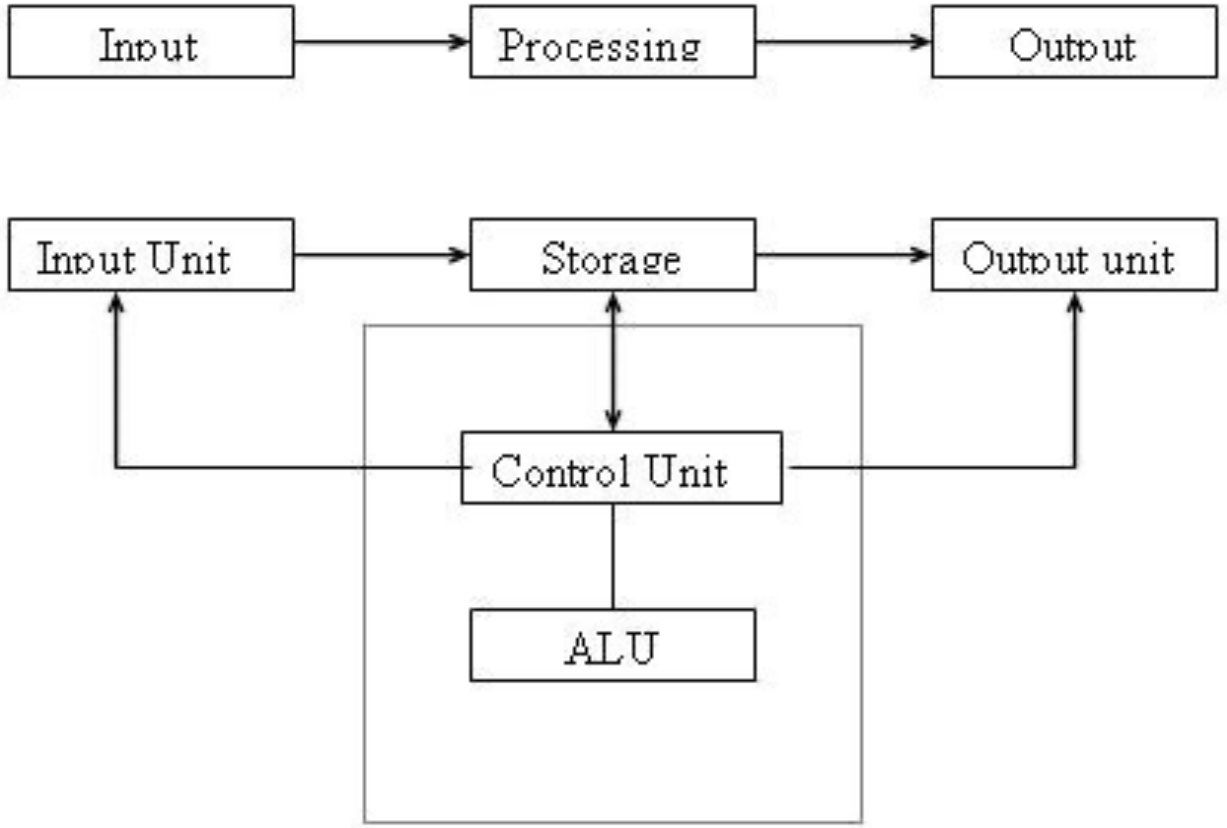
Introduction to Programming

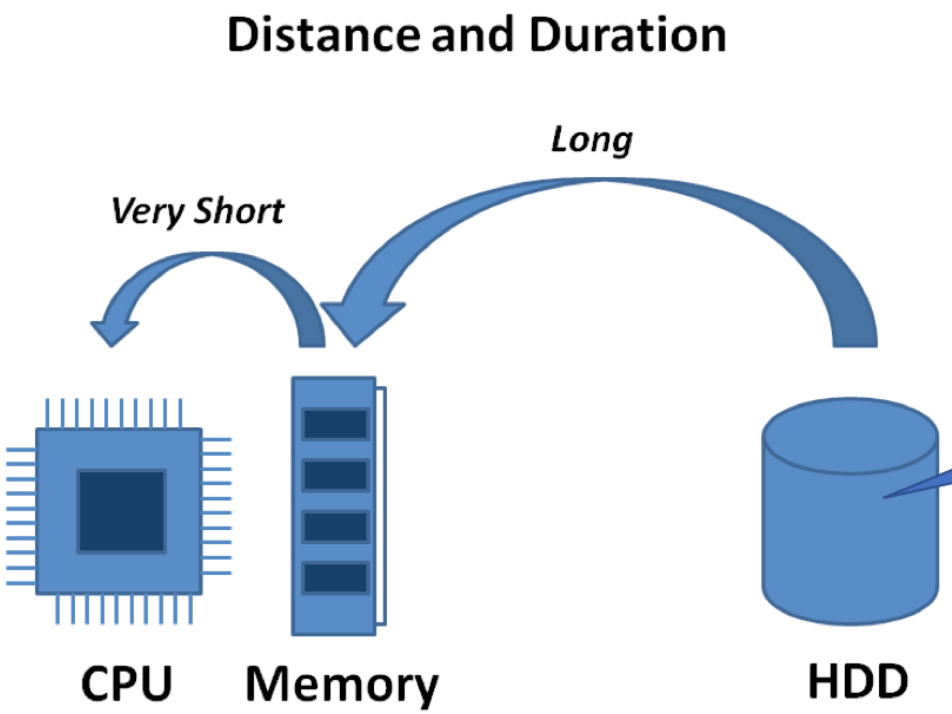
Computational Approaches to Problem Solving

What Does a Computer Do?

- Three things:
  - Takes inputs
  - Performs calculations
  - Remembers the results
- Extremely fast
  - A15s in iPhone 13 Pro runs at 3.32GHz, has 6 cores. Its performance is 1.5 TFLOPS: 1.5 trillion float number calculations per second.
  - The speed of light is about 300,000 km/s (1 ns per foot).
  - 4,500 instructions when photons travel from your screen to your eyes (about 3 feet distance).



