Integrated circuits and the ALU

DOES ANYONE KNOW WHAT CAME NEXT?
SURE, WONDER WOMAN, I DO! I REMEMBER THAT FROM OUR EARLIER LESSONS ON COMPUTERS.
AFTER THE TRANSISTOR CAME THE INTEGRATED CIRCUIT... COMBINING MANY TINY CIRCUITS INTO ONE UNIT!
11

Recap & Agenda

Last week, we:

- Played with transistor circuits on our breadboards
- Learned about NOT, AND, OR, XOR logic gates

Today we will:

- Jump into the ALU!
- Bump up a level of abstraction with our breadboards with integrated circuits
- Make our breadboard into a calculator

Review

Last week we covered a lot of logic gates.

In order to review, can each group draw a truth table for one of the follow logical operations:

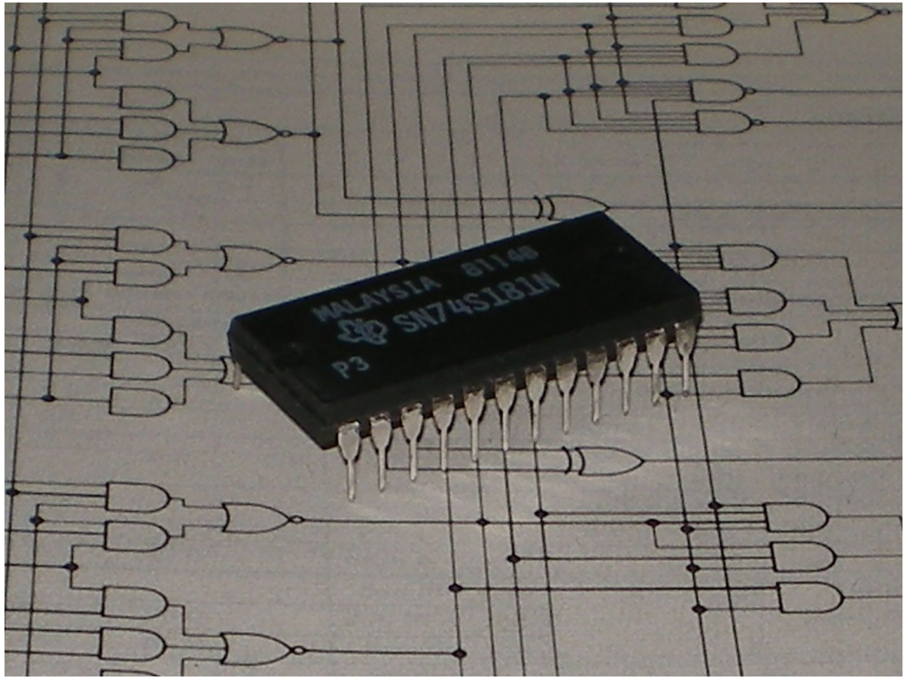
- AND
- OR
- NOT
- XOR

(If there are not enough gates per group, some groups may want to explore gates we didn’t get a chance to cover, like NAND, or NOR gates)

The ALU

The ALU stands for arithmetic and logic unit. It is universal to all computers and is responsible for the bread and butter of computing!

