CMPE 411
Computer Architecture

Lecture 11

Multi-cycle Processor Design

October 5, 2017

www.csee.umbc.edu/~younis/CMPE411/CMPE411.htm

Lecture's Overview

❑ *Previous Lecture:*

- Processor design steps
  (ISA analysis, component selection, datapath assembly, control unit)
- Building a datapath
  (Instruction fetch, register transfer requirements)
- Control unit design
  (Steps of control design, register transfer logic)
- Single cycle processor
  (Advantage and disadvantage, integration of datapath and control)
- Circuit implementation of control unit
  (Logic equations, truth tables, combinational circuit)

❑ *This Lecture*

➔ Multi-cycle datapath

➔ Multi-cycle control

Overview of Processor Design

Design Steps:

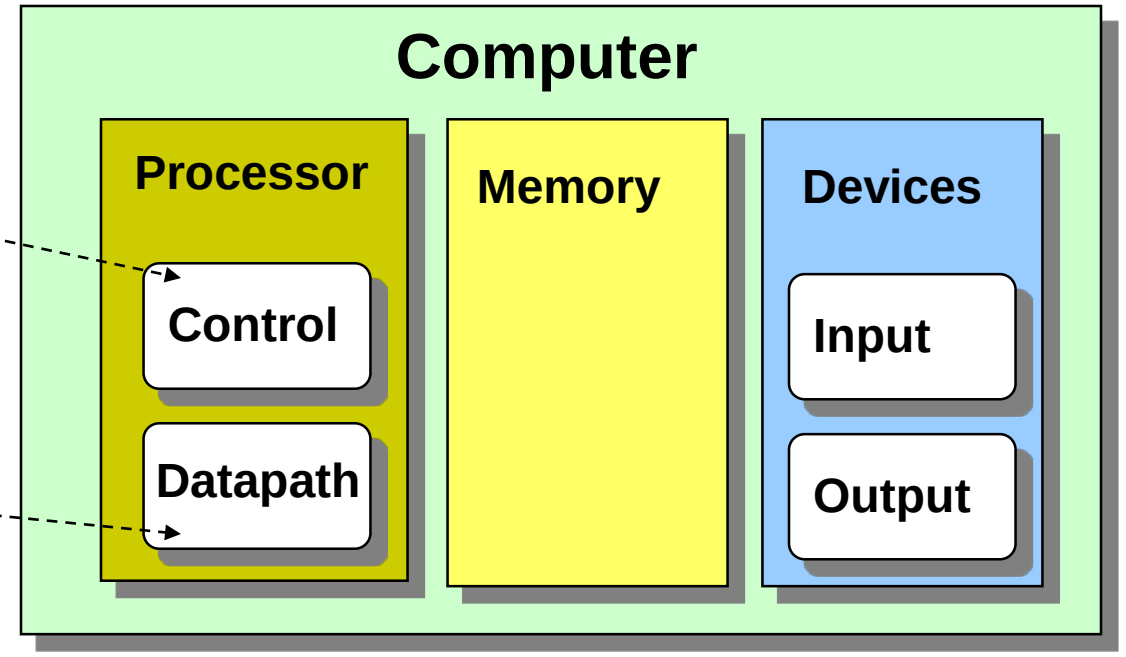
1. Analyze instruction set => datapath requirements
2. Select set of datapath components and establish clocking methodology
3. Assemble datapath meeting the requirements
4. Analyze implementation of each instruction to determine setting of control points that affects the register transfer
5. Assemble the control logic

✓ Applied for single-cycle processor

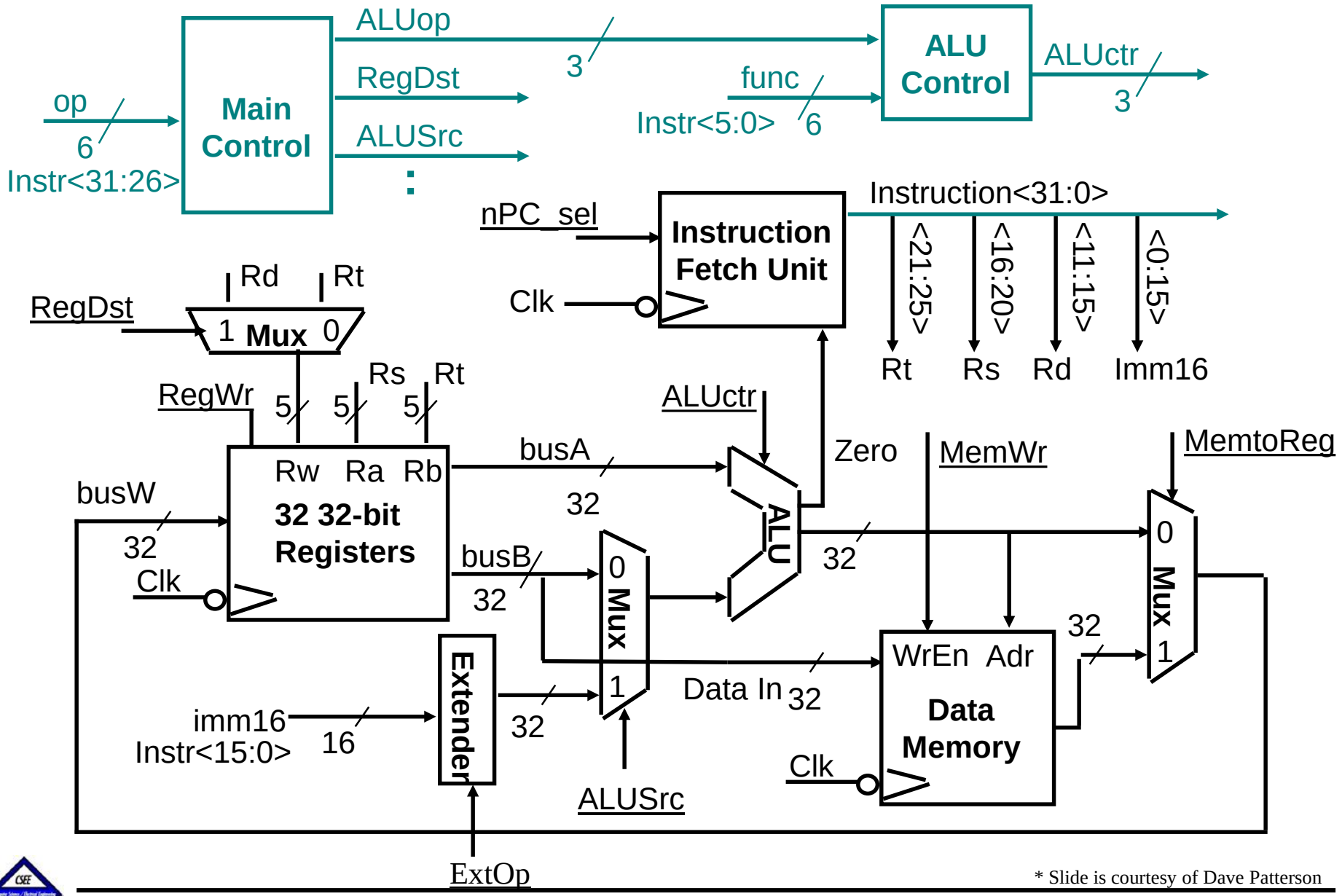
In today's class we design a multi-cycle processor

