CS 61C
Summer 2020

Great Ideas in Computer Architecture

Midterm 2

INSTRUCTIONS

This is your exam. Complete it either at exam.cs61a.org or, if that doesn't work, by emailing course staff with your solutions before the exam deadline.

This exam is intended for the student with email address `cs61c@berkeley.edu`. If this is not your email address, notify course staff immediately, as each exam is different. Do not distribute this exam PDF even after the exam ends, as some students may be taking the exam in a different time zone.

For questions with **circular bubbles**, you should select exactly *one* choice.

○ You must choose either this option

○ Or this one, but not both!

For questions with **square checkboxes**, you may select *multiple* choices.

☐ You could select this choice.

☐ You could select this one too!

You may start your exam now. Your exam is due at <DEADLINE> Pacific Time. Go to the next page to begin.

Preliminaries

Please complete and submit these questions before the exam starts.

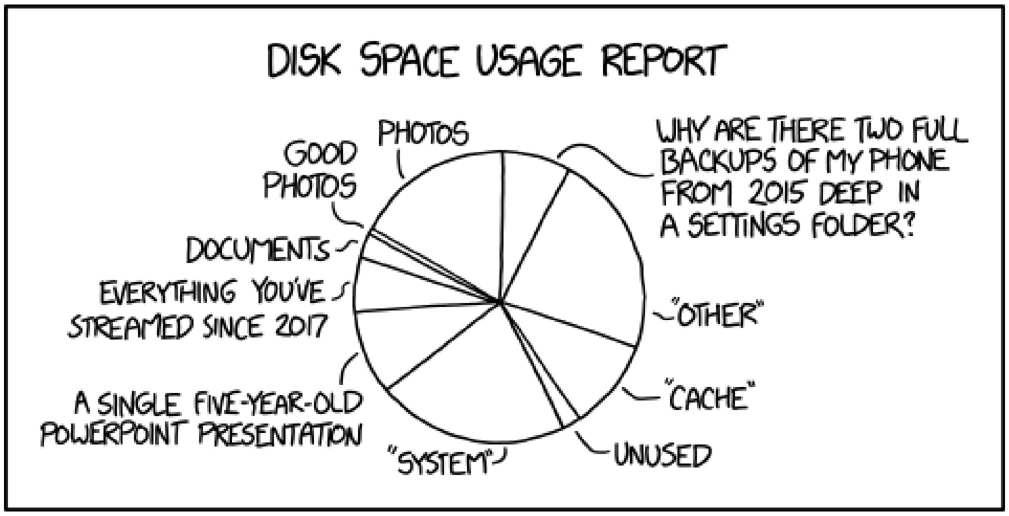
(a) What is your full name?

(b) What is your student ID number?

(c) If an answer requires hex input, make sure you only use capitalized letters! For example, 0xDEADBEEF instead of 0xdeadbeef. You will be graded incorrectly otherwise! Please always add the hex (0x) and binary (0b) prefix to your answers or you will receive 0 points. For all other bases, do not add the suffix or prefixes.

Do not add units unless the problem explicitly tells you to!

Some of the questions may use images to describe a problem. If the image is too small, you can click and drag the image to a new tab to see the full image. You can also right click the image and download it or copy its address to view it better. You can use the image below to try this. You can also click the star by the question if you would like to go back to it (it will show up on the side bar). In addition, you are able see a check mark for questions you have fully entered in the sidebar. Questions will auto submit about 5 seconds after you click off of them, though we still recommend you click the save button.


DISK SPACE USAGE REPORT
PHOTOS
GOOD PHOTOS
DOCUMENTS
EVERYTHING YOU'VE STREAMED SINCE 2017
A SINGLE FIVE-YEAR-OLD POWERPOINT PRESENTATION
"SYSTEM"
UNUSED
"CACHE"
"OTHER"
WHY ARE THERE TWO FULL BACKUPS OF MY PHONE FROM 2015 DEEP IN A SETTINGS FOLDER?

Good luck!

1. It's All an Illusion

(a) Consider a system with 4 GiB of physical memory and 64 GiB of Virtual Memory. The page size is 4 KiB. Recall that the page table is stored in physical memory and consists of PTE's, or page table entries. Please fully simplify your answer and leave it in decimal. Fully simplify your exponents down to decimal! Please round your decimal values to two places if needed (do not include unnecessary 0's).

i. (3.0 pt) If, for each PTE, we choose to also store 12 bits of metadata (e.g. permission bits, dirty bit), how many page table entries can we now store on a page?

