Translators

Lesson objectives

- Advantages and disadvantages of compilers
- Advantages and disadvantages of interpreters

Assessment criteria

- Describes differences between compilers and interpreters
- Lists the advantages and disadvantages of each translator

Vocabullary

Assembler
Compiler
Interpreter
Object code
Source code
Intermediate code

https://www.youtube.com/watch?v=_C5AHaS1mOA

1) Watch the video
2) Make questions for each other

Translators

- Assembler
- Compiler
- Interpreter

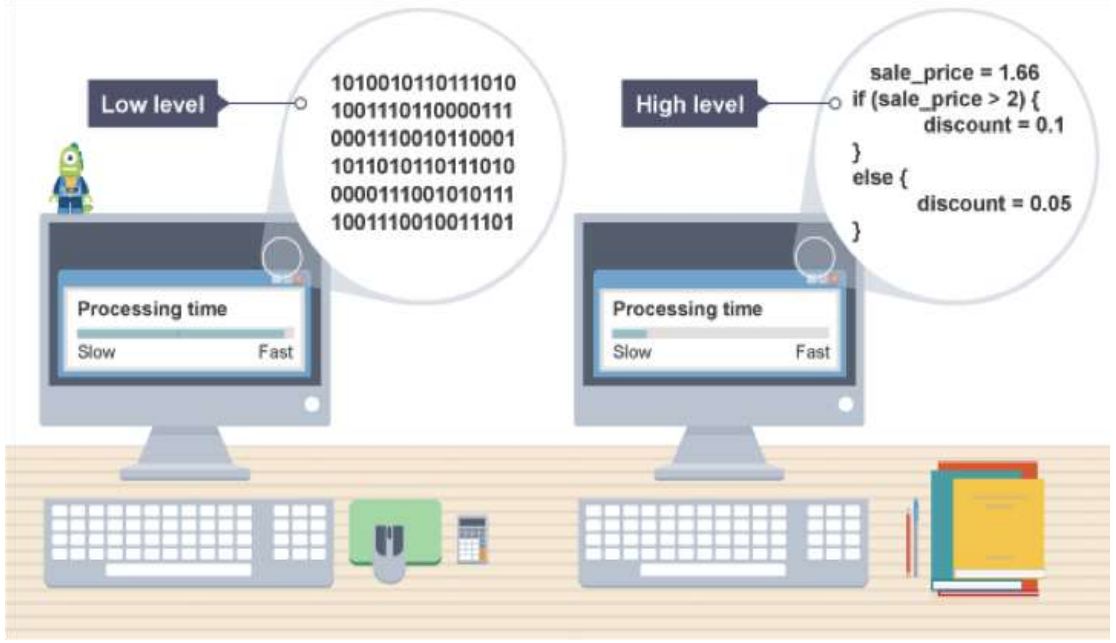
Concise definitions!

Give a definition for the following three terms.

- Each definition must be 15 words or less.
- Each definition must make at least 3 valid points.

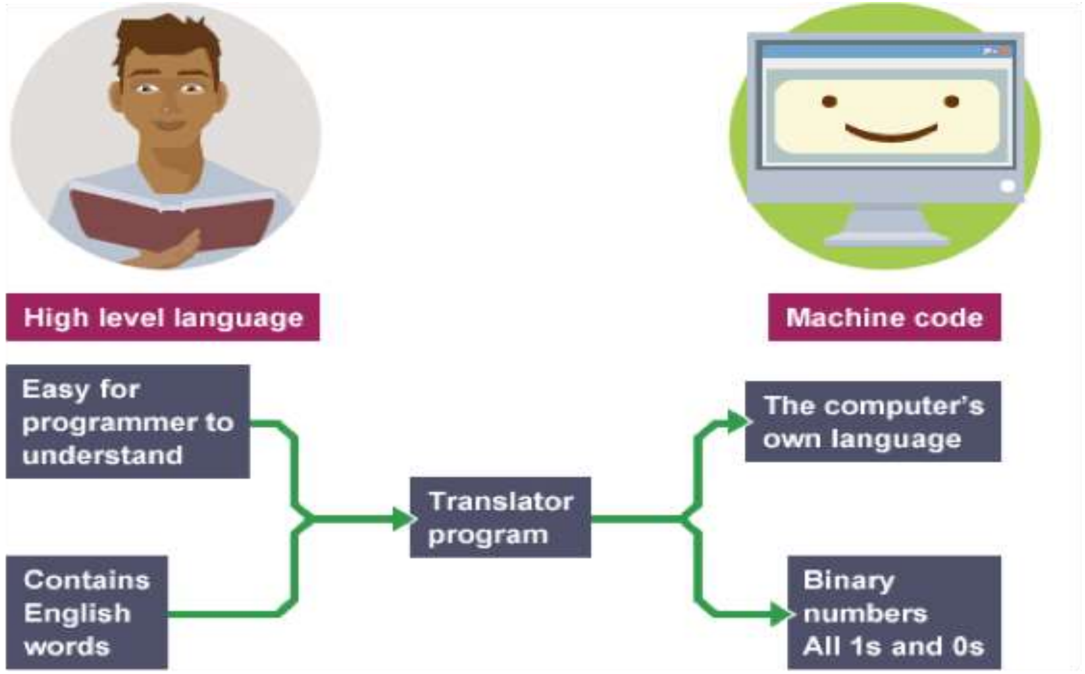
Interpreter	Definition here…
Compiler	Definition here…
Assembler	Definition here…

