A7-Generation of Intermediate Code using Lex and Yacc

Sneha Sriram Kannan 185001157

16-04-2021

1 Code

tac.y

```
%{
  #include <stdio.h>
  #include <math.h>
    extern FILE * yyin;
  #define YYSTYPE struct node*
  void yyerror();

  struct node {
    char var[3];
    char code[200];
    char true[3];
    char out[3];
  };

  int err_flag = 0;
  int count=0;
  int labelcount=0;
  struct node* newtemp(char id);
%}

%token IF
%token THEN
%token ELSE
%token ENDIF
%token DT
%token ID
%token NUM
%token ARITHOP
%token RELOP
%token ADD
```

```
%token MUL
%token BEG
%token END
%token VAR
%left MUL
%right ADD
%%

START : DECL BEG B END {};

DECL : VAR ID':'DT'='NUM';' DECL {}
  |VAR ID':'DT'='ID';' DECL {}
  |VAR ID':'DT';' DECL {}
  |;


B : IF '('C')' THEN S ELSE S ENDIF
    {
      printf("%s\n%s\ngoto %s\n%s:\n%s%s:",$3->code,$8->code,$3->out,$3->true,$6->code,$3->out);
    };
  |IF '('C')' THEN S ENDIF
    {
      printf("%s\ngoto %s\n%s:\n%s%s:",$3->code,$3->out,$3->true,$6->code,$3->out);
    };

C : ID RELOP ID
    {
      $$ = newtemp('o');
      sprintf($$->true,"L%d",labelcount++);
      sprintf($$->out,"L%d",labelcount++);
      sprintf($$->out,"L%d",labelcount++);
      sprintf($$->code,"if %c %c %c goto %s",$1,$2,$3,$$->true);
    };

S : ID'='E';'
    {
      $$ = newtemp('o');
      sprintf($$->code,"%s\n%c = %s\n", $3->code,$1, $3->var);
    }  ;

E : T MUL E
    {
      $$ = newtemp('t');
            sprintf($$->code,"%s%s\n%s = %s %c %s",$1->code,$3->code,$$->var,$1->var,$2,$3->var);
    }
```

```
  |T
      {
        $$ = $1;
    };

T : T ADD F
    {
      $$ = newtemp('t');
           sprintf($$->code,"%s\n%s = %s %c %s",$1->code,$$->
   var,$1->var,$2,$3->var);
    }
  |F
    {
        $$ = $1;
    };

F : ID
    {
      $$ = newtemp($1);
    };

%%

struct node* newtemp(char id) {
  struct node *temp;
  temp = malloc(sizeof(struct node));
    temp->var[0]=id;
    if(id=='t')
    {
        count++;
        temp->var[1]='0'+count;
        temp->var[2]='\0';
    }
    else
        temp->var[1]='\0';
    strcpy(temp->code,"");
  return temp;
}

void yyerror()
{
  return;
}

void main()
{
    if( !(yyin = fopen("in.txt","r")) ){
        printf("cannot open file\n");exit(1);
    }
```

```
    yyparse();
}
```

tac.l

```
%{
    #include "y.tab.h"
  #include <stdio.h>
    #include <stdlib.h>
  extern YYSTYPE yylval;
%}

%%
"var" {return VAR;}
"integer"|"real"|"char" {return DT;}
"IF"|"if" { return IF;}
"THEN"|"then" { return THEN;}
"ELSE"|"else" { return ELSE;}
"END IF"|"end if" { return ENDIF;}
"begin" { return BEG;}
"end" { return END;}
[0-9]+ { return NUM;}
[a-zA-Z] { yylval = yytext[0]; return ID;}
"+"|"-"  {yylval = yytext[0]; return ADD;}
"*"|"/" {yylval = yytext[0]; return MUL;}
"<="|">="|">"|"<"|"=="|"!=" { yylval = yytext[0];return RELOP;}
[ ] ;
\t  ;
.   {return *yytext;};
%%
int yywrap(){
  return 1;
}
```

2 Output Screenshots

```
1   var i:integer=1;
2   var a:integer=4;
3   var b:integer=3;
4   var c:integer=6;
5   var d:integer=2;
6   var x:integer;
7   begin
8   if(a<b) then
9       x=a+b*c/d;
10  else
11      x=a*b*c-d;
12  end if
13  end
```

Figure 1: Input 1

```
if a < b goto L0

t4 = c - d
t5 = b * t4
t6 = a * t5
x = t6

goto L2
L0:

t1 = a + b
t2 = c / d
t3 = t1 * t2
x = t3
L2:
```

Figure 2:[5]Output 1

```
1   var a:integer=4;
2   var b:integer=3;
3   var c:integer=6;
4   var d:integer=2;
5   var x:integer;
6   begin
7   if(a<b) then
8       x=a+b*c/d;
9   end if
10  end
```

Figure 3: Input 2

```
if a < b goto L0
goto L2
L0:

t1 = a + b
t2 = c / d
t3 = t1 * t2
x = t3
L2:
```

Figure 4: Output 3