3. Compiler und Interpreter

Jeder Computer arbeitet nach dem **EVA – Prinzip**, wobei das **E** für die Eingabe, das **V** für die Verarbeitung und das **A** für die Ausgabe steht. Natürlich gilt dieses allgemeine Prinzip auch für die Programmierung. Aus den Eingabedaten erzeugt das Programm Ausgabedaten. Die Vorgänge bei der Verarbeitung sollen im Folgenden etwas genauer betrachtet werden.

Die Eingabe des Quelltextes erfolgt gewöhnlich in einem Editor in einer **höheren Programmiersprache**. Das der Prozessor des Computers diese Anweisungen aber nicht verarbeiten kann, muss der Quelltext vorher in **Maschinensprache** übersetzt werden. Diese Aufgabe übernehmen sogenannte **Übersetzer**. Man unterscheidet nach ihrer Arbeitsweise zwei Arten von Übersetzern: **Compiler** und **Interpreter**.

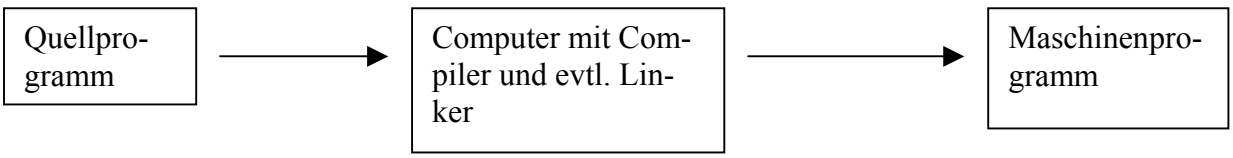
3.1 Interpreter

Der Interpreter führt das in einer höheren Sprache geschriebene Programm nach den notwendigen syntaktischen Überprüfungen sofort aus. Im Gegensatz zu einem Compiler wird das Programm dabei nicht als Ganzes vorher in eine andere Sprache übersetzt. Im Ergebnis liegt deshalb auch kein ausführbares Programm (z.B. eine *.exe – Datei) vor. Die Übersetzung muss daher bei jedem Programmstart erneut vorgenommen werden. Beispiele für solche Programmiersprachen sind BASIC und LISP.
Am einfachsten lässt sich die Arbeitsweise eines Interpreters mit derjenigen eines Dolmetschers vergleichen. Der Dolmetscher muss simultan, d.h. sofort von der einen in die andere Sprache übersetzen. Nach der Übersetzung liegt kein übersetztes Dokument vor. Die Übersetzung müsste jedes mal von Neuem durchgeführt werden.

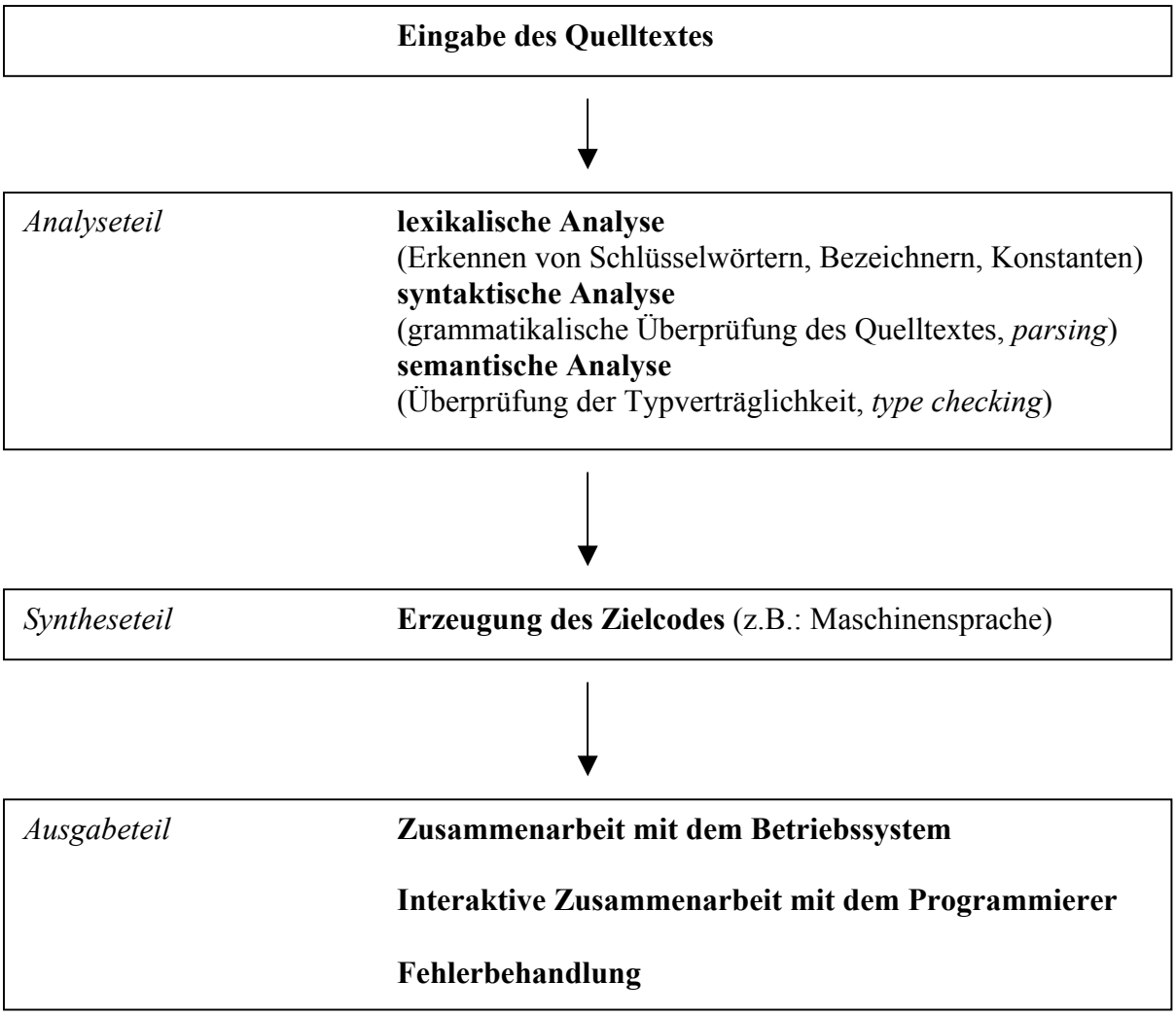
3.2 Compiler

Ein Compiler überträgt das in einer höheren Programmiersprache geschriebene Programm als Ganzes in eine andere Sprache, z.B. in Maschinensprache. Das erzeugte Maschinenprogramm muss nun noch an die speziellen Erfordernisse des jeweiligen Betriebssystems angepasst werden. Diese Aufgabe erledigt der **Linker**. Beispiele für solche Programmiersprachen sind ObjectPascal und C.
Die Arbeitsweise eines Compilers lässt sich mit der Arbeit eines Übersetzers vergleichen, der ein schriftliches Dokument in eine andere Sprache übertragen soll. Nach der Übersetzung liegt das Dokument schriftlich in der gewünschten Übersetzung (ein für alle mal!) vor.

Die folgende Abbildung verdeutlicht die prinzipielle Arbeitsweise eines Compilers.



Die logische Struktur der Arbeitsweise eines Compilers verdeutlicht das folgende Schema:
(Engelmann, u.a.: *Kleiner Leitfaden Informatik*, Paetec – Verlag, Berlin 2000)



„Der Übersetzungsprozess orientiert sich gleichsam an der Struktur des Quelltextes. Dieser wird gemäß der *Syntax* der Sprache in seine Teile zerlegt. An den Teilen ist auch deren Bedeutung (*Semantik*) erkennbar, und die Bedeutung des Gesamten ergibt sich aus der Zusammensetzung der Bedeutung der Teile. Die Bedeutung muss natürlich bei der Übersetzung erhalten bleiben.“[1]

(Niklaus Wirth, Entwickler der Programmiersprache Pascal)

[1] Niklaus Wirth: *Grundlagen und Techniken des Compilerbaus*, Addison – Wesley, 1996