EE 203 Digital Systems Design

Term Project Report

22.01.2024

Team 40

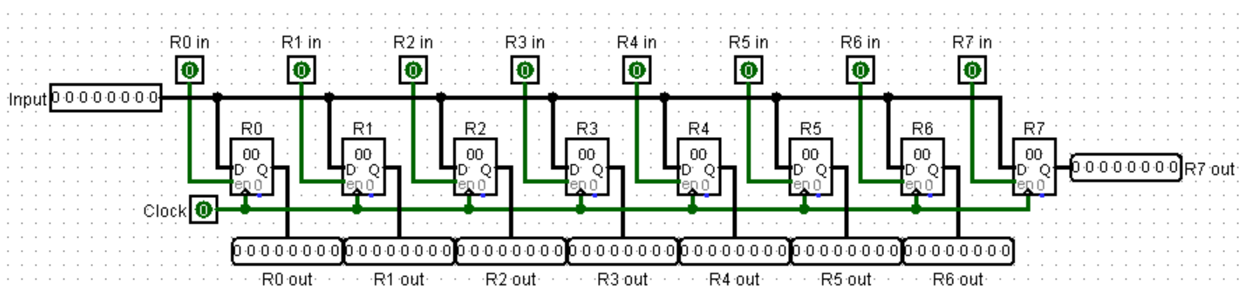
Tevfik Tarık Alim / 042001033
Abdullah Kiraz / 042201083
Kaan Edip Özoğuz / 041801091

1. Introduction

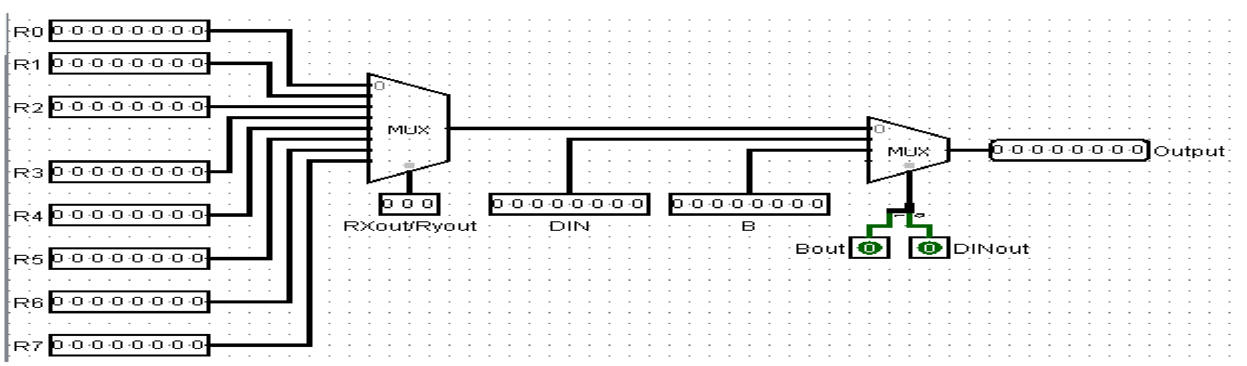
In this project, We designed a simple multi-cycle 8-bit central processing unit. We have 8 registers that can store 8-bit data, an ALU which can do addition or subtraction, a Control Unit which can manage 4 operations which are mv, mvi, add and sub. In addition We have a simple counter which can give us the time step information and a ROM for storing the instructions and executing them quickly.

