Lecture-3

How computer works?

- How to represent 0, 1
  - Transistor
- Logical Operations
- Program
- Algorithm
- …

Last week: complex data representation in binary

Integers

- Signed integers
- 2s complement Two's complement - Wikipedia

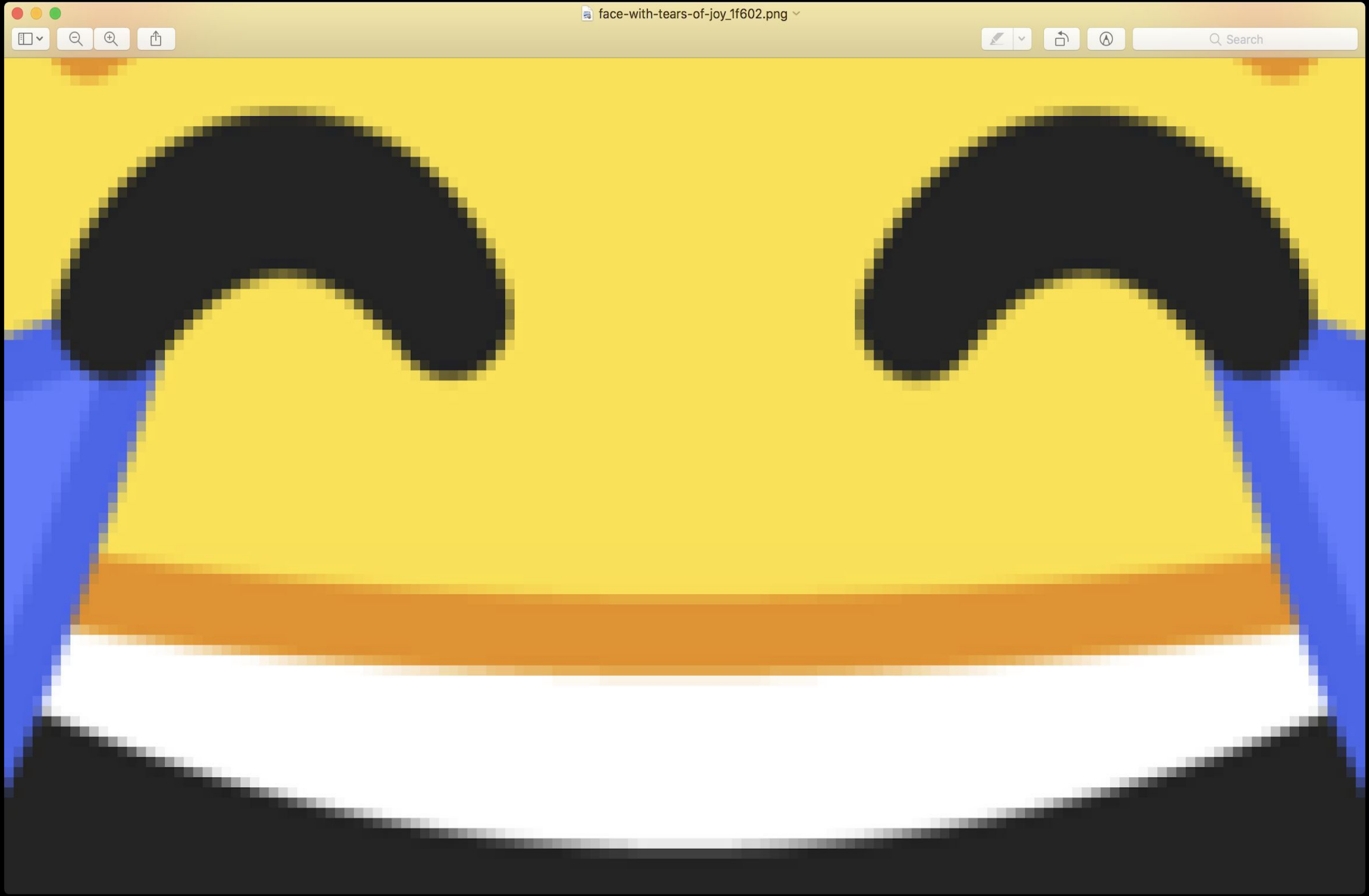
Floating point numbers IEEE Standard 754 Floating Point Numbers - GeeksforGeeks

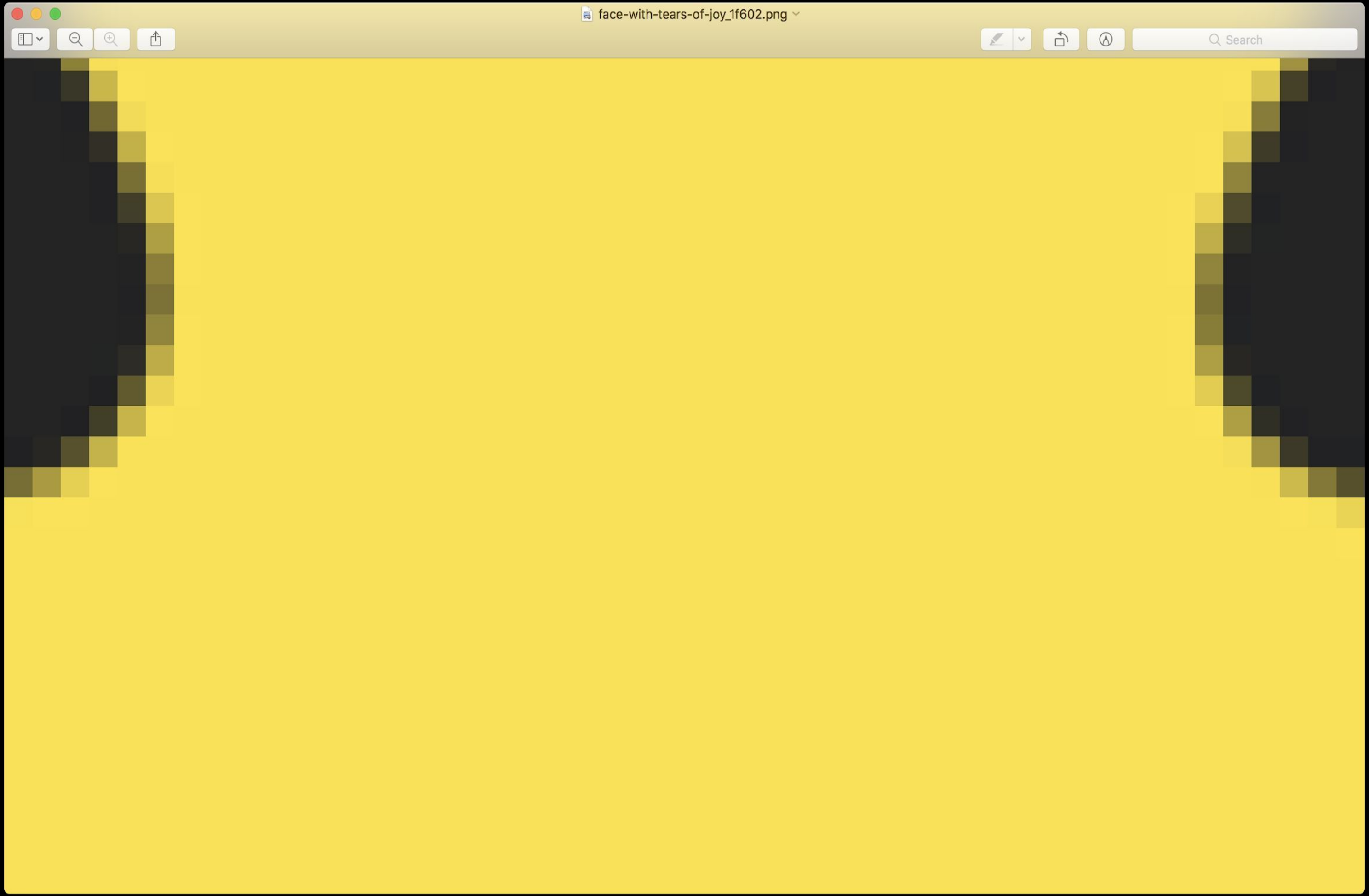
- 1.4
- 5.25
- IEEE 754

- ➔ Characters (symbols):
  - 1,0, s, x_,!$%^alsjkdom;lsmdf;l/*65
  - ASCII codes ASCII table
  -
  -
- ➔ Images ?
  - ◆ RGB values
    - Colors RGB and RGBA
- ➔ Musics ?


face-with-tears-of-joy_1f602.png
Search


face-with-tears-of-joy_1f602.png
Search


face-with-tears-of-joy_1f602.png
Search


face-with-tears-of-joy_1f602.png
Search

RGB

72	73	33

This week

How computer works?

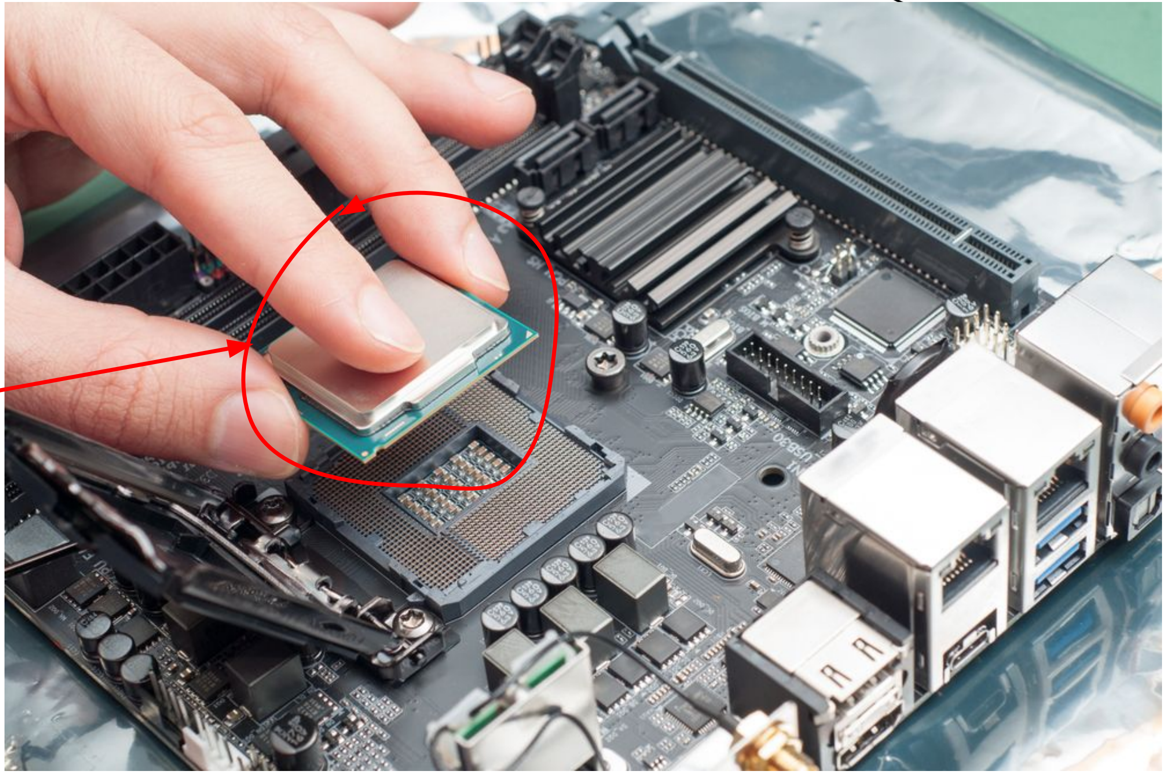
- How to represent 0, 1
  - Transistor
  - Digital circuits
- Logic Gates
  - Logical operations
- Program
- Algorithm
- …

How a computer works?

