

ENGN1640: Design of Computing Systems

Topic 04: Single-Cycle Processor Design

Professor Sherief Reda

<http://scale.engin.brown.edu>

Electrical Sciences and Computer Engineering

School of Engineering

Brown University

Spring 2017



[material from Harris]

Processor organization (microarchitecture)

- Multiple implementations for a single architecture:
 - Single-cycle
 - Each instruction executes in a single cycle
 - Multi-cycle
 - Each instruction is broken up into a series of shorter steps
 - Pipelined
 - Each instruction is broken up into a series of steps
 - Multiple instructions execute at once.
 - Superscalar
 - Multiple instructions fetched, decoded and executed simultaneously.

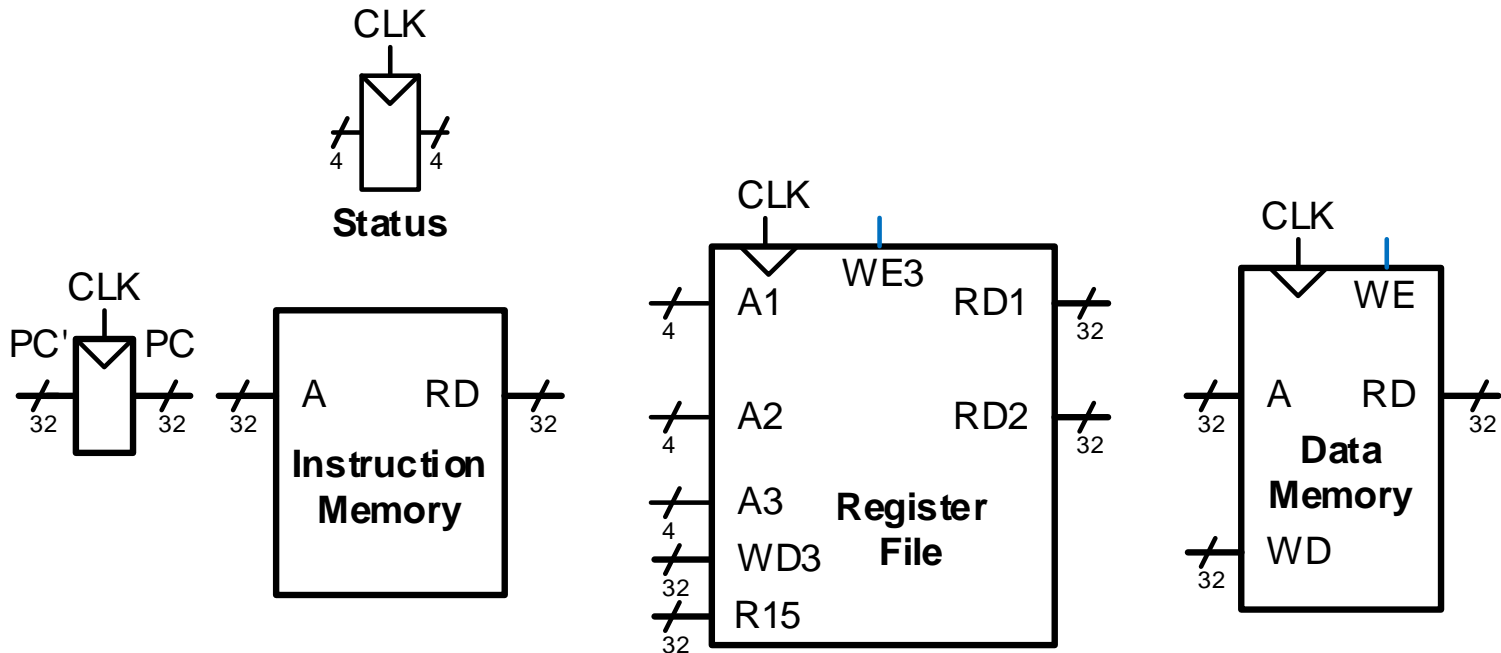
Application Software	programs
Operating Systems	device drivers
Architecture	instructions registers
Micro-architecture	datapaths controllers
Logic	adders memories
Digital Circuits	AND gates NOT gates
Analog Circuits	amplifiers filters
Devices	transistors diodes
Physics	electrons

Introduction

- CPU performance factors
 - Instruction count
 - Determined by ISA and compiler
 - CPI and Cycle time
 - Determined by CPU hardware
- We will examine a number of ARM implementations
 - A simplified single-cycle version
 - A more realistic pipelined version
- Simple subset, shows most aspects
 - Memory reference: **LDR, STR**
 - Arithmetic/logical: **ADD, SUB, AND, ORR**
 - Control transfer: **B**
- Will not implement in lab post-index, pre-index or shifted operands.

Architectural state

- Determines everything about a processor:
 - PC
 - 16 registers
 - Memory



Single-Cycle ARM Processor

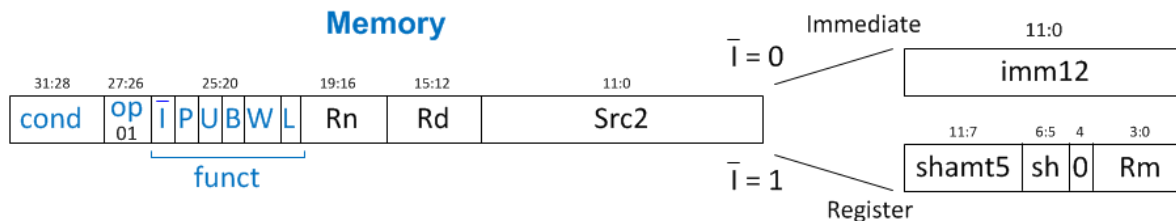
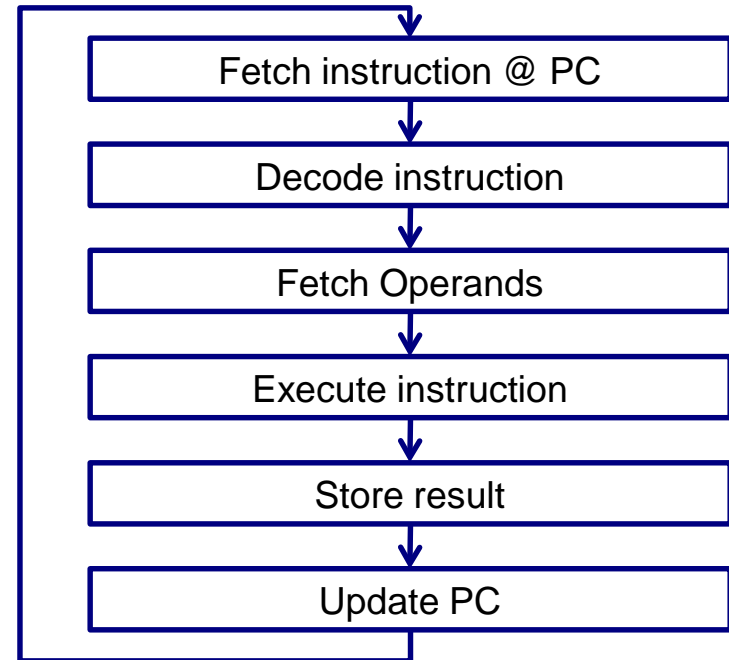
- Datapath
- Control

Datapath: start with LDR instruction

- **Example:**

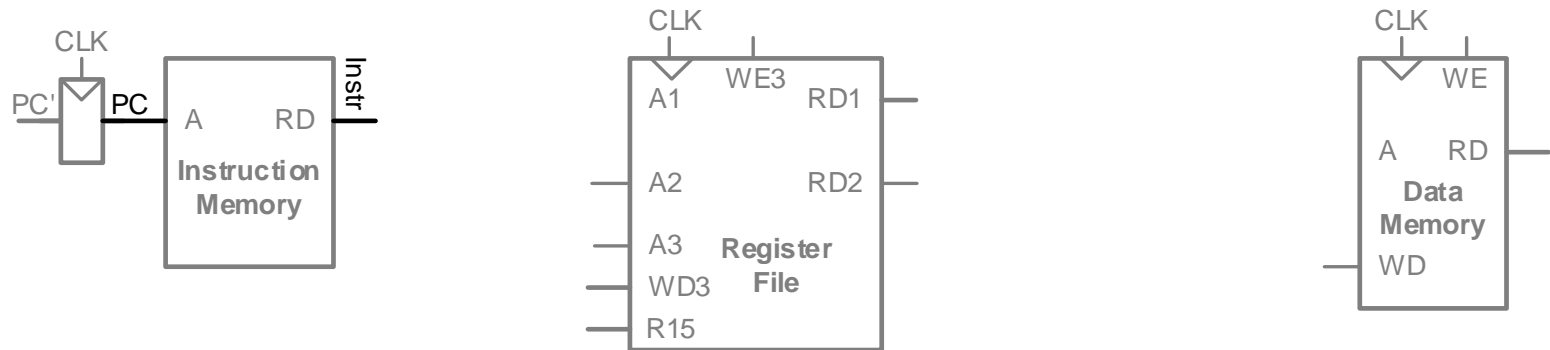
LDR R1, [R2, #5]

LDR Rd, [Rn, imm12]



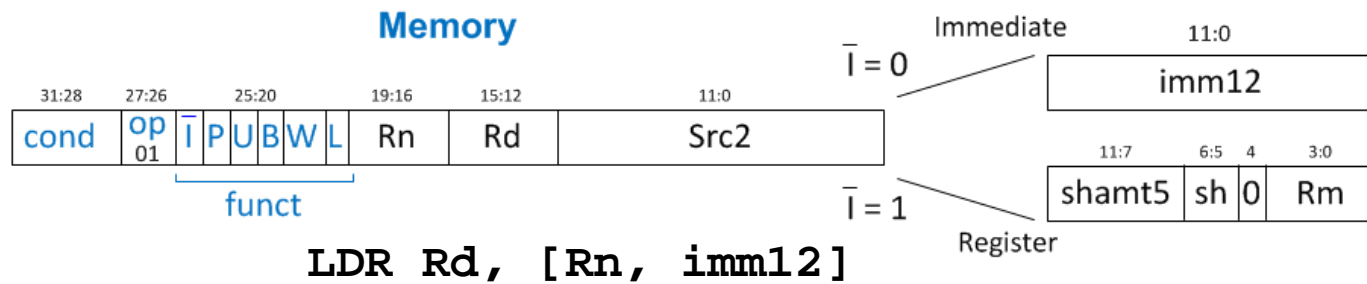
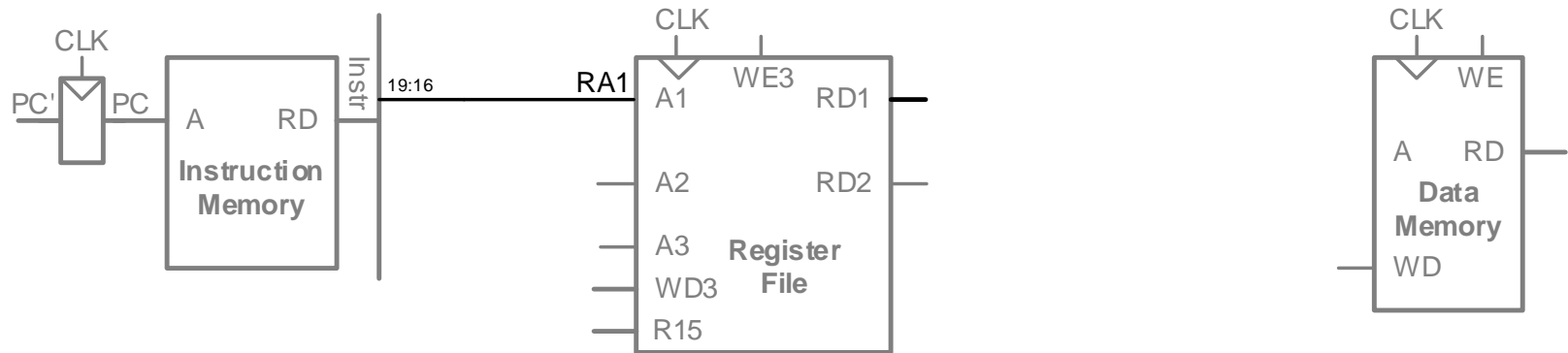
Single-Cycle Datapath: LDR fetch

