ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT 1 REPORT

CRN	:	21334
LECTURER	:	Assoc. Prof. Gökhan İnce

GROUP MEMBERS:

150220916	:	Rampıa Perente
150210020	:	Abdullah Saatçi

SPRING 2024

CONTENTS

Contents

1 INTRODUCTION [10 points]

During this project, we have experienced the design of a basic computer. In the first part, we have designed a register controlled by a FunSel. In the second part, an Instruction Register(IR) has been designed which uses the L'H signal to load either the bits 7-0 or bits 15-8. Subsequently, using the register designed in part 1, a system consisting of 8 registers controlled by RegSel and ScrSel has been implemented. Furthermore, with the help of a program counter(PC), address register(AR), and stack pointer(SP) an address register file(ARF) has been created. In the third part, we have designed an Arithmetic Logic Unit(ALU) performing operations controlled by Funsel using the A and B inputs and checking if carry or overflow occurs. Also, it checks if the output of the operation done by ALU is zero or negative and afterward stores this information in a 4-bit register. In the last part, an Arithmetic Logic Unit System has been implemented using the previously created ARF, ALU, RF, and IR, and the newly created Memory, multiplexer(MUX) A, multiplexer(MUX) B, and multiplexer(MUX) C.

2 TASK DISTRIBUTION

The whole project has been done together.

3 MATERIALS AND METHODS [40 points]

3.1 MATERIALS

- Vivado

3.2 METHODS

3.2.1 Part 1

Initially, using a procedural ”always” block gets executed only when the rising edge of the clock arrives, we have implemented a 16-bit register controlled by 3-bit Funsel and Enable(E) that functions according to the table below:

Enable(E)	Funsel	Q+
0	-	Q = Q
1	000	Q = Q - 1
1	001	Q = Q + 1
1	010	Q = I
1	011	Q = 0
1	100	Q[15:8] = 0
		Q[7:0] = I[7:0]
1	101	Q[7:0] = I[7:0]
1	110	Q[15:8] = I[7:0]
1	111	I[7] = 0 ⇒ Q[15:8] = 0
		I[7] = 1 ⇒ Q[15:8] = 1
		Q[7:0] = I[7:0]

Table 1: Characteristic Table

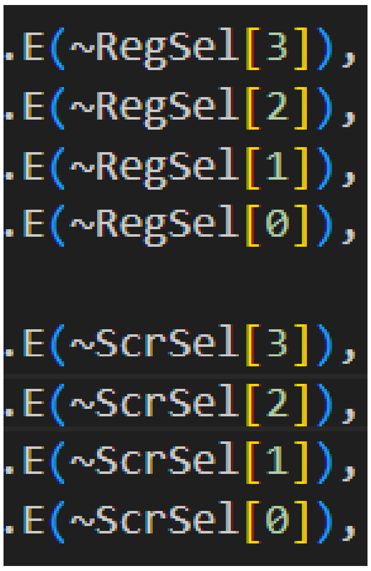
3.2.2 Part 2

For part 2a, using a procedural "always" block gets executed only when the rising edge of the clock arrives, we have designed an IR register with 16-bit capacity but gets 8-bit inputs and loads either the lower or higher half according to the table below:

L'H	Write	IR+
-	0	IR = IR
0	1	IR[7:0] = I
1	1	IR[15:8] = I

Table 2: Characteristic Table

For part 2b, with the help of the register designed in part 1, a system with 8 registers has been designed. One-half of the registers have been selected using the complement of the common bit of 4-bit for 4 registers RegSel assigned to E and the other half using the complement of the common bit of 4-bit ScrSel assigned to E.



```
.E(~RegSel[3]),
.E(~RegSel[2]),
.E(~RegSel[1]),
.E(~RegSel[0]),

.E(~ScrSel[3]),
.E(~ScrSel[2]),
.E(~ScrSel[1]),
.E(~ScrSel[0]),
```

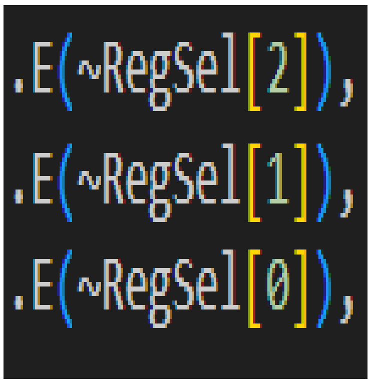
Figure 1: Register Selection

Furthermore, we have implemented two 3-bit selectors, OutASel and OutBSel, selecting registers to be output for the 16-bit outputs Out A and Out B according to the table below:

OutASel/OutBSel	OutA/OutB
000	R1
001	R2
010	R3
011	R4
100	S1
101	S2
110	S3
111	S4

Table 3: Characteristic Table

For part 2c, with the help of the register designed in part 1, a system with three 16-bit address registers, a program counter(PC), an address register(AR), and a stack pointer(SP) has been designed. Again, registers have been selected using the complement of the common bit for 3 registers of 3-bit RegSel assigned to E.



```
.E(~RegSel[2]),
.E(~RegSel[1]),
.E(~RegSel[0]),
```

Figure 2: Register Selection

Furthermore, we have implemented two 2-bit selectors, OutCSel and OutDSel, selecting registers to be output for the 16-bit outputs Out C and Out D according to the table below:

OutCSel/OutDSel	OutC/OutD
00	PC
01	PC
10	AR
11	SP

Table 4: Characteristic Table

3.2.3 Part 3

For this part, a 16-bit input Arithmetic Logic Unit(ALU) has been implemented to perform different operations controlled by a 5-bit Funsel using the A and B inputs. This system checks whether the 16-bit output(ALUOut) is equal to zero or negative or if a carry bit or overflow exists. Afterward, necessary implementations are done and the Z, C, N, and O flags are stored in a 4-bit register.

3.2.4 Part 4

Lastly, an Arithmetic Logic Unit System has been implemented using the previously created ARF, ALU, RF, and IR, and the newly created 8-bit input and 8-bit output Memory, 4-bit input and 16-bit output MUX A, 4-bit input and 16-bit output MUX B, and 2-bit input and 8-bit output MUX C. Furthermore, we have implemented three 2-bit selectors for multiplexers, MUXASel, MUXBSel, and MUXCSel to select which part of the the ALU system will be used and direct them to the 16-bit outputs of multiplexers, MuxAOut, MuxBOut, and MuxCOut.

MuxASel/MuxBSel	MuxAOut/MuxBOut
00	ALUOut
01	ARF OutC
10	Memory Output
11	IROut[7:0]

Table 5: Characteristic Table

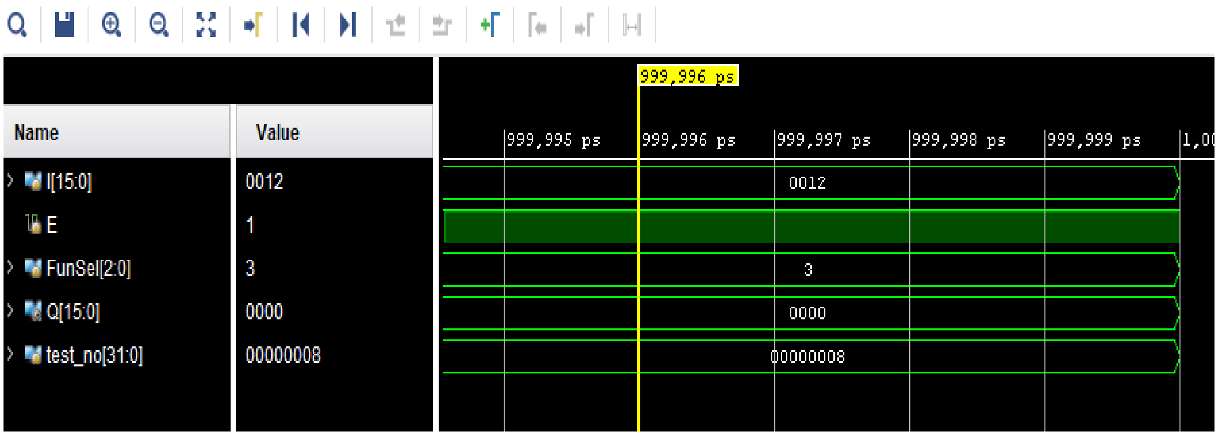
MuxCSel	MuxCOut
0	ALUOut[7:0]
1	ALUOut[15:8]

Table 6: Characteristic Table

4 RESULTS [15 points]

4.1 Part 1

Firstly, we have implemented a 16-bit register controlled by 3-bit Funsel and Enable(E).



