You are viewing this thread in readonly mode.

[Final] Past Exams - 2021 #986

E

Eric Che STAFF
7 months ago in Exam – Final

1,335
VIEWS

3

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 (if available), are also linked on that page!

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

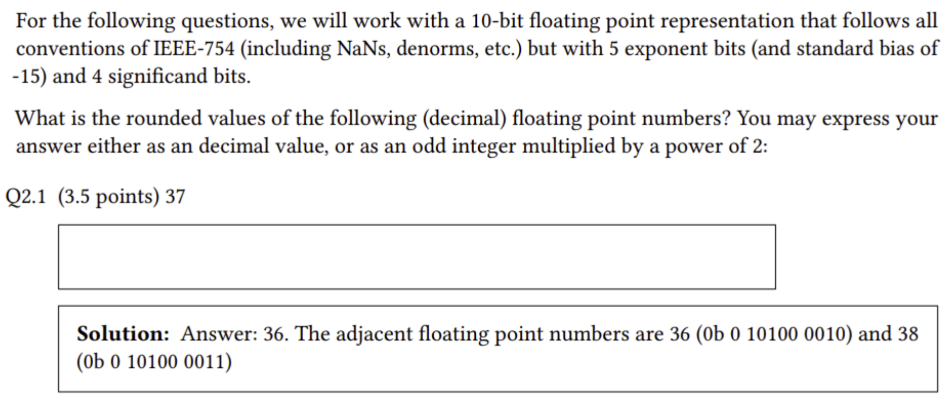
For example: **SP22-Final-Q1**, or **SU22-MT-Q3**

As a note, some questions will be out of scope because set-associative caches are out of scope this semester.

Anonymous Wombat 7mth #986dcd ✓ Resolved

FA21-Final-Q2.1

How exactly do we figure out that 36 and 34 are our options for rounding 37?



For the following questions, we will work with a 10-bit floating point representation that follows all conventions of IEEE-754 (including NaNs, denorms, etc.) but with 5 exponent bits (and standard bias of -15) and 4 significand bits.

What is the rounded values of the following (decimal) floating point numbers? You may express your answer either as an decimal value, or as an odd integer multiplied by a power of 2:

Q2.1 (3.5 points) 37

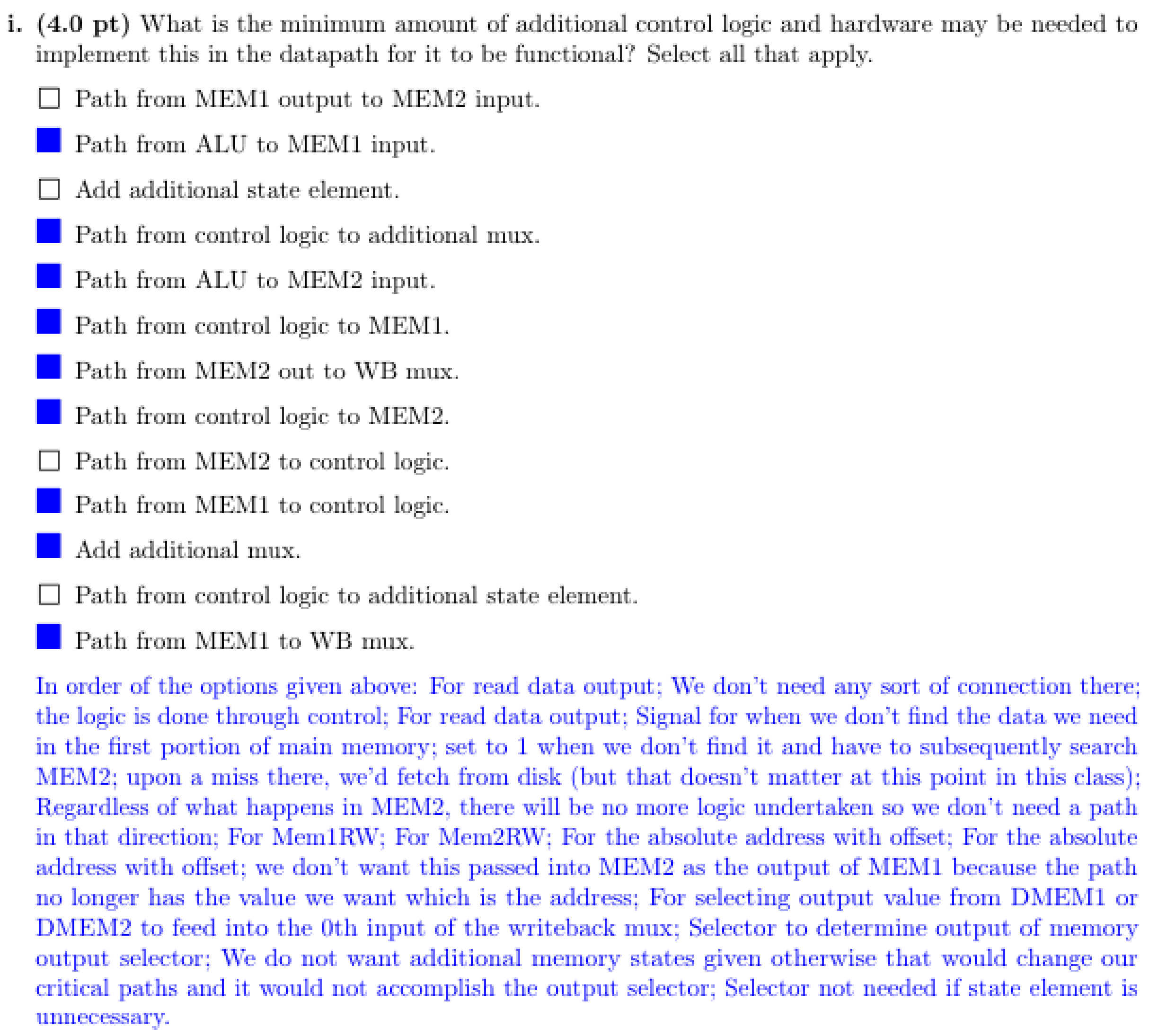
Solution: Answer: 36. The adjacent floating point numbers are 36 (0b 0 10100 0010) and 38 (0b 0 10100 0011)

How do we convert numbers to minifloat representations in general? Do we just truncate the bits that we can't fit into the significand?

edit: I guess you can just figure out that 37 will necessarily include 2^5, and that a mantissa of 0001 corresponds to 34. 2^5 * 2^-4 = 2, so the smallest possible step size is 2

Anonymous Dove 7mth #986dcb ✓ Resolved

sp21-mt-q5b link



i. **(4.0 pt)** What is the minimum amount of additional control logic and hardware may be needed to implement this in the datapath for it to be functional? Select all that apply.

☐ Path from MEM1 output to MEM2 input.

■ Path from ALU to MEM1 input.

☐ Add additional state element.

■ Path from control logic to additional mux.

■ Path from ALU to MEM2 input.

■ Path from control logic to MEM1.

■ Path from MEM2 out to WB mux.

■ Path from control logic to MEM2.

☐ Path from MEM2 to control logic.

■ Path from MEM1 to control logic.

■ Add additional mux.

☐ Path from control logic to additional state element.

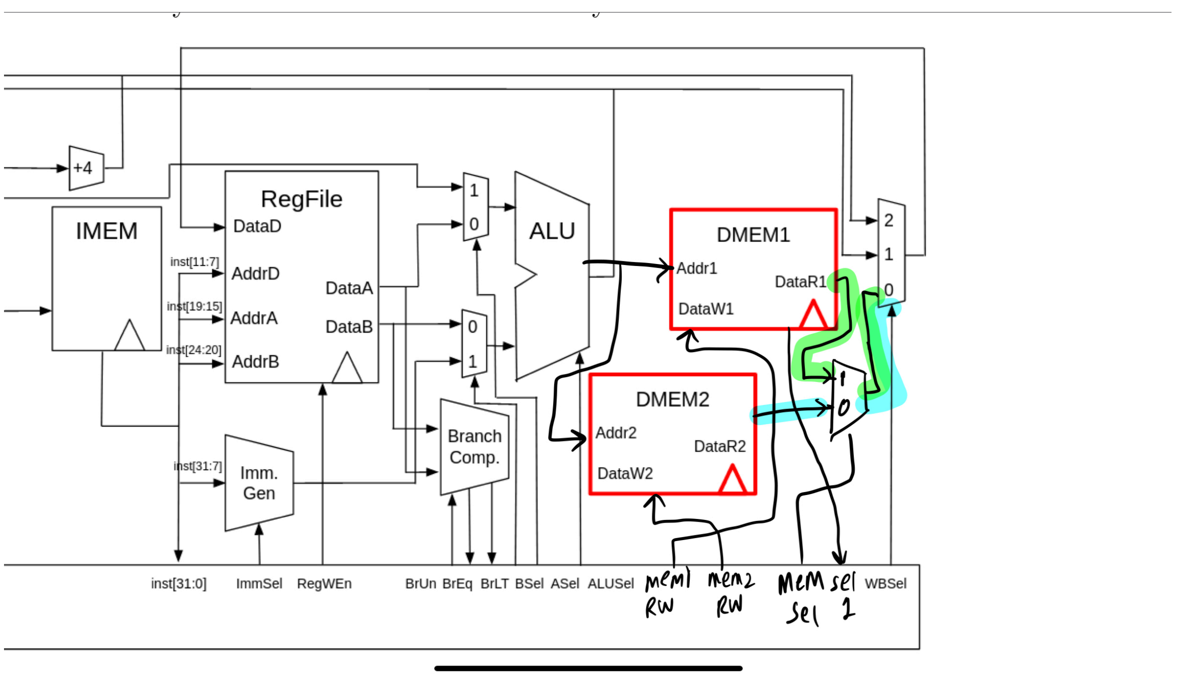
■ Path from MEM1 to WB mux.

In order of the options given above: For read data output; We don't need any sort of connection there; the logic is done through control; For read data output; Signal for when we don't find the data we need in the first portion of main memory; set to 1 when we don't find it and have to subsequently search MEM2; upon a miss there, we'd fetch from disk (but that doesn't matter at this point in this class); Regardless of what happens in MEM2, there will be no more logic undertaken so we don't need a path in that direction; For Mem1RW; For Mem2RW; For the absolute address with offset; For the absolute address with offset; we don't want this passed into MEM2 as the output of MEM1 because the path no longer has the value we want which is the address; For selecting output value from DMEM1 or DMEM2 to feed into the 0th input of the writeback mux; Selector to determine output of memory output selector; We do not want additional memory states given otherwise that would change our critical paths and it would not accomplish the output selector; Selector not needed if state element is unnecessary.

I'm very confused by this explanation. Is there a diagram to show where the additional mux is? If it's to choose between the output of MEM1 and MEM2, then why are there paths from MEM1 and MEM2 directly to WB mux? And why do the paths from MEM1 to control logic and from control logic to MEM1 both exist?

S Sasha Singh STAFF 7mth #986ddf

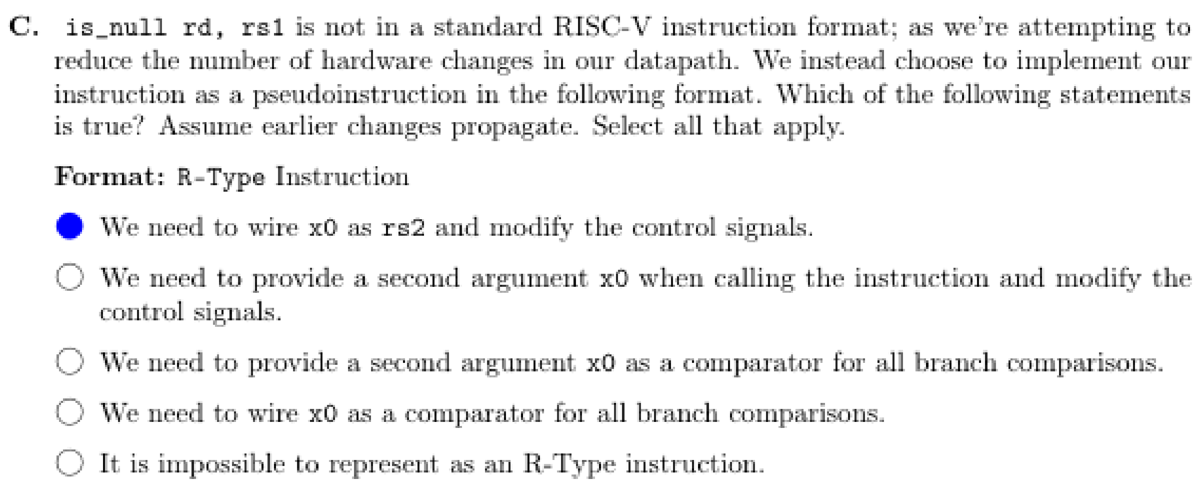
great question! like you correctly pointed out, the additional mux is to choose between the output of mem1 and mem2, but we would still say we need we need to add paths from mem1 to the wbmux and mem2 to the wbmux and these paths would pass *through* the additional mux (path from mem1 to wbmux highlighted in green and mem2 to wbmux highlighted in blue)



1 ···

Anonymous Dove 7mth #986dbd ✓ Resolved

sp21-final-q7AiC



C. `is_null rd, rs1` is not in a standard RISC-V instruction format; as we're attempting to reduce the number of hardware changes in our datapath. We instead choose to implement our instruction as a pseudoinstruction in the following format. Which of the following statements is true? Assume earlier changes propagate. Select all that apply.

Format: R-Type Instruction

- [✓ We need to wire `x0` as `rs2` and modify the control signals.]
- We need to provide a second argument `x0` when calling the instruction and modify the control signals.
- We need to provide a second argument `x0` as a comparator for all branch comparisons.
- We need to wire `x0` as a comparator for all branch comparisons.
- It is impossible to represent as an R-Type instruction.

Why isn't option 3 correct? I thought it was doing the same thing as option 1.

···

M Minh Nguyen STAFF 7mth #986ddc

Option 3 is incorrect because it generalizes the functionality of the `is_null` pseudo-instruction into a requirement for that applies for all branch comparisons.

···

Anonymous Mantis 7mth #986cff ✓ Resolved

sp21-midterm-Q5c, d

why is the number of NOPs needed 10?

and how do we calculate the NOPs needed for part d?

···

Anonymous Octopus 7mth #986dcc

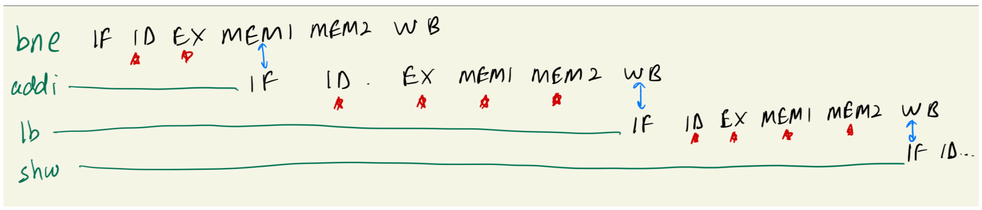
yes, same question, i calculated 11.

Does it have to do with the fact that shw and lb dont use the second DMEM?

S

Sasha Singh STAFF 7mth #986dec

these are great questions!
-> for the bne, the right value gets written to the PC in the EX stage (since there is a direct path from the output of the ALU to the PC register), so we need to make sure the addi starts after the EX stage to make sure we have the right PC value during IF which gives us two NOPS between bne and addi
-> the lb can only start its ID stage after t1 has the right value (so after the WB of addi is complete), this gives us 4 NOPS between addi and lb
-> the shw can only start its ID stage after s0 has the right value (so after the WB of lb is complete), this gives us 4 NOPS between lb and shw



Anonymous Seahorse 7mth #986ced ✓ Resolved

FA21-Final-Q7.4

For the last one, when there is a tag in the cache but isn't valid, what type of miss should it be?

1

A

Abhinav Vedati STAFF 7mth #986daa

Edit: we don't know, because we don't know why the data is currently invalid. If the data at the address has been loaded into the cache before, and then its entry was invalidated for some reason, then this is a conflict miss. Otherwise, it's a compulsory miss.

Anonymous Opossum 7mth #986cec ✓ Resolved

SP21-Final-Q6

How many bytes is a half word? I though a half word was 2 bytes, but then in term of being half word aligned, wouldn't the structure size always be aligned?

A

Abhinav Vedati STAFF 7mth #986dab

On a 32-bit system, a half word is 2 bytes (16 bits). Where does it say that the structure or its size has to be half word aligned?

Anonymous Opossum 7mth #986dbc

In the walkthrough video, the TA explained that if the struct was 11 bytes, we would have to round up to 12?

A

Abhinav Vedati STAFF 7mth #986dbf

Replying to Anonymous Opossum

Ah, I see. That's because C wants structs to be half-word aligned (as mentioned in the video), so the structure size might not always be half-world aligned. If the structure is an even number of bytes in size, its size is aligned. If its an odd number of bytes in size, its size is not aligned, and we need an extra byte of padding.

