From Zero to Emulator: CHIP-8 and the Art of C# Abstraction

David Guida

Feedback form

www.davidguida.net

Who am I? Why am I here?

David Guida

Senior Software Engineer @ Microsoft

www.davidguida.net

github.com/mizrael

emulation vs simulation

Simulation:
Recreating system *outcomes* without replicating internal hardware behavior.

Emulation:
Reproducing the behavior of one system using a different system

Why Developers Build Emulators

- Learning low-level architecture
- Reverse engineering & preservation
- Performance experimentation
- Building cross-platform solutions

Core Concepts Behind Emulation

- CPU INSTRUCTIONS / OPCODE DECODING
- MEMORY MODEL
- REGISTERS AND TIMERS
- INPUT HANDLING
- GRAPHICS OUTPUT
- CLOCK CYCLES

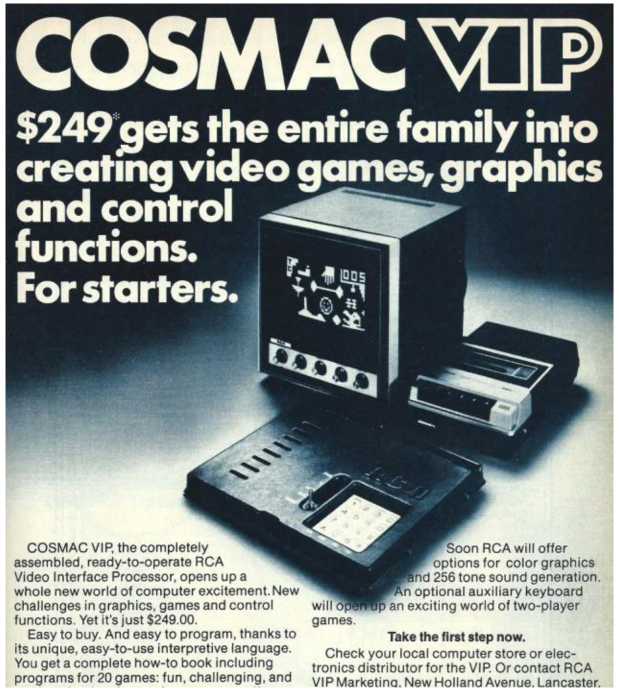
Why emulate a CHIP-8 ?

Simple architecture

Great for beginners

Few instructions

Fun programs & classic games



COSMAC VIP

$249* gets the entire family into creating video games, graphics and control functions. For starters.

COSMAC VIP, the completely assembled, ready-to-operate RCA Video Interface Processor, opens up a whole new world of computer excitement. New challenges in graphics, games and control functions. Yet it's just $249.00.

Easy to buy. And easy to program, thanks to its unique, easy-to-use interpretive language. You get a complete how-to book including programs for 20 games: fun, challenging, and

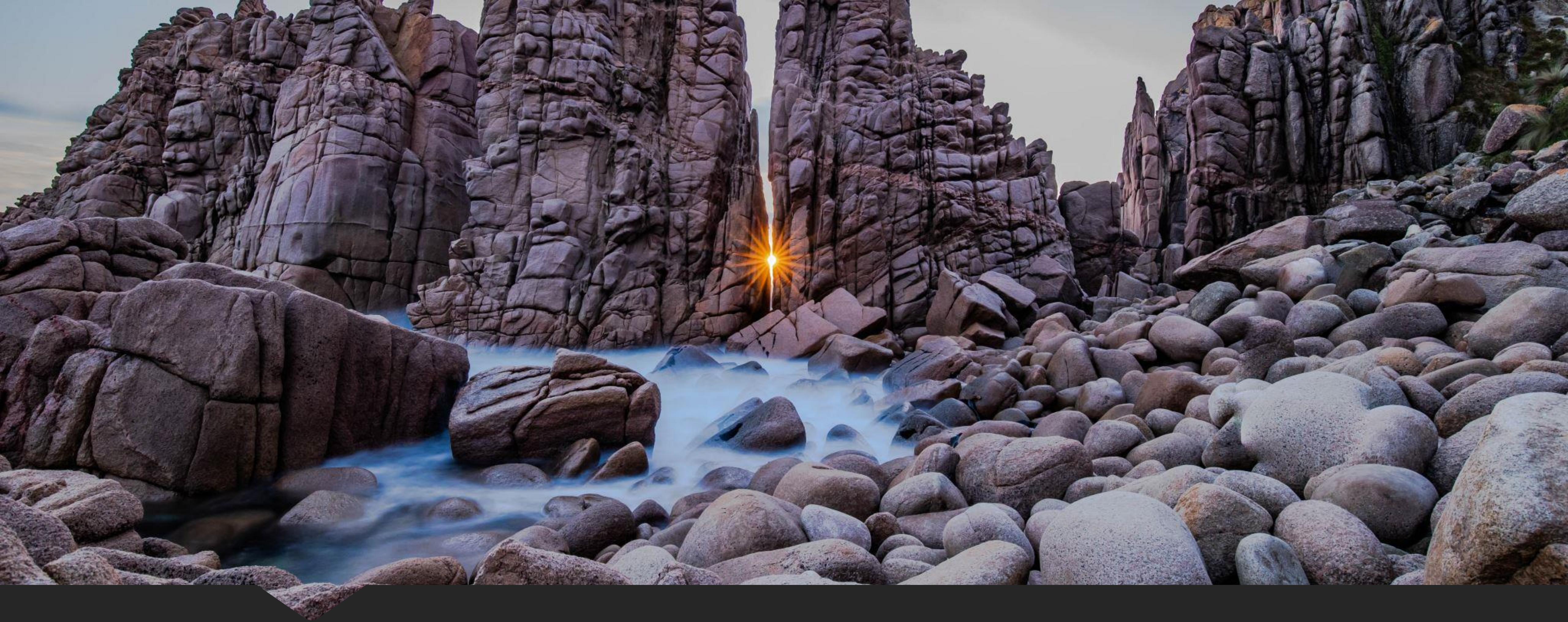
Soon RCA will offer options for color graphics and 256 tone sound generation.

An optional auxiliary keyboard will open up an exciting world of two-player games.

Take the first step now.

Check your local computer store or electronics distributor for the VIP. Or contact RCA VIP Marketing, New Holland Avenue, Lancaster,

A bit of history



Hic sunt dracones

CHIP-8 Architecture Overview: Memory Layout

- 4KB in total, 0x000 to 0xFFF
- 0x000–0x1FF: Interpreter / Fonts
- 0x200+: Program space (ROMs)

CHIP-8 Architecture Overview: the Registers

- 16 8-bit registers (V0–VF)
- one 12-bit Index register (I)
- Program counter (PC)
- Stack & stack pointer

CHIP-8 Architecture Overview: the display

Resolution: 64 × 32 monochrome pixels

Pixel state: on/off (no color, no grayscale)

Drawing method: XOR-based sprite rendering

CHIP-8 Architecture Overview: timers and input

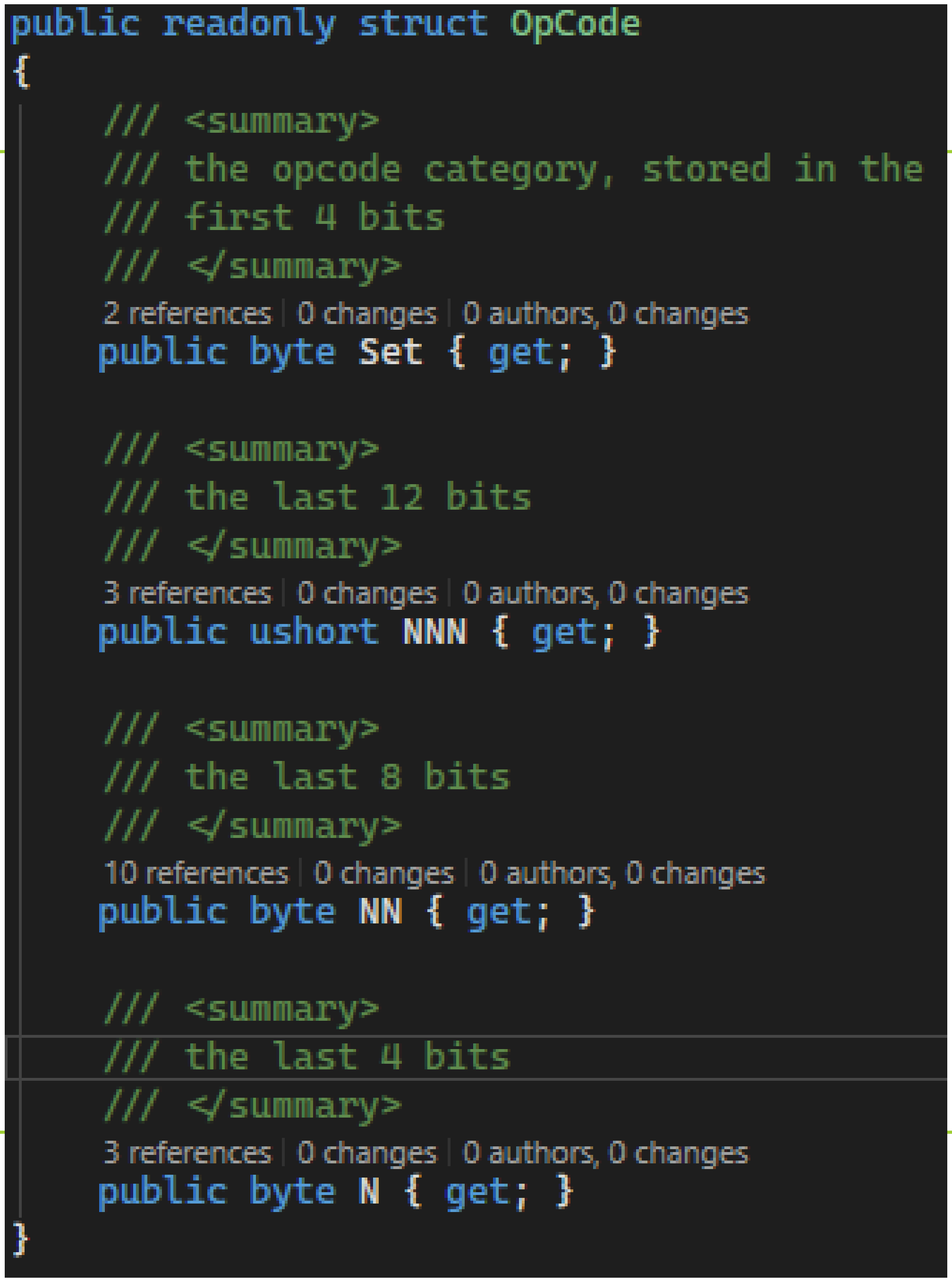
Timers: delay & sound

Hex keypad (16 keys)

The Instruction Set

- 2 bytes per opcode, big-endian
- Divided into categories: display, math, control, memory

The Opcode structure



```
public readonly struct OpCode
{
    /// <summary>
    /// the opcode category, stored in the
    /// first 4 bits
    /// </summary>
    2 references | 0 changes | 0 authors, 0 changes
    public byte Set { get; }

    /// <summary>
    /// the last 12 bits
    /// </summary>
    3 references | 0 changes | 0 authors, 0 changes
    public ushort NNN { get; }

    /// <summary>
    /// the last 8 bits
    /// </summary>
    10 references | 0 changes | 0 authors, 0 changes
    public byte NN { get; }

    /// <summary>
    /// the last 4 bits
    /// </summary>
    3 references | 0 changes | 0 authors, 0 changes
    public byte N { get; }
}
```

The "Clear the Screen" instruction

- Binary: **0000 0000 1110 0000** Hex: 0x00E0
- Clears the entire display by setting all pixels to 0.
- How the emulator handles it:
  - Fetches the opcode **0x00E0**, detects the set is **0x0**
  - Decodes NN = 0xE0 → this matches the "clear screen" pattern
  - Sets all values in the display buffer to 0

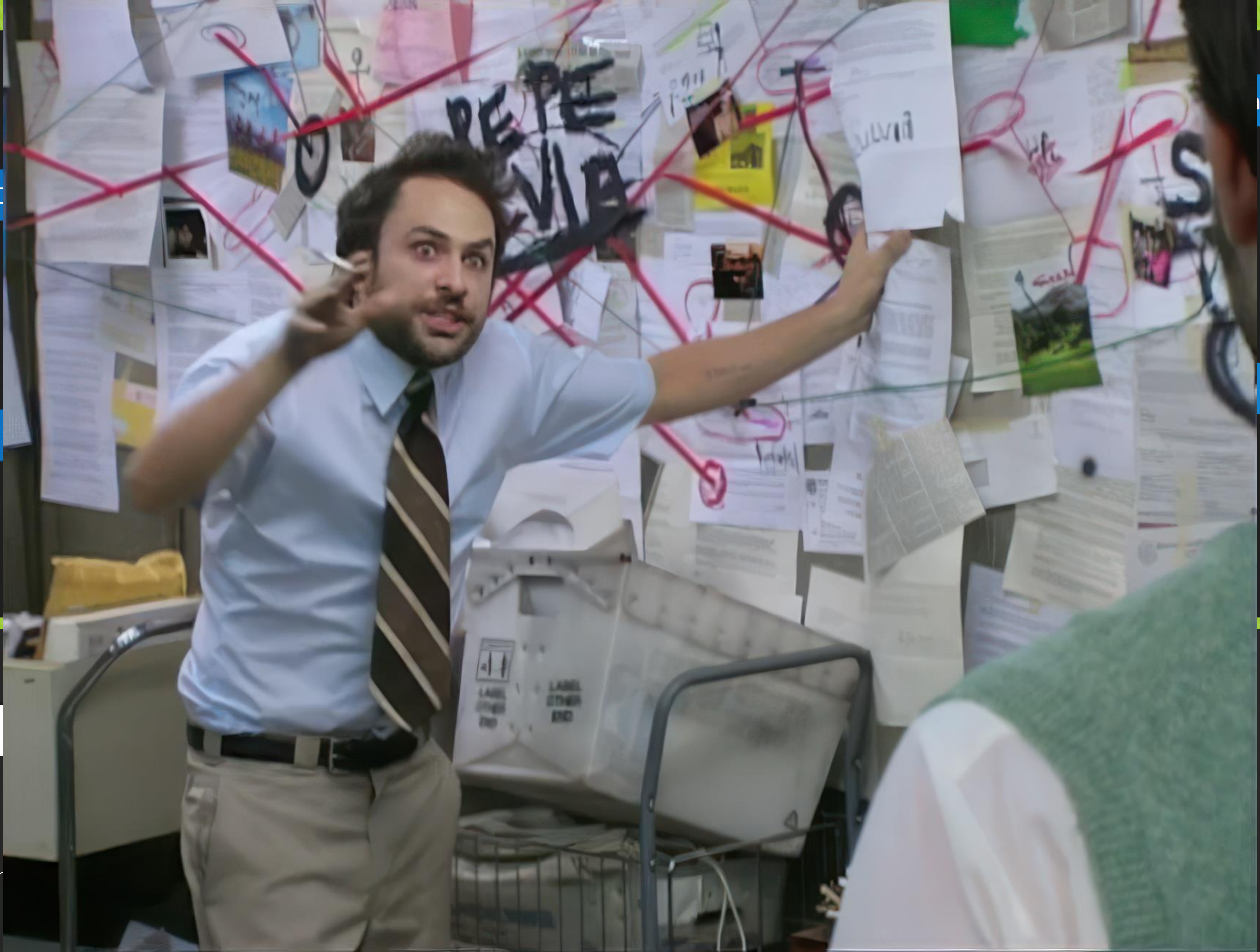
The “Call subroutine” instruction

- Binary representation: **0010 NNNN NNNN NNNN**
- Jumps to a subroutine located at address NNN.
- How the emulator handles it:
  - Fetches the opcode **0x210A**, detects the set is **0x2**
  - Decodes NNN = 0x10A
  - Pushes PC onto the stack
  - Calls subroutine at memory address 0x10A.

ıyer

Put

www.davidguida.n



Demo Time

Q&A

Feedback?