Introduction to the NES Hardware.

1983–2003 (Famicom)

1993–1995 (NES-101)

1985–1995 (NES)

Design Overview

https://www.pcworld.com/article/207891/inside_the_nes.html


AUDIO OUT
COMPOSITE
VIDEO OUT

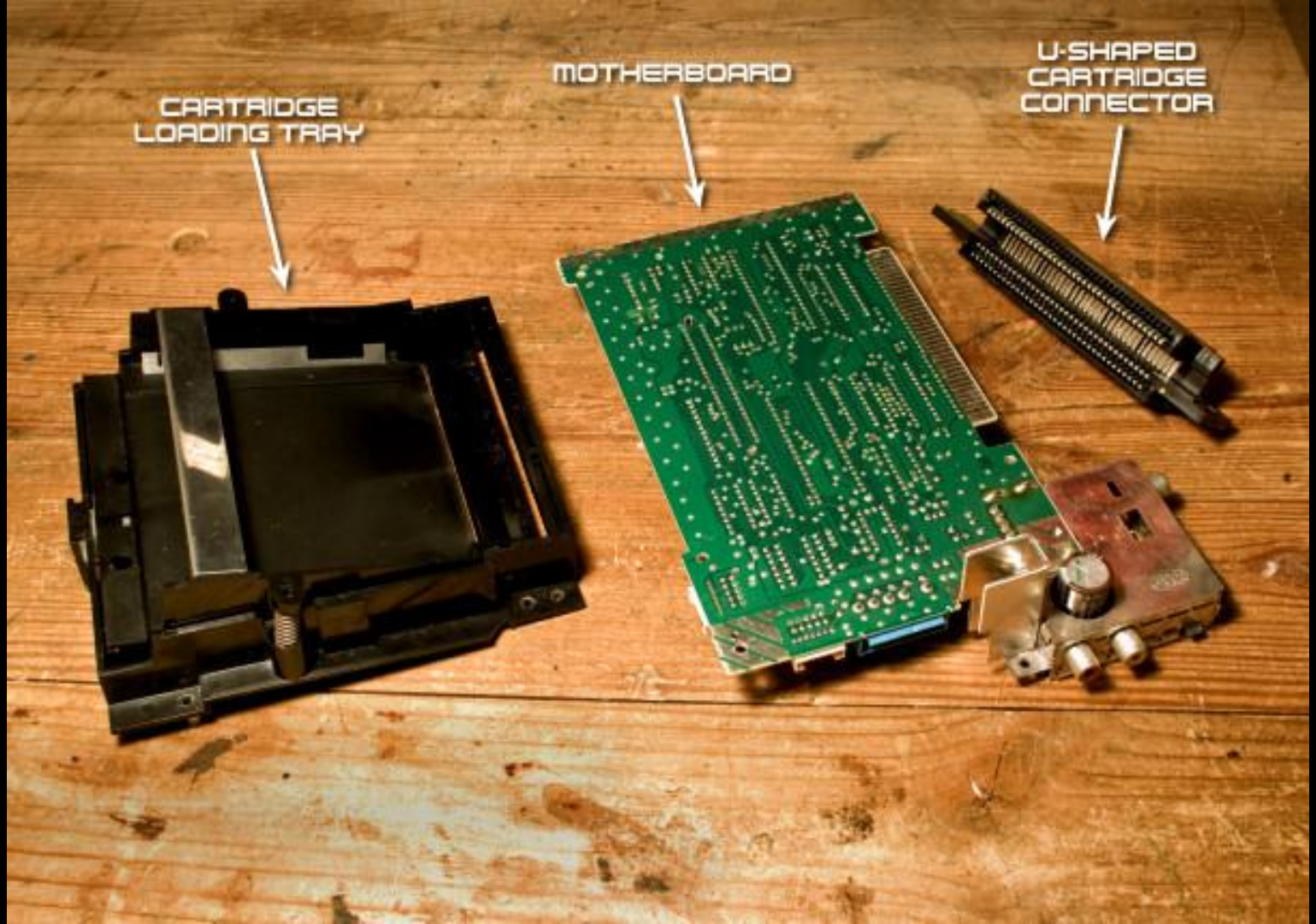

POWER IN
CHANNEL SWITCH
RF SWITCH CONNECTOR


EXPANSION PORT

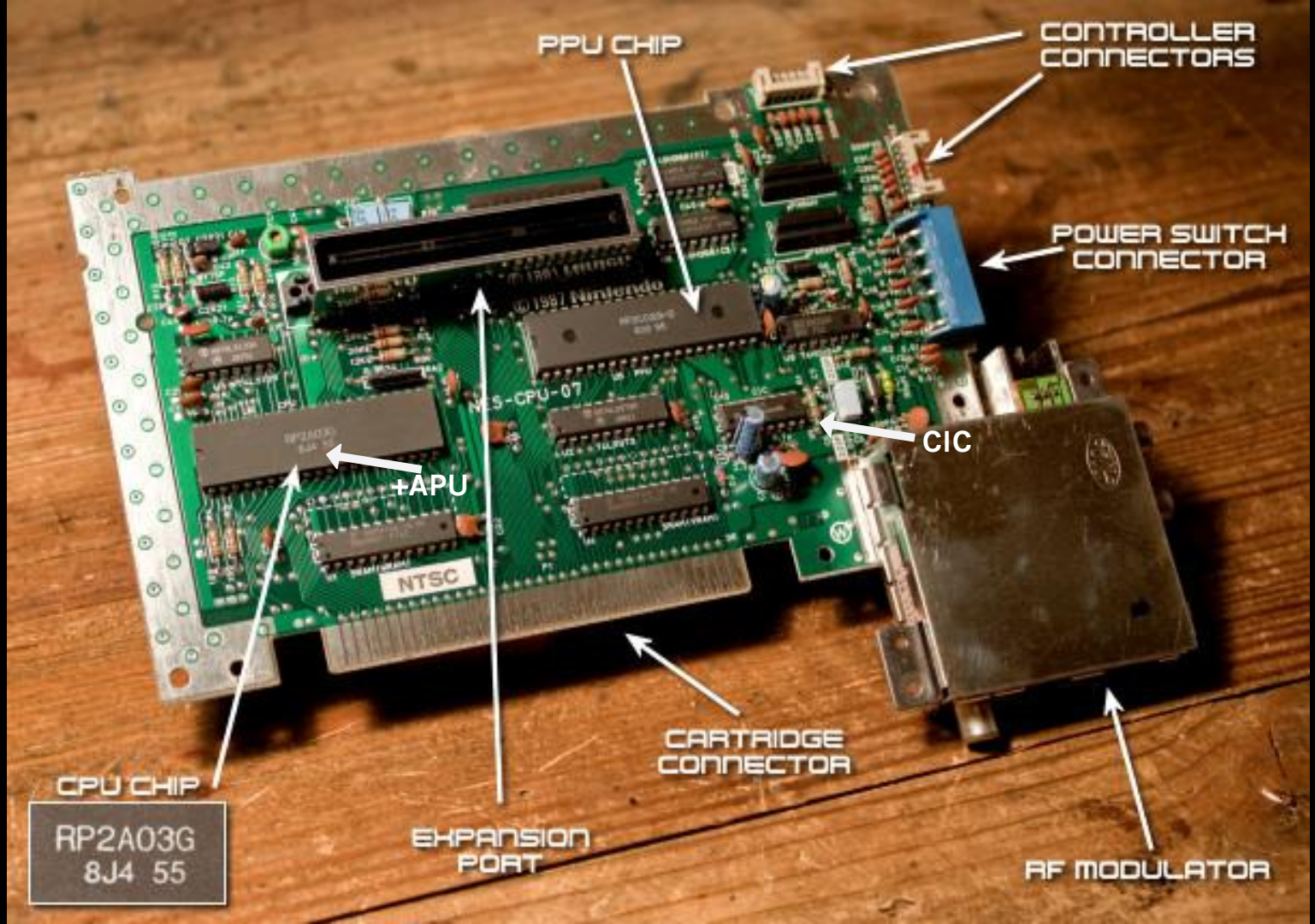

RF SHIELDING

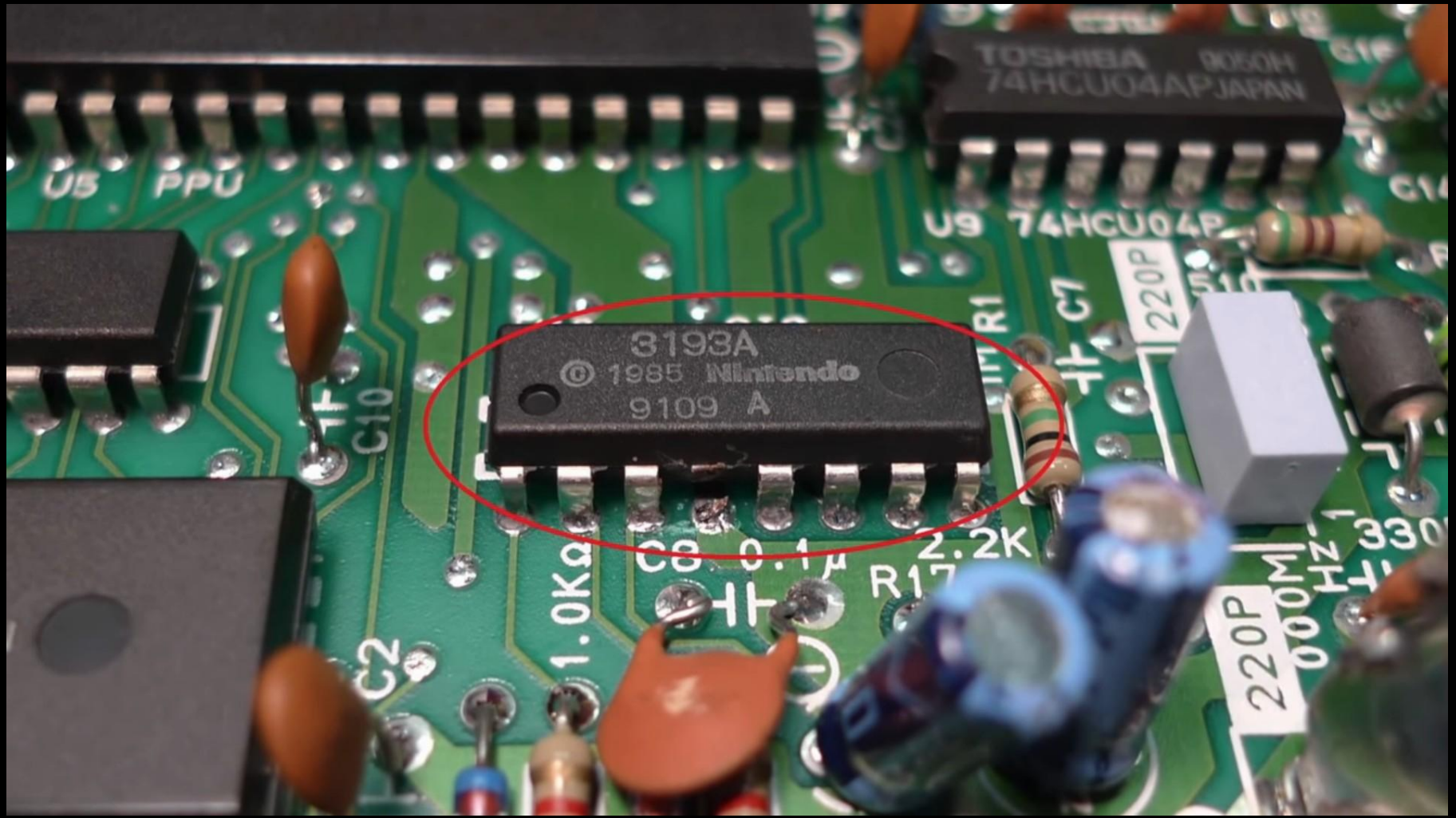
https://www.pcworld.com/article/207891/inside_the_nes.html

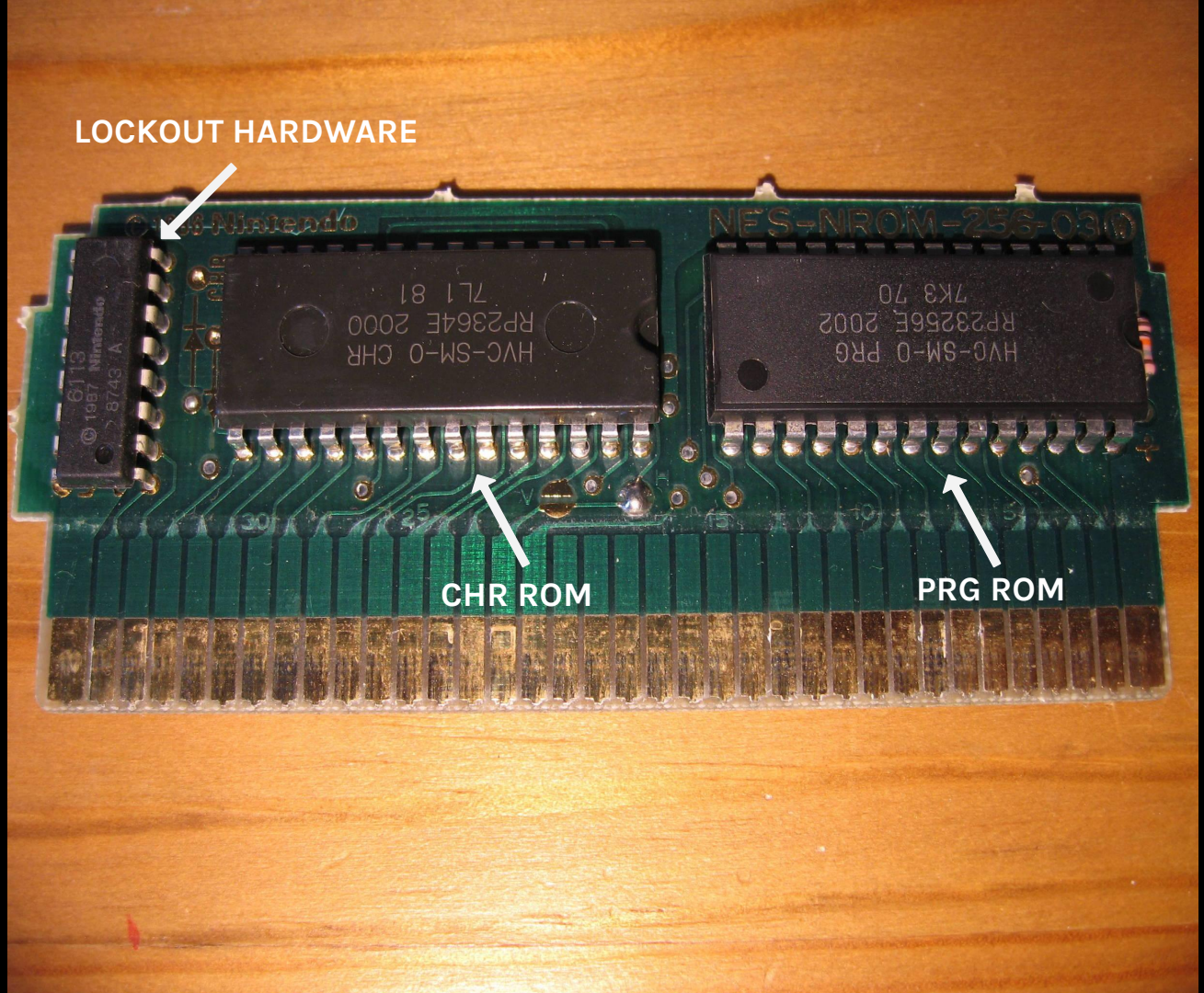

CONTROLLER
CONNECTORS
POWER / RESET
SWITCHES


CARTRIDGE
LOADING TRAY
MOTHERBOARD
U-SHAPED
CARTRIDGE
CONNECTOR

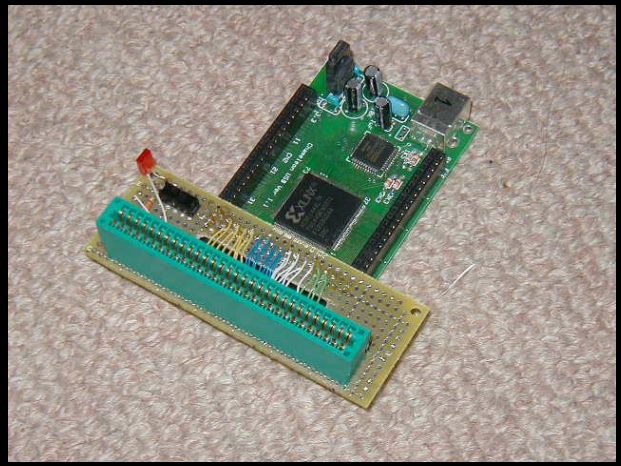

PROFILE VIEW

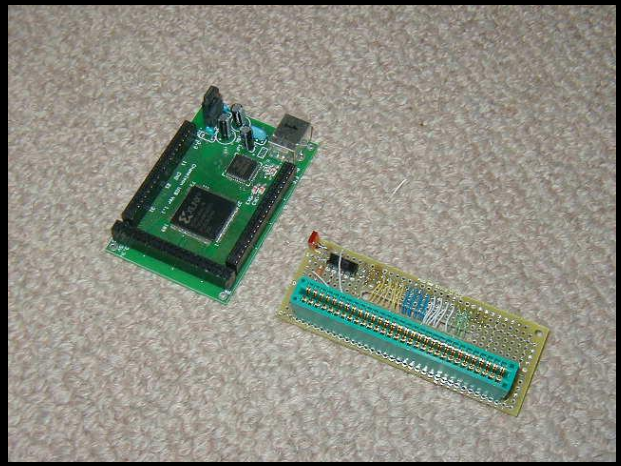

PPU CHIP
CONTROLLER CONNECTORS
POWER SWITCH CONNECTOR
CIC
+APU
CPU CHIP
RP2A03G
8J4 55
EXPANSION PORT
CARTRIDGE CONNECTOR
RF MODULATOR
NES-CPU-07
NTSC


U5 PPU
TOSHIBA 9050H
74HCU04APJAPAN
U9 74HCU04P
3193A
© 1985 Nintendo
9109 A
C10
R1
C7
220P
C8 0.1μ
R17
2.2K
1.0KΩ
C2
220P
330


LOCKOUT HARDWARE
NES-NROM-256-03
CHR ROM
PRG ROM

http://www2.teamknox.com/teamknox_old/ChameleonNES/ChameleonNES.html

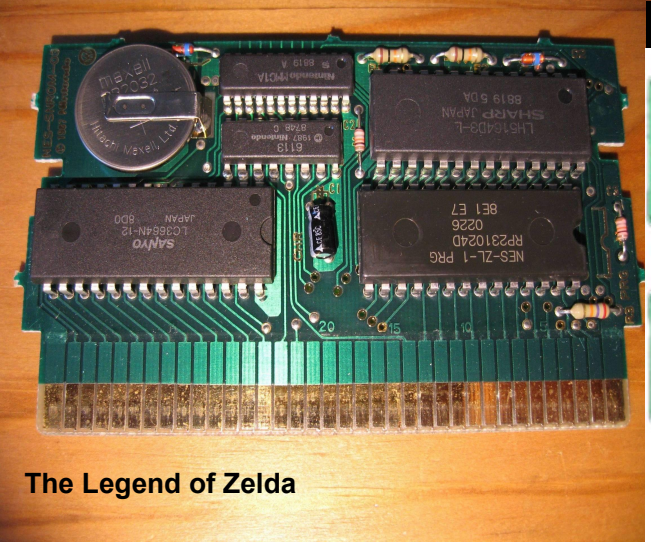




Castlevania

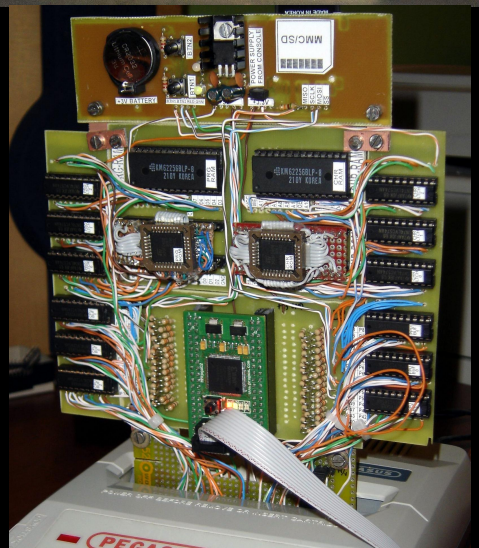
Super Mario Brothers

Nintendo World Championships 90



The Legend of Zelda

Super Mario Bros 3



Aladdin™

DECK ENHANCER™

Use Aladdin Deck Enhancer to play a complete range of Compact Cartridges

Includes: Compact Cartridge "Dizzy the Adventurer" & Deck Enhancer

UPGRADE YOUR NINTENDO ENTERTAINMENT SYSTEM™

Nintendo Entertainment System™ is a trademark of Nintendo of America Inc. The Aladdin Deck Enhancer is a product of Camerica Corporation and is not manufactured, distributed, endorsed or licensed by Nintendo of America Inc.

Aladdin™
DECK ENHANCER™

Aladdin™
DECK ENHANCER™

- Designed by the inventors of Game Genie™
- 64K memory upgrade for better graphics, bigger games!
- Includes Compact Cartridge "Dizzy the Adventurer"
- Aladdin is the future in console game play

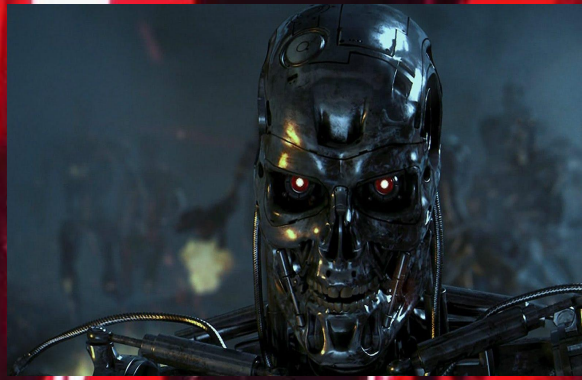
CPU - Ricoh 2A03

CPU SPECS

CPU - Ricoh 2A03

- **MOS6502** without decimal mode.
- 8-bit microprocessor.
- 16-bit address width (64kb of RAM).
- 6 registers (A, Y, X, PC, SP and P(status))
- Complex Instruction Set Computer (CISC)
  - 56 instructions
  - **13 address modes**
  - 143 valid opcodes (+113 unofficial opcodes)
  - Microcoded design
- Very popular CPU:
  - Atari 2600, Atari 8-bit family, Apple II, Nintendo Entertainment System, Commodore 64, Atari Lynx, BBC Micro, Tamagotchi and T-800 (Terminator)

```
 8 ********************
 9
10            ORG   $4000
11 A1         =     $3C
12 A2         =     $3E
13 A4         =     $42
14 AUXMOVE    =     $C311
15
16 ****************************
17 * SETUP - move data for VTOC
18 * and catalog to auxmem at
19 * BC00-BFFF (pseudo trk 11
20 * 0-3)
21 ****************************
22 SETUP      LDA   #<VTOC
23            STA   A1
24            LDA   #>VTOC
25            STA   A1+1
26            LDA   #<END
27            STA   A2
28            LDA   #>END
29            STA   A2+1
30            LDA   #$00
31            STA   A4
32            LDA   #$BC
33            STA   A4+1
34            SEC
35            JMP   AUXMOVE
36
```



HI	LO-NIBBLE															
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	BRK impl	ORA X,ind				ORA zpg	ASL zpg		PHP impl	ORA #	ASL A			ORA abs	ASL abs	
10	BPL rel	ORA ind,Y				ORA zpg,X	ASL zpg,X		CLC impl	ORA abs,Y				ORA abs,X	ASL abs,X	
20	JSR abs	AND X,ind			BIT zpg	AND zpg	ROL zpg		PLP impl	AND #	ROL A		BIT abs	AND abs	ROL abs	
30	BMI rel	AND ind,Y				AND zpg,X	ROL zpg,X		SEC impl	AND abs,Y				AND abs,X	ROL abs,X	
40	RTI impl	EOR X,ind				EOR zpg	LSR zpg		PHA impl	EOR #	LSR A		JMP abs	EOR abs	LSR abs	
50	BVC rel	EOR ind,Y				EOR zpg,X	LSR zpg,X		CLI impl	EOR abs,Y				EOR abs,X	LSR abs,X	
60	RTS impl	ADC X,ind				ADC zpg	ROR zpg		PLA impl	ADC #	ROR A		JMP ind	ADC abs	ROR abs	
70	BVS rel	ADC ind,Y				ADC zpg,X	ROR zpg,X		SEI impl	ADC abs,Y				ADC abs,X	ROR abs,X	
80		STA X,ind			STY zpg	STA zpg	STX zpg		DEY impl		TXA impl		STY abs	STA abs	STX abs	
90	BCC rel	STA ind,Y			STY zpg,X	STA zpg,X	STX zpg,Y		TYA impl	STA abs,Y	TXS impl			STA abs,X		
A0	LDY #	LDA X,ind	LDX #		LDY zpg	LDA zpg	LDX zpg		TAY impl	LDA #	TAX impl		LDY abs	LDA abs	LDX abs	
B0	BCS rel	LDA ind,Y			LDY zpg,X	LDA zpg,X	LDX zpg,Y		CLV impl	LDA abs,Y	TSX impl		LDY abs,X	LDA abs,X	LDX abs,Y	
C0	CPY #	CMP X,ind			CPY zpg	CMP zpg	DEC zpg		INY impl	CMP #	DEX impl		CPY abs	CMP abs	DEC abs	
D0	BNE rel	CMP ind,Y				CMP zpg,X	DEC zpg,X		CLD impl	CMP abs,Y				CMP abs,X	DEC abs,X	
E0	CPX #	SBC X,ind			CPX zpg	SBC zpg	INC zpg		INX impl	SBC #	NOP impl		CPX abs	SBC abs	INC abs	
F0	BEQ rel	SBC ind,Y				SBC zpg,X	INC zpg,X		SED impl	SBC abs,Y				SBC abs,X	INC abs,X	

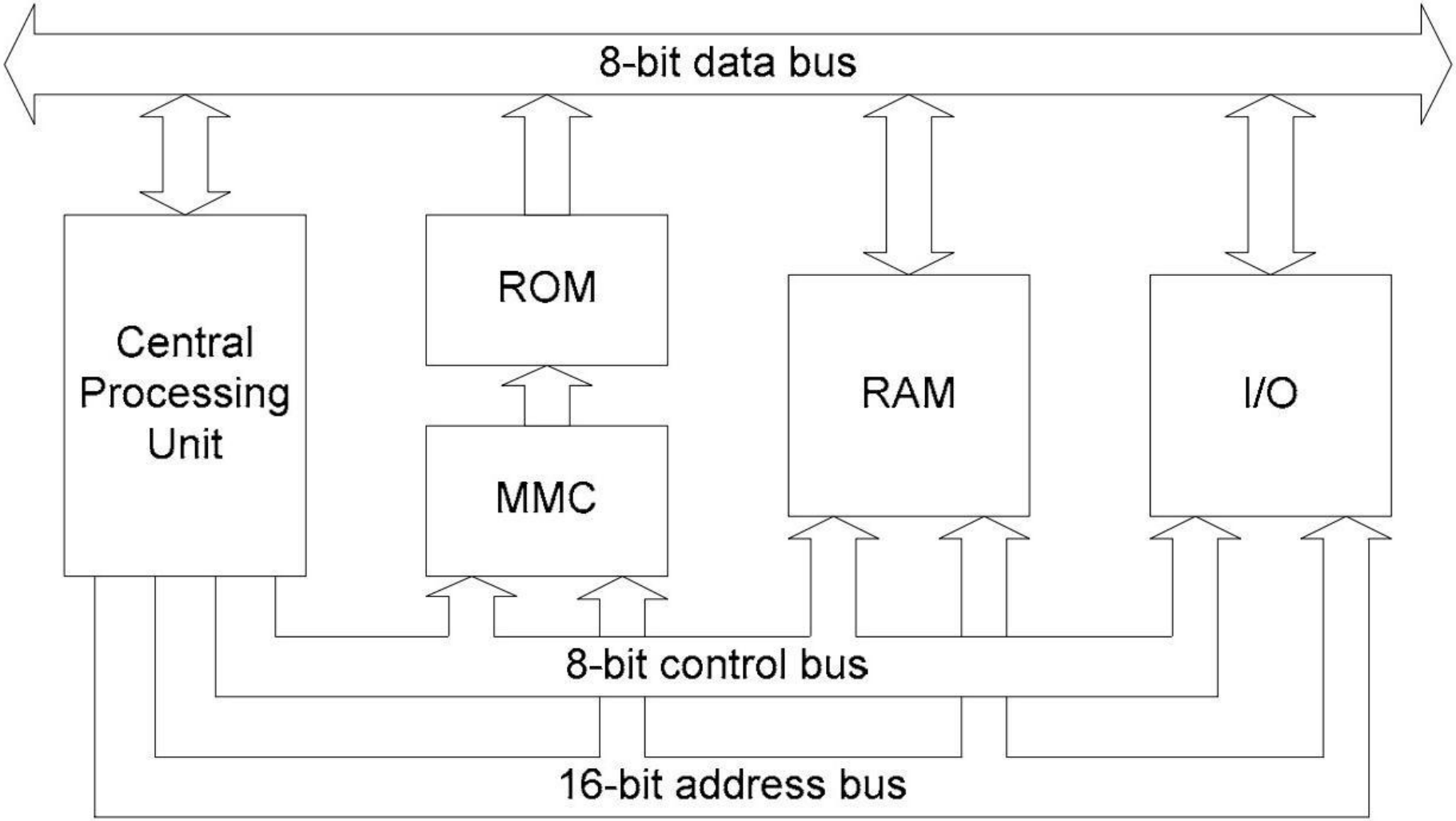
Addressing Modes

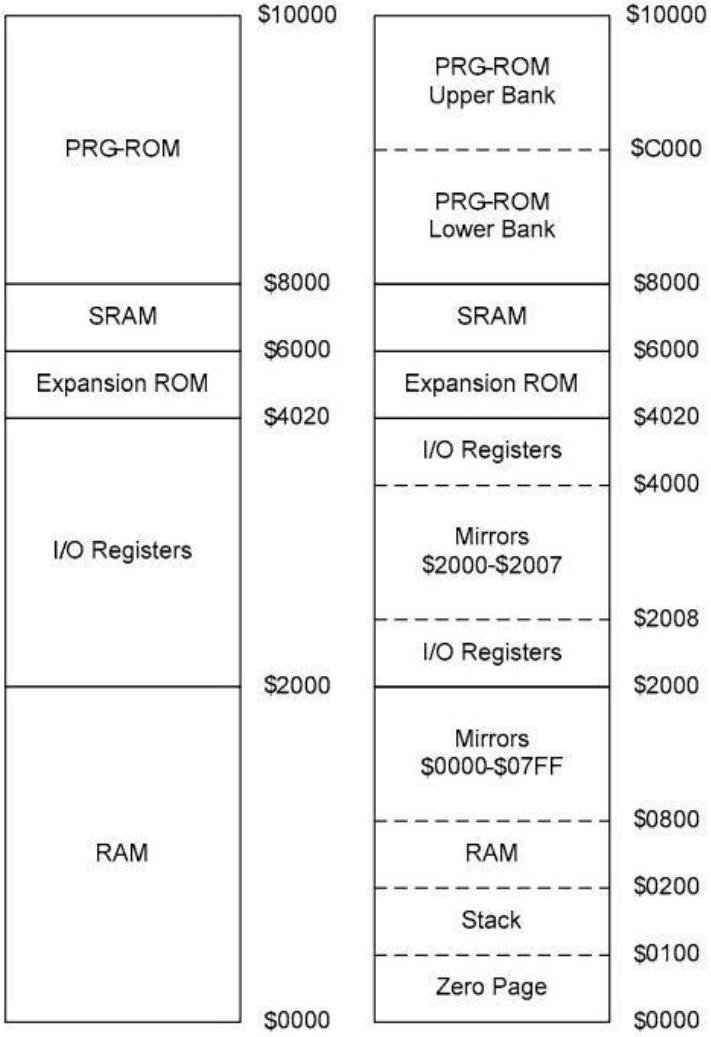
- **Implicit:**
    - Clear Carry Flag' (CLC) and Return from Subroutine (RTS).
- **Accumulator:**
    - ROR A ;Rotate right one bit
    - LSR A ;Logical shift right one bit
- **Immediate:**
    - LDA #10 ;Load 10 ($0A) into the accumulator
- **Zero Page (0x0-0xFF):**
    - LDA $00 ;Load accumulator from $00
- **Zero Page,X:**
    - LDA $00, X ; A = Memory[X & 0xFF]
- **Zero Page,Y:**
    - LDA $10,Y ; A = Memory[Y & 0xFF]
- **Relative:**
    - BEQ LABEL ;Branch if zero flag set to LABEL
    - BNE *+4 ;Skip over the following 2 byte instruction

Addressing Modes

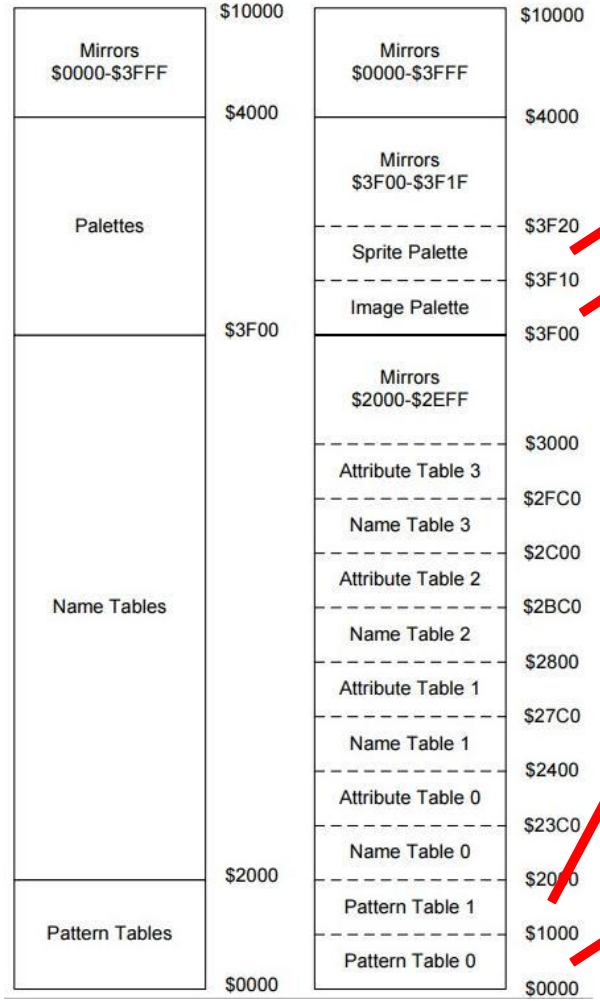
- **Absolute, X:**
    - STA $3000,X ;Store accumulator between $3000 and $30FF
    - ROR CRC,X ;Rotate right one bit
- **Absolute,Y**
    - AND $4000,Y ;Perform a logical AND with a byte of memory
    - STA MEM,Y ;Store accumulator in memory
- **Indirect: (R = ((Memory[Addr] << 8) | Memory[Addr + 1]))**
    - JMP ($FFFC) ;Force a power on reset
    - JMP (TARGET) ;Jump via a labelled memory area
- **Indexed Indirect: (R = Memory[Memory[addr] + x])**
    - LDA ($40,X) ;Load a byte indirectly from memory
    - STA (MEM,X) ;Store accumulator indirectly into memory
- **Indirect Indexed: (R = Memory[Memory[addr]] + y)**
    - LDA ($40),Y ;Load a byte indirectly from memory
    - STA (DST),Y ;Store accumulator indirectly into memory

Memory Mapping


8-bit data bus
Central Processing Unit
ROM
MMC
RAM
I/O
8-bit control bus
16-bit address bus


$10000
PRG-ROM
$8000
SRAM
$6000
Expansion ROM
$4020
I/O Registers
$2000
RAM
$0000
$10000
PRG-ROM
Upper Bank
$C000
PRG-ROM
Lower Bank
$8000
SRAM
$6000
Expansion ROM
$4020
I/O Registers
$4000
Mirrors
$2000-$2007
$2008
I/O Registers
$2000
Mirrors
$0000-$07FF
$0800
RAM
$0200
Stack
$0100
Zero Page
$0000

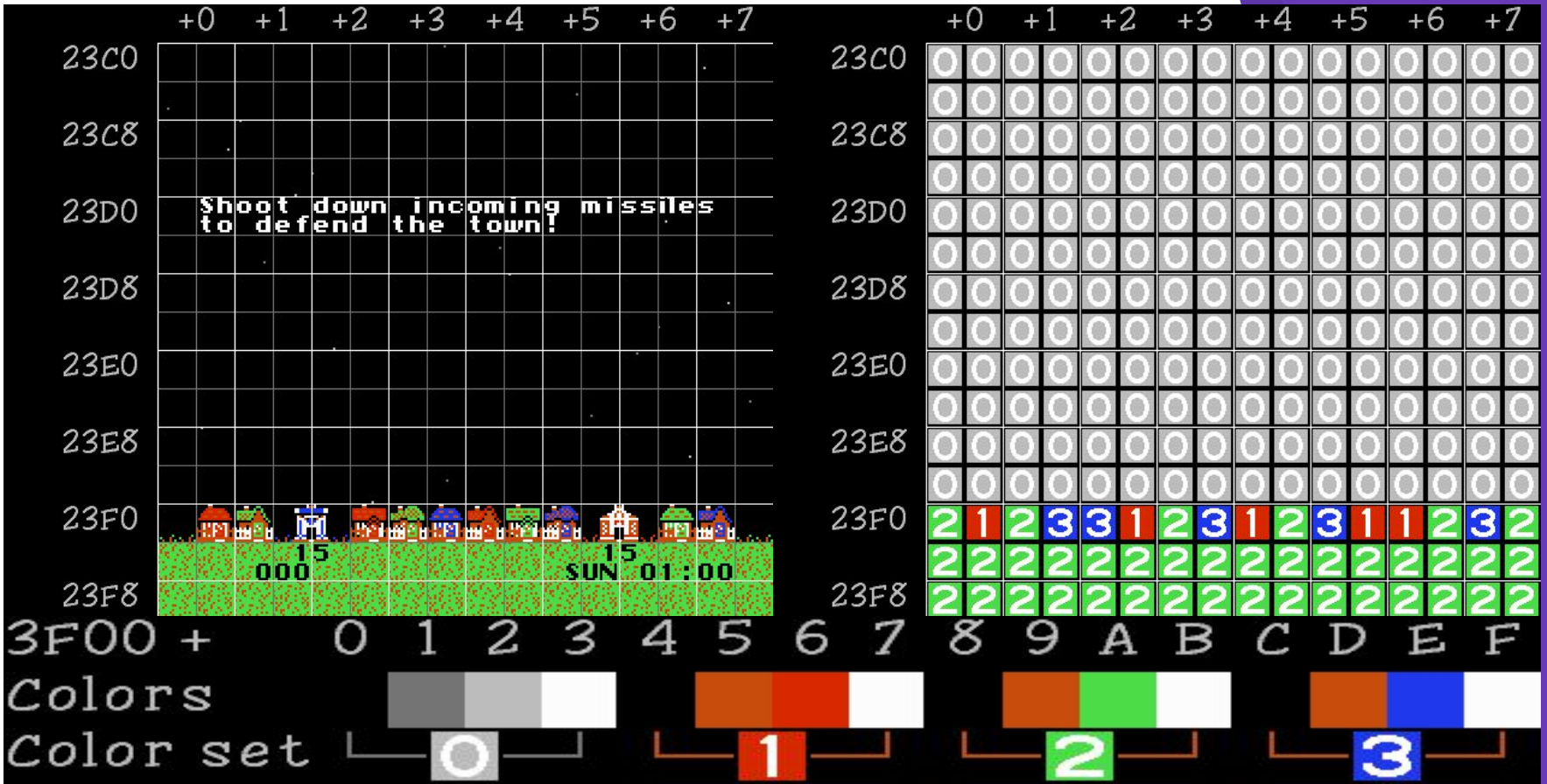
PPU




```
$0xx0=$41  01000001
$0xx1=$C2  11000010
$0xx2=$44  01000100
$0xx3=$48  01001000
$0xx4=$10  00010000
$0xx5=$20  00100000          .1.....3
$0xx6=$40  01000000          11....3.
$0xx7=$80  10000000  =====   .1...3..
                             .1..3...
$0xx8=$01  00000001  =====   ...3.22.
$0xx9=$02  00000010          ..3....2
$0xxA=$04  00000100          .3....2.
$0xxB=$08  00001000          3....222
$0xxC=$16  00010110
$0xxD=$21  00100001
$0xxE=$42  01000010
$0xxF=$87  10000111
```

PPU attribute/name tables

- Attribute tables controls which palette is assigned to each part of the background.
- Name tables are essentially a matrix of tile numbers, pointing to the tiles stored in the pattern tables.



The OAM (Object Attribute Memory)

- is internal memory inside the PPU that contains a display list of up to 64 sprites, where each sprite's information occupies 4 bytes:

b1 - Y Position - vertical position of the sprite on screen. $00 is the top of the screen. Anything above $EF is off the bottom of the screen.

b2 - Tile Number - this is the tile number (0 to 256) for the graphic to be taken from a Pattern Table.

b3 - Attributes - this byte holds color and displaying information:

```
76543210
| ||   ||
| ||   ++- Color Palette of sprite.  Choose which set of 4 from the 16 colors to use
| ||
| |+------ Priority (0: in front of background; 1: behind background)
| +------- Flip sprite horizontally
+-------- Flip sprite vertically
```

b4 - X Position - horizontal position on the screen. $00 is the left side, anything above $F9 is off screen.

Hello World!!!

Recommendations:

Learn the 6502 assembly programming:

- Best tutorial: http://skilldrick.github.io/easy6502/:

Make your own NES game:

- Best tutorial: http://nintendoage.com/forum/messageview.cfm?catid=22&threadid=7155

Javascript emulator:

- git clone https://github.cohttp://nintendoage.com/forum/messageview.cfm?catid=22&threadid=7155m/takahirox/nes-js
- cd ~/nes-js && firefox index.html

NROM template cartridge for ASM6:

- http://forums.nesdev.com/viewtopic.php?%20p=58138#58138

More info:

https://www.ic.unicamp.br/~rodolfo/Cursos/mc861/2019s2/

Just for fun:

- 1 bit guy: https://www.youtube.com/watch?v=Tfh0ytz8S0k
- AVGN: https://www.youtube.com/watch?v=vnnf42BQTfo