# [Midterm] Past Exams - 2023 #469

Eric Che STAFF
9 months ago in Exam – Midterm

2,920
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!

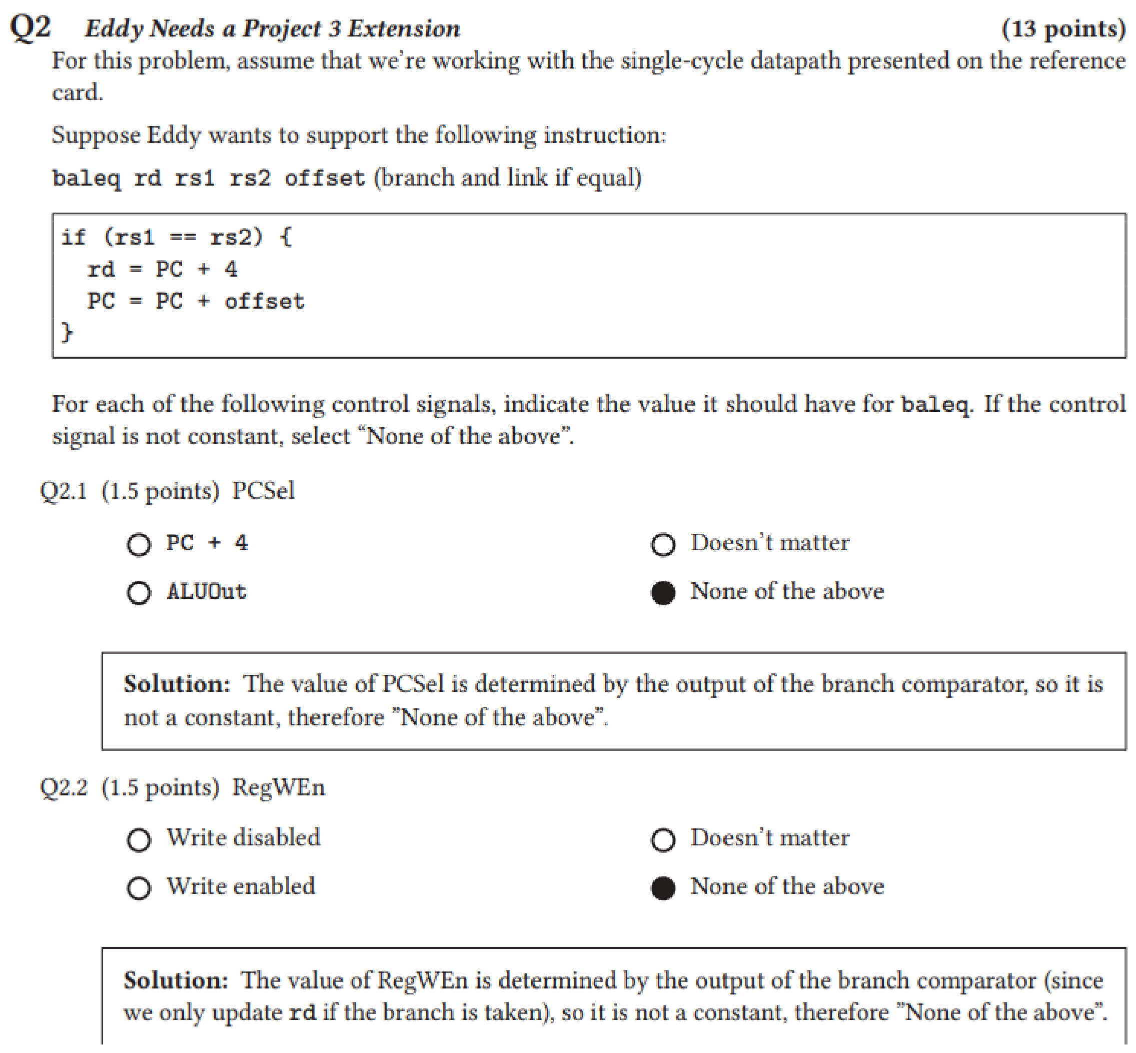
When posting questions, please reference the semester, exam, and question in this format so it's easier for students and staff to search for similar questions:

**Semester-Exam-Question Number**

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

Anonymous Turtle 8mth #469adbb ✓ Resolved

fa23 final Why are these "None of the above" and not PC+4 and write enabled



**Q2** ***Eddy Needs a Project 3 Extension*** **(13 points)**

For this problem, assume that we're working with the single-cycle datapath presented on the reference card.

Suppose Eddy wants to support the following instruction:

`baleq rd rs1 rs2 offset` (branch and link if equal)

```
if (rs1 == rs2) {
  rd = PC + 4
  PC = PC + offset
}
```

For each of the following control signals, indicate the value it should have for `baleq`. If the control signal is not constant, select "None of the above".

Q2.1 (1.5 points) PCSel

- `PC + 4`
- `ALUOut`
- Doesn't matter
- [✓ None of the above]

**Solution:** The value of PCSel is determined by the output of the branch comparator, so it is not a constant, therefore "None of the above".

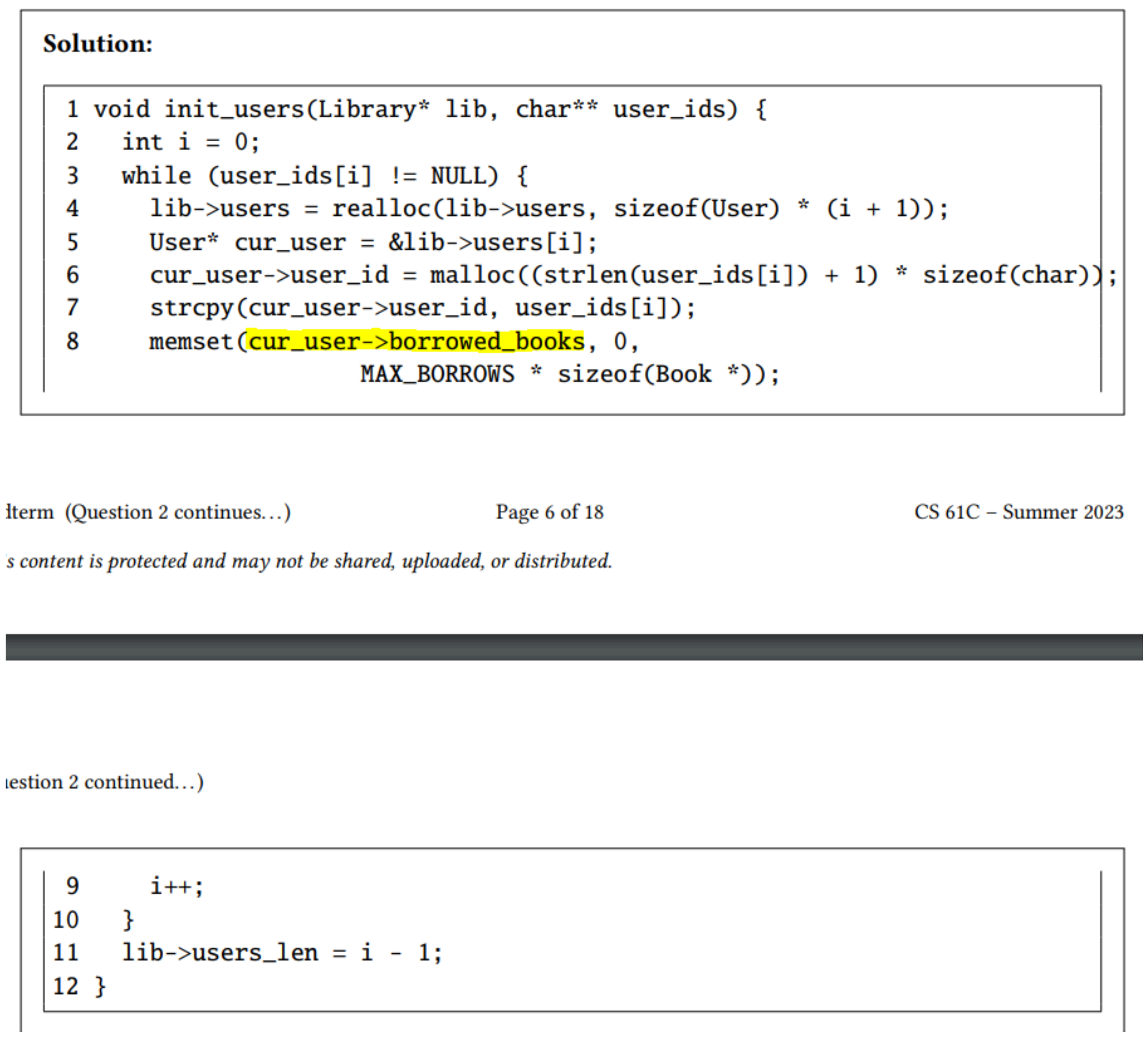
Q2.2 (1.5 points) RegWEn

- Write disabled
- Write enabled
- Doesn't matter
- [✓ None of the above]

**Solution:** The value of RegWEn is determined by the output of the branch comparator (since we only update `rd` if the branch is taken), so it is not a constant, therefore "None of the above".

damien toh 8mth #469aced ✓ Resolved

su23-mt-q2


**Solution:**

```
void init_users(Library* lib, char** user_ids) {
  int i = 0;
  while (user_ids[i] != NULL) {
    lib->users = realloc(lib->users, sizeof(User) * (i + 1));
    User* cur_user = &lib->users[i];
    cur_user->user_id = malloc((strlen(user_ids[i]) + 1) * sizeof(char));
    strcpy(cur_user->user_id, user_ids[i]);
    memset(cur_user->borrowed_books, 0,
                  MAX_BORROWS * sizeof(Book *));
```

lterm (Question 2 continues…) Page 6 of 18 CS 61C – Summer 2023

*s content is protected and may not be shared, uploaded, or distributed.*

iestion 2 continued…)

```
    i++;
  }
  lib->users_len = i - 1;
}
```

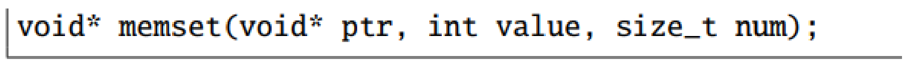
hi before running memset, what is the value of cur_user->borrowed_books? isn't it garbage value, since we created it using **realloc**?

Justin Yokota STAFF 8mth #469acfa

Yes. But note that borrowed_books is an array, not a pointer. So it is correct to set the data starting at borrowed_books to Book*s

damien toh 8mth #469acff

cur_user->borrowed_books should be a pointer right? since memset function signature says that the first argument should be a pointer

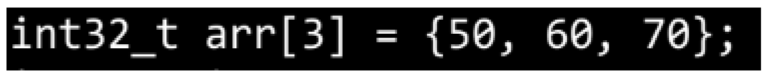


```
void* memset(void* ptr, int value, size_t num);
```

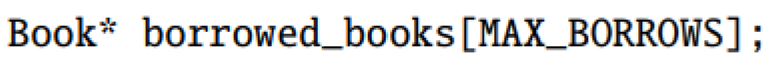
and isn't an array just a pointer to the first element? so what does it mean when you say borrowed_books is an array and not a pointer

damien toh 8mth #469adaa

Replying to damien toh



```
int32_t arr[3] = {50, 60, 70};
```



```
Book* borrowed_books[MAX_BORROWS];
```

also what's the difference between the 2? there's a * in borrowed books and no * in arr. thanks

Justin Yokota STAFF 8mth #469adac

Replying to damien toh

The first is an array of 3 integers. The second is an array of MAX_BORROWS Book pointers.

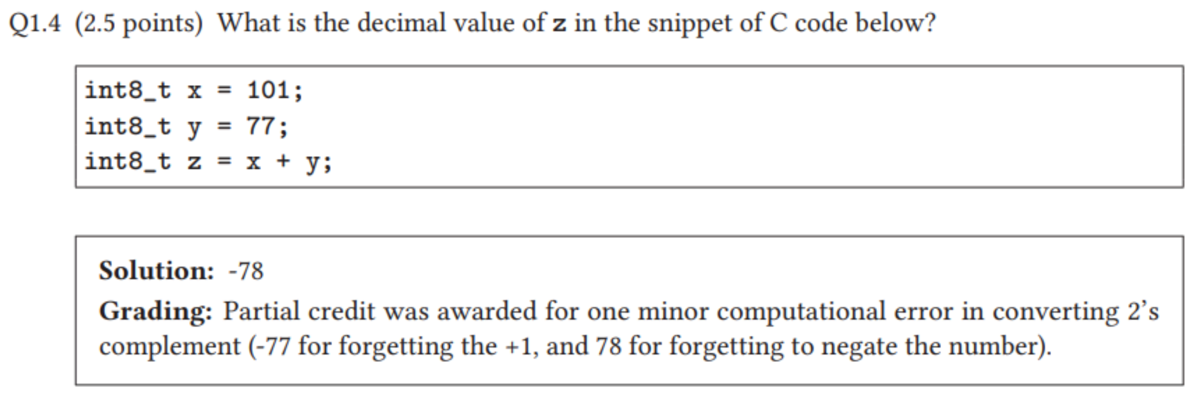
Justin Yokota STAFF 8mth #469adab

Replying to damien toh

An array, when used as a variable (such as as a memset argument) acts as a pointer to its first element. But in memory, it acts differently, in that a pointer sets aside space for a pointer (and thus you need to set the pointer to an allocated location in memory before using it), while an array sets aside space for the entire sequence of items (and therefore already has space allocated)

Anonymous Duck 8mth #469acdd ✓ Resolved

Why this is getting -78?



Q1.4 (2.5 points) What is the decimal value of **z** in the snippet of C code below?

```
int8_t x = 101;
int8_t y = 77;
int8_t z = x + y;
```

**Solution:** -78

**Grading:** Partial credit was awarded for one minor computational error in converting 2's complement (-77 for forgetting the +1, and 78 for forgetting to negate the number).

Jedidiah Tsang STAFF 8mth #469acde

Convert those two numbers to binary, then add them together - notice that there's overflow, causing the resulting number to be interpreted as a negative number

Daniel Wang 8mth #469acea

Wait, I thought this was because of representing numbers as Two's Complement?

101 = 0b0110 0101
77   = 0b0100 1101

z = 0b10110010

Two's Complement:

0b10110010 --> 0b01001101 --> 0b01001110 --> 78 --> -78 (interpret as negative number).

Justin Yokota STAFF 8mth #469acec

Replying to Daniel Wang

That's also correct. It's considered overflow because numerically, 101+77 is not equal to -78; it's only -78 because of the limitations we have with an 8-bit integer.

Daniel Wang 8mth #469acee

Replying to Justin Yokota

Ah I see, it's because if we were to have kept the MSB sign bit as 0, we would have overflow since the bit width exceeds 8 bits (into 9). Thank you so much!

Anonymous Parrot 8mth #469accd ✓ Resolved

fa 23-mt-q2.3

Why do we need an "&" here?

Can we just do: slice ->data = v ->data[start_index]?

is the & here used in the context of &(v->data[start_index]) ? Need & to get value of at index?

vector_t* vec_b = vector_slice(vec_a, 1, 3); vec_b->data[1] = 10;

This appears to be legal but I'm not sure why 2.3 needs an &

Anonymous Tarsier 8mth #469acce

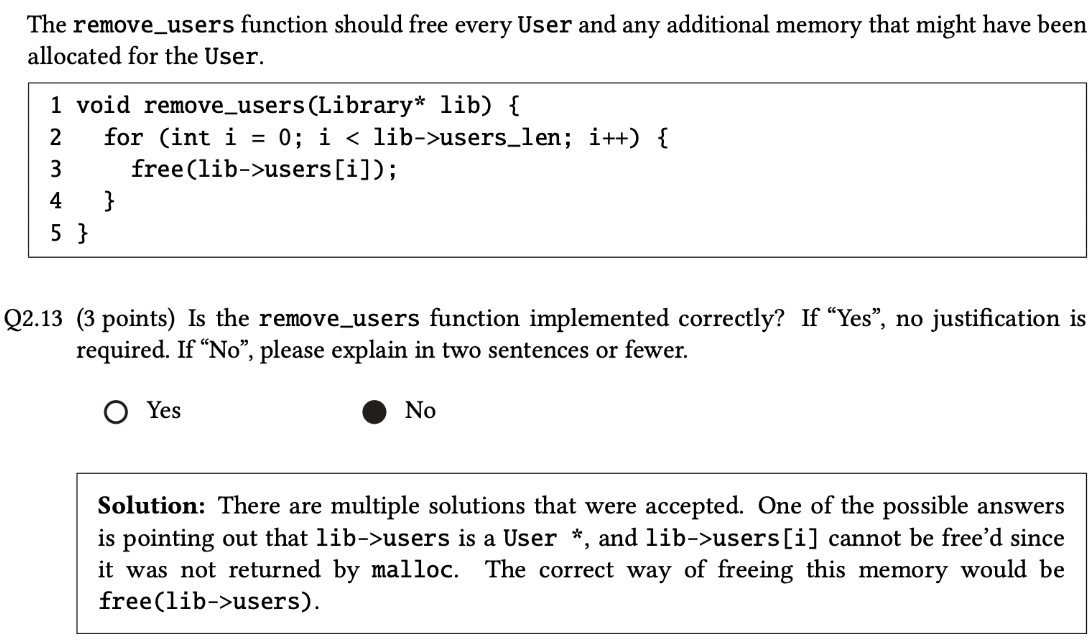
`data` is a pointer so we need to use & to get the address of `v->data[start_index]` (an integer in the `data` array).

Anonymous Spoonbill 8mth #469acca ✓ Resolved

SU23-MT-Q2.13, FA23-MT-Q2

Why can't we free users by each element in this question for Summer 2023 MT below, but we can free each element of v->slices in the Fa23 MT? Thanks

Summer 23:


The `remove_users` function should free every `User` and any additional memory that might have been allocated for the `User`.

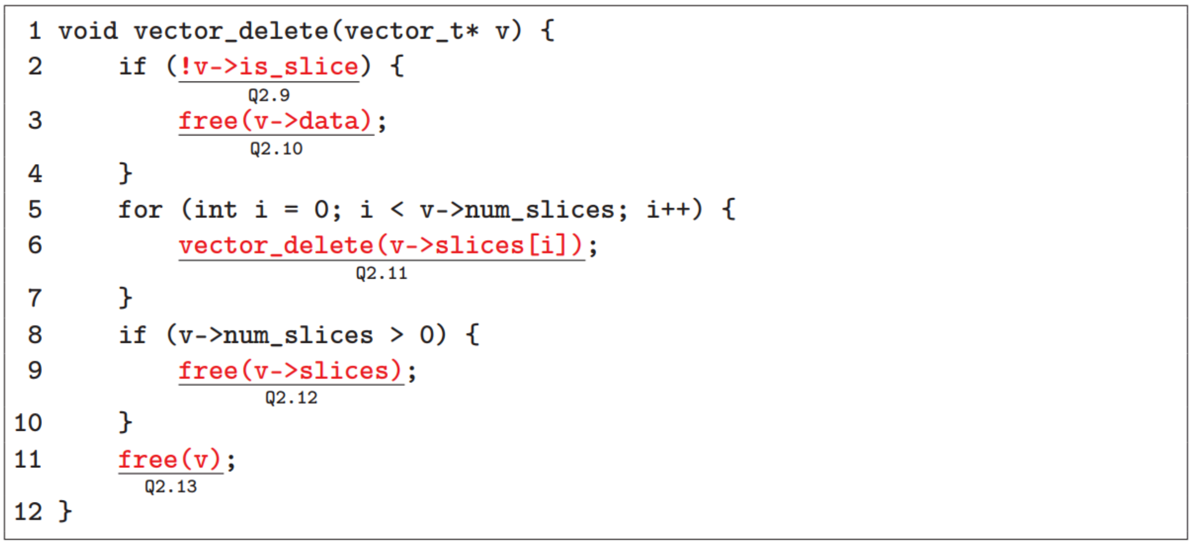
```
void remove_users(Library* lib) {
  for (int i = 0; i < lib->users_len; i++) {
    free(lib->users[i]);
  }
}
```

Q2.13 (3 points) Is the `remove_users` function implemented correctly? If "Yes", no justification is required. If "No", please explain in two sentences or fewer.

Yes [✓ No]

**Solution:** There are multiple solutions that were accepted. One of the possible answers is pointing out that `lib->users` is a `User *`, and `lib->users[i]` cannot be free'd since it was not returned by `malloc`. The correct way of freeing this memory would be `free(lib->users)`.

Fall 23:


```
void vector_delete(vector_t* v) {
    if (!v->is_slice) {
              Q2.9
        free(v->data);
          Q2.10
    }
    for (int i = 0; i < v->num_slices; i++) {
        vector_delete(v->slices[i]);
                   Q2.11
    }
    if (v->num_slices > 0) {
        free(v->slices);
            Q2.12
    }
    free(v);
     Q2.13
}
```

Myrah Shah STAFF 8mth #469aceb

You can only free space that was allocated by `malloc` / `calloc` .

Anonymous Tarsier 8mth #469acbc ✓ Resolved

FA23-MT-Q2

For `vector_delete` , can the if condition also be `if (v->num_slices == 0)` ? Also, when we delete slices of a vector `v` doesn't `v->num_slices` need to get set back to 0? I don't see how the solution accounts for this.

Minh Nguyen STAFF 8mth #469acbf

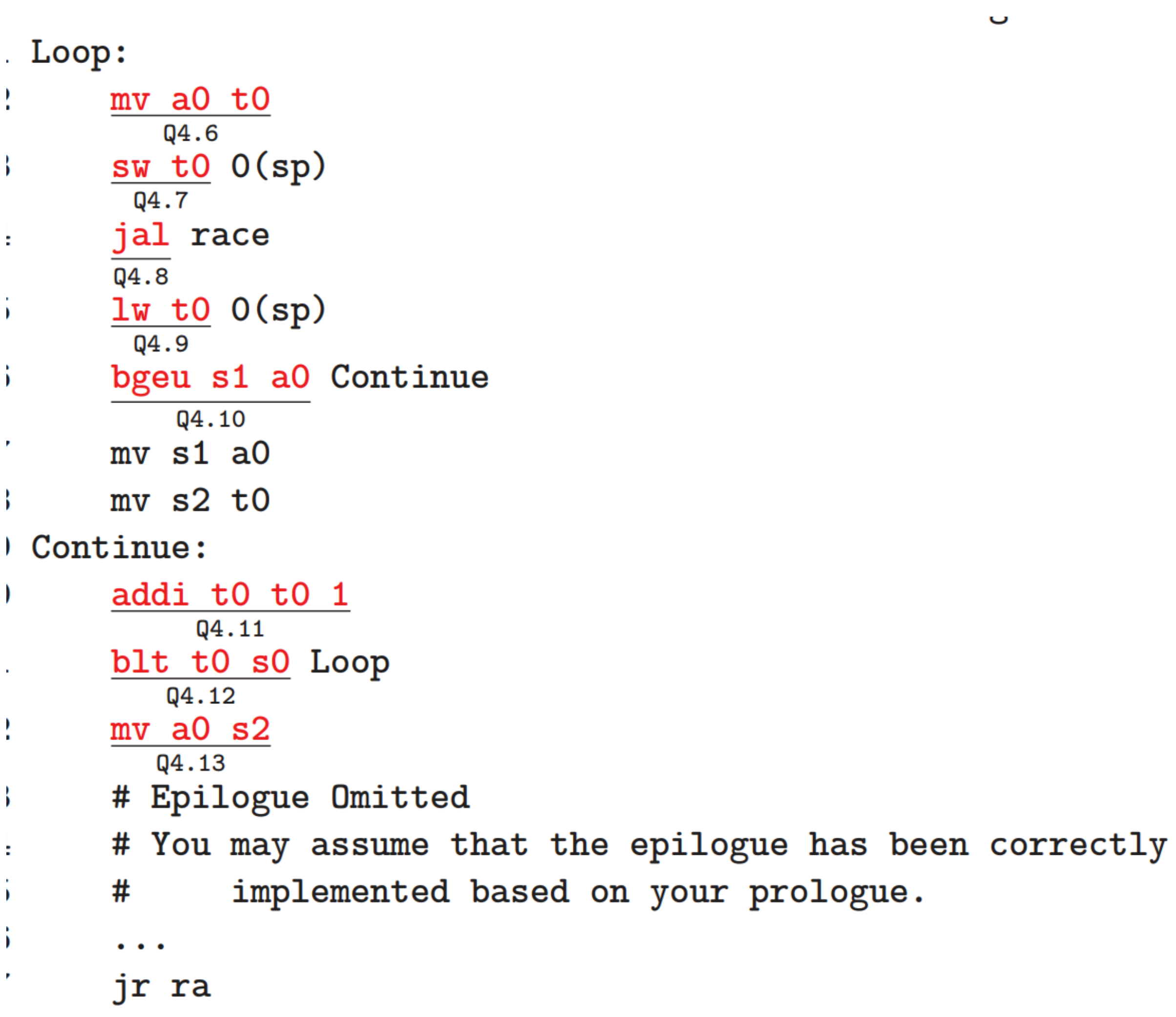
#469abec

1

Anonymous Panther 8mth #469acba ✓ Resolved

FA23- MT-Q4

why is 4.10 unsigned branch instruction but 4.12 a regular signed branch instruction? When do we know which to use?



```
Loop:
        mv a0 t0
           Q4.6
        sw t0 0(sp)
         Q4.7
        jal race
        Q4.8
        lw t0 0(sp)
         Q4.9
        bgeu s1 a0 Continue
            Q4.10
        mv s1 a0
        mv s2 t0
Continue:
        addi t0 t0 1
             Q4.11
        blt t0 s0 Loop
            Q4.12
        mv a0 s2
          Q4.13
        # Epilogue Omitted
        # You may assume that the epilogue has been correctly
        #     implemented based on your prologue.
        ...
        jr ra
```

1 ···

M Minh Nguyen STAFF 8mth #469acbd

`bgeu s1 a0` compares the distances, which cannot be negative, so we need to use an unsigned comparison. `blt t0 s0` compares indices, which could be both negative/positive, so we use signed comparison. To summarize, we use unsigned branching when dealing with non-negative values, and signed branching for values that can be both -/+, and for cases in which we want to respect the original arithmetic sign :D

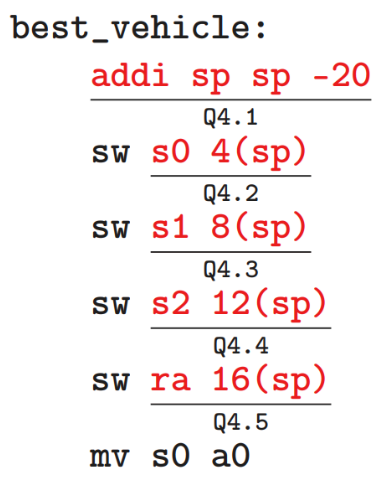
···

Anonymous Panther 8mth #469acae ✓ Resolved

FA23-MT-Q4

Why did we have addi sp sp -20 and start our first offset at 4? Can't we just do

addi sp sp -16
sw s0 0(sp)
sw s1 4(sp) and so on?



```
best_vehicle:
    addi sp sp -20
        Q4.1
    sw s0 4(sp)
        Q4.2
    sw s1 8(sp)
        Q4.3
    sw s2 12(sp)
        Q4.4
    sw ra 16(sp)
        Q4.5
    mv s0 a0
```

M

Minh Nguyen STAFF 8mth #469acbb

The solution decrements the stack by `-20` so that we can `sw t0 0(sp)` before calling `race`. You approach works but remember to follow CC and decrements the stack + storing `t0` before calling a function!

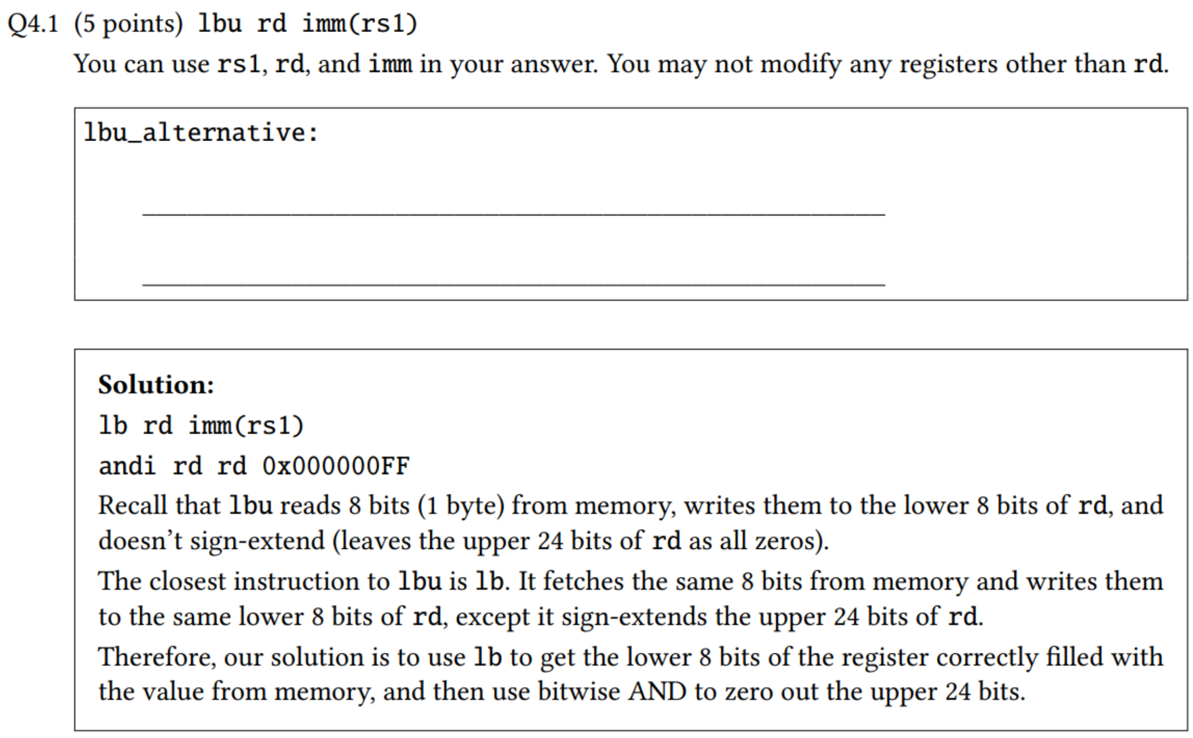
Anonymous Gorilla 8mth #469acaa ✓ Resolved

**SP23-MT-Q4.1**

Curious if

```
lhu rd imm(rs1)
andi rd rd 0x01
```

is a valid alternate solution to



Q4.1 (5 points) `lbu rd imm(rs1)`

You can use `rs1`, `rd`, and `imm` in your answer. You may not modify any registers other than `rd`.

`lbu_alternative:`

________________________________________

________________________________________

**Solution:**

```
lb rd imm(rs1)
andi rd rd 0x000000FF
```

Recall that `lbu` reads 8 bits (1 byte) from memory, writes them to the lower 8 bits of `rd`, and doesn't sign-extend (leaves the upper 24 bits of `rd` as all zeros).

The closest instruction to `lbu` is `lb`. It fetches the same 8 bits from memory and writes them to the same lower 8 bits of `rd`, except it sign-extends the upper 24 bits of `rd`.

Therefore, our solution is to use `lb` to get the lower 8 bits of the register correctly filled with the value from memory, and then use bitwise AND to zero out the upper 24 bits.

My idea here is that we load 2 unsigned bits from imm and imm+1 and then we zero out the immediate + 1 with the andi.

Justin Yokota STAFF 8mth #469acda

1: Note that "lbu" loads 1 byte, not one bit.

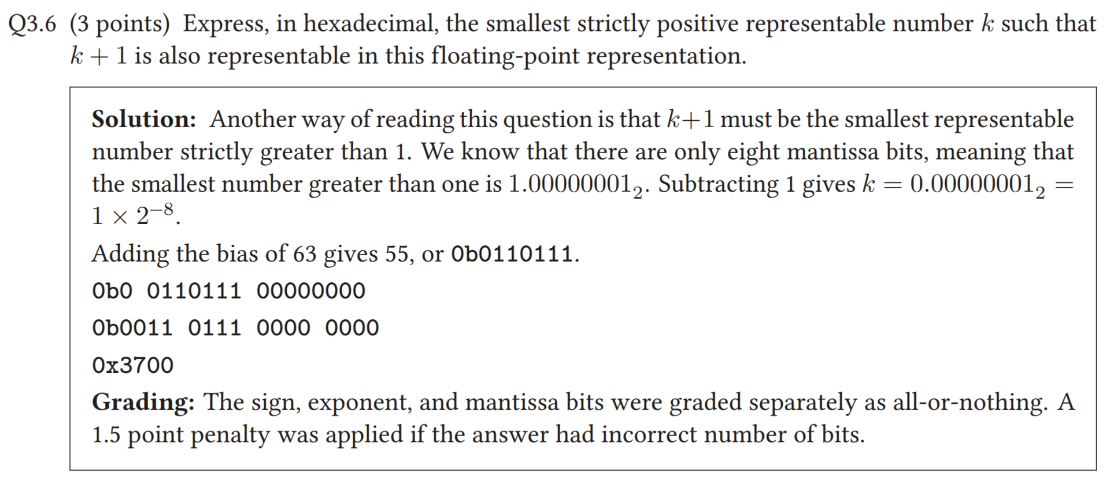
2: lhu only works on addresses that are multiples of 2. So you'll have a 50% chance of the instruction not being valid.

Anonymous Dinosaur 8mth #469abfb ✓ Resolved

**FA23-MT-Q3.6**

i'm confused how "another way of reading this question is that k+1 must be the smallest representable number strictly greater than one"

what exactly does "such that k+1 is also representable in this FP rep" mean?


Q3.6 (3 points) Express, in hexadecimal, the smallest strictly positive representable number $k$ such that $k+1$ is also representable in this floating-point representation.

**Solution:** Another way of reading this question is that $k+1$ must be the smallest representable number strictly greater than 1. We know that there are only eight mantissa bits, meaning that the smallest number greater than one is $1.00000001_2$. Subtracting 1 gives $k = 0.00000001_2 = 1 \times 2^{-8}$.

