Journal of Engineering and Applied Science Vol. 08, No. 02, pp. 90–96, December 2024

Journal of Engineering and Applied Science

Contents are available at www.jeas.ruet.ac.bd

SP-2: Extending 4-bit SP-1 CPU with Addressing Modes and Interfacing Features on Logisim for Computer Architecture Education

Afsana Afrin[1*], Nahin Ul Sadad[2], Md. Nazrul Islam Mondal[2]

[1]*Institute of Information and Communication Technology, Rajshahi University of Engineering & Technology, Bangladesh*
[2]*Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Bangladesh*

ARTICLE INFORMATION

Received date: 16 Oct 2024
Revised date: 26 Dec 2024
Accepted date: 27 Dec 2024

Keywords

Computer Architecture
Logisim
Learning Support

ABSTRACT

The Computer Organization and Architecture (COA) course often lacks practical activities to help students deeply understand CPU design. To address this, we present SP-2, a 4-bit CPU architecture designed and simulated using the Logisim logic simulator. SP-2 builds on the previously developed SP-1 by introducing significant enhancements, including 40 instructions (up from 33), 5 addressing modes (up from 3, adding Register Indirect, Based, and Based Indexed with Displacement), and Input/Output (I/O) interfacing capabilities, which were absent in SP-1. A comparison with educational CPU models—Mic-1, SEP, DLX, and SP-1—shows SP-2's extended support for ALU operations, branching mechanisms, and I/O interfacing, surpassing SP-1 in versatility. Additionally, SP-2 aligns with most Learning Outcomes (LO) outlined in CS2013 and CE2016 curricula, particularly the Interfacing and Communication outcome, which SP-1 did not support. Despite its limitations, such as the lack of stack support, SP-2 bridges the gap between theory and practical application, offering students a valuable hands-on learning experience in CPU design.

1. Introduction

Computer Organization and Architecture (COA) is a cornerstone course in Computer Science (CS) and Computer Engineering (CE) curricula, as defined by the Association for Computing Machinery (ACM) [1][2]. This course is fundamental to understanding the inner workings of computers, particularly the Central Processing Unit (CPU), which is central to modern computing systems. Achieving the Learning Outcomes (LO) outlined by ACM requires more than theoretical knowledge, it necessitates practical, hands-on experience with CPU design. While COA is critical in both CS and CE programs, it often demands more core hours in CE to meet its LO. The learning media selected for this course must ensure these outcomes are achieved but traditional textbooks and many simulators used in COA courses often focus too heavily on theory, creating a disconnect between high-level abstractions and foundational principles [3].

Integrating CPU design into undergraduate programs bridges the gap between hardware and software. Practical exposure deepens students' understanding of instruction set architecture and system interactions, offering hands-on experience in microarchitecture design [4].

* Corresponding authors: Institute of Information and Communication Technology, Rajshahi University of Engineering & Technology, Bangladesh
E-mail addresses: afsana_afrin@iict.ruet.ac.bd (Afsana Afrin)

The paper is organized as follows: Section 2 reviews related works, Section 3 covers the SP-1 CPU, Section 4 details the SP-2 architecture, Section 5 compares SP-2 with other works, and Section 6 concludes with future directions.

2. Literature Review

The study of computer organization and architecture covers many concepts, but textbooks often emphasize theory over practical applications. Despite students' prior experience as computer users, there's a gap in teaching internal organization of computer [5] due to increasing abstraction layers. The traditional approach to teaching COA involves a theoretical review of the internal components of various CPUs, such as MIPS, x86, and ARM architectures [6]. But Computer Engineering Learning Outcomes require students to go further, designing CPUs that align with architectural principles.

