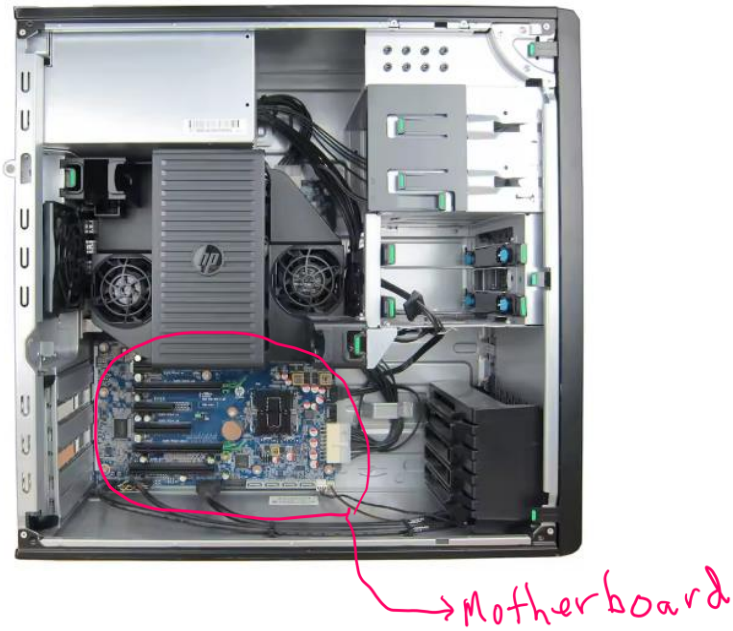
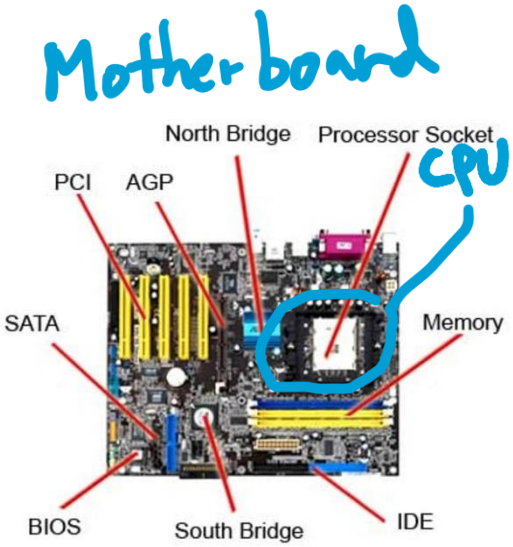
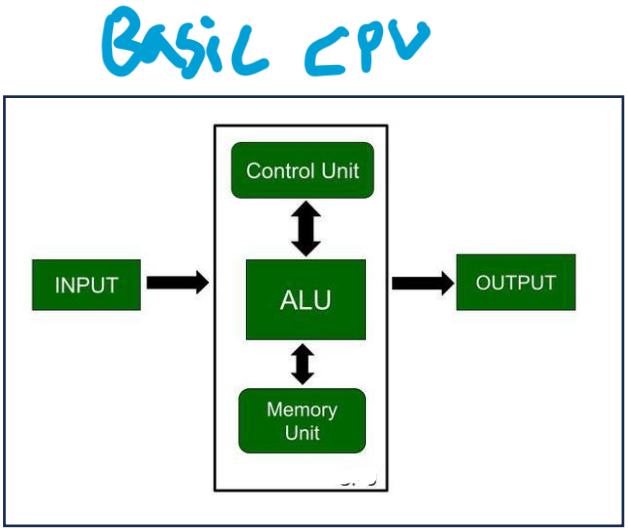
CIS 240: **MICROCOMPUTER ARCHITECTURE & PROGRAMMING**

Lab 2, 3: Building with gates. Hierarchy (symbols)

This week we will design a very important part of a computer: The ALU. The ALU is the part of a computer that does calculations and data manipulations. There are four main blocks of a basic, traditional computer: ALU (does math and data manipulations), the controller (tells the other blocks when and what to do – The boss), memory (to hold data and programs), and the I/O block (that talks with the world off the CPU).







Inside the ALU, there might be adders (circuits that add two binary numbers), multipliers, dividers, subtractors, and other data manipulators. Depending on what your computer does, there might be more functions or less. We haven't talked about floating point numbers yet but an ALU might work (only) on 2's complement numbers or it might only work on floating point numbers. If your computer processes a lot of **image** data, the ALU might have special blocks that do linear algebra efficiently. We talked about 2's complement numbers but there are other ways to represent numbers. Depending on who designed the computer, they might have decided to represent numbers in a different way like sign-magnitude, 1's complement, or DBNS (double-based number systems – A number system I've done some research on).

