Nick Weaver CS 61C Sp 2019 Great Ideas in Computer Architecture MT 2

PRINT your name: _____, _____
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct will be reported to the Center for Student Conduct, and may result in partial or complete loss of credit. I am also aware that Nick Weaver takes cheating personally and, like the Hulk®, you don't want to see him angry.

SIGN your name: _____

PRINT your class account login: `cs61c-`_____ and SID: _____

Your Favorite 61C TA's name: _____

Exam # for person sitting to your left: _____

Exam # for person sitting to your right: _____

You may consult one sheet of paper (double-sided) of notes. You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted.

You have 110 minutes. There are 7 questions, of varying credit (90 points total). The questions are of varying difficulty, so avoid spending too long on any one question. Parts of the exam will be graded automatically by scanning the **bubbles you fill in**, so please do your best to fill them in completely. For short answer question use only the space provided and your answers must be short. You do not need to read any footnotes.

If you have a question, raise your hand, and when an instructor motions to you, come to them to ask the question.

DO NOT SEPARATE THE PAGES OF THE EXAM

Do not turn this page until your instructor tells you to do so.

Question:	1	2	3	4	5	6	7	Total
Points:	10	10	10	12	18	12	18	90

Problem 1 ***Boolean Expressions*** **(10 points)**

Look at the Boolean Expression Table. We'd like to implement its functionality as a circuit!

A	B	C	D	OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Table 1: Boolean Expression Table

(a) Write a boolean expression that represents this truth table as a sum of products. You do not need to simplify your answer.

Solution: $\bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}C\bar{D} + A\bar{B}CD$

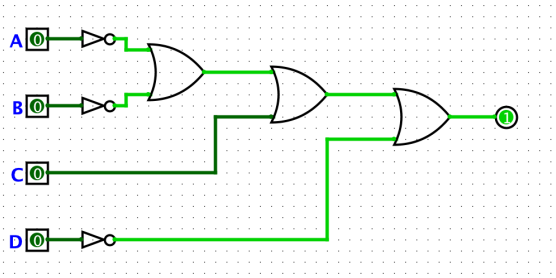
(b) Simplify the boolean expression from the previous part such that your expression uses the minimum number of gates. Show your work clearly.

Solution: $C(A \oplus B)$

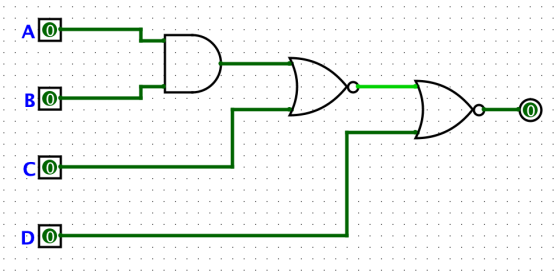
(c) Now we'll work with a new expression:

$$\overline{\overline{\overline{(\overline{AB}+C)}D}}$$

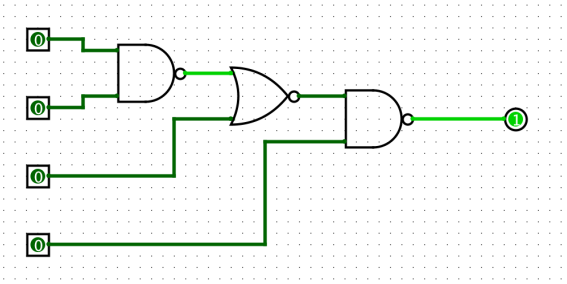
Select the correctly simplified circuit.



