Dasar Pemrograman Komputer

Agenda

- High Level Lang. Vs Low Level Lang.
- Compiled Lang Vs Interpreted Lang.
- Compilation Process
- Interpreter Process
- Class operation

- A program makes a computer usable. Without a program, a computer, even the most powerful one, is nothing more than an object. Similarly, without a player, a piano is nothing more than a wooden box.

- A language is a means (and a tool) for expressing and recording thoughts. There are many languages all around us. Some of them require neither speaking nor writing, such as body language; it's possible to express your deepest feelings very precisely without saying a word.

- Another language you use each day is your mother tongue, which you use to manifest your will and to think about reality. Computers have their own language, too, called machine language, which is very rudimentary.

- A computer, even the most technically sophisticated, is devoid of even a trace of intelligence. You could say that it is like a well-trained dog - it responds only to a predetermined set of known commands.

- A complete set of known commands is called an instruction list, sometimes abbreviated to IL. Different types of computers may vary depending on the size of their ILs, and the instructions could be completely different in different models.

- Note: machine languages are developed by humans.

Programming Language

- A formal language, which comprises a set of instructions that produce various kinds of output.
- Programming languages are used in computer programming to implement specific algorithms.
- Most programming languages consist of instructions for computers.

Computer Program

- A computer program - Is a collection of instructions that performs a specific task when executed by a computer.
- Most computer devices require programs to function properly.
- A computer program is usually written by a computer programmer and can be written in either high or low-level languages, depending on the task and the hardware being used.

Elements of Language

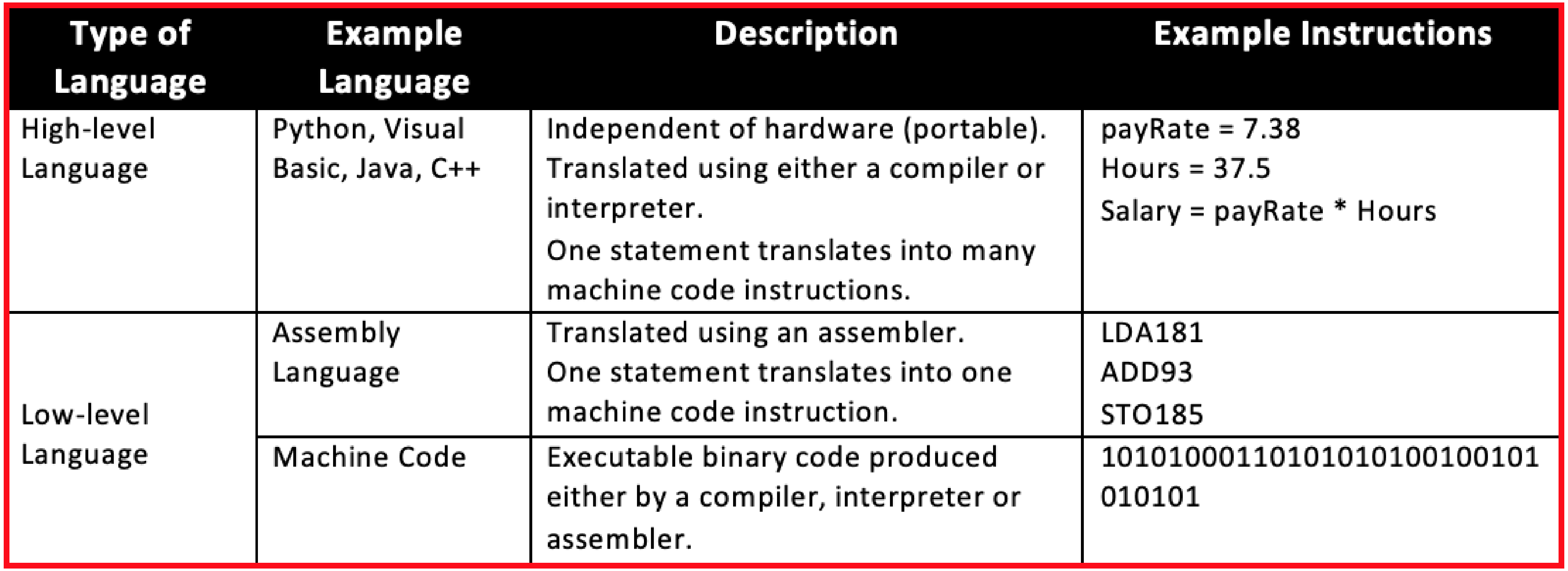
We can say that each language (machine or natural, it doesn't matter) consists of the following elements:

- **an alphabet**: a set of symbols used to build words of a certain language (e.g., the Latin alphabet for English, the Cyrillic alphabet for Russian, Kanji for Japanese, and so on)
- **a lexis**: (aka a dictionary) a set of words the language offers its users (e.g., the word "computer" comes from the English language dictionary, while "cmoptrue" doesn't; the word "chat" is present both in English and French dictionaries, but their meanings are different)
- **a syntax**: a set of rules (formal or informal, written or felt intuitively) used to determine if a certain string of words forms a valid sentence (e.g., "I am a python" is a syntactically correct phrase, while "I a python am" isn't)
- **semantics**: a set of rules determining if a certain phrase makes sense (e.g., "I ate a doughnut" makes sense, but "A doughnut ate me" doesn't)

Programming Lang. Level

- When we think about computer programmers, we probably think about people who write in high-level language.
- Most computer programming languages are written in a high-level programming language. They use the common English language to help make the code more understandable and to speed up the process of writing and debugging programs.
- Computers, however, use their own language written using binary called Machine code. This is known as a low-level language.

Programming Lang. Level



Type of Language	Example Language	Description	Example Instructions

High-level Language	Python, Visual Basic, Java, C++	Independent of hardware (portable). Translated using either a compiler or interpreter.
		One statement translates into many machine code instructions.
			payRate = 7.38
			Hours = 37.5
			Salary = payRate * Hours

Low-level Language	Assembly Language	Translated using an assembler. One statement translates into one machine code instruction.	LDA181 ADD93 STO185
	Machine Code	Executable binary code produced either by a compiler, interpreter or assembler.	10101000110101010100100101 010101

High-level Languages

- **Programming languages** such as *Python, Visual Basic, Java, SQL* and *C++* are all classed as high-level languages as they have been **developed** to make it **easier** for **programmers** to **read** and **write programs**. This is because **high-level language code** looks more like *normal human languages*.
- The code details how a problem is to be solved rather than giving instructions on how the computer will provide a solution.
- **High-level languages** can be **used** on a *variety of different machines*; however, they are **not as efficient** as **low-level languages**.
- Here are a few features that high-level languages have, which are not available in low-level languages:
  - Selection and iteration constructs such as, IF…THEN…ELSE, FOR…ENDFOR, WHILE…ENDWHILE.
  - Boolean operators such as AND, OR and NOT which enable complex conditional statements to be constructed.
  - Identifiers using an unlimited number of alphabetic and numeric characters and some special characters to allow variable names to be made meaningful and sensible. For example, firstName, lastName etc.
  - Data structures such as arrays, lists, tuples and dictionaries (records).

Advantages of High-level Languages

- Most software is developed using high-level languages for the following reasons:
- High-level languages are relatively easy to learn and much faster to program in.
- Statements within these languages look like the natural English language, including mathematical operators making it easier to read, understand, debug and maintain.
- Complexed assignment statements such as:

X = (sqrt(b^2 – 4 * a * c))/(2 * a)

- Allowing the programmer to show how the algorithm will solve a problem in a clearer and more straightforward way.
- Specialised high-level languages have been developed to make the programming as quick and easy as possible for particular applications such as, SQL specially written to make it easy to search and maintain databases. HTML, CSS and JavaScript were also developed to help people create web pages and websites.