1024

First we need to find the side of each PTE so we need to figure out how many bits of physical memory we need to have to address. $\log_2(\frac{4GiB}{4KiB}) = \log_2(\frac{2^{32}}{2^{12}}) = 20bits$

Now we can calculate the size of each PTE: 20bits (number of physical pages)+12bits (metadata bits) = $32bits = 4Bytes$

Then we need to find the size of a page: $4KiB = 4096Bytes$

Then we divide: $4096Bytes/4Bytes = 1024$

ii. Regardless of what you got for the previous part, for the next two parts, let's now assume each page could store 2048 PTEs. The page, physical, and virtual memory sizes are unchanged.

A. (3.0 pt) How many pages does our page table occupy (aka how many valid (active) pages is our page table) if we have a single level page table which has only one valid data page?

8192

Since this is a single level page table, we need to find the total number of pages we need, since we do not have compression.

We have 2^{36} addresses of virtual memory, or $\frac{2^{36}}{2^{12}} = 2^{24}$ PTEs in our PT.

Now we divide this value by the number of PTEs/page to get the number of pages we need: $\frac{2^{24}}{2048} = 2^{13} = 8192$ pages.

B. (5.0 pt) How many pages does our page table occupy (aka how many valid (active) pages is our page table) if we have a two level page table which has only one valid data page? Each level uses an equal amount of bits of the page number.

4

Since we have a two level page table, we need the top level page table and a single next level of the table. Since each level has an equal number of bits, this means each level has 12 bits. Now we need to figure out how many pages the top level will be. So we have 2^{12} page numbers and 2^{11} entries per page, so we only need 2 pages for the top level page table. Since the next level is the same size, it would also only take 2 pages. Also since the top level page table is able to show the other pages except for the one which is valid, we do not need to create any other page tables than the upper one and the single lower page table. This means we have 2 pages for L1 and 2 pages for L2 which means we need a total of 4 pages.

(b) **(1.0 pt)** Adding a TLB may make process context switching slower.

● True

○ False

Each process has its own page table. Thus, the process needs to flush the TLB on context switching which will lead to increasing the context switching time. You can have more complex TLB's which have a process's ID as the identifier for if an entry is for that page table. This reduces the overhead of needing to fully flush the TLB on a context switch. Do note that any change to the page table would still require the OS to flush the TLB.

2. Sunay-Cycle Datapath

For each instruction below, fill in the value for the specified control signals. Put `*` if the value does not matter. Each instruction executes independently of the others. Before execution of each, the `t1` register holds the value `0xFFFFFFF0` and `t2` holds `0x00000006`. For your reference (and to help prevent excessive scrolling), here are the instructions in no particular order:

```
lbu s0, 4(t1)
bge t2, t1, label
sh t1, 0(t2)
```

(a) **i.** `lbu s0, 4(t1)`

A. (0.1 pt) `PCSel`

○ *
[✓ PC + 4]
○ ALU

B. (0.1 pt) `RegWEn`

[✓ Write]
○ *
○ Do Not Write

C. (0.1 pt) `BrUn`

[✓ *]
○ 1
○ 0

D. (0.1 pt) `BrLT`

[✓ *]
○ 0
○ 1

E. (0.1 pt) `ASel`

[✓ RS1]
○ PC
○ *

F. (0.1 pt) `BSel`

[✓ Imm]
○ RS2
○ *

G. (0.2 pt) `ALUSel`

- ○ OR
- ○ SUB
- [✓ ADD]
- ○ `*`
- ○ MUL
- ○ AND
- ○ XOR

H. (0.1 pt) `MemRW`

- ○ Write
- ○ `*`
- [✓ Read]

I. (0.1 pt) `WBSel`

- ○ ALU
- ○ `*`
- [✓ Mem]
- ○ PC + 4

ii. `bge t2, t1, label`

A. (0.1 pt) `PCSel`

- ◯ *
- [✓ ALU]
- ◯ PC + 4

B. (0.1 pt) `RegWEn`

- ◯ Write
- ◯ *
- [✓ Do Not Write]

C. (0.1 pt) `BrUn`

- ◯ *
- [✓ 0]
- ◯ 1

D. (0.1 pt) `BrLT`

- ◯ *
- [✓ 0]
- ◯ 1

E. (0.1 pt) `ASel`

- [✓ PC]
- ◯ *
- ◯ RS1

F. (0.1 pt) `BSel`

- ◯ *
- [✓ Imm]
- ◯ RS2

G. (0.2 pt) `ALUSel`

- ◯ SUB
- ◯ OR
- [✓ ADD]
- ◯ MUL
- ◯ AND
- ◯ XOR
- ◯ *

H. (0.1 pt) `MemRW`

- ○ Write
- [✓ Read]
- ○ *

I. (0.1 pt) `WBSel`

- ○ ALU
- ○ Mem
- [✓ *]
- ○ PC + 4

iii. `sh t1, 0(t2)`

A. (0.1 pt) `PCSel`

- ○ ALU
- ○ *
- [✓ PC + 4]

