
63 & 1: 0000 0000 0000 0000 0000 0000 0000 0001
```

♡ ...

**Anonymous Newt** 8mth #468ade

✓ Resolved

[Su22 Q2.1], could we say `(word + i) - 97`?

♡ ...

A **Andy Chen** STAFF 8mth #468aeb

Since `word` is a pointer to a `char (char*)` you could not, since `(word + i)` would then be a pointer, and then you would be doing pointer arithmetic when subtracting 97. But you could do `*(word + i) - 97` since `*(word + i)` gets the value at the pointer (effectively doing the same thing as `word[i]` )!

♡ ...

**Anonymous Manatee** 8mth #468add

✓ Resolved

SU22-MT-Q4.1


would `ret` be valid instead of `jr ra` for the last line? looking at the reference sheet it seems they do the same thing?

♡ ...

E **Eddy Byun** STAFF 8mth #468afb

Sure, `ret` is valid.

♡ ...

 **Anonymous Sandpiper** 8mth #468adc ✓ Resolved  
SU22-MT-Q5


I assume that we need to simplify the expression as much as possible in order to receive full credit? Is partial credit awarded if the expression is correct but not fully simplified?

♡ ...

 **E Eddy Byun** STAFF 8mth #468afc

Yes, you need to simplify the expression to get full credit.

♡ 1 ...

 **Anonymous Seal** 8mth #468ace ✓ Resolved  
Sp22 Mt Q1.8

What is the implicit bit? What other instructions contain this implicit bit?

Q1.8 (1.5 points) TRUE or FALSE: Branch instructions can represent a larger immediate value than I-type instructions.

TRUE

FALSE


**Solution:** True; Branch instructions encode 12 bits worth of immediate, but we include an implicit 0th index bit of 0, bringing up the immediate to be 13 bits. I-type instructions encode only 12 bits, without any implicit bits.

♡ ...

 **N Nikhil Kandkur** STAFF 8mth #468ada

The implicit bit is the index 0 bit that is default set to zero, since that instruction format does not ask for imm[0] bit. The instruction formats that do not ask for imm[0] contain the implicit bit, which are the B-type instructions and the J-type instructions.

♡ ...

 **Anonymous Spoonbill** 8mth #468acd ✓ Resolved  
SP22-MT-Q5


Is this in scope?

♡ ...

 **N Nikhil Kandkur** STAFF 8mth #468acf

Q5 focuses on SDS, which is in scope.

♡ ...

 **Anonymous Butterfly** 8mth #468aaf ✓ Resolved  
SP22-MT-Q2.5: Why do we do + 1 when calculating the exponent here? I thought that exponent (in bits) is equal to true exponent - bias + 1. Because exponent in bits is 0, I thought our true exponent would be -4 instead of -2.

Q2.5 (3 points) What is the smallest positive number that can be represented by this system?  
Express your answer as an odd integer multiplied by a power of 2.

**Solution:**

Smallest significand = 0b0001

Smallest exponent = 0b000 = 0 + (-3) + 1 = -2

$0.0001 \times 2^{-2} = 1 \times 2^{-6}$

For normalized floats:

$$\text{Value} = (-1)^{\text{Sign}} \times 2^{(\text{Exponent} - \text{Bias})} \times 1.\text{significand}_2$$

For denormalized floats:

$$\text{Value} = (-1)^{\text{Sign}} \times 2^{(\text{Exponent} - \text{Bias} + 1)} \times 0.\text{significand}_2$$

♡ ...

 **Anonymous Turtle** 8mth #468abb

how did we know to find the denorm value as well here?

♡ ...

 **A Andy Chen** STAFF 8mth #468abf

Denorm numbers are designed to be able to get really small numbers! The smallest number should always be a denorm number, since the "implicit 1" becomes an "implicit 0".

♡ ...

 **A Andy Chen** STAFF 8mth #468abe

Actually it's the other way around! The exponent in (unsigned) bits corresponds to the "Exponent" value in the equations. In this case, the bits are 0b000, so we would plug 0 into the equation:

$$2^{(\text{Exponent} + \text{Bias} + 1)} = 2^{(0 + (-3) + 1)} = 2^{-2} \text{ (Exponent part of the equation)}$$

For normalized floats:

$$\text{Value} = (-1)^{\text{Sign}} * 2^{\text{Exp} + \text{Bias}} * 1.\text{significand}_2$$

For denormalized floats:

$$\text{Value} = (-1)^{\text{Sign}} * 2^{\text{Exp} + \text{Bias} + 1} * 0.\text{significand}_2$$

(From Disc 2)

♡ ...

