STM JOURNALS

Journal of
Electronic Design Technology

ISSN: 2229-6980 (Online)
ISSN: 2321-4228 (Print)
Volume 15, Issue 3, 2024
September–December
DOI (Journal): 10.37591/JoEDT

https://journals.stmjournals.com/joedt

Research JoEDT

Building Knowledge with a Hands-on 8-bit CPU Architecture Simulation Kit for Students

Ajinkya Kshirsagar[1], Ashit Gajjar[1,*], Hetal Shah[2]

Abstract

It is now crucial for students studying electronics and information technology to comprehend computer architecture and logic design. Hands-on, practical instruction is required to assist students in understanding Computer Organization and Architecture (COA). Our kit, which covers essential components including bus interfaces, arithmetic, and logic units (ALU), memory structures, and functional registers, is based on the Von Neumann architecture. Theoretical techniques like block diagrams and top level diagrams are typically used to teach COA, and they frequently ignore the digital design component. Students get a deeper knowledge of how theoretical concepts appear in practical systems by creating a completely functional CPU from scratch. The purpose of this research is to use digital logic gates to construct an 8-bit CPU based on the SAP-1 Architecture. Students will gain a deeper understanding of Computer Architecture and Organization (CAO) with the use of this simulation kit. Students can study digital construction, gain an understanding of CPU operation, and develop the skills necessary to construct customized computers using this kit by working with logic gates. Using digital logic gates in a real hardware simulation, this research presents a model that illustrates the intricate workings of an 8-bit CPU. Students who use this method could construct personalized computers and gain more intuitive knowledge.

Keywords: 8-bit CPU, student learning kit, computer architecture, digital logic design, COA, SAP-1 Architecture

INTRODUCTION

Computer systems play a vital role in all engineering professions. Users consistently seek a balance among speed, power, and affordability. In response to user demands, system designers must understand processor components, including their design, functionality, and effects on the overall performance of computer systems. Currently, progress in the Faculty of Computer Science has made Computer Architecture and Organization (CAO) a fundamental subject for undergraduate students across all departments within the faculty. Among the various departments that embrace advancements in CAO are computer science (CS) and computer engineering (CE).

***Author for Correspondence**
Ashit Gajjar
E-mail: ashitgajjar762@gmail.com

[1]Student, Department of Electronics and Communication Engineering. Dharmsinh Desai University, Nadiad, Gujarat, India
[2]Professor, Department of Electronics and Communication Engineering. Dharmsinh Desai University, Nadiad, Gujarat., India

Received Date: September 03, 2024
Accepted Date: September 18, 2024
Published Date: September 27, 2024

Citation: Ajinkya Kshirsagar, Ashit Gajjar, Hetal Shah. Building Knowledge with a Hands-on 8-bit CPU Architecture Simulation Kit for Students. Journal of Electronic Design Technology. 2024; 15(3): 1–9p.

Universities employ computer-based graphical simulators to enhance the teaching of computer architecture. These simulators vary from straightforward visual tools to sophisticated intricate platforms utilized in both research and product development. Such simulators are frequently employed to assist students in grasping intricate technologies that can be challenging to comprehend and visualize without the aid of graphical animation. Visualization of different architectures enhances the learning process among students using simulators, and processor performance has improved at a faster

rate than Moore's Law anticipated because designers have been able to take advantage of increasing on-chip transistors to extract greater parallelism from software [1].

Several universities utilize cost-effective microcontrollers as educational tools for instructing CAO; however, they still fall short in terms of fostering the skills required for designing computer systems because they are working with preexisting systems. Few universities use VHDL with the help of FPGA to design a multicore CPU, but this approach assumes that students are already familiar with basic knowledge and can be carried out within a semester. This approach is valid at the master's level, but it is difficult for bachelors/undergraduate level people to be more interested in knowing where things are headed than in reflecting on the past [2].

Computer Architecture (CAO) focuses on the attributes of a computer system relevant to programmers, encompassing elements such as instruction sets, arithmetic operations, addressing methods, and I/O mechanisms. One of the simplest CPU designs is the 8-bit CPU, which uses a simple -1 (SAP-1) architecture. This design includes registers, a memory address latch, an arithmetic, and logic unit (ALU), a Control Unit, a Data Bus, memory, and status words. Several CPUs have been designed using various approaches; however, from a learning perspective, the simplest methods are required. SAP-1 utilizes the Von Neumann architecture, which is characterized by the integral concept of storing data and instructions within the same memory system in a consistent manner. Derived from the Von Neumann architecture, there have been subsequent developments, including the SAP-1, SAP-2, and SAP-3 architectures. However, to meet students' educational requirements, the SAP-1 Architecture is favored owing to its simplicity and suitability [3].

