

[Final] Past Exams - 2020 and Older #1041

J

Jero Wang STAFF
12 months ago in **Exam - Final**

799
VIEWS



You can find the past exams here: <https://cs61c.org/fa23/resources/exams/>. Please check the linked past Piazza/Ed Q&A PDFs first before asking here. Many of the questions are already answered in those! Video walkthroughs (if available), are also linked on that page!

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

Semester-Exam-Question Number

For example: **SP22-Final-Q1**, or **SU22-MT-Q3**



Elana Ho 12mth #1041adf ✓ Resolved

b) Mark the following questions as either **True** or **False**:

<input checked="" type="radio"/> True <input type="radio"/> False	If we have a TLB which contains a number of entries equal to $\text{MEMORY_SIZE} / \text{PAGE_SIZE}$, every TLB miss will also be a page fault.
<input type="radio"/> True <input checked="" type="radio"/> False	If we change our TLB to direct-mapped, we're likely to see fewer TLB misses.
<input type="radio"/> True <input checked="" type="radio"/> False	Every TLB miss is equally expensive in terms of the amount of time it takes for us to resolve our virtual address to a physical address.
<input type="radio"/> True <input checked="" type="radio"/> False	A virtual address will always resolve to the same physical address.

True If we have a TLB which contains a number of entries equal to $\text{MEMORY_SIZE} / \text{PAGE_SIZE}$, every TLB miss will also be a page fault.

for fall 2018 final QF1b, why is the first one true?



Andy Chen STAFF 12mth #1041aea

I think MEMORY_SIZE is meant to be the size of virtual memory, so $\text{MEMORY_SIZE} / \text{PAGE_SIZE}$ would be equal to the number of virtual pages. The page table contains a mapping/entry for every virtual page. This means that the TLB can hold all of the information in the page table, so there will never be any information that is not in the TLB (TLB miss) but in the page table.



Anonymous Viper 12mth #1041adc Unresolved

SU20-Final-Q6ciii)

Why is the state from proc 0's point of view owner and not shared. I narrowed it down to either shared or owner and my thought process was that since shared says we haven't modified this block which we haven't and that we are not allowed to make modifications which process 0 can't since it only handles even indices it should be shared. Owner on the other hand necessitates that it is the only thread allowed to modify it, how can that be the case for proc 0 which only handles even indices?

♡ ...



Anonymous Viper 12mth #1041aee

Any chance anyone knows why this is the case?

♡ ...



Anonymous Viper 12mth #1041adb

Unresolved

SU20-Final-Q4b)

My thought process was that the minimum clock period is longest CL + tclktoq + setup time

the setup time + tclktoq is 4+3 = 7

For the longest CL I thought it would be reg 1 going down through or gate then through and gate then wrapping around above through the other and gate thus giving a value of 14 + 9 + 9 = 32 and thus my answer was 32 + 7 = 39. Why is it that the answer is 42 and not 39?

♡ ...



A Andy Chen STAFF 12mth #1041aeb

This one is a bit tricky, you are right about the longest CL path being through the OR and AND gate but the critical path in this circuit is actually from x_input through the AND gate to the input of Reg1. For this path, from the rising edge of the clock, x_input changes value after 30 ns (you can think of this kind of like a clk-to-q for x_input), then it goes through the AND gate (9 ns), and then we add on the setup time (3 ns) which is 42 ns total. So in this case it's because x_input sort of has a longer "clk-to-q" type delay than the other registers.

♡ ...



Anonymous Viper 12mth #1041aec

Ah so x_input is a register in that case? I'm assuming since it says it switches value, it is essentially a flip flop so a register. That makes sense, thank you!

♡ ...



