ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION
PROJECT REPORT

PROJECT NO : 1

PROJECT DATE : 28.02.2020 - 12.03.2020

GROUP NO : 29

GROUP MEMBERS:

150160802 : MEHMET CAN GÜN

150170054 : ZAFER YILDIZ

150180012 : MUHAMMED SALİH YILDIZ

150180080 : BAŞAR DEMİR

SPRING 2020

Contents

1 INTRODUCTION

In this project, we are expected to design registers that have different sizes and register files. Registers are groups of flip-flops, where each flip flop is capable of one bit of information. The basic function of a register is to hold information in a digital system and make it available to the logic elements for the computing process. While we were implementing registers, we used some built-in components that have already designed in Logisim program and we also designed our own components in implementing process of registers and register files.

2 MATERIALS

In this project we used following items:

- 8-bit Parallel Adder
- 8-bit Parallel Subtractor
- 16-bit Parallel Adder
- 16-bit Parallel Subtractor
- 8-bit register
- 16-bit register
- D Type Flip-Flop
- 4:1 Multiplexer
- Splitter
- Hex Digit Display

2.1 PART 1

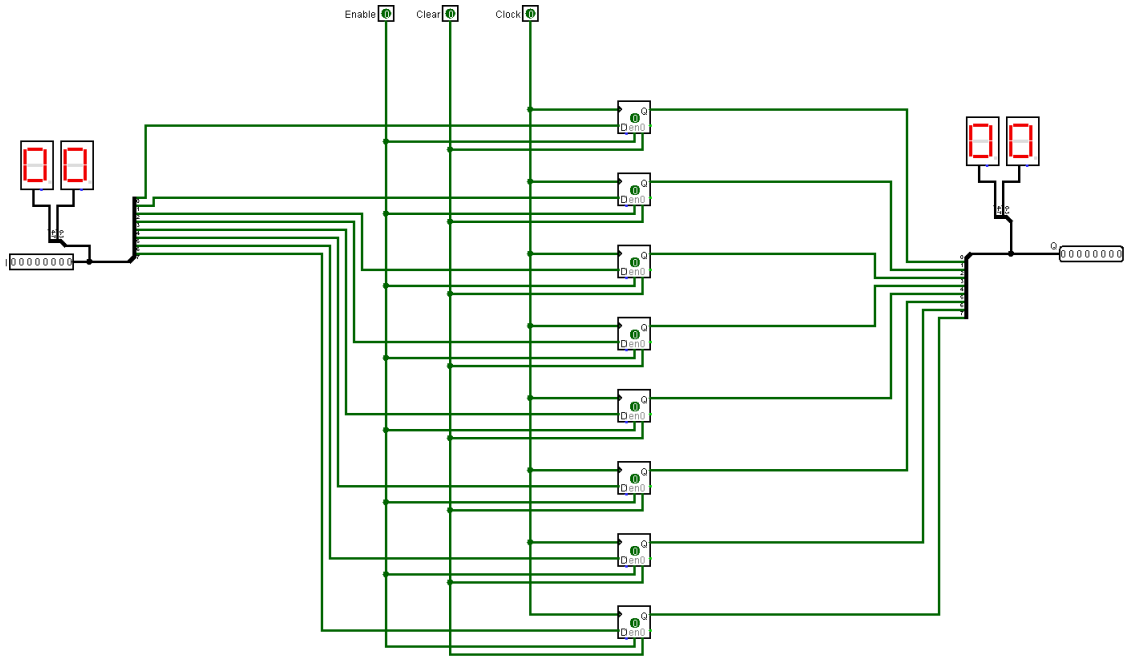


Figure 1: 8-Bit Register

Registers occur inputs, outputs, enable state, clear state, flip-flop for synchronizing data and bit-size to store data. The preferred flip-flop is D flip-flop shortly what the get data from inputs and give them as output without any operation in the clock cycle. In the implementation and design of the 8-bit register circuit in Figure 1, used 8-bit input pins are distributed by splitter for each corresponding the D flip-flops as an input to D-pin. The output of D flip-flop is connected to 8-bit output pins. The clear and enable states of D flip-flops are pinned corresponding inputs called as 0 for clear and EN for enable. Hex Digit Display, also, is connected inputs and outputs to make more human-readable.

Testing the registers, inputs are given to circuit and make sure that enable status is 1. In one clock cycle is determined with Clock input as 1 for transferring and 0 for holding the data. When the clock input is in rising edge, the data will be transferred to Q. The clear state does not depend on the clock or enable because of properties of D flip-flop. In the end, the register is packed to use at any time.

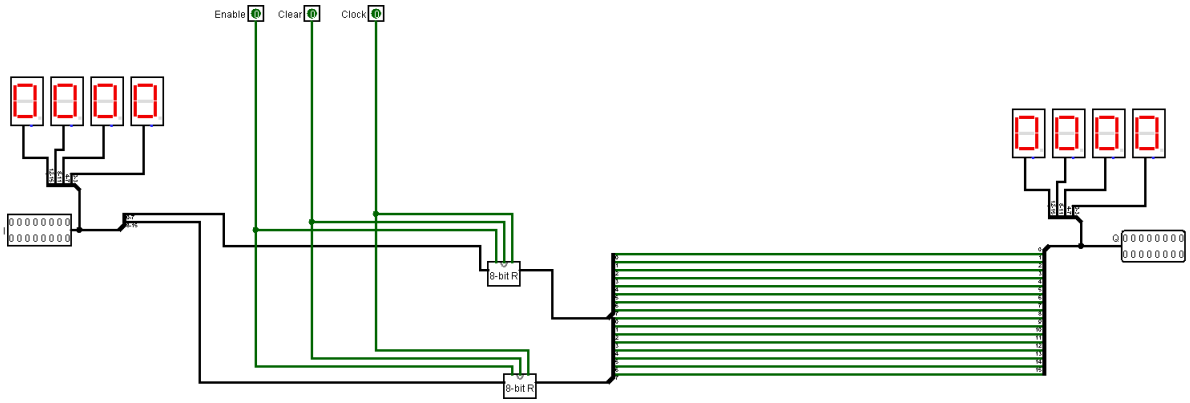
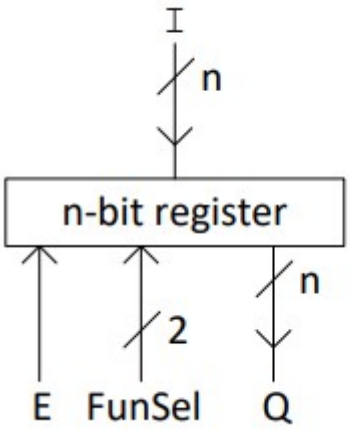


Figure 2: 16-Bit Register

In the 16-registers as can be shown in Figure 2, two packages of created 8-bit registers are used. The connection between the two registers is provided with the splitter. The input data between 0-7 bits and 8-15 bits are separated with splitter uniformly to registers. These registers' outputs are connected to 16-bit output pins by using the splitter. The enable and clear inputs are pinned as the same inputs in the registers. Hex Digit Display, also, is connected inputs and outputs to make more human-readable. Testing process of 16-register is same as 8-bit register.



E	FunSel	Q^+	
0	ϕ	Q	(Retain Value)
1	00	0	(Clear)
1	01	I	(Load)
1	10	Q+1	(Increment)
1	11	Q-1	(Decrement)

Figure 3: Graphic symbol of the registers (Left) and the characteristic table (Right)

After designing pure 8-bit and 16-bit registers, the inputs, outputs, enable and FunSel properties will be easily added. Firstly, the inputs are connected to the 4:1 Multiplexer's load signal. To make more functional, the 2-bit control signals(FunSel) is connected to the circuit as 4:1 Multiplexer with Enable input. The multiplexer's output is connected to the 8-bit register. The multiplexer's signals are designed according to the characteristics of FunSel as can be seen in Figure 3.

