Steuerung

1. Generation: Maschinencode

Da man einem Prozessor nicht sagen kann „mach dies, tu das“, muss man mit ihm in einer Sprache kommunizieren, die er versteht. Diese Sprache ist die Maschinensprache (oder auch Maschinencode). Die Maschinensprache besteht aus Einsen und Nullen (1 und 0).
Daher ist es für Menschen schwer, sie zu verstehen, bzw. mit ihr zu arbeiten. Man kann den Maschinencode zwar auch Hexadezimal darstellen, aber auch das ist nicht wirklich verständlich.

2. Generation: Assembler

Um das Programmieren zu vereinfachen kam man bald auf die Idee, den Maschinencode abzukürzen, für bestimmte Gruppen von Nullen und Einsen also Abkürzungen zu verwenden. Diese Abkürzungen werden Mnemonics genannt. So wird aus acht Zeichen langem Binärcode ein Befehl aus zwei bis drei Buchstaben. Diese „Sprachen“ werden Assemblersprachen genannt und die Übersetzungsprogramme nennt man Assembler.
Das Problem mit den Assemblersprachen war allerdings, dass sie bei jedem Prozessor unterschiedlich waren. So kamen mit jedem Prozessor ein paar hundert Befehle, was das Programmieren erschwerte. Ein anderer Nachteil der Assemblersprachen ist, dass es immer noch sehr aufwendig ist, mit ihnen zu programmieren, da man zwar nicht mehr in Maschinensprache schreibt, aber trotzdem noch genau sagen muss, was der Prozessor machen soll.

3. Generation: höhere Programmiersprachen

Um das Programmieren weiter zu vereinfachen erfand man dann die höheren Programmiersprachen (z.B. Java oder Python). In den höheren Programmiersprachen sagt man dem Prozessor nicht mehr genau wie er was machen soll, sondern erstellt „große“ Operationen, die in viele kleine Maschinenoperationen umgewandelt werden.
Programme, die im höheren Programmiersprachen geschrieben wurden werden, um ausgeführt werden zu können in Maschinencode oder Assembler übersetzt. Dazu gibt es zwei Möglichkeiten.

Die erste Möglichkeit ist das Kompilieren. Dabei wird das Programm bevor es ausgeführt wird von einem sogenannten Compiler in Assembler oder Maschinencode übersetzt, um danach ausgeführt zu werden.

Die zweite Möglichkeit ist das interpretieren. Dabei wird das Programm zur Laufzeit von einem Interpreter analysiert und ausgeführt. Das Programm wird also im Gegensatz zum Kompilieren nicht vor der Ausführung in Maschinencode übersetzt, sondern währenddessen analysiert und vom Interpreter ausgeführt.