! This class has been made inactive. No posts will be allowed until an instructor reactivates the class.

note @2128 408 views

Actions

[Past Exams] 2021

You can find the past exams here: https://cs61c.org/sp22/resources/exams/

Spring 2021 final walkthrough

When posting questions, please reference the semester, exam, and question in this format so it's easier for students and staff to search for similar questions:

Semester-Exam-Question Number

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

exam exam/final

good note | 0 Updated 5 months ago by Jerry Xu and Peyrin Kao

followup discussions, *for lingering questions and comments*

Resolved Unresolved @2128_f1

This was marked a duplicate to the question/note above by Peyrin Kao 5 months ago

Anonymous Atom 5 months ago

[fa21-final] Questions

[fa21-final-5.3] Can anyone explain this solution?

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

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

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

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

```
mac a0 a1 a1
mac a0 a1 a1
```

[fa21-final-5.2] Why "add a new rs3 input to RegFile" is incorrect?

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

`mac rd, rs1, rs2`

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

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

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

[fa21-final-1.8] Why 9??

Q1.8 (0.5 points) RAID 5?

[ ] TiB

Solution: In RAID 5, we get one parity block for every 9 data blocks, so we get a total capacity of 9 TiB

helpful! | 0

Adelson Chua 5 months ago

Can anyone explain this solution?

Since the mac instruction needs the value of rd to complete the operation (rd + rs1*rs2), the following sequence will cause a data hazard due to dependency on a0 (rd)

mac a0 a1 a1 #a0 will be updated here

mac a0 a1 a1 #updated a0 is needed here, data hazard.

However, if this is replaced with a mac instead, there is no dependency on rd, since it is just the destination register.

mul a0 a1 a1 #a0 will be updated here

mul a0 a1 a1 #a0 will be updated here, no data dependency, no hazard.

[fa21-final-5.2] Why "add a new rs3 input to RegFile" is incorrect?

Look at the instruction description, there's no rs3 needed. It's just rs1, rs2, rd.

[fa21-final-1.8] Why 9??

Because there are 10 disks, 1 disk worth of parity is distributed among the remaining disk. 10-1 = 9.

good comment | 0

Anonymous Mouse 5 months ago

Why we need to wait for the value of rd for mul?

helpful! | 0

Anonymous Mouse 5 months ago

don't need

helpful! | 0

Adelson Chua 5 months ago

The second mul instruction does not depend on the result of the first one.

good comment | 0

Anonymous Beaker 5 months ago

What is `mac` ?

helpful! | 0

Peyrin Kao 5 months ago

`mac` is a new instruction defined in the context of that exam question.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f2

Anonymous Comp 5 months ago

FA21-Final-Q6.3

Q6.3 (2.5 points) Par 3

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

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

Solution: Staff solution: $W \wedge Z$

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

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

Could someone help me with this problem if I were to brute force this problem with boolean algebra? This is my work so far:
(~W & Y & Z) | (W & Y & ~Z)
Y((~W&Z) | (W&~Z))
Y(W ^ Z)

I'm not really sure how you get rid of the Y

helpful! | 0

Anonymous Comp 5 months ago
FA21 Final 6.3 ^^
helpful! | 0

Peyrin Kao 5 months ago
Your first sum-of-products expression is assuming that the four X values in the table are 0s, so brute-forcing here won't help you get rid of Y. In other words, that sum-of-products expression isn't equivalent to the truth table because of the X values.
good comment | 0

Anonymous Comp 5 months ago
Does that mean this problem can be only solved by eye? Or is there a correct way to create a sum-of-products expression?
helpful! | 0

Adelson Chua 5 months ago
There is a methodical way using K-maps but this isn't covered in 61C. I'm guessing for this problem, this is mainly through inspection.
good comment | 1

Eugenia Chien 5 months ago
As someone who took this final, I solved this problem by inspection :)

I like to watch for which inputs affect the output, i.e. what inputs can change without affecting output, what inputs are always the same when the output is some value, etc.
helpful! | 0

Anonymous Comp 3 5 months ago
Wait isn't the answer (W $\oplus$ Z)? Since (W $\wedge$ Z) evals to 0 when W and Z are 1
helpful! | 0

Anonymous Comp 3 5 months ago

Nvm didn't see the logic table in instructions

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f3**

Anonymous Helix 5 months ago

Question 2 Sp 21 Final:

Working with cache size 512 KiB, 128 B Blocks.

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

For the part below, why is the solution 8?

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

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f4**

Anonymous Atom 5 months ago

[q6b5] Why is slli t1, t1, 4 instead of 2?

v. load sl->next[level] into t0

```
slli t1 t1 4 add t0 t0 t1 lw t0 t0(4)
```

helpful! | 0

Peyrin Kao 5 months ago

You're right; it's a typo in the solutions. @909_f71

good comment | 0

Anonymous Helix 3 5 months ago

Why do we slli by 2? I'm confused at what multiplying it by 4 does?

helpful! | 0

Peyrin Kao 5 months ago

See @2128_f47 and remember that a pointer is 4 bytes long.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f5**

Anonymous Gear 5 months ago

Can someone explain to me how this works? If x = -1 doesn't the while loop keep running still? Since -1>0.

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

```
1  int y = 0;
2  int x = 10;
3  #pragma omp parallel
4  {
5    while(x>0)
6    {
7      y = y+1;
8      x = x-1;
9    }
10 }
```

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

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

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

helpful! | 0

Anonymous Gear 5 months ago

Oh wait. Sorry I get why it stops the loop but is this basically saying that thread 2 goes through 10 iterations of the while loop, but Thread 1 is still asleep? How is this possible?

helpful! | 0

Adelson Chua 5 months ago

Yes, that is possible. Different threads run at different times. Interleaving is not guaranteed.

good comment | 0

Anonymous Gear 5 months ago

If it was #pragma omp parallel for instead, would the threads have to run 1 iteration of the loop each or no?

helpful! | 0

Adelson Chua 5 months ago

There's no for loop though...

good comment | 0

Anonymous Gear 5 months ago

Sorry if it was a for loop too

helpful! | 0

Adelson Chua 5 months ago

#pragma omp parallel for

for(int i = 0; i<something; i++){
}

Each thread will have their own i values.

good comment | 0

Anonymous Comp 2 5 months ago

To clarify, even though x and y are shared variables, each thread can keep a local copy after they perform a read, and it's when the threads write to the shared variables that the value becomes visible to all threads?

helpful! | 2

Edrees Saied 5 months ago

^ I had the same question. How can thread 1 read x = 0, but y = 1? Shouldn't it also read y = 10?

helpful! | 0

Anonymous Poet 2 5 months ago

^ I also don't understand how thread 1 can write y=1. in order for it to write, it must have gone thru the loop once, but x is already 0 so how can it enter the loop?

helpful! | 0

Adelson Chua 5 months ago

To clarify, even though x and y are shared variables, each thread can keep a local copy after they perform a read, and it's when the threads write to the shared variables that the value becomes visible to all threads?

Correct.

You guys should put this back to "Unresolved" if you want us to see this.

good comment | 0

Anonymous Gear 3 5 months ago

Where does thread 1 read x = 0?

helpful! | 0

Adelson Chua 5 months ago

Once thread2 completes the loop (x becomes 0 at the end of the loop), only then will thread1 start to read x (at which point, x is already 0)

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f6**

Anonymous Scale 5 months ago

SP21-Final-Q2.A.I

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

As we're working on running the code snippet, we realise we want to run different instances of the

same code. We choose to employ virtual memory on our memory space. We have 4 GiB of virtual memory and 16 MiB of physical memory mapped with a single level page table with a page size of 4 KiB. We choose to store 8 bits of metadata with each page table entry.

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

1024

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

So the page table has 2^32 entries because the virtual memory has 2^32 Bytes. Each PTE must take 4 bytes => The page table takes 2^32 * 2^2 = 2^34 Bytes. Each page is 4KiB = 2^12 bytes. So the number of physical pages that the Page Table takes up should be 2^34 / 2^12 = 2^22 right?

Where am I miss interpreting the question? Thanks in advance.

helpful! | 0

Anonymous Scale 5 months ago

Nvm I'm an idiot, the number of PTEs is 2^(number of VPN bits), which in this case is 2^20, so then we have number of bytes in page table = 2^20 * 2^2 = 2^22. Which means that the number of pages that the page table takes up is 2^22 / 2^12 = 2^10 = 1024.

helpful! | 0

Anonymous Scale 5 months ago

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

8

In practice does this actually ever occur where we are trying to fit our page table into a cache? Doesn't the TLB serve this purpose?

helpful! | 0

Adelson Chua 5 months ago

Yeah. TLBs are the caches just serving a different purpose.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f7**

Anonymous Scale 5 months ago

SP21-Final-Q6.b

```
struct SLN{
    void *data;
    struct SLN **next;
}

/* Only the following in the code actually matters:
```

```
 cmp(find, sl->next[level]->data))
But if you are curious, look up "Skiplist"
*/


int SL_find(void *find, int level, SkipListNode *sl,
            (int)(*cmp)(void *, void *){
   if(cmp(find, sl->data) == 0) return 1;
   if(level == 0 && sl->next[level] == null) return 0;
   if(sl->next[level] == null) {
    return SL_find(find, level-1, sl, cmp);
   }
   if(cmp(find, sl->next[level]->data)) > 0)){
```

I don't quite understand the [level] part of the sl->next[level]->data part of the code. Looking at the struct, sl->next gives us back a pointer to a pointer to an SLN (skip list node). next[level] is the same operation as *(next + level), which makes it seem like this code wants to return a pointer to the level'th SLN node. But how do we know that SLN** pointers are exactly consecutive in memory?

helpful! | 0

Anonymous Scale 5 months ago

Just realized that the struct SLN** next is treated as an array not a pointer.

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f8**

Anonymous Scale 5 months ago

SP21-Final-Q7.C

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

Format: `R-Type` Instruction

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

I don't quite understand the solution, why do we need to modify the control signals? Are we trying to implement is_null using the ALU instead of the branch comparator?

https://inst.eecs.berkeley.edu/~cs61c/sp21/pdfs/exams/Sp21_Final_Solutions.pdf

helpful! | 0

Adelson Chua 5 months ago

Are we trying to implement is_null using the ALU instead of the branch comparator?

The branch comparator will be comparing inputs from the two registers, so using x0 as rs2 is feasible.

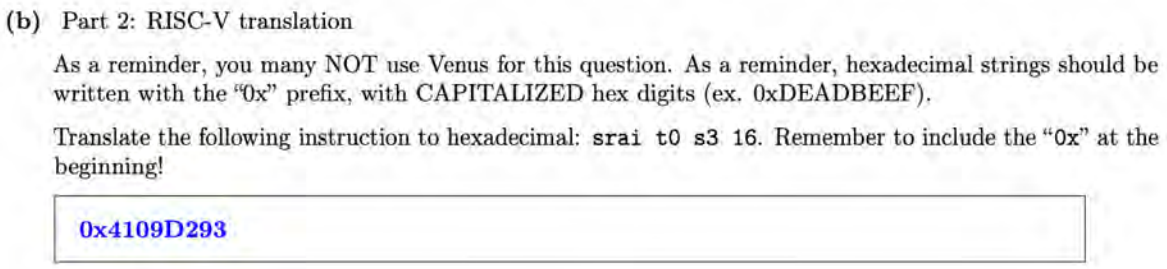
good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f9

Anonymous Scale 5 months ago

SP21-Final-Q8.b



(b) Part 2: RISC-V translation

As a reminder, you many NOT use Venus for this question. As a reminder, hexadecimal strings should be written with the "0x" prefix, with CAPITALIZED hex digits (ex. 0xDEADBEEF).

Translate the following instruction to hexadecimal: `srai t0 s3 16`. Remember to include the "0x" at the beginning!

0x4109D293

I got the same answer as the solution except for the first 4. The first 12 bits are the immediate, and the immediate of 16 should be 0b0...010000. And in hex this gives us 0x010 right? I'm not sure where the 4 comes from.

helpful! | 0

Adelson Chua 5 months ago

srai is I* instruction type. There's a funct7 included in the upper bits.

Be very careful.

good comment | 1

Anonymous Scale 5 months ago

ah got it.

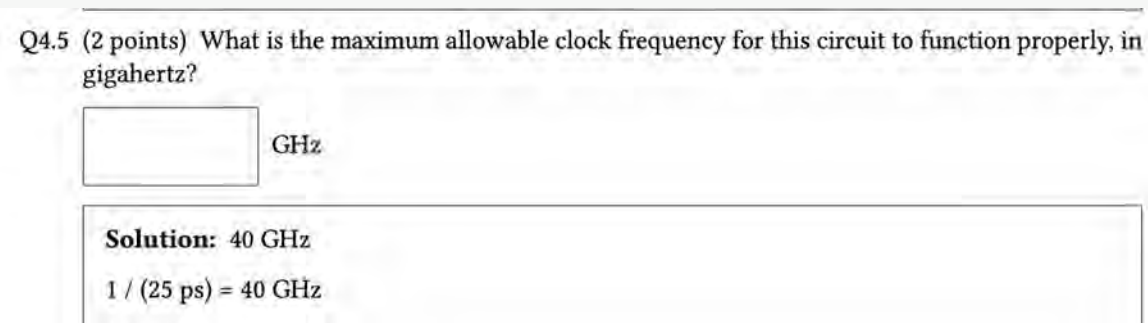
helpful! | 0

Reply to this followup discussion

Resolved Unresolved @2128_f10

Anonymous Poet 5 months ago

FA21-Final-Q4.5, where did this equation 1/(25ps) come from, and how did we get 40Ghz ?



Q4.5 (2 points) What is the maximum allowable clock frequency for this circuit to function properly, in gigahertz?

GHz

Solution: 40 GHz

1 / (25 ps) = 40 GHz

\

helpful! | 0

Peyrin Kao 5 months ago

Frequency is the inverse of period. If one clock tick takes 25 ps, or 25×10^{-12} seconds, then in one second, there are $\frac{1}{25 \times 10^{-12}} = \frac{1}{25} \times 10^{12} = 0.04 \times 10^{12} = 40 \times 10^{9}$ clock cycles, which is 40 GHz.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f11

Anonymous Calc 5 months ago

Su21 1.10.A (practice final on PrarieLearn)

Lecture 24 slide 53 says polling should be used for keyboards and this question says interrupts should be used. Am I misunderstanding the question or is this inconsistent?

Interrupts vs Polling

Computer Science 61C Spring 2022 — McMahon and Weaver

	Interrupts	Polling
If no I/O activity	No wasted cycles	Lots of wasted cycles
If lots of I/O activity	Expensive – saving/restoring state	Less expensive - only poll when we are already context switching
Better suited for events that are	• asynchronous (unsure when event will occur) • urgent • infrequent	• synchronous (occurs at fixed intervals) • not urgent • frequent (majority of polls are a hit)
Examples	Disk	Keyboard, mouse

PF1.10. Potpourri

PART A (2 pts.)

We want a CPU to process input from a keyboard. Which technique should we use to process incoming data?

(a) DMA

(b) Polling ✗

(c) Interrupts

(d) A mix of polling and interrupts

✗ 0%

helpful! | 0

Nicholas Weaver 5 months ago

This is one of those "it gets complicated" things. IN an ideal world we probably would actually want interrupts for keyboards due to the low data rate. But the technology we use for keyboards does not actually support interrupts! So instead its polling.

Overall the Polling/Interrupt distinction is really really messy.

Low data rate the ideal would actually be interrupts, but we don't have interrupts on USB ("interrupt" devices in USB speak literally must be polled!). But the rate is low enough that we can shove the polling into the timer interrupt so the overhead for polling in practice becomes very low.

High data rates you want to use interrupts to start/end the transaction through DMA, but if you need to process it as it comes in (e.g. network packets in high performance networking) you'll switch to polling because interrupts in practice are a huge overhead, as it effectively wipes out all the caches!

good comment | 0

Anonymous Calc 5 months ago

Thanks, that's interesting. But what should I answer if this question comes up on the test?

helpful! | 0

Nicholas Weaver 5 months ago

Uhh, scream that Nick said it won't because the answer is properly "Its Complicated"?

good comment | 6

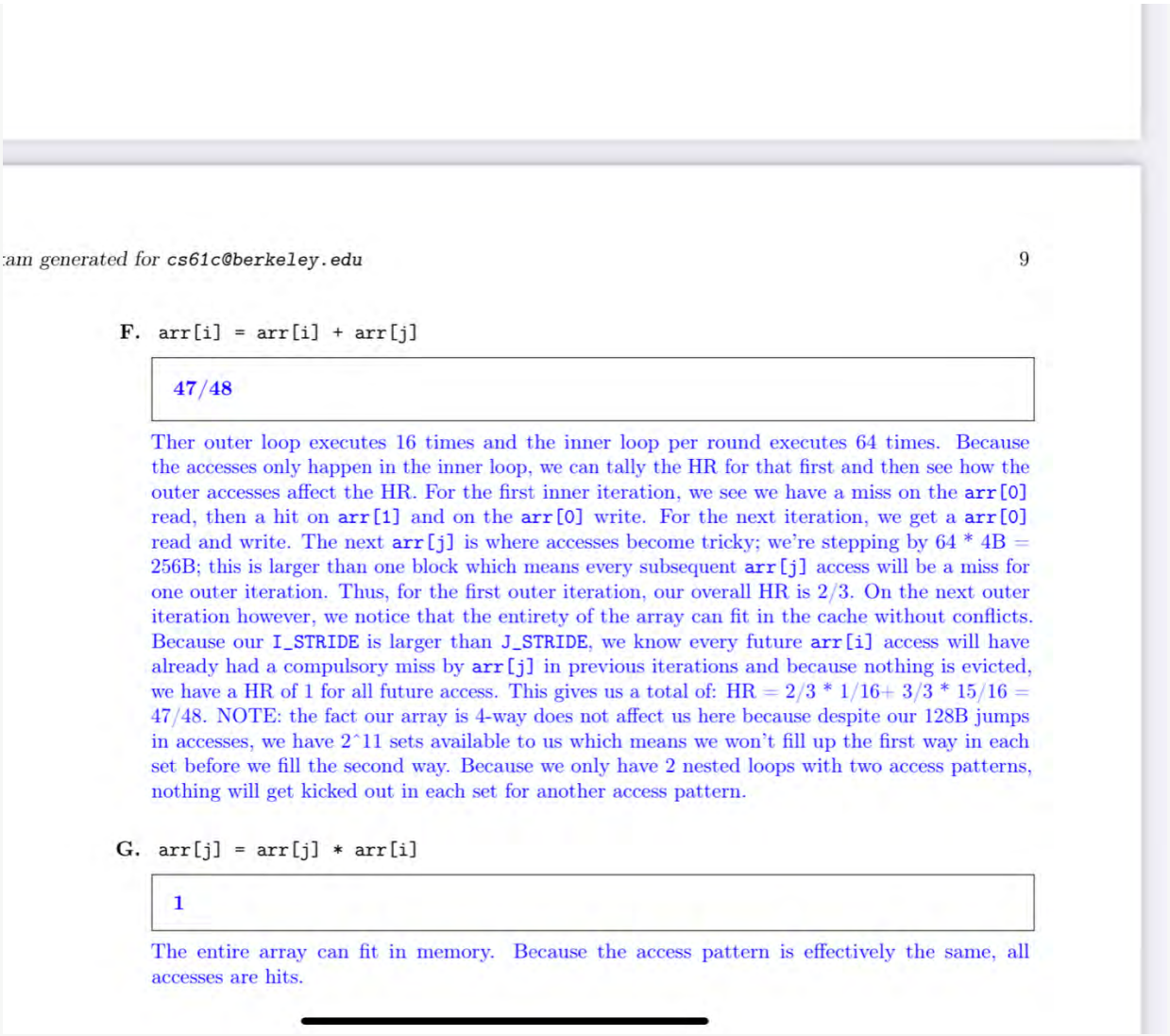
Reply to this followup discussion

Resolved Unresolved **@2128_f12**

Anonymous Poet 5 months ago

E.

```
1 #define base_arr_addr 0x12345678
2 #define ARR_SIZE 4096
3 #define I_BOUNDARY 2048
4 #define J_BOUNDARY 4096
5 #define I_STRIDE 128
6 #define J_STRIDE 64
7 int arr[ARR_SIZE];
8
9 uint32_t dummy_func(void) {
10   //Loop 1
11   for (int i = 0; i < I_BOUNDARY; i += I_STRIDE) {
12     for (int j = 0; j < J_BOUNDARY; j += J_STRIDE) {
13       arr[i] = arr[i] + arr[j];
14     }
15   }
16
17   //Loop 3
18   for (int j = 0; j < J_BOUNDARY; j += J_STRIDE) {
19     for (int i = 0; i < I_BOUNDARY; i += I_STRIDE) {
20       arr[j] = arr[j] * arr[i];
21     }
22   }
23 }
```



SP21-FinalQ2

I'm confused how we get 47/48, I see that for the first inner iteration, we have 2/3, since arr[j] will always be a compulsory miss, but why would we not have miss anything in 2nd to last outter iteration? I understand what it says about getting hit becuase arr[j] already accessed it, but I don't understand how I_stride larger than J_stride comes into play here.

