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

note @2127 272 views

Actions

[Past Exams] 2020

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

Summer 2020 midterm 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: **SP20-Final-Q1**, or **FA20-MT2-Q3**

FA20-MT1 Video Walk Through Links
Bit Manipulations Walkthrough
Slip Walkthrough

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 @2127_f1

This was marked a duplicate to the question/note above by Peyrin Kao 5 months ago
Anonymous Comp 5 months ago
[sp20-final] questions
There's no solution for q5a.iii. Does this answer make sense? I'm not really sure what does the pointer in the instruction "if the pointer is currently pointing at a 0" referring to

```
(memory_grid[x][y] == 0 && dir==1)? dir=0 : dir=1
```

helpful! 0

Adelson Chua 5 months ago
Almost correct.
You only change dir from forward (1) to backward (0), and not vice versa.
good comment 0

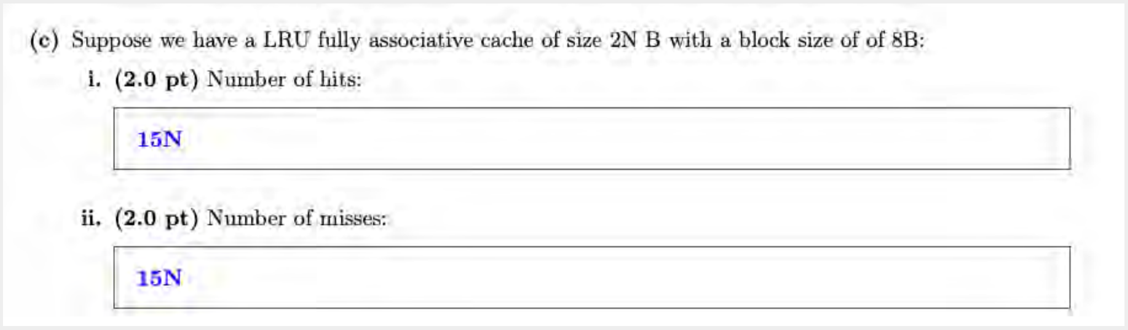
Anonymous Comp 5 months ago
How to write that logic in one line? We can't really have two ?: groups right?
helpful! 0

Adelson Chua 5 months ago

```
dir |= (memory_grid[x][y] == 0);
```

good comment | 0

Anonymous Comp 5 months ago
[q2c] Why is this 15N and 15N?


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

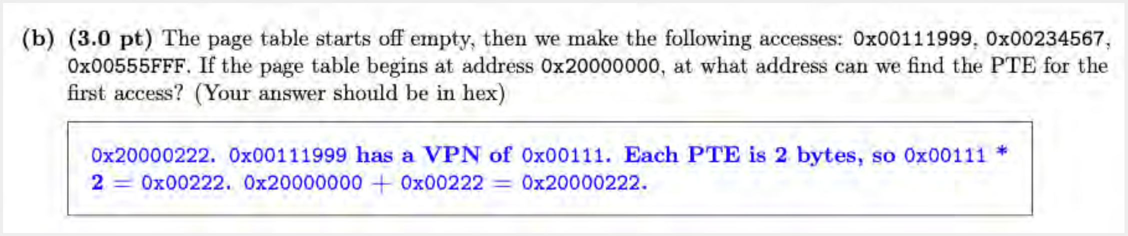
i. **(2.0 pt)** Number of hits:

15N

ii. **(2.0 pt)** Number of misses:

15N

[q4b] Why is each PTE 2 bytes?


(b) **(3.0 pt)** The page table starts off empty, then we make the following accesses: 0x00111999, 0x00234567, 0x00555FFF. If the page table begins at address 0x20000000, at what address can we find the PTE for the first access? (Your answer should be in hex)

0x20000222. 0x00111999 has a VPN of 0x00111. Each PTE is 2 bytes, so 0x00111 * 2 = 0x00222. 0x20000000 + 0x00222 = 0x20000222.

helpful! | 0

Adelson Chua 5 months ago
Regarding the cache:
@1774

Regarding the PTE, it was computed in 4a that PTE is 16 bits = 2 bytes.

good comment | 0

Anonymous Helix 2 5 months ago
for q4b, why do we multiply the VPN by the PTE?

helpful! | 0

Adelson Chua 5 months ago
Memory is byte-addressable. Each PTE is 2 bytes. VPN would correspond to the number of rows in the page table. Every row contains 1 PTE.

good comment | 0

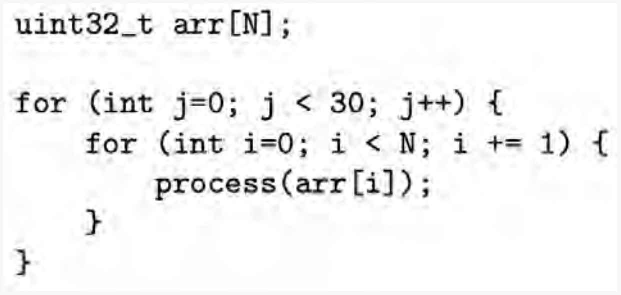
Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f2

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

Anonymous Gear 5 months ago

Sp20 Final Q2



```
uint32_t arr[N];

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

What are the number of hits + what type of locality are the following caches taking advantage of? All caches start out empty:

1. LRU fully associative cache of size 4N B and a block size of 4B

2. LRU fully associative cache of size 2N B and a block size of 4B

3. LRU fully associative cache of size 2N B and a block size of 8B

I would like to confirm that my thinking process is correct for the first two caches and am also confused on the answer for the third cache.

Block size == line size, which means that there is one 32-bit integer (4B) stored in each line of the first and second caches. Because there are N lines in the first cache (4N / 4), we have N integers that are able to be stored. Since the inner for loop repeats 30 times, we only need to store a total of N integers which can be repeatedly accessed each of the 30 times. So, for the first cache, there are N misses due to the first N compulsory misses and 29N hits. It takes advantage of temporal locality due to the LRU policy. For the second cache, because there are only half the number of cache lines available, we would continually need to replace elements in the cache based on LRU, which means we have 0 hits and 30N misses. This also means the second cache doesn't take advantage of any type of locality.

But for the third cache, I thought that each line could store 2 integers, which makes up for the cache having only half the number of lines as the first cache. I thought this meant that we would also only have N misses, but the answer says there are 15N misses and 15N hits.

Could someone explain if my thinking for the first and second caches are correct, and why the third one is as is? Thank you!

helpful! | 0

Adelson Chua 5 months ago

The explanation for 1st cache architecture is correct. The cache fits the whole array, 1st of the 30 iterations are compulsory misses, next 29 are hits.

The explanation for 2nd cache is also correct. The cache cannot fit the whole array, there will be replacements when you access the 2nd half of the array, thus there will be no hits afterward.

For 3rd cache, for every miss, you get one hit (because the block size is 2 ints, 1 int for the one you missed, and the next int after that). So that's always 50% miss 50% hit.

good comment | 0

Anonymous Gear 5 months ago

I understand now that there would be a 50-50 miss/hit, but wouldn't this happen only for the 1st of the 30 iterations leading to N/2 misses? Because there wouldn't need to be any replacements given that the cache can hold all N elements?

helpful! | 0

Adelson Chua 5 months ago
Cache size is 2N though. So it can only fit half of the array. There will be replacements, just like in cache 2.
good comment | 0

Anonymous Gear 5 months ago
Ohh right, got it! Thank you so much!
helpful! | 0

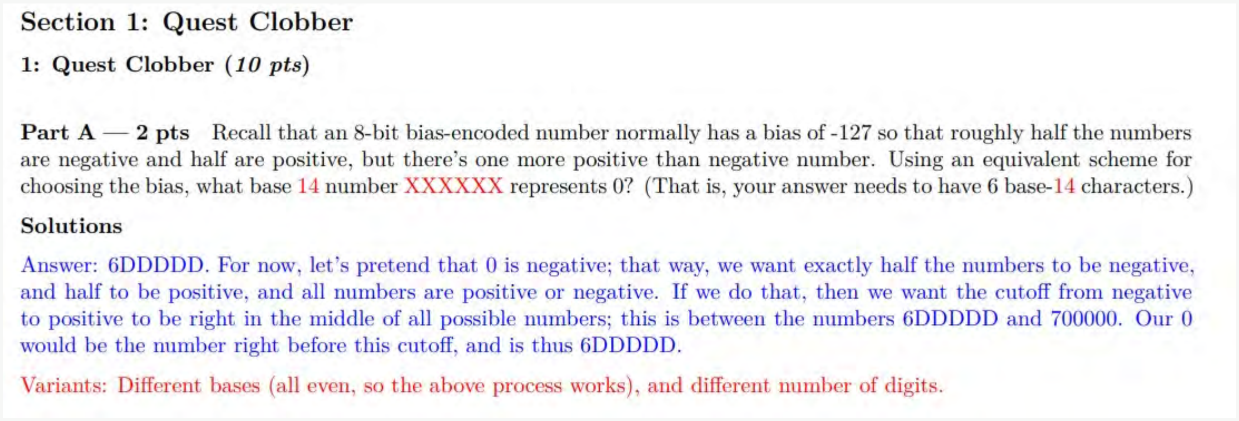
Anonymous Comp 3 5 months ago
What is Quasi-balistic locality?
helpful! | 0

Adelson Chua 5 months ago
Not sure. It's not covered in the lecture anyway. Just an added choice.
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f3

Anonymous Scale 5 months ago
FA20-Final-S1-Q1A question



Section 1: Quest Clobber

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.

I understand that the bias would be -(14^(6-1) - 1) -(DDDDD) based on the equation and the concept that 0 would be in between the cutoff of negative and positive numbers. However, I do not understand why 6DDDDD is 0's biased representation when on discussion 01 on part 3 0 for an 8 bit-biased notation with biased -127 is 127.
helpful! | 0

Adelson Chua 5 months ago
I do not know what you are trying to connect between this exam question and the discussion. This exam question is in 6-digit base-14. The discussion is in 8-bit binary.
good comment | 0

Anonymous Scale 5 months ago
Sorry for the confusion. I am just confused on where did the 6 came from in representation of 0 in

the 6-digit-base-14.

helpful! | 0

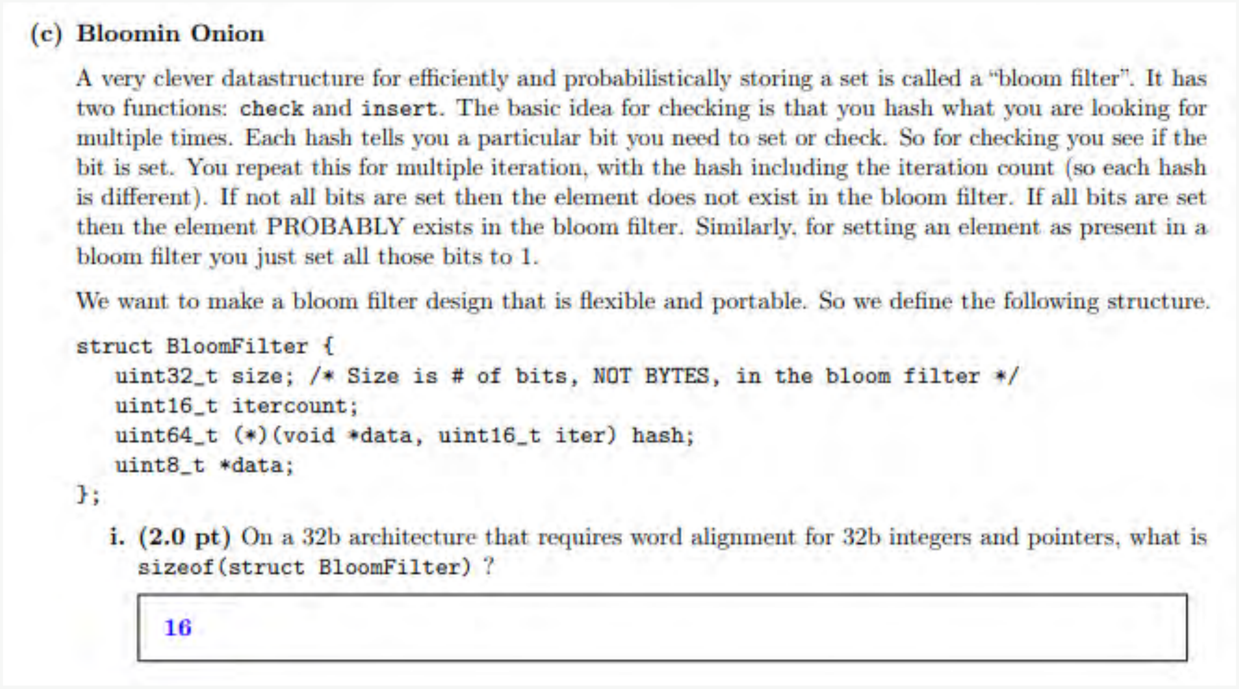
Adelson Chua 5 months ago
That's about halfway between 0 and D (the range of 1 digit in base 14).

good comment | 1

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f4

Anonymous Poet 5 months ago
I can't see why this answer is correct, my understanding is that we would need 2 bytes of padding after itercount to align the function pointer to a word boundary which would put the size at least 17 bytes?



(c) Bloomin Onion

A very clever datastructure for efficiently and probabilistically storing a set is called a "bloom filter". It has two functions: `check` and `insert`. The basic idea for checking is that you hash what you are looking for multiple times. Each hash tells you a particular bit you need to set or check. So for checking you see if the bit is set. You repeat this for multiple iteration, with the hash including the iteration count (so each hash is different). If not all bits are set then the element does not exist in the bloom filter. If all bits are set then the element PROBABLY exists in the bloom filter. Similarly, for setting an element as present in a bloom filter you just set all those bits to 1.

We want to make a bloom filter design that is flexible and portable. So we define the following structure.

```
struct BloomFilter {
    uint32_t size; /* Size is # of bits, NOT BYTES, in the bloom filter */
    uint16_t itercount;
    uint64_t (*)(void *data, uint16_t iter) hash;
    uint8_t *data;
};
```

i. **(2.0 pt)** On a 32b architecture that requires word alignment for 32b integers and pointers, what is `sizeof(struct BloomFilter)` ?

16

helpful! | 0

Peyrin Kao 5 months ago
`size` is 32 bits = 4 bytes.

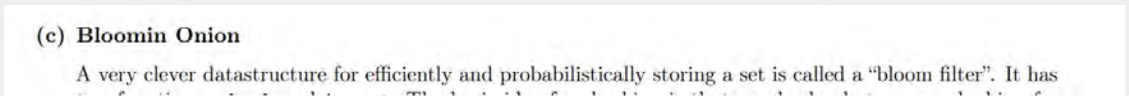
`itercount` is 16 bits = 2 bytes. Add 2 bytes of padding to hit a word-aligned boundary.

`hash` is 4 bytes since it's a pointer to a function.

`data` is 4 bytes since it's a pointer to a `uint8_t`.

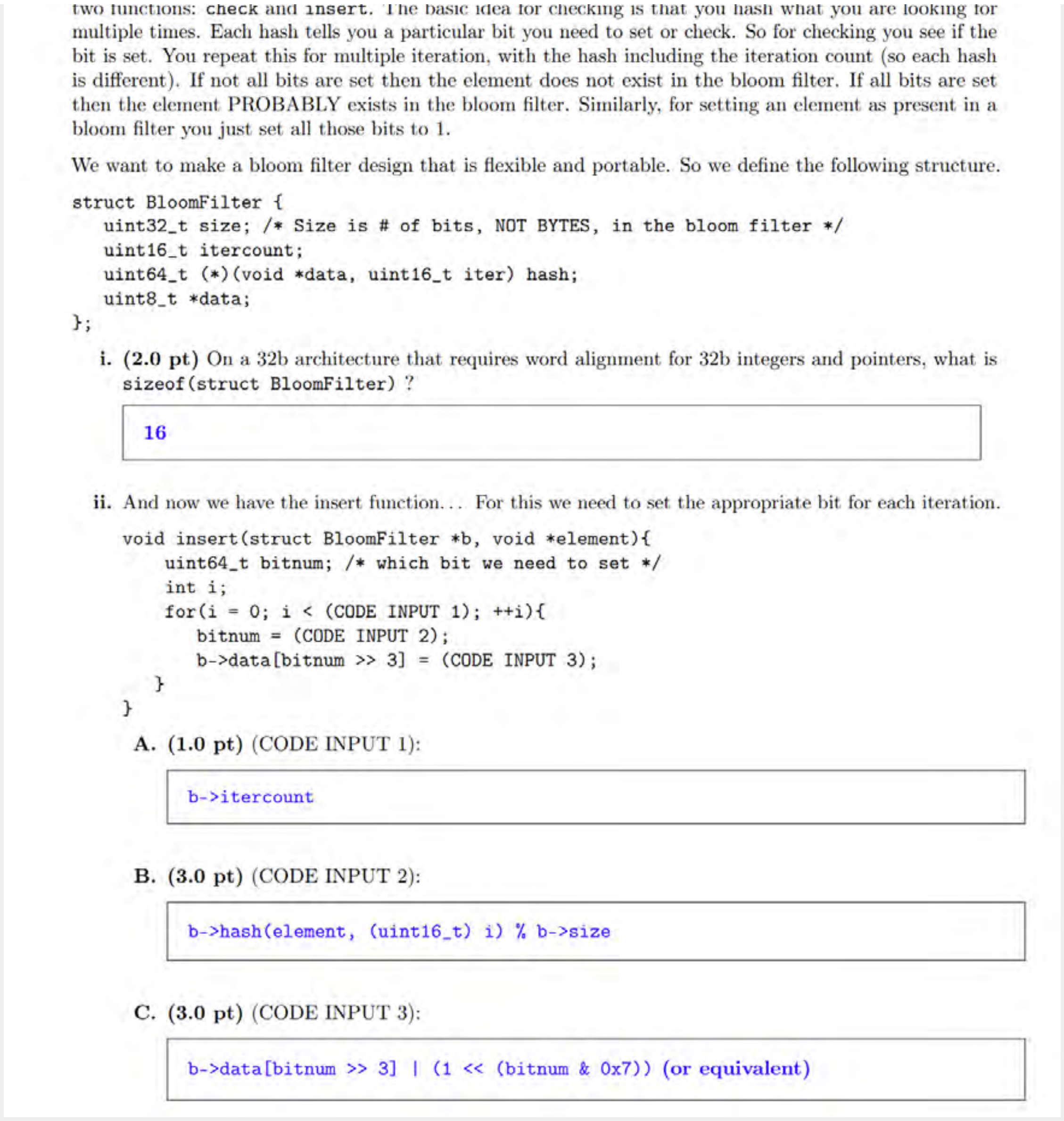
good comment | 1

Anonymous Poet 5 months ago
I also don't understand why we need to mod by b->size or the final answer...



(c) Bloomin Onion

A very clever datastructure for efficiently and probabilistically storing a set is called a "bloom filter". It has



two functions: `check` and `insert`. The basic idea for checking is that you hash what you are looking for multiple times. Each hash tells you a particular bit you need to set or check. So for checking you see if the bit is set. You repeat this for multiple iteration, with the hash including the iteration count (so each hash is different). If not all bits are set then the element does not exist in the bloom filter. If all bits are set then the element PROBABLY exists in the bloom filter. Similarly, for setting an element as present in a bloom filter you just set all those bits to 1.

We want to make a bloom filter design that is flexible and portable. So we define the following structure.

```
struct BloomFilter {
    uint32_t size; /* Size is # of bits, NOT BYTES, in the bloom filter */
    uint16_t itercount;
    uint64_t (*)(void *data, uint16_t iter) hash;
    uint8_t *data;
};
```

i. (2.0 pt) On a 32b architecture that requires word alignment for 32b integers and pointers, what is `sizeof(struct BloomFilter)` ?

16

ii. And now we have the insert function... For this we need to set the appropriate bit for each iteration.

