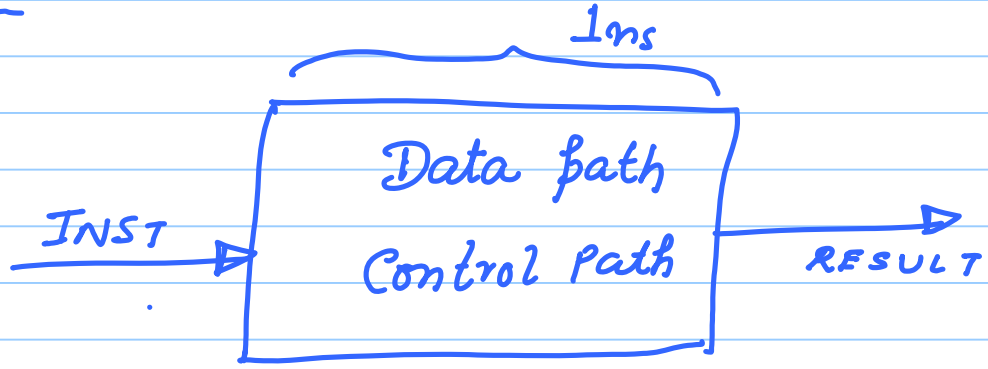
Oct-4

Note Title 04-10-2012

Pipelining



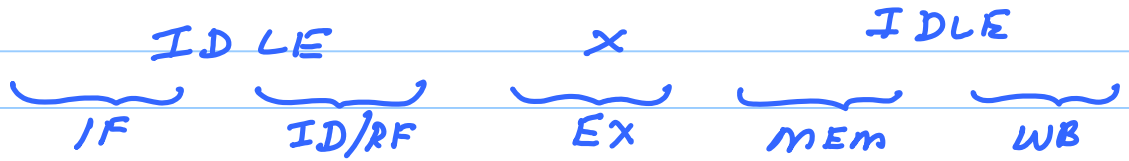
Register File.

→ Memory (Data & Instruction)

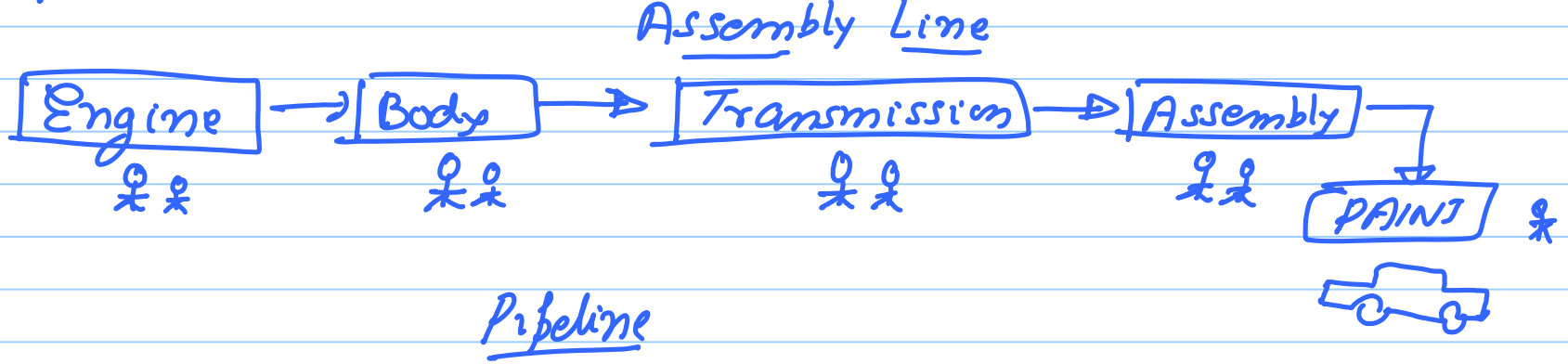
MATHS

Inst. per second → $\frac{1}{1\,ns} = 10^9 = 1$ billion

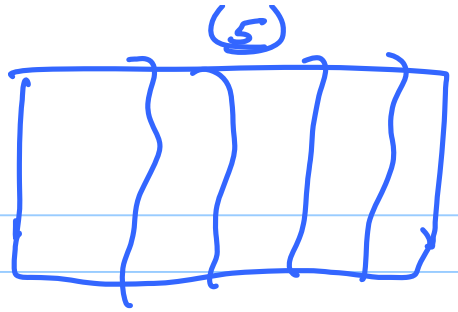
Wastage!

