**EX NO:9  
  
DATE:**

**IMPLEMENTATION OF BACK END OF THE COMPILER**

**AIM:**

To implement the back end of the compiler which takes the three address codes as input and produces assembly language instructions that can be assembled and run using an 8086 assembler. The target assembly instructions can be simple move, add, sub, and jump.

**ALGORITHM:**

**STEP 1**: Start

**STEP 2**: Get the input expression from the user.

**STEP 3**: Declare the character array to store the assembly instruction strings

mov[]="MOVF",mul[]="MULF",add[]="ADDF",sub[]="SUBF" STEP 4: Scan the input character by character.

**STEP 5**: Initialize the variable r to track the number of registers

.**STEP 6**: If there is a = operator enter the mov "%c%c%c,R%d",a[j],a[j+1],a[j+2],r++ to the file

**STEP 7**: If the identifier is found "\n%s

#%c%c,R%d",mov,a[i],a[i+1],r++ to the file

**STEP 8**: Then search for the identifier if + use add , \* use mul , - use sub

**STEP 9**: Enter the operator assembly instruction with the "R%c,R%d",e,r-1 where r-1 gives previous register and e gives the e=a[i-1];   
**STEP 10**: MOV the value from register to id1.   
**STEP 11**: End

**PROGRAM:**

#include<stdio.h>

#include<ctype.h>

int ag=0,z=1;

int main()

{

char

a[50],id[50],mov[]="MOVF",mul[]="MULF",add[]="ADDF",su b[]="SUBF";

int i=0,j=0,len=0,r=1,s=0,e=0;

FILE \*fp;

fp=fopen("out1.txt","w");

printf("\nEnter the Input:");

gets(a);

len=strlen(a);

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

{

if(a[i]=='=')

{

for(j=1;j<len;j++)

if(a[j]=='i')

{

fprintf(fp,"\n%s ",mov);

fprintf(fp,"%c%c%c,R%d",a[j],a[j+1],a[j+2],r++); }

}

else if((a[i]<=57)&&(a[i]>=48))

if((a[i+1]<=57)&&(a[i+1]>=48))

fprintf(fp,"\n%s #%c%c,R%d",mov,a[i],a[i+1],r++); }

for(i=len-1;i>=0;i--)

{

if(a[i]=='+')

{

fprintf(fp,"\n%s ",add);

e=a[i-1];

e--;

s=e;

if(a[i+1]=='i')

fprintf(fp,"R%c,R%d",e,r-1);

}

else if(a[i]=='-')

{

fprintf(fp,"\n%s",sub);

e=a[i-1];

e--;

s=0;

if(a[i+1]=='i')

fprintf(fp,"R%c,R%c",(a[i+3]-1),s); else

fprintf(fp,"R%c,R%d",e,r-1); }

else if(a[i]=='\*')

{

fprintf(fp,"\n%s ",mul);

e=a[i-1];

e--;

s=e;

if(a[i+1]=='i')

fprintf(fp,"R%c,R%c",(a[i+3]-1),s); else

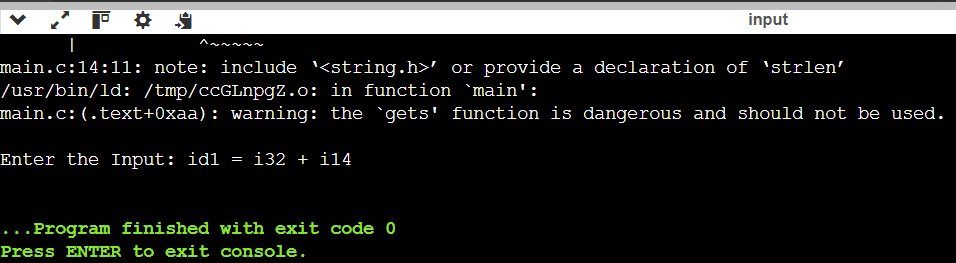
fprintf(fp,"R%c,R%d",e,r-1); }

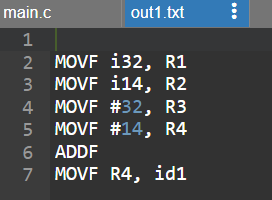
}

fprintf(fp,"\n%s R%d,id1",mov,r-1); return 0;

}

**OUTPUT**:





**RESULT:**

Thus the implementation of the back end of the compiler which takes the three address codes as input and produces assembly language instructions that can be assembled and run using an 8086 assembler. The target assembly instructions can be simple move, add, sub, and jump is completed and verified successfully.