
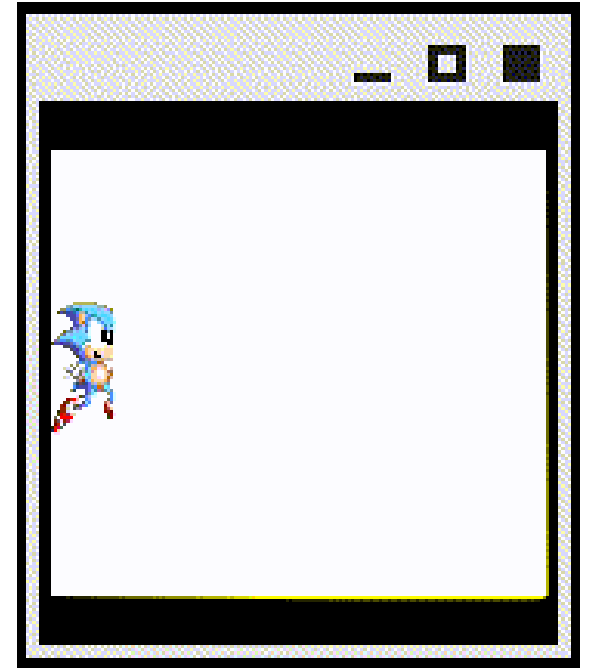


How do you write an emulator  
anyway ?

# Whoami

- Anisse Astier
-  @Aissen@treehouse.systems
- <https://anisse.astier.eu>
- Writing gears, a Game Gear emulator
- Not an emulation expert
- Presented Z80 : the last secrets in 2022 and WASM101 yesterday



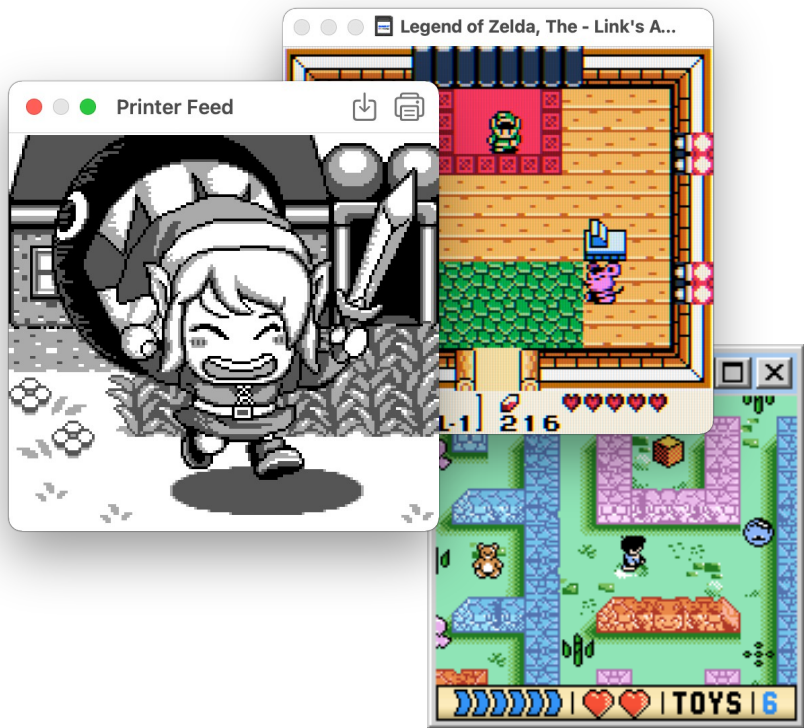
# Why this presentation

- Will :
  - Give pointers
  - Help you start
- Not :
  - Exhaustive
  - Why
- Focus on simpler (8bits) platforms

# What is an emulator ?

- A software program
- To run software from another computer

# What is an emulator ?



FOSDEM 2024



Anisse Astier

How do you write an emulator anyway ?

# What is an emulator ?

- Spectrum of emulation and accuracy
  - One software
  - All software
  - Clock accurate

