ORACLE®

JavaOne™

ORACLE®

Let’s Write Our Own CHIP-8 Interpreter!

CON3584

David Buck
Principal Member of Technical Staff
Java Platform Group
October 3rd, 2017

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Hi There!

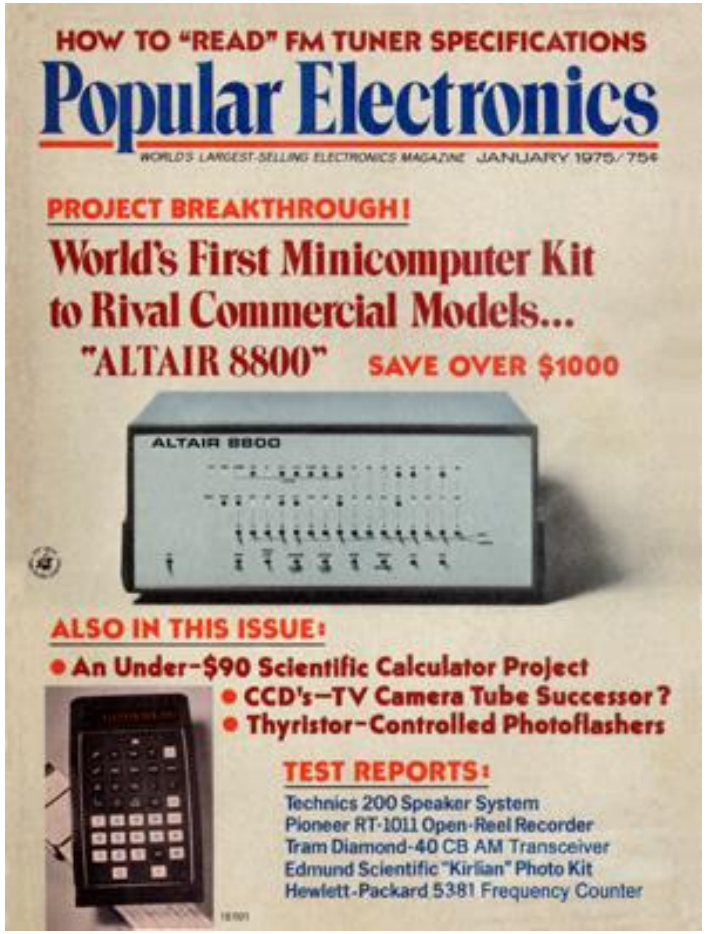
- JVM Sustaining Engineer
- OpenJDK 8 Update Project Maintainer
- JavaOne Rock Star
- Co-author of Oracle WebLogic Server 11g 構築・運用ガイド
- @DavidBuckJP
- https://blogs.oracle.com/buck/

Agenda

1. CHIP-8 History
2. CHIP-8 Architecture
3. Emulating CHIP-8
4. Advanced Emulation Topics

Introducing CHIP-8

Before CHIP-8



HOW TO "READ" FM TUNER SPECIFICATIONS

Popular Electronics

WORLD'S LARGEST-SELLING ELECTRONICS MAGAZINE JANUARY 1975/75¢

PROJECT BREAKTHROUGH!

World's First Minicomputer Kit to Rival Commercial Models... "ALTAIR 8800" SAVE OVER $1000

ALSO IN THIS ISSUE:

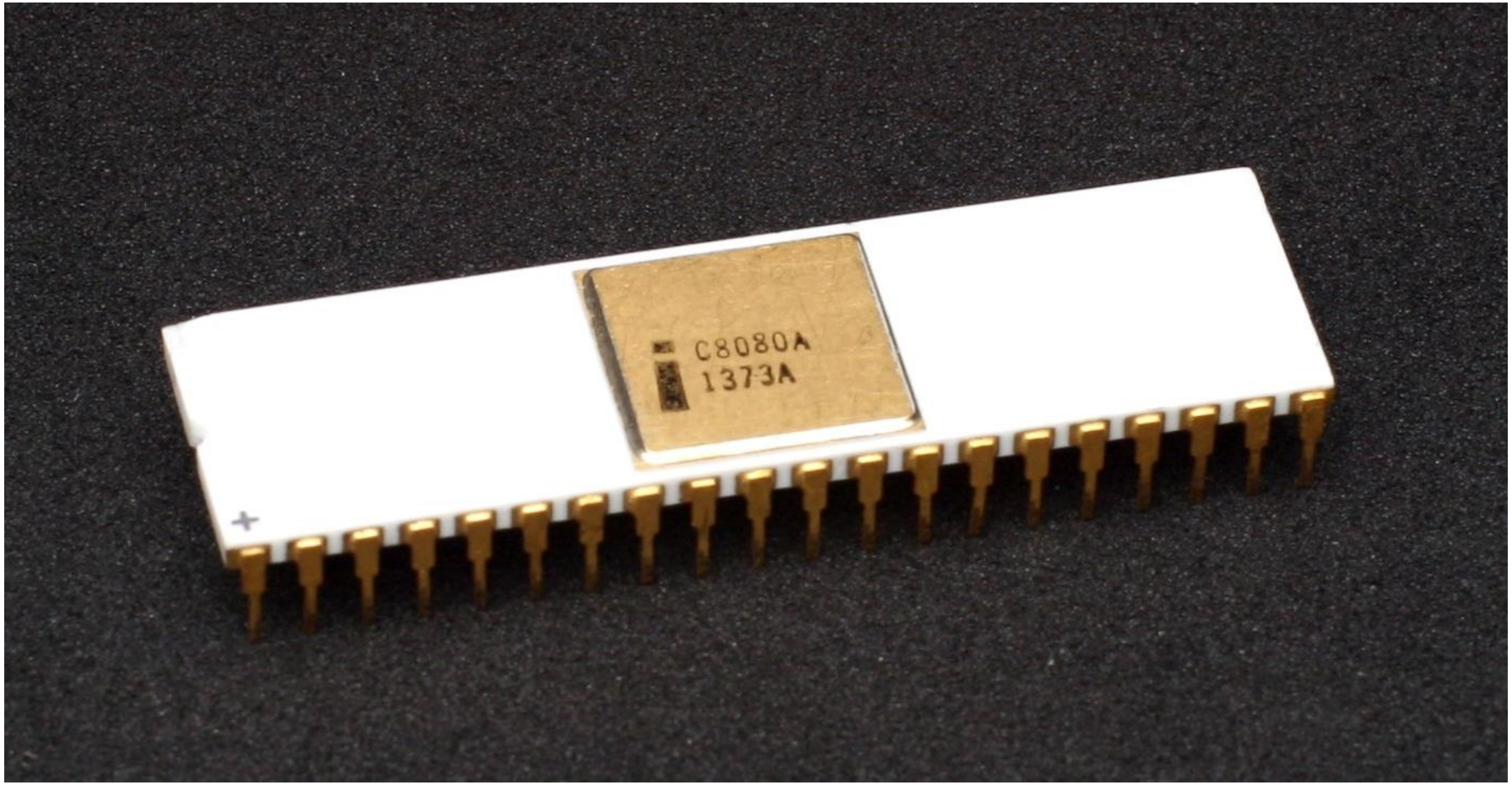
- An Under-$90 Scientific Calculator Project
- CCD's—TV Camera Tube Successor?
- Thyristor-Controlled Photoflashers

