PR-5: Stage Pipelined LEGv8 Architecture

Revision 1.0

1 Overview

The PR-5 is a 32-bit pipelined processor implementing the LEGv8 ISA designed to serve as a reference model for understanding pipeline behavior, hazard resolution, stage-level data flow in a five-stage architecture.

This document assumes the reader has prior experience with digital logic design.

2 Architectural Features

The following features define the architectural scope of the PR-5 processor and establish the assumptions used throughout the pipeline description.\

Features:

- 32-bit fixed width instructions.
- 32 general purpose registers (R0-R31).
- Single cycle ALU operations.
- Separate instruction and data memory interfaces.
- Hazard detection and forwarding unit.
- Static branch predication (predict not taken).

Pipeline Stages (in execution order):

1. Instruction Fetch (IF)
2. Instruction Decode (ID)
3. Execute (EX)
4. Memory Access (MEM)
5. Write Back (WB)

3 Pipelined stages

Each pipeline stage is documented independently so that readers can reference specific stages without reading the document linearly.

3.1 Instruction Fetch (IF)

The Instruction Fetch stage is the first step in the pipeline where the CPU will retrieve the next program instruction from instruction memory.

Overview

- Program Counter (PC) increments by 4 bytes per cycle.
- Instruction memory is accessed synchronously with the clock's rising edge.
- Branch resolution occurs in the EX stage. If a branch is taken, IF stage is flushed.

Inputs

- PC
- Branch target address (from EX stage)
- Branch taken signal

Outputs

- Instruction
- PC+4

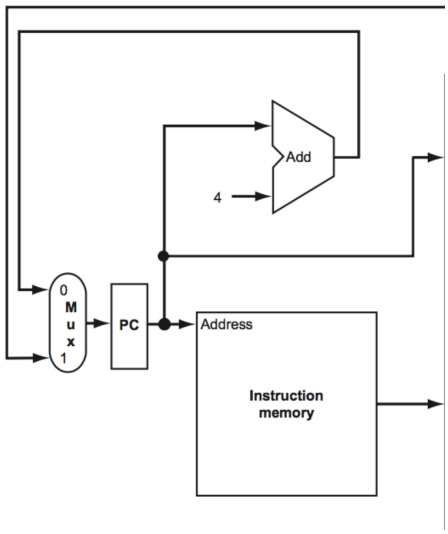
3.1.1 Components

MUX

The Mux determines which input signal is loaded into the PC.

- Input 0: Branch Target address
- Input 1: PC
- Select: Branch taken signal

Note: Branch decision logic is determined in the Execute stage, and the branch outcome signal is written back during the Memory stage.



PC

The PC holds the address to the current instruction. Its output is routed to two locations, the instruction memory and an adder which increments the current address every cycle.

- Output: Connected to instruction memory

- Input: current PC+4 or Branch Target

Note: The branch target is determined in the Instruction Decode stage.

Instruction Memory
Holds the instructions for a program. Each instruction is given a unique address, the output of instruction memory is determined by the selected address. The address is controlled by the PC.

- Output: Connected to instruction decode
- Input: PC

3.2 Instruction Decode (ID)

The Instruction Decode stage is where the binary representation of a fetched instruction is interpreted and used to determine the necessary actions the control unit needs to take. LEGv8 has 3 types of instructions.

Overview
- Register File (2 Read 2 Writes)
- Control Unit
- Hazard Detection Unit

The following table gives details on the bit map of each type of instruction.

Field	opcode	Rm	shamt	Rn	Rd
Bit positions	31:21	20:16	15:10	9:5	4:0

a. R-type instruction

Field	1986 or 1984	address	0	Rn	Rt
Bit positions	31:21	20:12	11:10	9:5	4:0

b. Load or store instruction

Field	180	address	Rt
Bit positions	31:26	23:5	4:0

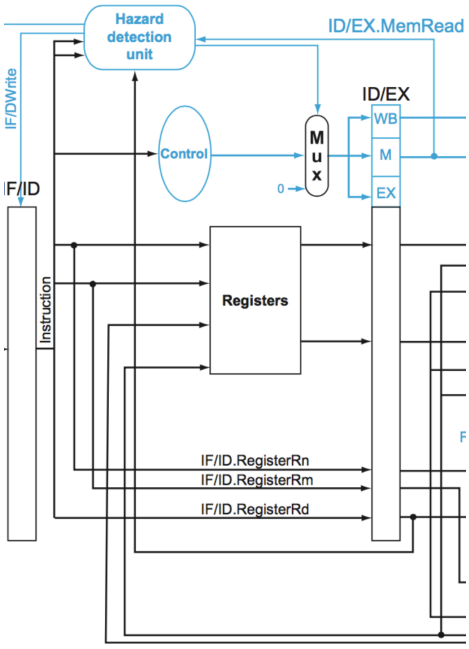
c. Conditional branch instruction

R-Type: Arithmetic operations (ADD, SUB, MULT)
L-Type: Load and store operations (Ld, St)
B-Type: Branch logic (B, BEQ, BLT, BGT)

