Everything you ever *wanted / needed to know* about programming

Dominik Buchegger

Content / Disclaimer

- Theory
- Mock Up Questions
- Programming

This content corresponds to what I understand to be the core contents after reviewing the lecture slides and should only be understood as an overview.

This does not necessarily correspond to the opinion of the lecturer and does not claim to be complete, nor does it allow conclusions to be drawn about the quiz questions or the exam (which are both unknown to me).

IDE

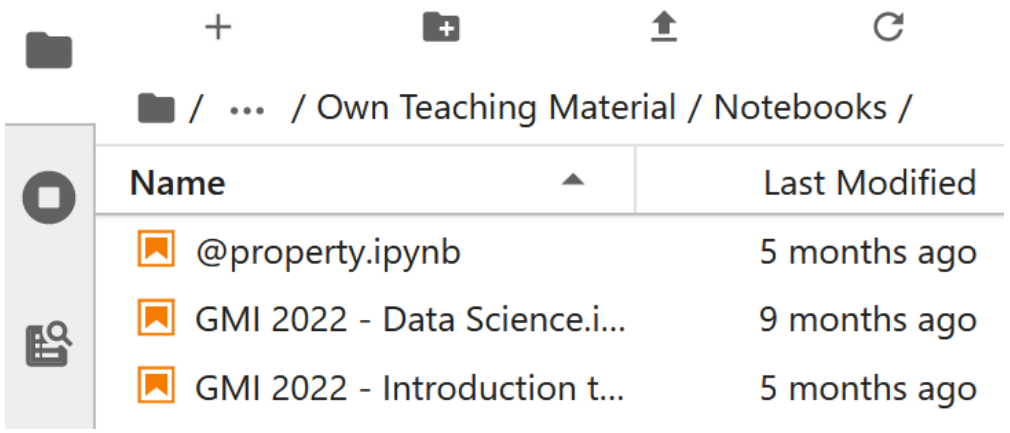
You can use an IDE of your choice as long as it supports Jupyter Notebooks (.ipynb).

We recommend: https://www.anaconda.com/products/distribution

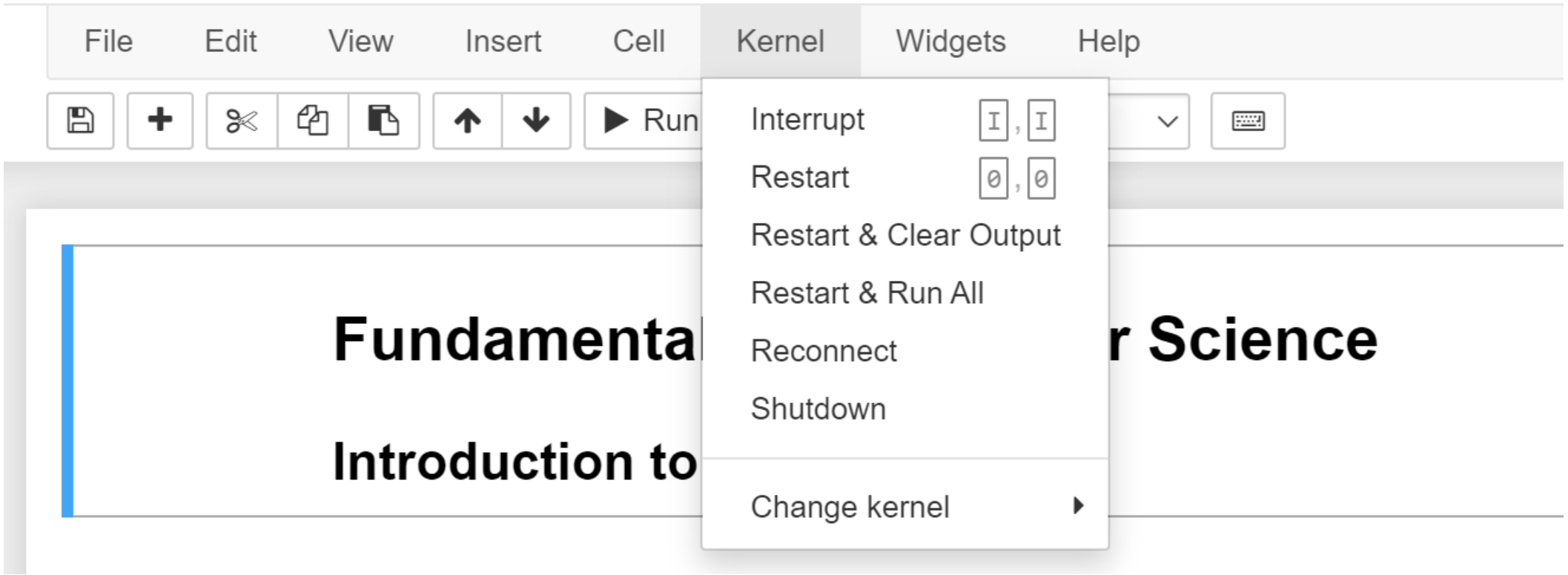
Open: ANACONDA.NAVIGATOR

Launch: lab JupyterLab

Open Notebook from the folder you stored it in:



My Jupyter Notebook does not run anymore



Lecture 1

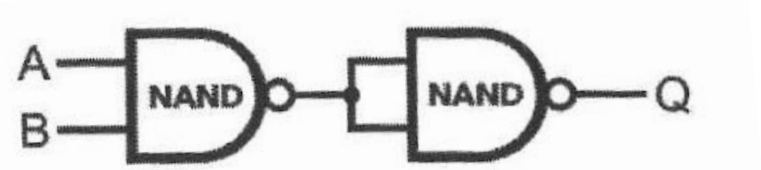
Bits (Binary Digits) & Bytes

- Information = Contextualized Bits (0 / 1) with Encoding
- For Data we talk about Bytes (normally 8 bits, numbers 0-255 [2^8])

Hour = 20
Minute = 55
Second = 59

Hour	10100	$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$	$16 + 0 + 4 + 0 + 0$
Minute	110111	$1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	$32 + 16 + 0 + 4 + 2 + 1$
Second	111011	$1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	$32 + 16 + 8 + 0 + 2 + 1$

Boolean Algebra & NAND Gates



<u>Boolean Algebra:</u>

Bsp. X=True=1, y=False=0

NOT ~x = 0;

AND (x & y) = 0;

OR (x | y) = 1;

XOR (x ^ y) = 1;

Transistors (electronical switches) can represent 0 & 1 (True & False)
→ in combination we can build all logical symbols

<u>Functional Completeness of NAND:</u>

NOT: A NAND A

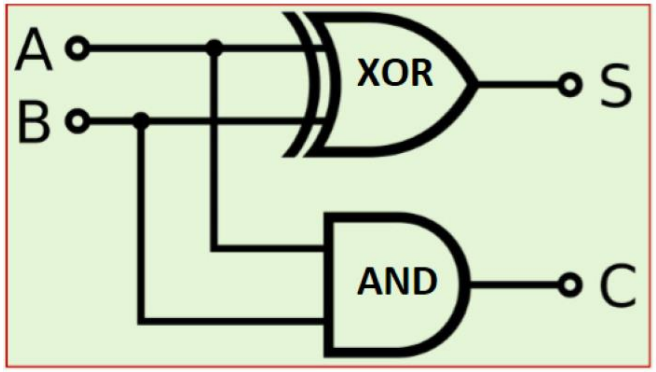
AND: (A NAND B) NAND (A NAND B)

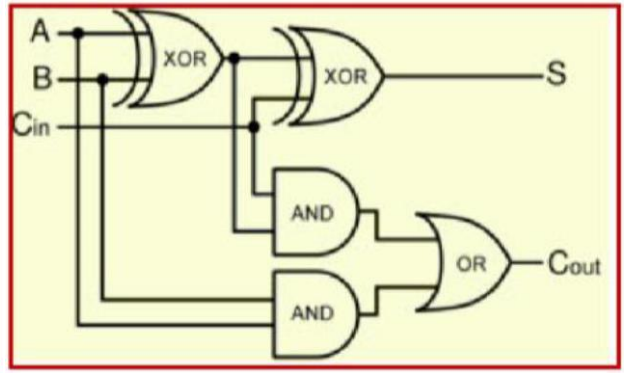
OR: (A NAND A) NAND (B NAND B)

XOR: (A NAND (A NAND B)) NAND (B NAND (A NAND B))

NOR: NOT(A OR B)

XNOR: NOT(A XOR B)





Half-adder & Full-adder

Arithmetic/Logic Unit (ALU) & Central Processing Unit (CPU)

