ISTANBUL TECHNICAL UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

BLG 222E
COMPUTER ORGANIZATION
PROJECT REPORT

PROJECT NO : 3
Due Date : 03.06.2020
GROUP NO : G2

GROUP MEMBERS:

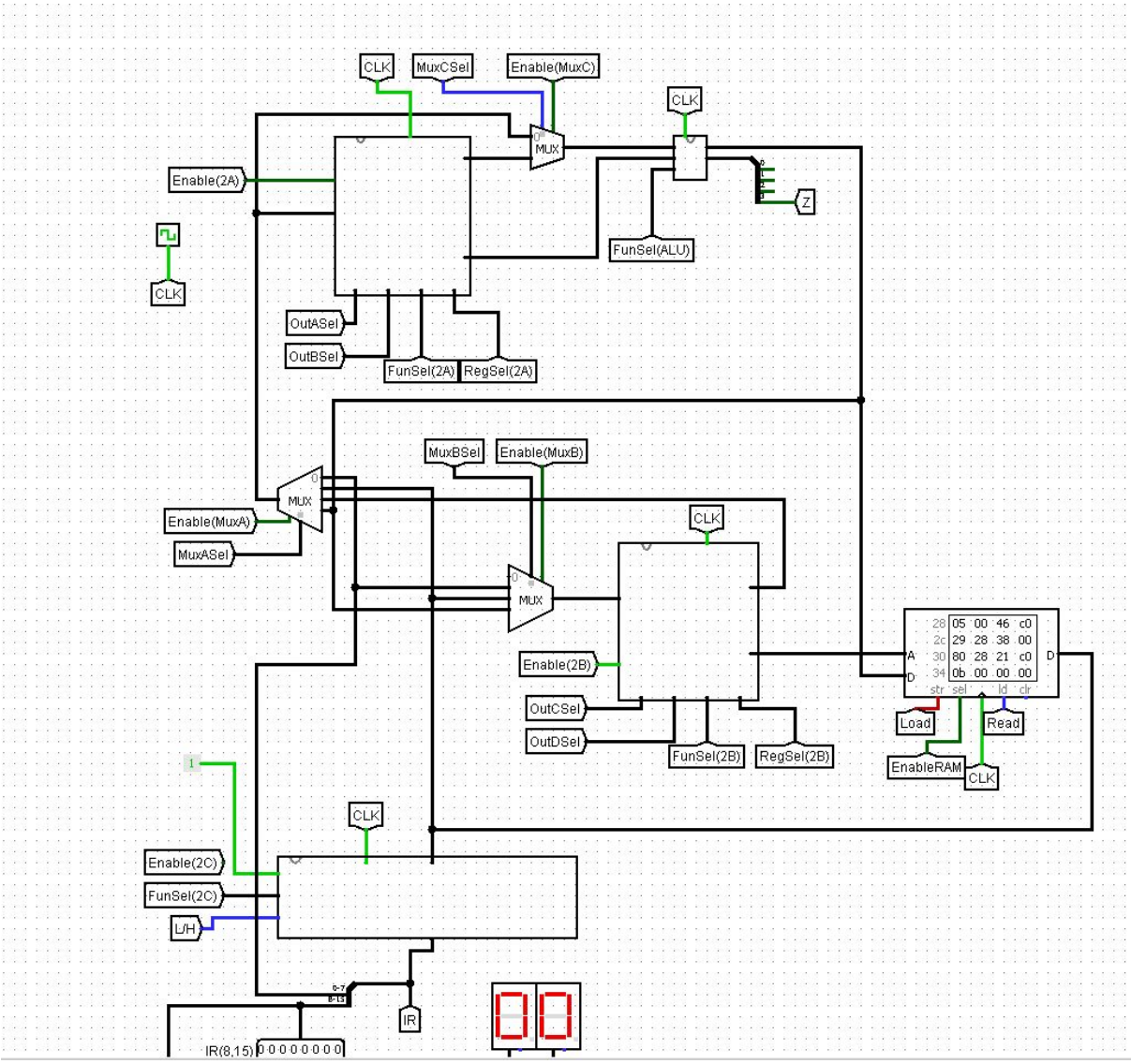
150180008 : Lal Verda Cakir
150170909 : Elaa Jamazi
150140009 : Serhat Vural
150180038 : Karahan Sezer

1 INTRODUCTION

In project 3, we tried to implement a design of a small computer, using project 2' content and a Control Unit (CU) that we designed during this project. All the inputs of the part coming from project 2 are provided by the CU. The CU provides these inputs by processing the instructions provided in the text of the assignment (Instruction type 1, Instruction type 2). We have a total of 19 Opcodes that control the outputs of the CU, while taking in consideration the remaining part of the instruction (exp: SRCREG, DESTREG, Adressing Mode, Adress,etc…).