B. (0.1 pt) `RegWEn`

- ○ Write
- [✓ Do Not Write]
- ○ *

C. (0.1 pt) `BrUn`

- ○ 1
- [✓ *]
- ○ 0

D. (0.1 pt) `BrLT`

- ○ 0
- ○ 1
- [✓ *]

E. (0.1 pt) `ASel`

- ○ PC
- [✓ RS1]
- ○ *

F. (0.1 pt) `BSel`

- ○ *
- [✓ Imm]
- ○ RS2

G. (0.2 pt) `ALUSel`

- [✓ ADD]
- ○ OR
- ○ *
- ○ SUB
- ○ MUL
- ○ AND
- ○ XOR

H. (0.1 pt) `MemRW`

- ○ Read
- ○ `*`
- [✓ Write]

I. (0.1 pt) `WBSel`

- ○ Mem
- ○ PC + 4
- ○ ALU
- [✓ `*`]

(b) Suppose that we wanted to revise the datapath to implement `mv rd, rs1` as part of the base instruction set. Fill in the appropriate control signals for the following calls to those instructions.

i. `mv s0, t1`

A. (0.2 pt) `BrLT`

○ 1

○ 0

[✓ *]

B. (0.2 pt) `BrUn`

[✓ *]

○ 1

○ 0

C. (0.4 pt) `RegWEn`

[✓ Write]

○ *

○ Do Not Write

D. (0.4 pt) `MemRW`

○ *

○ Write

[✓ Read]

E. (0.2 pt) `ASel`

○ PC

○ *

[✓ RS1]

F. (0.2 pt) `PCSel`

○ ALU

○ *

[✓ PC + 4]

G. (0.4 pt) `WBSel`

○ Mem

○ *

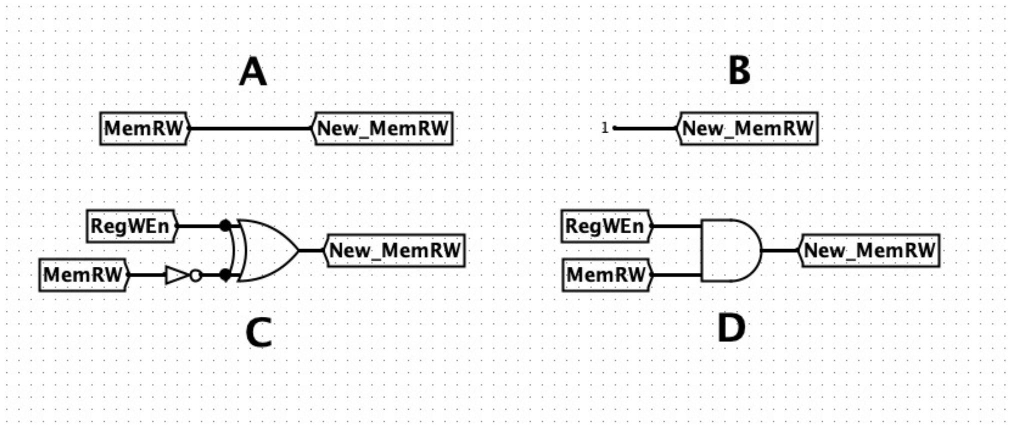
[✓ ALU]

○ PC + 4

(c) Consider the following situation: our single-cycle CPU is running a program that generates new bank account numbers for wealthy clients, and saves these to memory. Sunay wants these numbers, but doesn't have access to memory; he's only able to snoop and read what's in register x17. So, he wants to wire the hardware so that every time a store instruction is performed, the data is also written to this register. Help him rewire the CPU of this computer to steal the desired information and carry out his most glorious crime!

In this question, MemRW, RegWEn, and WBSel are the original control signals generated by the instructions, while their New versions are what is getting passed into the appropriate components. In addition, remember that Logism represents its constants as hexadecimal numbers. If you're having a hard time reading the diagrams, drag the images into a new tab to enlarge them.

i. (1.0 pt) How should he change the MemRW control signal, while keeping the functionality of all other RISC-V instructions the same? Select all valid ways.



