MONTANA
STATE UNIVERSITY

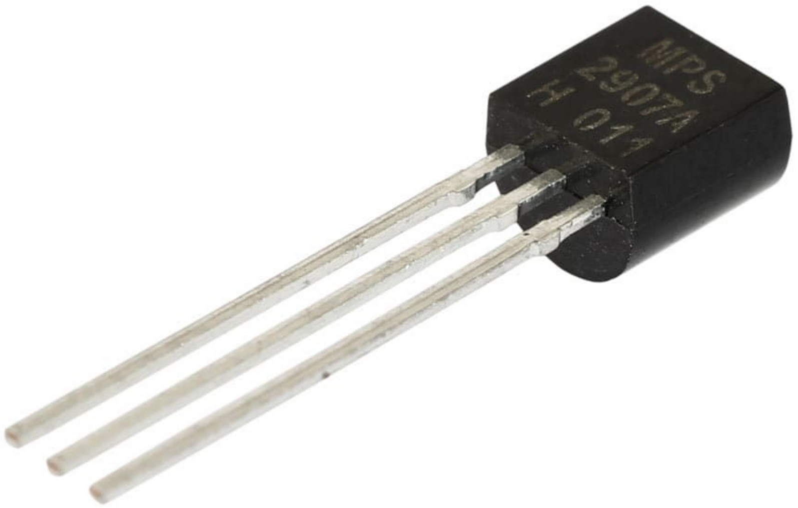
Introduction To Binary Computing

•••

Register & Adder Implementations

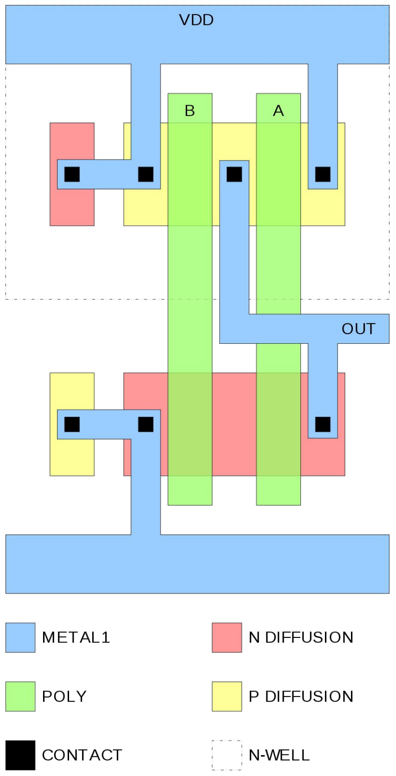
Last Lecture

- In the last lecture we discussed transistors and how transistors can be used to implement logical gates
- We discussed the importance of the NAND gate and a few different implementations of this gate



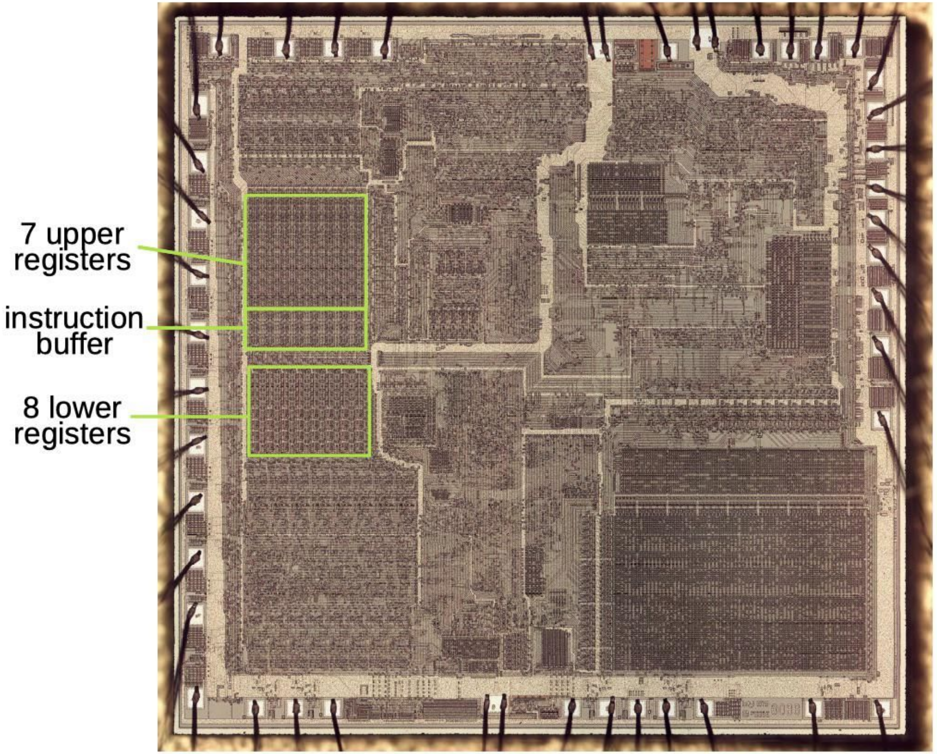
Last Lecture

- In the last lecture we discussed transistors and how transistors can be used to implement logical gates
- We discussed the importance of the NAND gate and a few different implementations of this gate