TEST REPORTS:

Technics 200 Speaker System
Pioneer RT-1011 Open-Reel Recorder
Tram Diamond-40 CB AM Transceiver
Edmund Scientific "Kirlian" Photo Kit
Hewlett-Packard 5381 Frequency Counter

https://en.wikipedia.org/w/index.php?curid=4327124

Intel 8080

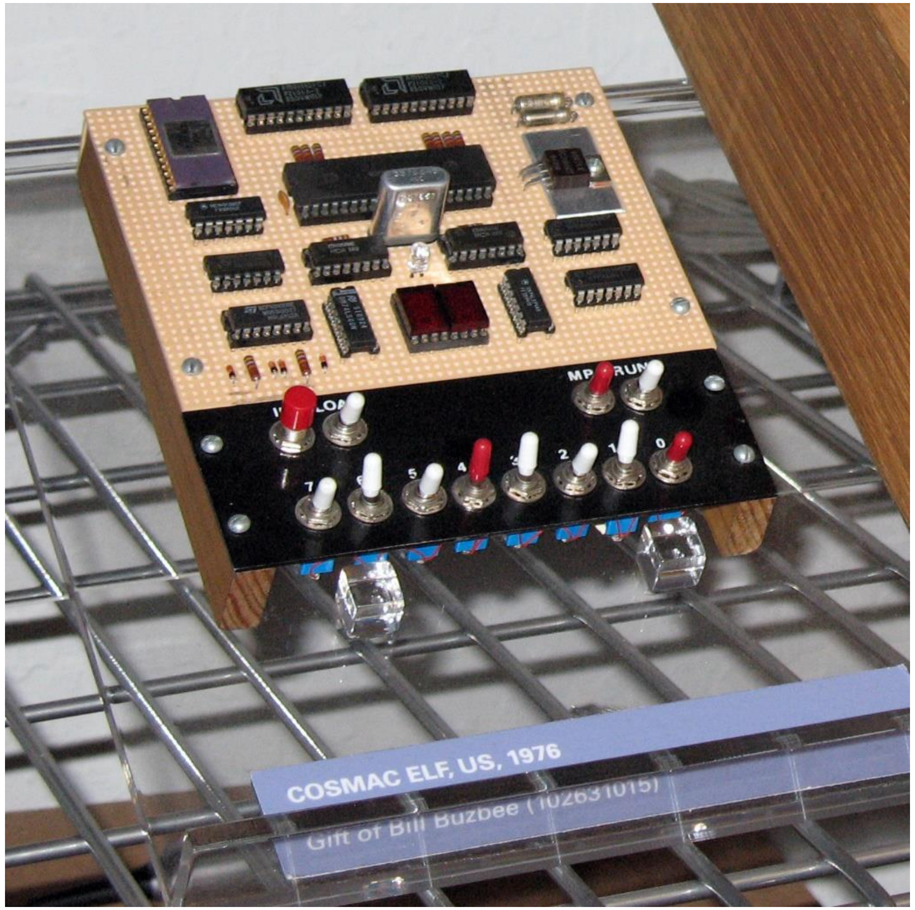


By Konstantin Lanzet - CPU collectionCamera: Canon EOS 400D, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=7028099

Not a Poor Nerd’s Hobby…

- 1975 kit price: 439 USD
- In 2017 currency: 1,997 USD

COSMAC ELF



By Swtpc6800 en:User:Swtpc6800 Michael Holley - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=3471056

