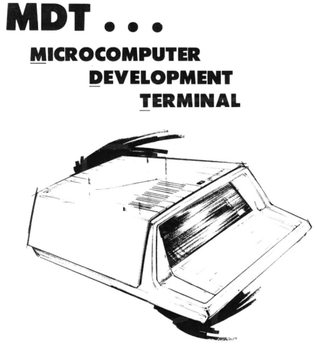
MOS 6502 Architecture Intro

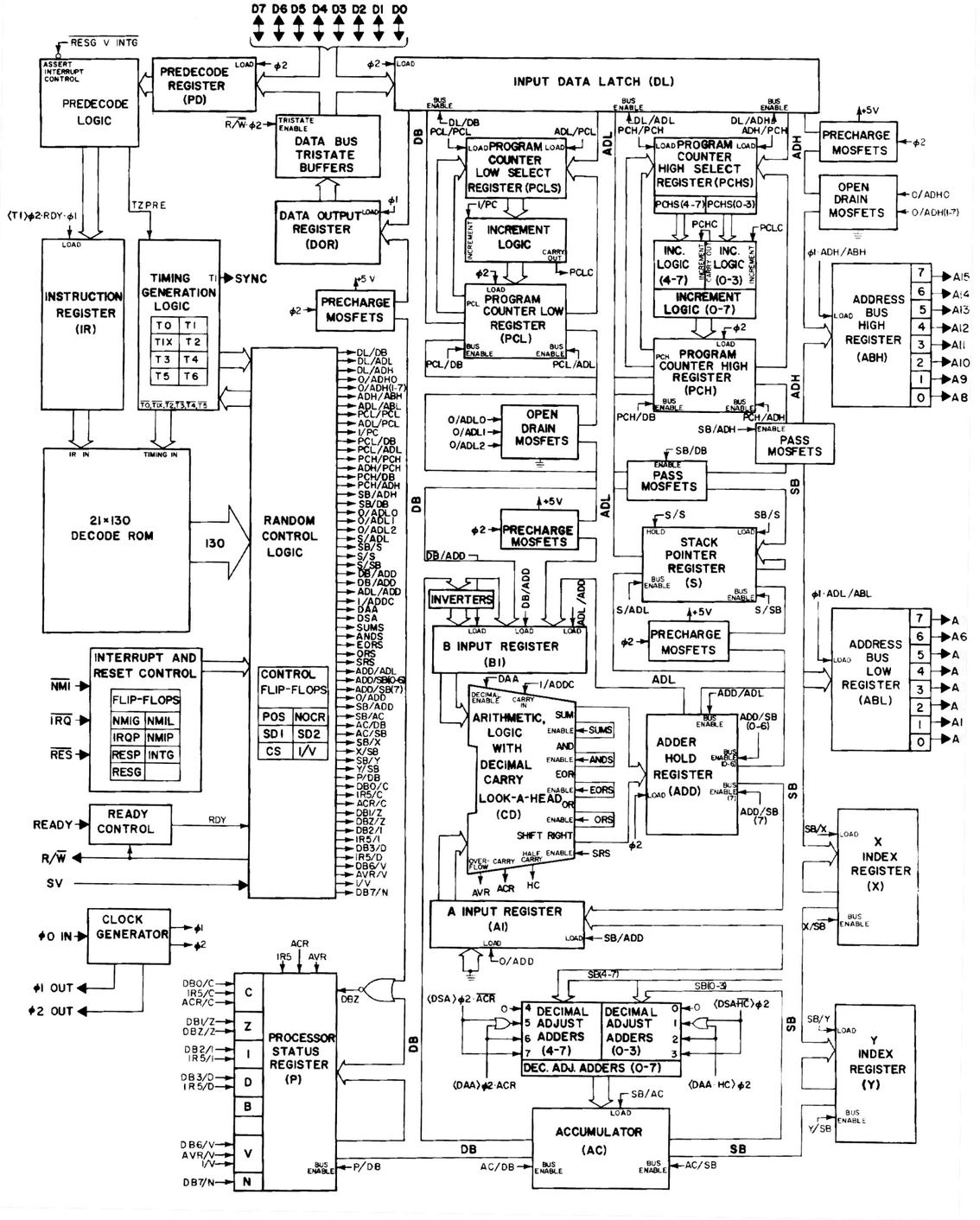
History

- Origins lie in the Motorola 6800. Was very expensive for consumers. ($300, or about $1700 in 2022 $s)
- Chuck Peddle proposes lower-cost, lower-area 6800 design (~$25, ~$100 today). Motorola won't have it. Chuck moves to MOS tech.
- Chooses much simpler design, only about 3500 transistors (!)