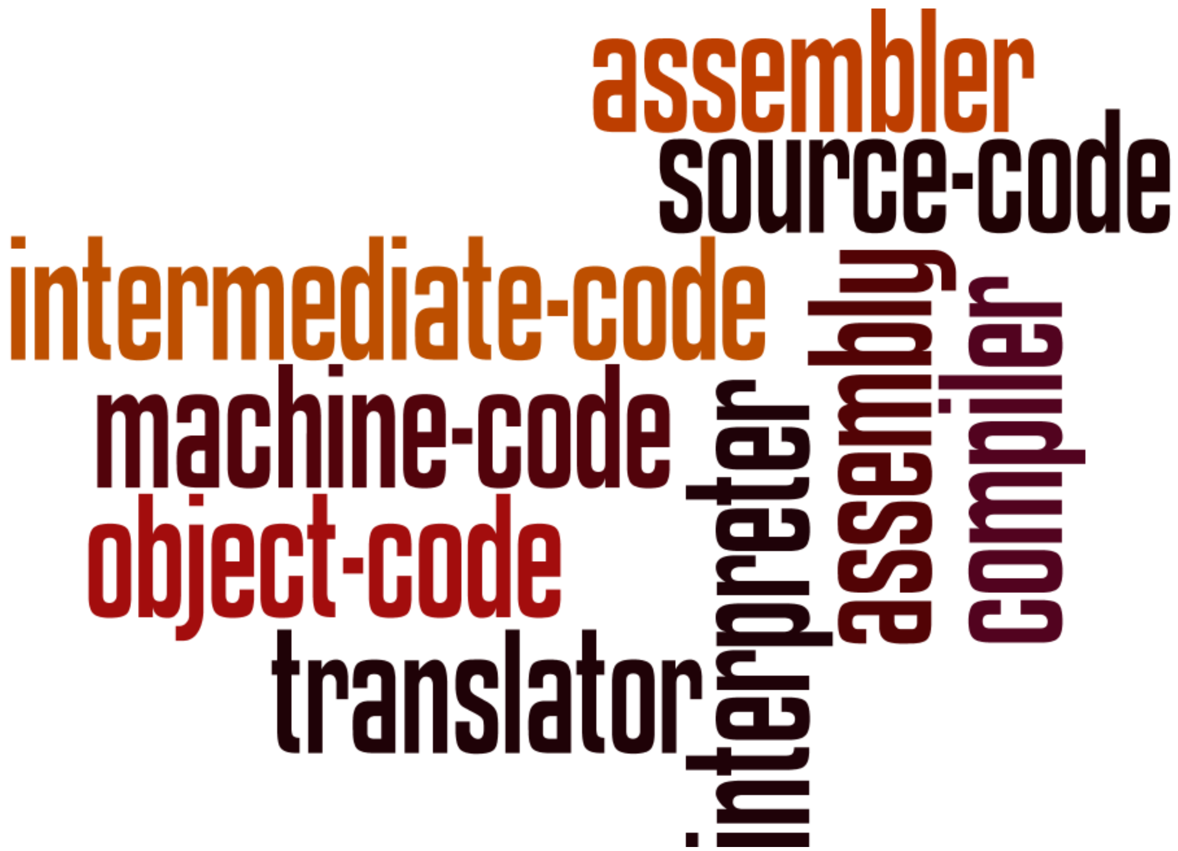
Interpreter	Takes one line of code, translates it, then runs it right away.
Compiler	Takes source code, translates it all into object code before allowing it to run.
Assembler	Translates a program written in assembly language into machine code.



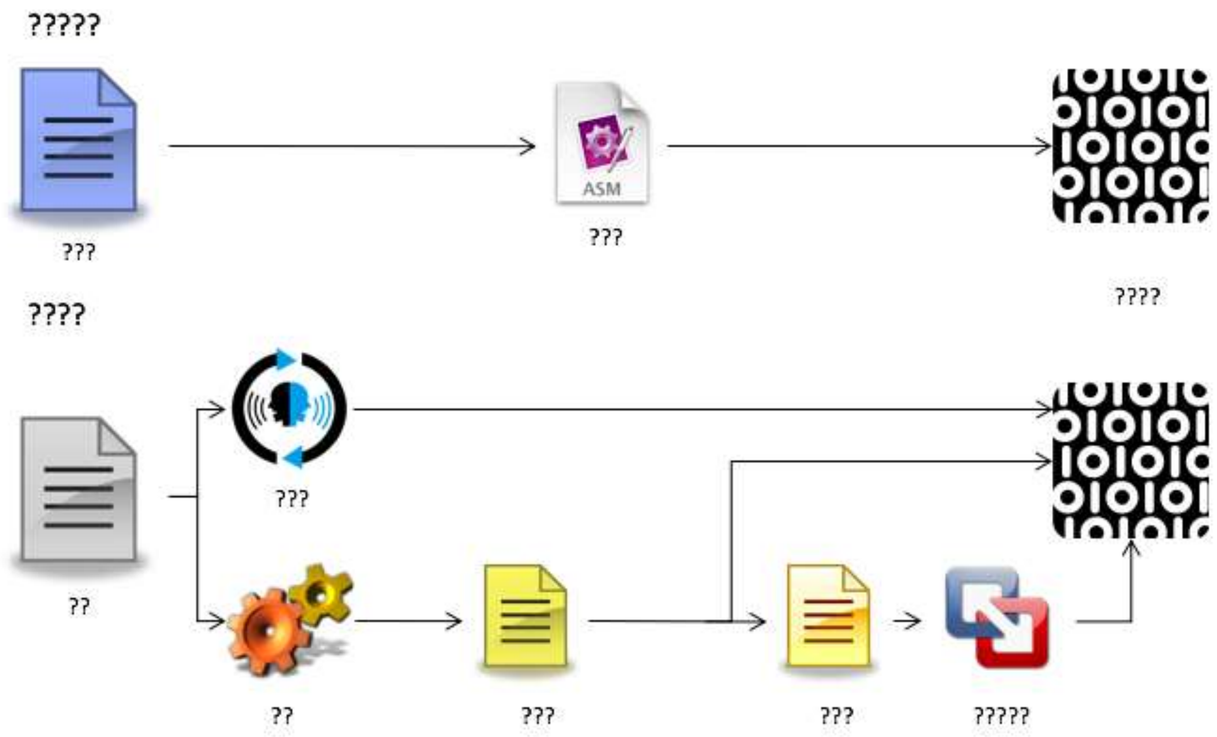
```
1010010110111010
1001110110000111
0001110010110001
1011010110111010
0000111001010111
1001110010011101
```

```
sale_price = 1.66
if (sale_price > 2) {
        discount = 0.1
}
else {
        discount = 0.05
}
```

