Virtualisation - Video game system emulation

Christophe Simon & Maxime Bourguignon

January 7, 2020

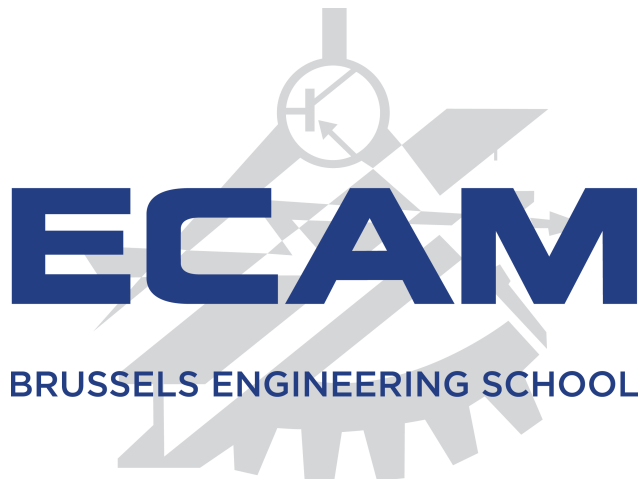


Table of Contents

Objectives

At the end of this presentation about *emulation*, you should be able to understand:

- The *challenges*
- The *operation*
- The main *disadvantages* and *limits* (performances, costs, ...)
- The *Game Boy* example

What is Emulation

"An emulation is an imitation of software or hardware, which gives you the ability to use a product, even if you don't have the original. An emulation has to imitate the real product as much as possible and even replicate known errors."

Troels Ynddal, Mads Ynddal, Asger Lund Hansen [2016]

Types of emulation 1/2

- Software emulation

It simulates the hardware with *a layer of software* on top of *a general hardware*.

- Hardware emulation

It simulates the hardware with *a layer of dedicated (programmable) hardware* like coprocessor or FPGA.

Types of emulation 2/2

	Software emulation	**Hardware emulation**
Widespreadness	+	-
Price	-	+
Efficiency	-	+

This is a tendency to use with caution as it depends on the system to emulate.

Challenges of the Emulation

- Emulate the hardware

It's easier to emulate a hardware if we have *a similar hardware* with *the same set of instructions* and *the same handling errors*.

- Emulate the OS

It needs an OS with *an identical process scheduling queue* and *same environment* for the programs to run.

- Respond like the original

For gaming, it needs *the same behaviour* than the original console. It needs *the same "speed"* for the frames and for the gamer inputs responses.

Limits of emulation

Emulation depends on two main points of the system to emulate:

- Knowledge

The first step is *reverse engineering*, *the understanding* of *original device* operation (PCB, chips, software, OS, etc.).

- Hardware power

The hardware must be *powerful enough* to handle the OS and the gaming emulation. The emulation's application needs to run faster than the original; the system *bottleneck must be the emulated clock*.

Consoles emulation status

Console	Reverse-engineering	Powerfull hardware
Console without OS		
Game Boy	*v* - Lot of doc.	*v* - Old hardware
Playstation 2	*v* - Lot of doc.	*v* - Old hardware
Console with OS		
Switch	*v* - No security	*v* - No powerful hardware
Playstation 3	*x* - Security	*o* - Actual hardware
Playstation 4	*x* - Security	*o* - Actual hardware
Xbox 360	*x* - Lost doc.	*v* - Old hardware
Xbox One	*x* - Security	*o* - Actual hardware

Difficulty color guide: easy, medium, hard

Playstation 1/2

Emulation of Playstation 1 with Playstation 2

- Dedicated hardware

The "Fat" Playstation 2 has a coprocessor to run PS1 games. The emulation is also possible because they share a similar hardware architecture.

- Dedicated software

The "Slim" Playstation 2 uses software to emulate the PS1. The hardware of the "Slim" PS2 is more powerful than the "Fat" PS2.

Playstation 2/2

Xbox

Emulation of Xbox 360 with Xbox One

- Dedicated hardware

The Xbox One has specifics chips for Xbox 360 emulation: A Xbox 360 VGPU dedicated to Xbox 360 emulation, etc.

- Dedicated software

The Xbox One has a dedicated software to emulate the Xbox 360 OS.

- *Recompile* the game

The games are recompiled to match the Xbox One architecture.

Example: Game Boy

- Software emulation: mGBA, VisualBoyAdvance, etc.

It uses software on computer to emulate the GB/GBA hardware.

- Hardware emulation: Analogue pocket

It uses FPGA to emulate the hardware of the GB/GBA.

