无04 2019012137 张鸿琳

5.

引起流水线冒险的原因是在后的指令所使用的变量由在前的指令生成。

MIPS流水线处理器可能的冒险类型有：

- 结构冒险：流水执行的多条指令在同一个周期使用同一个硬件资源。
- 数据冒险：由于不同指令的操作数具有相互依赖关系而造成的流水线冒险。
- 控制冒险：由于 PC 的值依赖指令的执行结果而产生的冒险。

6.

- 第三条指令add和第一条指令addi有数据冒险，因为用到的$t0的值还未写入
- 第三条指令add和第二条指令addi有数据冒险，因为用到的$t1的值还未写入
- 第四条指令add和第二条指令addi有数据冒险，因为用到的$t1的值还未写入
- 第四条指令add和第三条指令add有数据冒险，因为用到的$t2的值还未写入
- 第五条指令add和第三条指令add有数据冒险，因为用到的$t2的值还未写入
- 第五条指令add和第四条指令add有数据冒险，因为用到的$t0的值还未写入

7. a)

第5个时钟周期末，$6、$1寄存器正在被读取，$4寄存器将被写入。

7. b)

在第5个时钟周期，转发单元的作用是检测是否发生了数据冒险，以及转发操作在哪两条指令之间进行，是从第三条指令转发到第四条指令，还是从第二条指令转发到第四条指令，并据此发出控制信号，使流水线进行转发操作。具体的比较操作如下：

- 检测EX/MEM.RegWrite是否为1，即判断第三条指令（前一条）是否需要写入寄存器，若不需要则不会导致数据冒险
- 检测EX/MEM.RegWrAddr是否为0，如果第三条指令（前一条）结果更新到零寄存器，那么也不会导致数据冒险
- 检测EX/MEM.RegWrAddr 与 ID/EX.RegisterRs是否相等，也就是判断第三条指令（前一条）写入的地方是不是即将执行的指令（第四条指令）运算所需的数据
- 检测EX/MEM.RegWrAddr 与 ID/EX.RegisterRt是否相等，也就是判断第三条指令（前一条）写入的地方是不是即将执行的指令（第四条指令）运算所需的数据
- 检测MEM/WB.RegWrite是否为1，也就是判断第二条指令（前前条）是否需要将结果写回寄存器，若不需要则不会导致数据冒险
- 检测MEM/WB.RegWrAddr是否为0，如果第二条指令（前前条）结果更新到零寄存器，那么也不会导致数据冒险
- 检测MEM/WB.RegWrAddr 与 ID/EX.RegisterRs是否相等，也就是判断第二条指令（前前条）写入的地方是不是即将执行的指令（第四条指令）运算所需的数据
- 检测EX/MEM.RegWrAddr 与 ID/EX.RegisterRs 是否相等以及EX/MEM.RegWrite是否为0，也就是保证从第二条指令（前前条）转发数据时，不会从第三条指令（前一条）转发数据

7. c)

在第5个时钟周期，冒险检测单元的作用是检测是否发生了load-use冒险，并据此发出控制信号，使流水线发生阻塞。具体的比较操作如下：（此处不考虑前lw后sw且sw地址不用到lw存入寄存器的值这一不用阻塞的特殊情况）

- 检测ID/EX.MemRead是否为1，即判断第三条指令是否为lw，若不是则不会发生load-use冒险
- 检测ID/EX.RegisterRt与IF/ID.RegisterRs是否相等，也就是假如第三条指令为lw，那么判定第四条指令是否用到了第三条指令的输出数据
- 判断ID/EX.RegisterRt与IF/ID.RegisterRt是否相等，也就是假如第三条指令为lw，那么判定第四条指令是否用到了第三条指令的输出数据

10. a)

前lw后add时，会发生一次阻塞，而前add后lw时，通过转发，可以不用阻塞。故而实际CPI（由于指令数较多，可以认为形成了稳定的流水线）为$\frac{3}{2} = 1.5$。（lw，stall，add）

10. b)

没有转发时，前lw后add，需要阻塞两个周期，前add后lw，也需要阻塞两个周期。故而实际CPI（由于指令数较多，可以认为形成了稳定的流水线）为$\frac{6}{2} = 3$。（lw，stall，stall，add，stall，stall）

14. a)

数据转发通路不需要修改。

14. b)

冒险检测单元需要修改。load-use冒险发生时需要阻塞两个周期。同时前lw后sw时，需要阻塞一个周期再转发（如果计算sw的地址时用到了lw存入寄存器的值，那么也需要阻塞两个周期），这是一个新的冒险。

14. c)

可以阻塞足够长周期数来保证流水线正常工作，也就是阻塞到lw、sw这样的指令完全结束。