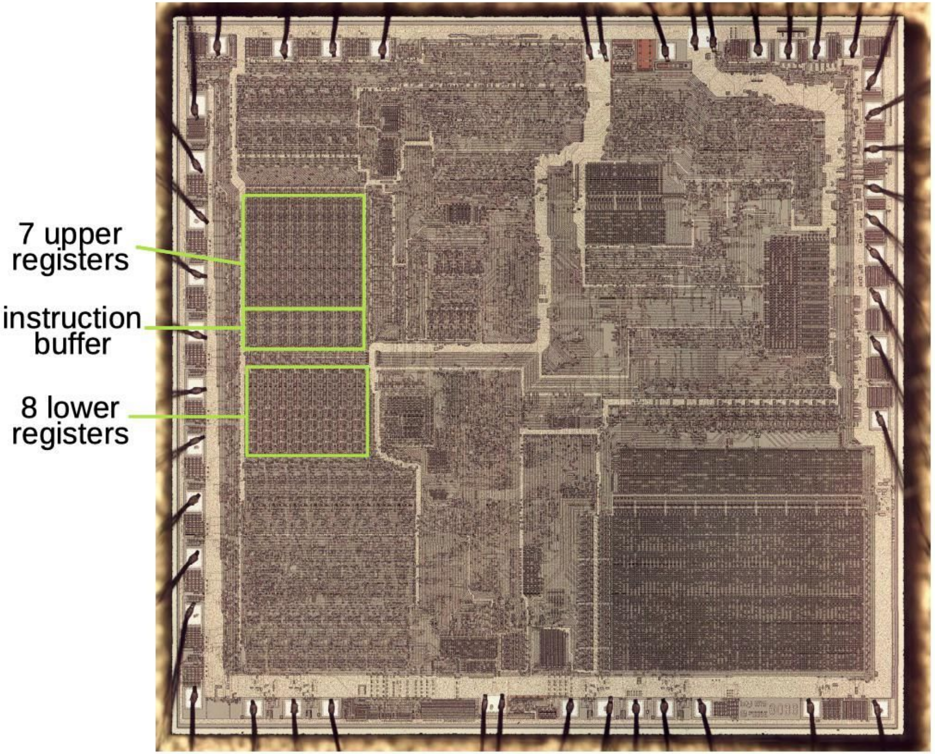
This Lecture

- In this lecture we will look at
  - The implementation of registers in the 8086 chip
  - The implementation of an adder (half and full) circuit



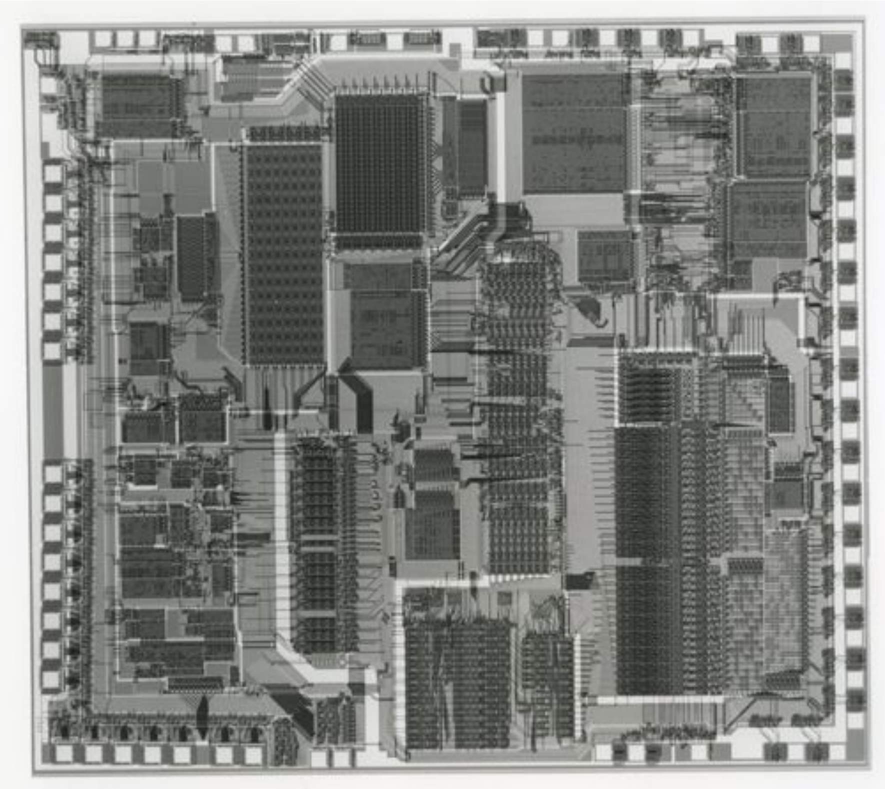
The 8086 chip

- Produced by intel from 1978 to 1998
- 16 bit chip
    - Supported 20 bit addressing
- First in a series of x86 chips which, unexpectedly, would take over the world



