ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION
PROJECT REPORT

Project : 1

DATE : 09.03.2019

GROUP NO : 34

GROUP MEMBERS:

150160150 : MEHMET ZULFIKAR BARMAN

150170005 : SONER OZTURK

150170062 : MEHMET FATIH YILDIRIM

150170098 : SAHIN AKKAYA

150170715 : MUSTAFA TEMURTAS

SPRING 2019

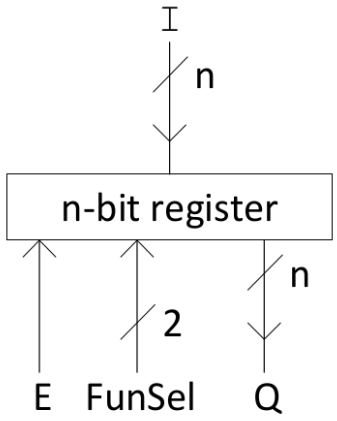
Contents

1 INTRODUCTION

In this project, we designed and implemented registers and register files.

2 Part 1

In this part we designed two different types of registers. One of them is 8-bit and the other is 16-bit with the same functionalities which are controlled by 2-bit control signals (FunSel) and an enable input (E). The graphic symbol of the registers and the characteristic table is shown in Figure 1.



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

Figure 1: The graphic symbol of registers and the characteristic table

2.1 Designing 8-bit register

In order to implement decrement and increment operations in the 8-bit register while FunSel is 01 and 10, we used 8 full adders. When FunSel is 01, 0 is taken as an input by full adders except the first one via splitter which is connected to FunSel. Instead of the input taken from splitter, constant 1 is connected to first full adder as an input as seen in the Figure 2. For example, consider an 8-bit input named INPUT. Decrement operation is implemented as INPUT - 00000001. When 2's complement is applied, new value will be equal to INPUT + 11111110 + 1 which is INPUT + 11111111 where one of the inputs of all full adders is 1 and the other input is the one bit of INPUT number respectively. Increment operation on the same input is implemented as INPUT + 00000001 where one of the inputs of all full adders is one bit of INPUT and the other input of all full adders is 0 except the first one which has the input value 1. Therefore, first full adder has constant input value 1 and the others have 0 or 1 which is changing according to the operation.

