CHIP8

CHIP8 is an interpreter (emulator) for CHIP-8 . I wrote it after a friend told me he was going to write a CHIP-8 interpreter (which I hadn't heard of before), and it looked interesting. It's been a while since I've had so much fun writing code.

CHIP8 is written in C with inline assembly, probably in the most useless way possible. What makes it useless is that it targets a DOS machine to run on -- as an old assembly language programmer, I have a LOT of experience with programming down to the bare metal of a DOS machine and lots of code to draw from.

As a result, on modern machines it needs to be run in DOSBOX (www.dosbox.com). This makes it an interpreter running in an interpreter.

SPEED

Nevertheless, it's quite fast -- on an i7-4770 with DOSBOX running at full speed, non-graphics instructions run at about 2.8 MIPS (2.8 million instructions per second). Graphics instructions are slower, but the worst case of constantly drawing a 16 x 16 sprite in Super CHIP-8 graphics (128 x 64) (which updates video memory then re-writes the entire graphics screen) runs at about 1060 instructions per second.

A "mixed instruction set" program like BLINKY.CH8, at full speed, runs at about 37,000 instructions per second.

Speeds like that make most CHIP-8 programs unplayable, so part of the design for my CHIP8 interpreter was an instruction execution clock. I used a default of 2040 instructions per second (2040 Hz) for a couple of reasons:
1) About 2000 Hz sounded like a good target.
2) The DOS environment only has 1 timer available, and I needed to have 3 time bases derived from it:
    a) 2000 Hz execution clock
    b) 60Hz time base for delay timer and sound timer
    c) 18.2Hz

Doing the math, 2040 / 60 = 34 (update the 60 Hz timer every 34th execution tick), and 2040 / 18.2 = 112 (update the system timer every 112th execution tick).

The program allows the user to change both execution ticks per instruction cycle and number of instructions executed every execution cycle. So, for example, changing the execution ticks per instruction cycle to 4 gives a rate of 510 instructions per second. And changing the execution ticks per instruction cycle to 8 and number of instructions executed to 3 would give a rate of 765 instructions per second.

Instruction Set Problems

There are inconsistencies in the documentation for CHIP-8 instructions.

The Cowgod CHIP-8 technical reference (http://devernay.free.fr/hacks/chip8/C8TECH10.HTM) says that the instructions
  SHL Vx,Vy
and
  SHR Vx,Vy
shift the Vx register, Vy is not used. It also does not say that the I register is updated in the LD [I], Vx and LD Vx, [I] instructions.

Another source, Matthew Mikolay (http://mattmik.com/files/chip8/mastering/chip8.html) says that the instructions
  SHL Vx, Vy
and
  SHR Vx,Vy
shift the Vy register and store the result in the Vx register. It also states that the LD [I], Vx and LD Vx, [I] instructions do update the I register.

Another source, Wikipedia (https://en.wikipedia.org/wiki/CHIP-8) notes that the ADD I, Vx instruction sets CARRY in VF, but this is undocumented.

The problem is that **ALL** the various conflicting definitions are "correct" -- there are programs that rely on a particular implementation to run correctly.

For example, the program BLINKY.CH8 (https://github.com/stianeklund/chip8/blob/master/roms/BLINKY.ch8) relies on the Cowgod definition of SHL and SHR, using Mikolay's definition breaks the program.

The program EATY.CH8 (https://johnearnest.github.io/chip8Archive/play.html?p=eaty) rely on Mikolay's definition of SHL and SHR, using Cowgod's definitions breaks the program.

The program Spacefight 2091! (https://github.com/michaelarnuats/chip8-java/tree/master/Binary/roms) relies on the CARRY out of the ADD I, Vx instruction.

CHIP8 has both behaviors for each instruction type programmed, controlled by flags that can be set via the command line. The default is the Cowgod definitions and no CARRY from ADD I, Vx (I used BLINKY.CH8 as my standard test while developing CHIP8).

COMMAND LINE

```
Command line parameters (numbers are DECIMAL):
   -F  : execution Freerun (run code as fast as possible)
   -Tn : Execution ticks (2040 ticks/second) to wait before executing CHIP8 instructions.  Default is 1.
   -In : CHIP8 instructions to execute every time execution tick timer expires.  Default is 1.
   -K  : Use Alternate keypad (1..4/q..r/a..f/z..v) instead of PC keypad.
   -V  : Default High-res video (128 x 64) instead of standard video(64 x 32)
   -Sn : Sound frequency.  Default is 330Hz.  Range is 100 to 1000.
   -Ln : Load address for program.  Valid values are 512 and 1536.  Default value is 512.
   -W  : Wrap sprites that go beyond the right edge of the screen.  Default is to truncate sprites at the right edge of the
         screen.
   -M  : Use Mikolay's instruction behaviors (equivalent to using both  -X and -Y options).
   -X  : SHR Vx,Vy and SHL Vx,Vy -- shift Vy register and store it in Vx
         Default is that Vx is shifted.
   -Y  : LD [I],Vx and LD Vx,[I] instructions DO update the I register.
         Default is that they DO NOT update the I register.
   -Z  : ADD I,Vx DOES set carry on overflow
         Default is that carry is NOT set on overflow.
   program : The CHIP-8 program to load and run.  This is required.
```

EXITING CHIP8

The Esc key will exit CHIP8. The number of instructions executed and run time are displayed, just in case you're interested.