計算機結構 Exercise 03

B03902062 資工三 董文捷

4.4.1 critical path = Instruction memory
cycle time = $200ps$

4.4.2 critical path = Instruction memory + Sign extend + Shift left + Add + Mux
cycle time = $200ps + 15ps + 10ps + 70ps + 20ps = 315ps$

4.4.3 time to calculate PCsrc = Regs + Mux + ALU = $90ps + 20ps + 90ps = 200ps$
time to calculate branch target = Sign extend + Shift left + Add = $15ps + 10ps + 70ps = 95ps$
critical path = Instruction memory + Regs + Mux + ALU + Mux
cycle time = $200ps + 200ps + 20ps = 420ps$

4.4.4 Branch instructions and jumps require Shift-left-2.

4.4.5 For branch instructions, it takes more time to decide whether to branch or not than calculating the branch target, so Shift-left-2 is not on the critical path. Shift-left-2 is on the critical path of jumps.

4.4.6 For `beq` instructions, since Shift-left-2 is not on the critical path, latency is not affected. `add` instructions take Instruction memory + Regs + Mux + ALU + Mux = $420ps$, which is not affected either. The only case that latency of Shift-left-2 will affect the cycle time is Shift-left-2 $> 115ps$, then it is on the critical path of `beq`, and `beq` will take more time than `add`, making the cycle time longer. Since Shift-left-2 is usually quite effective, this case is almost impossible.

4.8.1 clock cycle time in a non-pipelined processor
= $250ps + 350ps + 150ps + 300ps + 200ps = 1250ps$
clock cycle time in a pipelined processor
= maximum latency for an individual stage = $350ps$

4.8.2 total latency in a non-pipelined processor
= IF + ID + EX + MEM + WB = $1250ps$
total latency in a pipelined processor
= total latency in a non-pipelined processor
= $1250ps$ (Latency for each instruction not reduced)

4.8.3 Since clock cycle in a pipelined processor is decided by the maximum latency for an individual stage, we should split ID into two stages, each with latency of $175ps$. The new clock cycle time is then decided by MEM, which is $300ps$.

4.8.4 utilization of the data memory = lw + sw = 35%

4.8.5 utilization of the write-register port = alu + lw = 65%

4.8.6 clock cycle time
single-cycle : $1250ps$
multi-cycle : maximum latency for an individual stage = $350ps$
pipelined : $350ps$
$\longrightarrow$ single-cycle > multi-cycle = pipelined

execution time
single-cycle :
$1250ps \times 100\% = 1250ps$
multi-cycle :
alu = IF + ID + EX + WB = 4 cycles = $1400ps$
beq = IF + ID + EX = 3 cycles = $1050ps$
lw = IF + ID + EX + MEM + WB = 5 cycles = $1750ps$
sw = IF + ID + EX + MEM = 4 cycles = $1400ps$
$1400ps \times 45\% + 1050ps \times 20\% + 1750ps \times 20\% + 1400ps \times 15\% = 1400ps$
pipelined :
$350ps \times 100\% = 350ps$
$\longrightarrow$ multi-cycle > single-cycle > pipelined

4.9.1 According to Data dependency, there are three types of data dependences, anti-dependence, flow dependence, and output dependence.
Between `or r1, r2, r3` and `or r2, r1, r4` : anti-dependence + flow dependence
Between `or r2, r1, r4` and `or r1, r1, r2` : anti-dependence + flow dependence
Between `or r1, r2, r3` and `or r1, r1, r2` : output dependence

4.9.2 If there is no forwarding
Data hazard between `or r1, r2, r3` and `or r2, r1, r4` :
To wait result of `r1`, ID of second instruction can not be executed earlier than WB of first instruction $\rightarrow$ 2 bubbles
Data hazard between `or r2, r1, r4` and `or r1, r1, r2` :
To wait result of `r2` $\rightarrow$ 2 bubbles

