University of California, Berkeley – College of Engineering
Department of Electrical Engineering and Computer Sciences
Summer 2019 Instructors: Branden Ghena, Morgan Rae Reschenberg, Nicholas Riasanovsky 2019-07-29

CS61C MIDTERM 2

***Last** Name (Please print clearly)*				
***First** Name (Please print clearly)*				
Student ID Number				
Circle the name of your Lab TA	**Ayush Maganahalli**	**Chenyu Shi**	**Gregory Jerian**	**Jenny Song**
	John Yang	**Lu Yang**	**Ryan Searcy**	**Ryan Thornton**
Name of the person to your: Left | Right				
All my work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who haven't taken it yet. ***(please sign)***				

Instructions

- This booklet contains 22 pages including this cover page. **The back of each page of this exam is blank and can be used for scratch work, but will not be graded**.
- Please turn off all cell phones, smartwatches, and other mobile devices. Remove all hats and headphones. Place *everything* except your writing utensil(s), cheat sheet, and beverage underneath your seat.
- You have 80 minutes to complete this exam. The exam is closed book: no computers, tablets, cell phones, wearable devices, calculators, or cheating. You are allowed two pages (US Letter, double-sided) of *handwritten* notes.
- There may be partial credit for incomplete answers; write as much of the solution as you can.
- Please write your answers within the boxes and blanks provided within each problem!

Question	**1**	**2**	**3**	**4**	**5**	**Total**
Possible Points	10	16	14	29	21	90

If you have the time, **feel free to doodle on the front page!**

Question 1: Floating *Points to your Cheat Sheets - 10 pts

For all of the following questions we are using the IEEE 754 single precision floating point from lecture. If you do not remember the details, some can be found on the back side of the green sheet.

1. Represent **14.75** in its floating point representation. Put your answer in hexadecimal.

sign = 0
14.75 = 1110.11 = $1.11011 * 2^3$
significand = 1101100….0
exp - bias = 3 -> exp - 127 = 3 - > exp = 130 = 0b10000010

0x416C0000

2. Represent $\mathbf{-2^{-147}}$ in its floating point representation. Put your answer in hexadecimal.

sign = 1
$2^{-147} < 1 * 2^{1 - 127}$, so we are working with a denormalized number
exp = 0
denorm fomula = $-1^{sign} * 0.significand * 2^{-126}$
$2^{-147} = 2^{-21} * 2^{-126}$
significand = 2^{-21} = 0b000..0100

0x80000004

3. What value is represented by 0xFF800001?

NaN

For the remaining questions we are going to consider 2 possible changes:

- **Option 1**: Adding a bit to signficand and removing a bit from the exponent
- **Option 2**: Adding a bit to the exponent and removing a bit from the significand

For each of the following questions select whether **option 1, option 2, neither,** or **both** will accomplish the presented task. Assume that the bias also shifts to be $\mathbf{2^{exp\_bits - 1} - 1}$.

4. Represent pi more accurately than our IEEE 754 single precision floating point.

Ⓐ Option 1 Ⓑ Option 2 Ⓒ Neither Ⓓ Both

More significand bits leads to more digits of pi.

5. Represent smaller positive numbers than IEEE 754 single precision floating point.

Ⓐ Option 1 Ⓑ Option 2 Ⓒ Neither Ⓓ Both

smallest denorm number is always $(2^{-num\_significand\_bits}) * 2^{- bias + 1}$

Because the bias of option 2 is roughly twice as big as IEEE 754 our smallest number is much smaller.

6. Represent more numbers in the range [1, 2) than IEEE 754 single precision floating point.

Ⓐ Option 1 Ⓑ Option 2 Ⓒ Neither Ⓓ Both

There are $2^{num\_significand\_bits}$ numbers in the range [1, 2). As a result option 1 has the most numbers.

7. Represent more numbers than IEEE 754 single precision floating point.

Ⓐ Option 1 Ⓑ Option 2 Ⓒ Neither Ⓓ Both

This one is tricky. All values represented are numbers except for NaN. With 32 bits we represent 2^{32} values, so now we just need to remove the NaN. For every scheme this is the number of values at the largest exponent minus positive and negative infinity or:

$2^{num\_significand\_bits} - 2$

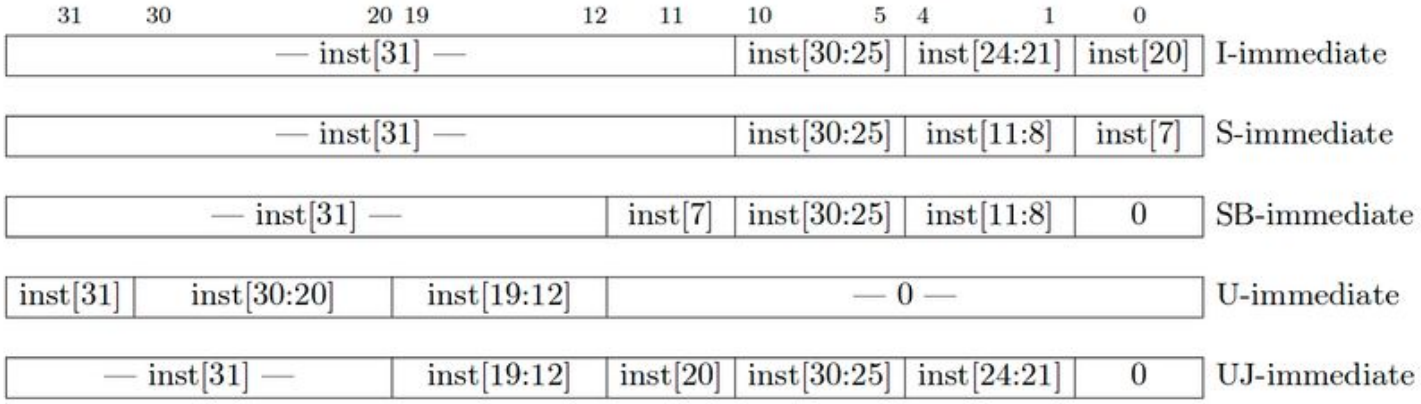
So we can represent $2^{32} - 2^{num\_significand\_bits} + 2$ numbers in any floating point scheme. Since Option 2 has the fewest exponent bits it represents the most numbers.

