Pipeline

Complete Revision

Part 2

- Deepak Poonia

(IISc Bangalore; *GATE AIR 53 & 67*)

Content of this Lecture:

1. Data Dependency & Hazards
2. Control Dependencies
3. GATE PYQs

GATE 2024 Complete Course Link:

https://www.goclasses.in/s/pages/gatecompletecourse

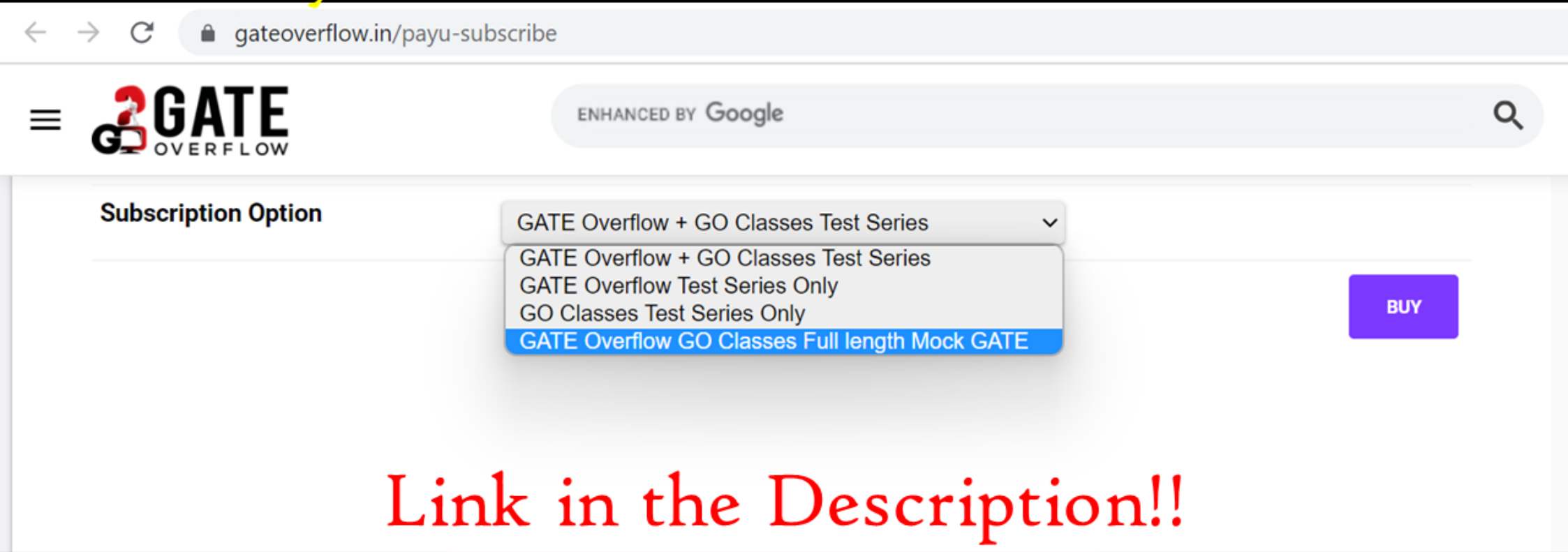
Instructor:

Deepak Poonia

IISc Bangalore

GATE CSE AIR 53; AIR 67; AIR 206; AIR 256

Subscribe for ALL 15 Mock Tests By GateOverflow + GO Classes



Link in the Description!!

GO CLASSES

GATE 2024
COMPLETE COURSE CS-IT

(1 YEAR)

GO CLASSES

GATE 2025

COMPLETE COURSE CS-IT

(2 YEARS)

2023 Discrete Mathematics

★ ★ ★ ★ ★ 5.0 (62 ratings)

Deepak Poonia (MTech IISc Bangal...

Free

2023 C Programming

★ ★ ★ ★ ★ 5.0 (59 ratings)

Sachin Mittal (MTech IISc Bangalor...

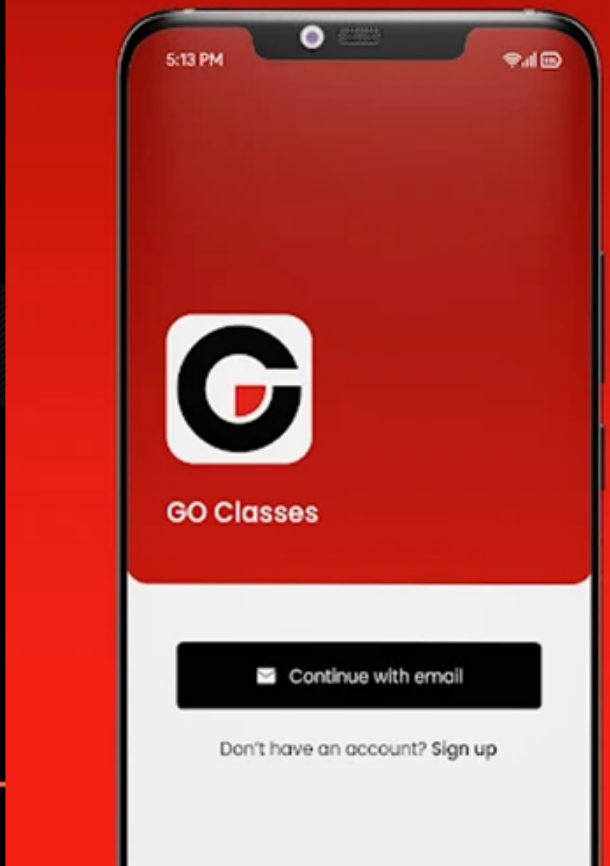
Free

On “GATE-Overflow” Website

Download the GO Classes Android App:

https://play.google.com/store/apps/details?id=com.goclasses.courses

Search "GO Classes" on Play Store.



GATE PYQs

mcq

1.18.21 Pipelining: GATE CSE 2014 Set 3 | Question: 9 top https://gateoverflow.in/2043

Consider the following processors (ns stands for nanoseconds). Assume that the pipeline registers have zero latency.

- P1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.
- P2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.
- P3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.
- P4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.

2 marks

Which processor has the highest peak clock frequency?

A. $P1$
B. $P2$
C. $P3$
D. $P4$

silly mistake

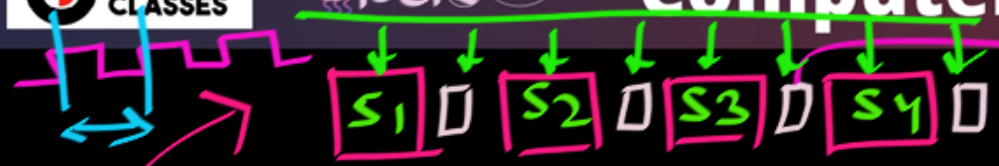
55 marks

Vs

57.66 marks

NIT

IIT



1.18.21 Pipelining: GATE CSE 2014 Set 3 | Question: 9

top

https://gateoverflow.in/2043

Consider the following processors (ns stands for nanoseconds). Assume that the pipeline registers have zero latency.

- P1: Four-stage pipeline with stage latencies 1 ns, 2 ns, 2 ns, 1 ns.
- P2: Four-stage pipeline with stage latencies 1 ns, 1.5 ns, 1.5 ns, 1.5 ns.
- P3: Five-stage pipeline with stage latencies 0.5 ns, 1 ns, 1 ns, 0.6 ns, 1 ns.
- P4: Five-stage pipeline with stage latencies 0.5 ns, 0.5 ns, 1 ns, 1 ns, 1.1 ns.

Which processor has the highest peak clock frequency?

A. P1
B. P2
C. P3
D. P4

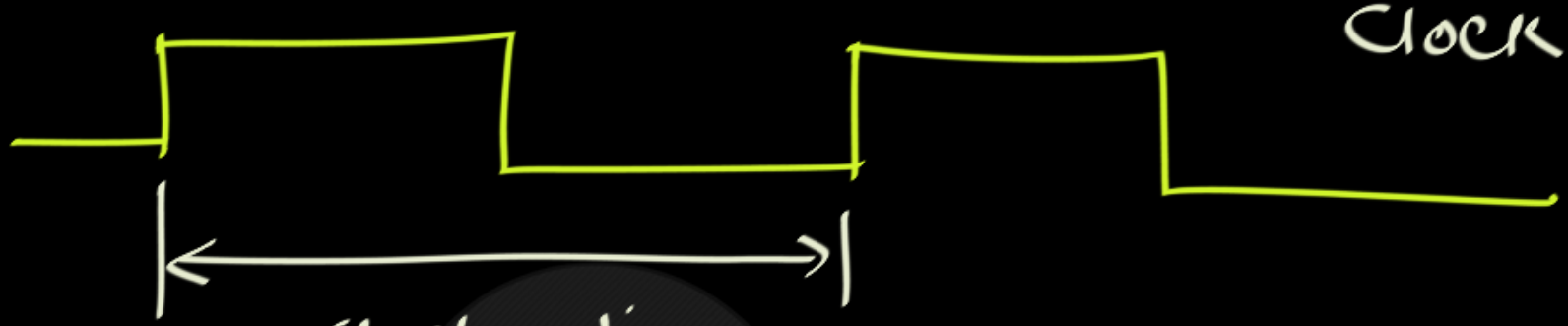
silly mistake

max (stage delay + Register Delay)

cycle time:

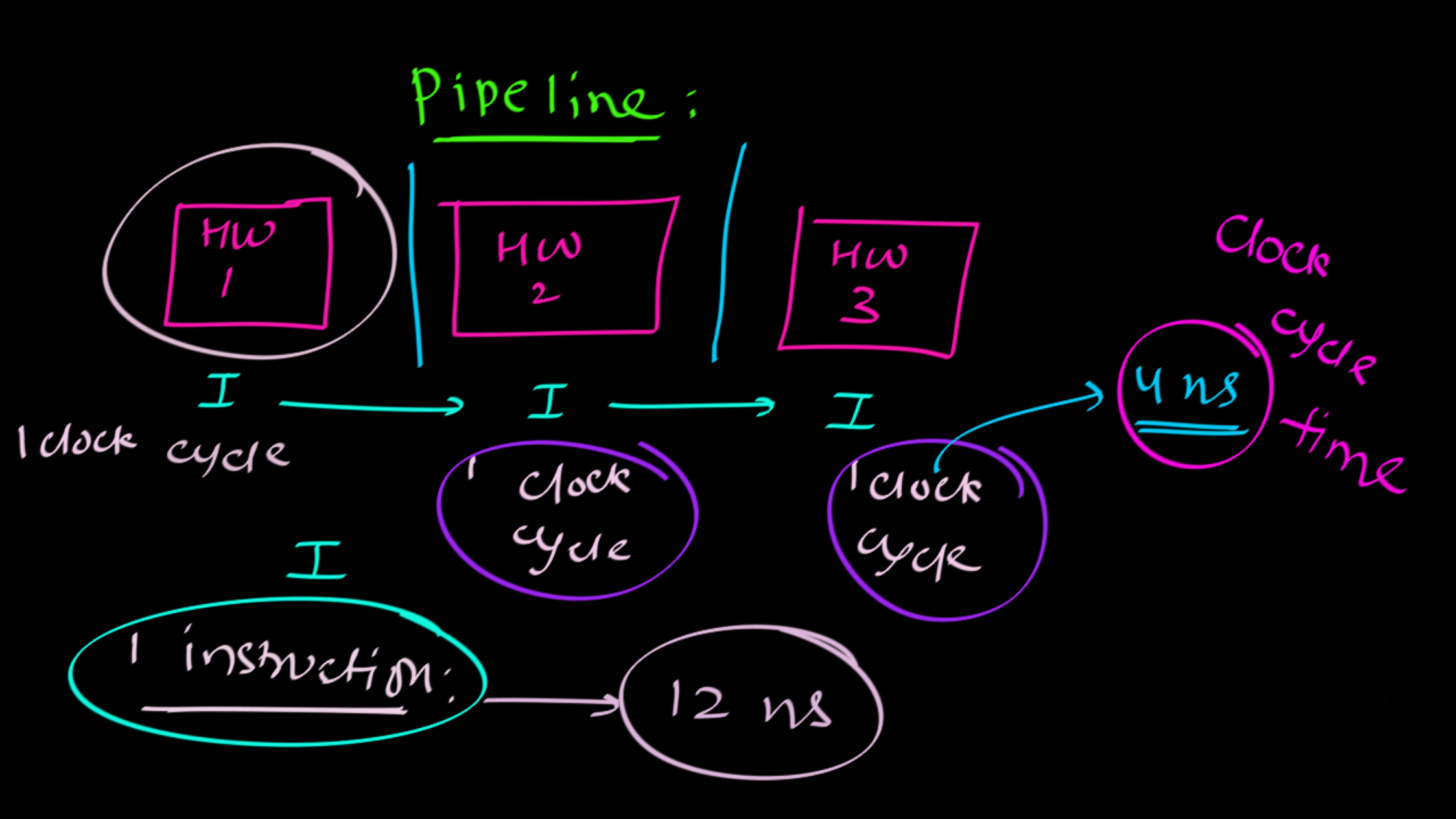
each stage Completes its work.

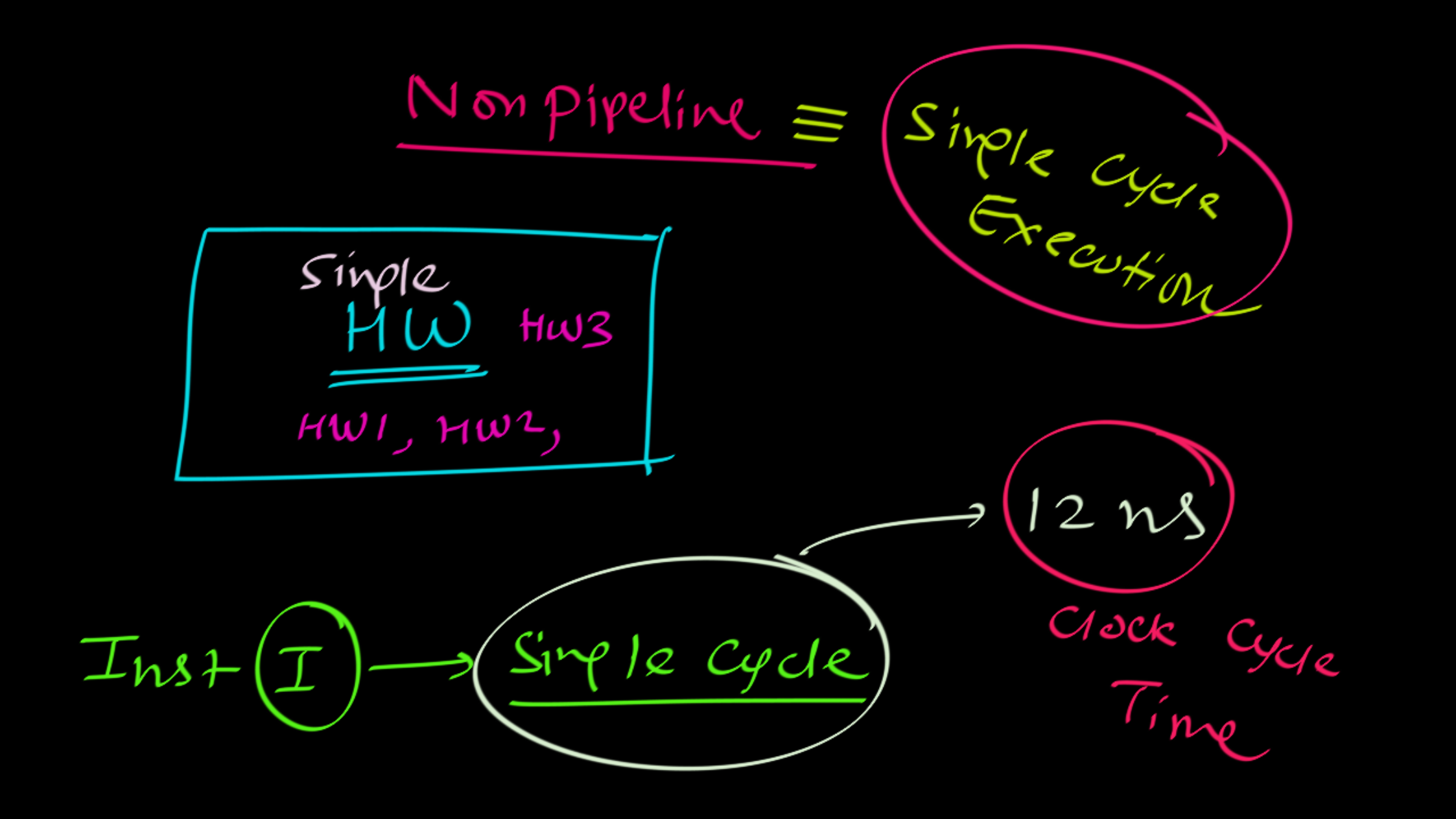
P_1: 2 ns
P_2: 1.5 ns
P_3: 1 ns
P_4: 1.1 ns



Cycle time in Pipelined Processor = max (Stage Delays + Register Delay)

$$\text{frequency} = \frac{1}{\text{cycle time}}$$


Pipeline:
HW 1
HW 2
HW 3
I
I
I
1 clock cycle
1 clock cycle
1 clock cycle
4 ns
Clock cycle time
I
1 instruction:
12 ns


Non pipeline ≡ Simple cycle Execution
Simple HW
HW3
HW1, HW2,
12 ns
Clock cycle Time
Inst I
Simple cycle

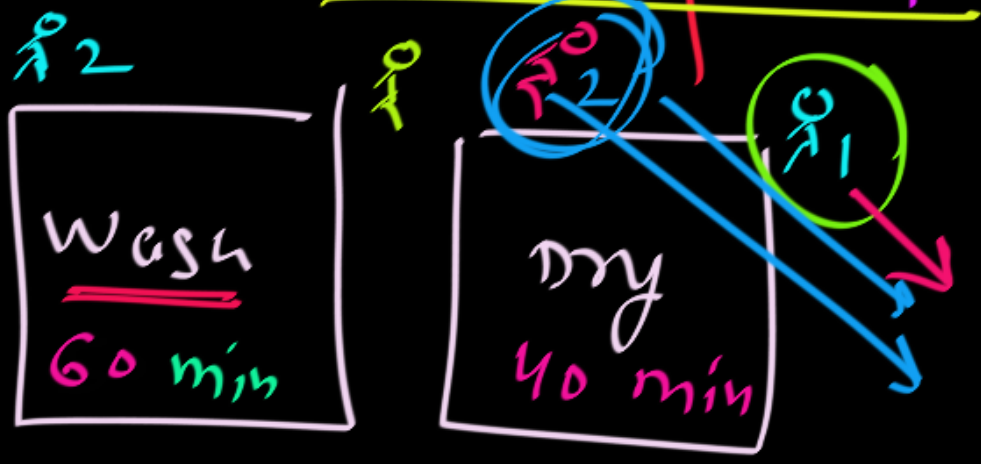
Home Washing m/c (IISc)

90 minutes

1 person : 90 min

100 people : 90 x 100 min

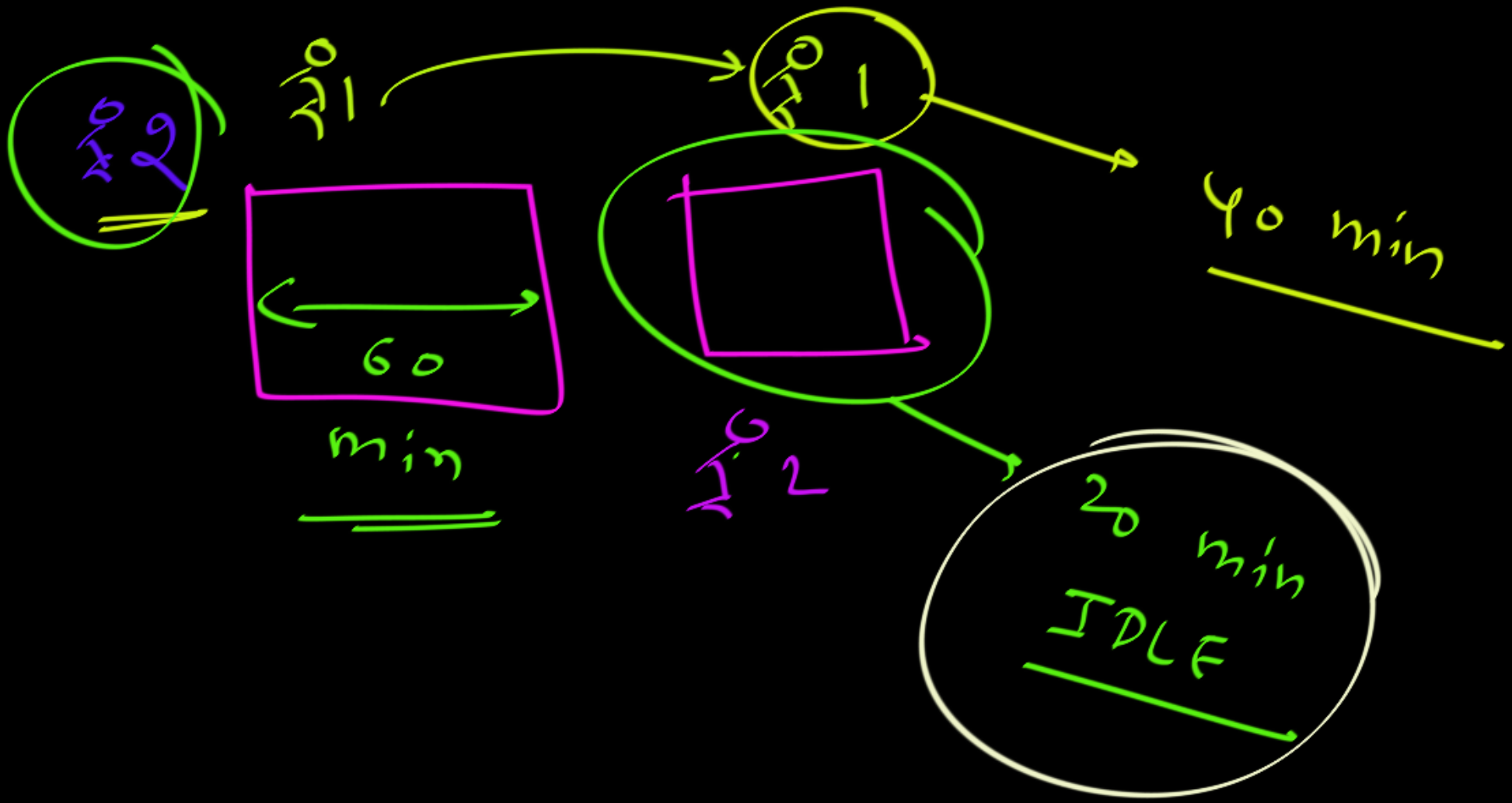
IISc Washing shop



1 person : → 100 min

100 people : (2 + 99) 60 min

>>


x0
x1
x0
1
40 min
60
min
x2
20 min
IDLE

1.18.2 Pipelining: GATE CSE 2000 | Question: 1.8 top

https://gateoverflow.in/631

Comparing the time T1 taken for a single instruction on a pipelined CPU with time T2 taken on a non-pipelined but identical CPU, we can say that

A. $T1 \leq T2$
B. $T1 \geq T2$
C. $T1 < T2$
D. T1 and T2 plus the time taken for one instruction fetch cycle

No Concept of Stages

1.18.2 Pipelining: GATE CSE 2000 | Question: 1.8 top

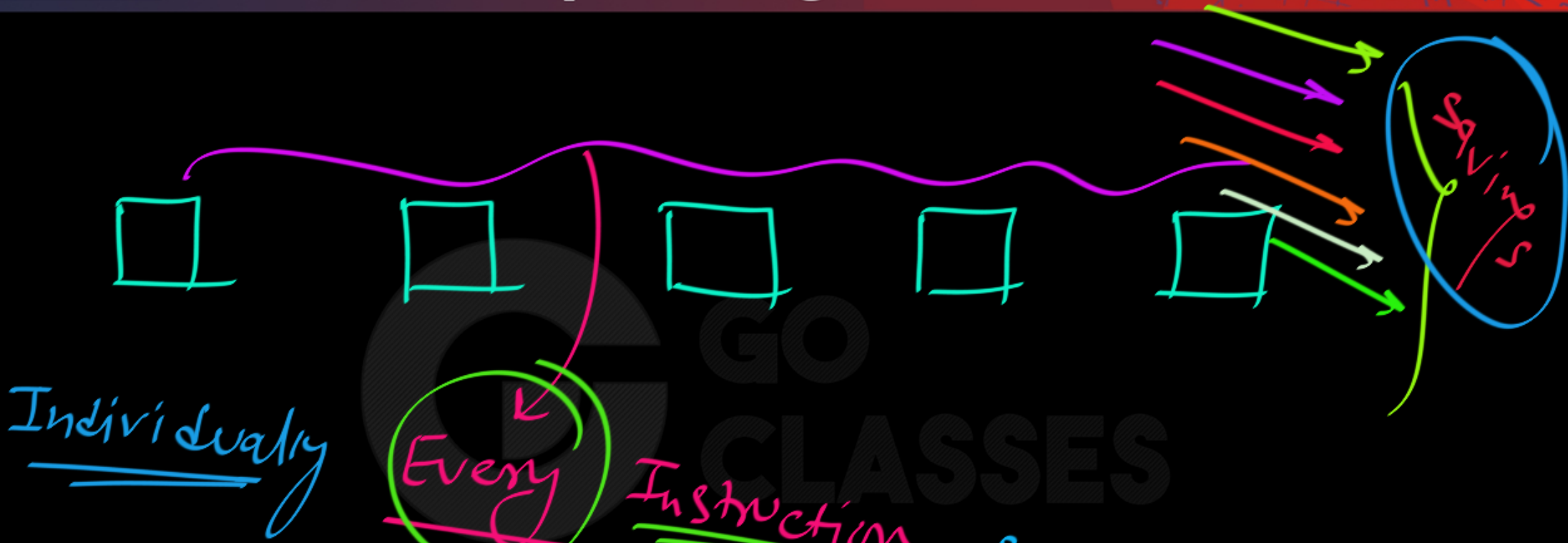
https://gateoverflow.in/631

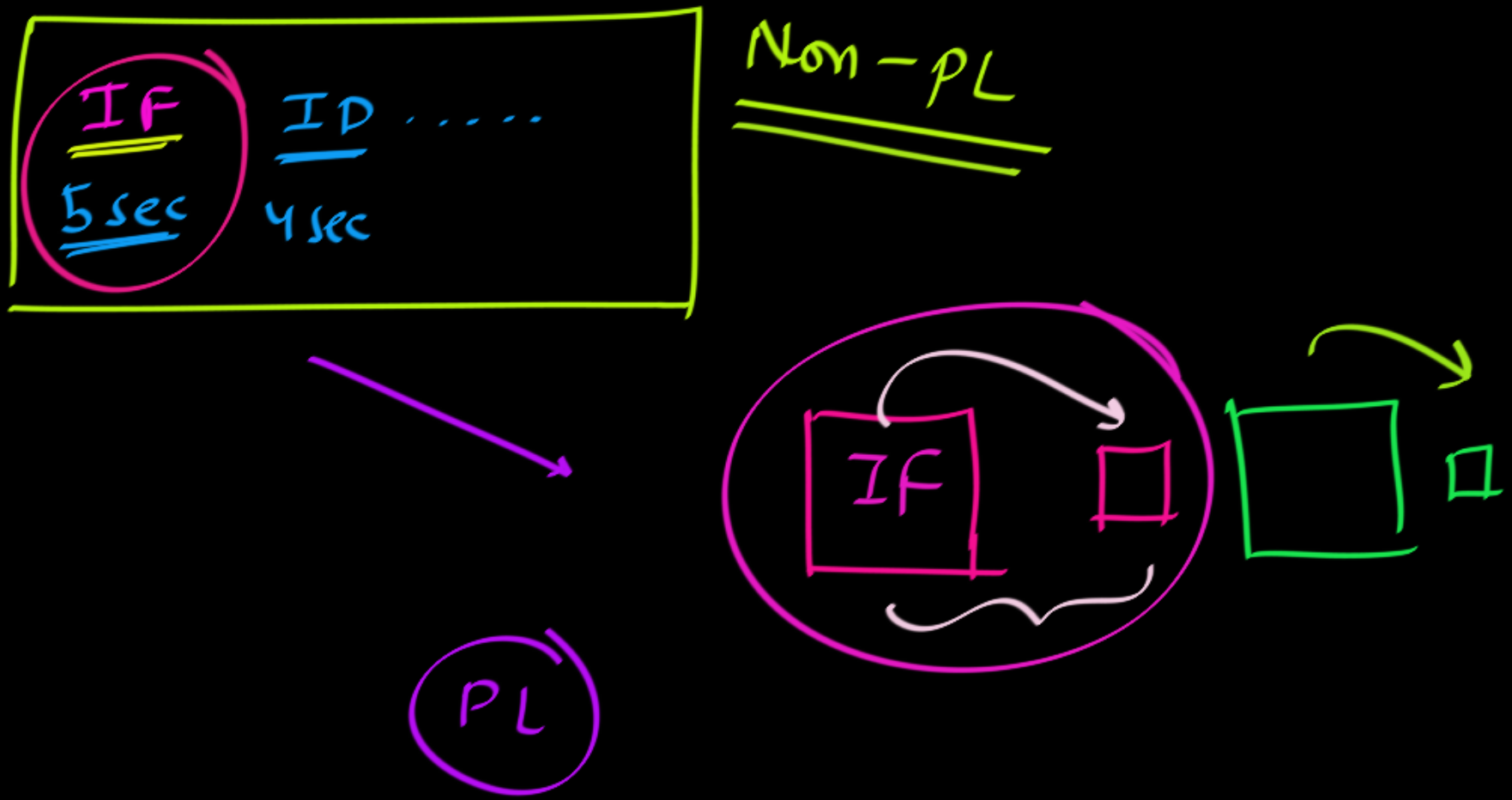
Comparing the time T1 taken for a single instruction on a pipelined CPU with time T2 taken on a non-pipelined but identical CPU, we can say that

