02132 Assignment 2 report

Hardware implementation in Chisel of a small CPU running the image erosion

Group: 22

Mikkel Arn Andersen **s224187**
Niclas Juul Schæffer **s224744**
Rasmus Kronborg Finnemann Wiuff **s163977**
github.com/rwiuff/02132Assignment2

November 5th

1 Work distribution

Table 1 shows the work distribution in the group for this project.

Table 1: *Work distribution on the project*

Name	Development tasks	Report tasks
Mikkel Arn Andersen Niclas Juul Schæffer Rasmus Wiuff	ISA	Section 2.1

2 Design

2.1. ISA and encoding

The ISA instructions are inspired from Appendix A in the assignment description. It is listed in Table 2.

Table 2: *Instruction-set architecture used in the assignment*

Instruction	**Syntax**	**Meaning**
Arithmetic instructions		
Addition	`ADD Rx, Ry, Rz;`	`Rx = Ry + Rz`
Subtraction	`SUB Rx, Ry, Rz;`	`Rx = Ry - Rz`
Immediate addition	`ADDI Rx, Ry, z;`	`Rx = Ry + z`
Immediate subtraction	`SUBI Rx, Ry, z;`	`Rx = Ry - z`
Logic instructions		
Bitwise OR	`OR Rx, Ry, Rz;`	`Rx = Ry | Rz`
Bitwise AND	`AND Rx, Ry, Rz;`	`Rx = Ry & Rz`
Bitwise NOT	`NOT Rx, Ry;`	`Rx = ~Ry`
Memory instructions		
Load immediate	`LI Rx, y;`	`Rx = y`
Load data	`LD Rx, Ry;`	`Rx = memory(Ry)`
Store data	`SD Rx, Ry;`	`memory(Ry) = Rx`
Control and flow instructions		
Jump	`JMP x`	`GOTO INST x`
Jump if equal	`JEQ Rx, Ry, z;`	`if(Rx == Ry) GOTO INST z`
Jump if less than	`JLT Rx, Ry, z;`	`if(Rx < Ry) GOTO INST z`
Jump if greater than	`JGT Rx, Ry, z;`	`if(Rx > Ry) GOTO INST z`
Do nothing	`NOP;`	`No operation`
END	`END;`	`Terminate`

To design the instructions, first the bit sizes are considered. Some are given in the assignment. If there are 16 registers, these can be reached with $\log_2 16 = 4$ bits. Values for the logic and arithmetic operations are 16 bit as well as addresses in the memory. The opcodes fit within 4 bits.
The instruction layout is laid out in Fig. 1.

Figure 1: *Instruction layout. R1 and 2 are operands, Rd is the destination register. Remaining bits are used for either memory address or immediate value.*

2.1.1. Opcodes As seen in Fig. 1 there are 4 bits allocated to opcodes. Table 2 accounts for seven register type operations, four jump types, three immediate types and two runtime operations.

Table 3: *OPCODE instruction bits.*

Instruction type	OPCODE bits	Instruction
Register	0001	ADD
	0010	SUB
	0011	OR
	0100	AND
	0101	NOT
	0110	LD
	0111	SD
Jump	1000	JMP
	1001	JEQ
	1010	JLT
	1011	JGT
Immediate	1100	ADDI
	1101	SUBI
	1110	LI
Runtime	0000	NOP
	1111	END

