Chapter 7 :: Microarchitecture

Digital Design and Computer Architecture

David Money Harris and Sarah L. Harris

Lectured by Jeong-Gun Lee

School of Software, Hallym University

CPU 만들기

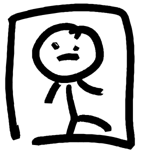
나만의

MIPS processor → "Verilog HDL" → 회로

다운로드

*HDL ?
Hardware Description Language

이력서

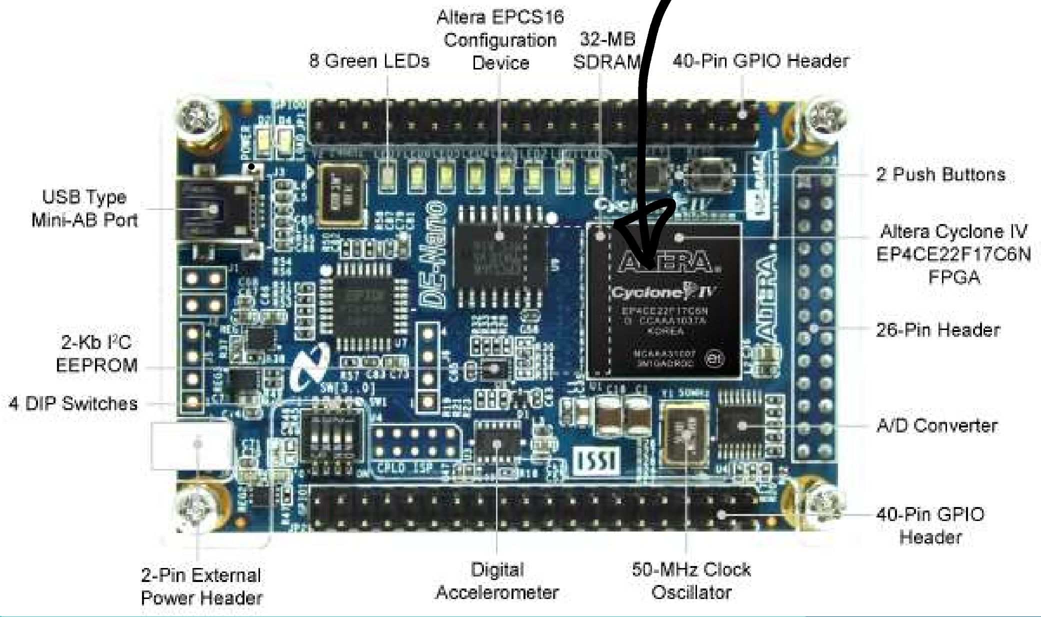


홍길동

Hallym ...

* MIPS 프로세서 설계 및 구현
- FPGA 이용

↖ 완전 뽀대나죠!



Chapter 7 :: Topics

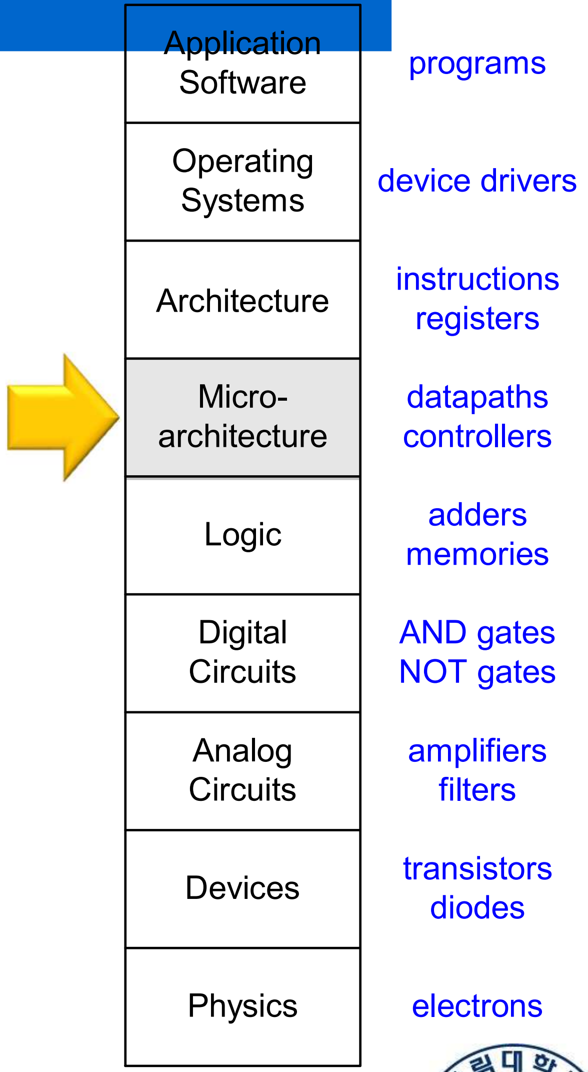
- **Introduction**
- **Performance Analysis (성능 분석)**
- **Single-Cycle Processor (단 사이클 프로세서)**
- Multicycle Processor
- Pipelined Processor
- Exceptions
- Advanced Microarchitecture

Introduction

- Microarchitecture: how to implement an architecture in hardware (마이크로아키텍쳐: 아키텍쳐를 어떻게 하드웨어로 구현할 것인가!!!)

- Processor:
  - Datapath: functional blocks
  - (데이터패스: 기능 블럭)
  - Control: control signals
  - (제어: 제어신호)

- 아키텍쳐 ? → 명령어 집합



Microarchitecture

- Multiple implementations for a single architecture:
- (하나의 아키텍쳐에 여러 개의 구현 방법!)

  - Single-cycle (단일 싸이클)
    - Each instruction executes in a **single cycle**
  - Multicycle (다수 싸이클)
    - Each instruction is broken up into a series of shorter steps
  - Pipelined (파이프라인)
    - Each instruction is broken up into a series of steps
    - Multiple instructions execute at once.

Processor Performance

- Program execution time (프로그램 수행시간)

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

- Definitions(정의):
  - Cycles/instruction = CPI // 명령어당 요구되는 클럭 싸이클
  - Seconds/cycle = clock period
  - 1/CPI = Instructions/cycle = IPC // 한클럭당 수행되는 명령어수

- Challenge is to satisfy constraints of:
- (도전은 다음 제약사항들을 만족시키는 것)
  - Cost (비용)
  - Power (파워소모)
  - Performance (성능)

Clock Cycle ?

MIPS Processor

- We consider a subset of MIPS instructions:
  - **R-type instructions:** `and, or, add, sub, slt`
  - **Memory instructions:** `lw, sw`
  - **Branch instructions:** `beq`

+ addi, bne, j, jr, jal

* Architecture < 아키텍처)

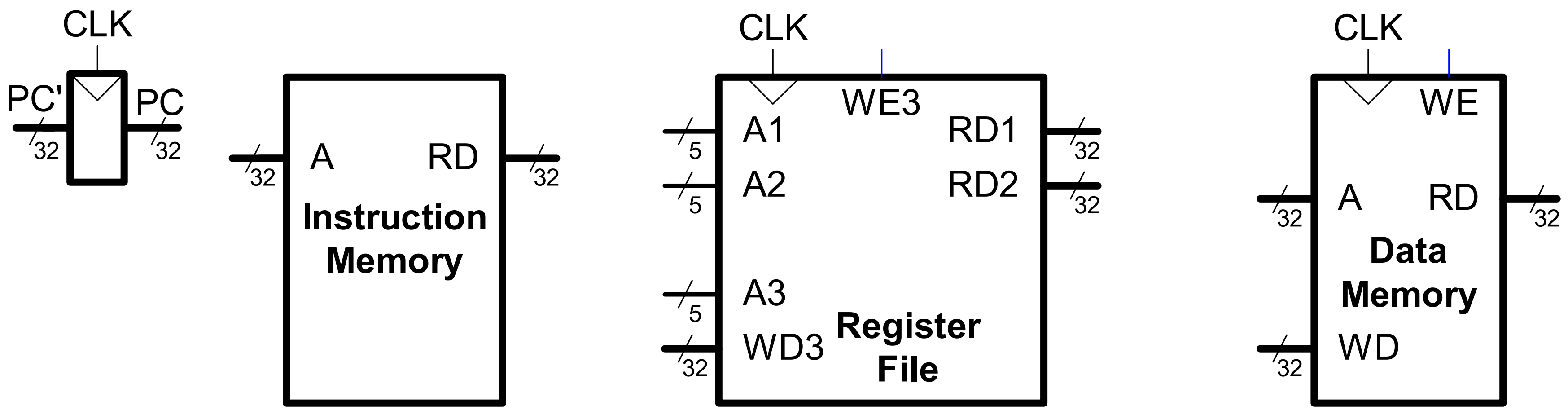
⇒ 명령어 집합

→ { and, or, add, sub, slt, lw, sw, beq ... }

Architectural State (아키텍쳐 상태)

- Determines everything about a processor:
  프로세서에 대한 모든 것을 결정하자!
  - PC
  - 32 registers
  - Memory

MIPS State Elements (상태 요소들)