The objective of this study is to create an 8-bit CPU following the SAP-1 Architecture by employing digital logic gates for its implementation. This simulation kit aims to fulfill students' needs by facilitating their comprehension of CAO. By utilizing logic gates, this kit offers students an opportunity to delve into digital design, gain insight into CPU operations, and ultimately empower them to design purpose-specific computers. This section describes the development of an 8-bit microprocessor trainer kit [4, 5]. It consists of building the registers from scratch on the breadboards followed by rigorous testing of each block, which consists of a random access memory (RAM) module, memory address register, control unit, and the program counter.

Further, the design was implemented on a Printed Circuit Board (PCB) and finally developed into an educational trainer kit for the learning purpose of the students to obtain in-depth knowledge of the registers. The sections involved are as follows.

1. Contribution and motivation
2. Architecture and working
3. Discussion
4. Conclusion
5. References

The contribution and motivation to delve into this complete hardware architecture can be elaborated on with the help of the following points:

1. The 8-bit CPU design serves as a practical learning tool for understanding the inner workings of computer processors. Through the construction process, enthusiasts gain insights into fundamental concepts, such as binary arithmetic, logic gates, instruction decoding, and memory addressing.
2. By physically assembling and programming a CPU, learners develop a deeper appreciation of the complexities of computing systems, paving the way for further exploration and experimentation.
3. Regardless of the academic background, this kit can be useful to the students in clearing their concepts of the microprocessor from scratch.
4. The main intention behind the development of this microprocessor trainer kit is to empower individuals with the knowledge and skills necessary to understand and create computing technologies.

5. To close the gap between the theoretical understanding and real-world applications, an 8-bit CPU was developed. Conventional computer science courses tend to concentrate on abstract ideas rather than giving students the chance to put what they have learned into use.
6. This kit fills this need by providing a practical application that strengthens theoretical principles through hands-on learning.
7. By building a fully functional CPU from scratch, students gained a deeper understanding of how theoretical principles manifest in real-world systems.
8. By implementing this kit, one can delve into new areas of interest and push boundaries of understanding.

ARCHITECTURE AND WORKING

The SAP-1 (Simple as Possible) architecture is a basic design for a simple computer created by Albert Paul Malvino in the late 1960s. It was primarily developed as an educational tool to introduce students to the fundamental concepts of COA. The SAP-1 computer employs the Von Neumann architecture and is organized using a bus system. It utilizes an 8-bit central bus and comprises ten primary components. A visual representation of its architecture is presented in Figure 1.

