May 27, 2019

Dual-Core Computer Design

COMP 206 Computer Architecture

Berfin Erdoğan

041701005

Ezgi Nihal Subaşı

041701022

Introduction:

In this project, it is aimed to design a dual-core computer with the benefit of a simple/basic level of software. In reality, assembler is the computer program that interprets software programs written into machine language, and instructions that can be executed by a computer [1]. After execution of the code, according to ISA (Instruction Set Architecture), the simple core microarchitecture must have a datapath and control unit to handle incoming 8-bit instructions. Besides, creating control signals will be beneficial for transferring between registers which are known as register transfer level (RTL) and ALU operations over a common 8-bit bus. Finally building Core Management Unit (CMU) circuit provide control of the execution of program instructions simultaneously in dual-core computer design.

Theory:

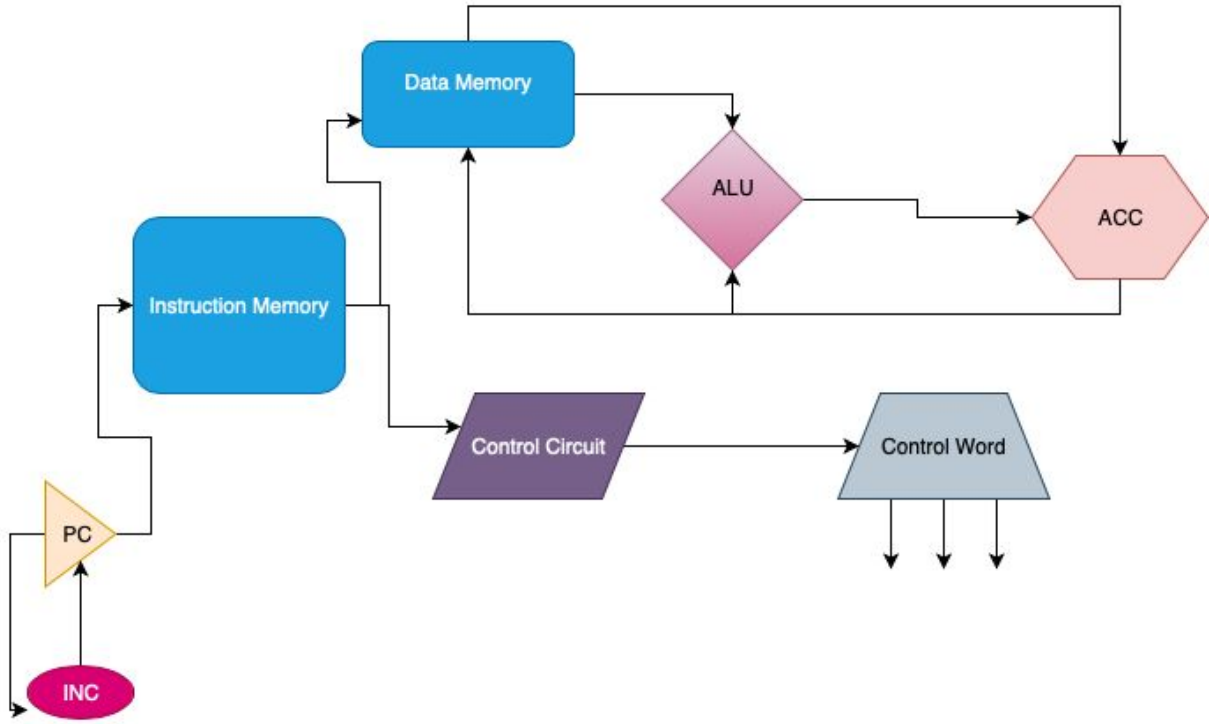

Fig 1.block diagram

Addresses in the ROMs determined with cores' program counters the instruction in that cells taken. CMU checks the first bit of instruction and determines if there will be a transfer between two cores. If the first bit is 1 than move operation will be executed. CMU selects send the cores the necessary signals.

The instructions that going the cores reduced to 8 bits. First 3 bits are opcode and remaining 5 bits are address. According to opcode control logic creates the necessary control word. And from that control word the appropriate operations performed.

Design Work:

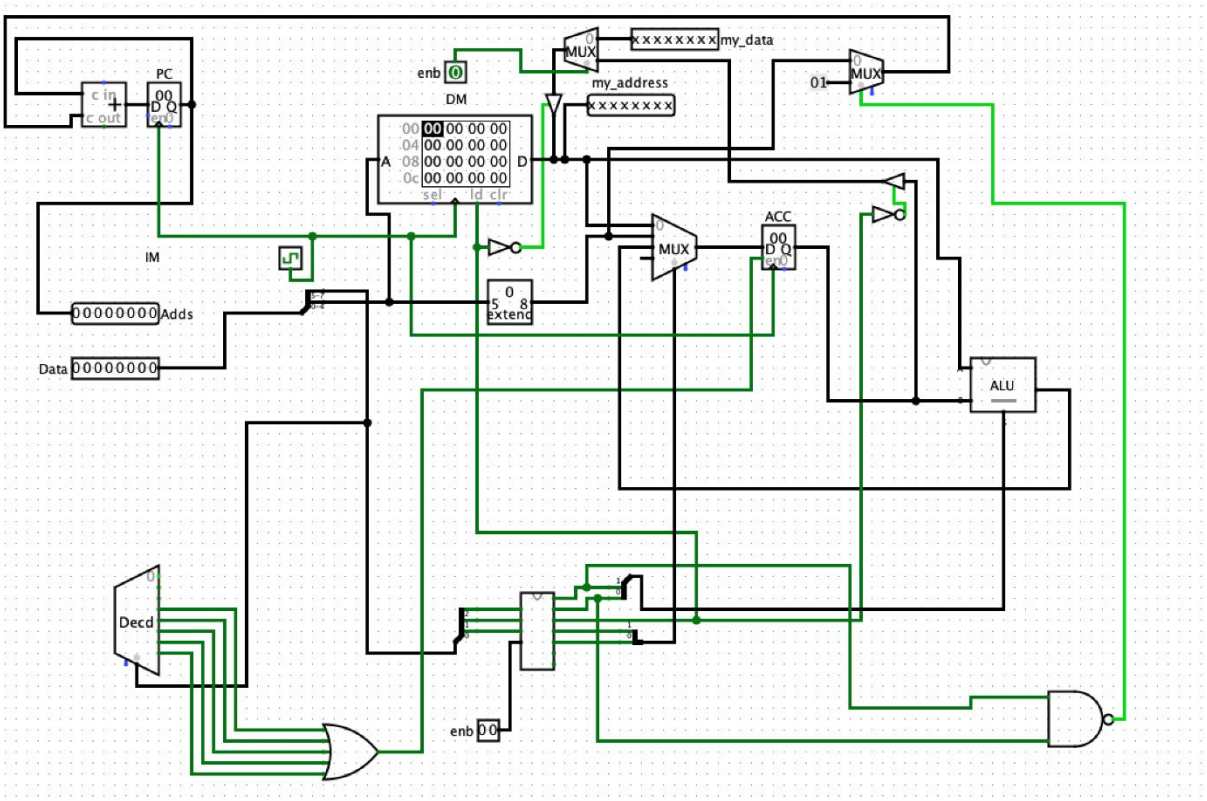

Fig 2.Main

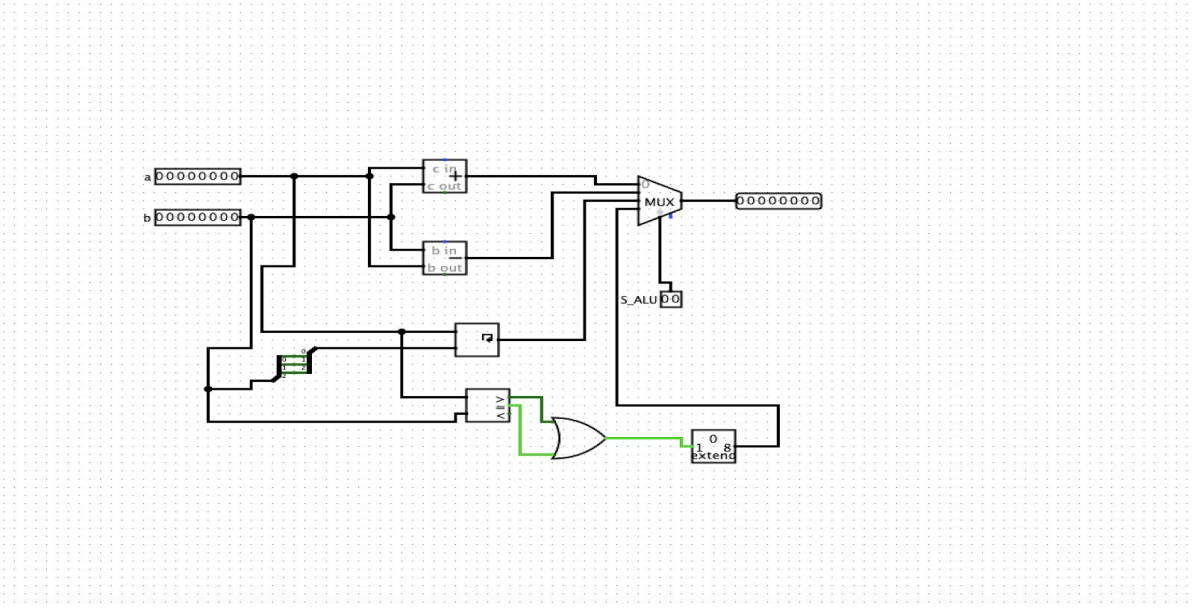

Fig 3.ALU

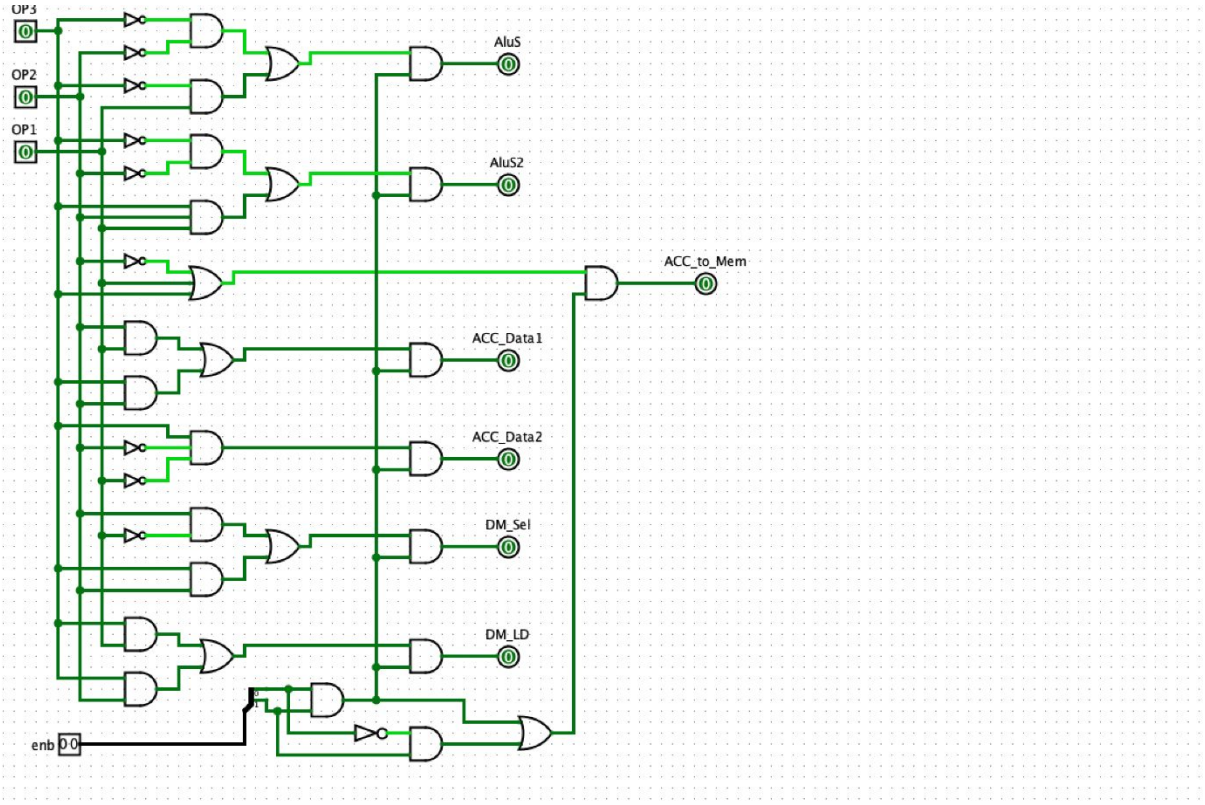