- **STEP 1:** Fetch instruction



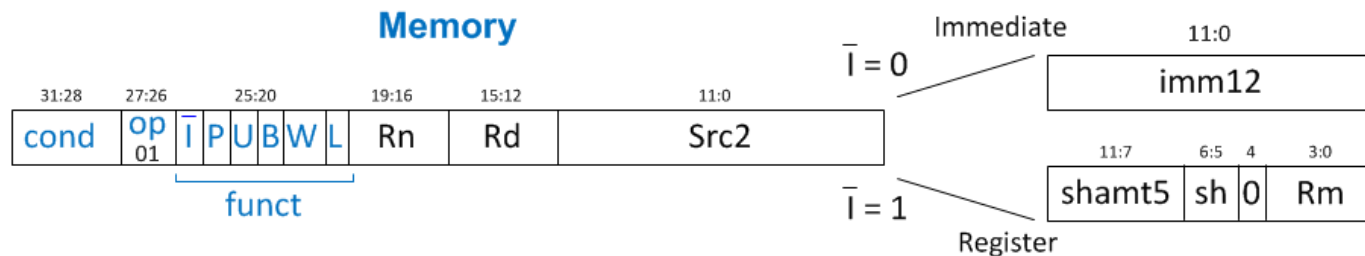
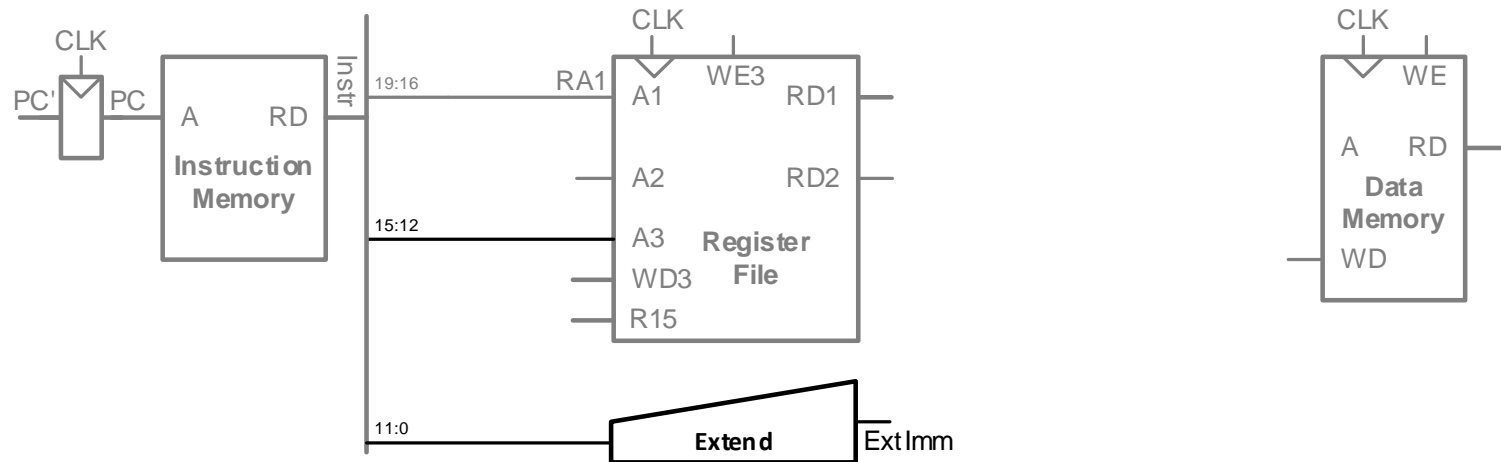
Single-Cycle Datapath: LDR register read

STEP 2: Read source operands from RF



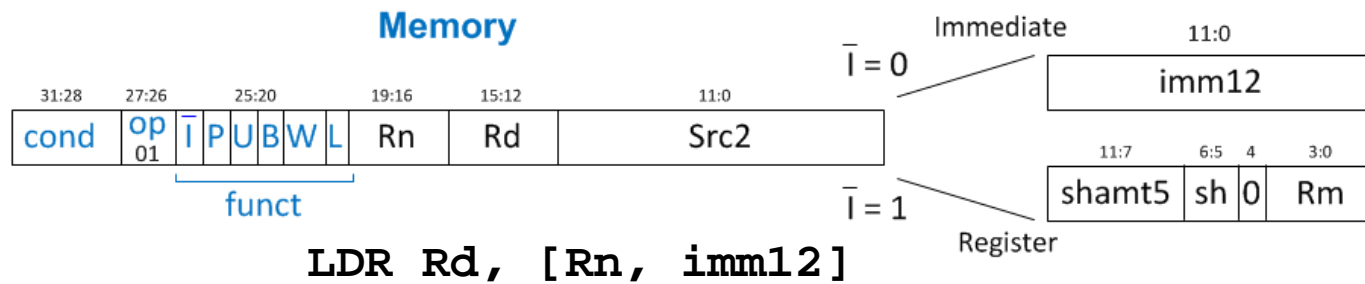
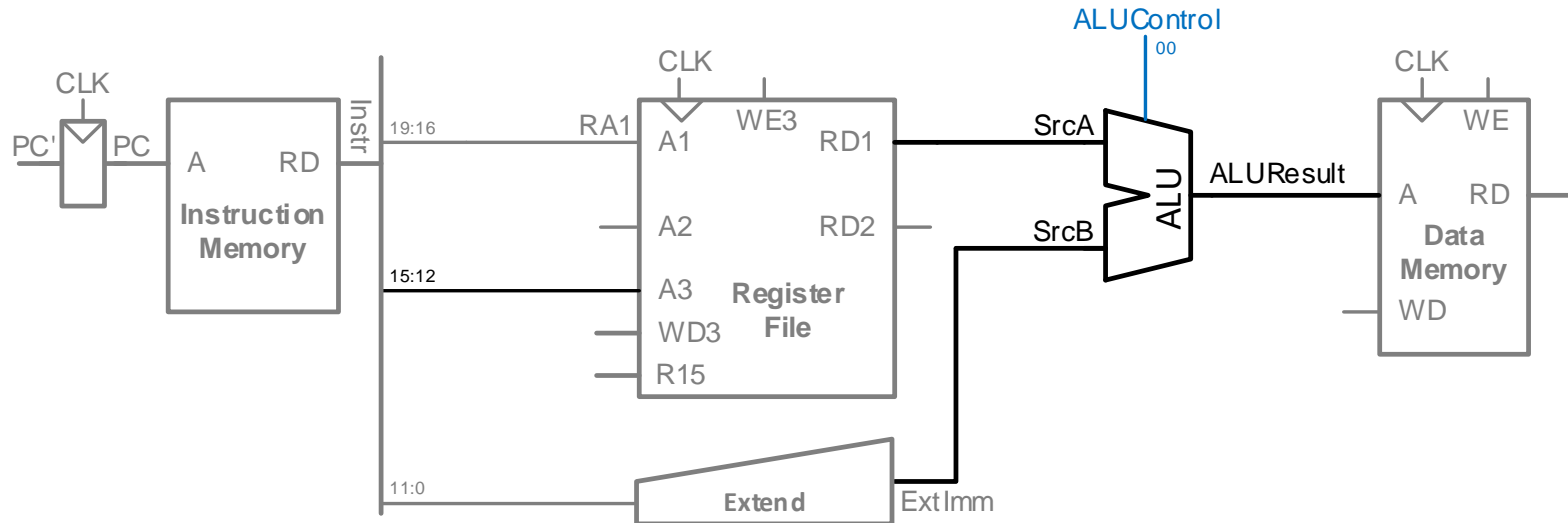
Single-Cycle Datapath: LDR immediate

STEP 3: Extend the immediate



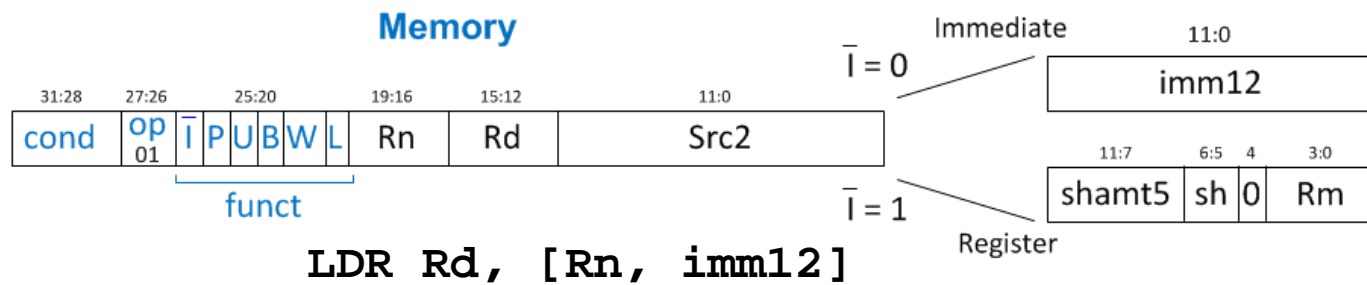
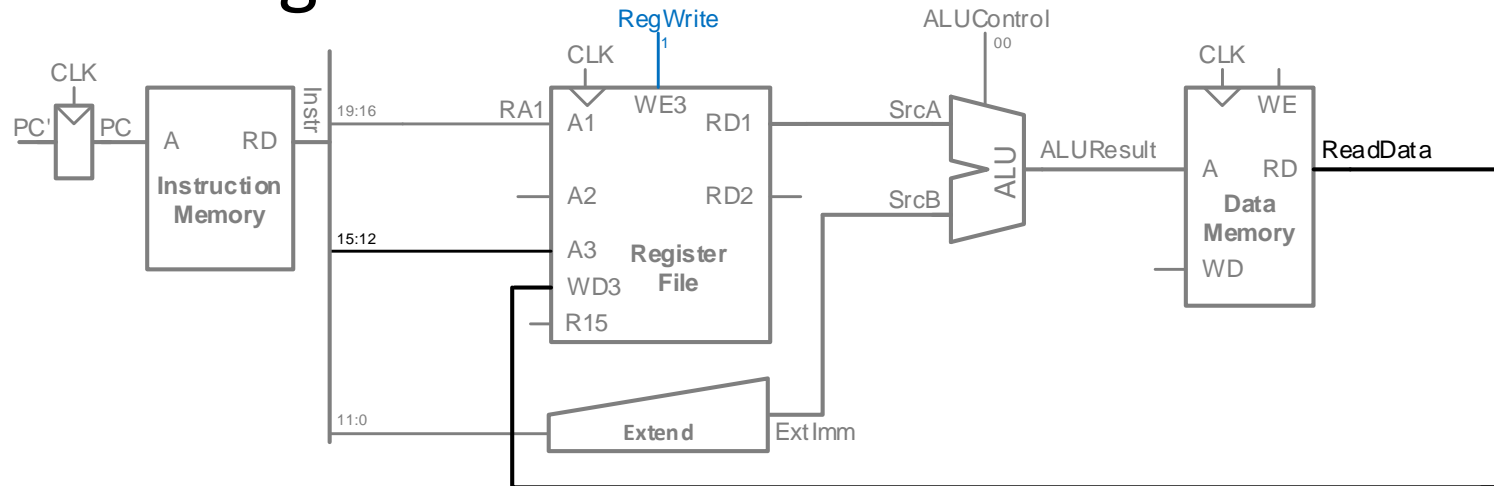
Single-Cycle Datapath: LDR address

STEP 4: Compute the memory address



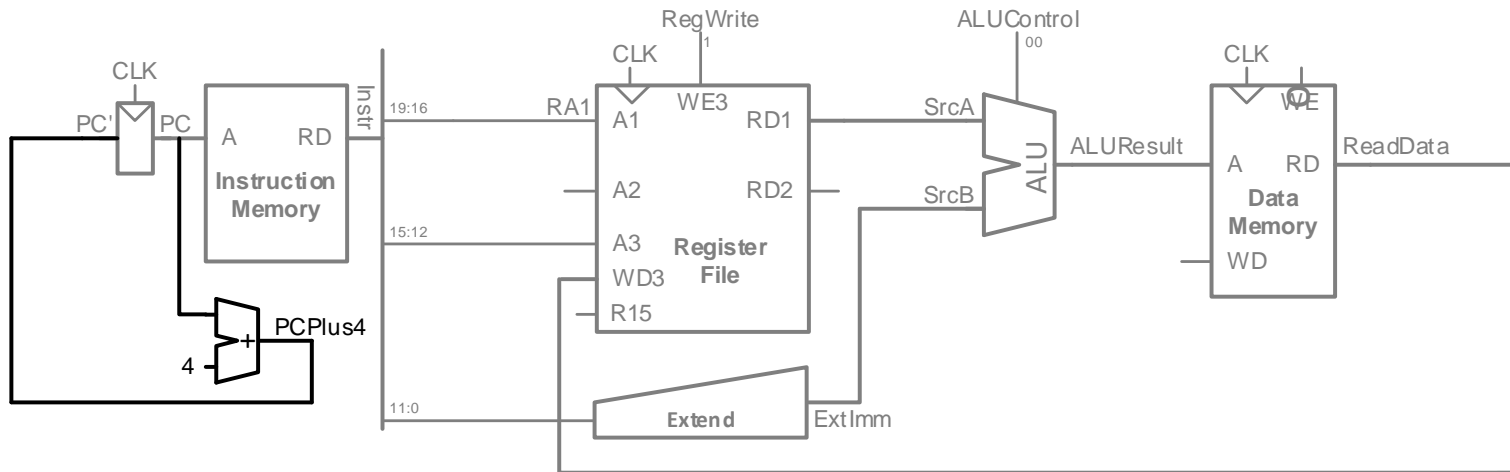
Single-Cycle Datapath: LDR memory read

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



Single-Cycle Datapath: LDR PC increment

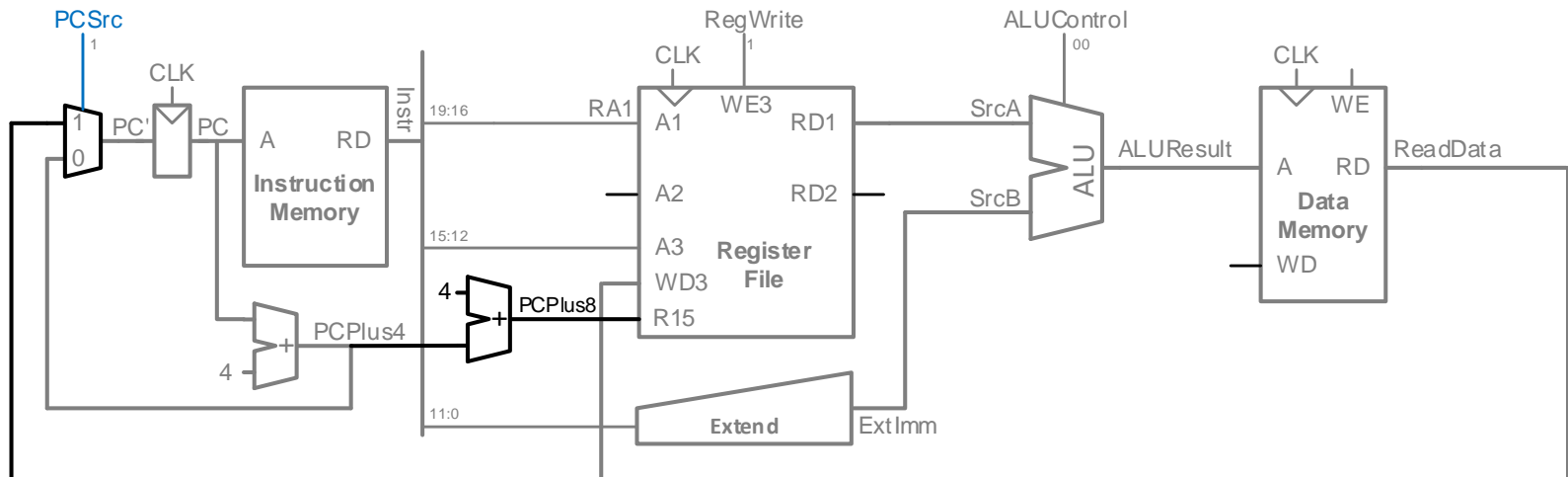
STEP 6: Determine address of next instruction