```
void insert(struct BloomFilter *b, void *element){
    uint64_t bitnum; /* which bit we need to set */
    int i;
    for(i = 0; i < (CODE INPUT 1); ++i){
       bitnum = (CODE INPUT 2);
       b->data[bitnum >> 3] = (CODE INPUT 3);
   }
}
```

A. (1.0 pt) (CODE INPUT 1):

`b->itercount`

B. (3.0 pt) (CODE INPUT 2):

`b->hash(element, (uint16_t) i) % b->size`

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

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

helpful! | 0

Peyrin Kao 5 months ago

The hash outputs a `uint64_t`, but the `data` array only has `b->size` elements, so we need to use the mod operator to convert the hash output into one of the elements in the array of hashes.

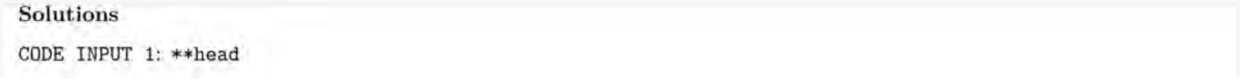
`data` is an array of bytes (8 bits each), and we want to set a specific bit given by the index `bitnum`. `bitnum >> 3` gives all but the lowest 3 bits of the index, which are used to choose what byte we want to set. `bitnum & 0x7` gives us only the lowest 3 bits of the index, which are used to choose which bit within the byte we want to set.

good comment | 0

Reply to this followup discussion

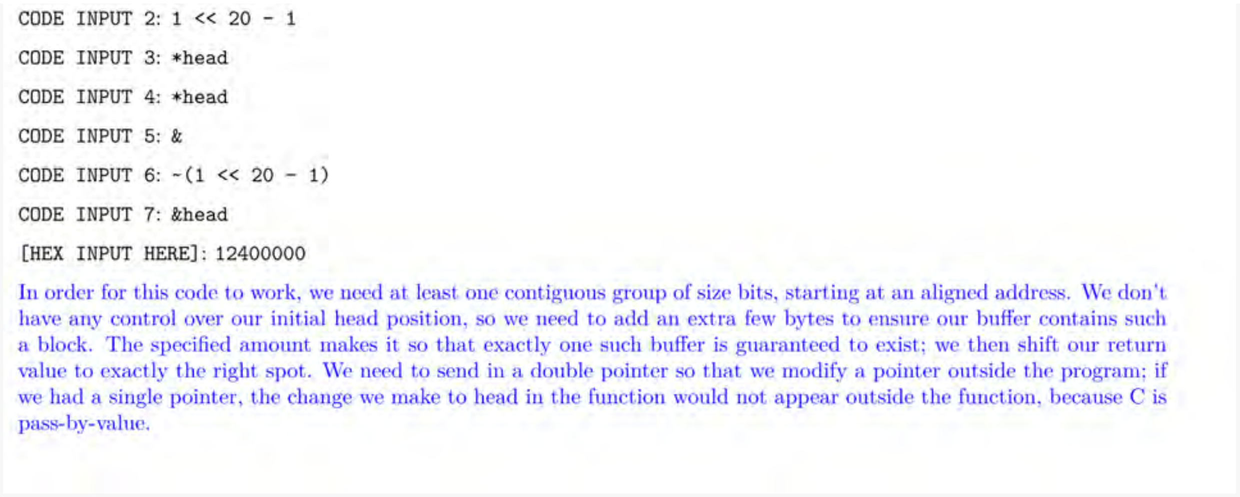
[✓ Resolved] Unresolved @2127_f5

Anonymous Atom 5 months ago



Solutions

```
CODE INPUT 1: **head
```



```
CODE INPUT 2: 1 << 20 - 1
CODE INPUT 3: *head
CODE INPUT 4: *head
CODE INPUT 5: &
CODE INPUT 6: ~(1 << 20 - 1)
CODE INPUT 7: &head
[HEX INPUT HERE]: 12400000
```

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

For fa20 final question 1B, how is 1 << 20 -1 change to a memory address aligned to 2 20-byte boundaries?

helpful! | 0

Anonymous Atom 5 months ago

also how we get 12400000?

helpful! | 0

Adelson Chua 5 months ago

1 << 20 is basically 2^20.

There's a -1 to ensure a malloc'd size of 1 byte will still get 2^20 allocated.

0x12400000 might have been a typo. I believe it should be 0x12300000

good comment | 0

Anonymous Atom 5 months ago

then how we get 0x12300000?

helpful! | 0

Adelson Chua 5 months ago

The return value is calculated as follows:

0x12345678 & ~(1 << 20 -1) // this is 1 followed by 20 zeros minus 1
0x12345678 & ~(0x0007FFFF) // which is 19 1s
0x12345678 & (0xFFF80000) // bitwise negate
0x12300000 // bitwise AND

It might help to review boolean logic.

good comment | 0

Anonymous Gear 5 months ago

When it says "we want to malloc the *fewest extra bytes possible*", why is the fewest extra bytes 2^{20} byte boundaries? I thought it meant we would need to offset a certain amount to fill the boundary given the size passed in.

helpful! | 0

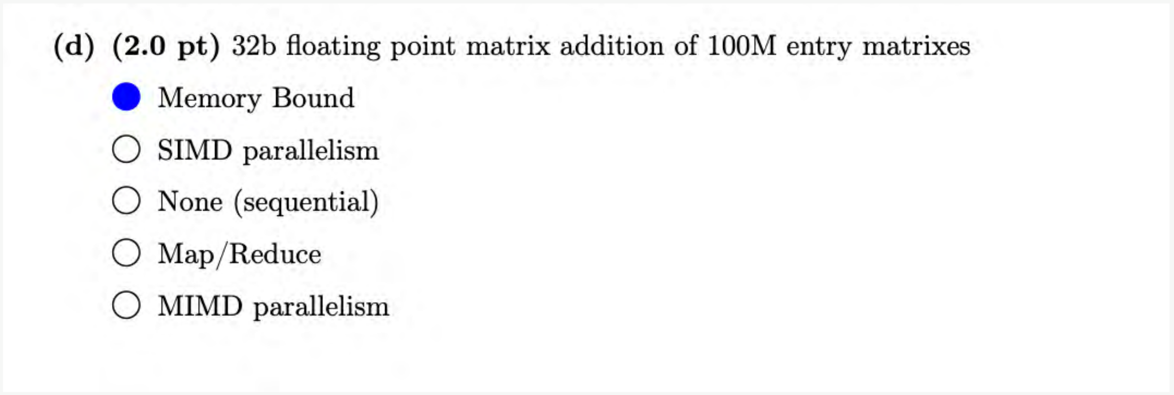
Adelson Chua 5 months ago
The problem requires that the allocated block need to be at least 2^20 to align to the boundary. At the minimum, we allocate just 1 byte, so we add 2^20 - 1 to force the alignment.
good comment | 0

Anonymous Gear 5 months ago
Ohh got it, thank you!
helpful! | 0

Reply to this followup discussion

Resolved Unresolved @2127_f6

Anonymous Helix 5 months ago
Sp20 Final Q3



(d) (2.0 pt) 32b floating point matrix addition of 100M entry matrixes

● Memory Bound
○ SIMD parallelism
○ None (sequential)
○ Map/Reduce
○ MIMD parallelism

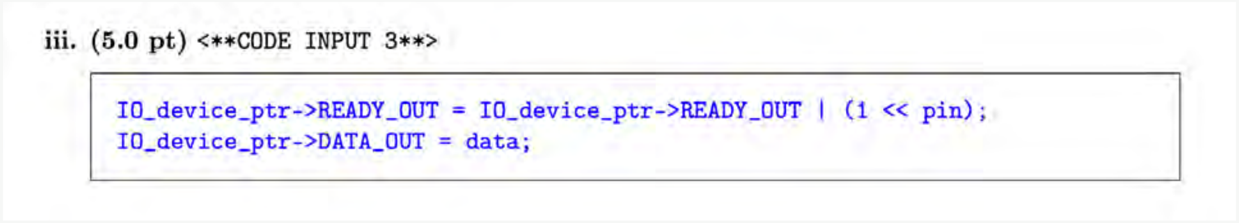
Could someone please explain memory bound a little bit (maybe what lecture it's from?) and also why it's the correct answer here? Thank you!
helpful! | 0

Peyrin Kao 5 months ago
Memory bound is out of scope this semester.
good comment | 1

Reply to this followup discussion

Resolved Unresolved @2127_f7

Anonymous Atom 5 months ago



iii. (5.0 pt) <**CODE INPUT 3**>

