BOBS: an Intel 8085 processor simulator

Antônio Drumond Cota de Sousa, Mateus Henrique Medeiros Diniz, Davi Ferreira Puddo, Henrique Cota de Freitas

Computer Architecture and Parallel Processing Team (CArT)
Department of Computer Science – Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte – MG – Brazil

{adcsousa, mateus.diniz.880541, davi.puddo}@sga.pucminas.br

cota@pucminas.br

Abstract. *To aid in computer architecture education, this paper introduces a new Intel 8085 processor simulator built in the Rust programming language. The project's primary motivation is to leverage a new accumulator-based simulator to compose a set of educational tools based on different architectures. Our simulator offers a high-fidelity emulation of the 8085 architecture, featuring both a terminal and a graphical user interface to cater to diverse learning styles. It includes an integrated assembler and powerful debugging capabilities, most notably a time-travel function that allows users to step both forwards and backwards through their program's execution. The result is a modern, cross-platform, and open-source simulator designed to provide a comprehensive and safe environment for learning low-level computing concepts.*

1. Introduction

Understanding microprocessor fundamentals (Núñez-Mori and Vara, 2023; Nolasco et al., 2023; Liang et al., 2023; Vollmar and Sanderson, 2006) is crucial in computing education, and the Intel 8085's simple architecture makes it an ideal pedagogical tool. Software simulators are essential for providing students with hands-on access to assembly language, allowing them to explore low-level concepts without the need for physical hardware.

Our project contributes to this ecosystem by introducing BOBS, a new Intel 8085 simulator developed entirely in Rust. The primary motivation is to leverage an accumulator-based simulator to compose a set of tools for learning processor architecture. BOBS is a simple architecture that can be discussed in the early stages of education, leaving students engaged to learn more complex architectures.

This paper is organized as follows. Section 2 reviews related work in the field. Section 3 provides an overview of the Intel 8085 processor. Section 4 presents the design and implementation of our simulator. Finally, Section 5 concludes the paper.

2. Related work

Numerous processor simulators have been developed to aid in computer architecture education, often focusing on the MIPS architecture as a pedagogical example. One of the

The name BOBS is derived from the processor's model number, 8085. The letters B, O, and S were chosen for their visual resemblance to the digits 8, 0, and 5.

Antônio, Mateus and Davi are undergraduate students in Computer Science at PUC Minas. The authors thank CNPq and PUC Minas FIP for the support provided in carrying out this scientific initiation.

most prominent is MARS, proposed by Vollmar and Sanderson (2006). Leveraging Java's cross-platform capabilities, MARS became a widely adopted tool. However, its terminal interface can be considered a limitation, highlighting the need for intuitive, non-graphical interfaces for users who prefer a command-line environment.

In a different approach, Liang et al. (2023) developed Sparrow, a Windows-only simulator designed to support a specific textbook ("Computer Composition Principle") with a minimal instruction set of only 15 instructions. On the other hand, focusing on instruction-level parallelism, Nolasco et al. (2023) proposed a RISC-V instruction set simulator to learn Tomasulo's algorithm.

While these simulators have proven valuable, they also highlight an opportunity for a modern, cross-platform simulator focused on classic 8-bit architectures. Our work builds on these ideas by providing a high-fidelity simulation of the Intel 8085 in Rust, emphasizing the correctness afforded by a memory-safe language, and allowing users to choose between a graphical or terminal-based interface.

3. The 8085 Processor

Introduced by Intel in March 1976, the 8085 processor (Núñez-Mori and Vara, 2023) was an enhanced version of its predecessor, the 8080. Key hardware improvements included consolidating to a single +5V power supply and adding four new interrupt pins. To maintain software compatibility, the instruction set remained almost identical, with only a few new instructions added to manage the new interrupt functionality.

This architectural simplicity makes the 8085 an excellent platform for educational purposes. For students new to machine language, managing the dozens of registers or complex instruction sets of modern CPUs can be overwhelming. The 8085 provides a constrained yet functional environment with only nine registers (including the Accumulator and Program Counter) and approximately 60 distinct instructions.

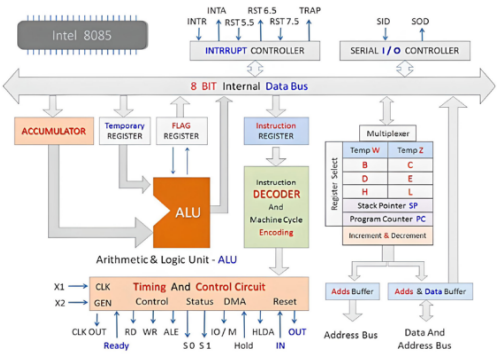

Figure 1. Internal structure of Intel 8085 (Núñez-Mori and Vara, 2023)

This focused design enables students to master the fundamentals of processor operation and low-level problem-solving without the steep learning curve associated with more complex architectures. Furthermore, the 8085 serves as an excellent case study of its era. Its instruction set shares architectural similarities with contemporaries like the Intel 8080 and Zilog Z80, making it a representative example of a classic accumulator-based architecture.

4. BOBS Simulator

BOBS simulator[1] was designed with a modular architecture to faithfully reproduce the behavior of the 8085 while remaining easy to maintain, extend, and integrate with different user interfaces. The Rust programming language was chosen for its powerful default library, high performance, and memory safety. Our implementation emphasizes transparency, allowing students not only to execute machine code but also to observe and understand the processor's internal state at every step.

At the core of the simulator is the central processing unit (CPU) module, which encompasses the register file, flag register, interrupt system, and serial input/output (I/O) pins. Each instruction of the 8085 is implemented as a dedicated function, and a central decoding routine maps binary opcodes to their corresponding behaviors. This design ensures a clear separation between the instruction decoding process and the execution logic, facilitating both testing and future expansion.

Communication between the CPU and other components is mediated by a simulated 16-bit address bus. This bus exposes read and write operations that access the memory module (RAM) and the I/O subsystem. The memory is implemented as a contiguous, byte-addressable array, while the I/O interface follows a similar abstraction, allowing simple integration of peripherals. This layered approach mirrors the actual 8085 architecture, reinforcing the educational value of the tool.

To support user interaction, the simulator provides two complementary front-ends: a terminal interface and a graphical interface, shown in Figure 2. Both allow users to load and execute machine code, inspect internal processor state, and debug their programs. The interface displays the current values of registers, flags, and selected memory cells, giving students immediate feedback on how their code affects the machine.

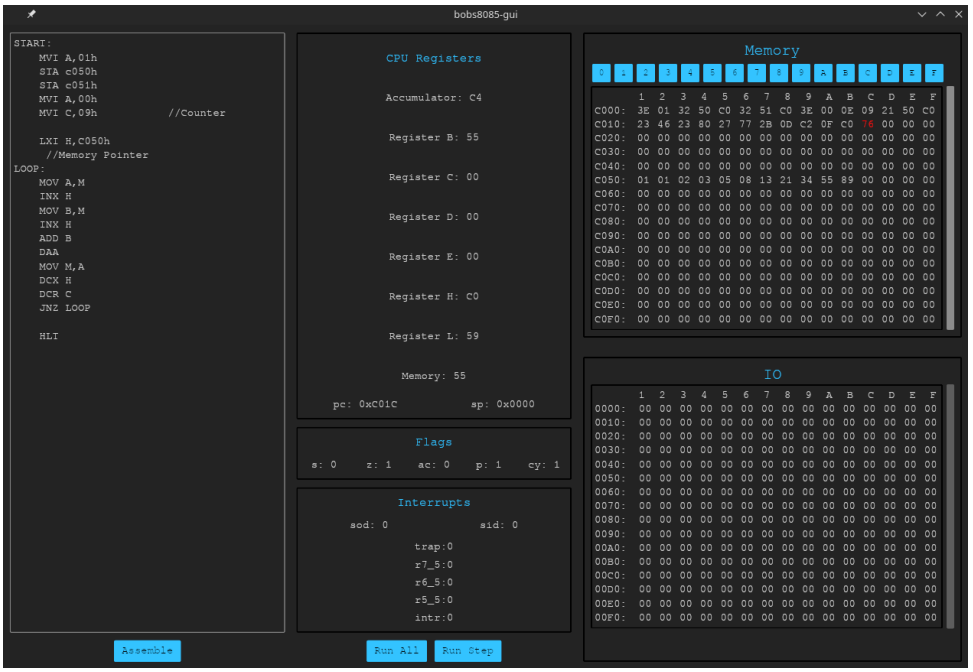

Figure 2. The simulator's interface.

[1]BOBS is available at https://github.com/cart-pucminas/bobs8085

In addition to real-time execution, the simulator supports single-step execution mode. In this mode, users can advance the processor one instruction at a time, pausing between steps to analyze the effects of each operation. Internally, the simulator caches the complete CPU state after every instruction cycle, enabling users not only to step forward but also to roll back to previously recorded states. This time-travel debugging capability is handy for identifying subtle programming errors and understanding control flow.

The simulator's implementation was validated using a test program that executes the complete 8085 instruction set. We confirmed the correct behavior of each instruction by verifying its impact on the CPU's registers, flags, and memory against the processor's documented specifications.

Finally, the simulator includes an integrated assembler component. This assembler supports symbolic labels and descriptive error messages, making it easier for students to write and debug assembly programs without relying on external tooling. Together, these components form a cohesive and educationally focused environment that strikes a balance between realism, usability, and robustness.

5. Conclusion

In this paper, we presented the design and implementation of a new Intel 8085 microprocessor simulator. Developed entirely in Rust, the simulator provides a high-fidelity emulation of the 8085 instruction set, accessible through both a clean and efficient terminal-based interface and a more didactic and illustrative graphical interface.

The main contribution of this work is a modern, cross-platform, and open-source educational tool that leverages the benefits of a modern systems language. BOBS simulator provides a reliable platform for students and hobbyists to explore classic computer architecture. Future work includes enhancing the graphical interface with features such as real-time code linting and adding support for standard 8085 peripherals, as well as conducting a classroom experiment to evaluate the simulator's efficiency as a learning tool.

References

B. Liang, T. Wang, X. Bai, and H. Zhao. Sparrow: A teaching cpu simulator based on windows with graphical user interface. In *2023 7th International Symposium on Computer Science and Intelligent Control (ISCSIC)*, pages 381–386, 2023. doi: 10.1109/ISCSIC60498.2023.00084.

T. Nolasco, D. Vieira, J. Silva, and H. Freitas. Simulador do algoritmo de tomasulo com conjunto de instruções risc-v. In *Anais Estendidos do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 1–8, Porto Alegre, RS, Brasil, 2023. SBC. doi: 10.5753/wscad_estendido.2023.235759. URL `https://sol.sbc.org.br/index.php/sscad_estendido/article/view/26442`.

O. Núñez-Mori and F. Vara. El microprocesador intel 8085 en la educación actual. *UCV Hacer*, 12:97–110, 08 2023. doi: 10.18050/revucvhacer.v12n3a8.

K. Vollmar and P. Sanderson. Mars: an education-oriented mips assembly language simulator. *SIGCSE Bull.*, 38(1):239–243, Mar. 2006. ISSN 0097-8418. doi: 10.1145/1124706.1121415. URL `https://doi.org/10.1145/1124706.1121415`.