Educators can choose from various educational and practical CPU architectures, typically classified into four categories [7]: Theoretical Architectures focusing on concepts and theory without actual implementation, CPU Simulators simulating CPU operations without focusing on internal hardware, Logic Simulator-based Implementations offering a balance between theory and practice by enabling CPU design through software simulators like Logisim, and finally FPGA & Logic IC-based Implementations providing the most hands-on experience, using physical hardware for CPU design. These options provide different levels of complexity, practical experience, and modification potential. Summary table of these architecture types with their advantages, disadvantages and examples is shown in Table 1:

Table 1. Summary table of types of CPU architecture

Type	Advantages	Disadvantages	Example CPUs
Theoretical Architectures	Simple, concept-focused, no hardware required	No real implementation, lacks hands-on experience	MIC-1 [8], DLX [9], MERIE [10], SAP [11]
CPU Simulators	Simulate CPU behavior, including support for assemblers	Cannot modify or explore internal hardware design	CADSS [12], WepSIM [13]
Logic Simulator-based Implementations	Allow students to design and simulate hardware in software	Limited to virtual design, lacks real-world hardware interaction	MIC-1 [14], 8-bit SEP [7], N2T [15]
FPGA & Logic IC-based Implementations	Hands-on with physical hardware, allows real-world applications	Higher complexity, requires specialized hardware	TINYCPU [16], YASAC [17], VSCPU [18], SVEU16 [19], CHUMP (Logic ICs) [20]

Simulators can be perfect tools for achieving the Learning Outcomes (LO) of a COA course, especially in CE programs. The first step to using them effectively is selecting the right simulator that includes features covering the majority of COA topics [21]. Popular options include Logisim [22], Digital [23], CPU Sim [24] and CPU-OS simulator [25]. Among these, logic simulators, like Logisim, are often preferred over CPU simulators because they help students grasp the underlying working principles of a CPU [26][27]. Positive learning outcomes have been reported when using logic simulators to teach computer architecture through the implementation of specific computer designs [21]. Furthermore, incorporating a series of assignments that gradually lead to the implementation of a computer architecture is beneficial for student learning [14]. Logisim, a GUI-based educational logic simulator, lets users design digital circuits and build a CPU from basic combinational and sequential components, providing a hands-on, comprehensive learning experience.

This paper proposes the design and simulation of a 4-bit CPU architecture, SP-2 (Simple Processor Version 2), using the Logisim logic simulator which is built on the foundation of the previous SP-1 (Simple Processor Version 1). This paper has the following contributions:

1. SP-2 improves upon SP-1 by adding advanced addressing modes (Register Indirect Mode, Based with Displacement Mode, and Based Indexed with Displacement Mode), and Input/Output interfacing, enriching the learning experience for students.
2. It provides a hands-on platform for students to design and simulate a fully integrated 4-bit CPU, progressing from individual components like the ALU, memory, and register set, which helps bridge the gap between theoretical knowledge and practical application in CPU design.
3. SP-2 aligns closely with the Learning Outcomes (LO) specified by the ACM curricula for Computer Organization and Architecture (COA), supporting core educational goals in computer architecture courses.

In this paper, section 3 begins with the Background Study, which covers the previously designed SP-1 CPU architecture. Section 4 then outlines the SP-2 CPU architecture specifications, followed by its design in Section 5. Next, Section 6 compares the SP-2 CPU architecture with other existing works and evaluates whether SP-2 CPU architecture met the Learning Outcomes (LO) defined by the ACM CS and CE curricula. Finally, the paper concludes with a discussion that summarizes the key findings followed by an outline of potential future work.

3. Background Study (SP-1 CPU)

The SP-1 CPU, as outlined in [4], is a 4-bit Reduced Instruction Set Computer (RISC) that utilizes 13-bit machine instructions. In this format, the initial 6 bits are allocated for opcodes, with 2 bits dedicated to identifying the type of instruction and 4 bits specifying the operation itself. This structure not only streamlines the instruction handling process but also supports multiple operational modes. The machine code format is illustrated in Table 2.

