ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT REPORT

PROJECT NO	:	1
DUE DATE	:	19.05.2021
GROUP NO	:	G18

GROUP MEMBERS:

150180027	:	Talha Sezer Çakır
150180049	:	Mehmet Yiğit Ateş
150190725	:	Mehmed Esad Akçam

1 INTRODUCTION

In this project 16-bit Instruction Register(IR), Register File, Address Register File(ARF), Program Counter(PC), Address Register(AR), Stack Pointer (SP) and Arithmetic Logic Unit (ALU) are designed. With using all elements designed were used to generate basic computer design.

2 PROJECT PARTS

2.1 PART 1

In this part, two different type of registers are designed.

The 8-bit register has increment,decrement,clear and load functionalities that are controlled by 2-bit control signals (FunSel) and an enable input (E) is implemented as shown in the Figure 1.

The 16-bit register has increment,decrement, clear, load and load to most significant 8 bit or least significant 8 bit functionalities that are controlled by 2-bit control signals (FunSel), an enable input (E) and L/H signals is implemented as shown in the Figure 2.

2.1.1 PART 1A

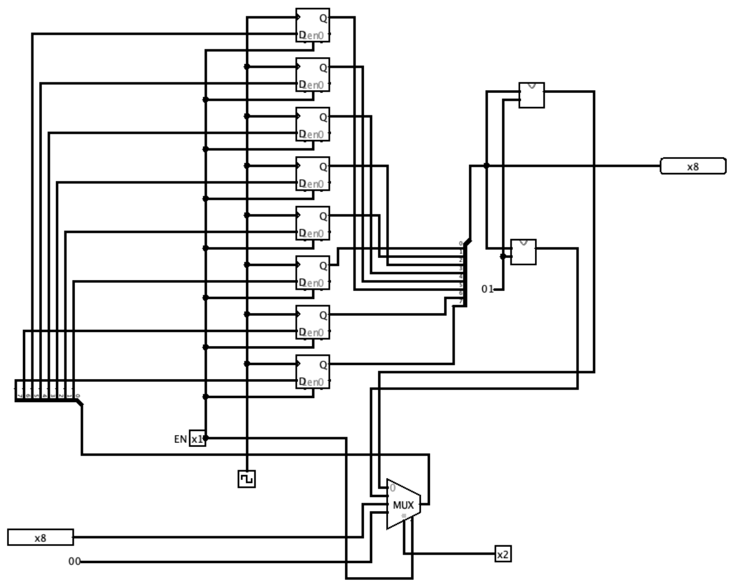

Figure 1: Logisim Circuit For Part-1a

As shown in Figure 1, there are following components in our design:

- Multiplexer: We use multiplexer in order to determine which functionalities should get process and keep it in flip-flop.

- Adder/Subtractor: We use it for increment and decrement operations.
- Flip-Flop: We use it for keeping input to increment or decrement operation after load or clear operation.