# A crazy example

The screenshot shows the Scratch IDE interface with a Linux on Scratch emulator running. The top menu bar includes Settings, File, Edit, Addons, Advanced, Linux on Scratch rv32 emul, See Project Page, TurboWarp Feedback, and Save to your computer. The left sidebar shows the Scratch block palette with categories like Motion, Looks, Sound, Events, Control, Sensing, Operators, Variables, and My Blocks. The main workspace displays a Scratch script with several 'go to random position' and 'glide' blocks. A terminal window is open, showing the output of a 'coremark' benchmark. The terminal output includes performance metrics such as CoreMark Size (666), Total ticks (13252), Total time (13.252000), Iterations/Sec (4.527618), and Compiler flags. The terminal also displays a welcome message for Linux on Scratch and instructions on how to use the 'ed' text editor and 'duktape' JavaScript engine. The bottom right of the IDE shows the Stage area with a RISC-V sprite, size, and direction controls, and a 'Run' button.

```
FlzzBuzz
91
92
Flzz
94
Buzz
Flzz
97
98
Flzz
Buzz
~ #
~ # coremark
2k performance run parameters for coremark.
CoreMark Size : 666
Total ticks : 13252
Total time (secs): 13.252000
Iterations/Sec : 4.527618
Iterations : 60
Compiler version : GCC12.2.0
Compiler flags : -march=rv32ima -mabi=ilp32 -fPIE -pie -Os -s -static -Icoremark-1.01 -DPERFORMANCE_RUN=1 -Icoremark-1.01/linux
Memory location : Please put data memory location here
(e.g. code in flash, data on heap etc)
seedcrc : 0xe9f5
[]crclist : 0xe714
[]cncmatrix : 0x1fd7
[]cncstate : 0x8e3a
[]cncfinal : 0x14c
Correct operation validated. See readme.txt for run and reporting rules.
CoreMark 1.0 : 4.527618 / GCC12.2.0 -march=rv32ima -mabi=ilp32 -fPIE -pie -Os -s -static -Icoremark-1.01 -DPERFORMANCE
Run! -Icoremark-1.01/linux / Heap
~ # cat readme.txt
Welcome to Linux on Scratch!

This is overall a pretty bare-bones Linux installation, but there's a few cool things installed you can check out.

You can use the 'ed' text editor to edit files.
You can use 'duktape' to run JavaScript files, try out 'duktape fizzbuzz.js'!
There's also a benchmarking utility installed called 'coremark' which can be used to test the speed of the RISC-V emul
~ # echo hello fosdem
hello fosdem
~ #
```

GAME  
GEAR

POWER

SEGA  
PORTABLE  
VIDEO GAME  
SYSTEM



x01

STARTING



# Where to start ?

- Pick a target
- Pick a host platform
- Make sure you have time

# Where to start ?

- Start simple: one CPU instruction
  - Debug code, minimal disassembler : can you recognize the instruction ?
  - Execution : run it, model CPU state and change it
  - Test : verify state change
- Learn CPU concepts as you go


# Interesting CPU concepts

- State: registers
- Instruction
  - Assembly
  - Encoding
- Interrupts
- Memory access

GAME  
GEAR

POWER



 x02 | STRUCTURE

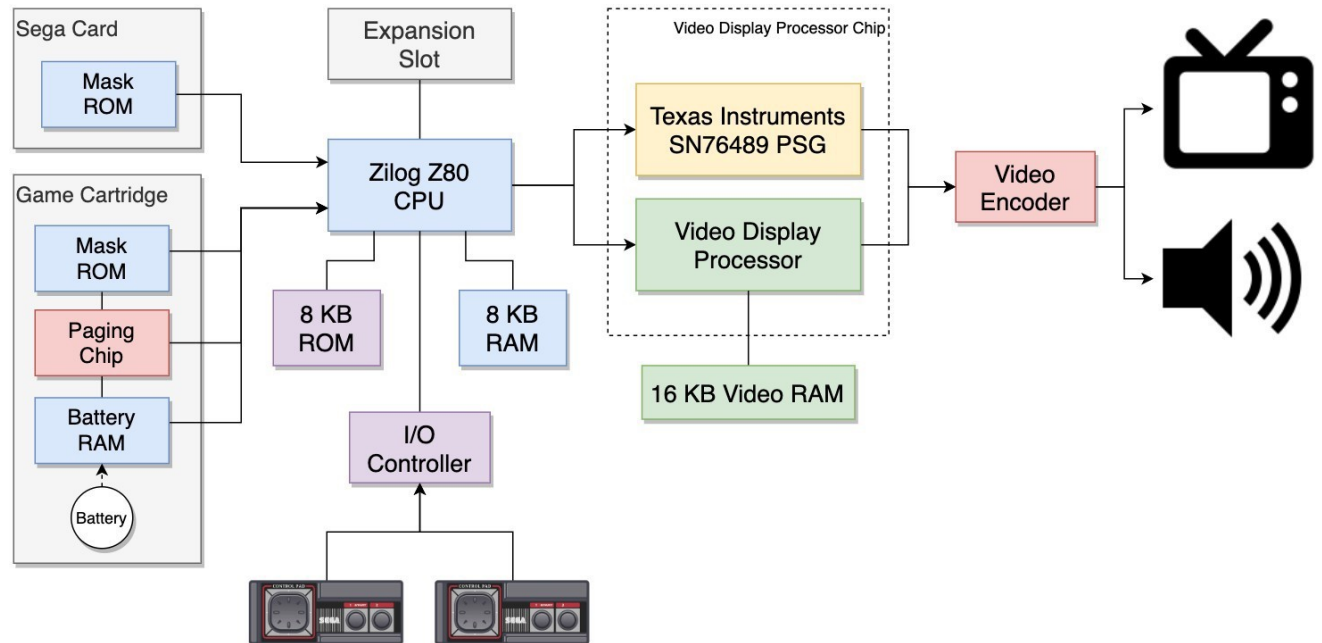
SEGA  
PORTABLE  
VIDEO GAME  
SYSTEM

# Emulator structure

- Examples of 8-bit consoles
  - NES, SMS, GB, Atari 2600, Game Gear

- Various parts :