Low-level Languages

- **Low-level languages** have a **limited** number of **programming constructs**, with **selection** and **iteration** being **performed** using '**compare and branch**' instructions.
- These languages are **much harder to learn**, more **time-consuming** to code and **difficult to debug**.
- Once the program has been **translated** into **binary** it is referred to a **machine code**.
- **To write code** at the processor level we can use **assembly language** that is **specific** to a **processor architecture** or family of processor.
- The **assembly code** is written using **mnemonics**, abbreviated text commands such as *LDA (LOAD), STO (STORE) and ADD.*
- **Machine code** is *stored* in **binary** and looks a little like this:

  00100010 10110101 01001101 01011101 00100010 00010101 11010100 10111001
- **Assembly language** is **slightly** more **user friendly** but is still **too technical** for **most people** to **understand** and only specialist programmers can work using it.
- The **machine code** we **looked at previously** could be *converted* to **assembly language** as follows:

  LDA181 ADD93 LDA21 STO185
- It is still not very user friendly, but it is better than working directly with machine code.
- However, a **CPU** can only **understand machine code** so anything written in **assembly language** still **needs to be translated** into **machine code** for this to work. This is quite a fast process as there is a direct translation that can occur between the instruction and the address number.
- It is also **used by people** in cases of **computer forensics** or **brute-force cyber-attacks** and is sometimes used in extreme cases of debugging code to determine exactly what's going on but can only generally be done by people with very specialist knowledge.

Advantages of Low-level Languages

- Assembly language is often used in embedded systems such as systems that control a washing machine, traffic lights or a robot.
- It has features that make it suitable for the following types of applications:
- - It gives complete control to the programmer over the system components, so it can be used to control and manipulate specific hardware components.
- - It has very efficient code that can be written for a particular processor architecture, so will occupy less memory and execute (run) faster that a compiled/interpreted high-level language.

Differences between Assembly Code and Machine Code

- **Machine code** is the code **executed** by the **CPU** and **consists** only as **binary digits 0's** and **1's.**
- **Each type of processor** has its **own machine code instruction set**, and **all programs** no matter if they are written in **high-level languages** or in **assembly language** will **have to** be **translated** into **machine code** before **being executed**.

• Program Translators

- High-level languages and assembler languages need to be translated into machine code for a computer system to understand it.
- There are three types of translator programs that will do this:
  - Compiler
  - Interpreter
  - Assembler

Interpreters

- Once the program has been created, it needs to be saved before it can be run.
- At this point the programming language translates the source code into machine code one command/line at a time and immediately executes them.
- You must have the interpreter installed on your computer in order to run the software.
- Every time the program runs, it has to be translated again as there is no secondary file that is created to store the machine code and therefore must be translated each time the program is run.
- A long, complex program will take a considerably more time to execute if it is being interpreted.
- This makes interpreted code slower to run than compiled code, but it shows any errors as soon as it finds them, so it is easier to debug than compiled code. Python, Basic, JavaScript and Pascal are all interpreter-based programming languages.