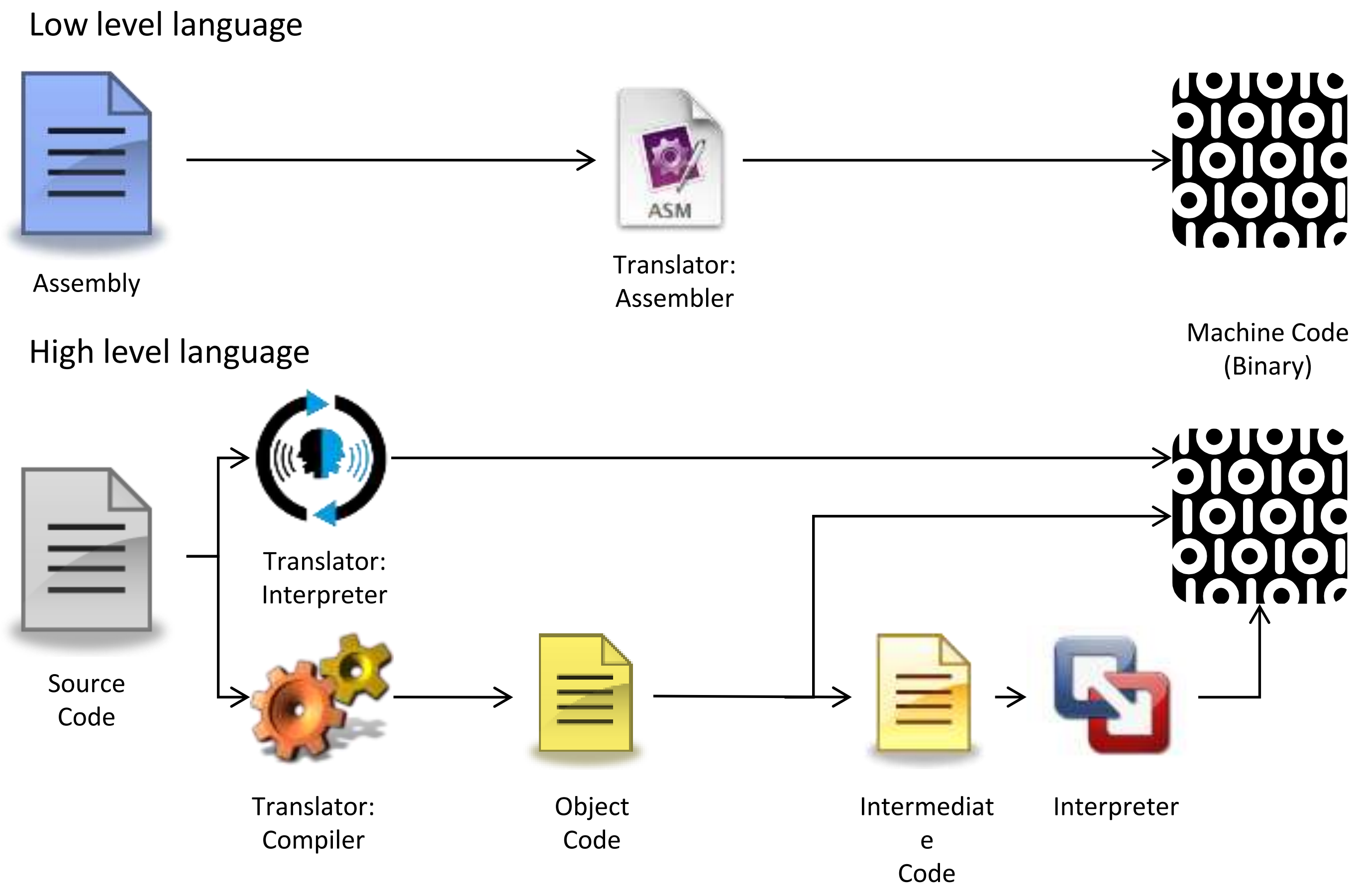

High level language
Machine code
Easy for programmer to understand
Contains English words
Translator program
The computer's own language
Binary numbers All 1s and 0s

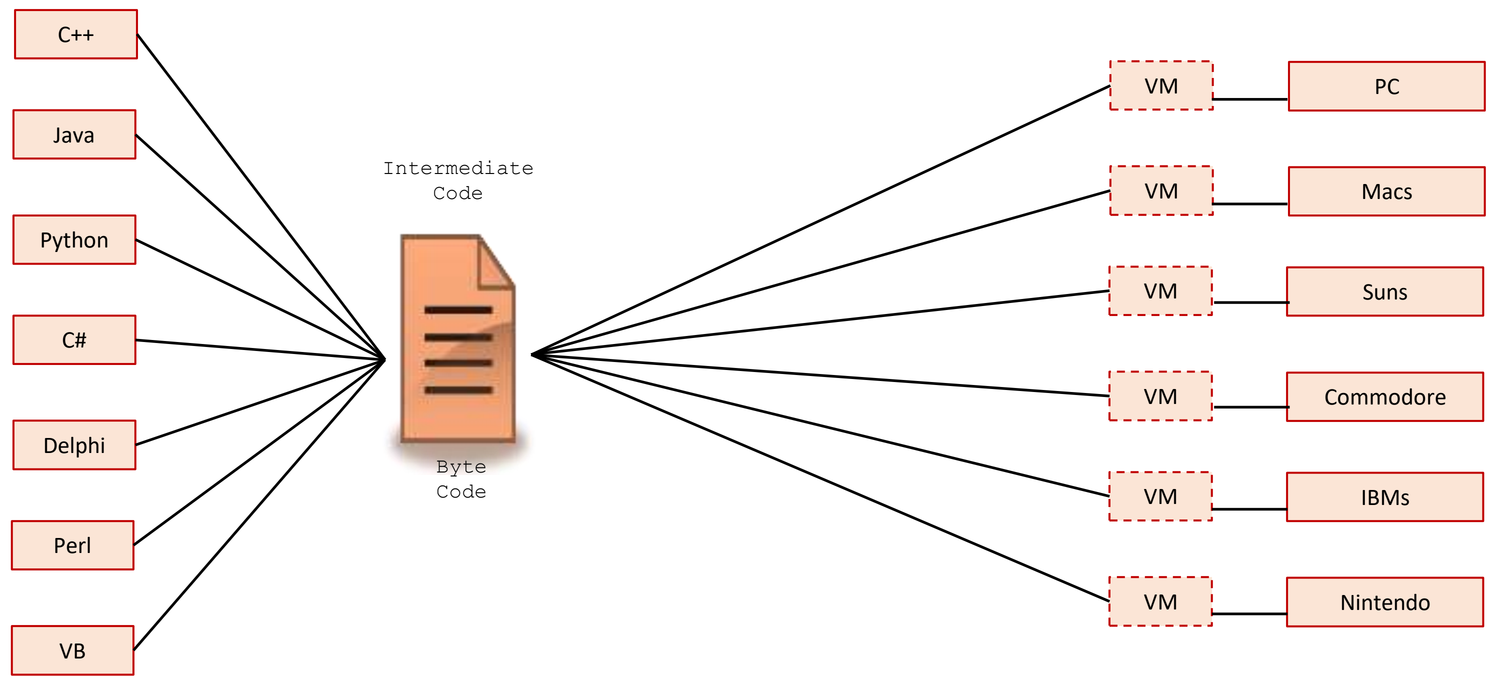


Using all the words in this word-wall, create a diagram to show how all the concepts are linked together.



Answer


Low level language
Assembly
Translator: Assembler
Machine Code (Binary)
High level language
Source Code
Translator: Interpreter
Translator: Compiler
Object Code
Intermediat e Code
Interpreter



A solution was developed to have the translators generate to a kind of "half-way" standard intermediary code which could then be translated to each computers own specific machine code.

This half way language is called "**intermediate code**", often known as "**bytecode"**. It is kind of useless on its own as it won't run without any further translation to turn it into machine code.

It does however run on a sort of 'pretend' machine that it was designed for, although this machine does not physically exist, it is installed on each make of computer, and it performs the job of taking the "generic" intermediate code and translating it into machine code specific for that machine.

