# Digital Systems Design – EE203

## Simple Multi-Cycle Central Processing Unit Design

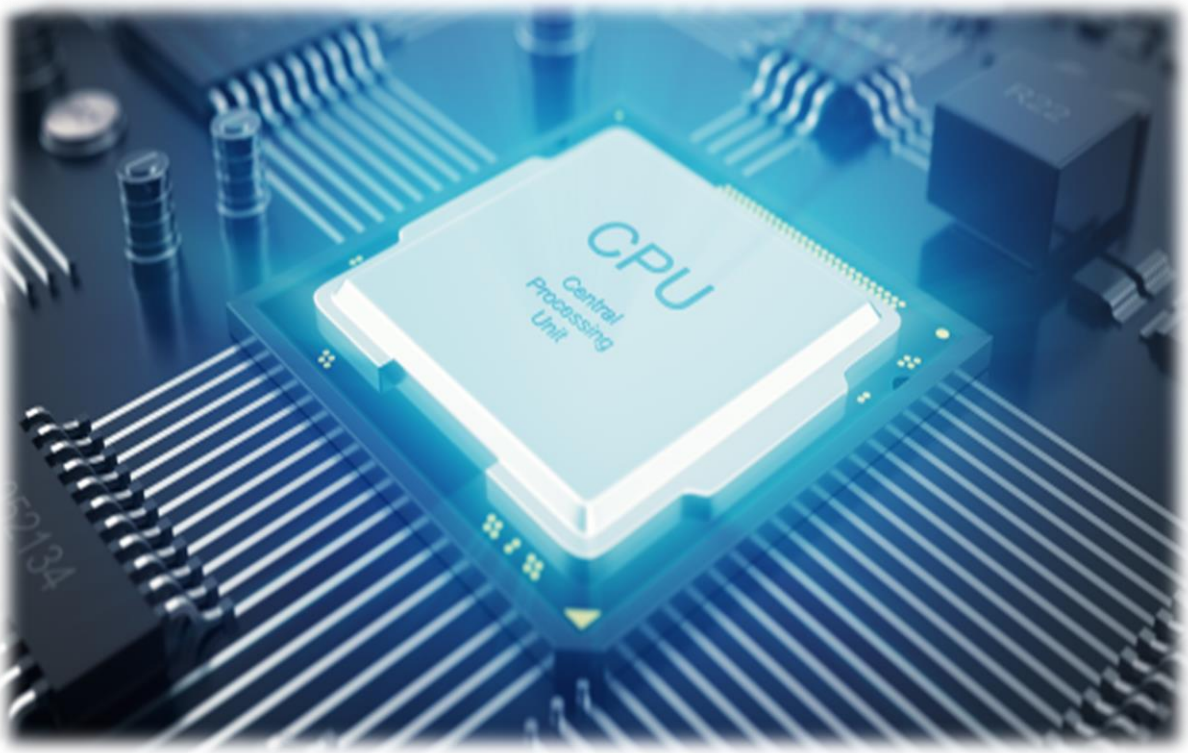
Muhammed Rahmetullah Kartal – 041701008

Ramazan Yetişmiş – 041701013

Instructor: Şuayb Ş. Arslan

Instution: MEF University

Date: Oct 17, 2019



# Introduction

In the Project firstly, we read the Project description carefully and then begin to work. Firstly, we implement our 8bit-ALU.Then we implement our control logic. In the control logic there are bunch of outputs which determine the operations due to the 8-bit DIN. In the logic there is 8-bit input which has the instruction in itself. By the Project instructions we understood that we need RAM. Then we figure out how to run our Instructions by using Ram. We cope with that situation by using Program counter to lead the instructions in RAM. Overall our CPU has 3 cycle and the shortest operations are Move operations (1 clock cycle) and the longest ones are Sum and Sub (3 clock cycle).

- **Objective**

These projects main goal is to teach us how to design and implement a simple multi-cycle central processing unit (CPU). In the Project we made various operations with using a great deal of registers and 8-bit BUS by the help of them we can manage to make bunch of arithmetic operations.