♡ 1 ···

A Abhinav Vedati STAFF 7mth #986dca

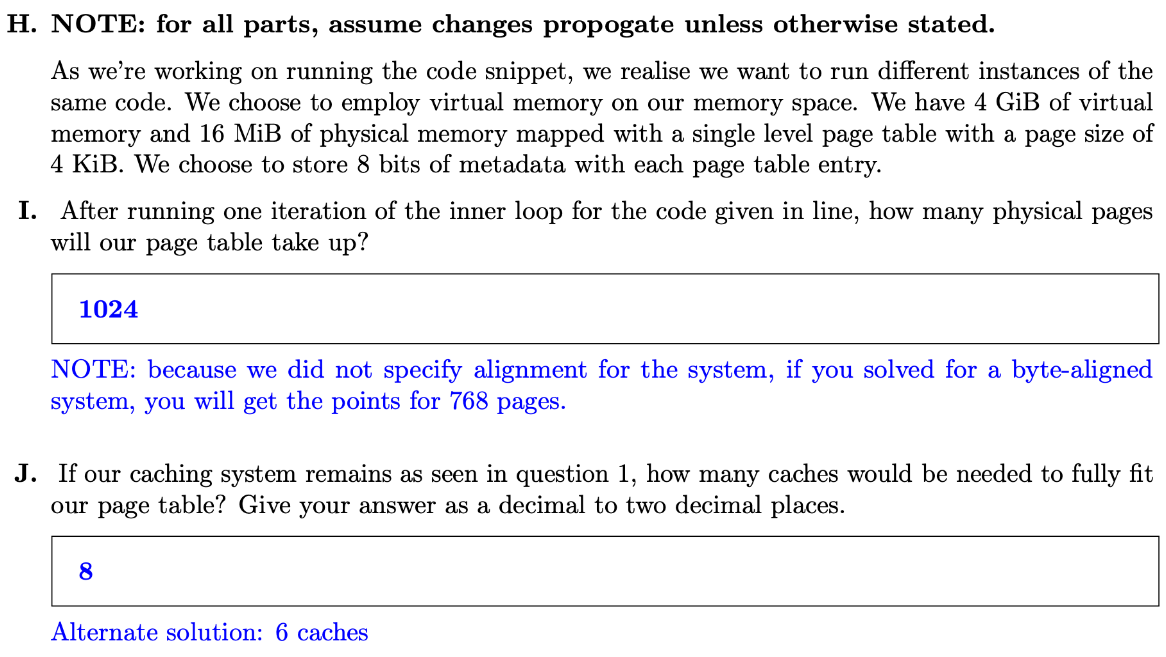
Replying to Abhinav Vedati

So for example, if we had `struct bar { char c; }` , then the size of the one element of the struct is 1 byte. However, following half-word (2 byte, in this case) alignment rules, we need an extra byte of padding such that `sizeof(struct bar) == 2` .

♡ 1 ···

Anonymous Wolverine 7mth #986ceb ✓ Resolved

SP21-Final=Q2 H.



H. NOTE: for all parts, assume changes propogate unless otherwise stated.

As we're working on running the code snippet, we realise we want to run different instances of the same code. We choose to employ virtual memory on our memory space. We have 4 GiB of virtual memory and 16 MiB of physical memory mapped with a single level page table with a page size of 4 KiB. We choose to store 8 bits of metadata with each page table entry.

I. After running one iteration of the inner loop for the code given in line, how many physical pages will our page table take up?

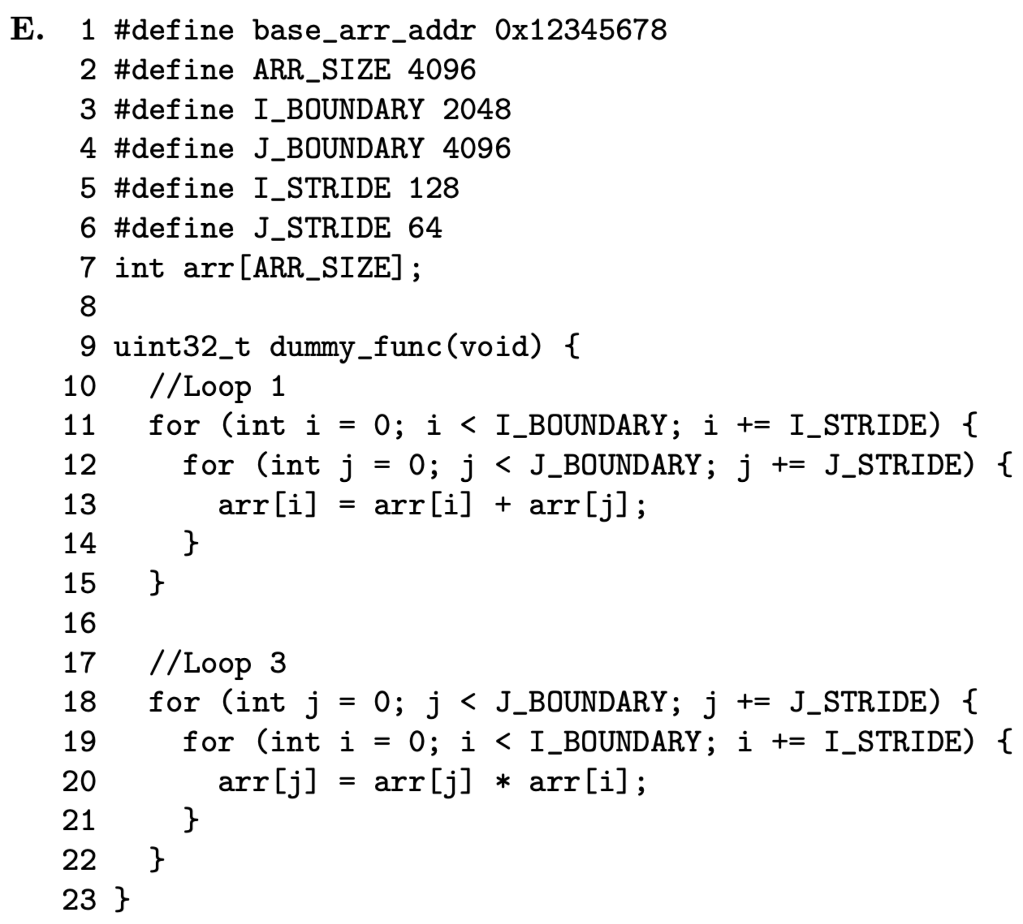
1024

NOTE: because we did not specify alignment for the system, if you solved for a byte-aligned system, you will get the points for 768 pages.

J. If our caching system remains as seen in question 1, how many caches would be needed to fully fit our page table? Give your answer as a decimal to two decimal places.

8

Alternate solution: 6 caches



E.

```
#define base_arr_addr 0x12345678
#define ARR_SIZE 4096
#define I_BOUNDARY 2048
#define J_BOUNDARY 4096
#define I_STRIDE 128
#define J_STRIDE 64
int arr[ARR_SIZE];

uint32_t dummy_func(void) {
  //Loop 1
  for (int i = 0; i < I_BOUNDARY; i += I_STRIDE) {
    for (int j = 0; j < J_BOUNDARY; j += J_STRIDE) {
      arr[i] = arr[i] + arr[j];
    }
  }

  //Loop 3
  for (int j = 0; j < J_BOUNDARY; j += J_STRIDE) {
    for (int i = 0; i < I_BOUNDARY; i += I_STRIDE) {
      arr[j] = arr[j] * arr[i];
    }
  }
}
```

why there's 1024 pages? If the array have size 4096 and each element is an int, then there's total 2^14 bytes, each page size is 2^12, then total pages would be 2^14/2^12 = 4?

Anthony Salinas Suarez STAFF 7mth #986dac

- Page Size is 4 KiB = 2^{12} Bytes
- Physical Memory is 16 MiB = 2^{24} Bytes
- Physical Memory is 4 GiB = 2^{32} Bytes
- Offset is $\log_2$(Page Size in Bytes) = $\log_2(2^{12})$ = 12 bits
- Number of bits in Physical Address is $\log_2$(Physical Memory in Bytes) = $\log_2(2^{24})$ = 24
- Number of bits in Virtual Address is $\log_2$(Virtual Memory in Bytes) = $\log_2(2^{32})$ = 32
- So PPN is therefore 24 - 12 = 12 bits
- VPN is 32 - 12 = 20 bits

Now using this information, recall that under normal circumstances, the size of each Page Table Entry (PTE) will be equal to the (number of bits of our PPN) + (number of bits for Metadata) and given we have 8 bits worth of Metadata then 12 + 8 = 20 bits. **Note however, that under the assumption of a word aligned system, we round up to the nearest multiple of 4 bytes so 20 --> 32 bits --> 4 bytes.**

Now because the Page Table maps virtual pages to physical pages and because we found that we have 20 bits making up our VPN, then this means we have 2^{20} virtual pages and therefore 2^{20} PTE's. Given that each PTE is 4 bytes in size then this means the size of our Page Table is $2^{20} * 2^2 = 2^{22}$ bytes.

Last but not least, we have that each page is of size 2^{12} bytes. So our page table takes up (2^{22} / 2^{12}) --> 2^{10} --> 1024 physical pages in memory.

Note: There is also a nice walkthrough video available on the course site under the exam resources page which I will link here: https://cs61c.org/sp24/resources/exams/ and if you are interested in watching the walkthrough for this portion, the video discusses this problem around the 1 hour 50 minute mark!

♡ 1 ···

E Eric Che STAFF 7mth #986dad Visible to staff only

anthony master chef

♡ ···

A Anthony Salinas Suarez STAFF 7mth #986dae Visible to staff only

Replying to Eric Che

I think this is one of, if not my first ever Ed response. I had to make a statement LMAO

♡ ···

Anonymous Spoonbill 7mth #986cdf Unresolved

is Page table alignment in scope?

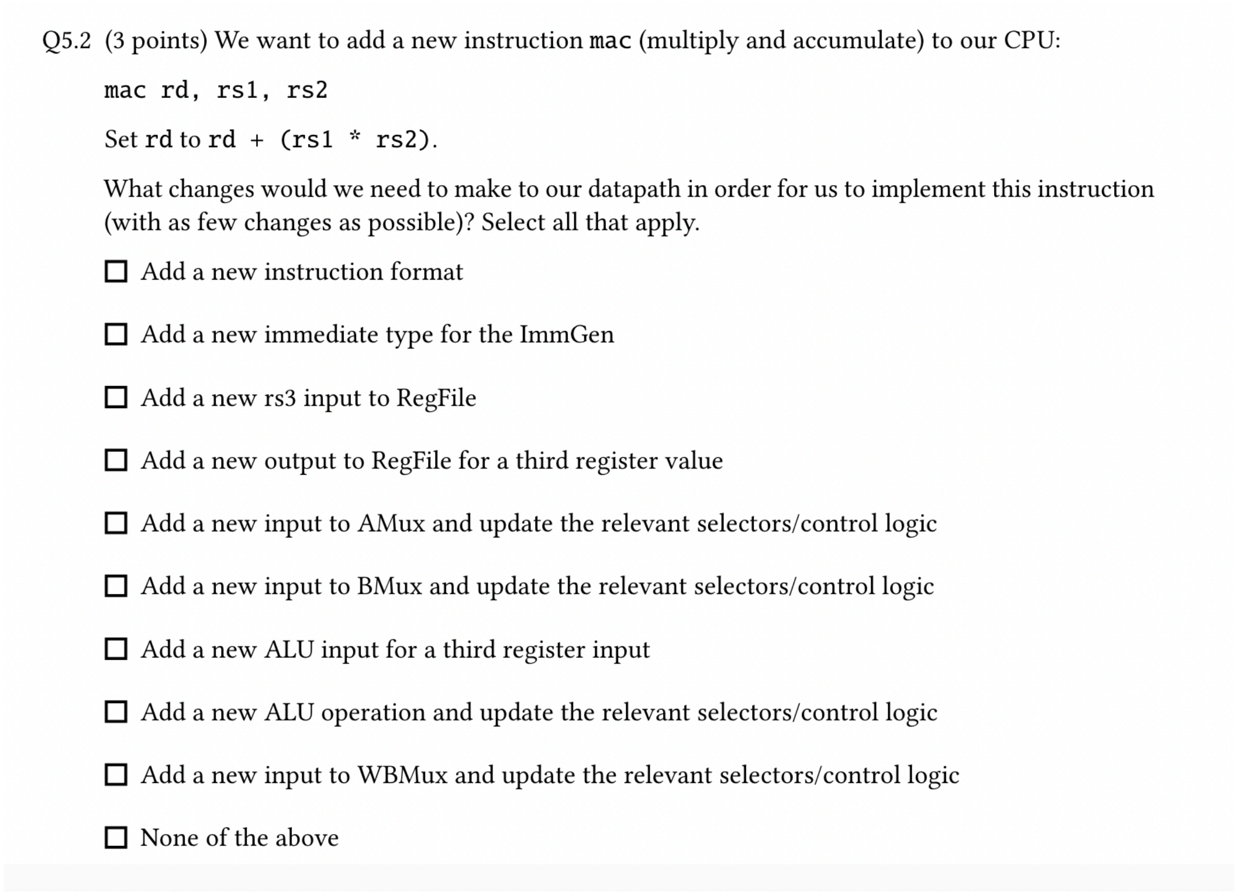
In SP21-Final-Q2aH the solution says to round up the PTE bits from 20 to 32 assuming word alignment. Do we need to know this and what other forms of alignment are there?

Also what about RISC-V alignment? Did we learn about how to store structs like in SP21-Final-Q6a

♡ ···

Anonymous Rabbit 7mth #986cde ✓ Resolved

This is somewhat a general question because I've seen questions like this on a lot of the past exams, but I've been having a lot of trouble with questions like this that present a new instruction and ask how the datapath should be changed. For each answer choice, what are some telltale signs from the new instruction that point to the answer choice needing to be changed? I hope that makes sense. Thanks!


Q5.2 (3 points) We want to add a new instruction `mac` (multiply and accumulate) to our CPU:

`mac rd, rs1, rs2`

Set `rd` to `rd + (rs1 * rs2)`.

What changes would we need to make to our datapath in order for us to implement this instruction (with as few changes as possible)? Select all that apply.

- ☐ Add a new instruction format
- ☐ Add a new immediate type for the ImmGen
- ☐ Add a new rs3 input to RegFile
- ☐ Add a new output to RegFile for a third register value
- ☐ Add a new input to AMux and update the relevant selectors/control logic
- ☐ Add a new input to BMux and update the relevant selectors/control logic
- ☐ Add a new ALU input for a third register input
- ☐ Add a new ALU operation and update the relevant selectors/control logic
- ☐ Add a new input to WBMux and update the relevant selectors/control logic
- ☐ None of the above

J

Justin Yokota STAFF 7mth #986cee

There's not really a general answer I can give to this. Mostly, I would suggest thinking about how I would use the existing datapath to implement the desired behavior, and what if anything needs to change for me to actually make that change.

Anonymous Mantis 7mth #986cdd ✓ Resolved

Sp21-FInal-Q5b,

in the explanation, does the first part "for read data output" apply to the first chosen choice or just the first choice?

M

Myrah Shah STAFF 7mth #986dce

Can you double check the exam/question #?

Anonymous Mantis 7mth #986cdb ✓ Resolved

Sp21-Final-Q7a, what's the reasoning behind choosing to modify control logic for WBSel, parsing instr[31:0] and branch comparator? Also what's the difference bw modifying control signals vs control logic vs ALU buses for the ALU?

M

Minh Nguyen STAFF 7mth #986ddd

Adjusting the `WBSel` ensures that the correct value (the result of the zero comparison) is written back to `rd` and not the usual `ALU` result or data mem value. The control logic for `instr[31:0]` needs to recognize the new opcode for `is_null` . Modifying the branch comparator = checking if `rs1==0` .

`Control signals` are the binary values that determine the operations of the pipeline components. These are generated by the control logic based on the current instruction being executed. `Control logic` is the circuitry that interprets the instruction to produce the necessary control signals in the data path. `ALU` buses include regfile's inputs to ALU and ALU's output to the rest of the pipeline.

- Changing control signals = changing what each signal does/make new ones
- Changing control logic = changes how instructions are decoded
- Changing ALU buses = changing the path that data travels through to reach ALU and where it goes afterward

1 ...

Anonymous Mantis 7mth #986dde

thanks! but if we wanted to write back to rd in the regfile couldn't we already use the existing 0 control signal in the WBSel?

...

J

Jackson Wei STAFF 7mth #986ded

Replying to Anonymous Mantis

We want the result of the zero comparison to be written back to rd so we would need to modify wb such that it writes the result of the branch comparator instead of ALU, PC+4, or data from mem.

...

Anonymous Mantis 7mth #986cce ✓ Resolved

Fa21-Final-Q5.2,

how can we see if we can reuse an already existing format or not? like in this case, how did we know that we could simply just use the R type? also what exactly do the regreadindex and regwriteindex signify again?