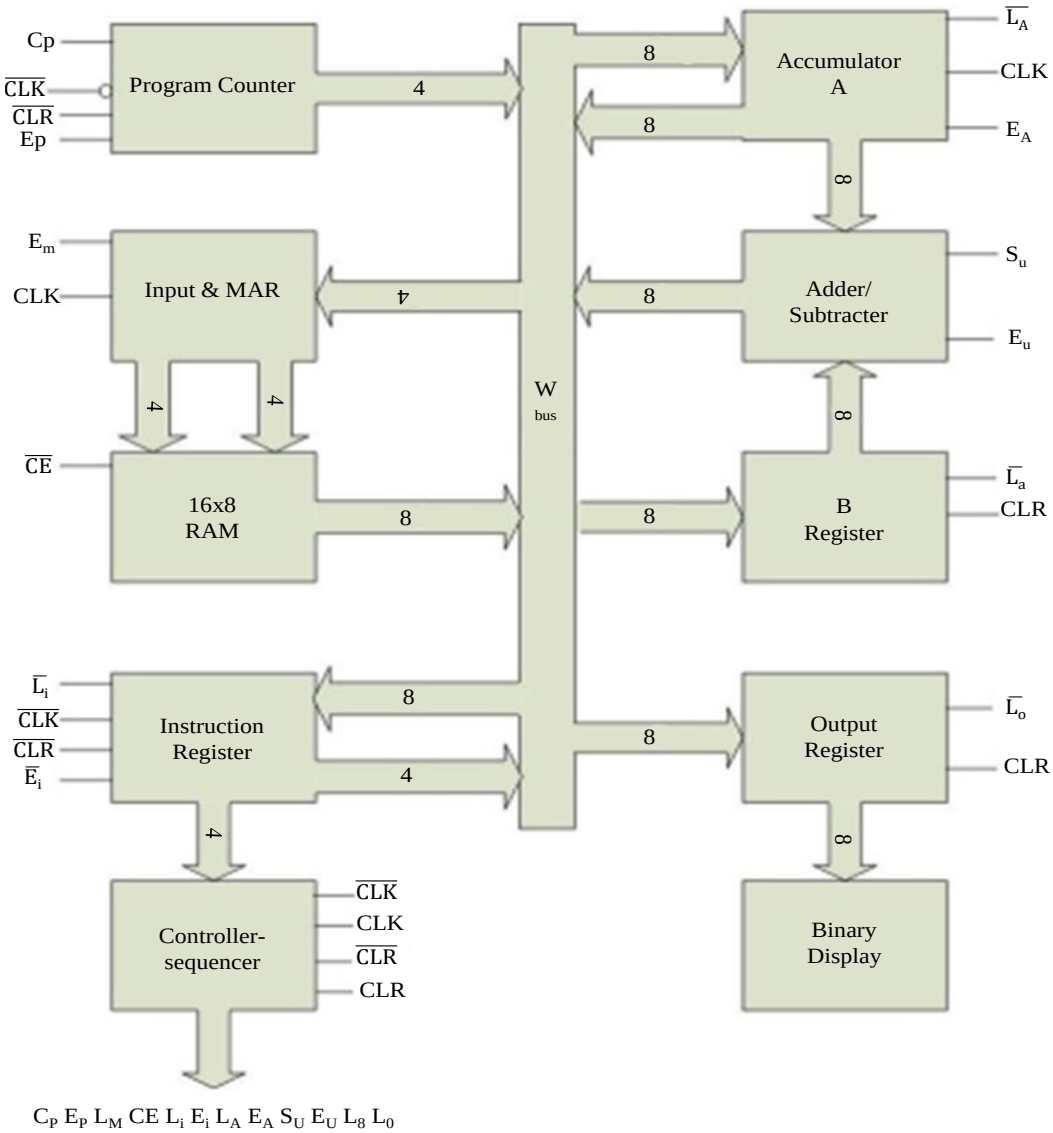

Figure 1. Block diagram: SAP-1 Architecture.

As mentioned in the block diagram 1, each block has its unique functionality, and its importance is as provided by the SAP-1 Architecture:

1. General purpose registers
2. Random access memory
3. Instruction set
4. Instruction format
5. Instruction execution
6. Program counter
7. Arithmetic logic unit
8. Instruction register
9. Control unit

Below is the detailed work on the above-mentioned blocks, and their description is as follows.

General Purpose Registers

General purpose registers, which operate similarly to RAM and utilize conventional flip-flops, serve as a type of memory. Their proximity to arithmetic and logic units offers the significant advantage of faster calculations compared to retrieving data from data memory. As a result, registers are commonly utilized as parameters in almost all instructions. The following steps must be executed when performing a write operation [6].

The reading process was straightforward. The only requirement is to designate the desired address for the source and/or destination registers, after which content is efficiently transferred to the corresponding outputs.

Random Access Memory

The data produced by the main program are stored in the data memory, which is a subsection of RAM. The stored data can consist of variable values or constants utilized by the program to perform calculations. The data memory address bus has a width of 8 bits, enabling access to all locations ranging from 0x00 to 0xFF. Because this memory is volatile, data are lost when the processor shuts down. Write and read are the two modes of operation for data memory. A write operation requires the following steps. At the address bus, enter the required address.

1. Fill in the input data bus with the desired data.
2. Establish a clock pulse as the signal for control.
3. The data are accessible on the output bus and saved where desired.

The reading process is much easier. All that is required is setting the necessary address at the address bus, after which the content of the location is instantly transferred to the matching data output bus.

Instruction Set

Every processor requires a unique instruction set, encompassing the assembly code and binary format in the machine language. The instruction set must address two critical considerations: its simplicity and robustness. It is essential for the instruction set to be straightforward to comprehend while also ensuring reliability in its functionality. To achieve this, the simplest instructions are chosen, enabling the processor to perform any operation or routine in the fewest possible steps. The three groups of instructions are categorized based on their intended use:

1. *Operations* directives influence the values in the registers.
2. *Instructions* influencing the sequence of execution are known as program control.
3. *Data transferring* commands that modify the information stored in memory.