Also, for G, how can hit rate be one? The first access should always be a compulsory miss right?

helpful! | 0

Adelson Chua 5 months ago

Write the consecutive memory accesses please.

For i = 0, all iterations of the inner loop will always be 2/3 hit rate. There are 16 total iterations of the outer loop. The other 15 iterations will be all hits because the first iteration already commited the compulsory misses. That's why there's the equation 2/3 * 1/16 + 15/16 = 47/48.

For G, this loop happens right after the first loop, where the cache has been loaded already.

good comment | 0

Yiteng Zhou 5 months ago

Why there are 2^11 sets? the index bit is only 10, right?

I know this doesn't affect the answer. I have a question that, what's the intuitive that the set won't be full and some blocks are evicted? Is that just because there are only 2^14 B integers, and our cache is 2^19B big?

helpful! | 0

Adelson Chua 5 months ago

Yeah 2^11 is probably a typo. 2^10 is logical.

Is that just because there are only 2^14 B integers, and our cache is 2^19B big?

Essentially yes. The cache fits the array, no replacements needed.

good comment | 0

Anonymous Scale 5 5 months ago

Are all the arr[j] entries in separate cache blocks? How do we know there are never any conflict misses?

helpful! | 0

Adelson Chua 5 months ago

At this point, it might be better to write out the consecutive addresses, split them into T/I/O, and check what the corresponding Index would be for every access.

You can start with array[0] = 0x00000000, then go from there. Iterate over the loop, take note of the stride length and see the consecutive addresses and how will they go into the cache.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f13**

Anonymous Poet 5 months ago

32 of 46

(d) (2.0 pt) What is the overall hit rate? Leave your answer as a fully simplified fraction.

1/2

The pattern above continues and repeats for all 8 blocks, giving us a 50% HR.

(e) (2.0 pt) What fraction of misses are coherency misses? Leave your answer as a fully simplified fraction.

3/4

Out of the 4 misses in each "access pattern block", 1 is compulsory, while the other 3 are coherency misses, so 75% of the overall misses.

(f) (1.0 pt) In total, how many times did we need to go to main memory to write-back?

0

As the array fits perfectly into the cache, we never need to evict an block and write-back, so 0.

(g) (2.0 pt) We want to avoid all the coherency misses, so we look to see if we can rewrite our code to optimize for cache performance. Which of the following methods will lead to a higher HR than that from the interleaved accesses?

- ☐ None of the other options
- ■ Letting processor 0 start and finish, then processor 1 starts and finishes
- ■ Letting processor 1 start and finish, then processor 0 starts and finishes

Both of these approaches would be better, since then there would be no coherency misses during the first processor's execution (load in the block, then hit on the other 3 WRW, so 75% HR). Then, the second processor would begin, but instead of compulsory missing, just coherency miss, but get the same HR pattern of MHHH for each block.

Su20-Final Q6, why would writback be 0? What does it mean by array fits into the cache perfectly?

helpful! | 0

Adelson Chua 5 months ago

Array length is 32 ints = 128 bytes. The cache size is 128 bytes as given.

good comment | 0

Anonymous Poet 5 months ago

But at the end, do we need to evict all cache lines from cache and writeback to main memory?

helpful! | 0

Adelson Chua 5 months ago

What do you mean by 'at the end'? End of what?

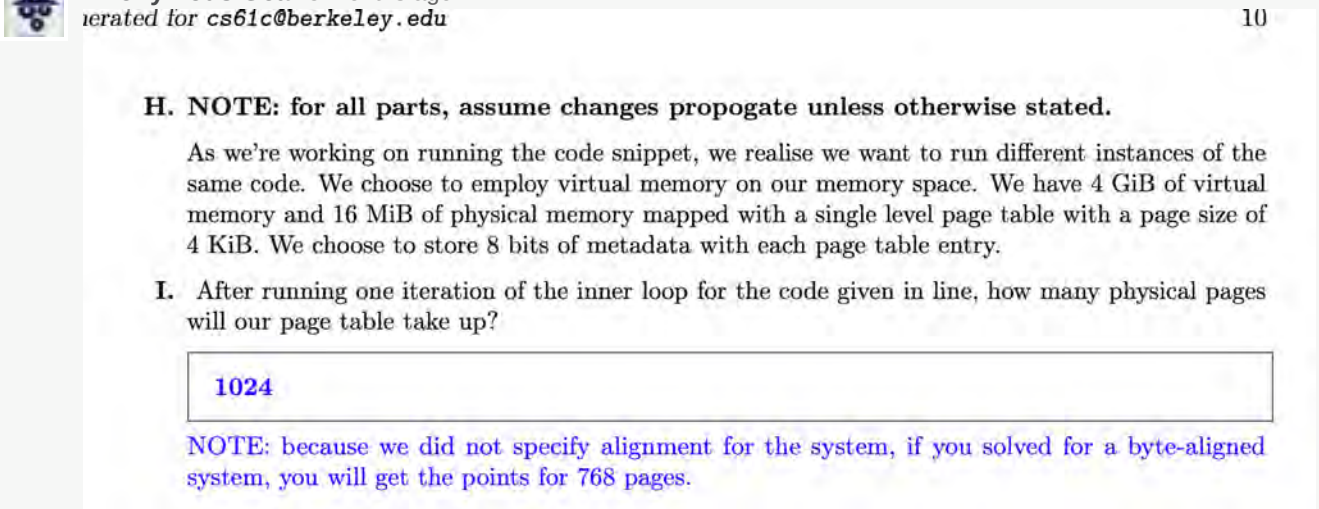
It is just saying that given the problem at hand, there's no need for cache evictions which leads to main memory write.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f14**

Anonymous Gear 5 months ago



ıerated for cs61c@berkeley.edu 10

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

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

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

1024

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

Can someone help me with this question in Sp 21? I can't understand how they got this. I tried adding up the bits of the VPN and PPN as well as the metadata, then multiplying by 64.

helpful! | 0

Adelson Chua 5 months ago

page offset = 12 bits

VPN = 32-12 = 20 bits

PPN = 24-12 = 12 bits

page table entry = 8 (metadata) + 12 (PPN) = 20 bits

Now we can either treat this as 3 bytes (if byte-aligned) or 4 bytes (if word-aligned)

If byte-aligned:

page table size = 2^20 (number of page table rows, dependent on VPN) * 3 = 3,145,728 (if byte-aligned)

how many pages the page table takes? page table size / page size = 3,145,728/(2^12) = **768**

If word-aligned:

page table size = 2^20 (number of page table rows, dependent on VPN) * 4 = 2^22

how many pages the page table takes? page table size / page size = 2^22/2^12 = 2^10 = **1024**

This is somewhat similar to the homework problem. You only need to find how large the page table entry is.

... Now where did you get 64?

good comment | 0

Anonymous Calc 3 5 months ago

Can you explain this sentence? I'm very confused

> Now we can either treat this as 3 bytes (if byte-aligned) or 4 bytes (if word-aligned)

helpful! | 0

Adelson Chua 5 months ago

20 bits is not divisible by 8 (a byte).

So we round up to the nearest byte. 24 bits = 3 bytes.

If word-aligned it should be divisible by 32 bits always. So in this case, 32 bits = 4 bytes.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f15**

Anonymous Atom 2 5 months ago

SP21-Final-Q2B

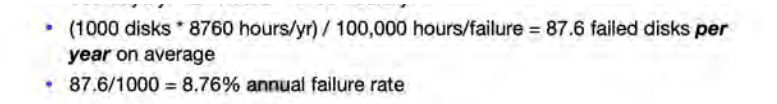
ii. Q2B

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

100

Can someone help me with how we get 100? I tried using the following but it does not seem correct.

- Another is average number of failures per year:
  Annualized Failure Rate (AFR)
  - E.g., 1000 disks with 100,000 hour MTTF
  - 365 days/yr * 24 hours = 8760 hours/yr



helpful! | 1

Peyrin Kao 5 months ago
I think the difference here is that in the exam, both MTTF and failure rate are measured in years, whereas in the slide, MTTF is in hours and the failure rate is measured in years.

For the exam: 1% of machines fail every year. To achieve that, any one machine fails once every 100 years.

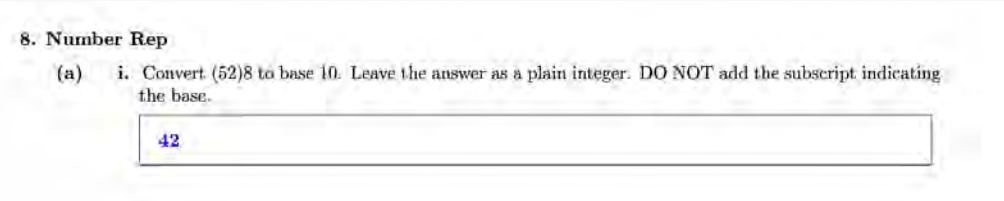
good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f16**

Anonymous Helix 2 5 months ago
SP21 Final, Question 8

What form is 52(8) in? Address 8 in hex with offset 52 bytes?


8. Number Rep

(a) i. Convert (52)8 to base 10. Leave the answer as a plain integer. DO NOT add the subscript indicating the base.

42

helpful! | 0

Adelson Chua 5 months ago
52 base 8.

good comment | 1

Anonymous Helix 2 5 months ago
lol maybe silly question, but where have we seen that notation to describe a base before?
I can only remember it as being a subscript or explicitly told to us

helpful! | 0

Peyrin Kao 5 months ago
I think it was subscript on the online exam, but the rendering got messed up on the PDF. The parentheses definitely aren't standard notation.

good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f17**

Anonymous Gear 5 months ago

iii. Which of the following is always true about our caching setup?

- (●) Assembly with few control-flow instructions will cause a high hit rate for Cache A.
- ( ) The AMAT for Cache A and Cache B will be the same since the caches are identical.
- ( ) Intended IMEM accesses will cause cache incoherence with intended DMEM accesses if the memory addresses are close.
- ( ) Instuction and data accesses will cause both Cache A and Cache B's states to change for every access.
- ( ) None of the above.

SP21 Final, Q7ciii:

Hi, Can someone explain to me why this is the answer? I thought the fourth one was the answer. Can someone please explain why it isn't? I thought the first one wasn't correct because I didn't think few control-flow instructions would affect caching.

helpful! | 0

Adelson Chua 5 months ago

Few control flow instructions allow you to maximize spatial locality of Cache A leading to high hit rates.

I'm not sure what 'states' are that is being referred to at the fourth option.

good comment | 0

Anonymous Atom 2 5 months ago

Could you also explain if this is the case, why subquestion 3 in the same question part B would not select option 4?

helpful! | 0

Adelson Chua 5 months ago

Because in that case, the cache is shared between instruction and data. If you are writing a lot of data, there might be cache block replacements going on which can evict blocks intended for instructions. Thus, it is not guaranteed that we are getting high hit rates in that scenario.

good comment | 0

Anonymous Scale 5 months ago

what is a "control flow" instruction?

helpful! | 0

Adelson Chua 5 months ago

Branches. Jumps.

good comment | 1

Reply to this followup discussion

Resolved | Unresolved @2128_f18

Anonymous Poet 5 months ago

(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: $4096Bytes/4Bytes = 1024$

SU20-MT2-Q1

I'm confused about the concpet of storing page table entries on a page, my understanding is that page table have PTE, and PTE map virtual pages to physical pages, and the page size correspond to the size of each page in virtual page and phyical page. For this question, I don't understand why we want to store page table entries on a page.

helpful! | 0

Adelson Chua 5 months ago

But... this is how it works. This should have been covered in the lecture. Also, there is a similar problem like this in the homework.

Page tables are stored as pages in the main memory, just like data pages are stored as pages as well.

Lab 10 slides (uploaded in the lab specs) illustrate this concept following CAMERAs setup.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f19**

Anonymous Calc 5 months ago

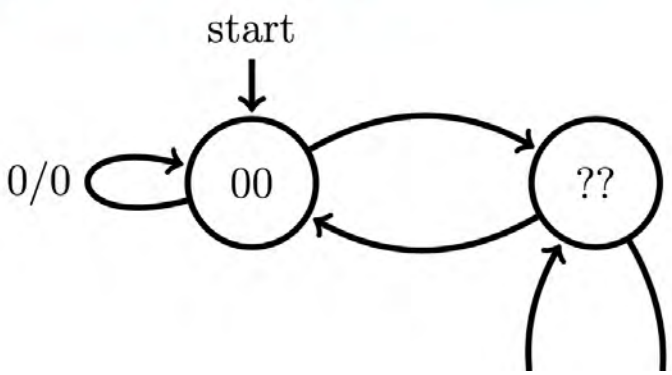
SU21-FINAL-Q3

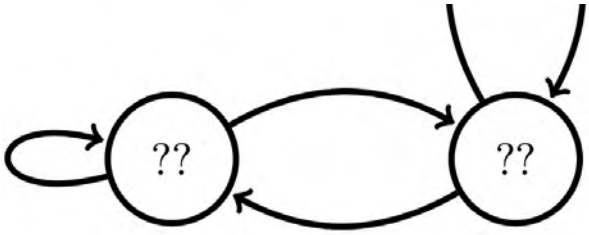
I don't remember anything about finite state machines, are they in scope?

PART A (8 pts.)

Fill in the transitions for the below FSM which represents the aforementioned 2-bit Gray code counter.

Note: `state_msb` is the most significant bit of the counter, and `state_lsb` is the least significant. That is, if the counter is at `10`, then `state_msb` is 1 and `state_lsb` is 0.





helpful! | 0

Anonymous Calc 5 months ago
Sorry this is supposed to be question 3!!
helpful! | 0

Adelson Chua 5 months ago
Out of scope. We didn't cover this in the lecture.
good comment | 1

Reply to this followup discussion

Resolved Unresolved @2128_f20

Anonymous Scale 5 months ago
FA21-FINAL-Q1.9

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

```
1  int y = 0;
2  int x = 10;
3  #pragma omp parallel
4  {
5     while(x>0)
6     {
7        y = y+1;
8        x = x-1;
9     }
10 }
```

Just curious, do we ever use Open MP with while loops in practice? It seems hard to divide the work amongst several hardware threads if we don't know how many iterations the while loop is going to run. My guess is some compilers can figure it out, but there are also cases of "while (keep looping until we get a result)" which may be more difficult for the compiler to figure out how many iterations the loop will run.

helpful! | 0

Adelson Chua 5 months ago
Typically, you use this in a for loop. **#pragma omp parallel for** is the best way to go, just like how

you did it in Proj 4.

Doing multi-threading on while loops are indeed hard and tricky, haha.

good comment | 1

Anonymous Mouse 5 months ago

So the answer is y = 1?

helpful! | 0

Adelson Chua 5 months ago

Yes

good comment | 0

Anonymous Poet 3 5 months ago

I'm confused why y can't be = 0. does x have to be read at the same time y was read before the for loop?

why can't thread 1 read x after thread 2 has run to completion and read x as 0?

helpful! | 1

Anonymous Poet 3 5 months ago

what line is thread 1 sleeping/stopping at?

I dont get how thread 1 is executing the while loop but reading x as 0.

helpful! | 1

Adelson Chua 5 months ago

Oh, so you mean thread1 does not start at all while thread2 starts and runs to completion?

That's a good question.

The solution assumes that when the threads spawn, they execute the *while (x>0)* statement at the same time allowing them to enter the loop.

Your claim is that thread1 does not start at all (does not execute any instructions). Not sure if that's possible.

I'll pull in some help.

good comment | 2

Justin Yokota 5 months ago

If thread 1 doesn't start at all, then thread 2 fully executes, then thread 1 fully executes, then thread 1 would set y to 10, and thread 1 wouldn't change; the end result would be that y = 10 at the end.

The specific order of operations that yields a y=1 state is if thread 1 goes to sleep after reading y in line 7, but before writing back the value of y in line 7. In that case, the work done by thread 2 on y gets undone by setting y to 0+1 = 1, and since thread 2 set x to 0, thread 1 won't continue to the next iteration of the while loop. The end result would be y = 1.

good comment | 3

Anonymous Comp 4 5 months ago

For this problem, after thread 1 reads y = 1, why does thread 2 get to read until completion? I though for pragma omp parallel, the stuff within the brackets from lines 4-10 get copied onto both threads, and both threads individually complete the loop

helpful! | 0

Adelson Chua 5 months ago

But they are sharing the same x and y variable (those are declared outside the pragma), so one thread can modify it and the other thread can see that modification. x and y are not private.

The execution of the threads are also non-deterministic. There's no guarantee that they start at the same time, at the same rate.

good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f21**

Anonymous Helix 2 5 months ago

Sp21 Question 6b

Is this a typo? I thought the syntax was always offset(mem location/register)

i. Load `find` into `a0`

```
lw a0 sp(0)
```

ii. The next thing we need to do is get sl->next[level]->data into a1.
The RISC-V code for that would be (fill in based on comments):

Load `sl` into `t0`

```
lw t0 sp(8)
```

iii. Load `level` into `t1`

```
lw t1 sp(4)
```

iv. load `sl->next` into `t0`

```
lw t0 t0(4)
```

helpful! | 0

Adelson Chua 5 months ago

Yeah, most likely a typo.

good comment | 0

Anonymous Gear 4 5 months ago

v. load `sl->next[level]` into `t0`

```
slli t1 t1 4 add t0 t0 t1 lw t0 t0(4)
```

vi. load `data` into `a0`

```
lw a0 t0(0)
```

vii. Finally, how do we call `cmp` (using `t0` as a temporary)

```
lw t0 12(sp) jalr ra t0
```

For v, why do we load 4(t0) and not 0(t0)?

helpful! | 0

Adelson Chua 5 months ago

@2128_f47

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f22**

Anonymous Scale 5 months ago

FA21-FINAL-Q8.4

Regardless of your answer to Q8.2, assume that physical addresses are 20 bits long. We run two programs (with no shared memory), which access the following virtual memory addresses in order. For each memory access, determine the physical address that gets accessed, writing your answer in hexadecimal.