Fig 4.Control Logic

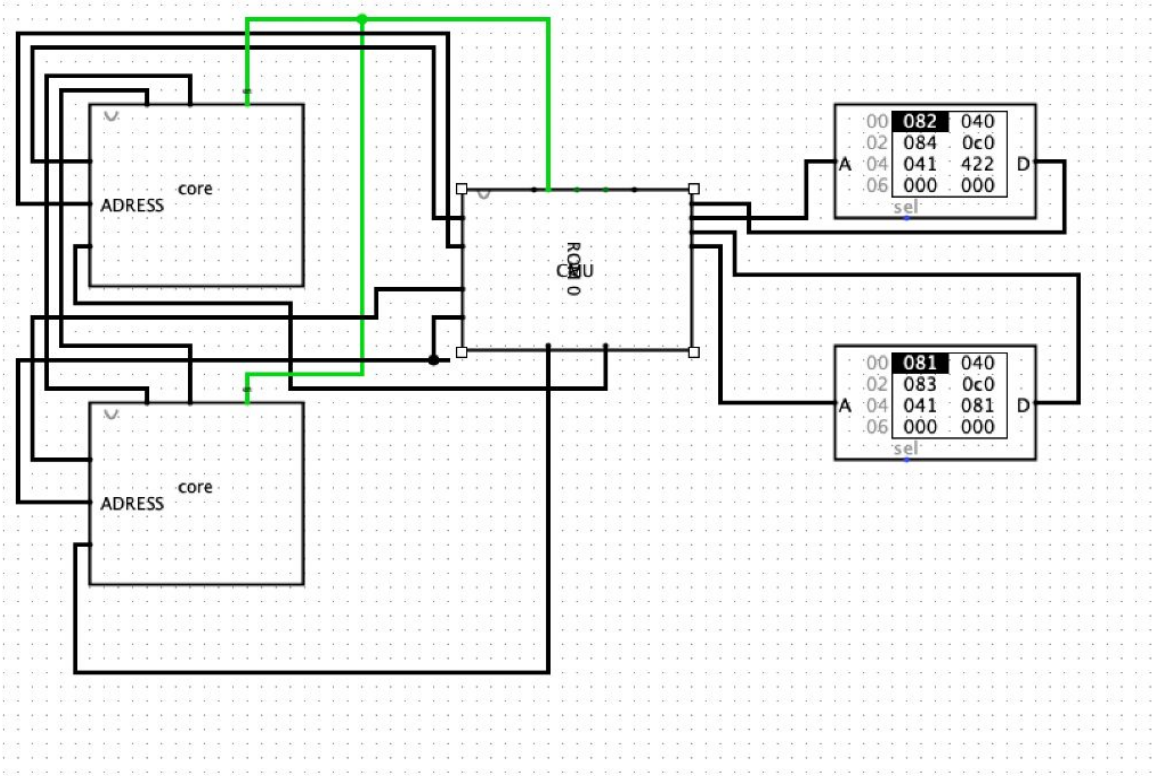

Fig 5.Dual Core

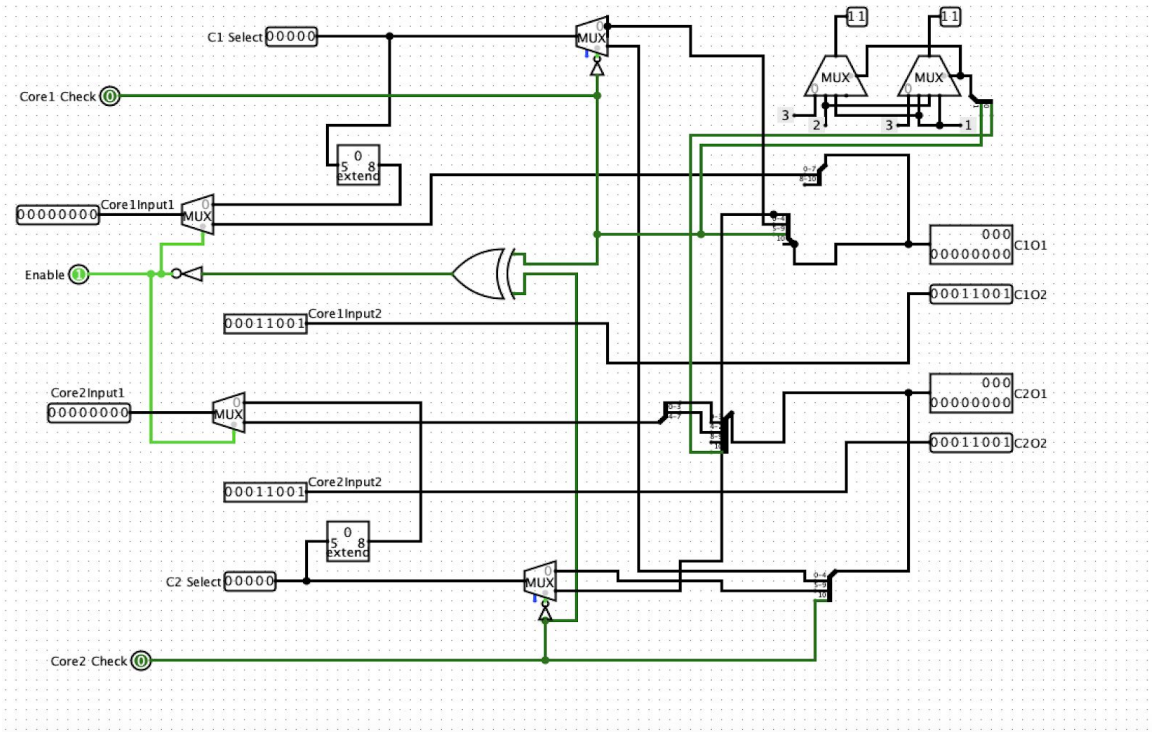

Fig 6.CMU

Demonstration:

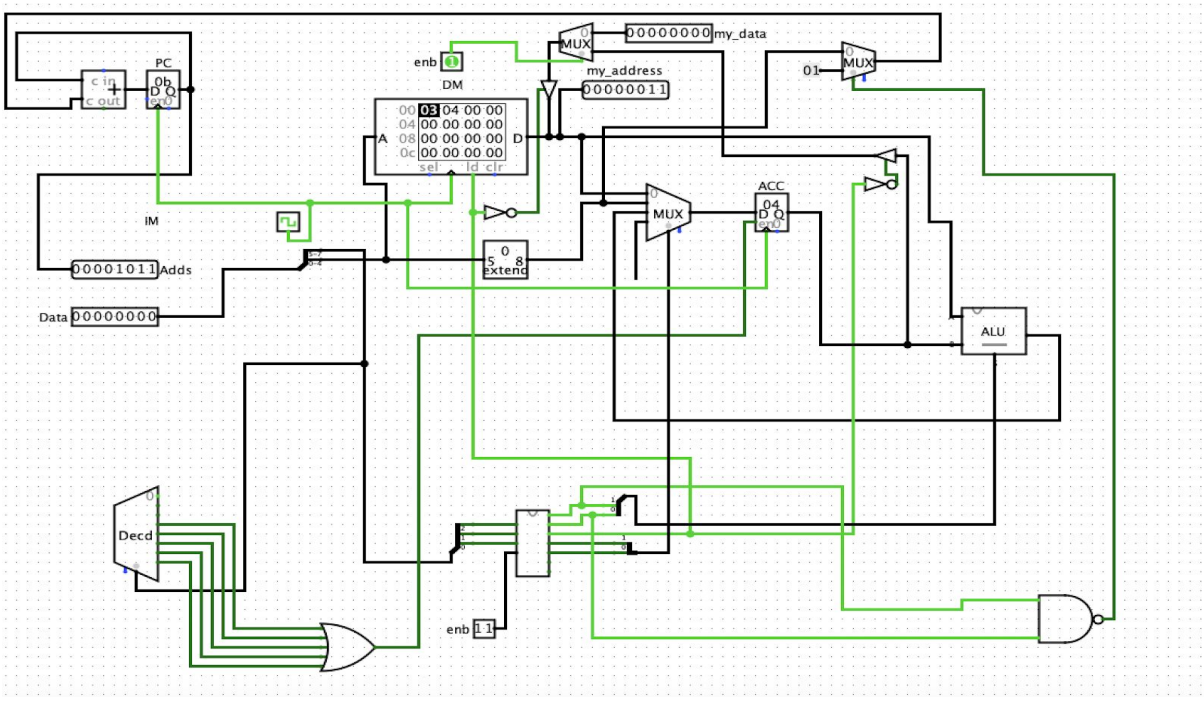

Fig 7.Single Core Demo

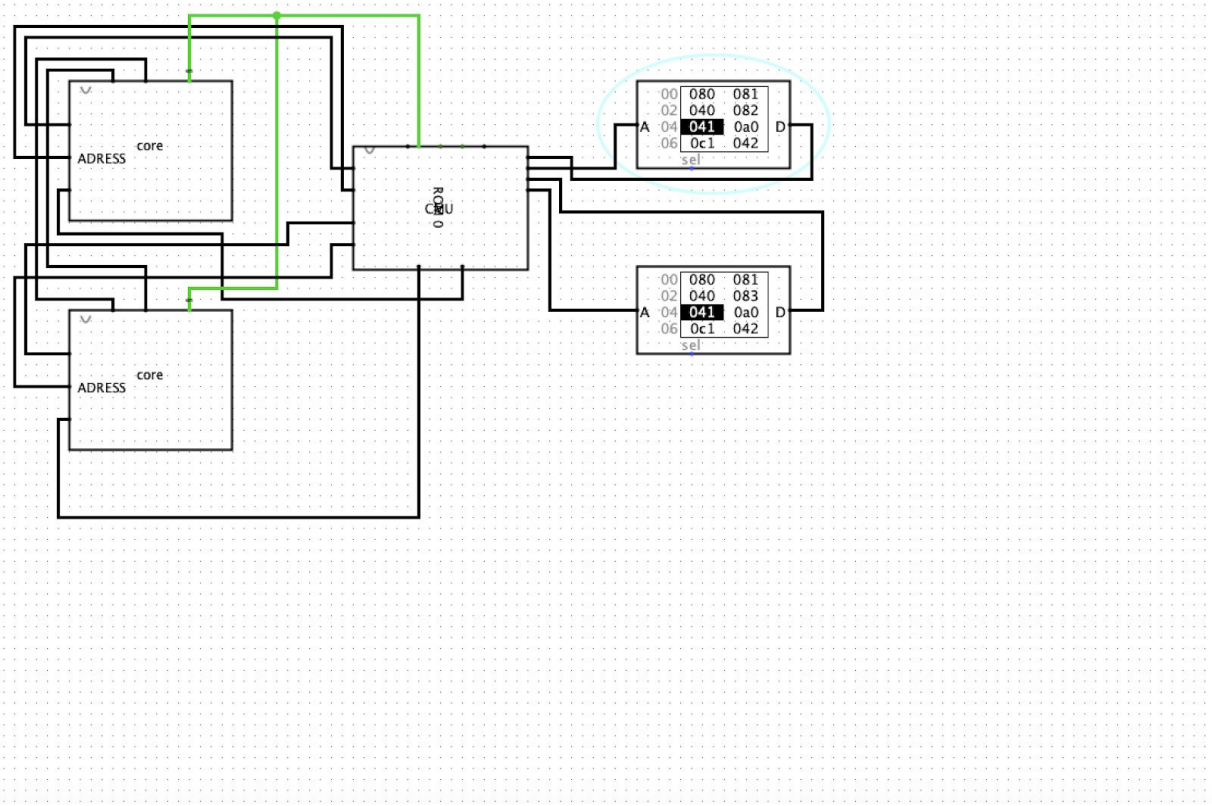

Fig 8.Multi 1 Main

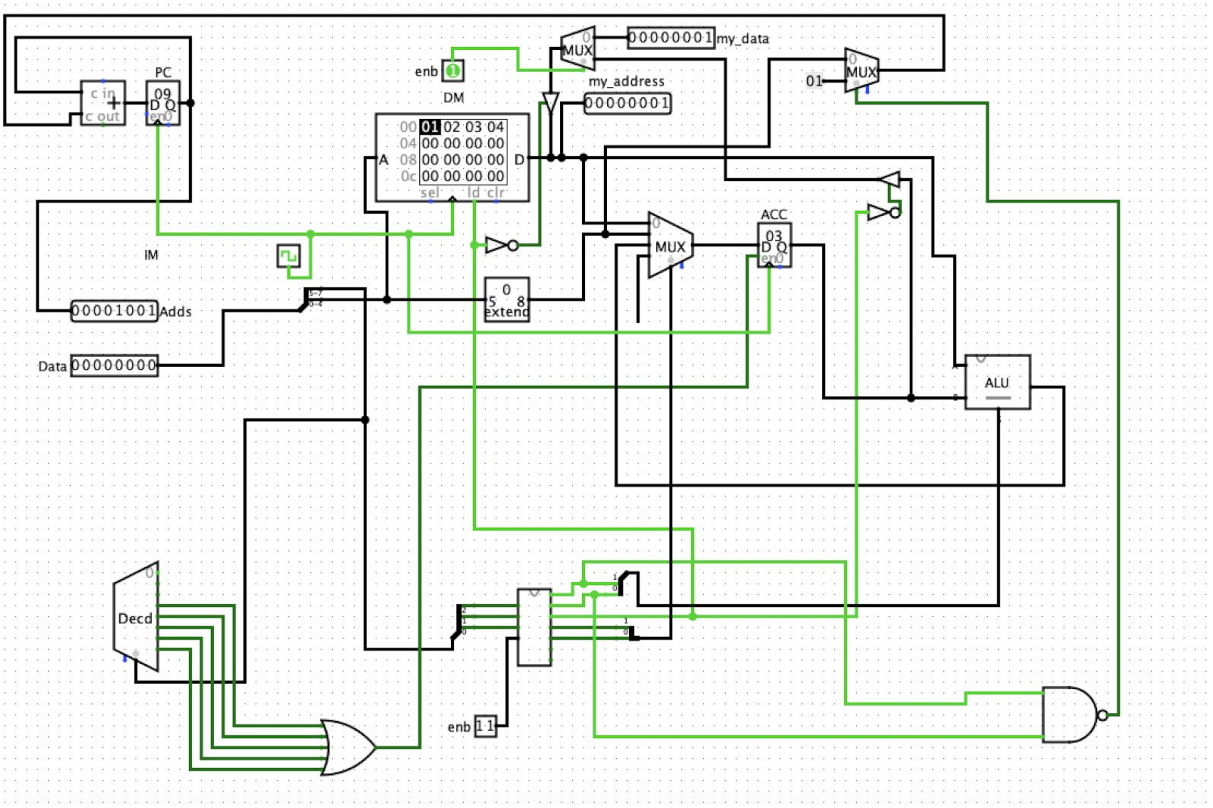

Fig 9.Multi 1 core 1

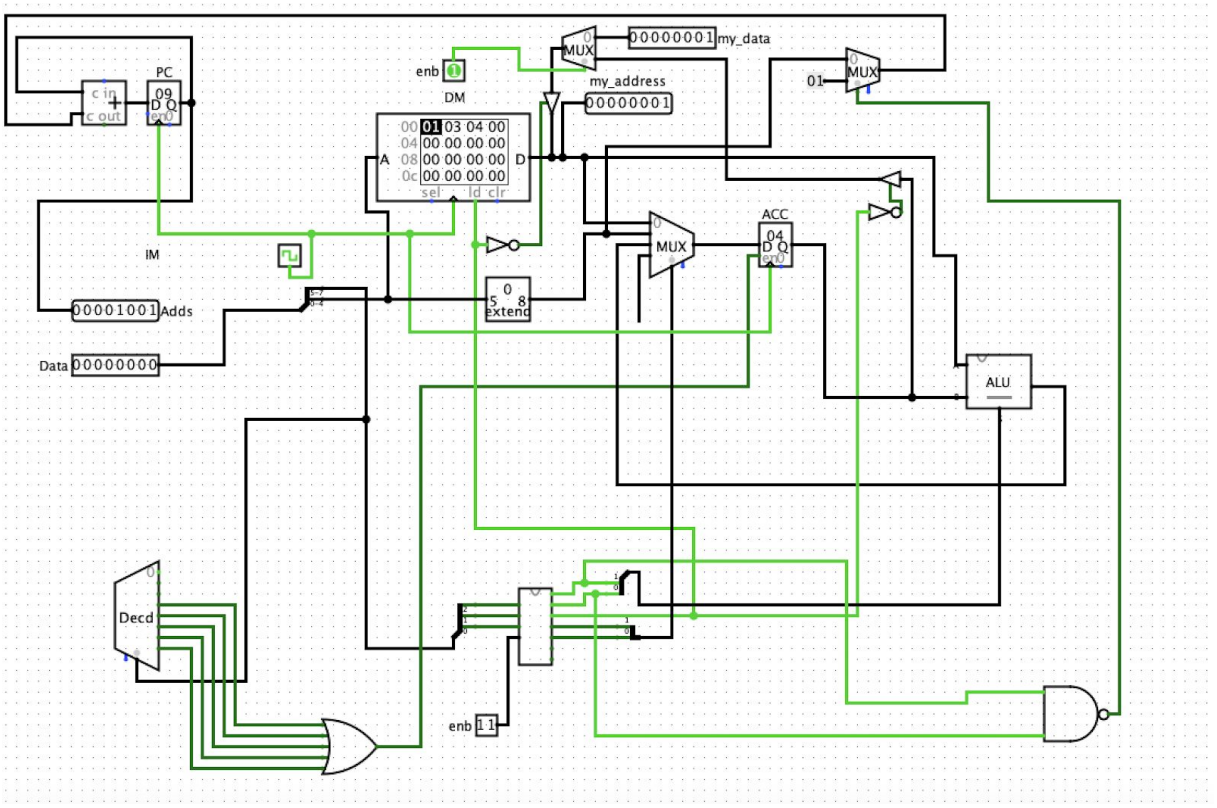

Fig 10.Multi 1 core 2

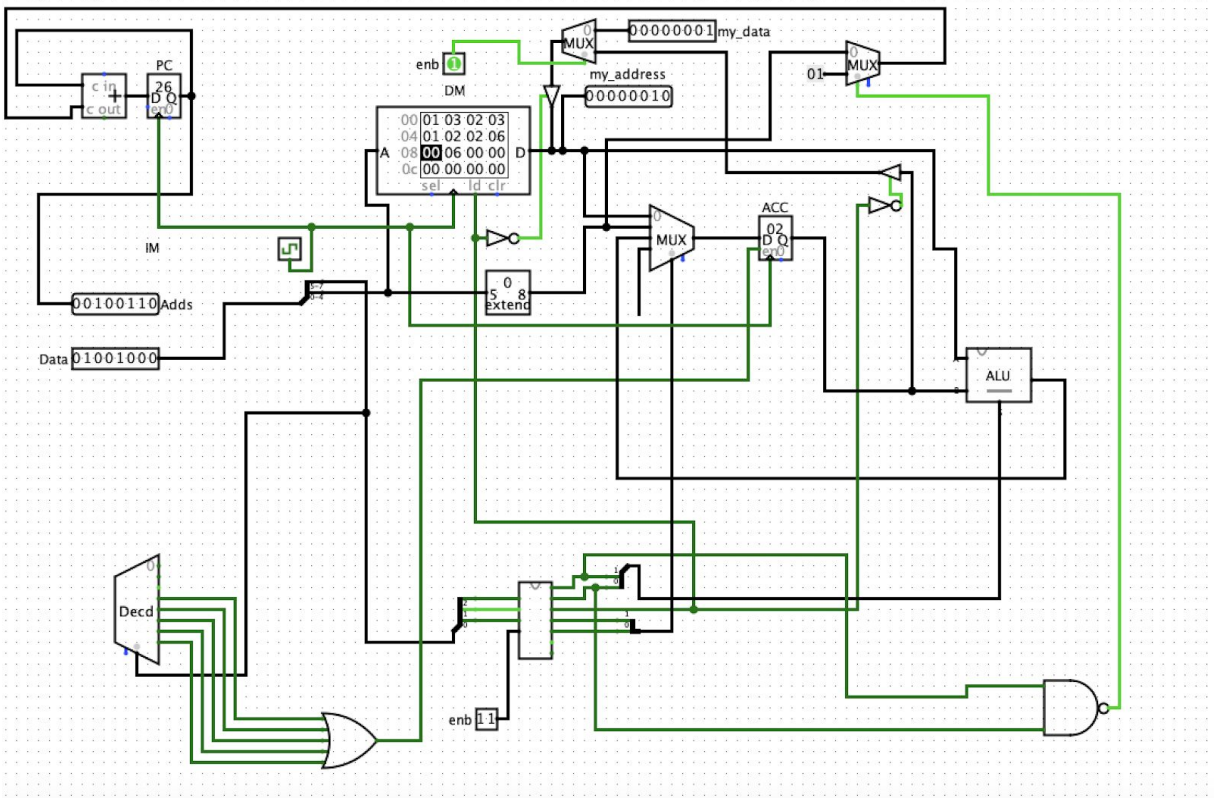

Fig 11.Multi 2 core 1

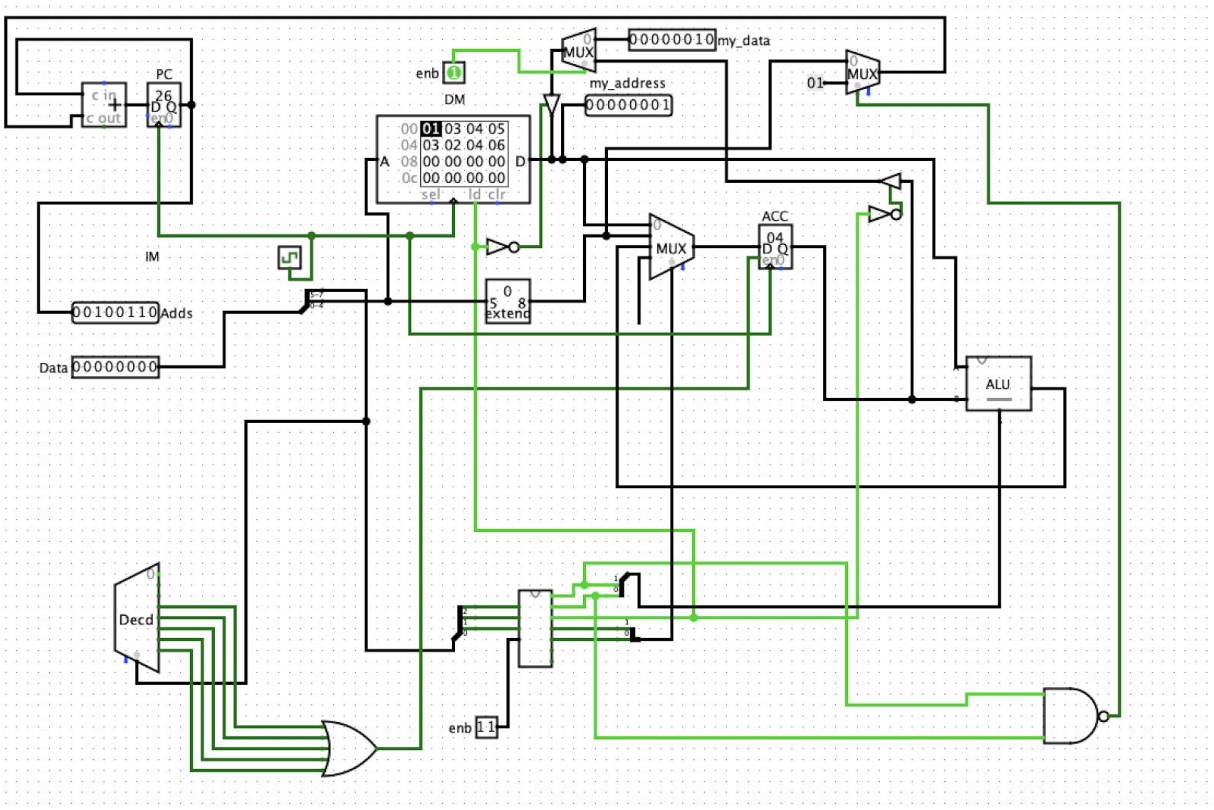

Fig 12.Multi 2 core 2

```
