E85 Digital Design & Computer Engineering

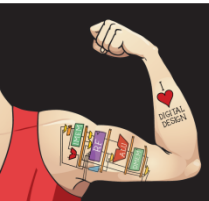
Lecture 19:

Single Cycle Processor Datapath

HARVEY MUDD COLLEGE

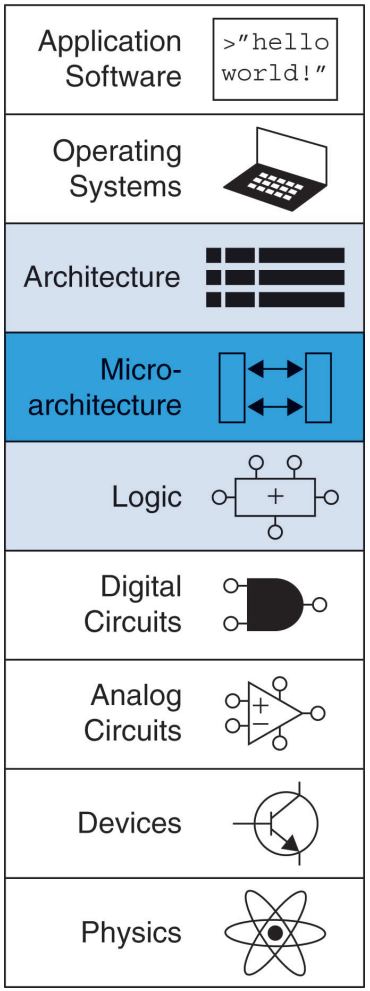
Lecture 19

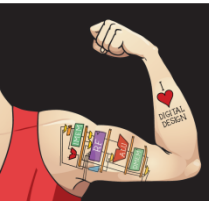
- **Single Cycle Processor Datapath**



Introduction

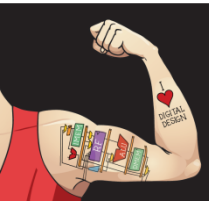
- **Microarchitecture:** how to implement an architecture in hardware
- Processor:
  - **Datapath:** functional blocks
  - **Control:** control signals





Microarchitecture

- Multiple implementations for a single architecture:
  - **Single-cycle:** Each instruction executes in a single cycle
  - **Multicycle:** Each instruction is broken up into series of shorter steps
  - **Pipelined:** Each instruction broken up into series of steps & multiple instructions execute at once

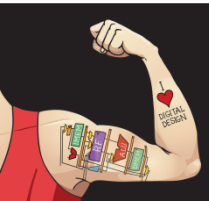


Processor Performance

- **Program execution time**

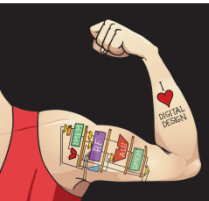
Execution Time = (#instructions)(cycles/instruction)(seconds/cycle)

- **Definitions:**
  - CPI: Cycles/instruction
  - clock period: seconds/cycle
  - IPC: instructions/cycle = IPC
- **Challenge is to satisfy constraints of:**
  - Cost
  - Power
  - Performance



ARM Processor

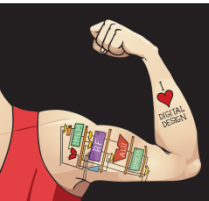
- Consider **subset** of ARM instructions:
  - **Data-processing instructions:**
    - **`ADD, SUB, AND, ORR`**
    - with register and immediate Src2, but **no shifts**
  - **Memory instructions:**
    - **`LDR, STR`**
    - with **positive immediate offset**
  - **Branch instructions:**
    - **`B`**



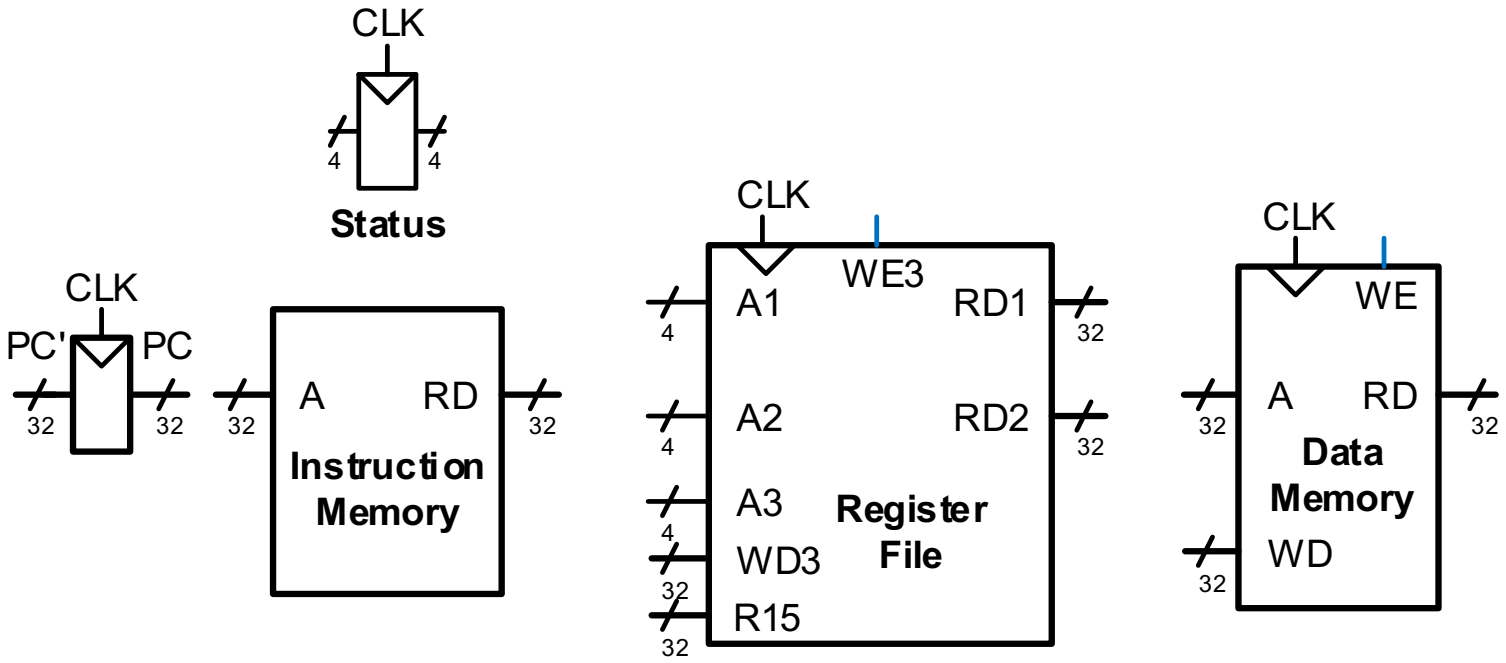
Architectural State Elements

Determines everything about a processor:

- Architectural state:
  - 16 registers (including PC)
  - Status register
- Memory