Assume that no physical pages are in use prior to the first memory access, and that physical pages get assigned in order of physical page number (so page 0 is assigned first, then page 1, and so on).

Q8.4 (1 point) Program 1: 0xABCDEFAB

Solution: This is our first page, so we get the physical page $0x00$. The page offset is the last 12 bits of our address (since our page size is 2^{12} bytes), so we take the last 12 bits of the virtual address: 0x00FAB.

So with this question do we assume that the page tables of program 1 and 2 are stored in some place that isn't in memory? Because if they were stored in memory then it is possible that the page tables took physical page 0x00.

helpful! | 0

Adelson Chua 5 months ago

Oh nice, that's a good point.

You could ask for a clarification if this was asked in the exam. :)

I guess in this case, the page table is stored somewhere on a different address far away from 0x00.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f23

Anonymous Gear 2 5 months ago

I'm a bit confused as to how the last three questions of Part C in the Summer 2021 Practice Final on Prairie Learn are TLB misses and a page fault respectively.

ii) `0x00400C4B`

(b) TLB miss but page table hit

iii) `0x00000764`

(a) TLB hit

iv) `0x00000D6E`

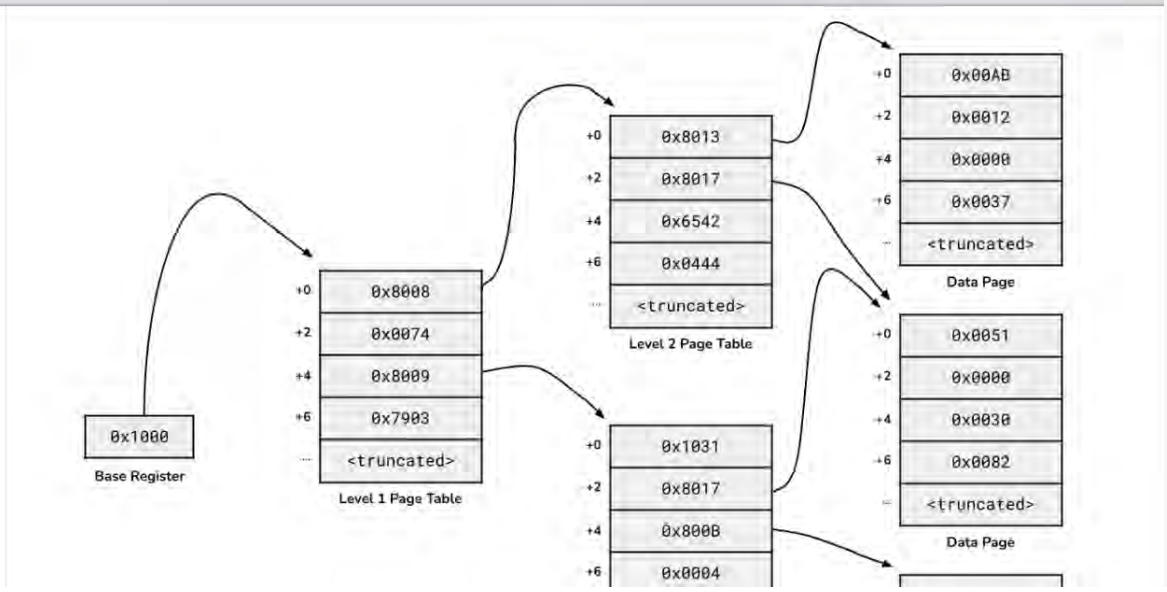
(b) TLB miss but page table hit

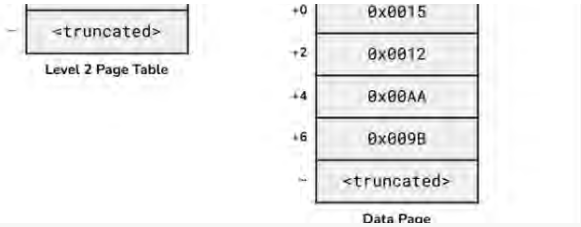
v) `0x00400FAE`

(b) TLB miss but page table hit

vi) `0x000016A7`

(c) Page fault





I understand that the first two will be TLB misses as the TLB starts off empty, but then after the next should be a TLB hit as it has the same PTE as the very first hexadecimal value checked in this problem. Due to this intuition, I was under the impression that the next hexadecimal value being checked would also be a TLB hit as it has the same PTE as the previous value check. My next question might be how exactly do we determine what PTE value is within the given hexadecimal? I assumed that the last 4 bits are the offset, while the rest are the PTEs split amongst 3 Page Table levels. I'm just confused how (iv) and (v) are TLB misses, and how (vi) is a page fault when they all have their upper 5 bits in the TLB.

helpful! | 0

Adelson Chua 5 months ago

This is only a 2-level page table...

Given my solution on the follow-up below. Can you follow the correct procedure and write out the corresponding VPN - PPN translations for every memory access?

We need to correctly identify the VPNs for the different addresses.

good comment | 0

Anonymous Calc 2 5 months ago

For vi for this question, is it a page fault because the level 2 page table or ppn that we receive has no arrow to a data page?

helpful! | 0

Adelson Chua 5 months ago

The valid bit is 0.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f24**

Anonymous Gear 2 5 months ago

I'm not quite sure how to solve part B of the Summer Final Practice Exam on Prairie Learn:

PART B (3 pts.)

What physical address does virtual address `0x0000041A` map to? Write your answer in hexadecimal, **without the leading** `0x`, or write `N/A` if this does not map to a valid physical address.

Displayed correctness for this blank may be inaccurate, but total points on this question should be correct (see errata in Ed grades post).

0x `9C1A`

To solve this, I originally decided to split up the given hexadecimal value into its respective parts: the upper

4 bits are the PTEs, while the last 4 bits represent the offset. I used the bits I assumed to be the PTEs to get to the desired page, and then copied the four bits as the first four bits of my desired output, and I simply added the offset as the last 4 bits (I got `00AB041A`). So I'm a bit confused how the actual answer is just 4 bits.

helpful! | 0

Adelson Chua 5 months ago

This question is somewhat similar to the homework question, only with different offset bits (this one is 11 bits only).

0x0000041A (VPN is 20 bits + offset is 11 bits = 31 bits)

000 0000 0000 0000 0000 0100 0001 1010 (This is the virtual address in 31-bit binary)

Partitioning:

0000000000 0000000000 10000011010

VPN1 VPN2 offset

Check the table.

VPN1 = VPN2 = 0. This points to the entry 0x8013. @2187 (take note of this)

PPN is part of the page table entry, and is only 11 bits long (the last 11 bits of the page table entry).

Last 11 bits is 0x013 => 000 0001 0011

Now, combine:

Physical address = PPN + offset

00000010011 10000011010

Regroup and rewrite to hex

00 0000 1001 1100 0001 1010

0x009C1A

The solution shows 0x9C1A which is also equivalent (PrairieLearn ignores leading zero for this problem for some reason)

good comment | 4

Reply to this followup discussion

Resolved Unresolved **@2128_f25**

Junsang Yoon 5 months ago

I don't understand FA2021 Q7

2: Miss; Our tag isn't in our index. This can either be compulsory or conflict, since we don't know if this block has been accessed before and ejected. This can't be a capacity miss, because there are still empty slots in our cache.

3: Miss; Our tag isn't in our index. This must be compulsory, because we still have an empty slot in this index. We only ever kick out a block when the cache is full, and only to replace that block with a new one; as such, we can't have kicked out a block while there is still empty space in the cache.

Where is the empty space in the cache?

helpful! | 0

Adelson Chua 5 months ago

Check the given cache state. There are cache slots that have a valid bit of 0. That's the empty space.

good comment | 0

Junsang Yoon 5 months ago

aren't we accessing index 00 in the 2), which has both valid bits of 1?

helpful! | 0

Adelson Chua 5 months ago

You look at the overall cache state when determining what type of miss it is.

@1270_f6

If the address was in the cache, but isn't anymore (got evicted) and the cache is still not full: Conflict miss

If the address was in the cache, but isn't anymore (got evicted) and the cache is full: Capacity miss

If the address was never in the cache (cold start): Compulsory miss

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f26**

Anonymous Mouse 2 5 months ago

Could someone explain to me the answer to sp21 mt q 5 parts c iv and d i ii?

(c) (1.5 points) Pipelining

Assume we've added pipeline registers to create a 6-stage pipeline with our updated datapath, as seen below (Figure 3). This pipelined datapath acts similarly to our standard RISC-V 5-stage pipeline with the additional change of the memory section being separated into two sections. Assume no forwarding logic has been implemented, no branch prediction, and one register operation per cycle.

For the given code, what hazards might exist and due to which lines? Assume all registers have been initialised and that all labels are defined and that all branches are taken. **HINT: having a table open will be useful for scratch work.**

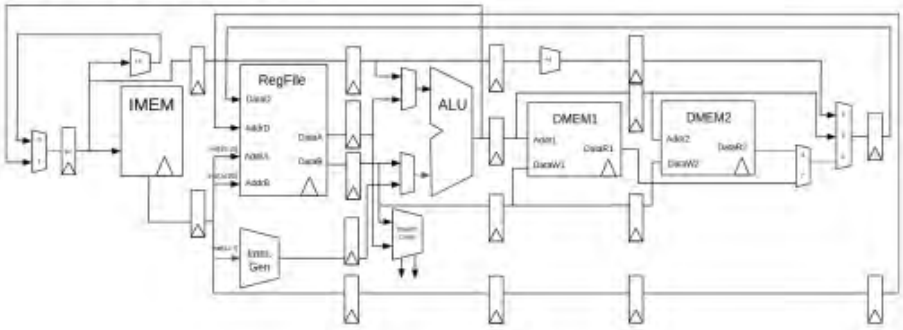


Figure 3

```
1 bne x0, t0, next
2 addi t1, t0, 1
3 lb s0, 0(t1)
4 shw s0, 4(t0)
```

helpful! | 0

Peyrin Kao 5 months ago

Instruction 1 needs to reach the execute stage to calculate the new PC address in the ALU, which can be written back into the PC on the next cycle. This requires two NOPs to line up the execute stage of instruction 1 to be just before the fetch stage of instruction 2.

```
1     F  D  E  M  M  W
NOP      F  D  E  M  M  W
NOP         F  D  E  M  M  W
2              F  D  E  M  M  W
```

run code snippet Visit Manage Class to disable runnable code snippets

Instruction 2 needs to reach the write-back stage to write the new value of t1 back into the regfile. This requires four NOPs to line up the write-back stage of instruction 2 to be just before the decode stage of instruction 3 (where the regfile is read).

```
2     F  D  E  M  M  W
NOP      F  D  E  M  M  W
NOP         F  D  E  M  M  W
NOP            F  D  E  M  M  W
NOP               F  D  E  M  M  W
3                    F  D  E  M  M  W
```

run code snippet Visit Manage Class to disable runnable code snippets

Instructions 3 and 4 have the same data hazard as instructions 2 and 3, so it requires another four NOPs.

2 + 4 + 4 = 10 NOPs

good comment | 0

Peyrin Kao 5 months ago

Not 100% sure about the second part of your question, but my thinking so far:

Adding forwarding paths doesn't change the two NOPs between instruction 1 and 2, because the path from the ALU output to the PC is unchanged (no forwarding added).

Adding the thick purple forwarding path from the output of the ALU to the input of the ALU means that the result of instruction 2 can be used after the execute stage of instruction 2 completes.

The thick green forwarding path from the output of DMEM2 to the input of the ALU means that

the memory read from instruction 3 can be used after the second memory stage of instruction 3 completes. This requires two NOPs.

```
<kbd>2   F  D  E  M  M  W
NOP </kbd>   F  D  E  M  M  W
NOP       F  D  E  M  M  W
NOP          F  D  E  M  M  W
3               F  D  E  M  M  W
```

run code snippet Visit Manage Class to disable runnable code snippets

2 + 0 + 3 = 5 NOPs

Forwarding paths from the end of DMEM1 to the start of DMEM1, and the end of DMEM2 and the start of DMEM2 don't help because it seems like the regfile output/ALU input is the bottleneck that's waiting for results in each of these instructions.

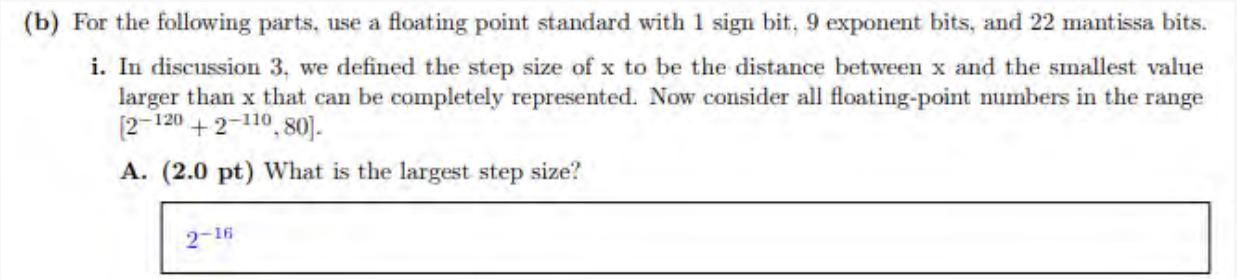
good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f27

Anonymous Mouse 2 5 months ago

Could someone explain why this answer is correct?



(b) For the following parts, use a floating point standard with 1 sign bit, 9 exponent bits, and 22 mantissa bits.

i. In discussion 3, we defined the step size of x to be the distance between x and the smallest value larger than x that can be completely represented. Now consider all floating-point numbers in the range $[2^{-120} + 2^{-110}, 80]$.

A. (2.0 pt) What is the largest step size?

2^{-16}

helpful! | 0

Adelson Chua 5 months ago

Step size increases as the exponent increases. For the given range, the highest exponent there would be the corresponding exponent for 80.

Write 80 in the 'scientific notation' = 1.01 * 2^6.

Given that we have 22 mantissa bits, adding 1 to the least significant bit of the mantissa translates to 0.0000...1 * 2^6 (we copy the exponent from 80 earlier, the largest exponent given the range).

0.000...1 = 1 * 2^-22 (since we have 22 mantissa bits).

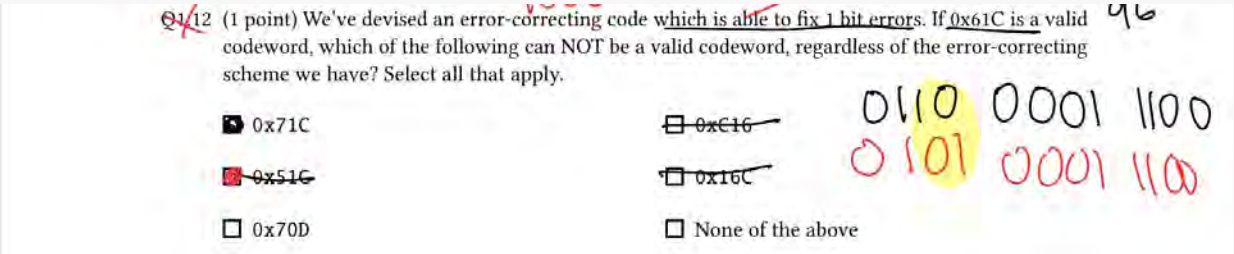
2^-22 * 2^6 = **2^-16**

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f28

Anonymous Scale 2 5 months ago



fa21-final-q1.12

I understand why 0x71c is correct, but not 0x51c. To not be a valid code word here, a number should be 1 bit away from a correct code word (0x61c), right? I'm confused because to go from 0x61c to 0x51c, it looks like two bits would have to be flipped?

helpful! | 0

Adelson Chua 5 months ago

This looks like it comes from a video, can you give the link? How did they explain the 0x51C?

good comment | 0

Anonymous Scale 2 5 months ago

Oh this screenshot isn't from a video, this is the work I did to try to solve the problem.

helpful! | 0

Adelson Chua 5 months ago

Oh ok. Does the provided explanation makes sense?

It's basically just saying that a valid possible codeword is something that has the *most number of bits away* from the original codeword. That's why in terms of hamming distance, you want the most number of bits changing as possible to encode two consecutive codewords.

good comment | 0

Anonymous Scale 2 5 months ago

Can you please clarify what the "most number of bits away from the original code word" means?

helpful! | 0

Adelson Chua 5 months ago

Most number of bit differences.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f29

Anonymous Beaker 2 5 months ago

Can a TA quickly explain RISC-V alignment rules vs. non-RISC-V alignment rules? Also, byte vs. word-aligned? Thank you!

helpful! | 0

Adelson Chua 5 months ago
Can you provide some context? What lecture/exam are you looking at?
good comment | 0

Anonymous Beaker 2 5 months ago
Spring 2021 Q6
helpful! | 0

Anonymous Beaker 2 5 months ago
Akshat mentions how we round the size to 12 if the amount of bytes in the struct hypothetically would be 11
helpful! | 0

Adelson Chua 5 months ago
RISC-V typically follows word-aligned memory (every 4 bytes).
If it is byte-aligned, the answer will be 11 bytes. That is, all memory allocations are rounded up to the nearest byte.
If something is word-aligned, memory allocations would be rounded to the nearest word.
That's all there is to it.
good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f30**

Philippe Hansen-Estruch 5 months ago
How exactly would Fall 2021 7(a).c be representable with R type? I'm not following how this would work if you are not using the branch comparator
helpful! | 0

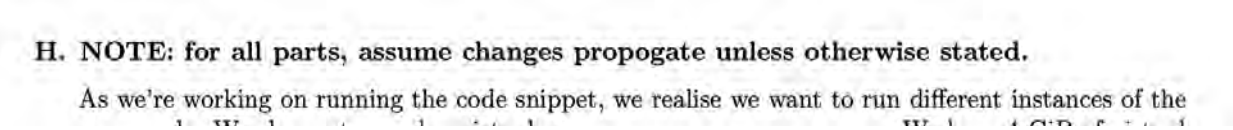
Peyrin Kao 5 months ago
(this is SP21-7.a.i.c)

The question says to assume earlier changes propagate, so I think you're still using the branch comparator here. Hardwiring rs2 to be x0 lets you pass rs1 and zero into the branch comparator.
good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f31**

Anonymous Poet 2 5 months ago
[sp21-final-2]


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

As we're working on running the code snippet, we realise we want to run different instances of the

same code. We choose to employ virtual memory on our memory space. We have 4 GiB of virtual memory and 16 MiB of physical memory mapped with a single level page table with a page size of 4 KiB. We choose to store 8 bits of metadata with each page table entry.