Table 2. Machine Code Format of SP-1 CPU [4]

Types	Opcode (6 bits)		Rest (7 bits)	Example ASM Code
	Opcode [5:4]	Opcode [3:0]		
ALU Instructions (Register Mode)	00	0000-1100 (13 operations)	Register 1 (3 bits) + Register 2 (3 bits) + Unused (1 bit)	XOR R2, R3
ALU Instructions (Immediate Mode)	01	0000-1100 (13 operations)	Register 1 (3 bits) + Value (4 bit)	XOR R2, 6
Branching Instructions	10	0000-0100 (13 operations)	Address (7 bits)	JMP LABEL JC LABEL
Memory Instructions	11	0000-0001 (2 operations)	Register 1 (3 bits) + Address/ Displacement (4 bit)	LOAD R0, 2 STORE R1, 10

The Arithmetic and Logic Unit (ALU) of the SP-1 CPU processes 4-bit data and is capable of executing 13 distinct operations. These operations include logical functions like **`AND`** (Opcode: **`0000`**), **`OR`** (Opcode: **`0001`**), and **`XOR`** (Opcode: **`0010`**), as well as other key operations such as **`NOT`** (Opcode: **`0011`**), Shift Left (**`SHL`**, Opcode: **`0100`**), Shift Right (**`SHR`**, Opcode: **`0101`**), Division (**`DIV`**, Opcode: **`0110`**), Multiplication (**`MUL`**, Opcode: **`0111`**), Subtraction (**`SUB`**, Opcode: **`1000`**), and Addition (**`ADD`**, Opcode: **`1001`**). It also supports rotation functions like Rotate Left (**`ROL`**, Opcode: **`1010`**) and Rotate Right (**`ROR`**, Opcode: **`1011`**), alongside a comparison operation (**`CMP`**, Opcode: **`1100`**), which is similar to subtraction but without storing the result. The operation executed by the ALU is determined by the last 4 bits of the machine instruction opcode.

The SP-1 CPU is equipped with 8 general-purpose registers (**`R0-R7`**) for data storage, in addition to two special-purpose registers: the Program Counter (**`PC`**) and the FLAG register. The FLAG register holds two status indicators: the Sign Flag (**`SF`**), which signifies a negative result, and the Zero Flag (**`ZF`**), which is set to 1 if the ALU result equals zero. The CPU supports three distinct addressing modes: Register, Immediate, and Direct. Register and Immediate modes are primarily used for ALU-related tasks, while Direct mode is designated for memory operations.

Furthermore, the SP-1 CPU is capable of executing 24 ALU-based instructions (12 in register mode and 12 in immediate mode), along with 7 branching instructions and 2 memory operations. Each instruction is encoded in machine language using unique opcodes. The CPU can access memory addresses ranging from 0 to 127 using a 7-bit addressing scheme. The memory is structured with a word size of 13 bits, identical to the length of the machine instructions. This memory architecture, which employs Static Random Access Memory (SRAM) built with D Flip-flops, provides learners with a hands-on understanding of how conventional RAM functions.

Control Unit of SP-1 CPU is shown in Table 3.

Table 3. Control Unit of SP-1 CPU [4]

Types	Input				Output					
	Opcode [5:4]	Opcode [3:0]	ZF	SF	Op[3:0]	REG_EN	Imm_Sel	Jmm_Sel	LD_EN	ST_EN
ALU Instructions (Register Mode)	00	AAAA	X	X	AAAA	1 (Except 1011)	0	0	0	0
ALU Instructions (Immediate Mode)	01	BBBB	X	X	BBBB	1 (Except 1011)	1	0	0	0
Branching Instructions	10	0000 (JMP)	X	X	XXXX	0	0	1	0	0
		0001 (JE)	1	0						
		0010 (JNE)	0	X						
		0011 (JL)	0	1						
		0100 (JLE)	SF = 1 or ZF =1							
		0101 (JG)	0	0						
		0110 (JGE)	SF = 0 or ZF = 1							
Memory Instructions	11	0000 (LOAD Direct)	X	X	XXXX	1	0	0	1	0
		0100 (STORE Direct)				0	0	0	0	1

4. SP-2 CPU

This section describes SP-2 CPU Description and Design.

SP-2 CPU Description

The SP-2 CPU converts Memory Instruction type to new Memory & I/O instructions type by adding new memory instructions supporting register indirect, based and based indexed addressing modes and 3 new I/O instructions. Table 4 shows their machine code format, application and example assembly code.