2. Implementation of Circuit

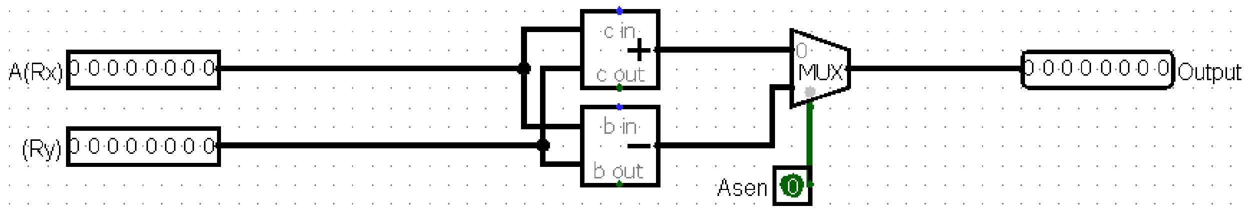
In circuit design, subcircuits are employed to create a more streamlined structure. These subcircuits are ultimately integrated over the datapath, culminating in the formation of a comprehensive CPU architecture. The designs include Counter (default in Logisim) and subcircuits such as, Multiplexers (MUX), Arithmetic Logic Units (ALU), Registers, and Control units, leading to the final representation of the overall CPU structure. Logisim, a digital logic simulator, has been used for these circuit designs.