Question 2: ReCALL This Information (or have it written down I guess) - 16 pts

Consider the following assembly code in a file foo.s:

```
      .text
1.              mv s1 a0
2.              addi s2 s2 4
3.    Start:    beq s1 x0 End
4.              lw a0 0(s1)
5.              jal ra printf
6.              add s1 s2 s1
7.              lw s1 0(s1)
8.              jal x0 Start
9.    End:      jalr x0, ra, 0
```

Recall that immediate values are generated from instructions with the following table:



We will refer to the number produced after this process is completed as the “immediate value.”

1. Fill in all fields (or write Does Not Apply) for the machine code generated for **beq s1 x0 End** (line 3).

Immediate value: 24 funct3: 0x0

opcode: 0x63 funct7: N/A

rs1: 9 rs2: 0 rd: N/A

Given the hex representation, which line number in the above program does it correspond to?

2. 0x0004A483

Line: 7

3. 0xFEDFF06F

Line: 8

4. After generating the object file (foo.o) of the previous code (foo.s), this object file is run through a linker with static library lib.a. Assuming any labels not found in the object file are found in lib.a, which of the following will be used to resolve the instruction `jal ra printf`?

[ ] foo.o's symbol table
[X] foo.o's relocation table
[X] lib.a's symbol table
[ ] lib.a's relocation table
[ ] None of the Above

For each of the following questions select which stage of CALL (Compiler, Assembler, Linker, Loader) the action occurs in:

5. Command line arguments are placed into memory

Ⓐ Compiler Ⓑ Assembler Ⓒ Linker Ⓓ Loader

6. Static data is placed in memory

Ⓐ Compiler Ⓑ Assembler Ⓒ Linker Ⓓ Loader

7. External labels are resolved

Ⓐ Compiler Ⓑ Assembler Ⓒ Linker Ⓓ Loader

8. Operator precedence is resolved

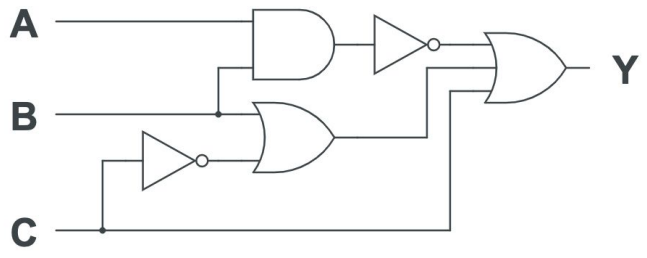
Ⓐ Compiler Ⓑ Assembler Ⓒ Linker Ⓓ Loader

Question 3: Are Vulcans good at digital logic? - 14 pts

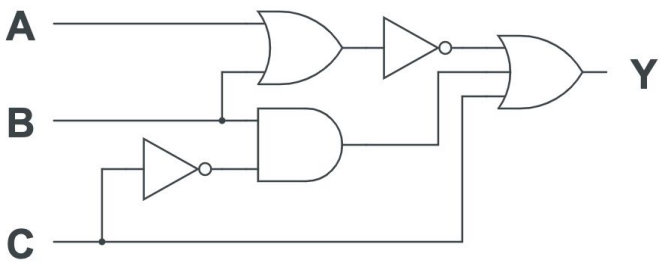
Which circuit diagram exactly matches the following boolean algebra expression?

$$Y = C(\overline{A + B})(B\overline{C})$$

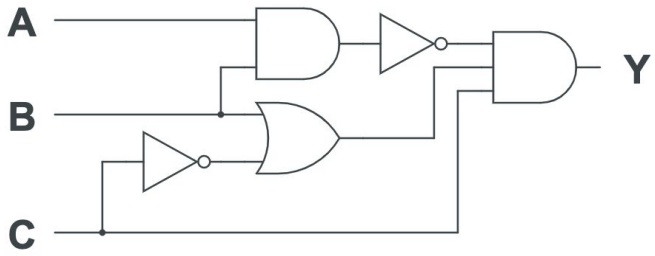
1)



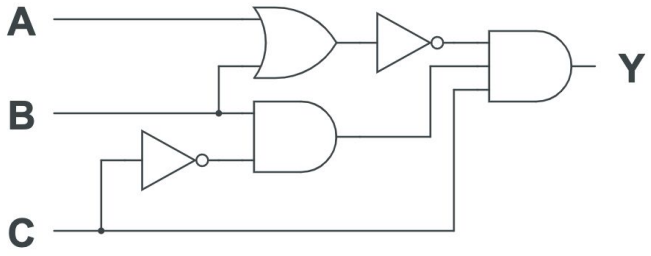
2)



