8-Bit CPU Implementation Report

Contents

1 Project Overview

In this project, we implement the working of a Central Processing Unit (CPU) based on the architecture and hardware design of an 8-bit computer. The hardware is implemented on a breadboard using TTL logic chips, with an Arduino used for the control and memory units.

1.1 System Specifications

- **Control Signals:** A total of 20 control signals are used to coordinate the operation of the CPU components.
- **Memory:**
  - Data Memory: 2^4 locations
  - Instruction Memory: 2^4 locations
  - Each instruction is 8 bits wide.
- **ALU Implementation:** The Arithmetic Logic Unit (ALU) is implemented with TTL chips.
- **Bus System:** A common 8-bit data bus is used for communication among all CPU components.
- **Control Wiring:** Separate wiring connects control signals from the Arduino to all hardware components.
- **Data Flow Control:** The control signals are connected to bus transceivers, managing the exchange of data between components and the common bus.

1.2 CPU Components

- **MAR (Memory Address Register):** Stores the address of the value required for instruction execution.
- **IR (Instruction Register):** Holds the instruction currently being executed.
- **A Register:** Stores results from the ALU and other operations, such as LDI.
- **Buffer Register:** Stores intermediate values temporarily.
- **ALU:** Performs arithmetic and logic operations using combinational circuits.
- **PC (Program Counter):** Holds the address of the next instruction to be fetched.
- **Memory Unit:** Used to store both data and instructions. The Arduino handles memory access and data transfer.
- **Output Register (OUT):** Holds final results to be displayed.
- **Control Unit:** Implemented using three Arduinos that manage:
  - Generation of control signals
  - Instruction decoding
  - Clock synchronization
- **Clock:** Provides timing pulses for synchronized operation.
- **Flag Registers:**
  - **Carry Flag:** Set to 1 if addition produces a carry.
  - **Zero Flag:** Set to 1 if result is zero.

1.3 List of Control Signals

1. MAR Read
2. MAR Write
3. IR Read
4. IR Write
5. A Read
6. A Write
7. Buffer Read
8. JCZ (Jump if Carry Zero)
9. JSZ
10. Unconditional Jump
11. Memory Address Read
12. Data Read/Write (based on direction)
13. PC Write
14. OUT Enable
15. CU Read (Instruction Decode)
16. CU Write (Send Address/Data to Bus)
17. ALU Operation - ADD
18. ALU Operation - SUB
19. ALU Operation - OR
20. ALU Operation - AND

1.4 Instruction Opcodes

The following 4-bit opcodes are used for their respective instructions:

Instruction	Opcode
LDA	0000
ADD	0001
SUB	0010
STA	0011
LDI	0100
JMP	0101
JC	0110
JZ	0111
OUT	1000
HLT	1001
AND	1010
OR	1011

1.5 Clock Cycles and Signal Flow

Common 4 Cycles for All Instructions

1. **1st Clock Cycle:** MAR Read, PC Write
2. **2nd Clock Cycle:** MAR Write, Memory Address Read
3. **3rd Clock Cycle:** Data Write/Read (to bus), IR Read
4. **4th Clock Cycle:** IR Write, CU Read

For LD (Load) Instruction

- **5th cycle:** CU Write, MAR Read
- **6th cycle:** MAR Write, Memory Address Read
- **7th cycle:** Data Write/Read (from memory to bus), A Read (load to accumulator)

For LDI (Load Immediate) Instruction

- **5th cycle:** CU Write (places immediate value from IR onto bus), A Read (loads value from bus)

For ST (Store) Instruction

- **5th cycle:** CU Write, MAR Read
- **6th cycle:** MAR Write, Memory Address Read
- **7th cycle:** A Write (to bus), Data Write/Read (store into memory)

For ADD Instruction

- **5th cycle:** CU Write, MAR Read
- **6th cycle:** MAR Write, Memory Address Read
- **7th cycle:** Data Write/Read (from memory to bus), B Read (load operand)
- **8th cycle:** ALU Write, A Read (store result)

For OR and AND Instructions

ALU Operation codes:

- ADD: ALU Operation = 00
- SUB:. ALU Operation = 01
- OR: ALU Operation = 10
- AND: ALU Operation = 11

SUB, OR, AND are same as ADD instruction.

For OUT Instruction

- **5th cycle:** OUT Read (ALU Operation = XX)

For Jump Unconditional (JU), JSZ, and JCZ Instructions

Unconditional Jump (JU):

- **5th cycle:** MAR Read, CU Write
- **6th cycle:** Perform Unconditional Jump (load new address)
- **Logic Equation:** $L_D = U_J \& [\overline{B_A + J_C}] \& [C_S + X_S]$

JSZ Instruction:

- **5th cycle:** MAR Read, CU Write
- **6th cycle:** Jump if signal active
- **Conditions:** $U_J = 1$, $J_{C_s} = 1$

JCZ Instruction:

- **Note:** Works similarly to JSZ, performing jump when Carry or Zero flag is set.

Notes

- Data Write/Read is sent according to the direction of data, taken from bus or given to bus.
- All control signals are set to 0 till the clock cycles reach 8 if the instruction is done early than 8 clock cycles.