2 PROJECT PARTS

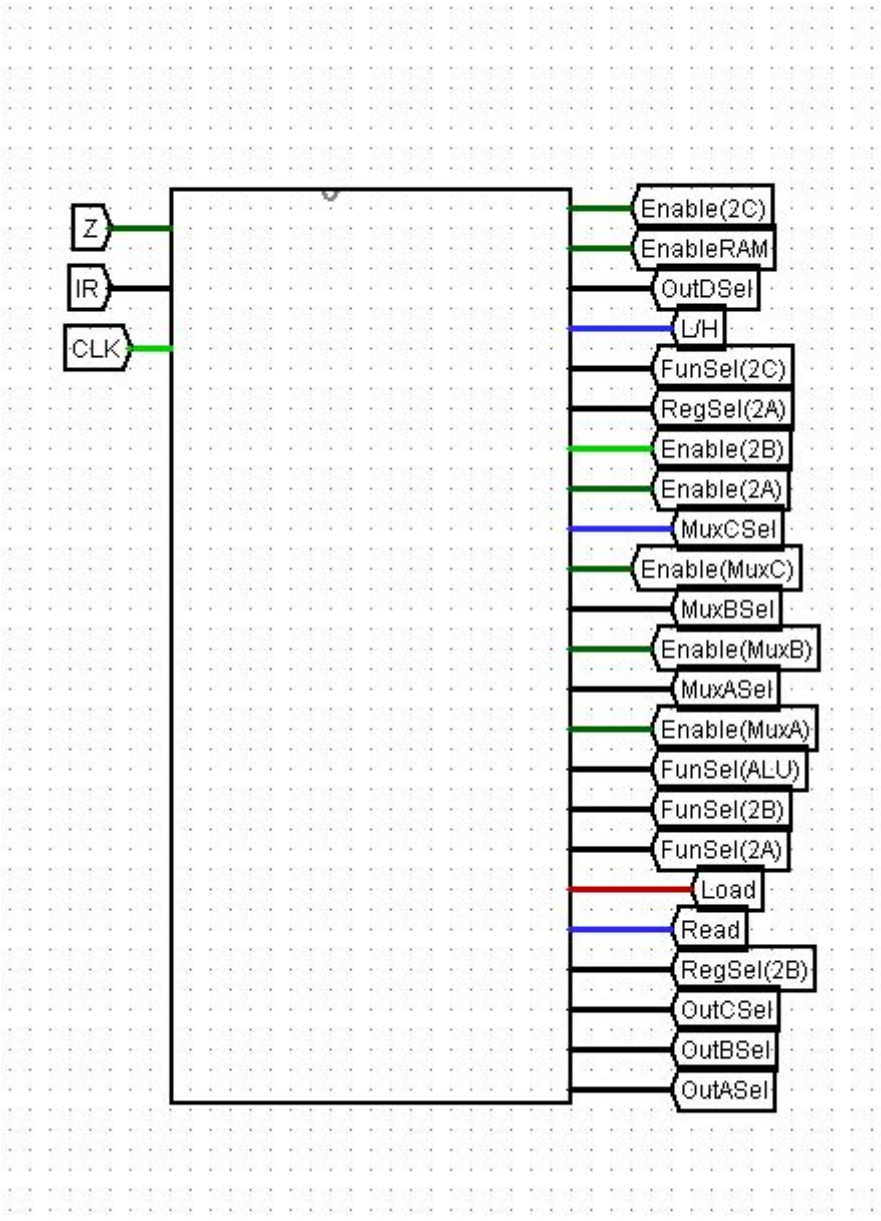
2.1 FIRST PART



The first part consists of the circuit we prepared during project 2. We used tunnels to supply all the inputs of this circuit. These inputs are actually outputs of the Control Unit (Read

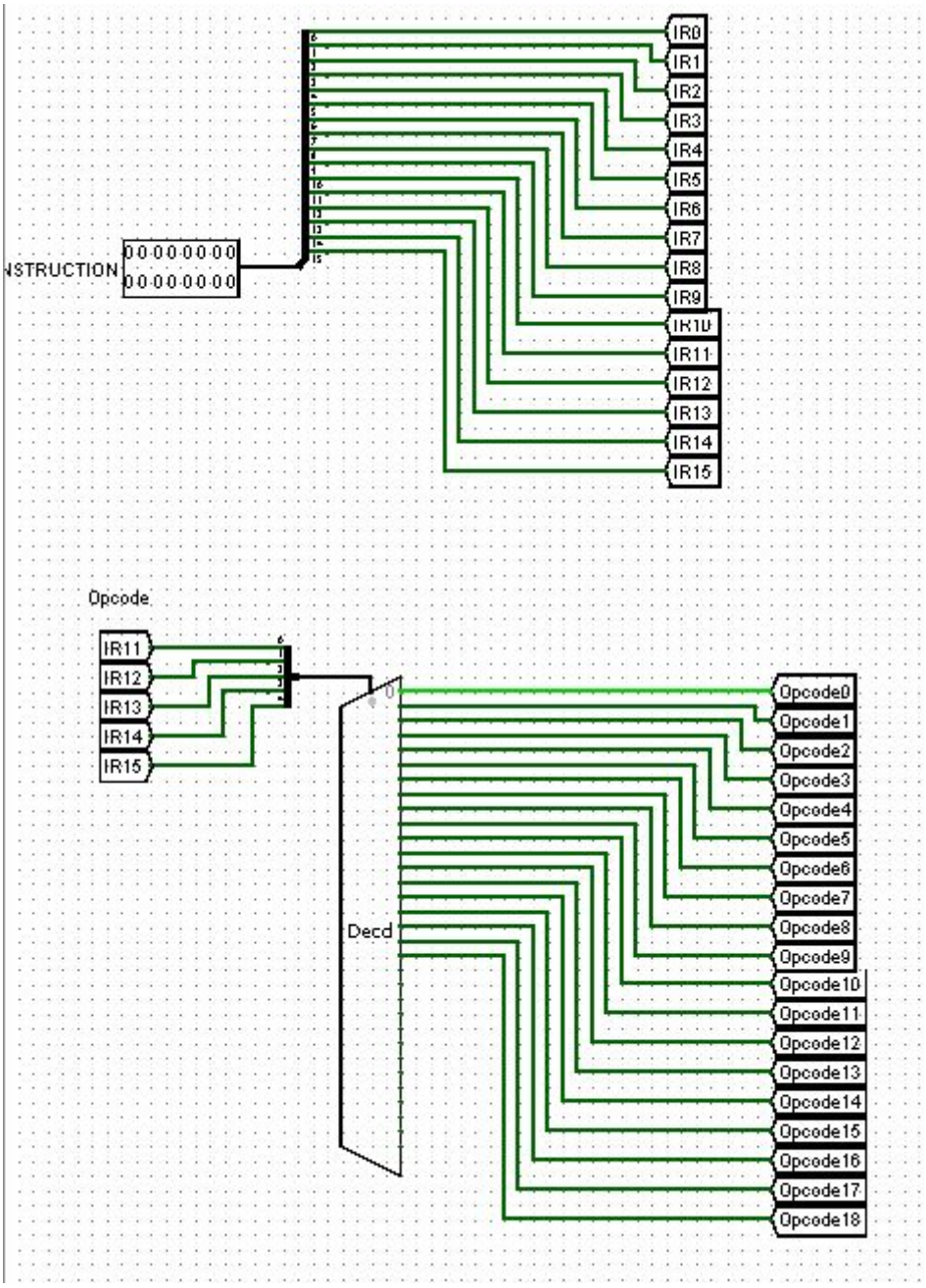
Second Part for more details). We had some mistakes before with our 2nd project. Overflow output was wrong and the ordering of MUXA inputs and MUXB inputs was wrong as well but we fixed them. Other than that we did not change anything about this part that is why we will not delve into more details about it since it was explained during Project 2.

