[Final] Past Exams - 2020 and Older #987

E Eric Che STAFF
7 months ago in Exam – Final

2

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 Snail 7mth #987abe ✓ Resolved

Fa 19 q8

b) What is the hit rate for the code above? Assume C processes expressions left-to-right.

50%

SHOW YOUR WORK

Every iteration it's
ARRAY[i] read MISS
ARRAY[i+1] read HIT
ARRAY[i+256] read CONFLICT→ MISS
ARRAY[i] write CONFLICT→ MISS
ARRAY[i] read HIT
ARRAY[i] write HIT
3 MISSES, 3 HITS. 50% hit rate.

Why does arr[i+256] miss and then why does arr[i] miss again next line?

♡ ···

S Sonika Vuyyuru STAFF 7mth #987abf

We are given in the question that the cache has 16-byte blocks. 16-byte blocks can hold 4 ints. Therefore, when we pull in the block from the previous lines, ARRAY[i+256] is not pulled into that cache block and is a miss. When we try to pull in ARRAY[i] again in the next line, it is no longer in our direct-mapped cache since it was kicked out by the block containing ARRAY[i+256], so it is a miss again.

♡ ···

Anonymous Snail 7mth #987adb

Does that mean arr[i+256] and arr[i] share the same index? What do you mean by "ARRAY[i+256] is not pulled into that cache block"?

Also why would they share an index?

♡ ···

J Jackson Wei STAFF 7mth #987add

Replying to Anonymous Snail

arr[i+256] and arr[i] share the same index. We can see this by breaking down the TIO bits. We have 1 KiB sized direct mapped cache with 16 byte blocks. So this means that we would have 6 index bits and 4 offset bits, with the rest being tag. Now if we were to add 256 x 4 (basically array[i] vs array[i+256]) (ints are 4 bytes) to any address, we would see the address go from something like ...0**1011101010** to ...1**1011101010**. Note that this only changes the tag as the lower 10 bits are untouched, so they map to the same index in the cache but have different tags, leading to conflict miss and kicking each other out.

♡ 1 ···

Anonymous Snail 7mth #987ade

Replying to Jackson Wei

I see. Thank you!

♡ ···

Anonymous Scorpion 7mth #987abb ✓ Resolved

su20-final-4d

(d) (3.0 pt) How long will y_output remain equal to 1 before switching to 0?

41

If y_output changes to 1 at 102, the next rising edge is at 125. From there, it takes Clk-Q (4) + OR (14) + AND (9) = 27 ns to update y_output to 0 again, so at 125 + 18 = 152 ns, 152 - 102 = 50 ns

Shouldn't the answer be 50 according to the solution?

♡ ···

S Sasha Singh STAFF 7mth #987acf

yep, this seems like a typo!

♡ ···

Anonymous Scorpion 7mth #987aae ✓ Resolved

[Fall 2019 Final Q7] Can anyone explain why insert 3 nop times instead of 2 nop times? (Because I think there are only 2 instructions between branches "greater"taken)

Case 2: *After fixing that hazard,* the following case fails:

```
      addi  s0 x0 4
      slli  t1 s0 2
      bge   s0 x0 greater
      xori  t1 t1 -1
      addi  t1 t1 1
greater:
      mul t0 t1 s0
```

When this test case is run, `t0` contains `0xFFFFFFC0`, which is not what it should have been.
Pro tip: you shouldn't even need to understand what the code does to answer this.

c) What caused the failure? (select ONE)	**d)** How could you fix it? (select all that apply)
● Control Hazard	■ Insert a nop 3 times if you detect this specific error condition
○ Structural Hazard	☐ Forward execute to write back if you detect this specific error condition
○ Data Hazard	☐ Forward execute to memory if you detect this specific error condition
○ None of the above	☐ Forward execute to execute if you detect this specific error condition
	■ Flush the pipeline if you detect this specific error condition

The issue with the code above is we do not clear/flush the instructions if the branch determines it is taken. Remember that we are running on a five stage pipeline CPU which just assumes PC + 4 unless an instruction says otherwise. This means that we will not determine if the branch is taken until the branch is in the execute phase. This means that we will have the next two instructions already in the pipeline (one in instruction fetch, the other in instruction decode). So we have a Control Hazard as we are not executing the correct instructions. Some ways how to fix it: insert nops if you detect a branch instruction in the instruction fetch stage OR flush the pipeline if the branch is in the opposite direction of what was predicted. Forwarding data in this case will not help at all.

♡ ···

S Sonika Vuyyuru STAFF 7mth #987aca

The number of stalls here has more to do with how many stages in our pipeline we need to wait for than how many instructions are between the the branch and "greater" line. Specifically, we will not determine if the branch is taken until the branch is in the execute phase, so we will need to insert a nop 3 times until we figure out this information.

♡ ···

Anonymous Viper 7mth #987fb ✓ Resolved

SU-18-MT2-Q4

Why is there only one stall for the control hazards? If there are three stages (IFD, EX, MWB) and there's no double pumping, then wouldn't we have to wait until after the MWB of the previous instruction before we can start IFD/fetching the PC for the next instruction? That would make two stalls.

