Consider two competing 8-bit floating point formats. Each contains the same fields (sign, exponent, significand) and follows the same general rules as the 32-bit IEEE standard (denorms, biased exponent, non-numeric values, etc.), but allocates its bits differently. To save you time, *you only need to complete and circle the (LEFT or RIGHT) blank whose value is closest to zero*, that's the only one we'll grade! (If they're the same value, write the answer in both, & circle both). E.g,. The number represented by `0x00` was 0 for both, so we circled both. But for "exponent bias", just from the # of `EE…E` bits in each, we know |LEFT's bias| < |RIGHT's bias|, so there's no need to calculate or write the answer on the RIGHT.

"LEFT" format: | S | EE | MMMMM |

scratch space (show all work here)

"RIGHT" format: | S | EEEEEE | M |

scratch space (show all work here)

Number represented by `0x00`: 0

Exponent Bias: 1

a) # Numbers ($0 \le n < 1$): _____

b) # Numbers ($1 \le n < 2$): _____

c) Difference between two smallest positive values: _____

d) Difference between two biggest non-∞ values: _____

e) Positive Integer closest to 0 it <u>cannot</u> represent: _____

Number represented by `0x00`: 0

Exponent Bias: 31 (not graded, so no need to write)

Numbers ($0 \le n < 1$): _____

Numbers ($1 \le n < 2$): _____

Difference between two smallest positive values: _____

Difference between two biggest non-∞ values: _____

Positive Integer closest to 0 it <u>cannot</u> represent: _____

f) Which implementation is better for *approximating* π, LEFT or RIGHT ? (circle one)

1) For a 12-bit integer represented with two's complement, what is the:

a) Most positive value (in decimal): _____

b) Binary representation of that number: _____

c) Most negative value (in decimal): _____

d) Hex representation of that number: _____

e) In general, for an n-bit, two's complement integer:

i) What is the most positive you can represent, in decimal? _____

ii) What is the most negative you can represent, in decimal? _____

2) Fill in the blank below so that the function `mod16` will return the remainder of `x` when divided by 16. The first blank should be a bitwise operator, and the second blank should be a single decimal number:

```
unsigned int mod16(unsigned int x) {
    return x ________ _________________;
}
```

Connect the definition with the name of the process that describes it.

a) Compiler
b) Assembler
c) Linker
d) Loader

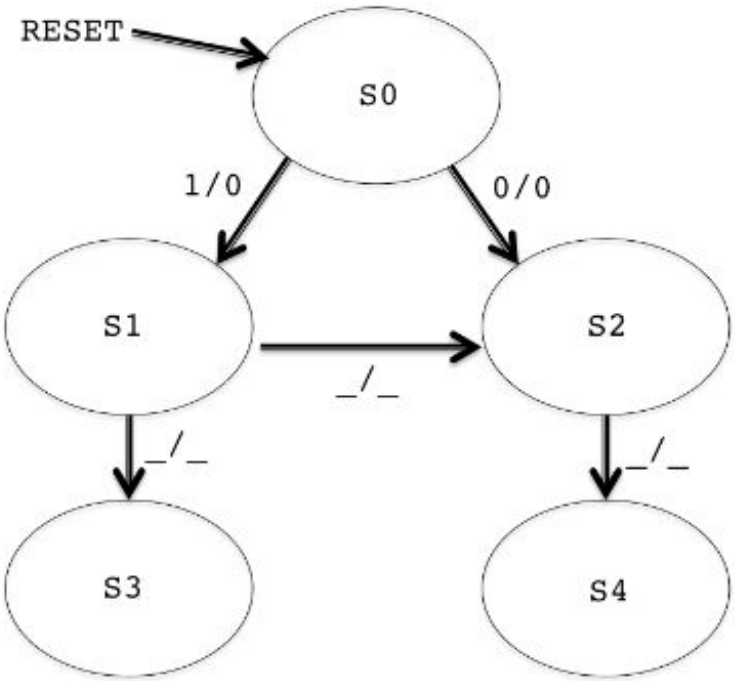
1) Outputs code that may still contain pseudoinstructions. _____
2) Takes binaries stored on disk and places them in memory to run. _____
3) Makes two passes over the code to solve the "forward reference" problem. _____
4) Creates a symbol table. _____
5) Combines multiple text and data segments. _____
6) Generates assembly language code. _____
7) Generates machine language code. _____
8) Only allows generation of TAL. _____
9) Only allows generation of binary machine code. _____