History (contd.)

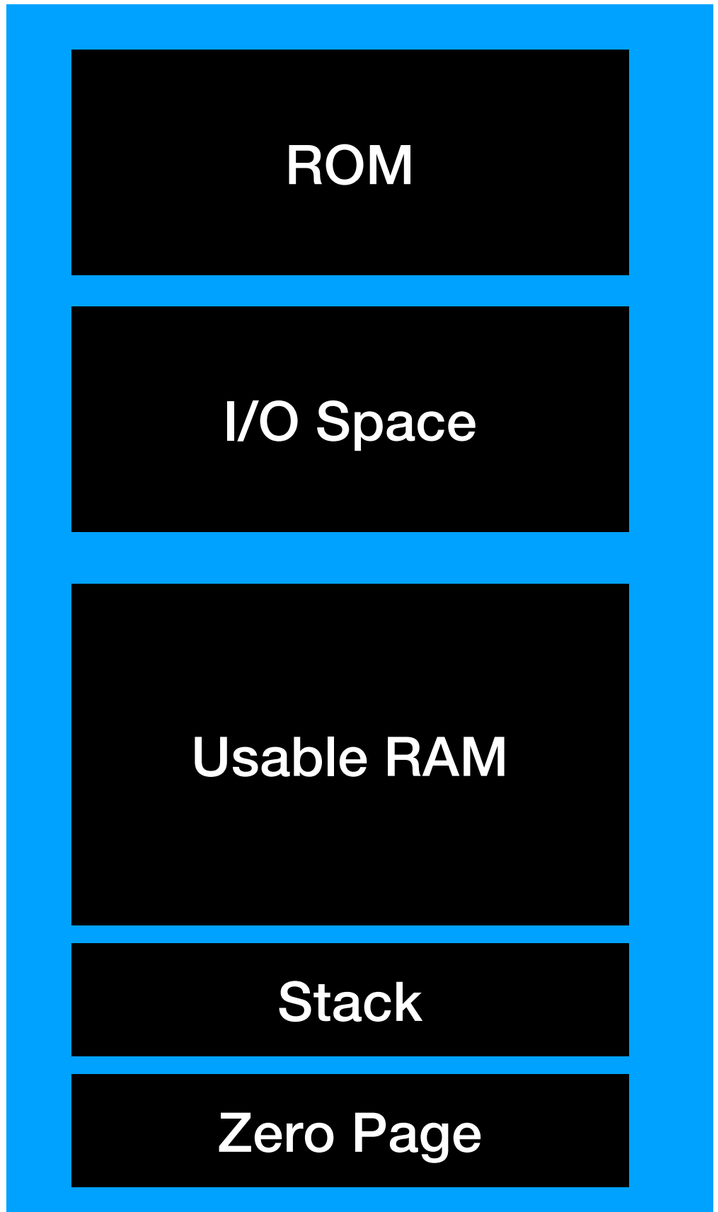
- 1975 sees a recession, sales of 6502 not good
- Peddle comes up with MDT-650, a single-board minicomputer/ dev. platform, which hobbyists eat up
- Other players see the potential, and the hobbyist game is on (Apple, Atari, Commodore, etc.)
- 6502 then used in Apple][, Commodore PET, BBC Micro, Atari 800, and more


