

Cody Anderson

Ben Nollan


Morgan Skrabut

Ryan Price

HERCULES

ECE360L
ECE460L

Prof. Thomas

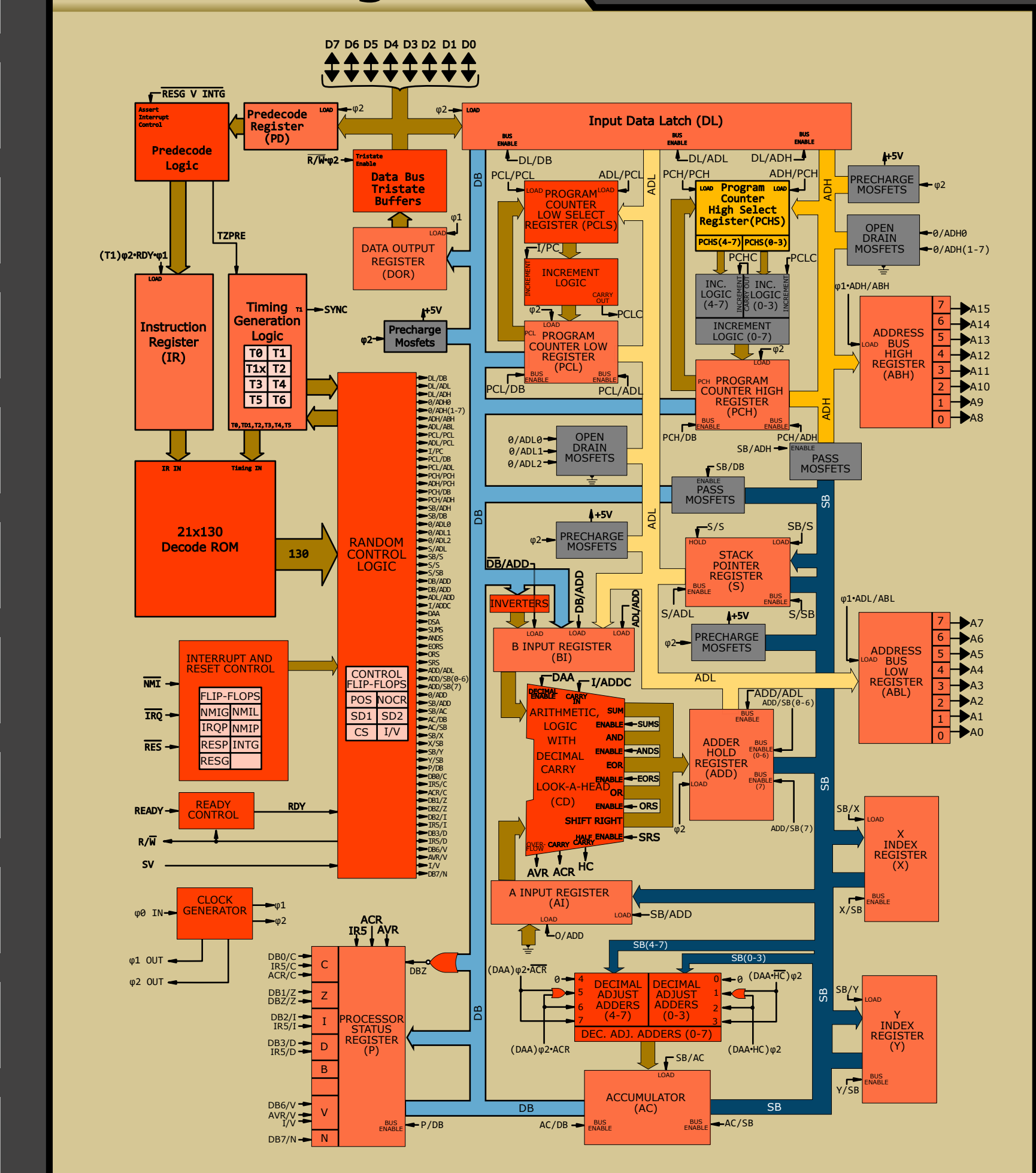


DigiPen
INSTITUTE OF
TECHNOLOGY

An FPGA Implementation of a Nintendo Entertainment System

Central Processing Unit (CPU)

Block Diagram



ALU

The 6502 ALU is made entirely of combinational logic. This means that every part of the ALU is active at all times, there are no clock signals directing operation.

