**Processor Microarchitecture Design Specification**

**Project: 8-Bit Processor Simulator**

**1. Introduction**

This document defines the **Processor Microarchitecture** for an 8-bit processor designed for **text-based communication** and **mobile money transactions**. The processor’s microarchitecture implements the previously designed Instruction Set Architecture (ISA), focusing on efficient instruction execution, low power consumption, and cost-effective implementation.

The microarchitecture’s design emphasizes modularity, with distinct **Control**, **Datapath**, and **Memory Interface** sections that work together to execute instructions efficiently.

**2. Microarchitecture Overview**

The microarchitecture consists of the following major components:

1. **Control Unit**: Decodes instructions and generates control signals.
2. **Datapath**: Executes instructions, processes data, and stores results.
3. **Memory Interface**: Facilitates communication with external memory.
4. **Instruction Fetch and Decode**: Manages instruction retrieval and interpretation.

**3. Components of the Processor Microarchitecture**

**Datapath**

The **Datapath** is the execution core, where data flows and computations occur. It consists of:

**Register Bank**

* **General-Purpose Registers**:
  + Eight 4-bit registers (R0–R3) for holding data and intermediate results.
* **Special-Purpose Registers**:
  + **Program Counter (PC)**: Holds the address of the next instruction.
  + **Accumulator (ACC)**: Stores results of arithmetic and logical operations.
  + **Stack Pointer (SP)**: Points to the top of the stack, supporting subroutine calls and temporary data storage.

**Arithmetic Logic Unit (ALU)**

* **Operations Supported**: ADD, SUB, INC, DEC, AND, OR, XOR, NOT.
* **Input/Output**:
  + Inputs: Two operands from the Register Bank.
  + Output: Result stored in the Accumulator (ACC).
  + Flags Register: Updates based on operation outcomes.

**Data Pathways**

* Data flows between:
  + Register Bank → ALU
  + ALU → Accumulator
  + Registers → Memory via the Memory Interface

**Control Unit**

The **Control Unit** manages instruction execution by generating control signals to coordinate Datapath and Memory Interface operations.

**Instruction Decoder**

* Decodes the **Opcode** from the Instruction Register (IR) and determines the required operation.
* Generates control signals for:
  + ALU operations (e.g., ADD, SUB, AND).
  + Data routing (e.g., Register Bank to ALU).
  + Memory access (e.g., read/write operations).

**Control Signals**

* **ALU Signals**: Specify the operation (e.g., addition, subtraction).
* **Register Signals**: Enable/disable register read/write operations.
* **Memory Signals**: Direct memory read/write operations.

**Memory Interface**

Facilitates data exchange between the processor and external memory.

**Interaction**

**Instruction Register (IR)**

Holds the current instruction fetched from memory. The IR provides the Opcode to the Control Unit and Operand to the Datapath.

**4. Data Flow and Execution**

The processor executes instructions in a multi-stage process:

1. **Fetch**:
   * The Program Counter (PC) provides the address of the next instruction.
   * The address is loaded into the MAR, and the instruction is fetched from memory into the IR.
2. **Decode**:
   * The Control Unit decodes the instruction’s Opcode and generates control signals.
   * The Operand is routed to the Datapath.
3. **Execute**:
   * Arithmetic/logic operations are performed in the ALU.
   * Data is transferred between registers or memory as required.
4. **Write Back**:
   * Results are written to the Accumulator, registers, or memory.

**5. Control and Timing**

**Clock Cycle**

Each instruction executes in one or more clock cycles, depending on complexity:

* **Fetch**: 1 cycle
* **Decode**: 1 cycle
* **Execute**: n+1 cycles[n=pipeline stages] (e.g., arithmetic vs. memory access)