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

1024

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

in the solution video, they did 4 bytes * 2^20 / 2^12, where 4 bytes is the PTE entry size, 2^20 is the number of pages, and 2^12 is page size. This results in 2^10. But, why can't the answer just be 2^20, which is the number of pages that you get when you do 2^32 / 2^12? Isn't the question asking how many pages will be taken up?

helpful! | 0

Adelson Chua 5 months ago

It's asking for the number of pages **the page table** is taking up.

good comment | 0

Anonymous Poet 2 5 months ago

Then what kind of pages does 2^20 represent? Also, what's the diff b/w a page and PTE. Are they the same?

helpful! | 0

Anonymous Poet 2 5 months ago

Oh I think I understand a bit now. The 2^20 represents how many pages there are in the virtual address space, but the question is asking for how much space the PT will take up. So that's why we have to multiply the number of VPN possibilities by how many bytes will be in each corresponding PPN's PTE, and then divide by page size. Is this correct?

helpful! | 0

Adelson Chua 5 months ago

Think of the page table as a matrix.

The number of rows is 2^VPN = 2^20. You are indexing the matrix using the VPN bits.

The size of the column is the PTE, that's the entry of the page table when you access it using the VPN.

The PTE contains the PPN. That's why a VPN will have a PPN translation (you access the row using VPN, you get the PTE containing the PPN).

That matrix (page table) is divided into chunks called pages, think of pages as the unit element of the main memory. Just like how caches have blocks.

Then there is also a concept of data pages, which is the actual data the program is using. Because remember, page tables are just translations, there no data in there.

Each PPN contained in the PTE is a pointer to the data pages.

Makes sense?

good comment | 0

Anonymous Poet 2 5 months ago

Using the matrix analogy, the rows are the pages corresponding to the VPN, and the columns represent the PTE's containing the PPN, which point to the physical location of the data in memory/disc. Also, was my reasoning for how to get to # of pages of the PT above correct?

helpful! | 0

Adelson Chua 5 months ago

Wait, pages are a group of rows. It depends on the page size, which is given in the problem.

Yes, getting the number of pages from the page table size is correct. It's basically just page table size / page size. Page table size = 2^VPN * PTE size.

good comment | 0

Anonymous Poet 2 5 months ago

I thought pages would be the rows, and then u index into the column using the offset for the PPN. Is it not like that?

helpful! | 0

Adelson Chua 5 months ago

No...

There's no terminology regarding the rows.

One row only contains one PTE. 1 PTE is not equal to the page size.

I shouldn't have said the cache:block::mainmemory:pages analogy, you're getting confused.

good comment | 0

Anonymous Poet 2 5 months ago

Hmm may you explain how pages are stored in a page table? Afaik, with coarse-grained mapping, the entries in the page table store more mappings per PTE. Like instead of storing an individual word, it can store addresses 0-4095 (4 KiB). So doesn't that mean that 1 PTE is equal to the page size (4 KiB)?

helpful! | 0

Anonymous Poet 2 5 months ago

oh wait, a page is loaded into the page table at a time i think. so each entry in the page table is a mapping to the PPN, and the other pages are still in memory elsewhere.

helpful! | 0

Anonymous Poet 2 5 months ago

nvm, the page table has mappings to PPN, and then if there's a hit, it'll go to memory and read from that PPN and then use the offset to get the data. the PPN represent data stored in pages in memory.

helpful! | 0

Adelson Chua 5 months ago

Yes, PPN which are contained in PTE are just pointers.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f32**

Anonymous Calc 2 5 months ago

> Blank 7:
>
> `(output[0] == 0) && (output[1] == 0) && (output[2] == 0) && (output[3] == 0)`
>
> Other solutions may exist. Note that the solution:
>
> `output[0]+output[1]+output[2]+output[3] == 0` is incorrect, since we could have received outputs that happened to add to 0, even if they aren't all 0. As an example consider the inputs $A = [0, 0, 0, 0]$, $B = [0, 0, 0, 0]$, $C = [1 << 30, 1 << 30, 1 << 30, 1 << 30]$.

https://inst.eecs.berkeley.edu/~cs61c/fa21/pdfs/exams/fa21-final-sols.pdf

Hi, this was the last question for the fall 21 exam linked above.
But, I was still confused on why output[0] + output[1] + ... is incorrect since it was what I did.
Because in the above solution, we are saying that output[0] == 0 ,...., output[3] == 0.
If all the outputs such as output[i] are == 0, doesn't that mean if we add a bunch of output[i's] they should also be equal to 0?

helpful! | 0

Adelson Chua 5 months ago

No. The explanation already gives you an example.
Say output[0] = 0xFFFFFFFF and output[1] = 0x00000001.
output[0] + output[1] = 0

good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f33**

Anonymous Gear 2 5 months ago

Summer 2021 Practice Test (PraireLearn)

Sequence 1

```
sw  s1, 0(a2)
add s0, s1, a0
```

Cycle Count 8

Sequence 2

```
lw a2, 4(t2)
```

```
sw a2, 4(t2)
```

Cycle Count 12

Sequence 3

```
sw a2, 0(t2)
lb a0, 1(t2)
```

Cycle Count 9

Sequence 4

```
lw a2, 4(a1)
sw a1, 4(a2)
```

Cycle Count 12

I'm a bit confused on why the number of cycles for sequences 2 and 4 are 12. I know that it would take 8 cycles for the first sw instructions to run, meaning that the load instructions following it would stay in the X stage for 3 cycles. The load instruction would then spend 2 cycles in the M stage and then finish off with 1 cycle in the W stage. I assumed that the load instruction would only take 8 cycles, ending in the 9th cycle.

helpful! | 0

Peyrin Kao 5 months ago
@2181
good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f34**

Philippe Hansen-Estruch 5 months ago
How do you do Fall 2021 1.12?

Q1.12 (1 point) We've devised an error-correcting code which is able to fix 1 bit errors. If 0x61C is a valid codeword, which of the following can NOT be a valid codeword, regardless of the error-correcting scheme we have? Select all that apply.

- ☐ 0x71C
- ☐ 0xC16
- ☐ 0x51C
- ☐ 0x16C
- ☐ 0x70D
- ☐ None of the above

helpful! | 0

Adelson Chua 5 months ago
@2128_f28
good comment | 0

Philippe Hansen-Estruch 5 months ago
The explanation given didn't make much sense, could you walkthrough one or two examples?
helpful! | 0

Adelson Chua 5 months ago

Let's use the example given in the solution:

Say we have a system that is noisy, a string of bits I send to you will have 1-bit error somewhere along the string. You, as the receiver, just have to identify specifically where the error is and determine what codeword was originally sent.

Say in all the following scenarios, I will be sending 0x61C, then it will get corrupted by 1-bit error becoming 0x71C/0x51C. This is what you will receive.

Scenario 1: We agree that I will either send **0x61C or 0xC16** (these are our only codewords)
I sent 0x61C.... You received 0x71C... Well, you know that it is either 0x61C or 0xC16 since that is what we agreed upon. You determined that the 0x71C that you received was originally 0x61C. That was my original message. I'm happy.

Scenario 2: We agree that I will either send **0x61C or 0x16C** (these are our only codewords)
I sent 0x61C.... You received 0x71C... Well, you know that it is either 0x61C or 0x16C since that is what we agreed upon. You determined that the 0x71C that you received was originally 0x61C. That was my original message. I'm happy.

Scenario 3:
We agree that I will either send **0x61C or 0x70D** (these are our only codewords)
I sent 0x61C.... You received 0x71C... Well, you know that it is either 0x61C or 0x70D since that is what we agreed upon. 0x71C has a 1-bit difference relative to 0x61C. 0x71C has 2-bit difference relative to 0x70D. 0x61C is 'closer' to the codeword 0x61C. Thus, you determine that I originally sent 0x61C. That was my original message. I'm happy.

Scenario 4: We agree that I will either send **0x61C or 0x71C** (these are our only codewords)
I sent 0x61C.... You received 0x71C... We'll 0x71C was part of the code we agreed upon. Thus, you would think that I sent 0x71C. But I originally sent 0x61C. You failed at decoding my message. I am disappointed.

Scenario 5: We agree that I will either send **0x61C or 0x51C** (these are our only codewords)
I sent 0x61C.... You received 0x51C (also 1-bit difference to 0x61C)... We'll 0x51C was part of the code we agreed upon. Thus, you would think that I sent 0x51C. But I originally sent 0x61C. You failed at decoding my message. I am disappointed.

See now?

Bottomline: The 'farther away' the codewords we have are, the easier it is to determine what was the original message even with bit corruption.
'Farther away' = the more bit differences, the better.

good comment | 4

Philippe Hansen-Estruch 5 months ago

Yes, this was amazing thank you!

helpful! | 0

Anonymous Helix 3 5 months ago

How are 61C and 51C a 1 bit difference? Don't we need to flip two bits, since 0101 and 0110?

helpful! | 0

Anonymous Mouse 3 5 months ago

Since we can have 1 bit error, both 0x61c (0110) and 0x51c (0101) have the possibility of corrupting into 0x71c (0111) via changing 1 bit. In this case, we will not be able to correct the error if we receive 0x71c. Since we know 0x61c is a valid codeword, 0x51c cannot be valid.

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f35**

Anonymous Comp 5 months ago

SU21-Final-Q3: Why is malloc not valid for some of these questions? for example, we allocate memory for new_node->label and then immediately copy the value from source_labels[i] into it for part a blanks 4 and 5. Why do we need to use calloc to allocate the memory instead of malloc?

helpful! | 0

Adelson Chua 5 months ago

Disclaimer: We haven't checked the correctness of the solutions before making this available.

There might be some statement in the problem saying that it should be initialized to zero?
If not, then, yes, I agree. Using malloc should also be fine.

In any case, this will be explicitly stated in the exam problem if ever we include something like this.

good comment | 0

Fehmi Arda Akman 5 months ago

I thought about this issue a lot, and I think it is because when you calloc, the initial data consists of 0's. This means that if we calloc, the node we generated callocing will have the field 'children' to be 0, meaning that the new node does not have any children by the use of calloc (and, new nodes should not have any children so voila). It is a tricky bit to see that, but I think this might be the reason.

~ An instructor (Adelson Chua) thinks this is a good comment ~

helpful! | 2

Reply to this followup discussion

Resolved Unresolved **@2128_f36**

Anonymous Calc 2 5 months ago

For the rest of the problem, consider the following C code:

```
#define CHUNKSIZE (8)
void interesting_function(uint16_t *arr, int n) {
  for (size_t i = 0; i < (n / 2) / CHUNKSIZE; i++) {
    uint16_t sum = 0;
    for (size_t j = 0; j < CHUNKSIZE / 2; j++) {
      sum += arr[(i * CHUNKSIZE / 2) + j];
```

```
    }
    arr[n / 2 + i * CHUNKSIZE / 2] += sum;
  }
}
```

You may make the following assumptions:

- `arr` is a cache-aligned address that points to an array of `n` 16-bit unsigned integers
- `n` is very large (greater than 1 million), and is a multiple of the cache line size. It is NOT a multiple of the size of the whole cache.
- All local variables are stored in registers
- The cache is cold when the function is first called
- Memory addresses are 32 bits

PART B (10 pts.)

Consider running `interesting_function` on a system with a **direct-mapped L1 cache** that has a line size of 128 bytes, and the cache can hold 4 KiB of data.

i) How many bits are in the tag, index, and offset fields? (1 pt. each)

# of tag bits 20 # of index bits 5 # of offset bits 7

ii) How many memory accesses occur in one iteration of the outermost loop (the one where `i` is incremented)? (3 pts.)

Displayed correctness for this blank may be inaccurate, but total points on this question should be correct (see errata in Ed grades post).

6

iii) What is the L1 cache hit rate when this function is run? Leave your answer as a fraction or decimal, not a percentage. (4 pts.)

`0.979`

So, I'm kind of unsure of what the best way to calculate the hit rate here is and was unsure of how we got 0.979.

Also, in this question, how come if we increase the associativity to a 4 set associative , why does the hit rate stay the same?

helpful! | 0

Adelson Chua 5 months ago

@2186

Follow the logic and apply your 8 chunks version to the analysis provided.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f37**

Anonymous Gear 5 months ago

I wanted to know about 10aiv, shouldn't the first instruction be lb t0 1(a0)? If t0 is equal to null, then it will still add 1 to a0 right after.

iv. `1: endofstring:`

Inputs: `a0` is a pointer to a **nonempty** string

Output: `a0` returns a pointer immediately after the end of the string.

Example: Let a0 be the pointer 0x10000000 to string s, which is the string "Hello". Then the expected output would be 0x10000006.

```
endofstring:
___t0_____ #Code line 4. You must use t0 as the first register of this instruction.
__________ #Code line 5
__________ #Code line 6
ret
```

You **may not** use strlen or any other library function

Code line 4:

```
lb t0 0(a0)
```

v. Code line 5:

```
addi a0 a0 1
```

vi. Code line 6:

```
bneq t0 x0 endofstring OR bneq x0 t0 endofstring OR bnez t0 endofstring
```

helpful! | 0

Adelson Chua 5 months ago

But the problem states that a0 points to a **non-empty string**. So it is at least 1 character long with the byte after it being the NULL terminator.... and the problem wants the address of that NULL terminator.

good comment | 0

Anonymous Gear 5 months ago

But when it loads 0(a0) in the first instruction, and it is null, then a0 is currently pointing to the address of the null terminator. Adding a0 and 1 after that will be the index after the null right?

helpful! | 0

Adelson Chua 5 months ago

But again the problem states that it cannot start with a NULL.

"But the problem states that a0 points to a **non-empty string**. "

good comment | 0

Anonymous Gear 5 months ago

I know, but I am talking about when it reaches the end of the string "HELLO." When a0 points to the null terminator, the code will still add a0 and 1 instead of returning a0?

helpful! | 0

Adelson Chua 5 months ago

I had to recheck the problem statement again. It actually asks you to point after the NULL terminator.

Check the example: "Hello" starting at address 0x0. H = 0x0, e = 0x1, l = 0x2, l = 0x3, o = 0x4,

NULL = 0x5, after the string = 0x6 (this the what the problem wants you to return)

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f38

Anika Kartik 5 months ago

SP21-Final-Q7C

I am confused about why the second answer doesn't work. Isn't it saying the same thing - passing x0 in for rs2?

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

Format: `R-Type` Instruction

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

helpful! | 0

Caroline Liu 5 months ago

The main difference between the two is whether or not we can have `is_null rd, rs1` be considered a true instructions that just uses the R-format or if it has to be translated into `is_null rd, rs1, rs2` . By hardcoding it, we don't have to add `rs2` explicitly as the user to get a comparison.

Does that answer your question?

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f39

Anonymous Comp 5 months ago

SU21-final-1.11

For the rest of the problem, consider the following C code:

```
#define CHUNKSIZE (8)
```

```
void interesting_function(uint16_t *arr, int n) {
  for (size_t i = 0; i < (n / 2) / CHUNKSIZE; i++) {
    uint16_t sum = 0;
    for (size_t j = 0; j < CHUNKSIZE / 2; j++) {
      sum += arr[(i * CHUNKSIZE / 2) + j];
    }
    arr[n / 2 + i * CHUNKSIZE / 2] += sum;
  }
}
```

iii) What is the L1 cache hit rate when this function is run? Leave your answer as a fraction or decimal, not a percentage. (4 pts.)

number (3 significant figures)

How do we solve this? for the first outer loop iteration, I believe we hit 5 out of 6 times due to a compulsory miss. However, I don't know how to go further.

helpful! | 0

Adelson Chua 5 months ago

@2186

Just follow the logic and use chunk size of 8 instead.

good comment | 0

Harrison 5 months ago

I'm a bit confused about this.... does this mean that the inner loop runs 8/2 = 4 times?

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f40**

Mallika Parulekar 5 months ago

E.

```
1 #define base_arr_addr 0x12345678
2 #define ARR_SIZE 4096
3 #define I_BOUNDARY 2048
4 #define J_BOUNDARY 4096
5 #define I_STRIDE 128
6 #define J_STRIDE 64
7 int arr[ARR_SIZE];
8
9 uint32_t dummy_func(void) {
10   //Loop 1
11   for (int i = 0; i < I_BOUNDARY; i += I_STRIDE) {
12     for (int j = 0; j < J_BOUNDARY; j += J_STRIDE) {
13       arr[i] = arr[i] + arr[j];
14     }
15   }
16
17   //Loop 3
18   for (int j = 0; j < J_BOUNDARY; j += J_STRIDE) {
19     for (int i = 0; i < I_BOUNDARY; i += I_STRIDE) {
20       arr[j] = arr[j] * arr[i];
21     }
22   }
23 }
```

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

47/48

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

helpful! | 0

Mallika Parulekar 5 months ago

For this question Q2 D- F I'm confused as to how we know that the cache will have enough room for everything we load into it. I get that there are 64 different lines that need to be in the cache, but how do we know that each of these 64 lines for example won't map to the same set? Thanks!

helpful! | 0

Adelson Chua 5 months ago

The addresses are consecutive. You will not map to the same set that way.

If you want to convince yourself, write out a list of increasing addresses and divide it into T/I/O segments. You'll see there that as long as the cache is big enough to contain the array, there will be no conflicts.

good comment | 0

Mallika Parulekar 5 months ago

Ah got it that makes sense - thank you!!

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f41**

Anonymous Gear 2 5 months ago

(this isn't necessarily a question I have about a certain exam problem, but rather about the information needed to solve Calling convention problems).

Are the number values next to the registers on the RISC V sheet (ra, t registers, s registers, etc) in decimal or hexadecimal form. I never quite figured out which type of value did it hold.

helpful! | 0

Adelson Chua 5 months ago

Decimal

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f42

Anonymous Gear 2 5 months ago

Summer 2021 Practice Exam PraireLearn:

How exactly is the largest odd number calculated? I had assumed to just turn all of the bits into 1s (except the last bit = 0) in the Exponent bits. From there I would simply subtract by 1 to get my largest possible odd number, but I'm confused as to how the largest possible odd number is 255 if the the exponent component only has 5 bits.

For this entire section, you are not allowed to use the C programming language as a calculator, an actual calculator, a floating point simulator, nor any other computational aid.

Consider a 13-bit floating-point number with the following components (1 sign bit, 5 exponent bits, 7 mantissa bits); that is,

SEEEEEMMMMMMM

All other properties of IEEE-754 apply (bias, denormalized numbers, ∞, NaNs, etc). The bias is the usual value of $-(2^{E-1}-1)$, which here would be $-(2^4-1)$ = -15.

PART A (4 pts.)

Under this scheme, what is the floating-point representation **(in hex)** of the fraction **-11/4**? **Do NOT include the 0x prefix when writing your answer.** Answers given in binary will not be given partial credit.

Displayed correctness for this blank may be inaccurate, but total points on this question should be correct (see errata in Ed grades post).

0x `1830` ✓ 100%

PART B (6 pts.)

What is the largest odd integer that we can represent under this scheme? Write your answer as a base 10 integer.