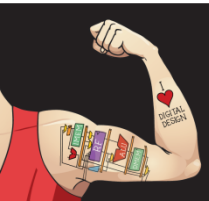


ARM Architectural State Elements



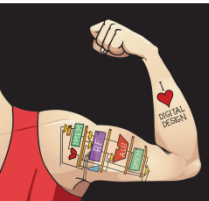
Single-Cycle ARM Processor

- Datapath
- Control



Single-Cycle ARM Processor

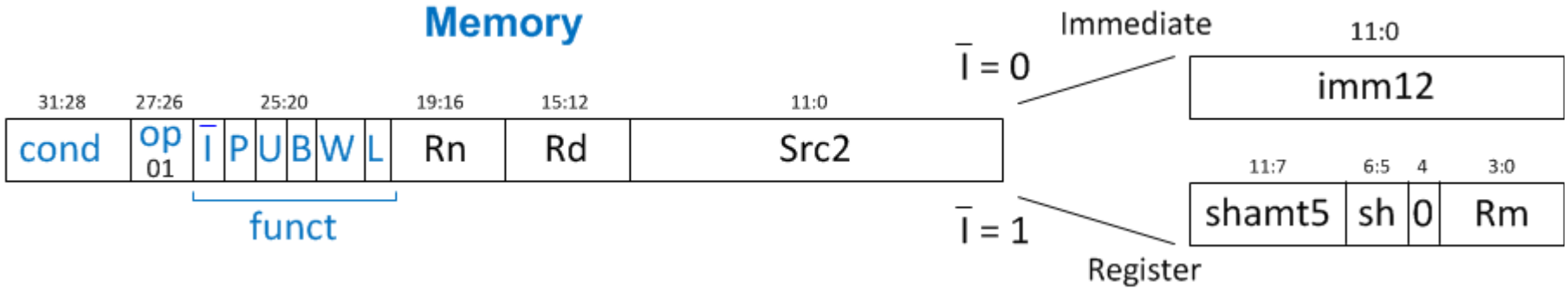
- **Datapath**
- Control

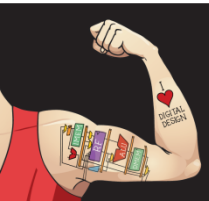


Single-Cycle ARM Processor

- **Datapath:** start with `LDR` instruction
- **Example:** `LDR R1, [R2, #5]`