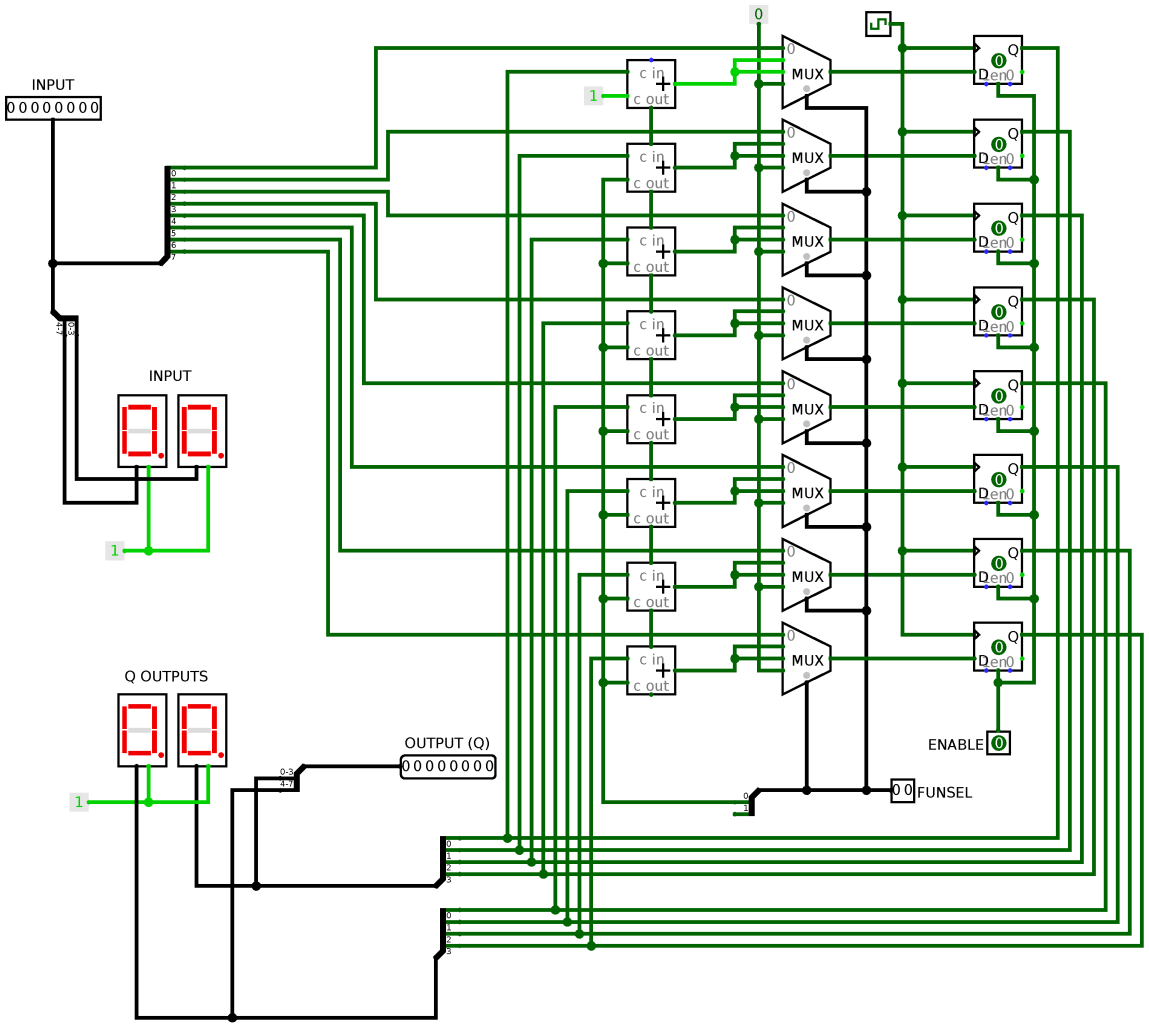


Figure 2: 8-bit register with given functionalities

In order to implement which operation is applied for each FunSel values, we used 8 4:1 multiplexers. Each multiplexers take one bit of 8-bit input respectively as input0. While FunSel is 00, input0 is selected, load operation applied and exactly the same value of input is seen on displays and output. Input1 and input2 of each multiplexers are connected to one full adder's output. While FunSel is 01, decrement operation is applied and while FunSel is 10, increment operation is applied as explained above. When FunSel is 11, clear operation should be applied. Therefore, constant 0 is connected to all multiplexers as input3.

When enable input of D type flip-flop is 0, no matter what FunSel value is the output value is retained.

2.2 Designing 16-bit register

16-bit register is designed in the same way with 8-bit register as seen in Figure 3.