Adding the bias of 63 gives 55, or `0b0110111`.

```
0b0 0110111 00000000
0b0011 0111 0000 0000
0x3700
```

**Grading:** The sign, exponent, and mantissa bits were graded separately as all-or-nothing. A 1.5 point penalty was applied if the answer had incorrect number of bits.

Justin Yokota STAFF 8mth #469abfd

A number is representable if some bit pattern gets "translated" to that number. So for instance, 2 is representable, but 2.1 is not representable (because no floating point number is exactly 2.1).

Anonymous Tarsier 8mth #469acdb

If we're trying to find the smallest positive `k` such that `k+1` is also representable, how does this translate to "`k+1` must be the smallest representable number **strictly greater than 1**"? What's the reasoning for the last part?

Justin Yokota STAFF 8mth #469acdc

Replying to Anonymous Tarsier

If k+1 was less than or equal to 1, then k would be less than or equal to 0, and therefore not positive.

1

Anonymous Tarsier 8mth #469abea ✓ Resolved

FA23-MT-Q2

Why don't we need to set `slice->num_slices` to 0?

Eddy Byun STAFF 8mth #469abec

We `calloc` memory for `slice`, which means `num_slices` will be set to 0.

Anonymous Tarsier 8mth #469acaf

oh okay, so `calloc` sets everything in the struct to 0? (so all integer values are set to 0 and all pointers are set to the null pointer?)

E Eddy Byun STAFF 8mth #469accb

Replying to Anonymous Tarsier

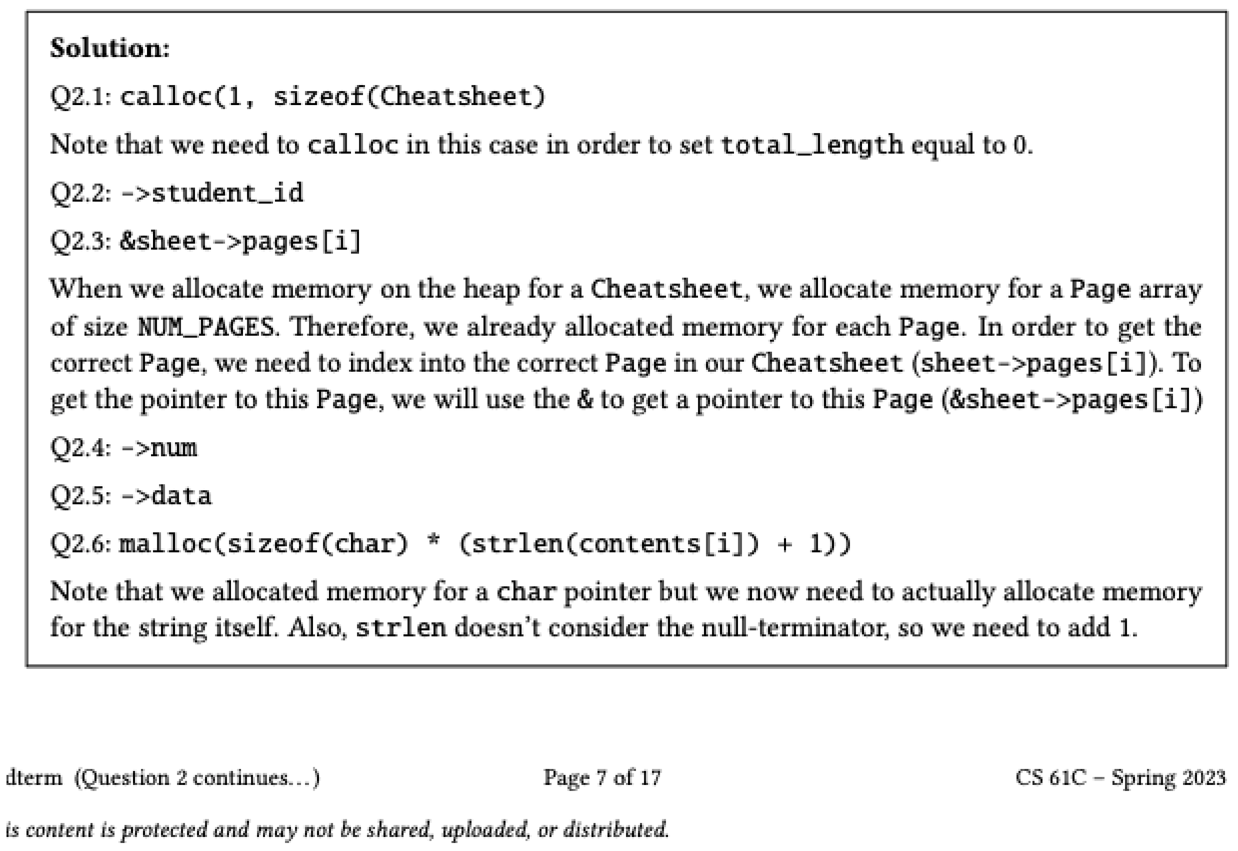
`calloc` is going to initialize every byte to 0 during memory allocation: https://en.cppreference.com/w/c/memory/calloc

♡ 1 ···

Anonymous Herring 8mth #469abda ✓ Resolved

SP23-MT-Q2

For part 2.3, are we able to do ((sheet->pages) + i) since that would give us a pointer? When we do pages[i] we are effectively doing *(pages + i) and then turning it into a pointer by getting its address so can we instead just do + i and keep it a pointer?



**Solution:**

Q2.1: `calloc(1, sizeof(Cheatsheet)`

Note that we need to `calloc` in this case in order to set `total_length` equal to 0.

Q2.2: `->student_id`

Q2.3: `&sheet->pages[i]`

When we allocate memory on the heap for a `Cheatsheet`, we allocate memory for a `Page` array of size `NUM_PAGES`. Therefore, we already allocated memory for each `Page`. In order to get the correct `Page`, we need to index into the correct `Page` in our `Cheatsheet` (`sheet->pages[i]`). To get the pointer to this `Page`, we will use the `&` to get a pointer to this `Page` (`&sheet->pages[i]`)

Q2.4: `->num`

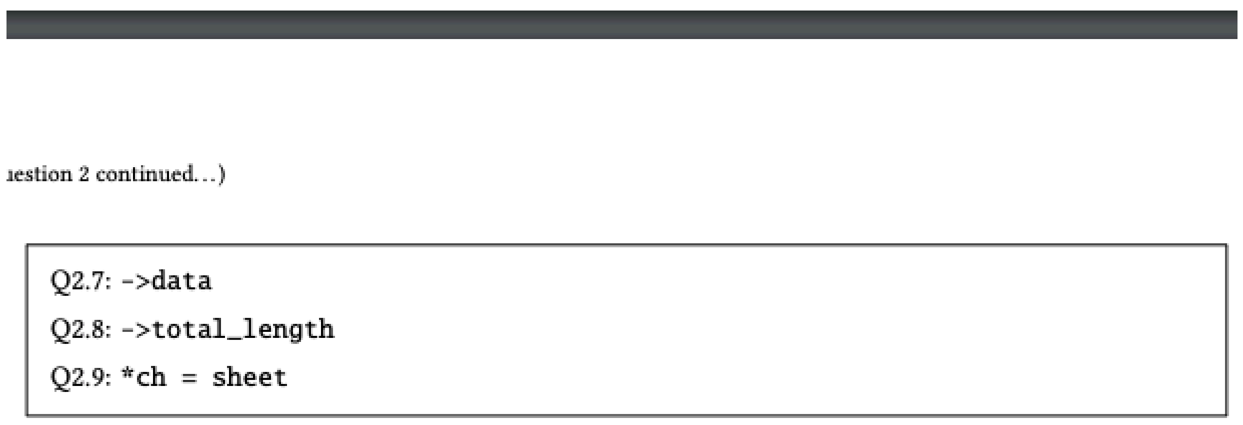
Q2.5: `->data`

Q2.6: `malloc(sizeof(char) * (strlen(contents[i]) + 1))`

Note that we allocated memory for a `char` pointer but we now need to actually allocate memory for the string itself. Also, `strlen` doesn't consider the null-terminator, so we need to add 1.

dterm (Question 2 continues...) Page 7 of 17 CS 61C – Spring 2023

*is content is protected and may not be shared, uploaded, or distributed.*



iestion 2 continued...)

Q2.7: `->data`

Q2.8: `->total_length`

Q2.9: `*ch = sheet`

♡ ···

E Eddy Byun STAFF 8mth #469abde

Yes, `sheet->pages + i` is equivalent

♡ ···

Anonymous Gnu 8mth #469abcf ✓ Resolved

FA23-MT-Q6.4

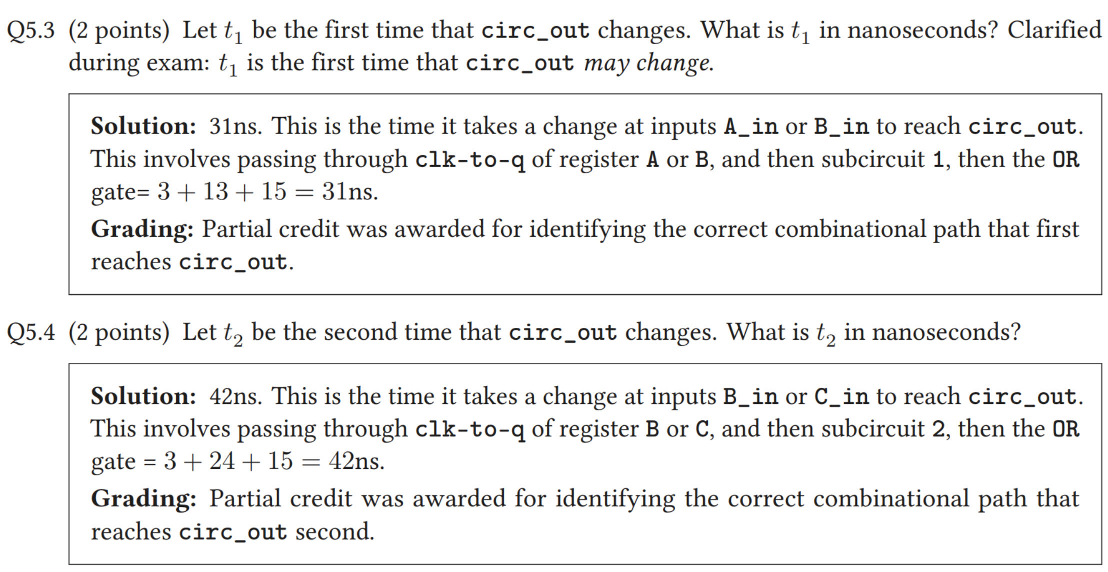
Why can't we do 6.4 bit by bit just like we did in 6.3 to achieve O(N)?

Justin Yokota STAFF 8mth #469abfe

XOR won't distinguish values bit-by-bit; it will only ever return True if your number is exactly the same as Luxor's favorite number.

Anonymous Flamingo 8mth #469abcd ✓ Resolved

Q5.3: Why do we not have to count the setup time here? Further, does the OR gate change the value of circ_out without having access to the other input?



Q5.3 (2 points) Let $t_1$ be the first time that `circ_out` changes. What is $t_1$ in nanoseconds? Clarified during exam: $t_1$ is the first time that `circ_out` *may change*.

**Solution:** 31ns. This is the time it takes a change at inputs `A_in` or `B_in` to reach `circ_out`. This involves passing through `clk-to-q` of register `A` or `B`, and then subcircuit `1`, then the `OR` gate= $3 + 13 + 15 = 31$ns.

**Grading:** Partial credit was awarded for identifying the correct combinational path that first reaches `circ_out`.

Q5.4 (2 points) Let $t_2$ be the second time that `circ_out` changes. What is $t_2$ in nanoseconds?

**Solution:** 42ns. This is the time it takes a change at inputs `B_in` or `C_in` to reach `circ_out`. This involves passing through `clk-to-q` of register `B` or `C`, and then subcircuit `2`, then the `OR` gate = $3 + 24 + 15 = 42$ns.

**Grading:** Partial credit was awarded for identifying the correct combinational path that reaches `circ_out` second.

1

Daniel Wang 8mth #469abdd

I think for the setup time, it's because `circ_out` is updated before the input of the last register.

As for the OR gate, this is because the wire between subcircuit 1 and the OR gate and the wire between subcircuit 2 and the OR gate can be assumed to have a default state of 0. Thus, at time 0ns $\leq t < t_1$, keeping $t < t_1$ in mind, the OR gate has not been updated yet with the result from subcircuit 1, meaning it will use the default wire states of 0.

From $t_1 \leq t < t_2$, the OR gate has now been updated with the output of subcircuit 1, giving $A\&B + 0 = A\&B$. Note that we still use the other wire's default state of 0 here.

Anonymous Wallaby 8mth #469abca ✓ Resolved

FA23-MT-Q3

How come we apply the bias to the floating point in the binary representation in 3.1 but not 3.2? Is it because 3.1 is normal when 3.2 is denormal?

Daniel Wang 8mth #469abdf

Yea! Implicitly, we apply the bias to a denormal number. I like to think of the exponent as being $1 - B$ in the case of denorms. This is because we use the smallest valid exponent bit value, which is all bits equaling 0 except the LSB (which is 1).

Anonymous Hippopotamus 8mth #469abbf ✓ Resolved

FA23-MT-Q2.6

Why are we indexing into `v->slices[v->num_slices]` and not `v->slices[(v->num_slices) - 1]` ? I thought we wanted the slices array to be 0-indexed thus it should be the amount - 1? Would this implementation also be considered correct?

Eddy Byun STAFF 8mth #469abeb

`v->slices[(v->num_slices) - 1]` would be off by 1. We increment `num_slices` on line 10 after the indexing happens!

Anonymous Gnu 8mth #469abbe ✓ Resolved

Sp23 Q3.4

Wouldn't it just be 2^15 - 2^11? Why do we need to "throw out" the infinities and NaNs if the infinities and NaNs can't even have a LSB exponent bit of 0. Shouldn't this mean that we didn't even count them through 2^15?

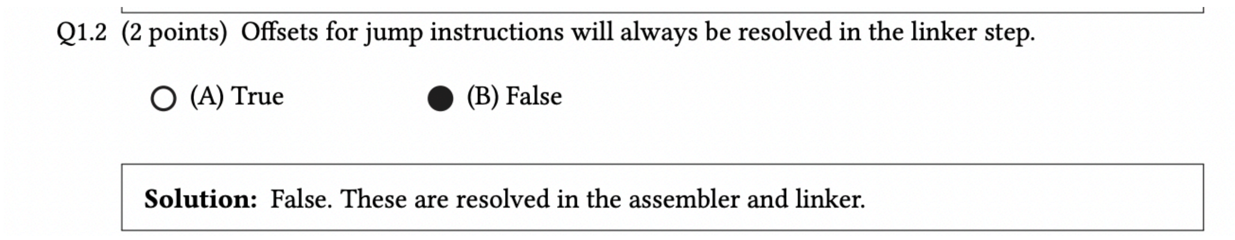
Justin Yokota STAFF 8mth #469abff

#469acc

Anonymous Sardine 8mth #469abbc ✓ Resolved

Sp23 Q1.2

What dictates where it is resolved?


Q1.2 (2 points) Offsets for jump instructions will always be resolved in the linker step.

○ (A) True [✓ ● (B) False]

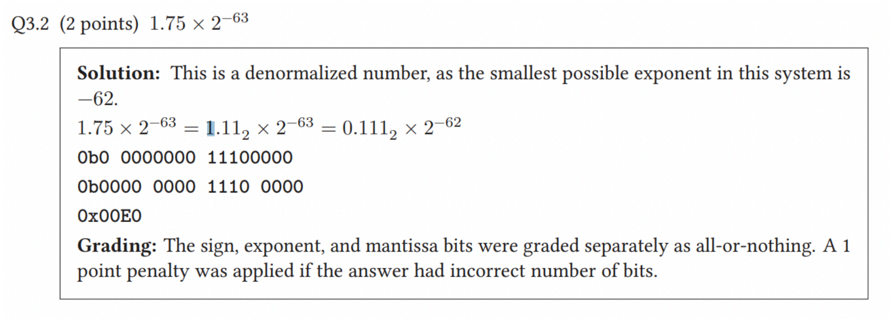
**Solution:** False. These are resolved in the assembler and linker.

Jedidiah Tsang STAFF 8mth #469acdf

Whether the assembler can locate that label or not!

Anonymous Sardine 8mth #469abae ✓ Resolved

FA23 - Q2.3, how do I figure out that this is a denormalized number? I referred to guide video in homework and Alan's notes.


Q3.2 (2 points) $1.75 \times 2^{-63}$

**Solution:** This is a denormalized number, as the smallest possible exponent in this system is $-62$.

$1.75 \times 2^{-63} = 1.11_2 \times 2^{-63} = 0.111_2 \times 2^{-62}$

```
0b0 0000000 11100000
0b0000 0000 1110 0000
0x00E0
```

**Grading:** The sign, exponent, and mantissa bits were graded separately as all-or-nothing. A 1 point penalty was applied if the answer had incorrect number of bits.

Andy Chen STAFF 8mth #469abbb

One way you could figure out that this must be a denormalized number is by thinking about the smallest possible (positive) normalized number. In this case, since we have 7 exponent bits,

0 0000001 00000000 --> $2^{(1-63)}$ * $1.00...00_2$ = $2^{-62}$ would be the smallest positive normalized number (Exp = 0000001, Mantissa = all 0's). Since 1.75 * $2^{-63}$ is smaller than this smallest positive normalized number, it cannot be represented as a normalized number, and we can deduce that it must be a denorm number.

Anonymous Porpoise 8mth #469abad ✓ Resolved

SP23-MT-Q2.6:

just to confirm calloc would work in this situation because we are using strcpy (so initializing all the indexes to be null terminator wont affect anything)

Eddy Byun STAFF 8mth #469abdc

Yes, for blank 2.6, you can use either `malloc` or `calloc`

Anonymous Mallard 8mth #469aaff ✓ Resolved

SP23-MT-Q4.2:

Is the following solution valid as well?

blt rs1 rs2 label

bgeu rs1 rs2 label

Eddy Byun STAFF 8mth #469abdb

We should not branch to label if `rs1` and `rs2` are equal to each other. The second line `bgeu rs1 rs2 label` will branch if `rs1` and `rs2` are equal

Anonymous Mallard 8mth #469abed

Oh right. Thanks!

Anonymous Duck 8mth #469aafe ✓ Resolved

SP23-Midterm-Q2.10

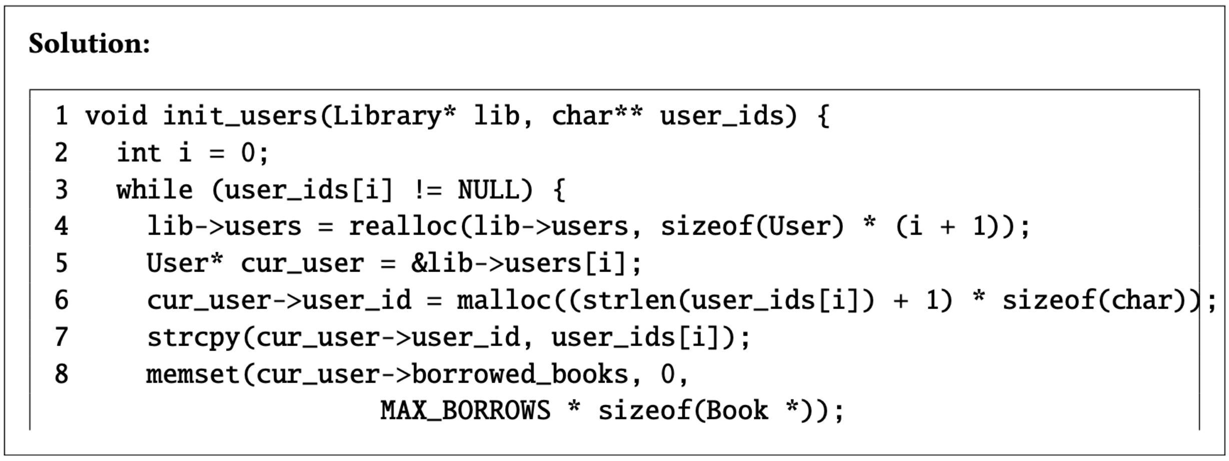
Why does NUM_PAGES is code? Isn't NUM_PAGES defined as a global variable, so it is data/static?

M Myrah Shah STAFF 8mth #469abaf

#469fdd

Anonymous Yak 8mth #469aafd ✓ Resolved

SU23-MT-Q2



**Solution:**

```
1 void init_users(Library* lib, char** user_ids) {
2   int i = 0;
3   while (user_ids[i] != NULL) {
4     lib->users = realloc(lib->users, sizeof(User) * (i + 1));
5     User* cur_user = &lib->users[i];
6     cur_user->user_id = malloc((strlen(user_ids[i]) + 1) * sizeof(char));
7     strcpy(cur_user->user_id, user_ids[i]);
8     memset(cur_user->borrowed_books, 0,
                         MAX_BORROWS * sizeof(Book *));
```

In line 3, why does lib->users[i] return a User and not a pointer to the User? In line 8, why is it sizeof(Book *) and not sizeof(Book)?

Daniel Wang 8mth #469abee

`lib->users[i]` accesses the users field of the struct pointer `lib`, and additionally indexes into the `users` User pointer, where we do pointer arithmetic and dereference.

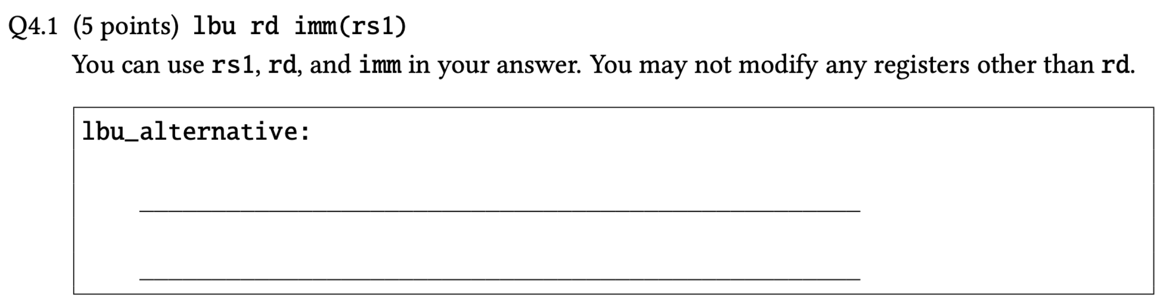
To break it down, `lib` is a `Library` pointer. Arrow notation is used to access the field(s) of a pointer pointing to a struct, meaning we can access the fields of the `Library` struct `lib` points to. `users` is an array of type of `User`, so when we index into it, we grab individual objects of type `User`.

This is similar for `borrowed_books`, where we have an array of pointers (see why?). The `Book*` indicates its type and brackets indicate `borrowed_books` is an array with size `MAX_BORROWS`, meaning `borrowed_books` is an array of `Book*` or Book pointers. So, when we want to initialize this array with the values of NULL or 0, we do so across how many items it has (`MAX_BORROWS`) as well as considering the space in memory each item takes up (`sizeof(Book *)`). Let me know if there's anything else that's unclear!

Anonymous Dragonfly 8mth #469aafc ✓ Resolved

Sp23-MT-q4

In the video, it says that we cannot xor to cancel out the sign extension. I am confused as to why we could not have just used xor rd rd 0xFFFFFF00? I thought xor 0 0 was just 0?


Q4.1 (5 points) `lbu rd imm(rs1)`

You can use `rs1`, `rd`, and `imm` in your answer. You may not modify any registers other than `rd`.

```
lbu_alternative:

    ________________________________________________

    ________________________________________________
```

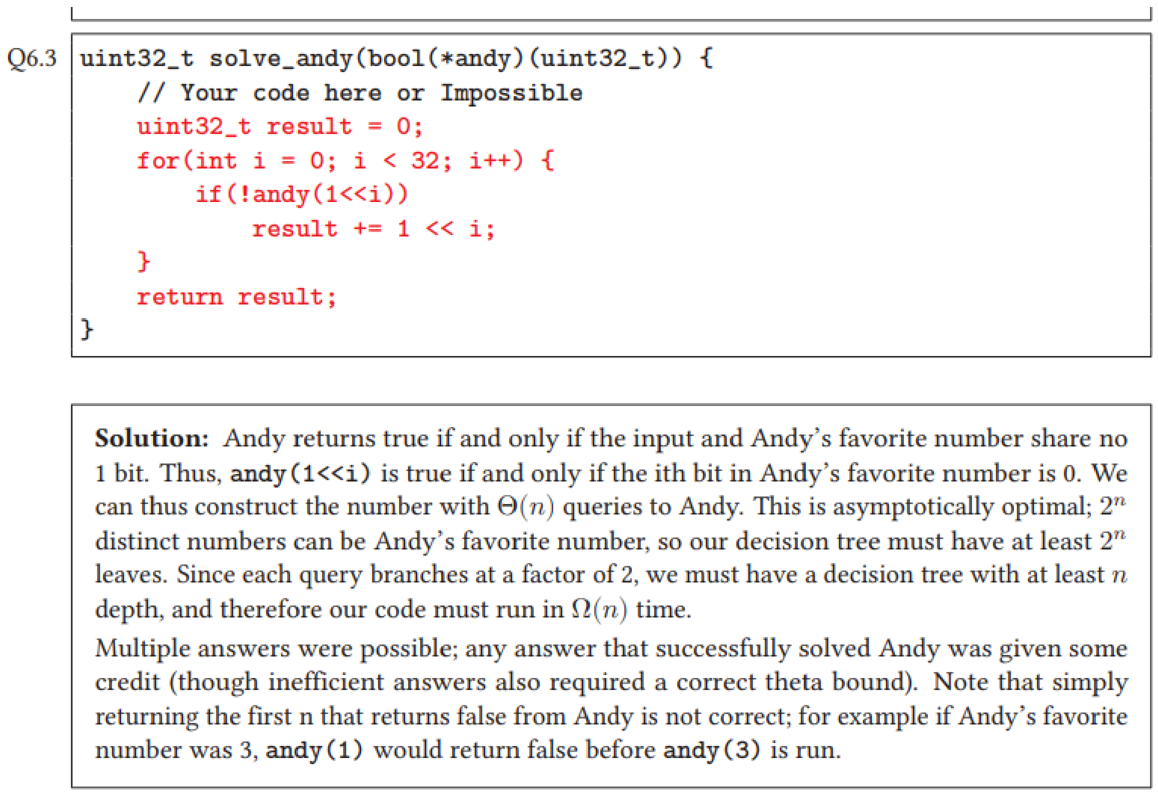
Justin Yokota STAFF 8mth #469acfb

Yes, but xor 1 0 would be 1. This would work only if those top bits happened to be 0xFFFFFF, and it could be 0x000000 instead.

Adrian Bao 8mth #469aafb ✓ Resolved

FA23-Q6.3

Why is 1 << i necessary, since that is just 2^i, and that if the andy call is false, we increment by 1 << i?


Q6.3

```
uint32_t solve_andy(bool(*andy)(uint32_t)) {
    // Your code here or Impossible
    uint32_t result = 0;
    for(int i = 0; i < 32; i++) {
        if(!andy(1<<i))
            result += 1 << i;
    }
    return result;
}
```

**Solution:** Andy returns true if and only if the input and Andy's favorite number share no 1 bit. Thus, `andy(1<<i)` is true if and only if the ith bit in Andy's favorite number is 0. We can thus construct the number with $\Theta(n)$ queries to Andy. This is asymptotically optimal; $2^n$ distinct numbers can be Andy's favorite number, so our decision tree must have at least $2^n$ leaves. Since each query branches at a factor of 2, we must have a decision tree with at least $n$ depth, and therefore our code must run in $\Omega(n)$ time.

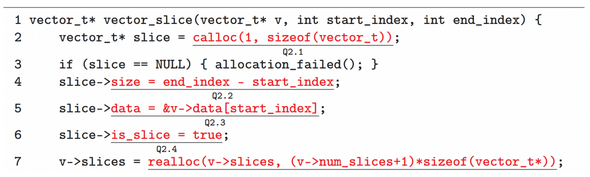
Multiple answers were possible; any answer that successfully solved Andy was given some credit (though inefficient answers also required a correct theta bound). Note that simply returning the first n that returns false from Andy is not correct; for example if Andy's favorite number was 3, `andy(1)` would return false before `andy(3)` is run.

Justin Yokota STAFF 8mth #469abfc

2^i is a completely different number, since ^ is XOR, not POW. The other issue with just defining i as powers of 2 is that it ends up overflowing the int, and therefore infinite-looping.

Anonymous Sardine 8mth #469aaeb ✓ Resolved

FA23 - Q2 : Why do we only do sizeof(vector_t) in line 2 for calloc, but do sizeof(vector_t*) in line 7 for line 7 for realloc? Why is there a star for line 7, but not line 2?



```
1 vector_t* vector_slice(vector_t* v, int start_index, int end_index) {
2     vector_t* slice = calloc(1, sizeof(vector_t));
                                Q2.1
3     if (slice == NULL) { allocation_failed(); }
4     slice->size = end_index - start_index;
                       Q2.2
5     slice->data = &v->data[start_index];
                      Q2.3
6     slice->is_slice = true;
                  Q2.4
7     v->slices = realloc(v->slices, (v->num_slices+1)*sizeof(vector_t*));
```

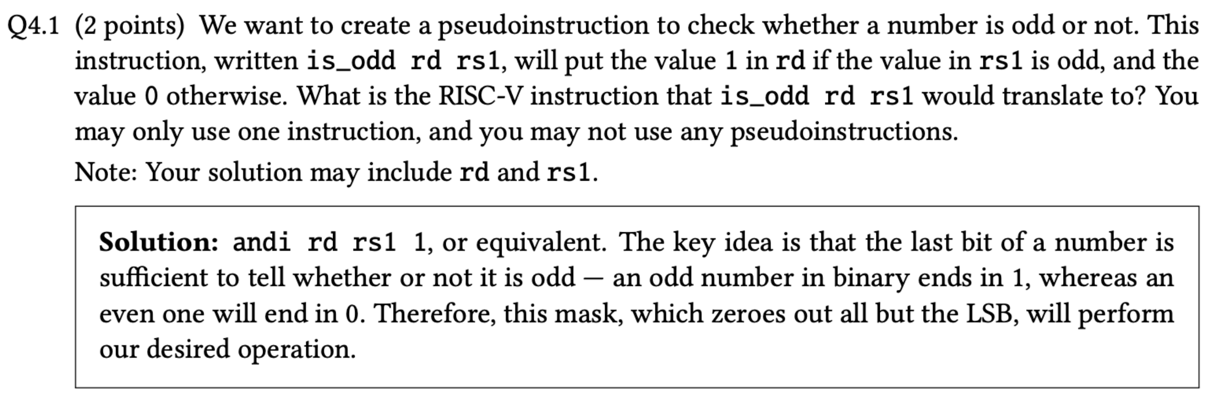
Daniel Wang 8mth #469aaed

For line 2, remember that `calloc` and other memory allocating methods create space for what the pointer is *pointing to*, in other words the `vector_t` struct.

For line 7, `slices` is a pointer to a pointer, meaning we need to allocate space for the pointer, `vector_t*`.

Ben Yu 8mth #469aaea ✓ Resolved

Spring 2023 4.1, how does the **andi** instruction with **imm** 1 zero out all the digits except on LSB? For example, if I let s0 = 0b1100, which is 12 in decimal then **andi rd s0 1b** would assign rd as 1100 because of bitwise AND, which yields a positive result since 1100 = 12 nonzero. In other words, how do we separate LSB out from the result?



Q4.1 (2 points) We want to create a pseudoinstruction to check whether a number is odd or not. This instruction, written `is_odd rd rs1`, will put the value 1 in `rd` if the value in `rs1` is odd, and the value 0 otherwise. What is the RISC-V instruction that `is_odd rd rs1` would translate to? You may only use one instruction, and you may not use any pseudoinstructions.

Note: Your solution may include `rd` and `rs1`.

**Solution:** `andi rd rs1 1`, or equivalent. The key idea is that the last bit of a number is sufficient to tell whether or not it is odd — an odd number in binary ends in 1, whereas an even one will end in 0. Therefore, this mask, which zeroes out all but the LSB, will perform our desired operation.

Justin Yokota STAFF 8mth #469aaec

Are you sure that 0b1100 & 0b0001 is equal to 0b1100?

Ben Yu 8mth #469abab

Oh, so by giving 1 in the **imm** section, we only do bitwise **and** for the LSB? Not operating with every possible digit of **rs1?**

Daniel Wang 8mth #469abef

Replying to Ben Yu

You still would use all digits of rs1, it's just that bitwise AND with a 0-digit will always give 0. So we can think of it like this:

0b1100 & 0b0001 (1 in decimal)
= 0b0000

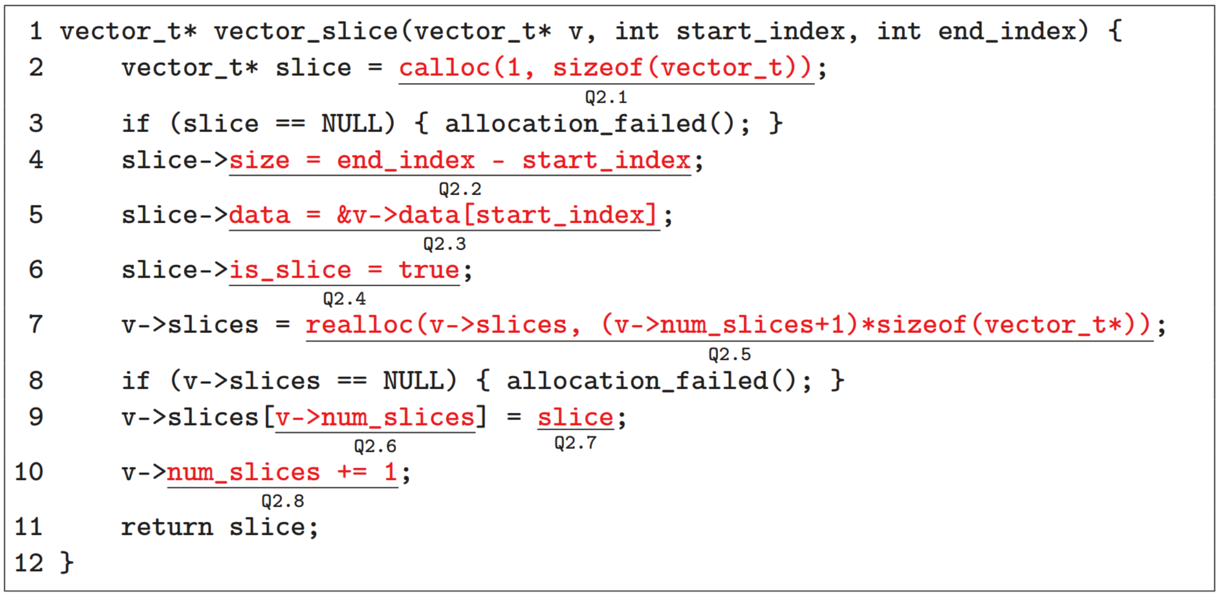
Do you see how this can determine whether or not a digit is even or odd? There should be a pattern when performing AND with 1, which in effect just ANDs the LSB of the number we are ANDing with.