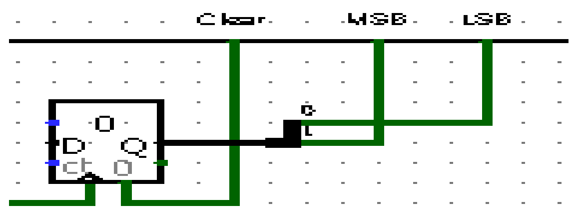
Register Subcircuit



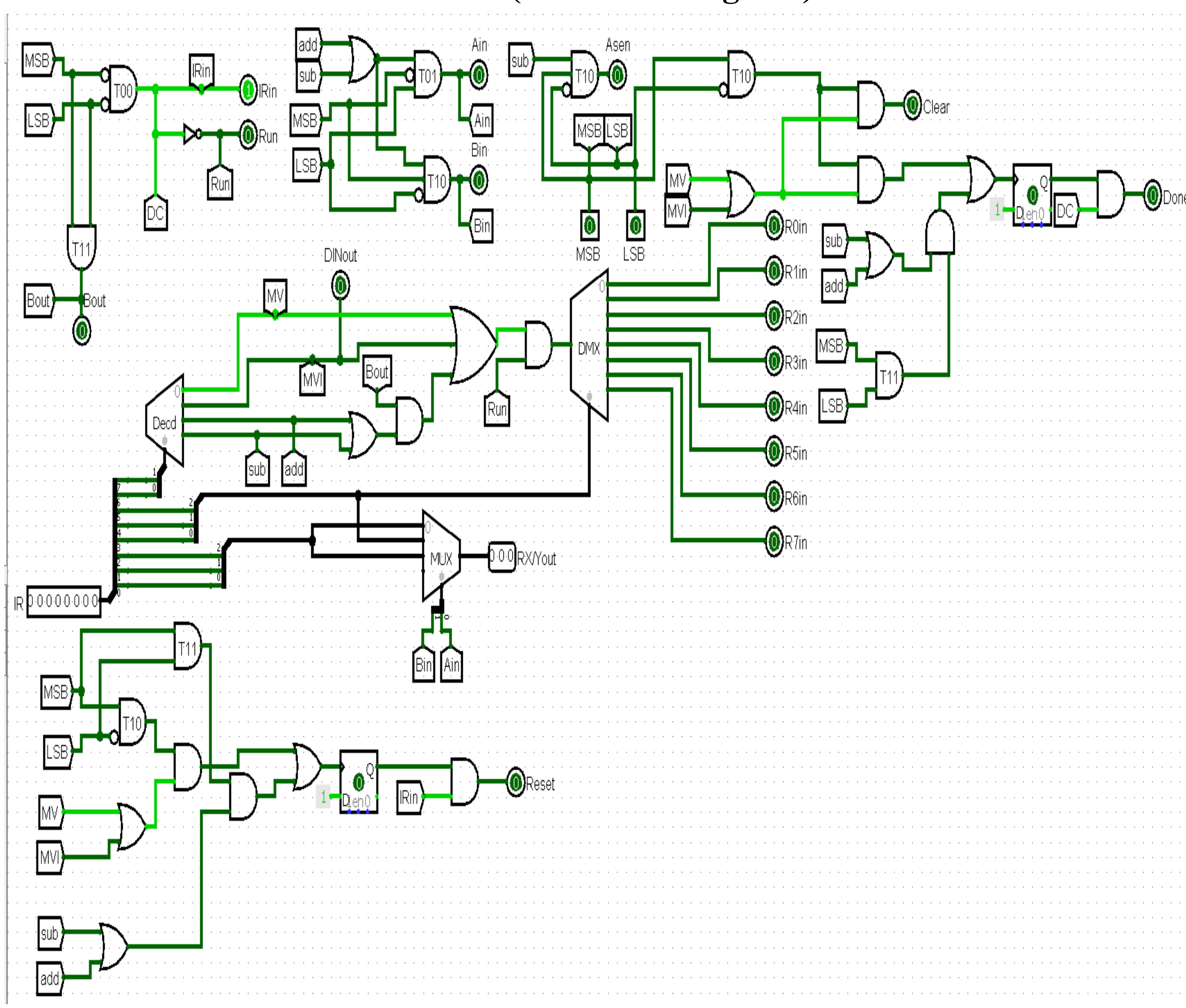
Multiplexer Subcircuit



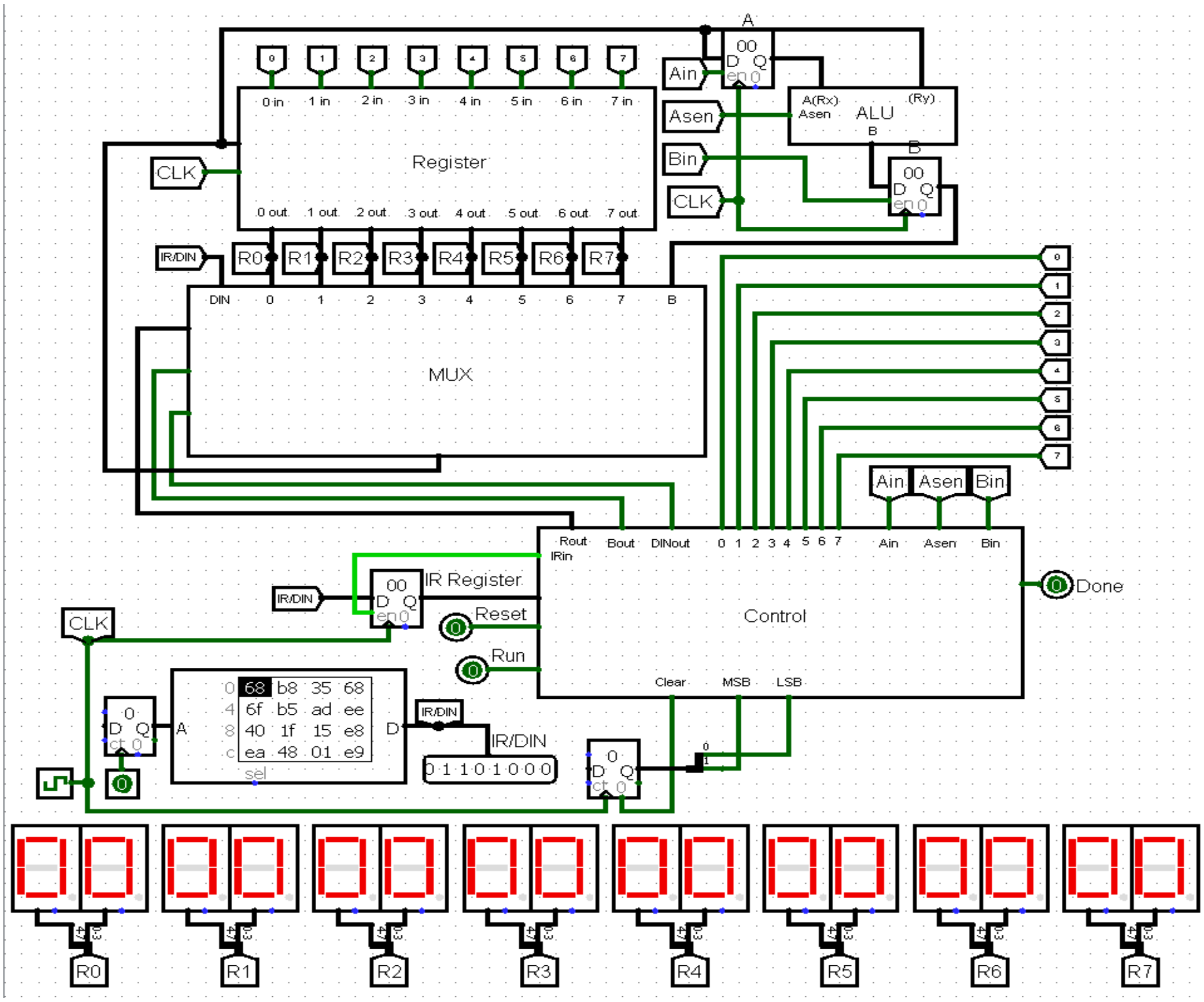
ALU Subcircuit



Counter (Default in Logisim)



Control Subcircuit



Overall CPU (Datapath)

Type	OP0	OP1	T0	T1	T2	Done	RXin	RXout	RYout	Ain	Bin	Bout	Asen	DINout
mv	0	0	1	0	0	1	1	0	1	0	0	0	0	0
mvi	0	1	1	0	0	1	1	0	0	0	0	0	0	1
add	1	0	1	0	0	0	0	1	0	1	0	0	0	0
add	1	0	0	1	0	0	0	0	1	0	1	0	0	0
add	1	0	0	0	1	1	1	0	0	0	0	1	0	0
sub	1	1	1	0	0	0	0	1	0	1	0	0	0	0
sub	1	1	0	1	0	0	0	0	1	0	1	0	1	0
sub	1	1	1	0	1	1	1	0	0	0	0	1	0	0

Truth Table for Control Subcircuit

Table 1: The instructions of the processor

OP code	Operation and Operands	Executed Function
00	$mv\ Rx, Ry$	$Rx \leftarrow [Ry]$
01	$mvi\ Rx, D$	$Rx \leftarrow D$
10	$add\ Rx, Ry$	$Rx \leftarrow [Rx] + [Ry]$
11	$sub\ Rx, Ry$	$Rx \leftarrow [Rx] - [Ry]$

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, $Ryout$, $Asen$	$Rxin$, $Bout$, $Done$

ISA Tables for CPU

3. Demonstration

3.1

Instruction	IR	DIN	Machine Code for IR	Machine Code for DIN
mvi R5, 0xB8 | R5 ← 0xB8	01101000	10111000	0x68	0xb8
mv R6, R5 | R6 ← [R5]	00110101	-	0x35	-
mvi R5, 0x6F | R5 ← 0x6F,	01101000	01101111	0x68	0x6f
add R6, R5 | R6 ← R6 + [R5]	10110101	-	0xb5	-
add R5, R5 | R5 ← R5 + [R5]	10101101	-	0xad	-
sub R5, R6 | R5 ← R5 − [R6]	11101110	-	0xee	-

Instructions Table and Machine Codes for ROM

As seen in the instruction table, the processes we need to implement during the demo have been presented through binary conversions. In accordance with what is requested from us in the handout, if the operation to be performed is an 'mvi' operation, the input in hexadecimal format to be used as DIN has been converted into the appropriate binary code. In addition, we have been asked to use ROM. The ROM has allowed us to record these instructions and perform the operations quickly during the demo. To accurately store the instructions in the ROM, the binary codes have been converted into suitable machine codes and presented in the table.