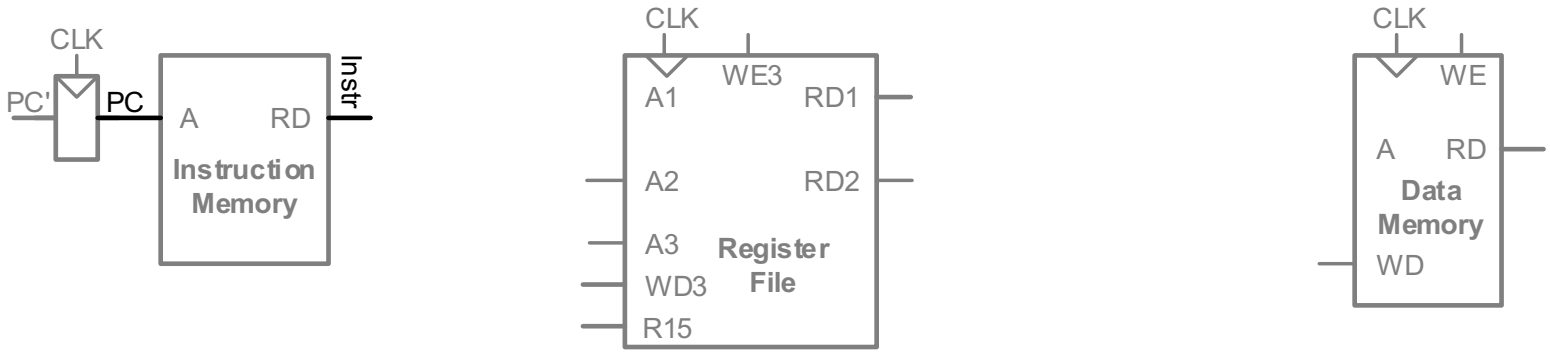
  `LDR Rd, [Rn, imm12]`





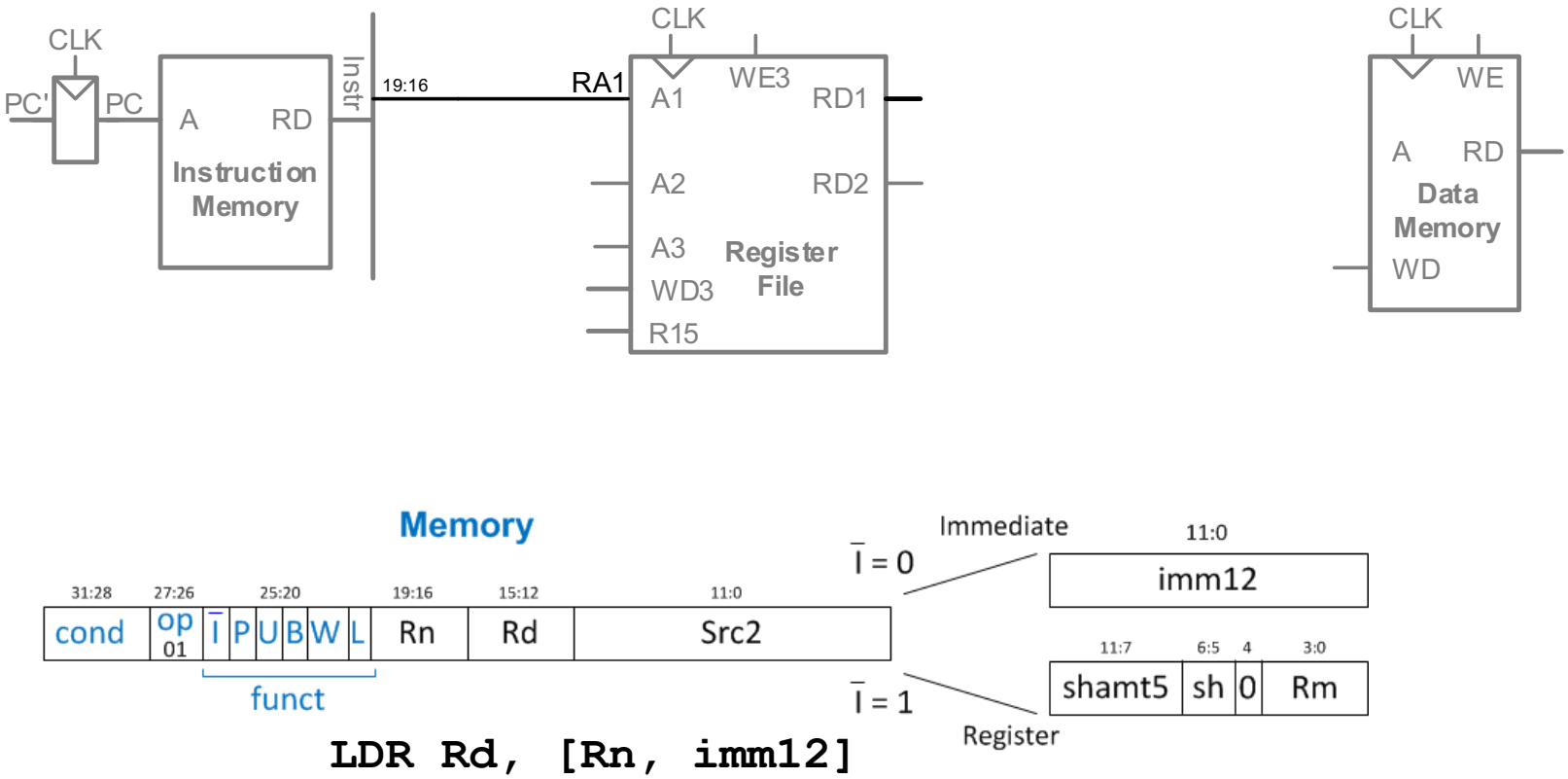
Single-Cycle Datapath: LDR fetch

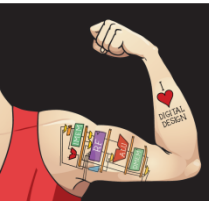
STEP 1: Fetch instruction



Single-Cycle Datapath: LDR Reg Read

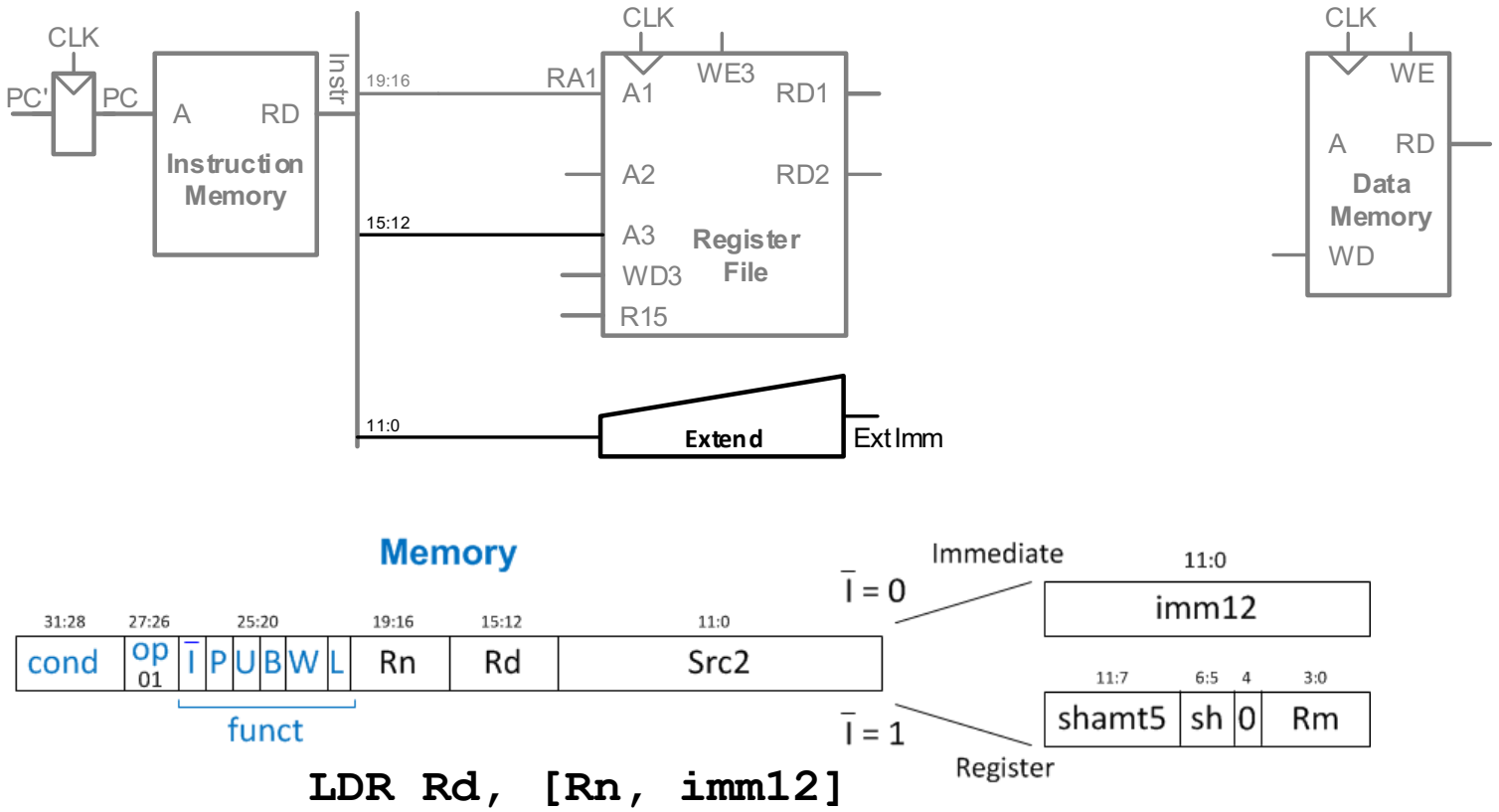
STEP 2: Read source operands from RF





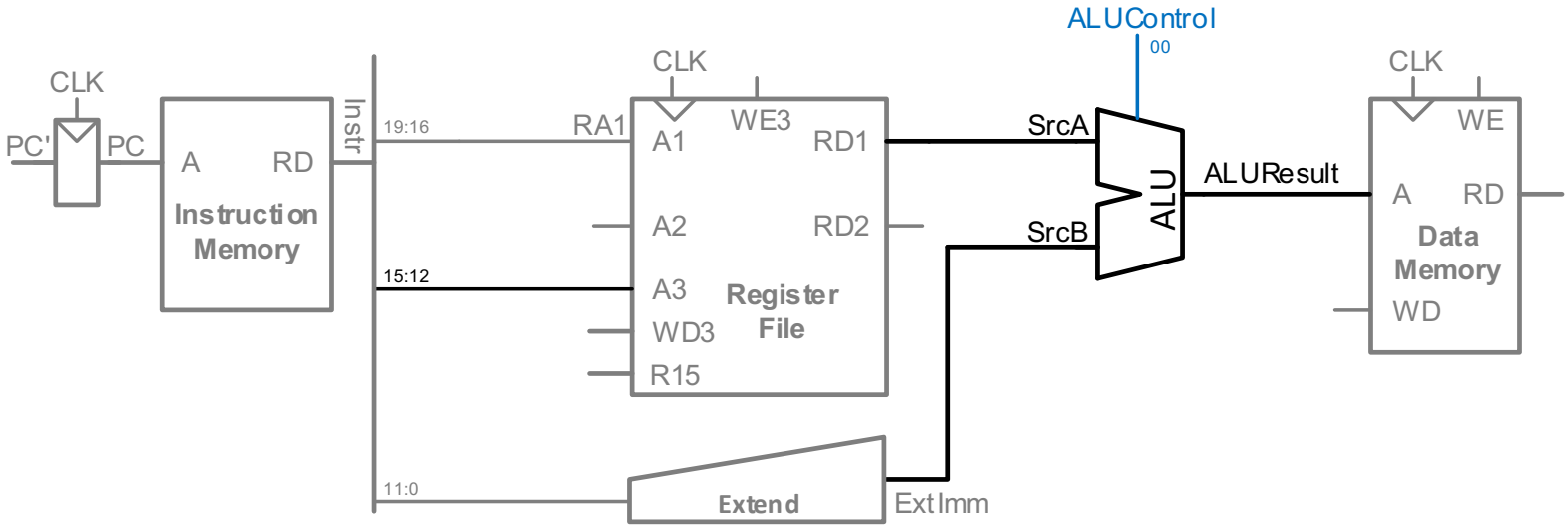
Single-Cycle Datapath: LDR Immed.