A. $T1 \leq T2$
B. $T1 \geq T2$
C. $T1 < T2$
D. T1 and T2 plus the time taken for one instruction fetch cycle

(1) Inter stage buffer → Some Delay

$T_1 \geq T_2$

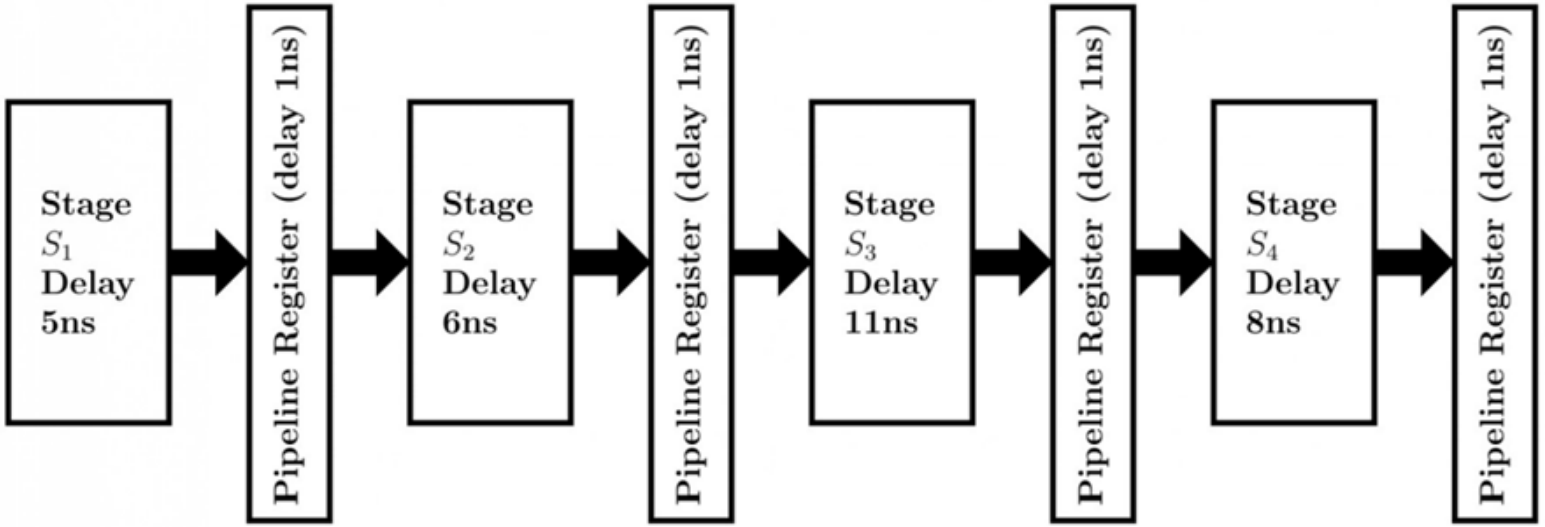

Individually
Every
Instruction
goes through
all stages.


IF
ID .....
5 sec
4 sec
Non-PL
IF
PL

1.18.16 Pipelining: GATE CSE 2011 | Question: 41 top

https://gateoverflow.in/2143

Consider an instruction pipeline with four stages (S1, S2, S3 and S4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays for the stages and for the pipeline registers are as given in the figure.



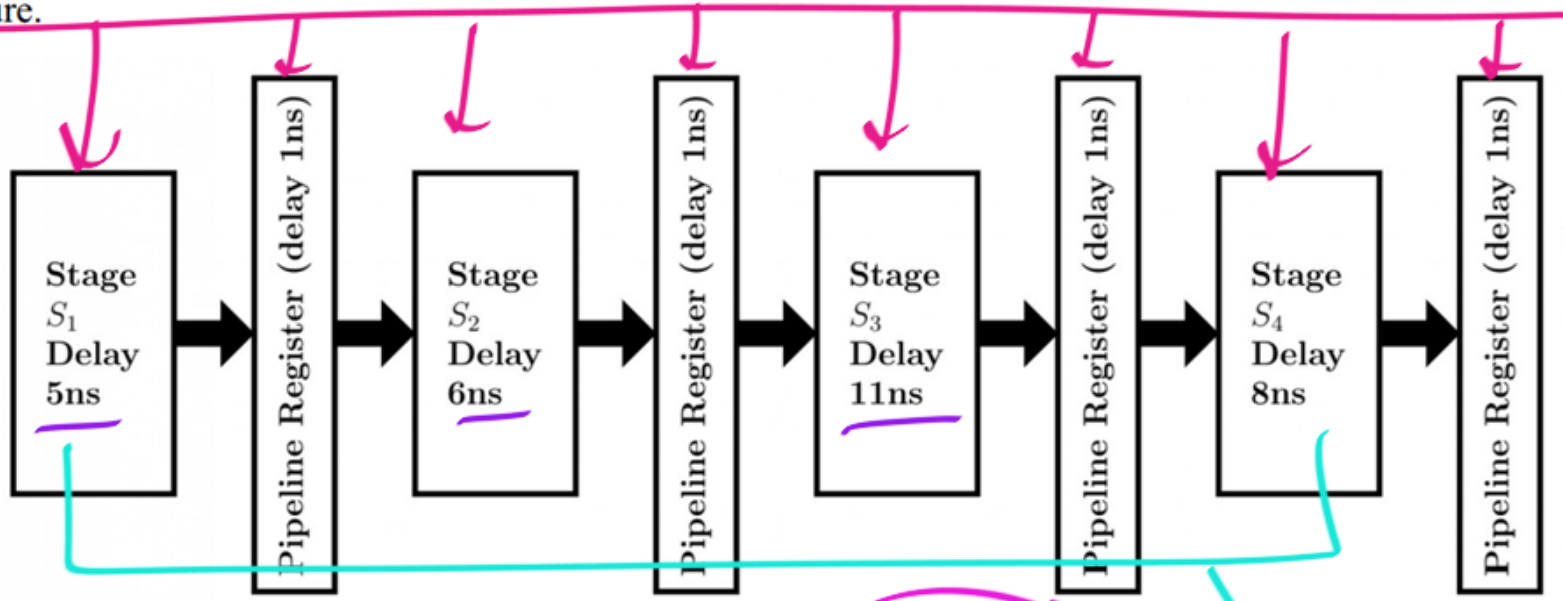
What is the approximate speed up of the pipeline in steady state under ideal conditions when compared to the corresponding non-pipeline implementation?

A. 4.0
B. 2.5
C. 1.1
D. 3.0

1.18.16 Pipelining: GATE CSE 2011 | Question: 41 top

https://gateoverflow.in/2143

Consider an instruction pipeline with four stages (S1, S2, S3 and S4) each with combinational circuit only. The pipeline registers are required between each stage and at the end of the last stage. Delays for the stages and for the pipeline registers are as given in the figure.



Clock Cycle Time : 11+1 = 12

What is the approximate speed up of the pipeline in steady state under ideal conditions when compared to the corresponding non-pipeline implementation?

5+6+11+8 Non-PL

A. 4.0
B. 2.5
C. 1.1
D. 3.0

n instructions :

Pipeline :

$(4 + n-1)$ 12 ns

4 → 1st instruction (4 cycles)

n-1 → Complete per cycle After 1st instr is over.

Non-pipeline : $n * (30 ns)$

Speedup of PL over NPL:

$$= \frac{\text{NPL exc}^{n} \text{ time}}{\text{PL exc}^{n} \text{ time}}$$

$$= \frac{n \times 30 \text{ ns}}{(4 + n - 1) \times 12 \text{ ns}}$$

$\frac{30}{12} = 2.5$

n large

1.18.22 Pipelining: GATE CSE 2015 Set 1 | Question: 38 top

https://gateoverflow.in/8288

Consider a non-pipelined processor with a clock rate of 2.5 GHz and average cycles per instruction of four. The same processor is upgraded to a pipelined processor with five stages; but due to the internal pipeline delay, the clock speed is reduced to 2 GHz. Assume that there are no stalls in the pipeline. The speedup achieved in this pipelined processor is_____.

gate2015-cse-set1 | co-and-architecture | pipelining | normal | numerical-answers

Non-PL :

Cycle time : $\frac{1}{2.5\ GHz} = \frac{1}{2.5}\ n\ sec$

1.18.22 Pipelining: GATE CSE 2015 Set 1 | Question: 38 top https://gateoverflow.in/8288

Consider a non-pipelined processor with a clock rate of 2.5 GHz and average cycles per instruction of four. The same processor is upgraded to a pipelined processor with five stages; but due to the internal pipeline delay, the clock speed is reduced to 2 GHz. Assume that there are no stalls in the pipeline. The speedup achieved in this pipelined processor is _____.

gate2015-cse-set1 | co-and-architecture | pipelining | normal | numerical-answers

m instructions : $4m * \frac{1}{2.5}\ ns$ } Non-PL

PL :

Cycle time : $\frac{1}{2\text{ GHz}} = \frac{1}{2}$ nsec

m instr : $\left(\underbrace{5}_{\text{=}} + \underbrace{m-1}_{\text{=}}\right) \frac{1}{2}$ nsec

Speedup : $\frac{4m}{2.5} \Big/ (5+m-1)/2 = \frac{4 \times 2}{2.5} = \frac{8}{2.5}$

m large

$$\frac{80}{25} = 3.2$$

1.18.25 Pipelining: GATE CSE 2016 Set 1 | Question: 32 top

https://gateoverflow.in/39691

The stage delays in a 4-stage pipeline are 800, 500, 400 and 300 picoseconds. The first stage (with delay 800 picoseconds) is replaced with a functionality equivalent design involving two stages with respective delays 600 and 350 picoseconds. The throughput increase of the pipeline is _____ percent.

gate2016-cse-set1 co-and-architecture pipelining normal numerical-answers

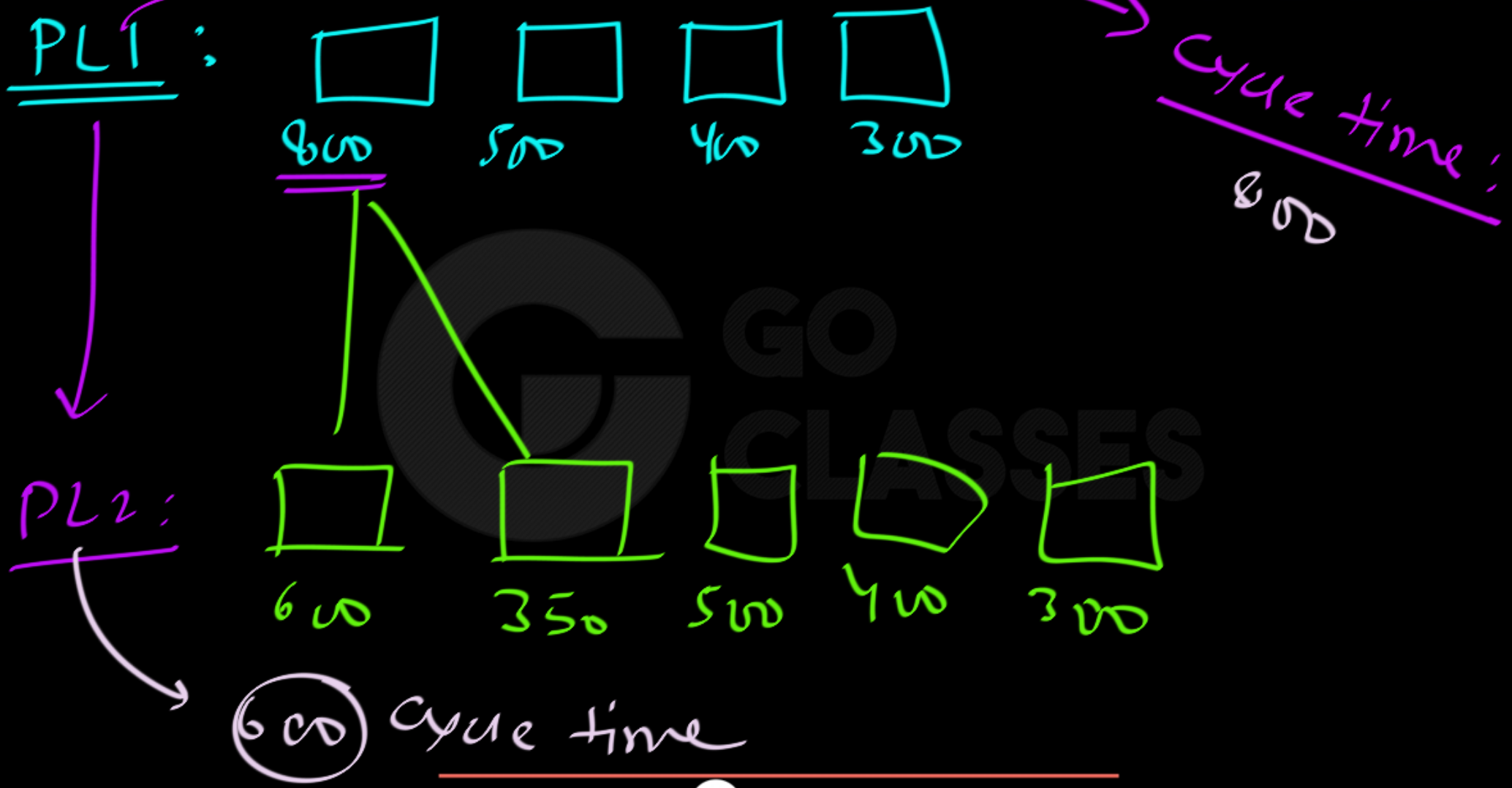
1.18.25 Pipelining: GATE CSE 2016 Set 1 | Question: 32 top

https://gateoverflow.in/39691

The stage delays in a 4-stage pipeline are 800, 500, 400 and 300 picoseconds. The first stage (with delay 800 picoseconds) is replaced with a functionality equivalent design involving two stages with respective delays 600 and 350 picoseconds. The throughput increase of the pipeline is _____ percent.

gate2016-cse-set1 | co-and-architecture | pipelining | normal | numerical-answers

33.33


PL1 :
800
500
400
300
cycle time :
800
PL2 :
600
350
500
400
300
600 cycle time

n instructions:

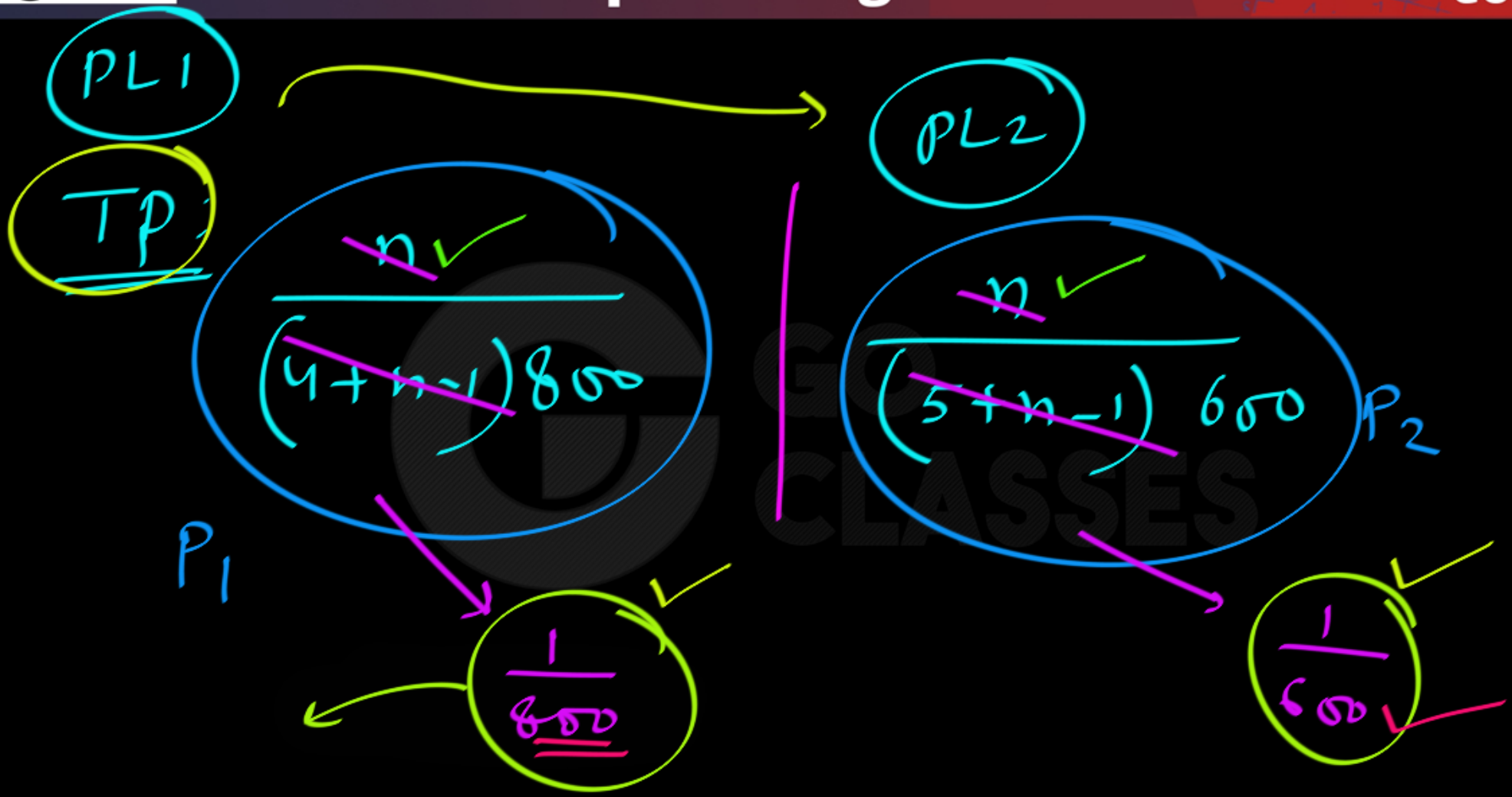
PL1: $\longrightarrow$ $(4 + n - 1)$ 800 ps $\longrightarrow$ n inst.

Throughput: 1 ps $\longrightarrow$ $\frac{n}{(4 + n - 1)\ 800}$ inst.

n instructions:

PL2 ⟶ $(5 + n - 1)$ 600 ps ⟶ n inst.

Throughput: 1 ps ⟶ $\frac{n}{(5 + n - 1)\ 600}$ inst.


PL1
TP
PL2
n
(4+n-1)800
n
(5+n-1) 600
P1
P2
1
800
1
600

Ans :

$$\frac{P_2 - P_1}{P_1} \times 100$$

$$= \frac{\frac{1}{600} - \frac{1}{800}}{\frac{1}{800}} \times 100$$

$$= \frac{200}{600} \times 100$$

$$= 33.33\%$$

Life is Sooo Good with Pipeline...

Life is Sooo Good with Pipeline...

But Wait...


Hazards are Coming...

Pipeline:

In this chapter you will learn about:

- Pipelining as a means for executing machine instructions concurrently
- Various hazards that cause performance degradation in pipelined processors and means for mitigating their effect
- Hardware and software implications of pipelining

Pipeline :

Unwanted Situations : (Hazards)

1. Structural Hazard
2. Data Hazard
3. Control "

} Very Imp.

Data Hazards

CPI : #cycles per instruction

(when we have a large no. of instructions)

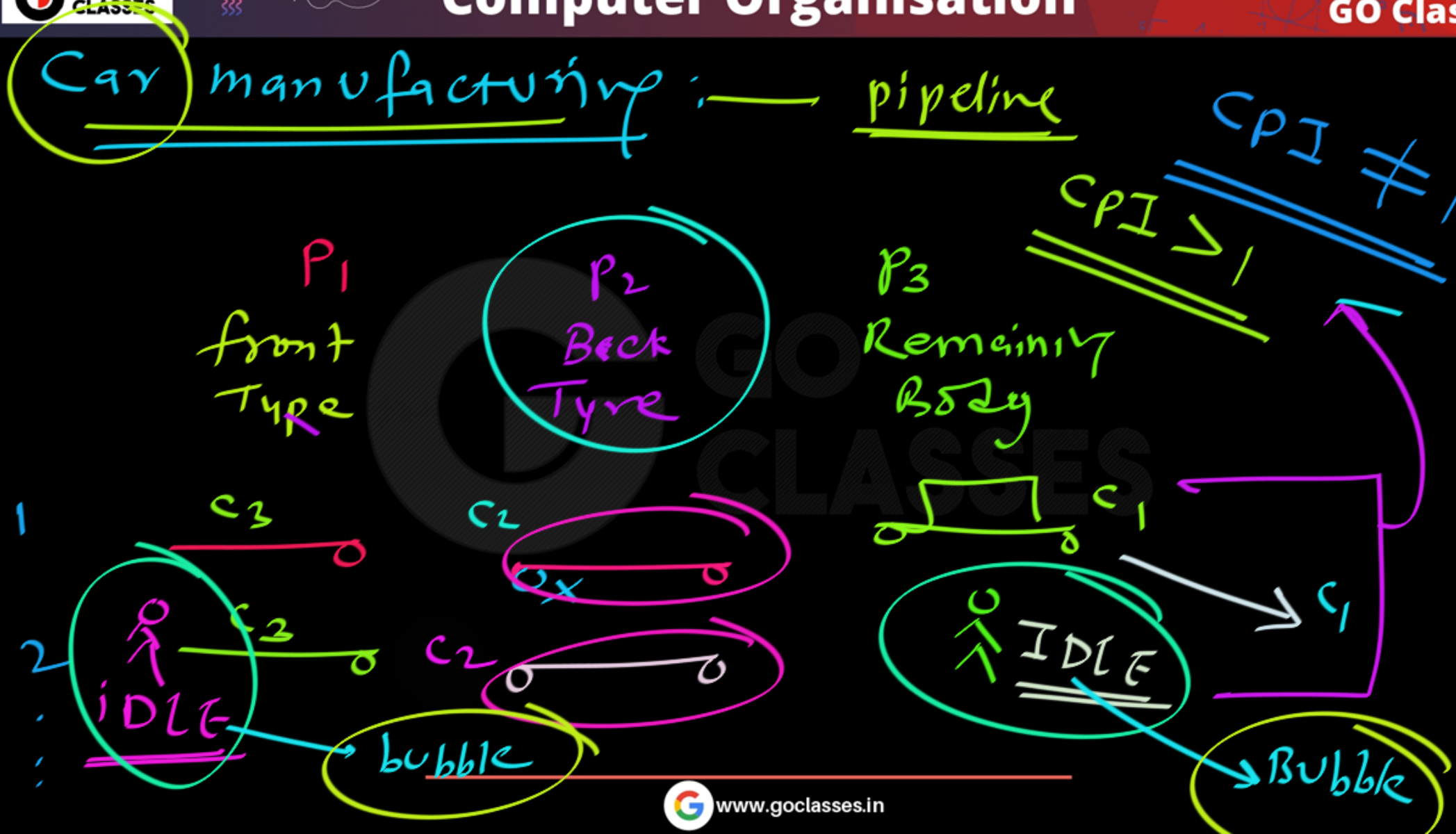
Car manufacturing :— pipeline

$C_1 \rightarrow$ 3 cycles
$C_2 \rightarrow$ 3 cycles
$C_3 \rightarrow$ 3 cycles

P_1 front Tyre

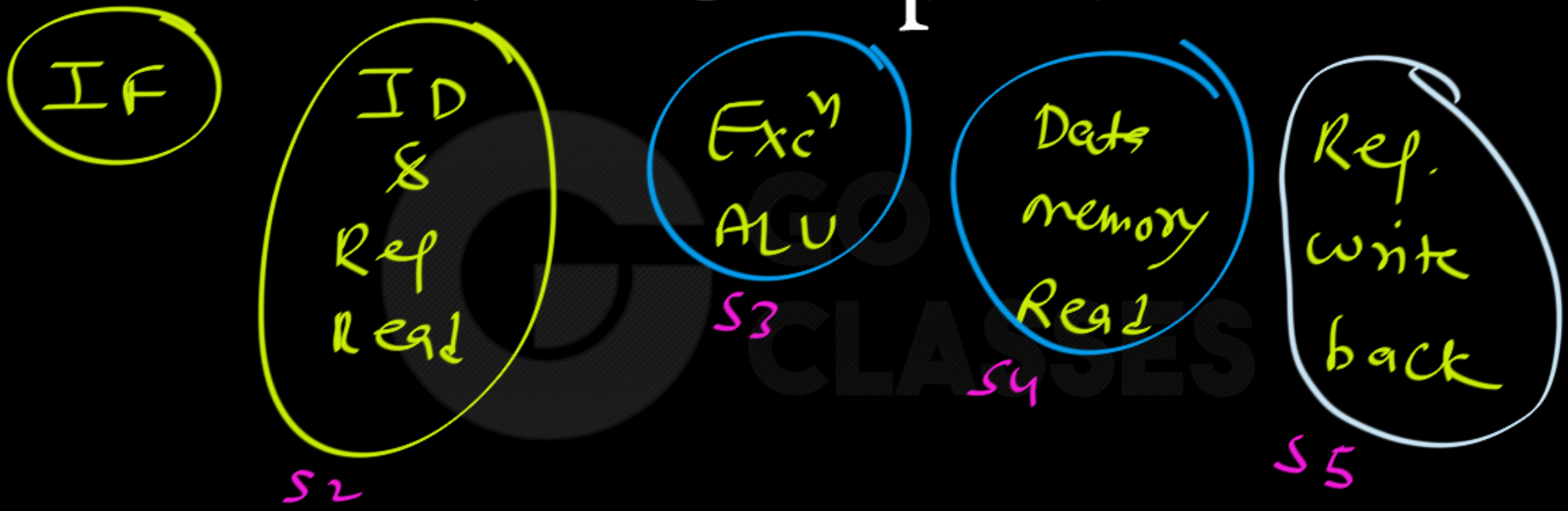
P_2 Back Tyre

P_3 Remaining Body/Chasis

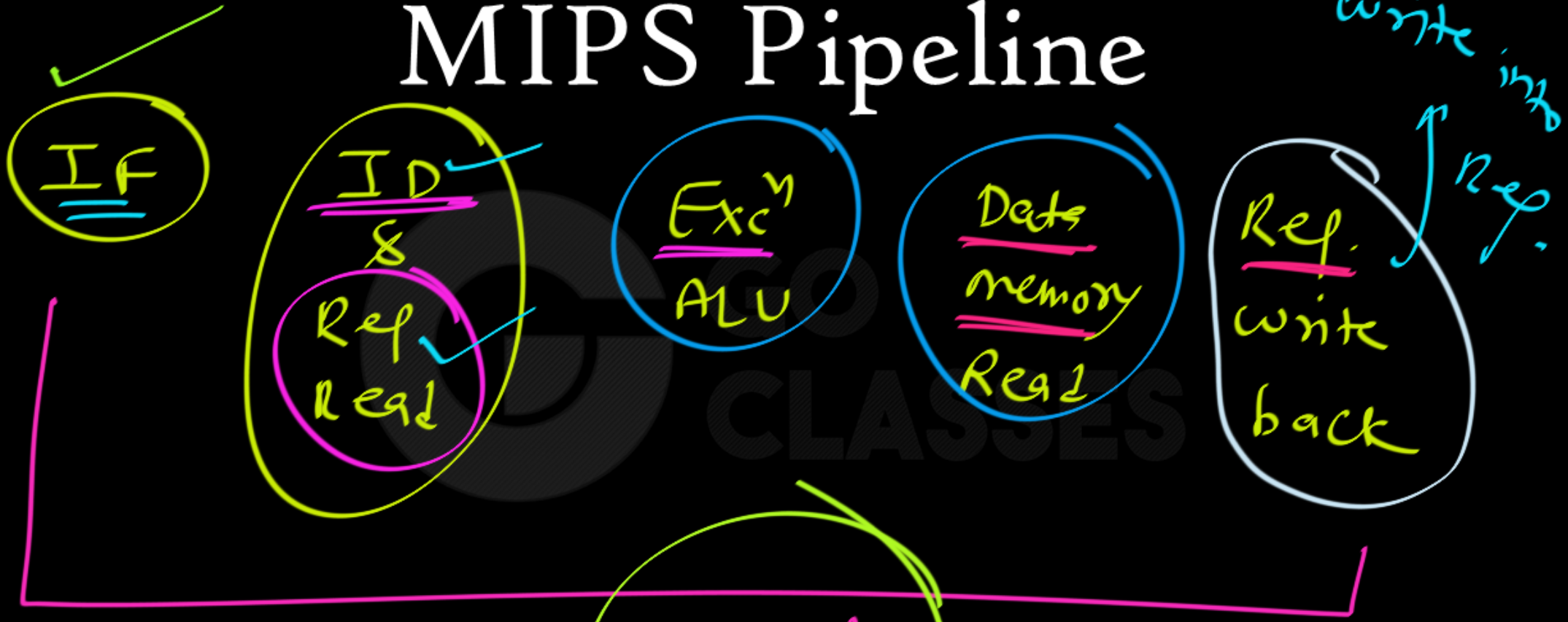

Car manufacturing :— pipeline
CPI ≠ 1
CPI > 1
P1
front Tyre
P2
Back Tyre
P3
Remaining Body
C3
C2
C1
IDLE
bubble
Bubble
www.goclasses.in


