[Midterm] Past Exams - 2021 #882

J Jero Wang ADMIN
2 years ago in Exam – Midterm

1,726
VIEWS

1

You can find the past exams here: https://cs61c.org/sp23/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!

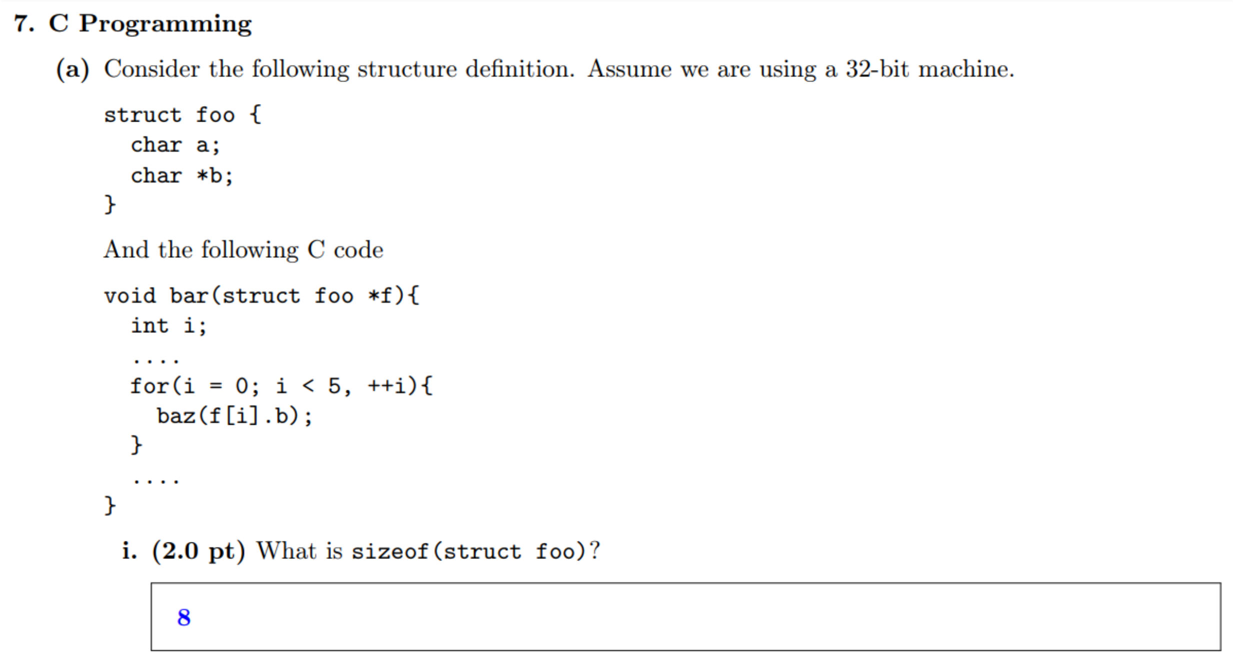
When posting questions, please reference the semester, exam, and question in this format so it's easier for students and staff to search for similar questions:

Semester-Exam-Question Number

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

Spring 2021 final walkthrough

Anonymous Otter 2y #882efc ✓ Resolved



7. C Programming

(a) Consider the following structure definition. Assume we are using a 32-bit machine.

```
struct foo {
  char a;
  char *b;
}
```

And the following C code

```
void bar(struct foo *f){
  int i;
  ....
  for(i = 0; i < 5, ++i){
    baz(f[i].b);
  }
  ....
}
```

i. (2.0 pt) What is `sizeof(struct foo)`?

8

SP-21-7ai,

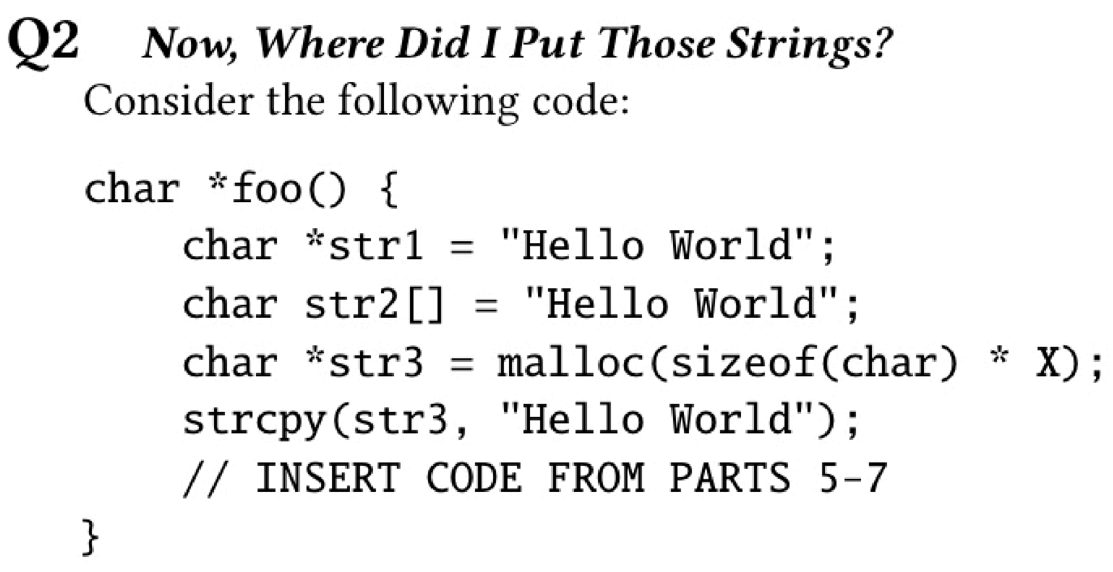
Isn't char 1 byte, and pointer to char 4 bytes, shouldn't it be 5?

E Erik Yang TA 2y #882efe

#882cb

W Wending Zhao 2y #882eec ✓ Resolved

just curious, is str1 stored in static and &str1 is stored in stack?



Q2 *Now, Where Did I Put Those Strings?*

Consider the following code:

```
char *foo() {
    char *str1 = "Hello World";
    char str2[] = "Hello World";
    char *str3 = malloc(sizeof(char) * X);
    strcpy(str3, "Hello World");
    // INSERT CODE FROM PARTS 5-7
}
```

Erik Yang TA 2y #882eef

yeah

Anonymous Opossum 2y #882edf ✓ Resolved

SP21-MT-Q3

what is logic component delay?

Rosalie Fang ADMIN 2y #882eeb

An AND gate, for example, is a logic component, and has a delay of 7ns in this question. The "Black Box" contains different logical components and has a delay of 9ns.

1

Anonymous Hamster 2y #882ecd ✓ Resolved

SP21-MT_Q7

7.a.i, why sizeof(struct foo) == 8?

Could anyone explain what q7.b does? Don't understand the relation between data and size.

Thanks!

Jero Wang ADMIN 2y #882ece

#882cb

Anonymous Hamster 2y #882edc

Could you explain what q7.b does? Thanks!

Erik Yang TA 2y #882edd

Replying to Anonymous Hamster

Size is how much the data can store in bits

Anonymous Cat 2y #882ecc ✓ Resolved

FA21-MT-Q2.8

If we didn't strcpy anything into str3, can we still print str3?

J Jero Wang ADMIN 2y #882ecf

No, because it might contain garbage since you're using malloc.

Anonymous Gnu 2y #882ecb ✓ Resolved

FA21-MT-2.9

Why does ((uint32_t*) str1)[2] evaluate to 0x00646C72? Wouldn't this value be the way it's stored in memory and the actual value is the hex value of rld\0? Or is the actual value returned to be 0x00646C72?

J Jero Wang ADMIN 2y #882eda

The memory is little endian, meaning the byte at the lowest address is at the least significant bits. Here, the lowest address contains `r`, which is stored in the least significant bits of the `uint32_t`. I would recommend drawing out which byte is stored at which address.

Anonymous Sand Dollar 2y #882ebf ✓ Resolved

SP21-MT1-Q8bi

I still don't understand how 2^-16 was derived. I looked at the posts below and the piazza forum links.... What does step size mean? The amount between the largest number possible to represent and the next number?

1

M Melanie McKune 2y #882eca

Lwk like maybe it has to do with the closest you can represent 80, ie 2^6?

J Jero Wang ADMIN 2y #882edb

The exponent for representing 80 in this FP system would be 2^6. If the exponent was 2^6, and we have 22 mantissa bits, incrementing the least significant bit of the mantissa would add 2^{-16} to the value, this is also known as the step size.

D Deniz Demirtas 2y #882ebd ✓ Resolved

SP21-MT2-Q8a

I am totally lost in terms of finding the how many numbers a certain floating point representation can represent. I tried watching the guide videos for homework and the slides, but I am still confused. I would really appreciate some help on this. Thanks!

E Erik Yang TA 2y #882ebe

#882fa #882ad

Anonymous Bison 2y #882dfc ✓ Resolved

FA21-MT-Q5

In Get20chars, the input says a0 is a pointer to a buffer. Does "buffer" always means the location of the stack pointer in general? Thanks!

E Erik Yang TA 2y #882dfd

buffer just means it's a position in memory that will store information. It could be an array that holds 20 bytes of info that you can use to store things in.

Anonymous Chicken 2y #882dfb ✓ Resolved

SP21-MID-Q2.b, why can translator produce interpreted code? On the course slides, says

- Interpreter: Directly executes a program in the source language
- Translator: Converts a program from the source language to an equivalent program in another language

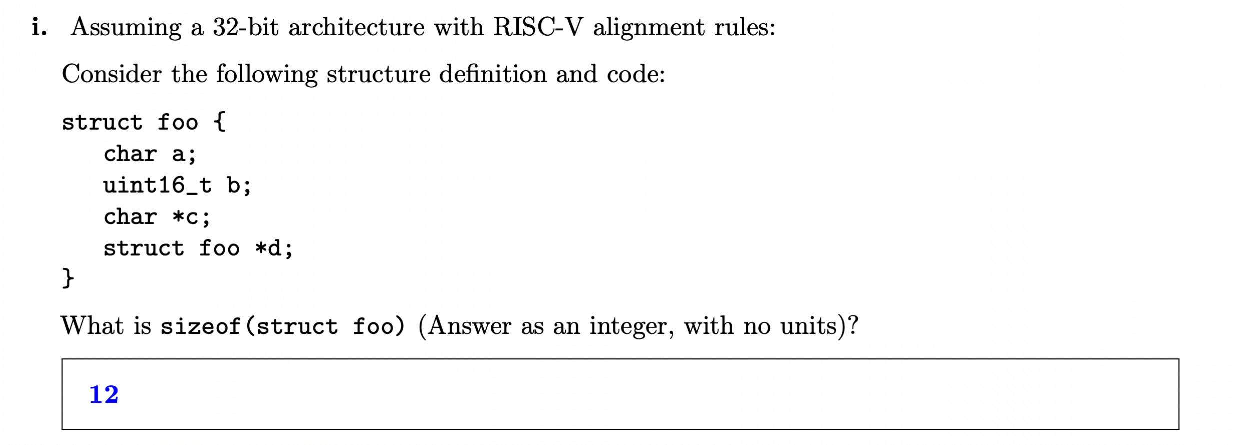
I think they are different.

R Rosalie Fang ADMIN 2y #882eaa

Translators convert a program from the source language to another language, if the destination language is an interpreted language, the translator will produce interpreted code.

Anonymous Rail 2y #882ded ✓ Resolved

SP21-Final-Q6A


i. Assuming a 32-bit architecture with RISC-V alignment rules:

Consider the following structure definition and code:

```
struct foo {
   char a;
   uint16_t b;
   char *c;
   struct foo *d;
}
```

What is `sizeof(struct foo)` (Answer as an integer, with no units)?

12

For the struct foo, I understand how char a is padded to 4 bytes, uint16_t b is 4 bytes, char* c is also 4 bytes. So why do we ignore struct foo *d?

E

Erik Yang TA 2y #882dee

A is padded to 2 bytes bc b is 16 bits or 2 bytes. Then, c and d are both pointers so it's 2 + 2 + 4 + 4 = 12

1

Anonymous Rail 2y #882def

Thank you! If b was an integer in a 32 bit operating system, would the sizeof(struct foo) = 16 then?

