Principles of Computer Architecture

CSE 240A

Fall 2024

Hadi Esmaeilzadeh

hadi@ucsd.edu

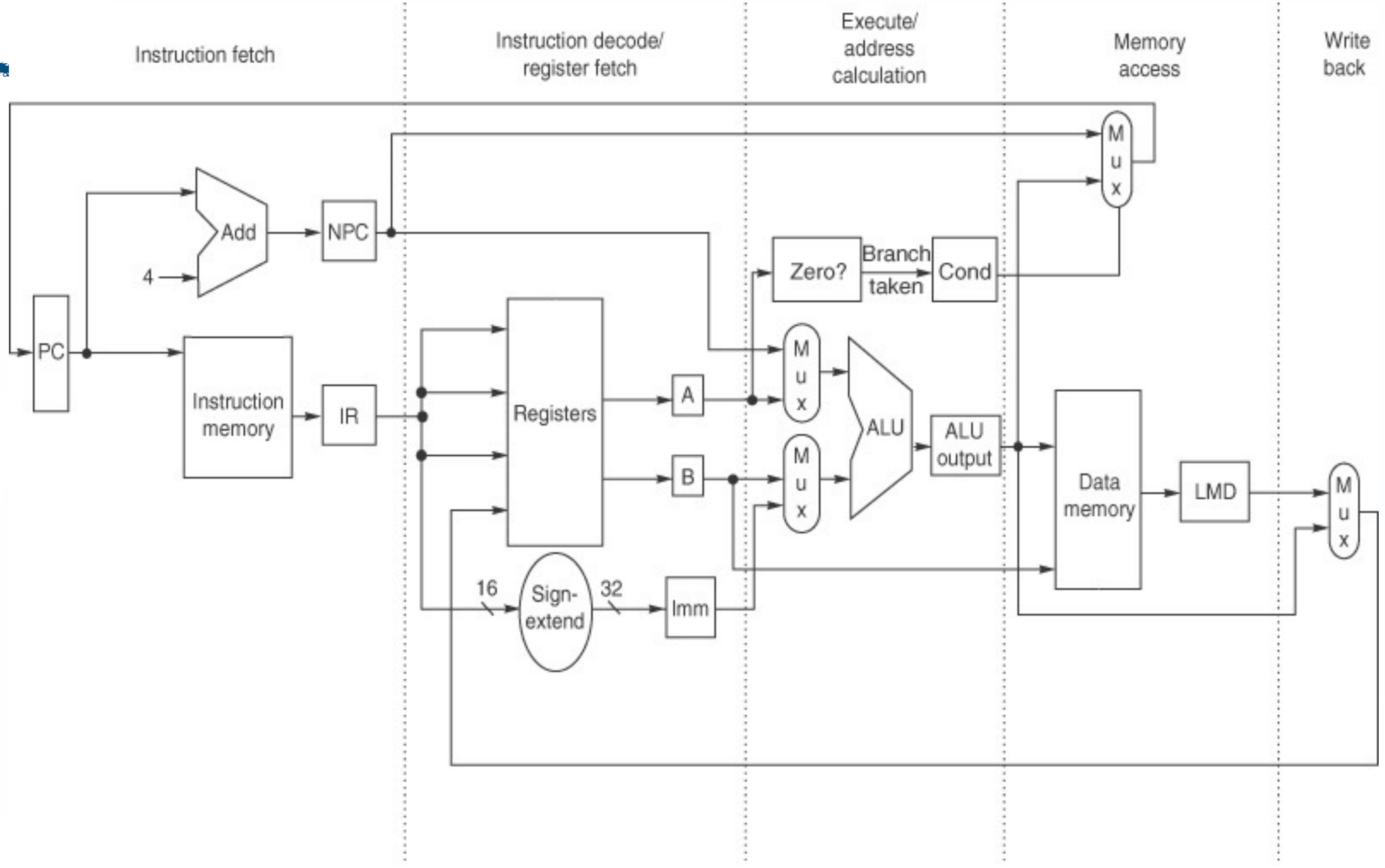
University of California, San Diego

Modern Times
Charlie Chaplin
Modern Times © Roy Export SAS
JONES-BURKE
TOOL CO.

Pipelining (some general principles)

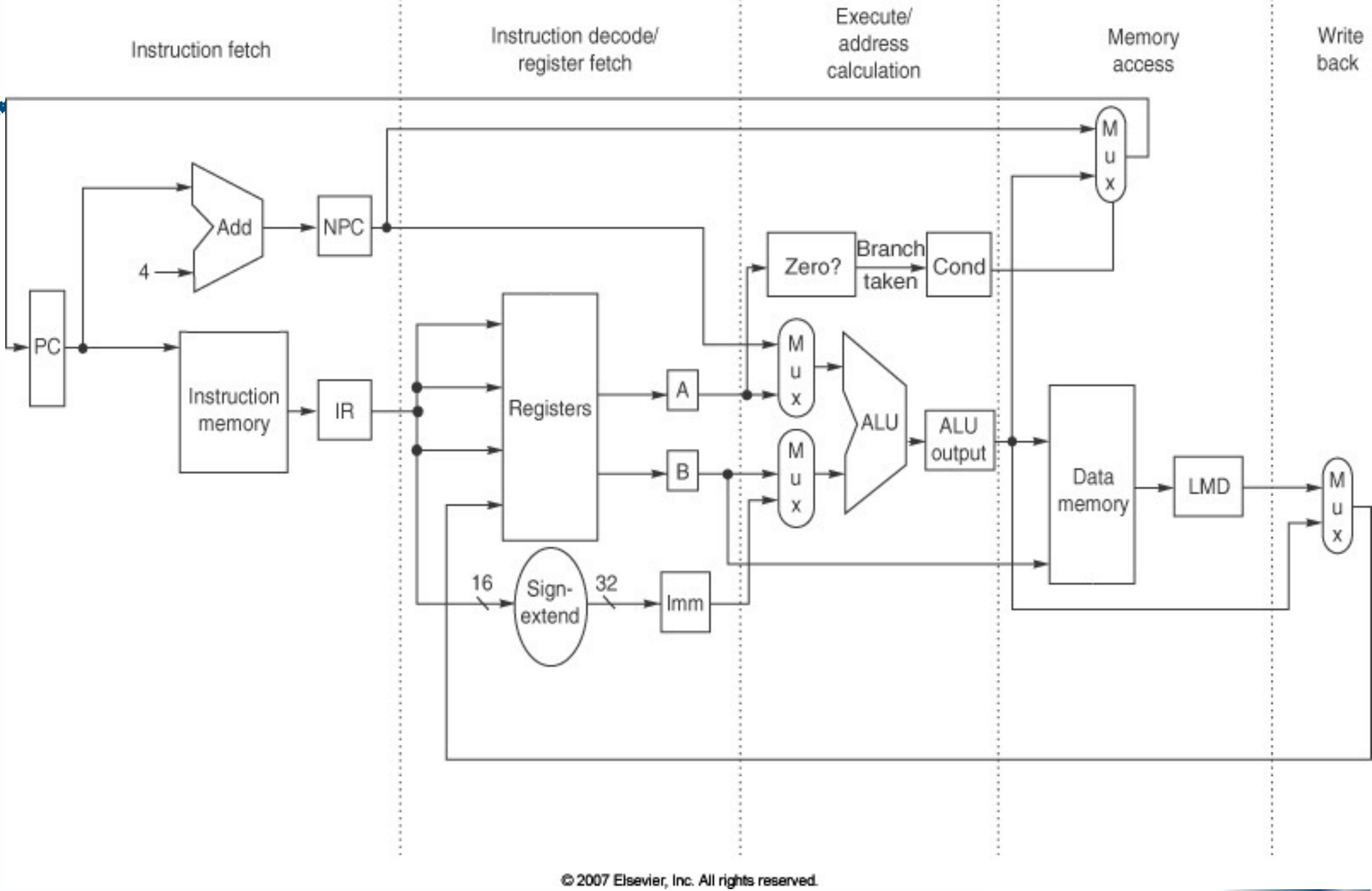
- Requires separable jobs/stages
- Requires separate resources
- Achieves parallelism without replication
- Pipeline efficiency (keeping the pipeline full) critical to performance