Consider the following piece of code

```
1.        add t1 x0 x0
2.        add t2 x0 x0
3.        addi a0 x0 2
4.        slli a0 a0 2
5.  L2:   bge t1 a0 End
6.        add t3 sp t1
7.        lw t3 0(t3)
8.        add t2 t2 t3
9.        addi t1 t1 4
10.       j L2
    .
    .
    .
End:
```

1) Assume that instead of branch prediction, the CPU always stalls to resolve a control hazard. How many stall(s) are necessary for each control hazard **each time** it is encountered? You may not need all boxes.

Line Number	# Stalls/Encounter
5	1
10	1

♡ ···

M Minh Nguyen STAFF 7mth #987abc

The CPU is stalling until it knows the branch decision to fetch the next correct instruction. When `bge` is at the `EX` stage, the next instruction would normally be fetched. However, since we don't know the branch outcome, we must stall for one cycle. The branch decision is confirmed at the end of `EX` stage, so the pipeline doesn't need to wait until MWB stage.

♡ ···

Anonymous Viper 7mth #987abd

Why do we normally have to take 3 stalls with a 5 stage pipeline then? Can't we just take two stalls if we already have the branch decision during MEM?

♡ ···

Brendan Roberts STAFF 7mth #987adf

Replying to Anonymous Viper

In a 5 stage pipeline, the output of the ALU isn't passed to the PC until after the pipeline register. Therefore, when the branch instruction is in MEM, the correct PC value will be sitting at the PC register but the PC register won't read it until the following clock cycle. This leads to 3 stalls.

♡ ···

Anonymous Porcupine 7mth #987ef ✓ Resolved

SU20-Final-Q5b, why is A the correct answer? Also, if A is correct, wouldn't that make option B correct as well?

and for part 5biii, why is choice G and D correct?

♡ ···

S Sonika Vuyyuru STAFF 7mth #987acb

For Q5bi: In the question, we are given that if the jas instruction is ran, the corresponding named signal is set to 1. Choice A behaves in this way, where if jas=1, the selector bit chooses spMinus4, as that is the value that should be written to sp if jas is ran. On the other hand, Choice B behaves opposite, and would set sp_wb to sp when jas=1, which is not what we want. This is because of the not that is placed on the signal before it becomes the selector bit.

For Q5biii: The save to the stack operation should behave such that when jas=1 (meaning jas instruction is taken) we want to set the write data to be pc+4 and the memaddr to be sp-4. This is the behavior described in the question. If jas is not taken, we want the behavior of the circuits to be the same. Choice D does this for MemAddr and Choice G does this for WriteData.

♡ ...

Anonymous Porcupine 7mth #987ee ✓ Resolved

FA19-Final-Q5: how did we get from the first line to 2nd and 2nd to third? I thought the inner expression should be ~A + 1 and not (A+1) like they have here. Also, how can we just delete that (A+1) from the overall expression?

$$\text{out} = \overline{A}B\overline{C} + \overline{A}\,\overline{C}\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D} + B\overline{C} + \overline{C}D \quad \textbf{(Inverse)}$$
$$\text{out} = B\overline{C}(A+1) + \overline{A}\,\overline{C}\,\overline{D} + \overline{B}\,\overline{C}\,\overline{D} + \overline{C}D \quad \textbf{(Distributive)}$$
$$\text{out} = \overline{C}(B + \overline{A}\,\overline{D} + \overline{B}\overline{D} + D) \quad \textbf{(Distributive)}$$

could someone also explain these last two?

$$\text{out} = \overline{C}((B + \overline{B}\,\overline{D}) + (\overline{A}\,\overline{D} + D)) \quad \textbf{(Associative)}$$
$$\text{out} = \overline{C}(B + \overline{D} + \overline{A} + D)$$
$$\text{out} = \overline{C}(\overline{A} + B + 1) \quad \textbf{(Inverse)}$$
$$\text{out} = \overline{C} \quad \textbf{(Identity)}$$

♡ ...

S Sonika Vuyyuru STAFF 7mth #987acc

1st to 2nd line: Group like terms together and factor out variables using the distributive law. For example, bringing together the first term and the fourth term of line one and factoring out B*~C is how you get the first term of the second line. When you factor out the B*~C, nothing changes about (A+1) since we are not taking out an extra not or anything. We are just factoring out common variables in both terms.

2nd line to 3rd line: Same idea. We are factoring out a common variable in all the terms, which is ~C.

The reason we can just "delete" A+1 is because of the additive identity of boolean algebra. A+1 is saying A or 1, which we know is always true aka 1. So it's like multiplying B*~C by 1.

For your second question, in the first picture, we are able to make this simplification because of the absorption law.

In the second picture, it is the same reasoning as above regarding the additive identity rule.

♡ ...

Anonymous Dolphin 7mth #987ed ✓ Resolved

SP20-FINAL-Q4a

How did they arrive at 2^20 PTE's?

Question 4 ***Virtual Reality! I mean Memory...*** **(8 points)**

Consider a system with 2 MiB of physical memory and 4 GiB of virtual memory. Page size is 4KiB. Recall that the single level page table is stored in physical memory and consists of PTE's, or page table entries.

Q4.1 (3 points) If we choose to store seven information bits in each PTE, how big is the page table in bytes?

