2章
コンパイラは以下の動きで動く
1.読み込み… 原始プログラムを通常レコード単位で読み込む。
2.字句解析… 読み込みによって読み込まれた文字を1字ごとに基本的要素ごとに取り出す。
また、取り出した文字が変数名か定数か演算子かの判別もする。
3.構文解析… プログラムのどの部分が文法のどの規則に対応するかを解析して、プログラムが文法的に正しいかどうか判定する。
中間語が生成される。その形は解析木である。
4.中間語生成
構文解析によって得られた解析木を実行すべき演算の列を作り出す。
5.最適化
目的プログラムを実行時効率の良いものとすること。
6.目的コード生成
中間語のプログラムから目的プログラムを生成する。
尚、2~6に関しては、変数名、定数、などの各種情報が入っている表と連携しながら、コンパイラが実行する。

4章
字句解析は文字読み取りと字句読み取りに分けることができる。
文字読み取りでは空白も含めた文字を読み取る。
そして読み取った文字を(if等の)最小単位の字句に分割する。
また、実際の字句解析において、読み込んだ文字をソースコードに落とすという流れ記述するために便利な以下の手順が一般的である
<バッカス記述>→正規表現→非有限オートマトン→有限オートマトン→ソースコードである。
しかし、私達はNFA以降、利用しない。
正規表現とは
正規表現とは文字列の中から探す文字を決める事である。
また、正規表現をA={a1,a2,...,an}とすると、
正規表現の定義は以下の様になる。
(i)εは正規表現である。
(ii)aiは正規表現である。
(iii)RとSが正規表現ならば、R|S,RS,{R}も正規表現である。

さらに、正規表現Rの値を[R]と書くと正規表現は以下の様に定義される。
(a) [ε]={ε}
(b) [ai]={ai}
(c) [R|S]=[R]U[S]
(d) [RS]=[R][S]
(e) [{R}]=[R]*

これが正規表現である。
文脈自由文法 G は (V_N, V_T, P, S) で定義される
V_N は非終端記号の集合である。
V_T は終端記号の集合である。
P は生成規則の集合であり，生成規則とはGがどのように作られるかの規則である。
記号Sの組である。Sは出発記号であり，Pの中の少なくとも一つの生成規則の左辺に現れる。

前後の単語関係に関係なく、開始記号から初めて非終端文字から終端文字、非終端文字への生成規則の有限集合内にある規則を適用して文字列を生成していき、終端文字に含まれる文字になったら終了になる形式のルールだけで表現できる文法のことである。