Coordination for proper operation

Connections for Information flow

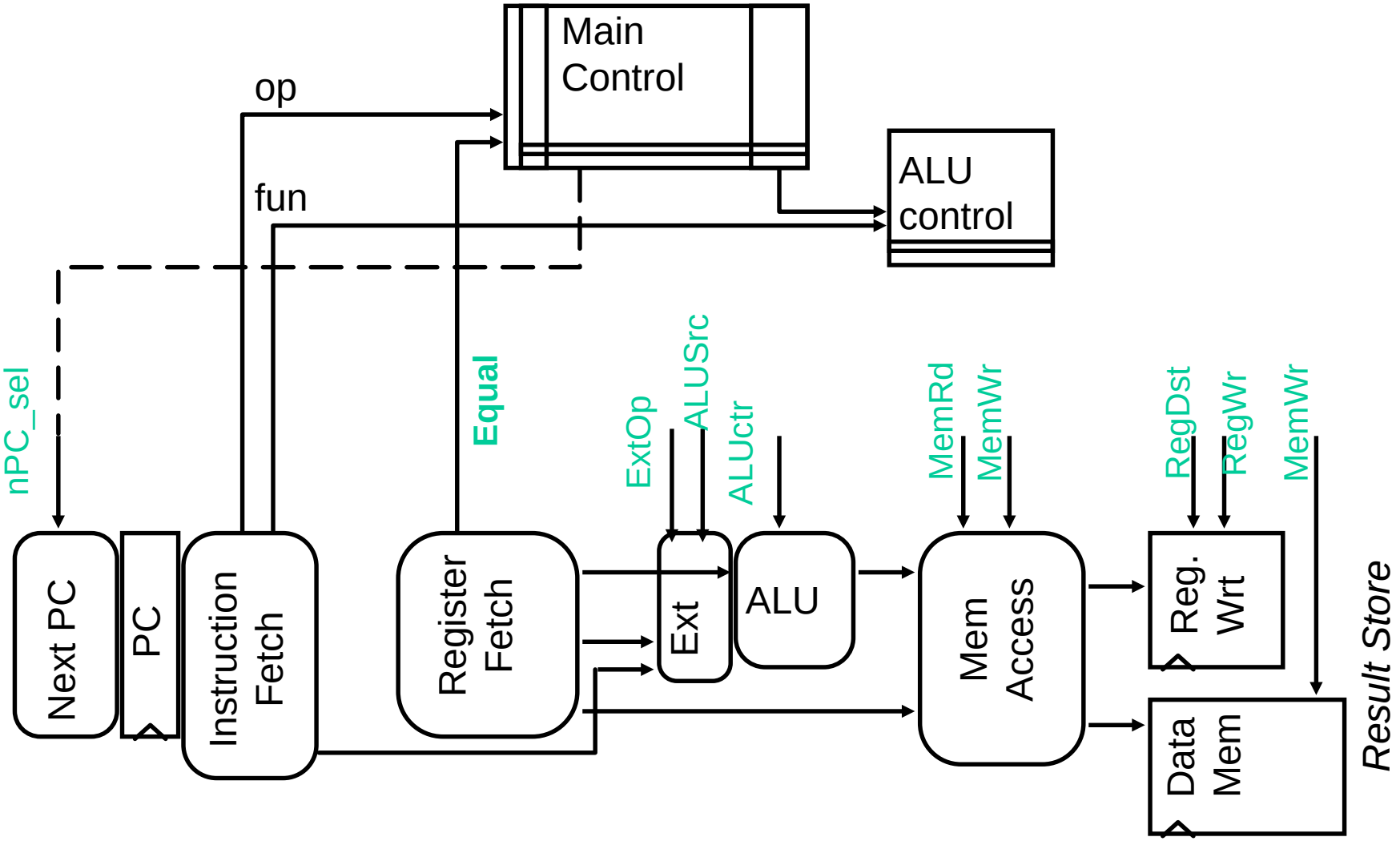


A Single Cycle Processor



* Slide is courtesy of Dave Patterson

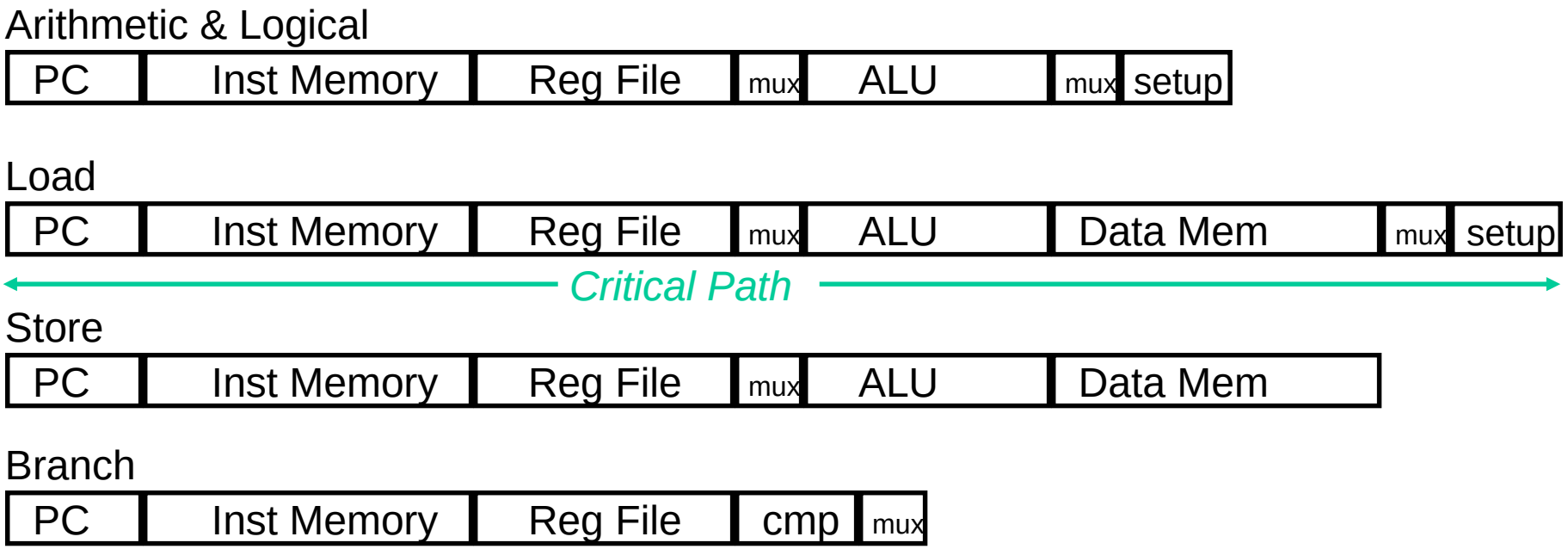
Abstract View of single cycle processor



➔ looks like a FSM with PC as a state

* Slide is courtesy of Dave Patterson

What's wrong with our CPI=1 processor?

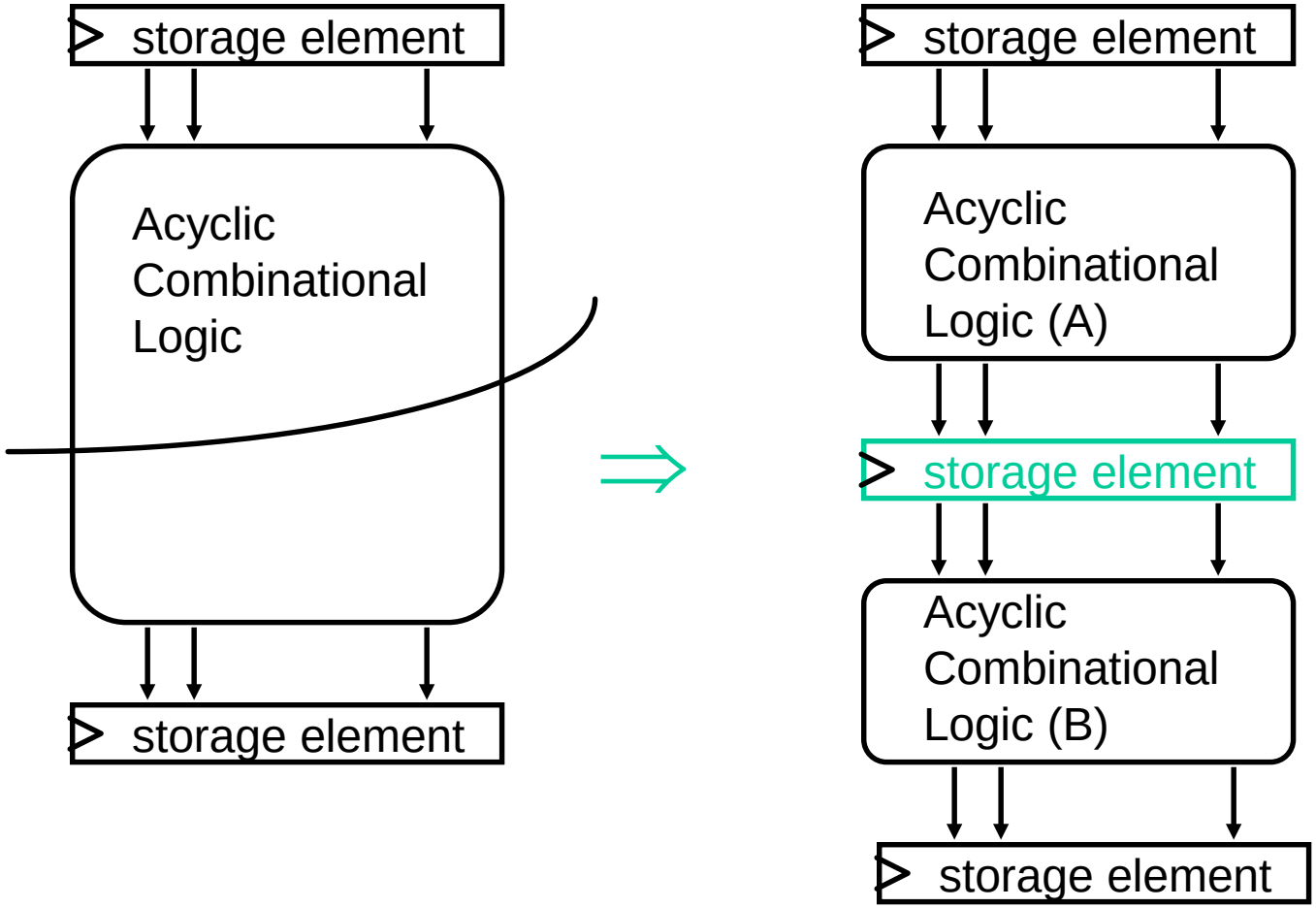


- Long Cycle Time
- All instructions take as much time as the slowest
- Real memory is not so nice as our idealized memory
  - ➔ cannot always get the job done in one (short) cycle