...

Brendan Roberts STAFF 7mth #986dea

One way to tell if we can reuse an existing type is by looking at the inputs. An R type instruction is used when we want our inputs to be two registers. Similarly, an I type is used when we want one register and an immediate.

For this question, we want to add a new operation between three registers where we multiply rs1 and rs2 and add the result to rd. R type instructions have an encoding for rs1, rs2, and rd, so we can just use the R type. An ALU can already perform operations on rs1 and rs2, but to add rs1*rs2 to rd, we need to give the ALU a third input value and add a new rd+(rs1*rs2) operation.

Regreadindex is the index of the register you want to read the value of. We have 32 registers indexed from 0-31, so this index tells us which value to read.

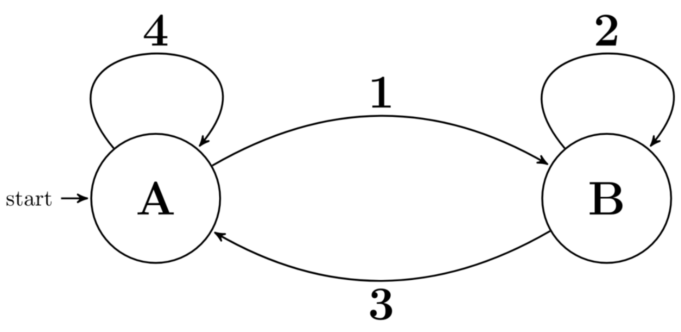
Similarly, regwriteindex is the index of the register you want to write to, and it works in the exact same way as regreadindex.

...

Anonymous Mantis 7mth #986ccd ✓ Resolved

Sp21-Final-Q3:

How should we go about thinking about this FSM? i'm confused as how there can be multiple answers for the transitions. Are these different combinations?



♡ ···

M Myrah Shah STAFF 7mth #986dcf

These are the different possibilities for the current x and y. First, I would think about how we would know if we have overflow when we add 2 numbers. This would be when we have a "carried over" 1 from the addition (0b1 + 0b1 == 0b10). This idea guides the rest of the question.

If we ever encounter two 1 bits, we know we have to carry over a 1. So, B will represent this carried-over state (aka do we currently have a carry-over). This explains why 1,1/1 is our arrow from A to B -- the two 1's will move us to an overflow state, and the 1 is outputted as such.

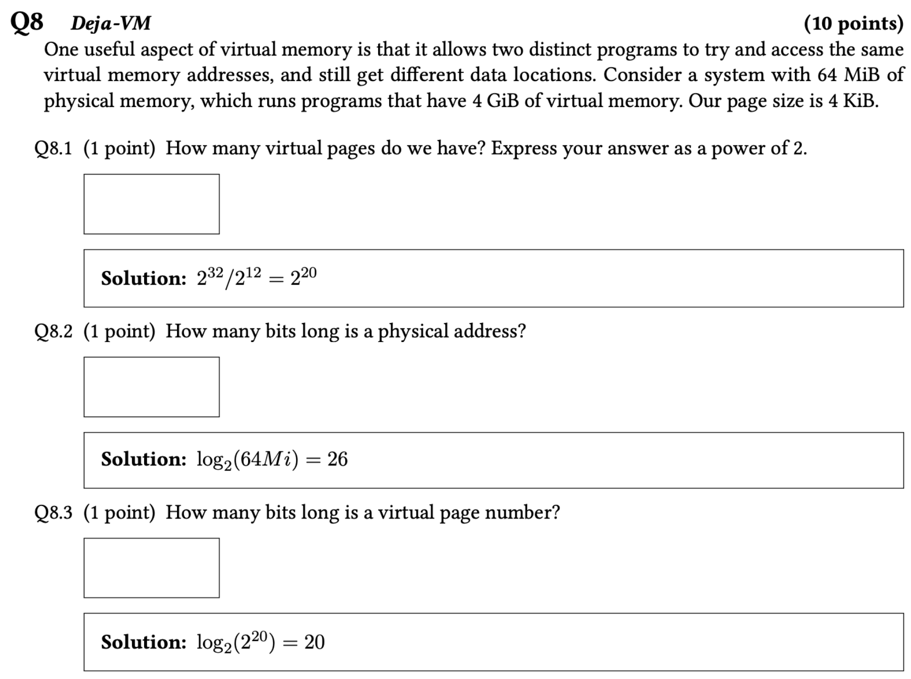
Once we're at B, we know that we have an extra 1 in our addition (the carried over 1), so the possibility to stay in the carried over state would be for (1,0), (0,1) or (1,1) since these would all overflow. Using this, the only way to no longer be in the carried over state is (0,0), since the carried over 1 would equate this to 0+0+1. Once at A, we would stay there as long as there is no overflow, which would be for (1,0), (0,1) or (0,0).

I hope this makes sense, let me know if you have any followups!

♡ 1 ···

Anonymous Buffalo 7mth #986ccb ✓ Resolved

Fa21-Final-Q8.3 How to get the bits of VPN? I thought the offset bits are 20 and virtual memory bits are 32, why isn't the VPN bits 32-20=12?



Q8 ***Deja-VM*** **(10 points)**

One useful aspect of virtual memory is that it allows two distinct programs to try and access the same virtual memory addresses, and still get different data locations. Consider a system with 64 MiB of physical memory, which runs programs that have 4 GiB of virtual memory. Our page size is 4 KiB.

Q8.1 (1 point) How many virtual pages do we have? Express your answer as a power of 2.

Solution: $2^{32}/2^{12} = 2^{20}$

Q8.2 (1 point) How many bits long is a physical address?

Solution: $\log_2(64Mi) = 26$

Q8.3 (1 point) How many bits long is a virtual page number?

Solution: $\log_2(2^{20}) = 20$

A

Anthony Salinas Suarez STAFF 7mth #986dba

Recall:

- The bits for the offset are calculated by the **Page Size**
  - Page Size in Bytes = 4 KiB = 2^2 * 2^{10} = 2^{12} Bytes
  - So offset bits = $\log_2$(Page Size in Bytes) = $\log_2(2^{12})$ = 12 bits
- To get the amount of bits that make up our **Virtual Address,** we do the following:
  - Size of Virtual Memory = 4 GiB = 2^2 * 2^{30} = 2^{32} Bytes
  - Number of bits in Virtual Address = $\log_2$(Size of Virtual Memory in Bytes) = $\log_2(2^{32})$ = 32 bits
- Last but not least, we can use these two pieces of information to get the number of bits for our VPN
  - 12 offset bits
  - 32 virtual address bits
  - VPN bits = (Number of VA bits) - (Number of offset bits) = 32 - 12 = 20 bits

The method followed in the staff solutions utilizes the number of virtual pages to calculate the number of bits that make up the VPN which as you can see still gives the same result!

Anonymous Mantis 7mth #986cca ✓ Resolved

Sp21-Final-Q5

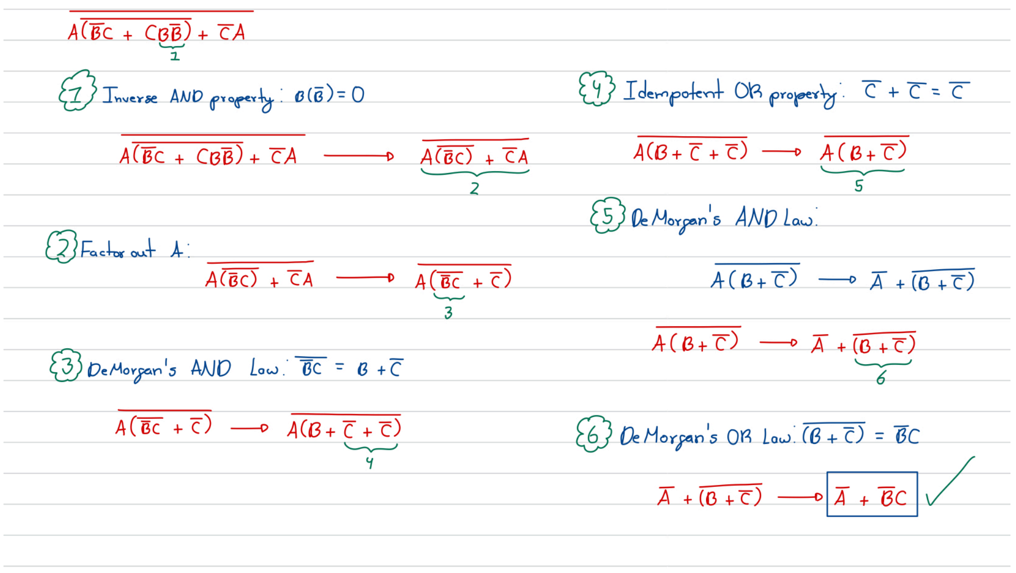
Can anyone show how to simplify choices 1 and 4?

(a) Select all of the following which are equivalent to $\overline{A} + \overline{B}C$.

- ■ $\overline{A(\overline{\overline{B}C + CB\overline{B}}) + \overline{C}A}$
- ☐ $\overline{C\overline{\overline{B}A} + C(\overline{CA} + BC)}$
- ☐ $A\overline{B} + \overline{C}$
- ■ $A\overline{A} + \overline{(ABC + \overline{C}A)}$

A Anthony Salinas Suarez STAFF 7mth #986daf

Hello, below I will attach photos for choices 1 and 4! Do let me know if you have additional questions!

- The photo below is for choice 1
    - 

$\overline{A(\overline{\overline{B}C + C\underbrace{B\overline{B}}_{1}}) + \overline{C}A}$

{1} Inverse AND property: $B(\overline{B}) = 0$

$\overline{A(\overline{\overline{B}C + CB\overline{B}}) + \overline{C}A} \longrightarrow \underbrace{\overline{A(\overline{\overline{B}C}) + \overline{C}A}}_{2}$

{2} Factor out A:

$\overline{A(\overline{\overline{B}C}) + \overline{C}A} \longrightarrow \overline{A(\underbrace{\overline{\overline{B}C}}_{3} + \overline{C})}$

{3} DeMorgan's AND Law: $\overline{\overline{B}C} = B + \overline{C}$

$\overline{A(\overline{\overline{B}C} + \overline{C})} \longrightarrow \overline{A(B + \underbrace{\overline{C} + \overline{C}}_{4})}$

{4} Idempotent OR property: $\overline{C} + \overline{C} = \overline{C}$

$\overline{A(B + \overline{C} + \overline{C})} \longrightarrow \underbrace{\overline{A(B + \overline{C})}}_{5}$

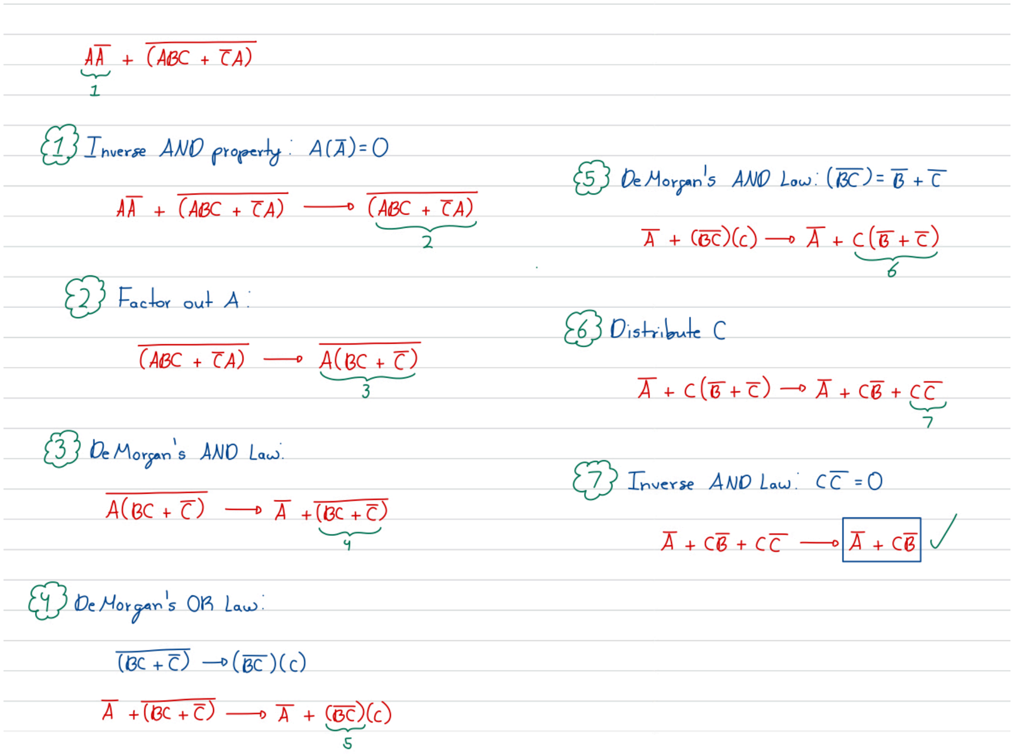
{5} De Morgan's AND Law:

$\overline{A(B + \overline{C})} \longrightarrow \overline{A} + \overline{(B + \overline{C})}$

$\overline{A(B + \overline{C})} \longrightarrow \overline{A} + \underbrace{\overline{(B + \overline{C})}}_{6}$

{6} De Morgan's OR Law: $\overline{(B + \overline{C})} = \overline{B}C$

$\overline{A} + \overline{(B + \overline{C})} \longrightarrow \boxed{\overline{A} + \overline{B}C}$ ✓

- The photo below is for choice 4
    - 

$\underbrace{A\overline{A}}_{1} + \overline{(ABC + \overline{C}A)}$

{1} Inverse AND property: $A(\overline{A}) = 0$

$A\overline{A} + \overline{(ABC + \overline{C}A)} \longrightarrow \underbrace{\overline{(ABC + \overline{C}A)}}_{2}$

{2} Factor out A:

$\overline{(ABC + \overline{C}A)} \longrightarrow \underbrace{\overline{A(BC + \overline{C})}}_{3}$

{3} De Morgan's AND Law:

$\overline{A(BC + \overline{C})} \longrightarrow \overline{A} + \underbrace{\overline{(BC + \overline{C})}}_{4}$

{4} De Morgan's OR Law:

$\overline{(BC + \overline{C})} \longrightarrow (\overline{BC})(c)$

$\overline{A} + \overline{(BC + \overline{C})} \longrightarrow \overline{A} + \underbrace{(\overline{BC})}_{5}(c)$

{5} De Morgan's AND Law: $(\overline{BC}) = \overline{B} + \overline{C}$

$\overline{A} + (\overline{BC})(c) \longrightarrow \overline{A} + \underbrace{c(\overline{B} + \overline{C})}_{6}$

{6} Distribute C

$\overline{A} + c(\overline{B} + \overline{C}) \longrightarrow \overline{A} + C\overline{B} + \underbrace{C\overline{C}}_{7}$

{7} Inverse AND Law: $C\overline{C} = 0$

$\overline{A} + C\overline{B} + C\overline{C} \longrightarrow \boxed{\overline{A} + C\overline{B}}$ ✓

Note: There is a walkthrough for this exam (and this question as well) which is featured on the exam resources page of the official course website which I will link here: https://cs61c.org/sp24/resources/exams/. If you scroll down to to the walkthroughs section and are interested in watching the walkthrough for this exam, the video covers this question at around the 31 minute mark!

Anonymous Mantis 7mth #986cbe ✓ Resolved

SP21-FInal-Q4:

why doesn't making the clock speed faster result in a decrease in the critical path? also why wouldn't transistors decrease the critical path?

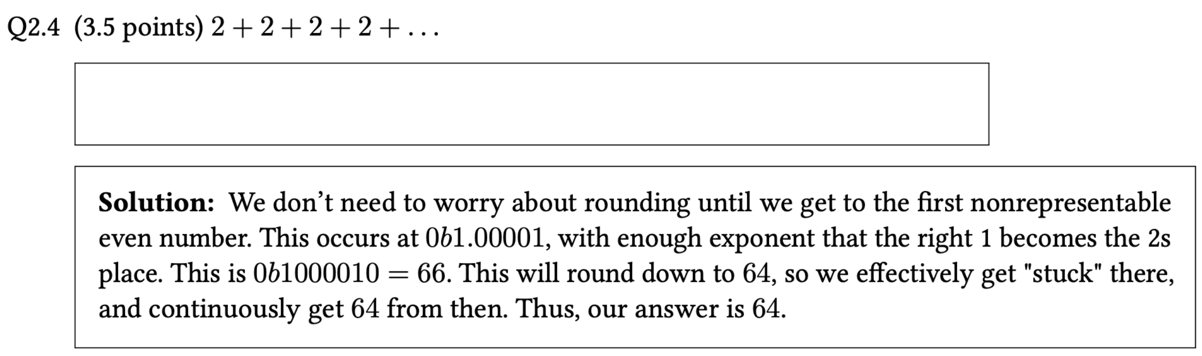
Brendan Roberts STAFF 7mth #986deb

Increasing the clock frequency won't change the critical path. This is because the critical path is determined by the delay of each component on the path. We are then limited by the critical path with how fast we can make the clock because we have to make sure that every operation on the path can finish before the next rising edge of the clock. If we make the clock faster, the critical path will still be the same, but you will now probably have setup time violations since signals won't be ready in time for a register to read them.