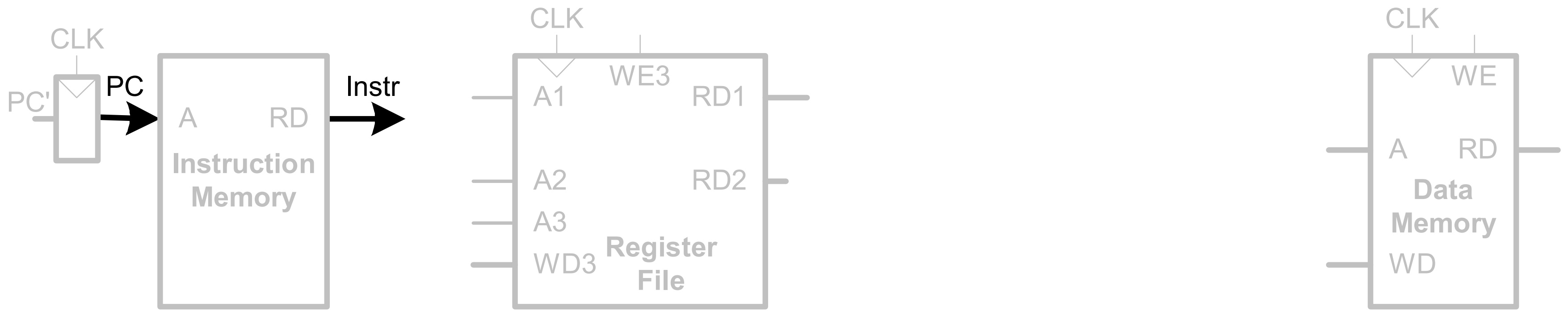
상태 요소들 == 메모리 요소들

Single-Cycle MIPS Processor

- Datapath (데이터가 계산되는 흐름)
- Control (제어신호를 생성하는 흐름)

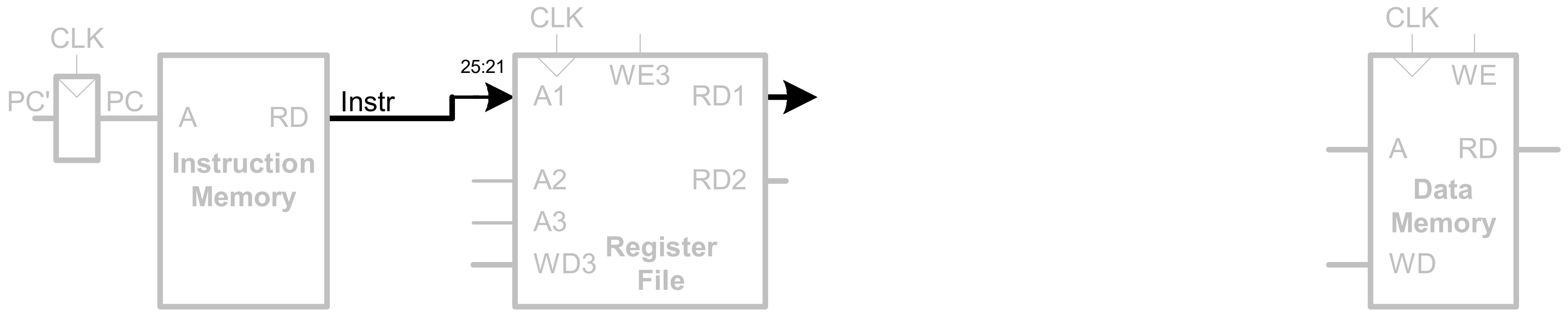
Single-Cycle Datapath

- First consider executing **lw**
  우선, **lw** 명령어의 실행을 고려해 봅시다!
- **STEP 1:** Fetch instruction (명령어 가져오기)



Single-Cycle Datapath

- **STEP 2:** Read source operands from register file
  레지스터 파일로 부터 소스 오퍼런드 읽어오기!

