

[Midterm] Past Exams - 2019 and Older #880



Jero Wang ADMIN
2 years ago in **Exam – Midterm**

662
VIEWS



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 2019 final walkthrough](#)

[Summer 2019 final walkthrough](#)

- Q1 Potpourri: <https://youtu.be/FY5dAMrXvx0>
- Q2 FSM: <https://youtu.be/gmHbw6LSeSw>
- Q3 C Coding: <https://youtu.be/v4B1WTs5UNU>
- Q4 RISC-V: <https://youtu.be/2VHjG-gy9Dk>
- Q5 Data-Level Parallelism: <https://youtu.be/oG9Rrzmi0M4>
- Q6 RAID and ECC: <https://youtu.be/rfcNTIzNZ2M>
- Q7 Caches: <https://youtu.be/xojc8YZaO3Q>
- Q8 Spark: <https://youtu.be/A37BFXRXmm0>
- Q9 Datapath: <https://youtu.be/q-T4N3hBhUM>
- Q10 Digital Logic: <https://youtu.be/3RI36lsDSg4>
- Q11 Virtual Memory: https://youtu.be/5_2fKsK4I34



Anonymous Magpie 2y #880adb

✓ Resolved

SP22-MT1-Q3b

Why does line 2 not print?



Erik Yang TA 2y #880ade

Which midterm is this



Anonymous Magpie 2y #880aea

Sorry— it's SP18



R [Rosalie Fang](#) ADMIN 2y #880aeb

← Replying to Anonymous Magpie

```
create_song() assigned song->artist = artist, with char artist[100] = "mac demarco";
```

artist lives on the stack, so `song->artist` will not be retrievable after the function `create_song()` closes. That space will have to be either `malloc`-ed, or be assigned a global data (e.g. directly doing `song->artist = "Mac Demarco"`) for it to work.

♡ 1 ...



[Anonymous Porpoise](#) 2y #880ada

✓ Resolved

SP19-MT1-q2e

```
int size = 0;
struct map_entry {
    char *key;
    char *value;
};

void add_entry(struct map_entry *m, char *k, char *v) {
    int *zero = NULL;
    m[size].key = k;
    m[size].value = v;
    size++;
}

void main(void) {
    struct map_entry *map = malloc(sizeof(struct map_entry) * 10);
    char *key = malloc(sizeof(char) * 10);
    char value[20];
    add_entry(map, "k", "v");
    add_entry(map, key, value);
}
```

(e) How many bytes of memory are leaked by this program?

Solution: 90 Bytes

How did we get to the answer of 90? I think I'm having trouble with the `sizeof` of the `map_entry` struct. From what I can tell, there are 32 bytes for the characters, 20 bytes for the pointers (each `map_entry` contains 2 pointers and there are 2 `map_entries`, and the `map` pointer itself), but I'm not sure what else to account for.

♡ ...



[Andrew Liu](#) ADMIN 2y #880adc

There are no `free` calls in the program, so we count the total amount of space `malloc`'d, as it will all be leaked. `sizeof(struct map_entry)` is 8 bytes, since each pointer is 4 bytes (The instructions say "For this problem, assume all pointers are four bytes and all characters are one byte."), so `malloc(sizeof(struct map_entry) * 10)` allocates 80 bytes, and then `malloc(sizeof(char) * 10)` allocates 10 bytes, for a total of 90 bytes.

♡ 1 ...

Anonymous Armadillo 2y #880acf ✓ Resolved

Su18-MT1-Q3.2

Hello, would it be possible to explain the blue sections? Thank you

<pre>1. foo: 2. slli t6 a0 2 3. sub sp sp t6 4. mv t4 sp 5. sw zero 0(t4) 6. addi t1 zero 1 7. L1: bge t1 a0 Next 8. andi t2 t1 1 9. slli t3 t1 2 10. add t3 t3 t4 11. sw t2 0(t3) 12. addi t1 t1 1 13. j L1 14. Next: mv t1 zero 15. mv t2 zero 16. slli a0 a0 2 17. L2: bge t1 a0 End 18. add t3 t4 t1 19. lw t3 0(t3) 20. add t2 t2 t3 21. addi t1 t1 4 22. j L2 23. End: mv a0 t2 24. add sp sp t6 25. jr ra</pre>	<p>Translate the RISC-V Assembly on the left into C code to complete the function foo:</p> <pre>unsigned foo(unsigned n) { unsigned arr[n]; unsigned total = 0; unsigned *ptr = arr; ptr[0] = 0; for (int i=1; i<n; i++) { ptr[i] = i&1; } for (int i=0; i<n; i++) { total += ptr[i]; } return total; }</pre>
--	---

♡ ...

Andrew Liu ADMIN 2y #880add

Lines 2-4 all factor into the purple line you have marked. Line 2 multiplies `a0` (which is the input `n`) by 4, since `unsigned` is 4 bytes. Then, you create space on the stack by subtracting this value from the stack pointer, which creates $4 * n$ bytes on the stack to be used. Then, `mv t4 sp` moves the pointer to the stack pointer to `t4`, meaning that the pointer to `arr` now 'lives' in `t4`.

Line 5 then sets `0(t4)`, which is `arr[0]` in C to 0.

In L1, we run the `for` loop, with `i` being stored in `t1`. What the code does is first it `and`s `i` and `1` together, which you marked, and saves it in `t2`. Then, since assembly doesn't automatically do pointer arithmetic for us (`ptr[i] = *(ptr + i * sizeof(unsigned))`), we have to manually multiply `t1` by 4 (the size of `unsigned`) using `slli`, then add it to our pointer to get to the next element, and then save `t2`, which is `i & 1` to `ptr[i]` (which is `ptr + 4 * i`)

Lastly, line 16 just multiplies `a0` by 4 to make `a0` be the number of bytes in `ptr` so that the loop works because assembly doesn't have automatic pointer arithmetic.

♡ ...

Anonymous Armadillo 2y #880adf
Thank you for the detailed explanation!



Anonymous Armadillo 2y #880ace

✓ Resolved

Su18-MT1

How did we get these answers? I don't think I understood the question :(Thank you

For each of lines that have the numbered comment select the best answer from

- A. Legal and Initialized
- B. Legal and Uninitialized
- C. Possibly Legal
- D. Illegal

For example for question 6 you should answer about the line with the `/**** 6 ****/` comment from when the program runs.

6.	<input type="radio"/> A	<input type="radio"/> B	<input checked="" type="radio"/> C	<input type="radio"/> D
7.	<input type="radio"/> A	<input checked="" type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D
8.	<input checked="" type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D
<hr/>				
9.	<input type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input checked="" type="radio"/> D
10.	<input checked="" type="radio"/> A	<input type="radio"/> B	<input type="radio"/> C	<input type="radio"/> D



Deniz Demirtas 2y #880acc

✓ Resolved

Fa19-MT2-Q4: How do we determine the immediate in part B?



Eric Kusnanto TA 2y #880acd

The immediate for B-type instructions is the byte offset from our current instruction to the label (returnnode), which can be calculated by the number of instructions * sizeof(instruction), which in this case gives us a byte offset and immediate of +32. As these local jumps are always going to be relative, adding libraries won't change the relative offset in the machine code for Part C



Anonymous Armadillo 2y #880aaf

✓ Resolved

Fa19-Final-Q5a: Why is the path for register 1: path of output of reg1 --> not --> or? i'm confused also how to find the longest setup time.

♡ ...

E Erik Yang TA 2y #880abb

to find the longest setup time, you're given the equation for critical path

$\text{setup} + \text{clk to q} + \text{longest combo path} \geq \text{min clock period}$

You're given the frequency which you can do $1/\text{freq}$ to find clock period, and then you want to find the longest combo path from reg 1, which is through reg1 --> not --> or = 14. Now, all you want to do is maximize setup time. So for reg1: $\text{setup} + 16 = 20$ so $\text{setup} = 4$ ns. Then, you have to find the same thing for reg2, and then find the min between the two

♡ 1 ...

Anonymous Armadillo 2y #880abc

i see, when do you consider the delay of the register for a combo path? i thought it was just the propagational delay of the gates, not the registers itself also.

♡ ...

E Erik Yang TA 2y #880abd

↩ Replying to Anonymous Armadillo

Yes you're right, for a combo path it is just the delay of the gates. You want to include clk to q bc it's part of the setup time equation

$\text{Clk to q} + \text{setup} + \text{longest combo}$

♡ 1 ...

Anonymous Armadillo 2y #880abe

↩ Replying to Erik Yang

that makes sense thank you! and last question, for register 2 why do we consider the time for X + AND, i thought it would be OR + AND

♡ ...