- CPU
- Display
- Sound
- Memory
- ROM mapping
- Input

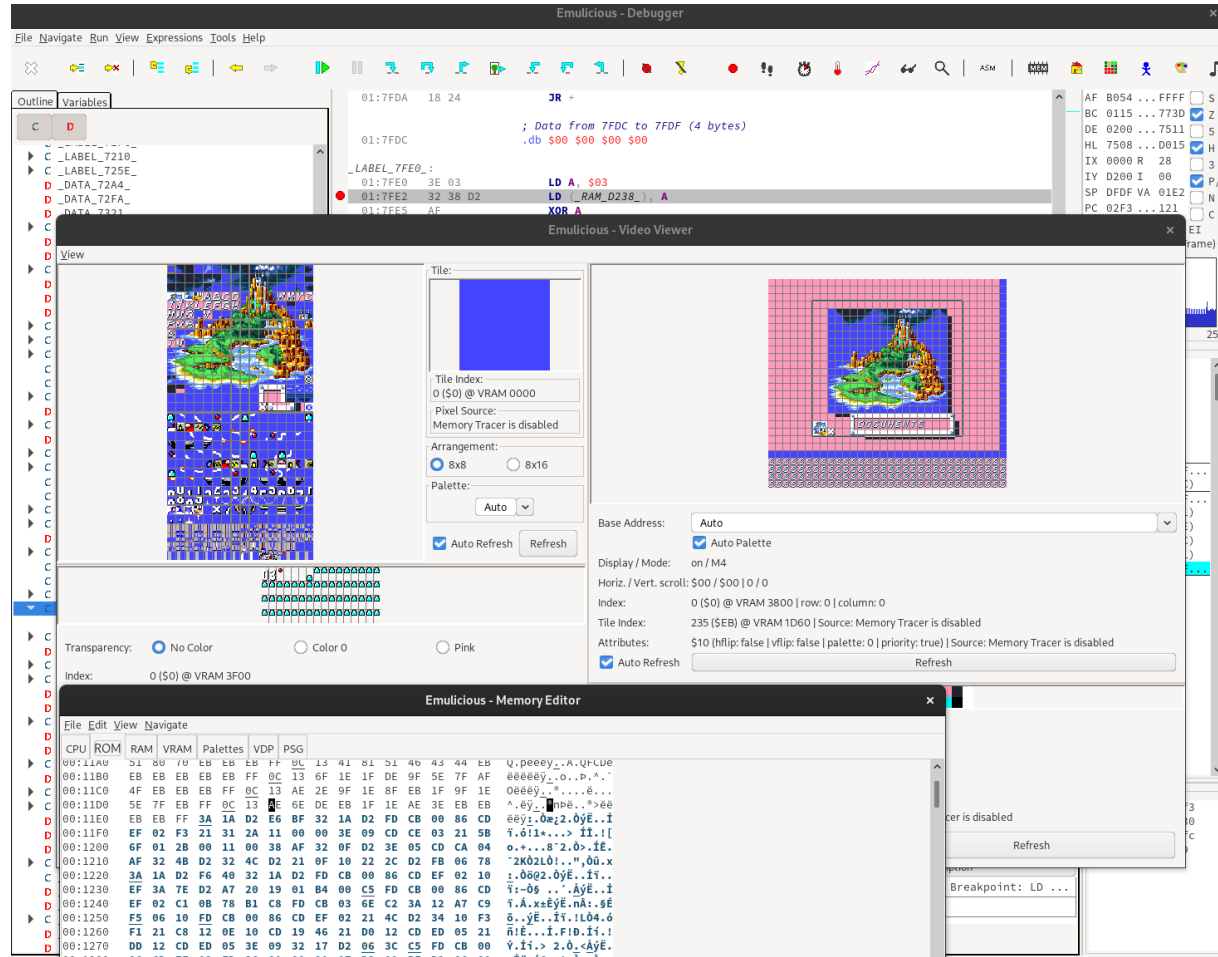


# Structure: tricks

- Devices : simple code boundary
- CPU : you are allowed to optimize a bit in advance (no allocation, jump tables)
- Vertical slice : common advice
  - Do what works for you