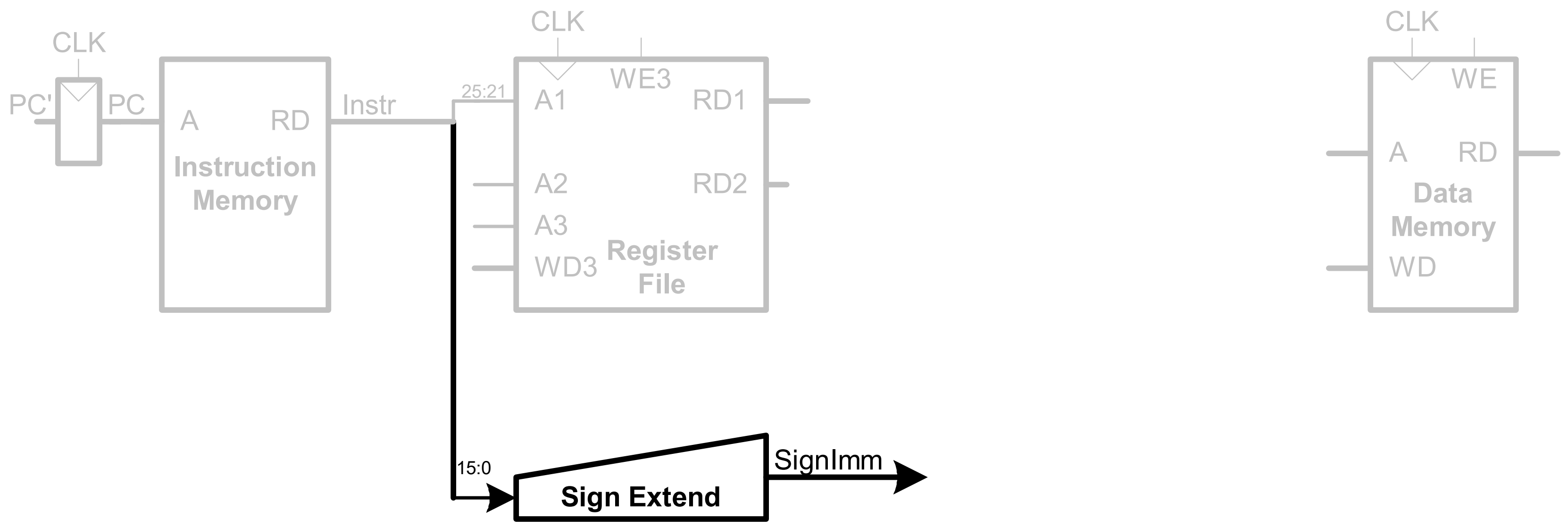


I-Type

op	rs	rt	imm
6 bits	5 bits	5 bits	16 bits

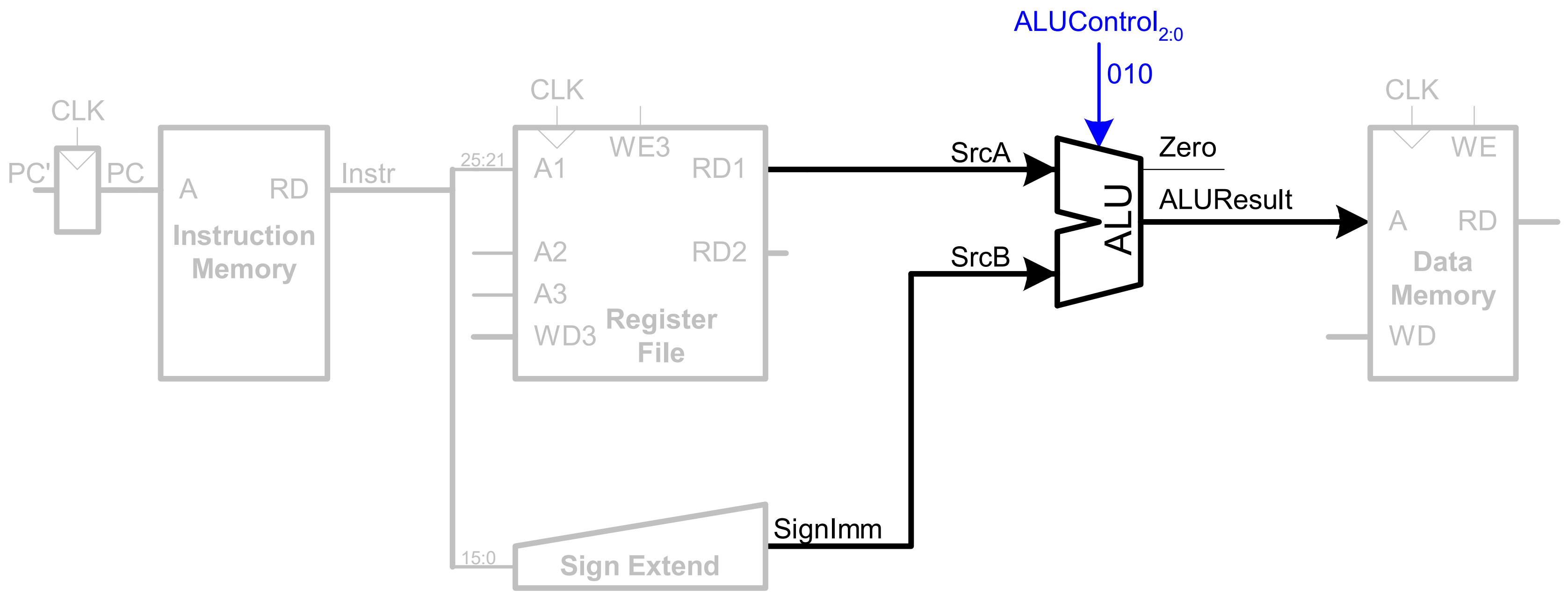
Single-Cycle Datapath

- **STEP 3:** Sign-extend the immediate



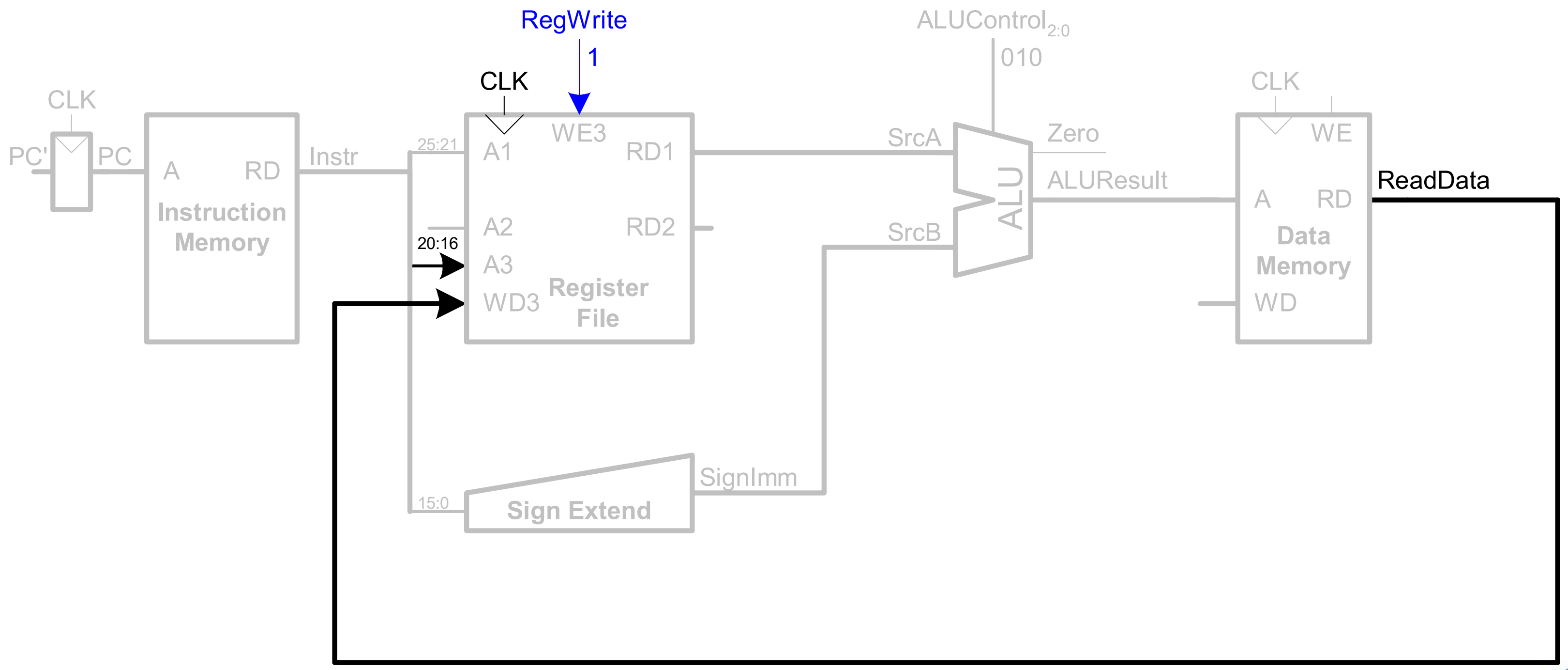
Single-Cycle Datapath

- **STEP 4:** Compute the memory address
  메모리 주소 계산하기



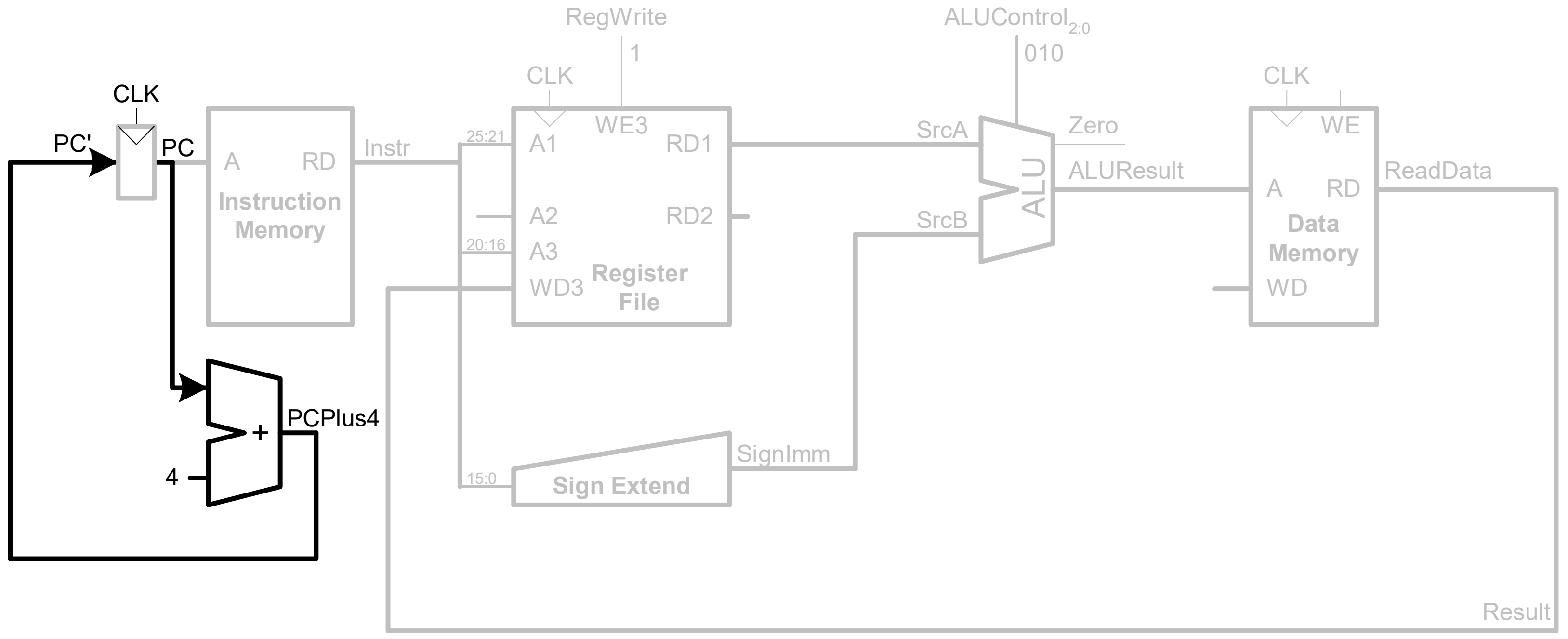
Single-Cycle Datapath

- **STEP 5:** Read data from memory and write it back to register file
  메모리로부터 데이터 읽고 그 데이터를 레지스터 파일에 쓰기!



