CHIP 8 emulator

Ronan Bocquillon, Pierre Gaucher, Nicolas Monmarché

Polytech Tours
64 avenue Jean Portalis, F-37200 Tours

1 Introduction

CHIP 8 is an interpreted programming language using a very simple virtual machine. Made up of only 35 instructions, it was developed by Joseph Weisbecker in the seventies to allow video games to be easily programmed and of course run on low range computers of the time. Still popular among nostalgic retrogamers, many iconic video games have been ported to CHIP 8 for the last fifty years, such as Pong, Space Invaders, Tetris and Pacman.

In this project, you will write a fully compliant CHIP 8 emulator. Your program is expected to allow any CHIP 8 ROM file to be read and run (see Figure 1). This implies simulating the virtual machine on a modern architecure and supporting all the 35 possible CHIP 8 instructions.

2 Recommended roadmap

To achieve this goal, you may follow this roadmap:

1. First read this document to its end!
2. Then read carefully Cowgod's CHIP 8 technical reference. This clear and synthetic document describes the virtual machine and the 35 original instructions. Note that sections 2.1 and 2.2 merit special attention, as memory and registers will be implemented first.
3. Implement RAM as a data structure encapsulating an array of 4096 bytes (that is 4096 8-bit unsigned integer variables). Define functions to read from and to write to memory at a given address. Use appropriate data types for data and addresses.

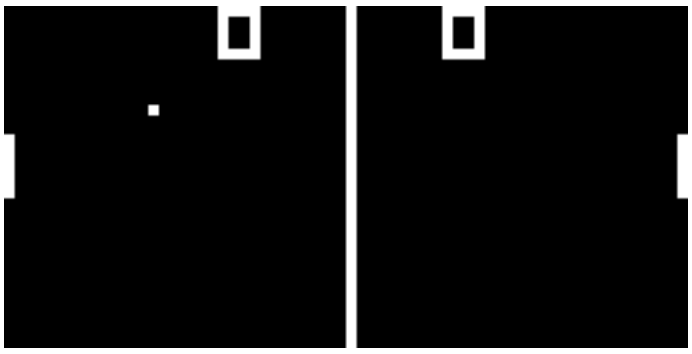

Figure 1: Screenshot of Pong implemented in CHIP 8 [wikipedia].

4. Implement the processor as a data structure encapsulating general purpose registers, timers, a program counter and a stack. Define functions to manipulate a processor. Look at technical specifications to implement relevant operations. For example, you may want to write functions manipulating the stack with a view to implementing instructions `RET`, `JP` and `CALL`.

5. Inspect provided source files to figure out how the display and the keyboard are emulated.

6. Implement your emulator as a data structure encapsulating the processor, memory, a display and a keyboard. Define functions to:

   - initialize the emulator (*e.g.* load predefined sprites),
   - read a program/ROM, that is a binary file containing a sequence of 16-bit instructions, and load it into (virtual) memory,
   - implement the fetch-decode-execute loop (you may want to handle timers here).

7. Complete the `main` function and test your CHIP 8 emulator with Timendus' test suite, from the easiest to the hardest ROMs. Update your code until all issues are resolved.

8. Have fun!

3 Specification notes

CHIP 8 has evolved over time and Cowgod's technical reference is not fully compliant with original specifications. Consequently, you may ended up with an interpreter that does not pass Timendus' tests. Do not worry about this before Step 7; modifications will always be small.

4 Disclaimer (a word to the wise)

For sure you will find plenty of resources on the internet to help you carrying out the project, which is fine. You may come across repositories containing high quality code. Besides, ChatGPT is doing a great job on writing a CHIP 8 emulator...

Please, do not yield to temptation! Your evaluation mostly depends on an oral exam, and you should better understand and master a small/unfinished code, rather than the opposite.