1

Anonymous Snail 8mth #469aada ✓ Resolved

[FA23 Q2]

Would `malloc(sizeof(vector_t))` work for the first line instead of calloc?



```
vector_t* vector_slice(vector_t* v, int start_index, int end_index) {
    vector_t* slice = calloc(1, sizeof(vector_t));
                                      Q2.1
    if (slice == NULL) { allocation_failed(); }
    slice->size = end_index - start_index;
                       Q2.2
    slice->data = &v->data[start_index];
                      Q2.3
    slice->is_slice = true;
                Q2.4
    v->slices = realloc(v->slices, (v->num_slices+1)*sizeof(vector_t*));
                                           Q2.5
    if (v->slices == NULL) { allocation_failed(); }
    v->slices[v->num_slices] = slice;
                  Q2.6          Q2.7
    v->num_slices += 1;
            Q2.8
    return slice;
}
```

Nikhil Kandkur STAFF 8mth #469aadd

We cannot do `malloc`, since we want to set slices->num_slices to zero since we are not slicing the new slice, and the only way to guarantee this is by callocing the struct.

1

Daniel Wang 8mth #469aaee

Is this because `malloc` doesn't allow us to access the fields of the struct?

Erik Yang STAFF 8mth #469aaef

Replying to Daniel Wang

Yes, if we malloc, then you will have unintialized data that we cannot access yet

2

Daniel Wang 8mth #469aafa

Replying to Erik Yang

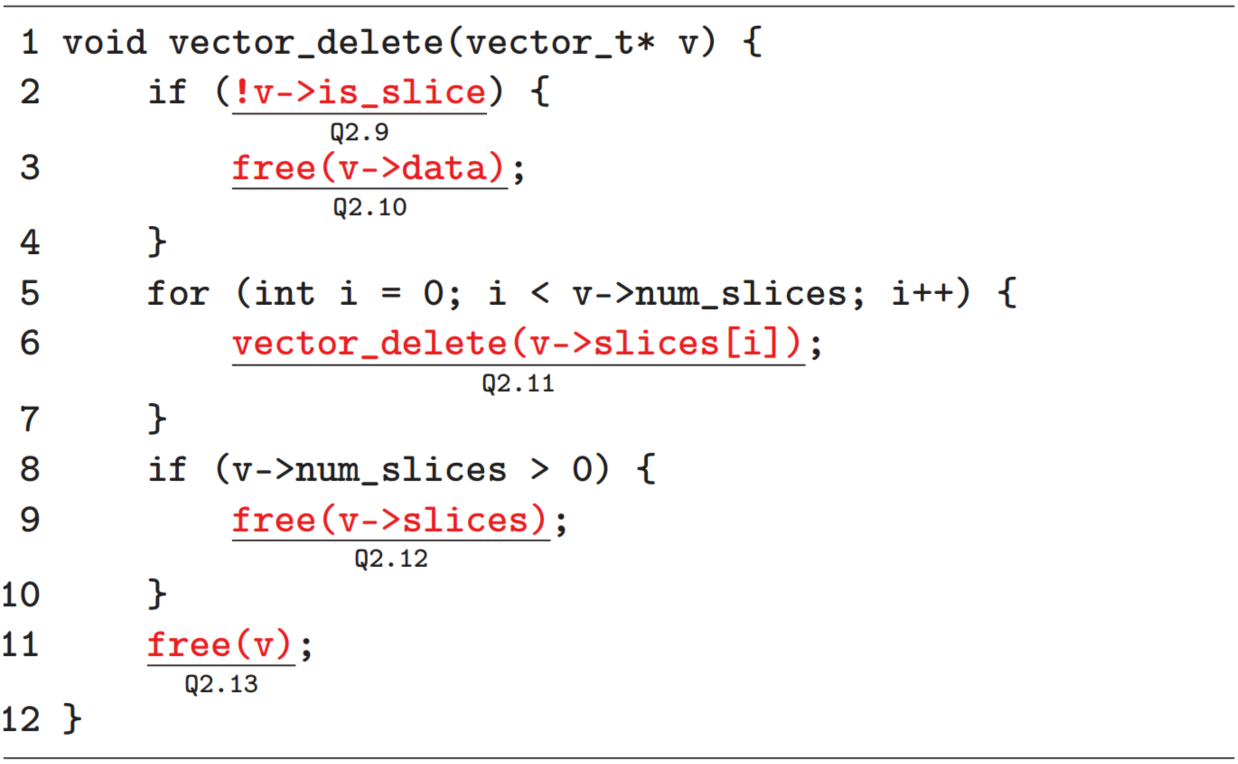
Thanks!

Anonymous Dinosaur 8mth #469aacc ✓ Resolved

**FA23-MT-Q2**

i dont really get line 2: !v->is_slice

why are we freeing the data if v isn't a child slice of another vector?



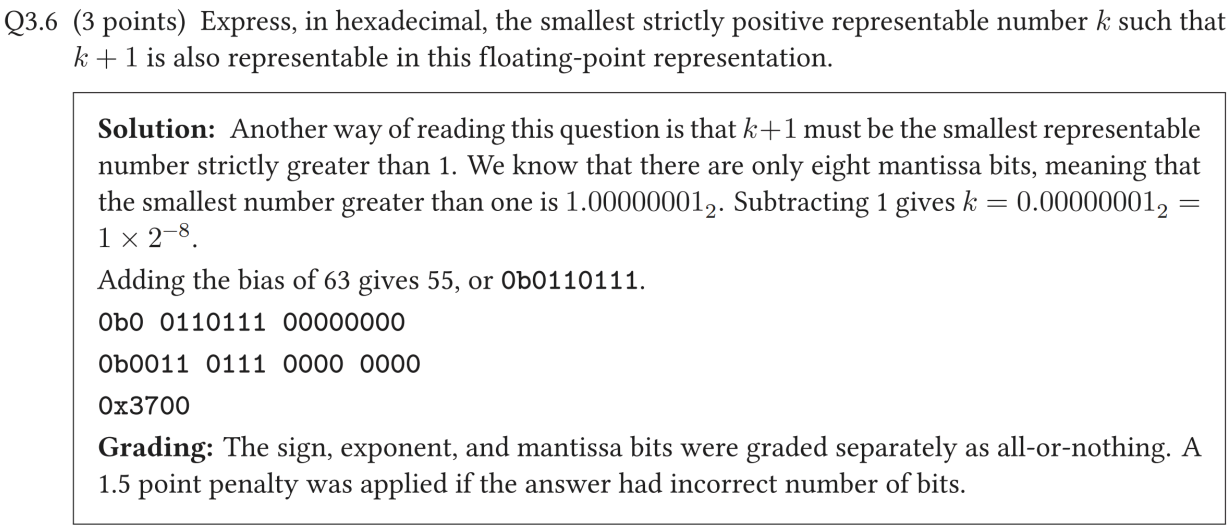
```
void vector_delete(vector_t* v) {
    if (!v->is_slice) {
         Q2.9
        free(v->data);
          Q2.10
    }
    for (int i = 0; i < v->num_slices; i++) {
        vector_delete(v->slices[i]);
                  Q2.11
    }
    if (v->num_slices > 0) {
        free(v->slices);
           Q2.12
    }
    free(v);
     Q2.13
}
```

Sophie Xie STAFF 8mth #469aadb

If v is a child slice of another vector, then its `data` is a portion of the `data` array of its parent vector. Freeing `v->data` of a parent vector also frees the `data` of all child vectors, so we don't need to free `v->data` if it's a child slice. We need to free `v->data` if v isn't a child slice of another vector because it's a pointer to the start of v's `data` array.

Anonymous Jellyfish 8mth #469aacb ✓ Resolved

Fa23-MT-q3



Q3.6 (3 points) Express, in hexadecimal, the smallest strictly positive representable number $k$ such that $k+1$ is also representable in this floating-point representation.

**Solution:** Another way of reading this question is that $k+1$ must be the smallest representable number strictly greater than 1. We know that there are only eight mantissa bits, meaning that the smallest number greater than one is $1.00000001_2$. Subtracting 1 gives $k = 0.00000001_2 = 1 \times 2^{-8}$.

Adding the bias of 63 gives 55, or `0b0110111`.

`0b0 0110111 00000000`

`0b0011 0111 0000 0000`

`0x3700`

**Grading:** The sign, exponent, and mantissa bits were graded separately as all-or-nothing. A 1.5 point penalty was applied if the answer had incorrect number of bits.

I thought this question was asking basically find the smallest number with the step size of one so we can represent k+1. But looking at solutions, I have no idea what it was asking. Plz explain.

Myrah Shah STAFF 8mth #469acac

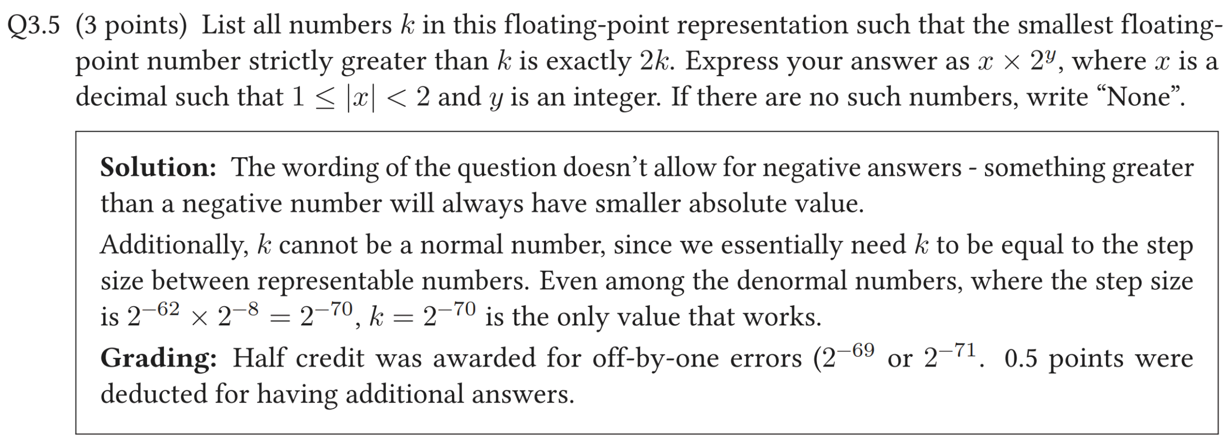
#469abfd

#469cb

♡ ···

Anonymous Jellyfish 8mth #469aabd ✓ Resolved

Fa23-MT-q3



Q3.5 (3 points) List all numbers $k$ in this floating-point representation such that the smallest floating-point number strictly greater than $k$ is exactly $2k$. Express your answer as $x \times 2^y$, where $x$ is a decimal such that $1 \leq |x| < 2$ and $y$ is an integer. If there are no such numbers, write “None”.

**Solution:** The wording of the question doesn’t allow for negative answers - something greater than a negative number will always have smaller absolute value.

Additionally, $k$ cannot be a normal number, since we essentially need $k$ to be equal to the step size between representable numbers. Even among the denormal numbers, where the step size is $2^{-62} \times 2^{-8} = 2^{-70}$, $k = 2^{-70}$ is the only value that works.

**Grading:** Half credit was awarded for off-by-one errors ($2^{-69}$ or $2^{-71}$. 0.5 points were deducted for having additional answers.

I don't understand this.

1. Don't we need all 1s in the significand so that the next largest number is the next power which is 2*last num?

2. What does it mean by we need $k$ to be equal to the step size? Why?

♡ 1 ···

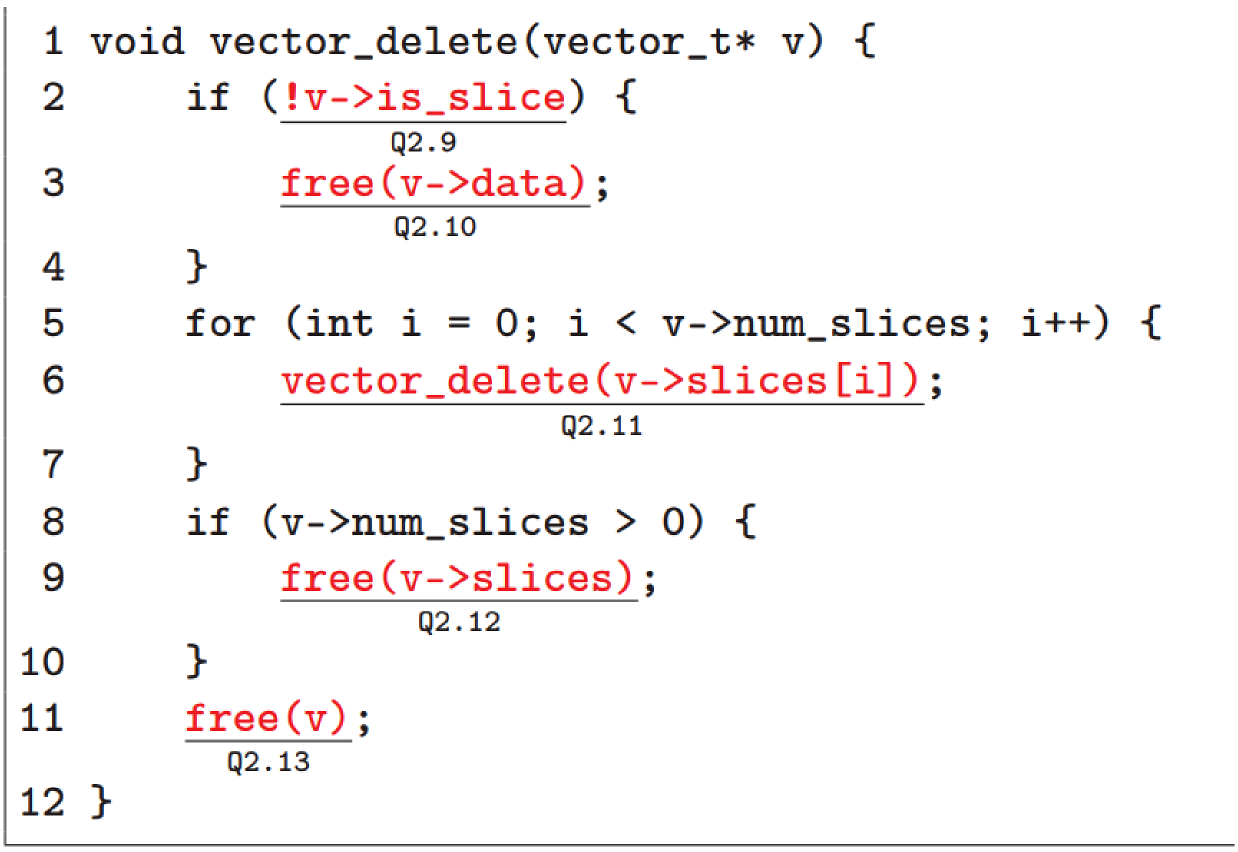
J Justin Yokota STAFF 8mth #469aadf

1. Not necessarily. For example, 3.99999 is between 2 and 4, while 4.000001 is between 4 and 8. So 3.999999 is in the "previous power of two", but is almost the same number as 4.000001.

♡ ···

Anonymous Jellyfish 8mth #469aabb ✓ Resolved

Fa23-MT-Q2



```
 1 void vector_delete(vector_t* v) {
 2     if (!v->is_slice) {
              Q2.9
 3         free(v->data);
              Q2.10
 4     }
 5     for (int i = 0; i < v->num_slices; i++) {
 6         vector_delete(v->slices[i]);
                       Q2.11
 7     }
 8     if (v->num_slices > 0) {
 9         free(v->slices);
               Q2.12
10     }
11     free(v);
        Q2.13
12 }
```

why do we need line 9? In line 6 we are already freeing v->slices[i] so why do we need to free more?

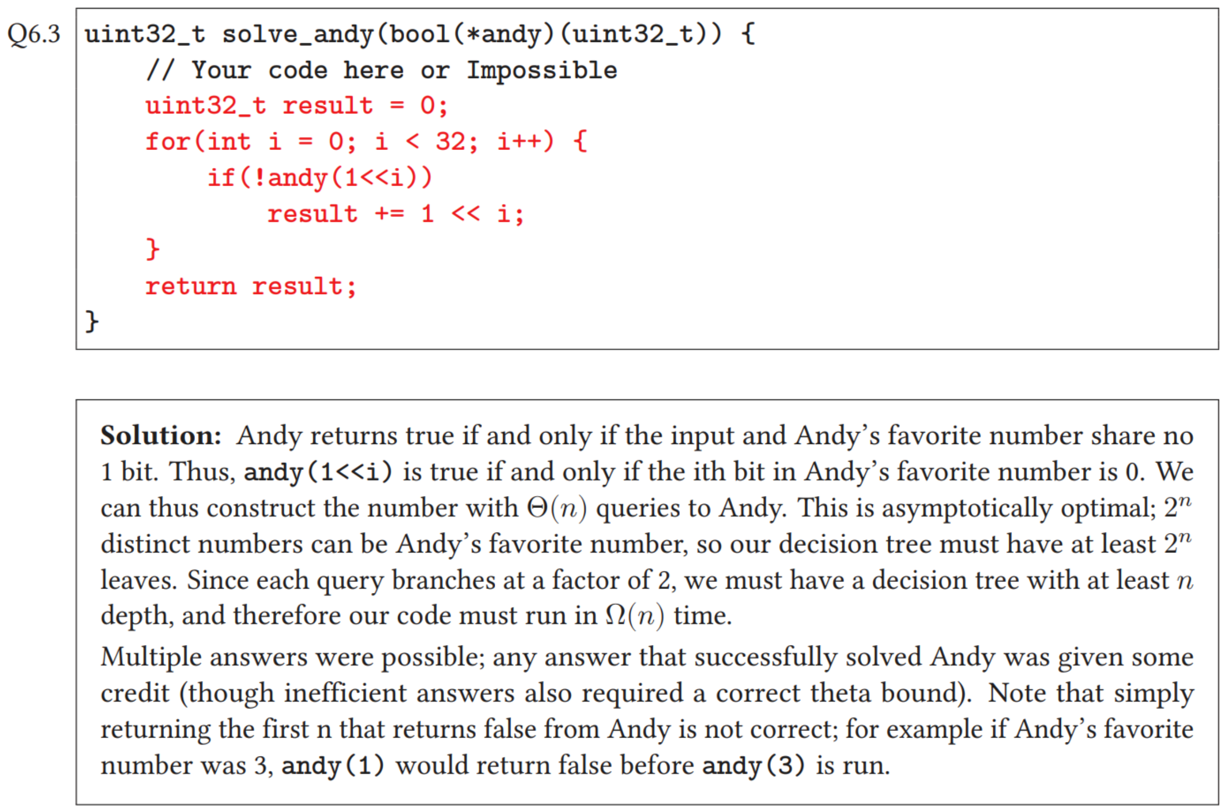
Sophie Xie STAFF 8mth #469aabc

We need to free v->slices because v->slices is an array of vector_t structs. So, even though we freed v->slices[i] that only freed the vector_t structs in v->slices, not the actual array itself.

1

Anonymous Shrew 8mth #469aaba ✓ Resolved

In this question, **FA23-MT-Q 6.3** Is talking about runtime error like theta of n is relevant in our scope? Also, can someone clarify this please, with the theta/run time error. Also the queries and trees, branches, and leaves stuff, are they relevant to our stuff? I am confused, mostly on shifting and why left shift needed?


Q6.3
```
uint32_t solve_andy(bool(*andy)(uint32_t)) {
    // Your code here or Impossible
    uint32_t result = 0;
    for(int i = 0; i < 32; i++) {
        if(!andy(1<<i))
            result += 1 << i;
    }
    return result;
}
```

**Solution:** Andy returns true if and only if the input and Andy's favorite number share no 1 bit. Thus, `andy(1<<i)` is true if and only if the ith bit in Andy's favorite number is 0. We can thus construct the number with $\Theta(n)$ queries to Andy. This is asymptotically optimal; $2^n$ distinct numbers can be Andy's favorite number, so our decision tree must have at least $2^n$ leaves. Since each query branches at a factor of 2, we must have a decision tree with at least $n$ depth, and therefore our code must run in $\Omega(n)$ time.

Multiple answers were possible; any answer that successfully solved Andy was given some credit (though inefficient answers also required a correct theta bound). Note that simply returning the first n that returns false from Andy is not correct; for example if Andy's favorite number was 3, `andy(1)` would return false before `andy(3)` is run.

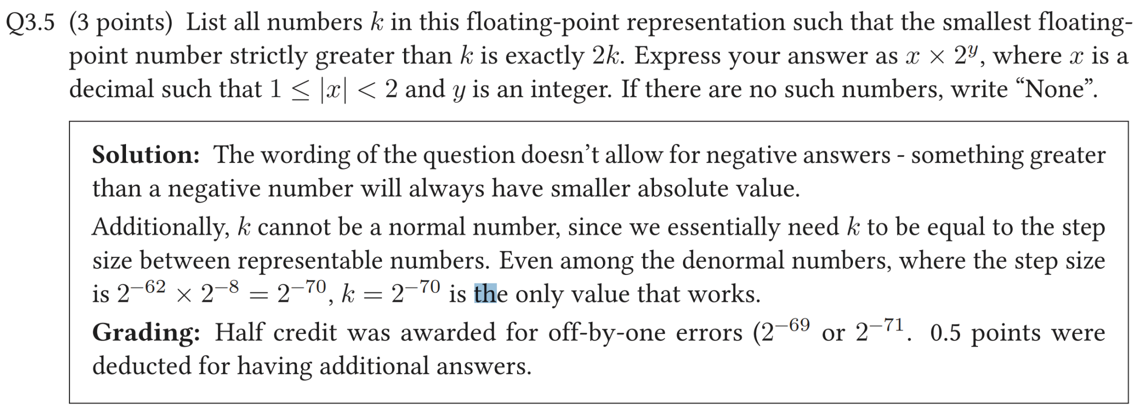
J

Justin Yokota STAFF 8mth #469aabe

Yes, any material covered in prerequisite classes are considered fair game. Asymptotic runtime analysis was covered in 61B. The decision tree analysis is more for the proof that Theta(N) is optimal; it wasn't strictly necessary to solve the problem, and is more to prove definitively that there are no better solutions.

Anonymous Echidna 8mth #469aaac ✓ Resolved

FA23-MT-Q3.5


Q3.5 (3 points) List all numbers $k$ in this floating-point representation such that the smallest floating-point number strictly greater than $k$ is exactly $2k$. Express your answer as $x \times 2^y$, where $x$ is a decimal such that $1 \leq |x| < 2$ and $y$ is an integer. If there are no such numbers, write "None".

**Solution:** The wording of the question doesn't allow for negative answers - something greater than a negative number will always have smaller absolute value.

Additionally, $k$ cannot be a normal number, since we essentially need $k$ to be equal to the step size between representable numbers. Even among the denormal numbers, where the step size is $2^{-62} \times 2^{-8} = 2^{-70}$, $k = 2^{-70}$ is the only value that works.

**Grading:** Half credit was awarded for off-by-one errors ($2^{-69}$ or $2^{-71}$. 0.5 points were deducted for having additional answers.

Why isn't it none since technically if x is 1<= x <= 2, then 2x would go above 2, so it's not possible

A

Andrea Lou STAFF 8mth #469aaaf

The answer is in the form (x * 2^y) not just x, kinda like scientific notation with num * 10^(something)

Anonymous Yak 8mth #469aaab ✓ Resolved

SP23-MT-Q2.3

Why do we have to get the address of sheet using &sheet? I thought -> works on pointers, and sheet already is a pointer.

M Myrah Shah STAFF 8mth #469aaad

#469ccf

Anonymous Squid 8mth #469fff ✓ Resolved

SP23-MT-Q5

What does each edge represent?

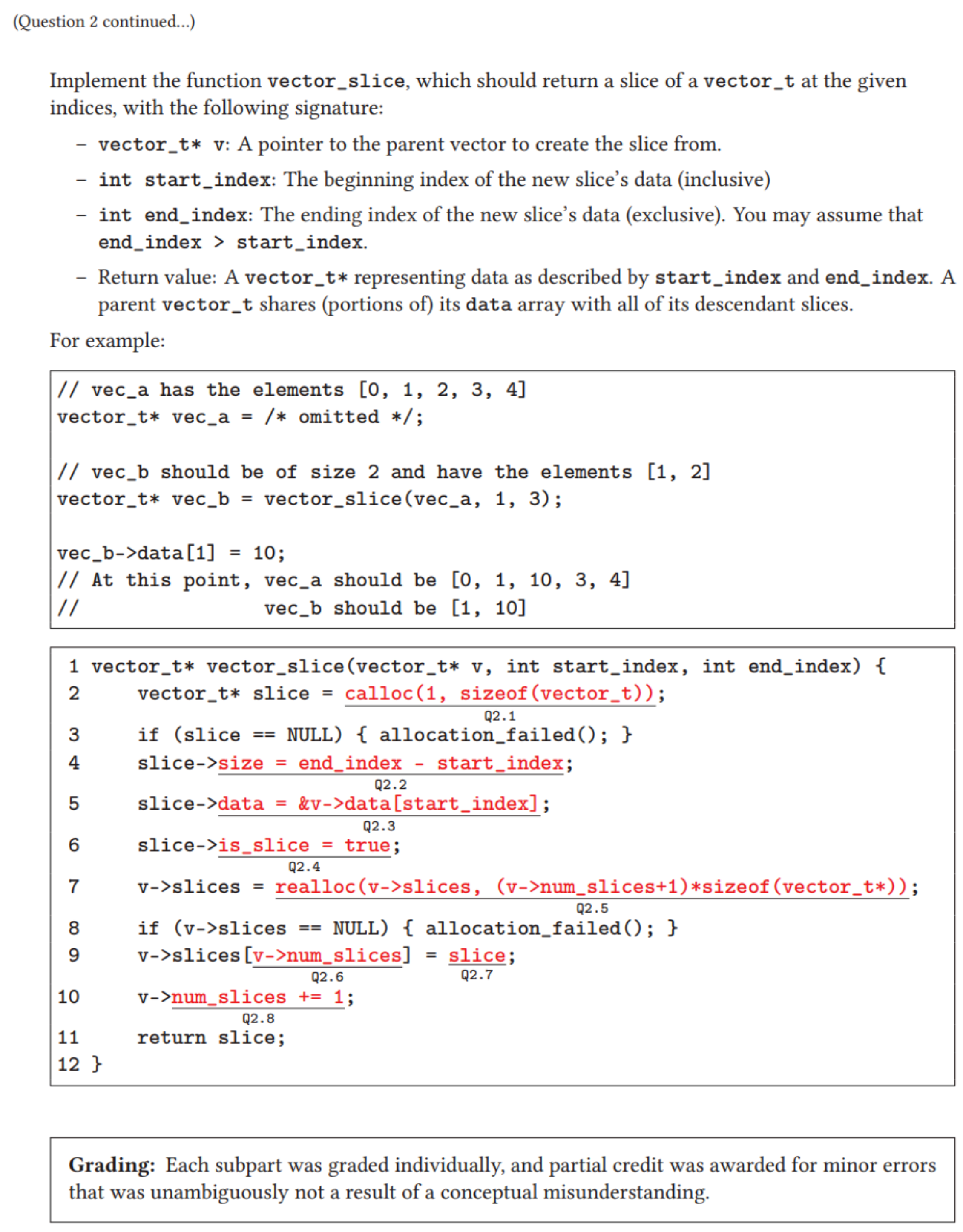
M Myrah Shah STAFF 8mth #469aaae

FSMs are not in scope.

Anonymous Echidna 8mth #469ffd ✓ Resolved

FA23-MT-Q2

Can we use malloc or just calloc? And why?



(Question 2 continued...)

Implement the function `vector_slice`, which should return a slice of a `vector_t` at the given indices, with the following signature:

- `vector_t* v`: A pointer to the parent vector to create the slice from.
- `int start_index`: The beginning index of the new slice's data (inclusive)
- `int end_index`: The ending index of the new slice's data (exclusive). You may assume that `end_index > start_index`.
- Return value: A `vector_t*` representing data as described by `start_index` and `end_index`. A parent `vector_t` shares (portions of) its `data` array with all of its descendant slices.

For example:

```
// vec_a has the elements [0, 1, 2, 3, 4]
vector_t* vec_a = /* omitted */;

// vec_b should be of size 2 and have the elements [1, 2]
vector_t* vec_b = vector_slice(vec_a, 1, 3);

vec_b->data[1] = 10;
// At this point, vec_a should be [0, 1, 10, 3, 4]
//                vec_b should be [1, 10]
```

```
vector_t* vector_slice(vector_t* v, int start_index, int end_index) {
    vector_t* slice = calloc(1, sizeof(vector_t));
                              Q2.1
    if (slice == NULL) { allocation_failed(); }
    slice->size = end_index - start_index;
                       Q2.2
    slice->data = &v->data[start_index];
                      Q2.3
    slice->is_slice = true;
                Q2.4
    v->slices = realloc(v->slices, (v->num_slices+1)*sizeof(vector_t*));
                                          Q2.5
    if (v->slices == NULL) { allocation_failed(); }
    v->slices[v->num_slices] = slice;
                  Q2.6          Q2.7
    v->num_slices += 1;
            Q2.8
    return slice;
}
```

**Grading:** Each subpart was graded individually, and partial credit was awarded for minor errors that was unambiguously not a result of a conceptual misunderstanding.

S

Sophie Xie STAFF 8mth #469ffe

#469ebf

Anonymous Quelea 8mth #469fec ✓ Resolved

SP23-MT-Q6

Why do we look for the shortest path between two elements for maximum hold time?

Q6.2 (3 points) What is the maximum hold time the registers can have so that there are no hold time violations in the circuit above? Reminder: you may assume that `Input` will not cause any hold time violations.

**Solution:** 25 ps

The shortest path between any two timed elements is actually the path from the SEL signal, which changes instantly at the rising edge of the clock, to the right register. This path has only delay 25 ps from the mux.

If you didn't see this path, the next-shortest path starts from the rightmost register and goes around, through the NOT gate, to the top-left register. This path has a delay of 30 ps (clk-to-q from the rightmost register) and 8 ps (from the NOT gate), for a total of 38 ps. Partial credit was given for this answer.

E Erik Yang STAFF 8mth #469fee

hold time <= shortest path + clk-to-q; this is because if hold time was any longer than this, the input (for the next clock cycle) would not change when it needs to

Anonymous Quelea 8mth #469fef

i'm still kinda confused how they got 25ps when there are smaller numbers such as 8 and 2 ps for NOT and Shifter?

if the shortest path was 25 then shouldn't they have added 25+30 = 55ps by adding the clk-to-q?

E Erik Yang STAFF 8mth #469ffb

Replying to Anonymous Quelea

Note: SEL is a single bit control signal that updates instantaneously at the rising edge of every clock cycle and remains stable during any given clock cycle. (from the question)

This behavior is similar to any other path, but has no clk-to-q time since the signal updates instantly. This means that we do not have to factor in clk-to-q here.

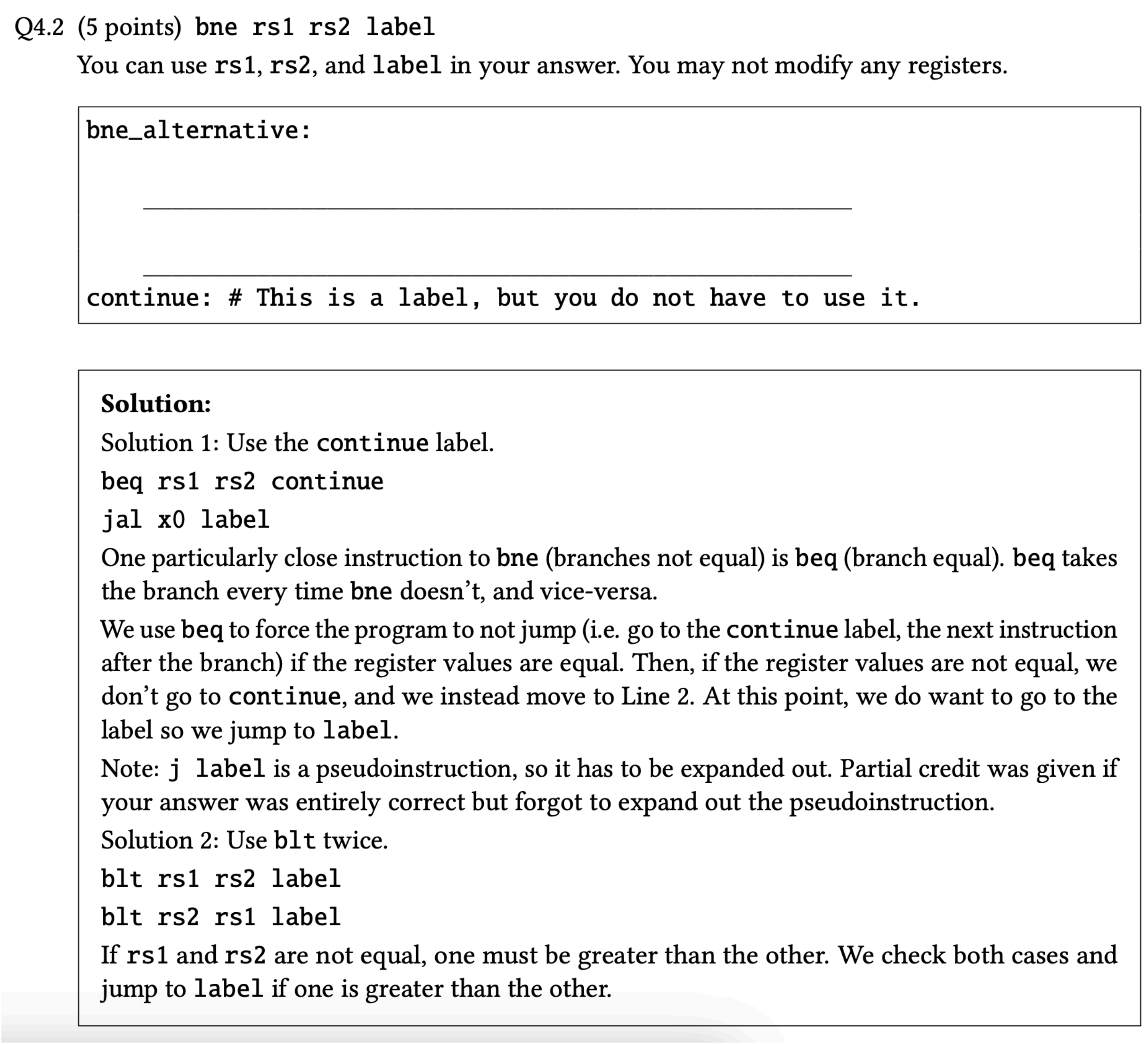
1

Anonymous Quelea 8mth #469aaaa

Replying to Erik Yang

oh okay thanks :)

Anonymous Quelea 8mth #469fde ✓ Resolved


Q4.2 (5 points) `bne rs1 rs2 label`

You can use `rs1`, `rs2`, and `label` in your answer. You may not modify any registers.

```
bne_alternative:

    ____________________________________________________

    ____________________________________________________
continue: # This is a label, but you do not have to use it.
```

**Solution:**

Solution 1: Use the `continue` label.

```
beq rs1 rs2 continue
jal x0 label
```

One particularly close instruction to `bne` (branches not equal) is `beq` (branch equal). `beq` takes the branch every time `bne` doesn't, and vice-versa.