This pretend machine is known as a "virtual machine".
Writing an interpreter to translate bytecode is a much easier task than writing an interpreter to translate high-level source code.
Bytecode is very portable and very compact.
Interpreting bytecode programs are faster than high-level source code programs.

machine codes

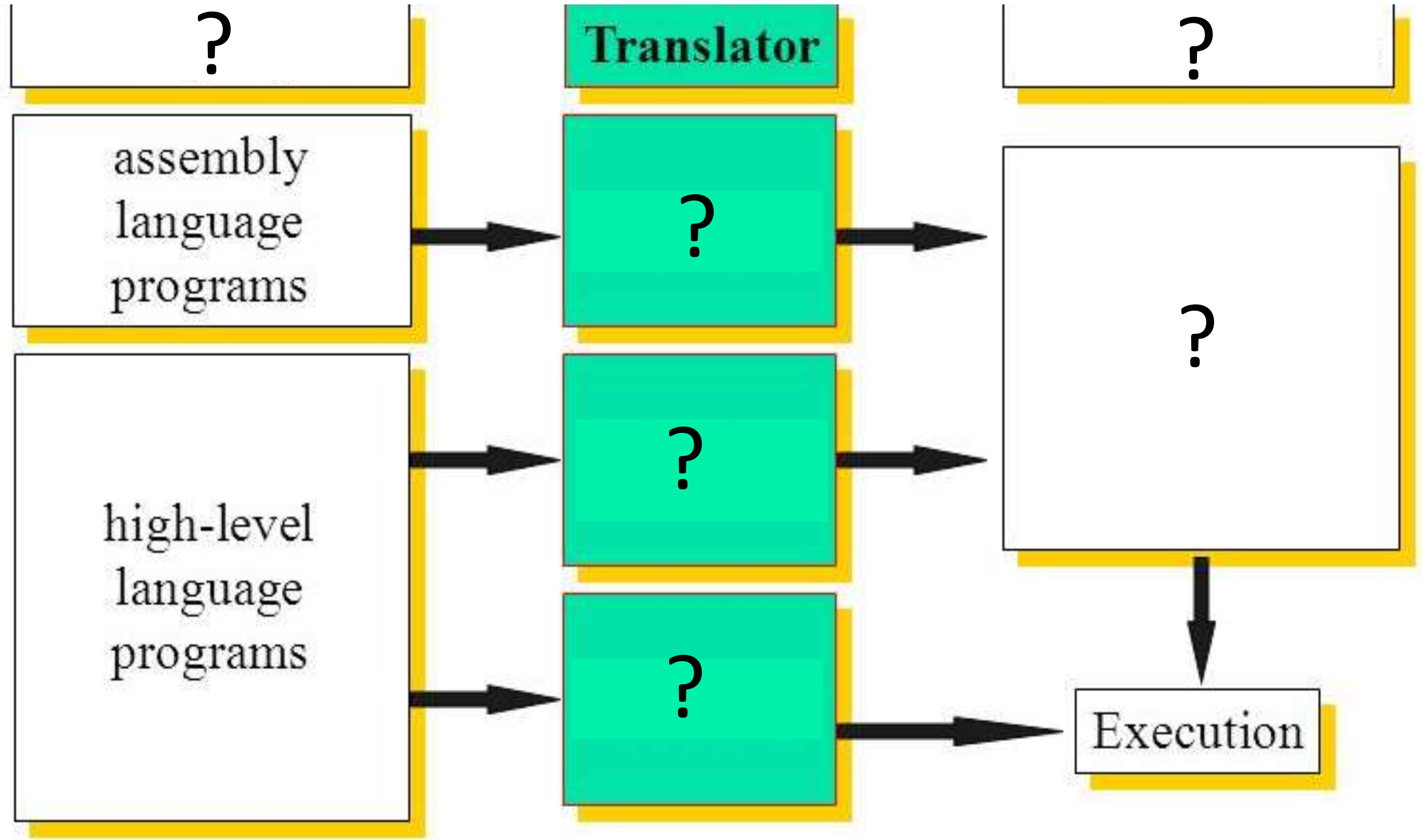
compiler

interpreter