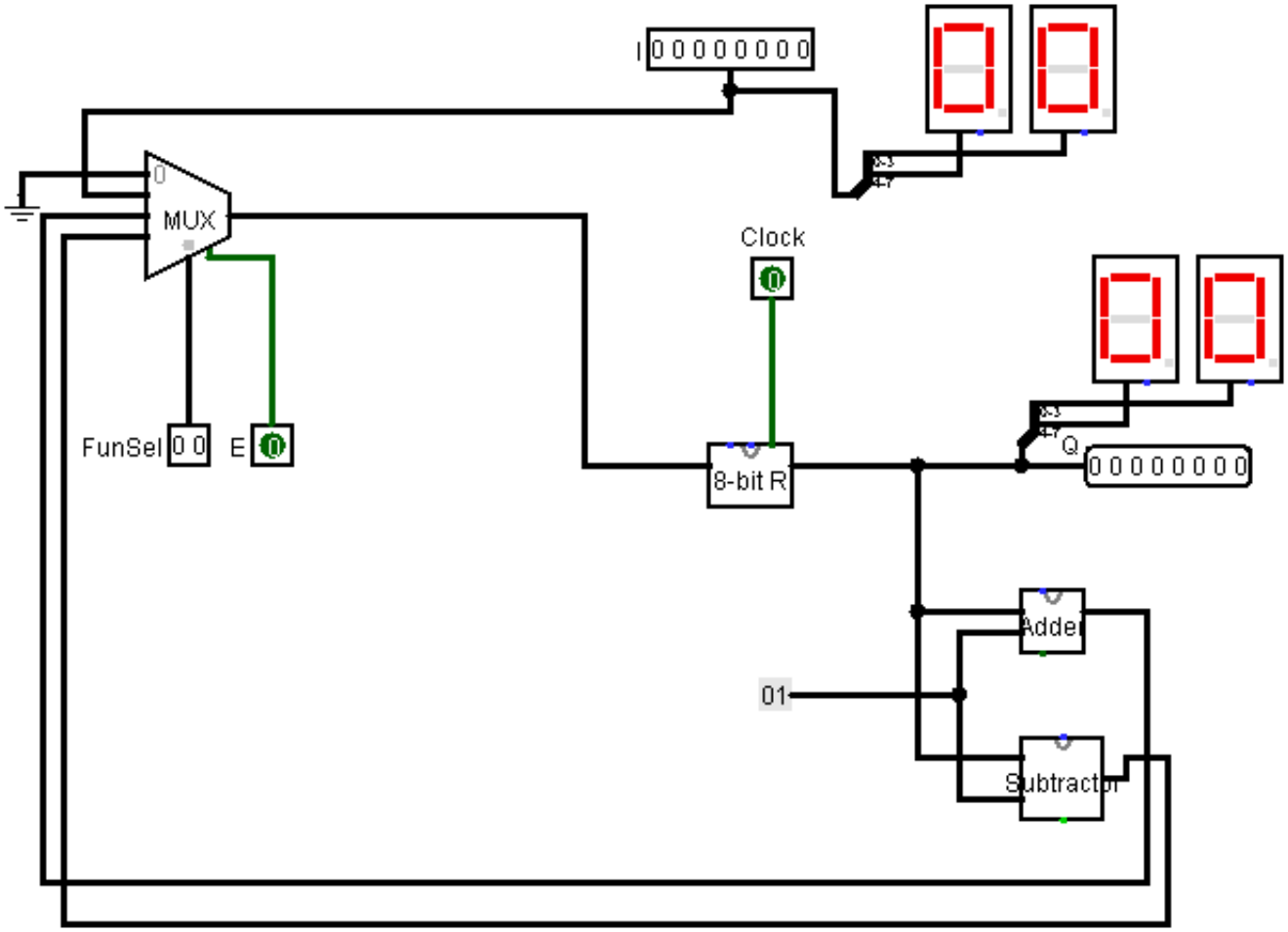


Figure 4: The circuit with Inputs, outputs, enable input and FunSel

While designing the circuit, 0 used as clear with the multiplexers' signal because the clear input should work with the clock cycle. As an enable input, also, the multiplexer used to change the status of enabling or disabling. The circuit must be stable so after the addition and subtraction operations, the next state is given to multiplexers' addition and subtraction signal correspondingly. The addition and subtraction operations have 2 inputs, one is the given from user and second is consist of 8-bit constant for 1 as 00000001. The output of these operations is given to multiplexer as keeping the circuit stable in one clock cycle. To test the circuit, multiplexer's enable status