```
IO_device_ptr->READY_OUT = IO_device_ptr->READY_OUT | (1 << pin);
IO_device_ptr->DATA_OUT = data;
```

for sum 20 final, question 12b iii, why io_device_ptr -> read_out | instead of just a if statement like
(if data != null) {
io_device_ptr -> read_out = 1;

} else {
io_device_ptr -> read_out = 0;
}

?

helpful! 0

Adelson Chua 5 months ago
Your proposed if statement block will only work for pin = 0.
... actually maybe not, because I don't know why you are check for data != null. What is data?

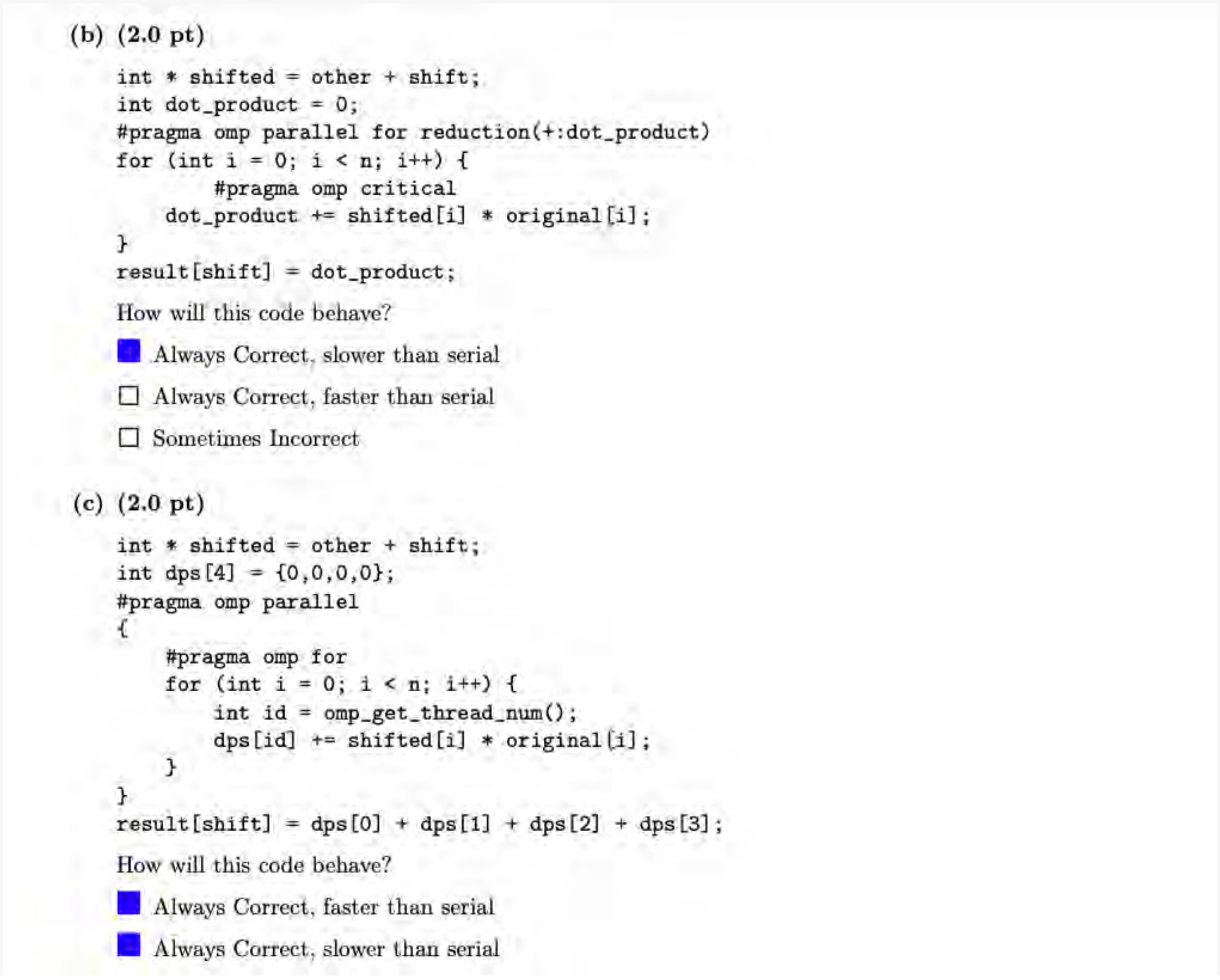
The (1<<pin) is important since the READY_OUT is a collection of bits corresponding to different pin numbers. See the definition of this in the main problem statement.

good comment 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f8

Anonymous Mouse 5 months ago
Can you explain how all these multiple choice questions were selected. I literally got a 0 on this whole question which is statistically impressive but also means I really had 0 clue what was going on. There is no explanations for the solutions.



(b) (2.0 pt)

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

How will this code behave?

[✓ Always Correct, slower than serial]

☐ Always Correct, faster than serial

☐ Sometimes Incorrect

(c) (2.0 pt)

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

How will this code behave?

[✓ Always Correct, faster than serial]

[✓ Always Correct, slower than serial]

☐ Sometimes Incorrect

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

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

2. TLP

In signal processing, the technique of **cross-correlation** (or *sliding dot-product*) is often used to determine the delay of a signal. In this problem, we will implement a cross-correlation function in C, parallelized of course! (You don't need to know anything about EE to ace this problem!) :0

The following function, `sliding_dot`, takes two arrays, `original` contains an array of length n and `other` contains `n + k` elements. We will shift `other` `k` times, and for each shift, compute its dot product with `original`. We then store these values in `result`.

We want to parallelize `sliding_dot` with OpenMP. Examine our attempts below and choose the behavior(s) you expect from each version. Assume the processor has four threads, 32B cache blocks, and `sizeof(int) =` 4.You may also assume that all calls to calloc() succeed.

Here is the template of the code where we will replace the `/* OPTIMIZED CODE */` with the code on each question.

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

(a) (2.0 pt)

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

How will this code behave?

☐ Always Correct, faster than serial

[✓ Sometimes Incorrect]

☐ Always Correct, slower than serial

(d) (2.0 pt)

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

How will this code behave?

☐ Always Correct, slower than serial

☐ Always Correct, faster than serial

[✓ Sometimes Incorrect]

helpful! | 0

Peyrin Kao 5 months ago

(b) `dot_product` is being updated in a critical section, so the threads will wait to update it one at a time. This means there are no data races, so the code is correct. However, the time it takes to start up all the threads means that it would have just been faster to run that one line of code in the for loop serially.

(c) Every thread is writing to a different element of `dps` so there are no data races, so the code is correct. The speedup depends on whether all the elements of `dps` are in one cache block, which would lead to false sharing (as explained in the solutions, but feel free to follow up if you have specific questions).

(a) The `dot_product` is marked private, so every thread gets its own separate copy. This code is only correct if all the work runs on one thread; otherwise, there's nothing to specify how the different copies of `dot_product` get combined.

(d) `dot_product` is declared outside the for loop, so every thread writes to the same single copy of the variable. This is vulnerable to data races.

good comment | 2

Anonymous Calc 5 months ago

For (a), so for the last line `result[shift] = dot_product;` , the system will just randomly pick up a thread and use its `dot_product` value?

helpful! | 0

Peyrin Kao 5 months ago

I think it just uses the `dot_product` value before any of the threads get run. (In other words, the private variables are local to each thread and get deleted after the thread finishes.)

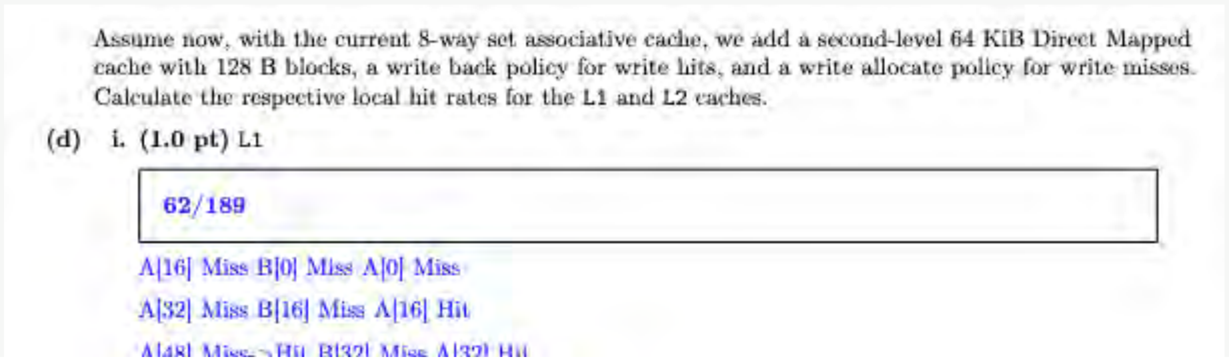
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f9

Anonymous Beaker 5 months ago

[Su20-MT2-4d]



Assume now, with the current 8-way set associative cache, we add a second-level 64 KiB Direct Mapped cache with 128 B blocks, a write back policy for write hits, and a write allocate policy for write misses. Calculate the respective local hit rates for the L1 and L2 caches.

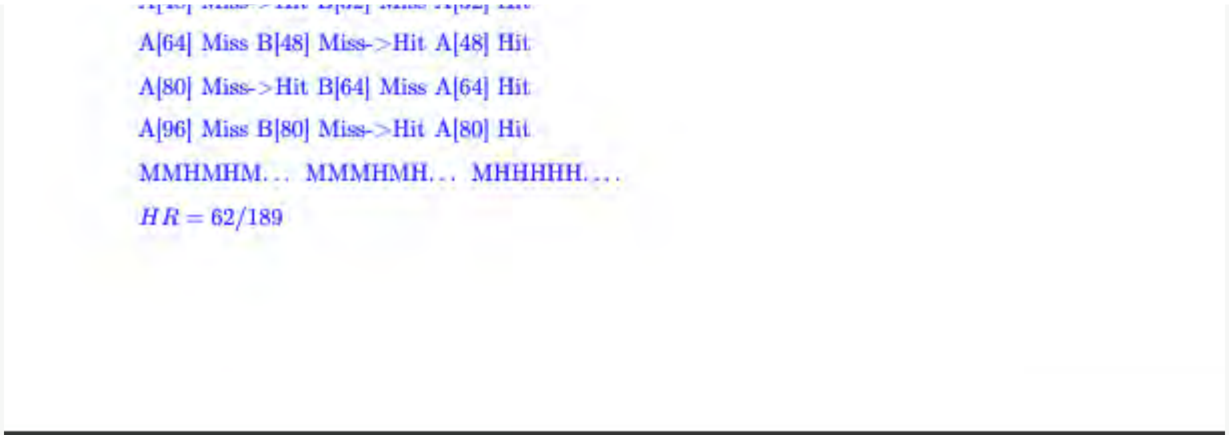
(d) i. (1.0 pt) L1

62/189

A[16] Miss B[0] Miss A[0] Miss

A[32] Miss B[16] Miss A[16] Hit

A[48] Miss->Hit B[32] Miss A[32] Hit

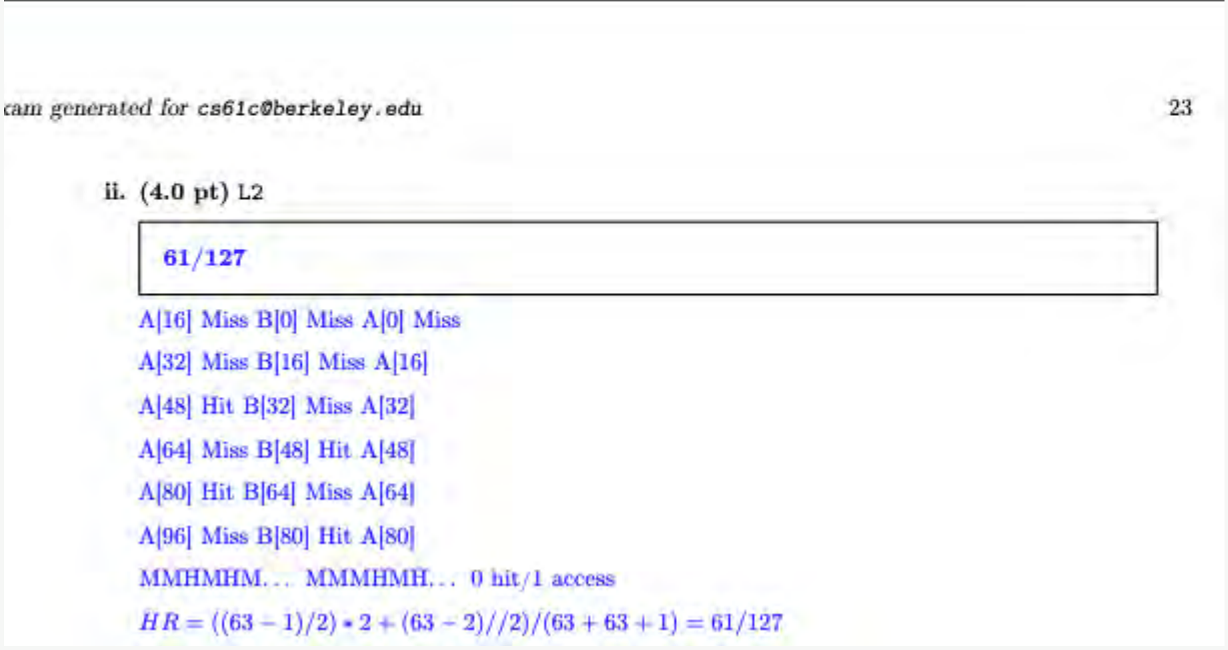

A[64] Miss B[48] Miss->Hit A[48] Hit

A[80] Miss->Hit B[64] Miss A[64] Hit

A[96] Miss B[80] Miss->Hit A[80] Hit

MMHMHM... MMMHMH... MHHHHH....

$HR = 62/189$



ii. **(4.0 pt)** L2

61/127

A[16] Miss B[0] Miss A[0] Miss

A[32] Miss B[16] Miss A[16]

A[48] Hit B[32] Miss A[32]

A[64] Miss B[48] Hit A[48]

A[80] Hit B[64] Miss A[64]

A[96] Miss B[80] Hit A[80]

MMHMHM... MMMHMH... 0 hit/1 access

$HR = ((63 - 1)/2) * 2 + (63 - 2)//2)/(63 + 63 + 1) = 61/127$

Can someone explain how the values are brought back into the cache? The L1 cache has 64B block and L2 has 128B block. So, if the L1$ were to miss, go to L2$ and also miss, would it bring back 32 integers into the L2$ and then 16 integers into the L1$?

According to the solution i part i, A[32] is a miss. But, I was thinking it'd be a miss->hit because the first time A[16] is accessed in the line above, it'd bring in 128 bytes into the L2$, which corresponds to A[16]-A[47]. Then shouldn't A[32] be a hit in the L2$? I know I'm messing up but I'm not sure where.

helpful! 0

Adelson Chua 5 months ago

would it bring back 32 integers into the L2$ and then 16 integers into the L1$?

Yes

When L2 missed on A[16], it brought back A[0]-A[31]. That's the whole cache block for L2.
The cache block does not always start on the address that you missed.

It might be helpful to write the corresponding addresses in binary then see what the index is. You bring in data from a single index.

good comment 0

Anonymous Beaker 5 months ago

Then wouldn't A[0] be a miss->hit in the first group of access? Since we access A[16] first, I'm assuming it brings A[16:31] into L1 and A[0:31] into L2. So, when we try A[0], it'd miss in L1 and hit in L2, right?

helpful! | 0

Adelson Chua 5 months ago

Yeah, that's a good point. That reasoning sounds correct.

A[16] Miss B[0] Miss A[0] Miss
A[32] Miss B[16] Miss A[16]
A[48] Hit B[32] Miss A[32]
A[64] Miss B[48] Hit A[48]
A[80] Hit B[64] Miss A[64]
A[96] Miss B[80] Hit A[80]
MMHMHM... MMMHMH... 0 hit/1 access

$HR = ((63-1)/2)*2 + (63-2)//2)/(63+63+1) = 61/127$

The solution isn't labeling the A[16] as anything (not hit or miss label).

I don't know what is going on with the math, but it's as if it is treating that as a miss? Not sure.

If it is, then I think there's an error in the solution.

good comment | 0

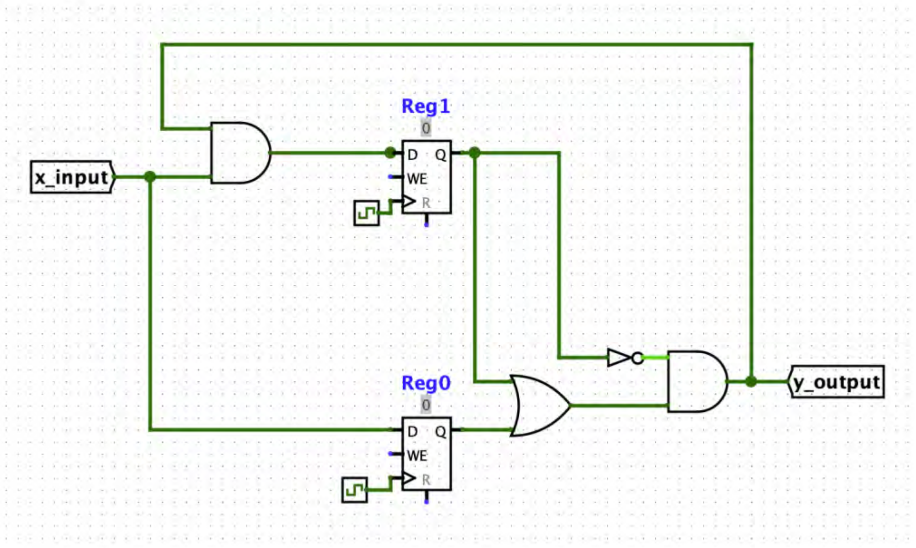
Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f10

Anonymous Beaker 5 months ago

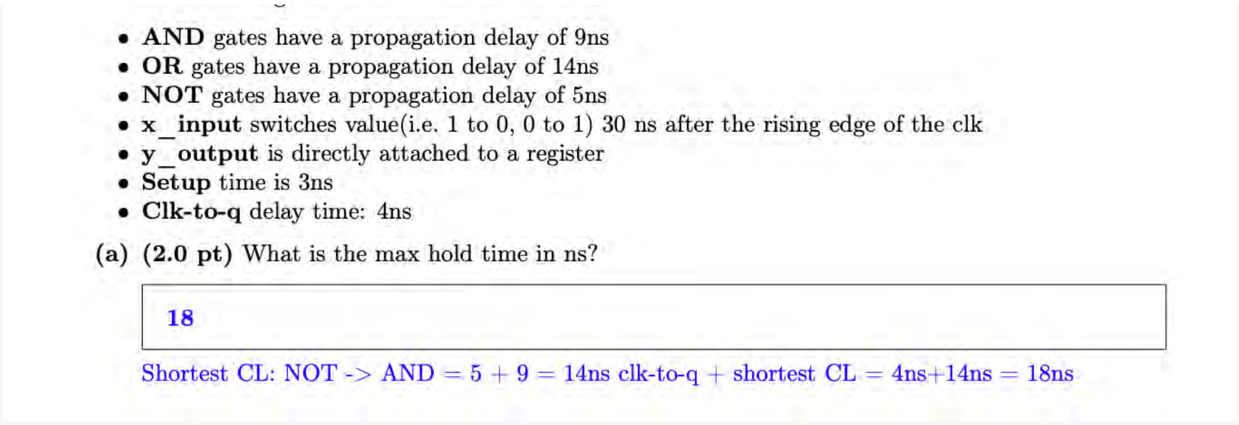
4. SDS, Logic

We will be analyzing the following circuit:



Circuit

Given the following information:



- **AND** gates have a propagation delay of 9ns
- **OR** gates have a propagation delay of 14ns
- **NOT** gates have a propagation delay of 5ns
- **x_input** switches value(i.e. 1 to 0, 0 to 1) 30 ns after the rising edge of the clk
- **y_output** is directly attached to a register
- **Setup** time is 3ns
- **Clk-to-q** delay time: 4ns

(a) **(2.0 pt)** What is the max hold time in ns?

18

Shortest CL: NOT -> AND = 5 + 9 = 14ns clk-to-q + shortest CL = 4ns+14ns = 18ns

How is the CL defined? Is it register to register, or does it just have to end at a register. I was thinking of x_input -> AND -> Reg1, which comes out to be 13ns

helpful! | 0

Anonymous Beaker 5 months ago

This is Su20-Final-Q4 btw. For 4c, why is the clock period (50ns) being added? I'm not sure what exactly clock period represents. Is it how long it takes for signal to be realized? I thought that was the c2q.

helpful! | 0

Adelson Chua 5 months ago

You start checking for combinational logic delay from a point that is clocked (typically a register, sometimes inputs are also presented as coming from a register) to another point that is clocked (which is usually another register, sometimes outputs are also presented as attached to a register).

In this case, y_output is attached to a register, x_input also comes from a register (however, with a c2q delay of 30ns).

What is now the shortest path?

The solutions say the path from Reg1 to y_output is the shortest. Nothing else is shorter than that. You proposed x_input -> AND -> Reg1 is longer because x_input is delayed by 30ns.

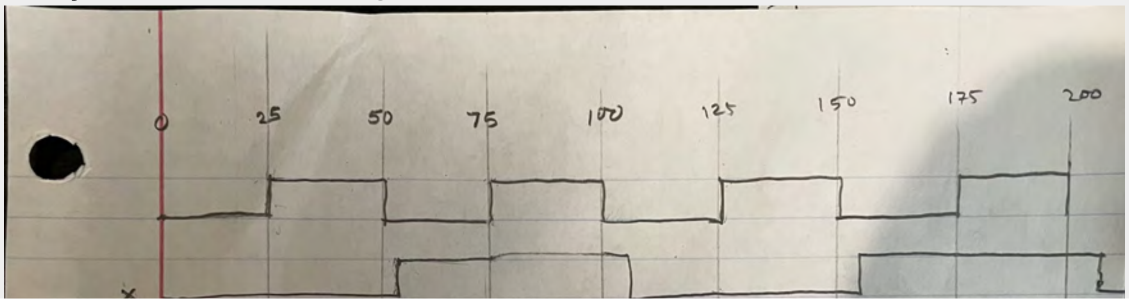
good comment | 1

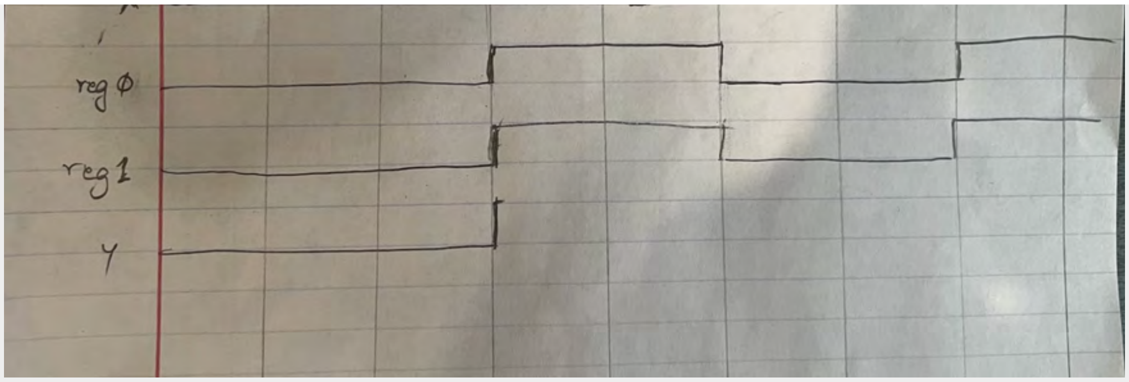
Adelson Chua 5 months ago

I'll update this follow-up to answer your next question. That requires some drawing.

good comment | 1

Anonymous Beaker 5 months ago





I did the timing diagram and got y to change it's value to 1 at 75 ns. I add in the clk2q and delays for the or and and gate, and get 102 ns. Is this the correct way of doing i?

helpful! | 0

Adelson Chua 5 months ago

Nice! Very nice! Yes this is correct. :)

good comment | 0

Anonymous Atom 2 5 months ago

why do reg 0 reg 1 and y change at 75? I get why x is changing at 55 but not how each of them change at 75 especially since theres logic gates betweenn the registers and y

helpful! | 0

Adelson Chua 5 months ago

The problem states that there is a 30ns delay from x_input.

Time step 75 is a rising edge of the clock, where the registers update their values.

good comment | 0

Anonymous Atom 2 5 months ago

so the registers depend on x input. I guess I"m still confused on how the logic gates play a part here. I get why reg 0 would take the x input, but would the and gate before reg 1 not add any propogation delay? and would the and gate not create a 0 for the input of reg1?

and why would y immediatly change as well? is there no clock to q for the register to output a value?

helpful! | 0

Adelson Chua 5 months ago

The reg0 and reg1 values shown there are the outputs of the registers, which only update during the rising edge of the clock.

The actual values of the wires does not matter for this problem actually, we are only interested in the delays.

Yeah, the y_output in the timing diagram is wrong, it should be delayed by the propagation time of the gates from the register to the y_output.

good comment | 1

Anonymous Atom 2 5 months ago

okay that makes a lot of sense. thank you!

helpful! | 0

Reply to this followup discussion

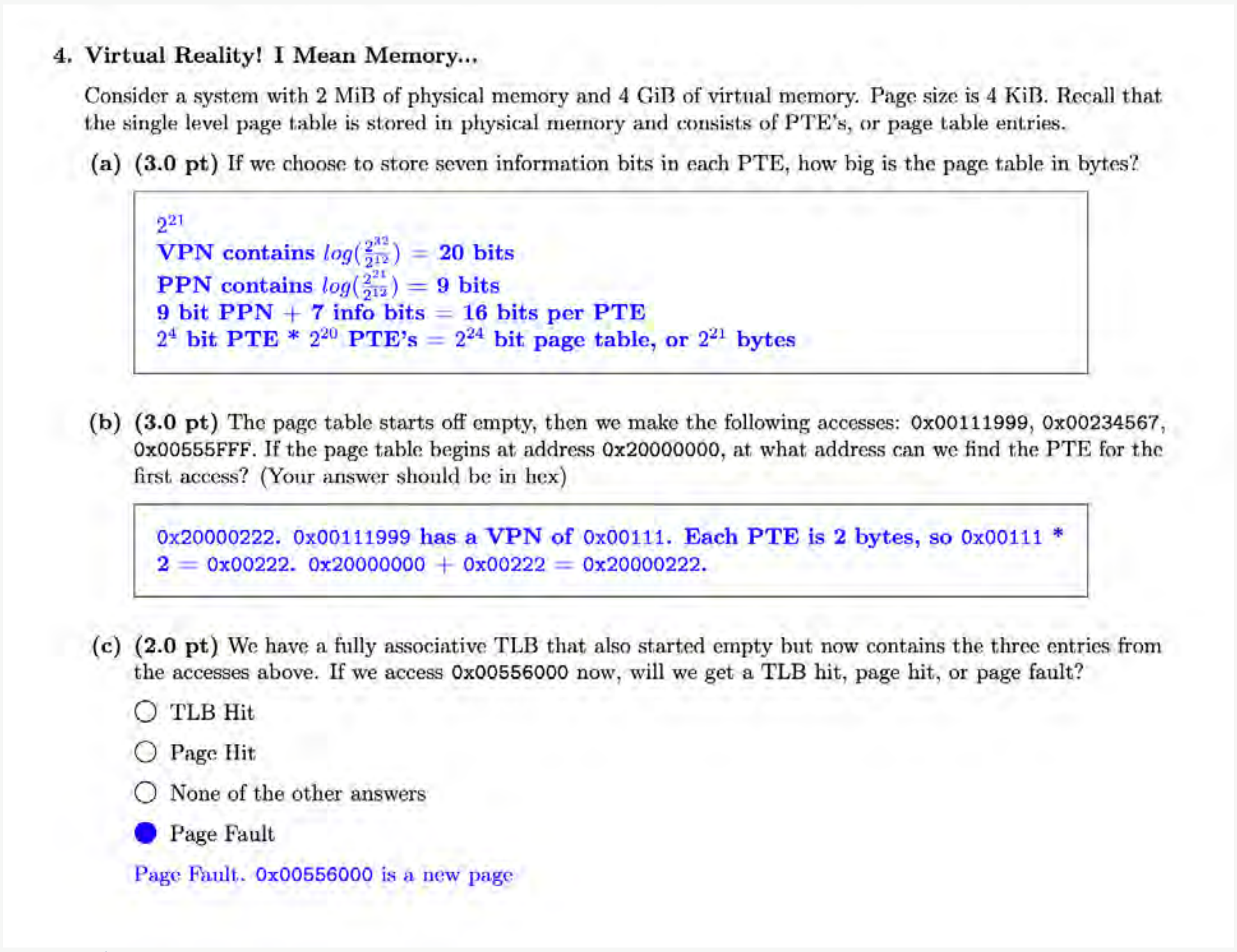
