

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

- PC (Program Counter):** Receives the next instruction address from the PC+4 adder or the jump address. It outputs the instruction address to Instruction Memory.
- Instruction Memory:** Receives the instruction address from the PC and outputs the instruction to the Instruction Register.
- Instruction Register:** Holds the current instruction, providing fields to the Control Unit, Registers, ALU, and other components.
- Control Unit:** Decodes the instruction and generates control signals for the Registers, ALU, and other components.
- Registers:** Store data and are controlled by the Control Unit. They provide data to the ALU and the Data Memory.
- ALU (Arithmetic Logic Unit):** Performs operations on data from the registers, controlled by the ALU Control. It outputs the ALU result to the ALU result register.
- ALU result register:** Holds the result of the ALU operation, which is then used by the Data Memory or the PC+4 adder.
- Data Memory:** Receives data from the registers and outputs data to the registers or the ALU result register.
- PC+4 Adder:** Adds 4 to the current PC value to determine the next instruction address.
- Jump Address Adder:** Adds the jump address to the PC value to determine the next instruction address.
- Shifters:** Perform shift operations on data from the registers, controlled by the Control Unit.
- MUXes (Multiplexers):** Select between different data sources based on control signals.

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:

