TED ÜNİVERSİTESİ

CMPE361 Computer Organization

Department of Computer Engineering
TED University- Fall 2023

Single Cycle Processor - I

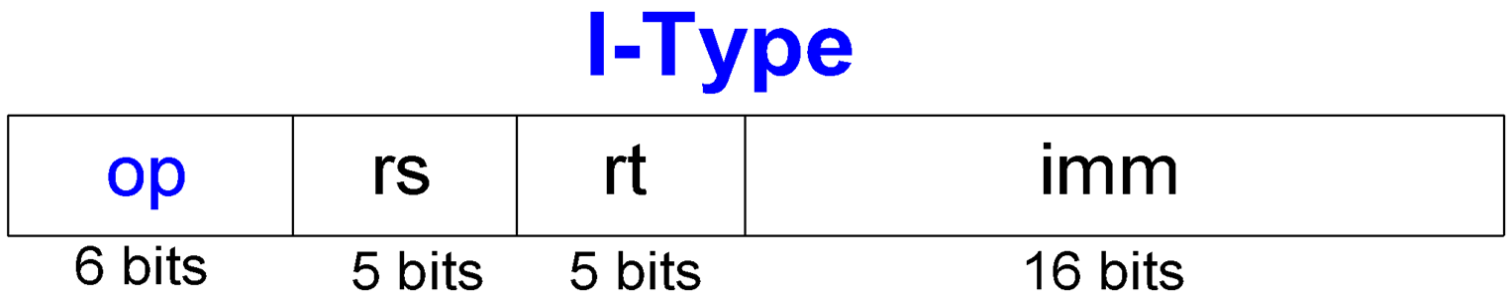
These Slides are mainly based on slides of the text book (downloadable from the book’s website).

Single-Cycle Processor Design

- Use the state components (blocks) described so far, to design a MIPS microarchitecture that executes each instructions in one clock cycle.
  - ✓ A clock cycle can be defined as the time it takes from a rising edge to the next one.
- Construct the datapath: connect the state elements with combinational logic to prepare for the data travel through the blocks.
- Construct the controller: generate the appropriate signals to control the datapath, in line with the instructions' requirements, when they are executed.

Sample Instruction Implementations

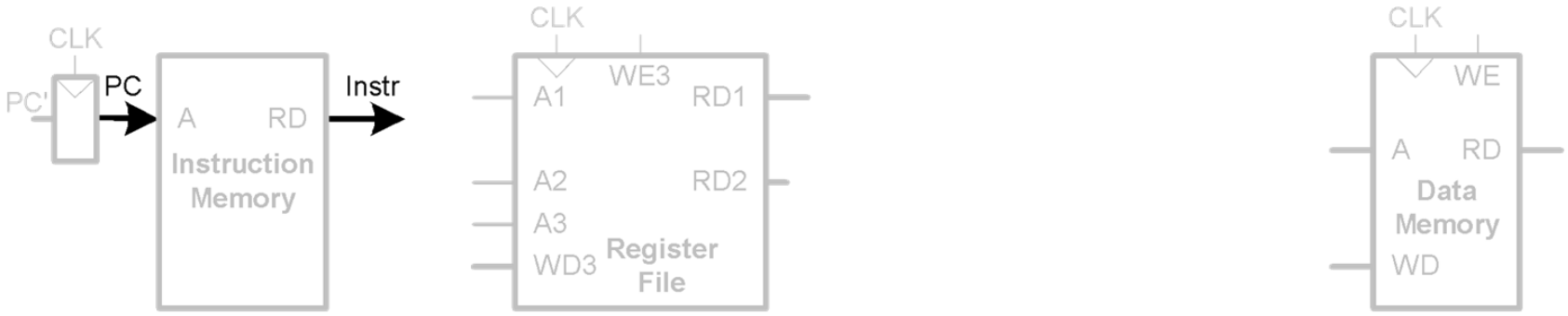
`lw` (fetch): Load a word from memory to a register



- Read the instruction from the instruction memory
- The source register's id bits are connected to the address input of one of the register block's input ports,
- The register block reads the identified register value to 32 bit port RD1.
- Meantime, 16-bit immediate must be sign-extended to 32 bits,.

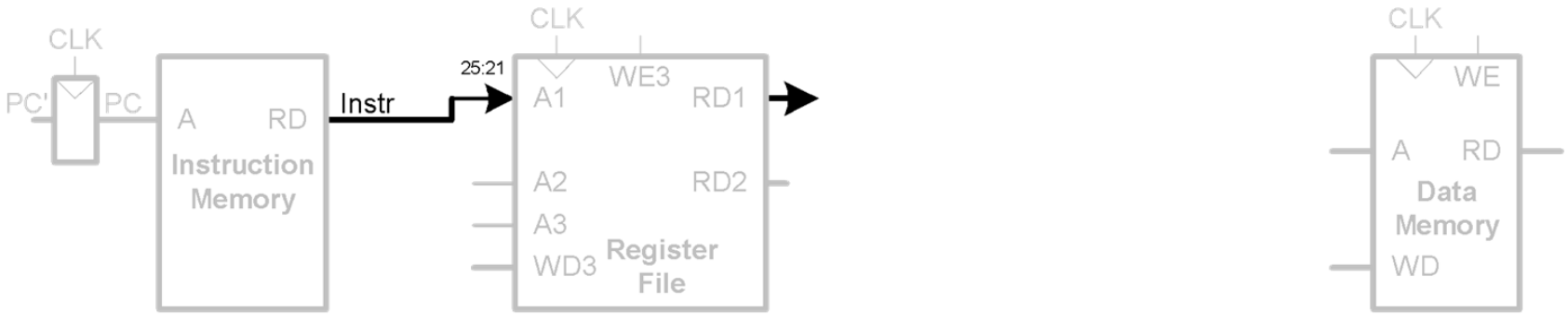
`lw`: get instruction from inst memory

STEP 1: Fetch instruction



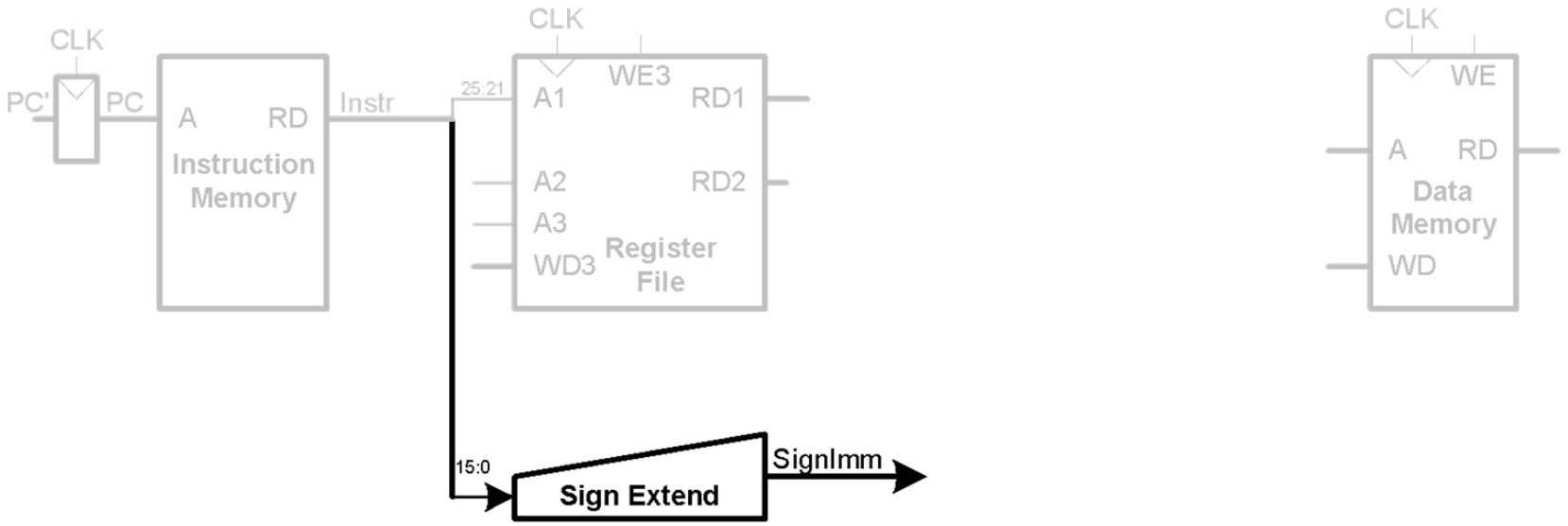
Lw: get content of source reg from RF

STEP 2: Read source operands from RF



Lw: Sign extend immediate

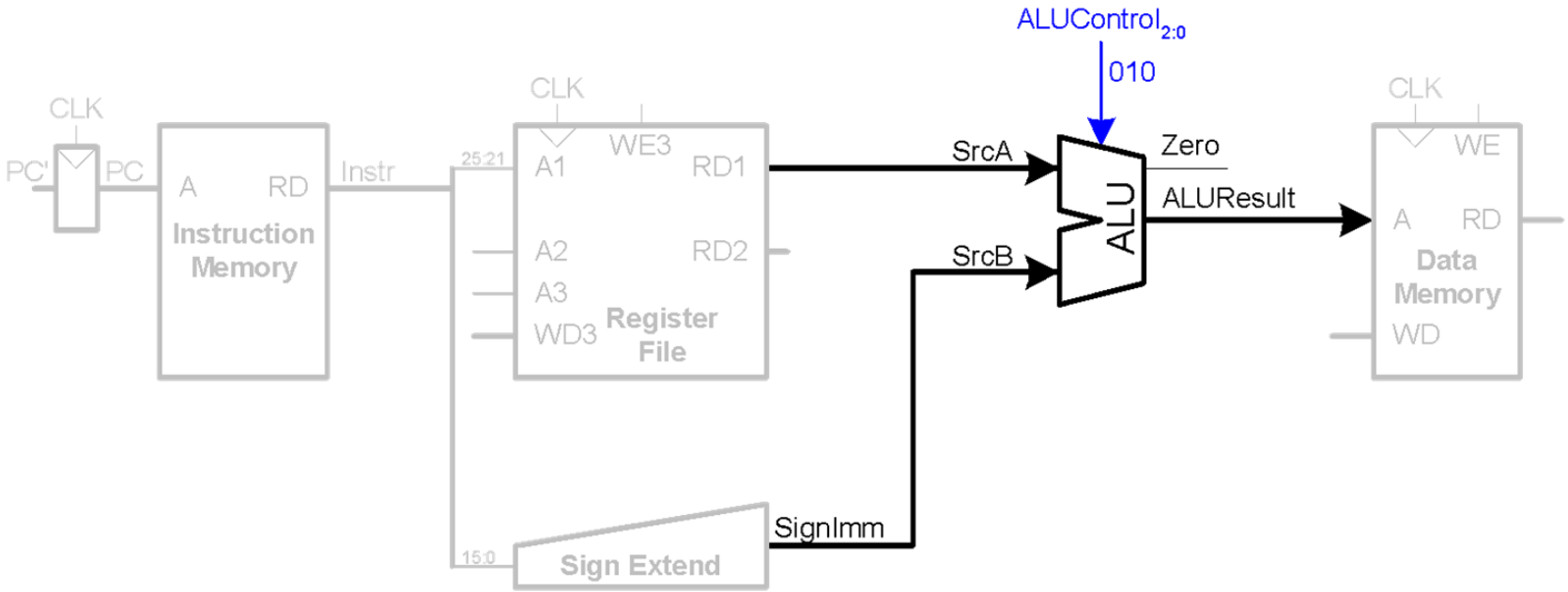
STEP 3: Sign-extend the immediate from the instruction, in parallel with the previous operation



lw: add immediate and reg

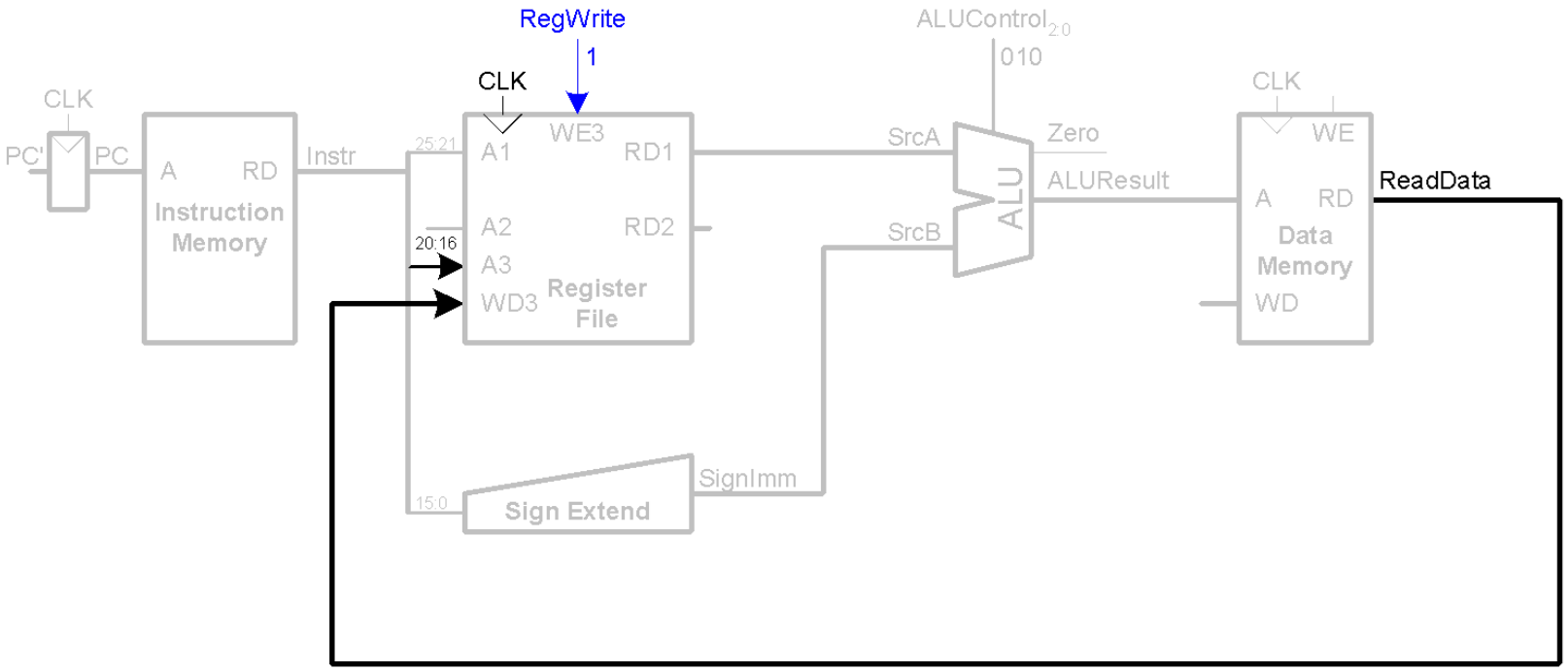
STEP 4:

- Compute the memory address (addition of base register and immediate)
- Make the available on the data memory port A, to fetch the its content



`lw`: read data from the mem address computed

- **STEP 5:**
- write the data read back to write-port of RF to be copied to the destination register
  - The write takes place on the rising edge of the clock at the end of the cycle.

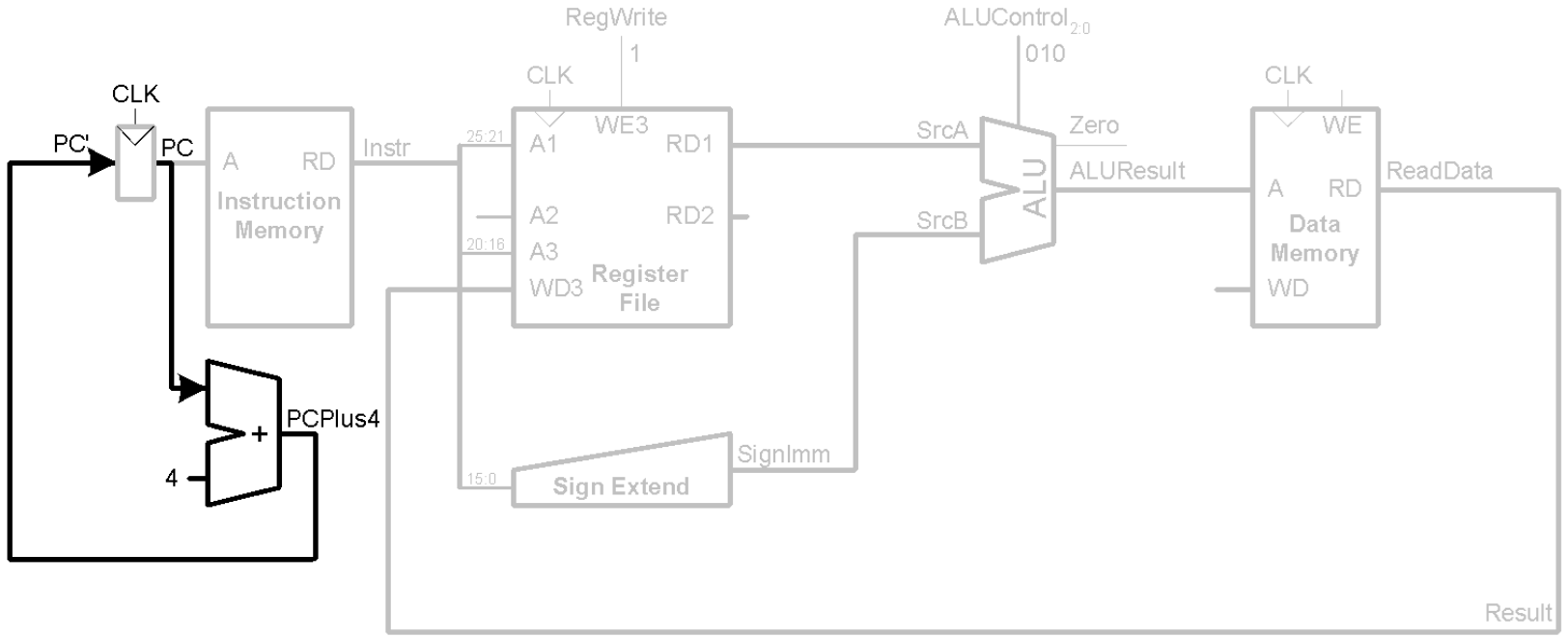


lw: Increment PC by 4

STEP 6:

Compute the address of the next instruction:

The new address is written into the program counter on the next rising edge of the clock. This completes the datapath for the lw instruction.



Other instructions: sw, beq

Every instruction requires a number of operation on the relevant data.