# Questions and Demonstration:

*Question 2.1:*

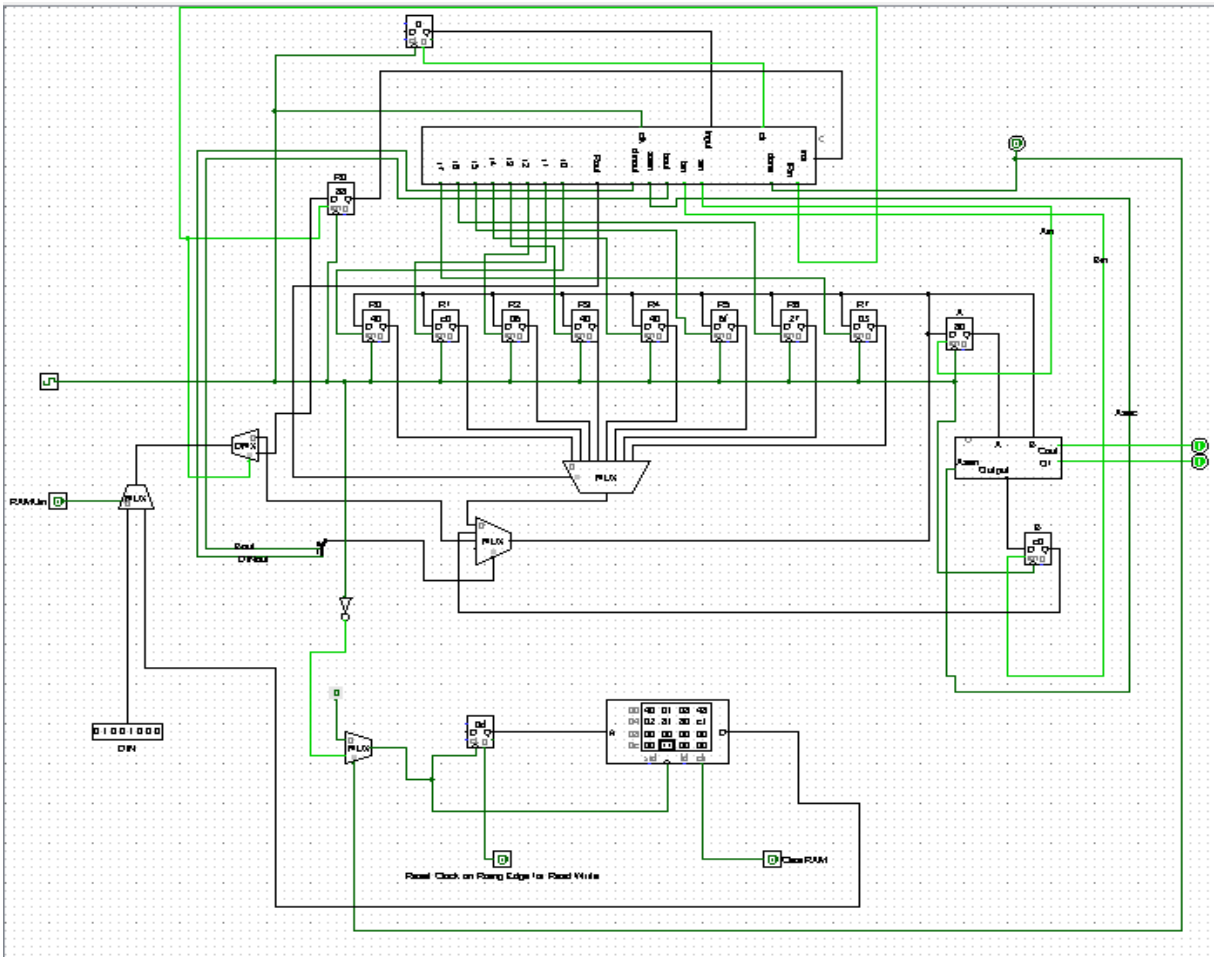


*Figure 1: General Structure of Datapath and RAM*

- **Datapath**

The Datapath have 8 8-bit registers by using the outputs of Control Logic Circuit each of these registers are becomes writable (by enabling them). By an 8:1 MUX we can select the Rout register, for the add and sub operations we need to use A and B registers on different clock cycles. We can select the Rout's value, B register's value or DIN input by a 4:1 MUX. By the way, we can determine the T0 T1 and T2 times with a mod 3 2-bit counter if counter's output is 00 the time is on T0, if it is 01 the time is on T1, if it is 10 the time is on T2. To enable IRin in right time we checked the Done output of Control Unit in a DEMUX (when it becomes 1, we are enabling the IR and cleaning the value of 2-bit counter to restart counting.). At last, by the RAMUin input we can get instructions from RAM or DIN.

- **Arithmetic Logic Unit (ALU)**

The ALU is for subtraction and addition operations it takes inputs from A register and B register, we can select the operation of ALU by Asec output of Control Logic Circuit, if its 1 the operation will be subtraction else the operation will be addition.

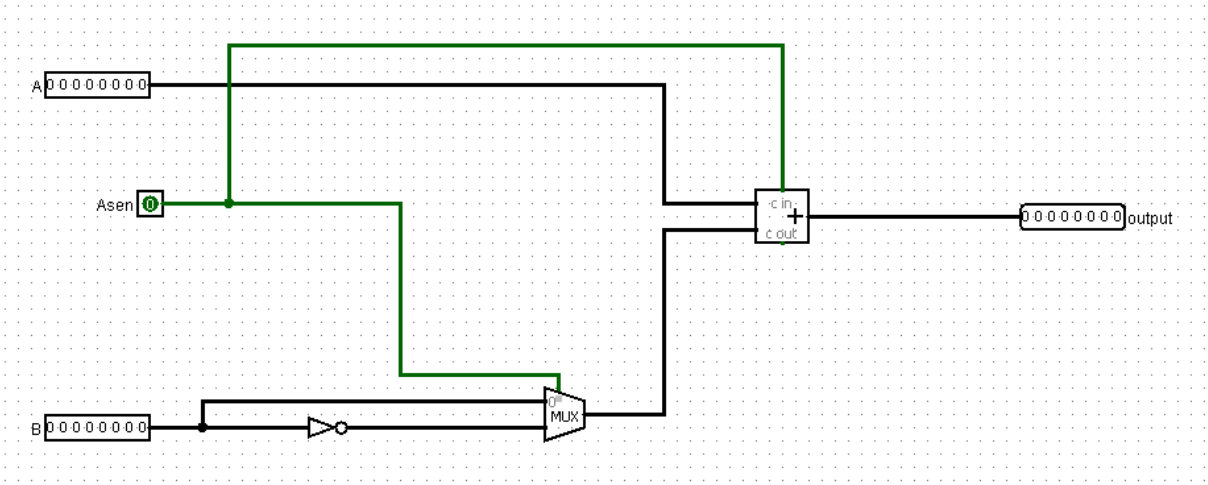



*Figure 2: ALU without overflow and Underflow*

*Question 2.2:*

*Table 1: Truth Table for Control Circuit*

| OP[0] | OP[1] | T0 | T1 | T2 | Done | Rxin | Rxout | Ryout | Ain | Bin | Bout | Asen | DINout |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | X | X | X | 1 | 1 | 0 | 1 | X | X | 0 | X | 0 |
| 0 | 1 | X | X | X | 1 | 1 | 0 | 0 | X | X | 0 | X | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | X | 0 | X | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | X | 0 | 1 | X | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | X | 0 | X | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | X | 0 | 1 | X | 0 |

This truth table is created by using the Table 2, after that we used that table for creating the Control Circuit. The inputs of that circuit are the instruction and outputs of a 2-bit counter by the instruction code we get X register Y register and the operation that we are going to do, by putting the outputs of 2-bit register to a decoder we selected the cycle times ($T_0$, $T_1$, $T_2$).

*Table 2:The set of control signals asserted for each instruction and time step*

| Instruction Type | | $T_0$ | $T_1$ | $T_2$ |
|---|---|---|---|---|
| *mv* | *IRin* | *Rxin, Ryout, Done* | | |
| *mvi* | *IRin* | *Rxin, DINout, Done* | | |
| *add* | *IRin* | *Ain, Rxout* | *Bin, Ryout* | *Rxin, Bout, Done* |
| *sub* | *IRin* | *Ain, Rxout* | *Bin, Ryout, Asen* | *Rxin, Bout, Done* |

