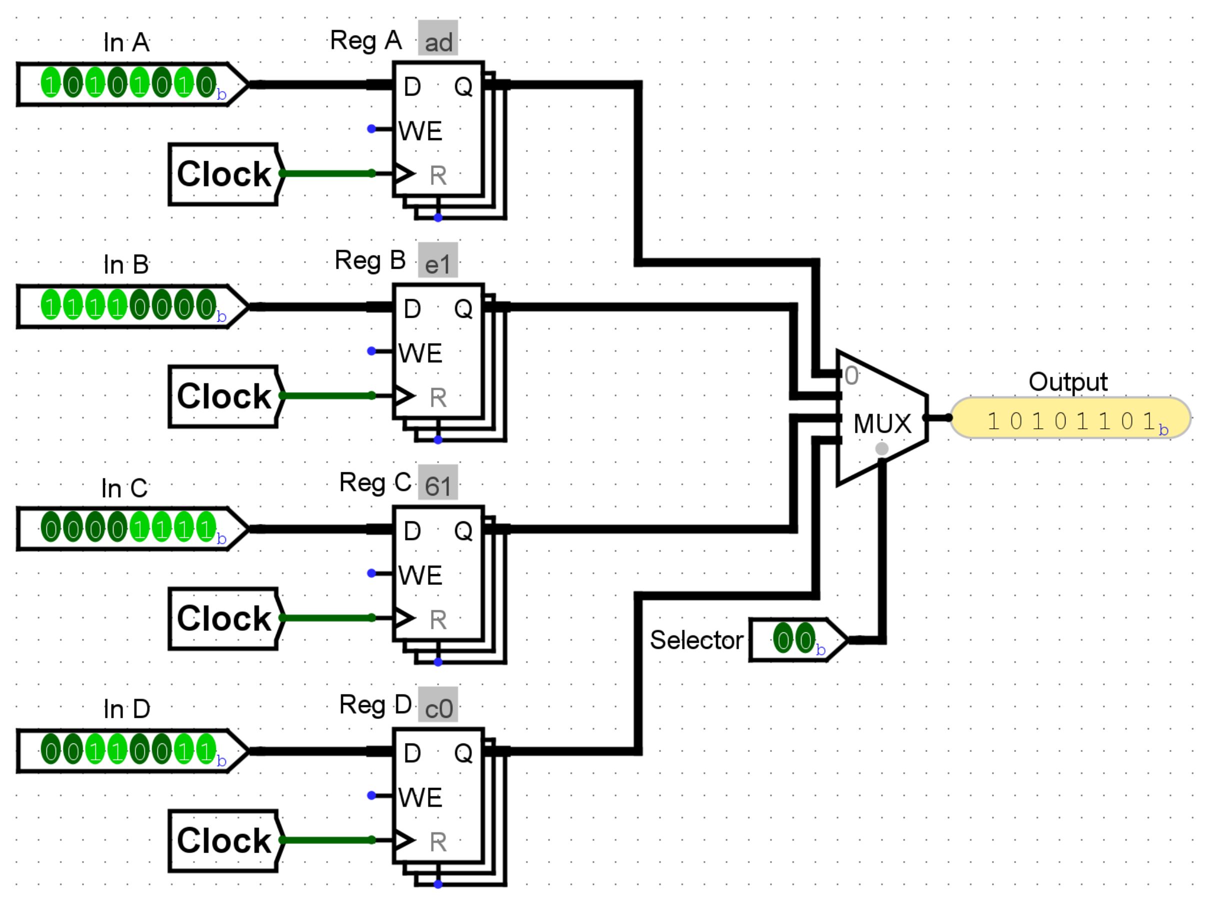
HW6.1. Register file implementation

A register file is a collection of X Y-bit registers. In RISC-V, for example, we have 32 different registers that we can access and each register is 32-bits each, giving us a 32 32-bit register file. To read a specific register, we can utilize a multiplexer to select the corresponding register that will show up at the output. A sample implementation of a 4 8-bit register file is shown:



This current register file setup allows us to read specific registers selected by the "Selector" input. Currently, we have 2 bit selector since we are just selecting from 4 registers. In RISC-V, 32 registers will correspond to a 5-bit selector.