V_{CC}

Digital machine

CPU

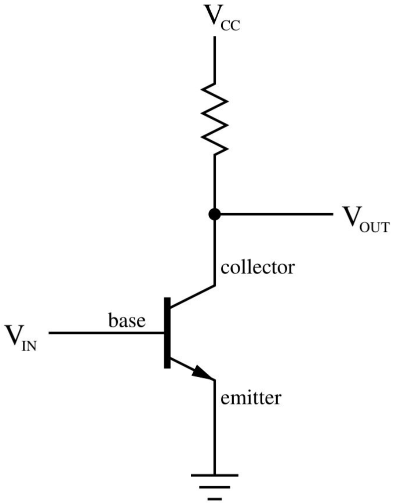


CPU socket (Image credit: Mastermilmar/Shutterstock)

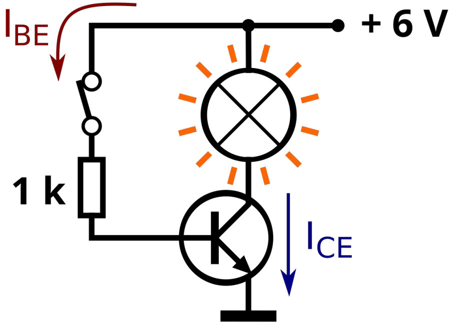
How a computer works?

CPU

transistor



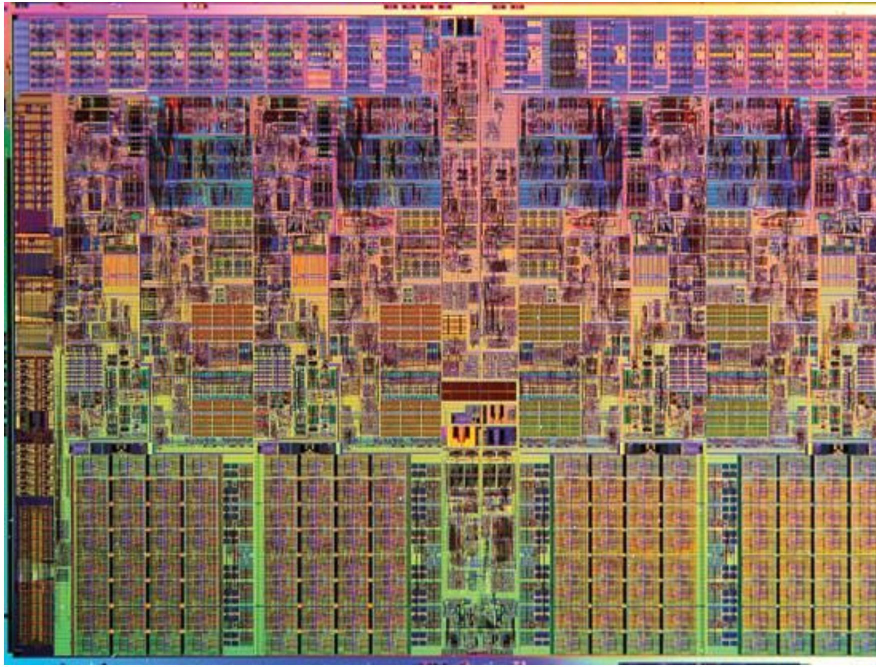
Transistor as switch



https://en.wikipedia.org/wiki/Transistor

How many transistors in CPU?

- 1st microprocessor, Intel 4004 (1971), had 2,300 transistors.
- 1st 32-bit microprocessor, Motorola 68000 (1979), had 68,000 transistors.
- 1st 64-bit microprocessor, MIPS R4000 (1991), had 1.35 million transistors.
- 1st Pentium processor, Intel Pentium (1993), had 3.1 million transistors.
- 1st 2-core processor, AMD Athlon 64 X2 (2005), had 233.2 million transistors.
- 1st 4-core processor, Intel Core 2 Quad (2006), had 582 million transistors.
- 1st 6-core processor, Intel Core i7-980X (2010), had 1.17 billion transistors.
- 1st 8-core processor, AMD FX-8150 (2011), had 1.2 billion transistors.
- 1st 10-core processor, Intel Core i7-6950X (2016), had 3.2 billion transistors.
- 1st 12-core processor, AMD Ryzen Threadripper 1920X (2017), had 9.6 billion transistors.
- 1st 16-core processor, AMD Ryzen Threadripper 1950X (2017), had 19.2 billion transistors.
- 1st 18-core processor, Intel Core i9-7980XE (2017), had 6.5 billion transistors.
- 1st 20-eight-core processor, Intel Xeon W-3175X (2019), had 8.6 billion transistors.
- 1st 30-two-core processor, AMD Ryzen Threadripper 2990WX (2018), had 19.2 billion transistors.
- 1st 64-core processor, AMD Ryzen Threadripper 3990X (2020), had 39.54 billion transistors.



https://www.icdrex.com/the-brain-behind-the-machine-transistors-in-cpu-architecture/

How to use a transistor or similars to do binary operations?

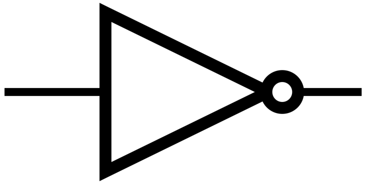
Logic operations: NOT(Negation)

Truth table for Proposition P

P	$\neg P$
True	False
False	True

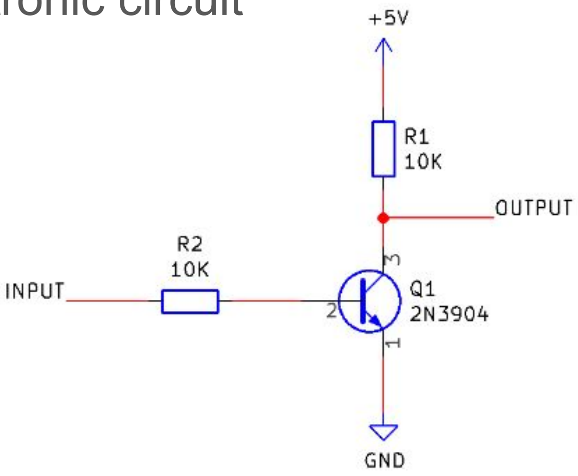
Gate representation (digital circuits)

NOT gate



The RTL NOT Gate

An electronic circuit for NOT



Input	Output
1 (VDD)	0 (GND)
0 (GND)	1 (VDD)

Supply voltage shown as 5V but RTL gates can operate on voltages as low as 1V and as high as 12V

Input is off (Input = 0V)

- Q1 is turned off
- the output is connected to 5V via the 10K resistor
  - the output is on (5V).

Input is on (Input = VSupply)

- Q1 fully turns on
  - It connects the output to ground through the transistor.
- the output switches to 0V
- therefore the NOT Gate function is realised.

https://www.mitchelectronics.co.uk/documents/RTL.pdf

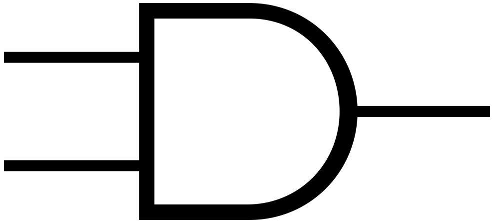
Logic operations: AND (logical conjunction)

Truth table of **A and B**

A	B	$A \wedge B$
F	**F**	F
F	**T**	F
T	**F**	F
T	**T**	T

Gate representation (digital circuits)

AND gate



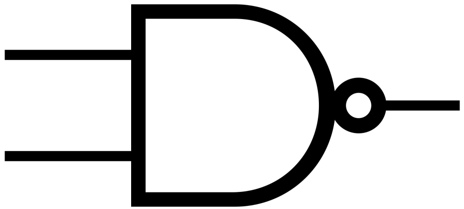
Logic operations: NAND

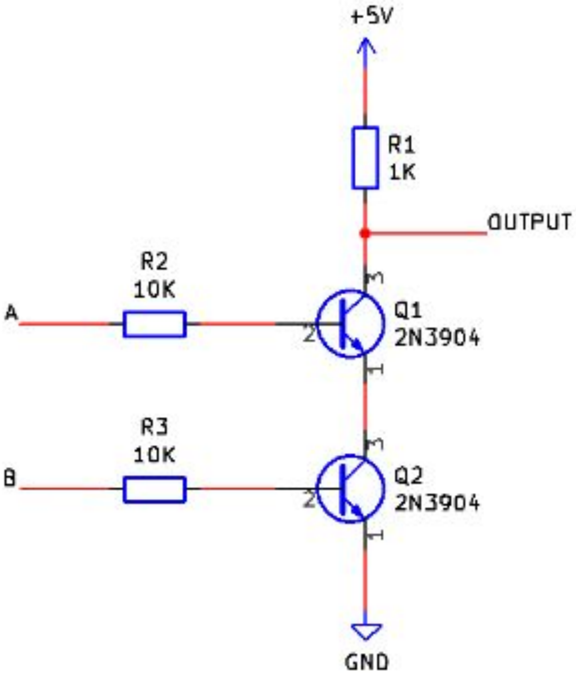
Truth table of **A NAND B**

A	B	$A \uparrow B$
F	**F**	T
F	**T**	T
T	**F**	T
T	**T**	F

Gate representation (digital circuits)

NAND gate





NAND Gate

NAND gate truth table

Input (A)	Input (B)	Output
0 (GND)	0 (GND)	1 (VDD)
0 (GND)	1 (VDD)	1 (VDD)
1 (VDD)	0 (GND)	1 (VDD)
1 (VDD)	1 (VDD)	0 (GND)

https://www.mitchelectronics.co.uk/documents/RTL.pdf

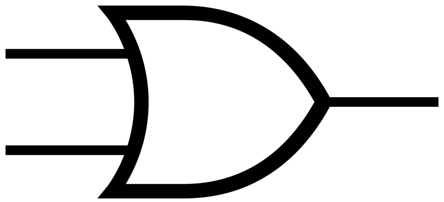
Logic operations: OR (logical disjunction)