The ALU is fed values from two different registers, called the A and B registers, whose only purpose is to feed values to the ALU. There are four basic operations that are used to perform every instruction: AND, OR, XOR, and ADD. The values of the A and B registers are ran though four different circuits simultaneously, each of which performs one of those operations. Afterwards, the result from the desired operation is selected by a multiplexer, and the other three are ignored. This result is then clocked into the ADDER HOLD register, whose purpose is to hold the result of the ALU operation. From there the result is sent to where it is needed by the ADDRESS LOW BUS or the SYSTEM BUS.

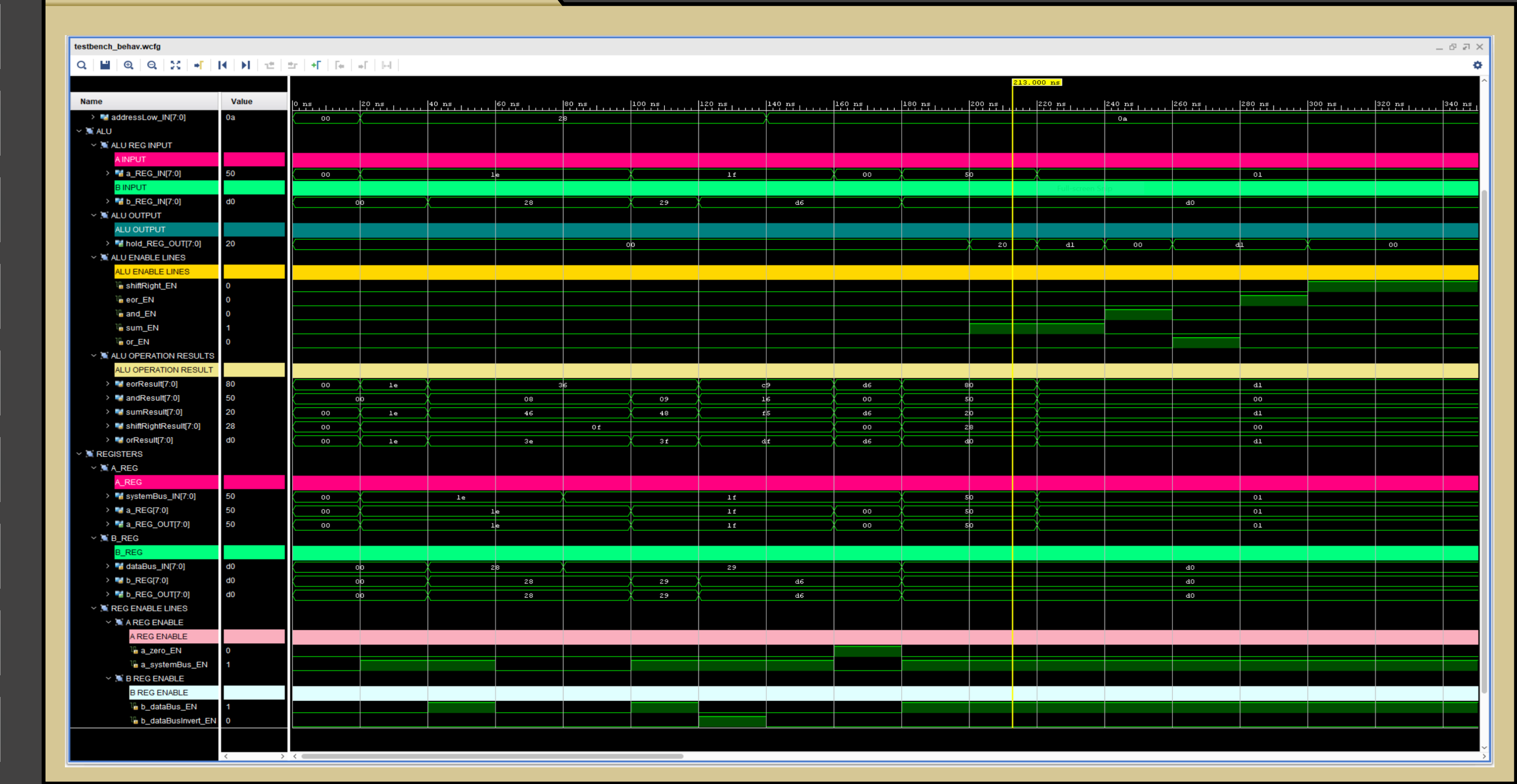
6502

The MOS 6502 is an 8-bit microprocessor designed by MOS Technology in 1975. What set the 6502 apart from other microprocessors of the time was it's extremely low cost, starting at just \$25 (compare to the Zilog Z80 at \$65). This low cost led to the chip's adoption for many mass produced products, including the Atari 2600, Commodore 64, Apple II, and the Nintendo Entertainment System.