E

Erik Yang TA 2y #882dfa

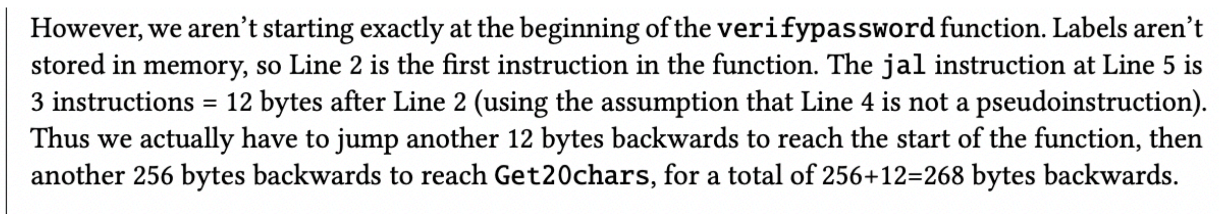
Replying to Anonymous Rail

Yup the a would be padded to 4 bytes

Anonymous Hawk 2y #882dec ✓ Resolved

Fall 21 Midterm

For question 5.3: it says we have to jump 12 bytes. But shouldn't it be 16. The start of verify password is line 1, not line 2 right? When we jump back to the function, we consider the offset as the line of the function label, not its first line of code.

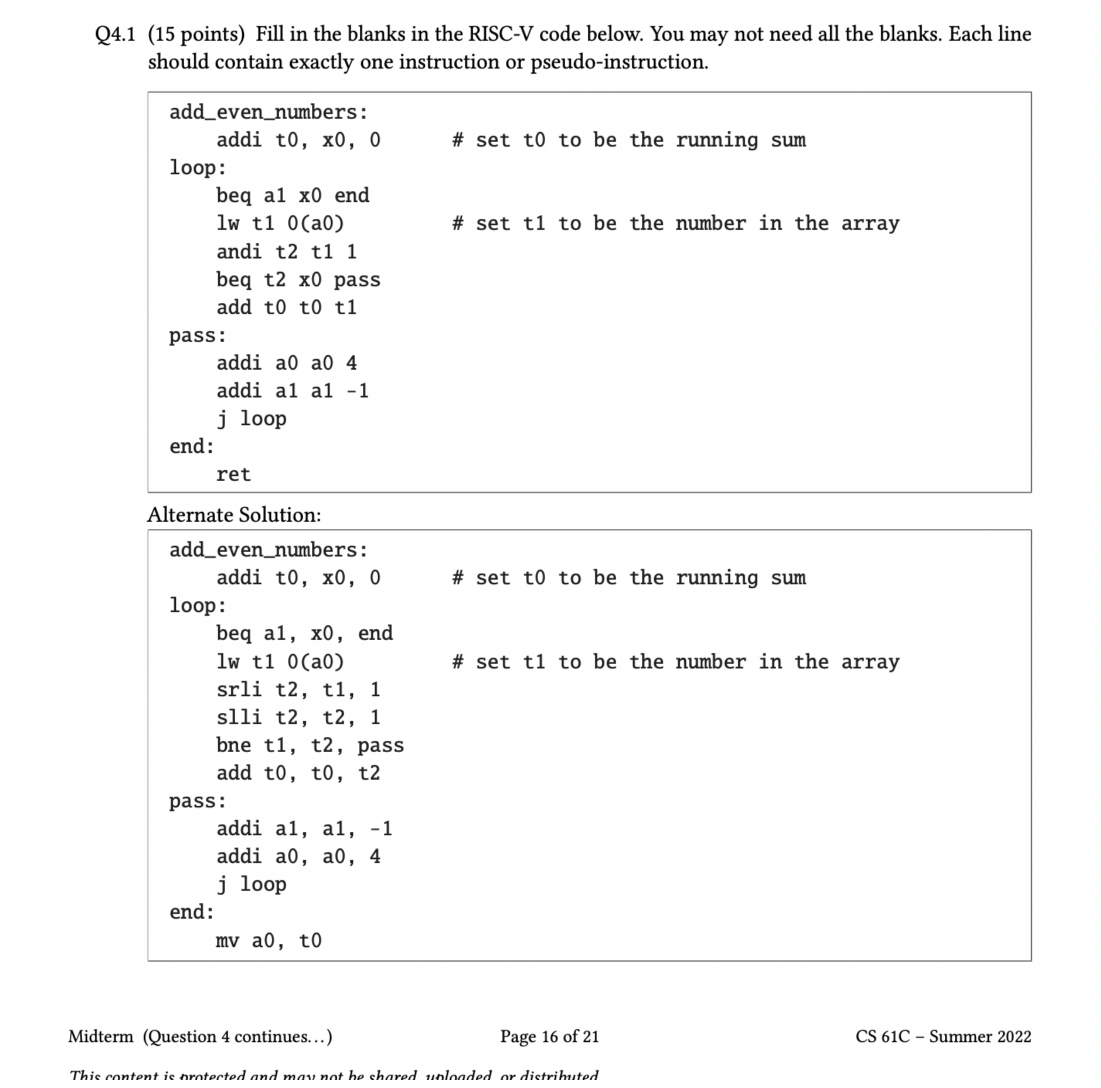

However, we aren't starting exactly at the beginning of the `verifypassword` function. Labels aren't stored in memory, so Line 2 is the first instruction in the function. The `jal` instruction at Line 5 is 3 instructions = 12 bytes after Line 2 (using the assumption that Line 4 is not a pseudoinstruction). Thus we actually have to jump another 12 bytes backwards to reach the start of the function, then another 256 bytes backwards to reach `Get20chars`, for a total of 256+12=268 bytes backwards.

E

Erik Yang TA 2y #882eac

labels don't count as code, so the first line is actually line 2 where the function starts

Anonymous Hawk 2y #882ead

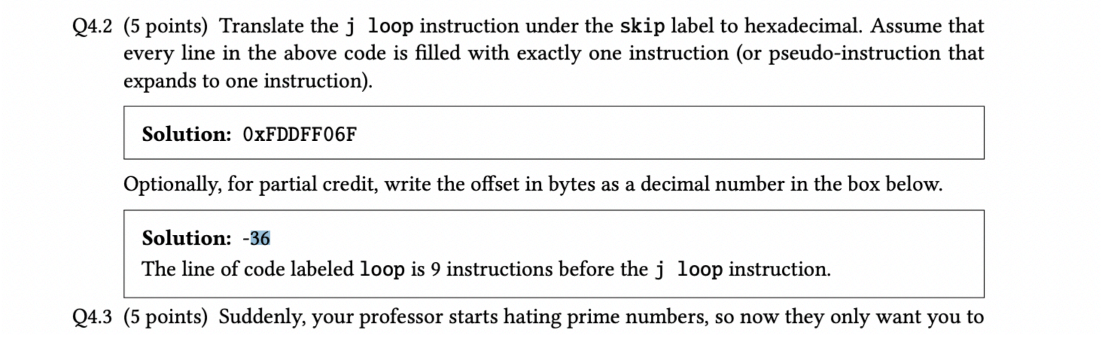


Q4.1 (15 points) Fill in the blanks in the RISC-V code below. You may not need all the blanks. Each line should contain exactly one instruction or pseudo-instruction.

```
add_even_numbers:
    addi t0, x0, 0      # set t0 to be the running sum
loop:
    beq a1 x0 end
    lw t1 0(a0)         # set t1 to be the number in the array
    andi t2 t1 1
    beq t2 x0 pass
    add t0 t0 t1
pass:
    addi a0 a0 4
    addi a1 a1 -1
    j loop
end:
    ret
```

Alternate Solution:

```
add_even_numbers:
    addi t0, x0, 0      # set t0 to be the running sum
loop:
    beq a1, x0, end
    lw t1 0(a0)         # set t1 to be the number in the array
    srli t2, t1, 1
    slli t2, t2, 1
    bne t1, t2, pass
    add t0, t0, t2
pass:
    addi a1, a1, -1
    addi a0, a0, 4
    j loop
end:
    mv a0, t0
```



Q4.2 (5 points) Translate the `j loop` instruction under the **skip** label to hexadecimal. Assume that every line in the above code is filled with exactly one instruction (or pseudo-instruction that expands to one instruction).

Solution: `0xFDDFF06F`

Optionally, for partial credit, write the offset in bytes as a decimal number in the box below.

Solution: -36
The line of code labeled `loop` is 9 instructions before the `j loop` instruction.

Q4.3 (5 points) Suddenly, your professor starts hating prime numbers, so now they only want you to

For 4.2 on Summer 22, it looks like they do consider labels as code. If we didn't consider the labels loop and pass, then the number of lines would be 7. Where did they get 9 from?

E

Erik Yang TA 2y #882eae

Replying to Anonymous Hawk

oh i do see the difference, thanks for bringing this up. I feel like this is a valid clarification that could be brought up during the exam

Anonymous Hawk 2y #882eaf

Replying to Erik Yang

ok thanks. So for the exam tomorrow, is it safe to say I can do the latter method if no clarification is provided?

E Erik Yang TA 2y #882eba

Replying to Anonymous Hawk

let me double check with you just to be safe

E Erik Yang TA 2y #882ebb

Replying to Erik Yang

would assume the fa21 version if no clarification

1

E Erik Yang TA 2y #882ebc

Replying to Erik Yang

also, the instructions say to assume that every line in above code is filled with exactly one instruction so it actually should be 9

Anonymous Deer 2y #882dea ✓ Resolved

Sp21-MT-Q8bii

What are the steps involved in getting the answers to parts A and B? I am getting. different answers and want to see where I am going wrong

E Erik Yang TA 2y #882deb

For A #882bda

1

R Rosalie Fang ADMIN 2y #882eab

For B:

0xC07C0000 => 0b 1100 0000 0111 1100 0000 0000 0000 0000 ;

exponent: 100 0000 01 = 1 + 2^8 = 257

mantissa: 11 1100 000... 0000

sign: 1

Number = - 2^(257 + bias) * 1.mantissa = - 2^(257 - 255) * 1.1111 = - 2^2 * 1.1111 = 111.11 = 7.75

Anonymous Rhinoceros 2y #882faa

how do we know what the bias is for the exponent? is it always:

2 ^ (number of exp bits - 1) - 1: so 2^8 - 1?

Anonymous Deer 2y #882ddb ✓ Resolved

FA21-MT-Q4.5

Why is the answer 2^-70 and not 2^-71? The smallest mantissa is 2^-8 multiplied by bias 2^-63 should give 2^-71 right?

E Erik Yang TA 2y #882ddc

#882db

Anonymous Deer 2y #882ddd

Oh thanks! So for denorm numbers, the absolute value of the bias always decreases by 1?

E Erik Yang TA 2y #882dde

Replying to Anonymous Deer

it's exp + bias + 1

Anonymous Jaguar 2y #882dcd ✓ Resolved

FA21-MT-Q5.3

What does it mean when labels are not stored in memory? Does that mean that if verifypassword is located at 0x00001000, then line 2 (addi sp, sp, -24) is also located at 0x00001000?

N Nikhil Kandkur TA 2y #882dce

It means that you cannot consider a label its own instruction. For example,

```
label:
        addi sp sp -24
```

counts only as one instruction instead of 2. AKA it should be interpreted as

```
label: addi sp sp -24
```

1

Anonymous Jaguar 2y #882dcf

Would the address in memory for line 2 be 0x00001000 then?

N Nikhil Kandkur TA 2y #882dda

Replying to Anonymous Jaguar

Yup!

2

Anonymous Hedgehog 2y #882dbd ✓ Resolved

FA21-MT-Q4.3: The solution suggests "only difference in representable numbers comes from NaNs.", but I'm confused why that's true. Why don't the infinities and the positive/negative zero count as duplicate numbers?

1

E Erik Yang TA 2y #882dca

Well since both representations have 2^16 bit reps that means that fp will have less number representations because of the Nans. Even though fp can represent inf, the unsigned can represent integers not in fp as well. These numbers are accounted for since fp can represent 2^16 - NaN different numbers which includes inf, but unsigned can represent 2^16 different numbers

Anonymous Flamingo 2y #882efa

unsigned cant represent -0? but fp can? do we account for that?

E Erik Yang TA 2y #882efd

Replying to Anonymous Flamingo

Fp can represent 2^16 - NaN numbers and unsigned can represent 2^16 num but what each version represents could be different numbers. Ex: fp could represent 0 and -0 while unsigned could represent 1014 or smthng like that. They each can vary on what they represent but the amount of numbers that they represent could be the same

1

Anonymous Alpaca 2y #882dba ✓ Resolved

SP21-MT-Q8b

How is the largest step size calculated in part i?

1

E Erik Yang TA 2y #882dbc

#882cbf

Anonymous Alpaca 2y #882dbf

I don't think Yile is able to see my reply because I can't unresolve the original question, but by "weight of the LSB in the significand", does he mean the number of mantissa bits?

E Erik Yang TA 2y #882dcb

Replying to Anonymous Alpaca

I'm not sure what he means by weight of LSB, but he'll see ur reply
Also #882bbb might help

Anonymous Hedgehog 2y #882dab ✓ Resolved

FA21-MT-Q3.1

Is `union` in scope for this semester? I don't recall learning about it, but if it is in scope a link to where it was discussed this semester would be greatly appreciated!

E Erik Yang TA 2y #882dac

in scope

Anonymous Hedgehog 2y #882dad

thanks - where was it discussed?