Single-Cycle Datapath: Access to PC

PC can be source/destination of instruction

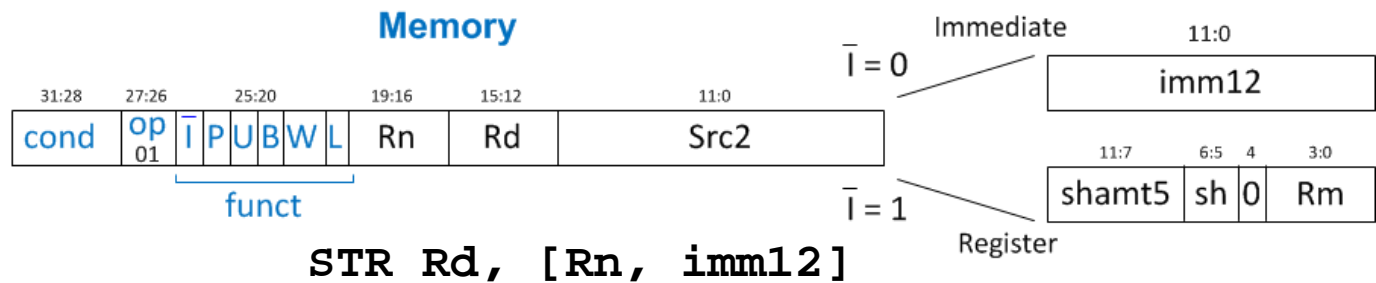
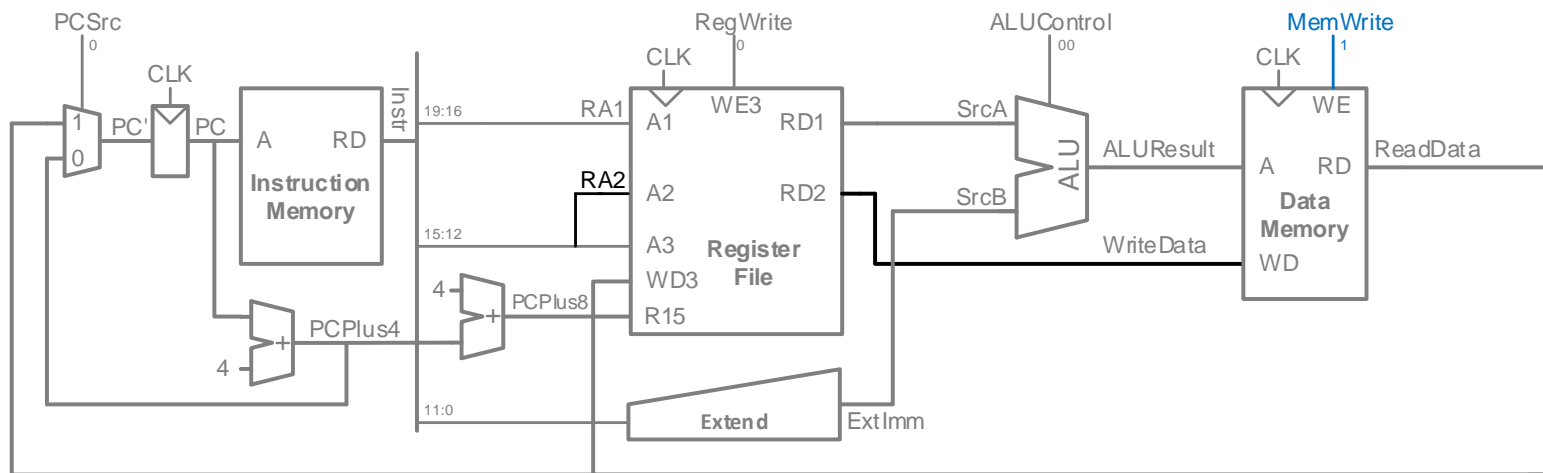
- **Source:** R15 must be available in Register File
 - **PC** is read as the current **PC plus 8**
- **Destination:** Be able to write result to PC



Single-Cycle Datapath: STR

Expand datapath to handle STR:

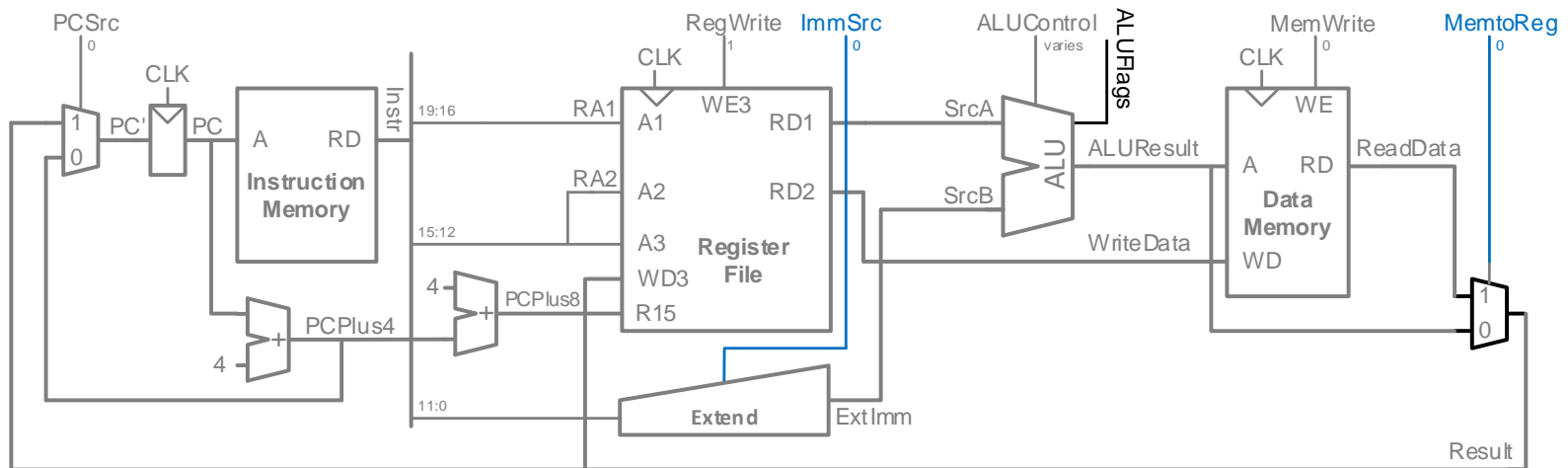
- Write data in Rd to memory



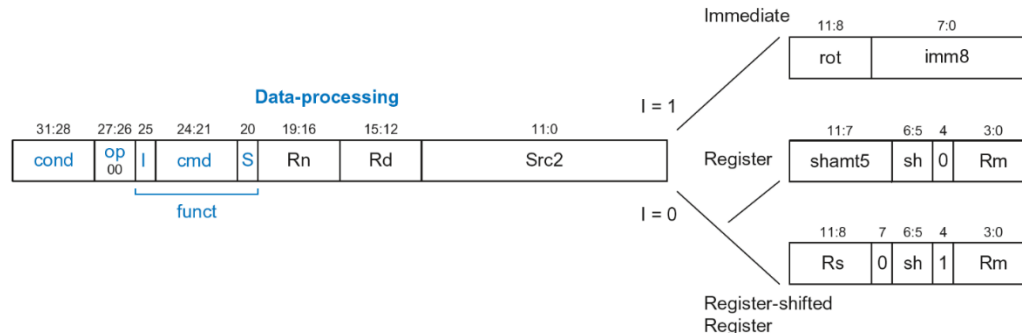
Single-Cycle Datapath: data processing

With **immediate Src2**:

- Read from R_n and $Imm8$ ($ImmSrc$ chooses the zero-extended $Imm8$ instead of $Imm12$)
- Write $ALUResult$ to register file
- Write to R_d



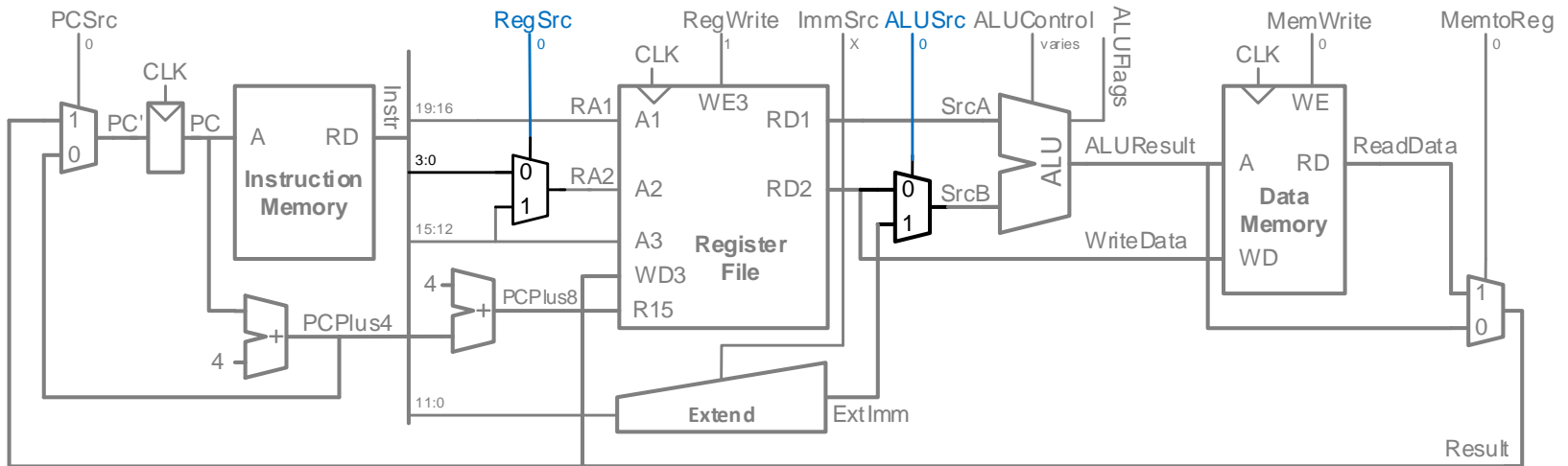
ADD R_d , R_n , $imm8$



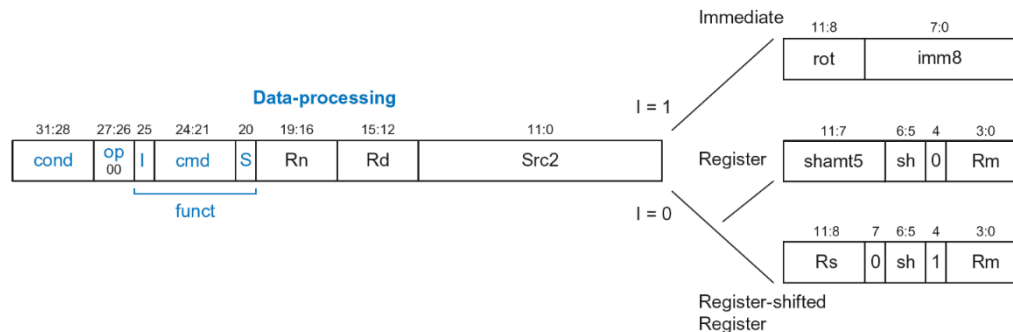
Single-Cycle Datapath: data processing

With register Src2:

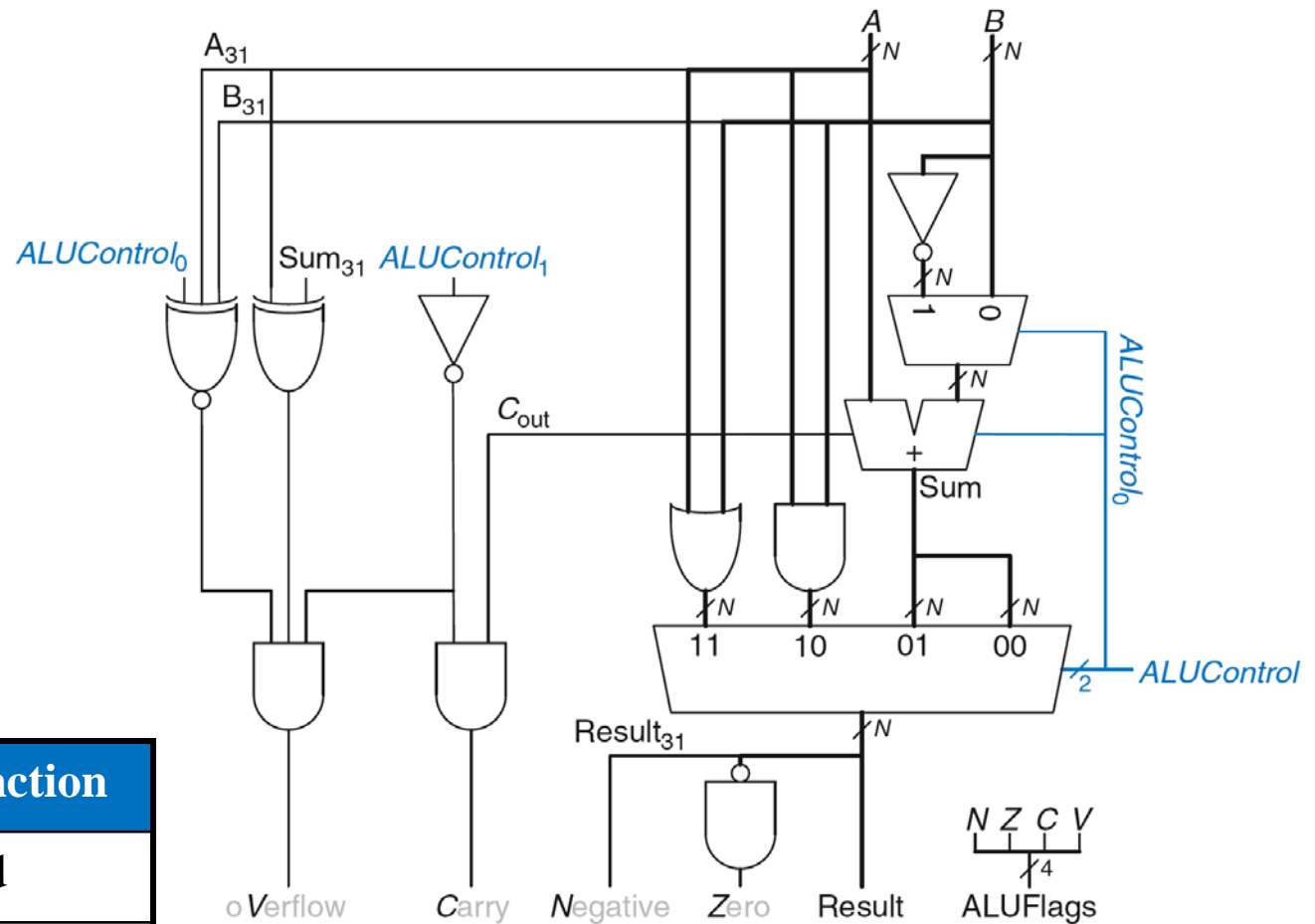
- Read from Rn and Rm (instead of Imm8)
- Write *ALUResult* to register file
- Write to Rd



ADD Rd, Rn, Rm



ALU



ALUControl _{1:0}	Function
00	Add
01	Subtract
10	AND
11	OR

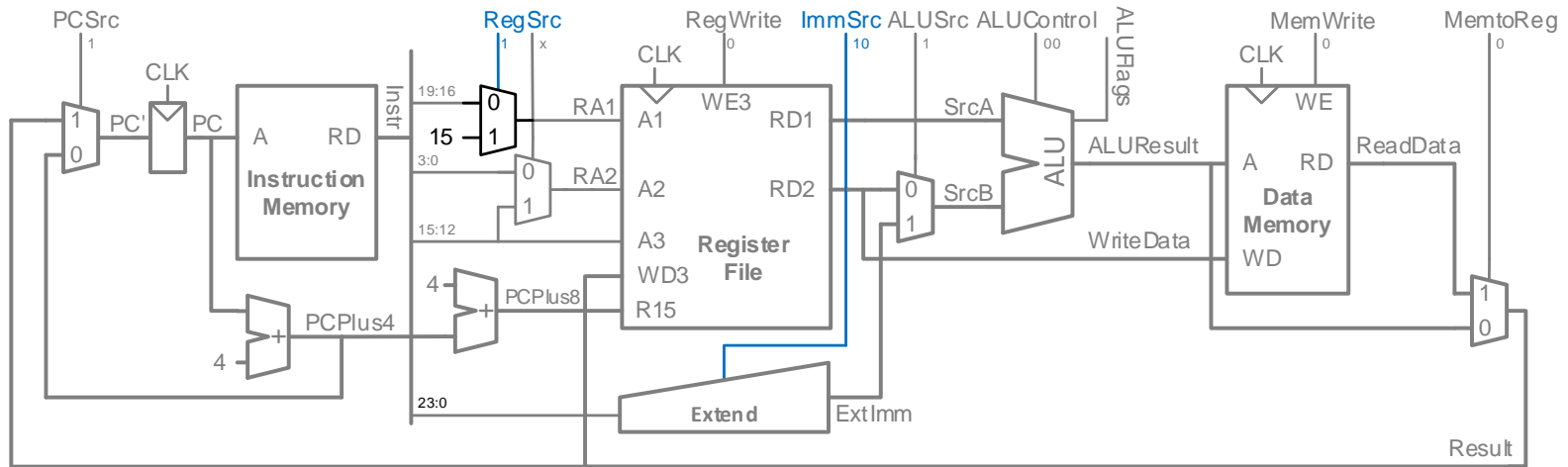
Flags must be stored in FFs if required by by instruction (e.g., CMP, TST, SUBS, ADDS, etc)

Single-Cycle Datapath: B

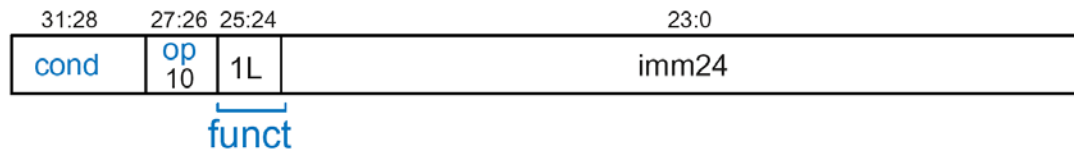
Calculate branch target address:

$$\text{BTA} = (\text{ExtImm}) + (\text{PC} + 8)$$

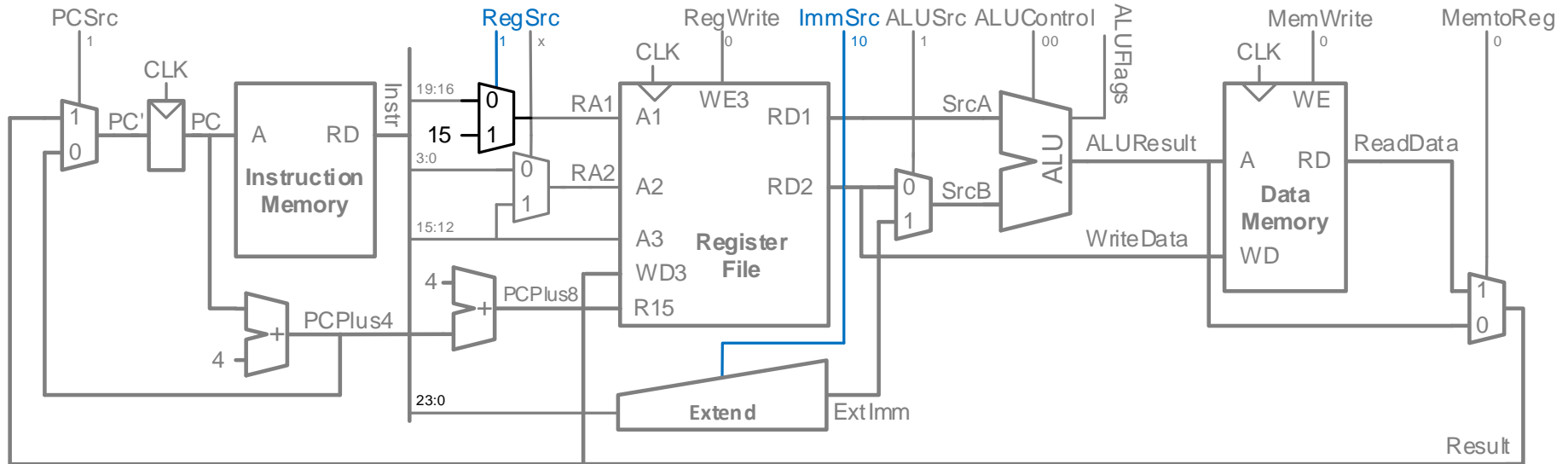
$$\text{ExtImm} = \text{Imm24} \ll 2 \text{ and sign-extended}$$