Name	Value
I[15:0]	0012
E	1
FunSel[2:0]	3
Q[15:0]	0000
test_no[31:0]	00000008

Figure 3: Register Simulation

4.2 Part 2

4.2.1 Part 2a

We have designed an IR register with a 16-bit capacity that gets 8-bit inputs and loads either the bits 7-0 or bits 15-8 according to the L'H signal.

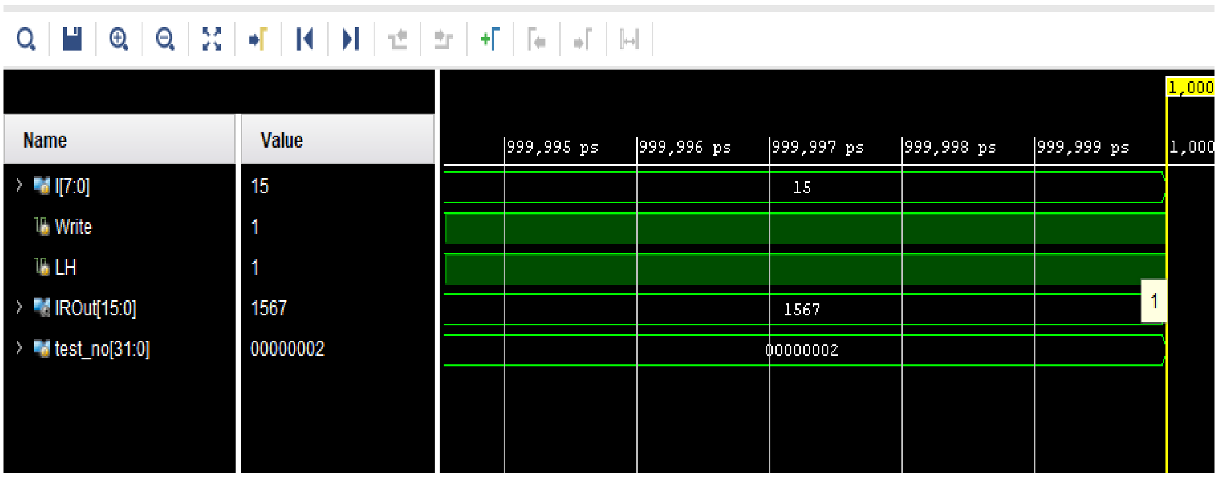
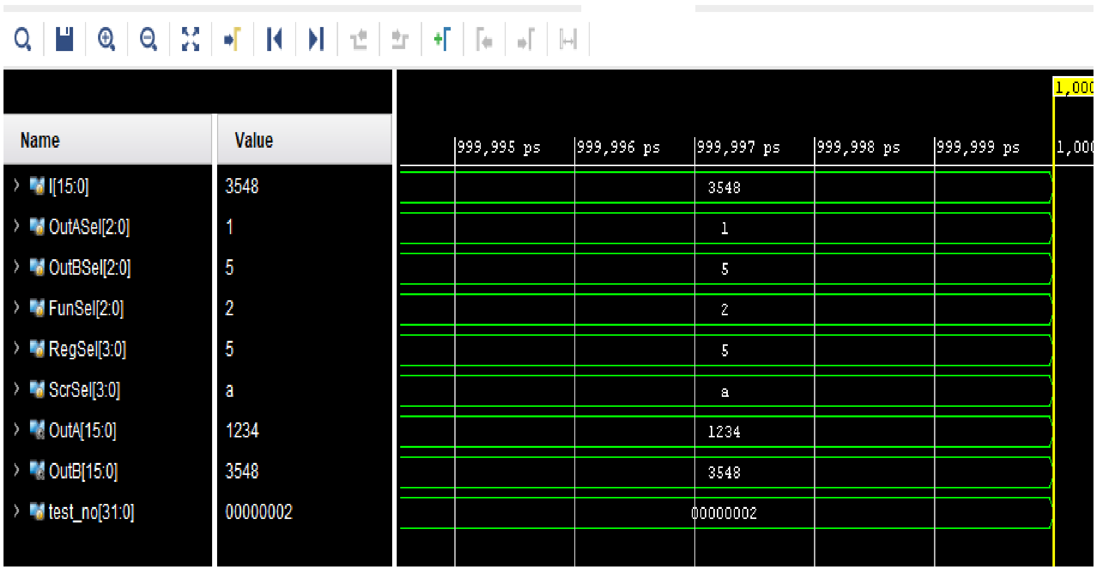