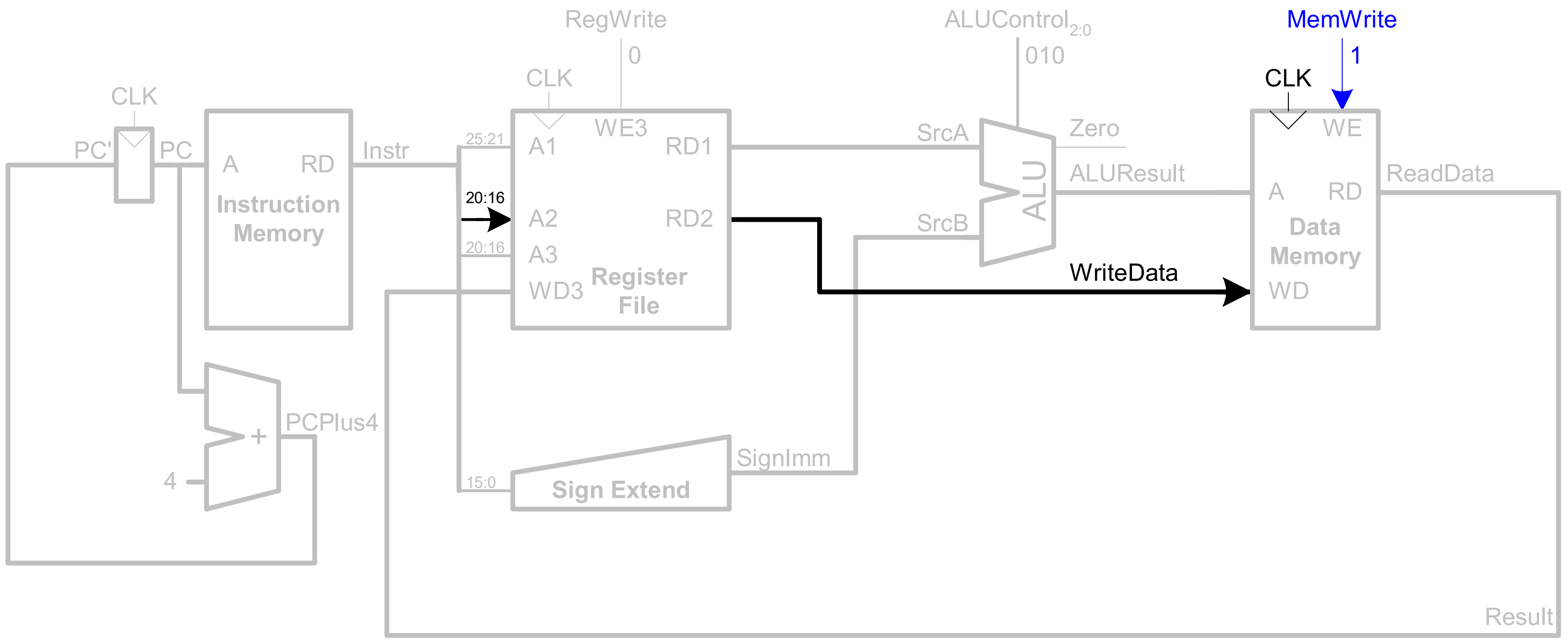
Single-Cycle Datapath

- **STEP 6:** Determine the address of the next instruction
  다음 명령어의 주소를 결정하기



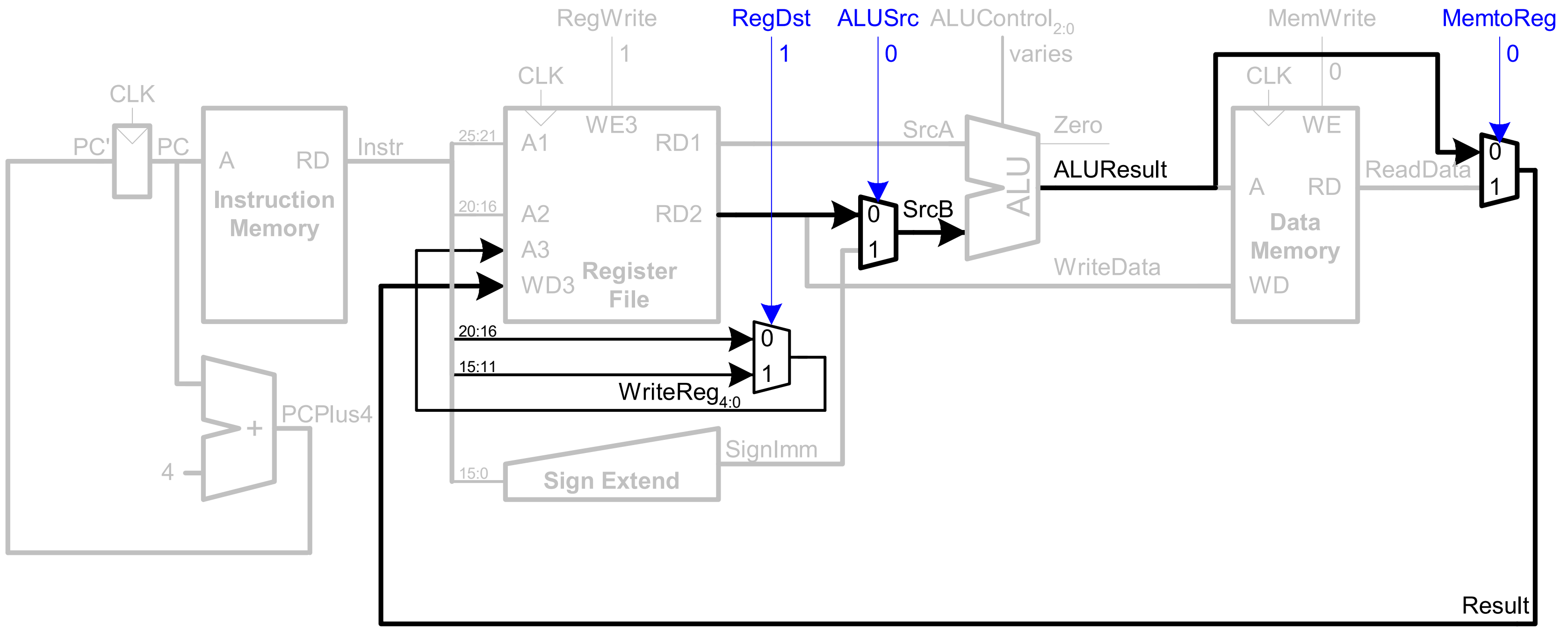
Single-Cycle Datapath

- Consider **sw**
- Need to write data to memory



Single-Cycle Datapath

- Consider **R-type** instructions: **`add, sub, and, or`**
- Write *ALUResult* to register file
- Writing to `rd` field of instruction (instead of `rt`)

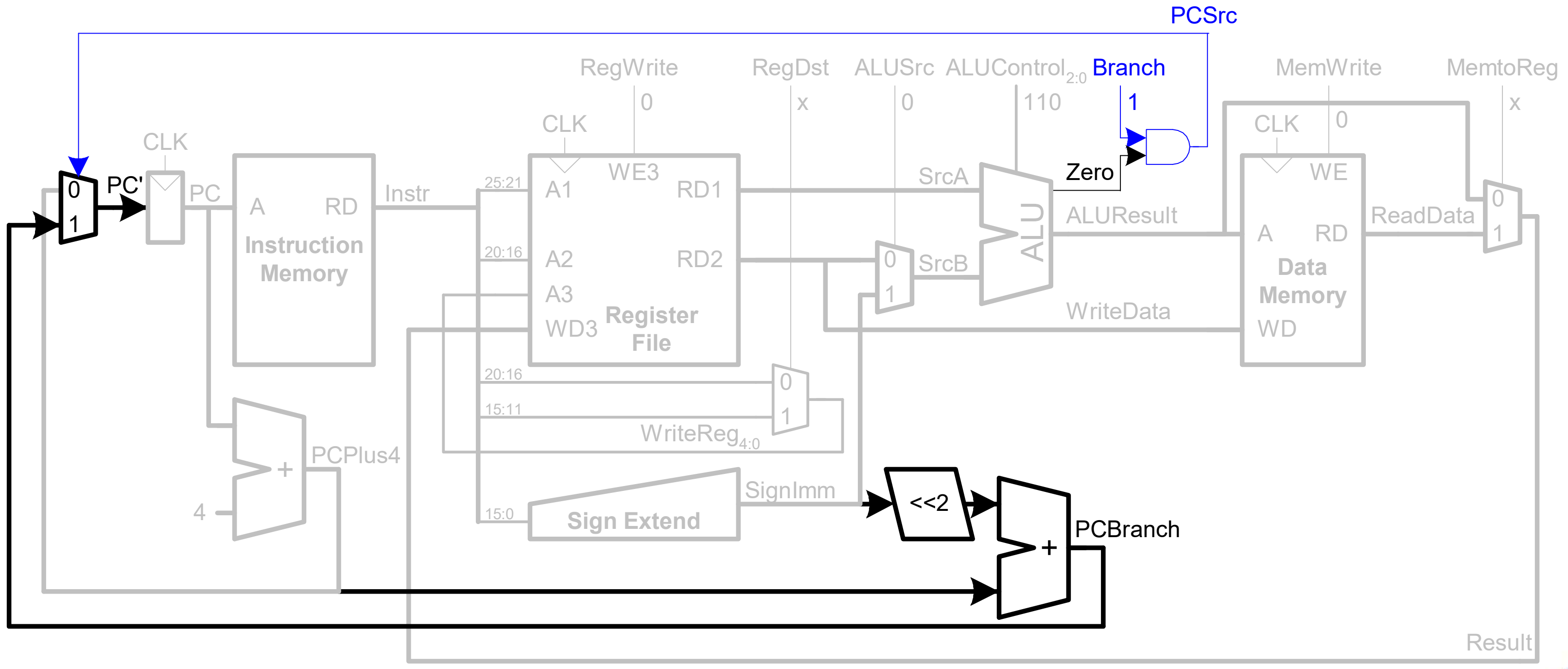


R-Type

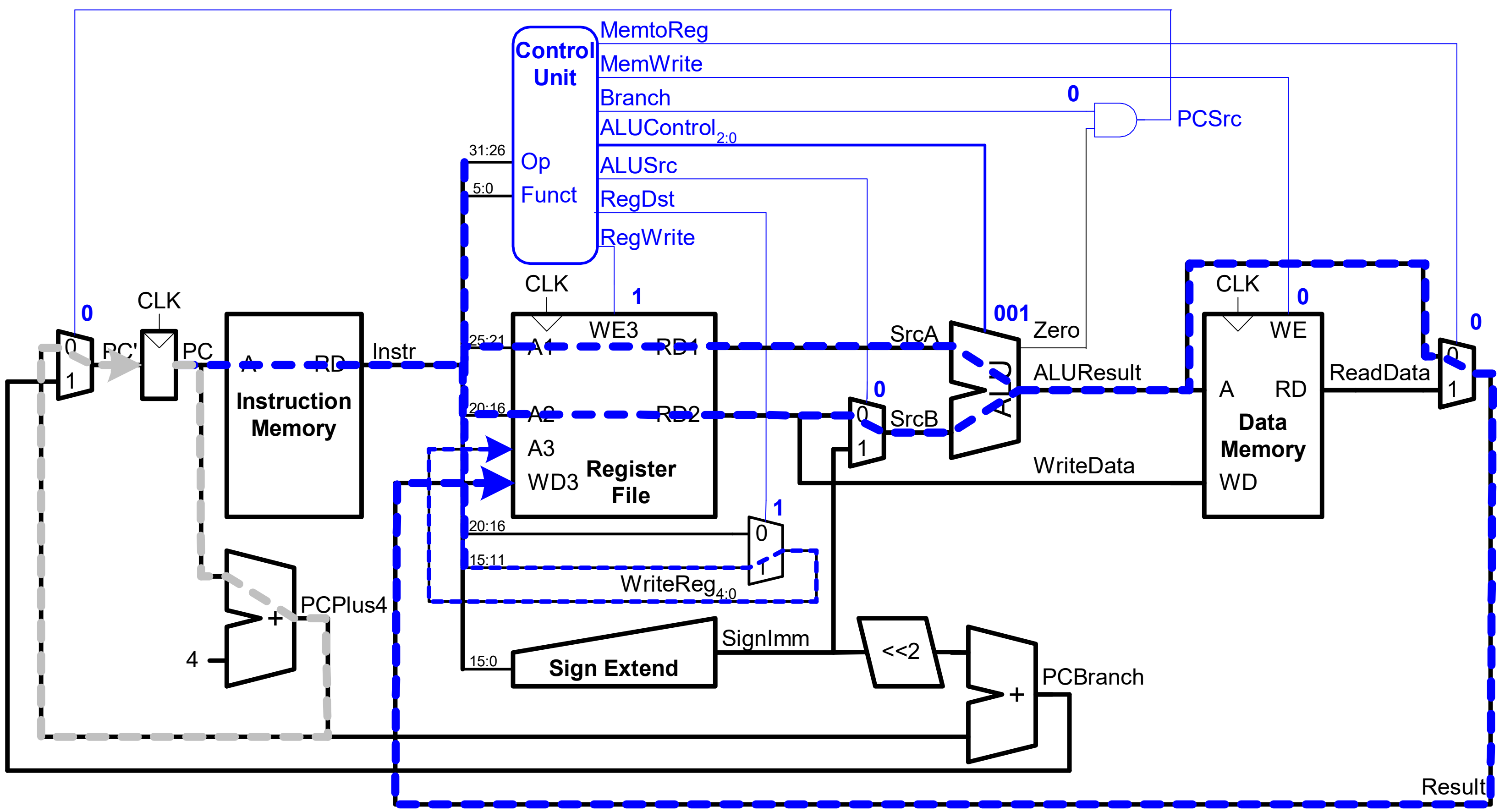
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Single-Cycle Datapath