Branch

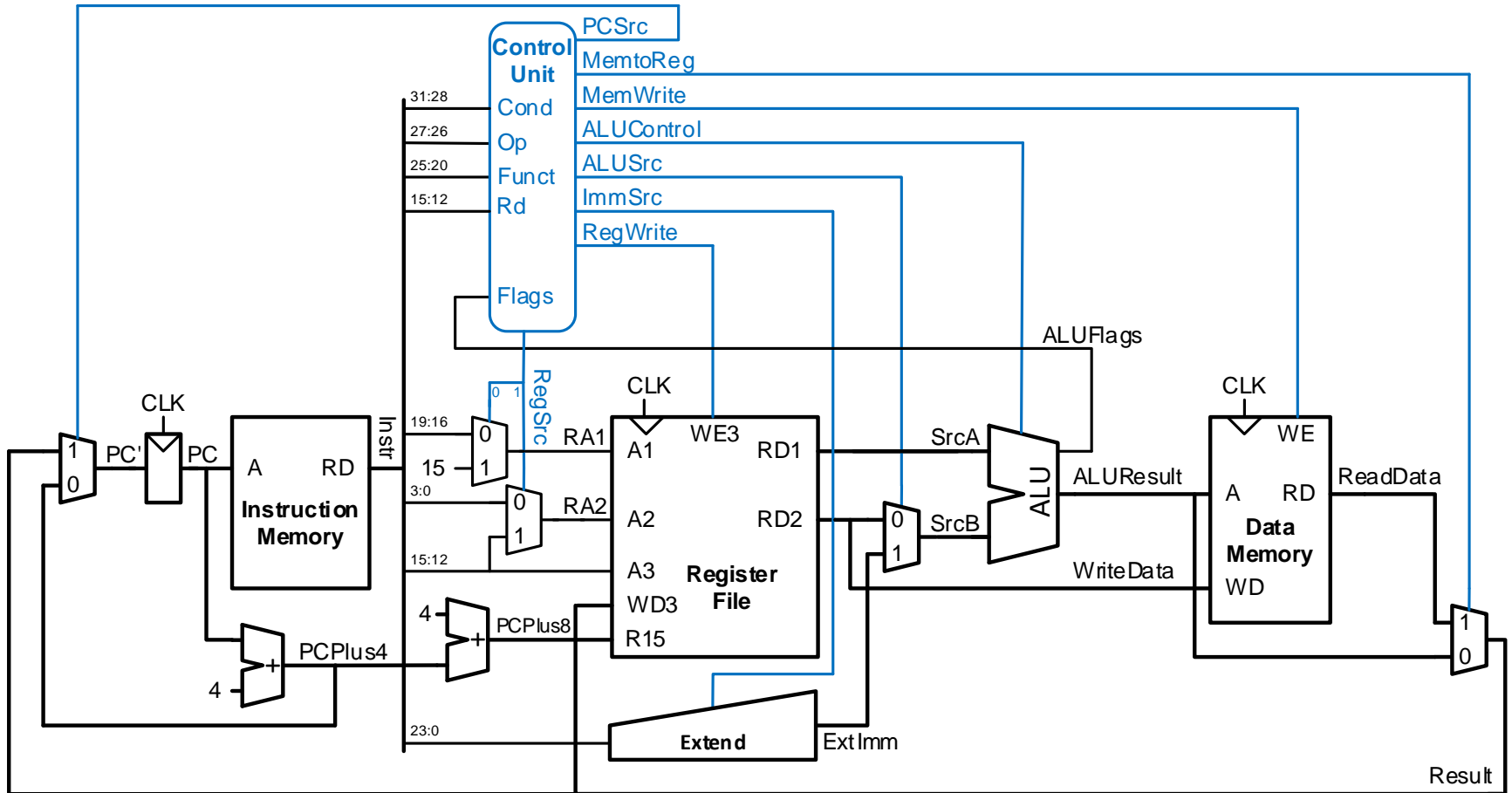


Single-Cycle Datapath: ExtImm

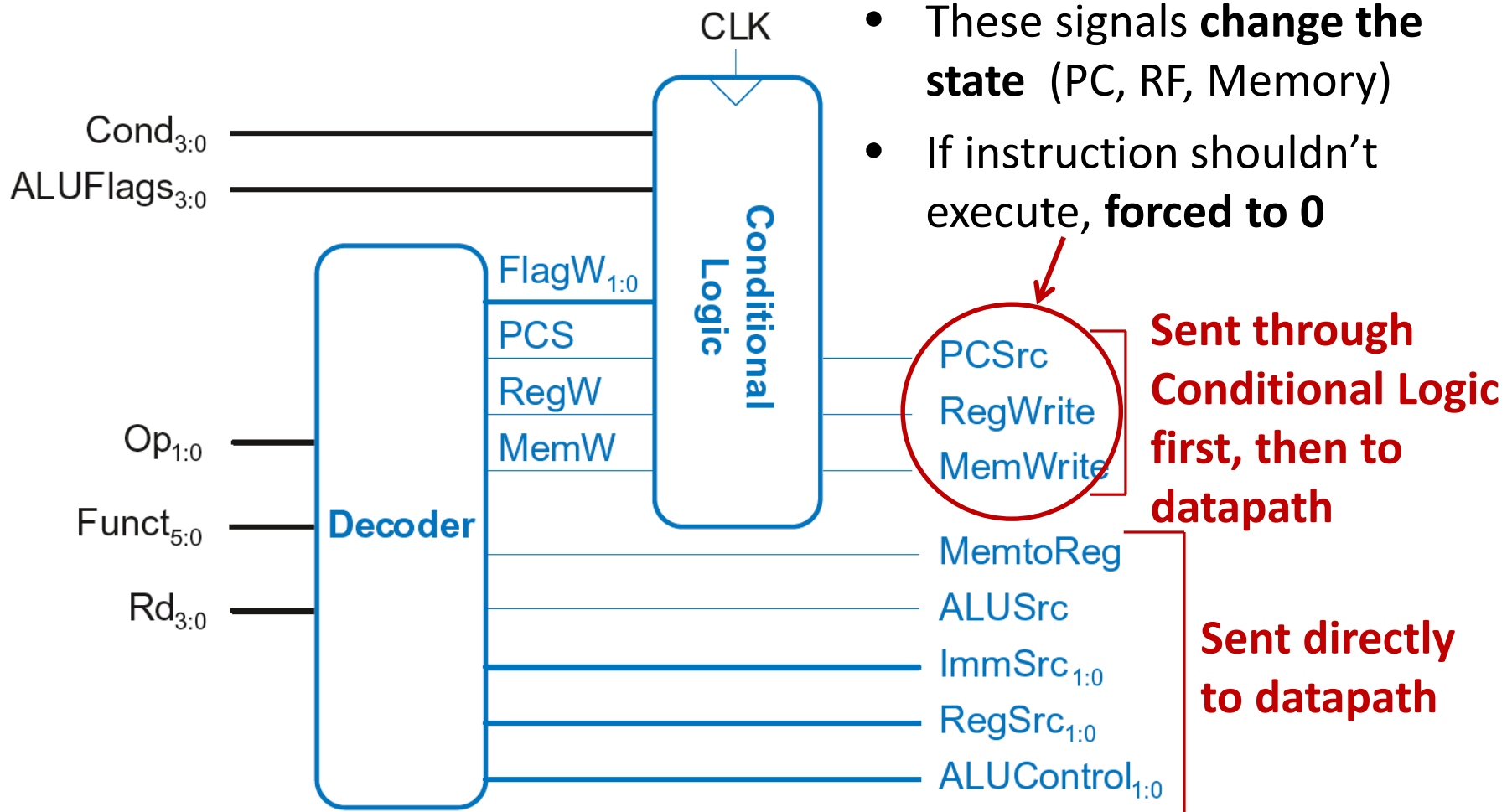


ImmSrc _{1:0}	ExtImm	Description
00	{24'b0, Instr _{7:0} }	Zero-extended <i>imm8</i>
01	{20'b0, Instr _{11:0} }	Zero-extended <i>imm12</i>
10	{6{Instr ₂₃ }, Instr _{23:0} }	Sign-extended <i>imm24</i>

Single-Cycle ARM processor

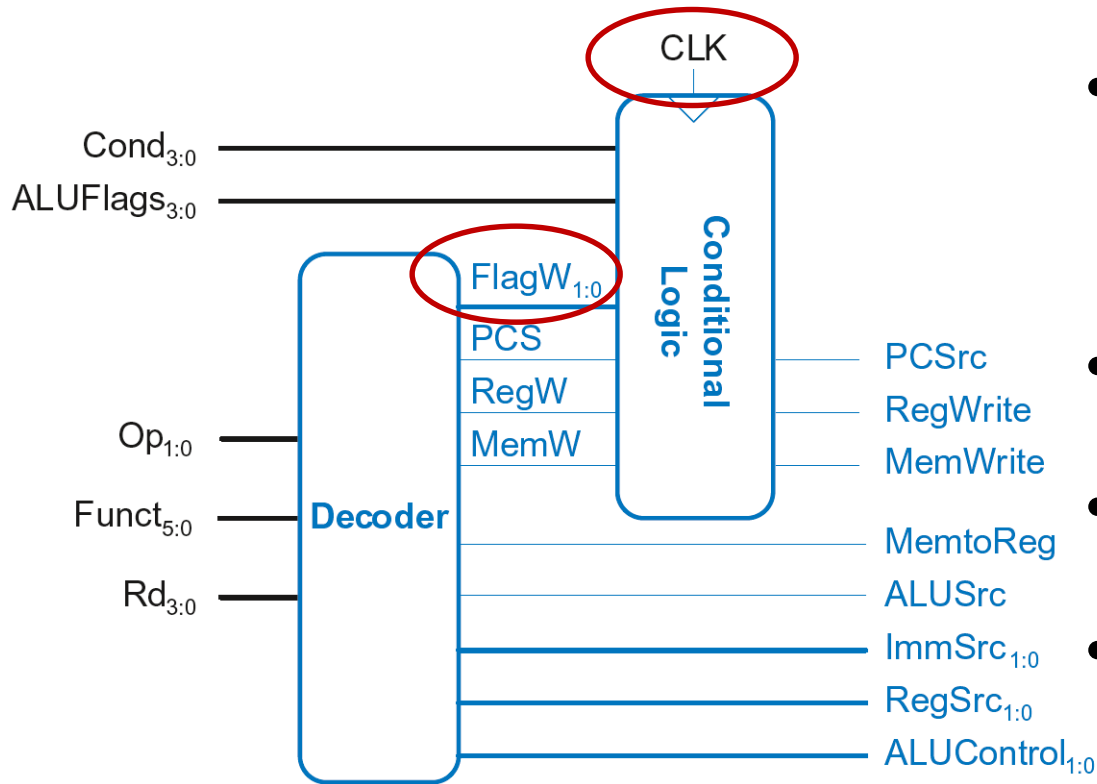


Single-Cycle Control



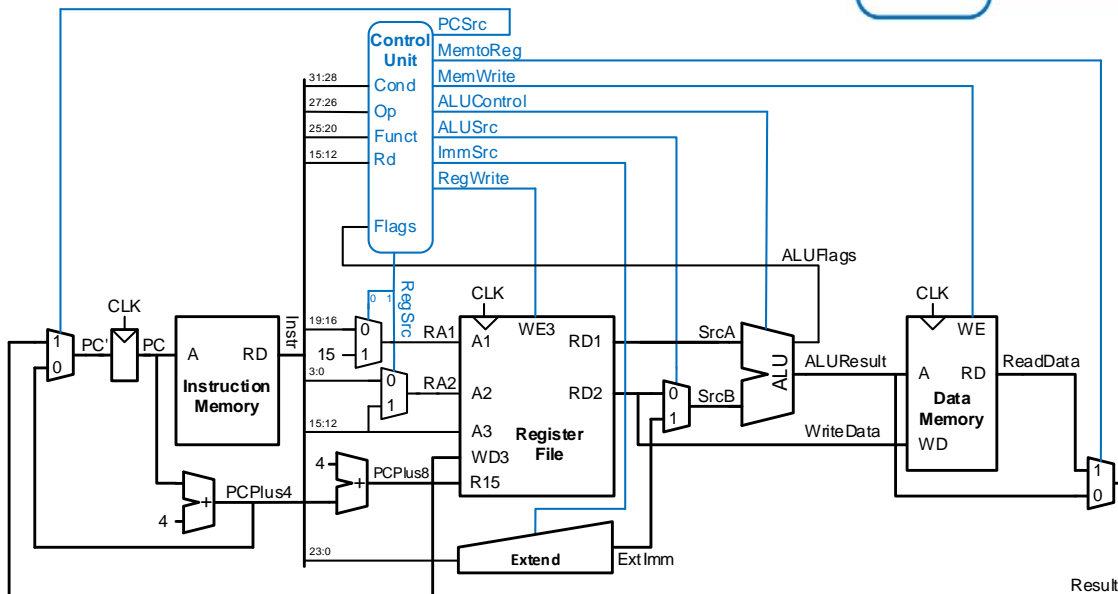
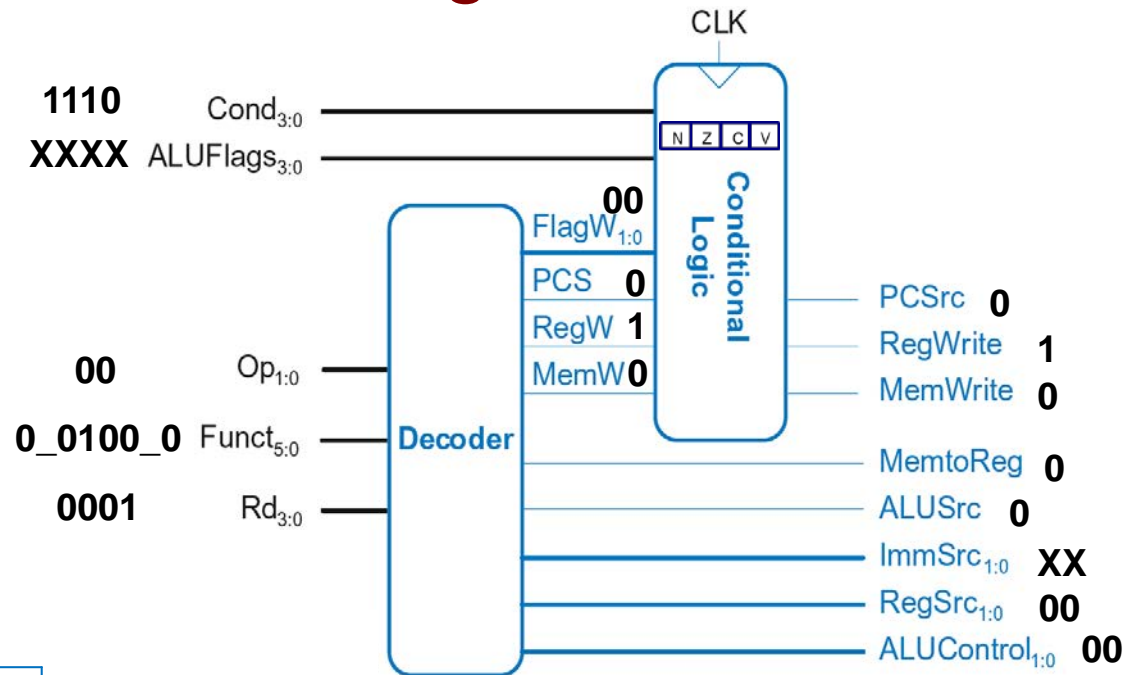
- These signals **change the state** (PC, RF, Memory)
- If instruction shouldn't execute, **forced to 0**