[✓ Resolved] Unresolved @2127_f11

Anonymous Calc 5 months ago

SP20-Final-Q4

for (a), (1)just to confirm, is it always true that #of PTE = #of VPN bits? 2) What is this 'information bit' and why we could replace PPN bits with information bits?

(c) how do we find it's a new page? I think it's related to page size, but I found it hard to transform page size into offsets on a single page.



4. Virtual Reality! I Mean Memory...

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

(a) **(3.0 pt)** If we choose to store seven information bits in each PTE, how big is the page table in bytes?

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

(b) **(3.0 pt)** The page table starts off empty, then we make the following accesses: 0x00111999, 0x00234567, 0x00555FFF. If the page table begins at address 0x20000000, at what address can we find the PTE for the first access? (Your answer should be in hex)

0x20000222. 0x00111999 **has a VPN of** 0x00111. **Each PTE is 2 bytes, so** 0x00111 * 2 = 0x00222. 0x20000000 + 0x00222 = 0x20000222.

(c) **(2.0 pt)** We have a fully associative TLB that also started empty but now contains the three entries from the accesses above. If we access 0x00556000 now, will we get a TLB hit, page hit, or page fault?

○ TLB Hit
○ Page Hit
○ None of the other answers
[✓ Page Fault]

Page Fault. 0x00556000 is a new page

helpful! | 0

Adelson Chua 5 months ago

(1)just to confirm, is it always true that #of PTE = #of VPN bits?

Yes, VPN indexes the page table. This is covered in the lecture with corresponding illustrations...

(2) What is this 'information bit' and why we could replace PPN bits with information bits?

Those are your valid/dirty/permission bits. Those don't replace the PPN, they are in addition to them. This is also shown in the lecture...

(c) how do we find it's a new page? I think it's related to page size, but I found it hard to transform page size into offsets on a single page

Check the VPN, it points to a new entry. Since the page table starts empty, that entry is not valid.

good comment | 0

Anonymous Comp 3 5 months ago

How do you know that 0x00111999 has a VPN of 0x00111?

helpful! | 0

Reply to this followup discussion

[✓ Resolved] Unresolved **@2127_f12**

Anonymous Comp 2 5 months ago

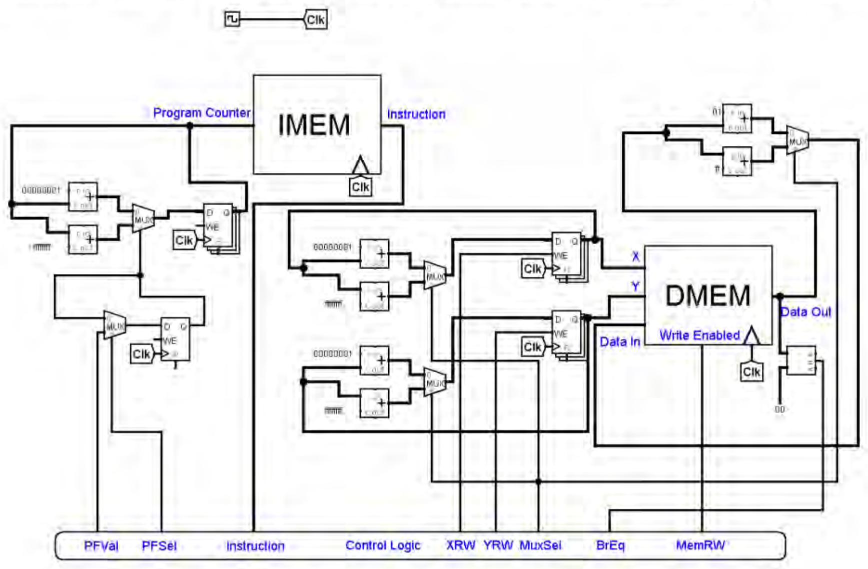
[Spring 2020 Final Q5 part 4]

I'm confused on how to come up with the solution. Could someone explain the intuition behind this?

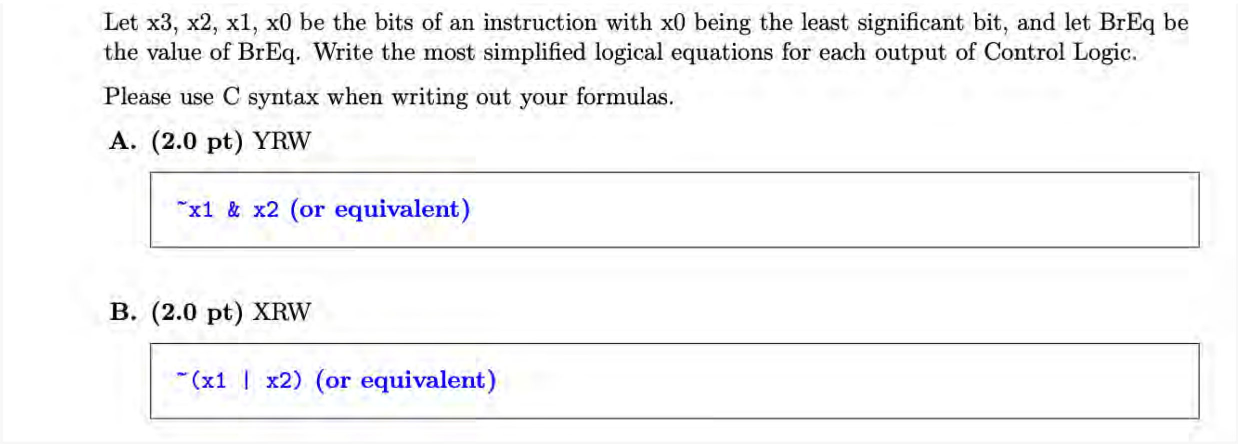
iv. You now want to create a circuit that runs Mover. In order to do that, you assign each Mover command a unique 4-bit code (explanations of each command have been copied for reference):

- `(0001)` `>` Moves the pointer one step right (+1 to x)
- `(1001)` `<` Moves the pointer one step left (-1 to x)
- `(0101)` `^` Moves the pointer one step up (+1 to y)
- `(1101)` `v` Moves the pointer one step down (-1 to y)
- `(0011)` `+` Increments the value at the pointer by 1
- `(1011)` `-` Decrements the value at the pointer by 1
- `(0111)` `]` If the pointer is currently pointing at a 0 and the current program flow is "FORWARD", change the program flow to "BACKWARD". Otherwise do nothing.
- `(1111)` `[` If the pointer is currently pointing at a 0 and the current program flow is "BACKWARD", change the program flow to "FORWARD". Otherwise do nothing.

You have finished the general structure of the circuit, and just need to finish writing the control logic. The memories in question are asynchronous read but synchronous write (thus the CLK). During startup the registers are set to 0. Note that the direction register now holds zero (not one) when the program is going forward. We just run forever and it is simply undefined behavior to either overflow or underflow the PC so we aren't worrying about having to check for any of that:



Singe Cycle Mover



Let x3, x2, x1, x0 be the bits of an instruction with x0 being the least significant bit, and let BrEq be the value of BrEq. Write the most simplified logical equations for each output of Control Logic.

Please use C syntax when writing out your formulas.

A. (2.0 pt) YRW

`~x1 & x2` **(or equivalent)**

B. (2.0 pt) XRW

`~(x1 | x2)` **(or equivalent)**

helpful! 0

Adelson Chua 5 months ago

Looking at the diagram, YRW controls the WE of the register that determines the value of Y (you can also see it coming from a multiplexed adder adds 1 or adds ffffff (which is -1))

Thus, this would correspond to the 0101 (+1 to y) and 1101 (-1 to y) instructions given in the problem.

Following the same logic, XRW controls the WE for X. This corresponds to the 0001 (+1 to x) and 1001 (-1 to x) instructions.

Now, we have a truth table:

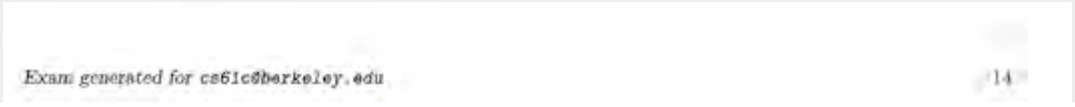
x3	x2	x1	x0	**YRW**	**XRW**
0	0	0	1	**0**	**1**
1	0	0	1	**0**	**1**
0	1	0	1	**1**	**0**
1	1	0	1	**1**	**0**
0	0	1	1	**0**	**0**
1	0	1	1	**0**	**0**
0	1	1	1	**0**	**0**
1	1	1	1	**0**	**0**

At this point, just write the corresponding Boolean expression for YRW and XRW in terms of X3 X2 X1 X0 and simplify.

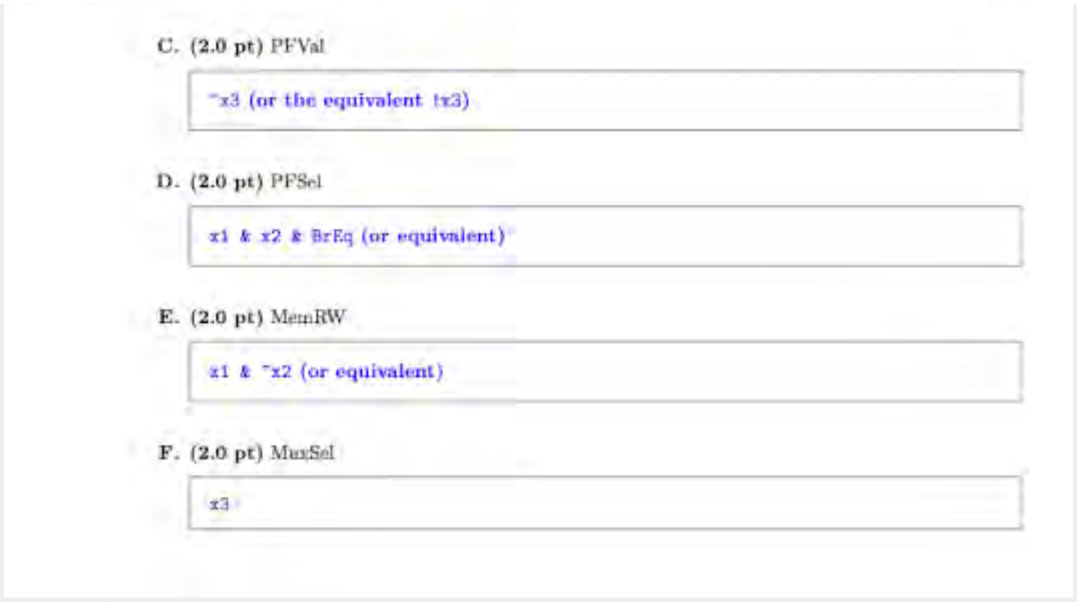
good comment 0

Anonymous Comp 2 5 months ago

Okay I see. Thank you! How would I come up with the rest? Do I follow the same logic? Sorry I'm really lost on this part.



Exam generated for cs61c@berkeley.edu 14


C. (2.0 pt) PFVal

~x3 (or the equivalent !x3)

D. (2.0 pt) PFSel

x1 & x2 & BrEq (or equivalent)

E. (2.0 pt) MemRW

x1 & ~x2 (or equivalent)

F. (2.0 pt) MuxSel

x3

helpful! | 0

Adelson Chua 5 months ago

You would need to understand how the datapath works and relate how the control signals affect the flow of data.

There's really no methodical way of solving this aside from just looking at the datapath and see what the purpose of the different multiplexers/registers/blocks do in the context of the described functionality.

good comment | 0

Anonymous Scale 2 5 months ago

For MemRW, why is it not ~x1 & x2? I was thinking about when the intruction is 0011 or 1011 which is when x1 is 0 and x2 is 1, we want to set MemRW to 1.

helpful! | 0

Adelson Chua 5 months ago

The ordering of the bits is x3,x2,x1,x0, so that actually translates to x2=0 and x1=1.

Good to know you were able to understand the datapath though!

good comment | 0

Anonymous Helix 2 5 months ago

Would it be incorrect if we added a "&x0" to some parts? (I realized after looking at the answers that x0 was always 0)

helpful! | 0

Anonymous Helix 2 5 months ago

for C and D, is PFVal our "new" program flow value if the cases for "]" or "[" pass and PFSel whether or not we choose that "new" value?

helpful! | 0

Adelson Chua 5 months ago

Would it be incorrect if we added a "&x0" to some parts?

It is logically correct. But I think the problem requires that the expression is in the most simplified form.

is PFVal our "new" program flow value if the cases for "]" or "[" pass and PFSel whether or not we choose that "new" value?

Sounds correct.

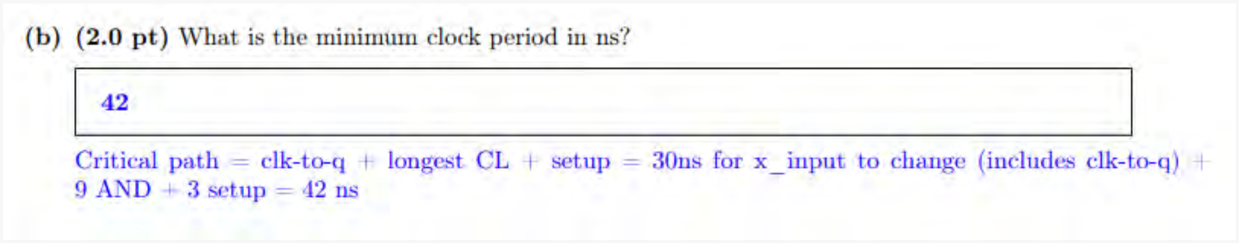
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f13

Anonymous Gear 2 5 months ago

[Summer 2020 Final Q4 part b]



(b) (2.0 pt) What is the minimum clock period in ns?

$\boxed{42}$

Critical path = clk-to-q + longest CL + setup = 30ns for x_input to change (includes clk-to-q) + 9 AND + 3 setup = 42 ns

Is it safe to assume that clk-to-q will always implicity be included in the time it takes for an input to change? I'm confused on what actually clk-to-q is, formally.

helpful! | 1

Peyrin Kao 5 months ago

clk-to-q is the time it takes for the register output to change after the rising edge of the clock. In this circuit, the critical path is between two registers, so after a rising edge, you have to first wait the clk-to-q time before the register output changes and the combinatorial logic can begin running. In general, whether you include clk-to-q depends on the context of the question. For example, if the question said that the input changes instantly, then you wouldn't need to wait the clk-to-q time for the combinatorial logic to start running.

good comment | 0

Eric Lu 5 months ago

Thank you!

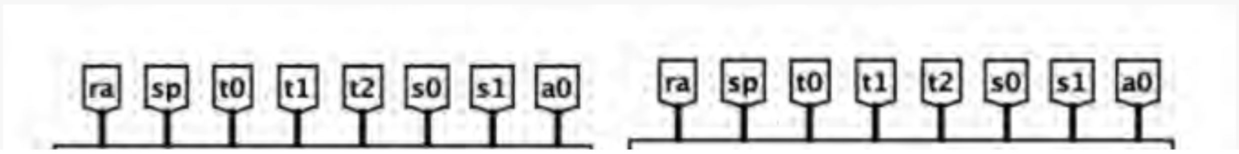
helpful! | 0

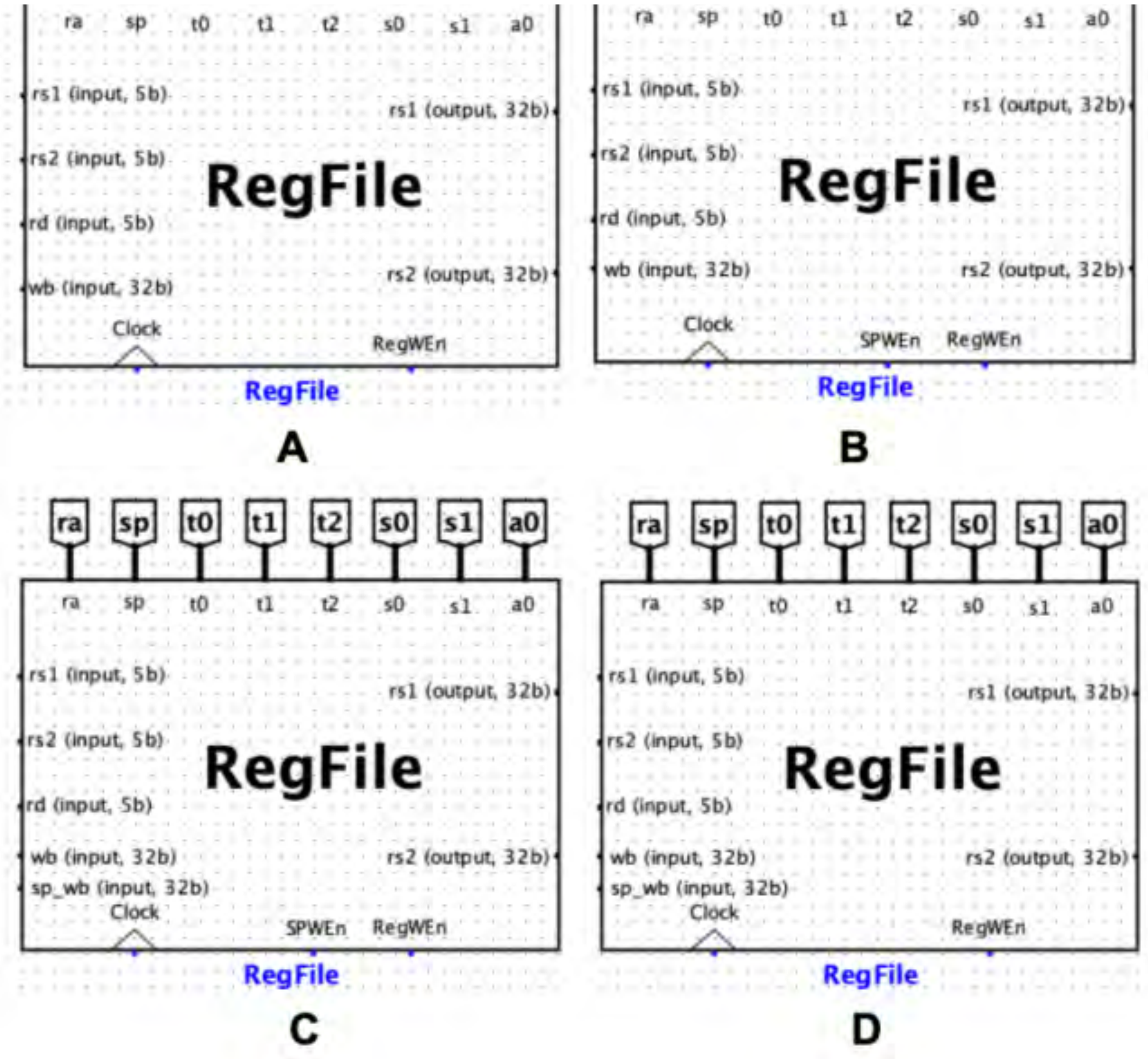
Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f14

Anonymous Gear 2 5 months ago

[Summer 2020 Final Q5 part bii]





ii. (2.0 pt) Now that we have `sp_wb`, which of the following will correctly write it back to the RegFile? `SPWEn` is a signal analogous to `RegWEn`: it is 1 when we wish to write to `sp` and 0 otherwise. Furthermore, if `SPEn` is false, `SP` will not be updated, even if `RegWEn` is true.

☐ A

■ C

☐ D

☐ B

Could someone please explain this question? I didn't know how to approach it.

helpful! | 0

Adelson Chua 5 months ago

You're just looking for the RegFile that has the sp_wb and SPWEn ports, that's all.

That's what the problem has been focusing on in the previous subquestions.

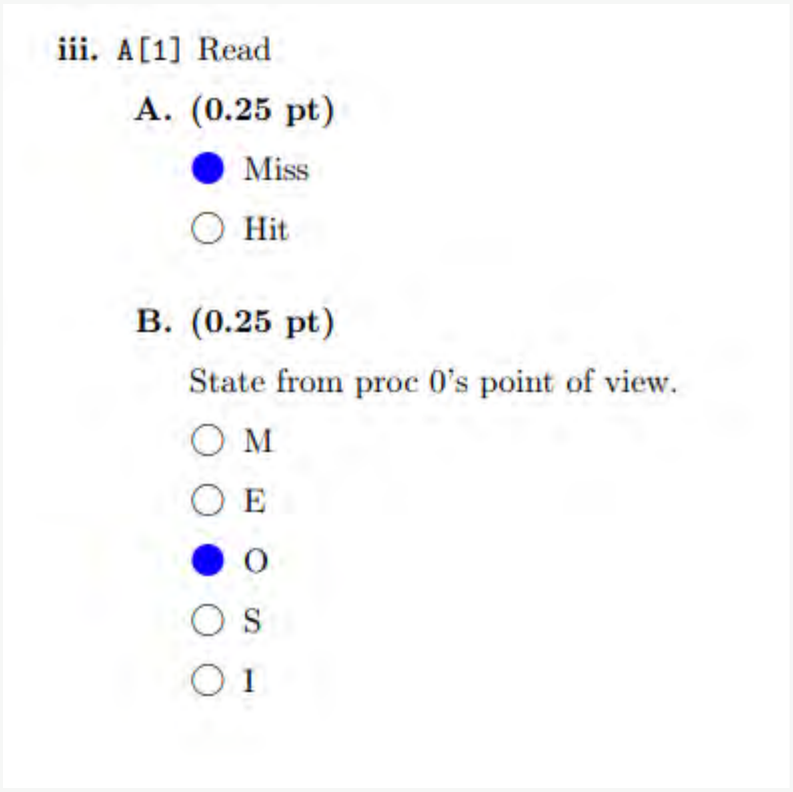
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f15

Anonymous Gear 2 5 months ago

[Summer 2020 Final Q6 part cii]



iii. A[1] Read

A. (0.25 pt)

● Miss

○ Hit

B. (0.25 pt)

State from proc 0's point of view.

○ M

○ E

● O

○ S

○ I

Why is the state from proc 0's point of view not Invalid? In this question, if processor 1 is the one that reads A[1], doesn't that mean from proc 0's perspective, isn't that value Invalid?

helpful! 0

Adelson Chua 5 months ago

If you check the diagram for MOESI. Proc0 was in Modified state before this question, then a probe read hit (another processor reads) goes to Owned.

Why would it be invalid? The other processor is just reading the data...

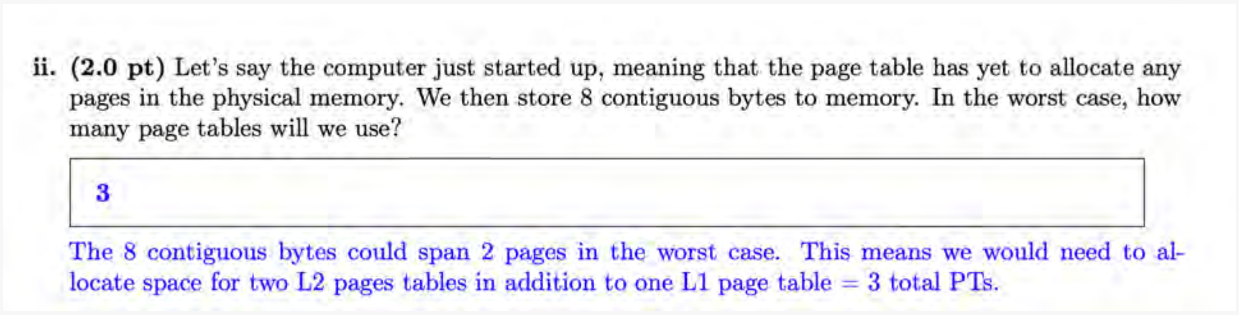
good comment 1

Reply to this followup discussion

● Resolved ○ Unresolved @2127_f16

Anonymous Beaker 5 months ago

[su20-final-1bii]



ii. (2.0 pt) Let's say the computer just started up, meaning that the page table has yet to allocate any pages in the physical memory. We then store 8 contiguous bytes to memory. In the worst case, how many page tables will we use?

3

The 8 contiguous bytes could span 2 pages in the worst case. This means we would need to allocate space for two L2 pages tables in addition to one L1 page table = 3 total PTs.

do we need 2 L2 page tables in the worst case bc the 8 contig bytes of memory could be split across 2 physical pages in memory, and the PPN for these two physical pages are split into two separate page tables (ie one PPN is at the end of one L2 page table and the other PPN is at the start of the second L2 page table)?

helpful! | 0

Adelson Chua 5 months ago
Yes! Exactly!

good comment | 0

Anonymous Beaker 2 5 months ago
Why couldn't the 8 byte be split for L1?

helpful! | 0

Adelson Chua 5 months ago
You can't split the L1. There's only 1 L1 page table in a two-level page table set up.

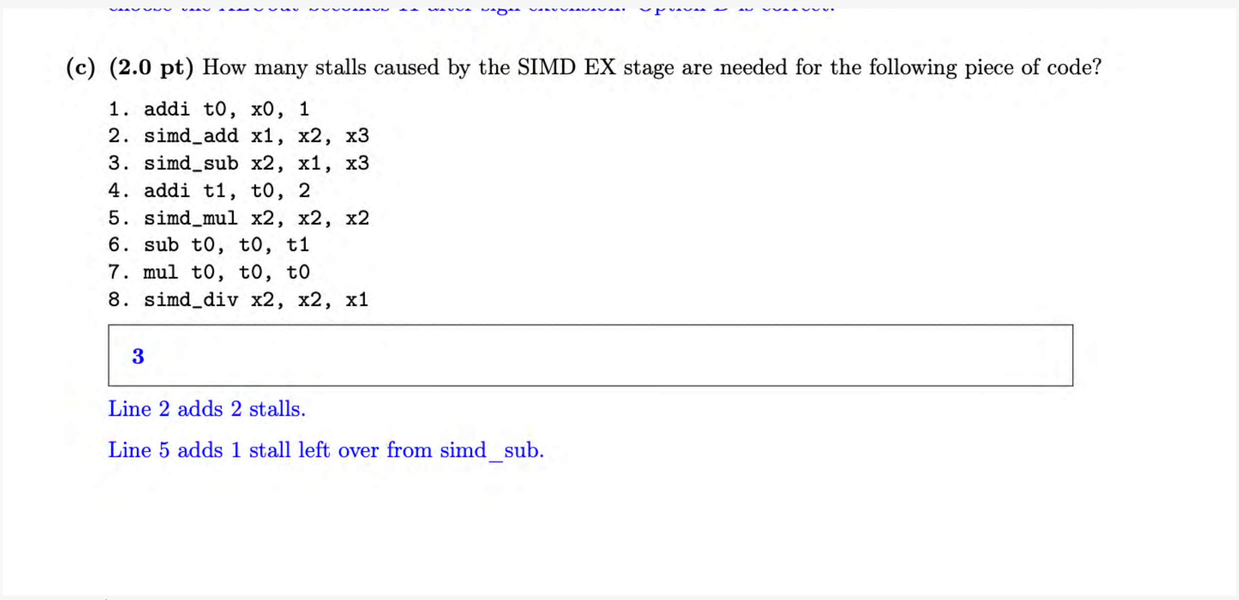
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f17

Anonymous Scale 2 5 months ago
[su20-final-Q3c]

I'm confused about how stalling works here. I thought we need to wait for 3 cycles for SIMD instructions and therefore 3 stalls for each SIMD instruction.


(c) (2.0 pt) How many stalls caused by the SIMD EX stage are needed for the following piece of code?

