

Exno 8 Implement the backend of the compiler to produce three address code generation

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>

struct three
{
char data[10],temp[7];
}s[30];
void main()
{
char d1[7],d2[7]="t";
int i=0,j=1,len=0;
FILE *f1,*f2;
clrscr();
f1=fopen("sum.txt","r");
f2=fopen("out.txt","w");
while(fscanf(f1,"%s",s[len].data)!=EOF)
len++;
itoa(j,d1,7);
strcat(d2,d1);
strcpy(s[j].temp,d2);
strcpy(d1,"");
strcpy(d2,"t");
if(!strcmp(s[3].data,"+"))
{
fprintf(f2,"%s=%s+%s",s[j].temp,s[i+2].data,s[i+4].data);
j++;
}
else if(!strcmp(s[3].data,"-"))
{
fprintf(f2,"%s=%s-%s",s[j].temp,s[i+2].data,s[i+4].data);
j++;
}
for(i=4;i<len-2;i+=2)
{
itoa(j,d1,7);
strcat(d2,d1);
```

```

strcpy(s[j].temp,d2);
if(!strcmp(s[i+1].data,"+"))
fprintf(f2,"\n%s=%s+%s",s[j].temp,s[j-1].temp,s[i+2].data);
else if(!strcmp(s[i+1].data,"-"))
fprintf(f2,"\n%s=%s-%s",s[j].temp,s[j-1].temp,s[i+2].data);
strcpy(d1,"");
strcpy(d2,"t");
j++;
}
fprintf(f2,"\n%s=%s",s[0].data,s[j-1].temp);
fclose(f1);
fclose(f2);
getch();
}

```

Input: sum.txt

out = in1 + in2 + in3 - in4

Output : out.txt

```

t1=in1+in2
t2=t1+in3
t3=t2-in4
out=t3

```

Exno 9 Symbol Table

ALGORITHM:

Start the program for performing insert, display, delete, search and modify option in symbol table

Define the structure of the Symbol Table

Enter the choice for performing the operations in the symbol Table

If the entered choice is 1, search the symbol table for the symbol to be inserted. If the symbol is

already present, it displays “Duplicate Symbol”. Else, insert the symbol and the corresponding address in the symbol table.

If the entered choice is 2, the symbols present in the symbol table are displayed.

If the entered choice is 3, the symbol to be deleted is searched in the symbol table. If it is not found in the symbol table it displays “Label Not found”. Else, the symbol is deleted.

If the entered choice is 5, the symbol to be modified is searched in the symbol table.

Program

```
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
void main()
{
    int i=0,j=0,x=0,n;
    void *p,*add[5];
    char ch,srch,b[15],d[15],c;
    printf("Expression terminated by $:");
    while((c=getchar())!='$')
    {
        b[i]=c;
        i++;
    }
    n=i-1;
    printf("Given Expression:");
    i=0;
    while(i<=n)
    {
        printf("%c",b[i]);
        i++;
    }
    printf("\n Symbol Table\n");
    printf("Symbol \t addr \t type");
    while(j<=n)
    {
        c=b[j];
        if(isalpha(toascii(c)))
        {
            p=malloc(c);
            add[x]=p;
            d[x]=c;
        }
    }
}
```

```

printf("\n%c \t %d \t identifier\n",c,p);
x++;
j++;
}
else
{
ch=c;
if(ch=='+' || ch=='-' || ch=='*' || ch=='=')
{
p=malloc(ch);
add[x]=p;
d[x]=ch;
printf("\n %c \t %d \t operator\n",ch,p);
x++;
j++;
}}}}

```

Input :

Expression terminated by \$:A+B=C \$

Output:

Given Expression:A+B=C

Symbol Table

Symbol	addr	type
A	22014656	identifier
+	22014736	operator
B	22014800	identifier
=	22014880	operator
C	22014960	identifier

Exno 11 Implementation of code optimization Techniques

```
#include<stdio.h>
#include<string.h>
struct op
{
char l;
char r[20];
}
op[10],pr[10];
void main()
{
int a,i,k,j,n,z=0,m,q;
char *p,*l;
char temp,t;
char *tem;
printf("Enter the Number of Values:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("left: ");
scanf(" %c",&op[i].l);
printf("right: ");
scanf(" %s",&op[i].r);
}
printf("Intermediate Code\n") ;
for(i=0;i<n;i++)
{
printf("%c=",op[i].l);
printf("%s\n",op[i].r);
}
for(i=0;i<n-1;i++)
{
temp=op[i].l;
for(j=0;j<n;j++)
{
p=strchr(op[j].r,temp);
if(p)
{
pr[z].l=op[i].l;
strcpy(pr[z].r,op[i].
```

```

r);
z++;
}
}
}
pr[z].l=op[n-1].l;
strcpy(pr[z].r,op[n-1].r);
z++;
printf("\nAfter Dead Code Elimination\n");
for(k=0;k<z;k++)
{
printf("%c\t=",pr[k].l);
printf("%s\n",pr[k].r);
}
for(m=0;m<z;m++)
{
tem=pr[m].r;
for(j=m+1;j<z;j++)
{
p=strstr(tem,pr[j].r);
if(p)
{
t=pr[j].l;
pr[j].l=pr[m].l;
for(i=0;i<z;i++)
{
l=strchr(pr[i].r,t) ;
if(l)
{
a=l-pr[i].r;
printf("pos: %d\n",a);
pr[i].r[a]=pr[m].l;
}}}}
printf("Eliminate Common Expression\n");
for(i=0;i<z;i++)
{
printf("%c\t=",pr[i].l);
printf("%s\n",pr[i].r);
}
for(i=0;i<z;i++)
{
for(j=i+1;j<z;j++)

```

```

{
q=strcmp(pr[i].r,pr[j].r);
if((pr[i].l==pr[j].l)&&!q)
{
pr[i].l='\0';
}
}
}
printf("Optimized Code\n");
for(i=0;i<z;i++)
{
if(pr[i].l!='\0')
{
printf("%c=",pr[i].l);
printf("%s\n",pr[i].r);
}
}
}

```

INPUT & OUTPUT:

Enter the Number of Values:5

left: a right: 9

left: b right: c+d

left: e right: c+d

left: f right: b+e

left: r right: f

Intermediate Code

a=9

b=c+d

e=c+d

f=b+e

r=f

After Dead Code Elimination nbt=c+dnft=b+enrt=fnpos:

Eliminate Common Expression

b =c+d

b =c+d

f =b+b

r =f

Optimized Code

b=c+d

$$f=b+b$$

$$r=f$$