

RESEARCH PROJECT SUBMISSION

Architecture Final Assessment

**Program: CESS**

***Course Code: CSE 112***

***Course Name: Computer Organization and Architecture***

***Examination Committee***

**Dr. Tamer Mostafa**

**Ain Shams University**

**Faculty of Engineering**

**Spring 2020 Semester**

**Student Personal Information**

Students’ Names & IDs: Sherif Ahmed Naiem Mohamed 18P6546

Yomna Hussein Mohamed Abd El Hamid 18P5794

Omar Mohamed Lotfy EL-Sayed 18P5606

Nour El-Din Mohamed Hesham 18P5806

Reda Mohsen Reda 18P5141

**Plagiarism Statement**

I certify that this assignment / report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they are books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment / report has not been previously been submitted for assessment for another course. I certify that I have not copied in part or whole or otherwise plagiarized the work of other students and / or persons.

**Sherif Ahmed/ Yomna Hussein/ Omar Lotfy/ Nour El-Din Mohamed/ Reda Mohsen**

**Signature/Student Name:**

**3-6-2020**

**Date:**

**Submission Contents**

**01: Coding**

**02: Testing**

**01**

***First Topic***

**Table of Contents**

[I. Coding 4](#_Toc41805585)

[1. Purpose of the code 4](#_Toc41805586)

[II. Testing the code 55](#_Toc41805587)

[1. Testing the first case: 56](#_Toc41805588)

[2. Testing the second case 58](#_Toc41805589)

# Coding

## Purpose of the code

This code is made to translate MIPS instructions and execute it.