On the right is the 74181, the first integrated circuit to function as an ALU

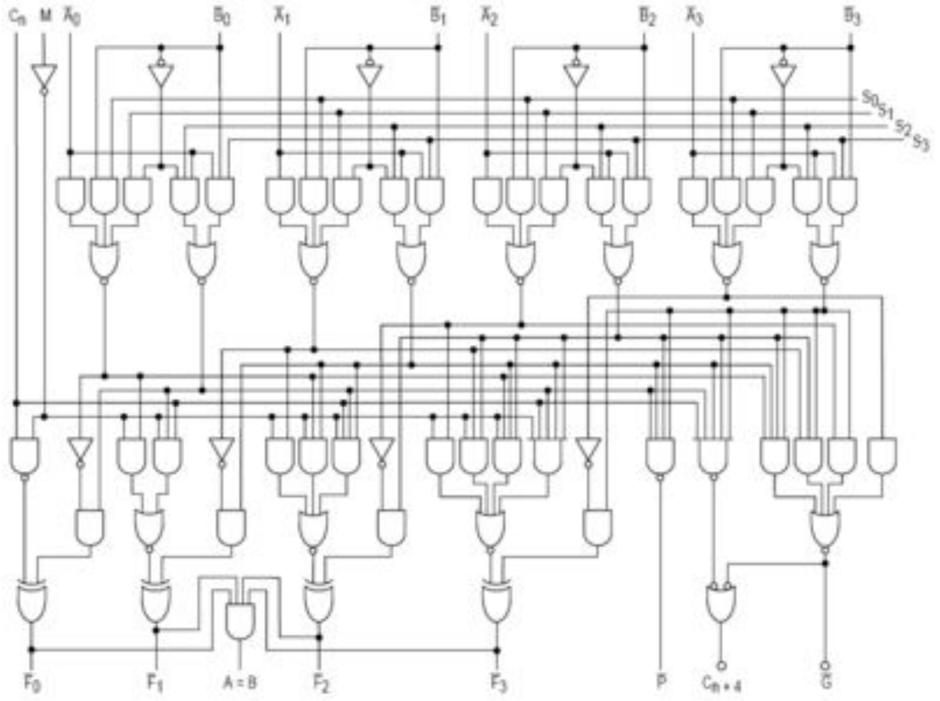


What is does

The ALU can do a bunch of different commands, depending on what the processor tells it to do.

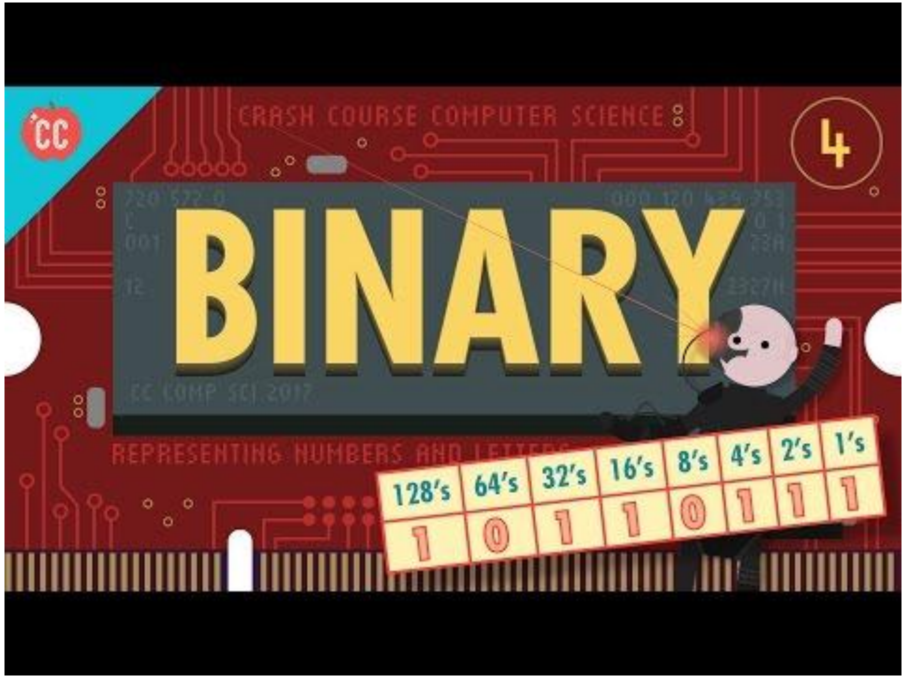
It does calculations, like adding and subtracting, as well as other nifty features, all with the use of logic gates we covered earlier.

Let’s make the A part of an ALU !



But first, return to binary

If we want to make a calculator, we're going to need to understand more concretely how computers store numbers.


CC
CRASH COURSE COMPUTER SCIENCE
4
BINARY
REPRESENTING NUMBERS AND LETTERS
128's
64's
32's
16's
8's
4's
2's
1's
1
0
1
1
0
1
1
1

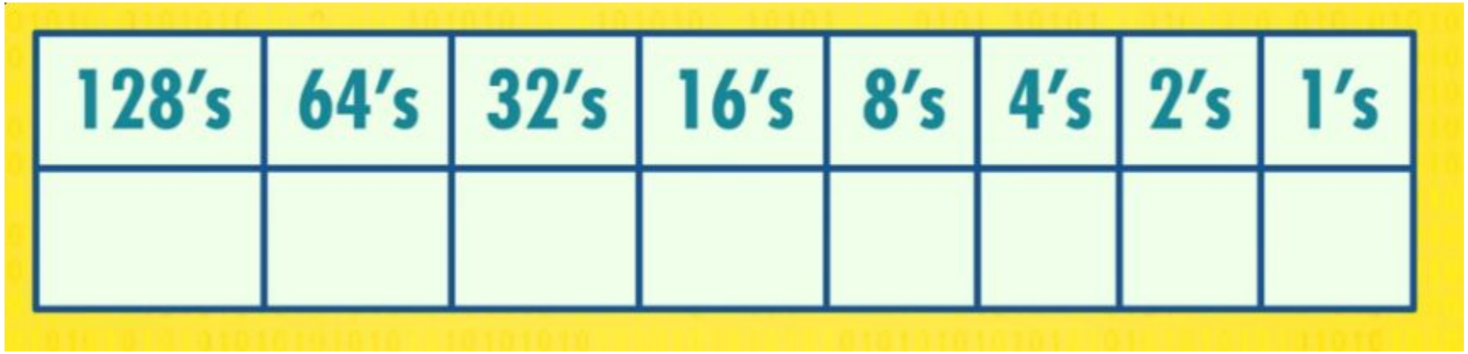
Count like a computer

Challenge approaches! Convert the following numbers into base 10:

00010010   10101010   00110011   10110011

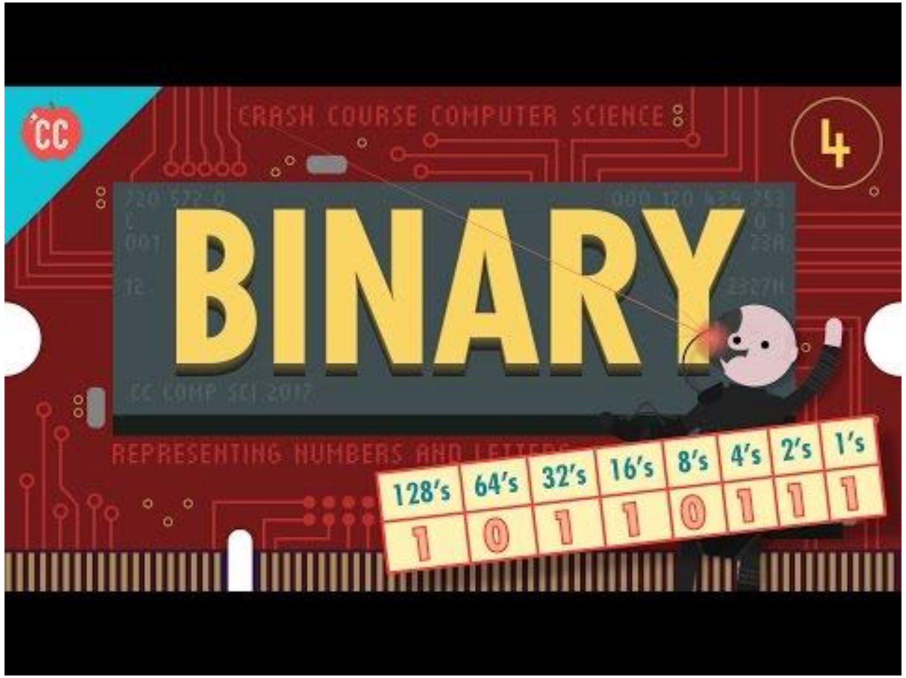
First group to complete the calculations gets candy!!