need to be high(1) and to see changes in circuit, clock cycle transfers data in rising edge(0 to 1). Hex Digit Display, also, is connected inputs and outputs to make more human-readable while testing the circuit. In the end, the circuit in Figure 4 is packaged to use as a combinational circuit at any time. The packaged circuit is used in Figure 5 below.

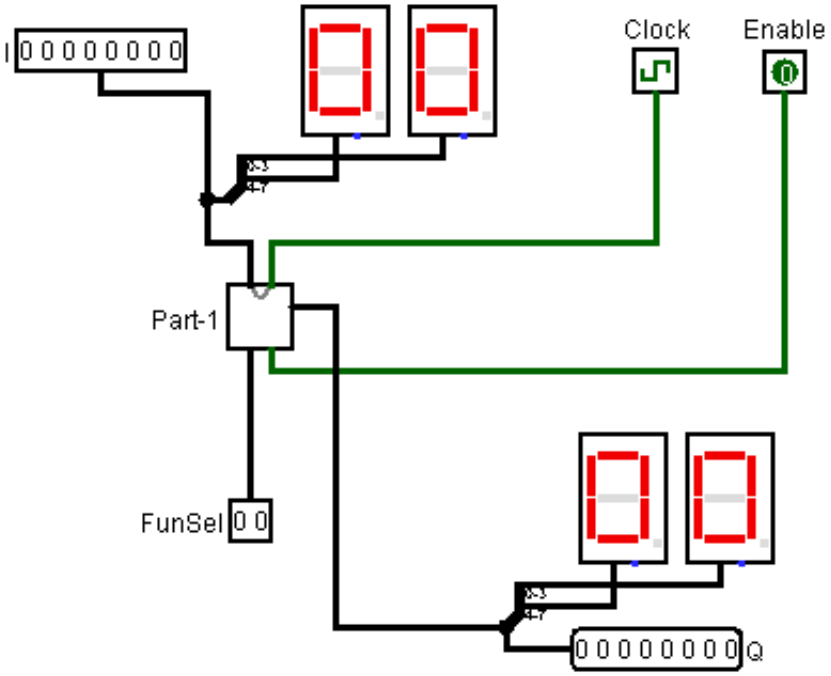


Figure 5: The simulation of circuit with clock

E	I	FunSel	Q	Q+
1	4C	01	Ø	4C
1	Ø	11	4C	4B
1	Ø	10	4B	4C
1	Ø	00	4C	00
1	64	01	00	64
0	Ø	Ø	64	64

Figure 6: The test cases of the circuit

The circuit is validated with some test cases shown in Figure 6. The test cases works step by step that are listed below.

- Load $->$ 4C
- Decrement by 1
- Increment by 1
- Clear
- Load $->$ 64
- Enable $->$ OFF