Solution: 2^{21}
VPN contains $log(\frac{2^{32}}{2^{12}}) = 20$ bits
PPN contains $log(\frac{2^{21}}{2^{12}}) = 9$ bits
9 bit PPN + 7 info bits = 16 bits per PTE
2^4 bit PTE $\cdot 2^{20}$ PTE's $= 2^{24}$ bit page table, or 2^{21} bytes

♡ ···

S Sonika Vuyyuru STAFF 7mth #987acd

The page table contains a mapping for every possible VPN. Since the VPN contains 20 bits, there must be 2^20 VPNs, and thus we need 2^20 PTEs.

♡ ···

Anonymous Porcupine 7mth #987ec ✓ Resolved

SP18-Final-Q5

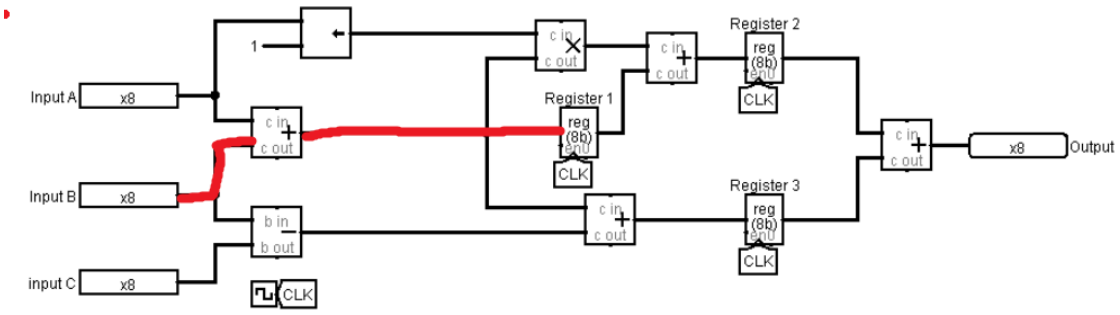
Where can we access the solutions for this? it doesn't seem to let us see the answers.

part a) why isn't the maximum possible hold time 8ps?
part b) why isn't the min possible clock period 19ps?
part d) how can we go about solving this one?

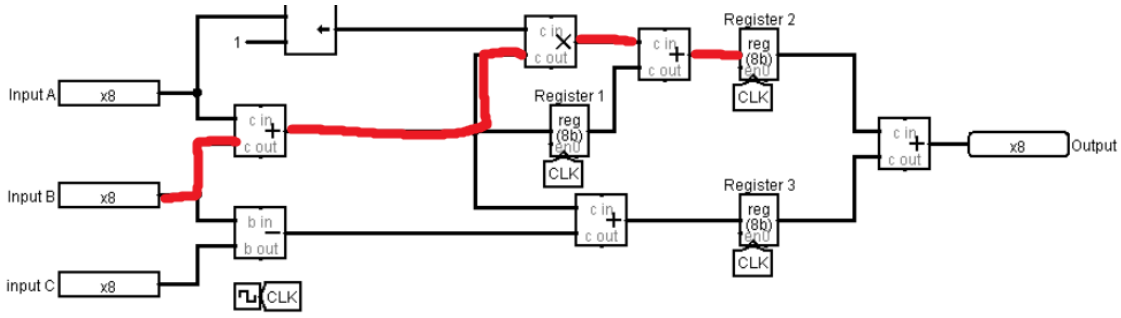
♡ ···

J Jackson Wei STAFF 7mth #987ada

part a) Since the inputs A, B, and C take on their new values exactly at the rising edge of every clock cycle, they are kinda like registers that have a clk-to-q delay of 0. And since hold time is just the amount of time the inputs to any register needs to be held stable after the rising edge, for a path like the one below, the input to register 1 changes just after 5ps (0 clk to q + 5 adder delay), so the max amount of time register 1 can expect its input to be held stable is 5ps (max hold time).



part b) for the max cl delay path, its from input B to register 2, so that adds up to 0 clk-to-q, 5 + 6 + 5 CL, and 4 setup time which adds up to 20ps.



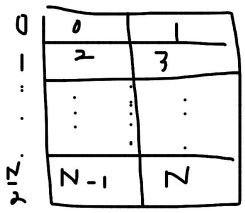
part d) we want to know the amount of time it takes for the output to be computed. Since this is pipelined, and clk period is 11ps, we see that in clk cycle 1 we will reach registers 4,5 and 6. In clk cycle 2, we will reach register 1, and then in clk cycle 3, we will reach register 2. At this point, after 3 clk cycles, register 2 has the correct value stored, it just needs a clk-to-q delay + adder delay to properly propagate to the output. This is 3 x clk cycle + 3 clk to q + 5 adder which adds up to 41.

♡ ···

Anonymous Dolphin 7mth #987eb ✓ Resolved

SP20-Final-Q2b-iv

Why is it that the answer won't change when the cache is a 2 way set-associative? I have a hard time visualizing the way the solution explains. I was thinking that if the cache has 2 cols with N/2 rows each, the first iteration of the outerloop would fill in the entire cache with all N values then when we iterate another outerloop all values would already be in the cache. eg:



♡ ···

Jackson Wei STAFF 7mth #987adc

Yea I think you are right in that the first iteration of the outer loop would fill in the entire cache with all N values. But this is also what happens if we have a fully associative cache. The answer to part I and II hint at the fact that the first iteration of the outer loop are all compulsory misses (N misses) and then the 29 iterations of the outer loop after that are all hits (29N hits). So the cache being 2 way set associative wouldn't change anything here.