This image might come in handy



128's	64's	32's	16's	8's	4's	2's	1's

More on bits, bytes, and nibbles



Add like a computer

In order to make a calculator, we’re going to need to know how to add numbers together. Let’s start with something familiar, like adding 2 base 10 numbers together.

Adding in binary is even easier! Let’s run another example.

Think like a computer

Now we know that it’s possible to add numbers in binary, let’s implement an adder on our breadboards that adds 2 bits together!

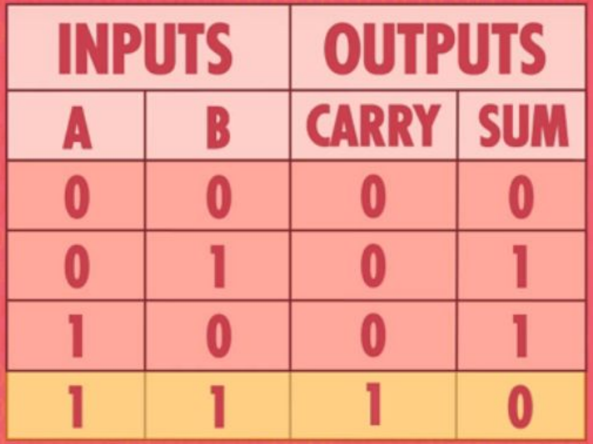
To do this, we’re going to need to run through every possible combination of inputs to construct an idea as to how we’ll implement this with logic gates.

Which logic gate is closest to our combinations?

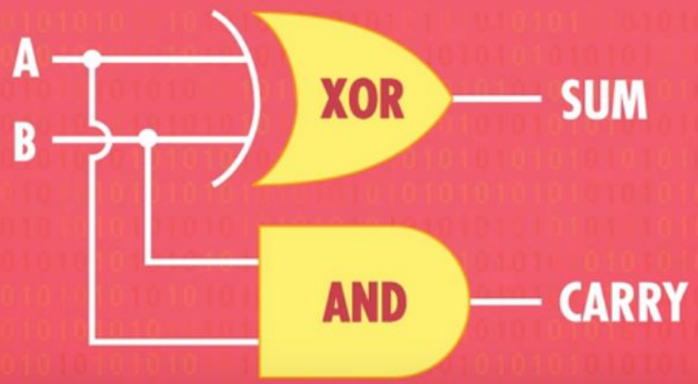
Carrying on

XOR gate nearly matches our input combinations, but it’s missing something. What gate, when both values are true, returns true?

Add the AND gate to our adder for the carry, which also means we need to add an output field to our adder’s truth table called “carry”.



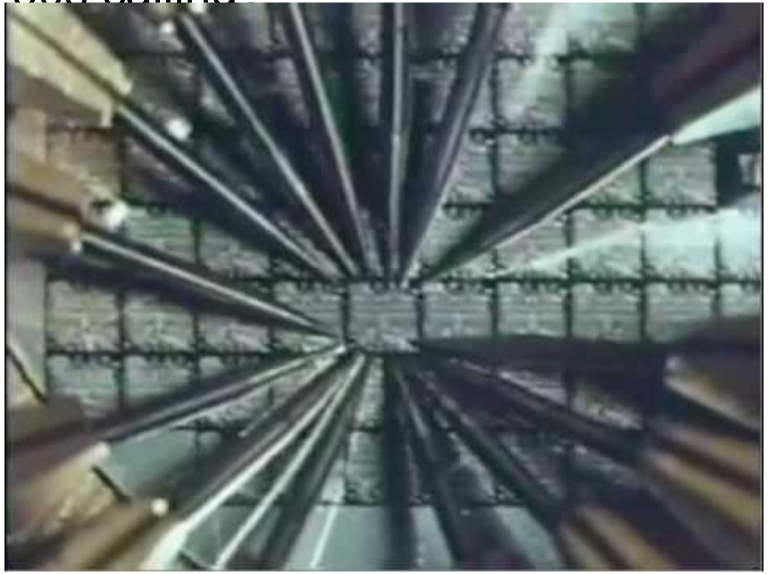
INPUTS		OUTPUTS	
A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



That’s going to require a lot of transistors...