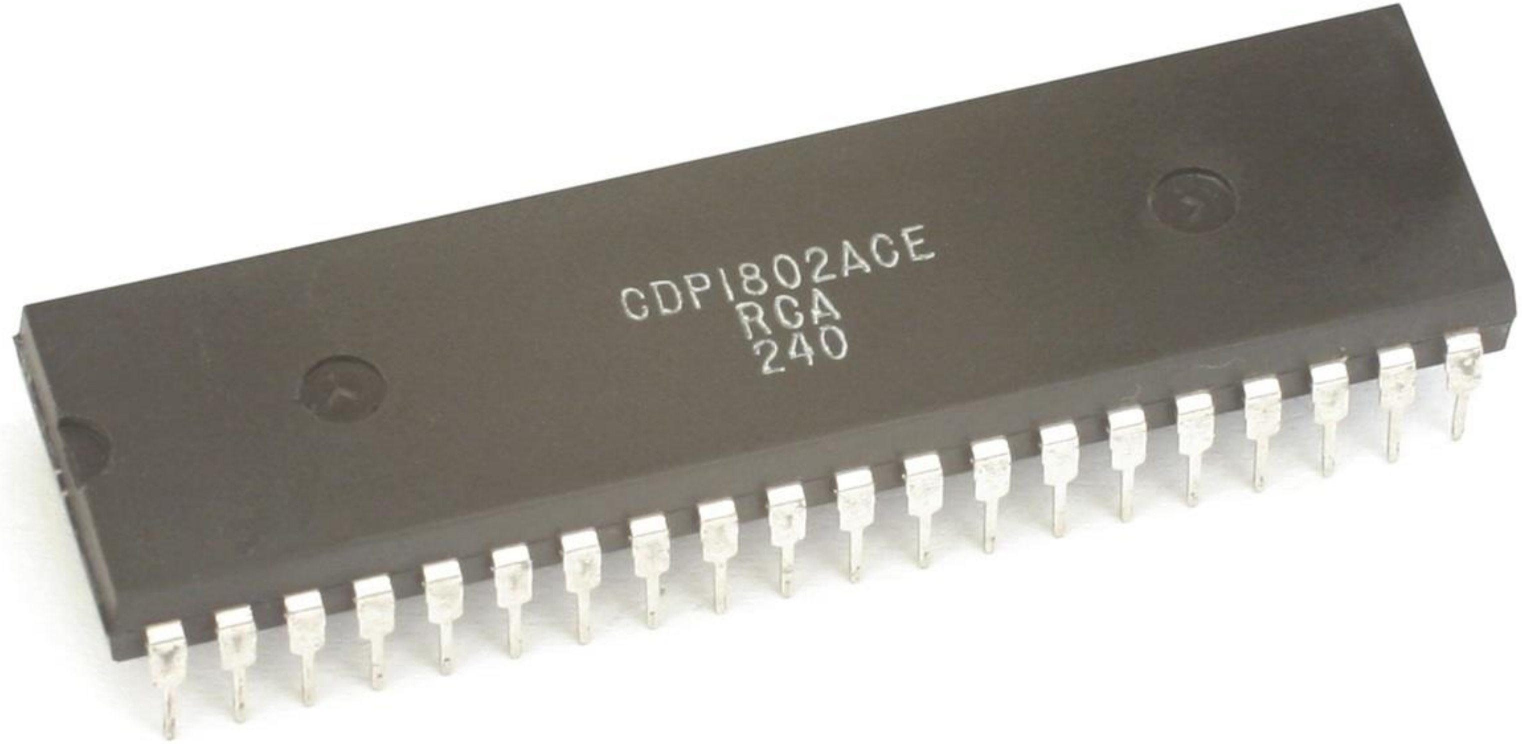
COSMAC ELF

- DIY project documented in Popular Electronics in 1976-1977
- Could be built for under 100 USD (1976 prices)
- Kits and pre-assembled boards offered from many vendors
- Very popular both then and now

Pixie

- RCA 1861 video generator IC
- 64 × 32 square pixels
- Needed 256 bytes of external RAM

RCA COSMAC 1802

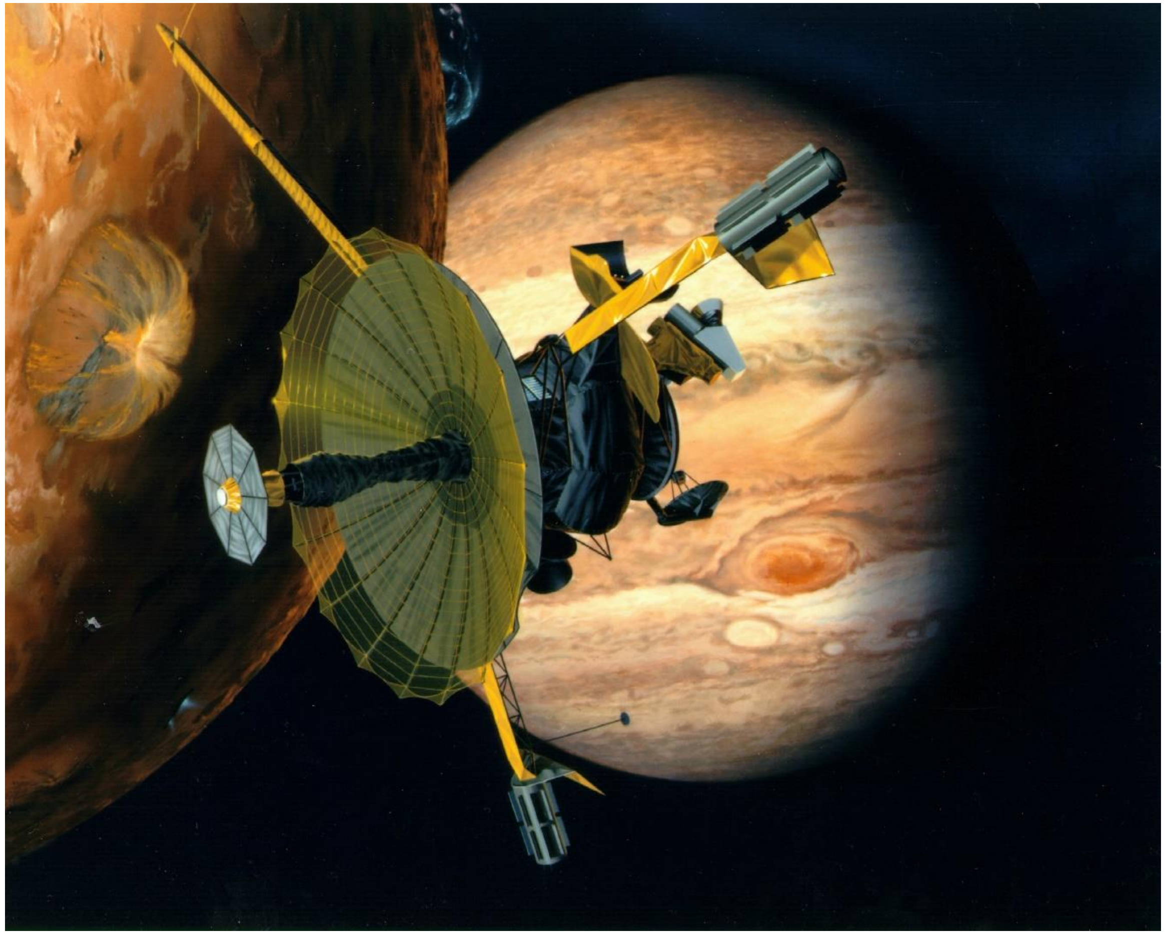


By Konstantin Lanzet (with permission) - CPU collection Konstantin Lanzet, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4774289

RCA COSMAC 1802

- Complementary Symmetry Monolithic Array Computer
- 1 chip revision of an earlier 2 chip design (1801)
- Very low cost compared to 8080
- Still in use in many applications today