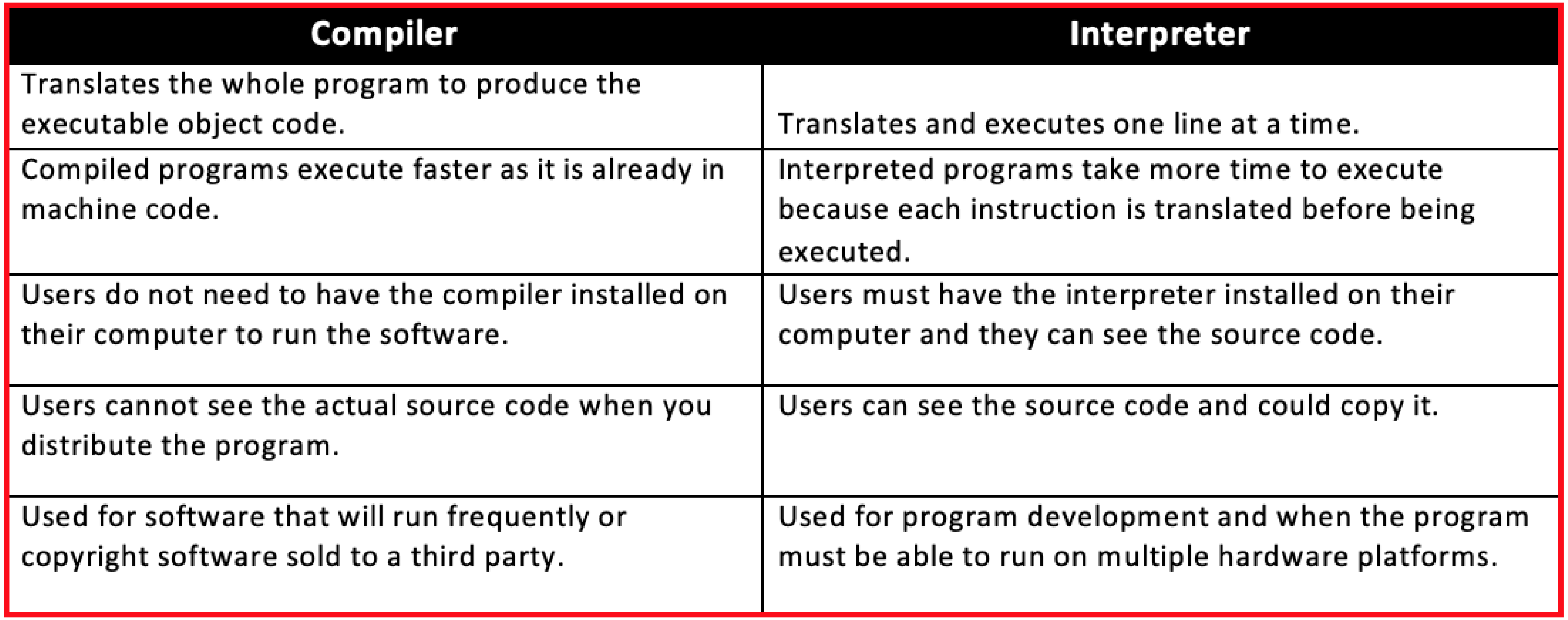
Compilers

- A compiler translates all the source code at the same time to create the compiled code, or machine code, also known as the object code.
- The machine code is saved and stored in a separate file to the high-level programming language.
- Once the programmer has created the program, they need to request it to be compiled before they can run and test the file which can take a while.
- This can make testing small sections of the program slower as the whole program needs to be compiled before any of it can be run.
- Compiling can take a long time, but once complete the compiled code runs quickly and reports a list of errors, if any have occurred.
- Overall, once the program has finished compiling, compiled programs are faster to run. Java and C++ are compiler programming languages.
- For the developer, compiled code has the advantage that the user of the software cannot see the source code or copy it.

Assembler

- An assembler translates assembly language into machine code and is effectively a compiler for the assembly language, but can also be used interactively like an interpreter.
- Assembly language uses words called ‘mnemonics’, such as LOAD, STORE and ADD. The instructions are specific to the hardware being programmed because different CPUs use different programming languages.
- Finally, every assembly language instruction is translated into a single machine code instruction.

Compiler Vs Interpreter



Compiler	Interpreter
Translates the whole program to produce the executable object code.	Translates and executes one line at a time.
Compiled programs execute faster as it is already in machine code.	Interpreted programs take more time to execute because each instruction is translated before being executed.
Users do not need to have the compiler installed on their computer to run the software.	Users must have the interpreter installed on their computer and they can see the source code.
Users cannot see the actual source code when you distribute the program.	Users can see the source code and could copy it.
Used for software that will run frequently or copyright software sold to a third party.	Used for program development and when the program must be able to run on multiple hardware platforms.