Single-Cycle control

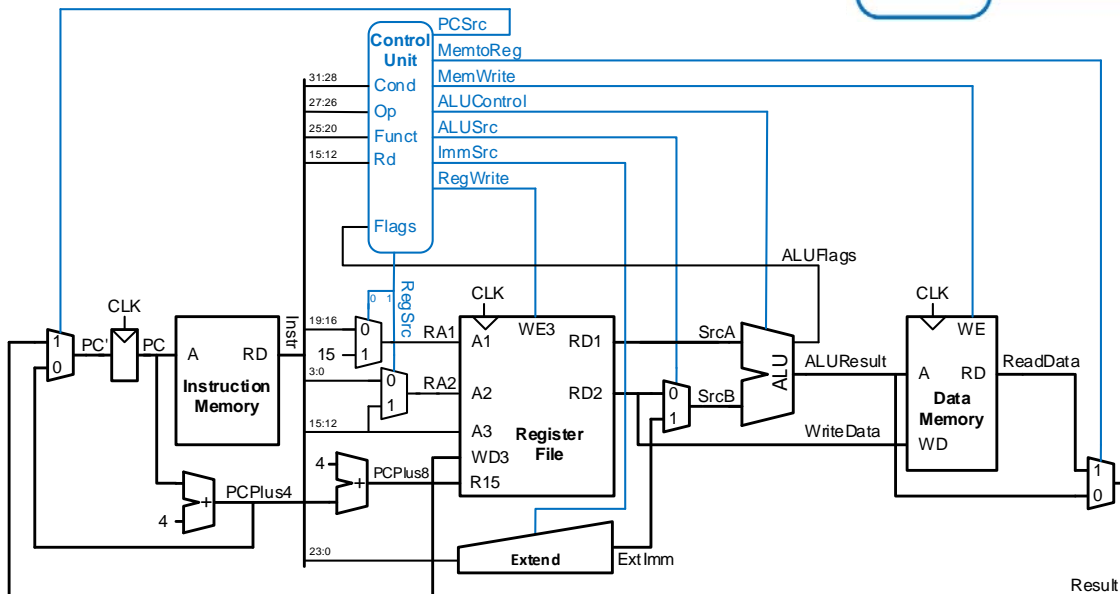
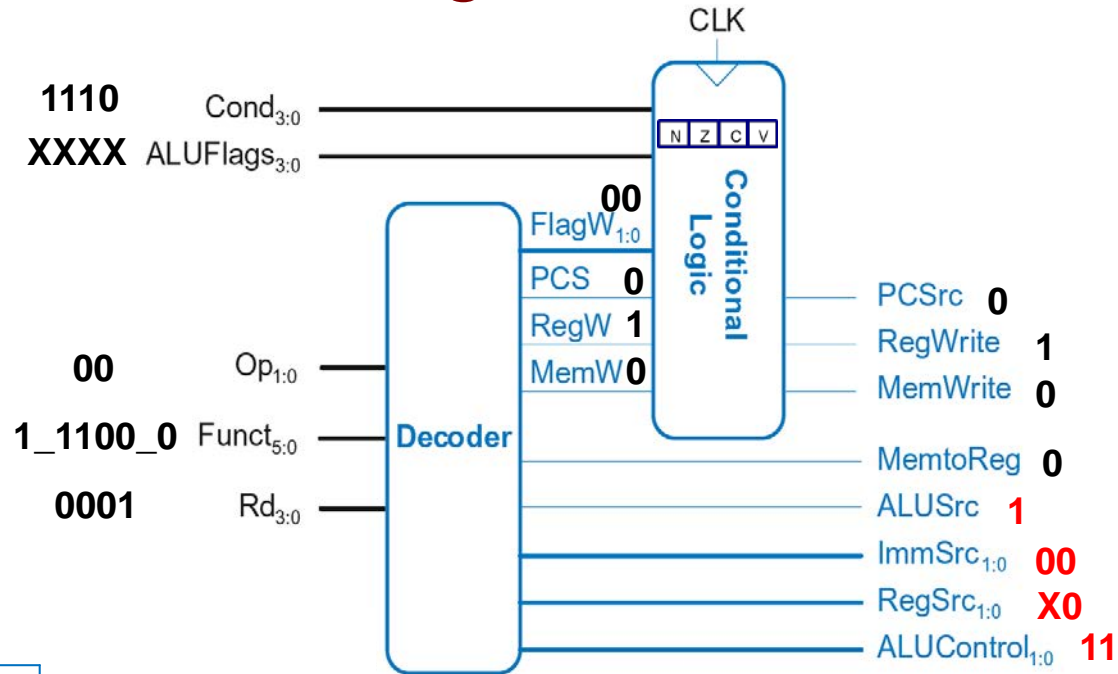


- **$FlagW_{1:0}$** : Flag Write signal, asserted when $ALUFlags$ should be saved (i.e., on instruction with $S=1$)
- ADD, SUB update all flags (**NZCV**)
- AND, ORR only update **NZ** flags
- So, two bits needed:
 - **$FlagW_1 = 1$** : NZ saved ($ALUFlags_{3:2}$ saved)
 - **$FlagW_0 = 1$** : CV saved ($ALUFlags_{1:0}$ saved)

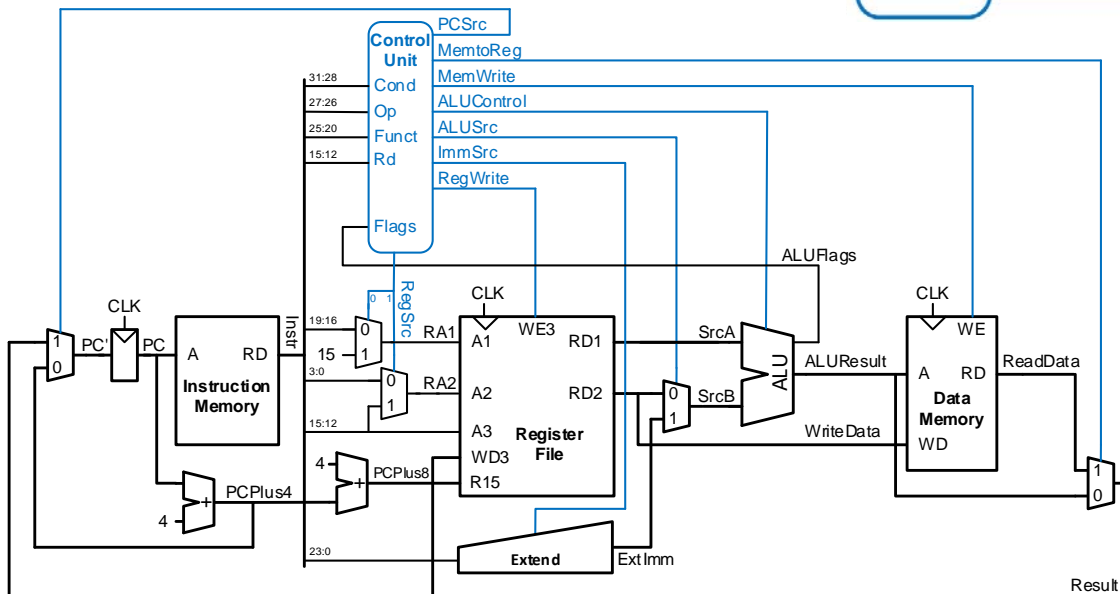
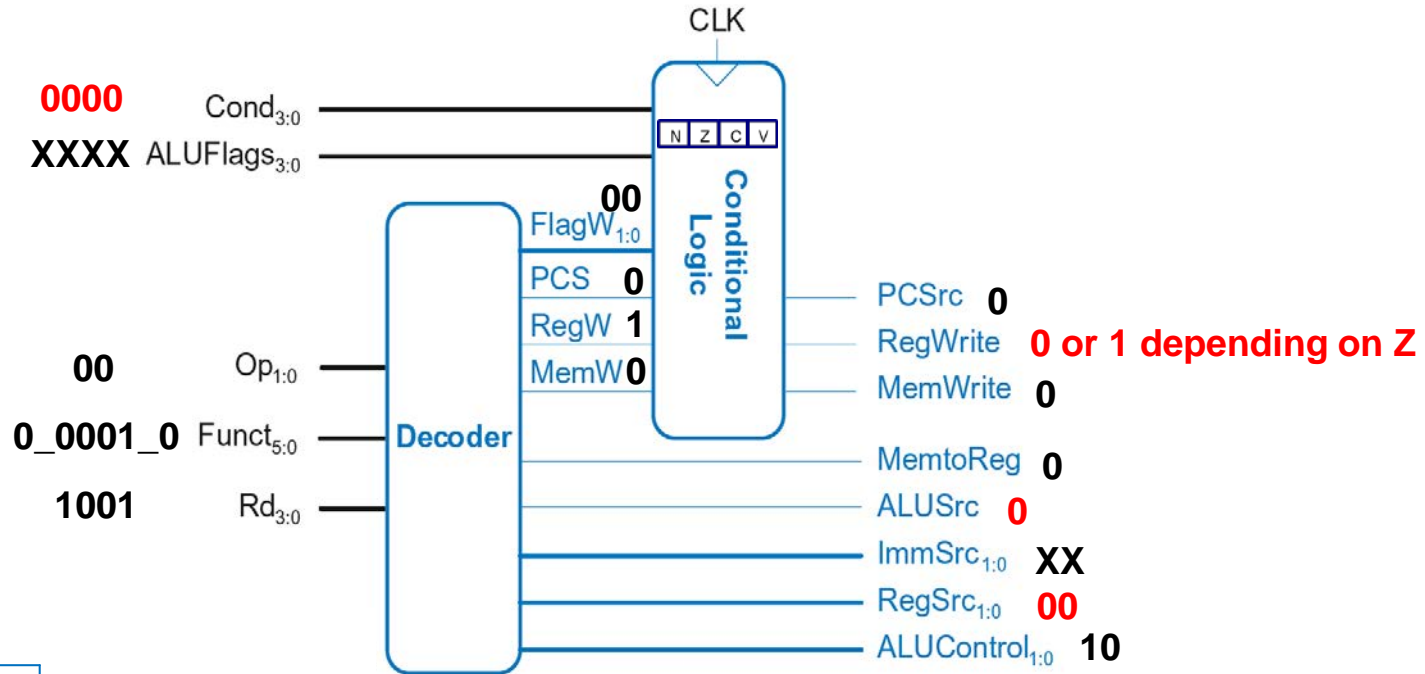
Example of control logic: ADD R1, R2, R3



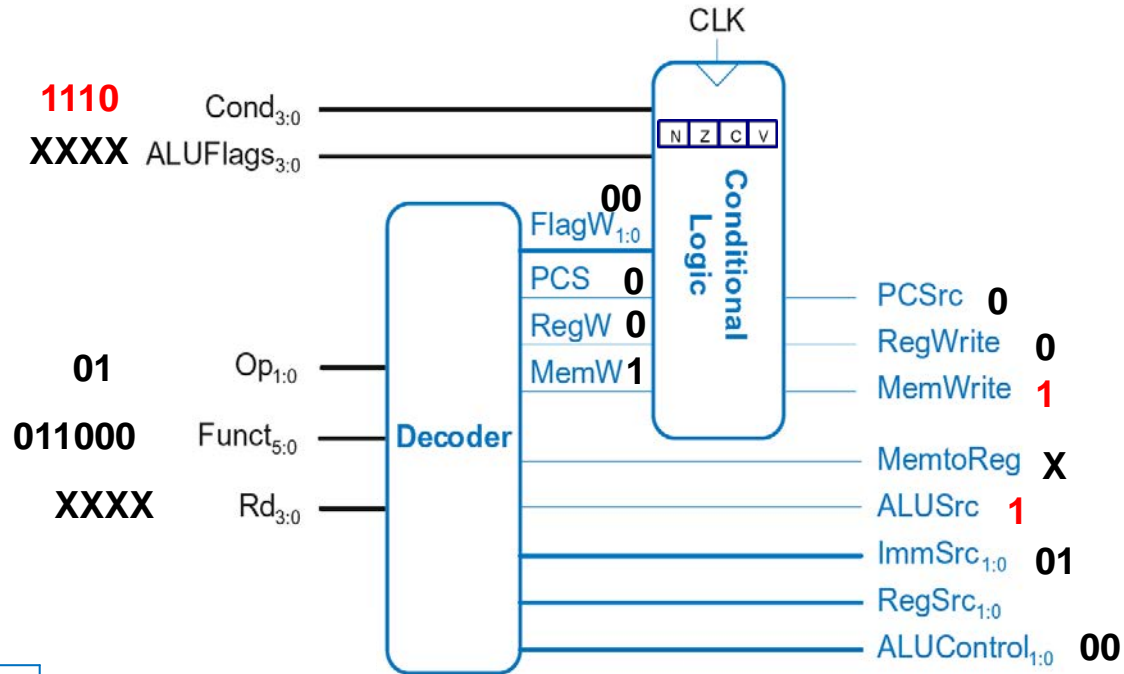
Example of control logic: ORR R1, R2, #F0



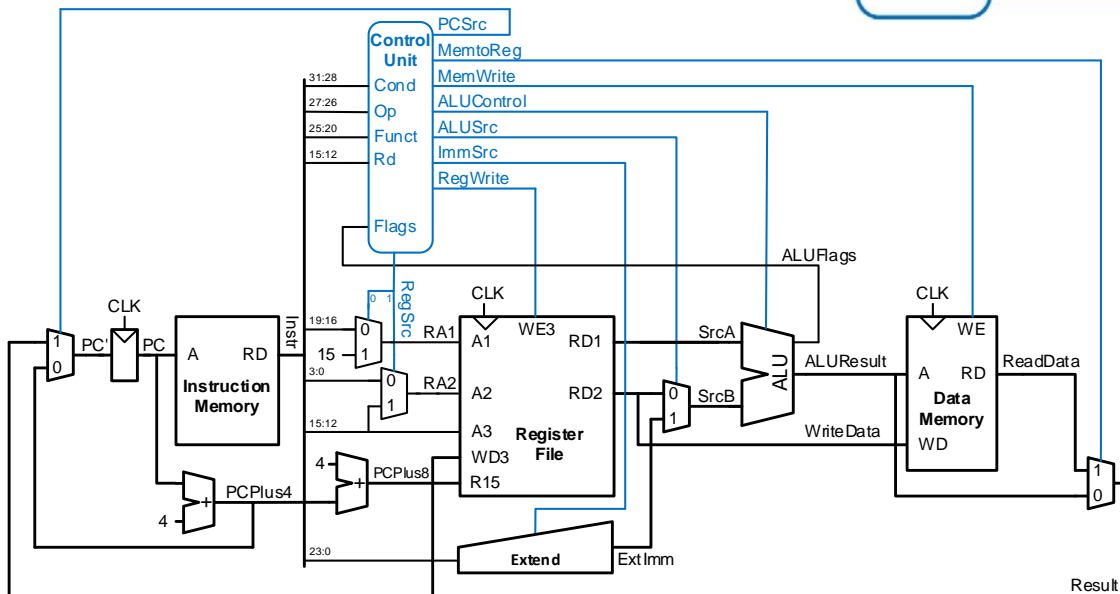
Example of control logic: ANDEQ R5, R6, R7



