note @432

[Exams] Past Exams 2018 Q&A

Discuss all questions pertaining to exams which took place in 2018 here.

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

When posting questions, you **MUST** reference the semester, exam, **AND** question so we can help you.
Please put this at the beginning of your post in this format: **[{Semester}-{Exam}]:Q{Question Number}**
For example: **[SP-MT1]:Q1**, or **[SU-MT2]:Q3**

{Semester} is one of these: SP, SU, FA
{Exam} is of of these: Q, MT, MT1, MT2, F
Please separate out parts with periods: 1.2.ii.a.b.3.a

If you follow this format, it will make it very easy to search for similar questions!

midterm1 midterm2 final

~ An instructor (Jerry Xu) thinks this is a good note ~

Updated 27 days ago by Stephan Kaminsky

followup discussions *for lingering questions and comments*

Resolved Unresolved

Xxxxxxx 2 months ago
[Spring-MT1]: Q3a
Why is song1 an address on the heap? Isn't song1 a pointer to memory on the heap, and isn't that pointer stored on the stack? I thought *song1 would have been on the heap.
helpful! | 0

Caroline Liu 2 months ago The question is asking what kind of **address** does each variable evaluate to. song1 is referring to the ptr to the memory you just allocated with malloc (heap). The ptr is just an address and that address is the address starting point of what you just malloced. &song1 would be on the stack. In this case, *song1 isn't an address, it's a struct.
helpful! | 0

Resolved Unresolved

Xxxxxxx 2 months ago
[Spring-MT1]:Q5dii
Why do we multiply by 17 here?
helpful! | 0

Xxxxxxx 2 months ago The size of an instruction is defined as 17 bits in this question. So we need to multiply it by 17 to get the maximum number of bits away from current pc.

helpful! | 0

Anonymous Scale 2 months ago But why it is the 2^(n-1) -1 instead of 2^n -1

helpful! | 0

Anonymous Scale 2 months ago I got it

helpful! | 0

Resolved Unresolved

Xxxxxxx 2 months ago

[Spring-MT1]:Q6c

Why do we return sum in the function? I believe the values of a1 is 8, and a0 and a2 are both pointers by the end of the execution of MAGIC. Shouldn't we be returning a pointer instead?

helpful! | 0

Caroline Liu 2 months ago In the second to last line we do mv a0 s0 which stores the sum we've been calculating in the return register a0. Then we jr ra which returns to main. This corresponds to returning sum from the magic C function.

helpful! | 0

Resolved Unresolved

Anonymous Comp 2 months ago

[Summer-MT1]:Q2.7

Why is it possibly legal? I thought because we only copied 'h' and 'i' and not the null pointer, we cannot legally compute the strlen(x) since it only stops when it reaches a null terminator?

helpful! | 0

Anonymous Comp 2 months ago I meant Q2.6

helpful! | 0

Anonymous Atom 2 months ago +1.

My understanding is that msg is "hi\0" and that when the call

```
strncpy(x,msg,strlen(msg));
```

is made, strlen(msg) is equal to 2. Therefore x would be equal to "hi" (no null terminator). I looked into what strlen(x) would result in in such a case, and it said that it would keep searching until it hit a null terminator or there was an error. Therefore I guess it is possibly legal as the space strlen is referencing is allocated but depending on the value in x[2]? Not 100% sure

helpful! | 0

Resolved Unresolved

Anonymous Gear 2 months ago

[Summer-MT1]:Q5.1(second)

Why the largest number of registers that can now be supported in hardware in 64? I don't quit understand the explanation in the solution.

helpful! | 0

Anonymous Gear 2 months ago nvm I understand

helpful! | 0

Resolved Unresolved

Xxxxxxx 2 months ago

[SP-MT1]:Q2.b

I noticed that the solution did not initialize "curr->next = NULL;" in the 2nd if block. Can we assume this to have been done automatically when curr was initialized? If not, wouldn't this be a problem if find_end is called, given the implementation in Q2.a?

helpful! | 0

Resolved Unresolved

Anonymous Scale 2 months ago

helpful! | 0

Resolved Unresolved

Anonymous Scale 2 months ago

```
foo:
            slli  t6 a0 2
            sub   sp sp t6
            mv    t4 sp
            sw    zero 0(t4)
            addi  t1 zero 1
L1:         bge   t1 a0 Next
            andi  t2 t1 1
            slli  t3 t1 2
            add   t3 t3 t4
            sw    t2 0(t3)
            addi  t1 t1 1
            j     L1
```

```
Translate the RISC-V Assembly on
the left into C code to complete
the function foo:

unsigned foo(unsigned n) {
  unsigned arr[n];
  unsigned total = 0;
  unsigned *ptr = arr;
  ptr[0] = 0;
  for (int i = 1; i < n; i++) {
    ptr[i] = i & 1;
  }
  for (int i = 0; i < n; i++) {
    total +=  ptr[i];
```

I do not understand why these two instruction are equal?Thanks.

helpful! | 0

Anonymous Gear 2 months ago I believe they are not equal.
andi t2, t1, 1 is only i & 1
ptr[i] = i & 1 is sw t2, 0(t3)

helpful! | 0

Xxxxxxx 2 months ago Anonymous Gear is right.

andi t2 t1 1 --> t2 = t1 & 1
slli t3 t1 2 + add t3 t3 t4 --> t3 is now the address of ptr[i]
sw t2 0(t3) --> t3 = ptr[i] = t2 = t1 & 1 = i & 1
helpful! | 1

Resolved Unresolved

Anonymous Poet 2 months ago
Rendering markdown...
helpful! | 0

Anonymous Poet 2 months ago
Rendering markdown...
helpful! | 0

Resolved Unresolved

Anonymous Scale 2 months ago
[Fa-quest]:
Q3.a: backup is a variable created in function, it should be stack, *backup should be in heap, why solution shows the heap?

Q3b backup[1] is G, why?
backup = copy,
when copyH increrase and copy is modifies, backup should be modified as well, right?
helpful! | 0

Caroline Liu 1 month ago Which question are you talking about? (Also sorry about the late reply!)
helpful! | 0

Resolved Unresolved

Anonymous Atom 2 months ago
[SU18]: Clarification

For Question 4, I saw that the function was void, so I assumed it didn't return anything. I see from the solutions that it returns a pointer. What does a function returning void mean then?
helpful! | 0

Xxxxxxx 2 months ago **[SU-MT1]:Q4**
The function actually says `void *user_malloc (size_t n)`, so it will return a void pointer, which is just a generic pointer that can point to any data type.
helpful! | 0

Anonymous Atom 2 months ago Thank you!
helpful! | 0

Resolved Unresolved

Anonymous Helix 1 month ago

[SU-18] Q5

I am confused by the answers, what does FwdOutA and FwdOutB do in the datapath here?

helpful! | 0

Xxxxxxx 1 month ago They refer to the output from the pipeline registers.

helpful! | 0

Resolved Unresolved

Xxxxxxx 1 month ago

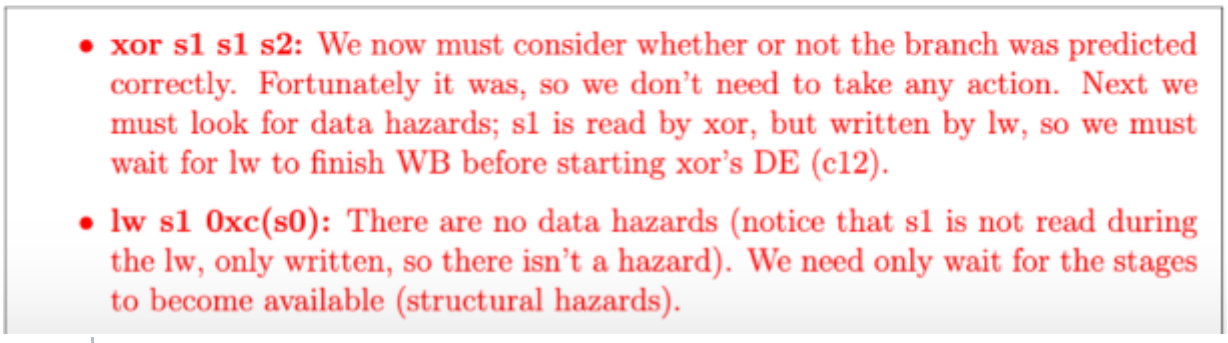
[SP-18] 4A according to the explanation below, shouldn't the D stage for both beq and xor come after the stalling? Because I thought beq instruction decode stage cannot happen until andi instruction writes back.

Instructions	Cycles															
	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16
ori s1 x0 0xf	F	D	E	M	W											
andi s2 x0 0		F	D	E	M	W										
beq s1 s2 exit			F	D	*	*	*	E	M	W						
lw s1 0xc(s0)				F	*	*	*	D	E	M	W					
xor s1 s1 s2								F	D	*	*	*	E	M	W	
lw s1 0xc(s0)									F	*	*	*	D	E	M	W

Solution:

The first thing to notice for this question is that the datapath does not implement bypassing, and a register cannot be simultaneously read and written in the same cycle. Recall that instructions read their registers in stage DE, and write registers in WB. These restrictions mean that if instruction B needs a register that instruction A writes, then B cannot start its DE stage until the cycle after A's WB stage (this is a "data hazard"). The other type of hazards to worry about are "structural hazards", this means that no two instructions can be in the same stage at the same time. Now lets go through the answer instruction-by-instruction:

- **andi s2 x0 0:** This doesn't have any data depencies, so we just need to worry about structural hazards. It can start as soon as the F stage is available (c2).
- **beq s1 s2 exit:** This instruction reads register s2 which was written by the previous instruction. Therefore we must wait until the andi has finished its WB stage before running beq's DE stage (c7).
- **lw s1 0xc(s0):** At this point, the result of the branch doesn't matter because it is always predicted to be taken. Also, the branch doesn't write any registers, so we don't have any data dependencies and can start as soon as the stages are available. In this case, the fetch can start on c4, but the decode has to wait until beq is done with it (c8).



- **xor s1 s1 s2:** We now must consider whether or not the branch was predicted correctly. Fortunately it was, so we don't need to take any action. Next we must look for data hazards; s1 is read by xor, but written by lw, so we must wait for lw to finish WB before starting xor's DE (c12).
- **lw s1 0xc(s0):** There are no data hazards (notice that s1 is not read during the lw, only written, so there isn't a hazard). We need only wait for the stages to become available (structural hazards).

helpful! | 0

Xxxxxxx 1 month ago beq enters decode stage at c4 but it has no effect on the result since only andi takes control of execute stage. It stalls until c7 after andi finishes writeback stage.

helpful! | 0

Resolved Unresolved

Xxxxxxx 1 month ago

[SP-18] 5aiii I thought if we run this program with a fully associative cache, there will be a point where the cache is full so there should be capacity misses as well.

helpful! | 0

Xxxxxxx 1 month ago But for this question aren't we talking about direct mapped cache? Only one block is used and we keep replacing it with new data. There's no capacity miss.

helpful! | 0

Resolved Unresolved

Anonymous Comp 1 month ago

[SU-18] 5.2B I am wondering how to solve this type of problem in general. I am not sure where this came from (block size of L2).

helpful! | 0

Xxxxxxx 1 month ago We only access L2 if there's a L1 miss. From the L2 hit rate we know that there's a miss every 4 accesses of L2. Therefore, every time L2 loads in a new block, it will contain consecutive data that L1 can load in 4 times before there's a L1 miss. So a L2 block is basically 4 times as large as a L1 block, in this case 4Y bytes.

helpful! | 0

Anonymous Comp 1 month ago ah okay thank you!

helpful! | 0

Resolved Unresolved

Xxxxxxx 1 month ago

[SP-MT2]:Q12.f

Why is it 3=4<1=2? Don't really know where to begin on this one.

helpful! | 0

Daniel Fan 1 month ago resolved in OH

helpful! | 0

Xxxxxxx 1 month ago @Daniel I thought so too, but we hastily assumed that 4096 bytes = 4096 ints, but it's only 1024 ints. There would actually be 4-5 TLB evictions for T=1, depending on whether or not main memory is page aligned, and it gets hideously complicated after that for other values of T, as another TA and I found out later yesterday. Please resolve!

helpful! | 0

Daniel Fan 1 month ago Ah, I see, MB LOL. 4096 ints are indeed different than 4096 bytes. I just looked at the problem again and this is my reasoning for the answer.

First of all, the cache is completely irrelevant for all 4 values of T as in every case the first loop will load all the values into the cache, meaning there will never be a miss in the second loop. Thus, the only difference in execution time will be due to TLB hits and misses.

For T = 1, the TLB miss/hit pattern will be: miss at start of loop as i = 5 * 1024 * 1024 which is a multiple of 4096, meaning we are loading a new page into the TLB. It will then continue to have TLB hits until i = 5 * 1024 * 1024 + 1024 when a new page will be loaded into the TLB. So the overall pattern will be 1 TLB miss every 1024 accesses until the end of the loop.

For T = 2, the TLB miss/hit pattern is almost identical. Instead of only missing on i = 5 * 1024 * 1024, it will also miss on i = 5 * 1024 * 1024 + 1 as this is 4 pages away from the previous access. Because it is % 2, the rest of the 1022 accesses will stay on these two pages and so we'll continue to have TLB hits until i = 5 * 1024 * 1024 + 1024 like before. So the overall pattern is 2 TLB misses every 1024 accesses.

For T = 3 and 4, due to the TLB evicting previous entries every on every access, the TLB will never hit. So both of them will have only TLB misses.

So that is why 3 = 4 < 1 and 3 = 4 < 2. I'm guessing that because of the wording "likewise, you could write 8=2 if 8 is about as fast as 2" (keyword:"about as fast"), they then write 1 = 2 as they only differ by 1 TLB miss every 1024 accesses.

Sorry for the mouthful and late response—I just saw this lol.

helpful! | 0

Resolved Unresolved

Xxxxxxx 1 month ago

Rendering markdown...

helpful! | 0

Anonymous Mouse 1 month ago If we had the traditional IF, ID, EX, MEM, WB pipeline, I believe we would need to stall for 2 cycles since we would need to wait until after EX to start our next IF. However, since IF and ID are combined into 1 phase (IFD), that is why we only need to stall for 1 cycle.

helpful! | 0

Resolved Unresolved

Anonymous Beaker 1 month ago

[SU-MT2]:Q2

Question 2: Simple Democratic Selection (12 pts)

As the semester is reaching a close, Steven, Nick, and Damon are busy determining the difficulty of the final exam. All three will vote on whether the final should be easy or hard, but the final decision will always be made based on the following rules:

Rule 1. If the vote is unanimously hard or unanimously easy, then it will be hard or easy, respectively.
Rule 2. If Damon disagrees with Steven and Nick, then Damon's vote will be chosen.
Rule 3. If Steven and Nick differ, then the minority vote will be chosen.
Else In all other situations, the outcome can be either easy or hard (i.e. they can be anything)

We will represent Steven's vote with the variable **S** which takes on values of 0 (easy) and 1 (hard). Similarly, Nick's vote is represented as **N** and Damon's vote is represented as **D**. For each rule, write out the **simplest** boolean logic expression using these three binary inputs that outputs whether or not the final exam will be easy or hard. Note: the symbol for XOR is ⊕.

Rule 1: SND **Rule2:** $(D \oplus S)(D \oplus N)D$ *or* $\overline{S \oplus N}D$ **Rule3:** $(S \oplus N)\overline{D}$

```
X = ( a ⊻ b ) && ( a ⊻ c ) && a
```

a	b	c	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

#N : 9

Question 2: Simple Democratic Selection (12 pts)

As the semester is reaching a close, Steven, Nick, and Damon are busy determining the difficulty of the final exam. All three will vote on whether the final should be easy or hard, but the final decision will always be made based on the following rules:

Rule 1. If the vote is unanimously hard or unanimously easy, then it will be hard or easy, respectively.
Rule 2. If Damon disagrees with Steven and Nick, then Damon's vote will be chosen.
Rule 3. If Steven and Nick differ, then the minority vote will be chosen.
Else In all other situations, the outcome can be either easy or hard (i.e. they can be anything)

We will represent Steven's vote with the variable **S** which takes on values of 0 (easy) and 1 (hard). Similarly, Nick's vote is represented as **N** and Damon's vote is represented as **D**. For each rule, write out the **simplest** boolean logic expression using these three binary inputs that outputs whether or not the final exam will be easy or hard. Note: the symbol for XOR is ⊕.

Rule 1: _____ **Rule2:** _____ **Rule3:** _____

Below is a boolean algebra expression that models this problem. Simplify it into as few gates as possible:

my answer : rule2: D.!S.!N

why is my answer not correct ?

helpful! | 0

Resolved Unresolved

Anonymous Beaker 1 month ago

Using the assumptions above, what is the smallest value that the clk-to-q can be and **not** cause a hold-time violation?

clk-to-q: 11 - 5 + 5 = 1 ps

3

parenthesis pb

helpful! | 0

Resolved Unresolved

Xxxxxxx 1 month ago

SU18, can someone please explain 5.2A. I don't understand why it's 16B and not 128B.

helpful! | 0

Anonymous Mouse 1 month ago The way I thought about it is that we want to hit 7 out of every 8 accesses to arr[i]. This means that we want to only have that initial compulsory miss every 8 iterations, bring in that whole block, and then hit the next 7 within that block. This means we need 8 uint16_t's per block. A uint16 is composed of 16 bits, so 2 bytes. Therefore, if we want to fit 8 uint16_t's in a block, we need to do 8 * 2 = 16B per block.

helpful! | 0

Resolved Unresolved

Xxxxxxx 1 month ago

SU18 4.1a, why does line 9 cause a stall for line 5?

5. L2: bge t1 a0 End

.

.

.

9. addi t1 t1 4

10. j L2

Since this question is referring to a 3 stage pipeline, shouldn't t1 be safe by the time we loop back to it?

helpful! | 0

Anonymous Mouse 1 month ago I think it's because we can't read and write to the same register in the same cycle. So, you cannot write to t1 in line 9 and read from t1 in line 5 in the same clock cycle, meaning you can't have the MWB and IFD stages line up.

helpful! | 0

Resolved Unresolved

Anonymous Calc 1 month ago [SU-Final]:Q6.3

How do you determine whether or not to cache a block with the no write allocate policy? Why is it that C is never cached if it is being accessed every iteration of the inner for-loop?

helpful! | 0

Jenny 1 month ago No write allocate policy means after we request the address to cache, if we get a write miss, we directly find the data in memory then write to memory, so we never brought in the cache block containing the data that C accessed

helpful! | 2

Resolved Unresolved

Anonymous Calc 1 month ago [SU-Final]:Q11.9

Why does the PPN of the page table entry corresponding to VPN 0x2 change, and why does the entry corresponding to VPN 0x1 change at all?

helpful! | 0

Jenny 1 month ago The 4 physical pages are all being mapped in the initial page table because there are 4 valid entries in the page table. With memory request 0x2F4, the VPN is 0x2, but the page table entry is invalid, meaning there is no mapping exists for that VPN in the current page table, so a page fault occurs. Therefore, we need to evict a page from physical memory to disk and use that page as a free page for Virtual page 0x2, and invalidate the previous entry from the page table. Because the question says: assume we evict from main memory and the TLB by evicting the smallest VPN. Therefore, we evict the physical page that's mapped by VPN 0x01, which is 0x2 and assign that to the virtual page 0x2.

helpful! | 1

Resolved Unresolved

Anonymous Mouse 1 month ago

[SP-FINAL]

I understand how they get the VPN, but does anyone understand how they get the PPN from the VPN? I don't get how to use the given page table.

Solution:

Virtual Address	Virtual Page Number	Physical Page Number	Physical Address	TLB Hit, Page Table Hit, Page Fault?
0x10	0x1 = 0b00 01	0x12	0x120	Page Table Hit
0x5C	0x5 = 0b01 01			Page Fault
0x39	0x3 = 0b00 11	0x5C	0x5C9	Page Table Hit
0x1F	0x1 = 0b00 01	0x12	0x12F	TLB Hit

TLB:

VPN	PPN
0x1 → 0x2	0x12 → 0x9
0x3 → 0x1	0x5C → 0x12

helpful! | 0

Jenny 1 month ago Let's walk through the process:
virtual address = 0b0001 0000 → VPN = 0001
TLB empty, so go straight to page table
L1 index: 00
Entry with index 0: data at address 0x00
M[0x00] = 0x20, so L2 table starts there
L2 index: 01

Entry with index 1: data at address 0x24
M[0x24] = 0x12 → PPN

VA to PA: 0x10 to 0x120
Add entry entry to TLB: 0x1 → 0x12
helpful! | 1

Resolved Unresolved

Anonymous Comp 2 1 month ago
[SU-F]:Q5 For part I, i understand that we want to check rd == rs1 for FwdOutA and rd == rs2 for FwdOutB, but don't we want to check that the rd for the instruction at the MWB stage is equal to either rs1 or rs2 for the instruction at the X stage? If so, how can we know that indexing into inst[11:7] will retrieve the rd of the instruction at the MWB stage and indexing into inst[19:15] will retrieve the rs1 for the instruction at the X stage?
helpful! | 0

Resolved Unresolved

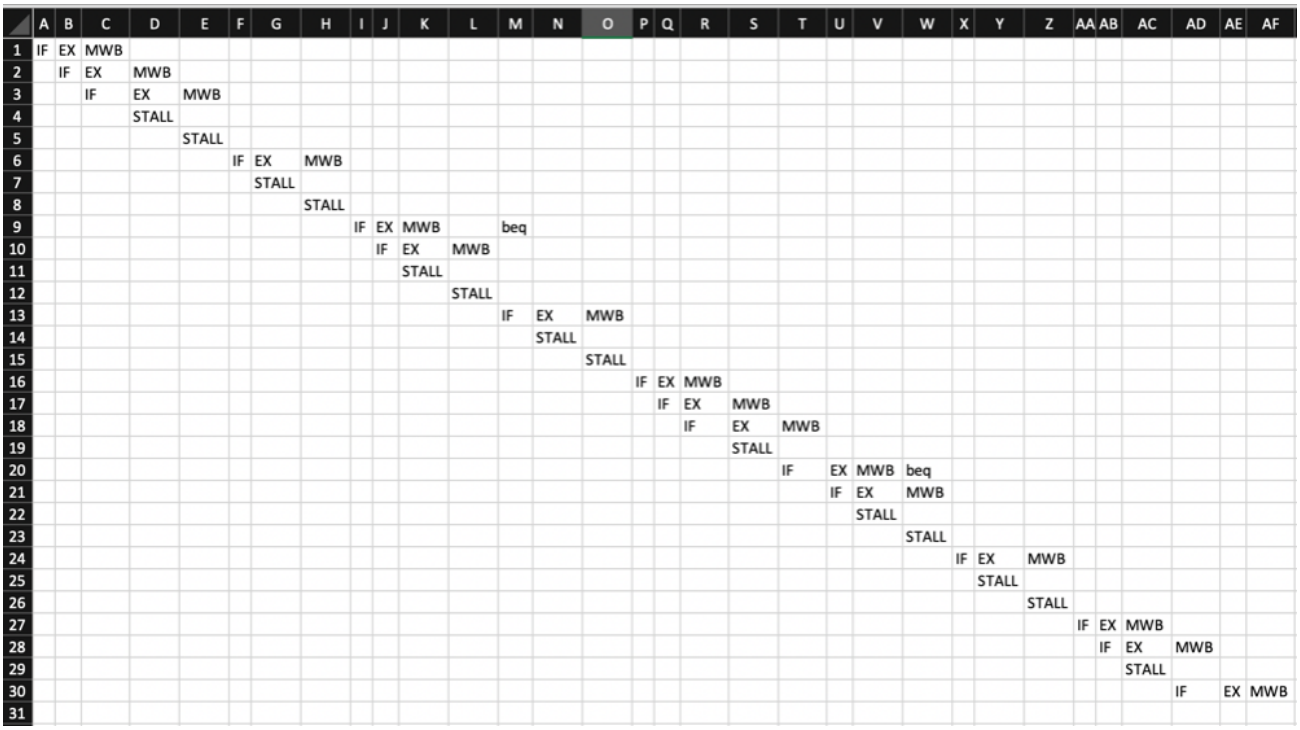
Anonymous Beaker 1 month ago
[SU-MT2]:Q4 - 2)b

Consider the following piece of code

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

b) How many **total cycles** will it take to complete the code above? Assume the pipeline is cleared upon reaching End. In addition, assume that there is perfectly accurate branch and jump prediction in the IFD stage. Thus, **ONLY CONSIDER STALLS DUE TO DATA HAZARDS. ASSUME THERE ARE NO STALLS FOR STRUCTURAL OR CONTROL HAZARDS.** Remember to **calculate the total number of cycles for fully executing the code**. A workspace was provided with this exam for you to show your work, which will only be graded if your final answer is not correct.

Total Cycles: _____



AF = 32 cycles

I don't understand why the correction says 33 cycles. Did I miss something ?

helpful! | 0

Cynthia Zhong 28 days ago Hey Antoine, if you're still stuck on this, check out the gsheet I made here: https://docs.google.com/spreadsheets/d/1Vtyf3Rg1CbfAyk8YvnIkZu3ICQBHzo89IRhEHVk5hx4/edit?usp=sharing. On it I also included a quicker (but perhaps less safe) solution that doesn't require drawing out the entire datapath. I also got 32 cycles originally, because I missed a stall somewhere lol. Looks like you missed an instruction (either the j or the addi) in the last couple of lines.

helpful! | 0

Resolved Unresolved

Anonymous Comp 2 1 month ago

[SU-MT2] Q5.3: Why can we use AMAT to calculate the number of cycles spent accessing memory? My understanding was that the time spent accessing memory was 2048 (num accesses) * 1/8 (L1 miss rate) * 1/4 (L2 miss rate) = 3 * 2^11?

helpful! | 0

Max Litster 27 days ago AMAT is Average Memory Access Time. The total comes from multiplying the average time per access (AMAT) by the total number of accesses (2048).

Time spent accessing memory includes (somewhat unintuitively) the time spent missing in the cache and triggering a memory access. AMAT accounts for these cycles.

helpful! | 0

Resolved Unresolved

Anonymous Atom 29 days ago

[Su-Final] #6: I'm a little confused as to how the hit rate for B is 27/32 . I thought we would miss B every four iterations of j, which would make it 3/4, just like C. I understand the logic for A and C, just stuck on B.

helpful! | 0

Anonymous Gear 28 days ago I feel like since i = 1 - 3, B will miss twice per iteration, because 0-3 will store in set 0, 4-7 store in set 1, 7-11 store in set 0 and 12-15 store in set 1. Thus, in per iteration, old B's data will be only be evicted twice since C also store the data in the same way.

helpful! | 0

Resolved Unresolved

Anonymous Gear 2 29 days ago

[Su-Final] #6 (2): The question states that execution is from left to right. Does this mean the access order is "CAB" or "ABC"?

helpful! | 0

Max Litster 27 days ago First we read from A and B, and then the result is written to C. So the access order is is ABC. We don't access C until we write to it.