 **Anonymous Loris** 8mth #468aae ✓ Resolved

Su22-MT-Q4.2: how can we know the immediate value for loop for translation?

♡ ...

 **A Andy Chen** STAFF 8mth #468aca

Jumps are PC relative, meaning that the immediate is encoded as an offset in bytes from that instruction. I think there may be a typo in the solutions, for the top solution, the instruction corresponding to "loop" (beq a1 x0 end) is 7 instructions *before* j loop so the offset should be  $-(7 * 4) = -28$  (since each instruction is 4 bytes). For the alternate solution, "loop" is 8 instructions before, so the offset should be -32.

♡ 1 ...



Anonymous Gerbil 8mth #468aec

Does the j loop or loop: beq a1 x0 end count as an instruction when calculating the offset?

♡ ...

A

Andy Chen STAFF 8mth #468aee

↩ Replying to Anonymous Gerbil

j loop is considered to have an offset of 0 relative to the PC, and we count from there! If we were jumping to the instruction right before j loop the offset would be -4 and if we were jumping to the instruction right after j loop the offset would be 4. If we attempted to jump to the instruction j loop (infinite loop) the offset would be 0. So I think in some sense you could think of j loop as not being counted (it is being used as a point of reference) and loop: beq a1 x0 as being counted.

♡ 1 ...



Anonymous Gerbil 8mth #468aff

↩ Replying to Andy Chen

thanks!!!!

♡ 1 ...



Anonymous Loris 8mth #468aad

✓ Resolved

Su22-MT-Q2.1: why do we minus 97 here?

Q2.1 (5 points) contains takes in a trie root node (node), and a pointer to a string (word). It should return true if word is in the trie, and false otherwise.

```
1 bool contains(AlphaTrieNode* node, char* word) {
2     for (int i = 0; i < strlen(word); i++) {
3         int char_to_ascii = (int) word[i] - 97;
4         if (node->next[char_to_ascii] == NULL) {
5             return false;
6         }
7         node = node->next[char_to_ascii];
8     }
9     return node->last;
10 }
```

♡ ...



Andy Chen STAFF 8mth #468aba

The value of a char corresponds to its ASCII code. The characters a-z correspond to ASCII codes 97-122, so we subtract 97 from them to get their corresponding index in the node->next array.

(i.e. If word[i] is 'a', then its ASCII code is 97, so we subtract 97 to get 0 which is the index that it should be at in the node->next array)

♡ 1 ...



Anonymous Loris 8mth #468aac

✓ Resolved

Su22-MT-Q3.6: Why is the step size of the floating point system  $2^{-6}$ ? What is the largest positive number representable by the floating point system?

♡ ...



A

Andy Chen STAFF 8mth #468abd

Since we are looking for a number smaller than  $2^5$ , we are actually concerned with the step size when the exponent term is  $2^4$ . When the exponent term is  $2^4$ , the step size (the gap between representable numbers) is  $2^{-6}$  because incrementing the mantissa term by the smallest possible amount increases the represented floating point number by  $2^{-6}$ .

For example, with an exponent value of 19 (0b10011) (since the bias is -15):

$$0\ 10011\ 0000000000 \rightarrow 2^{(19-15)} * 1.0000000000_2 = 2^4 * (2^0) = 2^4$$

$$0\ 10011\ 0000000001 \rightarrow 2^{(19-15)} * 1.0000000001_2 = 2^4 * (2^0 + 2^{-10}) = 2^4 + 2^{-6}$$

And for the next smallest number from  $2^5$ :

$$0\ 10100\ 0000000000 \rightarrow 2^{(20-15)} * 1.0000000000_2 = 2^5 * (2^0) = 2^5$$

$$0\ 10011\ 1111111111 \rightarrow 2^{(19-15)} * 1.1111111111_2 = 2^4 * (2^0 + 2^{-1} + \dots + 2^{-10}) = 2^5 - 2^{-6}$$

(Since the exponent is 10011 we know the step size is  $2^{-6}$ )

The largest positive floating point number would be much larger than the largest positive fixed point number (it would be 0 11110 1111111111 which should be  $2^{16} - 2^{-5}$ . Exponent can't be 11111 since that corresponds to infinity/NaN)

♡ 1 ...