We use `beq` to force the program to not jump (i.e. go to the `continue` label, the next instruction after the branch) if the register values are equal. Then, if the register values are not equal, we don't go to `continue`, and we instead move to Line 2. At this point, we do want to go to the label so we jump to `label`.

Note: `j label` is a pseudoinstruction, so it has to be expanded out. Partial credit was given if your answer was entirely correct but forgot to expand out the pseudoinstruction.

Solution 2: Use `blt` twice.

```
blt rs1 rs2 label
blt rs2 rs1 label
```

If `rs1` and `rs2` are not equal, one must be greater than the other. We check both cases and jump to `label` if one is greater than the other.

SP23-MT-Q4

Why do we do jal x0 label? I though x0 could not be changed because it contains the value 0?

…

E Erik Yang STAFF 8mth #469fed

jal x0 label is the psuedoinstruction for j label - it means jump to label but don't return anywhere

1 …

Anonymous Quelea 8mth #469ffa

ahh i see okay thank you!

…

Anonymous Quail 8mth #469fdc ✓ Resolved

SP23-MT-Q2.10

NUM_PAGES is defined using the define directive, which I thought was only used for constants. Therefore, why does the answer say that NUM_PAGES is stored in code, rather than data?

…

M Myrah Shah STAFF 8mth #469fdd

#469cdd

Anonymous Turtle 8mth #469fcc ✓ Resolved

Su23 4.1

Why do we use slli on line 6 but srai on line 11?

Andrea Lou STAFF 8mth #469fce

The slli (shift left) on line 6 shifts the number left by 1 bit, or multiplies it by two. This is to set up for the (3n + 1) end product by doing (2n + n + 1)

The srai (shift right arithmetic) on line 11 shifts the number right by one bit, or divides it by two for the desired n/2

Anonymous Turtle 8mth #469fcf

Why is one arithmetic but the other logical?

Andrea Lou STAFF 8mth #469fdb

Replying to Anonymous Turtle

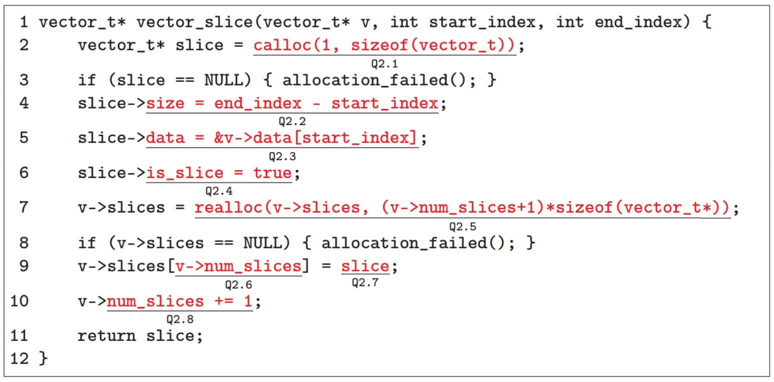
Note that we don't actually have an arithmetic left shift in the reference card. This is because left shifting preserves the sign either way, which is what we care about when multiplying the number by an immediate.

We use the arithmetic right shift in case we get a negative even number, so that dividing the number by two preserves the sign.

1

Anonymous Buffalo 8mth #469fbf ✓ Resolved

FA23-MT-Q2



```
1 vector_t* vector_slice(vector_t* v, int start_index, int end_index) {
2     vector_t* slice = calloc(1, sizeof(vector_t));
                                Q2.1
3     if (slice == NULL) { allocation_failed(); }
4     slice->size = end_index - start_index;
                       Q2.2
5     slice->data = &v->data[start_index];
                      Q2.3
6     slice->is_slice = true;
                  Q2.4
7     v->slices = realloc(v->slices, (v->num_slices+1)*sizeof(vector_t*));
                                           Q2.5
8     if (v->slices == NULL) { allocation_failed(); }
9     v->slices[v->num_slices] = slice;
                  Q2.6          Q2.7
10    v->num_slices += 1;
              Q2.8
11    return slice;
12 }
```

I was wondering why in line 5 we cannot put v->data + start_index? I saw a thread about this earlier, but am still a bit confused, because doesn't this account for the size of the elements as well? For example, if v->data was at 0xFFFFFFF0 then v->data + 1 would be 0xFFFFFFF4 (assuming it's an int array and an int is 4 bytes)?

Erik Yang STAFF 8mth #469fdf

this is still valid since it is pointer arithmetic

Annika Liu 8mth #469fbd ✓ Resolved

FA23-MT-Q3

Hello, for Q3.5, I don't really understand how we got 2^-70 from 2^-62 * 2^-8. Thank you!

Andrea Lou STAFF 8mth #469fca

From the problem description, we have a standard bias of -63. With denorm numbers, the exponent comes out to 2^(exp + bias + 1), or 2^(-62). The 2^(-8) comes from the 8 significand bits, since we want the step size between denorm numbers.

2^(-62) * 2^(-8) = 2^(-62 + -8) = 2^(-70)

Annika Liu 8mth #469fcb

I'm sorry I think I'm still not too sure about the concept behind the answer. Why do we need the step size between denorm numbers and why is it 2^(-8)? Thanks!

Andrea Lou STAFF 8mth #469fcd

Replying to Annika Liu

The problem asks for a number k such that the next smallest number is 2k. In order for this to happen, there can't be any representable numbers between k and 2k, thus the step size must equal the distance between k and 2k, which is just (2k - k = k).

It's 2^(-8) because of the 8 significand bits. If you imagine trying to increment the number by the smallest amount possible, you'd add one to the rightmost smallest bit of the significand -> 0.00000001 or 2^(-8)

1

Annika Liu 8mth #469acab

Replying to Andrea Lou

I understand that this is the smallest floating point representable in this system such that the smallest floating point number strictly greater than k is exactly 2k. However according to the question's descriptions, I don't understand why do we have to look for the case where the number k is the smallest representable. Isn't it also the case that when the least significant bit of the mantissa is pushed forward to become the 1st positive bit, step size is also 2? Thank you!

Andrea Lou STAFF 8mth #469acad

Replying to Annika Liu

I think for this problem, the thing to focus on is that the step size must equal k. In your example when the step size is 2, k would need to be enormous, so (number + 2) wouldn't equal (number * 2)

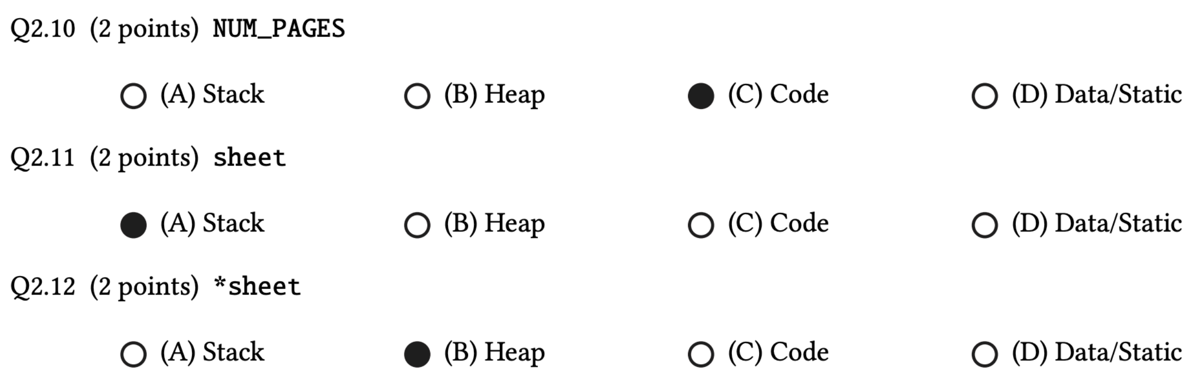
You can see this in one of the past homeworks, there was a question on when we start "skipping" whole numbers, or when step size is 2. At that point, we already have numbers in the millions

1

Anonymous Quelea 8mth #469fba ✓ Resolved

SP23-MT-Q2

can someone explain why this is the case?



Q2.10 (2 points) `NUM_PAGES`

(A) Stack (B) Heap [✓ (C) Code] (D) Data/Static

Q2.11 (2 points) `sheet`

[✓ (A) Stack] (B) Heap (C) Code (D) Data/Static

Q2.12 (2 points) `*sheet`

(A) Stack [✓ (B) Heap] (C) Code (D) Data/Static

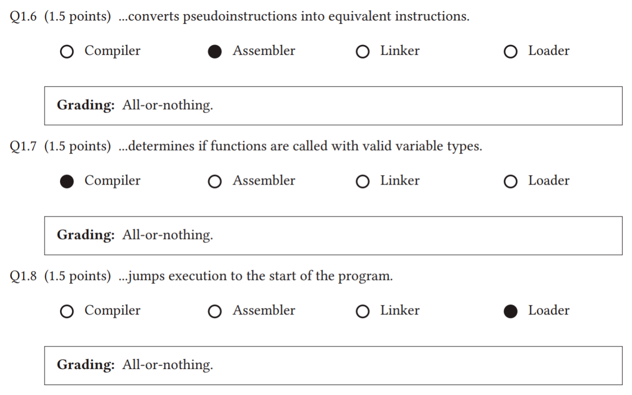
Myrah Shah STAFF 8mth #469fbb

Can you take a look at the Memory Management section of this discussion and follow up on which part is confusing you?

Anonymous Echidna 8mth #469fad ✓ Resolved

FA23-MT-Q1.6-8

Can someone explain why for each option?



Q1.6 (1.5 points) ...converts pseudoinstructions into equivalent instructions.

Compiler [✓ Assembler] Linker Loader

**Grading:** All-or-nothing.

Q1.7 (1.5 points) ...determines if functions are called with valid variable types.

[✓ Compiler] Assembler Linker Loader

**Grading:** All-or-nothing.

Q1.8 (1.5 points) ...jumps execution to the start of the program.

Compiler Assembler Linker [✓ Loader]

**Grading:** All-or-nothing.

Myrah Shah STAFF 8mth #469fbe

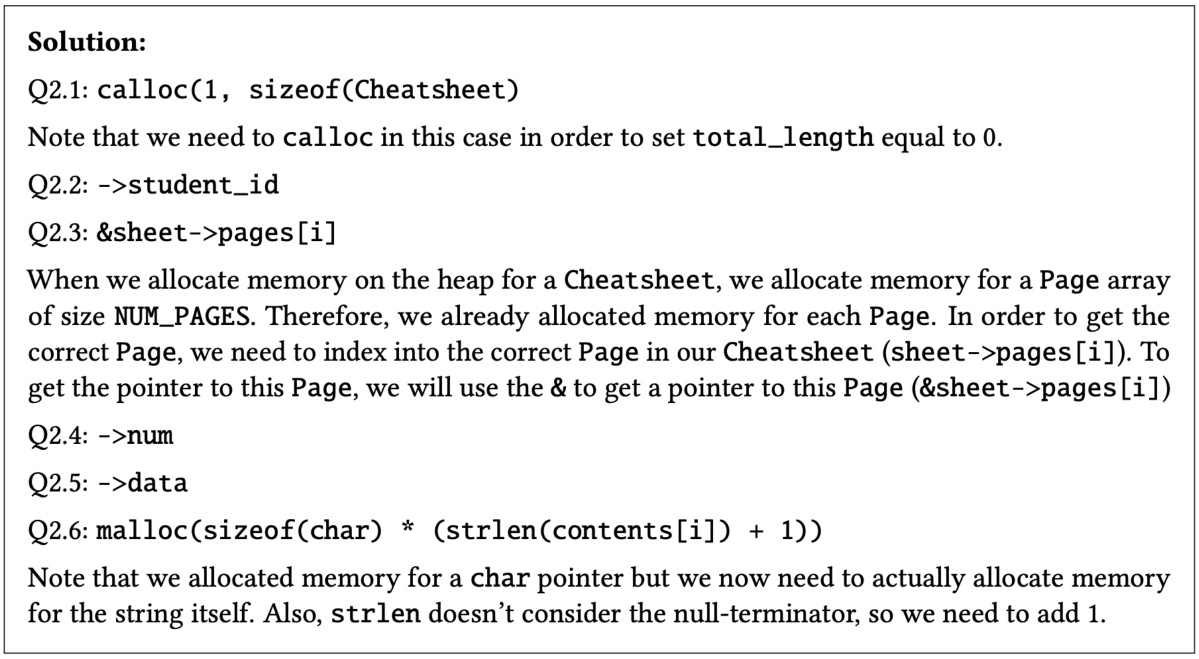
I would recommend taking a look at Lecture 14, it covers these examples and also goes more in-depth as to what each stage does. Feel free to ask follow-up questions!

Anonymous Quelea 8mth #469eff ✓ Resolved

SP23-MT-Q2

What is the significance of -> here?

Why do we need to calloc specifically in this case in order to set total_length equal to 0? Would that not happen with malloc?



**Solution:**

Q2.1: `calloc(1, sizeof(Cheatsheet)`

Note that we need to `calloc` in this case in order to set `total_length` equal to 0.

Q2.2: `->student_id`

Q2.3: `&sheet->pages[i]`

When we allocate memory on the heap for a `Cheatsheet`, we allocate memory for a `Page` array of size `NUM_PAGES`. Therefore, we already allocated memory for each `Page`. In order to get the correct `Page`, we need to index into the correct `Page` in our `Cheatsheet` (`sheet->pages[i]`). To get the pointer to this `Page`, we will use the `&` to get a pointer to this `Page` (`&sheet->pages[i]`)

Q2.4: `->num`

Q2.5: `->data`

Q2.6: `malloc(sizeof(char) * (strlen(contents[i]) + 1))`

Note that we allocated memory for a `char` pointer but we now need to actually allocate memory for the string itself. Also, `strlen` doesn't consider the null-terminator, so we need to add 1.

Erik Yang STAFF 8mth #469fac

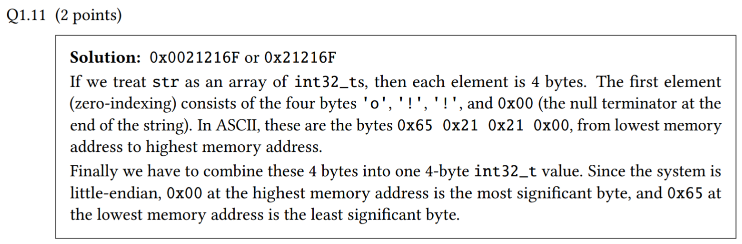
the arrow notation is used for accessing the fields of a struct **pointer**. Calloc automatically creates the memory and initializes ALL data with 0. Malloc only creates the memory

1

Anonymous Quelea 8mth #469fae

ahhh i see okay thank you so much

Anonymous Crane 8mth #469eed ✓ Resolved



Q1.11 (2 points)

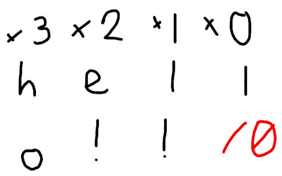
**Solution:** `0x0021216F` or `0x21216F`

If we treat `str` as an array of `int32_ts`, then each element is 4 bytes. The first element (zero-indexing) consists of the four bytes `'o'`, `'!'`, `'!'`, and `0x00` (the null terminator at the end of the string). In ASCII, these are the bytes `0x65 0x21 0x21 0x00`, from lowest memory address to highest memory address.

Finally we have to combine these 4 bytes into one 4-byte `int32_t` value. Since the system is little-endian, `0x00` at the highest memory address is the most significant byte, and `0x65` at the lowest memory address is the least significant byte.

SP23-MT-Q1.11

How exactly is the str 'hello!!' stored in little endian? I thought it was like this? So shouldn't str[1] be read 'o!!/0' ?



Daniel Wang 8mth #469efe

Little-endian is a way of representing our address, but not the actual value of the address itself. So when we take the 1 index, we are still accessing the `char` array from its point in memory + `1 * sizeof(int32_t)` or `pointer + 1 * 4`. This grabs us the 5th byte due to zero-indexing or `o`. However, since each element of our casted `int32_t` pointer has a size of 4 bytes, we need to grab 4 bytes worth of memory, or 4 `chars` each of size 1 byte, `o`, `!`, `!`, `\0`. Converting this into hexadecimal, we have `0x6F, 0x21, 0x21, 0x00`.

Then representing this address, we can now use little-endian where the LSB is at the lowest address, meaning the `0x6F` byte will be stored first, giving `0x0021216F.`

Anonymous Crane 8mth #469faf

I see! But why is 0x6F the LSB, and why is it stored first? Where can I find more information on this?

Daniel Wang 8mth #469fbc

Replying to Anonymous Crane

For pointers, successive elements have higher memory addresses. So the first element, `o` ( `0x6F` ), will have the lowest memory address. Therefore, we will make it the LSB or the element stored first.

Lecture 3 goes into this as well as HW 1, specifically HW1.7. Memory Endianness. Let me know if anything else is unclear!

Anonymous Hornet 8mth #469abbd

Replying to Daniel Wang

Is it always the case (regardless of endianess) that for pointers, successive elements have higher memory addresses?

Daniel Wang 8mth #469abfa

Replying to Anonymous Hornet

I believe so yes. This is because when we construct an array, we start from the the arrays address (pointing to the element at index 0), and successively add the size of the array's type to this address in order to accommodate more items. This means subsequent elements will have higher memory addresses.

Anonymous Quelea 8mth #469edf ✓ Resolved

SP23-MT-Q1.7

How did we simplify to get this?

I got stuck at ~Y | (Y*Z)?

Q1.7 (3 points) Write a Boolean expression that evaluates to the truth table below. You may use at most 2 Boolean operations. ~ (NOT), | (OR), & (AND) each count as one operation. We will assume standard C operator precedence, so use parentheses when uncertain.

| W | Y | Z | Out |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Solution:** `~Y | Z`
Other solutions may exist.

Daniel Wang 8mth #469eea

Using the distributive property in the OR form, we can simplify this as $(\sim Y + Y)(\sim Y + Z)$, which then simplifies to $\sim Y + Z$ or $\sim Y|Z$.

1

Anonymous Quelea 8mth #469eeb

oh okay thank you

Erik Yang STAFF 8mth #469eee

!W*!Y*!Z + !W*!Y*Z + !W*Y*Z + W*!Y* !Z + W*!Y*Z + W*Y*Z =

!W!Y * (!Z + Z) + YZ * (!W + W) + W!Y (!Z + Z)

= !W!Y + YZ + W!Y

= !Y (!W +W) + YZ

= !Y + YZ

!Y = !Y + !YZ using the absorption law; we can plug this into !Y

= !Y + !YZ + YZ

= !Y + Z*(!Y + Y)

= !Y + Z

1

Anonymous Quelea 8mth #469efa

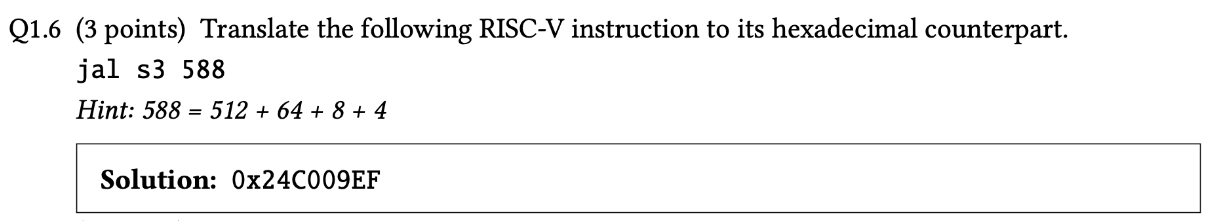
Replying to Erik Yang

thank you :)

Anonymous Quelea 8mth #469ede ✓ Resolved

SP23-MT-Q1

How did they get this?



Q1.6 (3 points) Translate the following RISC-V instruction to its hexadecimal counterpart.

```
jal s3 588
```

*Hint: 588 = 512 + 64 + 8 + 4*

**Solution:** `0x24C009EF`

Erik Yang STAFF 8mth #469eec

Opcode: 110 1111

`s3` = `x19` = 10011

588 = 0b0 0000 0000 0010 0100 1100 but we don't write the last bit since it's implicitly 0 so bits [20:1] would be 0b0000 0000 0001 0010 0110

Bit 20: 0

Bit [10:1]: 0100100110

Bit 11: 0

Bit [19:12]: 00000000

Put it all together!

0 0100100110 0 00000000 10011 1101111

0010 0100 1100 0000 0000 1001 1110 1111 = 0x24C009EF

1

Anonymous Quelea 8mth #469efb

how did we know that `s3` = `x19` = 10011?

Erik Yang STAFF 8mth #469efc

Replying to Anonymous Quelea

from the ref card

Anonymous Quelea 8mth #469efd

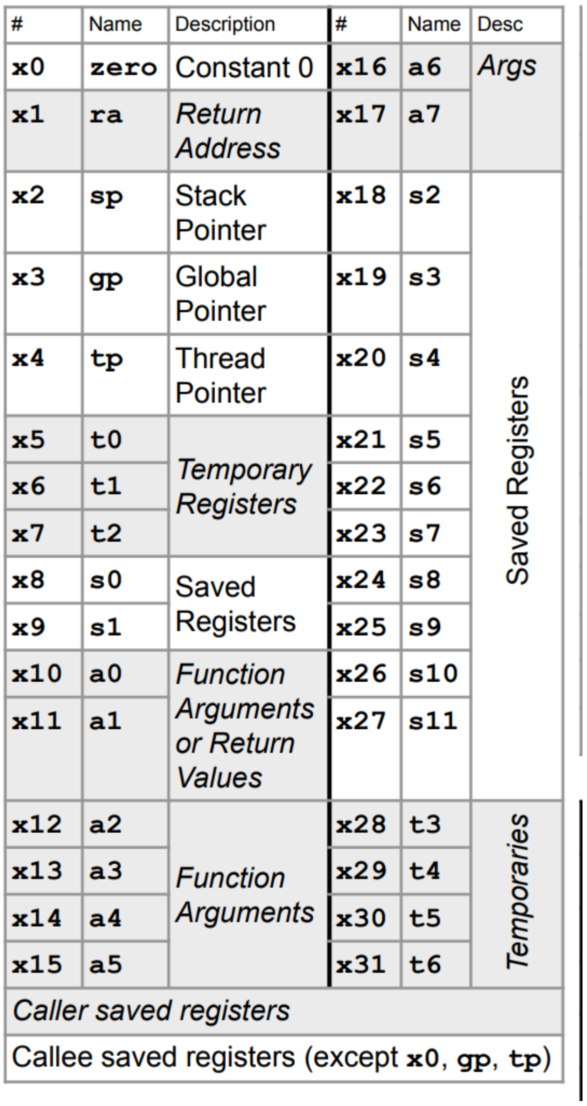
Replying to Erik Yang

do you mind pasting a screenshot of where you found it in the ref card?

Erik Yang STAFF 8mth #469faa

Replying to Anonymous Quelea

ofc, note that x19 = 10011 in binary

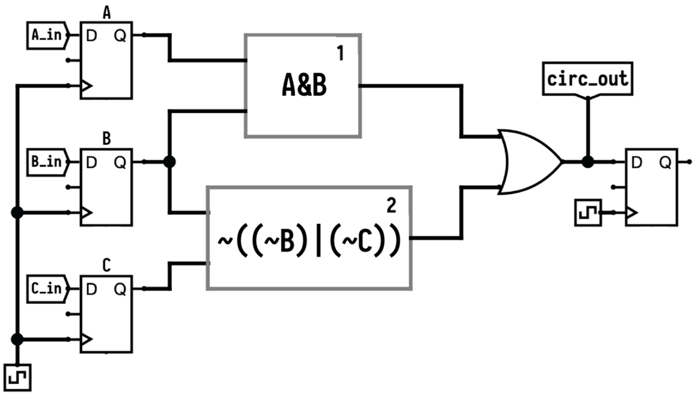


| # | Name | Description | # | Name | Desc |
|---|---|---|---|---|---|
| x0 | zero | Constant 0 | x16 | a6 | *Args* |
| x1 | ra | *Return Address* | x17 | a7 | |
| x2 | sp | Stack Pointer | x18 | s2 | Saved Registers |
| x3 | gp | Global Pointer | x19 | s3 | |
| x4 | tp | Thread Pointer | x20 | s4 | |
| x5 | t0 | *Temporary Registers* | x21 | s5 | |
| x6 | t1 | | x22 | s6 | |
| x7 | t2 | | x23 | s7 | |
| x8 | s0 | Saved Registers | x24 | s8 | |
| x9 | s1 | | x25 | s9 | |
| x10 | a0 | *Function Arguments or Return Values* | x26 | s10 | |
| x11 | a1 | | x27 | s11 | |
| x12 | a2 | *Function Arguments* | x28 | t3 | *Temporaries* |
| x13 | a3 | | x29 | t4 | |
| x14 | a4 | | x30 | t5 | |
| x15 | a5 | | x31 | t6 | |

*Caller saved registers*

Callee saved registers (except **x0**, **gp**, **tp**)

Anonymous Dotterel 8mth #469edc ✓ Resolved



$t_{\text{OR}} = 15\text{ns}$

$t_{\text{subcircuit 1}} = 13\text{ns}$

$t_{\text{subcircuit 2}} = 24\text{ns}$

$t_{\text{clk-q}} = 3\text{ns}$

$t_{\text{setup}} = 5\text{ns}$

Until this circuit settles at the correct input, it calculates two other Boolean expressions at the tunnel `circ_out`. Assume that register outputs are all 0 at $t = 0$, and that there are no setup or hold time violations.

Q5.3 (2 points) Let $t_1$ be the first time that `circ_out` changes. What is $t_1$ in nanoseconds? Clarified during exam: $t_1$ is the first time that `circ_out` *may* change.

**Solution:** 31ns. This is the time it takes a change at inputs `A_in` or `B_in` to reach `circ_out`. This involves passing through `clk-to-q` of register `A` or `B`, and then subcircuit `1`, then the `OR` gate= $3 + 13 + 15 = 31$ns.

**Grading:** Partial credit was awarded for identifying the correct combinational path that first reaches `circ_out`.

In these type of questions, do we never have to consider the fact that we would have to wait for subcircuit 2 to process before the OR gate because it needs two inputs. Is this because as long as one of the inputs changes, the OR output will change as subcirc2 will always be guaranteed to be outputting some value (maybe garbage)?

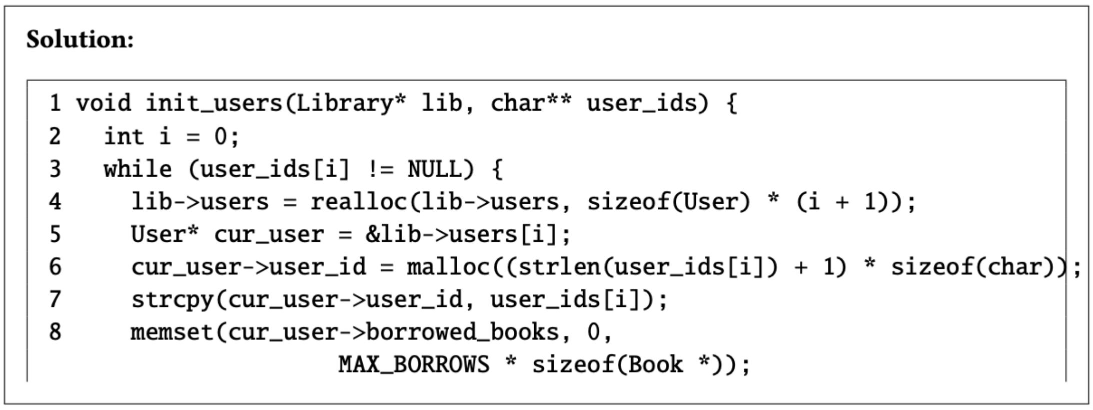
Did we assume there was a rising clock edge at 0 for this answer?

M Myrah Shah STAFF 8mth #469fab

#469d may help. Also, for your first question, as long as either input to an OR is 1, the output is 1.

Anonymous Jackal 8mth #469ecf ✓ Resolved

SU23-MT-Q2



**Solution:**

```
1 void init_users(Library* lib, char** user_ids) {
2   int i = 0;
3   while (user_ids[i] != NULL) {
4     lib->users = realloc(lib->users, sizeof(User) * (i + 1));
5     User* cur_user = &lib->users[i];
6     cur_user->user_id = malloc((strlen(user_ids[i]) + 1) * sizeof(char));
7     strcpy(cur_user->user_id, user_ids[i]);
8     memset(cur_user->borrowed_books, 0,
                          MAX_BORROWS * sizeof(Book *));
```

How do we know that the memset has to be set to sizeof(Book *), the book *pointer*, rather than the Book itself? Is memsetting to 0 the same thing as nulling it out?

Finally, in line 5, is &lib->users[i] basically saying I want to create a User object at the already-alloc'd place in memory?

E Erik Yang STAFF 8mth #469fea

borrowed_books is an array of Book pointers, not Book objects so that is why we are setting it to sizeof(Book*)

line 5 is trying to grab the current user (at index i) from the lib->users array

Anonymous Partridge 8mth #469aace

Is &lib->users[i] equivalent to &(lib->users[i])?

In other words, are we evaluating the lib->users[i] first and then getting the address of that user? I am confused because it looks like we are getting the address of lib first and then doing ->users[i].

If the former is correct, does this always apply when we are finding the address of something? Do we always evaluate and then take address (assuming no parentheses)?

E Erik Yang STAFF 8mth #469aade

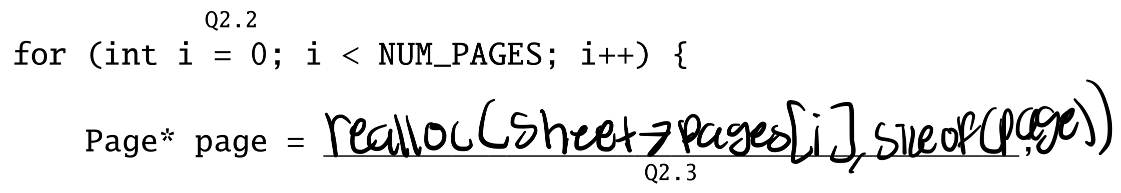
Replying to Anonymous Partridge

good question, it looks like they are equivalent, as the pointer has higher precedence than the address (&)

source: https://en.cppreference.com/w/c/language/operator_precedence

Anonymous Jellyfish 9mth #469ead ✓ Resolved

SP23-MT-Q2



```
                 Q2.2
for (int i = 0; i < NUM_PAGES; i++) {
    Page* page = realloc(sheet->pages[i], sizeof(page))
                                 Q2.3
```

Does this work as well even though we already allocated memory?

Andrew Liu STAFF 9mth #469ece

No, since `pages` is stored within the struct. It's memory is allocated by Q2.1, and you cannot reallocate it, since it is not a pointer returned by a memory allocating function, but rather a pointer within an allocated block.

Anonymous Goshawk 9mth #469eaa ✓ Resolved

SU23-MT-Q4.3 Why must we have addi rd rd 8? After lui t0 imm, I thought we could just do addi rd rd 4 and then add rd rd t0.

J Justin Yokota STAFF 9mth #469eab

Per the question, we need to get the PC value of the last instruction. The jal gets the PC value of the label, which is slightly off from the PC of the last instruction.

Anonymous Goshawk 9mth #469eac

Ohh ok. How come we don't do add rd rd 12 though? To go from temp_label to the fourth line, doesn't that require 4+4+4?

J Justin Yokota STAFF 9mth #469eae

Replying to Anonymous Goshawk

No; the temp_label doesn't use a line, since it's just a label pointing to the next line.

Anonymous Dotterel 9mth #469dfe ✓ Resolved

Q3.5 (3 points) List all numbers $k$ in this floating-point representation such that the smallest floating-point number strictly greater than $k$ is exactly $2k$. Express your answer as $x \times 2^y$, where $x$ is a decimal such that $1 \leq |x| < 2$ and $y$ is an integer. If there are no such numbers, write "None".

**Solution:** The wording of the question doesn't allow for negative answers - something greater than a negative number will always have smaller absolute value.
Additionally, $k$ cannot be a normal number, since we essentially need $k$ to be equal to the step size between representable numbers. Even among the denormal numbers, where the step size is $2^{-62} \times 2^{-8} = 2^{-70}$, $k = 2^{-70}$ is the only value that works.
**Grading:** Half credit was awarded for off-by-one errors ($2^{-69}$ or $2^{-71}$. 0.5 points were deducted for having additional answers.

can someone please explain why only 2^-8 works and not 2^-7 and 2^-6 as pairs because wouldn't i have 2^-6 * 2^-62 and 2^-7 *-62 which gives me 2^-68 and 2^-69 which differ by a factor of two?