**The code:**

|  |
| --- |
| **Class ArchiProject (Main args)** |
| package archiproject;  import java.util.Scanner;  public class ArchiProject {    public static String binaryConverter(int convertthis,int bits){ //This method takes a number and number of bits we want and then convert this number to binary and fill the rest with zeros  //for example (5,16) = 00000000000000101  return String.format("%"+bits+"s", Integer.toBinaryString(convertthis)).replaceAll(" ", "0");  }  public static void main(String[] args) {    String instruction [][]= new String[32][2]; //[x][0] for the instruction lines //[x][1] for the address  int Number\_of\_Instruction\_Lines = 0;  int Base\_Instruction\_Address = 0;  Scanner sc = new Scanner(System.in);  System.out.println("Please Enter Your Assembly Code and type Exit: to execute code");  String AssemblySentence;  do{  AssemblySentence = sc.nextLine();  instruction[Number\_of\_Instruction\_Lines][0]=AssemblySentence;  instruction[Number\_of\_Instruction\_Lines][1]=Integer.toBinaryString(Base\_Instruction\_Address);  Number\_of\_Instruction\_Lines++;  Base\_Instruction\_Address=Base\_Instruction\_Address+4;  }while(!(AssemblySentence.equalsIgnoreCase("Exit:")));    String First\_word\_of\_each\_instruction[]=new String[Number\_of\_Instruction\_Lines+1]; //making an array of string that takes first word of each sentence  for(int i=0; i<First\_word\_of\_each\_instruction.length-1;i++){  String arrr[] =instruction[i][0].split(" ", 2);  First\_word\_of\_each\_instruction[i]=arrr[0];  }      String instructions\_to\_binary[] = new String[32]; // A 1-D array that takes each line of translated instruction        for (int i=0;i<Number\_of\_Instruction\_Lines; i++){            String op="00000";  String f="00000";  String rd="00000";  String rs="00000";  String rt="00000";  String shift="00000";  String address ="0000000000000000";    String a= instruction[i][0];  String array[]= a.split(" ");    if(array.length==4){      String w1= array[0];  String w2= array[1];  String w3= array[2];  String w4= array[3];      if(w1.equals("add") || w1.equals("sub")|| w1.equals("and") || w1.equals("or") || w1.equals("nor") || w1.equals("sll") || w1.equals("srl") || w1.equals("sra") || w1.equals("slt")){  switch(w1){  case "add":  op="000000";  f="100000";  break;  case "sub":  op="000000";  f="100000";  break;  case "and":  op="000000";  f="100100";  break;  case "or":  op="000000";  f="100101";  break;  case "nor":  op="000000";  f="100111";  break;  case "sll":  op="000000";  f="000000";  break;  case "srl":  op="000000";  f="000010";  break;  case "sra":  op="000000";  f="000011";  break;  case "slt":  op="000000";  f="101010";  break;  }  switch(w2){  case "$zero,":  rd="00000";  break;  case "$at,":  rd="00001";  break;  case "$t0,":  rd="01000";  break;  case "$t1,":  rd="01001";  break;  case "$t2,":  rd="01010";  break;  case "$t3,":  rd="01011";  break;  case "$t4,":  rd="01100";  break;  case "$t5,":  rd="01101";  break;  case "$t6,":  rd="01110";  break;  case "$t7,":  rd="01111";  break;  case "$t8,":  rd="11000";  break;  case "$t9,":  rd="11001";  break;  case "$s0,":  rd="10000";  break;  case "$s1,":  rd="10001";  break;  case "$s2,":  rd="10010";  break;  case "$s3,":  rd="10011";  break;  case "$s4,":  rd="10100";  break;  case "$s5,":  rd="10101";  break;  case "$s6,":  rd="10110";  break;  case "$s7,":  rd="10111";  break;  }  if(w1.equals("sll") || w1.equals("srl") || w1.equals("sra")){  switch(w3){  case "$zero,":  rt="00000";  break;  case "$at,":  rt="00001";  break;  case "$t0,":  rt="01000";  break;  case "$t1,":  rt="01001";  break;  case "$t2,":  rt="01010";  break;  case "$t3,":  rt="01011";  break;  case "$t4,":  rt="01100";  break;  case "$t5,":  rt="01101";  break;  case "$t6,":  rt="01110";  break;  case "$t7,":  rt="01111";  break;  case "$t8,":  rt="11000";  break;  case "$t9,":  rt="11001";  break;  case "$s0,":  rt="10000";  break;  case "$s1,":  rt="10001";  break;  case "$s2,":  rt="10010";  break;  case "$s3,":  rt="10011";  break;  case "$s4,":  rt="10100";  break;  case "$s5,":  rt="10101";  break;  case "$s6,":  rt="10110";  break;  case "$s7,":  rt="10111";  break;  }  switch(w4){  case "0":  shift ="00000";  break;  case "1":  shift ="00001";  break;  case "2":  shift ="00010";  break;  case "3":  shift ="00011";  break;  case "4":  shift ="00100";  break;  case "5":  shift ="00101";  break;  case "6":  shift ="00110";  break;  case "7":  shift ="00111";  break;  case "8":  shift ="01000";  break;  case "9":  shift ="01001";  break;  case "10":  shift ="01010";  break;  case "11":  shift ="01011";  break;  case "12":  shift ="01100";  break;  case "13":  shift ="01101";  break;  case "14":  shift ="01110";  break;  case "15":  shift ="01111";  break;  case "16":  shift ="10000";  break;  case "17":  shift ="10001";  break;  case "18":  shift ="10010";  break;  case "19":  shift ="10011";  break;  case "20":  shift ="10100";  break;  case "21":  shift ="10101";  break;  case "22":  shift ="10110";  break;  case "23":  shift ="10111";  break;  case "24":  shift ="11000";  break;  case "25":  shift ="11001";  break;  case "26":  shift ="11010";  break;  case "27":  shift ="11011";  break;  case "28":  shift ="11100";  break;  case "29":  shift ="11101";  break;  case "30":  shift ="11110";  break;  case "31":  shift ="11111";  break;  }  } // closing small if  else{  switch(w3){  case "$zero,":  rs="00000";  break;  case "$at,":  rs="00001";  break;  case "$t0,":  rs="01000";  break;  case "$t1,":  rs="01001";  break;  case "$t2,":  rs="01010";  break;  case "$t3,":  rs="01011";  break;  case "$t4,":  rs="01100";  break;  case "$t5,":  rs="01101";  break;  case "$t6,":  rs="01110";  break;  case "$t7,":  rs="01111";  break;  case "$t8,":  rs="11000";  break;  case "$t9,":  rs="11001";  break;  case "$s0,":  rs="10000";  break;  case "$s1,":  rs="10001";  break;  case "$s2,":  rs="10010";  break;  case "$s3,":  rs="10011";  break;  case "$s4,":  rs="10100";  break;  case "$s5,":  rs="10101";  break;  case "$s6,":  rs="10110";  break;  case "$s7,":  rs="10111";  break;  }  switch(w4){  case "$zero":  rt="00000";  break;  case "$at":  rt="00001";  break;  case "$t0":  rt="01000";  break;  case "$t1":  rt="01001";  break;  case "$t2":  rt="01010";  break;  case "$t3":  rt="01011";  break;  case "$t4":  rt="01100";  break;  case "$t5":  rt="01101";  break;  case "$t6":  rt="01110";  break;  case "$t7":  rt="01111";  break;  case "$t8":  rt="11000";  break;  case "$t9":  rt="11001";  break;  case "$s0":  rt="10000";  break;  case "$s1":  rt="10001";  break;  case "$s2":  rt="10010";  break;  case "$s3":  rt="10011";  break;  case "$s4":  rt="10100";  break;  case "$s5":  rt="10101";  break;  case "$s6":  rt="10110";  break;  case "$s7":  rt="10111";  break;  }  } // closing small else  instructions\_to\_binary[i]=op+rs+rt+rd+shift+f;  } // closing large if  else if(w1.equals("addi") || w1.equals("andi") || w1.equals("ori") || w1.equals("slti") || w1.equals("lw") || w1.equals("sw") || w1.equals("beq") || w1.equals("bne")){  switch (w1){  case "addi":  op="001000";  break;  case "andi":  op="001100";  break;  case "ori":  op="001101";  break;  case "beq":  op="000100";  break;  case "bne":  op="000101";  break;  case "lw":  op="100011";  break;  case "sw":  op="101011";  break;  case "slti":  op="001010";  break;  }  switch(w2){  case "$zero,":  rt="00000";  break;  case "$at,":  rt="00001";  break;  case "$t0,":  rt="01000";  break;  case "$t1,":  rt="01001";  break;  case "$t2,":  rt="01010";  break;  case "$t3,":  rt="01011";  break;  case "$t4,":  rt="01100";  break;  case "$t5,":  rt="01101";  break;  case "$t6,":  rt="01110";  break;  case "$t7,":  rt="01111";  break;  case "$t8,":  rt="11000";  break;  case "$t9,":  rt="11001";  break;  case "$s0,":  rt="10000";  break;  case "$s1,":  rt="10001";  break;  case "$s2,":  rt="10010";  break;  case "$s3,":  rt="10011";  break;  case "$s4,":  rt="10100";  break;  case "$s5,":  rt="10101";  break;  case "$s6,":  rt="10110";  break;  case "$s7,":  rt="10111";  break;  }  if(w1.equals("lw") || w1.equals("sw")){  switch (w3){  case "0":  address ="0000000000000000";  break;  case "1":  address ="0000000000000001";  break;  case "2":  address ="0000000000000010";  break;  case "3":  address ="0000000000000011";  break;  case "4":  address ="0000000000000100";  break;  case "5":  address ="0000000000000101";  break;  case "6":  address ="0000000000000110";  break;  case "7":  address ="0000000000000111";  break;  case "8":  address ="0000000000001000";  break;  case "9":  address ="0000000000001001";  break;  case "10":  address ="0000000000001010";  break;  case "11":  address ="0000000000001011";  break;  case "12":  address ="0000000000001100";  break;  case "13":  address ="0000000000001101";  break;  case "14":  address ="0000000000001110";  break;  case "15":  address ="0000000000001111";  break;  case "16":  address ="0000000000010000";  break;  case "17":  address ="0000000000010001";  break;  case "18":  address ="0000000000010010";  break;  case "19":  address ="0000000000010011";  break;  case "20":  address ="0000000000010100";  break;  case "21":  address ="0000000000010101";  break;  case "22":  address ="0000000000010110";  break;  case "23":  address ="0000000000010111";  break;  case "24":  address ="0000000000011000";  break;  case "25":  address ="0000000000011001";  break;  case "26":  address ="0000000000011010";  break;  case "27":  address ="0000000000011011";  break;  case "28":  address ="0000000000011100";  break;  case "29":  address ="0000000000011101";  break;  case "30":  address ="0000000000011110";  break;  case "31":  address ="000000000011111";  break;  case "32":  address ="0000000000100000";  }  switch (w4){  case "($zero)":  rs="00000";  break;  case "($at)":  rs="00001";  break;  case "($t0)":  rs="01000";  break;  case "($t1)":  rs="01001";  break;  case "($t2)":  rs="01010";  break;  case "($t3)":  rs="01011";  break;  case "($t4)":  rs="01100";  break;  case "($t5)":  rs="01101";  break;  case "($t6)":  rs="01110";  break;  case "($t7)":  rs="01111";  break;  case "($t8)":  rs="11000";  break;  case "($t9)":  rs="11001";  break;  case "($s0)":  rs="10000";  break;  case "($s1)":  rs="10001";  break;  case "($s2)":  rs="10010";  break;  case "($s3)":  rs="10011";  break;  case "($s4)":  rs="10100";  break;  case "($s5)":  rs="10101";  break;  case "($s6)":  rs="10110";  break;  case "($s7)":  rs="10111";  break;  }  }else{  switch (w3){  case "$zero,":  rs="00000";  break;  case "$at,":  rs="00001";  break;  case "$t0,":  rs="01000";  break;  case "$t1,":  rs="01001";  break;  case "$t2,":  rs="01010";  break;  case "$t3,":  rs="01011";  break;  case "$t4,":  rs="01100";  break;  case "$t5,":  rs="01101";  break;  case "$t6,":  rs="01110";  break;  case "$t7,":  rs="01111";  break;  case "$t8,":  rs="11000";  break;  case "$t9,":  rs="11001";  break;  case "$s0,":  rs="10000";  break;  case "$s1,":  rs="10001";  break;  case "$s2,":  rs="10010";  break;  case "$s3,":  rs="10011";  break;  case "$s4,":  rs="10100";  break;  case "$s5,":  rs="10101";  break;  case "$s6,":  rs="10110";  break;  case "$s7,":  rs="10111";  break;  }  switch(w4){  case "0":  address ="0000000000000000";  break;  case "1":  address ="0000000000000001";  break;  case "2":  address ="0000000000000010";  break;  case "3":  address ="0000000000000011";  break;  case "4":  address ="0000000000000100";  break;  case "5":  address ="0000000000000101";  break;  case "6":  address ="0000000000000110";  break;  case "7":  address ="0000000000000111";  break;  case "8":  address ="0000000000001000";  break;  case "9":  address ="0000000000001001";  break;  case "10":  address ="0000000000001010";  break;  case "11":  address ="0000000000001011";  break;  case "12":  address ="0000000000001100";  break;  case "13":  address ="0000000000001101";  break;  case "14":  address ="0000000000001110";  break;  case "15":  address ="0000000000001111";  break;  case "16":  address ="0000000000010000";  break;  case "17":  address ="0000000000010001";  break;  case "18":  address ="0000000000010010";  break;  case "19":  address ="0000000000010011";  break;  case "20":  address ="0000000000010100";  break;  case "21":  address ="0000000000010101";  break;  case "22":  address ="0000000000010110";  break;  case "23":  address ="0000000000010111";  break;  case "24":  address ="0000000000011000";  break;  case "25":  address ="0000000000011001";  break;  case "26":  address ="0000000000011010";  break;  case "27":  address ="0000000000011011";  break;  case "28":  address ="0000000000011100";  break;  case "29":  address ="0000000000011101";  break;  case "30":  address ="0000000000011110";  break;  case "31":  address ="000000000011111";  break;  case "32":  address ="0000000000100000";  break;    }  if(w1.equals("beq")){ // this is special case of beq  // na-(ca+4))/4    for(int sony=0; sony<Number\_of\_Instruction\_Lines; sony++){    String label= First\_word\_of\_each\_instruction[sony];  if((w4+":").equals(label)){  String current\_address = instruction[i][1];  String next\_address = instruction[sony][1];  int ca = Integer.parseInt(current\_address,2);  // System.out.println(ca);  int na = Integer.parseInt(next\_address,2);  // System.out.println(na);  int calc = ((na-(ca+4))/4);  // System.out.println(calc);  address = ArchiProject.binaryConverter(calc, 16);  }      }      }    } //closing of else that makes other i-format insturctions like addi,..etc        instructions\_to\_binary[i]=op+rs+rt+address;  }//closing of if case of beginning of i-format      }//closing if case of array length of 4  if(array.length==3){  String w1= array[0];  String w2= array[1];  String w3= array[2];  op="001111";  switch(w2){  case "$zero,":  rt="00000";  break;  case "$at,":  rt="00001";  break;  case "$t0,":  rt="01000";  break;  case "$t1,":  rt="01001";  break;  case "$t2,":  rt="01010";  break;  case "$t3,":  rt="01011";  break;  case "$t4,":  rt="01100";  break;  case "$t5,":  rt="01101";  break;  case "$t6,":  rt="01110";  break;  case "$t7,":  rt="01111";  break;  case "$t8,":  rt="11000";  break;  case "$t9,":  rt="11001";  break;  case "$s0,":  rt="10000";  break;  case "$s1,":  rt="10001";  break;  case "$s2,":  rt="10010";  break;  case "$s3,":  rt="10011";  break;  case "$s4,":  rt="10100";  break;  case "$s5,":  rt="10101";  break;  case "$s6,":  rt="10110";  break;  case "$s7,":  rt="10111";  break;  }  switch (w3){  case "0":  address ="0000000000000000";  break;  case "1":  address ="0000000000000001";  break;  case "2":  address ="0000000000000010";  break;  case "3":  address ="0000000000000011";  break;  case "4":  address ="0000000000000100";  break;  case "5":  address ="0000000000000101";  break;  case "6":  address ="0000000000000110";  break;  case "7":  address ="0000000000000111";  break;  case "8":  address ="0000000000001000";  break;  case "9":  address ="0000000000001001";  break;  case "10":  address ="0000000000001010";  break;  case "11":  address ="0000000000001011";  break;  case "12":  address ="0000000000001100";  break;  case "13":  address ="0000000000001101";  break;  case "14":  address ="0000000000001110";  break;  case "15":  address ="0000000000001111";  break;  case "16":  address ="0000000000010000";  break;  case "17":  address ="0000000000010001";  break;  case "18":  address ="0000000000010010";  break;  case "19":  address ="0000000000010011";  break;  case "20":  address ="0000000000010100";  break;  case "21":  address ="0000000000010101";  break;  case "22":  address ="0000000000010110";  break;  case "23":  address ="0000000000010111";  break;  case "24":  address ="0000000000011000";  break;  case "25":  address ="0000000000011001";  break;  case "26":  address ="0000000000011010";  break;  case "27":  address ="0000000000011011";  break;  case "28":  address ="0000000000011100";  break;  case "29":  address ="0000000000011101";  break;  case "30":  address ="0000000000011110";  break;  case "31":  address ="000000000011111";  break;  case "32":  address ="0000000000100000";  break;  }  instructions\_to\_binary[i]=op+rs+rt+address;  }//closing if case of array length of 3  if(array.length==2){  String w1= array[0];  String w2= array[1];    op="000010";        for(int sony=0; sony<Number\_of\_Instruction\_Lines; sony++){    String label= First\_word\_of\_each\_instruction[sony];  if((w2+":").equals(label)){  String current\_address = instruction[i][1];  String next\_address = instruction[sony][1];  int ca = Integer.parseInt(current\_address,2);  // System.out.println(ca);  int na = Integer.parseInt(next\_address,2);  // System.out.println(na);  int calc = ((na-(ca+4))/4);  // System.out.println(calc);  address = ArchiProject.binaryConverter(calc, 26);  }      }    instructions\_to\_binary[i]=op+address;  }        if(array.length==5){    String w1= array[1];  String w2= array[2];  String w3= array[3];  String w4= array[4];  if(w1.equals("add") || w1.equals("sub")|| w1.equals("and") || w1.equals("or") || w1.equals("nor") || w1.equals("sll") || w1.equals("srl") || w1.equals("sra") || w1.equals("slt")){  switch(w1){  case "add":  op="000000";  f="100000";  break;  case "sub":  op="000000";  f="100000";  break;  case "and":  op="000000";  f="100100";  break;  case "or":  op="000000";  f="100101";  break;  case "nor":  op="000000";  f="100111";  break;  case "sll":  op="000000";  f="000000";  break;  case "srl":  op="000000";  f="000010";  break;  case "sra":  op="000000";  f="000011";  break;  case "slt":  op="000000";  f="101010";  break;  }  switch(w2){  case "$zero,":  rd="00000";  break;  case "$at,":  rd="00001";  break;  case "$t0,":  rd="01000";  break;  case "$t1,":  rd="01001";  break;  case "$t2,":  rd="01010";  break;  case "$t3,":  rd="01011";  break;  case "$t4,":  rd="01100";  break;  case "$t5,":  rd="01101";  break;  case "$t6,":  rd="01110";  break;  case "$t7,":  rd="01111";  break;  case "$t8,":  rd="11000";  break;  case "$t9,":  rd="11001";  break;  case "$s0,":  rd="10000";  break;  case "$s1,":  rd="10001";  break;  case "$s2,":  rd="10010";  break;  case "$s3,":  rd="10011";  break;  case "$s4,":  rd="10100";  break;  case "$s5,":  rd="10101";  break;  case "$s6,":  rd="10110";  break;  case "$s7,":  rd="10111";  break;  }  if(w1.equals("sll") || w1.equals("srl") || w1.equals("sra")){  switch(w3){  case "$zero,":  rt="00000";  break;  case "$at,":  rt="00001";  break;  case "$t0,":  rt="01000";  break;  case "$t1,":  rt="01001";  break;  case "$t2,":  rt="01010";  break;  case "$t3,":  rt="01011";  break;  case "$t4,":  rt="01100";  break;  case "$t5,":  rt="01101";  break;  case "$t6,":  rt="01110";  break;  case "$t7,":  rt="01111";  break;  case "$t8,":  rt="11000";  break;  case "$t9,":  rt="11001";  break;  case "$s0,":  rt="10000";  break;  case "$s1,":  rt="10001";  break;  case "$s2,":  rt="10010";  break;  case "$s3,":  rt="10011";  break;  case "$s4,":  rt="10100";  break;  case "$s5,":  rt="10101";  break;  case "$s6,":  rt="10110";  break;  case "$s7,":  rt="10111";  break;  }  switch(w4){  case "0":  shift ="00000";  break;  case "1":  shift ="00001";  break;  case "2":  shift ="00010";  break;  case "3":  shift ="00011";  break;  case "4":  shift ="00100";  break;  case "5":  shift ="00101";  break;  case "6":  shift ="00110";  break;  case "7":  shift ="00111";  break;  case "8":  shift ="01000";  break;  case "9":  shift ="01001";  break;  case "10":  shift ="01010";  break;  case "11":  shift ="01011";  break;  case "12":  shift ="01100";  break;  case "13":  shift ="01101";  break;  case "14":  shift ="01110";  break;  case "15":  shift ="01111";  break;  case "16":  shift ="10000";  break;  case "17":  shift ="10001";  break;  case "18":  shift ="10010";  break;  case "19":  shift ="10011";  break;  case "20":  shift ="10100";  break;  case "21":  shift ="10101";  break;  case "22":  shift ="10110";  break;  case "23":  shift ="10111";  break;  case "24":  shift ="11000";  break;  case "25":  shift ="11001";  break;  case "26":  shift ="11010";  break;  case "27":  shift ="11011";  break;  case "28":  shift ="11100";  break;  case "29":  shift ="11101";  break;  case "30":  shift ="11110";  break;  case "31":  shift ="11111";  break;  }  } // closing small if  else{  switch(w3){  case "$zero,":  rs="00000";  break;  case "$at,":  rs="00001";  break;  case "$t0,":  rs="01000";  break;  case "$t1,":  rs="01001";  break;  case "$t2,":  rs="01010";  break;  case "$t3,":  rs="01011";  break;  case "$t4,":  rs="01100";  break;  case "$t5,":  rs="01101";  break;  case "$t6,":  rs="01110";  break;  case "$t7,":  rs="01111";  break;  case "$t8,":  rs="11000";  break;  case "$t9,":  rs="11001";  break;  case "$s0,":  rs="10000";  break;  case "$s1,":  rs="10001";  break;  case "$s2,":  rs="10010";  break;  case "$s3,":  rs="10011";  break;  case "$s4,":  rs="10100";  break;  case "$s5,":  rs="10101";  break;  case "$s6,":  rs="10110";  break;  case "$s7,":  rs="10111";  break;  }  switch(w4){  case "$zero":  rt="00000";  break;  case "$at":  rt="00001";  break;  case "$t0":  rt="01000";  break;  case "$t1":  rt="01001";  break;  case "$t2":  rt="01010";  break;  case "$t3":  rt="01011";  break;  case "$t4":  rt="01100";  break;  case "$t5":  rt="01101";  break;  case "$t6":  rt="01110";  break;  case "$t7":  rt="01111";  break;  case "$t8":  rt="11000";  break;  case "$t9":  rt="11001";  break;  case "$s0":  rt="10000";  break;  case "$s1":  rt="10001";  break;  case "$s2":  rt="10010";  break;  case "$s3":  rt="10011";  