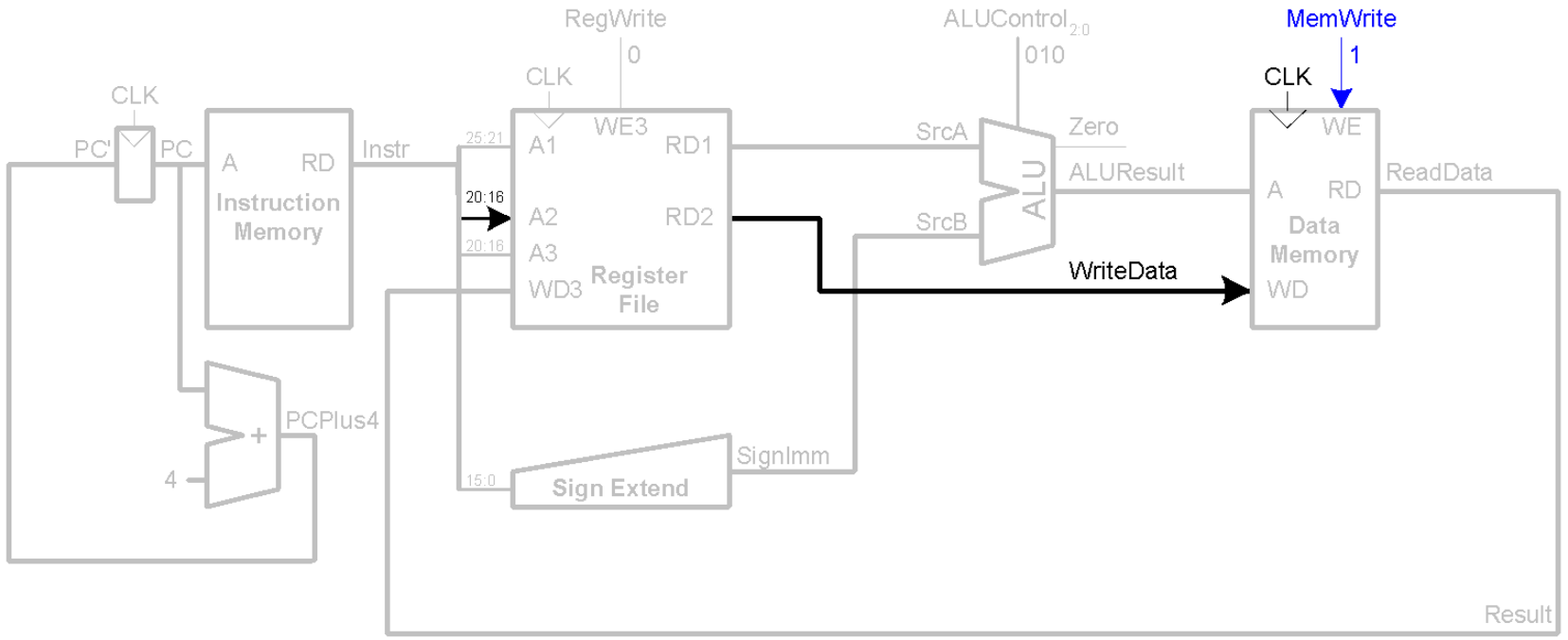
Similar procedure needs to be applied for each such instruction

sw- store word instruction

- The sw instruction reads the register from the RF and writes it to the data memory.
  - ✓ The register specified in the rt field is connected to the second RF read port A2.
  - ✓ The register value is copied to the RF's RD2 port, which is connected to the write-data port of the data memory.
  - ✓ MemWrite = 1, to write the data to memory; ALUControl = 010, to **add** the base address and offset; RegWrite=0 as nothin is written to the register.

Sw cont.

Write data in `rt` to memory



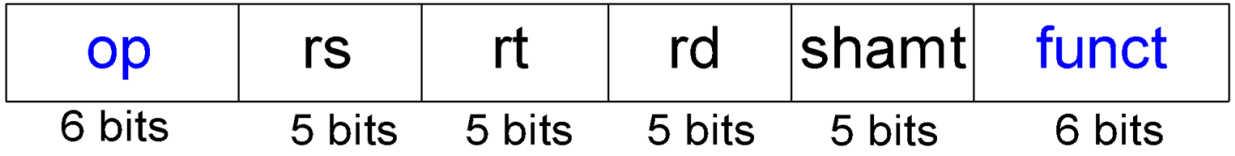
R-Type Instructions

- Extending the datapath to handle the R-type instructions add, sub, and, or, and slt (Set Less Than).
- All of these instructions read two registers from the RF, perform some ALU operation on them, and write the result back to the third register.
- Fortunately they can all be handled by the same hardware, with different combination of ALUControl signals.

