基礎から学ぶ Verilog HDL & FPGA 設計

第14回
コンパイラの設計(1)

中野浩嗣，伊藤靖朗

デバイスの記事　ビギナーズ

前回（本誌2009年1月号，pp.99-104）は，学習用CPU「TINYCPU」をターゲットとするアセンブラをPerlで設計した．このアセンブラはニーモニックとラベルで書かれたアセンブリ言語プログラムを16進数の機械語命令列に変換する．今回はさらにTINYCPU向けのプログラミングを容易にするために，C言語風プログラムをアセンブリ言語プログラムに変換するコンパイラを設計する．（筆者）

● “コンパイラ・コンパイラ”を使ってコンパイラを作成

コンパイラは，高級言語により記述されたプログラムをアセンブリ言語プログラムに翻訳（Compile）するプログラムです．コンパイラは，入力された高級言語プログラムに対して，字句解析，構文解析，最適化，コード生成を順に行い，アセンブリ言語プログラムを出力します（**図1**）．ここでは，比較的単純なC言語風言語を対象とし，最適化は行わず，字句解析，構文解析，コード生成のみを行います．

コンパイラの設計は複雑で，C言語などを用いて直接設計するのは困難です．そこで，言語の定義情報を決めれば，コンパイラを自動生成してくれる“コンパイラ・コンパイラ”を用いることにします．ここでは，字句解析プログラムを自動生成する「Flex[1]」と，構文解析プログラムを自動生成する「Bison[2]」の二つです．構文解析プログラムの中で同時にコード生成も行います．

今回は，字句解析と構文解析を理解するために，ごく小規模な学習用のC言語風言語「MICROC」を設計します．そのためにまず，字句解析定義microc.lと構文解析定義microc.yを記述します．これらの定義からFlexとBisonを用いてコンパイラのC言語プログラムが自動的に生成されます．このC言語プログラムを，gccを用いてコンパイルすることにより，MICROCのコンパイラが作られます．

全体の流れを**図2**に示します．字句解析プログラムFlexは，定義ファイルmicroc.lを基に，字句解析を行うC言語プログラムlex.yy.cを生成します．また構文解析プログラムBisonは，定義ファイルmicroc.yを基に，構文解析を行うC言語プログラムy.tab.cとそのヘッダ・ファイルy.tab.hを生成します．

これらのC言語プログラムを，gccを用いてコンパイルすると，MICROCコンパイラが生成されます．このMICROCコンパイラを用いると，C言語風プログラムをアセンブリ言語プログラムに変換できます．アセンブリ言語プログラムは，前回（本誌2009年1月号，pp.99-104）設計したアセンブラを用いて，機械語プログラムを出力できます．

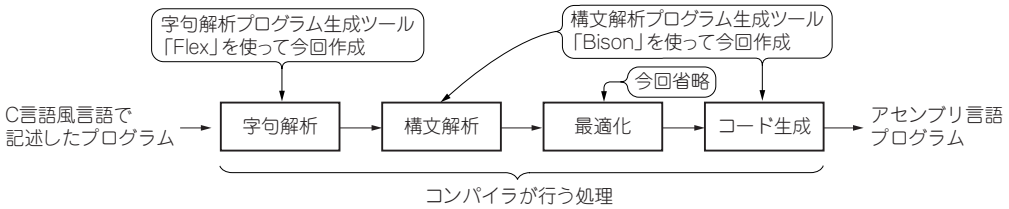

図1
コンパイラが行う処理と今回の設計範囲

Keyword　コンパイラ，コンパイラ・コンパイラ，Flex，Bison，字句解析，構文解析，トークン，文脈自由文法，還元，if，goto，unless

● 学習用のC言語風言語「MICROC」の仕様

まず，字句解析と構文解析を理解するために，ごく小さなC言語風言語MICROCのコンパイラを設計します．MICROC言語仕様を**表1**に示します．

リスト1はMICROCのプログラムの例です．このプログラムをコンパイルすると，**リスト2**のアセンブラ・プログラムが得られます．この処理を行うコンパイラを設計します．

● 簡単な例“sample.l”で字句解析を理解しよう

字句解析は，C言語風プログラムの予約語（goto，ifなど），名前（L1，L2，nなど），2文字以上からなる演算子（==）などの文字列を一つの字句（トークン）としてまとめて扱うように変換します．**リスト3**を例に，字句解析とその定義の記述法を理解しましょう．