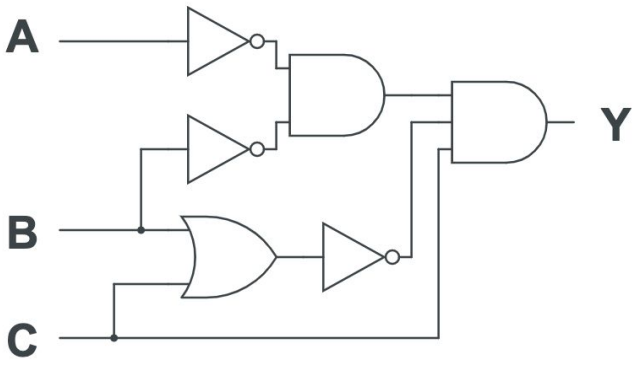
3)



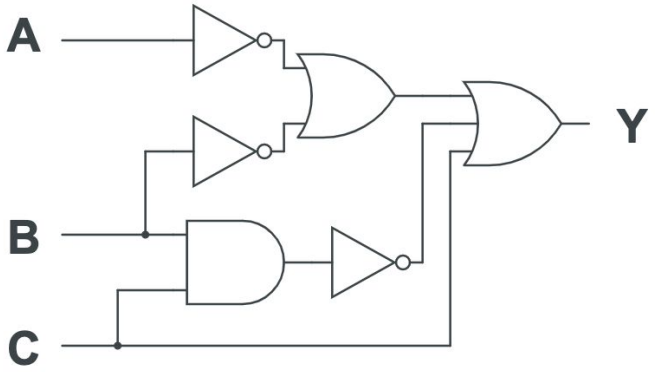
4)



5)



6)



The correct circuit is number: 4

Simplify the following boolean algebra expression. Show your work for partial credit, and you may use any method to simplify.

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

$$Y = B(AB + A\neg B)(\neg(AC) + C)$$

$$Y = (ABB + A\neg BB)(\neg A + \neg C + C)$$

Because $XX = X$, $\neg XX = 0$, and $X + \neg X = 1$

$$Y = (AB + 0)(\neg A + 1)$$

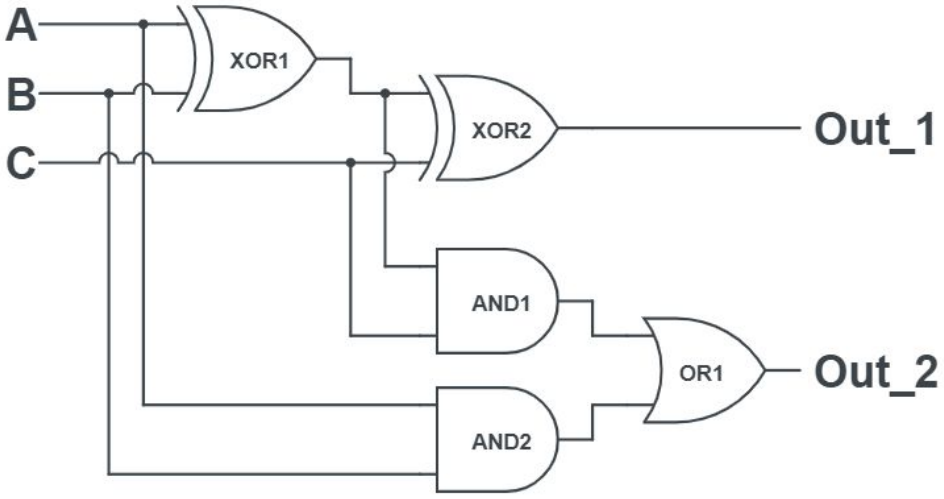
Because $X + 0 = X$, $\neg X + 1 = 1$

$$Y = AB(1)$$

Y=AB

Simplified Solution: AB

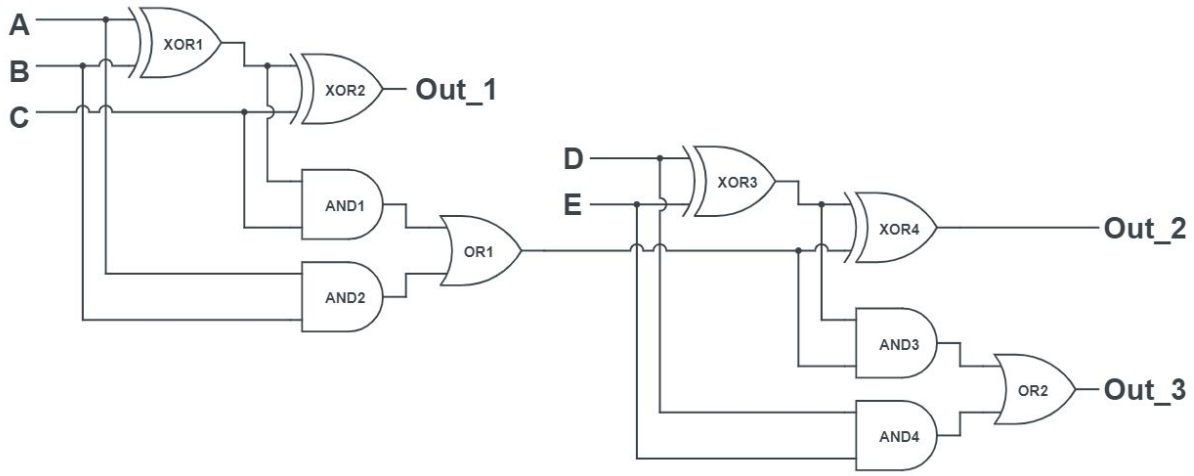
Fill out the following truth table that corresponds to the following circuit.



