实验 2 流水线及流水线中的冲突

张梓良
2021212484

日期：2024 年 4 月 17 日

1 实验目的

1. 加深对计算机流水线基本概念的理解。
2. 理解 MIPS 结构如何用 5 段流水线来实现，理解各段的功能和基本操作。
3. 加深对数据冲突和资源冲突的理解，理解这两类冲突对 CPU 性能的影响。
4. 进一步理解解决数据冲突的方法，掌握如何应用定向技术来减少数据冲突引起的停顿。

2 实验平台

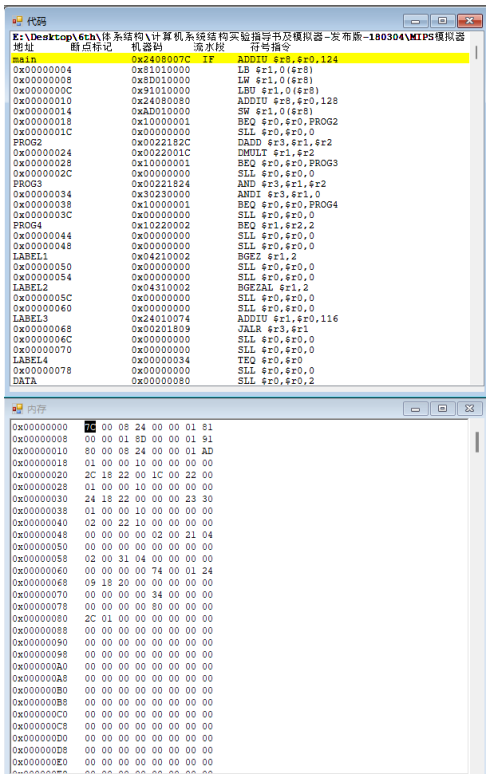
指令级和流水线操作级模拟器 MIPSsim。

3 实验内容和步骤

阅读 MIPSsim 模拟器的使用方法，了解 MIPSsim 的指令系统和汇编语言。

1. 启动 MIPSsim。
2. 进一步理解流水线窗口中各段的功能，掌握各流水寄存器的含义。（鼠标双击各段，即可看到各流水寄存器的内容）
   (a) IF/ID.IR：流水段 IF 与 ID 之间的指令寄存器；
   (b) IF/ID.NPC：流水段 IF 与 ID 之间的下一指令程序计数器；
   (c) ID/EX.A：流水段 ID 与 EX 之间的第一操作数寄存器；
   (d) ID/EX.B：流水段 ID 与 EX 之间的第二操作数寄存器；
   (e) ID/EX.Imm：流水段 ID 与 EX 之间的立即数寄存器；
   (f) ID/EX.IR：存放从 IF/ID.IR 传过来的指令；
   (g) EX/MEM.ALUo：流水段 EX 与 MEM 之间的 ALU 计算结果寄存器；
   (h) EX/MEM.IR：存放从 ID/EX.IR 传过来的指令；
   (i) MEM/WB.LMD：流水段 MEM 与 WB 之间的数据寄存器，用于存放从存储器读出的数据；
   (j) MEM/WB.ALUo：存放从 EX/MEM.ALUo 传过来的计算结果；
   (k) MEM/WB.IR：存放从 EX/MEM.IR 传过来的指令。

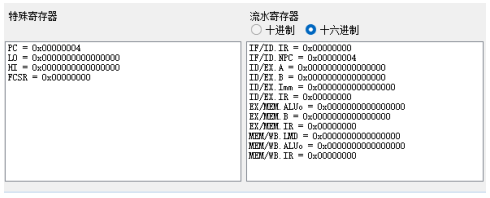
3. 载入样例程序`alltest.s`，然后分别以单步执行一个周期、执行多个周期、连续执行、设置断点等方式运行程序。
4. 选择配置菜单中的“流水方式”选项，使模拟器工作于流水方式下。
5. 观察程序在流水方式下的执行情况。



(a) 单步执行一个周期：

可以观察到正在执行第一条指令，同时内存是以小端存储。

同时寄存器的变化情况如下：

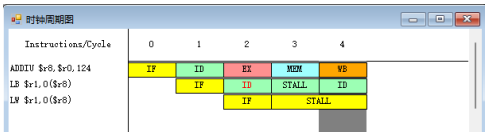


表明当前执行的指令地址为0x0000000000000000，指令内容为0x000000002408007C，下一条指令的地址为0x0000000000000004

(b) 执行 5 个周期:

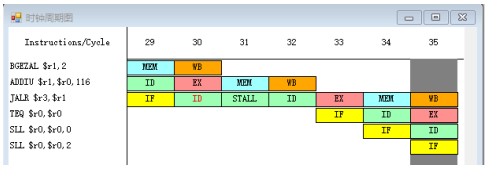
可以观察到现在流水线中共有 3 条指令正在执行，同时第 2、3 条指令的执行过程中均产生了停顿。

同时寄存器的变化情况如下：



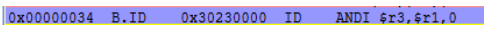
- IF/ID.NPC中存储着下一条指令的地址0x000000000000000C;
- IF/ID.IR中存储着第 3 条指令的内容0x000000008D010000;
- ID/EX.A中存储着第 3 条指令的第一个操作数0x000000000000007C, 为寄存器R8的值;
- EX/MEM.ALUOUT中存储着第 1 条指令的 ALU 输出0x000000000000007C，为寄存器R0的值0+ 立即数124的结果。