COA :
1 Hamcher
2 henessey Patterson
MIPS pipeline
RISC Processors
used in

MIPS Pipeline



MIPS Pipeline



Pipeline Generally Studies

MIPS PL ≡ RISC PL ≡ Classical 5 stage PL

≡ Standard 5 stage PL

1: lw R1, 2(R2)

2: R3 ← R1 + R4

I_1

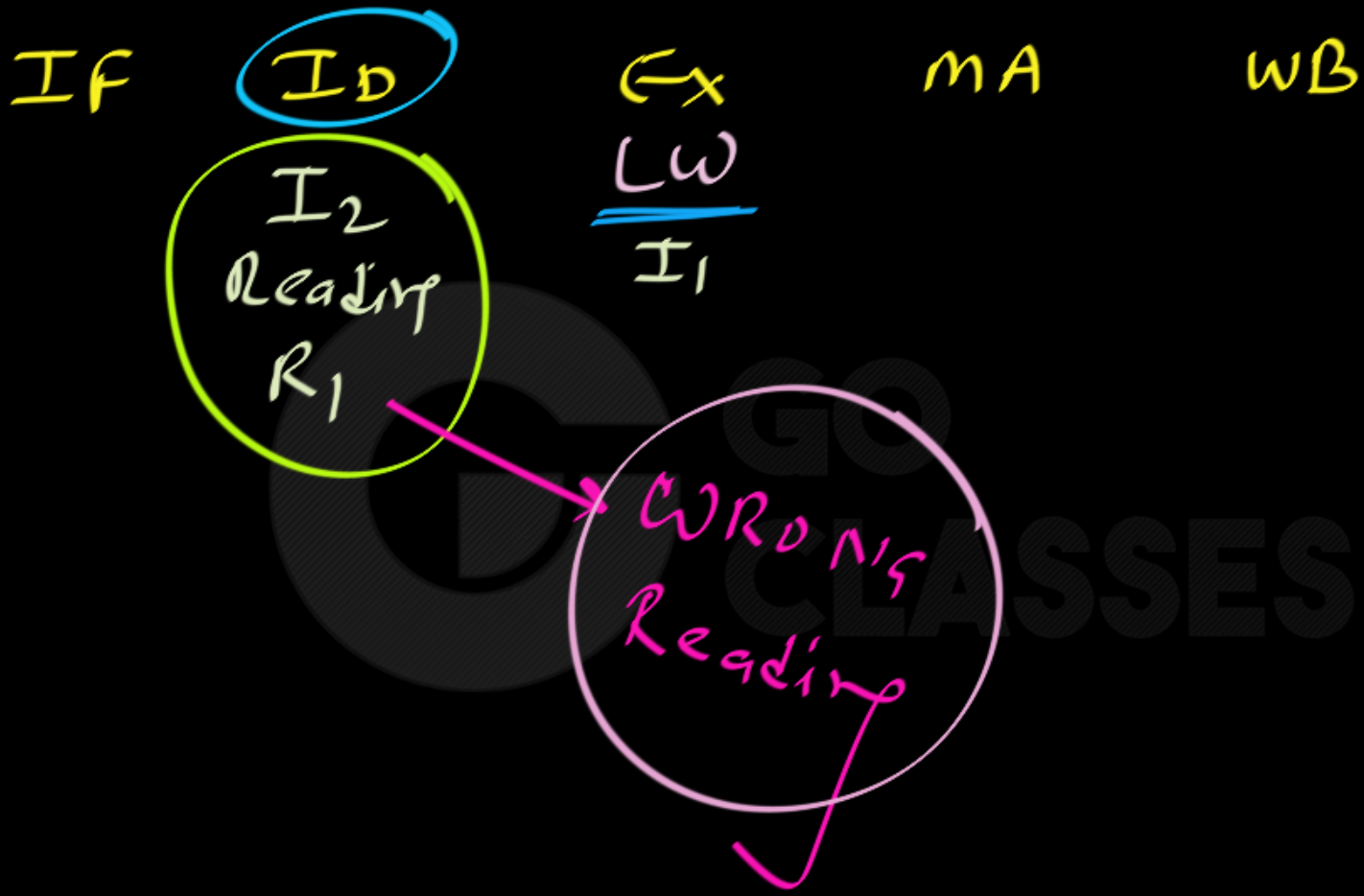
I_2

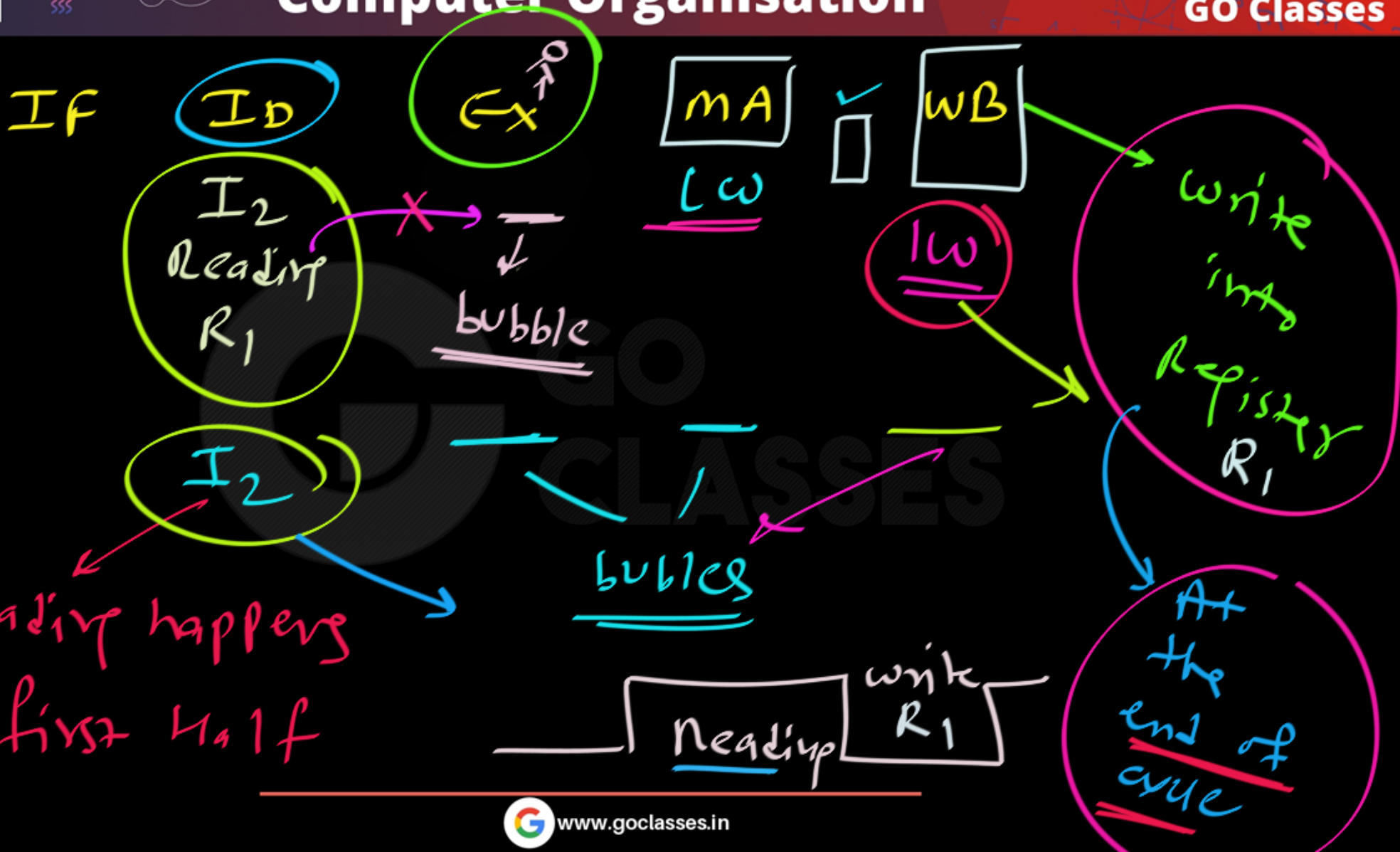
program

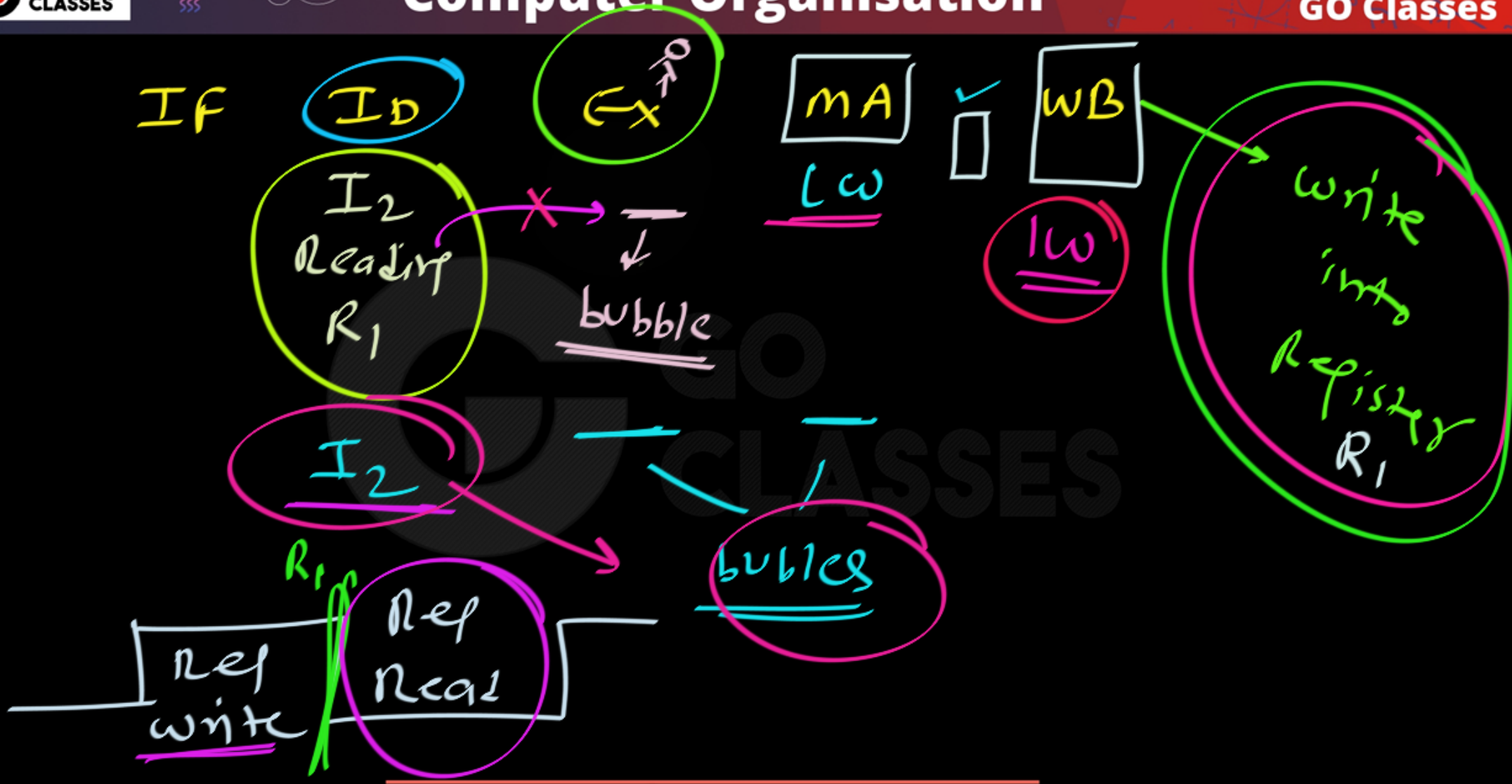
lw R1, 2(R2) means: $R_1 \leftarrow M[2+R_2]$

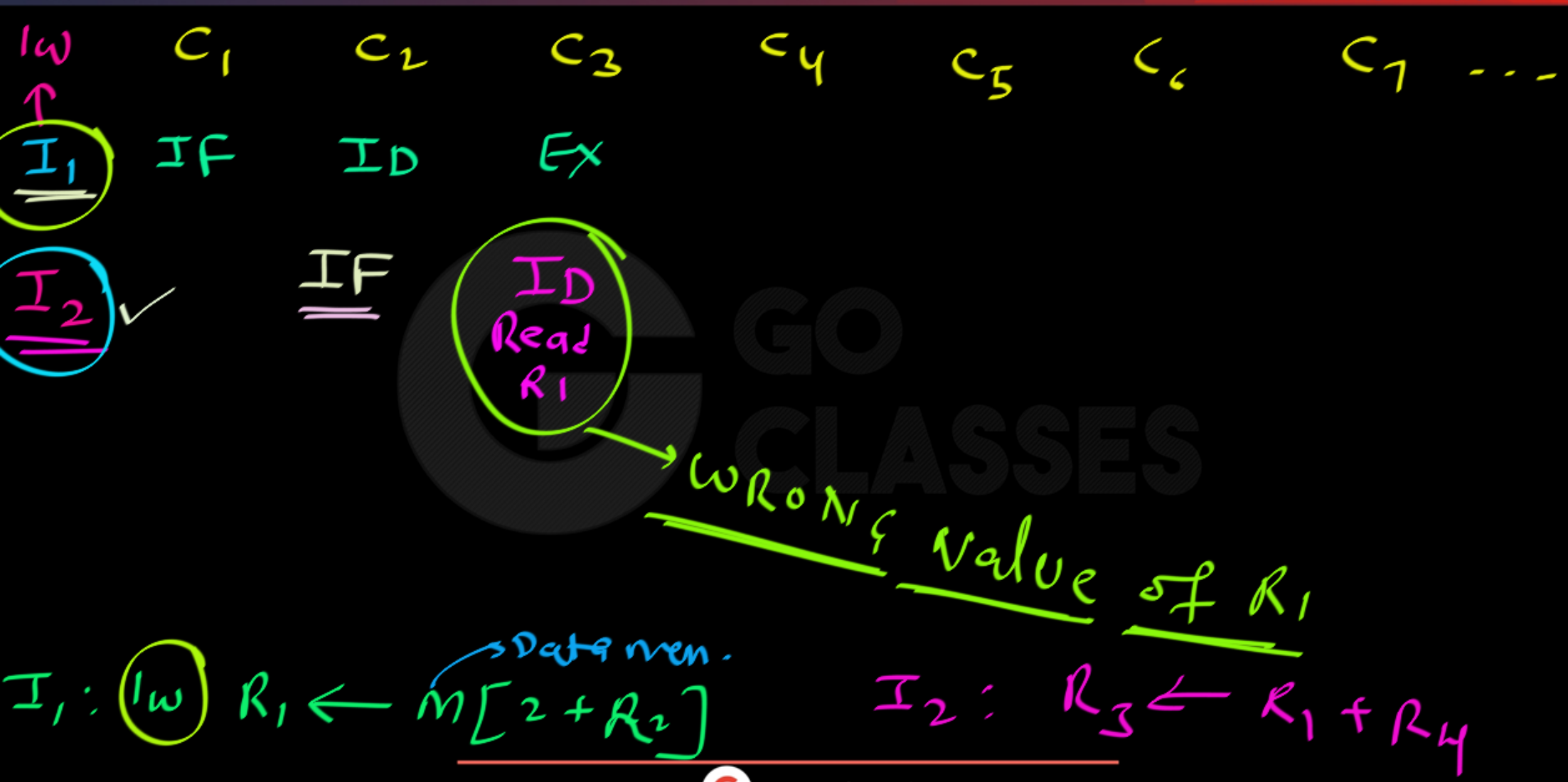
lw → load word

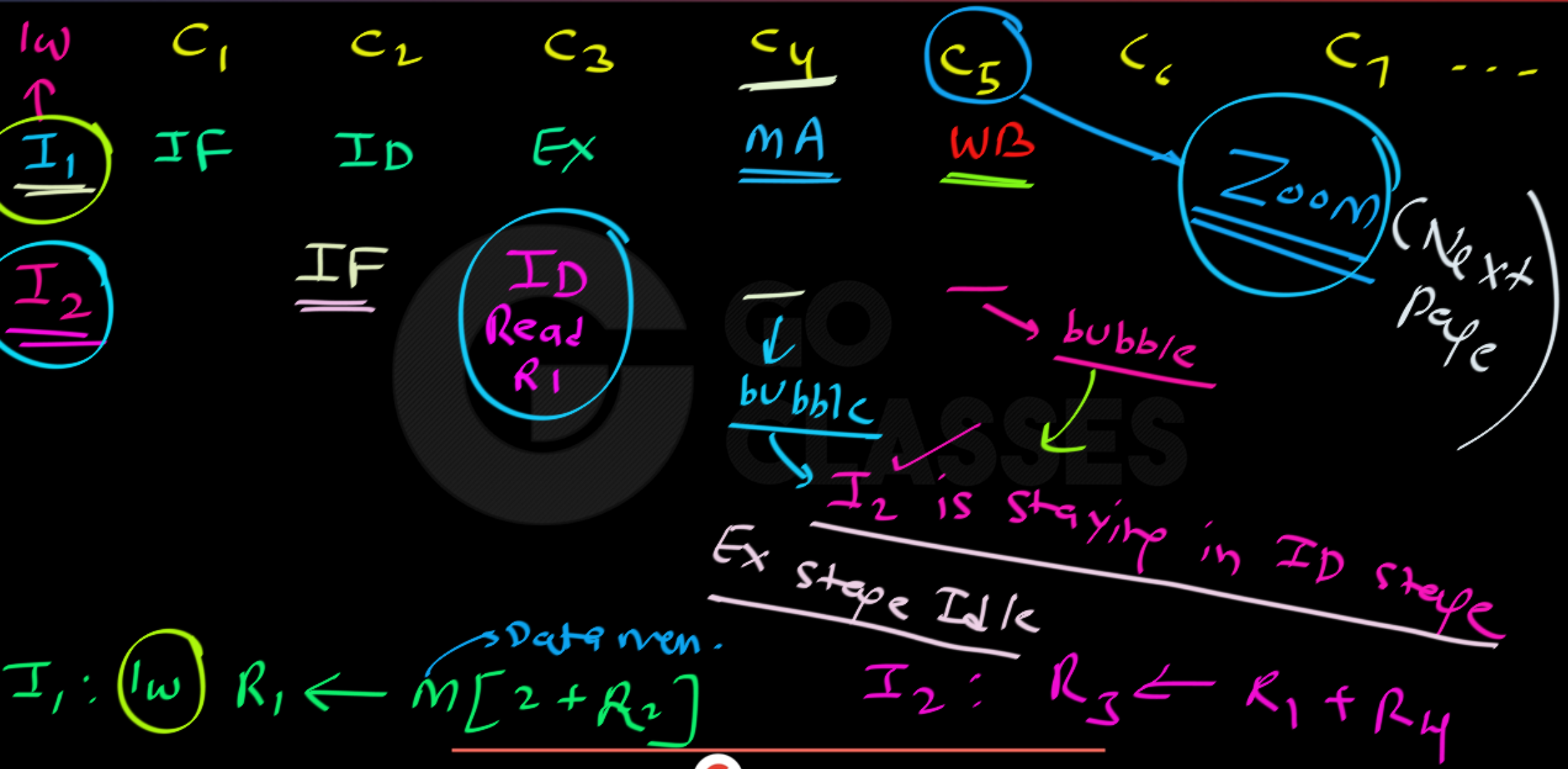
$2+R_2$ → Data location


IF
ID
EX
LW
I1
MA
WB
I2
Reading
R1
WRONG
Reading


If
ID
EX
MA
WB
I2
Reading
R1
X
bubble
lw
I2
bubbles
Reading happens
in first half
Reading
write
R1
write
into
register
R1
At
the
end of
cycle


IF
ID
EX
MA
WB
I2
Reading
R1
X
bubble
lw
lw
write
into
Register
R1
I2
bubbles
R1
Reg
write
Reg
Read

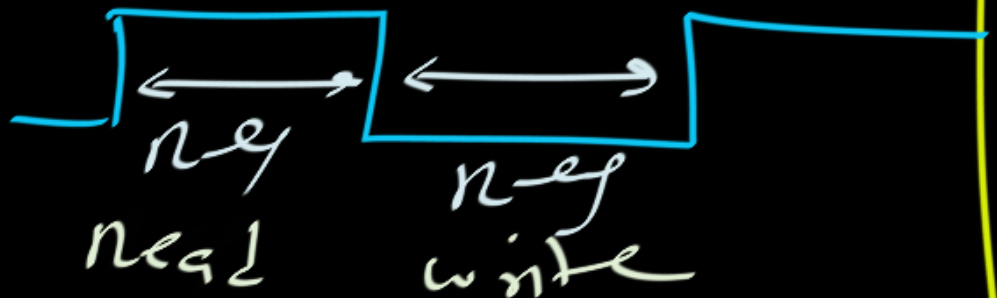

lw
C1 C2 C3 C4 C5 C6 C7 ...
I1 IF ID Ex
I2 IF
ID
Read
R1
WRONG Value of R1
I1 : lw R1 ← M[2 + R2]
Data mem.
I2 : R3 ← R1 + R4


lw
C1
C2
C3
C4
C5
C6
C7 ...
I1
IF
ID
Ex
MA
WB
Zoom (Next page)
I2
IF
ID
Read R1
bubble
bubble
I2 is staying in ID stage
Ex stage Idle
I1: lw R1 ← M[2 + R2]
Data men.
I2: R3 ← R1 + R4

Two Type of Implementations:

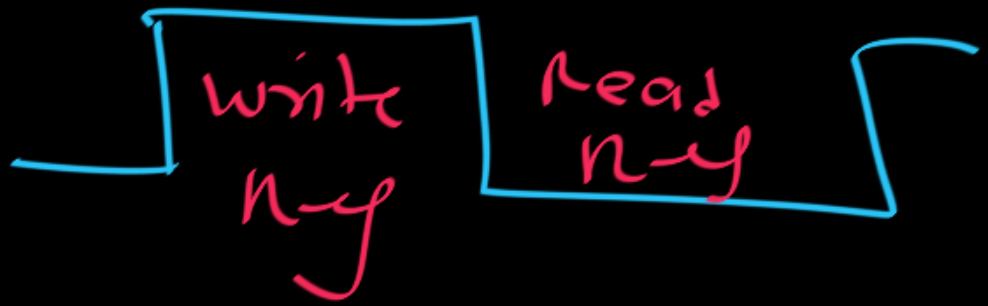
① Rep Read : 1st Half of cycle

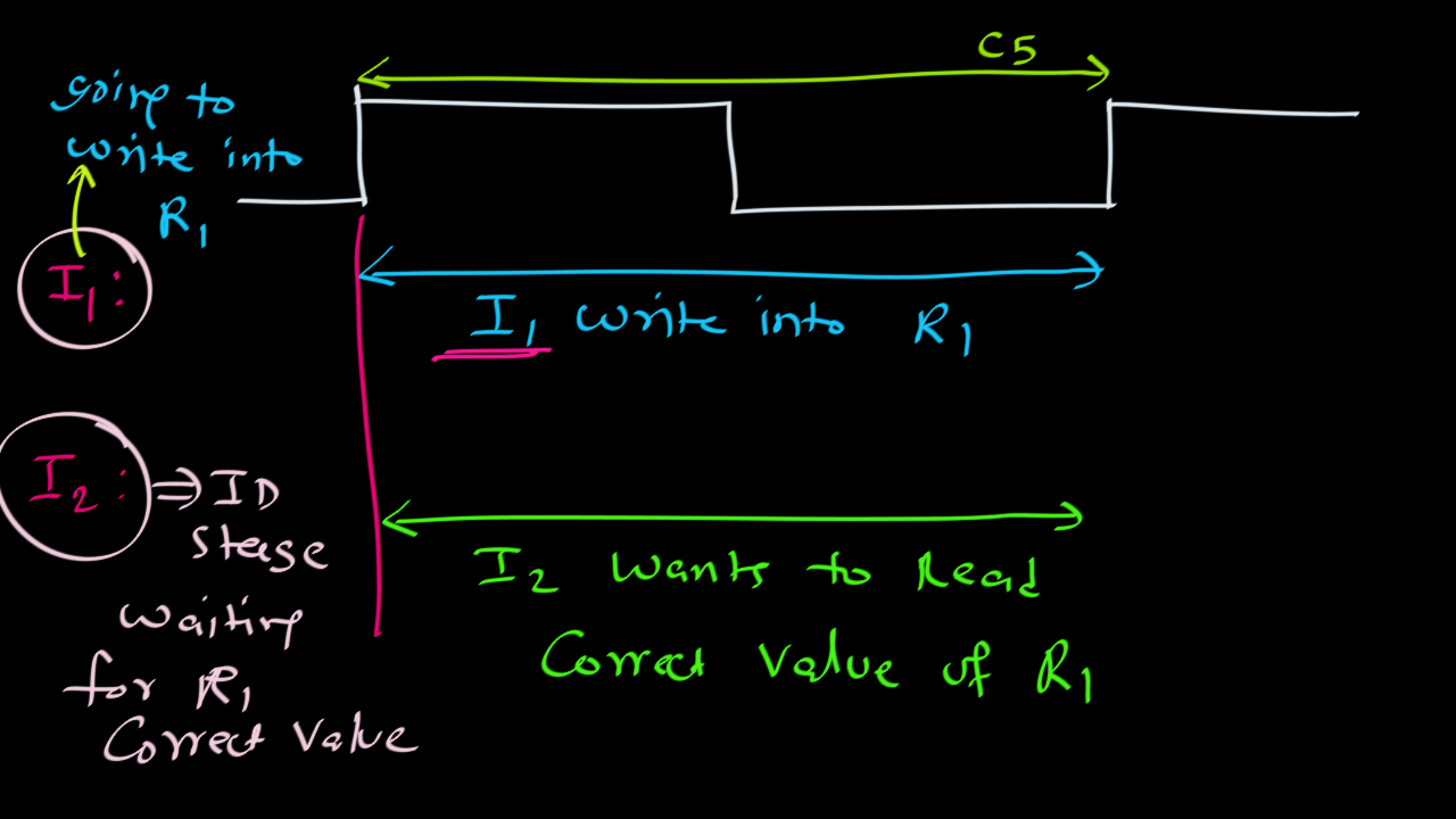
Rep write : 2nd Half of cycle

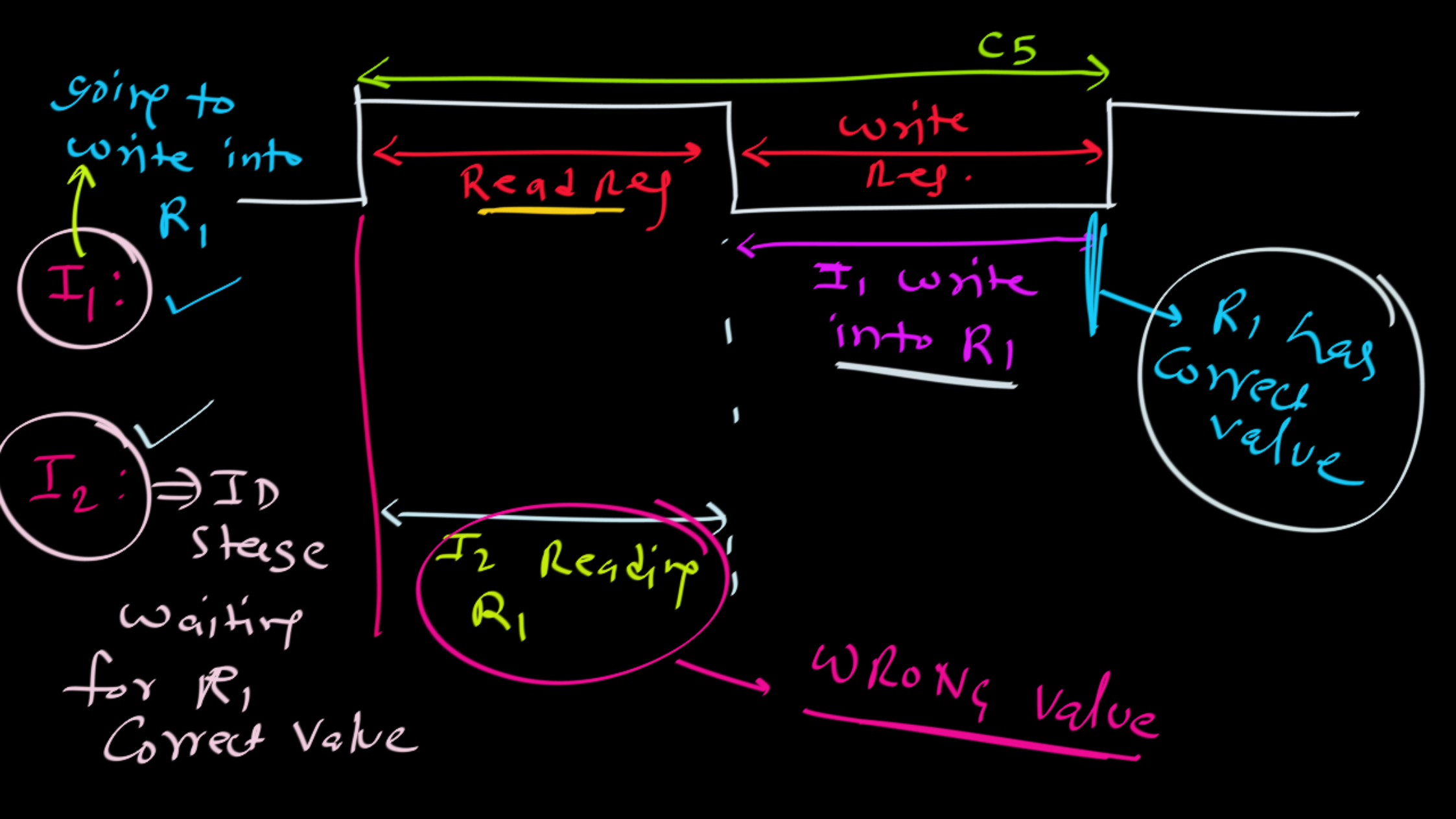


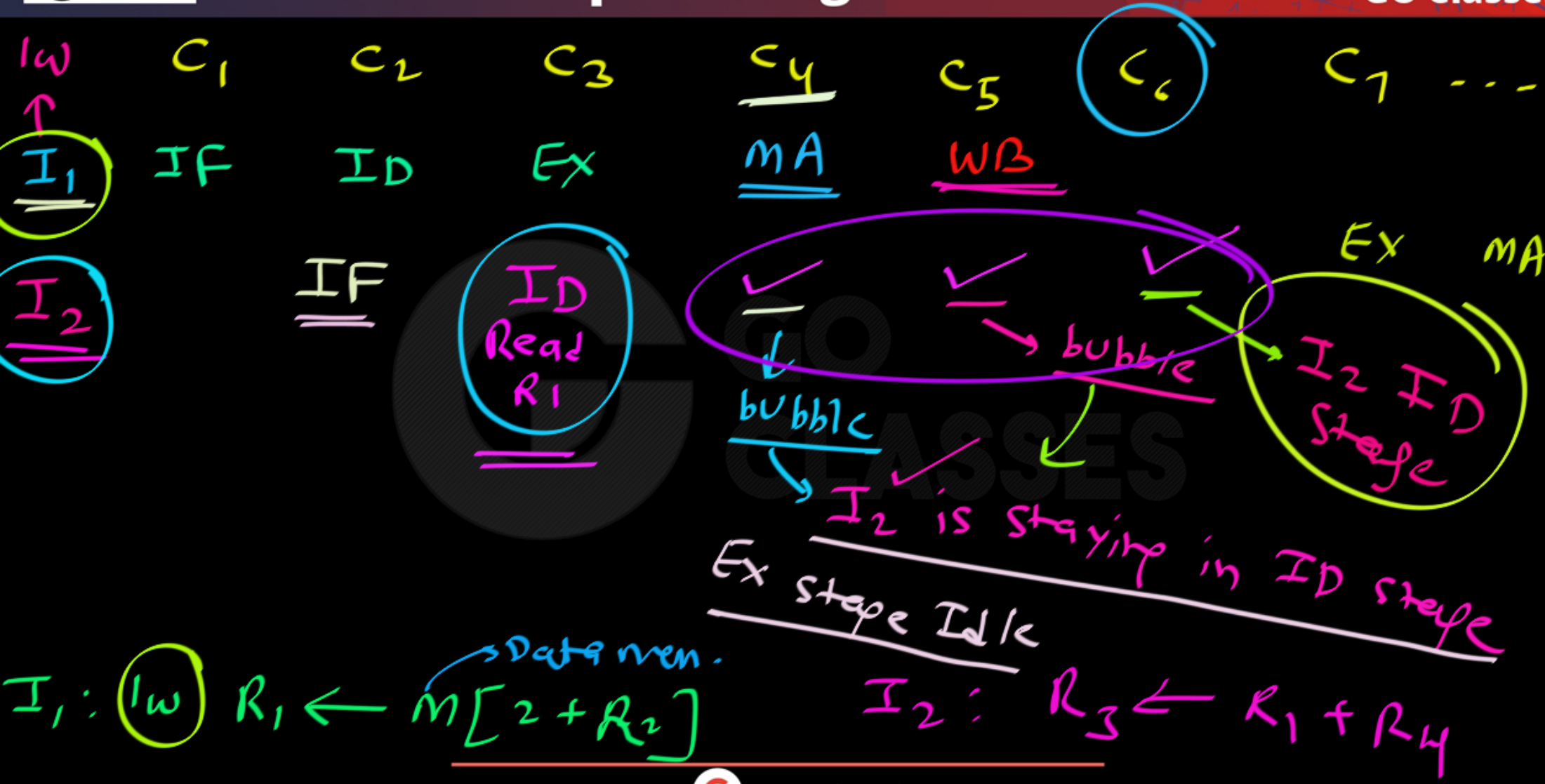
② Rep Read : 2nd Half

Rep write : 1st Half



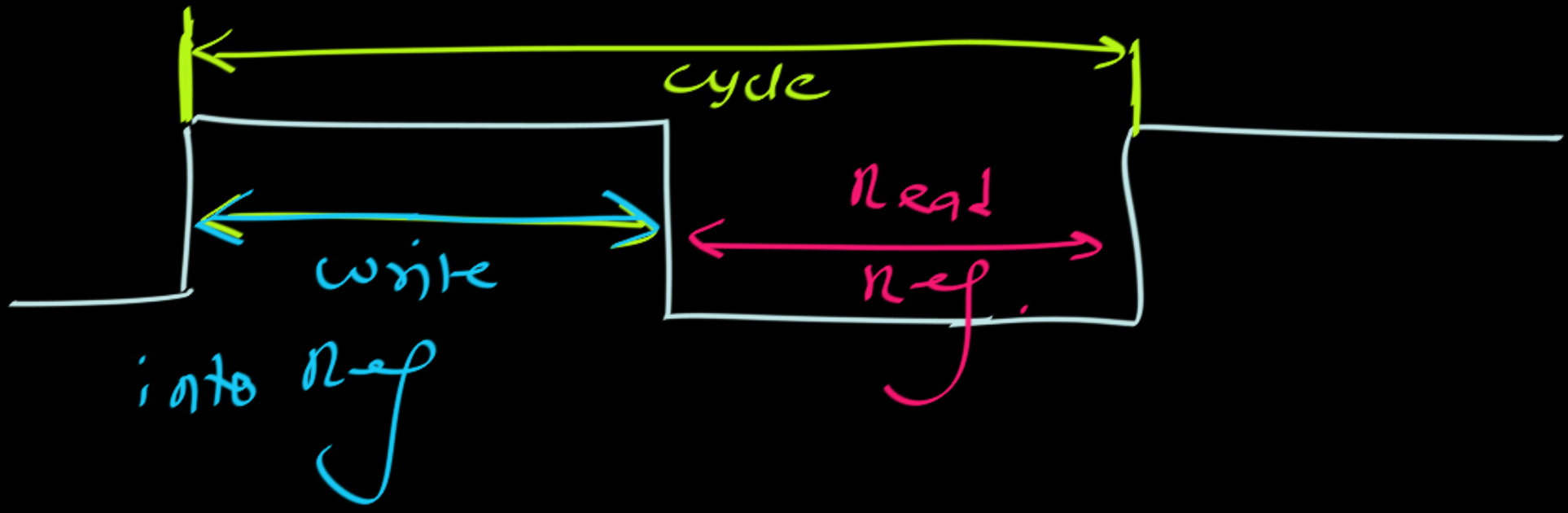

C5
going to write into R_1
I_1:
I_1 write into R_1
I_2: ⇒ ID Stage waiting for R_1 Correct Value
I_2 wants to Read Correct Value of R_1

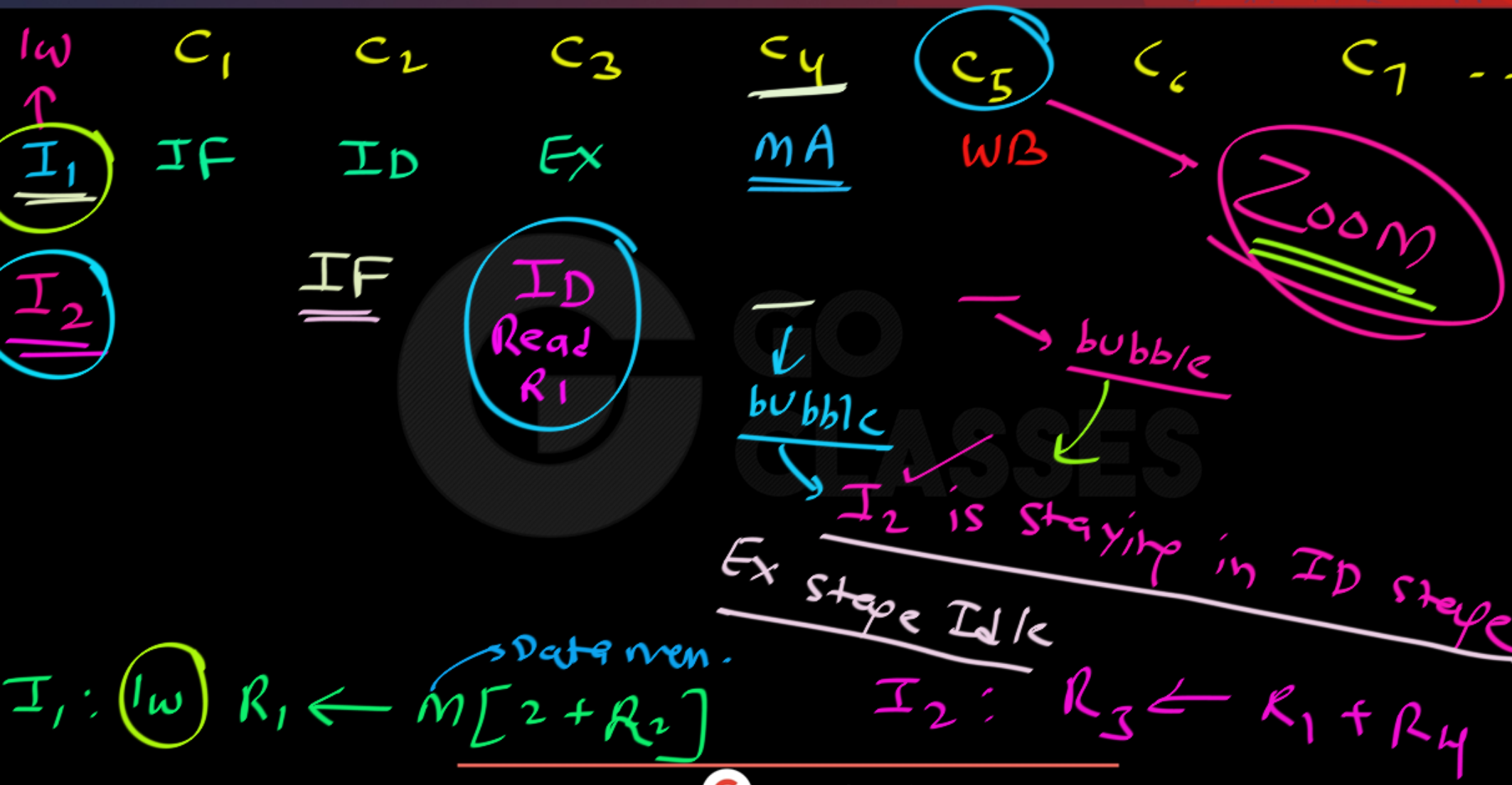

C5
Going to write into R1
Read req
write req.
I1 write into R1
R1 has Correct value
I1:
I2:
ID stage
waiting for R1 Correct Value
I2 Reading R1
WRONG Value

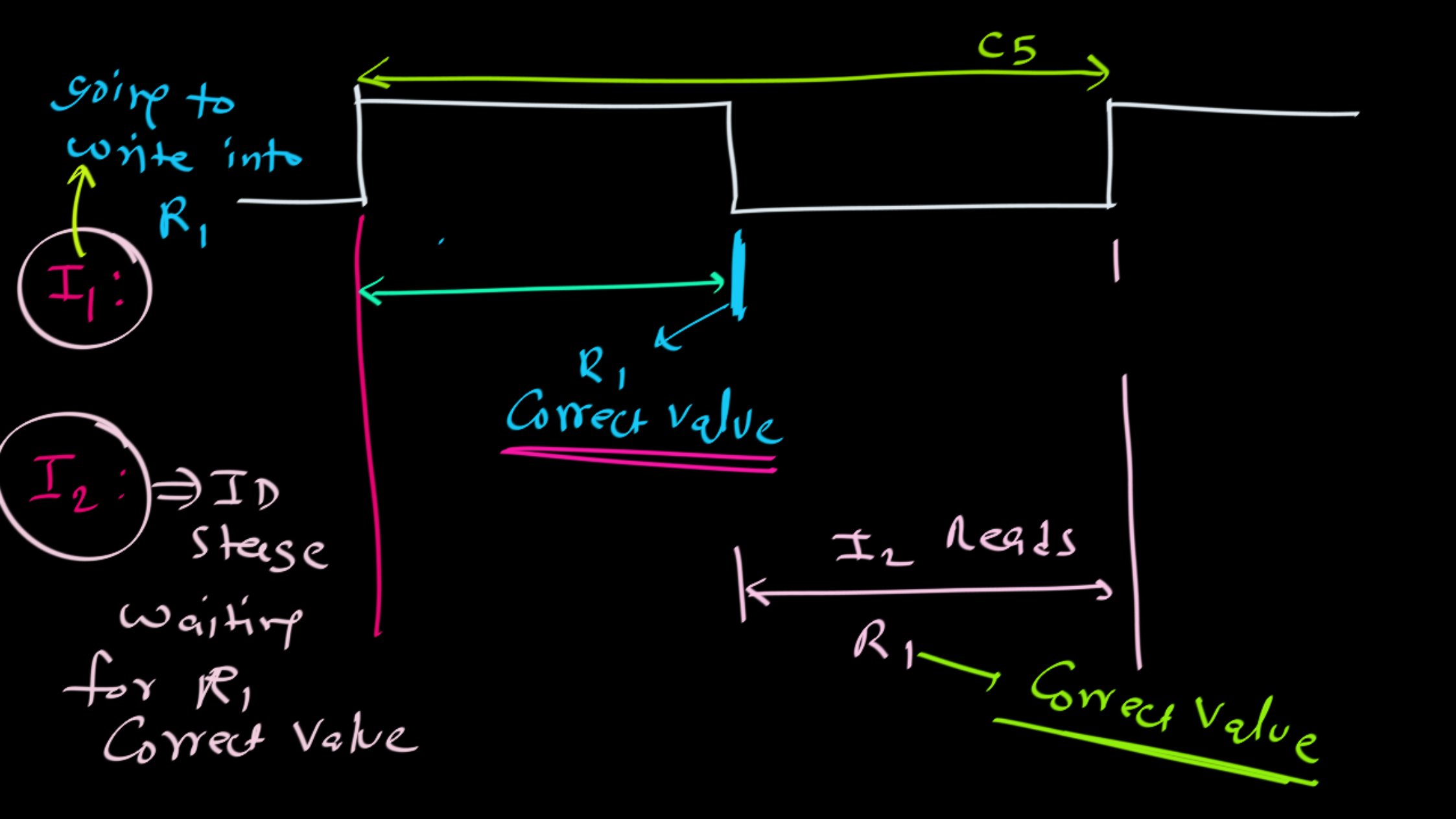

lw
C1
C2
C3
C4
C5
C6
C7 ...
I1
IF
ID
Ex
MA
WB
I2
IF
ID
Read
R1
bubble
bubble
Ex
MA
I2 ID
stage
I2 is staying in ID stage
Ex stage Idle
I1: lw R1 ← M[2+R2]
Data mem.
I2: R3 ← R1 + R4

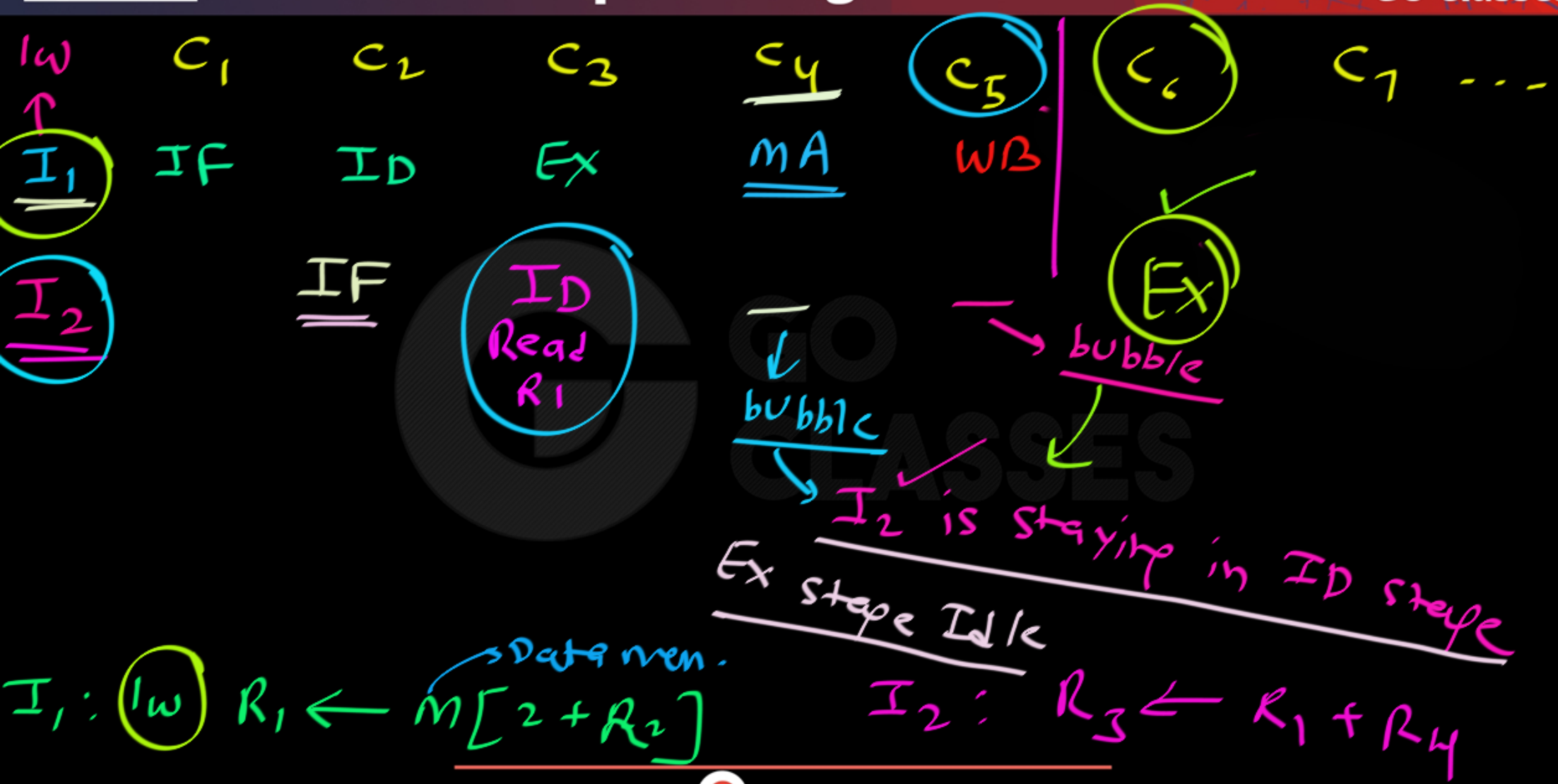
2^{n1} Implementation:

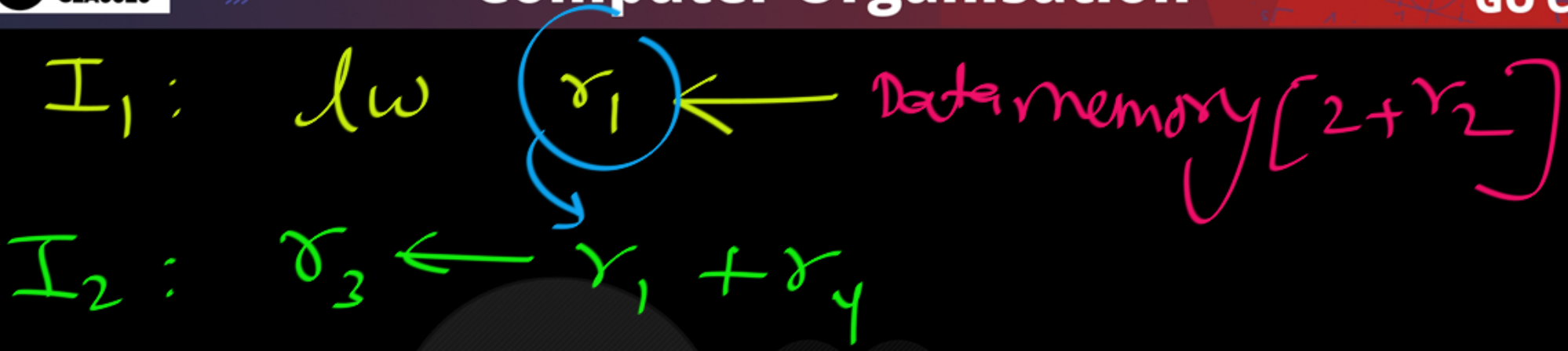
(Split-phase)

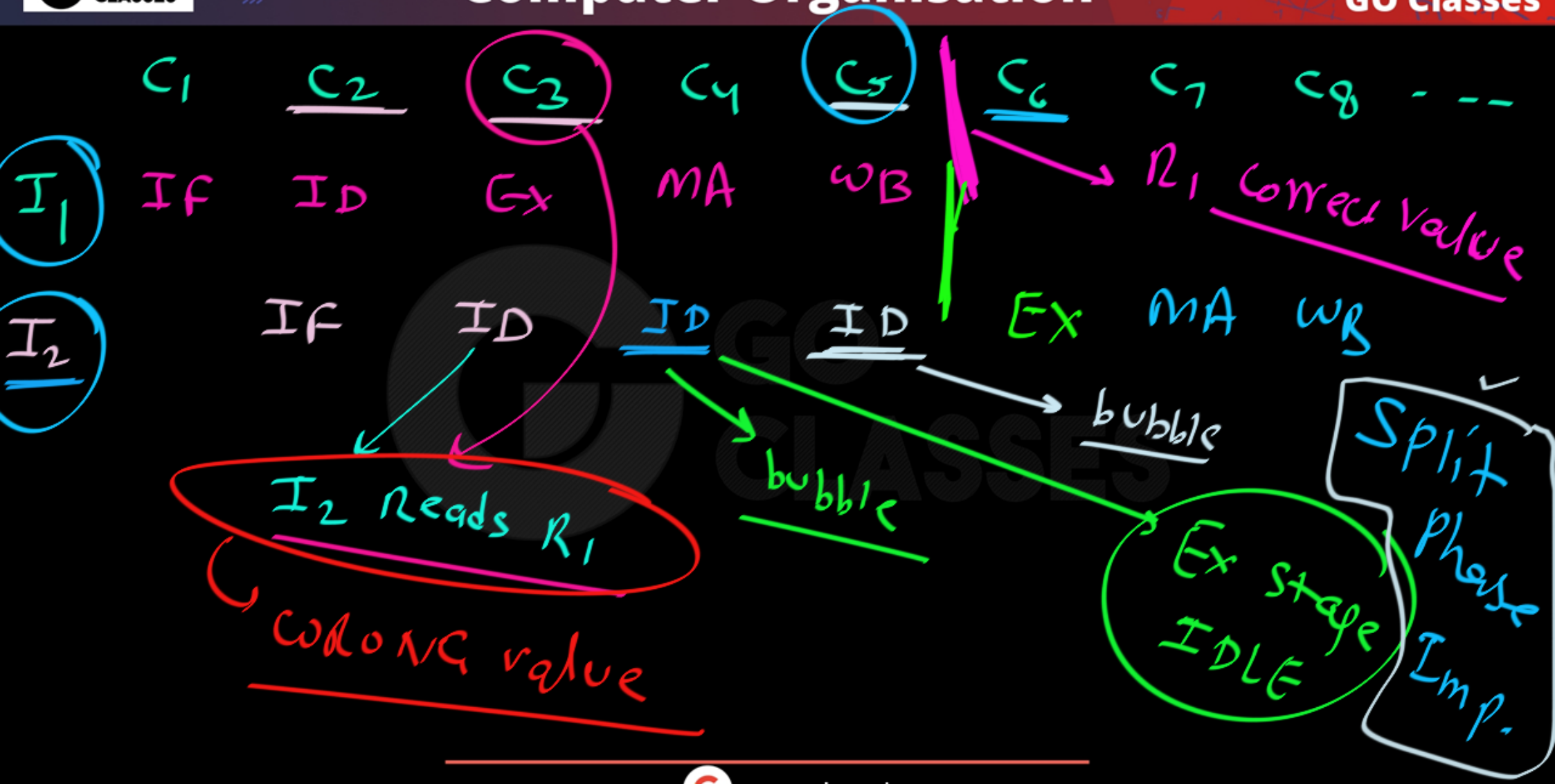


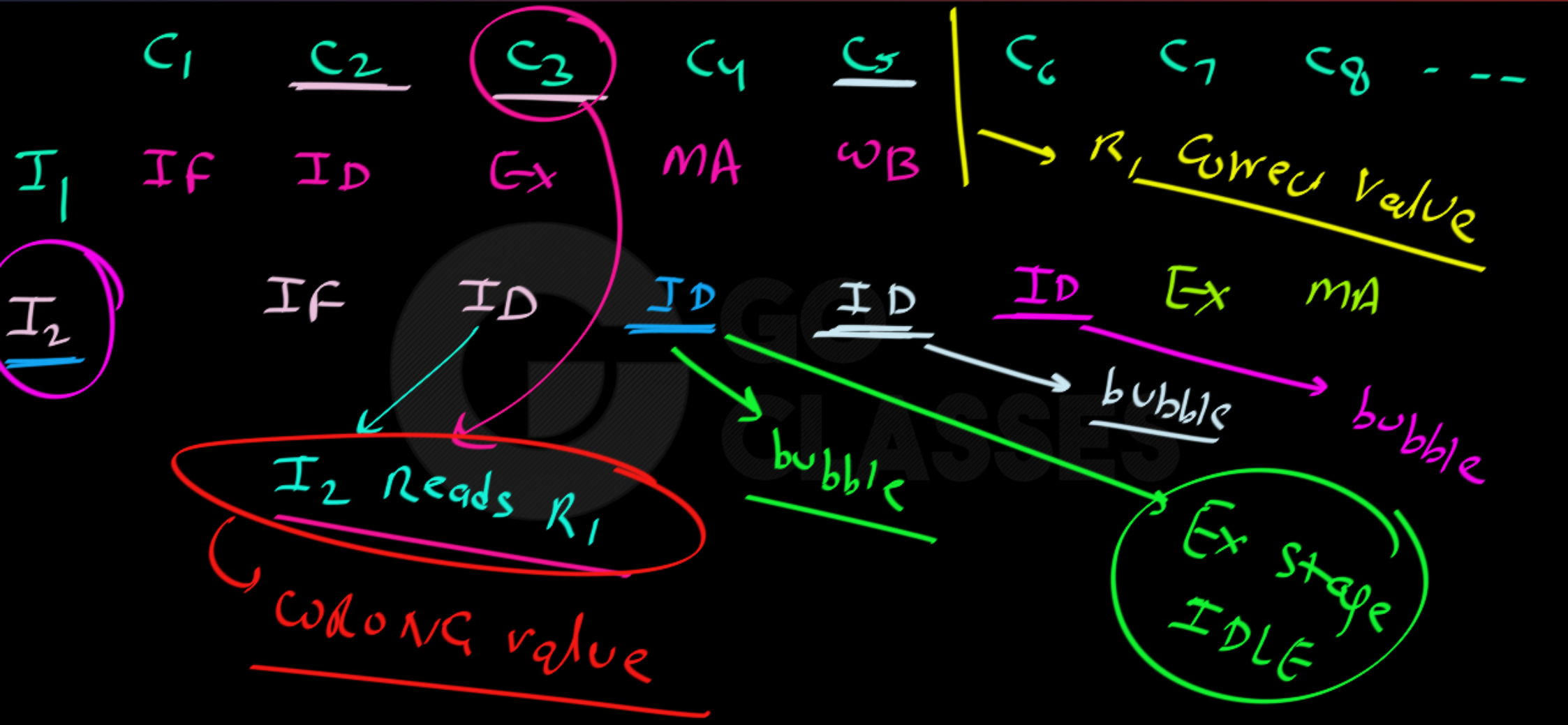

lw
C1
C2
C3
C4
C5
C6
C7 ...
I1
IF
ID
EX
MA
WB
Zoom
I2
IF
ID
Read R1
bubble
bubble
I2 is staying in ID stage
EX stage Idle
I1: lw R1 ← M[2 + R2]
Data men.
I2: R3 ← R1 + R4


C5
going to write into R1
I1:
R1 Correct value
I2: ⇒ ID stese waiting for R1 Correct Value
I2 reads R1 Correct Value


lw
C1
C2
C3
C4
C5
C6
C7 ...
I1
IF
ID
Ex
MA
WB
I2
IF
ID
Read R1
bubble
bubble
Ex
I2 is staying in ID steps
Ex steps Idle
I1: lw R1 ← M[2 + R2]
Data men.
I2: R3 ← R1 + R4

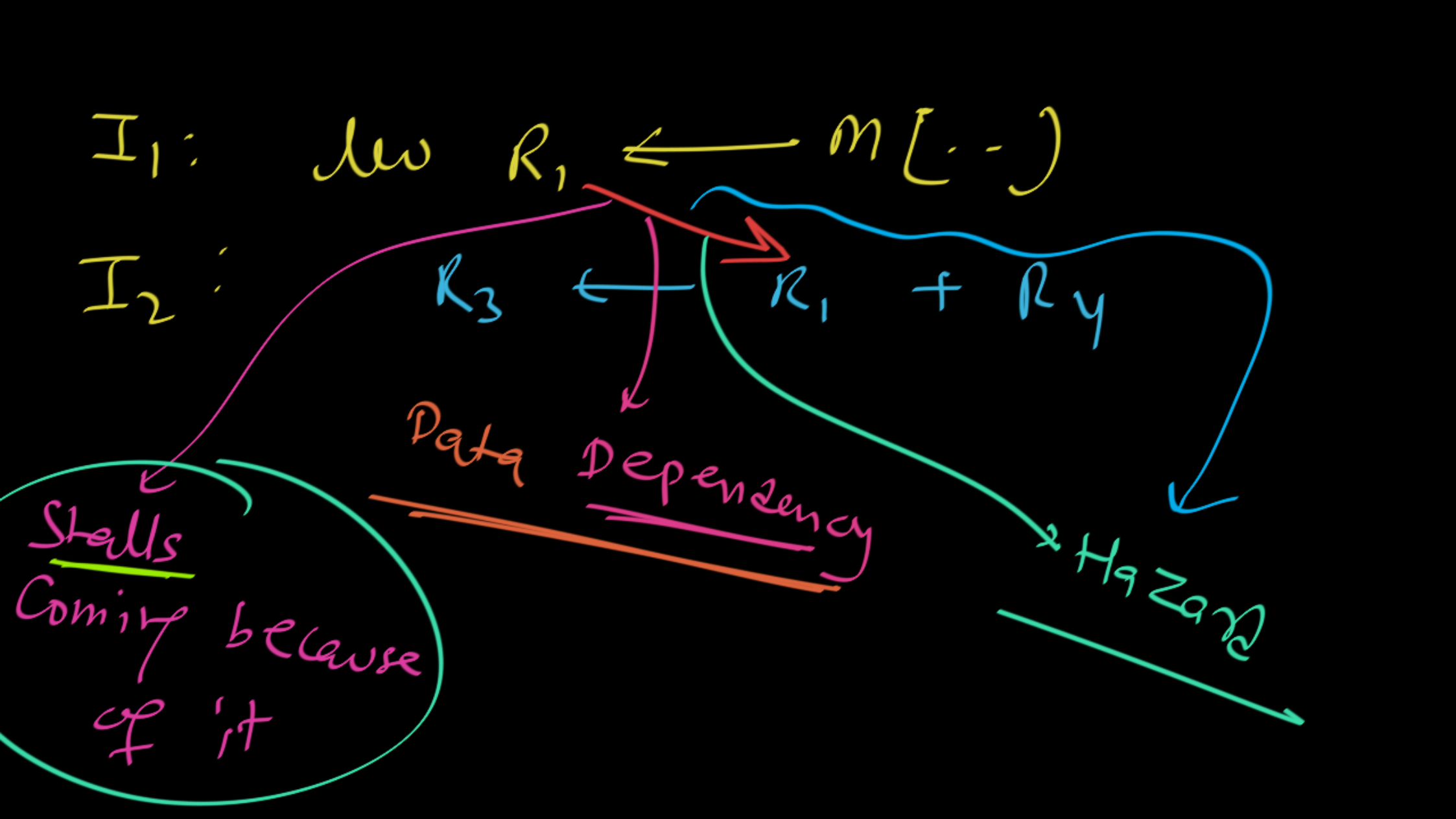

I_1: lw r_1 ← Data memory [2+r_2]
I_2: r_3 ← r_1 + r_4


C1 C2 C3 C4 C5 C6 C7 C8 - - -
I1
IF ID Ex MA WB
R1 Correct value
I2
IF ID ID ID Ex MA WB
bubble
I2 Reads R1
bubble
Wrong value
Ex stage IDLE
Split Phase Imp.


C1 C2 C3 C4 C5 C6 C7 C8 - - -
I1 IF ID Ex MA WB
R1 Correct Value
I2
IF ID ID ID ID Ex MA
bubble
bubble
bubble
I2 Reads R1
Wrong value
Ex stage IDLE

Bubbles ≡ Stalls

Any condition which creates Stalls is called Hazard.


I1: Mw R1 ← M[..]
I2: R3 ← R1 + Ry
Data Dependency
Stalls
Coming because
of it
Hazard

MISConception :

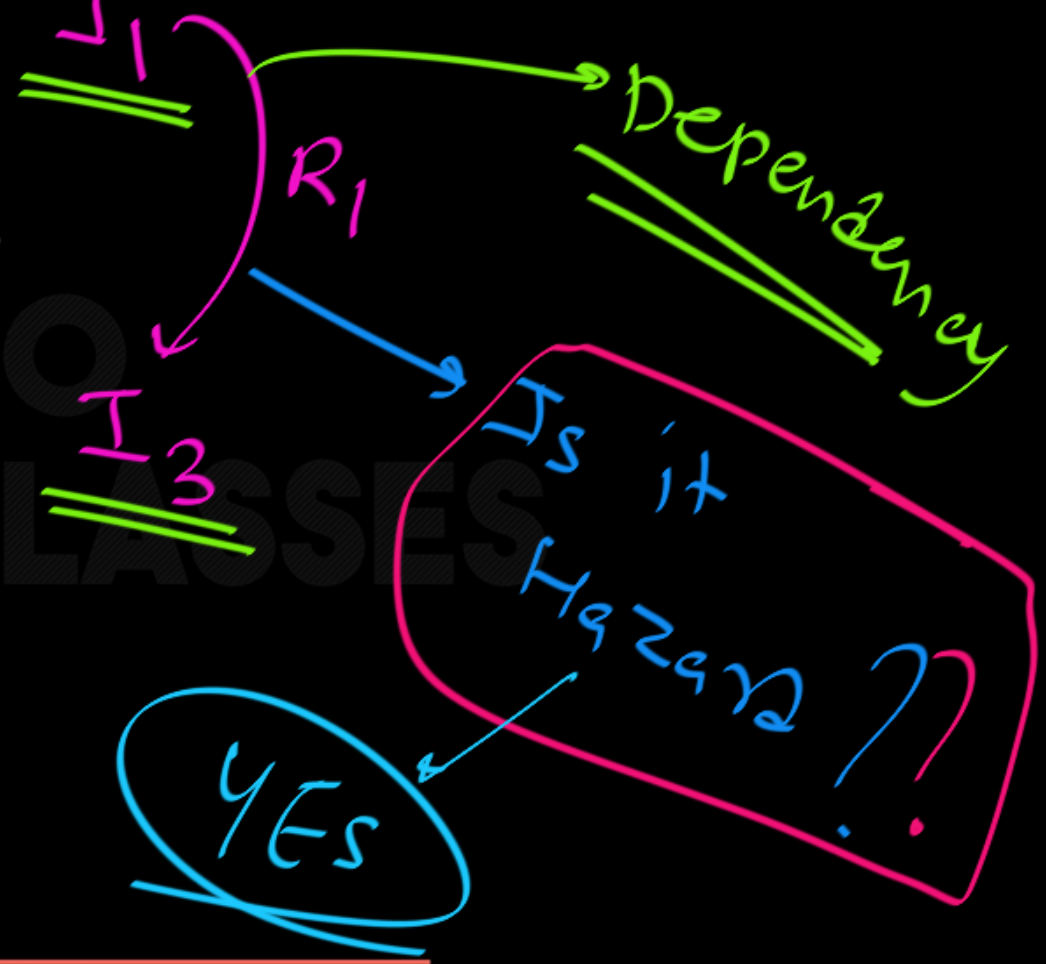
Hazard Happens Consecutive Instruction only by

WRONG

1: $R_1 \leftarrow R_2 + R_3$

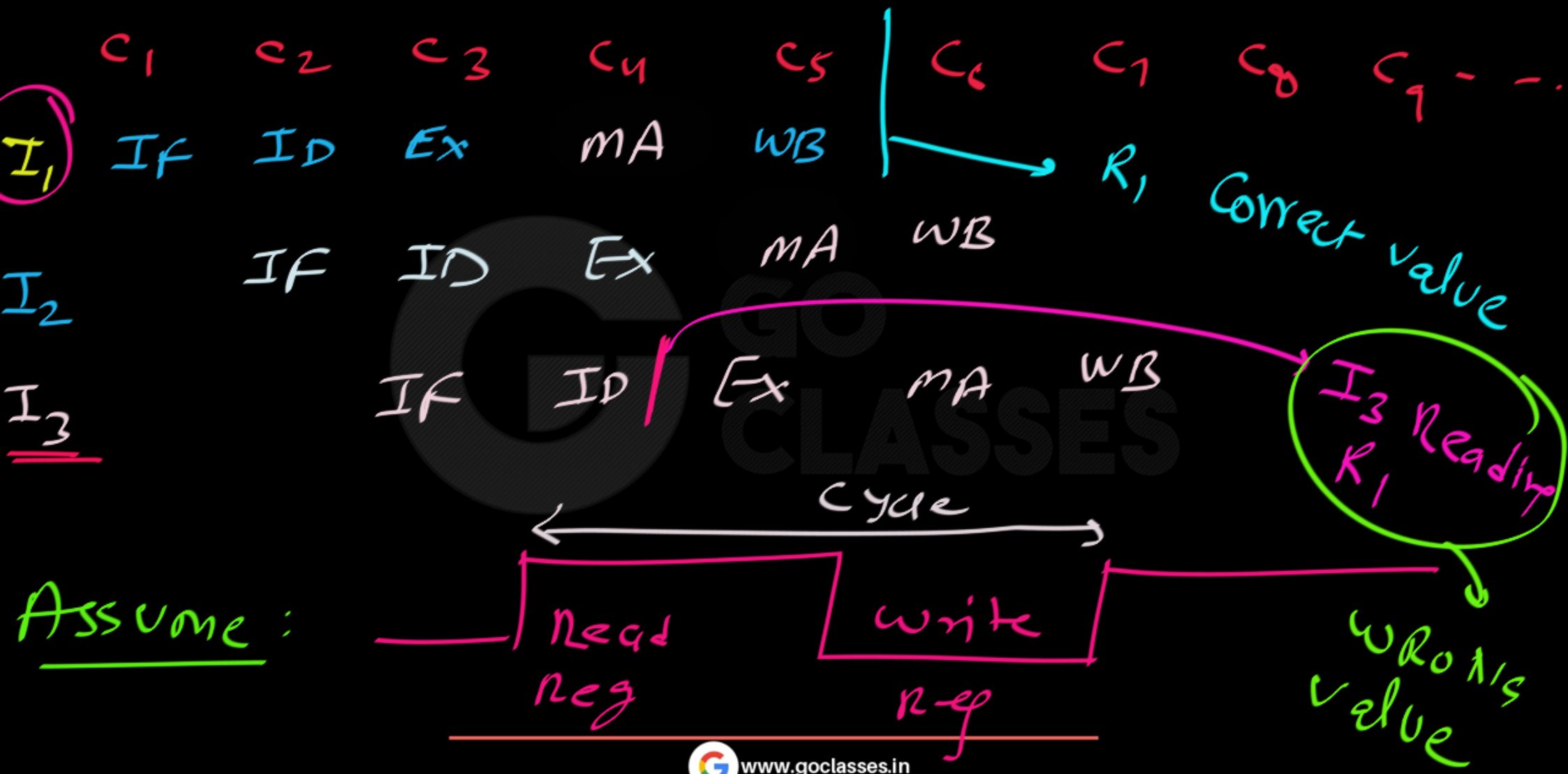
2: $R_4 \leftarrow R_5 + R6$

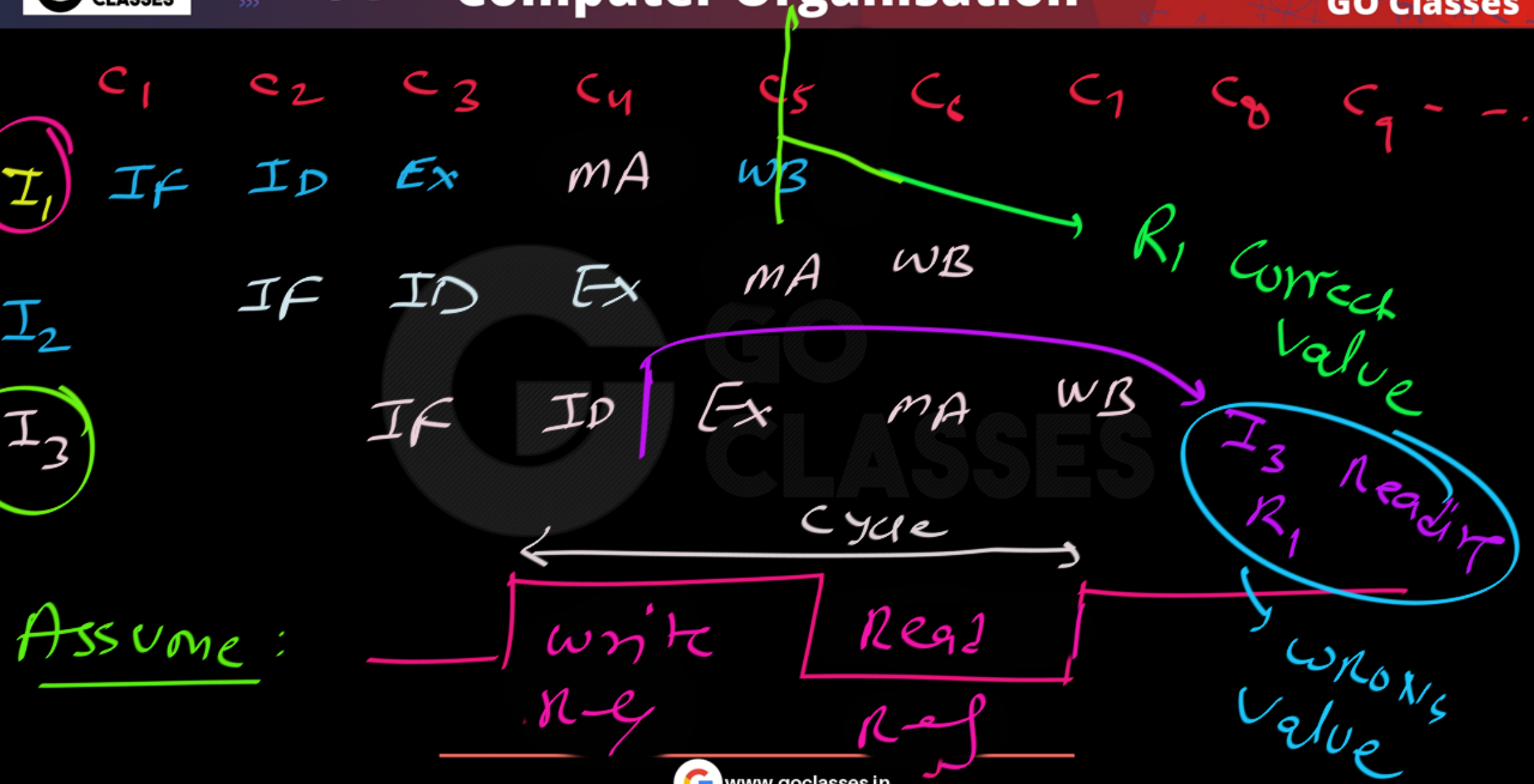
3: $R_7 \leftarrow R_1 + R8$



How to check if something is Hazard?

Just Do Normal pipeline execution, & then check if any problem.


C1 C2 C3 C4 C5 C6 C7 C8 C9 - - -
I1 IF ID EX MA WB
R1
Correct value
I2 IF ID EX MA WB
I3 IF ID EX MA WB
I3 Reading R1
Cycle
Assume :
Read
write
Reg
Reg
WRONG value


C1 C2 C3 C4 C5 C6 C7 C8 C9 - - -
I1 IF ID EX MA WB
I2 IF ID EX MA WB
I3 IF ID EX MA WB
R1 Correct Value
I3 reads R1
Wrong Value
Cycle
Assume :
write
Read
Re
Rd

8.2 Data Hazards

A data hazard is a situation in which the pipeline is stalled because the data to be operated on are delayed for some reason, as illustrated in Figure 8.3. We will now examine the issue of availability of data in some detail.

Any condition that causes the pipeline to stall is called a hazard.

We have just seen an example of a data hazard.
A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result some operation has to be delayed, and the pipeline stalls.

Data Dependency Vs Data Hazards

First we see:

Data Dependency

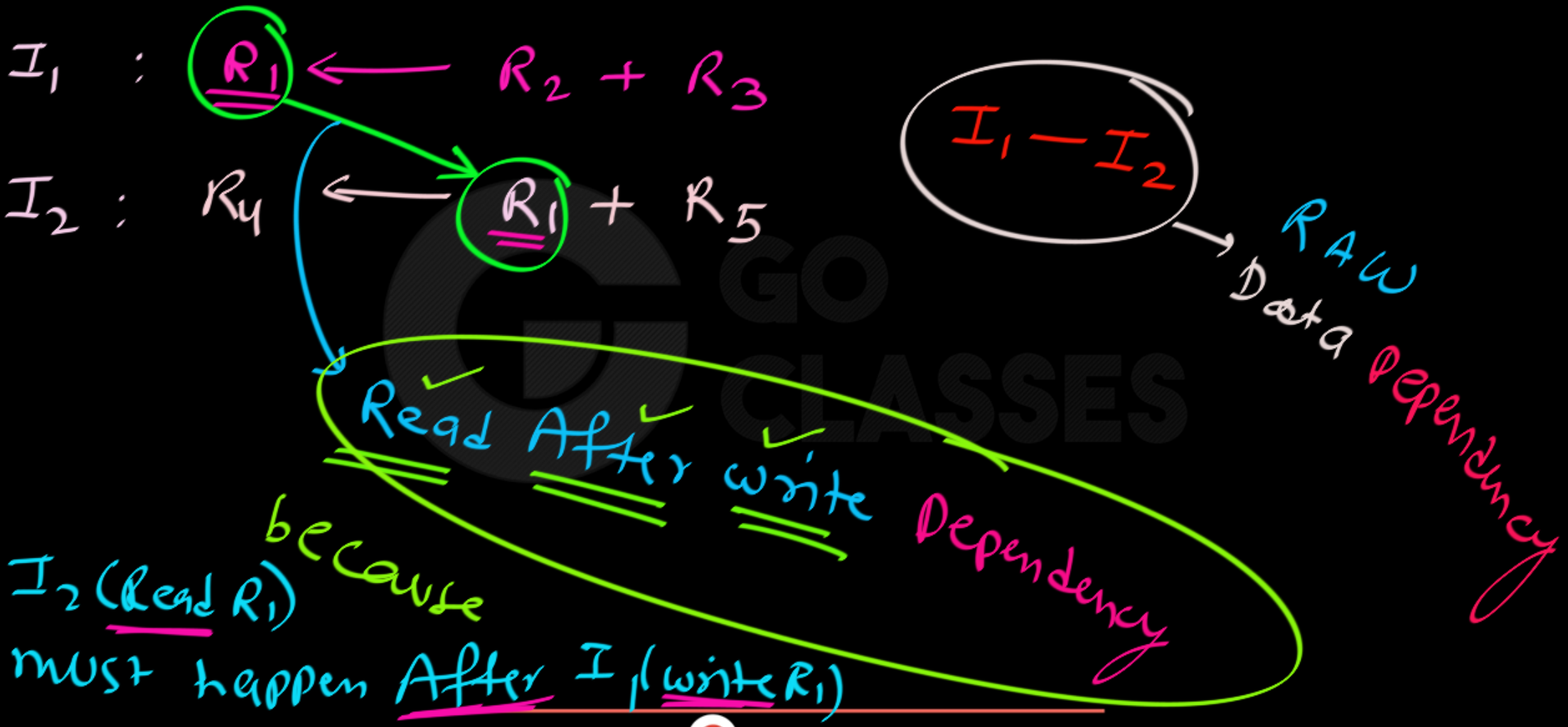
DOES NOT Depend on pipeline

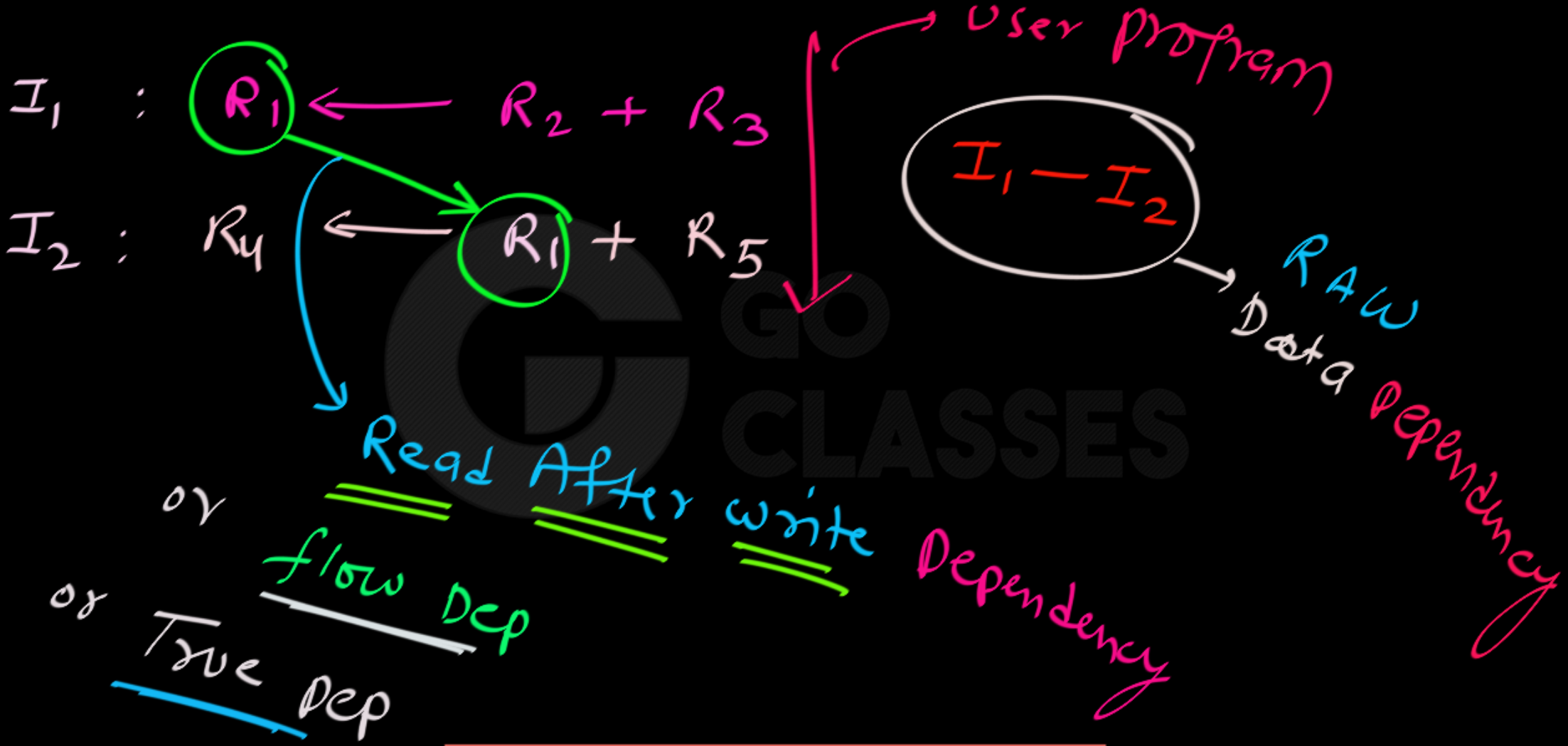
First we see:

Data Dependency

Property of Code only :

Purely


I1 : R1 ← R2 + R3
I2 : R4 ← R1 + R5
I1 — I2
RAW
Data Dependency
Read After write Dependency
because
I2 (Read R1)
must happen After I1 (write R1)


I1 : R1 ← R2 + R3
I2 : R4 ← R1 + R5
User Program
I1 — I2
RAW
Data Dependency
Read After write Dependency
or
or
flow Dep
True Dep