C Programming Language

C Language

- *C is a structured programming language. It is considered a high-level language because it allows the programmer to concentrate on the problem at hand and not worry about the machine that the program will be using. That is another reason why it is used by software developers whose applications have to run on many different hardware platforms.*

History

- Created in 1972 to write operating systems (Unix in particular)
- By Dennis Ritchie
- Bell Labs
- Evolved from B
- Can be portable to other hardware (with careful design – use Plauger’s The Standard C Library book)
- Built for performance and memory management – operating systems, embedded systems, real-time systems, communication systems

Standardization

- 1989 ANSI and ISO -> Standard C
- 1999 C99
- 2011 C11

- Don’t get thrown when you lookup information on websites and find conflicts based upon standards

- 1979 C++ by Bjarn Stroustrup also at Bell
  - Object orientation
- 1991 Java by Sun
  - Partial compile to java bytecode: virtual machine code
  - Write once, run anywhere
  - Memory manager – garbage collection
  - Many JVMs written in C / C++

C Program Phases

- Editor - code by programmer
- Compiling using gcc:
  - Preprocess – expand the programmer’s code
  - Compiler – create machine code for each file
  - Linker – links with libraries and all compiled objects to make executable
- Running the executable:
  - Loader – puts the program in memory to run it
  - CPU – runs the program instructions