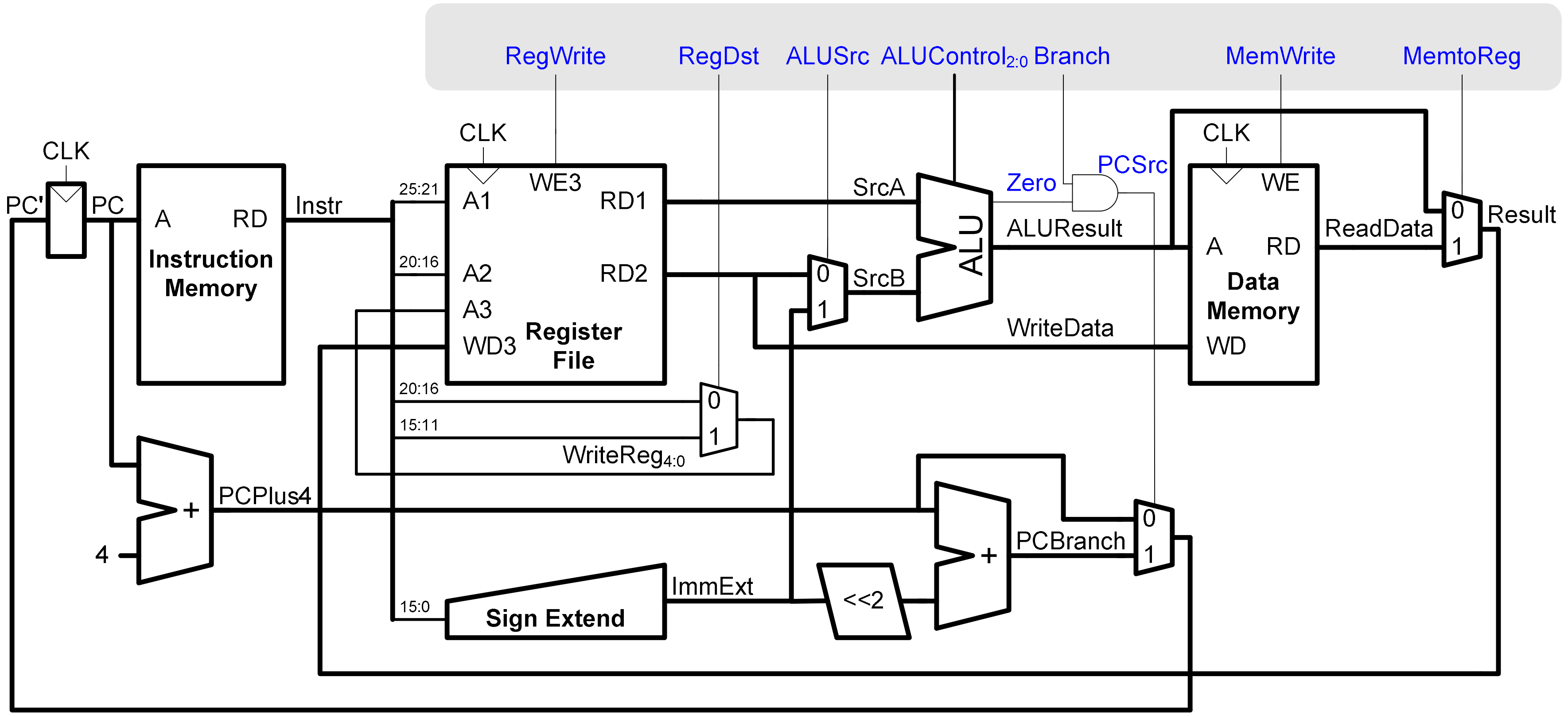
- Consider branch instruction: **beq**
- Determine whether register values are equal
- Calculate branch target address (BTA) from sign-extended immediate and PC+4