How Computer Works

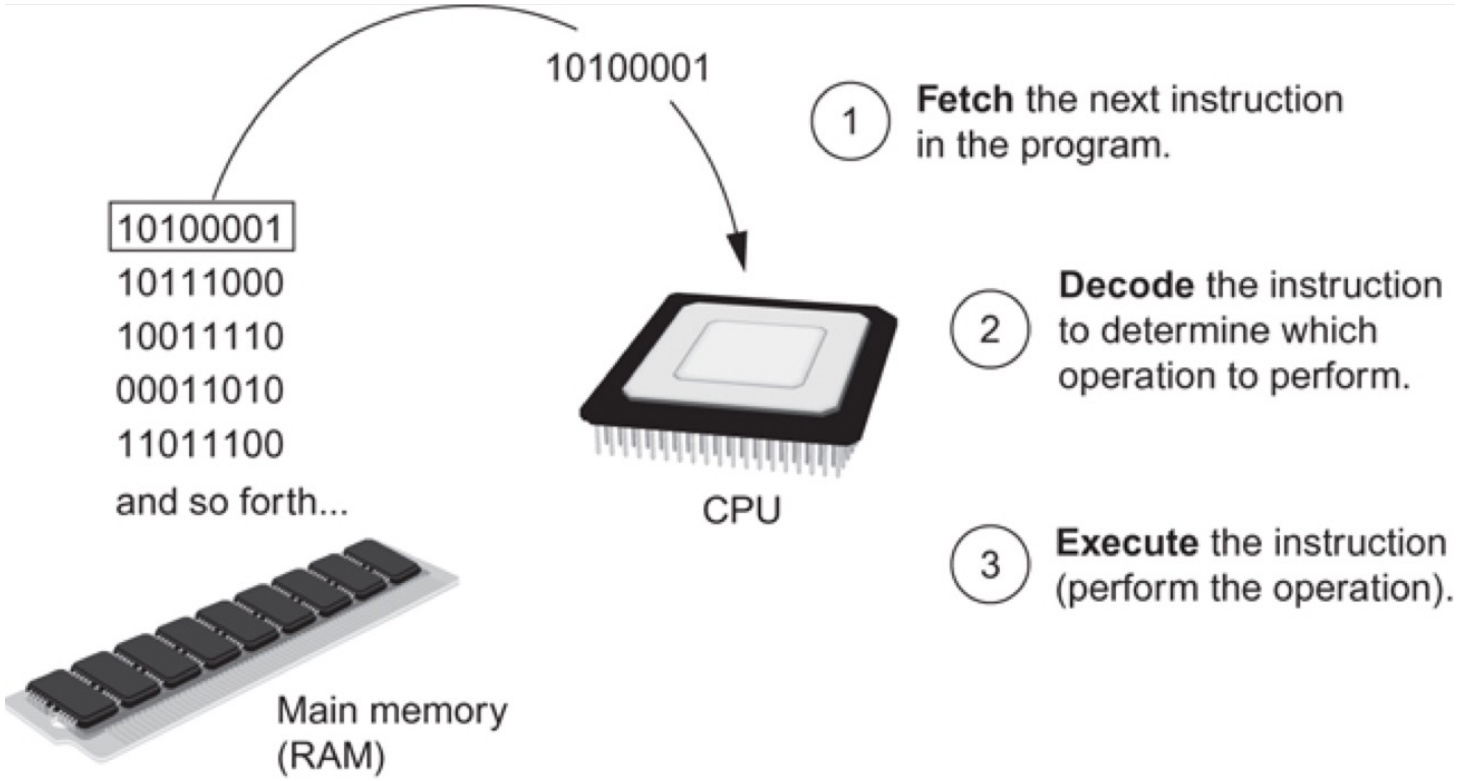


Applications (programs)
- source code files
- machine code(compiled) files

Data files
- Text
- multimedia

Program Execution

- Fetch: read the next instruction from memory into CPU
- Decode: CPU decodes fetched instruction to determine which operation to perform
- Execute: perform the operation



