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	Implementation tasks	Report tasks
Mikkel Arn Andersen		
Niclas Juul Schæffer		
Rasmus Wiuff		

2 Design

Explain here what the design process was. List and describe your ISA and how the instructions are encoded. List and describe your compiled program (put a reference to attached file if the compiled program is too long to fit here). Show and describe the block diagram of your CPU. Motivate the design decision you made.

2.1. ISA

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

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

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`
Multiplication	`MUL Rx, Ry, Rz;`	`Rx = Ry · Rz`
	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) GOT INST z`
Jump if less than	`JLT Rx, Ry, z;`	`if(Rx < Ry) GOT INST z`
Jump if greater than	`JGT Rx, Ry, z;`	`if(Rx > Ry) GOT INST z`
Do nothing	`NOP;`	`No operation`
END	`END;`	`Terminate`

2.2. Instruction encoding

To design the instructions, first the bitsizes are considered. Some are given in the assignment. If there are 16 registers, these can be reached with $\log_2 16 = 4$ bits. The sizes are listed in Table 3.

Table 3: *Bitsizes used in the instructions and hardware*

Instruction size	32 bits
Operand size	32 bits
Address size	16 bits
ALU operand	32 bits
Register address	4 bits

The instructions have 3 types: Register type, immediate type and jump type. They are laid out as shown in Fig. 1.

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

Figure 1: *Instruction layouts.*

(a) *Register type instruction. R1 and 2 are operands, Rd is the destination register.*

OPCODE (5 bits) 27···31	R1 (5 bits) 22···26	R2 (5 bits) 17···21	Rd (5 bits) 12···16	

(b) *Immediate type instruction. R1 is an operand. Rd is the destination register. Value is the other operand.*

OPCODE (5 bits) 27···31	R1 (5 bits) 22···26	Rd (5 bits) 17···21	Value (16 bits) 1···16	

(c) *Jump type instruction. R1 and R2 are operands. Target is the target address.*

OPCODE (5 bits) 27···31	R1 (5 bits) 22···26	R2 (5 bits) 17···21	Target (16 bits) 1···16	

2.3. Compiled program

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