

Changelog:

I have changed the template for this lecture note as some of the figures were not visible with the light background (white on white is not visible 🤦‍♂)

I have also added slide numbers for your reference

Topics you need to know from this lecture:

- Everything in the *Building blocks of a computer* (from Slide 8 to 22): Transistor, Basic Gates (AND, OR, NOT) and their Truth Table, Combinational Circuits (designing circuits from boolean expression, see the example of XOR)
- Everything in *Abstraction in Hardware* (from Slide 23 to 27): What is a binary and decimal number system, Conversion from Binary to Decimal and vice versa.
- Everything in *Arithmetic Logic Unit* (from slides 35 to 39): What is an ALU, what number and purpose of inputs and outputs (what is an opcode and status)
- Everything in *More Abstraction CPU* (from slide 40 to 49): Structure of the CPU (Control Unit, ALU, Registers and RAM), Instruction Sets, Given a sample instruction set writing a very simple machine code (see example on Slide 47), Moore's Law (you need to know what it is)

Topics in this lecture only for your reference (non-graded), you do not need to know this for your exam:

- Everything in “Creating a calculator capable of adding two numbers using a combinational circuit” (from slides 28 to 34): No Half adders or full adders!
- Everything in Computer Dissection (from Slide 50 to Slide 67): I still expect you to know what a hard disk is and what the difference between GPU and CPU is.

Computing Hardware

BCS1110

Dr. Ashish Sai

 Week 1 - Lecture 2

 bcs1110.ashish.nl

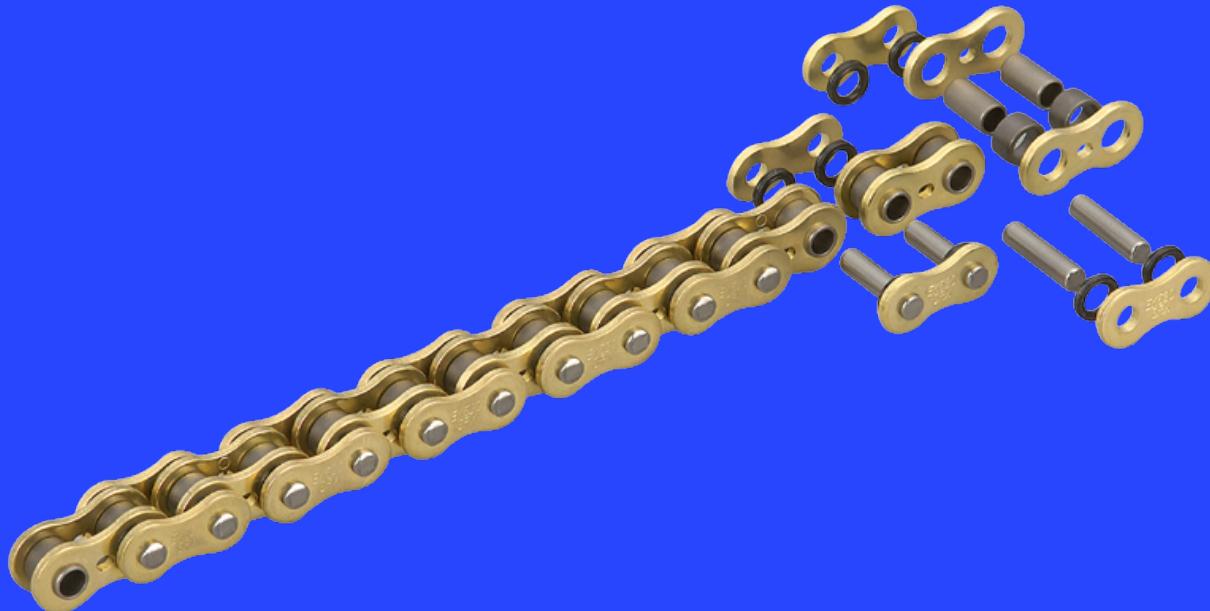
 PHS1 C0.020

Plan for today

- Building blocks of a computer
- Abstraction in Hardware
- Arithmetic Logic Unit
- Computing Hardware Overview



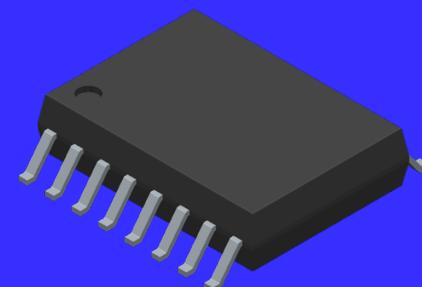
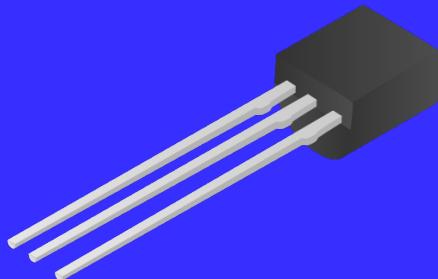
Roller Chain



Building blocks of a computer

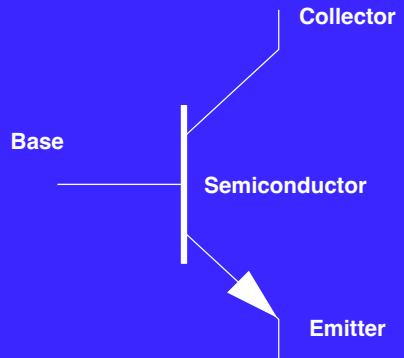
Part 1/4

Computers are constructed using individual **transistors**, which form **circuits** that enable various operations and logic



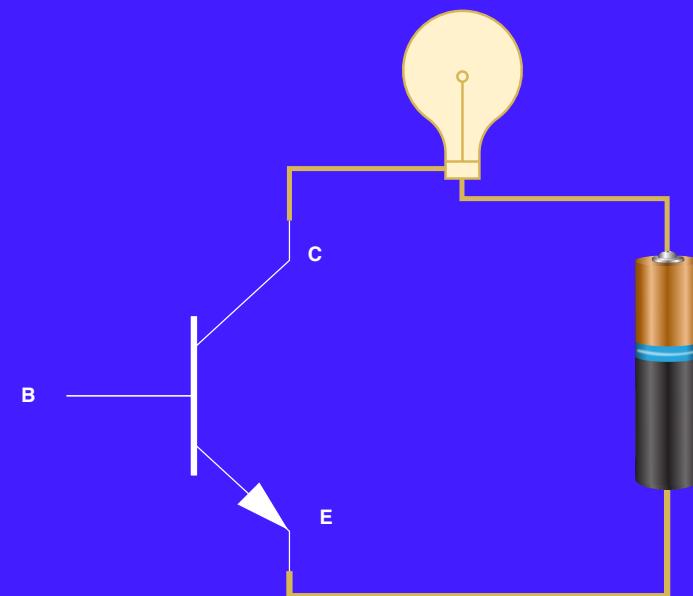
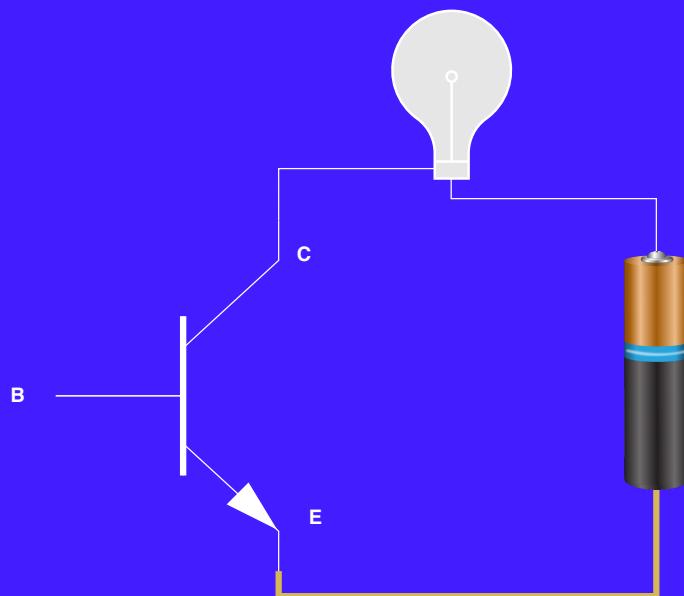
Transistors

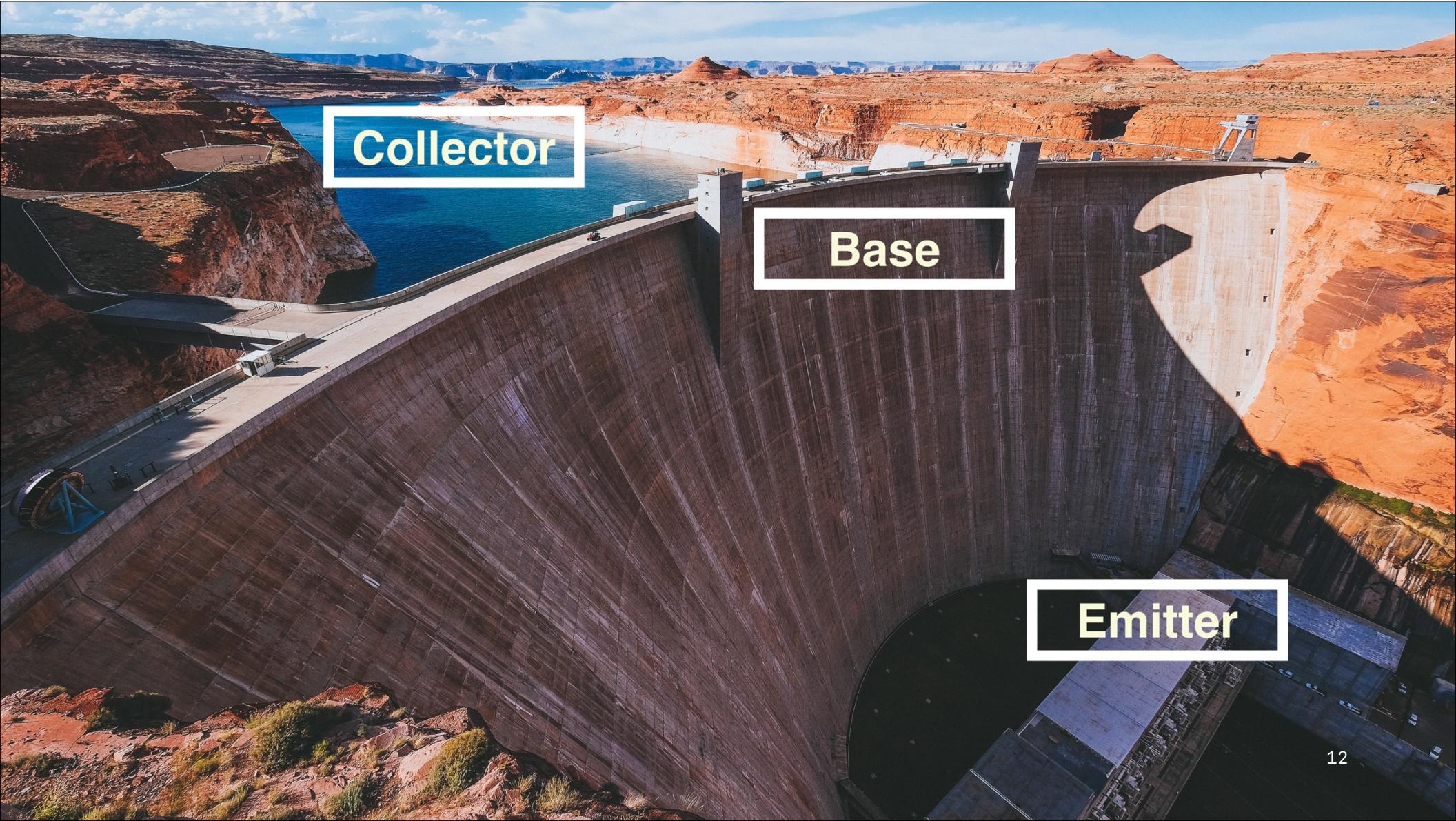
- A transistor is an electronic device made of semiconductor materials¹ that can amplify or switch electronic signals and electrical power
- It consists of three layers (emitter, base, and collector) and can control the flow of current by applying a small input signal



1. Semiconductors are materials that have properties in between conductors (which allow the flow of electricity easily such as metals) and insulators (which block the flow of electricity such as ceramics)

If enough voltage is applied to the base electrode,
current can flow between emitter and collector and
the transistor can like a switch ►II





Collector

Base

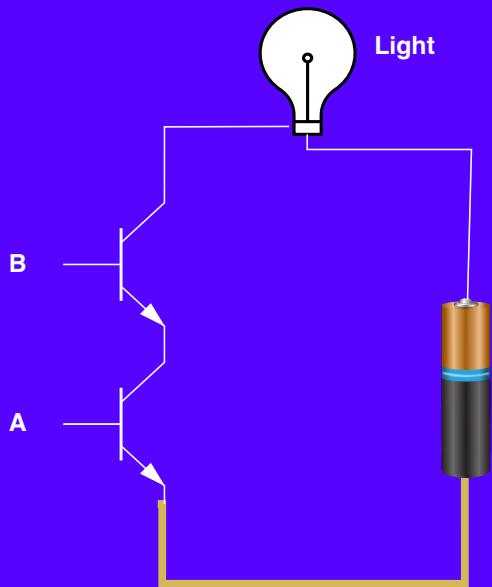
Emitter

iPhone 14

- Has over **16,000,000,000** transistors (switches)
- to count from 1 to 16B would take you about one *thousand and seventeen years!*



Combining Transistors



You can do quite a lot
when you combine these
transistors

Truth Table

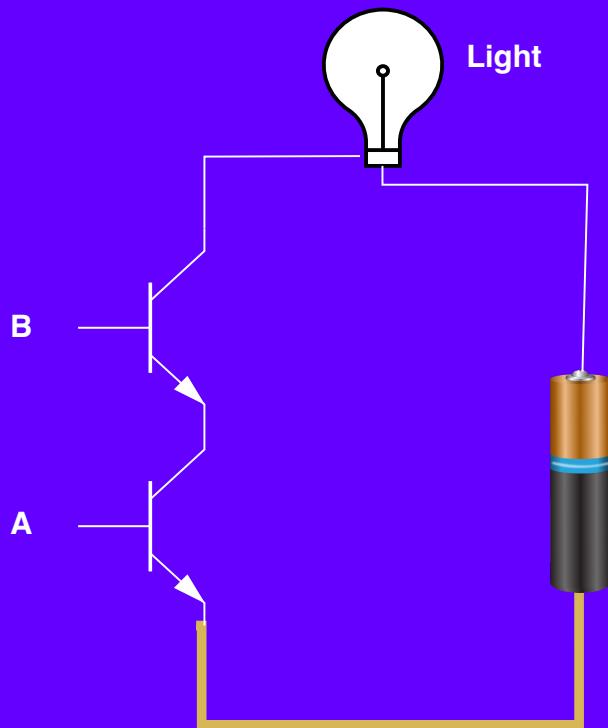
A	B	Light
True	True	True
True	False	False
False	True	False
False	False	False

Current and Bits

Current is the flow of electric charge through a conductor, like a wire, measured in units of amperes (A)

- When current flows: 1 
- When current is not flowing: 0 

AND Gate¹



Truth Table

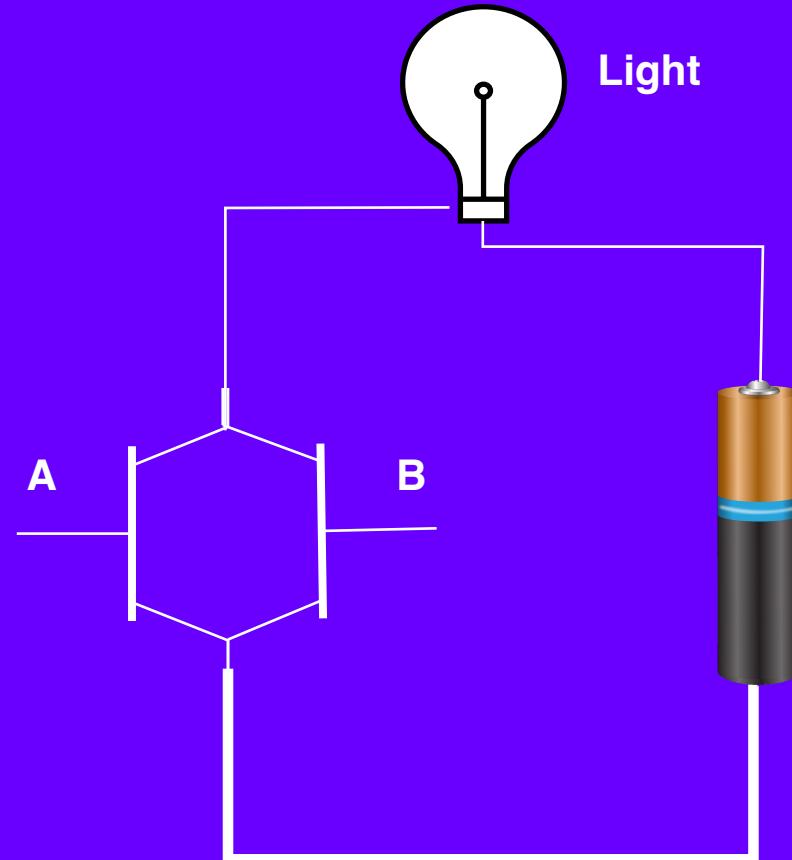
A	B	Light
True (1)	True (1)	True (1)
True (1)	False (0)	False (0)
False (0)	True (1)	False (0)
False (0)	False (0)	False (0)

1. There are many different ways to draw this

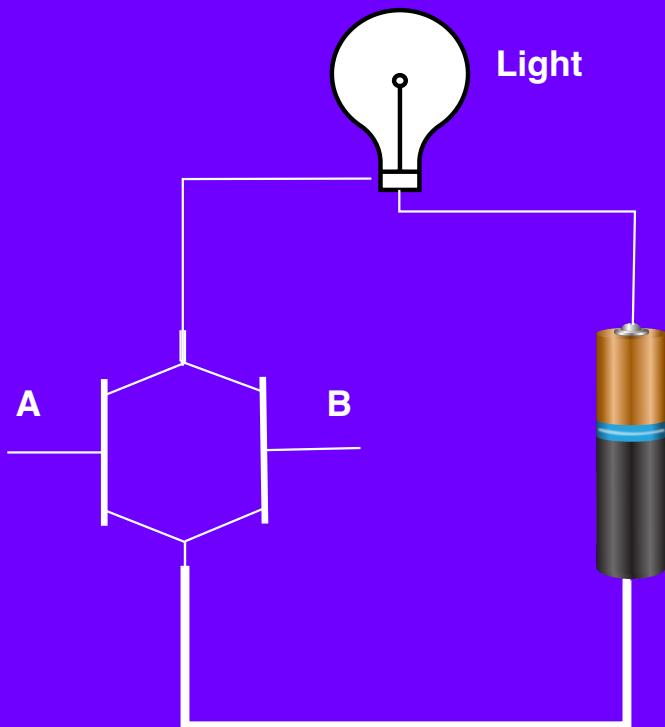
What else can we do with these transistors?

Put them next to each other!

This is an **OR gate**



OR Gate¹



Truth Table

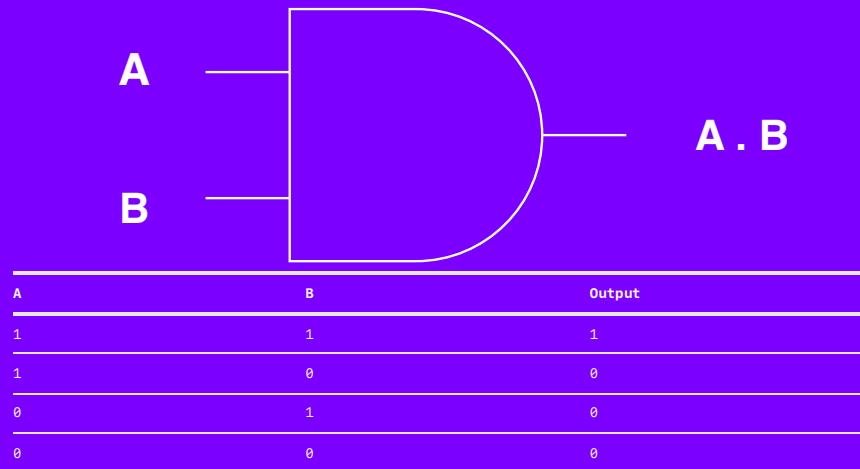
A	B	Light
True (1)	True (1)	True (1)
True (1)	False (0)	True (1)
False (0)	True (1)	True (1)
False (0)	False (0)	False (0)

1. There are many different ways to draw this

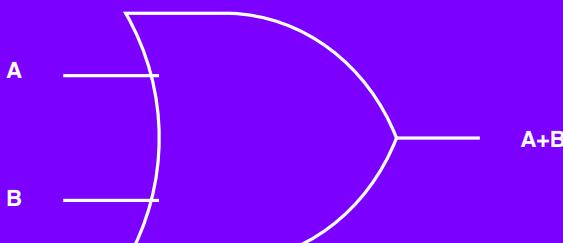
**There is a lot that goes on
with a transistor and gates,
we only scratch the surface in
this course**

The Three Basic Gates and Their Symbols

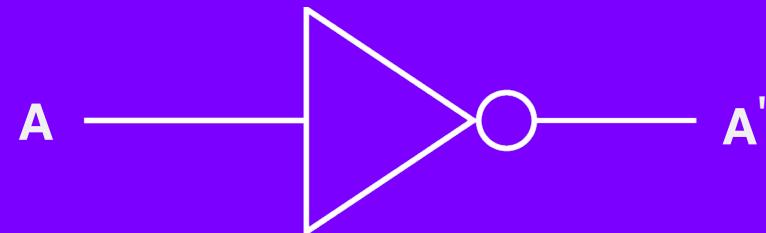
And Gate



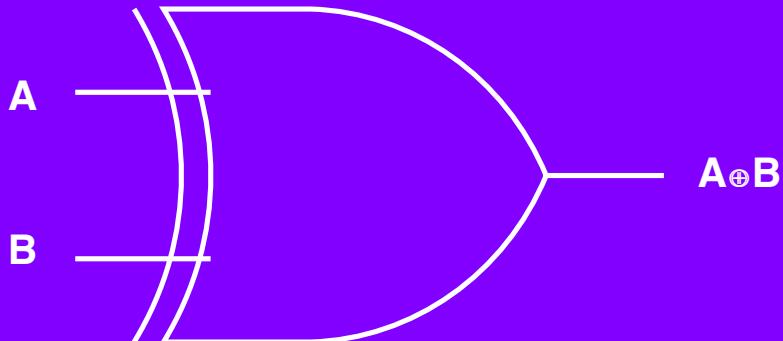
OR Gate



NOT Gate



XOR Gate



A	B	Output (A XOR B)
0	0	0
0	1	1
1	0	1
1	1	0

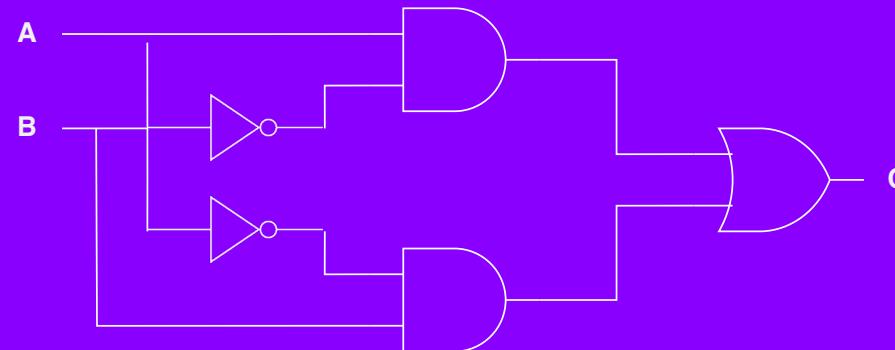
Boolean Algebra

- Boolean algebra is a mathematical system that deals with true and false values, represented as 1 and 0
- It provides a framework for manipulating logical expressions using operators like AND, OR, and NOT

Boolean Expression for XOR Gate:

$$A \cdot B' + A' \cdot B$$

Combinational Circuits



$$\text{XOR: } A \cdot B' + A' \cdot B$$

Abstraction in Hardware

Part 2/4

Abstraction in hardware design

- Map hardware devices¹ to Boolean logic
- Design more complex devices in terms of logic, not electronics
- Conversion from logic to hardware design may be automated

1. Such as the combinational gates you just looked at

Some Background: Binary Number System

- Humans use Decimal number system
 - $7809 = 7 \times 10^3 + 8 \times 10^2 + 0 \times 10^1 + 9 \times 10^0$
 - Each digit is from $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ – Base 10
 - (We happen to have ten fingers 
- Computers use Binary number system
 - $(1101) = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$
 - Each binary digit (bit) is $\{0, 1\}$ – Base 2
 - (IT people have 2 fingers¹)

1. not really!

Convert Decimal Number

10 to Binary 2

Conversion steps:

- Divide the number by 2
- Get the integer quotient for the next iteration
- Get the remainder for the binary digit
- Repeat the steps until the quotient is equal to 0

Example:

Division

by 2	Quotient	Remainder	Bit #
13/2	6	1	0
6/2	3	0	1
3/2	1	1	2
1/2	0	1	3

So $13_{10} = 1101_2$

Convert Binary 2 to Decimal Number 10

For binary number with n digits:

$$d_{n-1} \dots d_3 d_2 d_1 d_0$$

The decimal number is equal to the sum of binary digits (d_n) times their power of 2 (2^n):

$$\text{decimal} = d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 + \dots$$

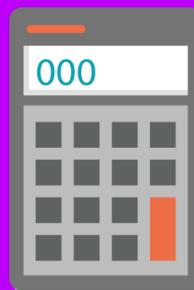
Example:

Find the decimal value of 111001_2 :

binary number:	1	1	1	0	0	1
power of 2:	2^5	2^4	2^3	2^2	2^1	2^0
111001_2	=	$1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	=	57_{10}		

Creating a calculator capable of adding two numbers using a combinational circuit

Part 3/4



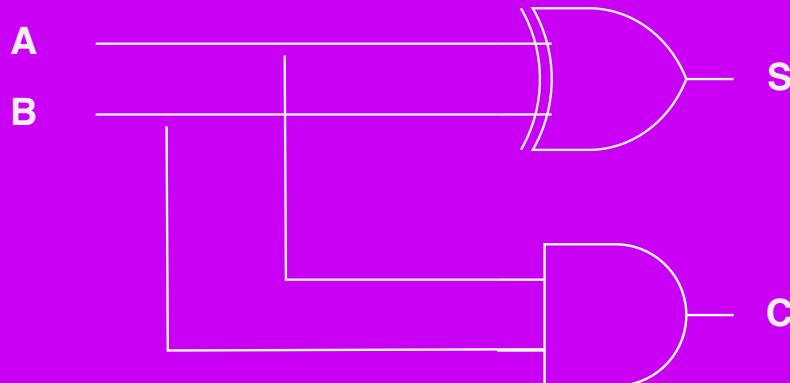
Easy Case: 2 Digit Addition

A	B	Output (A+B)	C S
0	0	0	0 0
0	1	1	0 1
1	0	1	0 1
1	1	2	1 0

For now, we only add two digits without a carry forward number

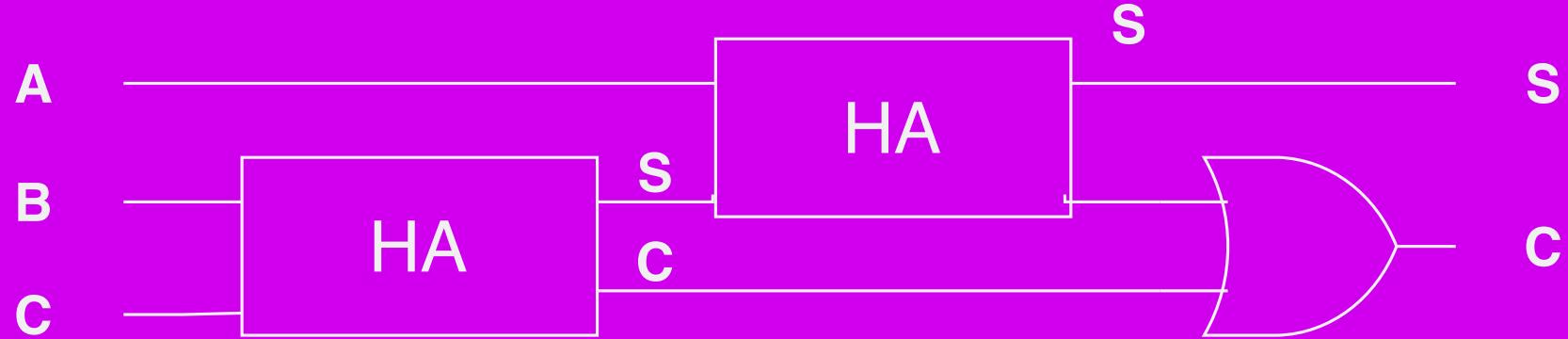
You need one AND gate and XOR gate to get this output

Half Adder (HA)