Assembly & Machine Languages

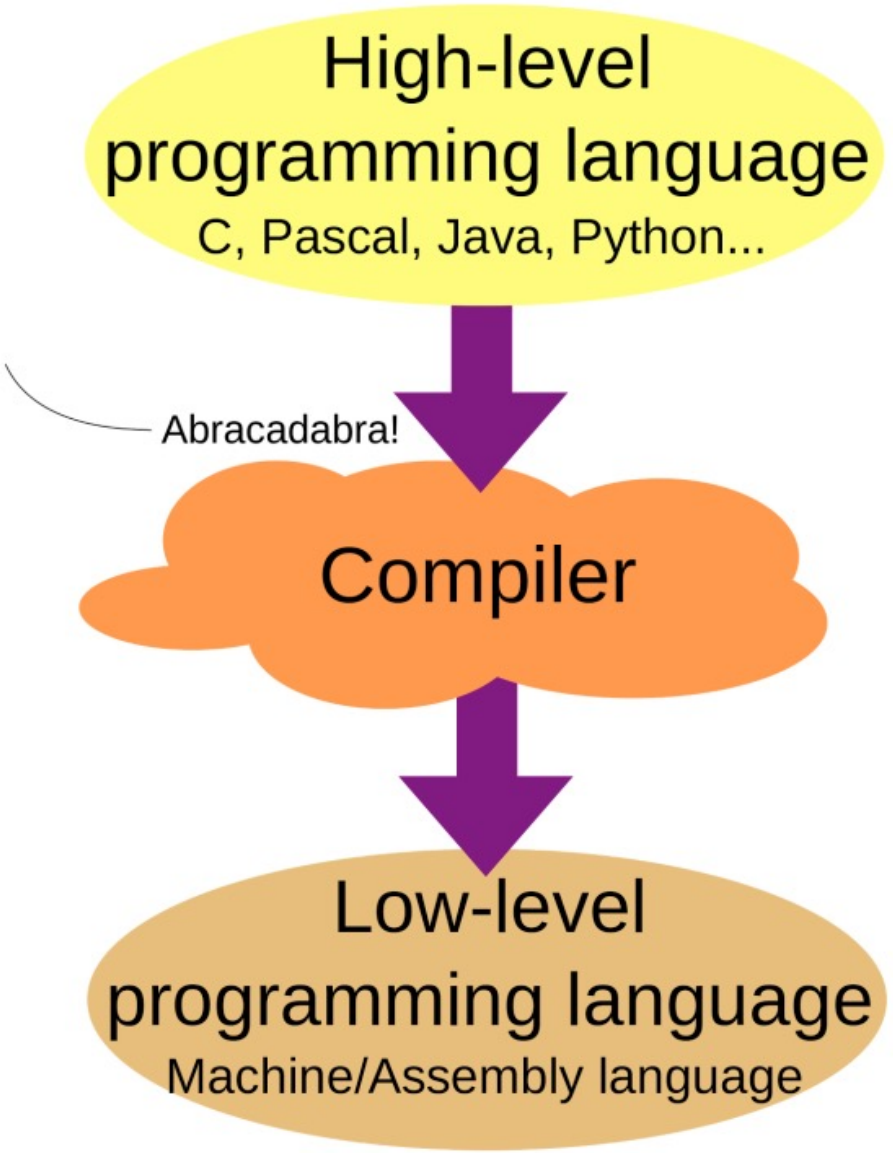
- Computers can only understand and execute machine languages.
- Machine languages are impractical for people to write and understand.
- Computer scientists invented assembly languages that use short words (mnemonics) for instructions instead of binary numbers.
- There is a so-called **Assembler** to translate an assembly language to a machine language for execution by CPU.

Assembly & Machine Language

Assembly Language	*Machine Language*
`ST 1,[801]`	`00100101 11010011`
`ST 0,[802]`	`00100100 11010100`
`TOP: BEQ [802],10,BOT`	`10001010 01001001 11110000`
`INCR [802]`	`01000100 01010100`
`MUL [801],2,[803]`	`01001000 10100111 10100011`
`ST [803],[801]`	`11100101 10101011 00000010`
`JMP TOP`	`00101001`
`BOT: LD A,[801]`	`11010101`
`CALL PRINT`	`11010100 10101000`
	`10010001 01000100`

High-Level Programming Languages

- Assembly languages are close in nature to machine language. Both are low-level languages because they have relatively simple instructions.
- High-Level languages are invented to allow simple creation of powerful and complex programs.



