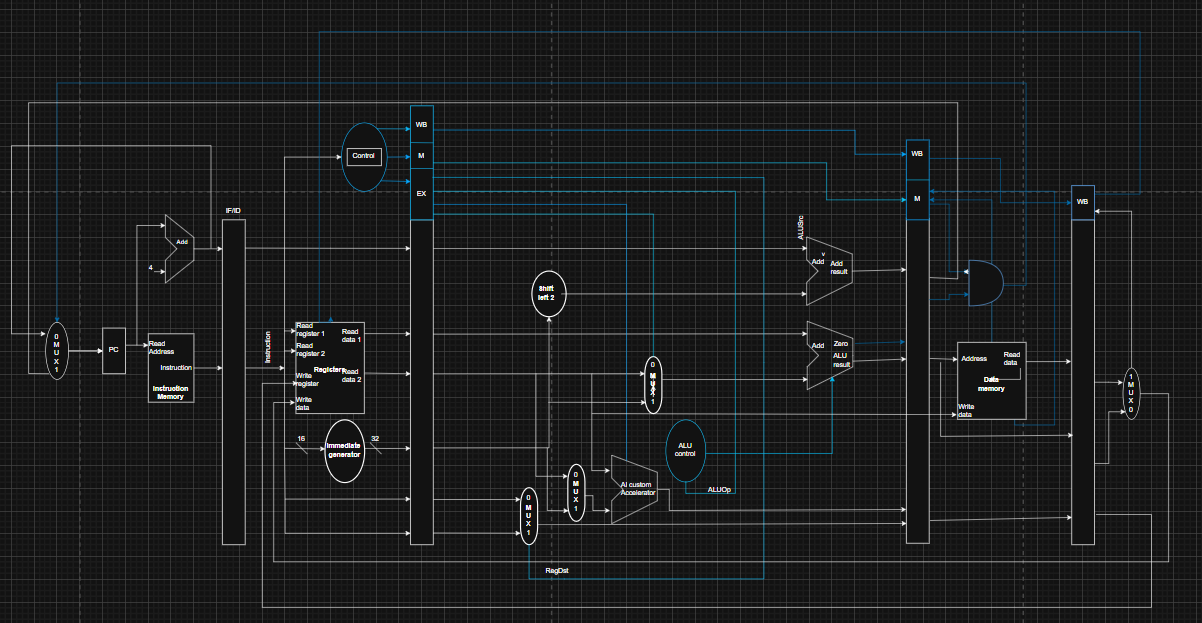
## Microarchitecture Specification Report



**Section 1: Pipelined Datapath Description**

The processor utilizes a **Simple 5-Stage In-Order Pipeline** architecture to achieve a high instruction throughput while maintaining control simplicity. The datapath is a direct translation of the ISA requirements, focusing on a parallel execution path to handle the custom AI instructions. The flow of control and data is partitioned by dedicated pipeline registers, as detailed in the accompanying diagram.

### 1.1 Datapath Stage Breakdown

#### 1.1.1 Instruction Fetch (IF)

* **Function:** Retrieves the next instruction from the memory hierarchy.
* **Components:** **Program Counter (PC)**, PC Adder, Instruction Cache.
* **Flow:** The PC provides the address to the **Instruction Cache**. The **PC Adder** calculates the next sequential address {PC}+4 for 32-bit instructions. The instruction and the updated {PC} are registered in the **IF/ID Pipeline Register**.

#### 1.1.2 Instruction Decode (ID)

* **Function:** Decodes the instruction, fetches source operands, and generates control signals.
* **Components:** **Control Unit**, **Register File (32 GPRs)**, Immediate Generator, Decompression Logic.
* **Flow:** The **Control Unit** decodes the opcode, determining the instruction type (R-Type, Load, Custom AI). The **Register File** reads the values of up to three source registers {Rsrc1}, {Rsrc2}, {Rsrc3}. The **Decompression Logic** expands any 16-bit compressed instructions back to the 32-bit format. All necessary data, control signals, and register destination addresses (Rdest) are passed to the **ID/EX Pipeline Register**.

#### 1.1.3 Execute (EX) - The Acceleration Core

The EX stage is the core of the processor's customization, featuring parallel execution units to maximize throughput for the target workload.

* **Parallel Execution Units:**
  + **Standard ALU:** Handles arithmetic, logic, and memory address calculation (Base + Offset).
  + **Custom AI Accelerator (CAIA):** Contains dedicated hardware (MAC Unit for MAC\_D32 and SAD Unit for {SAD\_8x4) that executes complex AI instructions in a single cycle.
* **Result Selection:** A large **Multiplexer** selects the output of the correct execution unit (ALU, or CAIA) based on the instruction type, governed by the Control Unit.

#### 1.1.4 Memory Access (MEM)

* **Function:** Handles data loads and stores to the Data Cache.
* **Components:** **Data Cache (D-Cache)**.
* **Flow:** For Load/Store instructions, the calculated address from the EX stage is used to either read a word from the **D-Cache** or write a word to it. All computational instructions MAC\\_D32 and SAD\\_8x4 bypass this stage entirely, saving cycles and power.
* **Flow Out:** The memory read result (if a Load) or the computational result is stored in the **MEM/WB Pipeline Register**.

#### 1.1.5 Write Back (WB)

* **Function:** Updates the processor state by writing the final result back to the Register File.
* **Components:** Final Write MUX, **Register File**.
* **Flow:** A final **Multiplexer** selects the data source: either the data read from the **D-Cache** (for Loads) or the result calculated in the **EX stage** (for R-Type/Custom instructions). The selected value is then written back to the **Register File** at the Rdest address.