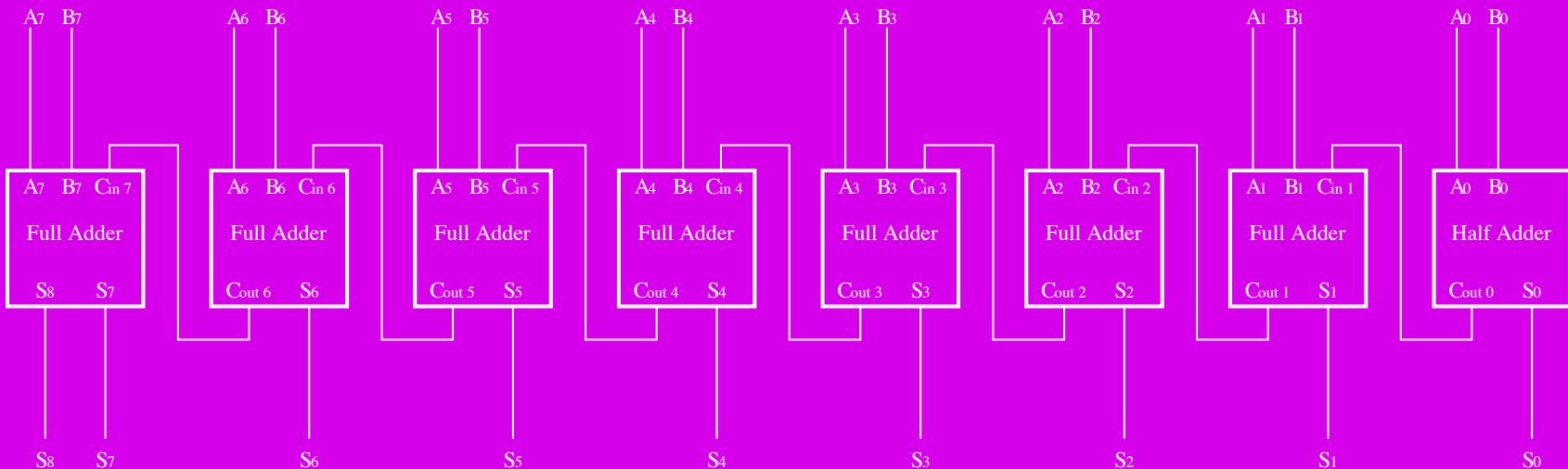
A	B	C	D
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

More abstraction (Handling the carry bit)

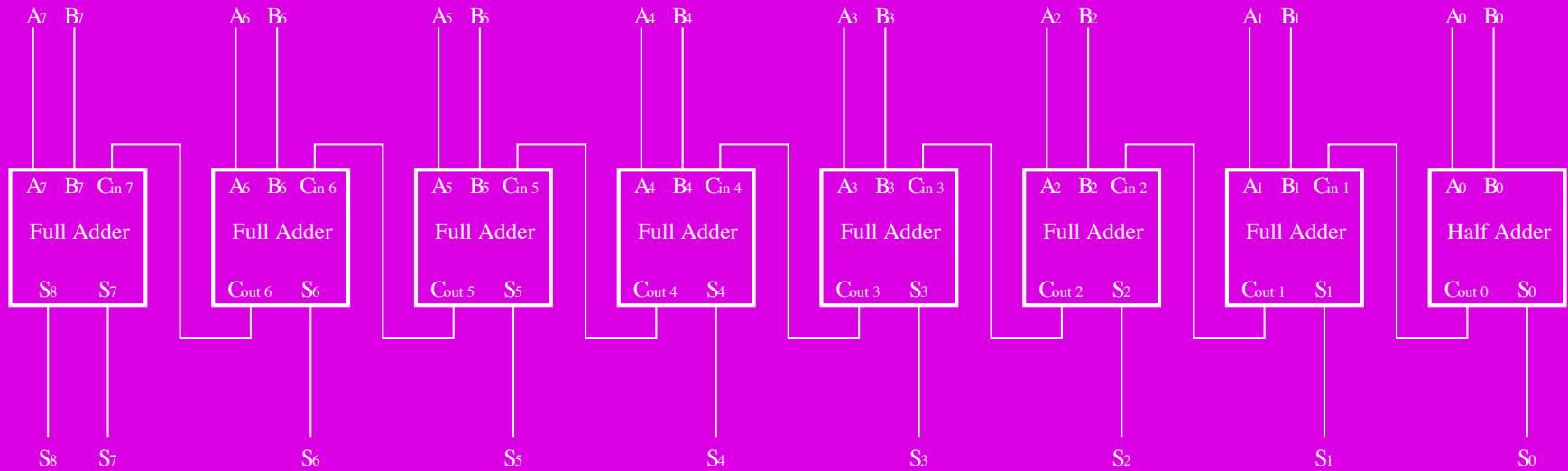


A	B	C	S (Sum)	C (Carry)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

8 Bit Full Adder 1

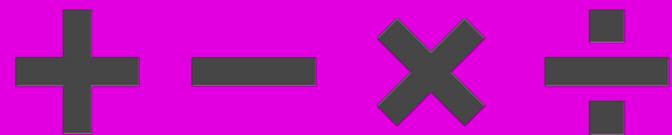


1. This is the most complex adder we would see in this class



Calculate: **153+75**

Binary: **10011001+01001011**

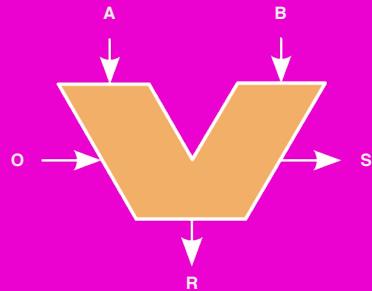


- You already know how to add
- You can also build subtractors¹
- You can substitute multiplication with addition (5*4 is 5+5+5+5)
- You can substitute division with subtraction

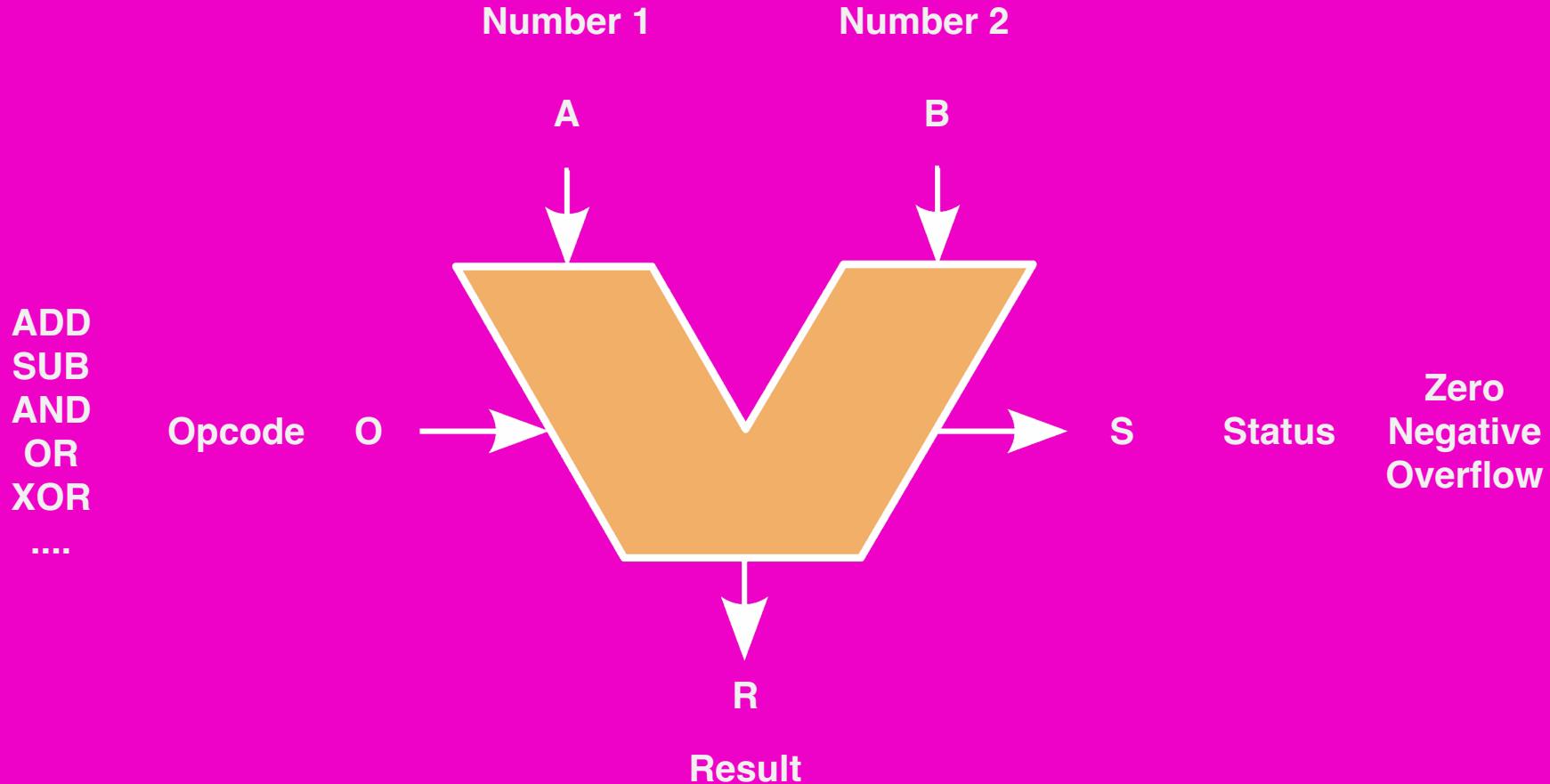
1. We do not cover subtractors in this course

