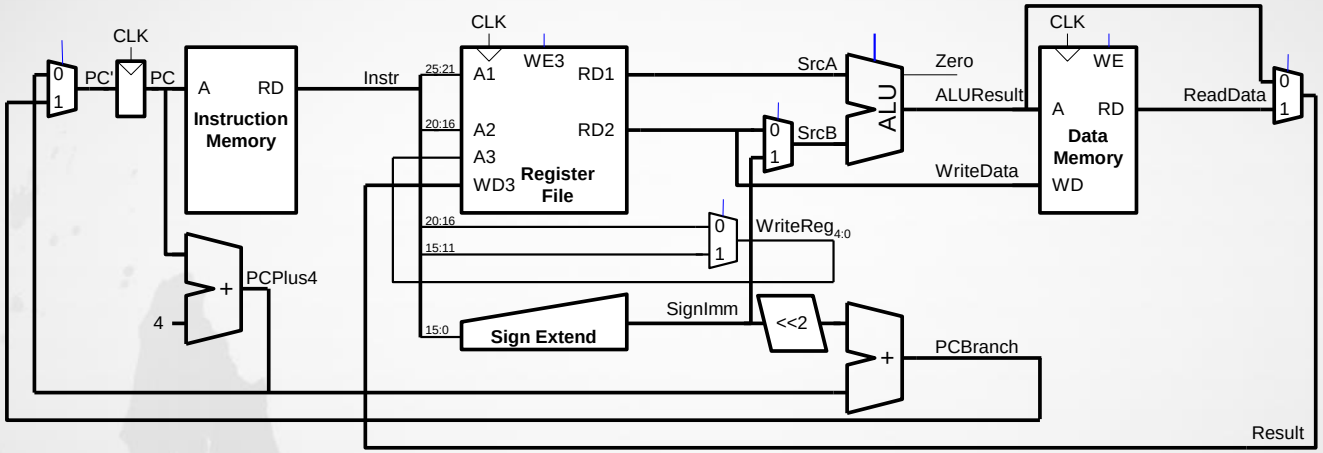
Computer Architecture

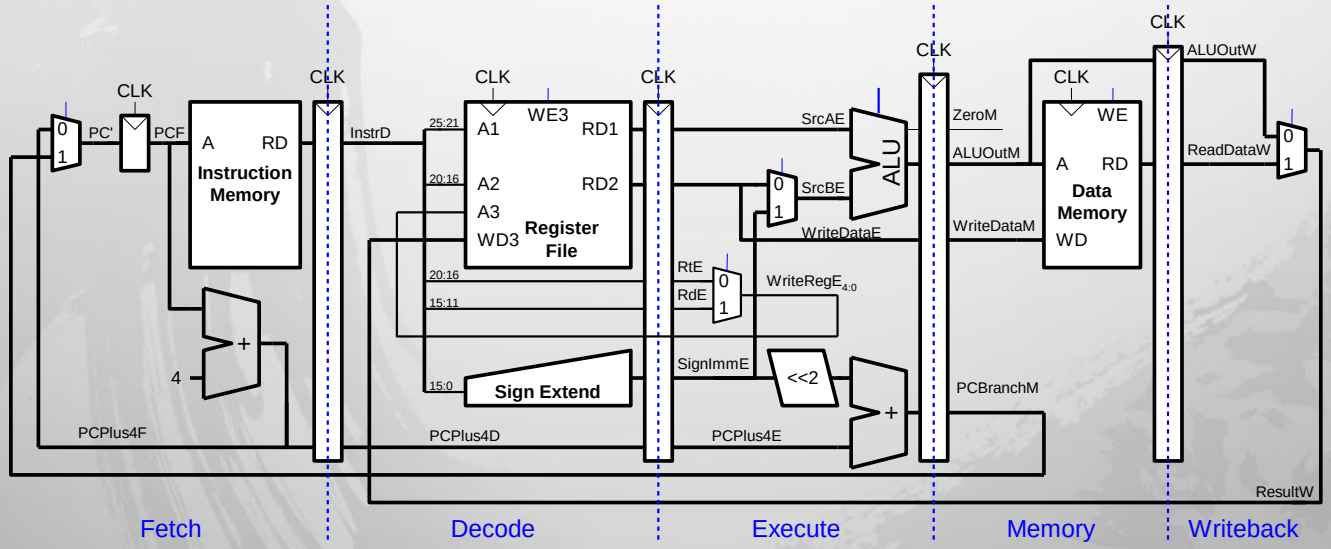
Lab 10: Pipeline

Getting Started

- Download alu.sv, regfile.sv, imem.sv, dmem.sv and controller.sv from Blackboard
- Open a web browser and go to https://www.edaplayground.com/

Single-Cycle & Pipelined Datapath





Pipelining Sample

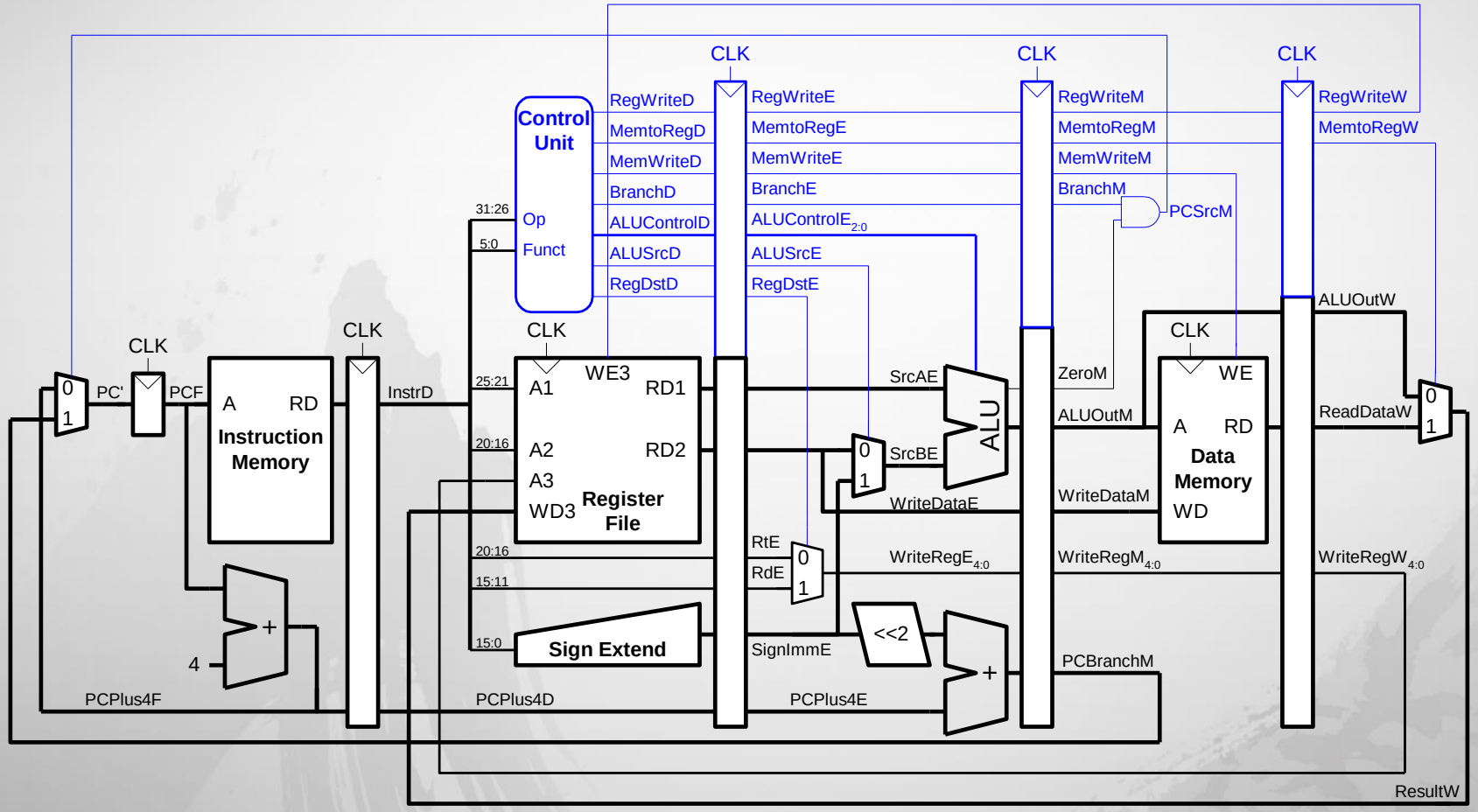
```
logic [31:0] IMEM_InstF;
logic [31:0] IMEM_InstD;


imem IMEM(
  .iAddr        (pc),
  .oRdata       (IMEM_InstF)
);


always_ff@(posedge iClk, posedge iReset)
  if(iReset) begin
    IMEM_InstD <= 0;
  end else begin
    IMEM_InstD <= IMEM_InstF;
  end


assign RtD = IMEM_InstD[20:16];
assign RdD = IMEM_InstD[15:11];
```