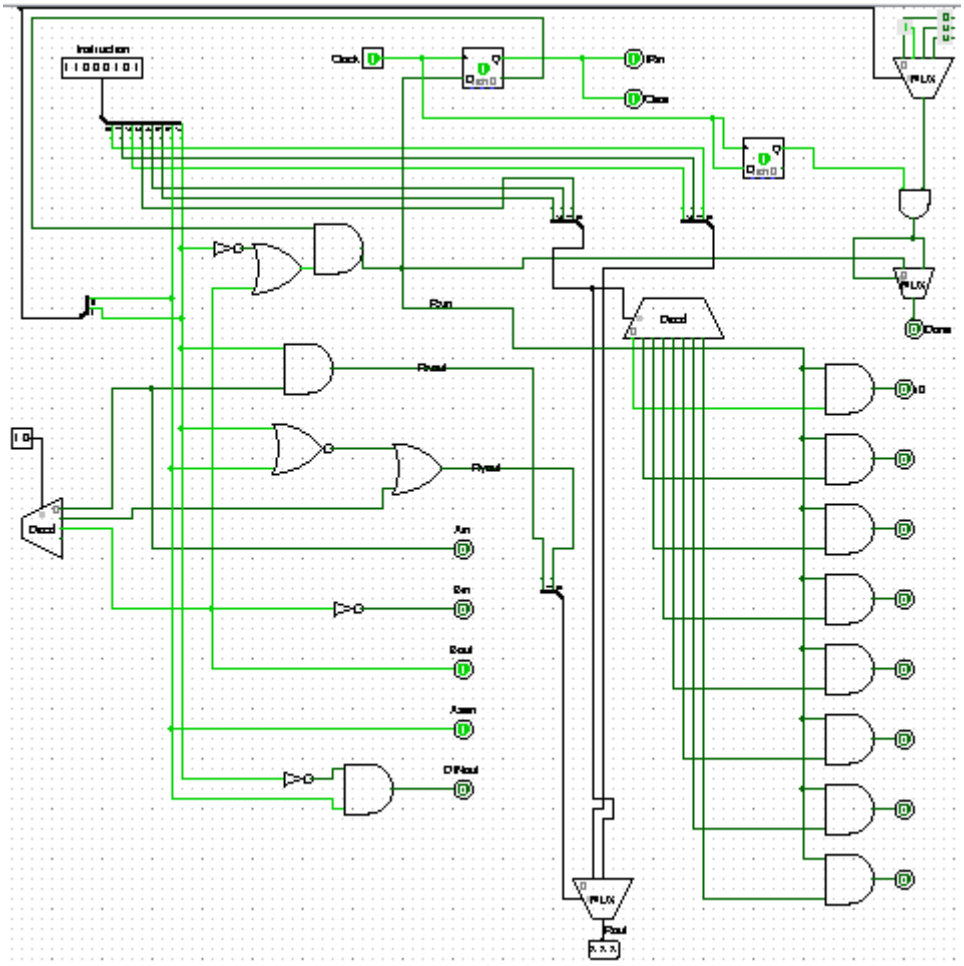



*Figure 3: Control Logic Circuit*

To determine Instruction Register's enable we hold the value of Rxin/Done (They have same values) output on a D-FF, the purpose of that is clearing the 2-bit counter's value after the done operation. On the RAM usage stage, we faced with a problem, the problem was getting the Data in value on mvi operation, we overcome this by holding the clock signals in a register for creating delay and ANDed that delayed clock by the 01 situation of operations code, so we set the Done output's value on every instruction getting stage after mvi operation(That mvi-done operation is just for using RAM).