3.2

At the end of the instructions, R5 contains $(B7)_{16}$ and R6 contains $(27)_{16}$. We used a hex digit display to see content of the registers as asked from us in the handout.

3.3

$$
\begin{aligned}
Rb &\longleftarrow 0x1F \\
Rc &\longleftarrow [Ra] \\
Ra &\longleftarrow Ra - [Rb] \\
Ra &\longleftarrow Ra - [Rc] \\
Ra &\longleftarrow Ra - 1
\end{aligned}
$$

Instructions for Part 3.3

Instruction	IR	DIN	Machine Code for IR	Machine Code for DIN
mvi R0, 0x1f | R0 $\leftarrow$ 0x1f	01000000	00011111	0x40	0x1f
mv R2, R5 | R2 $\leftarrow$ [R5]	00010101	-	0x15	-
sub R5, R0 | R5 $\leftarrow$ R5 - [R0]	11101000	-	0xe8	-
sub R5, R2 | R5 $\leftarrow$ R5 - [R2]	11101010	-	0xea	-
mvi R1, 0x01 | R1 $\leftarrow$ 0x01	01001000	00000001	0x48	0x01
sub R5, R1 | R5 $\leftarrow$ R5 $-$ [R1]	11101001	-	0xe9	-

Instructions Table for Part 3.3 and Machine Codes for ROM

At this stage, we have been asked to select our preferred registers for the given operations and execute them. Since our ALU subcircuit does not have the capability to perform the -1 operation, to execute the final operation, we first insert '1' into one of our registers using DIN from step 3.2 of the previous stage. Subsequently, we perform a subtraction operation between two registers. For this process, registers R5, R0, and R2 have been assigned as Ra, Rb, and Rc, respectively. At the end of these operations, register Ra contains the value 0xe0, Rb contains 0x1f, and Rc contains 0x67. The completion of all these operations took 18 clock cycles. To quickly execute these operations, as requested, we have stored these instructions in the ROM. This approach streamlines the process, enabling the system to rapidly access and execute the instructions during operation, leveraging the efficiency and speed of ROM for immediate retrieval and implementation

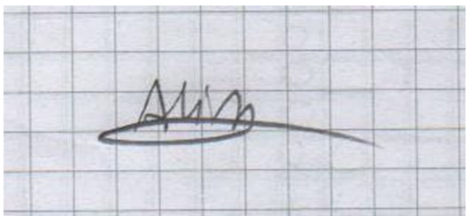
4. Discussion

As observed, our CPU successfully executes operations using the Instruction Set Architecture (ISA) designed for it. The control unit plays a pivotal role at this point, overseeing the overall operation of all components. Addressing one of the noticeable shortcomings in the design, a deficiency in the Counter's reset system becomes apparent. For 'mv' and 'mvi' operations, a reset process is implemented in our Counter at the appropriate stage. With the design added to our Logic subcircuit, since the 'add' and 'sub' operations are completed within 3 clock cycles, there's no need to add an extra reset for these operations in this circuit design. This is because, after the 3rd cycle, the overflow will automatically result in the Most Significant Bit (MSB) and Least Significant Bit (LSB) being 00. In this design, since we do not require more than 3 clock cycles to perform an operation, the counter effectively fulfills its role. However, for more advanced operations that require more clock cycles, more complex designs will be necessary for counter resetting. As mentioned, for the instructions given to us, this design successfully fulfills its intended function.

Labor of Distribution

In this project, Tevfik Tarık Alim took on the roles of circuit design and the design of the Control unit. Abdullah Kiraz also contributed to the designs alongside Tevfik Tarık Alim. At the final stages of the designs, Kaan Edip Özoğuz resolved the existing problems. This collaborative effort resulted in the completion of a comprehensive circuit design.

Tevfik Tarık Alim / 042001033



Abdullah Kiraz / 042201083

Kaan Edip Özoğuz / 041801091