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：实验四. 结构相关 | | | |
| 实验学时：4 | | 实验日期： 2025.5.21 | |
| 实验目的：  通过本实验，加深对结构相关的理解，了解结构相关对 CPU 性能的影响。 | | | |
| 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). 用WinDLX 模拟器运行程序 structure\_d.s 。  (2). 通过模拟，找出存在结构相关的指令对以及导致结构相关的部件。  (3). 记录由结构相关引起的暂停时钟周期数，计算暂停时钟周期数占总执  行周期数的百分比。  (4). 论述结构相关对 CPU 性能的影响，讨论解决结构相关的方法。  直接运行，在 trap 0x0 处暂停：    可以在左下角统计的 Stalls 内看到具体的数量为 30 RAW Stalls，总 Cycles 为 139，所以显示比例为 21.58%，具体冲突内容如下：  RAW stalls（30次，占21.58%）  代码中频繁出现的ADDD F0, F0, F4和ADDD F2, F0, F2存在直接的数据依赖关系：第二个ADDD必须等待第一个ADDD的F0结果完成才能执行。这种连续的浮点加法操作导致流水线因数据依赖而阻塞。此外，LD指令（如LD F0, 0(R2)）虽然未直接产生stall，但其加载的F0值被后续ADDD依赖，间接引发RAW stalls，这产生了10次 LD stalls。  Branch/dump stalls（10次，占RAW stalls的33.33%）  循环末尾的BNEZ R5, loop指令与SUB R5, R4, R2存在控制依赖：BNEZ的跳转条件依赖SUB的计算结果。在DLX架构中，分支指令通常需要等待前序计算完成才能确定目标地址，导致流水线清空并等待结果，形成stall。尽管SUB和BNEZ之间已插入ADDI等操作，但分支预测机制未启用时，仍会因条件判断延迟产生stall。  Floating point stalls（10次，占RAW stalls的33.33%）  连续的ADDD指令（如ADDD F0, F0, F4和ADDD F2, F0, F2）争夺同一浮点加法单元（FPU）。FPU需要2个周期（我们在 FPS 中设置的值）完成一次浮点加法，而代码中未留出足够间隙让FPU空闲。这种资源争用导致后序ADDD必须等待前序操作完成，形成浮点单元级的流水线阻塞。  查看指令的详细信息：    可以印证我们的想法。  代码分析：  loop:  LD F0, 0(R2) # 从地址R2处加载双精度浮点数到F0（内存访问）  LD F4, 0(R3) # 从地址R3处加载双精度浮点数到F4（内存访问）  ADDD F0, F0, F4 # F0 = F0 + F4（浮点加法操作）  ADDD F2, F0, F2 # F2 = F0 + F2（浮点加法操作） ← 此处发生结构冒险  # 连续两个ADDD指令争夺浮点加法单元（FPU）  # 当前DLX架构只有一个FPU，导致第二个ADDD必须等待  ADDI R2, R2, #8 # R2 = R2 + 8（更新第一个数组指针）  ADDI R3, R3, #8 # R3 = R3 + 8（更新第二个数组指针）  SUB R5, R4, R2 # R5 = R4 - R2（计算剩余循环次数）  BNEZ R5, loop # 如果R5≠0，跳转到loop继续循环  TRAP #0 # 程序终止（系统调用退出） | | | |
| 结论分析与体会：  通过WinDLX模拟器运行structure\_d.s程序，直观观察到结构相关对CPU性能的影响。实验中，连续的ADDD指令因争夺浮点加法单元导致流水线阻塞，而LD指令虽未直接停顿，但其后依赖的数据冒险间接加剧了性能损耗。尽管代码存在潜在的内存与FPU资源争用，实际模拟中未检测到结构相关停顿，推测是模拟器隐式优化或指令间隔合理缓解了冲突。实验通过统计30.22%的总停顿周期占比，验证了结构相关对执行效率的显著影响，进一步认识到资源复制、指令调度和硬件并行设计是解决此类问题的关键手段。 | | | |