**Practical: 8**

**Program: Program for intermediate code generation.**

#include<stdio.h>

#include<math.h>

#include<string.h>

# include<ctype.h>

#include<stdlib.h>

// How to run

// Compile: gcc -o codeOptimization codeOptimization.c -lm

// Run: ./codeOptimization

struct quad

{

char ope[5];

char arg1[5];

char arg2[5];

char res[5];

}QUAD[5];

int i=0,n,c=0;

void get()

{

printf("\nEnter no of lines in a block");

scanf("%d",&n);

printf("enter ICG in form operator arg1 arg2 result:");

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

scanf("%s\n%s\n%s\n%s",&QUAD[i].ope,&QUAD[i].arg1,&QUAD[i].arg2,&QUAD[i].res);

}

void const\_folding()

{

int j,c1=0,d=0;

char ch[5],ch1[5],num[10];

int flag1 =1, flag2 =1;

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

{

flag1 =1;flag2 =1;

for (j=0;j<strlen(QUAD[i].arg1);j++)

{

if(!isdigit(QUAD[i].arg1[j]))

{ flag1 = 0;printf("Operand1 is not contstant, Constant folding can not applied to quadruple %d\n",i);

break;

}

}

for (j=0;j<strlen(QUAD[i].arg2);j++)

{

if(!isdigit(QUAD[i].arg2[j]))

{ flag2 = 0; printf("Operand2 is not contstant, Constant folding can not applied to quadruple %d\n",i);

break;

}

}

if(flag1 == 1 && flag2 ==1)

{

c=atoi(QUAD[i].arg1);

c1=atoi(QUAD[i].arg2);

if(strcmp(QUAD[i].ope,"\*")==0)

{

d=c\*c1;

//itoa(d,ch,10);

snprintf(ch, 10, "%d", d);

strcpy(QUAD[i].ope,"=");

strcpy(QUAD[i].arg1,ch);

strcpy(QUAD[i].arg2,"\0");

}

if(strcmp(QUAD[i].ope,"/")==0)

{

d=c/c1;

//itoa(d,ch,10);

snprintf(ch, 10, "%d", d);

strcpy(QUAD[i].ope,"=");

strcpy(QUAD[i].arg1,ch);

strcpy(QUAD[i].arg2,"\0");

}

if(strcmp(QUAD[i].ope,"+")==0)

{

d=c+c1;

//itoa(d,ch,10);

snprintf(ch, 10, "%d", d);

strcpy(QUAD[i].ope,"=");

strcpy(QUAD[i].arg1,ch);

strcpy(QUAD[i].arg2,"\0");

}

if(strcmp(QUAD[i].ope,"-")==0)

{

d=c-c1;

//itoa(d,ch,10);

snprintf(ch, 10, "%d", d);

strcpy(QUAD[i].ope,"=");

strcpy(QUAD[i].arg1,ch);

strcpy(QUAD[i].arg2,"\0");

}

}

}

}

void strength\_reduction()

{

int j=0,n1=0,m=0,c=0,tempo=0,t=0;

char ch[5],cc[5],ct[2],pres[5];

int flag;

strcpy(ct,"s");

for(i=0;i<n;i++){

c=0;

if(strcmp(QUAD[i].ope,"\*")==0||strcmp(QUAD[i].ope,"/")==0)

{ j = 1;

if(strcmp(QUAD[i].ope,"\*")==0)

flag =0;

else

flag =1;

if((atoi(QUAD[i].arg2))>0)

{

m=atoi(QUAD[i].arg2);

while(n1<=m)

{

n1=pow(2,j);

j++;

}

j=j-2;

n1=pow(2,j);

c=m-n1;

printf("number! is 2^%d + %d",j,c);

if(c==0)

{

//itoa(j,ch,10);

snprintf(ch, 10, "%d", j);

if(flag==0)

strcpy(QUAD[i].ope,"<<");

else

strcpy(QUAD[i].ope,">>");

// strcpy(QUAD[i].arg1,ch);

strcpy(QUAD[i].arg2,ch);

// strcpy(QUAD[i].res,"t2");

}

else

{

strcpy(pres,QUAD[i].res);

//itoa(j,ch,10);

snprintf(ch, 10, "%d", j);

if(flag==0)

strcpy(QUAD[i].ope,"<<");

else

strcpy(QUAD[i].ope,">>");

strcpy(QUAD[i].arg2,ch);

// strcpy(QUAD[i].res,"t2");

i++;

for(t=0;t<c;t++)

{

for(j=n;j>=i;j--)

QUAD[j+1] = QUAD[j];

if(c==1)

{

//itoa(c,ch,10);

snprintf(ch, 10, "%d", j);

if(flag==0)

strcpy(QUAD[i].ope,"+");

else

strcpy(QUAD[i].ope,"-");

tempo=i-1;

strcpy(QUAD[i].arg1,QUAD[tempo].res);

strcpy(QUAD[i].arg2,ch);

//itoa(i,cc,10);

snprintf(cc, 10, "%d", i);

strcat(ct,cc);

printf("CT is %s",ct);

strcpy(QUAD[i].res,ct);

}

else

{

strcpy(ct,"s");

//itoa(c-(c-1),ch,10);

snprintf(ch, 10, "%d", c-(c-1));

if(flag==0)

strcpy(QUAD[i].ope,"+");

else

strcpy(QUAD[i].ope,"-");

tempo=i-1;

strcpy(QUAD[i].arg1,QUAD[tempo].res);

strcpy(QUAD[i].arg2,ch);

//strcat("t",i);

//itoa(i,cc,10);

snprintf(cc, 10, "%d", i);

strcat(ct,cc);

strcpy(QUAD[i].res,ct);

}

i++;

n=n+1;

}

}

}

}

printf("n value =%d\n",n);

for(j=i;j<n;j++)

{

if(strcmp(QUAD[j].arg1, pres) ==0)

strcpy(QUAD[j].arg1,QUAD[i-1].res);

else if (strcmp(QUAD[j].arg2, pres) ==0)

strcpy(QUAD[j].arg2,QUAD[i-1].res);

}

if(c!=0)

i = i-1;

}

}

void disp()

{

printf("\nQuadraple\noperator\targ1\targ2\tresult\n");

printf("n value is %d\n",n);

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

printf("\t%s\t%s\t%s\t%s\n",QUAD[i].ope,QUAD[i].arg1,QUAD[i].arg2,QUAD[i].res);

}

void main()

{

get();

disp();

const\_folding();

printf("Quadruples after constant folding\n");

disp();

strength\_reduction();

printf("Quadruples after strength reduction\n");

disp();

}

**Output:**