The current register values are shown above each register (written in hex). Reg A, for example, has a current output of 0XAD or `10101101`. "Selector" = `00` corresponds to Reg A, "Selector" = `01` corresponds to Reg B, and so on.

For the following questions, **write the corresponding 8-bit output assuming everything else is held constant in the given figure, except for those indicated in the question.** Write your answer in binary without the 0b prefix.

Note that each question is independent of each other. Changes done in the previous questions will not reflect on the current question.

Hint: Recall that registers only update their outputs on the positive edge of the clock.

Q1.1: "Selector" = `01`

11100001 ? ✓ 100%

Q1.2: "In A" = `00001111`

10101101 ? ✓ 100%

Q1.3: "In D" = `00001111`, "Selector" = `11`

11000000 ? ✓ 100%

Q1.4: "In C" = `01101001`. *Positive edge of the clock occurs.* "Selector" = `10`

01101001 ? ✓ 100%

Q1.5: "In B" = `00001111`. *Positive edge of the clock occurs.* "Selector" = `10`

00001111 ? ✓ 100%

The main problem with this setup is that we need dedicated input ports *per register*. A true register file should allow us to write to a specific register on a single write port using a selector as well. One way to do this is to make all registers share the same input, and then control the "WE" ports of the registers so that only a specific register is updated depending on another selector input. If the "WE" of the corresponding register is 1, the register updates its value on the positive edge of the clock, else, the register will just hold its current value.

