CS222: Computer Architecture

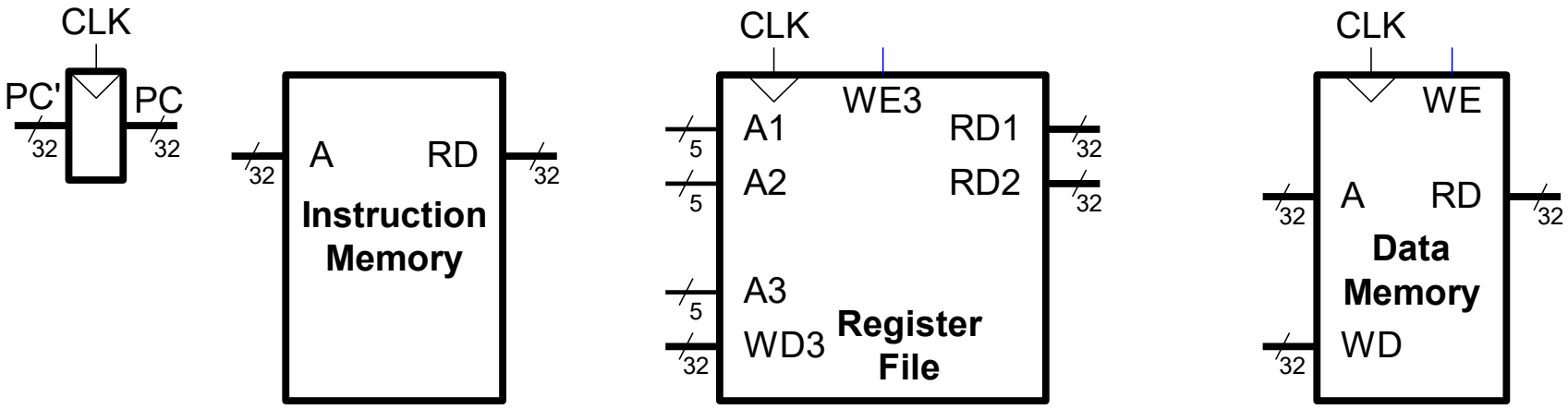
Instructors:
Dr Ahmed Shalaby http://bu.edu.eg/staff/ahmedshalaby14#

الاحترام - الادب - الاخلاق
الطالب - المعيد - الدكتور

© Digital Design and Computer Architecture, 2nd Edition, 2012

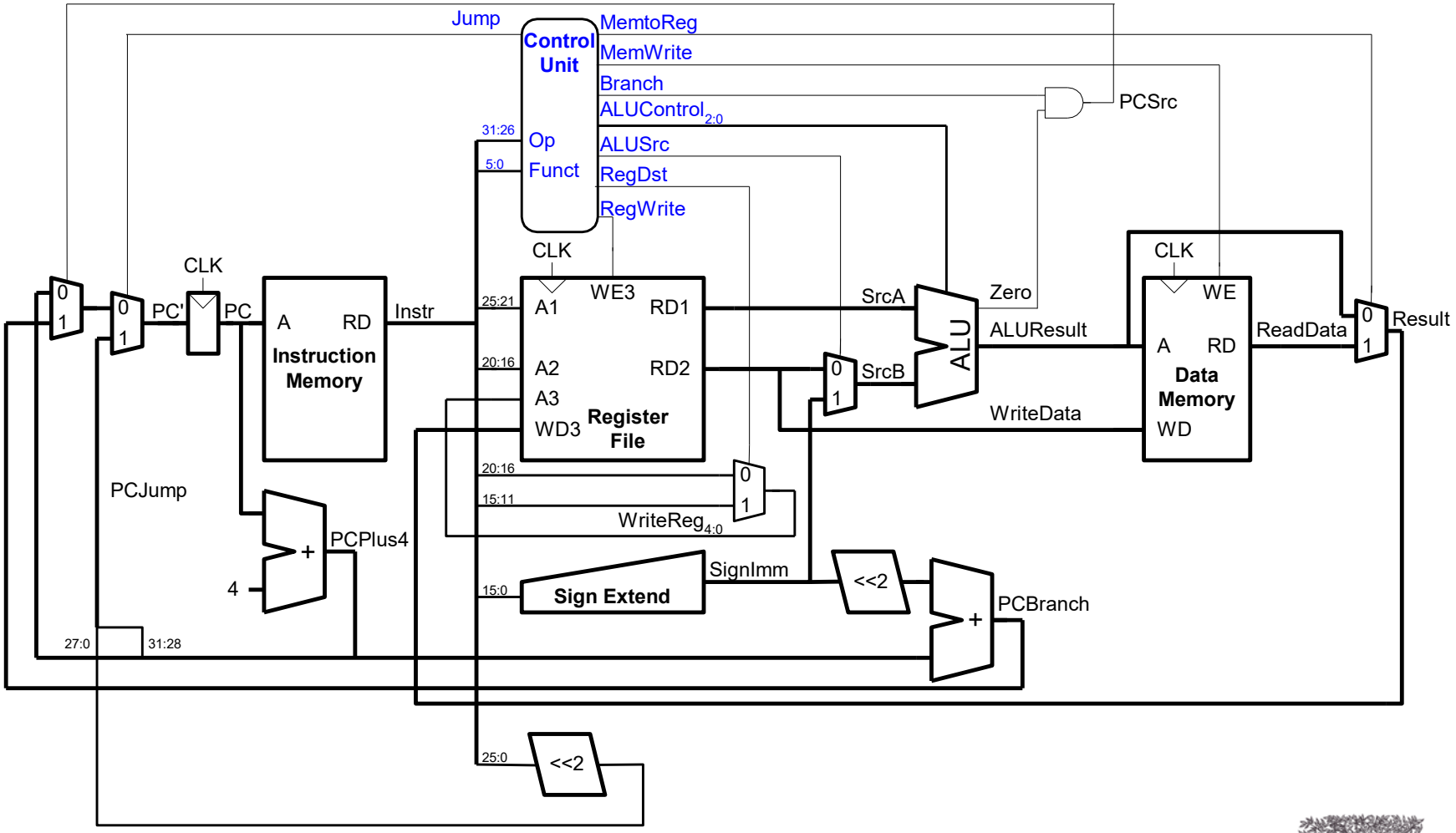
ELSEVIER

Single-Cycle Processor: State Elements



ELSEVIER

Review: Single-Cycle Processor



MICROARCHITECTURE
ELSEVIER

Review: Processor Performance

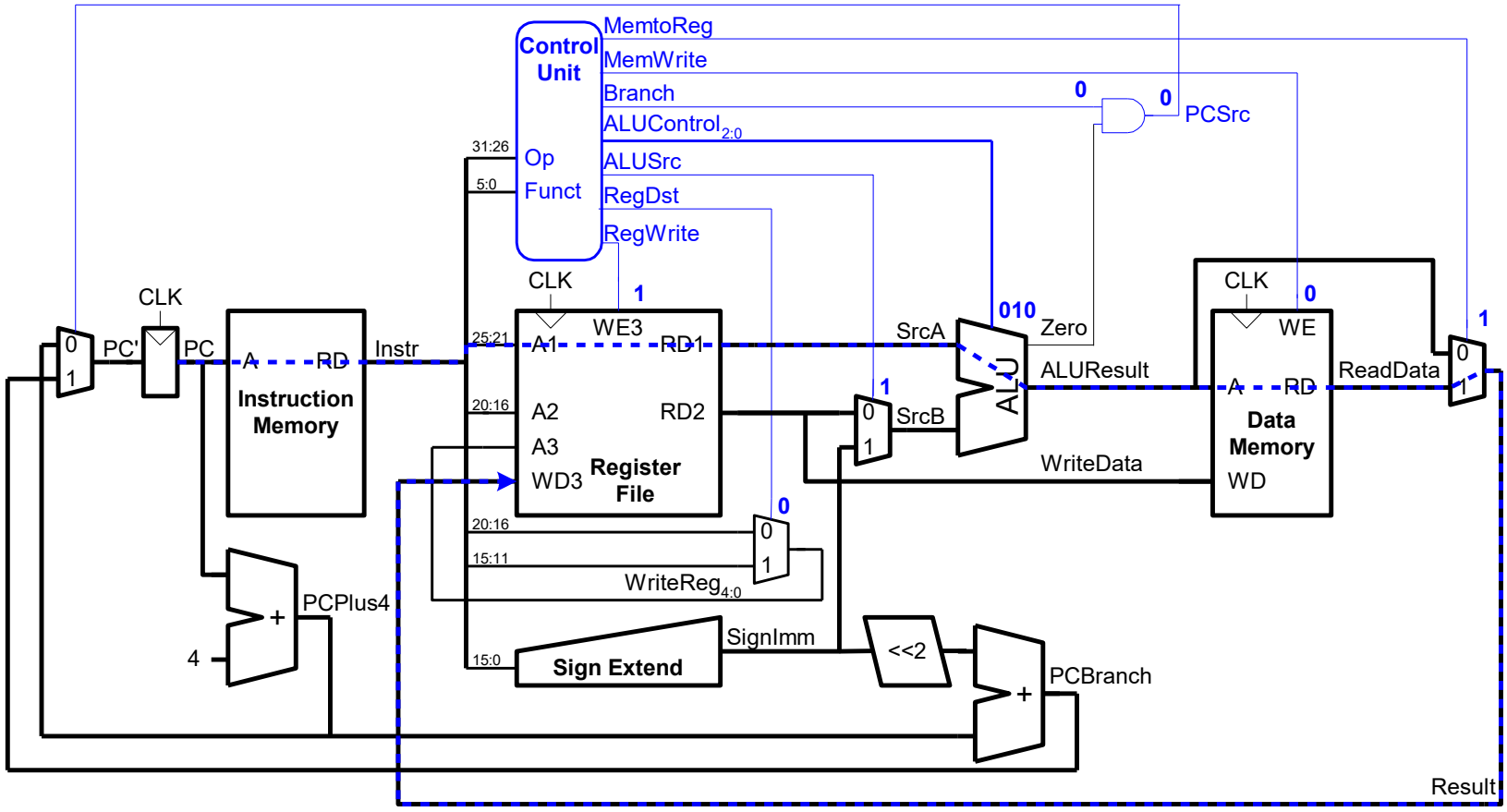
Program Execution Time

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

= # instructions x CPI x T_C

ELSEVIER

Single-Cycle Performance



T_C limited by critical path (`lw`)

MICROARCHITECTURE
ELSEVIER

Single-Cycle Performance

- Single-cycle critical path:

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

- Typically, limiting paths are:
  - memory, ALU, register file
  - $T_c = t_{pcq\_PC} + 2t_{\text{mem}} + t_{RF\text{read}} + t_{\text{mux}} + t_{\text{ALU}} + t_{RF\text{setup}}$

ELSEVIER

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_{RFread}	150
Register file setup	$t_{RFsetup}$	20

$$T_c = t_{pcq\_PC} + 2t_{mem} + t_{RFread} + t_{mux} + t_{ALU} + t_{RFsetup}$$
$$= [30 + 2(250) + 150 + 25 + 200 + 20] \text{ ps}$$
$$= 925 \text{ ps}$$

ELSEVIER

Single-Cycle Performance Example

Program with 100 billion instructions:

Execution Time = # instructions x CPI x T_C

$= (100 \times 10^9)(1)(925 \times 10^{-12}\ \text{s})$

$=$ **92.5 seconds**

ELSEVIER

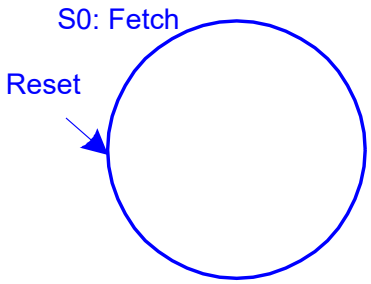
Microarchitecture

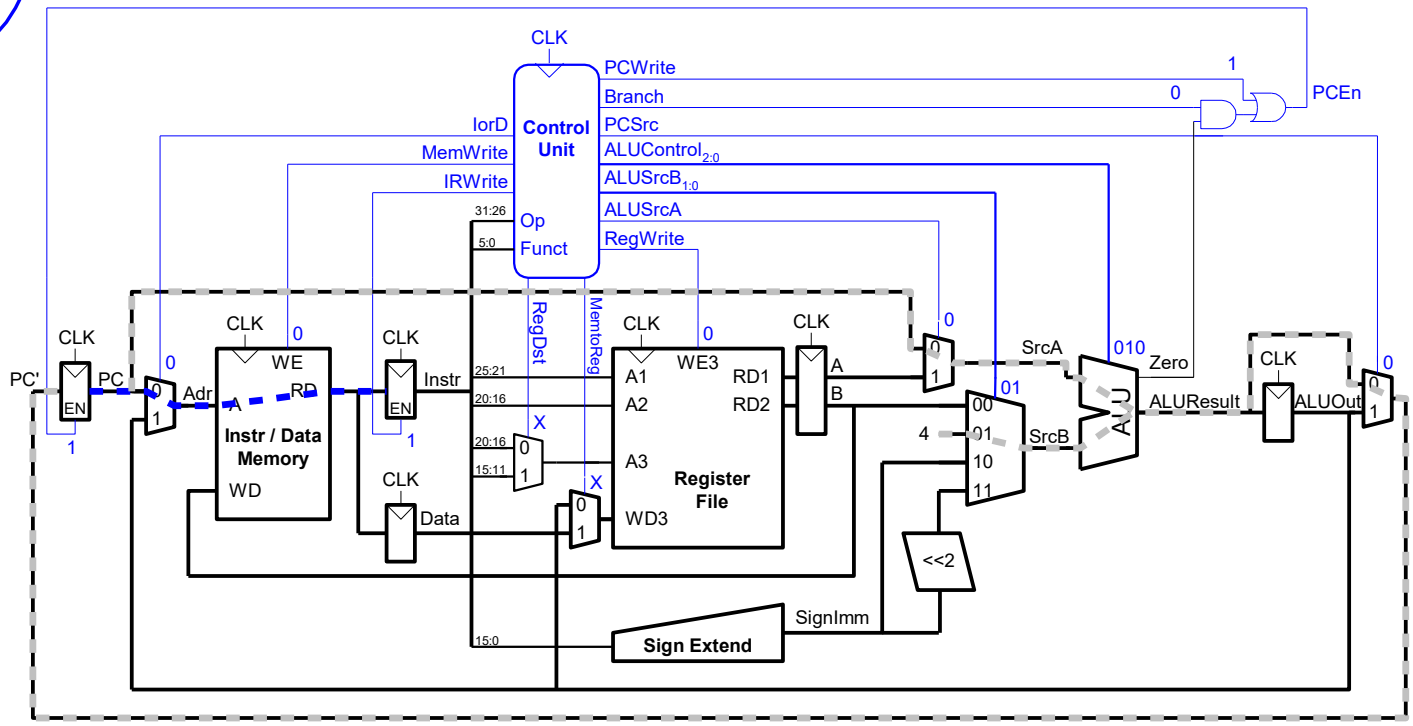
- Multiple implementations for a single architecture:
  - **Single-cycle:** Each instruction executes in a single cycle
  - ~~**Multicycle:** Each instruction is broken into series of shorter steps~~
  - **Pipelined:** Each instruction broken up into series of steps & multiple instructions execute at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

Pipelined MIPS Processor

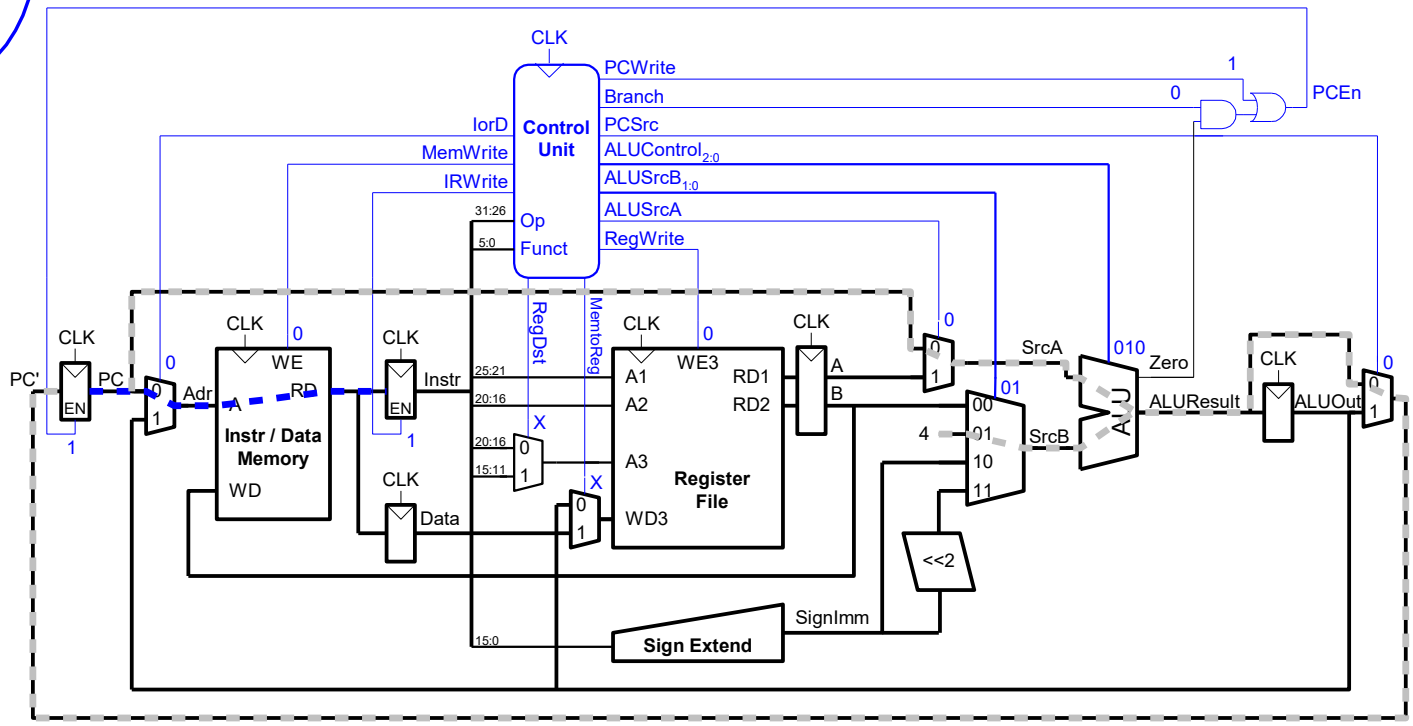
- Temporal parallelism
- Divide single-cycle processor into 5 stages:
  - Fetch
  - Decode
  - Execute
  - Memory
  - Writeback
- Add pipeline registers between stages

