4-Bit Processor Using Logisim

Acronyms

Control Unit – CU

Data Memory – DM

Instruction Memory – IM

Arithmetic and Logic Unit – ALU

Program Counter – PC

Program Counter Increment Enable – PCIE

Program Counter Enable – PCE

Accumulator – ACC

Output Register – OR

Introduction

This project demonstrates the working of a simple 4-bit processor; The processor was simulated in Logisim. A processor consists of three fundamental components – CU, DM, and ALU. The data memory and the control unit are combined together in the architecture of this processor as its size is only $2^4 = 16$ bits long. The following sections describe the architecture and functionality of each of the components in detail.

Architecture and Functionality

Start Block

This block comprises a self-starting shift register that is responsible for starting the processor. Here, the word "self-starting" indicates that the register starts as soon as the simulation starts i.e. the clock begins ticking. It is analogous to the "power button" on any electronic device. Strictly speaking, this block is not a part of the processor but rather a simulation only block. Fig 1 shows the implementation of this block. Note that the combinatorial logic that sets the PCIE flag and PCE flags simultaneously. The PC is enabled only when it's time to increment it thus both, PCIE and PCE are set simultaneously.

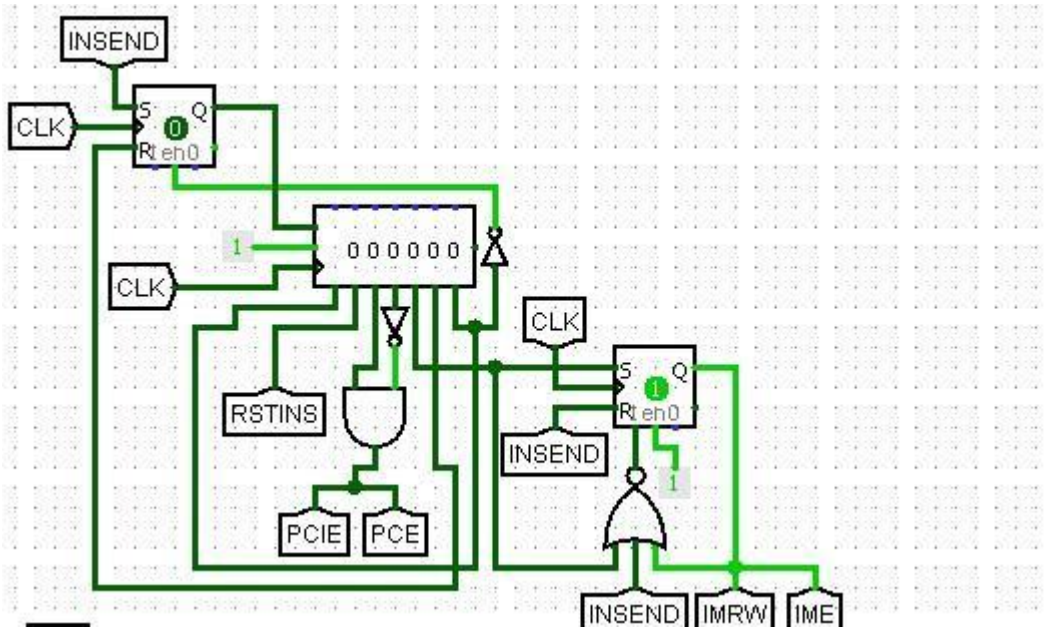

Figure 1: Start Block

Control Unit

The CU is the heart of any processor (overview shown in Fig 2). It is fundamentally responsible to run the current instruction and increment the PC which stores the address of the next instruction. The CU comprises of the PC and PCI, DM, IM, temporary register bank, ACC, and instruction decoder for the ALU (shown in Fig 3B).

As shown in Fig 3A, the 4-bit adder is used to increment the PC register which is subsequently connected to the IM. The instruction size is 10-bits and format of instruction used in this processor as well as the list of instructions are shown in Fig 4 and Fig 5 respectively. The first four bits (from MSB) of the 10-bit long instruction are the selector bits for the instruction decoder to select one of the operations listed in Fig 5. The instruction list is derived from the MIPS instruction set.