Truth table

A	B	$A \lor B$
F	**F**	F
F	**T**	T
T	**F**	T
T	**T**	T

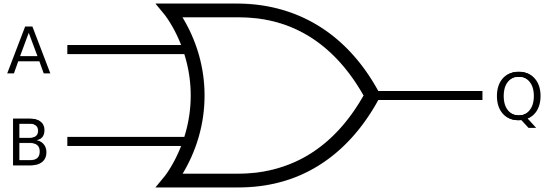
Gate representation (digital circuits)

OR gate

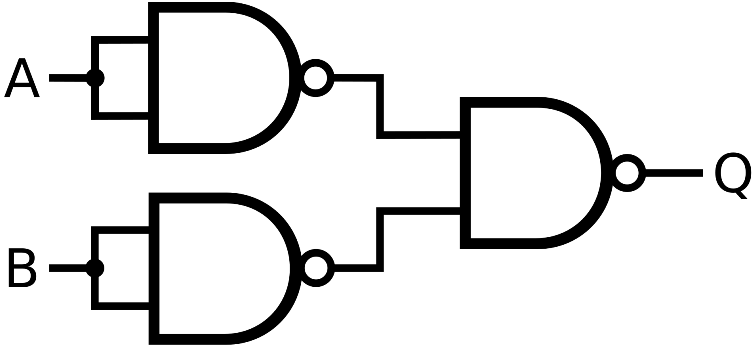


https://en.wikipedia.org/wiki/Logical_disjunction

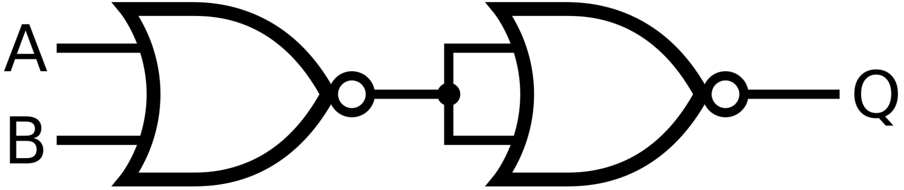
Combining logic circuits: OR gate construction



With NAND gates

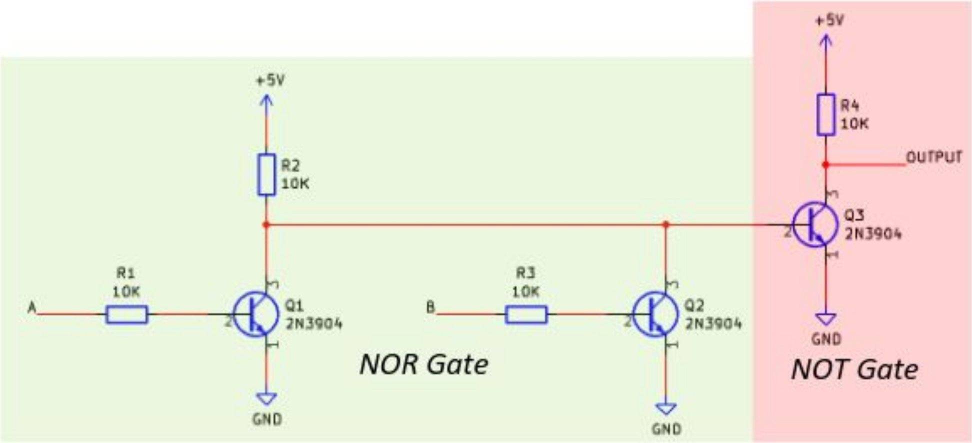


With XOR gates



https://en.wikipedia.org/wiki/OR_gate

Combining circuits
OR gate = NOR + NOT



OR Gate

NOR gate truth table

Input (A)	Input (B)	Output
0 (GND)	0 (GND)	1 (VDD)
0 (GND)	1 (VDD)	0 (GND)
1 (VDD)	0 (GND)	0 (GND)
1 (VDD)	1 (VDD)	0 (GND)

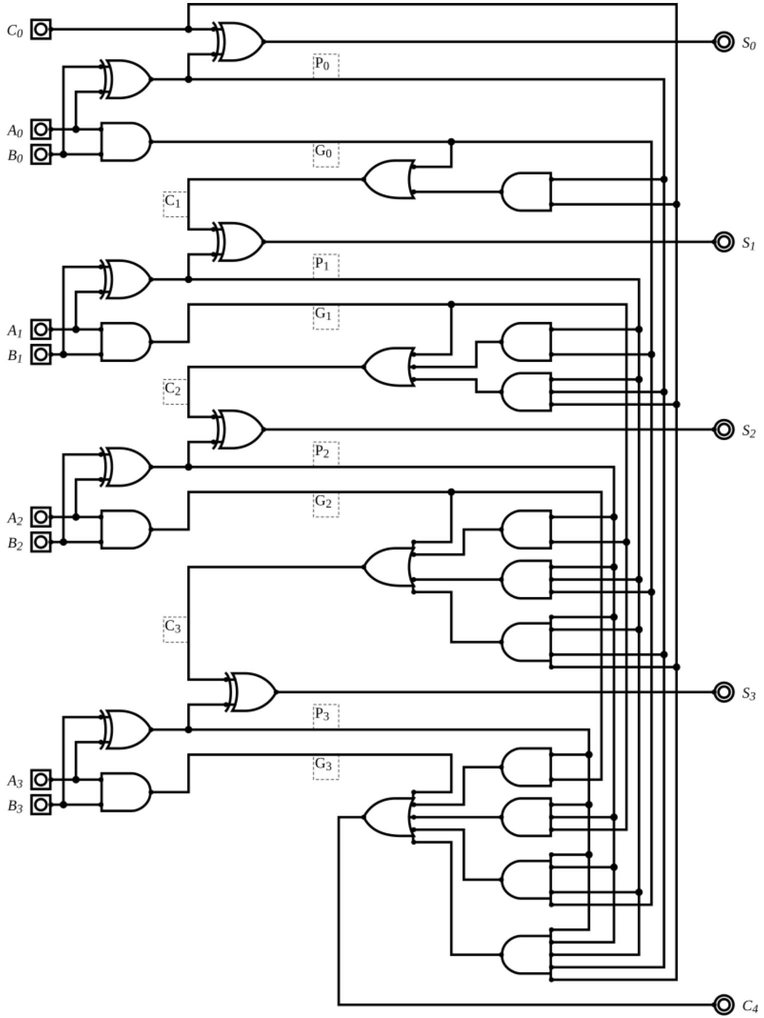
OR gate truth table

Input (A)	Input (B)	Output
0 (GND)	0 (GND)	0 (GND)
0 (GND)	1 (VDD)	1 (VDD)
1 (VDD)	0 (GND)	1 (VDD)
1 (VDD)	1 (VDD)	1 (VDD)

https://www.mitchelectronics.co.uk/documents/RTL.pdf

More complex logic circuits

A logic circuit diagram for a 4-bit carry lookahead binary adder design using only the AND, OR, and XOR logic gates.

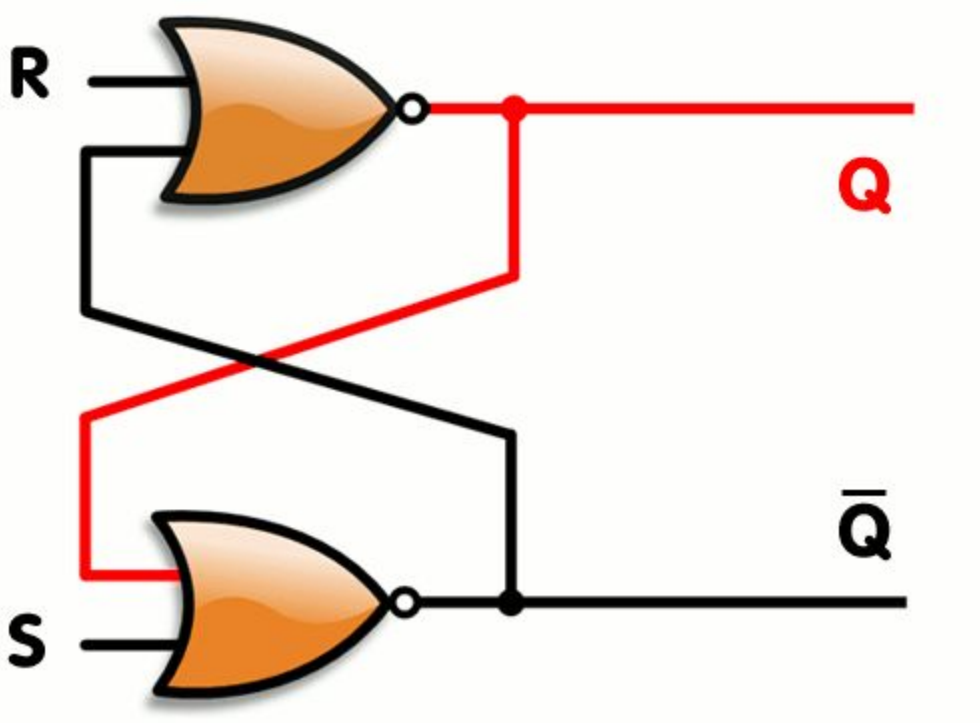


https://en.wikipedia.org/wiki/Logic_gate#

Data storage

Output is given as input

- S
  - Set
- R
  - reset



Animation of how an SR NOR gate latch works.

https://en.wikipedia.org/wiki/Logic_gate#

So far…

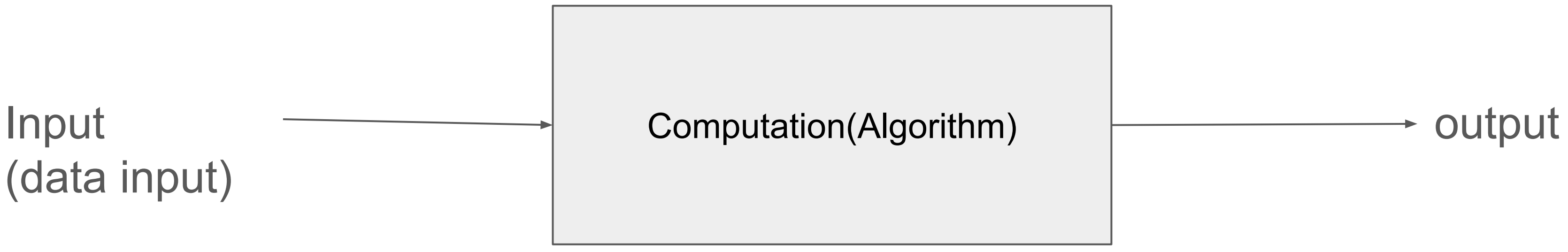
We have learned

- Logic gates
- Digital circuits
- Computer works based on binaries
- Logical operations
  - AND, OR, XOR, etc
- Arithmetic operations
  - By combining logical gates

Remember where we are coming from

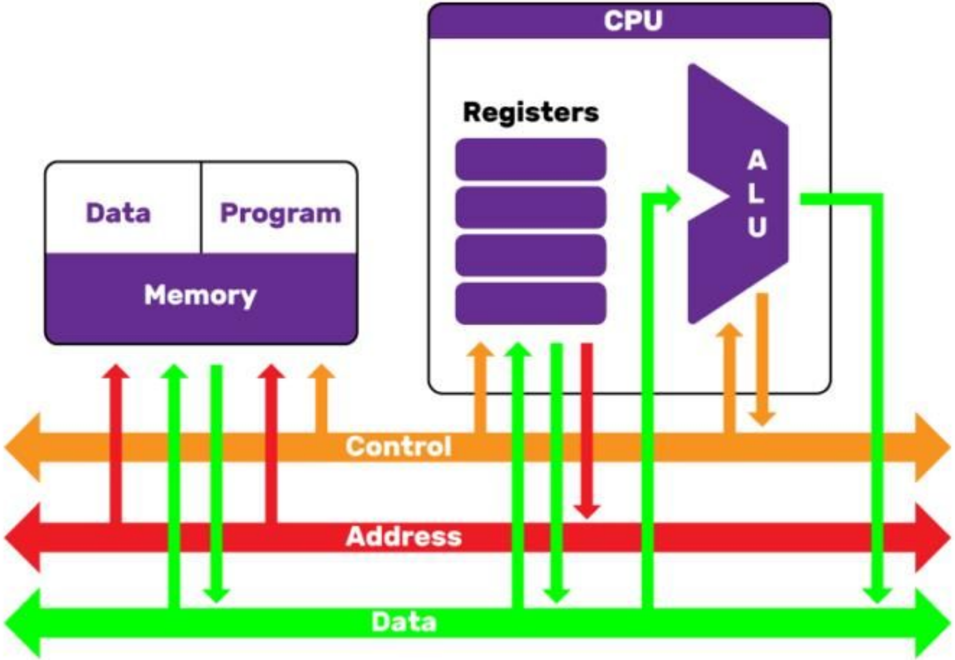
How to solve a problem by a computer program?

Program



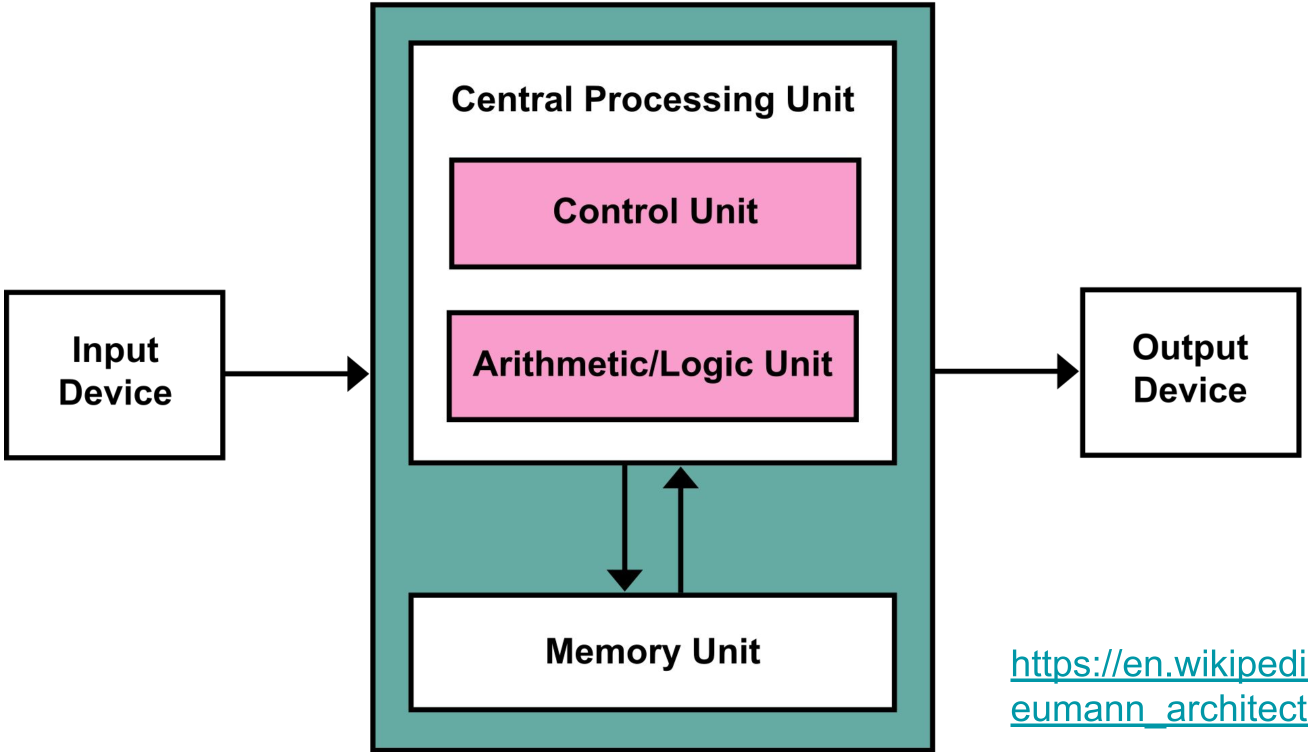
How to represent algorithms?

Computer Architecture

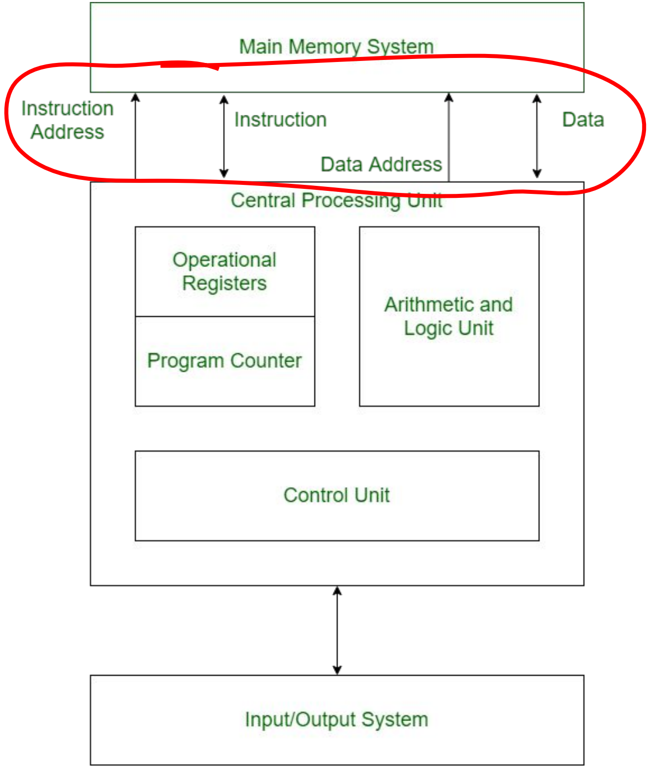


https://www.icdrex.com/the-brain-behind-the-machine-transistors-in-cpu-architecture/

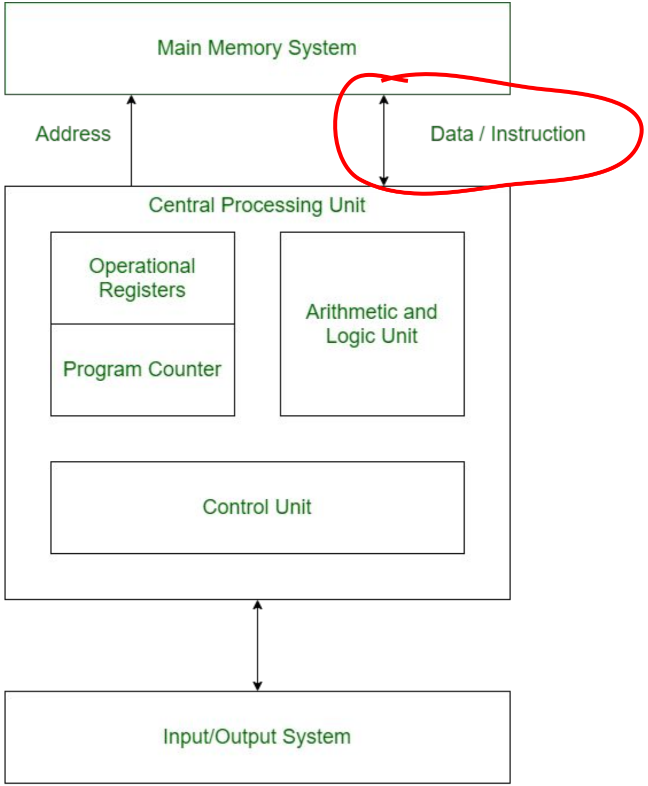
Von Neumann Model



https://en.wikipedia.org/wiki/Von_Neumann_architecture



Harvard Architecture



Von Neumann Architecture

https://www.geeksforgeeks.org/difference-between-von-neumann-and-harvard-architecture/

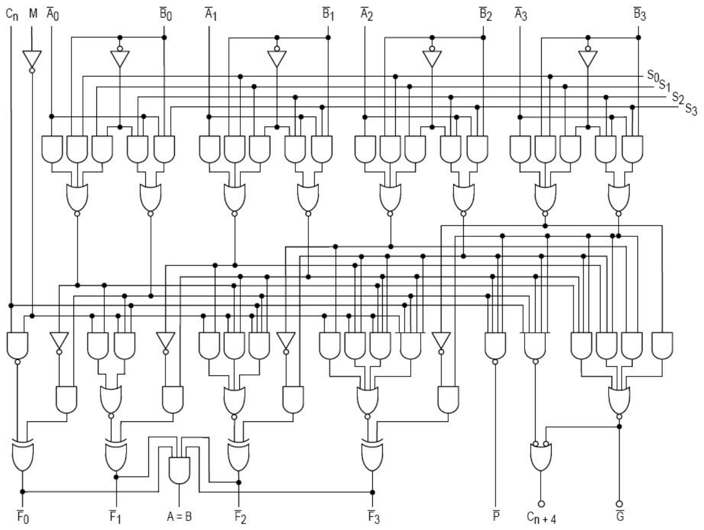
Arithmetic logic unit (ALU)

