BLG222E Computer Organization

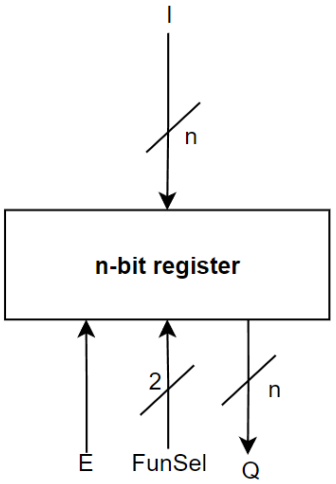
Project 1

Due Date: 12.04.2023 23:59

In this project, registers and register files will be designed and implemented. It has 4 parts. 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 an n-bit register. This register has 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. This register will be used in later parts.



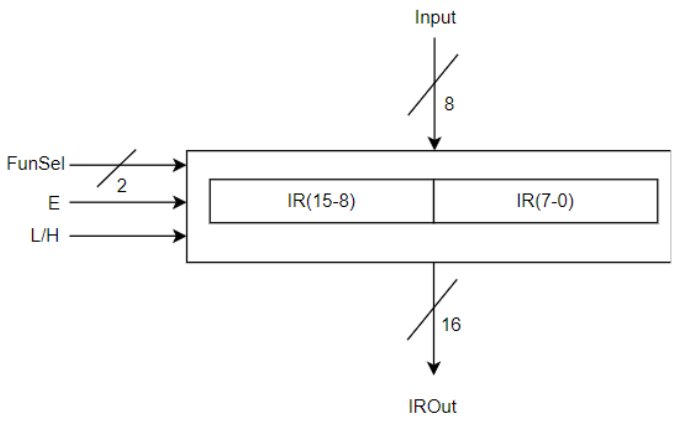
E	FunSel	Q⁺
0	ϕ	Q (retain value)
1	00	0 (Clear)
1	01	I (Load)
1	10	Q-1 (Decrement)
1	11	Q+1 (Increment)

Figure 1: Graphic symbol of the registers and the characteristic table, respectively.

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

(Part-2a) Design a 16-bit **IR** register whose block diagram and function table are given in Figure 2.

This register can store 16 bit binary data. However, the input of this register file is only 8 bits. Therefore, using the 8-bit input bus, you can load either the lower (bits 7-0) or higher (bits 15-8) half. This decision is made by the **L'/*H*** signal.



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

Figure 2: Graphic symbol of IR register and its characteristic table, respectively.

(Part-2b) Design the system shown in Figure 3 which consists of four 8-bit general purpose registers: **R1**, **R2**, **R3,** and **R4** and four 8-bit temporary registers: **T1**, **T2**, **T3,** and **T4**. The details of inputs and outputs are as follows.

