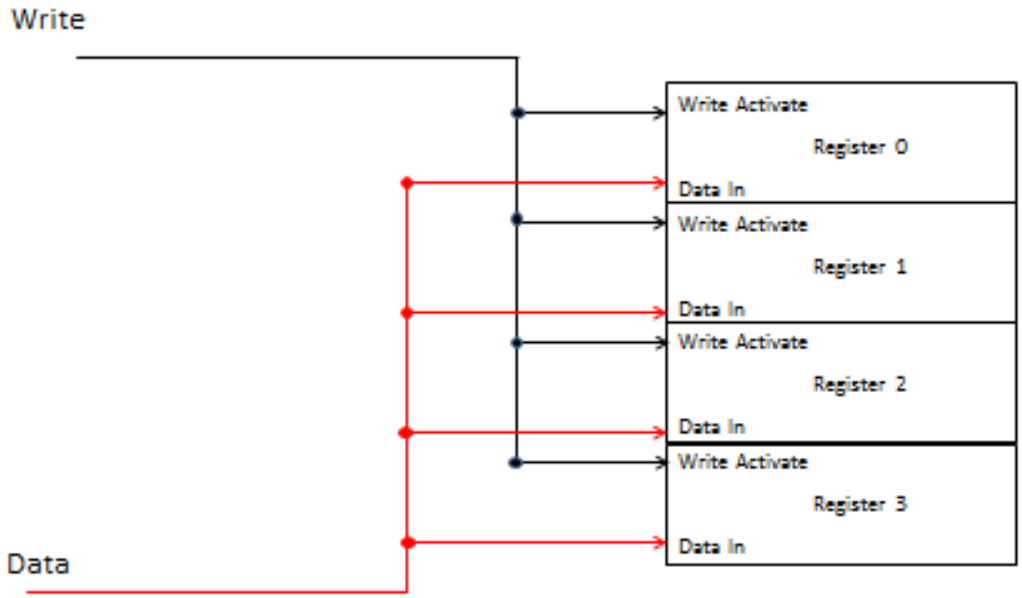
Example. The Binary Decoder

A binary decoder is a circuit that has *n* inputs and (usually) up to 2^n outputs. Based on the *n* inputs, one of the decoder's outputs is selected.

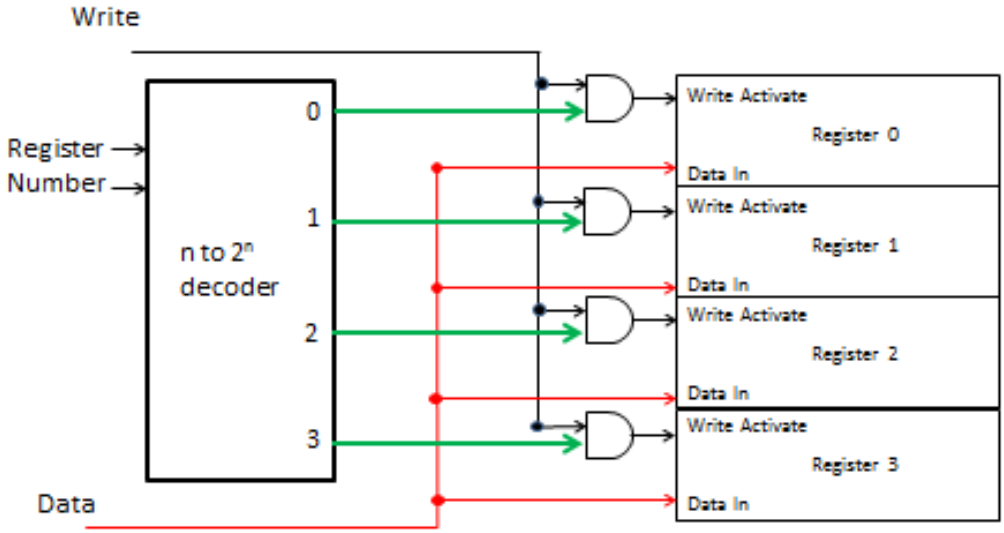
To motivate the idea, consider the small section of a Central Processing Unit (CPU) shown below. This CPU has a four special purpose high-speed *registers* to hold important data. To write data to a register, we need to place a logical 1 on its *Write Activate*, and then the data can be fed in on the input marked *Data In*.