(a) ○



(b) ○



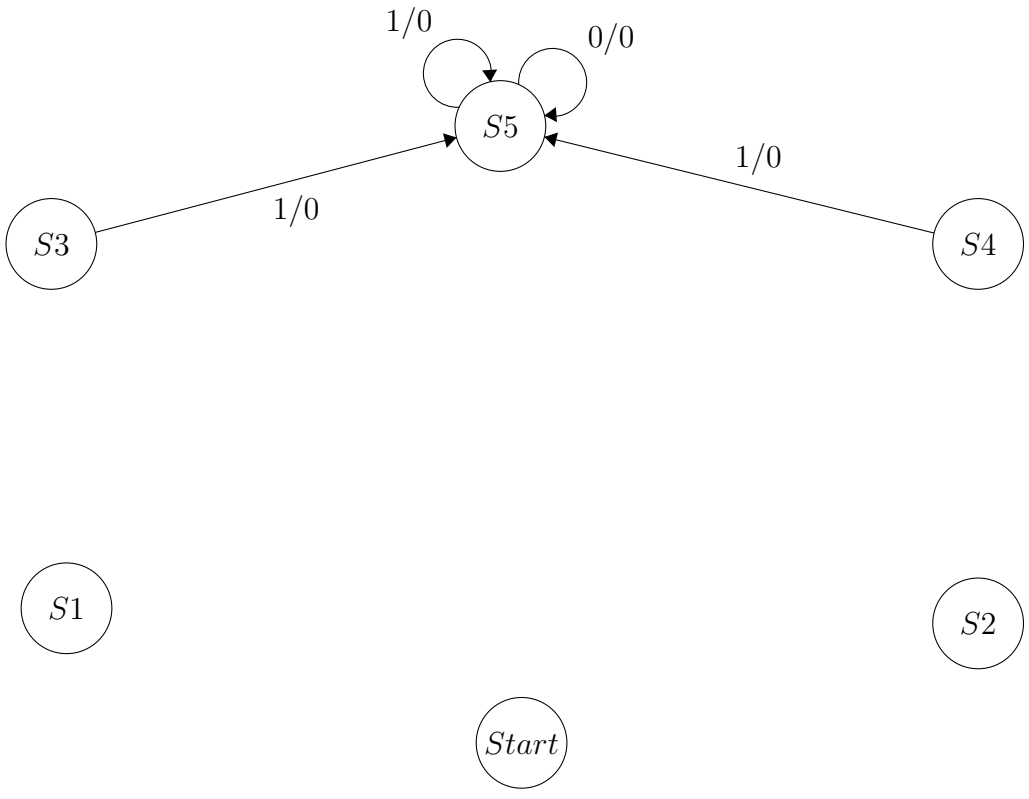
(c) ●

Problem 2 ***FSMs*** **(10 points)**

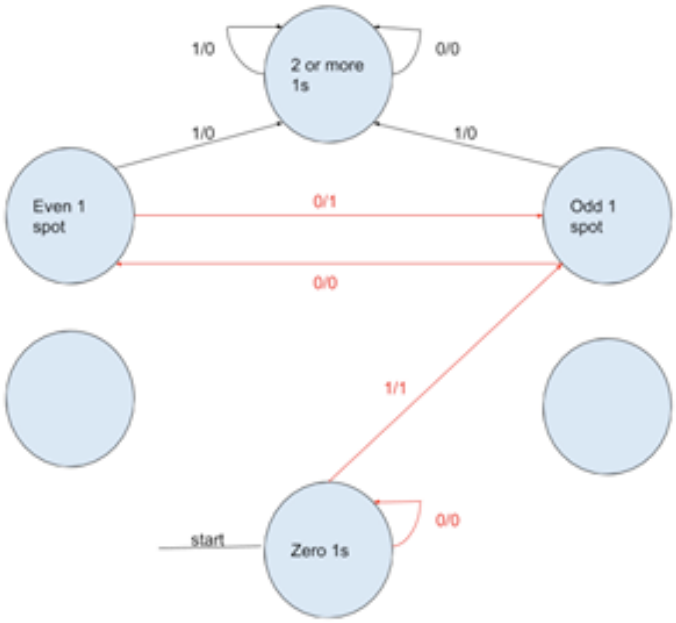
Complete the states and transitions on the FSM below. For full credit, use the minimum number of states. Our FSM should output 1 if the bit stream is currently a power of 4 and 0 otherwise. Note that 0 is not a power of 4. Label all transitions with *input/output* notation used in discussion section. The first set of transitions has been done for you. Assume we append bits to the least-significant end of the bit stream and our number is interpreted as an unsigned integer.

