Instructions:

- Print your name in the space provided below.
- This examination is closed book and closed notes, aside from the permitted one-page formula sheet and the MIPS reference card. No calculators or other computing devices may be used.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- There are 7 questions, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- Note that either failing to return this test, or discussing its content with a student who has not taken it is a violation of the Honor Code.

Do not start the test until instructed to do so!

Name _____
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_____
signed

For questions 1 and 2, feel free to refer to the copy of Fig 5.24 from P&H that was distributed along with the test.

1. [6 points] Note the list of MIPS instructions that the given datapath supports. For which supported instructions is the component labeled Q1 needed?

   This multiplexor determines whether the data value sent back to the register file is taken from the ALU or from the data memory. This is needed for any R-type instruction (add, sub, and, or, and slt), since those use the ALU to compute a value that will be stored in a register, and for the load word instruction, since it causes a value to be fetched from data memory and stored in a register.

   So: `add, sub, and, or, slt, lw`

   It is not needed for the store word instruction, or any of the branch/jump instructions since they do not cause a value to be stored into a register. (However, the setting of the RegWrite control line would matter for those.)

   [10 points] Pick one such instruction, and describe carefully how that the control line for that component is used in the execution of the instruction.

   For the R-type instructions, as soon as the main control unit determines that we have an R-type instruction, the control line MemToReg must be set to 0 so the value from the ALU is sent to the register file's write data port.

   For the load word instruction, as soon as the main control unit determines that we have a load word instruction, the control line MemToReg must be set to 1 so the value fetched from the data memory is sent to the register file's write data port.

2. [6 points] For which supported instructions is the component labeled Q2 needed?

   This multiplexor determines whether the second operand to the ALU comes from the register file or from the low 16 bits of the instruction itself. This matters for every instruction that uses the ALU, which would include all of them except for the jump instruction:

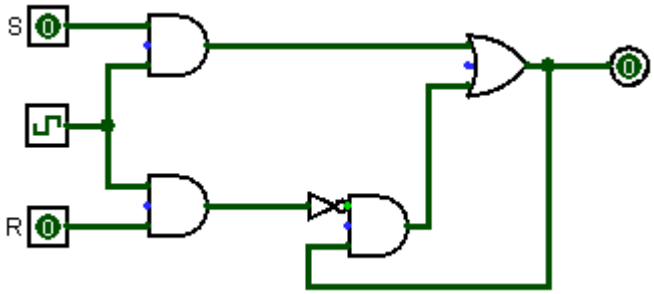
   `add, sub, and, or, slt, lw, sw, and beq.`

   [10 points] For which instructions, should the control line to this component be set to 1? Why?

   Setting the control line to 1 will cause the ALU to receive an operand from the instruction word. That is necessary for any instruction that contains a literal that must be passed to the ALU. In our case, that would be lw and sw since each of those contains a literal that needs to be added to the value in a base register.

   The beq instruction does include a literal, but in this datapath design, that value is shifted and then added to the value PC+4 using the second ALU unit shown at the top of the diagram. For beq, the control line for this mux should be set to 0 so the ALU will receive the values from the two registers that are to be compared.

3. The circuit shown below is a clocked SR-latch:

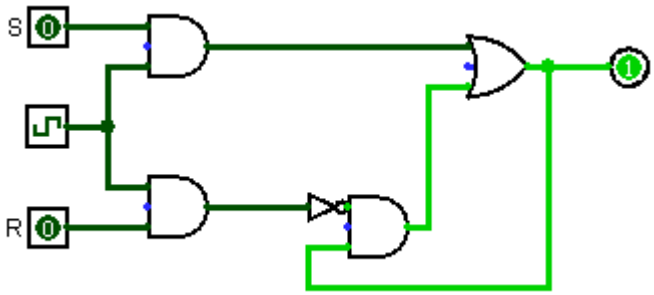


a) [8 points] Suppose that the input S is set to 0 (low). Describe precisely how the output value will change as the clock cycles from low to high and back to low.

When S ← 0, the output from the top AND gate does not change when the clock goes high. As a result, the output from the OR gate remains 0. Nothing changes for the bottom-left AND gate, since R is still 0. The bottom-right AND gate still receives one input of 0, so its output remains 0, which goes to the OR gate. When the clock goes low, the OR gate still receives two zeros, and so its output remains at 0.