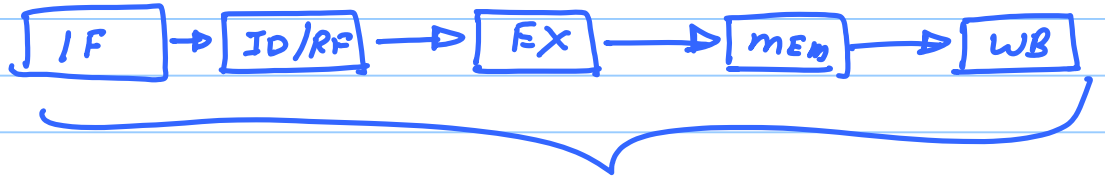


Example : CAR FACTORY



Pipeline





Instruction Pipeline

IF — Instruction Fetch

ID/RF — Instruction Decode

+

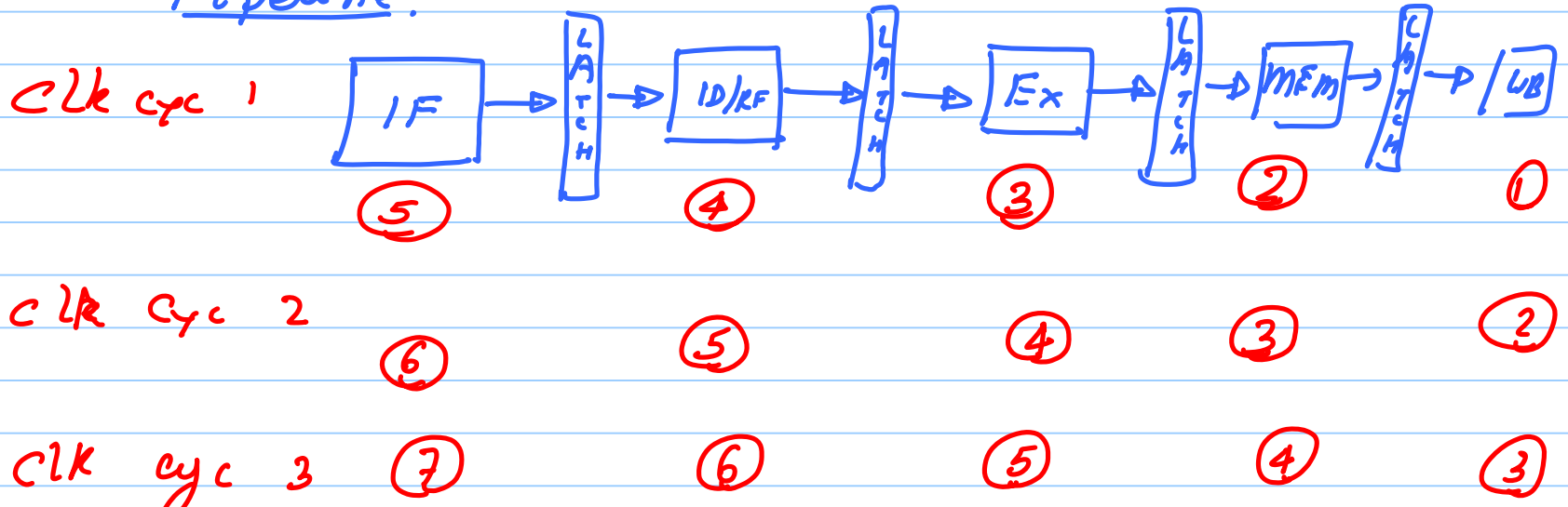
Register File Read.

Ex — Execute (ADD/MULTIPLY)

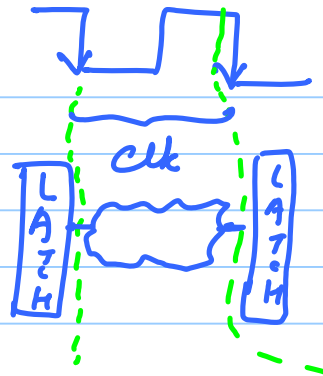
MEM — Memory Access (Load/Store)

WB — Write result back to register.

Pipeline.