# Structure: tricks

- You will need a debugger
  - Build tooling early
  - Or use already existing

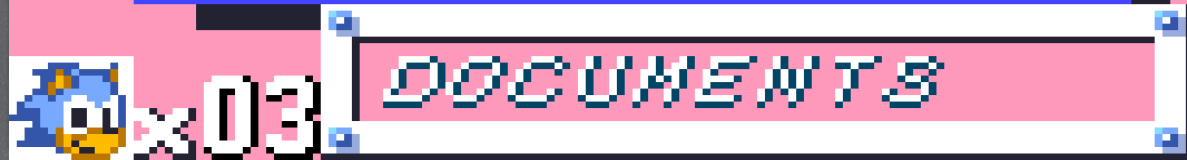




GAME  
GEAR



POWER



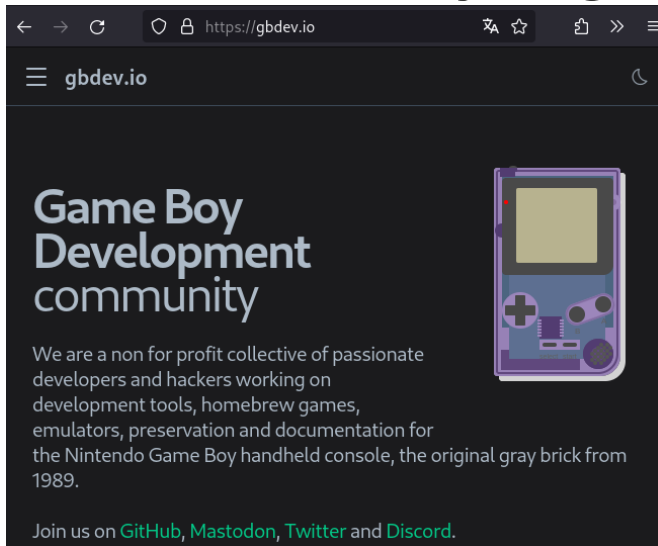
SEGA

PORTABLE  
VIDEO GAME  
SYSTEM



# Finding documentation

- Game boy : gbdev.io, Gekkio's CTR



## Game Boy: Complete Technical Reference

gekkio

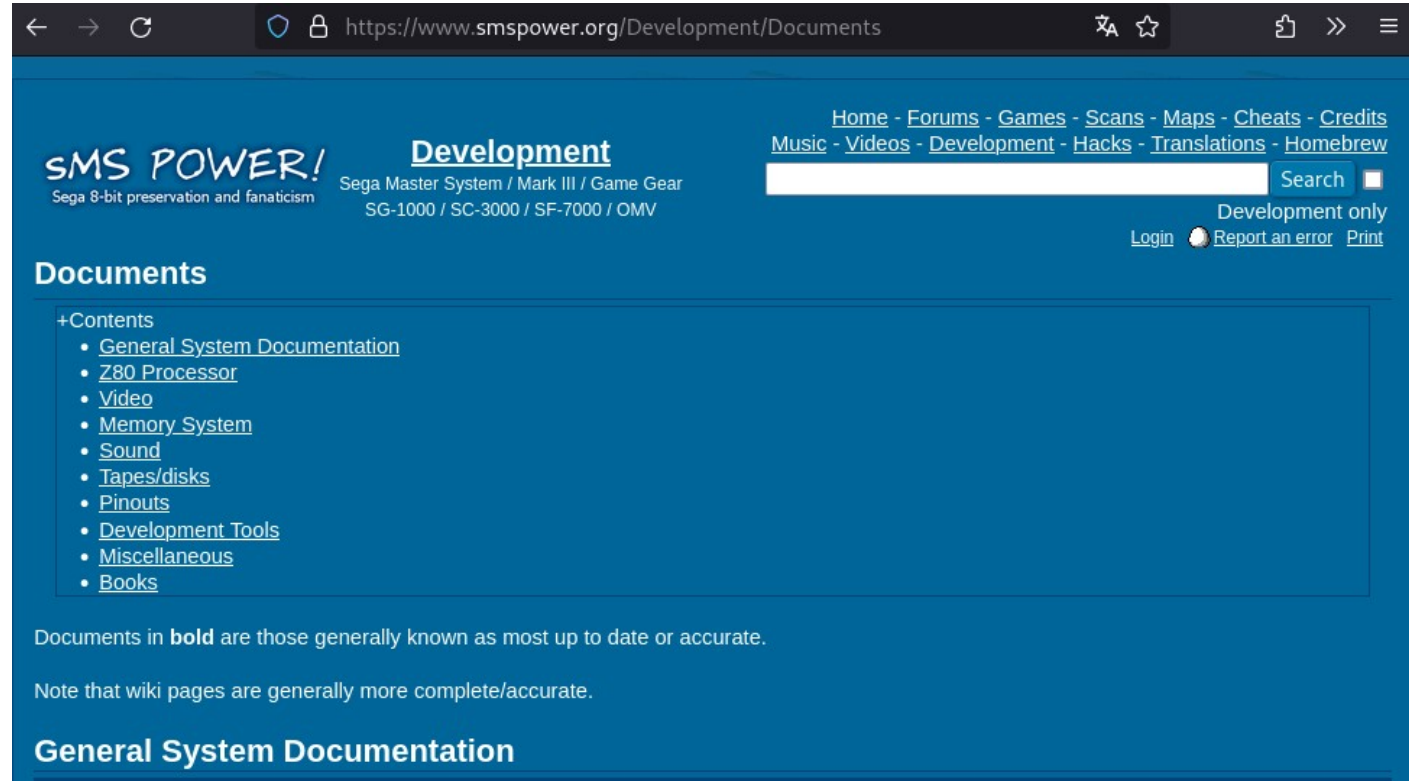
<https://gekkio.fi>

January 7, 2024

Revision 135

# Finding documentation

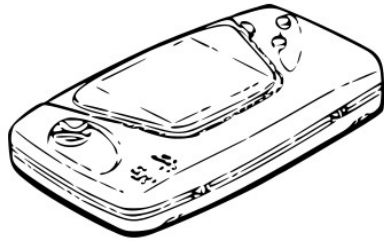
- SMS
- Game Gear



The screenshot shows a web browser window with the URL <https://www.smspower.org/Development/Documents>. The page has a blue header with the site logo "SMS POWER! Sega 8-bit preservation and fanaticism" and a "Development" section listing "Sega Master System / Mark III / Game Gear" and "SG-1000 / SC-3000 / SF-7000 / OMV". A search bar and navigation links like "Home", "Forums", "Games", "Scans", "Maps", "Cheats", "Credits", "Music", "Videos", "Development", "Hacks", "Translations", and "Homebrew" are visible. The main content area