High-Level PL

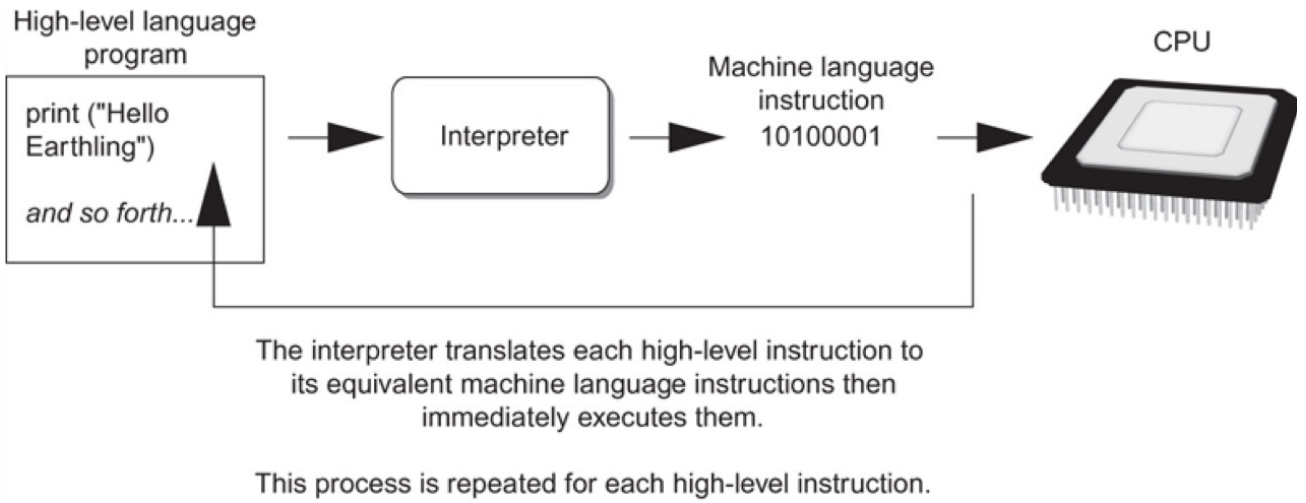
- A PL with strong abstraction from the details of the computer. (https://en.wikipedia.org/wiki/High-level_programming_language)
- It may
  - Use natural language elements
  - Automate/hide areas of computing systems
  - Handle data type automatically (dynamic type)
  - Specify

Four Levels

- Low level: see hw details such as CPU registers (assembly PL)
- System level: less hw details but still deal with memory management (system PL C, C++, Rust)
- General PL: no hw details, very little memory management
  - Static data type: explicit data type such as int, float (Java, C#, Swift, Go)
  - Dynamic data type: Python, Perl, PHP
- Domain-specific PL: no hw and memory details
  - Prolog: logic
  - 4GL: SQL

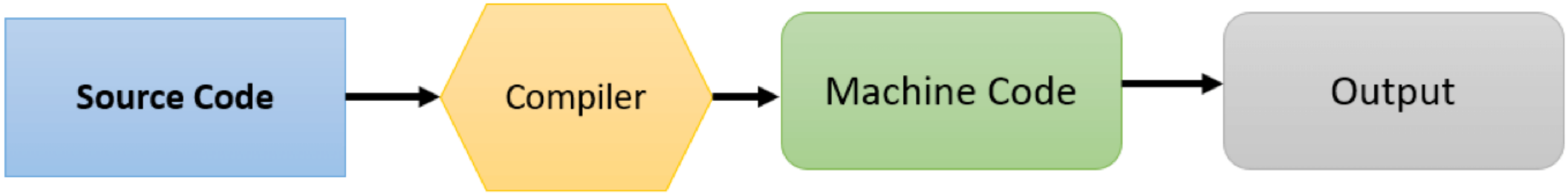
Interpreter

- An interpreter is a ”virtual machine (engine, runtime)" that translates and executes instructions in high-level language program such as Python.
- It interprets and executes one instruction in a real computer at a time.