Anonymous Oryx 12mth #1041ace

Unresolved

FA19-Final-Q5 d

how do we go from line 2 to line 3. this says

$$A + (\bar{A})B = A + B$$

but i couldnt find this anywhere

$$\text{out} = \bar{C}(B + \bar{A}\bar{D} + \bar{B}\bar{D} + D) \text{ (Distributive)}$$

$$\text{out} = \bar{C}((B + \bar{B}\bar{D}) + (\bar{A}\bar{D} + D)) \text{ (Associative)}$$

$$\text{out} = \bar{C}(B + \bar{D} + \bar{A} + D)$$

$$\text{out} = \bar{C}(\bar{A} + B + (D + \bar{D})) \text{ (Associative)}$$

♡ ...




Anonymous Emu 12mth #1041acf


One of the laws of Boolean algebra is $X + YZ = (X+Y)(X+Z)$, which explains how $A + !AB = A+B$

Line 2 becomes $C[(B+!B)(B+!D)+(D+!A)(D+!D)]$
which simplified, becomes line 3

♡ ...


 **Anonymous Oryx** 12mth #1041ada
thanks!

♡ ...

 **Anonymous Chicken** 12mth #1041acd **Unresolved**
su20-final-q12

can someone please walk me through the thought process of part a and what it is asking for? I don't get it.

♡ ...

 **Anonymous Clam** 12mth #1041acc **Unresolved**
Fa19 Q9 c

Why is Translation and Data Access AMAT calculated separately? How do we know which is translation and which is Data Access. I ask this since we are accessing data throughout this process. First from TLB, then physical memory, then disk as well as L1 Cache. So why is the access from cache calculated separately?

Q9) We've got VM! Where? (15 pts = 2 + 3 + 5 + 5*1)

Your system has a 32 TiB virtual address space with a single level page table. Each page is 256 KiB. On average, the probability of a TLB miss is 0.2 and the probability of a page fault is 0.002. The time to access the TLB is 5 cycles and the time to transfer a page to/from disk is 1,000,000 cycles. The physical address space is 4 GiB and it takes 500 cycles to access it. The system has an L1 physically indexed and tagged cache which takes 5 cycles to access and a hit rate of 50%. On a TLB miss, the MMU checks physical memory next.

- a) How many bits is the Virtual Page Number?

27 bits

SHOW YOUR WORK

Number of reachable virtual addresses: $\log_2(32 \text{ TiB}) = 45$
Bits needed to reach all addresses in a page: $\log_2(256 \text{ KiB}) = 18$
So the virtual page number bits are: $45 - 18 = 27$

- b) What is the total size of the page table (in bits), assuming we have no permission bits or any other metadata in a page table entry, just the translation?

14 x 2²⁷ bits

SHOW YOUR WORK

We need to figure out the number of bits in the physical page number. It is the same method except we use the physical address space:
Number of reachable physical addresses: $\log_2(4 \text{ GiB}) = 32$
So PPN size is $32 - 18 = 14$. We do not have any metadata bits so the total number of bits in a PTE is 14. To figure out how many entries we need, we need to look at the total number of virtual page numbers we have = 27. This means we need 2²⁷ entries in the page table. This means we need a total of 14×2^{27} bits in our page table.

- c) What is the average memory access time (in cycles) for a single memory access for the current process? Assume the page table is resident in DRAM.

760 cycles

SHOW YOUR WORK

Translation AMAT = $5 + \frac{1}{2}(500 + 2/1000(1M))$
= $5 + \frac{1}{2}(500 + 2000)$
= $5 + \frac{1}{2}(2500)$
= $5 + 500$
= 505
plus
Data access AMAT = $5 + 50\% (500)$
= $5 + 250$
= 255
AMAT (overall) = $505 + 255 = 760$

♡ ...



Anonymous Clam 12mth #1041acb

Unresolved

SP2020 Final 4c

1. How do we know this is a page fault. It could have been a page hit right? If we had access to the page table, we'd be able to verify so how do we know for sure that this is a page fault in this case?

2. Also, looking at the explanation, how do we know it's a new 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 $\log_2\left(\frac{2^{32}}{2^{12}}\right) = 20$ bits
 PPN contains $\log_2\left(\frac{2^{21}}{2^{12}}\right) = 9$ 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


♡ ...