2.2 SECOND PART

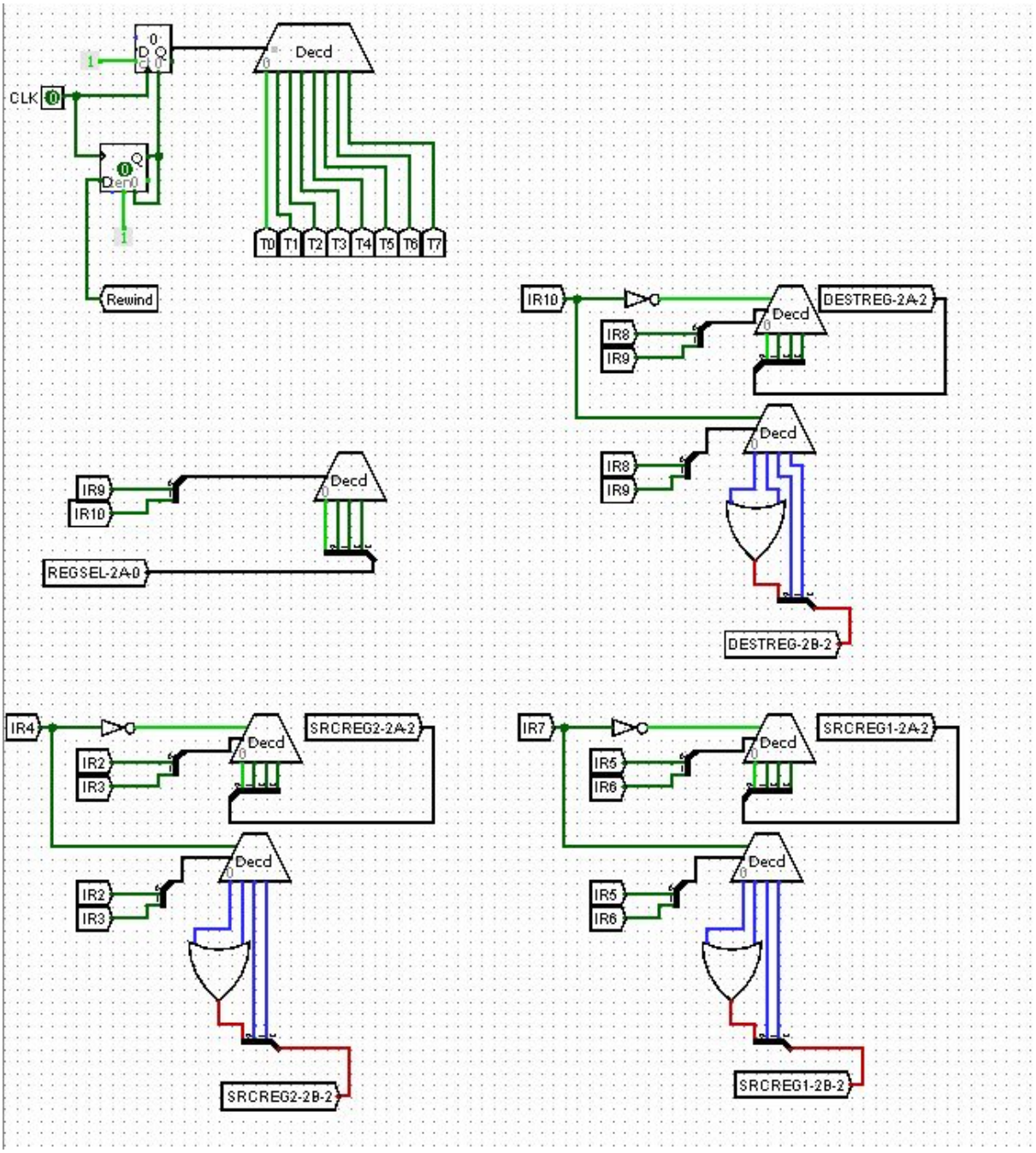


The second part of our project is a Control Unit. The inputs of the control unit are IR: the instruction supplied from part2C, Z: the Zero flag from the ALU, and CLK: a clock signal. The CU has various outputs that are used to control the functioning or part 1 depending on the provided instruction.

