**Ex. No:**

**CODE GENERATION**

**Aim:**

To implement the back end of the compiler, this takes the three address code and produces assembly language instructions.

**Description:**

***GETREG function:***

The GETREG function returns the location L to hold the value of A for the assignment

A = B op C

1. If the name B is in a register that holds the value of no other names and B has no next use and not live after the computation A = B op C, then return the registers of B for L. Update the address descriptor of B as “B is no longer in L”
2. Failing step (1), return an empty register for L if there is one.
3. Failing step (2), if A has a next-use in the block, or op is an operator such as indexing that requires register to perform the operation, then find an occupied register R. Store the value of R into a memory location if it is not already in the proper memory location M. Update the address descriptor for M and return R.
4. If A is not used in the block, select the memory location of A as L.

**Algorithm:**

For each three address code A = B op C in the given basic block, do the following

1. Invoke the function GETREG () to determine the location L where the computation ***B op C*** should be performed. L may be either register or memory location
2. Consult the address descriptor for B to determine B. Prefer register for B’, if the value of B is currently in memory and in register. If the value of B is not in L, generate the instruction MOV B,L to place a copy of B in L
3. Generate the instruction OP C , L where C is the current location of C. Update the address descriptor of A to “ A in L”. If L is a register, update its descriptor to “L contains A”
4. If the current values of B and/or C have no next uses or not live on exit from the block and are in registers, then update the register descriptors that, those registers no longer will contain B and/or C.

**Source Code**

#include<stdio.h>

#include<string.h>

#include<ctype.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);

}

void 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

a=b-c

c=a\*b

exit

The Equivalent Assembly Code is:

Mov R0,b

Add c,R0

Mov a,R0

Mov R1,b

Sub c,R1

Mov a,R1

Mov R2,a

Mul b,R2

Mov c,R2