2.2 PART 2

2.2.1 Part 2-a

RegSel	Enabled Registers
0000	N0 register is enabled, All registers retain their values
0001	Only R0 is enabled, Function selected by FunSel will be applied to R0
0010	Only R1 is enabled, Function selected by FunSel will be applied to R1
0011	R0 and R1 are enabled, Function selected by FunSel will be applied to R0 and R1
0100	Only R2 is enabled, Function selected by FunSel will be applied to R2
0101	R0 and R2 are enabled, Function selected by FunSel will be applied to R0 and R2
0110	R1 and R2 are enabled, Function selected by FunSel will be applied to R1 and R2
0111	R0, R1 and R2 are enabled, Function selected by FunSel will be applied to R0, R1 and R2
1000	Only R3 is enabled, Function selected by FunSel will be applied to R3
1001	R0 and R3 are enabled, Function selected by FunSel will be applied to R0 and R3
1010	R1 and R3 are enabled, Function selected by FunSel will be applied to R1 and R3
1011	R0, R1 and R3 are enabled, Function selected by FunSel will be applied to R0, R1 and R3
1100	R2 and R3 are enabled, Function selected by FunSel will be applied to R2 and R3
1101	R0, R2 and R3 are enabled, Function selected by FunSel will be applied to R0, R2 and R3
1110	R1, R2 and R3 are enabled, Function selected by FunSel will be applied to R1, R2 and R3
1111	R0, R1, R2 and R3 are enabled, Function selected by FunSel will be applied to R0, R1, R2 and R3

Figure 7: The characteristics of the RegSel

FunSel	R_x^+	
00	0	(Clear)
01	I	(Load)
10	R_x+1	(Increment)
11	R_x-1	(Decrement)

Figure 8: The truth table of the FunSel

In this part, we connected 8-bit input to register file which consist of 4 created registers as R0, R1, R2 and R3. These registers are correspondingly connected with RegSel as shown in Figure 7 by using splitter in order to make a selection between registers. The FunSel as shown in Figure 8, also, is connected to these registers in order choose function like clear, load, increment and decrement. The clock is connected to these registers respectively so as to change outputs with connected functions.

OutASel	Output A
00	R0
01	R1
10	R2
11	R3

OutBSel	Output B
00	R0
01	R1
10	R2
11	R3

Figure 9: The truth table of the OutASel and OutBSel

The OutASel and OutBSel are 2 selectors as can be seen in Figure 9. The OutASel is placed to select one of the 4-bit registers connected 4:1 multiplexer's input to show in 4-bit 4:1 multiplexer's output signal as OutA. The OutBSel is placed to select one of the 4-bit registers connected 4:1 multiplexer's input to show in 4-bit 4:1 multiplexer's output signal as OutB. Hex Digit Display, also, is connected inputs and outputs to make more human-readable. In the end, our implemented circuit is given in Figure 10 below.