* Slide is courtesy of Dave Patterson

Reducing Cycle Time

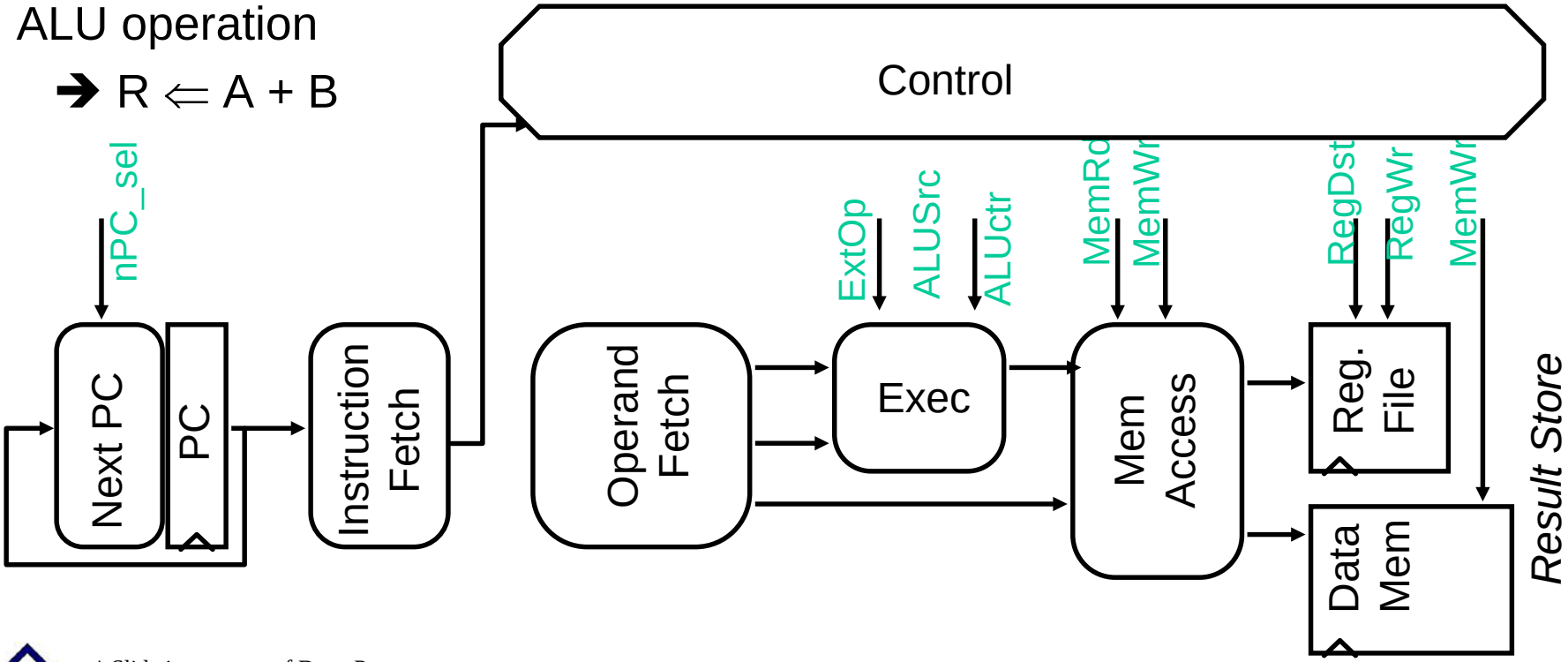
- Cut combinational dependency graph and insert register / latch
- Do same work in two fast cycles, rather than one slow one



* Slide is courtesy of Dave Patterson

Basic Limits on Cycle Time

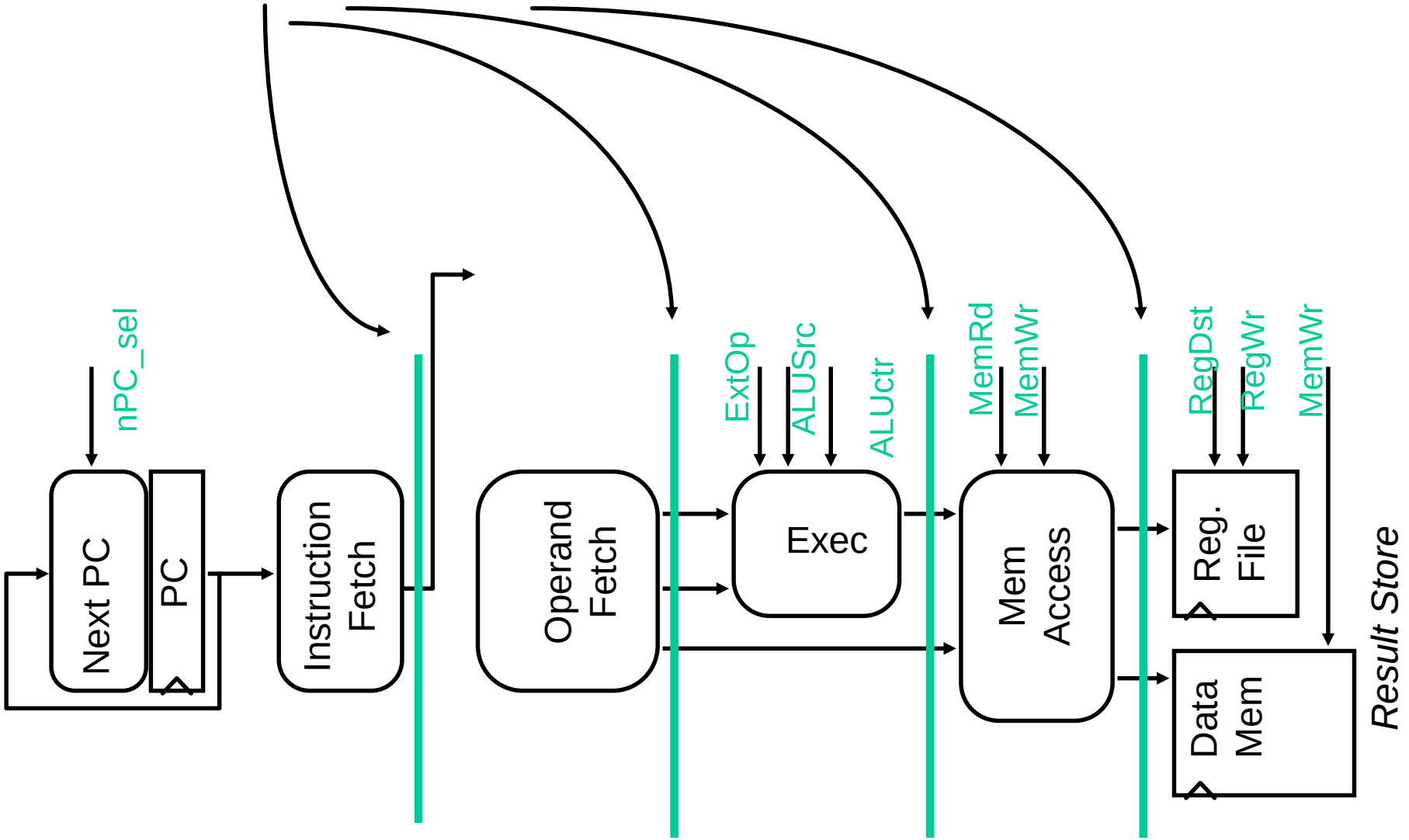
- Next address logic
  - ➔ PC ⇐ branch ? PC + offset : PC + 4
- Instruction Fetch
  - ➔ InstructionReg ⇐ Mem[PC]
- Register Access
  - ➔ A ⇐ R[rs]
- ALU operation
  - ➔ R ⇐ A + B



* Slide is courtesy of Dave Patterson

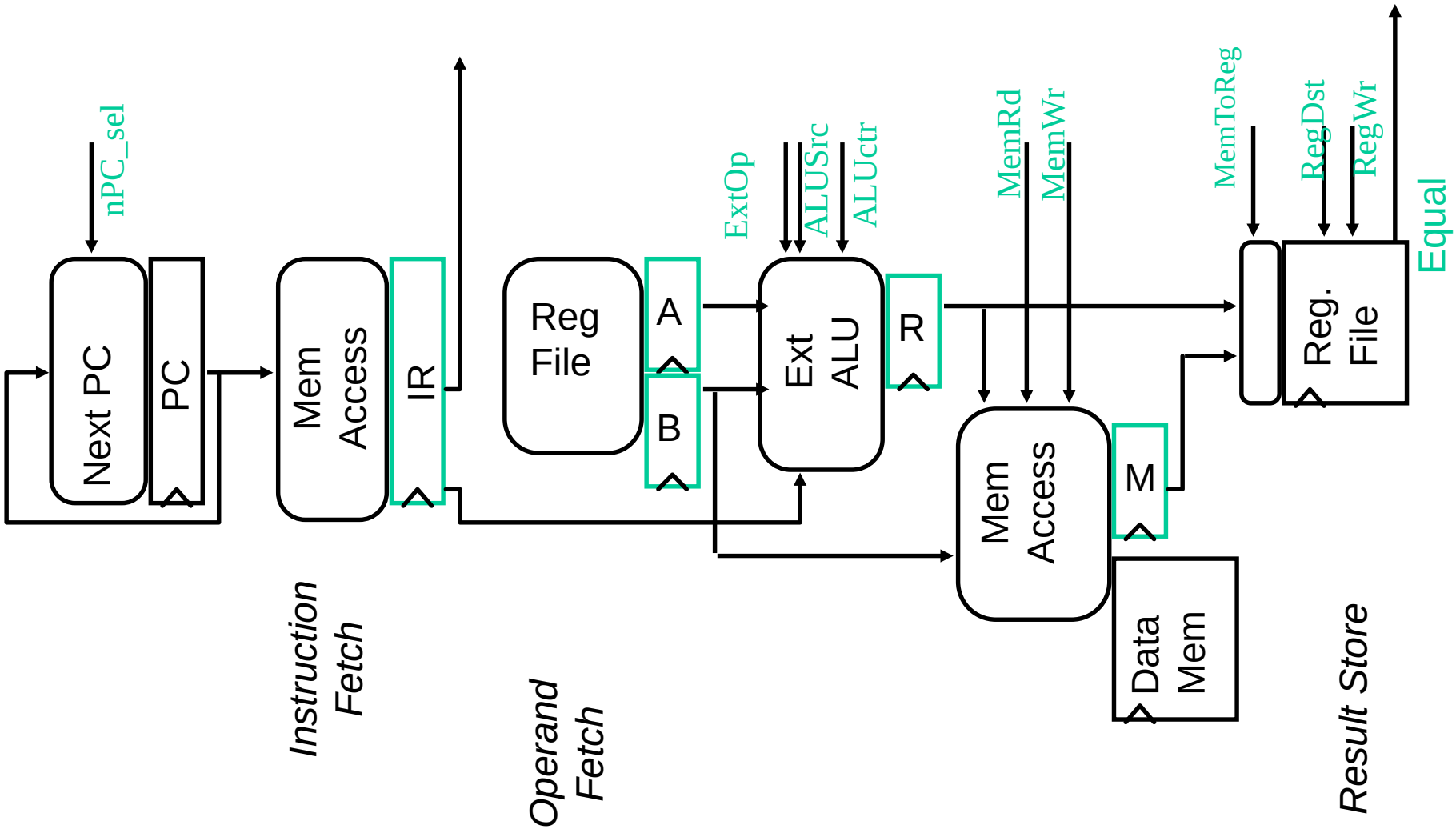
Partitioning the CPI=1 Datapath

Add registers between smallest steps



* Slide is courtesy of Dave Patterson

Example Multi-cycle Datapath



* Slide is courtesy of Dave Patterson

Recall: Step-by-step Processor Design

Step 1: Analyze instruction set => datapath requirements

(ISA => Logical Register Transfers)

Step 2: Select set of datapath components and establish clocking

Step 3: Assemble datapath meeting the requirements

(RTL + Components => Datapath)

Step 4: Analyze implementation of each instruction to determine setting of control points that effects the register transfer

(Datapath + Logical RTs => Physical RTs)

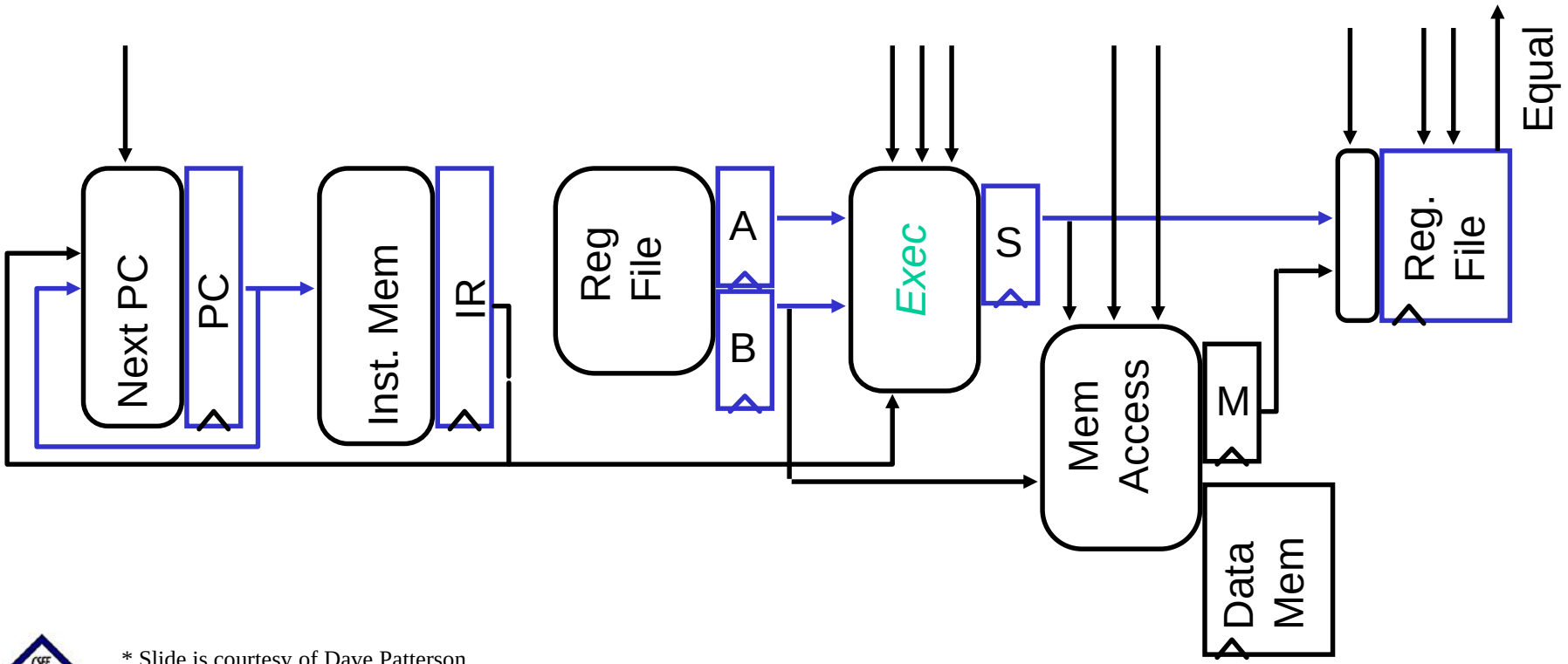
Step 5: Assemble the control logic

(Physical RTs => Control)

Step 4: R-type (add, sub, . . .)

inst	Logical Register Transfers
ADDU	R[rd] ⇐ R[rs] + R[rt]; PC ⇐ PC + 4

inst	Physical Register Transfers	
	IR ⇐ MEM[pc]	
ADDU	A⇐ R[rs]; B ⇐ R[rt]	
	S ⇐ A + B	
	R[rd] ⇐ S;	PC ⇐ PC + 4

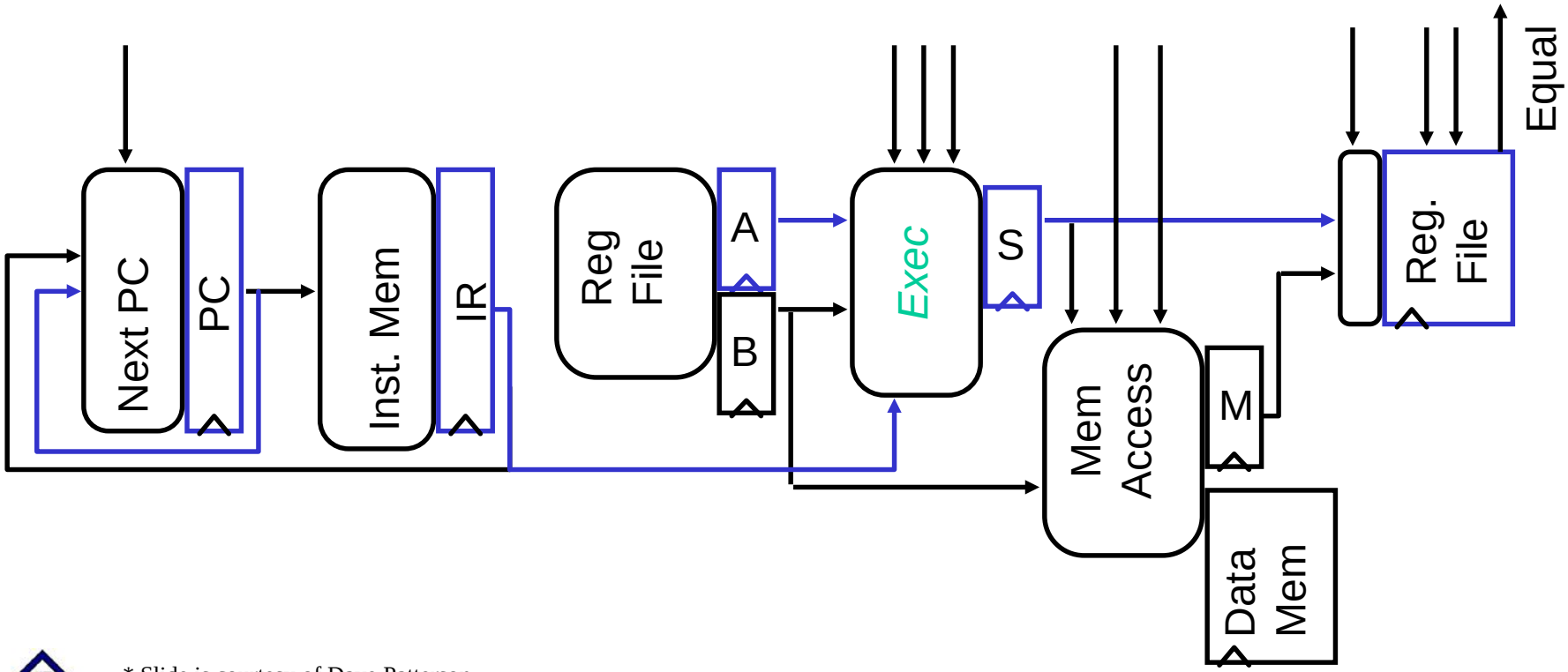


* Slide is courtesy of Dave Patterson

Step 4: Logical immediate

inst	Logical Register Transfers
ORI	R[rt] ⇐ R[rs] OR zx(Im16);
	PC ⇐ PC + 4

inst	Physical Register Transfers	
	IR ⇐ MEM[pc]	
ORI	A ⇐ R[rs]; B ⇐ R[rt]	
	S ⇐ A *or* ZeroExt(Im16)	
	R[rt] ⇐ S;	PC ⇐ PC + 4