The 8086 chip

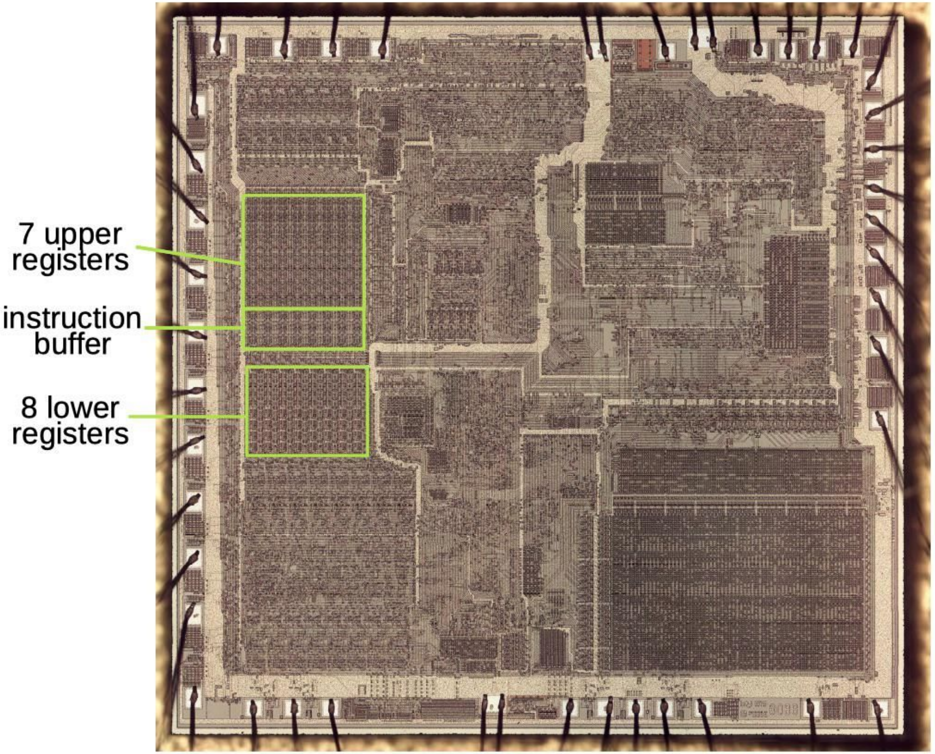
- Chip was a reaction to the delay of the iAPX 432 chip
- iAPX 432 was an interesting chip
  - No user-facing registers
  - Stack machine
  - Hardware support for garbage collection, object oriented programming



The 8086 chip

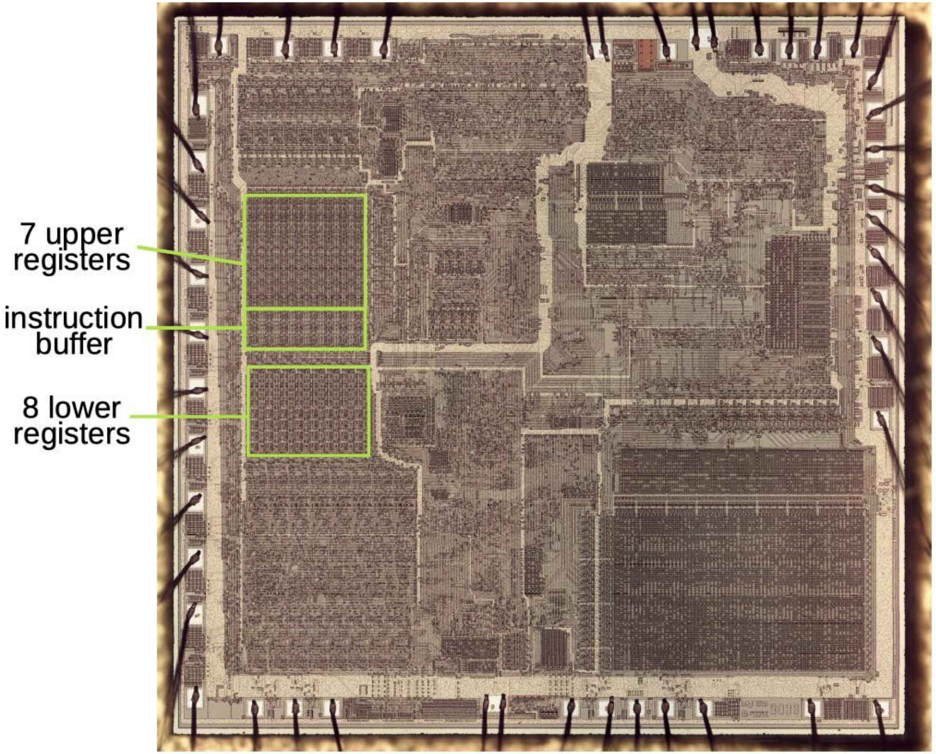
- A wonderful tear-down of an original 8086 chip:

  http://www.righto.com/2020/07/the-intel-8086-processors-registers.html



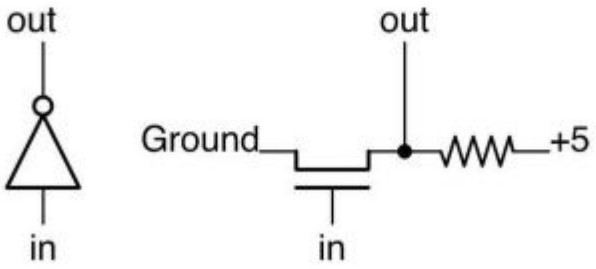
8086 Registers

- Registers, as we know, are very fast memory stores located close to the CPU
- CPUs use registers for storing and mutating data
- As with all binary systems, data is stored in 1s and 0s