Anonymous Loris 8mth #468acc

Thank you so much !

♡ 1 ...



Anonymous Dotterel 8mth #468fe

✓ Resolved

SU22-Midterm-Q2.4:

I thought calloc didnt work because it initializes everything to 0, does this mean that 0 = NULL in c?

Also for 2.5 I thought realloc did not work because it doesn't increase the **next** memory, only the struct memory, how does it know that it is supposed to increase the **next** memory? If there was two string variables (so like two **next**), how would it increase those memories when doing realloc?

♡ ...



E

Erik Yang STAFF 8mth #468aaa

0 is not the same as NULL. Calloc helps initialize the memory, which means we can call on any pointer.

For 2.5, that is true. Realloc reallocates the memory for the new struct, but since it is uninitialized, we will run into pointer issues

♡ ...





**Anonymous Dotterel** 8mth #468abc

oh I think I get it to confirm. since Calloc initializes the memory that the pointer is then its ok to call on any pointer.

But since malloc and realloc don't initialize it then the pointer is pointing at garbage which makes us error?

♡ ...



**Erik Yang** STAFF 8mth #468acb

↩ Replying to Anonymous Dotterel

yup exactly!

♡ ...



**Adrian Bao** 9mth #468ed

✓ Resolved

SP22-Q4.2

I am confused on what this code does. Also, I was wondering why we don't need to manipulate the stack pointer. Please clarify.

```
lw a0, 0(s0)
jalr ra, s1, 0
sw a0, 0(s2)
lw a0, 4(s0)
jalr ra, s1, 0
sw a0, 4(s2)
```

♡ ...



**Adrian Bao** 9mth #468ee

Edit: I was also wondering why jalr is necessary over jal. Thanks!

♡ ...



**Jero Wang** STAFF 9mth #468fc

We first load the `x` of `self` (stored in `a0`), then we apply the function `f` to it (stored in `s1`), then we save `f(self->x)` (which is in `a0` at this point) into the `x` field of `newVector` (in `s2`). We then repeat for `y`. We need to use `jalr` here because we're jumping to an address stored in a register (since we can call `transform` with different `f`s), as opposed to a label.

♡ 1 ...



**Adrian Bao** 9mth #468de

✓ Resolved

SP22-Q5.1

I was wondering how the clock influences the values of RegA, RegB, and the output, or if it doesn't and just keeps time. Is it possible for the circuit to have register values influenced by the clock?

**Solution:** 1, 2, 3, 5, 8, 13, 21, 34

After the first clk-to-q time during clock cycle 0, Q of A is 0, and Q of B is 1. The sum outputted is 1, which gets fed back to RegA to be used for clock cycle 1. The next value taken in for RegB is the **previous** value outputted from Q by RegA. For the next clock cycle, the value of RegA becomes the value of the output from the previous cycle (1) and the value of RegB becomes the output of RegA from the previous cycle (0), so at clock cycle 1, the adder adds together values 0 (from RegA) and 1 from (RegB) and outputs 1. This cycle continues:

Clock	RegA	RegB	Output
0	0	1	1
1	1	0	1
2	1	1	2
3	2	1	3
4	3	2	5
5	5	3	8
6	8	5	13
7	13	8	21
8	21	13	34

At some point, you may notice that these values are the Fibonacci sequence! Checking the circuit, we see that each iteration, we are effectively doing  $a, b = a+b, a$ , which matches the behavior of the Fibonacci sequence.

♡ ...



Jero Wang STAFF 9mth #468fb

Clock is just referring to the clock cycle here, not the actual value of the clock. You can have a register whose input is a clock...though I think plugging a clock into the input of a register will leave you with some setup/hold violations.

♡ 1 ...

A

Adrian Bao 9mth #468dd

✓ Resolved

SP22-Q4.1

Why wouldn't we use `calloc(1, sizeof(Vector))` instead of `malloc(sizeof(Vector))`? Thanks!

