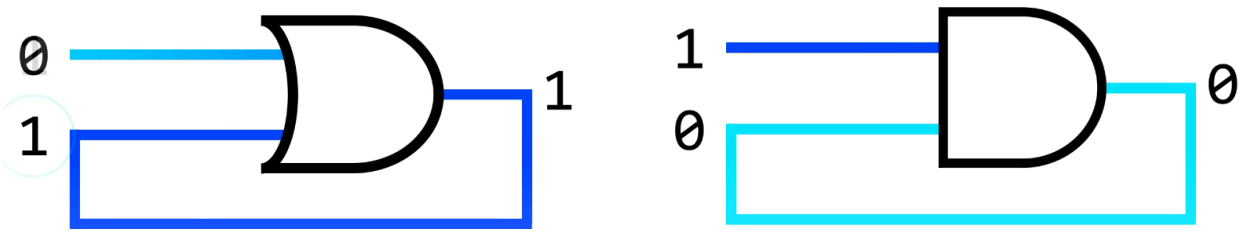
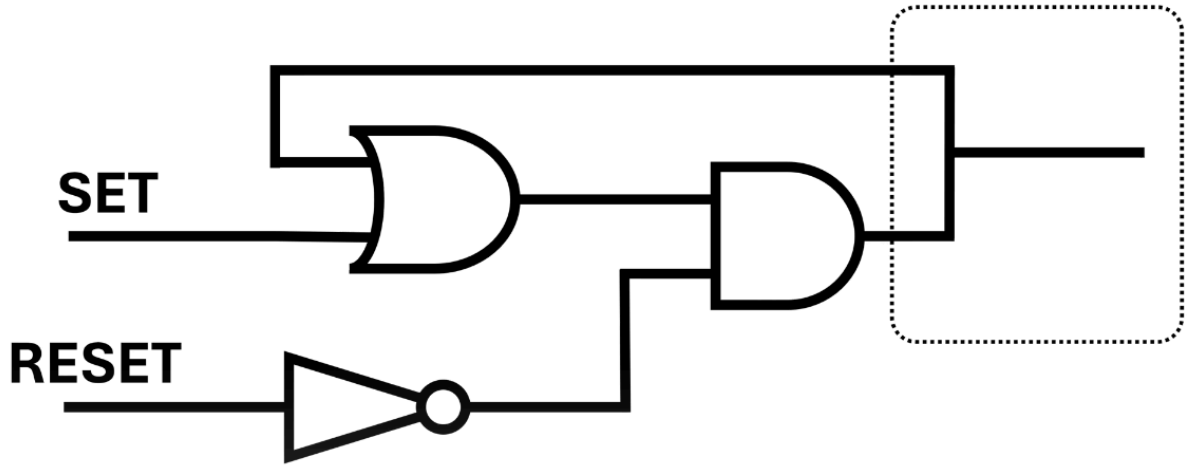
Chapter 2 - Memory

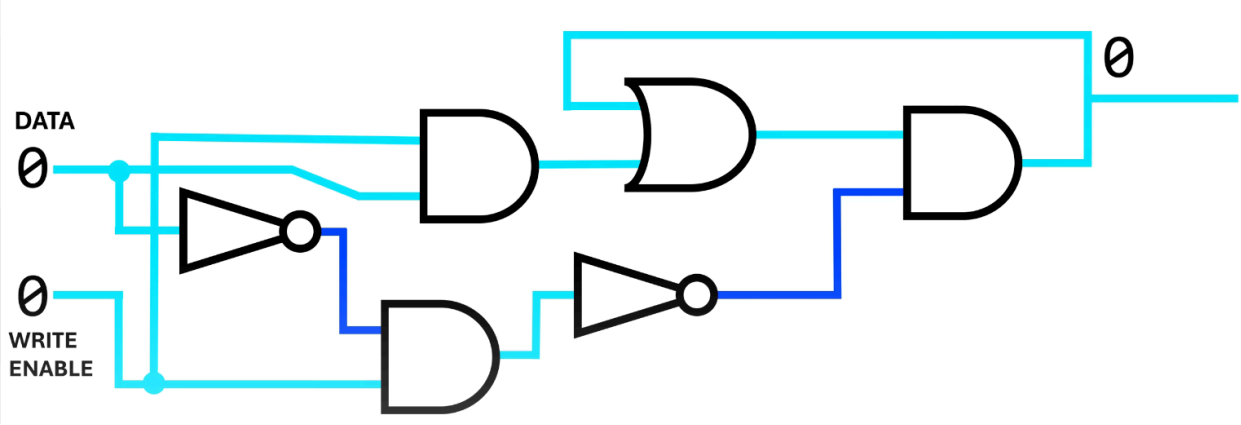
Previously, we learnt about the ALU. It takes in two operands, along with an opcode and outputs a single value along with a overflow/zero/negative flag. We notice that this whole block is purely combinational - that it doesn't depend on previous inputs - it does not use memory. But this is not the case with a lot of components. We definitely NEED memory for our CPU to function and since our goal is to understand everything from the transistor level, we should ask ourselves - how can we make transistors "remember" data? To answer this question, lets look at two particular structures - an AND gate and an OR gate with feedback as seen below.



These two structures can be used to remember 0 and 1 respectively. If we input a 1 into an OR gate, the output will remain 1 no matter what the next input is - the AND gate is similar with 0. But these structures are not much use on their own, as once set they cannot be reset. We can solve this problem by combining the two to form a Set-Reset latch - this is a structure that can be 'Set' to remember a 1 but can also be reset by setting the 'Reset' line high.



The problem with this structure becomes pretty obvious if we try to set both ‘Set’ and ‘Reset’ high. We do not get a fixed defined behavior here - referred to as a race condition. We can take this one step higher now, to obtain a gated latch which fixes this problem.



The structure of a Gated Latch is slightly more complex but this functions in a way such that the data is only sent to output if write-enable is high. If we want to remember a value at a particular time, we can set write enable low at that instant as the data would be stored at the output then until write enable is high again.

A gated latch does not have the problem of race conditions and is relatively simple and is therefore perfect for storing memory. A single gated latch can remember one bit of data but if we connect multiple, we can remember multiple bits of data and they become more useful. For instance, we can put 8 of these in parallel with

all the write enables connected together as below. This gives us the ability to store 8 bits, making this a 8 bit register!