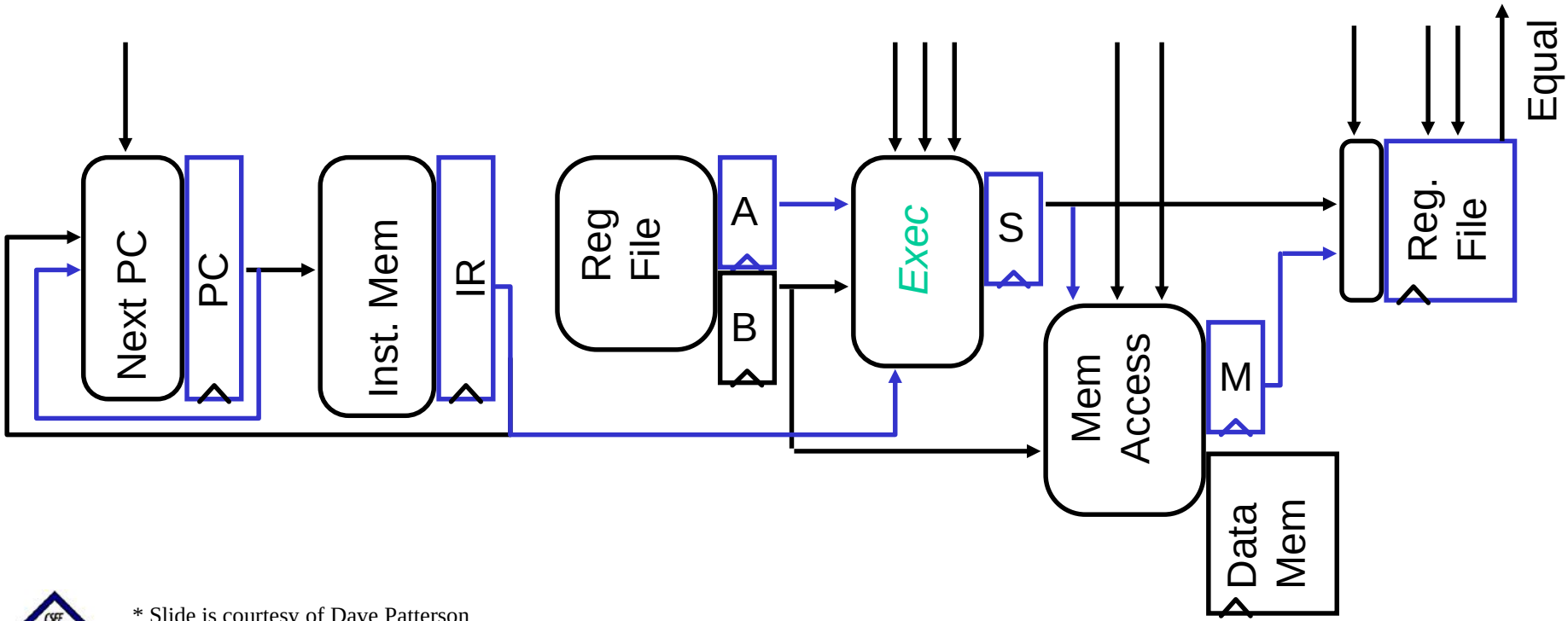


* Slide is courtesy of Dave Patterson

Step 4 : Load

inst	Logical Register Transfers
LW	R[rt] ⇐ MEM(R[rs] + sx(Im16); PC ⇐ PC + 4

inst	Physical Register Transfers
	IR ⇐ MEM[pc]
LW	A ⇐ R[rs]; B ⇐ R[rt]
	S ⇐ A + SignEx(Im16)
	M ⇐ MEM[S]
	R[rd] ⇐ M; PC ⇐ PC + 4

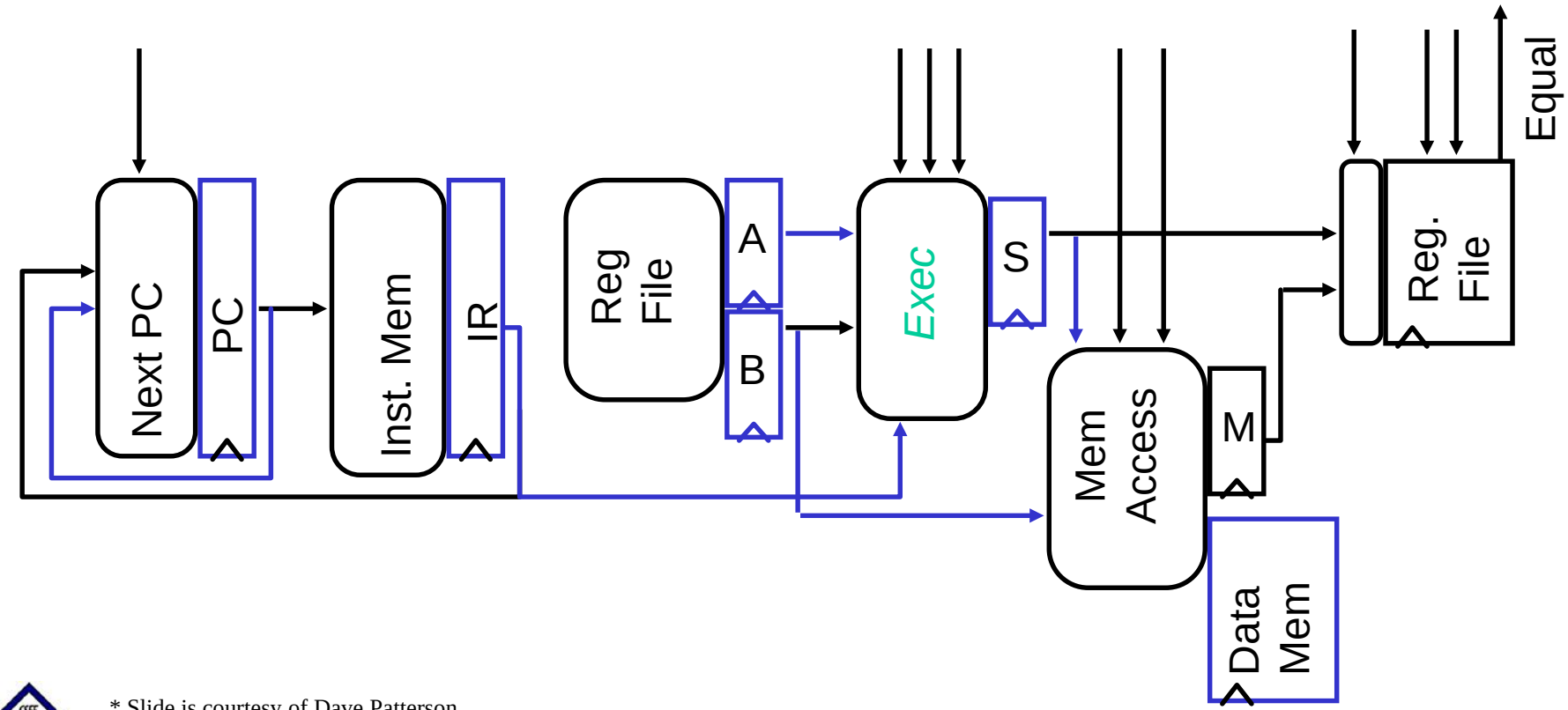


* Slide is courtesy of Dave Patterson

Step 4 : Store

inst	Logical Register Transfers
SW	MEM(R[rs] + sx(Im16) ⇐ R[rt];
	PC ⇐ PC + 4

inst	Physical Register Transfers
	IR ⇐ MEM[pc]
SW	A ⇐ R[rs]; B ⇐ R[rt]
	S ⇐ A + SignEx(Im16);
	MEM[S] ⇐ B PC ⇐ PC + 4



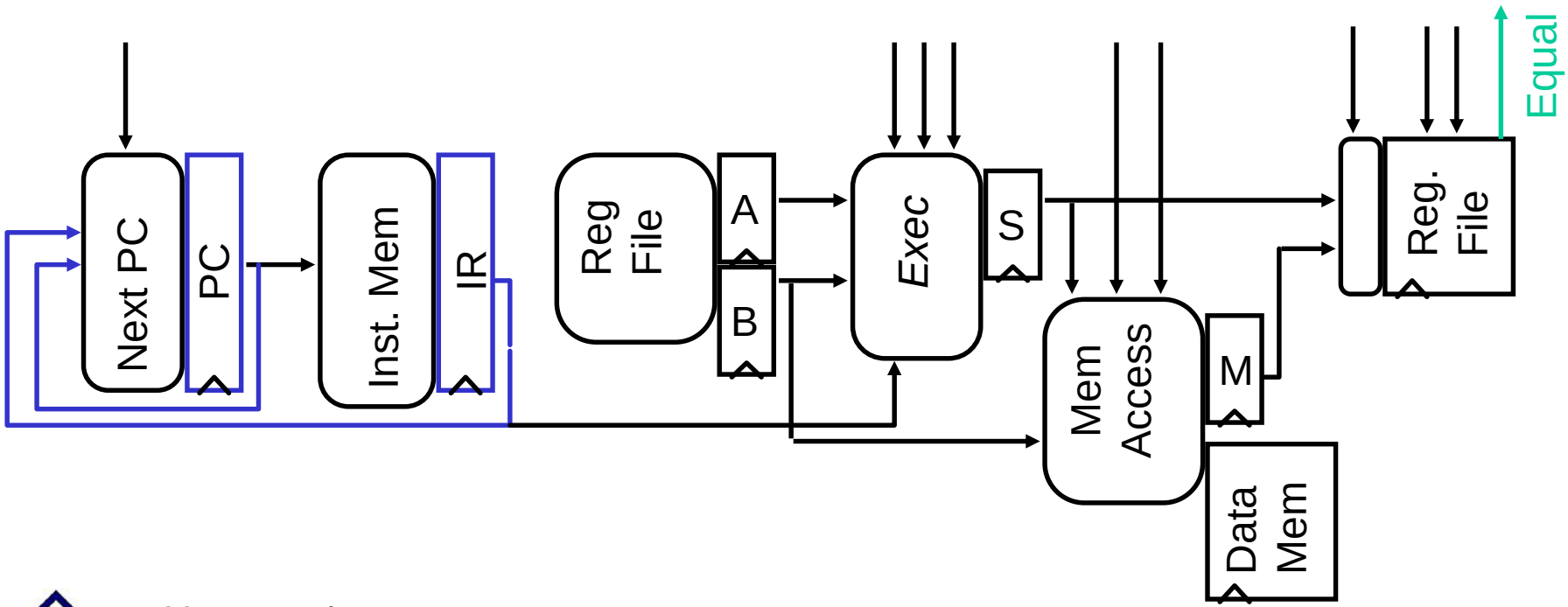
* Slide is courtesy of Dave Patterson

Step 4 : Branch

inst	Logical Register Transfers
BEQ	if R[rs] == R[rt]
	then PC ⇐ PC + sx(Im16) || 00
	else PC ⇐ PC + 4