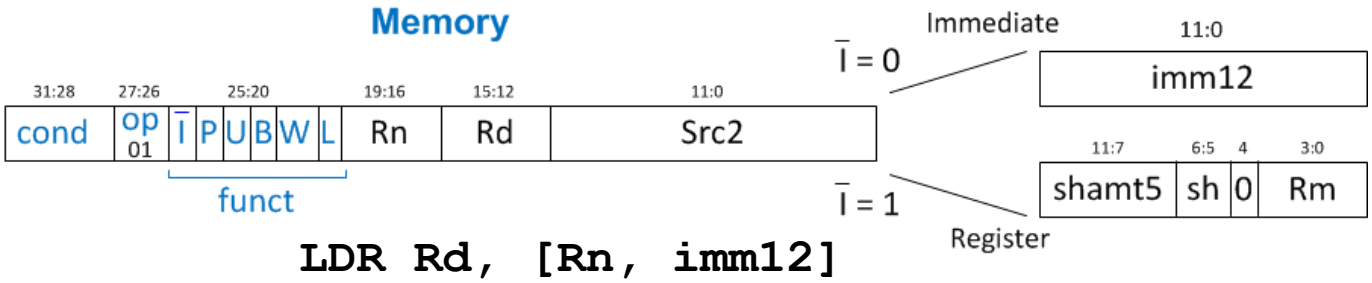
STEP 3: Extend the immediate



Single-Cycle Datapath: LDR Address

STEP 4: Compute the memory address

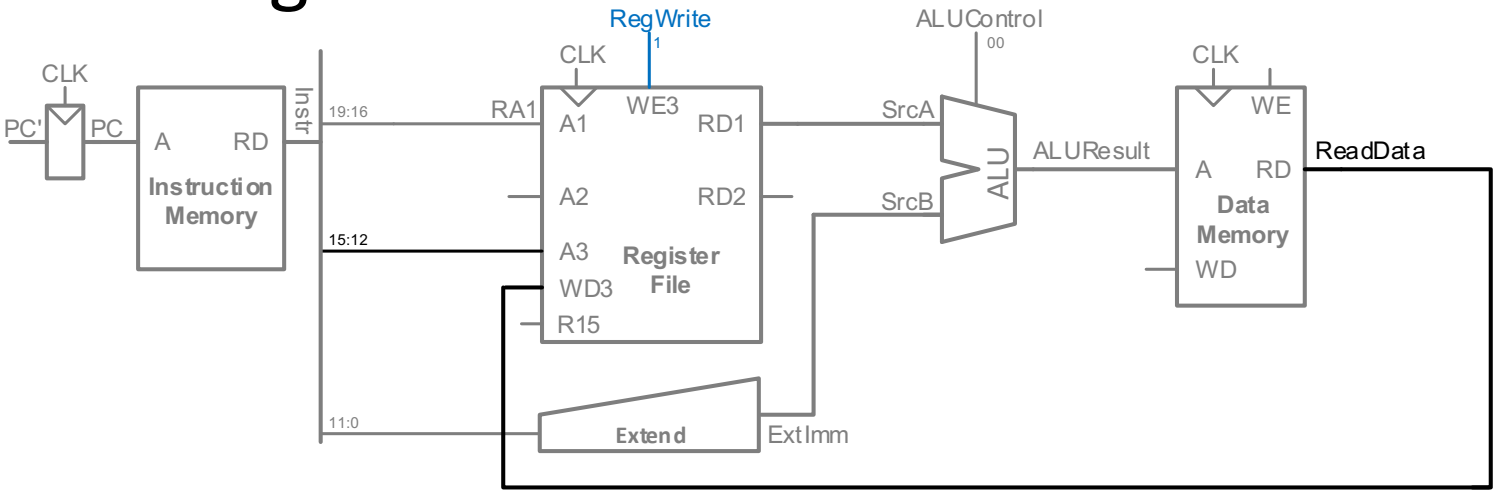


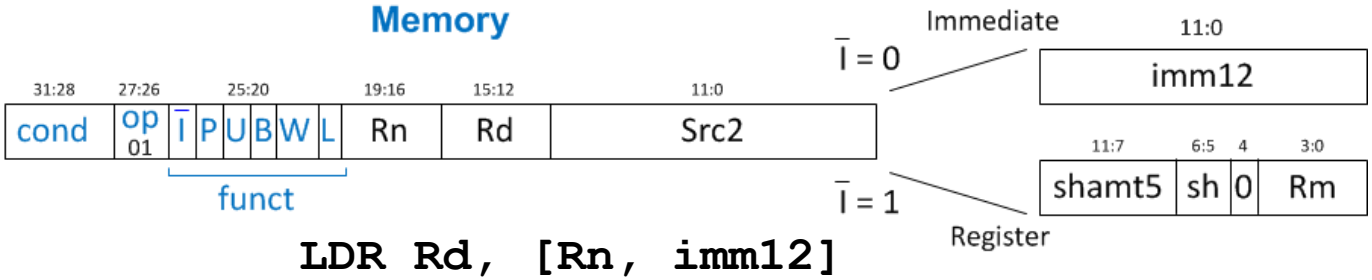



```
LDR Rd, [Rn, imm12]
```

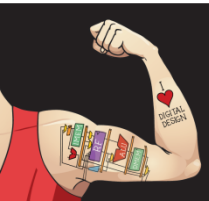
Single-Cycle Datapath: LDR Mem Read

STEP 5: Read data from memory and write it back to register file



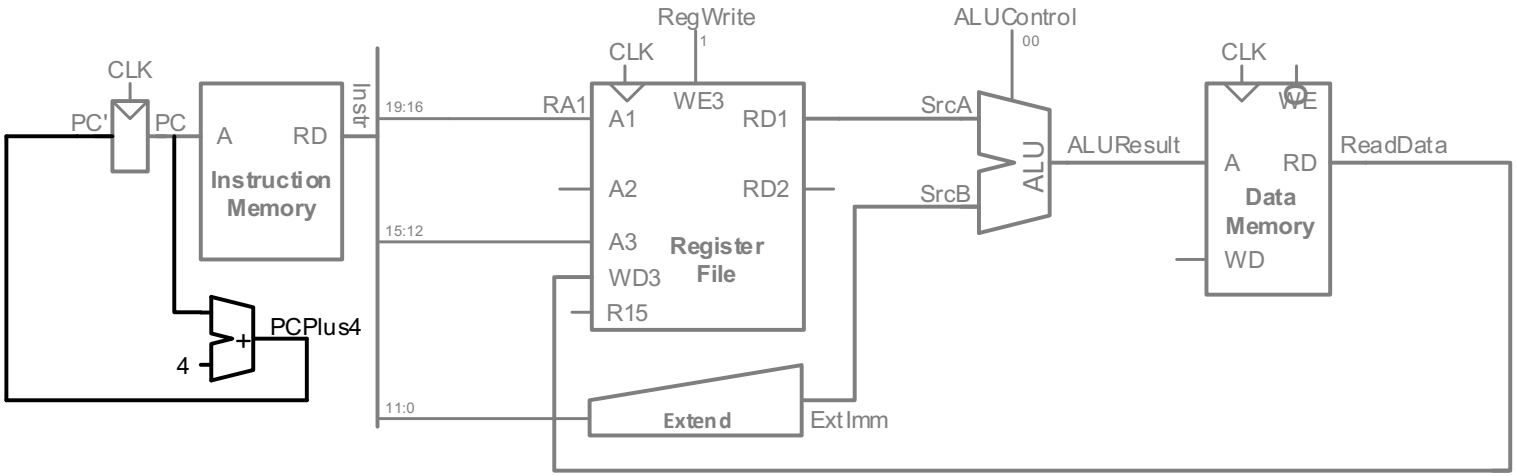


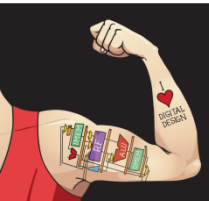