(b) 2-input NOR gates are said to be complete because any Boolean function can be computed with them. Prove this fact. Hint: implement a subset of the standard gates (AND, NOT, OR, NOR, NAND, XOR, XNOR) using just NOR gates, then apply a standard boolean algebra technique using these gates.

(c) We want to implement a very simple finite state machine that determines its next state by the result of and AND operation on the current state and the input. The output is always the current state. Assume registers have a CLK to Q delay of 5ns, a setup time of 2ns, and a hold time of 3ns. To achieve a clock rate of 25MHz, what is the maximum propogation delay that a NOR gate could have, assuming we are implementing AND as a combination of one or more of the gates built in part (b)?

(d) Complete the state diagram for a finite state machine that outputs 1 if and only if it has just seen the input sequence 101 and it has never seen the input sequence 001. You may add more arrows or more states as you see fit. Provide a brief description of each state.

```
Example
Input : 1101010100101
Output: 0001010100000
```



Considering the standard 32-bit RISC-V instruction formats, convert `lw t5, 17(t6)` to machine code:

(a) 0x _____

Prof. Wawrzynek decides to design a new ISA for his ternary neural network accelerator. He only needs to perform 7 different operations with his ISA: XOR, ADD, LD, SW, LUI, ADDI, and BLT. He decides that each instruction should be 17 bits wide, as he likes the number 17. There are no funct7 or funct3 fields in this new ISA.

(b) What is the minimum number of bits required for the opcode field?

_____

(c) Suppose Prof. Wawrzynek decides to make the opcode field 6 bits. If we would like to support instructions with 3 register fields, what is the maximum number of registers we could address?

_____

Assume we have two arrays `input` and `result`. They are initialized as follows:

```
int *input = malloc(8*sizeof(int));
int *result = calloc(8, sizeof(int));
for (int i = 0; i < 8; i++) {
    input[i] = i;
}
```

You are given the following RISC-V code. Assume register `a0` holds the address of `input` and register `a2` holds the address of `result` when `MAGIC` is called by `main`.

```
main:
    ...
    # Start Calling MAGIC
    addi a1, x0, 8
    jal ra, MAGIC      # a0 holds input, a2 holds result
    # Checkpoint:  finished calling MAGIC
    ...
exit:
    addi a0, x0, 10
    add a1, x0, x0
    ecall      # Terminate ecall
MAGIC:
    # TODO: prologue.  What registers need to be stored onto the stack?
    mv s0, x0
    mv t0, x0
loop:
    beq t0, a1, done
    lw t1, 0(a0)
    add s0, s0, t1
    slli t2, t0, 2
    add t2, t2, a2
    sw s0, 0(t2)
    addi t0, t0, 1
    addi a0, a0, 4
    jal x0, loop
done:
    mv a0, s0
    # TODO: epilogue.  What registers need to be restored?
    jr ra
```

(a) Consider the function `MAGIC`. The prologue and epilogue for this function are missing. Which registers should be saved/restored in `MAGIC`'s prologue/epilogue? Select all that apply.

- O t0
- O t1
- O t2
- O s0
- O a0
- O a1
- O a2
- O ra
- O x0

(b) Assume you have the prologue and epilogue correctly coded. You set a breakpoint at "Checkpoint: finish calling `MAGIC`" and call `main`. What does `result` contain when your program pauses at the breakpoint? Please write the 8 numbers starting at `result` in the blanks below.