♡ ...

Justin Yokota STAFF 9mth #469dff

If you write 2^-69 as a float, you'd find that the bits would be 0b000...00010. The next smallest number is 0b000...00011, which is 1.5*2^-69, which is not 2k. Similarly, 2^-68 won't work as well, because the next largest float is 1.25*2^-68

♡ 1 ...

Anonymous Dotterel 8mth #469eda

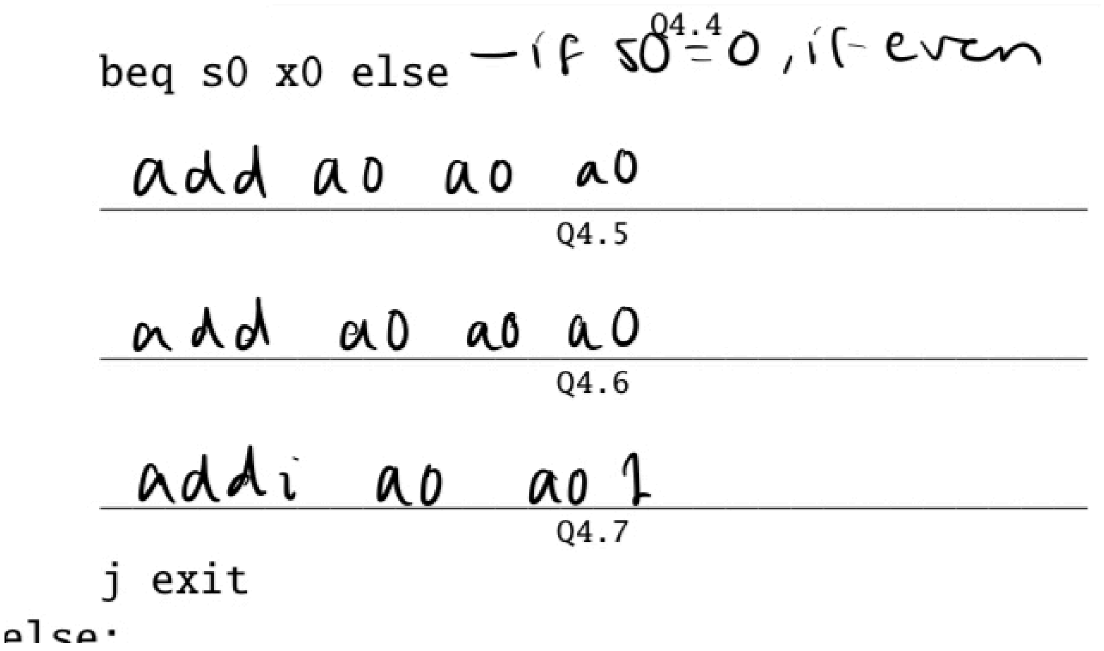
Thanks! any tips on how to get intuition for these type of questions?

♡ 1 ...

Anonymous Hippopotamus 9mth #469dfb ✓ Resolved

SU23-MT-Q4.1

Would this also be a valid solution? Since we are just adding a0 to itself twice more because 3*num is just the num plus 2x itself? Then we add 1 to the 'multiplied' number to get 3n+1.



```
    Q4.4
beq s0 x0 else —if s0 = 0, if even
  add a0 a0 a0
    Q4.5
  add a0 a0 a0
    Q4.6
  addi a0 a0 1
    Q4.7
j exit
else:
```

♡ ...

Anonymous Okapi 9mth #469dfc

I think this will result in 4 * a0 + 1since the second add instruction is adding two 2 * a0 resulted from the first add instruction.

♡ 1 ···

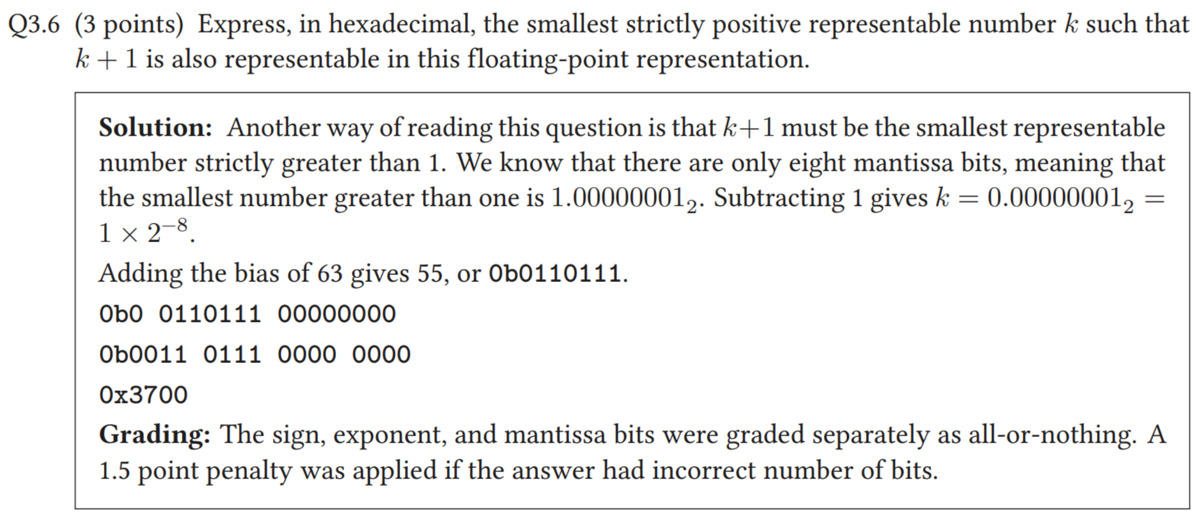
Anonymous Hippopotamus 9mth #469dfd

ohh i think you're right, thanks for pointing that out!

♡ ···

B Ben Yu 9mth #469dee ✓ Resolved

Fall 2023 3.6, why would one add the bias 63 while the problem states that the bias is -63?



Q3.6 (3 points) Express, in hexadecimal, the smallest strictly positive representable number $k$ such that $k + 1$ is also representable in this floating-point representation.

**Solution:** Another way of reading this question is that $k+1$ must be the smallest representable number strictly greater than 1. We know that there are only eight mantissa bits, meaning that the smallest number greater than one is $1.00000001_2$. Subtracting 1 gives $k = 0.00000001_2 = 1 \times 2^{-8}$.

Adding the bias of 63 gives 55, or `0b0110111`.

```
0b0 0110111 00000000
0b0011 0111 0000 0000
0x3700
```

**Grading:** The sign, exponent, and mantissa bits were graded separately as all-or-nothing. A 1.5 point penalty was applied if the answer had incorrect number of bits.

♡ ···

Anonymous Mouse 9mth #469dfa

I think they left out a step. We have our smallest number such that k+1 is also representable: $1 * 2^{-8}$ meaning our $exp + bias = -8$ Thus we "technically" subtract our bias but since our bias is negative we're adding it so $exp = -8 + 63$

♡ 1 ···

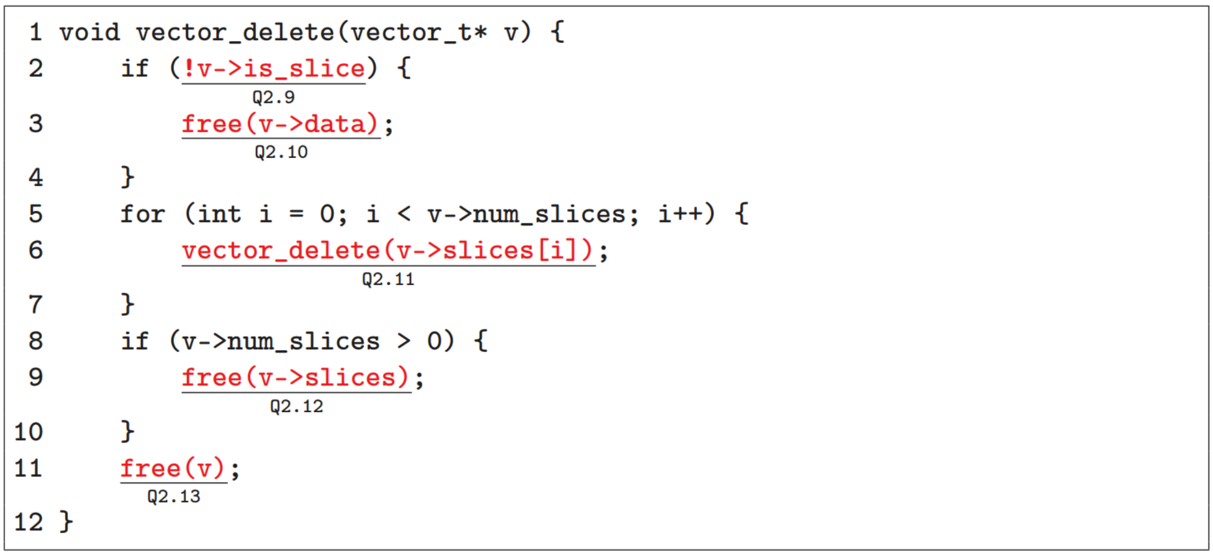
B Ben Yu 8mth #469abac

So the acutual exponent in the instruction code should be equivalent to 55. Thank you I got it!

♡ ···

B Ben Yu 9mth #469dea ✓ Resolved

2.2, why do we free (v) at the end? Even though all the malloc data such as v->data and v->slices are already freed out? My thinking is that the other elements of struct v is in the stack so they would be automatically freed after the function call.



```
void vector_delete(vector_t* v) {
    if (!v->is_slice) {
          Q2.9
        free(v->data);
           Q2.10
    }
    for (int i = 0; i < v->num_slices; i++) {
        vector_delete(v->slices[i]);
                      Q2.11
    }
    if (v->num_slices > 0) {
        free(v->slices);
            Q2.12
    }
    free(v);
     Q2.13
}
```

Ben Yu 9mth #469dec

This is Fall 2023 Midterm 2

(__m256i *) Jasmine Angle STAFF 9mth #469def

We are passed `v`, which itself is a pointer to a `vector_t` struct. This struct gets created earlier via the `calloc` call in `vector_slice`, which means that the struct that `v` points to is stored in the heap. Therefore, even though we have freed the data and slices, we must still free the `vector_t` struct itself to free everything that was initially stored on the heap.

1

Ben Yu 8mth #469abaa

Replying to (__m256i *) Jasmine Angle

Understood. Thank you!

Anonymous Shrew 9mth #469dde ✓ Resolved

Hey, I am confused with this:

Here, I don't quite understand how those read ones are correct, its quite tedious to follow how it works.

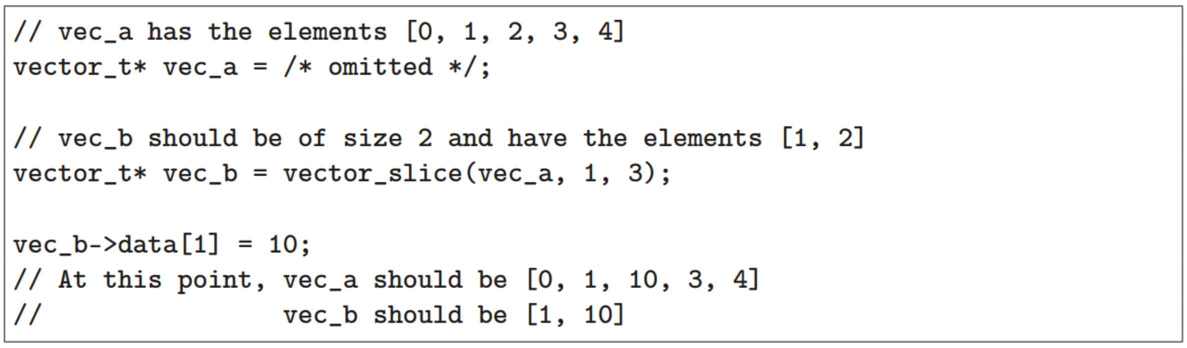
**FA23-MT-Q2**

To do so, we've made some updates to the **`vector_t`** type from lab. You may assume that all necessary standard libraries are included.

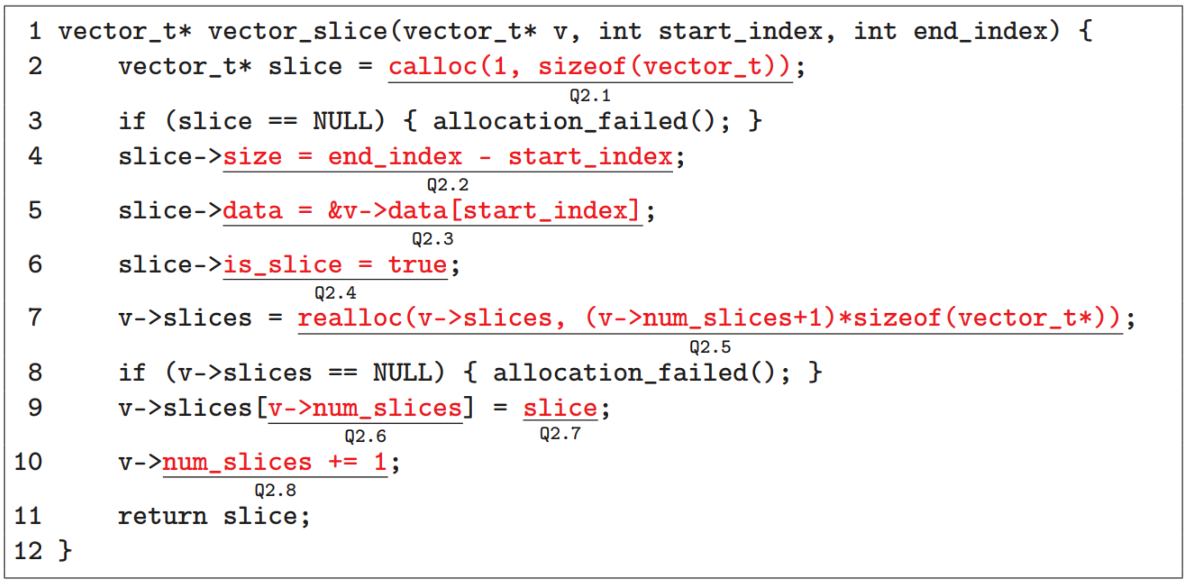
```
typedef struct vector_t {
    // Number of elements in the vector; you may assume size > 0
    size_t size;

    // Pointer to the start of the vector
    int* data;

    // Number of child slices
    size_t num_slices;

    // Array of the vector's child slices, or NULL if num_slices == 0
    struct vector_t** slices;

    // true if the vector is a child slice of another vector, otherwise false
    bool is_slice;
} vector_t;
```

Useful C function prototypes:

```
void* malloc(size_t size);
void free(void *ptr);
void* calloc(size_t num_elements, size_t size);
void* realloc(void *ptr, size_t size);
```



```
// vec_a has the elements [0, 1, 2, 3, 4]
vector_t* vec_a = /* omitted */;

// vec_b should be of size 2 and have the elements [1, 2]
vector_t* vec_b = vector_slice(vec_a, 1, 3);

vec_b->data[1] = 10;
// At this point, vec_a should be [0, 1, 10, 3, 4]
//                vec_b should be [1, 10]
```



```
 1 vector_t* vector_slice(vector_t* v, int start_index, int end_index) {
 2     vector_t* slice = calloc(1, sizeof(vector_t));
                                  Q2.1
 3     if (slice == NULL) { allocation_failed(); }
 4     slice->size = end_index - start_index;
                         Q2.2
 5     slice->data = &v->data[start_index];
                        Q2.3
 6     slice->is_slice = true;
                 Q2.4
 7     v->slices = realloc(v->slices, (v->num_slices+1)*sizeof(vector_t*));
                                         Q2.5
 8     if (v->slices == NULL) { allocation_failed(); }
 9     v->slices[v->num_slices] = slice;
                     Q2.6         Q2.7
10     v->num_slices += 1;
              Q2.8
11     return slice;
12 }
```

Andrew Liu STAFF 9mth #469ecb

The algorithm is to:

1. Create a new slice and set all its fields to 0 (Q 2.1)
    1. If allocation failed, indicate so (line 3)
2. Set the size (Q 2.2)
3. Link the data of the new slice to the start of its appropriate data (Q 2.3)
4. Set the new vector to be a slice (Q2.4)
5. Increase the number of slices of the parent by 1 to add in our new slice, and check the allocation (Q2.5, line 8)
6. Set the new vector as a slice within the parent vector (Q2.6)
7. Increment the number of slices that reference the parent (Q2.7)
8. Return

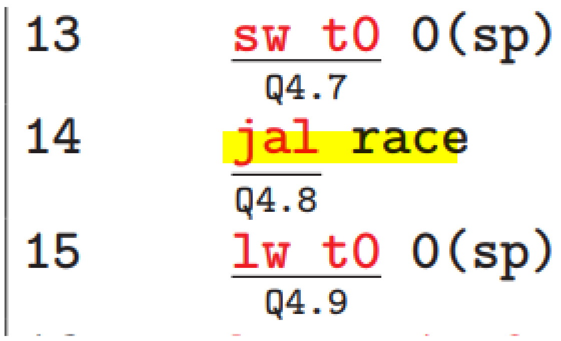
Anonymous Shrew 8mth #469edb

I understand that but I dont get the reason why use calloc in such situations?

1

damien toh 9mth #469ddd ✓ Resolved

SP23-MT-Q4.8



```
sw t0 0(sp)
Q4.7
jal race
Q4.8
lw t0 0(sp)
Q4.9
```

hello why is 4.8 "jal race" and not "jal ra race"

1 ···

(__m256i *) Jasmine Angle STAFF 9mth #469ddf

`jal label` is a pseudoinstruction for `jal ra label`, so `jal race` here is translated to `jal ra race`.

···

Anonymous Goshawk 9mth #469ddc ✓ Resolved

FA23-MT-Q4 Why do we do jal race and not jal ra race?

···

Myrah Shah STAFF 9mth #469ded

#469ddd

···

Anonymous Elephant 9mth #469ddb ✓ Resolved

FA23-Q6

for each of the functions, since they are passed in as a pointer don't we need to dereference before calling? so for 6.3, would it be

!(*andy(1<<i))

···

Justin Yokota STAFF 9mth #469eaf

No; C functions act correctly both when dereferenced, and not dereferenced.

···

Anonymous Elephant 9mth #469eba

So the function can be used even though it is a pointer to a function passed in? Such as this example from the review session would work without * using just int output = op(x, y)?

```
void print_function_output(int x, int y, int(*op)(int, int))
{
        int output = (*op)(x, y);
        printf("output of function is %d!!\n", output);
}

int main() {
        print_function_output(3, 4, add_nums); // 7
        print_function_output(3, 4, multiply_nums); // 12

        return 0;
}
```

···

Andrew Liu STAFF 9mth #469ecc

Replying to Anonymous Elephant

Yep!

Anonymous Elephant 9mth #469dce ✓ Resolved

FA23-Q2-part 2

what is the difference between free(v) and free(v-> data) since v is the pointer to the vector and data is defined as the pointer to the start of the vector? and is the reason why we need to call vector_delete in the for loop because each slice could have descendant slices of their own?

Andrew Liu STAFF 9mth #469ecd

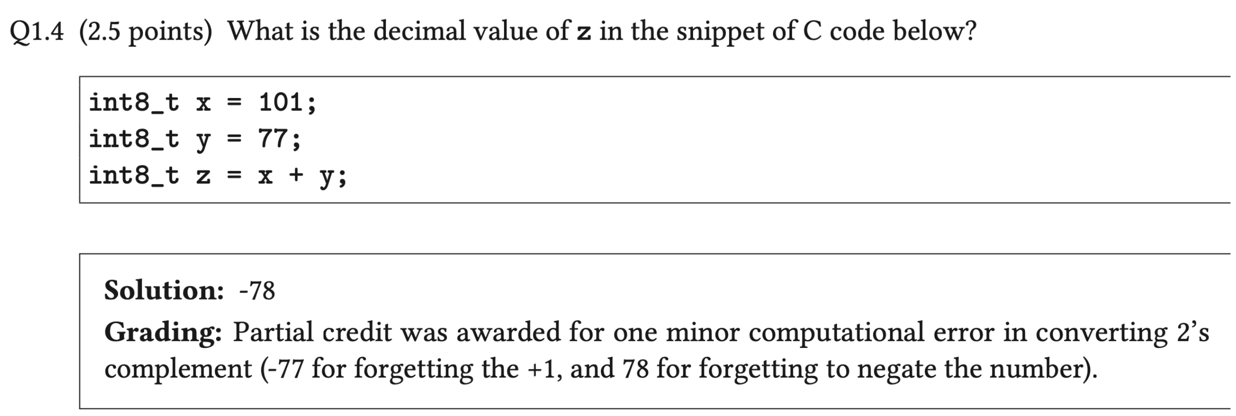
Freeing `v` is freeing the `vector` struct. Freeing `v->data` is freeing the data that the `vector` struct represents.

You need the recursion since the slices can form a tree (slices can have slices made from them)

Anonymous Grasshopper 9mth #469dcd ✓ Resolved

FA23-MT-Q1

Can someone explain why we use 2's Compliment here? Thank you so much!



Q1.4 (2.5 points) What is the decimal value of **z** in the snippet of C code below?

```
int8_t x = 101;
int8_t y = 77;
int8_t z = x + y;
```

**Solution:** -78

**Grading:** Partial credit was awarded for one minor computational error in converting 2's complement (-77 for forgetting the +1, and 78 for forgetting to negate the number).

1

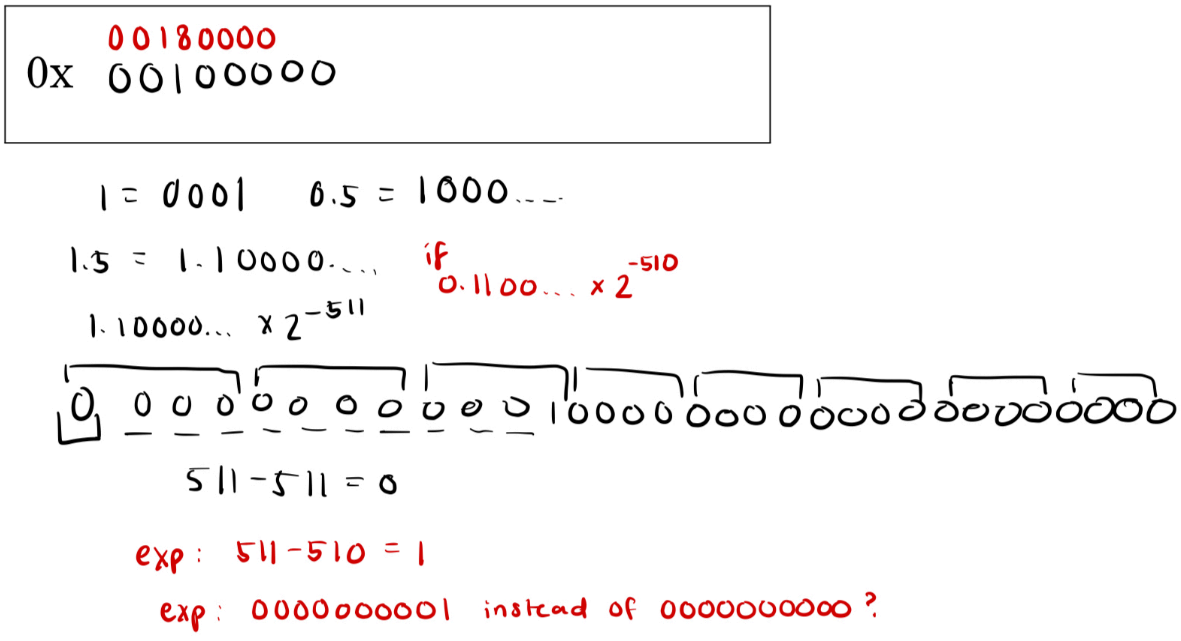
Anonymous Elephant 9mth #469dda

https://edstem.org/us/courses/51705/discussion/4462328?comment=10481382

Anonymous Hippopotamus 9mth #469dbf ✓ Resolved

SU23-MT-Q1.11

After reading the solutions, I still don't really understand why we took the scientific number conversion to be 0.110..E-510 instead of 1.1000..E-511. If we did 0.11E-510 wouldn't the exp value be 511-510=1 instead of all 0s the way we want it?



0x 00100000

00180000

1 = 0001    0.5 = 1000...

1.5 = 1.10000...

if 0.1100... x $2^{-510}$

1.10000... x $2^{-511}$

0 000000000000 | 10000000000000000000000000

511 − 511 = 0

exp : 511 − 510 = 1

exp : 0000000001 instead of 0000000000 ?

♡ ···

Justin Yokota STAFF 9mth #469dcb

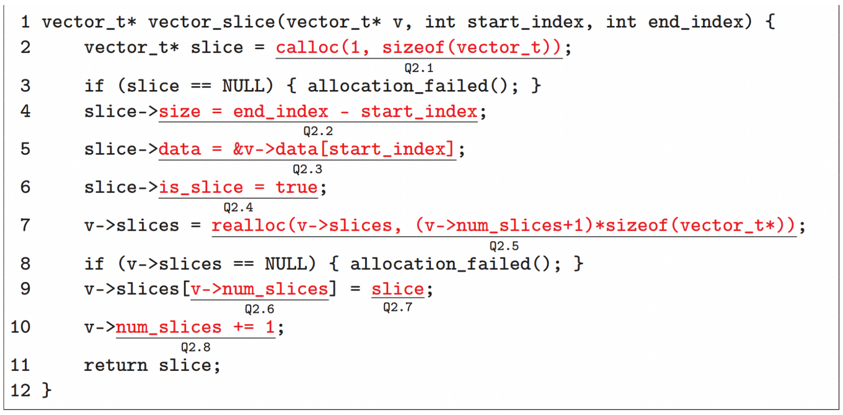
Denormalized numbers use an exponent one greater than normal.

♡ ···

Anonymous Monkey 9mth #469dbc ✓ Resolved

FA23-Midterm-Q2

Why do we not have to typecast the calloc call in line 2? Wouldn't the correct way to calloc space be vector_t* slice = (vector_t*) calloc(1, sizeof(vectore_t)) similar to the way we call it for int*?



```
 1 vector_t* vector_slice(vector_t* v, int start_index, int end_index) {
 2     vector_t* slice = calloc(1, sizeof(vector_t));
                                  Q2.1
 3     if (slice == NULL) { allocation_failed(); }
 4     slice->size = end_index - start_index;
                         Q2.2
 5     slice->data = &v->data[start_index];
                        Q2.3
 6     slice->is_slice = true;
                  Q2.4
 7     v->slices = realloc(v->slices, (v->num_slices+1)*sizeof(vector_t*));
                                       Q2.5
 8     if (v->slices == NULL) { allocation_failed(); }
 9     v->slices[v->num_slices] = slice;
                      Q2.6           Q2.7
10     v->num_slices += 1;
              Q2.8
11     return slice;
12 }
```

♡ ···

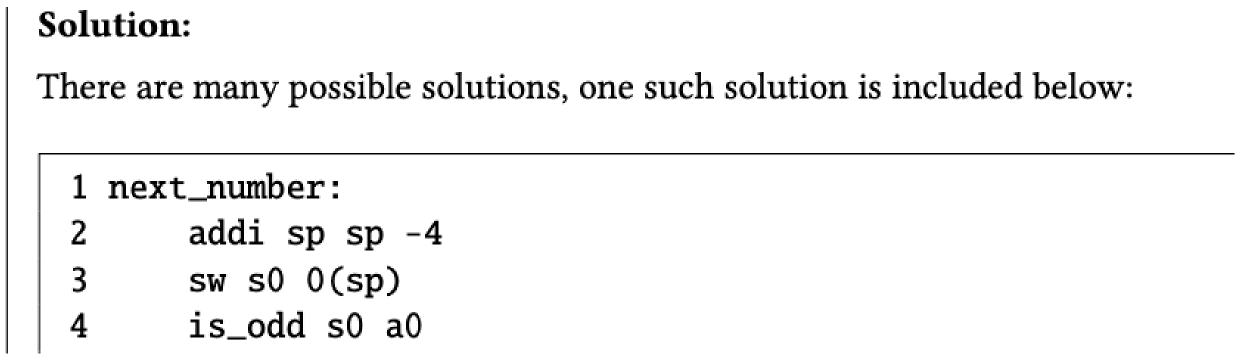
Myrah Shah STAFF 9mth #469dbe

In relation to casting the result of `malloc` / `calloc` : #114db

♡ 1 ···

Anonymous Jellyfish 9mth #469dbb ✓ Resolved

SU23-mt1-q4



**Solution:**

There are many possible solutions, one such solution is included below:

```
next_number:
    addi sp sp -4
    sw s0 0(sp)
    is_odd s0 a0
```

Why didn't we store ra in stack?

1

Andrew Liu STAFF 9mth #469ebe

We don't need to, since we do not call any other function from `next_number`.

Anonymous Goshawk 9mth #469dac ✓ Resolved

FA23-MT-Q6.4 Why is this the most optimal approach? Couldn't we use 1<< like in 6.3? Using XOR, if the bits are the same (0, so luxor returns true) then we know that bit of the favorite number must be 1. If the bits are different (1, so luxor returns false), then we know that bit of the favorite number must be 0. Then just add this to a uint32_t result.

Wouldn't this result in O(N)?

Justin Yokota STAFF 9mth #469dae

Is that true? If I do 7 ^ 1, the result is 6, which is not 0, so Luxor will return false.

Anonymous Squid 9mth #469daf

Isn't that we want? We can flip the bits in the number 1 that are value 1 in the output 6 to get 7 as the result.

Justin Yokota STAFF 9mth #469dba

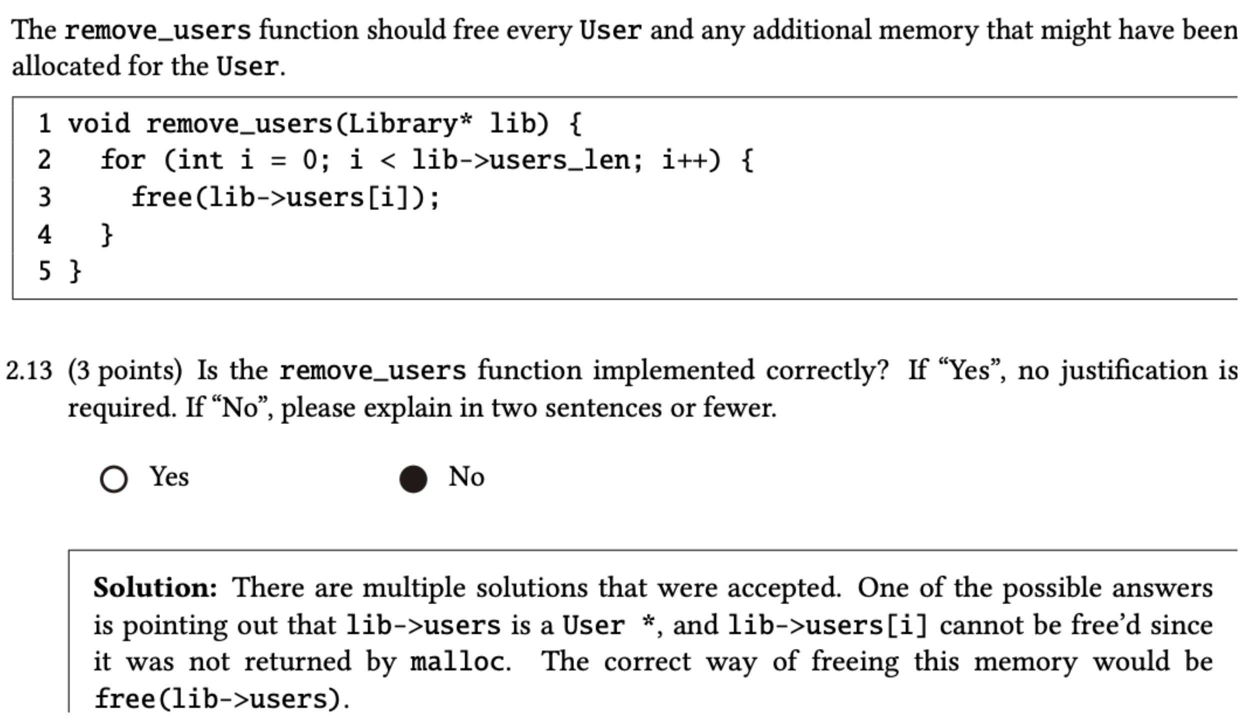
Replying to Anonymous Squid

Depends on what your plan is, but either way, it shouldn't work properly. 6^1 will equal 7, which is not 0, so Luxor will return false as well. Either way, Luxor returning false doesn't distinguish between the last bit being 0 or 1.

The only time Luxor returns true is if you successfully guess the right number.

Anonymous Jellyfish 9mth #469dab ✓ Resolved

SU23-mt-Q2.13



The `remove_users` function should free every `User` and any additional memory that might have been allocated for the `User`.

```
void remove_users(Library* lib) {
  for (int i = 0; i < lib->users_len; i++) {
    free(lib->users[i]);
  }
}
```

2.13 (3 points) Is the `remove_users` function implemented correctly? If "Yes", no justification is required. If "No", please explain in two sentences or fewer.

Yes [✓ No]

**Solution:** There are multiple solutions that were accepted. One of the possible answers is pointing out that `lib->users` is a `User *`, and `lib->users[i]` cannot be free'd since it was not returned by `malloc`. The correct way of freeing this memory would be `free(lib->users)`.

Is it also valid to say that we malloced each user's user_id and we didn't free that therefore we didn't free everything in Users?

Anonymous Gorilla 9mth #469dcf

Same question

Myrah Shah STAFF 8mth #469eef

#469ebd

Anonymous Stork 9mth #469daa ✓ Resolved

FA23 MT Q2.1
For the first line, instead of assigning slice of calloc(1, sizeof(vector_t)), can we malloc(1*sizeof(vector_t)) ??

Also, in the line where we realloc, why do we multiply by sizeof(vector_t*) instead of sizeof(vector_t) ?

1

Laurence Yang 9mth #469dbd

I have the same question.

Also, is it correct if I write `data = v->data + start_index` for line 5?