 **Anonymous Viper** 12mth #1041aca Unresolved

Su20-Final-Q1b)i)

I understand that to find the number of PTEs in L1, we need to multiply the number of L1 PTs by the number of PTE in an L1 PT. However why do we not do similar for L2 and instead multiply the number of L2 PTs by the size of the L2 PT? I also just don't get where 2^6 and 2^{12} are coming from, where were these sizes defined and if it was from part a), how does it relate (ie can someone walk through their thought process for this part?)

♡ ...

 **Anonymous Wombat** 12mth #1041aae ✓ Resolved

Fa17-Final-Q10

I think this code can cause data race. Why it is correct?

```

double omp5(double arr[], int size) {
    int n = 4;
    int res[4] = {0,0,0,0};
    // num_threads(4) runs the code block with 4
    threads.
    #pragma omp parallel num_threads(4) {
        int thread_id = omp_get_thread_num();
        for (int i = thread_id; i < size; i+=n) {
            if (arr[i] != 0)
                res[thread_id]++;
        }
    }
    return res[0] + res[1] + res[2] + res[3];
}

```

♡ ...



Justin Yokota STAFF 12mth #1041abe

Each thread accesses a different index of the array, so no data race is possible. It will, however, cause a lot of thrashing and cache misses, so this code is likely slow.

♡ ...



Anonymous Emu 12mth #1041aad

Unresolved

SU20-Final-Q6d-e-f

The prompt of the question says: 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.