Table 4. Machine Code Format of New Memory/IO Instructions of SP-2 CPU

Operations	Opcode (6 bits)		Rest (7 bits)	Example ASM Code
	Opcode [5:4]	Opcode [3:0]		
LOAD Direct Mode	11	0000 (LOAD)	Register 1 (3 bits) + Address/ Displacement (4 bit)	LOAD RA, [Disp]
LOAD Register Indirect Mode		0001 (LOAD)	Register 1 (3 bits) + Register 2 (3 bits) + Unused (1 bit)	LOAD RA, [RB]
LOAD Based Indexed Mode		0010 (LOAD)	Register 1 (3 bits) + Register 2 (2 bits) +Address/ Displacement (2 bits)	LOAD RA, [RB+Disp]
STORE Direct Mode		0011 (STORE)	Register 1 (3 bits) + Address/ Displacement (4 bit)	STORE [Disp], RA
STORE Register Indirect Mode		0100 (STORE)	Register 1 (3 bits) + Register 2 (3 bits) + Unused (1 bit)	STORE [RB], RA
STORE Based Indexed Mode		0101 (STORE)	Register 1 (3 bits) + Register 2 (2 bits) +Address/ Displacement (2 bits)	STORE [RB+Disp], RA
Waiting for inputs. Send input data to input register.		1101 (ACCEPT_INPPUT)	-	ACCEPT_INPUT
Send data to be printed to output register		1110 (PRINT_OUTPUT)	-	PRINT_OUTPUT
Clear output display		1111 (PRINT_CLEAR)	-	PRINT_CLEAR

In order to add support for interfacing, Input Register is needed to receive input and Output Register is needed to send output. **R6** is now converted to Output Register (**OUT**) which can only be written by CPU and **R7** is now converted to Input Register (**IN**) which can only be written by external input devices.

SP-2 CPU Design

The addressing modes of the SP-2 CPU can be explained using an example of the memory instruction "**STORE**" in the based-indexed mode. In this example, the assembly instruction **STORE [RB+Disp], RA** stores the value of the **RA** register into the memory address calculated as **[RB + Disp]**. First, the register index for **RA** (3 bits) is sent to the **Ra** port of the Register Set, which provides the value stored in the **RA** register to the **A** port. This value is then directed to the **WD** port of the SRAM via a multiplexer for writing operations. Simultaneously, the **RB** register index (2 bits, extended to 3) is sent to the **Rb** port of the Register Set, which sends the **RB** register's value to the **B** port of the ALU. The displacement value (2 bits, extended to 4) is sent to the **A** port of the ALU due to the **Based_Disp_Sel** signal from the Control Unit. The ALU computes the sum of **RB** and the **disp** (**RB+Disp**) and sends this result to the **WA** port of the SRAM via a multiplexer selected by the Control Unit's **ST_Sel** signal. This completes the execution of the STORE instruction. Other memory instructions in SP-2 follow similar execution patterns.

For input/output operations, SP-2 takes input values from four switches, which are sent to the **InRData** port of the Register Set to store the input in the Input Register. If any switch is on, the **INT_INPUT_AVAIL** signal is set to 1 and sent to the Control Unit. SP-2 accepts input only when the **ACCEPT_INPUT** instruction is called, and **INT_INPUT_AVAIL** is active, indicating an interrupt has occurred. The CPU then jumps to the address 01, selected by the **INT_INPUT_SEL** signal, to handle the interrupt. Once SP-2 processes the input, it writes the output to the Output Register, and this output value is sent from the **OutRD** port to a TTY display. The output value is extended to 7 bits and adjusted by adding the hex value 30 to get the corresponding ASCII character. When the **PRINT_OUTPUT** instruction is called, the **INT_PRINT_EN** signal activates interrupting the output device, sending the output to the TTY display. Additionally, the **PRINT_CLEAR** instruction sets the **INT_PRINT_CLR** signal to 1, clearing the TTY display by interrupting.