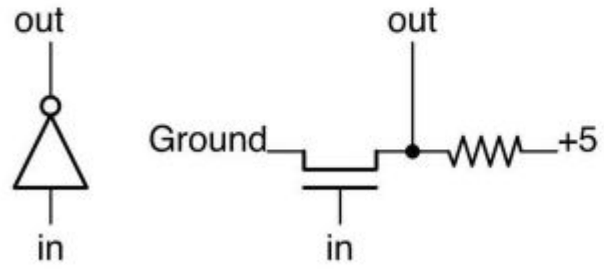
8086 Registers

- NOT gate implementation
- The in wire, when activated, opens a channel to ground, causing current to flow to ground
- When not activated, the channel is blocked, so current flows to out



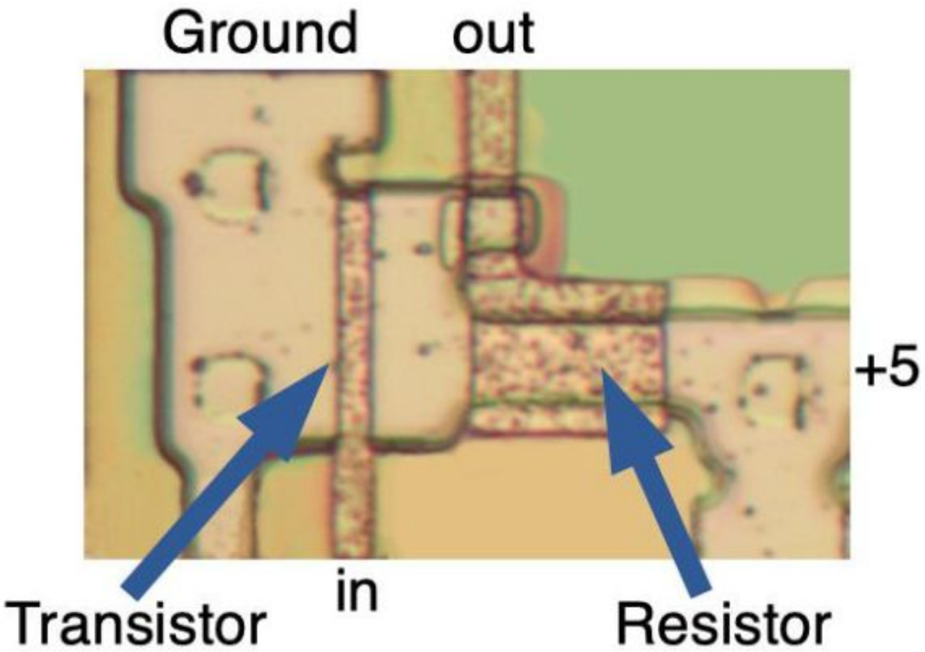
8086 Registers

- You can see how this *inverts* the signal on in



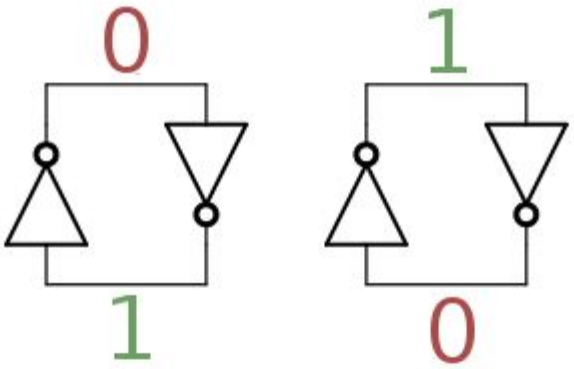
8086 Registers

- Physical layout of a NOT gate
  - Ground and +5 carry electric current
  - Current on IN opens the resistor, causing current to flow to ground, rather than out



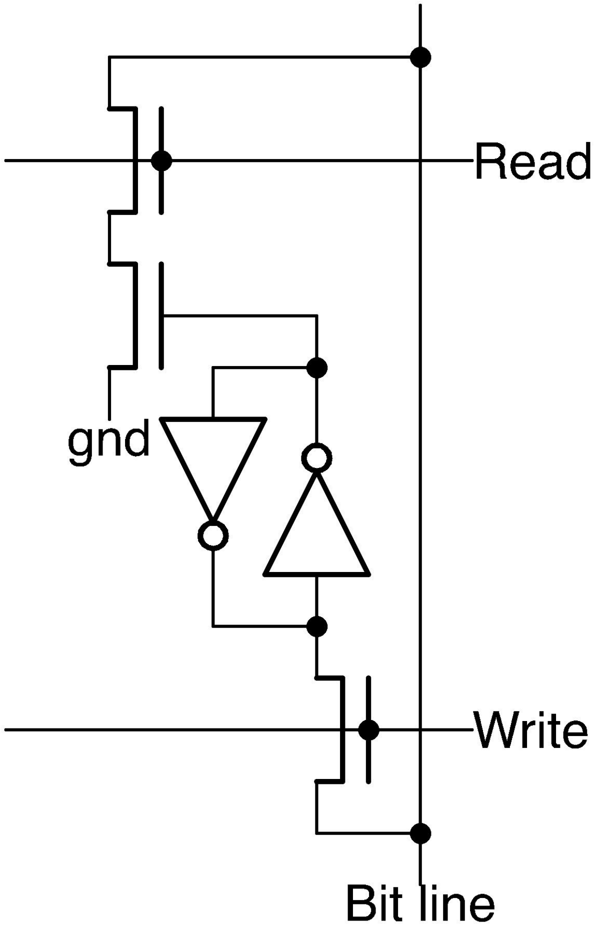
Implementing A Bit

- How can we implement a stable bit value using NOT gates?
- Chain them!
- If top is 0, it will stay 0 as it goes through the bottom gate
- And vice versa