LDR Rd, [Rn, imm12]



Single-Cycle Datapath: PC Increment

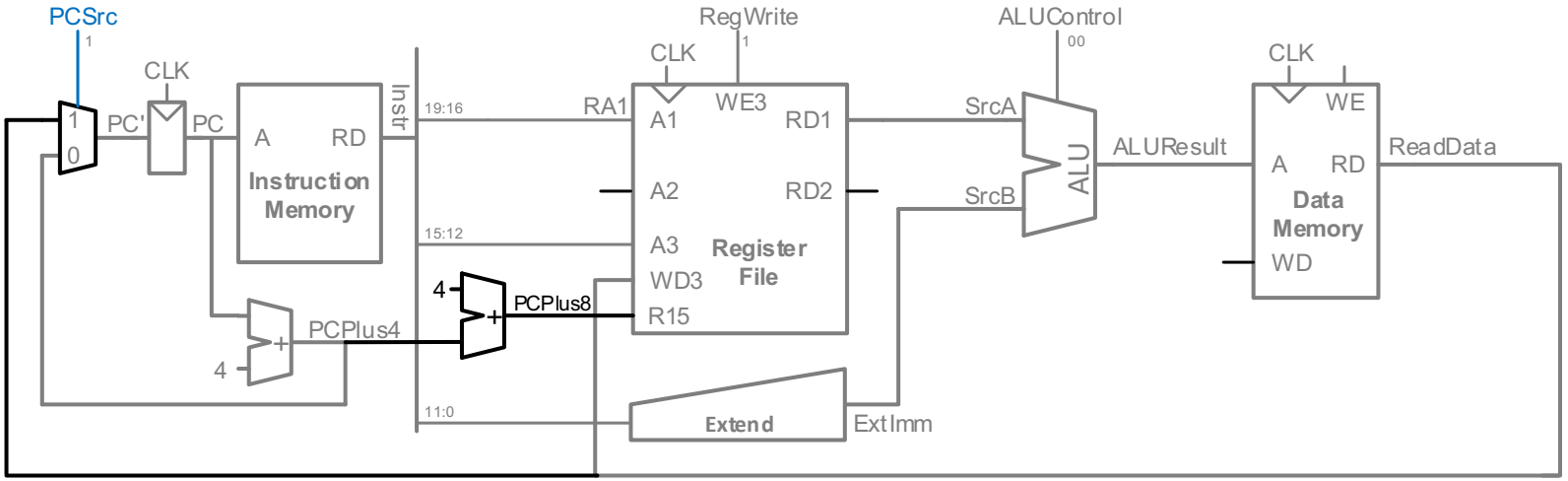
STEP 6: Determine address of next instruction





Single-Cycle Datapath: Access to PC

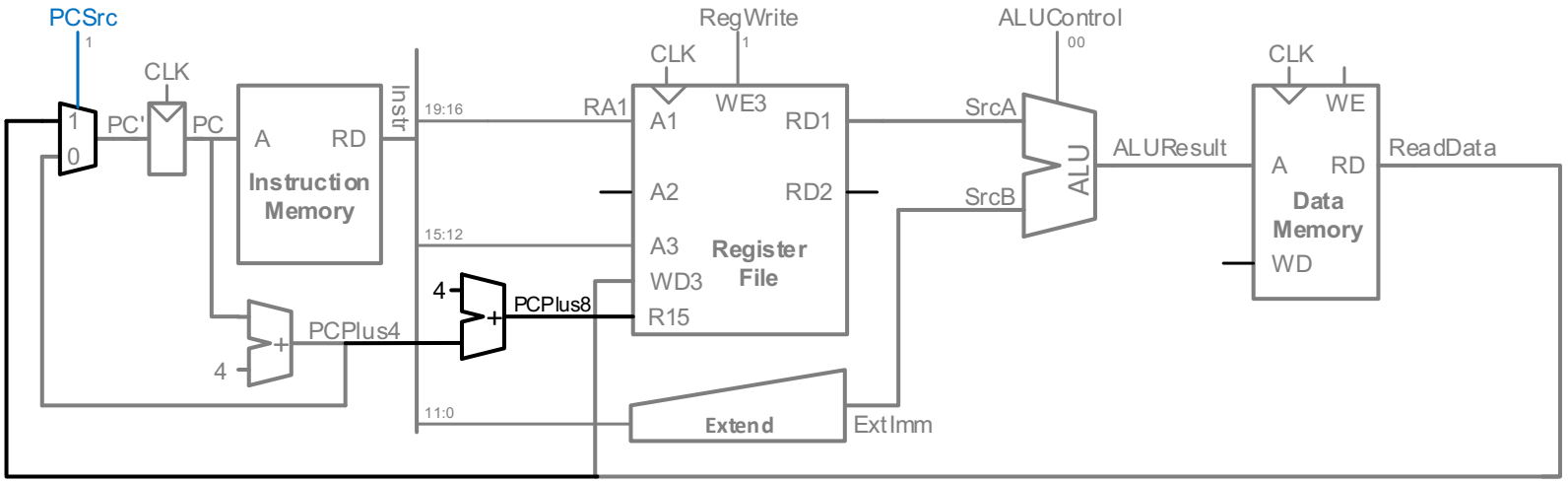
PC can be source/destination of instruction

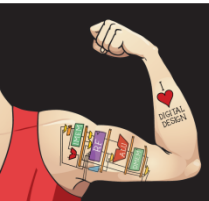


Single-Cycle Datapath: Access to PC

PC can be source/destination of instruction

- **Source:** R15 must be available in Register File
  - **PC** is read as the current **PC plus 8**

