JAK NAPISAĆ WŁASNY EMULATOR

Czym jest emulator?

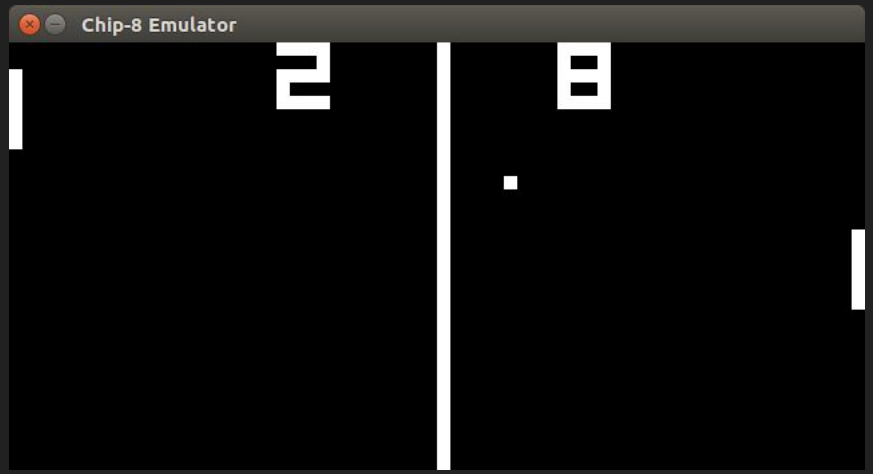

CCS64 V3.1
-File -View -Options -State -Help
**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY.
384 x 282
NUM


Atari800Win PLus 3.0: XL/XE (64 KB)
File Atari Input View Sound Misc Help
Speed: 100%


Nestopia
58
MARIO
000000
×00
WORLD
0-1
TIME
250

Jaki jest cel pisania własnego emulatora?