Radiation Hardened Versions

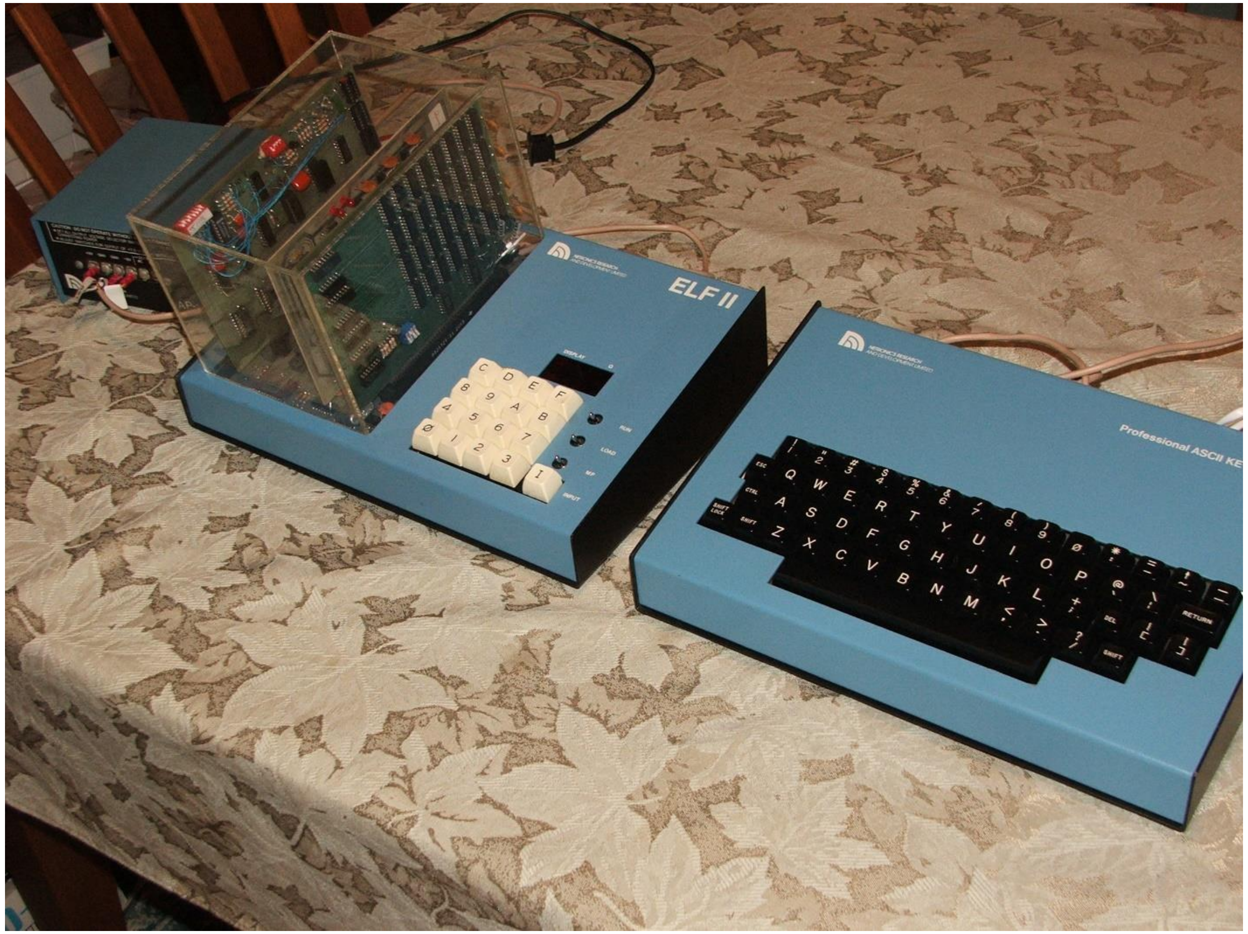


By NASA - http://solarsystem.nasa.gov/multimedia/display.cfm?IM_ID=2071 (image link)http://photojournal.jpl.nasa.gov/catalog/PIA18176 (image link) Public Domain, https://commons.wikimedia.org/w/index.php?curid=408298

COSMAC VIP

- Developed by Joseph Weisbecker
- Version of the ELF targeting the video game market
- Came pre-assembled
- Hex-keyboard was standard

Hex Keyboard



By John crane 59 - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=15795865

Hex Keyboard

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F

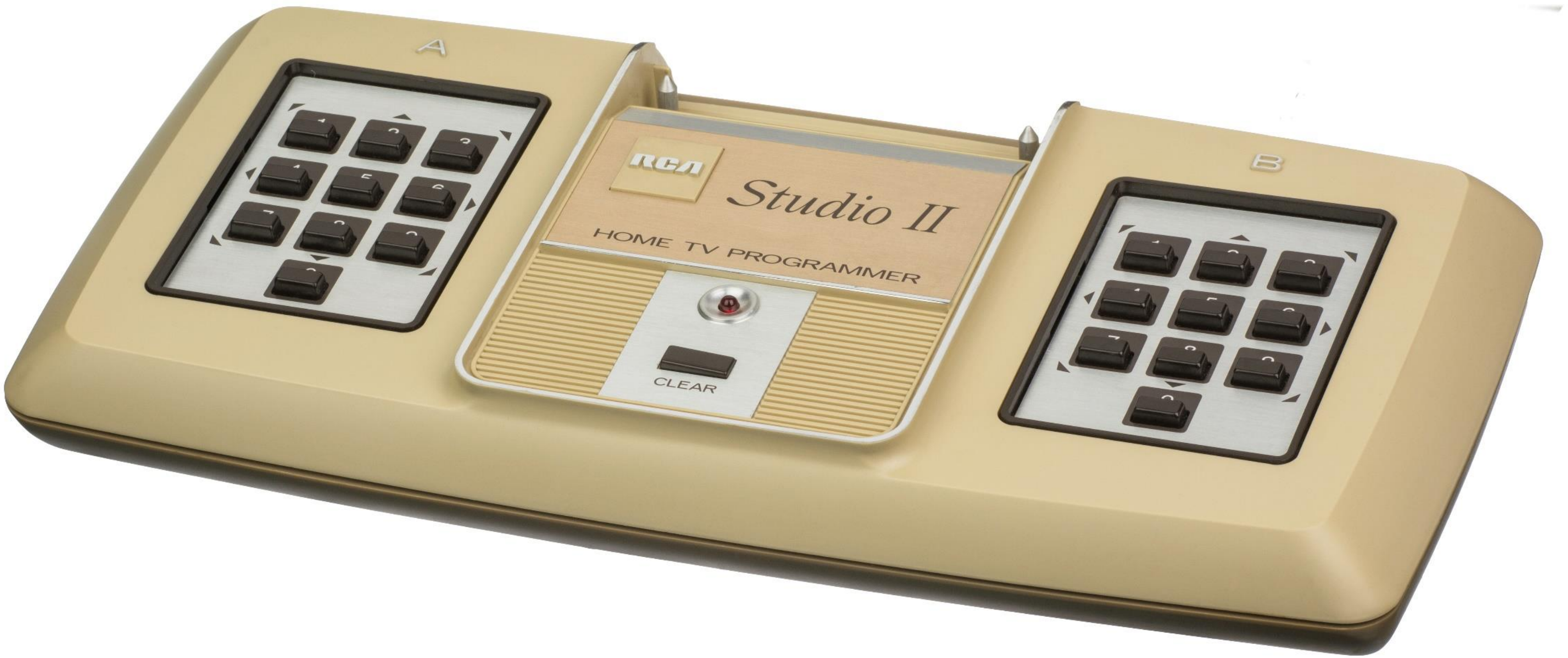
COSMAC VIP

- CPU: 1802
- ROM: 512-byte
- RAM: 2k~4k on-board (up to 32k external)
- Cassette interface: 100 bps
- 40 years old this year!