I_1 : $R_1 \leftarrow R_2 + R_3$

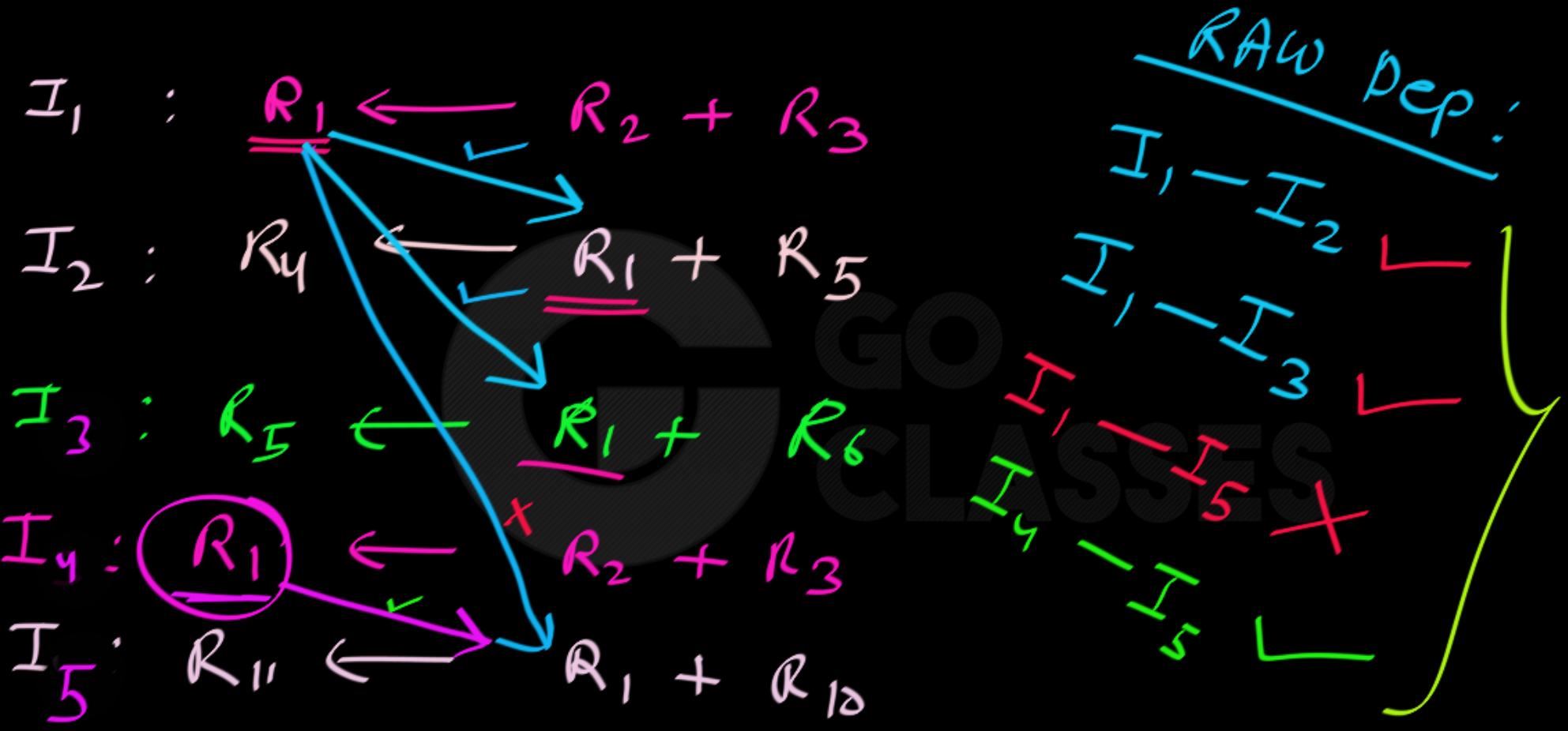
I_2 : $R_4 \leftarrow R_1 + R_5$

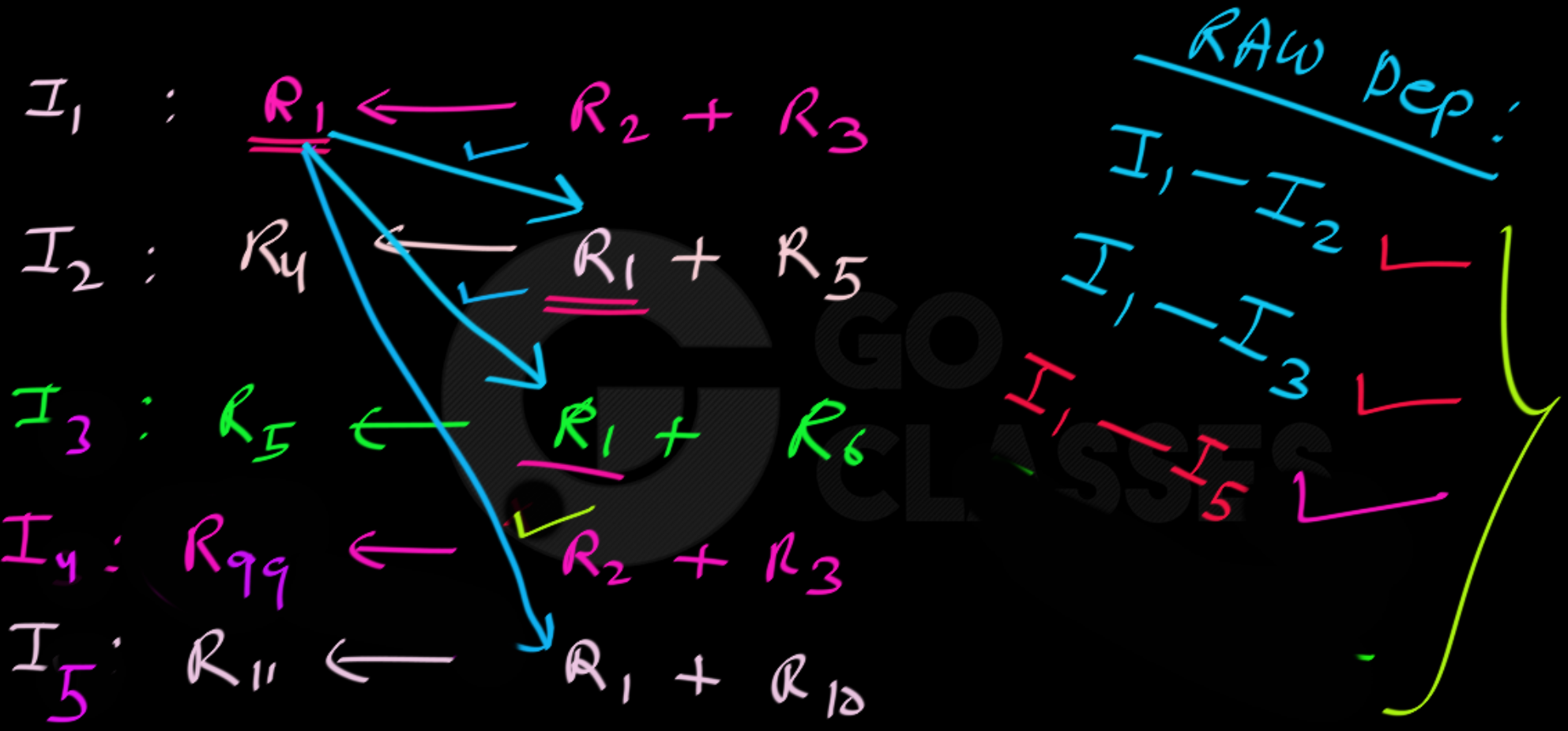
I_3 : $R_5 \leftarrow R_1 + R_6$

I_4 : $R_1 \leftarrow R_2 + R_3$

I_5 : $R_{11} \leftarrow R_1 + R_{10}$

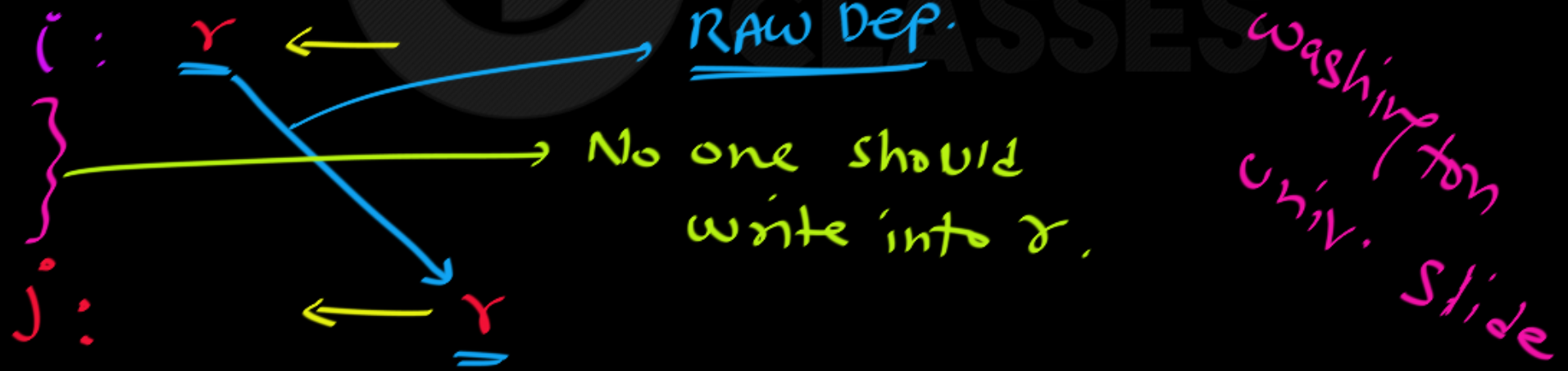
How many RAW Dep. ?


I1 : R1 ← R2 + R3
I2 : R4 ← R1 + R5
I3 : R5 ← R1 + R6
I4 : R1 ← R2 + R3
I5 : R11 ← R1 + R10
RAW dep :
I1 – I2
I1 – I3
I1 – I5
I4 – I5


I1 : R1 ← R2 + R3
I2 : R4 ← R1 + R5
I3 : R5 ← R1 + R6
I4 : R99 ← R2 + R3
I5 : R11 ← R1 + R10
RAW dep :
I1 – I2
I1 – I3
I1 – I5

Detecting Data Dependencies

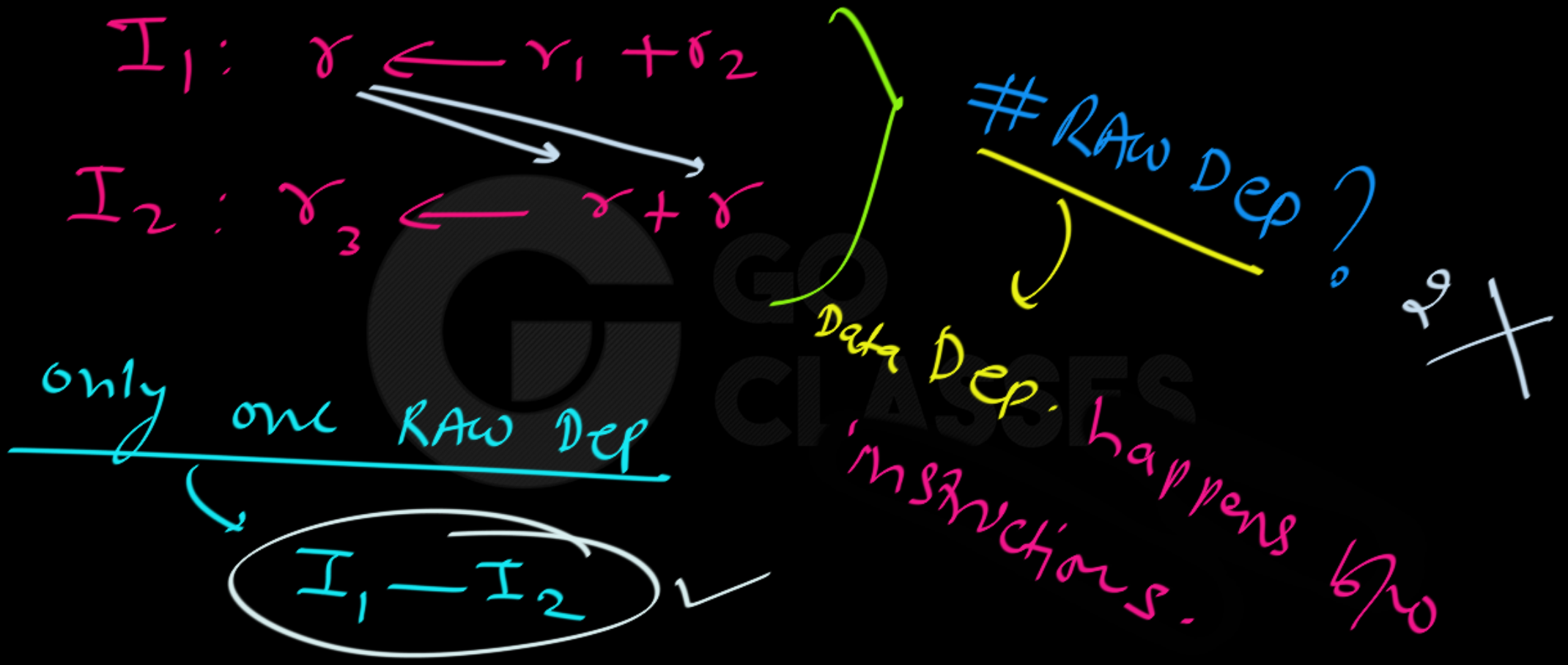
- Dependencies: Given two instructions, *i* and *j* (*i* occurs before *j*).
- We say a *dependence* exists between *i* and *j* if *j* reads the result produced by *i*, and there is no instruction *k* which occurs between *i* and *j* and that produces the same result as *i*.



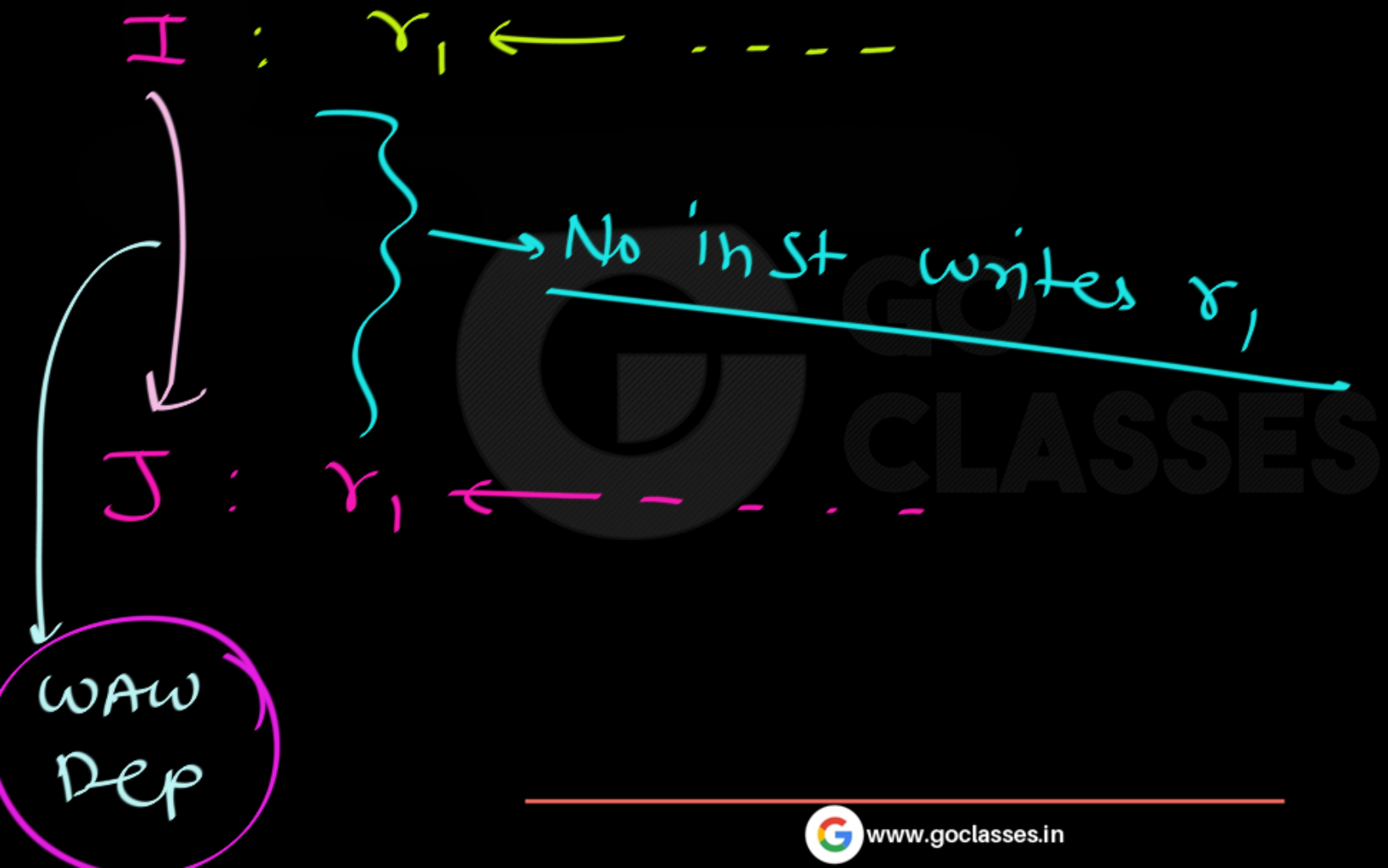
$I_1: r \leftarrow r_1 + r_2$

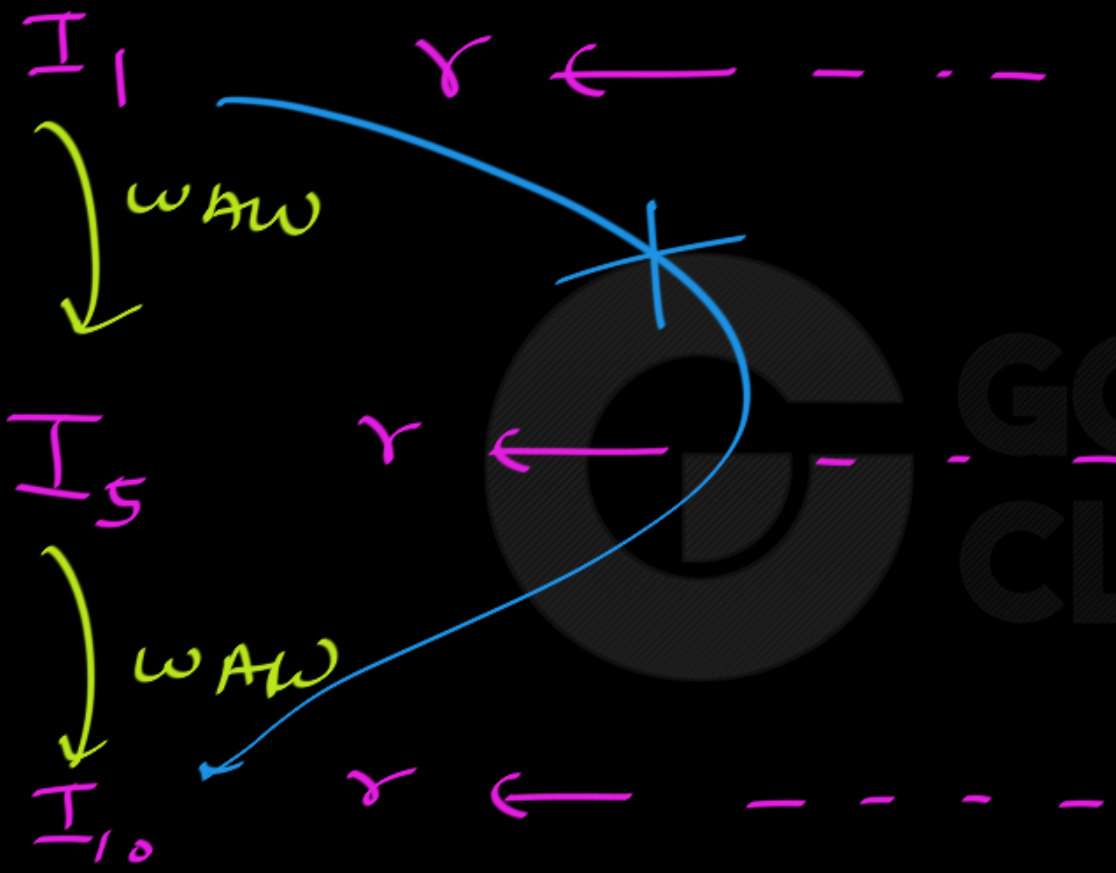
$I_2: r_3 \leftarrow r + r$

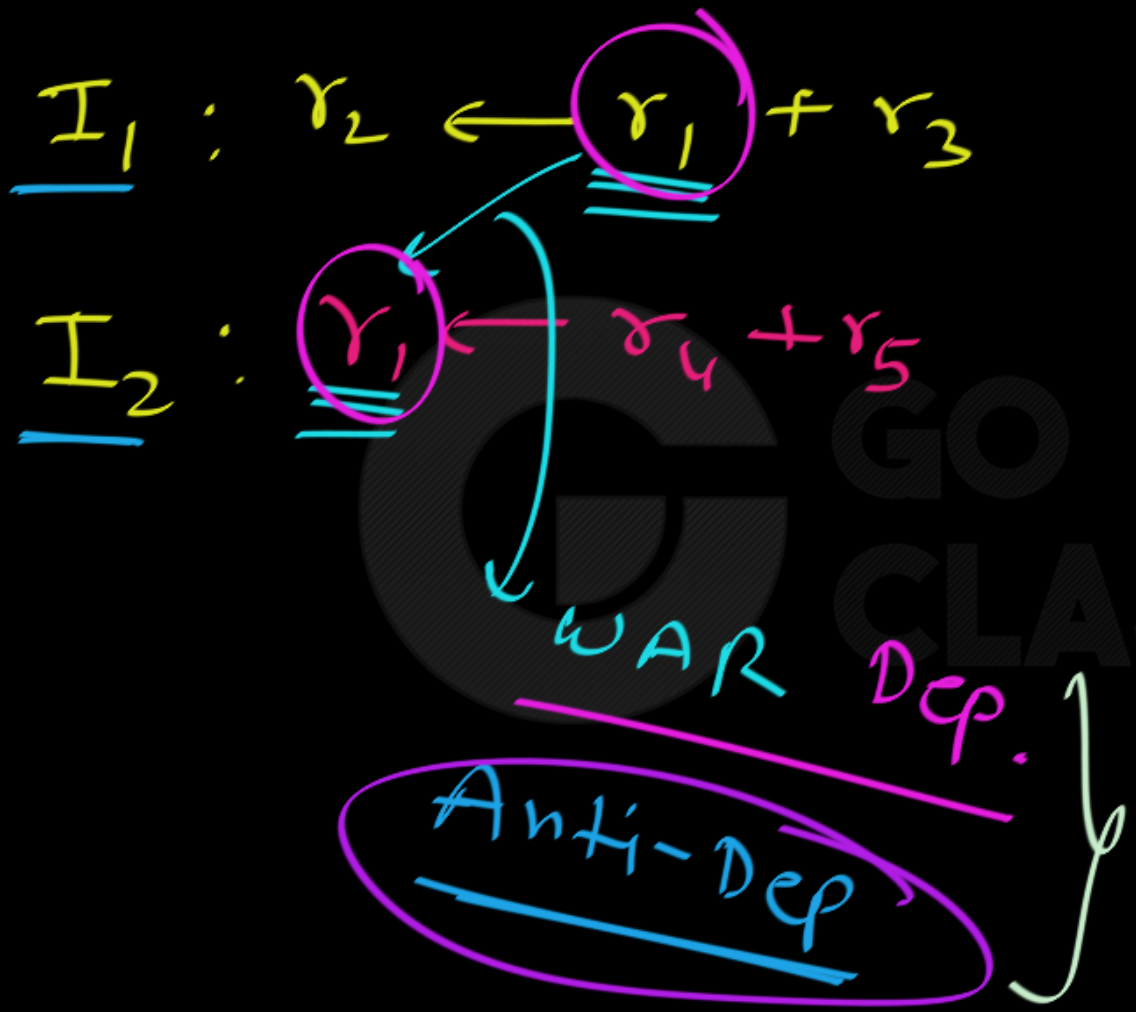
RAW Dep ?

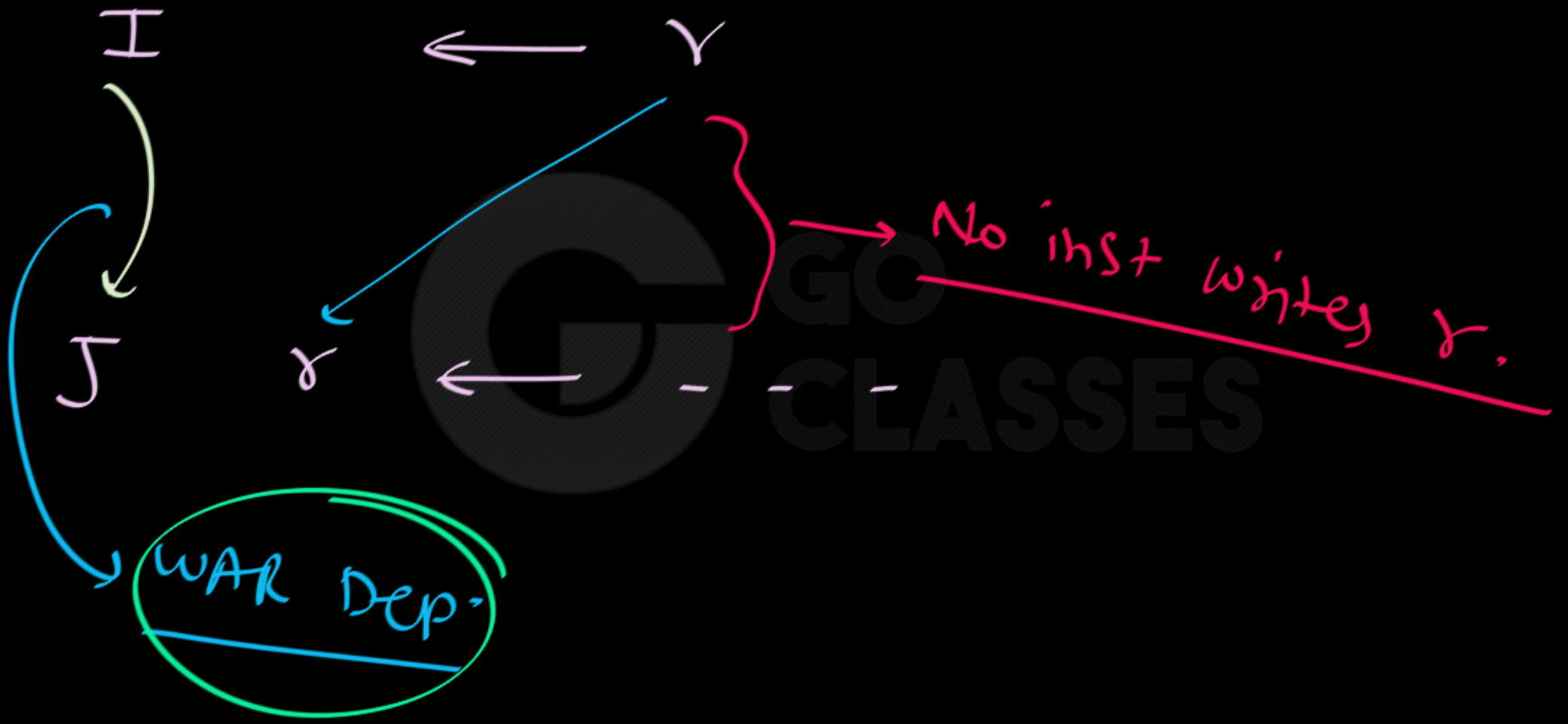

I1 : r ← r1 + r2
I2 : r3 ← r + r
RAW Dep ?
Data Dep. happens b/w
instructions.
Only one RAW Dep
I1 — I2

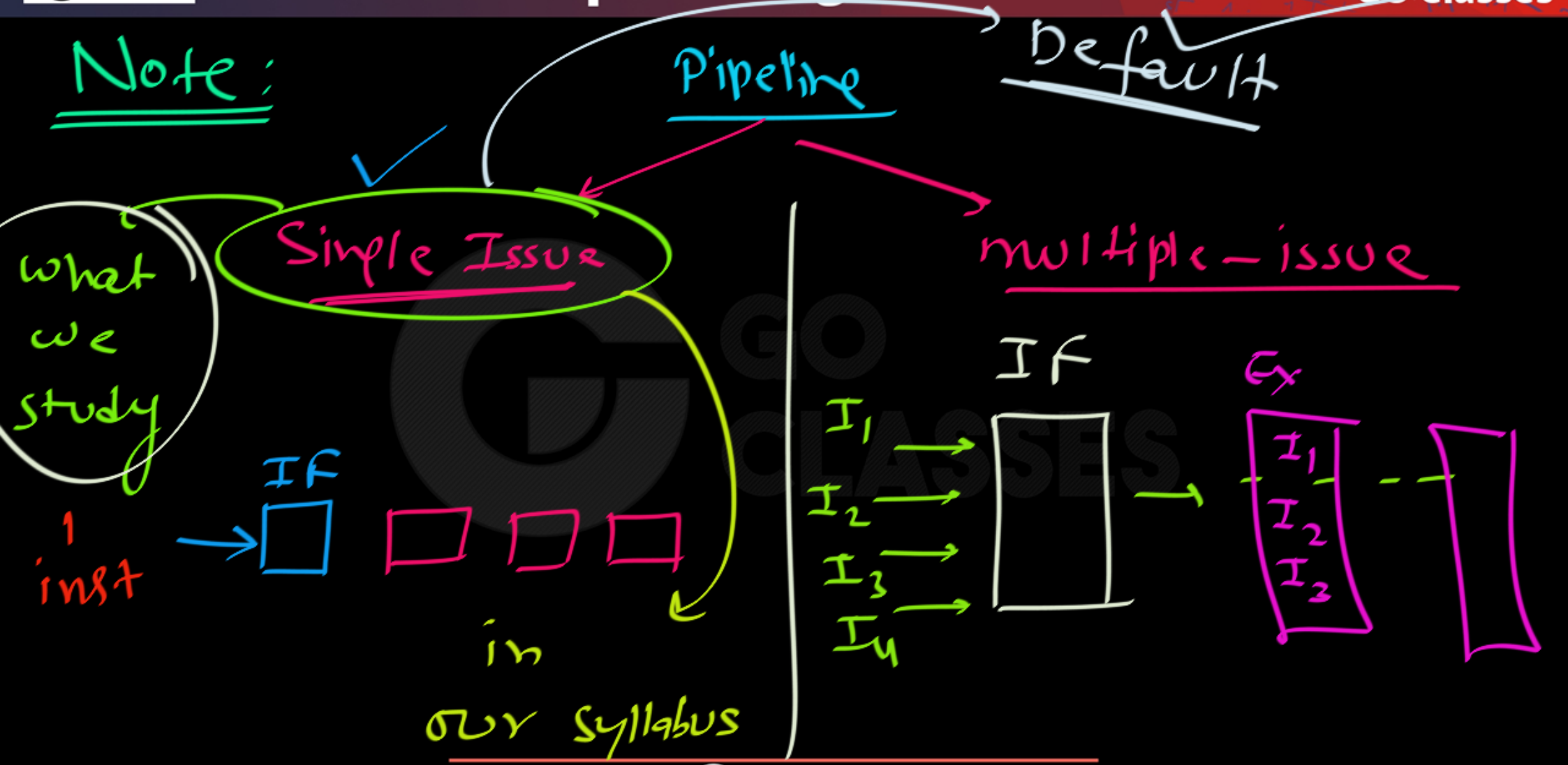

I1 :
I2 :
Output Data Dependency
(WAW Dep)
future instruction


H : r_1 ←
No 'in St writes r_1
J : r_1 ←
WAW Dep


I1
r
WAW
I5
r
WAW
I10
r


I1 : r2 ← r1 + r3
I2 : r1 ← r4 + r5
WAR Dep.
Anti-Dep


I
J
WAR Dep.
No inst writes r.