Erik Yang TA 2y #882daf

Replying to Anonymous Hedgehog

it was introduced in lecture 5
https://drive.google.com/file/d/1MGOqAsFPYATMusoKxBgg3D_o_oVHhAJ-/view

slide 39

Anonymous Deer 2y #882cff ✓ Resolved

FA21-MT-Q2.8

I am confused as to how str3 is a valid null-terminated string since it just has "Hello World" in it without the '\0'

Yile Hu TUTOR 2y #882daa

All string literals have an implicit null terminator

Anonymous Deer 2y #882cfc ✓ Resolved

SP21-MT-Q5

Is this question in scope?

Catherine Van Keuren TA 2y #882cfe

Not in scope - the data path is out of scope for the midterm.

Anonymous Red deer 2y #882cfb ✓ Resolved

SP21-Q2a

Could somebody explain the reasoning behind the solution for this problem?

I understand that compiled code is only able to run on one ISA generally --> but im not sure I understand the rest of the options. here is my though process:

- "Compilers produce larger code than interpreters but do it faster"
  - Is this because compilers will produce machine code that will be longer because instructions are broken down, but it will be executed faster?
- "The code produced is always more efficiency and higher performance than that produced by interpreters"
  - Is this because if I compiled something like machine code, there would be no point?
- "There is only one compiler per language"
  - We can construct our own compilers
- "Compilers are always more difficult to write than interpreters"
  - Isn't this true? I can't find a counter example

- "The easiest step of CALL is compilation; the harder parts are assembling, linking, and loading"
    - What would be the easiest part?

I was hoping my thought process would be verified and my questions could be resolved. Thank you in advance!

E Erik Yang TA 2y #882dbb

1. compilers take more time to compile your code because (ex: it checks your code for compiler issues), but it makes your program run faster as a result

2. Be careful about the word always, there's times where compiled code does not run more efficiently

3. Yup

4. Again, be careful about always. It's generally more difficult, but there are instances where interpreters would be a little more difficult.

5. it's hard to say what the easiest step of CALL is, but i'd argue it's the linker or assembler

1

Anonymous Red deer 2y #882dfe

Can you elaborate a little on #5? Do we have a defined hierarchy for the easiest/hardest step? This wasn't covered in lecture but this concept keeps on showing up on exams, so any clarification would be appreciated!

E Erik Yang TA 2y #882dff

Replying to Anonymous Red deer

I'm actually not sure of a hierarchy but it could maybe do with which one has the most steps. If anyone else has an idea, it would be nice to know too

Anonymous Alpaca 2y #882cef ✓ Resolved

FA21-Q5

Why do we do "la t0 Password"? Doesn't this load &Password, which is the address of the label? Since Password is a pointer, I feel like we should be loading in the value at Password, not the address of the label itself.

E Eric Kusnanto TA 2y #882cfa

You're correct, t0 holds &Password, a pointer to a char array. We load in the values of Password as we iterate through the array on Line 9

Anonymous Alpaca 2y #882cfd

If &Password is the pointer to the char array, what is Password itself?

E Erik Yang TA 2y #882dae

Replying to Anonymous Alpaca

Password is just a label that stores an address to a pointer to char array

Anonymous Jaguar 2y #882cea ✓ Resolved

FA21-MT-Q2.8

Does printf always dereference string pointers passed into the function?

E Erik Yang TA 2y #882ced

For %s printed, found on the man pages: The const char * argument is expected to be a pointer to an array of character type

1

Anonymous Jaguar 2y #882cdf ✓ Resolved

FA21-MT-Q2.5

Returning str3 returns a pointer to the allocated memory in the heap, right? Is returning the pointer the same thing as returning the string?

E Erik Yang TA 2y #882cec

Yes it is the pointer to the heap, not really because the function wants us to return a pointer so you can't jsut return the literal string

Anonymous Jaguar 2y #882cee

The question said "returning the string," so I was unsure whether it was asking for the literal string, or a pointer. Are we supposed to assume that we're returning a pointer?

Anonymous Rook 2y #882cce ✓ Resolved

SP21-MT-Q7

Why is sizeof(struct foo) = 8 bytes? Is it not 4 bytes for pointer and 1 byte for char?

J Jero Wang ADMIN 2y #882cda

Padding adds an additional 3 bytes so it is 8 bytes total.

Anonymous Rook 2y #882cde

is padding for all structs? Do all structs pad 3 bytes?

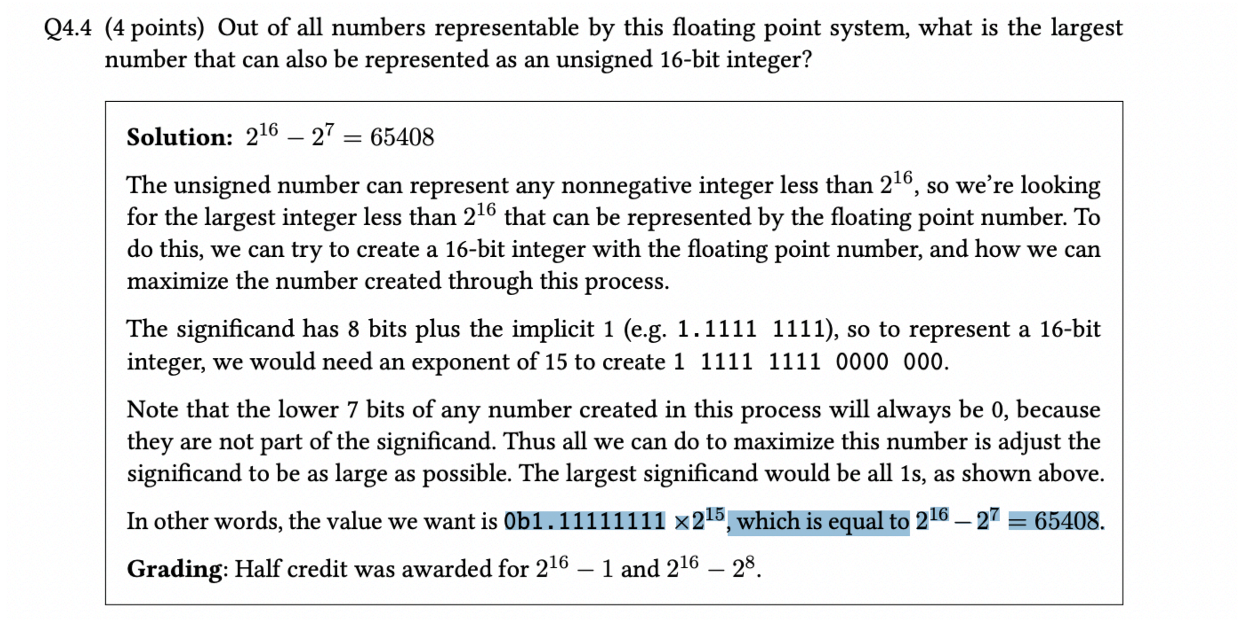
E Erik Yang TA 2y #882ceb

Replying to Anonymous Rook

No padding is unique to each struct #882aaa

Anonymous Rail 2y #882ccd ✓ Resolved

FA21-MT-Q4.4



Q4.4 (4 points) Out of all numbers representable by this floating point system, what is the largest number that can also be represented as an unsigned 16-bit integer?

Solution: $2^{16} - 2^7 = 65408$

The unsigned number can represent any nonnegative integer less than 2^{16}, so we're looking for the largest integer less than 2^{16} that can be represented by the floating point number. To do this, we can try to create a 16-bit integer with the floating point number, and how we can maximize the number created through this process.

The significand has 8 bits plus the implicit 1 (e.g. `1.1111 1111`), so to represent a 16-bit integer, we would need an exponent of 15 to create `1 1111 1111 0000 000`.

Note that the lower 7 bits of any number created in this process will always be 0, because they are not part of the significand. Thus all we can do to maximize this number is adjust the significand to be as large as possible. The largest significand would be all 1s, as shown above.

In other words, the value we want is `0b1.11111111` $\times 2^{15}$, which is equal to $2^{16} - 2^7 = 65408$.

Grading: Half credit was awarded for $2^{16} - 1$ and $2^{16} - 2^8$.

I'm following along until the end. How do we go from the 0b1.11111111 x 2^15 to 2^16 - 2^7?

Jero Wang ADMIN 2y #882cdd

$$0b1.11111111 \cdot 2^{15} = 2^{15} + 2^{14} + 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^7 = 2^{16} - 2^7$$

Alternatively, you can add $2^7 = 0b0.00000001 \cdot 2^{15}$ to $0b1.11111111 \cdot 2^{15}$ and get $10.00000000 \cdot 2^{15} = 2^{16}$, then rearrange that equation.

1

Anonymous Toad 2y #882ccc ✓ Resolved

SP21-MT-7bi

Shouldn't calloc have 2 parameters? Or is the first first one optional?

Sam Xu TUTOR 2y #882ccf

calloc should have two arguments.

Anonymous Chimpanzee 2y #882dcc

The answer is calloc(size/8 + 1) though :o

Anonymous Porcupine 2y #882cca ✓ Resolved

Sp21-MT-Q4:

Hi, I am wondering about this question instead of storing half and then storing a byte. Can I use 'sh' three times?

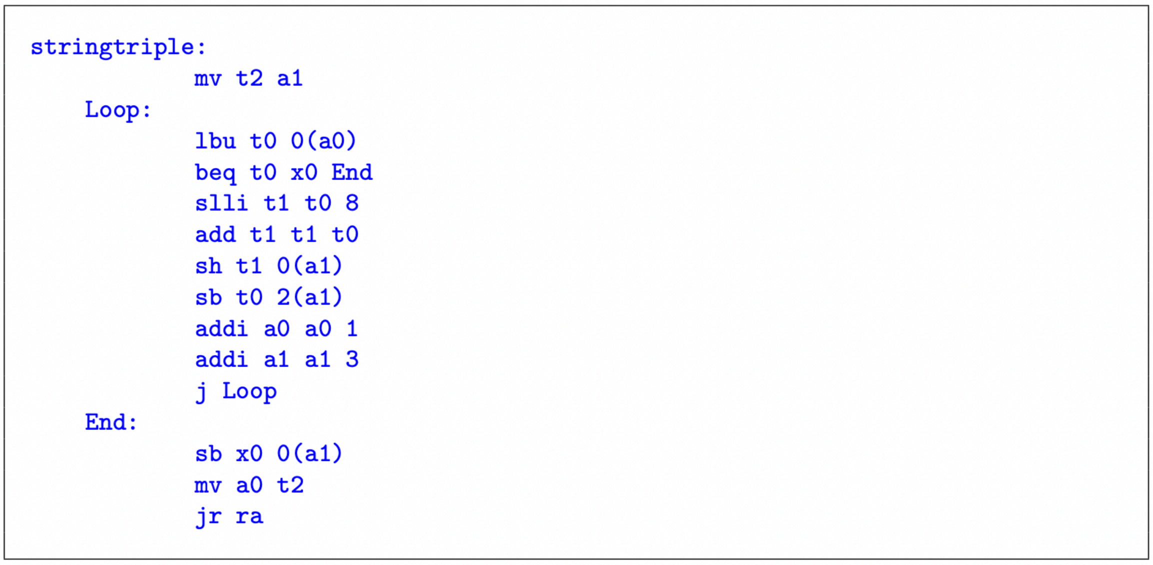
sb t0 0(a1)

sb t0 1(a1)

sb t0 2(a1)

Others stay the same.

```
stringtriple:
```



```
stringtriple:
            mv t2 a1
    Loop:
            lbu t0 0(a0)
            beq t0 x0 End
            slli t1 t0 8
            add t1 t1 t0
            sh t1 0(a1)
            sb t0 2(a1)
            addi a0 a0 1
            addi a1 a1 3
            j Loop
    End:
            sb x0 0(a1)
            mv a0 t2
            jr ra
```

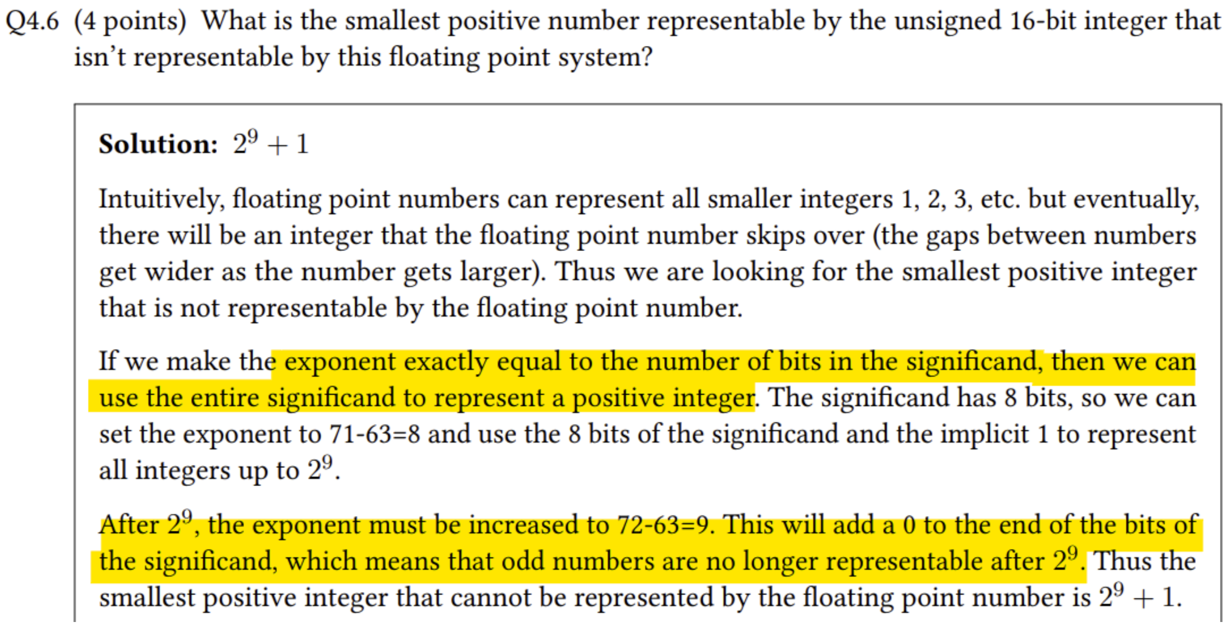
Jero Wang ADMIN 2y #882cdb

I think that would work here as well.