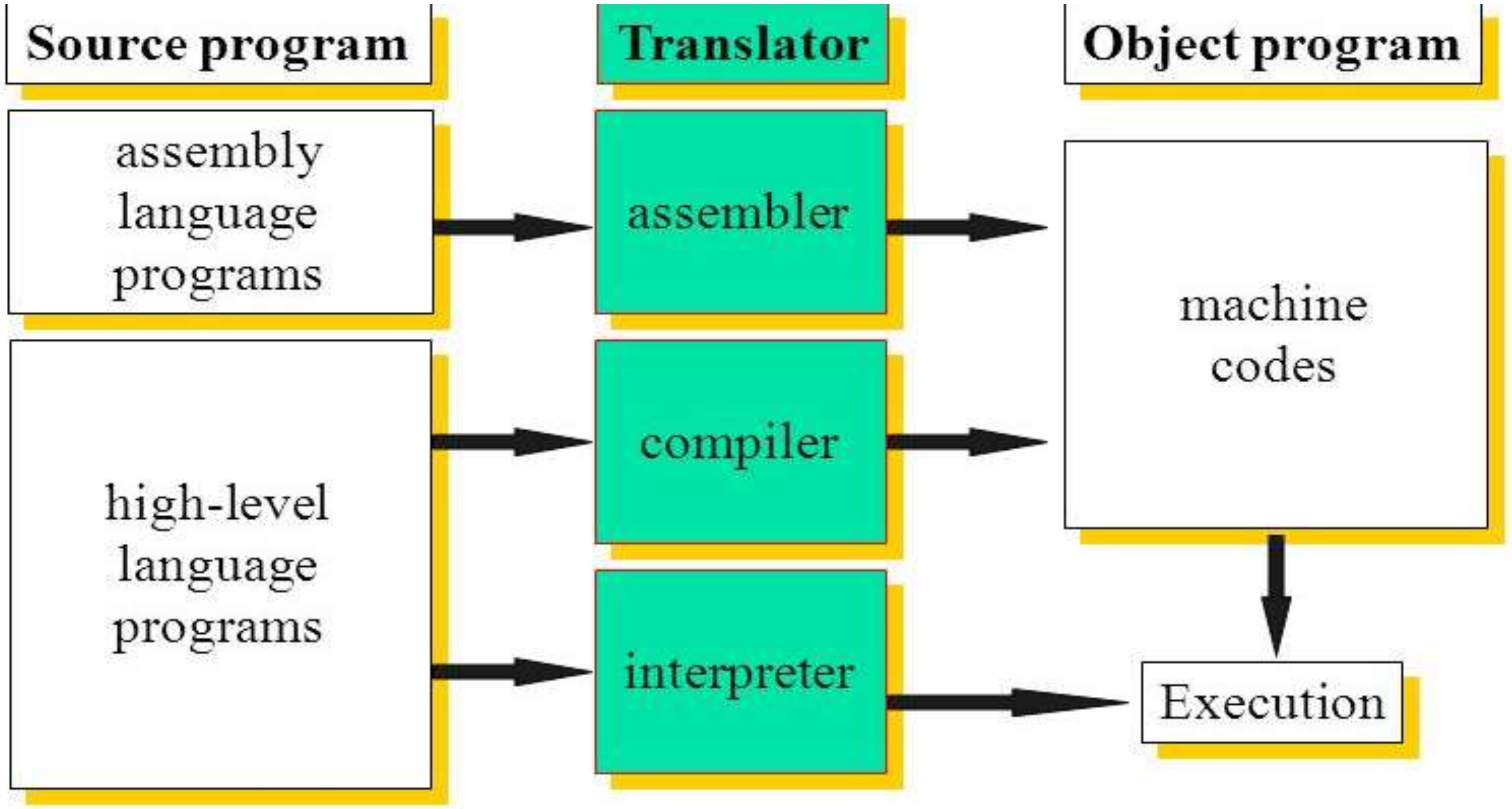
assembler

Object program

Source program



The functions of the three types of translators

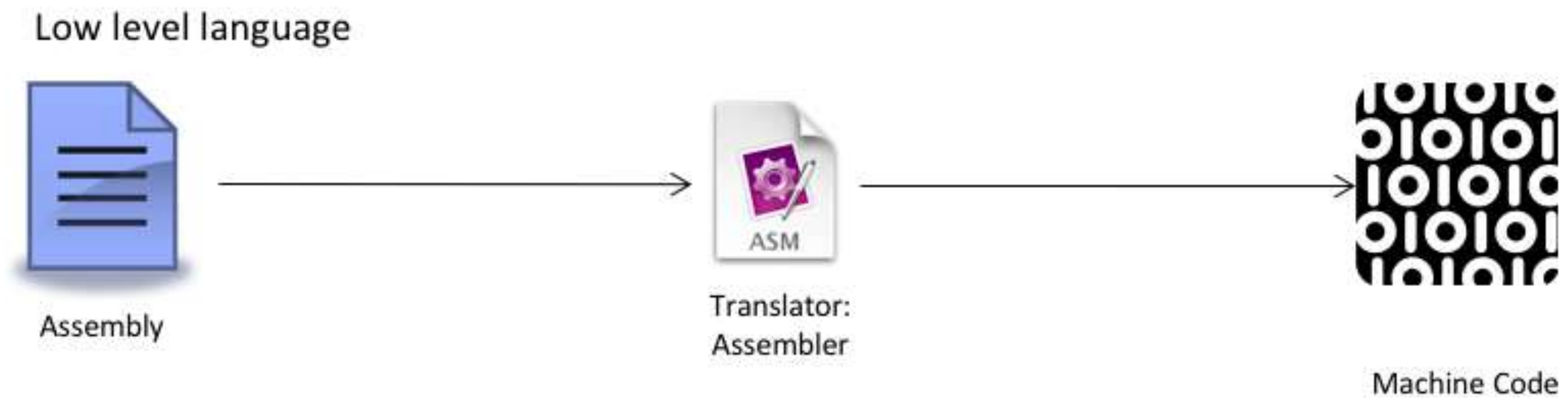


The functions of the three types of translators

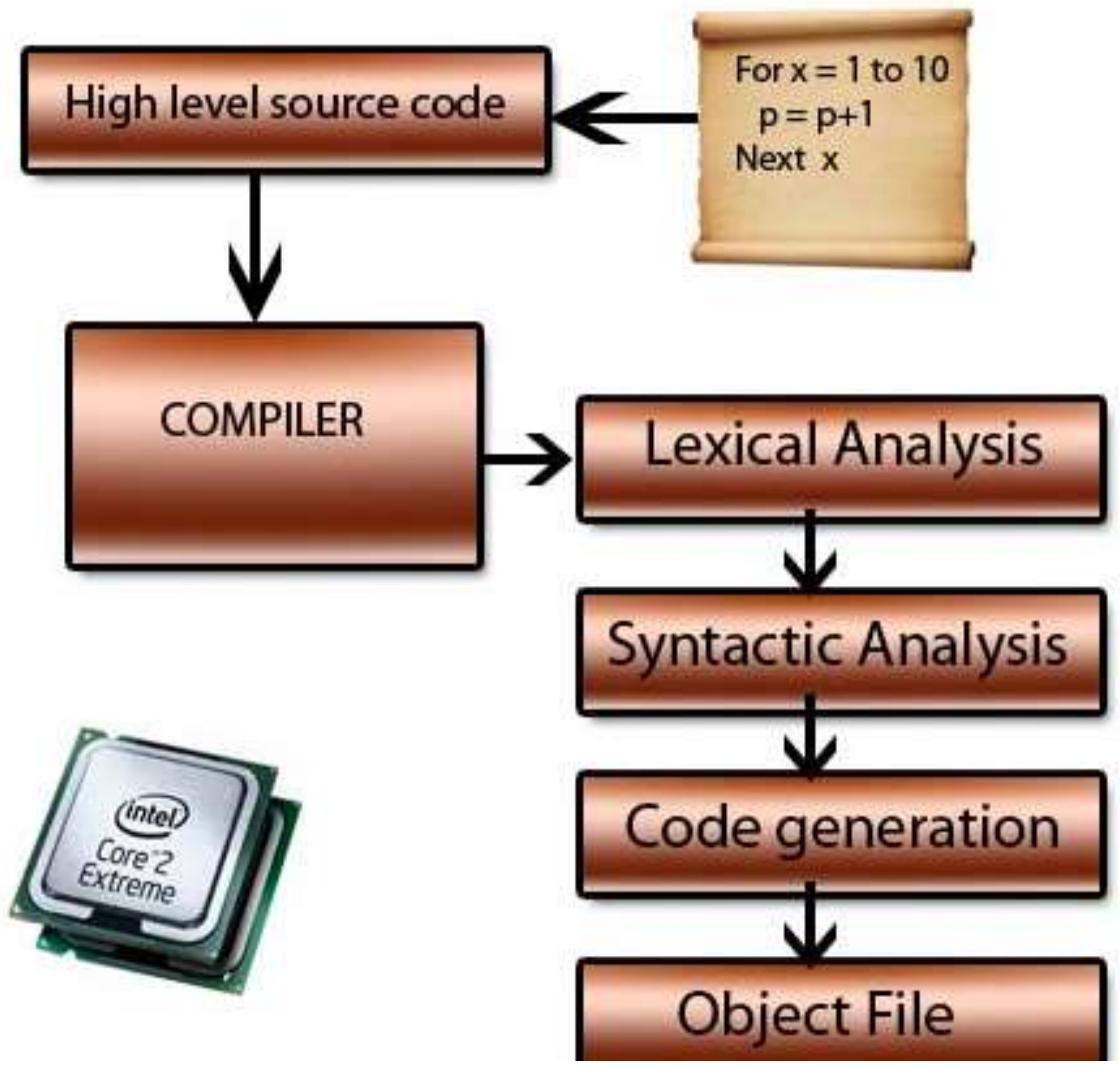
assembler

Assembler

- To convert the assembly language into machine code.
- Translate mnemonic operation codes to their machine language equivalents.



