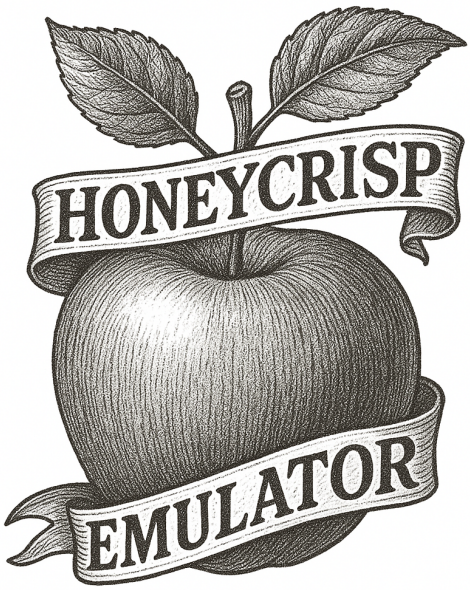

HONEYCRISP EMULATOR

Technical Manual

VERSION: 1.01

Written by Landon J. Smith

HoneyCrisp Emulator: Technical Manual

Version 1.01 | **Author:** Landon J. Smith | **Date:** September 30th, 2025
Welcome to the official technical manual for the HoneyCrisp Emulator. This document is your guide to understanding the inner workings of this Apple-1 emulator. It covers a variety of topics concerning the technicalities of the emulation system, and also includes the complete source code for HoneyCrisp v1.0 under the MIT license.

1. Introduction

The HoneyCrisp Emulator is a browser-based recreation of the original Apple-1 computer designed by Steve Wozniak in 1976. This manual provides comprehensive technical documentation for developers, historians, and hobbyists who desire to understand the internal workings of this emulator.

1.1. Purpose

This emulator serves multiple purposes:

- Provide an educational tool for understanding 6502 architecture.
- Enable execution of original Apple-1 software.
- Offer an accessible platform for retrocomputing enthusiasts.
- A "test bed" for the author's personal use/expansion of knowledge

1.2. Design Philosophy

HoneyCrisp prioritizes hardware accuracy over performance shortcuts. The emulator implements cycle-accurate 6502 instruction execution, authentic video display characteristics including the VRAM power-on behavior (_@ pattern), proper timing for I/O operations, and more.

1.3. System Requirements

The emulator requires:

- A modern web browser with JavaScript ES6 support.
- A keyboard for input operations.
- No additional plugins or installations.

2. System Architecture

2.1. CPU Emulation (6502)

The core of HoneyCrisp is a complete MOS Technology 6502 CPU emulator implemented in JavaScript as the CPU6502 class. The 6502 CPU emulation includes:

2.1.1. Register Implementation

All 6502 registers are emulated as JavaScript properties:

- A - Accumulator (8-bit)
- X, Y - Index registers (8-bit each)
- PC - Program Counter (16-bit)
- S - Stack Pointer (8-bit)
- C, Z, I, D, B, V, N - Status flags (1-bit each)

2.1.2. Instruction Execution

The step() method executes one instruction per call by performing the following actions:

1. Fetch opcode from memory at PC.
2. Increment PC.
3. Decode opcode using an instruction map.
4. Execute the corresponding operation.
5. Update the cycle count.
6. Check for and apply page crossing penalties.

2.1.3. Crash Detection

The emulator includes safety mechanisms to detect and recover from crashes. If execution reaches address $0000 with a BRK instruction, the system automatically performs a hard reset to prevent infinite loops.

2.2. Memory Map

HoneyCrisp implements the original Apple-1 memory layout:

Address Range	**Size**	**Description**
$0000-$0FFF	4KB	RAM (configurable to 8KB: $0000-$1FFF)
$C100-$C1FF	256B (Program compatibility only)	Apple Cassette Interface ROM
$D010-$D013	4B	I/O Mapped Registers
$E000-$EFFF	4KB	Integer BASIC ROM
$FC00-$FFFF	1KB	WOZMON Monitor

2.2.1. I/O Registers

- $D010 - Keyboard data input (bit 7 = ready flag)
- $D011 - Keyboard status (bit 7 = data available)
- $D012 - Display output register
- $D013 Display Control (modern convenience.)

2.2.2. Memory Access Methods

The read(addr) and write(addr, val) methods handle all memory access, implementing proper I/O mapping and ROM write protection.

2.3. Video Display System

The video subsystem emulates the Apple-1's 40-column by 24-row character display using a virtual framebuffer.

2.3.1. Display Architecture

- **Screen buffer:** 2D array (24 rows × 40 columns)
- **Cursor tracking:** x,y coordinates
- **Scrolling:** Automatic when the bottom of the screen is reached
- **Rendering:** Character-based (no bitmap graphics used to keep low file size)

2.3.2. Character Set

The emulator uses a custom font reproducing the Signetics 2513 character ROM, limited to the authentic Apple-1 character set:

- **Uppercase letters:** A-Z
- **Digits:** 0-9
- **Special characters:** @ [\ ] ^ _ ! " # $ % & ' ( ) * + , - . / : ; < = > ?
- **Blank Space** (BS)

2.3.3. Cursor Blinking

The WOZMON prompt cursor (@) blinks at 530ms intervals, matching the original hardware behavior. This is implemented via setInterval() calling the renderScreen() function.

2.3.4. Boot Pattern

Upon initialization, the screen displays an alternating underscore and @ pattern (_@...) representing uninitialized video RAM, faithfully recreating the Apple-1 power-on appearance.

3. Technical Specifications

3.1. Instruction Set Implementation

HoneyCrisp implements all 151 documented 6502 opcodes plus common "illegal" opcodes used by some Apple-1 software. (Such as WOZMON) Not all operation codes are listed within this manual for brevity.

3.1.1. Documented Instructions

Category	Opcodes
Load/Store	LDA, LDX, LDY, STA, STX, STY
Transfer	TAX, TXA, TAY, TSX, TXS
Stack	PHA, PLA, PHP, PLP
Arithmetic	ADC, SBC, INC, INX, INY, DEC, DEX, DEY
Logic	AND, ORA, EOR
Shift	ASL, LSR, ROL, ROR
Compare	CMP, CPX, CPY, BIT
Brach	BCC, BCS, BEQ, BNE, BMI, BPL, BVC, BVS
Jump	JMP, JSR, RTS, RTI
Flag	CLC, SEC, CLI, SEI, CLV, CLD, SED
System	BRK, NOP

3.1.2. Illegal Opcodes

The following undocumented opcodes are supported for compatibility:

Category	Opcodes
Load/Store Combos	LAX, SAX
Decrement/Increment Combos	DCP, ISP
Shift/Rotate Combos	SLO, RLA, SRE, RRA
Accumulator Ops	ANC, ALR, ARR, XAA, AXS
Store High Byte	SHY, SHX

3.1.3. Addressing Modes

All 13 standard 6502 addressing modes are implemented, including: Immediate, Zero Page, Absolute, Indirect, Relative, and all indexed variations.

3.2. Cycle Timing

HoneyCrisp implements cycle-accurate timing for all instructions. The cycleTable array stores the base cycle count for each opcode.

3.2.1. Timing Accuracy

- Base cycles are pulled from a 256-entry lookup table.
- Page crossing penalties are added automatically.
- Branch taken penalties are calculated (+1 cycle, or +2 if page is crossed).
- A total cycle counter tracks the emulation's progress.

3.2.2. Execution Throttling

The tick() function runs continuously via requestAnimationFrame(), executing 16,667 cycles per frame to approximate the original 1MHz operation at 60fps.

3.3. I/O Handling

3.3.1. Keyboard Input

Keyboard input is buffered in the _kbdBuf array.

- Characters are converted to uppercase.
- ASCII codes are OR'd with $80 (bit 7 is set).
- Special keys are mapped: Enter→$8D, Backspace→$DF, Escape→$9B.
- Reading $D010 returns a character and clears the buffer.
- Reading $D011 returns $80 if a character is available.

(With the exception of BS character + return, $0D)

3.3.2. Display Output

Character output is handled through the _videoHook.

- Writing to $D012 triggers output.
- The character is masked to 7-bit ASCII.
- Output is queued in videoOutputQueue and rate-limited to 30 chars/sec.

(The videoOutputQueue is implemented to prevent output throttling)

- A carriage return ($0D) triggers a newline.

4. ROM Components

4.1. WOZMON Monitor

WOZMON (Wozniak Monitor) is the system monitor that provides an environment for programming the APPLE-1.

Location:
- $FF00-$FFFF (256 bytes, mapped at $FC00-$FFFF).
- The reset vector at $FFFC points to $FF00.

Commands:
- Examine memory (AAAA),
- Modify memory (AAAA: BB CC...),
- Run programs (AAAA R).

Implementation:
- WOZMON is loaded from a base64-encoded ROM image via loadWozmonROM().

4.2. Integer BASIC

Apple Integer BASIC provides a complete programming environment.

- **Location:** $E000-$EFFF (4KB). The entry point is $E000.
- **Features:** Integer-only arithmetic, 26 variables (A-Z), arrays (DIM), loops (FOR/NEXT), subroutines (GOSUB/RETURN), and memory access (PEEK/POKE).
- **Activation:** From WOZMON, enter **E000R** to start BASIC.

4.3. Apple Cassette Interface (ACI)
The ACI ROM provides cassette tape loading and saving functionality.
As of HoneyCrisp v1.0, it serves as a compatibility layer for specific software.
Location: $C100-$C1FF (256 bytes).
Commands: C100R (Read) and C100W (Write).

Note: Due to compatibility only functionality, the commands above are non-functional.

5. User Interface

5.1. Control Buttons

HARD RESET: Clears RAM, resets CPU registers, and un-initalizes video RAM.
SOFT RESET: Restarts to WOZMON while preserving RAM contents.
CLEAR SCREEN: Erases the display without changing the system state.
BREAK: Interrupts an Integer BASIC program (Only applicable to $E000-$EFFF env.)

5.2. Memory Configuration

Radio buttons allow the user to select the RAM size between **4KB** (**$0000-$0FFF)** and **8KB ($0000-$1FFF). If memory configuration is changed, the page will automatically reload.**

6. Program Loading (.hc Format)

6.1. File Format
HoneyCrisp programs use the .hc file format for memory loading.
An example of the formatting is listed below.

AAAA (addr1)
: BB BB BB ...
CCCC (addr2)
: DD DD DD ...

A four-digit hex address **(AAAA)** sets the load location.
A colon **(:)** indicates a line of data bytes.
Bytes are space-separated hexadecimal values **(BB)**.

6.2. Loading Process

The user clicks the browse button labeled **"Load a Program (.hc)"** and selects a file.
A parser reads the file, sets the memory pointer based on the address lines, and writes the data bytes into the specified memory address.

7. Source Code & UI

7.1. Source Code Organization

The full source code for HoneyCrisp Emulator is included within this emulator. The code is essentially divided into several different sections:

- **CPU6502 class implementation**
- **Memory management functions**
- **Video display subsystem**
- **ROM loading utilities**
- **User interface event handlers**
- **File I/O operations**

—--

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width,initial-scale=1.0" name="viewport"/>
<!--
HONEYCRISP EMULATOR v1.0

Copyright (c) 2025 Landon James Smith

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE SOFTWARE. -->
```

```
<!--Styles for the body/terminal display-->
<style>
body {
  zoom: 0.80;
}

#terminal {
  font-family: 'AppleI', monospace;
  font-size: 16px;
  line-height: 1em;
  white-space: pre;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  color: white;
  background-color: black;
  width: calc(40ch);
  height: calc(24em);
  padding: 20px;
  border-radius: 12px;
  box-shadow: 0 0 12px rgba(0, 0, 0, 0.6);
  border: 1px solid #333;
}

@font-face {
  font-family: "AppleI";
  src: url("https://landonjsmith.com/hc.ttf") format("truetype");
  font-display: swap;
}
</style>
<title>HoneyCrisp</title>
</head>
<body>
<center>
<div class="container">
<header>
<img src="https://landonjsmith.com/hclogo.png" alt="HONEYCRISP EMULATOR"
width="200" height="300">
</header>
<!--Sets up the emulated hardware controls...These buttons (clearly) map to specific
functions.-->
```

```
<div class="nav">
<!--version number/copyright notice. At top of file as well.-->
    HoneyCrisp 1.0 (92925.327) | © 2025 Landon J. Smith |
    <a href="https://landonjsmith.com/">Home</a> |
    <a href="https://landonjsmith.com/blog.html">My Blog</a> |
    <a href="https://soundcloud.com/therealsjl">My Music</a> |
    <a href="https://youtube.com/@landonjsmith">My Videos</a> |
</div>
<br>
<div class="controls">
<div class="mem-options">
<br>
<label>Set a memory amount:</label>
<input type="radio" name="memsize" value="4" checked> 4KB
<input type="radio" name="memsize" value="8"> 8KB
<br>
</div>

<button class="btn btn-warning" id="btnReset">HARD RESET</button>
<button class="btn btn-primary" id="btnWarmReset">SOFT RESET</button>
<button class="btn" id="btnClear">CLEAR SCREEN</button>
<button class="btn btn-danger" id="btnBreak">BREAK</button>
<br>
<br>
<!--Browser for .hc based program files-->
<label>Load a Program (.hc): </label>
  <input type="file" id="hcLoader" accept=".hc">
</div>
<br>
<div class="status-bar"><div id="ramstatus">RAM: 4KB</div>
<!--Program counter for the emulated 6502...I may or may not keep this for
debugging-->
<div> CPU PC=<span id="pc">0000</span> <span class="indicator" id="indicator"
title="running"></span></div>
<div id="err"></div>
</div>
<pre aria-label="terminal" autofocus="" id="terminal" tabindex="0"></pre>
</center>
<hr>
<div class="legal">
<a href="https://landonjsmith.com/hcfaqs">Frequently Asked Questions</a> | <a
href="https://landonjsmith.com/hcdownload">Download .hc APPLE-1 programs</a>
<!--legal (kinda?) and thanks.-->
```

```
<p>Questions? Comments? <a href="mailto:landon@producerjason.com">Email
Me!</a></p>
<p>All rights to the original Apple-1 software and hardware belong to Apple Inc.</p>
<p>This site and emulator are <strong>not affiliated</strong> with Apple Inc.</p>
<p>WOZMON/Integer BASIC were originally written in 1976 by Steve Wozniak.</p>
<p>Special thanks to <a href="https://sbprojects.net" target="_blank">San
Bergmans</a> and <a href="https://github.com/whscullin/apple1js"
target="_blank">whscullin</a>. <br>
Their documentation on the APPLE-1 was very helpful in the creation of this emulator!
Check them out!</p>
</div>
<hr>
</div>
<script>
// Universal constant for WOZMON state. Helps with several different display oriented
functions.
let wozmonActive = false;
// Beginning of the class for the 6502 CPU emulation
class CPU6502 {
 constructor(ramSizeKB = 4) {
  const selectedMem =
document.querySelector('input[name="memsize"]:checked')?.value || "4";
  this.ram = new Uint8Array(ramSizeKB === 8 ? 0x2000 : 0x1000);
  this.rom = new Uint8Array(0x0400);
  this.basic = new Uint8Array(0x1000);
  this.aci = new Uint8Array(0x0100).fill(0xFF);

  this.A=0; this.X=0; this.Y=0; this.PC=0; this.S=0xFF;
  this.C=0; this.Z=0; this.I=0; this.D=0; this.B=0; this.V=0; this.N=0;
  this.opcode=0; this.cycles=0; this.totalCycles=0; this.pageCrossed=false;
  this.instructionMap={}; this.cycleTable={};
  this.setupInstructions(); this.setupCycleTiming();
  this._kbdBuf = []; this._videoHook = null;
  this._breakFlag = false;
 }
 // Read/Write routines.

 read(addr) {
  addr &= 0xFFFF;
  if (addr === 0xD011) return (this._kbdBuf.length ? 0x80 : 0x00);
  if (addr === 0xD010) return (this._kbdBuf.length ? this._kbdBuf.shift()&0xFF : 0x80);
  if (addr === 0xD012) return 0x00;
  if (addr === 0xD013) return 0x00;
```

```
   if (addr <= (this.ram.length - 1)) return this.ram[addr] & 0xFF;
   if (addr >= 0xC100 && addr <= 0xC1FF) return this.aci[addr - 0xC100] & 0xFF;
   if (addr >= 0xE000 && addr <= 0xEFFF) return this.basic[addr-0xE000] & 0xFF;
   if (addr >= 0xFC00) return this.rom[addr-0xFC00] & 0xFF;
   return 0xFF;
  }

  write(addr, val) {
   addr &= 0xFFFF; val &= 0xFF;

   if (addr === 0xD012) {
    if (typeof this._videoHook === 'function') this._videoHook(val & 0x7F);
    return;
   }

   if (addr <= (this.ram.length - 1)) {
    this.ram[addr] = val;
    return;
   }
  }

  checkPageCross(a,b){ return (a & 0xFF00) !== (b & 0xFF00); }
  push(v){ this.write(0x100 + (this.S & 0xFF), v & 0xFF); this.S = (this.S - 1) & 0xFF; }
  pop(){ this.S = (this.S + 1) & 0xFF; return this.read(0x100 + (this.S & 0xFF)); }
  setZN(v){ v&=0xFF; this.Z = (v===0)?1:0; this.N = (v & 0x80)?1:0; }
  getStatus(){ return
(this.N<<7)|(this.V<<6)|(1<<5)|(this.B<<4)|(this.D<<3)|(this.I<<2)|(this.Z<<1)|this.C; }
  setStatus(v){ this.N=(v>>7)&1; this.V=(v>>6)&1; this.B=(v>>4)&1; this.D=(v>>3)&1;
this.I=(v>>2)&1; this.Z=(v>>1)&1; this.C=v&1; }

  imm(){ return this.read(this.PC++); }
  zp(){ return this.read(this.read(this.PC++)); }
  zpx(){ return this.read((this.read(this.PC++) + this.X) & 0xFF); }
  zpy(){ return this.read((this.read(this.PC++) + this.Y) & 0xFF); }
  abs(){ const lo=this.read(this.PC++),hi=this.read(this.PC++); return
this.read((hi<<8)|lo); }
  absAddr(){ const lo=this.read(this.PC++),hi=this.read(this.PC++); return
((hi<<8)|lo)&0xFFFF; }
  absXAddr(){ const base=this.absAddr(); const a=(base+this.X)&0xFFFF;
this.pageCrossed=this.checkPageCross(base,a); return a; }
  absYAddr(){ const base=this.absAddr(); const a=(base+this.Y)&0xFFFF;
this.pageCrossed=this.checkPageCross(base,a); return a; }
  ind(){ const lo=this.read(this.PC++),hi=this.read(this.PC++); const
```

```
p=((hi<<8)|lo)&0xFFFF; const low=this.read(p); const
high=this.read((p&0xFF00)|((p+1)&0xFF)); return ((high<<8)|low)&0xFFFF; }
 indx(){ const zp=(this.read(this.PC++)+this.X)&0xFF; const lo=this.read(zp&0xFF);
const hi=this.read((zp+1)&0xFF); return ((hi<<8)|lo)&0xFFFF; }
 indy(){ const zp=this.read(this.PC++)&0xFF; const lo=this.read(zp); const
hi=this.read((zp+1)&0xFF); const base=((hi<<8)|lo)&0xFFFF; const
a=(base+this.Y)&0xFFFF; this.pageCrossed=this.checkPageCross(base,a); return a; }

 LDA(v){ this.A=v&0xFF; this.setZN(this.A); } STA(a){ this.write(a,this.A); }
 LDX(v){ this.X=v&0xFF; this.setZN(this.X); } STX(a){ this.write(a,this.X); }
 LDY(v){ this.Y=v&0xFF; this.setZN(this.Y); } STY(a){ this.write(a,this.Y); }
 TAX(){ this.X=this.A&0xFF; this.setZN(this.X); } TXA(){ this.A=this.X&0xFF;
this.setZN(this.A); }
 TAY(){ this.Y=this.A&0xFF; this.setZN(this.Y); } TYA(){ this.A=this.Y&0xFF;
this.setZN(this.A); }
 TSX(){ this.X=this.S&0xFF; this.setZN(this.X);} TXS(){ this.S=this.X&0xFF; }
 PHA(){ this.push(this.A);} PLA(){ this.A=this.pop(); this.setZN(this.A);}
 PHP(){ this.push(this.getStatus()|0x10);} PLP(){ this.setStatus(this.pop());}
 INX(){ this.X=(this.X+1)&0xFF; this.setZN(this.X);} DEX(){ this.X=(this.X-1)&0xFF;
this.setZN(this.X);}
 INY(){ this.Y=(this.Y+1)&0xFF; this.setZN(this.Y);} DEY(){ this.Y=(this.Y-1)&0xFF;
this.setZN(this.Y);}
 CLC(){ this.C=0;} SEC(){ this.C=1;} CLI(){ this.I=0;} SEI(){ this.I=1;}
 CLV(){ this.V=0;} CLD(){ this.D=0;} SED(){ this.D=1;} NOP(){}

 JMP(a){ this.PC=a&0xFFFF; }
 JSR(a){ const ret=(this.PC-1)&0xFFFF; this.push((ret>>8)&0xFF); this.push(ret&0xFF);
this.PC=a&0xFFFF; }
 RTS(){ const lo=this.pop(),hi=this.pop(); this.PC=(((hi<<8)|lo)+1)&0xFFFF; }
 BRK(){ this.B=1; this.PC=(this.PC+1)&0xFFFF; this.push((this.PC>>8)&0xFF);
this.push(this.PC&0xFF); this.push(this.getStatus()|0x10); this.I=1; const
lo=this.read(0xFFFE); const hi=this.read(0xFFFF); this.PC=((hi<<8)|lo)&0xFFFF; }
 RTI(){ this.setStatus(this.pop()); const lo=this.pop(),hi=this.pop();
this.PC=((hi<<8)|lo)&0xFFFF; }

 ADC(v){ v&=0xFF; const sum=this.A+v+(this.C?1:0); this.C=sum>0xFF?1:0;
this.V=((~(this.A^v)&(this.A^sum))&0x80)?1:0; this.A=sum&0xFF; this.setZN(this.A); }
 SBC(v){ v&=0xFF; const inv=(v^0xFF)&0xFF; this.ADC(inv); }
 AND(v){ this.A&=(v&0xFF); this.setZN(this.A); }
 ORA(v){ this.A|=(v&0xFF); this.setZN(this.A); }
 EOR(v){ this.A^=(v&0xFF); this.setZN(this.A); }
 CMP(v){ const r=(this.A-(v&0xFF))&0x1FF; this.C=(r<0x100)?1:0; this.setZN(r&0xFF); }
 CPX(v){ const r=(this.X-(v&0xFF))&0x1FF; this.C=(r<0x100)?1:0; this.setZN(r&0xFF); }
```

```
  CPY(v){ const r=(this.Y-(v&0xFF))&0x1FF; this.C=(r<0x100)?1:0; this.setZN(r&0xFF); }
  BIT(v){ v&=0xFF; this.Z=(this.A&v)?0:1; this.N=(v&0x80)?1:0; this.V=(v&0x40)?1:0; }

  ASL_A(){ this.C=(this.A>>7)&1; this.A=(this.A<<1)&0xFF; this.setZN(this.A); }
  ASL(a){ let v=this.read(a); this.C=(v>>7)&1; v=(v<<1)&0xFF; this.write(a,v);
this.setZN(v); }
  LSR_A(){ this.C=this.A&1; this.A=(this.A>>>1)&0xFF; this.setZN(this.A); }
  LSR(a){ let v=this.read(a); this.C=v&1; v=(v>>>1)&0xFF; this.write(a,v); this.setZN(v); }
  ROL_A(){ const oldC=this.C; this.C=(this.A>>7)&1; this.A=((this.A<<1)|oldC)&0xFF;
this.setZN(this.A); }
  ROL(a){ let v=this.read(a); const oldC=this.C; this.C=(v>>7)&1; v=((v<<1)|oldC)&0xFF;
this.write(a,v); this.setZN(v); }
  ROR_A(){ const oldC=this.C; this.C=this.A&1; this.A=((this.A>>>1)|(oldC<<7))&0xFF;
this.setZN(this.A); }
  ROR(a){ let v=this.read(a); const oldC=this.C; this.C=v&1; v=((v>>>1)|(oldC<<7))&0xFF;
this.write(a,v); this.setZN(v); }

  INC(a){ let v=(this.read(a)+1)&0xFF; this.write(a,v); this.setZN(v); }
  DEC(a){ let v=(this.read(a)-1)&0xFF; this.write(a,v); this.setZN(v); }

  LAX(v){ this.A=this.X=v&0xFF; this.setZN(this.A); }
  SAX(a){ this.write(a,this.A & this.X); }
  DCP(a){ this.DEC(a); this.CMP(this.read(a)); }
  ISC(a){ this.INC(a); this.SBC(this.read(a)); }
  SLO(a){ this.ASL(a); this.ORA(this.read(a)); }
  RLA(a){ this.ROL(a); this.AND(this.read(a)); }
  SRE(a){ this.LSR(a); this.EOR(this.read(a)); }
  RRA(a){ this.ROR(a); this.ADC(this.read(a)); }
  ANC(v){ this.AND(v); this.C=this.N; }
  ALR(v){ this.AND(v); this.LSR_A(); }
  ARR(v){ this.AND(v); this.ROR_A(); this.C=(this.A>>6)&1;
this.V=((this.A>>6)^(this.A>>5))&1; }
  XAA(v){ this.A = this.X & (v & 0xFF); this.setZN(this.A); }
  AXS(v){ const temp=(this.A & this.X)&0xFF; const result=temp-(v&0xFF);
this.X=result&0xFF; this.C=(result>=0)?1:0; this.setZN(this.X); }
  SHY(a){ const val=this.Y & (((a>>>8)+1)&0xFF); this.write(a,val); }
  SHX(a){ const val=this.X & (((a>>>8)+1)&0xFF); this.write(a,val); }
  rel(){ const off=this.read(this.PC++); const
target=(off&0x80)?(this.PC+off-0x100)&0xFFFF:(this.PC+off)&0xFFFF;
this.pageCrossed=this.checkPageCross(this.PC,target); return target; }
  BCC(addr){ if (this.C===0){ this.PC=addr; this.cycles+=1; if (this.pageCrossed)
this.cycles+=1; } }
  BCS(addr){ if (this.C===1){ this.PC=addr; this.cycles+=1; if (this.pageCrossed)
```

```
this.cycles+=1; } }
  BEQ(addr){ if (this.Z===1){ this.PC=addr; this.cycles+=1; if (this.pageCrossed)
this.cycles+=1; } }
  BNE(addr){ if (this.Z===0){ this.PC=addr; this.cycles+=1; if (this.pageCrossed)
this.cycles+=1; } }
  BPL(addr){ if (this.N===0){ this.PC=addr; this.cycles+=1; if (this.pageCrossed)
this.cycles+=1; } }
  BMI(addr){ if (this.N===1){ this.PC=addr; this.cycles+=1; if (this.pageCrossed)
this.cycles+=1; } }
  BVC(addr){ if (this.V===0){ this.PC=addr; this.cycles+=1; if (this.pageCrossed)
this.cycles+=1; } }
  BVS(addr){ if (this.V===1){ this.PC=addr; this.cycles+=1; if (this.pageCrossed)
this.cycles+=1; } }

// Setup for Cycle Timing
  setupCycleTiming(){ this.cycleTable = [
    7,6,2,8,3,3,5,5,3,2,2,2,4,4,6,6,2,5,2,8,4,4,6,6,2,4,2,7,4,4,7,7,
    6,6,2,8,3,3,5,5,4,2,2,2,4,4,6,6,2,5,2,8,4,4,6,6,2,4,2,7,4,4,7,7,
    6,6,2,8,3,3,5,5,3,2,2,2,3,4,6,6,2,5,2,8,4,4,6,6,2,4,2,7,4,4,7,7,
    6,6,2,8,3,3,5,5,4,2,2,2,5,4,6,6,2,5,2,8,4,4,6,6,2,4,2,7,4,4,7,7,
    2,6,2,6,3,3,3,3,2,2,2,2,4,4,4,4,2,6,2,6,4,4,4,4,2,5,2,5,5,5,5,5,
    2,6,2,6,3,3,3,3,2,2,2,2,4,4,4,4,2,5,2,5,4,4,4,4,2,4,2,4,4,4,4,4,
    2,6,2,8,3,3,5,5,2,2,2,2,4,4,6,6,2,5,2,8,4,4,6,6,2,4,2,7,4,4,7,7,
    2,6,2,8,3,3,5,5,2,2,2,2,4,4,6,6,2,5,2,8,4,4,6,6,2,4,2,7,4,4,7,7
  ];}
// It's a big boy of a list, isn't it! @__@

  setupInstructions(){
    for (let i = 0; i < 256; i++) {
      this.instructionMap[i] = () => {
        console.warn(`KIL instruction executed: ${i.toString(16).padStart(2, '0')}`);
        this.PC = (this.PC - 1) & 0xFFFF;
      };
    }
    this.instructionMap[0x00] = () => this.BRK();
    this.instructionMap[0xEA] = () => this.NOP();
    this.instructionMap[0xA9] = () => this.LDA(this.imm());
    this.instructionMap[0xA5] = () => this.LDA(this.zp());
    this.instructionMap[0xB5] = () => this.LDA(this.zpx());
    this.instructionMap[0xAD] = () => this.LDA(this.abs());
    this.instructionMap[0xBD] = () => this.LDA(this.read(this.absXAddr()));
    this.instructionMap[0xB9] = () => this.LDA(this.read(this.absYAddr()));
    this.instructionMap[0xA1] = () => this.LDA(this.read(this.indx()));
```

```
this.instructionMap[0xB1] = () => this.LDA(this.read(this.indy()));
this.instructionMap[0x85] = () => this.STA((this.read(this.PC++) ) & 0xFF);
this.instructionMap[0x95] = () => this.STA((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x8D] = () => this.STA(this.absAddr());
this.instructionMap[0x9D] = () => this.STA(this.absXAddr());
this.instructionMap[0x99] = () => this.STA(this.absYAddr());
this.instructionMap[0x81] = () => this.STA(this.indx());
this.instructionMap[0x91] = () => this.STA(this.indy());
this.instructionMap[0xA2] = () => this.LDX(this.imm());
this.instructionMap[0xA6] = () => this.LDX(this.zp());
this.instructionMap[0xB6] = () => this.LDX(this.zpy());
this.instructionMap[0xAE] = () => this.LDX(this.abs());
this.instructionMap[0xBE] = () => this.LDX(this.read(this.absYAddr()));
this.instructionMap[0xA0] = () => this.LDY(this.imm());
this.instructionMap[0xA4] = () => this.LDY(this.zp());
this.instructionMap[0xB4] = () => this.LDY(this.zpx());
this.instructionMap[0xAC] = () => this.LDY(this.abs());
this.instructionMap[0xBC] = () => this.LDY(this.read(this.absXAddr()));
this.instructionMap[0x86] = () => this.STX(this.read(this.PC++));
this.instructionMap[0x96] = () => this.STX((this.read(this.PC++) + this.Y) & 0xFF);
this.instructionMap[0x8E] = () => this.STX(this.absAddr());
this.instructionMap[0x84] = () => this.STY(this.read(this.PC++));
this.instructionMap[0x94] = () => this.STY((this.read(this.PC++)+ this.X) & 0xFF);
this.instructionMap[0x8C] = () => this.STY(this.absAddr());
this.instructionMap[0xAA] = () => this.TAX();
this.instructionMap[0x8A] = () => this.TXA();
this.instructionMap[0xA8] = () => this.TAY();
this.instructionMap[0x98] = () => this.TYA();
this.instructionMap[0xBA] = () => this.TSX();
this.instructionMap[0x9A] = () => this.TXS();
this.instructionMap[0x48] = () => this.PHA();
this.instructionMap[0x68] = () => this.PLA();
this.instructionMap[0x08] = () => this.PHP();
this.instructionMap[0x28] = () => this.PLP();
this.instructionMap[0xE8] = () => this.INX();
this.instructionMap[0xCA] = () => this.DEX();
this.instructionMap[0xC8] = () => this.INY();
this.instructionMap[0x88] = () => this.DEY();
this.instructionMap[0xE6] = () => this.INC(this.read(this.PC++));
this.instructionMap[0xF6] = () => this.INC((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0xEE] = () => this.INC(this.absAddr());
this.instructionMap[0xFE] = () => this.INC(this.absXAddr());
this.instructionMap[0xC6] = () => this.DEC(this.read(this.PC++));
```

```
this.instructionMap[0xD6] = () => this.DEC((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0xCE] = () => this.DEC(this.absAddr());
this.instructionMap[0xDE] = () => this.DEC(this.absXAddr());
this.instructionMap[0x69] = () => this.ADC(this.imm());
this.instructionMap[0x65] = () => this.ADC(this.zp());
this.instructionMap[0x75] = () => this.ADC(this.zpx());
this.instructionMap[0x6D] = () => this.ADC(this.abs());
this.instructionMap[0x7D] = () => this.ADC(this.read(this.absXAddr()));
this.instructionMap[0x79] = () => this.ADC(this.read(this.absYAddr()));
this.instructionMap[0x61] = () => this.ADC(this.read(this.indx()));
this.instructionMap[0x71] = () => this.ADC(this.read(this.indy()));
this.instructionMap[0xE9] = () => this.SBC(this.imm());
this.instructionMap[0xE5] = () => this.SBC(this.zp());
this.instructionMap[0xF5] = () => this.SBC(this.zpx());
this.instructionMap[0xED] = () => this.SBC(this.abs());
this.instructionMap[0xFD] = () => this.SBC(this.read(this.absXAddr()));
this.instructionMap[0xF9] = () => this.SBC(this.read(this.absYAddr()));
this.instructionMap[0xE1] = () => this.SBC(this.read(this.indx()));
this.instructionMap[0xF1] = () => this.SBC(this.read(this.indy()));
this.instructionMap[0x29] = () => this.AND(this.imm());
this.instructionMap[0x25] = () => this.AND(this.zp());
this.instructionMap[0x35] = () => this.AND(this.zpx());
this.instructionMap[0x2D] = () => this.AND(this.abs());
this.instructionMap[0x3D] = () => this.AND(this.read(this.absXAddr()));
this.instructionMap[0x39] = () => this.AND(this.read(this.absYAddr()));
this.instructionMap[0x21] = () => this.AND(this.read(this.indx()));
this.instructionMap[0x31] = () => this.AND(this.read(this.indy()));
this.instructionMap[0x09] = () => this.ORA(this.imm());
this.instructionMap[0x05] = () => this.ORA(this.zp());
this.instructionMap[0x15] = () => this.ORA(this.zpx());
this.instructionMap[0x0D] = () => this.ORA(this.abs());
this.instructionMap[0x1D] = () => this.ORA(this.read(this.absXAddr()));
this.instructionMap[0x19] = () => this.ORA(this.read(this.absYAddr()));
this.instructionMap[0x01] = () => this.ORA(this.read(this.indx()));
this.instructionMap[0x11] = () => this.ORA(this.read(this.indy()));
this.instructionMap[0x49] = () => this.EOR(this.imm());
this.instructionMap[0x45] = () => this.EOR(this.zp());
this.instructionMap[0x55] = () => this.EOR(this.zpx());
this.instructionMap[0x4D] = () => this.EOR(this.abs());
this.instructionMap[0x5D] = () => this.EOR(this.read(this.absXAddr()));
this.instructionMap[0x59] = () => this.EOR(this.read(this.absYAddr()));
this.instructionMap[0x41] = () => this.EOR(this.read(this.indx()));
this.instructionMap[0x51] = () => this.EOR(this.read(this.indy()));
```

```
this.instructionMap[0xC9] = () => this.CMP(this.imm());
this.instructionMap[0xC5] = () => this.CMP(this.zp());
this.instructionMap[0xD5] = () => this.CMP(this.zpx());
this.instructionMap[0xCD] = () => this.CMP(this.abs());
this.instructionMap[0xDD] = () => this.CMP(this.read(this.absXAddr()));
this.instructionMap[0xD9] = () => this.CMP(this.read(this.absYAddr()));
this.instructionMap[0xC1] = () => this.CMP(this.read(this.indx()));
this.instructionMap[0xD1] = () => this.CMP(this.read(this.indy()));
this.instructionMap[0xE0] = () => this.CPX(this.imm());
this.instructionMap[0xE4] = () => this.CPX(this.zp());
this.instructionMap[0xEC] = () => this.CPX(this.abs());
this.instructionMap[0xC0] = () => this.CPY(this.imm());
this.instructionMap[0xC4] = () => this.CPY(this.zp());
this.instructionMap[0xCC] = () => this.CPY(this.abs());
this.instructionMap[0x24] = () => this.BIT(this.zp());
this.instructionMap[0x2C] = () => this.BIT(this.abs());
this.instructionMap[0x0A] = () => this.ASL_A();
this.instructionMap[0x06] = () => this.ASL(this.read(this.PC++));
this.instructionMap[0x16] = () => this.ASL((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x0E] = () => this.ASL(this.absAddr());
this.instructionMap[0x1E] = () => this.ASL(this.absXAddr());
this.instructionMap[0x4A] = () => this.LSR_A();
this.instructionMap[0x46] = () => this.LSR(this.read(this.PC++));
this.instructionMap[0x56] = () => this.LSR((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x4E] = () => this.LSR(this.absAddr());
this.instructionMap[0x5E] = () => this.LSR(this.absXAddr());
this.instructionMap[0x2A] = () => this.ROL_A();
this.instructionMap[0x26] = () => this.ROL(this.read(this.PC++));
this.instructionMap[0x36] = () => this.ROL((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x2E] = () => this.ROL(this.absAddr());
this.instructionMap[0x3E] = () => this.ROL(this.absXAddr());
this.instructionMap[0x6A] = () => this.ROR_A();
this.instructionMap[0x66] = () => this.ROR(this.read(this.PC++));
this.instructionMap[0x76] = () => this.ROR((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x6E] = () => this.ROR(this.absAddr());
this.instructionMap[0x7E] = () => this.ROR(this.absXAddr());
this.instructionMap[0x90] = () => this.BCC(this.rel());
this.instructionMap[0xB0] = () => this.BCS(this.rel());
this.instructionMap[0xF0] = () => this.BEQ(this.rel());
this.instructionMap[0x30] = () => this.BMI(this.rel());
this.instructionMap[0xD0] = () => this.BNE(this.rel());
this.instructionMap[0x10] = () => this.BPL(this.rel());
this.instructionMap[0x50] = () => this.BVC(this.rel());
```

```
this.instructionMap[0x70] = () => this.BVS(this.rel());
this.instructionMap[0x4C] = () => this.JMP(this.absAddr());
this.instructionMap[0x6C] = () => this.JMP(this.ind());
this.instructionMap[0x20] = () => this.JSR(this.absAddr());
this.instructionMap[0x60] = () => this.RTS();
this.instructionMap[0x40] = () => this.RTI();
this.instructionMap[0x18] = () => this.CLC();
this.instructionMap[0x38] = () => this.SEC();
this.instructionMap[0x58] = () => this.CLI();
this.instructionMap[0x78] = () => this.SEI();
this.instructionMap[0xB8] = () => this.CLV();
this.instructionMap[0xD8] = () => this.CLD();
this.instructionMap[0xF8] = () => this.SED();
this.instructionMap[0xA7] = () => this.LAX(this.zp());
this.instructionMap[0xB7] = () => this.LAX(this.zpy());
this.instructionMap[0xAF] = () => this.LAX(this.abs());
this.instructionMap[0xBF] = () => this.LAX(this.read(this.absYAddr()));
this.instructionMap[0xA3] = () => this.LAX(this.read(this.indx()));
this.instructionMap[0xB3] = () => this.LAX(this.read(this.indy()));
this.instructionMap[0x87] = () => this.SAX(this.read(this.PC++));
this.instructionMap[0x97] = () => this.SAX((this.read(this.PC++) + this.Y) & 0xFF);
this.instructionMap[0x8F] = () => this.SAX(this.absAddr());
this.instructionMap[0x83] = () => this.SAX(this.indx());
this.instructionMap[0xC7] = () => this.DCP(this.read(this.PC++));
this.instructionMap[0xD7] = () => this.DCP((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0xCF] = () => this.DCP(this.absAddr());
this.instructionMap[0xDF] = () => this.DCP(this.absXAddr());
this.instructionMap[0xDB] = () => this.DCP(this.absYAddr());
this.instructionMap[0xC3] = () => this.DCP(this.indx());
this.instructionMap[0xD3] = () => this.DCP(this.indy());
this.instructionMap[0xE7] = () => this.ISC(this.read(this.PC++));
this.instructionMap[0xF7] = () => this.ISC((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0xEF] = () => this.ISC(this.absAddr());
this.instructionMap[0xFF] = () => this.ISC(this.absXAddr());
this.instructionMap[0xFB] = () => this.ISC(this.absYAddr());
this.instructionMap[0xE3] = () => this.ISC(this.indx());
this.instructionMap[0xF3] = () => this.ISC(this.indy());
this.instructionMap[0x07] = () => this.SLO(this.read(this.PC++));
this.instructionMap[0x17] = () => this.SLO((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x0F] = () => this.SLO(this.absAddr());
this.instructionMap[0x1F] = () => this.SLO(this.absXAddr());
this.instructionMap[0x1B] = () => this.SLO(this.absYAddr());
this.instructionMap[0x03] = () => this.SLO(this.indx());
```

```
this.instructionMap[0x13] = () => this.SLO(this.indy());
this.instructionMap[0x27] = () => this.RLA(this.read(this.PC++));
this.instructionMap[0x37] = () => this.RLA((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x2F] = () => this.RLA(this.absAddr());
this.instructionMap[0x3F] = () => this.RLA(this.absXAddr());
this.instructionMap[0x3B] = () => this.RLA(this.absYAddr());
this.instructionMap[0x23] = () => this.RLA(this.indx());
this.instructionMap[0x33] = () => this.RLA(this.indy());
this.instructionMap[0x47] = () => this.SRE(this.read(this.PC++));
this.instructionMap[0x57] = () => this.SRE((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x4F] = () => this.SRE(this.absAddr());
this.instructionMap[0x5F] = () => this.SRE(this.absXAddr());
this.instructionMap[0x5B] = () => this.SRE(this.absYAddr());
this.instructionMap[0x43] = () => this.SRE(this.indx());
this.instructionMap[0x53] = () => this.SRE(this.indy());
this.instructionMap[0x67] = () => this.RRA(this.read(this.PC++));
this.instructionMap[0x77] = () => this.RRA((this.read(this.PC++) + this.X) & 0xFF);
this.instructionMap[0x6F] = () => this.RRA(this.absAddr());
this.instructionMap[0x7F] = () => this.RRA(this.absXAddr());
this.instructionMap[0x7B] = () => this.RRA(this.absYAddr());
this.instructionMap[0x63] = () => this.RRA(this.indx());
this.instructionMap[0x73] = () => this.RRA(this.indy());
this.instructionMap[0x0B] = () => this.ANC(this.imm());
this.instructionMap[0x2B] = () => this.ANC(this.imm());
this.instructionMap[0x4B] = () => this.ALR(this.imm());
this.instructionMap[0x6B] = () => this.ARR(this.imm());
this.instructionMap[0x8B] = () => this.XAA(this.imm());
this.instructionMap[0xCB] = () => this.AXS(this.imm());
this.instructionMap[0x9C] = () => this.SHY(this.absXAddr());
this.instructionMap[0x9E] = () => this.SHX(this.absYAddr());
this.instructionMap[0x1A] = () => this.NOP();
this.instructionMap[0x3A] = () => this.NOP();
this.instructionMap[0x5A] = () => this.NOP();
this.instructionMap[0x7A] = () => this.NOP();
this.instructionMap[0xDA] = () => this.NOP();
this.instructionMap[0xFA] = () => this.NOP();
this.instructionMap[0x04] = () => { this.PC++; this.NOP(); };
this.instructionMap[0x44] = () => { this.PC++; this.NOP(); };
this.instructionMap[0x64] = () => { this.PC++; this.NOP(); };
this.instructionMap[0x14] = () => { this.PC++; this.NOP(); };
this.instructionMap[0x34] = () => { this.PC++; this.NOP(); };
this.instructionMap[0x54] = () => { this.PC++; this.NOP(); };
this.instructionMap[0x74] = () => { this.PC++; this.NOP(); };
```

```
    this.instructionMap[0xD4] = () => { this.PC++; this.NOP(); };
    this.instructionMap[0xF4] = () => { this.PC++; this.NOP(); };
    this.instructionMap[0x0C] = () => { this.PC += 2; this.NOP(); };
    this.instructionMap[0x1C] = () => { this.absXAddr(); this.NOP(); };
    this.instructionMap[0x3C] = () => { this.absXAddr(); this.NOP(); };
    this.instructionMap[0x5C] = () => { this.absXAddr(); this.NOP(); };
    this.instructionMap[0x7C] = () => { this.absXAddr(); this.NOP(); };
    this.instructionMap[0xDC] = () => { this.absXAddr(); this.NOP(); };
    this.instructionMap[0xFC] = () => { this.absXAddr(); this.NOP(); };
    this.instructionMap[0x80] = () => { this.PC++; this.NOP(); };
    this.instructionMap[0x82] = () => { this.PC++; this.NOP(); };
    this.instructionMap[0x89] = () => { this.PC++; this.NOP(); };
    this.instructionMap[0xC2] = () => { this.PC++; this.NOP(); };
    this.instructionMap[0xE2] = () => { this.PC++; this.NOP(); };
  }
// END SETUP TABLE.
  checkBreak() {
    if (this._breakFlag) {
      this._breakFlag = false;

      if (this.PC >= 0xE000 && this.PC <= 0xEFFF) {
        this.S = 0xFF;

        videoOutputQueue = [];

        this._videoHook(0x0D);

        this.PC = 0xE8C3;

        this.I = 1;
        return true;
      } else {
        return false;
      }
    }
    return false;
  }
// hard-reset/power-cycle emulation. Resets CPU and rest of system back to the bootup
state.
  reset(){
    this.A = this.X = this.Y = 0;
    this.S = 0xFF;
    this.C = this.Z = this.I = this.D = this.B = this.V = this.N = 0;
```

```
    this.I = 1; this.B = 0;
    const lo = this.read(0xFFFC); const hi = this.read(0xFFFD);
    this.PC = ((hi << 8) | lo) & 0xFFFF;
    this.cycles = 7; this.totalCycles += 7;
    this._breakFlag = false;
  }
// Soft-reset. Jumps back to WOZMON prompt at $FF00
  warmReset() {
    this.PC = 0xFF00;
    this.I = 1;
    this.cycles = 7;
    this.totalCycles += 7;
    this._breakFlag = false;
  }
// Break button trigger for Integer BASIC at $E000-$EFFF
  triggerBreak() {
    this._breakFlag = true;
  }

// Crash detection. If the program counter returns 0x0000, then throw crash error.
// HALTS THE SYSTEM.

  step(){
    if (this.PC === 0x0000 && this.read(this.PC) === 0x00) {
      console.error("Crash detected: Executing BRK at $0000. Resetting.");
      throw new Error("CPU_CRASH_0000");
    }

    if (this.checkBreak()) {
      return this.cycles;
    }

    this.opcode = this.read(this.PC++);
    this.pageCrossed = false;
    this.cycles = this.cycleTable[this.opcode] || 2;
    const instr = this.instructionMap[this.opcode & 0xFF];
    if (instr) instr(); else console.warn(`Unimplemented opcode:
${this.opcode.toString(16)}`);
    if (this.pageCrossed && this.needsPageCrossPenalty(this.opcode) ) this.cycles += 1;
    this.totalCycles += this.cycles;
    return this.cycles;
  }
```

```
  needsPageCrossPenalty(op){ const s=new Set([
    0x1D,0x19,0x39,0x3D,0x59,0x5D,0x79,0x7D, 0xB1,0xB9,0xBD,0xBE,0xBC,
    0xD1,0xD9,0xDD,0xF1,0xF9,0xFD, 0x1C,0x3C,0x5C,0x7C,0xDC,0xFC, 0xBF,0xB3,
    0x1B,0x3B,0x5B,0x7B,0xDB,0xFB, 0xDF,0xFF,0x9C,0x9E
  ]); return s.has(op); }

  runCycles(target){
    const start = this.totalCycles;
    while ((this.totalCycles - start) < target) this.step();
    return this.totalCycles - start;
  }
  getState(){ return {
    A:this.A.toString(16).padStart(2,'0'), X:this.X.toString(16).padStart(2,'0'),
    Y:this.Y.toString(16).padStart(2,'0'), PC:this.PC.toString(16).padStart(4,'0'),
    S:this.S.toString(16).padStart(2,'0'), P:this.getStatus().toString(16).padStart(2,'0'),
    cycles:this.cycles, totalCycles:this.totalCycles,
    flags:{C:this.C,Z:this.Z,I:this.I,D:this.D,B:this.B,V:this.V,N:this.N}
  };}
}
const ROWS = 24, COLS = 40;
let screen = Array.from({length:ROWS}, () => Array(COLS).fill(' '));
let cursor = {x:0,y:0};

let userTyped = Array.from({length: ROWS}, () => Array(COLS).fill(false));
let pendingUserAt = false;
let showPromptCursor = false;

const termEl = document.getElementById('terminal');
const pcEl = document.getElementById('pc');
const errEl = document.getElementById('err');
const indicatorEl = document.getElementById('indicator');
// Sets blink interval for the WOZMON @ cursor.
let blinkState = true;
setInterval(() => { blinkState = !blinkState; renderScreen(); }, 530);

// Screen rendering routines for the terminal display.
function renderScreen(){
  let display = screen.map(row => row.slice());
  for (let y = 0; y < ROWS; y++) {
    for (let x = 0; x < COLS; x++) {
      if (display[y][x] === "@" && !userTyped[y][x]) {
        if (!blinkState && (x > 0 && display[y][x-1] === "_" || (cursor.y === y && cursor.x ===
x))) {
```

```
        display[y][x] = " ";
      }
     }
    }
   }
  if ((wozmonActive && document.activeElement === termEl) || showPromptCursor) {
   if (cursor.y < ROWS && cursor.x < COLS) {
    display[cursor.y][cursor.x] = blinkState ? '@' : screen[cursor.y][cursor.x];
   }
  }

  termEl.textContent = display.map(row => row.join('')).join('\n');

  if (termEl.scrollHeight > termEl.clientHeight) {
    const lineHeight = termEl.clientHeight / ROWS;
    const desiredScrollTop = Math.max(0, (cursor.y + 1) * lineHeight -
termEl.clientHeight);
    if (Math.abs(termEl.scrollTop - desiredScrollTop) > 1) {
      termEl.scrollTop = desiredScrollTop;
    }
  }

  const rect = termEl.getBoundingClientRect();
  const lineHeightPx = rect.height / ROWS;
  const cursorLineY = rect.top + (cursor.y + 0.5) * lineHeightPx;

  if (cursorLineY < 0 || cursorLineY > window.innerHeight) {
    const target = window.scrollY + (cursorLineY - window.innerHeight / 2);
    const clamped = Math.max(0, Math.min(document.documentElement.scrollHeight -
window.innerHeight, target));
    window.scrollTo({ top: clamped, behavior: 'smooth' });
  }
}

// Screen scrolls with displayed content like a teletype!!
function scroll() {
    screen.shift();
    screen.push(Array(COLS).fill(' '));
    cursor.y = ROWS - 1;
}
// Detects if a new line is to be created.
function newline() {
    cursor.x = 0;
```

```
    cursor.y++;
    if (cursor.y >= ROWS) {
      scroll();
    }
}
// Character printing routines
function printChar(ch) {
    if (ch === '\n') {
      newline();
      return;
    }

    let c = String.fromCharCode(ch & 0x7F);
// Patch for the font used for the 2513 ROM recreation. Basically patches it to only have
APPLE-1 charset
    const apple1Charset = /^[@A-Z\[\\\]\^_ !\"#$%&'()*+,\-\.\/0-9:;<=>?]$/;

    if (apple1Charset.test(c)) {
    if (pendingUserAt && c === '@') {
      userTyped[cursor.y][cursor.x] = true;
      pendingUserAt = false;
    } else {
      userTyped[cursor.y][cursor.x] = false;
    }

screen[cursor.y][cursor.x] = c;
      cursor.x++;
    }

    if (cursor.x >= COLS) {
      newline();
    }
}



// Initializes the garbage video RAM pattern at startup and writes it to RAM.
function initVideoPattern() {
  screen = Array.from({ length: ROWS }, (_, row) =>
    Array.from({ length: COLS }, (_, col) =>
      col % 2 === 0 ? '_' : '@'
    )
  );
  cursor = { x: 0, y: 0 };
```

```
  userTyped = Array.from({length: ROWS}, () => Array(COLS).fill(false));
  showPromptCursor = false;
  renderScreen();
}

// Function to clear screen of all characters. Will leave blinking @ for WOZMON input if
applicable.
function clearScreen() {
  screen = Array.from({ length: ROWS }, () => Array(COLS).fill(' '));
  cursor = { x: 0, y: 0 };
  userTyped = Array.from({length: ROWS}, () => Array(COLS).fill(false));

  showPromptCursor = true;
  // re-renders everything
  renderScreen();
}

// converts base64 strings to readable machine bytes.
function b64ToBytes(base64) {
  const binString = atob(base64);
  const len = binString.length;
  const bytes = new Uint8Array(len);
  for (let i = 0; i < len; i++) {
    bytes[i] = binString.charCodeAt(i);
  }
  return bytes;
}
// Creates a "new" CPU for each time the HoneyCrisp session is initialized.
let cpu = null;
function initCPU() {
  const selectedMem =
document.querySelector('input[name="memsize"]:checked')?.value || "4";
  cpu = new CPU6502(parseInt(selectedMem));
  console.log(`Initialized HoneyCrisp with ${selectedMem}KB of RAM`);
  const ramStatusEl = document.getElementById('ramstatus');
  if (ramStatusEl) ramStatusEl.textContent = `RAM: ${selectedMem}KB`;
  cpu._videoHook = (byte7) => {
    videoOutputQueue.push(byte7 & 0x7F);
  };
}
initCPU();
const _ACIRomBytes = new
Uint8Array([0xA9,0xAA,0x20,0xEF,0xFF,0xA9,0x8D,0x20,0xEF,0xFF,0xA0,0xFF,0xC8,0x
```

```
AD,0x11,0xD0,0x10,0xFB,0xAD,0x10,0xD0,0x99,0x00,0x02,0x20,0xEF,0xFF,0xC9,0x9B,
0xF0,0xE1,0xC9,0x8D,0xD0,0xE9,0xA2,0xFF,0xA9,0x00,0x85,0x24,0x85,0x25,0x85,0x2
6,0x85,0x27,0xE8,0xBD,0x00,0x02,0xC9,0xD2,0xF0,0x56,0xC9,0xD7,0xF0,0x35,0xC9,0
xAE,0xF0,0x27,0xC9,0x8D,0xF0,0x20,0xC9,0xA0,0xF0,0xE8,0x49,0xB0,0xC9,0x0A,0x9
0,0x06,0x69,0x88,0xC9,0xFA,0x90,0xAD,0x0A,0x0A,0x0A,0x0A,0xA0,0x04,0x0A,0x2
6,0x24,0x26,0x25,0x88,0xD0,0xF8,0xF0,0xCC,0x4C,0x1A,0xFF,0xA5,0x24,0x85,0x26,0
xA5,0x25,0x85,0x27,0xB0,0xBF,0xA9,0x40,0x20,0xCC,0xC1,0x88,0xA2,0x00,0xA1,0x26
,0xA2,0x10,0x0A,0x20,0xDB,0xC1,0xD0,0xFA,0x20,0xF1,0xC1,0xA0,0x1E,0x90,0xEC,0
xA6,0x28,0xB0,0x98,0x20,0xBC,0xC1,0xA9,0x16,0x20,0xCC,0xC1,0x20,0xBC,0xC1,0xA
0,0x1F,0x20,0xBF,0xC1,0xB0,0xF9,0x20,0xBF,0xC1,0xA0,0x3A,0xA2,0x08,0x48,0x20,0
xBC,0xC1,0x68,0x2A,0xA0,0x39,0xCA,0xD0,0xF5,0x81,0x26,0x20,0xF1,0xC1,0xA0,0x3
5,0x90,0xEA,0xB0,0xCD,0x20,0xBF,0xC1,0x88,0xAD,0x81,0xC0,0xC5,0x29,0xF0,0xF8,
0x85,0x29,0xC0,0x80,0x60,0x86,0x28,0xA0,0x42,0x20,0xE0,0xC1,0xD0,0xF9,0x69,0x
FE,0xB0,0xF5,0xA0,0x1E,0x20,0xE0,0xC1,0xA0,0x2C,0x88,0xD0,0xFD,0x90,0x05,0xA
0,0x2F,0x88,0xD0,0xFD,0xBC,0x00,0xC0,0xA0,0x29,0xCA,0x60,0xA5,0x26,0xC5,0x24
,0xA5,0x27,0xE5,0x25,0xE6,0x26,0xD0,0x02,0xE6,0x27,0x60]);
if (typeof loadACIRom === 'function') loadACIRom(_ACIRomBytes);
if (window._pendingACI && cpu && cpu.aci) { cpu.aci.set(window._pendingACI);
window._pendingACI = null; console.log('Applied pending ACI ROM'); }
let videoOutputQueue = [];
// Character output delay to make sure it's not getting ahead of the what the CPU trying
to keep up with.
const CHAR_OUTPUT_DELAY = 1000 / 30;
let lastCharOutputTime = 0;
let running = true;

cpu._videoHook = (byte7) => {
  videoOutputQueue.push(byte7 & 0x7F);
};

function enqueueKey(ascii){
  const upper = String.fromCharCode(ascii).toUpperCase().charCodeAt(0) & 0x7F;
  cpu._kbdBuf.push(upper | 0x80);
}

document.addEventListener("keydown", (e) => {
    if (!wozmonActive) return;
    keyBuffer.push(e.key);
   });


termEl.addEventListener('keydown', (e) => {
    if (!wozmonActive) {
```

```
      e.preventDefault();
      return;
    }

    if (['Backspace','ArrowLeft','ArrowRight','ArrowUp','ArrowDown','Tab'].includes(e.key))
e.preventDefault();
    if (e.ctrlKey && e.key.toUpperCase() === 'C') {
      cpu.triggerBreak();
      e.preventDefault();
    } else if (e.key.length === 1) {
  if (e.key === '@') pendingUserAt = true;
enqueueKey(e.key.charCodeAt(0));
    } else if (e.key === 'Enter') {
      cpu._kbdBuf.push(0x8D);
    } else if (e.key === 'Backspace') {
      cpu._kbdBuf.push(0xDF);
    } else if (e.key === 'Escape') {
      cpu._kbdBuf.push(0x9B);
      e.preventDefault();
    }
  });
function tick(timestamp){
  if (running) {
    try {
      cpu.runCycles(16667);
      indicatorEl.classList.add('running');
    } catch (e) {
      if (e.message === "CPU_CRASH_0000") {
        hardReset();
      } else {
        console.error("An unexpected error occurred:", e);
        running = false;
      }
    }


    if (videoOutputQueue.length > 0 && (timestamp - lastCharOutputTime >
CHAR_OUTPUT_DELAY)) {
      const ch = videoOutputQueue.shift();
      if (ch === 0x0D) {
        printChar('\n');
      } else {
        printChar(ch);
```

```
    }
    lastCharOutputTime = timestamp;
   }

   pcEl.textContent = cpu.PC.toString(16).padStart(4,'0').toUpperCase();
   renderScreen();
  }
  requestAnimationFrame(tick);
}

// ROM HANDLING

let WOZMON_BASE64 =
"2Figf4wS0KmnjRHQjRPQyd/wE8mb8APIEA+p3CDv/6mNIO//oAGIMPatEdAQ+60Q0JkA
AiDv/8mN0NSg/6kAqgqFK8i5AALJjfDUya6Q9PDwybrw68nS8DuGKIYphCq5AAJJsMkKk
AZpiMn6kBEKCgoKogQKJigmKcrQ+MjQ4MQq8JckK1AQpSiBJuYm0LXmJ0xE/2wkADAr
ogK1J5UllSPK0PfQFKmNIO//pSUg3P+lJCDc/6m6IO//qaAg7/+hJCDc/4YrpSTFKKUl5Smw
weYk0ALmJaUkKQcQyEhKSkpKIOX/aCkPCbDJupACaQYsEtAw+40S0GAAAAAPAP8AAA
==";


let INTBASIC_BASE64 =
"TLDirRHQEPutENBgiikg8COpoIXkTMnjqSDFJLAMqY2gByDJ46mgiND4oACx4ubi0ALm
42AgFecgduWl4sXmpePl57DvIG3gTDvgpcqF4qXLheOlTIXmpU2F59DeIBXnIG3lpeSF4q
XlheOwx4bYqaCF+iAq4JiF5CAq4KogKuAgG+UgGOCE+qoQGAoQ6aXk0AMgEeCKIMnjq
SUgGuCqMPWF5MkB0AWm2EzN40iEzqLths/JUZAExs/pUEixzqqIsc4Q+uDAsATgADDyq
mjpAdDpJOQwAyD477HOEBCqKT+F5BhpoCDJ44jgwJDsIAzgaMld8KTJKNCK8J4gGOG
VUNV4kBGgK0zg4yA07tVQkPQg5O+VeEwj6CA07vDnOOkBYCAY4ZVQGPV4TALhoBTQ1i
AY4ei1UIXaZc5IqLV4hdtlz0jEyuXLsOOl2mn+hdqp/6hl24XbyLHa2cwA0A+Y8PVokdqZzA
CIEPfoYOqggNCVqQAgCuegApR4IArnqb8gyeOgACCe4pR46urqtVGFzrV5hc/o6CC84bV
O1XawFfZOqLHOtFDE5JAEoIPQwZHa9lCQ5bRQipHa6OhgtVGF2jjpAoXktXmF2+kAheW
gALHkGOXaheRgtVOFzrV7hc+1UYXatXmF2+jo6KAAlHiUoMiUULVN1XUISLVP1XeQB2gos
AJWUGCosc6F5GioKLDzsdrF5NDt9k/2TbDXINfhTDbnIFTiBs4mz5ANGKXmZdqF5qXnZd
uF54jwCQbmJucQ5Ex+56XmIAjnpeeVoAblkChMb+epVYXlIFvipc6F2qXPhdsgFeeE5oTn
pc8QCcoG5SBv5yAV56AQYCBs7vDF/8mE0AJG+Mnf8BHJm/AGmQACyBAKoIsgxOOgAY
gw9iAD4OrqIMnjyY3Q1qnfmQACYCDT7yDN40bZqb4gyeOgAIT6JPgQDKb2pfcgG+Wpo
CDJ46L/miCe4oTxioXIoiAgkeSlyGkAheCpAKppAoXhoeAp8Mmw8ANMg+igArHgmc0Ai
ND4IIrjpfHlyMkE8KiR4KXK8eCF5KXL6QCF5aXkxcyl5eXNkEWlyvHghealy+kAheexypHm
5srQAubLpeLFyqXj5cuw4LXklcrKEPmx4KiIseCR5pjQ+CT4EAm193X1lffo8PcQfgAAAACg
FNBxIBXnpeKF5qXjhecgdeWl4oXkpeOF5dAOIBXnIG3lpeaF4qXnheOgAKXKxeSly+XlsBa
l5NACxuXG5KXm0ALG58bmseSR5pDgpeaFyqXnhctgIMnjyLkA6zD3yY3QBqkAhSSpjeYk
LBLQMPuNEtBgoAYg0+4k2TADTLbiTJrrKmmg3QAC0FOx/gowBoix/jApyIbImEiiAKH+qkp
JSBH+ycCQAejI0PNoqIpMwOTm8abx8LydAAJgpsipoOjdAAKw+rH+KT9K0La9AAKwBmk
```

/yRqQb2lPyQqQaab9yLH+KeDJIPB6taiFyLXRhfGIsf4KEPqIsDgKMDW0WIT/tIDoENrws8l
+sCLKEASgBhAplICk/5RYpMiUqKTxlNEpH6i5IOwKqKl2KoX/0AHIyIb9sf4whNAFoA5M4O
PJA7DDSqbI6L0AApAEyaLwCsnf8AaGyCAc5MiIpv2x/ogKEM+0WIT/tIDosf4pn9DthfKF8
5hIhv200ITJGKkKhfmiAMi5AAIpD2XySIpl8zAcqmjG+dDyhfKG88Tx0N6kyciE8SAc5Giopf
OwqaAAEIuF84byogSGyamwhfml8t1j5aXz/WjlkA2F86Xy/WPlhfLm+dDnpfnoyvAOybDw
AoXJJMkwBKX68AsgyeMk+BAEmQACyMoQwWABCmToEAAAAAMnpcqF5qXLhefopeeF
5aXmheTFTKXl5U2wJqABseTlzsix5OXPsBmgAKXmceSF5pAD5ucYyKXO8eTIpc/x5LDKY
Eb4pUyFyqVNhculSoXMpUuFzakAhfuF/IX+qQCFHWCl0GkFhdKl0WkAhdOl0sXKpdPly5
ADTGvjpc6R0KXPyJHQpdLIkdCl08iR0KkAyJHQyJHQpdKFzKXThc2l0JBDhc6EzyD/5jAO
yUDwCkwo5gbJSdAHqUmFzyD/5qVLhdGlSoXQxcyl0eXNsJSx0MjFztAGsdDFz/AOyLHQ
SMix0IXRaKAA8Nul0GkDIArnpdFpAJV4pc/JQNAciJggCueIlHigA/Z4yLHQMPkQCakAhd
SF1aIgSKAAseAQGAowgSD/5iAI5yD/5pWgJNQQAcog/+aw5sko0B+l4CAK56XhlXgk1DAL
qQEgCuepAJV49ngg/+Yw+bDTJNQQBskEsNBG1KiF1rmY6SlVCoXXaKi5mOkpqsXXsAmY
SCD/5qXWkJW5EOqFzrmI6oXPIPzmTNjmbM4A5uDQAubhseBglHfKMAOVUGCgZkzg46
AAtVCFzrWghc+1ePAOhc+xzkjIsc6Fz2iFzojoYCBK5yAV55ggCOeVoMXO0AbFz9AC9lBgII
LnIFnnIBXnJM8wG8pgIBXnpc/QBKXO8POp/yAI55WgJM8w6SAV55g45c4gCOeY5c9QI6
AAEJAgb+cgFeelzoXapc+F2yAV5xilzmXaIAjnpc9l23DdlaBgIBXnpM7wBYilz/AMYKUkCQe
oyKmgIMnjxCSw92AgsecgFeelzxAKqa0gyeMgcudQ74iE1YbPps4gG+Wmz2AgFeelzoX2p
c+F94iE+MipCoX0hPVgIBXnpc6kzxDyIBXntVCF2rV4hdulzpHayKXPkdroYGhoJNUQBSD
N40bVYKD/hNdgIM3v8AepJYXWiITU6GClyqTL0FqgQaX8yQiwXqjm/KXgmQABpeGZCA
Gl3JkQAaXdmRgBIBXnIG3lkASgN9A7peSk5YXchN0sEdAwTxhpA5AByKL/htmaheCE4SB
55iTZEEkYoACl3HHcpN2QAcjFTNDRxE3QzaA0RtlM4OOgSqX88PfG/k
i5DwGF3LkXAYXdvv8AuQcBqIpMeuigYyDE46ABsdyqyLHcIBvlTLPixvugW6X78MSotVDZ
HwHQ8LV42ScB0Om5LwGF2rk3AYXbIBXnyiCT5yAB6Mqk+7lnAZWfuV8BoAAgCOcggucg
WecgFeek+6XO8AVZNwEQErk/AYXcuUcBhd2+TwG5VwHQh8b7YKBUpfvJCPCa5vuotV
CZIAG1eJkoAWAgFeek+6XOmV8Bpc+ZZwGpAZkvAakAmTcBpdyZPwGl3ZlHAaXgmU8Bp
eGZVwFgIBXnpPulzpkvAaXPTGbpAAAAAAAAAAAAAAAAAACrAwMDAwMDAwMDAwM
DAwM/P8DAPDw8PDw8PDAPwMz/VQCrqwMD//9V//9Vz8/Pz8//VcPDw1Xw8M9WVlZV//9
VAwMDAwMDA////wMDAwMDAwMDAwMDAwMDAwMAqwNXAwMDAwcDAwMDAwMDA
wMDAwOq//////8X//8ZXTVL8uyHb6234vhUgJaFgiIQM0oTBgtKAUBHegD/IwlbFrbL///7//8
k9k5ZUAD/I6NvNiPXHCLCrroj//8hMB4DxCAAwf///6AwHqTTtryqOgFQftjYpTz/FlsoA8Qd
AAxOAD4AprAAvMZXjAEn///////o///o4ODg7+/j4+Xl5+fu7+/n5+Lv5+fs7Ozn7Ozs4gD/6OH
o6O/r///g///v7u/n5wD/6Ofn5+jh4u7u7u7o///h4e/u5+ju5////+7h7+fo7+/r6ejp6ejo6Oj/6Ojo7
ufo7+/u7+7v7u7v7u7u4ejo//////++s7K3tjfUz8+gzM/OR9PZztTBWM3FzaDG1cxM1M/PoM3
Bztmg0MHSxc5T09TSyc5Hzs+gxc5EwsHEoMLSwc7DSL64oMfP09XCU8LBxKDSxdTV0k
6+uKDGz9JTwsHEoM7F2FTT1M/Q0MXEoMHUIKqqqiCgxdLSDb6ytTXSwc7HRcTJTdPU0
qDP1sZM3A3SxdTZ0MWgzMnOxY0/RtmQA0zD6KbPmqbOoI3QAqCZIMTjhs66hs+g/oTZ
yITIIJnihPGiIKkwIJHk5tmmzqTICoXOyLkAAsl08NJJsMkKsPDIyITIuQACSLn/AaAAIAjnaJ
Wgpc7Jx9ADIG/nTAHo////UCAT7NAVIAvs0BAggucgb+dQAyCC5yBZ51ZQTDbn///B/3/RzM
fPzsWamIuWlZO/sjItK7ywrL41jmH////d+yDJ7xVPEAUgye81T5VQEMtMye9AYI1giwB+jDM
AAGADvxIAQInJR50XaJ0KAEBgjWCLAH6MPAAAYAO/G0tntKEHjAeuqayoZ4wHtK+ssGe
dsq+sr6NnjAelq6+w9K6psrB/Die0rqmysH8OKLSuqbKwZAemqWevtK+neLSlrHh/Aq2lsm
eitbOvp+6ytbSlsn6MObS4pa5nsKW0syevtAedGbKvpn8FN7S1sK6pfwUotLWwrql/BSq0t

```
bCuqeSupQD//0eiobR/DTCtqaR/DSOtqaRnrKyhowBAgMDBgABHjGiM22ebaJtQjGOMfw
FRB4gphIDEgFdxB4gU7aWtr6ztpa2pqPKvrK+jcQiIrqWsaIMIaJ0IcQeIYHa0r652jXaLUQeI
GbikrrLys7XzoqHup7PkrrLrpaWwUQeIOYHBT38PLwBRBogpwgyCV4xqjEKupai0YK6lqLR
Pfh41jCdRB4gJi/7kr63yr+SuodzenN2c3t2ew93Pys3LAEedraWtr6x2na2lramo5qavYIwgr
7S1ofKso/Kjs2CMIKylpO61smCutbL0s6msYIwgtLOprHp+miIgAGADv2ADvx8gsefo6LVPh
dq1d4XbtE6Y1XawCbHaIMnjyEwP7qn/hdVg6KkAlXiVoLV3OPVPlVBMI+j/IBXnpc/QKKXOY
CA07qTIyTCwIcAosB1g6uogNO5g6oqiAbTOlEy0SJTKyvD1qmCgd0zg46B70PkgVOKl2tA
HpdvQA0x+5wbOJs8m5ibnpebF2qXn5duQCoXnpebl2oXm5s6I0OFg////////IBXnbM4ApU
zQAsZNxkylSNACxknGSKAAsUyRSKXKxUyly+VNkOBMU+7JKLCbqKXIYOrqmKqgbiDE4
4qoIMTjoHJMxOMgFecGzibPMPqw3NAExc6w1mAgFeexzpSfTAjnIDTupc5IIBXnaJHOYP//
/yBs7qXOhealz4XnTETiIOTuTDThIOTutHi1UGn+sAGIhdqE2xhlzpVQmGXPlXigALVQ0drIt
Xjx2rCATCPoIBXnpU4gCOelT9AExU5pACl/hU+VoKARpU8KGGlACiZOJk+I0PKlziAI56XPl
aBMeuIgFeekzsRMpc/lTZAfhEilz4VJTLbuIBXnpM7EyqXP5cuwCYRKpc+FS0y35UzL7urq6
uogye8gceFMv+8gA+6p/4XIqXSNAAJgIDbn6CA257VQYKkAhUqFTKklhUupEIVNTK3l1Xj
QARhMAuEgt+VMNuggt+VMW+jggNABiEwM4A==";

function loadWozmonROM(wozBytes) {
  const rom = new Uint8Array(0x0400).fill(0xFF);

  if (wozBytes.length !== 256) {
    throw new Error("WOZMON must be exactly 256 bytes (256-byte image for
$FF00-$FFFF).");
  }



  rom.set(wozBytes, 0x300);
  rom[0x3FA] = 0x00; rom[0x3FB] = 0xFF;  // $FF00
rom[0x3FC] = 0x00; // low byte
rom[0x3FD] = 0xFF; // high byte

  rom[0x3FE] = 0x00; rom[0x3FF] = 0xFF;  // $FF00

  cpu.rom.set(rom);
}
function loadACIRom(aciBytes) {
  const aci = new Uint8Array(0x100).fill(0xFF);
  const len = Math.min(aciBytes.length, 0x100);
  aci.set(aciBytes.slice(0, len), 0);
  if (typeof cpu !== 'undefined' && cpu && cpu.aci) {
    cpu.aci.set(aci);
    console.log('ACI ROM loaded (' + len + ' bytes) into $C100-$C1FF');
  } else {
```

```
    window._pendingACI = aci;
  }
}
function loadIntegerBasicROM(b64){
  const bytes = b64ToBytes(b64);
  const rom = new Uint8Array(0x1000).fill(0xFF);
  rom.set(bytes.slice(0,0x1000),0);
  cpu.basic.set(rom);
}

function hardReset() {
    cpu._kbdBuf = [];
    videoOutputQueue = [];
    wozmonActive = false;


    cpu.A = cpu.X = cpu.Y = 0;
    cpu.S = 0xFF;
    cpu.C = cpu.Z = cpu.I = cpu.D = cpu.B = cpu.V = cpu.N = 0;
    cpu.ram.fill(0);
    initVideoPattern();
    cpu.PC = 0x0000;
    running = false;
    termEl.focus();
  }
function warmReset() {
    cpu._kbdBuf = [];
    videoOutputQueue = [];
    cpu.PC = 0xFF00;
    wozmonActive = true;
  if (screen && screen[0]) {
    let isBootRow = screen[0].every((ch, i) => (i % 2 === 0 ? ch === '_' : ch === '@'));
    if (isBootRow) {
      screen[0] = Array(COLS).fill(' ');
      renderScreen();
    }
  }
    running = true;
    termEl.focus();
  }
(function boot(){
  termEl.focus();
  loadWozmonROM(b64ToBytes(WOZMON_BASE64));
```

```
 loadIntegerBasicROM(INTBASIC_BASE64);
 hardReset();
 requestAnimationFrame(tick);
})();

document.getElementById('btnReset').addEventListener('click', () => {
 hardReset();
});

document.getElementById('btnWarmReset').addEventListener('click', () => {
 warmReset();
});

document.getElementById('btnClear').addEventListener('click', () => {

 clearScreen();
});

document.getElementById('btnBreak').addEventListener('click', () => {
 cpu.triggerBreak();
});

</script>
<script>
document.addEventListener("keydown", (e) => {
    if (!wozmonActive) return;
    keyBuffer.push(e.key);
  });
</script>
<script>
 const terminal = document.getElementById("terminal");
 terminal.addEventListener("keydown", function(e) {
   if (e.code === "Space") {
    e.preventDefault();
   }
 });
 </script>
<script>
document.addEventListener("DOMContentLoaded", () => {
 const term = document.getElementById("terminal") ||
document.querySelector("#terminal");
 const pasteBtn = document.getElementById("pasteBtn");
 if (!term || !pasteBtn) return;
```

```
 term.setAttribute("tabindex", "0");
 term.focus();
 term.addEventListener("click", () => term.focus());

 pasteBtn.addEventListener("click", async () => {
  try {
   const text = await navigator.clipboard.readText();
   if (text) injectIntoEmulator(text);
  } catch (err) {
   console.warn("Clipboard read failed:", err);
  }
 });

 function injectIntoEmulator(text) {
  const normalized = text.replace(/\r\n/g, "\n").replace(/\r/g, "\n");
  for (const ch of normalized) {
   if (ch === "\n") {
    if (typeof enqueueKey === "function") {
     enqueueKey(0x0D); // CR
    } else if (window.cpu && Array.isArray(cpu._kbdBuf)) {
     cpu._kbdBuf.push(0x8D);
    }
   } else {
    const code = ch.toUpperCase().charCodeAt(0) & 0x7F;
    if (typeof enqueueKey === "function") {
     enqueueKey(code);
    } else if (window.cpu && Array.isArray(cpu._kbdBuf)) {
     cpu._kbdBuf.push(code | 0x80);
    }
   }
  }
  if (typeof renderScreen === "function") {
   try { renderScreen(); } catch(e) {}
  }
  term.focus();
 }
});
</script>

<!--Keeps terminal display focused and ready for input at all times.-->
<style>
.keep-focus-target { caret-color: auto !important; outline: none !important; }
```

```
</style>
<script>
(function(){
 function keepFocus(){
  const term =
document.querySelector('textarea,[contenteditable="true"],[contenteditable],canvas,#
terminal,.terminal,#term,.term,#console,.console');
  if(!term) return;
  if(term.tagName === 'CANVAS' && !term.hasAttribute('tabindex'))
term.setAttribute('tabindex','0');
  term.classList.add('keep-focus-target');
  setInterval(()=>{
   if(document.activeElement !== term && !document.hidden){
    term.focus();
    if(term.setSelectionRange){
     const pos = term.value.length;
     term.setSelectionRange(pos,pos);
    }
   }
  },300);
 }
 document.addEventListener('DOMContentLoaded', keepFocus);
})();
</script>



<script>
if (typeof window.keyBuffer === "undefined") window.keyBuffer = [];
function findMemoryTarget() {
  if (typeof window !== 'undefined' && window.memory instanceof Uint8Array) return
window.memory;
  if (typeof window !== 'undefined' && window.ram instanceof Uint8Array) return
window.ram;
  if (typeof cpu !== 'undefined') {
    if (cpu.memory instanceof Uint8Array) return cpu.memory;
    if (cpu.ram instanceof Uint8Array) return cpu.ram;
    if (cpu.mem instanceof Uint8Array) return cpu.mem;
  }
  for (let k in window) {
    try {
      if (window[k] instanceof Uint8Array && window[k].length >= 0x1000) return
window[k];
```

```
      } catch(e){}
    }
    return null;
}
document.getElementById('hcLoader').addEventListener('change', function(e) {
  const file = e.target.files[0];
  if (!file) return;

  const reader = new FileReader();
  reader.onload = function(ev) {
    const lines = ev.target.result.split(/\r?\n/).filter(l => l.trim() !== "");

    let mem = findMemoryTarget();
    if (!mem) {
      alert("No writable memory array found!");
      return;
    }

    let addr = 0;
    let addressCount = 0;

    for (let i = 0; i < lines.length; i++) {
      const line = lines[i].trim();
      if (line === "") continue;


      if (!line.startsWith(":") && /^[0-9A-Fa-f]{4}$/.test(line)) {
        addr = parseInt(line, 16);
        addressCount++;
        console.log(`Setting load address to $${addr.toString(16).toUpperCase()}`);
        continue;
      }


      if (line.startsWith(":")) {
        const bytes = line.substring(1).trim().split(/\s+/);
        for (const b of bytes) {
          if (b === "") continue;
          const val = parseInt(b, 16);
          mem[addr++] = val;
        }
      }
    }
```

```
  console.log(`Loaded HC file with ${addressCount} address block(s)`);
 };
 reader.readAsText(file);
});
</script>
<script>
document.querySelectorAll('input[name="memsize"]').forEach(radio => {
 radio.addEventListener('change', () => {
  location.reload();
 });
});
</script>

</body>
</html>
```

---

8. License and Attribution

8.1. MIT License

8.2. Acknowledgments

- Steve Wozniak - Original Apple-1 hardware/software design
- 6502 CPU designers **Chuck Peddle, Rod Orgill, and Wil Mathys** at MOS Technology
- San Bergmans at www.sbprojects.net for his extensive Apple-1 documentation
- Will Scullin's APPLE-1JS as inspiration for the development of this emulator.

8.3. Contact Information

For questions, bug reports, or contributions, please contact landon@producerjason.com.