Specification Document

Omkar Girish Kamath

June 1, 2023

Contents

1 Description

The Tinycpu is a *8-bit* Instruction Set Architecture, *8-bit* wide databus CPU built on a custom designed 2 layer PCB board with a seperate daughter PCB board for the Power supply.
The whole board is encased in a box for insulation and safety purposes, it has a passive cooling system. The board has a 8 x 8k EEPROM device acting as the data and instruction memory for the CPU which can be programmed using programmer device.
It displays outputs using Hex Displays. Buttons and switches on its externals can be used to dynamically provide inputs such as reset signals, and address signals for the hex display output. The software version is completely built in verilog and python, tested and verified with a diverse set of Assembly codes.

2 Instruction Set of the Processor

2.1 Arithmetic and Logical Instructions

2.1.1 AND

Opcode: 0000
Instruction Description: Bit-wise "ANDs" values stored in register A and register B and stores the result in register A.

rA & rB ->rA

2.1.2 OR

Opcode: 0001
Instruction Description: Bit-wise "ORs" values stored in register A and register B and stores the result in register A.

rA | rB -> rA

2.1.3 INV

Opcode: 0010
Instruction Description: Bit-wise "Inverts" values stored in register A and stores the result in register A.

not(rA) -> rA

2.1.4 ADD

Opcode: 0011
Instruction Description: "ADDs" values stored in register A and register B and stores the result in register A.

rA + rB -> rA

2.2 Memory Access and Immediate Instructions

2.2.1 LDI

Opcode: 0100
Instruction Description: Left Shifts value in register A and the 4-bit immediate value provided with the instruction is stored at the lower 4-bits of rA.

$$\mathbf{rA}[7:4] \ <- \ rA[3:0]$$

$$\mathbf{rA}[3:0] \ <- \ imm[3:0]$$

2.2.2 LDM

Opcode: 0101
Instruction Description: Loads the value stored at memory address [*register M*] in register A.

2.2.3 STM

Opcode: 0110
Instruction Description: Stores the value stored in register A at memory address [*register M*].

2.3 Memory Access and Immediate Instructions

2.3.1 SWAB

Opcode: 1000
Instruction Description: Swaps the values stored in register A and register B.

rA <-> rB

2.3.2 SWMB

Opcode: 1001
Instruction Description: Swap the values stored in register M and register B.

rM <-> rB

2.3.3 CPPA

Opcode: 1010
Instruction Description: Copies the value stored in register P (program counter) to register A.

rA <- rP

2.3.4 CPAM

Opcode: 1011
Instruction Description: Copies the value stored in register A to register M.

rM <- rA

2.4 Unconditional and Conditional Jumps

2.4.1 JU

Opcode: 1100
Instruction Description: Copies value stored in register M to register P (program counter).

rM -> rP

2.4.2 JE

Opcode: 1101
Instruction Description: Copies value stored in register M to register P (program counter) if values stored in register A and register B are equal.

rM -> rP if (rA = rB)

2.4.3 JL

Opcode: 1110
Instruction Description: Copies value stored in register M to register P (program counter) if value stored in register A is lesser than value stored in register B.

rM -> rP if (rA < rB)

2.4.4 JG

Opcode: 1111
Instruction Description: Copies value stored in register M to register P (program counter) if value stored in register A is greater than value stored in register B.

rM -> rP if (rA > rB)

3 Block Diagram

4 Software Design and Verification Tools

4.1 Emulator

It is a reference model of the processor written in Python. It is used to verify the correctness of the other models such as the Behavioural model, the Structural model and the 74 series structural model.

4.2 Assembler

It is a utility which converts Assembly level instructions (.s) files and converts them into machine code level files which are then used to feed to the processor RAM using $readmemh.
It is completely written in Python.

4.3 Makefile

5 I/O Interface Description

6 Behavioural Model

A behavioural model of the processor designed fully in Verilog. The behavioral model refers to a modeling style that describes the behavior or functionality of a digital circuit using procedural code. It focuses on how the circuit operates and behaves rather than its physical implementation.

In a behavioral model, you describe the circuit's functionality using high-level constructs, such as if-else statements, loops, and assignments, which resemble programming languages like C or Java. This allows you to specify the desired behavior of the circuit without getting into the details of the circuit's structure or timing.

7 Structural Model

The structural modeling style involves describing a digital circuit as an interconnected network of lower-level components or modules. It focuses on the physical structure of the circuit, specifying how the components are connected to form the desired functionality.

In structural modeling, you instantiate and connect instances of predefined or user-defined modules to create the circuit. Each module represents a lower-level component or a subcircuit, and its internal structure is described separately. The connections between modules are made using signals or wires.

8 74 series Structural Model

It is similar to the structural model but in this model we create modules of the ICs that we require after observing the approximate BOM and place them as per their functionalities in the structural model such as the IC-SN7408 is a quad 2-input AND gate IC, so we replace AND gates in the structural model with instantiated modules of the mentioned ICs. The result is we are able to fully express the structural design in terms of ICs of 74 series and similar kind. This is the 74 series Structural Model that we require.

9 SRAM / EEPROM

9.1 Components Required

9.1.1 IC

The IC required is the **AT28C64B**. Its a 32-pin PLCC type package, J type. We need a 32-pin PLCC to DIP package socket for the above IC.