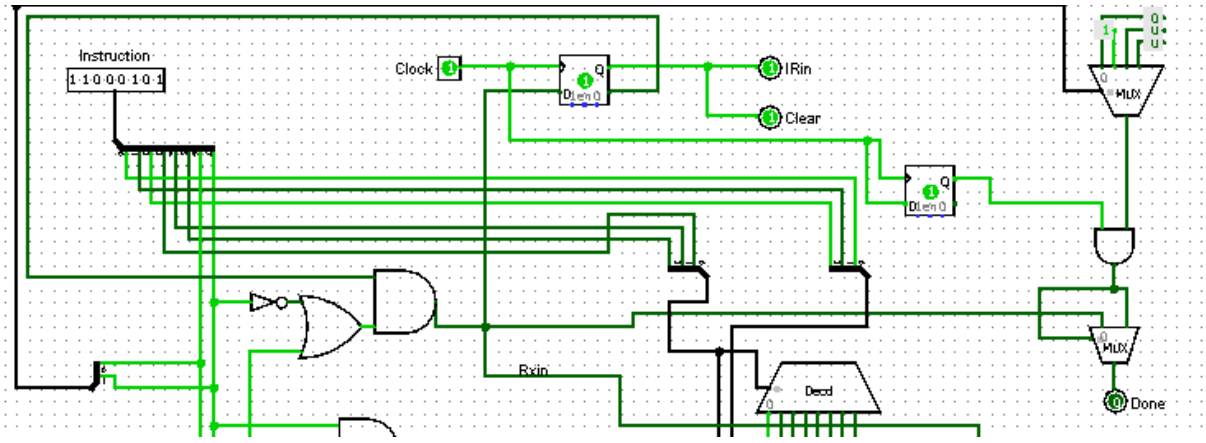



*Figure 4: Delayed clock, normal and mvi types of Done operation.*

To select Rout register we compared Rxout and Ryout inputs by the opcode of instruction.

*Question 2.3:*

In the bonus part we split our 8-bit inputs individually into 2 parts. First part includes 7 bit of the input and second part 1 bit. We added 2 7-bit with 7-bit adder and took the sum of it and then later we added 1-bit inputs with 1-bit adder and took the carry out of it. Finally, we put 2 carry outs into XOR gate to determine under/overflow. Over/under flow means the number we get from the operations has more bits than we have.

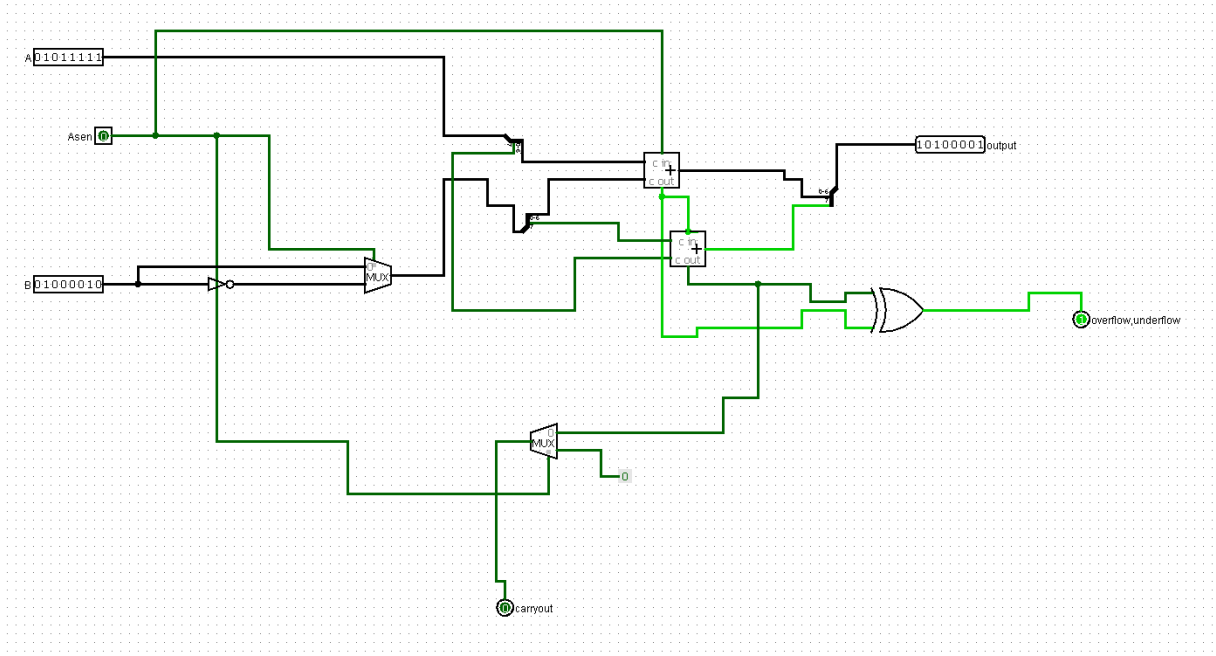



*Figure 5: After implementing the overflow and carryout functionality*

*Question 3.1:*

1)We choose to demonstrate the move immediate operation. Firstly, we give the instruction that can be simply seen in red colour in the figure “01101000” means that write data in to R5.