Datapath

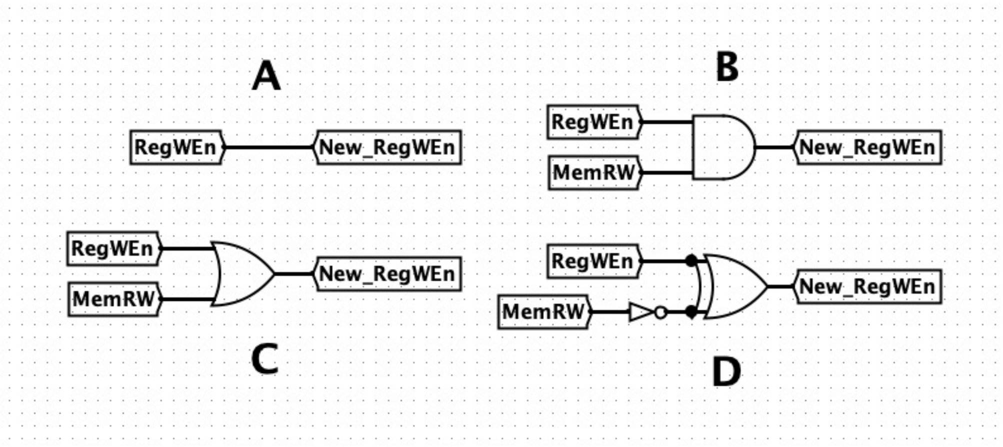
☐ D

☐ B

[✓] A

☐ C

ii. (1.0 pt) How should he change the RegWEn control signal to allow for his desired functionality, while keeping the functionality of all other RISC-V instructions the same? Select all valid ways.



Datapath

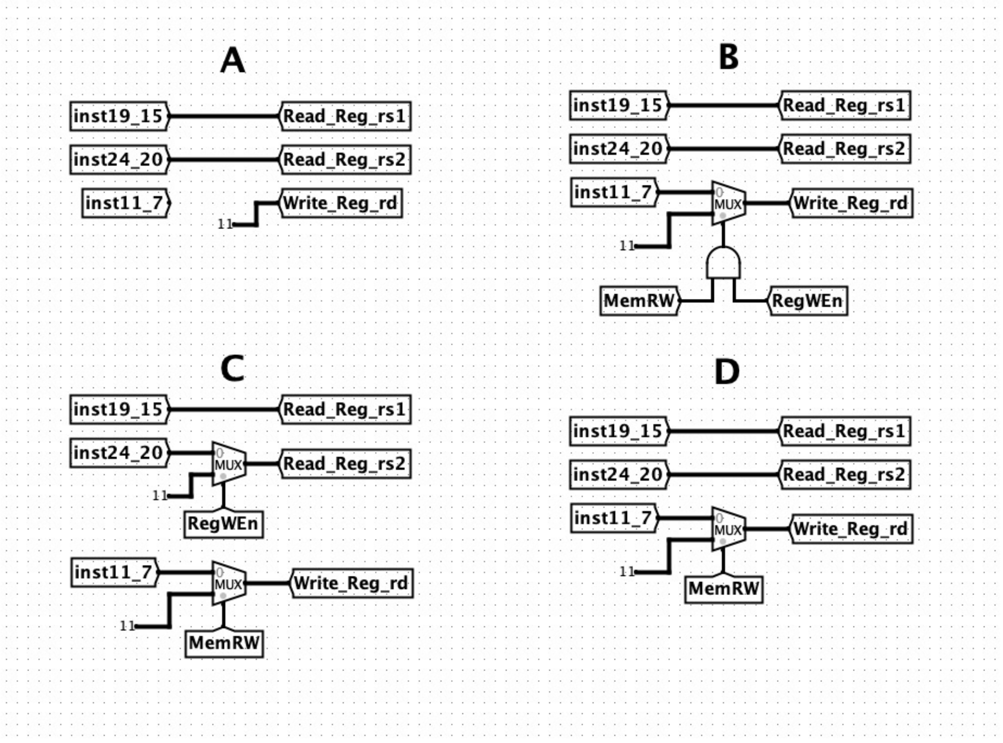
[✓ C]

☐ A

☐ D

☐ B

iii. (1.0 pt) Sunay wants to make sure that the information is being written to the correct register. How should he wire the inputs to the register file so as not to disturb normal functionality? Select all valid ways.