5 Steps of the MIPS Datapath



Notice only supports beqz

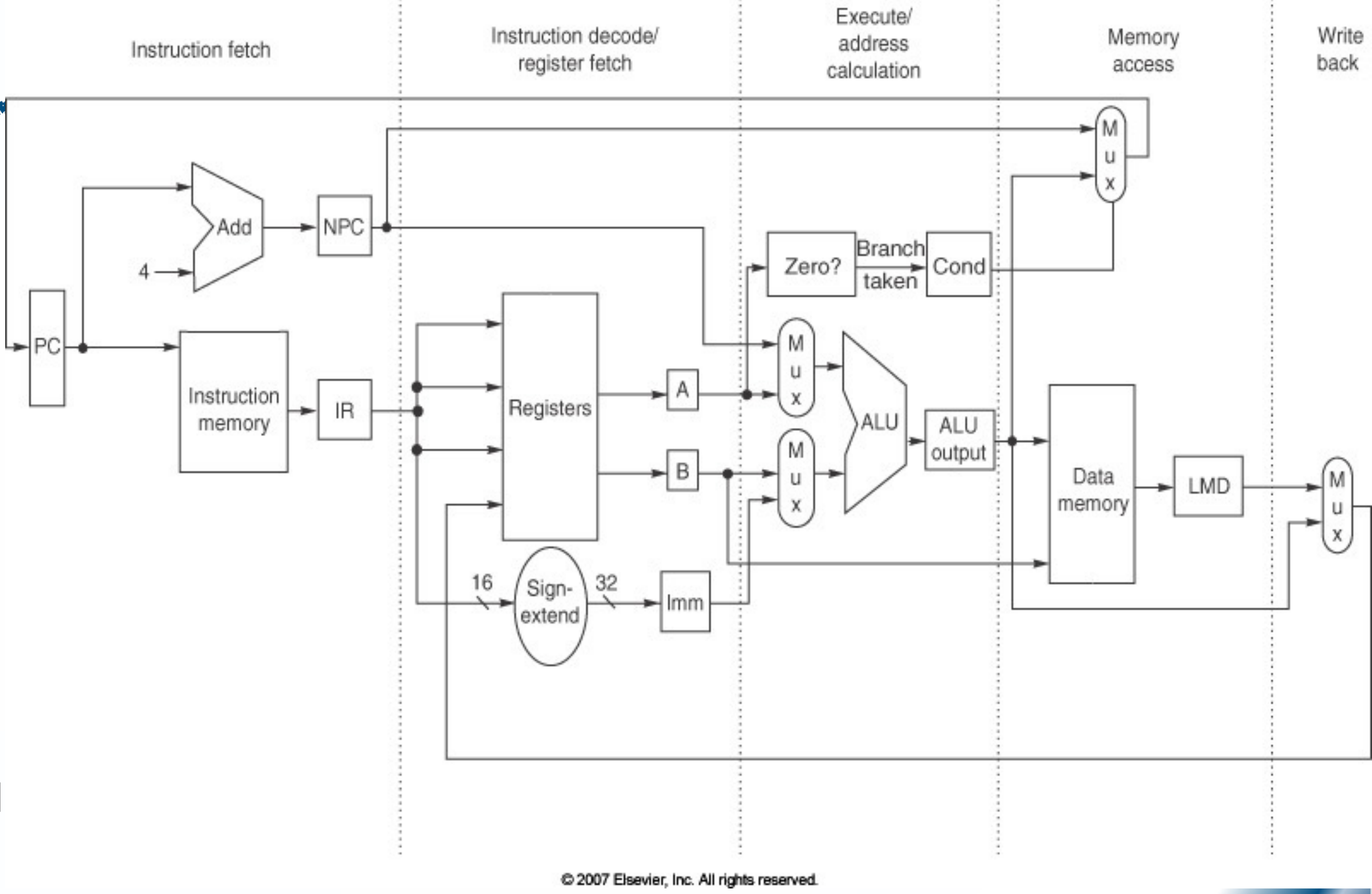
5 Steps of the MIPS Datapath



Instruction Fetch (**IF**)

- IR <- M[PC]
- NPC <- PC + 4

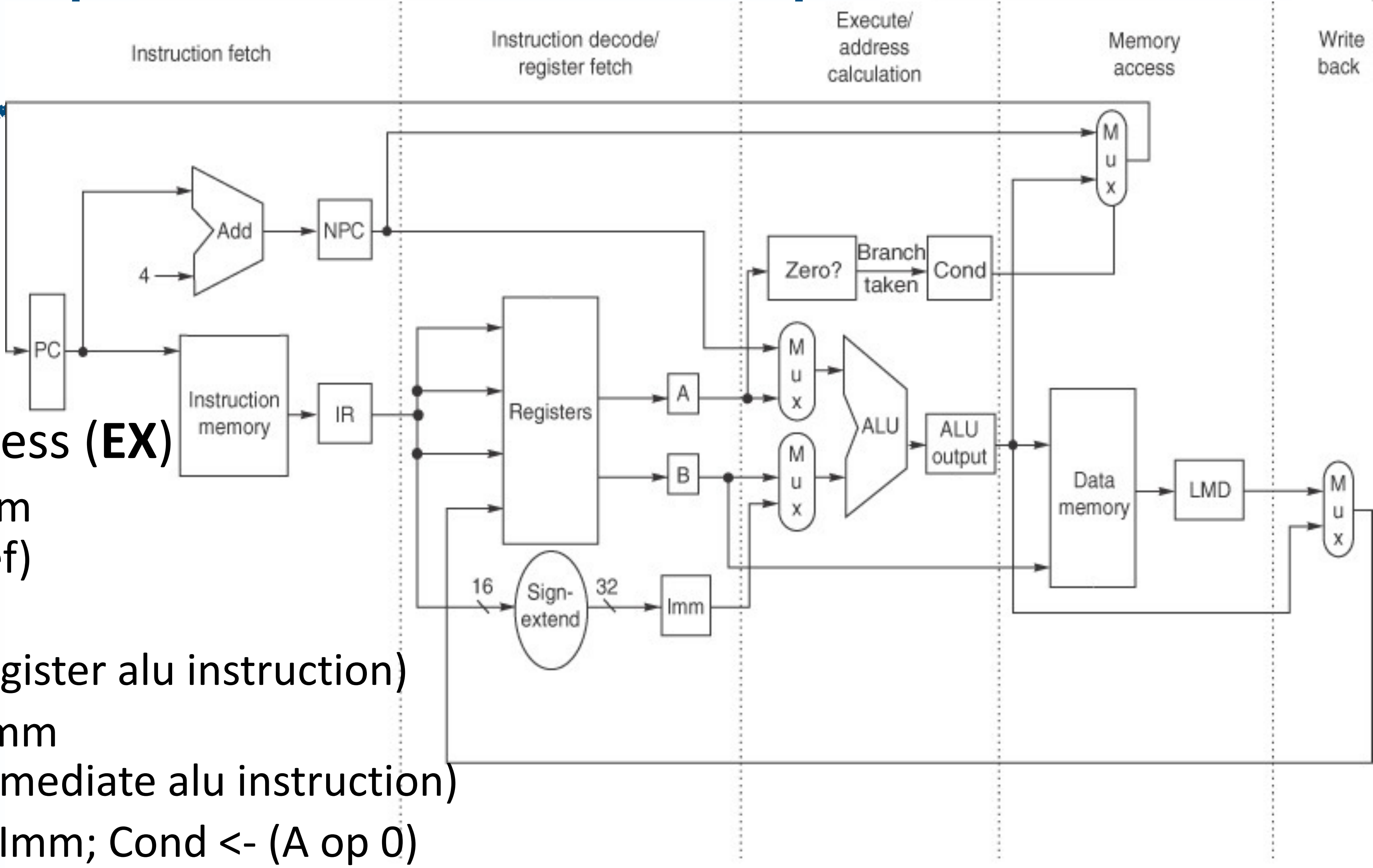
5 Steps of the MIPS Datapath



Instruction Decode/ register read (**ID**)