The following table shows the inputs/outputs at each step as we try to parse the bit-string 001000

Input Stream	Output Stream
0	0
00	00
001	001
0010	0010
00100	00101
001000	001010



Solution:


1/0
2 or more 1s
0/0
1/0
1/0
Even 1 spot
0/1
Odd 1 spot
0/0
1/1
start
Zero 1s
0/0

Problem 3 ***Single Cycle Datapath*** **(10 points)**

In an (incorrect) implementation of the RISC-V Datapath, the ASel and BSel signal are sent to the opposite MUXes. ASel to Mux B and BSel to Mux A.

(a) Which instruction types still work all of the time?

- [x] R
- [x] SB
- [x] U
- [x] UJ
- [ ] I

(b) Which of the following lines of code will execute incorrectly on the wonky datapath?

- [x] `addi t0 t0 2`
- [ ] `beq t1 s0 LABEL`
- [ ] `jalr ra 0(a0)`

Problem 4 ***Floating Point*** **(12 points)**

Consider a modified floating point scheme where we opt to use 7 bits for the exponent and 24 bits for the significand but is otherwise the same as IEEE 754 Single Precision Floating Point (what you learned in lecture).

(a) What is the new bias for our Floating Point Scheme? Write your answer in decimal.

Bias: _____

Solution: Bias = $2^{7-1} - 1 = 63$

(b) Represent 18.75 in our new Floating Point Scheme. You may leave your answer in hex or binary.

Sign: _____

Exponent: _____

Significand: _____

Solution: Sign = 0
18.75 = 10010.11 = 1.001011 * 2^4
Exp - Bias = 4 —¿ Exp = 63 + 4 = 67 = 0b1000011
Significand = 0010110...0

Number = 0x432C0000

(c) What is the largest finite value you can represent?

Sign: _____

Exponent: _____

Significand: _____

Solution: Sign: 0
Exp = 126
Signficand = all 1s = 2 - 2^{-24}

$2^{126-63} * (2 - 2^{-24}) = 2^{64} - 2^{39}$

Select the best answer to the following multiple choice questions. No explanation is required.

(d) We can represent _____ distinct numbers in our new floating point scheme than we can in single precision floating point.

- ○ More
- ○ The same amount
- ● Fewer

Solution: Fewer. We still have the same number of bits which means we have $2^3 2$ possible distinct values. However all NaNs are the same, of which there are $2^{significantbits} - 2$. Since we increased our significand bits by one there are now 2^{23} more representation of NaN.

(e) We can represent _____ positive numbers less than 2 in our new floating point scheme than we can in single precision floating point.

- ○ More
- ● The same amount
- ○ Fewer

Solution: The same. We are looking at all values up to the bias for each scheme.

Single Precision 0, 127 —¿ 7 bits
Our scheme 0, 63 —-¿ 6 bits

For each value of the exponent we can represent $2^{significand}$ different values. Putting this all together, we see:

Single precision: $2^{23} numbersperexp * 2^7 exp - 1(zeroisnotpositive) = 2^{30} - 1$
Our scheme: $2^{24} numbersperexp * 2^6 exp - 1(zeroisnotpositive) = 2^{30} - 1$
So the two answers are the same.

Problem 5 ***Short Cache/AMAT Questions*** **(18 points)**

(a) Consider a 2-way set associative cache with four word blocks, a cache size of 2 KiB, and a 256 TiB physical address space. What is the T:I:O breakdown for the cache? Write your answer on the blanks provided.

Tag: _____

Index: _____

Offset: _____

Solution: Tag: 38 bits, Index: 6 bits, Offset: 4 bits

(b) Mark each of the following as True or False and justify your answer in a tweet-length response. Answers without justification (or with incorrect justification) will not receive full credit.

1. Increasing a cache's block size always decreases the AMAT.

○ True ● False

Solution: False: There reaches a point where increasing the block size starts decreasing the hit rate. Furthermore, the larger the block size, the longer it takes to fetch the block from memory.