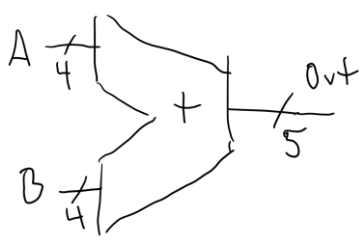
This week we'll do addition and subtraction. If everyone finishes quickly, we'll add a multiplier. There are many styles of adders but the one we will build is called a ripple carry adder (RCA). There are many places you can look up how to build this circuit – **<u>Please don't</u>**. This is practice on designing so please do design.

Products (what to turn in) will be a report that documents your design work. I'd like this to be written so that if, in future years, someone asks you to build a circuit for them in gates and you can't remember how, you can look at this report and use it for help. The first part of this handout is background and then what's due is at the end.

Part 1: Design background

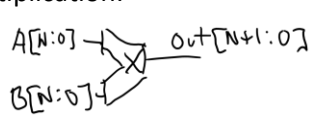
When you design, you should follow the following steps:

1. Make sure you understand the problem. You will eventually be building and adder (though today you will build just a single block). Your final adder (which you'll do on Wednesday) should take in two 4-bit numbers, A, and B and produce the sum of A and B, Out. This is the symbol for an adder:

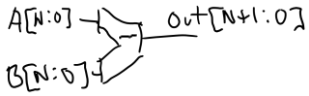


Means that there are N wires in parallel. So "A" has 4-wires (bits) and "B" has 4-wires (bits). We can write this as an array: A[3:0] (4-wires: A[3], A[2], A[1], and A[0]). I array notation, B would be B[3:0] and Out would be Out[4:0]. We'll assume A[3] is "A"'s MSB.

Other arithmetic operations use the same symbol but with the operation changed from the plus sign to whatever operation the block actually does.
Multiplication:



Subtraction:



Division:

Adding in base-2 (binary) is like adding in base-10 but in base-10 you carry when a sum in a column is 10 or greater whereas, in base-2, you carry when you have a sum in a column that is 2 or greater. In other words, if the sum in a column is the base or greater, you carry.

Example from lecture slides:

Binary → Base-2 (**2's comp**)

	Carry	0	1	1	1	1	1		
A		0	0	0	1	1	0	1	1
B	+	0	0	1	1	0	1	1	0
Sum		**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**

Base-10

	1	
	2	**7**
+	**5**	**4**
	8	**1**

The adder that you will build today is going to use a trick. Note that each column does the same thing. Look at the circled column below. In that column (as you do with ALL the other columns), to get the sum, you add the carry-in (0 or 1), a bit from "A" and one bit from "B". To see if there is a carry out to the next column on the left, you see if the sum is 2 or greater. The circled column's carry-in is '1', the bit from "A" is '1', and the bit from "B" is '1'. So the sum is 1+1+1=3. That's greater than 2 so there is a carry. Each carry subtracts the base so… The sum is 3 and when you carry a '1', that subtracts the base, '2' from '3'. So the **sum** becomes 3-2=1 and there is a **carry** of '1'.

Binary → Base-2 (**2's comp**)

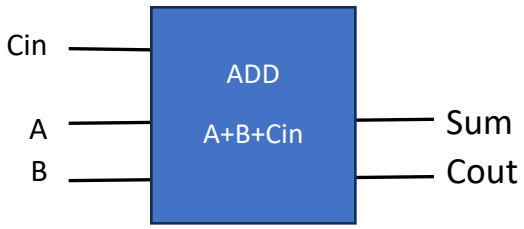
	Carry	0	1	1	1	1	1		
A		0	0	0	1	1	0	1	1
B	+	0	0	1	1	0	1	1	0
Sum		**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**