Figure 4: Instruction Register Simulation

4.2.2 Part 2b

A system with 8 registers has been designed using the register implemented in part 1. Registers R1, R2, R3, and R4 have been selected using the complement of the common bit of 4-bit for 4 registers RegSel assigned to E and the registers S1, S2, S3, and S4 using the complement of the common bit of 4-bit ScrSel assigned to E. Moreover, we have implemented OutASel and OutBSel to control registers to be output for Out A and Out B.

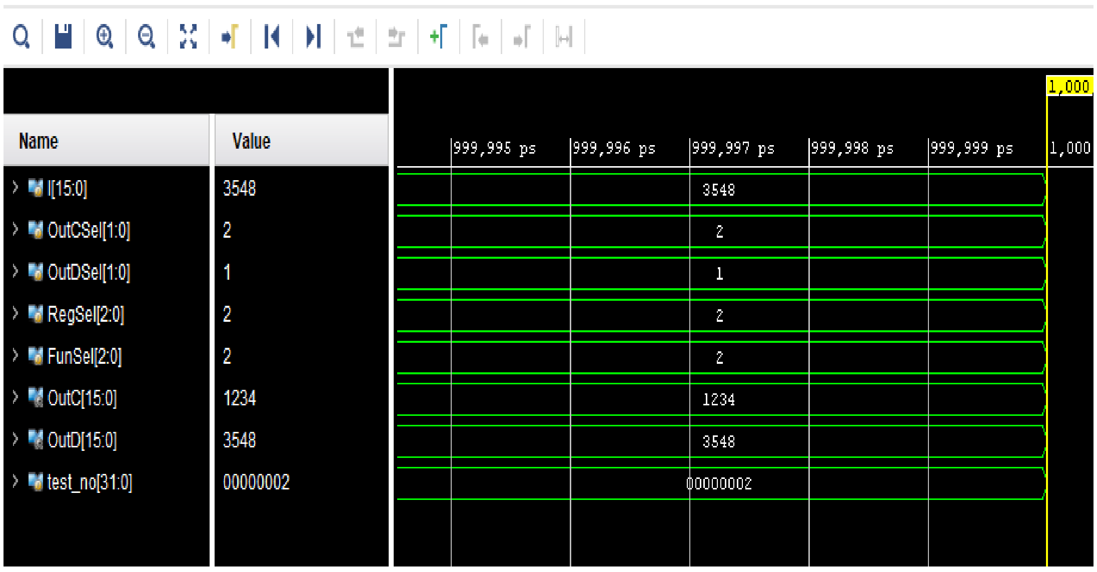


Name	Value
I[15:0]	3548
OutASel[2:0]	1
OutBSel[2:0]	5
FunSel[2:0]	2
RegSel[3:0]	5
ScrSel[3:0]	a
OutA[15:0]	1234
OutB[15:0]	3548
test_no[31:0]	00000002

Figure 5: Register File Simulation

4.2.3 Part 2c

A system with a PC, an AR, and an SP has been designed using the register implemented in part 1. Again, registers have been selected using the complement of the common bit for 3 registers AR, PC, and SP of RegSel assigned to E. Furthermore, we have implemented OutCSel and OutDSel, to select registers to be output for Out C and Out D.