Datapath

☐ A

☐ C

☐ B

■ D

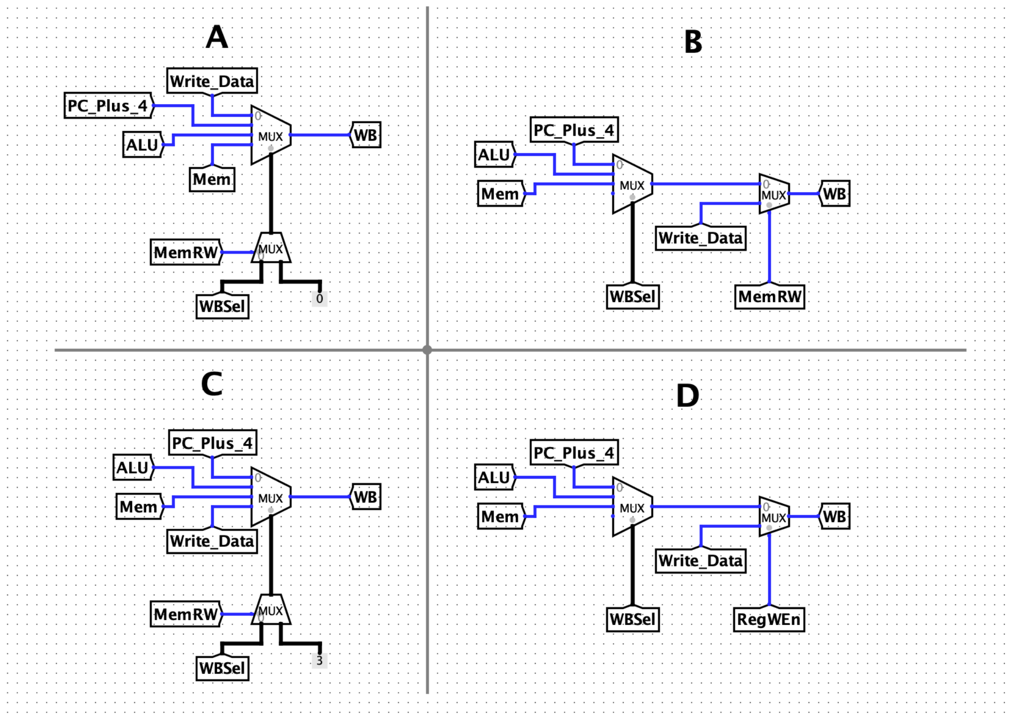
iv. (2.0 pt) How should he change the WB logic to always write the stored value back to x17, while keeping the functionality of all other RISC-V instructions the same? Select all valid ways.

■ C

☐ D

■ B

☐ A


A
Write_Data
PC_Plus_4
ALU
Mem
MUX
WB
MemRW
MUX
WBSel
0
B
PC_Plus_4
ALU
Mem
MUX
MUX
WB
Write_Data
WBSel
MemRW
C
PC_Plus_4
ALU
Mem
Write_Data
MUX
WB
MemRW
MUX
WBSel
3
D
PC_Plus_4
ALU
Mem
MUX
MUX
WB
Write_Data
WBSel
RegWEn

Datapath

3. Hazardous Situa-Seans

The economy is ruined and registers have become a scarce resource. Therefore, Sean is forced to combine stages of the standard 5-stage RISC-V pipeline to make it 4 stages instead. He chooses to combine the EX and MEM stage to create a new EM stage. Thus, our pipeline now consists of the following stages: IF, ID, EM, WB.

(a) How many stalls will it take to resolve a data hazard induced by any instruction (i.e. it hasn't stored the result back in enough time)...

i. (0.5 pt) If he only has access to forwarding?

0

1	2	3	4	5	6	7
F	D	EM	W			
	F	D	EM	W		
		F	D	EM	W	
			F	D	EM	W

Everything that can possibly be written back in the WB stage has been computed by the end of the EM stage.

ii. (0.5 pt) If he only has access to double pumping?

1

1	2	3	4	5	6	7
F	D	EM	W			
	F	D	EM	W		
		F	D	EM	W	
			F	D	EM	W

Everything that can possibly be written back in the WB stage has been computed by the end of the EM stage.

iii. (0.5 pt) If he does not have access to double pumping nor forwarding?

2

1	2	3	4	5	6	7
F	D	EM	W			
	F	D	EM	W		
		F	D	EM	W	
			F	D	EM	W

Everything that can possibly be written back in the WB stage has been computed by the end of the

EM stage.

iv. (0.5 pt) If he has access to both double pumping and forwarding?

0

1	2	3	4	5	6	7
F	D	EM	W			
	F	D	EM	W		
		F	D	EM	W	
			F	D	EM	W

Everything that can possibly be written back in the WB stage has been computed by the end of the EM stage.

(b) To cut costs, he decides to not utilize forwarding and double pumping, nor does he use a branch predictor.

```
    lw t1, 0(t0) # the value at M[t0] is nonzero
    addi s0, t1, 1
    sub s1, t1, x0
    bne s1 x0 label
    srli s2, t0, 1
    lui s3, 0xFFF
label:
    sw s1, 0(t0)
```

i. (6.0 pt) On this new pipeline, how many stalls will it take to successfully execute the code?