How does one stage work:



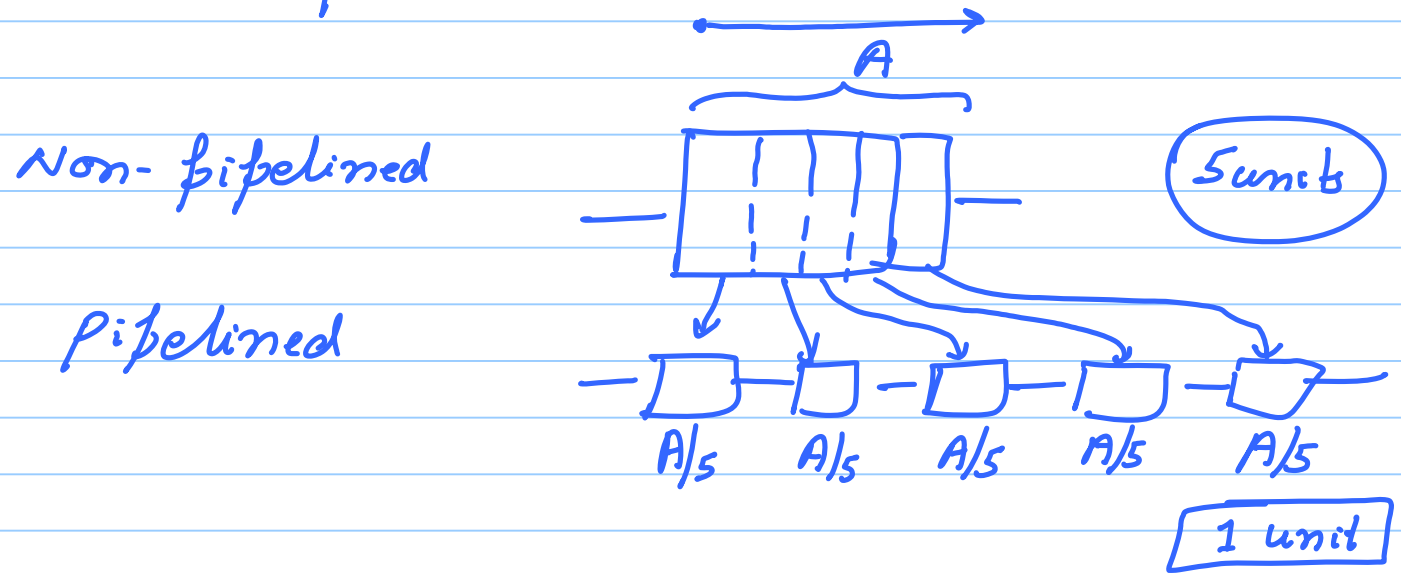
New Data is visible

written into the dest. latch

$$P = \frac{IPC \times f}{\#\ insts.}$$

Assume: Pipeline has k stages

	IPC	f	# insts
Non-Pipelined	~~SAME~~	1	~~SAME~~
Pipelined.		k	



A → Algorithmic work.

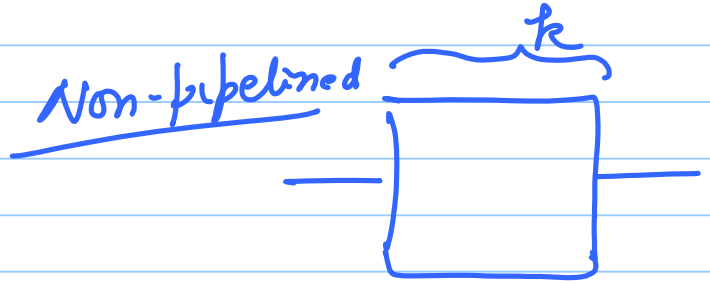
clock cycle time $\propto A$

⇒ A pipelined processor can have a smaller clock cycle

Speedup with a pipeline:

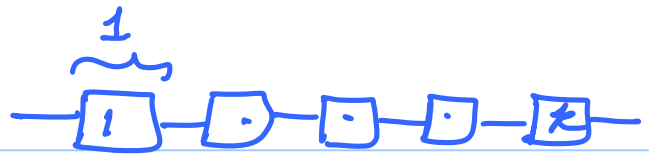
Assume: k stages.

n instructions.



$Tot\_Time_{NP} = nk$

Pipelined



$$\text{Tot\_time}_p = n + k - 1$$

$$\text{Speedup}_{RATIO} = \frac{nk}{n+k-1} = \frac{k}{1+\left(\frac{k-1}{n}\right)}$$

$$\left(\text{Speedup}_{RATIO}\right)_{n \to \infty} = k$$

More you pipeline → More is the speedup

Question

What stops me from having a thousand stage pipeline?