is titled "Documents" and contains a list of links under "+Contents": [General System Documentation](#), [Z80 Processor](#), [Video](#), [Memory System](#), [Sound](#), [Tapes/disks](#), [Pinouts](#), [Development Tools](#), [Miscellaneous](#), and [Books](#). Below the list, a note states: "Documents in **bold** are those generally known as most up to date or accurate. Note that wiki pages are generally more complete/accurate." The page footer shows the title "General System Documentation".

# Finding documentation

Hardware Reference Manual  
for the  
SEGA Game Gear Console



**zilog**  
Embedded in Life  
An IXYS Company

**Z80 Microprocessors**

**Z80 CPU**

**User Manual**

UM008011-0816

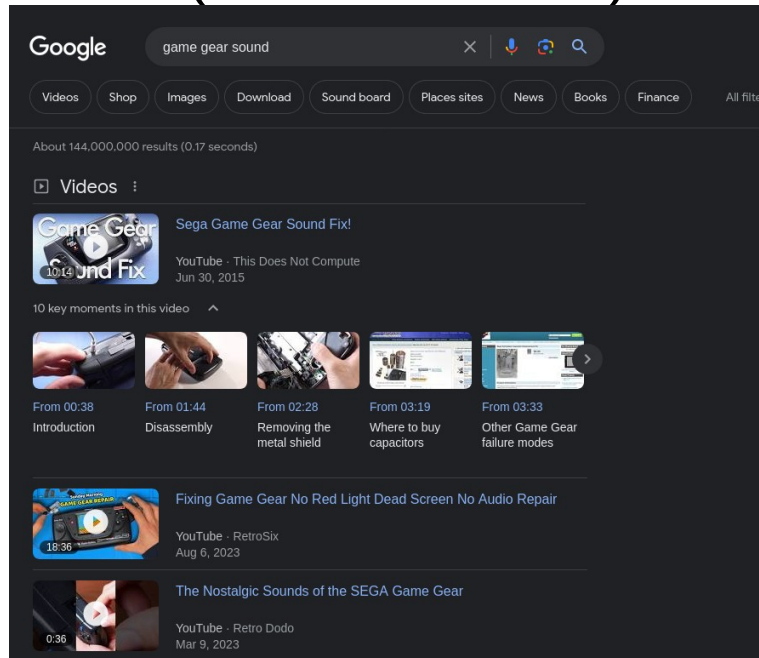
The Undocumented Z80 Documented

Sean Young

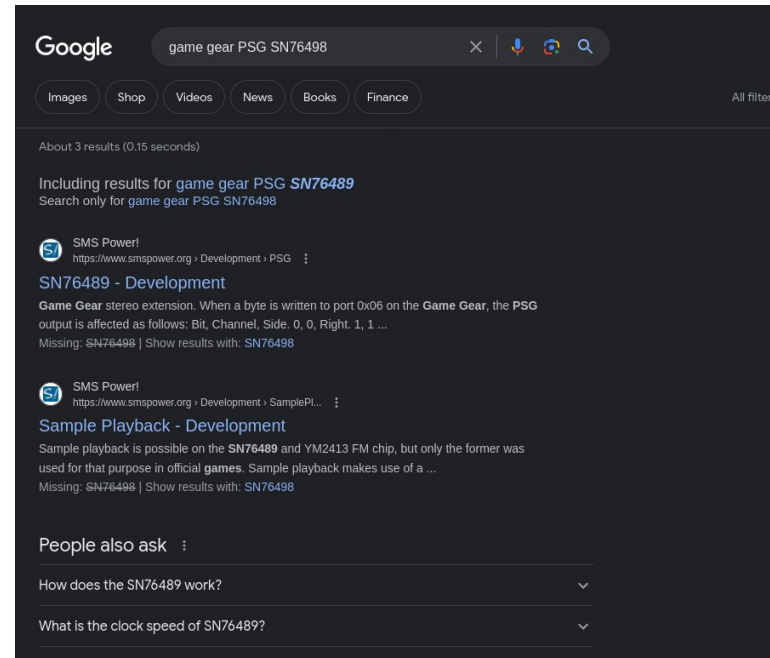
Version 0.91, 18th September, 2005

# Finding documentation

- Do web searches with technical terms. Ex : search for <sound chip name> (SN76498/PSG) instead of <console sound>



FOSDEM 2024



Anisse Astier

How do you write an emulator anyway ?