Say we want to write the data value `2015` to register 2 (and only register 2!). When we want to write data to any register, we activate the `Write` signal. We would then place the data `2015` on the line labeled `Register Data`. What happens?



How do we ensure that the data 2015

This is where a decoder comes in! On the decoder's 2-bit input, we will place the



Design a two-input, four-output decoder. Truth table:

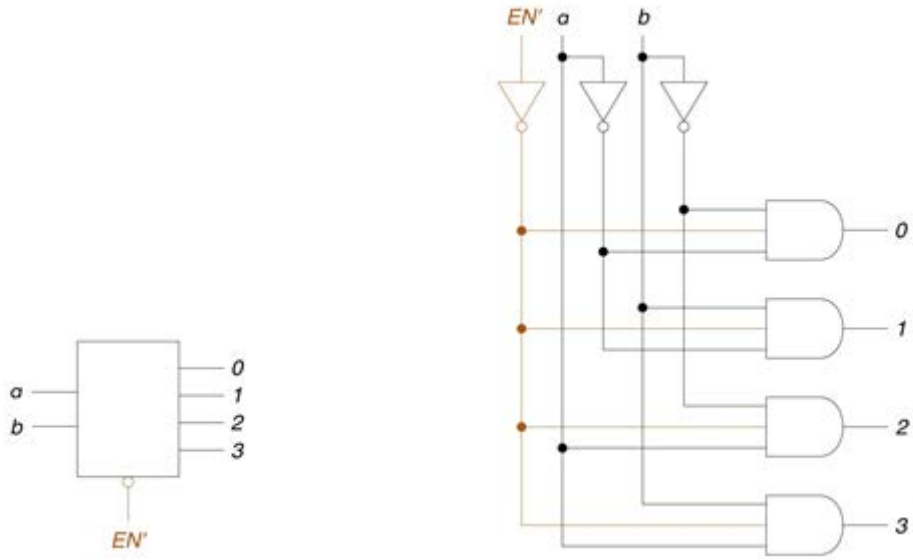
Circuit:

Suppose you would like to implement this decoder using an active low output. What would be your truth table?

Circuit:

Decoders usually have one or more *enable* inputs. When an enable input is active, the encoder works as described above. When the enable input is not active, all of the decoder's outputs are set to the inactive state (i.e., 0 if active high, 1 if active low).

Consider the decoder shown below, as a block diagram and as a circuit.



From A. Marcovitz, *Introduction to Logic Design*, McGraw Hill, 2010

Are the decoder outputs active high or active low?

True or False: Each of the outputs corresponds to a minterm.

Looking at the diagram on the right, would you say that the enable signal (EN′) is active high or active low?

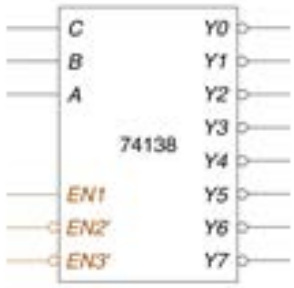
_____on the block diagram (see above right).

So, the decoder will behave normally when the input on the line labeled EN′ is equal to

Your coworker presents a truth table for the circuit above, shown below. You immediately notice that the truth table has three inputs (good!) but only 8 lines. You seem to remember that truth tables with three inputs usually have $2^3 = 8$ lines. Is your coworker a doofus?

EN'	a	b	0	1	2	3
1	X	X	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

Example. Consider the block diagram for the 74138 decoder shown below, where the inputs are *C*, *B* and *A*.



What are the outputs?

Why are there 8 outputs?

Are the outputs active high or active low?

What can you say about the enable signals?

What values for the enable signals will enable the decoder?

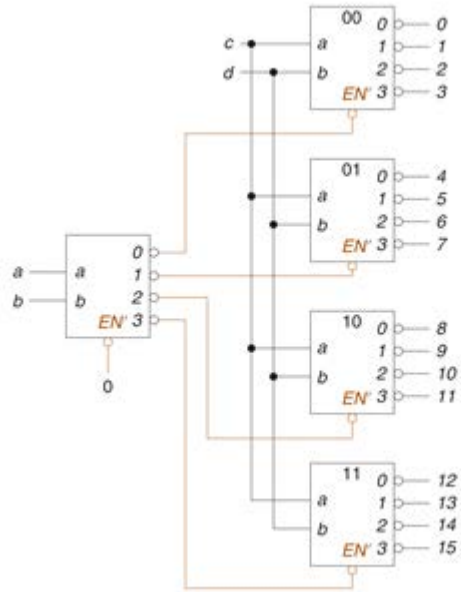
Write the truth table just for output Y0.

