b05202068 劉耿宇

4.7

① R-type: Register Read + I-Mem + Reg. File + Mux + ALU + Mux + Reg. setup
30 ps + 250 ps + 150 ps + 25 ps + 200 ps + 25 ps + 20 ps
= 700 ps ※

② ld: 30 + 250 + 150 + 25 + 200 + 250 + 25 + 20 = 950 ps ※※
Reg. Read, I-Mem, Reg. file, Mux, ALU, D-mem, Mux, Reg. setup

③ sd: 30 + 250 + 150 + 200 + 250 + 25 = 905 ps
ALU, D-mem, Mux

④ beq: 30 + 250 + 150 + 25 + 200 + 5 + 25 + 20 = 705 ps ※
AND gate, Reg. setup

⑤ I-type: 30 + 250 + 150 + 25 + 200 + 25 + 20 = 700 ps ※
no D-mem

⑥ longest latency as minimum clock period → 950 ps.

2017 6
2018 6
2019 6
2020 6

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)
```

② Cannot reduce # of NOPs. by rearranging.

③ Hazard detection → instruction following load uses the result of load.
The seq. of instruction does not have this problem
→ the code works correctly.

④

Cycle	1	2	3	4	5	6	7	8
add	IF	ID	EX	MEM	WB			
ld		IF	ID	EX	MEM	WB		
ld			IF	ID	EX	MEM	WB	
or				IF	ID	EX	MEM	WB
sd					IF	ID	EX	MEM

no stall (no instructions following load use the result of load) → PCWrite, IF/IDWrite on

mux before ID/EX is set to let control values pass through

Cycle:	Forward A	Forward B	
1	X	X	(only one instruction in IF)
2	X	X	(no instruction in EX)
3.	0	0	(no forwarding → values take from reg.)
4.	$10_2 = 2_{10}$	0	(EX/MEM Register Rd = ID/EX Rs1. ↳ base addr.
5	1	1	(MEM/WB Register Rd = ID/Ex Rs1,2. base addr.
6	0 (Rs1 = x15 taken from reg.)	1	(MEM/WB Register Rd = ID/Ex Rs=). ↓ Rs2 = x13 taken from 1st ld
7	0	2	base addr. taken from reg. file. data to be written (x13) is from previous instruction

⑤ If no forwarding, the hazard detection unit needs the values of rd from MEM/WB

∴ The instruction currently in the ID stage have to be stalled if it depends on values forwarded from the instruction in EX or MEM stage.

→ check the destination reg. of these two instructions.

No further output needed. Can stall the pipeline using three existing output signals.

- the value of rd from EX/MEM → detect data hazard btw add & ld
- the value of rd from MEM/WB → " btw 1st ld & or

⑥.

Cycle	1	2	3	4	5	6.
add	IF	ID	EX	ME	WB	
ld		IF	ID	-	-	EX
ld			IF	-	-	ID

Cycle:	PC write	IF/ID write	mux
1	1	1	0
2	1	1	0
3	1	1	0
4	0	0	1
5	0	0	1

→8.

① ∵ incorrect predicted branch → 3 instructions (in IF, ID, EX stages) flushed

Not taken ⇒ CPI increases to $1 + 0.55 \times 3 \times 0.25$ (beq) $= 1.4125$ ✱✱

(at MEM; the branch instruction → go update PC with the next instruction)

② $1 + 0.25 \times 3 \times 0.45 = 1.3375$

(3 = # of stall cycles; 0.45 = mispredict by always-not-taken)

③ $1 + 0.25 \times 3 \times 0.15 = 1.1125$ ✱✱

④

∵ half of branch instruction → ALU instruction

25% → 12.5%

∵ the predicted & mispredicted branch are replaced equally

⇒ new CPU : CPI ⇒ $1 + 0.125(1 - 0.85) = 1.01875$

$CPI|_{\text{no replace}} \Rightarrow 1 + 0.25(1 - 0.85) = 1.0375$.

⇒ $\frac{1.0375}{1.01875} = 1.0184$ speedup ✱✱

⑤ Two ADD instructions as a branch

- 0.5 replaced → 1 extra cycle
- 0.5 not replaced
  - 0.15 (mispredicted) → 1 cycle (pipeline flush)
  - 0.85 (predicted)

∴ $CPI|_{new} = 1 + 0.5 \times 0.25 \times 1 + 0.5 \times 0.25 \times 0.15 = 1.14375$

speedup $= \frac{1.0375}{1.14375} = 0.91$.

⑥ $0.8 \times 1 + 0.2 \times x = 0.85$ (overall accuracy) ⇒ $x = 0.25$ (accuracy of 2-bit predictor on the 20% of branch instructions) ✱

29.

Taken $\frac{3}{5} \Rightarrow 60\%$ Not taken $\frac{2}{5} \Rightarrow 40\%$

② Start from bottom left state

T, NT, T, T

↓ predict outcome

NT NT NT NT. = only hit on the second one → 25% accuracy. ※

(0 1 0 0) ← predictor value

10 D

③ repeating pattern : T NT T T NT

predictor outcome : T T T T T. (assume starting from upper left).

(∵ the decision can only change when there are two successive wrong predictions)

∴ accuracy $= \frac{3}{5} \times 100 = 60\%$

if repeating forever, the prediction outcome would be

T NT T T NT T NT T T NT - - -

T T T T T T T T T T - - -

In the first 5k predicts, there are 3k correct predicts

→ $\frac{3k}{5k} \Rightarrow 60\%$ ※

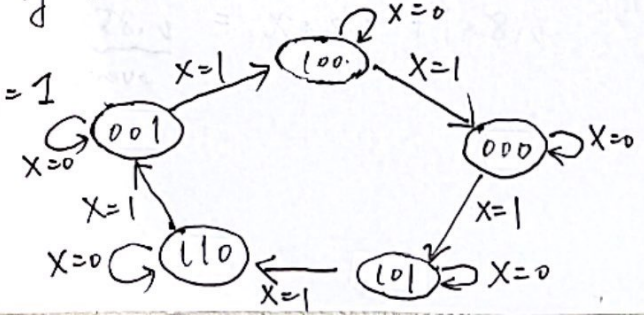
④ For T NT T T NT, a sequential circuit (predictor) to get perfect accuracy (i.e., it outputs T NT T T NT repeatedly). takes.

- A counter circuit with 3 D-flip-flops ∴ → 5 states are generated.
- A control signal : X (input) → specify branch instruction.
- A clock signal : C

the state transition occurs only when X = 1

T NT T T NT

100 000 101 110 001.



Hence, in general, the predictor should be an N-bit shift register
(N = the # of branch outcomes in the target pattern)
the shift register is initialized with the pattern itself ($\begin{matrix}0: NT\\1: T\end{matrix}$).
→ the prediction is the leftmost bit of the shift register.
* The register should be shifted after each predicted branch.

⑤
∴ the perfect predictor's output is now always the opposite of the actual outcome of the branch instruction → accuracy = 0 *

⑥ The predictor is similar to that in ④, except that it should compare the prediction with the actual outcome, if will invert (do NOT) all bits in the shift register if the prediction is incorrect.

no warm-up.

- if the given pattern is not inverted → the predictor perfectly predicts ∧ outcomes.
- if " is inverted → the 1st prediction will be ∧ missed, then the remaining are correct. #

warm-up period (one branch)