_____ _____ _____ _____ _____ _____ _____ _____

(c) Translate `MAGIC` into C code. You may or may not need all of the lines provided below.

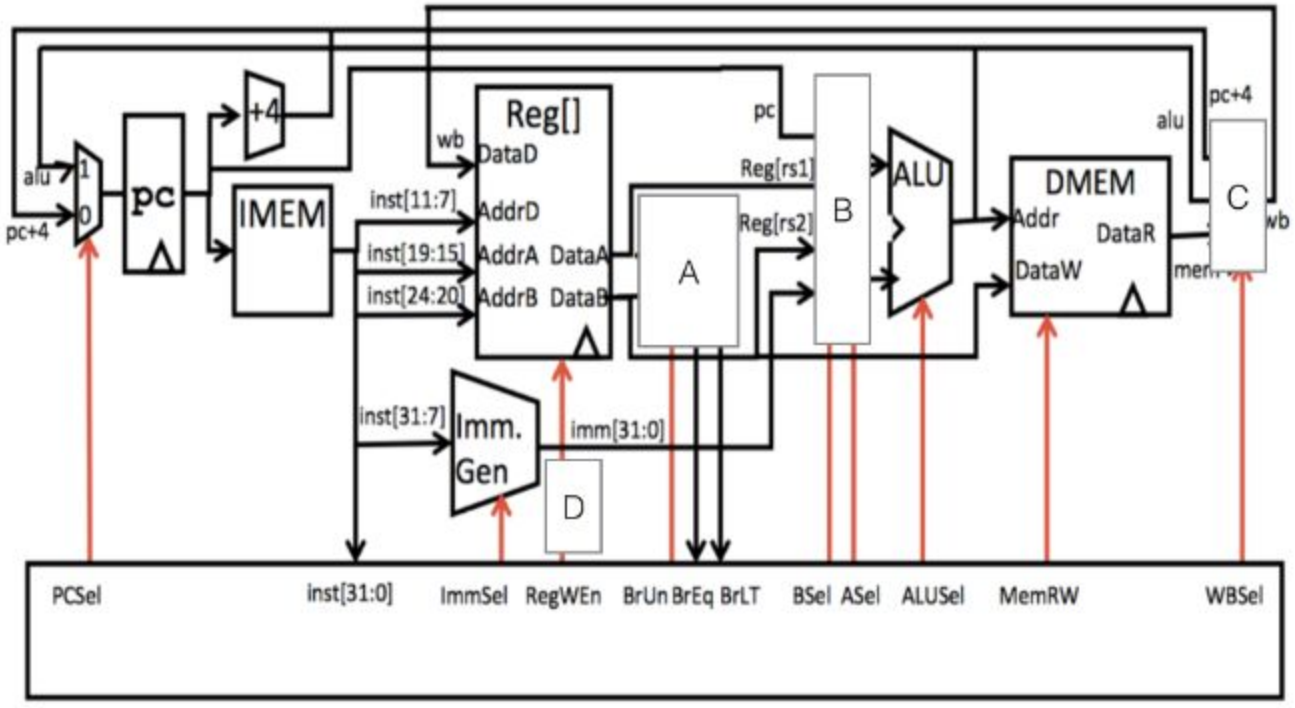
```
// sizeof(int) == 4
int MAGIC(_________ a, _________ b, _________ c) {

    _____________________________________________

    _____________________________________________

    _____________________________________________

    _____________________________________________

    _____________________________________________

    _____________________________________________

}
```

We wish to introduce a new instruction into our single-cycle datapath. The instruction **SIZ** (set if zero) works as follows:

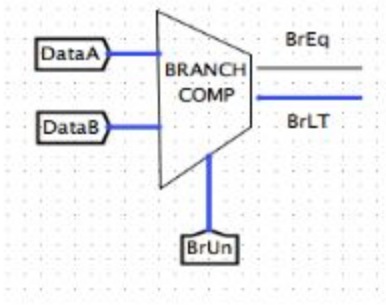
```
if (R[rs2] == 0):
    R[rd] = R[rs1]
```