Homework 6

Assessment overview

Total points:	10/100
Score:	10%

Question

Value:	10
History:	10
Awarded points:	10/10

Report an error in this question

Previous question

Next question

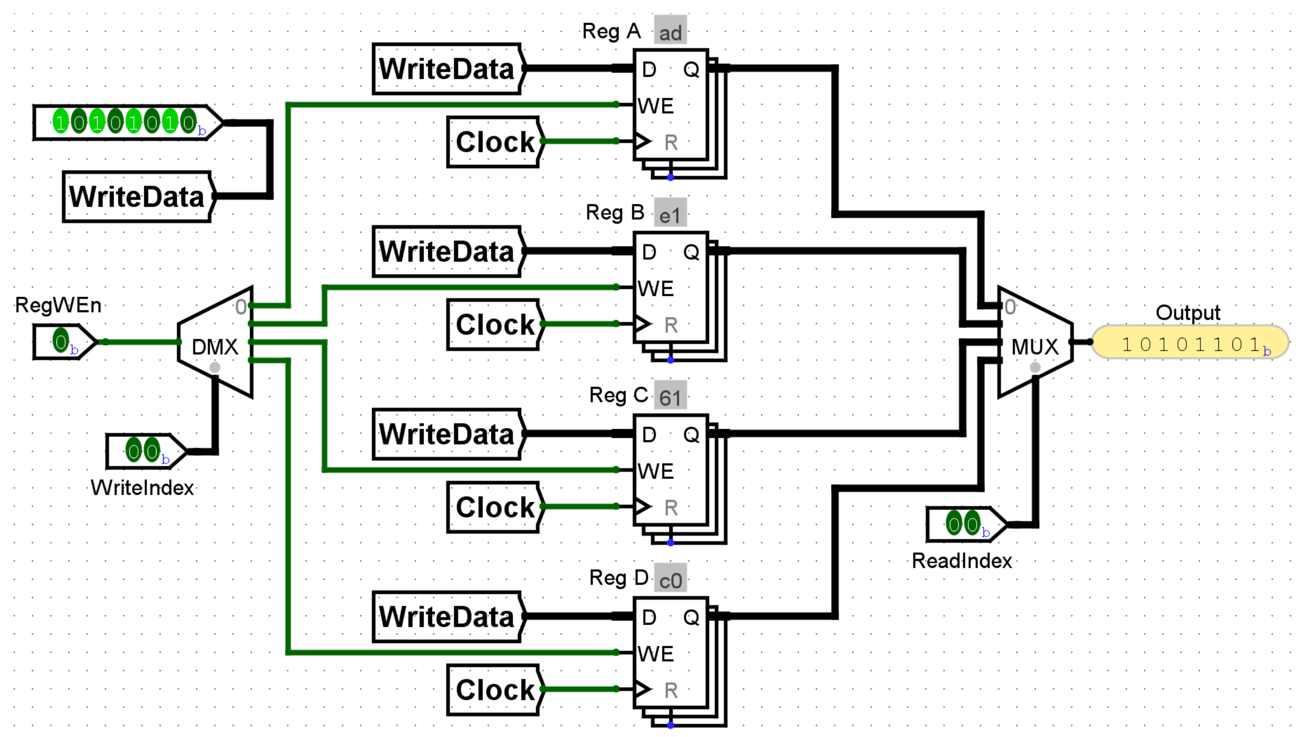
Attached files

No attached files

Attach a file

Attach text

Now, I introduce another combinational logic block called demultiplexer (or demux, for short) that will allow us to do that functionality. A demux is like the inverse of the multiplexer in terms of functionality. Depending on the selector bits, the input will be transferred to the corresponding output. The rest of the output bits that are not selected will be 0. A demux can be used to control the "WE" ports of each register in the register file as shown below:



For the following questions, **write the corresponding 8-bit output assuming everything else is held constant in the given figure, except for those indicated in the question.** Write your answer in binary without the 0b prefix.

Note that each question is independent of each other. Changes done in the previous questions will not reflect on the current question.

Hint: Recall that registers only update their outputs on the positive edge of the clock and if "WE" = `1`.

Q2.1: "ReadIndex" = `01`

11100001 ✓ 100%

Q2.2: Positive edge of the clock occurs.

10101101 ✓ 100%

Q2.3: "WriteIndex" = `01`, "ReadIndex" = `01`

11100001 ✓ 100%

Q2.4: "WriteIndex" = `10`. *Positive edge of the clock occurs.* "ReadIndex" = `10`

01100001 ✓ 100%

Q2.5: "WriteIndex" = `11`. "RegWEn" = `1`. *Positive edge of the clock occurs.* "ReadIndex" = `11`

10101010 ✓ 100%

Which of the following statements are true about register files?

Also, once you have correctly answered the following questions, spend some time reading the provided explanations.

- [ ] (a) Let's say *after* the positive edge of the clock, "RegWEn" becomes `1`. Since "WriteIndex" = `00`, we will see an updated value of RegA right away.
- [✓] (b) If we want to read two register values at once, we need another multiplexer with a different "ReadIndex" selector. ✓
- [ ] (c) Since all registers share the same input "WriteData", all registers will change their values on the positive edge of the clock.
- [ ] (d) Due to the presence of the demux, writing/updating the register values is purely combinational (i.e. no need to wait for the positive edge of the clock to update the register value).
- [✓] (e) If "RegWEn" = `1`, then on the next positive edge of the clock, the corresponding register selected by "WriteIndex" will update its value. ✓
- [ ] (f) In RISC-V, if an instruction does not need to write to the register file (e.g. store instructions), then we don't care what "RegWEn" should be.

- [✓] (g) Reading a specific register value is purely combinational (i.e. no need to wait for the positive edge of the clock), since we only need to change the "ReadIndex". ✓
- [✓] (h) If "RegWEn" = `0`, we don't care what "WriteIndex" is since the registers will not change their current values anyway. ✓

Select all possible options that apply.

✓ 100%

Try a new variant

Correct answer

Q1.1: "Selector" = `01`

`11100001`

Q1.1: "Selector" = `01`, then you are selecting the current output of Reg B, which is 0xE1 or `11100001`

Q1.2: "In A" = `00001111`

`10101101`

Q1.2: "In A" = `00001111`, "Selector" remains `00`, so it is still selecting the current output of Reg A, which is 0xAD or `10101101`

Q1.3: "In D" = `00001111`, "Selector" = `11`

`11000000`

Q1.3: "In D" = `00001111`, "Selector" = `11`, then it is selecting the current output of Reg D, which is 0xC0 or `11000000`, changing "In D" does not matter if there's no positive edge of the clock that will update the register output.

Q1.4: "In C" = `01101001`. *Positive edge of the clock occurs.* "Selector" = `10`

`01101001`

Q1.4: "In C" = `01101001`. *Positive edge of the clock occurs.* "Selector" = `10`, then it is selecting the current output of Reg C, which has been updated to whatever the value of "In C" = `01101001` due to the positive edge of the clock.

Q1.5: "In B" = `00001111`. *Positive edge of the clock occurs.* "Selector" = `10`

`00001111`

Q1.5: "In B" = `00001111`. *Positive edge of the clock occurs.* "Selector" = `10`, then it is selecting the current output of Reg C, which has been updated to whatever the value of "In C" = `00001111` due to the positive edge of the clock.