Main Controller FSM: Fetch



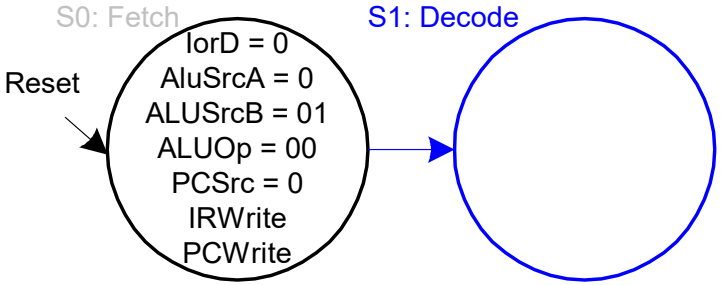


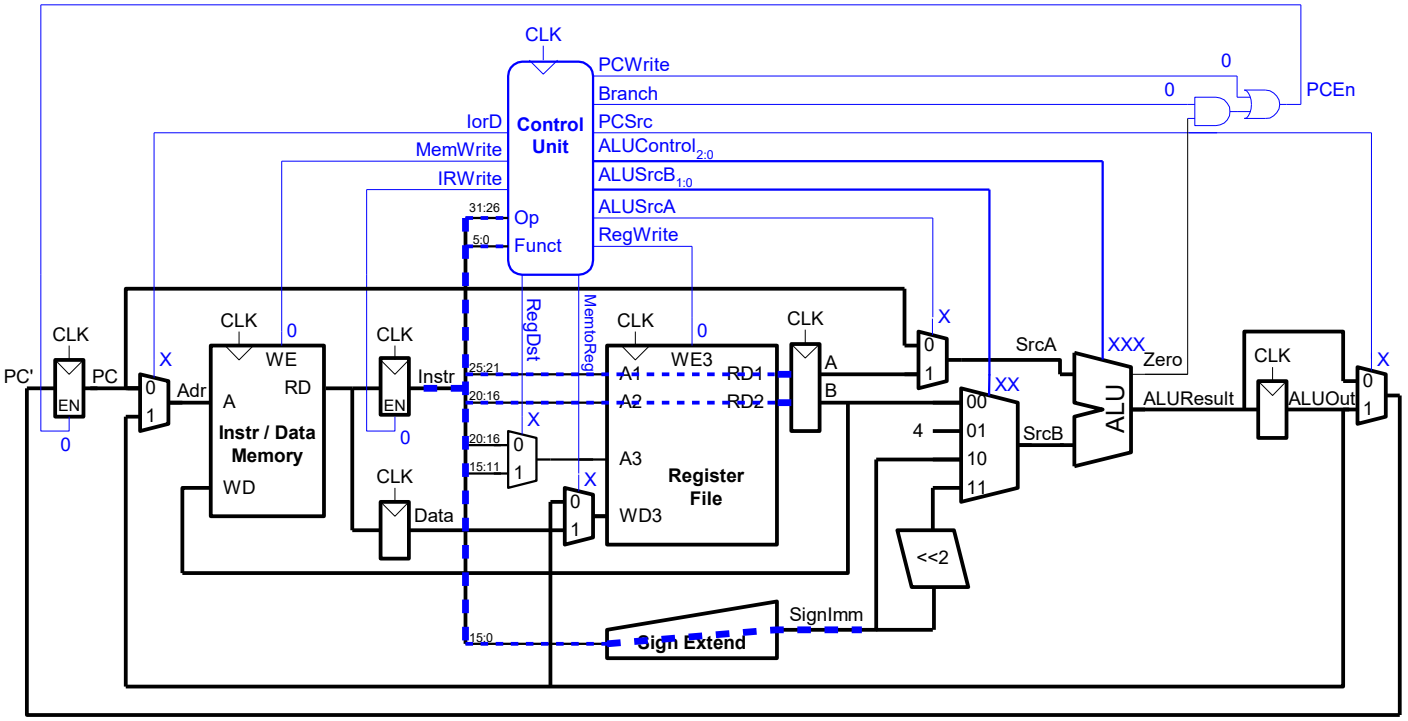
Main Controller FSM: Fetch



ELSEVIER

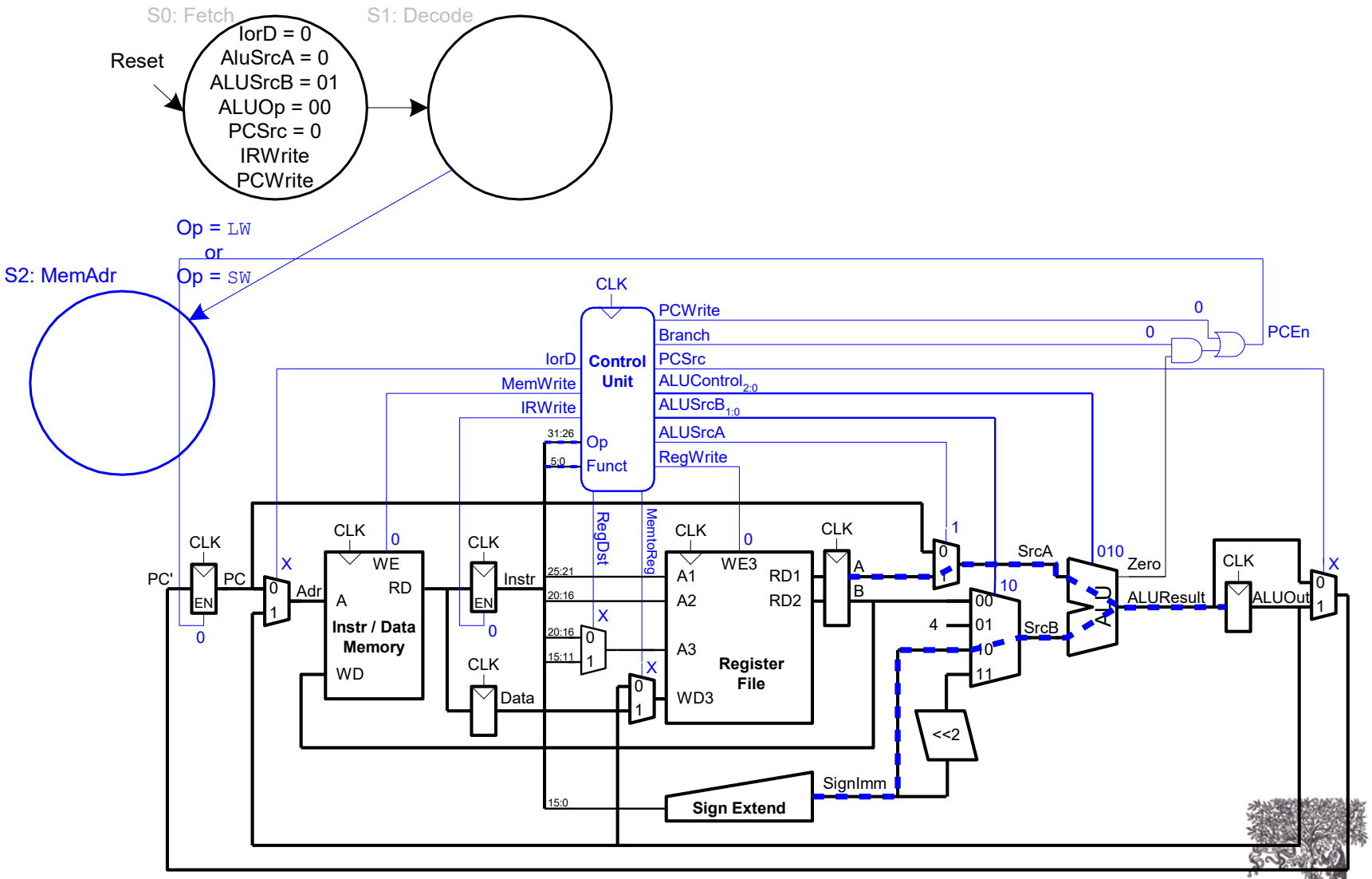
Main Controller FSM: Decode





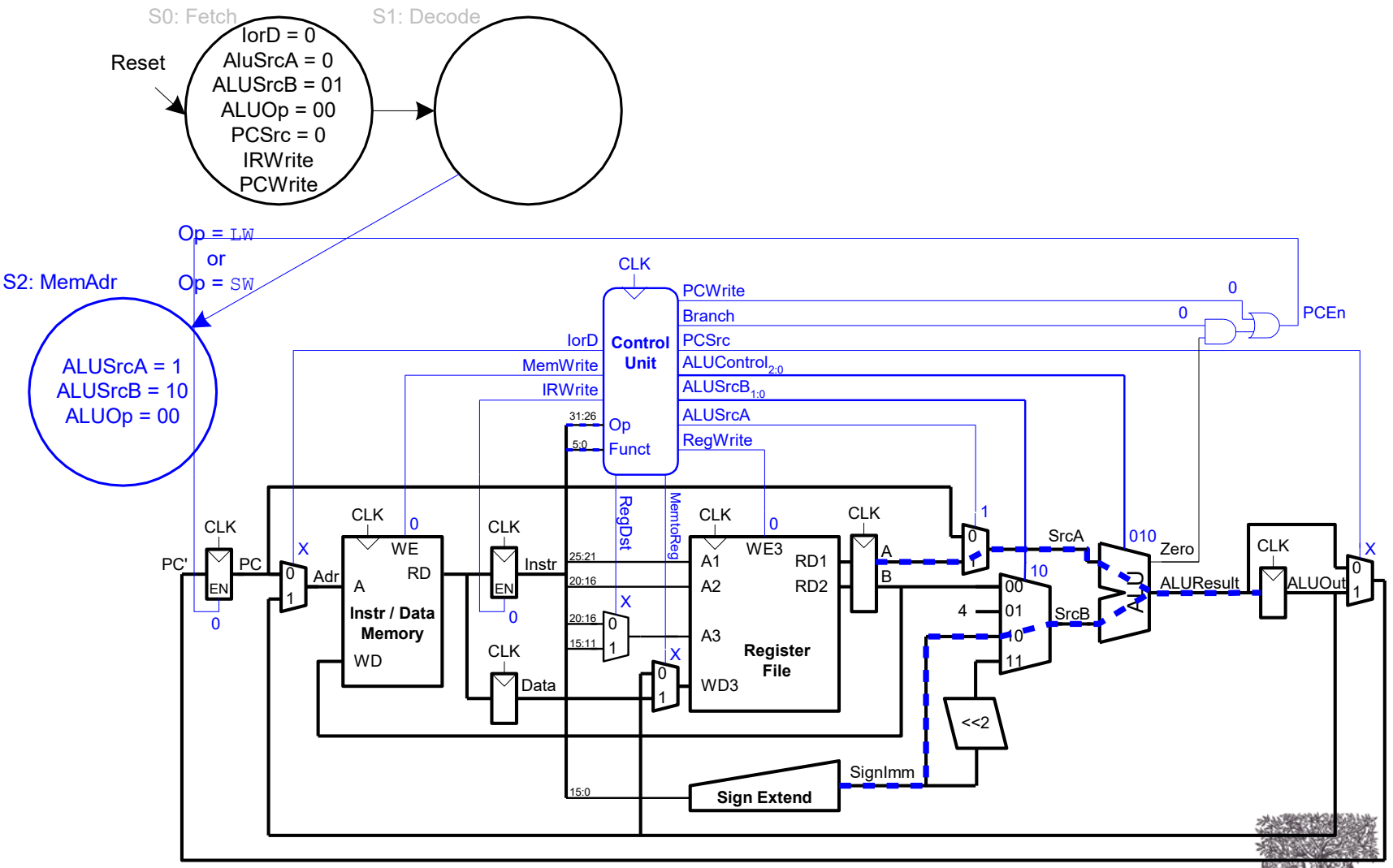
ELSEVIER

Main Controller FSM: Address

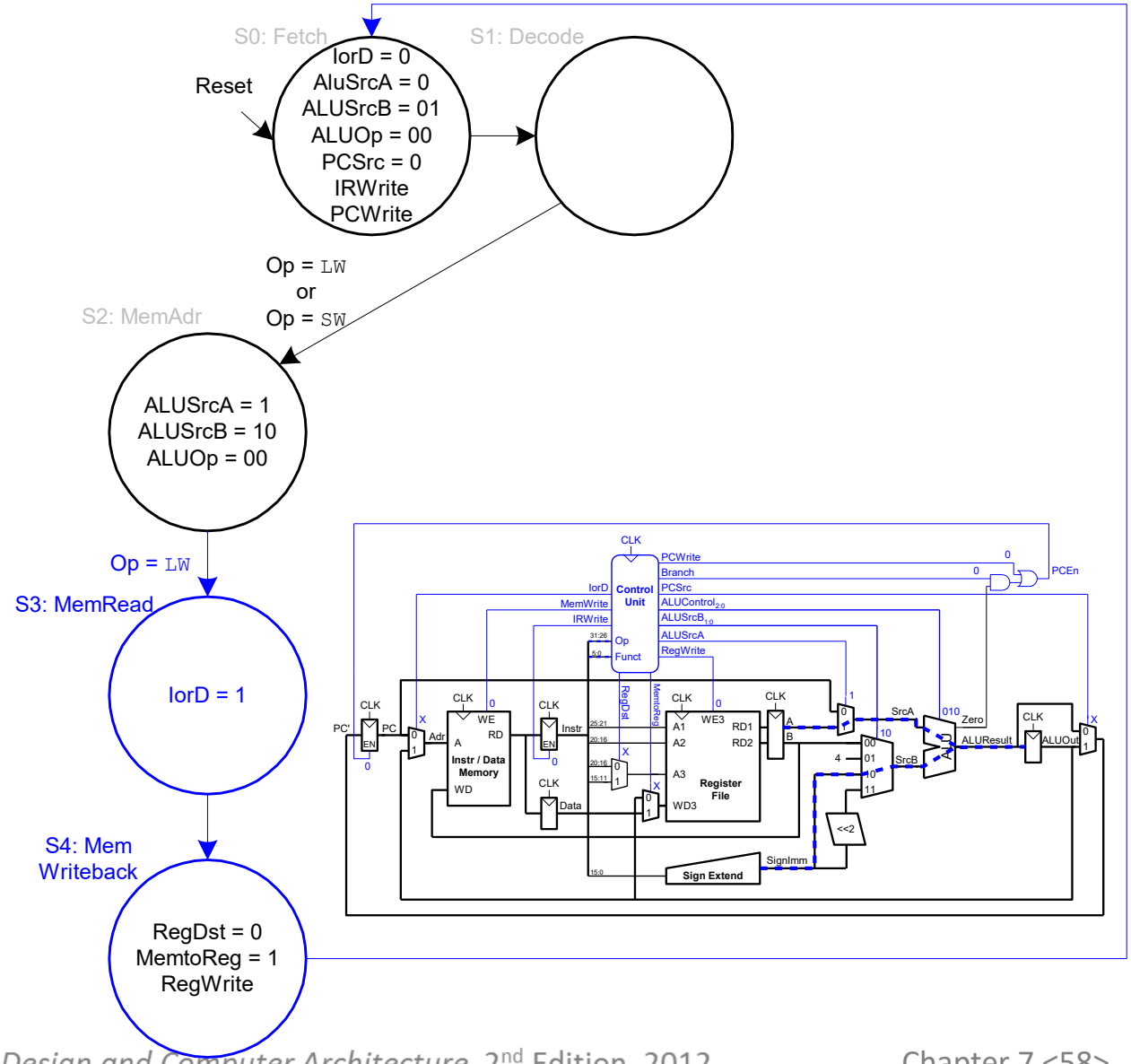


MICROARCHITECTURE
ELSEVIER

Main Controller FSM: Address

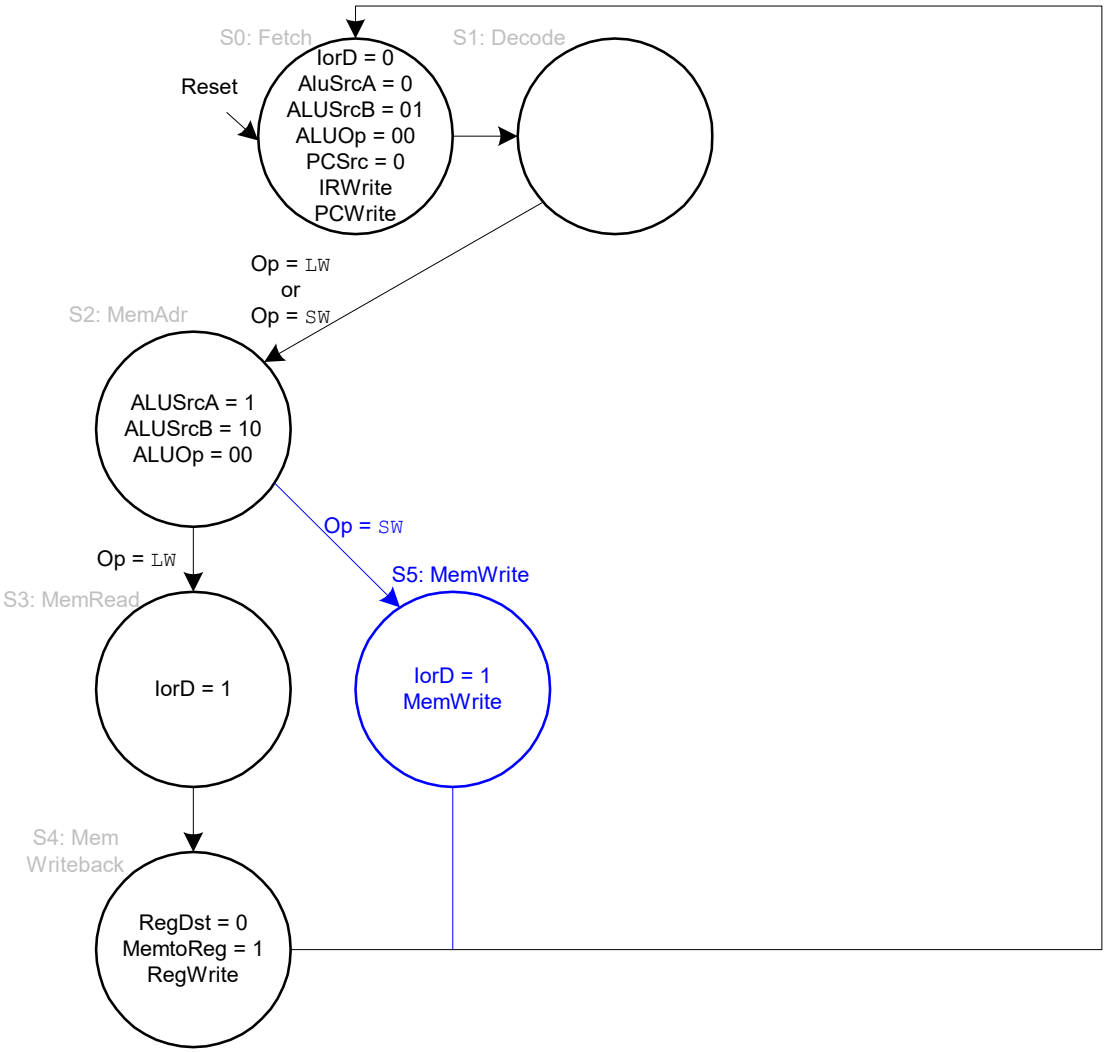


Main Controller FSM: lw

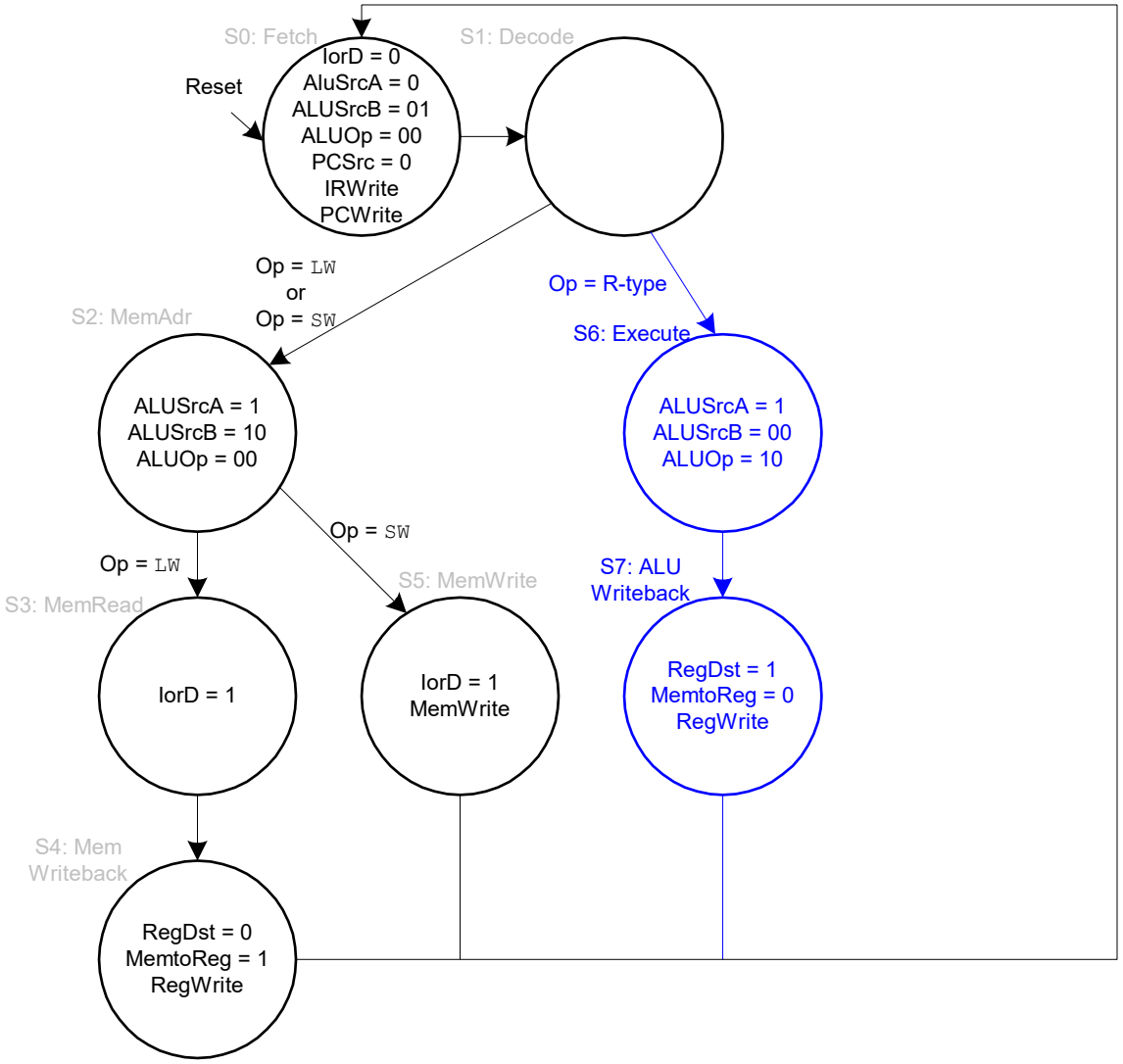


MICROARCHITECTURE
ELSEVIER

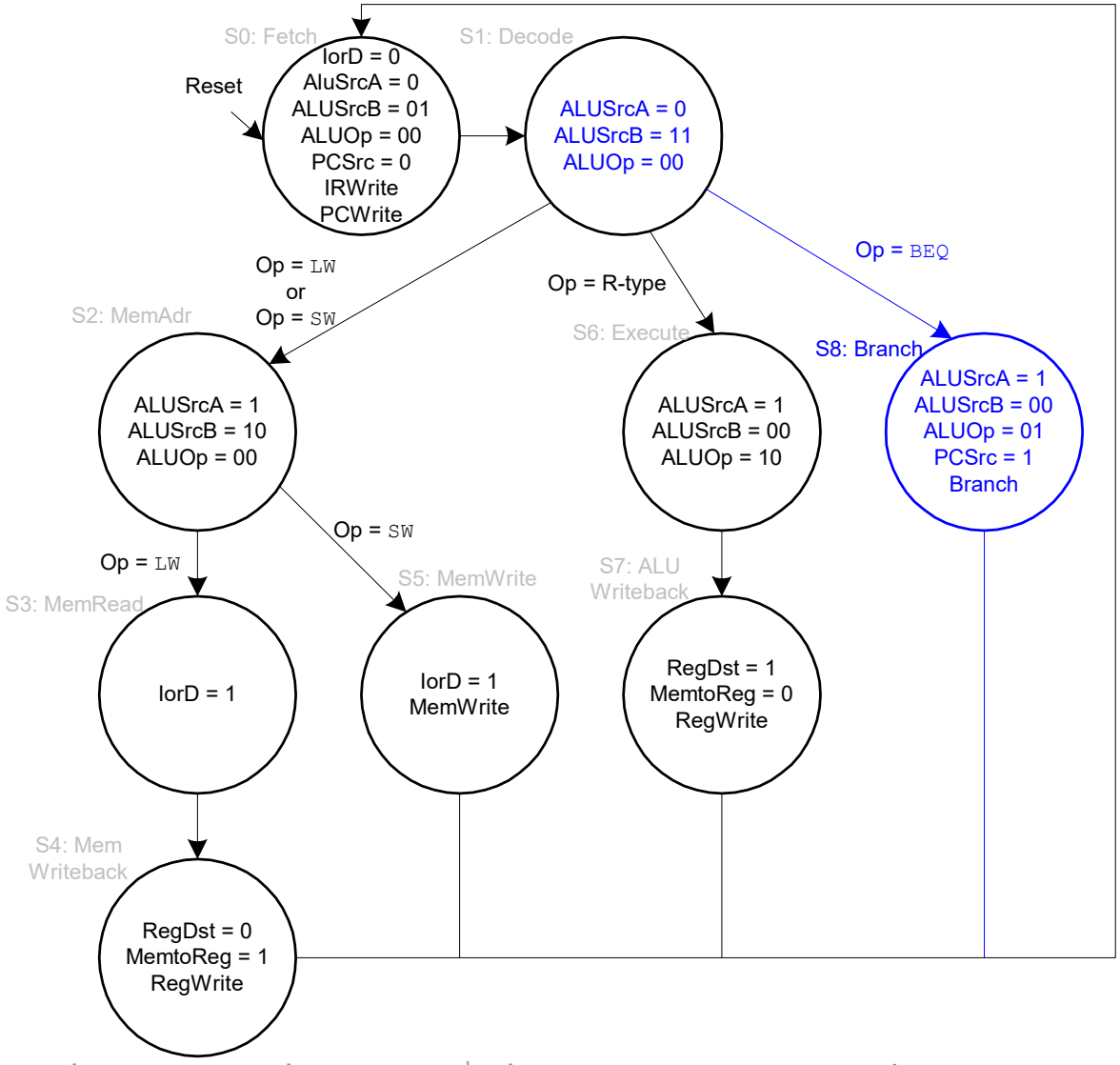
Main Controller FSM: SW



Main Controller FSM: R-Type