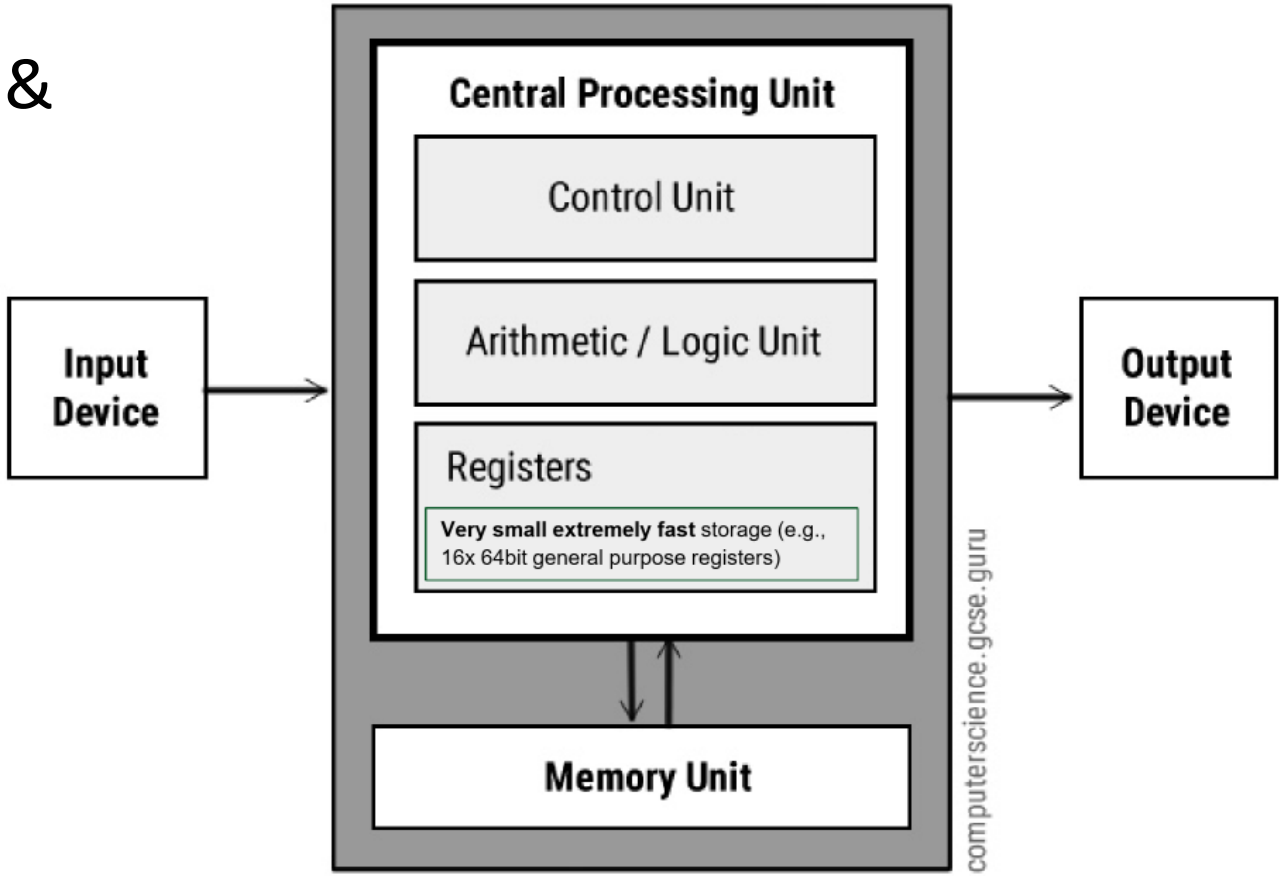
CPU runs in a loop. In each cycle the CPU uses instructions to load data in the registers and computes the result to store in the memory unit.

Adding, Multiplier, Substractor & Divisor Circuits (4 Units) **+**

Memory for Input/Output **+**

Function select bits

Everything is a binary string interpreted as executable instructions.



A simple processor

4 Memory Slots of size 4 bits each
2 Registers of size 4 bits each
An ALU that can subtract & add
Instruction Set

- **Load** from M[x] into R[y]: 000 xx y
- **Store** from R[x] into M[y]: 001 x yy
- **Add** the register contents and store the result in R[x]: 010 x
- **Subtract** the contents of R[0] from R[1] and store the result in R[x]: 011 x
- **Insert** the 4-bit number n into M[x] 100 nnnn xx

How many bits to address 4 memory slots?
How many bits to address 2 registers?
How many bits to encode 5 instructions?

Register Address	Register Content
0	
1	

Memory Address	Memory Content
00	
01	
10	
11	

Optimize the encoding
by omitting prefix 0

// Machine-level Program that adds the numbers 7 and 9

Insert 7 into M[00]:	100 0111 00
Insert 9 into M[01]:	100 1001 01
Load from M[00] to R[0]:	000 00 0
Load from M[01] to R[1]:	000 01 1
Add registers into R[0]:	010 0
Store R[0] into M[10]	001 0 10

