**19. Problem Statement: A program to generate machine code from the abstract syntax tree generated by the parser.**

**AIM:** To write a C Program to Generate Machine Code from the Abstract Syntax Tree

using the specified machine instruction formats.

**THEORY:** Abstract Syntax Tree is a kind of tree representation of the abstract syntactic structure of source code written in a programming language. Each node of the tree denotes a construct occurring in the source code.

Machine code is a computer programming language comprising hexadecimal or binary instructions that computers are able to respond to directly. Machine code is written in a machine language. Therefore, a machine, i.e., a computer, can execute it without any translation or conversion

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int label[20];

int no=0;

int main()

{

FILE \*fp1,\*fp2;

char fname[10],op[10],ch;

char operand1[8],operand2[8],result[8];

int i=0,j=0;

printf("\n Enter filename of the intermediate code");

scanf("%s",&fname);

fp1=fopen(fname,"r");

fp2=fopen("target.txt","w");

if(fp1==NULL || fp2==NULL)

{

printf("\n Error opening the file");

exit(0);

}

while(!feof(fp1))

{

fprintf(fp2,"\n");

fscanf(fp1,"%s",op);

i++;

if(check\_label(i))

fprintf(fp2,"\nlabel#%d",i);

if(strcmp(op,"print")==0)

{

fscanf(fp1,"%s",result);

fprintf(fp2,"\n\t OUT %s",result);

}

if(strcmp(op,"goto")==0)

{

fscanf(fp1,"%s %s",operand1,operand2);

fprintf(fp2,"\n\t JMP %s,label#%s",operand1,operand2);

label[no++]=atoi(operand2);

}

if(strcmp(op,"[]=")==0)

{

fscanf(fp1,"%s %s %s",operand1,operand2,result);

fprintf(fp2,"\n\t STORE %s[%s],%s",operand1,operand2,result);

}

if(strcmp(op,"uminus")==0)

{

fscanf(fp1,"%s %s",operand1,result);

fprintf(fp2,"\n\t LOAD -%s,R1",operand1);

fprintf(fp2,"\n\t STORE R1,%s",result);

}

switch(op[0])

{

case '\*': fscanf(fp1,"%s %s %s",operand1,operand2,result);

fprintf(fp2,"\n \t LOAD",operand1);

fprintf(fp2,"\n \t LOAD %s,R1",operand2);

fprintf(fp2,"\n \t MUL R1,R0");

fprintf(fp2,"\n \t STORE R0,%s",result);

break;

case '+': fscanf(fp1,"%s %s %s",operand1,operand2,result);

fprintf(fp2,"\n \t LOAD %s,R0",operand1);

fprintf(fp2,"\n \t LOAD %s,R1",operand2);

fprintf(fp2,"\n \t ADD R1,R0");

fprintf(fp2,"\n \t STORE R0,%s",result);

break;

case '-': fscanf(fp1,"%s %s %s",operand1,operand2,result);

fprintf(fp2,"\n \t LOAD %s,R0",operand1);

fprintf(fp2,"\n \t LOAD %s,R1",operand2);

fprintf(fp2,"\n \t SUB R1,R0");

fprintf(fp2,"\n \t STORE R0,%s",result);

break;

case '/': fscanf(fp1,"%s %s %s",operand1,operand2,result);

fprintf(fp2,"\n \t LOAD %s,R0",operand1);

fprintf(fp2,"\n \t LOAD %s,R1",operand2);

fprintf(fp2,"\n \t DIV R1,R0");

fprintf(fp2,"\n \t STORE R0,%s",result);

break;

case '%': fscanf(fp1,"%s %s %s",operand1,operand2,result);

fprintf(fp2,"\n \t LOAD %s,R0",operand1);

fprintf(fp2,"\n \t LOAD %s,R1",operand2);

fprintf(fp2,"\n \t DIV R1,R0");

fprintf(fp2,"\n \t STORE R0,%s",result);

break;

case '=': fscanf(fp1,"%s %s",operand1,result);

fprintf(fp2,"\n\t STORE %s %s",operand1,result);

break;

case '>': j++;

fscanf(fp1,"%s %s %s",operand1,operand2,result);

fprintf(fp2,"\n \t LOAD %s,R0",operand1);

fprintf(fp2,"\n\t JGT %s,label#%s",operand2,result);

label[no++]=atoi(result);

break;

case '<': fscanf(fp1,"%s %s %s",operand1,operand2,result);

fprintf(fp2,"\n \t LOAD %s,R0",operand1);

fprintf(fp2,"\n\t JLT %s, label#%d",operand2,result);

label[no++]=atoi(result);

break;

}

}

fclose(fp2); fclose(fp1);

fp2=fopen("target.txt","r");

if(fp2==NULL)

{

printf("Error opening the file\n");

exit(0);

}

do

{

ch=fgetc(fp2);

printf("%c",ch);

}while(ch!=EOF);

fclose(fp1);

return 0;

}

int check\_label(int k)

{

int i;

for(i=0;i<no;i++)

{

if(k==label[i])

return 1;

}

return 0;

}

Input :

$ vi int.txt

= t1 2

[]= a 0 1

[]= a 1 2

[]= a 2 3

\*t1 6 t2

+ a[2] t2 t3

- a[2] t1 t2

/ t3 t2 t2

uminus t2 t2

print t2

goto t2 t3

= t3 99

uminus 25 t2

\* t2 t3 t3

uminus t1 t1 + t1 t3 t4

print t4

**Output :**

Enter filename of the intermediate code: int.txt

STORE t1, 2

STORE a[0], 1

STORE a[1], 2

STORE a[2], 3

LOAD t1, R0

LOAD 6, R1

ADD R1, R0

STORE R0, t3

LOAD a[2], R0

LOAD t2, R1

ADD R1,R0

STORE R0,t3

LOAD a[t2],R0

LOAD t1,R1

SUB R1,R0

STORE R0,t2

LOAD t3,R0

LOAD t2,R1

DIV R1,R0

STORE R0,t2

LOAD t2,R1

STORE R1,t2

LOAD t2,R0

JGT 5, label#11

Label#11: OUT t2

JMP t2, label#13

Label#13: STORE t3, 99

LOAD 25, R1

STORE R1,t2

LOAD t2,R0

LOAD t3,R1

MUL R1,R0

STORE R0,t3

LOAD t1,R1

STORE R1,t1

LOAD t1,R0

LOAD t3,R1

ADD R1,R0

STORE R0,t4

OUT t4