

The diagram illustrates the MIPS processor architecture with the following components and data paths:

- PC (Program Counter):** Receives the next instruction address. It feeds into the **Read address** of **Instruction memory** and the **Add** block for **PC+4**.
- Instruction Memory:** Provides the instruction based on the **Read address**. The instruction is split into fields: **[31:0]** (OpCode), **[25:21]** (Read register 1), **[20:16]** (Read register 2), **[15:11]** (Write register), and **[15:0]** (OpCode).
- Control Unit:** Decodes the instruction fields into control signals: **RegDst**, **Jump**, **Branch**, **MemRead**, **MemtoReg**, **MemWrite**, **ALUOp**, **ALUSrc**, and **RegWrite**.
- Registers:** A set of 32 registers. **Read register 1** and **Read register 2** provide **Read data 1** and **Read data 2**. The **Write register** selects the register to be updated with **Write data**.
- ALU (Arithmetic Logic Unit):** Performs operations on **Read data 1** and **Read data 2** based on **ALUOp**. It produces the **ALU result** and a **Zero** flag.
- Shifters:** Two **Shift left 2** units. One shifts the **OpCode** field to the left by 2 bits. The other shifts the **ALU result** left by 2 bits.
- Multiplexers (MUX):**
 - PC+4 MUX:** Selects between the current **PC** and **PC+4** (calculated by adding 4 to the **PC**).
 - Write Data MUX:** Selects between the **ALU result** and **Read data 2** to be written to the register.
 - Jump Address MUX:** Selects between the **PC+4** and the **Jump address** (calculated by adding the shifted **OpCode** to the **PC**).
 - Read Data MUX:** Selects between the **ALU result** and **Read data 1** to be written to the data memory.
- Sign-extend 16-32:** Extends the 16-bit **OpCode** field to 32 bits.
- ALU Control:** Generates the **ALUOp** signal based on the instruction's **OpCode** field.

Specifications of supported MIPS assembly instructions:

```

add  $rd, $rs, $rt      # GPR[rd] = GPR[rs] + GPR[rt]
sub  $rd, $rs, $rt      # GPR[rd] = GPR[rs] - GPR[rt]
and  $rd, $rs, $rt      # GPR[rd] = GPR[rs] & GPR[rt]
or   $rd, $rs, $rt      # GPR[rd] = GPR[rs] | GPR[rt]
slt  $rd, $rs, $rt      # GPR[rd] = GPR[rs] < GPR[rt] ? 1 : 0
lw   $rt, imm16($rs)    # GPR[rt] = Mem[ GPR[rs] + imm16 ]
sw   $rt, imm16($rs)    # Mem[ GPR[rs] + imm16 ] = GPR[rt]
beq  $rs, $rt, offset   # PC <-- (rs == rt ? PC + 4 + offset << 2)
                                     #                               : PC + 4)
j    target             # PC <-- ( (PC+4)(31:28) || (target << 2) )

```

The preliminary pipelined MIPS32 datapath with interstage buffers for synchronization:

