

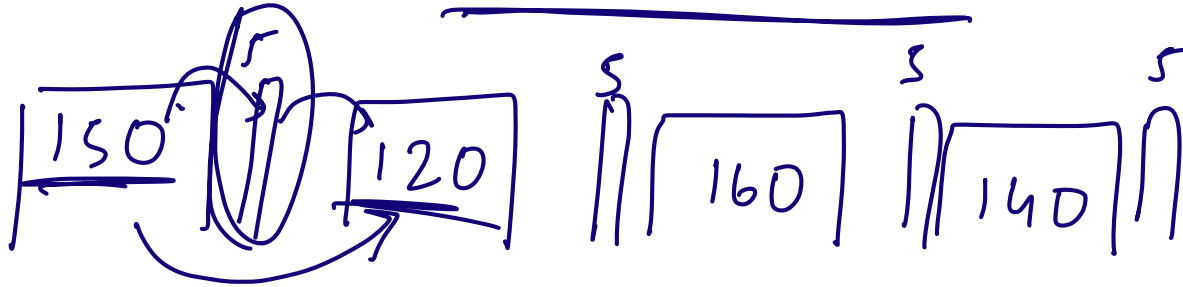
Stage delay 1

μs	ns	ps
10^{-6}	10^{-9}	10^{-12}

GHz

MHz.

A 4 stage pipeline has stage delays as 150, 120, 160 and 140 ns. Latches are used between stages and have a delay of 5 ns each. Assuming constant clock rate, total time taken to process 1000 data items will be?



clock cycle time $160 \text{ ns} + 5 \text{ ns} = 165 \text{ ns}$.

1 instruction = 4 cycles $\times 165 \text{ ns} \times 1 \text{ instruction}$.

999 instructions = 1 cycle $\times 165 \text{ ns} \times 999 \text{ in}$
 $165 \times 4 + 165 \times 999 = 165000 + 495$

Stage delay 2

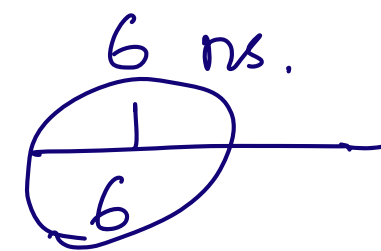
$$= 165495 \text{ ns}$$

$$= 165.5 \text{ us.}$$

The stage delays in a 4 stage pipeline are 5, 6, 4 and 5 ns. The second stage with
is replaced with 2 functionally equivalent stages, of delay 3 and 4 ns. The
throughput increase of the pipeline is?

4 stage pipeline. 5 6 4 5 ns.

clock period
 steady state through



$$\left(\frac{\frac{1}{5} - \frac{1}{6}}{\frac{1}{6}} \right)$$

5 stage pipeline 5 3 4 4 5.

clock period 5 ns through $\frac{1}{5}$

Stage delay 3

1st instruction \rightarrow # stages matter

Consider the following processors. Assume latches have 0 latency. Which processor has the highest clock frequency?

(i) 4 stage pipeline with stage latencies 1, 2, 2, 1 ns 2 ns.

$4 \times 2 \text{ ns} = 8 \text{ ns}$

(ii) 4 stage pipeline with stage latencies 1, 1.5, 1.5, 1.5 ns 1.5 ns.

$1.5 \text{ ns} \times 4 = 6 \text{ ns}$

(iii) 5 stage pipeline with stage latencies 0.5, 1, 1, 0.6, 1 ns 1 ns.

1 ns

(iv) 5 stage pipeline with stage latencies 0.5, 0.5, 1, 1, 1.1 ns 1.1 ns.

1.1 ns.

Stage delay 4

Calculate cycle time, latency, throughput of pipelined vs. non-pipelined processors. Latch latency is 20 ps. If you could split any stage for the pipelined into two equal halves, which one would you choose? What would be the new cycle time, latency, throughput?

420 ps

275 275.

	fetch	decode	execute	memory	writeback
a.	300ps	400ps	350ps	550ps $\frac{1}{570}$	100ps
b.	200ps $\frac{1}{220}$	150ps	100ps	190ps	140ps

latency $6 \times 40 \times 10^{-12}$ seconds.

throughput $\frac{1}{420 \times 10^{-12}}$

Stage delay 5

Clocks per instruction

A non-pipelined processor has a clock rate of 2.5 GHz and average CPI of 4. The processor is now pipelined to have 5 stages, but the clock speed is reduced to 2 GHz. Assuming no stalls in the pipelined processor, what is its speedup over non-pipelined?

speedup

time for each pipelined instruction
non pipelined.

