

**Big Question:** What are compilers, interpreters and assemblers and how are they different?

Monday, 04 April 2022

## Learning Intention

**To develop knowledge** by Describing compilers, interpreters and assemblers.

**To secure understanding** by Describe the purpose and give examples of the use of compilers, interpreters and assemblers.

**To achieve excellence** by Explain the principal stages involved in the compilation process



### **Executable**

*Be able to be run by a computer.*

### **Tier 2**

#### **Convert**

*To change something into a different form*

# Converting programming languages

Programs written in high-level and low level languages need converting into machine code (binary) so a computer can understand it. This is done by using one of the following:

Assembler

Interpreter

Compiler

**To develop knowledge** by

Describing compilers, interpreters and assemblers.

**To secure understanding**

by Describe the purpose and give examples of the use of compilers, interpreters and assemblers.

**To achieve excellence** by

Explain the principal stages involved in the compilation process

<https://www.bbc.co.uk/bitesize/guides/zmthsrd/revision/1>

# Interpreters

**To develop knowledge** by Describing compilers, interpreters and assemblers.

**To secure understanding by** Describe the purpose and give examples of the use of compilers, interpreters and assemblers.

- Converts High level languages into machine code
- Goes through the code line by line translating one **instruction** at a time.

## Advantages:

- Instructions are executed as soon as they are translated.
- Require less available memory since instructions are executed once translated, they are not stored for later use.
- Errors can be spotted quickly. Once an error is found, the program stops running and the user is notified at which part of the program the interpretation has failed.

## Disadvantages:

- Interpreted programs **run more slowly** as the processor has to **wait for each instruction** to be translated before it can be executed.
- The program has to **be translated every time it is run.**
- Interpreters do not produce an **executable** file that can be distributed. The source code **program has to be supplied** and this could be **modified without permission.**
- Interpreters do not optimise code (make it more efficient) - the translated code is executed as it is.

## **Executable**

*Be able to be run by a computer.*

# Compilers

**To develop knowledge** by Describing compilers, interpreters and assemblers.

**To secure understanding** by Describe the purpose and give examples of the use of compilers, interpreters and assemblers.



- Converts High level languages
- Translates the whole program in one go. This process is called **compilation**.
- It creates a separate **executable** file.

## Advantages:

- Compiled programs run quickly since they have already been translated.
- A compiled program can be supplied as an executable file (a file is a file that is ready to run). These are difficult for other people to modify without access to the source code.
- Compilers **optimise** code (make it more efficient). Optimised code can run quicker and take up less **memory** space.

## Disadvantages:

- Because the source code is translated as a whole, there must be **enough memory space to hold the source code, the compiler and the generated object code**.
- The program **has to be compiled and run before errors are encountered**. This makes it harder to see where the errors lie.
- The source code must be **recompiled every time the programmer changes the program**.
- Source code compiled on one platform will not run on another - the object code is specific to the **processor's** architecture.

# Assemblers

**To develop knowledge** by Describing compilers, interpreters and assemblers.

**To secure understanding by** Describe the purpose and give examples of the use of compilers, interpreters and assemblers.

The purpose of an assembler is to translate **assembly language** into machine code

Assemblers create one machine code instruction for each assembly instruction.

Translator	Plus points	Minus points
Assembler	<ul style="list-style-type: none"><li>• Gives precise and direct access to the computer hardware.</li></ul>	<ul style="list-style-type: none"><li>• Difficult to code.</li><li>• Few commands are available.</li></ul>

<https://www.bbc.co.uk/bitesize/guides/zmthsrd/revision/1>



# Task

S.P.I.R.I.T

- ✓ Self management
- ✓ Innovation

For each of the following :

- Interpreter
- Compiler
- Assembler

1. Write a definition
2. Explain what type of program uses it (e.g high level or assembly)
3. Advantages and disadvantages

## Learning Intention

**To develop knowledge** by  
Describing compilers, interpreters and assemblers.

**To secure understanding** by  
Describe the purpose and give examples of the use of compilers, interpreters and assemblers.



# Stages of compilation

S.P.I.R.I.T

- ✓ Self management
- ✓ Innovation

## **KNOW IT:**

List the stages of compilation in the correct order.

- Try and memorise the list.
- Cover and test yourself.
- Repeat and get a partner to **test you**

## **GRASP IT:**

Use the document **Program compiling simplified**

Write the stages in the correct order with a short description

## **THINK IT:**

Use the document **Program compiling**

Write the stages in the correct order.

**Write a description in your own words to explain each stage**

**Excellence:** Explain the principal stages involved in the compilation process:

- lexical analysis, **symbol table construction,**
- syntax analysis,
- semantic analysis,
- code generation,
- Code optimisation.

***To achieve excellence*** by

Explain the principal stages involved in the compilation process