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: **SP21-Final-Q1**, or **SU21-MT2-Q3**

Spring 2021 final walkthrough

Added 2022/08/10:

**Extremely rough draft SP21 final solutions**

**Extremely rough draft SU21 midterm solutions**

**Extremely rough draft SU21 final solutions**

---

**Anonymous Gnu** 2mth #1312add

**FA21 Final Q6.2**

How can we tell when the answer does not depend on W? Is there a fast way to tell?

...
Please tag your post! (FA21-Final-Q8.4)

The question states: "physical pages get assigned in order of physical page number (so page 0 is assigned first, then page 1, and so on)."

What type of miss do we classify a miss as if the cache table has the correct tag but the valid bit is set to zero? Specifically referring to Fa21 7.6, 6th entry here.

You can think of the valid bit off as the cache block being "blank" (nothing has ever entered that index before).

right, but the question asks us to bubble "all types of misses that this is" (i paraphrased here); so which would be the correct boxes to cross off?

FA21 Final Q6.3 do we ignore the X cases and only focus on situations where the answer is definitely 1?

For the rows with X, your expression can output either 0 or 1.

FA21 Final Q5.3
Is a valid answer just one mac instruction? I am not fully understanding what was meant by "Thus, a correct answer required a mac instruction up to 3 instructions after its rd got updated, and no other hazards involving other registers." Why is there a specification on 3 instructions?

Why is the number change from 0b1.11111 to 0b10.000?

Here 1.1111 = 1 + 1/2 + 1/4 + 1/8 + 1/16, then we add 1/32, but here we will have 10.000 is closer to that (the real result is 2 - 1/32, btw) and has less significant 0 at the end. Thus its 2 and converges from above (since from now on comparing any number to 0b10 will give 0b10 being the answer chosen for the 0 bit) Hope it helps :)
0b1.11111 cannot be represented in this system since we only have 4 mantissa bits. This number instead rounds up by adding 0b0.00001 to get 0b10.0000.

---

**Anonymous Gnu** 2mth #1312fc

**SP21-Final-Q5**

A!A + !(ABC + !CA) = 0 + !A!B!C x C!A, so when !C times C, isn't the whole thing 0?

---

**Zipeng Lin** 2mth #1312fe

it should be equal to A!A + !(ABC + !CA) = (!A + !B + !C)(C + !A) = !A C + !A + !B C + ! C ! A + !C C

= !A (C + !C) + !A + !B C + 0 = !A + !B C.

You seem to expand the last step wrong.

---

**Anonymous Gnu** 2mth #1312aaa

From (!A + !B + !C)(C + !A) to !A C + !A + !B C + !C ! A + !C C, where did the !A!B go?

---

**Zipeng Lin** 2mth #1312aab

Sorry my bad, there should be an !A !B, but it gets absorbed by !A in the last step: !A + !A !B = !A (1 + !B) = !A 1 = !A

---

**Anonymous Gnu** 2mth #1312aac

What about the first option?

I simplified it to:

(!A + !B + C) * (C + !A) = !A + !B C + C

---

**Zipeng Lin** 2mth #1312aad

the first term in the product should be (!A + !BC) * (C + !A) = !A C + !A + !B C C + !BC !A = !A (absorption) + !BC.

The left side thing under the biggest not should be: !(A * !D) = A + D, here D = !BC + C B !B = !BC.

TBH, for problems like this, just draw a truth table. It will save much time :)

---
Anonymous Gnu 2mth #1312aaf
Replying to Zipeng Lin
Sorry for the endless question, but how does becomes (!A + !BC)? I thought CB!B = 0, so it is !(A * !(BC)) = !(AB) = !(A + !B + C)
...

Zipeng Lin 2mth #1312aba
Replying to Anonymous Gnu
first take out the big no, it becomes

!A + !(B + C + B !B) (notice how we have 2 nos outside, one originates from the problem, the other comes from splitting)

Then, it is

!A + !B C + C B !B = ! A + !B C
...

Anonymous Gnu 2mth #1312abb
Replying to Zipeng Lin
I see. So you have to take out the big no then do the small one right?
...

Zipeng Lin 2mth #1312abc
Replying to Anonymous Gnu
you could do it either way. The way I showed might be more convenient.
...

