|  |  |  |
| --- | --- | --- |
| 《计算机组成与体系结构》实验报告 |  | 天津大学本科生实验报告 |
| 学院 智能与计算学部 年级 2019 级 班级 4 班 组号 14  课程名称 计算机组成与体系结构 成绩  组长 齐呈祥 同组实验者 高树韬 王志鸣 闫璟 刘梦迪 岳珺鹏 | **三．实验原理与步骤**  1. 请举例说明流水线中常见的相关性、产生的影响以及消除影响的办法。  流水线共有3种相关：结构相关、数据相关、控制相关  结构相关是指流水线中多条指令在同一时钟周期内争用一个功能部件，从而发生冲突，造成指令无法继续执行。结构相关可以通过添加功能部件来加以解决，例如哈佛结构中，分别使用两个存储器来作为指令存储器和数据存储器。  数据相关是指在流水线中，如果某一条指令必须等前面的指令的运行结果，才能继续执行，那么指令间就存在数据相关。数据相关可以通过指令定向前推来解决。  控制相关是一种由转移指令引起的相关。当执行转移指令时，依据对转移条件的判断，处理器可能会顺序取下一条指令（即PC = PC + 4）；也可能会转移到新的目标地址取指令（即PC = target\_addr）。由于微处理器每个时钟周期都会顺序取出一条指令，则在转移发生之前，转移指令之后的若干条指令已经被取到流水线中。此时，如果发生转移，则将造成转移指令之后进入流水线的那些指令是不应该被执行的，从而使程序运行发生错误。控制相关可以通过延迟转移或者分支预测来解决。  2. 请概述MiniMIPS32中的异常处理流程  当处理器检测到中断或异常时，处理器首先检查cp0的Status寄存器的EXL位是否被置为1，若被置为1则表明此时已经在中断/异常处理流程中，若不为1，则将其置为1，然后根据发生中断/异常的原因，为cp0的Cause寄存器赋值，随后将EPC寄存器保存为当前异常发生的下一条指令，随后将当前pc寄存器设置为固定异常处理地址（MIPS Realease1）或者中断向量中的地址（MIPS Release2）。当经过处理后执行 ERET 指令返回EPC寄存器记录的寄存器，并将Status寄存器的EXL位置为0。  3. 请简述MiniMIPS32处理器解决各种相关性问题的实现方案。  在 MiniMIPS32 中使用定向前推来解决数据相关，我们分别从执行阶段和访存阶段分别将信号进行信号前推，使用两个两位信号来判断取指所获得的数据是否来自执行阶段或者访存阶段。  在 MiniMIPS32 中使用分支延迟或者分支预测来解决控制相关性：  （1）分支延迟槽是一种通过编译器辅助的流水线优化技术，延迟转移规定转移指令（第i条）的后续一条指令（第i+1条）所在的位置称为“分支延迟槽”。编译器在进行指令调度的时候，选择某条指令位于延迟槽中，使得无论转移成功与否流水线都会执行这条指令，即“延迟指令”。由于延迟指令的选取是由编译器在指令调度阶段完成的，故只需将其正常发射到流水线，正常运行即可。  （2）分支预测是在在流水线取指阶段提前猜测分支指令的分支方向。如果猜测正确，则流水线不会停顿，继续执行；否则，清空流水线（已取出的错误指令），从正确分支目标地址处执行。这种猜测分支方向的技术称为分支预测。此处我们实现的版本是动态分支预测，即基于分支的历史信息来决定本次分支的方向。我们加入了几个存放历史信息的查找表（存放分支指令的类型、跳转情况以及跳转地址）。因为程序中分支指令并不多，我们决定采用直接映射的Cache 存放分支信息，这样可以将 PC 直接作为索引，简化逻辑也确保存入查找表的是分支指令。（因用到Cache 后期会继续调试和完善，先完成了查找表等基础部分代码）  其中，第16号寄存器为我们计算出的值，而第18号寄存器的值则是正确的结果，而我们的测试程序则根据这两个寄存器的值是否相同而选择是否跳转（相同则继续执行，跳转则表示指令测试失败）。  DIV 指令仿真波形图：    同理可以看到DIV指令测试结果也是正确的。  MFC0指令仿真波形图：    **五．