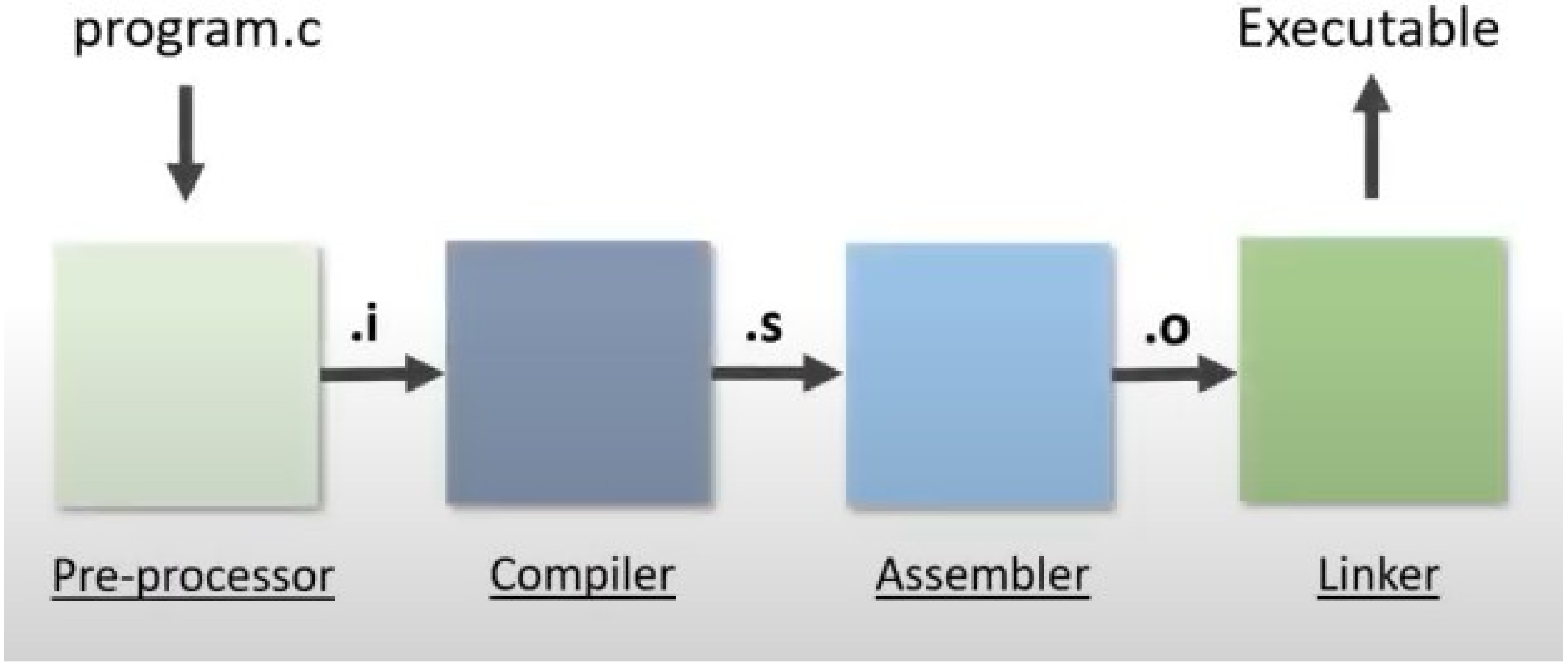
Install C Compiler

- For Linux (debian based):
  - apt install build-essential
- For Windows:
  - mingw
  - ...

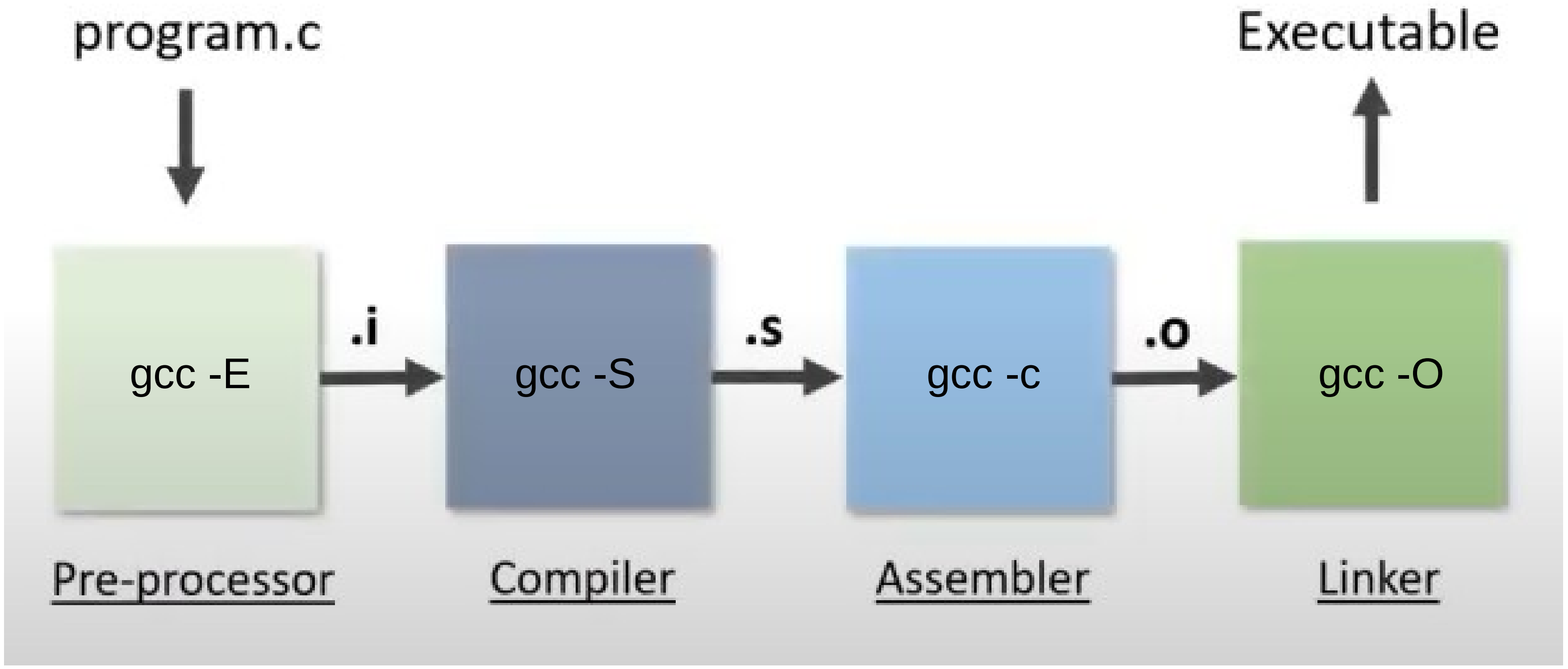
First Code

```
#include <stdio.h>
int main () {
    printf(“\nHello World!!!\nThis is my first C code\n\n”);
    return 0;
}
```