Anonymous Elephant 9mth #469dca

I also wrote this initially but I think that this line wouldn't guarantee we access the right element. For example, if start_index = 1 and we're working with integers stored in data, then we would want to increment the address by 4 instead of by 1 itself so using the brackets guarantees the pointer arithmetic is done correctly. Not sure if I'm right though.

Andrew Liu STAFF 9mth #469ebf

No, since you need to explicitly set some fields to `0 == NULL`.

We multiply by `sizeof(vector_t*)` since `slices` is an array of `vector_t*`

Jero Wang STAFF 9mth #469eca Visible to staff only

sad i got sniped, i'm moving

Anonymous Dinosaur 8mth #469aabf

sorry can you elaborate on the calloc part and how to correlates to setting fields?

Anonymous Wolf 9mth #469cda ✓ Resolved

SP-23-Q2.10
Why is num-pages in code and not data (since it the variable is a global variable) how does #define affect where it is stored in memory?

Myrah Shah STAFF 9mth #469cfb

This might help: #469cdd. Feel free to ask any followups!

Anonymous Turtle 9mth #469ccc ✓ Resolved

Su23 mt 3.6

Why is infinity not an option?

Justin Yokota STAFF 9mth #469ccd

How would you get infinity?

Anonymous Turtle 9mth #469ccb ✓ Resolved

Su23 MT 2.13

Would we get the points if we said the contents in each user was not freed?

Jero Wang STAFF 9mth #469ebd

No, freeing a `User` struct would free all memory associated with it - it frees the `borrowed_books` array, but should not free the `Book`s that are held in the array since they may be reused.

Anonymous Turtle 8mth #469aacf

When do we need to free multiple layers from “inside out” like in lab one?

Justin Yokota STAFF 8mth #469acfe

Replying to Anonymous Turtle

"free" only frees the pointer itself, and doesn't "recursively" free child pointers. You thus need to free multiple layers if you are trying to free all the data recursively down pointers.

Anonymous Turtle 8mth #469adba

Replying to Justin Yokota

Then why don't we need to do recursive "freeing" for this problem?

Justin Yokota STAFF 8mth #469adbc

Replying to Anonymous Turtle

The `borrowed_books` array isn't a pointer, but an array. As such, it isn't itself a child pointer.

Anonymous Turtle 9mth #469cca ✓ Resolved

Su23 mt

2.1 is i<strlen(user_ids) valid?

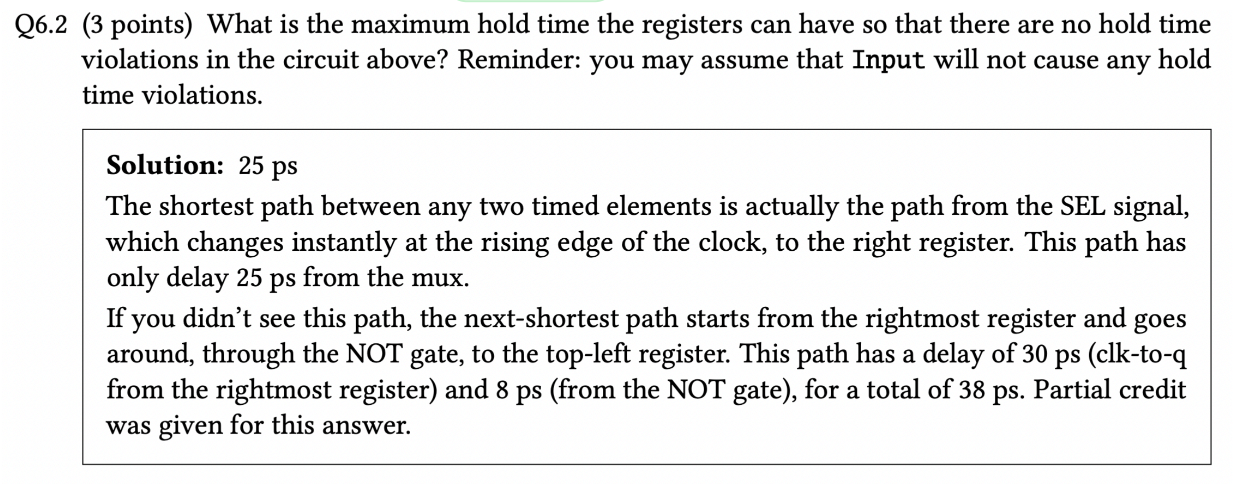
2 in general: how come we don't need to assign users_len and books to anything

Myrah Shah STAFF 9mth #469cfe

1. #469baf: Calling `strlen` on a `char**` is undefined behavior.

2. `users_len` is assigned on line 11 of `init_users` . `borrowed_books` is assigned on line 8 of the solution.

Anonymous Elephant 9mth #469cbe ✓ Resolved



Q6.2 (3 points) What is the maximum hold time the registers can have so that there are no hold time violations in the circuit above? Reminder: you may assume that Input will not cause any hold time violations.

**Solution:** 25 ps

The shortest path between any two timed elements is actually the path from the SEL signal, which changes instantly at the rising edge of the clock, to the right register. This path has only delay 25 ps from the mux.

If you didn't see this path, the next-shortest path starts from the rightmost register and goes around, through the NOT gate, to the top-left register. This path has a delay of 30 ps (clk-to-q from the rightmost register) and 8 ps (from the NOT gate), for a total of 38 ps. Partial credit was given for this answer.

SP23-MT-Q6.2

why is the SEL signal a timed element if there is no clock directly attached to it and has no clk-to-q delay? and it seems inconsistent with Q6.1 where we calculated the longest path to go from register, past multiplier, past mux, and t_setup so if the mux was clocked, we would end this past after the multiplier? in the walkthrough video, the answer that was calculated was also 38, not 25

Justin Yokota STAFF 9mth #469cce

The question directly states that the SEL signal changes instantly at the rising edge.

Anonymous Elephant 9mth #469cfa

Would that mean that our critical path should end before the SEL signal and not go past it to the top register?

Justin Yokota STAFF 9mth #469cfc

Replying to Anonymous Elephant

There's not really anything "before" the SEL signal? It's more that the paths starting from SEL are now also candidates for the path that most constrains the circuit.

Anonymous Goshawk 9mth #469caf ✓ Resolved

FA23-MT-Q5.3 Why isn't setup time added and only clock-to-q?

Anonymous Sardine 9mth #469cba

hold_time <= clk-to-q-delay + shortest-combinational-delay

Anonymous Goshawk 9mth #469cbc

ohh I see but how did you know that the question asks us to solve max hold time?

Anonymous Sardine 9mth #469cbd

Replying to Anonymous Goshawk

Wait, I'm sorry. There is a similar question in SU23, that I was confused on and the video and question showed the hold time relationship. I am not why the set-up time isn't added.

Jedidiah Tsang STAFF 9mth #469cbf

Replying to Anonymous Sardine

circ_out is not a state element (it's a tunnel), so the value in circ_out changes as soon as the wire reaches that tunnel. Setup time refers to the amount of time an input to a state element must be stable before the rising edge of the clock, so that has no bearing here since it's not a state element.

Anonymous Goshawk 9mth #469dad

Replying to Jedidiah Tsang

Hm I see. So if the question instead defined t1 as the time from register A/B to register D, then we would add setup? Register D will use setup time to ensure the output of circ_out is stable?

Also, does this question have anything to do with hold time?

Jero Wang STAFF 9mth #469ebc

Replying to Anonymous Goshawk

Yes, if we're looking at when the value **in** register D changes, we'd need to add setup time. 5.3 is not affected by hold time (also because we say assume no hold time violations).

Anonymous Sardine 9mth #469cad ✓ Resolved

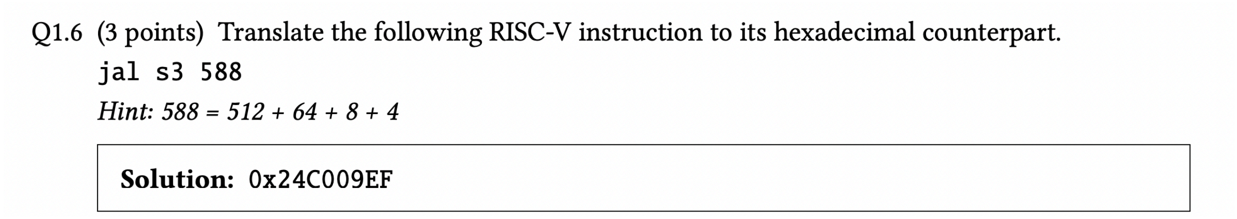
Is FA23 walkthrough posted, I can't find it?

Jedidiah Tsang STAFF 9mth #469cfd

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

1

Anonymous Elephant 9mth #469bff ✓ Resolved



Q1.6 (3 points) Translate the following RISC-V instruction to its hexadecimal counterpart.

```
jal s3 588
```

*Hint: 588 = 512 + 64 + 8 + 4*

**Solution:** 0x24C009EF

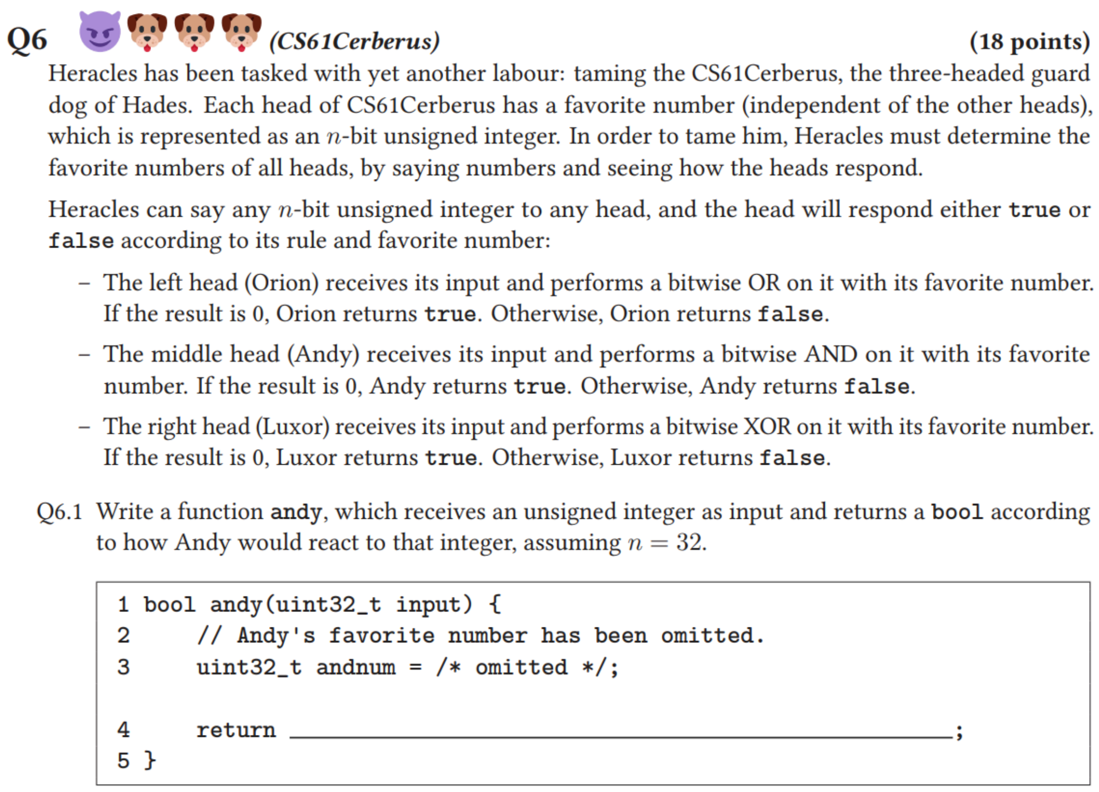
Sp23-MT-Q1.6 should the first hex number be 0 instead of 2 since bits 20, 10, 9, 8 are all zeros due to sign extension?

Justin Yokota STAFF 9mth #469cab

Are you sure that bit 9 is 0?

Anonymous Sand Dollar 9mth #469bef ✓ Resolved

Fa23 - Q6: is this question in scope for us? I know it is asking to calculate runtime which I don't think we covered in class:



**Q6** 😈🐶🐶🐶 *(CS61Cerberus)* **(18 points)**

Heracles has been tasked with yet another labour: taming the CS61Cerberus, the three-headed guard dog of Hades. Each head of CS61Cerberus has a favorite number (independent of the other heads), which is represented as an $n$-bit unsigned integer. In order to tame him, Heracles must determine the favorite numbers of all heads, by saying numbers and seeing how the heads respond.

Heracles can say any $n$-bit unsigned integer to any head, and the head will respond either `true` or `false` according to its rule and favorite number:

- The left head (Orion) receives its input and performs a bitwise OR on it with its favorite number. If the result is 0, Orion returns `true`. Otherwise, Orion returns `false`.
- The middle head (Andy) receives its input and performs a bitwise AND on it with its favorite number. If the result is 0, Andy returns `true`. Otherwise, Andy returns `false`.
- The right head (Luxor) receives its input and performs a bitwise XOR on it with its favorite number. If the result is 0, Luxor returns `true`. Otherwise, Luxor returns `false`.

Q6.1 Write a function `andy`, which receives an unsigned integer as input and returns a `bool` according to how Andy would react to that integer, assuming $n = 32$.

```
bool andy(uint32_t input) {
    // Andy's favorite number has been omitted.
    uint32_t andnum = /* omitted */;

    return ______________________________________________;
}
```

Justin Yokota STAFF 9mth #469bfa

Yes. Runtime calculations are considered fair game, as they were covered in the prerequisite class 61B.

Anonymous Hippopotamus 9mth #469bee ✓ Resolved

SP23-MT-Q2.3 & 2.11

2.3: Why do we need to get the address of sheet using `&sheet` instead of just calling `sheet`? I don't really understand how this is any different than calling eg. sheet->student_id or sheet->total_length

2.11: Why is sheet stored on the stack? I don't really understand how sheet and *sheet is different in this case.

Justin Yokota STAFF 9mth #469ccf

2.3: C operator precedence puts the address operator after -> and []. The address operator is actually applying to `sheet->pages[i]` , which is a Page, and thus yields a Page*.

2.11: The variable sheet itself is on the stack, but is pointing to something on the heap.

Anonymous Wolf 9mth #469bed ✓ Resolved

FA23-MT-Q2

For line 5 why is it &v->data[start_index] and not v->data[start_index]. How does the & affect the output.

J Jim Fang 9mth #469caa

& is needed here because v->data[start_index] would return an int. v->data gets a pointer to the 0th element, [start_index] does *(v->data+start_index). To then get the pointer to the value at v->data[start_index], you need to add & to the front of it, which returns the mem address of v->data[start_index]

♡ 2 ...

Anonymous Panther 9mth #469bec ✓ Resolved

Sp23-mt-q1.7,

At one point when we are at ( ~Y | Y * Z) we use De morgan's law by negating this whole statement. How do you know when to do this in other cases?

♡ ...

J Justin Yokota STAFF 9mth #469bfc

You don't really "know" to do this? It's just one of many different ways you can solve this problem, much as how there are often many ways to solve any math problem. In this particular case, this simplification ended up working, but it doesn't always work.

♡ ...

Anonymous Panther 9mth #469beb ✓ Resolved

Sp23-midterm-q1.11

If the string str didn't have a total of 8 bytes, and we wanted to convert the string str into int32_t, what would have happened? Also, could someone explain why we had to reverse the strings just inside each index instead of the whole string itself based on little endian?

♡ ...

J Justin Yokota STAFF 9mth #469cdb

It depends on the length of str; generally, every 4 chars would get read as an integer, and the conversion of 4 chars to an integer depends on the endianness of the system.

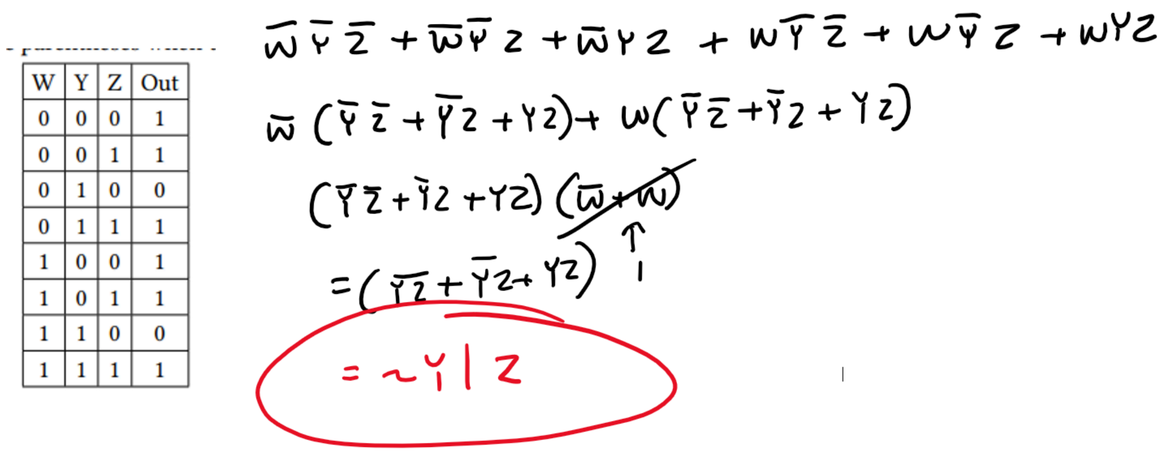
Arrays are always stored with the 0th element at the lowest address. Endianness affects how groups of bytes combine to form single values. In this case, an array of chars gets written, and we then group together the chars in 4-byte blocks, then interpret each 4-byte block individually as ints.

♡ ...

A Adrian Bao 9mth #469bdd ✓ Resolved

SP23-Q1.7

I was a bit confused on how we can go from the last step to the final answer (red). Would the last step in black be an accepted answer?



| W | Y | Z | Out |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

$\bar W \bar Y \bar Z + \bar W \bar Y Z + \bar W Y Z + W \bar Y \bar Z + W \bar Y Z + WYZ$

$\bar W(\bar Y \bar Z + \bar Y Z + YZ) + W(\bar Y \bar Z + \bar Y Z + YZ)$

$(\bar Y \bar Z + \bar Y Z + YZ)\cancel{(\bar W + W)}$ ↑ 1

$= (\bar Y \bar Z + \bar Y Z + YZ)$

$= \sim Y | Z$

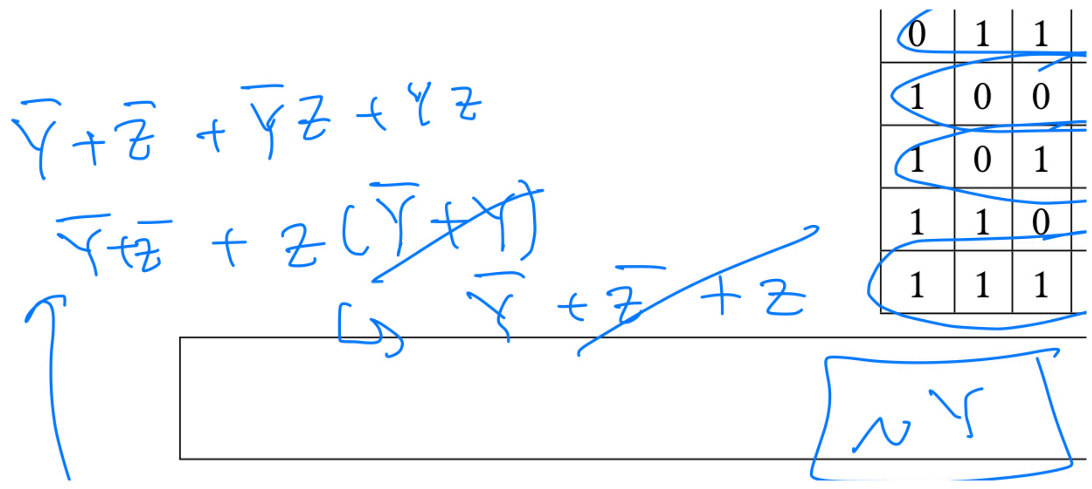
Myrah Shah STAFF 9mth #469bdf

Take a look at Q5.1b on Discussion 5, which has a similar format as the last part here. One way to simplify an expression like this is to use the strategy of adding an extra term (here we can add an extra ~Y&Z) to factor out like terms.

1

Anonymous Echidna 9mth #469bea

Isn't it valid to simplify via this? Or is this not correct



$\bar Y + \bar Z + \bar Y Z + YZ$

$\bar Y + \bar Z + Z\cancel{(\bar Y + Y)}$

$\hookrightarrow \bar Y + \cancel{\bar Z + Z}$

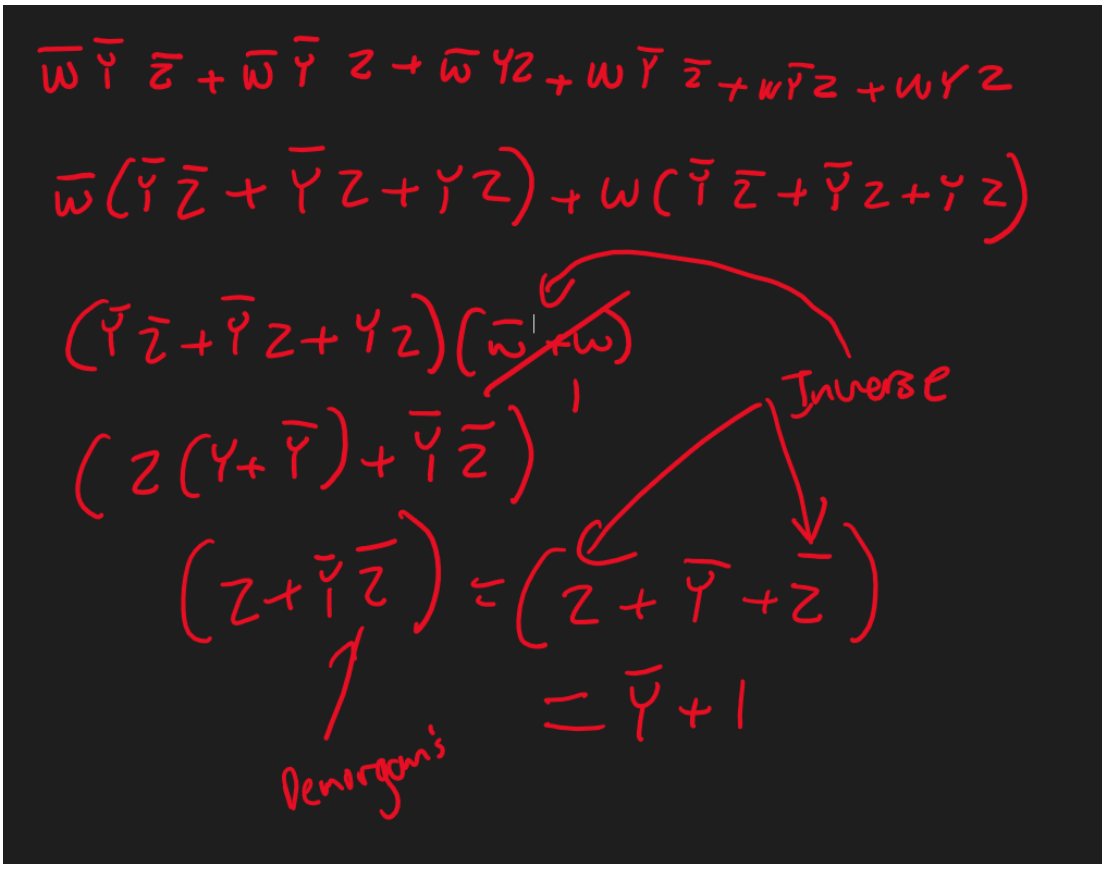
$\boxed{\sim Y}$

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

Adrian Bao 9mth #469bfe

Replying to Anonymous Echidna

Hi Echidna, I believe there may be an error with your calculation. $\not z + z$ is 1, not 0. Is my solution correct?



$\overline{W}\,\overline{Y}\,\overline{Z}+\overline{W}\,\overline{Y}Z+\overline{W}YZ+W\overline{Y}\,\overline{Z}+W\overline{Y}Z+WYZ$

$\overline{W}(\overline{Y}\,\overline{Z}+\overline{Y}Z+YZ)+W(\overline{Y}\,\overline{Z}+\overline{Y}Z+YZ)$

$(\overline{Y}\,\overline{Z}+\overline{Y}Z+YZ)(\cancel{\overline{W}+W})$ $1$

$(Z(Y+\overline{Y})+\overline{Y}\,\overline{Z})$

$(Z+\overline{Y}\,\overline{Z}) = (Z+\overline{Y}+\overline{Z})$

$=\overline{Y}+1$

Andrew Liu STAFF 9mth #469cbb

Replying to Adrian Bao

$\overline{Y}\cdot\overline{Z}$ is not equivalent to $\overline{Y}+\overline{Z}$ — I think you may have gotten confused by $\overline{Y}\cdot\overline{Z}$ looking like $\overline{YZ}$.

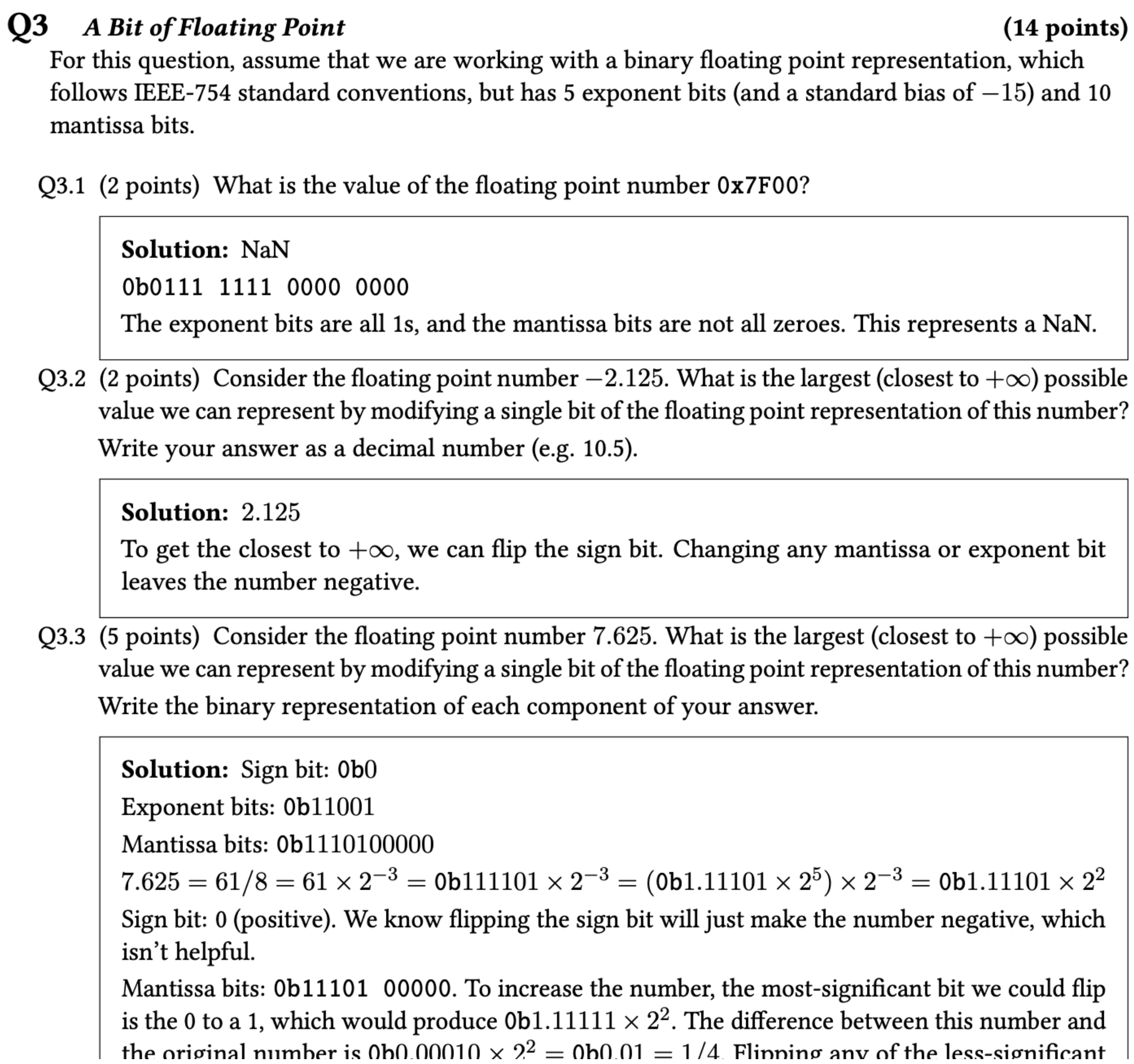
This is one solution below:

$$\begin{aligned}
Z+\overline{Y}\overline{Z} &= (Y+\overline{Y})Z+\overline{Y}\overline{Z} && \text{(Identity)}\\
&= YZ+\overline{Y}Z+\overline{Y}\overline{Z} && \text{(Distribution)}\\
&= YZ+\overline{Y}Z+\overline{Y}Z+\overline{Y}\overline{Z} && \text{(Idempotence)}\\
&= (\cancel{Y+\overline{Y}})Z+\overline{Y}(\cancel{Z+\overline{Z}}) && \text{(Distribution)}\\
&= (1)Z+\overline{Y}(1) && \text{(Cancellation)}\\
&= Z+\overline{Y} && \text{(Identity)}
\end{aligned}$$

Anonymous Echidna 9mth #469bdc ✓ Resolved

SP23-MT-Q3

Can we assume that all FP problems are normalized? I assumed it as such and got the answers right in this question (as it wasn't stated that it was normalized but I just assumed), but not sure if this is a fair assumption:



**Q3** ***A Bit of Floating Point*** **(14 points)**

For this question, assume that we are working with a binary floating point representation, which follows IEEE-754 standard conventions, but has 5 exponent bits (and a standard bias of $-15$) and 10 mantissa bits.

Q3.1 (2 points) What is the value of the floating point number `0x7F00`?

**Solution:** NaN

`0b0111 1111 0000 0000`

The exponent bits are all 1s, and the mantissa bits are not all zeroes. This represents a NaN.

Q3.2 (2 points) Consider the floating point number $-2.125$. What is the largest (closest to $+\infty$) possible value we can represent by modifying a single bit of the floating point representation of this number? Write your answer as a decimal number (e.g. 10.5).

**Solution:** $2.125$

To get the closest to $+\infty$, we can flip the sign bit. Changing any mantissa or exponent bit leaves the number negative.

Q3.3 (5 points) Consider the floating point number $7.625$. What is the largest (closest to $+\infty$) possible value we can represent by modifying a single bit of the floating point representation of this number? Write the binary representation of each component of your answer.

**Solution:** Sign bit: `0b0`

Exponent bits: `0b11001`

Mantissa bits: `0b1110100000`

$7.625 = 61/8 = 61 \times 2^{-3} = \texttt{0b}111101 \times 2^{-3} = (\texttt{0b}1.11101 \times 2^5) \times 2^{-3} = \texttt{0b}1.11101 \times 2^2$

Sign bit: 0 (positive). We know flipping the sign bit will just make the number negative, which isn't helpful.

Mantissa bits: `0b11101 00000`. To increase the number, the most-significant bit we could flip is the 0 to a 1, which would produce $\texttt{0b}1.11111 \times 2^2$. The difference between this number and the original number is $\texttt{0b}0.00010 \times 2^2 = \texttt{0b}0.01 = 1/4$. Flipping any of the less-significant

Justin Yokota STAFF 9mth #469bfd

This question specifies that the floating point number follows all IEEE-754 standards; this includes exactly when normalization is expected.

Anonymous Echidna 9mth #469bda ✓ Resolved

SP23-MT-Q4.3

Is this a valid answer?

jal t0 temp_label

---

lui rd imm

addi t0 t0 -4

add rd t0 rd

Not sure if the addi t0 t0 -4 is correct (my logic was that since t0 = PC + 4) in jal instruction, that we had to decrement by -4 to bring t0 back to PC, before adding it in the last line.

> Other answers are possible here, e.g. putting the PC in `t0` and the immediate in `rd` before adding.

Anonymous Echidna 9mth #469bdb

I'm also not sure as to why the solution says we need to add 8:

> **Solution:**
>
> ```
> jal rd temp_label
> addi rd rd 8
> lui t0 imm
> add rd rd t0
> ```
>
> This one is tricky. The first thing to remember is what `auipc` does. First, it takes a 20-bit `imm`, and creates an immediate with these 20 bits as the upper 20 bits and 0s as the lower 12 bits. Then, it adds this new immediate with the current PC.
>
> First, we have to get the value of the current PC. Looking through the reference card, the only instructions that put the PC in a register are the jump instructions. Here, we use `jal` to get the address of the `jal` instruction, plus 4 (i.e. the address of Line 2 here) into register `rd`.
>
> However, the question says that we should be adding to the PC of the final instruction in our answer. Since our answer uses all 4 lines, and we got the address of Line 2, we need to add 8 bytes = 2 instructions to the PC we got, to find the PC of the final instruction.

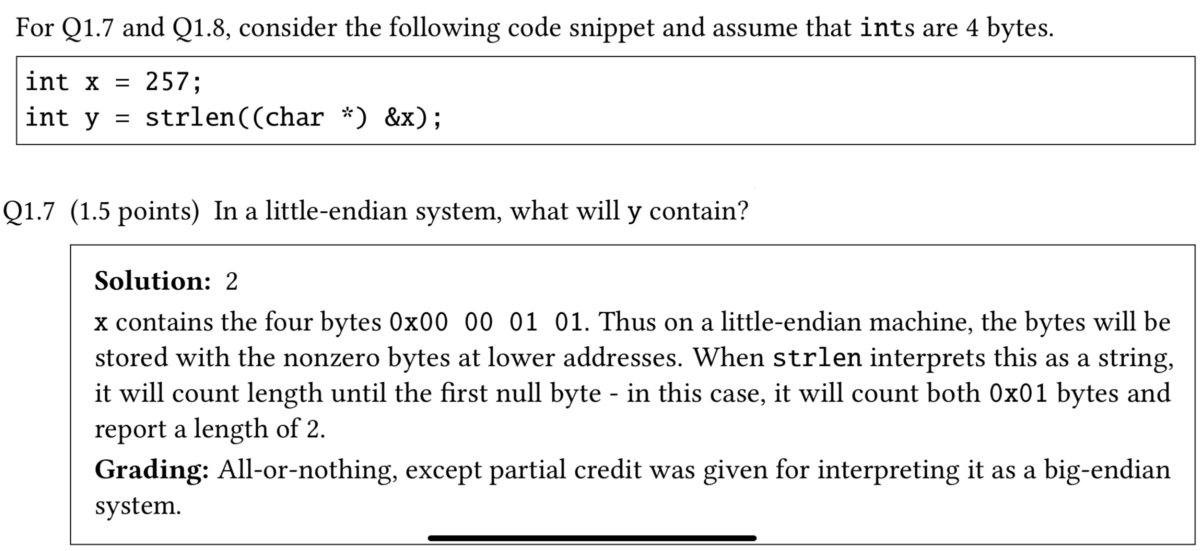
Why do we even need to add bits in the first place here?

