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.
3. Download all the provided resources from CELENE. Prepare a C projet to edit, compile and eventually run the given test program (the `go` function). This requires configuring the project to link with both a provided library, named `libprovided`, and the SDL 2.

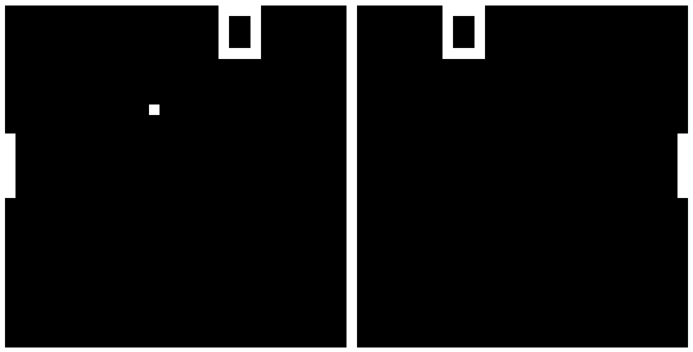


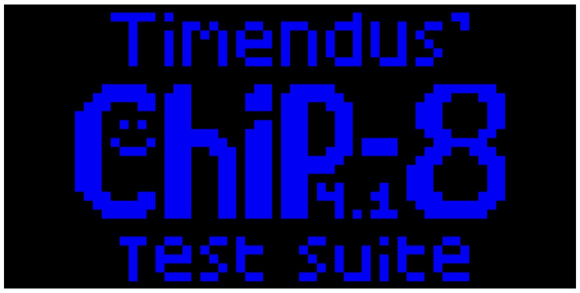
Figure 1: Screenshot of Pong implemented in CHIP 8 [wikipedia].

4. Create an empty *private* project on the git repository of the university, clone it locally, then prepare your initial commit: add relevant files (*e.g.*, `.workspace`, `.project`, `.c`) and exclude all others (*e.g.*, the generated `Makefile`, binary files, external libraries). Push it to the remote repository. You are now ready to go!

5. First implement the RAM, as a data structure encapsulating an array of 4096 bytes (that is an array of 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.

6. Define a function that reads a CHIP 8 ROM, that is a binary file composed of a sequence of 16-bit CHIP 8 instructions, and loads it into memory from a given address (using 2 bytes per instruction). Download the first ROM of the Timendus' test suite (the CHIP 8 splash screen ROM). Write a test program that loads this ROM into memory, reads the instructions from memory, and eventually outputs the program with the `instruction_to_str` function defined in `libprovided` (`misc/debug.h`). Push your work to your remote git repository.

7. Implement the processor as a data structure encapsulating an array `Vx` of 16 general purpose 8-bit registers, a 16-bit address register `I`, a program counter `PC`, a pointer to the RAM, and a pointer to a `Display` (`display/display.h`). Define a function that performs one iteration of the fetch-decode-execute loop. Only try to handle the instructions that are required to run the Timendus' CHIP 8 splash screen ROM. Finish off your test program, so that it runs the loaded ROM for at least 39 cycles. You should get the following result:

   Congratulations! Push your work to your remote git repository.

8. Try now to run the other Timendus' test ROMs, one after the other, from the easiest to the hardest, until you get a fully compliant CHIP 8 emulator. To this end, you will have to:

   - progressively implement the missing instructions,
   - emulate the timers,
   - integrate the provided keyboard and speaker.

   Regularly push your work to your remote git repository.

9. Try some games. Have fun!

3 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.

As a reminder, plagiarism is forbidden by law.