import java.awt.*;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Scanner;
public class Tests {
    public static void main(String[] args) throws IOException {


        //File fileTest = new File("/Users/berfinerdogan/Desktop/P-folder/Multi1/Presentation_M_core.txt");
        File fileTest = new File("/Users/berfinerdogan/Desktop/P-folder/Multi2/Presentation");
        //File fileTest = new File("/Users/berfinerdogan/Desktop/P-folder/Single/Presentation_Single_Core.txt");


        Scanner fileScanner = new Scanner(fileTest);

        FileOutputStream singleFileTest = new FileOutputStream("/Users/berfinerdogan/Desktop/single");
        FileOutputStream firstCore = new FileOutputStream("/Users/berfinerdogan/Desktop/core0");
        FileOutputStream SecondCore = new FileOutputStream("/Users/berfinerdogan/Desktop/core1");

        singleFileTest.write("v2.0 raw\n".getBytes());
        firstCore.write("v2.0 raw\n".getBytes());
        SecondCore.write("v2.0 raw\n".getBytes());


        while (fileScanner.hasNext()) {

            String testAssembly = fileScanner.nextLine();
            String assemblyControl = assemblyController(testAssembly) + " ";
            if (testAssembly.matches("\\w+\\s\\d+\\s\\d+") || testAssembly.matches("\\w+\\s\\w\\s\\d+")){
```

