Experiment Number 12

Aim: A simple code Generator.

Algorithm:

Step 1: Start

Step 2: Read code.

Step 3: Go from left to right.

Step 4: Identify operations and operands.

Step 5: Generate code and print.

Step 6: Stop.

Code:

```
#include<bits/stdc++.h>
typedef struct
{ char var[10];
int alive;
}
regist;
regist preg[10];
void substring(char exp[],int st,int end)
{ int i,j=0;
char dup[10]="";
for(i=st;i<end;i++)
dup[j++]=exp[i];
dup[j]='0';
strcpy(exp,dup);
} int getregister(char var[])
{ int i;
for(i=0;i<10;i++)
{
if(preg[i].alive==0)
{
strcpy(preg[i].var,var);
break;
}}
return(i);
}
void getvar(char exp[],char v[])
{ int i,j=0;
char var[10]="";
for(i=0;exp[i]!='\0';i++)
```

```
if(isalpha(exp[i]))
var[j++]=exp[i];
else
break;
strcpy(v,var);
}
int main()
{ char basic[10][10],var[10][10],fstr[10],op;
int i,j,k,reg,vc,flag=0;
printf("\nEnter the Three Address Code:\n");
for(i=0;;i++)
{
gets(basic[i]);
if(strcmp(basic[i],"exit")==0)
break;
}
printf("\nThe Equivalent Assembly Code is:\n");
for(j=0;j<i;j++)
{
getvar(basic[j],var[vc++]);
strcpy(fstr,var[vc-1]);
substring(basic[j],strlen(var[vc-1])+1,strlen(basic[j]));
getvar(basic[j],var[vc++]);
reg=getregister(var[vc-1]);
if(preg[reg].alive==0)
{
printf("\nMov R%d,%s",reg,var[vc-1]);
preg[reg].alive=1;
}
op=basic[j][strlen(var[vc-1])];
substring(basic[j],strlen(var[vc-1])+1,strlen(basic[j]));
getvar(basic[j],var[vc++]);
switch(op)
{ case '+': printf("\nAdd"); break;
case '-': printf("\nSub"); break;
case '*': printf("\nMul"); break;
case '/': printf("\nDiv"); break;
}
flag=1;
for(k=0;k<=reg;k++)
{ if(strcmp(preg[k].var,var[vc-1])==0)
{
printf("R%d, R%d",k,reg);
preg[k].alive=0;
flag=0;
break;
```

```
}} if(flag)
{
printf(" %s,R%d",var[vc-1],reg);
printf("\nMov %s,R%d",fstr,reg);
}strcpy(preg[reg].var,var[vc-3]);
}}
```

Output:

```
Enter the Three Address Code:
a=b+c
c=c*d
exit

The Equivalent Assembly Code is:

Mov R0,b
Add c,R0
Mov a,R0
Mov R1,c
Mul d,R1
Mov c,R1
```

Result: Thus, a simple code Generator implemented successfully.