2. Fully associative caches have a higher hit time than direct mapped caches of the same size.

● True ○ False

Solution: True: The processor has to iterate through (maximally) all entries to find the correct entry, which takes more time than lookup in a direct-mapped cache where we can just check if the mapped entry is a hit.

3. Doubling the size of our cache but keeping the block size and associativity the same will always produce at least the same hit rate.

● True ○ False

Solution: We cannot introduce any new misses. Why? Compulsory misses are unaffected cause our block size is unchanged. To have a new capacity/-conflict miss we would need to have a new collision in the cache (blocks map to the same set that didnt before) and depending on the number of accesses this would be either conflict or capacity. However since we doubled the number of sets we are adding a higher order bit. So things either already conflicted (unchanged) or no longer conflict for the same associativity. So we cannot have more misses.

4. If the majority of misses in our cache are conflict misses then switching to a fully associative cache will always produce at least the same hit rate.

○ True ● False

Solution: While we will remove all conflict misses this does not say anything about reducing the total misses. We could still convert previous hits into misses, which if there is a greater number of those than were previously conflict misses could produce a worse hit rate.

Consider the following possibly complex access pattern for a cache with just 4 blocks, comparing direct mapped to fully associative

Direct Mapped:

Repeat Forever

Tag: 1, Index: 0
Tag: 0, Index: 0
Tag: 1, Index: 0
Tag: 0, Index: 1
Tag: 0, Index: 2
Tag: 0, Index: 0
Tag: 1, Index: 0
Tag: 0, Index: 3
Tag: 0, Index: 1
Tag: 0, Index: 2
Tag: 0, Index: 3

Ignoring compulsory misses we find that the misses are access 2 (capacity), access 3 (conflict), access 6 (conflict), and access 7 (conflict).

However looking at the fully associative case we have

Fully Associative:

Repeat Forever

Tag: 4
Tag: 0
Tag: 4
Tag: 1
Tag: 2
Tag: 0
Tag: 4
Tag: 3
Tag: 1
Tag: 2
Tag: 3

Again Ignoring compulsory misses we find that the misses are access 2 (capacity), access 4 (capacity), access 5 (capacity), access 8 (capacity), access 9 (capacity), access 10 (capacity)

TLDR even though all conflict misses will become hits we can also introduce new misses.

(c) Assume that the local hit rate for our L1 cache is 40% and the local hit rate for our L2 cache is also 40%. The hit time for the L1 cache is 5 cycles, the hit time for the L2 cache is 25 cycles, and access time for main memory is 100 cycles. Find the average memory access time. Express your answer in terms of cycles.

AMAT = _____ Cycles

Solution: $AMAT = (L1_HT + (L1_MR * (L2_HT + (L2_MR * MEM))))$
$= 5 + (.6 * (25 + (.6 * 100)))$
$= 5 + (.6 * (85))$
$= 5 + 51 = 56 cycles$

Problem 6 ***Maybe cache(s) can buy happiness...*** **(12 points)**

You have secured a fancy summer internship analyzing cache performance at a data processing company Glamazon, and you don't want to F@#)(* it up. In your first week, you discover the average user has a hit rate of 20% and an 80% miss rate. Furthermore, you find the misses follow (roughly) the following distribution:

Compulsory: 20%
Capacity: 6%
Conflict: 74%

Your users' main system has 256 B direct mapped cache with 16 B blocks. Glamazon is also installing a fancy new machine with an 1024 B 2-way set associative cache with 4B blocks. Both machines feature a 32-bit architecture.

(a) After migrating all your users to the new machine, you get an angry call from a customer complaining their results arent rolling in as fast as they used to. Which of the following miss-classification profiles likely belongs to this customer?