```
1. addi t0, x0, 1
2. simd_add x1, x2, x3
3. simd_sub x2, x1, x3
4. addi t1, t0, 2
5. simd_mul x2, x2, x2
6. sub t0, t0, t1
7. mul t0, t0, t0
8. simd_div x2, x2, x1
```

3

Line 2 adds 2 stalls.

Line 5 adds 1 stall left over from simd_sub.

helpful! | 0

Peyrin Kao 5 months ago
Line 2 adds 2 stalls because the IFD stage of line 3 can execute while line 2 is in its third SIMD EX stage.

Line 5 adds one stall since it needs to wait for line 3 to finish its third SIMD EX stage.

```
 1   IFD   EX    MEM   WB
2        IFD   EX    EX    EX    MEM   WB
3              -     -     IFD   EX    EX    EX    MEM    WB
4                                IFD   EX     MEM   WB
5                                            -      IFD    EX      EX     EX     MEM
WB
```

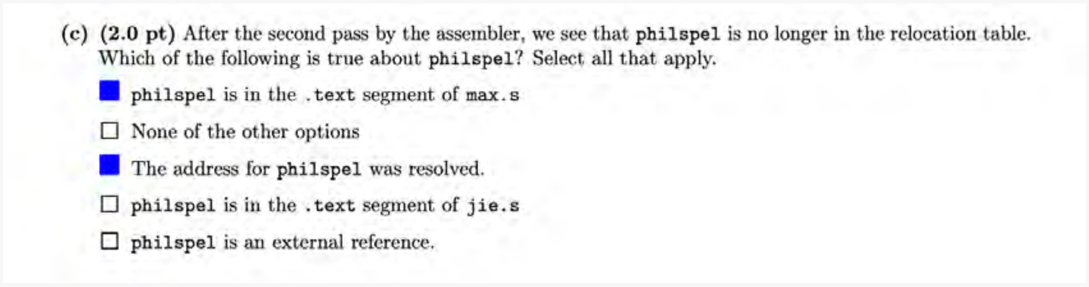
good comment | 2

Reply to this followup discussion

Resolved Unresolved @2127_f18

Anonymous Scale 2 5 months ago

[su20-final-Q7c]



(c) **(2.0 pt)** After the second pass by the assembler, we see that **philspel** is no longer in the relocation table. Which of the following is true about **philspel**? Select all that apply.

- ■ **philspel** is in the **.text** segment of **max.s**
- ☐ None of the other options
- ■ The address for **philspel** was resolved.
- ☐ **philspel** is in the **.text** segment of **jie.s**
- ☐ **philspel** is an external reference.

Could someone explain why it is true that philspel is in the .text segment?

helpful! | 0

Anonymous Scale 2 5 months ago

I thought it can also be in the code segment as long as it's referenced before it's defined therefore the first pass was not able to resolve it.

helpful! | 0

Caroline Liu 5 months ago

If in the first pass of the assembler a label is resolved, we know definitively it's in the code/text segment of that file **and** that it's defined before the label is used. The second pass takes a look at labels that are either in an external file/library or is defined **after** it is used.

Because at this stage our file does not have any knowledge beyond just the file, it cannot resolve anything not defined within itself. Thus, we know that `philspel` cannot be defined in an external file or library (that's taken care of by the linker).

Thus, `philspel` is defined within the file itself. There are two main chunks when it comes to an assembly file: the static/data section and the text section.

The static section **cannot** be resolved at the assembly stage. This is because static data belongs in the static segment of memory, which is independent from the text segment. We know the linker is in charge of merging and (re)ordering the combined static and text segments for all files linked at the linker stage, so as of right now, we have no idea where in static memory any static data will reside. Thus, it cannot be resolved and we know `philspel` cannot be defined in `.static` of `max.s`.

That leaves us with the `.text` segment of `max.s` by process of elimination. But let's prove it!

We know within a single file, the text segment will not be recorded (since they're instructions and instructions should be executed in the order the programmer/compiler intended for them to be). Thus, as long as we know where the label is defined within the scope of `max.s`, we can technically resolve it.

This is true for all branch instructions, since they're all PC-relative and thus, shifting and relocating the code will not affect its calculation.

For `jal` instructions (the main other one besides `b*` that uses PC-relative addressing), the label can be a local or external reference. External references need to be relocated, which means they cannot be resolved after the second pass of the assembler. However, local ones can be.

Thus this implies `philspel` is in the text of `max.s`.

Hope this makes sense!

good comment | 2

Anonymous Poet 2 5 months ago
is the .static you are referring to here the same as the .data assembler directive?
helpful! | 0

Caroline Liu 5 months ago
Yep!

good comment | 0

Anonymous Scale 3 5 months ago
Hi, Caroline. Thanks for the deep explanation. So it that means the branch, jump are in .text segment as well? Or typically, what is the factor we used to check something is in .text segment?
helpful! | 0

Caroline Liu 5 months ago
`.text` is just defined as anything that's considered to be code and would go in the code/text segment of main memory when loaded. This does include jump and branch instructions yes. There is no real "check" I think aside from the fact that when you look at an assembly file, everything after the `.text` directive, everything is considered to be code!

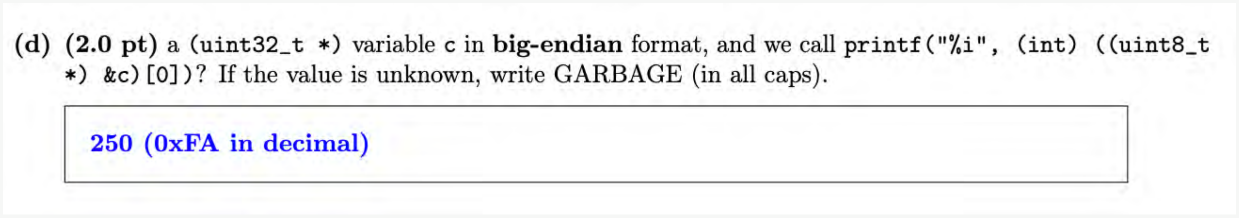
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f19

Anonymous Poet 2 5 months ago

SP20-Final-Q1d


(d) (2.0 pt) a `(uint32_t *)` variable `c` in **big-endian** format, and we call `printf("%i", (int) ((uint8_t *) &c)[0])`? If the value is unknown, write GARBAGE (in all caps).

`250 (0xFA in decimal)`

shouldn't this be -6 instead of 250? the variable c stores 0xFA00003. So we first cast the pointer it into a uint8_t array, and then we access the first element which is 0xFA in big endian. And then finally we cast it into a signed int with (int), and 0xFA is -6 with sign.

helpful! | 0

Peyrin Kao 5 months ago
I don't think casting an unsigned integer to an integer causes sign-extension. The 32-bit integer would be 0x0000 00FA, which is 250 in decimal.

good comment | 0

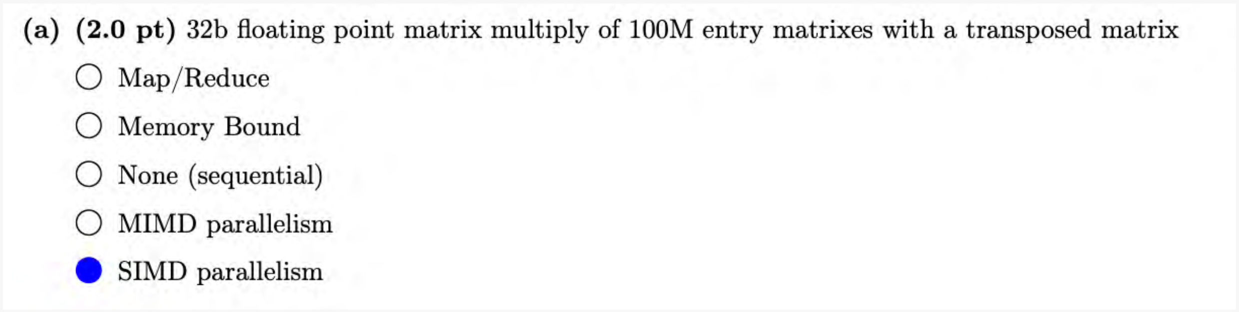
Anonymous Poet 2 5 months ago
ah i see thank you

helpful! | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f20

Anonymous Poet 2 5 months ago
SP20-Final-Q3a


(a) (2.0 pt) 32b floating point matrix multiply of 100M entry matrixes with a transposed matrix

- Map/Reduce
- Memory Bound
- None (sequential)
- MIMD parallelism
- [✓ SIMD parallelism]

I picked SIMD but I'm not sure why MIMD wouldn't work. When doing matrix multiplication, we aren't writing to the same locations in memory when doing matrix multiplication so there wouldn't be data race issues I don't think. And we could use SIMD with MIMD right as we did in project 4 for matrix multiply?

helpful! | 0

Peyrin Kao 5 months ago
I think SIMD is the most appropriate answer here because you're performing one operation (multiplication, with maybe some addition) on multiple data. MIMD is a better answer when there are many instructions that have to be executed on multiple data. You're right that Project 4 uses both SIMD and MIMD for this task, though, so it's not the most unambiguous question.

good comment | 0

Reply to this followup discussion

[● Resolved] ○ Unresolved @2127_f21

Anonymous Scale 2 5 months ago
[su20-final-Q2]

I'm confused about when we used temperal locality. Every cache used LRU, but how do we determine when they actually use temperal locality?

helpful! | 0

Adelson Chua 5 months ago
I don't see this at Su20-Final.

good comment | 0

Anonymous Scale 2 5 months ago
Sorry about that, it should be sp20.

helpful! | 0

Adelson Chua 5 months ago
Temporal locality means that you are getting cache hits because the data has been loaded in the cache before and it is still there when you need it to access it again.

So basically, ask yourself, the question says there are 29N hits, why is that so?

good comment | 1

Reply to this followup discussion

[● Resolved] ○ Unresolved @2127_f22

Anonymous Scale 2 5 months ago
[sp20-final-Q3d]

Why would part D be a memory bound while part A can be resolved using SIMD. They are the same size. It it because part A is using a transpose matrix?

helpful! | 0

Adelson Chua 5 months ago
It's mainly matrix multiplication vs matrix addition.

Matrix multiplication is a couple of parallel multiplications then an addition (dot product) writing to **one element at a time** of the resulting matrix.

Matrix addition is a couple of parallel additions writing to **several elements in parallel** of the resulting matrix. This causes the memory-bound issue.

good comment | 1

Anonymous Beaker 2 5 months ago
But isn't memory bound a problem when there's reading not writing as the problem says "If a problem requires reading as many memory locations as compute operations, mark it as “memory

bound""

helpful! | 0

Peyrin Kao 5 months ago

In matrix multiplication, computing each element of the result matrix requires n operations (dot product), and there are n^2 elements in the result matrix, for a total of n^3 computations and n^2 memory accesses.

In matrix addition, computing each element of the result matrix requires 1 operation (addition), and there are n^2 elements in the result matrix, for a total of n^2 computations and n^2 memory accesses.

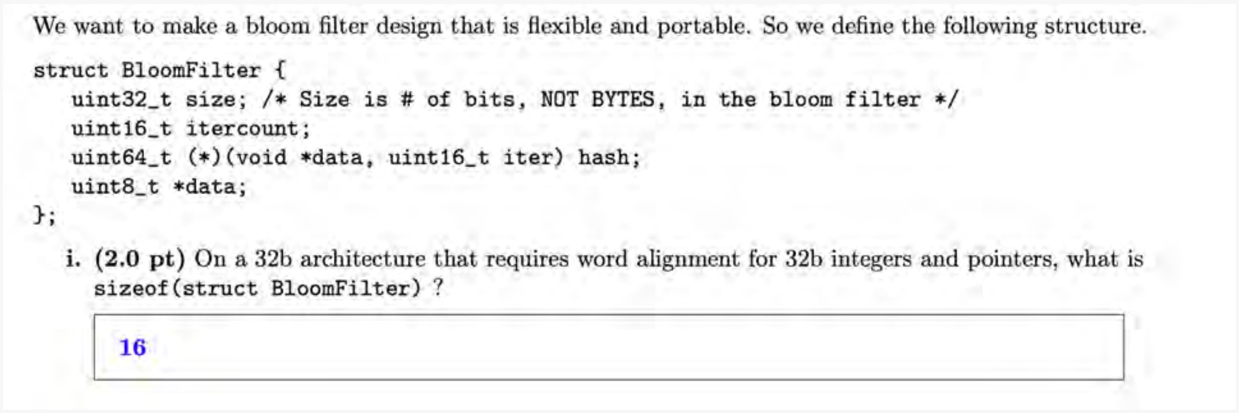
good comment | 0

Reply to this followup discussion

[● Resolved] Unresolved @2127_f23

Anonymous Scale 2 5 months ago

[sp20-final-Q7]


We want to make a bloom filter design that is flexible and portable. So we define the following structure.

```
struct BloomFilter {
   uint32_t size; /* Size is # of bits, NOT BYTES, in the bloom filter */
   uint16_t itercount;
   uint64_t (*)(void *data, uint16_t iter) hash;
   uint8_t *data;
};
```

i. **(2.0 pt)** On a 32b architecture that requires word alignment for 32b integers and pointers, what is `sizeof(struct BloomFilter)` ?

16

I'm confused on how we get 16. We are given that it requires word alignment for 32b integers and pointers(which is 4 bytes), and I also see the largest field in the struct is unit64_t (which is 8 bytes) (or is this actually a pointer to a unit64_t? I'm not really sure) . I recall from lecture that we need to slign with the largest field. I'm not sure if we need to align with 8 bytes or 4 bytes in this case. I was thinking about: assuming we need alignment for 32b, we first need 4 bytes for size, then 4 bytes for itercount, then 8 bytes for hash, lastly, 4 bytes for data, therefore a total of 20 bytes.

helpful! | 0

Adelson Chua 5 months ago

@2127_f4

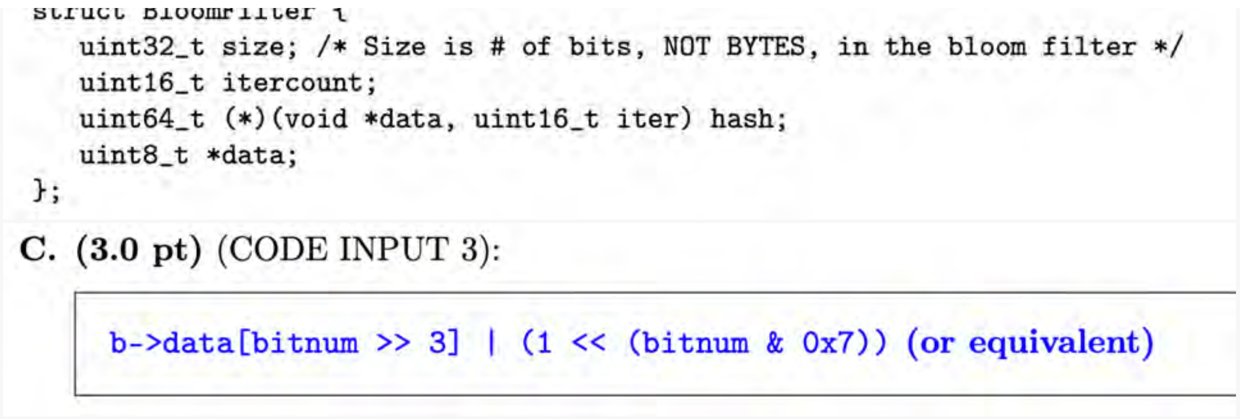
good comment | 1

Reply to this followup discussion

[● Resolved] Unresolved @2127_f24

Anonymous Poet 2 5 months ago

SP20-Final-Q7biii



```
struct BloomFilter {
    uint32_t size; /* Size is # of bits, NOT BYTES, in the bloom filter */
    uint16_t itercount;
    uint64_t (*)(void *data, uint16_t iter) hash;
    uint8_t *data;
};
```

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

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

So the elements of data are of type uint8_t. We don't have to cast (1 << (bitnum & 0x7)) as a uint8_t?

helpful! | 0

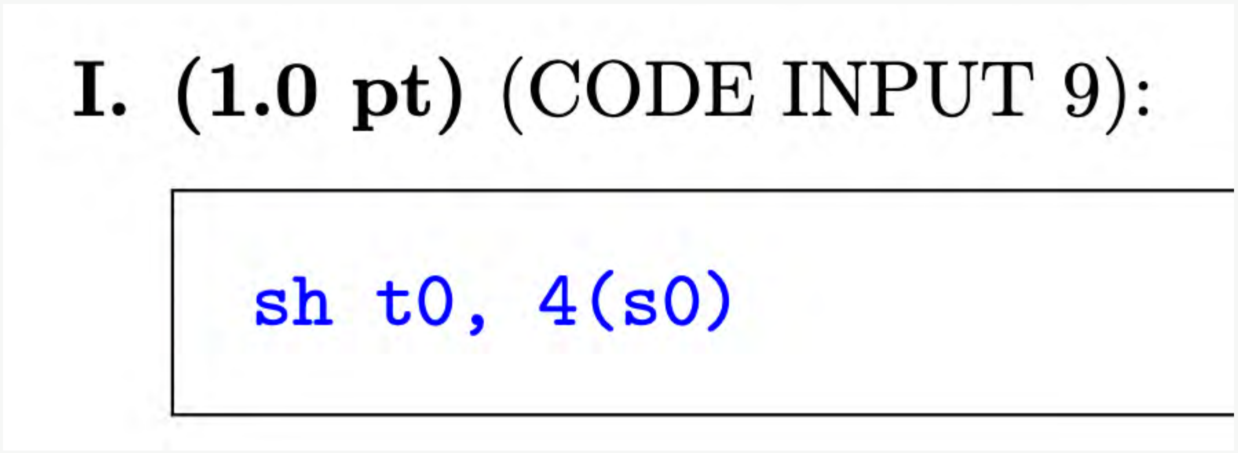
Peyrin Kao 5 months ago
Without an explicit cast, the upper bits of the `uint64_t` value `(1 << (bitnum & 0x7))` are discarded. Adding the cast probably wouldn't hurt, though.

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f25

Anonymous Poet 2 5 months ago
SP20-Final-Q7c.I



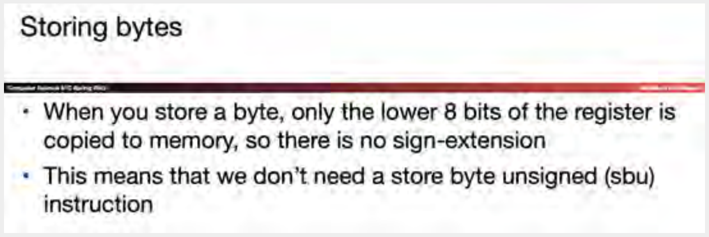
I. (1.0 pt) (CODE INPUT 9):

`sh t0, 4(s0)`

So only the last 16 bits of t0 have actual meaningful values. So just to clarify when I use the sh instruction, it takes the less significant half of register t0 and loads those 2 bytes to 4(s0) without affecting the 3rd and 4th bytes from address 4(s0)?

helpful! | 0

Anonymous Poet 2 5 months ago



Storing bytes

- When you store a byte, only the lower 8 bits of the register is copied to memory, so there is no sign-extension
- This means that we don't need a store byte unsigned (sbu) instruction

Found the answer.

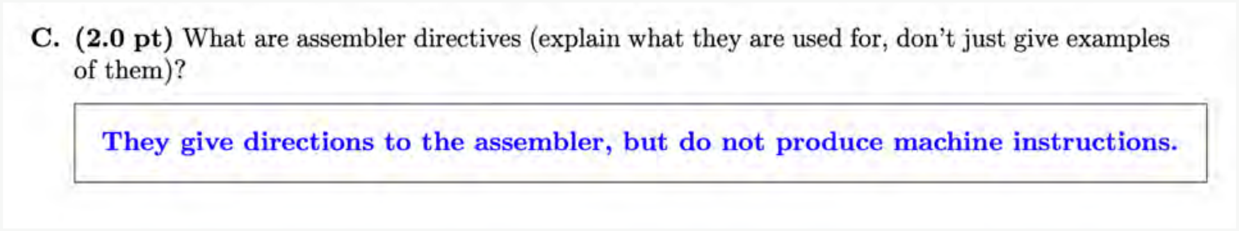
helpful! | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f26

Anonymous Poet 2 5 months ago

SP20-Final-Q8.C



C. (2.0 pt) What are assembler directives (explain what they are used for, don't just give examples of them)?

They give directions to the assembler, but do not produce machine instructions.

I thought that the assembler outputs .o files which are machine code. Doesn't this mean that the machine code is made up of machine instructions (albeit we still have to resolve absolute offsets)?

helpful! | 0

Caroline Liu 5 months ago

The assembler itself is what eventually helps provide the machine instructions without absolute references but the assembler directives themselves are not in charge of producing the machine instructions. The job of the directives is to provide information to the assembler, such as start and end of a program/file, memory locations, where start/end of macros/segments are, and more.

Does that answer your question?

good comment | 1

Anonymous Poet 2 5 months ago

yes ty!

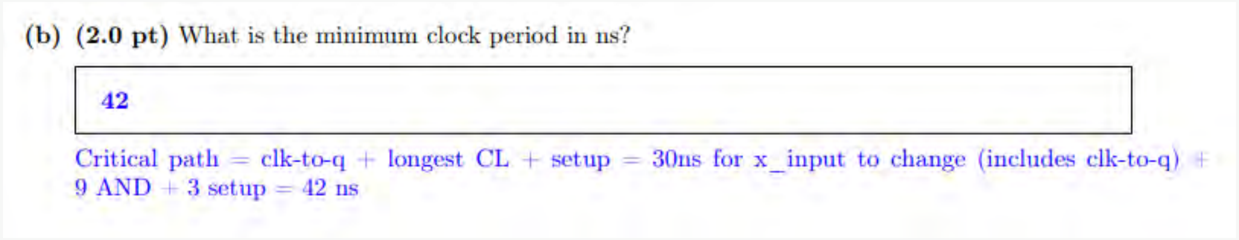
helpful! | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f27

Anonymous Atom 2 5 months ago

su20 q4 part b



(b) (2.0 pt) What is the minimum clock period in ns?

42

Critical path = clk-to-q + longest CL + setup = 30ns for x_input to change (includes clk-to-q) + 9 AND + 3 setup = 42 ns

Can someone help me understand why we use x input here? i thought min clock period was to find the longest cl between two registers, which x isnt, and add the clk to q and setup time. I think I'm missing something conceptually?

helpful! | 0

Anonymous Atom 2 5 months ago

i wouldve thought the answer wouldve been clock to q + OR + AND (from reg0 to y output) + setup

helpful! | 0

Anonymous Atom 2 5 months ago

since x input isn't a register, I dont get why we're only calculating from x input to reg1

helpful! | 0

Adelson Chua 5 months ago

@2127_f13

good comment | 0

Anonymous Atom 2 5 months ago

i understand the critical path is between two registers, is x input a register because the problem says it changes 30ns after the rising edge of the clock?

helpful! | 0

Adelson Chua 5 months ago

yes.

good comment | 1

Anonymous Atom 2 5 months ago

okay thank you!

helpful! | 0

Adelson Chua 5 months ago

You treat x input as a register with a clk-to-q of 30ns

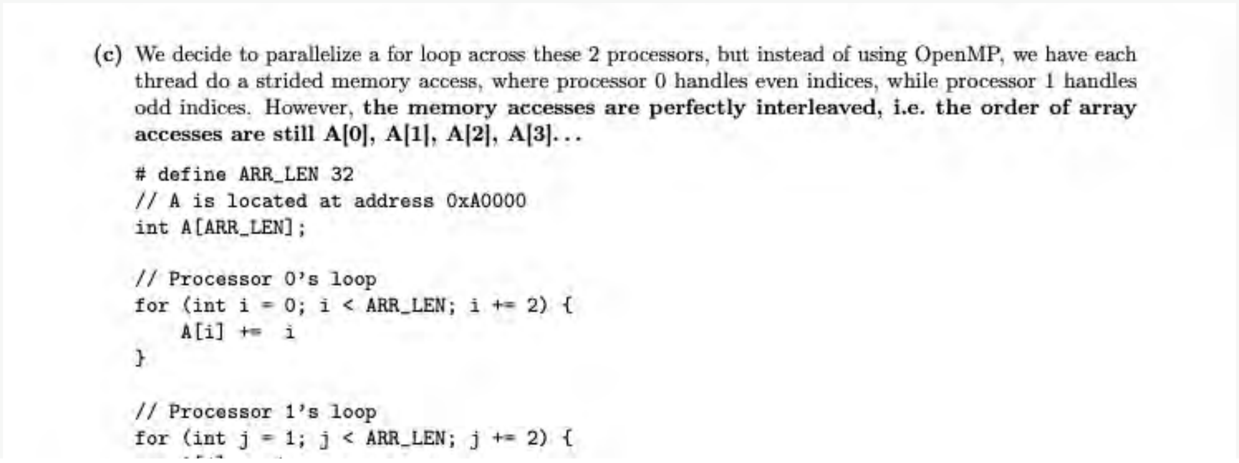
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved **@2127_f28**

Anonymous Mouse 2 5 months ago

Can someone explain how to go through Summer 2020 Final 6c? I'm confused with how to determine whether it is a hit or miss and when something is owned.

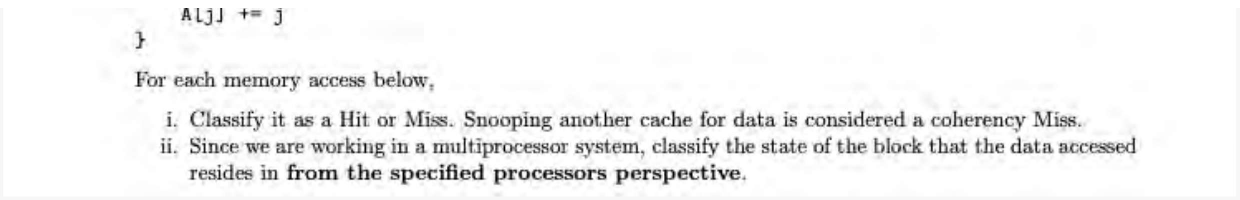


(c) We decide to parallelize a for loop across these 2 processors, but instead of using OpenMP, we have each thread do a strided memory access, where processor 0 handles even indices, while processor 1 handles odd indices. However, **the memory accesses are perfectly interleaved, i.e. the order of array accesses are still A[0], A[1], A[2], A[3]...**

