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

CSEN/CSIS402: Computer Organization and Systems programming
Spring Term 2024

Project milestone 2- Deadline: May 19, 2024

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

A) The PC register can be implemented using a counter instead of a regular register

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. Some edits may still be done by you.

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, bus signals, ALU signals and so on for each Time T_x.

Be careful that your registers do not have clear or increment so you must re-think the timing diagrams of each instruction. However, you can design your registers to have clear and increment.

Program	Assembly
	`ORG 0`
`int A, B, C, i; // variables`	`LOP, LDA A`
	`DIV B`
`for( i=0; i<3; i++)`	`ADD C`
`{`	`STA C`
`C += A/B;`	`LDA A`
`A += A * B;`	`MUL B`
	`ADD A`
`}`	`STA A`
	`ISZ i`
C= 8	`BUN LOP`
A = 32 + 16 = 48	`HLT`
C = 8 + 48/2 = 32	`A, DEC 16`
A = 48 + 96 = 144	`B, DEC 2`
C= 32 + 144/2 = 104	`C, DEC 0`
A = 144 + 288 = 432	`i, DEC -3`

The same instructions used in the lectures are used here with the addition of a new instructions called **MUL and DIV** which multiply or divide values in the memory with the accumulator value.

Examples:

MUL 98 means AC 🡐 AC * MEM[98]

DIV 44 means AC 🡐 AC / MEM[44]

Use the opcodes of BSA for MUL and the opcode of AND forDIV, since BSA and AND are not implemented or needed in this project.

The list of the instructions you need to work on:

`LDA, MUL, ADD, STA, DIV, ISZ, BUN`

You can ignore the HLT instruction.

You will need to 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. The above examples should give C= 104, A=432 and B=2 by the end of the 3rd iteration.

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

Deliverables:

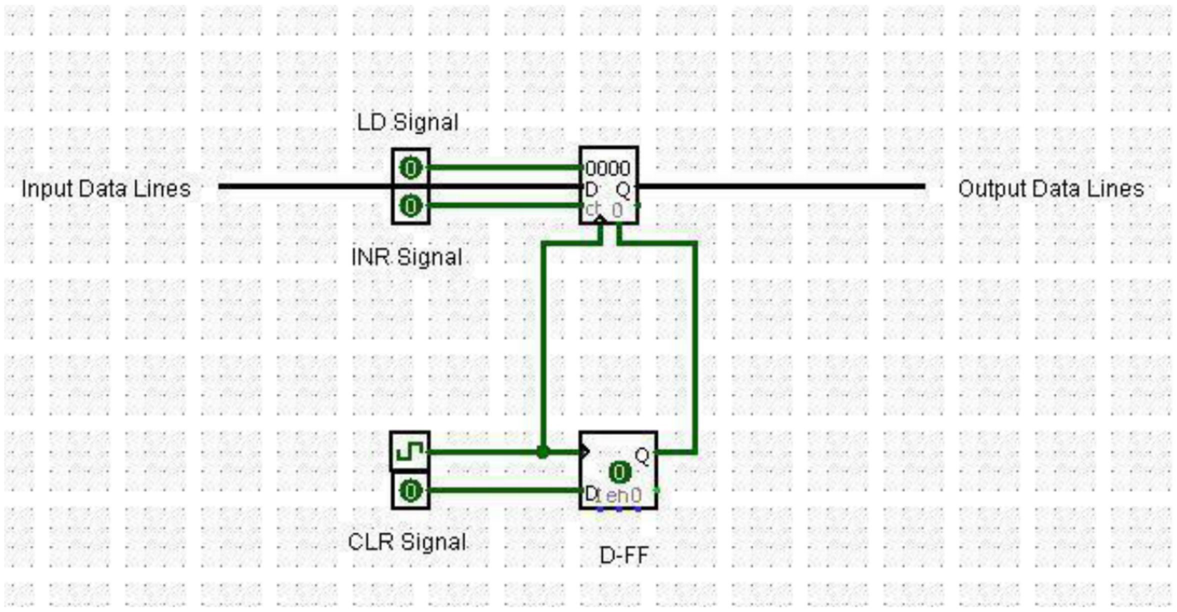
1) The circuit file in Logisim

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 and evaluation:

- Submission May 19 to **cospring2024@gmail.com**
- Evaluation: During revision week.
- Include Zip file, your group number, and all your names and student emails, as well as the report 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.