SSN COLLEGE OF ENGINEERING, KALAVAKKAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS1602 - Compiler Design

Programming Assignment-9 Implementation of code generation

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Source Code:
Lex file:

```
%{
#include<stdio.h>
#include<stdlib.h>
#include "y.tab.h"
void yyerror(char*);
extern YYSTYPE yylval;

%}


digit   [0-9]
letter  [a-zA-Z]
identifier (_|{letter})(_|{digit}|{letter})* relop(<)|(>)|(<=)|(>=)|(==)|(!=)
arithop(\+)|(\-)|(\*)|(\/)
space(\ )


%%
    if  {yylval.string=strdup(yytext);return ifStmt;} goto
{yylval.string=strdup(yytext);return gotoStmt;}
    L{digit}    {yylval.string=strdup(yytext);return label;} t{digit}
{yylval.string=strdup(yytext);return tempVar;}
    {identifier}        {yylval.string=strdup(yytext);return identifier;}
    {digit}+    {/*printf("%s\n",yytext);*/yylval.string=strdup(yytext);return number;}
    \:  {return *yytext;}
    \n  {return *yytext;}
    {relop} {yylval.string=strdup(yytext);return relop;}
    {arithop} {yylval.string=strdup(yytext);return arithop;}
    =   {return *yytext;}
    {space} {return *yytext;}
%%
```

