CODE OPTIMISATION

Mohammed Farhan
S7CSE-B18

```
#include<stdio.h>
#include<math.h>
#include<string.h>
void push(char);
char pop(void);
int ISP(char);
int ICP(char);
int finder(char a);
char S[100],expr[100],expr1[100],post[100],o,REGIS[100];
int top,max,n,num=0,rnum=0;
char item,x,y,j,res='0',rnumc='0';
FILE *f1;

struct inter
{
  char operator, arg1,arg2,result;
}INTER[10];

main()
{
  int i,j,k=0,flag=0;
  top=-1,max=50;

  printf("\n Enter the Infix expression : ");
  scanf("%s",expr);

  for(i=0;expr[i]!='\0';i++)
    {
      if(expr[i]=='=')
      break;
    }

  if(expr[i]!='=')
    strcpy(expr1,expr);
  else
    {
      flag=1;
      for(j=i+1;expr[j]!='\0';j++,k++)
      expr1[k]=expr[j];
      expr1[k]='\0';
    }

  for(i=0;expr1[i]!='\0';i++);
  expr1[i]=')';
  expr1[i+1]='\0';

  push('(');
  i=0,j=0;

  while(top>-1)
    {
      x=pop();
      item=expr1[i];

      if(isalpha(item))
      {
        push(x);
        post[j]=item;
        i++,j++;
      }
      else if(item==')')
      while(x!='(')
        {
          post[j]=x;
          i++,j++;
          x=pop();
        }
      else if((item=='+')||(item=='-')||(item=='*')||(item=='/')||(item=='^')||
(item=='('))
      if(ISP(x)>=ICP(item))
        {
          while(ISP(x)>=ICP(item))
            {
            post[j]=x;
            j++;
            x=pop();
            }
          push(x);
          push(item);
          i++;
        }
      else
        {
          push(x);
          push(item);
          i++;
        }
    }
  post[j]='\0';
  printf("\n The Postfix expression is :- ");
  if(flag==1)
    printf("%c%s=",expr[0],post);
  else
    printf("%s",post);

  top=-1;
  n=0,i=0;

  for(i=0;post[i]!='\0';i++);

  if(flag==1)
    {
      push(expr[0]);
      post[i]='=';
      post[i+1]='#';
      post[i+2]='\0';
    }
  else
    {
      post[i]='#';
      post[i+1]='\0';
     }
f1=fopen("10out","w");
  i=0;
 fprintf(f1,"\n.data\n");
 if(flag==1)
 {
      fprintf(f1,"\t%c\tdb\t?\n",expr[0]);
 }
while(post[i]!='#')
{
      if(isalpha(post[i]))
      {
            fprintf(f1,"\t%c\tdb\t?\n",post[i]);

      }
      i++;
}
fprintf(f1,".code\n");
i=0;
while(post[i]!='#')
{
      if(isalpha(post[i]))
      {
            for(k=0;k<rnum;k++)

                  if(REGIS[k]==post[i])
                        break;
                  if(k==rnum)
                        {
                              REGIS[k]=post[i];
                              fprintf(f1,"\tLD R%d,%c\n",k,post[i]);
                              rnum++;
                        }
                  push(post[i]);


            i++;
      }
      else if(post[i]=='+')
      {
            x=pop();
            y=pop();
            fprintf(f1,"\tADD R%d,R%d\n",finder(y),finder(x));
            push(rnumc);
            REGIS[finder(y)]=rnumc;
            rnumc++;
            i++;
      }
      else if(post[i]=='-')
      {
            x=pop();
            y=pop();
            fprintf(f1,"\tSUB R%d,R%d\n",finder(y),finder(x));
            push(rnumc);
            REGIS[finder(y)]=rnumc;
            rnumc++;
            i++;
      }
      else if(post[i]=='*')
      {
            x=pop();
            y=pop();
            fprintf(f1,"\tMUL R%d,R%d\n",finder(y),finder(x));
            push(rnumc);
            REGIS[finder(y)]=rnumc;
            rnumc++;
            i++;
      }
      else if(post[i]=='/')
      {
            x=pop();
            y=pop();
            fprintf(f1,"\tDIV R%d,R%d\n",finder(y),finder(x));
            push(rnumc);
            REGIS[finder(y)]=rnumc;
            rnumc++;
            i++;
      }
      else if(post[i]=='=')
      {
            x=pop();
            fprintf(f1,"\tMOV %c,R%d\n",expr[0],finder(x));
            i++;
      }


}
printf("Code Generated\n");
fclose(f1);
}
int ISP(char expr1)
{
  if(expr1=='^')
    return(3);
  if((expr1=='*')||(expr1=='/'))
    return(2);
  if((expr1=='+')||(expr1=='-'))
    return(1);
  if(expr1=='(')
    return(0);
}

int ICP(char expr1)
{
  if(expr1=='^')
    return(4);
  if((expr1=='*')||(expr1=='/'))
    return(2);
  if((expr1=='+')||(expr1=='-'))
    return(1);
  if(expr1=='(')
    return(4);
}
```

```
void push(char expr1)
{
  top++;
  S[top]=expr1;
}

char pop(void)
{
  char a;
  a=S[top];
  top--;
  return(a);
}
int finder(char a)
{
      int i;
      for(i=0;i<=rnum;i++)
      {
            if(REGIS[i]==a)
                  return(i);
      }
      printf("Error. Exiting %c",a);
      exit(0);

}
```

OUTPUT

```
42813@user:/mnt/42813/compiler/intermediate$ ./a.out

 Enter the Infix expression : a=b*c

 The Postfix expression is :- abc*=Code Generated




.data
      a     db    ?
      b     db    ?
      c     db    ?
.code
      LD R0,b
      LD R1,c
      MUL R0,R1
      MOV a,R0
```