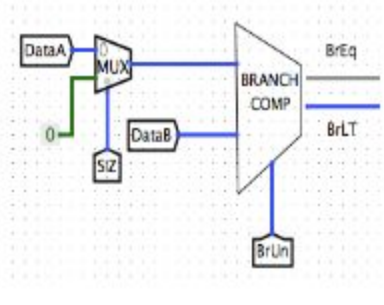
Given the single cycle datapath below, select the correct modifications in parts (a) - (d) such that the datapath executes correctly for this new instruction (and all core instructions!). You can make the following assumptions:

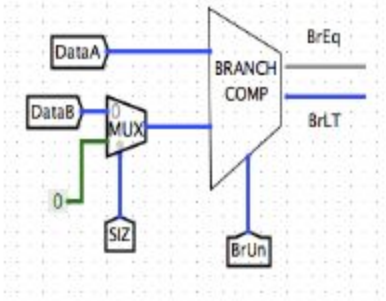
- the `SIZ` signal is 1 if and only if the instruction is `SIZ`
- `ALUSel` is **ADD** when when we a have `SIZ` instruction.
- **the immediate generator outputs ZERO** when we have a `SIZ` instruction.

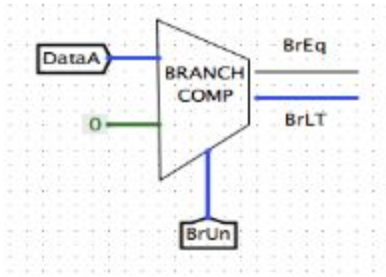


(a) Consider the following modifications to the branch comparator inputs. Which configuration will allow this instruction to execute correctly without breaking the execution of other instructions in our instruction set?

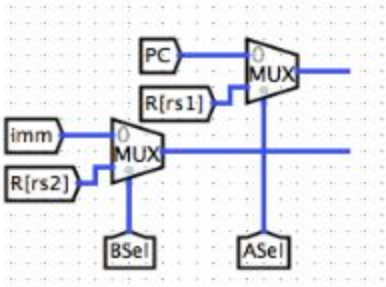
○ 

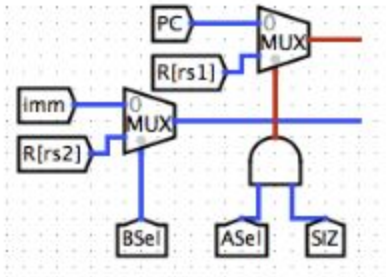
○ 

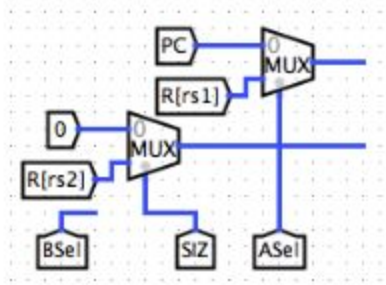
○ 

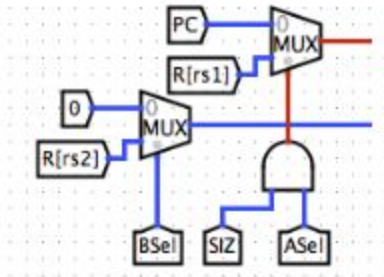
○ 

(b) Consider the following modifications to the ALU inputs. Which configuration will allow this instruction to execute correctly without breaking the execution of other instructions in our instruction set? Select the configuration that requires **minimum** modifications to the original datapath. Notice in the bottom left choice `BSel` is unused.