break;  case "$s4":  rt="10100";  break;  case "$s5":  rt="10101";  break;  case "$s6":  rt="10110";  break;  case "$s7":  rt="10111";  break;  }  } // closing small else  instructions\_to\_binary[i]=op+rs+rt+rd+shift+f;  } // closing large if  else if(w1.equals("addi") || w1.equals("andi") || w1.equals("ori") || w1.equals("slti") || w1.equals("lw") || w1.equals("sw") || w1.equals("beq") || w1.equals("bne")){  switch (w1){  case "addi":  op="001000";  break;  case "andi":  op="001100";  break;  case "ori":  op="001101";  break;  case "beq":  op="000100";  break;  case "bne":  op="000101";  break;  case "lw":  op="100011";  break;  case "sw":  op="101011";  break;  case "slti":  op="001010";  break;  }  switch(w2){  case "$zero,":  rt="00000";  break;  case "$at,":  rt="00001";  break;  case "$t0,":  rt="01000";  break;  case "$t1,":  rt="01001";  break;  case "$t2,":  rt="01010";  break;  case "$t3,":  rt="01011";  break;  case "$t4,":  rt="01100";  break;  case "$t5,":  rt="01101";  break;  case "$t6,":  rt="01110";  break;  case "$t7,":  rt="01111";  break;  case "$t8,":  rt="11000";  break;  case "$t9,":  rt="11001";  break;  case "$s0,":  rt="10000";  break;  case "$s1,":  rt="10001";  break;  case "$s2,":  rt="10010";  break;  case "$s3,":  rt="10011";  break;  case "$s4,":  rt="10100";  break;  case "$s5,":  rt="10101";  break;  case "$s6,":  rt="10110";  break;  case "$s7,":  rt="10111";  break;  }  if(w1.equals("lw") || w1.equals("sw")){  switch (w3){  case "0":  address ="0000000000000000";  break;  case "1":  address ="0000000000000001";  break;  case "2":  address ="0000000000000010";  break;  case "3":  address ="0000000000000011";  break;  case "4":  address ="0000000000000100";  break;  case "5":  address ="0000000000000101";  break;  case "6":  address ="0000000000000110";  break;  case "7":  address ="0000000000000111";  break;  case "8":  address ="0000000000001000";  break;  case "9":  address ="0000000000001001";  break;  case "10":  address ="0000000000001010";  break;  case "11":  address ="0000000000001011";  break;  case "12":  address ="0000000000001100";  break;  case "13":  address ="0000000000001101";  break;  case "14":  address ="0000000000001110";  break;  case "15":  address ="0000000000001111";  break;  case "16":  address ="0000000000010000";  break;  case "17":  address ="0000000000010001";  break;  case "18":  address ="0000000000010010";  break;  case "19":  address ="0000000000010011";  break;  case "20":  address ="0000000000010100";  break;  case "21":  address ="0000000000010101";  break;  case "22":  address ="0000000000010110";  break;  case "23":  address ="0000000000010111";  break;  case "24":  address ="0000000000011000";  break;  case "25":  address ="0000000000011001";  break;  case "26":  address ="0000000000011010";  break;  case "27":  address ="0000000000011011";  break;  case "28":  address ="0000000000011100";  break;  case "29":  address ="0000000000011101";  break;  case "30":  address ="0000000000011110";  break;  case "31":  address ="000000000011111";  break;  case "32":  address ="0000000000100000";  }  switch (w4){  case "($zero)":  rs="00000";  break;  case "($at)":  rs="00001";  break;  case "($t0)":  rs="01000";  break;  case "($t1)":  rs="01001";  break;  case "($t2)":  rs="01010";  break;  case "($t3)":  rs="01011";  break;  case "($t4)":  rs="01100";  break;  case "($t5)":  rs="01101";  break;  case "($t6)":  rs="01110";  break;  case "($t7)":  rs="01111";  break;  case "($t8)":  rs="11000";  break;  case "($t9)":  rs="11001";  break;  case "($s0)":  rs="10000";  break;  case "($s1)":  rs="10001";  break;  case "($s2)":  rs="10010";  break;  case "($s3)":  rs="10011";  break;  case "($s4)":  rs="10100";  break;  case "($s5)":  rs="10101";  break;  case "($s6)":  rs="10110";  break;  case "($s7)":  rs="10111";  break;  }  }else{  switch (w3){  case "$zero,":  rs="00000";  break;  case "$at,":  rs="00001";  break;  case "$t0,":  rs="01000";  break;  case "$t1,":  rs="01001";  break;  case "$t2,":  rs="01010";  break;  case "$t3,":  rs="01011";  break;  case "$t4,":  rs="01100";  break;  case "$t5,":  rs="01101";  break;  case "$t6,":  rs="01110";  break;  case "$t7,":  rs="01111";  break;  case "$t8,":  rs="11000";  break;  case "$t9,":  rs="11001";  break;  case "$s0,":  rs="10000";  break;  case "$s1,":  rs="10001";  break;  case "$s2,":  rs="10010";  break;  case "$s3,":  rs="10011";  break;  case "$s4,":  rs="10100";  break;  case "$s5,":  rs="10101";  break;  case "$s6,":  rs="10110";  break;  case "$s7,":  rs="10111";  break;  }  switch(w4){  case "0":  address ="0000000000000000";  break;  case "1":  address ="0000000000000001";  break;  case "2":  address ="0000000000000010";  break;  case "3":  address ="0000000000000011";  break;  case "4":  address ="0000000000000100";  break;  case "5":  address ="0000000000000101";  break;  case "6":  address ="0000000000000110";  break;  case "7":  address ="0000000000000111";  break;  case "8":  address ="0000000000001000";  break;  case "9":  address ="0000000000001001";  break;  case "10":  address ="0000000000001010";  break;  case "11":  address ="0000000000001011";  break;  case "12":  address ="0000000000001100";  break;  case "13":  address ="0000000000001101";  break;  case "14":  address ="0000000000001110";  break;  case "15":  address ="0000000000001111";  break;  case "16":  address ="0000000000010000";  break;  case "17":  address ="0000000000010001";  break;  case "18":  address ="0000000000010010";  break;  case "19":  address ="0000000000010011";  break;  case "20":  address ="0000000000010100";  break;  case "21":  address ="0000000000010101";  break;  case "22":  address ="0000000000010110";  break;  case "23":  address ="0000000000010111";  break;  case "24":  address ="0000000000011000";  break;  case "25":  address ="0000000000011001";  break;  case "26":  address ="0000000000011010";  break;  case "27":  address ="0000000000011011";  break;  case "28":  address ="0000000000011100";  break;  case "29":  address ="0000000000011101";  break;  case "30":  address ="0000000000011110";  break;  case "31":  address ="000000000011111";  break;  case "32":  address ="0000000000100000";  break;  case "Else":  address ="0000011100100000"; // we assumed that the adresse of Else is 0000011100100000  break;    }    } //closing of else that makes other i-format insturctions like addi,..etc        instructions\_to\_binary[i]=op+rs+rt+address;  }//closing of if case of beginning of i-format      }        }//END OF BIG FOR LOOP                  for(int i=0; i<Number\_of\_Instruction\_Lines-1 ;i++){  System.out.println(instructions\_to\_binary[i]);  } // part translation of code is now done and now moving to simulation part    System.out.println();    String register [][]= new String[32][2]; //2-D array of strings for the 32 register holding it's value and it's addresse  register[0][0]="00000"; register[0][1]="00000"; //$zero  register[1][0]="00001"; register[1][1]="00100"; //$at  register[2][0]="00010"; register[2][1]="00000"; //$v0  register[3][0]="00011"; register[3][1]="00000"; //$v1  register[4][0]="00100"; register[4][1]="00000"; //$a0  register[5][0]="00101"; register[5][1]="00000"; //$a1  