D7 D6 D5 D4 D3 D2 D1 D0
RESG V INTG
ASSERT INTERRUPT CONTROL
PREDECODE LOGIC
PREDECODE REGISTER (PD)
INPUT DATA LATCH (DL)
DATA BUS TRISTATE BUFFERS
DATA OUTPUT REGISTER (DOR)
INSTRUCTION REGISTER (IR)
TIMING GENERATION LOGIC
SYNC
PRECHARGE MOSFETS
21×130 DECODE ROM
130
RANDOM CONTROL LOGIC
INTERRUPT AND RESET CONTROL
FLIP-FLOPS
NMIG NMIL
IRQP NMIP
RESP INTG
RESG
CONTROL FLIP-FLOPS
POS NOCR
SDI SD2
CS I/V
READY CONTROL
CLOCK GENERATOR
PROCESSOR STATUS REGISTER (P)
PROGRAM COUNTER LOW SELECT REGISTER (PCLS)
PROGRAM COUNTER HIGH SELECT REGISTER (PCHS)
INCREMENT LOGIC
INC. LOGIC (4-7)
INC. LOGIC (0-3)
INCREMENT LOGIC (0-7)
PROGRAM COUNTER LOW REGISTER (PCL)
PROGRAM COUNTER HIGH REGISTER (PCH)
OPEN DRAIN MOSFETS
PASS MOSFETS
STACK POINTER REGISTER (S)
ADDRESS BUS HIGH REGISTER (ABH)
ADDRESS BUS LOW REGISTER (ABL)
INVERTERS
B INPUT REGISTER (BI)
ARITHMETIC, LOGIC WITH DECIMAL CARRY LOOK-A-HEAD (CD) SHIFT RIGHT
ADDER HOLD REGISTER (ADD)
A INPUT REGISTER (AI)
DECIMAL ADJUST ADDERS (4-7)
DECIMAL ADJUST ADDERS (0-3)
DEC. ADJ. ADDERS (0-7)
ACCUMULATOR (AC)
X INDEX REGISTER (X)
Y INDEX REGISTER (Y)
DB
ADL
ADH
SB

Arch. Overview

- Byte-addressable, 16-bit address width. Word size is one byte.
- 3 8-bit registers (1 accumulator, 2 index regs for addressing modes (e.g. array subscripting))
  - `A, X, Y`
- 8-bit stack pointer (`SP`)
- 8-bit status register (I'll call this `PSW`)
- 16-bit program counter (`PC`)
- **Little Endian** (like x86)

Physical Memory Map



0x???? - 0xFFFF

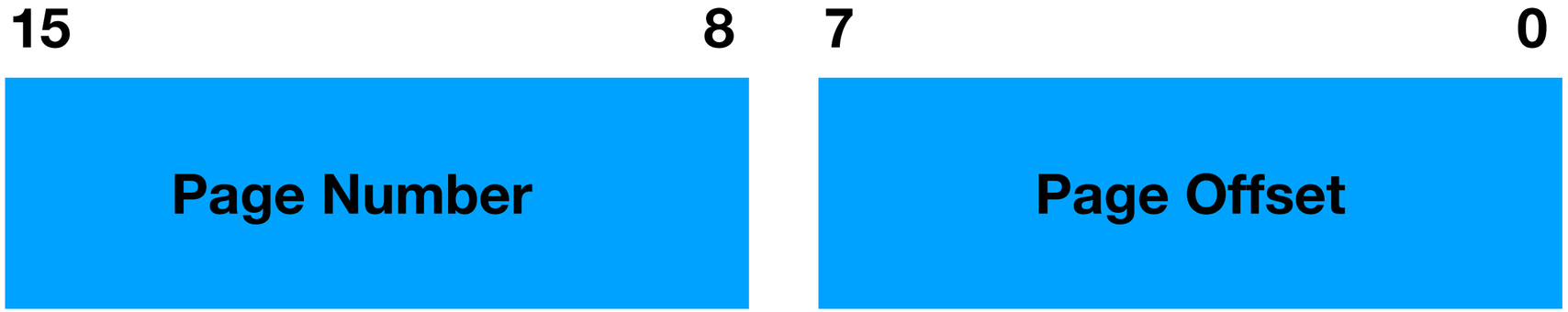
0x???? - 0x????

0x00200 - 0x????

