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：202200130048	姓名： 陈静雯	班级： 6
实验题目：实验六 指令调度		
实验学时：2	实验日期： 2025.5.30	

实验目的：
通过本实验，加深对指令调度的理解，了解指令调度技术对 CPU 性能改进 的好处。

硬件环境：
Windows

软件环境：
Otvdm

实验程序：

sch-before.s

```
  lf f1,ONE        ; 加载整数 1 到 f1
  cvti2f f7,f1     ; 将整数 1 转为浮点数 1.0 存入 f7
  nop              ; 空操作（等待转换完成）
  divf f1,f8,f7    ; Y = f8 (f1 = f8 / 1.0)
  divf f2,f9,f7    ; Z = f9 (f2 = f9 / 1.0)
  addf f3,f1,f2    ; f3 = Y + Z (需等待 f1,f2 就绪)
  divf f10,f3,f7   ; X = f3 (f10 = f3 / 1.0)
  divf f4,f11,f7   ; B = f11 (f4 = f11 / 1.0)
  divf f5,f12,f7   ; C = f12 (f5 = f12 / 1.0)
  multf f6,f4,f5   ; f6 = B * C (需等待 f4,f5 就绪)
  divf f13,f6,f7   ; A = f6 (f13 = f6 / 1.0)
```

sch-after.s

```
main:
  lf f1,ONE        ; 加载整数 1 到 f1
  cvti2f f7,f1     ; 整数 1 转浮点 1.0 存入 f7
  nop              ; 空操作
  divf f1,f8,f7    ; Y = f8
  divf f2,f9,f7    ; Z = f9
  divf f4,f11,f7   ; B = f11  <- 提前执行（与 Y/Z 无依赖）
  divf f5,f12,f7   ; C = f12  <- 提前执行（与 Y/Z 无依赖）
  addf f3,f1,f2    ; f3 = Y + Z
  multf f6,f4,f5   ; f6 = B * C  <- 与加法并行
  divf f10,f3,f7   ; X = f3
  divf f13,f6,f7   ; A = f6
```

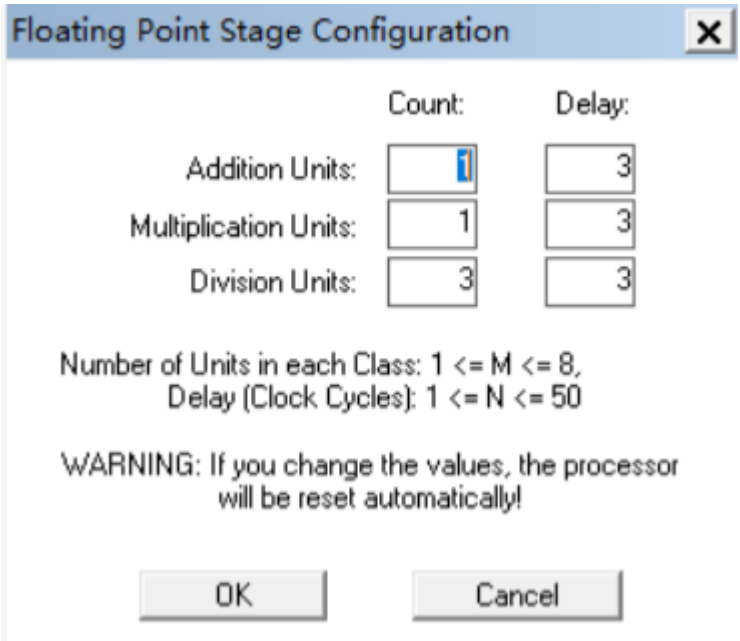
实验步骤：

（1） 通过 Configuration 菜单中的“Floating point stages”选项，把除法单 元数设置

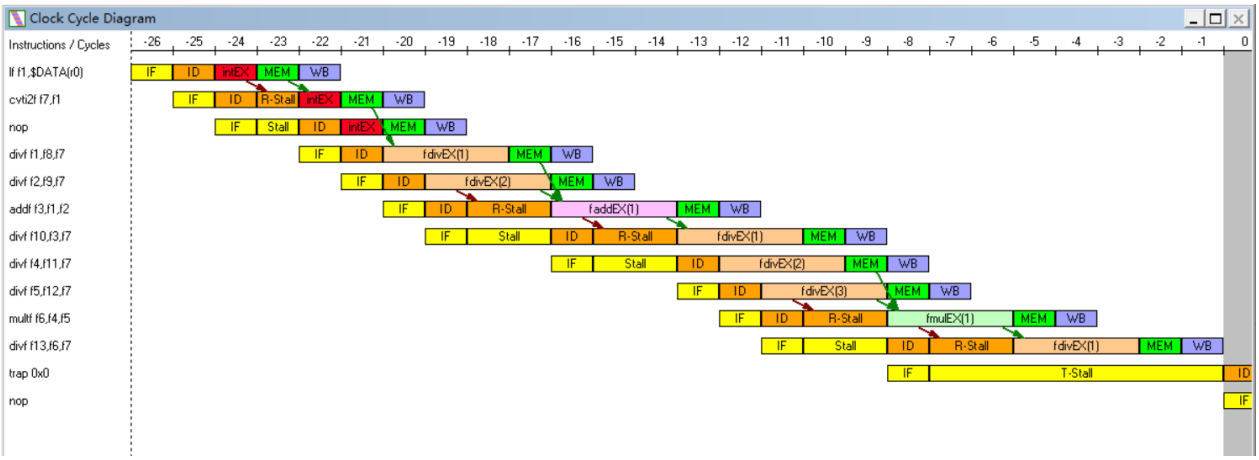
为 3，把加法、乘法、除法的延迟设置为 3 个时钟周期。
（2） 用 WinDLX 模拟器运行调度前的程序 sch-before.s 。记录程序执行 过程中各种相关发生的次数以及程序执行的总时钟周期数。
（3） 用 WinDLX 模拟器运行调度后的程序 sch-after.s ，记录程序执行过 程中各种相关发生的次数以及程序执行的总时钟周期数。
（4） 根据记录结果，比较调度前和调度后的性能。
（5） 论述指令调度对于提高 CPU 性能的意义。

