German University in Cairo
Faculty of Media Engineering and Technology
Dr. Mohamed ElMahdy

[CSEN402] **Computer Organization & System Programming,** Spring 2017
Milestone II Information
Deadline: Sunday, April 23rd, 2017 (11:59pm)

Description

- Use **Logisim** to implement the **combinational control circuit** for a program that multiplies two numbers as discussed in Lecture 8.(Sample program to be simulated can be found below)
- This requires analyzing and adding the control gates for the **registers**, the **memory**, and **E** for this program only.
- Build the full basic computer system needed to run this program by integrating the work of this milestone with the previous milestone. Please use the solution of Milestone I provided on the MET website.

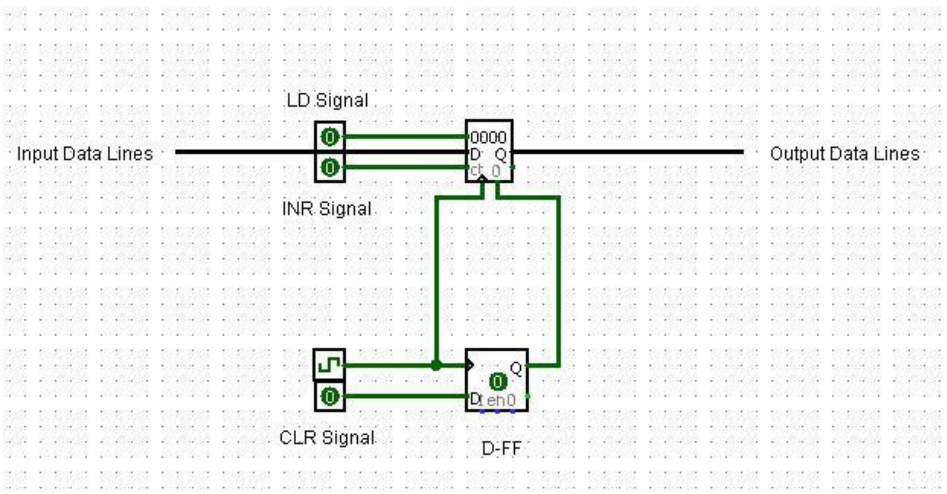
```
        ORG 0
        BSA MUL
        HEX 000F
        HEX 000B
        HEX 0000
        HLT
MUL,    HEX 0
        LDA MUL I
        STA Y
        ISZ MUL
        LDA MUL I
        STA X
        ISZ MUL
LOP,    CLE
        LDA Y
        CIR
        STA Y
        SZE
        BUN ONE
        BUN ZRO
ONE,    LDA X
        ADD P
        STA P
        CLE
ZRO,    LDA X
        CIL
        STA X
        ISZ CTR
        BUN LOP
        LDA P
        STA MUL I
        ISZ MUL
        BUN MUL I
CTR,    DEC -8
  X,    HEX 0000
  Y,    HEX 0000
  P,    HEX 0000
        END
```

Hints

1. Make sure that all unused control signals are equal to zero to ensure running a stable program.

2. For **HLT**, you need to add **S** flip flop. This flip flop is directly connected to the increment signal of **SC.** The **SC** should not get incremented once **S** is cleared.

3. 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)



4. Use **Tunnel** from **Wiring** components to prevent having a lot of interconnected wires.

5. After analyzing and connecting the corresponding control logic gates, convert the previous code to machine code. (Using Basic Computer Simulator posted on the MET website, or manually)

6. Edit the memory components by adding the machine code of the program after conversion.

7. 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.

8. Next time you open the circuit, right click on the circuit and load the pre-saved text file.

9. Your circuit file should include **3** circuits; main that includes the common bus system, ALU, and LCU that includes control logic gates.

10. You will need to add an encoder at the selection lines of the ALU

11. **<u>You can implement an additional part as a bonus:</u>** a program controlled I/O.

Submission Guidelines

- You are **not allowed** to submit more than one file. You have to submit a single **.circ** file that includes the complete circuit.

- The team representative is the one who is responsible for submitting the file in behalf of all team members.

- The circuit file should be sent to csen402ss2017@gmail.com as an attachment.

- The submission should be through any mail account except the **GUC mail** account in order to get an auto reply. You have to save this auto reply until your submission grade is posted, as this is the proof of your submission.

- **BOTH EMAIL SUBJECT** and **FILE NAME** must be in the format [TeamID][Milestone Code]. (i.e. **[003][M2]**). Otherwise, the file will be lost and all team members lose the milestone grade.

Appendix

Basic Computer Microoperations

Fetch	T_0:	AR ← PC
	T_1:	IR ← M[AR], PC ← PC + 1
Decode	T_2:	D0, ..., D7 ← Decode IR(12 ~ 14), AR ← IR(0 ~ 11), I ← IR(15)
Indirect	$D_7'IT_3$:	AR ← M[AR]
Memory-Reference		
AND	D_0T_4:	DR ← M[AR]
	D_0T_5:	AC ← AC $\wedge$ DR, SC ← 0
ADD	D_1T_4:	DR ← M[AR]
	D_1T_5:	AC ← AC + DR, E ← C_{out}, SC ← 0
LDA	D_2T_4:	DR ← M[AR]
	D_2T_5:	AC ← DR, SC ← 0
STA	D_3T_4:	M[AR] ← AC, SC ← 0
BUN	D_4T_4:	PC ← AR, SC ← 0
BSA	D_5T_4:	M[AR] ← PC, AR ← AR + 1
	D_5T_5:	PC ← AR, SC ← 0
ISZ	D_6T_4:	DR ← M[AR]
	D_6T_5:	DR ← DR + 1
	D_6T_6:	M[AR] ← DR, if(DR=0) then (PC ← PC + 1), SC ← 0
Register-Reference ($D_7I'T_3$ = r, IR(i) = B_i)		
	r:	SC ← 0
CLA	rB_{11}:	AC ← 0
CLE	rB_{10}:	E ← 0
CMA	rB_9:	AC ← AC′
CME	rB_8:	E ← E′
CIR	rB_7:	AC ← shr AC, AC(15) ← E, E ← AC(0)
CIL	rB_6:	AC ← shl AC, AC(0) ← E, E ← AC(15)
INC	rB_5:	AC ← AC + 1, E ← C_{out}
SPA	rB_4:	If(AC(15) =0) then (PC ← PC + 1)
SNA	rB_3:	If(AC(15) =1) then (PC ← PC + 1)
SZA	rB_2:	If(AC = 0) then (PC ← PC + 1)
SZE	rB_1:	If(E=0) then (PC ← PC + 1)
HLT	rB_0:	S ← 0