2

Anonymous Parrot 2y #882cbc ✓ Resolved

fa21-mt-q4.6 -- I'm having trouble understanding why these two highlighted lines are so. Could anyone explain this? Thanks!



Q4.6 (4 points) What is the smallest positive number representable by the unsigned 16-bit integer that isn't representable by this floating point system?

Solution: $2^9 + 1$

Intuitively, floating point numbers can represent all smaller integers 1, 2, 3, etc. but eventually, there will be an integer that the floating point number skips over (the gaps between numbers get wider as the number gets larger). Thus we are looking for the smallest positive integer that is not representable by the floating point number.

If we make the exponent exactly equal to the number of bits in the significand, then we can use the entire significand to represent a positive integer. The significand has 8 bits, so we can set the exponent to 71-63=8 and use the 8 bits of the significand and the implicit 1 to represent all integers up to 2^9.

After 2^9, the exponent must be increased to 72-63=9. This will add a 0 to the end of the bits of the significand, which means that odd numbers are no longer representable after 2^9. Thus the smallest positive integer that cannot be represented by the floating point number is $2^9 + 1$.

J

Jero Wang ADMIN 2y #882cbe

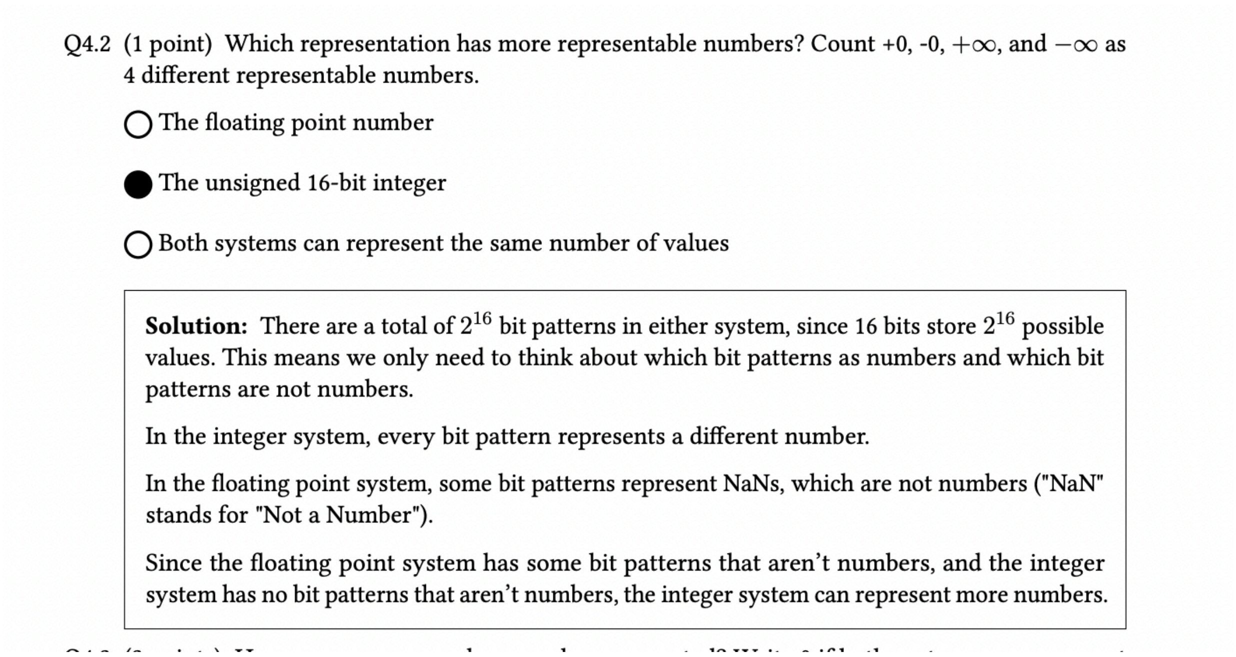
First line: Using their example of 8 bit mantissa, if the exponent is 2^8, then the smallest step we can have with a change in the mantissa is 1, since the smallest step would be $0.00000001 \cdot 2^8 = 1$.

Second line: As mentioned above, the smallest step for 2^8 is 1, so if we have 2^9, the smallest step we have is 2, which means that we would skip every other number (skip every odd number).

Anonymous Hawk 2y #882cae ✓ Resolved

Fall 21 4.2:

Is it true that for all floating point systems, the number of representable numbers is 2^number of bits. What is the range of numbers we can represent with the floating point?



Q4.2 (1 point) Which representation has more representable numbers? Count +0, -0, $+\infty$, and $-\infty$ as 4 different representable numbers.

- ○ The floating point number
- [✓ The unsigned 16-bit integer]
- ○ Both systems can represent the same number of values

Solution: There are a total of 2^{16} bit patterns in either system, since 16 bits store 2^{16} possible values. This means we only need to think about which bit patterns as numbers and which bit patterns are not numbers.

In the integer system, every bit pattern represents a different number.

In the floating point system, some bit patterns represent NaNs, which are not numbers ("NaN" stands for "Not a Number").

Since the floating point system has some bit patterns that aren't numbers, and the integer system has no bit patterns that aren't numbers, the integer system can represent more numbers.

E

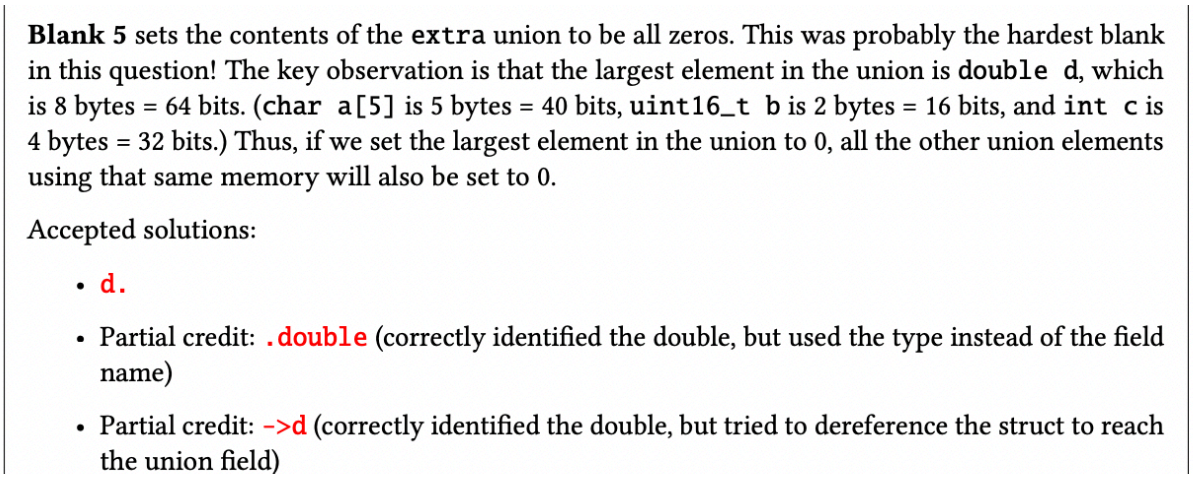
Erik Yang TA 2y #882cba

Yes. The range for fp is from neg infinity to pos infinity technically, but you can't represent all the numbers in this range

Anonymous Hawk 2y #882cad ✓ Resolved

Fall 21 3

Why do we assume all other elements of th union use the same memory. Is this the case for all unions and structs? Intuitively, I thought i c that different variables are allocated in different memory.



Blank 5 sets the contents of the `extra` union to be all zeros. This was probably the hardest blank in this question! The key observation is that the largest element in the union is `double d`, which is 8 bytes = 64 bits. (`char a[5]` is 5 bytes = 40 bits, `uint16_t b` is 2 bytes = 16 bits, and `int c` is 4 bytes = 32 bits.) Thus, if we set the largest element in the union to 0, all the other union elements using that same memory will also be set to 0.

Accepted solutions:

- `d.`
- Partial credit: `.double` (correctly identified the double, but used the type instead of the field name)
- Partial credit: `->d` (correctly identified the double, but tried to dereference the struct to reach the union field)

E

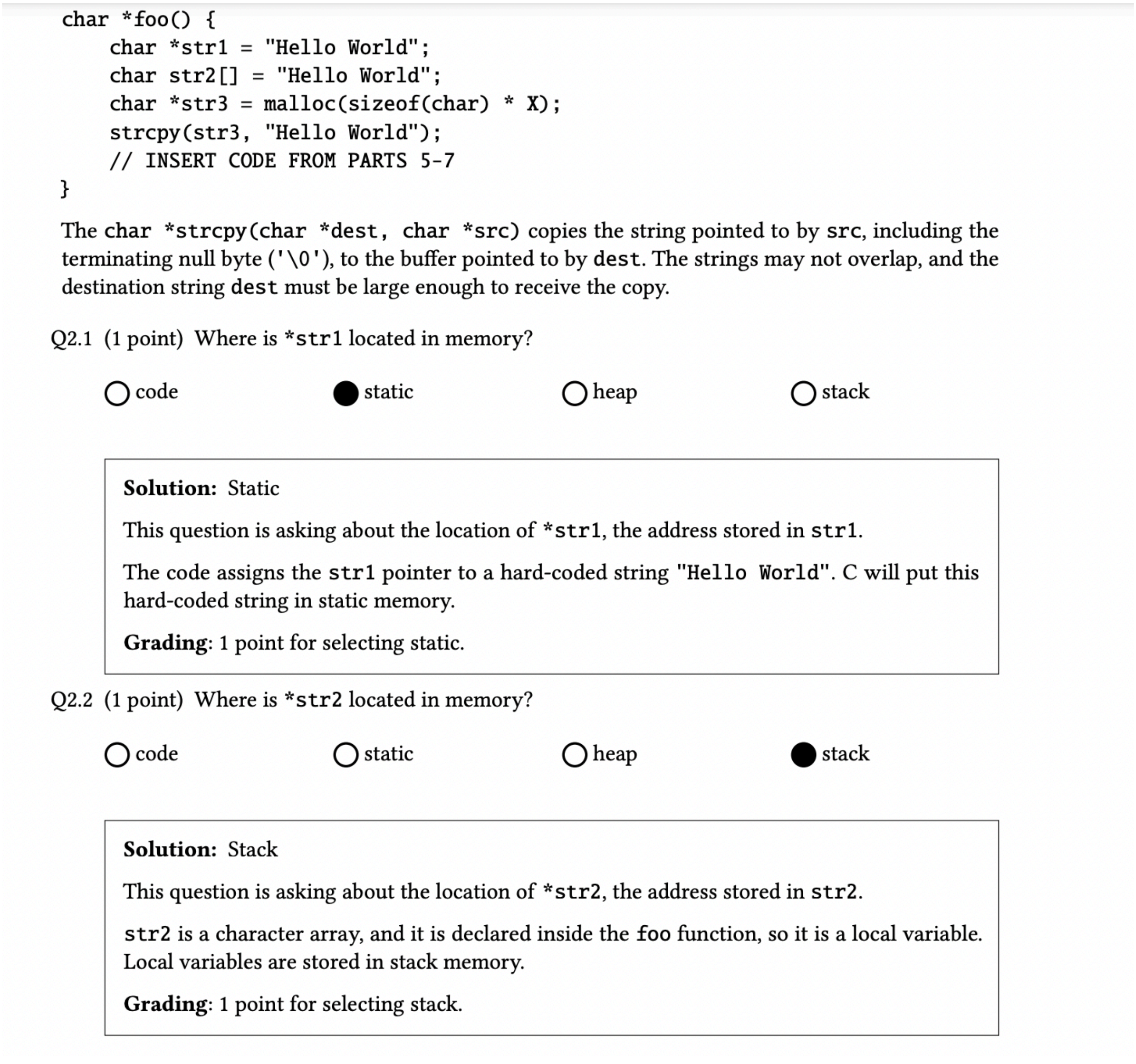
Erik Yang TA 2y #882caf

Union is different because it allocates memory for the biggest element in the union and all elements just share that piece of memory

Anonymous Hawk 2y #882caa ✓ Resolved

Fal; 2021 -

Both arrays and pointers are regarded the same. Do why are 2.1 and 2.2 different answers?



```
char *foo() {
    char *str1 = "Hello World";
    char str2[] = "Hello World";
    char *str3 = malloc(sizeof(char) * X);
    strcpy(str3, "Hello World");
    // INSERT CODE FROM PARTS 5-7
}
```

The `char *strcpy(char *dest, char *src)` copies the string pointed to by `src`, including the terminating null byte (`'\0'`), to the buffer pointed to by `dest`. The strings may not overlap, and the destination string `dest` must be large enough to receive the copy.