Note :
Pipeline
Default
what we study
Simple Issue
multiple-issue
1 inst
If
in our Syllabus
If
Ex
I_1
I_2
I_3
I_4

Code:

Data Dep.

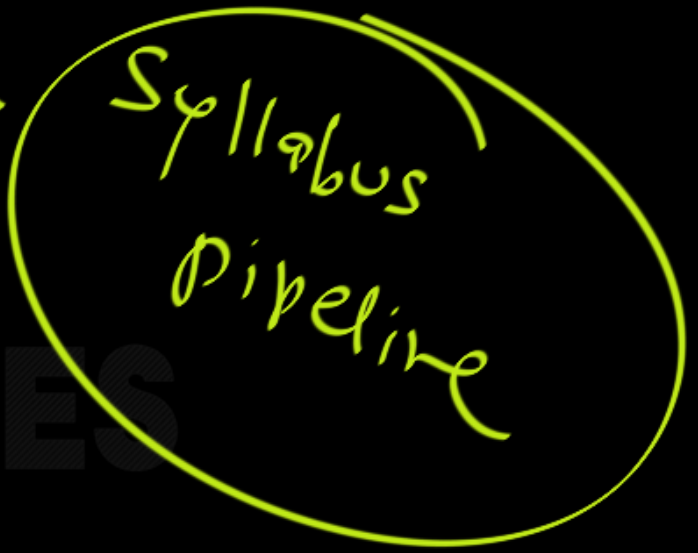
- RAW → only they may (may not) create stalls.
- WAR → NEVER create any problem (stalls)
- WAW

↓ Depend only on code

Our Simple Issue pipelines

In Simple Issue pipeline

any



we have NO

WAW Hazards ; WAR Hazards.

Code:

Data Dep.

- RAW
- WAR
- WAW

↓ Depend only on code

Hazards Possible

These Hazards can be 100% Eliminated Algorithmically.

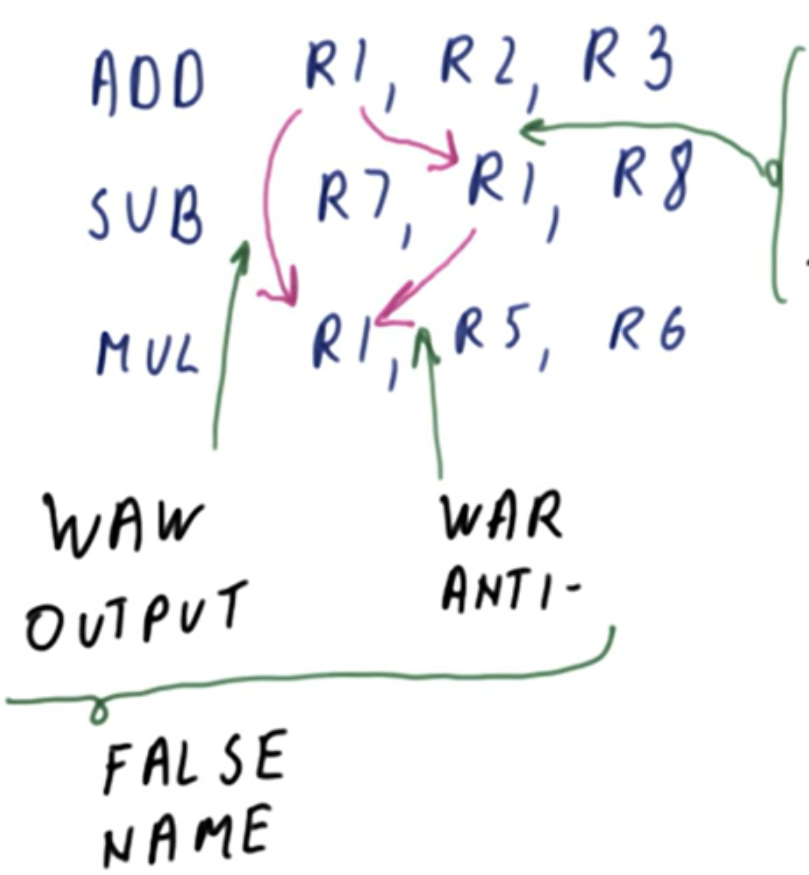
multiple Issue Pipeline

RAW Dep : True Dep

WAW, WAR Dep : false Dep

→ name Dep.

DATA DEPENDENCIES



HOW ABOUT RAR ?

ADD R1, R3, R4

ADD R2, R3, R7

DATA DEPENDENCIES QUIZ

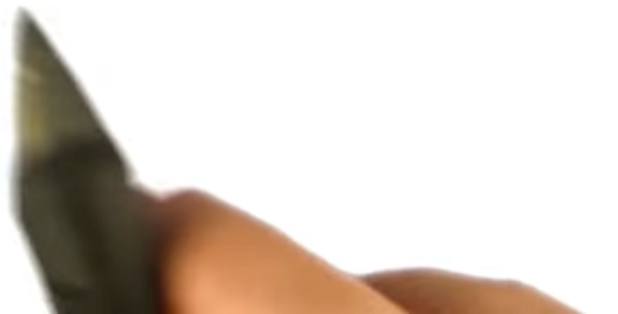
```
I1: MUL R1, R2, R3
I2: ADD R4, R4, R1
I3: MUL R1, R5, R6
I4: SUB R4, R4, R1
```

	RAW	WAR	WAW
I1 → I2	✓	/	/
I1 → I3	/		
I1 → I4			
I2 → I3			

DATA DEPENDENCIES QUIZ SOLUTION

```
I1: MUL R1, R2, R3
I2: ADD R4, R4, R1
I3: MUL R1, R5, R6
I4: SUB R4, R4, R1
```

	RAW	WAR	WAW
I1 → I2	✓	/	/
I1 → I3	/	/	✓
I1 → I4	/	/	/
I2 → I3	/	✓	/



Pipelining/Problem 01

Consider following assembly-language program:

```
1: MOV R3, R7

2: LD R8, (R3)

3: ADD R3, R3, 4

4: LOAD R9, (R3)

5: BNE R8, R9, L3
```

a) This program includes WAW, RAW, and WAR dependencies. Show these.

b) What is the difference between a dependency and hazard?

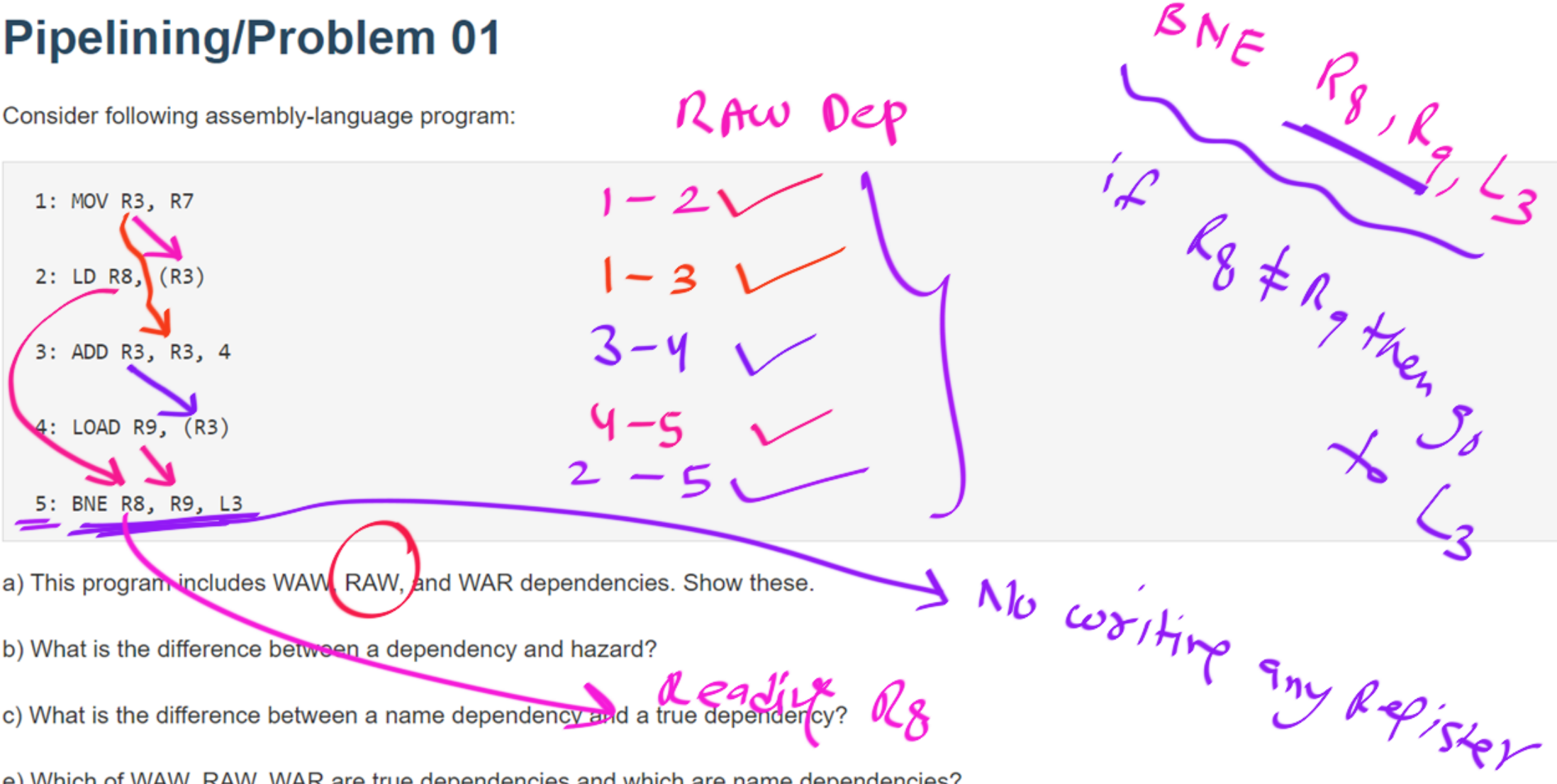
c) What is the difference between a name dependency and a true dependency?

e) Which of WAW, RAW, WAR are true dependencies and which are name dependencies?

Pipelining/Problem 01

Consider following assembly-language program:

```
1: MOV R3, R7
2: LD R8, (R3)
3: ADD R3, R3, 4
4: LOAD R9, (R3)
5: BNE R8, R9, L3
```



a) This program includes WAW, RAW, and WAR dependencies. Show these.

b) What is the difference between a dependency and hazard?

c) What is the difference between a name dependency and a true dependency?

e) Which of WAW, RAW, WAR are true dependencies and which are name dependencies?

Pipelining/Problem 01

Consider following assembly-language program:

```
1: MOV R3, R7

2: LD R8, (R3)

3: ADD R3, R3, 4

4: LOAD R9, (R3)

5: BNE R8, R9, L3
```

a) This program includes WAW, RAW, and WAR dependencies. Show these.

b) What is the difference between a dependency and hazard?

c) What is the difference between a name dependency and a true dependency?

e) Which of WAW, RAW, WAR are true dependencies and which are name dependencies?

udacity.com/wiki/hpca/problem-set-solutions/pipelining/problem01

Pipelining/Problem 01

Consider following assembly-language program:

```
1: MOV R3, R7

2: LD R8, (R3)

3: ADD R3, R3, 4

4: LOAD R9, (R3)

5: BNE R8, R9, L3
```

a) This program includes WAW, RAW, and WAR dependencies. Show these.

b) What is the difference between a dependency and hazard?

c) What is the difference between a name dependency and a true dependency?

e) Which of WAW, RAW, WAR are true dependencies and which are name dependencies?

Solution

WAW: L1-L3;

RAW: L1-L2,L1-L3, L3-L4,L2-L5,L4-L5

WAR: L2-L3;

b) A hazard is created when there is a dependency between instructions and they are close enough that the overlap caused by pipelining (or reordering) would change the order of access to the dependent operand.

c) In a true dependency information is transmitted between instructions, while this is not the case for name dependencies.

d) Name dependencies: WAR, WAW

True dependencies: RAW

Now we see:

Data Hazards

Any condition that causes the pipeline to stall is called a hazard.

We have just seen an example of a data hazard.
A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result some operation has to be delayed, and the pipeline stalls.

8.2 Data Hazards

A data hazard is a situation in which the pipeline is stalled because the data to be operated on are delayed for some reason, as illustrated in Figure 8.3. We will now examine the issue of availability of data in some detail.

Data Hazards:

Depends on → Code + Pipeline

How to check if some Data Dep is Hazard in a pipeline?

Just do Normal PL exⁿ & check if any problem.

1: $R_1 \leftarrow R_2 + R_3$

2: $R_4 \leftarrow R_5 + R_6$

3: $R_7 \leftarrow R_1 + R_8$

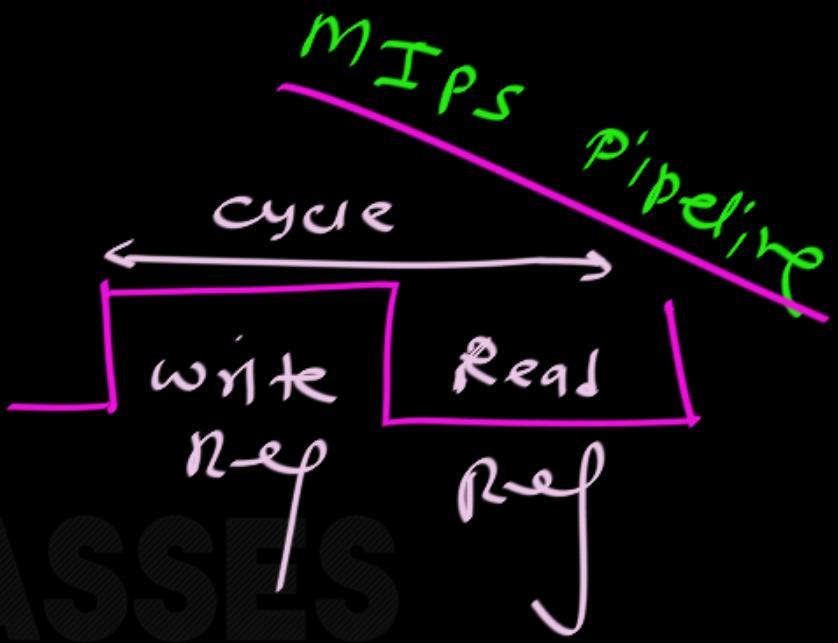
This RAW Dep is a RAW Hazard in MIPS pipeline.