GAME  
GEAR

POWER



x04 PRACTICAL

SEGA  
PORTABLE  
VIDEO GAME  
SYSTEM

# Device I/O

- Used to be almost as simple as writing to a memory address
- Dedicated instructions

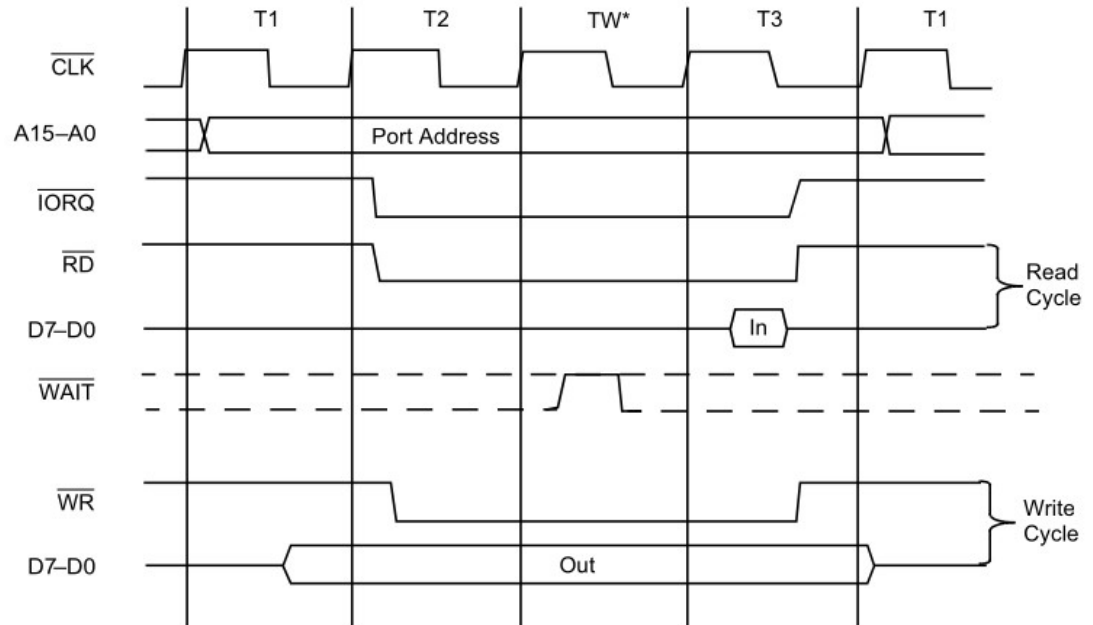


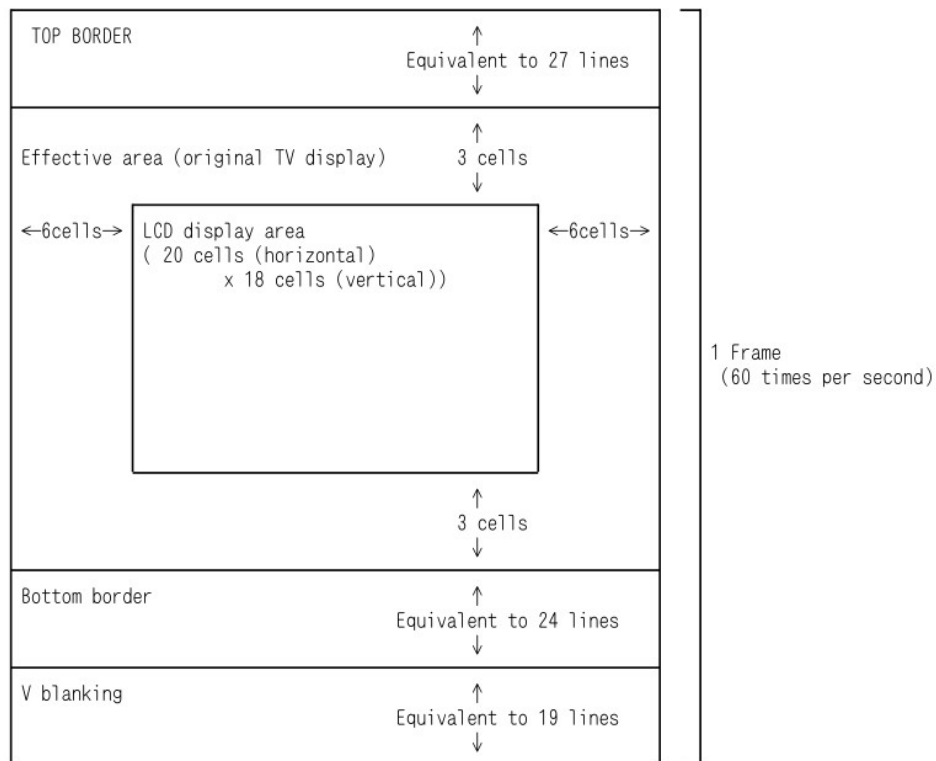
Figure 7. Input or Output Cycles

# Displaying something and making sound

- Understand your host platform first
- Hello world of video
  - Display a pixel buffer, change it 60 times a second
- Hello world of sound
  - Play a sine or square wave
- Nothing emulator specific