Q2.1 (1 point) Where is `*str1` located in memory?

code [✓ static] heap stack

Solution: Static

This question is asking about the location of `*str1`, the address stored in `str1`.

The code assigns the `str1` pointer to a hard-coded string `"Hello World"`. C will put this hard-coded string in static memory.

Grading: 1 point for selecting static.

Q2.2 (1 point) Where is `*str2` located in memory?

code static heap [✓ stack]

Solution: Stack

This question is asking about the location of `*str2`, the address stored in `str2`.

`str2` is a character array, and it is declared inside the `foo` function, so it is a local variable. Local variables are stored in stack memory.

Grading: 1 point for selecting stack.

Anonymous Hawk 2y #882cab

In general, are strings and arrays always considered static in c?

E

Erik Yang TA 2y #882cac

good question, str1 is a string literal, and str2 is a char array. When you have a char array, it puts all the letters on the stack, while the string literal goes into static. These two are different in terms of where the string gets put in memory. However, the pointer itself for both str1 and str2 lie in stack since it is a local variable inside the func foo.

1

Anonymous Nightingale 2y #882bee ✓ Resolved

SP21-MT-7bi: how do we know to use calloc instead of malloc?

E

Erik Yang TA 2y #882bff

Here we use calloc, because we try to access its elements without initializing them. Calloc will create an array with all 0s so when you try to do line C, you won't run into a memory issue. If you malloc, you won't know what bf -> data [n/8] is because it doesn't exist yet, but in calloc it is a 0

1

Anonymous Nightingale 2y #882bed ✓ Resolved

SP21-MT-7aiii: why are we doing slli s6 s6 3 and not 2? doesn't doing slli with immediate x lead us to multiply by 2^x? And we want to multiply our offset by 4, so I thought it would be:

```
slli s6 s6 2 # compute offset: i = i*4
addi a0 s6 s5 # add offset to function pointer s5
lw a0 0(a0)    # load word from offsetted location in memory
```

E

Erik Yang TA 2y #882bfd

#882aaf

1

Anonymous Nightingale 2y #882bec ✓ Resolved

SP21-MT-7ai: why is this 8 bytes and not 5? Aren't characters 1 byte?

E

Erik Yang TA 2y #882bfc

Need to account for padding #882aaa

1

Anonymous Nightingale 2y #882beb ✓ Resolved

in SP21-MT-4a: why do we use lbu and not lb? how do we know when to use which?

E Erik Yang TA 2y #882bfb

Since letters are always unsigned then you use lbu, use lb for numbers

Anonymous Frog 2y #882bea ✓ Resolved

SP21-MT-Q7

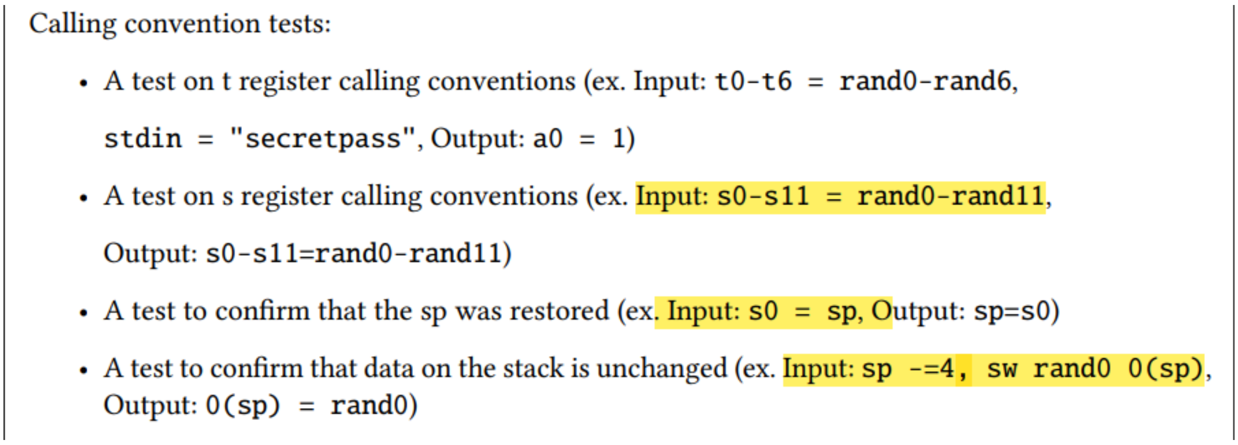
For aiii, why do we left shift s6 by 3? If i is an integer, wouldn't we left shift s6 by 2?

E Erik Yang TA 2y #882bfa

#882aaf

Anonymous Gull 2y #882bdf ✓ Resolved

FA21-Midterm-Q6



Calling convention tests:

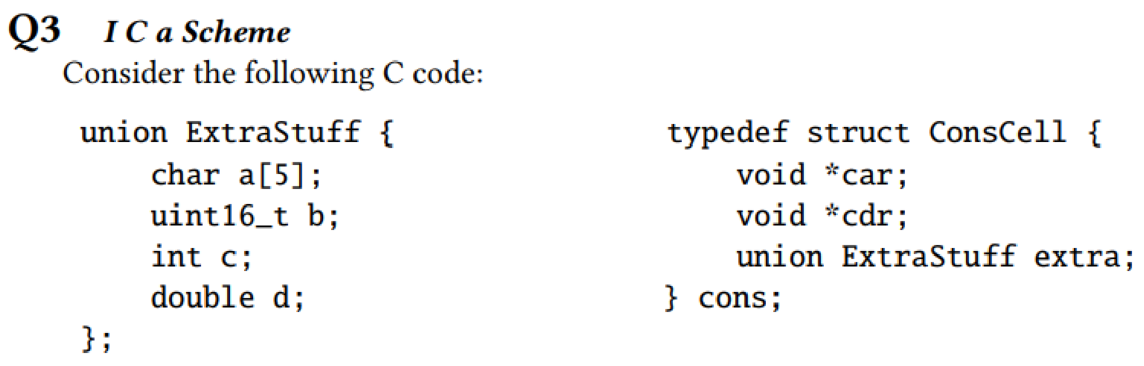
- A test on t register calling conventions (ex. Input: `t0-t6 = rand0-rand6`, `stdin = "secretpass"`, Output: `a0 = 1`)
- A test on s register calling conventions (ex. Input: `s0-s11 = rand0-rand11`, Output: `s0-s11=rand0-rand11`)
- A test to confirm that the sp was restored (ex. Input: `s0 = sp`, Output: `sp=s0`)
- A test to confirm that data on the stack is unchanged (ex. Input: `sp -=4, sw rand0 0(sp)`, Output: `0(sp) = rand0`)

For the highlighted input, I find that they don't include the stdin input. Is it still possible to run the program to test the calling convention?

E Erik Yang TA 2y #882bfe

I think it assumes the stdin is the same for all the other cases

Anonymous Gull 2y #882bdd ✓ Resolved



Q3 ***I C a Scheme***

Consider the following C code:

```
union ExtraStuff {                    typedef struct ConsCell {
    char a[5];                            void *car;
    uint16_t b;                           void *cdr;
    int c;                                union ExtraStuff extra;
    double d;                         } cons;
};
```

FA21-MT-Q3

Just want to confirm:

For union ExtraStuff, the char a[5] is stored as a whole array(5 bytes) rather than just a pointer to the array(4 bytes)?

What if it's initialized by the following, is it stored as a pointer?

```
char* a="something";
```

Yile Hu TUTOR 2y #882bde

1. For union ExtraStuff, the char a[5] is stored as a whole array(5 bytes) rather than just a pointer to the array(4 bytes)?

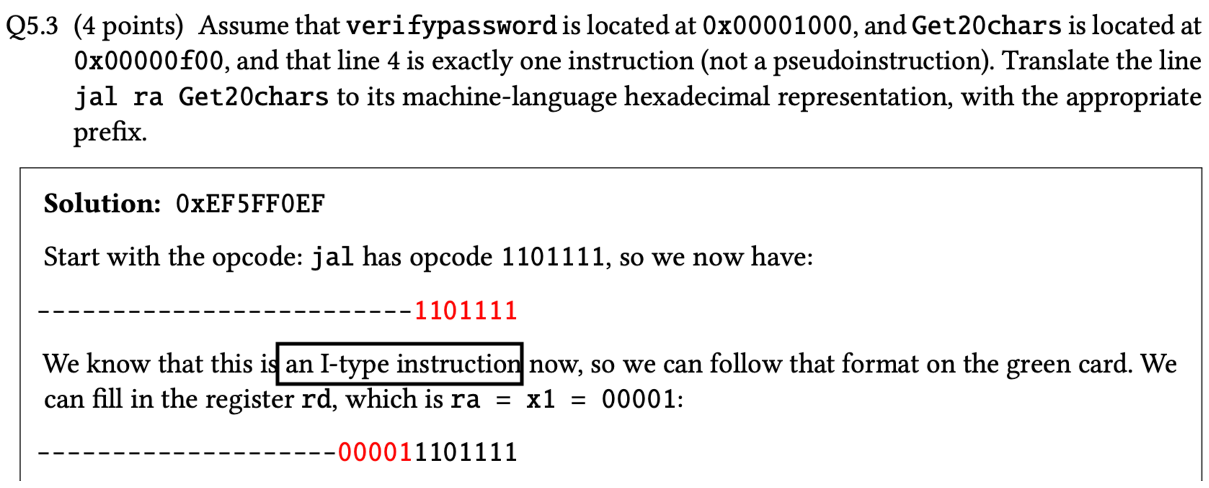
 Yes.

2. What if it's initialized by the following, is it stored as a pointer?

If you have the field changed to `char* a` , then yes it will be a pointer that points to string literal "something" located inside static memory.

Michael Nammour 2y #882bcf ✓ Resolved

FA21-MT-Q5



Q5.3 (4 points) Assume that `verifypassword` is located at `0x00001000`, and `Get20chars` is located at `0x00000f00`, and that line 4 is exactly one instruction (not a pseudoinstruction). Translate the line `jal ra Get20chars` to its machine-language hexadecimal representation, with the appropriate prefix.

Solution: `0xEF5FF0EF`

Start with the opcode: `jal` has opcode `1101111`, so we now have:

```
-------------------------1101111
```

We know that this is an I-type instruction now, so we can follow that format on the green card. We can fill in the register `rd`, which is `ra = x1 = 00001`:

```
--------------------000011101111
```

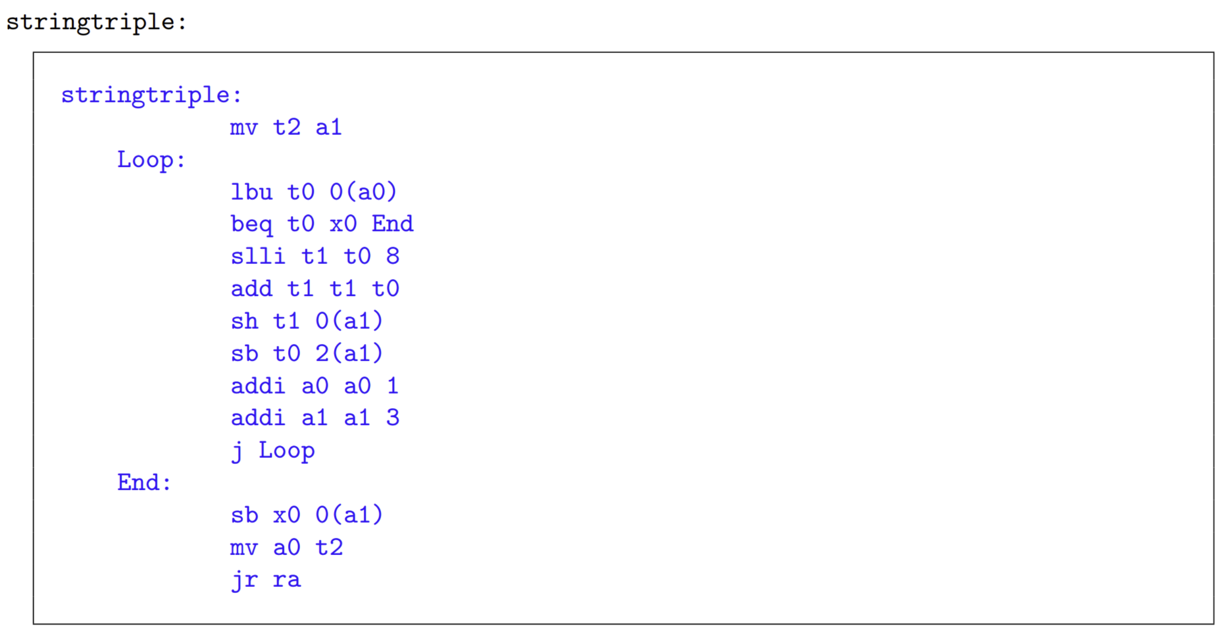
I assume this means to say 'J-type"? Jalr is I-type but jal is J-type in the reference card

1

Erik Yang TA 2y #882bdb

Yeah think you’re right

Anonymous Toad 2y #882bbc ✓ Resolved



stringtriple:

```
stringtriple:
            mv t2 a1
    Loop:
            lbu t0 0(a0)
            beq t0 x0 End
            slli t1 t0 8
            add t1 t1 t0
            sh t1 0(a1)
            sb t0 2(a1)
            addi a0 a0 1
            addi a1 a1 3
            j Loop
    End:
            sb x0 0(a1)
            mv a0 t2
            jr ra
```

why don't we need to store ra on the stack?

Jero Wang ADMIN 2y #882bbe

None of the instructions override ra.

Anonymous Toad 2y #882bcb

do we only need to do that when we call another method

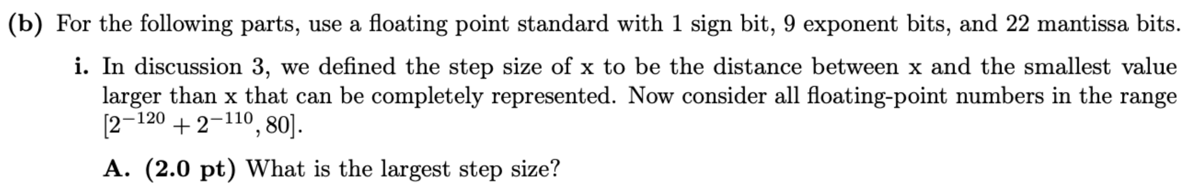
Erik Yang TA 2y #882bcc

Replying to Anonymous Toad

If you call another funciton, then yes you need to save ra

Jason Lee 2y #882bbb ✓ Resolved

SP21-MT-Q8b-i-A



(b) For the following parts, use a floating point standard with 1 sign bit, 9 exponent bits, and 22 mantissa bits.

i. In discussion 3, we defined the step size of x to be the distance between x and the smallest value larger than x that can be completely represented. Now consider all floating-point numbers in the range $[2^{-120} + 2^{-110}, 80]$.

A. (2.0 pt) What is the largest step size?

I have no idea how to approach this problem or what the range means. Some pointers would be much appreciated.

Erik Yang TA 2y #882bce

https://inst.eecs.berkeley.edu/~cs61c/sp22/pdfs/forum-threads/sp22_mt_past_exam_questions_2021.pdf @909_f33

1

Anonymous Parrot 2y #882baf ✓ Resolved

step size around the top part of the range. $80 = 0b1010000 = 0b1.010000 \times 2^6$. The next largest number we can represent is $0b1.010000\ 0000\ 0000\ 0000\ 0001 \times 2^6$ which is $0b0.000000\ 0000\ 0000\ 0000\ 0001 \times 2^6 = 2^{-16}$ larger than 80.

Sp21-MT-8b(A) -- How did we figure out that the next largest number we can represent is that of above? I don't understand why we added so many zeros.

Erik Yang TA 2y #882bcd

you take the mantissa bits, and just add 1, there's 22 mantissa bits, so that's why there's so many zeroes

Anonymous Cobra 2y #882bad ✓ Resolved

Sp21-MT-[q8 b ii B] When I solve this question, I get 0xA6A00000 but the answer is 0xA6900000. Could someone explain how can we get the answer in the solution. The exponent after considering bias is 155, which after modifying the 0.625 for implicit 1 becomes 154. So the exponent bits become 010011010 and the mantissa bits are 1000...0. Is there something wrong in what I'm doing?

2

Anonymous Parrot 2y #882bba

I believe that after considering bias, the exp is actually 1. But I agree, I get the .75 part, but am confused on how to get the 7. If anyone could explain the process, that would be greatly appreciated!

Erik Yang TA 2y #882bda

Exp: You modify 0.625 by multiplying by 2 to get the implicit 1. 0.625 * 2 = 1.25, meaning mantissa is 1.0100...

To multiply by 2, you have to multiply by 2^-1 so that it cancels. So it really is 2^-100 * 2^-1 * 2 * 0.625 = 2^(-101) * 1.25.

To find bias, bias = 2^(b-1) - 1. Exp + bias = -101. Exp + (-255) = -101. Exp = 154.

154 = 010011010 = 128 + 2 + 8 + 16. Sign = 1 because of the negative.

The start of the bits would be 1010 0110 1001.... which equals 0xA69... hopefully this hleps!

Anonymous Cobra 2y #882ede

Replying to Erik Yang

Why are there 9 bits of exponent here, as opposed to the IEEE standard of 8?

Erik Yang TA 2y #882eea

Replying to Anonymous Cobra

I think it's jsut defined in the problem statement that there are 9 bits of exp

Anonymous Parrot 2y #882bab ✓ Resolved

Sp21-MT-q7b(iii) --

I understand we're told to "set the n'th BIT in the bloom filter to 1," and C indexes in bytes. However, I'm a bit confused with why we ***mod*** 8 in `1 « (n % 8)`.

♡ 1 ···

J Jero Wang ADMIN 2y #882bdc

We mod 8 because each int8_t can store 8 bits, and we need to figure out which bit within the correct int8_t to set.

♡ 2 ···

Anonymous Louse 2y #882baa ✓ Resolved

SP21-MT-Q7bii

shouldn't it be (*bf->hashfunc)(element, i) % bf->size because bf->hashfunc evaluates to a pointer to a function, and then we have to dereference that?

♡ ···

J Jero Wang ADMIN 2y #882bbd

#883adc

♡ ···

W Wending Zhao 2y #882aff ✓ Resolved

FA21-MT-Q3.1

3.1 Why we can pass `c->cdr` in the first argument of `map`. It is void* but we should pass a cons type.

♡ ···

C Chenxin Jiang 2y #882bac

A pointer to `void` is a "generic" pointer type. A `void *` can be converted to any other pointer type without an explicit cast.

♡ ···

W Wending Zhao 2y #882bae

you are absolutely correct

♡ ···

Anonymous Salamander 2y #882afb ✓ Resolved

FA21-MT-Q3

I have a conceptual question related to this question. For example, if ExtraStuff was a struct instead of an union, would the sizeof(ExtraStuff) = 8 + 2 + 4 + 8 = 22 because we would have to pad the char a[5] array for it to be considered as 8 bytes since the size of it is 5 which is not a multiple of 4?

♡ 1 ···

E Erik Yang TA 2y #882afc

you would pad the char to be 6 bytes, since b is only a 2 byte integer. Since b is only 2 bytes, it is half word aligned, so it needs to be at a multiple of 2.

♡ 1 ···

Anonymous Salamander 2y #882afd

Oh, I see! Does this technically mean the entire size of the struct has to be a multiple of 4? How does padding work in the first place?

Erik Yang TA 2y #882afe

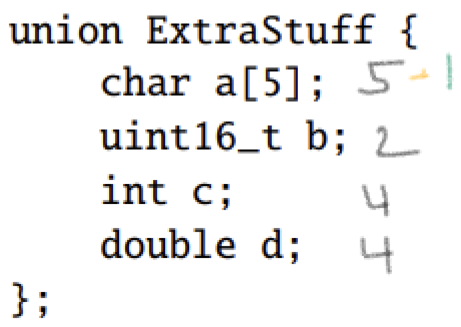
Replying to Anonymous Salamander

the entire size of the struct does not need to be a multiple of 4. Hopefully, #882aaa explains you question

Anonymous Parrot 2y #882cbd

Replying to Erik Yang

On the solutions, it says "no padding is necessary because every field size is a multiple of 4 bytes." However, based on your comment #882afc , what I'm interpreting is the below? (padding +1 to make char 6). Is this right?



```
union ExtraStuff {
    char a[5];      5+1
    uint16_t b;     2
    int c;          4
    double d;       4
};
```

Anonymous Louse 2y #882afa ✓ Resolved

SP-21-Q8b

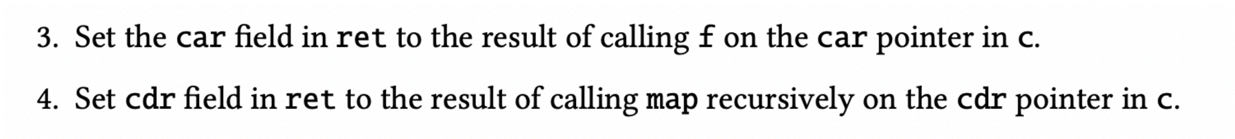
if *bfalloc a function that returns a pointer to a BloomFilter struct? I'm confused because it doesn't have a return statement.

Erik Yang TA 2y #882bbf

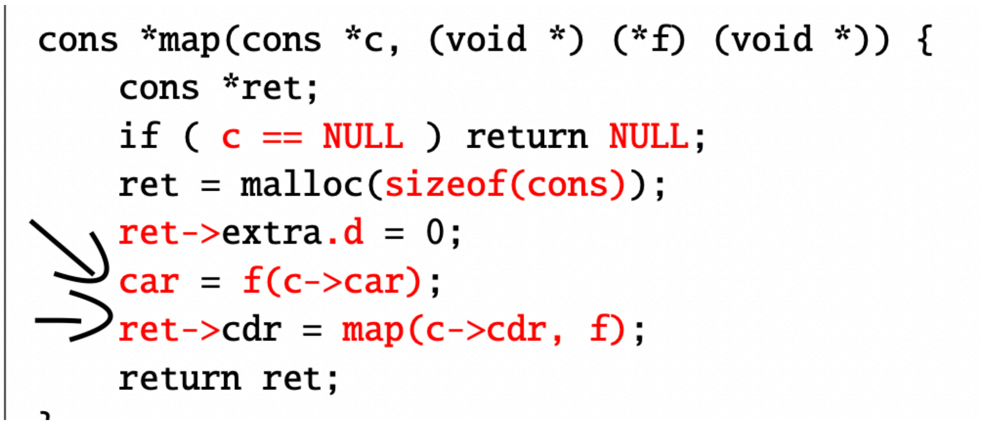
yeah i think the answer's missing a return statement

Michael Nammour 2y #882aec ✓ Resolved

FA21-MT-Q3:



3. Set the `car` field in `ret` to the result of calling `f` on the `car` pointer in `c`.
4. Set `cdr` field in `ret` to the result of calling `map` recursively on the `cdr` pointer in `c`.



```
cons *map(cons *c, (void *) (*f) (void *)) {
    cons *ret;
    if ( c == NULL ) return NULL;
    ret = malloc(sizeof(cons));
    ret->extra.d = 0;
    car = f(c->car);
    ret->cdr = map(c->cdr, f);
    return ret;
```

Hi, why is one of the. 'car' and the other 'ret->cdr' ? Isn't the first also supposed to be 'ret->car' ?

the instructions for both is worded identically and they are both similar fields in ret.

♡ 1 ···

E Erik Yang TA 2y #882bca

think that's a typo, should be ret -> car

♡ ···

Anonymous Sand Dollar 2y #882add ✓ Resolved

FA21-Q5

For line 6, why can't we do "mv t0 Password"? Instead it is "la t0 Password". la takes in a label but in the problem it says Password is a pointer to a statically stored string.

♡ ···

E Erik Yang TA 2y #882ade

mv takes in 2 registers; Password is a label that holds the address of the string

♡ ···

Anonymous Sand Dollar 2y #882adf

How is Password a label? How does storing an array into a label work?

♡ ···

E Erik Yang TA 2y #882aea

Replying to Anonymous Sand Dollar

in the .data section, you can define constants

♡ ···

Anonymous Sand Dollar 2y #882aeb

Replying to Erik Yang

If something is defined externally, then does it mean it is stored in the .data section?

♡ ···

E Erik Yang TA 2y #882aed

Replying to Anonymous Sand Dollar

yup at least for constants

♡ 1 ···

V Vishnu Suresh 2y #882adb ✓ Resolved

[Sp-21-MT-Q6] Is this question is in scope? Do we need to have any background info about 'philphix' before solving this question?

Erik Yang TA 2y #882adc

this is an fsm quesiton so yes. Any background info will be explained in the exam question, but philphix was the proj used instead of snek that sem.

Anonymous Flamingo 2y #882abe ✓ Resolved

sp21-q7 a i. why is calloc(size / 8 + 1) correct, since isn't the data field a pointer so only 4 bytes of memory need to be allocated regardless of the size?

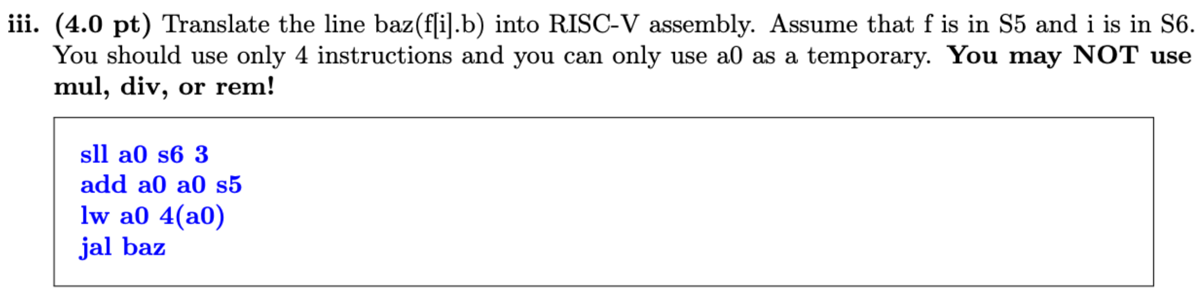
Also calloc(size / 8 + 1) only passes in the number of items and not the size since calloc has two parameters, is the size assumed to be 1?

Erik Yang TA 2y #882acf

the data field is a pointer of 1 byte ints. We have *size* amount of bits in the data field, so you divide by 8 to get the number of 1 byte ints in bloom filter

Anonymous Flamingo 2y #882abd ✓ Resolved

Sp21 7.a.iii what is the line sll a0 s6 3 doing? Why does s6 have to be shifted left by 3 bytes before being added to s5 to get f[i]?



iii. **(4.0 pt)** Translate the line baz(f[i].b) into RISC-V assembly. Assume that f is in S5 and i is in S6. You should use only 4 instructions and you can only use a0 as a temporary. **You may NOT use mul, div, or rem!**

```
sll a0 s6 3
add a0 a0 s5
lw a0 4(a0)
jal baz
```

Erik Yang TA 2y #882acb

#882aaf

Anonymous Parrot 2y #882abc ✓ Resolved

FA21-MT-Q6 -- Is this in scope? I believe we didn't learn about creating tests. If it is, what are some resources/lecs that I can refer to?

Erik Yang TA 2y #882aca

This is in scope

1

Anonymous Parrot 2y #882acd

Thanks! Do you know what resources/lecs that I can refer to learn more about testing?

Erik Yang TA 2y #882ace

Replying to Anonymous Parrot

this question wasn't really testing how much you know about testing, it's more about what you could pinpoint in terms of their riscv implementation

1

Anonymous Parrot 2y #882aba ✓ Resolved

FA21-MT-Q5: How come we store the ra into the allocated space (`sw ra 20(sp)`)? Doesn't this `20(sp)` allocated space get filled up with something else?

Jero Wang ADMIN 2y #882abb

Something else as in the password from `Get20chars`? Those 20 chars are stored from `0(sp)` until `19(sp)`, `ra` occupies `20(sp)` until `23(sp)`.

1

Anonymous Badger 2y #882aad ✓ Resolved

SP21-MT-Q3

for part iii, can someone help me break down how to translate the C code to risc V? thanks!

Jero Wang ADMIN 2y #882aaf

To access `f[i]`, you need to calculate the offset first, which can be done with `slli a0 s6 3` since each `struct foo` is 8 bytes long. Then, you can calculate a pointer to the `i` th element of `f` with `add a0 s5 a0`. Then, since the `b` member of `foo` is the second member at offset 4 due to padding, you can get the value by using `lw a0 4(a0)`. Finally, since your argument to `baz` is already in `a0`, you can call it directly with `jal baz`.

1

Anonymous Butterfly 2y #882aee

How do we know that `struct foo *f` is an array of foos? Could this parameter also mean just taking in one foo struct pointer? Do we just assume that it is an array because `f[i]` kinda suggests this.

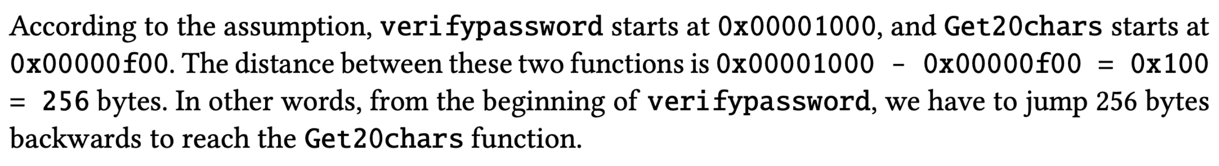
Erik Yang TA 2y #882bef

Replying to Anonymous Butterfly

Yeah usually a pointer f means an array of f structs. F[i] also does suggest this

Anonymous Gull 2y #882fd ✓ Resolved

FA21-MT-Q5.3: How do you subtract 2 hex numbers? In the screenshot why does the 2 addresses subtracted equal 0x100.


According to the assumption, `verifypassword` starts at `0x00001000`, and `Get20chars` starts at `0x00000f00`. The distance between these two functions is `0x00001000 - 0x00000f00 = 0x100 = 256` bytes. In other words, from the beginning of `verifypassword`, we have to jump 256 bytes backwards to reach the `Get20chars` function.

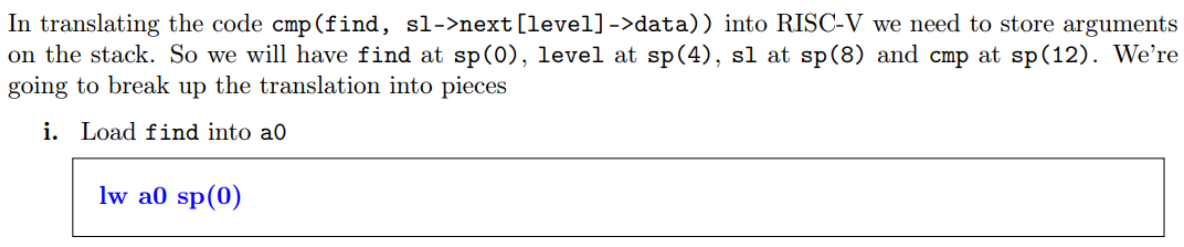
1

Rosalie Fang ADMIN 2y #882fe

You can convert into binary, then subtract in binary, then convert to hex.

♡ ···

Anonymous Gull 2y #882ef ✓ Resolved



In translating the code `cmp(find, sl->next[level]->data))` into RISC-V we need to store arguments on the stack. So we will have `find` at `sp(0)`, `level` at `sp(4)`, `sl` at `sp(8)` and `cmp` at `sp(12)`. We're going to break up the translation into pieces

i. Load `find` into `a0`

```
lw a0 sp(0)
```

SP21-Final-Q6 (B) i

Is sp(0) equivalent to 0(sp)?

♡ ···

E Eric Kusnanto TA 2y #882fb

Yes.

♡ ···

Anonymous Salamander 2y #882ee ✓ Resolved

SP21-MT-Q1c(iii)

I don't think I get the conversion of 64 in bias notation. How did we get all 1s?

♡ ···

E Erik Yang TA 2y #882fc

num + bias = 64 ; num = 127 = all 1s

♡ ···

Anonymous Parrot 2y #882de ✓ Resolved

SP21-MT-Q5 -- Is this in scope, and if so, what resources/lec can we refer to? I've watched all the lectures but don't recall seeing anything like this.

♡ 2 ···

E Erik Yang TA 2y #882eb

nope

♡ 3 ···

Anonymous Reindeer 2y #882dd ✓ Resolved

sp21-mt-q8a

why the answer is + 2^21 not - 2^21. I thought we should also subtract the inf and NaN for standard A/

♡ ···

E Eric Kusnanto TA 2y #882fa

They just distributed the sign because we're subtracting all possible positive values for standard A

$$(B_{pos} - B_{NaN,\,\text{inf}}) - (A_{pos} - A_{NaN,\,\text{inf}}) = B_{pos} - B_{NaN,\text{inf}} - A_{pos} + A_{NaN,\text{inf}}$$

♡ ···

Anonymous Gull 2y #882cf ✓ Resolved

FA21-MT-Q4.5: I got 2^-8 for the significand how does it multiply out to be 2^-70?

Erik Yang TA 2y #882db

it's 2 ^ (exp + bias + 1) * 0.mantissa = 2 ^ (0 - 63 + 1) * 2 ^ -8 = 2 ^ -70

1

Zhengnan Ma 2y #882cb ✓ Resolved

SP21-MT-Q7 I

How do we know the sizeof(struct)?

Zhengnan Ma 2y #882cc

SP21-MT-Q7 III

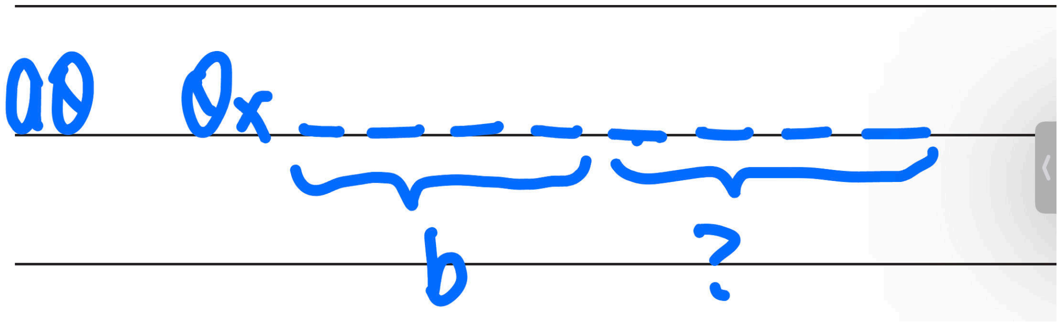
I'm also confused why the offeset of a0 is 4? What's stored in the a0 and where is the fib[i].b located in the a0?

Erik Yang TA 2y #882da

its 4(a0) because you want to access its *b field, which starts at the 4th byte of the struct

Zhengnan Ma 2y #882df

Replying to Erik Yang



Is is right?

Where is char a located?

Erik Yang TA 2y #882ea

Replying to Zhengnan Ma

char a is located at 0(a0)

Zhengnan Ma 2y #882ec

Replying to Erik Yang

But char a is only one byte. So the rest of place is padding?

Erik Yang TA 2y #882ed

Replying to Zhengnan Ma

yeah it's word aligned

E

Erik Yang TA 2y #882ce

you add up all the bytes of the structs' fields, plus padding

Anonymous Cheetah 2y #882ff

what do you mean padding

E

Erik Yang TA 2y #882aaa

Replying to Anonymous Cheetah

Padding means you word align fields of a struct so like a char will have 3 bytes of padding to make it 4 bytes

Sorry let me reexplain my logic here: padding is only relevant when the next field in the struct requires alignment. So if it goes char a, int b: this requires 3 bytes padding on the char because int b needs to be at a multiple of 4. However, if it was char a, char b, int c: char a does not need padding since b is just one byte, so it would just be 1 + 1 + 2 bytes of padding (to make c word aligned) + 4 int bytes

1

Anonymous Cheetah 2y #882aab

Replying to Erik Yang

Does it pad to the nearest 4 bytes, or to the max bytes of an element in the struct?

E

Erik Yang TA 2y #882aac

Replying to Anonymous Cheetah

every 4 bc it is 32 bit aligned; take a look at my edited response above hopefully it makes more sense

1

Anonymous Cheetah 2y #882aae

Replying to Erik Yang

are ints the only data type that require word alignment?

E

Erik Yang TA 2y #882abf

Replying to Anonymous Cheetah

Nope, there’s also ptrs and double and then shorts need to be every 2 bit aligned or half word aligned

D

David Babazadeh 2y #882eed

Replying to Erik Yang

what other structures are aligned/have this padding? structs are, unions aren't?