(c) 连续执行:

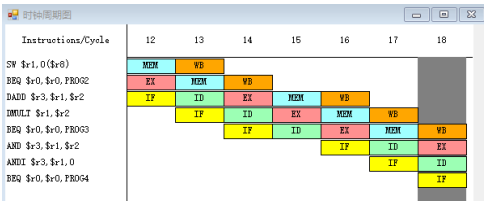


(d) 设置断点:

在下图所示指令的ID段设置断点:



运行结果如下:



6. 观察和分析结构冲突对 CPU 性能的影响，步骤如下:

(a) 加载structure_hz.s。

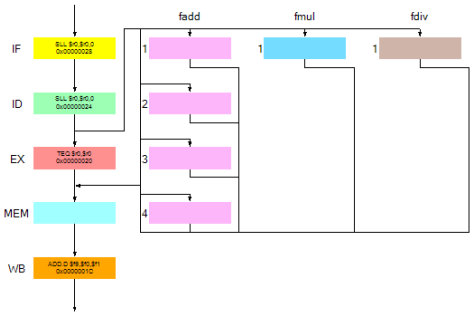
(b) 单步执行该程序直至结束。

可以发现0x00-0x1c的任意一条双精度浮点数加法指令都会引发结构冲突，冲突部件为浮点加法部件fadd。

(c) 记录由结构冲突引起的停顿周期数，计算停顿周期数占总执行周期数的百分比。

结构停顿：35；总时钟周期数：52；占周期总数的百分比：67.30769%。

(d) 把浮点加法器的个数改为 4 个。



(e) 再重复 a-c 的步骤。

结构停顿：2；总时钟周期数：19；占周期总数的百分比：10.52632%。

(f) 分析结构冲突对 CPU 性能的影响，讨论解决结构冲突的方法。

- 影响：发生结构冲突时，流水线会停顿，造成 CPU 性能降低；同时 CPU 中其余部件会空闲，造成资源浪费。
- 解决方法：
  - 资源重复：增加硬件资源的数量，例如使用多个浮点加法部件`fadd`、多个寄存器堆等。
  - 动态调度技术：在运行时动态地调整指令的执行顺序，优化资源的分配。

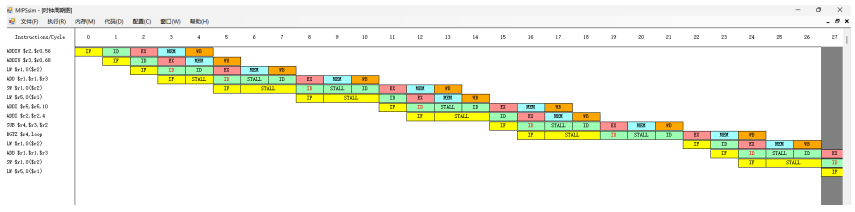
7. 观察数据冲突并用定向技术来减少停顿，步骤如下：

(a) 全部复位。

(b) 加载`data_hz.s`。

(c) 关闭定向功能。

(d) 用单步执行一个周期的方式执行该程序。时钟周期图为：



可以发现在第4,6,7,9,10,13,14,17,18,20,21,25,26,28,29,32,33,36,37,39,40,44,45,47,48,51,52,55,56,58,59周期发生了 RAW 冲突，总共发生了31次 RAW 冲突。

(e) 记录数据冲突引起的停顿周期数以及程序执行的总时钟周期数，计算停顿时钟周期数占总执行周期数的百分比。

RAW 停顿：31；总时钟周期数：65；占周期总数的百分比：47.69231%。

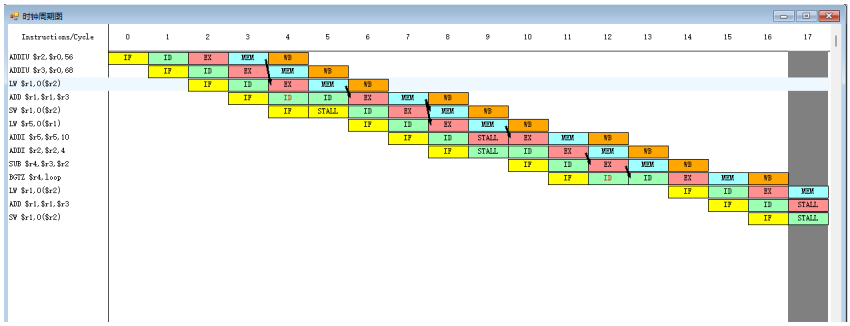
(f) 复位 CPU。

(g) 打开定向功能。

(h) 用单步执行一个周期的方式执行该程序，查看时钟周期图，列出什么时刻发生了 RAW

冲突，并与步骤 c 的结果比较。

时钟周期图为：



可以发现在第5,10,13,18,22,25,30,34,37周期发生了 RAW 冲突，总共发生了9次 RAW 冲突，相较于步骤 c，定向技术大大降低了数据冲突。

(i) 记录数据冲突引起的停顿周期数以及程序执行的总周期数。计算采用定向以后性能比原来提高多少。

RAW 停顿：9；总时钟周期数：43；占周期总数的百分比：20.93023%；性能提升：65/43=1.51。

4 总结

本次实验不仅加深了对 MIPS 架构和流水线技术的理解，而且通过实际操作，深刻体会到了数据冲突和资源冲突对 CPU 性能的影响及其解决方法。通过对比实验前后的数据，可以明显看到通过技术手段优化后 CPU 性能的提升。