Arithmetic Logic Unit

Part 3/4



The ALU combines multiple full adders and additional logic circuits to perform arithmetic and logical operations (AND, OR, XOR and even more) ✨



Opcode

Opcode	Instruction
--------	-------------

0000 A AND B

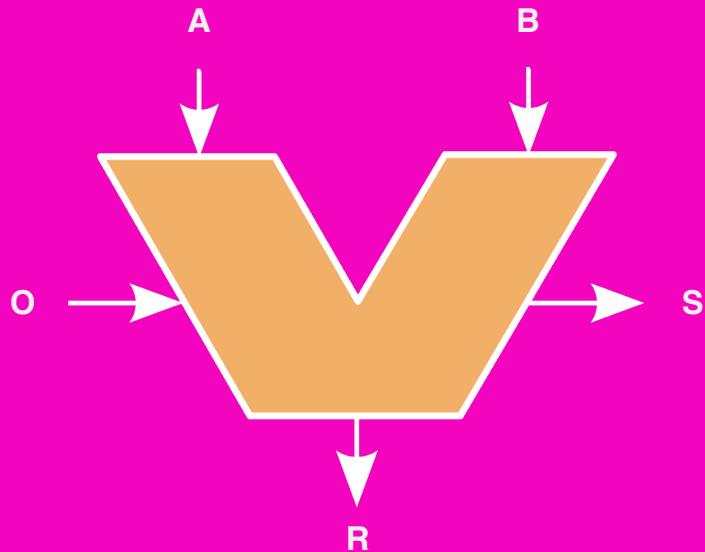
0001 A OR B

0010 A XOR B

0010 NOT A

0100 ADD A+B

0101 SUB A-B



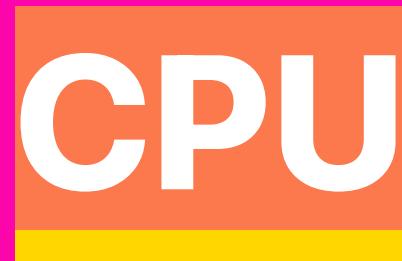
$$O = 0100$$

$$A = 00001010 \quad \& \quad B = \\ 01011101$$

$$R = ?$$

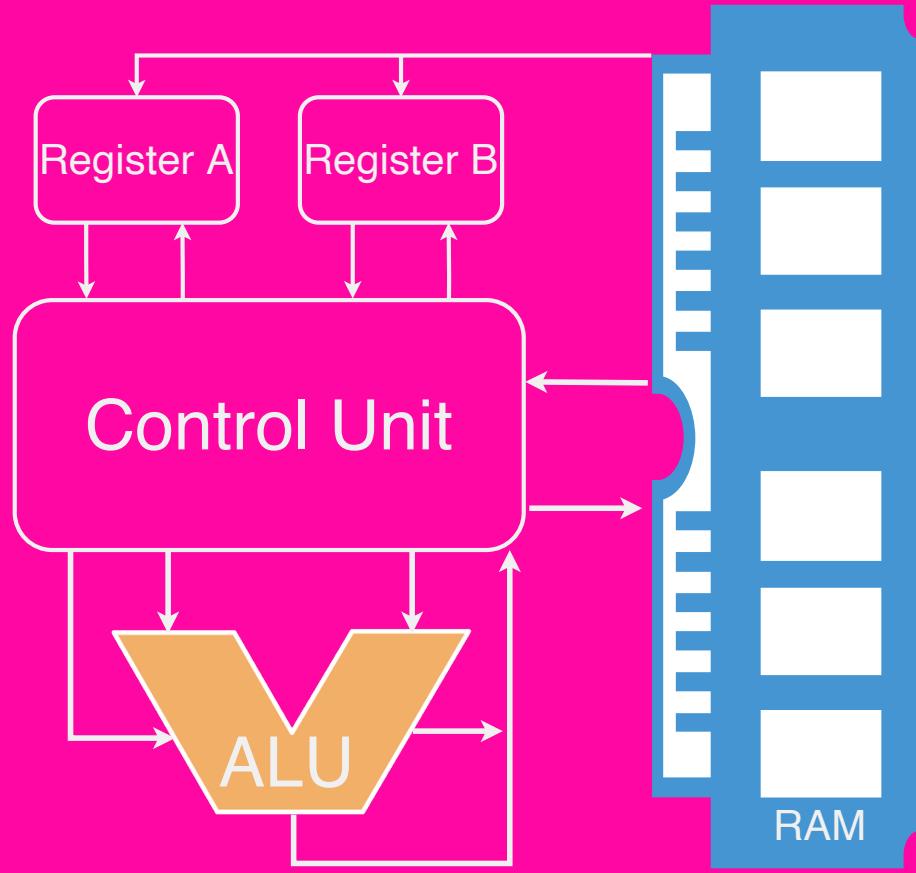
**What do you do when you
have to perform
multiplication?
(Or anything that requires more
than one instruction)**

More Abstraction



Central Processing Unit

Part 3/4



Control Unit

- The control unit receives instructions from memory and controls the flow of data within the CPU
- It interprets opcode (operation code) to determine the operation to be performed by the ALU or memory

Memory and Random Access Memory (RAM)

- Registers are temporary storage units within the CPU that hold data during processing
- RAM (Random Access Memory) stores data and instructions that the CPU accesses during execution
- Data and instructions are loaded from RAM into the CPU registers for processing

Instruction Set

- Instruction sets are collections of binary-coded instructions that a computer's CPU can execute
- These instructions represent specific operations like arithmetic, memory access, and control flow
- There are two main types: RISC with simple instructions for faster execution and CISC with more complex instructions to reduce program size