实验中遇到的问题和解决办法，并谈一下通过本次实验所获得的收获。**  在本次实验中我们学习到了CPU流水线在执行指令的过程中，由于控制相关，数据相关以及结构相关所引发的冲突以及对应的解决方案，通过数据定向前推，指令延迟执行或者分支预测等方法可以解决数据相关以及控制相关的问题，通过将数据存储器和指令存储器分开访问增大了存储器的带宽，可以解决结构相关问题。  在本次实验中我们使用自己设计的测试用例进行测试，在测试过程中也遇到了很多问题，主要通过波形图方法从出现错误的信号一个阶段一个阶段的前移来检查哪个阶段出了错误，极大地提高了设计CPU时Debug的能力。  在本次实验中我们也实现了一个分支预测机制，不过由于时间原因还没有很好地与我们的SOC结合在一起。    **教师签字：**  **年 月 日** |
| 实验项目名称 流水线处理器的相关性消除与异常处理   1. **实验目的**   在实验二的基础上，采用硬件描述语言设计定向前推消除数据冒险、采用分支延迟槽（或分支预测）消除控制冒险、设计暂停单元消除其它冒险、涉及异常处理单元和CP0 实现精确异常处理异常，并通过Vivado工具进行功能仿真。  所设计的处理器将为后续板级系统验证，包括功能点验证和C程序验证，奠定基础。   1. **实验内容**   实验平台：Linux, Windows均可  实验工具：Vivado 2019.2  编程语言：Verilog HDL 或 SystemVerilog HDL  所需支持的指令(19条)包括：   * 转移指令：BEQ、BNE、BGEZ、BGTZ、BLEZ、BLTZ、BLTZAL、BGEZAL、J、JAL、JR、JALR * 除法指令：DIV、DIVU * CP0指令：MFC0、MTC0 * 异常相关指令：SYSCALL、BREAK、ERET   测试程序的要求：   * 每个小组提供4个以上测试程序（定向前推、延迟槽、暂停和异常处理各一个，且不能与教材上相同） * 每个程序汇编指令不少于20条，尽可能覆盖各种边界条件 * 测试程序的编译方法与实验一相同，编译后会生成两个.coe文件（指令和数据），位与mips\_sc/build路径下，可加载到存储器中完成测试 * 指令存储器不小于8K，数据存储器不小于8KB   对于循环类型的分支结构（通常出现在 BEQ、BNE 等指令），采用缓存直接存储近期出现的  分支跳转地址和分支指令类型，这些信息由执行阶段提供。对于调用类型的分支结构，预测器中实现了一个返回地址栈。此时，JAL 指令继续采用缓存预测跳转地址，因为缓存中存放了跳转类型，那么在对JAL指令进行预测后可以将其对应的 PC+8 后压入栈，再遇到 JR 指令时直接从栈中获取跳转地址。  4. 请简述MiniMIPS32处理器实现精确异常处理的方案。  异常发生时，总会有一条指令被打断，称为exception victim，也可称为发生异常的指令。对精确异常而言，该指令之前进入流水线的所有指令都必须被正常运行完毕，而该指令及之后进入流水线的指令都必须从流水线清除，不影响任何处理器状态。  在实现精确异常时，必须保证异常产生的顺序与指令的处理顺序相同。发生异常时先不处理，而是对其标识并沿着流水线继续向后传递到某一阶段，如果在传递的过程中又发现该指令之前的指令发生了异常，则忽略该指令的异常。因为该指令之前的指令产生的异常将导致自己及其后的指令全部取消了。MiniMIPS将会在访存阶段统一对指令进行异常处理。  **四．仿真与实验结果（注：仿真需要给出波形图截图，截图要清晰，如果波形过长，可以分段截取；对仿真波形图中的关键结果数据进行标识和说明；实验结果以开发板截图形式呈现）**  由于我们的测试用例比较多，因此这里我们只选择几条有代表性的指令进行展示。  JAL 指令仿真波形图：    同理可以看到MFC0指令测试结果也是正确的。  BREAK指令仿真波形图：    同理可以看到BREAK指令测试结果也是正确的。 |