- A <- Reg[IR6..10]
- B <- Reg[IR11..15]
- Imm <- sign_extend(I

5 Steps of the MIPS Datapath

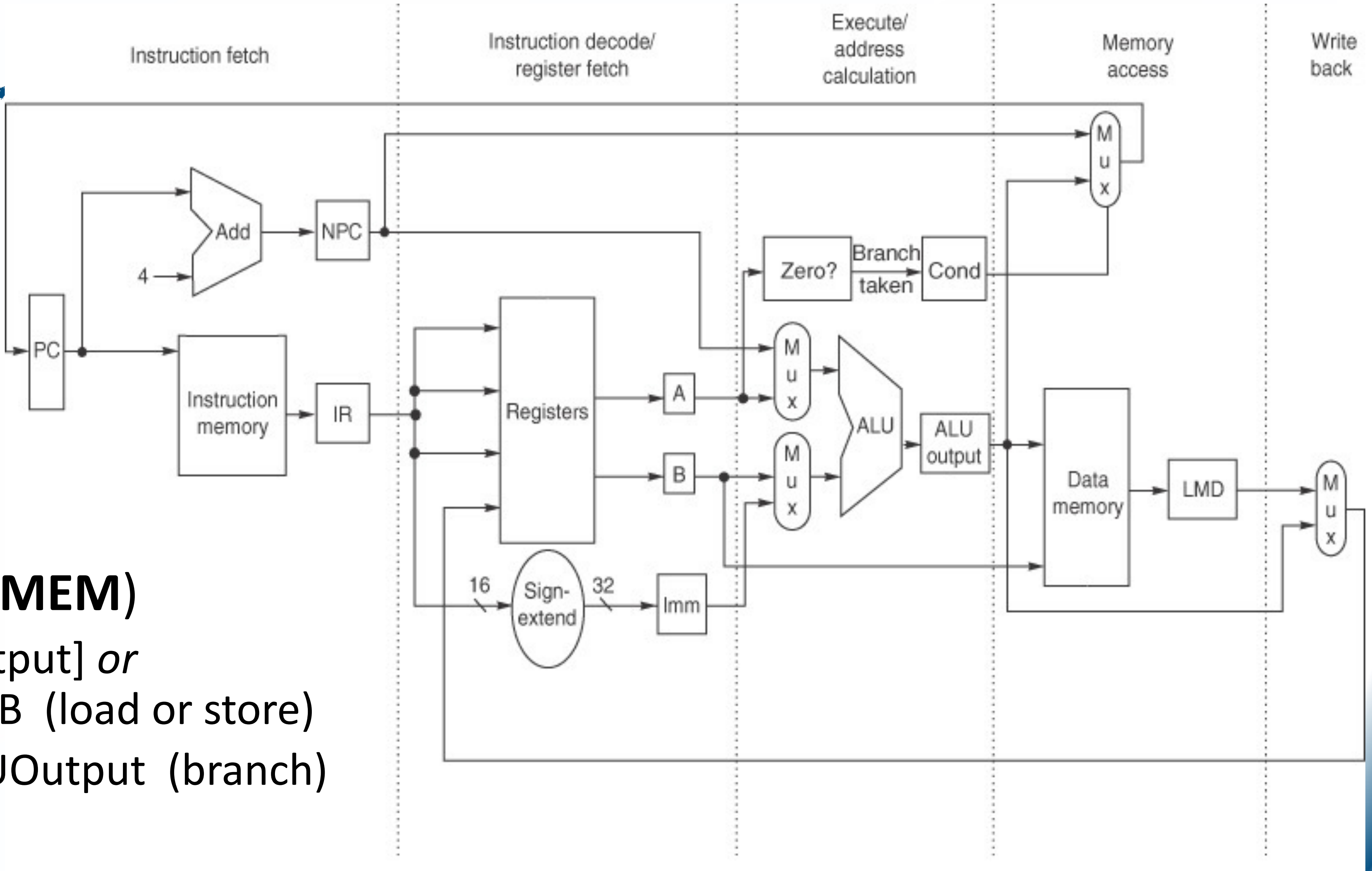


© 2007 Elsevier, Inc. All rights reserved.

Execute/Effective Address (**EX**)

- ALUOutput <- A + Imm
  (memory ref)
- ALUOutput <- A op B
  (register-register alu instruction)
- ALUOutput <- A op Imm
  (register-immediate alu instruction)
- ALUOutput <- NPC + Imm; Cond <- (A op 0)
  (Branch)

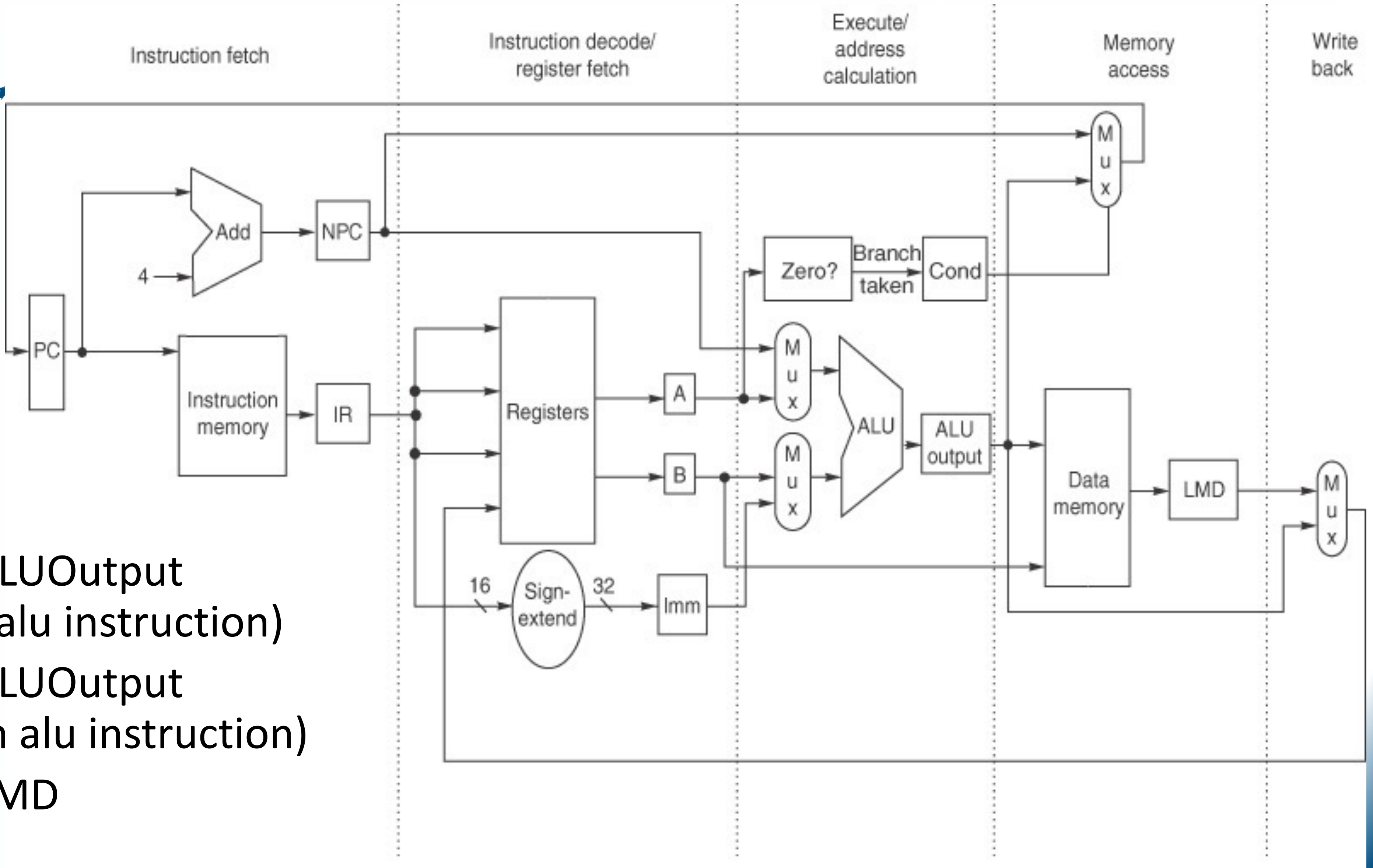
5 Steps of the MIPS Datapath



Memory access/
branch completion (**MEM**)

- LMD <- M[ALUOutput] *or* M[ALUOutput] <- B (load or store)
- if (cond) PC <- ALUOutput (branch)
  else PC <- NPC

5 Steps of the MIPS Datapath

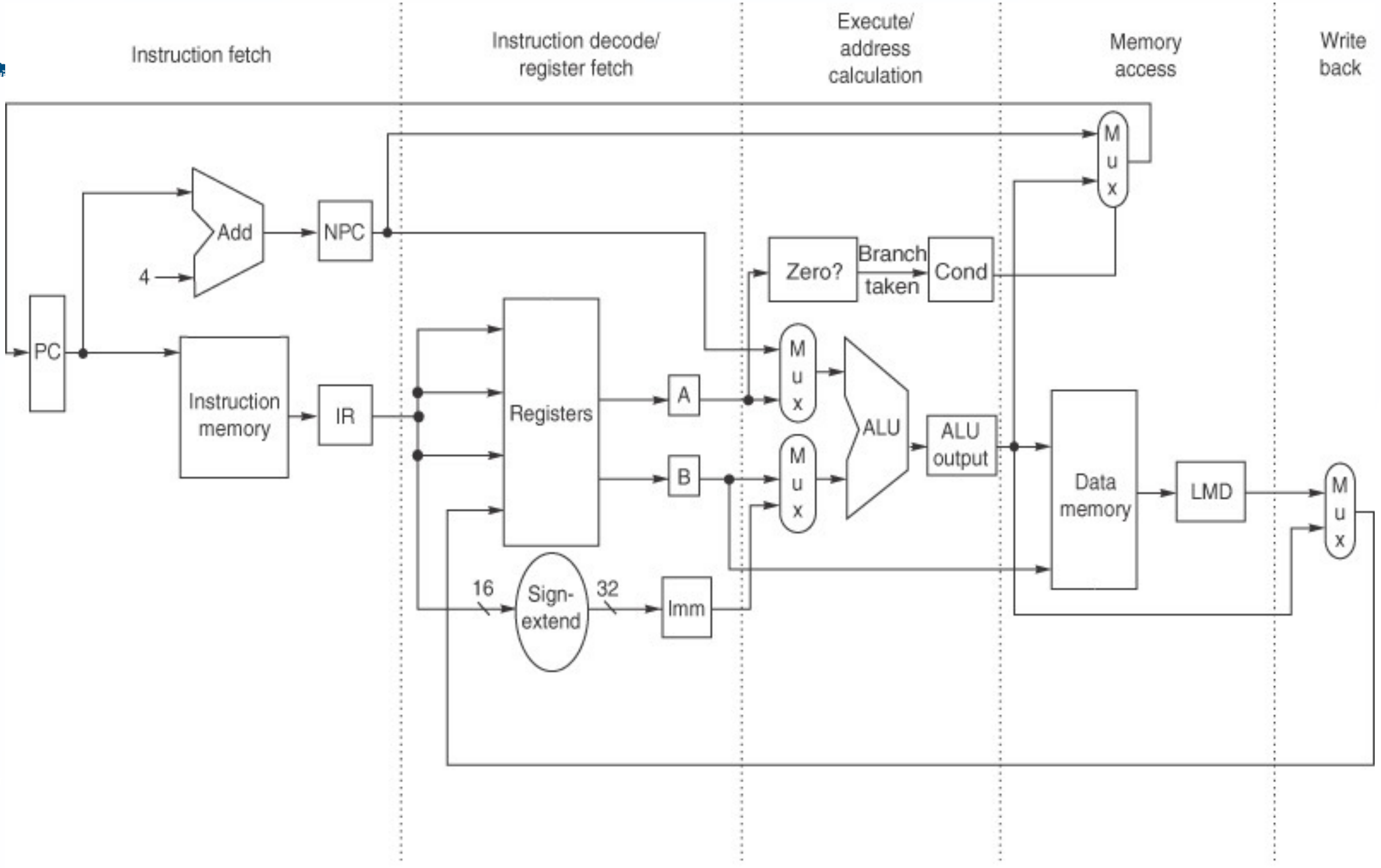


© 2007 Elsevier, Inc. All rights reserved.

Write-Back (**WB**)

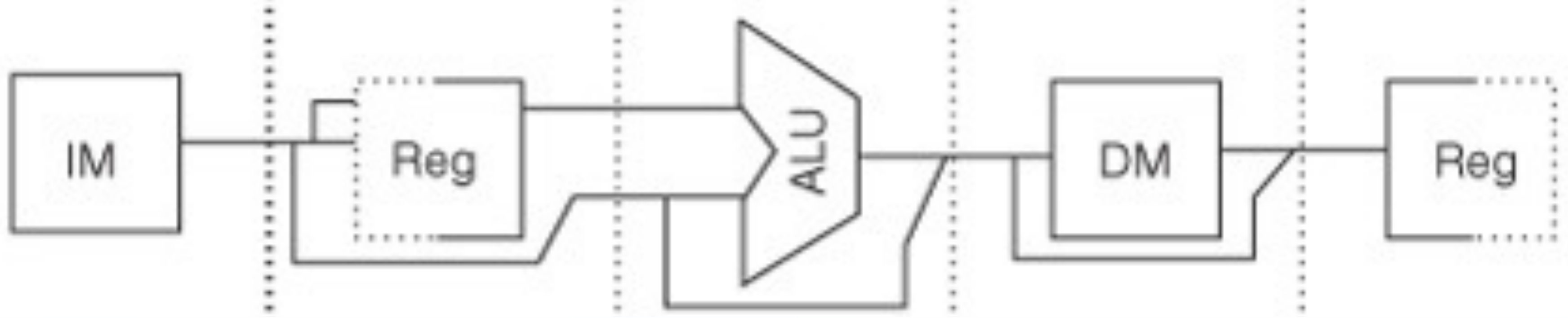
- Reg[IR16..20] <- ALUOutput
  (reg-reg alu instruction)
- Reg[IR11..15] <- ALUOutput
  (reg-imm alu instruction)
- Reg[IR11..15] <- LMD

ADDI R7, R2, #35

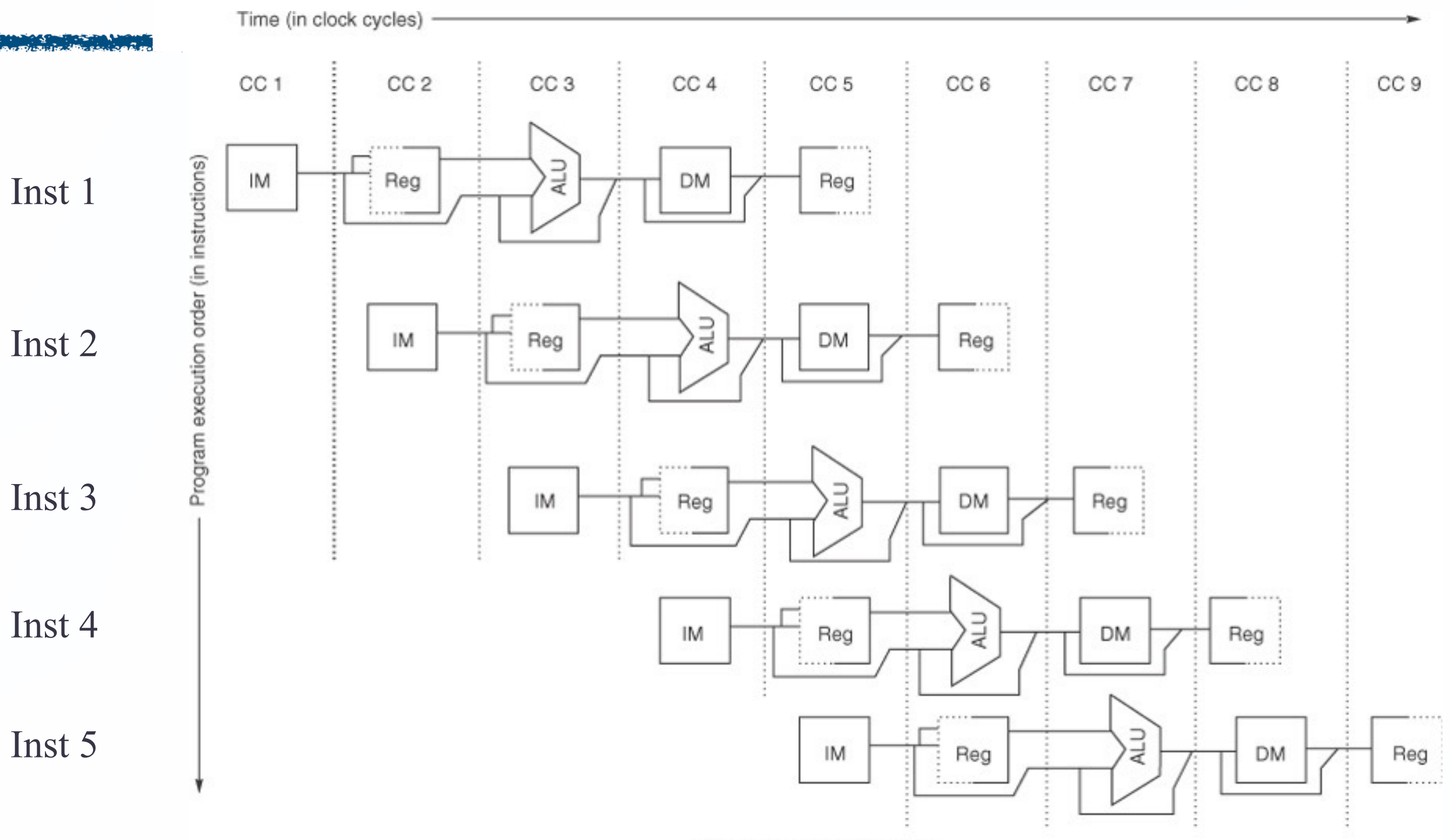


Instruction fetch
Instruction decode/ register fetch
Execute/ address calculation
Memory access
Write back
Add
4
NPC
PC
Instruction memory
IR
Registers
A
B
Zero?
Branch taken
Cond
M u x
ALU
ALU output
16
Sign-extend
32
Imm
Data memory
LMD

5 Steps of a MIPS Instruction



Visualizing Pipelining

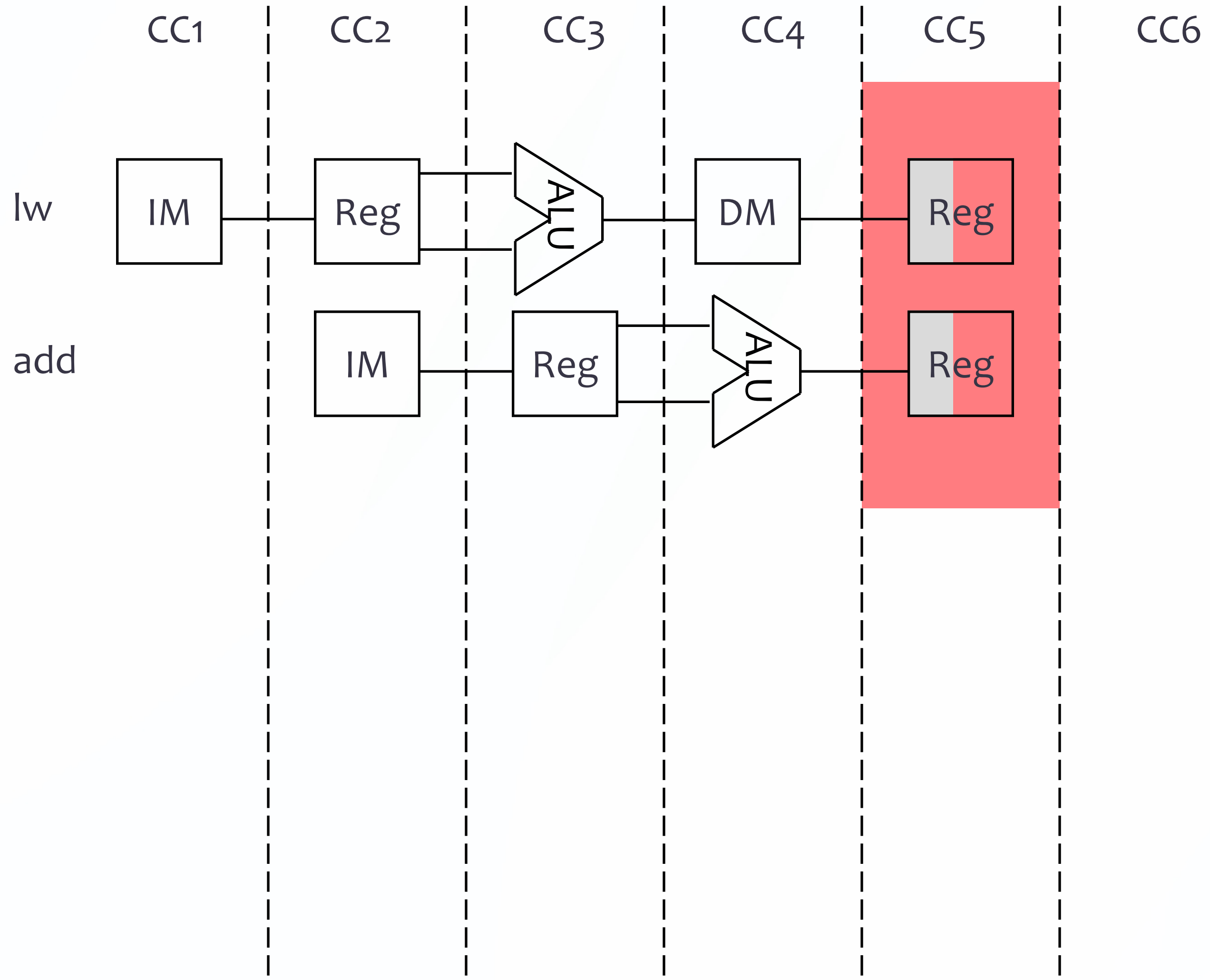


Pipeline Stages

Instead of forcing every instruction to go through all 5 stages, could we let them finish early if done early? For example, an R-type instruction can finish in 4 cycles, can we just let it finish early?

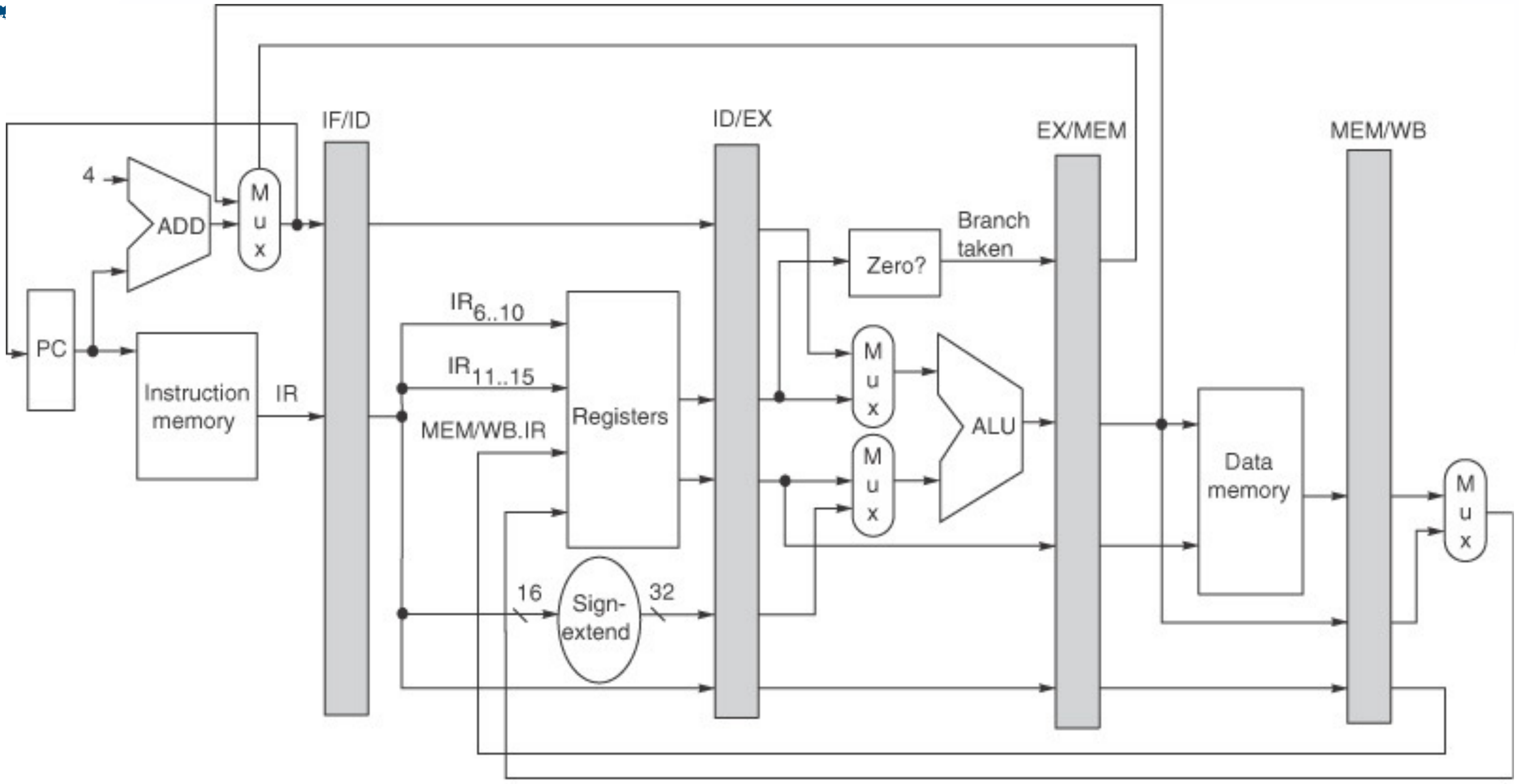
Selection	Yes/No	Reason (Choose BEST answer)
A	Yes	Decreasing R-type to 4 cycles improves instruction throughput
B	Yes	Decreasing R-type to 4 cycles improves instruction latency
C	No	Decreasing R-type to 4 cycles causes hazards
D	No	Decreasing R-type to 4 cycles causes hazards and doesn’t impact throughput
E	No	Decreasing R-type to 4 cycles causes hazards and doesn’t impact latency

Mixed Instructions in the Pipeline



The Pipelined MIPS Datapath

[illegible]

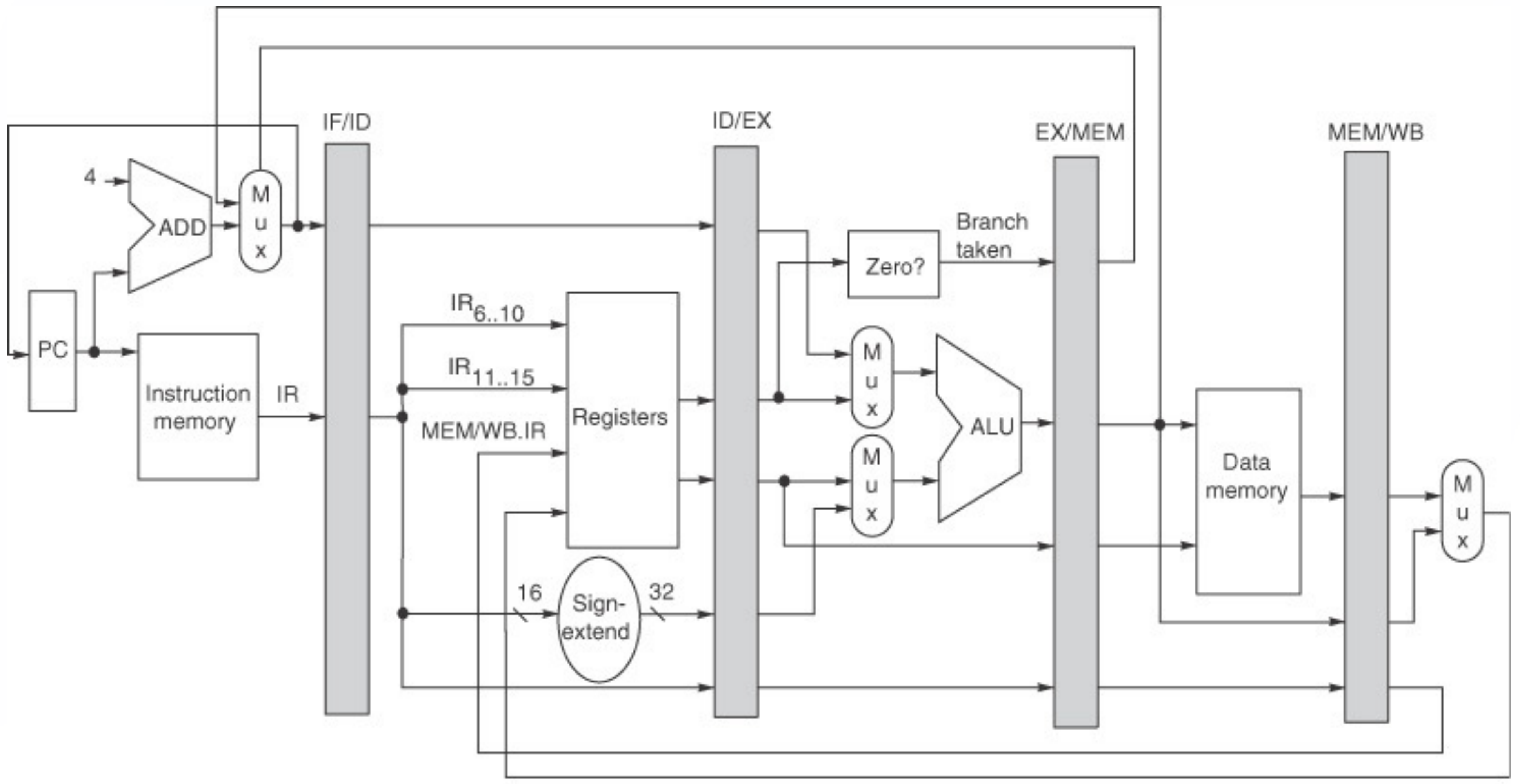


The Pipeline in Motion

- addi R5, R1, #35
- add R6, R2, R1
- lw   R8, 10000(R3)