- ● Compulsory: 80%
Capacity: 16%
Conflict: 4%
- ○ Compulsory: 3%
Capacity: 79%
Conflict: 18%
- ○ Compulsory: 13%
Capacity: 12%
Conflict: 75%
- ○ Compulsory: 80%
Capacity: 15%
Conflict: 15%

You open up the customers main processing program and see the following.

```
void genFakeReviews(char* products[], int countP, char* reviews[],
                    int countR) {
    for (int i = 0; i < countP; i++) {
        for (int j = 0; j < countR; j++) {
            postReview(products[i], reviews[j]);
        }
    }
}
```

You decide to test the function with the following parameters:

```
genFakeReviews(products, 20, fakeReviews, 20);
```

You may assume the arrays are block aligned and do not overlap or contain overlapping elements. When loaded into memory, `products` lives at 0x04000000 and `fakeReviews` lives at 0x08000000. Assume function call operands are always evaluated left to right.

(b) You simulate the code on the old cache (256 B direct mapped cache with 16 B blocks). What is the hit rate?

Hit Rate: _____

Solution: 627/800

(c) You simulate the code on the new cache (1024 B 2-way set associative cache with 4B blocks). What is the hit rate?

Hit Rate: _____

Solution: 19/20 == 760/800

(d) You find your simulation results confusing, so you run the code again on each simulation. In your haste you forget to clear the caches, so the data from the previous runs sticks around.

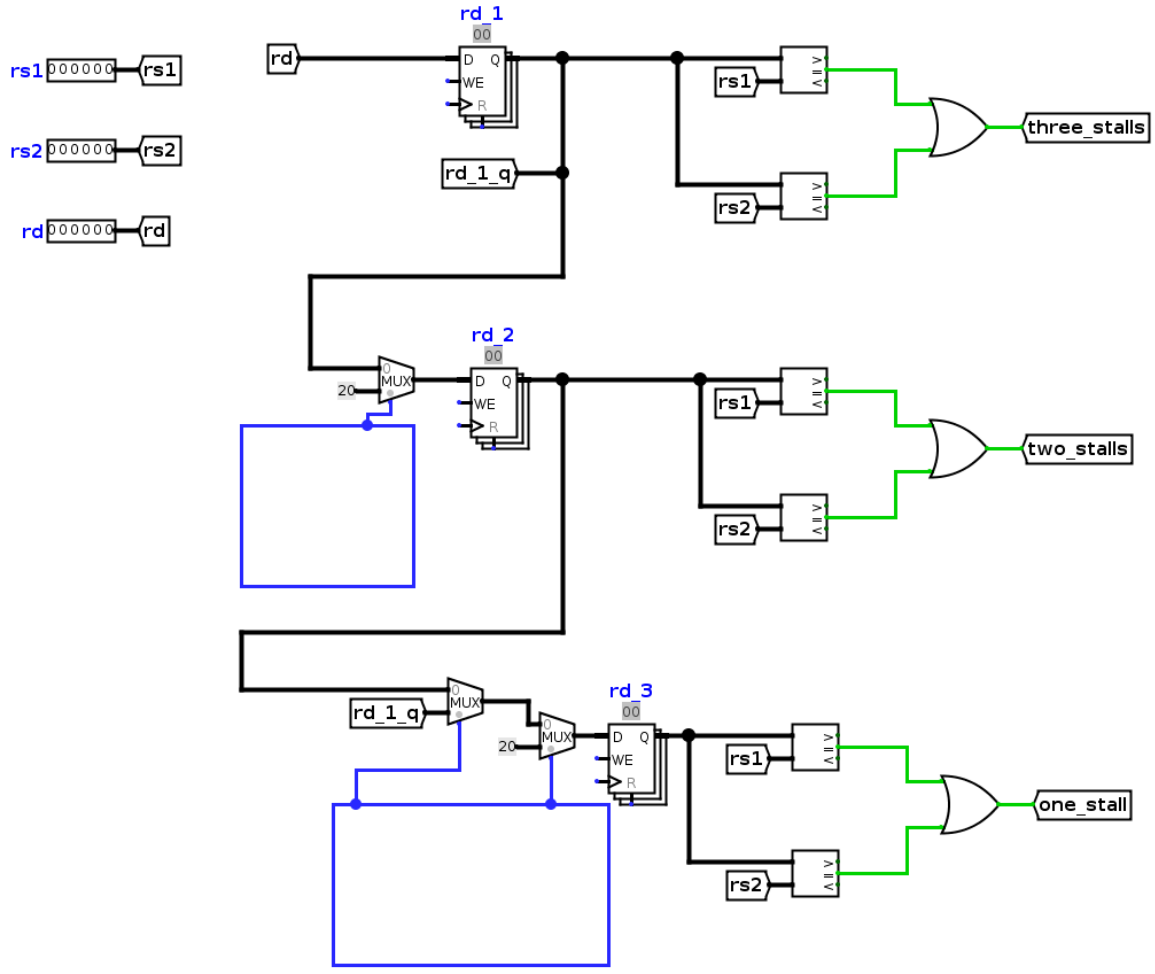
How do the new, second round hit rates compare?