C	B	A	Out_1	Out_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Find the combination logic delays for each output or each circuit given the following parameters. There is no setup or hold time from the inputs or outputs.

- XOR gate delay: 80 ps
- AND gate delay: 60 ps
- OR gate delay: 40 ps



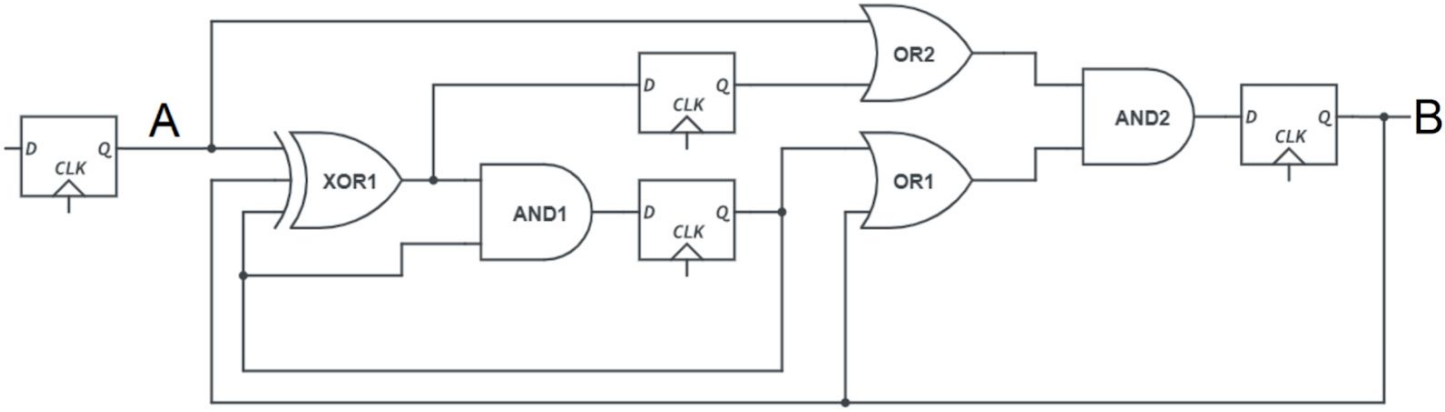
Out_1 Delay: 80ps + 80ps = 160ps

Out_2 Delay: 80ps + 60ps + 40ps + 80ps = 260ps

Out_3 Delay: 80ps + 60ps + 40ps + 60ps + 40ps = 280ps

For the next problems, consider the following pipelined circuit. Assume all registers have their clock inputs correctly connected to a global clock signal and that logic gates have the following parameters:

- XOR gate delay: 80 ps
- AND gate delay: 60 ps
- OR gate delay: 40 ps



When shopping for registers, we find two different models and want to determine which would be best for our circuit.

Register Type λ

- Setup Time: 40 ps
- Hold Time: 20 ps
- Clock-to-Q Delay: 30 ps

Register Type τ

- Setup Time: 10 ps
- Hold Time: 10 ps
- Clock-to-Q Delay: 80 ps

Critical Path = CLK_Q + XOR + AND + SETUP
Because this passes through 2 registers, our latency is 2 clock cycles.

Note after release we found 2 other interpretations to this question. 1 has just 1 critical path because it considers the latency to be just the top path A takes to B. The second also counts an extra clock to q to give A its value or propagate through the last register to B.

What is the minimum latency for the circuit from A to B if we use register type **λ**? 2 * (30ps + 80ps + 60ps + 40ps) = 420ps

What is the minimum latency for the circuit from A to B if we use register type **τ**? 2 * (80ps + 80ps + 60ps + 10ps) = 460ps

Question 4: I'm afraid of datapaths, so iarrn away - 29 pts

Morgan notices much of the assembly code she writes involves iterating through arrays of integers. Instead of using several instructions to calculate the address of the next element, she proposes a new instruction,