Control Unit of SP-2 CPU is shown in Table 5. SP-2 CPU design is shown in Figure 1.

* Corresponding authors: Institute of Information and Communication Technology, Rajshahi University of Engineering & Technology, Bangladesh
E-mail addresses: afsana_afrin@iict.ruet.ac.bd (Afsana Afrin)

Table 5. Control Unit of SP-2 CPU

Types	Input				Output											
	Opcode [5:4]	Opcode [3:0]	ZF	SF	INT_INPUT_AVAIL	Op [3:0]	REG_EN	Imm_Sel	Jmm_Sel	Based_Disp_Sel	LD_Sel [1:0]	LD_EN	ST_Sel [1:0]	RAM_EN	INT_INPUT_SEL	INT_PRINT_EN	INT_PRINT_CLEAR
All Previous Instructions	XX	XXXX	X	X	0	XX XX	X	X	X	0	X	X	X	X	X	X	X
Memory & I/O Instructions	10	0000 (LOAD Direct)	X	X	0	XX XX	1	0	0	0	01	1	00	0	0	0	0
		0001 (LOAD Register Indirect)			0		1			0	10	1	00	0			
		0010 (LOAD Based Indexed)			0		1			1	11	1	00	0			
		0011 (STORE Direct)			0		0			0	00	0	01	1			
		0100 (STORE Register Indirect)			0		0			0	00	0	10	1			
		0100 (STORE Based Indexed)			0		0			1	00	0	11	1			
		1101 (ACCEPT_INPUT)			1		0			0	00	0	00	0	1	0	0
		1110 (PRINT_OUTPUT)			0		0			0	00	0	00	0	0	1	0
		1111 (PRINT_CLEAR)			0		0			0	00	0	00	0	0	0	1