- ○ They are equal
- ○ The old caches second run is better than the new caches second run
- ● The new caches second run is better than the old caches second run

Problem 7 ***Stalling*** **(18 points)**

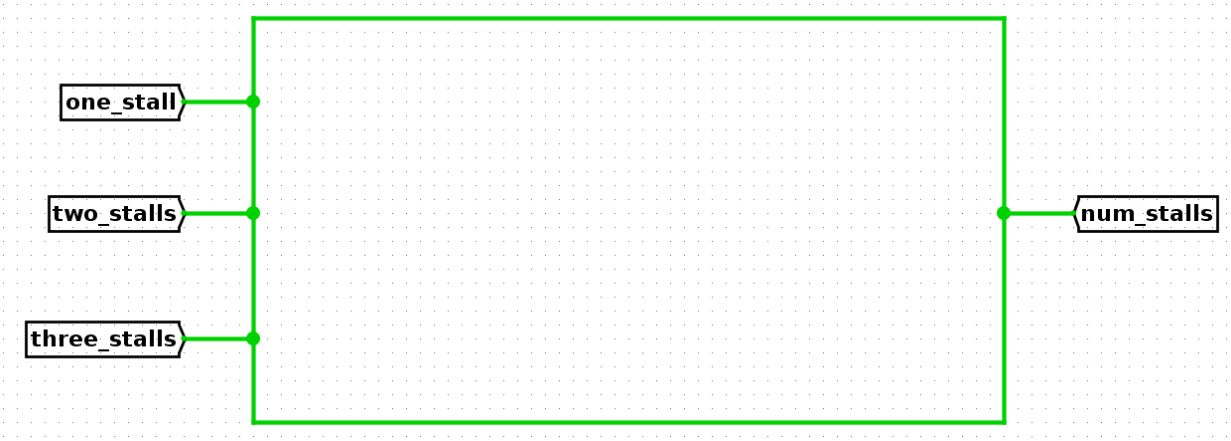
The missing part of a basic five-stage RISC-V pipeline before we added forwarding that we dont cover in lecture is the hardware logic that decides when to insert stalls if we have a data hazard. Lets put it together now! Heres an outline of a hardware unit that will decide if we need a stall due to a data hazard in that youll fill in the missing pieces for. Each clock cycle, it takes in the 5-bit signals for rd, rs1, and rs2 that come into the Instruction Decode phase and it outputs the number of stalls that are necessary. For this question we will assume all of our instructions are R-Type Instructions, every instruction will write to rd and use rs1 and rs2. Additionally we assume that you cannot read and write in the same clock cycle, which means you could need to stall as many as 3 clock cycles.

In the diagram below you will see that we store the value of rd for each of the past 3 instructions, which are from top to bottom, $rd[i-1]$, $rd[i-2]$, and $rd[i-3]$ respectively.