♡ ···

Anonymous Dolphin 7mth #987aea

I was referring to part b, when there are N/2 rows instead of N now. How would that change in that case?

♡ ···

Wesley Kai Zheng 7mth #987ea Unresolved

SP18-FInal-Q10d.

I do not understand how they got the speed up factor to be 7.5X. In my calculations, I figured as n approaches infinity, the unrolling one takes around 5/4 n to run, while the original one takes 6n to run. So the speed up factor, according to my calculations is calculated by 6/(5/4) = 24/5 = 4.8X. It seems like that answer used 4/5 instead to be divided by 6 to get 24, which I am very confused about.

♡ ···

Anonymous Squirrel 7mth #987df Unresolved

Sp19-MT-Q5

Why is lb and shw being stalled 4 times and not three since by hardware we write first then read wouldn't D be aligned with W of the previous row?

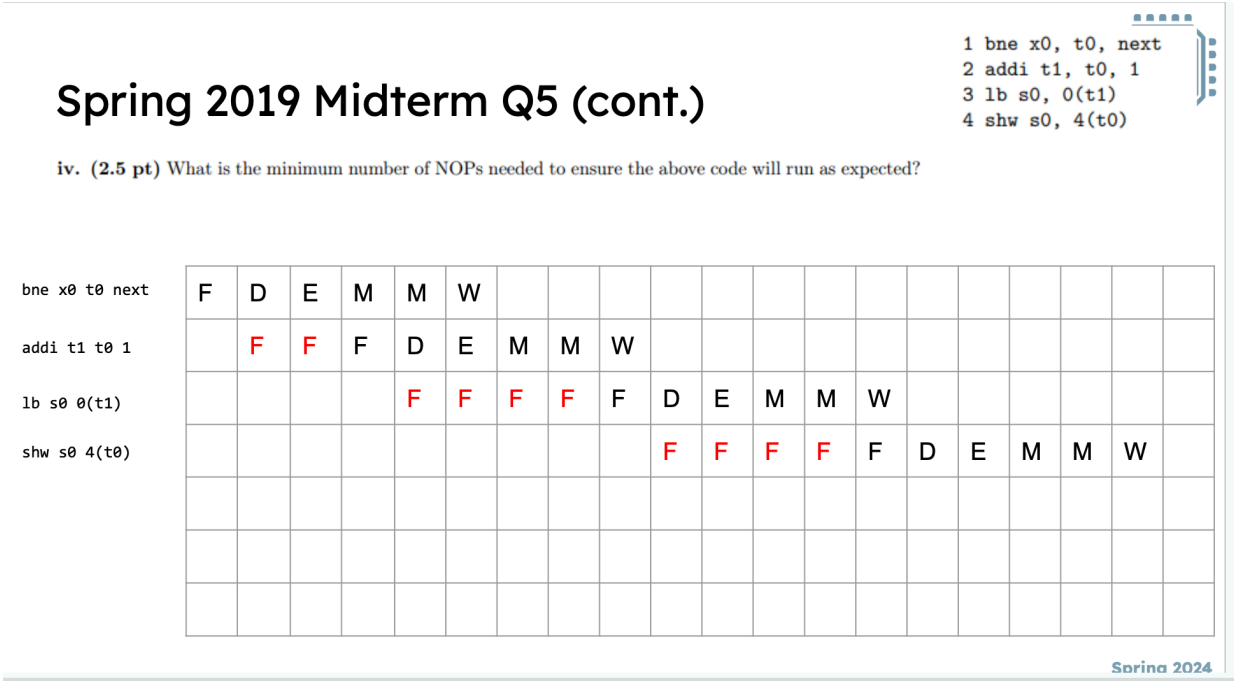
♡ 1 ···

Anonymous Scorpion 7mth #987fd

#987fc

♡ ···

Anonymous Antelope 7mth #987de ✓ Resolved



SP19-MT-Q5

I don't understand the logic behind this problem. Why are we stalling for 2 cycles for addi and then 4 cycles for both lb and shw?

♡ 1 ···

Anonymous Scorpion 7mth #987fc

Tell me if I'm wrong.

- `addi` 's Instruction Fetch needs to come after `bne` 's Execution (not sure why since the branch is taken, which means `addi` will be skipped?)
- `lb` 's Instruction Decode needs to come after `addi` 's Writeback to get correct `t1`
- `shw` 's Instruction Decode needs to come after `lb` 's Writeback to get correct `s0` .

