

1) (a) $T_{\text{pipelined}/\text{min}} = T_{\text{longest-stage}/\text{min}} + d_{\text{register}} = 2\text{ns} + 0.1\text{ns} = \underline{\underline{2.1\text{ns}}}$

(b) In the steady state, ⁱⁿ every 5 clocks, 4 instructions would ~~recaia~~ be completed.

$$\therefore \text{CPI}_{\text{pipelined}} = \frac{5 \text{ cycles}}{4 \text{ instructions}} = \underline{\underline{1.25}}$$

(c) Note that in a non-pipelined machine, there is no possibility of any stalls. Suppose, a program consists of "I" instructions.

\therefore time taken to execute the program on the non-pipelined machine:

$$\tau_{\text{non-pipelined}} = I * \text{CPI} * T_{\text{non-pipelined}} = I * 1.7 \text{ ns}$$

Similarly, $\tau_{\text{pipelined}} = I * 1.25 * T_{\text{pipelined}} = I * 1.25 * 2.1 \text{ ns}.$

$$\therefore \text{Speedup} = \frac{\tau_{\text{non-pipelined}}}{\tau_{\text{pipelined}}} = \frac{7I}{1.25 * 2.1 * I} = \underline{\underline{2.67}}$$

(d) If the # of pipeline stages become infinite, then, the delay of the pipelined machine is only the delay of the registers, i.e.,

$$\lim_{\text{\# of stages} \rightarrow \infty} T_{\text{pipelined}} = d_{\text{register}} = 0.1\text{ns}$$

$$\therefore \text{Speedup}_{\text{ignoring extra \& cycles}} = \frac{7I}{1 * 0.1I} = 70.$$

$$\text{Speedup}_{\text{considering extra stall \& cycles}} = \frac{7I}{1.25 * 0.1} = 56$$

Both answers would be accepted.

→ This answer also makes sense since the clock cycle time-period is now almost zero (because of infinite # of pipeline stages). Hence, the only pipeline delay now is because of the register delay.

(a) When there is no branch misprediction, the only stalls are because of RAW hazards, which cannot be resolved (\because there is no forwarding), except those which can be resolved because of the nature of the register file, where WB \rightarrow ID register file reading is possible.

Loop: ld, a1, 0(a2)

addi a1, a1, 1
sd a1, 0(a2)

addi a2, a2, 4
sub a4, a3, a2

bnez a4, Loop

True
Dependencies

①: Cannot be resolved (even if there had been forwarding)

②: Cannot be resolved (\because there is no forwarding). However, by delaying instruction fetch for sd (see pipeline timing diagram below), they do not effectively affect the performance (WB \rightarrow ID "forwarding through register file" helps)

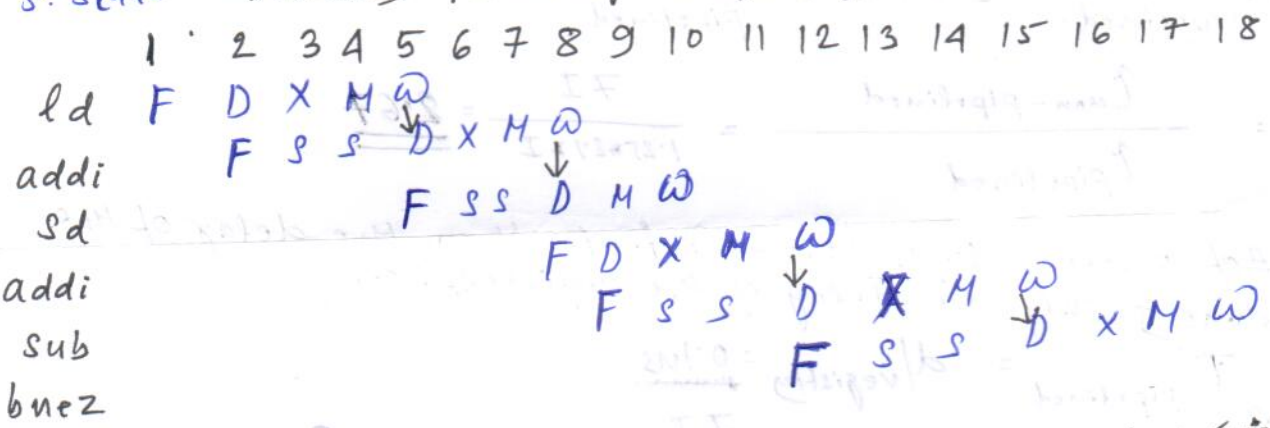
③: Same as for #2.

④: Same as for #2

Note that since branch outcomes are known in EX stage

there are 2 cycles penalty in case of branch mispredictions

S: stall \rightarrow denotes "forwarding through register file"



(b) In the 1st ϕ cycle, assuming the 1-bit branch predictor is in Not Taken state, there is a misprediction \Rightarrow 2 ϕ cycles penalty because of pipeline flush.

Again, in the last ϕ cycle, there is similarly a 2 ϕ cycle penalty.

The loop runs total $\frac{396}{4} = 99$ times

The middle $99 - 1 - 1 = 97$ iterations take 18 ϕ cycles each

\therefore total # of ϕ cycles = $97 * 18 + 20 + 20 = 1786$ ϕ cycles

Note: if the 2 cycle stall due to pipeline flushes in the last iteration is ignored, then # of ϕ cycles = 1784

Both answers are acceptable

