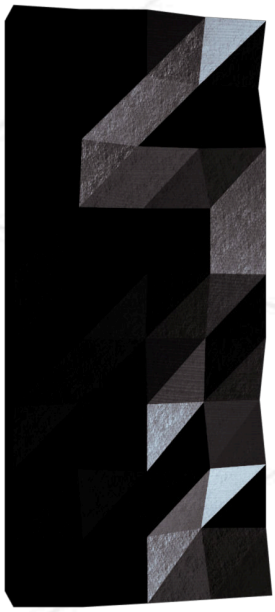

GBMU

Summary: You've probably played on a GameBoy™ emulator to play again games that marked your childhood. Today it's time to learn how it works and develop your own.

Version: 3

Contents

Chapter I

Forewords

Chapter II

Introduction

This subject aims to make you build a GameBoy™ / GameBoy™ Color emulator. In order to do this you will need to correctly reproduce the functionalities of the GameBoy™ using the documentation provided by Nintendo®.

A video game console emulator is a type of emulator that allows a computing device, usually a personal computer, but also other video game consoles and mobile devices running operating systems such as Android to emulate a video game console's hardware and behavior and play games for that platform. Emulators are most often used to play older video games on personal computers and video game consoles, but they are also used to play games translated into other languages or to modify (or hack) existing games. More often than not, emulators offer additional features above and beyond that of the original console, such as multi-controller compatibility (such as PSX controllers being used with N64 games and vice versa), timescale control, higher framerates, higher resolutions, unlocking of gameplay features, memory modifications (like GameShark), and one-click cheat codes. Emulators are also a useful tool in the development process of homebrew demos and the creation of new games for older or discontinued consoles.

Code and data of a game are typically supplied to the emulator by means of a ROM file (a copy of the data contained on a game cartridge) or an ISO image (for systems that use optical media). Most game titles retain their copyright even with the original system and games being many years past discontinuation and increasing rarity, so many resort to the obtaining of these games for free on various internet sites rather than purchasing and ripping the ROM from the game (although, this is popular among those who already own the games). Specialized adapters such as the Retrode allow emulators to directly access the data on game cartridges without the need to copy it into a ROM image first.

Chapter III

Objectives

This project has severals goals :

- Develop a real application from end to end
- Discover the principles of emulation
- Learn to read electronic documentation
- Play games from your childhood!

Chapter IV

General instructions

- This project will be evaluated only by humans
- You must have a GUI with at minimum: load, play, pause
- The project will be compiled on your machine, however it will be corrected on the machine of the corrector. The project will be launched a simple double-click (any other manipulation won't be tolerated).
- You must build an application for your linux distribution, ie your bundle will ship the dependencies required for its operation (Libraries, Frameworks, etc).

```
man install_name_tool
```

Chapter V

Mandatory part

V.1 CPU

The basic hardware emulation of the Game Boy™ starts with a good implementation of the CPU, ie the registers and instructions. You must also have a debugger allowing at least to:

- display the status of the registers
- display the next instruction to execute
- execute an instruction

V.2 LCD

What would be a Game Boy™ without display? Not a great console!
You must correctly implement the display system of the Game Boy ™. The debugger will execute a frame and/or one second in order to prove that there is some changes on the screen.

V.3 Running thread and timing

Now that the CPU and display are in place, it's time to make it all work together. Your emulator will run at normal speed without adversely affect the operations of the GUI.

V.4 Inputs

Playing the intro is nice though it's better to play !
You must correctly implement the controls of the GameBoy™: control pad and buttons.

V.5 MBCs

In addition to ROMs only, you will discover that there are several types of cartridges with different characteristics. You will need to correctly implement some of them:

- MBC1
- MBC2
- MBC3
- MBC5

You must also manage the in-game backup for games that propose this feature.

V.6 CGB

Black and white was cool until 1967!
Now that the GameBoy™ has no secrets for you, it is time to move to the GameBoy™ Color. These two machines are quite similar to a few details, you will have the implement both of them.

Chapter VI

Bonus part

This subject does not offer a free bonuses, only the following bonuses will be counted.

- Sound :
  As part of this bonus, you have the right to use a library to manage the sound as the emulation of a synthesizer is not the objective of this subject.
- BIOS :
  You must manage the GameBoy™ bootstrap (scrolling logo) and GameBoy™ Color.
- Forcing DMG/CGB :
  You must be able to force a GameBoy™ model so you can launch a DMG game on the GameBoy Color™ color and vice versa.
- Save States :
  You must be able to save the state of your game to resume it later.
- UX :
  Lots of super cool features enriching the user experience. examples:
  - reset
  - drag&drop
  - speed
  - etc.

`The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.`

Chapter VII

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.

- Only ROMs provided in resources will be tested and will count when assessing
- he project will be compiled on your machine, however it will be corrected on the machine of the corrector. The project will be launched a simple double-click (any other manipulation won't be tolerated).
- You must build an application, ie your bundle will ship the dependencies required for its operation (Libraries, Frameworks, etc).