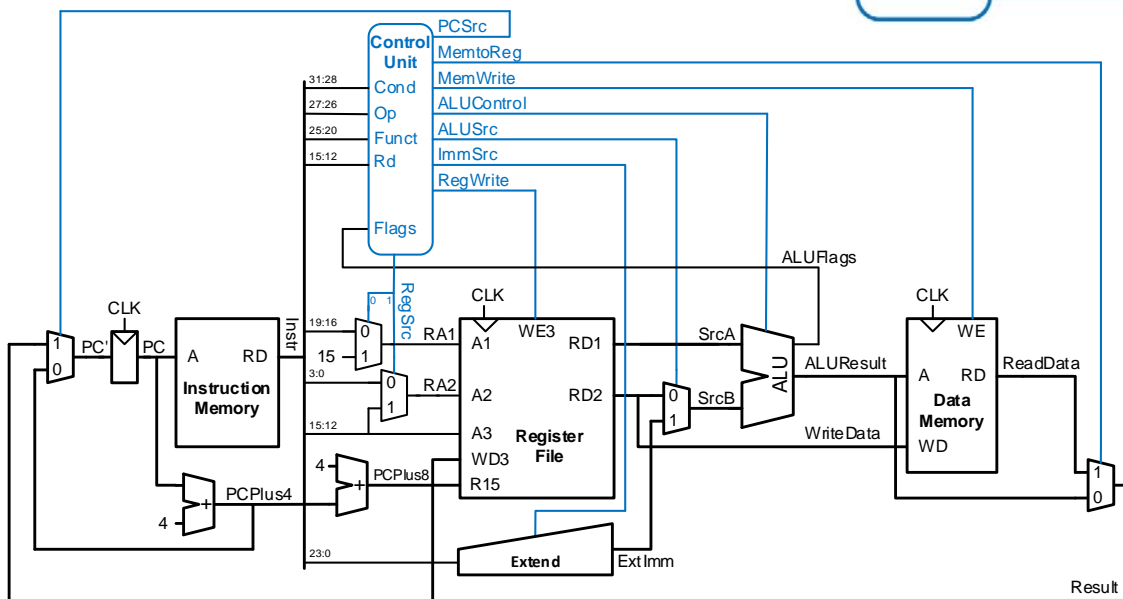
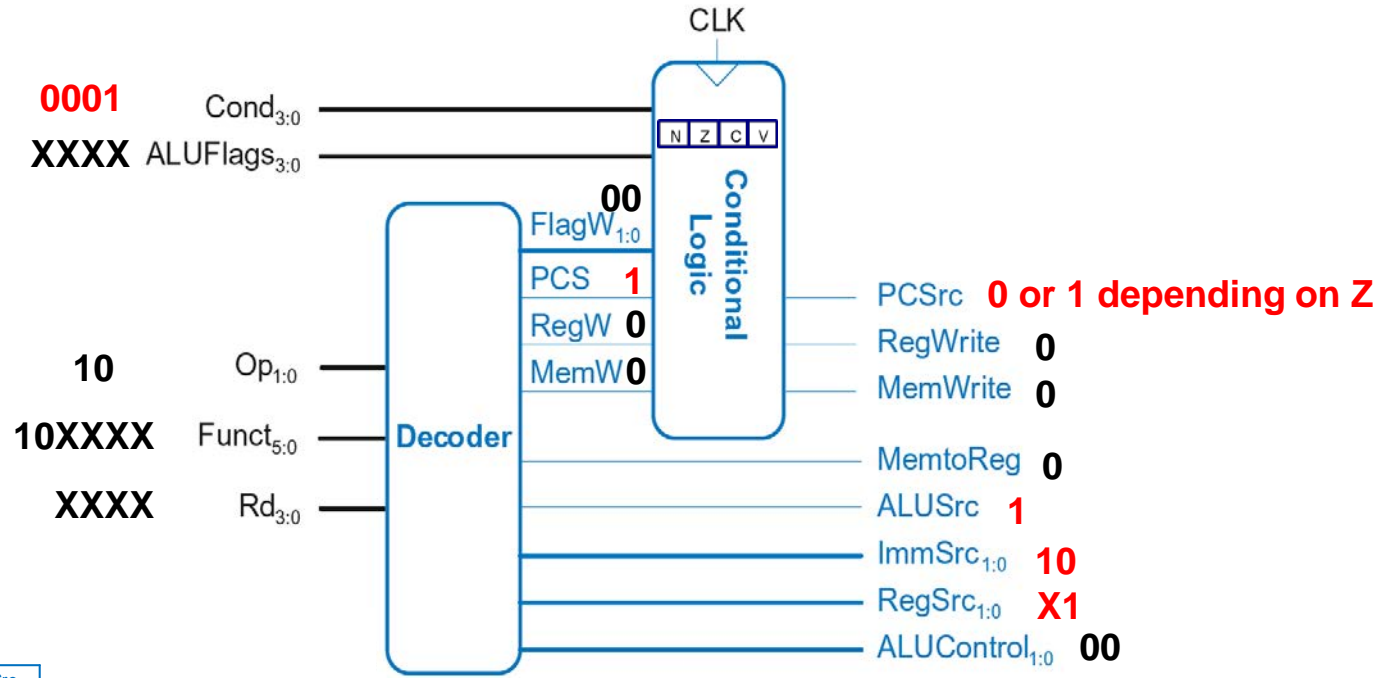
Example of control logic: STR R11, [R5, #8]



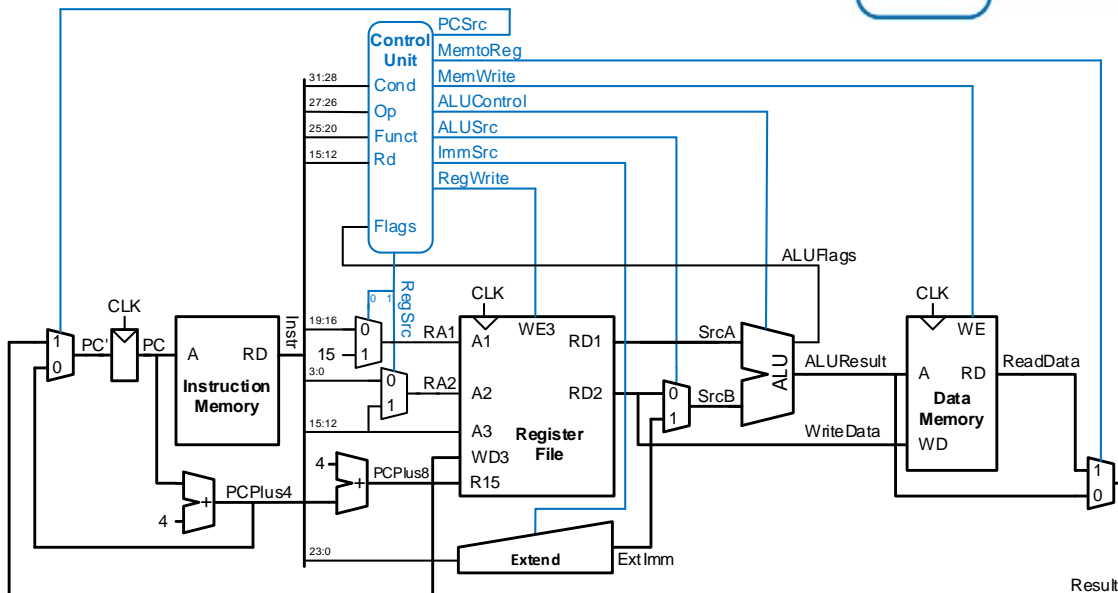
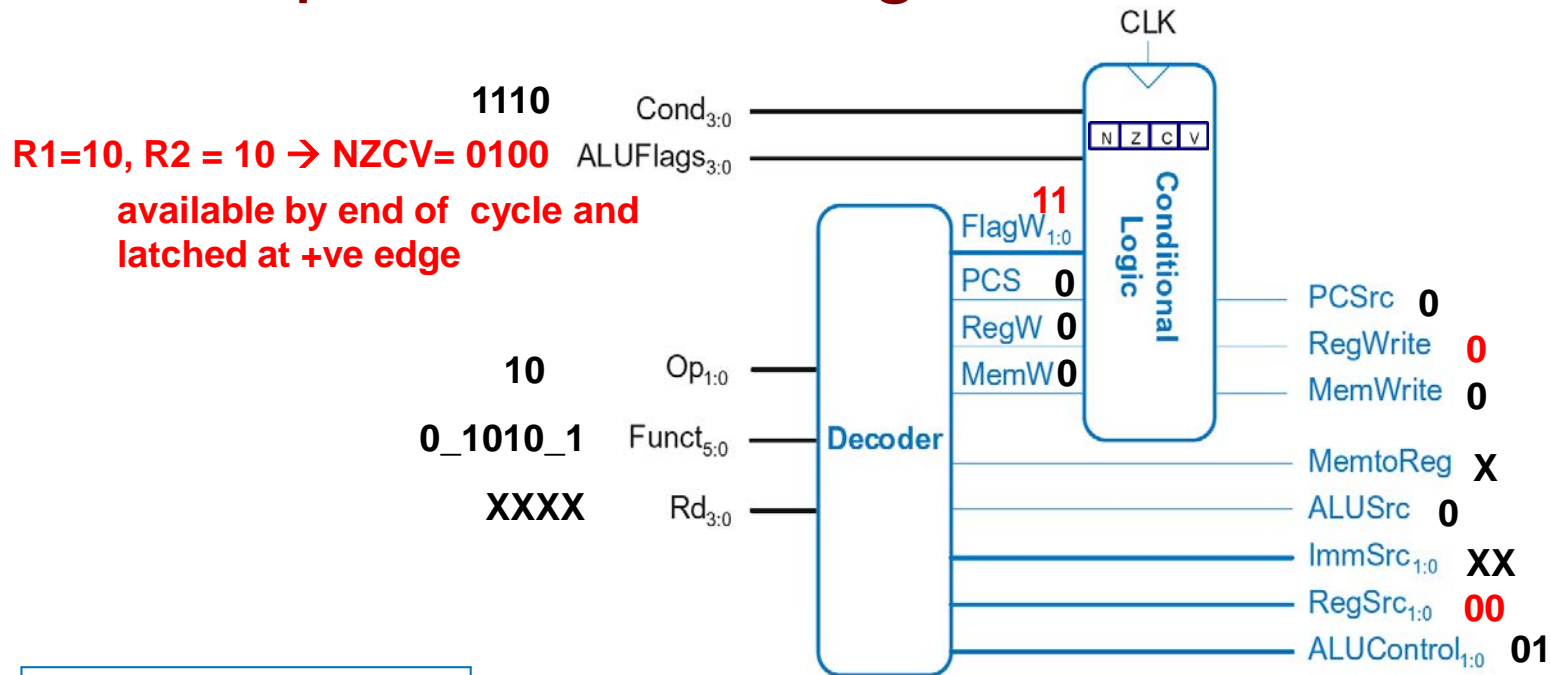
10