Different processors use specific instruction sets optimized for various applications and performance requirements

Instruction Set Example

Instruction	Opcode	Memory Location	Description
ADD	0 0 0 1	2* 2-bit register ID	Add two numbers ¹
AND	0 0 1 0	2* 2-bit register ID	Add operation
LOAD_A	0 1 1 0	4-bit memory address	Load memory address in register A
LOAD_B	0 1 1 1	4-bit memory address	Load memory address in register B
STORE_B	1 0 1 1	4-bit memory address	Write register A into memory address
HALT	0 1 0 0	N/A	Halt the program

1. Result is stored in the second register

Let's write your first program

Program to add two numbers

1. Load numbers into registers from RAM

- 1.1 Locate the number in RAM (use LOAD_A & LOAD_B Opcode)

LOAD_A + address 1 -> 0110 1110

LOAD_B + address 2 -> 0111 1111

2. Add the values at register A and B

Add opcode + 2 register IDs -> 0001 01 10

3. Save our result into the RAM

STORE_B + memory address -> 1011 1101

4. Stop the program

HALT -> 0100

Congratulations! You just wrote
your first program in machine
language (code)

```
0110 1110
```

```
0111 1111
```

```
0001 01 10
```

```
1011 1101
```

```
0100
```

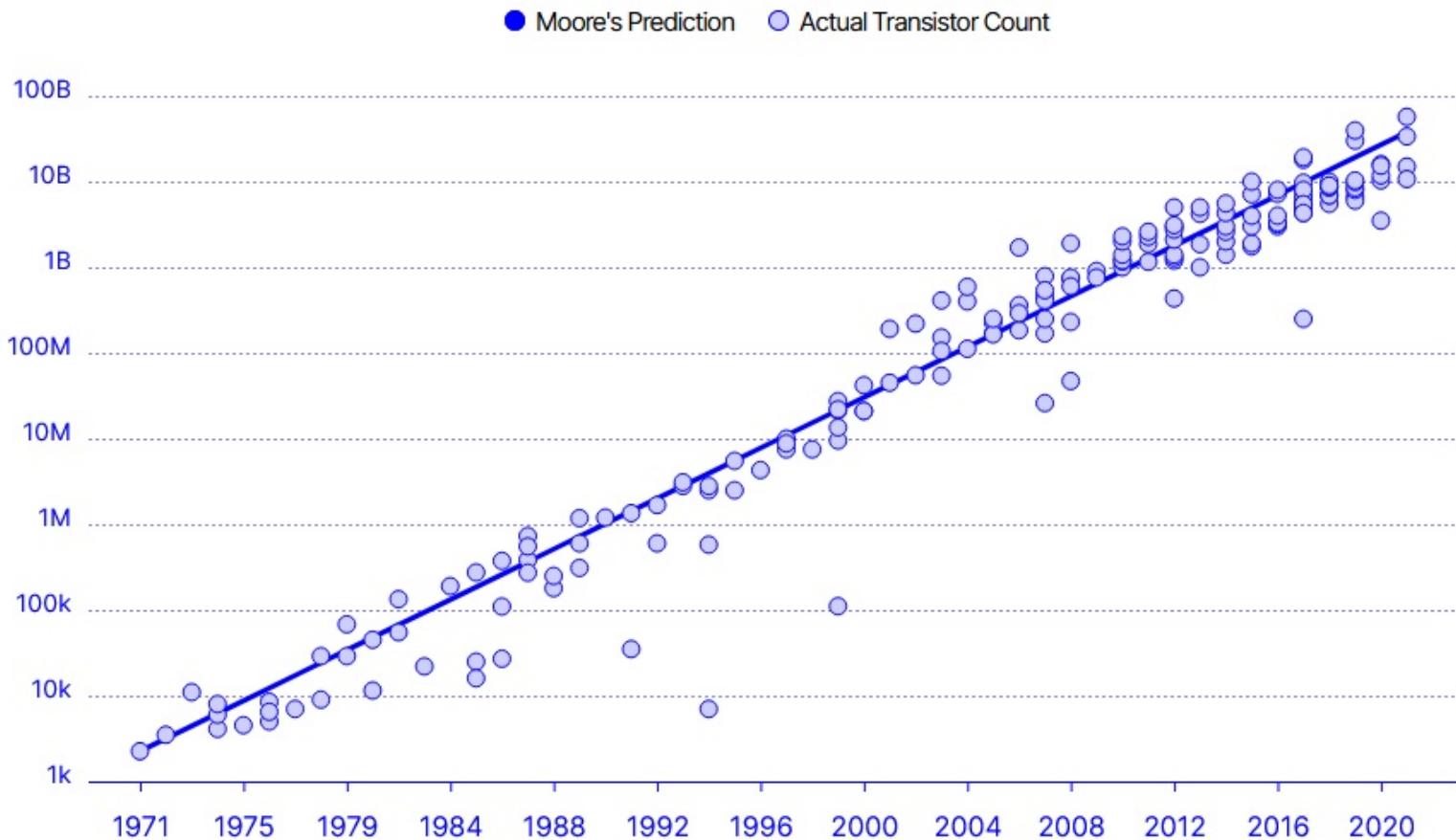
Machine Language

- Machine language consists of binary instructions (1s and 0s) that the CPU can directly execute
- Each instruction is represented by an opcode, specifying the operation, and memory addresses for data access

Very difficult for humans to work with machine code! → Use abstraction – high level programming languages such as C, C++ and Java

Moore's Law

"The number of transistors in a dense integrated circuit (IC) doubles about every two years."