♡ ···

Anonymous Squirrel 7mth #987fe

I agree with your first statement but in lecture didn't they discuss that we hardwire the datapath so that its always write first then read/decode. In that case wouldn't the decode be right underneath W

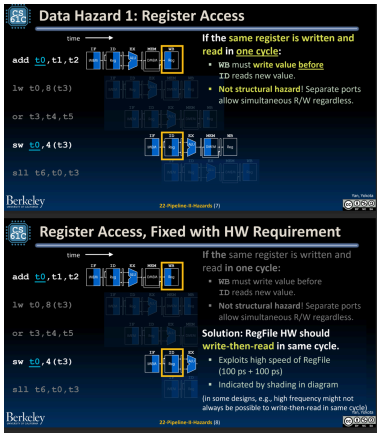
Anonymous Scorpion 7mth #987ff

Replying to Anonymous Squirrel

what do you mean? I thought they take the value of the registers before them, which are updated at the same time

Anonymous Squirrel 7mth #987aaa

Replying to Anonymous Scorpion



Im not completely sure if I understood the problem question correctly but I was assuming that the HW requirement was implemented and following this example the WB is in line with the ID

Anonymous Scorpion 7mth #987aab

Replying to Anonymous Squirrel

Ah thank you for the information! I have no idea.

Anonymous Scorpion 7mth #987aac

Replying to Anonymous Squirrel

Could you unresolve your question?

Anonymous Squirrel 7mth #987aad

Replying to Anonymous Scorpion

My question is unresolved still but antelope's is stated as resolved

Anonymous Antelope 7mth #987dc ✓ Resolved

5. `ASel`:

● 1 ○ 0 ○ X

Solution: Though we don't care about BSel, we do care about ASel because this is where we have the option of passing in rs1 to the ALU. If we want our write to contain the correct information, we must select DataA as our input here.

SP18-MT2-Q3

Why are we selecting ASel to be 1? According to the datapath 1 selects PC, not DataA.

♡ ···

J Justin Yokota STAFF 7mth #987dd

The datapath selector bits might have been swapped since this semester... We only really standardized around 2021.

♡ 1 ···

Anonymous Coyote 7mth #987da ✓ Resolved

FA19-Final-Q2:

part c): isn't this technically an invalid instruction (as we cannot load anything into the constant register x0)? Also I was wondering if all immediates in RISC-V instructions are stored in two's complement

♡ ···

A Abhinav Vedati STAFF 7mth #987aba

1. Trying to modify x0 doesn't do anything, and this property is how we implement no-ops in this class. 2. Yes, signed integer immediates are represented in two's complement in RISC-V instructions.

♡ ···

Anonymous Porcupine 7mth #987cc Unresolved

SP20-Final-Q8:

is relocation in scope? what's the reasoning behind never relocating PC-relative addressing and always relocating static data references?

Also, in one phases of CALL do we do the steps for "generating assembly code", "semantic analysis", "parsing C code", and "lexing the C code"?

♡ ···

Anonymous Porcupine 7mth #987cb ✓ Resolved

SP20-FInal-q6:

why is the type of value for w evaluate to stack address and *(v+1) not an address? Also, whats the rule again for classifying if it's stored on the stack vs heap vs static?

♡ ···

S Sonika Vuyyuru STAFF 7mth #987ace

v is a local variable defined within the function foo, and thus is stored on the stack. Thus, the type of value that v would evaluate to is a stack address. On the other hand, the value of u is an address on the heap, since we malloced space for this variable. Similarly, u+1 would still

evaluate to some heap address. However, when we try to access into this address through dereferencing, we get some data on the heap that is probably garbage and not an address.

♡ ···

Anonymous Mantis 7mth #987ca **Unresolved**

Fa20 - midterm q5c

for v1, can u explain the logic for why there are 2 nops in line 2-3 after forwarding; and similarly 3 in lines 3-4 after forwarding. How does the branch stuff work for each phase in pipelined circuit

Part C — 3 pts
Consider the 5-stage pipelining presented in lecture with the combinational-read IMEM and DMEM and the forwarding paths as drawn in the pipelined diagram in the Appendix.

Write the number of NOPs needed between each instruction, and list the hazard that causes the stall. If no hazard occurs, select "None", and list the number of NOPs as 0. Assume you can write to and read from the same address in the register file (Reg [ ]) in the same cycle. If there is a branch, assume that it is not taken, and there is no branch prediction. Consider two cases:
Case 1: Forwarding is not implemented (the diagram as seen in lecture)
Case 2: The forwarding muxes in the diagram are driven correctly by the forwarding logic. Note: If a hazard is resolved by forwarding and no other hazard is present, select "None" for the hazard

There are 6 versions.

NOTE: there are alternative answers depending on what version of the diagram you used and which clarifications you received. Please submit a regrade request detailing which diagram you used and your reasoning for your answer if you believe your answer is correct.

Solutions
Instruction Set
Instruction Pair: Case 1 Hazard, Case 1 nops, Case 2 Hazard, Case 2 nops

Version 1

```
1         addi x1, x0, 0xFF
2         ori x2, x1, 0x7FF
3         bge x1, x2, label
4         xori x2, x2, 1
```

1-2: Data, 2, None, 0
2-3: Data, 2, Data, 2
3-4: Control, 2, Control, 3

♡ ···

Anonymous Leopard 7mth #987aaf

same

♡ ···

Anonymous Mantis 7mth #987bf **Unresolved**

Su20-Final Q2

What is the reasoning behind each of the 3 choices - I think I'm unclear with how reduction and the private and critical keywords work

(b) (2.0 pt)

```
int * shifted = other + shift;
int dot_product = 0;
#pragma omp parallel for reduction(+:dot_product)
for (int i = 0; i < n; i++) {
        #pragma omp critical
    dot_product += shifted[i] * original[i];
}
result[shift] = dot_product;
```

How will this code behave?