15 ✗ 0%

helpful! | 0

Adelson Chua 5 months ago

You can follow the logic here: @44_f3

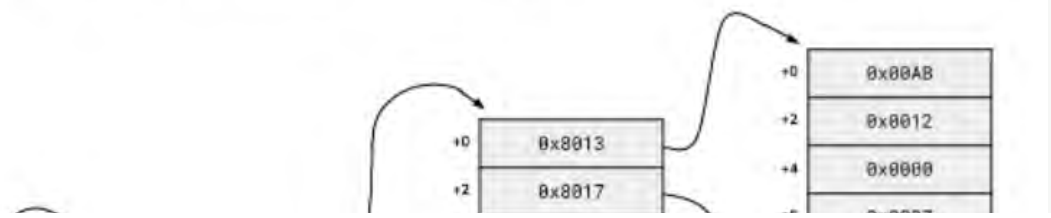
good comment | 0

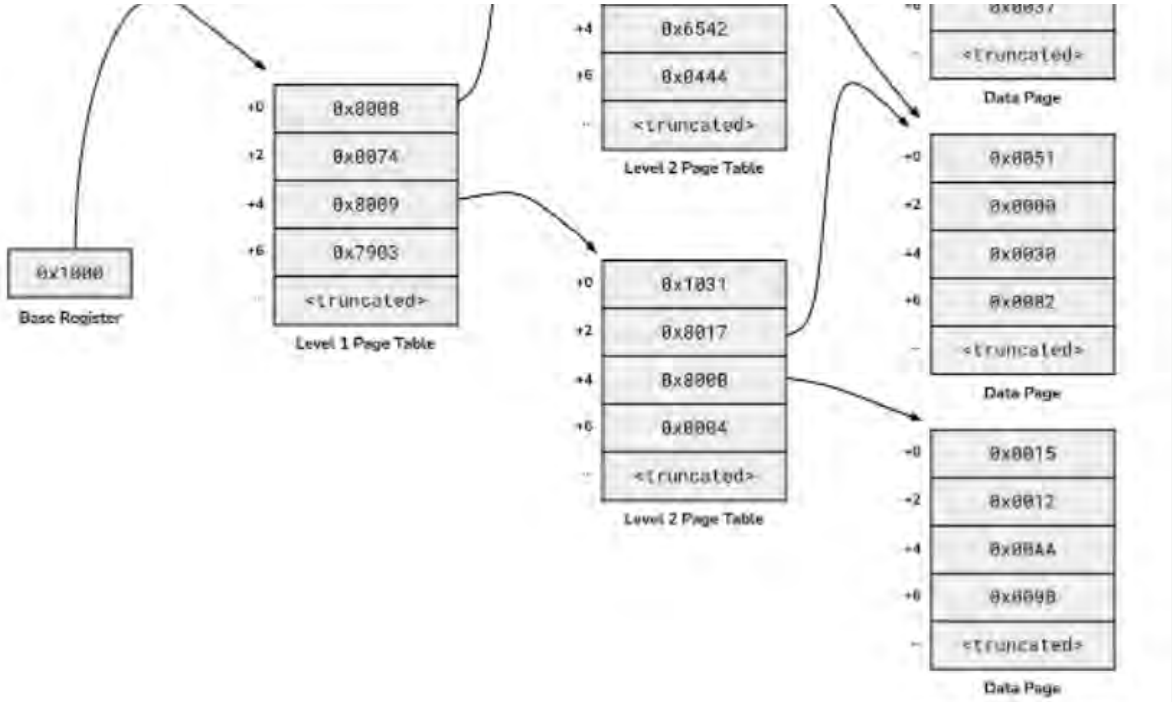
Reply to this followup discussion

Resolved Unresolved @2128_f43

Anonymous Comp 5 months ago

Now, suppose physical memory and the page table have this initial state. Each arrow indicates the location of either the next-level page table or the physical data page being pointed to.





PART B (3 pts.)

What physical address does virtual address `0x0000041A` map to? Write your answer in hexadecimal, **without the leading `0x`**, or write `N/A` if this does not map to a valid physical address.

Displayed correctness for this blank may be inaccurate, but total points on this question should be correct (see errata in Ed grades post).

0x `9C1A`

I am not sure how to solve this problem. This is my work so far:

I know from the previous part that the offset is the last 11 bits are the offset, which means that all the bits for VPN1 and VPN2 are 0 from 0x0000041A. I see the index 0 for page table 1 is 0x8008, which means there is a valid first bit. Then I check index 0 of the 2nd page table, where I see 0x8013. I see that there is 0x00AB in the Data Page Table, but then I'm not sure what to do. I tried adding the 0x00AB bits in front of the offset bits, but that doesn't get me the right answer. I'm probably doing something very wrong, could I have some guidance?

helpful! | 0

Adelson Chua 5 months ago

@2187

This is basically the homework. Just with added stuff that is not really relevant to the problem.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f44**

Anonymous Gear 2 5 months ago

Summer 2021 Practice Exam Prairie Learn

Correct answer

PART A (3 pts.)

Consider a memory system with an L1 and L2 cache before main memory. Calculate the AMAT (in ns) for a particular workload with the following parameters:

- L1 cache local hit rate: 40%
- L1 cache hit time: 2 ns
- L2 cache local hit rate: 90%
- L2 cache hit time: 10 ns
- Main memory access time: 100 ns

`14.0` ns

For the rest of the problem, consider the following C code:

```
#define CHUNKSIZE (8)
void interesting_function(uint16_t *arr, int n) {
  for (size_t i = 0; i < (n / 2) / CHUNKSIZE; i++) {
    uint16_t sum = 0;
    for (size_t j = 0; j < CHUNKSIZE / 2; j++) {
      sum += arr[(i * CHUNKSIZE / 2) + j];
    }
    arr[n / 2 + i * CHUNKSIZE / 2] += sum;
  }
}
```

I'm having trouble figuring out how to find the AMAT for this problem. From the review slides, I was able to determine that AMAT = hit_time + miss_rate(miss_penalty). I was unsure of how to use this formula since we're given 2 hit times due to it being a 2-level cache. We're not given a miss rate, but I know I can find the local miss rates by taking the inverse of the local hit rates given. I'm just not understanding how to exactly solve this: do I just add the local hit times together? Do I multiply the local miss rates? How exactly do I find the miss penalty?

helpful! | 0

Adelson Chua 5 months ago

Lecture 17. Slide 67.

Miss penalty is the main memory access time.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f45**

Selena Wong 5 months ago

sp21-final-q4b

(b) Select the different ways you can decrease the critical path (ignore wire delay)?

- ☐ Make the clock speed faster
- ☐ Move components closer together
- ■ Simplify circuit (boolean) logic
- ■ Add pipeline registers

☐ Use transistors instead of AND, OR, and NOT gates

☐ Use AND and OR gates instead of NAND and NOR gates

In regards to this question, what does clock speed refer to & it is different from clk-t-q delay, correct? Additionally, will add transistors usually always increase the critical path, or is it possible that it can sometimes decrease critical path? Lastly, are there other ways to decrease critical path that are not listed in these answer choices?

helpful! | 0

Adelson Chua 5 months ago

Clock speed = clock frequency.

Decreasing the number of logic gates decreases the delay. Adding more transistors along the path of the critical delay will increase it.

There are other techniques to decrease the critical path, mainly in terms of *transistor sizing*, but this is covered in 151/251 and is beyond the scope of 61C.

good comment | 1

Selena Wong 5 months ago

Ah, this makes sense; thank you!

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f46**

Selena Wong 5 months ago

sp21-final-q1bii

ii. **Q2B**

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

100

I tried looking at the equation provided in lecture, but it seems like there are slight differences that I'm not understanding. How is the final answer of 100 reached?

helpful! | 0

Adelson Chua 5 months ago

@2128_f15

good comment | 1

Selena Wong 5 months ago

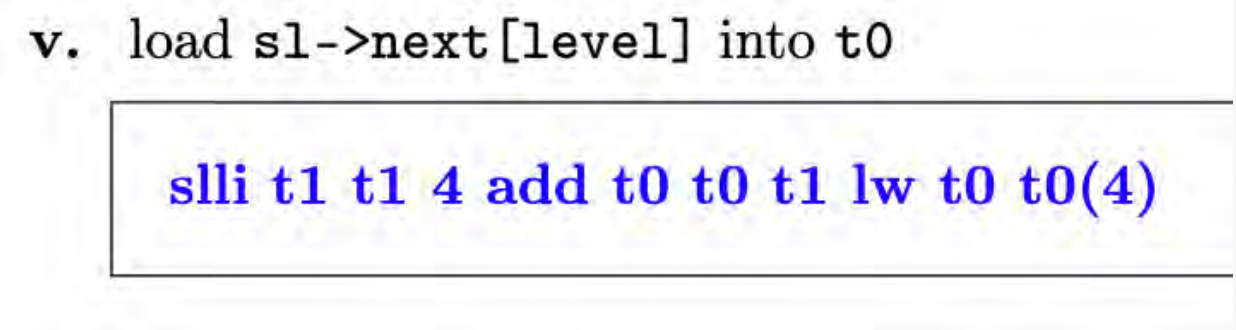
Just noticed it was above as well -- sorry for the duplicate; thank you!

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f47**

Anonymous Atom 3 5 months ago



Sp21-Final 6.b.v

For this problem, what is the purpose for the first 2 lines. I know that the first line is for finding bytes within our array and the second line is for finding the absolute address but why are those steps needed?

helpful! | 0

Peyrin Kao 5 months ago

The previous subpart put `s1->next` in t0, which is of type `struct SLN **`. Remember that arrays in C are pointers to their first element, so this is a pointer to the first element of an array of `struct SLN *` pointers. To access the `level` 'th element of the array, we need to calculate the offset from the start of the array, add that to the address of the start of the array to get the address of the desired element, then dereference the address to get the desired element.

good comment | 0

Anonymous Mouse 4 5 months ago

the answer key for this question is wrong

line 1 should be slli t1 t1 2

line 2 should be lw t0 0(t0)

this is the answer from the video walkthrough

helpful! | 0

Anonymous Mouse 4 5 months ago

*line 3

helpful! | 0

Anonymous Gear 3 5 months ago

Which one is correct? The video or the answer key?

helpful! | 0

Adelson Chua 5 months ago

Video.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f48**

Anonymous Mouse 3 5 months ago

SP21 Final Q2a K)

K. We now switch our system to having a 2-level page table instead. Assuming we restart our system and run one iteration of the inner loop, and that the VPN bits are split evenly between levels, how many pages will our active page tables span?

2

No alternate solution.

I am really confused about how to approach this problem, especially on how "running one iteration of the inner loop" matters. Could someone give me a brief explanation for this question? Thank you!

helpful! | 0

Adelson Chua 5 months ago

Restart our system = page table is clean, no other active pages.

1 iteration of the inner loop, so i = j = 0. All array accesses are at index 0 -> you are just accessing as a single memory address.

2 level page table = You need at least 2 page tables (one for L1 and one for L2) to get a translation.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f49

Anonymous Helix 3 5 months ago

i. Consider a 20-bit floating-point number with the following components (1 sign bit, 6 exponent bits, 13 mantissa bits); i.e.,

SEEEEEEMMMMMMMMMMMMM

All other properties of IEEE754 apply (bias, denormalized numbers, , NaNs, etc). The bias is the usual –(2E–1–1), which here would be –(25–1) = -31.

What is the bit representation (in hex) of the floating-point number -8.25? Your answer must be in hex, must be prepended with 0x, and all letters must be capitalized. Do not include any extra leading zeros.

0xC4100

8.25 = 1000.012 = 1.000012 * 23

Just to confirm, this is incorrect? Because 8.25 is 1000.01?

helpful! | 0

Peyrin Kao 5 months ago

Yeah, there can't be a 2 in a binary number, so the work here looks like a typo. Would need to write out the correct calculation to see whether the final answer is correct.

good comment | 0

Justin Yokota 5 months ago

I believe those 2s were meant as a subscript 2 (so a signifier of the base). The intended values are 1000.01_2 and 1.00001_2 * 2^3, and somehow, all the subscripts and superscripts got lost in

writing.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f50

Saikrishna Achalla 5 months ago

Fa21-MT2-Q4.3

Q4 ***To Float or Not to Float*** **(20 points)**

Consider a floating point system that has 16 bits with 7 bits of exponent and an exponent bias of -63, which otherwise follows all conventions of IEEE-754 floating point numbers (including denorms, NaNs, etc.). In this question, we will compare this system to an unsigned 16-bit integer system.

Q4.1 (4 points) What is the value of floating point number `0xC2C0` in decimal?

Q4.2 (1 point) Which representation has more representable numbers? Count +0, -0, $+\infty$, and $-\infty$ as 4 different representable numbers.

- ◯ The floating point number
- ◯ The unsigned 16-bit integer
- ◯ Both systems can represent the same number of values

Q4.3 (3 points) How many more numbers can be represented? Write 0 if both systems can represent the same number of values.

Q4.3 (3 points) How many more numbers can be represented? Write 0 if both systems can represent the same number of values.

Solution: 510

From the previous part, we know that the only difference in representable numbers comes from NaNs. Thus the question is asking how many NaNs exist in the floating point system.

Recall that NaNs are represented by all ones in the exponent, any value in the sign bit, and any non-zero value in the significand (a zero in the significand would represent infinity).

The significand is 8 bits, so there are $2^8 - 1$ possible non-zero values in the significand. There are 2 possible bits in the sign bit. In total, there are $2 \times (2^8 - 1) = 510$ NaNs.

Thus, the unsigned integer can store 510 more unique numbers.

Grading: Half credit was awarded for forgetting the infinity (512), and forgetting negative NaNs (255), but not both.

Working on part 4.3 of this question. I understand that the difference between the two representations is the NaN and infinity values. However, I do not understand why they ignore the infinity values that can be represented by the floating point but not by the 16-bit unsigned representation. The -1 in the calculation suggests that, for each sign value, the infinity is ignored but that is a value that cannot be represented by

the unsigned 16-bit rep right?

helpful! | 0

Peyrin Kao 5 months ago

Infinity is a number, so it's counted in the number of values that the floating-point system can represent.

good comment | 0

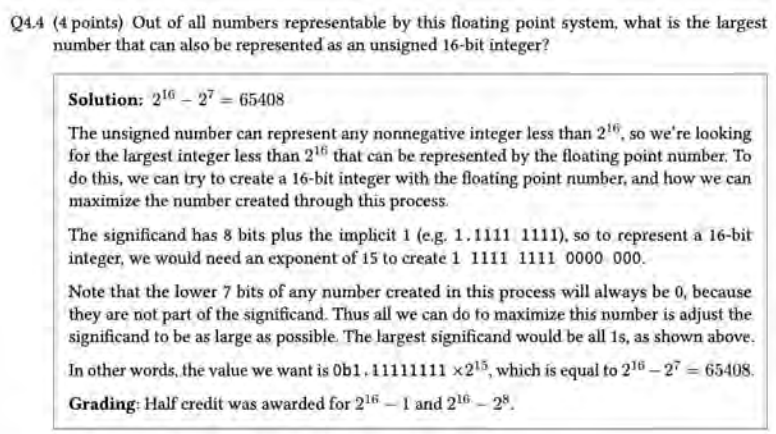
Reply to this followup discussion

Resolved Unresolved **@2128_f51**

Anonymous Scale 3 5 months ago

Fa21-MT2-Q4.4

I'm not really sure how to go about this question. The solutions are a bit confusing. Not sure how they got the exponent of 15 to create the number given and why are the bottom 7 bits 0?



Q4.4 (4 points) Out of all numbers representable by this floating point system, what is the largest number that can also be represented as an unsigned 16-bit integer?

Solution: $2^{16} - 2^7 = 65408$

The unsigned number can represent any nonnegative integer less than 2^{16}, so we're looking for the largest integer less than 2^{16} that can be represented by the floating point number. To do this, we can try to create a 16-bit integer with the floating point number, and how we can maximize the number created through this process.

The significand has 8 bits plus the implicit 1 (e.g. 1.1111 1111), so to represent a 16-bit integer, we would need an exponent of 15 to create 1 1111 1111 0000 000.

Note that the lower 7 bits of any number created in this process will always be 0, because they are not part of the significand. Thus all we can do to maximize this number is adjust the significand to be as large as possible. The largest significand would be all 1s, as shown above.

In other words, the value we want is $0b1.11111111 \times 2^{15}$, which is equal to $2^{16} - 2^7 = 65408$.

Grading: Half credit was awarded for $2^{16} - 1$ and $2^{16} - 2^8$.

helpful! | 0

Peyrin Kao 5 months ago

Given the 8-bit significand, we have to move the binary point over 15 places to create a 16-bit integer. The significand is only 8 bits, so you can't change the bottom 8 bits - they're always zeros inserted by moving the binary point over.

good comment | 1

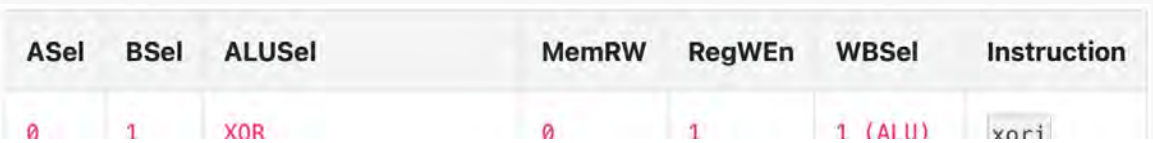
Reply to this followup discussion

Resolved Unresolved **@2128_f52**

Anonymous Beaker 3 5 months ago

Su21-Final-PF1.6

I was particularly confused about why beq and auipc were chosen for their respective control logic values. For beq, why do we ignore WBSel? I thought branch instructions would need to update the PC. For auipc, the ASel value is 1, which I believe means we are choosing to use data coming from Rs1 in our calculations. Why isn't PC (hence ASel = 0) used for auipc?



ASel	BSel	ALUSel	MemRW	RegWEn	WBSel	Instruction
0	1	XOR	0	1	1 (ALU)	xori

0	0	ShiftRightArithmetic	0	1	1 (ALU)	sra
0	1	Add	0	1	0 (MEM)	lb
1	1	Add	0	0	*	beq
1	1	Add	0	1	1 (ALU)	auipc
0	1	Add	1	0	*	sb
0	1	Add	0	1	2 (PC+4)	jalr
1	1	Add	0	1	2 (PC+4)	jal

helpful! | 0

Adelson Chua 5 months ago

For beq, why do we ignore WBSel? I thought branch instructions would need to update the PC.

WBSel is intended for the register file. Please check the digram provided.

Why isn't PC (hence ASel = 0) used for auipc?

Please check the provided diagram in the problem.

good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f53**

Anonymous Atom 5 months ago

[su21-final-q2b] How to get part b?

> **PART B (6 pts.)**
>
> What is the largest odd integer that we can represent under this scheme? Write your answer as a base 10 integer.
>
> 255

helpful! | 0

Adelson Chua 5 months ago

This might help. @44_f3

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f54**

Anonymous Mouse 5 months ago

Just curious, how can we find out the MTTR given the up time and down time?

helpful! | 0

Adelson Chua 5 months ago

I'm guessing during the down time you are repairing the system so that equals to MTTR?

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f55**

Anonymous Atom 5 months ago

[su21-final-c coding] I'm still not sure when to use malloc and when to use calloc in this question. I used malloc for all allocations and why is that incorrect?

helpful! | 0

Adelson Chua 5 months ago

@2128_f35

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f56**

Anonymous Mouse 5 months ago

Q7.4 (6 points) For each of the following memory accesses, determine if each access would be a hit or miss based on the cache state shown above, and if it's a miss, classify the possible miss type(s). If multiple miss types are possible depending on prior memory accesses, select all possible miss types.

Note: For this question, each memory access should be considered in isolation. In particular, do not update the cache state after each memory access.

Address				
0b 0011011011	☐ Hit	☐ Compulsory miss	☐ Capacity miss	☐ Conflict miss
0b 0011001101	☐ Hit	☐ Compulsory miss	☐ Capacity miss	☐ Conflict miss
0b 0110100010	☐ Hit	☐ Compulsory miss	☐ Capacity miss	☐ Conflict miss
0b 0010111100	☐ Hit	☐ Compulsory miss	☐ Capacity miss	☐ Conflict miss
0b 1110100010	☐ Hit	☐ Compulsory miss	☐ Capacity miss	☐ Conflict miss
0b 1111111111	☐ Hit	☐ Compulsory miss	☐ Capacity miss	☐ Conflict miss

Solution: The crux of this question is that data is always set to some garbage (there's no such thing as blank in binary, and we often don't zero out data if we can avoid it; as such, the cache generally ends up containing whatever data used to be there the last time a program ran), even if the cache block is empty. As such, the valid bit is needed to tell if a block is actually there. If the valid bit is 0, then the block is considered empty, regardless of the corresponding tag.

1: Hit; we notice that Tag 1 matches for our given index, and the valid bit is on.

2: Miss: Our tag isn't in our index. This can either be compulsory or conflict, since we don't

know if this block has been accessed before and ejected. This can't be a capacity miss, because there are still empty slots in our cache.

3: Miss; Our tag isn't in our index. This must be compulsory, because we still have an empty slot in this index. We only ever kick out a block when the cache is full, and only to replace that block with a new one; as such, we can't have kicked out a block while there is still empty space in the cache.

4: Miss; as with 3, this must be compulsory

5: Hit; this one hits the valid block in that index.

6: Miss; As noted above, we don't count this as a hit, since our valid bit is off.

For Fall 21 final 7.4 what is kind of miss for last one?

helpful! | 0

Adelson Chua 5 months ago

Compulsory since valid bit is off.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f57

Anonymous Atom 4 5 months ago

FA21-Final-Q2.3

We compute the following infinite sums under this floating point system, using left-association for addition (that is, a+b+c is evaluated in the order ((a+b)+c)), rounding after each addition. Eventually, this converges to some value, after which any further iterations don't change the sum. What is that value? You may express your answer either as an decimal value, or as an odd integer multiplied by a power of 2

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

Solution: We can look at how our mantissa changes:

0b1.0000

0b1.1000

0b1.1110

0b1.1111

0b1.11111 -> 0b10.000

Can someone explain this solution for me please? I'm thinking that the first line (0b1.0000) is 1, then the next line is 1 + 1/2, but isn't the next one after that 1 + 1/2 + 1/4 + 1/8? Also, I'm not sure what's happening on 5th and 6th lines.

helpful! | 0

Adelson Chua 5 months ago
It's getting rounded up. It is stated in the problem that the rounding process is rounding to the number with the least significant bit of 0. So, 1.1111 rounds to 10.0000 in binary.

good comment | 0

Anonymous Calc 2 5 months ago
This question is kind of confusing me because the reason we can't represent 33 with 4 mantissa bits is something like

I always thought that even though 1.00001 x 2^5 = 33. we can't do this because we have 4 mantissa bits so the last 1 gets dropped. so, like 1.00001 x 2^5 = 100001 but you drop the 1 at the end after calculating this because you only have 4 mantissa bits, so it just stays at 32.

Now, by that logic, whenever you add 1/2 + 1/4.. etc you get 1.1111...11. since we only have 4 mantissa bits here, the 1's after the 1.1111 will be dropped. So, like it would always get suck at 1.1111 was my logic.

but why does 1.11111 not just get rounded to 1.1111 and become 10.000 instead. thanks!

helpful! | 0

Anonymous Calc 4 5 months ago
after 1.111111, we will have 1.1111111, which rounds up to 10.000000? Which the sol is 10.000001?

helpful! | 0

Adelson Chua 5 months ago
It gets rounded up to the number with the least significant bit of 0. So you shouldn't get stuck at 1.1111 because the least significant bit of that is 1. If you add 1 more, you get 1.11111.... The nearest number with a least significant bit of 0 would be 1 more up, which is 10.000 as the solution shows. Then now that the least significant bit is now 0, any further additions are rounded down.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f58**

Anonymous Helix 4 5 months ago
sp21-final-10-a-iii

Why should we add 4 to ra? Why does ra represents the address of the current instruction? Isn't it the

return value of a function?

helpful! | 0

Adelson Chua 5 months ago

```
addi t0 x0 5
jal skipline
addi t0 t0 300
addi t0 t0 10
```

When **jal skipline** is called, **ra** will update such that it points to the **addi t0 t0 300** instruction.

The problem wants you to modify **ra** such that when **skipline** function returns, the **ra** will actually point to **addi t0 t0 10** (which is the the previous **ra +4**)

good comment | 1

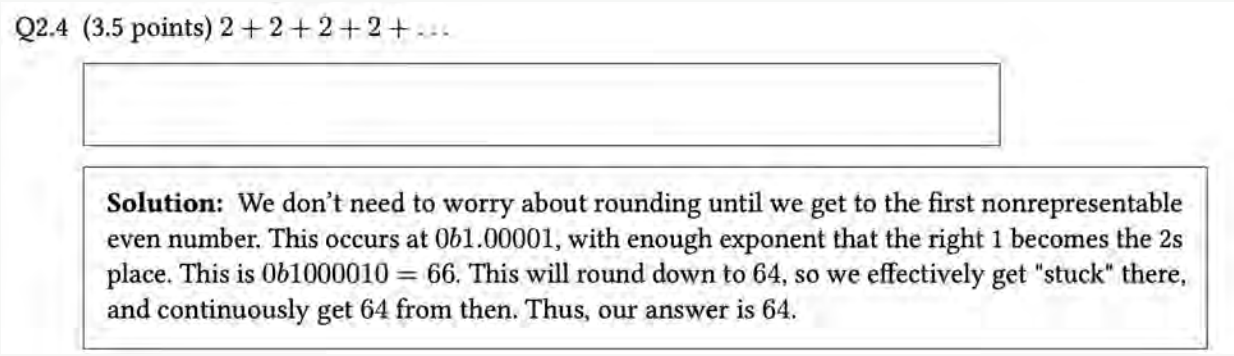
Reply to this followup discussion

Resolved Unresolved **@2128_f59**

Anonymous Comp 4 5 months ago

fa21-final-2.4

For this question, I'm confused why are trying to make the right 1 in the 2s place. I understand that 1.00001 is the first nonrepresentable number, since the max significand bits is 4.



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

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

helpful! | 0

Adelson Chua 5 months ago

The problem states the first nonrepresentable even number has the form 1.00001 (this is the point where you exceed the 4 bit significand)

What should the exponent be to make it the smallest even number that you can get?

1.00001 * 2^5 = 33, that's not even.

1.00001 * 2^6 = 66, that's even.

But the premise was that the form 1.00001 is not representable anymore, so 66 is not representable. So the even number before that is representable. That's 64.

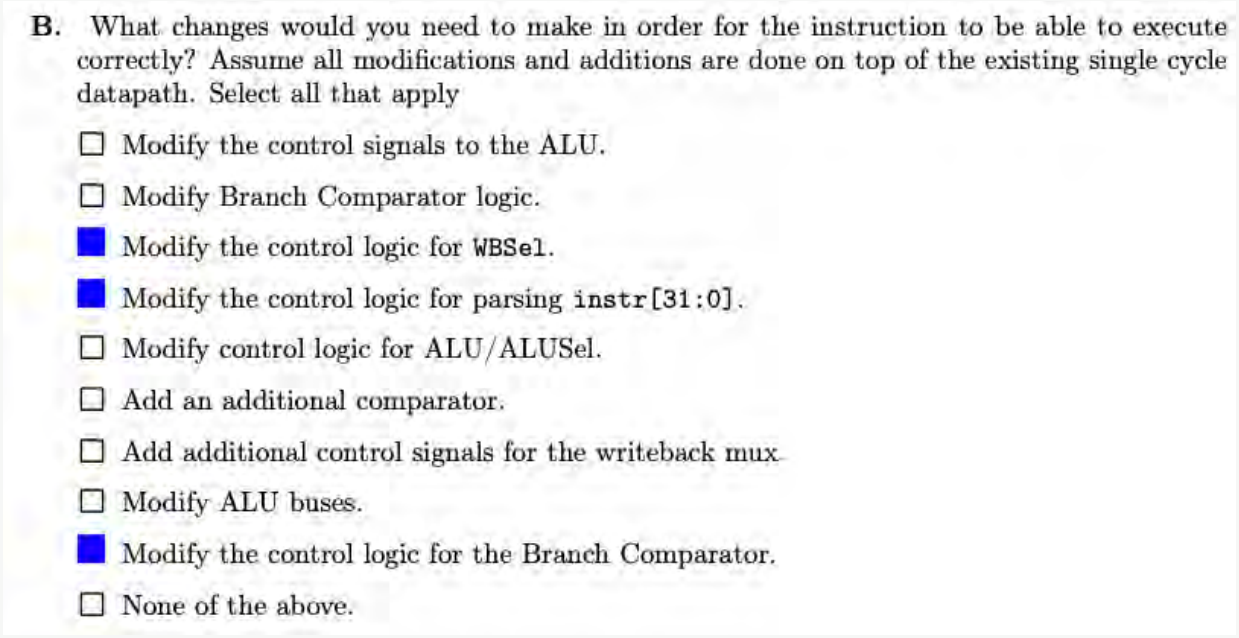
good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f60**

Anonymous Gear 4 5 months ago

sp21-final-q7b



Could a (u)GSI explain the logic behind solving this problem? What's the difference between "Branch Comparator logic" versus "control logic for the Branch Comparator", and what's the difference between control logic versus control signal?

helpful! | 0

Peyrin Kao 5 months ago

The branch comparator logic is the circuit in the branch comparator subcircuit of the datapath, as seen in lecture and project 3. The branch comparator control logic is the circuit in the control logic subcircuit of the datapath, as seen in lecture and project 3, which helps to determine the inputs to the branch comparator.

In this instruction, we need to read a value from register rs1 and compare it to 0x00000000. The solution seems to be achieving this by modifying the control logic signal so that the branch comparator receives the value in rs1 and 0x00000000, instead of the values in rs1 and rs2, when it sees an `is_null` instruction.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f61**

Anonymous Mouse 5 months ago

do lw have memory write back?

helpful! | 0

Peyrin Kao 5 months ago

Not sure if this is what you're asking, but in the datapath, the lw instruction requires reading from memory and writing back to a register. It might help to consult the reference card to see what the instruction is doing.

good comment | 0

Anonymous Mouse 5 months ago

how the reference card tells us we are reading from memory and back to a register, also in data hazards, are all lb, lh, lw and S type instruction must be done in WB before we start reuse in the next instruction? or we can use it in EX stage?

helpful! | 0

Anonymous Mouse 5 months ago

plus R type *

helpful! | 0

Adelson Chua 5 months ago

The reference card will contain the datapath diagram which will help you analyze this.

also in data hazards, are all lb, lh, lw and S type instruction must be done in WB before we start reuse in the next instruction? or we can use it in EX stage?

It depends if there is data forwarding in the pipeline. Else, you would have to stall. This is covered in the lecture I believe.

good comment | 0

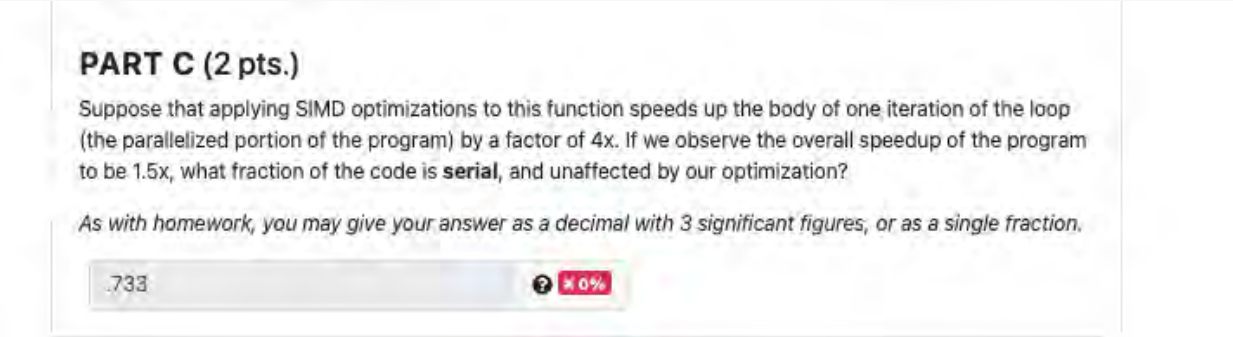
Reply to this followup discussion

Resolved Unresolved **@2128_f62**

Justin 5 months ago

SU21-Final-Q8

Can someone help me with this question, please? I am using Amdahl's law with S = 4 and speedup ratio = 1.25. I keep getting F = .266. I'm pretty sure F is the portion that is parallelizable.



helpful! | 0

Justin 5 months ago

The correct answer is .556.

helpful! | 0

Adelson Chua 5 months ago

speedup = 1/((1-p) + p/s)

speedup = 1.5

s = 4

p = unknown

1 equation, 1 unknown. You're looking for (1-p)
I'm getting 5/9 = 0.555..

good comment | 1

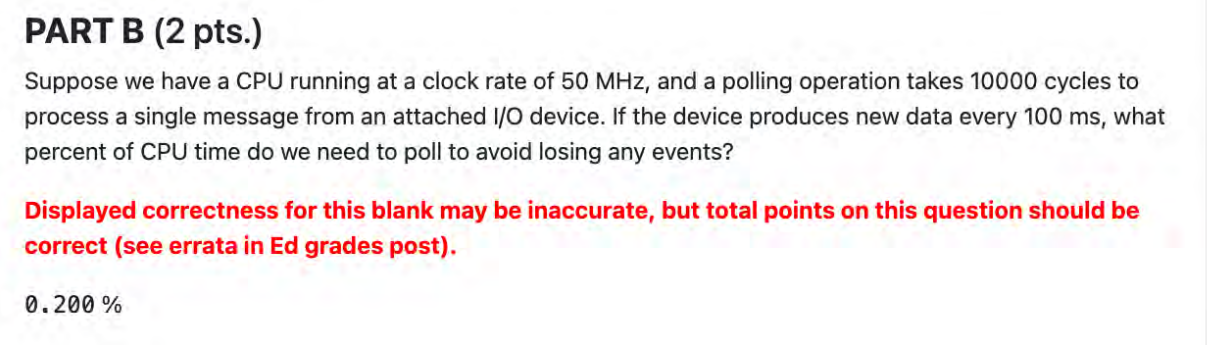
Justin 5 months ago
I was using the wrong value for speedup... 🤦 Thank you!
helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f63**

Anonymous Scale 4 5 months ago
I can't seem to get the right answer to this question.

My thinking is that we must poll twice every 100 ms, at the beginning and end, to see if data has come in. So with 2 polls every 100ms, we will have 20 polls a second. This leads to (20 polls * 10,000 cycles/poll) / 50Mhz, which is .002%. However, the answer is .2%.



helpful! | 0

Adelson Chua 5 months ago
Polling operation = 10,000 cycles
Clock rate = 50MHz = 50M cycles/second
Time between poll = 100ms

How many cycles for every poll check? 100ms * 50MHz = 100e-3 seconds * 50e6 cycles/seconds = 5e6 cycles
What percent of total cycles is used for polling? 10,000 cycles / 5e6 cycles = 0.002 => 0.2%
good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f64**

Anonymous Beaker 4 5 months ago
Sp21, why only A and D?

Which of the following code blocks would see a performance improvement if we placed a `#pragma omp for` over the outer for loop? Select all that apply.

i. A

```
for (int i = 0; i < 5000 - 3; i += 3) {
    a[i+2] = a[i] + a[i+1];
}
```

B

```
for (int i = 0; i < 5000 - 2; i += 2) {
    a[i+2] *= a[i];
}
```

C

```
for (int i = 0; i < 5000 - 3; i++) {
    a[i+2] = a[i] + a[i+1];
}
```

D

```
for (int i = 0; i < 5000; i++) {
    a[i] = 100;
}
```

- [x] A
- [ ] B
- [ ] C
- [x] D
- [ ] E. None of the above

helpful! | 0

Adelson Chua 5 months ago

B and C have data races present in them.

B increments i+=2, but the array accesses are a[i+2] and a[i] so there's an intersection of array accesses between two threads

C has the same problem incrementing i++ but array accesses go to a[i+2].

The right way is done in A, where i in incremented +3 but the array accesses go to a[i+2]. This avoids any data race.

D is intuitive, every iteration is independent of one another.

good comment | 1

Reply to this followup discussion

Resolved Unresolved **@2128_f65**

Anonymous Poet 4 5 months ago

Sp21Final, How do we solve question Q1.1?

Q1

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

speedup? Leave your answer as a single simplified fraction.

I'm not sure how to use the Amdahl's law here.

helpful! | 0

Peyrin Kao 5 months ago

Amdahl's law says:

$$\text{Speedup} = \frac{1}{(1-F)+\frac{F}{S}}$$

where F is fraction of the code that can be sped up, and S is maximum possible speedup.

F is what we want to solve for here (fraction of code that can be sequential = 1 - fraction of code that is parallelizable).

S is 8 (we have 8 threads, so we can get 8x speedup).

The desired total speedup is 4.

Plugging all this in:

$$4 = \frac{1}{(1-F)+\frac{F}{8}}$$

Solving gives $F = \frac{6}{7}$. If 6/7 of the code can be parallelized, then 1/7 of the code is sequential.

good comment | 1

Reply to this followup discussion

Resolved Unresolved @2128_f66

Anonymous Poet 4 5 months ago

For Sp21Final Q6. b.

Should these Risc-V commands have the index before the register?

Like Q6.b.i:
The answer is given lw a0 sp(0)
Shouldn't it be lw a0 0(sp)?

helpful! | 0

Adelson Chua 5 months ago

Typo

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f67

Rahul Iyer 5 months ago

Sp 21 Final 2A:

Why is this answer not 4000/4003?

helpful! | 1

Adelson Chua 5 months ago

I don't think I can search for the specific question you are asking. Be more specific or show the screenshot.

good comment | 0

Klaire Li 5 months ago

4000/4003 would be MTBF/(MTBF+MTTR) but the availability formula is MTTF/(MTTF+MTTR). MTBF = MTTF + MTTR. (I had the same issue bc I wrote the formula down wrong)

helpful! | 0

Adelson Chua 5 months ago

Ah okay, found it.

Klaire is correct.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f68

Emiliia Dyrenkova 5 months ago

Snippet 3

```
my_function:    mv   t2, a0
                srai a0, a0, 1
                addi sp, sp, -8
                sw   ra, 0(sp)
                sw   t2, 4(sp)
                jal  other_function
                lw   t2, 4(sp)
                lw   ra, 0(sp)
                addi sp, sp, 4
                mul  a0, t2, a0
                ret
```

Error Type	Affected Register
`Not saving/restoring callee-saved register`	`sp`

I don't remember ever seeing that we need to save sp as a callee-saved register. Is this right?

helpful! | 0

Adelson Chua 5 months ago

The issue is the fact that you did **addi sp sp -8**, but then put it back as **addi sp sp 4**.

You didn't restore it properly.

good comment | 1

Emiliia Dyrenkova 5 months ago

ohhh thanks.

helpful! | 0

Reply to this followup discussion

Resolved Unresolved @2128_f69

Anonymous Gear 2 5 months ago

PF1.1. Number Representation

For this entire section, you are not allowed to use the C programming language as a calculator, an actual calculator, nor any other computational aid.

PART A (2 pts.)

Convert 52_9 to base 6.

PART B (4 pts.)

Encode -51_{10} using **7** bits in each of the following representations. Write **N/A** if it is impossible.

Enter exactly 7 bits, and include any leading zeros if needed. **DO NOT include the leading `0b` prefix.**

I'm not exactly sure how I would do a change of basis. I know that they're different powers (i.e. powers of 9 and powers of 6), but I'm not quite sure how this can be done.

helpful! | 0

Adelson Chua 5 months ago

Convert to base 10 first since that's what you're familiar with.

Base 9 -> base 10 -> base 6

good comment | 0

Anonymous Gear 2 5 months ago

how exactly can we convert from base 9 to base 10 if they don't have any similar factors?

helpful! | 0

Adelson Chua 5 months ago

What do you mean 'similar factors'?

The logic is technically the same, when you're converting binary to decimal.

Every digit is of increasing powers of the base (in this case, 9). So from the right, the first digit is multiplied with 9^0, then the next one on the left is multiplied by 9^1.. and so on.

good comment | 0

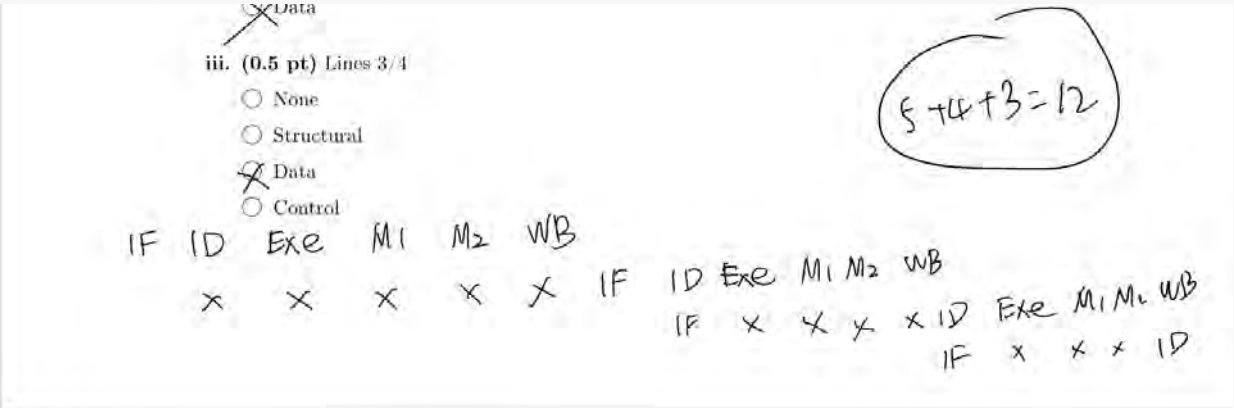
Reply to this followup discussion

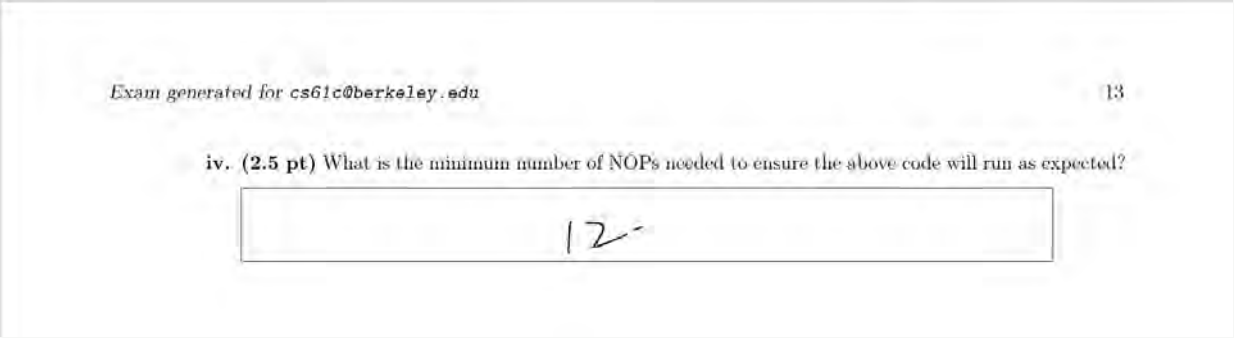
Resolved Unresolved @2128_f70

Anonymous Calc 4 5 months ago

Spring 2021 Midterm Q7

The answer is 10 Nops but I don't know why. I think my diagram is wrong.





helpful! | 0

Adelson Chua 5 months ago

Is this it? @2128_f26

good comment | 0

Anonymous Calc 4 5 months ago

Oh yes. Thanks for linking!

helpful! | 0

Anonymous Calc 4 5 months ago

I forgot where i saw this sentence from a TA's reply "Usually, the decode stage of branch can't happen until after write back stage and instruction happen". What does this mean or can this be illustrate by a diagram?

helpful! | 0

Adelson Chua 5 months ago

Is the branch before or after another instruction? I don't know the context of this.

If there's data dependency and no forwarding, you have to wait until the write back stage of the previous instruction.

If there's a control hazard instead without stalling, you should be putting NOPs or stalls after the branch to avoid executing the next instructions.

good comment | 1

Anonymous Calc 4 5 months ago

For this one, is that mean we know the branch result at executing stage therefore we can start Fetch after exe when we process next instruction? Is that always the case? If the problem says we resolve branch during the memory stage, we can start fetch next instruction after Mem?

helpful! | 0

Anonymous Calc 4 5 months ago

Sorry for the dumb or not-related question, what will the table like if we have branch prediction for line 1 and 2?

helpful! | 0

Adelson Chua 5 months ago

For this one, is that mean we know the branch result at executing stage therefore we can start Fetch after exe when we process next instruction? Is that always the case? If the problem says we resolve branch during the memory stage, we can start fetch next instruction after Mem?

This is exactly correct.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f71

Anonymous Calc 4 5 months ago

How's spring 2020 this question diff from Spring 2021 Q7?

C. (3.0 pt) (CODE INPUT 3):

```
b->data[bitnum >> 3] | (1 << (bitnum & 0x7)) (or equivalent)
```

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

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

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

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

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

```
1 « (n % 8)
```

How is n%8 comparing to bitnum &0x7? Are they both getting the low 3 bits of 1?

helpful! | 0

Anonymous Calc 4 5 months ago

Also, can you explain why we have to do size/8 +1? Can we just malloc(size) given unit_8 is always size 1.

helpful! | 0

Peyrin Kao 5 months ago

I think they're similar in that there's a number that measures something in bits (e.g. index of the bit you want to flip), which you then interpret as bytes. For example, we have to divide `size` by 8 because it's measured in bits, but malloc/calloc takes in the number of bytes.

good comment | 1

Reply to this followup discussion

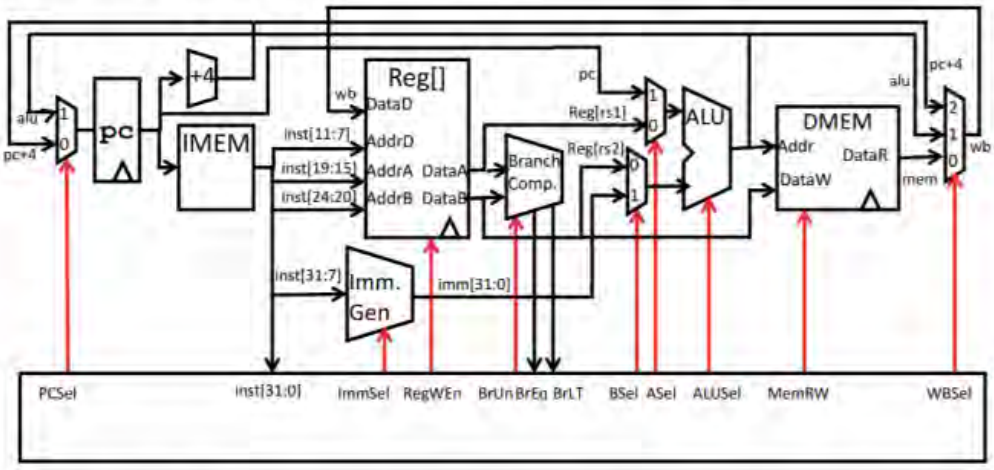
Resolved Unresolved **@2128_f72**

Anonymous Beaker 4 5 months ago

Could someone help explain fa21 Q5, please? really appreciate it!

Q5 ***Big Mac*** **(10 points)**

Below is the standard RISC-V CPU used in Project 3.



Q5.1 (4.5 points) For the instruction "lw", what are the control signals used? If a control signal doesn't matter, select "Don't Care".

Select one option per box. Each box is worth 0.5 points.

PCSel	● PC+4	○ ALU	○ Don't Care	
ImmSel	● I-type	○ S-type	○ B-type	○ U-type
	○ J-type	○ Don't Care		
ASel	○ PC	● rs1	○ Don't Care	

BSel	● Imm	○ rs2	○ Don't Care	
RegWEn	● 1	○ 0	○ Don't Care	
BrUn	○ 1	○ 0	● Don't Care	
ALUSel	● add	○ sll	○ slt	○ xor
	○ srl	○ or	○ and	○ mul
	○ mulh	○ sub	○ sra	○ bsel
	○ Don't Care			
MemRW	● Read	○ Write	○ Don't Care	
WBSel	○ ALU	● MEM	○ PC+4	○ Don't Care

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

`mac rd, rs1, rs2`

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

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

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

Solution:

We now have to read three values from the RegFile to compute `rd + (rs1 * rs2)`, so we need a third output from RegFile. However, we can reuse the R-type format, and use the rd input to RegFile to determine what the third output should be; as such, we don't need a new instruction format or rs3 input.

We also need to pass three values into the ALU to calculate `rd + (rs1 * rs2)`, and we need a new ALU operation to compute the multiplication and addition together on one cycle. The remaining components do not need to be updated.

Adelson Chua 5 months ago

5.1 is straightforward, this is Project 3.

5.2 requires you to implement the mac instruction which operates on 3 registers at the same time, which is why:

1) you need to have a third output for the register file (for rd).

2) you need an additional input to the ALU to perform the mac operation

3) you need to have a new ALU operation to support the mac operation

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f73

Anonymous Atom 5 5 months ago

Spring 2021 Q1

POTPOURRI

(a) Q1

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

$\frac{1}{7}$

Can anymore explain how to do this kind of problem? Do I use Amdahl's law?

helpful! | 0

Adelson Chua 5 months ago

@2128_f65

good comment | 1

Reply to this followup discussion

Resolved Unresolved @2128_f74

Anonymous Calc 4 5 months ago

Spring 2021 Midterm Q8

Conversions: convert the following floating-point representation into their decimal values or vice versa. If the conversion is impossible, write N/A. Please specify infinities as +inf or -inf, not a number as NaN, hex numbers as 0xddddddddd where each d is in [0, f]. If your answer is a decimal number, DO NOT round it.

A. (2.0 pt) $-2^{-100} \cdot 0.625$

0xA6900000

My work: 0.625 = 0.5 +0.125 = 0.101 in binary

so -2^-100 * 0.101 = -1 * 1.01 * 2^-101

exp 8 bit, bias = -127; x-127 = 101, x= 101+127 = 228= 11100100 binary

so the code is 1 11100100 01...000

shouldn't be F2100000 instead?

helpful! | 0

Adelson Chua 5 months ago

Did you consider that the exponent field is 9 bits here?

The bias will also effectively change because of that.

good comment | 0

Anonymous Calc 4 5 months ago

Oh, I copy down the wrong exp bit. Got it thanks!

helpful! | 0

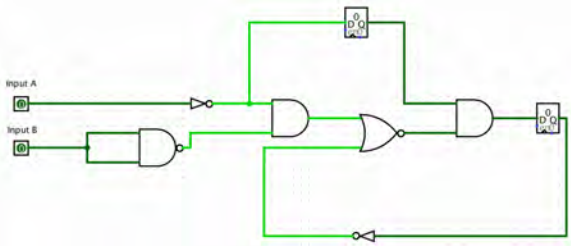
Reply to this followup discussion

Resolved Unresolved **@2128_f75**

Anonymous Scale 5 months ago

SP21-Final-Q4, SP18-Final-Q5

4. SDS



State Machine

Assume input A and input B come from registers. Please include ns in your answer.

(a) Assume all 2-input logical gates have a 10 ns propagation delay. The NOT gate has a 5 ns delay. All registers have a clk-to-q of 15 ns and setup time of 20 ns.

i. Find the minimum clock period to ensure the validity of the circuit.

75 ns

We have the following paths:

- Input A (clock-to-q) -> NOT -> Register (setup) = 15 ns + 5 ns + 20 ns = 40 ns
- Input A (clock-to-q) -> NOT -> AND -> NOR -> AND -> Register (setup) = 15 ns + 5 ns + 10 ns + 10 ns + 10 ns + 20 ns = 70 ns
- Input B (clock-to-q) -> NAND -> AND -> NOR -> AND -> Register (setup) = 15 ns + 10 ns + 10 ns + 10 ns + 10 ns + 20 ns = 75 ns
- Register (clock-to-q) -> NOT -> NOR -> AND -> Register (setup) = 15 ns + 5 ns + 10 ns + 10 ns + 20 ns = 60 ns

So we need the max of them which would be 75 ns.

Problem 5 ***[MT2-1] Circuits and Timing*** **(9 points)**

In this question, you will be working with a circuit that takes in three 8-bit inputs. For all parts, assume the delays below:

$t_{clk-to-q} = 3\text{ps}, \quad t_{setup} = 4\text{ps}, \quad t_{shifter} = 1\text{ps}$
$t_{adder} = 5\text{ps}, \quad t_{multiplier} = 6\text{ps}, \quad t_{subtracter} = 4\text{ps}$

Furthermore, assume that the inputs A, B, and C take on their new values exactly at the rising edge of every clock cycle and that all registers are initialized to zero.

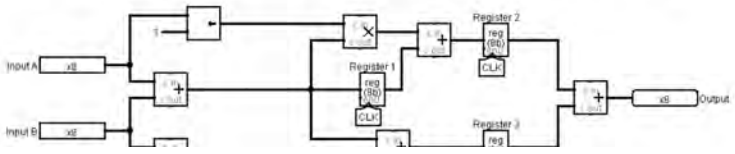


Figure 2: Non-pipelined circuit

(a) What is the maximum possible hold time that still ensures the correctness of the non-pipelined circuit in figure 2? (Select only one)

- ○ 1ps
- ● 5ps
- ○ 3ps
- ○ 7ps
- ○ 4ps

Solution: 5ps (Inputs A/B modify input to register 1 5ps after rising edge)

(b) What is the minimum possible clock period that still ensures the correctness of the non-pipelined circuit in figure 2? You may assume that for this question that all flip-flops have a 0ps hold time requirement. (Select only one)

- ○ 13ps
- ● 20ps
- ○ 16ps
- ○ 23ps

Solution: 20ps (Input B to Register 2)

<-- this is Sp 2018

There is a discrepancy between these 2 exams. For the 2021 question, we treat the input as a register and we add the clock-to-q to the minimum clock period. However, for the 2018 exam, when we have input nodes, we were asked not to add the clock-to-q time for either the min clock cycle or the maximum possible hold time. So on tomorrows exam, which one should we adhere to?

helpful! | 0

Adelson Chua 5 months ago

Assume input A and input B come from registers.

Therefore, consider clk-to-q

Assume that the inputs A, B, and C take on their new values exactly at the rising edge of every clock cycle.

Therefore, we model the inputs as coming from registers without clk-to-q (because it says 'exactly at the rising edge')

There is no discrepancy here. Just language. Difference in setup condition.

good comment | 1

Anonymous Scale 5 months ago

okay I see. I guess I misread the Spring 2018 one. Thank you.

helpful! | 0

Reply to this followup discussion

Resolved Unresolved @2128_f76

Anonymous Calc 4 5 months ago

Why we need to divide i by 8? Can't we just add s5 and s6 to get the correct start pointer address?

Spring 2021 midterm

iii. (4.0 pt) Translate the line baz(f[i].b) into RISC-V assembly. Assume that f is in S5 and i is in S6. You should use only 4 instructions and you can only use a0 as a temporary. **You may NOT use mul, div, or rem!**

```
sll a0 s6 3
add a0 a0 s5
lw a0 4(a0)
jal baz
```

helpful! | 0

Adelson Chua 5 months ago

You are indexing the structure:

```
struct foo {
char a;
char *b;
}
```

run code snippet Visit Manage Class to disable runnable code snippets

This has a size of 8 bytes. (1 byte for char a + 3 byte padding + 4 byte pointer).

So the index must be multiplied by 8 to access the next element.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f77

Anonymous Scale 5 months ago

SP21-Final-Q6.b.v.

v. load `sl->next[level]` into t0

```
slli t1 t1 4 add t0 t0 t1 lw t0 t0(4)
```

I understand that we need to move the pointer over by 4 bytes for each element we are trying to move over. In this case, shouldn't the instruction be "slli t1 t1 2" so we move over 4 bytes per element (as each element is a pointer which is 4 bytes in a 32 bit system, which I'm assuming is what we are working in)?

helpful! | 0

Anonymous Scale 5 months ago

More context: t1 stores level and t0 stores sl->next.

```
struct SLN{
   void *data;
```

```
    struct SLN **next;
}
```

iii. Load level into t1

```
lw t1 sp(4)
```

iv. load sl->next into t0

```
lw t0 t0(4)
```

helpful! | 0

Anonymous Scale 5 months ago

resolved above, it should be slli t1 t1 2

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f78**

Anonymous Scale 3 5 months ago

Fa20-Final-1a

1: Quest Clobber (*10 pts*)

Part A — 2 pts Recall that an 8-bit bias-encoded number normally has a bias of -127 so that roughly half the numbers are negative and half are positive, but there's one more positive than negative number. Using an equivalent scheme for choosing the bias, what base 14 number XXXXXX represents 0? (That is, your answer needs to have 6 base-14 characters.)

Solutions

Answer: 6DDDDD. For now, let's pretend that 0 is negative; that way, we want exactly half the numbers to be negative, and half to be positive, and all numbers are positive or negative. If we do that, then we want the cutoff from negative to positive to be right in the middle of all possible numbers; this is between the numbers 6DDDDD and 700000. Our 0 would be the number right before this cutoff, and is thus 6DDDDD.

Variants: Different bases (all even, so the above process works), and different number of digits.

Could someone please explain this. Cannot seem to understand how they approach the problem at all

helpful! | 0

Adelson Chua 5 months ago

@2127_f3

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f79**

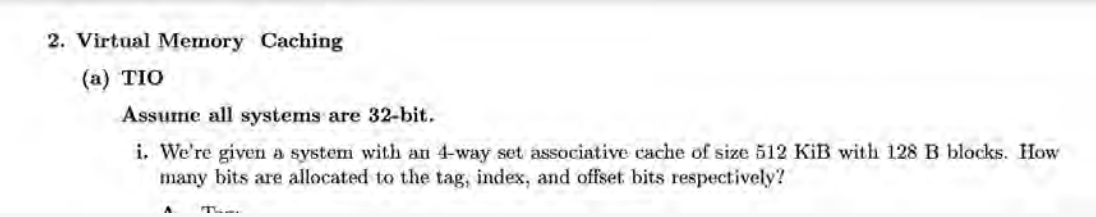
Anonymous Helix 2 5 months ago

Sp 2021 Final Question 2a

Relevant info: Block Size is 128 B blocks

In the walkthrough video, the TA asked how many ints we can fit in 1 block. He said do (128/32) to get 4 ints per block.

Shouldn't it be 32 ints per block? 128B per block / 4B per int = 32 ints per block? You don't need to figure out num ints until part e.



2. Virtual Memory Caching

(a) TIO

Assume all systems are 32-bit.

i. We're given a system with an 4-way set associative cache of size 512 KiB with 128 B blocks. How many bits are allocated to the tag, index, and offset bits respectively?

helpful! | 0

Anonymous Helix 2 5 months ago

Nevermind. They fix it later in the video.

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f80**

Anonymous Helix 2 5 months ago

Sp 2021 Final Question 2a

I have a question about how data is pulled into the cache.

If each block of the cache can hold 32 ints, and I want to access array[5], does the cache pull in array[0] to array[31] or does it pull in array[5] to array[36]?

helpful! | 0

Adelson Chua 5 months ago

array[0] to array[31].

You should always check the byte offset. The cache pull is from byte offset 0000... to 1111...

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f81**

Anonymous Comp 5 5 months ago

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

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

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

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

Solution: 2

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

Q7.2 (1.5 points) What is the T:I:O breakdown of memory addresses?

T	I	O

Solution: 4:2:4

From the table, each tag is 4 bits long, and each index is 2 bits long. Since the address is 10 bits long in total, the offset must be 10-4-2=4 bits long.

I understand that the number of index bits, as shown from the table, is 2. However, didn't we learn that the number of index bits in an N-set associative cache is log_2(# sets), which should mean that the # of index bits is 1?

helpful! | 1

Adelson Chua 5 months ago

Set = Index in the context of set-associative caches.

I see 4 lines, log2(4) = 2.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f82**

Harrison 5 months ago

SU21-MT2-Q3. For several of the subparts, both malloc and calloc are marked as correct. Does this mean that either option is acceptable?

helpful! | 0

Adelson Chua 5 months ago

Yeah, most likely.

Disclaimer: We haven't checked the correctness of the solutions before making this available.

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f83**

Anonymous Helix 5 5 months ago

Sp 21 10b:

Could someone please provide an explanation/walkthrough for this question?

helpful! | 0

Anonymous Helix 5 5 months ago

Sp 21 Final 10B*

helpful! | 0

Adelson Chua 5 months ago

The RISC-V translation? Isn't this similar to what has been done in the homework before?

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f84

Anonymous Scale 4 5 months ago

su21-final-q12

I got part a for this question and got the correct offset (1A) for Part B, but I'm pretty clueless on where the 9C comes from.

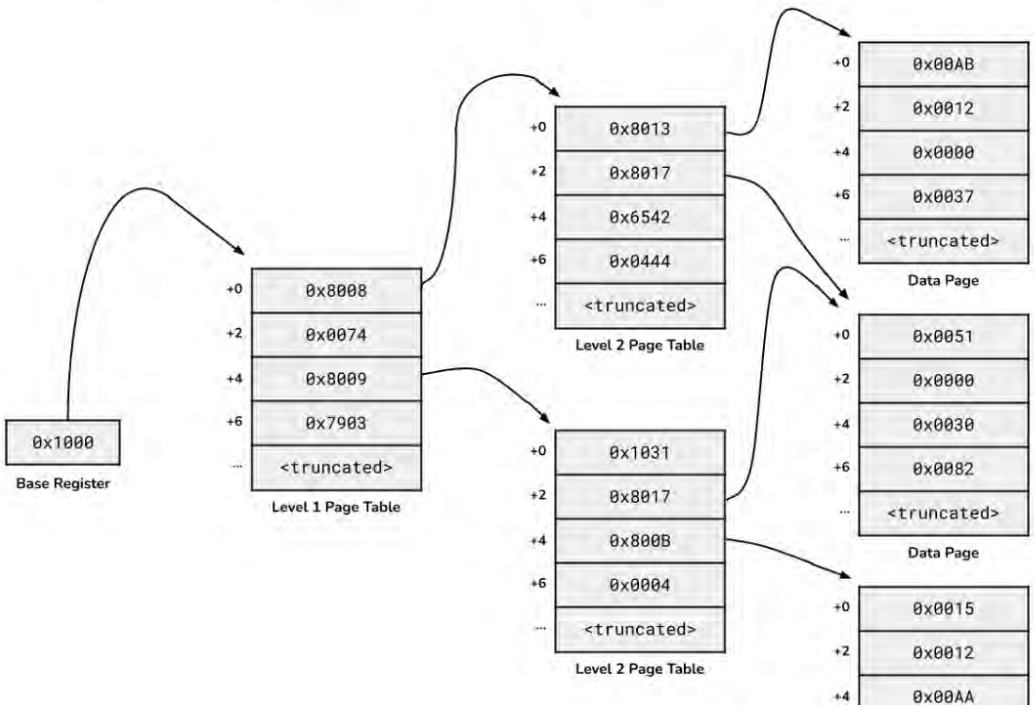
For this problem, consider a machine with a **2 GiB of virtual memory**, **4 MiB of physical memory**, and **2 KiB page size**. The system uses a **2-entry fully-associative LRU TLB**, and a **2-level hierarchical page table**. Assume VPN bits are evenly split between the two levels, so every PT at every level has the same number of entries. Assume also that a single PTE is **2 bytes**, where the most significant bit is 1 when the entry is valid.

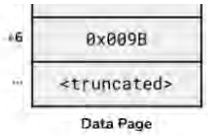
PART A (5 pts.)

How many bits are in each of the following fields of the virtual and physical addresses?

L1 VPN 10	L2 VPN 10	VA Page Offset 11
PPN 11		PA Page Offset 11

Now, suppose physical memory and the page table have this initial state. Each arrow indicates the location of either the next-level page table or the physical data page being pointed to.





PART B (3 pts.)

What physical address does virtual address `0x0000041A` map to? Write your answer in hexadecimal, **without the leading `0x`**, or write `N/A` if this does not map to a valid physical address.

Displayed correctness for this blank may be inaccurate, but total points on this question should be correct (see errata in Ed grades post).

0x `9C1A`

helpful! | 0

Adelson Chua 5 months ago
Show me your solution then.
good comment | 0

Anonymous Scale 4 5 months ago
I converted 0x0000041A, to binary:
0000 0000 0000 0000 0000 0100 0001 1010

The offset is the bottom 11 bits = 100 0001 1010. This corresponds to 8AC

not sure what else to do here?
helpful! | 0

Adelson Chua 5 months ago
You haven't done the homework similar to this one?
good comment | 0

Adelson Chua 5 months ago
Maybe this will help.
@2187
good comment | 0

Adelson Chua 5 months ago
and this.
@2128_f24

Please do the homeworks next time. You learn a lot of stuff from those. :)
good comment | 0

Anonymous Scale 4 5 months ago

Thanks for linking to the other post about this. I searched for it with the convention but it didn't show up that way.

helpful! | 0

Anonymous Scale 4 5 months ago

I'm still a bit confused on the relevance of "Assume also that a single PTE is 2 bytes"? When the entry for L1 VPN = L2 VPN =0 is 0x8013, that's more than two bytes?

helpful! | 0

Adelson Chua 5 months ago

1 byte = 8 bits.

good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f85

Anonymous Calc 4 5 months ago

Hi, can I get a walk-through of this part? I have no idea how to approach it. Spring 2021 Final Q2 part H

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

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

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

1024

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

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

8

Alternate solution: 6 caches

K. We now switch our system to having a 2-level page table instead. Assuming we restart our system and run one iteration of the inner loop, and that the VPN bits are split evenly between levels, how many pages will our active page tables span?

2

No alternate solution.

helpful! | 0

Adelson Chua 5 months ago

I'll give you this one first.

@2128_f14

good comment | 0

Anonymous Calc 4 5 months ago

If we have to align the page, which is either byte align or word align, would that still be matter how long the PTE is? For byte align, if PTE is 32 bit, the size of PTE becomes 4 byte(32/8)? If PTE is 33 bit, will it becomes 5 bytes?

Also, the wordalign we divide by 32? I though a word is 2 byte which is 16 bits...

helpful! | 0

Adelson Chua 5 months ago

For byte align, if PTE is 32 bit, the size of PTE becomes 4 byte(32/8)? If PTE is 33 bit, will it becomes 5 bytes?

This is correct if byte-aligned. Then if word-aligned 33 bits rounds up to 64 bits = 8 bytes.

Word is 32 bits.

good comment | 0

Anonymous Calc 4 5 months ago

Thank you! So for part K, since we have 2 level, each level has 10 bit VPN, for L2, it will have 2^10 PTEs. And L1 is 1 page table. Should L2 has total number of 2^10 pages tables since each entry in L1 will map to one L2 page table? Which result in 1+2^10 page tables in total?

helpful! | 0

Adelson Chua 5 months ago

Not all possible L2 page tables are loaded into the memory right away. It 'grows' as you demand more translation. So at the minimum, you only have 2 tables, 1 for L1 1 for L2.

good comment | 0

Anonymous Calc 4 5 months ago

So this question is asking for miniumn page table we will span?

helpful! | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f86**

Anonymous Gear 5 5 months ago

SP21 Final Q4 - I'm confused on why A has a performance improvement and not B or C. A would never experience a data race, but the other for loops would. And how does the incrementation of i in the for loop distinguish A's performance from C?

helpful! | 0

Adelson Chua 5 months ago

Are you referencing the correct question?

good comment | 0

Anonymous Mouse 5 5 months ago

I have the same question as Gear.

(d) Q4

Which of the following code blocks would see a performance improvement if we placed a `#pragma omp for` over the outer for loop? Select all that apply.

i. A

```
for (int i = 0; i < 5000 - 3; i += 3) {
    a[i+2] = a[i] + a[i+1];
}
```

B

```
for (int i = 0; i < 5000 - 2; i += 2) {
    a[i+2] *= a[i];
}
```

C

```
for (int i = 0; i < 5000 - 3; i++) {
    a[i+2] = a[i] + a[i+1];
}
```

D

```
for (int i = 0; i < 5000; i++) {
    a[i] = 100;
}
```

- [x] A
- [ ] B
- [ ] C
- [x] D
- [ ] E. None of the above

helpful! | 0

Adelson Chua 5 months ago

@2128_f64

good comment | 0

Reply to this followup discussion

Resolved Unresolved **@2128_f87**

Vihaan Doshi 5 months ago

[SU-21 Final Q9] I marked slower than serial and got a 100% but the correct answer says about the same. Which one is it? Were both answers given correct because it depends on the size of the array?

Naive

```
void f(int *arr, int len) {
  int old_value = 0;
  for (int i = 0; i < len; i++) {
    old_value = arr[i];
    arr[i] = arr[i] + 1;
  }
}
```

Optimized

```
void f(int *arr, int len) {
  int old_value = 0;
  #pragma omp parallel for
  for (int i = 0; i < len; i++) {
    old_value = arr[i];
    #pragma omp critical
    arr[i] = old_value + 1;
  }
}
```

Answer:

(a) always incorrect	(a) slower than serial ✗
(b) sometimes incorrect ✓	(b) about the same speed as serial
(c) always correct	(c) faster than serial
✓ 100%	✓ 100%

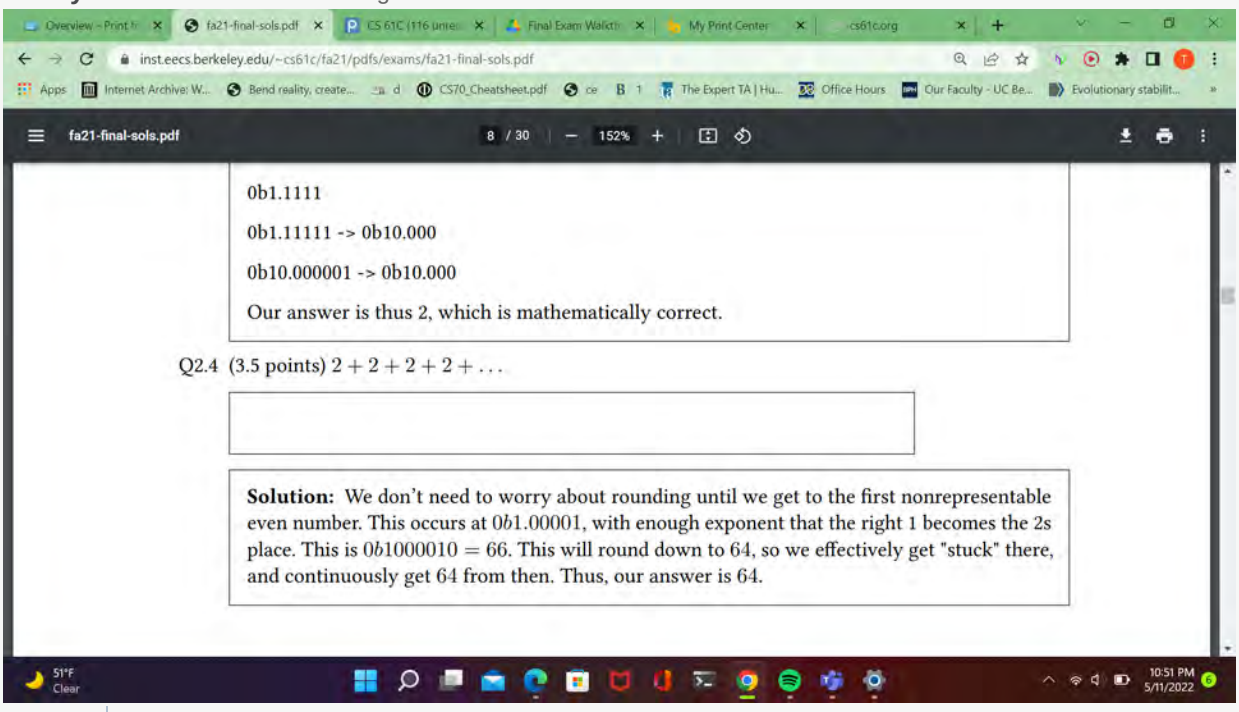
helpful! | 0

Adelson Chua 5 months ago
@2206_f33
good comment | 0

Reply to this followup discussion

Resolved Unresolved @2128_f88

Anonymous Scale 5 5 months ago



helpful! | 0

Anonymous Scale 5 5 months ago
Pretty confused about Q2.4 Fa21 explanation
helpful! | 0

Adelson Chua 5 months ago
@2128_f59
good comment | 0

Anonymous Scale 5 5 months ago
Thanks!

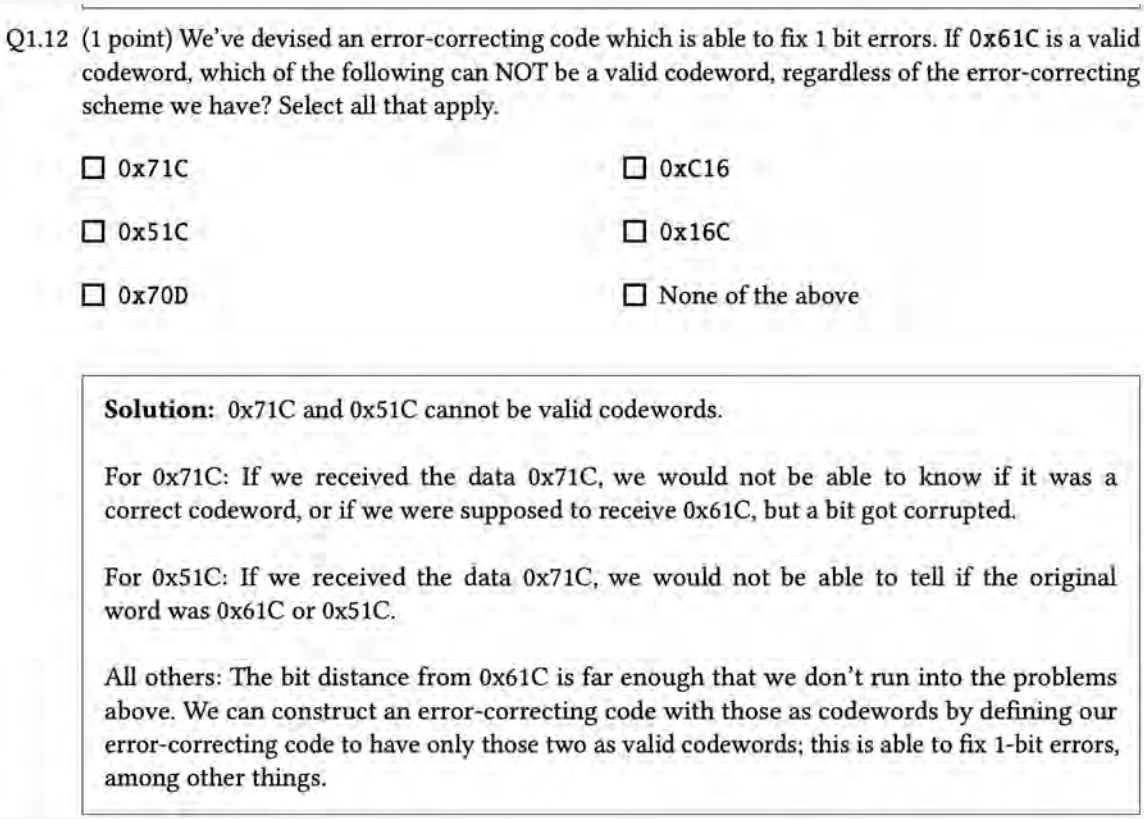
helpful! | 0

Reply to this followup discussion

Resolved Unresolved @2128_f89

Anonymous Calc 4 5 months ago

i am confused about this one. fall 2021



Q1.12 (1 point) We've devised an error-correcting code which is able to fix 1 bit errors. If 0x61C is a valid codeword, which of the following can NOT be a valid codeword, regardless of the error-correcting scheme we have? Select all that apply.

- ☐ 0x71C
- ☐ 0xC16
- ☐ 0x51C
- ☐ 0x16C
- ☐ 0x70D
- ☐ None of the above

Solution: 0x71C and 0x51C cannot be valid codewords.

For 0x71C: If we received the data 0x71C, we would not be able to know if it was a correct codeword, or if we were supposed to receive 0x61C, but a bit got corrupted.

For 0x51C: If we received the data 0x71C, we would not be able to tell if the original word was 0x61C or 0x51C.

All others: The bit distance from 0x61C is far enough that we don't run into the problems above. We can construct an error-correcting code with those as codewords by defining our error-correcting code to have only those two as valid codewords; this is able to fix 1-bit errors, among other things.

Is valid codeword mean something is correct? So if we have 0x71C, we one have one bit corrupted which should be able to fix through our error correcting code? I didn't get what question is asking even after reading the explanation.

helpful! | 0

Adelson Chua 5 months ago

@2128_f34

good comment | 0

Anonymous Calc 4 5 months ago

Thank you! So we actually want our code word to have more differences that they won't be easily mess up with each other? So I can know what my original word is?

helpful! | 0

Adelson Chua 5 months ago

Yes

good comment | 0

Reply to this followup discussion

Start a new followup discussion

Compose a new followup discussion