0x0100 - 0x0200

0x0000 - 0x0100

Address Breakdown



256 Byte Page Size

We can get the page number of an address by shifting off the lower 8 bits

Instruction Encoding

- 256 possible opcodes, only 56 are actually used by the architecture
- 1, 2, or 3-byte instructions
- Data movement (`LD`/`ST`)
- Arithmetic: `ADD` (with carry), `SUB` (with carry), `DEC`, `INC`, `CMP`
- Logic: `AND`, `OR`, `XOR`, shift/rotate
- Control Flow: branch cond, branch uncond, call subroutine, return
- Other: `PS` manipulation, bit testing, stacks ops

Program Counter

- (16 bits) Holds address of current instruction

Stack Pointer

- Contains address of first empty location on stack (page 2)

Processor Status Word

- Holds status information for the processor. Each bit represents a *flag*
- Bit 0: "C" - Carry, Carry out of MSB in arithmetic ops. Also set if borrow required in SUB. Also used for shift/rotate ops
- Bit 1: “Z” - Set to 1 if any arith/logic operation produces a zero
- Bit 2: “I” - Interrupt disable. If set, interrupts are disabled (ignored). (Except NMIs)
- Bit 3: “D” - decimal mode status. We will ignore this (like the NES). Indicates to arith. unit to use BCD representation
- Bit 4: “B” - Set when software interrupt is executed (BRK instruction)
- Bit 5: unused
- Bit 6: “V” - Overflow flag: when arithmetic operation overflows, this is set
- Bit 7: “N” - Sign bit. Set when result of an operation is negative.

Addressing Modes

- **Immediate**: operand's value is in the instruction
- E.g., `LDA #$B7` —> load accumulator reg with value 0xb7
- The # symbol indicates that this is an immediate (in `ca65` assembler)

Addressing Modes

- **Absolute**: memory address included as operand in instruction
- E.g. `LDA $07A3` —> Load accumulator with contents of memory at `0x07A3`. (opcode 0xa3)
- Note: If page number is zero, this means a “zero page” reference. Uses different opcode!
- E.g. `LDA $F7` —> `LDA $00F7` (opcode 0xa5)

Addressing Modes

- **Implied**: no operand necessary, implied by instruction
- E.g. TAX instruction (transfer contents of accumulator to X register) (opcode 0xaa)

Addressing Modes

- **Indexed:** Use a base register (either `X` or `Y`) and add it to the address given as operand
- E.g. `LDA $075A, X` —> A <- Mem[X + 0x75A]
- Same zero page rules apply. Note that **most instructions only use `X` with zero page**!

Addressing Modes

- **Indirect:** Add a level of indirection to address operand. NOTE: only used for `JMP` instruction!

- E.g. `JMP ($07A5)` —> Jump to the address stored in the 2 bytes at address `$07A5`

- Original 6502 actually had a bug when using this addressing mode (we'll talk about this later)

Addressing Modes

- **Indexed Indirect:** Get target address indirectly
  - (e.g. `LDA ($10,X)`)
- Assume X has #4, this mode means our target address is in memory at address $14
  - A <- Mem[Mem[(imm.addr & 0xff) + X]]
- Address must be in zero page (it's a two byte instruction, and addresses will wrap around)
- Only works with X register

Addressing Modes

- **Indirect Indexed (I know):** Get a target address from memory and offset it with an index register
  - E.g. `LDA ($73), Y`
- Assume `Y` has `#4`, this means our target address is in memory at address `$73` (we'll add `Y` to the address after we fetch it)
- A <- Mem[Mem[imm.addr & 0xff] + Y]
- Only works with `Y`, similar zero page restrictions apply (to the immediate addr)

Interrupts

- **IRQ** - Maskable interrupt. When invoked, `PC` and `PS` stored on stack
- Further interrupts are disabled by the processor until handled
- Processor jumps to address of handler that is stored in `0xFFFE` (2 bytes). Handler (likely in ROM) returns with `RTI` instruction.
- IRQ is masked/unmasked with `CLI`/`SEI` instructions

Interrupts (contd.)

- **NMI** - Non-maskable interrupt. Same sequence, but processor jumps to handler addr stored at `0xFFFA`. This interrupt can't be disabled!