The instruction set for the 8-bit CPU was covered above.

Instruction Format

The instruction format pertains to the specific arrangement of bits within an instruction based on the parameters utilized. The architecture of the CPU, the number of instructions required, and the operands

it manages to influence the design of the instruction format. Each instruction is required to have a distinct identifier, known as the operation code, commonly referred to as OPCODE. The OPCODE bit length was determined by the total number of instructions.

Instruction Execution

The processor is responsible for executing a source code program by following a predetermined sequence corresponding to the task at hand. Each instruction within the program produced the necessary results required by the subsequent instructions. The instructions in the program memory be executed sequentially, and each instruction must be completed before loading the next instruction. It is necessary to divide the processing time into basic operations that apply to all instructions to carry out an instruction.

The initial task involves retrieving the instruction from the program memory and loading it into instruction registers, commonly referred to as FETCH. The subsequent task involves decomposing the instructions into sub-instructions that are relevant to each of the subsequent functional units. This stage is known as DECODE.

After the instruction is decoded, the processor's functional units are informed of how to handle the resulting data. This stage is known as EXECUTE, during which the processor obtains a result based on instructions. The last job for an instruction is to save the WRITEBACK data created in the previous step.

It is important to note that not all instructions require saving a result, as some instructions do not produce any output. In such cases, the instructions are concluded in the preceding stage. It is worth mentioning that modern processors employ more advanced stages, further decomposing them into additional substages.

Binary data processing is based on a physical set of hardware components called functional units that comprise a processor. To perform their designed tasks efficiently each functional unit must meet a set of logical design requirement [4-7] below are the functional units of 8- bit micro processor are discussed.

Program Counter

The binary counter, known as the program counter (PC), oversees the creation of the address required to obtain an instruction from the program memory. It maintains the record of the address for the next instruction, which will be executed once the current instruction is completed. The PC must be able to

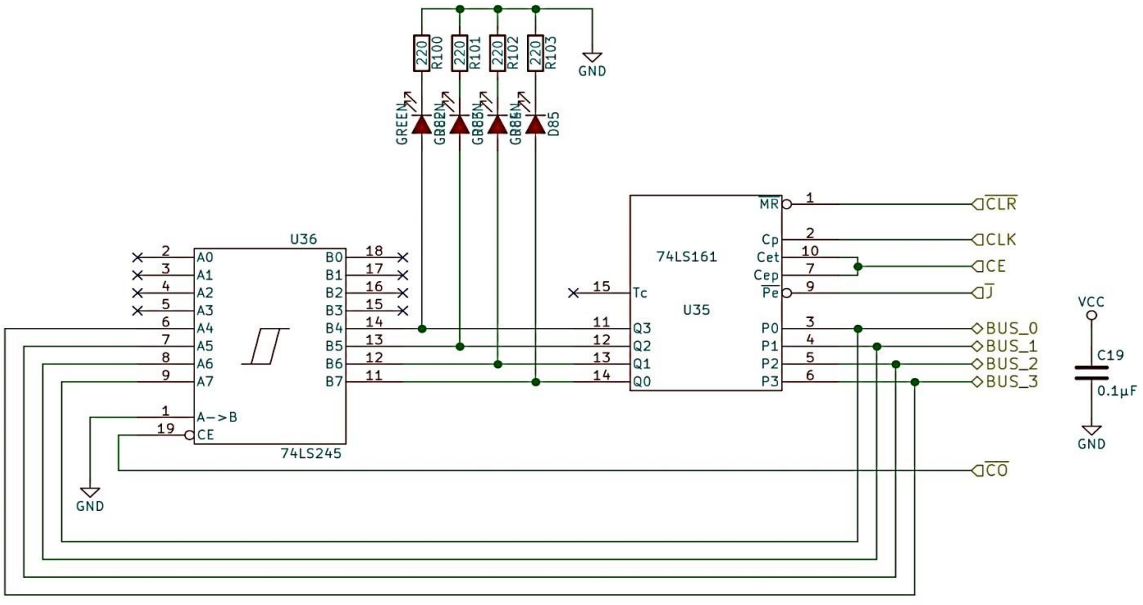

Figure 2. Program counter
Source: https://eater.net/8bit/pc

load a specified address as needed if the program calls for it, as in the case of loops or branches. A schematic of the PC is shown in Figure 2.