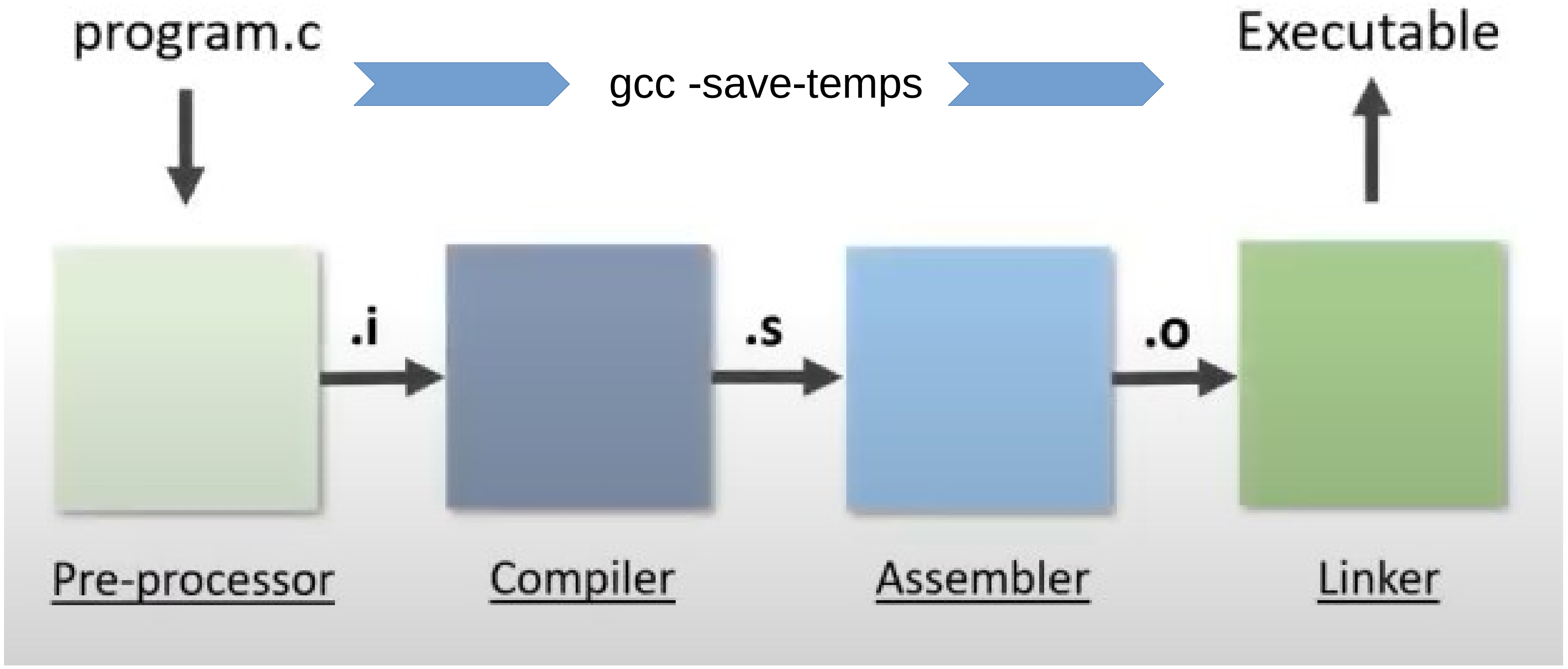
C Compilation Process



C Compilation Process



C Compilation Process



Coding Challenge Websites

- TopCoder (https://www.topcoder.com/challenges/)
- HackerRank (https://www.hackerrank.com/domains)
- Codewars(https://www.codewars.com/)
- And lots more

END

Questions?