Compiler



```
For x = 1 to 10
  p = p+1
Next x
```

Compiler

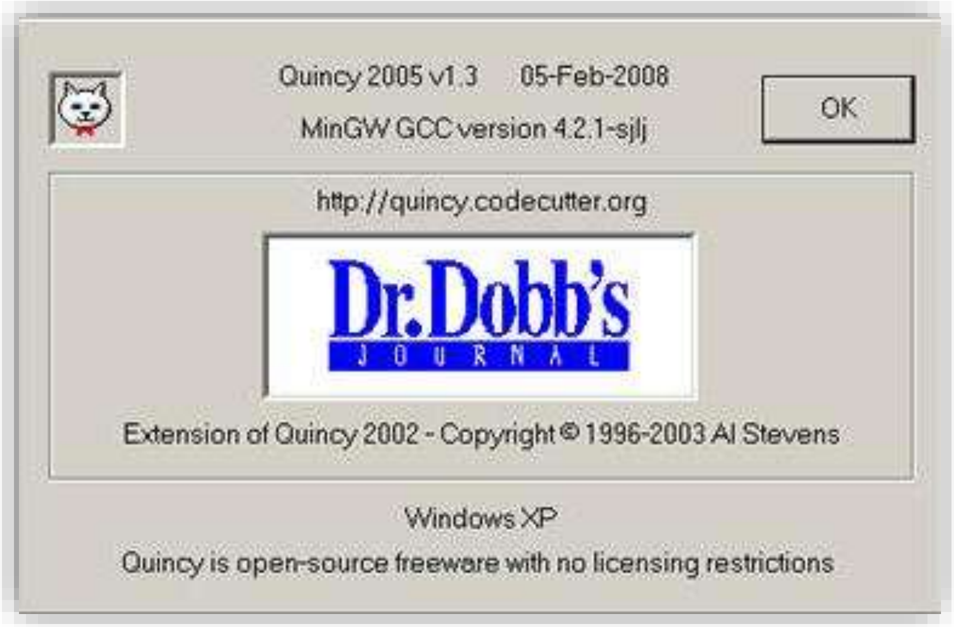
- Compiler:
  - Checks syntax of program
  - Checks at a time all the program
- Primary reason for compiling source code is to create an executable program
- Examples of compiler based language:
  - C, C++, JAVA

Executables

- Executables: Files that actually do something by carrying out a set of instructions.
- E.g., *.exe* files in Windows
- Once the executable is there, it can be called by the user to process the given inputs to a program and produce the desired outputs.

Example of Compiler

- Some of examples of Compiler:
  - Microsoft Visual Studio
  - BlueJ
  - Quincy 2005


