Chip8 Emulator in C

Kirjan Kohuladas/kprime21

Overview

Compact Hexadecimal Interpretive Programming – 8-bit

Components

- Memory - 4kB ram (4096 bytes - 4096 address lines each line is 1 byte)
- Display - 64 x 32 pixels
- Registers
  - Program Counter (16 bits)
  - Index Register (16 bits)
  - Stack - call subroutines and functions (16 bits)
  - Delay timer - decremented at 60Hz (8 bits)
  - Sound timer - decremented at 60Hz (8 bits)
  - 15 General purpose registers - V0 - VF (8 bits)

Memory

- all memory is RAM, 4096 bytes.
  - 4096 addressable lines
  - 12 bits needed
  - each addressable line represents an addresss of 1 byte.
- interpreter located 0x000-0x1FF (not in our case)
- program located 0x200 - 0x...
- font located before program 0x000-0x1FF (popular area - 0x050 - 0x09F)

Font

- font character should be 4px x 5px
- first byte is the character (draw vertically in nvim to see)
- stored in memory, index register set to specific font in memory to draw it

Display

- 60Hz - 60 times per second
- sprite consists of 8 bits
- sprites are between 1 and 15 bytes tall
- 0 bits are transparent and 1 bits flip pixel locations

Stack

- stack(LiFo) to call and return from subroutines
- 16 bit addresses (12 bits useful) are saved here

Timers

- two timer registers - the delay timer and sound timer
- one byte in size and if above 0, decremented by 1 60 times per second (60Hz)
- sound timer beeps as long as it's above 0

Keypad

- 123C
- 456D
- 789E
- A0BF

Fetch/decode/excute loop

- fetch the instruction from memory at current PC (program counter)
- decode the instruction to find what emulator should do
- execute instruction and do what it tells you
- this loop's speed has to be set so that it does not run too fast (700Hz)
- fetch: read instruction in PC, two successive bytes and combine into one 16 bit instruction, increment PC by 2
- decode: switch statement checking first half of instruction
- nibbles after first used for decoding, extract these before decoding from opcode
  - X - second nibble
  - Y - third nibble
  - N - fourth nibble
  - NN - third and fourth nibble - 8 bit immd number
  - NNN - second, third and fourth nibble - 12 bit immd address
- execute: do what each instruction should do in each case of the switch

Opcodes

1. 0nnn - ignored by modern interpreters.
2. 00E0 - CLS, clear the display.
3. 00EE - RET, return from subroutine.
4. 1nnn - JP addr, program counter set to nnn.
5. 2nnn - CALL addr, increment stack pointer then place current PC on top of stack, then set PC to nnn.
6. 3xkk - SE Vx byte, skip next instruction if Vx == kk, PC+=2.
7. 4xkk - SNE Vx byte, skip next instruction if Vx != kk, PC+=2.
8. 5xy0 - SE Vx, Vy, skip next instruction if Vx == Vy, PC+=2.
9. 6xkk - LD Vx, byte - load kk into Vx.
10. 7xkk - ADD Vx, byte - add value kk to register Vx, store result in Vx.
11. 8xy0 - LD Vx, Vy - store value of register Vy in register Vx.
12. 8xy1 - OR Vx, Vy - bitwise OR then store in Vx.
13. 8xy2 - AND Vx, Vy - bitwise AND then store in Vx.
14. 8xy3 - XOR Vx, Vy - bitwise XOR and then store in Vx.
15. 8xy4 - ADD Vx, Vy - add Vx, Vy, result greater than 255 sets VF flag, only lowest 8 bits stored in Vx.
16. 8xy5 - SUB Vx, Vy - Vx ¿ Vy, Vf set to 1, otherwise 0. Subtract Vy from Vx and store in Vx.
17. 8xy6 - SHR Vx (, Vy) - LSB of Vx is 1, set Vf to 1. Otherwise 0. Vx divided by 2.
18. 8xy7 - SUBN Vx, Vy - Vy ¿ Vx, Vf set to 1, otherwise 0. Subtract Vx from Vy and store in Vx.
19. 8xyE - SHL Vx (, Vy) - MSB of Vx is 1, set Vf to 1. Otherwise 0. Vx multiplied by 2.
20. 9xy0 - SNE Vx, Vy - Vx and Vy compared, if not equal, PC+=2.
21. Annn - LD I, addr - Index register is set to addr.
22. Bnnn - JP V0, addr - Program Counter is set to nnn+V0.
23. Cxkk - RND Vx, byte - Generate random number AND with kk, store in Vx.

24. Dxyn - DRW Vx, Vy, nibble - Read n bytes from memory staring at address stored in Index. Bytes are displayed as sprites on screen at coord(Vx,Vy). Sprites are XOR'd onto screen, any overlapping part of sprite will be wrapped around screen. If sprite collides with another sprite then set Vf to 1.
25. Ex9E - SKP Vx - Skip next instruction if key with value Vx is pressed, PC+=2.
26. ExA1 - SKNP Vx - Skip next instruction if key with value Vx is not pressed, PC+=2.
27. Fx07 - LD Vx, DT - Set the value of Vx to value of delay timer.
28. Fx0A - LD Vx, k - Execution stops until key is pressed, store value of key in Vx.
29. Fx15 - LD DT, Vx - Set the value of delay timer to value of Vx.
30. Fx18 - LD ST, Vx - Set the value of sound timer to value of Vx.
31. Fx1E - ADD I, Vx - Value of Index register and Vx added and stored in Index register.
32. Fx29 - LD F, Vx - Index register is set equal to the location for hexadecimal sprite corresponding to value in Vx.
33. Fx33 - LD B, Vx - Store BCD(Vx) in Index register, Index+1, Index+2.
34. Fx55 - LD [I], Vx - Store registers V0 through Vx in memory starting at location in Index register.
35. Fx65 - LD Vx, [I] - Read values from memory and store them from V0 through Vx.

File Structure

- cpu.c - Contains the registers for the Chip8 emulator, the memory that the program is read into and the font that is read at the start
- graphics.c - Contains the display graphics for the Chip8 emulator using SDL2
- main.c - Driver program that will read in the rom and start the CPU and graphics
- chip8d.c - Dissassembler to dissassemble the roms, use hexeditor for clarity

Main

- Work on keyboard input
- Work on graphics class
- work on CPU timer, graphics timer, delay and sound timer
- work on main method class

Reference

- https://tobiasvl.github.io/blog/write-a-chip-8-emulator/
- http://devernay.free.fr/hacks/chip8/C8TECH10.HTM