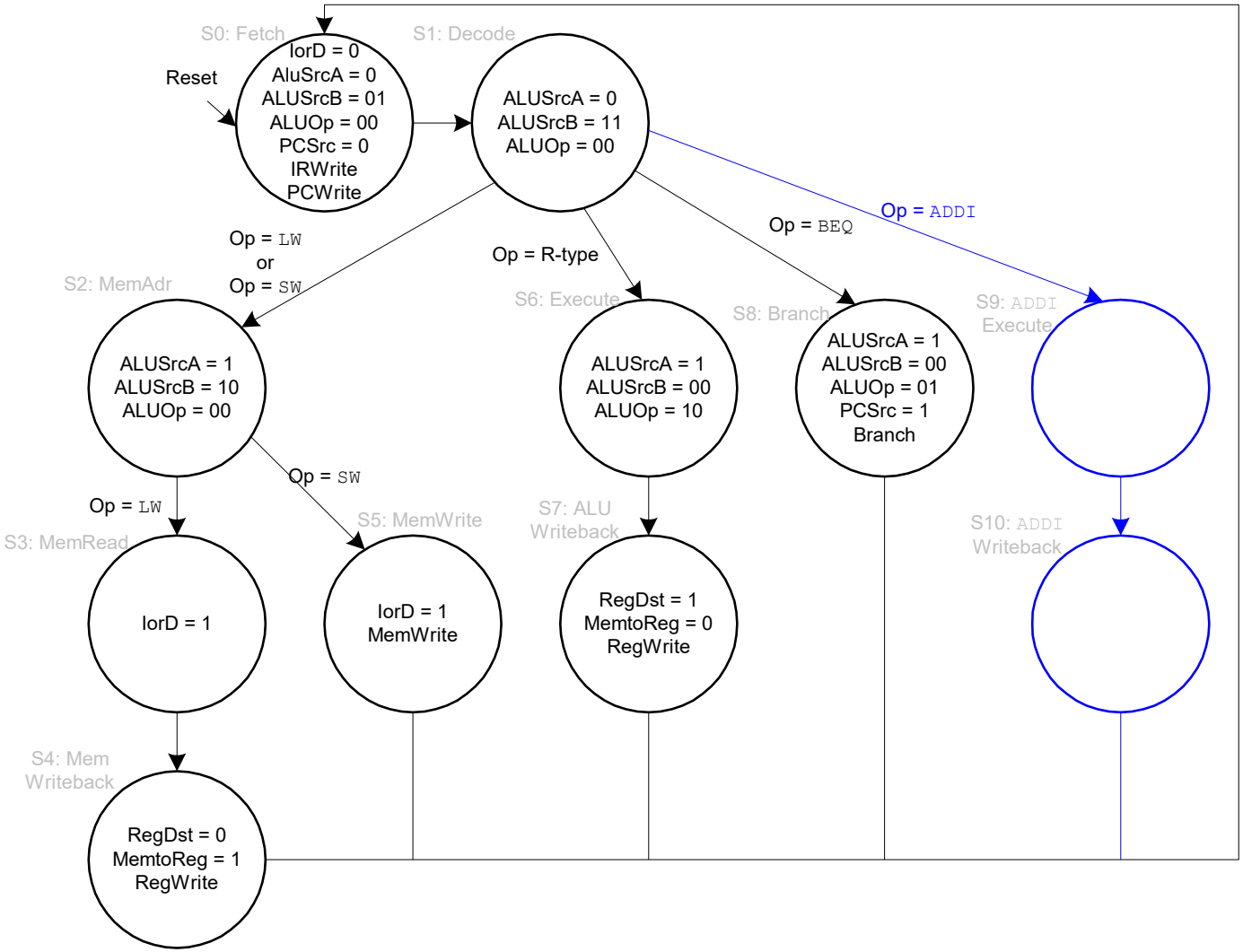


Main Controller FSM: beq

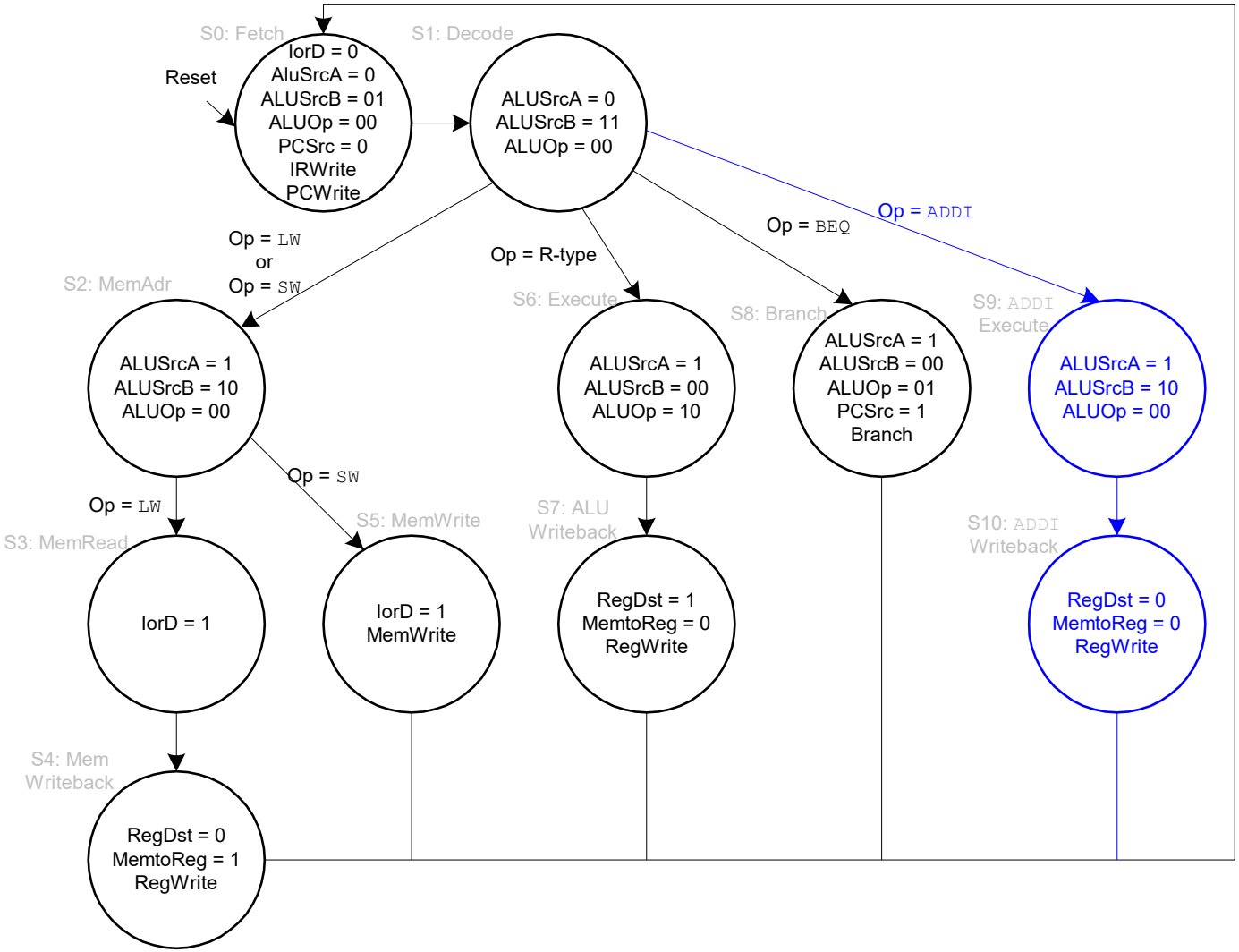


ELSEVIER

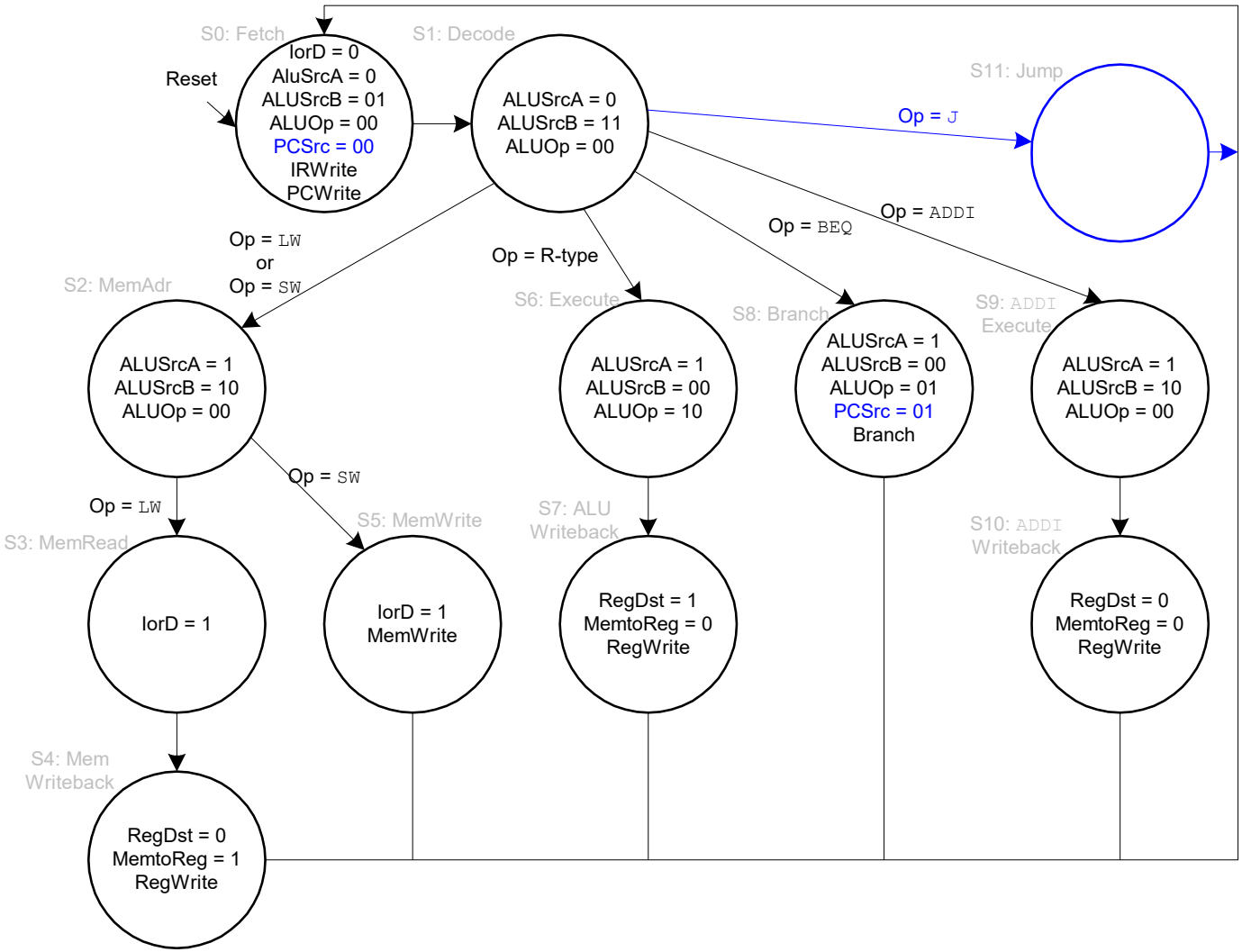
Extended Functionality: addi



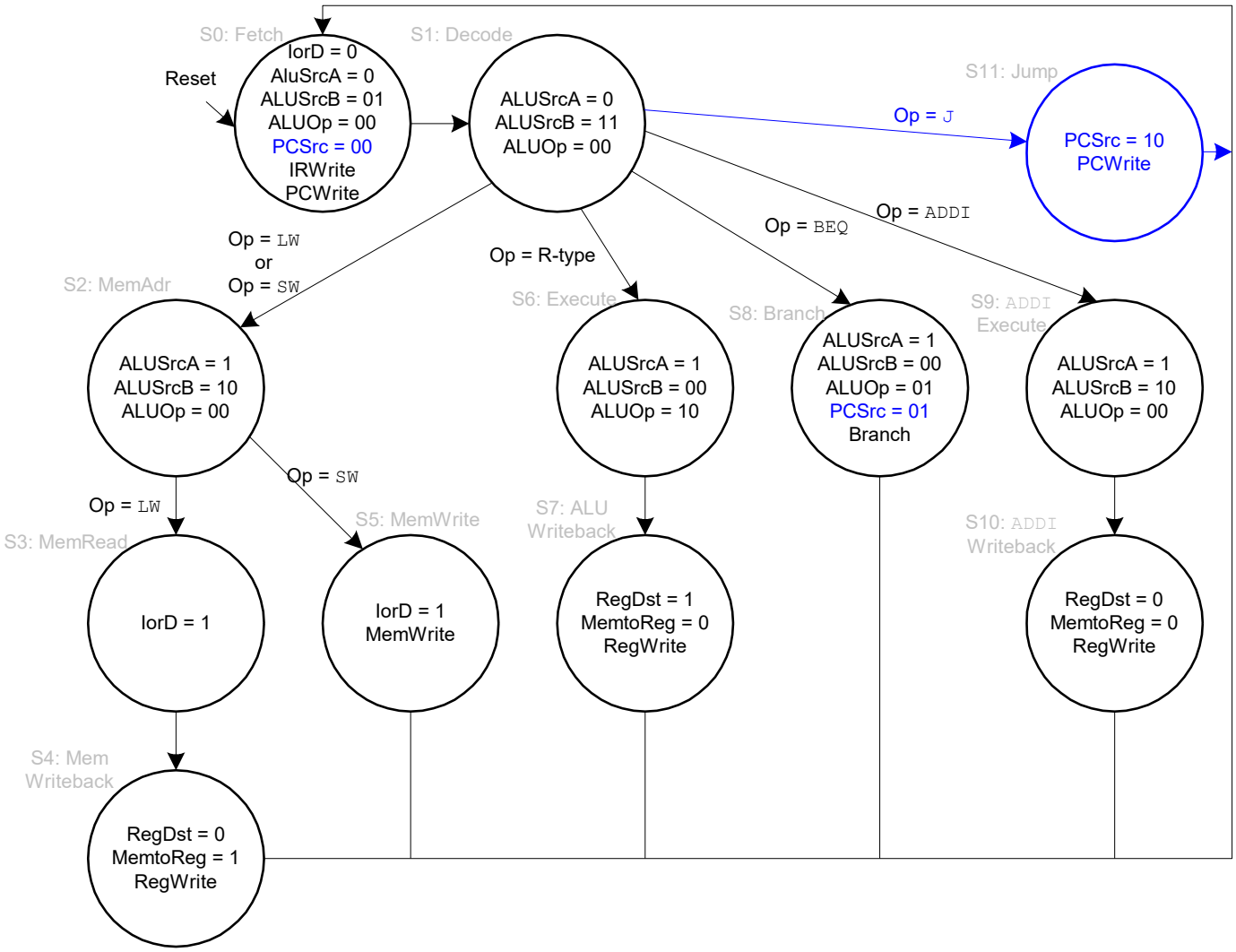
Main Controller FSM: addi



Main Controller FSM: j

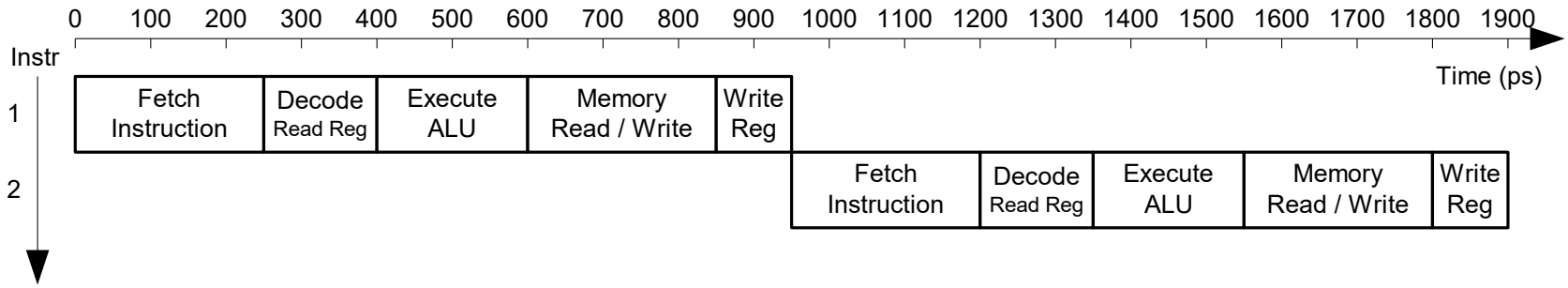


