Simple 8-Bit Microprocessor

Datasheet & Instruction Set Architecture

Documentation

February 1, 2026

1 Overview

The **CPU** is a simple 8-bit accumulator-based microprocessor. It features a Harvard architecture with separate internal storage for instructions and data. The processor operates on an 8-bit wide datapath with a 4-bit addressable memory space.

2 Signal Description

The CPU interface consists of standard clock and reset controls, along with debug/visibility outputs for the Program Counter, Accumulator, and Halt state.

Table 1: Pinout Description

Signal	Width	Direction	Description
`clk`	1	Input	System Clock. Operations occur on the rising edge.
`rst`	1	Input	Asynchronous Reset. Resets PC, Accumulator, and Halt status to 0.
`pc`	8	Output	Current Program Counter value.
`acc`	8	Output	Current Accumulator (ACC) value.
`halt`	1	Output	Status flag. Asserted high when a HLT instruction is executed.

3 Programmer's Model

3.1 Registers

- **Accumulator (ACC):** 8-bit general-purpose register used for arithmetic and logic operations: .
- **Program Counter (PC):** 8-bit register holding the address of the next instruction. Note: Only the lower 4 bits are used to address the internal 16-byte memory: 2, 4.

3.2 Memory Organization

The CPU contains two distinct internal memory arrays:

- **Instruction Memory (IMEM):** 16 bytes. Stores the program code: 2.
- **Data Memory (DMEM):** 16 bytes. Stores variables and data: 3.

4 Instruction Set Architecture

Instructions are 8 bits wide. The upper 4 bits constitute the Opcode, and the lower 4 bits constitute the Operand (or Address): 7.

Instruction Word (8 Bits)							
7	6	5	4	3	2	1	0
Opcode				**Operand / Address**			

4.1 Instruction Summary

Table 2: Instruction Set. (Mne = Mnemonic; OC = Opcode)

Mne	OC	Operand	Operation	Description
NOP	0x0	N/A	None	No Operation: 7.
LDA	0x1	Address	$ACC \leftarrow DMEM[addr]$	Load Accumulator from Data Memory: 8.
STA	0x2	Address	$DMEM[addr] \leftarrow ACC$	Store Accumulator to Data Memory: 9.
ADD	0x3	Address	$ACC + DMEM[addr]$	Add Data Memory to Accumulator: 10.
SUB	0x4	Address	$ACC - DMEM[addr]$	Subtract Data Memory from Accumulator: 1.
LDI	0x5	Immediate	$ACC \leftarrow Imm$	Load 4-bit Immediate into Accumulator: 12.
JMP	0x6	Address	$PC \leftarrow Address$	Unconditional Jump: 13.
HLT	0xF	N/A	$Halt \leftarrow 1$	Stop execution (freezes PC): 14.

5 Programming Example

The following SystemVerilog snippet demonstrates how to initialize the memory for a test program.
Program Logic: Load immediate 5, Add 3 (from memory), Store result, Load another value, Subtract, Store, then Halt.

```
// Initialize instruction memory with a simple program
// Program: Load 5, Add 3, Store result, Halt

dut.imem[0] = 8'h55;  // LDI 5 (Load immediate 5)
dut.imem[1] = 8'h31;  // ADD [1] (Add value at dmem[1])
dut.imem[2] = 8'h22;  // STA [2] (Store to dmem[2])
dut.imem[3] = 8'h13;  // LDA [3] (Load from dmem[3])
dut.imem[4] = 8'h42;  // SUB [2] (Subtract dmem[2])
dut.imem[5] = 8'h24;  // STA [4] (Store to dmem[4])
dut.imem[6] = 8'hF0;  // HLT (Halt)

// Initialize data memory
dut.dmem[1] = 8'h03;  // Value 3
dut.dmem[3] = 8'h0A;  // Value 10
```