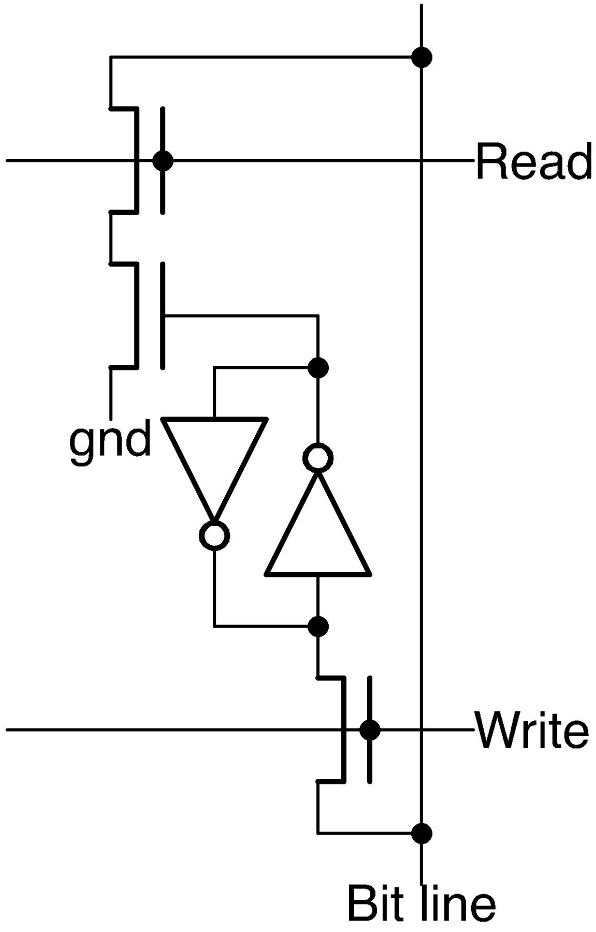
Implementing A Bit

- Adding reading and writing to the bit value
    - A read gate
    - A write gate
- When read gate is open, the the value is written to the bit line
- When write gate is open, the value of the bit line is written to the register



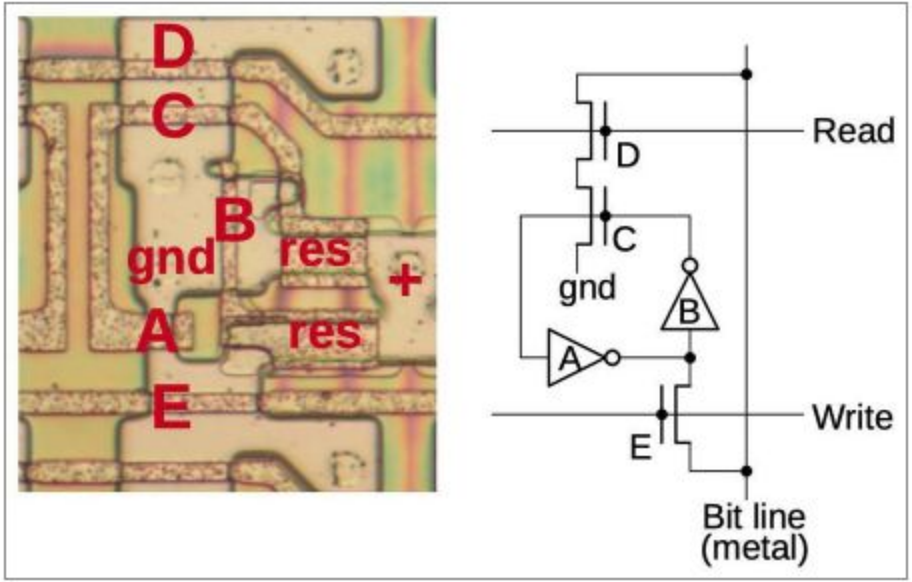
Implementing A Bit

- Note that the read gate is reading the inverted value of the bit and inverting it again



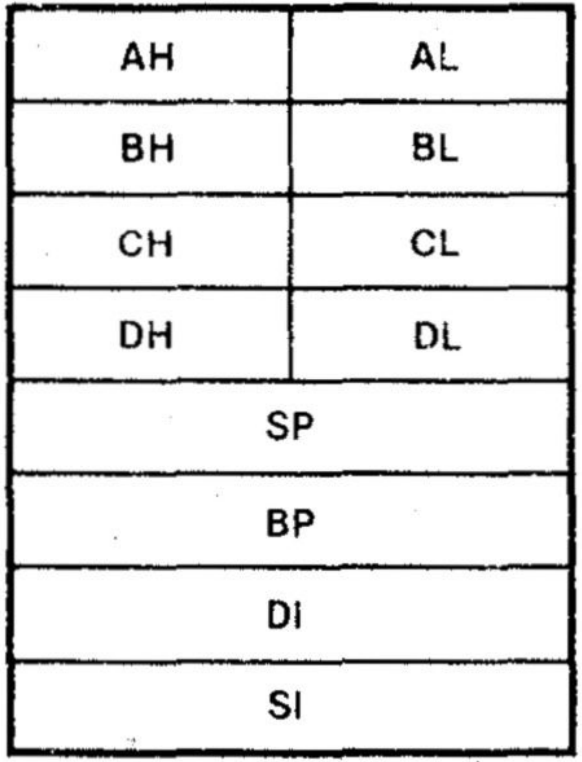
Implementing A Bit

- Physical layout of this register bit
- Note that A connection is much smaller than E
    - E will overpower A on write



