BLG222E Computer Organization

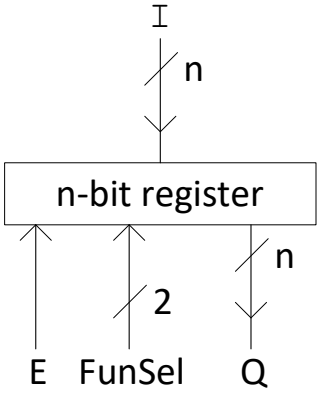
Project 1

Due Date: 11.05.2022 18:00

In this project consisting of 4 parts, registers and register files will be designed and implemented. Design each part **as a module**, so that it can be reused in other parts. **You must simulate each module for each combination of input.**

(Part-1) Design 2 types of registers: **(1)** 8-bit register and **(2)** 16-bit register. These registers have 4 functionalities that 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. Symbol ϕ means don't care. These registers will be used in **Part-2**.



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

Figure 1: Graphic symbol of the registers (Left) and the characteristic table (Right)

(Part-2) Design a register file (a structure that contains many registers) that works as follows.

(Part-2a) Design the system shown in Figure 2 which consists of four 8-bit general purpose registers: **R1**, **R2**, **R3,** and **R4**. The details of inputs and outputs are as follows.

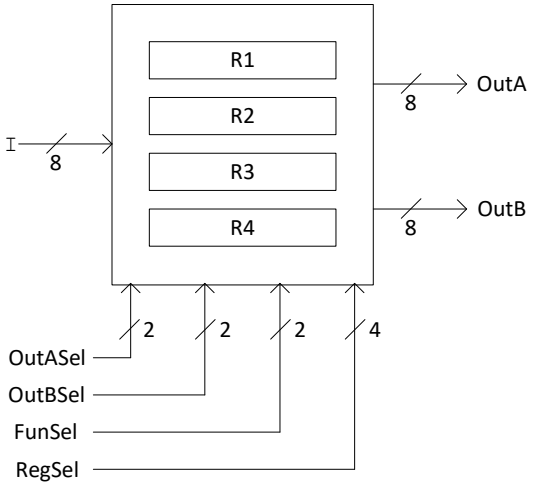


Figure 2: 8-bit general purpose registers, inputs, and outputs

OutASel and **OutBSel** are used to feed output lines **OutA** and **OutB**, respectively. 8 bits of the selected registers are output to **OutA** and **OutB**. Figure 3 shows selection of output registers based on the **OutASel** and **OutBSel** control inputs.

OutASel	Output A
00	R1
01	R2
10	R3
11	R4

OutBSel	Output B
00	R1
01	R2
10	R3
11	R4

Figure 3: OutASel and OutBSel controls

RegSel (*Figure 4*) is a 4-bit signal that selects the registers to apply the function that is determined by **FunSel** (*Figure 5*) signal.

RegSel	Enabled Registers
0000	All registers are enabled (Function selected by FunSel will be applied to R1, R2, R3 and R4)
0001	R1, R2 and R3 are enabled (Function selected by FunSel will be applied to R1, R2 and R3)
0010	R1, R2 and R4 are enabled, Function selected by FunSel will be applied to R1, R2 and R4
0011	R1 and R2 are enabled (Function selected by FunSel will be applied to R1 and R2)
0100	R1, R3 and R4 are enabled (Function selected by FunSel will be applied to R1, R3 and R4)
0101	R1 and R3 are enabled (Function selected by FunSel will be applied to R1 and R3)
0110	R1 and R4 are enabled (Function selected by FunSel will be applied to R1 and R4)
0111	Only R1 is enabled (Function selected by FunSel will be applied to R1)
1000	R2, R3 and R4 are enabled (Function selected by FunSel will be applied to R2, R3 and R4)
1001	R2 and R3 are enabled (Function selected by FunSel will be applied to R2 and R3)
1010	R2 and R4 are enabled (Function selected by FunSel will be applied to R2 and R4)
1011	Only R2 is enabled (Function selected by FunSel will be applied to R2)
1100	R3 and R4 are enabled (Function selected by FunSel will be applied to R3 and R4)
1101	Only R3 is enabled (Function selected by FunSel will be applied to R3)
1110	Only R4 is enabled (Function selected by FunSel will be applied to R4)
1111	N0 register is enabled (All R1, R2, R3 and R4 registers retain their values)

Figure 4:RegSel Control Input

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

Figure 5: FunSel Control Input

For example: If **RegSel** is 1001 and **FunSel** is 01, then the registers **R2** and **R3** will be incremented with next clock cycle. **R1** and **R4** will not be affected since they are not enabled by **RegSel**.

(Part-2b) Design the **address register file (ARF)** system shown in Figure 6 which consists of three 8-bit address registers: **program counter (PC), address register (AR), and stack pointer (SP)**. FunSel and RegSel works as in **Part-2a**.

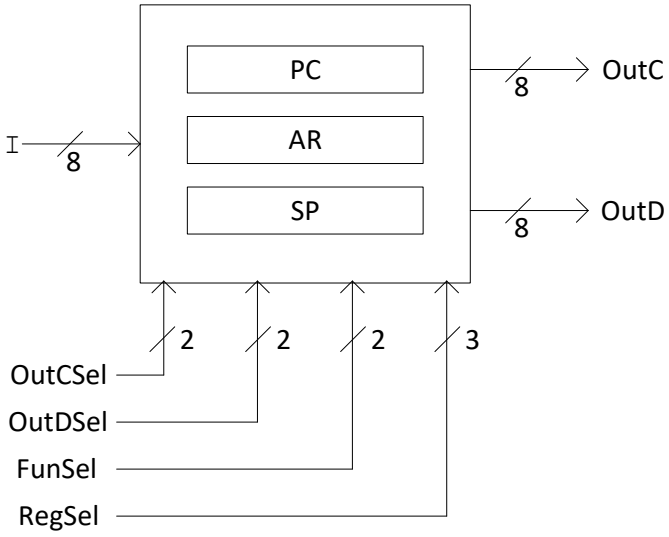


Figure 6: 8-bit address registers, inputs, and outputs

OutCSel and **OutDSel** are used to feed output lines **OutC** and **OutD**, respectively. 8 bits of the selected registers are output to **OutC** and **OutD**. Figure 7 shows selection of output registers based on the **OutCSel** and **OutDSel** control inputs.

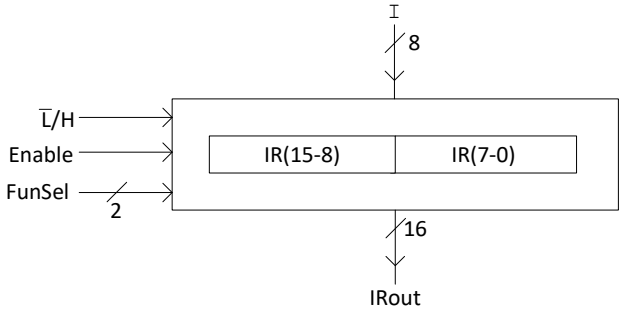
OutCSel	Output C
00	PC
01	PC
10	AR
11	SP

OutDSel	Output D
00	PC
01	PC
10	AR
11	SP

Figure 7: OutCSel and OutDSel controls

(Part-2c) Design a 16-bit **IR** register whose graphic symbol and characteristic table are given in Figure 8. You must use the 16-bit register developed in Part 1.

This register can store 16-bit binary numbers. However, the input of this register file is only 8 bits. Hence, using the 8-bit input you can load either the lower (bits 7-0) or higher (bits 15-8) half. This is determined by $\overline{L}/H$ signal.

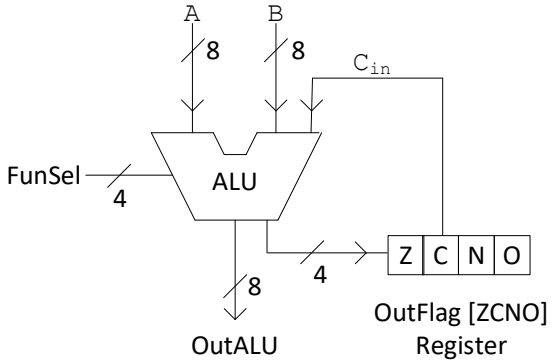


L/H	Enable	FunSel	IR+	
ϕ	0	ϕϕ	IR	(Retain Value)
ϕ	1	00	IR - 1	(Decrement)
ϕ	1	01	IR + 1	(Increment)
0	1	10	IR(15-8) <- I	(Load MSB)
1	1	10	IR(7-0) <- I	(Load LSB)
ϕ	1	11	0	(Clear)