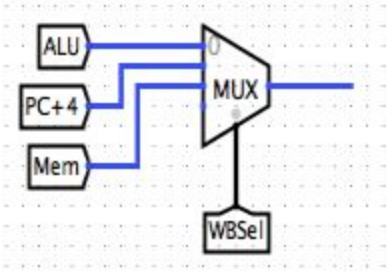
○ 

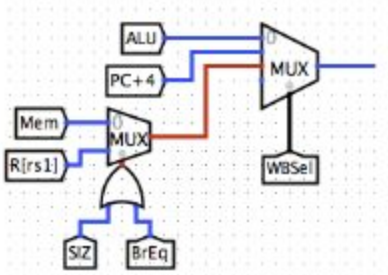
○ 

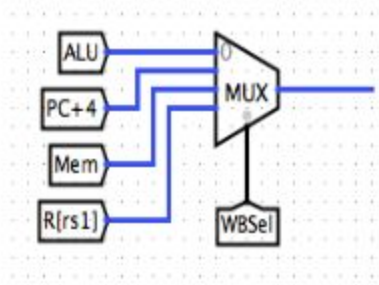
○ 

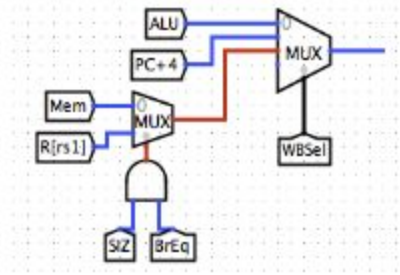
○ 

(c) Consider the following modifications to the WB mux inputs. Which configuration will allow this instruction to execute correctly without breaking the execution of other instructions in our instruction set? Select the configuration that requires **minimum** modifications to the original datapath.

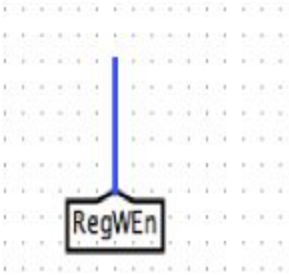
○ 

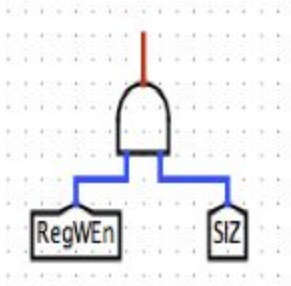
○ 

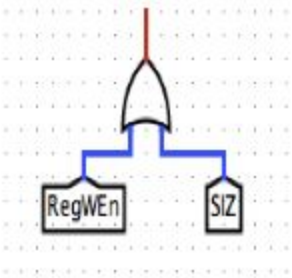
○ 

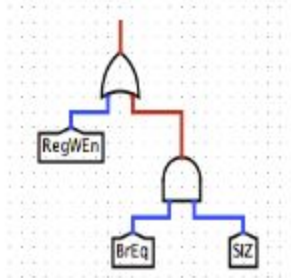
○ 

(d) Consider the following modifications to the RegWEn inputs. Which configuration will allow this instruction to execute correctly without breaking the execution of other instructions in our instruction set?

○ 

○ 

○ 

○ 

(e) Given your selections above, decide the rest of the control signals for this instruction based on the diagram given at the beginning of the problem. Select `X` when a signal's value doesn't matter. You can assume:

- the `SIZ` signal is 1 if and only if the instruction is `SIZ`
- `ALUSel` is **ADD** when when we a have `SIZ` instruction.
- the immediate generator outputs **ZERO** when we have a `SIZ` instruction.

1. `PCSel`:

   ○ 1 ○ 0 ○ X

2. `RegWEn`:

   ○ 1 (Enable) ○ 0 (Disable) ○ X

3. `BrUn`:

   ○ 1 (Signed) ○ 0 (Unsigned) ○ X

4. `BSel`:

   ○ 1 ○ 0 ○ X

5. `ASel`:

   ○ 1 ○ 0 ○ X

6. `MemRW`:

   ○ 1 (Enable) ○ 0 (Disable) ○ X

7. `WBSel`

   ○ ALUOut ○ MemOut

   ○ PC + 4 ○ Other: Please specify:_____