2.1.2 PART 1B

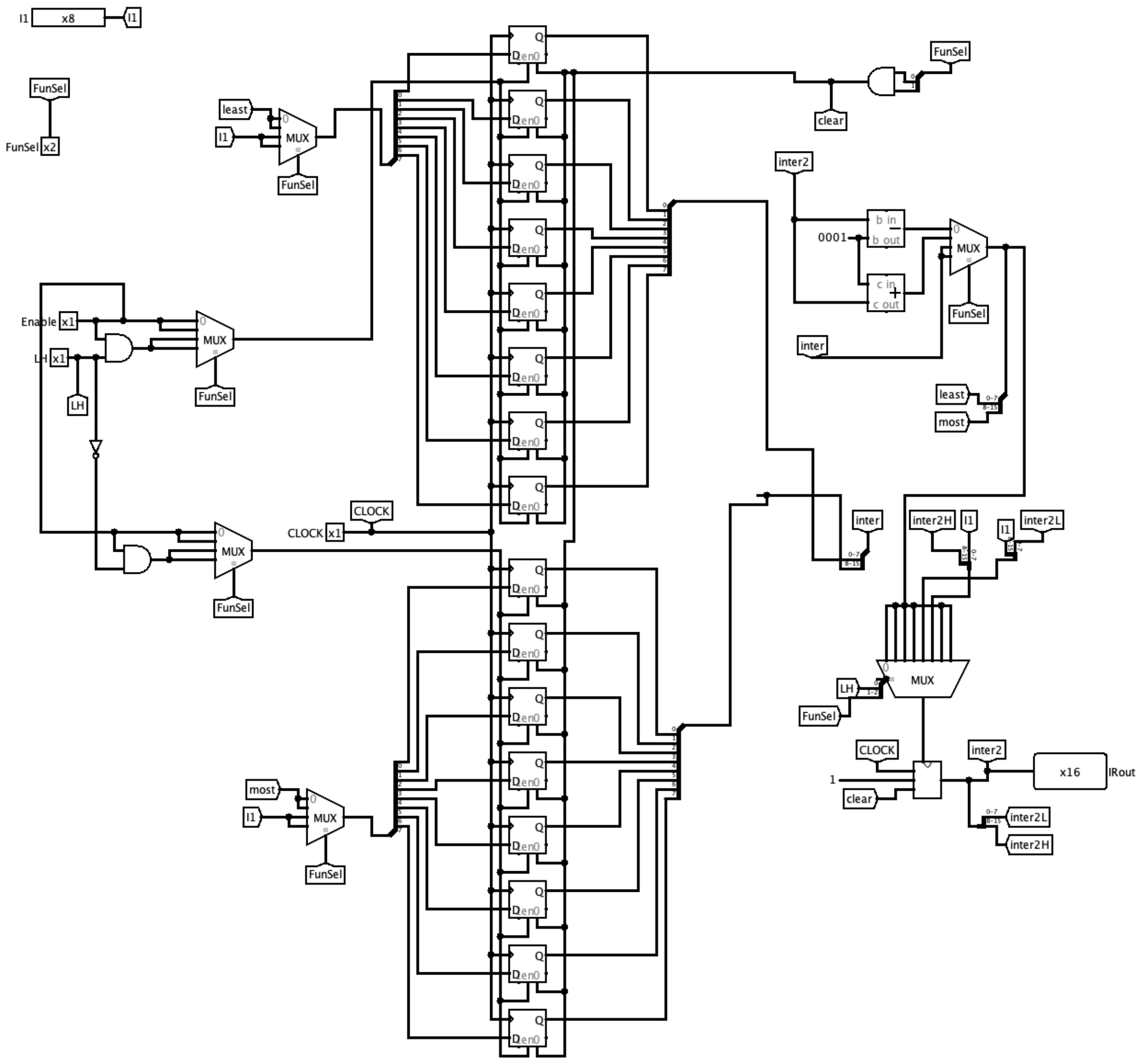

Figure 2: Logisim Circuit For Part-1b

- Multiplexer: We used mux to decide which 8 bits to load. We used it to load the register that we use as the second when loading.
- Adder/Subtractor: We use it for increment and decrement operations.
- Flip-Flop: We used it to store the current value of IR.

2.2 PART 2

In this part, we implemented organization of registers such as Register File and Address Register File.

Register File consists of 4 general purpose registers and 4 temporary registers as shown in Figure 3. It has an 8-bit input (I) to load into active registers, 2-bit FunSel input to select operations such as load, increment, decrement and clear, 4-bit RegSel and TmpSel inputs to select which registers will be enabled and disabled, 3-bit OutASel and OutBSel inputs to choose which register's value will be given as outputs.

Address Register File consists of 3 registers such as PC, AR, SR as shown in Figure 6. It has an 8-bit input (I) to load into active registers, 2-bit FunSel input to select operations such as load, increment, decrement and clear, 3-bit RegSel input to select which registers will be enabled and disabled, 2-bit OutCSel and OutDSel inputs to choose which register's value will be given as outputs.