2.2. Compile and encode

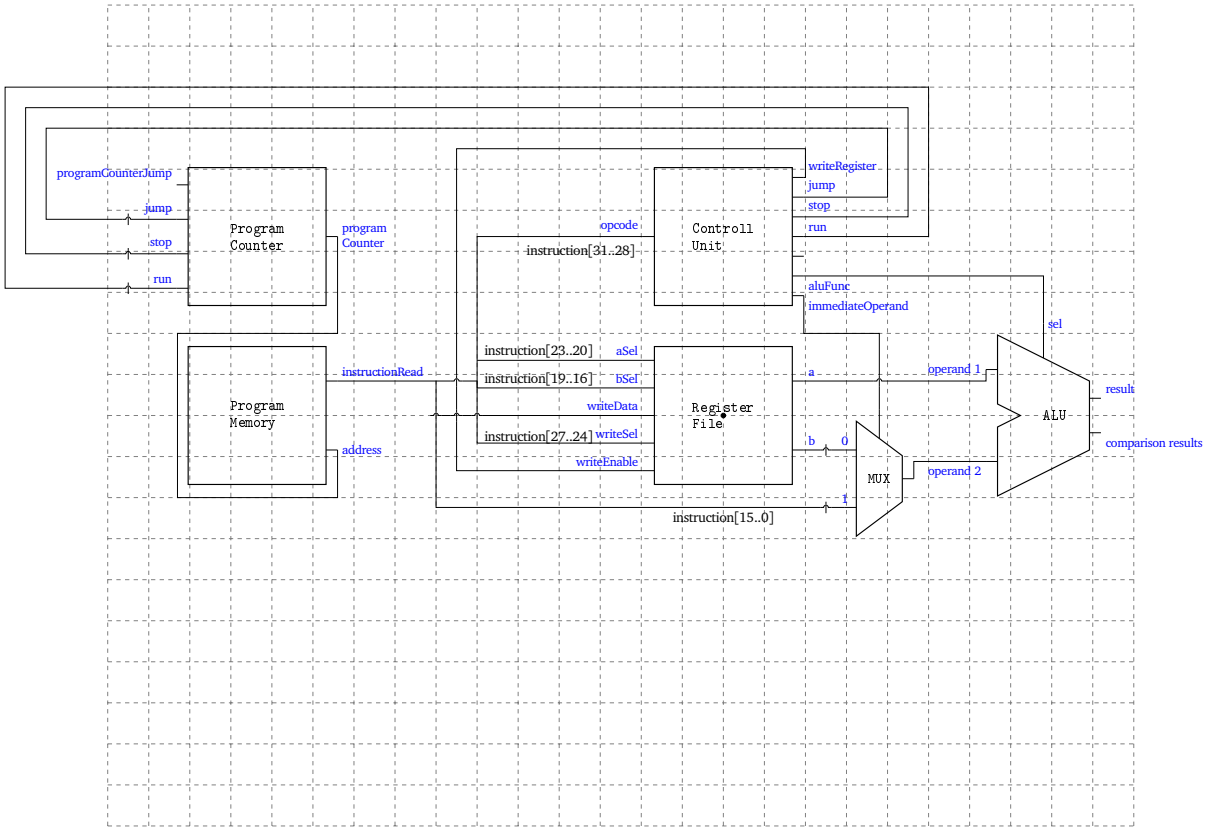
Mikkel Arn Andersen **s224187**
Niclas Juul Schæffer **s224744**
Rasmus Kronborg Finnemann Wiuff **s163977**

Listing 1: *The program compiled to assembly*

```
LOAD R1 <- 0
LOAD R2 <- 0
ADD R8 = R1 + R2
ADD R3 = R8 + 1
ADD R4 = R8 + 20
ADD R5 = R8 + 22
ADD R6 = R8 + 41
LOAD R3 R3
LOAD R4 R4
LOAD R5 R5
LOAD R6 R6
ADD R3 = R3 + R4
ADD R3 = R3 + R5
ADD R3 = R3 + R6
ADD R7 = 0 + 21
STORE R7 <- 255
JEQ R3 === 1020 -> 19
STORE R7 <- 000
ADD R1 = R1 + 1
JEQ R1 === 19 -> 22
JUMP -> 3
ADD R1 = 0 + 0
ADD R2 = R2 + 20
JEQ R2 === 380 -> 26
JUMP -> 3
END;1
```

2.3. CPU BLOCK

Figure 2: *Block diagram of the CPU architecture.*



3 Implementation

Briefly discuss the implementation in Chisel of your design. You can include some code snippets if these are relevant to explain certain aspects of the implementation. In other words, try to answer the question "What does a reader need to know about your Chisel implementation?"

4 Test and analysis

Report here the results from the test you have carried out. Present the test you have developed (if any). Remember to discuss the results and the test you have carried out, do not just present them, but explain and argue their meaning. Address the design evaluation questions listed in Task 11 in the Assignment 2 document.

References

[1] Arduino, José Bagur, Taddy Chung *Arduino Memory Guide (19/09/2023)*
https://docs.arduino.cc/learn/programming/memory-guide