- ■ Always Correct, slower than serial
- ☐ Always Correct, faster than serial
- ☐ Sometimes Incorrect

(c) (2.0 pt)

```
int * shifted = other + shift;
int dps[4] = {0,0,0,0};
#pragma omp parallel
{
    #pragma omp for
    for (int i = 0; i < n; i++) {
        int id = omp_get_thread_num();
        dps[id] += shifted[i] * original[i];
    }
}
result[shift] = dps[0] + dps[1] + dps[2] + dps[3];
```

How will this code behave?

- ■ Always Correct, faster than serial
- ■ Always Correct, slower than serial
- ☐ Sometimes Incorrect

The intended answer was that the code ends up being slower. Note that the due to false sharing and MOESI, elements of dps in the same cache block will have to continuously kick each other out, thus slowing down the code to slower than serial. Note that we specify the size of a cache block as 32 B.

However, this applies only when the array is block-aligned. In reality, we can expect that this array is randomly placed in a word-aligned location. Some of these will indeed be such that dps takes up two cache blocks, thus creating up to a 2x speedup. It is thus dependent on the exact location of dps, whether the parallelized version runs faster or slower than serial.

```
int * sliding_dot(int * other, int * original, int n, int k) {
    int * result = (int *) calloc(k * sizeof(int))
    // shift the array
    for (int shift = 0; shift <= k; shift++) {
        /* OPTIMIZED CODE */
    }
    return result;
}
```

(a) (2.0 pt)

```
int * shifted = other + shift;
int dot_product = 0;
#pragma omp parallel for private(dot_product)
for (int i = 0; i < n; i++) {
        #pragma omp critical
    dot_product += shifted[i] * original[i];
}
result[shift] = dot_product;
```

How will this code behave?

- ☐ Always Correct, faster than serial
- ■ Sometimes Incorrect
- ☐ Always Correct, slower than serial

♡ ···

Anonymous Mantis 7mth #987bc Unresolved

Su20-MT2 Q1: It seems like the definition of PTEs is different in both part a and b; part a uses size of physical memory but part b is using size of virtual memory; what exactly is my PTE?

1. It's All an Illusion

(a) Consider a system with 4 GiB of physical memory and 64 GiB of Virtual Memory. The page size is 4 KiB. Recall that the page table is stored in physical memory and consists of PTE's, or page table entries. Please fully simplify your answer and leave it in decimal. Fully simplify your exponents down to decimal! Please round your decimal values to two places if needed (do not include unnecessary 0's).

i. (3.0 pt) If, for each PTE, we choose to also store 12 bits of metadata (e.g. permission bits, dirty bit), how many page table entries can we now store on a page?

1024

First we need to find the side of each PTE so we need to figure out how many bits of physical memory we need to have to address. $\log_2(\frac{4GiB}{4KiB}) = \log_2(\frac{2^{32}}{2^{12}}) = 20bits$

Now we can calculate the size of each PTE: 20bits (number of physical pages)+12bits (metadata bits) = $32bits = 4Bytes$

Then we need to find the size of a page: $4KiB = 4096Bytes$

Then we divide: $4096Bytes/4Bytes = 1024$

ii. Regardless of what you got for the previous part, for the next two parts, let's now assume each page could store 2048 PTEs. The page, physical, and virtual memory sizes are unchanged.

A. (3.0 pt) How many pages does our page table occupy (aka how many valid (active) pages is our page table) if we have a single level page table which has only one valid data page?

8192

Since this is a single level page table, we need to find the total number of pages we need, since we do not have compression.

We have 2^{36} addresses of virtual memory, or $\frac{2^{36}}{2^{12}} = 2^{24}$ PTEs in our PT.

Now we divide this value by the number of PTEs/page to get the number of pages we need: $\frac{2^{24}}{2048} = 2^{13} = 8192$ pages.

B. (5.0 pt) How many pages does our page table occupy (aka how many valid (active) pages is our page table) if we have a two level page table which has only one valid data page? Each level uses an equal amount of bits of the page number.

4

Since we have a two level page table, we need the top level page table and a single next level of the table. Since each level has an equal number of bits, this means each level has 12 bits. Now we need to figure out how many pages the top level will be. So we have 2^{12} page numbers and 2^{11} entries per page, so we only need 2 pages for the top level page table. Since the next level is the same size, it would also only take 2 pages. Also since the top level page table is able to show the other pages except for the one which is valid, we do not need to create any other page tables than the upper one and the single lower page table. This means we have 2 pages for L1 and 2 pages for L2 which means we need a total of 4 pages.

♡ ···

Anonymous Hedgehog 7mth #987bb ✓ Resolved

Sp19-Final-Q4

Is this question in scope? I do not remember anything about atomic instructions in depth from lectures, like from what the question is asking for example aniswap, lr.c and sc.w for example. If we did where can I look to get well verse in this?

♡ ···

Candice Yang STAFF 7mth #987db

These instructions are out out scope.

