Lab 8 CPU Report

Ismail Akram

12/20/2021

CSc34200 & 34300

Prof Gertner

Contents

[**Objective** 3](#_Toc90866346)

[Screenshots 4](#_Toc90866347)

[Components 4](#_Toc90866348)

[CPU, 2:1 MUX, Sign Ext 4](#_Toc90866349)

[ALU 6](#_Toc90866350)

[ALU Control Adder 7](#_Toc90866351)

[Control Unit 8](#_Toc90866352)

[Data Memory 9](#_Toc90866353)

[Instruction Register 10](#_Toc90866354)

[Register File 11](#_Toc90866355)

[Simulations 12](#_Toc90866356)

[MIPS 18](#_Toc90866357)

[Conclusion 20](#_Toc90866358)

# **Objective**

The objective of this final lab is to create a simplified CPU that runs the program which computes the sum of five integers using MIPS instructions I have designed in previous labs: LW, ADD, SW.

We will be using the following components: Data Memory, Instruction Memory, 3-ported Register File, IR-Instruction Register, PC- Program Counter, 3 ADD/SUB units, Signed and Zero extension from 16 bits to 32 bits, 2:1 MUX.

# Screenshots

## Components

### CPU, 2:1 MUX, Sign Ext

This is the main vital code that connects all the subsequent components together. Includes multiplexer and sign ext and zero ext. It is the heart of this final lab.Graphical user interface, text

Description automatically generated

Fig 1: Akram\_12\_20\_2021\_Simplified\_CPU.vhdl (Code)Graphical user interface, text

Description automatically generated

Fig 1a: Akram\_12\_20\_2021\_Simplified\_CPU.vhdl (code) cont.

Graphical user interface, text, application, email

Description automatically generated

Fig 1b: Akram\_12\_20\_2021\_Simplified\_CPU.vhdl (Code) cont.

### ALU

Arithmetic Logic Unit, responsible for arithmetic operations; self-explanatory. This is the vital main code in our previous lab (7).

Graphical user interface, text, application, email

Description automatically generated

Fig 2: Akram\_12\_20\_2021\_ALU.vhdl (Code)

### ALU Control Adder

For ALU operation and function. We use this as a component in the CPU.

Graphical user interface, text, application, email

Description automatically generated

Fig 3: Akram\_12\_20\_2021\_ALU\_Control\_Adder.vhdl (Code)

### Control Unit

Generates all control signals: Akram\_JMP, Akram\_branch, Akram\_mem\_rden, Akram\_mem\_wren, Akram\_ALU\_src, Akram\_reg\_wr, Akram\_ext. It takes the OP (OP code) and the func values from the IR and assigns them appropriate values to each of the control signals.

Graphical user interface, text

Description automatically generated

Fig 4: Akram\_12\_20\_2021\_Control\_Unit.vhdl (Code)

Graphical user interface, text, application, email

Description automatically generated

Fig 4a: Akram\_12\_20\_2021\_Control\_Unit.vhdl (Code) cont.

### Data Memory

Reads and writes data in memory. The memory size of the component is 64 bit bytes and each word size is 32 bits. 5 inputs and 1 output. Output is 32-bit memory value that is read.

Graphical user interface, text, application, email

Description automatically generated

Fig 5: Akram\_12\_20\_2021\_Data\_Memory.vhdl (Code)

### Instruction Register

A register that stores the instruction that is currently executed. Being a register means that data gets read and stored only during the RISING EDGE of the CLK (clock). It gets its current instruction from the IR or Instruction Memory, for out simplified CPU, we are only concerned with storage.

Graphical user interface, text, application, email

Description automatically generated

Fig 6: Akram\_12\_20\_2021\_IR.vdhl (Code)

### Register File

32 32bit Register File. It has 3-port RAM to read and write data to registers. This was our previous lab (6).