- Backwards compatibility: GB/GBC/GBA

The GB and GBC shares the same hardware. The GB/GBC games are backwards compatible due to a coprocessor in the GBA.

Reverse engineering

Specific hardware of the Game Boy

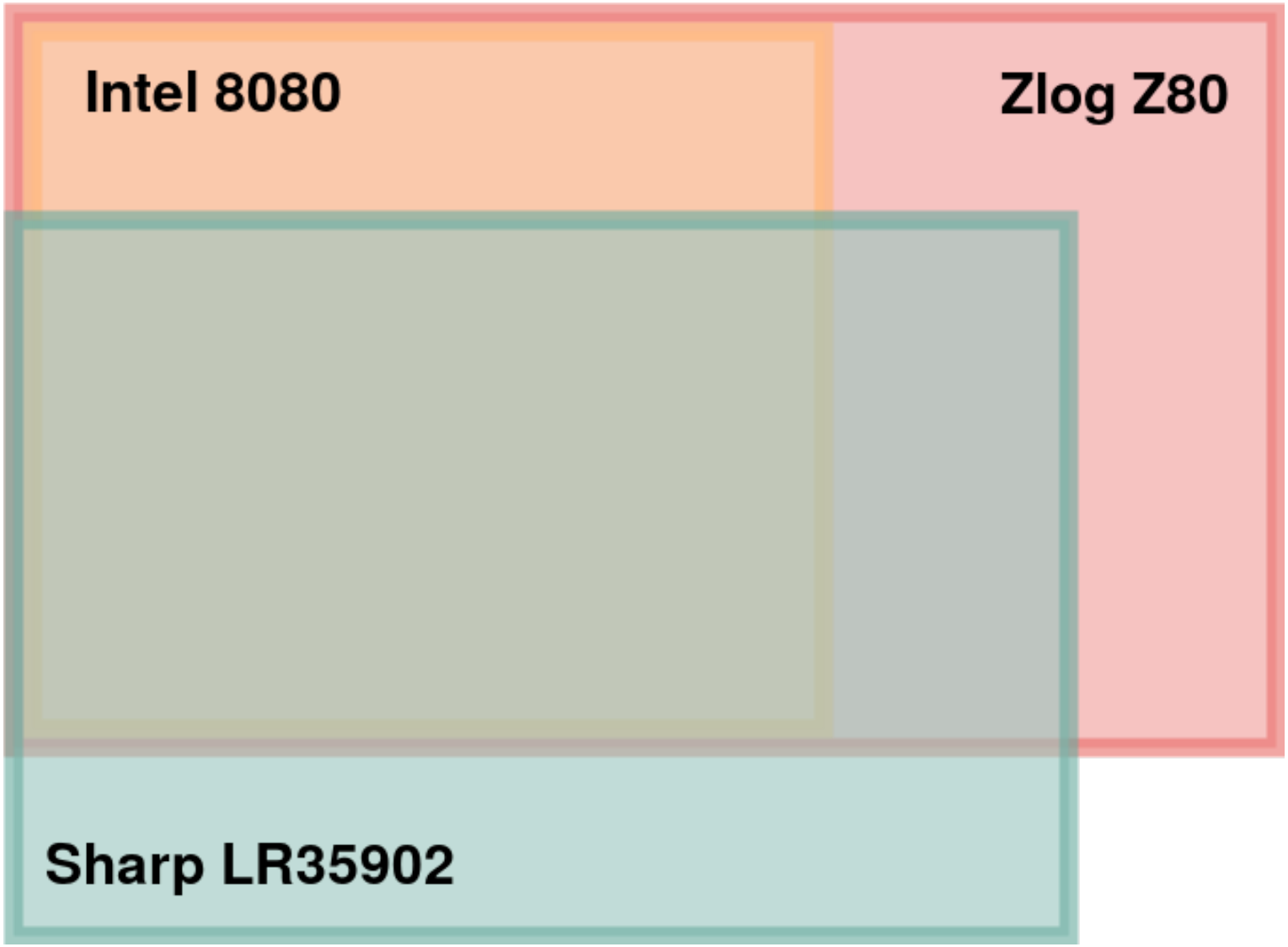
- CPU;

It is a eigth bits Sharp LR35902 control process unit with eight registers. The clock frequency is 4.19 MHz.

- ROM;

There are 256 bytes of bootstrap on the device. The rest is on game cartridges (different size and content for each game).