	MIPS Processor	ARM-9	Core-2-Duo	Intel P-Extreme
Pipeline Depth.	5	8	12	22

What limits the number of pipeline stages?

* You cannot arbitrarily increase the frequency.

→ Power

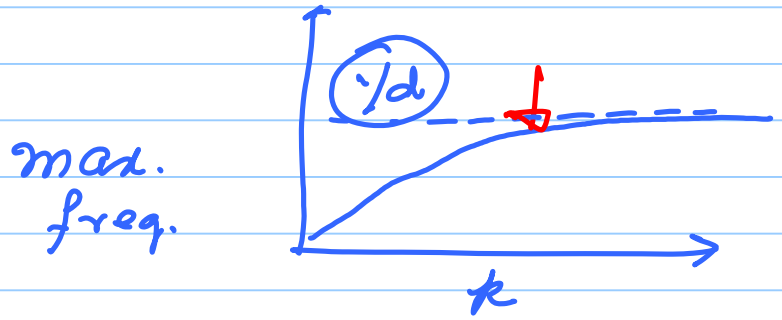
→ Algorithmic wk per stage $\left(\frac{A}{k}\right)$

Latch delay - d.

$$cc\_time \geq \frac{A}{k} + d$$

$$f = \frac{1}{cc\_time} \leq \frac{1}{\frac{A}{k} + d}$$

$$f \leq \frac{k}{A + kd}$$



After a certain point, frequency cannot increase further.