- (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
}

```

The actual code is above ^^.

How do we reach a 1/2 hit rate. I don't really understand how MOESI caches work because we didn't really go over them in depth here, but I'm assuming that if the hit rate is 1/2, that means that on each iteration of the loop, the first access to A (read) is always miss (compulsory first iteration, coherency after that), the second access (write) is always a hit. If we have coherency misses, how would it be possible to have 0 times writing to main memory? That doesn't make sense at all. If you evict a block, you go to memory and write to update its contents with the contents of the block you just evicted? Can someone explain?

Also, how do we get a 3/4 of misses are coherency misses for part e. The question prompt says that each process has its own cache. When we load ($i = 0$) that's compulsory miss because that address has never been in cache 0. When we load ($i = 1$) that's compulsory miss as well because that address has never been in cache 1. Now, when we load $i = 2$ and $i = 3$, those are coherencies, I could understand that. But wouldn't this mean that the true fraction and answer for question e would be 1/2 as well? I'm extremely confused

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

1/2

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

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

3/4

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

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

0

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

♡ 1 ...



Anonymous Emu 12mth #1041aac

Unresolved

SU20-FINAL-7d

I'm genuinely confused about every subpart in question 7 but part d doesn't really make sense to me. How did we get these answers?

- (d) (2.0 pt) After assembling `jie.s` to `jie.o` we have the following symbol table for `jie.o`. In linking `max.o` and `jie.o` we get `dan.out`. Which of the following could be true about 'sean' and 'jenny' after linking? Select all that apply.

Exam generated for `cs61c@berkeley.edu`

34

label	address
<code>sean</code>	0x061c
<code>jenny</code>	0x1620

- They are in the same segment.
- `sean` and `jenny` will have the same byte difference after linking as it did in `jie.o`.
- They are in different files.
- `sean` and `jenny` are in different sections of `jie.s`.
- None of the other options

♡ 1 ...



Anonymous Mongoose 12mth #1041aaa

✓ Resolved

SP18-Final-Q1

Part (b)(ii)

The original question asks how many valid numbers can be represented with a 4 digit base 4 number using this scheme. Would 127 be a valid answer? There are $2 * 4 * 4 * 4$ representations total, but 0 is counted twice in the form 1000 and 0000.

- (ii) Suppose rather than using a bias notation, we decide to do the following.

For each base 4 number, we will reserve the most significant digit to strictly be used as a sign bit. A digit value of 1 will indicate a negative number, and a digit value of 0 will indicate a positive number. Any other values will result in an invalid number. For instance:

$$0003_4 = +3 \quad 1003_4 = -3 \quad 2003_4 = \textit{Invalid}$$

How many valid numbers can we represent with a 4 digit base 4 number using this scheme?

(ii) Suppose rather than using a bias notation, we decide to do the following.

For each base 4 number, we will reserve the most significant digit to strictly be used as a sign bit. A digit value of 1 will indicate a negative number, and a digit value of 0 will indicate a positive number. Any other values will result in an invalid number. For instance:

$$0003_4 = +3 \quad 1003_4 = -3 \quad 2003_4 = \textit{Invalid}$$

How many valid representation can we represent with a 4 digit base 4 number using this scheme?

Solution: $2*4*4*4 = 128$

♡ ...



Justin Yokota STAFF 12mth #1041abf

This is ambiguous, I would say; some schools of thought treat +0 and -0 as different (for example, in doubles, $1/\text{inf} = +0$, while $1/-\text{inf} = -0$). This would warrant a clarification or valid ambiguity.

♡ ...



Anonymous Clam 12mth #1041fd

Unresolved

Fa19-Final-Q8

For the fourth step of part B, I'm failing to understand how a cache write can be a miss?

♡ ...



Anonymous Opossum 12mth #1041fc

Unresolved

SP18-MT2-Q4a

hi, i am having trouble understanding the solution to Q4a. I thought that since this pipeline does not use double pumping or forwarding, the ID stage can only be done after the WB stage (eg: beq s1 s2 exit, shouldn't the D stage be stalled until c6 as well as xor s1 s1 s2). Additionally, why is E stage for the beq instruction stalled until c8 instead of starting at c7? Same question for lw instruction.

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

♡ 1 ...



Anonymous Emu 12mth #1041fb

Unresolved

SP20-Final -Q5b

I'm very confused about memory locations. I had the understanding that pointers were in the stack, and the values of pointers were allocated in the heap. For example, in the example below, I thought that u is in the stack and that v is in the heap. Now, we also know that v points to the address of u (v also in the stack).

Where my confusion starts is, if v points to the address of u, then *v is just u, and we know that u is in the stack as specified above. So why does the exam solution say that u is in the heap.

Also, if u is in the stack, why would (u + 1) be in the heap. Wouldn't u + 1 also be in the stack? I'm just generally confused about what is considered to be in the stack and in the heap when it comes to pointers, as I had the understanding that pointers are in the stack, and what the pointers are pointing to are in the heap.

m generated for cs61c@berkeley.edu

(b) I Forget Where This Data Goes

For each question, select the option which best describes what an operation would evaluate to. If an operation would lead to something which is not a valid address, select "Not an Address". Consider the following snippet of a C program:

```
void foo() {
    int64_t w = 4;
    int64_t* u = malloc(100 * sizeof(w));
    int64_t** v = &u;
    ...
}
```

♡ ...



Anonymous Heron 12mth #1041fa

✓ Resolved

Fa19-Final-Q2d

The answer says 0x03 translates to ETX on the ascii table, but the ascii table provided on the reference sheet only starts at 0x20 Hex value and nothing before. Will we be expected to memorize/have written down the values before that?

d) a (`uint32_t *`) variable `c` in **little-endian** format, and we call `printf((char *) &c)`? If an error or undefined behavior occurs, write "Error". If nothing is printed, write "Blank". Please refer to the ASCII table provided on your reference sheet. For non-printable characters, please write the value in the Char column from the table. For example, for a single backspace character, you would write "BS".

ETX

SHOW YOUR WORK

Since the data is in little-endian format, the first byte printed is 0x03, which corresponds to ETX. The second character is 0x00, which is NULL, the null terminator. `printf` doesn't read past the first null terminator, so we finish printing after we write ETX. Note that the VALUE of `c` is our number in little-endian format which is why when we do `&c`, we are saying that value is a string when plugged into `printf`.

♡ ...



Catherine Van Keuren STAFF 12mth #1041abc

It looks like the FA19 final had a different reference card with 0x03 on it. You are not expected to memorize any additional ASCII, and if we include anything related to ASCII translation on the test, it will use the values in this semester's table.

♡ ...



Anonymous Goose 12mth #1041ee

✓ Resolved

Fa19-Final-Q8b

I don't understand why the fourth line in the solutions (`ARRAY[j]` write CONFLICT -> MISS) is a miss. In line 1 and 2, we put the block that contains `ARRAY[i]` into the cache, so I would assume when we access `ARRAY[i]` in the fourth line, is it a hit?

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

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

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

T:22 I:6 O:4

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

50%