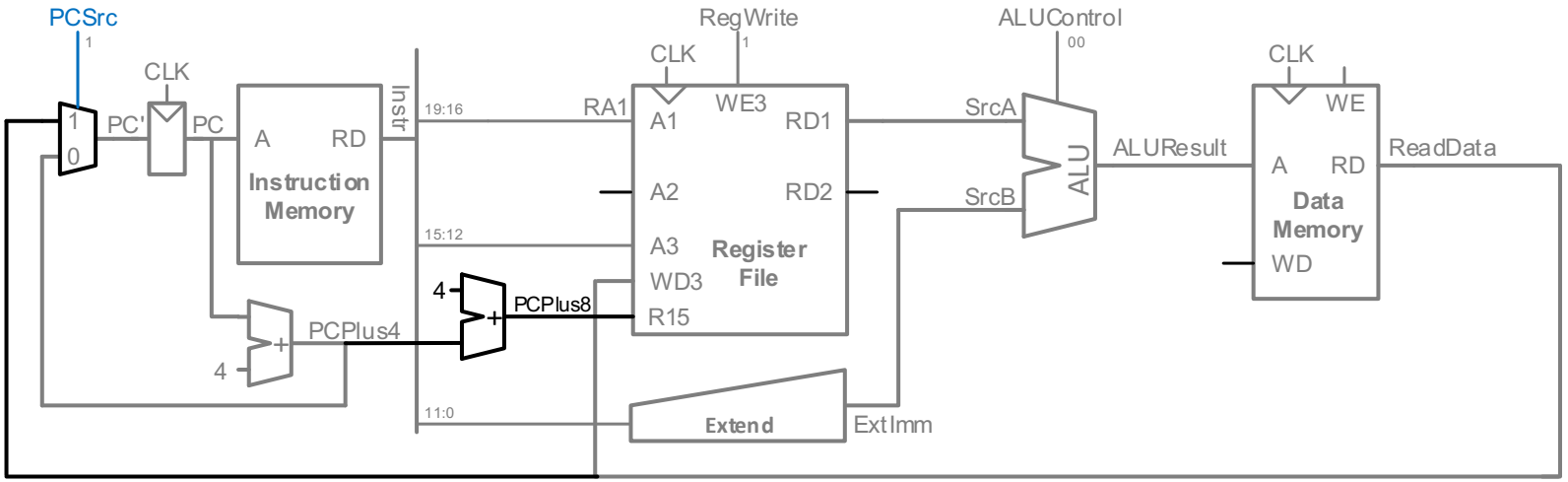




Single-Cycle Datapath: Access to PC

PC can be source/destination of instruction

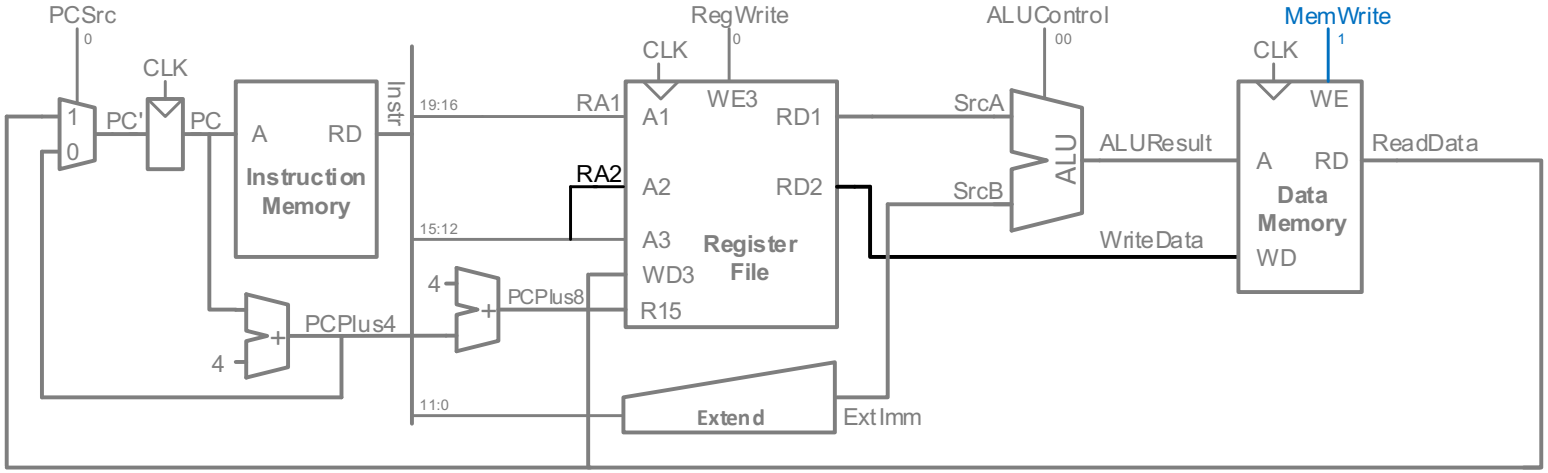
- **Source:** R15 must be available in Register File
  - **PC** is read as the current **PC plus 8**
- **Destination:** Be able to write result to PC

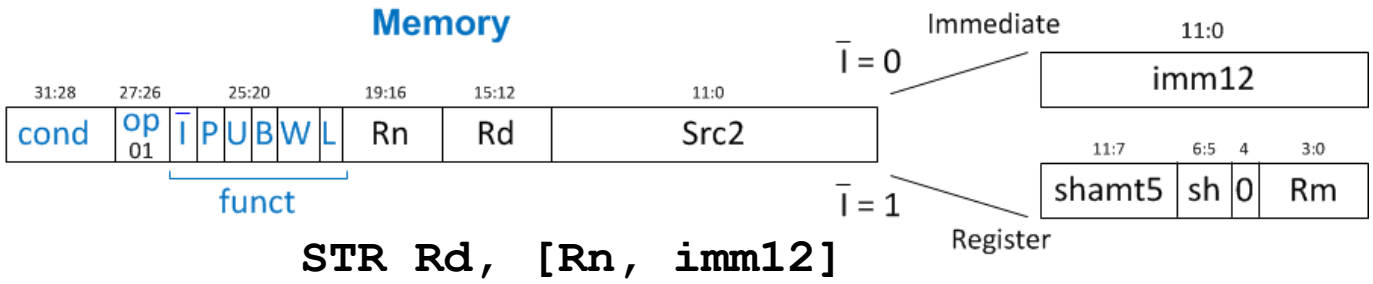


Single-Cycle Datapath: STR

Expand datapath to handle STR:

- Write data in Rd to memory



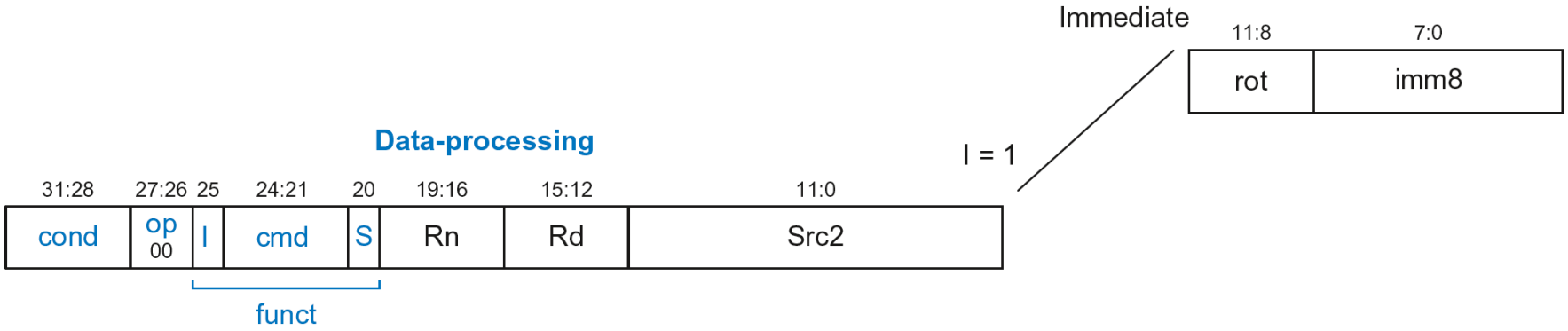


STR Rd, [Rn, imm12]

Single-Cycle Datapath: Data-processing

With immediate Src2:

- Read from `Rn` and `Imm8` (*ImmSrc* chooses the zero-extended `Imm8` instead of `Imm12`)
- Write *ALUResult* to register file
- Write to `Rd`

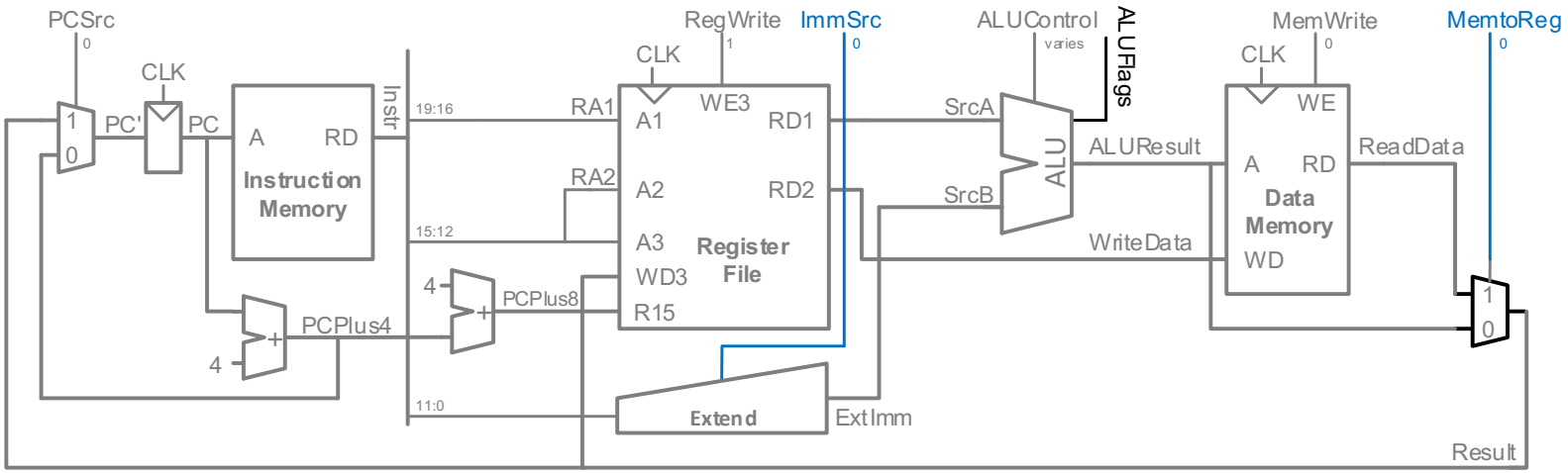


`ADD Rd, Rn, imm8`

Single-Cycle Datapath: Data-processing

With immediate Src2:

- Read from `Rn` and `Imm8` (*ImmSrc* chooses the zero-extended `Imm8` instead of `Imm12`)
- Write *ALUResult* to register file
- Write to `Rd`

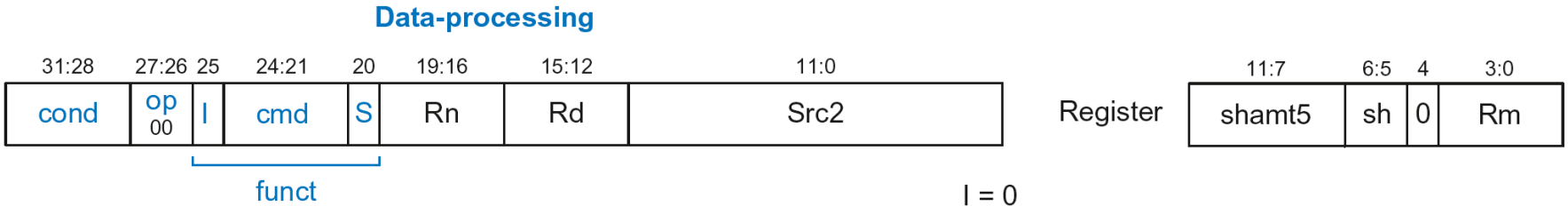


`ADD Rd, Rn, imm8`

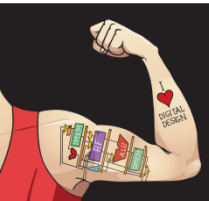
Single-Cycle Datapath: Data-processing

With register Src2:

- Read from `Rn` and `Rm` (instead of `Imm8`)
- Write *ALUResult* to register file
- Write to `Rd`