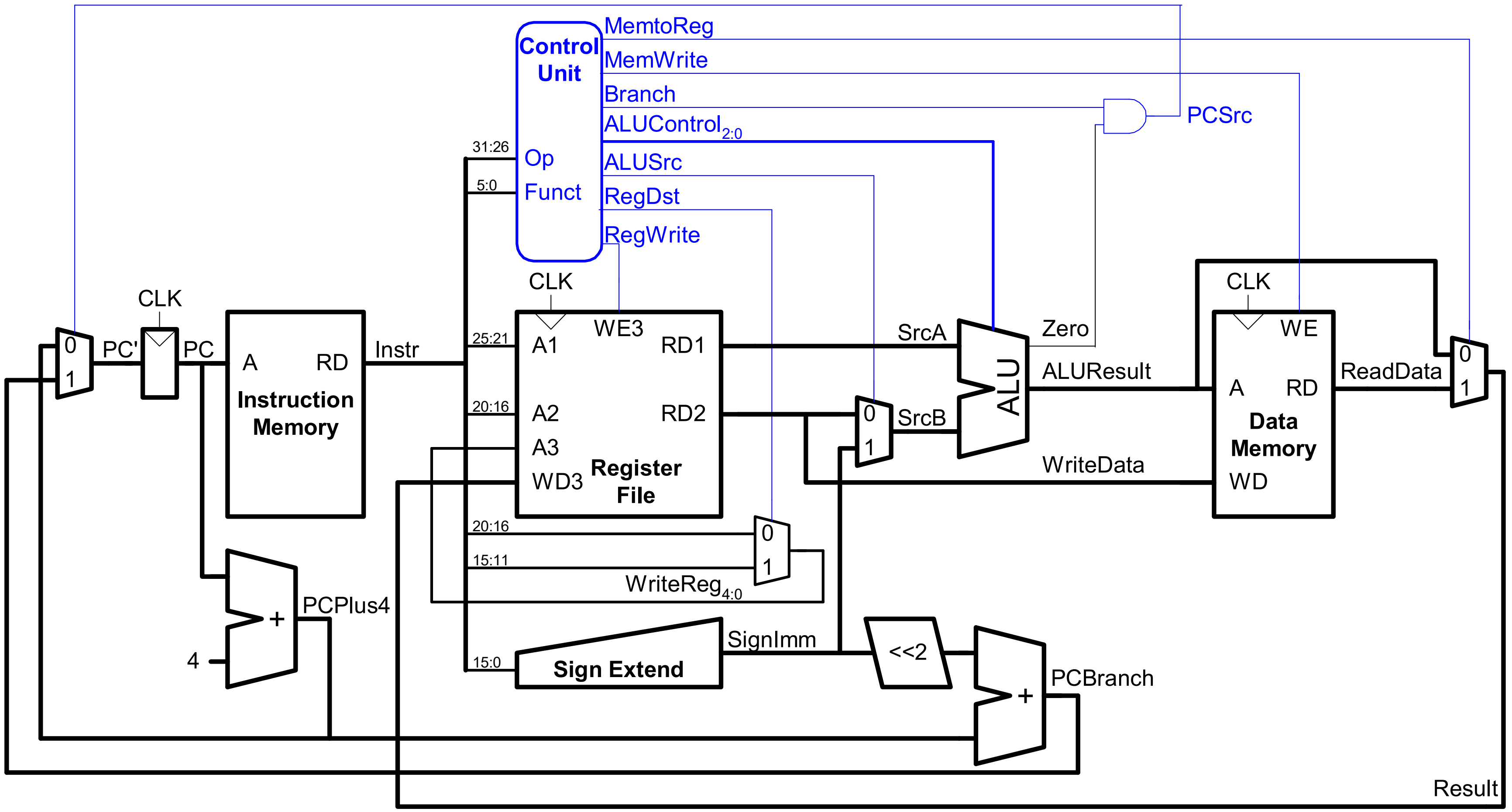
Single-Cycle Datapath Example: `or`



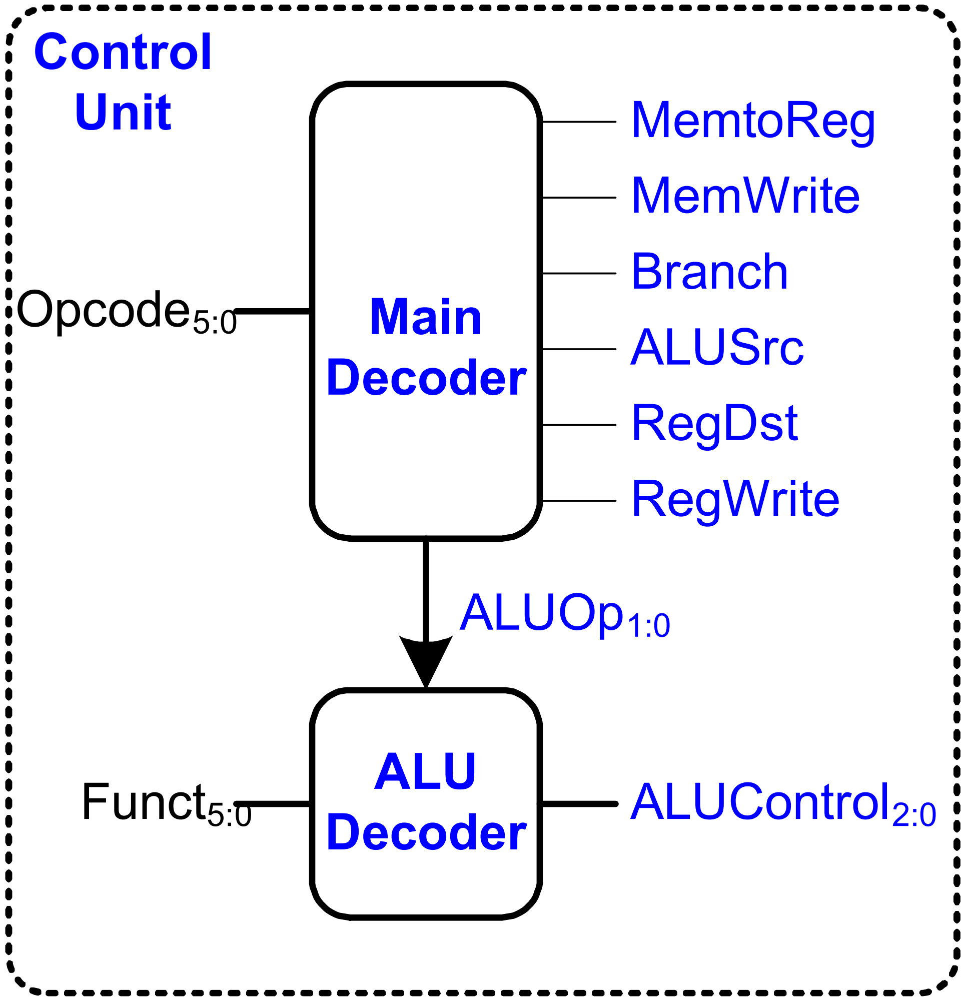
Single-Cycle Datapath: `sw`



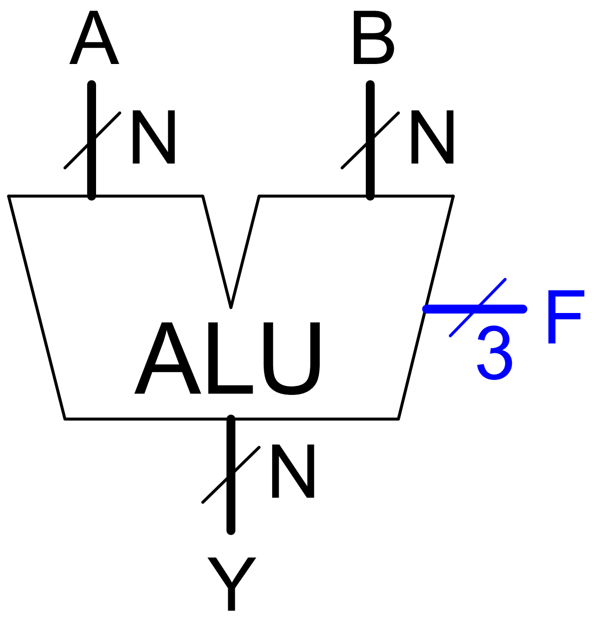
Single-Cycle Control



Control Unit

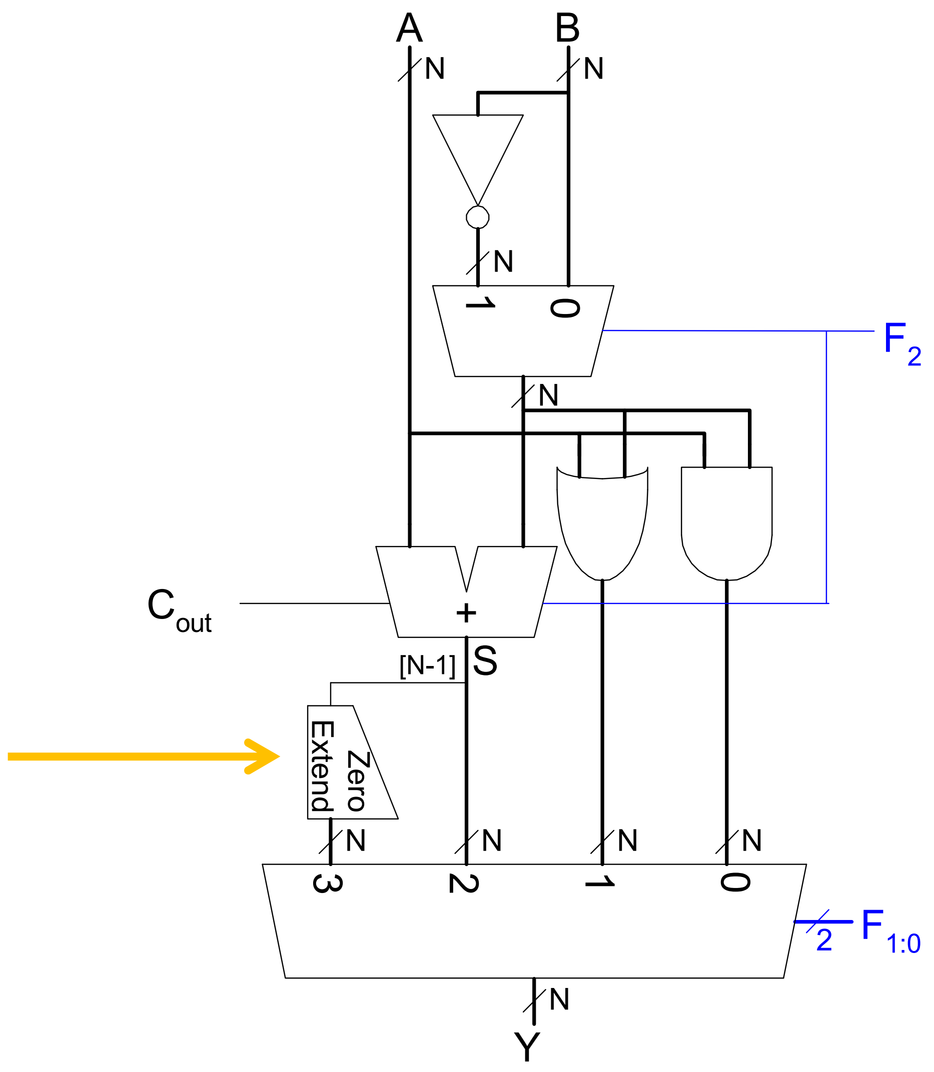


Review: ALU



$F_{2:0}$	Function
000	A & B
001	A | B
010	A + B
011	not used
100	A & ~B
101	A | ~B
110	A - B
111	SLT

Review: ALU



Inverting or not ?

Control Unit: ALU Decoder

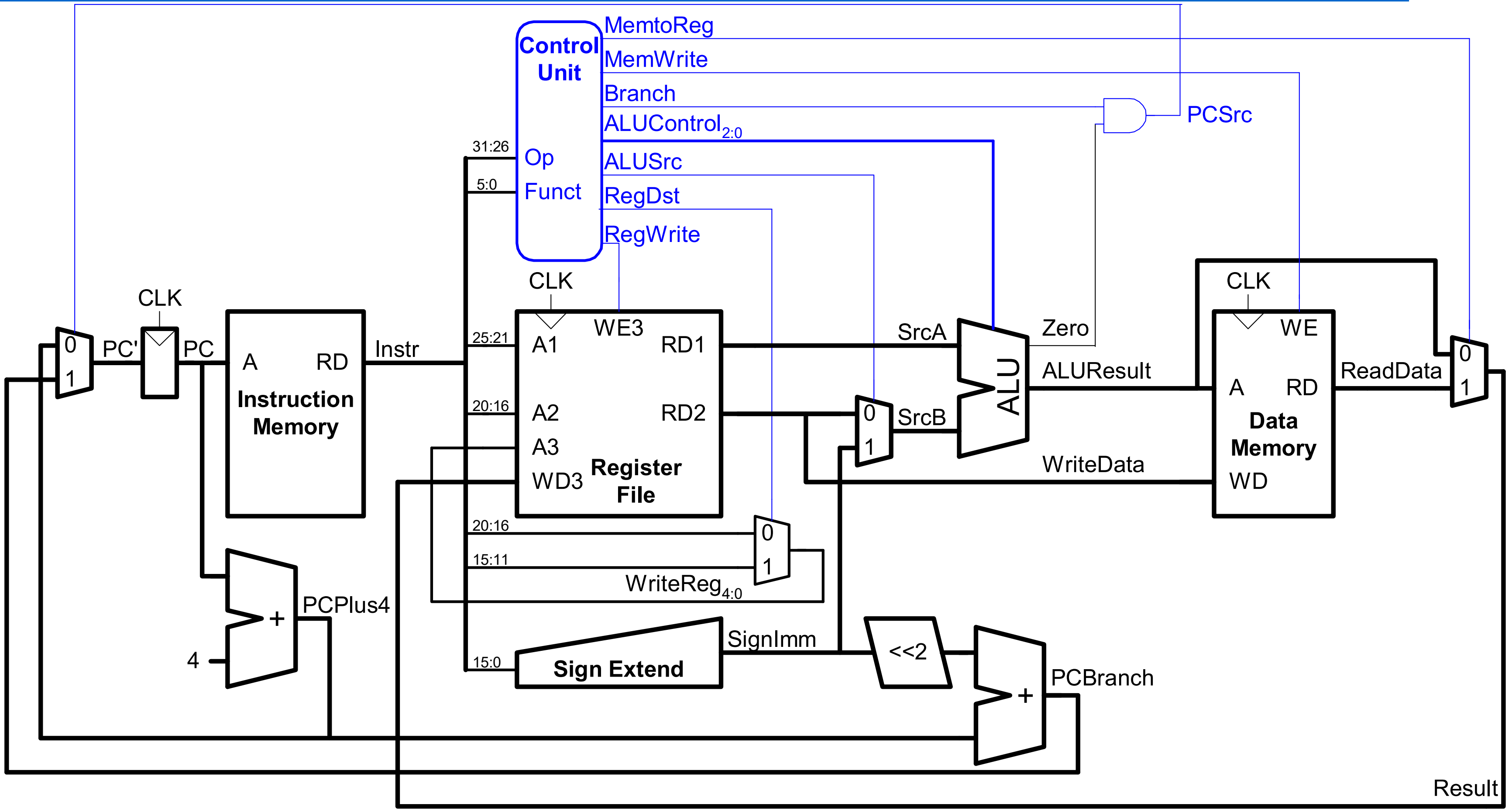
ALUOp$_{1:0}$	Meaning
00	Add
01	Subtract
10	Look at Funct
11	Not Used