Combinational digital circuit

Performs

- logical operations
  - And, or, xor, 1's complement
- Arithmetic operations
  - Add, subtract, 2's complement, increment, decrement
- Bit shift operations
  - Arithmetic shift(sign is preserved), logical shift

on integer binary numbers



https://en.wikipedia.org/wiki/Arithmetic_logic_unit

Instructions

An instruction: is a command CPU can understand

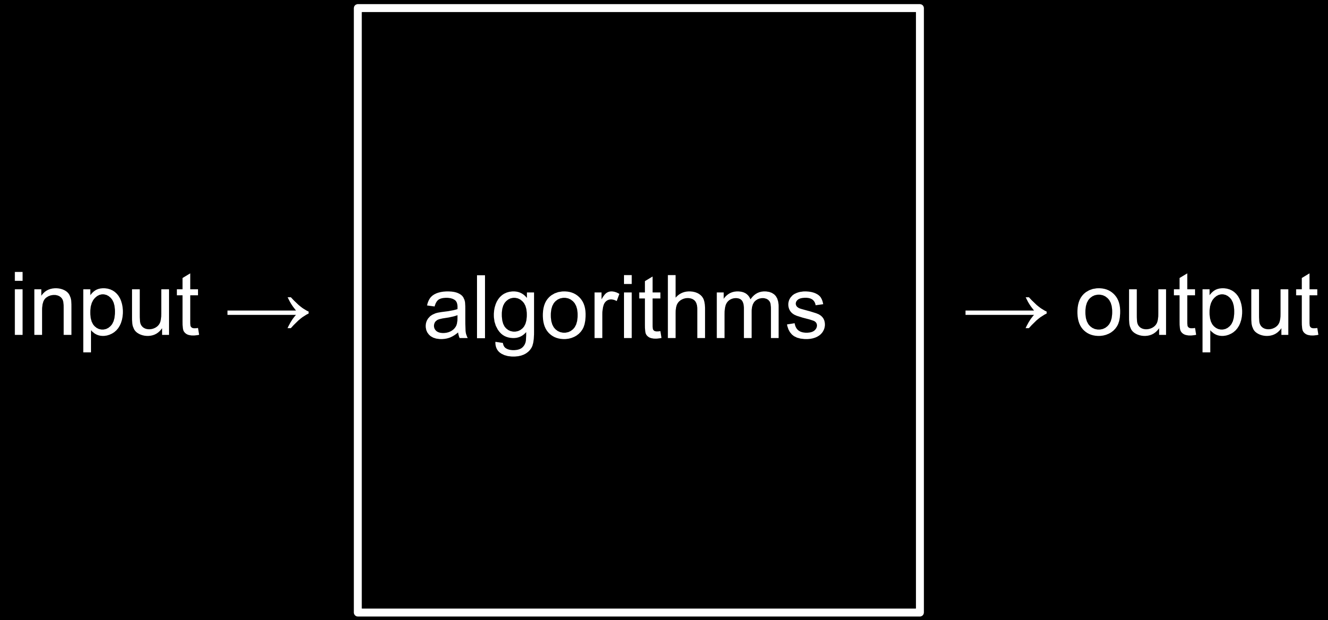
asm	machine code	Description
add	0x03 *ModR/M*	Add one 32-bit register to another.
mov	0x8B *ModR/M*	Move one 32-bit register to another.
mov	0xB8 *DWORD*	Move a 32-bit constant into register eax.
ret	0xc3	Returns from current function.
xor	0x33 *ModR/M*	XOR one 32-bit register with another.
xor	0x34 *BYTE*	XOR register al with this 8-bit constant.

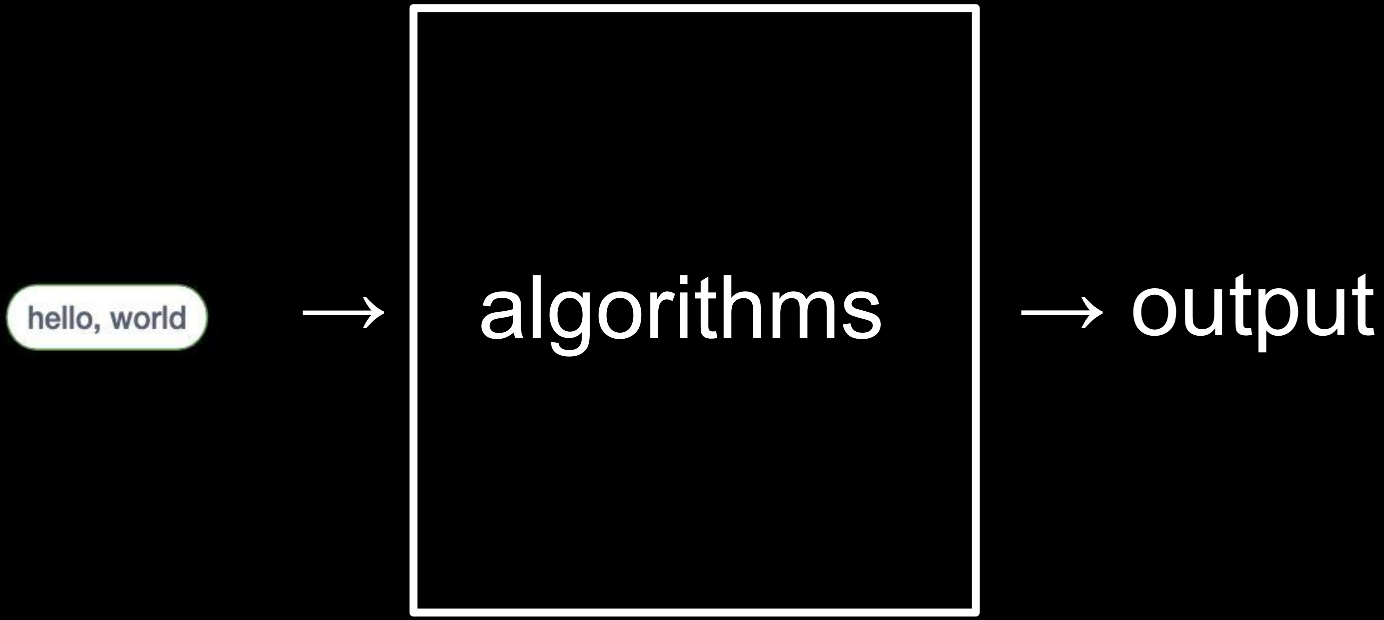
https://www.cs.uaf.edu/2016/fall/cs301/lecture/09_28_machinecode.html

Each binary byte represents a computation

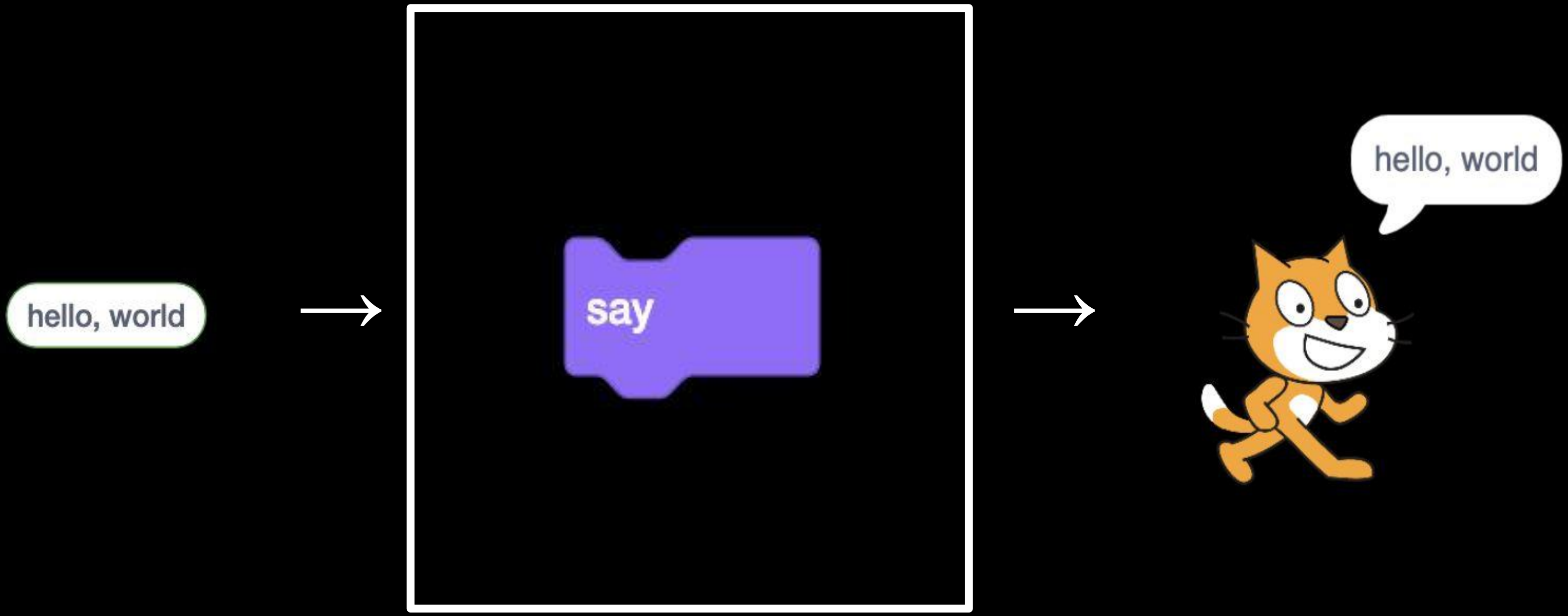
```
0:   b8 05 00 00 00        mov    eax,0x5
5:   c3                    ret
```

Example program say hello world!




hello, world
→
algorithms
→ output


hello, world
say
output


hello, world
say
hello, world

Hello world in assembly

```
section .text                               ;section declaration

                                            ;we must export the entry point to the ELF linker or
    global  _start                          ;loader. They conventionally recognize _start as their
                                            ;entry point. Use ld -e foo to override the default.

_start:

                                            ;write our string to stdout

    mov     edx,len                         ;third argument: message length
    mov     ecx,msg                         ;second argument: pointer to message to write
    mov     ebx,1                           ;first argument: file handle (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                            ;call kernel

                                            ;and exit

    mov     ebx,0                           ;first syscall argument: exit code
    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                            ;call kernel

section .data                               ;section declaration

msg db      "Hello, world!",0xa ;our dear string
len equ     $ - msg                         ;length of our dear string
```

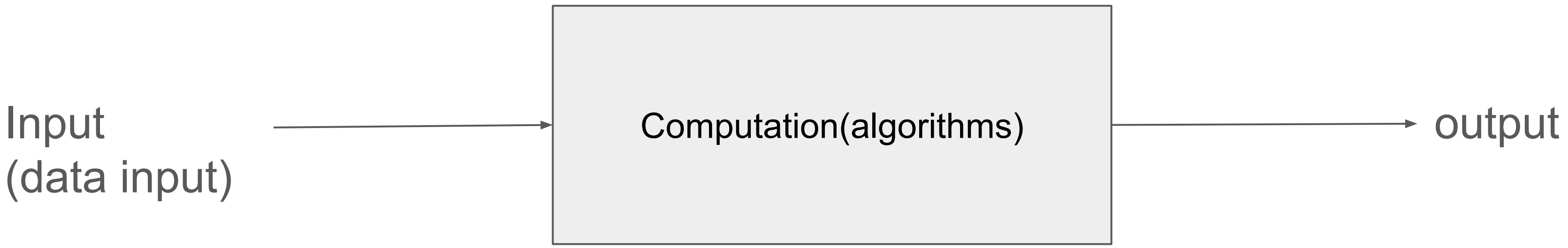
https://tldp.org/HOWTO/Assembly-HOWTO/hello.html

Hello world in C and Python

```c
#include <stdio.h>
int main(){
    printf("hello world!");
    return 0;
}
```

```python
print("Hello World!")
```

How to write more complex programs?



