**深 圳 大 学 实 验 报 告**

**课程名称： 计算机系统(3)**

**实验项目名称： 实验4处理器结构实验（1）**

**学院： 计算机与软件学院**

**专业： 计算机与软件学院所有专业**

**指导教师： 李琰**

**报告人： 叶茂林 学号： 2021155015 班级： 腾班**

**实验时间： 2023年11月16日**

**实验报告提交时间： 2023年11月19日**

**教务部制**

|  |
| --- |
| 实验目的与要求： 实验目的 了解MIPS的五级流水线，和在运行过程中的所产生的各种不同的流水线冒险  通过指令顺序调整，或旁路与预测技术来提高流水线效率  更加了解流水线细节和其指令的改善方法  更加熟悉MIPS指令的使用 |
| 方法、步骤： 二、实验步骤及相关说明 观察一段代码并运行，观察其中的流水线冒险，并记录统计信息。  对所给的代码进行指令序列的调整，以期避免数据相关，并记录统计信息。  启动forward功能，以获得性能提升，并且记录统计信息。  （选做：用perf记录x86中的数据相关于指令序列调整后的时间统计、调整指令，以避免连续乘法间的阻塞。） |
| 实验过程及内容： 三、实验内容  1. 分析冒险   首先，我们给出一段C代码，该段代码实现的是两个矩阵相加。  设有4\*4矩阵A和4\*4矩阵B相加，得到4\*4矩阵C：  for(int i = 0; i < 4; i++)  For(int j = 0; j < 4; j++)  C[i][j] = A[i][j] + B[i][j];  根据上述的C代码，我们将其转换成MIPS语言，然后运行，并进行分析。  MIPS代码如下：  .data  a: .word 1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4  b: .word 4,4,4,4,3,3,3,3,2,2,2,2,1,1,1,1  c: .word 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0  len: .word 4  control: .word32 0x10000  data: .word32 0x10008  .text  start:daddi r17,r0,0  daddi r21,r0,a  daddi r22,r0,b  daddi r23,r0,c  ld r16,len(r0)  loop1: slt r8,r17,r16  beq r8,r0,exit1    daddi r19,r0,0  loop2: slt r8,r19,r16  beq r8,r0,exit2    dsll r8,r17,2  dadd r8,r8,r19  dsll r8,r8,3    dadd r9,r8,r21  dadd r10,r8,r22  dadd r11,r8,r23    ld r9,0**(r9**)  ld r10,0(r10)  dadd r12,r9,r10  sd r12,0(r11)  daddi r19,r19,1  j loop2  exit2:daddi r17,r17,1  j loop1  exit1: halt  **实验前请保证winMIPS64配置中“Enable Forwarding”没有选中。将这段代码加载到WinMIPS64中，运行后观察结果（提供Statistic窗口截图）。 从Statistic窗口记录：本程序运行过程中总共产生了多少次RAW的数据相关。接下来，我们对产生数据相关的代码逐个分析，请列出产生数据相关的代码，并在下一步中进行分析和优化。**  没有选中winMIPS64配置中“Enable Forwarding”    本程序运行过程中总共产生了220次RAW的数据相关    经过代码分析，找出来这220次所涉及到的代码，总共框出的代码被执行110次，每次产生2个stalls，共2\*110=220次stalls。     1. 调整指令序列   **在这一部分，我们利用指令调整的方法对数据相关代码进行优化，规避数据相关。**  **通过调整序列来规避这个数据相关，在statics窗口中记录其效果。将此结果与初始的结果进行对比，报告RAW相关的次数减少的数量。**  经过代码分析，我们做出以下调整：  将ld r16,len(r0)移到上方，去掉slt指令的2次时钟等待；  将daddi r19,r0,0移到beq指令的前面，这样可以去掉beq指令的一次时钟等待，beq指令被执行5次，所以1\*5=5stalls，总共去掉5次stall，同时slt指令被执行4次，去掉2\*4=8stalls；  最后将无关的daddi r19,r19,1移到sd指令的前面，这样可以去掉sd指令的一次时钟等待，sd指令被执行16次，所以1\*16=16stalls，总共去掉16次stall。    