Computer Architecture

Assignment 1

Creating a Simple ALU

Bryan Yang

Table of Contents

[Phase One 3](#_Toc110955243)

[Phase Two 3](#_Toc110955244)

[Phase Three 4](#_Toc110955245)

[Phase Four 4](#_Toc110955246)

[References: 7](#_Toc110955247)

## Phase One

Start by building an 8-bit ALU using Logisim. This ALU can implement 16 instructions on 8-bit operands. We would suggest the following minimum list of instructions:

* Arithmetic addition
* Increment
* Decrement
* Comparison (with 3 outputs: one for equals, one for less than and one for greater than)
* Logic bitwise Not
* Logic bitwise And
* Logic bitwise Or
* Register right logic shift
* Register left logic shift
* Arithmetic division (/): This is one of the foundational arithmetic to compute average and other statistical operations.
* Arithmetic multiplication (\*): This is also one of the foundational arithmetic building blocks to compute statistical operations.
* Arithmetic subtraction (-): This is the base of the counter (countdown) operation. Therefore, it is important.
* Logic bitwise NOR: This instruction is very important for sequential circuits and shift registers Also; this is used in combinational circuits and logics such as multiplexers and half and full adders.
* Logic bitwise XOR: This instruction is heavily used in computer architectures such as electronic circuits of simple digital addition circuits.

In addition to these nine instructions, please suggest **five** more instructions that the ALU can implement for a total of **14 instructions** (we are reserving 2 instructions for branching). **Justify the importance of the five instructions you added in a Word doc to submit as part of this assignment. Label these instructions as 'Phase One.'**

After you've suggested and justified your five suggested instructions, please build at least the nine above-mentioned operations as blocks in Logisim.

Please see the attached Logisim File called ‘8-bit ALU ph1’ (Hanafi, 2020).

## Phase Two

In phase two of the project, you are required to design the instruction set of the ALU/CPU as follows:

* Create the opcode table for the ALU by giving a binary code and a name for each instruction you built in Logisim in phase one.
* Decide how many operands you want your instructions to handle and justify your choice. We suggest either one operand with accumulator or two operands with the result stored in one of the input registers.
* In Logisim, add a multiplexer to the circuit you built in phase one that chooses one of the available operations. The simplest way to create this part of the CPU is to connect the outputs of the multiplexer to the inputs of AND arrays connected to the output of the operation blocks.

Please see the attached Logisim File called ‘8-bit ALU ph2’ (Hanafi, 2020).

## Phase Three

In phase three, you are required to use Logisim to implement the control unit for at least the following three operations:

* addition
* logic bitwise AND
* right logic shift

In order to finish this phase, you need to add operand registers according to the decision you took for the number of operands in phase two and, if needed, a flag register.

Please see the attached Logisim File called ‘Controller ph3’ (Hanafi, 2020).

## Phase Four

In order to be able to write assembly language for the CPU we need to add to instructions (without implementation):

* branch to an address (name it JMP)

“

MOV ax, 00 ; Initializing ax to 0

MOV bx, 00 ; Initializing bx to 0

MOV cx, 01 ; Initializing cx to 1

L21:

ADD ax, 01 ; Increment ax

ADD bx, ax ; Add ax to bx

SHL cx, 1 ; shift left cx, this in turn doubles the cx value

JMP L21 ; repeats the statements

“

* conditional branch to an address (name it CJMP and suppose that the jump takes place if the comparison operation result is 'equals')

“

CMP al, bl

JE EQUAL

CMP al, bh

JE EQUAL

CMP al, cl

JE EQAUL

“

Now, write the following programs using the assembly language you designed in the previous phases of the project as well as these two branching additional instructions:

* Write a program that adds two operands.

“

ADD t0, a, b

“

* Write a program that adds operands until the new value to be added is 0. You do not need to implement the input operations to modify the contents of the registers. Just assume that by the end of each iteration, the register content is modified.

“

IF newValue != 0

ADD a, newValue

“

* Write a program that increments by 2 the content of a register 10 times.

“

INT count = 2

INC [count] \* 10 ; Increment the memory variable COUNT

“

* Write a program that shifts the content of a register until the least significant bit is 0. Think of a way to stop shifting if the content of the register is 11111111 and add it to your program.

“

IF cx != 11111111

SHL cx, 1

ELSE

MOV cx, [my\_program]

“

## References:

Hanafi Ichsan, M. H., & Kurniawan, W. (2020). CPU implementation using only logisim simulator to achieve computer architecture learning outcome. *Bulletin of Electrical Engineering and Informatics*, *9*(2), 747–754. https://doi.org/10.11591/eei.v9i2.1972

*Assembly - Quick Guide*. (2020). Tutorialspoint. https://www.tutorialspoint.com/assembly\_programming/assembly\_quick\_guide.htm#