Justin Yokota STAFF 9mth #469cdc

This would not work; per the question you were required to set rd according to the PC of the last line of code. Your solution sets rd according to the PC of the first line of code.

Anonymous Turtle 9mth #469bcf ✓ Resolved

Su23
How did we get these four bytes and how do we know it address?



For Q1.7 and Q1.8, consider the following code snippet and assume that ints are 4 bytes.

```
int x = 257;
int y = strlen((char *) &x);
```

Q1.7 (1.5 points) In a little-endian system, what will y contain?

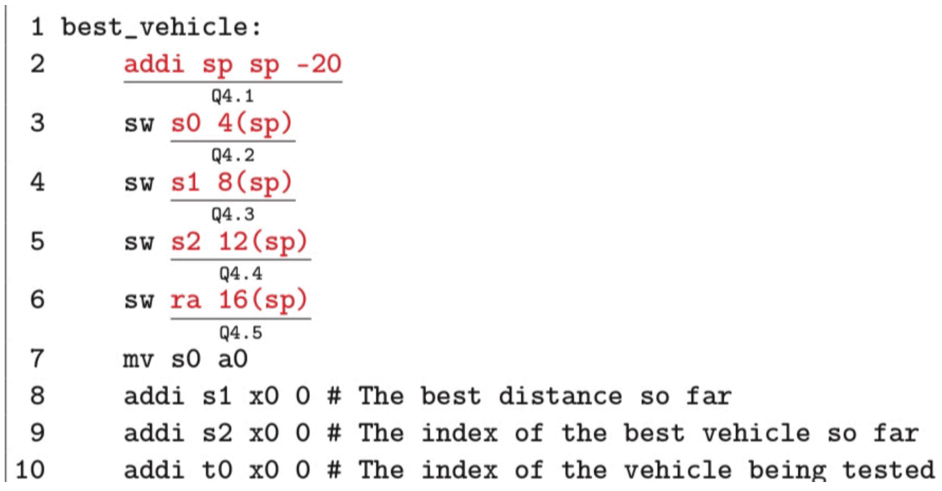
**Solution:** 2
x contains the four bytes 0x00 00 01 01. Thus on a little-endian machine, the bytes will be stored with the nonzero bytes at lower addresses. When strlen interprets this as a string, it will count length until the first null byte - in this case, it will count both 0x01 bytes and report a length of 2.
**Grading:** All-or-nothing, except partial credit was given for interpreting it as a big-endian system.

Justin Yokota STAFF 9mth #469bfb

Ints are 4 bytes long per the question, and endianness determines the exact order of the bytes. 257 is stored as hex 0x00 00 01 01.

Anonymous Dugong 9mth #469bce ✓ Resolved

FA23-MT-Q4:
Why is line 3 not `sw s0 0(sp) -16`?
I'm confused on this because on the calling convention document it's like stated in that way and I remembered following that during project 2b. Attached below is the solutions and the calling convention part im referencing.



```
 1 best_vehicle:
 2     addi sp sp -20
          Q4.1
 3     sw s0 4(sp)
          Q4.2
 4     sw s1 8(sp)
          Q4.3
 5     sw s2 12(sp)
           Q4.4
 6     sw ra 16(sp)
           Q4.5
 7     mv s0 a0
 8     addi s1 x0 0 # The best distance so far
 9     addi s2 x0 0 # The index of the best vehicle so far
10     addi t0 x0 0 # The index of the vehicle being tested
```

```
sum_squares:

prologue:
    addi sp sp -16
    sw s0 0(sp)
    sw s1 4(sp)
    sw s2 8(sp)
    sw ra 12(sp)

    li s0 1
    mv s1 a0
    mv s2 0
```

Anonymous Hippopotamus 9mth #469cac ENDORSED

It is because later on in the code, t0 is temporarily being stored at 0(sp) during the race function call. So it must be left open in the prologue, and we start saving at position 4(sp) instead.

Anonymous Moose 9mth #469bcd ✓ Resolved

SU23-MT-Q2.14

Wouldn't MAX_BORROWS be part of data/static since it is declared outside main? This question is also similar to SP23-MT-Q2.10 where NUM_PAGES is on the data/static.

Justin Yokota STAFF 9mth #469cdd

MAX_BORROWS is a define statement; it will get replaced by the actual number 25 before compilation, so it won't actually stay as a variable. The actual number 25 will get stored as an immediate of an instruction.

Anonymous Squid 8mth #469feb

So because the define gets replaced by a constant even before compilation, there's no "variable to reference", so it gets put into the text section instead of data?

*This comment was deleted*

damien toh 8mth #469acef

Replying to (Deleted)

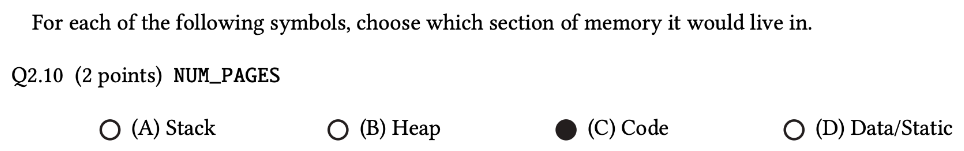
hi why is NUM_PAGES is on the data/static in SP23-MT-Q2.10

M

Myrah Shah STAFF 8mth #469acfc

Replying to damien toh

Sorry, can you point me to what you're looking at?


For each of the following symbols, choose which section of memory it would live in.

Q2.10 (2 points) NUM_PAGES

○ (A) Stack ○ (B) Heap [✓ (C) Code] ○ (D) Data/Static

damien toh 8mth #469adad

Replying to Myrah Shah

sorry! i think i saw the wrong things. thanks for checking anyways

M

Myrah Shah STAFF 8mth #469acfd

Replying to Anonymous Squid

Yep!

Anonymous Duck 9mth #469bcc ✓ Resolved

Fa23-MT-Q3.5

Where did 2^-62 come from? Can you explain the reasoning of how to approach this question?

J

Justin Yokota STAFF 9mth #469cde

2^-62 is the exponent of denorms in this question. Mostly our goal in this question is to find what numbers have no representable numbers between k and 2k. Floating point numbers are designed to represent numbers to a certain relative precision, and the goal in question requires an extremely low relative precision. This only happens at the tail end of underflow, during the denorms.

Anonymous Elephant 8mth #469abcb

Will the step size between denorms for a specified number of mantissa bits always the the same (since the exponent is always all zeros)? Whereas the step size between normal can increase/decrease since we can manipulate the exponent bits? So that is why calculating the step size of denorms is sufficient for this question?

Anonymous Cormorant 9mth #469bae ✓ Resolved

Fa23-Final-Q2.7

The solution says "*this instruction also performs one read and one write memory operation in the same cycle, which is currently not supported by our DMEM (as it only has one port). Therefore, we need to allow the DMEM to read and write in the same cycle*", does this mean that we can just add another output port to DMEM, who outputs the original value at `MemAddress` (i.e. `rs1+imm`), in case the direct output of that memory value could be overwritten before finishing writing into the `rd` register?

And is it correct to set the relative control signals for the instruction ( `alsw` ) to be `RegWEn=1` , `MemRW=1` and `WBSel=3` (selecting DMEM's new output port)?

Justin Yokota STAFF 9mth #469cdf

Which question are you referring to? Fa23 Q2 is a C coding question.

Anonymous Cormorant 9mth #469cec

It's the question from the FA 2023 final, which is related to single-cycle data path mentioned in week 9's announcement.

Justin Yokota STAFF 9mth #469ced

Replying to Anonymous Cormorant

Ah, for the final. We don't really need to add another output port; we just need to ensure the output port still works and keeps its old value even when we write a new value to that location.

Anonymous Cormorant 9mth #469cee

Replying to Justin Yokota

Oh, I see, thank you for your explanation! And in this case, we have to keep both RegWEn and MemRW to be 1, is this correct?

Justin Yokota STAFF 9mth #469cef

Replying to Anonymous Cormorant

Probably. You may need a new MemRW signal, depending on the exact way you change the DMEM.

Anonymous Quail 9mth #469bad ✓ Resolved

SU23-MT-Q2.1: Would

while(i < strlen(&user_ids )-1) be an acceptable answer?

Andrew Liu STAFF 9mth #469baf

No, since calling `strlen` on `&user_ids` (which is of type `char***` ) is undefined behavior.

Adrian Bao 9mth #469bac ✓ Resolved

SP23-Q4

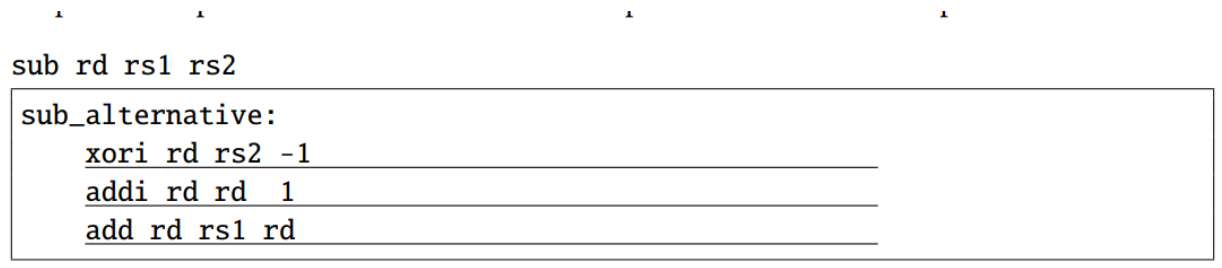
Why is xori needed and is initialized when we could have done

lw t0 0(rs1)

lw t1 0(rs2)

add rd t0 t1?

Thanks!



```
sub rd rs1 rs2
sub_alternative:
    xori rd rs2 -1
    addi rd rd  1
    add rd rs1 rd
```

Andrew Liu STAFF 9mth #469bba

`xori` here flips all the bits, since we are using 2's complement, the way to negate a number is to flip all the bits and add 1, which is what this is doing. Your solution would not have worked.

You cannot use extra registers, and you cannot load from arithmetic inputs. e.g. Your pseudo would segfault if asked

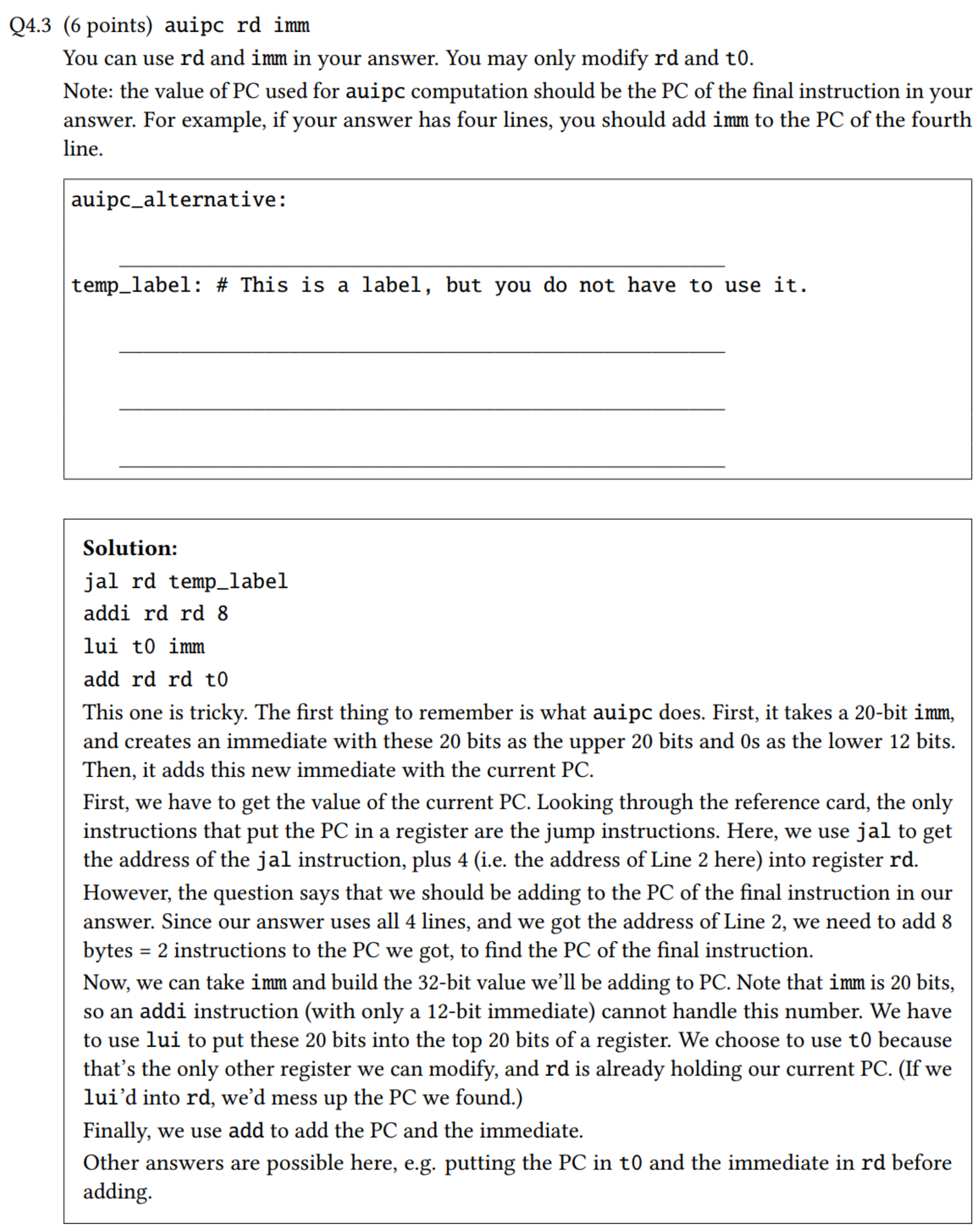
```
li t0 1
li t2 3
sub a0 t2 t0
```

1

Adrian Bao 9mth #469bab ✓ Resolved

SP23-Q4.3

Does this mean that the rd and PC are the same thing since auipc is meant to add to the PC and we are adding to rd?



Q4.3 (6 points) `auipc rd imm`

You can use `rd` and `imm` in your answer. You may only modify `rd` and `t0`.

Note: the value of PC used for `auipc` computation should be the PC of the final instruction in your answer. For example, if your answer has four lines, you should add `imm` to the PC of the fourth line.

```
auipc_alternative:

    ______________________________________________

temp_label: # This is a label, but you do not have to use it.

    ______________________________________________

    ______________________________________________

    ______________________________________________
```

**Solution:**

```
jal rd temp_label
addi rd rd 8
lui t0 imm
add rd rd t0
```

This one is tricky. The first thing to remember is what `auipc` does. First, it takes a 20-bit `imm`, and creates an immediate with these 20 bits as the upper 20 bits and 0s as the lower 12 bits. Then, it adds this new immediate with the current PC.

First, we have to get the value of the current PC. Looking through the reference card, the only instructions that put the PC in a register are the jump instructions. Here, we use `jal` to get the address of the `jal` instruction, plus 4 (i.e. the address of Line 2 here) into register `rd`.

However, the question says that we should be adding to the PC of the final instruction in our answer. Since our answer uses all 4 lines, and we got the address of Line 2, we need to add 8 bytes = 2 instructions to the PC we got, to find the PC of the final instruction.

Now, we can take `imm` and build the 32-bit value we'll be adding to PC. Note that `imm` is 20 bits, so an `addi` instruction (with only a 12-bit immediate) cannot handle this number. We have to use `lui` to put these 20 bits into the top 20 bits of a register. We choose to use `t0` because that's the only other register we can modify, and `rd` is already holding our current PC. (If we `lui`'d into `rd`, we'd mess up the PC we found.)

Finally, we use `add` to add the PC and the immediate.

Other answers are possible here, e.g. putting the PC in `t0` and the immediate in `rd` before adding.

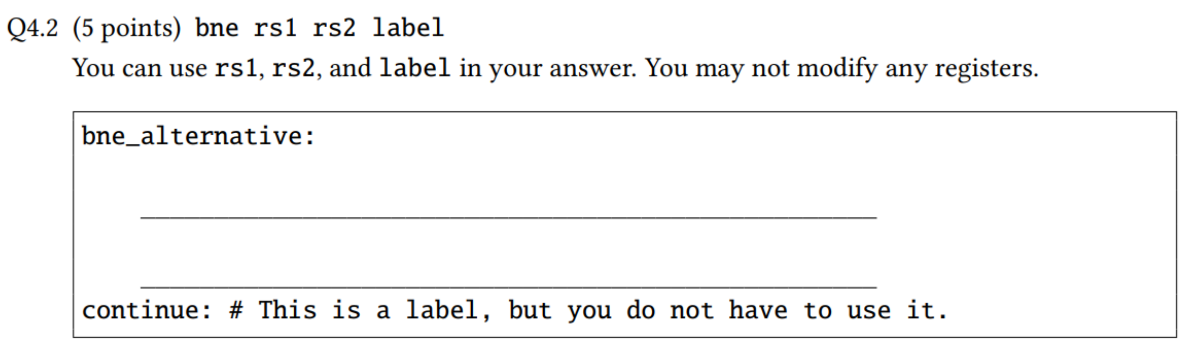
Andrew Liu STAFF 9mth #469bbb

`rd` stands for "register destination." It is a generic description describing where an instruction will write to.

Adrian Bao 9mth #469baa ✓ Resolved

SP23-Q4.2

For the last line, I said jal label instead of jal x0 label. What is the difference, and does my solution work?



Q4.2 (5 points) `bne rs1 rs2 label`

You can use `rs1`, `rs2`, and `label` in your answer. You may not modify any registers.

```
bne_alternative:

    ________________________________________________

    ________________________________________________
continue: # This is a label, but you do not have to use it.
```

Andrew Liu STAFF 9mth #469bbc

`jal label` is equivalent to `jal ra label`, while `jal x0 label` is equivalent to `j label`. Your solution would've likely received partial credit, however.

1

Adrian Bao 9mth #469bcb

Thanks. Where would the program execution go since the return address is blank instead of ra, which would drop us off right before continue?

Anonymous Snake 9mth #469afe ✓ Resolved

SP23-MT-Q4.3

Would this work:

jal rd temp_label

slli t0 imm 12

addi rd rd 8

add rd rd t0

Q4.3 (6 points) `auipc rd imm`

You can use `rd` and `imm` in your answer. You may only modify `rd` and `t0`.

Note: the value of PC used for `auipc` computation should be the PC of the final instruction in your answer. For example, if your answer has four lines, you should add `imm` to the PC of the fourth line.

```
auipc_alternative:

    ________________________________________________
temp_label: # This is a label, but you do not have to use it.

    ________________________________________________

    ________________________________________________

    ________________________________________________
```

**Solution:**

```
jal rd temp_label
addi rd rd 8
lui t0 imm
add rd rd t0
```

Andrew Liu STAFF 9mth #469bbd

This would not work in general, since `slli` only accepts 12 immediate bits, although this answer would likely receive partial credit

Anonymous Snake 9mth #469bde

Does that mean the imm(32bits) input for auipc is too large to be a valid input(12bits) for slli?

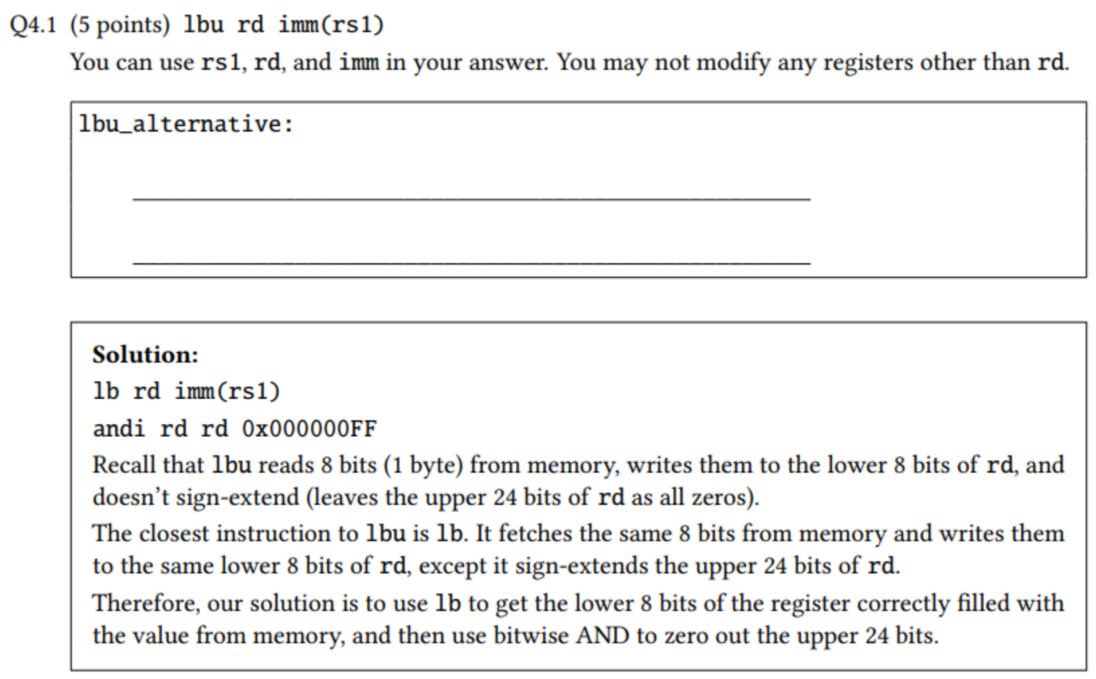
Justin Yokota STAFF 9mth #469cea

Replying to Anonymous Snake

Yes. Another issue is that "slli" takes in a register and an immediate. You try to send in two immediates, which is not allowed.

Anonymous Penguin 9mth #469afc ✓ Resolved

SP23-MT-Q4.1



Q4.1 (5 points) `lbu rd imm(rs1)`

You can use `rs1`, `rd`, and `imm` in your answer. You may not modify any registers other than `rd`.

```
lbu_alternative:

    ________________________________________

    ________________________________________
```

**Solution:**

```
lb rd imm(rs1)
andi rd rd 0x000000FF
```

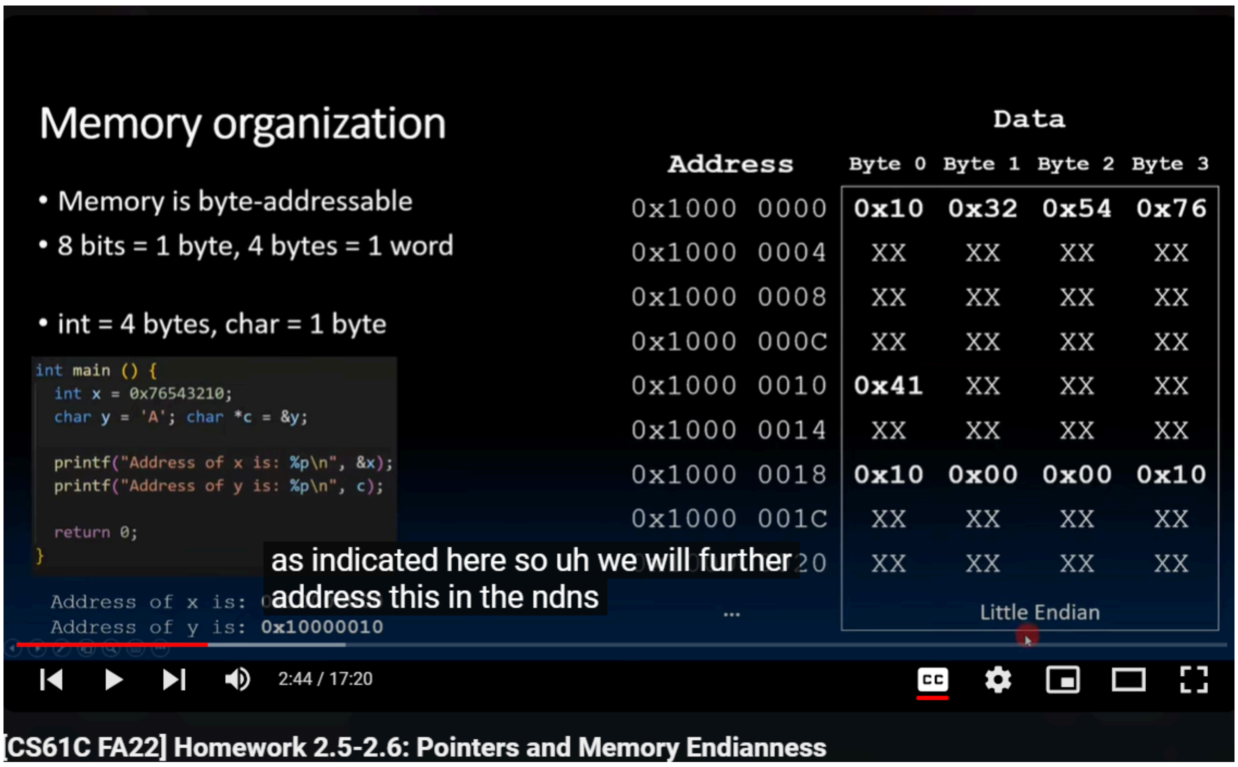
Recall that `lbu` reads 8 bits (1 byte) from memory, writes them to the lower 8 bits of `rd`, and doesn't sign-extend (leaves the upper 24 bits of `rd` as all zeros).

The closest instruction to `lbu` is `lb`. It fetches the same 8 bits from memory and writes them to the same lower 8 bits of `rd`, except it sign-extends the upper 24 bits of `rd`.

Therefore, our solution is to use `lb` to get the lower 8 bits of the register correctly filled with the value from memory, and then use bitwise AND to zero out the upper 24 bits.

I'm not sure why the hex is 0x000000FF. If we want to zero out the upper 24 bits and save the lower 8 bits, shouldn't it be 0xFF000000 instead since we are assumed to be in little endian where the lower bits are to the left? Thank you.

For additional context, how I thought lb works is by taking those 8 bits and writing them to the lowest 8 bits of rd like how  'A' in the example below was written to the lowest byte of address Y.



## Memory organization

- Memory is byte-addressable
- 8 bits = 1 byte, 4 bytes = 1 word
- int = 4 bytes, char = 1 byte

```
int main () {
  int x = 0x76543210;
  char y = 'A'; char *c = &y;

  printf("Address of x is: %p\n", &x);
  printf("Address of y is: %p\n", c);

  return 0;
}
```

```
Address of x is: 
Address of y is: 0x10000010
```

| | Data | | | |
|---|---|---|---|---|
| Address | Byte 0 | Byte 1 | Byte 2 | Byte 3 |
| 0x1000 0000 | 0x10 | 0x32 | 0x54 | 0x76 |
| 0x1000 0004 | XX | XX | XX | XX |
| 0x1000 0008 | XX | XX | XX | XX |
| 0x1000 000C | XX | XX | XX | XX |
| 0x1000 0010 | 0x41 | XX | XX | XX |
| 0x1000 0014 | XX | XX | XX | XX |
| 0x1000 0018 | 0x10 | 0x00 | 0x00 | 0x10 |
| 0x1000 001C | XX | XX | XX | XX |
| 20 | XX | XX | XX | XX |

Little Endian

as indicated here so uh we will further address this in the ndns

[CS61C FA22] Homework 2.5-2.6: Pointers and Memory Endianness

Adrian Bao 9mth #469aff

I believe it is FF because FF is a filler value that is true (1) since it is non-zero, and the eight bits we read is also true (1) since it is non-zero, and 1 AND 1 = 1.

Andrew Liu STAFF 9mth #469bbe

This is due to the immediate being "what you mean" and not necessarily "what the computer does"

For example, if I, the programmer, write `0x000000FF` I *definitely mean* a number where the 8 least significant bits are 1 and the 24 most significant bits are 0. Whatever the computer does, whether it converts it to little endian or not, is abstracted away.

Anonymous Penguin 9mth #469bca

Thanks for the explanation, but I'm still confused because if we are writing to the lower 8 bits, how can we sign extend if we are already at the leftmost limit of the register? Thanks again.

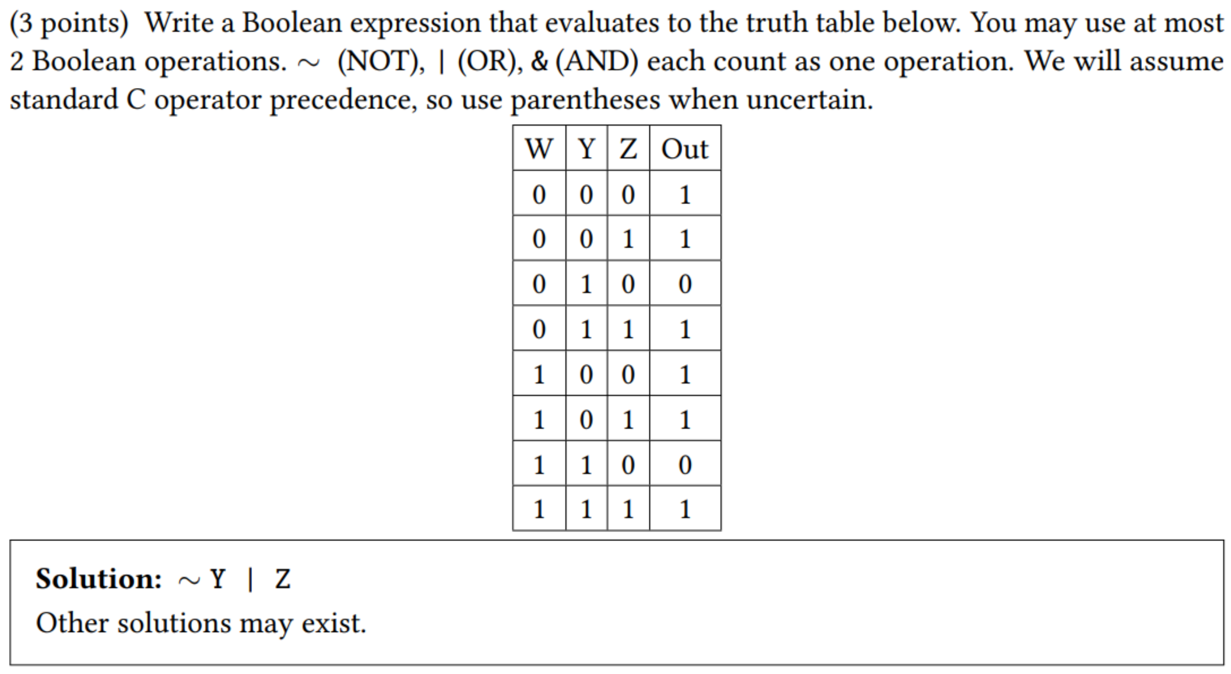
Justin Yokota STAFF 9mth #469ceb

Replying to Anonymous Penguin

What do you mean by "leftmost limit"? I don't think we want to sign-extend here, since we have a lbu.

Adrian Bao 9mth #469afa ✓ Resolved

SP23-Q1.7



(3 points) Write a Boolean expression that evaluates to the truth table below. You may use at most 2 Boolean operations. ~ (NOT), | (OR), & (AND) each count as one operation. We will assume standard C operator precedence, so use parentheses when uncertain.

| W | Y | Z | Out |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Solution:** ~ Y | Z
Other solutions may exist.

How do we know that we need to negate Y? Is there a good way to approach the problem apart from pattern detection? Thanks!

Justin Yokota STAFF 9mth #469afb

There's no algorithmic solution for this, unfortunately; in fact, if you find an efficient algorithmic solution, then P = NP.

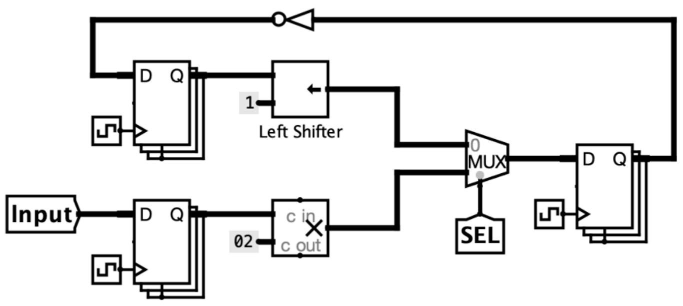
♡ 1 ···

Anonymous Sand Dollar 9mth #469aeb ✓ Resolved

Spring 2023, Q6.

**Q6** ***SDS*** **(12 points)**

Consider the following circuit diagram. SEL is a single bit control signal that updates instantaneously at the rising edge of every clock cycle and remains stable during any given clock cycle. You may assume that `Input` will not cause any hold time violations.



$$t_{\text{NOT}} = 8 \text{ ps}$$
$$t_{\text{mux}} = 25 \text{ ps}$$
$$t_{\text{multiplier}} = 1000 \text{ ps}$$
$$t_{\text{shifter}} = 2 \text{ ps}$$
$$t_{\text{clk-q}} = 30 \text{ ps}$$
$$t_{\text{setup}} = 20 \text{ ps}$$

The left shifter combinational logic block shifts the top input by the number of bits indicated by the bottom input. The shifter in the diagram shifts the output of the connected register left by 1 bit.

I get confused on how you determine where the CL path can start and end. Do you just always look for the time from any "timed" element to another? What exactly is considered a "timed" element? In this case, SEL didn't have a clock attached to it but was still considered a "timed" element because it's affecting directly by the rising edge.

Q6.2 (3 points) What is the maximum hold time the registers can have so that there are no hold time violations in the circuit above? Reminder: you may assume that `Input` will not cause any hold time violations.