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

4.9.3 All hazards are eliminated by forwarding the result of EX stage to next instruction.

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

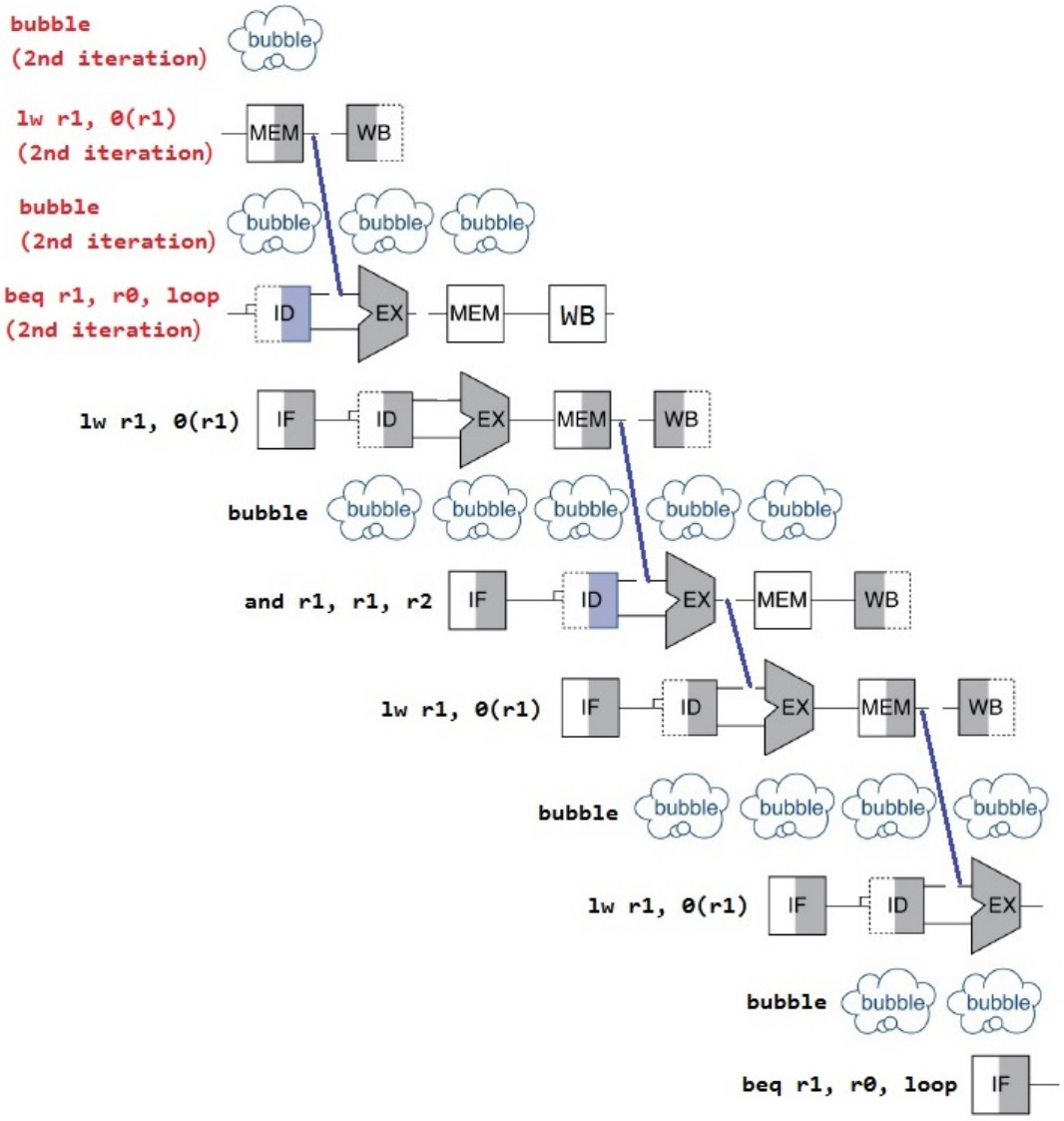
4.9.4 Without Forwarding : $250ps \times (5 + 2 + 1 + 2 + 1) = 2750ps$
With Full Forwarding : $300ps \times (5 + 1 + 1) = 2100ps$
Speedup $= \dfrac{2750ps}{2100ps} = 1.31$

4.9.5 To eliminate hazards, forwarding from the MEM to the EX stage is not needed because all instructions are ALU instructions.