Instruction Register

The Instruction Register is split into two distinct 8-bit registers: the Instruction Register (IR) and the Instruction Data Register (IDR). IR is responsible for storing the upper eight bits of the instruction obtained from the program memory. Typically, it contains OPCODE and registers parameters. However, the IDR typically holds 8-bit constant or immediate data utilized by the instruction. Both registers retain their data throughout the execution and writeback stages of the instruction. These are then updated when a new instruction is fetched. A schematic of the IR is shown in Figure 3.

Arithmetic Logic Unit

An ALU is a dedicated functional unit responsible for performing arithmetic and logical calculations. Because this microprocessor has an 8-bit architecture, it primarily employs basic operations. However, it remains feasible to construct more intricate operations by combining these fundamental operations. Furthermore, every ALU operation triggers a status flag based on the outcome of the operation. These status flags provide information about specific conditions or results, allowing for further control and decision-making within the processor.

Control Unit

The control unit was essential to coordinating the activities of all the functional units that came before it. It operates as a state machine, establishing the execution order for each instruction based on the OPCODE. This ensures that the instruction execution stages are properly fulfilled and synchronized. The following factors need to be considered when designing a control unit, which is accomplished by following a state diagram:

1. At start-up, all functional units must have a reset state that considers their initial conditions.

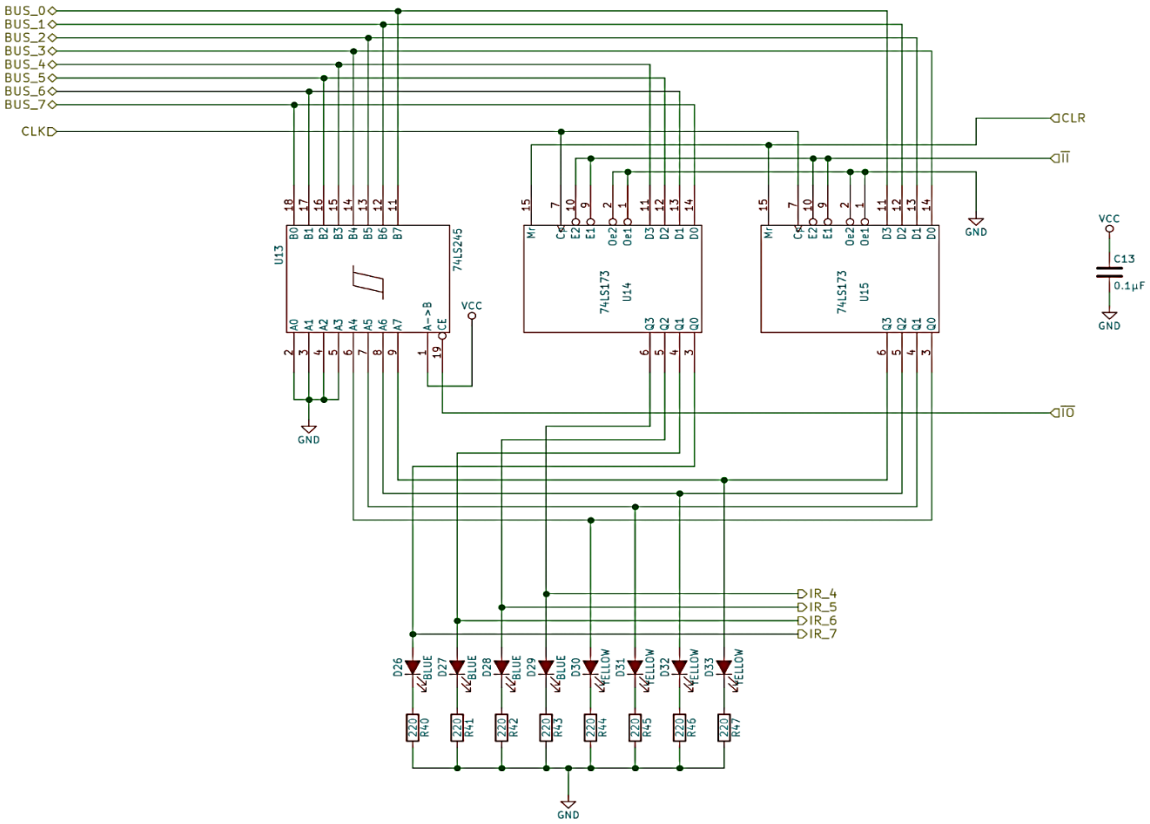

Figure 3. Instruction registers.