The insides of the CU:

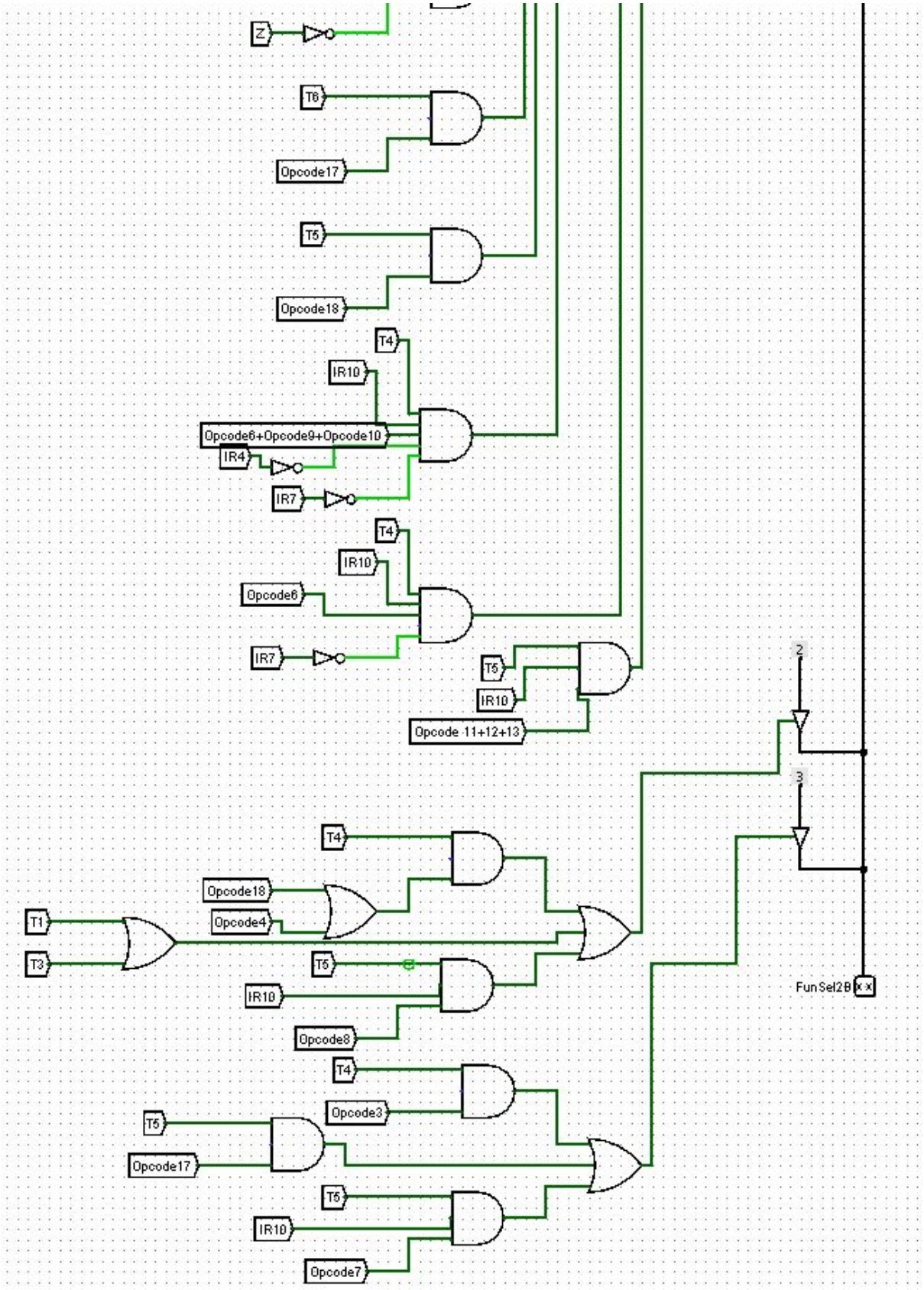


We used a splitter to divide the bits of the instruction to figure out its different parts. The OPCODE, the REGSEL, the ADDRESSING MODE, and the ADDRESS for type 1 instruction and the OPCODE, DESTREG, SRCREG1, and SRCREG2 for type 2 instruction. We used a decoder to figure out the correct OPCODE out of the first 5 bits of the instruction.



We further used Decoders to figure out the exact locations of DESTREGs and SRCREGs depending on the instruction. DESTREG-2A-2 output helps us decided Regsel for part 2A when the DESTREG is in part 2A. DESTREG-2B-2 output helps us decided Regsel for part 2B when the DESTREG is in part 2B. SRCREG-1A-2 and SRCREG-2A-2 are the outputs that help us decided Regsel for part 2A when the SRCREGs are in part 2A . SRCREG-1B-2 and SRCREG-2B-2 are the outputs that help us decided Regsel for part 2B when the SRCREGs are in part 2B of the first part of the project. We used a counter and a decoder to figure out the times (T0...T7) during which specific actions are taken in the circuit to deliver the output specified by the instruction input. Our Rewind input is used to restart the counter to T0 once the previous instruction is completed.

The Rewind part, alongside the different outputs of the CU (OutASel, FunSel, Load, L/H, RegSel, etc…..) are implemented using similar mechanism and logic:



For Example, here, Funsel2B output is set by a series of AND gates and OR gates that are supplied as enable to a buffer that has our desired FunSel2B output as its input. Most of the other outputs are designed in the same manner.

3 RESULTS

In this project we implemented a machine that can perform a set of operations, using the instructions that are given to it, in other words a hard-coded computer.