实验内容：

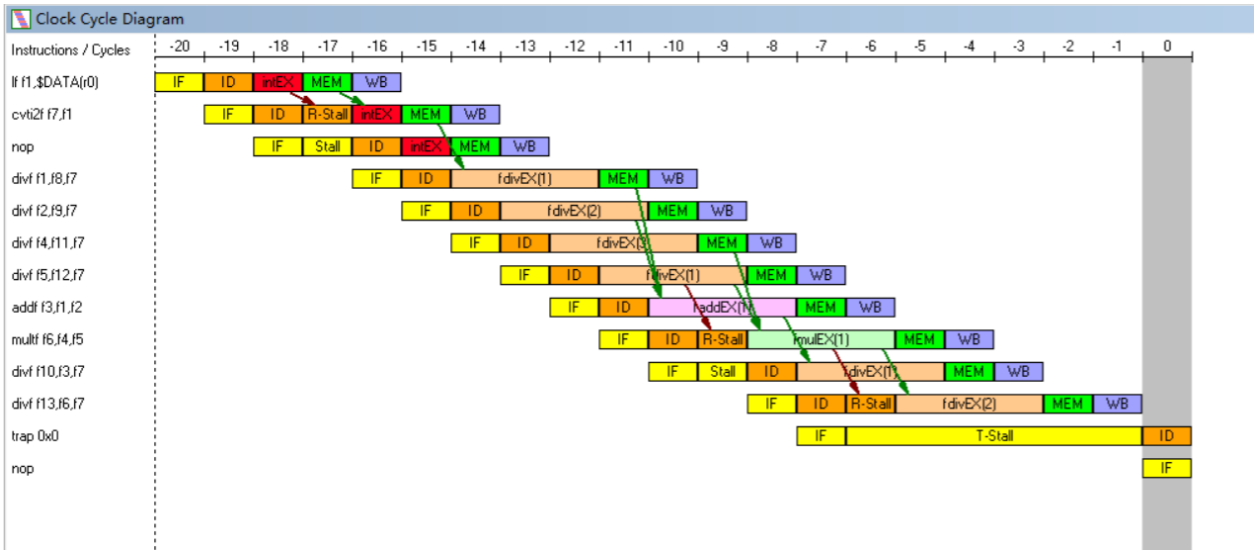
（1） 通过 Configuration 菜单中的“Floating point stages”选项，把除法单 元数设置为 3，把加法、乘法、除法的延迟设置为 3 个时钟周期。



（2） 用 WinDLX 模拟器运行调度前的程序 sch-before.s 。记录程序执行 过程中各种相关发生的次数以及程序执行的总时钟周期数。



（3） 用 WinDLX 模拟器运行调度后的程序 sch-after.s ，记录程序执行过 程中各种相关发生的次数以及程序执行的总时钟周期数。



（4） 根据记录结果，比较调度前和调度后的性能。

指标	调度前	调度后	优化效果
总时钟周期	26 周期	20 周期	减少 **19%**

（5） 论述指令调度对于提高 CPU 性能的意义。

1. 消除无关指令阻塞：
   - 原程序：addf 需等待 divf f1, f2 完成 → 阻塞 3 周期。
   - 优化后：将无关的 divf f4, f5 提前执行，隐藏除法延迟。
2. 提升指令级并行：
   - addf（Y+Z）和 multf（B×C）无数据依赖，可并行执行。
   - 调度后两者连续执行，充分利用流水线。
3. 减少 RAW 相关：
   - 原程序存在连续依赖链：
     divf→addf→divf 和 divf→multf→divf
   - 调度后缩短依赖路径，减少流水线停顿。

结论分析与体会：

1. 性能提升：总执行时间减少 19%（26→20 周期），消除了数据相关导致的阻塞
2. 硬件效率提升
3. 调度核心价值：

通过重组指令顺序：

（1） 将长延迟操作与独立指令交织

（2） 最大化并行执行独立任务

（3） 缩短关键路径依赖链

最终实现「用相同的硬件资源，完成更多有效工作」的 CPU 性能优化目标。