

Emulators and Web Assembly

Dr. Charles R. Severance

online.dr-chuck.com

Outline

- Start from our "two instruction CPU" from the previous section
- Hexadecimal Notation
- Review: What is Programming from PY4E
- Real CPUs – Intel x86, ARM, and MOS 6502
- The CDC6504 – Mostly 6502 with homage to CDC 6500
 - Architecture
 - Machine Language
 - Assembly Language

Uppercase a String

CLX

loop:

LDA \$00,X

BEQ done

CMP #'a'

BMI cont

SBC #0x20

STA \$00,X

cont:

INX

JMP loop

done:

BRK

DATA 'Hello'

Status: Stopped

Printed: No output yet

Reset

Step

Start

PC:

0x00

Z:

N:

ACC:

0x00

X:

0x00

Y:

0x00

Instructions

0x00: 11100010

0x01: 10110101

0x02: 00000000

0x03: 11110000

0x04: 00001011

0x05: 11001001

0x06: 01100001

0x07: 00110000

0x08: 00000100

Instructions

0x10: 00000000

0x11: 00000000

0x12: 00000000

0x13: 00000000

0x14: 00000000

0x15: 00000000

0x16: 00000000

0x17: 00000000

0x18: 00000000

Memory

0x00: 0x48

0x01: 0x65

0x02: 0x6C

0x03: 0x6C

0x04: 0x6F

0x05: 0x00

0x06: 0x00

0x07: 0x00

0x08: 0x00

```
executeStep() {
    // Read instruction from instruction memory
    const instruction = this.cpu.instructions[this.cpu.pc];

    // Decode 6502 opcodes
    if (instruction === 0x00) { // BRK - Break/Halt
        this.printString(); // Print data memory on BRK
        this.running = false;
    }
    ...
    else {
        console.log(`ERROR: unknown instruction - halting`);
        this.running = false;
    }

    // Update PC (if not already updated by branch/jump)
    if (pcIncrement > 0) {
        this.cpu.pc = (this.cpu.pc + pcIncrement) & 0xFF;
    }
}
```

Status: Halted
Printed: *

Reset Step Start

PC: Z: N:
0x05

ACC: X: Y:
0x2A 0x00 0x00

Instructions	Memory
0x00: 10101001	0x00: 0x2A
0x01: 00011011	0x01: 0x00
0x02: 01101001	0x02: 0x00
0x03: 00001111	0x03: 0x00
0x04: 10000101	0x04: 0x00
0x05: 00000000	0x05: 0x00

```
else if (instruction === 0xA9) { // LDA # - Load accumulator immediate
    const immediate = this.cpu.instructions[this.cpu.pc + 1];
    this.cpu.acc = immediate;
    this.updateCompareFlags(this.cpu.x, immediate);
    pcIncrement = 2;
} else if (instruction === 0xE9) { // SBC # - Subtract immediate
    const immediate = this.cpu.instructions[this.cpu.pc + 1];
    const diff = this.cpu.acc - immediate;
    this.cpu.acc = diff & 0xFF;
    this.updateStatusFlags(this.cpu.acc);
    pcIncrement = 2;
} else if (instruction === 0xB5) { // LDA $nn,X - Load indexed by X
    const baseAddr = this.cpu.instructions[this.cpu.pc + 1];
    const xValue = Number(this.cpu.x);
    const effAddr = (baseAddr + xValue) & 0xFF;
    this.cpu.acc = this.cpu.memory[effAddr];
    this.updateStatusFlags(this.cpu.acc);
    pcIncrement = 2;
}
```

Building Emulators for Historical CPUs

- It is not all that difficult to build an emulator for an older CPU (emulator.js is 1300 lines of code – 6502 is 4528 transistors)
- Modern processors on our phones and computers are so fast that JavaScript in a browser can emulate historical machine code faster than the original hardware
- The computer / phone / browser graphics are also far better
- And it can be great fun

Archive.org: Internet Arcade

The Internet Archive logo is at the top left. The main navigation menu includes WEB, TEXTS, VIDEO, AUDIO, SOFTWARE, IMAGES, SIGN UP | LOG IN, UPLOAD, and a search bar.