We're not doing anything with the old value of C[j + i * N], just assigning a new value to that location. So we don't go to the cache until we actually have to write that new value to C[j + i * N].

helpful! | 0

Resolved Unresolved

Anonymous Scale 2 28 days ago

[SU-Final] #11 [3-7]. I am a bit confused as how to approach this problem. None of the addresses listed seem to match up with the page table, is there a certain order we have to look at the bits in?

helpful! | 0

Anonymous Atom 2 28 days ago Notice the offset is 12 bits so VM address 0x7ABC has 0x7 as VPN, 0xABC as offset. Look for VPN in the TLB and PAGE TABLE you should be able to solve the problem.

helpful! | 0

Resolved Unresolved

Anonymous Poet 2 28 days ago

[SP-Final] #12

part a) why is it fully associative?

part b) why is the block size 16Kib? Is a block the same size as a page?

part d e) why is it write back and write allocate? I thought we're directly writing to memory since the cache is memory itself.

thx in advance!

helpful! | 0

Max Litster 27 days ago If we think of traditional caching, we have cache data caching main memory. This question is asking about the same concept, just tweaked slightly: our "cache" is now memory, and "memory" is now the disk. This is still *caching*, but from RAM -> Disk as opposed to Cache -> RAM.

Let's consider Part B: Under the definition above, a block is the small piece of the disk that we pull into our cache, RAM. It functions just like a cache block: if a certain page is accessed in memory, and isn't found, we go to disk and pull it and the data around it (the page/frame) into memory. Therefore, thinking of RAM as our "cache" we have a block size of 16KiB.

Part A: Why is it fully associative? When I pull a page from disk into RAM, I don't map it to any index or set. Similarly, I only kick a page out of RAM when my cache is full, meaning that I only have

capacity or compulsory misses. This is a feature of fully-associative caches.

With these factors in mind, let's think about Part C. While yes, we are writing directly to memory because the cache is memory itself, but memory is not what's being cached here. What's stored in our cache is data from *disk*. So therefore, we have a write-back cache because I don't write back to the disk - the data I'm caching - until that block (or page in this case) is booted from the cache. The write-allocate argument follows similarly.

Hope this helped! Let me know if you have any further questions.

helpful! | 0

Resolved Unresolved

Anonymous Atom 2 28 days ago

[SU-Final] #10.3

For option D, is because it does not optimized the code? Or is because the statement is false about false sharing? Can I get a clarification about the answer?

helpful! | 0

Anonymous Helix 27 days ago False sharing happens when there is a write and the block gets invalidated. In this code, there is no writes and invalidation happening so there is no optimization.

helpful! | 0

Resolved Unresolved

Anonymous Helix 27 days ago

SU-Final #10.1

For option D, how could this code work in this case? I think it is an infinite loop since t2 = old value +1 and t1 = old value. The only case that is can be true is that t1 gets incremented meaning that it has gone through amoswap but in that case it is also going through infinite loop.

helpful! | 0

Anonymous Helix 2 27 days ago I think option D is true when the operations happen to occur in the correct order, and t2 == t1 after the first iteration (therefore it will not enter the infinite loop).

helpful! | 0

Resolved Unresolved

Anonymous Comp 2 27 days ago

[SP-MT2] Q1f: For IMM_20, why don't we have any assembly code to perform (offset & 0x100000) (which is andi a5, a5, 0x100000 in assembly) after the srai a5, a1, 20 operation? Without this mask, bits 19 - 0 of a5 aren't guaranteed to be 0s, which means when we try to `or` those bits later on things won't work properly.

helpful! | 0