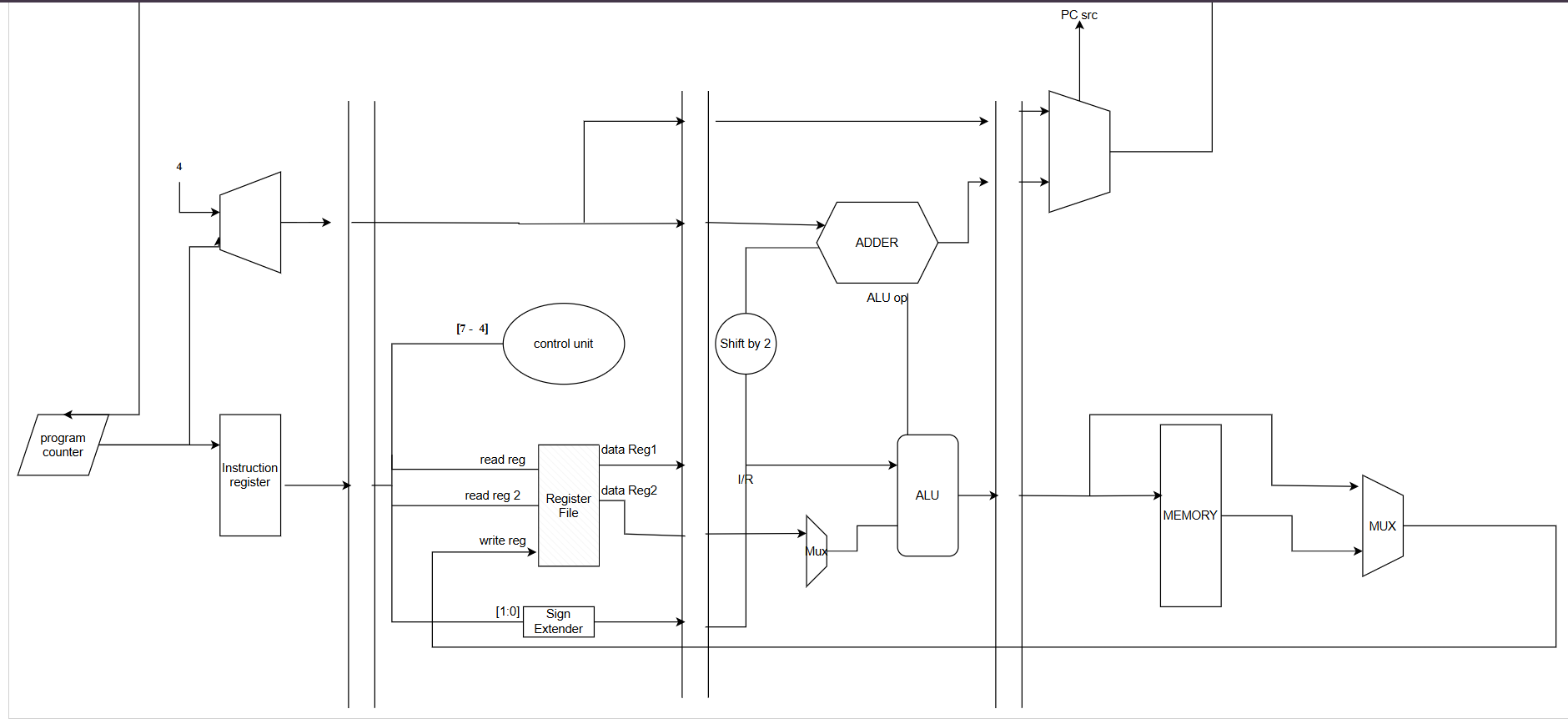
**Control Flow**

Control signals are synchronized with the clock to ensure sequential execution.

**6. Microarchitecture Diagram**

The microarchitecture diagram outlines the processor’s components and their interactions. Key elements include:

* **Control Unit**: At the top, managing instruction decoding and signal generation.
* **Datapath**: In the center, containing the Register Bank, ALU, and Flags Register.
* **Instruction Register**: Positioned between the Control Unit and Datapath.



**7. Conclusion**

This microarchitecture is designed to execute the ISA efficiently, balancing simplicity and functionality. The combination of modular components ensures:

* **Efficient Execution**: Single-cycle or minimal-cycle execution for most instructions.
* **Cost-Effectiveness**: Minimal hardware requirements and low complexity.
* **Scalability**: The architecture can adapt to additional functionality, such as extended memory or enhanced instructions.

This design is ideal for low-power, text-based communication and mobile transaction systems, meeting the project’s objectives.