# Graphics emulation

- Needs hardware understanding
- VDP example
  - VRAM/Regs I/O
  - Two planes : BG and sprites
  - BG is on torus, scrollable





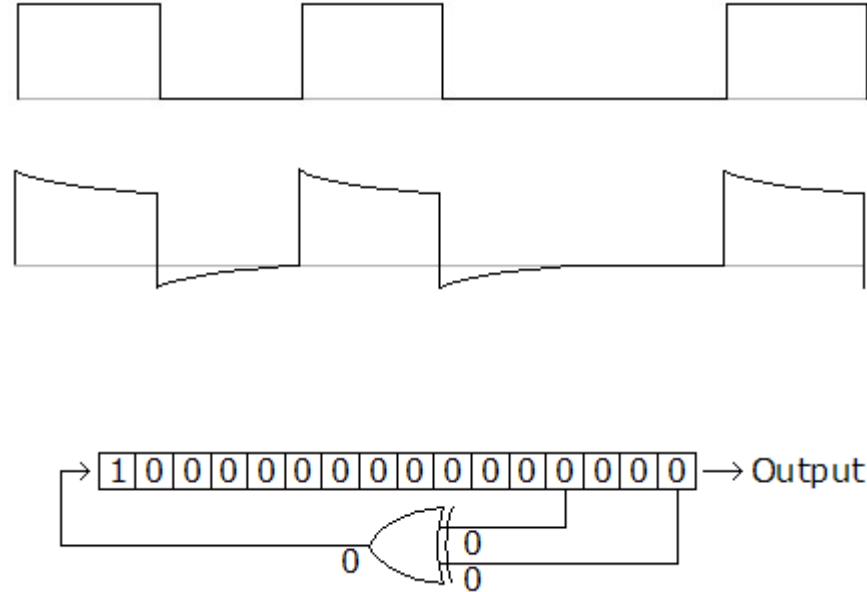
# Graphics emulation

- VDP example (cont)
  - Sprites are limited
  - Color encoding
  - Sprite pixel encoding
- Sync strategy: line



# Sound emulation

- Needs hardware understanding
- PSG example
  - Regs I/O
  - Channels
  - Tones
    - Counter
  - Noise
    - Random/LFSR
    - Periodic



# Sound emulation

- Start simple
  - Square wave
- Example : counter, not frequency



- Example : playing samples
- Sync strategy: cycles

GAME  
GEAR

POWER



 x05 TESTING

SEGA  
PORTABLE  
VIDEO GAME  
SYSTEM

# Emulator testing: CPU

- Unit test components using existing test suites.
- Examples for Z80
  - Zexall
  - Z80test
  - Fuse test suite



```
SpecEmu
File Z80 View Recording Monitor Options Tools Help
132 LD A, ( [BC, DE] ) OK
133 LD ( [BC, DE] ), A OK
134 LD A, (NN) OK
135 LD (NN), A OK
136 LD RR, NN OK
137 LD XY, NN OK
138 LD HL, (NN) OK
139 LD XY, (NN) OK
140 LD RR, (NN) OK
141 LD (NN), HL OK
142 LD (NN), XY OK
143 LD (NN), RR OK
144 LD SP, HL OK
145 LD SP, XY OK
146 LD I, A OK
147 LD R, A OK
148 LD A, I OK
149 LD A, R OK
150 EI+DI OK
151 IM N OK

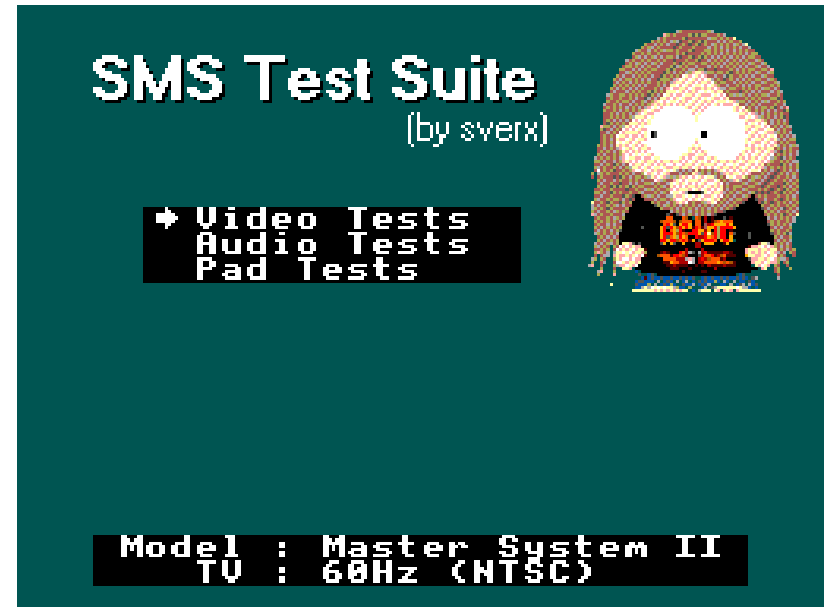
Result: all tests passed.
0 OK, 20:1
```

