%{

#include<ctype.h>

#include<stdio.h>

#include "y.tab.h"

%}

Digit [0-9]+

Letter [a-zA-Z]+

%% //end of declaration section

{Digit} {yylval.p=(char)yytext[0];return Digit;}

{Letter} {yylval.p=(char)yytext[0];return Letter;}

. {return yytext[0];}

\n {return 0;}

%% //end of rules section

%{

#include<stdio.h>

#include<string.h>

char addintotable(char,char,char);

int search(char);

struct icode

{

char op1;

char op2;

char op;

char res;

}code[20];

int index1=0;

%}

%union

{

char p;

}

%token <p> Letter Digit

%type <p> expr

%left '-' '+'

%left '\*' '/'

%% //end of declaration section

stat:Letter '=' expr ';' {addintotable((char)$1,(char)$3,'=');}

|expr ';'

;

expr:expr '+' expr {$$=addintotable((char)$1,(char)$3,'+');}

|expr '-' expr {$$=addintotable((char)$1,(char)$3,'-');}

|expr '\*' expr {$$=addintotable((char)$1,(char)$3,'\*');}

|expr '/' expr {$$=addintotable((char)$1,(char)$3,'/');}

|'('expr')' {$$=(char)$2;}

|Digit {$$=(char)$1;}

|Letter {$$=(char)$1;}

;

%% //end of rules section

int yywrap()

{

return 1;

}

yyerror(char \*s)

{

printf("Sammyak yaar!! error %s",s);

}

char temp='A';

char addintotable(char op1,char op2,char op)

{

temp++;

code[index1].op1=op1;

code[index1].op2=op2;

code[index1].op=op;

code[index1].res=temp;

index1++;

return temp;

}

void threeaddress()

{

int cnt=0;

char temp='A';

temp++;

printf("\nThe three address code is:\n");

printf("");

while(cnt<index1)

{

printf("%c:=\t",temp);

printf("%c\t",code[cnt].op1);

printf("%c\t",code[cnt].op);

printf("%c\t",code[cnt].op2);

temp++;

cnt++;

printf("\n");

}

}

void quadruple()

{

int cnt=0;

char temp='A';

temp++;

printf("\nThe Quadruple format is:\n");

printf("Operator\tArg1\tArg2\tResult\n");

while(cnt<index1)

{

printf("%c\t\t",code[cnt].op);

printf("%c\t",code[cnt].op1);

printf("%c\t",code[cnt].op2);

printf("%c\t",code[cnt].res);

temp++;

cnt++;

printf("\n");

}

}

void triple()

{

int cnt=0;

int flag;

char temp='A';

temp++;

printf("\nThe Triple format is:\n");

printf("Arg1\tArg2\toperator\n");

while(cnt<index1)

{

if(isalpha(code[cnt].op1) && isupper(code[cnt].op1))

{

flag= search(code[cnt].op1);

printf("%d\t",flag);

}

else if(isalpha(code[cnt].op1))

printf("%c\t",code[cnt].op1);

else

printf("%c\t",temp);

if(isalpha(code[cnt].op2) && isupper(code[cnt].op2))

{

flag= search(code[cnt].op2);

printf("%d\t",flag);

}

else if(isalpha(code[cnt].op2))

printf("%c\t",code[cnt].op2);

else

printf("%c\t",temp);

printf("%c\t",code[cnt].op);

printf("\n");

cnt++;

}

}

int search(char find)

{

int i;

for(i=0;i<index1;i++)

{

if(code[i].res==find)

return(i);

}

return 0;

}

void main()

{

yyparse();

threeaddress();

quadruple();

triple();

}

OUTPUT :

/\*

sammyak@sammyak-Inspiron-3542:~$ lex a5.l

sammyak@sammyak-Inspiron-3542:~$ yacc -d a5.y

sammyak@sammyak-Inspiron-3542:~$ cc lex.yy.c y.tab.c -w

sammyak@sammyak-Inspiron-3542:~$ ./a.out

a=b+c-d;

The three address code is:

B:= b + c

C:= B - d

D:= a = C

The Quadruple format is:

Operator Arg1 Arg2 Result

+ b c B

- B d C

= a C D

The Triple format is:

Arg1 Arg2 operator

b c +

0 d -

a 1 =

sammyak@sammyak-Inspiron-3542:~$