If only there was an abstraction of these transistor based logic gates into pretty little chips

What’s that? Is that the 60s calling?



Break!

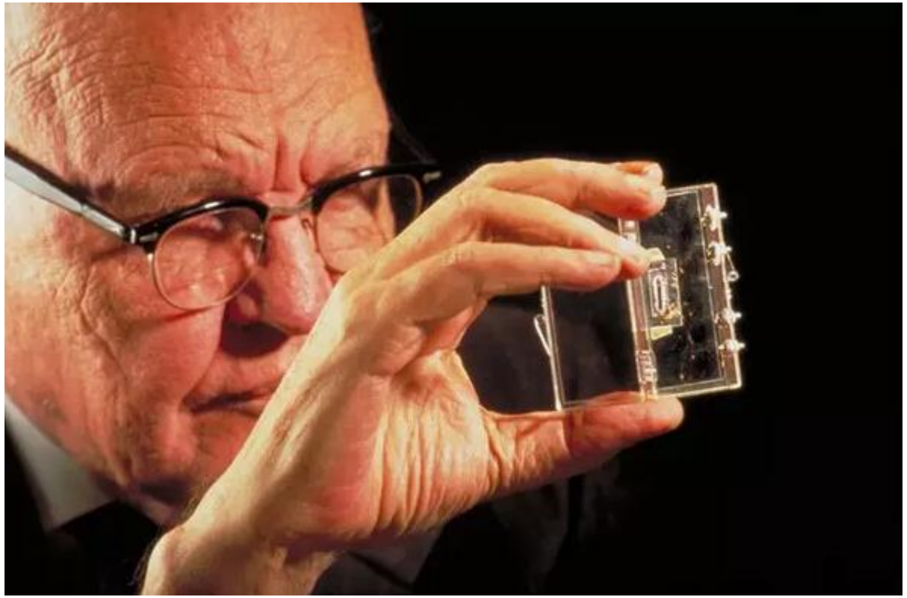
We’ll find out how to use the ICs to make our half adder after a quick pause!

Integrated Circuits (ICs)

Now that we have the jist of transistors, time to bump up a level of abstraction to the *integrated circuit* (IC).

Integrated circuits were first successfully demonstrated by Jack Kilby in 1958. ICs revolutionized electronics, as ICs are more cost effective and more reliable than discrete components like transistors.

Everything you learnt is still there, just packaged up nicely into a single chip



Jack Kilby, auctioning off his first IC (1958) in New York, expected to sell for $2 million.

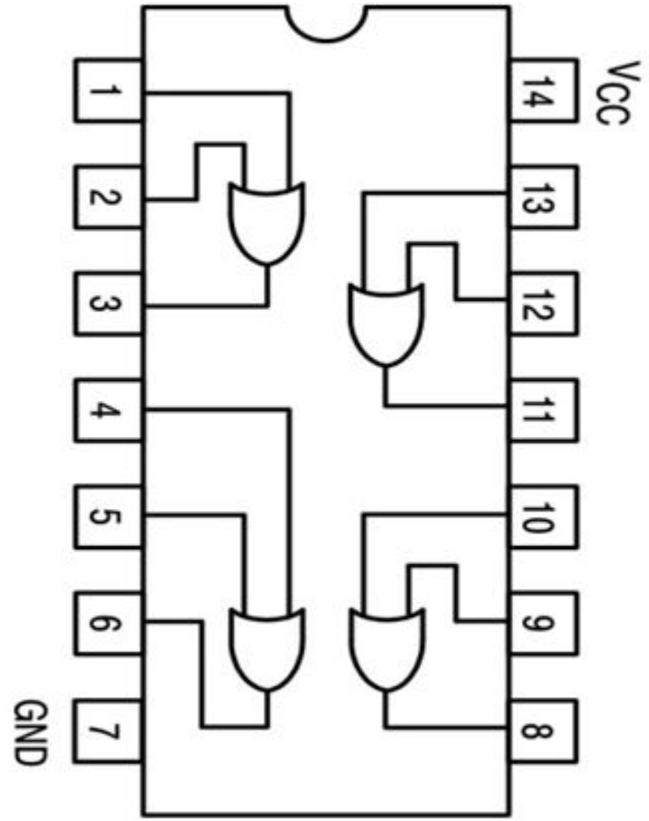
Now let’s use one!