6

Lw

Nop

Nop

Addi

Sub

Nop

Nop

Bne -> the branch is taken so we need to flush the pipeline

Nop

Nop

Sw

6 stalls

ii. (4.0 pt) Let's do some roleplay: Sean is the compiler and he wants to minimize the number of stalls caused by the code above. If he can reorder the instructions to use a lower amount of cycles/stalls, but must keep the functionality the same, how many stalls can he remove? Note: he still has no access to any other optimizations like the previous part, nor can he change the code in any other ways than reordering

1

We can switch the sub and addi instructions, so that the data hazard induced by s1 in the branch and sub requires one less nop. Since the srli and lui instructions are never executed, we can't move them earlier to eliminate earlier stalls since that would cause different functionality from the original code. Therefore, we can only eliminate 1 stall.

4. Money

For this question, assume `sizeof(int) == 4`. For questions which ask about the hit rate, leave your answer as a fully simplified fraction with **NO** spaces! This means if you have 4/8 (4 hits over 8 accesses), you should enter 1/2; 8/8, you should enter 1; 0/8, you should enter 0. You will not get points if you do not do this! Feel free to use a fraction simplification tool online.

Assume a 4KiB cache with 64B blocks, on a machine that has 4GiB of physical memory.

(a) Specify the size (in bits) of each field for the following cache types:

i. Fully Associative

A. (0.1 pt) Tag

26

B. (0.1 pt) Index

0

C. (0.1 pt) Offset

6

ii. Direct Mapped

A. (0.1 pt) Tag

20

B. (0.1 pt) Index

6

C. (0.1 pt) Offset

6

iii. Four-way Set Associative

A. (0.1 pt) Tag

22

B. (0.1 pt) Index

4

C. (0.1 pt) Offset

6

(b) (4.1 pt) Consider the following C code:

```
int array_size = 1024;
void looptoy_loop(int *A, int *B) {
    for (int i = 0; i < array_size - 16; i += 16) {
        A[i] = A[i + 16] + B[i];
    }
```

Recall that we have an 4KiB cache with 64B blocks, on a machine that has 4GiB of physical memory. For the following questions, assume the address of `A[0]` is `0x40000000` and the address of `B[0]` is `0x80000000`. If we have an 8-way set associative cache with an LRU replacement policy, a write allocate policy for write misses, and a write back policy for write hits, what is the hit rate?

62/189

$HR = \frac{62}{63*3} = \frac{62}{189}$

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

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

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

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

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

(c) (1.5 pt) Assume the cache and block sizes are held constant. What is the minimum cache associativity that results in a hit rate no worse than the hit rate in question (ii)?

○ 8-way

○ 1-way

[✓ 2-way]

○ 4-way

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

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

(e) (2.5 pt)

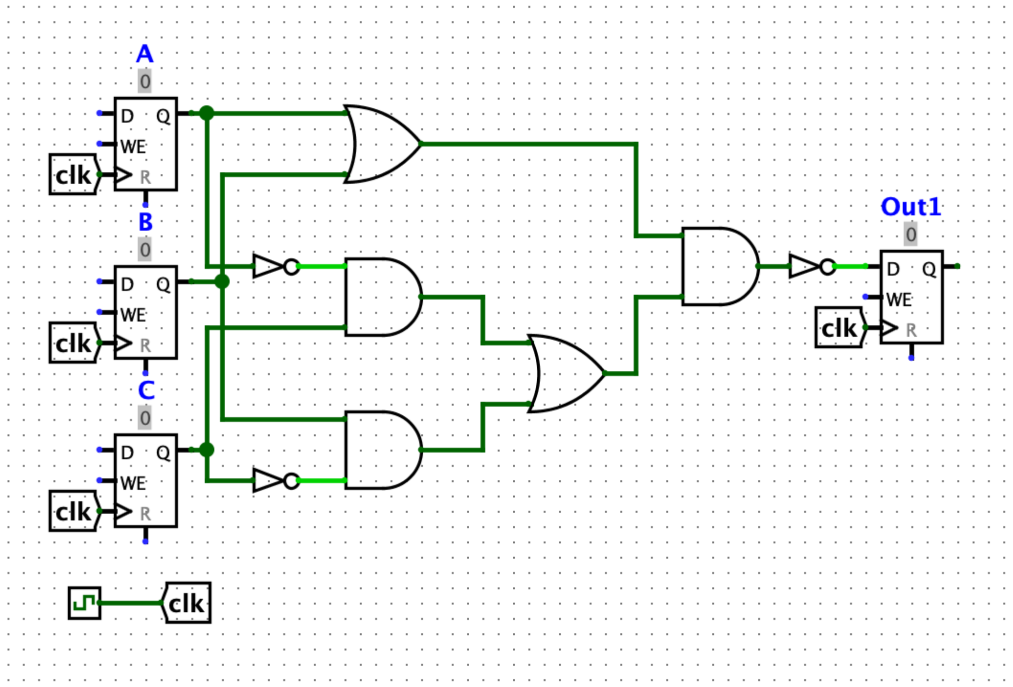
Regardless of our answers above, assume the L1 cache has a hit time of 10 ns, the L2 cache has a hit time of 20 ns, and main memory has a hit time of 100 ns. After a series of accesses, the local hit rate for L1 is $\frac{4}{5}$ and the local hit rate for L2 is $\frac{1}{2}$, what is the AMAT? Please round your decimal values to two places if needed (do not include unnecessary 0's).