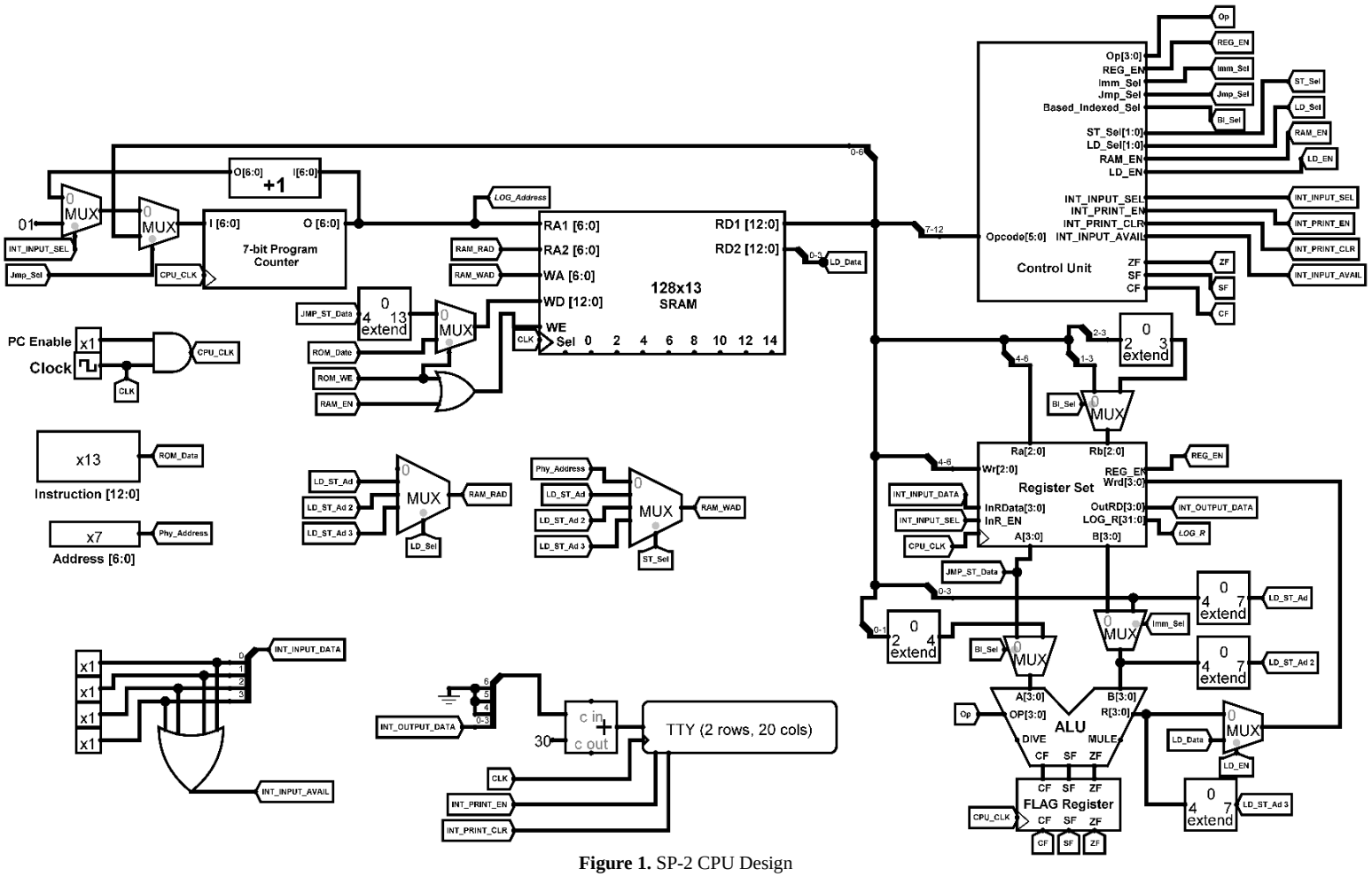

Figure 1. SP-2 CPU Design

5. Comparison with Other Works

In our analysis, we conducted a detailed comparison of the SP-2 architecture with four well-known and widely adopted educational CPU models: Mic-1 (outlined in [8] and implemented in [14]), SEP (both described and implemented in [7]), DLX (presented in [9]), and SP-1 (discussed and implemented in [4]). To validate SP-2's superiority, benchmarking results were carried out by comparing its architectural features with these established models. A summary of the key architectural features from this comparison is provided in Table 6, with parts of the table drawing on insights from reference [7].

Table 6. Comparison of SP-2 with other Works.

Features	MIC-1	DLX	SEP	SP-1	**SP-2**
Word Size of CPU	8	32	8	4	**4**
Registers (Programmable)	2	32	2	8	**8**
No. of Instructions	9	64	21	33	**40**
Addressing Modes	3	5	4	3	**5**
Arithmetic Instructions	2	4	2	5	**5**
Logic Instructions	2	3	1	4	**4**
Shift and Rotate Instructions	0	3	1	4	**4**
Branching Instructions	0	2	4	7	**7**
Flag Support	No	No	Yes	Yes	**Yes**
Stack Support	No	Yes	Yes	No	**No**
Input/Output Support	Both	Both	Both	No	**Both**

As shown in Table 6, in addition to supporting all the features of SP-1, SP-2 introduces several enhancements, including additional addressing modes such as Register Indirect, Based, and Based Indexed with Displacement. It also incorporates Input/Output (I/O) support, broadening the scope of practical application. When compared to other educational CPUs, SP-2 stands out by offering more comprehensive features in arithmetic operations, logical operations, shifting and rotating functions, branching mechanisms, flag handling, and I/O support. The only notable limitation is its lack of stack support. Overall, SP-2 provides students with a hands-on, real-life CPU design experience that bridges theoretical knowledge and practical application.

We also assessed whether our architecture's learning approach helps meet the Learning Outcomes (LO) specified in the Computer Science Curricula 2013 (CS2013) and the Computer Engineering Curricula 2016 (CE2016), as detailed in Table 7 [1][2].

