Olivier Mattmann, COLIN FINGERLIN, MATíAS carballo González

[Company address]

NES-EMULATOR

CAOS-PROJECT

# Table of Contents

[INHALTSVERZEICHNIS 1](#_Toc54905779)

[HARDWARE 2](#_Toc54905780)

[CPU 2](#_Toc54905781)

[Beschreibung 2](#_Toc54905782)

[APU 3](#_Toc54905783)

[Beschreibung 3](#_Toc54905784)

[PPU 3](#_Toc54905785)

[Memory 3](#_Toc54905786)

[Bus 3](#_Toc54905787)

# HARDWARE

The NES console is made up of the CPU, the APU (Audio Processing Unit), the PPU (Picture Processing Unit), a total 4kB of RAM, and of course the ROM on the game cartridge. These components communicate with each other using Buses, and are also made up of multiple subcomponents.

## 

## CPU

### Description

The CPU is a 6502 microprocessor, that runs with a **Clockspeed of 1,79MHz**. The CPU has 6 Register und 6 used Flags, that are described below.

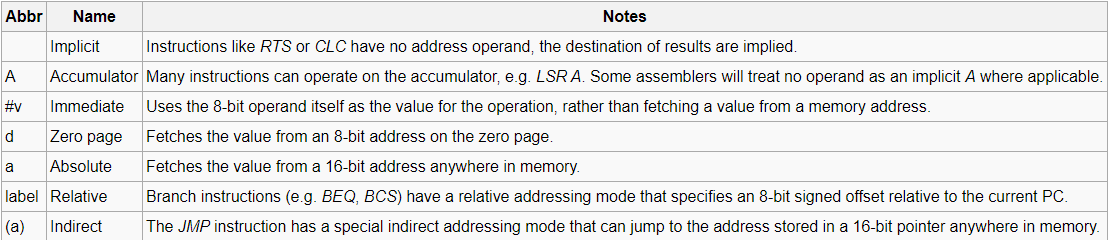
|  |  |  |
| --- | --- | --- |
| **Register** | **Size** | **Description** |
| PC (Programm Counter) | 16 Bit | Points to the next address in memory |
| S (Stack Pointer) | 8 Bit | Points to the next free Address in the stack memory |
| P (Processor status) | 8 Bit | Each bit represents a flag (Explained below) |
| A (Accumulator) | 8 Bit | Main register for CPU operations |
| X (Index Register X) | 8 Bit | Main register for data addressing |
| Y (Index Register Y) | 8 Bit | Secondary register for data addressing |

|  |
| --- |
| Flags |
| N (Negative) |
| V (Overflow) |
| 1 (immer 1) |
| B (Break) |
| D (Decimal Mode) |
| I (Interrupt Disable) |
| Z (Zero) |
| C (Carry) |

For the CPU to work the instructions and behaviour of the CPU must be emulated. For the implementation of the opcodes we will use different official and non-official collections of the opcodes.[[1]](#footnote-1)

#### Addressing Modes[[2]](#footnote-2)

Instructions can be used in combination with different addressing modes. That why we must implement all of the possible addressing modes in our CPU



#### 

#### Memory map[[3]](#footnote-3)

|  |  |  |
| --- | --- | --- |
| Address range | Size | Device |
| $0000-$07FF | $0800 | 2KB internal RAM |
| $0800-$0FFF | $0800 | [Mirrors](http://wiki.nesdev.com/w/index.php/Mirroring) of $0000-$07FF |
| $1000-$17FF | $0800 |
| $1800-$1FFF | $0800 |
| $2000-$2007 | $0008 | [NES PPU](http://wiki.nesdev.com/w/index.php/PPU_registers) registers |
| $2008-$3FFF | $1FF8 | Mirrors of $2000-2007 (repeats every 8 bytes) |
| $4000-$4017 | $0018 | [NES APU](http://wiki.nesdev.com/w/index.php/APU) and [I/O registers](http://wiki.nesdev.com/w/index.php/2A03) |
| $4018-$401F | $0008 | APU and I/O functionality that is normally disabled. See [CPU Test Mode](http://wiki.nesdev.com/w/index.php/CPU_Test_Mode). |
| $4020-$FFFF | $BFE0 | Cartridge space: PRG ROM, PRG RAM, and [mapper](http://wiki.nesdev.com/w/index.php/Mapper) registers (See Note) |

We use memory in the form of a char Array because it makes it easy to address using the cartridge addressdata.

#### Interrupts[[4]](#footnote-4)

#### Pins[[5]](#footnote-5)

## APU

### Beschreibung

Die APU kümmert sich um das Processing vom Audio.

## PPU

## Memory

## Bus

# Implementation Plan

All our addresses and data are represented by unsigned integers of the needed size. More specifically by the types **uint8\_t and uint16\_t**.

As we progressed through the project it became apparent that we had to use unsigned integers, so reading and writing data, addressing and reading flags and registers does not bug.

## CPU

### Registers

Registers are represented by uint8\_t variables, except for the PC, which is a uint16\_t. These variables are public and referenced directly.

### Opcodes

Opcodes give us a key that tells us three pieces of information[[6]](#footnote-6):

* Addressing Mode
* Instruction
* Clock Cycles

The addressing modes and instructions are implemented as functions, and clock cycles as an Integer. These three parts are put together in a structure, which is then put into an array and referenced by the opcode. This array has a length of 256 elements. The functions are referenced to by function pointers.

## Memory

The Memory is part of the bus because it can only be accessed by the bus. It consists of an uint8\_t array which currently has a length of 0xFFFF. The elements in the array are accessed by the read/write methods of the bus, which gets an address to read/write that represents the index in the array.

## Bus

Read/Write functions for 8 Bit data with 16 Bit addresses as parameter.

1. http://wiki.nesdev.com/w/index.php/CPU\_unofficial\_opcodes [↑](#footnote-ref-1)
2. http://wiki.nesdev.com/w/index.php/CPU\_addressing\_modes [↑](#footnote-ref-2)
3. http://wiki.nesdev.com/w/index.php/CPU\_memory\_map [↑](#footnote-ref-3)
4. http://wiki.nesdev.com/w/index.php/CPU\_interrupts [↑](#footnote-ref-4)
5. http://wiki.nesdev.com/w/index.php/CPU\_pin\_out\_and\_signal\_description [↑](#footnote-ref-5)
6. http://www.obelisk.me.uk/6502/instructions.html [↑](#footnote-ref-6)