24

$10 + (1 - 4/5) * (20 + (1 - 1/2) * 100) = 10 + (1/5) * (20 + 50) = 24ns$

5. Hold on, this Logic isn't that Hard-ware

For this question, do NOT include units of measurement (e.g. ns) in your answer boxes. Consider the following circuit, where A, B, and C are inputs and Out1 is the output:



Circuit

(a) (3.0 pt) Find the simplest (least gates used) Boolean expression that represents the same logic as above.

(Reminder: NOT is represented by '~', And by '*', and OR by '+'. That is also the order of presence. In addition, please use parentheses if needed to organize your logic correctly though do not add any extra ones. Please add spaces between '+' and '*' but do NOT add spaces between '~' and the letter. In each grouping, please organize alphabetically and put groupings before single letters. Try not to add unnecessary parentheses.)

A * C + ~B

$\overline{(A+B)*(B\overline{C}+\overline{A}C)}$

$\overline{A+B}+\overline{B\overline{C}+\overline{A}C}$

$\overline{A}*\overline{B}+(\overline{B}+\overline{\overline{C}})(\overline{\overline{A}}+\overline{C})$

$\overline{A}*\overline{B}+(\overline{B}+C)(A+\overline{C})$

$\overline{A}*\overline{B}+\overline{B}A+\overline{B}*\overline{C}+CA+C\overline{C}$

$\overline{A}*\overline{B}+\overline{B}(A+\overline{C})+AC$

$\overline{B}(\overline{A}+A+\overline{C})+AC$

$\overline{B}(1+\overline{C})+AC$

$A*C+\overline{B}$

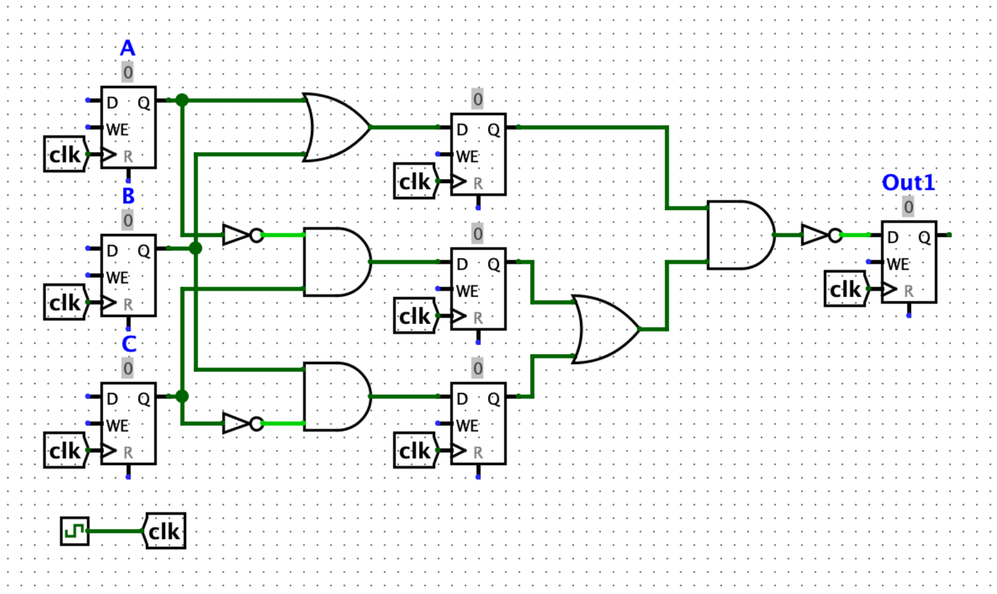
(b) Consider the unsimplified circuit. Say our OR gates take 5 ns each, the AND gates take 2 ns, and the NOT gates take 1 ns. In addition, the registers have setup time 2 ns and clk-2-q 3 ns.

i. A. (2.0 pt) How long is our critical path in ns?