Registers

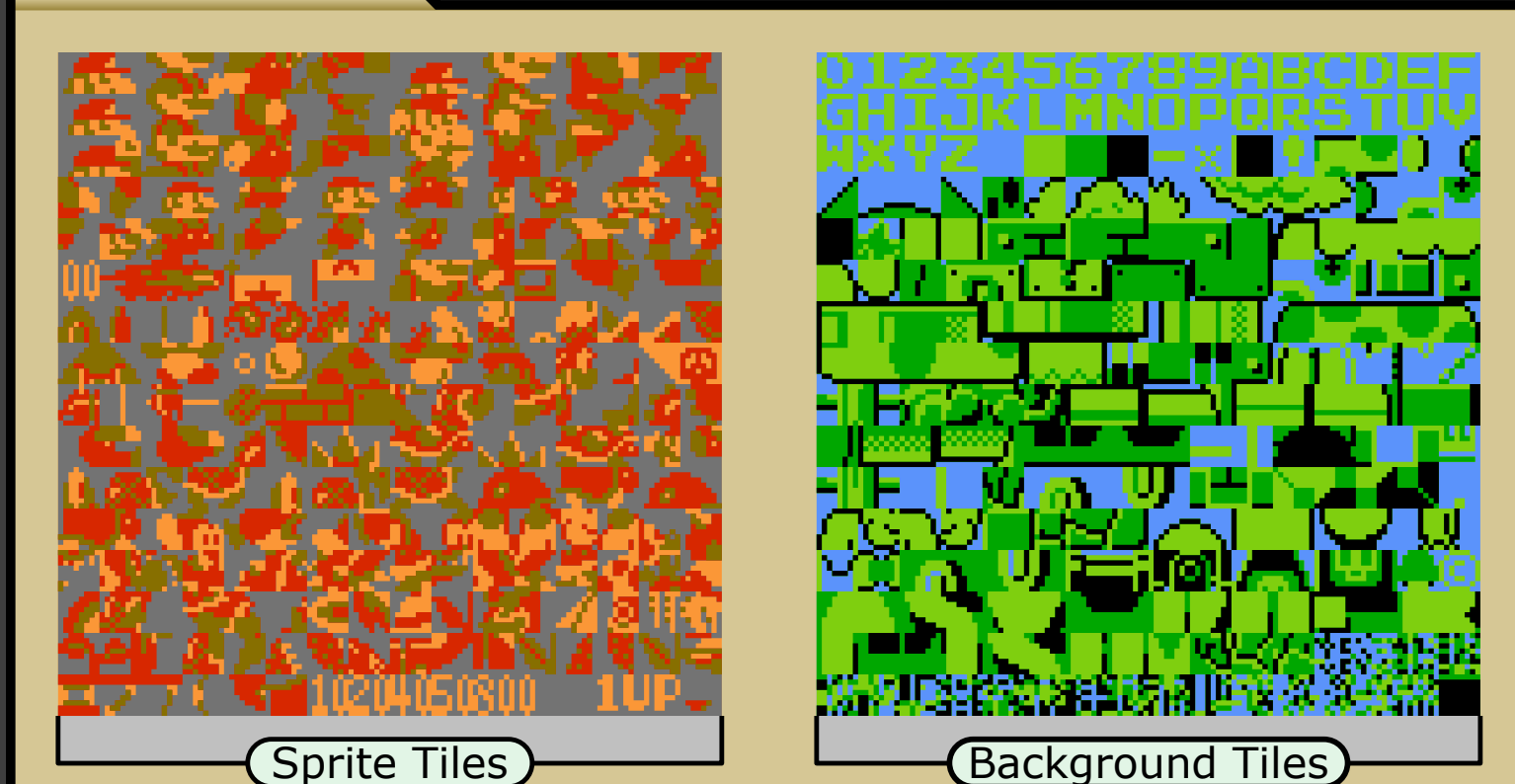
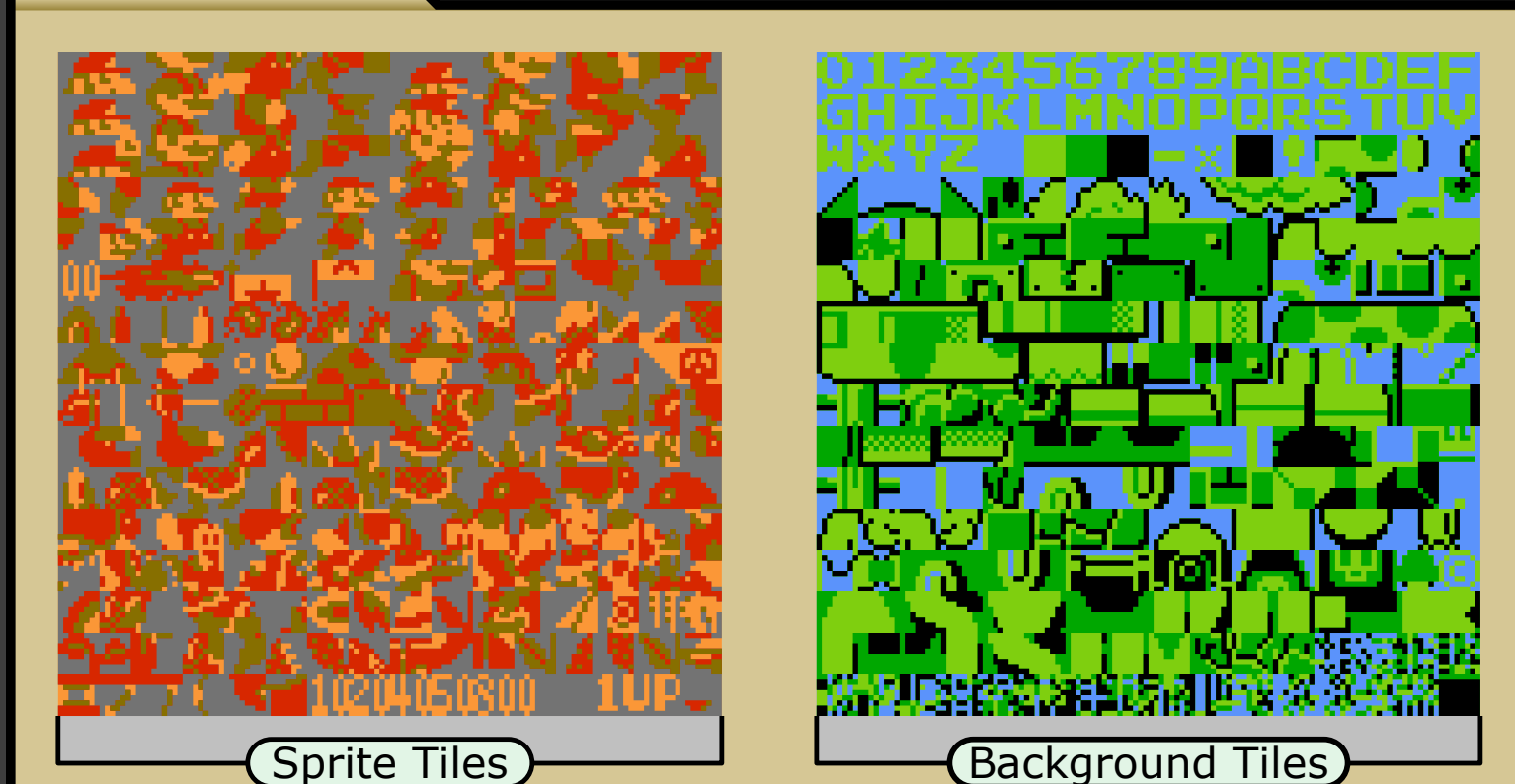
- Address High/Low Regs: Holds values to send out of CPU to memory.
- A/B Regs: Holds values to input into ALU.
- Accumulator: Used to perform successive arithmetic operations.
- Stack Pointer Reg: Holds stack pointer address.
- X/Y Regs: General purpose for use by program.
- PC High/Low Regs: Holds upper/lower 8-bits of program counter.
- Pre-Decode logic Reg: Holds instruction to send to decode logic.