E Erik Yang TA 2y #880abf

↩ Replying to Anonymous Armadillo

There's 2 possible paths that reg2 outputs to: the OR gate on the left (10 ns) and the NOT + AND ($4+2=6$) in the middle back to reg2. Now you just find the the max between the 2

♡ 1 ...

Anonymous Armadillo 2y #880aca

↩ Replying to Erik Yang

okay i see! so sorry, but why is the or gate being taken into consideration? if we are only looking at paths for output of reg1 to input of reg2 how is the left or gate on the path and if we r looking at the or gate shouldn't we also use the not gate?

♡ ...


E Erik Yang TA 2y #880acb


↩ Replying to Anonymous Armadillo


No worries. You see that the reg2 output follows all the way to the or gate on the left side and it also goes to the not gate in the middle. Therefore there's 2 possible paths.


The delay is only considered if it goes through that logic gate. The wire never goes through the not gate on the left so it does not need to consider that delay

♡ 1 ...


 **Anonymous Chough** 2y #880aae ✓ Resolved
Fa19-Final-Q2d: why does the byte "0x03" correspond to "ETX"?
♡ ...


 **Catherine Van Keuren** TA 2y #880aba
In this case ETX (end of text) is the ASCII character that equates to '0x03'. However, this value is not on the current reference sheet. If any kind of ASCII conversion is required, you can expect it to be on the reference sheet.
♡ ...


 **Anonymous Emu** 2y #880aaa ✓ Resolved
SU19-MT-Q4.2: it says to be careful of the arrow vs . notation. why do we use a dot here and not an arrow? Isn't caption_t a struct?
♡ ...


 **Sam Xu** TUTOR 2y #880aab
Arrow is used for the reference of struct. Dot is used for a struct object.
For instance:
struct caption_t *cap;
cap->a


Or
Struct caption_t cap;
cap.a
♡ ...

 **Anonymous Emu** 2y #880ff ✓ Resolved
SU19-MT-Q3.1: is the proper way to do this `malloc(len(arr)*4)`? or does malloc automatically multiply by 4?
♡ ...

 **Sam Xu** TUTOR 2y #880aac
Yes, we do need multiply 4 here.
♡ ...

 **Anonymous Emu** 2y #880fd ✓ Resolved
SP18-MT-Q9: when questions mention bias of 15, can we assume that they mean -15? bc i think in this question they make that assumption
♡ ...

 **Erik Yang** TA 2y #880fe
Yeah that assumption makes sense
♡ ...

 **Anonymous Emu** 2y #880fc ✓ Resolved
SP18-MT-Q9C: why are we subtracting 2^5 from 2^{16} ? i understand why the largest number representable is 0 11110 1111111111, but this equals

$$1. 1111111111 * 2^{(\text{maxexp-bias})}$$

$$= 1. 1111111111 * 2^{(30-15)}$$

$$= 2^{(-10)} * 2^{(15)} = 2^5$$

so why are we subtracting this from 2^{16} ? thanks!

♡ ...

↳ S **Sam Xu** TUTOR 2y #880aad

You are right that the largest number is $1.1111111111 * 2^{15}$. However, 1.1111111111 is not $2^{(-10)}$. It actually equals to $2^0 + 2^{-1} + 2^{-2} + 2^{-3} + \dots$

After simplified, the result will be $2^{16} - 2^5$.

♡ ...

Anonymous Squirrel 2y #880fa

✓ Resolved

Fa19-Final-Q5d

1. How does $x + !xy = x + y$?

$$\begin{aligned} \text{out} &= \overline{C}((B + \overline{B} \overline{D}) + (\overline{A} \overline{D} + D)) \text{ (Associative)} \\ \text{out} &= \overline{C}(B + \overline{D} + \overline{A} + D) \end{aligned}$$

2. How do you go from line 1 to line 2?

$$\begin{aligned} \text{out} &= \overline{C}(\overline{A} + B + 1) \text{ (Inverse)} \\ \text{out} &= \overline{C} \text{ (Identity)} \end{aligned}$$

♡ ...

↳ Anonymous Armadillo 2y #880fb

1. I think its because X is either x or !x and if X = x, then the expression is equivalent to x. The only other choice for X is !x and when X= !x, the equivalent expression is y. So we can just write the eq as $x + y$ since X cannot equal x if it equals !x.

2. Anything ORed with 1 is 1 so the expression in the parantheses is 1

♡ 1 ...

Anonymous Armadillo 2y #880ec

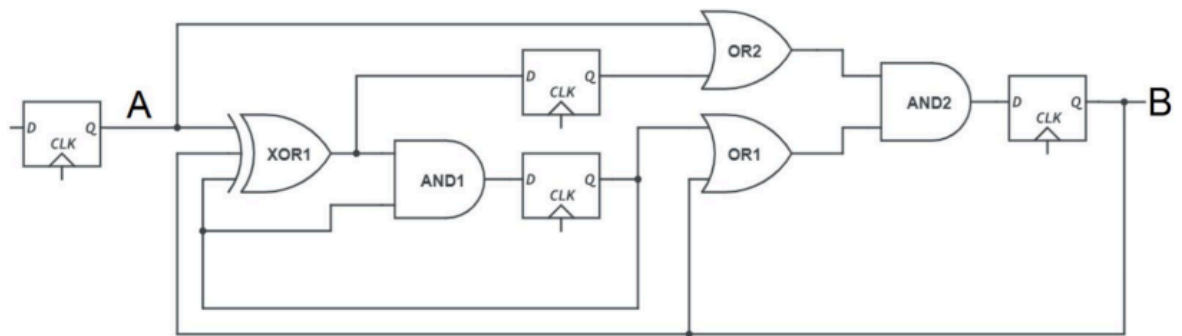
✓ Resolved

SU19-Q3-(no problem section?) last part

Hello, is latency in scope for the midterm? If so, what does it mean? Thank you

For the next problems, consider the following pipelined circuit. Assume all registers have their clock inputs correctly connected to a global clock signal and that logic gates have the following parameters:

- XOR gate delay: 80 ps
- AND gate delay: 60 ps
- OR gate delay: 40 ps



When shopping for registers, we find two different models and want to determine which would be best for our circuit.

Register Type λ

- Setup Time: 40 ps
- Hold Time: 20 ps
- Clock-to-Q Delay: 30 ps

Register Type τ

- Setup Time: 10 ps
- Hold Time: 10 ps
- Clock-to-Q Delay: 80 ps

Critical Path = CLK_Q + XOR + AND + SETUP

Because this passes through 2 registers, our latency is 2 clock cycles.

Note after release we found 2 other interpretations to this question. 1 has just 1 critical path because it considers the latency to be just the top path A takes to B. The second also counts an extra clock to q to give A its value or propagate through the last register to B.

What is the minimum latency for the circuit from A to B if we use register type λ ?

$$2 * (30\text{ps} + 80\text{ps} + 60\text{ps} + 40\text{ps}) = 420\text{ps}$$

♡ ...

Sam Xu TUTOR 2y #880ee

Latency is the time it takes for a packet of data to travel from source to a destination. I think it is in the midterm's scope.

♡ ...

Anonymous Armadillo 2y #880ef

Got it, thanks!

♡ ...

Anonymous Mole 2y #880eb

✓ Resolved

FA17-Final-Q1

2. Please fill in `power_of_16` to calculate whether the given integer is a power of 16.

Be sure to only use bitwise operators, `==` and `!=` and up to 1 subtraction. You may introduce constants but not any new variables.

Return 0 if it is not a power of 16, or 1 if it is. (**Note: 0 is not a power of 16**).

```
// sizeof(int) == 4
int power_of_16(int m) {
    # check sign bit & # of 1-bit in binary representation
    if (0 != ((m >> 31) & 1) || (m & (m-1)) != 0) {
        return 0;
    } else {
        return m != 0 & (m & 0xEEEEEEEE) == 0;
    }
}
```

I don't understand the solution. What do these lines of code mean? Thanks!

♡ ...

Y Yile Hu TUTOR 2y #880ed

This is definitely an interesting question.

Notice that negative integer cannot be power of 16, and any two's complement number is negative if its MSB is 1. `0 != ((m >> 31) & 1)` checks if the MSB is 1 (integers are 32 bit long), and if it is 1, output 0.

`m & (m-1) != 0` checks if the integer has more than one bit set to 1.

A quick example on why [this](#) is the [case](#)

If `m` has only one of the 32 bits set,

Let `m = 0b0...010...0`

then `m - 1 = 0b0...001...1`, and `m & m-1` outputs `0`

If `m` has at least two of the 32 bits set,

Let `m = 0b0...0110...0`

then `m - 1 = 0b0...0101...1`

notice that the left "1" bit will be preserved during the subtraction,

so `m & m-1` outputs a nonzero value

If the integer has more than one bit set to 1 then it cannot be a power of 16. We can get an intuition of why this is true by looking at a few examples of power of 16.

`1 = 16^(0) = 0x00000001`, ^ here means exponentiation.

`16 = 16^(1) = 0x00000010`

`256 = 16^(2) = 0x00000100`

...

Lastly, `m & 0xEEEEEEEE` will only be 0 if `m` has the 1 bit set at the 0th, 4th, 8th,... bit position (note that `0xE = 0b1110`), this fits into our earlier observation about powers of 16, so the function returns 1 only if `m & 0xEEEEEEEE` gives 0, and `m!=0` gives 1.

♡ ...



Anonymous Armadillo 2y #880df

✓ Resolved

Su19-MT1-Q3

Why is the implementation of the description correct? Isn't the code doing opposite of the description?

```
/* Function that takes in an integer, interprets it as a boolean value,
 * and returns a string that can be dereferenced outside the function
 * indicating if it was true or false.*/
char* bool_to_string (int i) {
    /* Allocates space for a pointer. */
    [ ] char* ret_val;

    /* Evaluates to true on all false values and false on all true values. */
    [X] if (i == 0) {
        ret_val = "false"; ← shouldn't this be true
    } else {
        ret_val = "true"; ← true?
    }

    /* Returns a pointer that can be dereferenced in other functions. */
    [ ] return ret_val;
}

[ ] no errors
```

♡ ...



E Erik Yang TA 2y #880ea

I think you're right

♡ ...



Anonymous Armadillo 2y #880cd

✓ Resolved

Sp19-MT1-2e (?)

How did we get the answers for parts b and d? The code is the same as the one in the picture in the post below this one. Thank you

(e) Bubble the comparators that would make the following expressions evaluate to true. If there is not enough information in the problem to answer conclusively, select the last option. Assume malloced memory grows upward, allocating the first available address.

(a) `map == zero`

`>`

`<`

`==`

Not enough information

(b) `key == &size`

`>`

`<`

`==`

Not enough information

(c) `map == key`

`>`

`<`

`==`

Not enough information

(d) `value == &zero`

`>`

`<`

`==`

Not enough information

♡ ...



Jero Wang ADMIN 2y #880da

`size` is in static, while `key` is on the heap, so the heap address would be larger.

`value` is in `main`, while `zero` is in `add_entry`, which is called by `main`, so `value`'s stack address will be higher since the stack grows downwards.

♡ 1 ...



Anonymous Armadillo 2y #880cc

✓ Resolved

Sp19-MT1-2bc

Would it be possible to explain the answers for 2b and 2c? Thanks!

Problem 2 Remember-y Management**(18 points)**

For this problem, assume all pointers are four bytes and all characters are one byte. Consider the following C code (all the necessary #includes are omitted). C structs are properly aligned in memory and all calls to malloc succeed.

```
int size = 0;
struct map_entry {
    char *key;
    char *value;
};

void add_entry(struct map_entry *m, char *k, char *v) {
    int *zero = NULL;
    m[size].key = k;
    m[size].value = v;
    size++;
}

void main(void) {
    struct map_entry *map = malloc(sizeof(struct map_entry) * 10);
    char *key = malloc(sizeof(char) * 10);
    char value[20];
    add_entry(map, "k", "v");
    add_entry(map, key, value);
}
```

(a) For each of the following, bubble the option that best describes where in the memory layout each variable is stored **You should select one answer per variable.**

(a) zero

- | | |
|--|------------------------------|
| <input checked="" type="radio"/> Stack | <input type="radio"/> Static |
| <input type="radio"/> Heap | <input type="radio"/> Code |

(b) *map[0].key

- | | |
|-----------------------------|---|
| <input type="radio"/> Stack | <input checked="" type="radio"/> Static |
| <input type="radio"/> Heap | <input type="radio"/> Code |

(c) map[1].value

- | | |
|---------------------------------------|------------------------------|
| <input type="radio"/> Stack | <input type="radio"/> Static |
| <input checked="" type="radio"/> Heap | <input type="radio"/> Code |

♡ ...

J Jero Wang ADMIN 2y #880dd

zero is on the stack because it is a local variable.