2.2.1 PART 2A

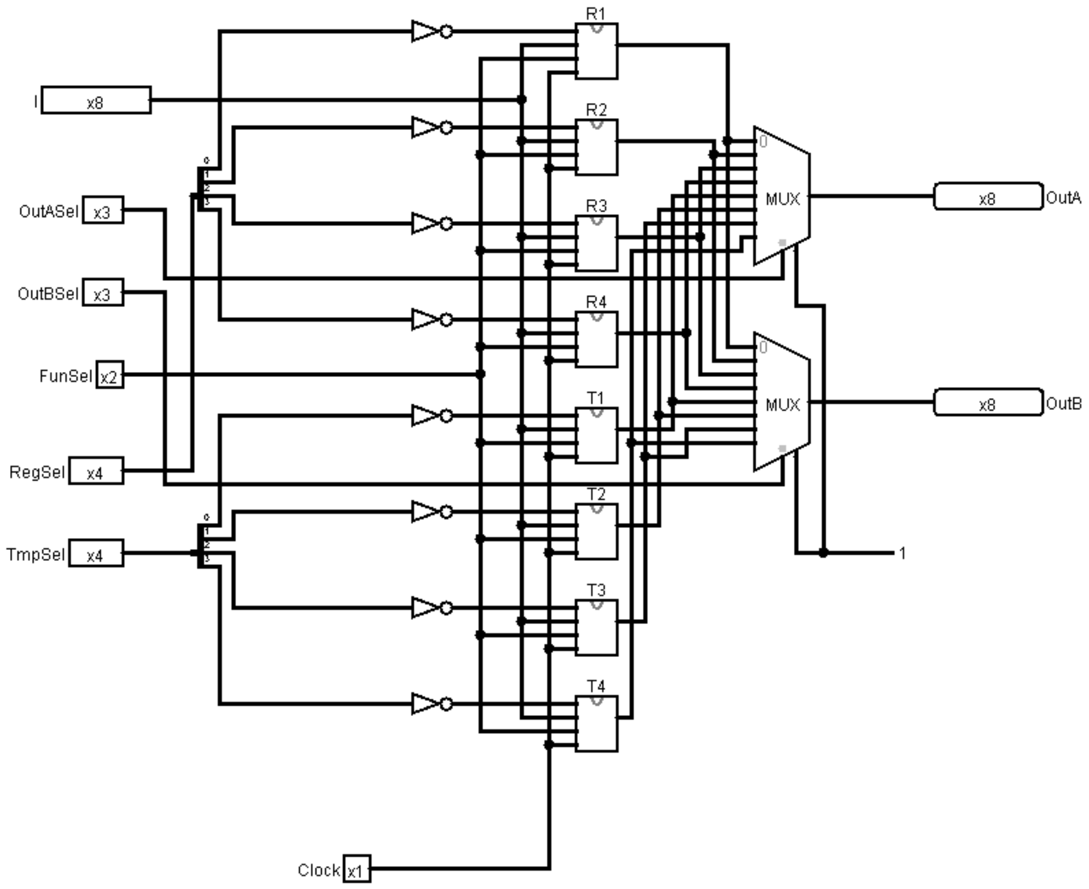

Figure 3: Logisim Circuit of Register File

As shown in Figure 3, there are following components in our design:

- Multiplexer: We use this to select which register to give as output A and output B. Multiplexers choose one of 8 registers according to given OutASel and OutBSel 3-bit inputs.
- 8-bit Register: We use registers to store values and give them as outputs when we want.