3.2.1Components

Hazard Detection Unit

The hazard detection unit identifies data and control hazards that would lead to incorrect execution if the pipeline continued normally. It monitors register dependencies between instructions in adjacent pipeline stages and determines when a stall or pipeline flush is required.



When a hazard is detected that cannot be resolved through forwarding, the unit inserts pipeline stalls to preserve correctness while minimizing unnecessary performance loss.

Control Unit

The control unit interprets the opcode and instruction fields to generate control signals required by each pipeline stage. These signals determine ALU behavior, register writes, memory access, and branch handling.

Register File

The register file stores the processor's general-purpose registers and provides operands to the pipeline during the Instruction Decode stage. In order to prevent control hazards the register file has two read and two write ports.

3.3 Execute (EX)

The execute stage performs arithmetic and logical operations and evaluates branch conditions. This stage represents the core computational step of the pipeline.

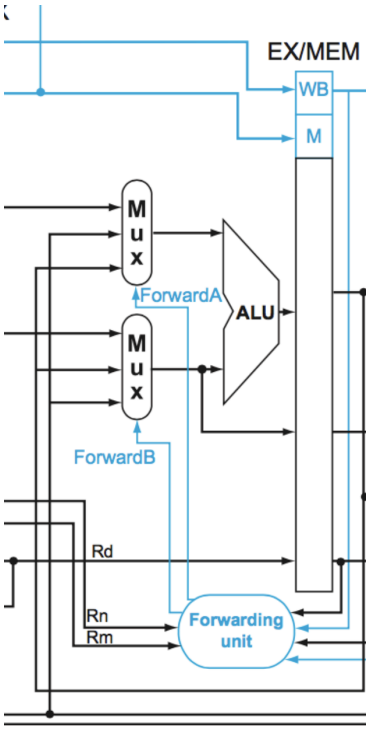
- ALU performs arithmetic and logical operations.
- Branch conditions evaluated.
- Forwarding logic resolves data hazards (from MEM and WB).

3.3.1Components

Arithmetic Logic Unit (ALU)

The ALU performs arithmetic and logical operations specified by the instruction.
ALU results are forwarded to the Memory access stage.

Note: See section 3.2 for more information on instructions



Forwarding Unit

The forwarding unit is responsible for identifying data hazards.

Data hazards occur when an instruction reads from a register or memory location before a previous instruction has finished writing to it.

The forwarding unit resolves data hazards by redirecting results from later pipeline stages directly to the ALU to prevent unnecessary stalls.

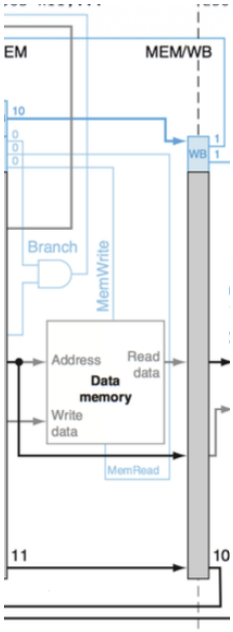
Mux

The mux determines the inputs on the ALU based on the instruction type. ALU inputs could be directly from memory, register file, or address.

3.4 Memory Access (MEM)

The memory access stage handles data memory operations for load and store instructions. It is responsible for reading data from memory or writing data to memory using the address computed during the execution stage.

Instructions that do not require memory access pass through this stage without modifying memory contents.



Overview

- Load and store instructions access data memory.
- Memory assumes single cycle latency.
- If branch is taken address is written back to IF

3.4.1Components

Memory

The data memory unit provides single cycle access for load and store operations. Memory addresses are supplied by the ALU result from the Execute stage.

3.5 Write Back

The write back stage completes an instruction by updating the register file with the results. Depending on the instruction the value written to register is either from the ALU or from data memory.

Overview

- ALU results or memory data written to the register file.
- Write occurs on a rising clock edge.

3.2.1 Components

Mux

The mux selects between the ALU result and memory data to determine the value written to the destination register.

Note: target register address is determined by the instruction. See section 3.2 for more information.

Mux select signal determines if the ALU result or data memory result is written to the register file.