

INTERPRETATION VS. COMPILATION

Computer Science III

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Lecturer
Computer Engineer
School of Engineering
Universidad Distrital Francisco José de Caldas

2024-III



Programming Languages

32-bit (4-byte) **ADD** instruction:

100000	00 100	00010	00011	000000000000
opcode	rc	ra	rb	(unused)

Could be something like: $\text{Reg}[4] \leftarrow \text{Reg}[2] + \text{Reg}[3]$

In assembly:

```
1  ADD(R2, R3, R4)
```

In any high-level language:

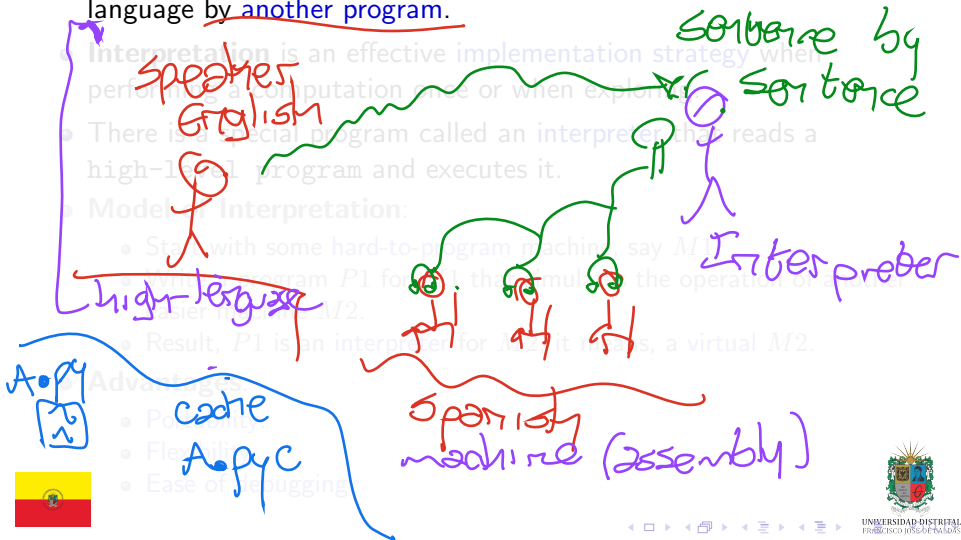
```
1  a = b + c;
```

← Performance



Interpretation

- **Interpretation** is the process of executing a program in a high-level language by another program.



Interpretation

- **Interpretation** is the process of **executing** a program in a high-level language by **another program**.
- **Interpretation** is an effective **implementation strategy** when performing a computation **once** or **when exploring**.

- There is a special program called an **interpreter** that reads a high-level program and executes it.

- **Model of Interpretation:**

- Start with some hard-to-program machine, say $M1$.
- Write a program $P1$ for $M1$ that simulates the operation of another machine $M2$.
- Result, $P1$ is an interpreter for $M2$, it means, a virtual $M2$.

- **Advantages:**

- Portability.
- Flexibility.
- Ease of debugging.

→ step 1

→ step 2

Python

cache C-

2-∞



Interpretation

- **Interpretation** is the process of **executing** a program in a high-level language by **another program**.
- **Interpretation** is an effective **implementation strategy** when performing a computation once or when exploring.
- There is a special program called an interpreter that reads a high-level program and executes it.

- **Model of Interpretation:**

- Start with some **hard-to-program machine**, say $M1$.
- Write a program $P1$ for $M1$ that performs the operation of another easier machine $M2$.
- Result, $P1$ is an interpreter for $M2$, it means a virtual $M2$.

- **Advantages:**

- Portability.
- Ease of debugging.

Python
Natural Language
Days 5
Interpreter

Ref. 0.1ms ✓
Mem. 20Mb
~~Days 50~~



Interpretation

- **Interpretation** is the process of **executing** a program in a high-level language by **another program**.
- **Interpretation** is an effective **implementation strategy** when performing a computation once or when exploring.
- There is a special program called an **interpreter** that reads a high-level program and executes it.
- **Model of Interpretation:**
 - Start with some **hard-to-program** machine, say **M1**.
 - Write a program **P1** for **M1** that simulates the operation of another easier machine **M2**. *Python*
 - Result, **P1** is an **interpreter** for **M2**, it means, a **virtual M2**.

Advantages:

- weakly-typed
 - Portability
 - Flexibility
 - Ease of debugging.
- Python*
- x = 5 4 bytes*
x = "hello" 5 bytes
x = False 1 byte



Interpretation

90's → today
diferentes s, mismo concepto
s, no close on
have

- **Interpretation** is the process of **executing** a program in a high-level language by **another program**.
- **Interpretation** is an effective **implementation strategy** when performing a computation once or when exploring.
- There is a special program called an **interpreter** that reads a high-level program and executes it. *Anding the party*
- **Model of Interpretation:** *english better programming language*
 - Start with some **hard-to-program** machine, say $M1$.
 - Write a program $P1$ for $M1$ that simulates the operation of another easier machine $M2$.
 - Result, $P1$ is an **interpreter** for $M2$, it means, a **virtual** $M2$.

