Practical 9

Name: Rajatkumar Patel
Roll No.: 18BCE191

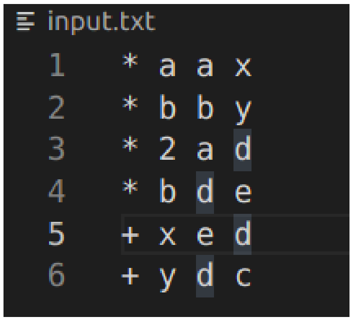
Aim

To implement assembly code generator

Code

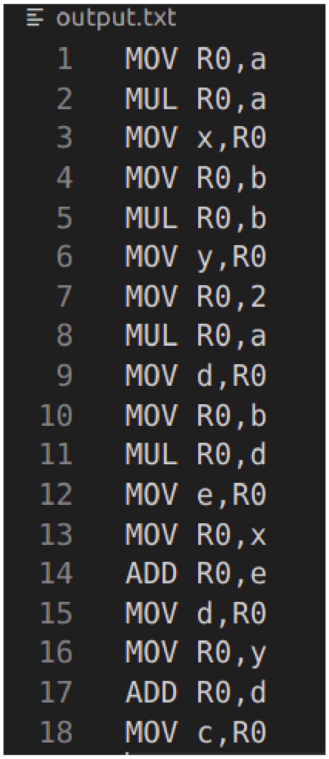
```
#include<stdio.h>
#include<string.h>
char op[2],arg1[5],arg2[5],result[5];
void main()
{
 FILE *fp1,*fp2;
 fp1=fopen("input.txt","r");
 fp2=fopen("output.txt","w");
 while(!feof(fp1))
 {

   fscanf(fp1,"%s%s%s%s",op,arg1,arg2,result);
   printf("%s %s %s %s",op,arg1,arg2,result);
   if(!strcmp(op,"+"))
   {
     fprintf(fp2,"MOV R0,%s",arg1);
     fprintf(fp2,"\nADD R0,%s",arg2);
     fprintf(fp2,"\nMOV %s,R0",result);
   }
   else if(!strcmp(op,"-"))
   {
     fprintf(fp2,"MOV R0,%s",arg1);
     fprintf(fp2,"\nSUB R0,%s",arg2);
     fprintf(fp2,"\nMOV %s,R0",result);
   }
   else if(!strcmp(op,"*"))
   {
     fprintf(fp2,"MOV R0,%s",arg1);
     fprintf(fp2,"\nMUL R0,%s",arg2);
     fprintf(fp2,"\nMOV %s,R0",result);
   }
   else if(!strcmp(op,"/"))
   {
     fprintf(fp2,"MOV R0,%s",arg1);
     fprintf(fp2,"\nDIV R0,%s",arg2);
     fprintf(fp2,"\nMOV %s,R0",result);
   }
   else if(!strcmp(op,"="))
   {
     fprintf(fp2,"MOV R0,%s",arg1);
     fprintf(fp2,"\nMOV %s,R0",result);
   }
       fprintf(fp2,"\n");
   }
   fclose(fp1);
   fclose(fp2);

   getchar();
}
```

Input



```
* a a x
* b b y
* 2 a d
* b d e
+ x e d
+ y d c
```

Output



```
MOV R0,a
MUL R0,a
MOV x,R0
MOV R0,b
MUL R0,b
MOV y,R0
MOV R0,2
MUL R0,a
MOV d,R0
MOV R0,b
MUL R0,d
MOV e,R0
MOV R0,x
ADD R0,e
MOV d,R0
MOV R0,y
ADD R0,d
MOV c,R0
```

Conclusion

Converting to assembly code is the final phase of the compiler. We implemented a basic assembly code converter program for a hypothetical machine for demonstration purposes. The input is a quadruple code.