Implementation of R-Type instructions

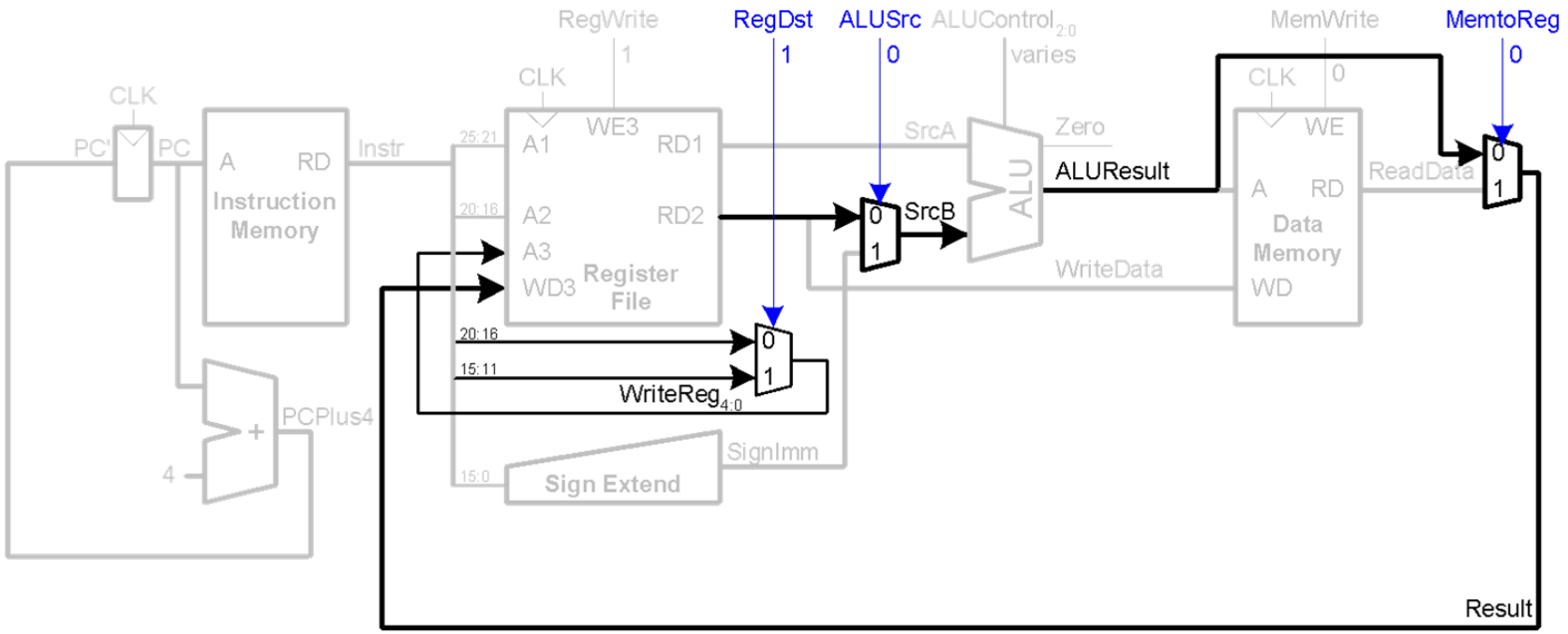
add, sub, and, or, and **slt** instructions.

R-Type



R-Type: Datapath enhancement

- Read from `rs` and `rt`
- Write *ALUResult* to rd



R-Type: control of datapath

- A multiplexer is used to choose SrcB either from the RF port RD2 or SignImm.
- The multiplexer is controlled by a new signal, ALUSrc.
  - ✓ ALUSrc is 0 for R-type instructions: choose SrcB from the RF
  - ✓ ALUSrc is 1 for lw and sw: choose SignImm .