ALU & Data Verification


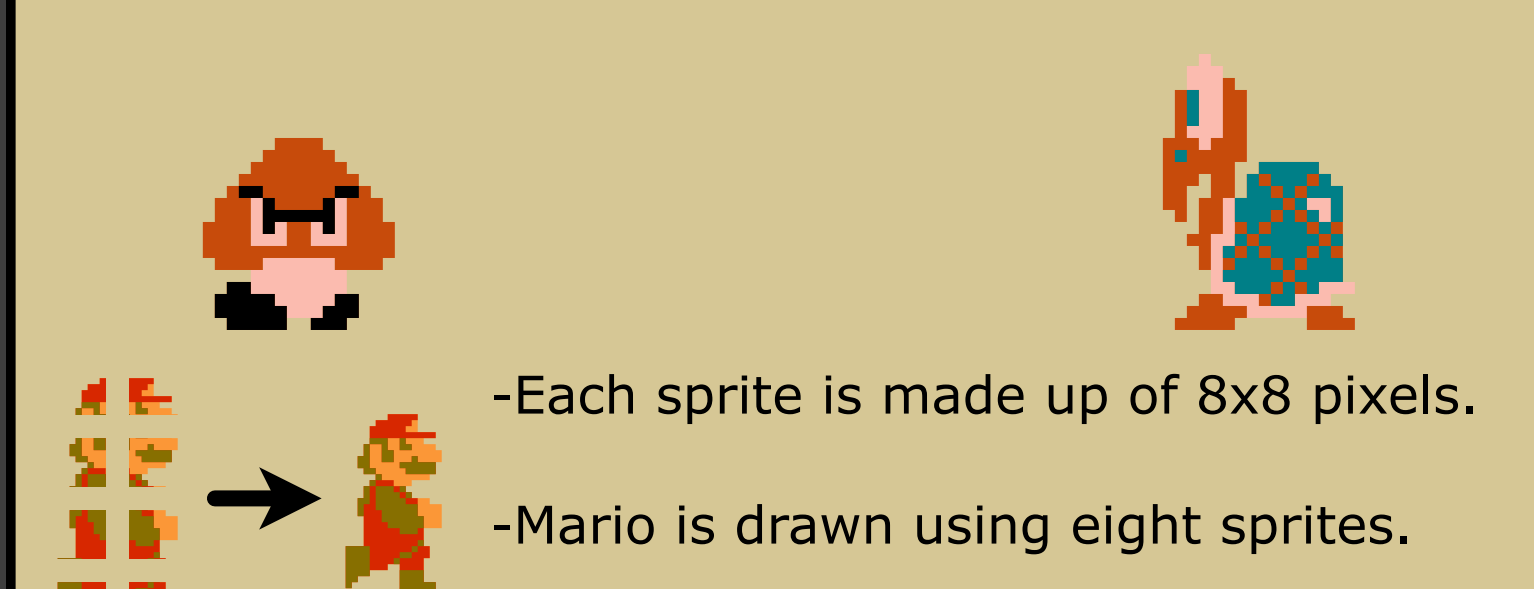


Picture Processing Unit (PPU)

ROM



Sprite Rendering



The diagram illustrates how a character is composed of multiple sprites. On the left, there is a 4x4 grid of 16 small 8x8 pixel sprites. These sprites represent different parts of Mario: his head, torso, arms, legs, and shoes. An arrow points from this grid to a larger, assembled 32x32 pixel sprite of Mario on the right. The assembled Mario is composed of these individual 8x8 pixel sprites arranged to form his full body.