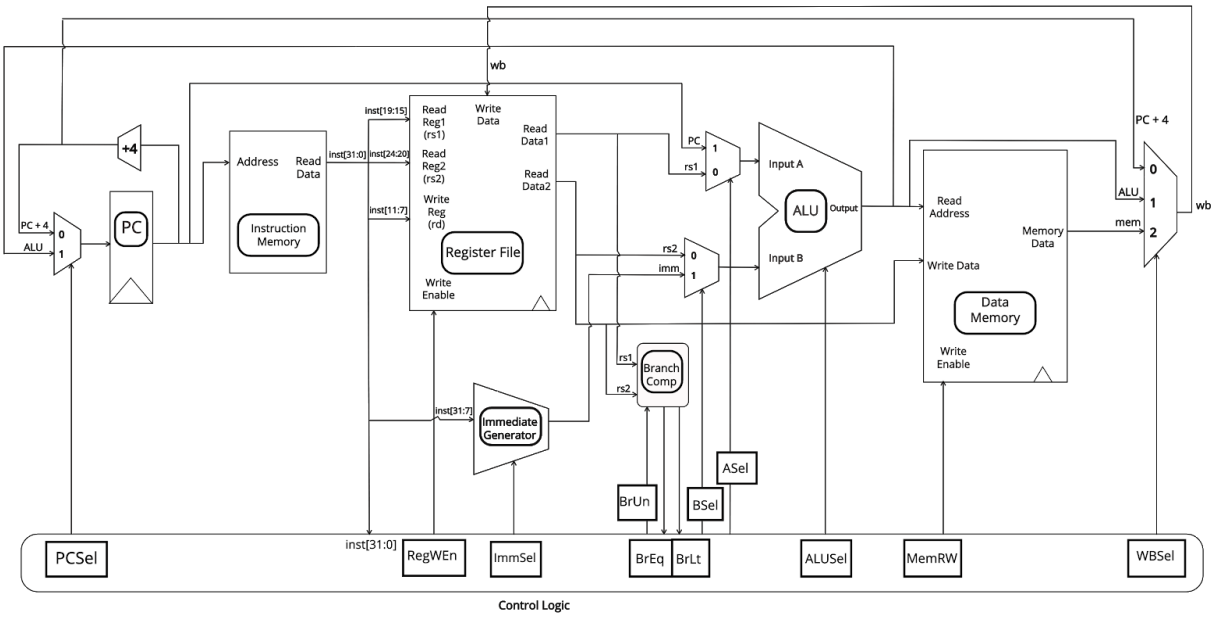
```
iarrn rd rs1 rs2
```

which places into rd the address of the rs2-th element of the array pointed to by rs1. This instruction does not do bounds checking and it assumes the size of an integer is 4B (32 bits). Do *not* assume this instruction belongs to a specific type.

In verilog, the instruction is described as follows:

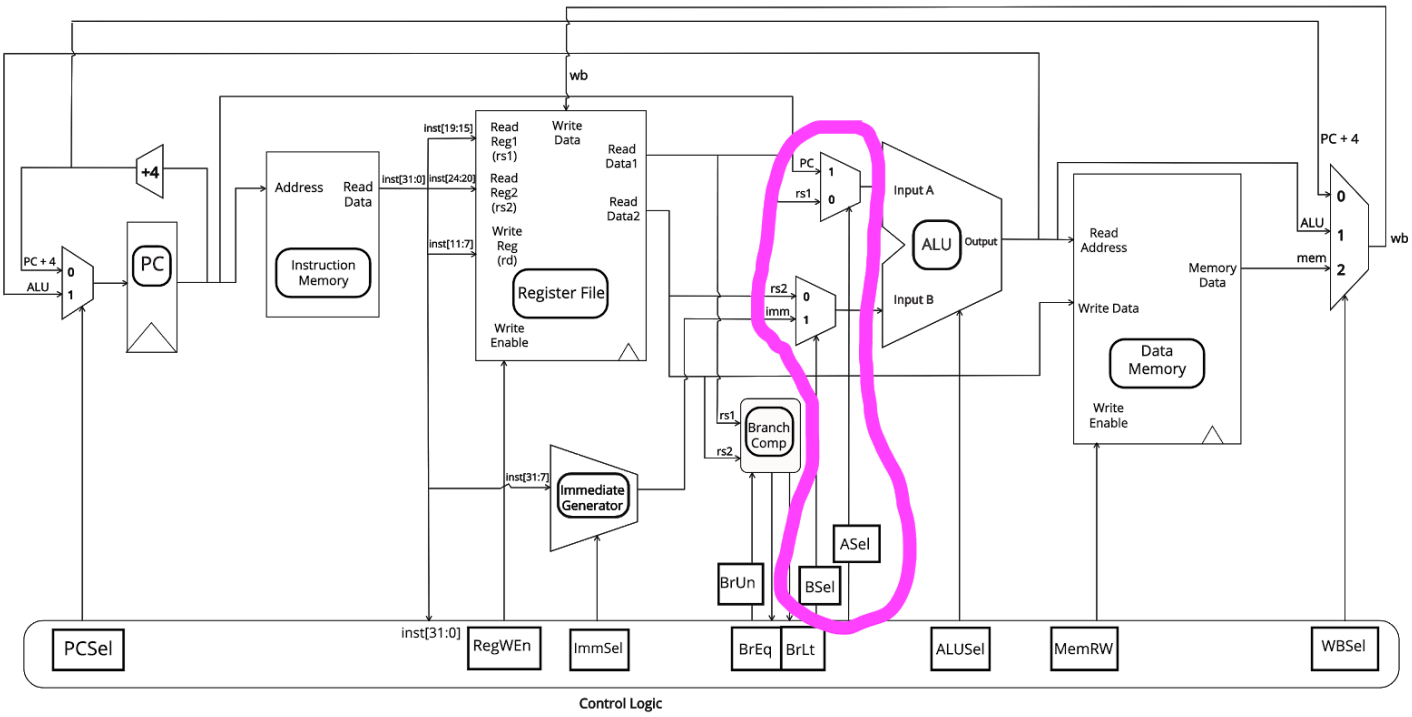
```
R[rd] = R[rs1] + (4 * R[rs2])
```

Morgan is interested in modifying our RISC-V datapath to support this instruction. Assume we have introduced a new control bit "IArrN" which is 1 when the current instruction is `iarrn` and 0 otherwise. Using the datapath below, fill in the following table with the rest of the control bits for this instruction. If the control bit can be set to "*", please draw an X in the table below.



IArrN	PCSel	RegWEn	MemRW	WBSel	BrUn	ALUSel
1	0	1	0	1	X	ADD

Morgan notices this instruction involves changing a few hardware pieces on the datapath in addition to changing control bits above. She proposes modifying the ASel and BSel muxes, and their associated control bits (circled below).