Quincy 2005 v1.3 05-Feb-2008

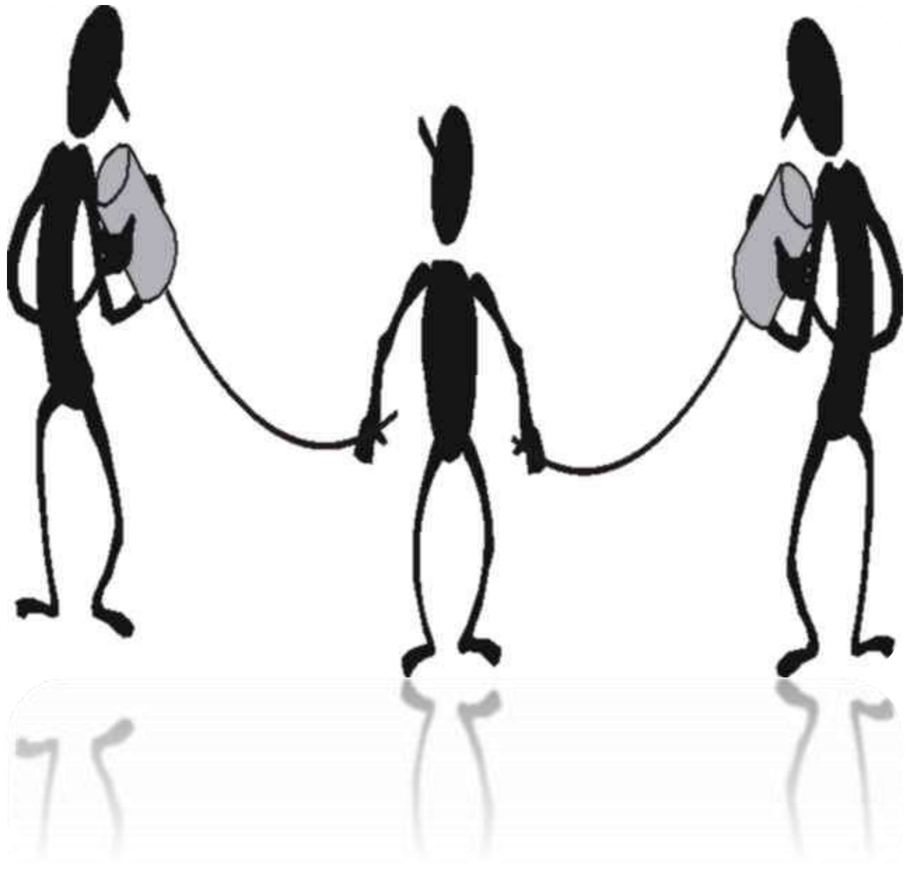
MinGW GCC version 4.2.1-sjlj

http://quincy.codecutter.org

Extension of Quincy 2002 - Copyright © 1996-2003 Al Stevens

Windows XP

Quincy is open-source freeware with no licensing restrictions



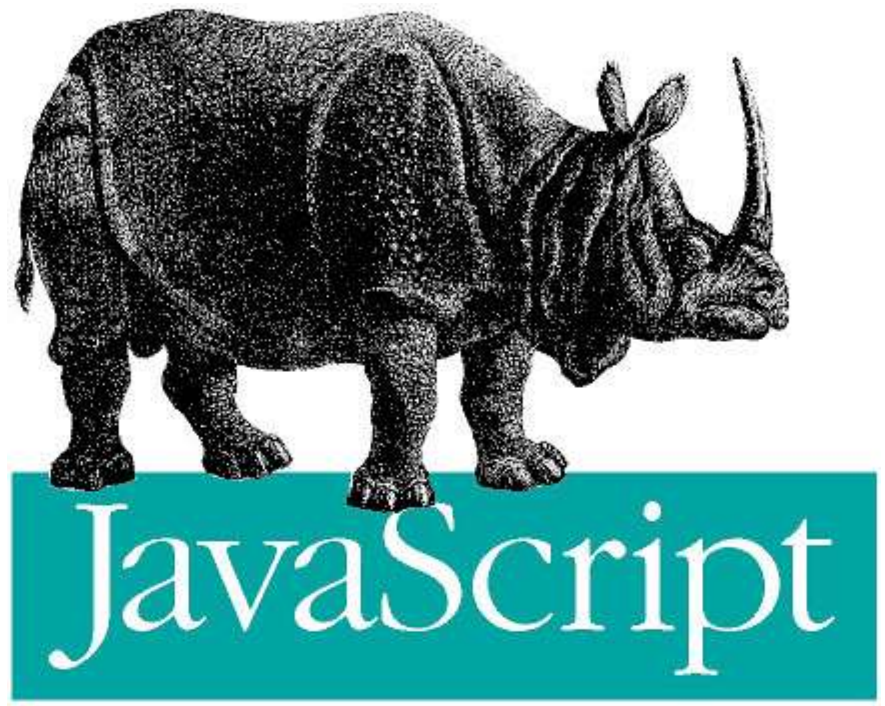
Interpreter

Interpreter

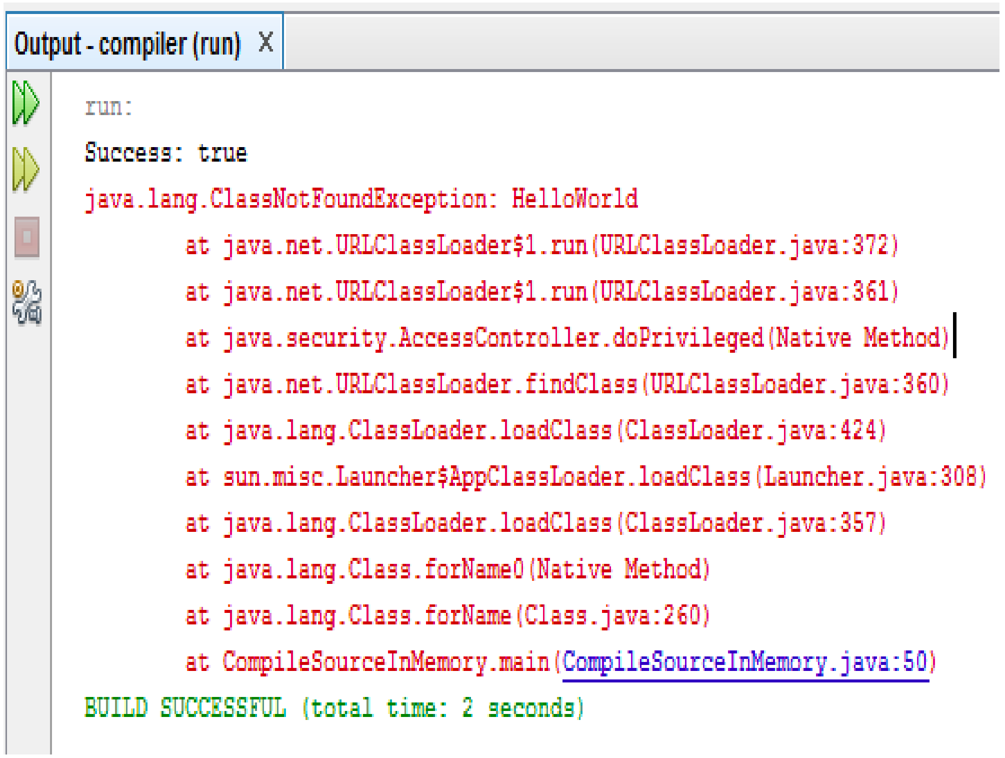
- A computer program that executes instructions written in a programming language and do not produces the executable file.
- Interpreter:
  - Checks the keywords of a program
  - Taking one instruction at a time and convert it into machine language before taking upon the next instruction.
- Examples of interpreter based language:
  - PHP, JavaScript, BASIC

Example of Interpreter

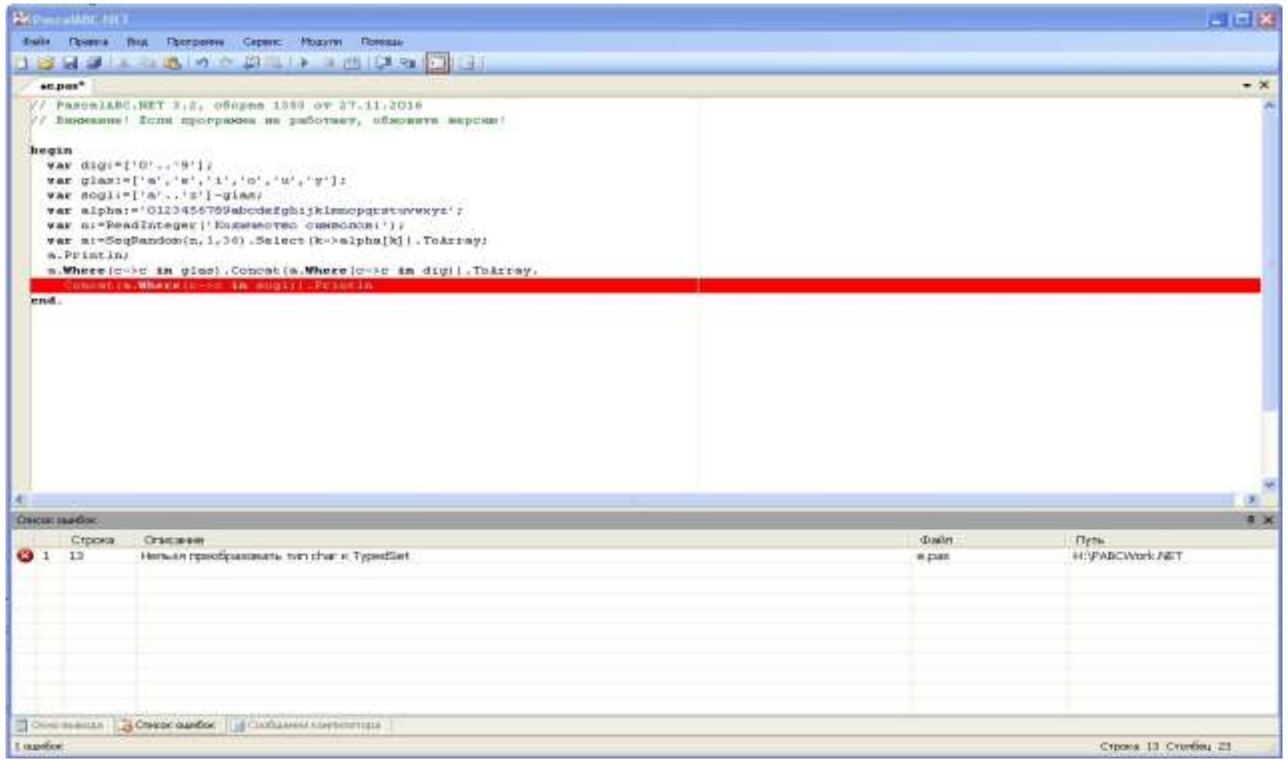
- We use JavaScript language.
- JavaScript engine is specialized computer software which interprets and executes JavaScript. Mostly used in web browsers.



Compiler VS Interpreter



```
run:
Success: true
java.lang.ClassNotFoundException: HelloWorld
        at java.net.URLClassLoader$1.run(URLClassLoader.java:372)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:360)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Class.java:260)
        at CompileSourceInMemory.main(CompileSourceInMemory.java:50)
BUILD SUCCESSFUL (total time: 2 seconds)
```



```
// PascalABC.NET 3.2, сборка 1359 от 27.11.2016
// Внимание! Если программа не работает, обновите версию!

begin
  var dig:=['0'..'9'];
  var glas:=['a','e','i','o','u','y'];
  var sogl:=['a'..'z']-glas;
  var alpha:='0123456789abcdefghijklmnopqrstuvwxyz';
  var n:=ReadInteger('Количество символов:');
  var a:=SeqRandom(n,1,36).Select(k->alpha[k]).ToArray;
  a.Println;
  a.Where(c->c in glas).Concat(a.Where(c->c in dig)).ToArray.
    Concat(a.Where(c->c in sogl)).Println
end.
```

		Строка	Описание	Файл	Путь
❌	1	13	Нельзя преобразовать тип char к TypedSet	e.pas	H:\PABCWork.NET

1 ошибок Строка 13 Столбец 23

Summary

- Assembler = To convert the assembly language into machine code.
- Compiler = A program that changes source code (*high-level language)* to object code which that can be executed by a machine.
- A computer program that executes instructions written in a programming language and do not produces the executable file.