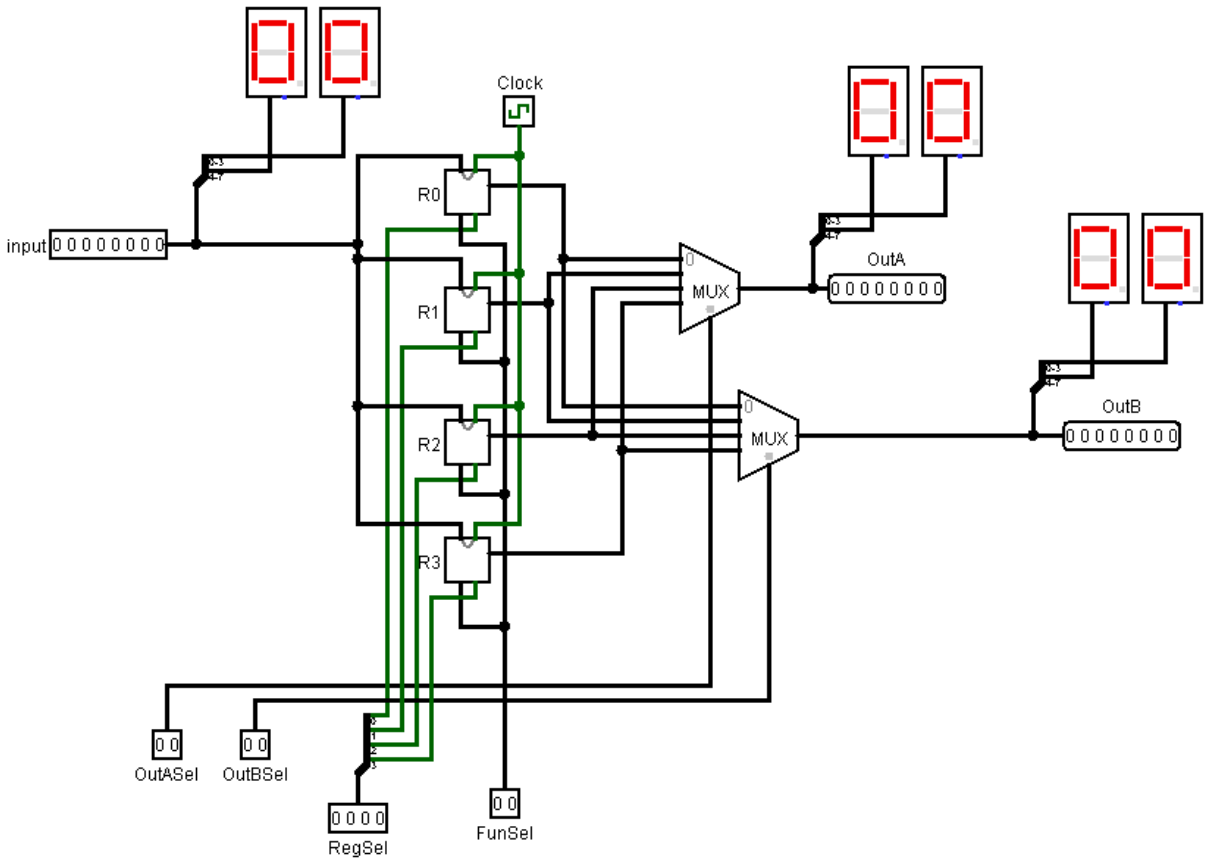


Figure 10: The circuit including register file and 4 functionalities

I	FunSel	RegSel	OutASel	OutBSel	OutA	OutB
24	01	0111	00	11	24	00
Ø	11	0101	01	11	24	00
Ø	10	1111	01	10	25	24
Ø	00	1000	00	11	24	00
60	01	1010	01	10	60	24
Ø	10	1111	01	10	61	25

Figure 11: The test cases of the circuit

The circuit is validated with some test cases shown in Figure 11. The test cases works step by step that are listed below.

- Load $->$ 24 to R0, R1, R2
- Decrement by 1 - RegSel in R0 and R2 but OutA-R1 and OutB-R3, changes not seen in output table
- Increment by 1
- Clear
- Load $->$ 60
- Increment

2.2.2 Part 2-b

In this part, we connected 8-bit input to register file which consist of 3 created registers as PC, AR and SP. These registers are respectively connected with RegSel as refered in Part 2-a by using splitter in order to make a selection between registers. The FunSel does the same operations in Part 2-a. The clock is connected to these registers correspondingly in order to make changes in outputs with connected functions.

OutCSel	Output C
00	PC
01	AR
10	SP
11	PC

OutDSel	Output D
00	PC
01	AR
10	SP
11	PC

Figure 12: The truth table of OutCSel and OutBSel.

The OutCSel and OutDSel are 2 selectors as can be seen in Figure 12. The OutCSel is placed to select one of the 4-bit registers connected 4:1 multiplexer's input to show in 4-bit 4:1 multiplexer's output signal as OutC. The OutDSel is placed to select one of the 4-bit registers connected 4:1 multiplexer's input to show in 4-bit 4:1 multiplexer's output signal as OutD. Hex Digit Display, also, is connected inputs and outputs to make more understandable. In the end, our implemented circuit is given in Figure 13 below.

