UrCPU

Susannah Cheezum
Mathematics
Ursinus College
sucheezum@ursinus.edu

Michael Cummins
Computer Science and Philosophy
Ursinus College
micummins@ursinus.edu

Isabelle Son
Computer Science
Ursinus College
isson@ursinus.edu

Eugene Thompson
Mathematics
Ursinus College
euthompson@ursinus.edu

***Abstract*—This CPU specification is an evaluation of the course project for Ursinus College's Computer Science Architecture and Organization course. The course was taken by the above students in the Spring of 2024, taught by Dr. Santiago Núñez-Corrales.**

Keywords—module, unit, register, design, collaboration

I. Introduction

We were tasked with creating a functioning central processing unit (CPU) completely from scratch, using the hardware description language (HDL): Verilog. This task includes creation and implementation of an arithmetic logic unit (ALU) with corresponding general purpose, program segment, pointer, status, and increment registers. The components interact via instructions from the control unit and are stored in a bank of internal memory.

To approach this task, we sought to capitalize on the individual strengths of each contributor while closely collaborating. One who felt comfortable tackling the ALU mainly focused on the intricate mathematical operations of the unit, while another chose to dive into the control unit and its corresponding registers through brute force. Both parties received support from the other collaborators providing other necessary registers and testbenches for each component. We met regularly to discuss the status of the project, including troubleshooting issues and celebrating successes.

II. Hardware Design Decisions

A. Overall Design

Using the modular approach, we implemented separate files containing modules pertaining to each component. Separation of modules alongside their respective testbenches allows for easier initial testing and debugging. The modular design will also allow for future ease of readability and scaling.

B. Registers

Each register is crucial in terms of its storage capacities and instructions within. The general-purpose registers are modules dedicated to memory addresses and numerical data. Each program segment and pointer register module corresponds with addresses and values that vary throughout runtime. The read-only status register is one module, complete with many flags needed for the testing of the CPU. The read-only increment register modules are needed to keep track of microprocessor execution.

C. ALU Design

The ALU is designed to support a variety of arithmetic and logical operations, including addition, subtraction, AND, OR, XOR, shift left, and shift right. It also includes circuitry for handling carry and overflow conditions, ensuring accurate operation for a wide range of calculations.

D. Memory Bank

The CPU includes a bank of internal memory for storing instructions and data. The memory bank is organized into addressable units, with each unit storing a single instruction or data value. The memory bank is accessed using the pointer register to retrieve or store information.

E. Control Unit

The control unit of the CPU is responsible for managing the execution of instructions. It decodes instruction opcodes and generates control signals to coordinate the operation of the ALU, registers, and memory bank.

Each component is carefully designed to ensure compatibility and efficient operation within the CPU architecture. Overall, the hardware design of the CPU is optimized for performance, versatility, and ease of integration into various computing systems.

III. Instruction Set Architecture Design

The instructions for the UrCPU consists of a 20-bit word written from the least significant bit to the most significant bit. Therefore, to read the instruction, it was necessary to parse through the word in reverse order, from index 19 to index 0. The instructions consisted of six bits allocated for the OP code, six total bits for two source registers for computation (three bits each to represent their address in the six general registers), six bits for two result registers to store the output of a computation (three bits each to represent their address in the six general registers), one bit to specify whether or not the ALU is in full or half-word mode(with a 0 corresponding to half word and 1 corresponding to full word), and one extra bit to allow for further expansion upon the instructions. The full instruction word format from index 19 to 0 is as follows:

|OPCODE|SRC_REG1|SRC_REG2|DEST_REG1|DEST_REG2|ALU_MODE|EXTRA|

The available operations through the instructions were trap mode, no operation, jump unconditional, jump if zero, jump if sign. jump if zero and sign, load status register, XOR status register, logical negation, logical AND, logical

OR, logical XOR, shift right, shift left, rotate right, rotate left, exchange values, increment, decrement, add, add with carry, subtract, subtract with carry, equal than, greater than, less than, greater or equal than, and less or equal than. The actual encoding was accomplished by simply labeling each operation a number with six bits in numerical order as listed above, starting with the trap mode represented by “000000”, the no operation represented by “000001”, etc.

To demonstrate an example of a fully formed instruction, the following binary word (written from index 19 to 0) would be decoded into an instruction resulting in an AND with inputs from registers 4 and 5 and store result in register 0 in full word mode.

instruction = 20'b00100110010100000010;

The binary word would the separated as 001001(AND)|100(register 4)|101(register 5)|000(register 0)|000(empty bits - destination register 2 is unused in this instruction)|1(full word mode)|0(extra bit).

IV. Implementation

The design of our CPU is formulaic and the generally expected format for such a unit. In terms of each piece of the CPU, sometimes it was difficult to decide when to separate modules into different files or to keep them grouped together. We found this project quite difficult to debug, as Verilog HDL errors are more difficult to catch in comparison to the object-oriented programming languages we are accustomed to.

As it stands, our code is likely not optimized for ultimate speed and versatility, it is a rather rudimentary processing unit. Additionally, our bank of internal memory was not able to be fully implemented. With more guidance, our combined efforts could have created something much more effective and applicable to several computing devices. Though potential errors may occur, professional debugging would be needed to fully solidify the unit.

As a group, we ran into little issues with collaborating, but were sometimes limited by individual skill level and overall comprehension. We all assisted each other as requested, similarly leaving each to their own devices when it was more efficient to do so. All in all, the final product is the result of the combined best efforts of all four contributors.

V. Discussion and Conclusions

Thompson went above and beyond in his efforts for the ALU. Cummins ambitiously tackled the control unit and general-purpose registers. Son provided working code for the remaining non-status registers. Cheezum took a supporting role in the efforts of her other contributors after implementing the flags needed for the status register.

Our group believes this CPU could get fully up and running with further intel from professors and peers. Although the project process is over, the group of us believe the CPU could become more optimal and applicable to several operating systems with considerable time to do so.

For scaling, we may revisit this project, perhaps transforming it from a 20-bit central processing unit to a larger CPU. In that instance, more registers could be added to increase the robustness of the unit. For this, perhaps more complex notions of the concepts within the project as well as updated software to run it would be necessary.

Acknowledgment

We would like to thank our professor, Dr. Núñez-Corrales for graciously guiding us through this process. His assistance in office hours as well as shared resources helped us to make headway on this cumbersome project.