♡ ···

Anonymous Mantis 7mth #987ba Unresolved

Fa19-Final Q8B;
why is Arr[i+1] a hit; can someone walk me through the logic of working through a problem like

this - how do we determine what is stored in the cache - how many neighboring elements are stored and so on {also what does block aligned mean}

Q8) This is for all the money! **(15 pts = 3 + 7 + 5)**

Assume we have a single-level, 1 KiB direct-mapped L1 cache with 16-byte blocks. We have 4 GiB of memory. An integer is 4 bytes. The array is block-aligned.

```
#define LEN 2048

int ARRAY[LEN];
int main() {
    for (int i = 0; i < LEN - 256; i+=256) {
       ARRAY[i] = ARRAY[i] + ARRAY[i+1] + ARRAY[i+256];
       ARRAY[i] += 10;
    }
}
```

a) Calculate the number of tag, index, and offset **bits** in the L1 cache.

T:22	I:6	O:4

SHOW YOUR WORK

Offset: log2(block size) = log2(16) = 4
Index: log2(cache size / block size) = log2(1 KiB / 16) = log2(64) = 6
Tag:
First find total address bits log2(4 GiB) = log2(4 * 2^30) = log2(2^32) = 32
Then 32 - Index - Offset = 32 - 6 - 4 = 22

b) What is the hit rate for the code above? Assume C processes expressions left-to-right.

50%

SHOW YOUR WORK

Every iteration it's
ARRAY[i] read MISS
ARRAY[i+1] read HIT
ARRAY[i+256] read CONFLICT→ MISS
ARRAY[i] write CONFLICT→ MISS
ARRAY[i] read HIT
ARRAY[i] write HIT
3 MISSES, 3 HITS. 50% hit rate.

♡ ···

Anonymous Mantis 7mth #987af ✓ Resolved

Given the following looping workload over an array where `N` is a large power of 2. The cache starts out empty, and the `process()` function doesn't introduce any significant cache pressure (so you can discount any hits or misses in the `process()` function)

```
uint32_t arr[N];

for (int j=0; j < 30; j++) {
    for (int i=0; i < N; i += 1) {
        process(arr[i]);
    }
}
```

Express the following answers as a function of `N`. **If you have a fraction, please fully simplify it.** If you believe that an answer is of the form `42 * N`, DO NOT include the multiply. You should format that answer like `42N` (Also in that exact order). Failure to do so or not capitalizing N will result is no points! If you have a fraction answer of the form `1 / 42 * N`, format it like `(1/42)N`. Note if it is NOT a fraction, you MUST not include the parentheses.

(a) Suppose we have a LRU fully associative cache of size 4N B and a block size of 4B:

i. (2.0 pt) Number of hits:

29N

ii. (2.0 pt) Number of misses:

N

iii. (2.0 pt) What type of locality is this cache taking advantage of (select all that apply)

- ☐ Quasi-balistic
- ■ Temporal
- ☐ None
- ☐ Spatial

iv. (2.0 pt) Does your answer change if the cache is 2 way set-associative? (Note: The cache size is still the same)

- ○ Yes
- ● No

(b) Suppose we have a LRU fully associative cache of size 2N B and a block size of 4B:

i. (2.0 pt) Number of hits:

0

ii. (2.0 pt) Number of misses:

30N

iii. (2.0 pt) What type of locality is this cache taking advantage of (select all that apply)

- [x] None
- [ ] Temporal
- [ ] Quasi-balistic
- [ ] Spatial

iv. (2.0 pt) Does your answer change if the cache is 2 way set-associative? (Note: The cache size is still the same)

- (●) No
- ( ) Yes

(c) Suppose we have a LRU fully associative cache of size 2N B with a block size of of 8B:

i. (2.0 pt) Number of hits:

15N

ii. (2.0 pt) Number of misses:

15N

iii. (2.0 pt) What type of locality is this cache taking advantage of (select all that apply)

- [ ] Temporal
- [ ] None
- [x] Spatial
- [ ] Quasi-balistic

iv. (2.0 pt) Does your answer change if the cache is 2 way set-associative? (Note: The cache size is still the same)

- ( ) Yes
- (●) No

what's the logic in all of these; can u walk me through the step by step reasoning

♡ ...

J Jedidiah Tsang STAFF 7mth #987bd

Sorry, can you tag your question accordingly and narrow the scope of your questions? It would be helpful to know what your confusion is.

♡ ···

F Fouad Sinno 7mth #987ae Unresolved

FA20-Final-Q1B:

Why does CODE INPUT 6 include the "~" symbol? Also, how did we get a [HEX INPUT HERE] of 12400000? I am assuming we get it by adding 0xFFFFF to CODE INPUT 7, but what's the point behind this?