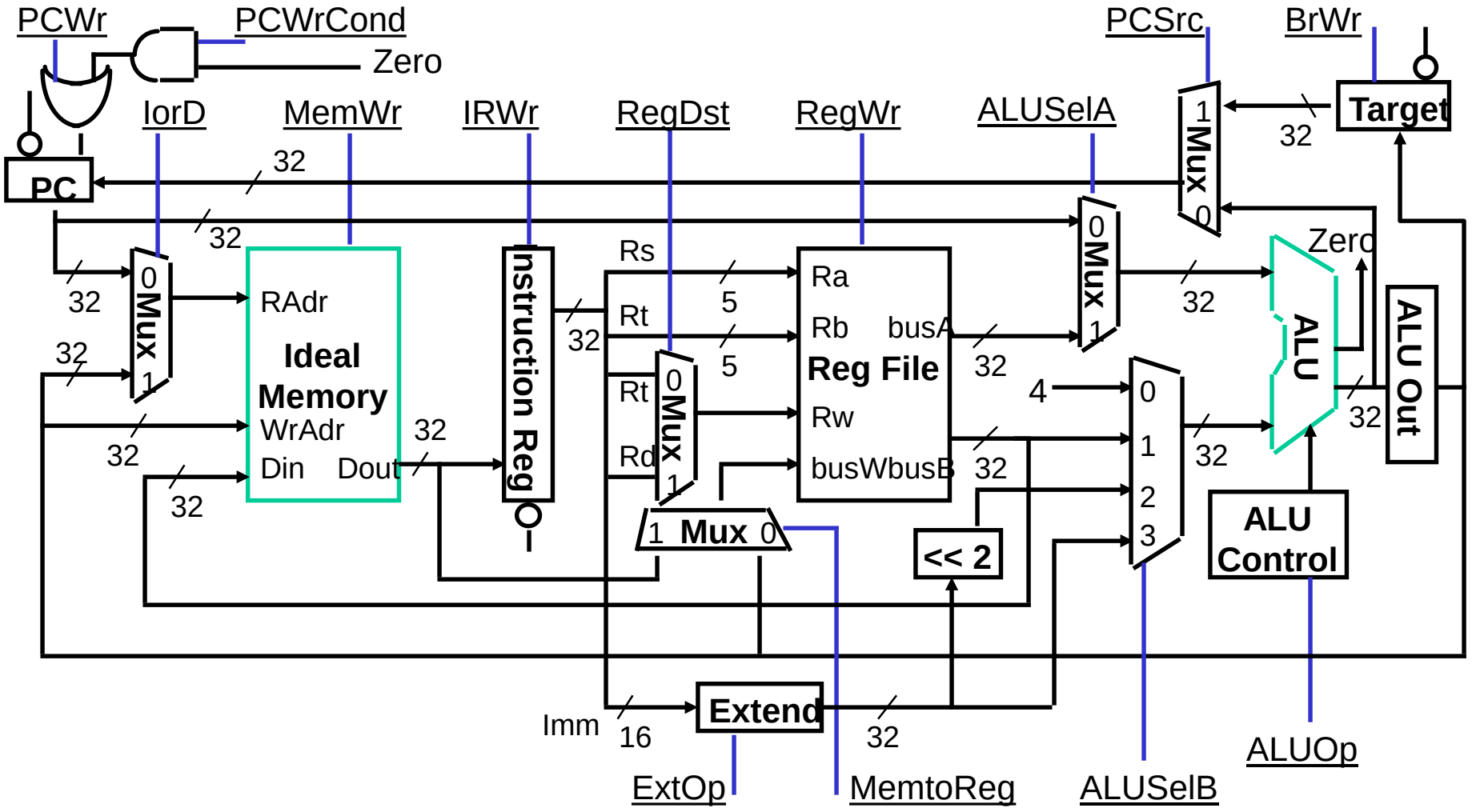
inst	Physical Register Transfers
	IR ⇐ MEM[pc]
BEQ|Eq	PC ⇐ PC + sx(Im16) || 00

inst	Physical Register Transfers
	IR ⇐ MEM[pc]
BEQ|Eq	PC ⇐ PC + 4



* Slide is courtesy of Dave Patterson

Multiple Cycle Datapath

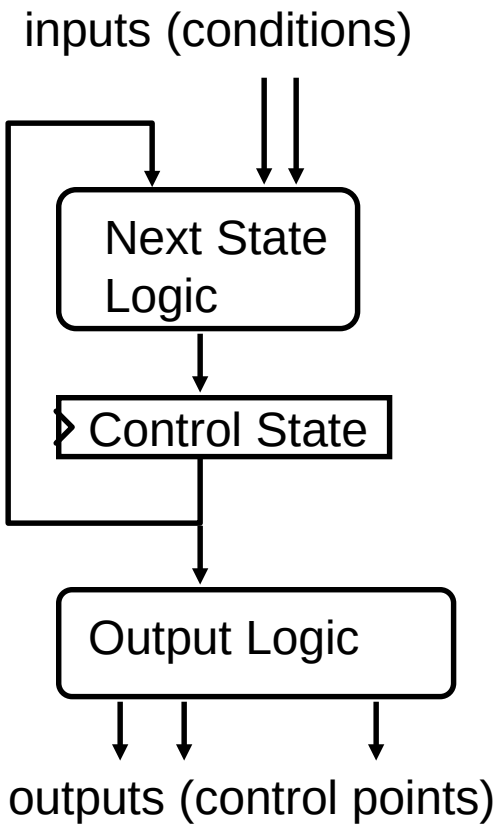


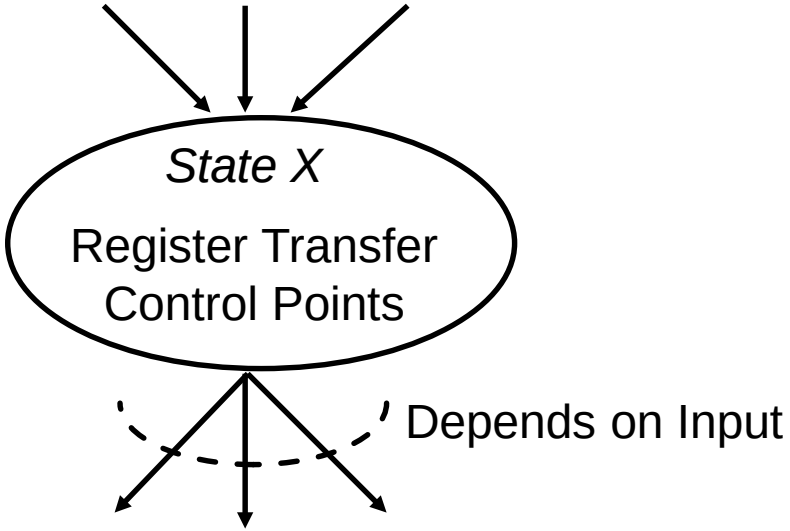
Minimizes Hardware: 1 memory, 1 adder

* Slide is courtesy of Dave Patterson

Control Model

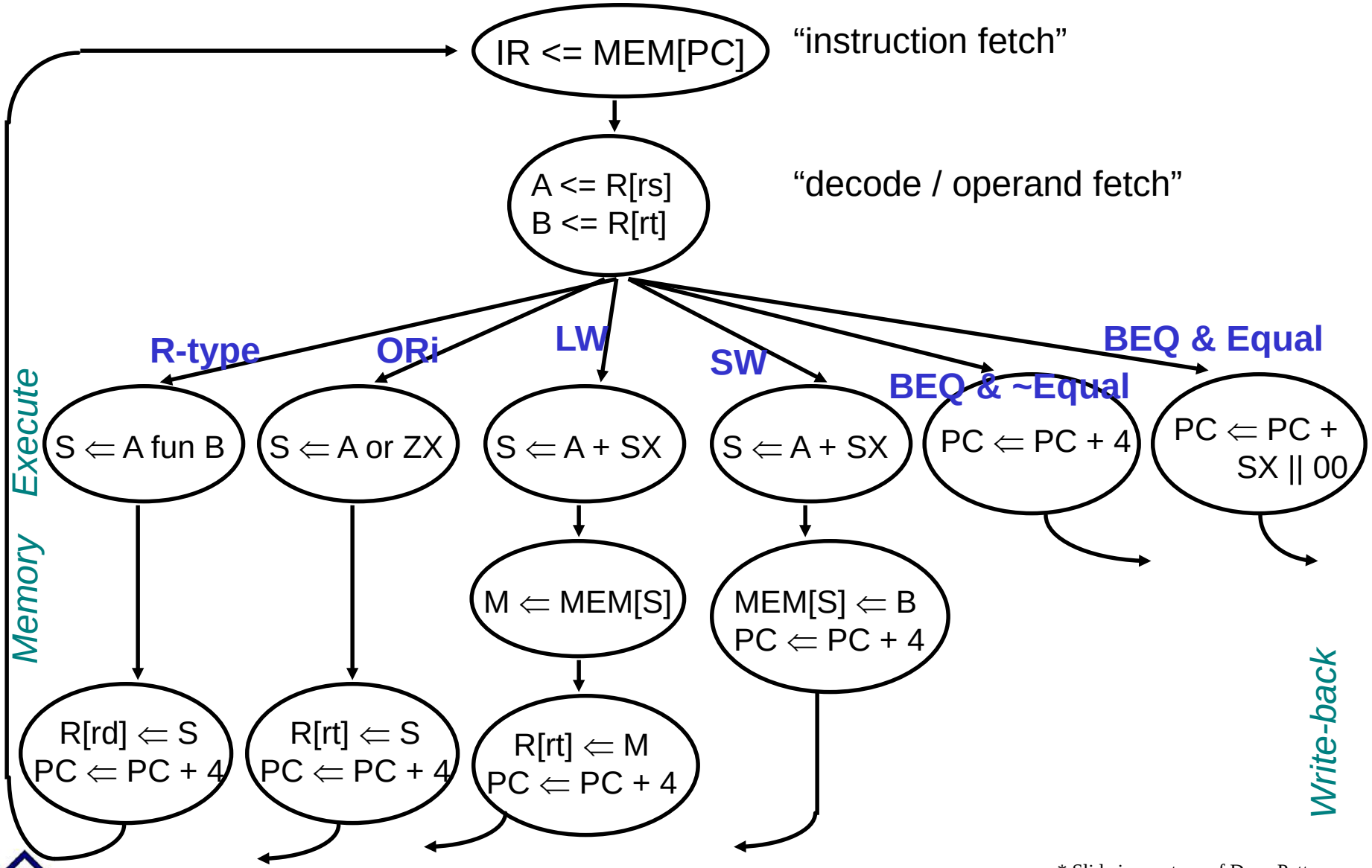
- State specifies control points for Register Transfer
- Transfer occurs upon exiting state (same falling edge)