CHIP-8

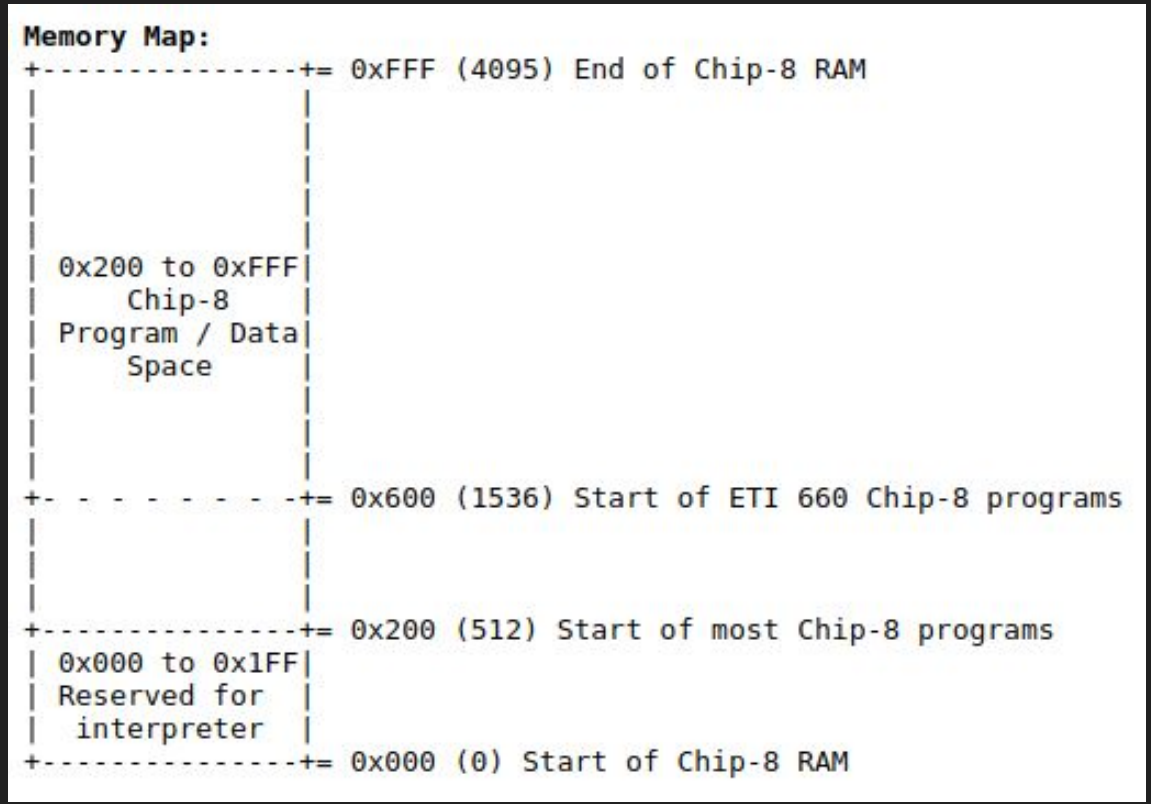


http://devernay.free.fr/hacks/chip8/C8TECH10.HTM

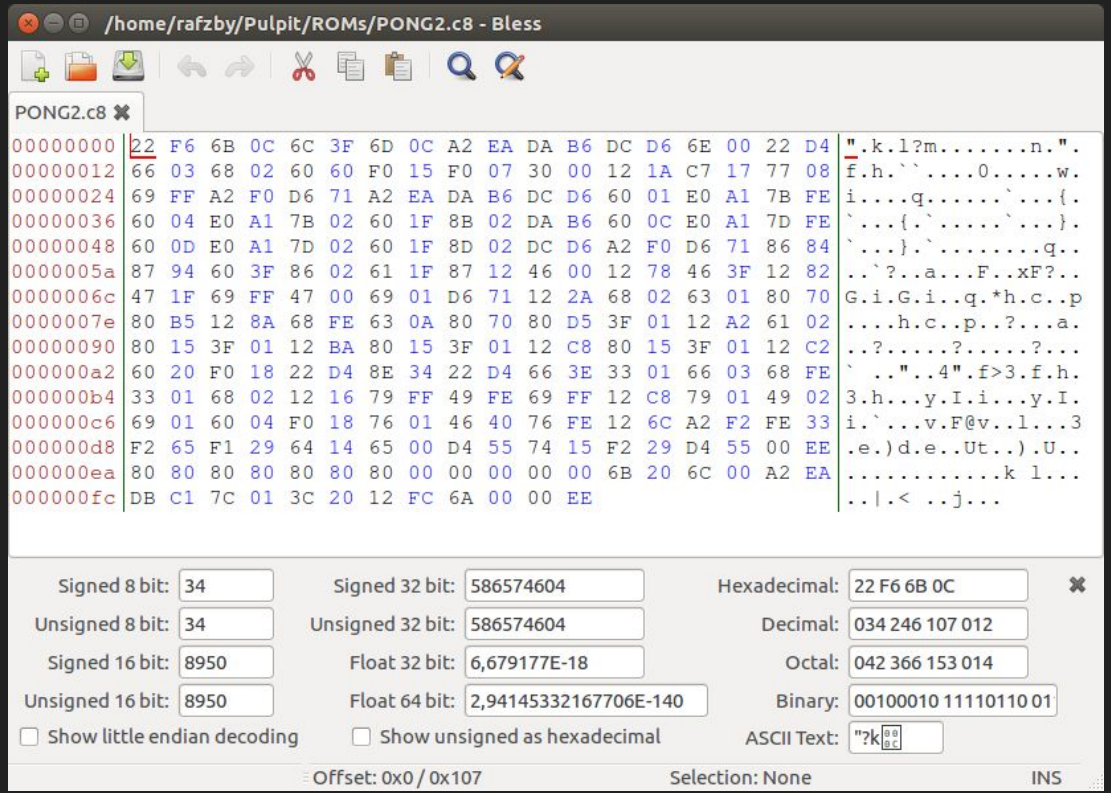
Budowa emulatora CHIP-8

1. Pamięć
2. CPU
3. Urządzenia wejścia-wyjścia

Pamięć - 4KB


Memory Map:
0xFFF (4095) End of Chip-8 RAM
0x200 to 0xFFF
Chip-8
Program / Data
Space
0x600 (1536) Start of ETI 660 Chip-8 programs
0x200 (512) Start of most Chip-8 programs
0x000 to 0x1FF
Reserved for
interpreter
0x000 (0) Start of Chip-8 RAM