Part B — 8 pts We want to write a helper function to return a memory address aligned to 2^{20}-byte boundaries. We also need to "return" the original malloced amount so we can free it later. We want to `malloc` the *fewest extra bytes possible*, and then do something to the pointer to align it. Fill in the code to complete it; don't worry about typecast warnings/errors (we removed casting for simplicity).

```
void *malloc_2totheN_aligned(size_t size, void /* CODE INPUT 1 */) {
    size_t offset = /* CODE INPUT 2 */;
    /* CODE INPUT 3 */ = malloc(size + offset); //smallest possible
    return ((/* CODE INPUT 4 */ + offset) /* CODE INPUT 5 */ /* CODE INPUT 6 */); //align
}

int main(int argc, char *argv[]) {
    void *head, *aligned;
    aligned = malloc_2totheN_aligned(59, /* CODE INPUT 7 */
    printf("head = 0x%p\n",head); // %p is what we use to print out a pointer
    printf("aligned = 0x%p\n",aligned);
    free(head);
}

unix% a.out
head    = 0x12345678
aligned = 0x[HEX INPUT HERE]
```

Solutions

CODE INPUT 1: `**head`

CODE INPUT 2: `1 << 20 - 1`

CODE INPUT 3: `*head`

CODE INPUT 4: `*head`

CODE INPUT 5: `&`

CODE INPUT 6: `~(1 << 20 - 1)`

CODE INPUT 7: `&head`

[HEX INPUT HERE]: 12400000

In order for this code to work, we need at least one contiguous group of size bits, starting at an aligned address. We don't have any control over our initial head position, so we need to add an extra few bytes to ensure our buffer contains such a block. The specified amount makes it so that exactly one such buffer is guaranteed to exist; we then shift our return value to exactly the right spot. We need to send in a double pointer so that we modify a pointer outside the program; if we had a single pointer, the change we make to head in the function would not appear outside the function, because C is pass-by-value.

♡ 1 ···

Anonymous Cobra 7mth #987ad Unresolved

Fa21-Final-Q2.2

How was the mantissa being rounded down? In the problem, it says to change the LSB to be 0, but I'm not sure how that happens here. How are you able to go from *2^-2 to * 2^-6?

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… $* 2^{-2}$. The mantissa rounds down, so our float would be 0b1.0101 $*2^{-2}$, or 0b10101 $*2^{-6}$ or $21 * 2^{-6}$

Anonymous Hedgehog 7mth #987aa ✓ Resolved

SP19-Final-Q1 Up to G

Just to make sure this is all out of scope? I do not remember learning any of this

M Myrah Shah STAFF 7mth #987ab

This question is related to endienness, which is in scope.

Anonymous Hedgehog 7mth #987ac

oops I just edited the question, I meant for spring 2019 final, sorry about that

Candice Yang STAFF 7mth #987cf

Yes, they are out of scope.

Anonymous Spider 7mth #987d Unresolved

FA17-MT2-Q4

c. Keeping the modified pipeline in mind, consider the program below with the following assumptions:

- **There are no pipelining optimizations** (no forwarding, load delay slot, branch prediction, pipeline flushing, etc.)...
- **We cannot read and write from registers in the same clock cycle.**
- **An integer 100 is stored at memory address 0x61C61C61, and that R[a0] = 0x61C61C61**.
- A hazard between two instructions should be counted as only 1 hazard.

```
lw t0, 0(a0)         # R[a0] = 0x61C61C61
srli s0, t0, 4
faddi s1, t0, 1.7
beq a0, s1, Label
add a1, t2, t3
Label ...
```

Write your answers to the questions below on your answer sheet.

```
lw t0, 0(a0)
nop
nop
nop
srli s0, t0, 4
```

```
faddi s1, t0, 1.7
nop
nop
nop
beq a0, s1, Label
nop
nop
nop
add a1, t2, t3
```

For this problem, why is 3 cops required between load word and srli?

♡ ···

Anonymous Hyena 7mth #987c ✓ Resolved

SP20 3a

Im confused why this one is SIMD. Don't we need to read from memory for the 100M matrix and the transposed matrix? That would be two memory accesses for every multiply operation and therefore should be Map/Reduce.

I understand for 3d through a similar logic why it is Map/Reduce, but do not understand why the same logic does not apply for 3a.

♡ ···

Candice Yang STAFF 7mth #987ce

To use map/reduce, you would first need to have the input as key-value pairs. For matrices, they are naturally not key-value data, and using SIMD would fit this task better. Also keep in mind that map/reduce is not in scope this semester.

♡ ···

Anonymous Hyena 7mth #987b ✓ Resolved

Sp20 Final q1

(c) (2.0 pt) a RISC-V instruction? If there's an immediate, write it in decimal. If it is an invalid instruction, write INVALID INSTRUCTION (in all caps).

```
lb x0, -96(x0)
```

Shouldn't this be considered an invalid instruction since we are trying to load the zero register? Also wouldnt trying to access -96(x0) also fail because it;s trying to access a negative memory address?

♡ ···

J Justin Yokota STAFF 7mth #987e

Keep in mind that all math in a computer is done mod 2^N. In this case, we'd access the memory address whose binary matches the 2's complement representation of the number -96.

♡ ···

Anonymous Coyote 7mth #987a ✓ Resolved

SP18-Final-Q3b
Might be a dumb question but I was a little confused by how the imm bits were determined for the instruction code. Does it count offsets from the very first line?

♡ ···

 Candice Yang STAFF 7mth #987cd

good question! On the ref card, bge adds offset to PC (PC = PC + offset). So, the offset should be counted from the current instruction (bge) to the destination label (Cont).

♡ ···