Fig 13. Java Code Part 1

```
        if (testAssembly.matches("\\w+\\s\\d+\\s\\d+") || testAssembly.matches("\\w+\\s\\w\\s\\d+")){
            String testAssemblyWords = testAssembly.substring(3,4);
            testAssemblyWords = testAssemblyWords.trim().toLowerCase();
            if (testAssemblyWords.equals("x")){
                firstCore.write(assemblyControl.getBytes());
                SecondCore.write(assemblyControl.getBytes());
            }
            else if (testAssemblyWords.equals("0")) firstCore.write(assemblyControl.getBytes());
            else    SecondCore.write(assemblyControl.getBytes());

        }


        else if (testAssembly.matches("\\w+\\s\\d+\\s\\d+\\s\\d+")){
            String[] wordAdressInputs = testAssembly.split(" ");
            wordAdressInputs[1] = wordAdressInputs[1].toLowerCase();
            String tempValue = "1"+calculateBinaryString(wordAdressInputs[2])+calculateBinaryString(wordAdressInputs[3]);

            if (wordAdressInputs[1].equals("0")) firstCore.write(Integer.toHexString(Integer.parseInt(tempValue.toString(), 2)).getBytes());
            else SecondCore.write(Integer.toHexString(Integer.parseInt(tempValue.toString(), 2)).getBytes());
        }
        else singleFileTest.write(assemblyControl.getBytes());
    }

}

public static String calculateBinaryString(String numberValue) {
    String binaryString = Integer.toBinaryString(Integer.parseInt(numberValue));
    if (binaryString.length() < 5){
        int length = binaryString.length();
        String finalValueOfNumber = "";
```

Fig 14. Java Code Part 2

```
public static String calculateBinaryString(String numberValue) {
    String binaryString = Integer.toBinaryString(Integer.parseInt(numberValue));
    if (binaryString.length() < 5){
        int length = binaryString.length();
        String finalValueOfNumber = "";
        for (int i = 0; i < 5 - length; i++)
            finalValueOfNumber+="0";
        finalValueOfNumber+=Integer.toBinaryString(Integer.parseInt(numberValue));
        return finalValueOfNumber.toString();
    }
    return binaryString;
}

public static String assemblyController(String control) {
    String[] controlWords = control.split("\\s");
    String finalOpCode = "";
    finalOpCode+=getOpCode(controlWords[0]);

    if (controlWords.length == 2) finalOpCode+=calculateBinaryString(controlWords[1]);
    else finalOpCode+=calculateBinaryString(controlWords[2]);
    int lastValue = Integer.parseInt(finalOpCode.toString(), 2);
    return Integer.toHexString(lastValue);
}

public static String getOpCode(String opCodes) {
    opCodes = opCodes.toUpperCase();
    if("BZ".equals(opCodes)) return "000";
    if("BN".equals(opCodes)) return "001";
    if("ST".equals(opCodes))  return "010";
    if("RL".equals(opCodes))  return "011";
    if("LI".equals(opCodes)) return "100";
```

Fig 15. Java Code Part 3

```
        }
        return binaryString;
    }

    public static String assemblyController(String control) {
        String[] controlWords = control.split("\\s");
        String finalOpCode = "";
        finalOpCode+=getOpCode(controlWords[0]);

        if (controlWords.length == 2) finalOpCode+=calculateBinaryString(controlWords[1]);
        else finalOpCode+=calculateBinaryString(controlWords[2]);
        int lastValue = Integer.parseInt(finalOpCode.toString(), 2);
        return Integer.toHexString(lastValue);
    }

    public static String getOpCode(String opCodes) {
        opCodes = opCodes.toUpperCase();
        if("BZ".equals(opCodes)) return "000";
        if("BN".equals(opCodes)) return "001";
        if("ST".equals(opCodes))  return "010";
        if("RL".equals(opCodes))  return "011";
        if("LI".equals(opCodes)) return "100";
        if("Lİ".equals(opCodes)) return "100";
        if("LM".equals(opCodes)) return "101";
        if("AD".equals(opCodes)) return "110";
        if("SB".equals(opCodes)) return "111";
        if("MV".equals(opCodes)) return "1";
        return null;
    }
}
```

Fig 16. Java Code Part 4

Discussion:

With this project, we observed using multi-core design architecture is more beneficial for processing instructions. Practical purposes and faster performance is provided usefully by multi-core design rather than single-core design.

Labor Authentication:

While preparing this report, doing this project and writing the code, we have made an equal and fair division of work.

References:

[1]"What is an Assembler? - Definition from Techopedia", *Techopedia.com*, 2019. [Online]. Available: https://www.techopedia.com/definition/3971/assembler. [Accessed: 28- May- 2019].

[2]*Ijarcsse.com*, 2019. [Online]. Available: http://ijarcsse.com/Before_August_2017/docs/papers/Volume_6/6_June2016/V6I6-0212.pdf. [Accessed: 28- May- 2019].

[3]"The Guide to Being a Logisim User", *Cburch.com*, 2019. [Online]. Available: http://www.cburch.com/logisim/docs/2.3.0/guide/index.html. [Accessed: 28- May- 2019].