Yacc file:

```
%{
```

```
#include<stdio.h> #include<stdlib.h> #include<string.h> #include "registersTable.h"
#include "y.tab.h"
int yylex(void); void

yyerror(char*); int yywrap(void);
char* hasRegister(char*);
void checkRelop(char*,char*);

char* checkArithop1(char*,char*,char*); char* checkArithop2(char*,char*,char*); extern
FILE *yyin;
int register_count=0; registers reg[20];
%}


%union
{ char
*string; int num;
};


%token <string> ifStmt gotoStmt label tempVar identifier relop arithop number


%type <string> Var ArithExpr


%%

S:  S Line

| Line
;


Line:   label ':' ' ' Var '=' number '\n' {




char *newReg=hasRegister($4);


if(newReg==NULL)



{




newReg=(char*)malloc(sizeof(char)*10);
```

```
sprintf(newReg,"R%d",register_count);



reg[register_count].var=strdup($4);



reg[register_count].registerName=strdup(newReg);

register_count+
+;




}



printf("MOV %s,#%s\n",newReg,$6);



}
|Var '=' number '\n'

{ char *newReg=hasRegister($1); if(newReg==NULL)
{



newReg=(char*)malloc(sizeof(char)*10); sprintf(newReg,"R

%d",register_count); reg[register_count].var=strdup($1);
reg[register_count].registerName=strdup(newReg); register_count++;
}




printf("MOV %s,#%s\n",newReg,$3);

}
```

```
$1,hasRegister($4),hasRegister($6));









}
| label ':' ' ' Var '=' tempVar '\n' {


//printf("%s: MOV %s,%s\n",









|Var '=' tempVar '\n' {

//printf("MOV %s,%s\n",hasRegister($1),hasRegister($3));









}
| label ':' ' ' Var '=' ArithExpr '\n' {


printf("%s: %s",$1,$6);
```

```
}
|Var '=' ArithExpr '\n'

{ printf("%s",$3);


}
| label ':' ' ' ifStmt ' ' Var relop number ' ' gotoStmt ' '


{

*newReg=(char*)malloc(sizeof(cha r)*10);

%d",register_count);


%s\n",newReg,$8);

*reg=hasRegister($6);

%s\n",$1,reg,newReg);


char
```

```
}




sprintf(newReg,"R


register_count++; printf("MOV %s,#


char

printf("%s: CMP %s, checkRelop($7,$12);

'\n'


*newReg=hasRegister($8);


*reg=hasRegister($6);


%s\n",$1,reg,newReg);
| label ':' ' ' ifStmt ' ' Var relop Var ' ' gotoStmt ' ' label


{







}

|gotoStmt ' ' label '\n' {


char char

printf("%s: CMP %s,


checkRelop($7,$12);
```

```
printf("JMP %s\n",$3);;

}
;




ArithExpr:  Var arithop Var {

$
$=strdup(checkArithop1($2,$1,$ 3));



number {

| Var arithop

$
$=strdup(checkArithop2($2,$1,$3));



}
;

Var: identifier {$$=strdup($1);}



%%

| tempVar   {$$=strdup($1);}

;


char* hasRegister(char* var) {
for(int i=0;i<register_count;i++)
{
if(strcmp(var,reg[i].var)==0)
{
return (char*)reg[i].registerName;
}
}
return NULL;
```

```
}


void checkRelop(char* relop1,char* label1)
{ if(strcmp(relop1,">")==0)
{

printf("JGT %s\n",label1);
} else if(strcmp(relop1,">=")==0)

{
printf("JGE %s\n",label1);
} else if(strcmp(relop1,"<")==0)
{
printf("JLT %s\n",label1);
} else if(strcmp(relop1,"<=")==0)
{
printf("JLE %s\n",label1);
} else if(strcmp(relop1,"!=")==0)
{
printf("JNE %s\n",label1);
} else if(strcmp(relop1,"==")==0)
{

printf("JE %s\n",label1);
}
}


char* checkArithop1(char* arithop1,char* var1,char* var2) { char
*code=(char*)malloc(sizeof(char)*128);
//printf("*Here\n"); if(strcmp(arithop1,"+")
==0)
{

sprintf(code,"ADD %s,%s\ n",hasRegister(var1),hasRegister(var2));

} else if(strcmp(arithop1,"-")==0)


{
sprintf(code,"SUB %s,%s\n",hasRegister(var1),hasRegister(var2));
} else if(strcmp(arithop1,"*")==0)
{

sprintf(code,"MUL %s,%s\ n",hasRegister(var1),hasRegister(var2));
} else if(strcmp(arithop1,"/")==0)
{

sprintf(code,"DIV %s,%s\n",hasRegister(var1),hasRegister(var2));
}
return code;
}
```

```
char* checkArithop2(char* arithop1,char* var1,char* num) { char
*newReg=(char*)malloc(sizeof(char)*10); sprintf(newReg,"R%d",register_count);
register_count++;
//printf("Here*\n");
char *code=(char*)malloc(sizeof(char)*128);
//printf("Here**\n");
char *code1=(char*)malloc(sizeof(char)*64); sprintf(code1,"MOV %s,#%s\ n",newReg,num);
//printf("Here***\n"); if(strcmp(arithop1,"+")
==0)
{

sprintf(code,"%sADD %s,%s\ n",code1,hasRegister(var1),newReg);
} else if(strcmp(arithop1,"-")==0)
{

sprintf(code,"%sSUB %s,%s\n",code1,hasRegister(var1),newReg);
} else if(strcmp(arithop1,"*")==0)
{
sprintf(code,"%sMUL %s,%s\ n",code1,hasRegister(var1),newReg);
} else if(strcmp(arithop1,"/")==0)

{


}
return code;
}


sprintf(code,"%sDIV %s,%s\n",code1,hasRegister(var1),newReg);



void yyerror(char *str)
{
fprintf(stderr,"%s\ n",str); return;
}


int yywrap()
{
return 1;
}


int main(int argc,char *argv[])
{

yyin=fopen(argv[1],"r");yyparse(); return 0;


}
```

Header file used:

```
typedef struct registers
{
    char *var;
    char *registerName;
} registers;
```

Output

```
MOV R0,#0
MOV R1,#1
MOV R2,#10
L3: CMP R1,R2
JLE L1
JMP L2
L1: ADD R0,R1
MOV R3,#1
ADD R1,R3
JMP L3
```

RESULT:

Machine code for given three address code is generated successfully.