- Each sprite is made up of 8x8 pixels.
- Mario is drawn using eight sprites.

 -Mario is drawn using eight sprites.

- All sprites are checked to see if they will appear on the next line.

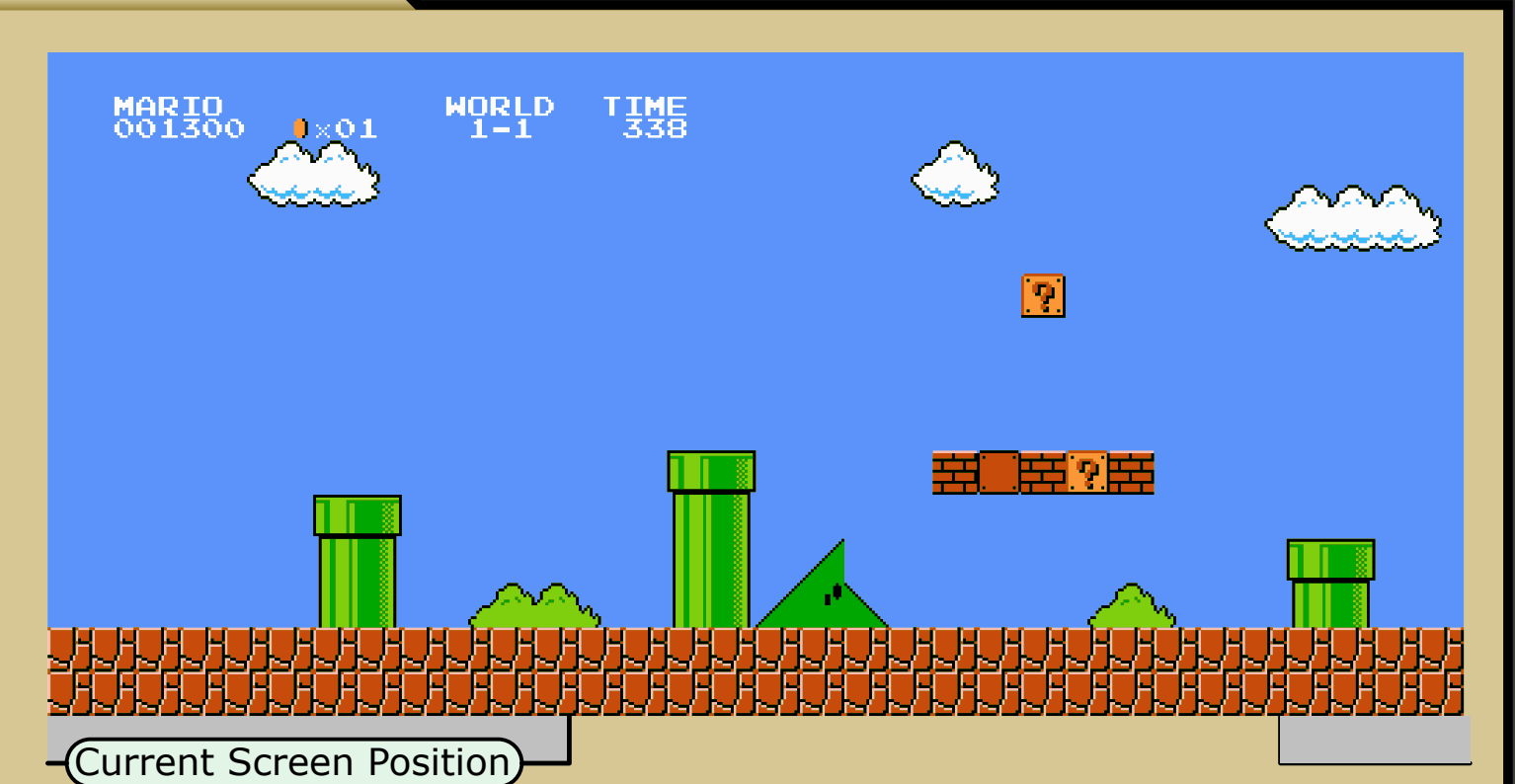
- The first eight are loaded in for drawing and tile data is fetched.