Console Living Room: Atari 2600

The Atari 2600 is a video game console released in September 1977 by Atari, Inc. It is credited with popularizing the use of microprocessor-based hardware and ROM cartridges containing game code, a format first used with the Fairchild Channel F, instead of having non-microprocessor dedicated hardware with all games built in. The console was originally sold as the Atari VCS, for Video Computer System. Following the release of the Atari 5200, in 1982, the VCS was renamed "Atari 2600", after the unit's Atari part number, CX2600. The 2600 was typically bundled with two joystick controllers, a conjoined pair of paddle controllers, and a cartridge game—initially Combat and later Pac-Man. The Atari 2600 was wildly successful, and during much of the 1980s, "Atari" was a synonym for this model in mainstream media and, by extension, for video games in general. The Atari 2600 was inducted into the National Toy Hall of Fame at The Strong in Rochester, New York in 2007. In 2009, the Atari 2600 was named the second greatest video game console of all time by IGN, who cited its remarkable role as the console behind both the first video game boom and the video game crash of 1983, and called it "the console that our entire industry is built upon."

[More...](#)

COLLECTION **ABOUT**

3,006 Results

Year Published 1978 - 2025

Part Of

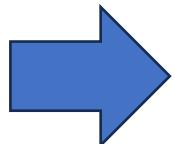
Console Living Room
The Emulation Station
software

Search **Search this collection** [Advanced Search](#)

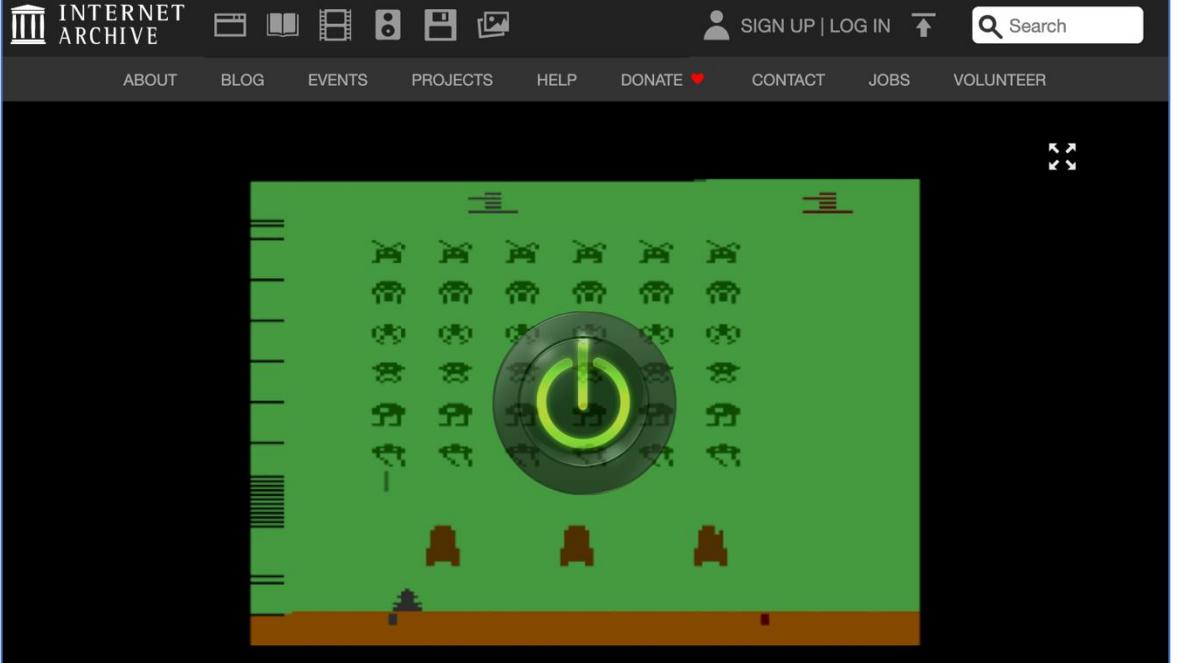
Sort by: Weekly views ▾ Title Date published ▾ Creator

Icon grid view:

Atari 2600 Space Invaders



The Atari 2600 has a 6502 processor.