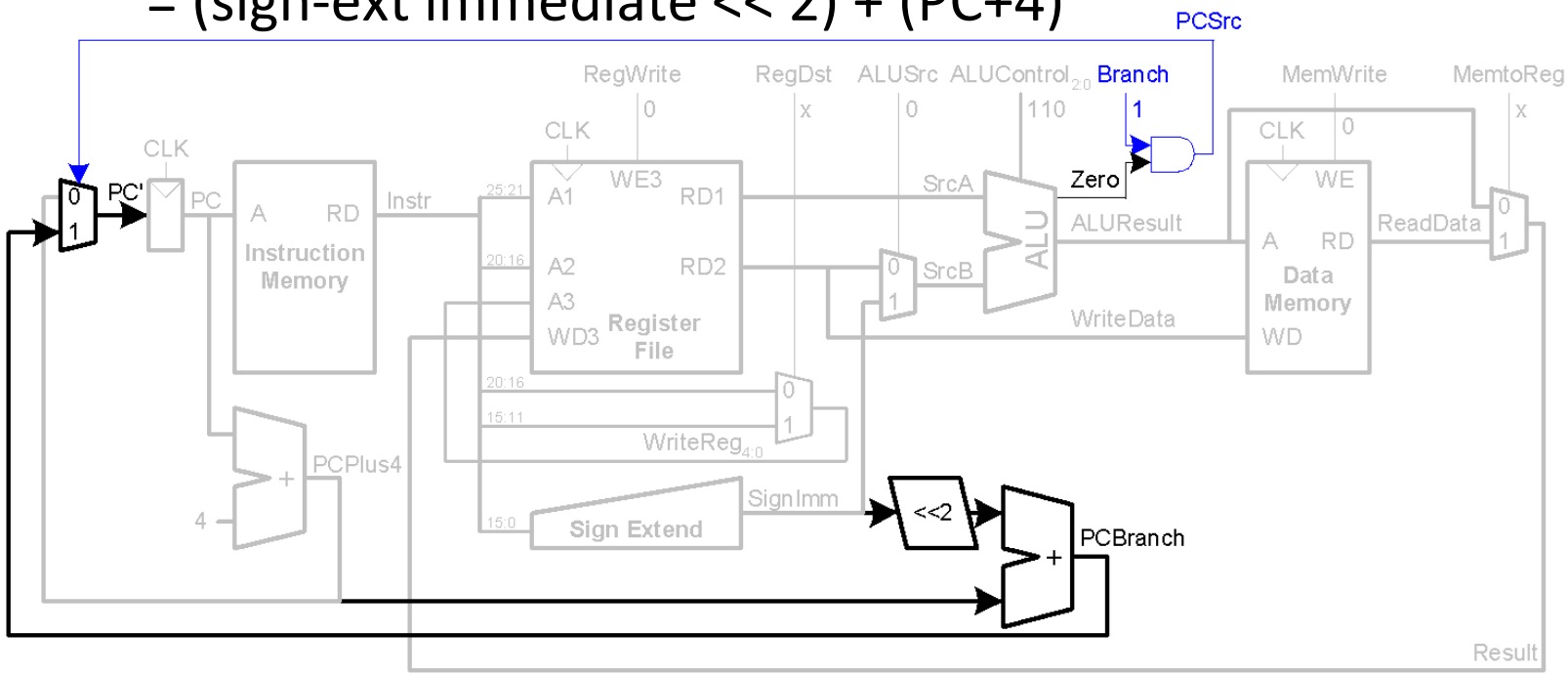
R-Type:

- A second multiplexer is used to choose memory (ReadData) or ALU output (ALUResult)
- This mux is controlled by MemtoReg signal.
  - ✓ MemtoReg is 0 for R-type instructions to choose Result from the ALUResult; it is 1 for lw to choose from memory (ReadData).
- A third multiplexer controlled by RegDst signal is added to choose WriteReg from the appropriate field of the instruction.
  - ✓ RegDst=1 for R-Type instr to choose reg rd in bits 15-11; RegRDst=0 for I-Type inst to choose rt in bits 20-16

Beq: branch equal

- "beq rs, rt, imm" inst calculates the branch target address (BTA) based on whether values in `rs` and `rt` are equal or not
- Branch target address-BTA

  = (sign-ext immediate << 2) + (PC+4)



beq extension (cont.)

- The offset indicates the number of instructions relative to branch insruction.
- Hence, the immediate must be sign-extended and multiplied by 4 to get the new program counter value:

  $PC' = PC + 4 + SignImm \times 4$.
- If SrcA-SrcB=0, Zero flag from ALU is 0
- A multiplexer is added to choose the address from PCPlus4 or PCBranch.

beq extension (cont.)

- PCBranch is selected if the instruction is a branch and the Zero flag is asserted.
  - ✓ Branch=1 and ALUControl=110 for beq instructions, Branch=0 for others.
  - ✓ ALUSrc=0 to choose SrcB from the register file.
  - ✓ RegWrite and MemWrite are 0, because a branch does not write to the register file or the memory.
  - ✓ The values of RegDst and MemtoReg are “don’t care”, because no write to RF takes place.

Single-Cycle datapath

- The single-cycle MIPS processor data-path is complete for a subset of instructions included.
- It can be extended later to include more instructions