*map[0].key is referring to the first character of the string "v", so it is in static memory.


map[1].value is referring to where the value pointer is stored in the struct map_entry, which is allocated on the heap.

♡ ...

J Anonymous Armadillo 2y #880de

Hi, thanks for the response. Why does *map[0].key refer to the first character of the string "v"?

♡ ...

 **Anonymous Armadillo** 2y #880ca ✓ Resolved
Sp19-MT2-1c

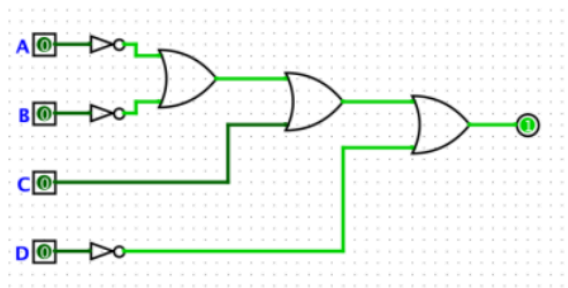
Why is a not a correct choice in this case?

(c) Now we'll work with a new expression:

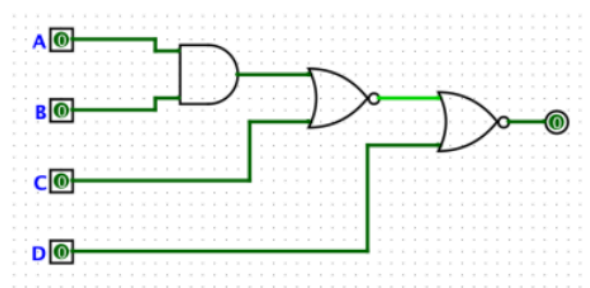
$$\overline{\overline{AB + C}D} = (\overline{AB} + C) + \overline{D}$$

$$= ((\overline{A} + \overline{B}) + C) + \overline{D}$$

Select the correctly simplified circuit.



(a)




(b)

♡ ...

 **Sam Xu** TUTOR 2y #880ce

A is correct in logic, but is not simplified. Choice A uses 6 gates (3 OR gates, three Inverters), however, choice C used only 3 gates (2 NOR gates, 1NAND gate). Besides number of gates used, we usually consider NOR/NAND gates are more simple than OR/AND gates. (In hardware, NOR/NAND are faster than OR/AND)

♡ 1 ...

 **Anonymous Newt** 2y #880bf ✓ Resolved
FA19-MT2-Q5B

$$out = \overline{A} + \overline{(B(A + C + \overline{A}C))}$$

Apply distributive law

$$out = \overline{A} + \overline{(B)(A + C)}$$

Apply absorption law

Just to make sure I'm understanding this step - the absorption law is being applied to $C + \overline{A}C \rightarrow C$ in the second term?

♡ ...



E Erik Yang TA 2y #880cb

I'm not sure what absorption law is but, $C(1+ !A) = C(1) = C$



Anonymous Magpie 2y #880bd

✓ Resolved

SP18-MT1-Q2A

Why can we "return start"? Because start is a pointer, doesn't it mean it is stored on the stack? I thought that we cannot return values from the stack.

```
Solution: list_node * find_end(list_node *start) {
    if (start == NULL) {
        return NULL;
    }
    while (start->next != NULL) {
        start = start->next;
    }
    return start;
}
```



E Erik Yang TA 2y #880be

This is different because the start is passed in as an argument, so you can return the pointer. When a function returns, all objects inside stack get deallocated and wiped away, but since start was declared outside the function, it won't do that here



Anonymous Hippopotamus 2y #880bb

✓ Resolved

SP18-MT1-Q3

for part b, why would print statement 2 not print?

```

typedef struct Song {
    char *title;
    char *artist;
} Song;

Song * createSong() {
    Song* song = (Song*) malloc(sizeof(Song));
    song->title = "this old dog";
    char artist[100] = "mac demarco";
    song->artist = artist;
    return song;
}

int main(int argc, char **argv) {
    Song *song1 = createSong();
    printf("%s\n", "Song written:");
    printf("%s\n", song1->title); // print statement #1
    printf("%s\n", song1->artist); // print statement #2

    Song song2;
    song2.title = malloc(sizeof(char)*100);
    strcpy(song2.title, song1->title);
    song2.artist = "MAC DEMARCO";
    printf("%s\n", "Song written:");
    printf("%s\n", song2.title); // print statement #3
    printf("%s\n", song2.artist); // print statement #4

    return 0;
}

```

♡ ...