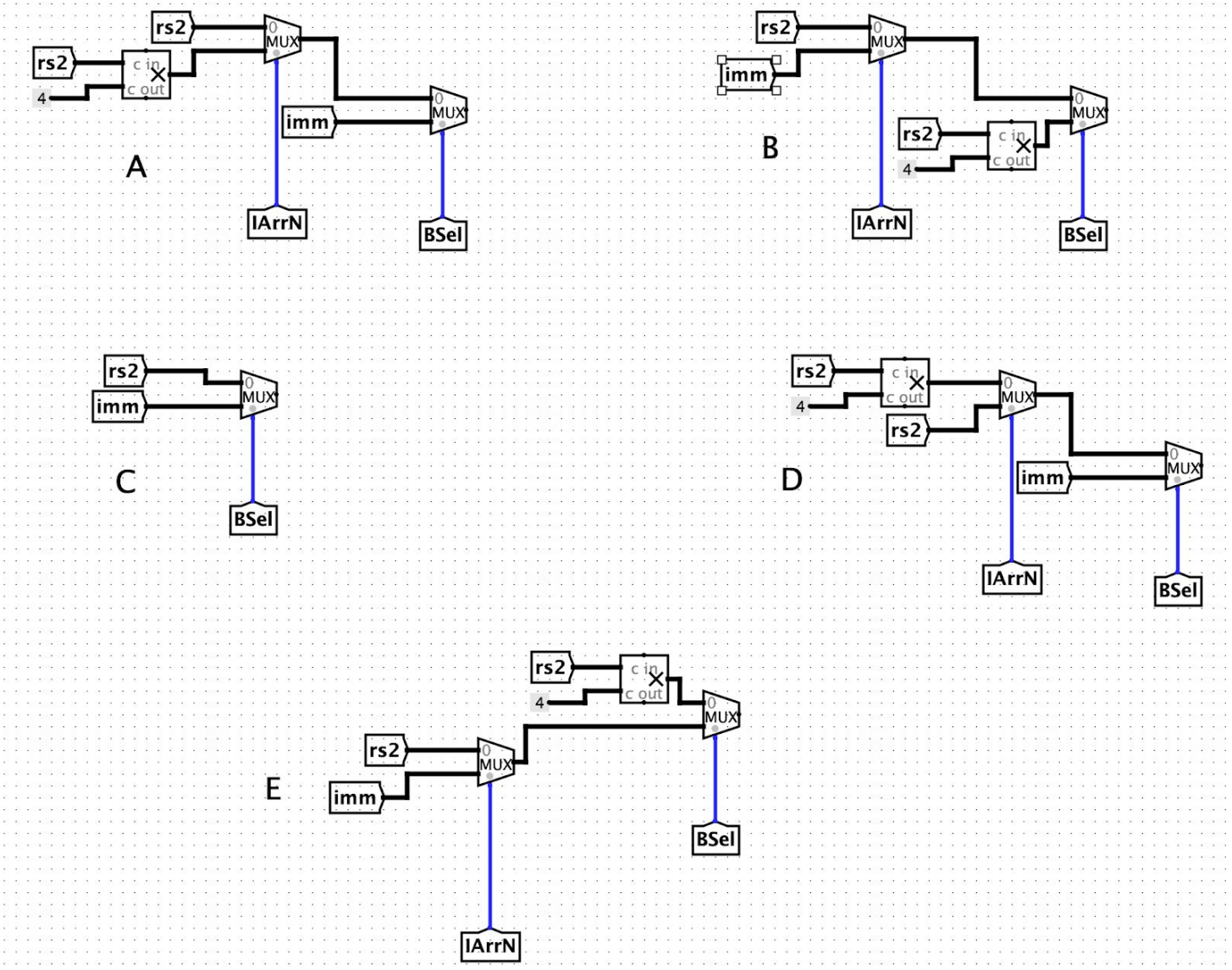


(Questions on next page)

How should we change BSel to allow our new instruction, and **all other RISC-V instructions**, to execute correctly?