Algorithm

Set of steps that can be used to solve a problem!

It can be turned into a program easily!

How to go from school to home?

Specific steps

- Anyone who follows these steps can go home

What happens when there is ambiguity in steps?

- **The person who follow your steps is LOST!**

Example

How to find a student in university?

Total number of computations

- Number of logical operations
- Number of arithmetic operations

Depends on

- Number of steps
- Number of repetitions
- Number of input
  - n

1024
512
256
128
64
32
16
8
4
2
1

University has 30,000 students

Search 1 student

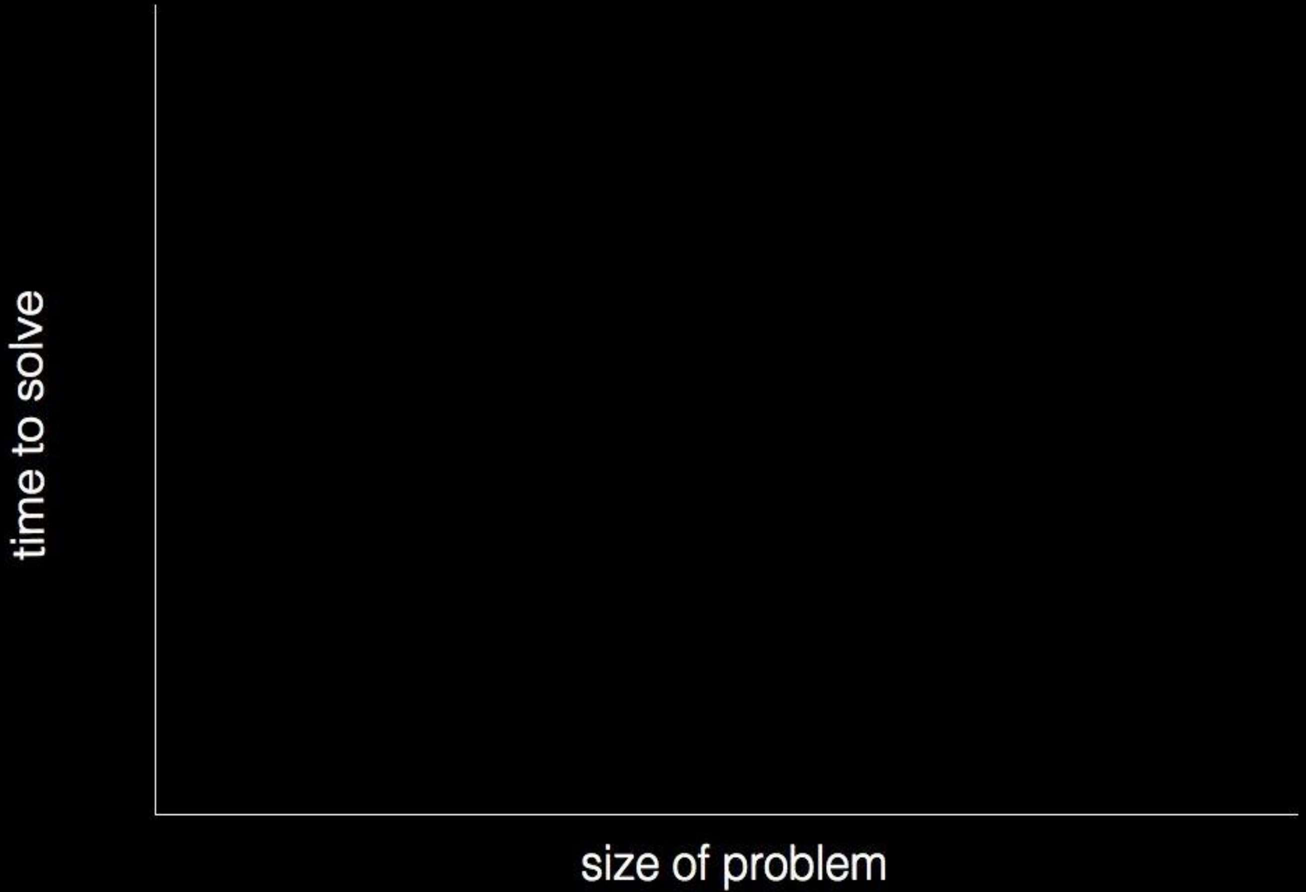
Input (Problem size)

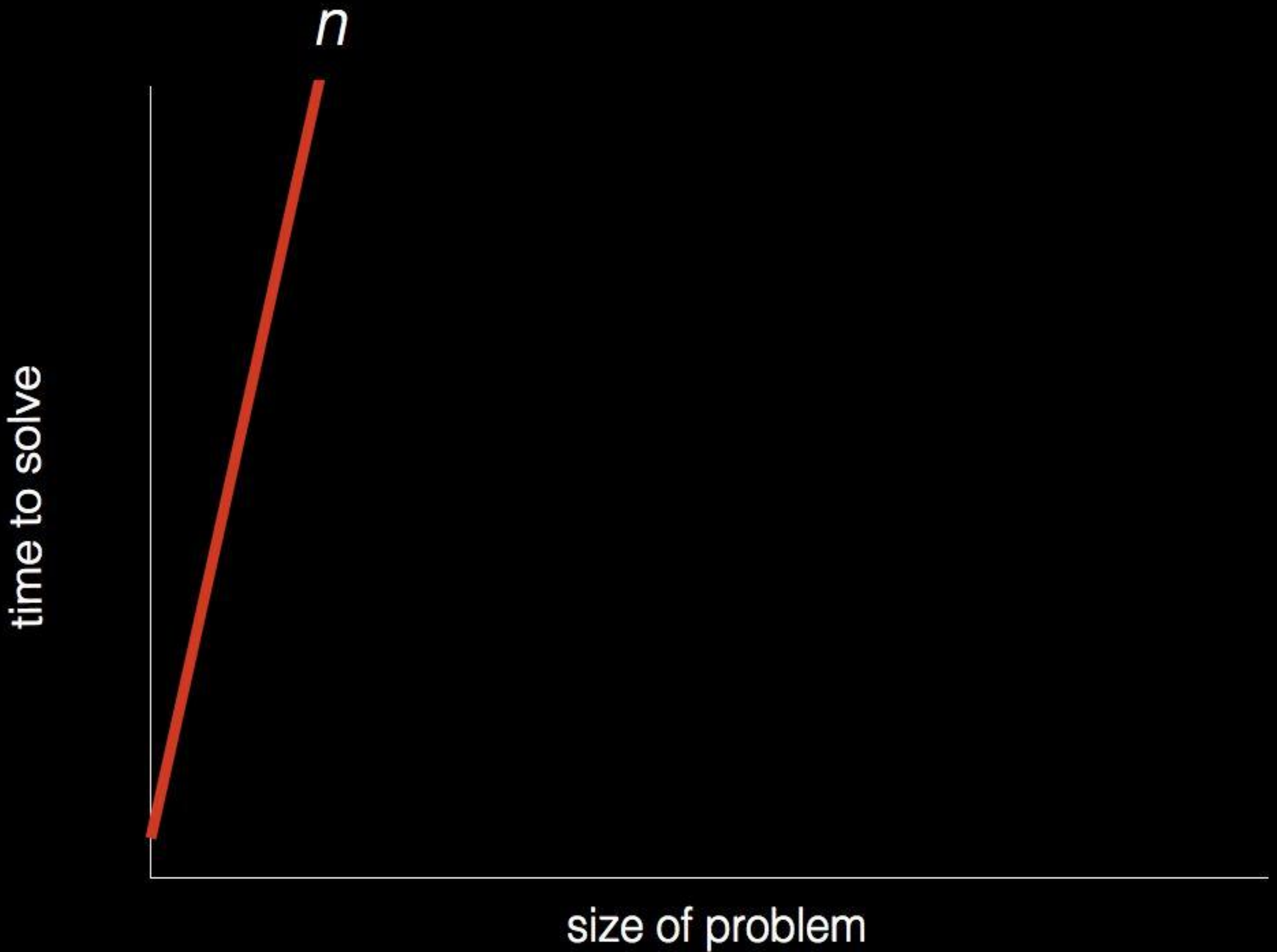
- 1 student
  - m = 1
- 30000 students
  - n = 30000

The number of computations in algorithm

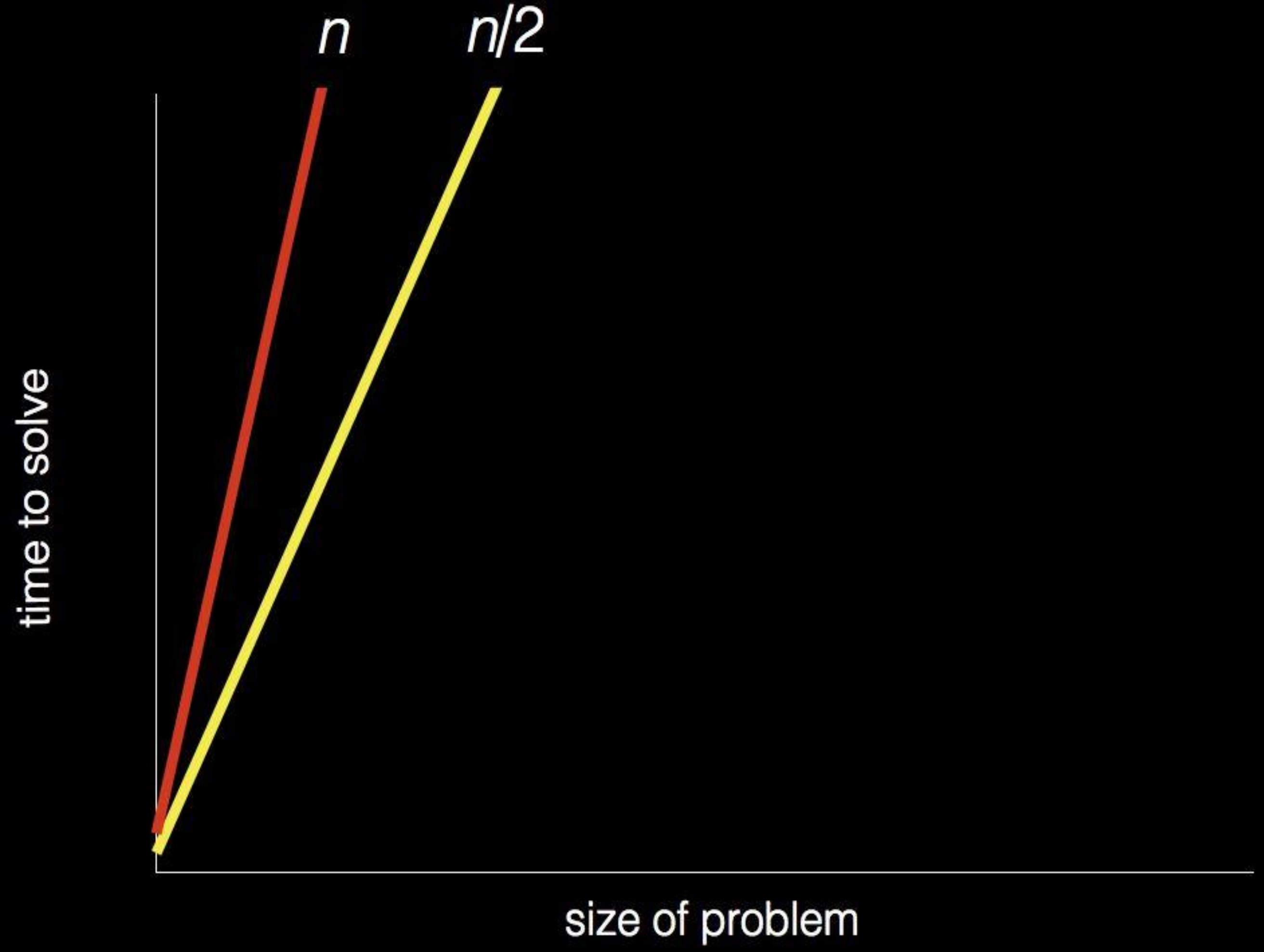
(Computational time to solve a problem)

