6.3 Pipelining

Name: _____ Class: _____ Date: _____ **Total: 24 marks**

KEY WORDS pipelining • CPI

Objective

Count pipeline cycles, price stalls into CPI, and classify hazards.

1 Key knowledge and worked examples

■ The formula

n instructions, k stages: $n+k-1$ cycles – five instructions take 9, not 25; at $n = 100$, 104 ✓. The stage time is the slowest stage: 250 ps in our machine ✓.

■ Worked example: CPI with stalls

1000 instructions, 25% loads, half of them load-use: stalls = 125; total 1000+125+4 = 1129 cycles; CPI 1.129 ✓.

2 Practice

2.1 Twenty instructions on a five-stage pipeline take how many cycles? [3]

2.2 Classify: `add R1,…` followed by `sub R4, R1, R1`. [3]

2.3 What does forwarding do, and when does it fail? [3]

3 Exam-style questions

3.1 A five-stage pipeline runs a program of 2000 instructions; 30% are loads and 40% of those are load-use pairs (one stall each); taken branches are 10% of instructions and each costs one bubble; forwarding covers everything else.

Compute the stall and bubble counts, the total cycles, and the CPI. Compute the same program's cycles unpipelined (single 900 ps cycle) and the speedup of the pipelined 250 ps machine, stating what the speedup tends to as n grows. [8]

3.2 These instructions run in order:

```
lw   R1, 0(R2)
add  R3, R1, R4
sub  R5, R6, R7
or   R8, R3, R5
```

State which pair creates a hazard that forwarding cannot remove, and the stall it forces. State which pairs forwarding covers at no cost. Give the extra cycles this sequence adds to the ideal pipeline count, and what changes if the `lw` and `add` swap places with an independent instruction between them. [7]

Solutions

2.1

- $20 + 5 - 1 = 24$ cycles ✓.

2.2

- A read-after-write hazard on R1 ✓.

2.3

- Routes an ALU result straight to the next ALU's inputs ✓. It fails when the producer is a load (data exists only after MEM): one stall ✓.

3.1

- Loads: 600; load-use stalls: $600 \times 0.4 = 240$ ✓.
- Bubbles: $2000 \times 0.10 = 200$ ✓.
- Total cycles $= 2000 + 240 + 200 + 4 = 2444$; CPI $= 2444/2000 = 1.222$ ✓.
- Unpipelined: 2000×900 ps $= 1.8$ ms; pipelined: 2444×250 ps $= 611$ µs – speedup $\approx 2.94\times$ ✓.
- As n grows the fill fades and the speedup tends to $900/250 = 3.6\times$ minus the stall tax ✓.

3.2

- `lw R1` feeding `add R3` is a load-use pair: forwarding cannot remove it – one stall ✓.
- `add R3` feeding `or R8` (ALU to ALU) forwards free; `sub R5` to `or` likewise; `lw`/`sub` are independent ✓.
- Extra cycles: 1 (the single load-use bubble) ✓.
- An independent instruction between them fills the bubble: zero extra cycles – scheduling is the cheaper fix ✓.

exercises.lol exercises.com.cn