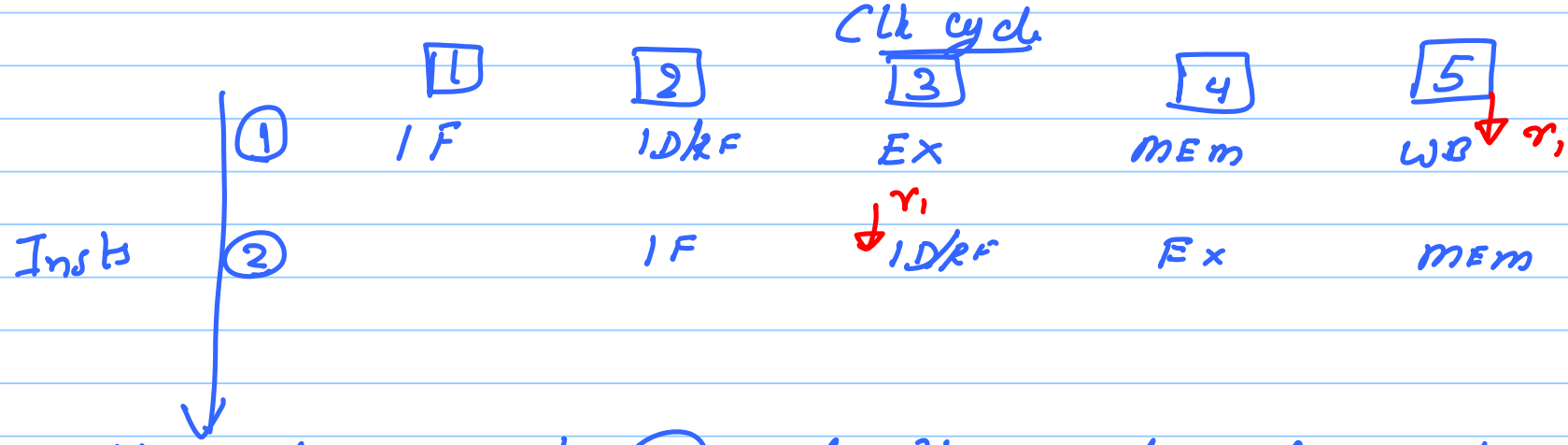
* There are several some other limiters

Hazards

HAZARDS

1) add r_1, r_2, r_3

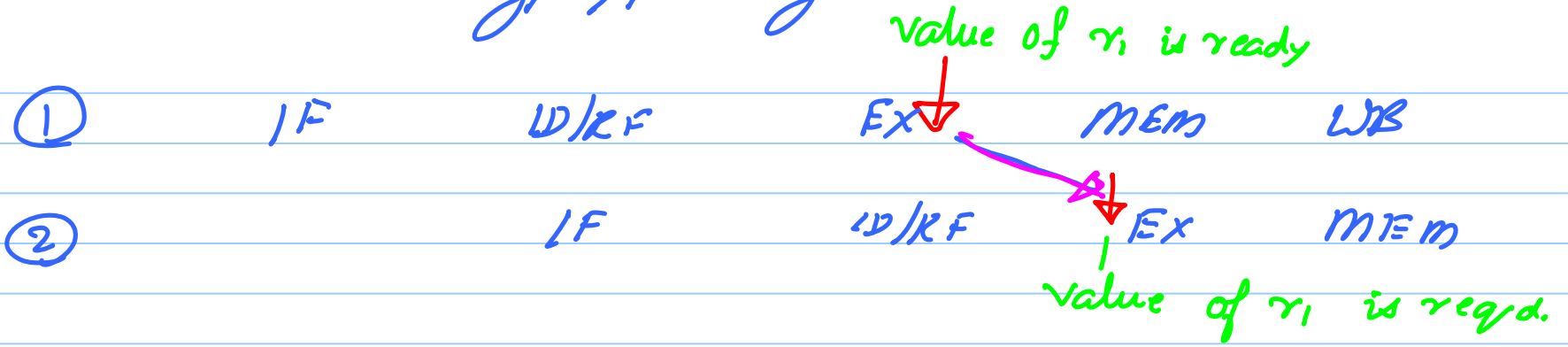
2) add r_4, r_1, r_6



How does inst. (2) get the value of register r_1

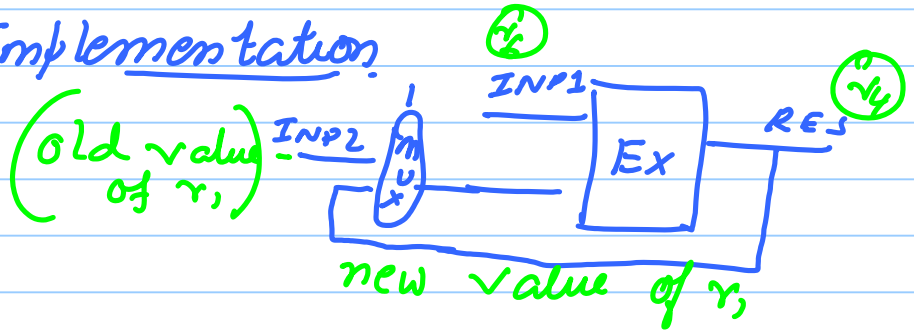
Data Hazard

FIX : Forwarding / Bypassing.



This will ensure that instructions 1 and 2 can execute consecutively

Implementation

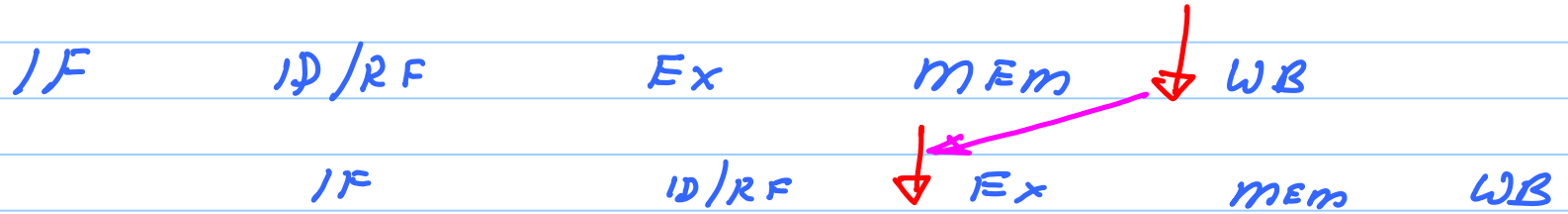


Can we do a forwarding trick all the time?

① lw r_1 20(r_2)

② add r_4, r_1, r_6

} LOAD-USE HAZARD



Forwarding.

Producer Instruction ① ⟶ Consumer Instr ②

1) Find the earliest point at which the producer is ready

2) Find the latest point at which the consumer needs the data

3) Draw a line from:

Production point → consumption point

4) IF THE LINE

- goes to bottom right (ahead in time) (can forward)
- goes to the bottom left (behind in time) (cannot forward)

How do you fix a load-use hazard?