Zawartość pliku z programem



```java
public class Memory {
    private char[] memory;


    public Memory(int size) {
        memory = new char[size];
    }

    public char readByte(int address) throws MemoryReadException {
        if (address < 0 || address >= memory.length) {
            throw new MemoryReadException("Attempt to read data from wrong memory address.");
        }

        return memory[address];
    }

    public char readOpcode(int address) throws MemoryReadException {
        return (char) ((readByte(address) << 8) | readByte(address + 1));
    }

    public void writeByte(int address, char value) throws MemoryWriteException {
        if (address < 0 || address >= memory.length) {
            throw new MemoryWriteException("Attempt to write date to the wrong memory address.");
        }

        memory[address] = (char) (value & 0xFF);
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();

        for (int i = 0; i < memory.length; i++) {
            try {
                builder.append(Integer.toHexString(i))
                        .append(": ")
                        .append(Integer.toHexString(readByte(i)))
                        .append("\n");
            } catch (MemoryReadException e) {
                e.printStackTrace();
            }
        }

        return builder.toString();
    }
}
```

0x - Prefix stosowany przy zapisie liczb szesnastkowych w różnych językach programowania.

```
MEMORY
---------+
0x200: 22|
0x201: F6|
0x202: 6B|
0x203: 0C|
---------+

readOpcode(200)

0x22 = 0010 0010

0010 0010 << 8
0010 0010 0000 0000

0xF6 = 1111 0110

0010 0010 0000 0000 | 1111 0110


0010 0010 0000 0000 |
          1111 0110
--------------------
0010 0010 1111 0110

opcode = 0010 0010 1111 0110 = 0x22F6
```

CPU - Central Processing Unit

- 35 instrukcji.
- 16 rejestrów ogólnego przeznaczenia (1B, 0x00 - 0xFF), jeden zarezerwowany na flagę przeniesienia (ang. carry flag) o adresie 0xF.
- Rejestr I - służący do przechowywania adresu pamięci do późniejszego użytku. Nie ma możliwości jego modyfikacji.
- Licznik instrukcji.
- Stos (16 elementów).
- Delay Timer.
- Sound Timer.

Urządzenia wejścia-wyjścia

- Wyświetlacz o rozmiarach 64x32px
- Klawiatura z 16 klawiszami

Wyświetlacz

- Zwykły JPanel
- 2 kolory pikseli - czarny i biały
- 16-elementowa czcionka (0-F)
- Dla lepszego odbioru ekran został powiększony do rozmiaru 640x320px
- Wartość każdego piksela pomnożona przez 10

Czcionka

- 16 znaków zapisanych w pamięci RAM zaczynając od adresu 0x50

"0"	Binary	Hex
****	11110000	0xF0
* *	10010000	0x90
* *	10010000	0x90
* *	10010000	0x90
****	11110000	0xF0

"1"	Binary	Hex
*	00100000	0x20
**	01100000	0x60
*	00100000	0x20
*	00100000	0x20
***	01110000	0x70

"2"	Binary	Hex
****	11110000	0xF0
*	00010000	0x10
****	11110000	0xF0
*	10000000	0x80
****	11110000	0xF0

"3"	Binary	Hex
****	11110000	0xF0
*	00010000	0x10
****	11110000	0xF0
*	00010000	0x10
****	11110000	0xF0

"4"	Binary	Hex
* *	10010000	0x90
* *	10010000	0x90
****	11110000	0xF0
*	00010000	0x10
*	00010000	0x10

"5"	Binary	Hex
****	11110000	0xF0
*	10000000	0x80
****	11110000	0xF0
*	00010000	0x10
****	11110000	0xF0