Using transistors instead of logic gates won't work because then our circuit won't be able to do logic operations. Logic gates are made of many transistors plugged together in such a way that you can compare signals and do logical AND, OR, and NOT. If we replaced them with individual transistors, this wouldn't be possible.

Anonymous Ibis 7mth #986cba ✓ Resolved

FA21-Final-Q2.4



Q2.4 (3.5 points) $2 + 2 + 2 + 2 + \ldots$

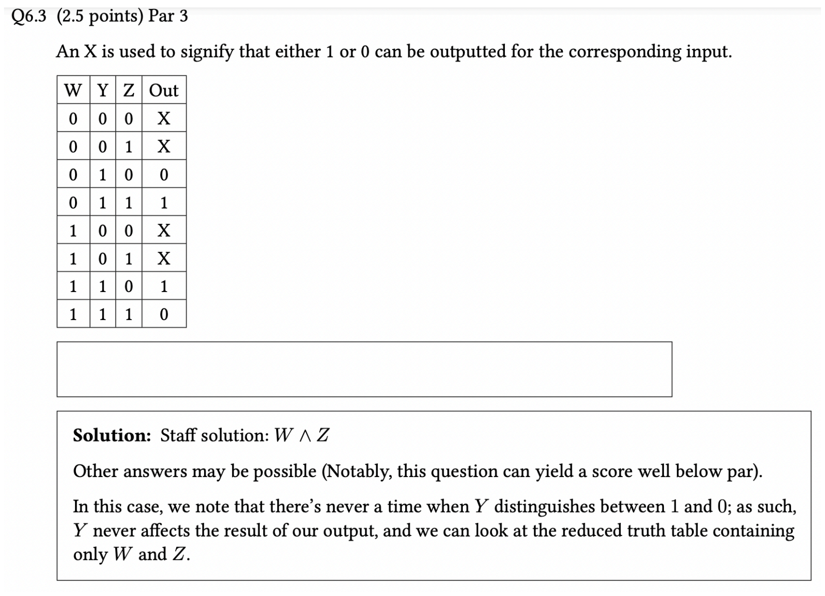
Solution: We don't need to worry about rounding until we get to the first nonrepresentable even number. This occurs at $0b1.00001$, with enough exponent that the right 1 becomes the 2s place. This is $0b1000010 = 66$. This will round down to 64, so we effectively get "stuck" there, and continuously get 64 from then. Thus, our answer is 64.

For this problem, how did they know that 66 is the first non representable even number?

Jackson Wei STAFF 7mth #986dee

Given our setup, 10 bit floating point 5 exponent bits 4 mantissa. They know that 66 is the first non representable even number by leveraging the fact that we only have 4 mantissa fields. If you take a look at 0b1.00001, you can see that this just exceeds our mantissa limit of 4. Then all they need to do is to shift the binary point until the 1 on the right is at the 2s place (we want a even number). After this, we get 0b1000010, which we can't represent so it shifts down to 0b1000000

Anonymous Cod 7mth #986bfc ✓ Resolved



Q6.3 (2.5 points) Par 3

An X is used to signify that either 1 or 0 can be outputted for the corresponding input.

W	Y	Z	Out
0	0	0	X
0	0	1	X
0	1	0	0
0	1	1	1
1	0	0	X
1	0	1	X
1	1	0	1
1	1	1	0

Solution: Staff solution: $W \wedge Z$

Other answers may be possible (Notably, this question can yield a score well below par).

In this case, we note that there's never a time when Y distinguishes between 1 and 0; as such, Y never affects the result of our output, and we can look at the reduced truth table containing only W and Z.

FA21-Final-Q6.3

what does the solution mean when it says that there's never a time when Y distinguishes between 1 and 0 so we can ignore it? does this mean that we would have `~WZ + W ~Z` since those are the rows where the truth table = 1, which equals one because it's essentially in the form of `~X + X` ?

Erik Yang STAFF 7mth #986bfd

If we break down the truth table you get: ~WYZ + WY~Z = 1. Note how when Y is 0, the answer doesn't really matter, so you can factor out Y, and since Y is always 1, it won't affect the output: Y(~WZ + W~Z). Then, the equation inside the parentheses is just W XOR Z.

Anonymous Cod 7mth #986caa

for the equation inside the parenthesis, if we treat ~WZ as a single value X, then would W~Z be equal to ~X? why would this not equal X + ~X which is = 1?

Erik Yang STAFF 7mth #986cac

Replying to Anonymous Cod

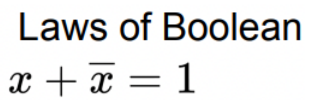
Here's an example: if W = 1 and Z = 0, then you get 0 * 0 + 0 * 1 = 0. It is not always the case that the inside equation equals 1.

Anonymous Cod 7mth #986cae

Replying to Erik Yang

if W = 1 and Z = 0, the expression would be 0 * 0 + 1 * 1 = 1? since ~W = 0 and we have ~WZ which is 0*0 plus W~Z which is 1*0?

does this law from the reference card apply here



Laws of Boolean

$$x + \overline{x} = 1$$

Erik Yang STAFF 7mth #986caf

Replying to Anonymous Cod

Oops sorry i meant W = 1, Z = 1; where you get 0 * 1 + 1 * 0, which equals 0.

Anonymous Cod 7mth #986cbb

Replying to Erik Yang

oh is the reason why we can't apply the identity of ~x + x = 1 because if we treat ~WZ as a single value x, then ~x = ~(~WZ) = (~~W) + (~Z) = W+(~Z) which is not the same as W~Z which is the other term in the expression of ~WZ + W~Z?

Erik Yang STAFF 7mth #986ccc

Replying to Anonymous Cod

yeah that makes sense

Anonymous Spoonbill 7mth #986bfb ✓ Resolved

FA21-Final-Q6.3 can we get a few alternative answers for this problem that still got full points. There's a lot of ways to do it so everytime ive done it I get a different answer. I even got 1 once.

Brendan Roberts STAFF 7mth #986def

Par was 3 so any solution that uses at most 3 operators would get full points. An example would be W /\ Z & Y. We have the W /\ Z from the simplest solution, along with a new & Y. This will have no effect on the output as Y is only 0 when W /\ Z is also 0, and Y is 1 every time W /\ Z is 1.

Anonymous Crow 7mth #986bfa ✓ Resolved

FA21-Final-Q7.4.6

What type of miss is this? Also for 7.4.3, how do we know that there is still space in the cache, leading to a compulsory miss?

1

Myrah Shah STAFF 7mth #986dda

If the valid bit is off, we interpret it as if the space is empty. Since capacity misses require that the cache is full, these are not capacity misses.

Anonymous Viper 7mth #986bbb ✓ Resolved

Fall21-Final7.4.2

I wonder why we can have compulsory mis and conflict miss simultaneously? and how do we differentiate between them?

I think compulsory miss happens when we try to request the memory for the very first time?

R Rohan Mathur STAFF 7mth #986bca

You are correct. If the bytes were never in the cache in the first place it is a compulsory miss. If it was in the cache at some point but then was kicked out for some other memory it is a conflict miss.

Anonymous Viper 7mth #986bed

Thanks for the reply. I see from the lecture that being fully associative could not cause conflict miss. However, if the bytes were kicked off before and then missed, should we still consider this as conflict miss in fully associative cache?

Anonymous Jay 7mth #986bba ✓ Resolved

Fall21-Final-5.3

I was wondering if this is a valid solution:

li s1 2

li s2 3

li s3 4

li s4 3

mac s3 s2 s1

and if not, then why? My thinking was that in this, for the mac, the s2 nd s3 would not be done with WB stage before the mac's ex stage. But if mac is replaced with mul, then s2 and s1 would be done with their WB stages before the mul operation.

Candice Yang STAFF 7mth #986beb

```
s2 F | D | X | M | W*
s3     F | D | X | M | W
s4         F | D | X | M | W
mac            F |*D | X | M | W
```

mac reads s2 at the start of `D` stage while s2 is still at `W` stage at that cycle. (see *) Usually we assume in the completely unoptimized pipeline, the value will be ready (s2 finishes writeback) after `W`. So your solution is slightly incorrect.

Anonymous Mantis 7mth #986bac ✓ Resolved

SP21-Midterm-Q7b,

Can someone explain why we got these answers? (why can't we use malloc also?)

}

i. **(3.0 pt)** What should be line A? You don't need a sizeof() because unit8_t is always defined as one byte.

calloc(size / 8 + 1)

ii. **(3.0 pt)** What should be line B?

bf->hashFunc(element, i) % bf->size

iii. **(3.0 pt)** What should be line C?

1 « (n % 8)

R Rohan Mathur STAFF 7mth #986bcd

```
calloc(size / 8 + 1)
```

We have to use calloc because we must 0 out all the memory in data. If we malloc'd, there would be random bits (many of which would be 1) which would make it look like there was more present elements than there actually were.

We divide by 8 and add 1 because we are allocating in `unit8_t` 's which are each **bytes**. Since we are testing if elements are present with one **bit**, the number of **bytes** we need 1/8 the number of bits because there are **8 bits in 1 byte**. We add 1 because the division is floor division so we could need extra space if the number of bits isn't a multiple of 8.

```
bf->hashFunc(element, i) % bf->size
```

This is similar logic to filling out elements in a hash table (not in scope but you can check out this 61b lecture if you're curious). The hash function returns some 64 bit value (which we can consider a massive unsigned integer). So, we mod by size to get a number from 0 to size (exclusive). This will be the data array's index. Remember, this is the **bit** number since size is in **bits**.

For example, if the size is 100, we have 100 bits in the data array. The hash function could return 477. `477 % 100 = 77` , so the 77th bit of the data array would be flipped to 1.

```
1 « (n % 8)
```

`bf->data[n/8]` returns us the **byte** at which our **bit** index is by floor dividing by 8. In the previous example, The **byte** at which the 77th **bit** is stored is 9. That **byte** will store **bits** [72 to 80).

To get the correct **bit** in the **byte**, we can mod by 8. `77%8==5` so we want to make sure the 5th **bit** in that **byte** is set to 1. This also makes sense because 5 **bits** up from 72 is 77. We get the 5th **bit** by left shifting 1 by 5 giving us a **byte** with a 1 in the right place and a 0 everywhere else. `1 << 5 == 00100000`. Can we just set the data **byte** to that?

Not quite.

If we had any other **bits** in that **byte** set to 1, we would wipe them out. We leave the other bits alone with the `...| bf->data[n/8];`. The bitwise `|` (or) will make sure that even if all the other bits in what we set are 0, the current 1s will stay ones.

Anonymous Mantis 7mth #986cfe

thank you so much for the explanation!

So with the different types, uint64_t, uint16_t, uint8_t, this pretty much means that for example if we have uint64_t, this variable of this type stores 64 bits? or bytes? How do we know when to use what? I think I'm a bit confused about why we even need or use these types when we could simply just use int or double?

Anonymous Rook 7mth #986bab ✓ Resolved

FA21-Final-Q5.2, 5.3

For 5.2: I understand R-types encode rd, rs1, and rs2 information but don't we still need to modify RegFile so that it knows to read and output all three of those registers? Why does this not affect the input but only the output for RegFile?

For 5.3: I honestly am confused by the explanation—why do we still need to stall regardless? Why with mul, we don't need to wait for the value of rd to get written back, but we do need to wait for rd for a mac? Don't both writeback to rd?

1

(__m256i *) Jasmine Angle STAFF 7mth #986bbf

5.2: We still have the signal for `rd` that gets passed to the `RegWriteIndex` port, so the RegFile already has everything it needs input-wise without any modifications on the datapath side (it already gets passed in `rd`, `rs1`, and `rs2`). Because of this, we can actually use the `RegWriteIndex` port jointly as a write and read port to read from the `rd` register, though this implementation would happen within the RegFile itself and not within our top-level datapath. Ultimately, we will require a new output port from the RegFile to contain the data from the register indexed by `rd`.

5.3: The key difference here is that our new `mac` instruction now has a data dependency on `rd`. Because of this, there are new opportunities for hazards to appear when using the `mac` instruction. In the example from the solution with two of the same `mac` instructions immediately following one another, the first one writes to `a0`, but since the second one is dependent on the value in `a0` as well, we must stall in order for the first `mac` to complete before the second can get the correct `a0` value. This would not occur with two subsequent `mul` instructions, as the destination register would be overwritten both times and no data dependency exists.

Anonymous Rook 7mth #986baa ✓ Resolved

FA21-Final-Q3.3

Could you also do jal ra a0 (or jr a0) for the 5th blank since the buffer is also stored at a0 (and this doesn't change after calling Get20Chars)?

R Rohan Mathur STAFF 7mth #986bce

`jal ra a0` is not a valid instruction because `jal` jumps to a **label** not a register. It only saves the next instruction in the register.

`jr a0` is a valid instruction but not correct because `a0` only points to the original 20 bytes of buffer meaning we'd only get access to 5 instructions. We only move the address in `sp` back. This does not change the pointer in `a0`. We specifically want to be able to jump to the all the extra instruction bytes we have.

Anonymous Rook 7mth #986cab

Thanks! That makes more sense, but why do a0 and sp point to different things?

J Justin Yokota STAFF 7mth #986cef

Replying to Anonymous Rook

We changed the value of sp, but not a0.

Anonymous Rook 7mth #986aff Unresolved

FA21-Final-Q2.3 and 2.4 I am lost as to when we round? I asked this question but since the thread is resolved I'll put it here to #986afe

Similarly, for 2.4, wouldn't we get closer if we choose to max out (ie. have 1 for every possible spot) rather than limit ourselves to only numbers with LSB 0?

Anonymous Rook 7mth #986afd ✓ Resolved

FA21-Final-Q1.9

For Q1.9, why wouldn't Thread 1 read y=0, go to sleep, Thread 2 runs to completion (x=0), Thread 1 reads x=0, skips the while loop and then has y=0? Also, why in the solution does it write to y before reading x? since the y write is inside the while loop which requires x to be known?

Q1.9 (1 point) We run the following code on two threads.

```
int y = 0;
int x = 10;
#pragma omp parallel
{
   while(x>0)
   {
      y = y+1;
      x = x-1;
   }
}
```

What is the smallest possible value y can contain after this runs?

Solution: This one's a bit tricky. The optimal sequence is:

Thread 1 reads y=0 and goes to sleep.
Thread 2 runs to completion.
Thread 1 wakes up and writes y=1, reads x=0, sets x=-1, then sees x == -1 and stops the loop.

♡ ...

N Nikhil Kandkur STAFF 7mth #986bcf

X would have to be read first in order for Thread 1 to enter the while loop, and after it sleeps and then wakes up in the while loop, the thread has to finish the while loop by adding 1 to y=0, and then x=x-1.

♡ ...

Anonymous Rook 7mth #986cad

Thanks! Just to make sure, this is what actually happens then: Thread 1 reads y=0 and x=10 and goes into the while loop (10>0) then goes to sleep. Thread 2 runs to completion s.t. x=0 and y = 10. Then Thread 1 wakes up and sets y=0+1 (using Thread 1's version of y=0) and x=0-1 (using Thread 2's version of x=0).

I'm confused on this last step, why are we able to choose which version of x and y to read?

♡ ...

N Nikhil Kandkur STAFF 7mth #986cbd

Replying to Anonymous Rook

Thread 1 reads the value of Y (which in the beginning is zero) and then goes to sleep. Remember that a statement like `y = y + 1` is a read-modify-write operation, so in this case, the read is done, and the modify and write will be done later. Before thread 1 goes to sleep, it will not have read x yet, so when the thread wakes up, it will use the value of y=0 that it read earlier, and following the modify and write operations of `y = y + 1`, a fresh value of X will be read, which is 0 at that point in time, and then modified to set X to -1.

♡ ...

Anonymous Seahorse 7mth #986ccf

Replying to Nikhil Kandkur

Then when is x=10 going to be read in thread 1?

Nikhil Kandkur STAFF 7mth #986cdc

Replying to Anonymous Seahorse

x = 10 will not be read by thread 1, because by the time thread 1 reads x, the stack address of X will hold 0.

Anonymous Hedgehog 7mth #986cfa

Replying to Nikhil Kandkur

To clarify, does thread 1 only go through line 1 where it reads y = 0 and then goes to sleep? If so, how can thread 1 go into the while loop after thread 2 runs to completion and sets x = 0? Doesn't thread 1 have to read line 2 at some point as well or does it do that before it goes to sleep?

Nikhil Kandkur STAFF 7mth #986cfb

Replying to Anonymous Hedgehog

Thread 1 goes all the way to line 7, where it reads y = 0, and then goes to sleep.