Ⓐ Option A Ⓑ Option B Ⓒ Option C Ⓓ Option D Ⓔ Option E

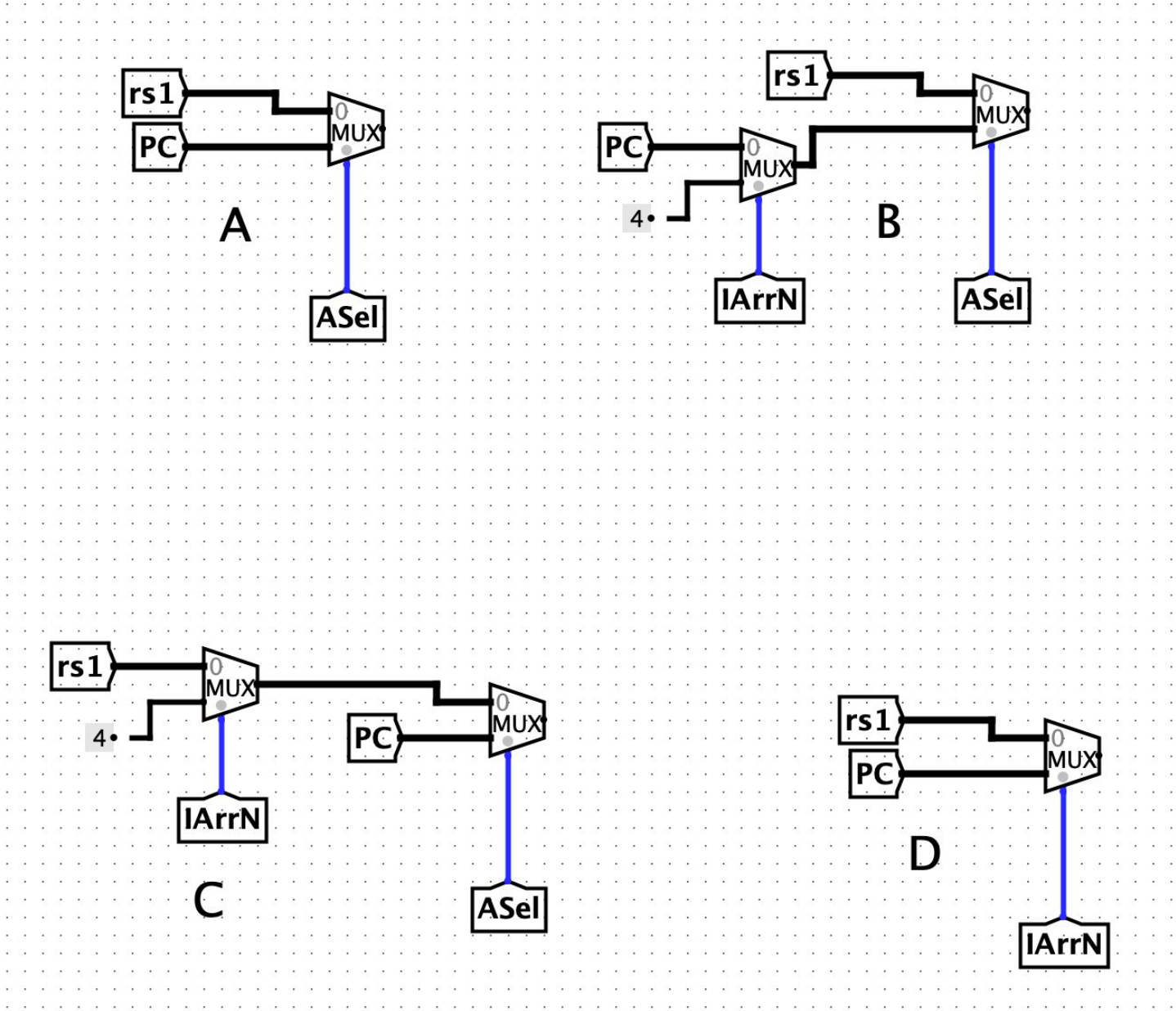
NOTE: some options showcase original hardware from the datapath. If you believe no changes are necessary, you should select this option. Assume our ALU, RegFile, and memory units remain unchanged internally.



How should we change ASel to allow our new instruction, and all other RISC-V instructions, to execute correctly?

Ⓐ Option A Ⓑ Option B Ⓒ Option C Ⓓ Option D

NOTE: some options showcase original hardware from the datapath. If you believe no changes are necessary, you should select this option. Assume our ALU, RegFile, and memory units remain unchanged internally.



Morgan later discovers she spends a lot of time writing assembly that increments the contents of integer arrays. She proposes another new instruction

```
iarrinc rd rs1 rs2
```

which reads the element pointed at by rs1, increments it by rs2, and writes the result to rd. In verilog:

```
R[rd] = M[R[rs1]] + R[rs2]
```

Morgan notices this instruction will *not* execute in our current datapath because it requires a memory access before the execute (ALU) stage. She proposes the following re-orderings. For each option, mark whether it would allow iarrinc to execute correctly and/or whether all other RISC-V instructions would execute correctly (given proper control is added).

Assume stages marked with numbers (ie. EX2) are duplications of the original stage. They may be idle or busy depending on the instruction's needs. Assume branch comparison happens in EX1. Assume all standard execution happens in EX1, and assume that EX2 is used only if an instruction requires additional ALU computation.

1. IF ID EX MEM WB
   [ ] iarrinc can execute
   [X] all other RISC-V instructions can execute

2. IF ID MEM EX WB
   [X] iarrinc can execute
   [ ] all other RISC-V instructions can execute

3. IF ID MEM1 EX MEM2 WB
   [X] iarrinc can execute
   [X] all other RISC-V instructions can execute

Morgan elects to create a new datapath with the following stages:

IF ID EX1 MEM EX2 WB

The following is a pipeline diagram for this CPU:

IF	ID	EX1	MEM	EX2	WB			
	IF	ID	EX1	MEM	EX2	WB		
		IF	ID	EX1	MEM	EX2	WB	
			IF	ID	EX1	MEM	EX2	WB

She pipelines her CPU and runs the following code segment.

```
        li t0 0
        iarrn t2 a0 t0

loop:
        beq t0 a2 end
        lw t1 0(t2)
        addi t1 t1 6
        addi a1 a1 4
        sw t1 0(a1)
        addi t0 t0 1
        iarrn t2 a0 t0
        j loop
end:
        ...
```

For each set of lines below, decide if a hazard exists between them, if there is none, select ‘no’ and leave the forwarding column blank. If a hazard exists and can be solved by forwarding, select the stage to forward *from* and forward *to*, otherwise, select “Must Stall”. Assume branch comparison happens in EX1. Assume execution can take place in either EX1 or EX2.