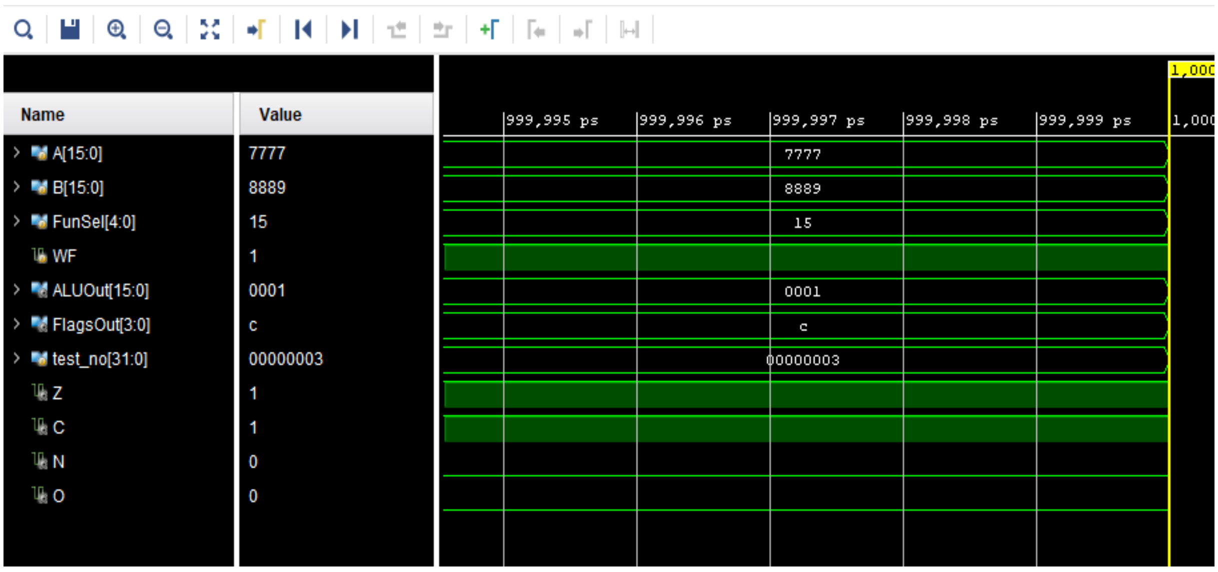


Name	Value
I[15:0]	3548
OutCSel[1:0]	2
OutDSel[1:0]	1
RegSel[2:0]	2
FunSel[2:0]	2
OutC[15:0]	1234
OutD[15:0]	3548
test_no[31:0]	00000002

Figure 6: Address Register File Simulation

4.3 Part 3

An ALU has been implemented to perform different operations selected via Funsel using the A and B inputs. This system checks whether the ALUOut is equal to zero or negative or if a carry bit or overflow exists.