- These sprites' x positions are decremented until zero.

- The content of the tile data regs are bitshifted out to the screen.

- Multiple sprites are resolved using a priority encoder.

RAM



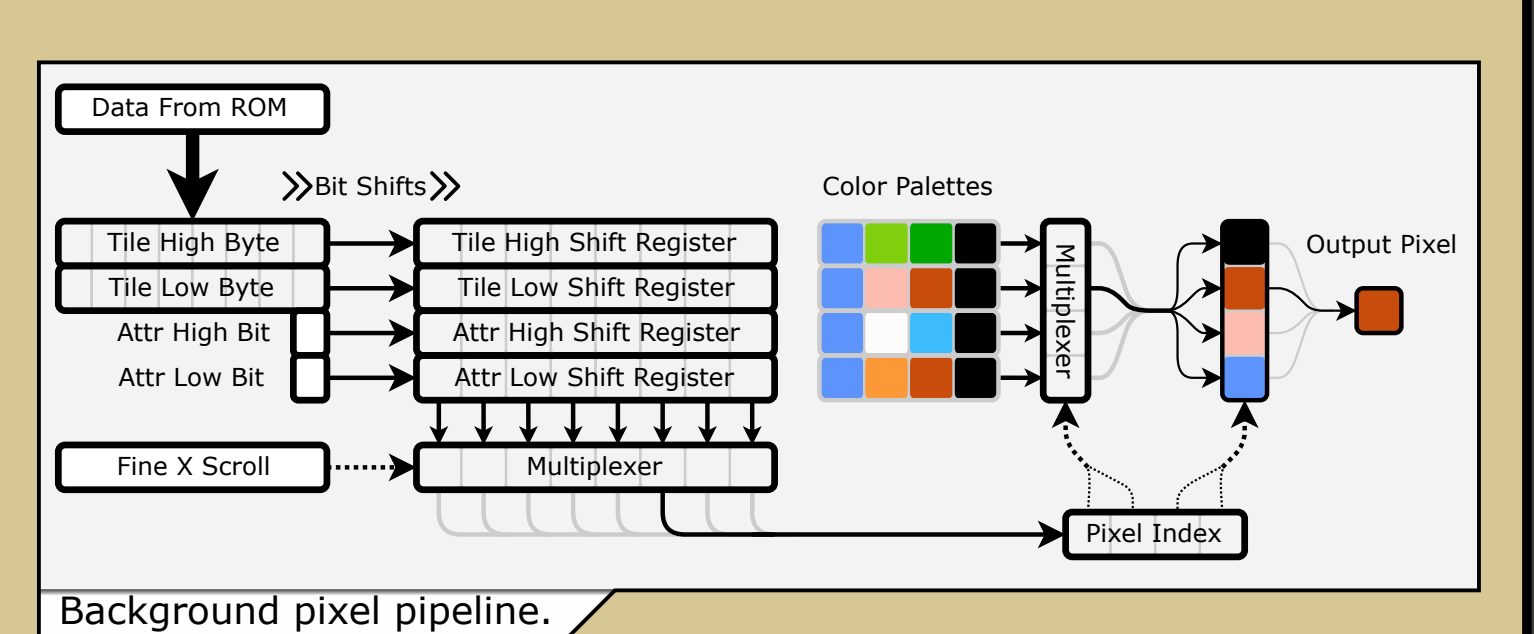
Background Rendering

- The current tile index is read from the RAM.

- This index is then used to grab the tile data from ROM.

- These Values are then passed into shift registers.

- They are selected by the adjustment of the X scrolling.



Final Output