Formative assessment 1

Place the each item under the respective title

An executable file of machine code is produced (object code)

Compiled programs no longer need the compiler

Error report produced once entire program is compiled. These errors may cause your program to crash

Interpreted code is run through the interpreter (IDE), so it may be slow, e.g. to execute program loops

An executable file of machine code is produced (object code)

One high-level language statement may be several lines of machine code when compiled

One low-level language statement is usually translated into one machine code instruction

Error message produced immediately (and program stops at that point)

Temporarily executes high-level languages, one statement at a time

Interpreted programs cannot be used without the interpreter

Translates high-level languages into machine code

Compiling may be slow, but the resulting program code will run quick (directly on the processor)

No executable file of machine code is produced (no object code)

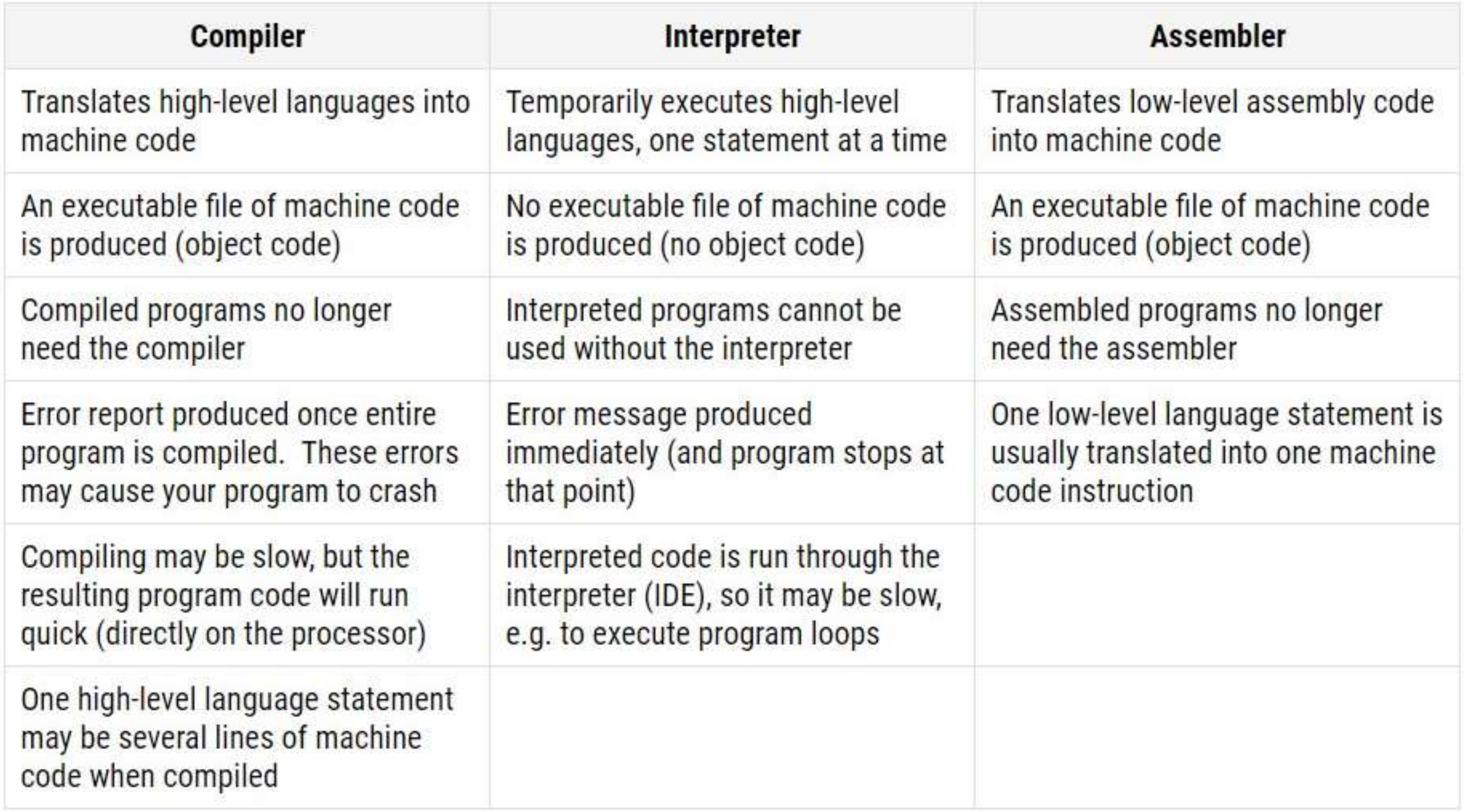
Translates low-level assembly code into machine code

Assembled programs no longer need the assembler

Place the each item under the respective title

Compiler	**Interpreter**	**Assembler**

Answers



Compiler	Interpreter	Assembler
Translates high-level languages into machine code	Temporarily executes high-level languages, one statement at a time	Translates low-level assembly code into machine code
An executable file of machine code is produced (object code)	No executable file of machine code is produced (no object code)	An executable file of machine code is produced (object code)
Compiled programs no longer need the compiler	Interpreted programs cannot be used without the interpreter	Assembled programs no longer need the assembler
Error report produced once entire program is compiled. These errors may cause your program to crash	Error message produced immediately (and program stops at that point)	One low-level language statement is usually translated into one machine code instruction
Compiling may be slow, but the resulting program code will run quick (directly on the processor)	Interpreted code is run through the interpreter (IDE), so it may be slow, e.g. to execute program loops	
One high-level language statement may be several lines of machine code when compiled		

Fill in the table with the advantages and disadvantages of the two types of translators (compiler and interpreter)

	Assembler	Compiler	Interpreter
Advantages			
Disadvantages			

2. Place the following statements into the correct locations to show your understanding of the different between an assembler, interpreter and a compiler. Some statements can straddle more than one category.

Assembler	Interpreter	Compiler
Translates low-level code into machine code	Translates high-level code into machine code	Translates high-level code into machine code
Processor architecture specific	Translates high-level code directly into machine code	Often translates high-level code into intermediate / byte code
Translates source code on a one-to-one basis	Translates source code on a one-to-many basis	Translates source code on a one-to-many basis
	Processor architecture independent	Processor architecture independent
	Translates one line of code at a time and then executes it	Translates entire source code and produces object code

Formative assessment 2

Complete a dry run test using a trace table.

Pair work

Analyze programming languages and separate them into two: compiler and interpreter-type languages. Discuss with other pairs and compare your results.

	Compiler-type	Interpreter-type
Programming Languages		