```
#define LEN 2048
int ARRAY[LEN];
int main() {
  for (int i = 0; i < LEN - 256; i+=256) {
    ARRAY[i] = ARRAY[i] + ARRAY[i+1] + ARRAY[i+256];
    ARRAY[i] += 10;
  }
}
```

SHOW YOUR WORK

Offset: $\log_2(\text{block size}) = \log_2(16) = 4$
Index: $\log_2(\text{cache size} / \text{block size}) = \log_2(1 \text{ KiB} / 16) = \log_2(64) = 6$
Tag:
First find total address bits $\log_2(4 \text{ GiB}) = \log_2(4 * 2^{30}) = \log_2(2^{32}) = 32$
Then $32 - \text{Index} - \text{Offset} = 32 - 6 - 4 = 22$

SHOW YOUR WORK

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

♡ ...



Catherine Van Keuren STAFF 12mth #1041abb

For this, an example might help. Imagine the address of `ARRAY[i]` is 0x00000400. Meaning our Offset = 0b0000, our index = 0b000000 and our tag = 0b0....01. This means that when we

put our entry into the cache we're putting it into index 0 with a tag of 1. When we jump to `ARRAY[I+256]`, we're loading from address $0x400 + (256 * 4$ (times four to account for size of int)) $(0x400)$. Thus when we add these together, we end up with a new address of $0x800$. From this new address we get a Tag of $0b0...10 = 2$ and an index of still 0 meaning that it would need to go in the same place as `ARRAY[I]` and thus is a conflict miss.

♡ ...



Anonymous Rook 12mth #1041ed Unresolved

Sp-Final-Q3

How do we get 27 for the number of bits in the PPN?

♡ ...



Anonymous Swan 12mth #1041eb Unresolved

Fa18-Final-Q3

I am very confused on how to approach 3a. Where are we getting byte2+byte3 from and how does the table yield the answers in the chart?

♡ ...



Anonymous Clam 12mth #1041dd Unresolved

Su20_MT1_3b (ii and iii)

Can someone explain what we are storing when we store ra (which line address).

Also, why do we store ra?

What is the flow of this program i.e when does it reach input 5,6 and 7.

- (b) Find the length of a null-terminated string in bytes. The function should accept a pointer to a null-terminated string and return an integer. Your solution must be recursive!

```
strlen:
    __<CODE INPUT 1>__
    beq t0, zero, basecase
    __<CODE INPUT 2>__
    __<CODE INPUT 3>__
    __<CODE INPUT 4>__
    jal strlen
    __<CODE INPUT 5>__
    __<CODE INPUT 6>__
    __<CODE INPUT 7>__
    ret
basecase:
    __<CODE INPUT 8>__
    ret
```

Fill in the following:

- i. (0.75 pt) <CODE INPUT 1>

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

- ii. (0.75 pt) <CODE INPUT 2>