Similar to “CPU Trainers” of the day…

Public Domain, https://commons.wikimedia.org/w/index.php?curid=422438

RCA-Studio II



By Evan-Amos - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=38826714

CHIP-8

- A “language” for programming video games
- Developed by Joseph Weisbecker for the VIP
- Interpreter built into the VIP ROM
- Made writing games for the VIP much easier

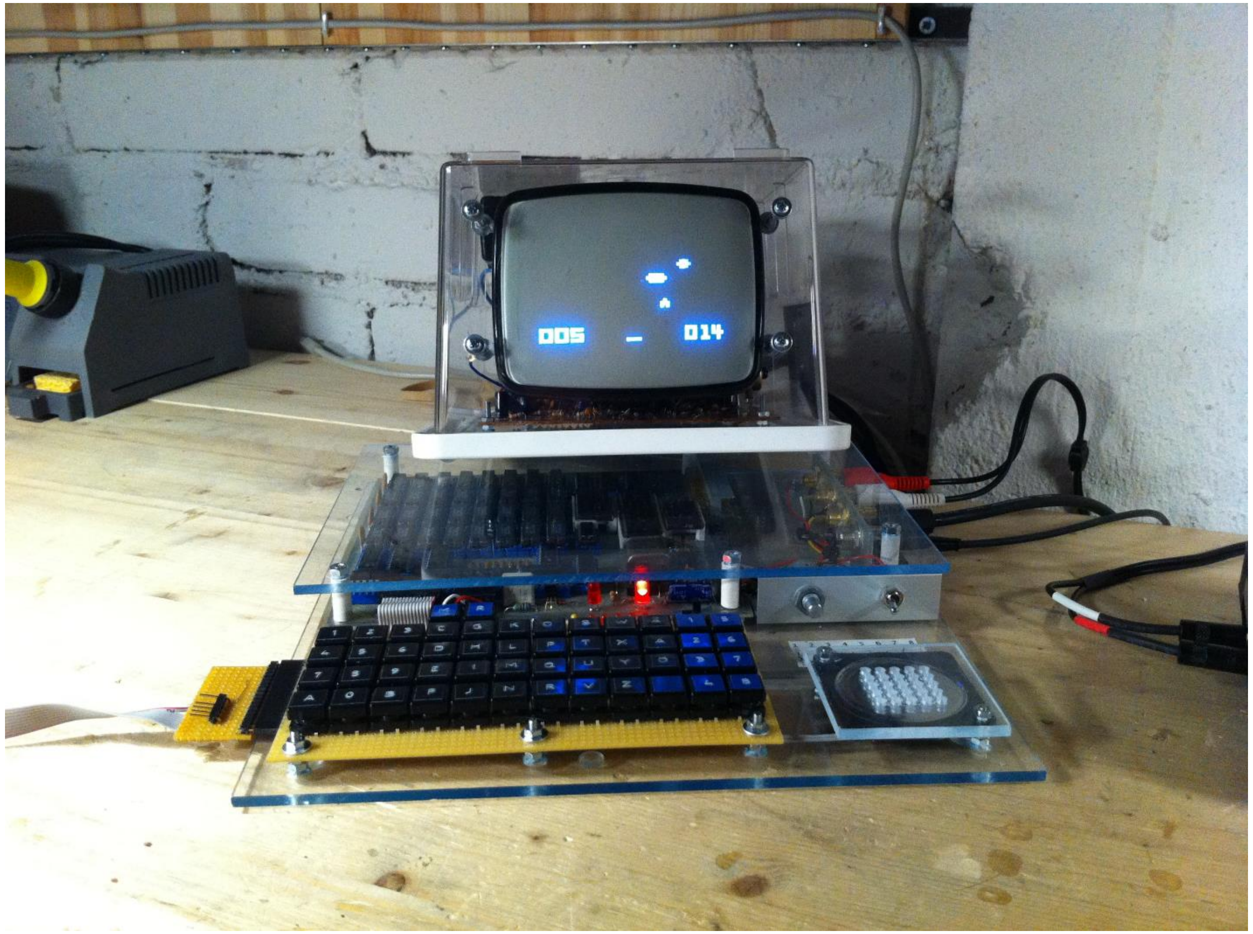
“Language”?

- Looks like machine code
- It was really a very early virtual machine specification

Why use a virtual machine?

- Abstract away much of the real hardware
  - Linier display buffer -> X,Y mapped grid
  - Sprite support
  - Sound support (q-line on 1802)
  - Key debouncing
- Very easy to program
- Very high code density

Other Platforms Followed (Telmac 1800)



By jpl - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=40541200

Fast Forward About 10 Years

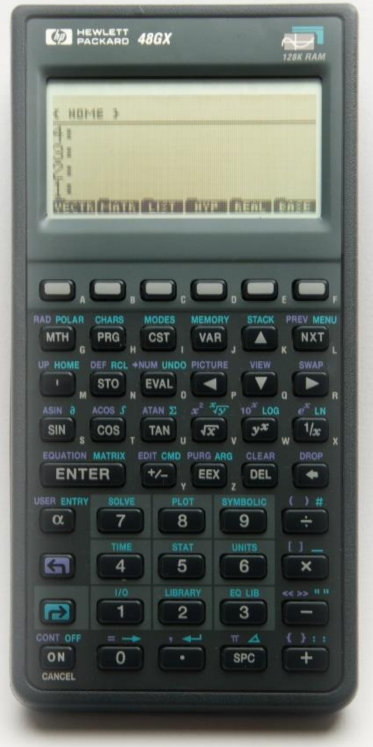
HP 28c released

- HP's first graphing calculator
- **Light-years** beyond anything else
  - RPL (lambda expressions!)
  - Symbolic math (CAS)

By Kurt Moerman (Kpmkpm) - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=4020998

The Legendary 48G

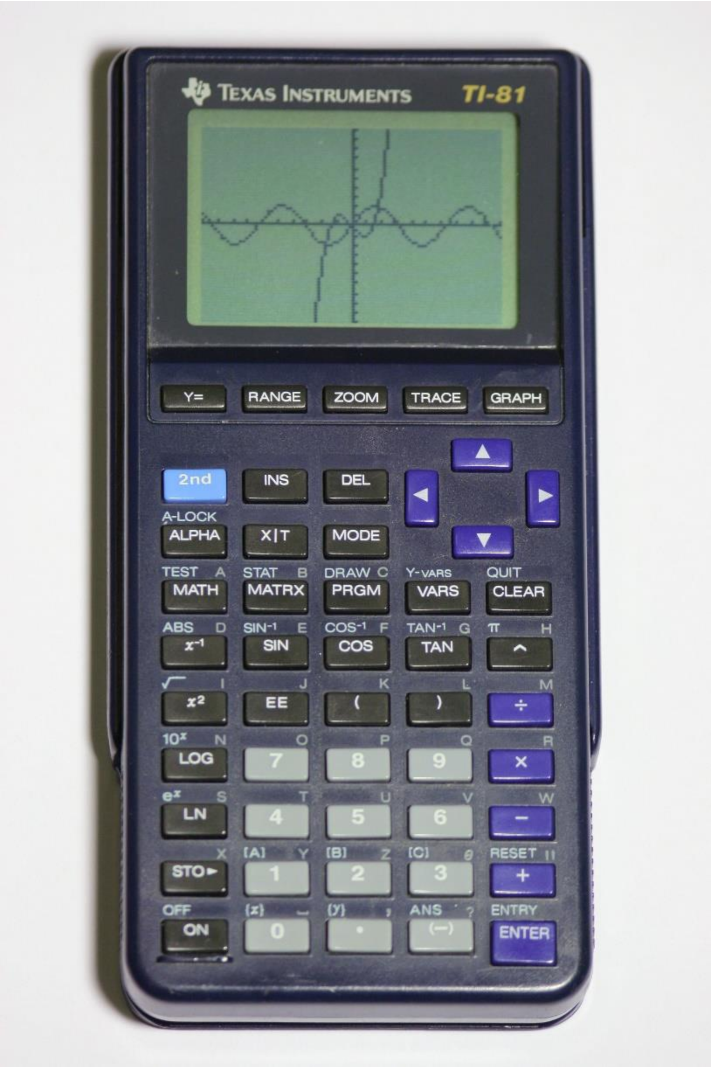
- Released in 1990
- HP’s most powerful calculator for almost a decade