A screenshot of the Atari 2600 version of the game Space Invaders. The screen shows a green playfield with a grid of alien invaders at the top. In the center is a circular laser cannon with a bright yellow beam. At the bottom, there are four small human-shaped player characters. The Internet Archive interface is visible at the top, showing navigation links like 'ABOUT', 'BLOG', 'EVENTS', 'PROJECTS', 'HELP', 'DONATE', 'CONTACT', 'JOBS', and 'VOLUNTEER'. Below the game image, there is a summary box with the following information:

	Atari 2600: Space Invaders
Publication date	1980
Item Size	492.3K
Each time you turn on SPACE INVADERS you will be at war with enemies from space who are threatening the earth. Your objective is to destroy these invaders by firing your "laser cannon." You must wipe out the invaders either before they reach the earth (bottom of the screen), or before they hit you three times with their "laser bombs."	
	Favorite
	Share
	Flag
24,529 Views	
27 Favorites	
1 Review	

Space Invaders Startup – JSMAME Emulator



INTERNET ARCHIVE

ABOUT BLOG EVENTS PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER

Search

MAME

Downloading game data...

✓	Game Metadata
✓	Game File List
✓	Emulator Metadata
✓	Game File (1 of 1)
✓	CFG File

WASM Binary (42%; 1.7 MiB of 4.2 MiB)

The screenshot shows the Internet Archive's download interface for the MAME emulator. The main title 'MAME' is displayed in large blue letters. Below it, the text 'Downloading game data...' is shown. A list of download items is presented in a table with a white background and black borders. Each item has a checked checkbox icon to its left. The items listed are: Game Metadata, Game File List, Emulator Metadata, Game File (1 of 1), and CFG File. The 'Game File (1 of 1)' item is currently being downloaded, as indicated by the progress bar below it. At the bottom of the screen, a status message shows 'WASM Binary (42%; 1.7 MiB of 4.2 MiB)'. The top navigation bar includes links for About, Blog, Events, Projects, Help, Donate, Contact, Jobs, and Volunteer, along with a sign-up/login form and a search bar. The Internet Archive logo is at the top left, and various document icons are at the top right.

WASM – Web Assembly

- There is a machine language emulator built into browsers called Web Assembly
- For a wide range of software, compiling to WASM and running in JavaScript is enough
- WASM is a different syntax than most Assembly languages – but it still compiles to a binary-machine code
- C Programming for Everybody (www.cc4e.com) C Playground compiles C to WASM and then runs the WASM in the user's browser

WASM versus traditional Assembly

- **Sandboxed by design:** Wasm runs in a strict sandbox with no direct access to memory, files, or OS — unlike native assembly which runs with full process privileges.
- **Safe execution model:** Structured control flow, validated code, and no undefined jumps help prevent many classic crashes and exploits.
- **Portable bytecode:** Same Wasm runs on any browser or runtime; traditional assembly is tied to a specific CPU (x86, ARM, etc.).
- **Abstract machine:** Wasm targets a virtual stack machine, not real hardware — traditional assembly maps directly to CPU instructions.
- **Optimized by the host:** Browsers and runtimes validate and compile Just In Time (JIT) - compile Wasm safely, while native assembly relies on machine code produced elsewhere with no validation.

```
;; Hello World WASM Example
(module
  (import "console" "log" (func $log (param i32 i32)))
  (memory 1)
  (data (i32.const 0) "Hello, World!")
  (func $main (result i32)
    (call $log (i32.const 0) (i32.const 13))
    (i32.const 42)
  )
  (export "main" (func $main))
)
```

WASM
Hello World

00000000:	00 61 73 6d 01 00 00 00 01 0a 02 60 02 7f 7f 00	.asm.....`....
00000010:	60 00 01 7f 02 0f 01 07 63 6f 6e 73 6f 6c 65 03	`.....console.
00000020:	6c 6f 67 00 00 03 02 01 01 05 03 01 00 01 07 11	log.....
00000030:	02 06 6d 65 6d 6f 72 79 02 00 04 6d 61 69 6e 00	..memory...main.
00000040:	01 0a 0c 01 0a 00 41 00 41 0d 10 00 41 2a 0b 0bA.A...A*..
00000050:	13 01 00 41 00 0b 0d 48 65 6c 6c 6f 2c 20 57 6f	...A...Hello, Wo
00000060:	72 6c 64 21	rld!