Graphical user interface, text, application

Description automatically generated

Fig 7: Akram\_12\_20\_2021\_Register\_File.vhdl (Code)

## Simulations

Ultimate test of design: inputting 5 integers and adding them. Note that throughout these screenshots, we see the gradual addition taking place of 1+2+3+4+5 = 15

Graphical user interface

Description automatically generated

Fig 8: ModelSim of Akram\_12\_20\_2021\_Simplified\_CPU

Akram\_CLK: Falling 250 ps

Akram\_reset: 0

@ 125 ps: Akram\_PC = 1, and we have Akram\_read\_data\_1 = 2 and Akram\_read\_data\_2 = 3

@ 375 ps: Akram\_ALU\_result = 3, Akram\_PC = 2

Graphical user interface

Description automatically generated

Fig 8a: ModelSim of Akram\_12\_20\_2021\_Simplified\_CPU

Akram\_reset: 1

@ 700 ps: Akram\_PC = 3, and we have Akram\_read\_data\_1 = 1 and Akram\_read\_data\_2 = 2

@ 1125 ps: Akram\_ALU\_result = 5, Akram\_PC = 5

Graphical user interface

Description automatically generated

Fig 8b: ModelSim of Akram\_12\_20\_2021\_Simplified\_CPU

@ 1300 ps: Akram\_PC = 5, and we have Akram\_read\_data\_1 = 3 and Akram\_read\_data\_2 = 2

@ 1375 ps: Akram\_ALU\_result = 6, Akram\_PC = 6, and Akram\_read\_data\_1 = 3 and Akram\_read\_data\_2 = 3

Graphical user interface

Description automatically generated

Fig 8c: ModelSim of Akram\_12\_20\_2021\_Simplified\_CPU

@ 2000 ps: Akram\_PC = 8, and we have Akram\_read\_data\_1 = 5 and Akram\_read\_data\_2 = 3

@ 2375 ps: Akram\_ALU\_result = 10, Akram\_PC = 10, and Akram\_read\_data\_1 = 6 and Akram\_read\_data\_2 = 4

Graphical user interface

Description automatically generated

Fig 8d: ModelSim of Akram\_12\_20\_2021\_Simplified\_CPU

@ 2875 ps: Akram\_PC = 12, and we have Akram\_read\_data\_1 = 10 and Akram\_read\_data\_2 = 5

@ 2875+ ps: Akram\_ALU\_result = 15, Akram\_PC = 12, and Akram\_read\_data\_1 = 10 and Akram\_read\_data\_2 = 5

Proof that PC values are not forced (note that we still get the same final sum of 15 in the end):  
Graphical user interface

Description automatically generated

Fig 9: (Same as Fig 8)

Graphical user interface

Description automatically generated

Fig 9a: (Same as Fig 8a)

Graphical user interface

Description automatically generated

Fig 9b: (Same as Fig 8b)

Graphical user interface

Description automatically generated

Fig 9c: (Same as Fig 8)c

Graphical user interface

Description automatically generated

Fig 9d: ModelSim of Akram\_12\_20\_2021\_Simplified\_CPU

@ 2875 ps: Akram\_PC = 12, and we have Akram\_read\_data\_1 = 10 and Akram\_read\_data\_2 = 5

@ 2875+ ps: Akram\_ALU\_result = 15, Akram\_PC = 12, and Akram\_read\_data\_1 = 10 and Akram\_read\_data\_2 = 5

# MIPS

Finally, we are testing for correctness in MIPS using MARS.

Graphical user interface, application

Description automatically generated

Fig 10: Akram\_12\_20\_2021\_Simplified\_CPU.asm (Code)

We are simply loading up this code and assembling it.

Graphical user interface, application, table

Description automatically generated

Fig 10a: Akram\_12\_20\_2021\_Simplified\_CPU.asm (Code)

We are running a loop and adding 1+2+3+4+5 to store in register $t3

Table

Description automatically generated

Fig 10c: Akram\_12\_20\_2021\_Simplified\_CPU.asm (Code)

And finally, once the loop exits, we see that register $t3 stores our final sum; F = 15 (highlighted in yellow.

### Conclusion

Our program output the correct sum of 15 by adding 1+2+3+4+5. Our simplified CPU passes the ultimate test. I learned a lot this semester from each lab and as we build each component together to form this final project. I learned how to form logical operators in VHDL and test our design using MIPS on MARS (via VM in Linux). We input data in the Data Memory which was built using LPM RAM from a previous lab. We load the address of the first instruction to the PC-Program Counter register (seen in waveforms). And we executed the code by fetching the first IR-Instruction Register and continued that step-by-step. And finally, we demonstrated correctness of our program via MIP program on MARS simulator.

Thank you.