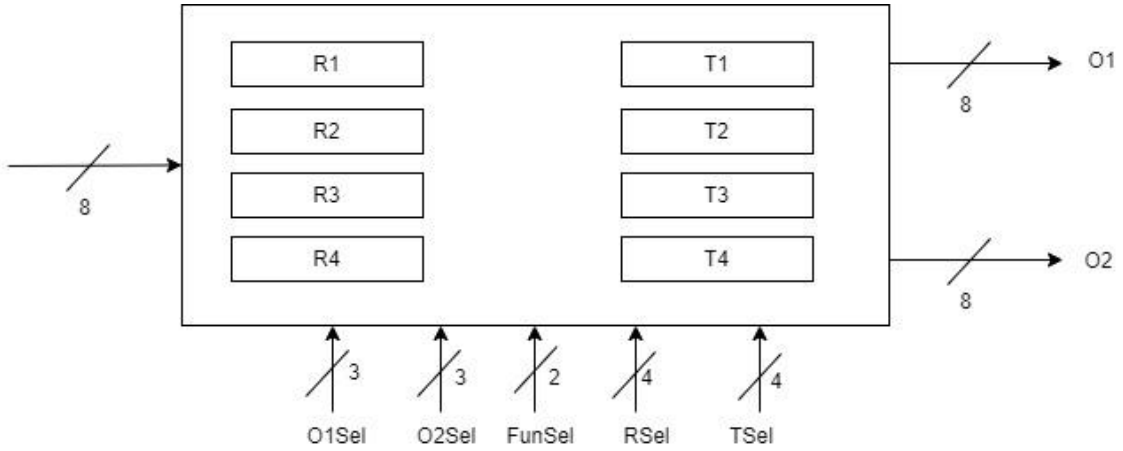


Figure 3: 8-bit general purpose and temporary registers, inputs, and outputs

O1Sel and **O2Sel** are used to feed the output lines **O1** and **O2**, respectively. 8 bits of selected registers are output to **O1** and **O2**. Table 1 shows the selection of output registers based on the **O1Sel** and **O2Sel** control inputs.

Table 1: O1Sel and O2Sel controls

O1Sel	O1	O2Sel	O2
000	T1	000	T1
001	T2	001	T2
010	T3	010	T3
011	T4	011	T4
100	R1	100	R1
101	R2	101	R2
110	R3	110	R3
111	R4	111	R4

RSel and **TSel** are 4-bit signals that select the registers to apply the function that is determined by the 2-bit **FunSel** signal which is given in Table 2. The selected registers by **RSel** and **TSel** are shown in Tables 3 and 4, respectively.

Table 2: FunSel Control Input

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

11	R_x+1 (Increment)

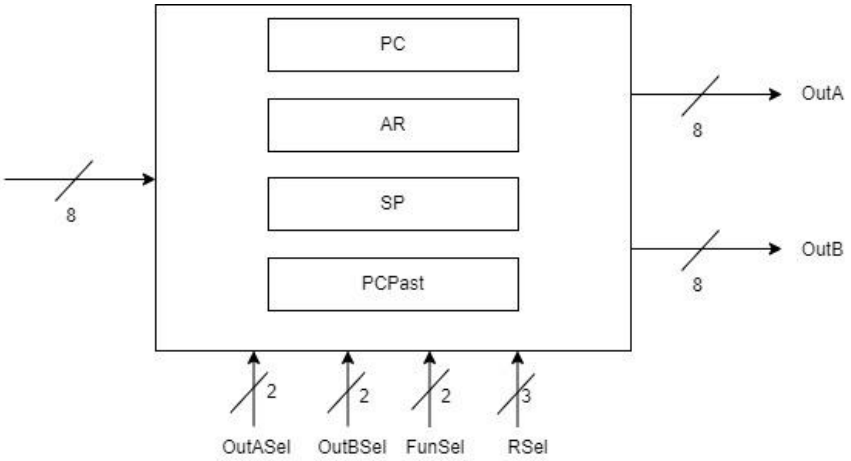
Table 3: RSel Control Input

RegSel	Enable General Purpose Registers
0000	NO general purpose register is enabled. (All registers retain their values.)
0001	Only R4 is enabled. (Function selected by FunSel will be applied to R4.)
0010	Only R3 is enabled. (Function selected by FunSel will be applied to R3.)
0011	R3 and R4 are enabled. (Function selected by FunSel will be applied to R3 and R4.)
0100	Only R2 is enabled. (Function selected by FunSel will be applied to R2.)
0101	R2 and R4 are enabled. (Function selected by FunSel will be applied to R2 and R4.)
0110	R2 and R3 are enabled. (Function selected by FunSel will be applied to R2 and R3.)
0111	R2, R3 and R4 are enabled. (Function selected by FunSel will be applied to R2, R3 and R4.)
1000	Only R1 is enabled. (Function selected by FunSel will be applied to R1.)
1001	R1 and R4 are enabled. (Function selected by FunSel will be applied to R1 and R4.)
1010	R1 and R3 are enabled. (Function selected by FunSel will be applied to R1 and R3.)
1011	R1, R3 and R4 are enabled. (Function selected by FunSel will be applied to R1, R3 and R4.)
1100	R1 and R2 are enabled. (Function selected by FunSel will be applied to R1 and R2.)
1101	R1, R2 and R4 are enabled. (Function selected by FunSel will be applied to R1, R2 and R4.)
1110	R1, R2 and R3 are enabled. (Function selected by FunSel will be applied to R1, R2 and R3.)
1111	All general purpose registers are enabled. (Function selected by FunSel will be applied to R1, R2, R3 and R4.)