By Feureau.com - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=9953201

Meanwhile…

- TI-81
- Z80 based
- BASIC-like programming language
- Very easy to program
- Very easy to make games



By Calcvids - I took this photo of my TI-81, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24252718

HP48G owners were jealous!

- RPL was **horrible** for gaming
  - Too abstract / high-level
  - Slow
  - Steep learning curve

CHIP-8 to the rescue!

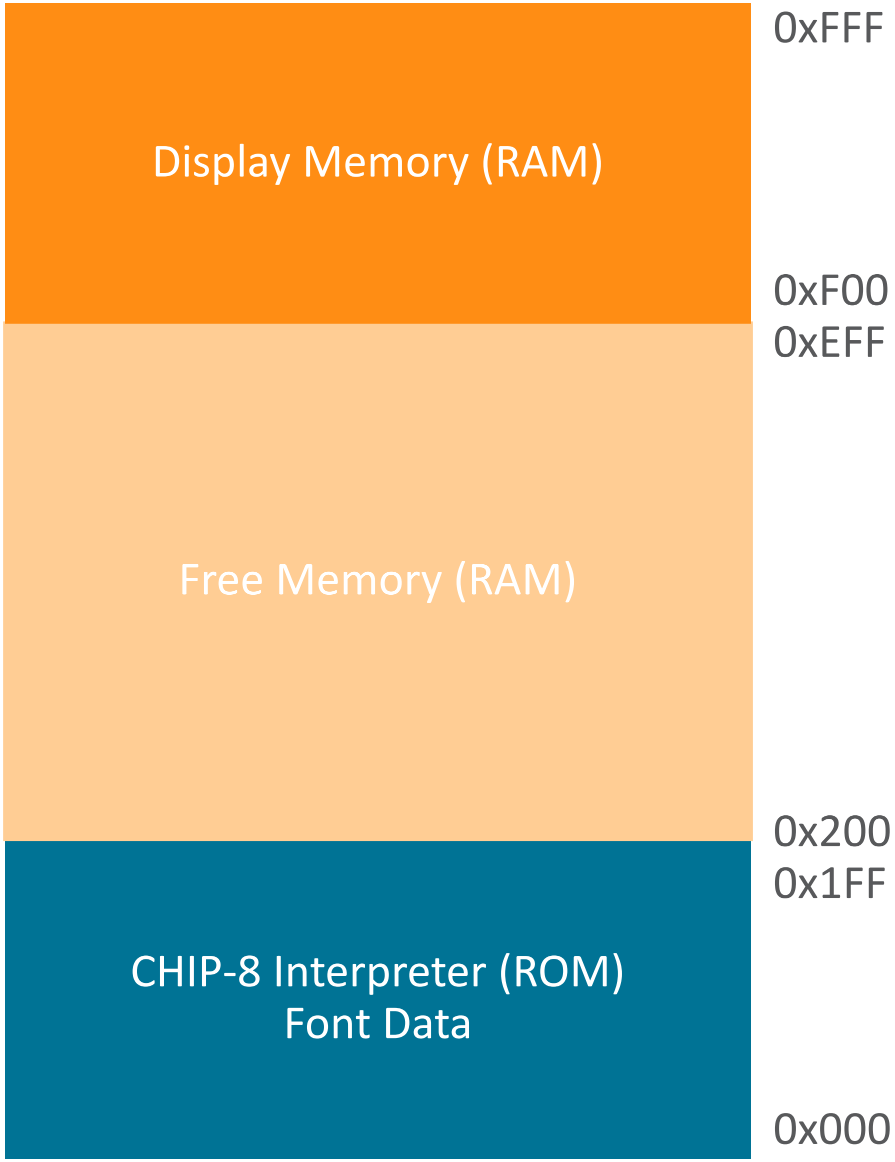
- Andreas Gustafsson wrote a CHIP-8 interpreter for the HP48
- HP48 users got access to library of preexisting software from the 70s
- New CHIP-8 software started to be written
- Super CHIP (SCHIP) expanded to take advantage of HP48 hardware

CHIP-8 Today

- A great way for future emulator authors to cut their teeth
- Probably the easiest platform to emulate that plays games
- Possibly the only platform with more emulators than native software

CHIP-8 Architecture

Memory Map


0xFFF
Display Memory (RAM)
0xF00
0xEFF
Free Memory (RAM)
0x200
0x1FF
CHIP-8 Interpreter (ROM)
Font Data
0x000

An Embarrassment of Registers

- 15 8-bit general registers (V0-VE)
- 1 8-bit “Flag” register (VF)
  - Only used for carry (ALU) and sprite hit detection
- 1 16-bit address register
  - Only 12-bit LSBs normally used
- 1 16-bit PC

Sound

- Write-only sound timer that decrements 60 ticks a second
- Buzzer / beeping sound plays until timer reaches zero
- No music here!