**Solution:** 25 ps

The shortest path between any two timed elements is actually the path from the SEL signal, which changes instantly at the rising edge of the clock, to the right register. This path has only delay 25 ps from the mux.

If you didn't see this path, the next-shortest path starts from the rightmost register and goes around, through the NOT gate, to the top-left register. This path has a delay of 30 ps (clk-to-q from the rightmost register) and 8 ps (from the NOT gate), for a total of 38 ps. Partial credit was given for this answer.

♡ ···

Andrew Liu STAFF 9mth #469bbf

Yep! The CL path can start at the Q end of *any* state element (element that requires or depends on, or is related to a clock input), and must end at the D end of *any* state element (including where it started)

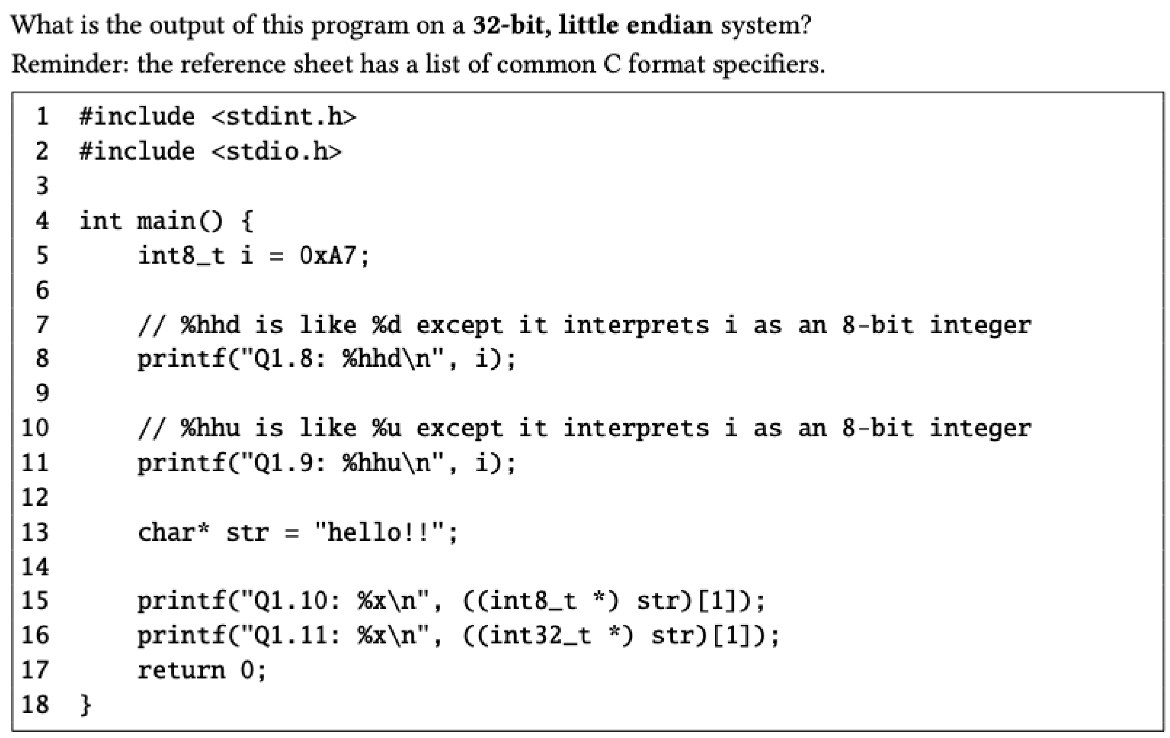
SEL is considered clocked because it behaves like a state element — only changing in relation to clock ticks.

P.S. "updates instantly on the clock" is the same as "from a register with 0 clk-to-q"

♡ ···

Anonymous Sand Dollar 9mth #469adf ✓ Resolved

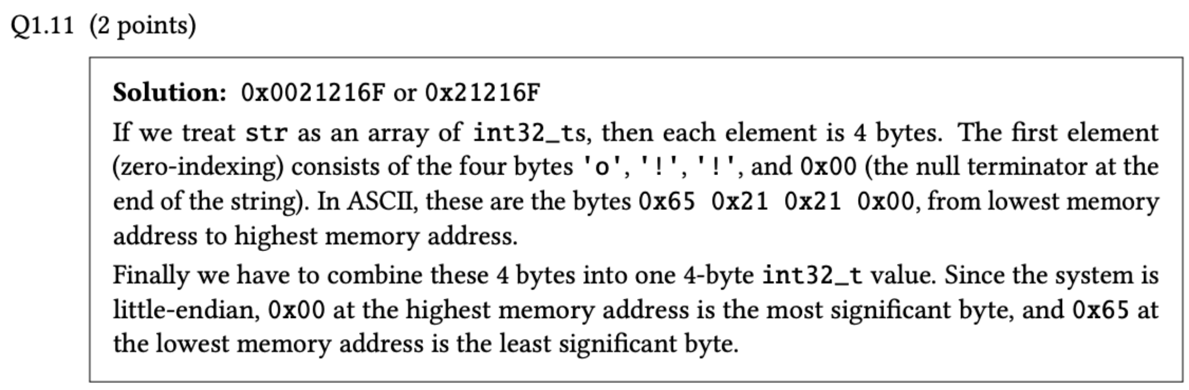
Spring 2023, Q. 1.



What is the output of this program on a **32-bit, little endian** system?
Reminder: the reference sheet has a list of common C format specifiers.

```
#include <stdint.h>
#include <stdio.h>

int main() {
    int8_t i = 0xA7;

    // %hhd is like %d except it interprets i as an 8-bit integer
    printf("Q1.8: %hhd\n", i);

    // %hhu is like %u except it interprets i as an 8-bit integer
    printf("Q1.9: %hhu\n", i);

    char* str = "hello!!";

    printf("Q1.10: %x\n", ((int8_t *) str)[1]);
    printf("Q1.11: %x\n", ((int32_t *) str)[1]);
    return 0;
}
```

I don't understand why little endian order doesn't matter when we are looking at just 8 bit int but it does matter for 32 bit int when it comes to question 1.11



Q1.11 (2 points)

**Solution:** 0x0021216F or 0x21216F

If we treat `str` as an array of `int32_t`s, then each element is 4 bytes. The first element (zero-indexing) consists of the four bytes `'o'`, `'!'`, `'!'`, and `0x00` (the null terminator at the end of the string). In ASCII, these are the bytes `0x65 0x21 0x21 0x00`, from lowest memory address to highest memory address.

Finally we have to combine these 4 bytes into one 4-byte `int32_t` value. Since the system is little-endian, `0x00` at the highest memory address is the most significant byte, and `0x65` at the lowest memory address is the least significant byte.

♡ 1 ···

Justin Yokota STAFF 9mth #469aea

Endianness affects how an array of bytes get read as a single value. If you have the letters "DRAW" on a piece of paper, you can determine that the leftmost letter is a "D" without thinking about whether you're reading left-to-right or right-to-left. But if you're trying to read the whole word, you'll either get "DRAW" or "WARD" depending on how you read.

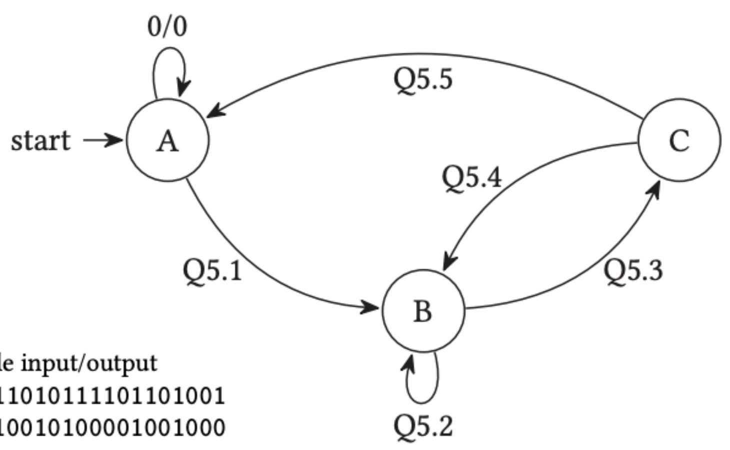
♡ 1 ···

Anonymous Sand Dollar 9mth #469ade ✓ Resolved

Spring 2023, Q5. Is this in-scope for our midterm and if so, where was this covered in class or discussion?

**Q5** ***FSM*** **(10 points)**

The following finite state machine (FSM) should output 1 if the last 3 input bits are 101. Otherwise, it should output 0.



Example input/output

```
Input:    1011010111101101001
Output:   0010010100001001000
```

For each of the transitions above, fill in the appropriate input and output values such that the FSM behaves as described.

Jedidiah Tsang STAFF 9mth #469aee

FSMs are not in scope for the midterm

Anonymous Chinchilla 9mth #469adc ✓ Resolved

**FA23-MT-Q2.1** I'm confused about line 7. Why are we reallocing v's slices and incrementing the number of slices? and I'm confused about the differences between data and slices in a vector_t in general.

Jero Wang STAFF 9mth #469ebb

`data` is the actual data of a vector (i.e. if i have the vector `[0, 1, 2]`, data would point to that). `slices` is a pointer to a list of `vector`s that are slices of the current vector, which all share the same `data` as the slice you're working with.

We increment `num_slices` and reallocate for more space here because we're creating a new slice and need to keep track of it in our current `vector_t` struct.

Anonymous Chinchilla 9mth #469adb ✓ Resolved

**FA23-MT-Q1.4** How do we know need to use 2's complement instead of sign magnitude? Or is it default to use 2's complement for negative numbers?

1

Justin Yokota STAFF 9mth #469add

2's complement is always used unless specified; it's considered mandated by both C and RISC-V, and has so many advantages over other formats that there aren't really any systems that use something else.

2

Anonymous Chinchilla 9mth #469ada ✓ Resolved

**FA23-MT-Q6.3 Q6.4** can we use decimal in the for loop to compare the favorite number and integer i starts from 0 to 2^4-1? In this case, how do we solve 6.3 differently and what is the runtime for 6.4?

J

Justin Yokota STAFF 9mth #469aef

You can use decimal, but that doesn't affect bitwise ops (since bitwise ops implicitly treat its inputs as binary).

Anonymous Parrot 9mth #469abc ✓ Resolved

**sp23-mt-2.3**

Why do we not not need to allocate memory with malloc(sizeof (Page)) but instead can do &sheets ->pages[i]?

Isn't Cheatsheet Page pages unintialized first of all?

Is &sheets ->pages[i] just setting Cheat sheet pages[i] to a pointer Page* page?

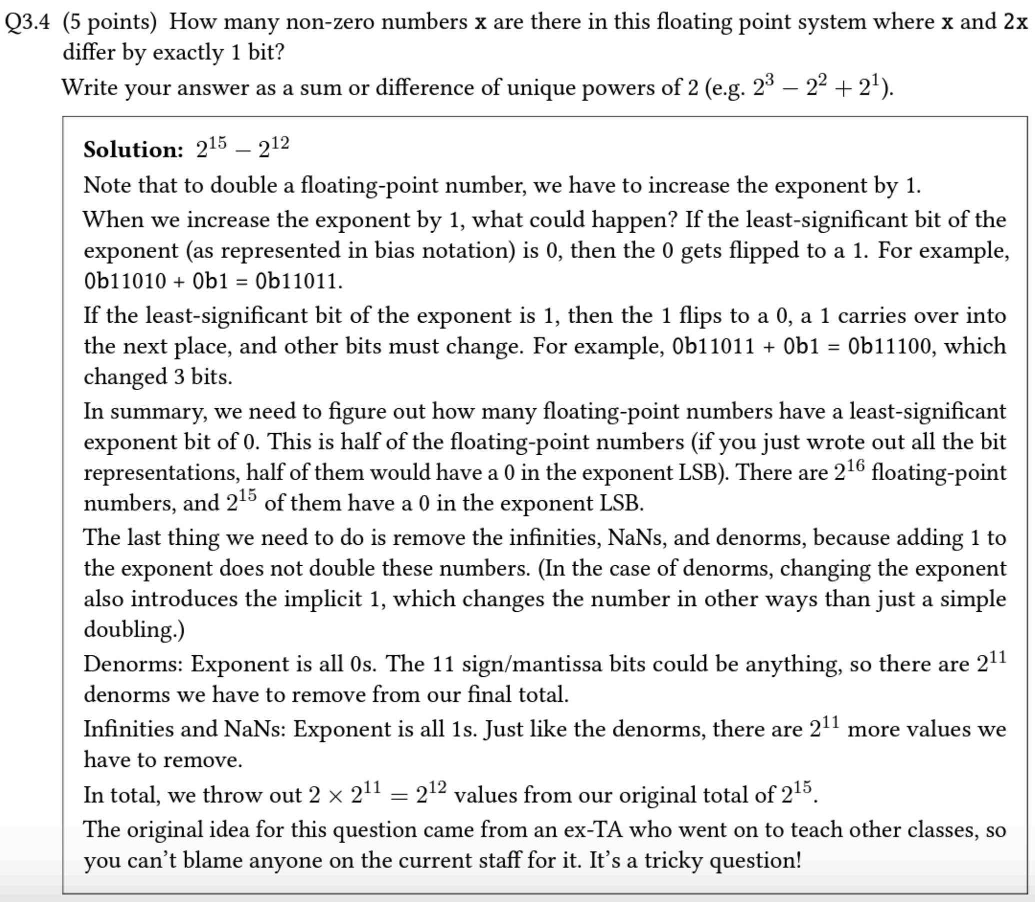
1

E

Erik Yang STAFF 9mth #469ace

when we calloc data for the Cheatsheet struct, then it automatically allocates memory for the pages array of size NUM_PAGES, so each page in the array is already instantiated. Note that pages is an array with a set size

Wesley Kai Zheng 9mth #469abb ✓ Resolved



Q3.4 (5 points) How many non-zero numbers **x** are there in this floating point system where **x** and **2x** differ by exactly 1 bit?

Write your answer as a sum or difference of unique powers of 2 (e.g. $2^3 - 2^2 + 2^1$).

**Solution:** $2^{15} - 2^{12}$

Note that to double a floating-point number, we have to increase the exponent by 1.

When we increase the exponent by 1, what could happen? If the least-significant bit of the exponent (as represented in bias notation) is 0, then the 0 gets flipped to a 1. For example, `0b11010 + 0b1 = 0b11011`.

If the least-significant bit of the exponent is 1, then the 1 flips to a 0, a 1 carries over into the next place, and other bits must change. For example, `0b11011 + 0b1 = 0b11100`, which changed 3 bits.

In summary, we need to figure out how many floating-point numbers have a least-significant exponent bit of 0. This is half of the floating-point numbers (if you just wrote out all the bit representations, half of them would have a 0 in the exponent LSB). There are $2^{16}$ floating-point numbers, and $2^{15}$ of them have a 0 in the exponent LSB.

The last thing we need to do is remove the infinities, NaNs, and denorms, because adding 1 to the exponent does not double these numbers. (In the case of denorms, changing the exponent also introduces the implicit 1, which changes the number in other ways than just a simple doubling.)

Denorms: Exponent is all 0s. The 11 sign/mantissa bits could be anything, so there are $2^{11}$ denorms we have to remove from our final total.

Infinities and NaNs: Exponent is all 1s. Just like the denorms, there are $2^{11}$ more values we have to remove.

In total, we throw out $2 \times 2^{11} = 2^{12}$ values from our original total of $2^{15}$.

The original idea for this question came from an ex-TA who went on to teach other classes, so you can't blame anyone on the current staff for it. It's a tricky question!

SP23-MT-Q3.4. I understand all the logic behind the answer except for one part. When I was solving the questions, I thought infinities and NaNs were already thrown out in the first step (as we already discarded everything that does not have a least significant bit of 1). In that case, would not the answer be 2^15-2^11? This is what I actually had as my final result.

Justin Yokota STAFF 9mth #469aca

Half of all NaNs have an LSB of 1.

Wesley Kai Zheng 9mth #469acb

Wait, I thought NaNs and infinities both have all 1s as the LSB of their exponent component.

Justin Yokota STAFF 9mth #469acc

Replying to Wesley Kai Zheng

Ah, yes, sorry. Misread your question. The main thing is that while we've removed infinities and NaNs, we haven't removed the real numbers that end up "doubling" to an infinity or NaN. If we have an exponent of 0b1111 1110, we'd count this in our initial count, but "doubling" it would yield 0b1111 1111 on the exponent, which would correspond to a NaN or infinity.

1

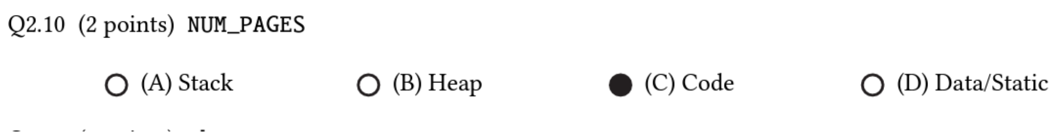
Wesley Kai Zheng 9mth #469acd

Replying to Justin Yokota

Ohhhhh. Great, I now understand it!

Wesley Kai Zheng 9mth #469aba ✓ Resolved

SP23-MT-Q2.10 Why is NUM_PAGES not stored in Data/Static?



Q2.10 (2 points) NUM_PAGES

(A) Stack (B) Heap [✓ (C) Code] (D) Data/Static

Anonymous Guanaco 9mth #469abd

It's by #define

Anonymous Quail 9mth #469aae ✓ Resolved

SP23-MT-Q2.1 - Why do you need to use calloc, rather than malloc?

Anonymous Guanaco 9mth #469aaf

You need to initialize total_length to be 0 to use it in the Q2.8 calculation, other wise total_length would contain some uninitialized garbage.

1

Anonymous Guanaco 9mth #469aab ✓ Resolved

[SU23-MT-Q2.9]

We didn't allocate memory for borrowed books on the heap, so right now borrowed_books is only an uninitialized place holder on the heap that doesn't point to a specific mem block, wouldn't this error if we call memset on it?

Erik Yang STAFF 9mth #469acf

Similar to #469abc, the User struct has a predefined size of MAX_BORROWS. When you allocate the entire User struct, it will allocate data for those predefined number of Books.

Anonymous Quail 9mth #469fe ✓ Resolved

SP23-MT-1.11

If 'o' is 0x65 in ASCII, how did it become 0x6F in the answer?

1

Anonymous Guanaco 9mth #469aaa

'o' is not 0x65 in ascII, it's actually 0x6F if you check the convert table

Anonymous Quail 9mth #469aac

I see that now, but I'm curious as to where 0x65 came from in the answer key?

Justin Yokota STAFF 9mth #469aad

Replying to Anonymous Quail

I think it was a typo? 0x65 is 'e', so I think it's a copy-paste error.

1

Anonymous Guanaco 9mth #469fb ✓ Resolved

[SU23-MT-Q3.5]

I think the answer should also include infinity:

if original number = 1 /1111 1111/ 00000.....1

after shifting it becomes = 0/ 1111 1111/ 000000....0000 = + inf

the answer appears to have overlooked this value.

Justin Yokota STAFF 9mth #469ff

Not quite; you'd end up with 0/ 1111 1111 / 10000000....0 = NaN

Anonymous Parrot 9mth #469ef ✓ Resolved

[sp23 1.11] Why do we start from after 4 chars? "hell | o!!", I thought we had to start from e because [1] and then include the next 4 char which are "llo!"

Nikhil Kandkur STAFF 9mth #469fd

We are casting our pointer to a int32_t pointer, which means that each index will look at 4 bytes at a time, instead of 1 like in a character pointer. Since `sizeof(int32_t) = 4`, we have to skip the first 4 bytes (which is the same as 4 characters) to get the bytes at index 1 as a result.

Anonymous Turtle 9mth #469ed ✓ Resolved

[sp23 2.12] is the answer heap because all pointers are stored in the heap?

Andrew Liu STAFF 9mth #469ee

The answer is heap because `*sheet` is dereferencing the pointer `sheet`, which was returned by `calloc`.

Anonymous Turtle 9mth #469eb ✓ Resolved

[sp23 1.11] why is "o!!" the first index?

Anonymous Guanaco 9mth #469abf

Each index is interpreted as 4bytes (chars) after casting

Anonymous Turtle 9mth #469dd ✓ Resolved

[sp23 2.2] Why are we using -> instead of . if we use the dereference operator?

Andrew Liu STAFF 9mth #469de

~~There is no dereference operator in that line; the function argument defines `lib` as a pointer to a `Library`, so we must use `->`~~

Oops sorry I looked at Su23 2.2

There still is no dereference operator in this line, it is actually part of the pointer declaration from the line directly above it, where `sheet` is of type `Cheatsheet*`

Anonymous Turtle 9mth #469ec

By derefernce operator, I mean &

Nikhil Kandkur STAFF 9mth #469fc

Replying to Anonymous Turtle

`&` will give you the address of the struct, which means that if we have a `Page p`, `&p` is equivalent to having a `Page*`. `sheet->page[i]` will give us a Page struct, so to make it a pointer, we have to tag a `&` before the call to actually get a pointer.

EDIT: I made a mistake in my explanation, the new explanation should be correct.

Anonymous Turtle 9mth #469aec

Replying to Nikhil Kandkur

So it's the same as &(sheet->pages[i])?

N Nikhil Kandkur STAFF 9mth #469aed

Replying to Anonymous Turtle

Yup!

Anonymous Coyote 9mth #469ce ✓ Resolved

[Sp23 Q3.3] I don't understand why the exponent bit is 0b11001, I thought because it's 17, it's 0b10001.

E Eric Che STAFF 9mth #469dc

> Exponent bits: 2 − (−15) = 17, which in unsigned 5-bit binary is 0b10001. We can increase the number by flipping the most-significant 0 to a 1, which would produce 0b11001.

Anonymous Coyote 9mth #469cc ✓ Resolved

[Sp23 Q5] Is this in scope of the midterm?

1

J Justin Yokota STAFF 9mth #469cd Visible to staff only

Finite State Machines are not in scope for the midterm, right?

E Eddy Byun STAFF 9mth #469db Visible to staff only

correct

E Eddy Byun STAFF 9mth #469da

This question is not in scope for the midterm

Anonymous Trout 9mth #469cb ✓ Resolved

[FA23-MT-Q3.5&3.6]

Q3.5 (3 points) List all numbers $k$ in this floating-point representation such that the smallest floating-point number strictly greater than $k$ is exactly $2k$. Express your answer as $x \times 2^y$, where $x$ is a decimal such that $1 \leq |x| < 2$ and $y$ is an integer. If there are no such numbers, write "None".

**Solution:** The wording of the question doesn't allow for negative answers - something greater than a negative number will always have smaller absolute value.

Additionally, $k$ cannot be a normal number, since we essentially need $k$ to be equal to the step size between representable numbers. Even among the denormal numbers, where the step size is $2^{-62} \times 2^{-8} = 2^{-70}$, $k = 2^{-70}$ is the only value that works.

**Grading:** Half credit was awarded for off-by-one errors ($2^{-69}$ or $2^{-71}$. 0.5 points were deducted for having additional answers.

Q3.6 (3 points) Express, in hexadecimal, the smallest strictly positive representable number $k$ such that $k + 1$ is also representable in this floating-point representation.

**Solution:** Another way of reading this question is that $k+1$ must be the smallest representable number strictly greater than 1. We know that there are only eight mantissa bits, meaning that the smallest number greater than one is $1.00000001_2$. Subtracting 1 gives $k = 0.00000001_2 = 1 \times 2^{-8}$.

Adding the bias of 63 gives 55, or `0b0110111`.

`0b0 0110111 00000000`

`0b0011 0111 0000 0000`

`0x3700`

**Grading:** The sign, exponent, and mantissa bits were graded separately as all-or-nothing. A 1.5 point penalty was applied if the answer had incorrect number of bits.

i don't really understand what these two questions are asking.

I'm assuming Q3.5 is asking for the smallest step size in this system?

and for Q3.6, I don't understand the what they mean by "k+1 must be the smallest representable number strictly greater than 1"... and then we subtract 1 to get k? kind of confused.

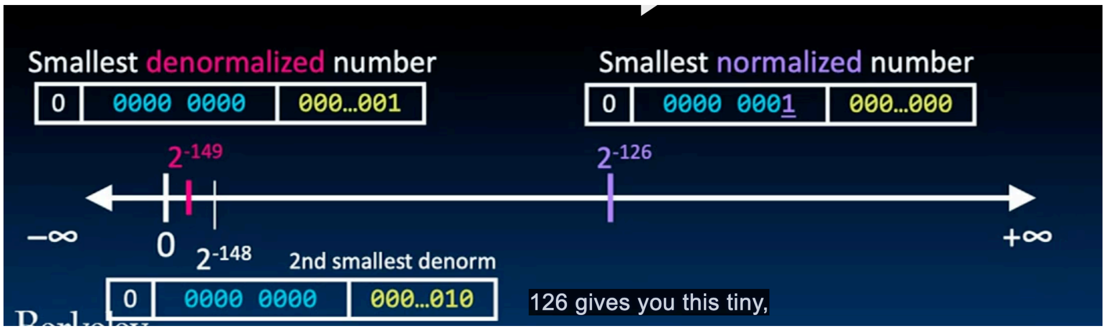
Andrew Liu STAFF 9mth #469df

Q3.5 is essentially asking for the step size (it is asking for a number $x$ such that $x$ + \text{step size associated with $x$} = $2x$)

Q3.6's solution hinges on the fact that in the pair $(x, x+1)$, $x+1$ should be as small as possible to get our desired answer.

Anonymous Koala 9mth #469deb

I am still confused on this question, like if I choose the smallest number of denorm as k, and I add 1 on to it, it will be definitley in range of this number line, since one is greater than the smallest normalized number, so shouldn't k be the smallest number of denormalized number?



Anonymous Snake 8mth #469ffc

Replying to Anonymous Koala

Same question on this^

Anonymous Squid 8mth #469aaca

Replying to Anonymous Koala

same^ not sure why the k wouldn't be the smallest denormalized number

Anonymous Alpaca 8mth #469adae

Replying to Anonymous Koala

Can you unsolve the question? I have the same question but no one answer. Thanks!

J

Justin Yokota STAFF 8mth #469adaf

Replying to Anonymous Alpaca

It'll be in range, but it won't be on a number that's exactly representable. There are infinitely many numbers in that range, but only finitely many actually correspond to a valid binary representation. As it turns out, the smallest positive number + 1 would end up in between two different representable numbers, and thus not be representable.

Annika Liu 9mth #469ca ✓ Resolved

[SU23-MT-Q2.9]

Hello, I have a question about how data are stored in memory for struct objects. In Question 2, I know that the struct type User, Book* is already initialized in a User struct because it has `borrowed_books[MAX_BORROWS]`. Is this stored on the stack or the heap? And so, why are we sure that this piece of memory will be saved?

Andrew Liu STAFF 9mth #469ea

What each `Book*` points to has not yet been allocated, but `borrowed_books[MAX_BORROWS]` is an array of `Book*` s of length `MAX_BORROWS`, so it is stored in the struct, and space for this array is created in the above `realloc` call.

Anonymous Quail 9mth #469be ✓ Resolved

FA23-MT-Q6.3

Wouldn't andy() only return true if all the bytes of the inputted value were 0? Since we're doing the bitwise and

Justin Yokota STAFF 9mth #469bf

Not always, as there are numbers x,y such that x != 0, y != 0, x&y==0. An example of this is 2 and 1; 2&1 = 0.

Annika Liu 9mth #469af ✓ Resolved

[SP23-MT-Q6.2]

Hello! I have a question about Q6.2. The solution says that the maximum hold time = the time from MUX to the register on the right, which has a 25ps delay. However, I don't understand a few points:

- since hold time <= clk-to-q delay + shortest combinatorial path delay, why didn't the solution add the 30ps clk-to-q delay?
- Also, isn't the shortest path between 2 timed registers the wire from the rightmost register to the top register where there is only a NOT gate? I used this to calculate and got 38ps as the maximum possible hold time.

Thank you!

Justin Yokota STAFF 9mth #469bc

Hold time constraints come from determining the first instance that an input to a register changes (after the clock tick). The equation you have is for paths that go from one register to another, which is why it includes a clk-to-q time.

However, in this circuit, there are paths that don't start at a register, but which change register inputs sooner than all register-to-register paths. The minimal one in this case is the path from the SEL input (which is said to update at exactly the rising edge) to the right register.

Annika Liu 9mth #469bd

Thank you! I understand.

Anonymous Echidna 9mth #469afd

Does this mean that the SCD path isn't just determined from register to register, it can start from a signal, such as the SEL signal? I put down 38 as I didn't realize that the path from SEL to the MUX counts as the max hold time.

Annika Liu 9mth #469ae ✓ Resolved

[SP23-MT-Q2.5 & Q2.9]

Hello! I want to double check the following:

- Q2.5: if we add a casting of (char *) before the malloc, does it also work? I'm thinking it should work because it is just casting a void * to a char *
- Q2.9: does `ch = &(sheet)` also work? This is logically the same as the solution `*ch = sheet`.

Thank you!

Justin Yokota STAFF 9mth #469ba

2.5: That should be fine if the types match, but there are style guides that prefer against this.

2.9: That would not work, as it is not logically the same. Your solution would change the local variable ch, and therefore never update the desired return pointer.

Annika Liu 9mth #469bb

I see I see. Thank you!

Anonymous Coyote 9mth #469ac ✓ Resolved

[Spring 2023 Q1.6] How to calculate the 588 immediate? I'm confused on how it is being calculated.

Justin Yokota STAFF 9mth #469ad

The main thing is that you need to convert 588 to its binary value, and use that as the immediate, putting the bits in the right spots.

Anonymous Nightingale 9mth #469e ✓ Resolved

FA23-MT-Q2

Are we expected to understand the vector struct? The question mentions working with vector_t in lab, but the one time we worked with vectors (lab 2), the struct given was different. I'm not familiar with the terms slice and children in this context.

Eric Che STAFF 9mth #469f

~~This is out of scope! You'll learn about vectors in the second half the class~~

Although the vector_t struct is different in this semester's lab 2, all of the information on the question about children and slices are explained in the question. If sometime like this were to show up, we will provide more information. However, you are still able to solve this question with the knowledge of this semester and explanations on the exam.

Eddy Byun STAFF 9mth #469aa Visible to staff only

This question is still in scope, right? These aren't SIMD vectors, and I believe we gave them an explanation on how children and slices work.

Eric Che STAFF 9mth #469ab Visible to staff only

Replying to Eddy Byun

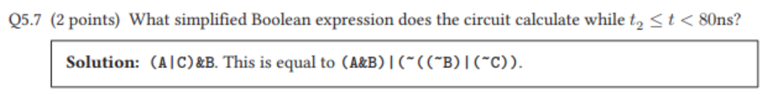
im rage

♡ ···

A Adrian Bao 9mth #469b ✓ Resolved

FA23-MT-Q5.7

How does (A&B) | (~((~B) | (~C)) equal (A | C) & B? Thanks!



Q5.7 (2 points) What simplified Boolean expression does the circuit calculate while $t_2 \leq t < 80\text{ns}$?

**Solution:** `(A|C)&B`. This is equal to `(A&B)|(~((~B)|(~C))`.

♡ ···

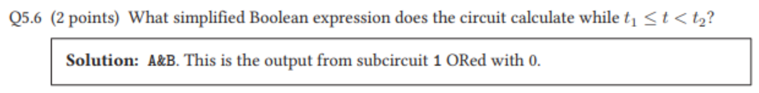
Andrew Liu STAFF 9mth #469c

$$\begin{aligned}(A\&B)|(\neg((\neg B)|(\neg C))) &= (A\&B)|(B\&C) && \text{De Morgan's}\\ &= (A\&B)|(C\&B) && \text{Comm. (AND)}\\ &= (A|C)\&B && \text{Dist. (AND over OR)}\end{aligned}$$

♡ 1 ···

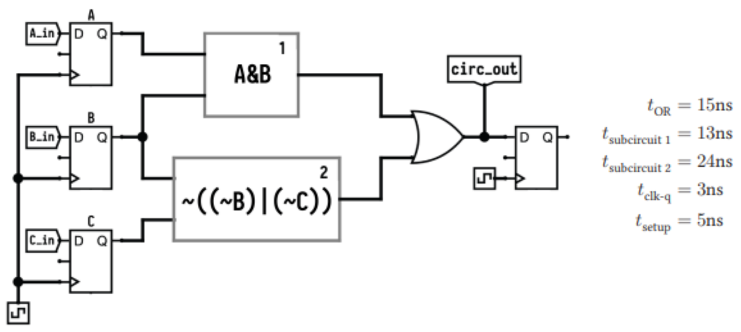
A Adrian Bao 9mth #469a ✓ Resolved

FA23-MT-Q5.6



Q5.6 (2 points) What simplified Boolean expression does the circuit calculate while $t_1 \leq t < t_2$?

**Solution:** `A&B`. This is the output from subcircuit `1` ORed with 0.



A_in
A
D Q
B_in
B
D Q
C_in
C
D Q
A&B
1
~((~B)|(~C))
2
circ_out
D Q
$t_{\text{OR}} = 15\text{ns}$
$t_{\text{subcircuit 1}} = 13\text{ns}$
$t_{\text{subcircuit 2}} = 24\text{ns}$
$t_{\text{clk-q}} = 3\text{ns}$
$t_{\text{setup}} = 5\text{ns}$

I thought the top would be 0 since 0&0 is 0, as A = B = 0, since the wires are initialized to 0. Then since t_1 <= t < t2, we are computing 0 | ~((~B) | (~C)), and since B = C = 0, and we are finding the negation of ((~B) | (~C)), which is 1, it is 0. Therefore, the top and bottom of the last OR gate is 0 | 0. Why is it A&B?

♡ ···

Andrew Liu STAFF 9mth #469d

The inputs are not given as 0. They may change at time $t = 0$, meaning we cannot make any assumptions about 1 or 0. The wires are indeed initialized to 0, hence the first answer was 0. However, the input may *change* at $t_1$ , meaning that by definition, we should expect it to not be identically zero, which happens after `A&B` and `1 | 2` are calculated, which means the answer is `(A&B) | 2 = (A&B) | 0` (since subcircuit 2 has not updated yet.)

♡ 1 ...

Anonymous Gnu 8mth #469abce

But won't the output of the registers only change when a clock period hits, and t1 < the clock period?

♡ ...