The figure shows the moment after the first rising edge.

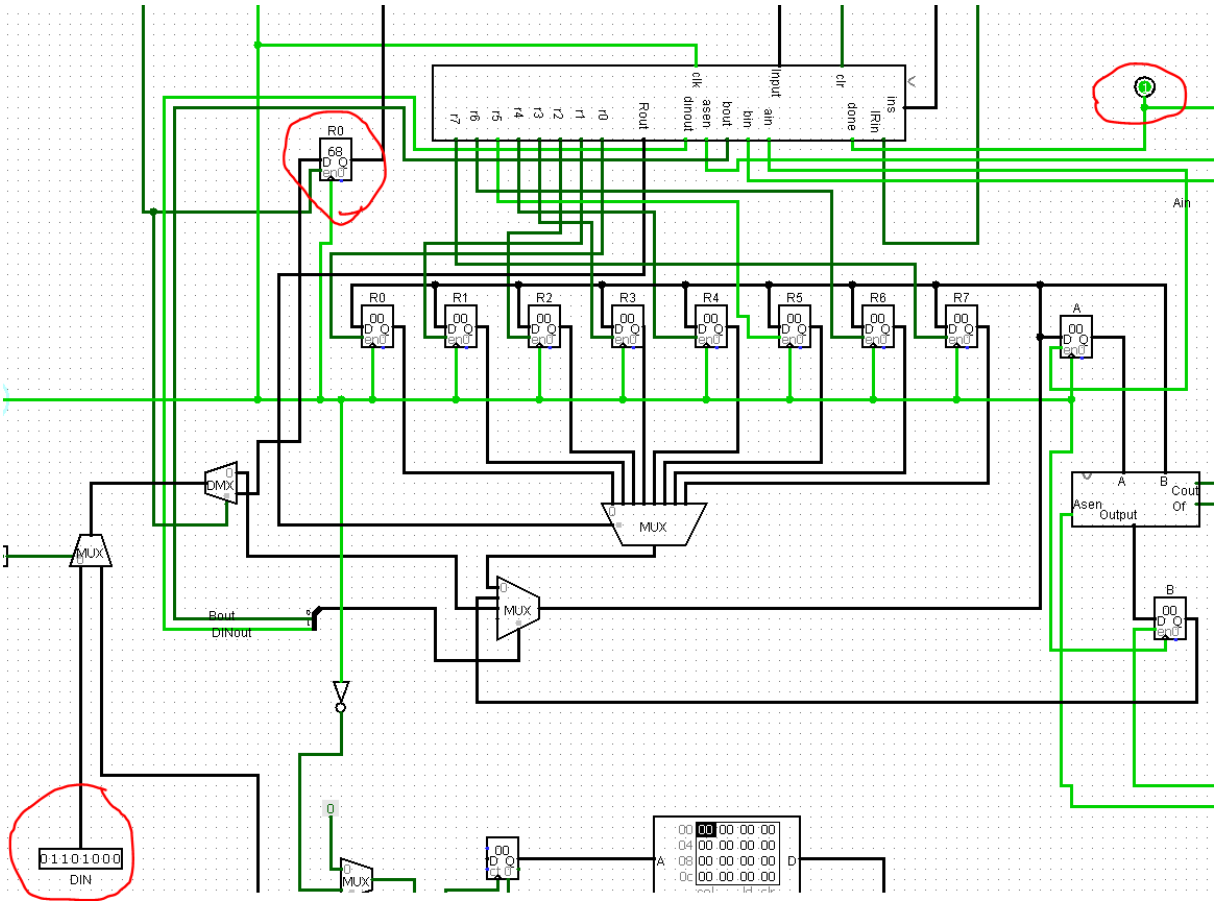



*Figure 6: The first rising edge part of the MVI*

2) After the rising edge by the help of the control unit CPU knows that this is a move immediate operation thus the next data is the number which We are going to write to the R5. Also, there can be understood by looking in the figure that the Instruction register's value is the same "instruction value(hex)68" but the number we gave as data is "0xB8" thus it did not write in the IR, it has written into R5 directly because IR enable was off because when by the help of the control unit IR knows after MVI operations the next data is not an instruction. Addition to this our Done is on in two steps.