**Solution:**

```
7 Vector *transform(Vector *self, int (*f)(int)) {
8     Vector* newVector = malloc(sizeof(Vector));
9     newVector -> x = f(self->x);
10    newVector -> y = f(self->y);
11    return newVector;
12 }
```

♡ ...



Justin Yokota STAFF 9mth #468eb

You can, but it would be less efficient, since you don't need a calloc'ed block.

♡ 1 ...



A Adrian Bao 9mth #468ec

Got it. If I said that, would I get credit?

♡ ...

J **Jero Wang** STAFF 9mth #468fa

↩ Replying to Adrian Bao

I think we did give credit for this exam.

♡ 1 ...



**Anonymous Mink** 9mth #468da

✓ Resolved

sp22-mt-Q3 Do we generally not calculate the arguments before we pass them into the function? For example we leave  $0+1$  as  $0+1$  not  $1$ ?

♡ ...



M **Myrah Shah** STAFF 9mth #468db

The way that these are used according to the `#define` statement makes it so that the result of using  $0+1$  !=  $1+0$ . If you have follow-up questions about the solution, feel free to ask!

♡ ...



**Anonymous Mink** 9mth #468dc

My question is do we always do this?

♡ ...



J **Justin Yokota** STAFF 9mth #468ea

↩ Replying to Anonymous Mink

No; this is specific to how define statements work, because define statements do a pure find-and-replace.

♡ ...



**Anonymous Dotterel** 9mth #468cc

✓ Resolved

For sp22 mt 1 Q3, im confused on how `kptr` turns into `k`.

```

#include <stdio.h>
#include <stdlib.h>
#define abs(x) ((x) < 0 ? -(x) : (x))
#define f(a,b) a*b/4
int main() {
    int a = 10;
    printf("Question 3.1: %d\n", a^2);
    int i = 0xA6004F4E;

    printf("Question 3.2: 0x%X\n", i|(i<<4));
    printf("Question 3.3: 0x%X\n", abs(i));

    int b = 10;
    printf("Question 3.4: %d\n", f(0+1, b));
    printf("Question 3.5: %d\n", f(1+0, b));


    int k = 100;
    int* kptr = &k;
    printf("Question 3.6: %d\n", f(k+,kptr));
    return 0;
}

```

**Solution:** Question 3.6: 125

The realization here is that `*`, previously used as the multiply operator, is now used as the de-reference operator. After substitution, we get `k+*kptr/4`, which evaluates to  $k + k/4 = 100 + 100/4 = 125$ .

♡ ...

 **Myrah Shah** STAFF 9mth #468cf

Remember that a `#define` statement replaces the macro, in this case, `f(a,b)` with the value `a*b/4`. This means for `f(k+,kptr)`, our `a` is `k+` and our `b` is `kptr`, so when we replace these, we get `k+ *kptr/4`. If we remember, the `*` will get the value at our pointer (`kptr`), which is equivalent to `k`, so we can simplify to `k+k/4`. Feel free to follow up if you have any more questions!

♡ ...

 **Anonymous Albatross** 9mth #468ca ✓ Resolved

SU22-MT-Q5: Is this fully secure machine in scope? If yes, what do they mean when they say 0/0 or 0/1. Is 0/1 meaning that it sees a 0 then a 1?

♡ ...

 **Justin Yokota** STAFF 9mth #468cb

FSMs are not in scope for the midterm

♡ ...



Anonymous Bee 9mth #468bd

✓ Resolved

SP22-Midterm-Q3.1

I understand how we got 8 from XOR but I'm confused why the answer is not 8.0 as we have the %d formatter and not %i.

♡ ...



Justin Yokota STAFF 9mth #468be

%d format is for integers. You're thinking of %f, which would be for floating point numbers.

♡ ...



Anonymous Stingray 9mth #468ae

✓ Resolved

SU22-Q4.1 - How come the main for loop has no instructions to increment the array pointer to the next element or decrease the length by 1?

♡ ...



Myrah Shah STAFF 9mth #468bf

We do both of these operations under the label `pass`, which executes after the loop.

♡ ...



