コンパイラ実験

学生番号: 09B54923549
中嶋 空偉 (NAKAJIMA, Sorai)

2026 年 1 月 18 日

1　実験の概要と目的

1.1　本実験の概要

この実験では、独自に定義した言語をソース言語として、アセンブリ言語を目的言語として変換するコンパイラを作成する。コンパイラ作成の際には、字句解析を行うプログラムを生成するものとして flex、構文解析を行うプログラムを生成するものとして bison、ast とコード生成に関しては c 言語でプログラムを作成する。

1.2　本実験の目的

3 年間の総仕上げとしてある程度大きなプログラムを作成する。仕様を決め、プログラムを書き、データ構造を決め、アルゴリズムを考え、テストパターンを決め、デバッグするなどの一連の流れを経験する。

2　言語の定義

本コンパイラが受理する言語の文法定義を以下に示す。これは Bison の構文規則から C 言語アクションコードを取り除いたものである。

```
 1: program
 2:     : declarations statements
 3: ;
 4:
 5: declarations
 6:     : decl_statement declarations
 7:     | decl_statement
 8: ;
 9:
10: decl_statement
11:     : "define" IDENT SEMIC
12:     | ARRAY IDENT L_BRACKET NUMBER R_BRACKET SEMIC
13:     | ARRAY IDENT L_BRACKET NUMBER R_BRACKET L_BRACKET NUMBER R_BRACKET SEMIC
14: ;
15:
16: statements
17:     : statement statements
18:     | statement
19: ;
20:
21: statement
```

```
22:     : assignment_stmt
23:     | loop_stmt
24:     | cond_stmt
25:     | expression SEMIC
26: ;
27:
28: assignment_stmt
29:     : IDENT ASSIGN expression SEMIC
30:     | IDENT L_BRACKET expression R_BRACKET ASSIGN expression SEMIC
31:     | IDENT L_BRACKET expression R_BRACKET L_BRACKET expression R_BRACKET ASSIGN expression SEMIC
32: ;
33:
34: expression
35:     : expression add_op term
36:     | term
37: ;
38:
39: term
40:     : term mul_op factor
41:     | factor
42: ;
43:
44: factor
45:     : var
46:     | NUMBER
47:     | L_PAREN expression R_PAREN
48:     | IDENT L_BRACKET expression R_BRACKET
49:     | IDENT L_BRACKET expression R_BRACKET L_BRACKET expression R_BRACKET
50: ;
51:
52: add_op
53:     : PLUS
54:     | MINUS
55: ;
56:
57: mul_op
58:     : MUL
59:     | DIV
60: ;
61:
62: var
63:     : IDENT
64: ;
65:
66: loop_stmt
67:     : WHILE L_PAREN condition R_PAREN L_BRACE statements R_BRACE
68: ;
69:
70: cond_stmt
71:     : IF L_PAREN condition R_PAREN L_BRACE statements R_BRACE ELSE L_BRACE statements R_BRACE
72:     | IF L_PAREN condition R_PAREN L_BRACE statements R_BRACE
73: ;
74:
75: condition
76:     : expression cond_op expression
77: ;
78:
79: cond_op
80:     : EQ
81:     | NE
82:     | LE
83:     | GE
84:     | LT
85:     | GT
86: ;
```

3　受理されるプログラム例

本コンパイラで受理できるプログラムの例を示す。

3.1　基本演算と While ループ

まず，基本的な演算と繰り返し処理の例として，1 から 10 までの和を計算するプログラムを示す。

```
 1: define i;
 2: define sum;
 3: sum = 0;
 4: i = 1;
 5: while (i < 11) {
 6:   sum = sum + i;
 7:   i = i + 1;
 8: }
 9: sum;
```

このプログラムでは，変数 i を 1 から 10 まで 1 ずつ増やしながら，sum に加算していくことで和を求めている。

3.2　条件分岐 (If-Else)

次に，条件分岐の例として，2 つの数値の大小比較を行い，大きい方から小さい方を減算するプログラムを示す。

```
 1: define a;
 2: define b;
 3: a = 10;
 4: b = 5;
 5: if (a > b) {
 6:   a = a - b;
 7: } else {
 8:   a = a + b;
 9: }
10: a;
```

このプログラムでは，if-else 文を使用して条件によって実行する処理を分岐している。本コンパイラでは，==，!=，<=，>=，<，>の比較演算子が利用可能である。

3.3　配列の使用

最後に，配列の例として，3 行 3 列の 2 次元配列の対角成分の和を計算するプログラムを示す。

```
1: array arr[3][3];
2: define sum;
3: define i;
4: arr[0][0] = 1;
5: arr[0][1] = 2;
6: arr[0][2] = 3;
7: arr[1][0] = 4;
```

```
 8: arr[1][1] = 5;
 9: arr[1][2] = 6;
10: arr[2][0] = 7;
11: arr[2][1] = 8;
12: arr[2][2] = 9;
13: sum = 0;
14: i = 0;
15: while (i < 3) {
16:   sum = sum + arr[i][i];
17:   i = i + 1;
18: }
19: sum;
```

このプログラムでは，2 次元配列を宣言し，対角成分（arr[0][0]，arr[1][1]，arr[2][2]）を while ループを用いて順に足し合わせている。1 次元配列と 2 次元配列の両方がサポートされている。

4 コード生成

5 特に工夫した点: 変数の管理方法

6 最終課題のプログラムと実行結果

6.1 1 から 10 までの数の和

6.1.1 プログラム

```
1: define i;
2: define sum;
3: sum = 0;
4: i = 1;
5: while(i < 11) {
6:   sum = sum + i;
7:   i = i + 1;
8: }
```

6.1.2 実行結果

6.2 5 の階乗

6.2.1 プログラム

```
 1: define i;
 2: define fact;
 3:
 4: fact = 1;
 5: i = 1;
 6: while(i < 6) {
 7:   fact = fact * i;
 8:   i = i + 1;
 9: }
10: fact;
```

6.2.2 実行結果

6.3 FizzBuzz

6.3.1 プログラム

```
 1: define fizz;
 2: define buzz;
 3: define fizzbuzz;
 4: define others;
 5: define i;
 6: fizz = 0;
 7: buzz = 0;
 8: fizzbuzz = 0;
 9: others = 0;
10: i = 1;
11: while(i < 31) {
12:   if((i / 15) * 15 == i) {
13:     fizzbuzz = fizzbuzz + 1;
14:   } else {
15:     if((i / 3) * 3 == i) {
16:       fizz = fizz + 1;
17:     } else {
18:       if((i / 5) * 5 == i) {
19:         buzz = buzz + 1;
20:       } else {
21:         others = others + 1;
22:       }
23:     }
24:   }
25:   i = i + 1;
26: }
27: others;
```

6.3.2 実行結果

7 考察