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

(Part-3) Design an Arithmetic Logic Unit (ALU) that has two 8-bit inputs and an 8-bit output. The ALU is shown on the left side of Figure 9. The ALU functions and the flags that will be updated (i.e., **-** means that the flag will not be affected and √ means that the flag changes based on the OutALU) are given on the right side of Figure 9:

- **FunSel** selects the function of the ALU.
- **OutALU** shows the result of the operation that is selected by **FunSel** and applied on A and/or B inputs.
- **Z (zero)** bit is set if **OutALU** is zero (e.g., when **NOT B** is zero).
- **C (carry)** bit is set if **OutALU** sets the carry (e.g., when **LSL A** produces carry).
- **N (negative)** bit is set if the ALU operation generates a negative result (e.g., when **A-B** results in a negative number).
- **O (overflow)** bit is set if an overflow occurs (e.g., when **A+B** results in an overflow)**.**
- Note that **Z|C|N|O** flags are stored in a **register!**



FunSel	OutALU	Z	C	N	O
0000	A	√	–	√	–
0001	B	√	–	√	–
0010	NOT A	√	–	√	–
0011	NOT B	√	–	√	–
0100	A + B	√	√	√	√
0101	A + B + Carry	√	√	√	√
0110	A - B	√	√	√	√
0111	A AND B	√	–	√	–
1000	A OR B	√	–	√	–
1001	A XOR B	√	–	√	–
1010	LSL A	√	√	√	–
1011	LSR A	√	√	√	–
1100	ASL A	√	–	√	√
1101	ASR A	√	–	–	–
1110	CSL A	√	√	√	√
1111	CSR A	√	√	√	√

Figure 9: The ALU (Left) and its characteristic table (Right)

(Circular | **A**rithmetic | **L**ogical) **S**hift (**L**eft | **R**ight) operations are depicted in *Figure 10*, *Figure 11, and Figure 12*.

