CA homework#4 handwritten

b07902076
資工三 許世儒

4.7.1

$$\begin{aligned}\text{Latency} &= \text{Register Read} + \text{I-Mem} + \text{Register File} + \text{Mux} + \text{ALU} + \text{Mux} \\ &\quad + \text{Register Setup} \\ &= 30 + 250 + 150 + 25 + 200 + 25 + 20 = 700(\text{ps})\end{aligned}$$

4.7.2

$$\begin{aligned}\text{Latency} &= \text{Register Read} + \text{I-Mem} + \text{Register File} + \text{Mux} + \text{ALU} \\ &\quad + \text{Data Memory} + \text{Mux} + \text{Register Setup} \\ &= 30 + 250 + 150 + 25 + 200 + 250 + 25 + 20 = 950(\text{ps})\end{aligned}$$

4.7.3

$$\begin{aligned}\text{Latency} &= \text{Register Read} + \text{I-Mem} + \text{Register File} + \text{ALU} \\ &\quad + \text{Data Memory} + \text{Mux} \\ &= 30 + 250 + 150 + 200 + 250 + 25 = 905(\text{ps})\end{aligned}$$

4.7.4

$$\begin{aligned}\text{Latency} &= \text{Register Read} + \text{I-Mem} + \text{Register File} + \text{Mux} + \text{ALU} \\ &\quad + \text{AND Gate} + \text{Mux} + \text{Register Setup} \\ &= 30 + 250 + 150 + 25 + 200 + 5 + 25 + 20 = 705(\text{ps})\end{aligned}$$

4.7.5

$$\begin{aligned}\text{Latency} &= \text{Register Read} + \text{I-Mem} + \text{Register File} + \text{Mux} + \text{ALU} \\ &\quad + \text{Mux} + \text{Register Setup} \\ &= 30 + 250 + 150 + 25 + 200 + 25 + 20 = 700(\text{ps})\end{aligned}$$

4.7.6

Minimum clock period should be the maximum among the above five types of instructions. Hence, that is **950(ps)**.

4.27.1

add x15, x12, x11
nop
nop
ld x13, 4(x15)
ld x12, 0(x2)
nop
or x13, x15, x13
nop
nop
sd x13, 0(x15)

4.27.2

We cannot reduce the number of NOPs no matter how we rearrange the code.

4.27.3

It still works with forwarding on the original code if there is no hazard detection.

4.27.4

Hazard detection unit:
PCWrite and IF/IDWrite are set to be 1
and the Mux is set to make the value of Control pass through.

Forwarding unit:

Cycle	Forward A	Forward B
1	X	X
2	X	X
3	00	00
4	10	00
5	01	01
6	00	10
7	00	10

4.27.5

When the instruction in the ID stage needs the result of the instruction in the EX or MEM stage, then the instruction should be stalled. Also, the Hazard detection unit has the value of rd coming out from the EX/MEM register as inputs. Hence, we need another value from the MEM/WB register as input. As for the output, we don't need any additional outputs.
In this instruction sequence, when the rd of add instruction is in the EX/MEM register, the data hazard should be detected. It's because the instruction ld relies on the value of the rd from the instruction add. Also, when the rd of the first ld instruction is in the MEM/WB register, the data hazard should be detected because of the or instruction.

4.27.6

Cycle	PCWrite	IF/IDWrite	Mux Control
1	1	1	0
2	1	1	0
3	1	1	0
4	0	0	1
5	0	0	1

4.28.1

incorrect is $(1 - 45\%)$, branch instructions is 25% and there are three instructions in IF, ID and EX stages should be flushed. Thus, $(1 - 45\%) \times 25\% \times 3 = 0.4125$ $\Rightarrow 1 + 0.4125 = 1.4125$

4.28.2

$1 + 25\% \times (1 - 55\%) \times 3 = 1.3375$

4.28.3

$1 + 25\% \times (1 - 85\%) \times 3 = 1.1125$

4.28.4

Converting half of branch instructions to some ALU instructions, the percentage changes from 25% to 12.5%.
Therefore, the CPI changes to $1 + 12.5\% \times (1 - 85\%) \times 3 = 1.05625$.
As a result, the speedup is $\frac{1.1125}{1.05625} \approx 1.0533$

4.28.5

the speedup is $\frac{1.1125}{1.125 \times (1 + \frac{12.5\%}{1+12.5\%} \times (1-0.85) \times 3)} \approx 0.9418$

4.28.6

$0.8 \times 1 + 0.2 \times x = 0.85$ solve for x $\Rightarrow x = 0.25$

4.29.1

always taken: $\frac{3}{5} = 0.6$
always not taken: $\frac{2}{5} = 0.4$

4.29.2

correct: TNTT
prediction: NNNN
accuracy = $\frac{1}{4} = 0.25$

4.29.3

correct: TNTTNTNTTN(TNTTNTNTTN)...
prediction: NNNNTNTNTT(TTTTTTTTT)...
accuracy = $\frac{3}{5} = 0.6$

4.29.4

It's a shift register with the number of bits in the target pattern. The leftmost bit is the output of the predictor. It will left shift for 1-bit after the predicted branch. Besides, it should be initialized with its pattern.

4.29.5

The accuracy is zero because it's opposite to the actual outcome.

4.29.6

It's similar to 4.29.4. The only difference is that if the prediction differs to the actual outcome, it should invert all the bits in the register. However, for the opposite pattern, there is a one cycle warm-up period.