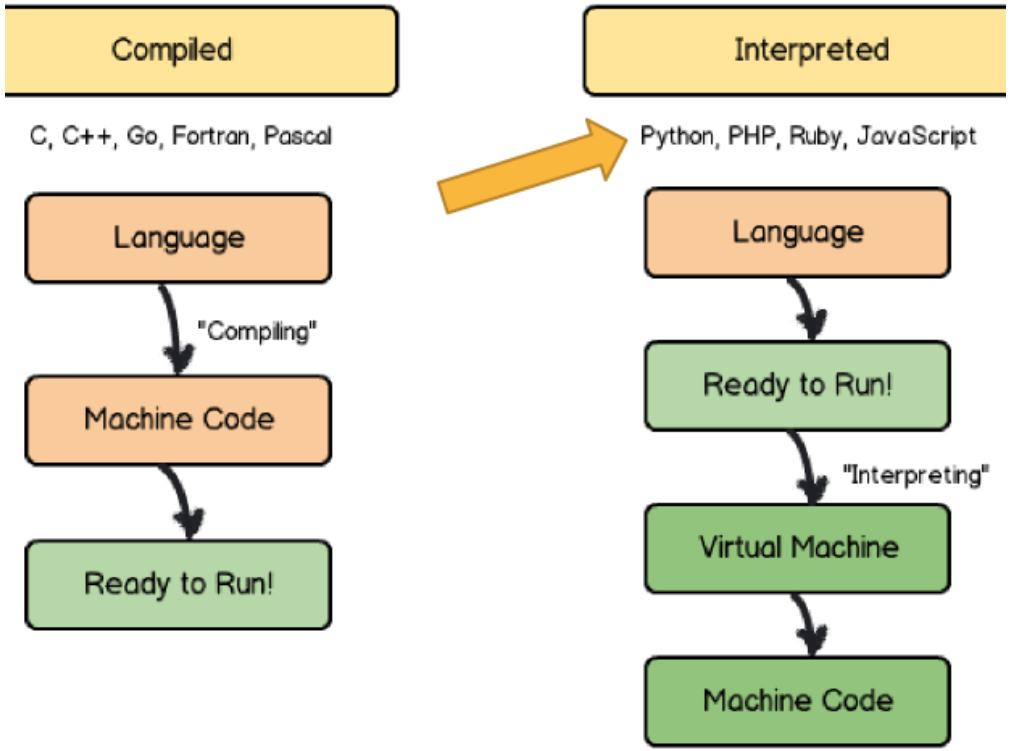
Note that the encoding of this machine-level program is unique!

We can thus omit the formatting and **write the program as**:

`1000111001001001010000000001101000010100`

Let's try it!

Compiled vs Interpreted Programming Languages



Compiled languages translate & validate the program before running, while interpreted languages are directly ready.

Python vs C++

#1

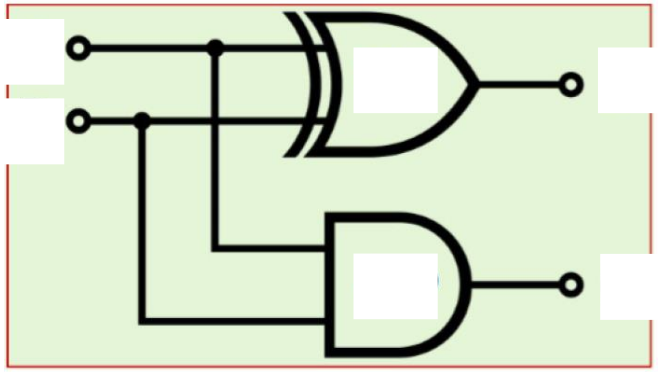
Advantages :
- Easy to learn
- Easy to access libraries
- Scientific community sharing (open source, many libraries)

Disadvantages:
- Slow
- Interpreted (dependencies)
- Not typed (errors at runtime)

Advantages :
- Execution speed
- Pre-Compiled (exe on machine)
- Typed (well defined)
- Modern professional libraries

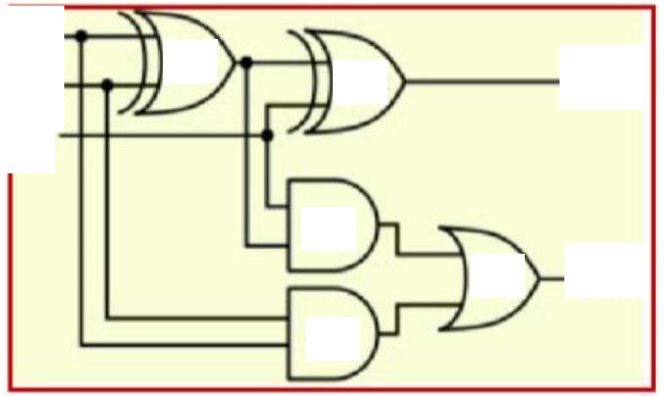
Disadvantages:
- Learning curve
- Harder to acces libraries (less sharing than in Python)

Mock Questions



Read the time:

Complete the drawing



Name the operator

(A NAND A) NAND

(B NAND B)

Name one advantage & disadvantage of interpreted languages (e.g. Python)

What does this program do?

```
10001110010010010100000000001101000010000
```

(with given instructions)