"6"	Binary	Hex
****	11110000	0xF0
*	10000000	0x80
****	11110000	0xF0
* *	10010000	0x90
****	11110000	0xF0

"7"	Binary	Hex
****	11110000	0xF0
*	00010000	0x10
*	00100000	0x20
*	01000000	0x40
*	01000000	0x40

"8"	Binary	Hex
****	11110000	0xF0
* *	10010000	0x90
****	11110000	0xF0
* *	10010000	0x90
****	11110000	0xF0

"9"	Binary	Hex
****	11110000	0xF0
* *	10010000	0x90
****	11110000	0xF0
*	00010000	0x10
****	11110000	0xF0

"A"	Binary	Hex
****	11110000	0xF0
* *	10010000	0x90
****	11110000	0xF0
* *	10010000	0x90
* *	10010000	0x90

"B"	Binary	Hex
***	11100000	0xE0
* *	10010000	0x90
***	11100000	0xE0
* *	10010000	0x90
***	11100000	0xE0

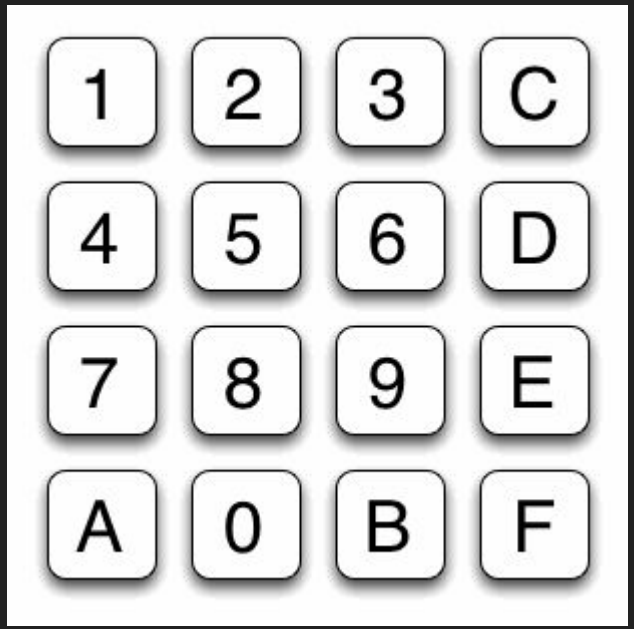
"C"	Binary	Hex
****	11110000	0xF0
*	10000000	0x80
*	10000000	0x80
*	10000000	0x80
****	11110000	0xF0

"D"	Binary	Hex
***	11100000	0xE0
* *	10010000	0x90
* *	10010000	0x90
* *	10010000	0x90
***	11100000	0xE0

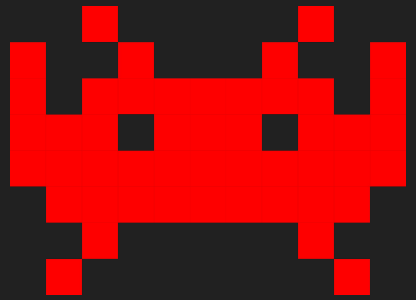
"E"	Binary	Hex
****	11110000	0xF0
*	10000000	0x80
****	11110000	0xF0
*	10000000	0x80
****	11110000	0xF0

"F"	Binary	Hex
****	11110000	0xF0
*	10000000	0x80
****	11110000	0xF0
*	10000000	0x80
*	10000000	0x80

Klawiatura



DEMO



Zasoby

http://devernay.free.fr/hacks/chip8/C8TECH10.HTM

http://mattmik.com/files/chip8/mastering/chip8.html

https://chris.beams.io/posts/git-commit/

https://en.wikipedia.org/wiki/CHIP-8

https://github.com/rafzby/chip8-emulator

https://helion.pl/ksiazki/zrozumiec-programowanie-gynvael-coldwind,e_1ott.htm

Kontakt

rafal.zbytniewski@gmail.com

https://www.linkedin.com/in/rafzby/

https://github.com/rafzby/

Q&A

Thanks!