ALUOp$_{1:0}$	Funct	ALUControl$_{2:0}$
00	X	010 (Add)
X1	X	110 (Subtract)
1X	100000 (`add`)	010 (Add)
1X	100010 (`sub`)	110 (Subtract)
1X	100100 (`and`)	000 (And)
1X	100101 (`or`)	001 (Or)
1X	101010 (`slt`)	111 (SLT)

Control Unit: Main Decoder

Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$
R-type	000000	1	1	0	0	0	0	10
lw	100011	1	0	1	0	0	0	00
sw	101011	0	X	1	0	1	X	00
beq	000100	0	X	0	1	0	X	01

Extended Functionality: addi

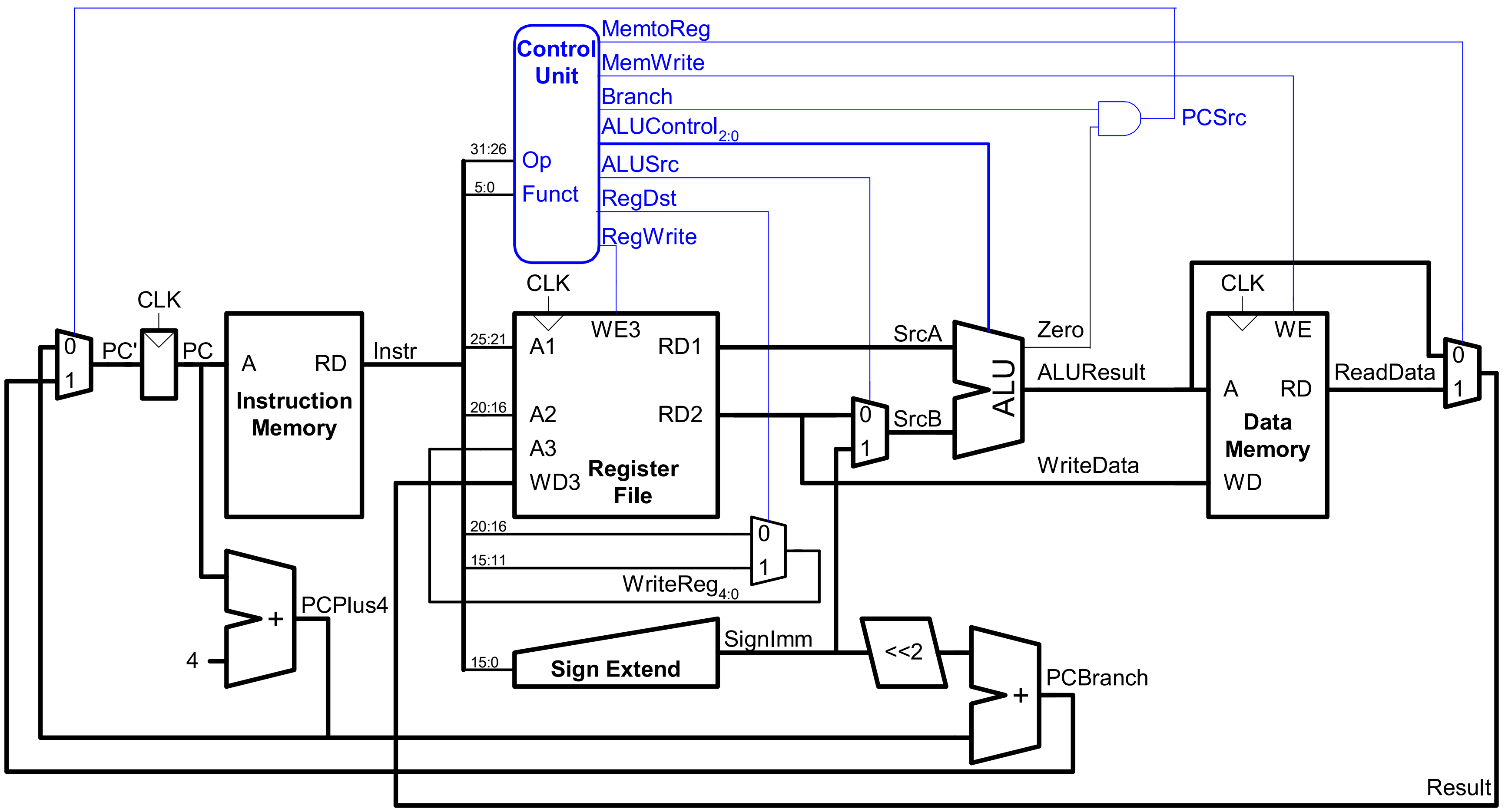


Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$
R-type	000000	1	1	0	0	0	0	10
lw	100011	1	0	1	0	0	0	00
sw	101011	0	X	1	0	1	X	00
beq	000100	0	X	0	1	0	X	01

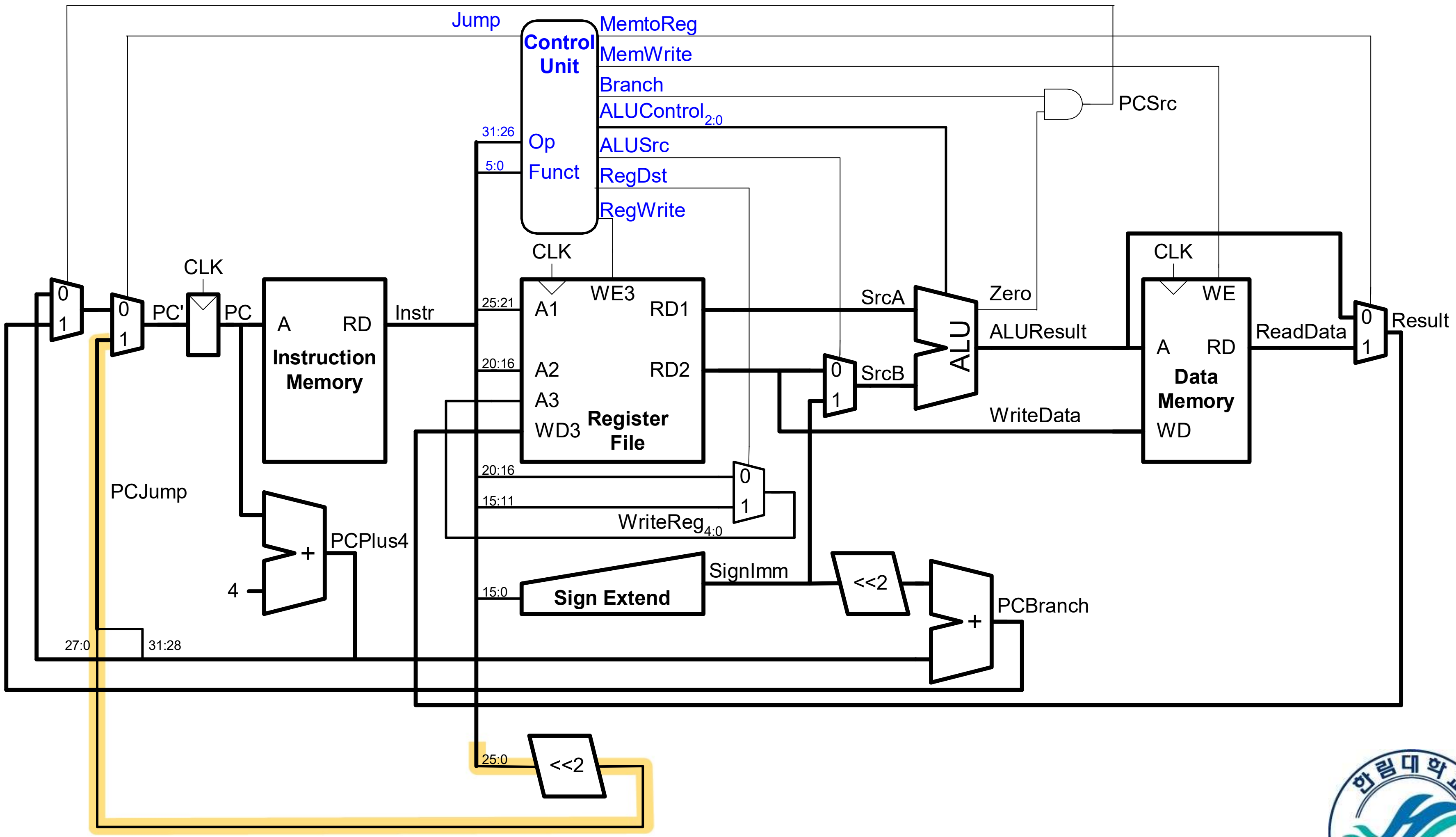
Control Unit: Main Decoder

Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$
R-type	000000	1	1	0	0	0	0	10
lw	100011	1	0	1	0	0	1	00
sw	101011	0	X	1	0	1	X	00
beq	000100	0	X	0	1	0	X	01
addi	**001000**	**1**	**0**	**1**	**0**	**0**	**0**	**00**