The Pipeline In Motion

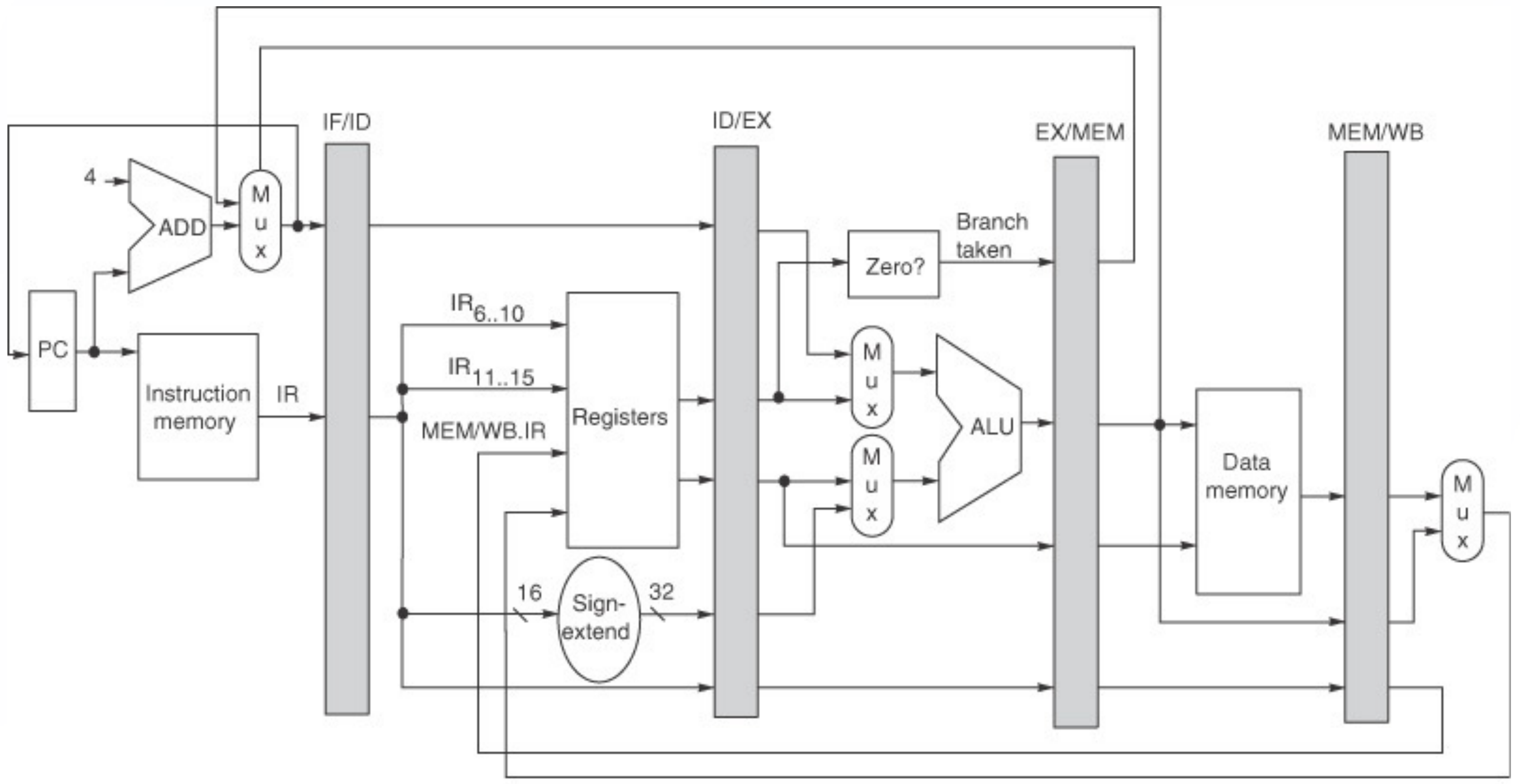
addi R5, R1, #35



The Pipeline In Motion

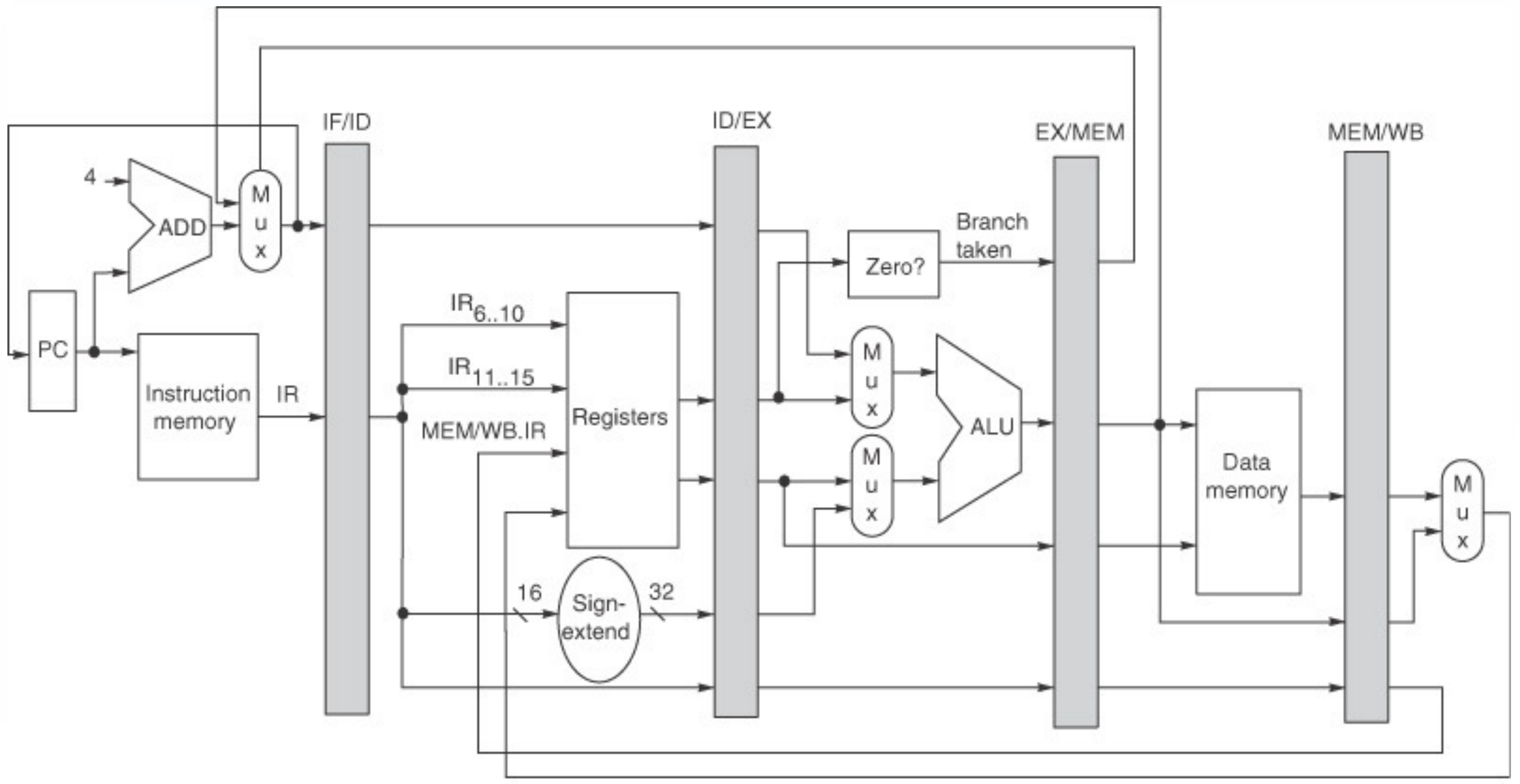
add R6, R2, R1

addi R5, R1, #35



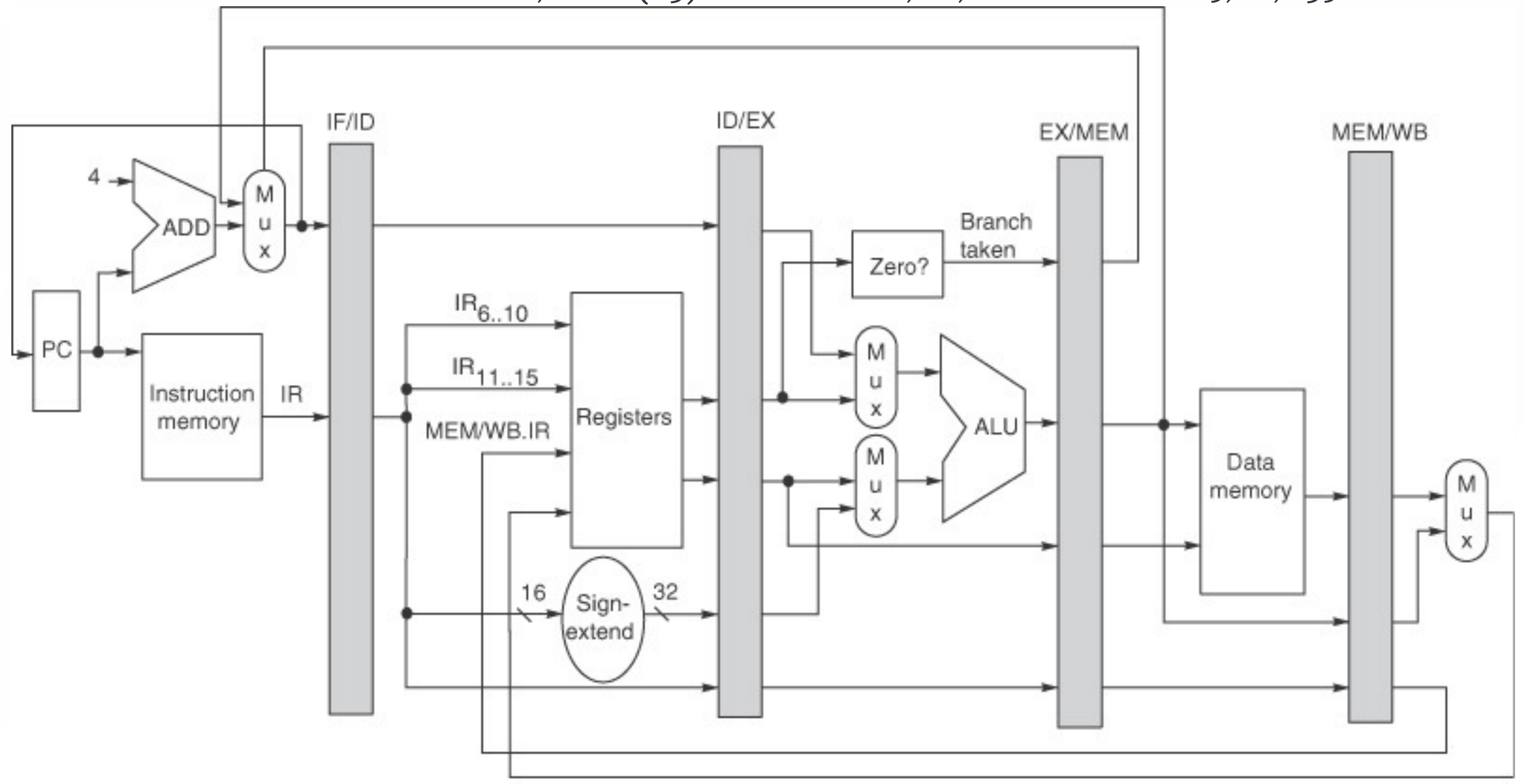
The Pipeline In Motion

lw R8, 10000(R3)   add R6, R2, R1   addi R5, R1, #35



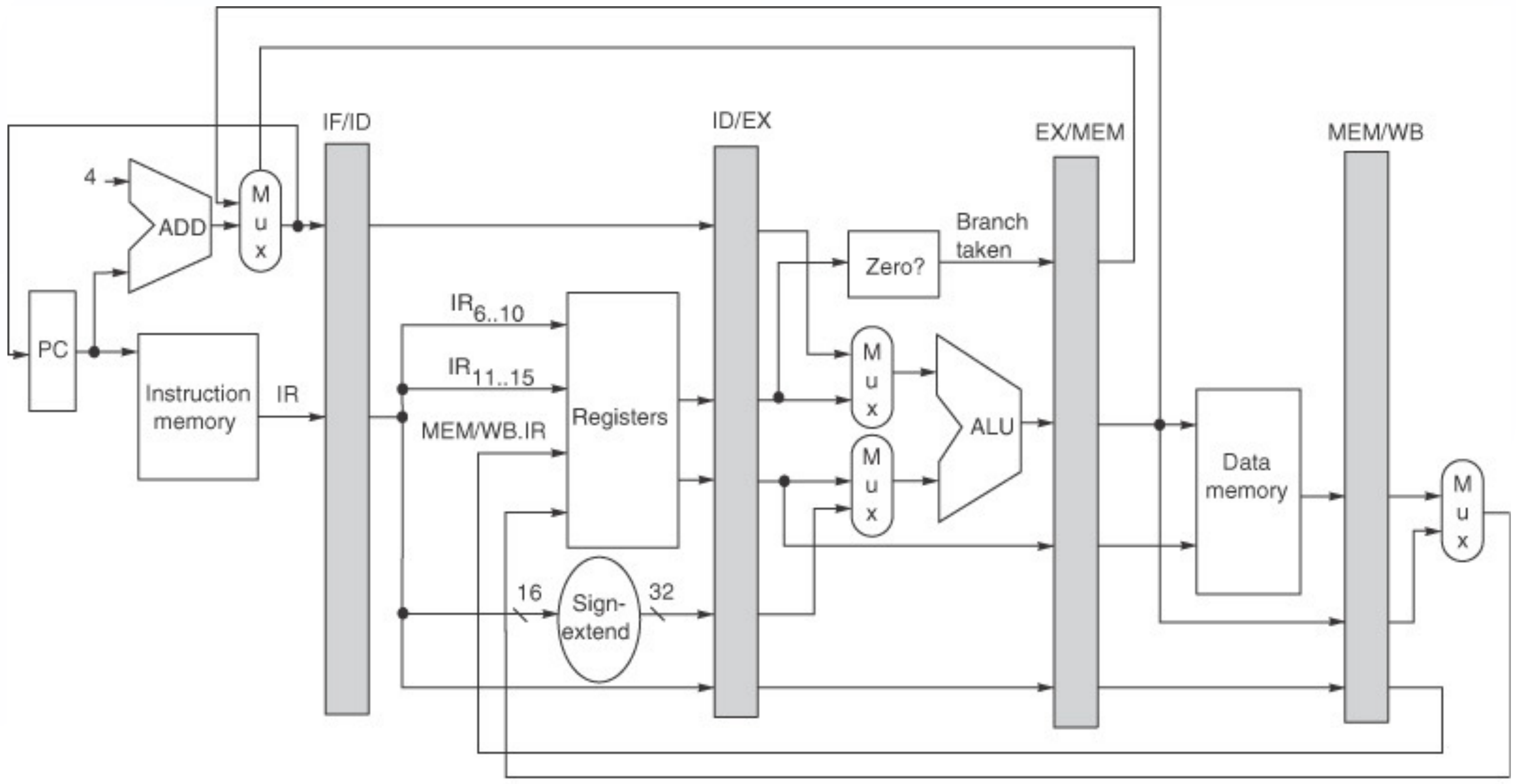
The Pipeline In Motion

lw R8, 10000(R3) add R6, R2, R1 addi R5, R1, #35



The Pipeline In Motion

lw R8, 10000(R3)   add R6, R2, R1   addi R5, R1, #35



The Pipeline in Motion

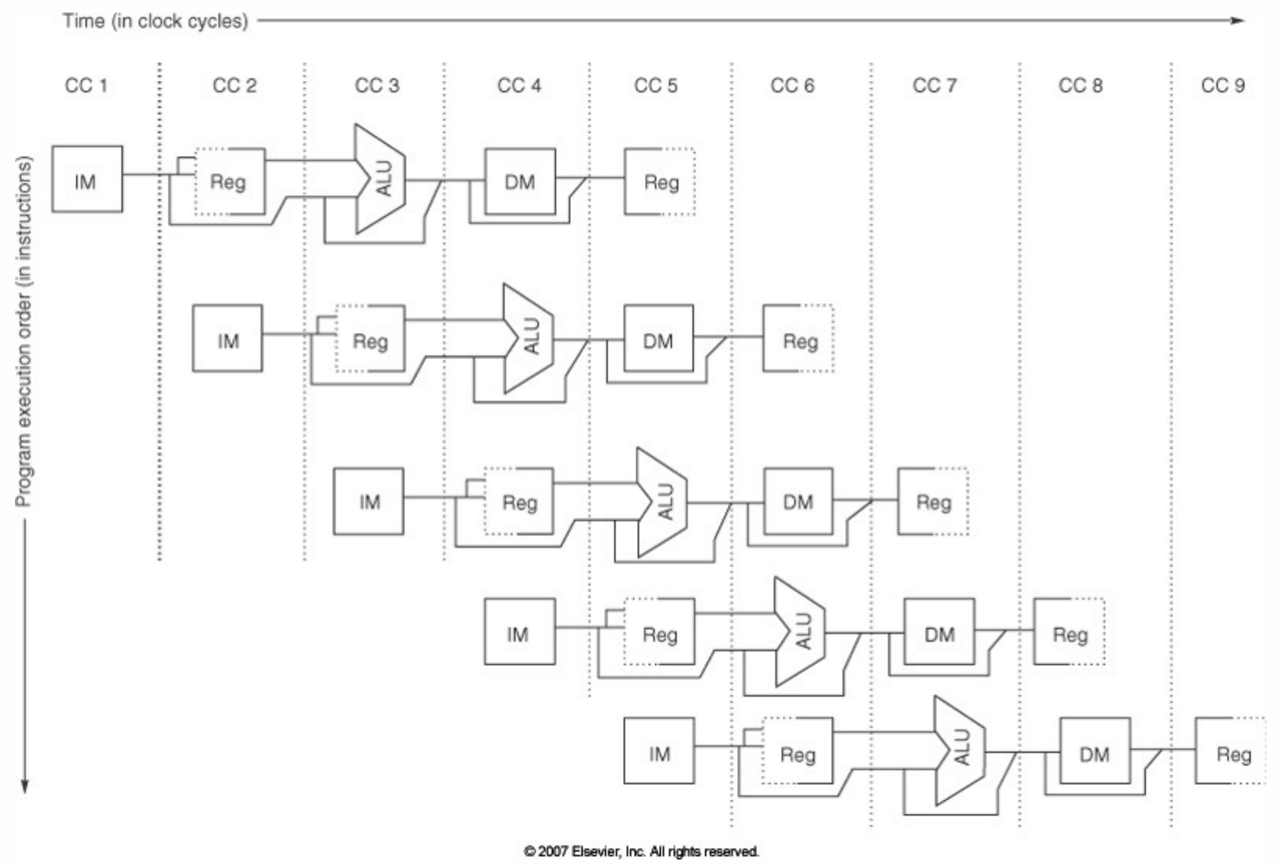
addi R5, R1, #35

add R6, R2, R1

lw R8, 10000(R3)

Inst 4

Inst 5


Time (in clock cycles)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
CC 7
CC 8
CC 9
Program execution order (in instructions)
IM
Reg
ALU
DM
Reg
© 2007 Elsevier, Inc. All rights reserved.

Instruction Latencies and Throughput

Which of the statements below is true about a pipelined processor?

Selection	Statement
A	Individual instruction latency remains essentially unchanged from single-cycle (excluding overheads); Instruction throughput increases
B	Individual instruction latency remains essentially unchanged from multi-cycle (excluding overheads); Instruction throughput decreases
C	Individual instruction latency improves by a factor of 5 over single-cycle (excluding overheads); Instruction throughput increases
D	Individual instruction latency improves by a factor of 5 over multi-cycle (excluding overheads); Instruction throughput decreases
E	None of the above

Single Cycle vs. Multi-cycle

	CPI	CT
Single Cycle		
Multi-cycle		
Pipeline		

Pipeline Performace

- ET = IC * CPI * CT
  - single-cycle processor
  - multiple-cycle processor
  - pipelined processor
- complexity has a cost
  - e.g., latch overhead
  - uneven stage latencies
- Can’t always keep the pipeline full
  - why not?

When Things Go Wrong -- Pipeline Hazards

- Limits to pipelining: **Hazards** prevent next instruction from executing during its designated clock cycle
  - *Structural hazards*: HW cannot support this combination of instructions
  - *Data hazards*: Instruction depends on result of prior instruction still in the pipeline
  - *Control hazards*: Pipelining of branches & other instructions that change the PC
- Common solution is to stall the pipeline until the hazard is resolved, inserting one or more “bubbles” in the pipeline

Key Points

Pipeline improves throughput rather than latency

Pipelining gets parallelism without replication

ET = IC * CPI * CT