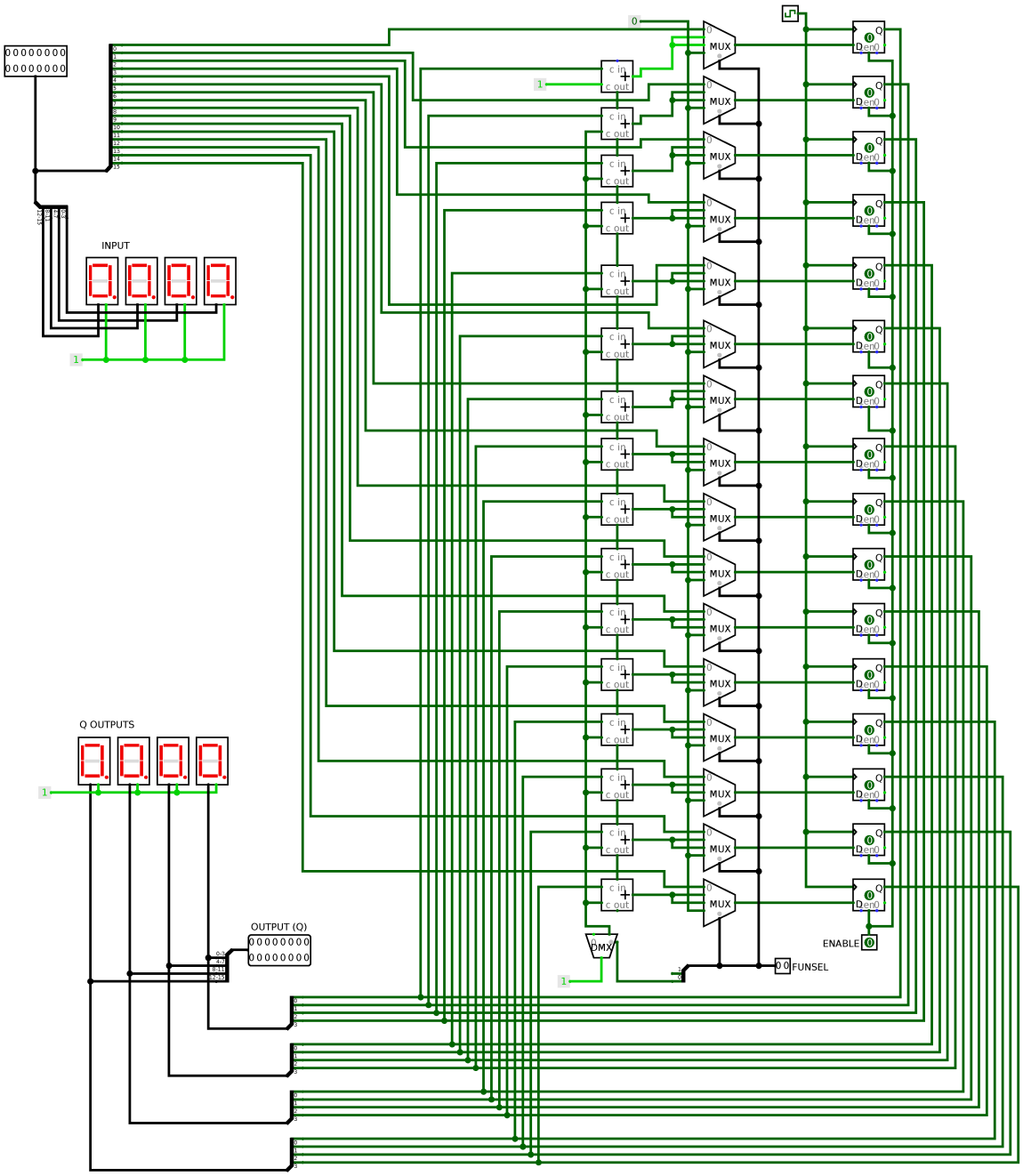
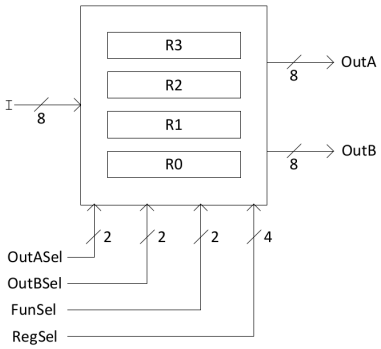


Figure 3: 16-bit register with given functionalities

3 Part 2

In this part we designed 3 different types of register files which are 8-bit general purpose register, 8-bit address register and 16-bit IR register respectively.

3.1 Designing 8-bit general purpose register



OutASel	Output A
00	R3
01	R2
10	R1
11	R0

OutBSel	Output B
00	R3
01	R2
10	R1
11	R0

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

Figure 4: The graphic symbol of registers and the characteristic table

The graphic symbol of the registers and the characteristic table is shown in Figure 4.