Table 5.3 The 74138 decoder.

Enables			Inputs			Outputs							
EN1	*EN2'*	*EN3'*	*C*	*B*	*A*	*Y0*	*Y1*	*Y2*	*Y3*	*Y4*	*Y5*	*Y6*	*Y7*
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

From A. Marcovitz, *Introduction to Logic Design*, McGraw Hill, 2010

Example. You need a 4-input, 16-output decoder, but you only have 2-input, 4-output decoders available. The input is *abcd*, with *a* being the MSB. Would the following design work? If yes, explain the operation of the circuit. If not, explain why the circuit does not work and attempt to modify the design to achieve a working circuit.



Example. You need to implement the function $f(A,B,C)=\sum m(1,3,6)$. You have scoured the lab, and the only two chips that do not have broken pins are one OR 4-input OR gate and a three-input eight-output active high decoder. Is there hope?

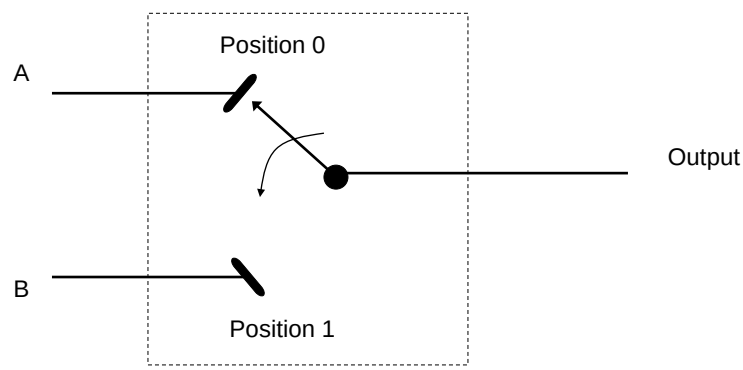
Example: The Binary Encoder

Easy! Just the inverse of a binary encoder. It has 2^n inputs and n outputs. In a simple binary encoder, only one of the inputs can be active, and the output gives which of the inputs are active.

For example, in a simple 4-input, 2-output encoder, the input 0010 would provide an output ___ and the input 1000 would provide an output of ___.

Example: The Multiplexer (MUX)

Consider the following device: The device has two inputs, labeled *A* and *B*, and one output. Internally, the device has a toggle switch that can be in one of two positions. In the "0" position, it connects *A* directly to the output. So, in this case, the bit values present at *A* will be fed directly to the output. In position "1", on the other hand, it connects *B* directly to the output. So, our device is a ***selector***: it chooses to send either *A* or *B* to the output. This device is more commonly referred to as a ***2-to-1 multiplexer***.

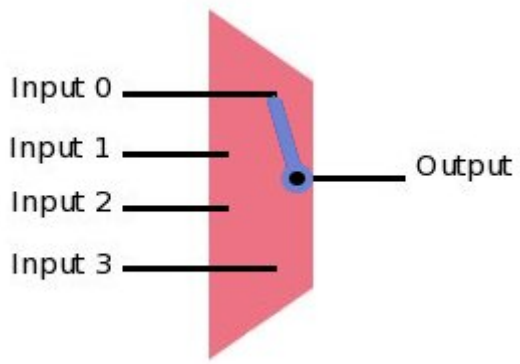


Suppose that the switch is controlled by the variable *S*. When $S = 0$, input *A* is passed to the output. When $S = 1$, input *B* is passed to the output. Draw the digital logic circuit that will implement a 2-to-1 multiplexer.