Table 4: TSel Control Input

TSel	Enable General Purpose Registers
0000	NO general purpose register is enabled. (All registers retain their values.)
0001	Only T4 is enabled. (Function selected by FunSel will be applied to T4.)
0010	Only T3 is enabled. (Function selected by FunSel will be applied to T3.)
0011	T3 and T4 are enabled. (Function selected by FunSel will be applied to T3 and T4.)
0100	Only T2 is enabled. (Function selected by FunSel will be applied to T2.)
0101	T2 and T4 are enabled. (Function selected by FunSel will be applied to T2 and T4.)
0110	T2 and T3 are enabled. (Function selected by FunSel will be applied to T2 and T3.)
0111	T2, T3 and T4 are enabled. (Function selected by FunSel will be applied to T2, T3 and T4.)
1000	Only T1 is enabled. (Function selected by FunSel will be applied to T1.)
1001	T1 and T4 are enabled. (Function selected by FunSel will be applied to T1 and T4.)
1010	T1 and T3 are enabled. (Function selected by FunSel will be applied to T1 and T3.)
1011	T1, T3 and T4 are enabled. (Function selected by FunSel will be applied to T1, T3 and T4.)
1100	T1 and T2 are enabled. (Function selected by FunSel will be applied to T1 and T2.)
1101	T1, T2 and T4 are enabled. (Function selected by FunSel will be applied to T1, T2 and T4.)
1110	T1, T2 and T3 are enabled. (Function selected by FunSel will be applied to T1, T2 and T3.)
1111	All general purpose registers are enabled. (Function selected by FunSel will be applied to T1, T2, T3 and T4.)

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



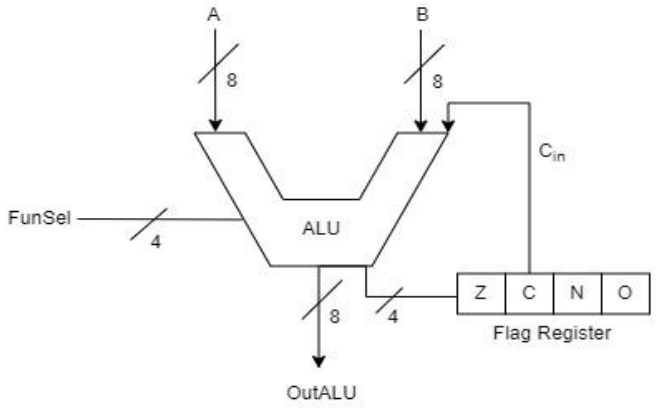
OutASel and **OutBSel** are used to feed output lines **OutA** and **OutB**, respectively. 8 bits of the selected registers are output to **OutA** and **OutB**. Table 5 shows selection of output registers based on the **OutASel** and **OutBSel** control inputs.

***Table 5**: OutASel and OutBSel controls*

OutASel	OutA	OutBSel	OutB
00	AR	00	AR
01	SP	01	SP
10	PCPrev	10	PCPrev
11	PC	11	PC

(Part-3) Design an Arithmetic Logic Unit (ALU) that has two 8-bit inputs, an 8-bit output, and a 4-bit output for **z**ero, **n**egative, **c**arry, and **o**verflow flags. The ALU is shown on the left side of Figure 10. 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 10:

- **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.
- **Arithmetic operations** are done using **2's complement** logic.
- **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	+	+	+	+
0110	Compare A, B	+	+	+	+
0111	A AND B	+	-	+	-
1000	A OR B	+	-	+	-
1001	A NAND B	+	-	+	-
1010	A XOR B	+	-	+	-
1011	LSL A	+	+	+	-
1100	LSR A	+	+	+	-
1101	ASL A	+	-	+	+
1110	ASR A	+	-	-	-
1111	CSR A	+	+	+	-

Figure 5: Graphic symbol of ALU and its characteristic table, respectively.