(Questions on next page)

Instructions	Hazard exists?	Forward from/to?
li t0 0 iarrn t2 a0 t0	[X] yes [ ] no (EX1-EX1, EX2-EX2, EX1-EX2, EX1-MEM, MEM-EX2, MEM-MEM)	[ ] Must Stall FROM: [ ] ID [ ] IF [X] EX1 [ ] MEM [ ] EX2 [ ] WB TO: [ ] ID [ ] IF [X] EX1 [ ] MEM [ ] EX2 [ ] WB
lw t1 0(t2) addi t1 t1 6	[X] yes [ ] no (MEM-MEM, MEM-EX2, EX2-EX2)	[ ] Must Stall FROM: [ ] ID [ ] IF [ ] EX1 [X] MEM [ ] EX2 [ ] WB TO: [ ] ID [ ] IF [ ] EX1 [ ] MEM [X] EX2 [ ] WB
addi a1 a1 4 sw t1 0(a1)	[X] yes [ ] no	[ ] Must Stall FROM: [ ] ID [ ] IF [X] EX1 [ ] MEM [ ] EX2 [ ] WB TO: [ ] ID [ ] IF [X] EX1 [ ] MEM [ ] EX2 [ ] WB
j loop beq t0 a2 end	[X] yes [ ] no	[X] Must Stall FROM: [ ] ID [ ] IF [ ] EX1 [ ] MEM [ ] EX2 [ ] WB TO: [ ] ID [ ] IF [ ] EX1 [ ] MEM [ ] EX2 [ ] WB

Question 5: Cache Only (WE DO NOT TAKE VENMO!!!) - 21 pts

Consider a write-back, write allocate cache with a total size of 128 B, with 2 sets each with 4 entries. If we have 512 KiB of total memory

1. How many bits are the tag, index, and offset fields of our address?

   Tag: $\log_2 2^{19} - 1 - 4 = 19 - 5 = 14$

   Index: $\log_2 2 = 1$

   Offset: $\log_2(128 / (2 * 4)) = \log_2 16 = 4$

Now imagine we have two different code programs we want to execute on our machine with the cache from above. Assume that all loads are executed from left to right, for all questions any arrays are block aligned, and that `sizeof(int) == 4`.

```
// Version 1
void reverse_array_1(int* arr, int size) {
      for (int i = 0; i < size / 2; i++) {
            arr[i] = arr[i] ^ arr[size - i - 1];
            arr[size - i - 1] = arr[i] ^ arr[size - i - 1];
            arr[i] = arr[i] ^ arr[size - i - 1];
      }
}
```

```
// Version 2
void reverse_array_2(int* arr, int size) {
      int* temp = malloc ((size / 2) * sizeof (int));
      for (int i = 0; i < size / 2; i++) {
            temp[i] = arr[i];
      }
      for (int i = size / 2; i < size; i++) {
            arr[size - i - 1] = arr[i];
            arr[i] = temp[size - i - 1];
      }
}
```

(Questions on next page)

Above we have two different working implementations that reverse the elements in an array.

2. If size = 16, what is the worst case HR for reverse_array_1?

We notice that the array does not fill the cache, so we can place every block at the cache at one time. Thus we only need to count the number of accesses and count compulsory misses.

We have 4 blocks, so 4 compulsory misses. We also have 8 iterations with 9 memory accesses per iteration.

Thus HR = 8 * 9 - 4 / 8 * 9 = 68 / 72 = 17 / 18

HR = 17/18

3. For reverse_array_1, assuming size is a power of 2, what is the largest value of size >= 16 that produces this worst case hit rate, or write no limit on the size.

Since we only work with 2 blocks at a time we only need space for 2 blocks in our cache. Thus no matter how big we make size we will maintain the same worst case hit rate.

Ⓐ 16 Ⓑ 32 Ⓒ 64 Ⓓ 128 Ⓔ 256 Ⓕ No Limit

4. If size = 16, what is the worst case HR for reverse_array_2?

Since the two arrays again fit inside our cache (96 < 128) we once more only have compulsory misses. Since there are 6 blocks we work with, there are 6 compulsory misses. Then we have to calculate the total accesses from the 2 loops.

Loop 1 has 8 *2 access. Loop 2 has 4 * 8 accesses. Putting those together we get

HR = (16 + 32 - 6) / (16 + 32) = 42 / 48 = 7 / 8

Note we don't have to worry about conflict misses because we have 4 entries per set

HR = 7 / 8

5. Keeping size = 16, what is the minimum associativity that keeps the same worst case hit rate for reverse_array_2 as question 4 while keeping the cache size and block size the same.

We are working with 3 different array sections. This makes it tempting to answer 4, but if we look at this a little more closely we see that we can actually have a lower associativity. First we examine arr. Notice how arr is laid out in memory. It consists of 4 total blocks:

Block 0	Block 1	Block 2	Block 3

Notice that 0, 1, 2, 3 never conflict mod 4, so each block of arr must be in a unique set. As a result our only conflict can come from the placement of temp. Since temp does not fill the cache, it can have at most 1 block conflict with each block in arr. This makes our answer N = 2.

Ⓐ 1 Ⓑ 2 Ⓒ 4 Ⓓ 8

6. Now consider a different cache setup. We have an L1 cache with a hit time of 2 cycles, L2 cache with a hit time of 18 cycles and a 15% local miss rate and a global miss rate of 5%. If main memory accesses take 60 cycles, what is the AMAT for this caching hierarchy in clock cycles.

L1 MR = .05 / .15 = ⅓

AMAT = 2 + ⅓ (18 + .15 * 60) = 2 + 6 + 60 * .05 = 2 + 6 + 3 = 11 cycles

AMAT = 11 cycles