This is a “pinout diagram” for the chips we’ll be using.

The top of the IC has a little divot to help you orient them.

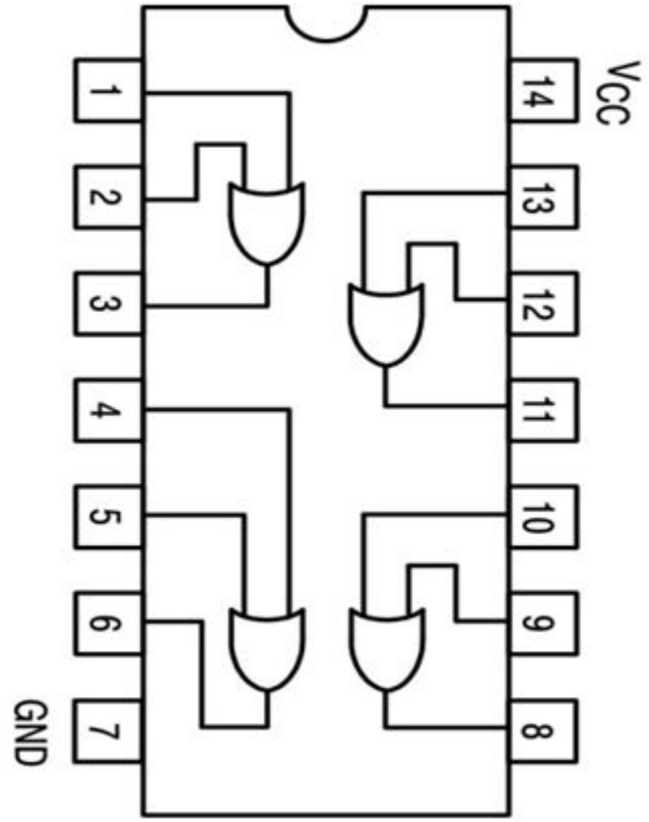
Make sure the VCC is connected to the positive and the GND is connected to the negative.

Test out the AND & OR ICs on your breadboard by observing their output using an LED! Don’t try switches because we don’t know how they work.