(**C**ircular | **A**rithmetic | **L**ogical) Shift (**L**eft | **R**ight) operations are depicted in Figures 6, 7 and 8.

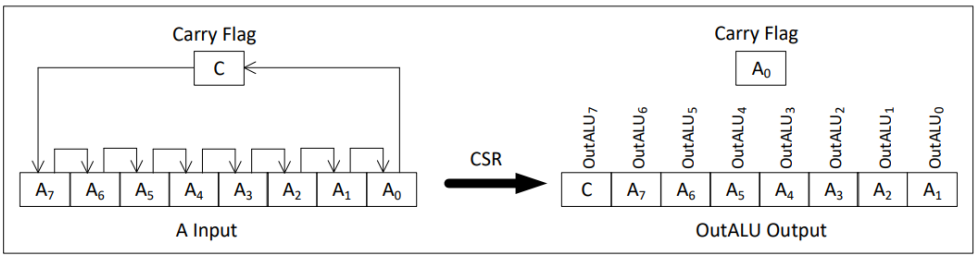


Figure 6: Circular Shift Operation

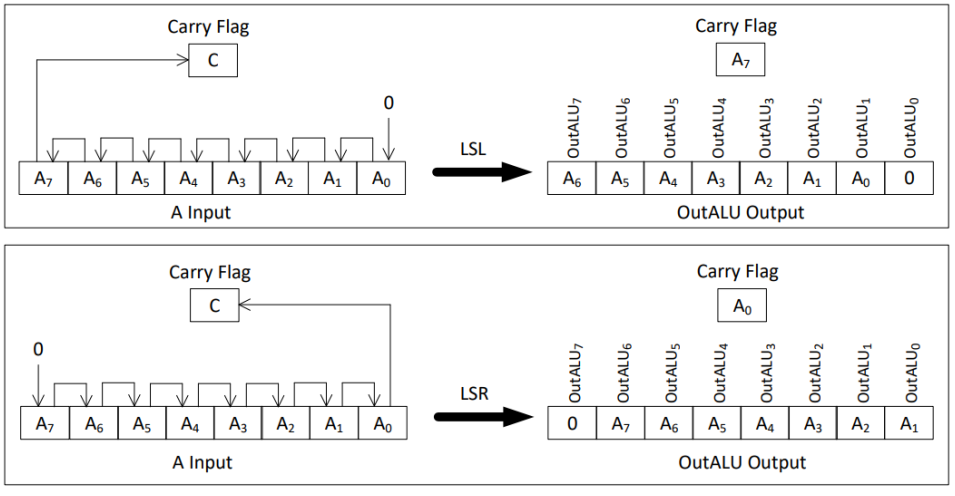


Figure 7: Logical Shift Operations

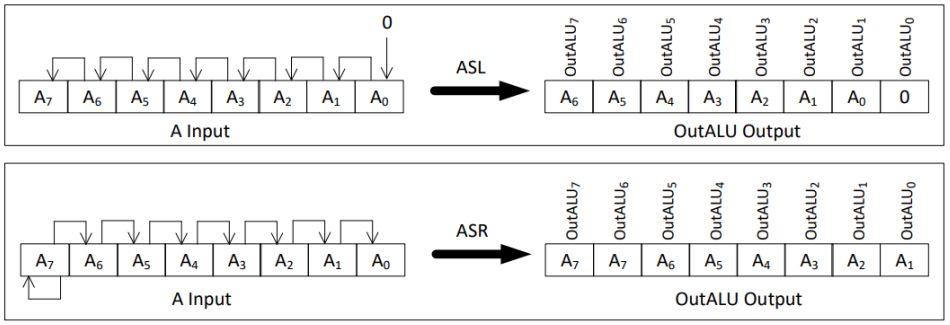
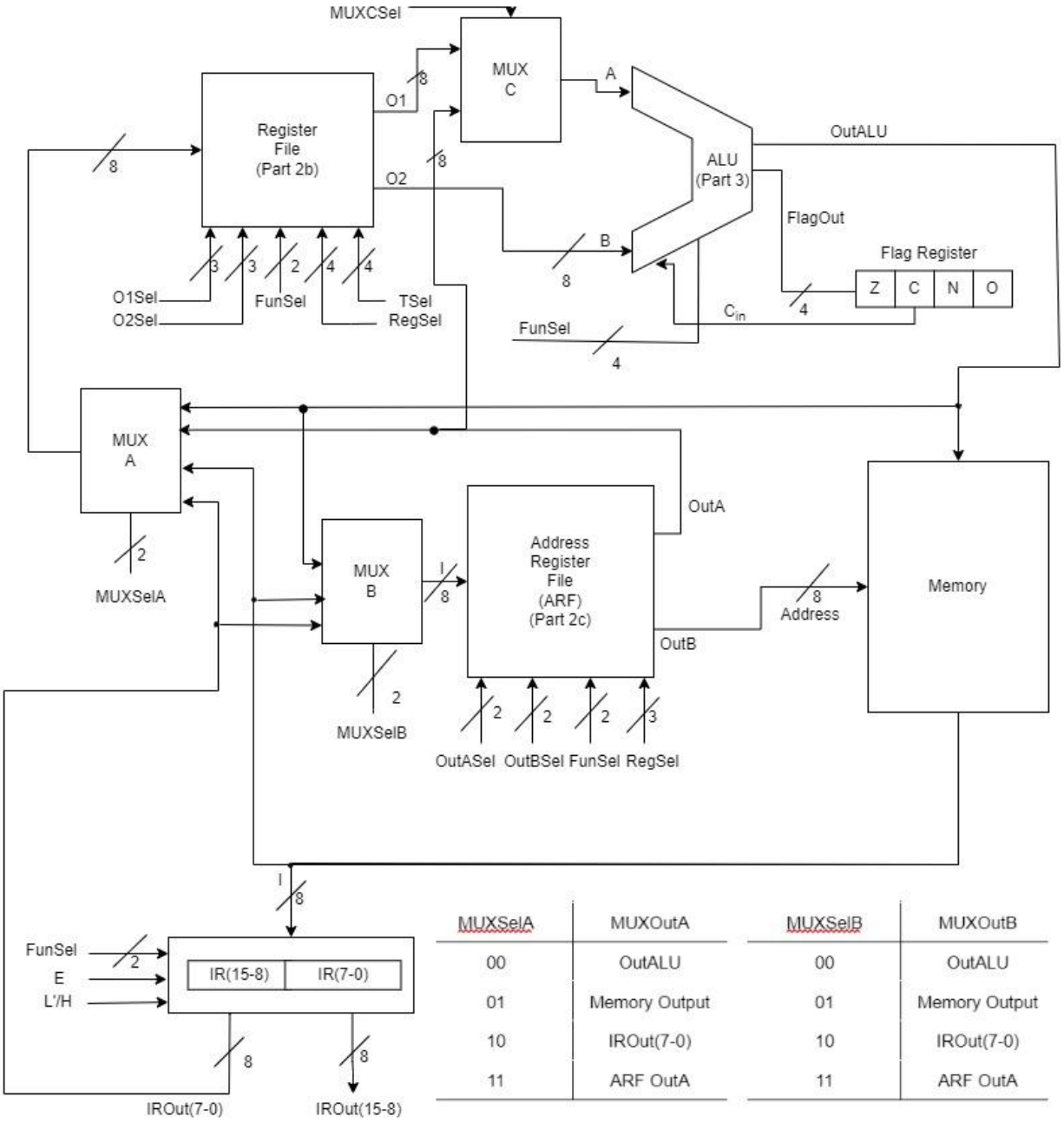


Figure 8: Arithmetic Shift Operations

(Part-4) Implement the organization in Figure 9. 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.)



MUXSelA	MUXOutA	MUXSelB	MUXOutB
00	OutALU	00	OutALU
01	Memory Output	01	Memory Output
10	IROut(7-0)	10	IROut(7-0)
11	ARF OutA	11	ARF OutA

Figure 9: ALU System.

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
- task distribution of each group member

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.