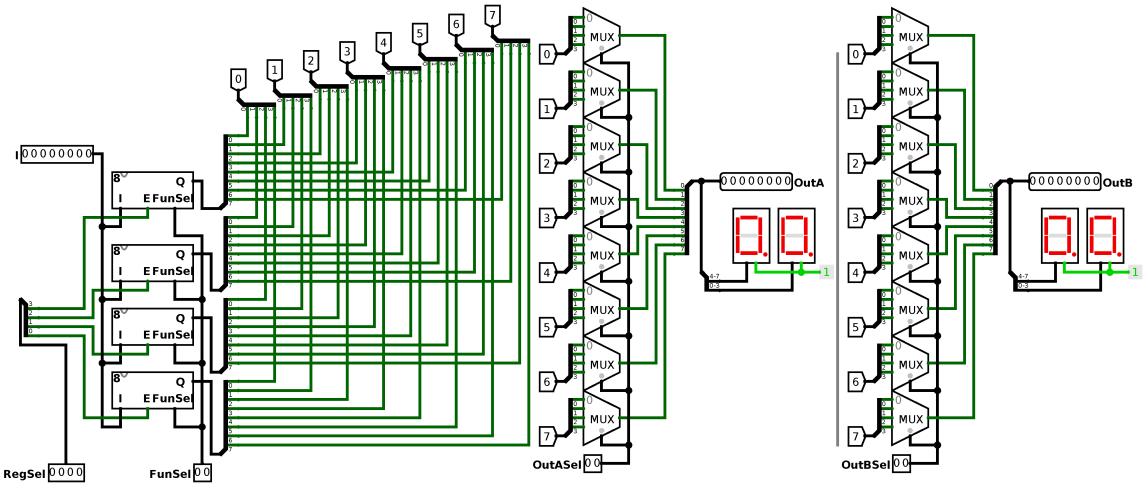


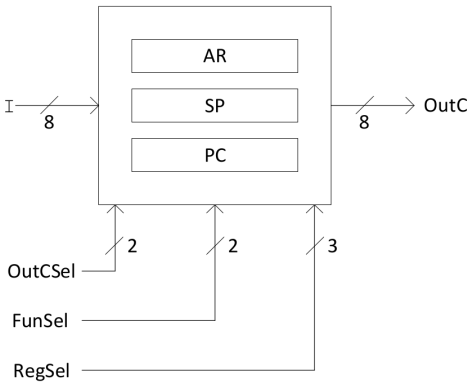
Figure 5: 8-bit general purpose registers, inputs, and outputs

In order to decide which register will be affected by the function selected by FunSel, we connected 4-bit signal (RegSel) to Enable (E) inputs of the registers. Then we connected

each of the 8-bit outputs of registers to 4:2 multiplexers. Outputs of R3 are connected to I0 of all multiplexers, outputs of R2 are connected to I1 of all multiplexers, so on so forth. By connecting 2-bit signal (OutASel) as selector line to all multiplexers, we achieved to display correct output on LED. Since OutASel and OutBSel operates same, we did the same things to show Output B.

3.2 Designing 8-bit address register

Inputs and outputs diagram of three address registers and which register is selected by input OutCSel for giving its contents to the output OutC is shown in Figure 6.



OutCSel	Output C
00	AR
01	SP
10	PC
11	PC

Figure 6: The graphic symbol of address registers and the characteristic table of OutCSel

In this part, we designed a system consisting three 8-bit address registers AR, SP and PC as seen in Figure 7.