Peyrin Kao STAFF 2mth #1312adb
My suggestion for solving this is to write out the truth table for each expression, and see which ones match.
...

Anonymous Otter 2mth #1312fa Resolved
FA21-Final-Q2.4: The question says we have 4 significand bits, but why in the answer here we are using 5 significand bits? Is that a typo or there is a specific reason why? Thanks.
...

Anonymous Otter 2mth #1312abd
nvm, i think i understand it now
...

Peyrin Kao STAFF 2mth #1312adc
I believe they're showing you the first non-representable integer.
...

Anonymous Seal 2mth #1312ed Resolved
FA21-Final-Q7.4: couldn't every address technically be a compulsory miss depending on what is actually in the cache because we don't know if the specific bytes of data have been accessed before?

...  

\[ \text{Peyrin Kao} \quad \text{STAFF} \quad 2mth \ #1312ee \]
Yes, every address could be a miss.

...  

\[ \text{Anonymous Seal} \quad 2mth \ #1312ef \]
Could the ones that the solution says hit also be conflict misses for the same reason then?

...  

\[ \text{Peyrin Kao} \quad \text{STAFF} \quad 2mth \ #1312fb \]
\[ \text{Replying to Anonymous Seal} \]
If it's in the cache, it can't be a miss.

...  

\[ \text{Anonymous Mandrill} \quad 2mth \ #1312eb \quad \text{Resolved} \]
Does byte-addressed system mean that every address's size is a byte? And does "a 64-bit machine" mean that our virtual memory is 64bit?

...  

\[ \text{Peyrin Kao} \quad \text{STAFF} \quad 2mth \ #1312ec \]
It means each address refers to one byte. 64-bit machine means that addresses are 64-bits (we don't have to worry about virtual memory in this question).

...  

\[ \text{Zipeng Lin} \quad 2mth \ #1312dd \quad \text{Resolved} \]
SP-21-Final Q9.9 explanation? Does control flow cause more cache accesses without locality and thus less hit rate?

...  

\[ \text{Peyrin Kao} \quad \text{STAFF} \quad 2mth \ #1312de \]
Fewer branches = better locality because you're accessing instructions in order.

...  

\[ \text{Zipeng Lin} \quad 2mth \ #1312dc \quad \text{Resolved} \]
For SP21-Final-Q7, why don't we consider the path between those two registers? (it would just be clk-q delay +and, so it is shorter?)

...  

\[ \text{Peyrin Kao} \quad \text{STAFF} \quad 2mth \ #1312df \]
I don't see this question on SP21 final, but this does look like a valid path between registers.
Anonymous Mandrill 2mth #1312db Resolved
Shouldn't this be shortest clock period = ctoq time + largest com delay + setup time? It should be setup time instead of hold time, right?

Peyrin Kao STAFF 2mth #1312ea
Yeah, that's a typo sorry.

Anonymous Otter 2mth #1312cd Resolved
For sp21-final-7(c)iii, I couldn't find it in the walkthrough video, but can I ask why we choose the first option instead of the third one? Thanks!

Peyrin Kao STAFF 2mth #1312ce
My thinking is that you should never have IMEM and DMEM addresses that are close by, because the code and static/stack/heap sections of memory are far apart.

Anonymous Ram 2mth #1312ca Resolved
SP21-Final-Q7(a).C. The question asks for implementing as a pseudo instruction. Why the control logic should be modified? If it is a pseudo instruction, isn't it translated already in assembly stage? And this instruction will not appear in the datapath.

Peyrin Kao STAFF 2mth #1312cf
Yeah, I agree with your reasoning.

FA-21 Final Q3
I'm unsure why we need to account for the 24 bytes in line 1 since the stack pointer doesn't get moved back up from its original position at the bottom of the stack at any time before these instructions are injected and jumped to, so those the sp should still be -24 and only needs to be moved -256 (unless Get20Chars doesn't follow calling convention I guess?)
Anonymous Viper 2mth #1312be
The code (5 instructions) are injected into our 20 byte buffer. Then after this the address of the first instruction is written past the buffer so lw ra 20(sp) has the address of the first instruction

Before we even start executing these 5 lines of code with jr ra we add 24 to the sp. The problem says to allocate a buffer that doesn't overlap with these 24 so we move it down -24 + -256 = -280. Hope this helps!