* Slide is courtesy of Dave Patterson

Control Spec. for multi-cycle processor



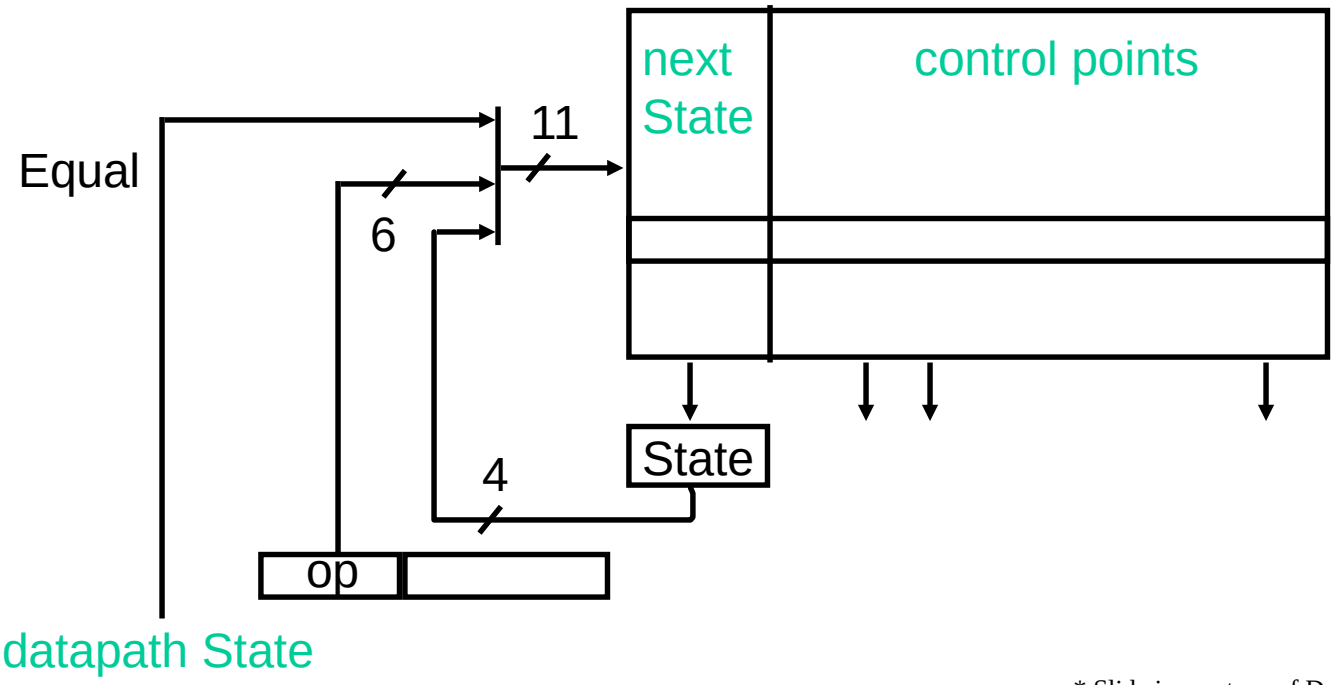
* Slide is courtesy of Dave Patterson

Step 5: Assemble Control

state	op	cond	next state	control points

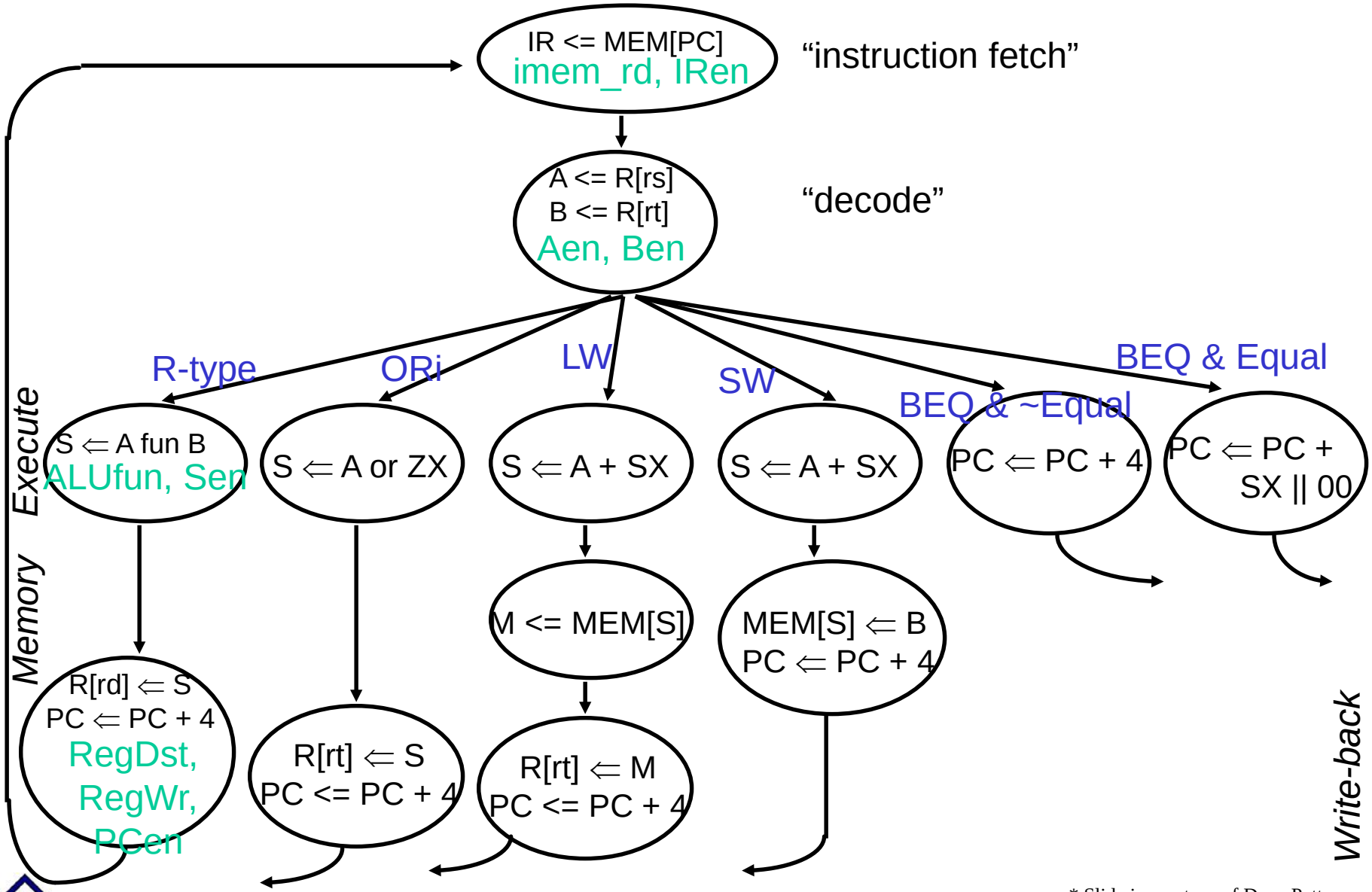
Truth Table

- Translate physical register transfers into control points
- Assign states
- Then go build the controller



