

You are viewing this thread in readonly mode.

[Final] Past Exams - 2022 #985

E **Eric Che** STAFF 1,503
7 months ago in **Exam - Final** VIEWS




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 Coyote** 7mth #985ddf Unresolved
SU22-FinalQ8

would it be alright to use vmul instead of vand, since the corresponding index if 0 would cancel the index and only multiply by one and keep the number if its greater than 0.

but you may not add more lines.

```
1 int32_t vector_mul_positive(int32_t *a, int32_t *b, int32_t N) {
2     int32_t result[4];

3     __m128i sum_v = vset(0);
4     __m128i cond_v = vset(0);

5     #pragma omp parallel for
6     for (int i = 0; i < n; i+=4) {
7         __m128i curr_v1 = vload((__m128i) a + i);
8         __m128i curr_v2 = vload((__m128i) b + i);
9         __m128i mul = vmul(curr_v1, curr_v2);
10        __m128i tmp = vcmpt(mul, cond_v);
11        temp = vmul(mul, temp);
12        #pragma omp critical
13        sum_v = vadd(sum_v, temp);
14    }

15    result = vstore(sum_v, result);

16    return result[0] + result[1] + result[2] + result[3];
17 }
```

also I thought that doing vand(number, 1) would just give us 1 or 0, 1 bit extends to 000000...1 and the bit wise operation and would just return either one or 0.

♡ ...



Anonymous Shrew 7mth #985dda

✓ Resolved

Su22-Final-Q2

Why can't we do new_file -> data.contents = contents? Why do we need to use strcpy (for the first funct)

♡ ...



Minh Nguyen STAFF 7mth #985ddc

new_file->data.contents is an array, not a pointer, and you cannot change the starting address of an array once it is defined, therefore, the address of data.contents is fixed. strcpy copies the string from the memory pointed by contents into the array new_file->data.contents. This is a character-by-character copy, so it doesn't affect the array's location in memory!

♡ 1 ...



Anonymous Oyster 7mth #985dbd

✓ Resolved

SP22-Final-Q5.5

Solution says push cause a data hazard if we write to sp immediately following a push instruction. I don't quite understand how that would cause a data hazard. I assume write to sp means sw something to sp. For example consider

```
push rs2
```

```
sw t0 0(sp)
```

From my understanding, in push, our ALU output is (original sp-4), while write to (original sp-4). Then we write t0 to (original sp-4). However, the since we are writing to DMEM without return, the only thing that might cause data hazard is addi sp sp -4. So it would cause data hazard if and only if

```
addi sp sp -4
```

```
sw t0 0(sp)
```

would cause a data hazard, which I think would not, since it's an EX to EX scenario. Am I missing something?

♡ ...



A Abhinav Vedati STAFF 7mth #985dca

We're using the standard 5-stage pipeline from the reference card. Thus, we haven't added any bypass paths to resolve data hazards more quickly. Thus, the value of the ALU will only be written back to the RegFile during the WB stage of the push instruction, whereas the sw instruction needs this value in ID stage.

♡ ...



Anonymous Mandrill 7mth #985dbc

✓ Resolved

SU22-Final-Q3.3

Would an alternate solution be jalr x0 ra -12? In line 13, we set the ra to be line 14. I subtract 12 from this value to hit line 10 which is where the Loop label is.

♡ ...



A Abhinav Vedati STAFF 7mth #985dcb

This should work. It's not very elegant, but it's technically correct.

♡ 1 ...



Anonymous Mandrill 7mth #985dbb

✓ Resolved

SU22-Final-Q8.1

Would a valid alternate solution be to leave line 11 blank, and then in line 13 set `sum_v = vadd(sum_v, vmul(mul, tmp))`? (All other lines matching the solution). My logic is that if `tmp = vcmpgt(mul, cond_v)` is a vector of 1s and 0s where 0s correspond to negative solutions, if we multiply `tmp` with `mul`, we will cancel out the negative products and get then add this cancelled out version to `sum_v`.

♡ ...

A **Abhinav Vedati** STAFF 7mth #985dcc

I don't think that it's explicitly stated in this exam, but `vcmpgt` probably returns a mask of all 1s for each positive element (32 in this case), and 0 for each negative element. This is why we use the `vand`, otherwise bitwise anding 5 with 1 will result in 1, when we want 5. Thus, this idea is almost there but it probably will not work.

♡ 1 ...

Anonymous Llama 7mth #985daf ✓ Resolved

SU22-Final-Q8

Will the bitwise and for 0b1011 and 0b1 produce 0b1011 or 0b1?

I think in this question it seems to produce 0b1011 but in **Fa23-Final-Q5** the result of bitwise and for a number and 0b1 should be 1.

♡ ...

Andrew Liu STAFF 7mth #985dde

It will produce 0b1, as $0b1 == 0b0001$. Note that in that question, `cmpgt` returns values of $0b000\dots000$ or $0b111\dots111$

♡ ...

Anonymous Shrew 7mth #985dae ✓ Resolved

SP22-Final-Q2.2

Doesn't this become Not B if we use the absorption law?

2 (2 points) Simplify the following Boolean expression. For full credit, your solution must use at most 3 boolean operators (OR, AND, NOT).

$$\neg((A + \neg A)B + (CD)) + CD$$

Solution: $\neg B + CD$

First, note that $(A + \neg A) = 1$, which reduces the expression to $\neg(B + (CD)) + CD$.
Then, using DeMorgan's law reduces the expression to $(\neg B)(\neg(CD)) + CD$.
Finally, using absorption reduces the expression to $\neg B + CD$.

An intuitive way to see the absorption step: if CD is 0, then the expression reduces to $\neg B$. If CD is 1, then the expression reduces to 1.

Grading: Partial credit was awarded for expressions that correctly matched the behavior of the given expression, but had more operators than optimal. Each extra operator lost 0.4 points, with the minimum of 0 points at 8 operators (the same number of operators as the original expression).

Absorption Law

$$x \cdot (x + y) = x$$

$$x + (x \cdot y) = x$$

Proof



YouTube

ABSORPTION LAW - Theorems in Bool...

As in if CD was X and B was Y, then wouldn't this reduce to !Y -> !B

♡ ...

M Myrah Shah STAFF 7mth #985dce

If CD was X and B was Y, the form of this would be !(Y + X) + X, which does not match the form of the law.

♡ ...

Anonymous Capybara 7mth #985dac ✓ Resolved

SP22-Final-Q6.1

for blank 1, can I write `calloc(n+1, sizeof(char))`? is this even better since we guarantee a null terminator here

♡ ...

M Myrah Shah STAFF 7mth #985dcf

No, because of the requirement of the output:

Output: An array of n 1-byte integers. The ith element of this list is 0 if i is prime, and 1 otherwise.

♡ ...

Anonymous Cheetah 7mth #985cfd ✓ Resolved

SU22-Final-Q8.1

Why is line 11 necessary? If `cond_v` is already 0, and `vcmpgt` will return the largest number in that position, won't all the negatives be zeroed out already? What is the need to and everything together.

```

1 int32_t vector_mul_positive(int32_t *a, int32_t *b, int32_t N) {
2     int32_t result[4];
3     __m128i sum_v = vset(0);
4     __m128i cond_v = vset(0);
5     #pragma omp parallel for
6     for (int i=0; i<N; i+=4) {
7         __m128i curr_v1 = vload(a+i);
8         __m128i curr_v2 = vload(b+i);
9         __m128i mul = vmul(curr_v1, curr_v2);
10        __m128i tmp = vcmpgt(mul, cond_v);
11        tmp = vand(tmp, mul);
12        #pragma omp critical
13        sum_v = vadd(sum_v, tmp);
14    }
15    vstore(result, sum_v);
16    return result[0] + result[1] + result[2] + result[3];
17 }

```

♡ ...



Oliver Ye STAFF 7mth #985cfe

vcmpgt will output a vector of 1s and 0s, where 1 corresponds to 'true' and 0 corresponds to 'false.' line 10 gives us a vector where we have some 1s and 0s corresponding to whether the result of vmul is positive or not in that position; therefore, we need to have line 11 in order to and the results of vmul and vcmpgt together in order to zero out (zero and any value gives 0) all the values returned by vmul which we do not care about

to clarify, vcmpgt does not return the largest number in some position, it returns 1 if the corresponding indexed value in mul is greater than the corresponding indexed value in cond_v (always zero).

♡ 1 ...



Anonymous Aardvark 7mth #985cfc

✓ Resolved

SU22-Final-Q7.4

how do you get the physical addresses and know the access type without having access to the TLB or page table?