根据调整指令的分析，理论上我们可以去掉2+5+8+16=31次stall，再次运行程序，这次的RAW相关的次数减少到了189次。    再看看运行结果，矩阵相加的结果还是对的，说明我们调整的代码没有影响到代码的功能。     1. Forwarding功能开启   **接下来，我们要展示Forwarding功能的优化效果。**  **首先，我们要知道如何开启Forwarding功能。法如下：点开*configure*下拉窗口，给*Enable Forwarding*选项左侧点上勾。**    **开启了Forwarding功能之后，我们再运行，查看结果，解释哪些数据相关的问题得到解决，并以截图说明问题解决前后的差异所在。**  开启forwarding。    再次运行程序，这次的RAW相关的次数减少到了36次。    Forwarding前推不需要等待计算结果保存到目的寄存器，使用增加的一条额外数据通路，可以将计算的结果直接传给下一条指令计算的输入。    如下图所示，forward优化前需要等待两个时钟周期，前推优化后可以直接拿到前一条指令的计算结果，无需等待。    但是并非所有的冒险都能通过前推来避免气泡停顿，当前一条指令计算的结果需要加载内存的时候，那么下一条指令需要使用前一个结果时，前一个结果还没计算出来，此时前推还不能解决数据冒险，仍然需要等待一个时钟周期。     1. 结构相关优化   流水线中的结构相关，指的是流水线中多条指令在同一时钟周期内争用同一功能部件现象。即因硬件资源满足不了指令重叠执行的要求而发生的冲突。  在WinMIPS64中，我们可以在除法中观察到这种现象。要消除这种结构相关，我们可以采取调整指令位置的方法进行优化。在这个部分，我们首先给出几条C代码，然后将该代码翻译成MIPS代码（为了观察的方便，我们这里MIPS代码并不是逐一翻译，而是调整代码，使得其他部分数据相关已经优化，而两条除法指令连续出现），运行并查看结果。接着，调整代码序列，重新运行。观察优化效果。  下面是给出的C代码：  a = a / b  c = c / d  e = e + 1  f = f + 1  g = g + 1  h = h + 1  i = i + 1  j = j + 1  根据上述的C代码，我们给出数据相关优化的指令如下：  .data  a: .word 12  b: .word 3  c: .word 15  d: .word 5  e: .word 1  f: .word 2  g: .word 3  h: .word 4  i: .word 5    .text  start:  ld r16,a(r0)  ld r17,b(r0)  ld r18,c(r0)  ld r19,d(r0)  ld r20,e(r0)  ld r21,f(r0)  ld r22,g(r0)  ld r23,h(r0)  ld r24,i(r0)  ddiv r16,r16,**r17**  ddiv r18,r18,r19  daddi r20,r20,1  daddi r21,r21,1  daddi r22,r22,1  daddi r23,r23,1  daddi r24,r24,1  halt  **上面的指令运行，在*Cycle*窗口结果如下（程序运行前请将configure->architecture->division latency改为10）：**    **在*Statistics*窗口的结果如下：**    **通过观察，我们可以发现，两个连续的除法产生了明显的结构相关，第二个除法为了等待上一个除法指令在执行阶段所占用的资源，阻塞了9个周期。**  **显然，这样的连续的除法所导致的结构相关极大的降低了流水线效率，为了消除结构相关，我们需要做的是调整指令序列，将其他无关的指令塞入两条连续的除法指令中。**  **给出指令序列的调整方案并给出流水线工作状态的截图，做出解释。**  将configure->architecture->division latency改为10。    运行程序可以发现，两个连续的除法产生了明显的结构相关，第二个除法为了等待上一个除法指令在执行阶段所占用的资源，阻塞了9个周期。    我们通过调整指令顺序，往两个除法指令之间插入九条无关指令。    运行程序，插入的九条指令消除了两个除法之间的执行阶段的结构冒险。 |

深圳大学学生实验报告用纸

|  |
| --- |
| 实验结论： |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。