```
addi sp, sp, -4
```

- iii. (0.75 pt) <CODE INPUT 3>

```
sw ra, 0(sp)
```

- iv. (0.75 pt) <CODE INPUT 4>

```
addi a0, a0, 1
```

- v. (0.75 pt) <CODE INPUT 5>

```
addi a0, a0, 1
```

- vi. (0.75 pt) <CODE INPUT 6>

```
lw ra, 0(sp)
```

- vii. (0.75 pt) <CODE INPUT 7>

```
addi sp, sp, 4
```

♡ ...



Anonymous Trout 12mth #1041dc

Unresolved

Fall-2019-Q9c

Why do we not need to consider page faults occurring for TLB hits? If page faults never occur for TLB hits, then why do we have a valid bit? Wouldn't valid bits be unnecessary if they just tell if the page is a resident of DRAM but page faults never occur since they're available in the TLB?

Thanks!

♡ ...

R **Raiyan Rizwan** 12mth #1041ec

I believe it would not be considered a TLB hit if the valid bit is 0. Hence, the valid bit being off would be captured by the miss category (please correct me if I'm wrong).

♡ ...

Anonymous Grouse 12mth #1041da **Unresolved**

SP15-Final-F1C:

Why would separating the 2 loops allow the array_size to be larger? Don't we still have the array a in memory even if it's done be written to? Also, do we not worry about page table taking up physical memory? It will be really helpful if someone can walk me through the thought process for this problem.

♡ ...

Anonymous Wombat 12mth #1041ce **Unresolved**

Fa20-MT-Q5C

When applying forwarding to store instruction, which stage(EX or MEM) in the store instruction accepts the forwarded value?

For 2-3: Data, 2, Data, 2 in Version 1, why in case 2 the answer is Data, 2? I think forwarding can solve the data hazard?

♡ ...

Anonymous Bear 12mth #1041cd **Unresolved**

Su19-MT1-Q3:

1.

```
/* Function that takes in an array of integers, mallocs space for a new array,
 * and copies the integers from the first array into the new array. It returns
 * the new array.*/
int* copy_ints(int* arr) {
    /* Allocates space to store all integers in arr. */
    [ ] int* new = malloc(sizeof(arr)); // Can't use sizeof (arr), need to pass in a len

    /* Iterates over all the elements in arr. */
    [ ] for (int i = 0; i < sizeof(arr); i++) { // Can't use sizeof (arr), need a len

        /* Loads an element from arr and stores it in new. */
        [ ] *(new + i) = *(arr + i);
    }

    /* Returns a pointer that can be dereferenced in other functions. */
    [ ] return new;
}

