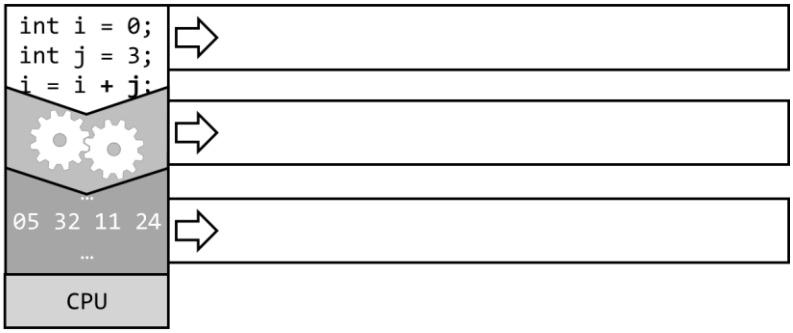
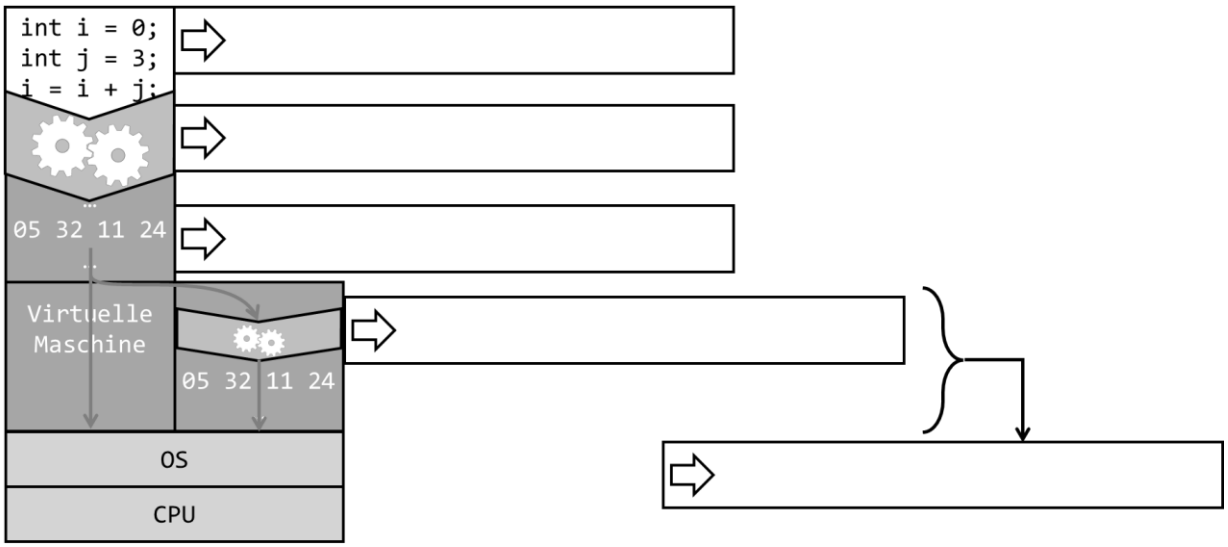
1 Computerseitige Verarbeitung des Codes

Der von uns geschriebene Code muss bei kompilierten Programmiersprachen mit Hilfe eines Übersetzers auf das Zielsystem vorbereitet werden.