Advantages:

- Portability.
- Flexibility.
- Ease of debugging.

↳ Run it any platform
→ easy to change/maintain
↳ execute in seconds
↳ ready if finish else hungry



Compilation

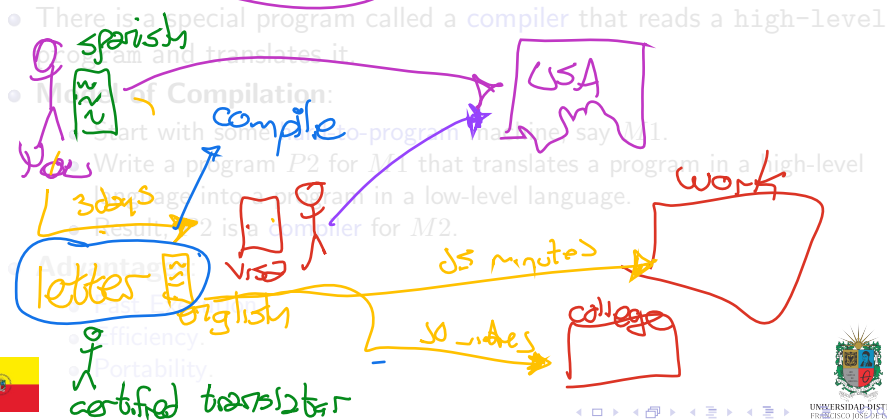
- **Compilation** is the process of translating a program in a high-level language into a low-level language.
- **Compilation** is an effective implementation strategy when performing a computation many times.
- There is a special program called a compiler that reads a high-level program and translates it.
- **Model of Compilation:**
 - Start with some hard-to-program machine, say $M1$.
 - Write a program $P2$ for $M1$ that translates a program in a high-level language into a program in a low-level language.
 - Result, $P2$ is a compiler for $M2$.
- **Advantages:**
 - Fast Execution.
 - Efficiency.
 - Portability.

C → Assembly → Machine



Compilation

- **Compilation** is the process of **translating** a program in a high-level language into a **low-level language**.
- **Compilation** is an effective **implementation strategy** when performing a computation **many times**.

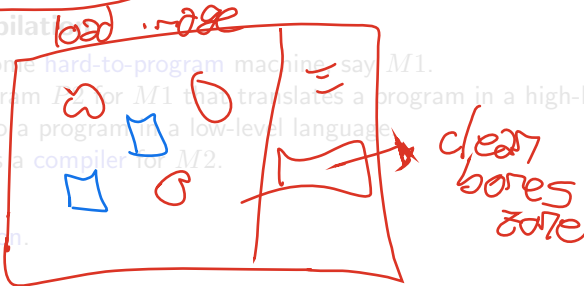


Compilation

- **Compilation** is the process of **translating** a program in a high-level language into a **low-level language**.
- **Compilation** is an effective **implementation strategy** when performing a computation many times.
- There is a special program called a **compiler** that reads a high-level program and translates it.

Model of Compilation

- Start with some **hard-to-program machine**, say $M1$.
- Write a program $P2$ for $M1$ that translates a program in a high-level language into a program in a low-level language.
- Result, $P2$ is a **compiler** for $M2$.
- **Advantages:**
 - Fast Execution.
 - Efficiency.
 - Portability.



Compilation

- **Compilation** is the process of **translating** a program in a high-level language into a **low-level language**.
- **Compilation** is an effective **implementation strategy** when performing a computation many times.
- There is a special program called a **compiler** that reads a high-level program and translates it.
- **Model of Compilation:**
 - Start with some hard-to-program machine, say $M1$.
 - Write a program $P2$ for $M1$ that translates a program in a high-level language into a program in a low-level language.
 - Result, $P2$ is a **compiler** for $M2$.
- **Advantages:**
 - Fast Execution.
 - Efficiency.
 - Portability.



Compilation

- **Compilation** is the process of **translating** a program in a high-level language into a **low-level language**.
- **Compilation** is an effective **implementation strategy** when performing a computation many times.
- There is a special program called a **compiler** that reads a high-level program and translates it.
- **Model of Compilation:**
 - Start with some **hard-to-program** machine, say $M1$.
 - Write a program $P2$ for $M1$ that translates a program in a high-level language into a program in a low-level language.
 - Result, $P2$ is a **compiler** for $M2$.
- **Advantages:**
 - Fast Execution.
 - Efficiency.
 - Portability.

strongly typed



Interpretation Vs. Compilation

Characteristics differences:

	Compilation	Interpretation
How does it treat input $x + 2$?	Generate a program that computes $x + 2$	Computes $x + 2$
When it happens?	Before Execution	During Execution
What it complicates/slows?	Program Development	Program Execution
Decisions made at	Compile Time	Run Time

2 steps

Compile & Run



Thanks!

Questions?



Repo: <https://github.com/EngAndres/ud-public/tree/main/courses/computer-science-iii>



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE PAZ