 **Nikhil Kandkur** TA 2y #880bc

The value stored in `artist` is a value that was created on the stack. Because we cannot assume that stack address can stay the same in a function that was just called, nothing (or garbage) would print. On the other hand, if we had dynamically allocated the string or statically declared the string like `song->title`, our code would work.

♡ ...

 **Anonymous Armadillo** 2y #880ae ✓ Resolved

FA19-MT-Q1

Part c.) What's the formula for smallest step size? Thanks!

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

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

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

Part A

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

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

Part B

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

Part C

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

♡ ...

E Eric Kusananto TA 2y #880ba

Step size isn't explicitly a part of this class anymore, but the formula would be

$$\text{step size} = 2^{-n} \cdot 2^{\text{exp}-\text{bias}}$$

where n = number of mantissa bits.

♡ 1 ...

Anonymous Sheep 2y #880ac ✓ Resolved

FA19-MT-Q6

I wasn't sure if this was in scope of the exam. It just is totally unrecognizable.

♡ ...

Anonymous Armadillo 2y #880ad

I don't think it should be in scope since according to the MT logistics page only up to Lecture 17 is in scope. RISC-V Architecture is covered in Lecture 18+. Hope a staff member could confirm though, since I also saw this problem when doing the exam earlier

♡ ...

E Erik Yang TA 2y #880af

not in scope

func3. So, we will have 7 bits for the opcode (which is the same as a normal jal instruction), 6 bits for the register field (compared to 5), which leaves us with 19 bits for the immediate.

♡ ...

 Anonymous Armadillo 2y #880f

Thank you!

♡ ...

 Anonymous Armadillo 2y #880a ✓ Resolved

FA19-MT-Q7

I'm having trouble understanding this question. What does it mean to get the value during decode? Also what does it mean to "write back the value of the previous instruction"? Also, what topic does this fall under so I can study further? Thank you!

Q7) RISC-V Exam-isim Debug – Pipelined (18 pts = 3 + 6 + 3 + 6)

After solving your datapath bug, you decide to introduce the traditional five-stage pipeline into your processor. You find that your unit tests with single commands work for all instructions, and write some test patterns with multiple instructions. After running the test suite, the following cases fail. You should assume registers are initialized to 0, the error condition is calculated in the fetch stage, and no forwarding is currently implemented.

Case 1: Assume the address of an array with all different values is stored in `s0`.

```
addi t0 x0 1
slli t1 t0 2
add t1 s0 t1
lw t2 0(t1)
```

Each time you run this test, there is the same incorrect output for `t2`. All the commands work individually on the single-stage pipeline.

Pro tip: you shouldn't even need to understand what the code does to answer this.

- | | |
|--|--|
| a) What caused the failure?
(select ONE) | b) How could you fix it? (select all that apply) |
| <input type="radio"/> Control Hazard | <input checked="" type="checkbox"/> Insert a nop 3 times if you detect this specific error condition |
| <input type="radio"/> Structural Hazard | <input type="checkbox"/> Forward execute to write back if you detect this specific error condition |
| <input checked="" type="radio"/> Data Hazard | <input type="checkbox"/> Forward execute to memory if you detect this specific error condition |
| <input type="radio"/> None of the above | <input checked="" type="checkbox"/> Forward execute to execute if you detect this specific error condition |
| | <input type="checkbox"/> Flush the pipeline if you detect this specific error condition |

The issue with the above code is the use of a register (aka we get the value during the decode phase) before we have written back the value of the previous instruction. This is a data hazard as the data which we want is not restored to the regfile. This means that we would have the current instruction in the execute phase while we have the previous in decode. This means that the next cycle, we would have to forward the execute output to the execute input to make sure the value is the correct, updated one. Inserting a nop when you realize this error happens will allow the system to do the write back. The other forwards in this problem are necessary for the given code above. Flushing the pipeline does not work as it means that we will no longer execute the instructions which were flushed. This means we would just drop instructions which would not get the correct value instead of just waiting till they can get the correct value.

♡ ...

 Erik Yang TA 2y #880b

pipelining and hazards are something you'll learn in a future lecture, i don't think this is in scope for this midterm yet

♡ ...

 Anonymous Armadillo 2y #880c

Got it, thanks!

♡ ...