16

It's the path of CLK-2-Q -> NOT -> AND -> OR -> AND -> NOT -> SETUP = 3 + 1 + 2 + 5 + 2 + 1 + 2 = 16 ns

ii. Regardless of our previous answer, we decide, in order to speed our circuit up, to pipeline it by adding registers in the following places.



Pipelined Circuit

A. **(3.0 pt)** What is the maximum clock frequency of the circuit? You should leave your answer as a decimal value in terms of MHz. Please round decimal answers to 2 decimal places. Please keep both decimal places even if they are 0!

76.92

The critical path is now between the middle and end registers. It is CLK-2-Q + OR + AND + NOT + SETUP = 3 + 5 + 2 + 1 + 2 = 13 ns. Therefore, our maximum clock frequency is 1/13 ns ≈ 76.92 MHz.

B. **(4.0 pt)** What is the maximum possible hold time (in ns), so that our circuit will function as expected?

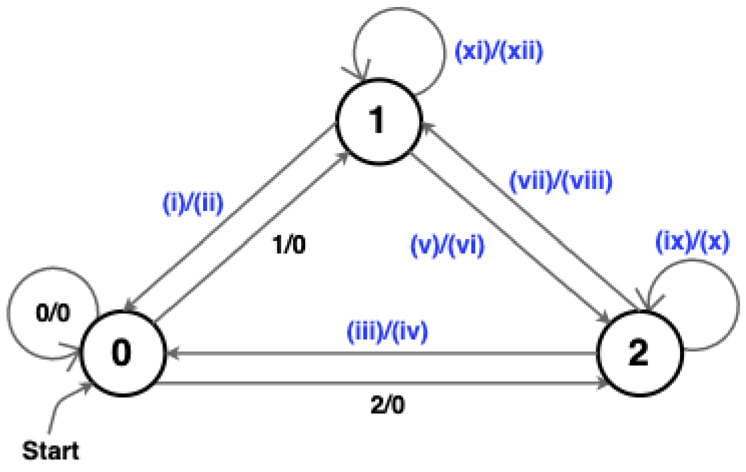
5

To find the maximum hold time, we need to find the shortest path between 2 registers: CLK-2-Q + AND = 3 + 2 = 5 ns.

6. What's Your Wi-Fi Password? Oh, It's 12. It didn't work. No, You Gotta Spell It Out: Oneoneonetwo

Consider a "ternary FSM" that takes in a ternary input. That is, we have inputs and outputs of 0, 1, and 2. Fill in the blanks below so that the FSM outputs 1 if we have seen the sequence "12", 2 if we have seen the sequence "21", and 0 otherwise. Note that our FSM should not reset after seeing either of these sequences, e.g. the sequence "121" should output "012". Please drag the image to another tab so you can refer to it without scrolling up and down.

Fill in the following table of transitions.



FSM

(a) **(1.0 pt)** `(i)` Input from State 1 to State 0

○ 2
● 0
○ 1

(b) **(1.0 pt)** `(ii)` Output from State 1 to State 0

○ 1
○ 2
● 0

(c) **(1.0 pt)** `(iii)` Input from State 2 to State 0

● 0
○ 2
○ 1

(d) **(1.0 pt)** `(iv)` Output from State 2 to State 0

● 0
○ 1
○ 2

(e) (1.0 pt) `(v)` Input from State 1 to State 2

- [✓ 2]
- ○ 0
- ○ 1

(f) (1.0 pt) `(vi)` Output from State 1 to State 2

- ○ 0
- ○ 2
- [✓ 1]

(g) (1.0 pt) `(vii)` Input from State 2 to State 1

- [✓ 1]
- ○ 0
- ○ 2

(h) (1.0 pt) `(viii)` Output from State 2 to State 1

- [✓ 2]
- ○ 1
- ○ 0

(i) (1.0 pt) `(ix)` Input from State 2 to State 2

- [✓ 2]
- ○ 1
- ○ 0

(j) (1.0 pt) `(x)` Output from State 2 to State 2

- ○ 1
- [✓ 0]
- ○ 2

(k) (1.0 pt) `(xi)` Input from State 1 to State 1

- [✓ 1]
- ○ 2
- ○ 0

(l) (1.0 pt) `(xii)` Ouput from State 1 to State 1

- [✓ 0]
- ○ 1
- ○ 2

No more questions.