Extended Functionality: j



Extended Functionality: j

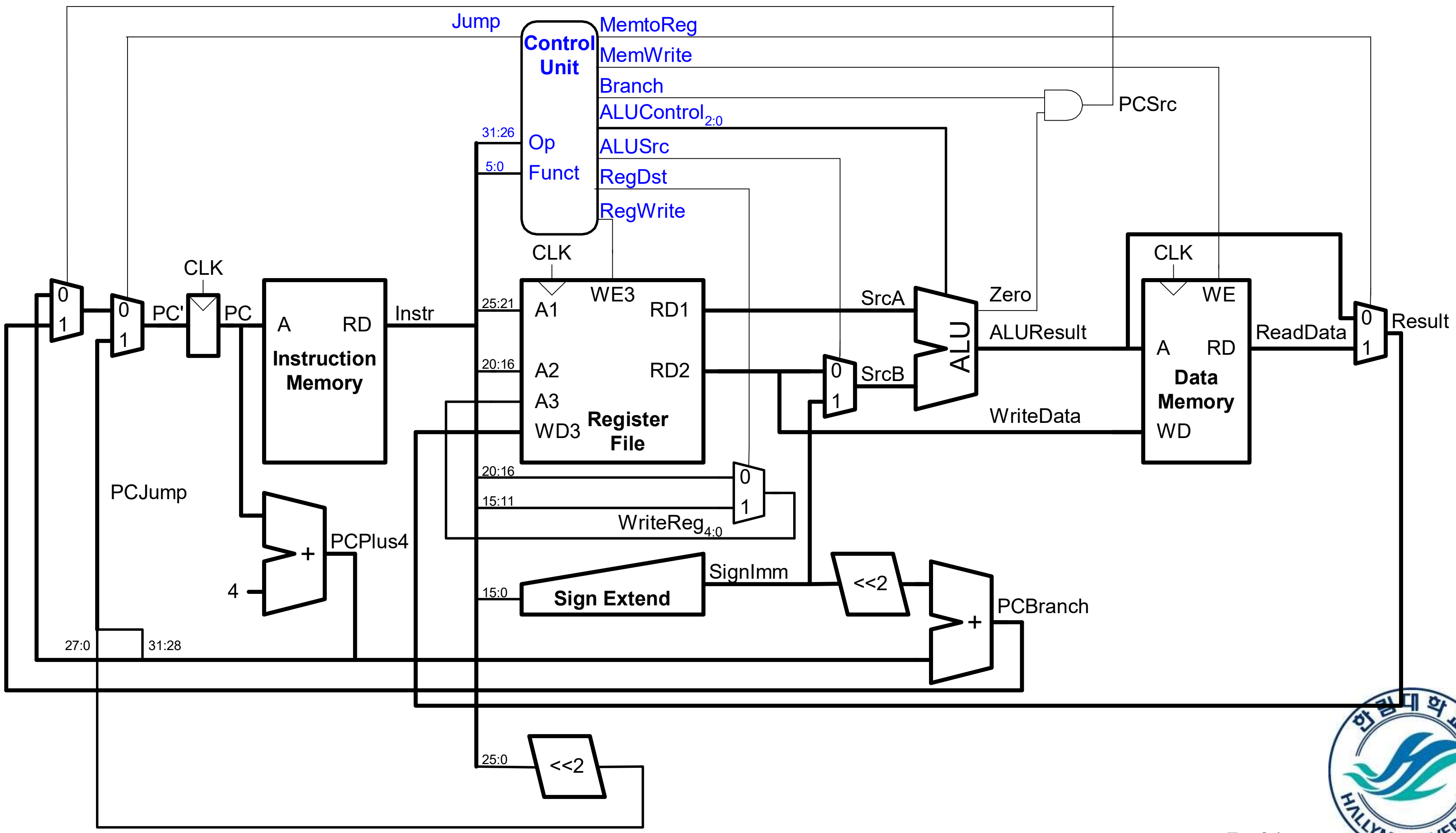


Control Unit: Main Decoder

Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	Jump
R-type	000000	1	1	0	0	0	0	10	0
lw	100011	1	0	1	0	0	1	00	0
sw	101011	0	X	1	0	1	X	00	0
beq	000100	0	X	0	1	0	X	01	0
addi	001000	1	0	1	0	0	0	00	0
j	**000100**	**0**	**X**	**X**	**X**	**0**	**X**	**XX**	**1**

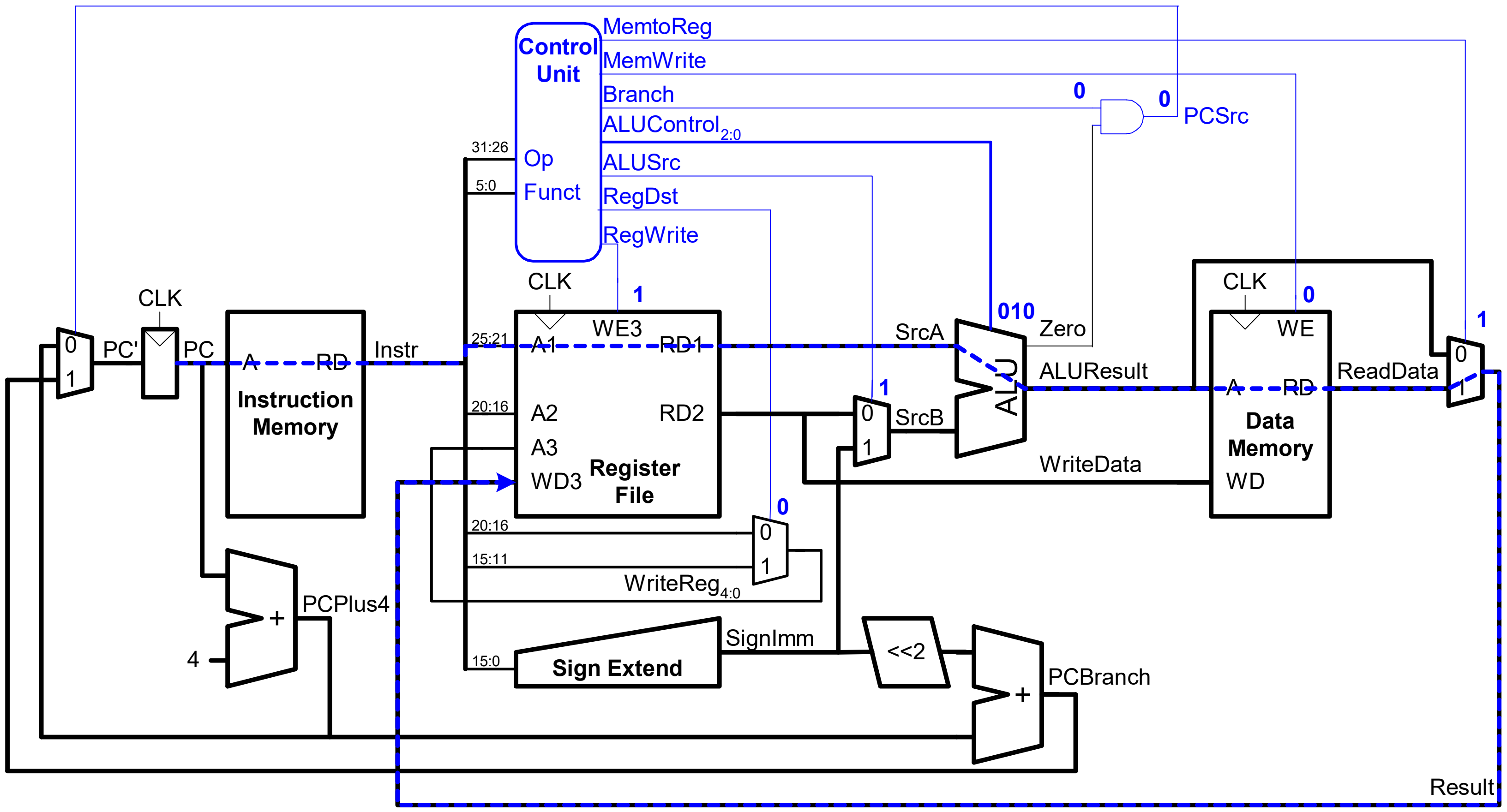
Single-Cycle Performance

How fast is the single-cycle processor?



Single-Cycle Performance

- Clock cycle time is limited by the critical path (`lw`)



Single-Cycle Performance

- Single-cycle critical path:

  $T_c = t_{pcq\_PC} + t_{\text{mem}} + \max(t_{RF\text{read}}, t_{sext}) + t_{\text{mux}} + t_{\text{ALU}} + t_{\text{mem}} + t_{\text{mux}} + t_{RF\text{setup}}$

- In most implementations, limiting paths are: memory, ALU, register file. Thus,

  $T_c = t_{pcq\_PC} + 2t_{\text{mem}} + t_{RF\text{read}} + 2t_{\text{mux}} + t_{\text{ALU}} + t_{RF\text{setup}}$

Single-Cycle Performance Example

Element	Parameter	Delay (ps)
Register clock-to-Q	t_{pcq_PC}	30
Register setup	t_{setup}	20
Multiplexer	t_{mux}	25
ALU	t_{ALU}	200
Memory read	t_{mem}	250
Register file read	$t_{RF\text{read}}$	150
Register file setup	$t_{RF\text{setup}}$	20

$$\begin{aligned} T_c &= t_{pcq\_PC} + 2t_{\text{mem}} + t_{RF\text{read}} + 2t_{\text{mux}} + t_{\text{ALU}} + t_{RF\text{setup}} \\ &= [30 + 2(250) + 150 + 2(25) + 200 + 20]\text{ ps} \\ &= 950\text{ ps} \end{aligned}$$

Single-Cycle Performance Example

- For a program with 100 billion instructions executing on a single-cycle MIPS processor,

$$\begin{aligned}\text{Execution Time} &= (\text{\# instructions})(\text{cycles/instruction})(\text{seconds/cycle}) \\ &= (100 \times 10^{9})(1)(950 \times 10^{-12}\text{ s}) \\ &= 95 \text{ seconds}\end{aligned}$$