1: $R_1 \leftarrow R_2 + R_3$

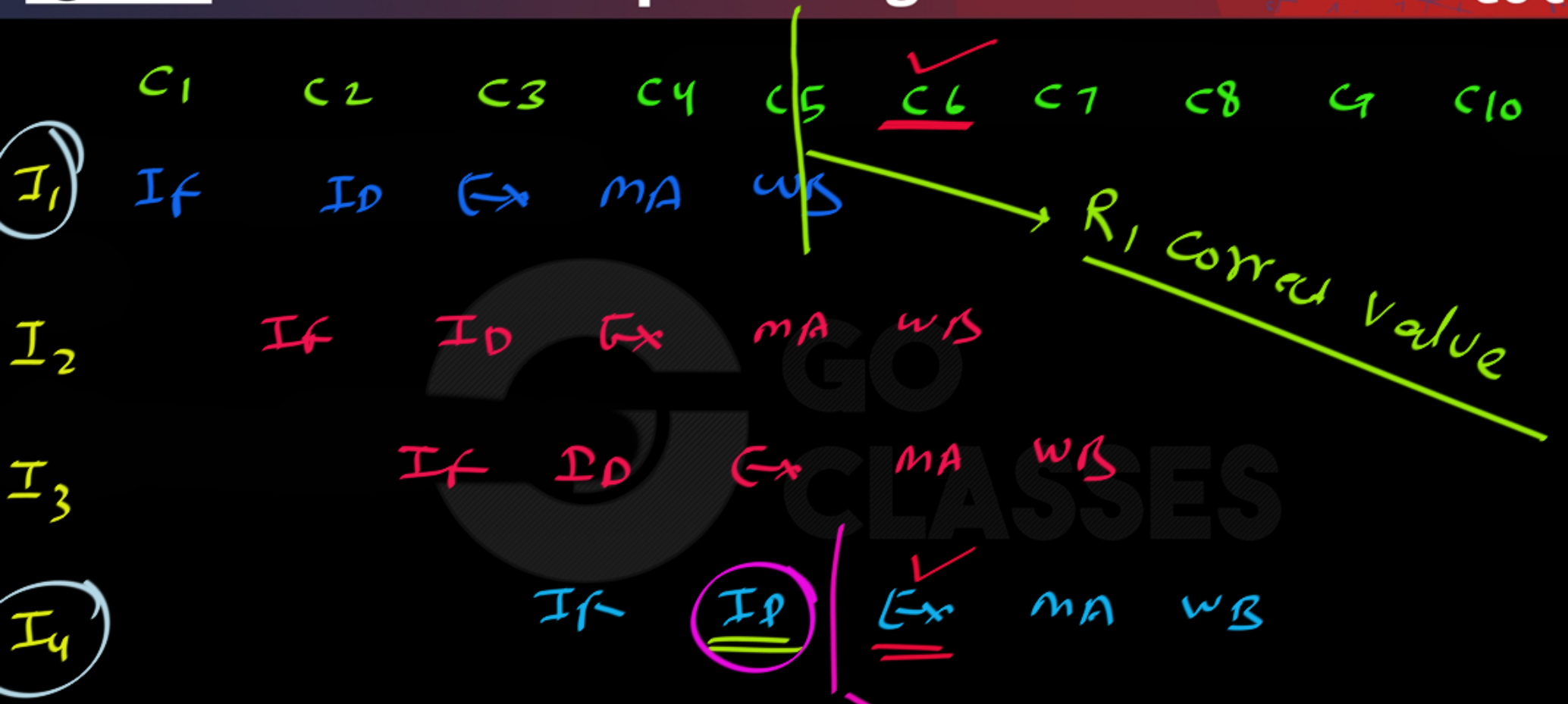
2: $R_4 \leftarrow R_5 + R_6$

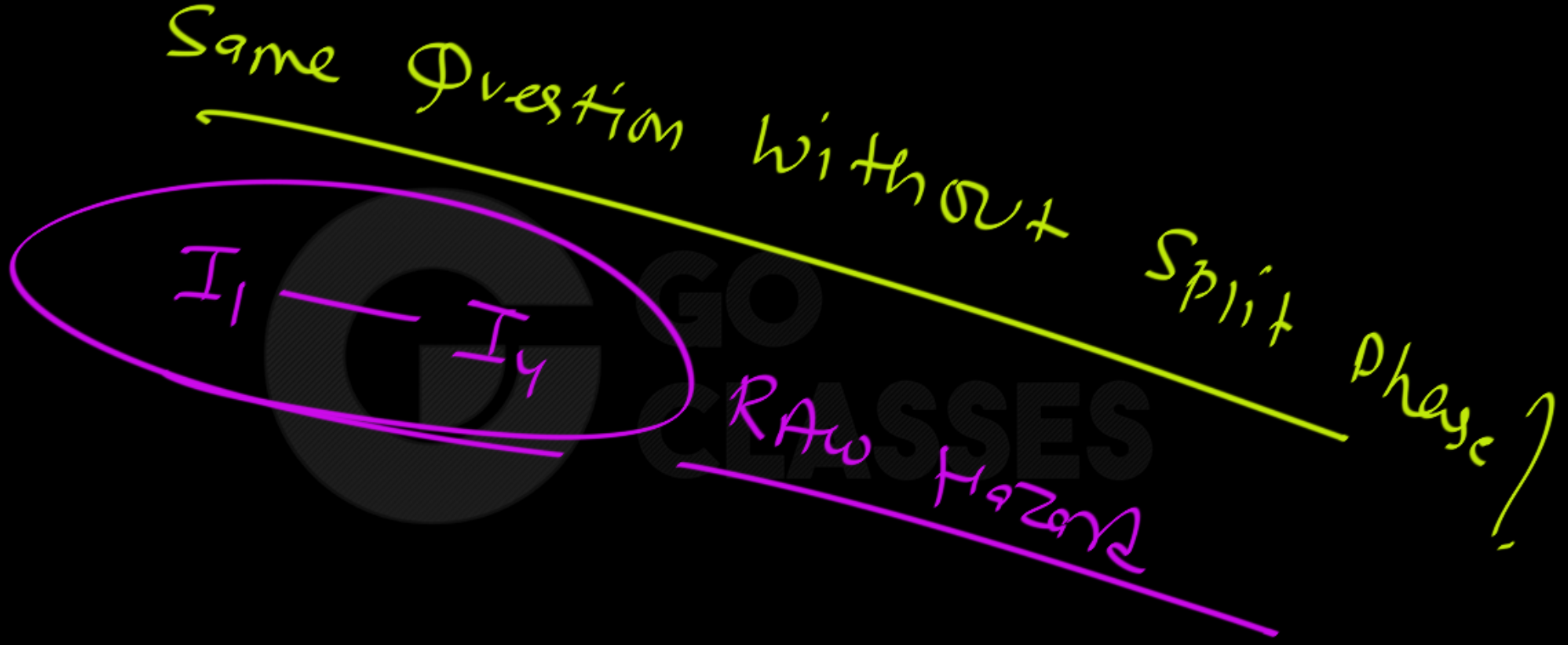
3: $R_7 \leftarrow R_9 + R_8$

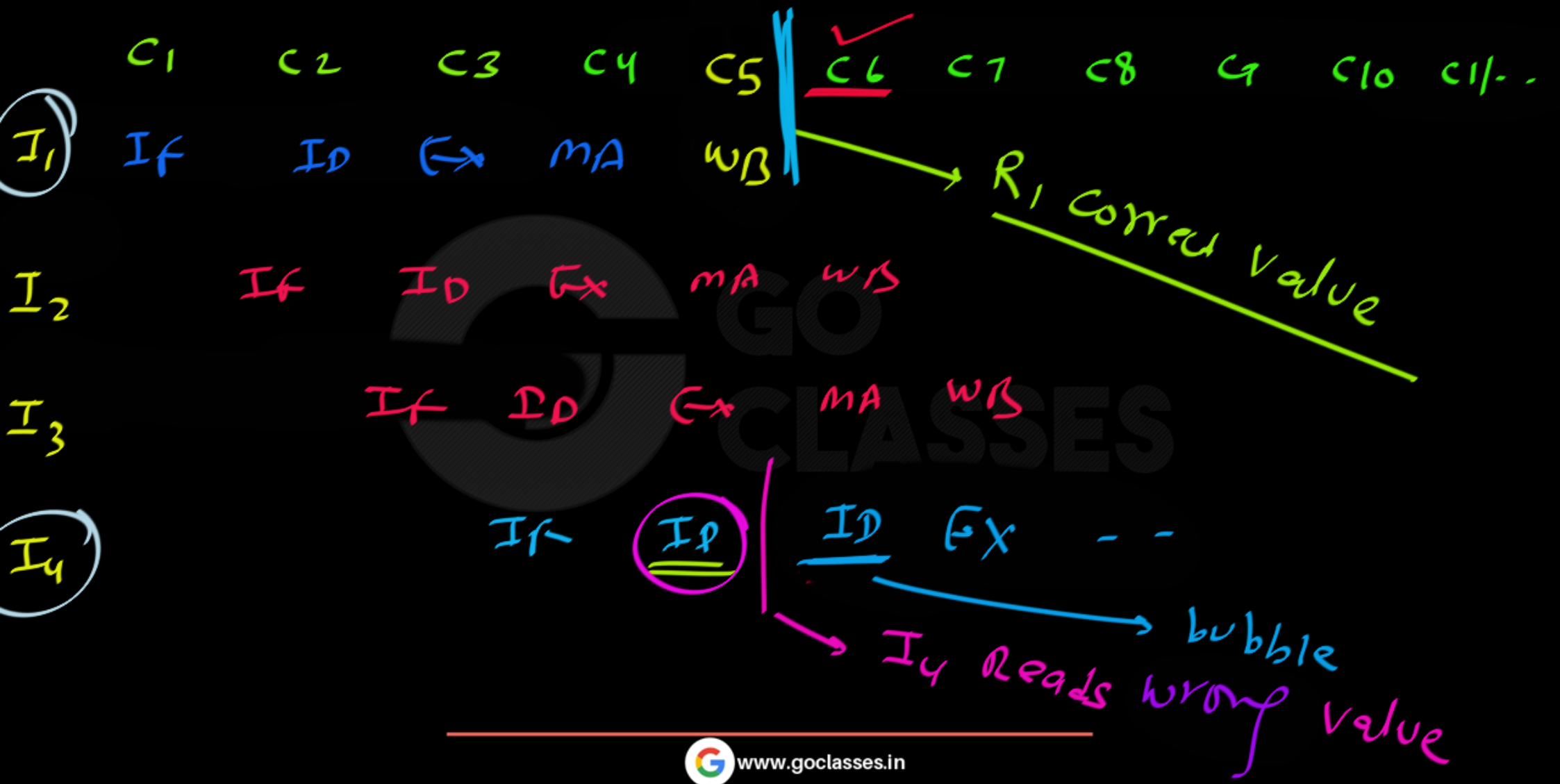
4: $R_{11} \leftarrow R_{12} + R_1$

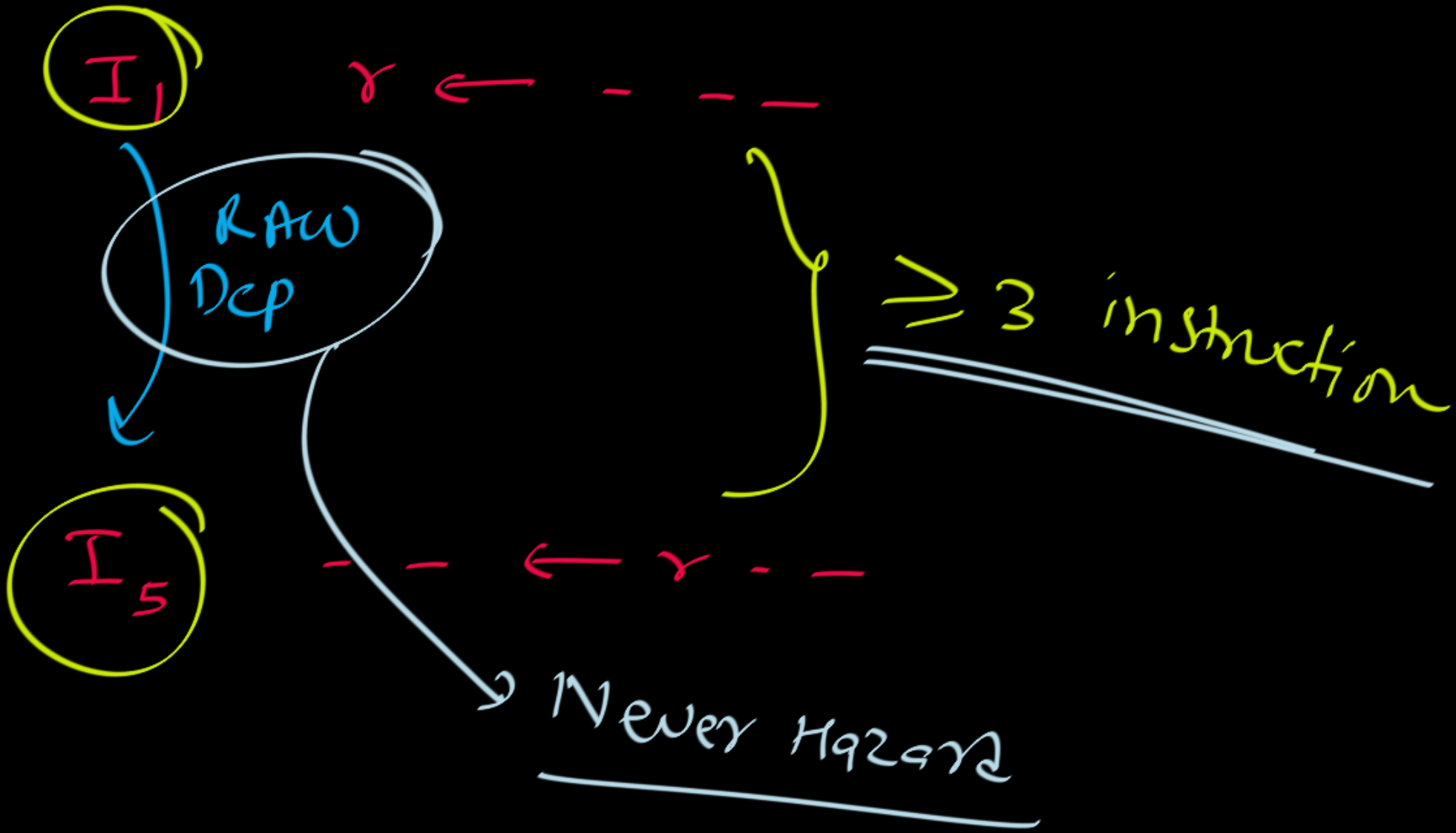


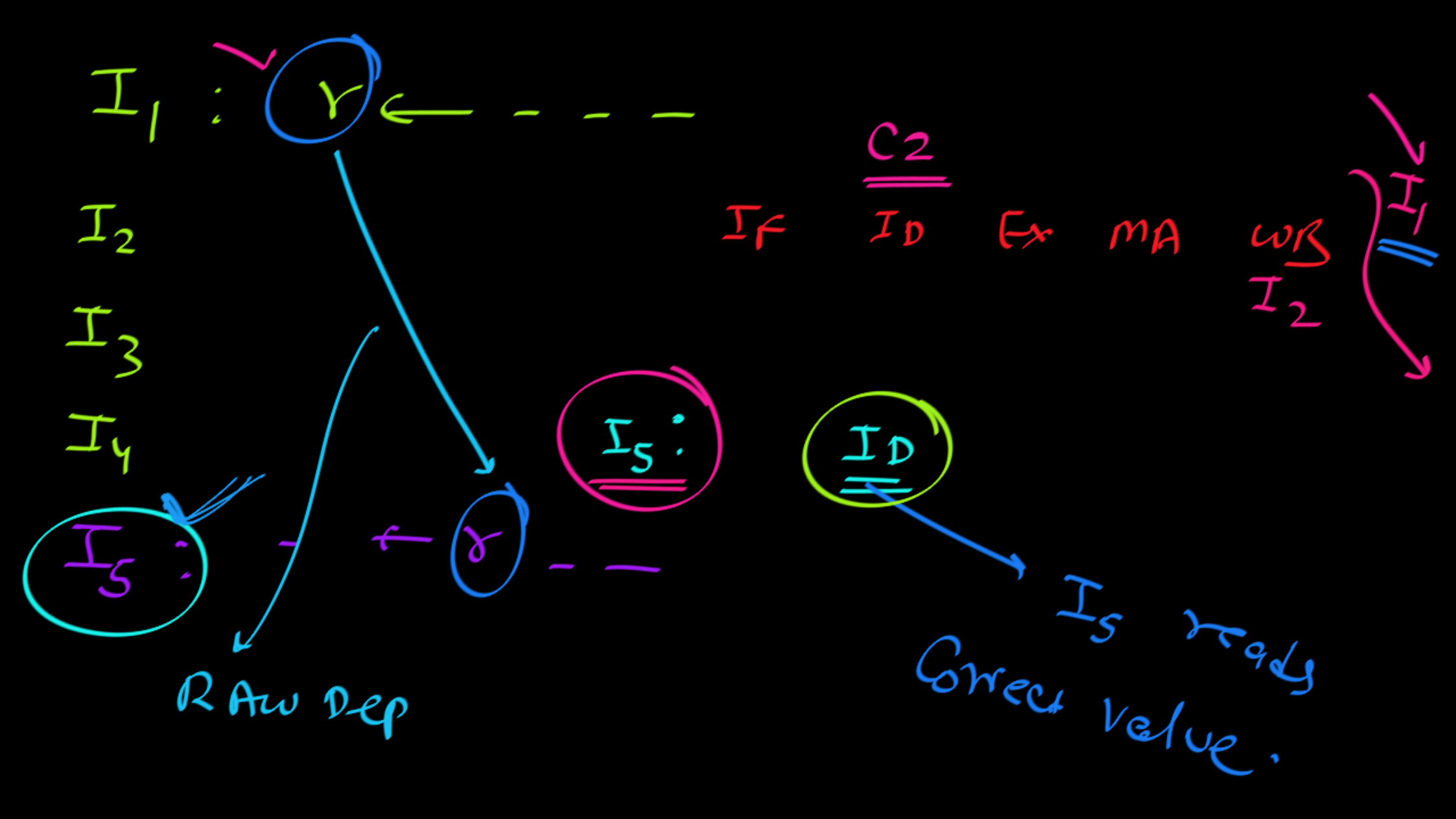
Is this RAW Hazard on MIPS pipeline? No


C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11...
I1 IF ID EX MA WB
R1 correct value
I2 IF ID EX MA WB
I3 IF ID EX MA WB
I4 IF ID EX MA WB
I4 Reads correct value


Same Question without Split phase?
I1 — I4
RAW Hazard


C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11
I1 IF ID EX MA WB
R1 correct value
I2 IF ID EX MA WB
I3 IF ID EX MA WB
I4 IF ID ID EX
bubble
I4 Reads wrong value


I_1
r ← - - -
RAW Dep
≥ 3 instruction
I_5
- - ← r - -
Never Hazard


I1 : r
I2
I3
I4
I5 : s
I5 :
ID
C2
IF ID Ex MA WB
I2
H
RAW Dep
I5 ready
Correct value.

In any pipeline ;

Some RAW Dep are Hazards ;

Some not ;

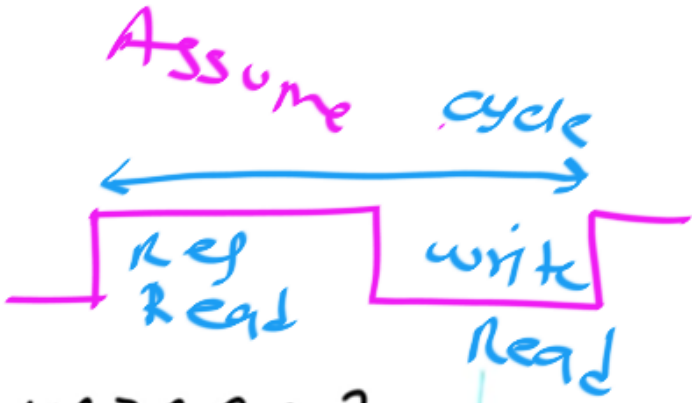
DEPENDENCIES AND HAZARDS QUIZ

FETCH	READ REG	MEM
DECODE	ALU	WR REG

		DEPENDENCE ?	HAZARD ?
I_1	ADD R1, R2, R3		
	SUB R5, R1, R4		
	DIV R6, R1, R7		
	MUL R7, R1, R8		

DEPENDENCIES AND HAZARDS QUIZ

FETCH DECODE	READ REG ALU	MEM WR REG
	SUB	ADD



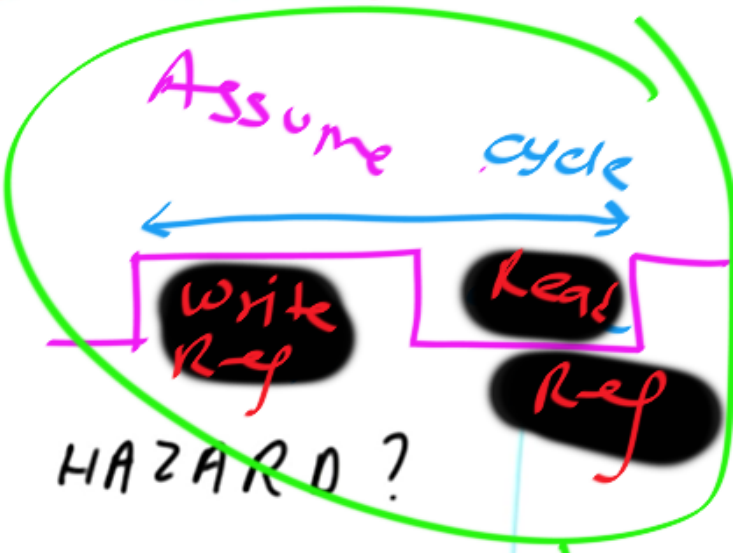
		RAW DEPENDENCE ?	HAZARD ?
I_1 ADD	R1, R2, R3		
SUB	R5, R1, R4	✓	✓
DIV	R6, R1, R7	✓	✗
MUL	R7, R1, R8	✓	✗

DEPENDENCIES AND HAZARDS QUIZ

FETCH DECODE	READ REG ALU	MEM WR REG
	SUB	ADD

RAW

			DEPENDENCE ?	HAZARD ?
I_1	ADD	R1, R2, R3		
	SUB	R5, R1, R4	✓	✗
	DIV	R6, R1, R7	✓	✗
	MUL	R7, R1, R8	✓	✗



Split Phase

DEPENDENCIES AND HAZARDS QUIZ

FETCH / DECODE	READ REG / ALU	MEM / WR REG
	SUB	ADD

		RAW DEPENDENCE ?	HAZARD ?
I_1 ADD	R1, R2, R3		
SUB	R5, R1, R4	✓	
DIV	R6, R1, R7	✓	
MUL	R7, R1, R8	✓	

DEPENDENCIES AND HAZARDS QUIZ

FETCH DECODE	READ REG ALU	MEM WR REG

RAW

		DEPENDENCE ?	HAZARD ?
I_1 ADD	R1, R2, R3		
SUB	R5, R1, R4	✓	
DIV	R6, R1, R7	✓	
MUL	R7, R1, R8	✓	

DEPENDENCIES AND HAZARDS QUIZ SOLUTION

FETCH DECODE	READ REG ALU	MEM WR REG

MUL

		DEPENDENCE ?	HAZARD ?
ADD	R1, R2, R3		
SUB	R5, R1, R4	✓	✓
DIV	R6, R1, R7	✓	/
MUL	R7, R1, R8	✓	/

Data hazards

- These exist because of pipelining

- Why do they exist???
  - Pipelining changes when data operands are read, written
  - Order differs from order seen by sequentially executing instructions on un-pipelined machine

- Consider this example:
  - ADD R1, R2, R3
  - SUB R4, R1, R5
  - AND R6, R1, R7
  - OR R8, R1, R9
  - XOR R10, R1, R11

All instructions after ADD use result of ADD

ADD writes the register in WB but SUB needs it in ID.

This is a data hazard

Note:

by default; Don't Assume Split Phase.

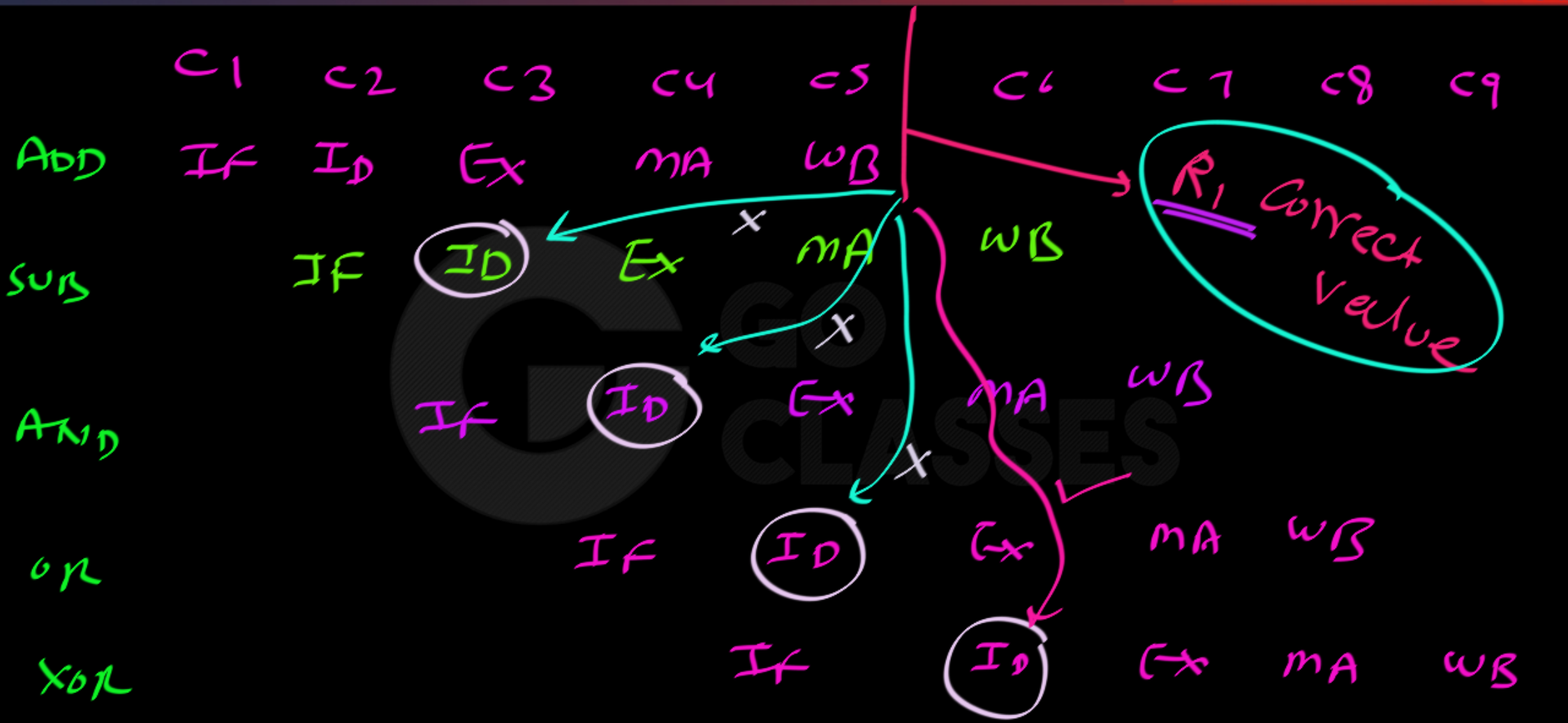
- Consider this example:
  - ADD R1, R2, R3
  - SUB R4, R1, R5
  - AND R6, R1, R7
  - OR R8, R1, R9
  - XOR R10, R1, R11

How many RAW Hazards in MIPS pipeline?

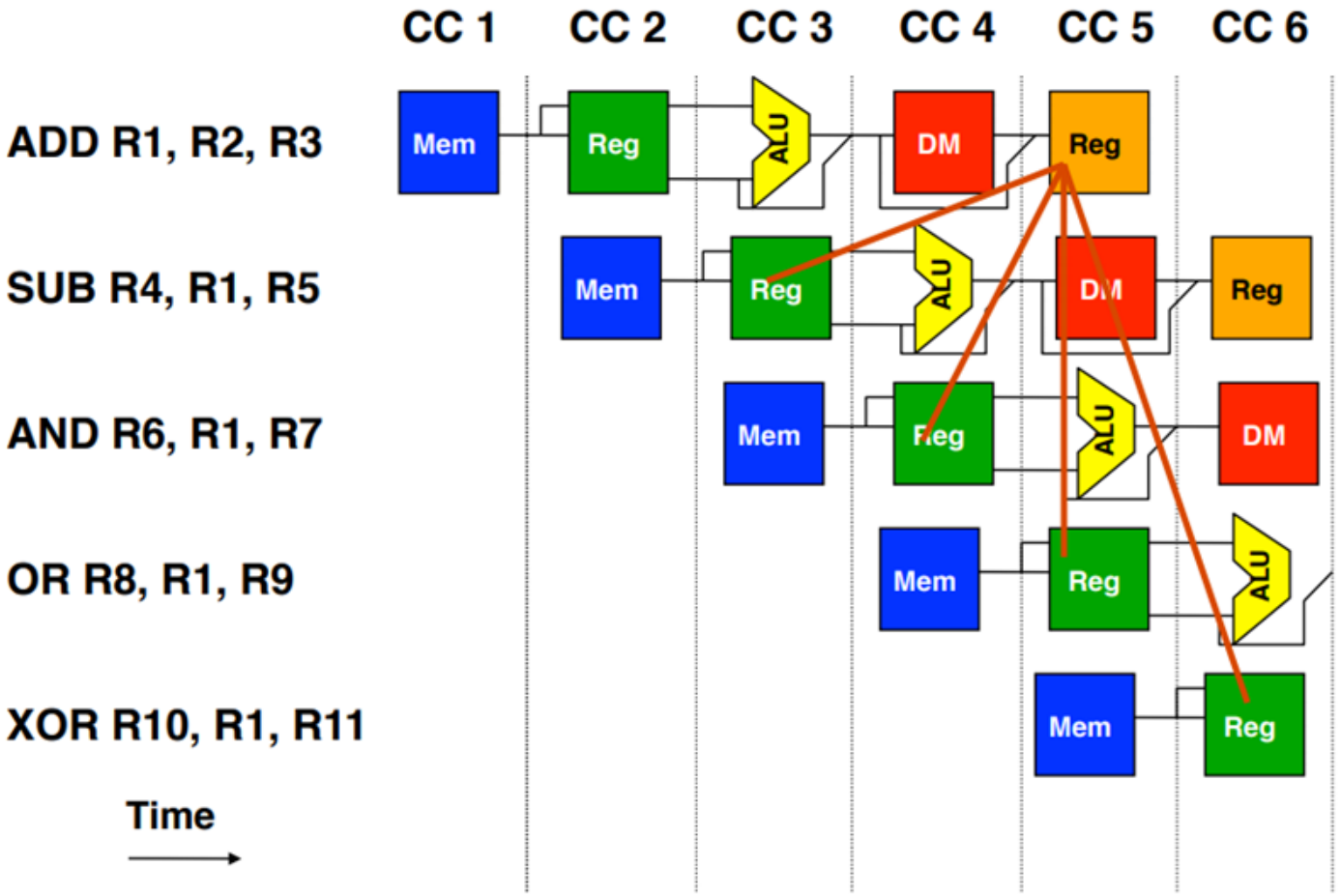
RAW Dep ! ✓ 4 ✓

- Consider this example:
  - ADD R1, R2, R3
  - SUB R4, R1, R5
  - AND R6, R1, R7
  - OR R8, R1, R9
  - XOR R10, R1, R11

How many RAW Hazards in MIPS pipeline ? ✓ = 3


C1 C2 C3 C4 C5 C6 C7 C8 C9
ADD IF ID EX MA WB
SUB IF ID EX MA WB
AND IF ID EX MA WB
OR IF ID EX MA WB
XOR IF ID EX MA WB
X
X
X
R1 Correct Value

Illustrating a data hazard



ADD instruction causes a hazard in next 3 instructions
b/c register not written until after those 3 read it.

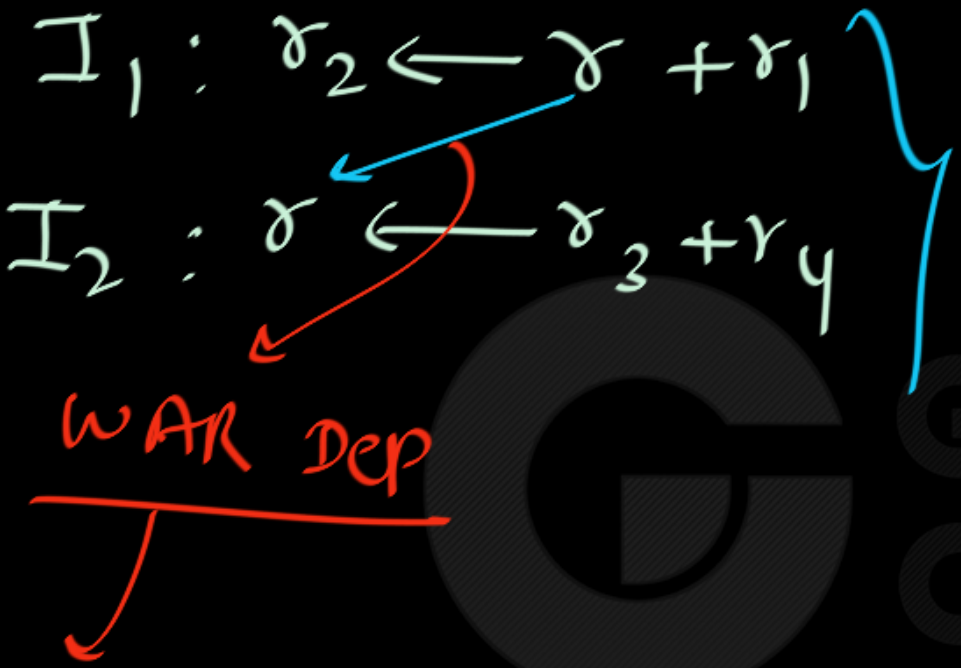
Detecting Data Dependencies

- Dependencies: Given two instructions, *i* and *j* (*i* occurs before *j*).
- We say a *dependence* exists between *i* and *j* if *j* reads the result produced by *i*, and there is no instruction *k* which occurs between *i* and *j* and that produces the same result as *i*.
- We call a data dependence a *hazard* when an instruction tries to read a register in stage 2 (ID) and this register will be written by a previous instruction that has not yet completed stage 5 (WB).
- This is sometimes called a read-after-write hazard.

Next:

Not All Data Dependencies are Data Hazards

Q: Prove that WAR Dep are Never Hazard in MIPS Pipeline?


$I_1 : r_2 \leftarrow r + r_1$
$I_2 : r \leftarrow r_3 + r_4$
WAR Dep
Is it Hazard?

I_1 : $r_2 \leftarrow r + r_1$

I_2 : $r \leftarrow r_3 + r_4$

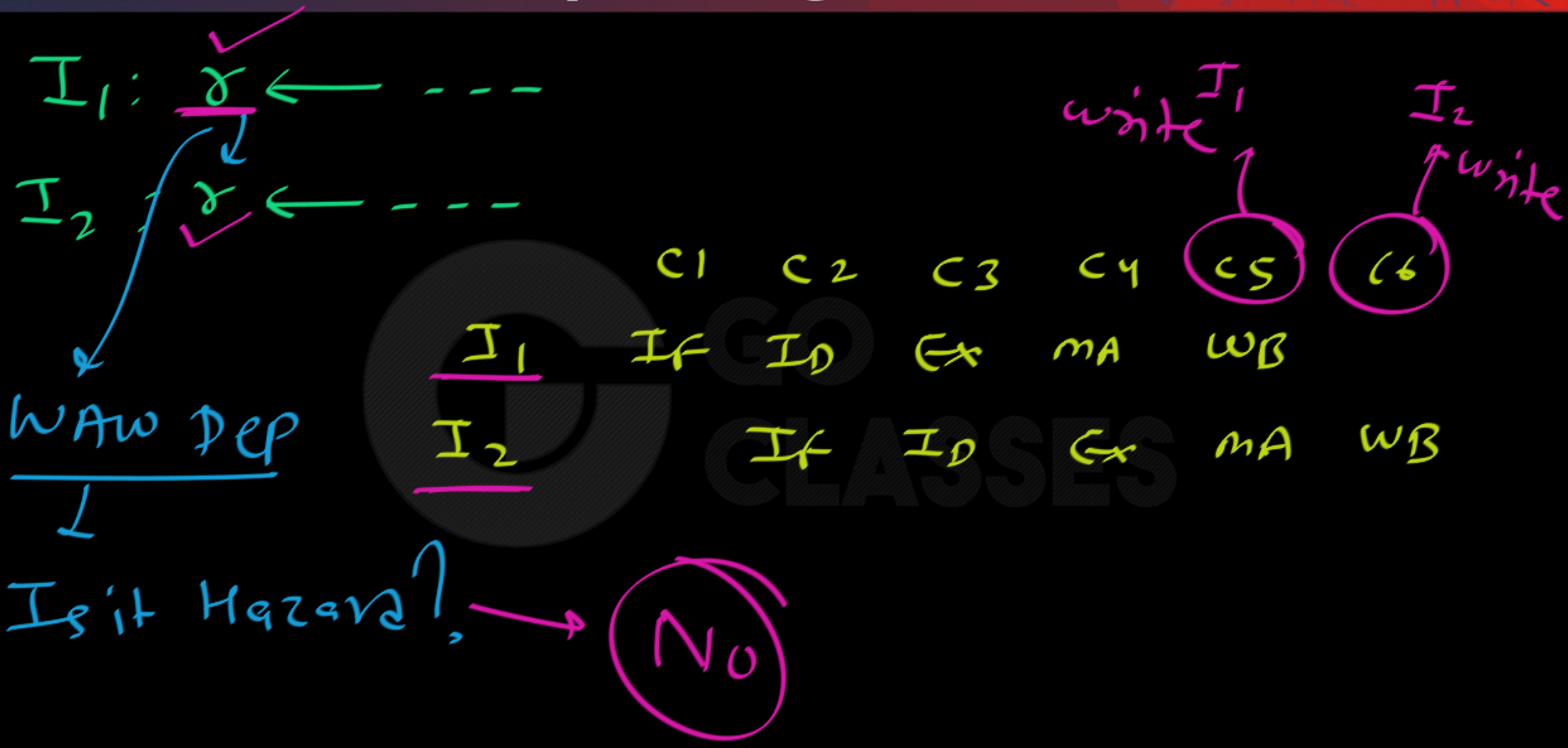
WAR Dep

Is it Hazard?

	C_1	C_2	C_3	C_4	C_5	C_6
I_1 :	IF	ID	EX	MA	WB	
I_2 :		IF	ID	EX	MA	WB

No

Q: Prove that WAW Dep are Never Hazard in MIPS Pipeline?


I1: R ←
I2 R ←
WAW Dep
I
write I1
I2 write
C1 C2 C3 C4 C5 C6
I1 IF ID EX MA WB
I2 IF ID EX MA WB
Is it Hazard? → No

Note:

Every Inst going through 4 stages

1. Not All Data Dependencies are Data Hazards.
2. WAW, WAR Dependencies are NEVER a Hazard in any Single-Issue, In-Order Pipeline.