As the name suggests, the temporary register bank consists of four temporary registers (of data size 4-bits) to be used in operations like load, store, load immediate, etc. The register bank consists of a demultiplexer and multiplexer. The former enables each of the four registers while the latter multiplexes the outputs from each of the four registers onto the data bus. The 4^{th} and 5^{th} instruction bits (from MSB) obtained from via latches L3 and L1 are the selector bits of the demultiplexer and multiplexer inside the register bank. Note that the latches shown (and the ones that are not shown) in Fig 3A are combinatorial logic implemented using logical gates. Latch L8 ensures that one operand flows directly into the ALU block while the other is in the ACC when executing an operation requiring two operands (like addition). Similarly, the latch L5 connected to the OR controls the flow of data (result obtained from ALU stored in OR) into the DM.

To ensure brevity, the implementation details of every latch as well as the process of setting the enable bits of various elements have not been included in the architecture. Interested readers are suggested to open the CU_v2_3 file in the folder v2.3 in Logisim for further details.

Note that the blue line indicates the path of data bits from/to the data memory (which is essentially a RAM block in Logisim).

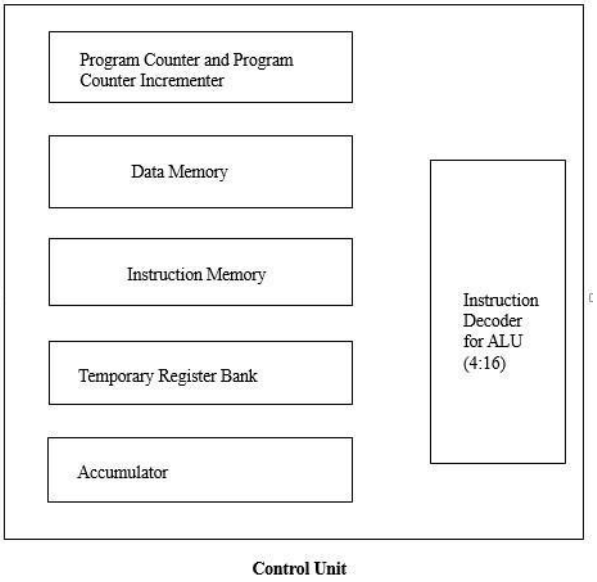

Figure 2: Control Unit Overview

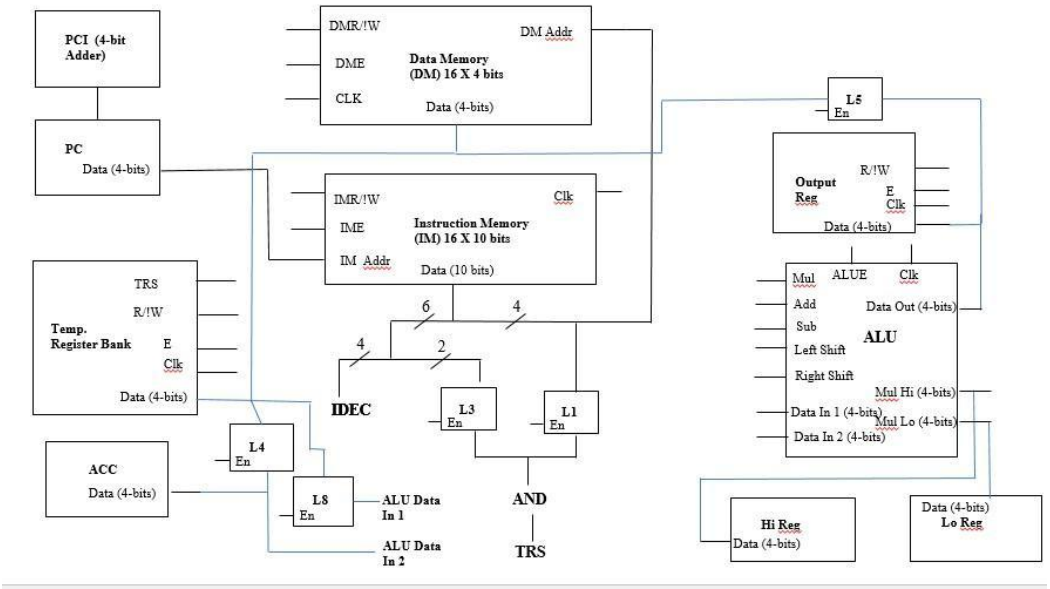

Figure 3 A: Control Unit Architecture

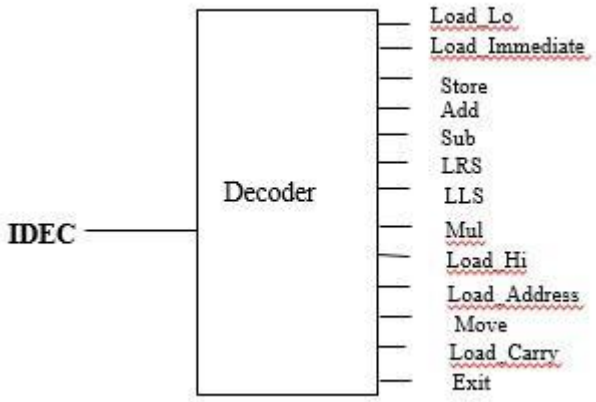

Figure 3B: Instruction Decoder

Command Format		
xxxx	xx	xxxx
Instruction bits	Register identifier	Data/secondary register identifier