Anonymous Stingray 9mth #468cd

Sorry for any confusion, but doesn't `pass` only execute if `t2` doesn't equal 0 (in other words, if it is odd)?

♡ ...



Myrah Shah STAFF 9mth #468ce

↩ Replying to Anonymous Stingray

Remember that in RISC-V we will continue executing instructions in order until we reach either a jump or branch instruction. Here we have two cases.

#### **t1 (our current element) is odd:**

- `bne t2 x0 pass` will take us to the label `pass`
- `pass` will increment our pointer and counter
- `j loop` takes us back to the top of our loop

#### **t1 (our current element) is even:**

- `bne t2 x0 pass` will NOT take us to the label `pass`
- We execute the next instruction which adds `t1` to our running sum
- Since we have no jump, we continue executing instructions in order, taking us to `pass`
- `pass` will increment our pointer and counter
- `j loop` takes us back to the top of our loop

The idea here is that we execute `pass` either way -- if something here doesn't make sense, feel free to follow up!

♡ ...




Anonymous Stingray 9mth #468df

↩ Replying to Myrah Shah


That makes much more sense, thank you for clarifying

♥ 1 ...

 **Anonymous Coyote** 9mth #468ac ✓ Resolved  
SU22-Q2.1


can we not do the cast for converting to ascii?

♥ ...

 **Justin Yokota** STAFF 9mth #468af

I don't think the typecast is strictly necessary, though I don't remember exactly.

♥ ...

 **Anonymous Coyote** 9mth #468aa ✓ Resolved

SP22-Q4.1 why we don't need to put a cast (Vector\*) before malloc? Also, why we don't need a \* before f when call the function (\*f)(self->x). thank you!

♥ ...

 **Jero Wang** STAFF 9mth #468ef

You never need to explicitly cast the result of malloc. Functions are automatically dereferenced (see [this](#)).

♥ ...

 **Anonymous Manatee** 8mth #468aab

would it be graded as wrong if you wrote (\*f)(self->x)?

♥ ...

 **Anonymous Coyote** 9mth #468f ✓ Resolved


SP22-Q3, I wonder where could we find the explanation of how to use #define in our course materials?

♥ ...

 **Justin Yokota** STAFF 9mth #468ba

We don't really cover specific use cases? We mostly discuss what these tools are more than when you might want to use those tools, leaving that to projects and personal projects.

♥ ...

 **Anonymous Gorilla** 9mth #468e ✓ Resolved


SU22-Q4.3, how do we know that we need a stack pointer for is\_prime, but not add\_even\_numbers, in the video it mentioned ra, but why do we need ra for this one and not the even one?

♥ ...

 **Eddy Byun** STAFF 9mth #468ad

We never make a function call in add\_even\_numbers, so there is no need for us to save the ra register.

♥ ...

 **Anonymous Gorilla** 9mth #468d ✓ Resolved


SU22-Midterm-Q3.5 and Q3.6, Are there walkthrough videos for these questions? I don't see them posted.

♥ ...

 **Jedidiah Tsang** STAFF 9mth #468bb


If they're not on the website, unfortunately we don't have them.

♡ ...

 **Anonymous Gorilla** 9mth #468c ✓ Resolved

SU22-Midterm-Q3.3 and Q3.4, why do we do a 11 digit number for Q3.3 (1.0000 0000 00) and why do we use a 14 digit number(0 10000 00000000). Later in the video, it says there's  $2^{11}$  numbers that can represent for floating point system, but it has  $2^{14}$ , but I don't see where these numbers came to be.

♡ ...

 **Jero Wang** STAFF 9mth #468fd


The "11 digit number" comes from the mantissa of the floating point number, plus the implicit 1. I'm not sure what the 14 digit number you're referring to is from, I don't see it in the solutions, is this from the recording?

♡ ...

 **Anonymous Gorilla** 8mth #468adb

Yes, this is from the recording

♡ ...

 **Anonymous Gorilla** 9mth #468a ✓ Resolved

SU22-Midterm-Q2.1, why do we do `next[char_to_ascii]`? Why do we need to index into it?

♡ ...

 **Andrew Liu** STAFF 9mth #468b

This is because `next` is an array of `AlphaTrieNode*`

♡ 1 ...