Instructions are written by the user to RAM and the system reads these instructions and passes them to the control unit, then the control unit decides the inputs of the components to perform that operation.

4 DISCUSSION

Our design process began with dividing the whole process into smaller parts. After deconstructing the OPCODE's, we made the circuit that produces the MUX outputs, OUT selectors, REG selectors and other necessary outputs. We had a chance to observe how to realize, load and run an instruction using the circuits we did earlier such as register systems and ALU. To control the time and loading procedures from RAM, we faced some problems such as clearing the time point (rewind) and making the RAM available for only reading or loading. Also there were some problems with overlapping instructions and we debugged (running the instructions manually) to find the errors. To run the instructions given in homework, we managed to transform into binary codes and then load them to RAM.

Here are the instructions break-down into clock signals.

T0: IR(8-15) ← M[PC]

RegSel(2B): 000;
OutDSel: 00;
Read: 1;
Load: 0;
Enable(2C): 1;
L/H:0;
FunSel(2C):11;

T1: PC ← PC + 1

FunSel(2B): 10;
RegSel(2B):001;
OutCSel: 00;
Enable(2C): 0;

T2: IR(0-7) ← M[PC]

OutDSel: 00;
RegSel(2B): 0000;
Read: 1;
Load: 0;
Enable(2C): 1;
L/H: 1;
FunSel(2C): 11;

T3: PC ← PC + 1

FunSel(2B): 10;
RegSel(2B): 001;
OurCSel: 00;
Enable(2C): 0;

After Fetch and Decode is done, each operation continues according to decode Opcode.

Opcode0: Rx ← Value

Note: Addressing mode IM, D
IR8 = 0
Update:
Opcode0T4:
MuxASel = 00
FunSel(2A) = 01
RegSel(2A) = REGSEL-2A-0

IR8 = 1
Opcode0T4:
OutDSel = 01;
Read = 1;
Load = 0;
Opcode0T5:
MuxASel = 01;
OutASel = IR(9,10);
FunSel(2A) = 01;
RegSel2A = REGSEL-2A-0;

Opcode1: Value ← Rx

Note: Addressing mode D
Opcode1T4:
RegSel(2A) = regsel-2a-0;

OutASel = IR(9,10);
MuxCSel = 1;
FunSel(ALU) = 0000;
OutDSel = 01;
Load = 1;
Read = 0;

Opcode2: DESTREG ← SRCREG1

IR7 = 1, IR10 = 1

Opcode2T4:

OutCSel = SRCREG1-2B-2[2][1]
MUXASEL = 10
MUXCSEL = 0
FUNSEL(ALU) = 0000
MUXBSEL=11
FUNSEL(2B) = 01
REGSEL(2B) = DESTREG-2B-2

IR7 = 1, IR10 = 0

Opcode2T4:

OutCSel = SRCREG1-2B-2[2][1]
MuxASel=10
FunSel(2A) = 01
RegSel(2A) = DESTREG-2A-2

IR7 = 0, IR10 = 1

Opcode2T4:

OutBSel=SRCREG1[1][0];
FunSel(ALU) = 0001;
MuxBSel = 11;
FunSel(2B)=01;
RegSel(2B) = DESTREG-2B-2;

IR7 = 0, IR10 = 0

Opcode2T4:

OutBSel=SRCREG1[1][0];
FunSel(ALU) = 0001;
MuxBSel = 11;
FunSel(2B)=01;
RegSel(2B) = DESTREG-2B-2;
MuxASel=11;
FunSel(2A)=01;
RegSel(2A) = DESTREG-2A-2;

Opcode3: M[SP] ← Rx, SP ← SP - 1

Opcode4T4:

Load = 1;
Read = 0;
OutBSel = REGSEL-2A-0;
FunSel(ALU)= 0001;
FunSel(2B) = 11;
RegSel(2B) = 10;

Opcode4: SP ← SP + 1, Rx ← M[SP]

Opcode4T4:

FunSel(2B) = 10;
RegSel(2B) = 100

Opcode4T5:

OutDSel = 10;
Read = 1;
Load = 0;
MuxASel = 01;
FunSel(2A) = 01;
RegSel(2A) = REGSEL-2A-0

Opcode5: DESTREG ← SRCREG1 + SRCREG2

rewind?

IR10 = 0, IR7=0, IR4 = 0

Opcode5T4:

OutBSel = IR(2,3)
OutASel = IR(5,6)
MuxCSel = 1;
FunSel(ALU) = 0100
MuxASel = 11;
RegSel = DESTREG-2A-2
FunSel (2A)= 01

IR10 = 0, IR7=0, IR4 = 1

Opcode5T4:

OutCSel = SRCREG1-2B-2[2][1]
MuxASel = 10
MuxCSel = 0;

Opcode5T5:

OutBSel = IR(2,3);
FunSel(ALU) = 0100
MuxASel = 11;
RegSel = DESTREG-2A-2
FunSel (2A)= 01

IR10 = 0, IR7=1, IR4 = 0

Opcode5T4:

OutCSel = SRCREG1-2B-2[2][1]
MuxASel = 10
MuxCSel = 0;

Opcode5T5:

OutBSel = IR(5,6);
FunSel(ALU) = 0100
MuxASel = 11;
RegSel = DESTREG-2A-2
FunSel (2A)= 01

IR10 = 1, IR7=0, IR4 = 0

Opcode5T4:

OutBSel = IR(2,3)
OutASel = IR(5,6)
MuxCSel = 1;
FunSel(ALU) = 0100
MuxBSel = 11
FunSel(2B) = 01
RegSel(2B) = DESTREG-2B-2

IR10 = 1, IR7=1, IR4 = 0

Opcode5T4:

OutCSel = SRCREG1-2B-2[2][1]
MuxASel = 10
MuxCSel = 0
OutBSel = IR(2,3)
FunSel(ALU) = 0100
MuxBSel = 11
FunSel(2B) = 01
RegSel(2B) = DESTREG-2B-2

IR10 = 1, IR7=0, IR4 = 1

Opcode5T4:

OutBSel = IR(5,6)
OutCSel = SRCREG2-2B-2(1,2)
MuxASel = 10
MuxCSel = 0
FunSelALU = 0100

IR10 = 1, IR7=1, IR4 = 1 condition is not a accepted instruction.