Q7.4 (10.5 points) Each row in the table represents a memory access. For each row, fill out the corresponding physical address, and whether the access causes a TLB hit, a TLB miss but page table hit, or a page fault.

Virtual Address	Process ID	Physical Address	Access Type
0xDEADBEEF	1	0x000EEF	<input type="radio"/> TLB Hit <input type="radio"/> TLB miss, page table hit <input checked="" type="radio"/> Page Fault
0xDEADBEEF	2	0x001EEF	<input type="radio"/> TLB Hit <input type="radio"/> TLB miss, page table hit <input checked="" type="radio"/> Page Fault
0x0000061C	2	0x00261C	<input type="radio"/> TLB Hit <input type="radio"/> TLB miss, page table hit <input checked="" type="radio"/> Page Fault
0xDEADB61C	2	0x00161C	<input checked="" type="radio"/> TLB Hit <input type="radio"/> TLB miss, page table hit <input type="radio"/> Page Fault
0xDEADB61B	1	0x00061B	<input type="radio"/> TLB Hit <input checked="" type="radio"/> TLB miss, page table hit <input type="radio"/> Page Fault
0xDEADB61C	1	0x00061C	<input checked="" type="radio"/> TLB Hit <input type="radio"/> TLB miss, page table hit <input type="radio"/> Page Fault
0x0000061A	2	0x00261A	<input type="radio"/> TLB Hit <input checked="" type="radio"/> TLB miss, page table hit <input type="radio"/> Page Fault
0x0000061A	1	0x00361A	<input type="radio"/> TLB Hit <input type="radio"/> TLB miss, page table hit <input checked="" type="radio"/> Page Fault

♡ 2 ...

M Myrah Shah STAFF 7mth #985dcd

We have 4 GiB of virtual memory, 16 MiB of physical memory, a 4 KiB page size, and a single-level page table. We also have a 2-entry, fully-associative TLB with LRU replacement policy. **Assume that the TLB starts empty, and that physical pages are assigned in order starting with page 0 (e.g. page 0, page 1, etc.).** Also assume that we are working with a single-core system that will context-switch between processes.

From the bolded part, we know that the TLB starts out empty and we also know that it has 2 entries. It also states that physical pages are assigned in incrementing order starting from 0. I would recommend tracking what is currently in the TLB at each access.

♡ ...

Anonymous Llama 7mth #985cee ✓ Resolved

SP22-Final-Q3.8

I am a bit confused about what's the answer to this question.

♡ ...

E Erik Yang STAFF 7mth #985cfb

One possible answer is that B is considered incorrect because of calling convention (not restoring s0 back on stack)

♡ ...

 **Anonymous Llama** 7mth #985daa

Why don't we need to restore t0 back on stack?


♡ ...

E Erik Yang STAFF 7mth #985dba

↩ Replying to Anonymous Llama

t0 is a caller saved register

♡ ...

 **Anonymous Mallard** 7mth #985cea ✓ Resolved

SP22-Final-Q4.4

For the answer choice "1", why can the IEEE-754 floating point representation represent 1? If all the mantissa bits are zero, isn't this equal to NaN or inf?

♡ ...

 **E Erik Yang** STAFF 7mth #985cef

That's only if the exponent is all ones, paired with a zero mantissa would represent INF, and a nonzero mantissa would represent NAN

♡ ...

 **T Tyler Yang** STAFF 7mth #985cfa

I think the answer explanation was a little misleading. But if we are assuming standard exponent bias (For single precision the bias is -127), then 1 can be represented by all 0s for the mantissa but exponent is 127 (01111111) This gives the desired answer of 1.

♡ ...

 **Anonymous Crow** 7mth #985cdf ✓ Resolved

sp22-Final-Q8.3, 8.4

I am kind of confused by these two questions, what exactly is happening inside the for loop?


♡ ...

 **O Oliver Ye** STAFF 7mth #985cff

8.3: try to draw two 3x3 matrices and multiply them together. how many vector dot products do you need? since each iteration of the forloop attempts to compute a full matmul on whatever the registers have in them (for 8.3 its a 3x3 matrix in each register), each iteration of the forloop needs to load in as many vectors as are needed for vector dot products -- in this case, we need 3 rows from vector A and then 3 columns from vector B, so 6.

8.4: from 8.2 and 8.3, we see that on a $k \times k$ kernel, we need to have k^2 registers (from 8.2) outside the for loop and $2k$ registers (from 8.3) to support the kernel size. the maximum kernel size would be the value of k which gives us a sum less than 64 (because the 'miss rate' is effectively minimized). we can see that a value of $k = 7$ gives us a total registers required of 63, which is our fastest and most efficient speedup. if we chose something like $k = 10$, then we'd need $100 + 20 = 120$ registers, but we only have access to 64 registers, so we'd lose some time moving values in and out of registers

♡ 1 ...

 **Anonymous Crow** 7mth #985cde ✓ Resolved

SP22-Final-Q6.4 and 6.5

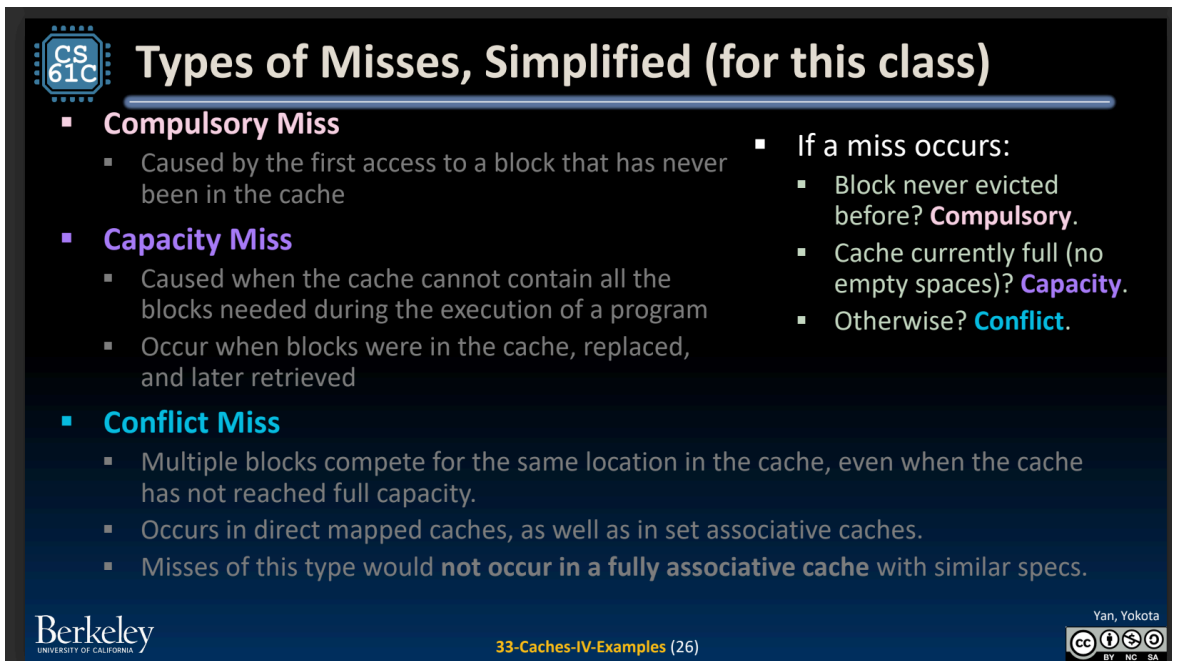
6,4: When does a direct mapped cache have a conflict miss versus a capacity miss? Also I'm still a little lost at how they did the prime math to figure out exactly how many capacity misses there are —I understand we can only store the first 1/4 of the array but how do we get the number $(\pi(128) - 1) * 128$?

6.5: Why can we say that all prime numbers are guaranteed to not be set to 1 by any loop (even composite ones), and as such any composite number will be set to 1 by some iteration of the inner for loop

♡ 1 ...

 **Andrew Liu** STAFF 7mth #985ddd

A direct mapped cache has a conflict miss when the cache is not full, and a capacity miss when the cache is full (Lecture 33, slide 26):




CS 61C **Types of Misses, Simplified (for this class)**

- **Compulsory Miss**
 - Caused by the first access to a block that has never been in the cache
- **Capacity Miss**
 - Caused when the cache cannot contain all the blocks needed during the execution of a program
 - Occur when blocks were in the cache, replaced, and later retrieved
- **Conflict Miss**
 - Multiple blocks compete for the same location in the cache, even when the cache has not reached full capacity.
 - Occurs in direct mapped caches, as well as in set associative caches.
 - Misses of this type would **not occur in a fully associative cache** with similar specs.