This is the www.cc4e.com code playground for writing C programs. You can also check [recent status](#) of this compiler page.

[Run Code](#) [Back to CC4E](#)

```
1 #include <stdio.h>
2
3 main() {
4     printf("Hello World\n");
5 }
6
```

Program Output:

Hello World

This page uses a server-based compiler called [Emscripten](#) that compiles your code to JavaScript and then executes your code in the browser. You can watch your browser developer console to monitor how your code is being executed. If this fails with an unexpected error, please add a note in the [Discussions](#) area.

From the 6502 to the Present

A lot has changed since the 6502 was available as a \$25 microprocessor

Modern CPUs

- Since the 1970's there have been about 50 distinct CPU families
 - Lots of research and innovation
 - Adoption and switching
 - Generational turn over
 - From CISC to RISC
 - From mainframe to microprocessor
- Today there are two major CPU families in use (computers + phones)
 - ARM based processors (90%)
 - Intel x86 processors (10%)

Case Study: Apple Processors

- MOS 6502 – Apple I / Apple II – 1977 - 1993
- Motorola MC68000 – Macintosh 1984-1994
- ARM – Apple Newton - 1993
- IBM PowerPC – Macintosh 1994 – 2006
- Intel x86 – Macintosh (2006-2020)
- Samsung ARM – iPhone (2007-2009)
- Apple ARM – iPhone (2010-Present)
- Apple ARM – Macintosh (2020-Present)

- Apple currently redesigns their CPUs based on workload over time – Image Processing / Audio / Video / Gaming / Artificial Intelligence

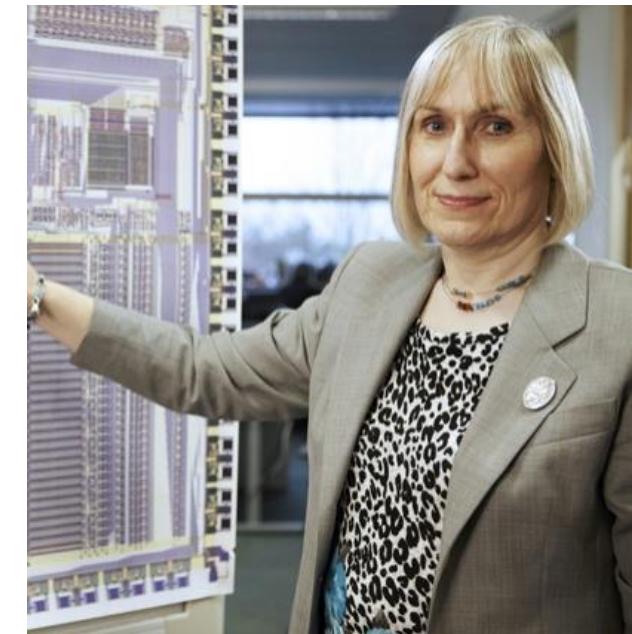
Fun Aside – Apple Built and Shipped Machine Code Translators

- 2005 Moved from PowerPC to Intel Processors – Rosetta 1
 - 2020 Moved from Intel to Apple ARM Processors – Rosetta 2
-
- Translated machine code as the application was launched
 - Greatly eases what would otherwise be a major upheaval for customers



Fun Aside – the 6502 influenced ARM

- Acorn's 6502-based BBC Micro
 - The ARM designers were 6502 experts
 - The BBC Micro was used to build and simulate the ARM design
 - ARM1 had ~25,000 transistors
 - Currently ARM based processors are the most widely used processors in the world.
-
- AI: Describe the historical connection between the 6502, ARM, and Raspberry Pi



Sophie Wilson

Summary

- CPU Evolution over time

Acknowledgements / Contributions

These slides are Copyright 2025- Charles R. Severance (online.dr-chuck.com) as part of www.ca4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

[Continue new Contributors and Translators here](#)

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates