ZX16 Pong Game Display and Graphics Rendering

Computer Organization and Assembly – Final Project Report

Mennatallah Essam Zaher Mennatallah Zaid Ahmed Mohamed

July 14, 2025

Course

Computer Organization and Assembly (CSCE 231) – Final Project

Project Overview

The goal of this project was to develop a ZX16 instruction set simulator capable of executing machine code programs and interacting with a 2D graphics system using custom system calls (ECALLs). The simulator was expected to support core ZX16 instructions, graphical rendering, and a GUI interface designed to emulate classic game behaviors.

Our team focused on demonstrating graphical output through a working user interface. Although we were not able to implement the full Pong game logic, we successfully displayed the game title **"Pong Game"** and rendered a paddle on the screen using the ZX16 graphics system. These graphics were driven by binary files and rendered using a GUI built with SFML (Simple and Fast Multimedia Library).

This implementation confirms our system's ability to process ZX16 binary files, execute drawing instructions, and handle system calls within the constraints of our custom graphics engine. While paddle control and ball animation were planned, these features were not completed in time.

Team Members and Contributions

1. Mennatallah Essam Zaher

Main Focus: Registers Class and Test Cases

- Developed the **Registers module** that emulates the 8 ZX16 registers (x0–x7), including proper handling of register x0 as hardwired to zero.
- Implemented logic for updating, reading, and tracking register values during instruction execution.
- Created a suite of **test cases** to verify correctness of arithmetic operations, memory access, branching, and ECALL behavior.
- Validated system behavior before integrating graphics.

2. Mennatallah Zaid

Main Focus: Decoder, Graphics Class, and GUI with SFML

- Implemented the **Instruction Decoder** that translates binary instruction formats into ZX16 operations.
- Built the **Graphics class**, which handled rendering the paddle and the "Pong Game" title using rectangle draw functions.
- Designed and developed the graphical user interface (GUI) using **SFML**, supporting frame updates, screen clearing, and drawing pixel blocks based on memory instructions.
- Enabled proper visualization of graphical ECALLs and ensured binary data was parsed into visual elements on screen.

3. Ahmed Mohamed

Main Focus: ALU Module and ECALL System

- Designed and implemented the **Arithmetic Logic Unit (ALU)**, supporting ZX16 operations like ADD, SUB, AND, OR, XOR, SLL, and SRL.
- Developed the **ECALL system**, enabling:
  1. ECALL #1 – Read String
  2. ECALL #2 – Read Integer
  3. ECALL #3 – Print String
  4. ECALL #4 – Play Tone
  5. ECALL #5 – Set Audio Volume
  6. ECALL #6 – Stop Audio
  7. ECALL #7 – Read Keyboard
  8. ECALL #8 – Dump Register State
  9. ECALL #10 – Exit Program
- Integrated ECALLs with the simulator and ensured graphical ECALLs correctly communicated with the SFML interface.

System Overview

The ZX16 simulator processes binary machine code and interprets it using:

- A functional ALU and register file
- Instruction decoding into RV32I-like formats
- Memory reads and writes
- Graphics instructions that interact with a 2D grid-based canvas

The final output consists of a functioning GUI that displays the paddle and title text ("Pong Game") as graphical blocks, demonstrating the core rendering pipeline through custom system calls and memory-mapped graphics logic.

Limitations

- The game logic (e.g., ball movement, scoring, collision) was not completed.
- User input (keyboard control for the paddle) was planned but not implemented.
- Audio ECALLs were defined but not demonstrated in the final output.
- Only the graphical display portion of Pong was delivered successfully.

Conclusion

This project provided hands-on experience with low-level instruction set simulation, binary file loading, graphical rendering, and modular system design. While the gameplay logic was not implemented, the successful display of graphical elements and GUI rendering validates the core infrastructure of our ZX16 simulator.

The final deliverable demonstrates that ZX16 binary instructions can be decoded, executed, and translated into visual output using ECALL-driven graphics and a modern GUI library.

Future Work

- Add full Pong gameplay: ball physics, scoring, collisions, and paddle control.
- Integrate keyboard input via ECALL #7.
- Add sound effects using ECALL #4 (e.g., bounce sound).
- Extend the simulator to support additional games or programs.
- Improve GUI responsiveness with frame control and smoother redraws.