If a miss occurs:

- Block never evicted before? **Compulsory.**
- Cache currently full (no empty spaces)? **Capacity.**
- Otherwise? **Conflict.**

Berkeley UNIVERSITY OF CALIFORNIA 33-Caches-IV-Examples (26) Yan, Yokota 

The prime math comes from the fact that we check all the primes up to $\sqrt{n} = \sqrt{16384} = 128$ in the sieve. When a prime is found, notice that we have to load in 128 blocks of memory — all of them needed to store the array basically. When we did this for the smallest prime, 2, all of them were compulsory misses since it was the first time the array was loaded, and the rest are capacity misses, leading to the $\pi(128) - 1$ factor.

For 6.5, The wording means that like if a thread runs for $N = 6$, then it will set only 6, 12, 18, ... to be composite, all of which are clearly composite, since they have a number that divides them (6)

♡ ...

 **Anonymous Wolf** 7mth #985cdc ✓ Resolved

SU22-Final-Q5

Is this question on hamming error correcting code in scope?

♡ ...



Jero Wang STAFF 7mth #985cdd

Hamming ECC is not, but constructing/reading/understanding circuit diagrams is.

♡ 1 ...



Anonymous Badger 7mth #985cce

✓ Resolved

SU22-Final-Q7.4

What is a "2-entry", fully associative TLB. I know what a fully associative TLB is, but not sure what "2-entry" is supposed to mean.

♡ ...



Jero Wang STAFF 7mth #985cda

It's a TLB that has 2 entries in it (like space for two sets of PTEs/tag/metadata bits)

♡ ...



Anonymous Magpie 7mth #985bdc

✓ Resolved

Sp22-Final-Q8.1,

why don't we make i go up to $n/4*4$ and instead make it less than N ? When do we know whether or not to make it go up to $N/4*4$ or just N ?

And for 8.4, if we have 64 vector registers, why wouldn't the answer be 8? (can't we just reuse some of the vector registers for when we need the 16 in the inner for loop for loading?)

♡ ...



Anonymous Louse 7mth #985bdd **ENDORSED**

It says in the problem that N is a multiple of 4, so we don't need to worry about any tail cases. But in general, you're right.

For 8.4, I think we're answering the question in regards to the specific implementation in the problem, not in general; I assume there are probably other ways to implement `matmul_kernel` but 8.4 is asking about what happens if we run the code given in the problem. Not sure if you would be able to reuse any vector registers here anyways tho, I guess if you're interested to talk about it how would you do it?

♡ ...



Jero Wang STAFF 7mth #985cbc

Adding on to Anon Louse, for 8.4, the bottleneck is the output registers. Side length of 7 means that the result matrix is 7×7 , which needs 49 output registers. If it were 8, it would need 64 registers just for the results, leaving no registers left for actual computations.

♡ ...



Anonymous Koala 7mth #985bce

✓ Resolved

Su22-Final-3.7

Why is `memRW` set to Read for `jal`. I thought it would be Doesn't matter since there is no data memory interaction involved in the DMEM for `jal`.

s1t

sri

dse1

Other

Q3.7 (0.5 point) MemRW

Read

Write

Doesn't matter

♡ ...



Jero Wang STAFF 7mth #985cbb

There's no 3.7 for SU22; this looks like SU23, can you post it in that thread instead?

♡ ...



Anonymous Wolf 7mth #985bcd

✓ Resolved

SP22-Final_Q1.5

Is this question in scope? If so, what lecture does this correspond to?

Q1.5 (1 point) TRUE or FALSE: Even with a direct memory access (DMA) controller, the CPU is required to initiate a memory transfer.

TRUE

FALSE

Solution: True. Even though the CPU is not directly performing the memory access, as the DMA controller does that, the CPU is still needed to initiate that action, since it's still a form of I/O, and the processor at the very least, always starts the process to handle them.

Grading: All-or-nothing.

♡ ...



Jero Wang STAFF 7mth #985cba

No, DMA is out of scope.

♡ 1 ...



Anonymous Goat 7mth #985bcc

✓ Resolved

Su22 Q2, why is `calloc` used instead of `malloc`? For `create_file`, how come there is no place for `is_folder` to be false? For `create_folder`, how come we don't initialize the array?

♡ ...



Anonymous Louse 7mth #985bcf **ENDORSED**

`calloc` is used instead of `malloc` because `calloc` automatically sets all bits at its memory address to 0. In C, I believe `false` is equivalent to 0 while `true` is equivalent to 1 (see the edit to the top answer here: <https://stackoverflow.com/questions/5369770/bool-to-int-conversion>), so when we call `calloc`, `is_folder` is automatically set to `false` meaning we don't need to set it to `false` manually.

The same reasoning holds for `create_folder`, we don't need to manually initialize the array because of how `calloc` works. This is also explained at the end of the solution to this problem:

To set `new_dir->data.children` to `NULL` correctly, we can either `calloc` the entire struct, since all 0 bits is equivalent logically to `NULL`. Alternatively, we can set the fields manually if we used `malloc` to allocate the new folder as follows:

```
15     ...
16     for (int i = 0; i < 16; i++) {
17         new_dir->data.childreni = NULL;
18     }
19     ...
```

♡ ...



Anonymous Magpie 7mth #985bcb

✓ Resolved

Sp22-Final-Q6:

is the reason we can't do `malloc(sizeof(char))` for 6.1 due to out line on line 7 where we check `primes[i]` and if we did `malloc`, then this would error?

for 6.3, the explanation notes that when we access `primes[0]` on line 4, this miss brings in a 128 byte block but how did we know that it would bring in a 128 byte block or is this just in general? Also, do we not have to take into account the hit or misses brought in from `primes[1]` on line 5, `primes[2]` on line 7 and `primes[4]`? the explanation just started mentioning these misses starting from `primes[6]` so I was confused why we didn't note the ones before

♡ ...



Jero Wang STAFF 7mth #985cae

6.1: We need the array to be initialized to 0's, so `malloc` would not have the same behavior. If we did `malloc`, line 7 would behave unexpectedly.

6.3: The directions before Q6.2 states that our cache has 128B blocks. Since `primes[1]` is in the same block as `primes[0]`, and that value is in the cache already so it's not a miss, and the same goes for everything up until 127. `primes[4]` should have the same result as `primes[6]`. I think I edited the solutions weirdly when trying to fix another issue, but we've noted this on our end, sorry!

♡ ...



Anonymous Mouse 7mth #985cbf

I'm misunderstanding something here. If the block is 128 bytes, why does it go from `primes[0]` to `primes[127]`, if each `primes[i]` stores an integer that's 4 bytes? Wouldn't this store `primes[0]` to `primes[31]`? Also: when this problem is solved in the video + the solutions pdf, nothing about tag/index is mentioned. How do we know that we don't have to worry about them in this problem? Is there a way to know that our tags/indices will match up and result in a hit?

♡ ...



Jero Wang STAFF 7mth #985ccc

↩ Replying to Anonymous Mouse

These are chars not ints so they're one byte each.

I don't have the exam on hand (on mobile) but if I recall correctly this specific function call ensures that the entire array fits in the cache, so we don't have to worry about it.

♡ 1 ...



Anonymous Shrew 7mth #985dbe

So just to confirm, basically primes[0] is a miss, bringing in a 128-byte block, and since we have a 2^{12} byte cache, and 128 is 2^5 , then that's why we have $2^{12}/2^5 = 32$ blocks and thus 32 misses?

♡ ...



Anonymous Magpie 7mth #985bca

✓ Resolved

Sp22-Final-Q5.3:

1. For the "add a new instruction format" choice, the explanation says that we are only encoding rs2 in the instruction but why aren't we doing that for sp as well? How do we know if we encode that register or not?
2. For the "add a second new imm type for immgen" and "add a new output to regfile", could someone provide a concrete example of when we would actually choose these?
3. For the "add a 2nd writeback input to regfile" choice, how do we know exactly if we need to write back to a reg or not?

Also for part 5.4 following directly afterwards, why did we "add a mux before the DMEM address input, including relevant inputs" but not check that for 5.3? And wouldn't we actually want sp+4 in this case aka the ALU output? why do we instead want just the sp?

Also, what's the reason for having to write to both sp and rs2 for pop but not for push? sorry for the many questions and ty!

♡ ...