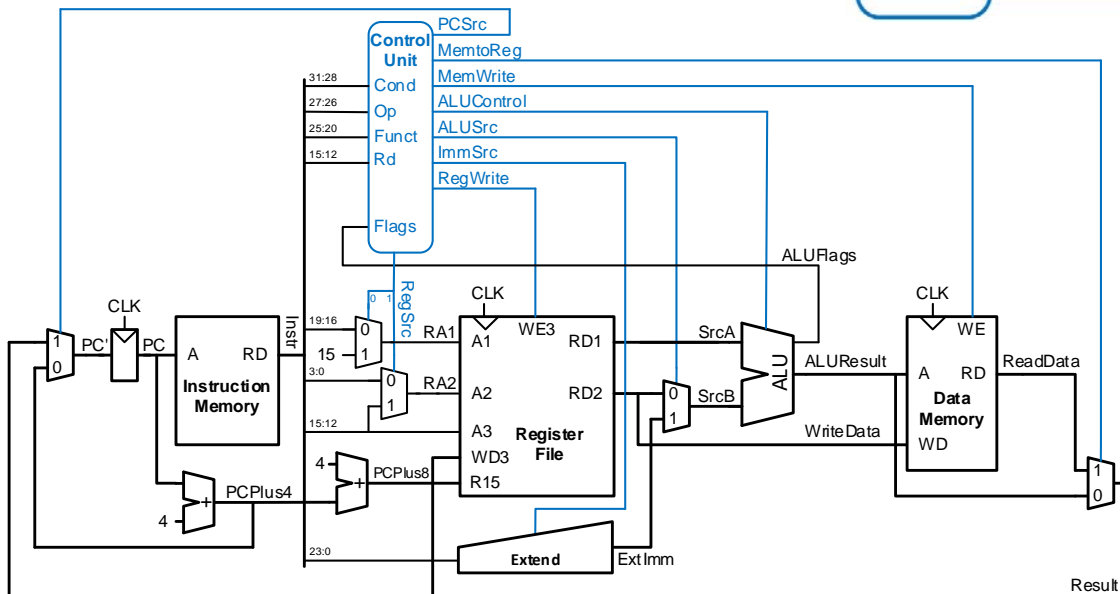
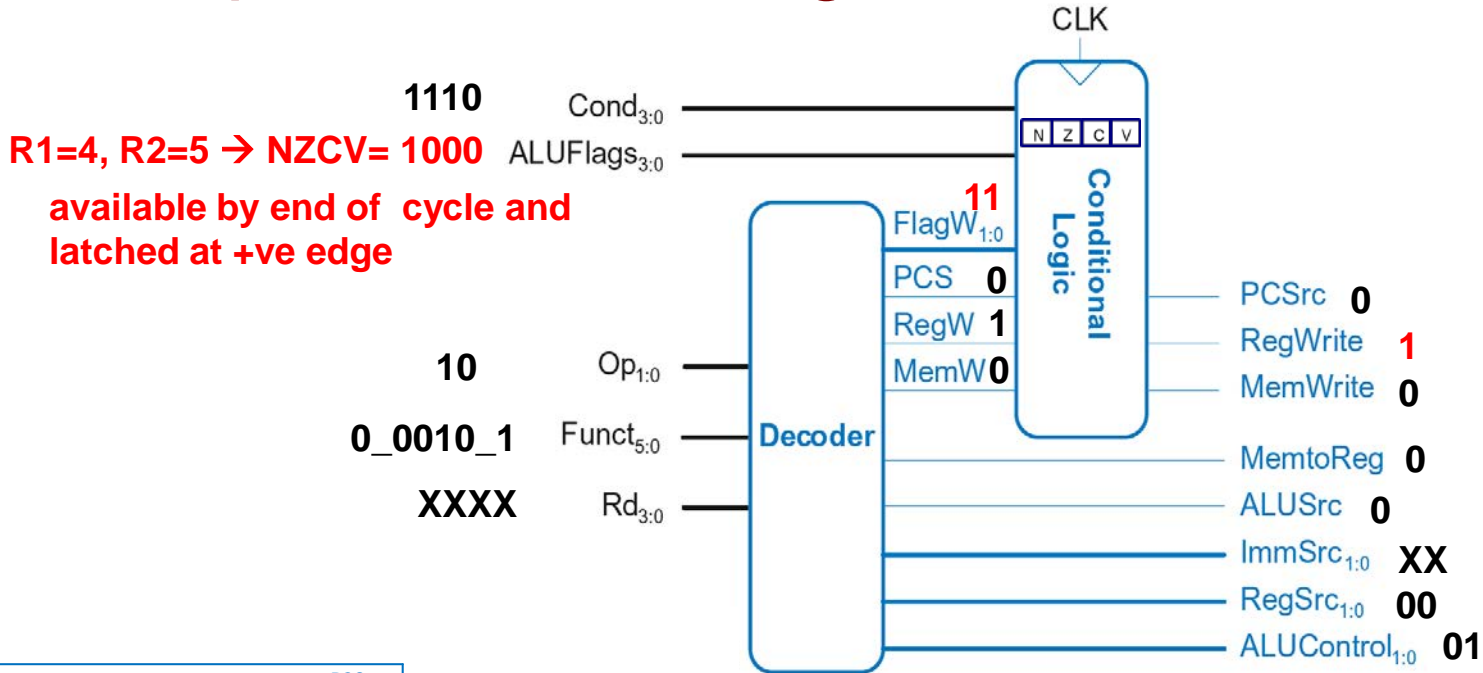
Example of control logic: BNE THERE



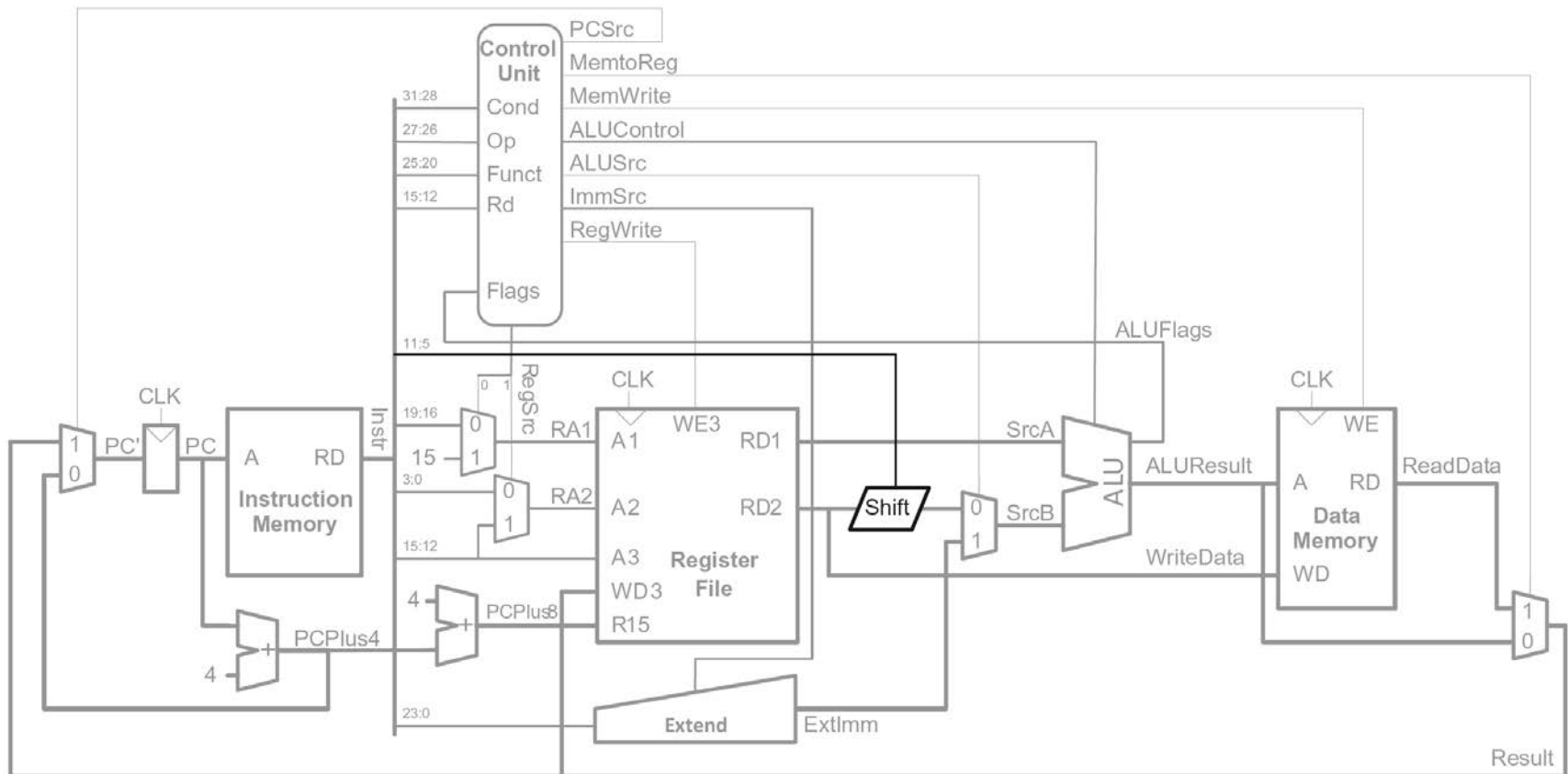
Example of control logic: CMP R1, R2



Example of control logic: SUBS R3, R1, R2



Extended functionality: shifted register (by constant)



Assembly Code

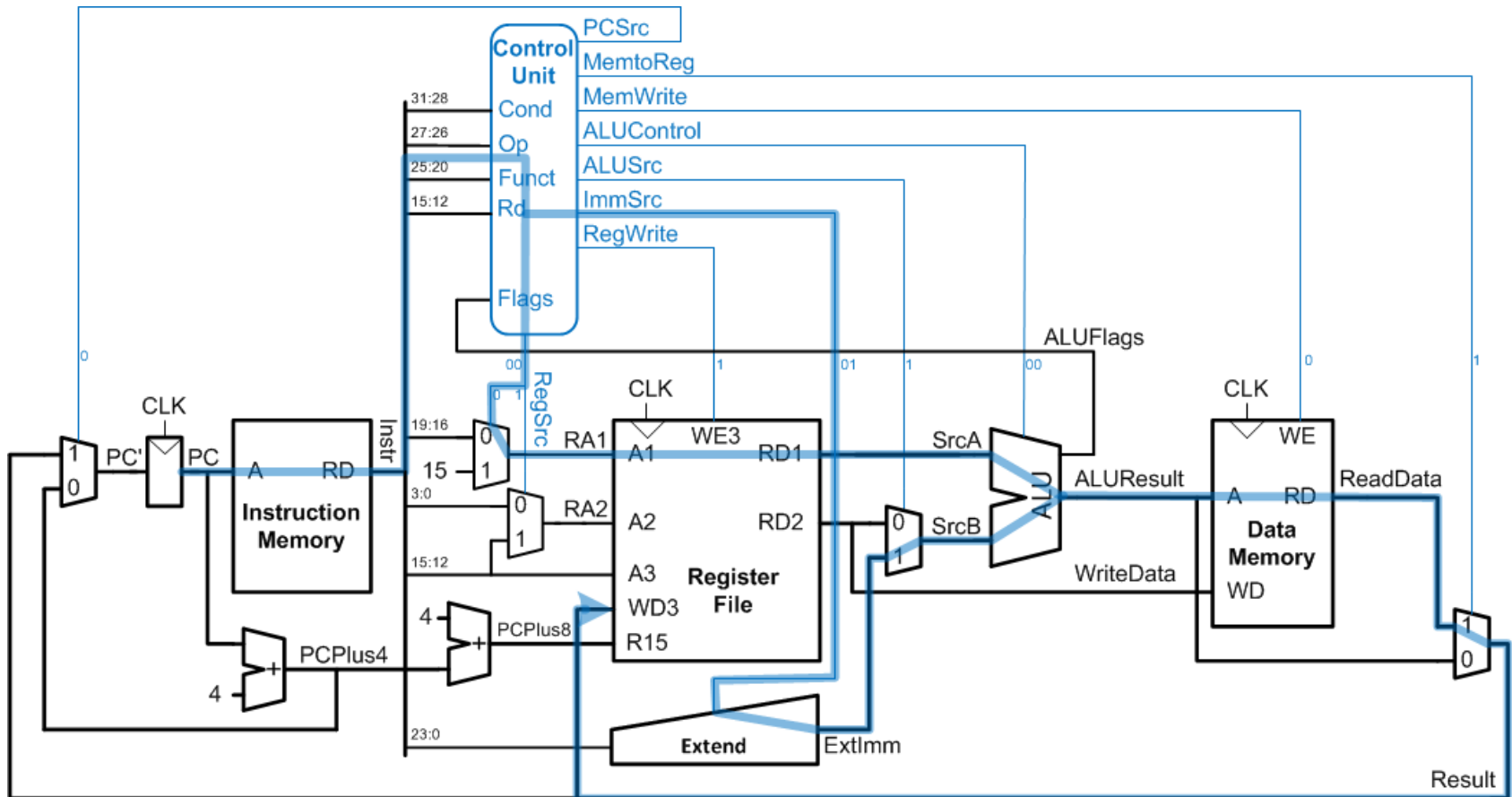
Field Values

	31:28	27:26	25	24:21	20	19:16	15:12	11:7	6:5	4	3:0
ADD R7, R2, R12, LSR #5	14	0	0	4	0	2	7	5	0	1 ₂ 0	12
	cond	op	l	cmd	S	rn	rd	shamt5	sh		rm

No change to controller

Critical path

- T_C is limited by the critical path (LDR)



- Single-cycle critical path:

$$T_{cl} = t_{pcq_PC} + t_{mem} + t_{dec} + \max[t_{mux} + t_{RFread}, t_{sext} + t_{mux}] + t_{ALU} + t_{mem} + t_{mux} + t_{RFsetup}$$

Critical path delay

- Single-cycle critical path:

$$T_{cl} = t_{pcq_PC} + t_{mem} + t_{dec} + \max[t_{mux} + t_{RFread}, t_{sext} + t_{mux}] + t_{ALU} + t_{mem} + t_{mux} + t_{RFsetup}$$

- In most cases:

$$- T_{cl} = t_{pcq_PC} + 2t_{mem} + t_{dec} + t_{RFread} + t_{ALU} + 2t_{mux} + t_{RFsetup}$$

Element	Parameter	Delay (ps)
Register clock-to-Q	t_{pcq_PC}	40
Register setup	t_{setup}	50
Multiplexer	t_{mux}	25
ALU	t_{ALU}	120
Decoder	t_{dec}	70
Memory read	t_{mem}	200
Register file read	t_{RFread}	100
Register file setup	$t_{RFsetup}$	60

$$\begin{aligned} T_{cl} &= t_{pcq_PC} + 2t_{mem} + t_{dec} + t_{RFread} + t_{ALU} + 2t_{mux} + t_{RFsetup} \\ &= [50 + 2(200) + 70 + 100 + 120 + 2(25) + 60] \text{ ps} \\ &= 840 \text{ ps} \end{aligned}$$

Summary

- Single-cycle processor design is simple
- Plenty of room for improvement:
 - Multi-cycle
 - Pipelining
 - Superscalar
- In Lab05 you will design, implement and boot your first processor!