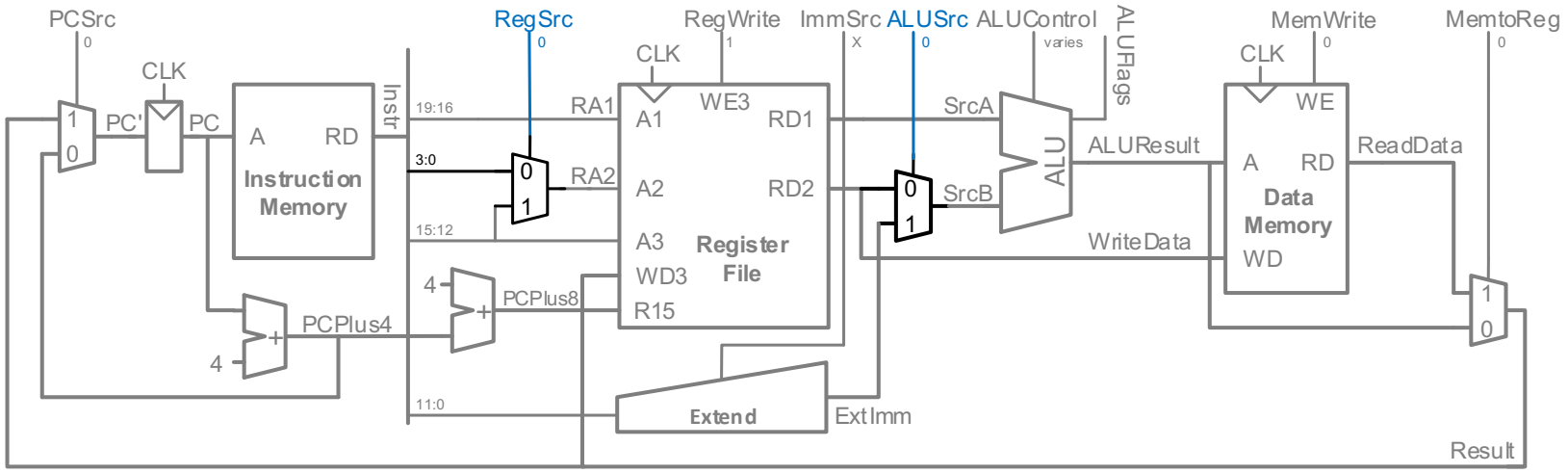
`ADD Rd, Rn, Rm`



Single-Cycle Datapath: Data-processing

With register Src2:

- Read from `Rn` and `Rm` (instead of `Imm8`)
- Write *ALUResult* to register file
- Write to `Rd`



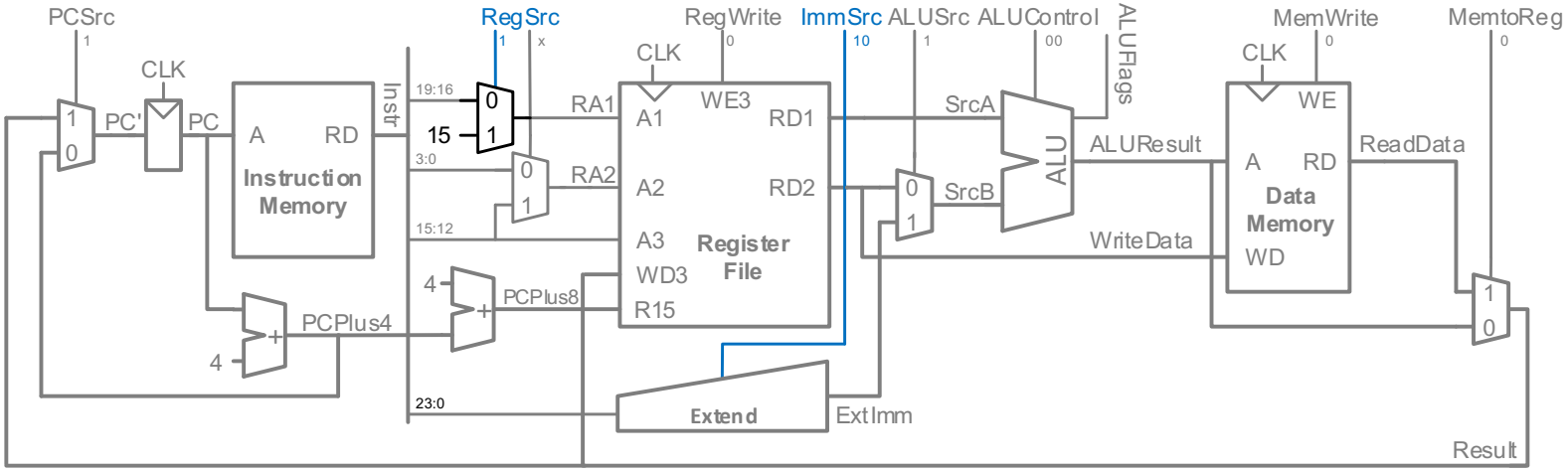
`ADD Rd, Rn, Rm`

Single-Cycle Datapath: B

Calculate branch target address:

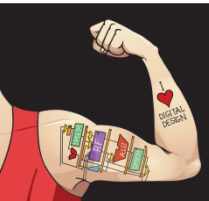
BTA = (*ExtImm*) + (PC + 8)

ExtImm = Imm24 << *2* and sign-extended

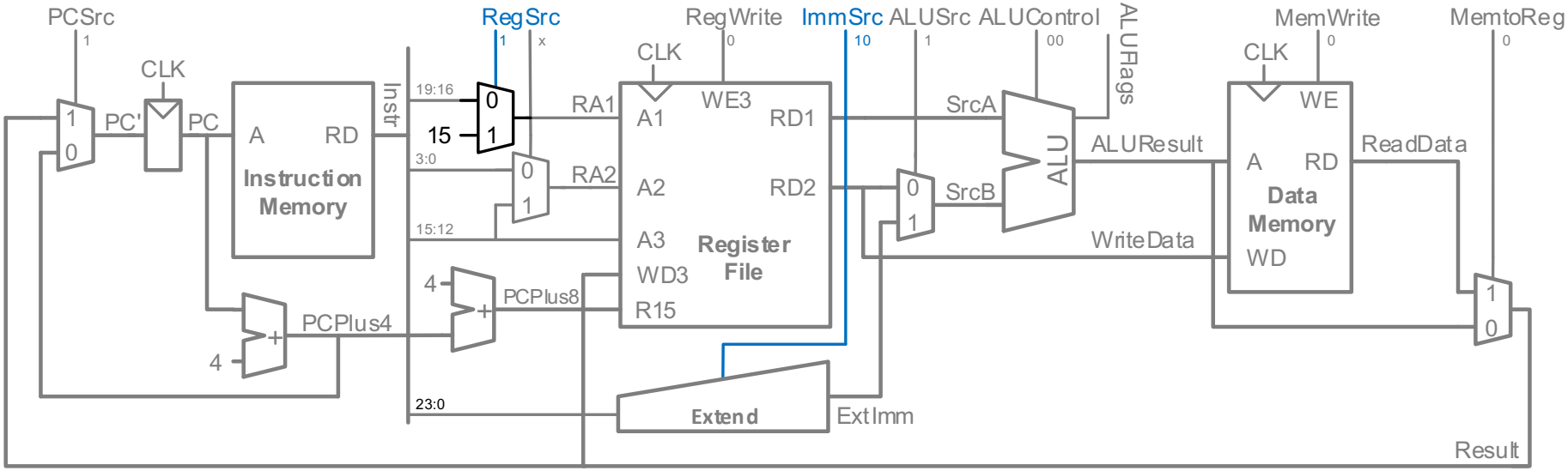


Branch

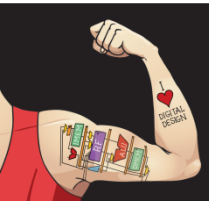
`B Label`



Single-Cycle Datapath: ExtImm



$ImmSrc_{1:0}$	ExtImm	Description
00	{24'b0, $Instr_{7:0}$}	Zero-extended *imm8*
01	{20'b0, $Instr_{11:0}$}	Zero-extended *imm12*
10	{6{$Instr_{23}$}, $Instr_{23:0}$}	Sign-extended *imm24*



Single-Cycle ARM Processor