Table 7. Checking SP-2 whether aligns with Learning Outcomes of CS 2013 and CE 2016

Curriculum	Knowledge Units	Can be Learned from SP-1?	**Can be Learned from SP-2?**
CS [2]	Digital Logic and Digital Systems	Yes	**Yes**
	Machine Level Representation of Data	Yes	**Yes**
	Assembly Level Machine Organization	No	**No**
	Memory System Organization and Architecture	Yes	**Yes**
	Interfacing and Communication (Elective)	No	**Yes**
	Functional Organization (Elective)	Yes	**Yes**
	Multiprocessing and Alternative Architectures (Elective)	No	**No**
	Performance Enhancements (Elective)	No	**No**
CE [1]	History and Overview	Yes	**Yes**
	Tools, Standards and/or Engineering Constraints	Yes	**Yes**
	Instruction Set Architecture	Yes	**Yes**
	Measuring Performance	No	**No**
	Computer Arithmetic	Yes	**Yes**
	Processor Organization	Yes	**Yes**
	Memory System Organization and Architectures	Yes	**Yes**
	Input/Output Interfacing and Communication	No	**Yes**
	Peripheral Subsystems	No	**Yes**
	Multi/Many-Core Architectures	No	**No**
	Distributed System Architectures	No	**No**

As shown in Table 7, learning the SP-2 architecture can satisfy most Learning Outcomes (LO) within the Architecture and Organization (AR) section of the Computer Science Curricula 2013 (CS2013) and the Computer Architecture and Organization (CAO) section of the Computer Engineering Curricula 2016 (CE2016). However, CS2013's Knowledge Unit 3 cannot be met because the SP-2 lacks assembler functionality. Moreover, Knowledge Units 7 and 8 in CS2013, along with Units 4, 10, and 11 in CE2016, involve more advanced and theoretical concepts in computer architecture.

6. Conclusion and Future Work

In this paper, we presented SP-2, an enhanced 4-bit version of the SP-1 CPU, designed for undergraduate Computer Organization and Architecture (COA) courses. SP-2 offers a comprehensive instruction set and real-world processor features while remaining simple for students to understand. However, limitations such as the absence of stack support, small 4-bit word size, and lack of advanced features like multiprocessing restrict its capability for complex tasks. Despite these challenges,

the Logisim simulator enables step-by-step design, providing hands-on experience. SP-2's modular design allows future extensions, such as increased word size, stack support, and pipelining, enhancing its relevance for advanced education. The architecture aligns well with most Learning Outcomes (LO) in ACM's Computer Science and Computer Engineering curricula.

Future work will enhance the SP-2 architecture by adding stack support, multi-cycle operations, floating-point instructions, expanded I/O capabilities, and assembler support, bringing it closer to real-world CPU designs. To validate its educational benefits, classroom trials and student feedback will be conducted, providing insights into its effectiveness in improving learning outcomes and advocating for its use in undergraduate courses.

References

[1] Computer Engineering Curricula, Association for Computing Machinery (ACM), IEEE Computer Society, 2016.

[2] Computer Science Curricula, Association for Computing Machinery (ACM), IEEE Computer Society, 2013.

[3] P. W. C. Prasad, A. Alsadoon, A. Beg, and A. Chan, "Using simulators for teaching computer organization and architecture," Journal of Computer Application in Engineering Education, vol. 24, no. 2, pp. 215–224, August 2015.

[4] N. U. Sadad, A. Afrin and M. N. I Mondal, "SP-1: Design and Simulation of 4-bit Simple CPU on Logisim for Computer Architecture Education", 2022 4th International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE), pp. 1-6, Rajshahi, Bangladesh, 2022.

[5] S. Card, "The psychology of human-computer interaction," Taylor Francis, Boca Raton, 2018.

[6] M. H. H. Ichsan and W. Kurniawan, "CPU implementation using only logisim simulator to achieve computer architecture learning outcome," Bulletin of Electrical Engineering and Informatics, vol. 9, no. 2, pp. 749-756, April 2020.

[7] A. Kara and M. Mostefai (2021), "Simple enough processor : From scratch elaborated simulated educational cpu" [Online]. Available: https://doi.org/10.36227/techrxiv.16539843.v1 [Last visited on: 29 Sep., 2024].