register[6][0]="00110"; register[6][1]="00000"; //$a2  register[7][0]="00111"; register[7][1]="00000"; //$a3  register[8][0]="01000"; register[8][1]="00000"; //$t0  register[9][0]="01001"; register[9][1]="00000"; //$t1  register[10][0]="01010"; register[10][1]="00000"; //t2  register[11][0]="01011"; register[11][1]="00000"; //t3  register[12][0]="01100"; register[12][1]="00000"; //$t4  register[13][0]="01101"; register[13][1]="00000"; //$t5  register[14][0]="01110"; register[14][1]="00000"; //$t6  register[15][0]="01111"; register[15][1]="00000"; //$t7  register[16][0]="10000"; register[16][1]="00000"; //$s0  register[17][0]="10001"; register[17][1]="00000"; //$s1  register[18][0]="10010"; register[18][1]="00000"; //$s2  register[19][0]="10011"; register[19][1]="00000"; //$s3  register[20][0]="10100"; register[20][1]="00000"; //$s4  register[21][0]="10101"; register[21][1]="00011"; //$s5  register[22][0]="10110"; register[22][1]="00101"; //$s6  register[23][0]="10111"; register[23][1]="00000"; //$s7  register[24][0]="11000"; register[24][1]="00000"; //$t8  register[25][0]="11001"; register[25][1]="00000"; //$t9  register[26][0]="11010"; register[26][1]="00000"; //$k0  register[27][0]="11011"; register[27][1]="00000"; //$k1  register[28][0]="11100"; register[28][1]="00000"; //$gp  register[29][0]="11101"; register[29][1]="00000"; //$sp  register[30][0]="11110"; register[30][1]="00000"; //$fp  register[31][0]="11111"; register[31][1]="00000"; //$ra    System.out.println("The system before applying instructions contain registers with these values:");  Print.print2Darray(register);        for(int s=0; s<Number\_of\_Instruction\_Lines-1; s++){ ///////////////////////////////////////////////////////////////////////////////BIG FOR LOOP THAT TAKES EACH LINE OF INSTRUCTION AND WORK ON IT  char[] bit = instructions\_to\_binary[s].toCharArray(); //this array of characters hold the 32 bits of each instrucion line  char[] opCode= new char[6];  char[] RS= new char[5];  char[] RT= new char[5];  char[] RD= new char[5];  char[] shamt= new char[5];  char[] funct= new char [6];  char[] offset= new char[16];  char[] immediate= new char [26];  for(int w=0; w<opCode.length; w++){  opCode[w]=bit[w];  }    String OpcodeString = new String(opCode);      int c = Operations.CheckType(OpcodeString);      if(c==1){ //This means that it is R-type and we will divide the rest of the bits based on that  int p1=6;  for(int w=0; w<RS.length; w++){ // to identify rs in the instruction  RS[w]=bit[p1];  p1++;  }  int p2=11;  for(int w=0; w<RT.length; w++){ // to identify rt in the instruction  RT[w]=bit[p2];  p2++;  }  int p3=16;  for(int w=0; w<RD.length; w++){ // to identify rd in the instruction  RD[w]=bit[p3];  p3++;  }  int p4=21;  for(int w=0; w<shamt.length; w++){ // to identify shamt in the instruction  shamt[w]=bit[p4];  p4++;  }  int p5=26;  for(int w=0; w<shamt.length; w++){ // to identify function in the instruction  shamt[w]=bit[p5];  p5++;  }      } else if(c==2){ //This means that it is I-type and we will divide the rest of the bits based on that  int p1=6;  for(int w=0; w<RS.length; w++){ // to identify rs in the instruction  RS[w]=bit[p1];  p1++;  }  int p2=11;  for(int w=0; w<RT.length; w++){ // to identify rt in the instruction  RT[w]=bit[p2];  p2++;  }  int p3=16;  for(int w=0; w<offset.length; w++){ //to identify the offset/immediate  offset[w]=bit[p3];  p3++;  }  }      else if(c==3){ //This means that it is J-type and we will divide the rest of the bits based on that  int p1=6;  for(int w=0; w<immediate.length; w++){ // to identify the 26-bit address in the instruction  immediate[w]=bit[p1];  p1++;  }  }    if(OpcodeString.equals("000000")){ //This means the instruction line is add  String RsString = new String(RS); //converting array of charcters of address rs to string  String RtString = new String(RT); //converting array of charcters of address rt to string  String RdString = new String(RD); //converting array of charcters of address rd to string  int add1 = Handling.GettingValueFromReg(register, RsString ); //getting value from register rs  int add2 = Handling.GettingValueFromReg(register, RtString ); //getting value from register rt  int answer = Instructions.add(add1, add2); //adding two vlaues together  Handling.SettingValuetoReg(register, RdString, answer); //setting value of rd with the addition result  }    if(OpcodeString.equals("000100")){ //This means the instruction line is beq  String RsString = new String(RS); //converting array of charcters of address rs to string  String RtString = new String(RT); //converting array of charcters of address rt to string  int value1 = Handling.GettingValueFromReg(register, RsString ); //getting value from register rs  int value2 = Handling.GettingValueFromReg(register, RtString ); //getting value from register rt  int answer = Instructions.beq(value1, value2); //comparing the values of two registers      if(answer==1){ // if the two values are equal  String offsetString= new String(offset); // converting the offset char array to string  int OffsetInNumber = Integer.parseInt(offsetString, 2); //converting string offset to int  int currentAddress= Integer.parseInt(instruction[s][1],2); //getting current address in int  int TogoAdress= (currentAddress+4)+(4\*OffsetInNumber); //getting the wanted address to go in int  String TogoAddress\_String = Integer.toBinaryString(TogoAdress); //converting this wanted address to binary string  for(int ss=0; ss<Number\_of\_Instruction\_Lines-1; ss++){ //small for loop to search for correct line address to jump to it  if(TogoAddress\_String.equals(instruction[ss][1])){  int skip=(ss-s)-1;  s=(skip+s)-1;  break;  }  }//closing searching for correct line loop  }  }//closing of if of beq instruction    if(OpcodeString.equals("000010")){//this means that this is j instruction  String givenAddress = new String(immediate);  int givenAddressInt = Integer.parseInt(givenAddress, 2);  int nowAddress= Integer.parseInt(instruction[s][1],2);  int wantedAddress = (nowAddress+4)+(4\*givenAddressInt);  String wantedAddress\_String = Integer.toBinaryString(wantedAddress);  for(int ss=0; ss<-1; ss++){ //small for loop to search for correct line address to jump to it  if(wantedAddress\_String.equals(instruction[ss][1])){  int skip=(ss-s)-1;  s=(skip+s)-1;  break;  }  }//closing searching for correct line loop    String LastWord = instruction[s][0];  String LastWordArray[]= LastWord.split(" ");  String q=LastWordArray[1];  if(q.equalsIgnoreCase("Exit")){  break;  }    }//closing of if j        if(OpcodeString.equals("001010")){//this means that this is slti instruction  String RsString2 = new String(RS);  String immediateString = new String (offset);    int FirstNumber = Handling.GettingValueFromReg(register, RsString2);  int SecondNumber = Integer.parseInt(immediateString, 2);  String RtString2 = new String (RT);  int Answer2= Instructions.slti(FirstNumber, SecondNumber);  Handling.SettingValuetoReg(register, RtString2, Answer2);  }      }//Closing of Big for loop of instructions    System.out.println("The system after applying instructions contain registers with these values:");  Print.print2Darray(register);      }  } |

