ECE 211P: Electronics Lab Project
Design and Implementation of an 8-bit TTL CPU

Team Report

September 17, 2025

Team Members

Eluri Sri Naga Sairam Kapish	IMT2024027
Kandagatla Venkata Rohith Sai	IMT2024042
Nuthigattu Yagneswar Gupta	IMT2024063
Garaga Karthikeya	IMT2024073
GVN Mokshagna	BT2024234
Gonuguntla Lohith Prapoorna Chandra	BT2024260

Abstract

This project focuses on building an educational 8-bit CPU using TTL-compatible ICs on breadboards. The CPU executes instructions independently, with all registers, ALU, and control logic implemented using discrete ICs. A Raspberry Pi (or Pi Pico) is used as an external instruction source, feeding programs directly to the CPU for execution.

The CPU datapath includes dedicated registers (Accumulator, B, Instruction Register, Output Register, Program Counter, MAR), an ALU supporting addition, subtraction, bitwise AND/OR, and Zero/Carry flags for conditional branching. Outputs are displayed on LEDs and 7-segment displays for real-time observation.

This setup provides a fully functional 8-bit CPU prototype that is simple, transparent, and highly educational, allowing step-by-step observation of instruction execution at the hardware level.

Why 8-bit Registers?

The CPU datapath is designed to be 8 bits wide, matching the instruction/data width provided by the Raspberry Pi. By explicitly constructing registers from discrete ICs (e.g., 74HC173), learners can observe how registers interact with the bus. Discrete registers make debugging and visualization easier, ensuring every transfer is visible during execution.

The Role of Transceivers

The 74HC245 octal bus transceiver is essential for safe and reliable bus communication. It ensures that only one device drives the bus at a time, avoiding short circuits or logic conflicts. It also provides directional control, allowing smooth transfer of data between registers, the ALU, and the instruction source (Raspberry Pi).

Methodology

The project is executed in stages:

1. **Assembler Development:** Write assembly programs and generate machine code using Python.
2. **Instruction Loading:** Use a Raspberry Pi (or Pi Pico) to provide instructions directly to the CPU via GPIO.
3. **Register and Datapath Construction:** Build A, B, IR, OUT, MAR, and PC using 74HC173 registers, 74HC161 counters, and bus transceivers (74HC245).
4. **ALU Implementation:** Realize `ADD`, `SUB`, `AND`, `OR` using 74HC283 adders and logic gates.
5. **Control Unit:** Construct a hardwired FSM with 74HC138 decoders and multiplexers to sequence fetch–decode–execute stages.
6. **Testing:** Run example assembly programs and observe outputs on LEDs/7-segment displays to validate correctness.

Instruction Set Architecture (ISA)

The CPU supports a compact 12-instruction ISA.

Instruction	**Description**
LDA addr	Load accumulator from instruction source.
ADD addr	$A \leftarrow A + [addr]$, updates Carry/Zero.
SUB addr	$A \leftarrow A - [addr]$, updates Carry/Zero.
STA addr	Store accumulator into internal register/memory.
LDI imm	Load immediate value into A.
JMP addr	Unconditional jump.
JC addr	Jump if Carry flag is set.
JZ addr	Jump if Zero flag is set.
OUT	Output A to OUT register (LEDs/display).
HLT	Halt execution.
AND addr	$A \leftarrow A \,\&\, [addr]$, updates Zero.
OR addr	$A \leftarrow A \,\vert\, [addr]$, updates Zero.

Components Used

Component	Purpose	Qty
Raspberry Pi (or Pi Pico)	Acts as programmable instruction source	1
74HC161/163	Program Counter	1–2
74HC173	Registers (A, B, IR, OUT)	4–5
74HC245	Bus Transceivers	2–3
74HC283	Adders (ALU core)	1
74HC138, 74HC157	Control Logic	1 each
Breadboards	Circuit assembly	Variable
Resistors	Pull-ups, current limiting	Variable
Capacitors	Decoupling, stability	Variable
LEDs	Output/status indicators	Variable
7-segment displays	Display output	1–2
Switches/push-buttons	Inputs, reset, clock	Variable
Wires/jumpers	Interconnections	Variable
5V Power Supply	TTL/CMOS power	1

Demonstration Plan

To showcase the working 8-bit TTL CPU, the following steps are proposed:

1. **Setup:** Power on the CPU and Raspberry Pi. Ensure all ICs, bus transceivers, registers, and LEDs/7-segment displays are properly connected.
2. **Program Loading:** Use the Raspberry Pi to send a small test program (e.g., addition or counting) to the CPU instruction input.
3. **Execution:** Observe the CPU fetching, decoding, and executing instructions cycle by cycle.
4. **Outputs:** Monitor the accumulator and output registers via LEDs and 7-segment displays to verify correct results.
5. **Conditional Operations:** Demonstrate jumps and flags (Zero and Carry) by running programs that use conditional instructions (JC, JZ).

Budget Estimate

Only non-lab items are costed. Standard items (breadboards, resistors, LEDs, etc.) are available in the lab.

Component	Approx. Cost (INR)	Notes
Extra 74xx ICs (spares)	200–300	Backup set