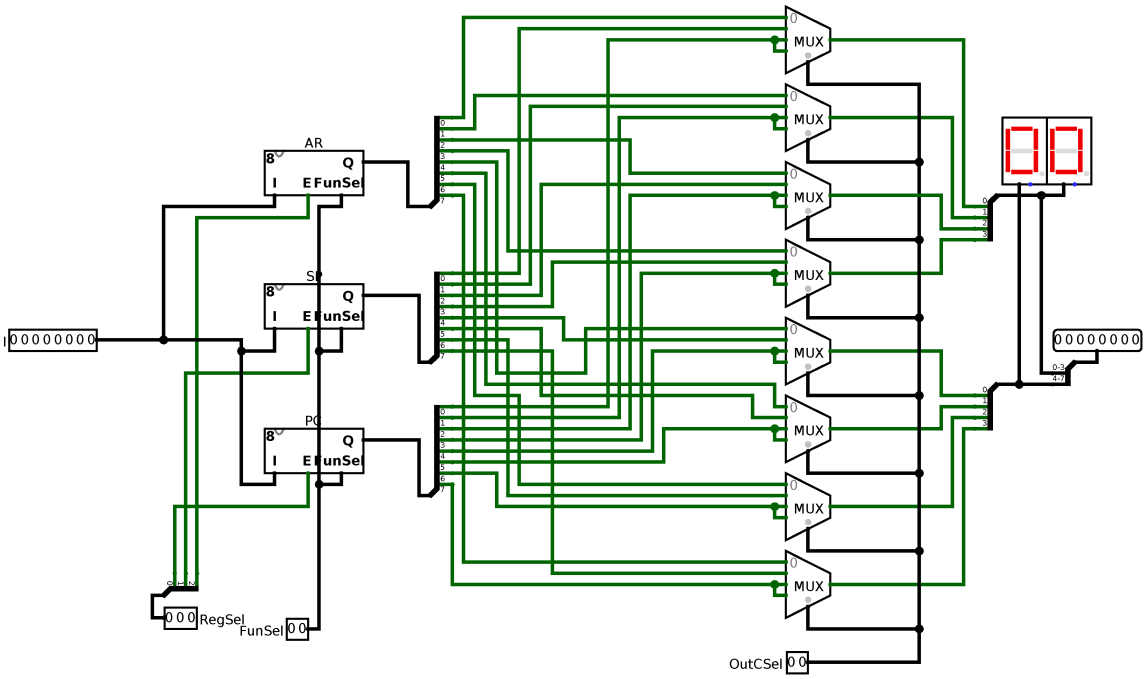
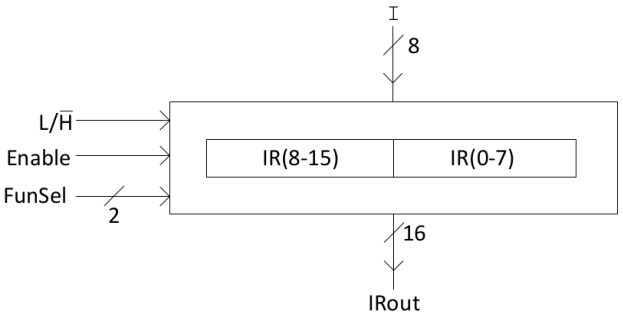


Figure 7: 8-bit address registers with given functionalities

We connected 8-bit input to each three registers and RegSel is connected as Enable input in order to choose which register is enable which is not. The outputs of registers are connected to eight 4x2 multiplexers with using splitters.AR register's output is connected to I0 input of all MUXs and SP's output to I1 input and PC's output to I2 and I3 inputs. OutCSel is connected to all multiplexers as selector line to select which register's content is given to the output.

3.3 Designing 16-bit IR register

In this part we designed a 16-bit IR register whose graphic symbol and characteristic table are shown in Figure 8.



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

Figure 8: Graphic symbol of the IR register (Left) and its characteristic table (Right)