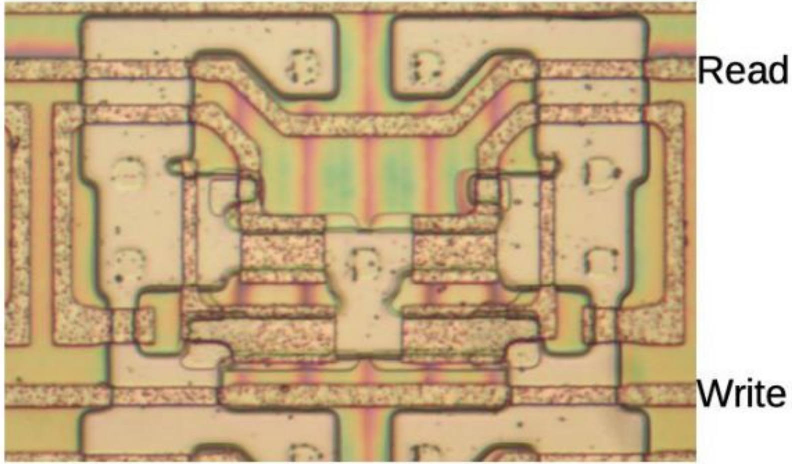
8086 Registers

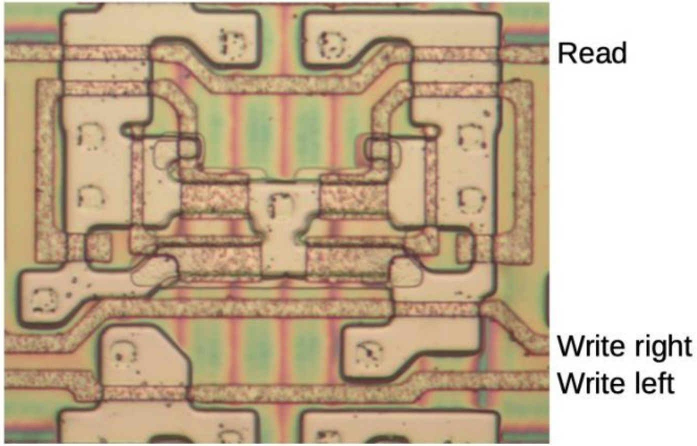
- Like with the current x86 architecture, some registers can be partially addressed
- This legacy is still with us today
    - AH, AL are ancestors of eax and rax



8086 Registers

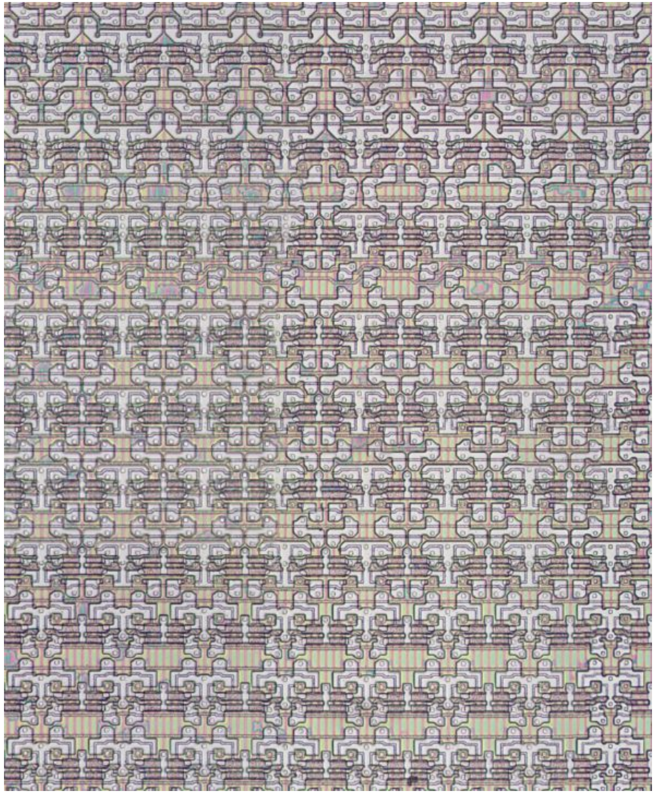
- Implementing this requires more wiring
- More “ports” to the register
- There are even more complex registers available
  - We are not going to go into the details





8086 Registers

- The register file: a collection of these register implementations, all wired together so that certain bit patterns read and write from certain register positions
- Pretty, isn’t it?



Adder Circuitry

- We have covered how a bit is stored in a register
- Let's now look at how to add two bits together
- What does binary data look like?

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Adder Circuitry

- 0 + 0 is… 0
- 1 + 0 is… 1
- 0 + 1 is… 1
- 1 + 1 is… we’ll come back to that

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Adder Circuitry

- Consider the result column to the right
- What logical operator is that?
  - Let's think about it for a bit...