Opcode6: DESTREG ← SRCREG2 - SRCREG1

IR10 = 0, IR7=0, IR4 = 0

Opcode5T4:

OutASel = SRCREG2(0,1);
MuxCSel = 1
OutBSel = SRCREG1(0,1);
FunSel(ALU) = 0110;
MuxASel = 11;
FunSel(2A) = 01;
RegSel = DESTREG-2A-2

IR10 = 1, IR7=0, IR4 = 0

Opcode5T4:

OutASel = SRCREG2(0,1);
MuxCSel = 1
OutBSel = IR(5,6)
FunSel(ALU) = 0110;
MuxBSel = 11
FunSel(2B) = 01;
RegSel(2B) = DESTREG-2B-2

IR10 = 0, IR7=0, IR4 = 1

Opcode5T4:

OutBSel = IR(5,6)
OutCSel) SRCREG2-2B-2(1,2)
MuxASel = 10;
MuxCSel = 0;

Opcode5T5:

FunSel(ALU) = 0110
MuxASel = 11
FunSel(2A) = 01
RegSel(2A) = DESTREG-2A-2

IR10 = 1, IR7=0, IR4 = 1

Opcode5T4:

OutBSel = IR(5,6)
OutCSel) SRCREG2-2B-2(1,2)
MuxASel = 10;
MuxCSel = 0;
FunSel(ALU) = 0110
MuxBSel = 11;
FunSel(2B) = 01;
RegSel(2B) = DESTREG-2B-2

Opcode7: DESTREG ← SRCREG1 - 1

IR10 = 1, IR7 = 1

Opcode7T4:

OutCSel = SRCREG-2B-2(1,2)
MuxASel = 10

MuxCSel = 0
FunSel(ALU) = 0000
MuxBSel=11
FUNSEL(2B) = 01
REGSEL(2B) = DESTREG-2B-2

Opcode7T5:

RegSel(2B) = DESTREG-2B-2
FunSel(2B) = 11

Opcode7T6:

OutCSel = DESTREG-2B-2(1,2)
MuxASel = 10
MuxCSel = 0;
FunSel(ALU) = 0000;

IR10 = 0, IR7 = 1:

Opcode7T4:

OutCSel = SRCREG-2B-2[2][1]
MuxASel=10
FunSel(2A) = 01
RegSel(2A) = DESTREG-2A-2

Opcode7T5:

REGSEL = DESTREG-2A-2
FUNSEL(2A) = 11

Opcode7T6:

OutASel = IR(5,6)
MuxCSel = 1;
FunSel(ALU) = 0000;

IR10 = 1, IR7 = 0:

Opcode7T4:

OutBSel=SRCREG1[1][0]
FunSel(ALU) = 0001
MUXBSEL = 11
FUNSEL(2B)=01
REGSEL(2B) = DESTREG-2B-2

Opcode7T5:

RegSel(2B) = DESTREG-2B-2
FunSel(2B) = 11

Opcode7T6:

OutCSel = DESTREG-2B-2(1,2)
MuxASel = 10
MuxCSel = 0;
FunSel(ALU) = 0000;

IR10 = 0, IR7 = 0:

Opcode7T4:

OutBSel=SRCREG1[1][0]
FunSel(ALU) = 0001
MuxASel=11
FunSel(2a)=01
RegSel(2A) = DESTREG-2A-2

Opcode7T5:

RegSel(2A) = DESTREG-2A-2
FunSel(2A) = 11

Opcode7T6:

OutASel = IR(5,6)
MuxCSel = 1;
FunSel(ALU) = 0000;

Opcode8: DESTREG ← SRCREG1 + 1

IR10 = 1, IR7 = 1

Opcode8T4:

OutCSel = SRCREG-2B-2[2][1]
MuxASel = 10
MuxCSel = 0
FunSel(ALU) = 0000
MuxBSel=11
FUNSEL(2B) = 01
REGSEL(2B) = DESTREG-2B-2

Opcode8T5:

RegSel(2B) = DESTREG-2B-2
FunSel(2B) = 10

Opcode8T6:

OutCSel = DESTREG-2B-2(1,2)
MuxASel = 10
MuxCSel = 0;
FunSel(ALU) = 0000;

IR10 = 0, IR7 = 1:

Opcode8T4:

OutCSel = SRCREG-2B-2[2][1]
MuxASel=10
FunSel(2A) = 01
RegSel(2A) = DESTREG-2A-2

Opcode8T5:

REGSEL = DESTREG-2A-2

FUNSEL(2A) = 10

Opcode8T6:

OutCSel = DESTREG-2B-2(1,2)
MuxASel = 10
MuxCSel = 0;
FunSel(ALU) = 0000;

IR10 = 1, IR7 = 0:

Opcode8T4:

OutBSel=SRCREG1[1][0]
FunSel(ALU) = 0001
MUXBSEL = 11
FUNSEL(2B)=01
REGSEL(2B) = DESTREG-2B-2

Opcode8T5:

RegSel(2B) = DESTREG-2B-2
FunSel(2B) = 10

Opcode8T6:

OutCSel = DESTREG-2B-2(1,2)
MuxASel = 10
MuxCSel = 0;
FunSel(ALU) = 0000;

IR10 = 0, IR7 = 0:

Opcode8T4:

OutBSel=SRCREG1[1][0]
FunSel(ALU) = 0001
MuxASel=11
FunSel(2a)=01
RegSel(2A) = DESTREG-2A-2

Opcode8T5:

RegSel(2A) = DESTREG-2A-2
FunSel(2A) = 10

Opcode8T6:

OutASel = IR(5,6)
MuxCSel = 1;
FunSel(ALU) = 0000;

Opcode9: DESTREG ← SRCREG1 AND SRCREG2

IR4=0,IR7=0,IR10=0

Opcode9T4:

OutASel = SRCREG1(0,1);
MuxCSel = 1;
OutBSel = SRCREG2(0,1)

FunSel(ALU) = 0111;
MuxASel = 11;
FunSel(2a) = 01;
RegSel(2a) = DESTREG-2A-2

IR4 = 0, IR7=0, IR10 = 1

Opcode9T4:

OutASel = SRCREG1(0,1);
MuxCSel = 1;
OutBSel = SRCREG2(0,1)
FunSel(ALU) = 0111;
MuxBSel = 11;
FunSel(2B) = 01;
RegSel(2B) = DESTREG-2B-2

IR4 = 1, IR7=0, IR10 = 0:

Opcode9T4:

OutBSel = SRCREG1(0,1)
OutCSel = SRCREG2-2B-2(2,1)
MuxASel = 10;
MuxCSel = 0;

Opcode9T5:

FunSel(ALU) = 0111
MuxASel = 11;
FunSel(2a) = 01;
RegSel(2A) = DESTREG-2A-2

IR4 = 1, IR7=0, IR10 = 1:

Opcode9T4:

OutBSel = SRCREG1(0,1)
OutCSel = SRCREG2-2B-2(2,1)
MuxASel = 10;
MuxCSel = 0;
FunSel(ALU) = 0111
MuxBSel = 11;
FunSel(2B) = 01;
RegSel(2B) = DESTREG-2B-2

IR4 = 0, IR7=1, IR10 = 0:

Opcode9T4:

OutBSel = SrCREG2(0,1);
OutCSel = SRCREG1-2B-2(1,2)
MuxASel = 10;
MuxCSel = 0;

Opcode9T5:

FunSel(ALU) = 0111

MuxASel = 11;
FunSel(2A) = 01;
RegSel(2a) = DESTREG-2A-2;

IR4 = 0, IR7=1, IR10 = 1:

Opcode9T4:

OutBSel = SRCREG2(0,1);
OutCSel = SRCREG1-2B-2(1,2)
MuxASel = 10;
MuxCSel = 0;
FunSel(ALU) = 0111;
MuxBSel:11
FunSel(2B):01
RegSel(2B):DESTREG-2B-2

Opcode10: DESTREG ← SRCREG1 OR SRCREG2

IR4=0,IR7=0,IR10=0

Opcode10T4:

OutASel = SRCREG1(0,1);
MuxCSel = 1;
OutBSel = SRCREG2(0,1)
FunSel(ALU) = 1000;
MuxASel = 11;
FunSel(2a) = 01;
RegSel(2a) = DESTREG-2A-2

IR4 = 0, IR7=0, IR10 = 1

Opcode10T4:

OutASel = SRCREG1(0,1);
MuxCSel = 1;
OutBSel = SRCREG'(0,1)
FunSel(ALU) = 1000;
MuxBSel = 11;
FunSel(2B) = 01;
RegSel(2B) = DESTREG-2B-2

IR4 = 1, IR7=0, IR10 = 0:

Opcode10T4:

OutBSel = SRCREG1(0,1)
OutCSel = SRCREG2-2B-2(2,1)
MuxASel = 10;
MuxCSel = 0;

Opcode10T5:

FunSel(ALU) = 1000
MuxASel = 11;
FunSel(2a) = 01;

RegSel(2A) = DESTREG-2A-2

IR4 = 1, IR7=0, IR10 = 1:

Opcode10T4:

OutBSel = SRCREG1(0,1)
OutCSel = SRCREG2-2B-2(2,1)
MuxASel = 10;
MuxCSel = 0;
FunSel(ALU) = 1000
MuxBSel = 11;
FunSel(2B) = 01;
RegSel(2B) = DESTREG-2B-2

IR4 = 0, IR7=1, IR10 = 0:

Opcode10T4:

OutBSel = SrCREG2(0,1);
OutCSel = SRCREG1-2B-2(1,2)
MuxASel = 10;
MuxCSel = 0;

Opcode10T5:

FunSel(ALU) = 1000
MuxASel = 11;
FunSel(2A) = 01;
RegSel(2a) = DESTREG-2A-2;

IR4 = 0, IR7=1, IR10 = 1:

Opcode10T4:

OutBSel = SrCREG2(0,1);
OutCSel = SRCREG1-2B-2(1,2)
MuxASel = 10;
MuxCSel = 0;
FunSel(ALU) = 1000;
MuxBSel:11
FunSel(2B):01
RegSel(2B):DESTREG-2B-2

Opcode11: DESTREG ← NOT SRCREG1

IR7=0,IR10=0

Opcode11T4:

OutASel = SRCREG1(0,1);
MuxCSel = 1;
FunSel(ALU) = 0010;

Opcode11T5:

MuxASel = 11;
RegSel(2A) = DESTREG-2A-2;
FunSel(2A) = 01

IR7=0,IR10=1

Opcode11T4:

OutASel = SRCREG1(0,1);
MuxCSel = 1;
FunSel(ALU) = 0010;

Opcode11T5:

MuxBSel = 11;
RegSel(2B) = DESTREG-2B-2;
FunSel(2B) = 10

IR7=1,IR10=0

Opcode11T4:

OutCSel = SRCREG1-2B-2
MuxASel = 10;
MuxCSeş = 1;
FunSel(ALU) = 0010;

Opcode11T5:

MuxASel = 11;
FunSel(2A) = 01;
RegSel(2A) = DESTREG-2A-2

IR7=1,IR10=1

Opcode11T4:

OutCSel = SRCREG1-2B-2
MuxASel = 10;
MuxCSeş = 1;
FunSel(ALU) = 0010;

Opcode11T5:

MuxBSel= 11;
FunSel(2B) = 01;
RegSel(2B) = DESTREG-2B-2

Opcode12: DESTREG ← LSL SRCREG1

IR7=0,IR10=0

Opcode12T4:

OutASel = SRCREG1(0,1)
MuxCSel = 1;
FunSel(ALU) = 1010;

Opcode12T5:

MuxASel = 11;
FunSel(2A) = 01;
RegSel(2A) = DESTREG-2A-2

IR7=0,IR10=1

Opcode12T4:

OutASel = SRCREG1(0,1)

MuxCSel = 1;
FunSel(ALU) = 1010;

Opcode12T5:

MuxBSel= 11;
FunSel(2B) = 10;
RegSel(2B) = DESTREG-2B-2

IR7=1,IR10=0

Opcode12T4:

OutCSel = SRCREG1-2B-2;
MuxASel = 10;
MuxCSel = 1;
FunSel(ALU) = 1010;

Opcode12T5:

MuxASel = 11;
FunSel(2A) = 10;
RegSel(2A) = DESTREG-2A-2

IR7=1,IR10=1

Opcode12T4:

OutCSel = SRCREG1-2B-2;
MuxASel = 10;
MuxCSel = 1;
FunSel(ALU) = 1010;

Opcode12T5:

MuxBSel = 11;
FunSel(2B) = 10;
RegSel(2B) = DESTREG-2B-2

Opcode13: DESTREG ← LSR SRCREG1

IR7=0,IR10=0

Opcode13T4:

OutASel = SRCREG1(0,1)
MuxCSel = 1;
FunSel(ALU) = 1011;

Opcode13T5:

MuxASel = 11;
FunSel(2A) = 01;
RegSel(2A) = DESTREG-2A-2

IR7=0,IR10=1

Opcode13T4:

OutASel = SRCREG1(0,1)
MuxCSel = 1;
FunSel(ALU) = 1011;

Opcode13T5:

MuxBSel= 11;
FunSel(2B) = 10;
RegSel(2B) = DESTREG-2B-2

IR7=1,IR10=0

Opcode13T4:
MuxASel = 10;
MuxCSel = 1;
FunSel(ALU) = 1011;

Opcode13T5:
MuxASel = 11;
FunSel(2A) = 10;
RegSel(2A) = DESTREG-2A-2

IR7=1,IR10=1

Opcode13T4:
OutCSel = SRCREG1-2B-2;
MuxASel = 10;
MuxCSel = 1;
FunSel(ALU) = 1011;

Opcode13T5:
MuxBSel = 11;
FunSel(2B) = 10;
RegSel(2B) = DESTREG-2B-2

Opcode14: PC ← Value

Note: Addressing mode IM

Opcode14T4:
MuxBSel = 01;
RegSel(2B) = 001;
FunSel = 01;

Opcode15: IF Z = 1 THEN PC ← Value

Note: Addressing mode IM

Z = 1:

Opcode15T4:
MuxBSel = 01;
RegSel(2B) = 001;
FunSel(2B)= 01;
OutDSel = 00

Opcode16: IF Z = 0 THEN PC ← Value

Note: Addressing mode IM

Z = 0:

Opcode16T4:
MuxBSel = 01;

RegSel(2B) = 001;
FunSel(2B)= 01;
OutDSel = 00

Opcode17: M[SP] ← PC, SP ← SP-1, PC ← Value

Note: Addressing mode IM

Opcode17T4:

Read = 0;
Load = 1;
OutDSel = 10;
OutCSel = 00;
MuxASel = 10;
MuxCSel = 0;
FunSel(ALU) = 0000;

Opcode17T5:

RegSel(2B) = 100;
FunSel(2B) = 11;

Opcode17T6:

MuxBSel = 01;
RegSel(2B) = 01;
FunSel(2B) = 01;

Opcode18: SP ← SP+1, PC ← M[SP]

Opcode18T4:

RegSel(2B) = 100;
FunSel(2B) = 10;

Opcode18T5:

OutDSel = 10;
Read = 1;
Load = 0
MuxBSel = 10;
RegSel(2B) = 001;
FunSel(2B) = 01;

5 CONCLUSION

In conclusion, during this project we gain technical knowledge as well as some soft skills. Due to COVID-19 pandemic we were forced to work online, this was a very challenging process but We believe that our abilities to do this have improved.

In Project 3 we needed to understand the capabilities of other components more to design the control unit. In other words, we understood Project1 and Project 2 more by implementing Project 3.

We needed to break down the instruction to clock cycles and determine which inputs were necessary to the system, this proved to be challenging because we didn't have the grasp on how a hardware system works, with trial and error we overcomed this issue. This was a detailed project because of this we made attention errors but after debugging hopefully we overcame all of them.