James DeLoye 2mth #1312cb
Oh nevermind lol I completely misinterpreted the code, thanks

Anonymous Viper 2mth #1312bb
Resolved

Fall 21 Final Q6b) part v

I believe there is an error on the answer key:

The size of this struct is 8 bytes since we are working on a 32 bit system. Each pointer is 4 bytes
t0 = sl->next
t1 = level

We want to get sl->next[level]

Since we need to manually do the pointer arithmetic and sizeof(SLN) = 8
slli t1 t1 3
add t0 t0 t1
lw t0 0(t0)

But the solution is

Am I thinking about this wrong or is the solution incorrect?

Peyrin Kao STAFF 2mth #1312bc
The solution is wrong yeah.

Anonymous Otter 2mth #1312aa
Resolved

Sp21-Final-Q4 i, I know D is one of the correct answers, but how do we choose between A, B, and C? What are the differences?
I don't see ABCD answer choices on this question.

Anonymous Otter  2mth  #1312bf
It's this question, thanks!

Anonymous Viper  2mth  #1312cc
Replying to Anonymous Otter
I didn't know how to do this one either until I checked out the video. I suggest skipping around in the provided video to questions you're not comfortable with also. But the reason is because B and C result in data races. say we have 5 threads and we split up the work among the 5000/2 iterations. There will be a point where one thread will be a value say i = 998 and another thread will have the value i = 1000

so \[ a[1000] = a[1000] \times a[998] \]

and the other thread will execute

\[ a[1002] = a[1002] \times a[1000] \]

The result depends on the order the threads execute their instructions which is random so the program is incorrect.

Anonymous Otter  2mth  #1312f  Resolved
Sp21-Final-Q2B, how do you approach this problem? Thanks!

Anonymous Viper  2mth  #1312e  Resolved
SP-21 Final Q2i

I don't understand why this question asks after the first loop iteration. I'm assuming its the inner for loop of the top most nested loops. Also wont the page table just take up however much it does per entry multiplied by the number of entries. Do we need to analyze the loop?

VPN = 20 bits

so

\[ 2^{20} \text{ entries} \]

Each entry has 8 bits of metadata and the PPN of 12 bits so 20 bits per Page Table Entry.

So
Would I then divide this by the number of bits per page. I know this reasoning is incorrect but I just wanted to show my thought process. What would be the correct approach?

- Bolong Zheng 2mth #1312ab • ENDORSED
  I think 20 bits could get padded to 32 bits to keep 4-byte alignment. In that case, that would be $2^{20} \times 4$ bytes, which if divided by 4kib, is the desired answer.

  If it's a byte-aligned system, 20 bits get padded to 24 bits = 3 bytes. Then $2^{20} \times 3 / 2^{12} = 768$, which is the alternative ans

- Anonymous Viper 2mth #1312ac
  That makes sense thanks!

- Bolong Zheng 2mth #1312ad
  Replying to Anonymous Viper
  Also I think the the code snippet is irrelevant to the solution. Asked a TA and it's probly just there to trick people except the last question on this part.

- Peyrin Kao STAFF 2mth #1312af
  It doesn't matter that you ran this after the first iteration of the loop, I think. The single-level page table needs to be there and be allocated the entire time the program is running. It might be to contrast against the two-level page tables, where some page tables only get allocated later in the program.

- Anonymous Mandrill 2mth #1312b • Resolved
  Fa21-Final-Q1.12. I couldn't understand the solution. I know that 0x71c and 0x51c is only 1 bit hamming distance between 0x61c, so we don't choose them. What if we have 0x01c, which is two bits humming distance with 0x61c?

- Peyrin Kao STAFF 2mth #1312c
  #1201a

- Anonymous Cattle 2mth #1312a • Resolved
  FA21-Final-Q7.4

  When we try to access 0b 1111111111, the tag and index are in the cache, but the valid bit is 0. Is it a compulsory miss?

  We categorize a miss as compulsory miss only when the address is not in the cache OR the valid bit is 0, right?

  Thank you!
The cache in this question doesn't have a valid bit.