Figure 4: Instruction Format

Sl. No.	Instruction bits				Description	Command
0	0	0	0	0	load LO	llo
1	0	0	0	1	load immediate	li
2	0	0	1	0	store	st
3	0	0	1	1	add	add
4	0	1	0	0	subtract	sub
5	0	1	0	1	shift right (logical)	lrs
6	0	1	1	0	shift left (logical)	lls
7	0	1	1	1	multiply	mul
8	1	0	0	0	load HI	lhi
9	1	0	0	1	load from memory	lw
10	1	0	1	0	move	move
11	1	0	1	1	load carry	lc
12	1	1	0	0	reserved for future	-
13	1	1	0	1	reserved for future	-
14	1	1	1	0	reserved for future	-
15	1	1	1	1	exit	exit

Figure 5: Instruction List

Arithmetic and Logic Unit

This block performs addition, subtraction, logical shift left, logical shift right, and multiplication on the operands stored in the DM. The block also comprises of three additional registers to store the carry bit as well as the first and last 4 bits of the result obtained after multiplication. The first 4 bits are stored in Mul Hi and the last 4 bits are stored in Mul Lo (shown in Fig 6). Each of the aforementioned operations are implemented by extending the simpler 1-bit versions to 4-bits. For instance, the 4-bit adder is an extension of 1-bit full adder.

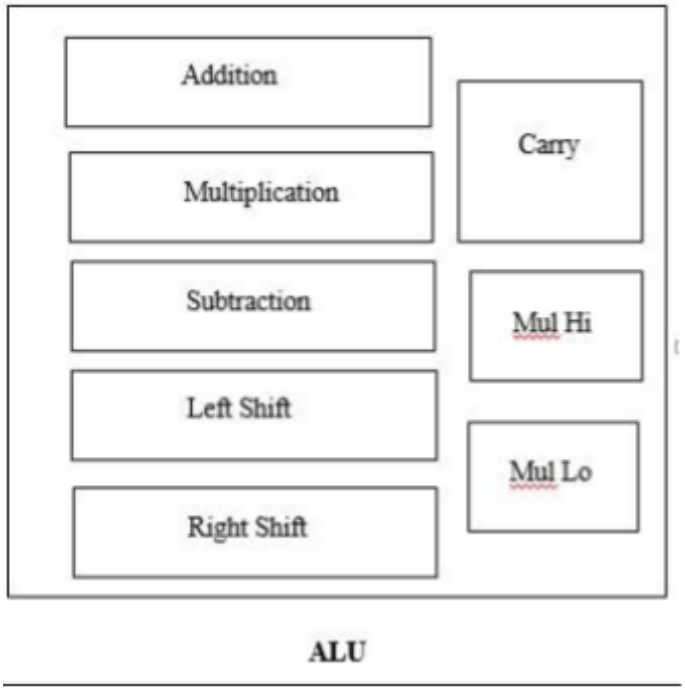

Figure 6: Overview of ALU

Implementation of Other Operations

Other operations are implemented using a shift register controlled by a clock. When a particular operation is to be executed, the IDEC sends the appropriate 4-bits to the decoder which subsequently enables the block responsible to execute that operation. For instance, when the LOAD_ADDRESS flag is enabled, the shift register sets all the remaining flags involved in execution of the "Load Address". The other instructions are also implemented in a similar manner (shown in Fig 7).