Main Controller FSM: j

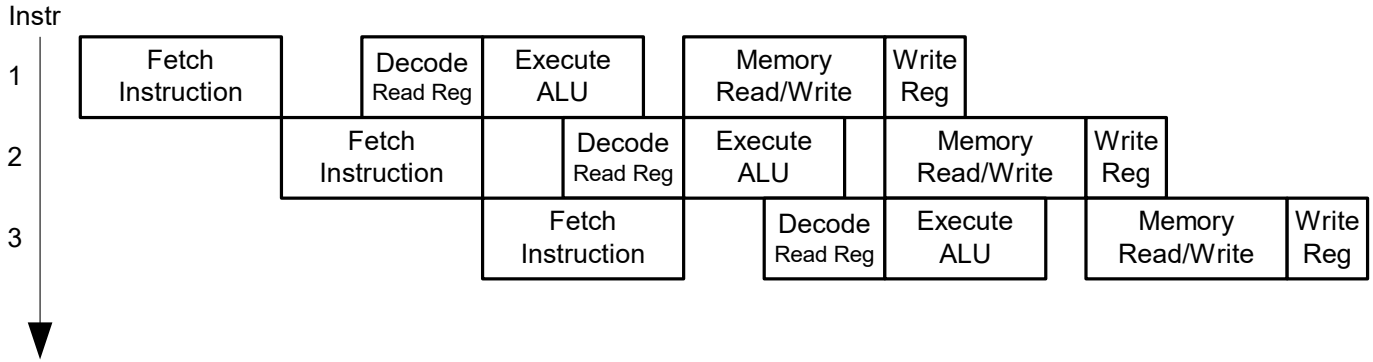


Single-Cycle vs. Pipelined

Single-Cycle

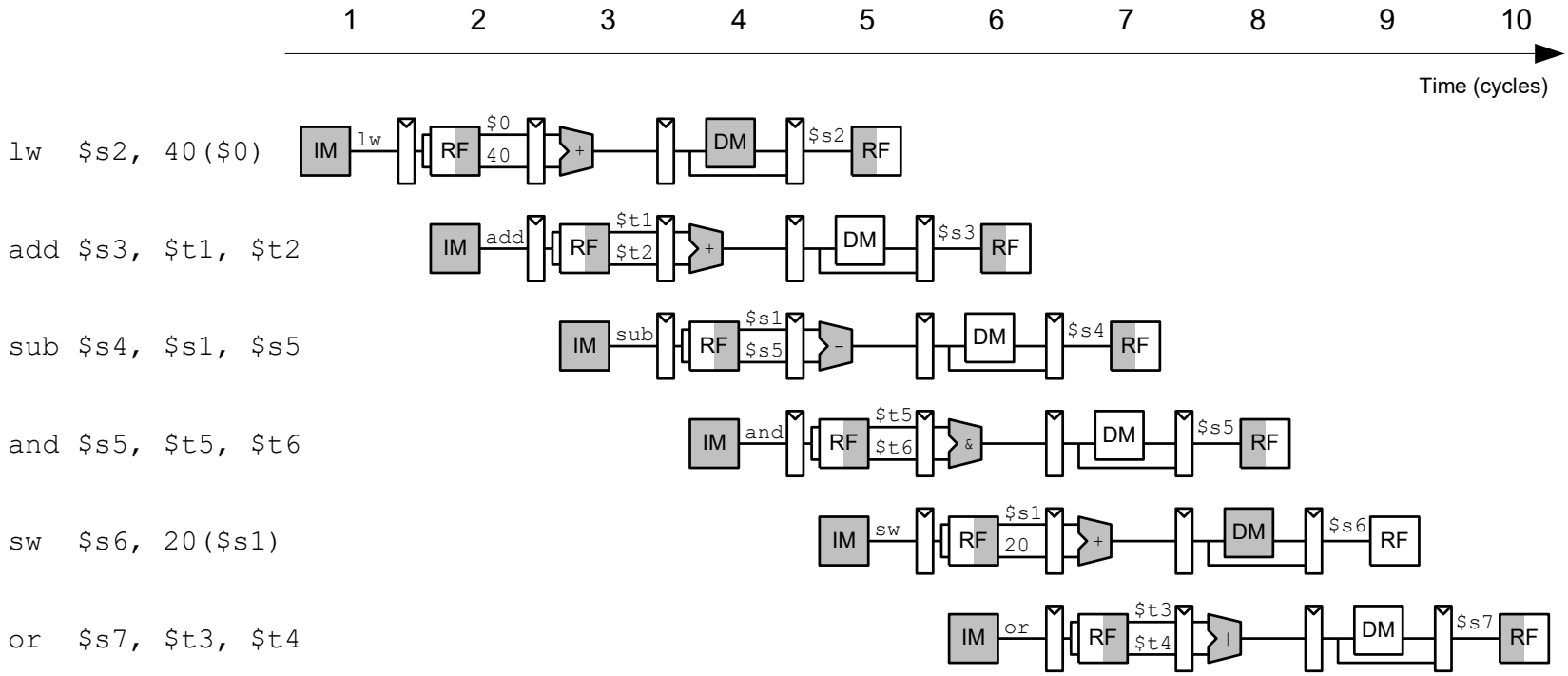


Pipelined



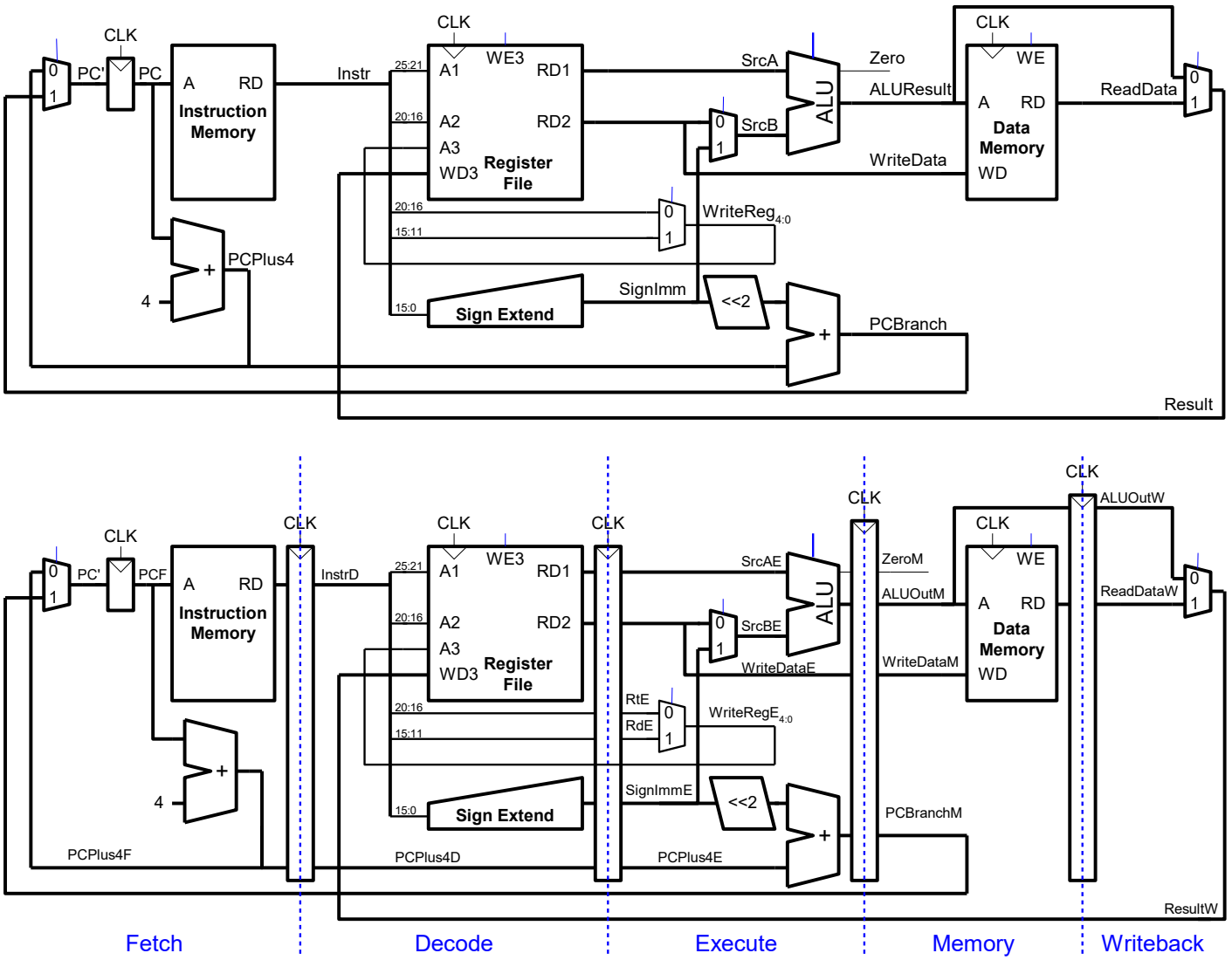
ELSEVIER

Pipelined Processor Abstraction



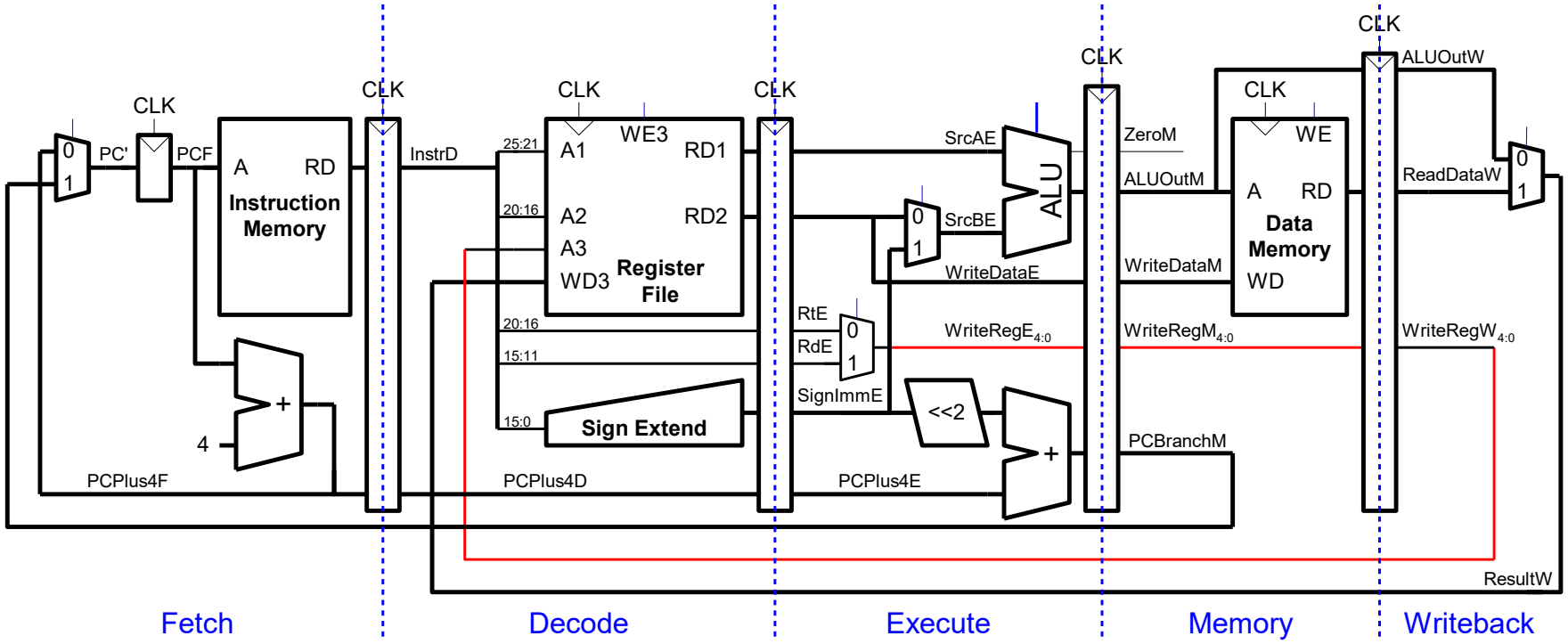
ELSEVIER

Single-Cycle & Pipelined Datapath