2.2.2 PART 2B

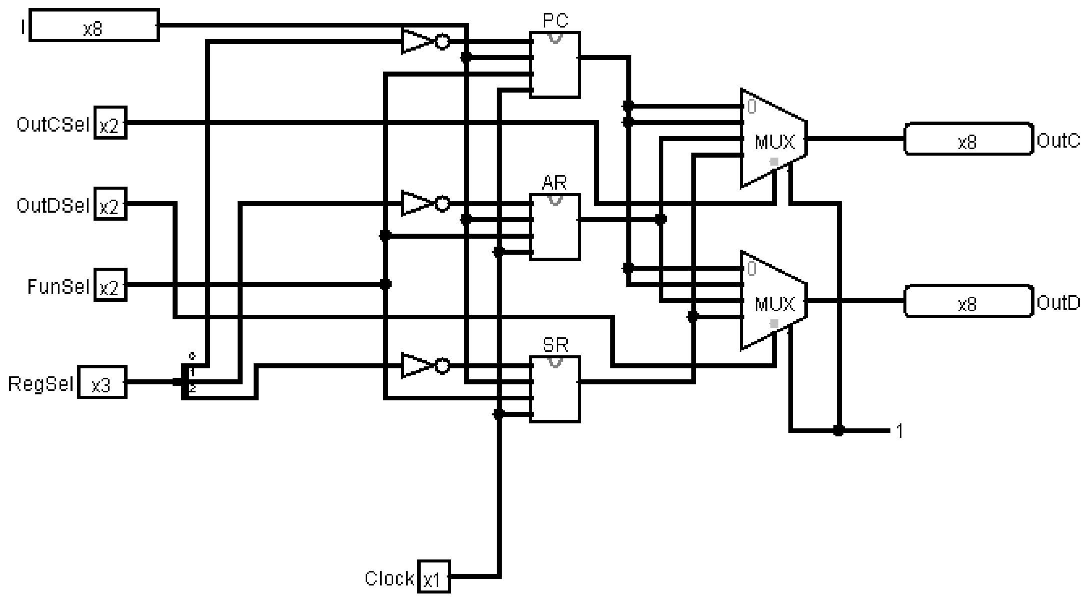

Figure 4: Logisim Circuit of Address Register File

As shown in Figure 6, there are following components in our design:

- Multiplexer: We use this to select which register to give as output C and output D. Multiplexers choose one of 3 registers according to given OutCSel and OutDSel 2-bit inputs.
- 8-bit Register: We use registers to store values and give them as outputs when we want.

2.3 PART 3

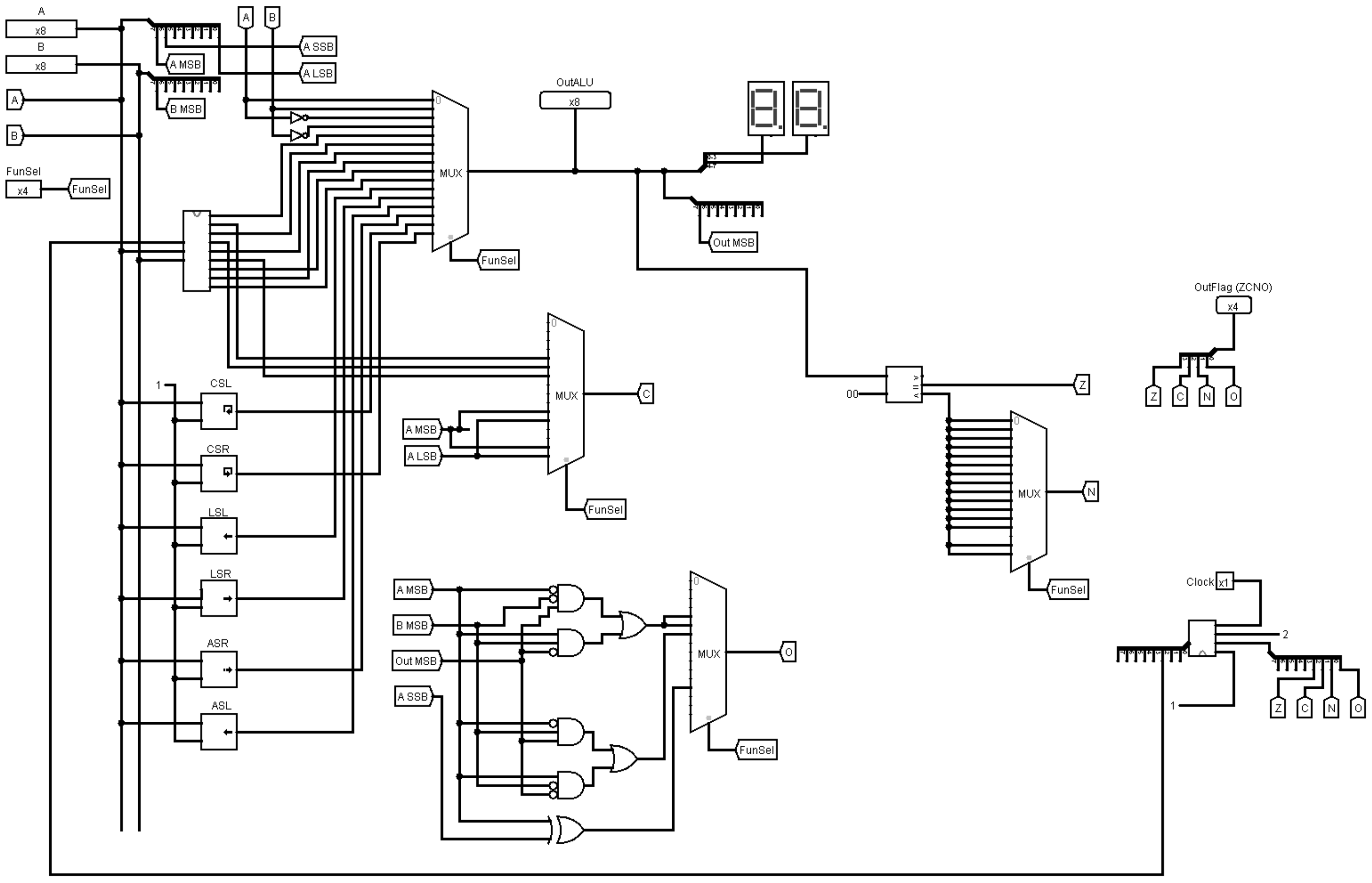

Figure 5: Logisim Circuit Of ALU

In this part we implemented an Arithmetic Logic Unit which handles given arithmetic and logic operations in project description. We used:

- Multiplexers: To choose which operations lead to carry and overflow, which operations' results are negative or zero. Lastly, obviously, to choose which operations are going to be handled according to FunSel input.

- Adder Unit: In order to complete addition and subtraction operations, we used Adder module in Logisim. We handled subtraction by 2's complement method with one Adder.

- Shifter Modules: In order to handle shift operations we used Logisim Shifter modules and we set shift bit attribute of every shifter to 0x7.

- Register Module: We used a 8 bit register module that is implemented by us in earlier chapter of project. In this register we store ZCNO flags.

- Operations Module: For simplicity we implemented a module that contains arithmetic and logic operations and this module is added to ALU module as a block.

2.4 PART 4

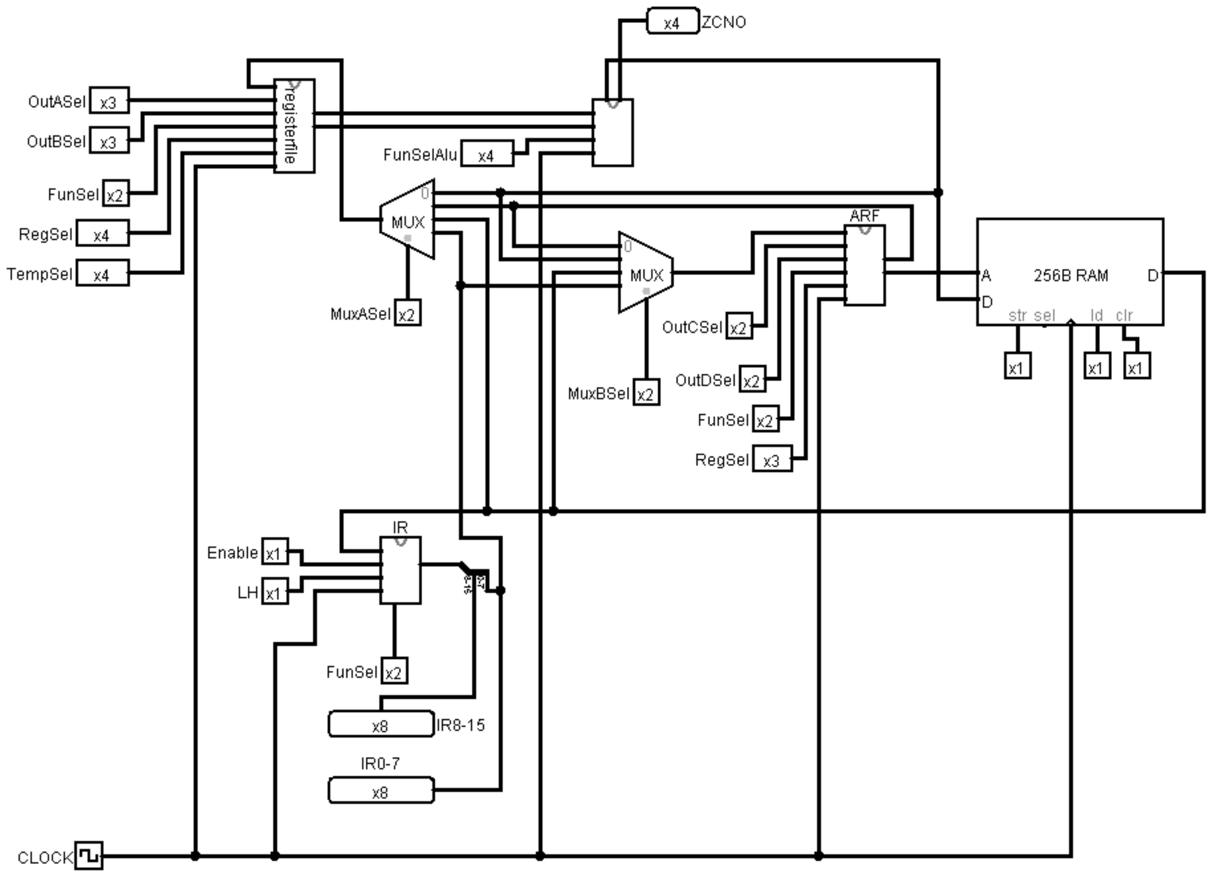

Figure 6: Organisation

In this part we implemented given organisation in the prject description. We used:

- Register File: We used it to store temporary 8-bit values in 8 registers.
- Address Register File: We used it to realize and organize Address Register components such as PC, AR, SP.
- ALU: We used ALU in order to handle arithmetic and logic operations.
- IR: We used IR in order to store the Instructions.
- RAM: We used RAM in order to store various data which will be fed into other modules.
- Multiplexer: In order to choose which data will be input of ARF and Register File.

3 RESULTS

3.1 Part 1

3.1.1 PART 1A

Test inputs and outputs for 8- Bit Register circuit. Each operation perform with clock cycle.

A	FunSel	Op	Output
00011111	10	Load	00011111
00011111	11	Clear	00000000
00000000	01	Increment	00000001
00000001	01	Increment	00000010
00000010	00	Decrement	00000001
00000001	11	Clear	00000000
00000000	00	Decrement	11111111
11111111	01	Increment	00000000

Each step perform with clock cycle.
1) Firstly we load input 00011111 to output.
2) Previous result 00011111 is cleared with next clock cycle.
3) After clearing, 00000000 result increments one by one.
4) Other operations similiar with previous operation above first three step.

3.1.2 PART 1B

Test inputs and outputs for 8- Bit Register circuit. Each operation perform with clock cycle.

A	FunSel	L/H	Op	Output(MSB-LSB)
00011111	10	1	Load LSB	00000000-00011111
00011110	10	0	Load MSB	00011110-00011111
00011110	11	X	Clear	00000000-00000000
00011110	01	X	Increment	00000000-00000001
00000001	10	0	Load MSB	00000001-00000001
00000001	00	X	Decrement	00000001-00000000
00000001	00	X	Decrement	00000000-11111111
00000001	01	X	Increment	00000001-00000000

Each step performs with clock cycle. And all steps result data is saved in flip-flops.
1) Firstly, we try to load input to least significant 8bit of output.
2) After loading LSB, input 00011110 input is loaded to most significant 8bit of output.
3) We clear our 16 bit data.
4) We increment new cleared data.
5) We have loaded first 8 bit data and we have 16 bit new data.
6) After loading MSB, decrementation operation is applied to new data.It decrements whole number 1.
7) After decrementation, decrementation and incrementation operation are applied to data. It decrements 1 and increments 1.

As shown in the test table, our implementation approaches to data as 16 bit number when operarion is clear, increment and decrement. In Load MSB and Load LSB operation, our implementation logic assign input to MSB or LSB accoring to L/H signal.

3.2 Part 2

We tested our register file and address register file giving convenient inputs in a well-planned way to realize functionalities such as load, increment, decrement on different registers. We tracked inputs and outputs and validated that our outputs give expected values.

3.2.1 Part 2A

Test inputs and outputs for Register File circuit.

I	OutASel	OutBSel	FunSel	RegSel	TmpSel	OutA	OutB
00000011	000	100	10	0000	1111	00000011	00000000
00000110	000	100	10	1111	0000	00000011	00000110
11111111	000	100	01	0111	1111	00000100	00000110
11111111	100	101	00	1111	0111	00000101	00000110
11111111	000	100	11	1111	0000	00000100	00000000
11111111	100	101	10	1111	0000	11111111	11111111

We wrote inputs we give and functionalities that we do in each clock cycle.
1) At the first cycle, we loaded 8-bit input value ”00000011” into every general purpose register. We selected OutB as T1 to show that it is not set to any value and it is initially ”00000000”.
2) We loaded 8-bit input value ”00000110” into every temporary register.
3) We incremented general purpose register 1 by 1. 8-bit input value is not important as we do not use load function.
4) We decremented temporary register 1 by 1. We gave T1 and T2 as outputs to show that only T1 is decremented.
5) We cleared all temporary registers. We gave one of temporary registers to show that they are set to 0.
6) We loaded a different value ”11111111” to all temporary registers.

3.2.2 Part 2B

Test inputs and outputs for Address Register File circuit.

I	OutCSel	OutDSel	FunSel	RegSel	OutC	OutD
00000011	00	11	10	001	00000011	00000000
00000000	00	10	01	011	00000100	00000011
00000000	11	00	00	110	11111111	00000100
11111111	11	01	11	000	00000000	00000000

We wrote inputs we give and functionalities that we do in each clock cycle.
1) At the first clock cycle, we loaded 8-bit value ”00000011” into PC and AR.
2) We incremented PC by 1.

3) We decremented SR by 1. As we can see, value in SR which is "00000000" is decremented to "11111111".
4) Finally, we cleared every register.

3.3 Part 3

Test inputs and outputs for ALU:

A	B	FunSel	Op	OutALU	ZCNO
00011111	00100000	0000	A	00011111	0x0x
00011111	00100000	0001	B	00100000	0x0x
00011111	00100000	0010	NOT A	11100000	0x1x
00011111	00100000	0011	NOT B	11011111	0x1x
1000000	10000000	0100	A+B	00000000	1101
0000000	00000000	0101	A+B+Carry	00000001	0000
0000000	00000001	0110	A-B	11111111	0010
00110011	11001100	0111	A & B	00000000	1x0x
00110011	11001100	1000	A | B	11111111	0x1x
00110010	11001100	1001	A $\bigoplus$ B	11111110	0x1x
1100001	11001100	1010	LSL A	10000010	011x
1100001	11001100	1011	LSR A	01100000	010x
1100001	11001100	1100	ASL A	10000010	0x10
1100001	11001100	1101	ASR A	11100000	0xxx
1100010	11001100	1110	CSL A	10000101	011x
1100010	11001100	1111	CSR A	01100001	000x

Note: While calculating A + B + Carry in the table; since carry value (form previous operation) in the register is equal to 1, result is 00000001.

3.4 Part 4

In this part since there are a lot of input we couldn't test comprehensively. We tested in three situations:

- Test Case 1: LSR(R1-1) is loaded into R2
  1) In the Register file, only R1 is enabled and 1 decreased from initial value. So Register File output is FF and it is fed to A out of the Register file.
  Inputs: RegSel = 0111, TempSel = 1111, FunSel = 00, OutASel = 000.
  2) FunSel of ALU is 1110. So the operation is LSR. OutAlu 7F ZCNO is 010x.
  3) MuxASel is 00 (OutALU) so input of Register file is the output of ALU which is 7F.
  4) RegSel is 1011 so load 7F to R2.
  5) OutBSel is 001 which means it will give output B as R2 and it will be 7F.

- Test Case 2: R1+R1 is loaded into AR and R3.
  1) Only R1 Enabled OutASel and OutBSel are 000 so OutA and OutB is equal to FE.
  2) FunSel of ALU is 0100 so it OutALU is A + B = FC (carry happened). Since both numbers are negative and result is negative there is no overflow but there is carry and the number is not zero. ZCNO = 0110.
  3) MuxBSel is equal to 01 so the output of MuxB is FC.
  4) FunSel of ARF is 10 and RegSel 101 so FC will be loaded into AR.
  5) OutCSel is 10 so the value in the AR which is FC will be fed into OutC of AR.
  6) MuxASel is 01 so OutC of AR which is FC is input of Register file.
  7) FunSel of Register file is 10 and RegSel is 1101 so FC is copied into R3.

- Test Case 3: Decremented IR and Value from RAM is loaded into IR
  1) IR is enabled and FunSel of IR is equal to 00. With one clock cycle IR out will be FFFF.
  2) L/H' is equal to 1 and FunSel is equal to 10. So the value from RAM will be loaded to IR[0:7].
  3) Load input of RAM is enabled. Since all initial values in RAM which is 0 will written into IR[0:7].

4 DISCUSSION

- Registers can have different functionalities, they can perform different tasks according to these functionalists.
- Microprocesses occur synchronously with the clock cycle. The duration of the clock cycle on a simple computer should be adjusted considering the operations that will occur in one clock cycle.
- Instead of register handling arithmetic, logic and shift microoperations,it is more efficient to creating a unit for this operations which is ALU.
- To write or read memory data, an address is needed and there is a specific register for this which is AR.
- There are Temporary registers for holding intermediate values.
- There is an Instruction Register to keep transactions to take place.

5 CONCLUSION

- It was so difficult for us to handle performing operation inside different flip-flops with same clock cycle at 16 bit IR register.
- We were enabling and disabling wrong registers in Register File and ARF. But we realized this and reversed splitters in selection of registers to solve this situation.
- Although all other slight difficulties we have learnt to build basic computer structure with various registers, RAM and ALU.

REFERENCES

[1] F. Buzluca, Lecture Notes Chapter 6, https://ninova.itu.edu.tr/tr/dersler/bilgisayar-bilisim-fakultesi/1591/blg-231e/ekkaynaklar?g167765 Last Visit: 17 May 2021.