Anonymous Hedgehog 7mth #986cfc

Replying to Nikhil Kandkur

Why do we use thread 2's value of x and thread 1's value of y then?

Nikhil Kandkur STAFF 7mth #986cfd

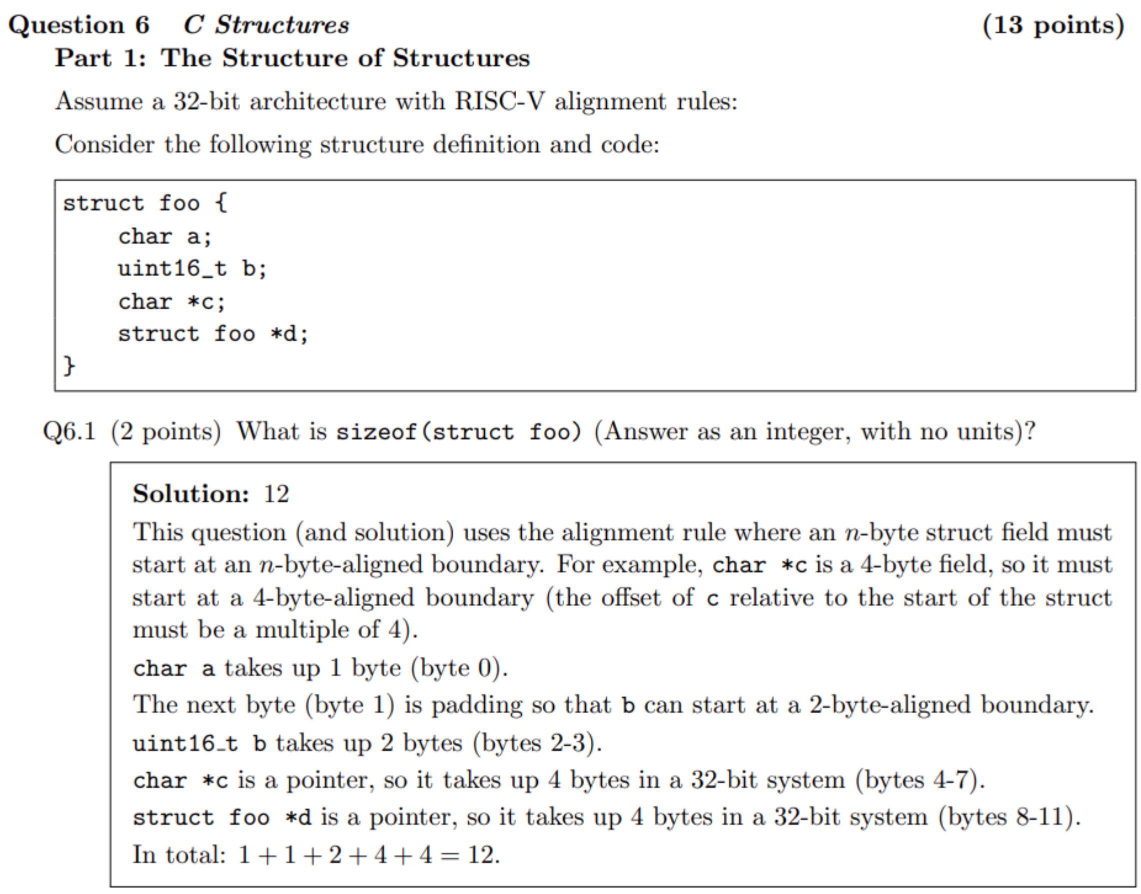
Replying to Anonymous Hedgehog

Right before thread 1 goes to sleep, thread 1 will read the value of y into its registers, and then go to sleep. When it wakes up, it will have the already-read value of y in its registers, so it can just use that value, complete its operations on line 7, and then move on to line 8, where it will have to read the value of x from memory, which was updated by Thread 2.

1

Anonymous Cat 7mth #986afc ✓ Resolved

sp 21 q6a,b,   i had a question on q6, don't remember learning this, which lecture covered this



Question 6 *C Structures* **(13 points)**

Part 1: The Structure of Structures

Assume 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;
}
```

Q6.1 (2 points) What is `sizeof(struct foo)` (Answer as an integer, with no units)?

Solution: 12

This question (and solution) uses the alignment rule where an n-byte struct field must start at an n-byte-aligned boundary. For example, `char *c` is a 4-byte field, so it must start at a 4-byte-aligned boundary (the offset of `c` relative to the start of the struct must be a multiple of 4).

`char a` takes up 1 byte (byte 0).

The next byte (byte 1) is padding so that `b` can start at a 2-byte-aligned boundary.

`uint16_t b` takes up 2 bytes (bytes 2-3).

`char *c` is a pointer, so it takes up 4 bytes in a 32-bit system (bytes 4-7).

`struct foo *d` is a pointer, so it takes up 4 bytes in a 32-bit system (bytes 8-11).

In total: $1 + 1 + 2 + 4 + 4 = 12$.

(__m256i *) Jasmine Angle STAFF 7mth #986bbe

A more detailed overview of this idea can be found in Lecture 3 - C Pointers, Arrays, and Strings on slides 43-46. Due to the RISC-V alignment rules (loading using `lw`, `lh`, and `lb`), the C compiler will add padding between variables of varying sizes within a struct that don't exactly align to fill a full word.

Anonymous Tapir 7mth #986afb ✓ Resolved

Fa21-Final 5.1

Can MemRW be just Don't Care and WBSel be ALU?

(__m256i *) Jasmine Angle STAFF 7mth #986bbd

For 5.1, since we are recreating the signals for an `lw` instruction, we know that we are loading data *from* memory, so we set the control signal for MemRW to "Read," meaning we are now retrieving values from the DMEM block. Because we want to write the value we pulled from DMEM back to a register, we select the MEM port for WBSel.

If we instead decided to choose MemRW to be "Don't Care," what would instead happen is we might end up writing garbage data to memory. If we also have the WBSel signal set to use the ALU port, then the instruction would instead write back the address we wanted to load from into the destination register, rather than the data we wanted to load from memory.

Since both MemRW and RegWEn both directly control whether or not we affect state elements, these will never have a proper condition where they would be set to a "Don't Care" value.

Anonymous Turkey 7mth #986aef ✓ Resolved

Sp21-Final 3.iC

Are we supposed to know what parse trees are w.r.t CALL?

Candice Yang STAFF 7mth #986bdf Visible to staff only

CALL is in scope in general, but none of this semester's material mentioned parse trees so I would say this question is slightly out of scope.

Candice Yang STAFF 7mth #986bea Visible to staff only

Can anyone ack :(

Jero Wang STAFF 7mth #986bee Visible to staff only

sending to instructors

Jero Wang STAFF 7mth #986bef

Parse trees are not in scope.

Anonymous Turkey 7mth #986aee ✓ Resolved

Fa21-Final Q6.3 Would a valid way to do this problem be to simply ignore any rows that have an X as the output (since we do not know if it is a 0 or 1).

(__m256i *) Jasmine Angle STAFF 7mth #986bbc

Yep! The X for the output here means that the output for that combination of inputs does not matter, so it is safe to ignore that row.

Anonymous Mantis 7mth #986cbf

how were we able to simplify ~WZ + W~Z into just W AND Z?

Anonymous Fish 7mth #986aeb ✓ Resolved

Sp21-Final Q1D

why would this increase performance if a #pragma omp parallel wasn't placed before the #pragma omp for?

Rohan Mathur STAFF 7mth #986bda

I think you're technically right. My guess is that with this question you're supposed to assume it's in a parallel block.

Anonymous Monkey 7mth #986adc ✓ Resolved

Sp21-Final Q2F,G: Can someone explain how to go about this question step by step; I did not really understand the explanation for part 1

D. Code Analysis

We run the following code with our caching setup unchanged from the previous question. What is the hit rate of the following lines of code?

E.

```
#define base_arr_addr 0x12345678
#define ARR_SIZE 4096
#define I_BOUNDARY 2048
#define J_BOUNDARY 4096
#define I_STRIDE 128
#define J_STRIDE 64
int arr[ARR_SIZE];

uint32_t dummy_func(void) {
  //Loop 1
  for (int i = 0; i < I_BOUNDARY; i += I_STRIDE) {
    for (int j = 0; j < J_BOUNDARY; j += J_STRIDE) {
      arr[i] = arr[i] + arr[j];
    }
  }