steady state

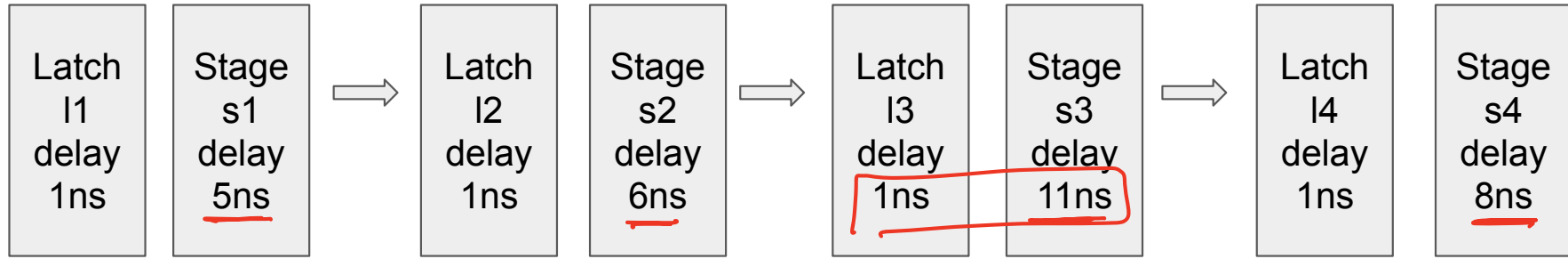
1 instruction.

$$\begin{array}{r} \textcircled{1} \leftarrow 5 \times \frac{1}{2 \times 10^9} \\ \hline 4 \times \frac{1}{2.5 \times 10^9} \end{array} \quad \begin{array}{r} 1 \times \frac{1}{2 \times 10^9} \\ \hline \textcircled{4} \times \frac{1}{2.5 \times 10^9} \\ \hline \frac{2.5}{8} \end{array}$$

Stage delay 6

→ no stalls, no hazards.

What is the speedup of this pipeline under ideal conditions in steady state, compared to non-pipelined?



non pipelined. 1 instruction completes in

$$5 + 6 + 11 + 8 = \underline{30 \text{ ns}}$$

speedup.

$$\frac{30}{12}$$

pipelined.

1 instruction (in steady state) completes in 12 ns.

Stage delay 7

D1 has 5 pipeline stages with delays 3, 2, 4, 2, 3 ns. D2 has 8 pipeline stages each with 2 ns execution time. How much time can D2 save over D1, in executing 100 instructions?

not steady state only.

$$1 \text{ instruction} \times 5 \times 4 = 20 \text{ ns.}$$

$$99 \times 4 \text{ ns.} = 396 + 20 \text{ ns.}$$

$$\leftarrow \underbrace{8 \times 2}_{\substack{\text{1st instruction} \\ \text{warmup}}} + \underbrace{2 \times 99}_{\substack{\text{steady state} \\ \text{remaining}}}$$

Stage delay 8

A non-pipelined single cycle processor operating at 100 MHz, is converted to a synchronous pipelined processor with 5 stages requiring 2.5, 1.5, 2, 1.5 and 2.5 ns. The delay of a latch is 0.5 ns. Find the speedup of the pipelined processor for a large number of instructions.



Data hazard 1

lw R15, 0(R2)

add R14, R15, R15

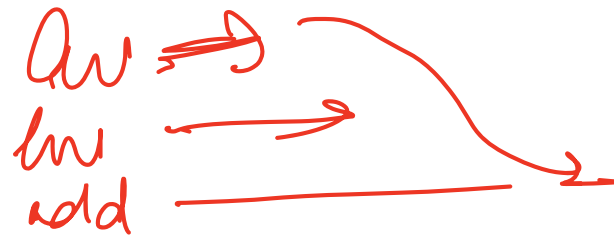
lw R16, 4(R2)

add R17, R16, R16

lw

add

lw
add



t1 t2 t3 t4 t5 t6 t7 t8

IF ID EX MEM WB
IF ID ID EX MEM WB

add

nop

Where do we need NOPs in a standard 5 stage pipeline with forwarding? Can the compiler do any instruction reordering to reduce NOPs?

Data hazard 2

An otherwise ideal 5-stage pipeline has a 1 cycle load-to-use delay (that is you have to wait 1 cycle between a load, and an ALU operation that uses the result of the load). A program has 20% loads, but the compiler can find independent instructions to put after the load, for only half of them. What is the slowdown due to NOPs?

1 cycle load-to-use delay.

100 instructions
no load instructions
100 clock cycles.

100 instructions
80 20 loads 10
10
90 cycles - 80 normal
- 10 loads (compris)

Data hazard 3

+ 20 cycles

10 loads + 10 nops

~~110~~

10% slowdown

How long will the following code take on a 5 stage pipeline, with appropriate forwarding and NOPs?

add R5, R5, R7

lw R6, 100(R7)

sub R7, R6, R8

load to use delay with forwarding -

t8

t1 t2 t3 t4 t5 t6 t7 t8

add IF ID EX MEM WB

lw - 1



Data hazard 4

Fill in the pipeline, with and without forwarding:

add R4, R5, R2

lw R15, 0(R4)

sw R15, 4(R2)

memory

sub/add

ALU/EX

mem stage

lw

MEM

LS

MEM

7 cycles / no stalls

Control hazard 1

What states do 1-bit and 2-bit branch predictors go through for the following branch patterns? Assume that they start in strongly not taken 00 state.

History: NNNTNNN
 ✓✓✓x✓✓

History: NTNTNTN
 x x x x x x

1 bit. 0 ← 5/7 correct.

1

2 predictor

00 x strongly not taken
 ✓ ✓ ✓ x ✓ ✓
 N N N T N N N

01 weakly x
 10 weakly taken ✓ 6/7 correct

0 (1/7)

x x x
 N T N T N T N

00
 01

4/7 correct

L11 strombolian ✓