German University in Cairo
Media Engineering and Technology
Dr. Milad Ghantous

CSEN402: Computer Organization
Spring Term 2022

Project milestone 2- Deadline: June 9, 2022

You need to modify your basic computer developed in milestone 1 by:

A) Adding a PC register (you can implement this using a counter instead of a regular register since we need an increment input) with bus control 101, and adding the IR register with bus control 110.
Optional: A version of milestone 1 solution is uploaded to the CMS in case you weren't able to make it work during milestone 1. Use this version, if you want, to achieve milestone 2.

B) Adding the control unit (decoders, sequence counter and logic gates) necessary to implement the following program only. No need for it to implement all the instructions of the basic computer. Make sure that all unused control signals are equal to zero to ensure running a stable program.
Remember you need to control all the loads, increments, bus signals, ALU signals and so on for each Time T_x.

Program	Assembly

`int A, B, C, D; // variables`
 	
`if (A+B-C == 0)`
`{D = A && B;} //&& means ANDing`
`else`
`{D = A++;}`
	`ORG 0`
	`LDA A`
	`ADD B`
	`SUB C`
	`SZA`
	`BUN N1`
	`LDA B`
	`BUN N2`
	`N1, LDA A`
	`INC`
	`BUN N3`
	`N2, AND A`
	`N3, STA D`
	`A, DEC 2`
	`B, DEC 5`
	`C, DEC 7`
	`D, DEC 0`

The same instructions used in the lectures are used here with the addition of a new instruction called **SUB** which subtracts a value in the memory from the accumulator value (just like the ADD operation works). It uses the subtractor you added in milestone 1 to the ALU. Example: SUB M means AC ← AC – M
Since BSA is not used in this project, we will consider its binary code to be used for the SUB instruction and has the following code: 5XXX or DXXX (direct or indirect).
In this project, all addresses are direct.

You will need to either use or develop the control and timings for each instruction above. You might also need to add some hardware in addition to the ones you know in the lecture (decoders, counters...).
→ Try with different values of A, B, and C to test your program.

N.B. 1
Since our instruction format for the basic computer includes a 12-bit memory address, while the one developed in this project has only a 7-bit address, you can **either**:
1) Leave all instruction formats unchanged, but only feed the lower 7 bits of the address portion to the memory.
Or
2) Increase your memory size to 4096 words and adjust the size of AR and PC accordingly.

N.B. 2
Also ignore the interrupt. Just use the normal instruction cycle in everything.

Deliverables:

1) The circuit file in Logisim

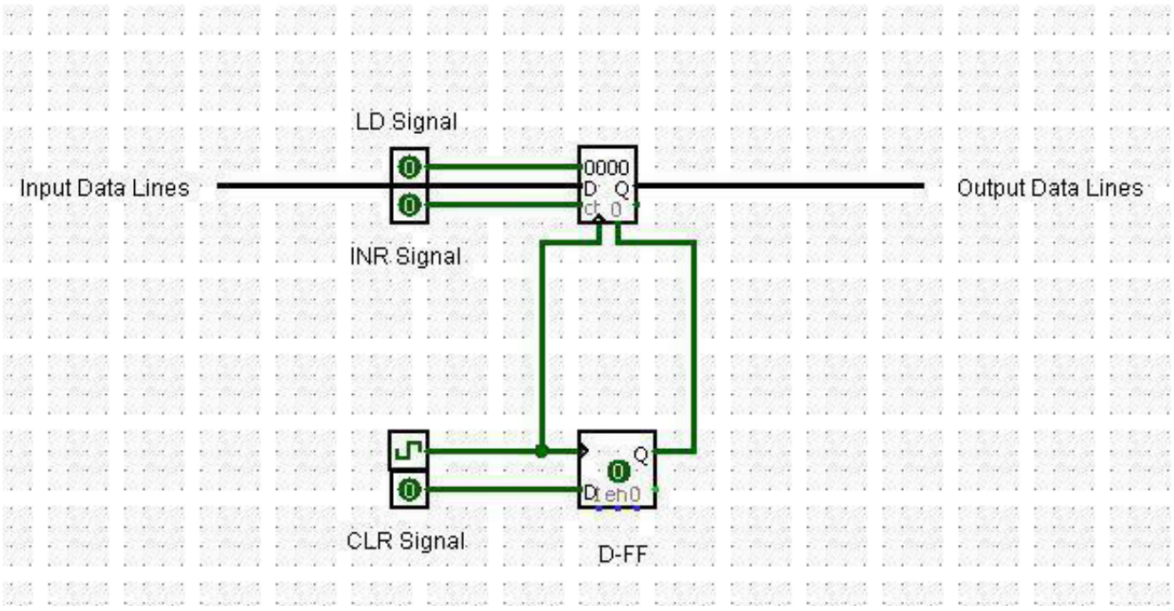
2) A small report containing each instruction you used with its timing (T0,T1,...) and control signals circuits that you developed for each register and component.

Due date and delivery:

June 9, 2022 email to co.spring22@gmail.com
Zip file, your group number, and all your names and emails in the email.

Extra notes

The clear signal for counter registers in Logisim is asynchronous, which means that once the signal becomes 1, the register clears without waiting for the next positive clock transition. To solve this, add a flip flop to the CLR signals of only the registers you will use. This flip flop will enforce synchronous signal receipt as it is only activated at the edge. (This connection is shown in the diagram below)



2. Use Tunnel from Wiring components to prevent having a lot of interconnected wires.
3. Edit the memory components by adding the machine code of the program after conversion.
4. To prevent clearing the memory contents each time you close the circuit file, save the contents as a .txt file by a right click on the memory.
5. Next time you open the circuit, right click on the circuit and load the pre-saved text file.