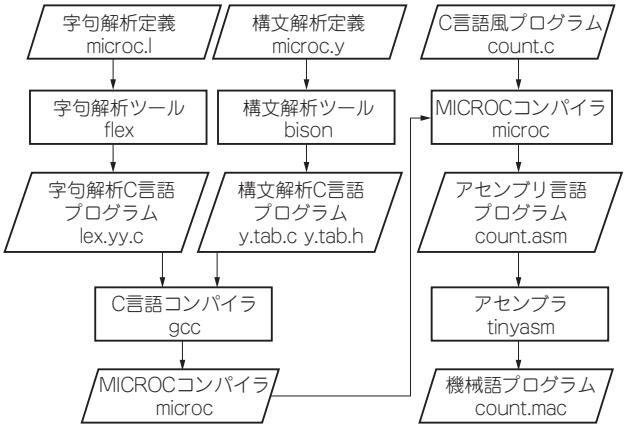

図2　コンパイラ・コンパイラによるコンパイラ生成とコンパイル手順

表1　C言語風言語「MICROC」の言語仕様

項　目	説　明
名　前	英字で始まり英数字が続く．変数名やラベル名に用いる．長さは無制限．
整　数	10進数で表される整数．
ラベル	「名前:」の形で，goto文などの分岐先になる．
変数宣言	「int n=0,m=1;」のように，プログラムで用いる変数とその初期値を宣言する．変数名は名前で，初期値は整数．型は16ビットの整数（2の補数）だけ．初期値を省略すると0で初期化する．
goto文	「goto ラベル;」の形で，指定したラベルに分岐する．
if-goto文とunless-goto文	「if（式）goto ラベル;」と「unless（式）goto ラベル;」の形をしている．if-goto文は式の計算結果が0でないとき，指定したラベルに分岐する．unless-goto文は0のとき，指定したラベルに分岐する．
halt文	機械語プログラムの実行を停止する．
out文	「out（式）;」の形をしており，式を計算し，計算結果を出力する．
代入文	「変数＝式;」の形をしており，式の計算結果を変数に代入する．
式	変数，整数，算術論理演算からなる式．簡単のため，算術論理演算は加算＋，減算－，乗算＊，等しい==の四つとする．

字句解析定義は次に示すように，%%で区切られた三つの部分，宣言部，文法規則部，追加のCプログラム部に分けられます．

リスト3の字句解析定義sample.lでは，1～3行目が宣言部です．この例では，文法規則部でprintfを用いるので，入出力のためのヘッダ・ファイルstdio.hをインクルードしています．

文法規則部は，5行目から16行目です．文法規則部の各行では，文法規則が「正規表現　動作」の形で記述されています．入力を見ながら，正規表現にマッチするかどうか調べ，マッチした場合，その正規表現に対応する動作を行います．5行目は[¥t¥n¥r]だけで動作の定義がないので，スペース（空白），タブ，改行は無視することを意味します．[と¥tの間にスペースがあることに注意してください．6行目は==にマッチしたとき，動作がprintf("[EQ]");なので，[EQ]を出力します．7行目はgotoにマッチした

リスト1　MICROCによるカウントダウン・プログラム（count.c）

```
1    n=in;
2  L1:
3    out(n);
4    unless(n) goto L2;
5    n=n-1;
6    goto L1:
7  L2:
8    halt;
9    int n;
```

リスト2　アセンブリ言語によるカウントダウン・プログラム（count.asm）

```
 1          IN
 2          POP n
 3  L1:
 4          PUSH n
 5          OUT
 6          PUSH n
 7          JZ L2
 8          PUSH n
 9          PUSHI 1
10          SUB
11          POP n
12          JMP L1
13  L2:
14          HALT
15  n: 0
```

リスト3　字句解析定義の例（sample.l）

```
 1  %{
 2  #include <stdio.h>
 3  %}
 4  %%
 5  [ ¥t¥n¥r]
 6  ==      {printf("[EQ]");}
 7  goto    {printf("[GOTO]");}
 8  halt    {printf("[HALT]");}
 9  if      {printf("[IF]");}
10  in      {printf("[IN]");}
11  int     {printf("[INT]");}
12  out     {printf("[OUT]");}
13  unless  {printf("[UNLESS]");}
14  [0-9]+  {printf("[NUMBER(%s)]",yytext);}
15  [a-zA-Z][a-zA-Z0-9]*
              {printf("[NAME(%s)]",yytext);}
16  .       {printf("[%c]",yytext[0]);}
17  %%
18  int yywrap(){ return(1);}
```