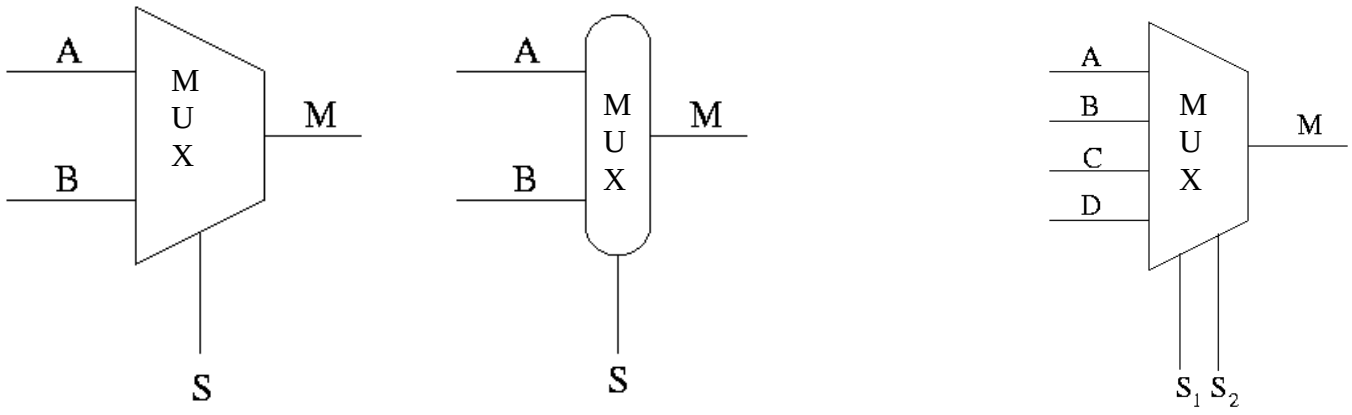
In general, a multiplexer has a number of inputs, and selects one of the inputs to send to the output.

Example. A 4-to-1 Multiplexer (a 4-way Multiplexer)

Design a 4-to-1 multiplexer. The multiplexer has four inputs, labeled *A*, *B*, *C* and *D*, and selects one of these four inputs to send to the output.



Multiplexers are used so often that they are fabricated as individual integrated circuits. The symbol for a multiplexer is usually either a quadrilateral or a flattened ellipse with the symbol **MUX** within it.



2-to-1 MUX

4-to-1 MUX

Example. Multiplexer Selector Bits

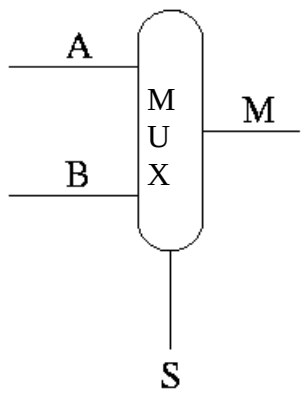
Suppose we wanted to design a multiplexer that would select one of 8 inputs to pass on to the output. How many selector bits would we need?

Suppose we wanted to design a multiplexer that would select one of 32 inputs to pass on to the output. How many selector bits would we need?

Suppose we wanted to design a multiplexer that would select one of *n* inputs to pass on to the output. How many selector bits would we need?

Example. Multiplexer with 2-bit-wide Inputs

Suppose that we want to design a multiplexer that will select from one of two inputs, but now each of the two inputs consists of two bits. In other words, the first input consists of the 2 bits A_1A_0 and the second input consists of the 2 bits B_1B_0. The output will be either A_1A_0 or B_1B_0. Design the multiplexer. To simplify your design you may use the 2-to-1 building block shown below (where, in this diagram, *A* and *B* are each one bit):

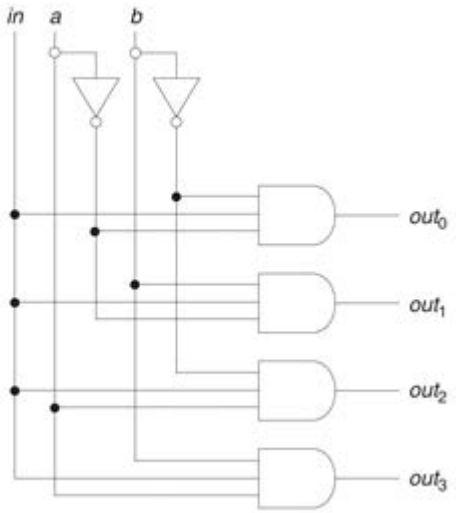


Example. Design a circuit that accepts 2 bits, *a* and *b*, as inputs, along with a bit designated Operation, and is able to perform the operations AND and OR on these 2 bits as indicated by the Operation bit.

Example. Design a four-way multiplexer using only tw0-way multiplexers.

Example: The Demultiplexer

Easy! Just the inverse of a multiplexer! It's input is a single signal which is routed to one of many possible output locations. A 4-way demultiplexer is shown below.



Example. (from text) Implement the following function:

a b c	f
0 0 0	1
0 0 1	1
0 1 0	1
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	0
1 1 1	0

You have available an 8-way multiplexer, four OR gates, 26 AND gates, some wire and a small rabbit. Get to work!