Set instruction



Codes: registers

```rust
pub struct Registers {
    pub a: u8,
    pub b: u8,
    pub c: u8,
    pub d: u8,
    pub e: u8,
    pub f: FlagsRegister,
    pub h: u8,
    pub l: u8,
}
```

[4]

Codes: instructions

```rust
pub enum Instruction {
    INC(IncDecTarget),
    DEC(IncDecTarget),
    ADD(ArithmeticTarget),
    SUB(ArithmeticTarget),

    AND(ArithmeticTarget),
    OR(ArithmeticTarget),
    XOR(ArithmeticTarget),

    JP(JumpTest),

    LD(LoadType),

    PUSH(StackTarget),
    POP(StackTarget),

    HALT,
    NOP,
}
```

[4]

Codes: instructions mapping

```
0x87 => Some(Instruction::ADD(ArithmeticTarget::A)),
0x80 => Some(Instruction::ADD(ArithmeticTarget::B)),
0x81 => Some(Instruction::ADD(ArithmeticTarget::C)),
0x82 => Some(Instruction::ADD(ArithmeticTarget::D)),
0x83 => Some(Instruction::ADD(ArithmeticTarget::E)),
0x84 => Some(Instruction::ADD(ArithmeticTarget::H)),
0x85 => Some(Instruction::ADD(ArithmeticTarget::L)),
0x86 => Some(Instruction::ADD(ArithmeticTarget::HLI)),
0xc6 => Some(Instruction::ADD(ArithmeticTarget::D8)),

0x97 => Some(Instruction::SUB(ArithmeticTarget::A)),
0x90 => Some(Instruction::SUB(ArithmeticTarget::B)),
0x91 => Some(Instruction::SUB(ArithmeticTarget::C)),
0x92 => Some(Instruction::SUB(ArithmeticTarget::D)),
0x93 => Some(Instruction::SUB(ArithmeticTarget::E)),
0x94 => Some(Instruction::SUB(ArithmeticTarget::H)),
0x95 => Some(Instruction::SUB(ArithmeticTarget::L)),
0x96 => Some(Instruction::SUB(ArithmeticTarget::HLI)),
0xd6 => Some(Instruction::SUB(ArithmeticTarget::D8)),
```

[4]

Codes: memory bus

```rust
pub struct MemoryBus {
    boot_rom: Option<[u8; BOOT_ROM_SIZE]>,
    rom_bank_0: [u8; ROM_BANK_0_SIZE],
    rom_bank_n: [u8; ROM_BANK_N_SIZE],
    external_ram: [u8; EXTERNAL_RAM_SIZE],
    working_ram: [u8; WORKING_RAM_SIZE],
    zero_page: [u8; ZERO_PAGE_SIZE],
    pub gpu: GPU,
    pub interrupt_enable: InterruptFlags,
    pub interrupt_flag: InterruptFlags,
    timer: Timer,
    divider: Timer,
    pub joypad: Joypad,
}
```

[4]

Conclusion

The emulation depends drastically on original and target systems.

- Reverse engineering $\rightarrow$ knowledge

This is the main job to develop an emulator.

- Two ways to implement the solution

It is mandatory to select the best option to emulate the original device: hardware or software (most common).

Due to technological limitations and lack of knowledge, the emulation of all devices is not necessarily possible !

Questions

Do you have any questions ?

Bibliography

[1] Analogue, Novembre 2019. https://www.analogue.co/pocket/.

[2] Ben Allen. A top-to-bottom explanation of game emulation, February 2018. https://infinigeek.com/top-bottom-explanation-game-emulation/.

[3] Vicki Pfau Antonio Vivace. Github: awesome-gbadev, Novembre 2019. https://github.com/gbdev/awesome-gbadev.

[4] Ryan Levick. Dmg-01, 2019. https://github.com/rylev/DMG-01.

[5] Vicki Pfau. Github: mgba, Novembre 2019. https://github.com/mgba-emu/mgba/graphs/contributors.

[6] Vicki Pfau. mgba emulator, October 2019. https://mgba.io/.

[7] Margaret Rouse. Emulation, September 2006. https://whatis.techtarget.com/definition/emulation.

[8] Asger Lund Hansen Troels Ynddal, Mads Ynddal. Emulation of nintendo game-boy (dmg-01), January 2016.

[9] Antonio Vivace. Github: awesome-gbdev, Novembre 2019. https://github.com/gbdev/awesome-gbdev.

[10] Wiki. Playstation developer wiki, 2016. https://www.psdevwiki.com/ps3/Landing_Page.