1. Project Overview

This project implements a **MIPS Pipeline Processor** designed using **VHDL**. It follows the **5-stage pipeline architecture**:

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

The processor supports arithmetic, logical, memory, and custom trigonometric (`SIN` and `COS`) operations.

7.3 Single-Cycle Processor 375

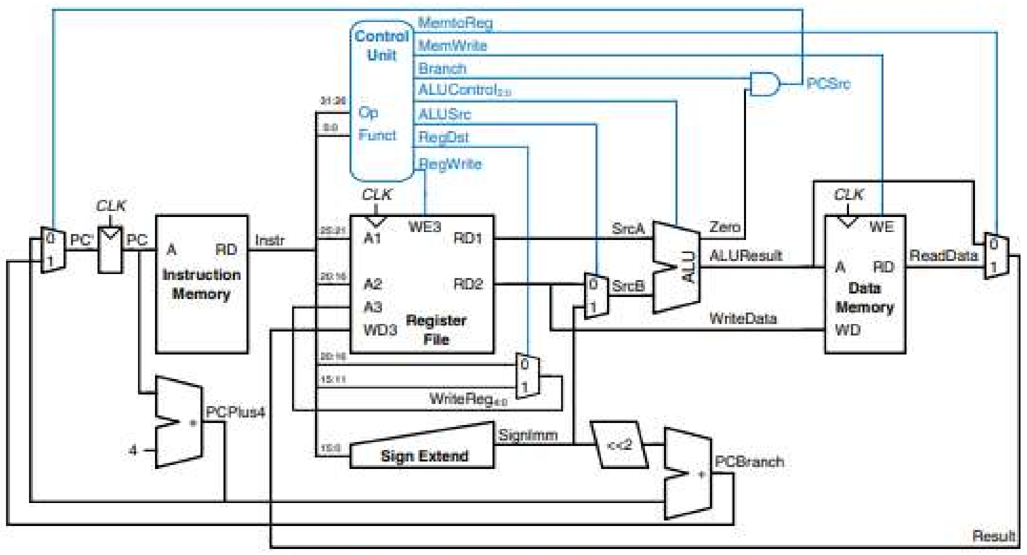

Figure 7.11 Complete single-cycle MIPS processor

This is the single cycle version of the mips which is what we used to create the single components

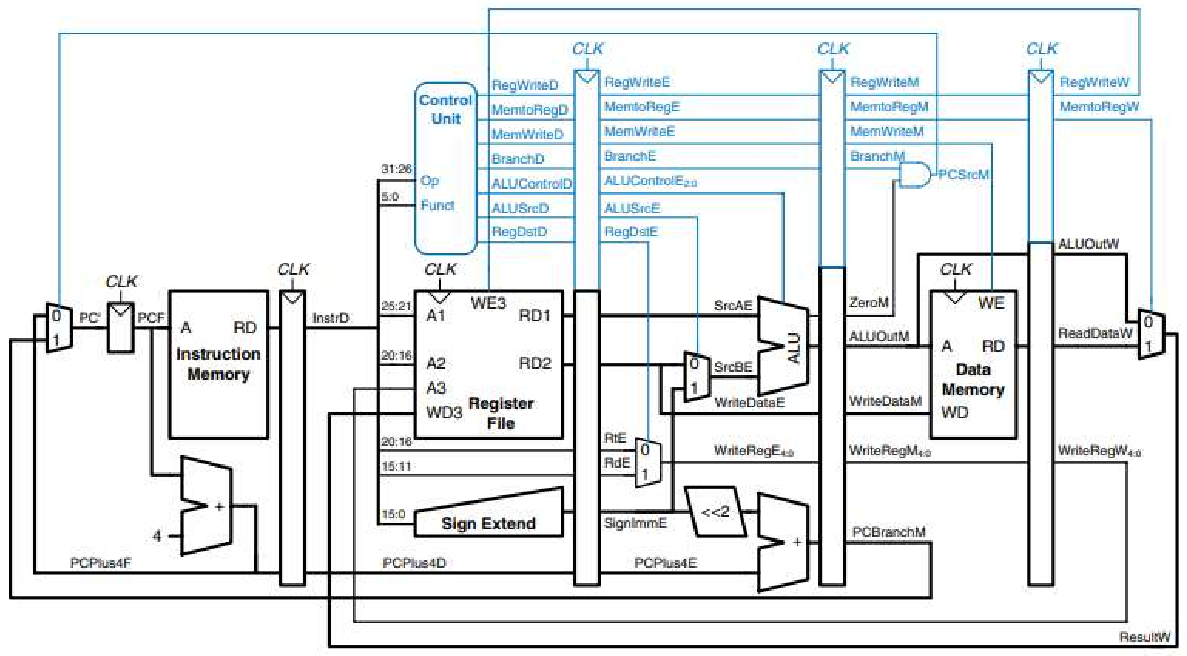

Figure 7.47 **Pipelined processor with control**

This is the pipelined version which is how we grouped the componants together and created the pipeline component from.

⚙️ 2. Design Methodology

The design follows the principles of **modular design**, with each stage implemented as an independent module:

2.1 Instruction Fetch (IF)

- Fetches the instruction from **Instruction Memory** based on the **Program Counter (PC)**.
- Supports **Branch** and **Jump** logic for control flow changes.

2.2 Instruction Decode (ID)

- Decodes fetched instructions to generate control signals**.**
- Accesses the **Register File** for operand retrieval.
- Performs **Immediate Value Extension** for ALU operations.

2.3 Execute (EX)

- Performs arithmetic and logical operations via the **ALU**.
- Handles custom `SIN` and `COS` operations for specific instructions.
- Calculates **Branch Target Addresses** when needed.

2.4 Memory Access (MEM)

- Accesses **Data Memory** for `LW` (Load Word) and `SW` (Store Word) instructions.

2.5 Write-Back (WB)

- Writes results from the **ALU** or **Data Memory** back into the **Register File**.

🛠️ 3. Custom Operations

3.1 SIN and COS Implementation

- **Opcode Mapping:**
  - `SIN`: `001001`
  - `COS`: `001010`
- **Degree-Based Approximation:**
  - SIN and COS are approximated using predefined integer values scaled by 100.
- Example Approximations:
  - `SIN(90°) → 100`
  - `COS(0°) → 100`

🧩 4. Components Used

4.1 Key Components

- **Program Counter (PC)**
- **Instruction Memory**
- **Data Memory**
- **Register File**
- **ALU (Arithmetic Logic Unit)**
- **Pipeline Registers (5 stages)**
- **Multiplexers (MUX)**

4.2 Control Unit

- Decodes opcodes and funct fields.
- Generates control signals for ALU, Memory, and Write-back stages.

📊 5. Testing and Validation

✅ 5.1 Test Cases

- **Basic Arithmetic:** ADD, SUB, AND, OR
- **Memory Operations:** LW, SW
- **Control Flow:** BEQ, JUMP
- **Custom Operations:** SIN, COS

✅ 5.2 Simulation Tools Used

- **ModelSim** or similar VHDL simulation tools.

📝 6. Key Results

- Accurate pipeline execution with minimal hazards.
- SIN and COS operations validated with predefined approximations.
- Correct instruction execution across all pipeline stages

📦 7. Future Improvements

- Implement floating-point support.
- Optimize pipeline hazard detection and forwarding.

📌 8. Conclusion

This project successfully implements a **MIPS Pipeline Processor** with support for arithmetic, logical, memory, and trigonometric operations (`SIN` and `COS`). The design is modular, scalable, and serves as an excellent foundation for further improvements.