# Emulator testing: audio

- Sound:
  - Listen to music, needs a good ear
  - FFTs
  - Can you hear samples (SEGA sound)

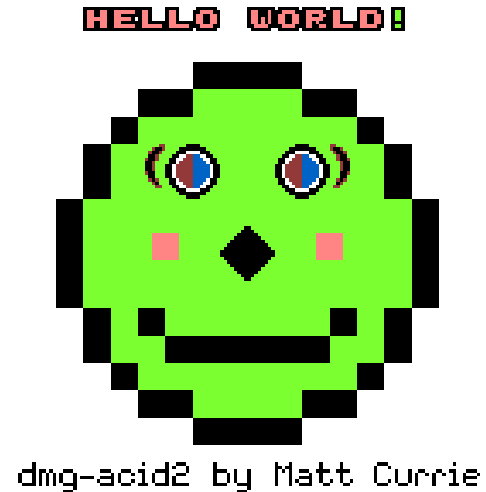
# Emulator testing: roms and display

- GG test suite, SMS test suite, inspired by 240p



# Emulator testing: roms and display

- Game boy examples:
  - dmg-acid2 ppu test
  - Blargg's gb tests
  - Mooneye test suite

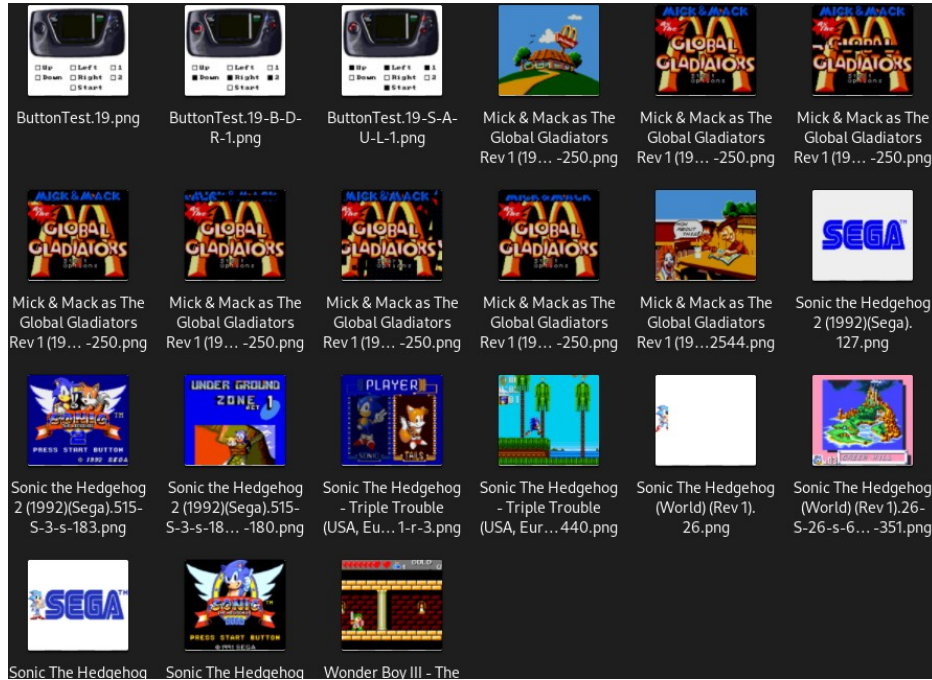




# Emulator testing: frame generation

- Integration testing: automate frame generation
  - Everything is deterministic
- Compare with other (good) emulators
- Use real hardware
  - Tip: use flash carts
- Try lots of software/games/roms

# Emulator testing: frame generation





GAME  
GEAR



POWER



 x06 SUMMARY

SEGA

PORTABLE  
VIDEO GAME  
SYSTEM

# Summary

- Pick platforms
- Start small
- Read lots of documentation
- Test
- Write and talk about it

# Thank you ! Questions ?



# Links

- My emulator, gears: <https://github.com/anisse/gears>
- gears in the browser: <https://anisse.github.io/gears>
- Romhack to generate level screens:  
<https://gist.github.com/anisse/c6e4101236708890381414f48804201b>
- Linux on scratch: <https://turbowarp.org/892602496/>
- <https://github.com/gbdev/awesome-gbdev#emulator-development>
- SMS test suite <https://github.com/sverx/SMSTestSuite>
- GG test suite <https://github.com/sverx/GGTestSuite/>
- Game boy test roms compilation: <https://github.com/c-sp/gameboy-test-roms>

# Sources

- Game gear by Evan-Amos - Own work, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=13317134>
- Screenshots from Sameboy <https://sameboy.github.io/>
- BGB: <https://bgb.bircd.org/shots.html>
- dmg-acid2 image: <https://github.com/mattcurrie/dmg-acid2>
- Z80test result on specemu: <https://www.spectrumcomputing.co.uk/forums/viewtopic.php?t=752>