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

4.9.6 Without Forwarding : $250ps \times (5 + 2 + 1 + 2 + 1) = 2750ps$
With ALU-ALU Forwarding Only: $290ps \times (5 + 1 + 1) = 2030ps$
Speedup $= \dfrac{2750ps}{2030ps} = 1.35$

4.11.1



4.11.2 According to the above figure, it takes 8 cycles for each iteration of the loop, but the situation that all five pipeline stages are doing useful work never happens because there are always some bubbles in one or two stages. $\frac{0}{8} = 0\%$.

4.13.1
```
add r5, r2, r1
nop
nop
lw r3, 4(r5)
lw r2, 0(r2)
nop
or r3, r5, r3
nop
nop
sw r3, 0(r5)
```

4.13.2 The only instruction that can be rearranged is `lw r2, 0(r2)`, but it does not cause any hazard. Orders of other instructions can not be changed because of their flow dependences. Therefore, no hazard can be avoided.

```
add r5, r2, r1
nop
nop
lw r3, 4(r5)
lw r2, 0(r2)
nop
or r3, r5, r3
nop
nop
sw r3, 0(r5)
```

4.13.3 If there is forwarding, all hazards are eliminated. Even without hazard detection unit, this code can still execute correctly.

4.13.4 Since there are no bubbles, `PCWrite` and `IF/IDWrite` is always 1, and `ID/EXZero` is always 0. For EX hazard, `ForwardA` or `ForwardB` will be `10`. For MEM hazard, `ForwardA` or `ForwardB` will be `01`. `ForwardA` and `ForwardB` of the first two cycles are unknown because we do not know whether forwarding from previous instructions is needed.

PCWrite = 1, IF/IDWrite = 1, ID/EXZero = 0, ForwardA and ForwardB are unknown.
PCWrite = 1, IF/IDWrite = 1, ID/EXZero = 0, ForwardA and ForwardB are unknown.
PCWrite = 1, IF/IDWrite = 1, ID/EXZero = 0, ForwardA = 00, ForwardB = 00.
PCWrite = 1, IF/IDWrite = 1, ID/EXZero = 0, ForwardA = 01, ForwardB = 00.
(`lw r3, 4(r5)` forwarding `r5` from `add r5, r2, r1`)
PCWrite = 1, IF/IDWrite = 1, ID/EXZero = 0, ForwardA = 00, ForwardB = 00.

4.13.5 If there is no forwarding, we have to check whether EX/MEM.RegisterRd or MEM/WB.RegisterRd is the same as ID/EX.RegisterRs or ID/EX.RegisterRt in Hazard detection unit, just like Forwarding unit does. Every forwarding requirement results in a hazard. PCWrite, IF/IDWrite, and ID/EXZero of Hazard detection unit are already able to handle bubbles, no extra output signals are needed.

4.13.6 PCWrite = 1, IF/IDWrite = 1, ID/EXZero = 0
PCWrite = 1, IF/IDWrite = 1, ID/EXZero = 0
PCWrite = 1, IF/IDWrite = 1, ID/EXZero = 0
PCWrite = 0, IF/IDWrite = 0, ID/EXZero = 1
PCWrite = 0, IF/IDWrite = 0, ID/EXZero = 1
(2 bubbles → r5 is still not ready)

4.16.1 accuracy of always-taken predictors $= \frac{3}{5} = 60\%$
accuracy of always-not-taken predictors $= \frac{2}{5} = 40\%$

4.16.2 (1) Predict NT, state transits to the bottom right state.
(2) Predict NT, state transits to the bottom left state.
(3) Predict NT, state transits to the bottom right state.
(4) Predict NT, state transits to the top right state.
accuracy $= \frac{1}{4} = 25\%$

4.16.3 If this pattern is repeated forever, the predictor will start off in the top right state in every loop, and then

(1) Predict T, state transits to the top left state.

(2) Predict T, state transits to the top right state.

(3) Predict T, state transits to the top left state.

(4) Predict T, state remains unchanged.

(5) Predict T, state goes back to the top right state.

$$\text{accuracy} = \frac{3}{5} = 60\%$$