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

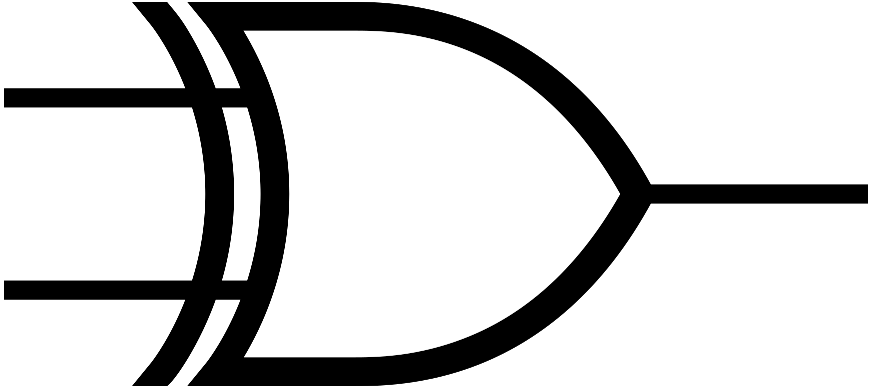
Adder Circuitry

- Upon reflection, I hope you can see that this is a logical XOR (exclusive OR)
- 1 if only A or B is 1, 0 otherwise

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Adder Circuitry

- Here is the symbol for XOR



Adder Circuitry

- OK, so the Carry Bit
  - Just like with decimal math, we must *carry* overflow in binary math
  - Simpler since there are only two possible values: 1 & 0
  - In the case of 1 + 1, we have a carry value to the 2’s place
  - 1 + 1 = 10
    - There are 10 types of people in the world...

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Adder Circuitry

- What is the logical operator expressed by the Carry column?

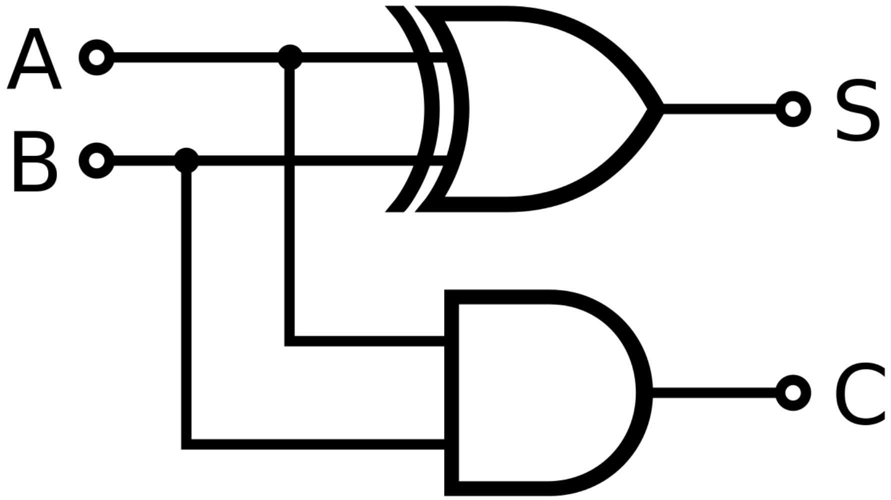
Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Adder Circuitry

- Again, I hope upon reflection it is apparent that this is a logical AND operator
- This is the symbol for a logical AND

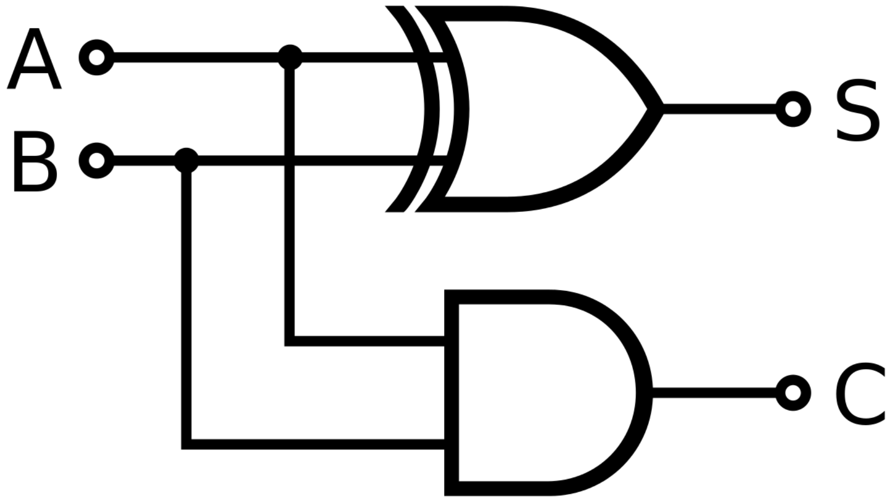
Adder Circuitry

- A full circuit for an adder would look like this
- An XOR for the addition and an AND for the carry bit