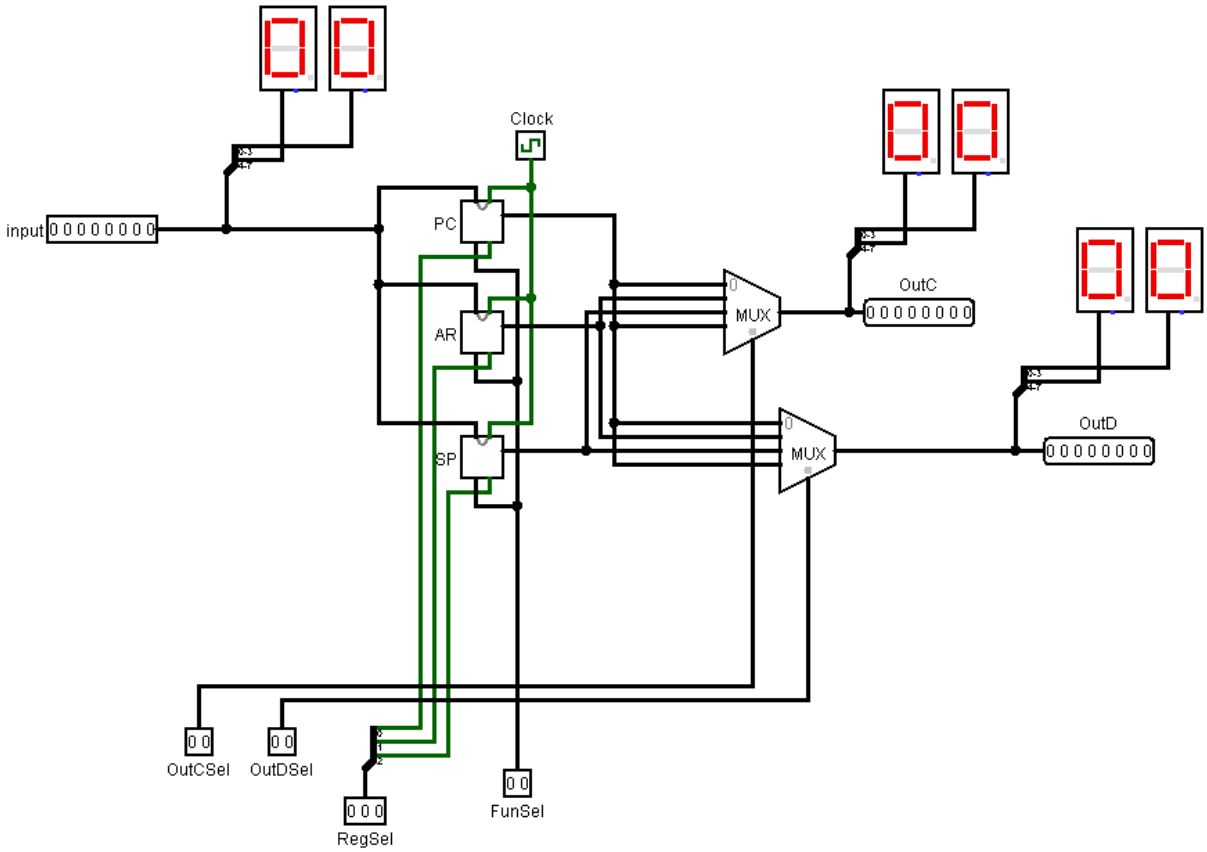


Figure 13: The circuit including register file(3 registers) and 4 functionalities

I	FunSel	RegSel	OutCSel	OutDSel	OutC	OutD
24	01	111	00	11	24	24
Ø	11	101	01	11	24	23
Ø	10	110	01	10	25	24
Ø	00	000	00	11	23	23
60	01	010	01	10	60	24
Ø	10	101	01	10	60	25

Figure 14: The test cases of the circuit

The circuit is validated with some test cases shown in Figure 14. The test cases works step by step that are listed below.

- Load $->$ 24 to PC, AR, SP
- Decrement by 1 - RegSel in PC and SP
- Increment by 1
- Clear
- Load $->$ 60
- Increment

2.2.3 Part 2-c

$\overline{L}$/H	Enable	FunSel	IR^+
ϕ	0	ϕϕ	IR
ϕ	1	00	0
ϕ	1	01	IR + 1
ϕ	1	10	IR - 1
0	1	11	IR(0-7) <- I
1	1	11	IR(8-15) <- I

Figure 15: The characteristic table of the IR Register

In this part, we have one fundamental multiplexer which is controlled by FunSel. This multiplexer have 4 entries which carry results of the choices to next the step. First input of multiplexer is 16-bit zero which is used for clear operation. Second and third ones are used for incrementation by one and decrementation by one in order. Fourth input is connected to a multiplexer which makes selection between registers to load with 8-bit input to register which decided by Low/High as shown in Figure 15. 16-bit output observed after concatenating outputs of 2 8-bit registers. Hex Digit Display, also, is connected inputs and outputs to make more understandable. In the end, our implemented circuit is given in Figure 16 below.