So, the output from the circuit remains 0 when the clock goes high, and then remains at 0, no matter what the clock signal is.

b) [8 points] Suppose that the circuit is in the state shown below:



Suppose that the input R is set to 1. Describe precisely how the output value will change as the clock cycles from low to high and back to low.

When R ← 1, and the clock goes high, the bottom-left AND gate emits 1, which is negated before it reaches the bottom-right AND gate. So, the bottom-right AND gate now emits a 0, which goes to the OR gate. Thus, the OR gate will now emit a 0, which is fed back into the bottom-right AND gate; note this doesn't change the output from that AND gate, so the circuit is now stable. When the clock signal goes low, both of the left AND gates emit 0's. Neither the OR gate nor the bottom-right will change its output value as a result.

So, the output of the circuit changes to 0 when the clock goes high, and then remains 0, no matter what the clock signal is.

4. Consider the following integer X, represented in 32-bit 2's complement form:

```
0000 1111 0000 1111 0000 1111 0000 1111
```

a) [4 points] What is the representation for –X?

1111 0000 1111 0000 1111 0000 1111 0000 flip the bits

1111 0000 1111 0000 1111 0000 1111 0001 add 1

b) [8 points] Why is 2's complement representation used for signed integers instead of sign-magnitude representation? Be precise.

The fundamental reason is that this allows us to use the same algorithm (and therefore the same hardware) to add both signed and unsigned integer values.

A lesser reason is that sign-magnitude representation produces two different ways to represent zero.

A still lesser reason is that this increases the number of different integer values that can be represented… by one.

5. When two 32-bit signed integers, A and B, are added to produce a sum C, it is possible that *overflow* will occur.

a) [6 points] Define *overflow*. (Note that the answer here is different from the answer to the next part.)

Overflow is the condition that the correct result of an arithmetic computation, like addition, is too large to be represented correctly in the hardware scheme that is being used.

b) [6 points] It is possible to determine whether overflow has occurred by considering only the three high bits, A[31], B[31] and C[31]. Describe how this can be done.

In the case of addition, overflow can only occur when the sign of the computed result is inconsistent with the signs of the two operands. In other words, if the computed sum of two positive numbers is negative or if the computed sum of two negative numbers is positive, then overflow has occurred.

Moreover, overflow cannot occur if the two operands have opposite signs.

Since the high-order bits define the sign of the number there is a relatively simple check: A[31] XNOR B[31] is 1 if and only if A[31] and B[31] are the same, and so overflow has occurred if an only if the following expression evaluates to true:

(A[31] XNOR B[31]) && (A[31] XNOR C[31])

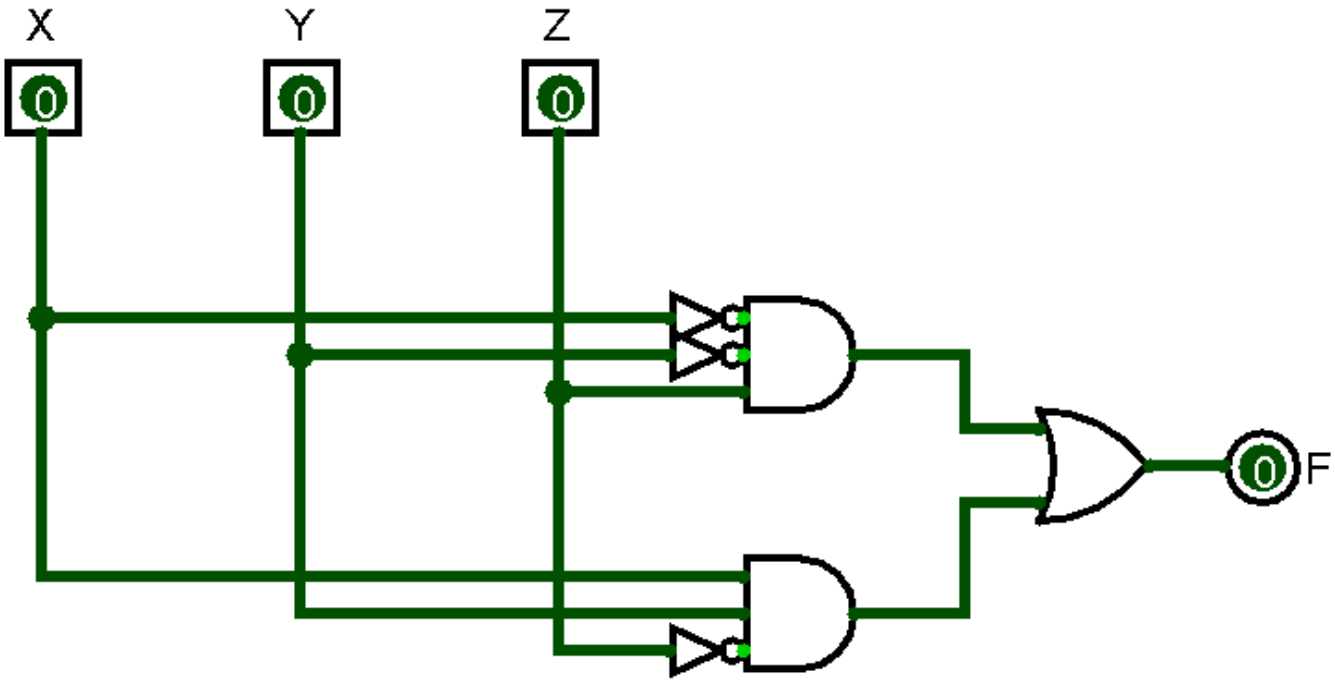
6. The following table defines a Boolean function F:

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

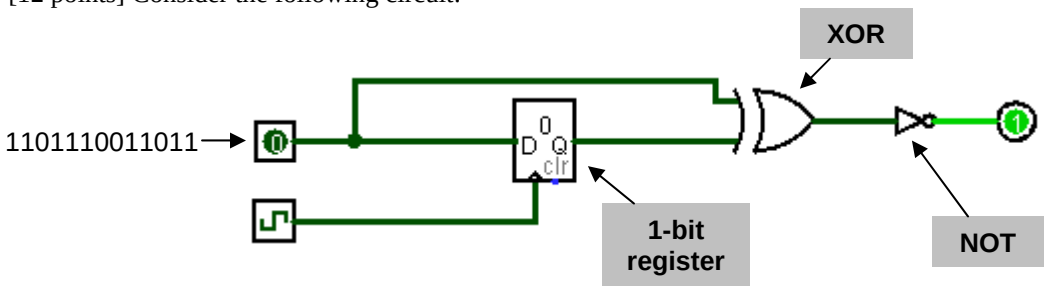
a) [8 points] Write the sum-of-products formula for the function F.

$$F = \bar{x} \cdot \bar{y} \cdot z + x \cdot y \cdot \bar{z}$$

b) [8 points] Draw the corresponding circuit to compute F.



7. [12 points] Consider the following circuit:



Suppose the input pin on the left side is connected to another circuit that can feed a sequence of bits into it (like a shift register). Assume the clock signal is enabled and that it ticks at exactly the same rate that bits are fed into the input pin.

a) [6 points] Starting with the initial state shown above, what would cause the output from the circuit change from 1 to 0?

If the value on the input pin is 1, then that will go to the XOR gate, and also be loaded into the register when the clock goes high. Since the output from the register also goes to the XOR gate, it would then receive two inputs of 1 and so its output would be 0, which would then be inverted to 1.

On the other hand, if the value on the input pin is 0, then, the XOR gate would receive an input of 0 from the input pin and still receive an input of 1 from the register (until the clock ticks), and so the XOR would output a 1 which would then be inverted to a zero. Once the clock ticks, the register would receive the input value of 0, and the XOR gate would revert to its previous state, and the circuit's output would revert to 1.

So, the circuit's output value would (temporarily) change to 0 if the input pin receives a value of 0.

b) [6 points] Give a precise, but brief description of what this circuit will do as a sequence of bits is fed into it. (Hint: consider attaching a counter circuit to the output line.)

The circuit's output is 1 as long as the value on the input pin is the same as the previous input bit (which is stored in the register until the clock ticks). The circuit's output temporarily changes to 0 when the value on the input pin is different from the previous input bit.

Now, any sequence of bits consists of "runs" of bits that are all of the same value. Reading from right to left, the example bit stream above starts with a run of two 1's, then has a run of one 0, then a run of two 1's, and so forth.

The circuit's output drops to 0 when we reach the end of a run, and then goes high again until we reach the end of the next run.

So, if we connected a counter to the output pin, along with some sort of synchronization to the clock signal, we could use this circuit to count the number of different runs, or even the length of runs in the input stream of bits.