  //Loop 3
  for (int j = 0; j < J_BOUNDARY; j += J_STRIDE) {
    for (int i = 0; i < I_BOUNDARY; i += I_STRIDE) {
      arr[j] = arr[j] * arr[i];
    }
  }
}
```

F. `arr[i] = arr[i] + arr[j]`

47/48

Ther outer loop executes 16 times and the inner loop per round executes 64 times. Because the accesses only happen in the inner loop, we can tally the HR for that first and then see how the outer accesses affect the HR. For the first inner iteration, we see we have a miss on the `arr[0]` read, then a hit on `arr[1]` and on the `arr[0]` write. For the next iteration, we get a `arr[0]` read and write. The next `arr[j]` is where accesses become tricky; we're stepping by 64 * 4B = 256B; this is larger than one block which means every subsequent `arr[j]` access will be a miss for one outer iteration. Thus, for the first outer iteration, our overall HR is 2/3. On the next outer iteration however, we notice that the entirety of the array can fit in the cache without conflicts. Because our `I_STRIDE` is larger than `J_STRIDE`, we know every future `arr[i]` access will have already had a compulsory miss by `arr[j]` in previous iterations and because nothing is evicted, we have a HR of 1 for all future access. This gives us a total of: HR = 2/3 * 1/16+ 3/3 * 15/16 = 47/48. NOTE: the fact our array is 4-way does not affect us here because despite our 128B jumps in accesses, we have 2^11 sets available to us which means we won't fill up the first way in each set before we fill the second way. Because we only have 2 nested loops with two access patterns, nothing will get kicked out in each set for another access pattern.

G. `arr[j] = arr[j] * arr[i]`

1

The entire array can fit in memory. Because the access pattern is effectively the same, all

accesses are hits.

♡ 1 ···

Anonymous Wombat 7mth #986afa

Im also confused, particularly on why the HR is 2/3 for the first outer iteration.

In the very first inner iteration, the HR will be 2/3 because the arr[i] read misses, but the arr[j] read and arr[i] store hit (since i = j = 0). However, once we go through the inner loop, as the solution says, arr[j] will miss because the step size is larger than the block size. So now arr[i] store hits, but arr[j] and arr[i] read miss making the HR 1/3 for the rest of the inner iterations. By this logic (and the fact that there are 64 iterations in the inner loop) we should have: 2/3 * (1/64) + 1/3 * (63/64) as the HR for one full inner loop iteration.

Where have I gone wrong in my thinking?

♡ 1 ···

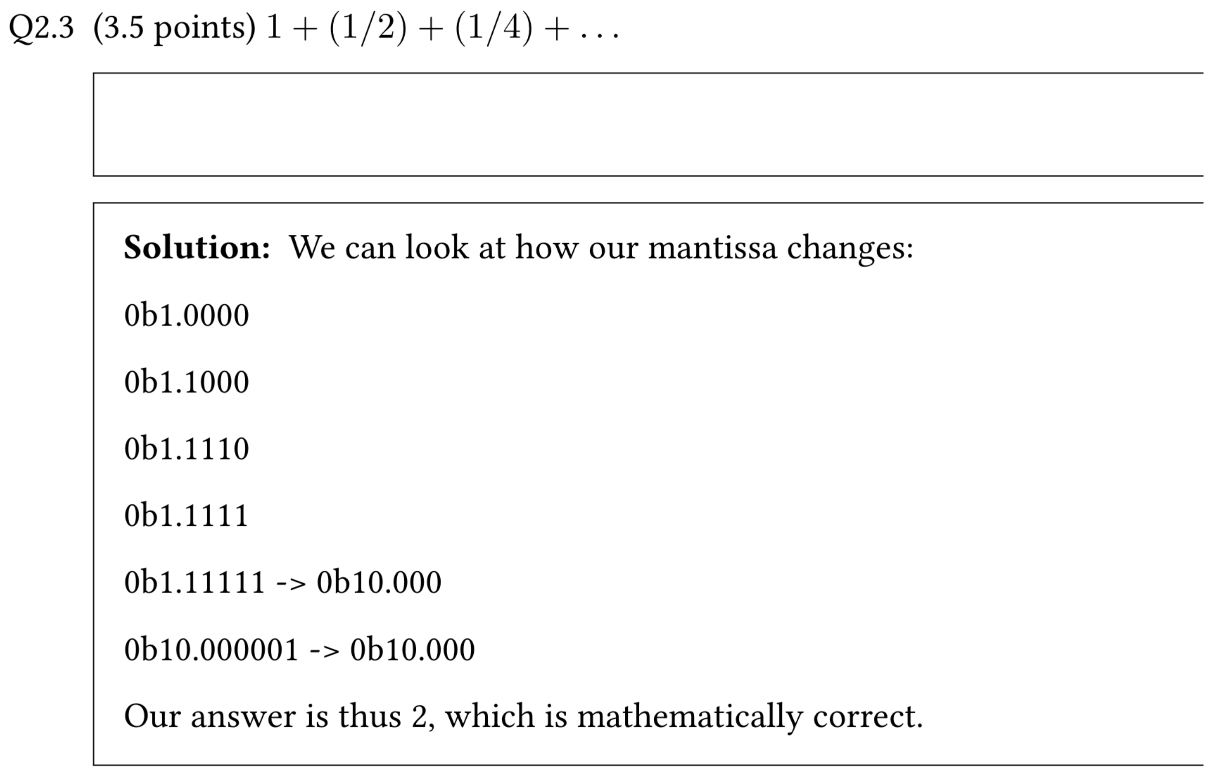
Tyler Yang STAFF 7mth #986bff

You are correct that the first inner iteration will have a hit rate of 2/3 (one miss to read the initial arr[i]). However, the rest of the inner iterations, i will always be 0. Thus, although j will continue to miss, both arr[i] accesses will be hits which keeps the hit rate at 2/3.

♡ ···

Anonymous Grouse 7mth #986acf ✓ Resolved

Fa 2021, A bit confused on the logic for these. Could someone explain for 2.3 how we get from 0b1.11111 -> 0b10.000?


Q2.3 (3.5 points) $1 + (1/2) + (1/4) + \ldots$

Solution: We can look at how our mantissa changes:

0b1.0000

0b1.1000

0b1.1110

0b1.1111

0b1.11111 -> 0b10.000

0b10.000001 -> 0b10.000

Our answer is thus 2, which is mathematically correct.

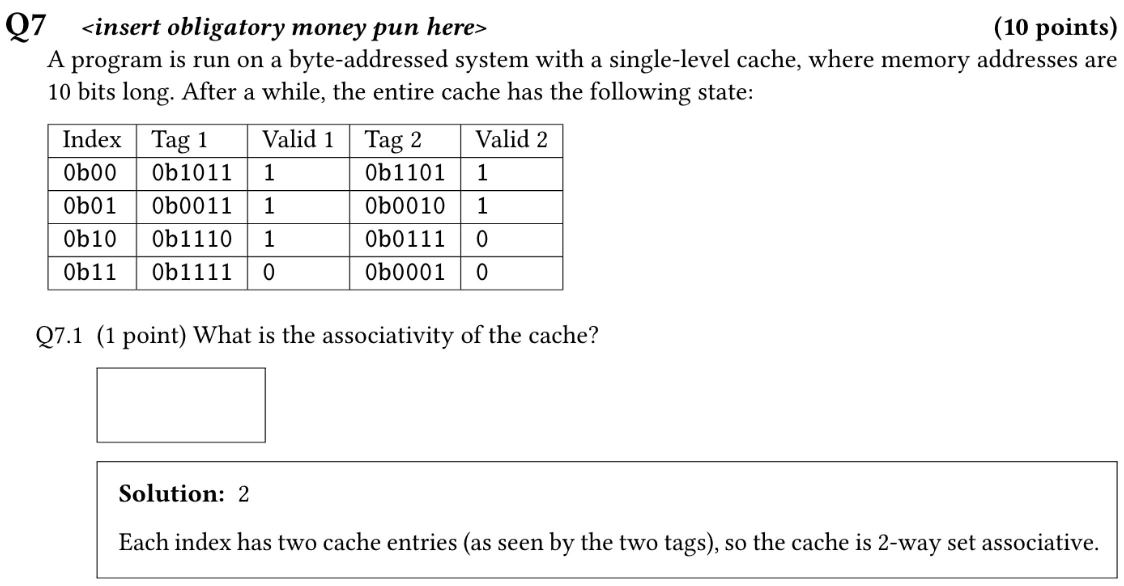
♡ ···

(__m256i *) Jasmine Angle STAFF 7mth #986bcc

Per the rounding standard defined earlier in the question, floating point numbers round to the "most even" number, or divisible by the most factors of two (LSB being 0). We do this rounding when we are in a situation where we are precisely between two floating point numbers. When we get to 0b1.11111, the least significant bit in the mantissa is not representable with only four mantissa bits, thus we find ourselves in precisely this situation (between 1.1111 and 10.000). Because 0b1.1111 is less even than 0b10.000, we round to 0b10.000.

Anonymous Grouse 7mth #986abf ✓ Resolved

Fa 2021, this question was focused on using a 2-way set associative. Are we suppose to know how these work?



Q7 ***<insert obligatory money pun here>*** **(10 points)**

A program is run on a byte-addressed system with a single-level cache, where memory addresses are 10 bits long. After a while, the entire cache has the following state:

Index	Tag 1	Valid 1	Tag 2	Valid 2
0b00	0b1011	1	0b1101	1
0b01	0b0011	1	0b0010	1
0b10	0b1110	1	0b0111	0
0b11	0b1111	0	0b0001	0

Q7.1 (1 point) What is the associativity of the cache?

Solution: 2

Each index has two cache entries (as seen by the two tags), so the cache is 2-way set associative.

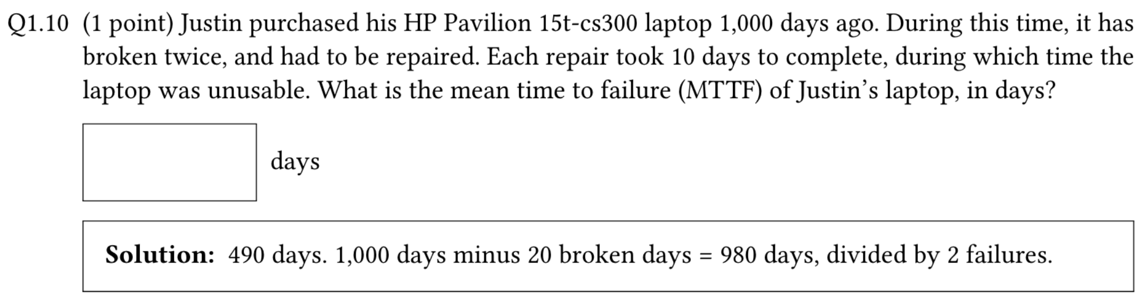
Anonymous Rhinoceros 7mth #986aca

In the post it says "As a note, some questions will be out of scope because set-associative caches are out of scope this semester" -- I just think we need to know what it is (like how it there are multiple blocks per index) , but not how to implement it.

1

Anonymous Grouse 7mth #986abe ✓ Resolved

Fa 2021 Are we suppose to know about mean time to failure?



Q1.10 (1 point) Justin purchased his HP Pavilion 15t-cs300 laptop 1,000 days ago. During this time, it has broken twice, and had to be repaired. Each repair took 10 days to complete, during which time the laptop was unusable. What is the mean time to failure (MTTF) of Justin's laptop, in days?

days

Solution: 490 days. 1,000 days minus 20 broken days = 980 days, divided by 2 failures.

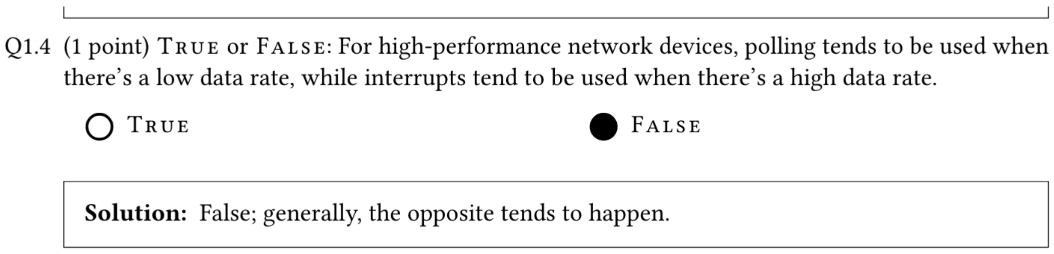
Myrah Shah STAFF 7mth #986adb

#986ef

Anonymous Grouse 7mth #986abd ✓ Resolved

Fa 2021

Is polling or interrupts in scope for us? I've seen a few questions on them in past exams but don't believe we covered them in class this year. Please correct me if I'm wrong.



Q1.4 (1 point) TRUE or FALSE: For high-performance network devices, polling tends to be used when there's a low data rate, while interrupts tend to be used when there's a high data rate.

○ TRUE ● FALSE

Solution: False; generally, the opposite tends to happen.

Catherine Van Keuren STAFF 7mth #986aea

I believe that polling and interrupts are out of scope. They're usually taught when we discuss I/O which was mainly out of scope except for one slide in the OS lecture (which I don't think talks about polling and interrupts at all).

1

Anonymous Lemur 7mth #986abc ✓ Resolved

SP21-Final-Q6b(v)

In the solutions, the first line is slli t1 t1 4, but in the video walkthrough, the answer is written is slli t1 t1 2. Which is correct?

Catherine Van Keuren STAFF 7mth #986adf

Nice catch - it should be slli t1 t1 2 since each element in the array is 4 bytes.

1

Anonymous Wombat 7mth #986cbc

Where is the video walkthrough?

Anonymous Elephant 7mth #986abb ✓ Resolved

FA21-Final-Q2.4

I know there's a short question about it already, but I was wondering if it would be possible to get a more in depth explanation of how we even pick 64 in the first place. In other words, I understand that when we get to 64, it then becomes 66 and then rounds down, but how would we even think about using 64 on an exam?

Justin Yokota STAFF 7mth #986acc

You don't really "pick" 64, per se. It's more that you determine what behavior would cause this code to stop increasing (the rounding issue), and based on that, what number you reach once that rounding starts happening.

Anonymous Lemur 7mth #986aba ✓ Resolved

SP21-Final-Q2a(i) Part K

Is the question on the 2-level page table in scope?

Myrah Shah STAFF 7mth #986ada

As in the post:

> As a note, some questions will be out of scope because set-associative caches are out of scope this semester.

1

Adrian Bao 7mth #986aae ✓ Resolved

FA21-Final-Q3.4

My immediate is wrong and my original answer was 0x**9**00300E7.

The reason for this inaccuracy is because I wrote my immediate representing -256 as

1 (sign bit) 001 0000 0000 and I didn't do 2's complement correctly.

How do I calculate this immediate correctly? Thanks!

Anonymous Rook 7mth #986bae

I think its because you will need to sign extend

Candice Yang STAFF 7mth #986bde

Edit: sorry I misinterpreted your question, here is the udpated answer.

jalr is I type instruction, which has 12 immediate bits. So note that you would need to convert -256 as a 12-bit two's complement number. So +256 is `0b 0001 0000 0000`, flipping it gives us `0b1110 1111 1111`, adding 1 to it yields `0b1111 0000 0000`, which is `0xF00`.

Adrian Bao 7mth #986aad ✓ Resolved

FA21-Final-Q3

3.3.1 - why are there 24 bytes of injected instructions from part 1?

3.3.2 - How does mv a0 sp accomplish the goal of putting a0 at the top of the buffer?

3.3.3 - Why does lui work since it takes the 12 byte immediate and puts it right after the 0x of the rd? 0x1000 is 16 bytes. Also, why wouldn't auipc work since it has the same function but adds to PC?

3.3.5 - Why is jr allowed since we aren't linking ra, but it is linking nothing (x0)? Would j also work? (jal x0 label)

1

Justin Yokota STAFF 7mth #986acb

3.3.1: By the time we run our injected code, the sp has been set to its value before running the verifypassword function. The first 24 bytes below the sp correspond to the data we injected, so we need to start our buffer AFTER that segment.

3.3.2: This sets a0 to point to the top of the buffer, since sp is just a pointer and mv copies that value.

3.3.3: lui doesn't take in 12 bits of immediate; it takes in 20 bits of immediate and sets the value of the register to that <<12. We don't want to add to PC here (we just want to set the register value to 4096)

3.3.5: There isn't really a return address, so there's no need to set ra. j doesn't work, since we're not jumping to a label; there's no label pointing to a random spot in the stack we're using to exploit this code, since the original programmer never intended us to jump there.

♡ 1 ···

A Adrian Bao 7mth #986aec

Thanks for your detailed response!

For 3.3.3, I am a bit confused with the use of lui. We are using lui t1 1.

I am not sure why lui would take 1, interpret it as 0x1000, then assign it to t1.

My theory is that lui interprets 1 as 0x10000 so that it would be 20 bits, and chops off the two last zeros so that it is 0x100 which is 12 bits.

But I am confused on how these 12 bits 0x100 would be left shifted correctly into t1. t1 is not necessarily all 0s, and it is 32 bits.

♡ ···

J Justin Yokota STAFF 7mth #986aed

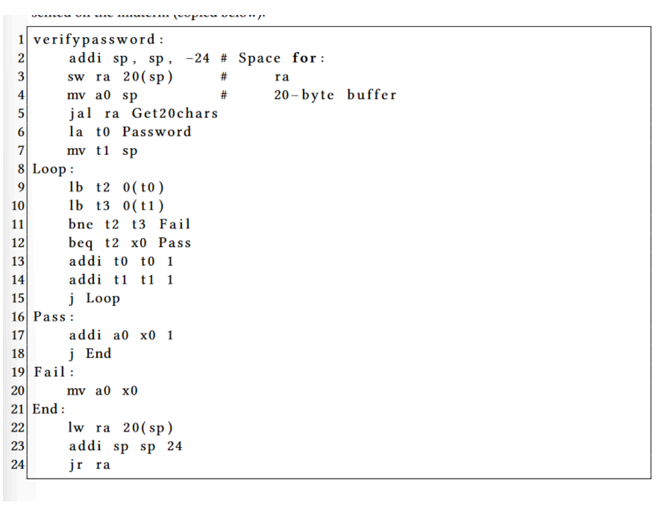
Replying to Adrian Bao

Not really... "lui t1 1" is "lui t1 0x00001", extending 1 to 20 bits. We thus set t1 to 0x00001<<12 = 0x0000 1000, per the behavior of lui specified on the reference card.

♡ 1 ···

A Adrian Bao 7mth #986aac ✓ Resolved

FA21-Final-Q3.1


sented on the midterm (copied below):