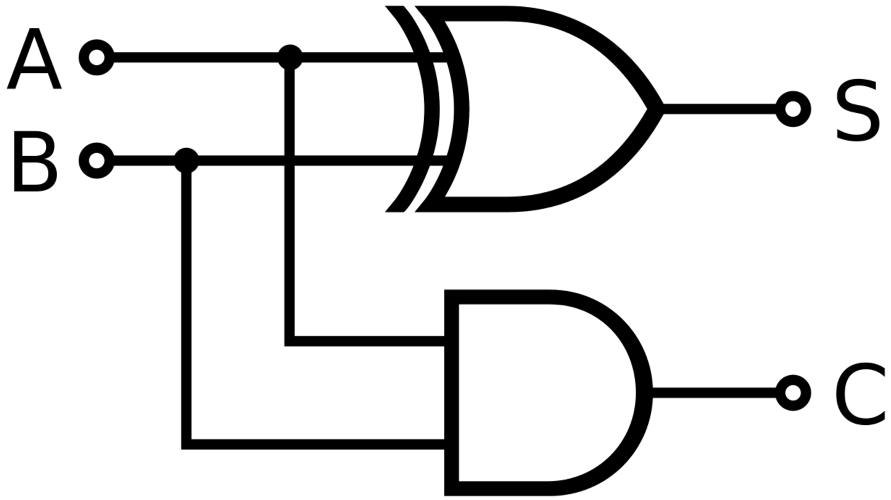
Adder Circuitry

- This is called a *Half Adder*
- Why?



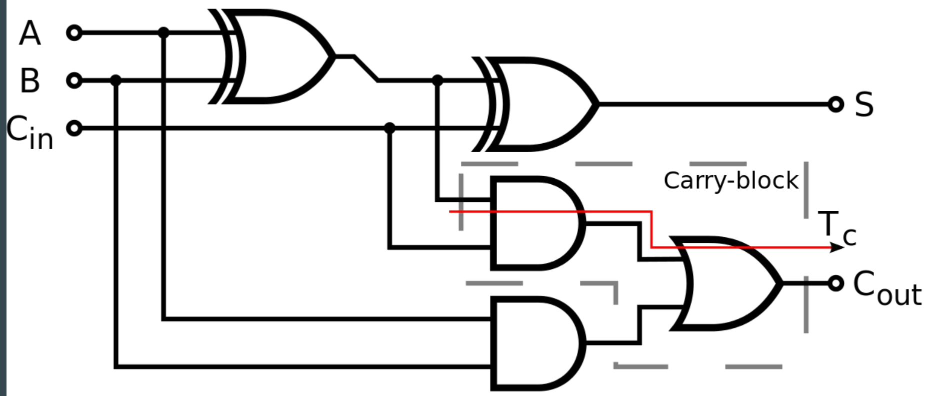
Adder Circuitry

- This is called a *Half Adder*
- Why?
  - Because it does not have an *input* for the carry bit from another binary addition



Adder Circuitry

- A full adder has a C_{in} as well as C_{out}
- C_{out} is true IF
  - A, B and C_{in} are 1
  - A and C_{in} are 1
  - B and C_{in} are 1
- This is the logical layout of a full-adder



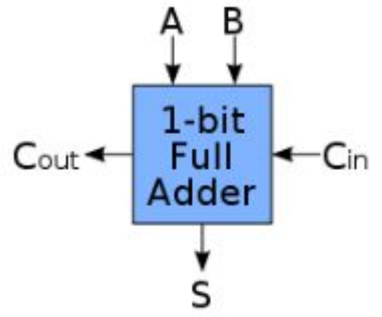
Adder Circuitry

- The full adder truth table

Inputs			Outputs	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

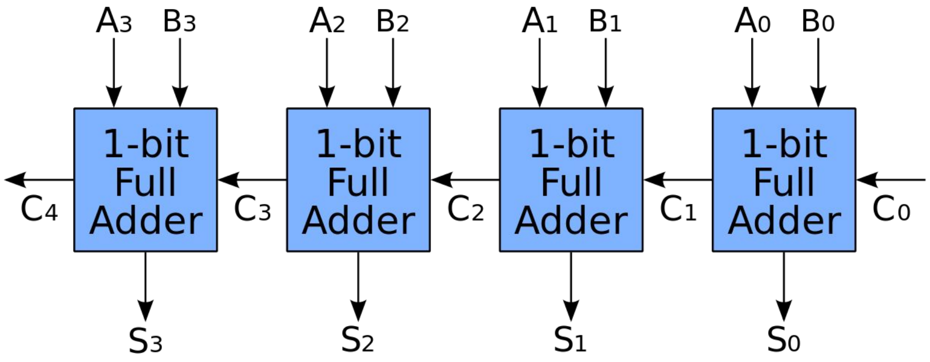
Adder Circuitry

- Schematic symbol for a 1 bit full adder



Adder Circuitry

- Here we see a four bit adder with a “ripple carry”
- The carry bit ripples through the gates from right to left
- This is slow and can be optimized with various additional wiring
  - Carry-lookahead



Adder Video

- A great video on how adders work, with good animations, by In One Lesson:

  https://www.youtube.com/watch?v=VBDoT8o4q00

- Re-explains transistors and shows how binary math works with adders
  - A bit of a slow spot around the 3:30 mark, but excellent otherwise
- Please watch this video!

Storing & Manipulating Binary Data

- We took a look at how the 8086 chip stores bits in a register
  - Both logically as well as physically!
- We took a look at how to do logical addition of unsigned binary data
  - Half-adder
  - Full-adder
  - Ripple carry
- *REMEMBER: IT'S JUST LIGHTNING TRAPPED IN ROCKS*

MONTANA
STATE UNIVERSITY