E

Erik Yang TA 2y #882eee

Replying to David Babazadeh

structs are padded, unions aren't. Unions just are the size of the biggest element in the union, in bytes.

D

David Babazadeh 2y #882efb

Replying to Erik Yang

thank you

Anonymous Otter 2y #882eff

Replying to Erik Yang

" However, if it was char a, char b, int c: char a does not need padding since b is just one byte, so it would just be 1 + 1 + 2 bytes of padding (to make c word aligned) + 4 int bytes "

Could you explain further on how those bytes in particular are chosen for padding?

E

Erik Yang TA 2y #882fab

Replying to Anonymous Otter

Yeah, so if it is {char a; char b; int c} then char a is one byte. Since b does not need to be aligned at all since it is just one byte, then a does not need any padding. However, b needs padding since c is an integer, which is word aligned. That's why b needs 2 bytes of padding because it needs to reach a byte that is a multiple of 4. So 1 (char a) + 1 (char b) + 2 (bytes of padding to make sum so far be a multiple of 4) + 4 (int bytes)

Anonymous Sand Dollar 2y #882ca ✓ Resolved

Fall22-midterm Q1.8 and Q1.7. I have a clarification question.

1. If a number is a twos complement number then when you have the binary form you add everything up as usualy except the leftmostbit will be negative number added tothe sum.

2. If a number is sign-mgnitude then when you have the binary form you add everything up and if the leftmostbit is 0, then what you added up will be positive and if it is 1 then it will be negative?

E

Erik Yang TA 2y #882dc

1. if first bit is 1, then it's negative, so just flip and add 1 to get your number and make it negative. If negative, just follow normally

2. yup

Anonymous Gorilla 2y #882bc ✓ Resolved

FA21-MT-Q4.5

Why is the answer 1*2^-70 rather than 1*2^-71?

Y

Yile Hu TUTOR 2y #882bf

For denorm the floating point is in the form `2^(bias+1) * 0.MMMMMMMM.` Since the smallest significand we can have is `2^-8` , this multiplies out to be `2^(-70)`

♡ 1 ···

Anonymous Toad 2y #882bb ✓ Resolved

FA21-MT-Q5.3

Just to double check what the solution means by "Labels aren't stored in memory", if the address of verifypassword is `0x00001000` , is the address of line2 also `0x00001000` or is it `0x00001004` ?

♡ ···

Yile Hu TUTOR 2y #882bd

The address of line2 is `0x00001000`

♡ 1 ···

Anonymous Toad 2y #882ba ✓ Resolved

FA21-MT-Q4.6

I'm a little confused on the explanation in the solution. If the significand has 8 bits and we include the implicit 1, we can get a big number of `1.1111_1111 * 2^(71-63) = 1_1111_1111` . Isn't this the largest value which is equal to `2^9 - 1`? Thus, the next number which is unrepresentable is `2^9-1+1 = 2^9` ?

♡ 2 ···

Yile Hu TUTOR 2y #882be

At `2^9` the step size becomes 2, so `2^9+1` is the first integer that is skipped over. `2^9` is representable by setting the mantissa to all zeros and exponent to `72`

♡ 1 ···

Anonymous Parrot 2y #882cbb

In general, does the step size become 2 once you exceed the number of bits available for the significand? Or how did you know that the step size becomes 2 at `2^9`?

♡ ···

Yile Hu TUTOR 2y #882cbf

Replying to Anonymous Parrot

For floating point number in the form `0b1.MMMM...M * 2^(x)` , the step size would be `2^(x) * (the weight of the LSB in the significand)` .

♡ 1 ···

Anonymous Alpaca 2y #882dbe

Replying to Yile Hu

By weight of the LSB in the significand, do you mean the number of mantissa bits?

♡ ···

Yile Hu TUTOR 2y #882ddf

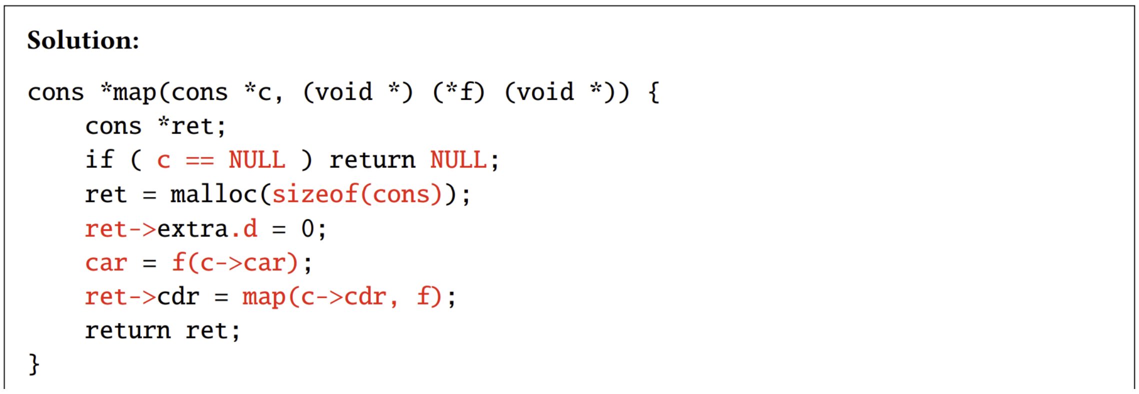
Replying to Anonymous Alpaca

I mean the weight of the rightmost bit in `0b1.MMM...M` . Here, since there are 8 mantissa bits, this weight is `2^(-8)` . And yes, this is equal to the number of mantissa bits.

♡ ···

D David Yang 2y #882ae ✓ Resolved

FA21-MT-Q3.1



Solution:

```
cons *map(cons *c, (void *) (*f) (void *)) {
    cons *ret;
    if ( c == NULL ) return NULL;
    ret = malloc(sizeof(cons));
    ret->extra.d = 0;
    car = f(c->car);
    ret->cdr = map(c->cdr, f);
    return ret;
}
```

Why is the solution `car = f(c->car);` and not `ret->car = f(c->car);`

♡ ···

E Eric Kusnanto TA 2y #882af

Thanks for the catch, it should be ret->car. Under the solutions rubric, it says that Blank 6 should be "ret->" or equivalent, but we missed that in the fully written code segment.

♡ 1 ···

Anonymous Gull 2y #882ac ✓ Resolved

SP21-MT-Q8a:

how do u get to the answer? i'm not sure how to go abt the question

♡ ···

E Eric Kusnanto TA 2y #882ad

Any combination of bits of the exponent + mantissa with sign bit of 0 will result in a unique positive integer, except for the cases where the exponent bits are 1...1. In this case, depending on the mantissa, the float is either +inf or NaN, so we subtract all those possibilities from our count of positive values.

♡ ···

Anonymous Gull 2y #882aa ✓ Resolved

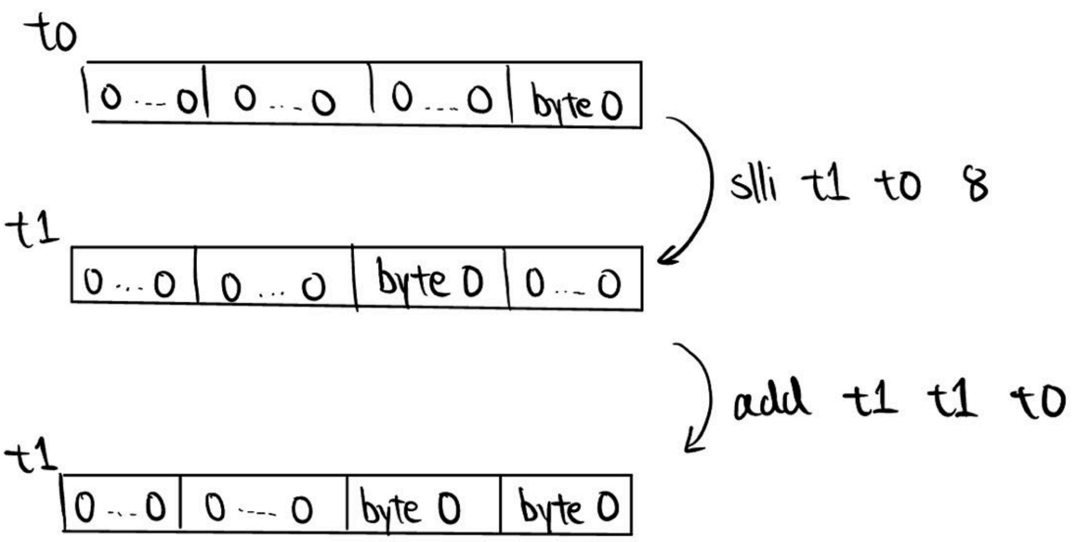
SP21-MT-Q4:

In the 3rd line of Loop, why is t0 slli by 8?

♡ ···

Y Yile Hu TUTOR 2y #882ab

We shift t0 left by 8 bits (or 1 byte equivalently) so that the 0th byte of t0 goes to the 1st byte of t1, the 1st byte of t0 goes to the 2nd byte of t1, etc...



t0
0...0 | 0...0 | 0...0 | byte 0
slli t1 t0 8
t1
0...0 | 0...0 | byte 0 | 0...0
add t1 t1 t0
t1
0...0 | 0...0 | byte 0 | byte 0

♡ 1 ···

Anonymous Gorilla 2y #882e ✓ Resolved

SP21-MT-Q2c

This part of the question asks about symbol and relocation tables - will these be in scope for our exam, and if so, which lecture/topic were they covered in?

♡ ···

E Erik Yang TA 2y #882f

I believe this is part of the CALL lecutre

♡ 1 ···

Anonymous Elk 2y #882c ✓ Resolved

SP21-MT-Q7

Hi, this is roughly related to SP21-MT-Q7. When we are calculating sizeof for a struct, do we generally consider the fields in the struct to be tightly packed or allow extra memory for word alignment? Thanks!

♡ 1 ···

E Erik Yang TA 2y #882d

you can generally assume it's padded for word alignment unless specified

♡ ···

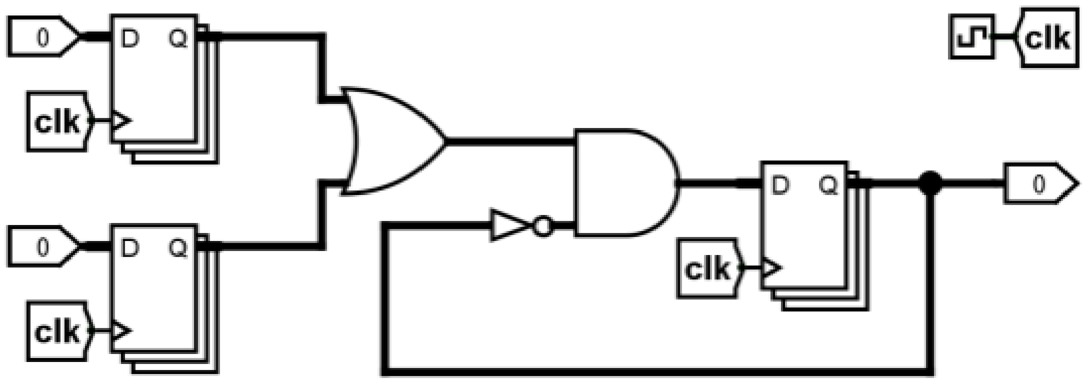
Anonymous Ant 2y #882a ✓ Resolved

FA21-Final-Q4

More of a conceptual question, when talking about max hold time is it the hold time for the register on the left? Also, when talking about the clock period and delay are we talking about the clock periods of all registers? Is it best to have the clock periods of all registers the same? Thank you!

Q4 *Bit of a Delay* **(10 points)**

Consider the following circuit. Assume that AND and OR gates have a delay of 8 ps (picoseconds), NOT gates have a delay of 4 ps, and all registers have a setup time constraint of 6 ps and clock-to-Q delay of 3 ps. Assume all wires are ideal, i.e. they have zero delay.



Q4.1 (2 points) What is the largest combinational delay of all paths in this circuit, in picoseconds?

[ ] ps

Solution: 16 ps

The longest path between two registers goes through the AND gate, then the OR gate, for a total delay of 8+8=16 ps.

Q4.2 (2 points) What is the smallest combinational delay of all paths in this circuit, in picoseconds?

[ ] ps

Solution: 12 ps

The shortest path between two registers goes through the NOT gate, then the AND gate, for a total delay of 8+4=12 ps.

Q4.3 (2 points) What is the maximum possible hold time constraint for registers to function properly in this circuit, in picoseconds?

[ ] ps

Solution: 15 ps

Hold time = smallest combinatorial delay + clock-to-Q delay = 12+3 = 15 ps

···

E Eric Kusnanto TA 2y #882b

Here and by convention (though it is not always the case! see HW5) every register is connected to the same clock signal. When calculating hold time, because each register triggers at the same time, we have to take in account each register to ensure no hold time violations.

1 ···