Princess Sumaya University for Technology

King Abdullah II Faculty of Engineering

Computer Engineering Department



|  |
| --- |
| **Full report**  **Design and Implementation of a CPU with ALU and Register File** |

|  |  |  |  |
| --- | --- | --- | --- |
| *Authors:* |  |  | *Supervisor:* |
| Mohammad AlJourishi |  | Computer Eng. | Eng. Munther Zabin |

*June 4, 2023*

***Abstract***

*This report presents the design and implementation of a CPU featuring an Arithmetic Logic Unit (ALU) and a Register File. The designed CPU serves as a foundation for studying computer architecture and digital logic design. The main objective of this investigation was to create a functional CPU capable of performing arithmetic and logical operations, storing data in registers, and accessing memory. The report outlines the key subjects covered and highlights the newly observed facts and main conclusions derived from the experiment.*

Table of Contents

[1 Introduction 2](#_Toc136806936)

[1.1 Objectives 2](#_Toc136806937)

[1.2 Theory 2](#_Toc136806938)

[2 Module Declarations 3](#_Toc136806939)

[3 Results and Discussions 4](#_Toc136806940)

[3.1 CPU 5](#_Toc136806941)

[3.2 MEMORY (RAM) 5](#_Toc136806942)

[3.3 Register file 6](#_Toc136806943)

[3.4 arithmetic-logic unit (ALU) 6](#_Toc136806944)

[3.5 Status Register Bits 8](#_Toc136806945)

[4 Conclusions 9](#_Toc136806946)

[5 References 10](#_Toc136806947)

# Introduction

The provided project represents a computer system implemented in hardware using the Verilog hardware description language (HDL). The system includes modules for a CPU (Central Processing Unit), RAM (Random Access Memory), and various ALU (Arithmetic Logic Unit) operations.

## Objectives

The main objective of conducting the experiment with this hardware system is to verify its functional correctness, evaluate its performance, identify any potential issues or improvements, and assess its practical usefulness for executing basic arithmetic and logical operations. The theoretical prediction is that the system will perform accurately, efficiently, and reliably, meeting the desired objectives and demonstrating its practical utility.

## Theory

In the laboratory experiment, the implementation of the hardware system is based on relevant theories and concepts from digital logic and computer architecture. The system is designed to execute basic arithmetic and logical operations using a CPU, interact with memory through RAM, and perform operations using an ALU. Here, we will provide a brief overview of the underlying theory involved in the design and functioning of the system.

1. Digital Logic Theory:
   * Boolean Algebra: The hardware system utilizes Boolean algebra to perform logical operations such as AND, OR, and NOT. Boolean algebra provides a mathematical foundation for designing logic circuits and expressing logical relationships.
   * Logic Gates: The fundamental building blocks of digital logic circuits are logic gates, which implement specific Boolean functions. These gates, such as AND, OR, and NOT gates, manipulate binary signals (0s and 1s) to perform logical operations.
   * Combinational Logic: Combinational logic circuits combine multiple logic gates to produce desired output based solely on the current input values. The ALU module in the hardware system utilizes combinational logic to execute arithmetic and logical operations.
2. Computer Architecture Theory:
   * CPU and Registers: The CPU module serves as the central controller of the system. It includes registers for storing data and an opcode for specifying the operation to be performed. The theory of computer architecture defines the structure and organization of the CPU and its components.
   * Memory Hierarchy: The RAM module represents the main memory of the computer system. It stores data for read and write operations. The memory hierarchy theory explains the organization and access patterns of different levels of memory, including main memory and cache.
   * Instruction Execution: The hardware system follows a fetch-decode-execute cycle, where instructions are fetched from memory, decoded by the CPU, and executed by the ALU. The theory of instruction execution explains the various stages involved in executing instructions and the control flow within the CPU.

Equation Guidelines:

* Equations can be used to express mathematical relationships involved in the theories mentioned above. For example, Boolean algebra equations can be used to represent logical operations.
* When writing equations, use appropriate notation and symbols. Clearly define each symbol used in the equation and provide a brief explanation of its meaning.
* Number the equations for easy reference within the text.
* Ensure that equations are properly formatted and aligned.

# Module Declarations

1. CPU Module:

* Description: Main module that integrates the components of the processor.
* Inputs:
  + clk: Clock input.
  + reset: Reset input.
  + instruction: Instruction input (32-bit).
* Outputs:
  + result: Result output (32-bit).

1. RAM Module:

* Description: Module responsible for implementing the RAM functionality.
* Inputs:
  + address: Address input (32-bit).
  + data\_in: Data input (32-bit).
  + write\_enable: Write enable input.
* Outputs:
  + data\_out: Data output (32-bit).

1. ALU Module:

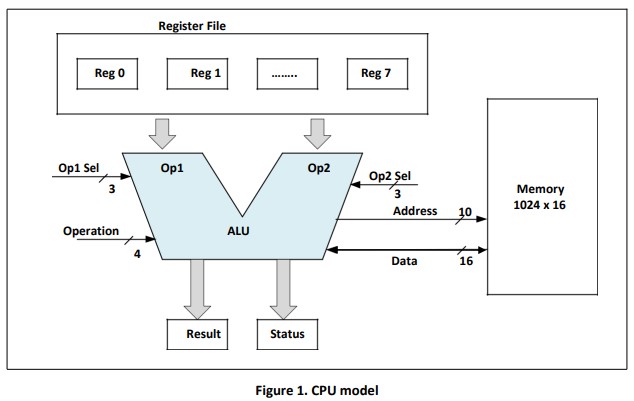
* Description: Arithmetic Logic Unit module for performing operations.
* Inputs:
  + operand1: Operand 1 input (32-bit).
  + operand2: Operand 2 input (32-bit).
  + opcode: Operation code input (3-bit).
* Outputs:
  + result: Result output (32-bit).

1. Control Unit Module:

* Description: Module responsible for generating control signals.
* Inputs:
  + opcode: Opcode input (6-bit).
* Outputs:
  + control: Control signals output (4-bit).

1. Register File Module:

* Description: Module responsible for implementing the register file functionality.
* Inputs:
  + reg\_addr1: Register address 1 input (5-bit).
  + reg\_addr2: Register address 2 input (5-bit).
  + write\_data: Data to be written input (32-bit).
  + write\_enable: Write enable input.
* Outputs:
  + data1: Data output from register 1 (32-bit).
  + data2: Data output from register 2 (32-bit).



# Results and Discussions

The Central Processing Unit (CPU) implemented using the Verilog hardware description language. The CPU consists of several modules, including **CPU**, **RAM**, **REG\_file**, **ALU**, and various arithmetic and logical modules. Let's discuss each module and its functionality in more detail:

## CPU

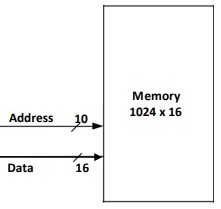
* Inputs: Reg0 to Reg7 (16-bit registers), op (4-bit operation code), clk (clock signal), op1\_sel and op2\_sel (3-bit register select), op1 and op2 (16-bit operands), and write\_data (data to be written to memory).
* Outputs: result (16-bit ALU result), status (3-bit status flags), address (10-bit memory address), and read\_data (data read from memory).

Description: The cpu module acts as the top-level module for the CPU implementation. It instantiates other modules such as reg\_file, alu, and RAM to perform the desired operations based on the provided inputs. The output result is determined based on the selected operation code (op), and the status flags (status) indicate conditions such as carry, zero, and overflow. The memory operations are handled by the RAM module.

## MEMORY (RAM)

* Inputs: clk (clock signal), address (10-bit memory address), and write\_data (data to be written to memory).
* Outputs: read\_data (data read from memory).

Description: The RAM module represents the memory unit of the CPU. It consists of a 1024x16 memory array (memory) and supports read and write operations. On the negative edge of the clock signal (clk), the module performs memory operations. If the provided address is within the valid range (0-1023), the module reads data from or writes data to the memory array accordingly.



## Register file

* Inputs: Reg0 to Reg7 (16-bit registers), op1\_sel and op2\_sel (3-bit register select).
* Outputs: op1 and op2 (16-bit operands).

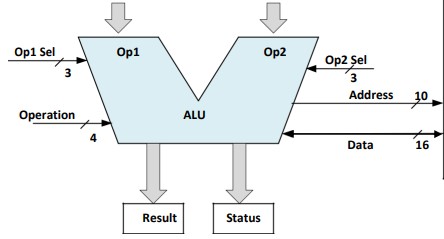
Description: The reg\_file module represents the register file of the CPU. It provides the selected registers (op1 and op2) based on the given register select inputs (op1\_sel and op2\_sel). The module uses a case statement to assign the appropriate register value to the output operands.

|  |  |
| --- | --- |
| Operand Selection | Selected Register |
| 000 | Reg 0 |
| 001 | Reg 1 |
| 010 | Reg 2 |
| 011 | Reg 3 |
| 100 | Reg 4 |
| 101 | Reg 5 |
| 110 | Reg 6 |
| 111 | Reg 7 |

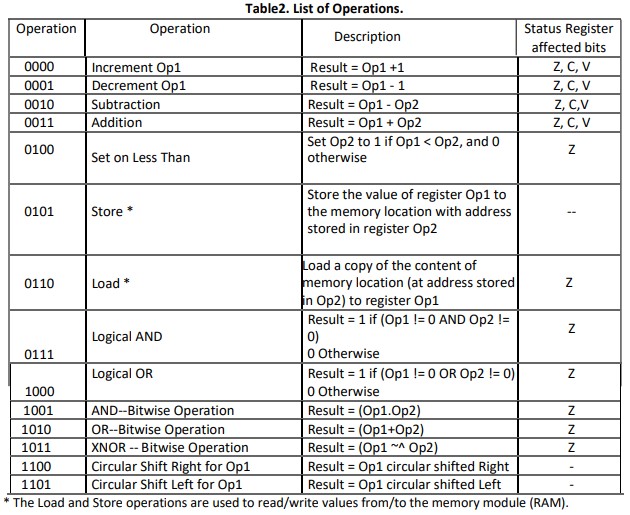
Table 1. Register Select

## arithmetic-logic unit (ALU)

* Inputs: op1 and op2 (16-bit operands), op (4-bit operation code), clk (clock signal), and write\_data (data to be written to memory).
* Outputs: result (16-bit ALU result), status (3-bit status flags), address (10-bit memory address), and read\_data (data read from memory).



Description: The alu module represents the Arithmetic Logic Unit (ALU) of the CPU. It performs various arithmetic and logical operations on the provided operands. The specific operation is determined by the op input. The module instantiates different arithmetic and logical modules to perform operations such as addition, subtraction, logical AND, logical OR, bitwise AND, bitwise OR, etc. The resulting output is assigned to result, and the status flags (status) are updated accordingly. The module also sets the memory address (address) for memory operations and updates the read\_data output with the data read from memory.



## Status Register Bits

The status register you mentioned has three bits or flags: C, Z, and V. These flags are used to indicate certain conditions or results of operations in a computer system. Here is a breakdown of each flag:

1. C (Carry Flag):

* Bit 2 of the status register.
* Set to 1 when an addition operation produces a carry.
* Set to 0 by default.
* The carry flag is used to detect if an addition operation exceeds the maximum representable value.

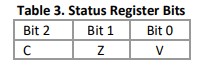
1. Z (Zero Flag):

* Bit 1 of the status register.
* Set to 1 when the result register equals zero.
* Set to 0 by default.
* The zero flag is used to indicate if the result of an operation is zero.

1. V (Overflow Flag):

* Bit 0 of the status register.
* Set to 1 when a subtraction operation produces an overflow.
* An overflow occurs when the result of a subtraction operation exceeds the maximum or minimum representable value.
* Set to 0 by default.

These flags are important for program execution and decision-making in the system. They can be checked or modified by the program to handle specific conditions or to control the flow of execution. For example, conditional branch instructions can be based on the values of these flags to perform different actions depending on the outcome of a previous operation.



# Conclusions

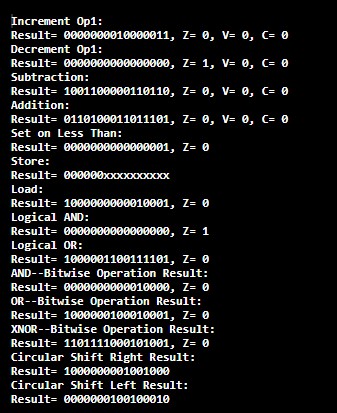
The CPU module with various sub-modules, including a register file, ALU (Arithmetic Logic Unit), and RAM (Random Access Memory).

The CPU module takes several inputs, including eight 16-bit registers (Reg0 to Reg7), an operation code (op), clock signal (clk), and control signals (op1\_sel and op2\_sel) for selecting registers. It also has output ports for the result of the ALU operation (result), status flags (status), memory address (address), and data read from memory (read\_data).

The RAM module implements a memory array with 1024 memory locations, each storing a 16-bit value. It has inputs for the memory address (address), data to be written (write\_data), and clock signal (clk). On each negative edge of the clock, the module performs read and write operations based on the provided address and data signals.

The reg\_file module acts as a register file, selecting registers based on the op1\_sel and op2\_sel control signals and outputting the selected register values (op1 and op2) to the ALU module.

The ALU module performs various arithmetic and logical operations based on the op code and input operands (op1 and op2). It uses sub-modules for specific operations like increment, decrement, addition, subtraction, logical AND, logical OR, bitwise AND, bitwise OR, bitwise XNOR, circular shift right, and shift left. The ALU module generates the result based on the selected operation and sets the status flags accordingly.

In conclusion, implements a simple CPU with register file, ALU, and RAM modules. It performs arithmetic and logical operations on the input operands and stores results in registers or memory. Demonstrates the basic functionality of a CPU and can be further expanded to support more complex operations and instructions.

Output example using the testbench in the code:

# References

|  |  |
| --- | --- |
| [1] | Hennessy, J. L., & Patterson, D. A. (2017). Computer Architecture: A Quantitative Approach. Morgan Kaufmann. |
| [2] | Patterson, D. A., & Hennessy, J. L. (2013). Computer Organization and Design: The Hardware/Software Interface. Morgan Kaufmann. |
| [3] | Intel. (2021). Intel Core Processor Family: Desktop, Mobile, and HEDT (High-End Desktop) Processors. |