实验四——流水线性能实验

一、 实验目的

1) 学习如何使用 gem5 对程序进行性能分析

2) 评估不同流水线配置对系统整体性能的影响

3) 实际运用 Amdahl 定律

二、 实验步骤

步骤 I：更改工作负载 Workload

请按照以下要求设置模拟配置参数：

1）设置 CPU 模型为 HW2TimingSimpleCPU

2）分别设置工作负载为 DAXPYWorkload 和 HelloWorldWorkload

步骤 II：更改发射延迟和浮点操作延迟

请按照以下要求设置模拟配置参数：

1）设置 CPU 模型为 HW2MinorCPU

2）设置工作负载为 DAXPYWorkload

3）设置不同的发射延迟和浮点操作延迟组合

步骤III：更改整数操作延迟和浮点操作延迟

请按照以下要求设置模拟配置参数：

1）设置 CPU 模型为 HW2MinorCPU

2）设置工作负载为 DAXPYWorkload

3）设置不同的整数操作延迟和浮点操作延迟组合

三、 实验结果

（一）模拟前的问题

1) 如果将程序中的指令分为整数、浮点和内存指令三类，你认为每个类别在程序中所占比例是否相等？

在同一个典型的程序中，整数、浮点和内存指令的比例通常是不相等的。例如，在一些密码学算法程序中，整数指令往往占程序总指令的大多数。

2) 你认为不同的程序是否会有不同的三类指令构成比例？为什么？

不同类型的程序，三类指令构成比例也不相同。

对于控制流程密集型应用程序而言，这类程序涉及大量的条件判断、循环和分支语句。这些语句通常使用整数指令来进行比较、计数、跳转等操作。编译器和处理器在处理这些应用程序时，会生成和执行大量的整数指令。

而科学计算和数值模拟应用程序通常涉及对复杂的数学模型和方程进行求解和模拟。这些应用程序需要大量的浮点运算，如加减乘除、三角函数、指数函数等。因此，科学计算和数值模拟应用程序通常需要执行大量的浮点指令。

此外，数据库管理系统涉及对大量数据的读取、写入和查询操作。这些操作通常需要频繁地访问内存中的数据，包括加载数据到寄存器、存储数据到内存以及内存间的数据传输等。因此，数据库管理系统通常需要大量的内存指令。

（二）步骤 I

①DAXPYWorkload：

```
board.processor.cores.core.commitStats0.numFpInsts        524289
# Number of float instructions (Count)
board.processor.cores.core.commitStats0.numIntInsts        786441
# Number of integer instructions (Count)
board.processor.cores.core.commitStats0.numLoadInsts       262145
# Number of load instructions (Count)
board.processor.cores.core.commitStats0.numStoreInsts      131072
# Number of store instructions (Count)
```

浮点操作占 30.77%，整数操作占 46.15%，内存操作占 23.08%（其中读取操作占 15.39%，写入操作占 7.69%）。

②HelloWorldWorkload：

```
board.processor.cores.core.commitStats0.numFpInsts            12
# Number of float instructions (Count)
board.processor.cores.core.commitStats0.numIntInsts          6647
# Number of integer instructions (Count)
board.processor.cores.core.commitStats0.numLoadInsts         1256
# Number of load instructions (Count)
board.processor.cores.core.commitStats0.numStoreInsts        1246
# Number of store instructions (Count)
```

浮点操作占 0.13%，整数操作占 72.56%，内存操作占 27.31%（其中读取操作占 13.71%，写入操作占 13.60%）。

通过以上模拟结果可以发现，同一个程序中不同类别指令占比并不相等，且不同程序也有着不同的指令构成比例。

（三）步骤 II

1）在这 3 种参数配置组合中，哪一种是你发现的最佳设计组合？

序号	发射延迟	浮点操作延迟	仿真时间（ms）
1	3	2	2.873
2	2	3	2.460

3	6	1	2.510

第二组（发射延迟 2，浮点操作延迟 3）是最佳设计组合。

2）为什么你认为在问题 1)中选择的设计能达到最佳性能？请解释为什么更倾向于优化其中一种延迟而不是另一种？

第二组参数配置减少了发射延迟，使得在流水线上插入两条指令的间隔周期数减少，同一时刻流水线上有多条指令正在处理，从而提升了流水线的并行度，达到了最佳性能。

以第一组参数配置作为基准线，可以发现第三组配置通过优化浮点操作延迟来提升性能，但由于发射延迟的增加，无法充分利用流水线的并行处理优势。而第二组配置相比于第一组虽然浮点操作延迟略有增加，但由于减少了发射延迟，使得流水线的并行处理能力得以充分发挥，从而总体性能提升最多。因此，更倾向于优化发射延迟而不是浮点操作延迟。

（四）步骤III

序号	整数发射延迟	整数操作延迟	浮点发射延迟	浮点操作延迟	仿真时间（ms）
1	1	4	1	8	2.575
2	1	2	1	8	2.574
3	1	4	1	4	2.458

1）使用 Amdahl 定律和你从第一步收集的信息来预测每种改进情况相对于 baseline 的加速比。你会选择哪种设计？注意：你只能使用

步骤Ⅰ收集到的数据来回答该问题。

Amdahl 定律可以表示为

$$S_{latency} = \frac{1}{(1-p) + \frac{p}{s}}$$

其中，$S_{latency}$是整个任务执行的理论加速比，s 是从改进的系统资源中获益的任务部分的加速比，p 是受益于改进资源的任务部分占总执行时间的比例。

这里，我们可以进行一个粗略的估计，认为整数（浮点）操作延迟减半相当于性能加倍，即 s=2。同时，按照如下方法对 p 进行粗略计算（以整数操作延迟减半为例）：p=(整数操作延迟*整数指令比例)/(整数操作延迟*整数指令比例+浮点操作延迟*浮点指令比例+发射延迟*内存指令比例)。

从而可以得出理论加速比如下：

序号	整数发射延迟	整数操作延迟	浮点发射延迟	浮点操作延迟	理论加速比
1	1	4	1	8	1
2	1	2	1	8	1.146
3	1	4	1	4	1.229

因此，应选择浮点操作延迟减半的设计。

2）每种改进情况相对于 baseline 的加速比是多少？

序号	整数发射延迟	整数操作延迟	浮点发射延迟	浮点操作延迟	实际加速比

1	1	4	1	8	1
2	1	2	1	8	1.0004
3	1	4	1	4	1.0476

3）如果你对问题 1）和 2）的回答之间存在差异，你认为可能的原因是什么？

问题 1）和 2）中都是浮点操作延迟减半的加速比最大，但问题 2）的实际加速比小于问题 1）的理论加速比。这主要是因为在计算理论加速比时简单地将性能收益 s 视为 2，但在实际中由于流水线的并行，延迟减半带来收益小于 2。其次，问题 1）中对于收益任务部分的时间占比计算也并不准确，从而带来了计算的误差。

四、 实验心得体会

通过本次实验，我学会了使用 gem5 对程序性能进行分析，并了解了不同流水线配置对系统整体性能的影响。此外，我还掌握了如何运用 Amdahl 定律计算优化后程序的理论加速比。