**Assignment No. 6**

**Name :** Sayali Shinde

**Roll No. :** 61

**Problem Statement :** Generate appropriate target code for given intermediate code assuming suitable processor details.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code :**

package codegeneration;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class CodeGeneration {

public static String nextUseInfo[][] = new String[][] {{"a","y","2"},{"b","y","1"},{"c","y","2"},{"d","y","4"},{"t","y","3"},{"u","y","5"},{"v","y","5"}};

public static String AddrDes[][] = new String[][] {{"a","-1"},{"b","-1"},{"c","-1"},{"d","-1"},{"t","-1"},{"u","-1"},{"v","-1"}};

public static String RegDes[] = new String[] {"-1","-1","-1"};

public static int nextUseInfoCount=7,AddrDesCount=7,RegDesCount=3;

public static int flag=0;//flag 1=already present, flag 2= empty assigned, flag 3=not next use var, flag =4 spill

public static void main(String[] args) throws FileNotFoundException {

int count=1,reg1=0,reg2=0,regRslt=0;

String instruction = null;

String arg1=null,arg2=null,result=null;

File fp = new File("D:\\input.txt");

Scanner in = new Scanner(fp);

System.out.println("Input code:");

while(in.hasNextLine())

{

System.out.println(in.nextLine());

}

//code generation

in = new Scanner(fp);

while(in.hasNextLine())

{

int reg;

instruction = in.nextLine();

if(instruction.length()==3 && instruction.contains("="))

{

arg1 = instruction.substring(2,3);

result = instruction.substring(0,1);

reg1 =getReg(arg1,count,instruction);

change(reg1,arg1);

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(result))

{

AddrDes[i][1]=Integer.toString(reg1);

}

}

store(result, "R"+reg1);

}

else

{

if(instruction.contains("\*"))

{

arg1 = instruction.substring(2,3);

arg2 = instruction.substring(4,5);

result = instruction.substring(0,1);

reg1 =getReg(arg1,count,instruction);

change(reg1,arg1);

reg2 =getReg(arg2,count,instruction);

change(reg2,arg2);

regRslt = getReg(result, count, instruction);

mul("R"+reg1,"R"+reg2,"R"+regRslt);

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(result))

{

AddrDes[i][1]=Integer.toString(regRslt);

}

}

print();

}

else if(instruction.contains("+"))

{

arg1 = instruction.substring(2,3);

arg2 = instruction.substring(4,5);

result = instruction.substring(0,1);

reg1 =getReg(arg1,count,instruction);

change(reg1,arg1);

reg2 =getReg(arg2,count,instruction);

change(reg2,arg2);

regRslt = getReg(result, count, instruction);

add("R"+reg1,"R"+reg2,"R"+regRslt);

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(result))

{

AddrDes[i][1]=Integer.toString(regRslt);

}

}

print();

}

else if(instruction.contains("/"))

{

arg1 = instruction.substring(2,3);

arg2 = instruction.substring(4,5);

result = instruction.substring(0,1);

reg1 =getReg(arg1,count,instruction);

change(reg1,arg1);

reg2 =getReg(arg2,count,instruction);

change(reg2,arg2);

regRslt = getReg(result, count, instruction);

div("R"+reg1,"R"+reg2,"R"+regRslt);

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(result))

{

AddrDes[i][1]=Integer.toString(regRslt);

}

}

print();

}

else if(instruction.contains("-"))

{

arg1 = instruction.substring(2,3);

arg2 = instruction.substring(4,5);

result = instruction.substring(0,1);

reg1 =getReg(arg1,count,instruction);

change(reg1,arg1);

reg2 =getReg(arg2,count,instruction);

change(reg2,arg2);

regRslt = getReg(result, count, instruction);

sub("R"+reg1,"R"+reg2,"R"+regRslt);

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(result))

{

AddrDes[i][1]=Integer.toString(regRslt);

}

}

print();

}

}

count++;

}

}

// register allocation

public static int getReg(String oprand, int currCount, String inst)

{

int i=-1;

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

{

if((RegDes[i].toString()).equals(oprand.toString()))

{

flag =1;

return i;

}

}

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

{

if((RegDes[i].toString()).equals("-1"))

{

RegDes[i]=oprand;

for(int j=0;j<AddrDesCount;j++)

{

if(AddrDes[j][0].equals(oprand))

{AddrDes[j][1]=Integer.toString(i);}

}

flag =2;

return i;

}

}

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

{

for(int j=0;j<nextUseInfoCount;j++)

{

if(RegDes[i].equals(nextUseInfo[j][0]) && nextUseInfo[j][1].equals("y") && Integer.parseInt(nextUseInfo[j][2])<currCount)

{

RegDes[i]=oprand;

for(int p=0;p<AddrDesCount;p++)

{

if(AddrDes[p][0].equals(RegDes[i]))

{

AddrDes[p][1]=Integer.toString(i);

}

if(AddrDes[p][0].equals(nextUseInfo[j][0]))

{

AddrDes[p][1]="-1";

}

}

flag =3;

return i;

}

}

}

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

{

if(!inst.contains(RegDes[i]))

{ for(int j=0;j<AddrDesCount;j++)

{

if(AddrDes[j][0].equals(RegDes[i]))

{

AddrDes[j][1]="-1";

return i;

}

}

flag =4;

}

}

return i;

}

public static void change(int num,String opr)

{

if(flag == 2)

{

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(opr))

AddrDes[i][1]=Integer.toString(num);

}

load("R"+num,opr);

}

else if(flag ==3)

{

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(RegDes[num]))

{

AddrDes[i][1]="-1";

RegDes[num]=opr;

}

}

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(RegDes[num]))

{

AddrDes[i][1]=Integer.toString(num);

}

}

load("R"+num,opr);

}

else if(flag ==4)

{

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(RegDes[num]))

{

store(AddrDes[i][0], "R"+num);

AddrDes[i][1]="-1";

RegDes[num]=opr;

}

}

for(int i=0;i<AddrDesCount;i++)

{

if(AddrDes[i][0].equals(RegDes[num]))

{

AddrDes[i][1]=Integer.toString(num);

}

}

load("R"+num,opr);

}

}

public static void load(String one, String two)

{

System.out.println("\nLD "+one+" , "+two);

}

public static void div(String one, String two, String three)

{

System.out.println("\nDIV "+one+" , "+two+" , "+three);

}

public static void sub(String one, String two, String three)

{

System.out.println("\nSUB "+one+" , "+two+" , "+three);

}

public static void add(String one, String two, String three)

{

System.out.println("\nADD "+one+" , "+two+" , "+three);

}

public static void mul(String one, String two, String three)

{

System.out.println("\nMUL "+one+" , "+two+" , "+three);

}

public static void store(String one, String two)

{

System.out.println("\nST "+one+" , "+two);

}

public static void print()

{

System.out.println("\nRegDes Content :");

for(int i=0;i<RegDesCount;i++)

{

System.out.print(RegDes[i]+" ");

}

System.out.println("\n AddrDes Content :");

for(int i=0;i<AddrDesCount;i++)

{

System.out.print(AddrDes[i][0]+" "+AddrDes[i][1]+" ; ");

}

}

}

**Input code:**

t=a\*b

u=a-c

v=t+u

a=d

d=v/u

**Output :**

LD R0 , a

LD R1 , b

MUL R0 , R1 , R2

RegDes Content :

a b t

AddrDes Content :

a 0 ; b 1 ; c -1 ; d -1 ; t 2 ; u -1 ; v -1 ;

LD R1 , c

SUB R0 , R1 , R2

RegDes Content :

a c t

AddrDes Content :

a 0 ; b -1 ; c 1 ; d -1 ; t -1 ; u 2 ; v -1 ;

LD R0 , u

ADD R2 , R0 , R1

RegDes Content :

u v t

AddrDes Content :

a -1 ; b -1 ; c -1 ; d -1 ; t -1 ; u 0 ; v 1 ;

LD R2 , d

ST a , R2

DIV R1 , R0 , R2

RegDes Content :

u v d

AddrDes Content :

a 2 ; b -1 ; c -1 ; d 2 ; t -1 ; u 0 ; v 1 ;