MICROARCHITECTURE
ELSEVIER

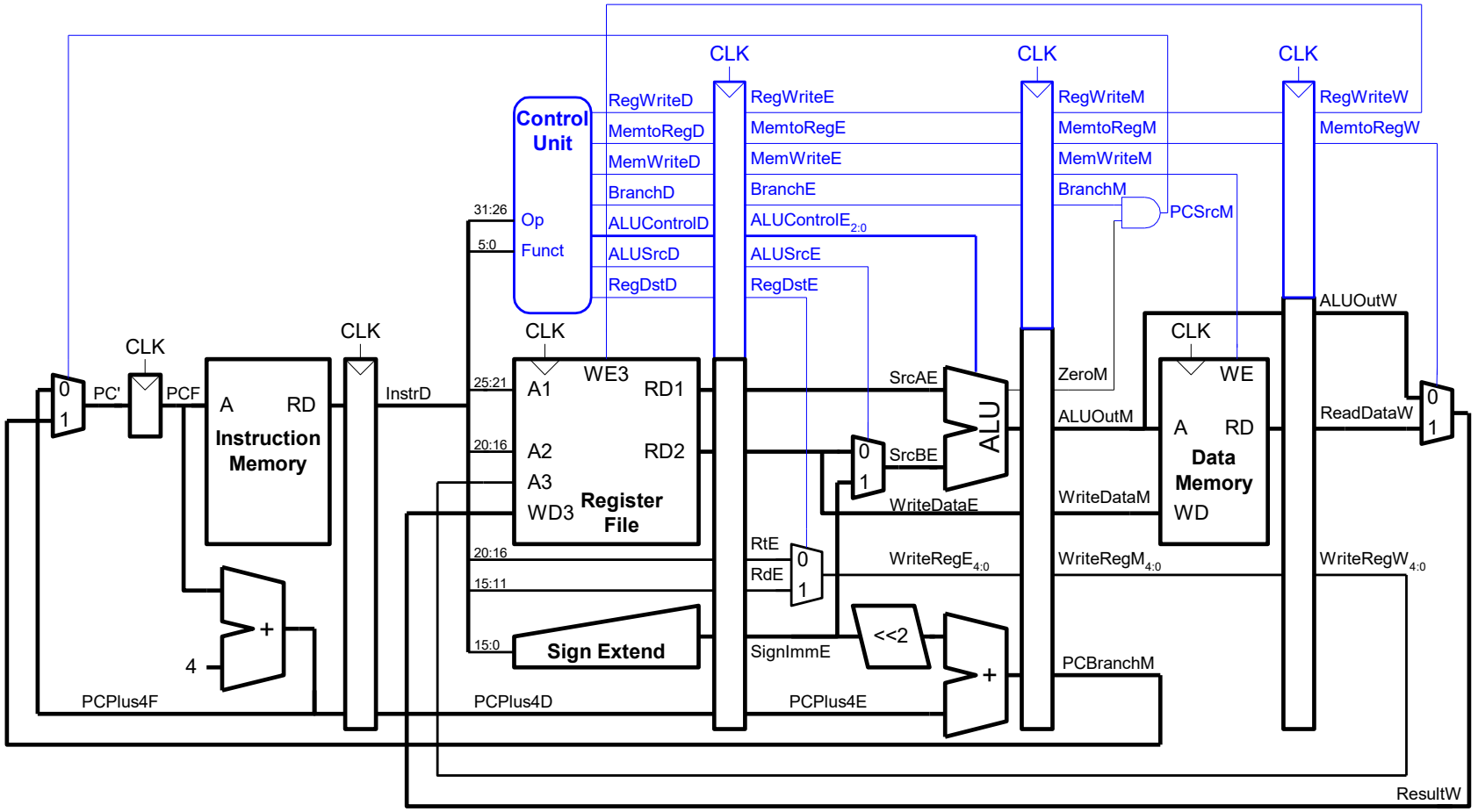
Corrected Pipelined Datapath



WriteReg* must arrive at same time as *Result

ELSEVIER

Pipelined Processor Control



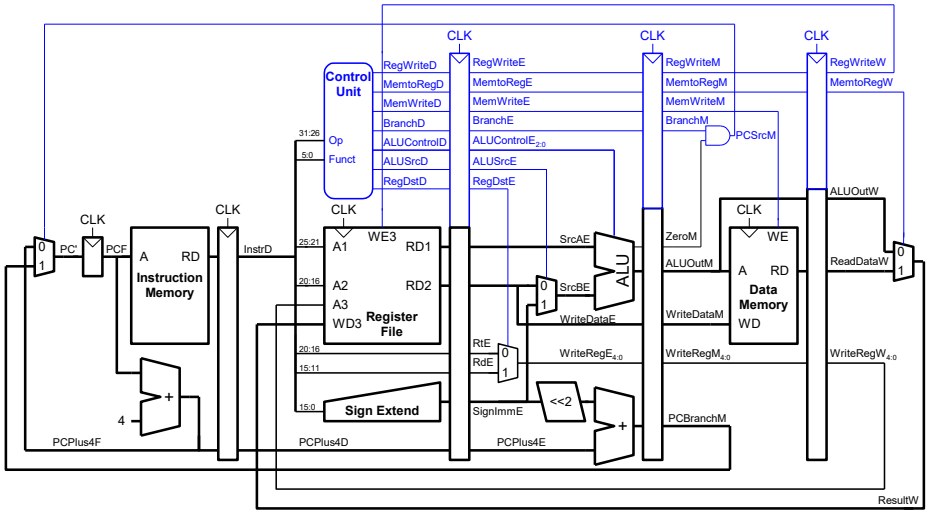
- **Same control unit as single-cycle processor**
- **Control delayed to proper pipeline stage**