Pipelined Processor Control



- **Same control unit as single-cycle processor**
- **Control delayed to proper pipeline stage**

Testbench

- Copy the following testbench code to “testbench.sv” on the left

```
module testbench_mips();
  logic clk;
  logic reset;

  mips dut(
    .iClk                          (clk),
    .iReset                        (reset)
  );

  always
    begin
      clk = 1; #5; clk = 0; #5;
    end

  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
    reset = 0; #21;
    reset = 1; #10;
    reset = 0; #50;
    #10; $stop;
  end
endmodule
```

Datapath

- Copy the following code to “design.sv” on the right

```
`include "alu.sv"
`include "regfile.sv"
`include "imem.sv"
`include "dmem.sv"
`include "controller.sv"

module mips(
  input logic iClk,
  input logic iReset
);

  logic [31:0] ALU_ALUResult;
  logic [31:0] REG_SrcA;
  logic [31:0] REG_WriteData;
  logic [31:0] IMEM_InstF;
  logic [31:0] IMEM_InstD;
  logic [31:0] DMEM_ReadData;
  logic [31:0] pc;
  logic [4:0] WriteReg;
  logic [31:0] SrcB;
  logic [31:0] Result;
  logic [15:0] SignImm;
  logic [4:0] Rt;
  logic [4:0] Rd;

  logic CTL_RegWrite;
  logic CTL_MemWrite;
  logic CTL_RegDst;
  logic CTL_ALUSrc;
  logic CTL_MemtoReg;
  logic [2:0] CTL_ALUControl;

  assign Rt = IMEM_InstD[20:16];
  assign Rd = IMEM_InstD[15:11];
  assign WriteReg = CTL_RegDst ? Rd : Rt;
  assign SignImm = {{16{IMEM_InstD[15]}}, IMEM_InstD[15:0]};
  assign SrcB = CTL_ALUSrc ? SignImm : REG_WriteData;
  assign Result = CTL_MemtoReg ? DMEM_ReadData : ALU_ALUResult;

  alu ALU(
    .iA                                                         (REG_SrcA),
    .iB                                                         (SrcB),
    .iF                                                         (CTL_ALUControl),
    .oY                                                         (ALU_ALUResult),
    .oZero                          ()
  );

  regfile REG(
    .iClk                           (iClk),
    .iReset                         (iReset),
    .iRaddr1(IMEM_InstD[25:21]),
    .iRaddr2(IMEM_InstD[20:16]),
    .iWaddr                         (WriteReg),
    .iWe                            (CTL_RegWrite),
    .iWdata                         (Result),
    .oRdata1(REG_SrcA),
    .oRdata2(REG_WriteData)
  );

  imem IMEM(
    .iAddr                          (pc),
    .oRdata                         (IMEM_InstF)
  );

  dmem DMEM(
    .iClk                           (iClk),
    .iReset                         (iReset),
    .iWe                            (CTL_MemWrite),
    .iAddr                          (ALU_ALUResult),
    .iWdata                         (REG_WriteData),
    .oRdata                         (DMEM_ReadData)
  );

  controller CTL(
    .iOp                                                        (IMEM_InstD[31:26]),
    .iFunc                                                      (IMEM_InstD[5:0]),
    .oRegWrite                      (CTL_RegWrite),
    .oMemWrite                      (CTL_MemWrite),
    .oRegDst                        (CTL_RegDst),
    .oALUSrc                        (CTL_ALUSrc),
    .oMemtoReg                      (CTL_MemtoReg),
    .oALUControl(CTL_ALUControl)
  );

  always_ff@(posedge iClk, posedge iReset)
    if(iReset)
      pc <= 0;
    else
      pc <= pc + 4;

    always_ff@(posedge iClk, posedge iReset)
      if(iReset) begin
                                                IMEM_InstD <= 0;
        end else begin
          IMEM_InstD <= IMEM_InstF;
        end

endmodule
```

Building Blocks

- Click "+" right after "design.sv"
- Upload "alu.sv"

- Upload regfile.sv, imem.sv, dmem.sv, and controller.sv in the same way

Lab Assignment

- Finish the implementation of the datapath module
- Save and submit the link of your design to the Blackboard.
  - Click 🔗 in the bottom window to copy the URL of your design.

Expected Result