|  |
| --- |
| **Class Handling** |
| package archiproject;  public class Handling {    public static int GettingValueFromReg( String register [][], String RegAdress){ // To get value stored in specific register  int m=0;  for(int i= 0; i<32; i++){  if(RegAdress.equals(register[i][0])){  m=Integer.parseInt(register[i][1], 2);  break;  }  }  return m;  }    public static void SettingValuetoReg( String register [][], String RegAdress, int j){ // To store value in specific register  String k= Integer.toBinaryString(j);  for(int i= 0; i<32; i++){  if(RegAdress.equals(register[i][0])){  register[i][1]=k;  break;  }  }  }  } |

|  |
| --- |
| **Class Instructions** |
| package archiproject;  public class Instructions {    public static int add(int i, int j){  int k = i + j;  return k;  }    public static int beq(int i, int j){  if(i==j)  return 1;  else  return 0;  }    public static int slti(int i, int j){  if(i<j)  return 1;  else  return 0;    }  } |

|  |
| --- |
| **Class Operations** |
| package archiproject;  public class Operations {    public static int CheckType(String OpcodeString){  if(OpcodeString.equals("000000"))  return 1; // This means that the the operation type is R-type  else if (OpcodeString.equals("001010") || OpcodeString.equals("000100")) // This means that the the operation type is I-type  return 2;  else if (OpcodeString.equals("000010")) // This means that the the operation type is j-type  return 3;  else {  System.out.println("There is an error with your input");  return 0;  }    }    } |

|  |
| --- |
| **Class Print** |
| package archiproject;  public class Print {    public static void print2Darray( String register [][]){    for(int i= 0; i<register.length; i++){    System.out.println("Register with address "+register[i][0]+" contains value "+ register[i][1]);  }  System.out.println();  }    } |

# Testing the code

The meaning of the code of the research project written in pseudocode is:

**Note:** Let $at = X , $s5= I , $s6= Y

If ( I < 5)

{

X = 1

}

Else

{

X = 0

}

If ( X = = 0)

{

Y = 0 + 0

}

Else

{

Y = I + 0

}

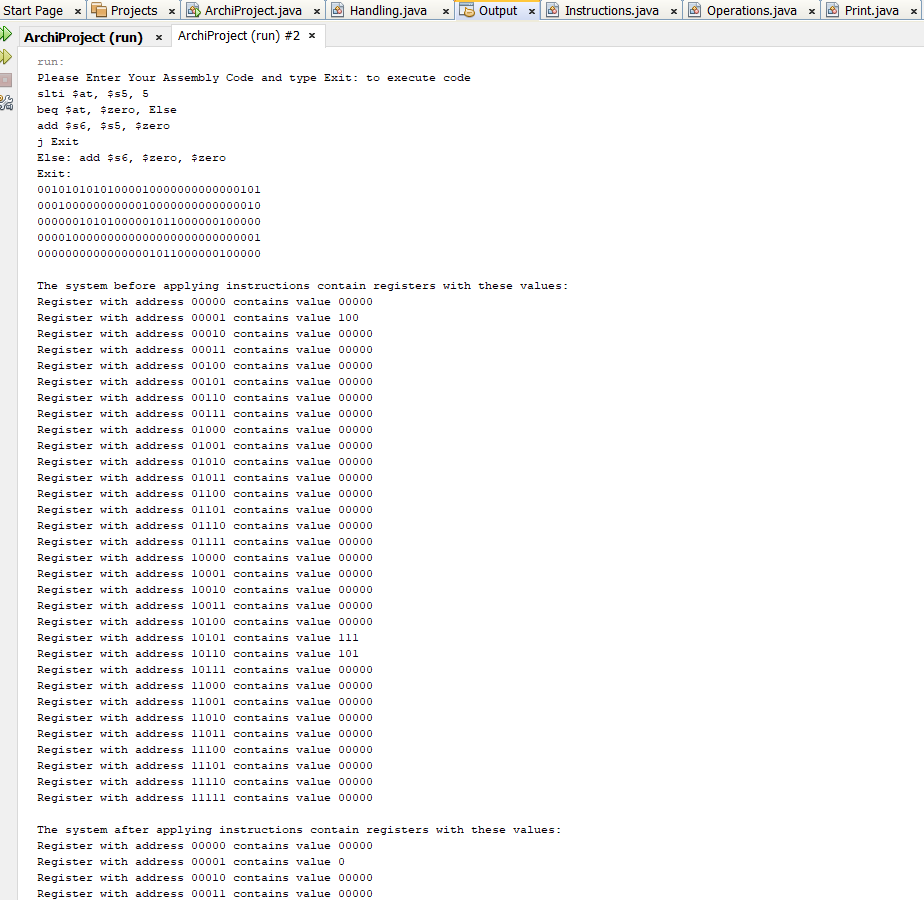
+

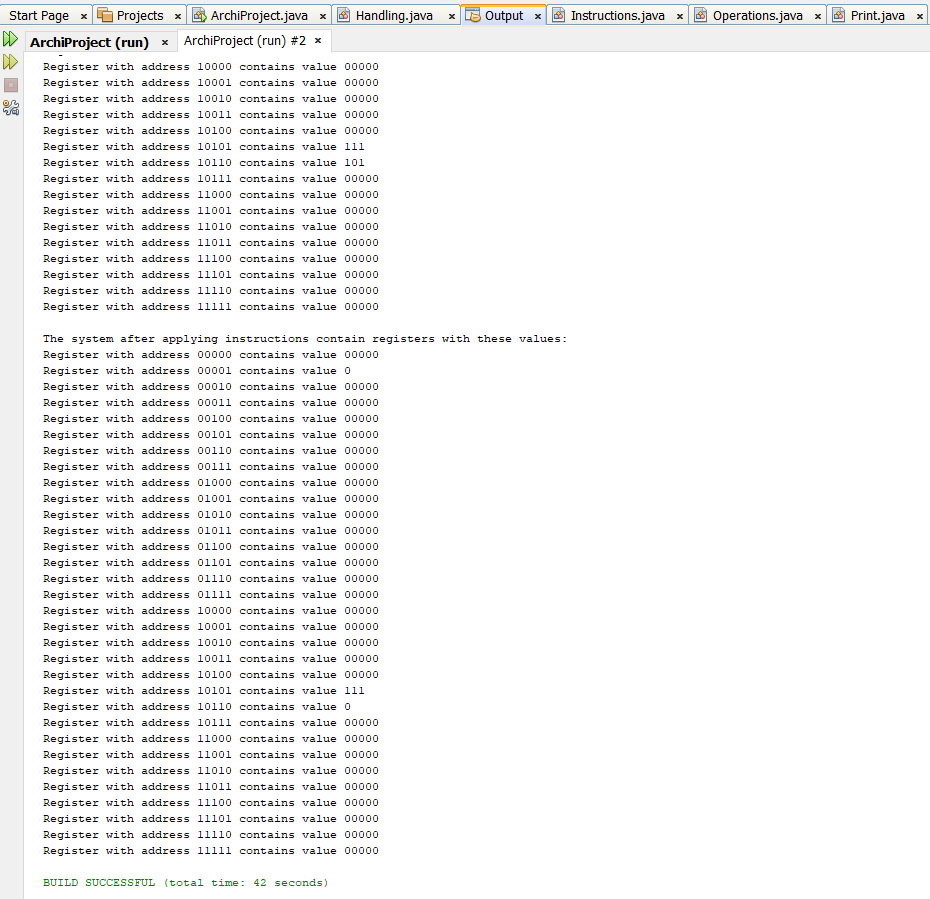
## Testing the first case:

To test the first code, we need to make register $s5 equal to a number larger than 5 so we decided to make the initial values in these register as the following

* **$s5 (with address 10101)** = 7 (in decimal) = 00111 (in binary)
* **$at (with address 00001) =** 4 (in decimal) = 00100 (in binary)
* **$s6 (with address 10110)=** 5 (in decimal) = 00101 (in binary)

Screenshots of the code execution:



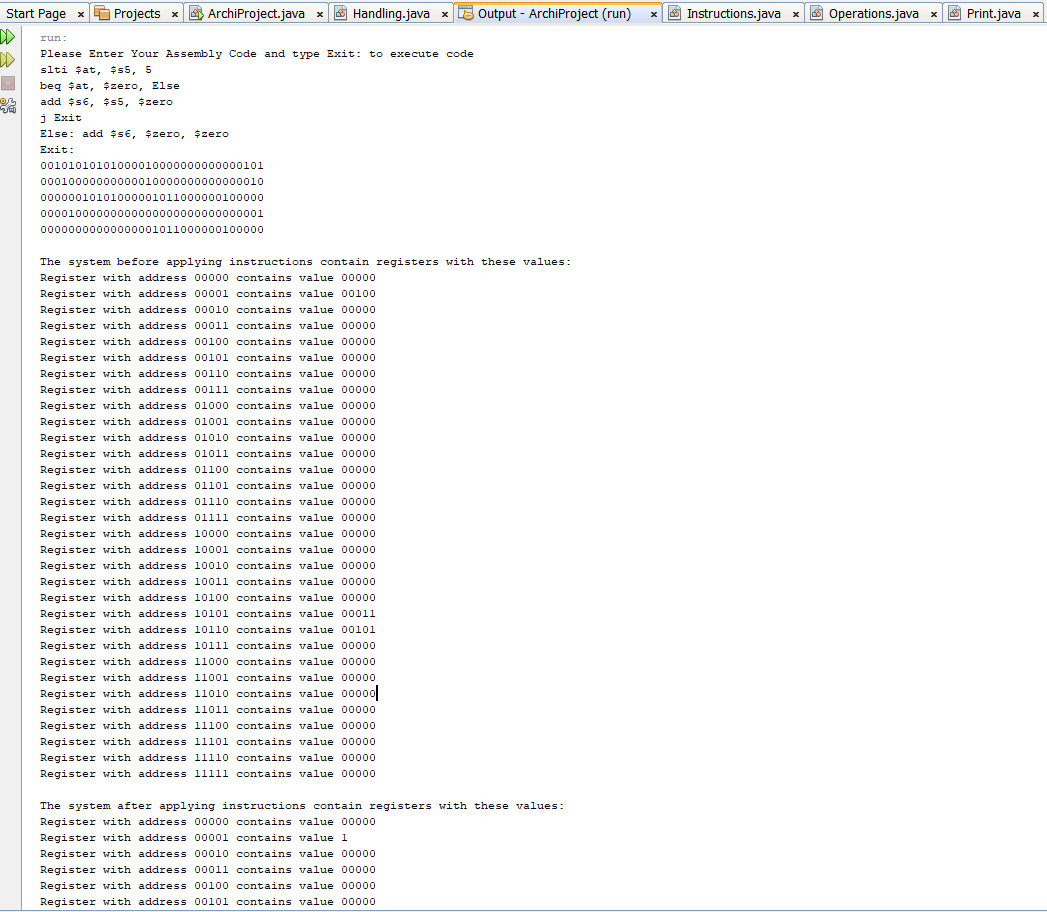


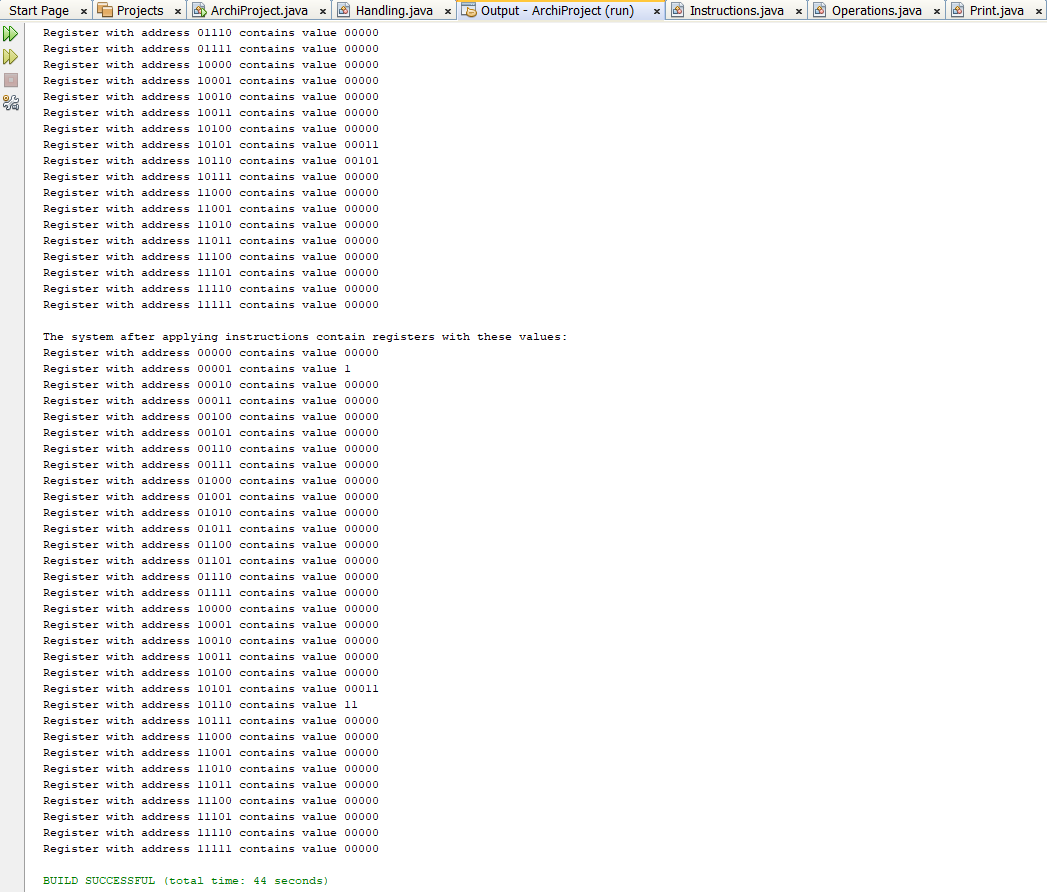
## Testing the second case

To test the second code, we need to make register $s5 equal to a number smaller than 5 so we decided to make the initial values in these register as the following

* **$s5 (with address 10101)** = 3 (in decimal) = 00011 (in binary)
* **$at (with address 00001) =** 4 (in decimal) = 00100 (in binary)
* **$s6 (with address 10110)=** 5 (in decimal) = 00101 (in binary)

Screenshots of the code execution:





.