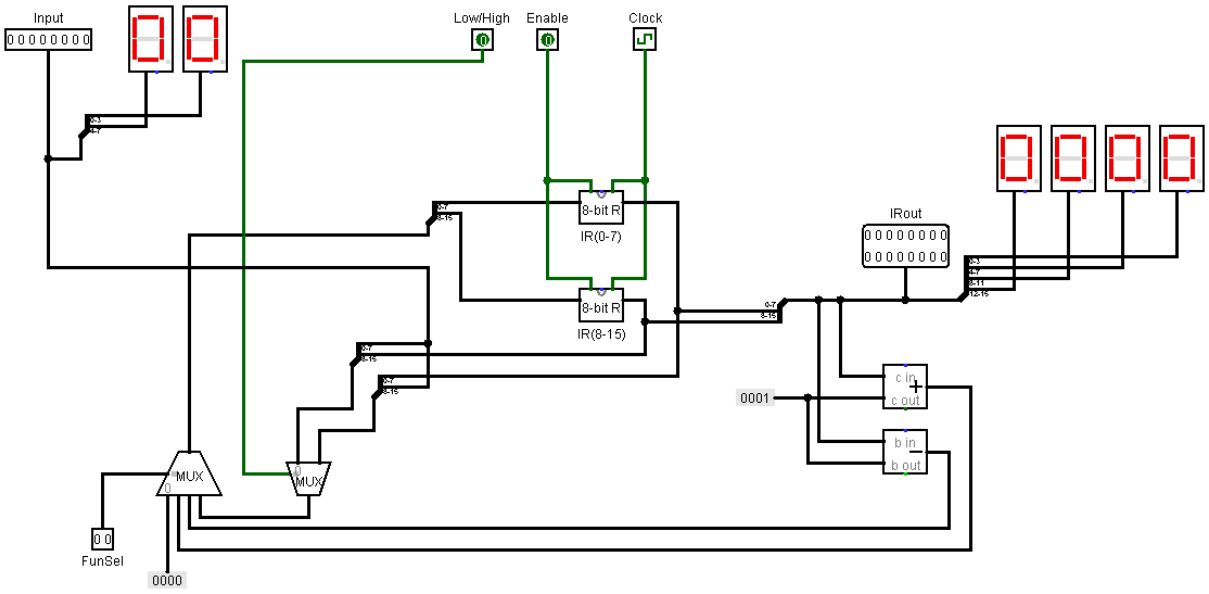


Figure 16: The circuit that 16-bit IR Register

E	I	FunSel	Low/High	Output
1	14	11	0	0014
1	14	11	1	1414
1	Ø	10	Ø	1413
1	Ø	01	Ø	1414
0	Ø	Ø	Ø	1414
1	Ø	0	Ø	0000

Figure 17: The test cases of the circuit

The circuit is validated with some test cases shown in Figure 17. The test cases works step by step that are listed below.

- Load $->$ 14 to IR(0-7)
- Load $->$ 14 to IR(8-15)
- Decrement by 1
- Increment by 1
- Enable $->$ OFF
- Enable $->$ ON: FunSel $->$ Clear

3 CONCLUSION

In conclusion, we have successfully implemented our own-designed 8-bit and 16-bit register and we constructed a register structure that has some functionalities which are clear, load, increment, and decrement. In clear mode, the register sets to zero all data it has and the register replaces the data that has with the given inputs in the load mode. Finally, in decrement and increment modes, the register increments or decrements depends on the selected mode the data that has with 1. As a final comment, what we have learned from this project, we had a opportunity to examine closely register structures and we have observed more clearly background of how to keep data in computers.