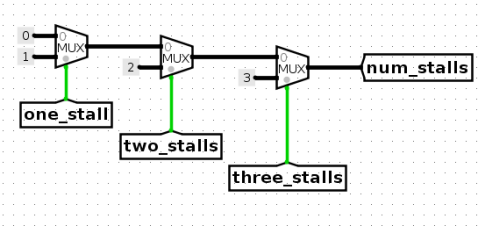


(a) Pick maximum stalls

Fill in the logic below that takes in three one-bit signals (one_stall, two_stalls, and three_stalls) and output a constant representing the maximum number of stalls (i.e. 3 if three_stalls is 1, 2 if two_stalls is 1, 1 if one_stall is 1, and 0 otherwise). If multiple signals are 1, for example, if one_stall and three_stalls are both 1, the circuit should output a 3. You may use any combination of constants and the components listed on the Logisim Cheat Sheet on the back this exam, but nothing else.

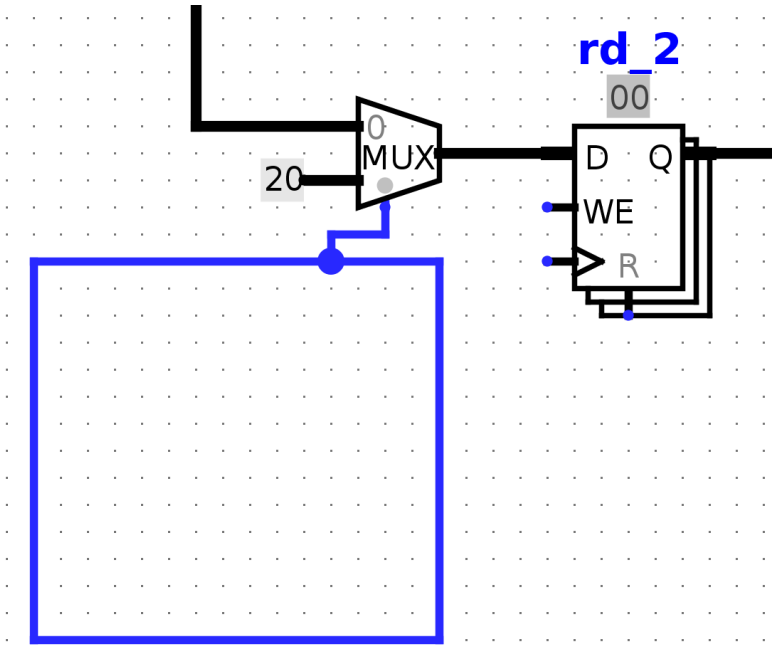


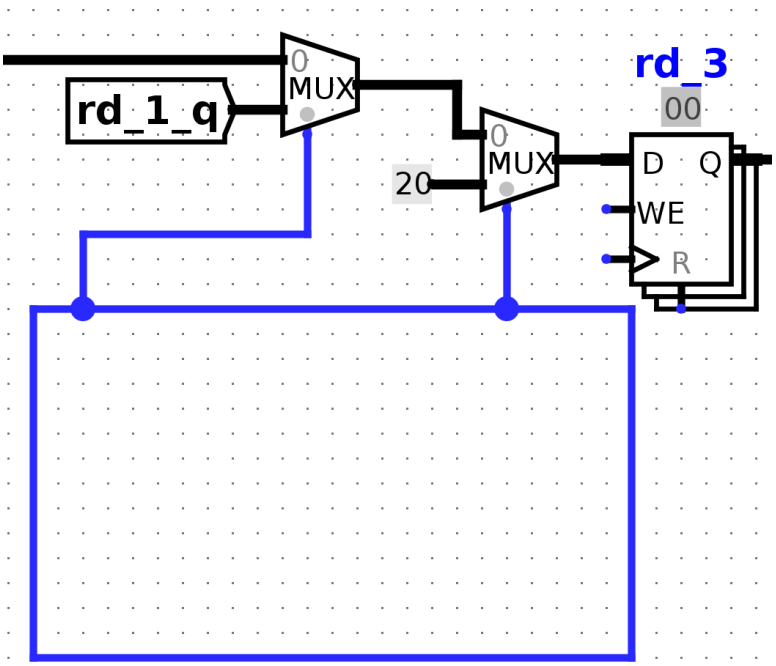
Solution:



(b) Flush

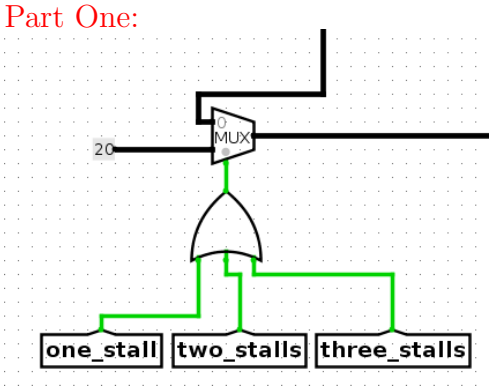
When we stall, we want to flush whatever is in the pipeline so that we dont have any unnecessary stalls in the future. Fill in the diagram below with the appropriate stalling logic for $rd[i-2]$, and $rd[i-3]$ so that we can output the correct number of stalls.



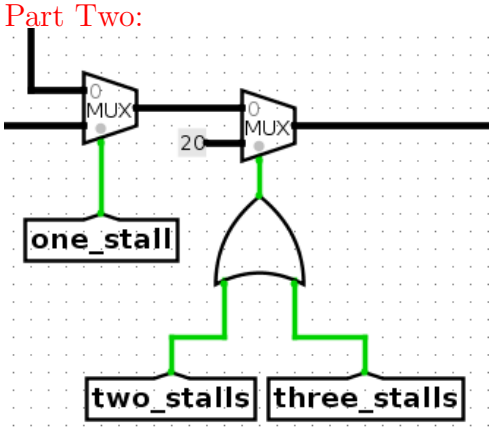


Solution:

Part One:

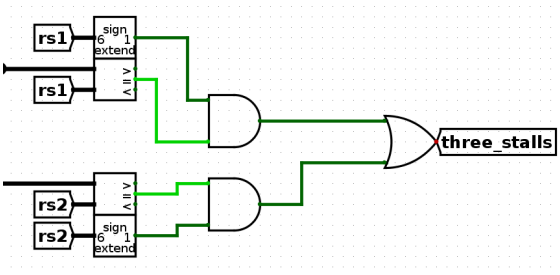


Part Two:

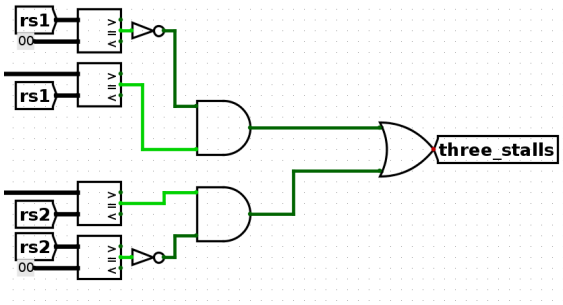


(c) x0 Optimization

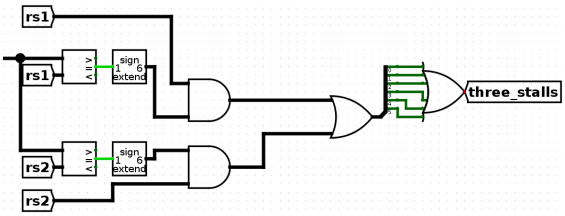
You'll notice that the logic we provided stalls even if the register we write back to is x0. This is unnecessary, and we can optimize it out. Select all of the following modifications that produce three stalls when necessary and not when $rd[i-1]$ is x0.



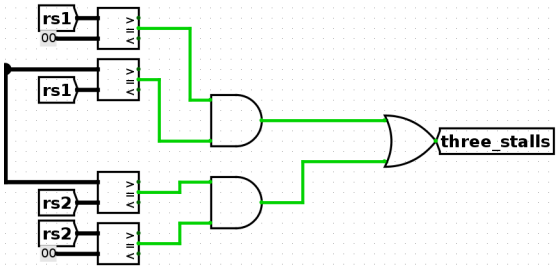
(a) ○



(b) ●



(c) ●



(d) ○

Figure 3: Good Luck
And Don't F*@)! It Up