山东大学 计算机科学与技术 学院

计算机体系结构 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号： | 姓名： | | 班级： |
| 实验题目：实验五: 数据相关 | | | |
| 实验学时：4 | | 实验日期： 2025.5.26 | |
| 实验目的：  通过本实验，加深对数据相关的理解，掌握如何使用定向技术来减少数据相关带来的暂停。 | | | |
| 硬件环境：  CPU: Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz 2.81 GHz  RAM: 24.0 GB | | | |
| 软件环境：  Windows 11 专业版  Ovtdm 0.9.0  WinDLX | | | |
| 实验步骤与内容：  (1) 在不采用定向技术的情况下（通过 Configuration 菜单中的Enable Forwarding 选项设置），用 WinDLX 模拟器运行程序 data\_d.s 。  (2) 记录数据相关引起的暂停时钟周期数以及程序执行的总时钟周期数，计算暂停时钟周期数占总执行周期数的百分比。  (3) 在采用定向技术的情况下，用 WinDLX 模拟器再次运行程序 data\_d.s。  (4) 记录数据相关引起的暂停时钟周期数以及程序执行的总时钟周期数，计算暂停时钟周期数占总执行周期数的百分比。  (5) 根据上面记录的数据，计算采用定向技术后性能提高的倍数。  首先关闭定向技术，运行程序：    在关闭定向技术的情况下，程序中的多处数据相关导致流水线频繁停顿：   * ADDUI R2, R2, 0x134 依赖 LHI R2, 0x0：LHI 的结果在WB阶段才写入寄存器，但 ADDUI 在ID阶段需读取 R2，导致必须等待2个周期（MEM→WB），形成2个stall。 * ADD R1, R1, R3 依赖 LW R1, 0(R2)：LW 的加载结果在WB阶段才可用，而 ADD 在ID阶段需读取 R1，同样需等待2个周期，形成2个stall。 * SW 0(R2), R1 依赖 ADD R1, R1, R3：ADD 的结果在WB阶段写入寄存器，而 SW 在ID阶段需读取 R1，导致2个stall。 * ADDI R5, R5, #10 依赖 LW R5, 0(R1)：LW 的结果在WB阶段才写入 R5，而 ADDI 在ID阶段需读取 R5，形成2个stall。 * SUB R4, R3, R2 依赖 ADDI R2, R2, #4：ADDI 的结果在WB阶段写入 R2，而 SUB 在ID阶段需读取 R2，形成2个stall。 * BNEZ R4, loop 依赖 SUB R4, R3, R2：SUB 的结果在WB阶段写入 R4，而 BNEZ 在ID阶段需读取 R4，形成2个stall。     从图中可以看到这些 RAW 类型数据相关在无定向技术时，均需等待前序指令的写回阶段，导致每个依赖引入2个stall，总计104次，占总周期数的51.48%。  启用定向技术，运行程序：    启用定向技术后，硬件通过数据转发路径（如 EX→EX、MEM→EX）直接传递中间结果，无需等待写回阶段：   * ADDUI R2, R2, 0x134 依赖 LHI R2, 0x0：LHI 的执行结果可在EX阶段通过 EX→EX 路径直接传递给 ADDUI，无需等待WB，消除2个stall。 * ADD R1, R1, R3 依赖 LW R1, 0(R2)：LW 的加载结果在MEM阶段通过 MEM→EX 路径直接传递给 ADD，节省2个周期。 * SW 0(R2), R1 依赖 ADD R1, R1, R3：ADD 的执行结果通过 EX→MEM 路径直接传递给 SW，无需等待WB。 * ADDI R5, R5, #10 依赖 LW R5, 0(R1)：LW 的结果通过 MEM→EX 路径传递给 ADDI。 * SUB R4, R3, R2 依赖 ADDI R2, R2, #4：ADDI 的执行结果通过 EX→EX 路径传递给 SUB。 * BNEZ R4, loop 依赖 SUB R4, R3, R2：SUB 的结果通过 EX→ID 路径传递给 BNEZ。   通过这些转发路径，上述数据相关引发的stall被完全或部分消除。最终，RAW stalls从104次降至30次，占比从51.48%降至23.44%。总周期数也从202减少到128，性能提升约1.58倍（202/128）。  代码分析：  loop:  LW R1, 0(R2) # 从R2指向的地址加载一个字到R1（读取数组元素A[i]）  ADD R1, R1, R3 # R1 = R1 + R3（将B加到当前数组元素上）  SW 0(R2), R1 # 将结果写回R2指向的地址（更新A[i]）  LW R5, 0(R1) # 从R1指向的地址加载一个字到R5（可能为链式访问）  ADDI R5, R5, #10 # R5 = R5 + 10（对加载的数据进行简单运算）  ADDI R2, R2, #4 # R2 += 4（移动到下一个数组元素地址）  SUB R4, R3, R2 # R4 = R3 - R2（用于计算循环终止条件）  BNEZ R4, loop # 如果R4不等于0，跳转到loop继续执行 | | | |
| 结论分析与体会：  通过WinDLX模拟器运行data\_d.s程序，验证了数据相关对CPU性能的影响及定向技术的优化效果。实验中，代码中LW→ADD、ADD→SW等典型数据依赖在关闭定向技术时因需等待寄存器写回导致大量停顿（占57.42%），而启用定向技术后，硬件通过转发路径直接传递中间结果（如MEM→EX、EX→EX），显著减少停顿至32.81%，程序总周期数从202降至128，性能提升约1.58倍。这一过程直观体现了数据转发机制在解决RAW数据相关中的关键作用，也表明合理设计指令流水线与硬件资源调度是提高处理器效率的核心手段。 | | | |