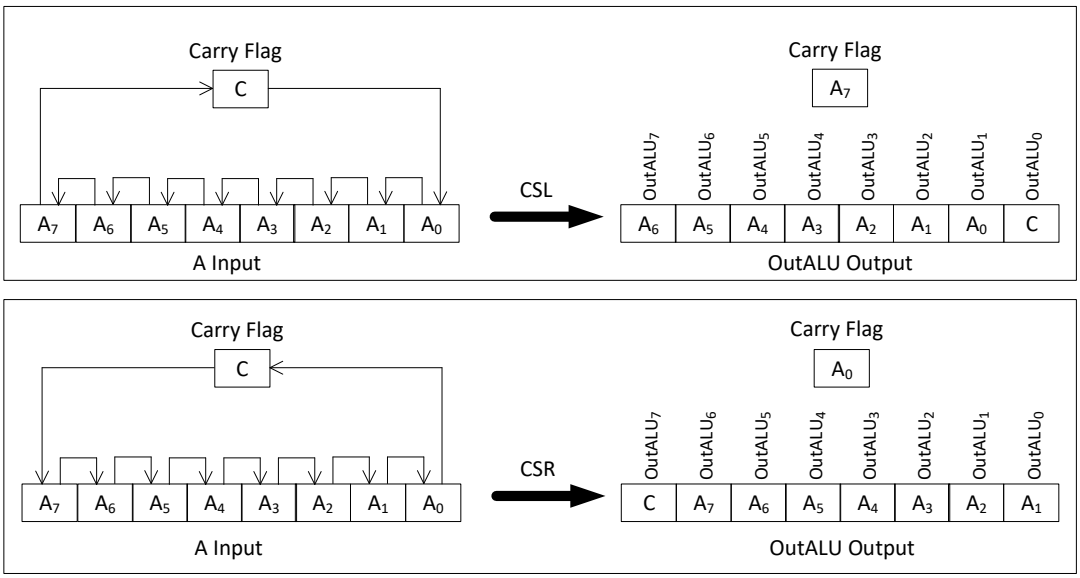


Figure 10: Circular Shift Operations

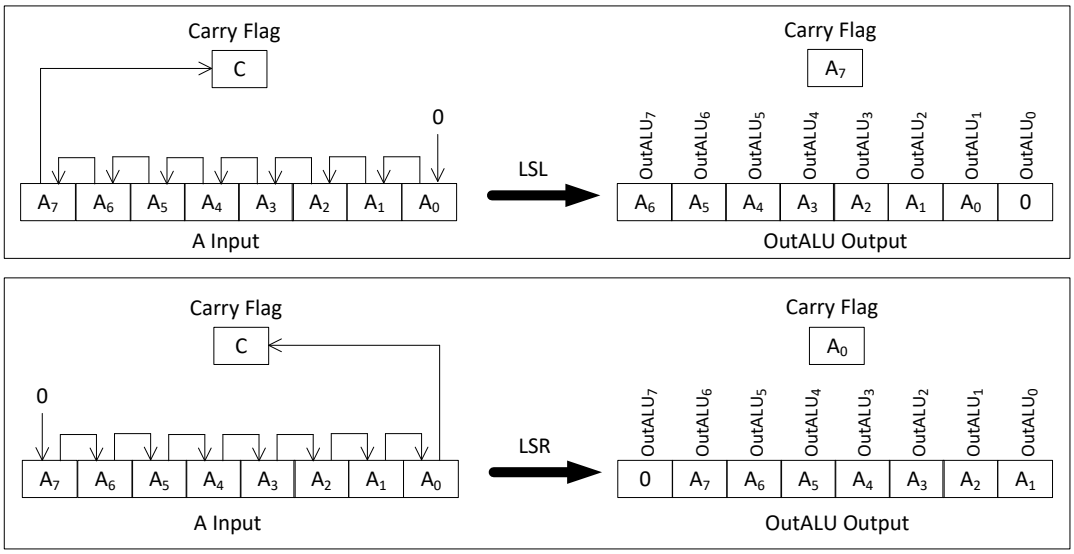


Figure 11: Logical Shift Operations

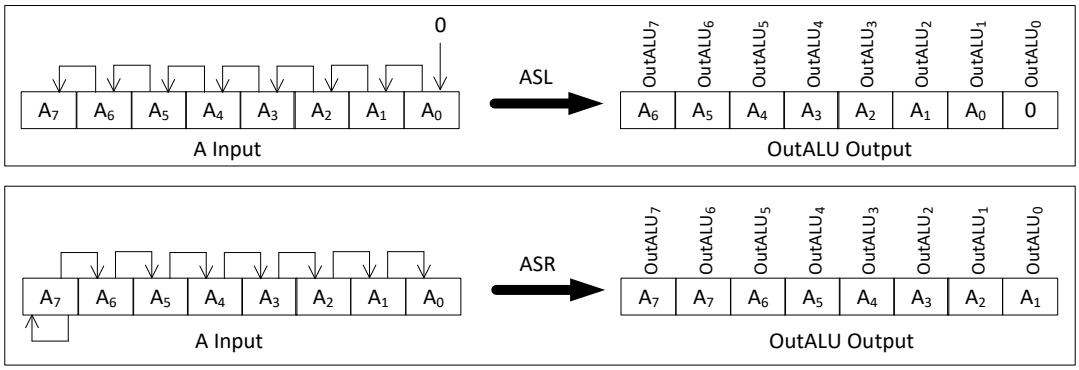
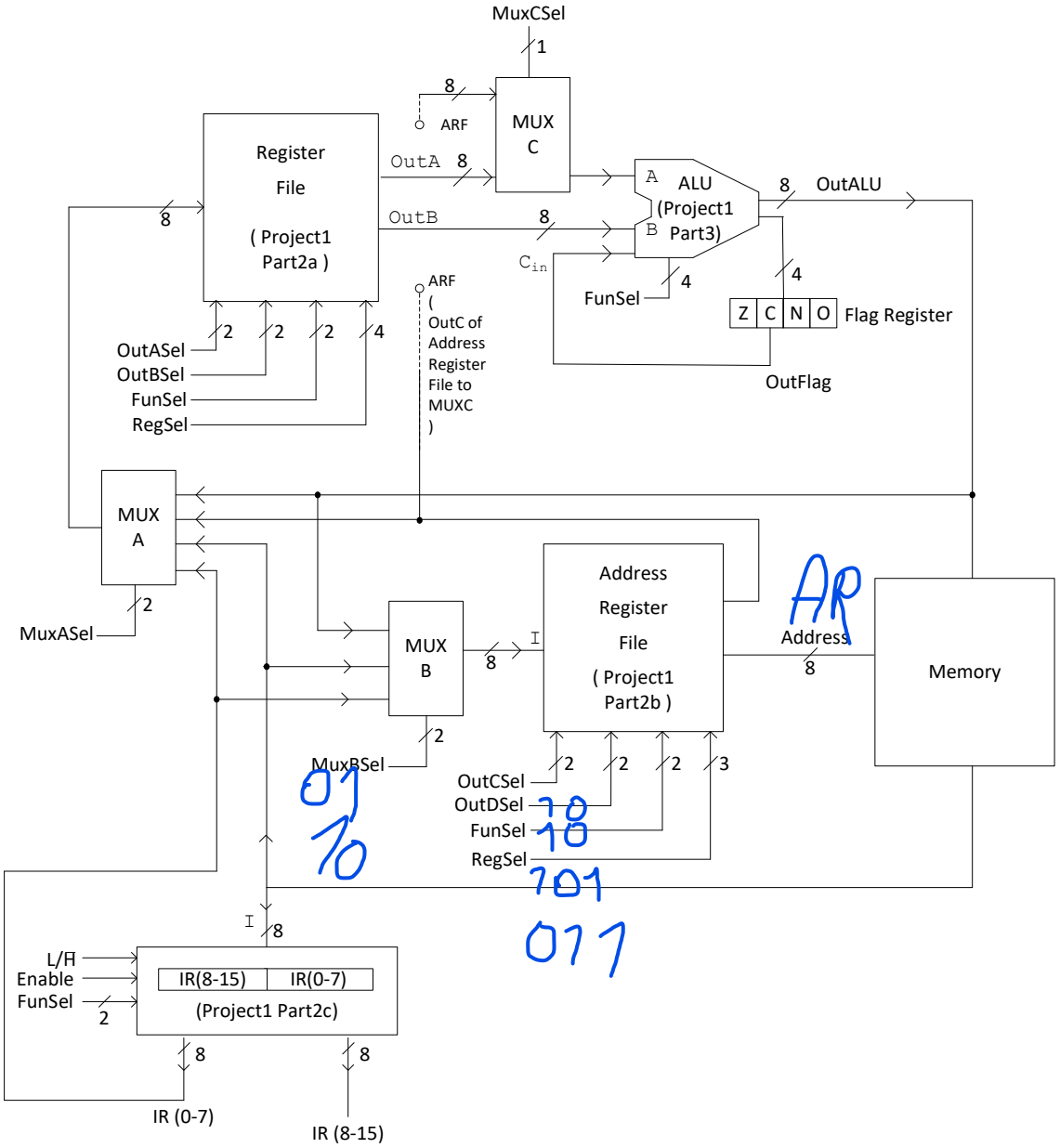


Figure 12: Arithmetic Shift Operations

(Part-4) Implement the organization in Figure 13. Please note that, **the whole system uses the same single clock.**

Hint: The RAM module can be found in the project files. You can directly add this module to your design files. You must also include the memory data to your project, the guideline has been added to homework files for

this operation.



MuxASel	MuxAOut
00	IROut (0-7)
01	Memory Output
10	Address Register OutC
11	OutALU

MuxBSel	MuxBOut
00	ϕ
01	IROut (0-7)
10	Memory Output
11	OutALU

MuxCSel	MuxCOut
0	ARF
1	OutA

Figure 13: ALU System

(Demo – Test Bench)

Each project must be able to run the Project1Test module (in TestBench.v) file. Your design must access and manipulate each value in this module. Otherwise, your program can not pass the automatic tests.

The Project1Test module can be found in the project files. You can directly add this module to your design files. You must also include the TestBench memory data in your project, the guideline has been added to homework files for this operation.

Submission:

Implement your design in Verilog HDL, upload a single compressed (zip) file to Ninova before the deadline. Only one student from each group should submit the project file (**select one member of the group as the group representative for this purpose and note his/her student ID**). This compressed file should contain your modules file (.v), simulation file (.v) and a report that contains:

- the number&names of the students in the group
- list of control inputs and corresponding functions for your design

Group work is expected for this project. All the 3 student members of the group **must** design together. Make sure to add simulations for all modules. You must ensure that all modules work properly. Simulation files will be altered in the demonstration session.

Please do not hesitate to contact Res. Asst. Kadir Özlem (kadir.ozlem@itu.edu.tr) for any question.