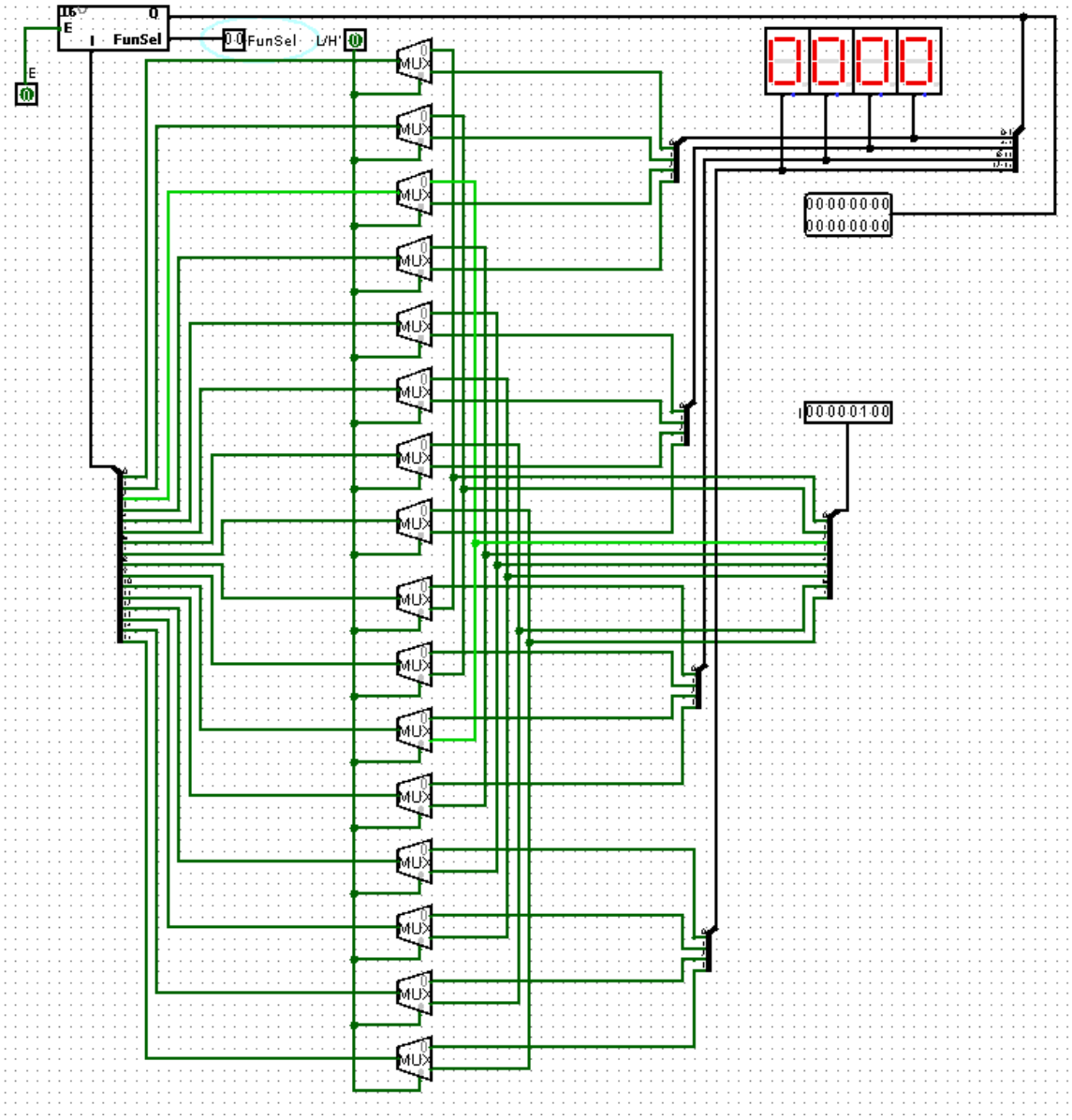


Figure 9: 16-bit IR register with given functionalities

We used 16-bit register which we designed at Part-1. The difference is input is 8-bit in this part. We connected a pin to all multiplexers as selector input to choose when FunSel is 00(load operation) which bits is shown as output. When pin is 1, from 0 to 7 bits, when pin is 0 from 8 to 15 bits is written.The other operations as same as 16-bit register which is designed in part 1.

4 CONCLUSION

In the project we designed some types of registers and register files. In part 1 we designed two registers which are 8-bit and 16-bit as library units which is used in the part 2. In part 2 with using part1 registers, we designed and implemented 3 different register files.

While designing the project, team work was essential and efficient for us. We all planned together how everything should be designed, implemented in each steps of projects and in groups we drew Logisim circuits, then revised each parts. Finally we gathered all work and wrote the report.