Jero Wang STAFF 7mth #985cad

1. #985bfc, generally we try to reduce the amount of changes as much as possible, so encoding here would help.
2. If you need to generate an immediate where say bits 30-31 of the instruction is the immediate, this would require a new immediate type. If you need to read three registers (e.g. needing rs1, rs2, and rs3), you would need a new output to regfile.
3. If you need to update a value in the regfile, you need to write back to the regfile. Since we're updating two values here (sp and rd), we would need 2 write inputs to the regfile.

For 5.3, we are writing to the address $sp-4$ and are storing $sp-4$ to sp , so the ALU output can be used as the address for DMEM. For 5.4, we are reading from the address sp and are storing $sp+4$ to the sp register, so we can't use the ALU output here for both, so we need a MUX to select the correct input. $sp+4$ is the new sp (bottom of the stack), but the value we're reading is the current bottom of the stack, at sp .

You need to write to both because the instruction dictates that you're reading a value from memory into a register, and also updating sp accordingly, whereas for push you're only updating sp (and writing to memory instead of reading).

♡ ...



Anonymous Magpie 7mth #985dab

thanks! had a couple follow up questions!

1. I'm still a little confused about what it means to know if a reg like rd is encoded in the instruction or not. why isn't sp encoded as well?
2. so pretty much if a new instruction needs an immediate that isn't given by any other type of instruction, we should generate a second new imm type for immgen?

3. So if we are writing to memory (like in a store instruction), we will never have to pass in a writeback input to Regfile so we aren't technically updating a register in this store instruction? And that's the reason why we aren't actually writing back to rs2 in "push"? But conversely in our load instruction in "pop", since we are reading a value, we must pass that into Regfile? I was confused as to how we know for different instructions like store, load, jump, arithmetic instructions whether or not we write back to a register..

For the ALU output, so we have to imagine that the ALU will only output sp-4 due to us doing `addi sp sp -4` in the previous push instruction so even though we want sp+4 in pop we can't get that unless we include a mux before DMEM?

And, I'm not sure why the answer also mentions that the pop instruction requires writing to sp and rs2 if we never even use rs2 in our pop instruction? Thank you!!

The existing Regfile has one writeback input for writing to one register. The pop instruction requires writing to two registers, sp and rs2, so a second writeback input is needed.



Anonymous Hyena 7mth #985bbd

✓ Resolved

Sp22 Q5.4

Why would we need to change the data path for pop but not push? I don't understand.



Anonymous Coyote 7mth #985bbf

✪ ENDORSED

push has everything needed in the data path, since it only has 2 registers that it needs, only writing back to one (sp), only needs to pass sp - 4 to both the DMEM and the writeback, etc.

However for pop, you need more things, primarily because there is now 2 things that you need to write back, the sp register, and the rs2 register. This would mean that we would want to have a second writeback input to RegFile, and also a second RegWEn. We also have the problem where sp+4 from the ALU does not make sense to pass it to the DMEM, since it would load in the wrong word, thus we want to add a mux before the DMEM so we are able to choose sp for the DMEM.

If my explanation is not that good and is confusing I would recommend watching the walkthrough it helped me understand it

<https://www.youtube.com/watch?v=3p8zQhChXsM&list=PLnvUoC1Ghb7xURp0arzx8Q3jzdGE4n1Dn&index=2&pp=iAQBsAQB>



Anonymous Coyote 7mth #985bbc

✓ Resolved

SP22-Final-Q4.1

Would a valid answer for this be that the numbers represented in this form would always have to start with 1 compared to the original system? So this means that the number 2000 cannot be represented since 1 is always at the start? I did not see this in the solution.



Jero Wang STAFF 7mth #985cac

Yeah, that sounds equivalent to the given solution.





Anonymous Crab 7mth #985bbb

✓ Resolved

Is there a specific reason the TLB always becomes empty when we context switch while the page table entries are preserved?

♡ ...



Nithila Poongovan STAFF 7mth #985bde

Yes, the TLB is cleared when we context switch because it holds recently accessed virtual-to-physical address translations specific to the currently running process, with each process operating in its own virtual address space. When context switching (switching from one process to another) happens, the newly scheduled process will have a different virtual space. As a result, using the old process's TLB entries would lead to incorrect address translations, where the new process might access the physical memory location for the old process.

Page table entries are preserved since they store the mapping of virtual addresses to physical addresses for each process, including permissions and flags. If page table entries were cleared with each context switch, the system would have to regenerate the map every time a process is returned to, making it inefficient and consuming lots of power.

♡ ...



Anonymous Crab 7mth #985bba

✓ Resolved

Su 22, Q3.3, can I do `AUIPC t0 0. jalr x0 t0 -32`. This way, I save the pc register successfully into a register and then since the offset is 8 instructions, I can go up 32 bytes

Also for 3.4, aren't we changing the add to sub during runtime, so how would we ever alternate between instructions? we would have executed add by then right?

♡ 1 ...



Jero Wang STAFF 7mth #985cab

I think it should be -28, since the address is relative to line 18 (auipc) and not the line after.

By xoring, we change the bit to 1 if it is currently 0, and vice versa, so each iteration of the loop would change sub->add or add->sub for the next iteration. In the first iteration, it executes add, then it changes it to a sub for the next iteration.

♡ ...



Anonymous Coyote 7mth #985baf

✓ Resolved

Sp22-Final-Q1.5

Is this in scope, does direct memory access mean direct mapped cache? Did we learn about the direct mapped access controller?

Also is SP22-Final_Q1.6 out of scope as well, do not remember really learning about this

♡ ...



Jero Wang STAFF 7mth #985bff 👁 Visible to staff only

checking w instructors


♡ ...





Jero Wang STAFF 7mth #985caa

1.6 is out of scope (lecture 38) and we're confirming whether DMA is in scope. DMA is related to I/O, not direct mapped caches.


♡ ...


 **Jero Wang** STAFF 7mth #985caf
1.5 (and DMA) is out of scope.
♡ 1 ...

 **Anonymous Coyote** 7mth #985ccf
Thank you!
♡ ...


 **Anonymous Louse** 7mth #985bae ✓ Resolved
SU22-Final


On the first page it says "You have 110 minutes", this is a typo right?
♡ ...


 **Jero Wang** STAFF 7mth #985bfe
Oops yeah, it was 170 minutes.
♡ ...

 **Anonymous Magpie** 7mth #985bad ✓ Resolved
Sp22-Final-Q3.5: is polling and interrupts in scope?

Also, is the answer to 3.7 A is incorrect or neither is incorrect bc the solution provides advantages for both. If A is incorrect, what makes it incorrect?
♡ ...

 **Anonymous Beaver** 7mth #985bda
#939aad
♡ ...

 **Jero Wang** STAFF 7mth #985bfd
If I recall correctly, we graded this question based on the justification, and there was no explicit correct/incorrect answer for the multiple choice part.
♡ ...

 **Anonymous Louse** 7mth #985adf ✓ Resolved
SP22-Final-Q5.4

Solution: We can reuse one of the existing instruction format types, as the `pop` instruction only needs `rd` encoded in the instruction. For example, we could make `pop` an I-type instruction, give it a new opcode (not in use by any other instruction), and leave the `funct3`, `immediate`, and `rs1` fields unused (e.g. fill them with all zeros, or hard-code them to a value consistent with how we're using the rest of the datapath).

`ImmGen` will be used to output 4 here (so we can add 4 to `sp`), so a second new immediate type is not needed.

The existing `Regfile` has two outputs for reading two registers. The `pop` instruction only requires reading the value in `sp`, so a third `Regfile` output is not needed.

The existing `Regfile` has one writeback input for writing to one register. The `pop` instruction requires writing to two registers, `sp` and `rs2`, so a second writeback input is needed.

The `ALU` will be used to add 4 to `sp`. We can use the first `Regfile` read output (`rs1`) to read the value in `sp`, so `AMux` selects the register output, not the `PC`. `BMux` selects the `immediate` (which is always outputting 4), not the `rs2` `Regfile` read output. Neither `AMux` nor `BMux` needs a new input, and the `ALU` does not need a third register input or a new operation (add already exists).

In the existing datapath, the `DMEM` address input is the `ALU` output. In the `pop` instruction, the `ALU` output is `sp+4`, but the address we want to load from is `sp`, so an additional `DMEM` address input is needed.

In the existing datapath, `WBMux` supports writing back the `ALU` output to `Regfile`. In the `pop` instruction, the `ALU` output is `sp+4`, which is what we want to write back to the `sp` register, so no new input to `WBMux` is needed.

I'm confused why the `ALU` output here is necessarily `sp+4`, when we do `lw rd 0(sp)` don't we need to technically add 0 to the `sp` pointer first?

♡ ...



Anonymous Deer 7mth #985aee

we need to calculate `sp + 4` in order to write back this `alu` output to update `sp` in `regwritedata`

♡ ...



Anonymous Louse 7mth #985aef

Yes I understand that, but why don't we also need to compute `0+sp` for the `lw` operation?

♡ ...



Anonymous Deer 7mth #985afa **ENDORSED**

↩ Replying to Anonymous Louse

because when doing the `pop` operation, it's always going to load from `0(sp)` so we can just use `sp` directly as the input address to `DMEM`. but since the options for input addresses to `DMEM` only have `ALU` output, we need to add a mux before the `DMEM` to select the `sp` instead of `ALU` output. this requires fewer changes than if we had to add a separate `ALU` for `0 + sp` as well as a mux before `DMEM` to choose which of the `ALUs` we want for the address

♡ 1 ...



Anonymous Louse 7mth #985aff

↩ Replying to Anonymous Deer

ah ok that makes sense, thanks! Also in the first paragraph when it says "**...leave the `funct3`, `immediate`, and `rs1` fields unused (e.g. fill them with all zeros, or hard-code them to a value consistent with how we're using the rest of the datapath, for `rs1`)**", isn't our only option to hard code the other fields? Like don't we need to hard code `sp` somewhere into the instruction, how can we fill them with all zeros?

♡ ...



Jero Wang STAFF 7mth #985bfc

↩ Replying to Anonymous Louse

Yes, for `pop`, `rs1` should be hard coded to `sp` and `imm` should be hard coded to `4`, we've noted that on our end.

I think the design of the question is that you connect DMEM directly to the second write port and use the first WBMux to select register (or vice versa), so you don't need a new WBMux.

♡ ...

 **Anonymous Louse** 7mth #985bac

↩ Replying to Anonymous Deer

Also since we write to both `sp` and `rd` registers, we have to use the WBMux twice, right? Once for selecting the output of DMEM to store into `rd`, and once for selecting the ALU output to store into `sp`. So I was just wondering, how are we able to use the WBMux twice within the same clock cycle? This would require changing the value of `WBSel` while the instruction is executing, so do we assume this is something we can implement in the control logic?


♡ 1 ...

 **Anonymous Mallard** 7mth #985ceb

↩ Replying to Anonymous Louse

I have a similar question. If the mux before DMEM chooses `sp`, do we hardwire the ALU output `sp + 4` to something else? I am confused as to where we write back the value `sp + 4`.

♡ ...

 **Anonymous Alpaca** 7mth #985ade ✓ Resolved

Summer 2022: Question 6: Can you say the rule is that if it ends in a 0 and the first bit is not a 1 it is accepted?

♡ ...

 **Jero Wang** STAFF 7mth #985bfb

No, since there must be at most one non-zero bit in between the first and last bits. I think your rule would be valid for `011110` when it should be invalid.

♡ ...

 **Anonymous Hummingbird** 7mth #985adb ✓ Resolved

sp22 Q2.5 took quite a bit of arithmetic to simplify the fraction — is arithmetic like this expected to be on the final?

♡ ...

 **Anonymous Louse** 7mth #985aea

What arithmetic did you do? It shouldn't be that bad if you keep things in terms of hours: The hive machine was down for a total of $6 * 12 = 72$ hours, meaning it wasn't available for 72 out of the total of $100 * 24$ hours, and then the fraction of time it wasn't available simplifies nicely into $\frac{3}{100}$. Then the fraction of the time it was available is just 1 minus this.

♡ 1 ...

 **Catherine Van Keuren** STAFF 7mth #985baa

Something to note is that this question is also out of scope since we didn't cover availability this semester

♡ ...



Anonymous Hyena 7mth #985ace

✓ Resolved

Sp22 Q2.1

How do we know that x28 is 0b11100 in unsigned binary? I thought it would be 0010 1000. Does it have to do something with rs1 and 2 only being 5 places? Sorry, I kinda forgot how to do this after the first midterm.

Q2.1 (2 points) Translate the following RISC-V instruction into its corresponding 32-bit hexadecimal value.

`sb t3, 31(t3)`

Solution: 0x01CE0FA3

From the reference card: `sb rs2 imm(rs1)` is an S-type instruction, its opcode is 0b0100011, and its funct3 is 000.

In addition to the opcode and funct3, an S-type instruction needs a 12-bit immediate, rs1, and rs2.

The 12-bit immediate is 31, which is 0b0000 0001 1111 in two's complement binary. Bits 11-5 are 0b0000000, and bits 4-0 are 0b11111.

rs1 and rs2 are both t3, which is register x28. 28 is 0b11100 in unsigned binary.



Justin Yokota STAFF 7mth #985acf

Register numbers are in decimal, not in hex; register x28 corresponds to decimal value 28 = 0x1C, not hex value 0x28 = 40.





Anonymous Crane 7mth #985acd

✓ Resolved

sp22 final-q5.3 , for the push function, why don't we need to set $sp=sp+4$: just like we do $sp=sp-4$, for q5.4 for the pop,

Q5.3 (3 points) What further changes would we need to make to our datapath in order for us to implement the **push** instruction with as few changes as possible? Select all that apply:

- Add a new instruction format
- Add a second new immediate type for the ImmGen
- Add a new output to Regfile for a third register value
- Add a second writeback input to Regfile, along with a second RegWEn
- Add a new input to AMux, with the relevant selectors
- Add a new input to BMux, with the relevant selectors
- Add a new ALU input for a third register input
- Add a new ALU operation, with a corresponding ALUSel
- Add a mux before the DMEM address input, including relevant inputs and selectors
- Add a new input to WBMux, with the relevant selectors
- No further changes are needed to the datapath, beyond adding additional control logic

♡ ...



Anonymous Louse 7mth #985aeb

You do need to do that, see explanation in the solutions

♡ ...



Catherine Van Keuren STAFF 7mth #985afe

Good question! We are actually setting $sp = sp + 4$ in pop like we did $sp = sp - 4$ for push, but the only difference is that we don't need to write back two values. For pop, we had to write back to both the sp and rd , thus need to modify our write back logic to support that, however for push since we only write back to sp , we can keep that logic the same. Hopefully this answers your questions and feel free to reply with follow ups!

♡ ...



Anonymous Goldfish 7mth #985acc

✓ Resolved

SP22-Final-Q3.4 and 3.5 are they in scope?

♡ ...



Anonymous Louse 7mth #985aec

#985ce

♡ ...



Anonymous Trout 7mth #985abc

✓ Resolved

SP22-Final-Q3.7

Is this in scope? Where did we learn the syntax of #define and the question marks?

♡ 1 ...



Justin Yokota STAFF 7mth #985aca

Yes, this is in scope. Define statements were discussed in our initial discussion of C, as were ternary operators. Note that this specific question was a follow-up of a question on SP22-Midterm, so it may be useful to review that midterm for more context.

♡ 1 ...



Anonymous Hummingbird 7mth #985abb

✓ Resolved

SP22-Final-Q1.4

"True or False: System calls (such as the `syscall` instruction in RISC-V) are executed by the program itself."

Is this question in-scope? If so, do you know where I can go to review it?

♡ ...



Catherine Van Keuren STAFF 7mth #985abd

Yep it's in scope - System calls were discussed in the OS lecture (lecture 24)!

♡ ...



Anonymous Louse 7mth #985aba

✓ Resolved

SP22-Final-Q3

Question 3.10

Advantages of B (Fully correct)

- Runs in general faster than A. This is because A's use of parallelism ends up interleaving the writes. The result isn't a data race (since each thread gets to run on its own values), but instead causes coherence misses on the L1 cache. Due to repeated data invalidations, memory accesses in A will tend to be around the same as L3 cache access times or longer; this is often longer than an 8x speedup.

I don't understand this, what is meant by "interleaving the writes" and "coherence misses"?

♡ ...



Catherine Van Keuren STAFF 7mth #985bab

I think this question would be considered out of scope since we didn't cover cache coherency this semester, but to give you a small explanation. The way the code is written we have each thread taking specific indices of the array. For example, thread 2 takes indices 2, 10, 18, etc. aka the threads are interleaved throughout the array. Say a cache line can fit 8 data entries. Thread 2 wants to modify `data[2]` at the same time as thread 3 wants to modify `data[3]` and they both bring the line into their caches. Thread 2 does its write, but now its cache line is different from Thread 3s cache line, so that line must get written back to memory and Thread 3 must re-bring it into its cache. You can imagine how this might slow down execution quite a bit! A way to fix this code to make these issues go away would be instead of having thread 0 take indices 0, 8, 16, etc. you have it take the first $(\text{num elements})/(\text{num threads})$ indices of the array. Therefore, each thread isn't simultaneously accessing the same chunks of memory. Hopefully this makes sense and take 152 if you want to learn more about cache coherency :)

♡ ...



Anonymous Loris 7mth #985ced

Another question on Cache coherency but in 2023 exams is when the cache line has the same index like 0x0001 and 0x0002 it would cause thrashing. Is this also out of scope?

♡ 2 ...



Anonymous Marten 7mth #985aac

✓ Resolved

Spring 2022 [Q 5.3]

Isn't the entire idea with not having a new instruction type to use one of the existing opcodes? If we generate a new opcode, doesn't this just get treated as creating a new instruction type. I was thinking more of you keep the opcode the same as one of the current instruction types but then change func3 or something else to differentiate the specific instruction as push?

Some more clarity on what actually defines a "new instruction format" would be helpful.

Solution:

We can reuse one of the existing instruction format types, as the **push** instruction only needs **rs2** encoded in the instruction. For example, we could make **push** an S-type instruction, give it a new opcode (not in use by any other instruction), and leave the **func3**, **immediate**, and **rs1** fields unused (e.g. fill them with all zeros, or hard-code them to a value consistent with how we're using the rest of the datapath, for **rs1**).

ImmGen will be used to output 4 here (so we can add 4 to **sp**), so a second new immediate type is not needed.

Q5.3 (3 points) What further changes would we need to make to our datapath in order for us to implement the **push** instruction with as few changes as possible? Select all that apply:

- Add a new instruction format
- Add a second new immediate type for the **ImmGen**
- Add a new output to **Regfile** for a third register value
- Add a second writeback input to **Regfile**, along with a second **RegWEn**
- Add a new input to **AMux**, with the relevant selectors
- Add a new input to **BMux**, with the relevant selectors
- Add a new ALU input for a third register input
- Add a new ALU operation, with a corresponding **ALUSel**
- Add a mux before the **DMEM** address input, including relevant inputs and selectors
- Add a new input to **WBMux**, with the relevant selectors
- No further changes are needed to the datapath, beyond adding additional control logic

♡ ...



Catherine Van Keuren STAFF 7mth #985abe

What we mean by new instruction format doesn't mean changing the opcode/func3, but actually changing which bits in the instruction equate to which part. For example, for R type instructions bits 7 - 11 represent **RS1**, whereas in S type instructions, those same bits represent part of the immediate. Creating a new instruction format would be dividing up the

bits to represent different fields (rs1, rs2, rd, func3, and more) in a way that isn't currently supported by our current instruction formats. This could happen if for example we wanted to add a third argument register to our data path.

♡ 1 ...

 **Anonymous Marten** 7mth #985add

Got it, but doesn't the opcode determine the type of instruction? So we'd keep the same opcode as instruction type S.

♡ ...

 **Anonymous Marten** 7mth #985aab ✓ Resolved

Spring 2022

I was fairly confused on how or why this is universally true? Isn't for example: $1.010 * 2^3$ an even number?

Q4.4 (2 points) Which of the following numbers can be represented exactly by our decimal floating point standard, but not by the IEEE-754 standard single precision (32 bit) floating point standard? Select all that apply.

0.1

0.8

$2^{32} = 4,294,967,296$

$\frac{1}{3} = 0.333333\dots$

1

10^{90}

0.5

10^6

None of these


Solution: The key realization to solving some of these answer choices is to note that every IEEE-754 floating-point number can be expressed as an odd integer multiplied or divided by a power of 2. (The odd integer comes from the significand, and the power of 2 comes from the exponent). If a number cannot be represented this way, then it must not be representable as an IEEE-754 floating-point number.

♡ ...

 **Justin Yokota** STAFF 7mth #985acb

$1.010 * 2^3$ is the same as $5 * 2^1$, which is the unique representation of the number as the product of an odd number and a power of 2. Conversely, 0.1 has no representation as the product of a finite odd number and a power of 2 (in number theory terms, it has an infinite 2-adic representation)

♡ ...

 **Anonymous Crab** 7mth #985ff ✓ Resolved

Sp22 Final Q1.1: If only one bit is set to 0, doesn't this mean we just allow for even offset instructions and not 16 bit offset. I am confused about the relationship between when we have RISC5 16 bit addresses and the implicit 0. Thank you!

♡ 1 ...

 **Jero Wang** STAFF 7mth #985bfa

Yes, one bit as 0 means we only allow even offsets (which are in bytes). Even offsets allows us to address any instruction as long as it is more than 2 bytes (or 16 bits), so we can address a RISC-V compact instructions.

♡ ...



Anonymous Crab 7mth #985cbe

Sorry but how does setting the last bit to 0 directly correlate with 16 BYTE offset? I understood the even part.



Rohan Mathur STAFF 7mth #985ccb

← Replying to Anonymous Crab

I'm not sure where you're getting 16 **byte** offset from. The offset is specifically 16 **bits** because an even number of **bytes** means a multiple of 2 and 2 **bytes** is 16 **bits**.



Adrian Bao 7mth #985fe

✓ Resolved

SP22-Final-Q8.1

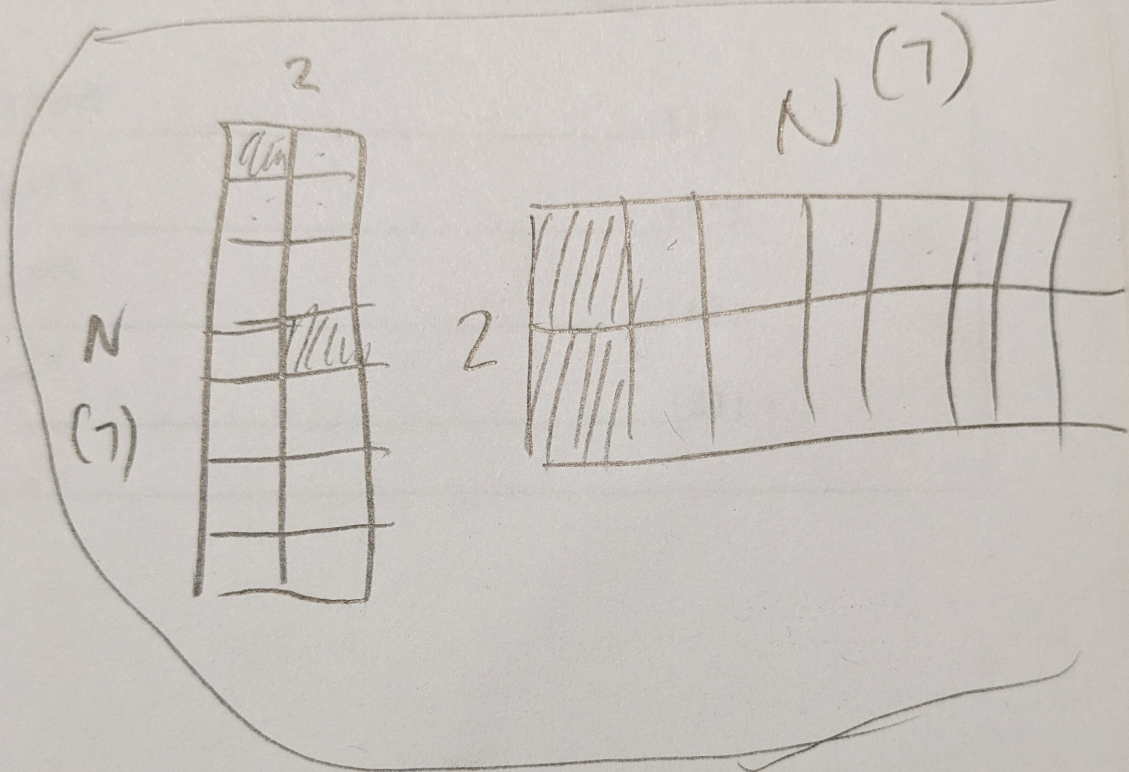
I don't understand how the registers are being assigned so that the kernel is being assigned.

```
vector* m1r0 = vec_load(mat1 + i);
vector* m1r1 = vec_load(mat1 + i + N);
vector* m2c0 = vec_load(mat2_T + i);
vector* m2c1 = vec_load(mat2_T + i + N);
```

Because m1r1 and m2c1 are the same as m1r0 and m2c0, respectively, except for adding i + N in the index, the first iteration would be like what I drew below.

I am confused how the multiplication with the kernel would then work since the left matrix seems wrong. I added 0 (i) + 7 (N) because N = 7 in my example. Thanks!

load 4 doubles at the given memory address A into a vector
 to a vector
 vector* B, vector* C): Performs a element-wise fused
 in a single operation
 all elements of the vector together: result = A[0] +



Anonymous Louse 7mth #985aad **ENDORSED**

It's given in the problem that matrix 2 is already transposed for us, so we're actually working with two $2 \times N$ matrices.



A

Adrian Bao 7mth #985fd

✓ Resolved

SP22-Final-Q6.1

Q6.1 (4 points) What lines of code should be written in the given blanks in the C code?

Solution:
<BLANK 1>: `calloc(n, sizeof(char))` or equivalent.
<BLANK 2>: `primes == NULL` or equivalent.

For blank 1, I wrote `malloc(n * sizeof(uint32_t))`; Would that work?

♡ ...

 **Anonymous Louse** 7mth #985aae

No, it's given in the problem that the output of `primes` is an array of n 1-byte integers. I put `calloc(n, sizeof(uint8_t))`, which I think is equivalent? You can't `malloc` here because you need to initialize all elements of the array to 0 (can you see why?)

♡ 1 ...

A **Adrian Bao** 7mth #985fc ✓ Resolved

SP22-Final-Q5.2

By doing `lw rd 0(sp)`, how do we know that we are "setting `rd` to the bottom-most word of the stack?"

♡ ...

 **Anonymous Louse** 7mth #985aaf

The stack pointer points to the bottom of the stack, where everything above it is memory that's being used and everything below is garbage memory.

♡ 1 ...

 **Catherine Van Keuren** STAFF 7mth #985afd

To add on to Louse, when we `lw` with offset of zero, we know that that's the item stored at the address of the stack pointer. If we say we were loading with an offset of 4, we would be loading from 4 bytes above the bottom of the stack.

♡ 1 ...

A **Adrian Bao** 7mth #985fb ✓ Resolved

SP22-Final-Q3.8

I thought both don't follow calling convention, as both `t0` and `s0` are not being saved in a register, instead they are being saved in `a0`, which will lose its value if another function is called.

Q3.8 (3 points) Given behavior: A function to return the square of a number.

A	B
square: mul t0 a0 a0 mv a0 t0 ret	square: mul s0 a0 a0 mv a0 s0 ret

A is incorrect B is incorrect Neither is incorrect (i.e. both are correct)

Solution:

Advantages of A

- (Answer if B is considered incorrect) Follows calling convention.

Advantages of B

- None

Common answers that were NOT given credit

- Any statement that suggests that calling convention violations are acceptable (including trying to read `t0` or `s0` after the function returns) was considered incorrect. From an engineering perspective, calling convention errors are the worst possible error, because they appear to work. It is in general preferable to have a program that doesn't work over a program that looks correct, but has a subtle bug that breaks everything three years down the line, since if you have the former, you know that you need to fix something.

♡ ...

 **Justin Yokota** STAFF 7mth #985afb

We don't call another function during square, so we don't need to worry about losing its value.

♡ 1 ...

A **Adrian Bao** 7mth #985fa ✓ Resolved

SP22-Final-Q1.1


What instructions are not 32 bits long?

♡ 1 ...

J **Justin Yokota** STAFF 7mth #985abf

Instructions in the 16-bit RISC-V extension, none of which were discussed in class (though you should know that they exist due to our discussion of branch and jump immediates).

♡ ...

 **Anonymous Louse** 7mth #985ee ✓ Resolved

SP22-Final-Q3

Question 3.2

Solution: C

- Pass by value makes things quick
- Faster runtime by far
- Lower-level language means it's easier to translate to assembly
- No explicit garbage collection
- Gives you greater freedom over your program (you can do things that Python won't let you)

Why do they say "Pass by value makes things quick"? I thought when you pass by value, you make a copy of the data, whereas in pass by reference you're referencing the same data? So isn't pass by reference more efficient in general?

♡ ...

J **Justin Yokota** STAFF 7mth #985ef

That's true if the values you're working with are objects or structs. In C, we always send around pointers instead of the objects, so we don't actually waste time setting up copies; it's practically instantaneous to copy a single int. But by avoiding pass-by-reference, we don't need an additional layer of pointers implicitly required by the language, and this saves time.

♡ ...

 **Anonymous Salamander** 7mth #985ed ✓ Resolved


SU22-Final-Q7.1 Why it's 2^{13} not 2^{12} since there are 3 hex digits for the offset?

♡ ...

J **Jero Wang** STAFF 7mth #985bef

Offsets are counted in bits, not nibbles, and the two numbers share the lowest 13 bits.

♡ ...

 **Annika Liu** 7mth #985de ✓ Resolved

SU22-Final-Q8

Hello! In the solutions for Q8 we don't need to typecast `int32_t *` to `__m128i *` but in project 4 we do need to typecast it, will we be counted as incorrect if we typecast them within functions like `vload` which specify that it should be `__m128i *` argument? Thank you!

♡ ...



Jero Wang STAFF 7mth #985bee

You shouldn't need to typecast it in Project 4 either, and you will not be marked down for casting/not casting. However, if you cast, please note order of operations (e.g. `(__m128i *) vec + i` is not the same as `(__m128i*) (vec + i)`).

♡ ...



Annika Liu 7mth #985dd

✓ Resolved

SU22-Final-Q2

Hi! I want to check if it is convention to type cast malloced memory space to the desired pointer type. Thanks!

♡ ...



Catherine Van Keuren STAFF 7mth #985afc

It's convention to have the casting, but it's not required. You won't be counted off for not having it.

♡ ...



Annika Liu 7mth #985bbe

thank you!

♡ ...



Annika Liu 7mth #985dc

✓ Resolved

SU22-Final-Q2

Hi! I want to make sure I got the concept right:

so for fields within a struct, if they are pointers e.g. `char *` they are not assigned to an allocated memory space when the struct is instantiated so they cannot directly be used as destination in `strcpy` but they can still be assigned to another pointer (i.e. assigned the memory space of that pointer). If it is an array, e.g. `char temp[n]` then it is allocated a memory space when the struct is instantiated so it can be used as a destination in `strcpy`. However, it cannot be directly assigned to a pointer because the number of elements in the pointer may not match the number of element in the array.

Are my understandings correct? Thank you so much!

♡ 2 ...



Jero Wang STAFF 7mth #985bed

Yes, that sounds right!

♡ ...



Annika Liu 7mth #985dad

thank you!

♡ ...

 **Anonymous Dove** 7mth #985db ✓ Resolved

No Fall 2022?

♡ ...

 **Catherine Van Keuren** STAFF 7mth #985df

I believe we removed FA22's exams due to them not being representative of our usual exams because of the strike. (Someone can correct me on this though)

♡ 1 ...

 **Anonymous Crow** 7mth #985bd ✓ Resolved

SU22-Final-Q7.2

What does it mean if all leading zeros are removed? The virtual memory address is 1010101111001101001000100011 in binary which has no leading zeroes?

♡ ...

 **Andrew Liu** STAFF 7mth #985ca

Hi, Q3.4 does not appear to be about virtual memory, can you update the question number please?

♡ ...

 **Anonymous Crow** 7mth #985cb

Fixed!

♡ ...

 **Andrew Liu** STAFF 7mth #985cd

↩ Replying to Anonymous Crow

Thanks! This question is asking about the minimum size of a VA. Since the given address has exactly 28 bits (with a 1 in the leading position), we know that the minimum size must be 28 bits.

We can't discount for example, a 32-bit address — maybe the entire address was 0x0ABCD123 or even 0x00000ABCD123. However, we do know that if we see 28 significant bits, we have a lower bound on VA length.

♡ ...

 **Anonymous Crow** 7mth #985bc ✓ Resolved

SU22-Final-Q3.4

Is self-modifying code in scope? How do we gain intuition for solving these kinds of problems?

♡ ...

 **Justin Yokota** STAFF 7mth #985be

Anything that constitutes valid RISC-V code is considered in scope. I would say that it's largely the same as problem-solving with other coding problems; the main difference is that here you lose the assumption that the code you write will remain constant through the entire run of the program.

♡ ...

 **Anonymous Marten** 7mth #985bb ✓ Resolved

Spring 2022,

I don't understand the highlighted absorption law step here. Could someone explain this?

Q2.2 (2 points) Simplify the following Boolean expression. For full credit, your solution must use at most 3 boolean operators (OR, AND, NOT).

$$!((A + !A)B + (CD)) + CD$$

Solution: $!B + CD$

First, note that $(A + !A) = 1$, which reduces the expression to $!(B + (CD)) + CD$.

Then, using DeMorgan's law reduces the expression to $!B + !(CD) + CD$.

Finally, using absorption reduces the expression to $!B + CD$.

An intuitive way to see the absorption step: if CD is 0, then the expression reduces to $!B$. If CD is 1, then the expression reduces to 1.

Grading: Partial credit was awarded for expressions that correctly matched the behavior of the given expression, but had more operators than optimal. Each extra operator lost 0.4 points, with the minimum of 0 points at 8 operators (the same number of operators as the original expression).



Andrew Liu STAFF 7mth #985cc

Absorption Laws are typically given as

$$x + (xy) = x \quad x(x + y) = x$$

But the following formulation is also sometimes referred to as an absorption law:

$$x + (\bar{x}y) = x + y$$



Anonymous Marten 7mth #985aaa

How is that being used here?



Anonymous Louise 7mth #985ec

You can also write $!B + !(CD) + CD$ as $!(B + CD) + CD$ using distributivity, and then it reduces nicely from there.



Anonymous Marten 7mth #985ba

✓ Resolved

Spring 2022

Do we need to know how to work with or do these calcs with a two-level hierarchical page table? I think we skipped over this in lectures this year.

Noticing that this is inefficient, we decide to use a two-level hierarchical page table with no TLBs. Our VPN bits are split evenly among the two levels.

Q7.5 (1.5 points) How many PTEs are in the L1 page table?

Solution: 2^{12} PTEs

From Q7.3, we know that there are 24 bits in the VPN. If we split these bits evenly among the two levels, we have 12 bits for uniquely identifying a L1 PTE. This gives us 2^{12} possible PTEs in the L1 page table.

Q7.6 (1.5 points) How many pages worth of memory does a single L2 page table take?

Solution: 8 pages

As in the previous part, splitting the bits evenly gives us 12 bits for uniquely identifying a page within a single L2 page table. This means that a single L2 page table contains 2^{12} PTEs. From the top of the question, each PTE is 4 bytes long, so a single L2 page table takes up $2^{12} \times 4 = 2^{14}$ bytes.

One page of memory is 2 KiB = 2^{11} bytes, so a single L2 page table takes up $2^{14}/2^{11} = 2^3 = 8$ pages of memory.



Catherine Van Keuren STAFF 7mth #985ea

Multilevel page tables are out of scope this semester



Anonymous Marten 7mth #985ae

✓ Resolved

Spring 2022

Is this in scope/did we ever go over this in lecture? If so, what is the main idea we should know?

Q3.4 (3 points)	
A RAID-0 array with 5 2TB disks	
A RAID-5 array with 5 2TB disks	

Solution: A RAID-0 array with 5 2TB disks

- Faster access time (reads, writes).
- Can physically store more data.
- No (technically very small, but out of the scope of this course) overhead to set up.
- Lower latency.
- Higher bandwidth.



Anonymous Marten 7mth #985af

Similarly, Is this in scope/did we ever go over this in lecture? If so, what is the main idea we should know?

Q3.5 (3 points)	
Polling	
Interrupts	

Solution: Polling

- Allows for “always-busy” devices to continuously send data.
- Has an overall average lower latency.
- There is a deterministic response time per event.
- Does not have to redirect CPU resources from processes after they’ve started to service I/O.

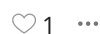
Solution: Interrupts

- Handles input from devices that require less CPU.
- Allows CPU to execute other processes when there is no data.
- When used correctly, typically will use less power as it relies on user input to be triggered.



Andrew Liu STAFF 7mth #985cf

I believe I/O was touched on in Lecture 24: FSM / OS, but I will check and get back to you if Polling / Interrupts are in scope this semester



Anonymous Marten 7mth #985eb

↩ Replying to Andrew Liu

Thanks!





Anonymous Hummingbird 7mth #985adc


↩ Replying to Andrew Liu

Any updates on interrupts?



 **Anonymous Beaver** 7mth #985bdb
↩ Replying to Anonymous Hummingbird
#939aad
♡ ...

 **Andrew Liu** STAFF 7mth #985ce
RAID is out of scope for this semester.
♡ 2 ...

 **Anonymous Marten** 7mth #985ad ✓ Resolved
Spring 2022

Could someone explain what is meant by "implicit 0" here?

Q1.1 (1 point) TRUE or FALSE: All RISC-V instructions are 4 bytes long, which is why the immediates for B-type and J-type instructions each have two implicit 0s.


TRUE


FALSE

Solution: False. Not all RISC-V instructions are 4 bytes (32 bits) long. The base set, RV32 which is what we work with in this class indeed uses 32-bit instruction lengths, but B-type and J-type instructions under RV32 encode immediates that support addressing of 16-bit addresses, resulting in both immediate encodings to have just **one** implicit 0 each. This is to not only help support legacy code for when memory spaces were small enough to be addressed by 16-bit addresses, but also to help 16-bit compressed RISC-V instructions.

Grading: All-or-nothing.


♡ ...


 **Andrew Liu** STAFF 7mth #985da
The implicit zero is like the zero that we assume to be the LSB of the branch / jump immediate. (Remember that we store all but the last bit in the instruction encoding)
♡ ...

 **Anonymous Crow** 7mth #985ac ✓ Resolved
SU22-Final-Q3.3

Would jalr x0 x0 -256 also work? Since the loop: label is defined 8 instructions above and we can do relative PC addressing (backwards $8 \cdot 32\text{b}$ instructions)?

♡ 1 ...

 **Anonymous Mallard** 7mth #985bdf
Wondering the same thing! I believe it should be $8 \cdot 4$ though because it is in bytes.
♡ ...

 **Jero Wang** STAFF 7mth #985bec
This would jump to the address -256, not 256 bytes before the current PC (since jalr jumps to $rs1 + imm$, not just imm).

Anon Mallard is correct though - should be in terms of bytes so 32 not 256.

♡ ...



Anonymous Bat 7mth #985cca

Could we use `auipc t0 0` to load the PC into `t0` then do `jalr x0 t0 -32`?



Jero Wang STAFF 7mth #985cdb

← Replying to Anonymous Bat

I think the offset is `-28` but yeah, `auipc` would work here.



Anonymous Crow 7mth #985ab

✓ Resolved

SU22-Final-Q2

Why don't you need to `strcpy` `name` (wouldn't it be cleared after a free to the input string pointer)?

Also, "It also doesn't work because you're assigning the static type `char contents[X]` to a `char*`; however the reverse is fine" why is this the case?



Jero Wang STAFF 7mth #985beb

I don't think this was worded well, but I think the expectation was that if `name` was freed, it would have undefined behavior. The reverse is fine because a `char *` can point to any string, whereas a `char contents[X]` is specifically a string on the stack, so it can't point to any string.



Anonymous Shrew 7mth #985aa

✓ Resolved

SU22-Final-Q5

Is this question in scope?



Andrew Liu STAFF 7mth #985bf

No, ECC is not in scope this semester.





Anonymous Alpaca 7mth #985e

✓ Resolved

Q1 True/False

(6 points)

Q1.1 (1 point) TRUE or FALSE: All RISC-V instructions are 4 bytes long, which is why the immediates for B-type and J-type instructions each have two implicit 0s.

TRUE

FALSE

Solution: False. Not all RISC-V instructions are 4 bytes (32 bits) long. The base set, RV32 which is what we work with in this class indeed uses 32-bit instruction lengths, but B-type and J-type instructions under RV32 encode immediates that support addressing of 16-bit addresses, resulting in both immediate encodings to have just **one** implicit 0 each. This is to not only help support legacy code for when memory spaces were small enough to be addressed by 16-bit addresses, but also to help 16-bit compressed RISC-V instructions.

Grading: All-or-nothing.

Q1.2 (1 point) TRUE or FALSE: Most computers natively use a little-endian data encoding.

SP22-Final-Q1.1

I'm confused where the 16 bits is coming from?

♡ ...



E Eddy Byun STAFF 7mth #985f

RISC-V supports 16-bit instructions, so our jump and branch instructions need to support these instructions as well!

♡ ...



Anonymous Loris 7mth #985d

✓ Resolved

SP22-Final-Q2.3-2.5

Are these questions in scope for the final?

♡ ...



J Jero Wang STAFF 7mth #985bea

2.3 and 2.4 are, 2.5 is not.

♡ ...



John Chang 7mth #985a

✓ Resolved

SP22-Final-Q3

Are statically and dynamically linked libraries in scope?

♡ ...



M Myrah Shah STAFF 7mth #985b

It's covered in [Lecture 14: Slides 30-31](#)

♡ 1 ...



J John Chang 7mth #985c

omg thank u so much GOAT

♡ ...