Important note about ICs

We need to use a voltage regulator with our ICs to prevent them from blowing up

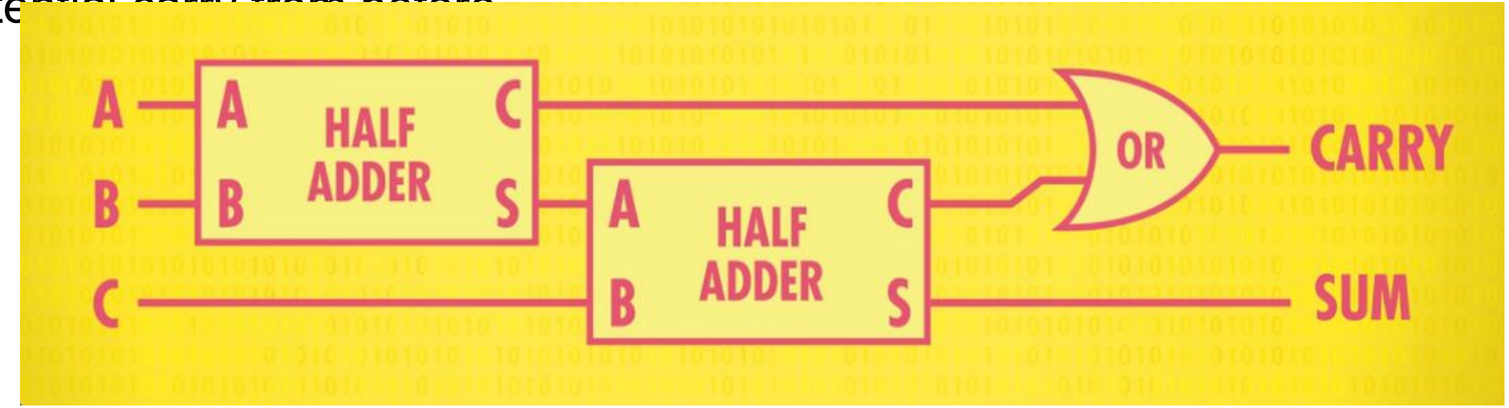


Upgrading to a full adder

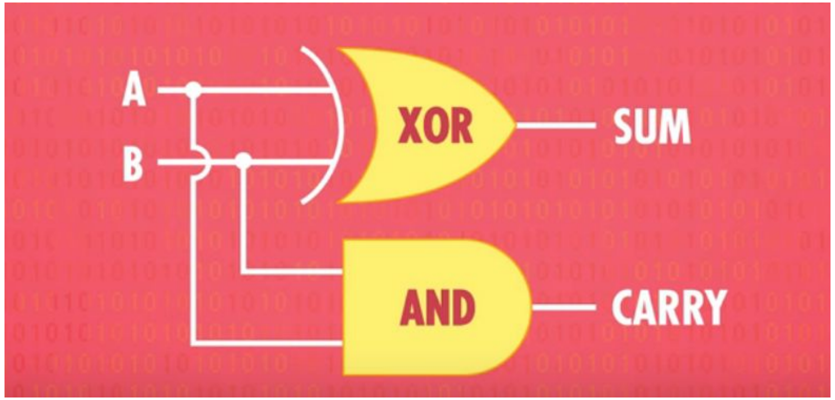
Half adder didn't account for the possibility of a carry coming in from the last addition.

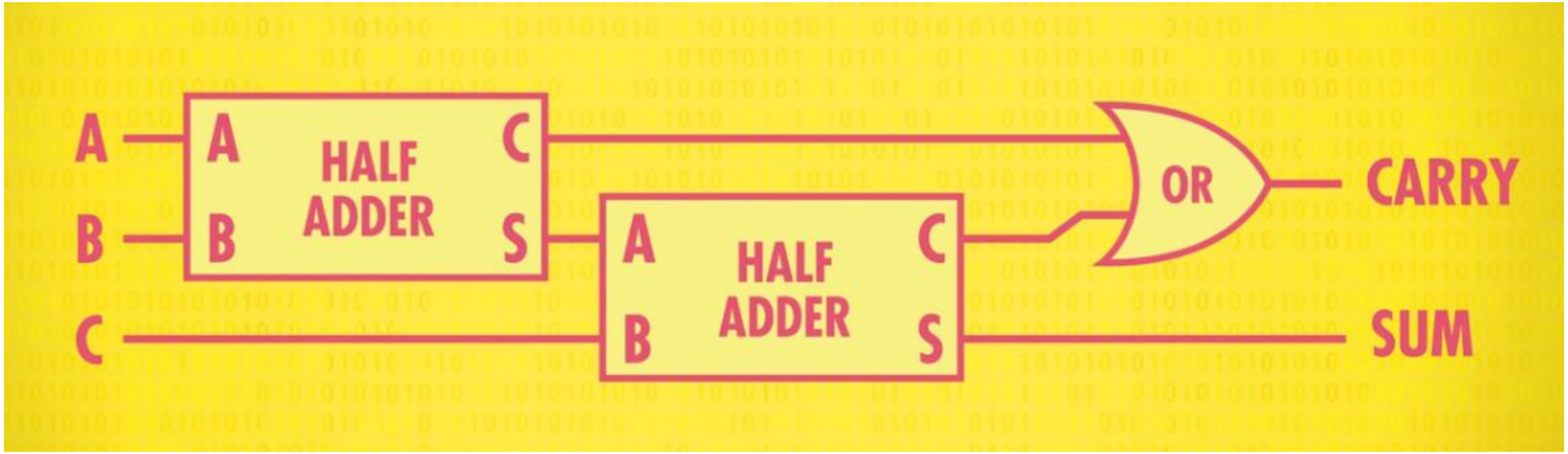
This means we're going to have to add an input field to our adder truth table.