Interrupts (contd.)

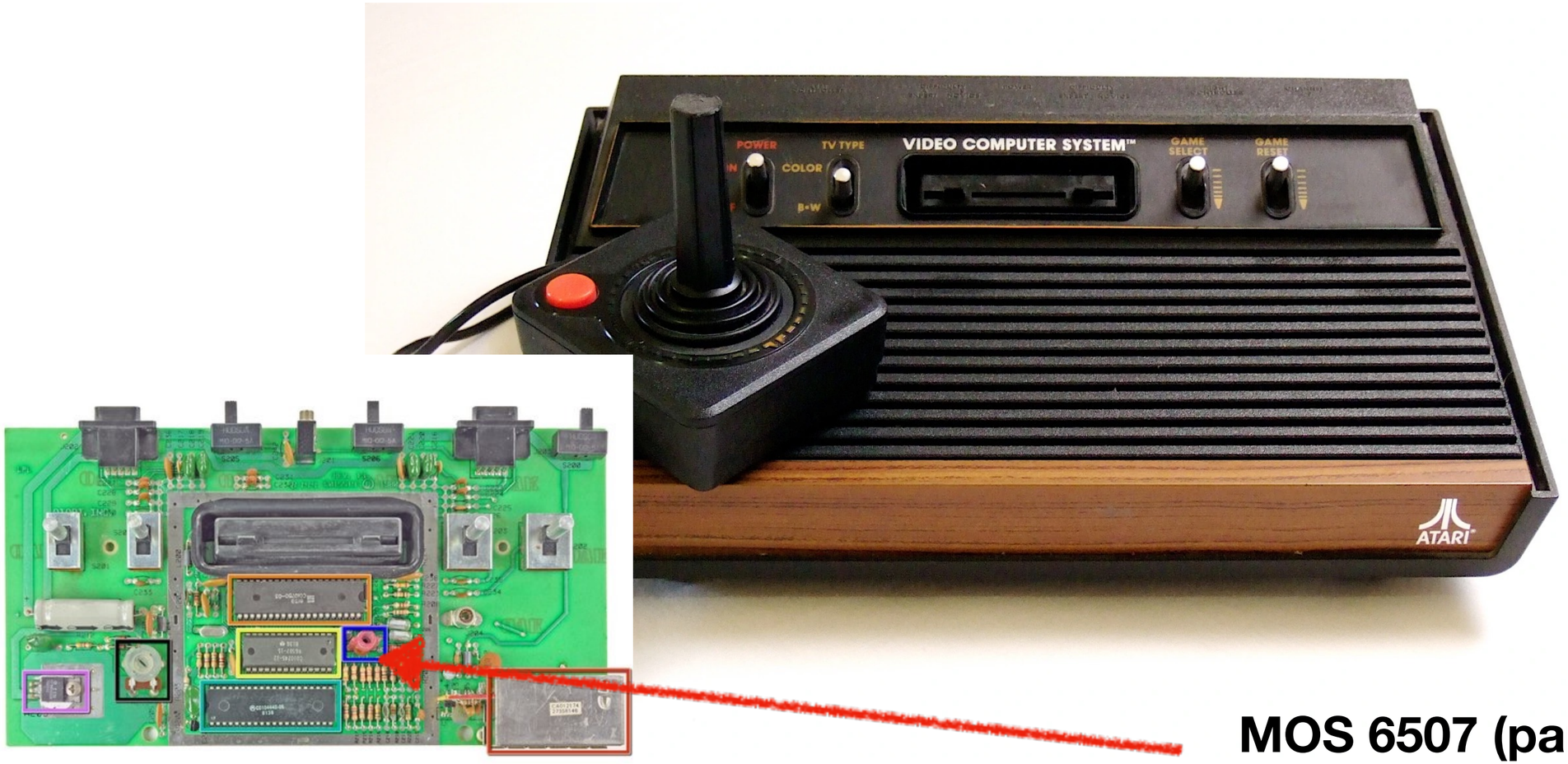
- **BRK** - Software interrupt. Same operation, but `B` flag is set in `PSW` stored on stack. CPU fetches from `0xFFFE` (same vector as IRQ!)