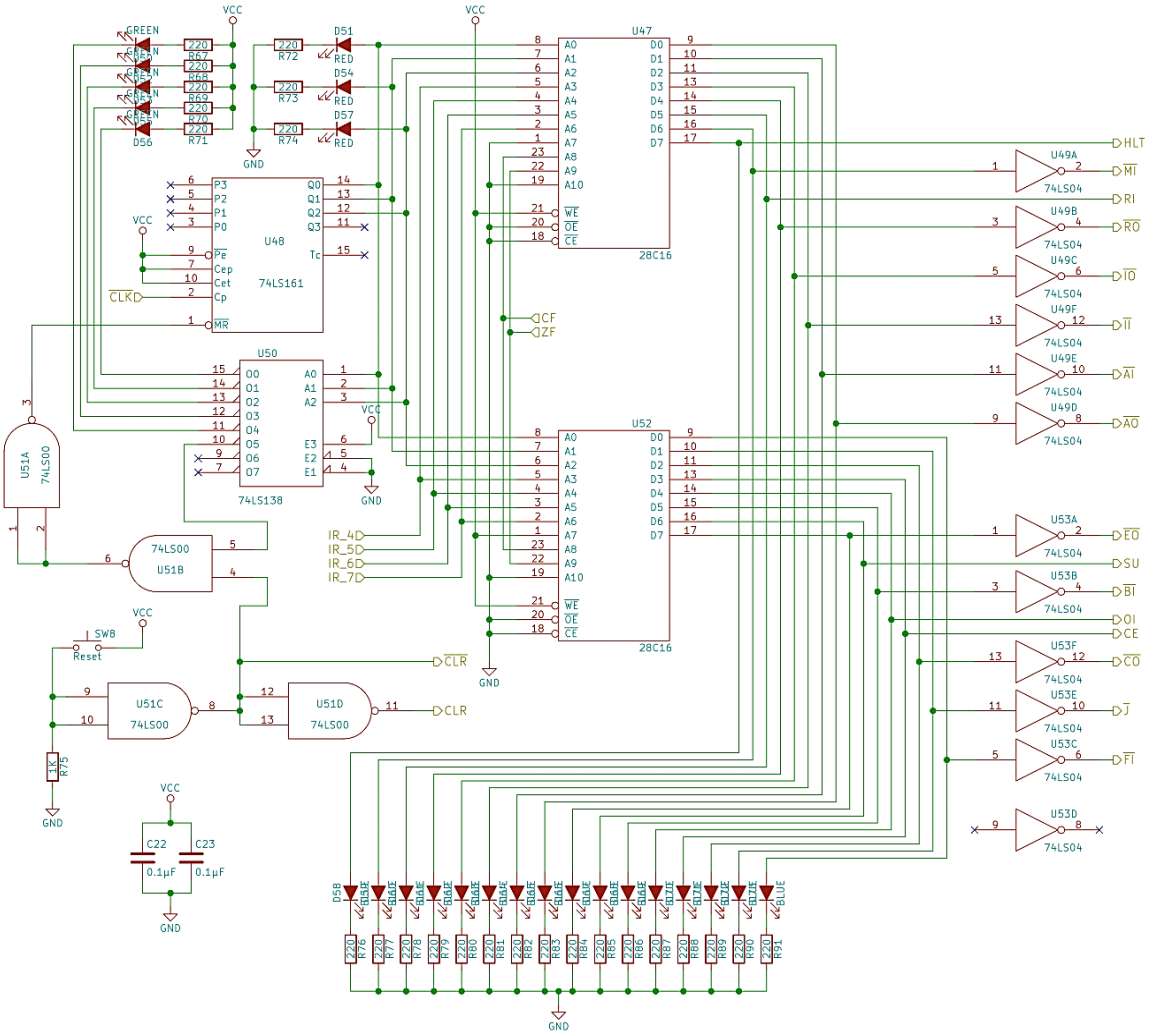

Figure 4. Control unit.

2. The instruction is recovered from the program memory and placed in the instruction registers during the FETCH step, which is represented by the second state.
3. Because the instruction registers function as a sort of gate that allows data to pass through, decoding the instruction is simple after it has been fetched.
4. The following state is equivalent to the EXECUTE stage, during which the instruction performs its primary function:
   i. Interestingly, some instructions, such as branch instructions, do not need to be written back.
   ii. The program counter is increased to address the subsequent instruction in the final state once the outcome of the operations is recorded in the corresponding functional unit. A schematic of the control unit is shown in Figure 4.

