CASPER

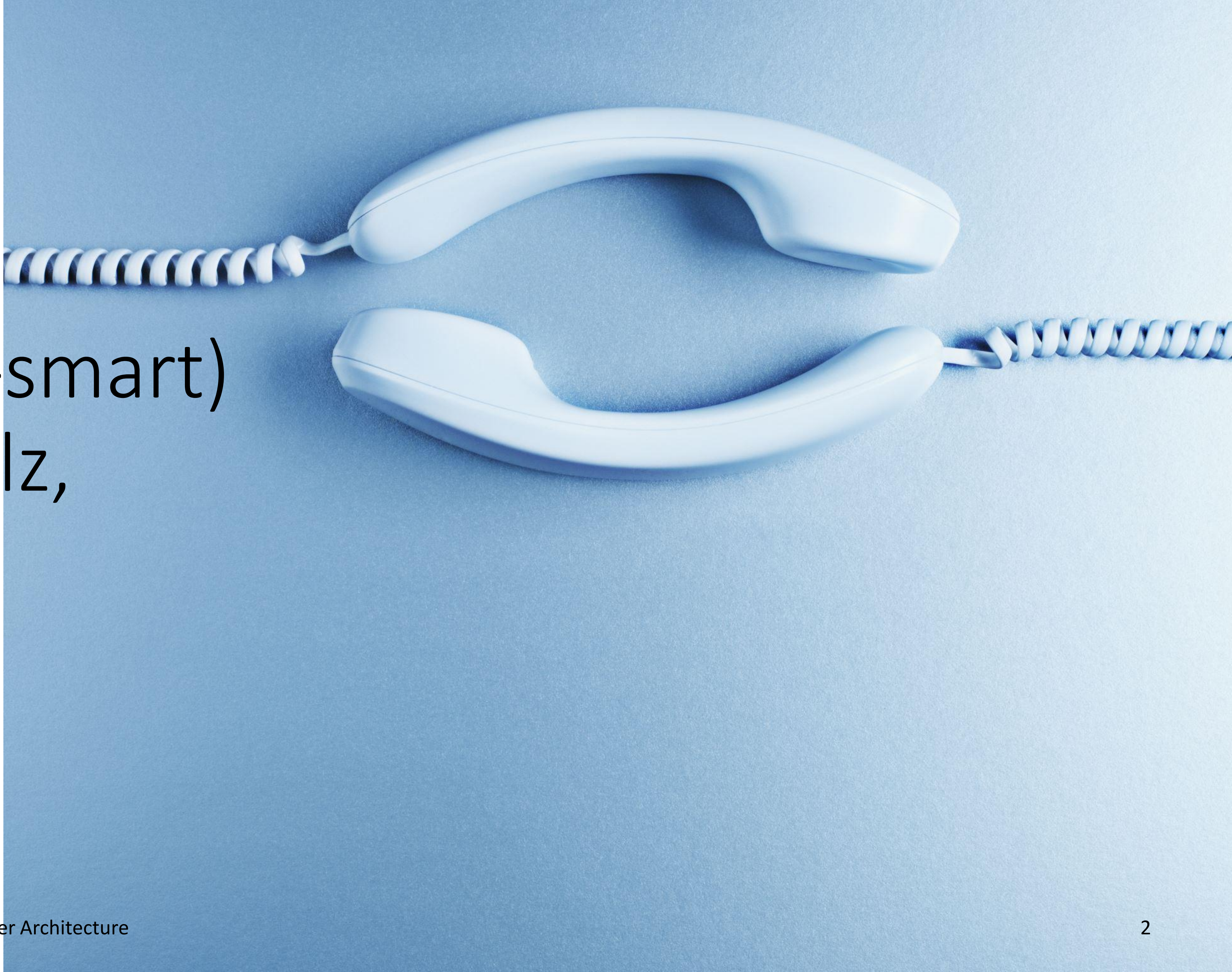
CS230: Digital Logic Design and Computer Architecture

Lecture 11: Instruction Pipelining

https://www.cse.iitb.ac.in/~biswa/courses/CS230/main.html

https://www.cse.iitb.ac.in/~biswa/

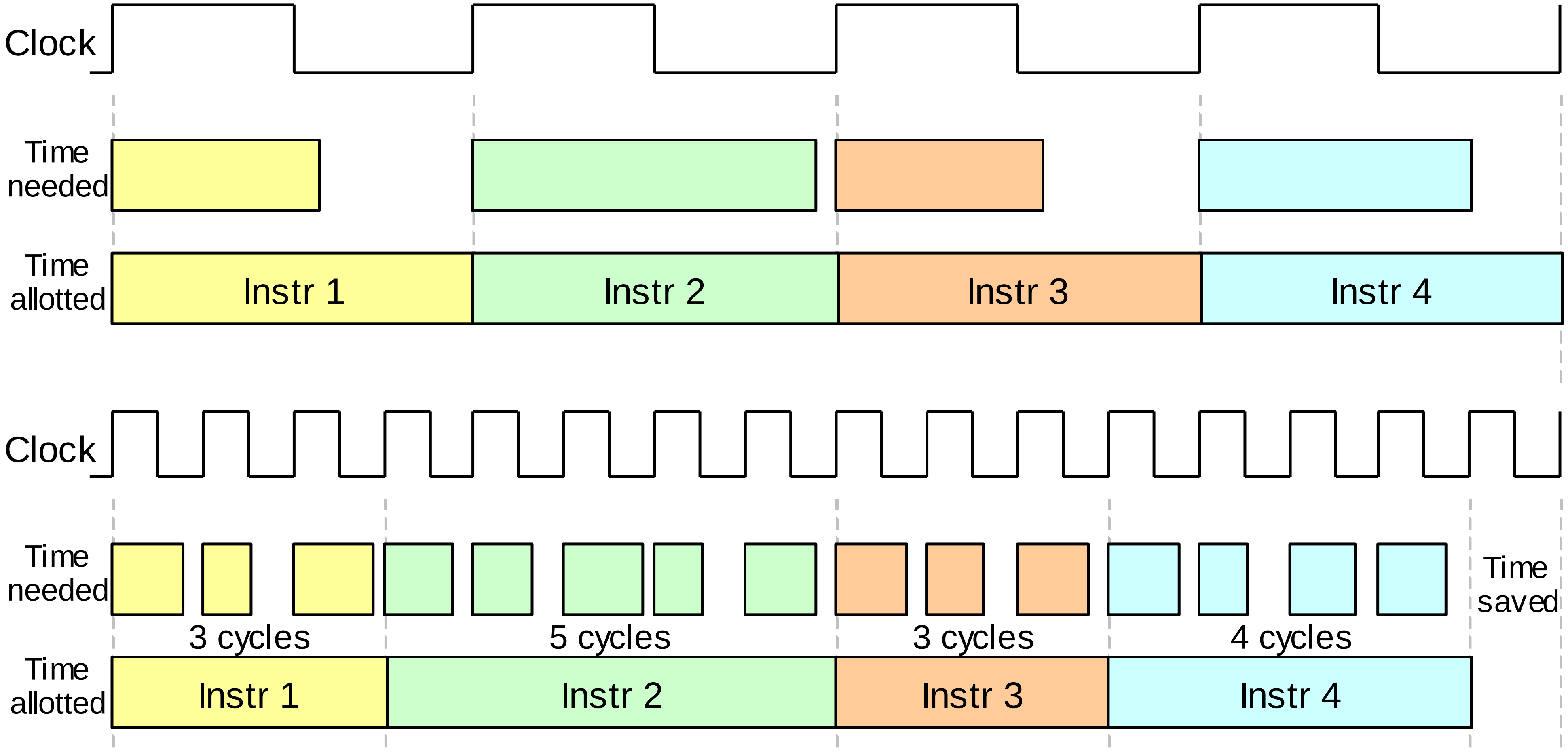
Phones (smart/non-smart) on silence plz, Thanks



Logistics

- First talk: March 3, 11 AM
- Second Quiz: March 13, 3 to 5 PM
- Content: Till March 1
- Third Quiz: April 10, 3 to 5 PM
- Project topics/Yes-No:
- On or before March 10

Single to Multi Cycle



Multicycle CPI

- For example, lw $t0, -4($sp) needs 8ns, assuming the delays shown here.

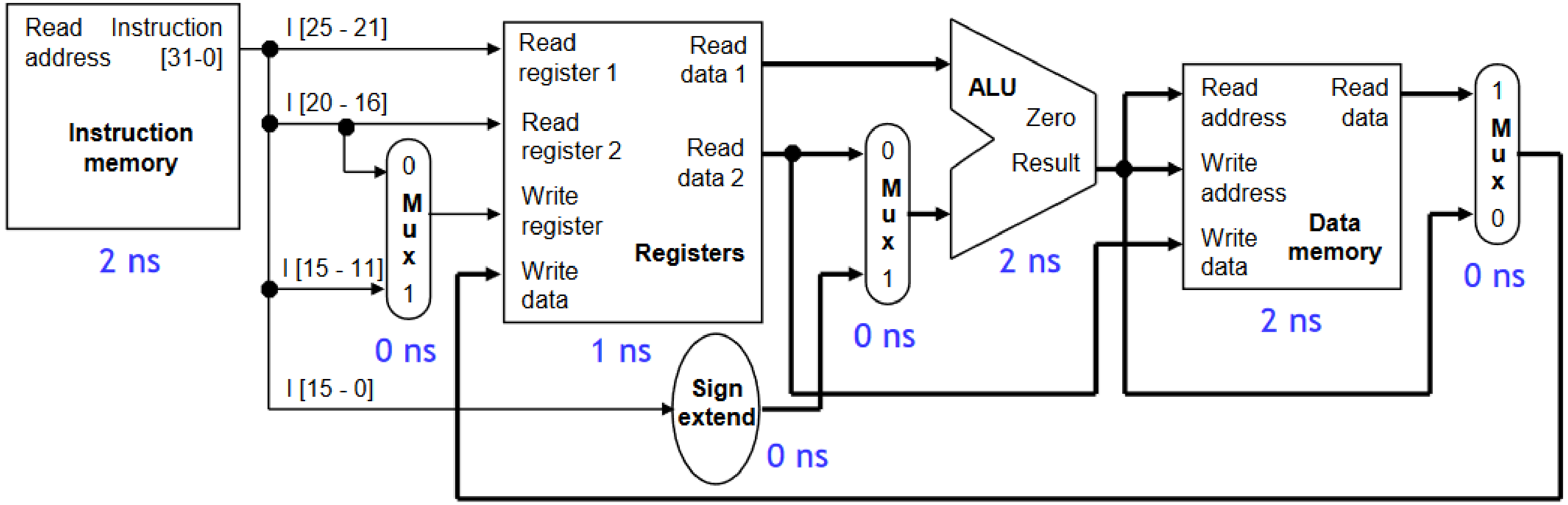
reading the instruction memory	2ns
reading the base register $sp	1ns
computing memory address $sp-4	2ns
reading the data memory	2ns
storing data back to $t0	1ns

8ns

one clock cycle: 2ns

Processor frequency: 500MHz

Cycle per Instruction (CPI) ~ 4



Single/multi-cycle (COVID19 vaccine schedule)

Single cycle (Worst case)

One shot will take 60 minutes one slot = 60 minutes

Multi cycle (average case kinda)

One shot: five to 60 minutes one slot = 15 minutes

Can We Have Both?

Faster clock rate and also CPI=1?

COVID19 Vaccination Schedule

Single cycle: one hour per person

verification: 25 minutes

5 minutes for vaccination

20 minutes post-vaccination

10 minutes, certificate ☺

COVID19 Vaccination Schedule

Single cycle: Three hour, three persons

verification: 25 minutes

5 minutes for vaccination

20 minutes post-vaccination

10 minutes, certificate ☺

COVID19 Vaccination Pipelined Schedule

Single cycle: one hour per person

Stage 1 verification: 15 minutes

Stage 2 verification: 10 minutes

Stage 3: 5 minutes for vaccination

Stage 4: 20 minutes post-vaccination

Stage 5: 10 minutes, certificate ☺

COVID19 Vaccination Pipelined Schedule

Single cycle: one hour per person

Stage 1 verification: 15 minutes

Stage 2 verification: 10 minutes

Stage 3: 5 minutes for vaccination

Stage 4: 20 minutes post-vaccination

Stage 5: 10 minutes, certificate ☺

COVID19 Vaccination Pipelined Schedule

- Single cycle: one hour per person
- Stage 1 verification: 15 minutes
- Stage 2 verification: 10 minutes
- Stage 3: 5 minutes for vaccination
- Stage 4: 20 minutes post-vaccination
- Stage 5: 10 minutes, certificate ☺

COVID19 Vaccination Pipelined Schedule

Single cycle: one hour per person

Stage 1 verification: 15 minutes

Stage 2 verification: 10 minutes

Stage 3: 5 minutes for vaccination

Stage 4: 20 minutes post-vaccination

Stage 5: 10 minutes, certificate ☺

COVID19 Vaccination Pipelined Schedule

Single cycle: one hour per person

Stage 1 verification: 15 minutes

Stage 2 verification: 10 minutes

Stage 3: 5 minutes for vaccination

Stage 4: 20 minutes post-vaccination

Stage 5: 10 minutes, certificate ☺

COVID19 Vaccination Pipelined Schedule

Pipelined: One hour: Three persons

Stage 1 verification: 15 minutes

Stage 2 verification: 10 minutes

Stage 3: 5 minutes for vaccination

Stage 4: 20 minutes post-vaccination

Stage 5: 10 minutes, certificate ☺

COVID19 Vaccination Pipelined Schedule

Pipelined: One hour: ? Persons, first person: 100 minutes ☹, after a while throughput= 3 persons/hr

Stage 1 (20 mins): verification

Stage 2 (20 mins): verification

Stage 3 (20 mins): 5 minutes for vaccination

Stage 4 (20 mins): 20 minutes post-vaccination

Stage 5 (20 mins):10 minutes, certificate ☺

Think about variable stages, does it improve latency and throughput?

Let’s pause a bit

Single cycle: CPI: 1 , Cycle time: long

Multi cycle: CPI: >1, Cycle time: short

Pipelined: CPI: 1, Cycle time: short (improves throughput but not latency)

Latency and Bandwidth (throughput)

- Latency
  - time it takes to complete one instance

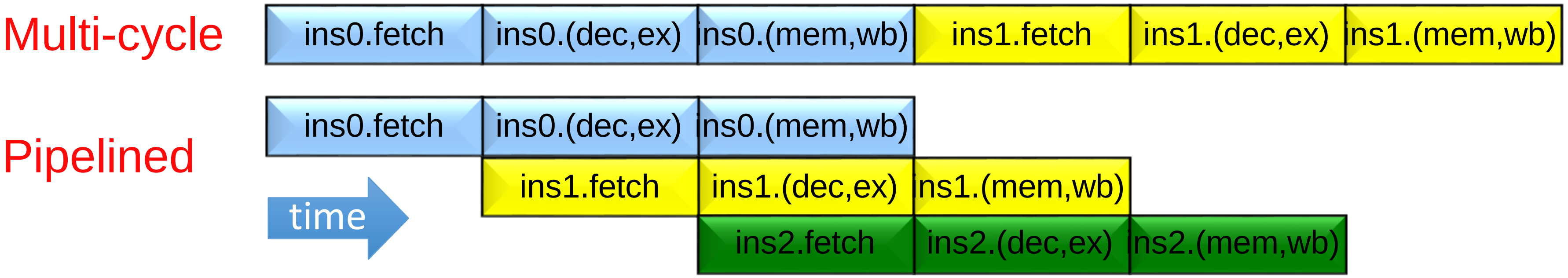
- Throughput
  - number of computations done per unit time

Pipelining and Richard Feynman

https://www.youtube.com/watch?v=9miKIWIYi4w

Jump to 1:25

Multi-cycle vs Pipelined



Multi-cycle

ins0.fetch | ins0.(dec,ex) | ins0.(mem,wb) | ins1.fetch | ins1.(dec,ex) | ins1.(mem,wb)

Pipelined

ins0.fetch | ins0.(dec,ex) | ins0.(mem,wb)

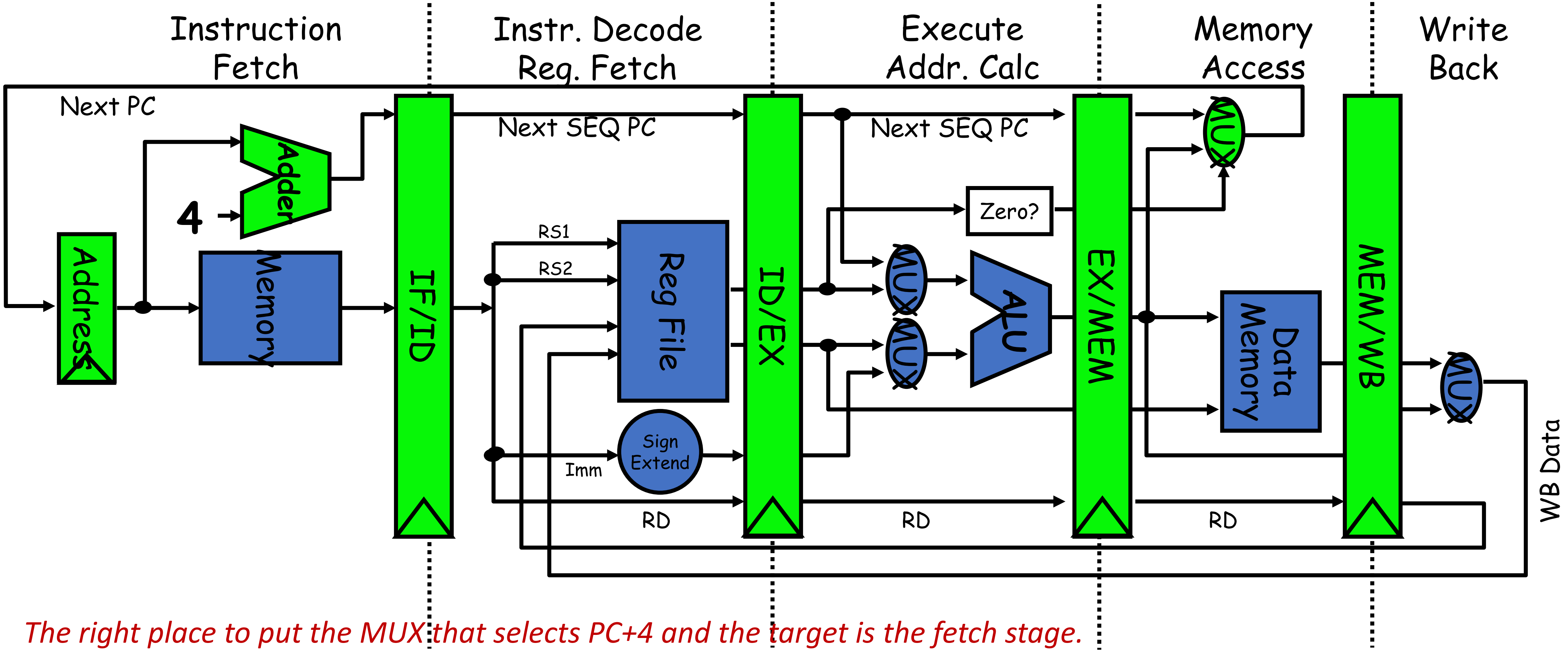
ins1.fetch | ins1.(dec,ex) | ins1.(mem,wb)

ins2.fetch | ins2.(dec,ex) | ins2.(mem,wb)

time

Real World

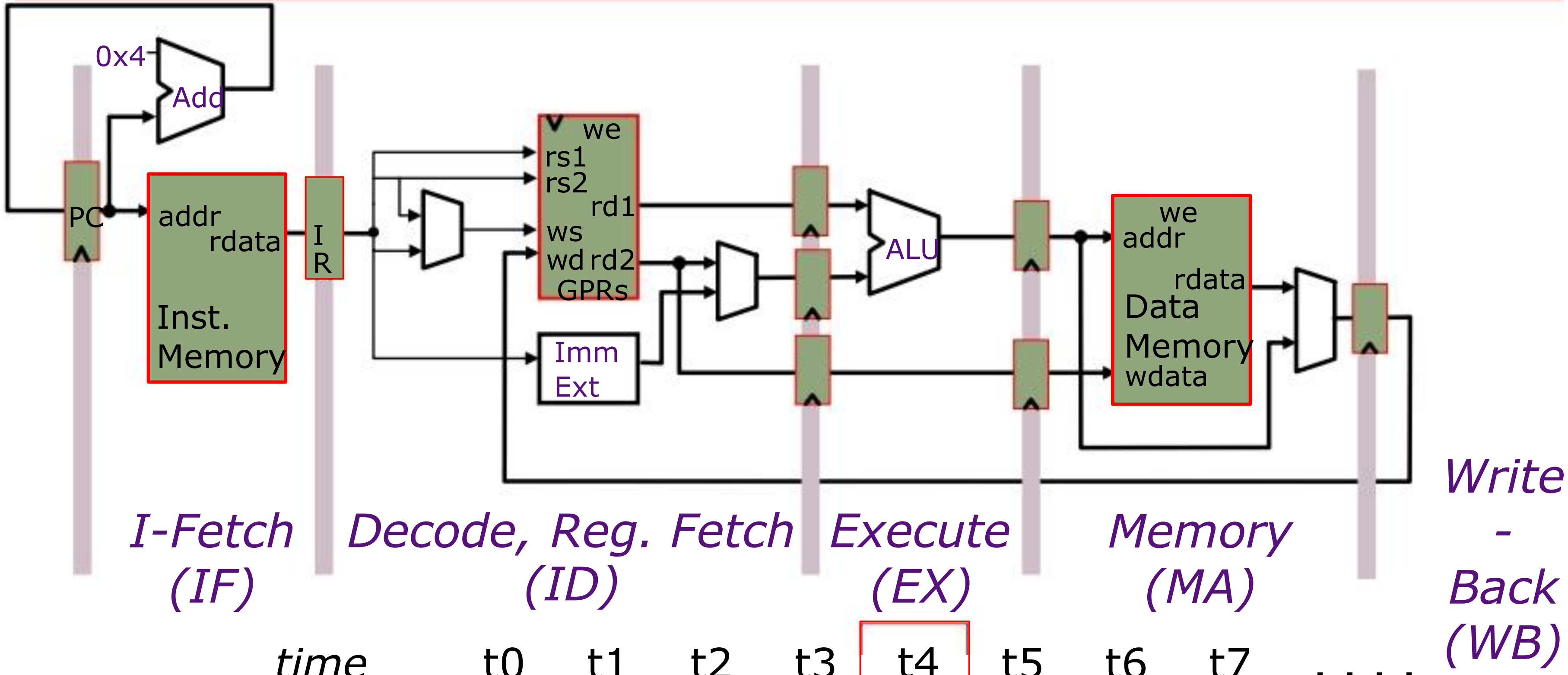
Vanilla 5-stage pipeline



The right place to put the MUX that selects PC+4 and the target is the fetch stage.

The slide shows a vanilla 5-stage pipeline if we just take a single cycle datapath and divide it into five stages.

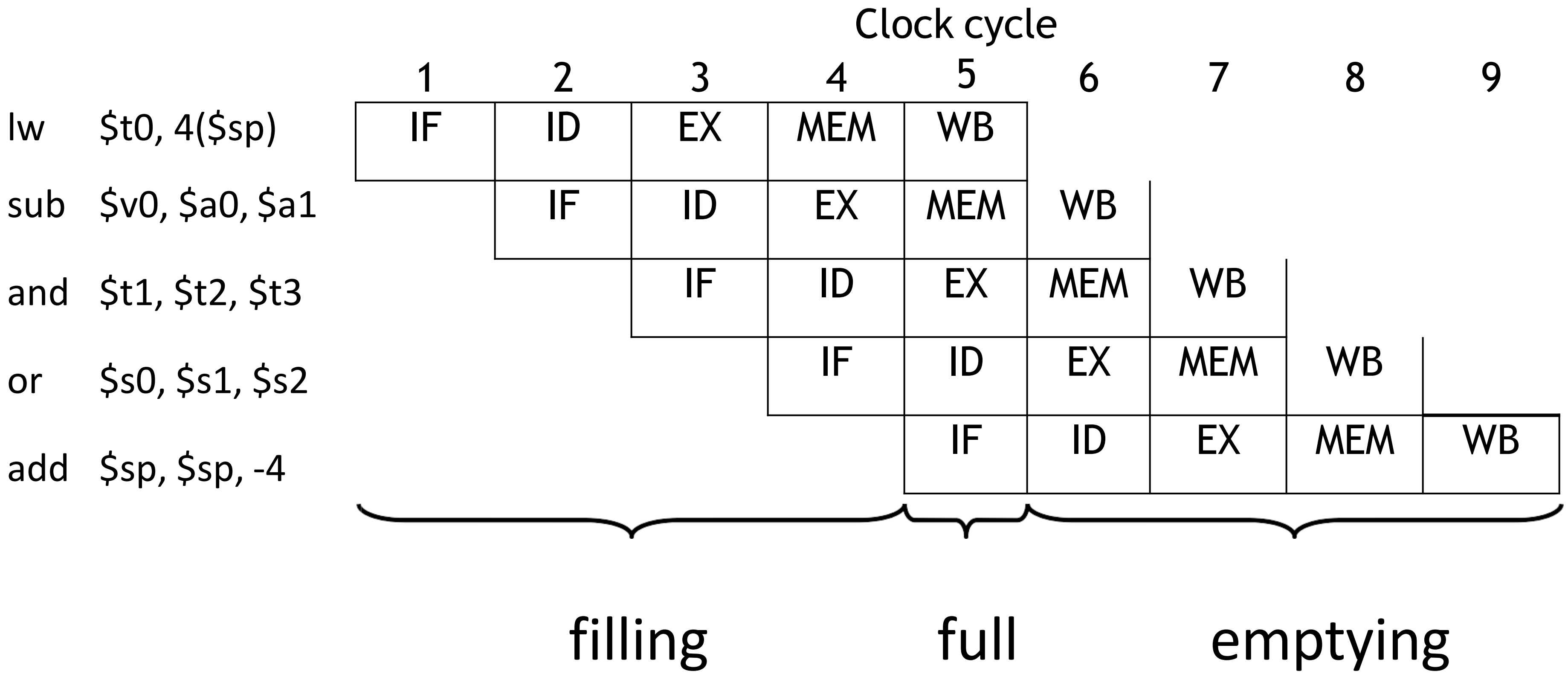
Resource Utilization



Resources

time	t0	t1	t2	t3	t4	t5	t6	t7	. . . .
IF	I_1	I_2	I_3	I_4	I_5				
ID		I_1	I_2	I_3	I_4	I_5			
EX			I_1	I_2	I_3	I_4	I_5		
MA				I_1	I_2	I_3	I_4	I_5	
WB					I_1	I_2	I_3	I_4	I_5

Visualizing Pipeline

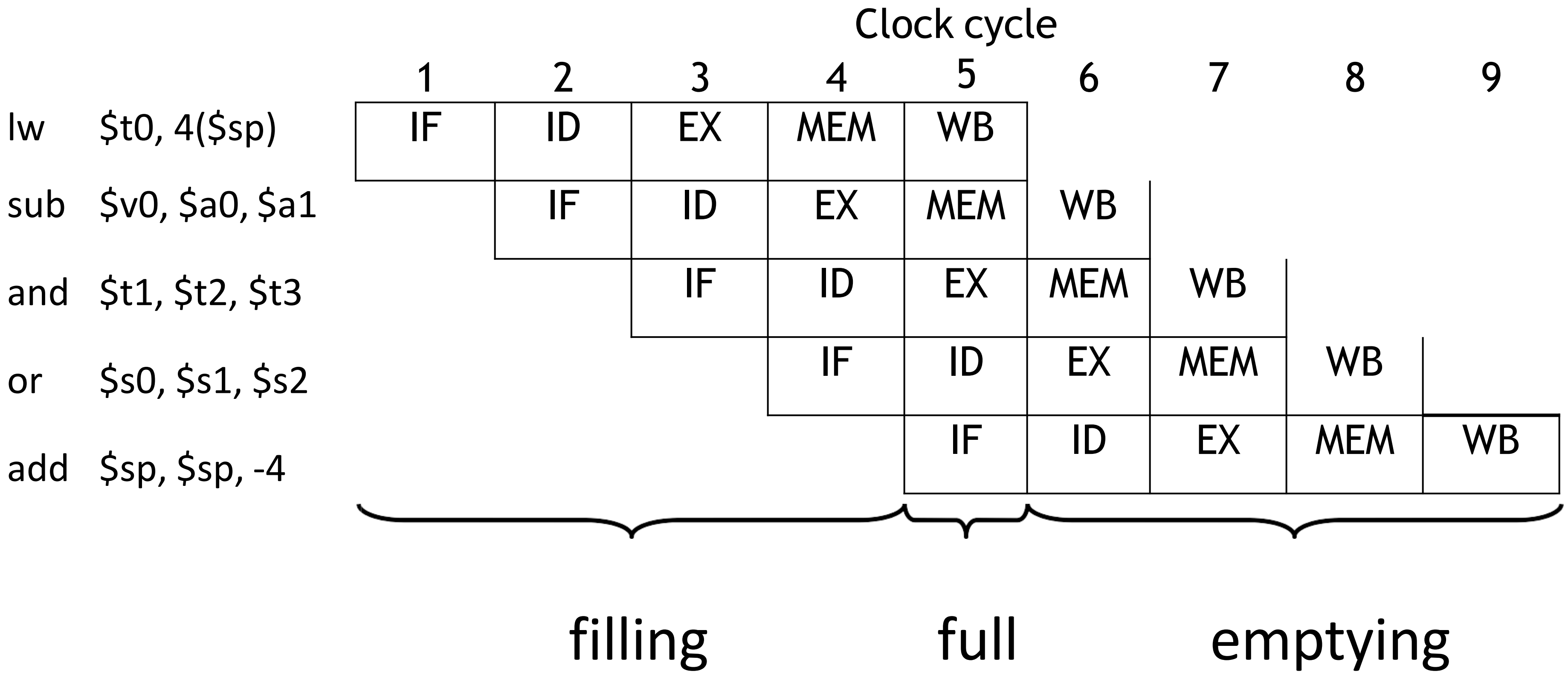


Clock cycle

	1	2	3	4	5	6	7	8	9
lw $t0, 4($sp)	IF	ID	EX	MEM	WB				
sub $v0, $a0, $a1		IF	ID	EX	MEM	WB			
and $t1, $t2, $t3			IF	ID	EX	MEM	WB		
or $s0, $s1, $s2				IF	ID	EX	MEM	WB	
add $sp, $sp, -4					IF	ID	EX	MEM	WB

filling full emptying

Visualizing Pipeline: Execution time



Clock cycle

	1	2	3	4	5	6	7	8	9
lw $t0, 4($sp)	IF	ID	EX	MEM	WB				
sub $v0, $a0, $a1		IF	ID	EX	MEM	WB			
and $t1, $t2, $t3			IF	ID	EX	MEM	WB		
or $s0, $s1, $s2				IF	ID	EX	MEM	WB	
add $sp, $sp, -4					IF	ID	EX	MEM	WB

For a k-stage pipeline executing N instructions

first instruction: K cycles

Next N-1 instructions: N-1 cycles, total = K + (N-1) cycles

Pipelined versus Single cycle CPU design

Instruction	Ifetch	Decode	Execute	Memory	Writeback	Total time
LOAD	200ns	100	200	200	100	800ns
STORE	200	100	200	200		700ns
ADD	200	100	200		100	600ns
BRANCH	200	100	200			500ns

Total latency in single cycle CPU: 3200 ns

Total latency in pipelined CPU (200ns clock cycle):
1000ns (1st instruction) + 3 X 200 ns (for next three) = 1600 ns

What’s the big deal

Speedup = 3200ns/1600ns = 2X

What if we have a billion instructions?

Single cycle = 1 billion X 800ns = 800 seconds

Pipelined = 1000ns + (1 billion -1) X 200ns ~ 200 seconds

Speedup = 4X ☺

Let’s include latch latency too

Inter-stage latch = 10ns

New clock cycle time in the pipelined design = 210ns

First instruction will get completed by 1040ns (five stages X 200 ns + four inter-stage latches X 10ns)

New Speedup = 800s/210s ~ 3.8X

How to Divide the Datapath?

Suppose memory is significantly slower than other stages. For example, suppose

t_{IM} = 10 units

t_{DM} = 10 units

t_{ALU} = 5 units

t_{RF} = 1 unit

t_{RW} = 1 unit

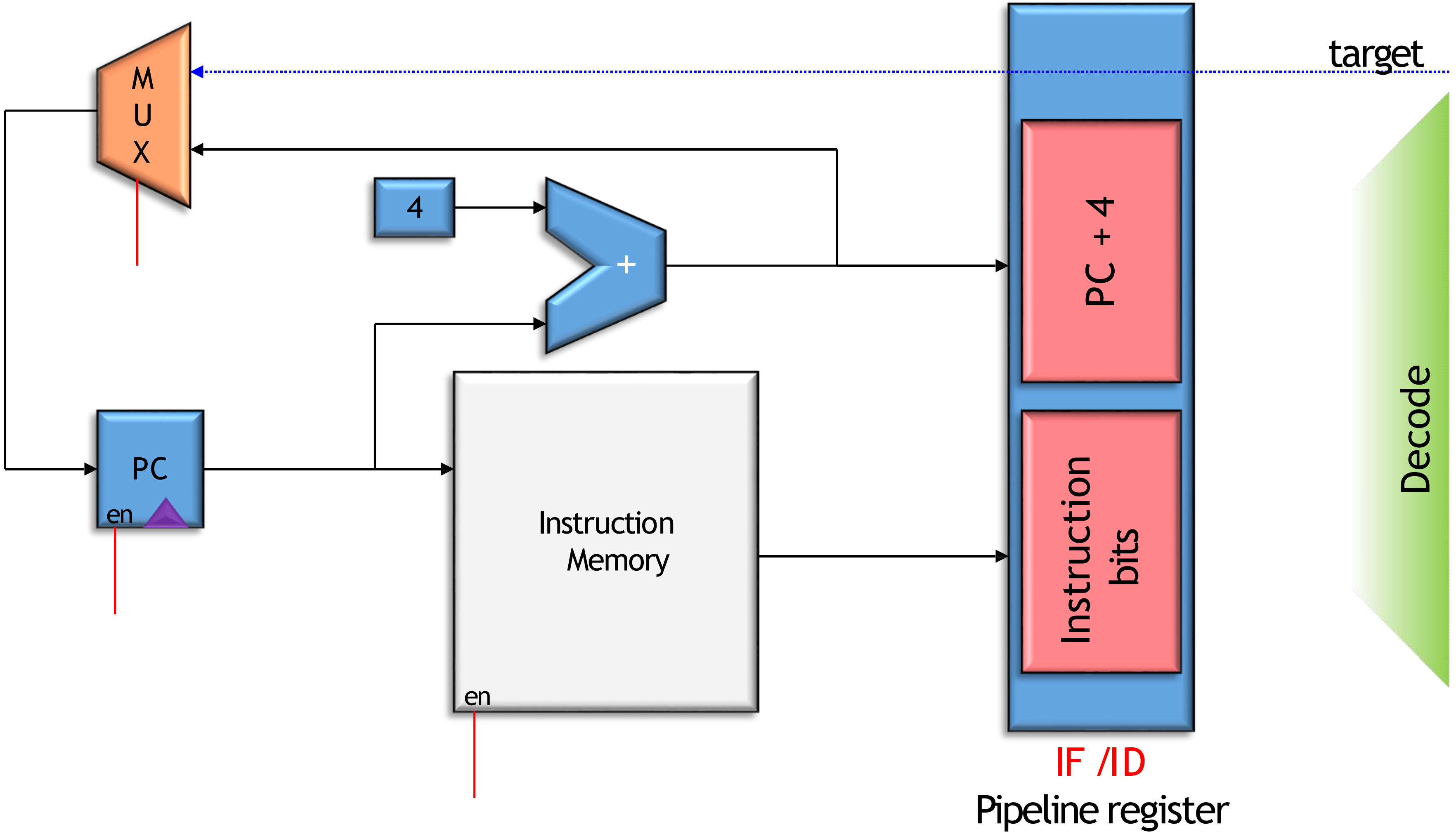
Since the slowest stage determines the clock, it may be possible to combine some stages without any loss of performance

#Stages and Speedup

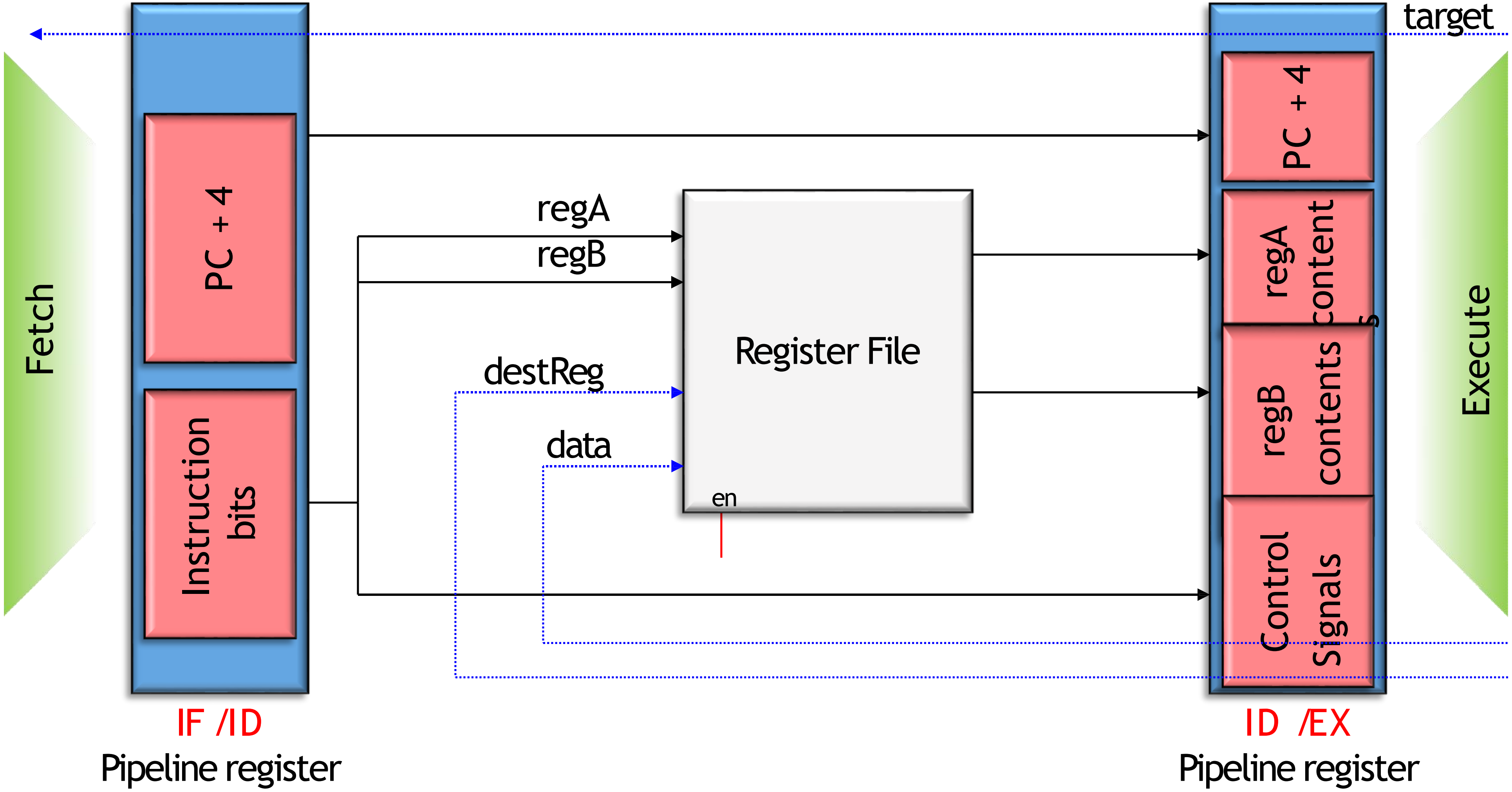
Assumptions	Unpipelined t_C	Pipelined t_C	Speedup
1. $t_{IM} = t_{DM} = 10$, $t_{ALU} = 5$, $t_{RF} = t_{RW} = 1$ 4-stage pipeline	27	10	2.7
2. $t_{IM} = t_{DM} = t_{ALU} = t_{RF} = t_{RW} = 5$ 4-stage pipeline	25	10	2.5
3. $t_{IM} = t_{DM} = t_{ALU} = t_{RF} = t_{RW} = 5$ 5-stage pipeline	25	5	5.0

Devil is in the details

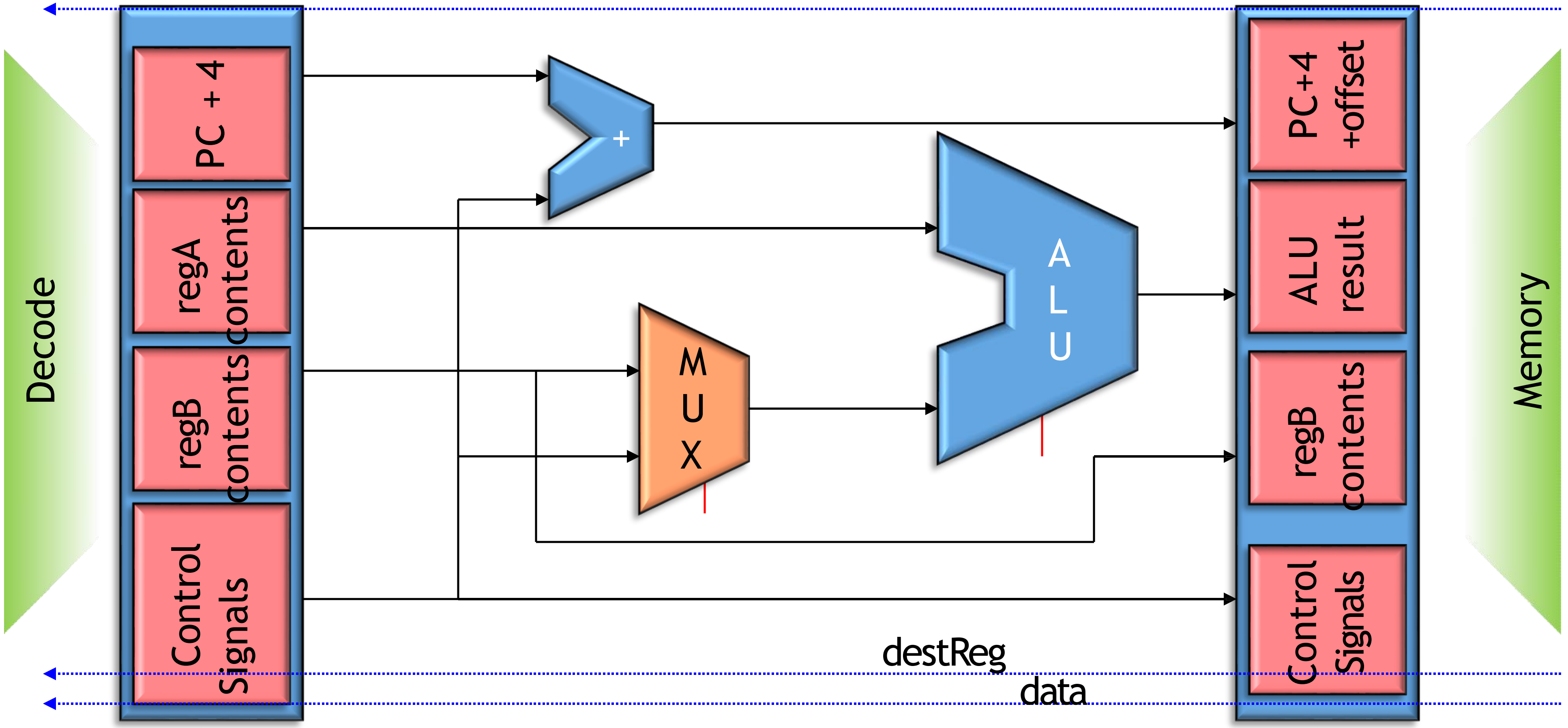
Stage-1: Fetch



Stage 2: Decode



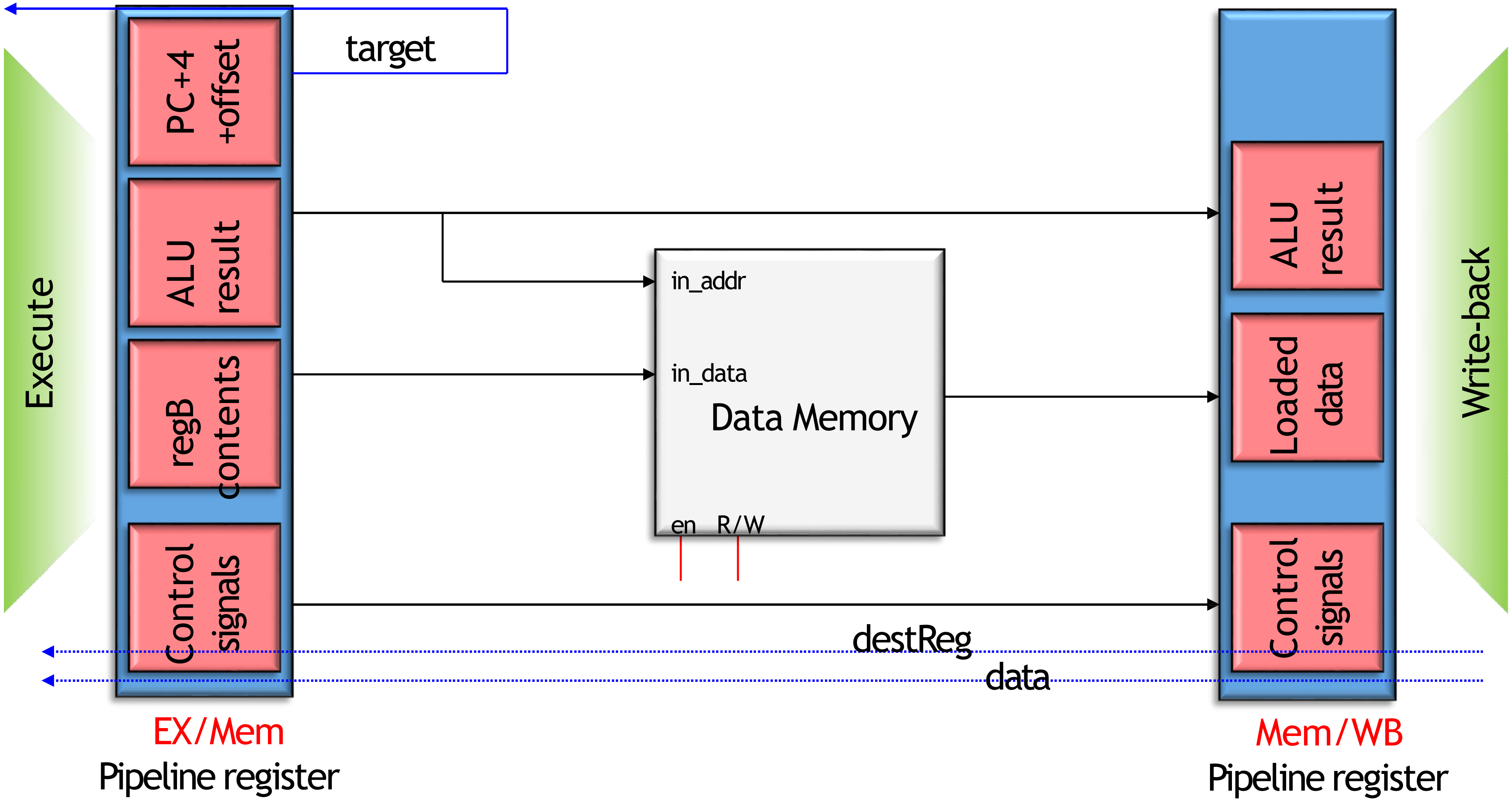
Stage 3: Execute



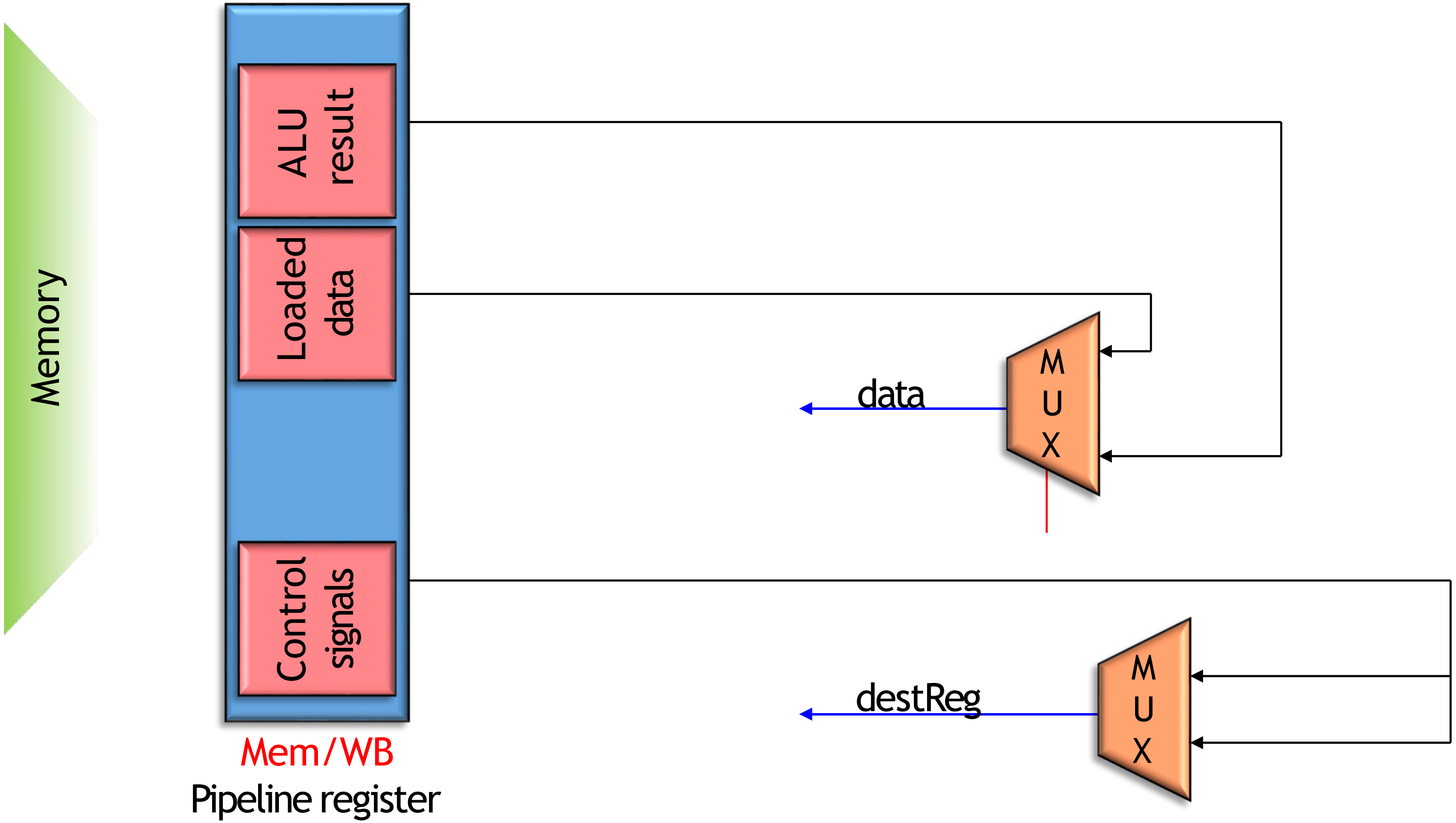
ID /EX
Pipeline register

EX/Mem
Pipeline register

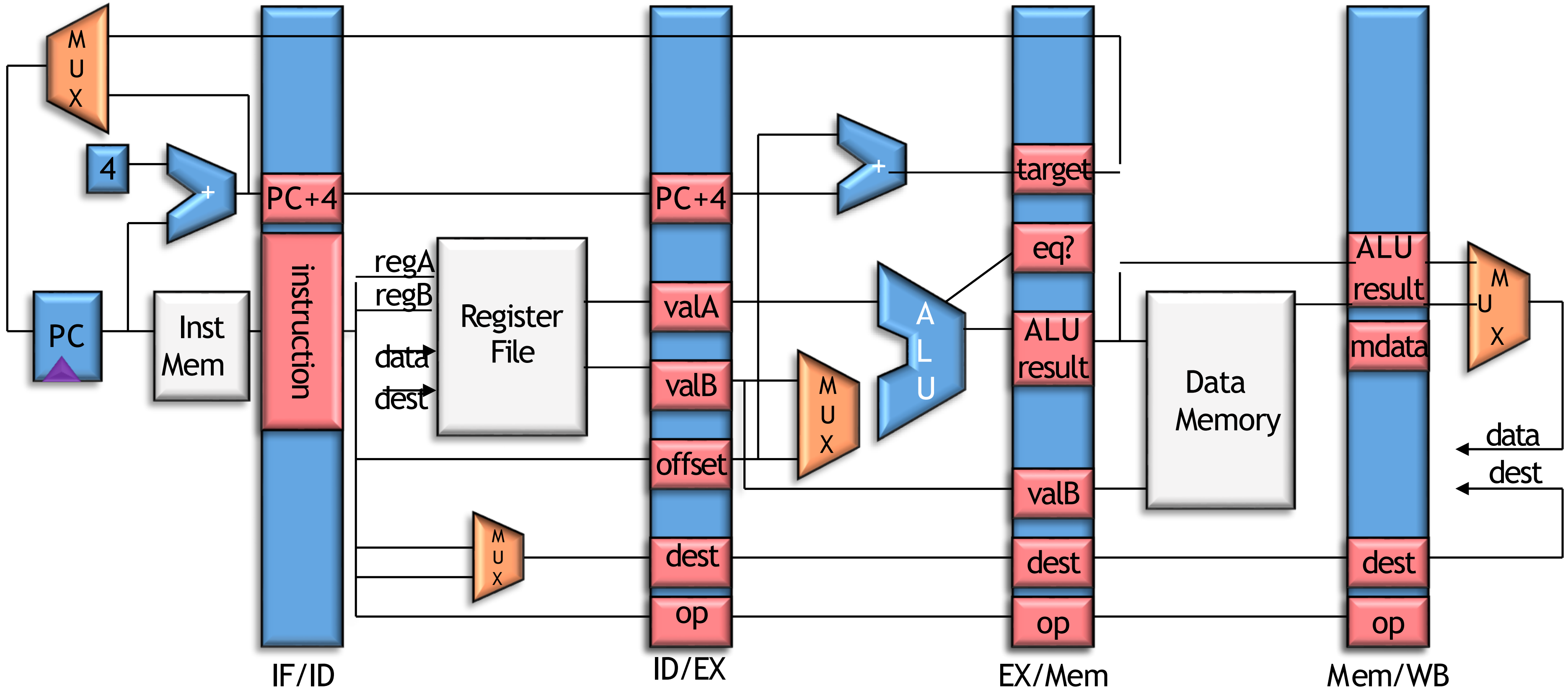
Stage 4: Memory Stage

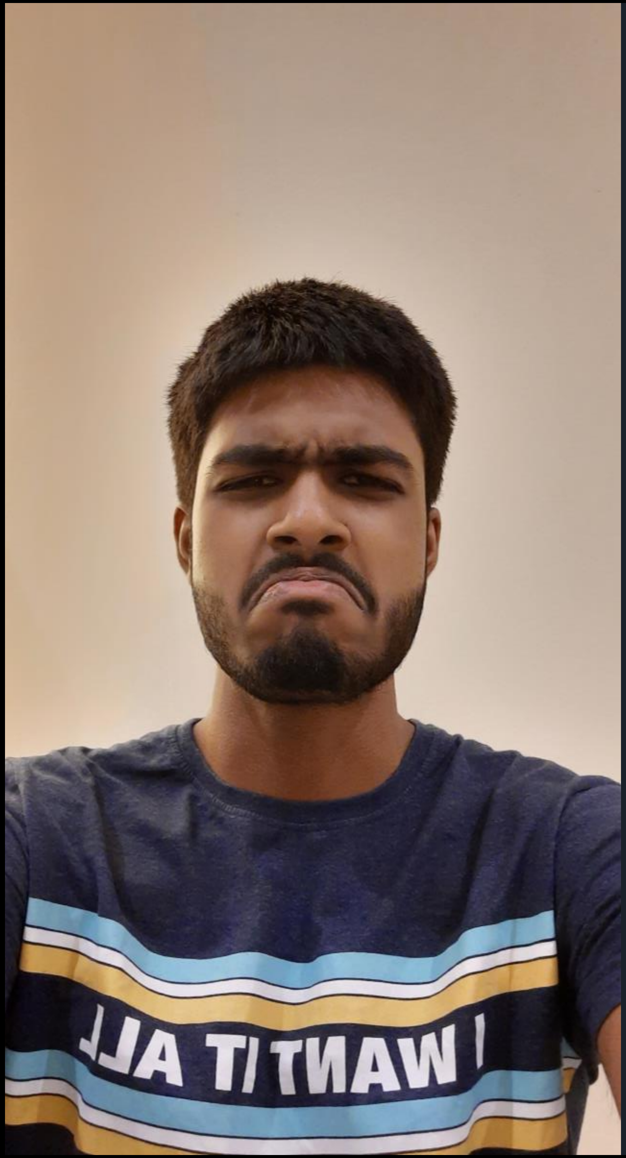


Stage 5: Write-back



The Complete Picture





Coffee Credits

Yashwant: +4

Arnav: +5

Carpe Diem