Q2.1: "ReadIndex" = `01`

`11100001`

Q2.1: "ReadIndex" = `01`, then you are selecting the current output of Reg B, which is 0xE1 or `11100001`

Q2.2: Positive edge of the clock occurs.

`10101101`

Q2.2: Positive edge of the clock occurs. While Reg A is being selected by "WriteIndex" = `00`, "RegWEn" = `0` so no register updates will occur. The output passes through the current value of Reg A, which is 0xAD or `10101101`, because it is being selected by "ReadIndex" = `00`.

Q2.3: "WriteIndex" = `01`, "ReadIndex" = `01`

`11100001`

Q2.3: "WriteIndex" = `01`, "ReadIndex" = `01`, then you are just changing the output being read to the current value of "Reg B" = `11100001`. No positive edge of the clock occurs and "RegWEn" = 0 so no register updates will occur.

Q2.4: "WriteIndex" = `10`. *Positive edge of the clock occurs.* "ReadIndex" = `10`

`01100001`

Q2.4: "WriteIndex" = `10`. *Positive edge of the clock occurs.* "ReadIndex" = `10`, then we are just changing the output being read to the current value of "Reg C" = `01100001` because "RegWEn" is not 1 when the positive edge of the clock occured.

Q2.5: "WriteIndex" = `11`. "RegWEn" = `1`. *Positive edge of the clock occurs.* "ReadIndex" = `11`

`10101010`

Q2.5: "WriteIndex" = `11`. "RegWEn" = `1`. *Positive edge of the clock occurs.* "ReadIndex" = `11`, then we will see the updated value of "Reg D" = `10101010`, since that is what "WriteData" currently is and "RegWEn" = 1 and "Reg D" is the one being selected by the "WriteIndex".

If we want to read two register values at once, we need another multiplexer with a different "ReadIndex" selector. This is true. The outputs of each multiplexer will correspond to the two read ports of the register file.

Reading a specific register value is purely combinational (i.e. no need to wait for the positive edge of the clock), since we only need to change the "ReadIndex". This is true. Register file reads are always combinational due to the multiplexer. Each register already holds their current values. The multiplexer is just selecting which register should be read outside the register file.

If "RegWEn" = `0`, we don't care what "WriteIndex" is since the registers will not change their current values anyway. This is true. Store instructions, for example, do not write to the register file. The "WriteIndex" can be anything, as long as "RegWEn" = `0`, then nothing in the register file will be updated.

If "RegWEn" = `1`, then on the next positive edge of the clock, the corresponding register selected by "WriteIndex" will update its value. This is true. Almost all RISC-V instructions will update the register file, and this is done by setting "RegWEn" = `1` and "WriteIndex" to the corresponding destination regiter.

Since all registers share the same input "WriteData", all registers will change their values on the positive edge of the clock. This is false. The main purpose of the demux is to control which registers will update by controlling the "WE" port. While all registers do share the same input, only 1 register will update if "RegWEn" = `1`.

Let's say *after* the positive edge of the clock, "RegWEn" becomes `1`. Since "WriteIndex" = `00`, we will see an updated value of RegA right away. This is false. If "RegWEn" just became 1 *after* the positive edge of the clock, then you need to wait for the next positive edge for the register to update its value.

Due to the presence of the demux, writing/updating the register values is purely combinational (i.e. no need to wait for the positive edge of the clock to update the register value). This is false. Updating the register values will always be synchronous to the clock, because that's how flipflops work. You need the positive edge of the clock for the register output to update.

In RISC-V, if an instruction does not need to write to the register file (e.g. store instructions), then we don't care what "RegWEn" should be. This is false. Don't care means it can be 0 or 1. If "RegWEn" = `1`, then the register file will update. If the store instruction executes, you should guarantee that "RegWEn" = `0` to ensure no register updates occur.

Submitted answer 6 correct: 100%
Submitted at 2022-10-23 04:14:26 (PDT)
show

Submitted answer 5 partially correct: 81%
Submitted at 2022-10-23 04:13:53 (PDT)
show

Submitted answer 4 partially correct: 81%
Submitted at 2022-10-23 04:13:23 (PDT)
show

Show/hide older submissions