ELSEVIER

Pipelined Performance

- Pipelined processor critical path:

$$T_c = \max \{ t_{pcq} + t_{mem} + t_{setup}$$
$$2(t_{RFread} + t_{mux} + t_{eq} + t_{AND} + t_{mux} + t_{setup})$$
$$t_{pcq} + t_{mux} + t_{mux} + t_{ALU} + t_{setup}$$
$$t_{pcq} + t_{memwrite} + t_{setup}$$
$$2(t_{pcq} + t_{mux} + t_{RFwrite}) \}$$



ELSEVIER

Pipelined 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_{RFread}	150
Register file setup	$t_{RFsetup}$	20
Equality comparator	t_{eq}	40
AND gate	t_{AND}	15
Memory write	$t_{memwrite}$	220
Register file write	$t_{RFwrite}$	100

$$T_c = 2(t_{\text{RFread}} + t_{\text{mux}} + t_{\text{eq}} + t_{\text{AND}} + t_{\text{mux}} + t_{\text{setup}})$$
$$= 2[150 + 25 + 40 + 15 + 25 + 20]\ \text{ps} = \mathbf{550\ ps}$$

ELSEVIER

Pipelined Performance Example

Program with 100 billion instructions

Execution Time = (# instructions) × CPI × T_c

$= (100 \times 10^9)(1.15)(550 \times 10^{-12})$

= 63 seconds

ELSEVIER

Advanced Architecture Techniques

- **Multithreading**
  - Wordprocessor: thread for typing, spell checking, printing
- **Multiprocessors**
  - Multiple processors (cores) on a single chip

Threading: Definitions

- **Process:** program running on a computer
  - Multiple processes can run at once: e.g., surfing Web, playing music, writing a paper
- **Thread:** part of a program
  - Each process has multiple threads: e.g., a word processor may have threads for typing, spell checking, printing

Threads in Conventional Processor

- One thread runs at once
- When one thread stalls (for example, waiting for memory):
  - Architectural state of that thread stored
  - Architectural state of waiting thread loaded into processor and it runs
  - Called **context switching**
- Appears to user like all threads running simultaneously

Multithreading

- Multiple copies of architectural state
- Multiple threads **active** at once:
  - When one thread stalls, another runs immediately
  - If one thread can't keep all execution units busy, another thread can use them
- Does not increase instruction-level parallelism (ILP) of single thread, but increases throughput

Intel calls this "hyperthreading"

ELSEVIER

Multiprocessors

- Multiple processors (cores) with a method of communication between them
- Types:
  - **Homogeneous**: multiple cores with shared memory
  - **Heterogeneous:** separate cores for different tasks (for example, DSP and CPU in cell phone)
  - **Clusters:** each core has own memory system

RISC-V

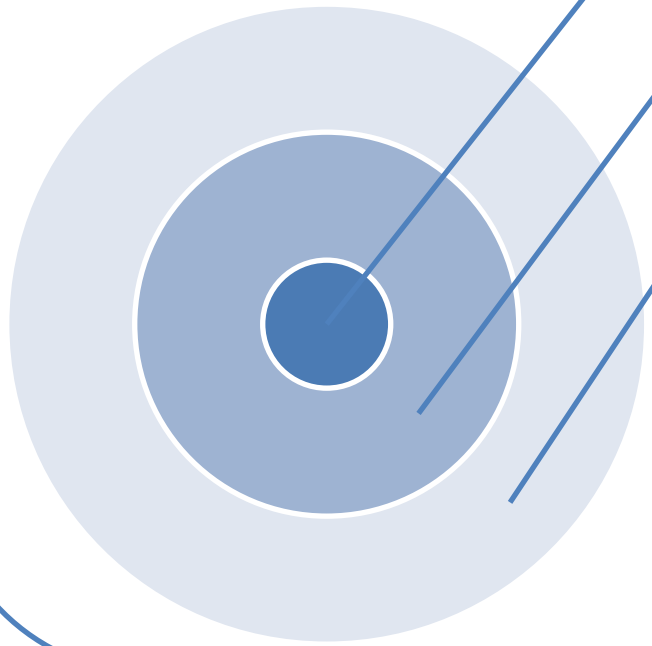
- RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration.
- Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation.

Other Resources

- Patterson & Hennessy's: *Computer Architecture: A Quantitative Approach*
- Conferences:
  - www.cs.wisc.edu/~arch/www/
  - ISCA (International Symposium on Computer Architecture)
  - HPCA (International Symposium on High Performance Computer Architecture)

ELSEVIER

Study: CS222: Computer Architecture



© *Digital Design and Computer Architecture*, 2nd Edition, 2012

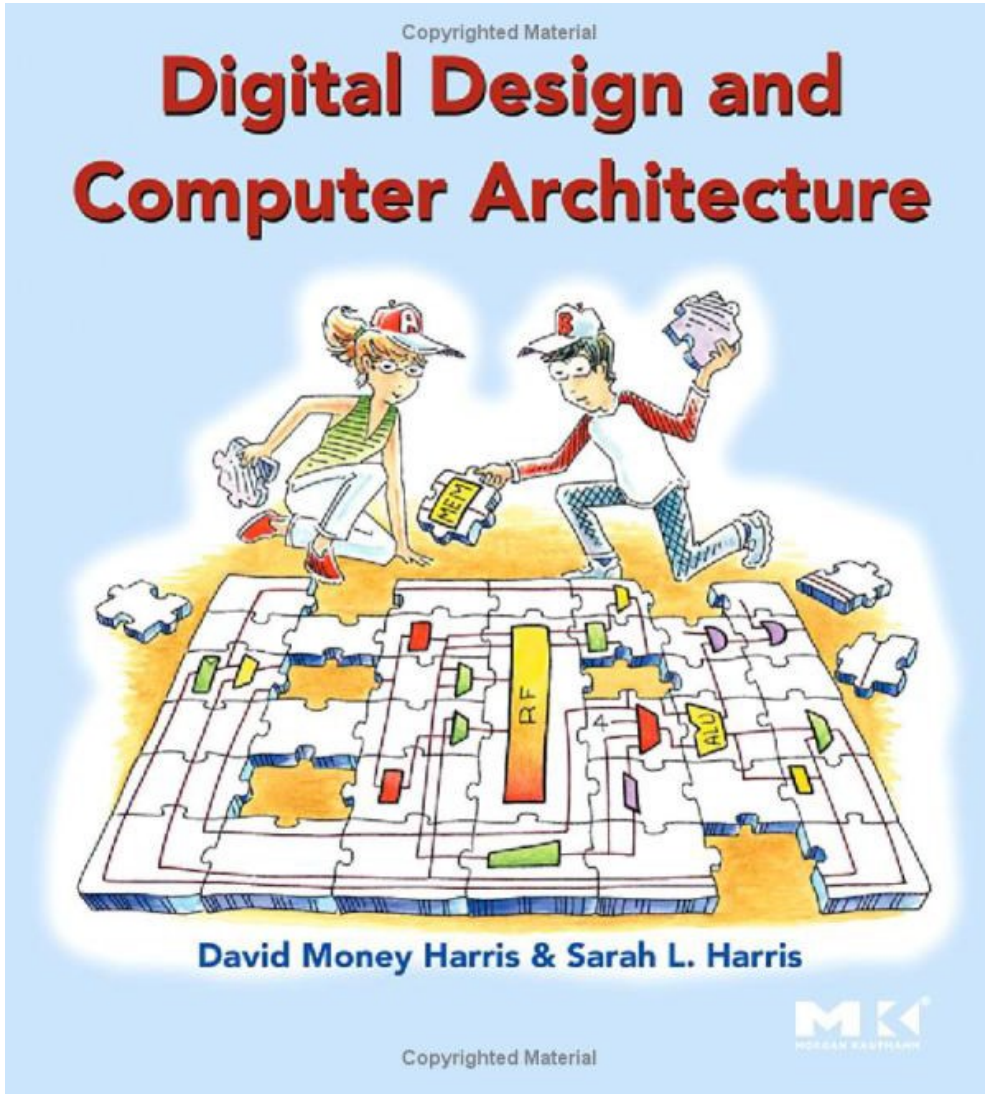
ELSEVIER

What ? Computer Architecture

- **computer architecture** defines how to command a processor.
- **computer architecture** is a set of rules and methods that describe the functionality, organization, and implementation of computer system.

ELSEVIER

How ? Course Book



ELSEVIER

How ? Course Content

Lec #	Subject	Week #
Lec1	Chapter 1: From Zero to One	Week #1
Lec2	Chapter 2: Combinational Logic Design	Week #2
Lec 3	Chapter 3: Sequential Logic Design	Week #3
Lec 4	Chapter 4: Hardware Description Language	Week #4
Lec 5	Chapter 4 : continue	Week #5
Lec 6	Chapter 5: Digital Building Blocks	Week #6
Lec 7	Chapter 5 continue	Week #7
	Midterm Exam	Week #8
Lec 8	Chapter 6: Computer Architecture	Week #9
Lec 9	Chapter 6 : continue	Week #10
Lec 10	Chapter 7: Microarchitecture	Week #11
Lec 11	Chapter 7 continue	Week #12
~~Lec 12~~	~~Chapter 7 continue~~	~~Week #13~~

© *Digital Design and Computer Architecture*, 2nd Edition, 2012

ELSEVIER

Assessment

Final-Term Exam **50**

Mid-Term Exam + lab Exam + Oral Exam +
Projects (Verilog – ModelSim + Quartus)

- **logic design Project in Verilog – the week after midterm (Lab)**
- **final project -> lab exam** **50**

Chapter 8 (Sections 1 to 3) is a self study - In Exam.

ELSEVIER

Exam Topics

- ~~Timing~~
- ~~Prefix and lookahead Adders~~
- ~~Multi Cycle Processor~~
- ~~Section 8.4, 8.5, 8.6~~

CS222: Computer Architecture

ما هو رأيك فيما يلى :

1. المحتوى العلمى للكورس موضحا العيوب من وجهة نظرك ، ثم اذكرنصائح لتحسين العيوب وتطوير الكورس للسنة القادمة.
2. المحاضر، طريقة التدريس ، العيوب بالمحاضرة ، ثم اذكر نصائح للتطوير.
3. العملى و الهيئة المعاونة ، العيوب ، نصائح للتطوير.

Course Evaluation form
https://docs.google.com/forms/d/e/1FAIpQLSc3B8pytOyfAJFcYP6_P1ugMoSBTEj5XK9osBjvFkLuhlPkVQ/viewform?usp=pp_url : feel free to say what you want.

© Digital Design and Computer Architecture, 2nd Edition, 2012

ELSEVIER