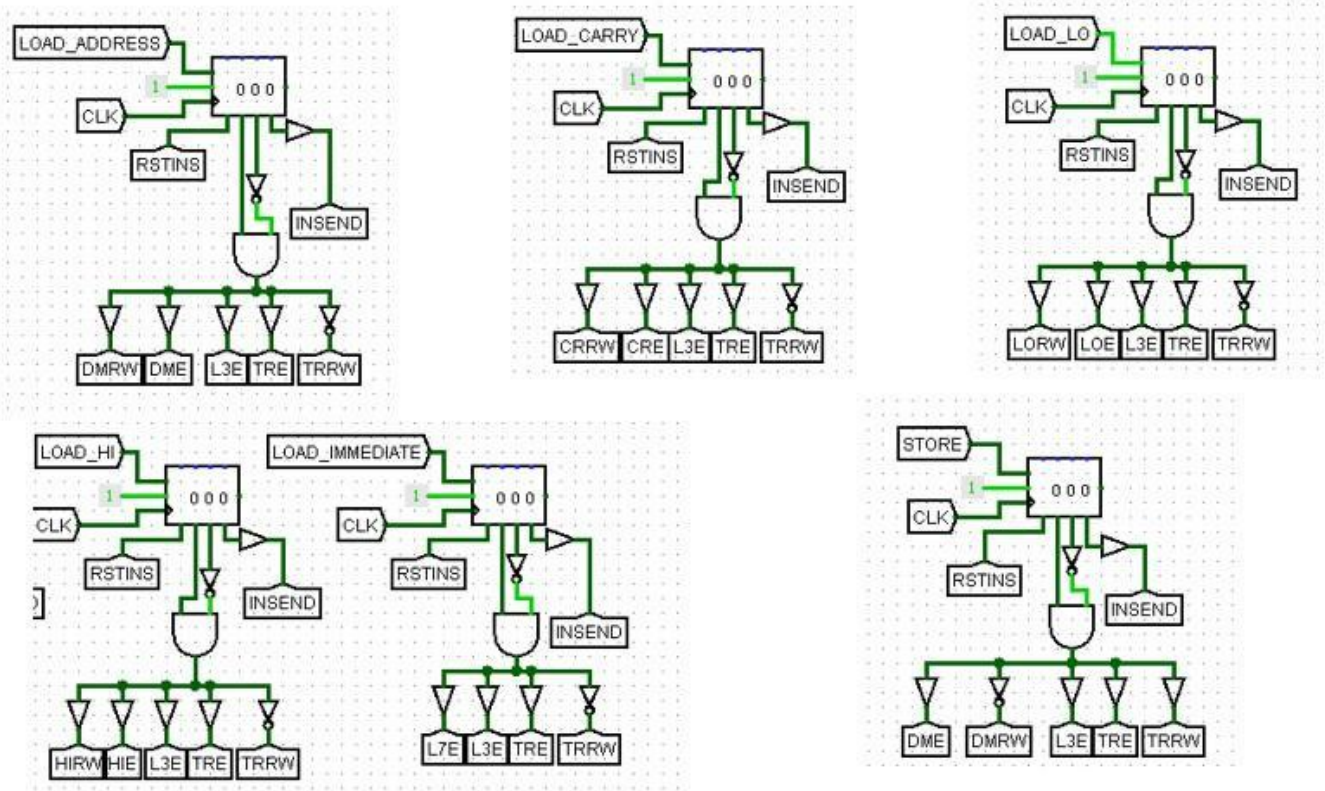

Figure 7: From top left to bottom right: (a) Load Address; (b) Load Carry; (c) Load Lo; (d) Load Hi and Load Immediate; (e) Store