```
# define ARR_LEN 32
// A is located at address 0xA0000
int A[ARR_LEN];

// Processor 0's loop
for (int i = 0; i < ARR_LEN; i += 2) {
    A[i] += i
}

// Processor 1's loop
for (int j = 1; j < ARR_LEN; j += 2) {
```



```
        A[j] += j
}
```

For each memory access below,

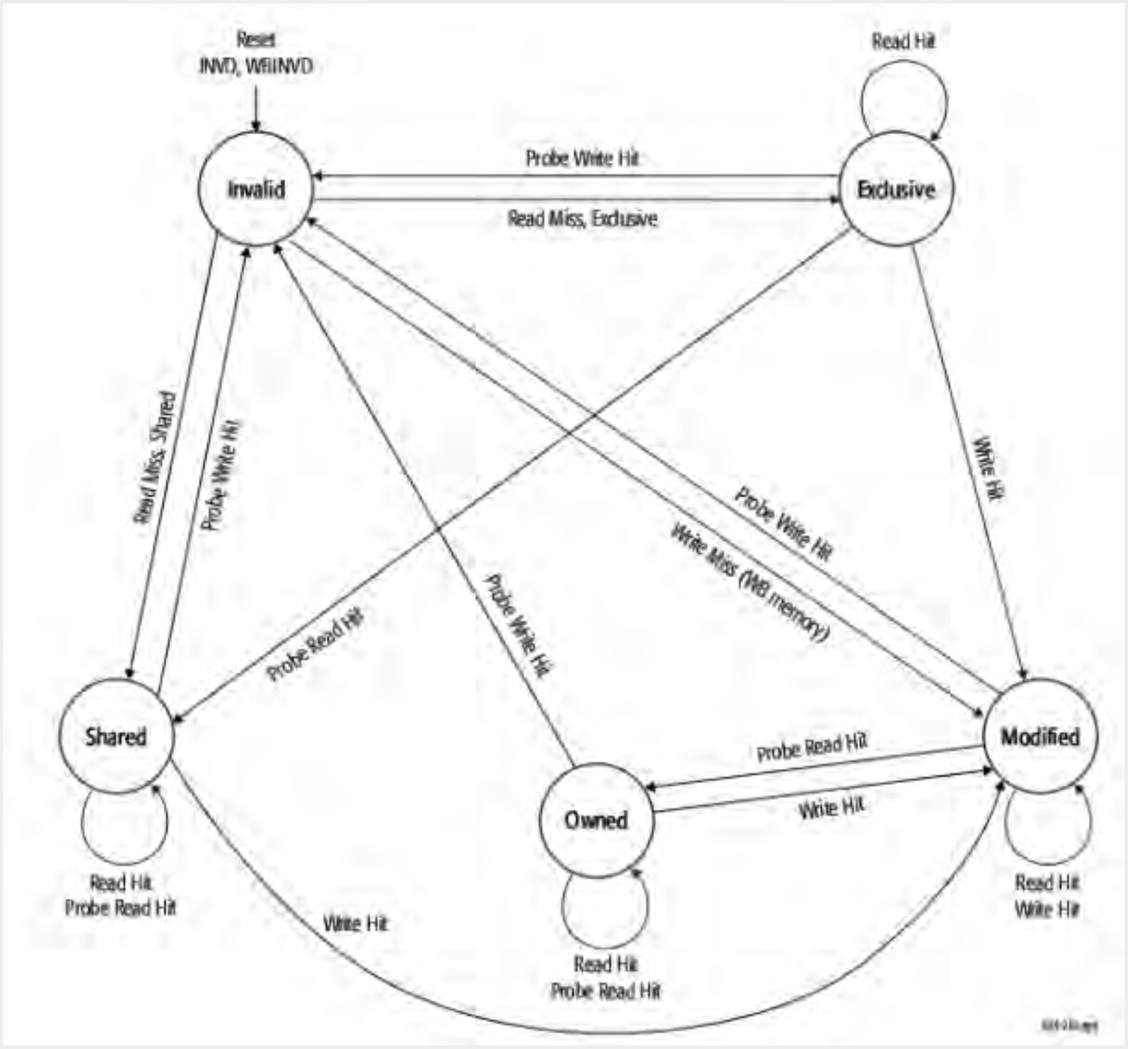
i. Classify it as a Hit or Miss. Snooping another cache for data is considered a coherency Miss.
ii. Since we are working in a multiprocessor system, classify the state of the block that the data accessed resides in **from the specified processors perspective**.

helpful! | 0

Adelson Chua 5 months ago

You would have to look at the state diagram to truly understand this.

Know that in the beginning, **you start at the Invalid state.**



i) **A[0] Read** (this is done by processor 0, as stated in the problem)

This is a **compulsory miss**. You already know this from a typical cache problem.

Looking at the diagram, from Invalid, you go to **Exclusive** after a read miss. Since this is the first processor to access that data, it gets 'exclusive' access to that data. Exclusive = I'm the only one who has this data.

ii) **A[0] Write** (this is still done by processor 0).

Due to the miss before, you pulled the data into the cache, so now this is a **hit**. Typical cache problem.

Looking at the diagram, from Exclusive state, you go to **Modified** after a write hit. Since the processor has made changes, it is now modified. Logically, it makes sense.

iii) **A[1] Read** (this is done by processor 1, as stated in the problem)

Since processor 1 is executing this, processor 1's cache is still empty. This is a **compulsory miss**.

Still at the POV of processor 0, looking at the diagram, from Modified state, you go to **Owned** state after a probe (probe means another processor, in this case, processor 1) read.

Additional info: In the POV of processor 1, starting from Invalid, you go to Shared after a read miss. Processor 0 is already holding this data in its cache, so now they're sharing it.

iv) **A[1] Write** (still done by processor 1)

After a miss previously, this is now a **hit**. Typical cache problem.

Now at the POV of processor 1, looking at the diagram, starting from Shared state earlier, you go to **Modified** after a write hit.

Additional info: In the POV of processor 0, starting from Owned state, you go to Invalid state after a probe (processor 1) write hit

v) **A[2] Read** (now back to processor 0, as stated in the problem)

As stated earlier, processor 0 is back at the Invalid state. So this one ends up as a **coherence miss**. Invalid state forces you to miss.

Now at the POV of processor 1, from Modified state earlier, you go to the **Owned** state after a probe (processor 0) read.

Additional info: as processor 0 had a miss, it will now go to Shared state (it shares data with processor 1).

vi) **A[2] Write** (still by processor 0)

After a miss, you **hit**. Typical cache problem.

Now at the POV of processor 0, from Shared state, you go to **Modified** state after a write hit.

Additional info: from processor 1's POV, it goes from Owned state to Invalid state due to a probe write hit.

vii) **A[3] Read** (done by processor 1)

From what I stated earlier, processor 1 is now Invalid, therefore this is a **coherence miss**.

From the POV of processor 0, from Modified state you go to **Owned** state after a probe (processor 1) read.

Additional info: from the processor 1's POV, it goes from Invalid to Shared.

viii) **A[3] Write** (done by processor 1)

After a miss, you **hit** on the same data. Easy.

From the POV of processor 1, you go from Shared to **Modified** after a write hit.

Additional info: from the processor 0's POV, you go from Owned state to Invalid state after a probe (processor 1) write hit.

Bottomline: you have to maintain two different 'states' for the two processors. Every cache access will update both of the states. You have to know which POV you are looking at.

good comment | 2

Anonymous Mouse 2 5 months ago

Thank you! Clarification about the Modified to Owned transition. The graph says that we go from modified to owned after a probe read hit for example. In iii, we have a probe read miss. Why do we still go to the Owned state?

helpful! | 0

Adelson Chua 5 months ago

Good question. There's really no concept of probe read miss or probe write miss, because they will technically hit once the data is loaded to the cache.

Should've just called it probe read/write for clarity.

good comment | 0

Anonymous Mouse 2 5 months ago

Thanks!

helpful! | 0

Anonymous Scale 3 5 months ago

How to do find out the partern for coherence miss for this problem?

helpful! | 0

Adelson Chua 5 months ago

Please go through my example.

Coherence miss happens when you should have loaded the data in the cache (because you accessed it before), but the state is now Invalid so you have to reload it again.

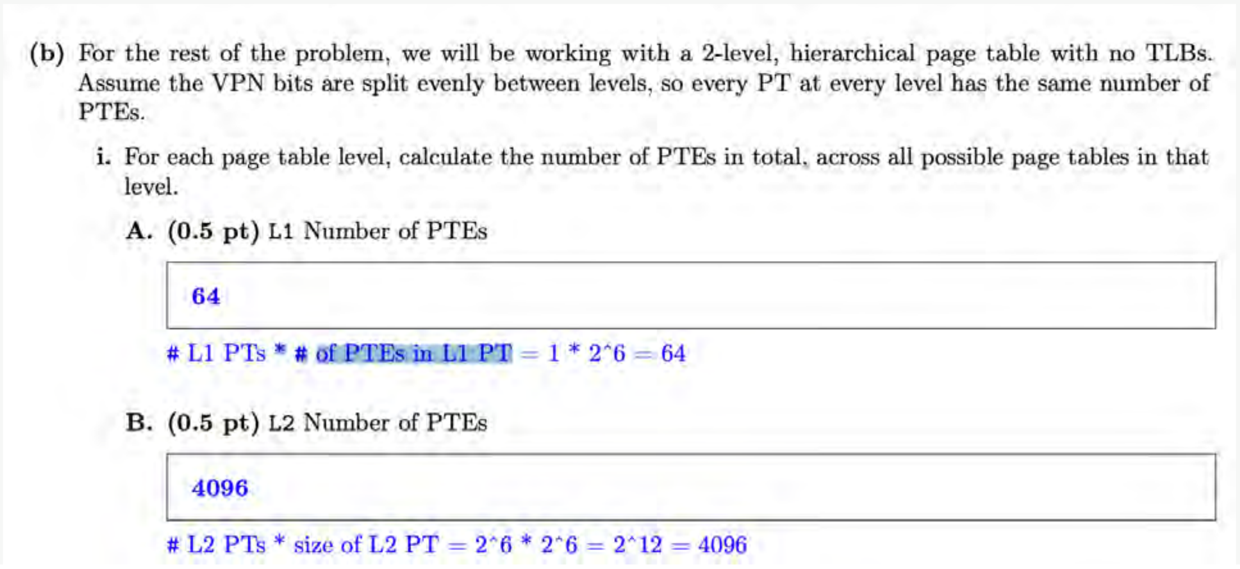
It is a type of miss that does not happen without multiprocessing.

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f29

Anonymous Beaker 2 5 months ago



(b) For the rest of the problem, we will be working with a 2-level, hierarchical page table with no TLBs. Assume the VPN bits are split evenly between levels, so every PT at every level has the same number of PTEs.

i. For each page table level, calculate the number of PTEs in total, across all possible page tables in that level.

A. (0.5 pt) L1 Number of PTEs

64

# L1 PTs * \# of PTEs in L1 PT = 1 * 2^6 = 64

B. (0.5 pt) L2 Number of PTEs

4096

# L2 PTs * size of L2 PT = 2^6 * 2^6 = 2^12 = 4096

How were the values for #L1/L2 PTs and # of PTEs in L1 PT calculated?

helpful! | 0

Adelson Chua 5 months ago

Exam number please.

good comment | 0

Anonymous Beaker 2 5 months ago

Su 2020

helpful! | 0

Adelson Chua 5 months ago

VPN1 = VPN2 = 6 (since L1 and L2 bits are divided evenly)

The L1 page table is indexed by the VPN1 bits. So you have 2^6 rows. That's already the number of PTEs present in L1.

Each entry of the L1 page table can point to an L2 page table. **So there are actually as many L2 page tables as there are the number of rows/entries in the L1 page table.**

The L2 page tables are indexed by the VPN2 bits. So you have 2^6 rows/entries. From the previous statement, there are also 2^6 L2 page tables. So, 2^6 * 2^6 is the total number of PTEs in L2.

good comment | 1

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f30

Anonymous Calc 2 5 months ago

SP20 Final Question 2 - what exactly does looping workload mean?

helpful! | 0

Adelson Chua 5 months ago

A loop. :)

In this context, a loop with array accesses.

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f31

Anonymous Comp 3 5 months ago

Spring 2020 1.a.iv:

Why is this value 250? I would have thought it is 251 (16*15 + 11).

helpful! | 0

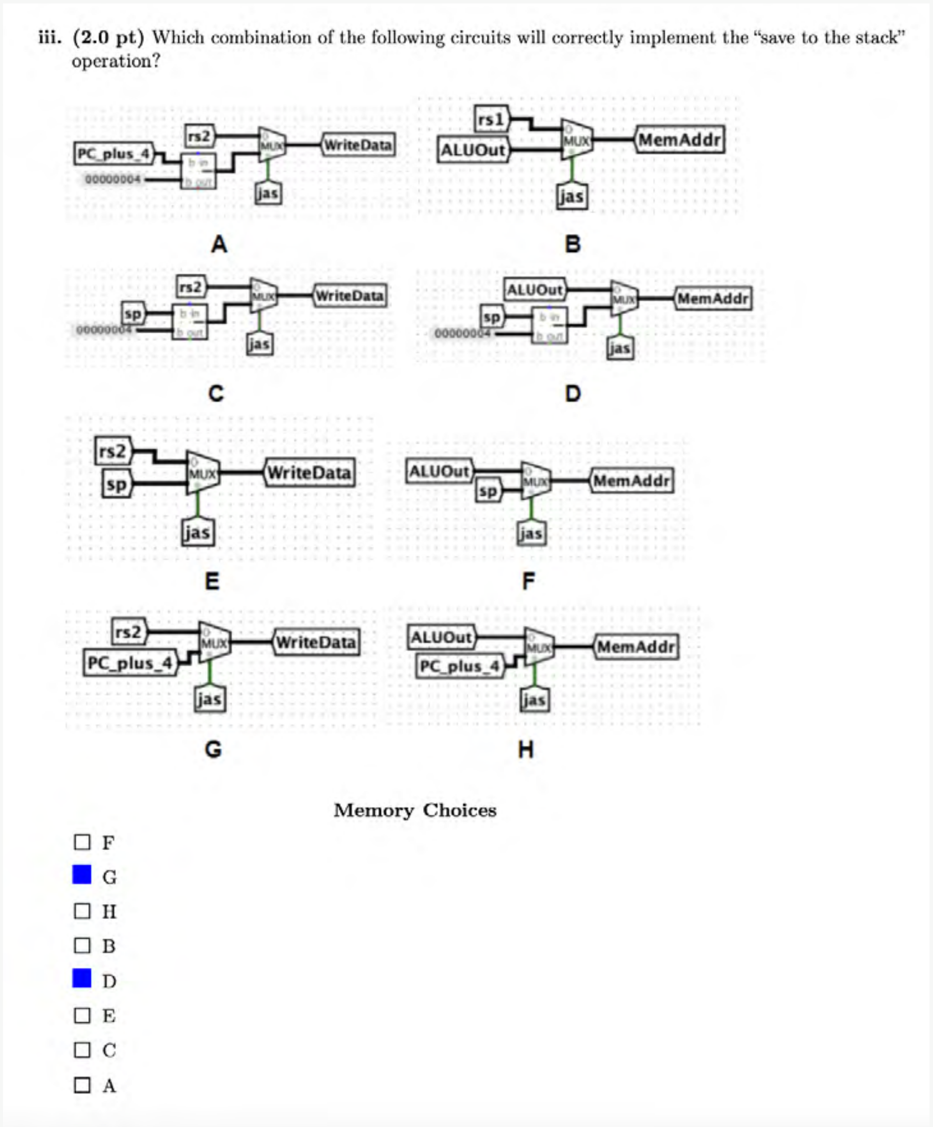
Adelson Chua 5 months ago
Can you clarify which specific exam this is?
good comment | 0

Adelson Chua 5 months ago
Found it: @2127_f19
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f32

Anonymous Beaker 2 5 months ago


iii. **(2.0 pt)** Which combination of the following circuits will correctly implement the "save to the stack" operation?

Memory Choices

☐ F
■ G
☐ H
☐ B
■ D
☐ E
☐ C
☐ A

I get why it's not B,E, or F? BUt I fail to see how H and D differ and C + G differ? Also why is G valid?
helpful! | 0

Anonymous Beaker 2 5 months ago
Su20
helpful! | 0

Adelson Chua 5 months ago

You want to do the following:

ra = pc + 4

sp = sp - 4

pc = pc+offset

Mem[sp-4] = PC+4

Choice **G** allows you to perform **ra = pc+4** (output of multiplexer is WriteData which goes to the register file to update ra)

Choice **D** allows you to do set the memory address to sp-4, basically in preparation for the **Mem[sp-4]** = pc+4 operation.

Choice **H** sets up the memory address to pc+4, so Mem[pc+4], we're not doing this.

Choice **C** sets up sp-4 to be written to the register file. But this problem focuses on the ra update. We are not doing ra = sp-4.

good comment | 1

Reply to this followup discussion

[✓ Resolved] Unresolved **@2127_f33**

Anonymous Calc 2 5 months ago

SP20 Final Question 2 - why are part a and part c answers different? My reasoning is that the fully associative cache can fit the entire array in both, and in part c, there are N lines left over after the first inner loop iteration as one 32 bit int fits in one block and there are 2N lines.

For part c - wouldn't the first iteration be all compulsory misses (N misses), and then the last 29 for loop iterations would be all hits because all the data of the array is already in the cache?

helpful! | 0

Adelson Chua 5 months ago

@2127_f2

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved **@2127_f34**

Anonymous Comp 3 5 months ago

Sp 2020 Final 5 a iii

What is the answer to this question?

helpful! | 0

Adelson Chua 5 months ago

@2127_f1

good comment | 0

Reply to this followup discussion

Resolved [✓] Unresolved @2127_f35

Anonymous Comp 3 5 months ago

Sp 2020 Final 5 b.

Not gonna lie I am not at all sure how to approach this problem. Could I get an explanation for the solutions?

helpful! 0

Yiteng Zhou 5 months ago

I assume 5.b is I Forget Where This Data Goes.

Note: the answersheet's u, v, w is not the same as questionsheet, so be careful if you check it wrong.

For the first two questions, you should figure it out what's the type of *u and u, which is straightforward.

For the next, be care what's the difference between x, &x and *x, which will help you to understand.

helpful! 0

Peyrin Kao 5 months ago

`v = &u` is the address of a local variable, which lives on the stack.

`*v = u` is the address of allocated memory (returned from the malloc call), which lives on the heap.

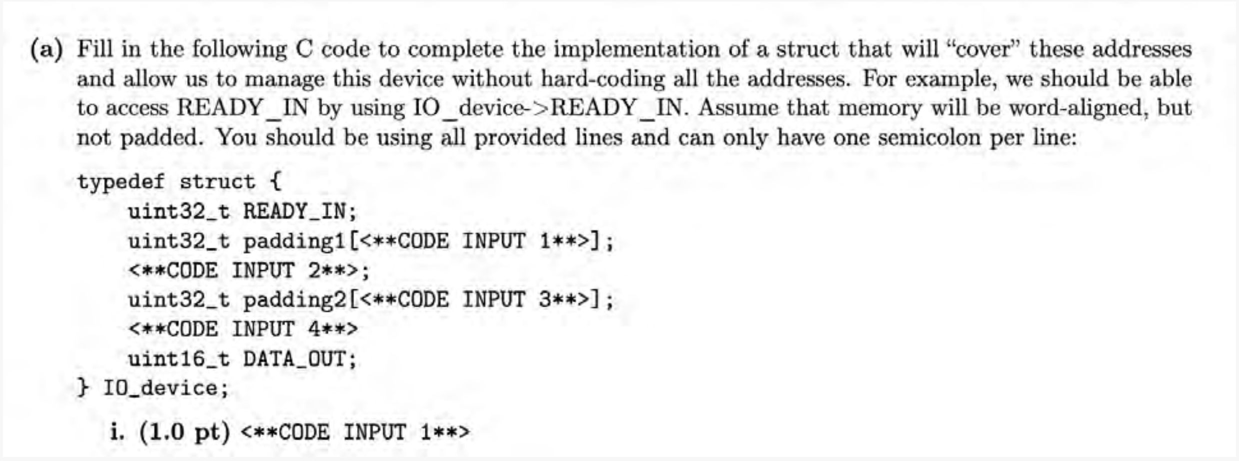
`u` is the address of the first element in an array of 100 elements (all stored side-by-side in memory), so `(u + 1)` is the address of the next element in that array. The entire array lives on the heap.

good comment 0

Reply to this followup discussion

Resolved [✓] Unresolved @2127_f36

Anonymous Beaker 2 5 months ago



(a) Fill in the following C code to complete the implementation of a struct that will "cover" these addresses and allow us to manage this device without hard-coding all the addresses. For example, we should be able to access READY_IN by using IO_device->READY_IN. Assume that memory will be word-aligned, but not padded. You should be using all provided lines and can only have one semicolon per line:

```
typedef struct {
    uint32_t READY_IN;
    uint32_t padding1[<**CODE INPUT 1**>];
    <**CODE INPUT 2**>;
    uint32_t padding2[<**CODE INPUT 3**>];
    <**CODE INPUT 4**>
    uint16_t DATA_OUT;
} IO_device;
```

i. (1.0 pt) <**CODE INPUT 1**>

What is padding? I thought structures were defined as __type__ __name__ ?

helpful! | 0

Adelson Chua 5 months ago

It is an array declaration.

uint32_t padding1[x]

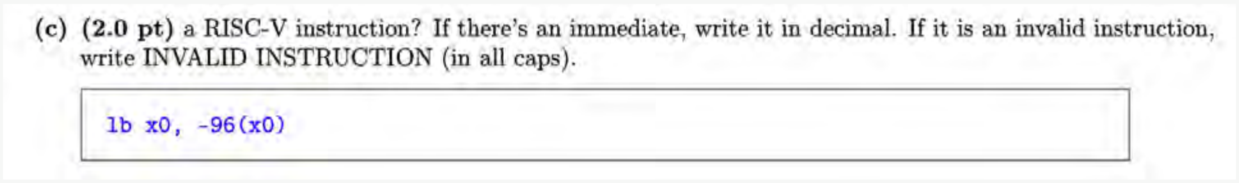
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f37

Anonymous Gear 3 5 months ago

SP20-Final-Q1(c), for this question I got lb x0 4000(x0), would this lead to the same answer?


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

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

helpful! | 0

Yiteng Zhou 5 months ago

The imm should be signed, so 4000 is wrong.

helpful! | 0

Peyrin Kao 5 months ago

Each hexadecimal value corresponds to exactly one RISC-V instruction (there aren't multiple instructions that assemble to the same hexadecimal value). In this case, -96 is the correct immediate because immediates are signed.

good comment | 0

Anonymous Atom 5 months ago

can someone explain to me how we get lb, x0, -96(x0) I have no clue how to transfer the hex to risc-v

helpful! | 0

Peyrin Kao 5 months ago

0xFA000003 = 0b1111 1010 0000 0000 0000 0000 0000 0011

Opcode (always bits 0-6) is 0b000 0011, so it's a load (I-type). We have to check funct3 (always bits 12-14) to figure out which instruction this is. funct3 is 0b000, so it's lb.

Now that we know it's an I-type, we can decode the rest of the bits. rd (bits 7-11) is 0b00000, which corresponds to x0. rs1 (bits 15-19) is 0b00000, which corresponds to x0. The immediate (bits 20-31) is 0b1111 1010 0000, which is -96.

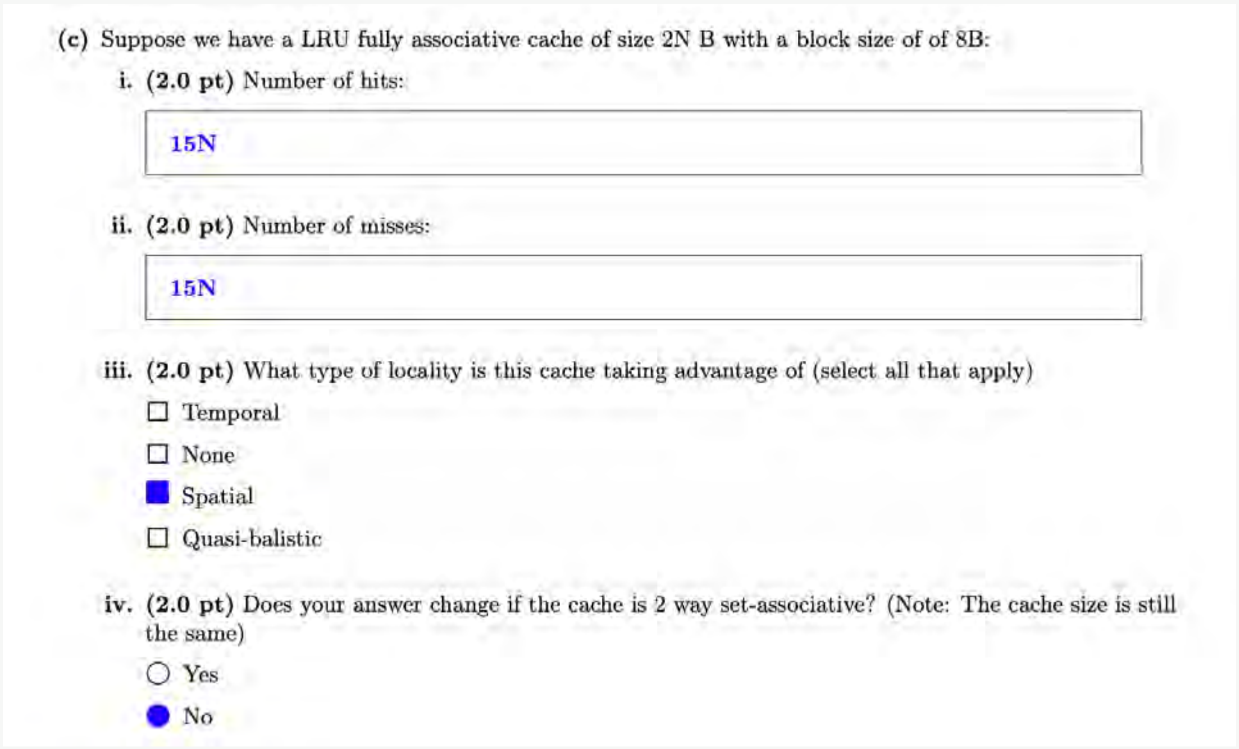
good comment | 0

Reply to this followup discussion

● Resolved ○ Unresolved @2127_f38

Anonymous Gear 3 5 months ago

SP20-Final-Q2, for all three cases, changing the cache from fully associative to 2 way set associative does not make a difference. What would be a case where changing the cache from fully associative to 2 way set associative makes a difference?



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

i. **(2.0 pt)** Number of hits:

15N

ii. **(2.0 pt)** Number of misses:

15N

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

☐ Temporal

☐ None

[✓ Spatial]

☐ Quasi-balistic

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

○ Yes

[✓ No]

helpful! | 0

Yiteng Zhou 5 months ago

Good question, in this sequencial loop, fully associative or direct-mapping won't make difference cuz you will evict the same order as LRU.

However, the difference appears when you have an unsequencial access, like, you have a 4 block cache. (the index bit is 0 for fully associative and 1 for 2-way and we don't consider cache size and others for simplicity) your access's block is a, b, c, d, which have the index bit 0, 1, 0, 0. If you use fully associative, they will both be stored in the cache, but 2-way associative will only have b, c and d, cuz a is evicted due to conflict.

helpful! | 0

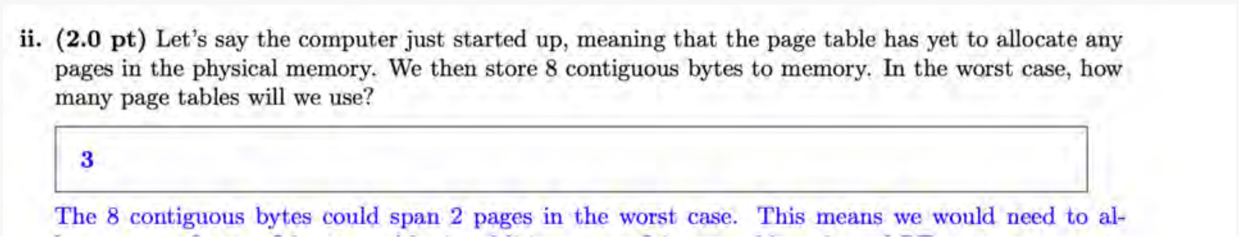
Reply to this followup discussion

● Resolved ○ Unresolved @2127_f39

Anonymous Scale 3 5 months ago

summer 2020, P1

Why we would span max page of 2 for L2 page table?



ii. **(2.0 pt)** Let's say the computer just started up, meaning that the page table has yet to allocate any pages in the physical memory. We then store 8 contiguous bytes to memory. In the worst case, how many page tables will we use?

3

The 8 contiguous bytes could span 2 pages in the worst case. This means we would need to allocate space for two L2 pages tables in addition to one L1 page table = 3 total PTs.

helpful! | 0

Adelson Chua 5 months ago
@2127_f16

good comment | 0

Anonymous Scale 3 5 months ago
I still don't get it. I know we for sure need 1 page for level 1. In terms of worst case, like the one we see on HW, should L2 consist of 8? So in total 9 page table...

helpful! | 0

Adelson Chua 5 months ago
I'm guessing it is because the memory is typically in groups of 4 bytes. So you split the 8 bytes into two 4 bytes, which are then placed into different sections in memory that would require 1 L2 page table each.

good comment | 1

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f40

Anonymous Scale 3 5 months ago
what is the difference between keyword reduction and private?

helpful! | 0

Peyrin Kao 5 months ago
This is covered in more detail in Lecture 20. The private keyword gives each thread a unique copy of a variable that gets deleted after the threads finish. The reduction keyword gives each thread a unique copy of a variable, then combines the copies into one final value after the threads finish.

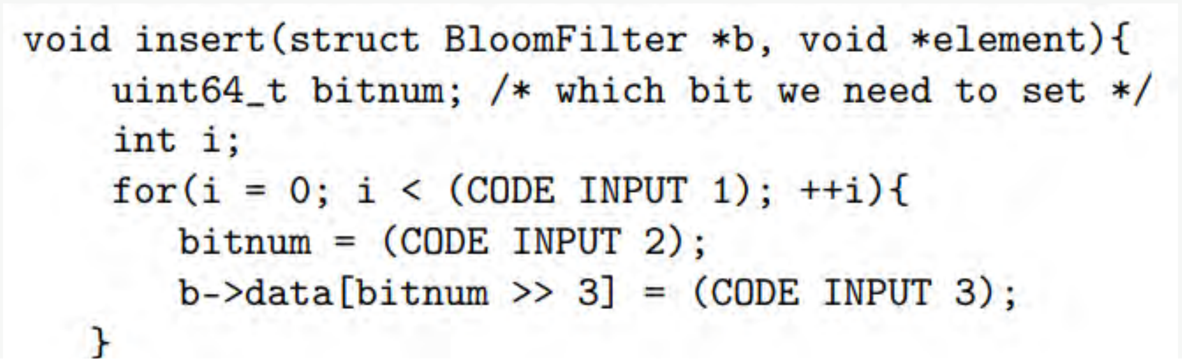
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f41

Yiteng Zhou 5 months ago
SP20-Final-Q5 (c) part ii.

Apart from I'm really confused that what the struct BloomFilter does.



```
void insert(struct BloomFilter *b, void *element){
    uint64_t bitnum; /* which bit we need to set */
    int i;
    for(i = 0; i < (CODE INPUT 1); ++i){
       bitnum = (CODE INPUT 2);
       b->data[bitnum >> 3] = (CODE INPUT 3);
   }
}
```

I have no idea what does the code do. I even can't very clearly understand the question. What's going on here? :(Can someone explain what the question asks?

(Also, I think in a previous midterm there's also a question that I cannot understand at all)

helpful! | 0

Peyrin Kao 5 months ago
Yeah, I agree that the wording of this question was pretty confusing. We'll try to be clearer on our exam.

@2127_f4 explains a few subparts, feel free to follow up if you have more specific questions.

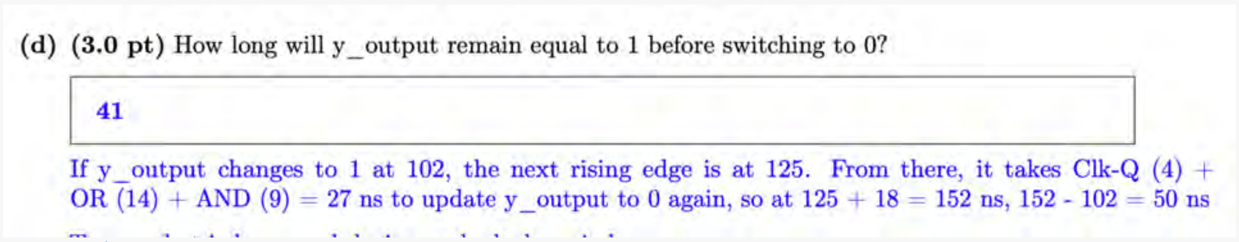
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved **@2127_f42**

Anonymous Scale 3 5 months ago
So the ans will be 50 in this case? not sure how 41 comes from

summer 2020 final Q4



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

41

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

helpful! | 0

Peyrin Kao 5 months ago
Solutions are wrong, answer is indeed 50.

good comment | 0

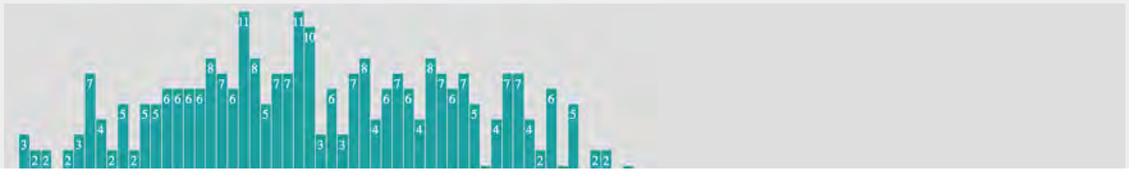
Reply to this followup discussion

[✓ Resolved] Unresolved **@2127_f43**

Anonymous Poet 3 5 months ago
Would it be possible to post the exam statistics for the SP20 Final? I'm curious to see what the average was. I also don't think it's on the other post with the other exam statistics. Thank you!

helpful! | 0

Peyrin Kao 5 months ago
Note that these numbers might be skewed as spring 2020 was a semester where P/NP counted for graduation requirements, but if you're curious:



MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
0.0	52.5	112.0	54.66	26.26

Note that the exam was out of 121.

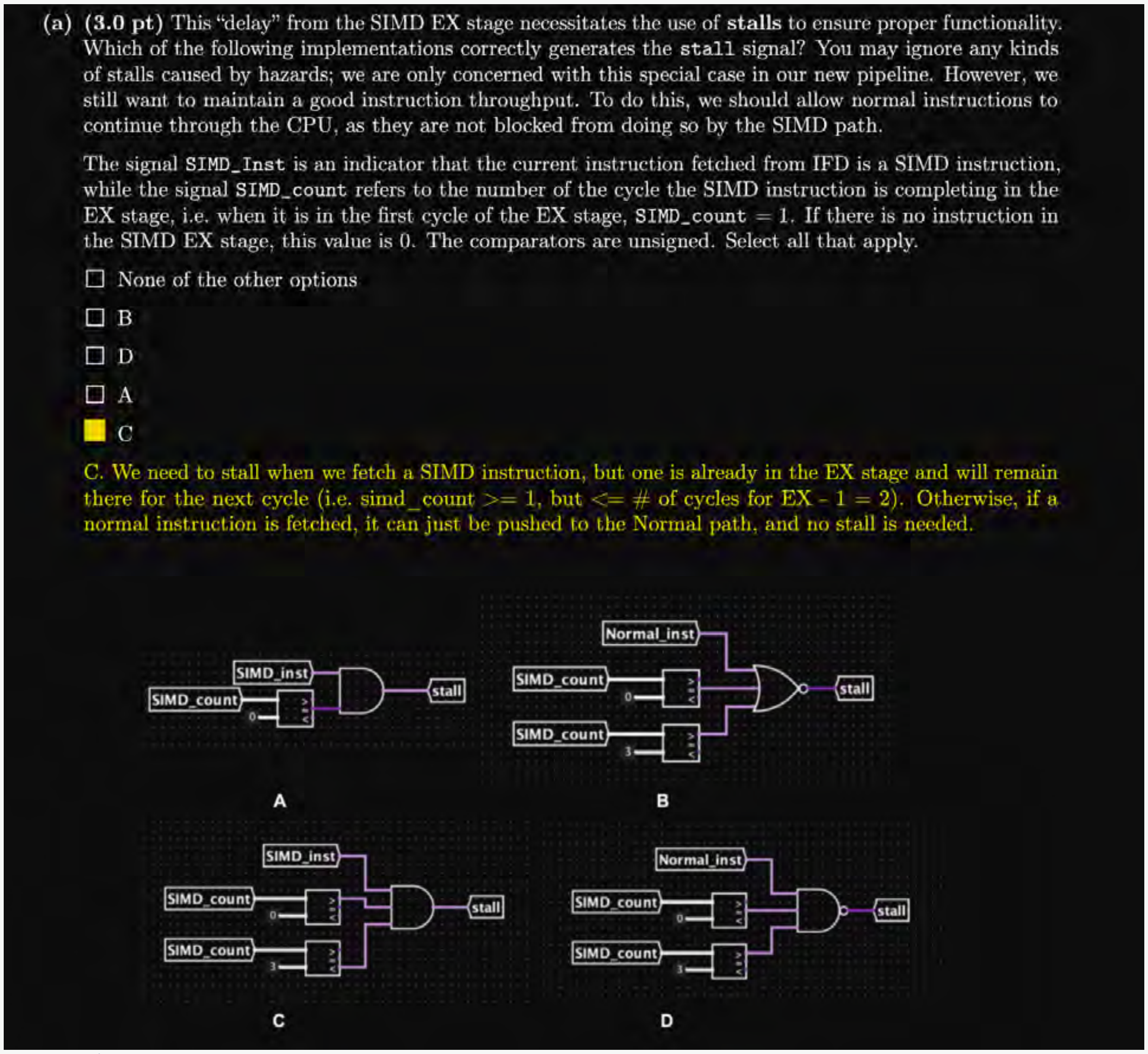
good comment | 1

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f44

Mallika Parulekar 5 months ago

SU 20 Q3 (a) Why isn't the circuit B also not valid in addition to C? It seems like it has the same boolean logic? Thanks!



(a) **(3.0 pt)** This "delay" from the SIMD EX stage necessitates the use of **stalls** to ensure proper functionality. Which of the following implementations correctly generates the `stall` signal? You may ignore any kinds of stalls caused by hazards; we are only concerned with this special case in our new pipeline. However, we still want to maintain a good instruction throughput. To do this, we should allow normal instructions to continue through the CPU, as they are not blocked from doing so by the SIMD path.

The signal `SIMD_Inst` is an indicator that the current instruction fetched from IFD is a SIMD instruction, while the signal `SIMD_count` refers to the number of the cycle the SIMD instruction is completing in the EX stage, i.e. when it is in the first cycle of the EX stage, `SIMD_count` = 1. If there is no instruction in the SIMD EX stage, this value is 0. The comparators are unsigned. Select all that apply.

☐ None of the other options

☐ B

☐ D

☐ A

[✓ C]

C. We need to stall when we fetch a SIMD instruction, but one is already in the EX stage and will remain there for the next cycle (i.e. simd_count >= 1, but <= # of cycles for EX - 1 = 2). Otherwise, if a normal instruction is fetched, it can just be pushed to the Normal path, and no stall is needed.

helpful! | 0

Peyrin Kao 5 months ago

~~B uses a NOR gate, so stall is 1 when all three inputs are 0. A uses an AND gate, so stall is 1 when all three inputs are 1.~~ Missed that the inputs to the gate are also somewhat different, see answer below

good comment | 0

Mallika Parulekar 5 months ago

Wouldn't all of the inputs here be zero though in the case that we want to return 1 for stall? It's not a normal instruction, SIMD count is not equal to 0 and SIMD count is not greater than 3?

helpful! | 0

Peyrin Kao 5 months ago

When it's a SIMD instruction and SIMD_count = 3, option C correctly does not stall (because 3 is not less than 3), but option B incorrectly chooses to stall (because 3 is not equal to 0 and not greater than 3).

good comment | 1

Mallika Parulekar 5 months ago

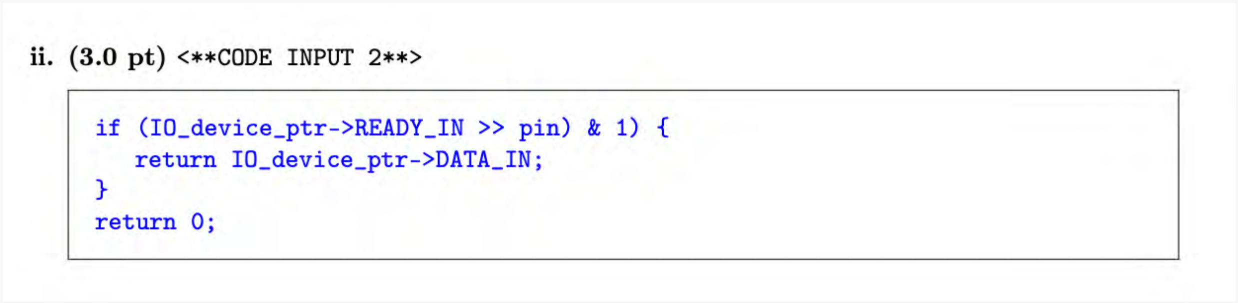
ah I completely missed that - thanks!

helpful! | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f45

Anonymous Beaker 2 5 months ago



ii. (3.0 pt) <**CODE INPUT 2**>