[ ] no errors
```


What is the 'len' mentioned in this problem? I assumed that sizeof(arr) would calculate the total length in bytes, which would be sufficient for alloc'ing memory to store all integers


♡ ...

Anonymous Tarsier 12mth #1041cf


IIRC sizeof doesn't work for pointer arrays on the heap.

♡ 1 ...

 **Anonymous Bear** 12mth #1041ff
thanks!
♡ ...


 **Anonymous Oryx** 12mth #1041cb ✓ Resolved
FA19-Final-Q4


Why does adding 0x80 to t5 and storing it in t6 change the instruction at 'loop'. also, why does it never hit 0 (referring to part b). on another note, i dont understand c or d either
♡ ...


 **Raiyan Rizwan** 12mth #1041de
The program is stored in memory, so t6 contains the absolute address in memory of the part of the program that has that instruction. This means that changing 0(t6) changes the instruction itself (correct me if I'm wrong); I did not get that while taking the exam either lol.

The code doesn't hit zero because as you increment the registers, at x10 you end up altering a0 in that lw / addi / sw to be equal to zero. This is before you then subtract one from a0 which makes it -1... and so on.


Unfortunately, I didn't really bother with c or d. Let me know if you figure it out :).
♡ ...


 **Anonymous Oryx** 12mth #1041ea
ah i see. i forgot that x10 refers to a0 lol. c seems like a straightforward extension of what happens after b. since a0 doesnt hit 0 and goes negative, we keep running the loop till Integer.MIN_Value is reached
thanks
♡ 1 ...

 **Anonymous Jaguar** 12mth #1041ca Unresolved
SU18-Final-Q3b What are the limits on the values that the addresses can be? how are they determined?
♡ ...

 **Anonymous Wombat** 12mth #1041bf ✓ Resolved
Su20-Final-Q2

Why (a) is incorrect and (b) is slower than serial?
♡ 2 ...

 **Anonymous Emu** 12mth #1041aab
+1
♡ ...

 **Justin Yokota** STAFF 12mth #1041abd
a) dot-product is defined as a private variable, so the public version never gets updated
b) The critical segment is only run by one thread at a time, so that part bottlenecks everything.
♡ ...

Anonymous Rhinoceros 12mth #1041be Unresolved

Sp19-Final-Q2d

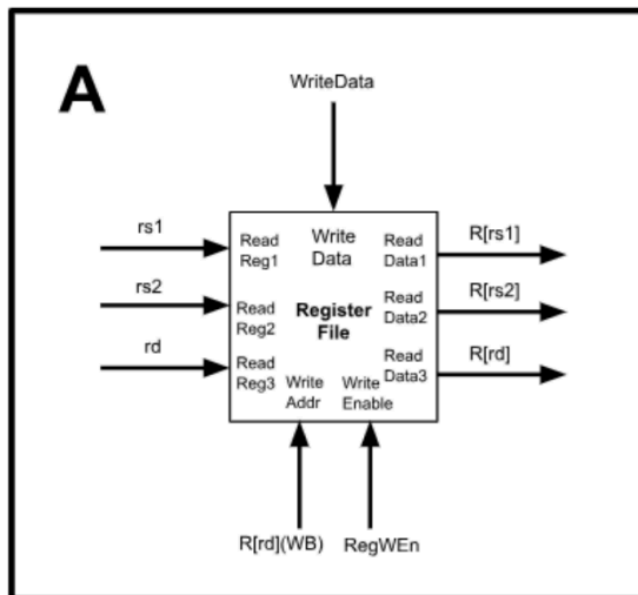
Can I get an explanation on why the answer to this question is 2 pages? Is a "page table page" the same as a page table entry?

♡ ...

Anonymous Rhinoceros 12mth #1041bd ✓ Resolved

Sp19-Final-Q2a

What does Write Addr mean for this question. Also, what is R[rd](WB)? This is for the bottom left arrow/input.



♡ ...

Sonika Vuyyuru STAFF 12mth #1041ef

"Write Addr" refers to write address, meaning the address that the data is being written to if you are writing to the Reg File. R[rd](WB) means the value in the rd register of the write-back stage. This syntax is explained in the problem description.

♡ ...

Anonymous Rhinoceros 12mth #1041bc Unresolved

Sp19-Final-Q1c (This question builds off the info in part (b))

For this question, my thought process is that after every 60 ms interval, we poll, which takes 1 ms of overhead. Therefore, the overall overhead = $1/(60+1) * 100\% = 1/61 * 100\%$. However, the answer is 1/60. Why is that? Is overhead included in the polling interval of 60 ms?

(c) If there is *no* network traffic, what percentage of the CPU will be spent polling?

Overhead = _____ %

Solution: 1.66% or $(1/60) * 100$

♡ ...



Anonymous Swallow 12mth #1041ba

✓ Resolved

SU18-Final-Q1.2

Suppose that the staff has implemented a function `frame_size` which takes in a C function's name as a string and outputs how many **bytes** that the function's local variables would need in the function frame:

```
int frame_size (char *func_name) {
    \\ Assume this function has been correctly implemented for you.
}
int main_stacks (int argc, char *argv[]) {
    char arr[] = "61C rox";
    return 0;
}
```

2) What would `frame_size ("main_stacks")` return?

2 * 4B + 8B = 16 B

How was the answer obtained? I suppose the 8B comes from the 7 char of the array + the string terminator. Where do the other 8B come from though?

♡ ...



Catherine Van Keuren STAFF 12mth #1041aba

I believe it comes from the two passed in arguments `argc` and `argv`.

♡ 1 ...



Anonymous Grouse 12mth #1041af

✓ Resolved

Su18-Final-Q12

Suppose we have a 5-disk RAID 3 system. Compared to a 5-disk RAID 5 system that is **byte striped**, fill in all the bubbles for the data request patterns / characteristics where RAID 5 **strictly outperforms** RAID 3.

- A long sequential reads
- B long sequential writes
- C recovering from 1 disk failure
- D small random reads
- E small random writes
- F capacity

Are RAID 3 and small random read /write in scope? I don't remember they are covered in lecture.

♡ ...



Catherine Van Keuren STAFF 12mth #1041aaf

Raid 3 is out of scope. You should typically know what small random read/writes are in comparison to long sequential read/writes when it comes to redundancy or cache replacement policies, but I would say that if the terminology isn't in lecture/discussion then it will most likely not be emphasized.

♡ ...




Anonymous Jaguar 12mth #1041ad

Unresolved

SP18-Final-Q12-h

Does anyone have any good resources to look at for problems like these? I'm really lost on how they did this, given that we only have the virtual address


♡ ...

 **Anonymous Jaguar** 12mth #1041ac Unresolved

SP18-Final-Q12-c

Would anyone be able to explain how to determine the address layout breakdown? Thank you!

♡ ...

 **Anonymous Jaguar** 12mth #1041aa Unresolved

SP18-Final-Q5-a and b

Why do the calculation not take the clk-to-q and setup times into consideration when calculating the hold/clock periods?

♡ ...

 **Anonymous Jaguar** 12mth #1041ab

Also, for part d, why is it that it only takes 30ps to complete 3 clock cycles when one clock cycle is 11ps?

♡ ...

 **Anonymous Flamingo** 12mth #1041e Unresolved

FA20-Final-QM2-e

Why is the bitwise AND taken of all the character bits? I was not sure how to tackle this question.

♡ ...

 **Jero Wang** STAFF 12mth #1041f


Which question is this? Not sure what QM2-e is referring to.

♡ ...

 **Anonymous Flamingo** 12mth #1041ae

Sorry, I had a typo and meant **FA18-Final-QM2-e**

♡ ...

 **Anonymous Swallow** 12mth #1041c ✓ Resolved

SP18-Final-Q6(a-g)

I'm not sure how the answers were obtained. Any clarifications would help.

♡ ...

 **Nikhil Kandkur** STAFF 12mth #1041d

The main trick for solving these problems is drawing out the pipeline diagram for each instruction, similar to how it may have been done in discussion:

C1	C2	C3	C4	C5	C6	C7
IF	ID	EX	MEM	WB		
	IF	ID	EX	MEM	WB	
		IF	ID	EX	MEM	WB

In order to figure out where stalls are needed (assuming absence of forwarding as specified by the problem), take a look at a pair of instructions where the first one sets the value of register rd, and the second register uses the value rd for some other computation.

Then you have the case of either write-read or read-write

- write-read: similar to lecture, you are going to have to stall the first instruction such that the WB of instruction 1 and ID of instruction 2 are lined up.
- read-write: this is a little tricky, but you are going to want WB of instruction 1 to be 1 cycle before ID of instruction 2 since the WB input from instruction 1 will be output of the rd register 1 cycle later, aka when we have ID of instruction 2.

Hope this helps!

♡ 1 ...



Anonymous Swallow 12mth #1041a

✓ Resolved

SP18-Final-Q1(c)

(c) Given the following function in C:

```
int shifter(int x, int shift) {
    if (x > 0) {
        return x >> shift;
    }
    return -1 * (x >> shift);
}
```

Given y is a negative integer, and that `shifter(y, 2)` outputs 4, what is the range of values of y ?

hint: $-8 \gg 1 = -4$

Solution: $-16 \sim -13$

Apparently, I have gaps in my understanding of how the bitwise shift works.

Why $y = -15, -14$, and -13 would output 4?

I get $-1111 \gg 2 = -0011 = -3$

♡ ...



Anonymous Swallow 12mth #1041b

Never mind. 2s complement is the answer.

♡ ...