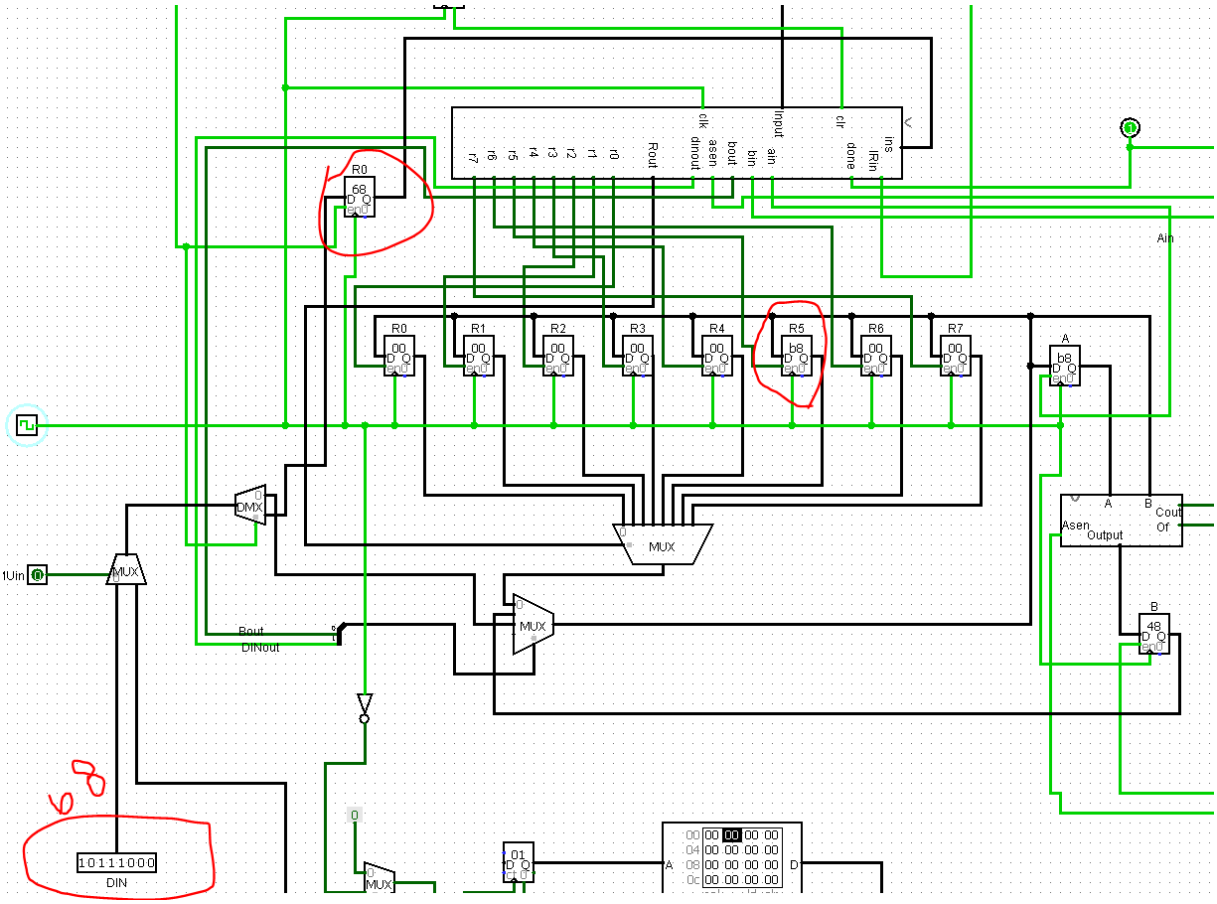



*Figure 7: Last step of the MVI operation*

*Question 3.2:*

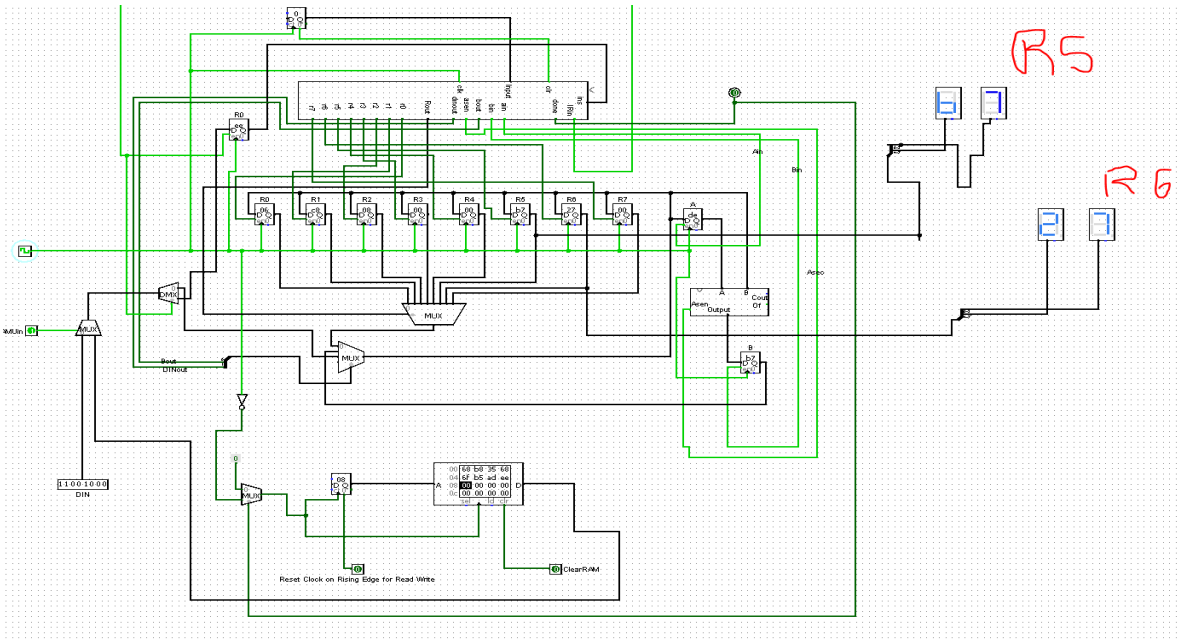



*Figure 7: R5 and R6 after operations*

For the question 3.2 we did all that operations which is in Figure 8.

$$
\begin{aligned}
&mvi\ R5, 0\text{xB8} \mid R5 \leftarrow 0\text{xB8}, && (1)\\
&mv\ R6, R5 \mid R6 \leftarrow [R5], && (2)\\
&mvi\ R5, 0\text{x6F} \mid R5 \leftarrow 0\text{x6F}, && (3)\\
&add\ R6, R5 \mid R6 \leftarrow R6 + [R5], && (4)\\
&add\ R5, R5 \mid R5 \leftarrow R5 + [R5], && (5)\\
&sub\ R5, R6 \mid R5 \leftarrow R5 - [R6] && (6)
\end{aligned}
$$

*Figure 8: Operations*

*Question 3.3:*

*RA is R0*
*RB is R1*
*RC is R2*
*another Register is for 1 is R7*

*01001000 datain:1F (mvi for B)*
*1 cycle for instruction*
*1 cycle for operation*

*00010000 (mv for a to c)*
*1 cycle for instruction*
*1 cycle for operation*

*11000001 (sub for a-b to a)*
*1 cycle for instruction*
*3cycle for operation*

*11000010 (sub for a-c to a)*
*1 cycle for instruction*
*3 cycle for operation*

*01111000 datain:01 (mvi 1 for using in -1 operation)*
*1 cycle for instruction*
*1 cycle for operation*

*11000111 (sub a-1 to a)*
*1 cycle for instruction*
*3 cycle for operation*

*A registers last value: e0*
*B registers last value: 1f*
*C registers last value: 00*

This process takes 16 clock cycle it gets the ones complement of the Rb -1 and writes it to Ra but holds the first value of Ra in Rc. We can do it just with a not gate, mv and sub immediate instruction in a lot less clock cycles.