```
if (IO_device_ptr->READY_IN >> pin) & 1) {
    return IO_device_ptr->DATA_IN;
}
return 0;
```

How does shifting by a pin work? I'm confused at what the purpose of that is?

helpful! | 0

Peyrin Kao 5 months ago

According to the question, `READY_IN` is a bitstring where "the i-th bit indicates whether or not the device has a value at pin i that should be read by the computer via a 1 or 0, respectively."

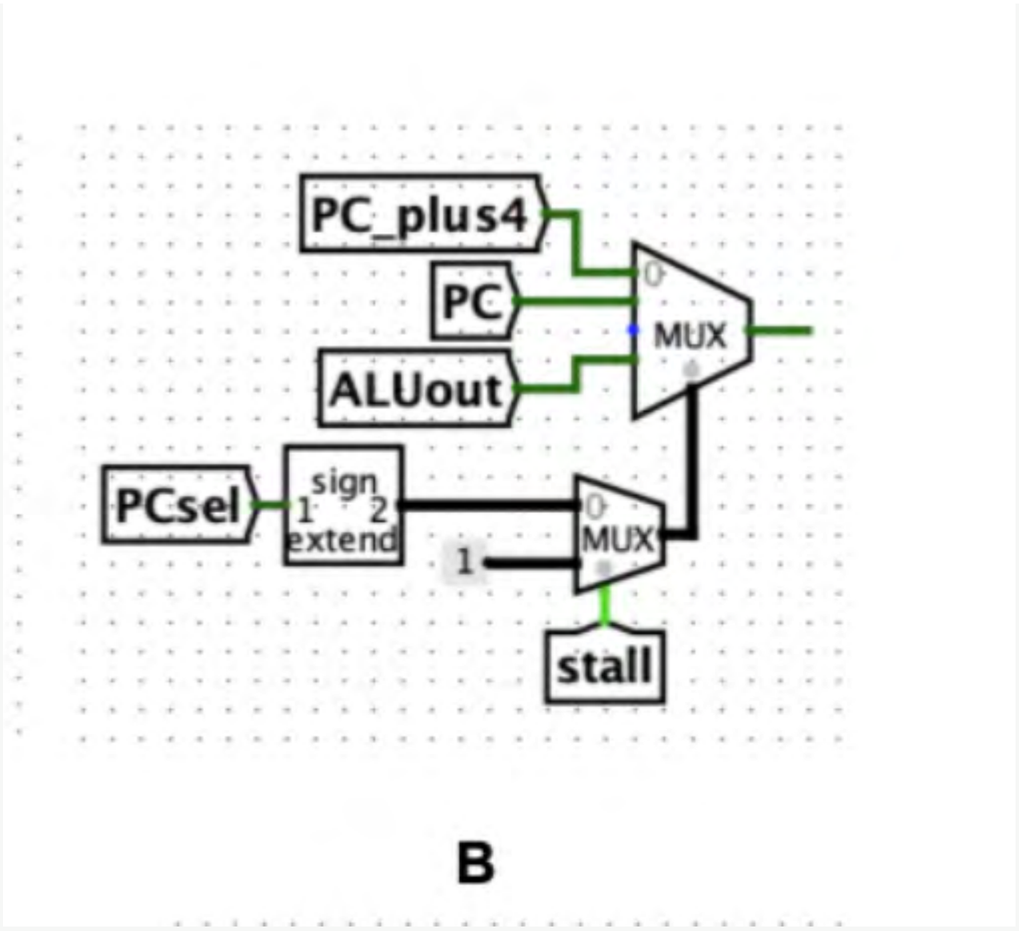
This fuunction gives us `int pin` as the number of the i-th pin we want to check. To check the i-th bit of `READY_IN`, we can right-shift the value by i so that the i-th bit becomes the lowest bit of the value, then AND the value with 1 so that only the lowest bit of the value is selected.

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f46

Anonymous Beaker 2 5 months ago



For Su20 q3b, isn't b also invalid because sign extending PCsel would be either 00 or 11?

helpful! 0

Adelson Chua 5 months ago

00 would correspond to PC+4 input.

11 would correspond to ALUout input.

These are the typical inputs of the PCmux without any stalling support (these are selected if stall = 0)

If stall = 1, the mux selects PC, which creates the stall.

This is correct.

good comment 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f47

Anonymous Comp 3 5 months ago

FA-2020: Section 2, 3b:

Hi, I am not sure at how this question was done. I would appreciate a walkthrough of how the next state and output were achieved.

helpful! 0

Adelson Chua 5 months ago

This is FSM, which is out of scope.

We didn't cover this in the lecture.

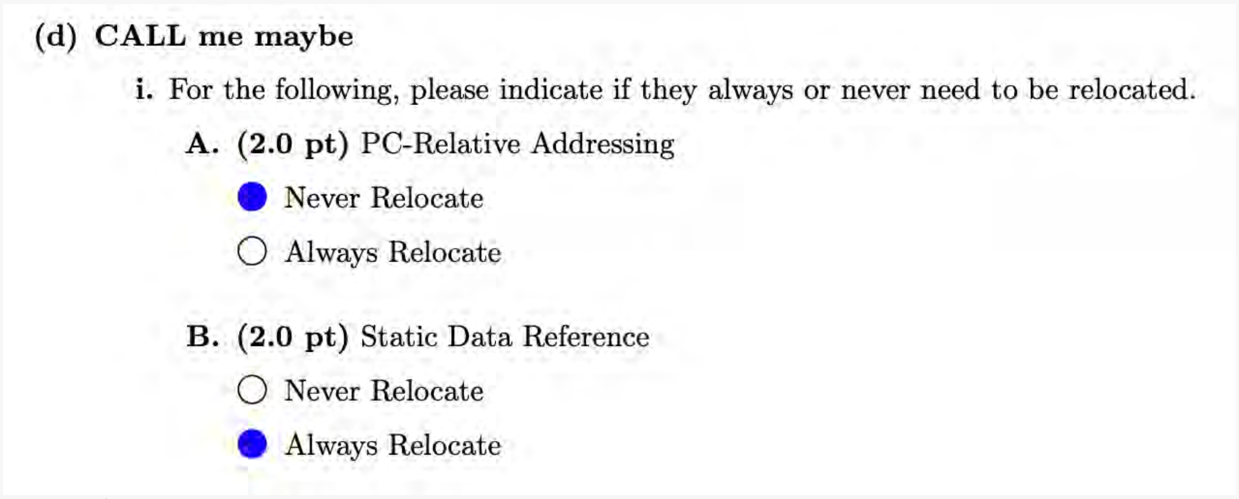
good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f48

Anonymous Gear 3 5 months ago

SP20-FINAL-Q8, where do I find lecture material related to this?



(d) **CALL me maybe**

i. For the following, please indicate if they always or never need to be relocated.

A. (2.0 pt) PC-Relative Addressing

[✓ Never Relocate]

Always Relocate

B. (2.0 pt) Static Data Reference

Never Relocate

[✓ Always Relocate]

helpful! | 0

Peyrin Kao 5 months ago

This was covered in Lecture 9 (CALL).

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @2127_f49

Anonymous Comp 3 5 months ago

FA-2020: Section 3 Potpurri A and B:

Could someone please provide a walkthrough for parts A and B? I know B has a solution, but I am unable to follow the train of thought.

helpful! | 0

Adelson Chua 5 months ago

https://www.youtube.com/watch?v=Xon5XPynv10

Go watch.

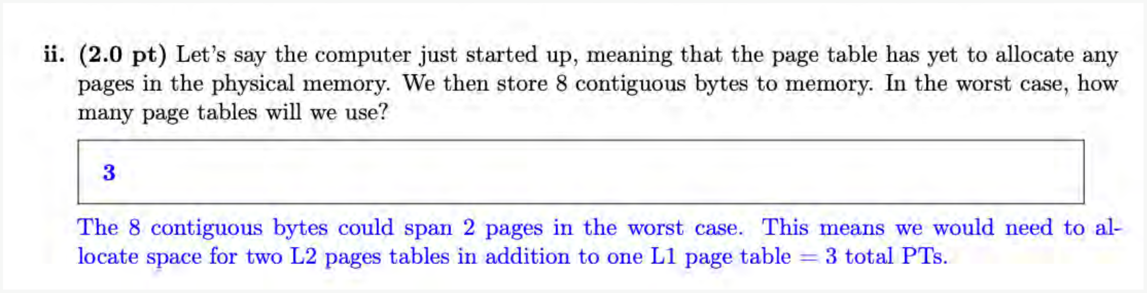
Original thread here: @2206

good comment | 0

Reply to this followup discussion

● Resolved ○ Unresolved @2127_f50

Anonymous Atom 2 5 months ago


ii. **(2.0 pt)** Let's say the computer just started up, meaning that the page table has yet to allocate any pages in the physical memory. We then store 8 contiguous bytes to memory. In the worst case, how many page tables will we use?

3

The 8 contiguous bytes could span 2 pages in the worst case. This means we would need to allocate space for two L2 pages tables in addition to one L1 page table = 3 total PTs.

su20 q1.b.ii. why does this span 2 pages in the worst case? can i get a visualization

helpful! | 0

Peyrin Kao 5 months ago

@2127_f16

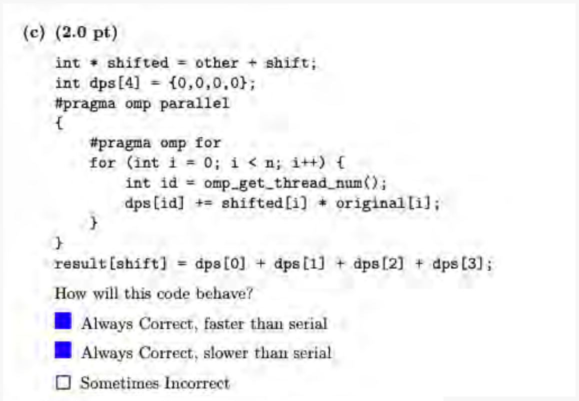
good comment | 0

Reply to this followup discussion

● Resolved ○ Unresolved @2127_f51

Anonymous Atom 2 5 months ago

su20. q2 part d


(c) **(2.0 pt)**

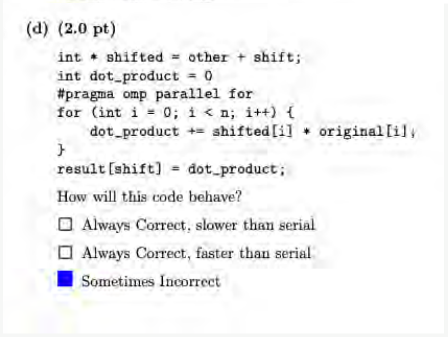
```
int * shifted = other + shift;
int dps[4] = {0,0,0,0};
#pragma omp parallel
{
    #pragma omp for
    for (int i = 0; i < n; i++) {
        int id = omp_get_thread_num();
        dps[id] += shifted[i] * original[i];
    }
}
result[shift] = dps[0] + dps[1] + dps[2] + dps[3];
```

How will this code behave?

■ Always Correct, faster than serial

■ Always Correct, slower than serial

☐ Sometimes Incorrect


(d) **(2.0 pt)**

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

How will this code behave?

☐ Always Correct, slower than serial

☐ Always Correct, faster than serial

■ Sometimes Incorrect

1. can someone please explain why d is wrong? what's causing the error?

2. how is this different from c? is c making a bunch of threads from the first pragma omp and then pragma omp for divides the work among each thread? but we have to calculate result in this process num_threads amount of times?

helpful! | 0

Adelson Chua 5 months ago

what's causing the error?

All threads are writing to the shared variable dot_product, which causes a data race.

how is this different from c?

Each thread writes on a separate array index, no shared variable = no data race.

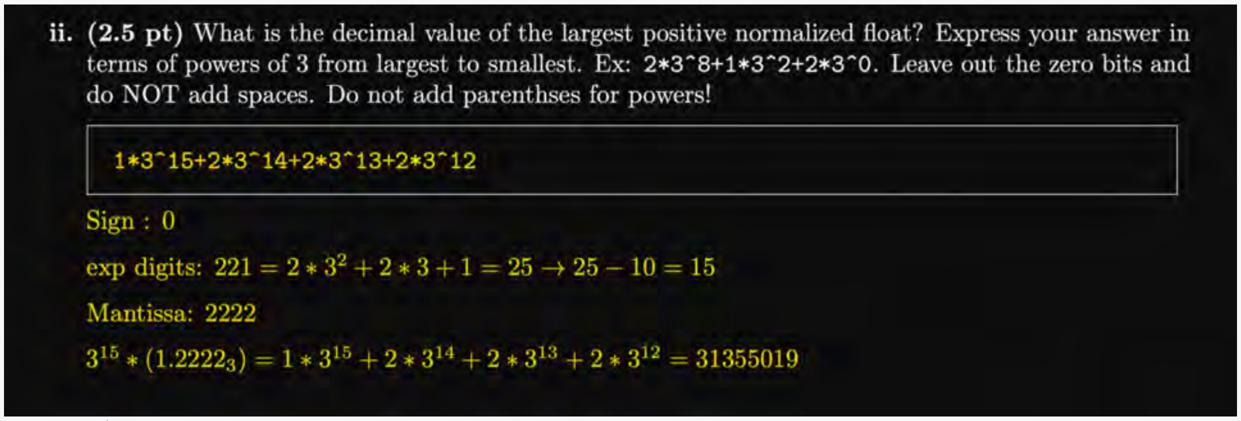
good comment 0

Reply to this followup discussion

[● Resolved] ○ Unresolved @2127_f52

Mallika Parulekar 5 months ago

Shoudn't there be a +2^11 in this for the last 2 in the mantissa? Thanks!



ii. **(2.5 pt)** What is the decimal value of the largest positive normalized float? Express your answer in terms of powers of 3 from largest to smallest. Ex: `2*3^8+1*3^2+2*3^0`. Leave out the zero bits and do NOT add spaces. Do not add parenthses for powers!

```
1*3^15+2*3^14+2*3^13+2*3^12
```

Sign : 0

exp digits: $221 = 2*3^2+2*3+1 = 25 \to 25-10 = 15$

Mantissa: 2222

$3^{15}*(1.2222_3) = 1*3^{15}+2*3^{14}+2*3^{13}+2*3^{12} = 31355019$

helpful! 0

Adelson Chua 5 months ago

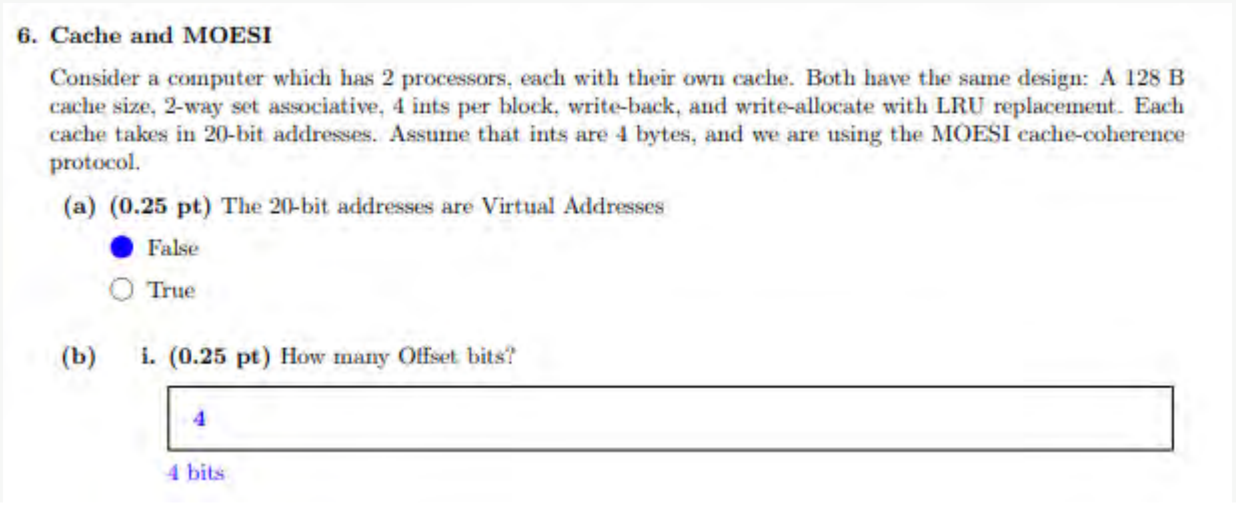
Looking at this screenshot alone, yes you are correct, there should be + 2 * 3^11 in there

good comment 0

Reply to this followup discussion

[● Resolved] ○ Unresolved @2127_f53

Anonymous Atom 3 5 months ago



6. Cache and MOESI

Consider a computer which has 2 processors, each with their own cache. Both have the same design: A 128 B cache size, 2-way set associative, 4 ints per block, write-back, and write-allocate with LRU replacement. Each cache takes in 20-bit addresses. Assume that ints are 4 bytes, and we are using the MOESI cache-coherence protocol.

(a) **(0.25 pt)** The 20-bit addresses are Virtual Addresses

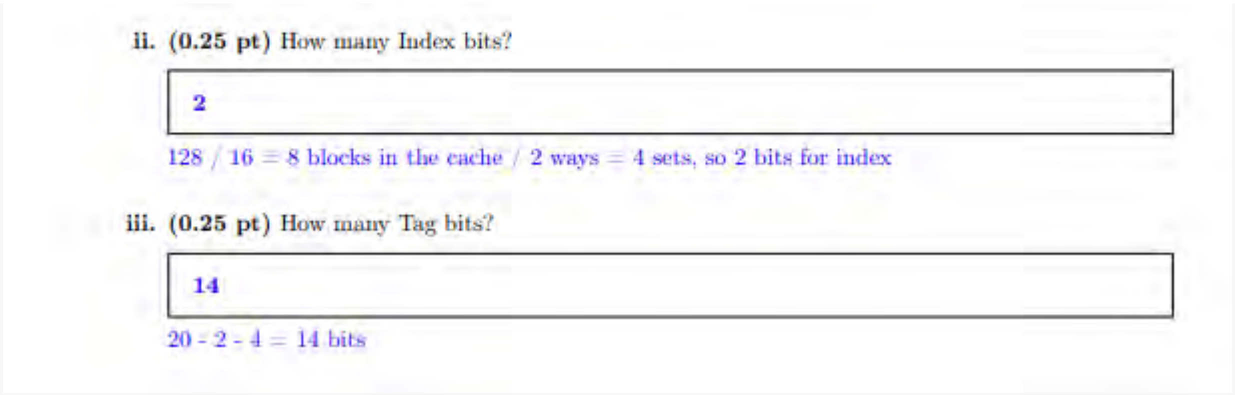
[✓ False]

○ True

(b) **i.** **(0.25 pt)** How many Offset bits?

```
4
```

4 bits



ii. (0.25 pt) How many Index bits?

2

128 / 16 = 8 blocks in the cache / 2 ways = 4 sets, so 2 bits for index

iii. (0.25 pt) How many Tag bits?

14

20 - 2 - 4 = 14 bits

How did we get 4 for the offset bits? Also why are the 20 bit addresses not virtual addresses? Are the caches the question is referencing not the TLB?

helpful! | 0

Anonymous Atom 3 5 months ago
^Su20-Final-Q6

helpful! | 0

Adelson Chua 5 months ago
Problem says 4 ints per block = 4*4bytes per block = 16 bytes per block.

Regarding virtual addresses: @2229_f9

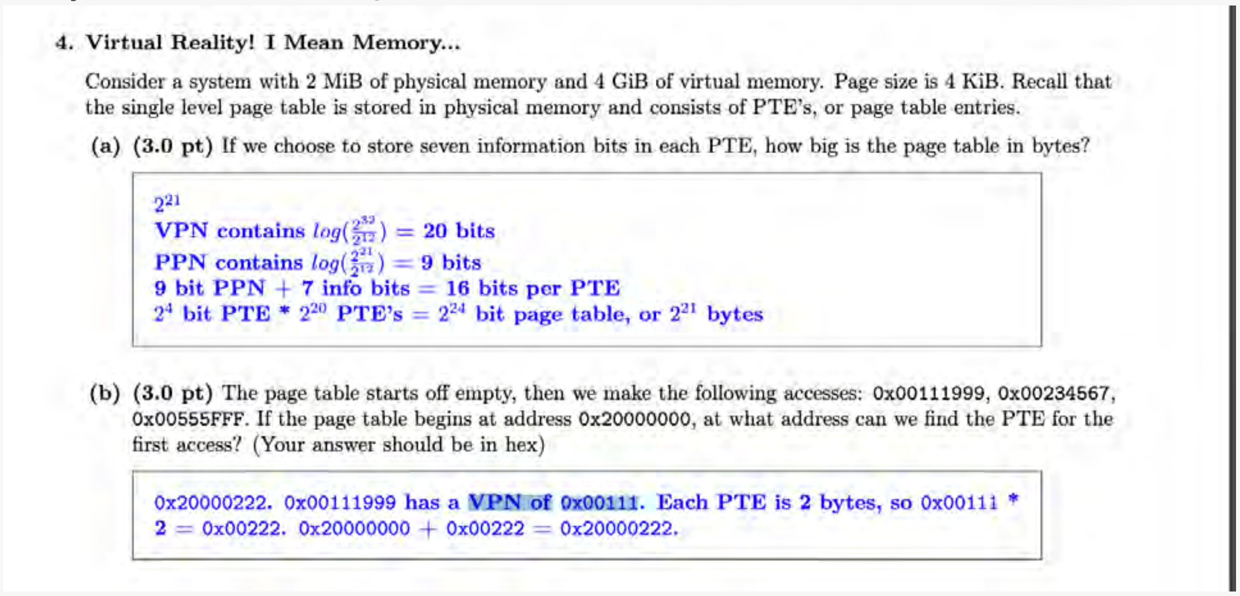
Caches in question is not the TLB.

good comment | 1

Reply to this followup discussion

[✓ Resolved] Unresolved **@2127_f54**

Anonymous Helix 3 5 months ago



4. Virtual Reality! I Mean Memory...

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

(a) (3.0 pt) If we choose to store seven information bits in each PTE, how big is the page table in bytes?

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

(b) (3.0 pt) The page table starts off empty, then we make the following accesses: 0x00111999, 0x00234567, 0x00555FFF. If the page table begins at address 0x20000000, at what address can we find the PTE for the first access? (Your answer should be in hex)

0x20000222. 0x00111999 has a VPN of 0x00111. Each PTE is 2 bytes, so 0x00111 * 2 = 0x00222. 0x20000000 + 0x00222 = 0x20000222.

For 2b why doesn't the address of the first entry just start at 0x2000111? We have a vpn of 0x00111. I was thinking that the first entry should start at the beginning. Is this because PTE is stored in like little - endian? Thanks!

helpful! | 0

Adelson Chua 5 months ago

VPN indexes the page table right? That's why we have 2^VPN number of rows in the page table. But it is given that PTE is 2 bytes.

Recall that memory is byte-addressable. So the actual memory address is at whatever index that is * 2 bytes.

Endianness does not matter here.

good comment | 0

Reply to this followup discussion

Start a new followup discussion

Compose a new followup discussion