Your PC can't even

Computer Dissection



Part 4/4

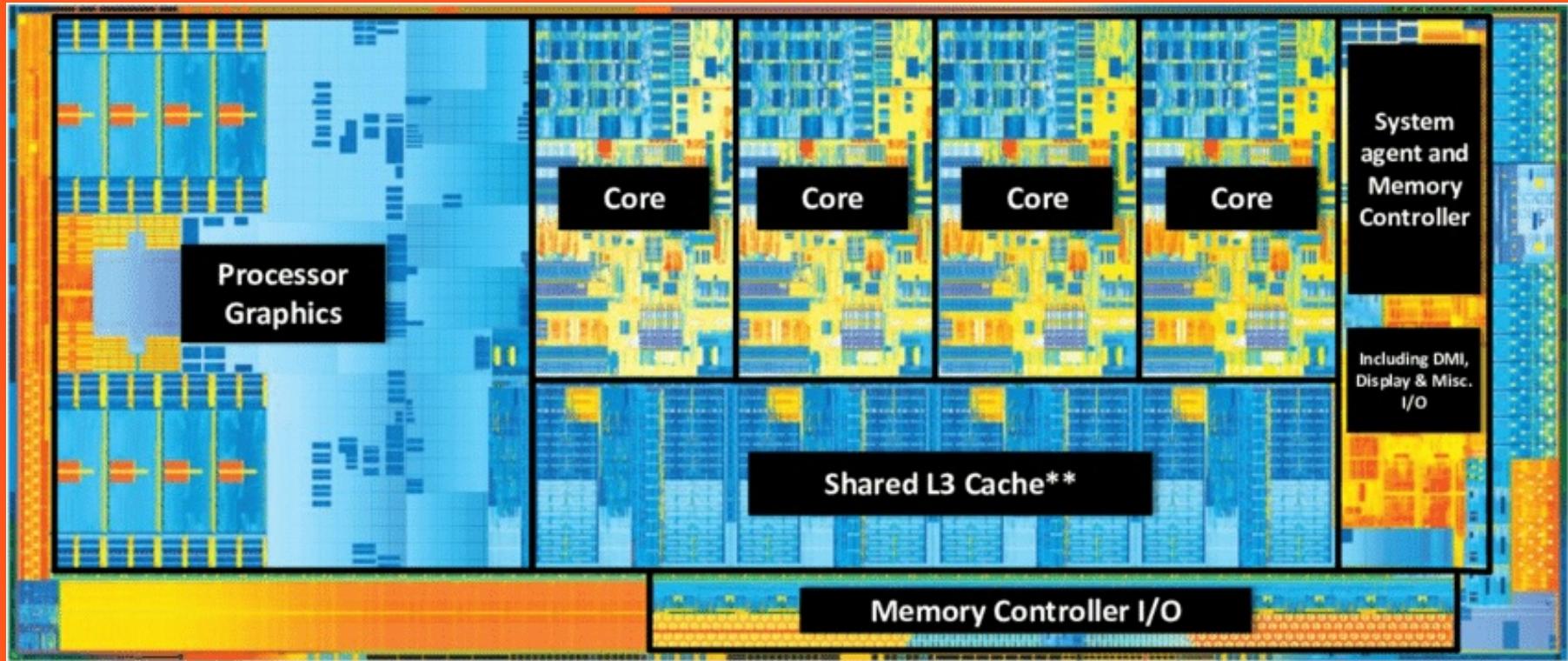
Central Processing Unit

- The CPU is the computer's brain 
 - It consists of an integrated  heat spreader cover, a metal package holding the integrated circuit (die), and a printed circuit board for connection to the motherboard
 - The die contains various sections, including cores for executing programs and instructions



CPU Functional Sections

- The CPU has additional sections, such as shared L3 memory cache, integrated graphics processor, memory controller, and system agent/platform I/O
- The memory controller manages data transfer to and from DRAM, while the system agent facilitates communication with the motherboard chipset

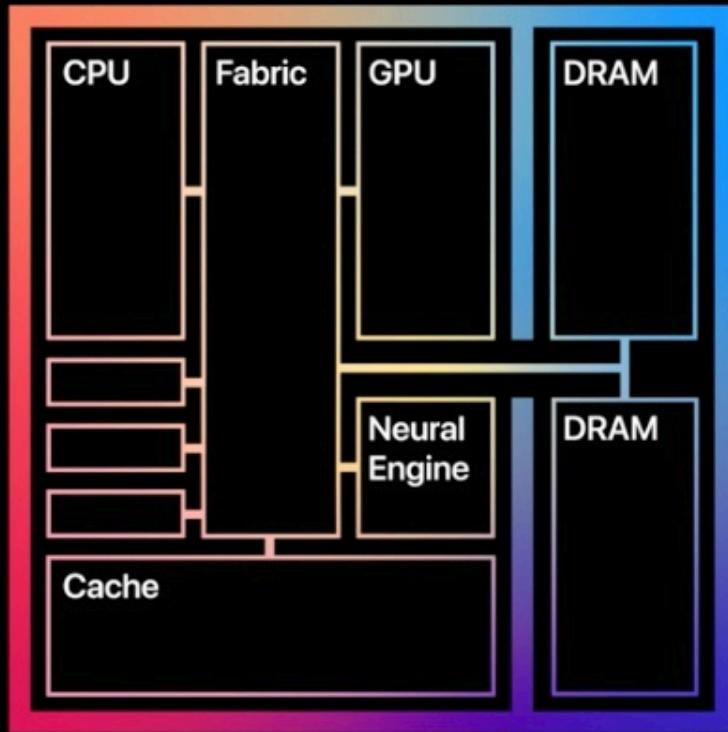


Intel Ivy Bridge





Apple M1

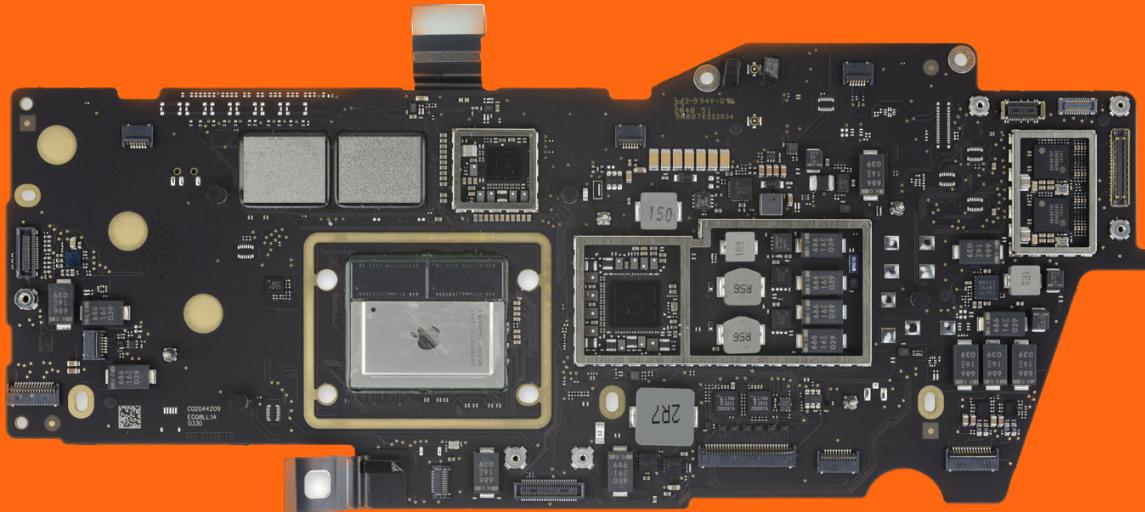


Motherboard

A large printed circuit board with numerous wires and various microchips, components, sockets, ports, slots, headers, and connectors.



Motherboard for a laptop



Power Supply

- The power supply unit (**PSU**) distributes power throughout the computer
- It contains a main transformer, control PCB, switching power transistor, and various components for voltage regulation and output stability



CPU Cooler



CPU cooler includes a pump, tubes, radiator, and fans to dissipate heat generated by the CPU

The liquid circulates through the system, transferring heat to the radiator, and then the fans cool the liquid

GPU



- The GPU is the brain of the computer's graphics capabilities
- It consists of a PCB, integrated circuit (IC), VRAM chips, voltage regulator module, and cooling system
- The GPU IC contains billions of transistors and performs parallel processing using multiple cores

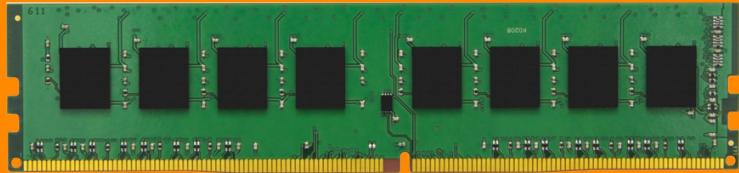
Storage



Part 4/4

Dynamic Random Access Memory

- The CPU communicates directly with the DRAM through memory channels on the motherboard
- DRAM chips store data temporarily and use capacitors and transistors organized into 2D arrays



Solid-State Drives (SSDs)



- SSDs store data permanently using 3D arrays of memory cells called 3D NAND
- These arrays are stacked within a single SSD chip, enabling the storage of terabytes of data

Hard Disk Drives (HDDs)

- HDDs use spinning disks and read/write heads to access data stored on magnetic surfaces.
- The read/write head moves across data tracks to read or write information





Conclusion

- Building blocks of a computer
- Construction of an Arithmetic Logic Unit (ALU)
- Central Processing Unit (CPU)
- Computing Hardware Overview

**See you in the
lab!**