[8] A. S. Tanenbaum and J. R. Goodman, Structured Computer Organization, 4th Ed, Prentice Hall PTR, USA, 1998.

[9] D. A. Patterson and J. L. Hennessy, Computer Architecture: A Quantitative Approach, 1st Ed, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[10] L. Null and J. Lobur, Essentials of Computer Organization and Architecture, 1st Ed, Jones and Bartlett Publishers, Inc., USA 2003.

[11] A. P. Malvino and J. A. Brown, Digital computer electronics, 3rd Ed, McGraw Hill, India, 1993.

[12] Brian P. Railing, "CADSS: Computer Architecture Design Simulator for Students," Proceedings of the Workshop on Computer Architecture Education (WCAE '23), pp. 34-40, Orlando, FL, USA, 2024.

[13] F. García-Carballeira, A. Calderón-Mateos, S. Alonso-Monsalve and J. Prieto-Cepeda, "Wepsim: An online interactive educational simulator integrating microdesign, microprogramming, and assembly language programming," IEEE Transactions on Learning Technologies, vol. 13, no. 1, pp. 211–218, March 2020.

[14] D. C. Schuurman, "Step-by-step design and simulation of a simple cpu architecture," Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13), pp. 335–340, Denver, Colorado, USA, 2013.

[15] I. Ilinkin, "Variations on "From Nand to Tetris" with Logisim and ARM," Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2023), pp. 424–429, Turku, Finland, 2023.

[16] K. Nakano and Y. Ito, "Processor, assembler, and compiler design education using an fpga," 2008 14th IEEE International Conference on Parallel and Distributed Systems, pp. 723–728, 2008.

[17] J. Juan-Chico, D. G. Martos, I. M. Gómez-González and J. V. Cortés, "YASAC: Yet Another Simple Academic Computer and Teaching Methodology," 2024 XVI Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica (TAEE), pp. 1-9, Malaga, Spain, 2024.

[18] A. Yildiz, H. F. Ugurdag, B. Aktemur, D. Iskender, and S. Goren, "Cpu design simplified," 2018 3rd International Conference on Computer Science and Engineering (UBMK), pp. 630–632, 2018.

[19] S. Ribić and K. Hodžić, "Educational Processor with Single-cycle Instructions SVEU16," 2023 46th MIPRO ICT and Electronics Convention (MIPRO), pp. 855-860, Opatija, Croatia, 2023.

[20] D. Feinberg, "A simple and affordable ttl processor for the classroom," Computer Science Education, vol. 17, pp. 107–116, June 2007.

[21] T. Stanley, V. Chetty, M. Styles, S.-Y. Jung, F. Duarte, T.-W. Lee, M. Gunter, and L. Fife, "Teaching computer architecture through simulation: (a brief evaluation of cpu simulators)," Journal of Computing Sciences in Colleges, vol. 27, pp. 37–44, April 2012.

[22] C. Burch, Logisim [Online]. Available: https://sourceforge.net/projects/circuit/ [Last visited on: 29 Sep., 2024].

[23] H. Neemann, Digital [Online]. Available: https://github.com/hneemann/Digital [Last visited on: 29 Sep., 2024].

[24] D. Skrien, CPU Sim: An Interactive Java-based CPU Simulator for use in Introductory Computer Organization Classes [Online]. Available: http://www.cs.colby.edu/djskrien/CPUSim/ [Last visited on: 29 Sep., 2024].

[25] B. Mustafa, CPU-OS Simulator [Online]. Available: https://teach-sim.com/ [Last visited on: 29 Sep., 2024].

[26] M. B. Birrer and C. Salazar, "Improving Student Comprehension with Logisim-based RISC Processors," 2022 IEEE Frontiers in Education Conference (FIE), pp. 1-5, Uppsala, Sweden, 2022.

[27] M. Kayaalp, "Using Logisim-evolution for Teaching Datapath and Control," 2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE), pp. 1-8, Raleigh, NC, USA, 2021.