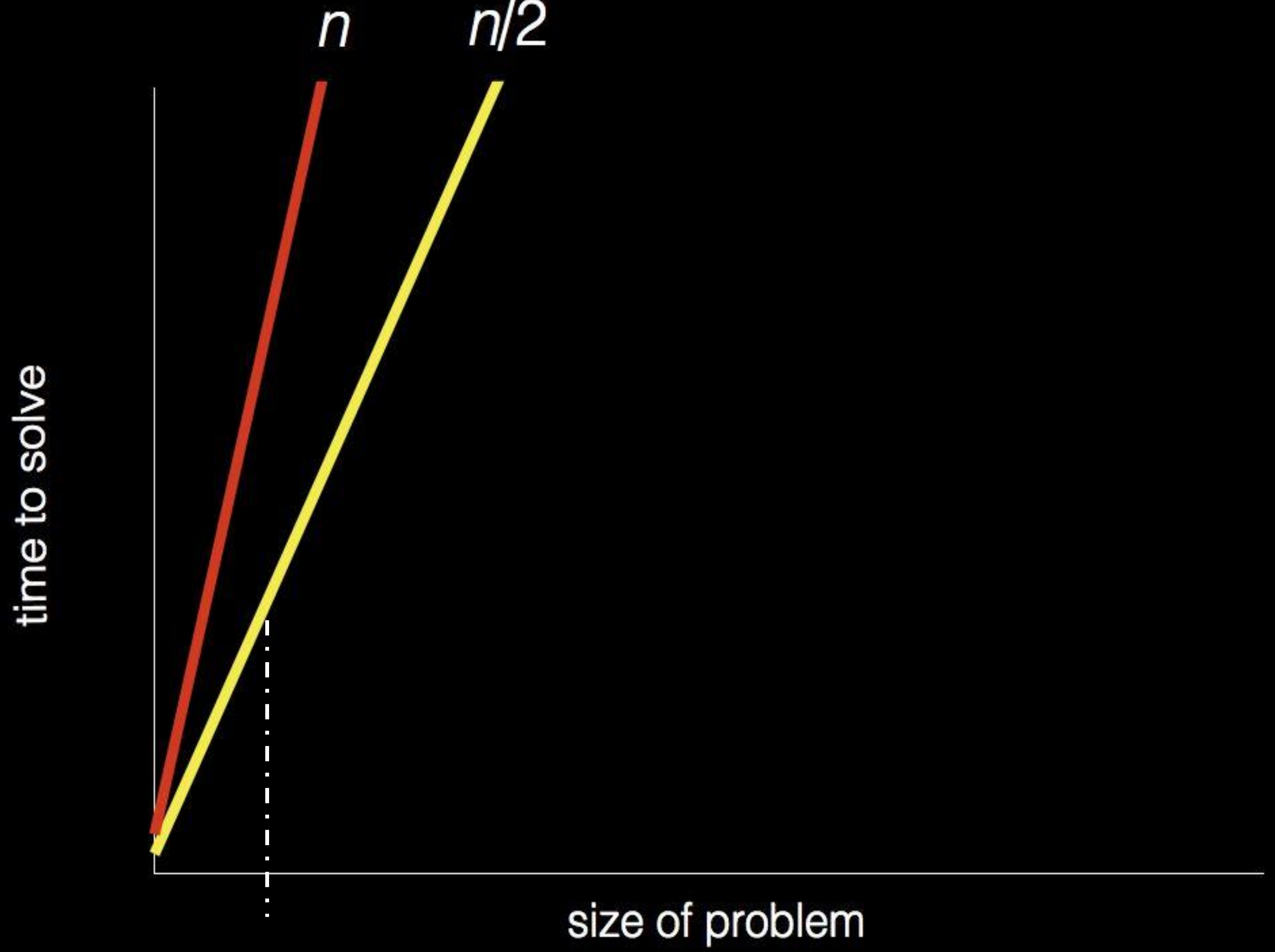
f(m,n) = ?