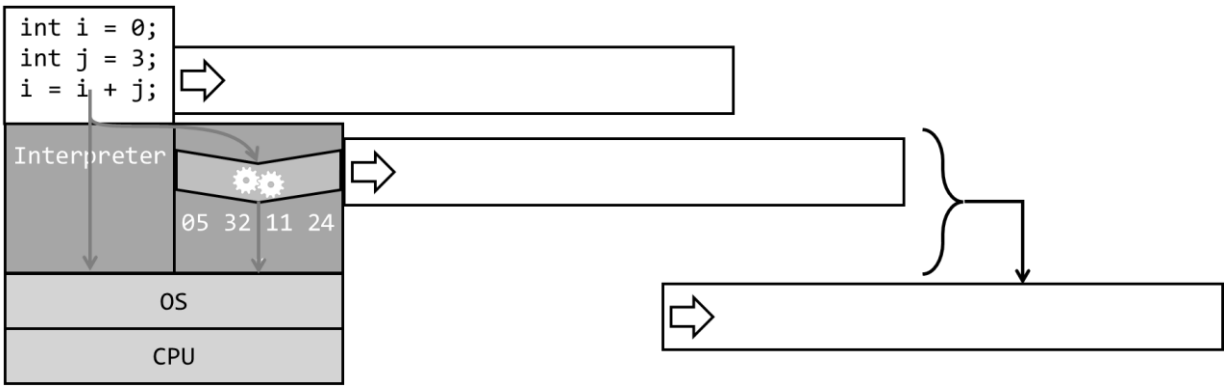


Diese Architektur finden wir im Regelfall bei Mikroprozessoren, welche oftmals mit C oder C++ programmiert werden. Bei PC- oder Serversystemen sitzt zwischen dem ausgeführten Maschinencode und dem Prozessor noch das **Betriebssystem** (OS). Dieses ist für das Ressourcenmanagement zuständig. Programme müssen somit vom Compiler auf das Betriebssystem des Zielsystems vorbereitet werden, da sie die Ressourcen über das OS beziehen. Ein für Windows kompiliertes Programm kann bspw. nicht direkt auf einem UNIX Rechner ausgeführt werden. Programmiersprachen, welche direkt für einen Prozessor oder ein Betriebssystem kompiliert werden, sind „nativ" kompiliert. Typische Vertreter sind C und C++. Wenn für C# die entsprechende Option beim Kompilieren gewählt wurde, wird es auch nativ kompiliert, ansonsten wird der Code als Common Intermediate Language kompiliert und in einer Laufzeitumgebung ausgeführt.



Java geht hier einen ähnlichen Weg. Bei Java heißt die Laufzeitumgebung „Java Runtime Environment" und bei C# „.NET Runtime Environment". Der Performancenachteil aufgrund des Overheads wird durch Just In Time Compiler größtenteils wett gemacht.

Die dritte, häufig genutzte Form von Architektur ermöglicht die Nutzung von Skriptsprachen. Hier entfällt der Kompiliervorgang, da Skripte direkt interpretiert werden.



Die am häufigsten genutzten Vertreter von Skriptsprachen sind:

- JavaScript
- Python
- PHP
- Visual Basic

2 Klassifikation von höheren Programmiersprachen

Die drei wichtigsten Dimensionen der Einteilung sind:

- Art der computerseitigen Verarbeitung
- Programmierparadigma
- Art der Befehlsstruktur

Bei den Programmierparadigmen unterscheiden wir:

- _____

Programme werden in Unterprogrammen mit (optionalen) Parametern zerlegt. Diese werden als Funktionen (also mit Rückgabewert) und Prozeduren (ohne Rückgabewert) umgesetzt. Zugriffe auf globale Variablen und Konstrukte werden als normale Umsetzungsvarianten angesehen. Vertreter sind C oder COBOL.

- _____

Mathematisch orientierte Modellierung von Funktionen. Programme werden ausschließlich in Funktionen zerlegt. Diese haben keine Seiteneffekte – sie verändern übergebene oder globale Elemente nicht. Funktionen sind als Parameter und Rückgabekonstrukte erlaubt. Globale Elemente werden möglichst ausgeschlossen. Vertreter sind F# oder Haskell. JavaScript und Python eignen sich ebenfalls zur funktionalen Programmierung.

- _____

Klassen modellieren die reale Welt. Objekte als instanziierte Klassen haben Eigenschaften und Methoden (also Funktionen und Prozeduren). Globale Zugriffe werden akzeptiert. Die meisten populären Programmiersprachen arbeiten objektorientiert, bspw. Java, C++, C#, PHP, Python, JavaScript

Die Befehlsstruktur teilt man ein in:

- _____

Das „wie“ steht im Vordergrund. Es wird Schritt für Schritt der Weg zum Ziel formuliert und kommt somit dem eigentlichen Prozessor entgegen. Fast alle gängigen Programmiersprachen sind imperativ.

- _____

„was“ steht im Vordergrund. Der Syntax ist darauf ausgerichtet, was man haben möchte. SQL ist ein typischer Vertreter deklarativer Programmiersprachen, aber auch im funktionalen Bereich finden wir diese Sprachen, wie bspw. Haskell.