Display

- 64x32 monochrome pixels
- Sprites
  - 8 pixels wide
  - 1~15 pixels tall
  - Drawn to video memory by XOR:ing
  - Collision flag (FV) set (or cleared)

Timer

- Like sound timer, but no output.
- Decremented 60 ticks a second
- Stops at zero
- Can be both read or written to (unlike RO sound timer)

Opcodes

- AAA: address
- KK: 8-bit constant (byte)
- K: 4-bit constant (nibble)
- X, Y: 4-bit register identifier
- I, PC: 16bit register

Call instruction

- 0AAA – jump to 1802 code at AAA (not used in modern CHIP-8)

Display

- 00E0 CLS
- FX29 set I register to font for hex digit stored in X
- DXYK draw sprite K pixels high at X, Y coordinates

Flow Control Instructions

00EE	return from sub ★
1AAA	jump to AAA
2AAA	call sub at AAA ★
3XKK	if VX == KK skip next instruction
4XKK	if VX != KK skip next instruction
5XY0	if VX == VY skip next instruction

★ push and pop return address use implementation-defined stack

Loads / Stores / Moves

- 6XKK Vx = NN
- 8XY0 VX = VY
- AAAA I = AAA

ALU (Math) Instructions

- 7XKK Vx += KK
- 8XY1 Vx=Vx|Vy
- 8XY2 Vx=Vx&Vy
- 8XY3 Vx=Vx^Vy
- 8XY4 Vx += Vy
- 8XY5 Vx -= Vy
- 8XY6 Vx=Vy=Vy>>1
- 8XY7 Vx=Vy-Vx
- 8XYE Vx=Vy=Vy<<1

Input Instructions

- EX9E if (key()==Vx) skip next instruction
- EXA1 if (key()!=Vx) skip next instruction
- FX0A Vx = next key press (blocking wait)

Timers / Sound Instructions

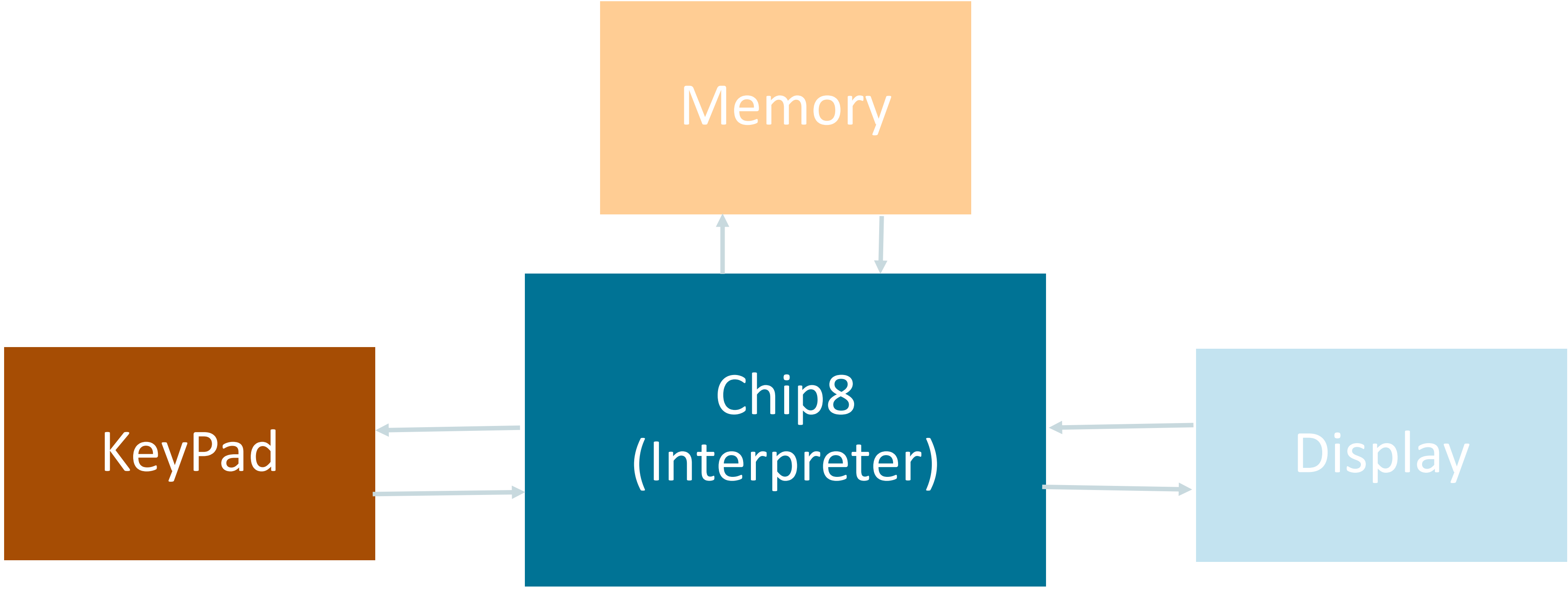
- FX07 Vx = delay_timer
- FX15 delay_timer = Vx
- FX18 sound_timer = Vx

Other Instructions

- CXKK Vx = random_number & KK
- FX33 stores 3 digits of BCD for value of Vx into I, I+1, I+2
- FX55 writes all V registers between V0 and Vx to I
- FX65 loads all V registers between V0 and Vx from I

Emulating CHIP-8

Overall Design



Overall Design

- Allows Display and KeyPad to be swapped out with different implementations
  - AWT
  - Swing
  - JavaFX
  - LWJGL
- Memory is isolated to be closer aligned to more advanced emulators

Memory

- One single 4K byte array
- Handles font data (must be initialized)
- Handles reading in of program (“rom”) during startup

Display

- ASCII ART inspired
- Appends characters to a StringBuilder
  - White Pixel ‘#’
  - Black Pixel ‘ ‘
- Note that screen buffer memory is independent from main memory
- Screen buffer is shared between Display and Chip8 (the interpreter)
- Screen refresh timing / method can have a huge performance impact

KeyPad

- Depends on enabling raw input from terminal
- Semantics of key press detection depend on automatic key repeat
  - By extension also depends on a very low repeat delay
- Shutdown hook is registered to restore terminal to useable state

Interpreter Loop

- Is a naive implementation
- Uses a simple series of nested switch statements
- Register values are stored in next larger primitive type
- Each iteration of the main loop updates timers as needed
- Each iteration of the main loop refreshes display as needed

Debugging

- You often want some way to debug what is running on the VM
  - Trace
  - Breakpoints
  - Memory dumps

Other General Emulation Tips

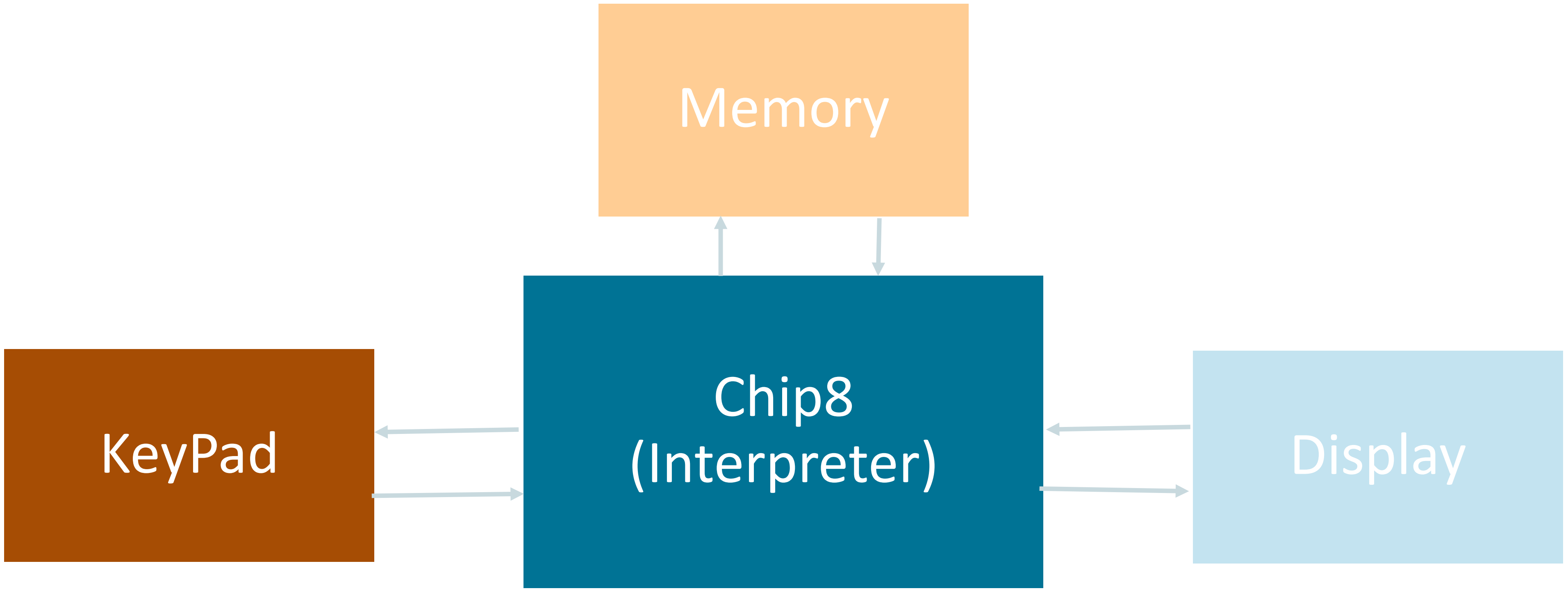
- Start with small test cases, not full games
- Write your own tests if you cannot find any
- Have some way to throttle execution speed
- Compare your emulator's behavior to other emulators

Advanced Emulation Topics

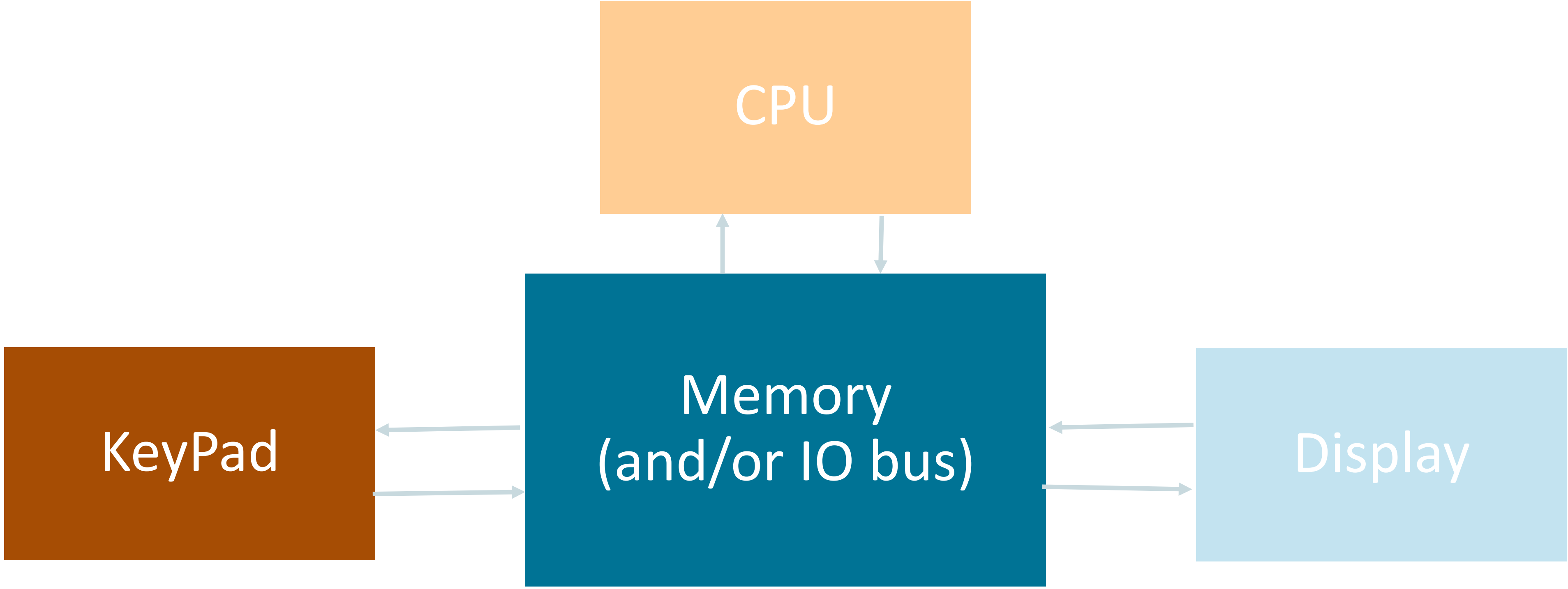
Your code should model the hardware

- Figure out how instructions are encoded / decoded (e.g. z80 decode)
- If there are multiple pieces of similar hardware, instantiating objects is probably best
- Multiple threads for interpreter loop / IO refresh (including display) may be the easiest design

My CHIP-8 Design



Typical Emulator Design



Dynamic Recompilation

- Use ASM to generate bytecode on the fly!
- Way easier than you might imagine
- Run your interpreter through ASMifier

  (temporarily refactoring each opcode implementation into its own method may help make things clearer)
- Write a new version of interpreter loop where implementation code is replaced by the corresponding ASMifier output
- The interpreter loop will now just linearly scan the “rom”, and generate corresponding bytecode.

Conclusions

- Emulation / Interpreters are fun to write in Java
- They are not as hard as many people imagine
- You can improve your general programming skillset by writing one
- Start out small. You can't get much simpler than CHIP-8

THANK YOU!

References

- Octo - Chip-8 Assembly Language & On-line IDE

https://johnearnest.github.io/Octo/

- Cowgod's Chip-8 Technical Reference

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

- Chip-8 Software for HP48 Archive

http://www.hpcalc.org/hp48/games/chip/

- Cosmac ELF

(A great fan site with a lots of ELF content)

http://www.cosmacelf.com/

- Matthew Mikolay's Retro Computing Site

(Has scans of VIPER magazine among a ton of other great information)

http://retro.mattmik.com/

Stay connected

- Join us: **DevOps Corner** (Developer Lounge – Moscone West)
- Learn more: **openjdk.java.net | wercker.com/java**
- Follow: **@OpenJDK, @wercker #JavaOne #DevOps**

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

JavaOne™

ORACLE®

ORACLE®