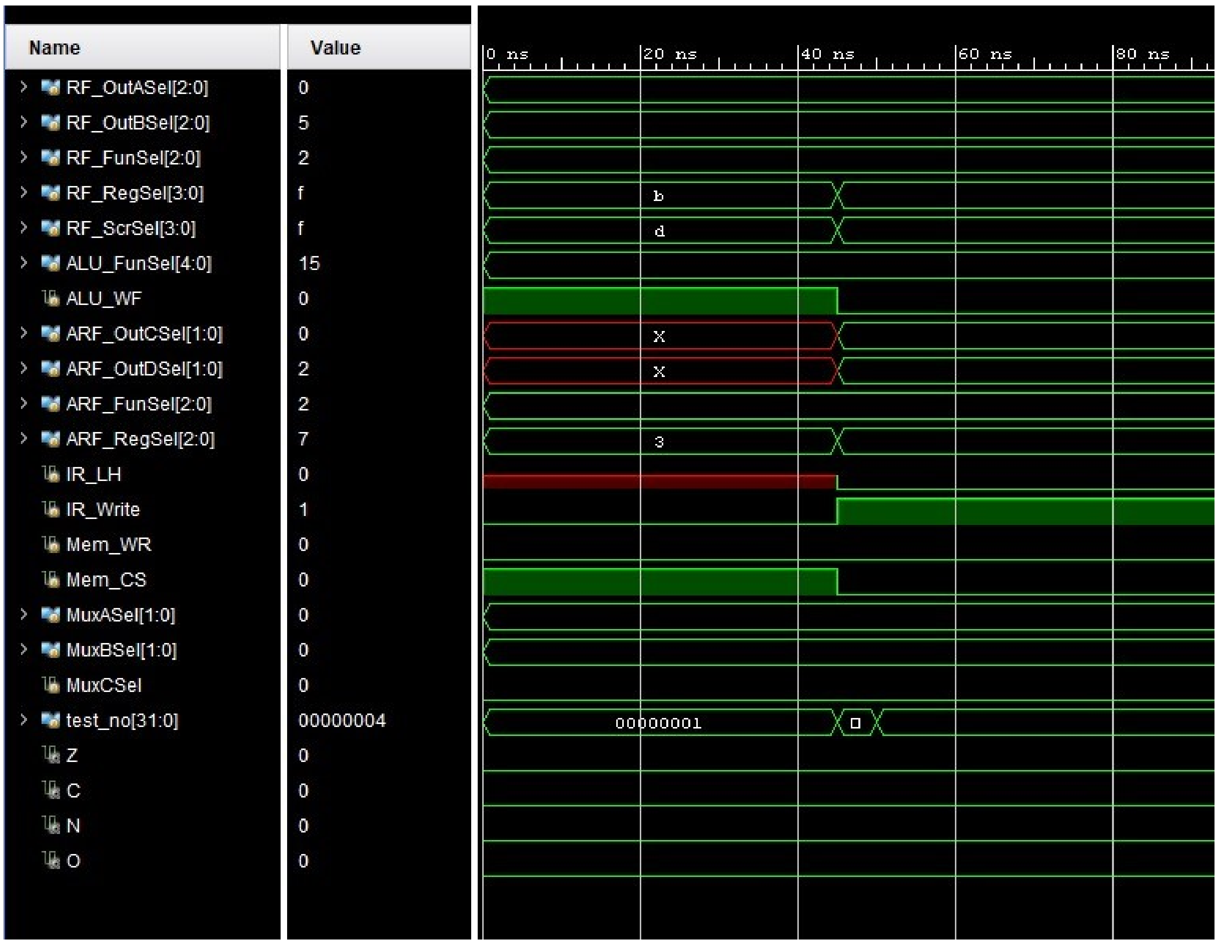


Name	Value
A[15:0]	7777
B[15:0]	8889
FunSel[4:0]	15
WF	1
ALUOut[15:0]	0001
FlagsOut[3:0]	c
test_no[31:0]	00000003
Z	1
C	1
N	0
O	0

Figure 7: Arithmetic Logic Unit Simulation

4.4 Part 4

An ALU System has been implemented using the previously created ARF, ALU, RF, and IR, and the newly created Memory, and multiplexers MUX A, MUX B, and MUX C. Furthermore, selectors for the multiplexers, MUXASel, MUXBSel, and MUXCSel have been designed to select which part of the ALU system will be used and direct them to the outputs of multiplexers, MuxAOut, MuxBOut, and MuxCOut.



Name	Value
RF_OutASel[2:0]	0
RF_OutBSel[2:0]	5
RF_FunSel[2:0]	2
RF_RegSel[3:0]	f
RF_ScrSel[3:0]	f
ALU_FunSel[4:0]	15
ALU_WF	0
ARF_OutCSel[1:0]	0
ARF_OutDSel[1:0]	2
ARF_FunSel[2:0]	2
ARF_RegSel[2:0]	7
IR_LH	0
IR_Write	1
Mem_WR	0
Mem_CS	0
MuxASel[1:0]	0
MuxBSel[1:0]	0
MuxCSel	0
test_no[31:0]	00000004
Z	0
C	0
N	0
O	0

Figure 8: Arithmetic Logic Unit System Simulation

5 DISCUSSION [25 points]

The project aimed at a basic computer, which consists of registers, an instruction register, an address register file, an ALU, and an ALU system. Implementing the register used in the register file and the address register file indicates to us how data is stored and manipulated. The instruction register, based on the L'H signal, has been loaded in either the lower or higher half of the input. The ALU system, controlled by the Funsel signal, performed arithmetic and logic operations using flags. Overall, the project served as an invaluable learning experience in computer organization and digital systems design, equipping the team with fundamental skills for in this field.

6 CONCLUSION [10 points]

Throughout the project, we gained a lot of knowledge about the use of ALU, memory units, and multiplexers in a basic computer. However, we had trouble with the clock signals while implementing the modules in part 1 and part 2. Also, checking the conditions for overflow occurrence was quite challenging in part 3.

[1]

REFERENCES

[1] Darte et al. Vivado. https://www.xilinx.com/developer/authors.html.