DISCUSSION

Over more than six decades, the field of computer architecture education has undergone a significant evolution. Technological progress has had a direct impact on how computer architecture is taught. Although simple hardware is obsolete nowadays, to understand concepts from scratch, cannot be understood directly from the top level; instead, clarity would be more at the circuit implementation level, which would be more useful and easily remember how and what the computer must process at the very initial level. There are a few benefits to our idea compared to other instructional processors. Although it uses the Von Neumann architecture, which has the drawback of using the same memory for data and programs, it is also one of the most straightforward computer architectures.

It uses 16-bit data in this instance, whereas our suggestion uses 8-bit data. In contrast to the former, which utilizes a maximum of three operands, including an accumulator, our approach requires a maximum of two operands, with the accumulator being any general purpose register.

CONCLUSION

In this study, we presented the design and implementation of an 8-bit CPU architecture tailored specifically for undergraduate learning support. Focusing on hands-on experience and practical understanding, our project bridges the gap between theoretical knowledge and real-world applications in computer architecture and digital logic design.

Through the development of this microprocessor trainer kit, we have addressed the pressing need for comprehensive educational tools in the field of CAO. We provide a platform that combines physical hardware emulation with digital logic gates, giving students a rare opportunity to learn about the complex workings of an 8-bit CPU.

Our motivation stems from a desire to empower students with the knowledge and skills necessary to comprehend and innovate computing technologies. By emphasizing practical implementation and experiential learning, we aim to cultivate a deeper understanding of fundamental concepts such as binary arithmetic, instruction decoding, and memory addressing.

The SAP-1 Architecture served as the foundation for our design, leveraging its simplicity and suitability for educational purposes. By carefully considering each component, from general purpose registers to the control unit, we created a comprehensive learning tool that caters to the diverse needs of undergraduate students.

Our project contributes to the evolution of computer architecture education by embracing a hands-on approach and integrating digital design principles into the curriculum. By engaging students in the construction and programming of a functional CPU, we can foster curiosity, exploration, and innovation in the next generation of computer scientists and engineers.

In summary, the creation of our 8-bit CPU architecture for learning support for undergraduates is a major advancement in computer science. We work together, try new things, and provide students with the information and abilities they need to succeed in the rapidly changing computer field.

REFERENCES

1. Olukotun K, Hammond L. The future of microprocessors: Chip multiprocessors' promise of huge performance gains is now a reality. Queue. 2005;3:26–9. DOI: 10.1145/1095408.1095418.
2. Borkar S, Chien AA. The future of microprocessors. Commun ACM. 2011;54:67–77. DOI: 10.1145/1941487.1941507.
3. Gray J. Hands-on computer architecture: Teaching processor and integrated systems design with FPGAs. Rt-PA of the 2000 workshop on Computer architecture education 2000 Jun 1. p. 17–es. DOI: 10.1145/1275240.1275262.
4. Hanafi Ichsan MH, Kurniawan W. CPU implementation using only logisim simulator to achieve computer architecture learning outcome. Bull Electr Eng Inform. 2020;9:747–54. DOI: 10.11591/eei.v9i2.1972.
5. Gray J. Hands-on computer architecture: Teaching processor and integrated systems design with FPGAs. Rt-PA of the 2000 workshop on Computer architecture education 2000 Jun 1. p. 17–es. DOI: 10.1145/1275240.1275262.
6. Hernandez Zavala A, Camacho Nieto O, Huerta Ruelas JA, Carvallo Domínguez AR. Design of a general purpose 8-bit RISC processor for computer architecture learning. Comput Sist. 2015;19:371–85. DOI: 10.13053/cys-19-2-1941.

7. Tocci RJ, Neal S, Moss GL. Widmer Digital Systems: Principles and Applications. 10th ed. Upper Saddle River, NJ: Pearson Education; 1991.
8. Stallings W. Computer Organization and Architecture. 7th ed. Upper Saddle River, NJ: Prentice Hall; 1994.
9. Mano M. Computer Architecture. 3rd ed. Mexico City, Mexico: Prentice Hall; 1994.
10. Jaramillo Gomez G, JA. VHDL: Style Guide and Laboratory Practices for Logic Circuits. Mexico City, Mexico: Editorial Ink; 2011.