Base-10

	1	
	2	**7**
+	**5**	**4**
	8	**1**

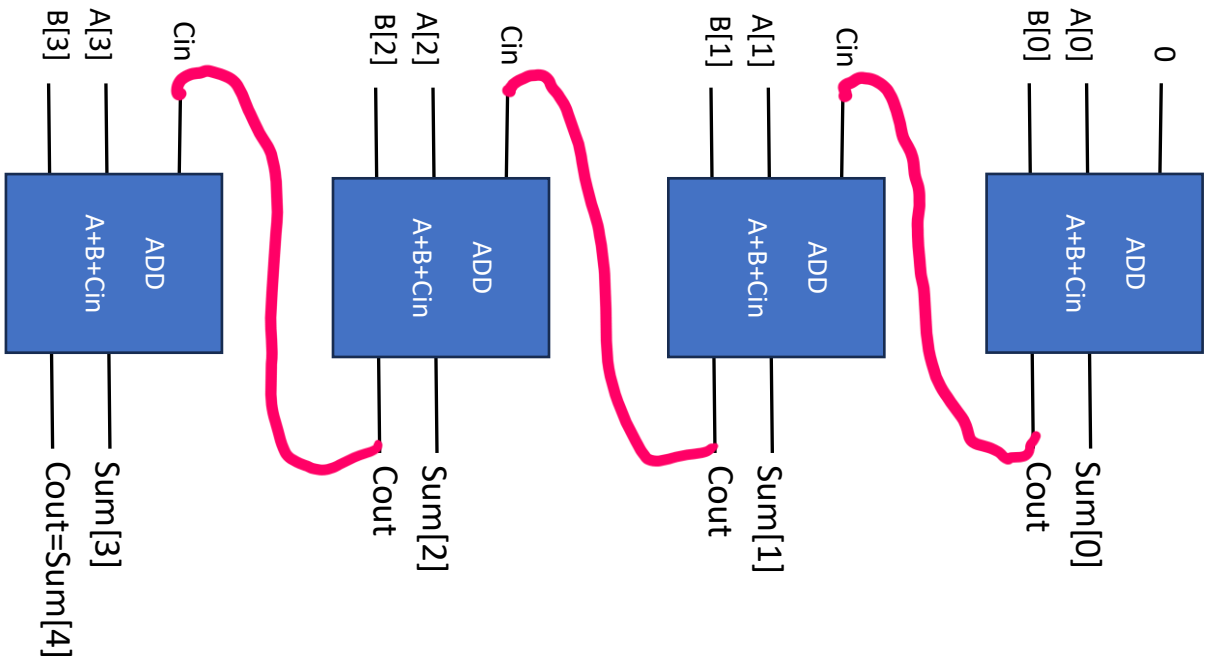
All the columns follow that same pattern. You add the carry-in bit from the column on your right, one bit from "A" and one bit from "B" to find a single sum bit and a single carry-out bit. So… if you build a circuit that adds three bits and calculates a sum bit and a carry-out bit, you can cascade or chain a copy of that circuit over and over again.

Let's say you have the block that adds three bits and gives you a sum and a carry bit out.



You could chain or cascade them like this:

(Red is the carry-in wire from the previous block, and Sum[4:0] is the final sum. The carry-in of the first block is always zero so I put a zero there).



Today we'll build the LSB block. Remember that the SUM bit is the sum of the "A" bit, the "B" bit, and the carry-in. A[0]+B[0]+Cin=SUM[0]. But for the LSB, Cin always equals zero so you can make that block a little bit simpler. You can ignore the Cin input and then your new equation equals: A[0]+B[0]=SUM[0]. To calculate the carry, you get to ignore the carry-in also. To see if there is a carry, you just need to see if the "A" bit plus the "B" bit is greater than or equal to zero.

What to do for Monday's lab:

If the reading so far doesn't make sense, please ask me or people around you to clarify.

Here are the steps to go through today (we'll work through this together). Please document all of what you do for your report. This part of the report will be combined with Wednesday's part for a complete report due on Monday 2/3. Here are the steps I want you to include in your report:

i. Design the block: The block we'll build today is the simpler one described in the last paragraph on the last page. The one with a carry-in equal to zero.
   a. Understand the problem – Describe what the block does.
   b. Create a truth table – Create two truth tables. One for SUM and one for CARRY-OUT.
   c. Find the logic for both the SUM and CARRY-OUT – Use SOP to find logic.
ii. Enter the circuits into LTSpice
iii. Make a symbol for the block
iv. Make a new schematic to test the block
v. Test - Simulate the block for all possible inputs. Include screen captures of your simulation.
vi. I'll ask you to submit the design files. So make sure to back up *.asc and *.asy files.

What to do for Wednesday's lab:

If the reading so far doesn't make sense, please ask me or people around you to clarify.

This together with Monday's work will make up the content of the report.

i. Today (Wednesday) is take 5 or 10 minutes with a nearby person and make sure both of you remember everything you did Monday. If you are getting errors, please tell me.