We're also going to add another half adder that will add the sum of A + B to the potential carry from before.

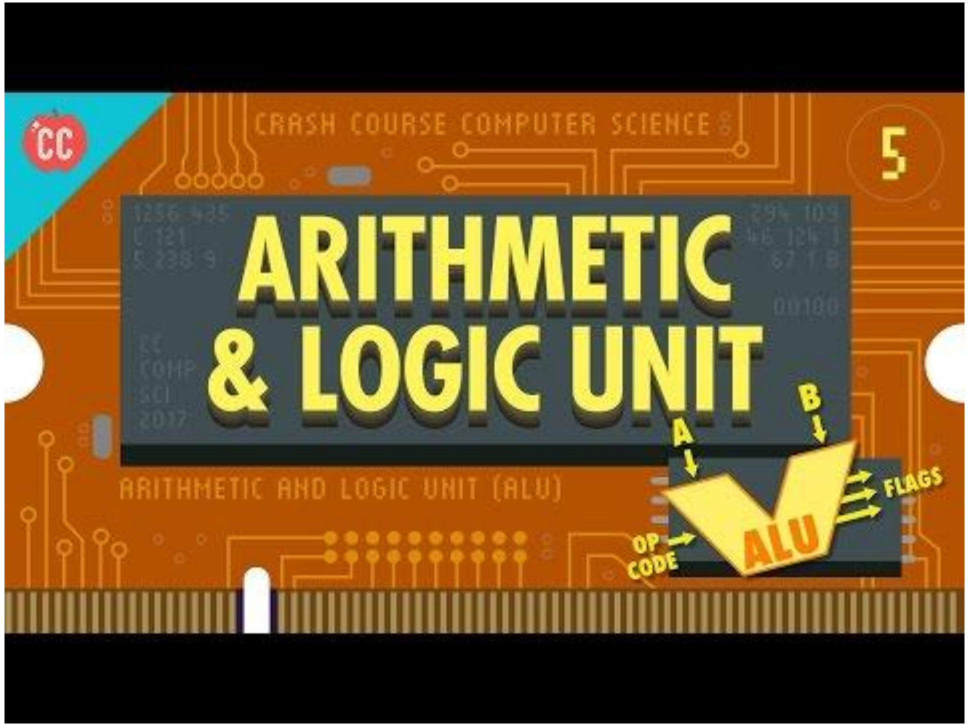


Back to the breadboard





Making a calculator


CRASH COURSE COMPUTER SCIENCE
5
ARITHMETIC
& LOGIC UNIT
ARITHMETIC AND LOGIC UNIT (ALU)
A
B
FLAGS
OP CODE
ALU

More than just calculations!

Like we mentioned earlier, the ALU can do a bunch of other things just using logic gates. Take for example, the logic gate diagram below. It checks to see if an input is all 0, useful for knowing if we can write over at that position.

This check if 0 function sets a flag in the ALU if its 0. ALU has a bunch of flags that are really useful!

ALU challenges!

Now that we have a solid understanding of the ALU, let’s make some! Try and implement any of the following functions using logic gates. For simplicity let’s suppose we’re using a 3 bit ALU:

- Checks if an input number is even
- Subtracts two numbers
-

That’s all folks!

Today we learned a lot, specifically:

- What an IC and ALU are
- How ALUs do math and other nifty things

Next week, we’ll discuss:

- Memory