```
verifypassword:
    addi sp, sp, -24 # Space for:
    sw ra 20(sp)     #     ra
    mv a0 sp         #     20-byte buffer
    jal ra Get20chars
    la t0 Password
    mv t1 sp
Loop:
    lb t2 0(t0)
    lb t3 0(t1)
    bne t2 t3 Fail
    beq t2 x0 Pass
    addi t0 t0 1
    addi t1 t1 1
    j Loop
Pass:
    addi a0 x0 1
    j End
Fail:
    mv a0 x0
End:
    lw ra 20(sp)
    addi sp sp 24
    jr ra
```

How would overflowing a0 with 20 "A"s and appending 0xDE 0xAD 0xBE 0xEF cause the overflow?

Does mv a0 sp put a0 at the very top (index 0) of the stack and the ra is on index 20 causing the overflow after spamming 20 "A"s?

But the answer says, "Looking at the code, we can see that we put the saved value of the ra register directly above the buffer on the stack." But the buffer isn't a0. Thanks!

...

Justin Yokota STAFF 7mth #986acd

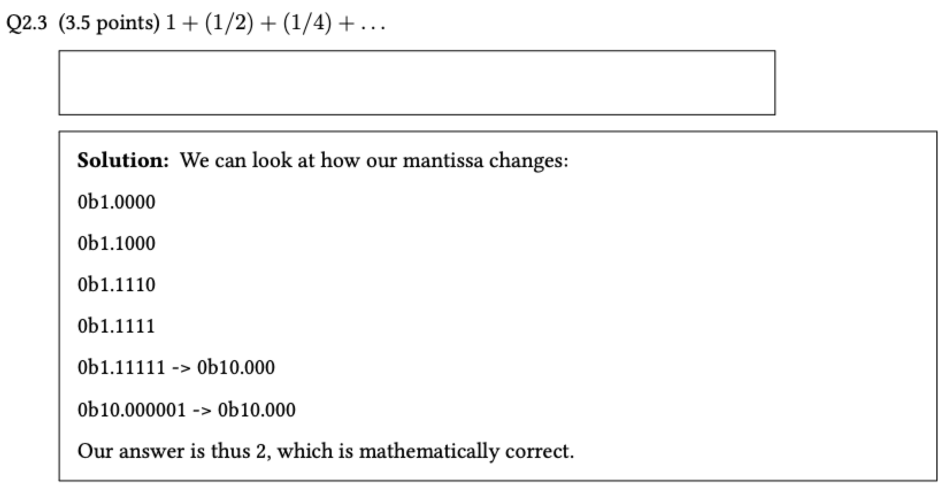
Yes, mv a0 sp puts a0 at index 0 of the stack, and ra is index 20. The buffer we send in to Get20chars is indeed a0, which is also sp+0.

1 ...

Anonymous Elephant 7mth #986aaa ✓ Resolved

FA21-Final-Q2.3

Could someone please explain the fifth step onwards where it becomes 10.000? Specifically, how does the rounding come into play here? Why isn't 1.11111 rounded to 1.11110 since that has a LSB of 0?



Q2.3 (3.5 points) $1 + (1/2) + (1/4) + \ldots$

Solution: We can look at how our mantissa changes:

0b1.0000

0b1.1000

0b1.1110

0b1.1111

0b1.11111 -> 0b10.000

0b10.000001 -> 0b10.000

Our answer is thus 2, which is mathematically correct.

...

Justin Yokota STAFF 7mth #986ace

Note that we only have 4 mantissa bits, so 1.11110 would instead be 1.1111, which has an LSB of 1.

1 ...

Anonymous Rook 7mth #986afe

But wouldn't 1.1111 still be closer to 1.11111 than 10.000? So we wouldn't even invoke the rule of choosing the one with an LSB of 1?

...

Justin Yokota STAFF 7mth #986bad

Replying to Anonymous Rook

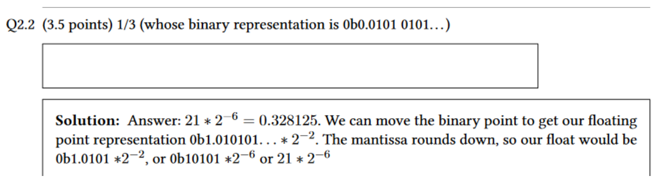
Nope, both are exactly 0b0.00001 away from 0b1.11111

...

Adrian Bao 7mth #986ff Unresolved

FA21-Final-Q2.2

Seems the other question labeled FA21-Final-Q2.2 is for 2.4.


Q2.2 (3.5 points) 1/3 (whose binary representation is 0b0.0101 0101…)

Solution: Answer: $21 * 2^{-6} = 0.328125$. We can move the binary point to get our floating point representation $0b1.010101\ldots * 2^{-2}$. The mantissa rounds down, so our float would be $0b1.0101 * 2^{-2}$, or $0b10101 * 2^{-6}$ or $21 * 2^{-6}$

What is meant by rounding down the mantissa?

For rounding the significand, why don't we round to the nearest 0, as 0b1.010101 is 21 after the 1, and it is between the representable floats 20 and 22?

Anonymous Lemur 7mth #986fe ✓ Resolved

SP21-Final-Q2a

How do we know that each block of the cache can store 32 integers (as stated in the video walkthrough)?

C Catherine Van Keuren STAFF 7mth #986ade

Each cache block is 128B so if we divide that by 4 bytes per int, then we get 128/4 = 32 ints!

1

Anonymous Turkey 7mth #986fd ✓ Resolved

SP21-Final-Q9.1

Why is "Add additional control signals for the writeback mux" not an answer? Don't we also need to be able to write back the bit 1 or 0 into rd?

Candice Yang STAFF 7mth #986bdc

Do you mean Q7a.i?

Anonymous Turkey 7mth #986bdd

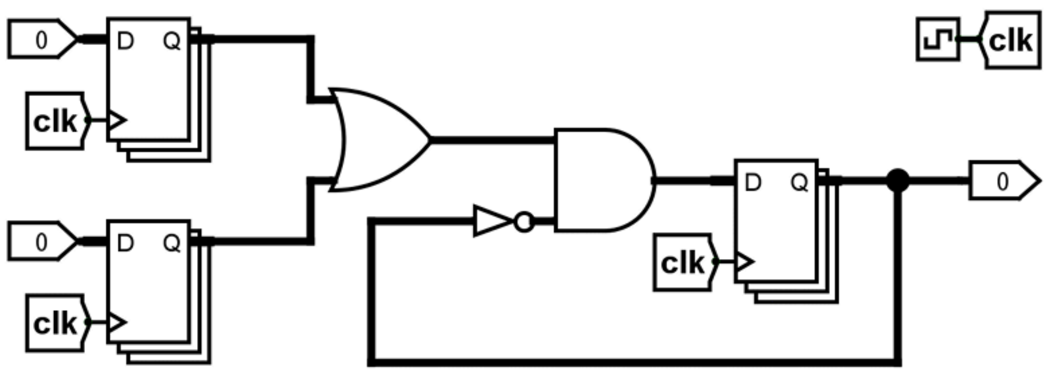
Oh yes sorry, I think the questions are numbered differently on the rewritten solutions PDF

Candice Yang STAFF 7mth #986bec

Replying to Anonymous Turkey

I see! WB mux already has one control signal `WBSel`, and we have also modified the control logic for `WBSel` to support selecting the correct output. So we don't need additional ones.

Anonymous Crow 7mth #986fb ✓ Resolved



FA21Q4: Can someone please confirm why the shortest CL is only the not gate and AND gate. Doesn't the data from the OR gate have to be output before the AND can run.

J

Justin Yokota STAFF 7mth #986fc

Gates don't "run" in a traditional sense; rather, they always are reading their inputs and changing their output to match. Once either input to the AND gate changes, the AND gate's output will begin changing to match the value of its inputs, delayed by however long the AND gate takes to transmit its data.

1

Anonymous Fish 7mth #986ee ✓ Resolved

Fa21 6.3 How would one go about solving this when we don't have all set values of X, Y, Z that correspond to 1 outputs?

1

E

Erik Yang STAFF 7mth #986bfe

#986bfc

Anonymous Dotterel 7mth #986ea ✓ Resolved

SP21 6.1 and 6.2

How does padding work? When does it align to 2 bytes and when does it align to 4 bytes?

C

Catherine Van Keuren STAFF 7mth #986add

It depends the type of the variable. Types that are 4 bytes need to be 4 byte aligned, whereas types that are 2 bytes need to be 2 byte aligned. This is demonstrated in the question. in 6.1, we first store "a", then "b" since its 2 bytes, needs to be 2 byte aligned aka we need to add one byte of padding to achieve this. In 6.2, we first store "a", then we need to store "c" which is 4 bytes and needs to be 4 byte aligned, hence we need to add 3 bytes of padding

Anonymous Dotterel 7mth #986df ✓ Resolved

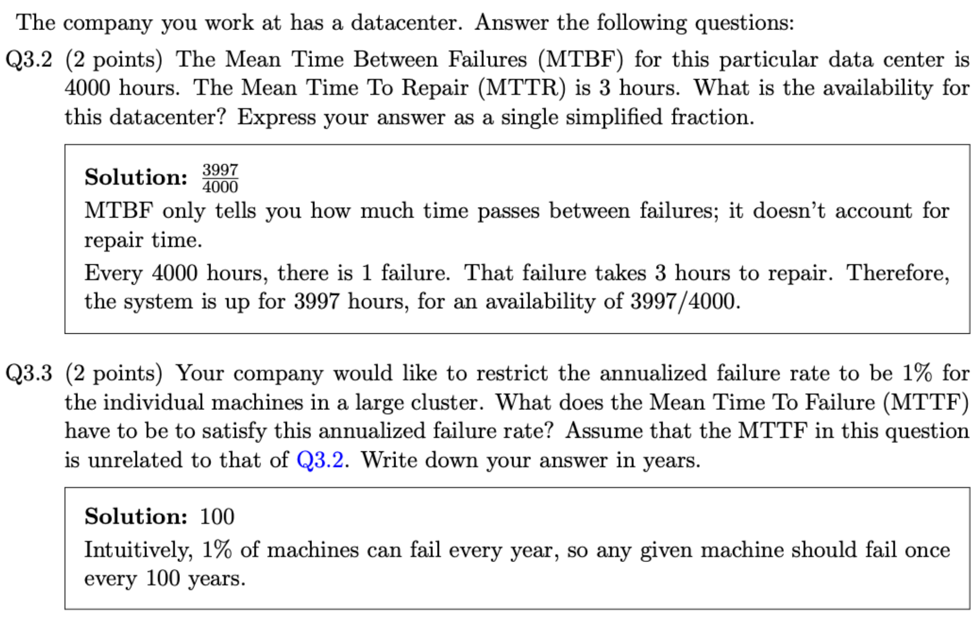
SP21 3.2 3.3

These are out of scope right?

C Catherine Van Keuren STAFF 7mth #986eb

I'm not quite sure which question you are referring to with 3.2 and 3.3 since question 3 on the SP21 Final only has 3.a, but if you are referring to question 3 on the SP21 final, this question is in scope. We covered FSMs earlier this semester. Follow up if this isn't the question you're referring to.

Anonymous Dotterel 7mth #986ed


The company you work at has a datacenter. Answer the following questions:

Q3.2 (2 points) The Mean Time Between Failures (MTBF) for this particular data center is 4000 hours. The Mean Time To Repair (MTTR) is 3 hours. What is the availability for this datacenter? Express your answer as a single simplified fraction.

Solution: $\frac{3997}{4000}$
MTBF only tells you how much time passes between failures; it doesn't account for repair time.
Every 4000 hours, there is 1 failure. That failure takes 3 hours to repair. Therefore, the system is up for 3997 hours, for an availability of 3997/4000.

Q3.3 (2 points) Your company would like to restrict the annualized failure rate to be 1% for the individual machines in a large cluster. What does the Mean Time To Failure (MTTF) have to be to satisfy this annualized failure rate? Assume that the MTTF in this question is unrelated to that of Q3.2. Write down your answer in years.

Solution: 100
Intuitively, 1% of machines can fail every year, so any given machine should fail once every 100 years.

E Eddy Byun STAFF 7mth #986ef

Replying to Anonymous Dotterel

These are out of scope

Anonymous Fish 7mth #986de ✓ Resolved

Fa21-Final Q1

Are 1.2, 1.4, 1.10, 1.11 and 1.13 in scope? Also for 1.5 what is a scheduler? What lecture was that covered in?

C Catherine Van Keuren STAFF 7mth #986ec

1.2 - in scope, we learned about Linkers during the CALL lecture

1.4 - out of scope, we didn't discuss I/O devices in detail this semester so no polling/interrupts

1.10 - out of scope, apart of our optional dependability lecture

1.11 - out of scope, same as above

1.13 - in scope, Amdahl's law is discussed in the first parallelism lecture

1.5 - It doesn't look like schedulers are explicitly defined in lecture so I think that question in particular is considered out of scope, but I would still understand the idea behind the ordering of threads and how it relates to program execution.

Anonymous Wombat 7mth #986cd ✓ Resolved

sp21 - 2i

I'm not sure how they got 1024 as the answer here. I calculated that each page is 12-bits long (by converting the 4KiB). Also since memory is 16MiB, this corresponds to 24-bits or 2 pages for each PTE.

In the first iteration, both i and j are equal to 0 so they are accessing the same memory. So wouldnt we create just 2 new pages in the page table?

2

Anthony Salinas Suarez STAFF 7mth #986dbb

Hey there! Apologies for not getting to this sooner but below I will attach my response to another student with a very similar question. I hope this helps!

- Page Size is 4 KiB = 2^{12} Bytes
- Physical Memory is 16 MiB = 2^{24} Bytes
- Physical Memory is 4 GiB = 2^{32} Bytes
- Offset is $\log_2$(Page Size in Bytes) = $\log_2(2^{12})$ = 12 bits
- Number of bits in Physical Address is $\log_2$(Physical Memory in Bytes) = $\log_2(2^{24})$ = 24
- Number of bits in Virtual Address is $\log_2$(Virtual Memory in Bytes) = $\log_2(2^{32})$ = 32
- So PPN is therefore 24 - 12 = 12 bits
- VPN is 32 - 12 = 20 bits

Now using this information, recall that under normal circumstances, the size of each Page Table Entry (PTE) will be equal to the (number of bits of our PPN) + (number of bits for Metadata) and given we have 8 bits worth of Metadata then 12 + 8 = 20 bits. **Note however, that under the assumption of a word aligned system, we round up to the nearest multiple of 4 bytes so 20 --> 32 bits --> 4 bytes.**

Now because the Page Table maps virtual pages to physical pages and because we found that we have 20 bits making up our VPN, then this means we have 2^{20} virtual pages and therefore 2^{20} PTE's. Given that each PTE is 4 bytes in size then this means the size of our Page Table is $2^{20} * 2^2 = 2^{22}$ bytes.

Last but not least, we have that each page is of size 2^{12} bytes. So our page table takes up (2^{22} / 2^{12}) --> 2^{10} --> 1024 physical pages in memory.

Note: There is also a nice walkthrough video available on the course site under the exam resources page which I will link here: https://cs61c.org/sp24/resources/exams/ and if you are interested in watching the walkthrough for this portion, the video discusses this problem around the 1 hour 50 minute mark!

1

Anonymous Wombat 7mth #986dbe

So should we always account for the word-alignment for PTEs unless the problem clearly says otherwise? I feel like I have done similar questions but arrived at the right answer without considering this. I could be wrong, but would like to know for future reference.

M Myrah Shah STAFF 7mth #986ddb

Replying to Anonymous Wombat

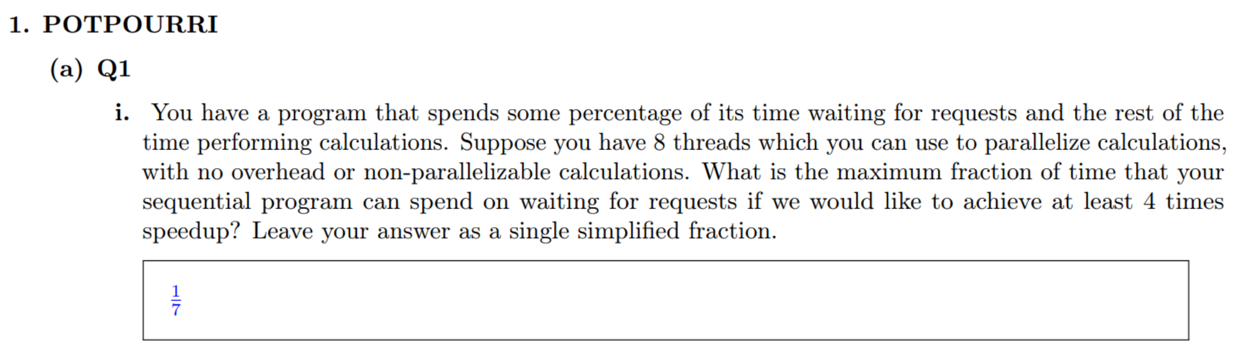
Note from the solutions:

> NOTE: because we did not specify alignment for the system, if you solved for a byte-aligned system, you will get the points for 768 pages.

I would say that it will generally be in the problem description.

F Fouad Sinno 7mth #986cc ✓ Resolved

SP21-Final Q1



1. POTPOURRI

(a) Q1

i. You have a program that spends some percentage of its time waiting for requests and the rest of the time performing calculations. Suppose you have 8 threads which you can use to parallelize calculations, with no overhead or non-parallelizable calculations. What is the maximum fraction of time that your sequential program can spend on waiting for requests if we would like to achieve at least 4 times speedup? Leave your answer as a single simplified fraction.

$\frac{1}{7}$

I proceeded to solve this using Amdahl's law with the formula explained in lecture:
Speedup = 1 / ((1 - F) + (F / S))
4 = 1 / ((1 - F) + (F / 8))
F = 6/7, which is not equal to 1/7.

Am I replacing the variables with the wrong values and/or am I applying Amdahl's law incorrectly here?

1

Andrew Liu STAFF 7mth #986dd

Nope, you are actually doing everything exactly correctly! Remember that F is the fraction that can be sped up. In this question, waiting for requests is the part of the work that cannot be sped up, so you just need to take 1-F to find out your desired value.

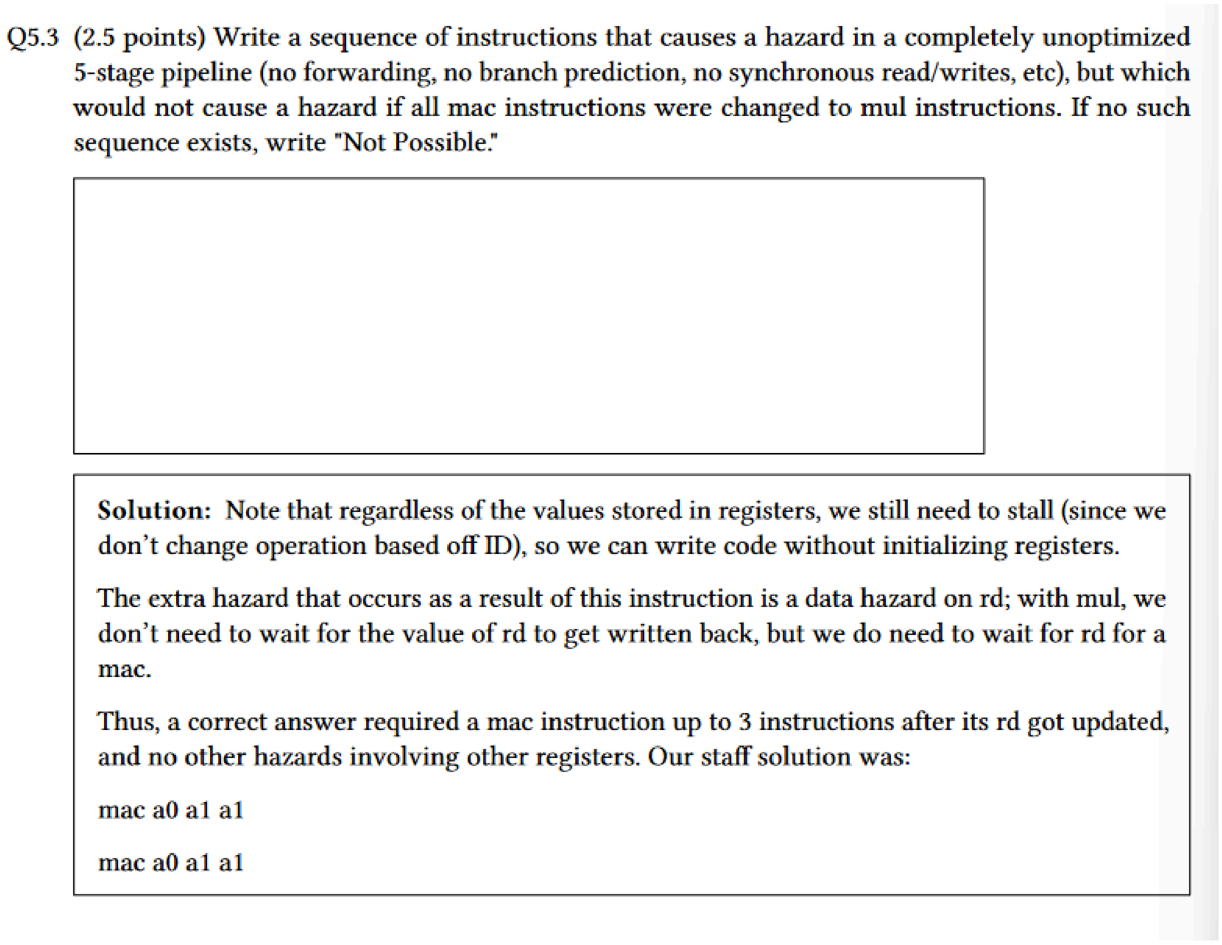
A Adrian Bao 7mth #986cb ✓ Resolved

FA21-Final-Q5.3

Why do we need to stall so that we can write code without initializing registers? I thought no matter what we need to initialize the registers, i.e. with li, and stalling wouldn't do that initialization.

Also, why with mul, we don't need to wait for the value of rd to get written back, but with mac (multiplying and adding), we need to wait for rd? Is that because we are doing the multiplication and then addition, but I thought that is one instruction inside the ALU, so there shouldn't be a data hazard with rd.

Thanks!



Q5.3 (2.5 points) Write a sequence of instructions that causes a hazard in a completely unoptimized 5-stage pipeline (no forwarding, no branch prediction, no synchronous read/writes, etc), but which would not cause a hazard if all mac instructions were changed to mul instructions. If no such sequence exists, write "Not Possible."

Solution: Note that regardless of the values stored in registers, we still need to stall (since we don't change operation based off ID), so we can write code without initializing registers.

The extra hazard that occurs as a result of this instruction is a data hazard on rd; with mul, we don't need to wait for the value of rd to get written back, but we do need to wait for rd for a mac.

Thus, a correct answer required a mac instruction up to 3 instructions after its rd got updated, and no other hazards involving other registers. Our staff solution was:

```
mac a0 a1 a1

mac a0 a1 a1
```

Candice Yang STAFF 7mth #986bdb

We don't need to initialize registers in this problem specifically because no matter how we initialize the registers, we still need to stall between the two mac instructions. If we are writing normal RISC-V code, we will definitely need to initialize registers.

mul doesn't need to wait for `rd` being written back because mul is `rd=rs1*rs2`, so it doesn't need to read `rd`. But `mac` is `rd = rd + rs1*rs2`, so the second `mac` needs to stall in order to read the correctly updated `rd`.

1

Anonymous Mantis 7mth #986cda

so if we had something like

mul rd rs1 rs2
mul rd rd rs2

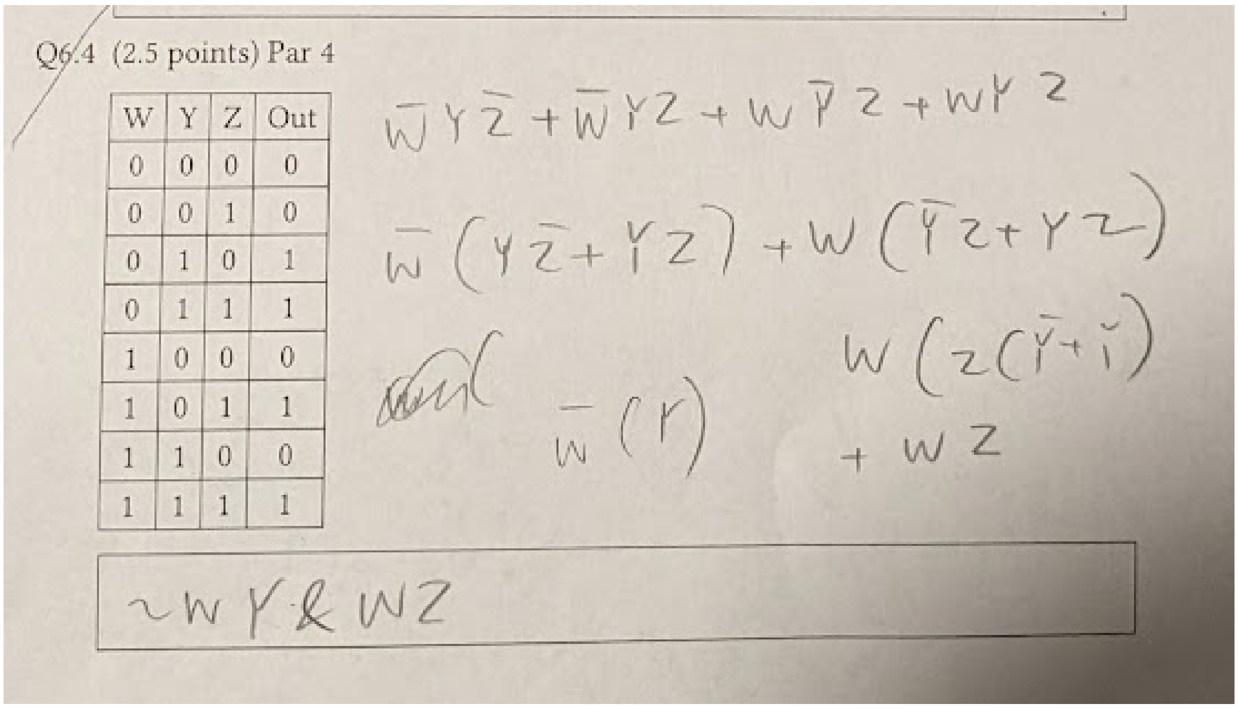
then we don't have to worry about a data hazard?

A

Adrian Bao 7mth #986ca ✓ Resolved

FA21-Final-Q6.4

Is this equivalent to the answer? If not, please let me know what's wrong. Thanks!



Q6.4 (2.5 points) Par 4

W	Y	Z	Out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$\bar{W}Y\bar{Z} + \bar{W}YZ + W\bar{Y}Z + WYZ$

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

[illegible] $\bar{W}(Y)$ $W(Z(\bar{Y} + Y))$

$+ WZ$

~WY&WZ

Q6.4 (2.5 points) Par 4

W	Y	Z	Out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Solution: Staff solution: $(\sim W\&Y)|(W\&Z)$

Other answers may be possible. Congratulations to the three students who found the following solution in 3 operations: $Y \wedge (W\&(Y \wedge Z))$

In this case, we can't easily do our analysis like we did in earlier questions. This can be either solved by starting with the sum-of-products form and simplifying, or by noting that the patterns when W is on and when W is off are both simple.

Author's note: This question was constructed by random number generator.

M Myrah Shah STAFF 7mth #986ce

Seems like you have the correct answer on the last line of your work, and a different answer in the box, how did your OR become AND?

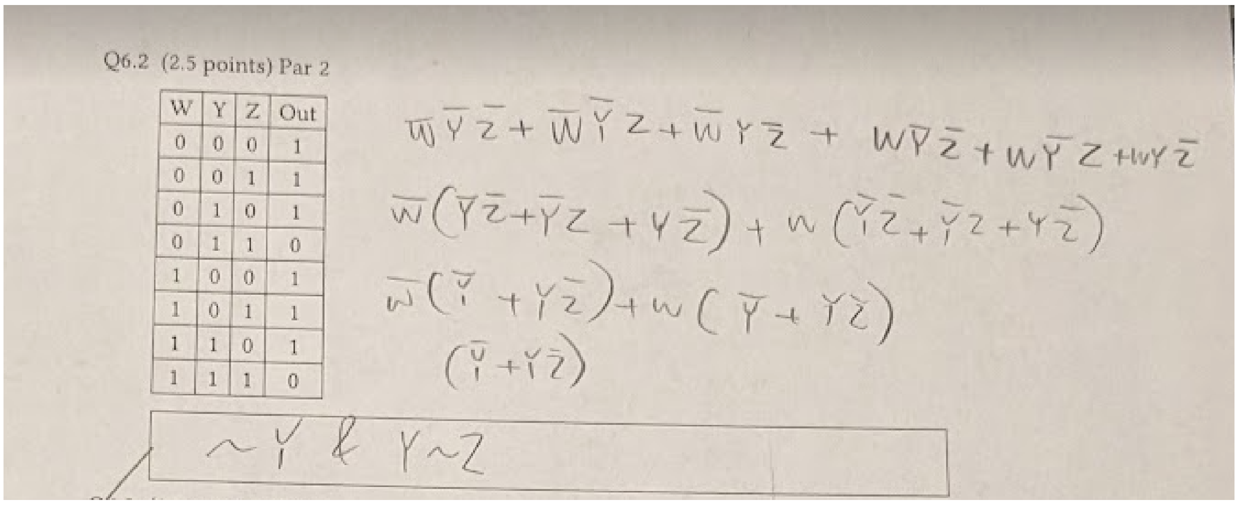
A Adrian Bao 7mth #986cf

Yep typo on my part. Thanks!

A Adrian Bao 7mth #986bf ✓ Resolved

FA21-Final-Q6.2

Is this equivalent to the answer? If not, please let me know what's wrong. Thanks!



Q6.2 (2.5 points) Par 2

W	Y	Z	Out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

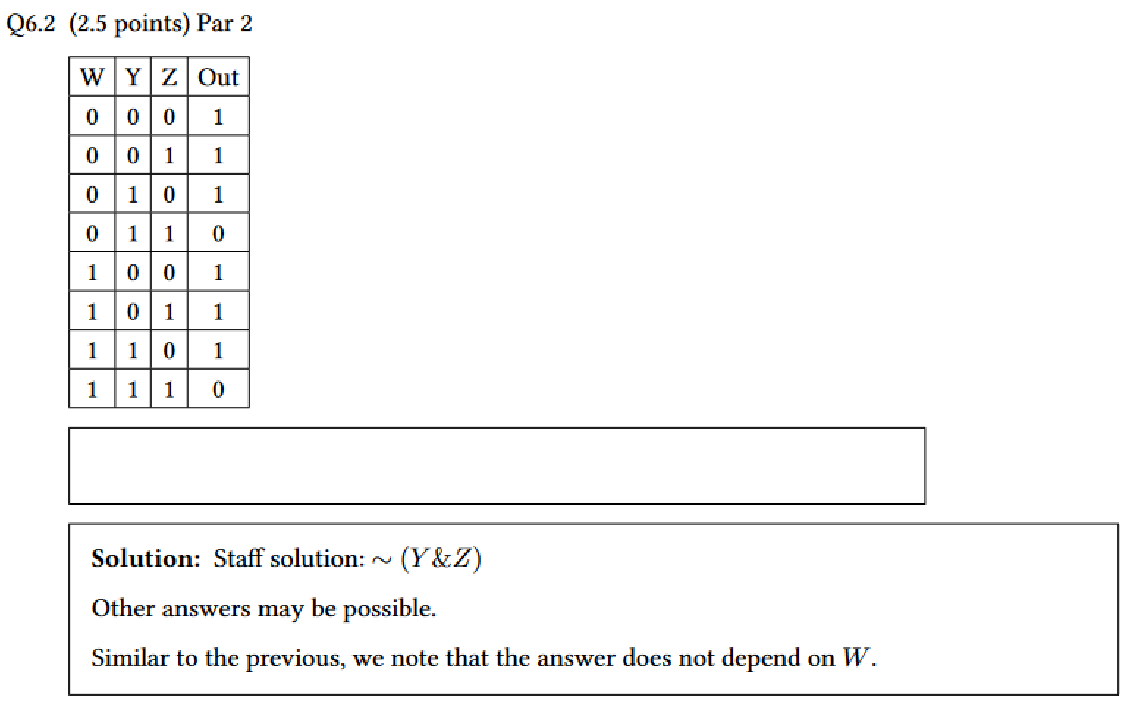
$\bar{W}\bar{Y}\bar{Z} + \bar{W}\bar{Y}Z + \bar{W}Y\bar{Z} + W\bar{Y}\bar{Z} + W\bar{Y}Z + WY\bar{Z}$

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

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

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

~Y & Y~Z



Q6.2 (2.5 points) Par 2

W	Y	Z	Out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Solution: Staff solution: $\sim(Y \& Z)$

Other answers may be possible.

Similar to the previous, we note that the answer does not depend on W.

Anonymous Wombat 7mth #986baf ENDORSED

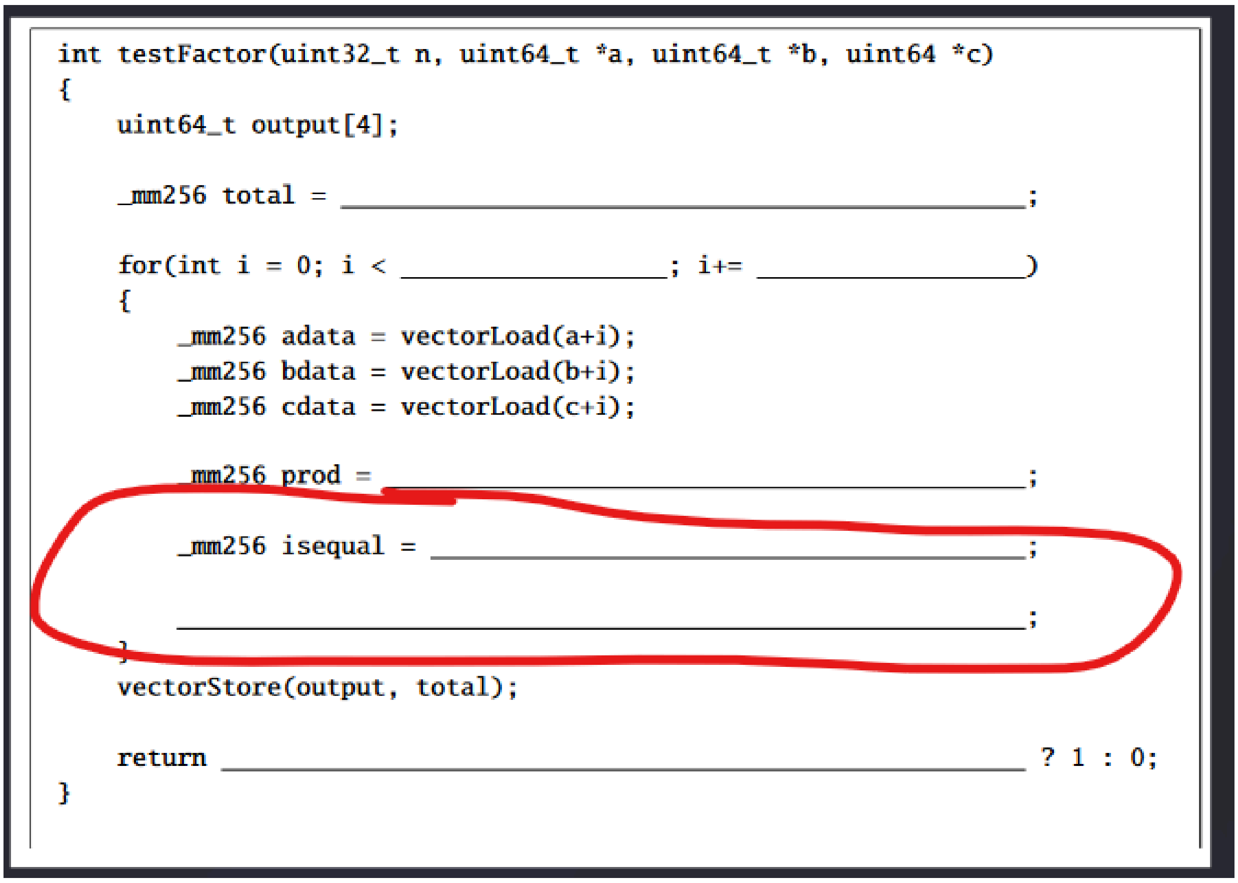
It should be the same, you can use the following law:

$$X + YZ = (X + Y)(X + Z)$$

In this case, you would get: (~Y + Y)(~Y+ ~Z) = (~Y+ ~Z) = ~(Y & Z)

A Adrian Bao 7mth #986be ✓ Resolved

FA21-Final-Q9



```
int testFactor(uint32_t n, uint64_t *a, uint64_t *b, uint64 *c)
{
    uint64_t output[4];

    _mm256 total = ________________________________________;

    for(int i = 0; i < _______________; i+= _______________)
    {
        _mm256 adata = vectorLoad(a+i);
        _mm256 bdata = vectorLoad(b+i);
        _mm256 cdata = vectorLoad(c+i);

        _mm256 prod = ____________________________________;

        _mm256 isequal = _________________________________;

        __________________________________________________;
    }
    vectorStore(output, total);

    return ______________________________________________ ? 1 : 0;
}
```

Q9.5 (isequal)
Why is it vectorXor instead of vectorOr since if a bit of prod and a bit of cdata is *both* 1 and that is Xored, it would be false (0), which suggests it is not equal.

Then, only if a bit of prod and a bit of cdata is different, then isequal is **true (1).** But isn't it supposed to be if a bit of prod and a bit of cdata is different, then isequal is **false (0)?**

Then, when Xoring, if there is a mismatch in any bit, i.e. 111011111, then the program would return 0. However, that is not the case since I think vectorXor incorrectly finds mismatches.

E Eddy Byun STAFF 7mth #986dc
When we return, we check to see that each chunk in the SIMD vector is equal to 0. If it is, then we return 1, else return 0.
1

Anonymous Horse 7mth #986bb ✓ Resolved
FA21-Final-Q2.2

why is the maximum value 66?

and for FA21-Final-Q7, is any of this in scope?

J Justin Yokota STAFF 7mth #986bc
Based on our rounding rules, we end up evaluating 64+2=64. So we never exceed 64.

Parts of this is in scope; you should be able to solve an equivalent problem using direct-mapped or fully associative caches, for example.

Adrian Bao 7mth #986aab

I thought this happens with 8 (1000), since it is the middle of 6 (0110) and 10 (1010). Thus, 6 + 2 always rounds down to 6 and gets stuck there.

Justin Yokota STAFF 7mth #986aaf

Replying to Adrian Bao

8's a representable number, since it's a power of 2. We only round if the number isn't representable.

1

Anonymous Horse 7mth #986ab ✓ Resolved

FA21-Final-Q1.6,1.5,1.12

Just to make sure, are these in scope?

Eddy Byun STAFF 7mth #986af

1.5 is in scope. 1.6 and 1.12 are out of scope

Anonymous Horse 7mth #986ba

Oh Sorry I meant 1.6 and 1.7

Eddy Byun STAFF 7mth #986bd

Replying to Anonymous Horse

1.6 and 1.7 are out of scope

Anonymous Opossum 7mth #986aa ✓ Resolved

SP21-Final-Q1

Just to confirm, since question 1.2 is about availability (lec 37), is the question out of scope?

Eddy Byun STAFF 7mth #986ae

Spring 2021, question 1.2 is out of scope.

1

Anonymous Crow 7mth #986e ✓ Resolved

SP21-Final-Q9.1, why can't we just use the ALU and use a XOR gate with all 0's and check if the output is all 1's meaning it is null. What is the intuition for branch because I don't see how we would write back the singular bit

Justin Yokota STAFF 7mth #986f

XOR with 0 does nothing to the input. Further, how do you check if the output is all 1s? The only equality block is in the branch comparator.

Anonymous Horse 7mth #986a ✓ Resolved

SP21-Final-Q2

I see parts A~C is about set-associative caches, but are the rest of the parts out of scope too?

Justin Yokota STAFF 7mth #986b

Parts D and E are not technically in scope, but would be if we changed the cache layout to a direct mapped cache or fully associative cache (as a checksum, if you multiply together the hit rates you get by solving parts D and E as a fully associative cache and write it as a decimal, you'll see the string "7916" within the first ten digits).