Compiler vs Interpreter

How Compiler Works

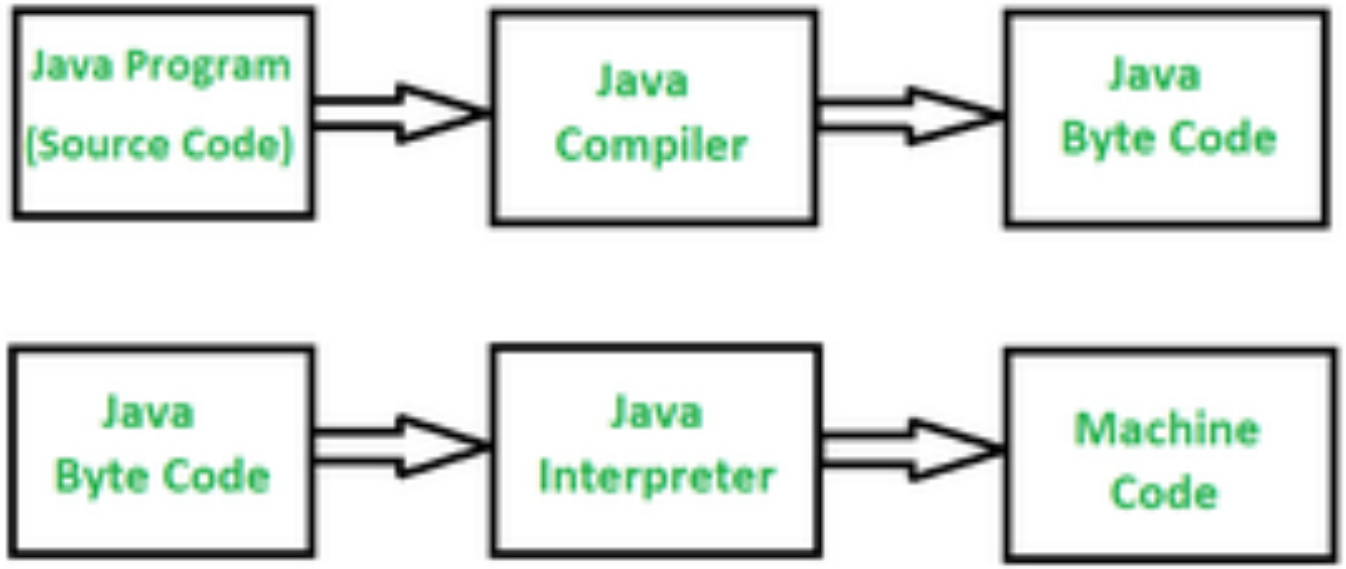


How Interpreter Works

Differences

- all vs one statement a step: a compiler compiles all instructions together. An interpreter interprets one instruction/statement at one time.
- executable files or not: A compiler produces executable machine code files from source code files. An interpreter doesn't produce any executable files from source code file.
- what to run: for a compiler, a user runs the executable file (program/application) in a computer or a virtual machine. An interpreter interpreters and executes a statement at a time directly from source code.

The Boundary Is Not Clear



Programming Language Concepts

- Syntax: rules to be followed when writing program.
  - Example: (+ 2 3) vs 2 + 3
- Semantics: the meaning
  - Rules define which syntactically valid strings have a meaning. For example: “Cats drink cars".
  - In code: 3 / "abc"
  - Dynamically typed languages don't check this until they run the code.

Program Errors

- Types
  - Syntactic error: code violates the syntactic rules. For example: `3 +- 5`.
  - Semantic error: the program has as unintended meaning. For example, you calculate the sum of two numbers by `n1 * n2`.
- What might happen if a program has an error
  - Syntactic error: This is a "Good" error because compiler/interpreter refuse to run the code. The program fails fast.
  - Semantic error: this is "Bad" error with bad consequences:
    - The program crashes
    - The program runs forever
    - The program appears normal but might give wrong results. This might cause big damages.

Elements of a Program

A program is a set of instructions written in a specific programming language.

- Expression: a single value such as 3 or operations such as 3 + 5 that produce a value.
- Statement: individual instruction that use expression(s) to perform a task. A statement doesn't produce a value.
  - For example: print(3 + 5), if … else…, while … do …
- Source code: statements written by programmer.

Turing Completeness

- A **Universal Turing Machine** can be programmed to compute any computable functions.
- A program language is **Turing complete** if it can be used to simulate a universal Turing machine. It sounds scary but it is simple in practice:
  - can read and write data/variables.
  - supports branches: if ... else ....
  - supports loops: do ... until....
- Most programming languages are Turing complete. But there are languages that are not. For example, Bitcoin script is not Turing complete, why?

Python is a ___ programming language

- high-level: not deal with memory and computer hardware details.
- Interpreted/script: source code is executed directly by Python interpreter.
- dynamically typed: check operations at runtime.
- Turing complete: variable, branch, loop