答案

1. 由于从 DM 向 ALU 反馈，使得从 EX/MEM 寄存器到其自身之间存在了组合逻辑，即

DM+ALU，其延迟为 2L。流水线频率变为 1/2L，因此图 b 流水线性能是图 a 的 50%。

2.

1) CPI = 1.2 + 0.1(N— 5) = 1.2 + 0.1N — 0.5 = 0.1N + 0.7

800

2) 𝑇c = N + 50

3) 一条指令执行时间 = CPI × 𝑇 = (0.1𝑁 + 0.7) × (800 + 50)

c

N

= 80 + 560 + 5𝑁 + 35

𝑁

= 115 + 560 + 5𝑁

𝑁

* 1. N=10，性能最好。
  2. N=6，性能改善最为显著。



5 6 7 8 9 10 11 12 13 14 15

-2%

205

-1%

-1%

210

-1% -1%

215 0%

0%

225

223

1%

222 221 221 0%222

225

220

1%

227

指令延迟(ps)

性能改善(%)

1%

230

225

2%

2%

230

3%

238

235

3%

240

4%

245

5%

252

5%

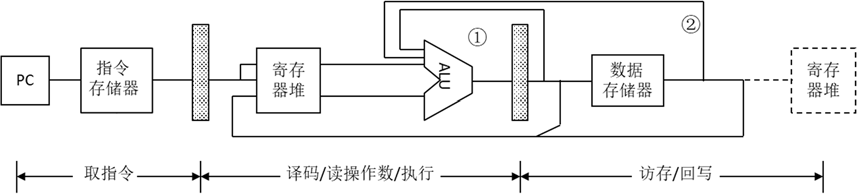
250

6%

255

3.

1. 会。beq 的比较判断电路只能前移至译码/读操作数/执行阶段。这与本章介绍的处理是完全一致的。可能被清除的指令仍然是 beq 后面的那条指令，因此只有 1 条。
2. 从 ALU 和 DM 分别向 ALU 的 A 端转发数据。注意，由于 RF 无内部转发，因此必须有路径 2。



1. 可以。因为对于{lw，add}来说，rs 只在 ALU 的 A 端使用。无论当前哪条指令需要使用 ALU，其前面只可能有 1 条指令位于最后一级。而无论是哪条指令位于最后一级，结果均已产生，因此必然可以通过旁路转发来消除数据冒险。

4.

1. 在 3 级流水线中，中间那段的延迟已经是最坏的 2L 了。
2. 转发电路的组合逻辑为 DM 读出+ALU，同样也是 2L。
3. 这表明转发并没有增加最坏延迟，因此不会导致性能下降。

5.

1. 指令 1 和指令 2 之间在$1 有数据冒险，在转发电路存在的前提下，仍然必须暂停

1 个时钟周期。指令序列调整为：

|  |  |  |
| --- | --- | --- |
| 1 | lw $1, 0($2) |  |
| 2 | nop |  |
| 3 | addi $1, $1, | $1 |
| 4 | sw $1, 0($2) |  |
| 5 | lw $1, 4($2) |  |
| 6 | sw $1, 8($2) |  |

原指令 4（现指令 5）与原指令 5（现指令 6）之间存在数据冒险。但原设计没有转发电路，因此必须将 sw 冻结在 IF/ID，直至 lw 进入 MEM/WB。此时通过 RF 内部转发，sw 就能得到正确的$1。暂停周期数为 2。

故上述代码总的暂停时间为 3 个时钟周期。

1. 前问的 2 个暂停周期是因为解决 lw-sw 之间的数据冒险而产生的，因此需要从

MEM/WB 增加一个转发至 DM 的 WD。



0

1

DM

WD

EX/MEM MEM/WB

6.

1. load-R：没有数据数据相关时，load 的 CPI 为 1。如果有数据相关，需暂停 1 个时钟周期，其 CPI 为 2。

CPIload–R = 1 × (1 — 30%) + 2 × 30% = 1.3

1. load-store：没有数据数据相关时，load 的 CPI 为 1。如果有数据相关，需暂停 2 个时钟周期，其 CPI 为 3。

CPIload–store = 1 × (1 — 5%) + 3 × 5% = 1.1

1. store：题目中未出现数据相关，因此 CPIstore 为 1。
2. 分支：预测成功，分支的 CPI 为 1。如果预测失败，需暂停 1 个时钟周期，分支的

CPI 为 2。

CPI分支 = 1 × 75% + 2 × (1 — 75%) = 1.25

1. R 型：CPI 为 1。

CPI = CPIload–R × 30% + CPIload–store × 5% + CPIstore × 10% + CPI分支 × 10% + CPIR 型

× 65%

= 1.3 × 30% + 1.1 × 5% + 1 × 10% + 1.25 × 10% + 1 × 65%

= 1.32

7.

1) lw-sw 在$1 有 2 次相关；lw-sub 在$1 相关；add-sub 在$3 相关。

2)

|  |  |  |  |
| --- | --- | --- | --- |
| I1: | lw | $1, | 0($2) |
| I3: | add | $3, | $2, $2 |
| I5: | or | $5, | $5, $6 |
| I2: | sw | $1, | 0($1) |
| I4: | sub | $4, | $1, $3 |

3) 优化前：13 个 cycle

由于仅在 W 和 D 之间存在转发，因此 lw-sw 以及 add~sub 之间必须分别插入 2 个

NOP。指令总数从5 条变为9 条，因此流水线共计需要9+（5-1）=13 个时钟周期。优化后：9 个 cycle

优化后，lw~sw 和 add~sub 的数据相关均通过转发解决了，因此无需插入 NOP， 故执行时间=5+（5-1）=9 个时钟周期