Here are the steps I want you to include in your report: (They are the same as Monday but the block is slightly different). Wednesday 1/29 adds to your Monday circuit:

ii. The circuit you built on Monday finds the sum of two bits. You will add another circuit to your Monday schematic that finds the carry-out of adding A and B. That means:
   a. Understand the problem – Describe what the block does.
   b. Create a truth table for CARRY-OUT.
   c. Find the logic for CARRY-OUT – Use SOP to find logic.
iii. Enter the circuits into LTSpice
iv. Make a symbol for the block with both sum and carry-out
v. Make a new schematic to test the block
vi. Test - Simulate the block for all possible inputs. Include a screen captures of your simulation.

Lab for Monday 2/3: Building the more complicated block with A, B, and Carry-in and Sum and Carry-out.

i. Design the block that has a carry-in and a carry-out. That means:
   d. Understand the problem – Describe what the block does.
   e. Create a truth table – Create two truth tables. One for SUM and one for CARRY-OUT. After you create this truth table, get checked off by me!!!!
   f. Find the logic for both the SUM and CARRY-OUT – Use SOP **KMaps** to find logic.

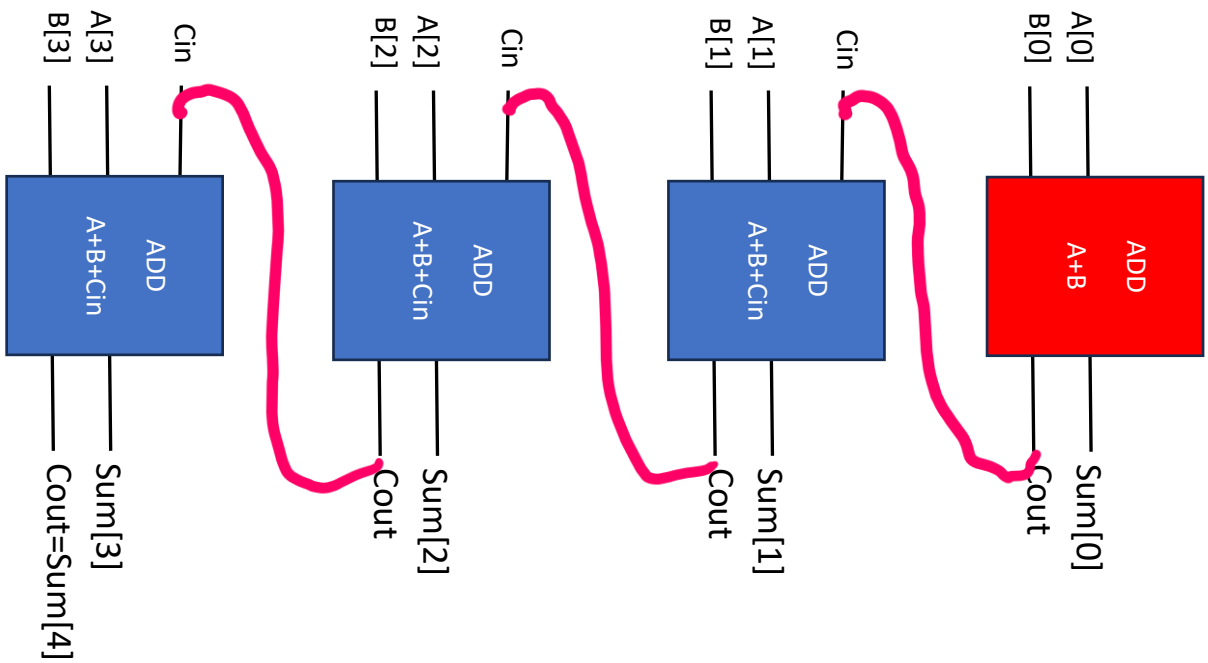
ii. Enter the circuits into LTSpice

iii. Make a symbol for the block

iv. Make a new schematic to test the block

v. Test - Simulate the block for all possible inputs. Include a screen captures of your simulation.

vi. Make another new schematic and chain together the following circuit:



The red block is the one you built Monday 1/27 and Wednesday 1/29.

vii. I'll ask you to submit the design files. So make sure to back up *.asc and *.asy files.

Again, I'd like the report you write to be useful to you in the future so please write it in a way that would be useful to you as a reference. There is no minimum or maximum page number. Whatever you need to document your work.

Next week you will add to your report when you add a subtractor and what is called the "register file" to your ALU.