CS202

Homework4

11911827 张皓淇

4.8

4.8.1

non-pipelined clock cycle: $250 + 350 + 150 + 300 + 200 = 1250ps$

pipelined clock cycle: $max(250, 350, 150, 300, 200) = 350ps$

4.8.2

total latency of an `LW` instruction in a non-pipelined processor: $1250ps$

total latency of an `LW` instruction in a pipelined processor: $350 * 5 = 1750ps$

4.8.3

Split the ID stage, because if we split other stage, the clock cycle will not change.

new pipelined clock cycle: $max(250, 175, 175, 150, 300, 200) = 300ps$

4.8.4

Only `sw` and `lw` will use data memory.

utilization: $20\% + 15\% = 35\%$

4.8.5

Only `alu` and `lw` will use the write-register port of the “Registers” unit.

utilization: $45\% + 20\% = 65\%$

4.8.6

clock cycle time:

single-cycle: $250 + 350 + 150 + 300 + 200 = 1250ps$

multi-cycle: $max(250, 350, 150, 300, 200) = 350ps$

pipelined clock cycle: $max(250, 350, 150, 300, 200) = 350ps$

execution time:

average execution time per instruction

single-cycle: $250 + 350 + 150 + 300 + 200 = 1250ps$

multi-cycle: $20\% * 5 * 350 + 80\% * 4 * 350 = 1470ps$

pipelined: $350ps$

4.9

4.9.1

Dependences:

1. The first instruction changes the value in `r1`, and the second instruction needs `r1`, the dependence type is EXE-EXE;
2. The first instruction changes the value in `r1`, and the third instruction needs `r1`, the dependence type is MEM-EXE;
3. The second instruction changes the value in `r2`, and the third instruction needs `r2`, the dependence type is EXE-EXE.

4.9.2

There is a data hazard between the first instruction and the second instruction, and a data hazard between the second instruction and the third instruction. (If the register read happens in the second half of the clock cycle and the register write happens in the first half of the clock cycle, then we only need to add two `nop` between the first instruction and the second instruction, and the second instruction and the third instruction. 4.9.5 is the same, we only need one `nop` between the second instruction and the third instruction.)

```
or r1,r2,r3
nop
nop
nop
or r2,r1,r4
nop
nop
nop
or r1,r1,r2
```

4.9.3

If there is full forwarding, there are no hazards and do not need to use `nop`.

```
or r1,r2,r3
or r2,r1,r4
or r1,r1,r2
```

4.9.4

total execution time without forwarding: $(5+8)*250=3250ps$

total execution time with full forwarding: $(5+2)*300=2100ps$

speedup: $3250/2100=\frac{65}{42}\approx 1.55$

4.9.5

If there is only ALU-ALU forwarding, we need to add two `nop` between the second instruction and the third instruction.

```
or r1,r2,r3
or r2,r1,r4
nop
nop
or r1,r1,r2
```

4.9.6

total execution time with only ALU-ALU forwarding: $(5+4)*290 = 2610ps$

speedup: $2610/2100 = \frac{87}{70} \approx 1.24$