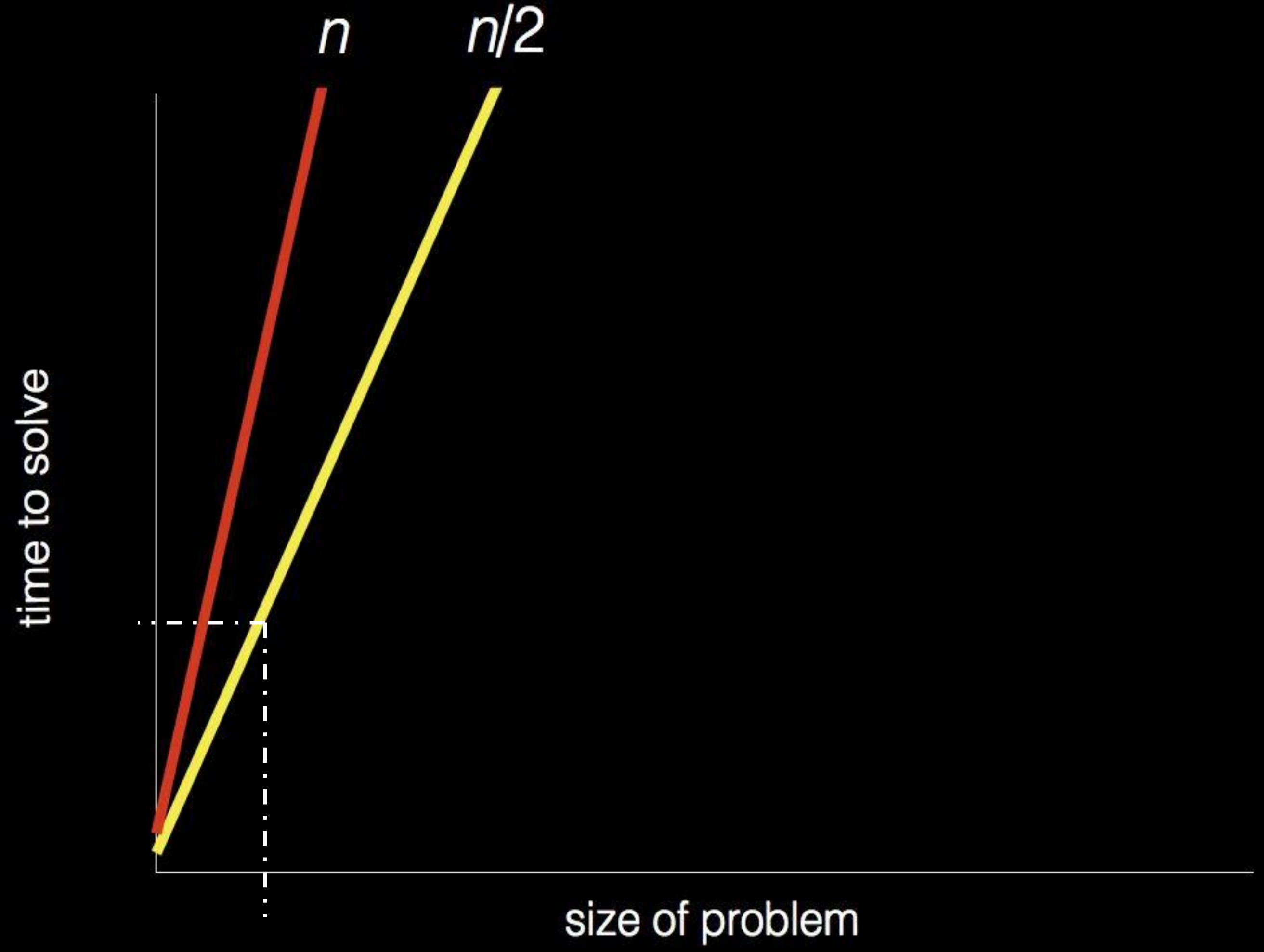

time to solve
size of problem

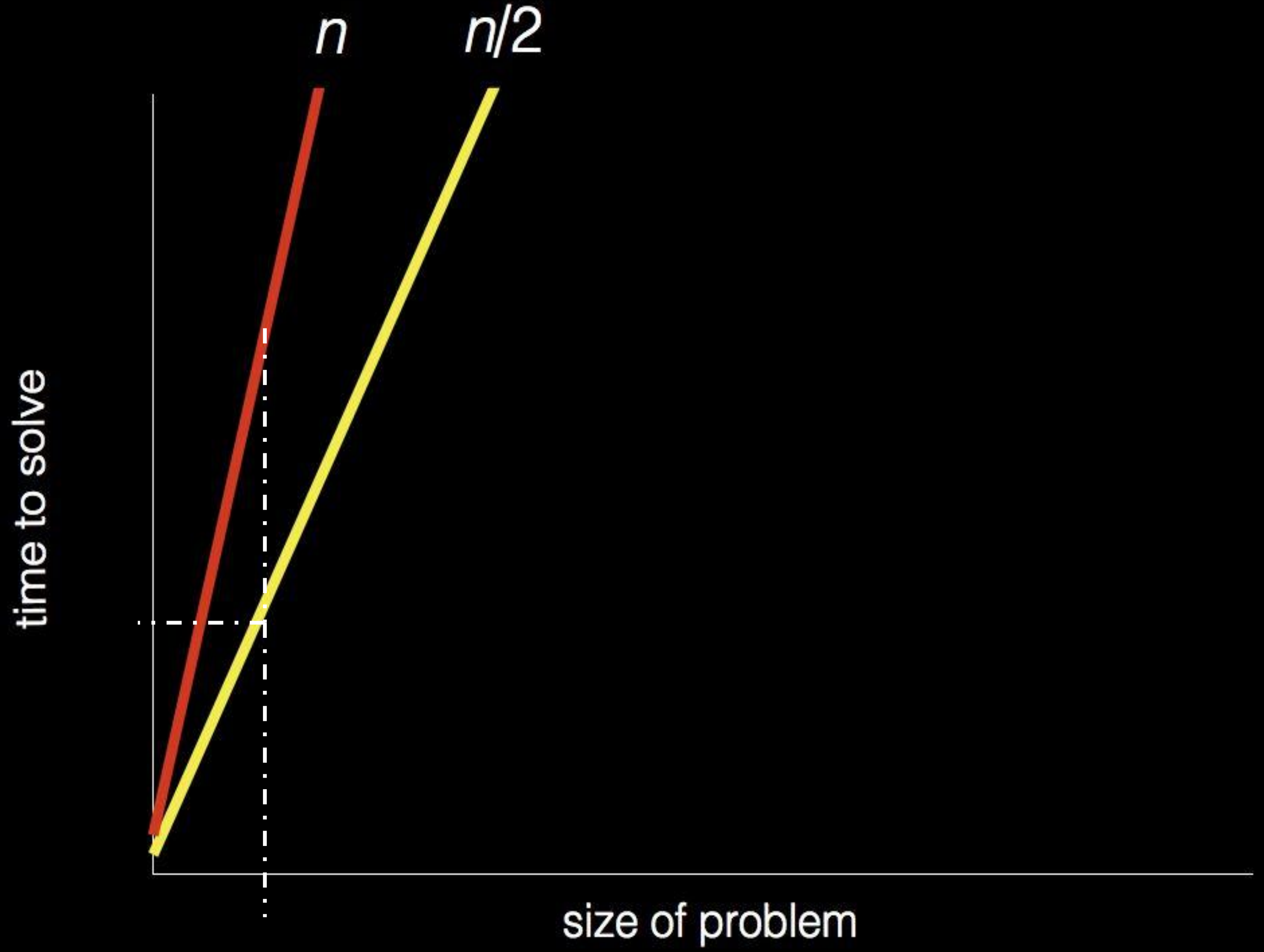


n
time to solve
size of problem

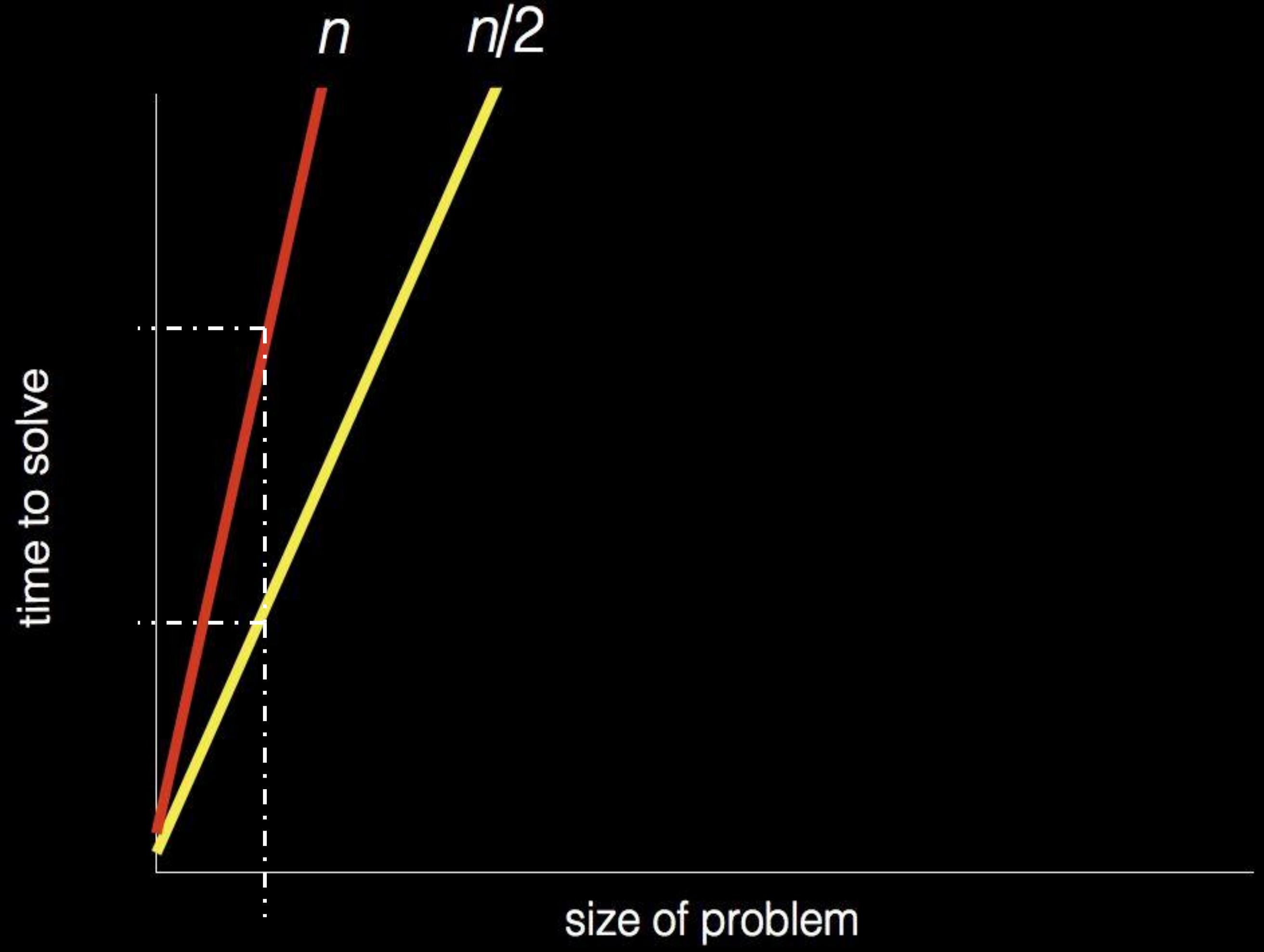

n
n/2
time to solve
size of problem

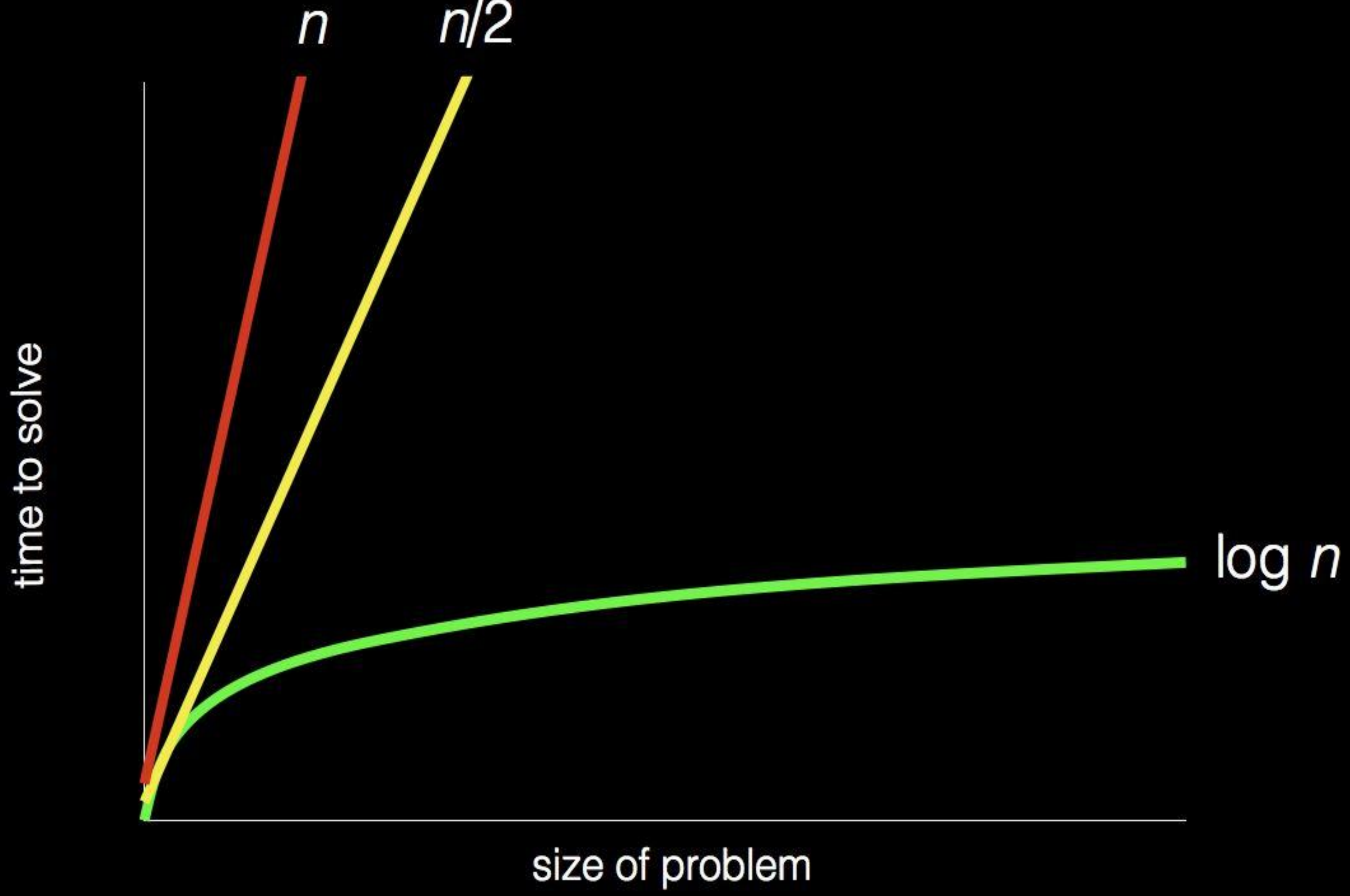

n
n/2
time to solve
size of problem


n
n/2
time to solve
size of problem



n
n/2
time to solve
size of problem


n
n/2
time to solve
size of problem


n
n/2
time to solve
log n
size of problem

Next week

Expression of algorithms

- Pseudocode
  - Going from algorithm to code
- Flow charts
  - Going from algorithm to code