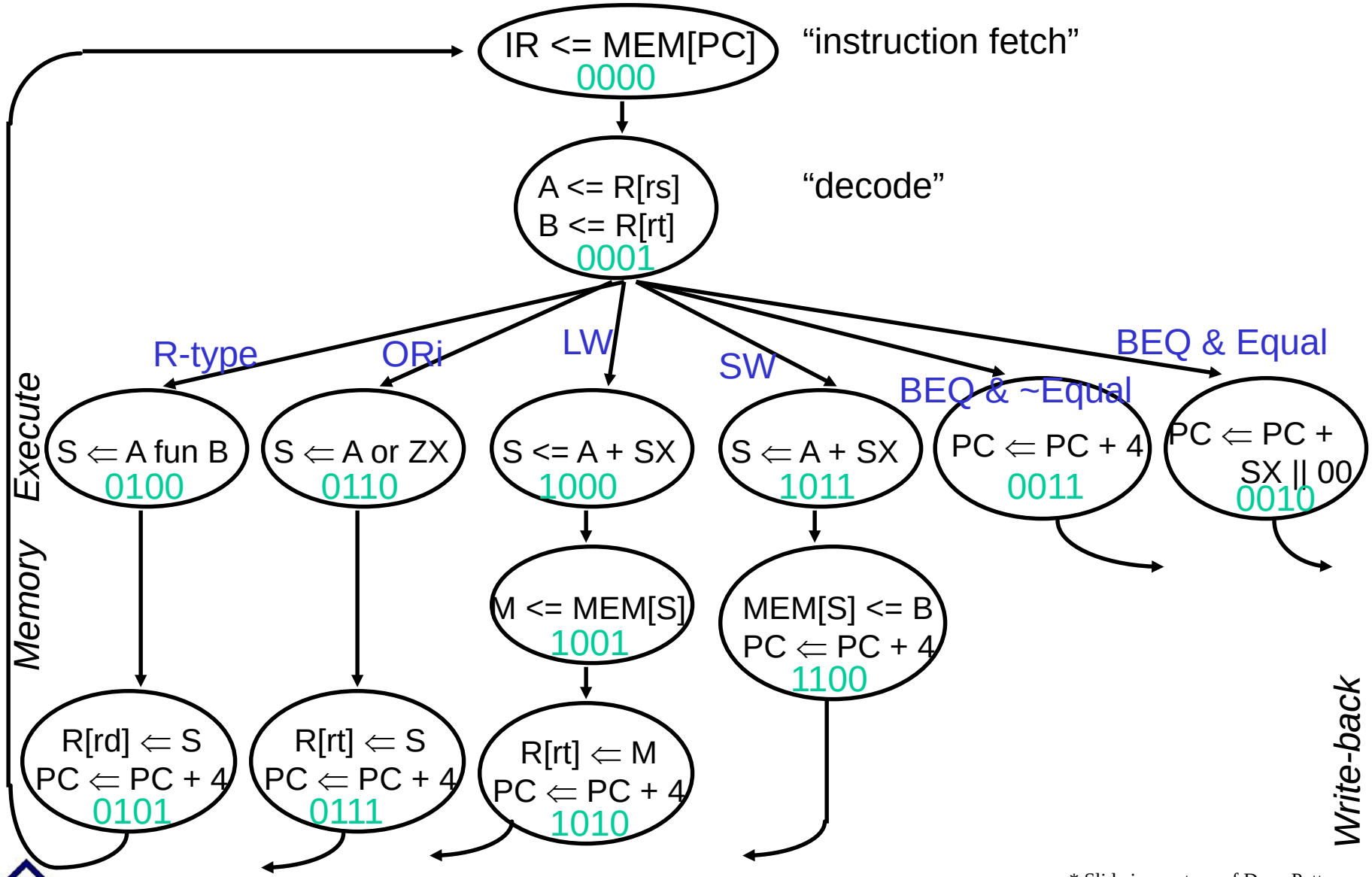
* Slide is courtesy of Dave Patterson

Mapping RTs to Control Points



* Slide is courtesy of Dave Patterson

Assigning States



* Slide is courtesy of Dave Patterson

Detailed Control Specification

	State	Op field	Eq	Next	IR	PC en	PC sel	Ops A	Ops B	Exec Ex	Exec Sr	Exec ALU	Exec S	Mem R	Mem W	Mem M	Write-Back M-R	Write-Back Wr	Write-Back Dst
	0000	??????	?	0001	1														
	0001	BEQ	0	0011				1	1										
	0001	BEQ	1	0010				1	1										
	0001	R-type	x	0100				1	1										
	0001	orI	x	0110				1	1										
	0001	LW	x	1000				1	1										
	0001	SW	x	1011				1	1										
	0010	xxxxxx	x	0000		1	1												
	0011	xxxxxx	x	0000		1	0												
R:	0100	xxxxxx	x	0101						0	1	fun	1						
	0101	xxxxxx	x	0000		1	0										0	1	1
ORi:	0110	xxxxxx	x	0111						0	0	or	1						
	0111	xxxxxx	x	0000		1	0										0	1	0
LW:	1000	xxxxxx	x	1001						1	0	add	1						
	1001	xxxxxx	x	1010										1	0	0			
	1010	xxxxxx	x	0000		1	0										1	1	0
SW:	1011	xxxxxx	x	1100						1	0	add	1						
	1100	xxxxxx	x	0000		1	0							0	1				

-all same in Moore machine

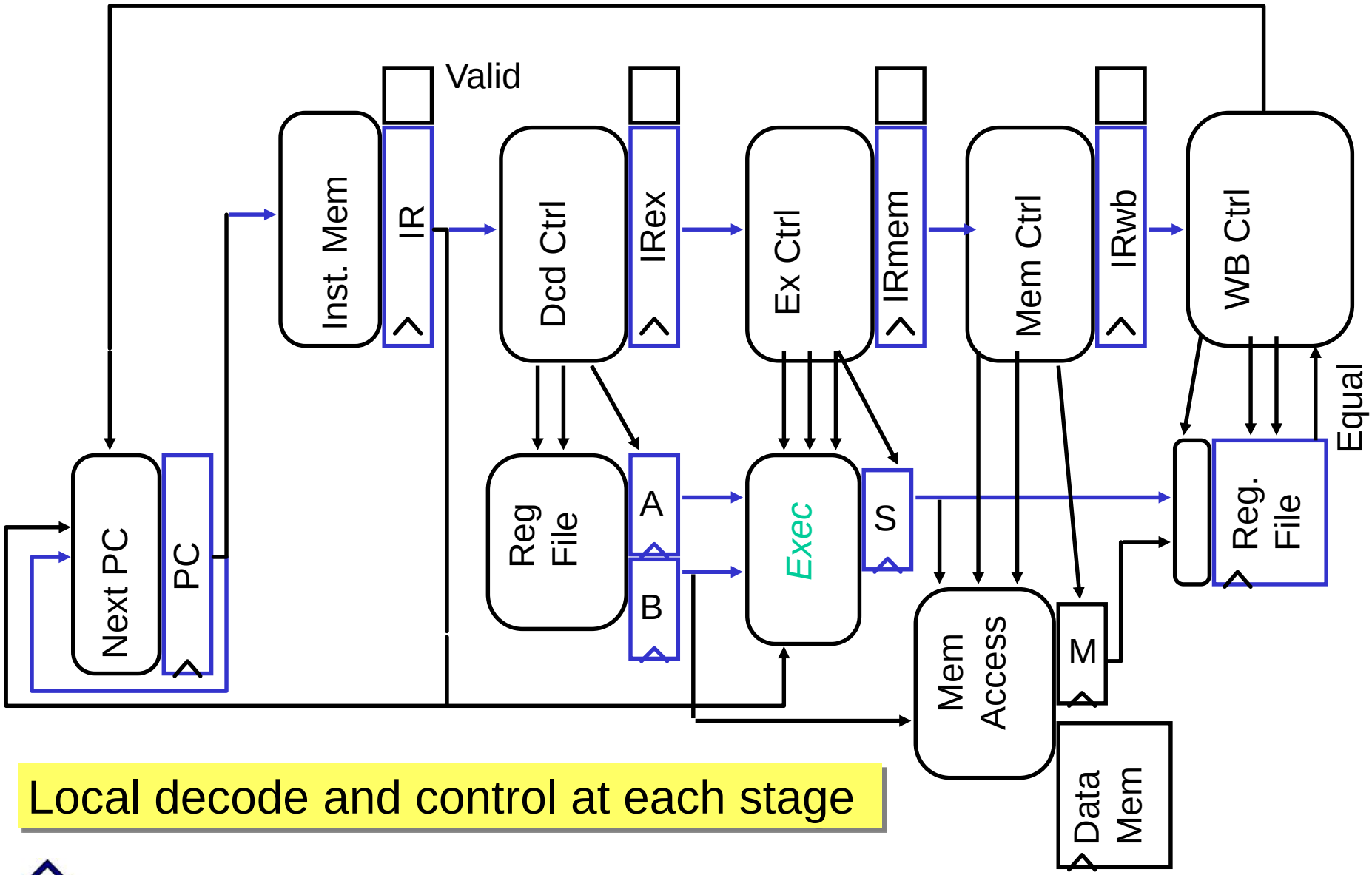
Performance Evaluation

- What is the average CPI?
  - ➔ state diagram gives CPI for each instruction type
  - ➔ workload gives frequency of each type

Type	CPI_i for type	Frequency	CPI_i x $freqI_i$
Arith/Logic	4	40%	1.6
Load	5	30%	1.5
Store	4	10%	0.4
branch	3	20%	0.6
		Average CPI:	4.1

* Slide is courtesy of Dave Patterson

Time-state Control Path



Local decode and control at each stage

Conclusion

❑ *Summary*

➔ Disadvantages of the Single Cycle Processor

- Long cycle time
- Cycle time is too long for all instructions except the Load

➔ Multiple Cycle Processor:

- Divide the instructions into smaller steps
- Execute each step (instead of the entire instruction) in one cycle

➔ Control is specified by finite state diagram

➔ Follow same 5-step method for designing “real” processor

❑ *Next Lecture*

➔ Micro-programmed control

➔ Processor exceptions

Read section 4.5 in 5th Ed., or 4.5 in the 4th Ed. of the textbook