Interrupts (contd.)

- **RESET** - system reset. Nothing pushed on stack, fetch at vector `0xFFFC`, otherwise same

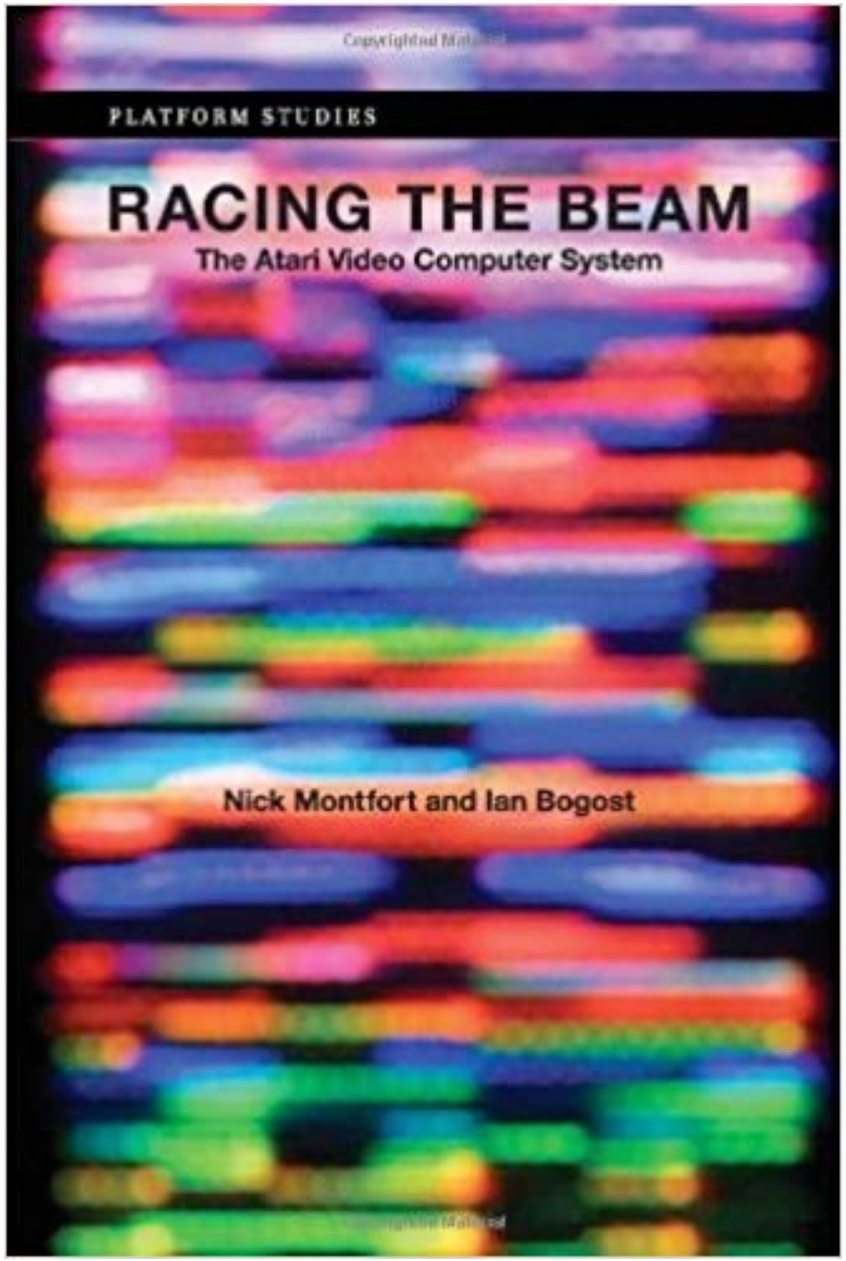
Notable 6502 systems

Atari VCS



MOS 6507 (pared down 6502)

Apple II


PLATFORM STUDIES
RACING THE BEAM
The Atari Video Computer System
Nick Montfort and Ian Bogost

Commodore 64

Famicom/Nintendo Entertainment System


Nintendo
ENTERTAINMENT SYSTEM
POWER
RESET