西安交通大学
XI'AN JIAOTONG UNIVERSITY

《计算机系统综合设计实验》

Experiment on Computer Organization and Architecture

计算机科学与技术学院 实验中心

2023年10月

西安交通大学
XI'AN JIAOTONG UNIVERSITY

实验四、流水线性能实验

实验内容

实验四 流水线性能实验

实验内容

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's 定律。

实验内容

工作负载

本次实验使用 DAXPY 作为工作负载。DAXPY 循环（双精度 aX + Y）是在处理矩阵和向量的程序中经常使用的操作。以下是 C++ 代码实现的 DAXPY。

```
#include <cstdio>
#include <random>

int main()
{
  const int N = 131072;
  double X[N], Y[N], alpha = 0.5;
  std::random_device rd; std::mt19937 gen(rd());
  std::uniform_real_distribution<> dis(1, 2);
  for (int i = 0; i < N; ++i)
  {
    X[i] = dis(gen);
    Y[i] = dis(gen);
  }
```

```
// DAXPY 循环开始
  for (int i = 0; i < N; ++i)
  {
    Y[i] = alpha * X[i] + Y[i];
  }
  // DAXPY 循环结束

  double sum = 0;
  for (int i = 0; i < N; ++i)
  {
    sum += Y[i];
  }
  printf("%lf\n", sum);
  return 0;
}
```

实验内容

实验设置

- 先将gem-assighment-template切换到assign-2

```
git checkout assign-2
```

- 主板模型：在 components/boards.py 中，你可以找到所有的主板模型。在这次实验中，你只需要使用 HW2RISCVBoard。

- CPU 模型：在 components/processors.py 中，你可以找到所有 CPU模型。在 components/processors.py 中定义了一些类。在这次实验中，你需要使用的类（模型）是 HW2TimingSimpleCPU 和 HW2MinorCPU。

- 缓存模型：在 components/cache_hierarchies.py 中，你可以找到缓存层次结构的所有模型。在这次实验中，你需要使用 HW2MESITwoLevelCache。

- 内存模型：在 components/memories.py 中，你可以找到所有的内存模型。在这次实验中，你需要使用 HW2DDR3_1600_8x8。

- 时钟频率：在所有模拟中使用 4 GHz 的时钟频率。

实验内容

实验设置

- 在主配置脚本中，请确保使用以下命令导入 exit_event_handler。

```
from workloads.roi_manager import exit_event_handler
```

- 创建模拟器对象时，必须将 exit_event_handler 作为 on_exit_event 的关键字参数传递。使用以下模板创建模拟器对象。

```
simulator = Simulator(board={你的板卡的名称}, full_system=False,
on_exit_event=exit_event_handler)
```

实验内容

分析与模拟：步骤 I

- 在运行模拟之前，尝试回答以下问题，做出合理的假设。

  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？

  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？

- 注意： 推荐学习这些概念：算术强度、Roofline 模型。

- TimingSimpleCPU 是 gem5的一个CPU 模型，特点时将所有非内存指令的执行时间模拟为一个时钟周期。这个 CPU 模型可用于提取上述三类程序指令组合信息。你可以在 components/processors.py 中找到基于 TimingSimpleCPU 的 HW2TimingSimpleCPU 的定义。

- 编写一个配置脚本，模拟在 HW2TimingSimpleCPU 上执行 DAXPYWorkload。确保跟踪模拟输出信息以备后用。

实验内容

分析与模拟：步骤 I

- 在报告中，回答上面提到的两个问题，并通过模拟结果支撑你的回答。使用 workloads/hello_world_workload.py 中的 HelloWorldWorkload 作为第二个程序，以比较三类指令比例。完整模拟数据集应包括两个配置（一个用于 DAXPYWorkload，另一个用于 HelloWorldWorkload）。
- 创建配置脚本

下面是一个用于在HW2TimingSimpleCPU上模拟执行DAXPYWorkload的配置脚本。确保设置正确模拟输出以供后续分析。

```
# 导入 exit_event_handler
from workloads.roi_manager import exit_event_handler

    board = HW2RISCVBoard(
        clk_freq="4GHz", processor=cpu, cache_hierarchy=cache, memory=memory
    )
    board.set_workload(daxpy_workload)
simulator = Simulator(board=board, full_system=False,
on_exit_event=exit_event_handler)
```

- 在run.py基础上继续完善

- 请根据需要修改主板名称。此脚本将运行HW2TimingSimpleCPU上的 DAXPYWorkload，并导出模拟输出数据供后续分析使用。

分析与模拟：步骤 I

■ 在stats.txt中查找以下内容：

```
board.processor.cores.core.commitStats0.numFpInsts              0
# Number of float instructions (Count)
board.processor.cores.core.commitStats0.numIntInsts             0
# Number of integer instructions (Count)
board.processor.cores.core.commitStats0.numLoadInsts            0
# Number of load instructions (Count)
board.processor.cores.core.commitStats0.numStoreInsts           0
# Number of store instructions (Count)
```

■ 该统计信息表示处理器执行不同类别操作指令的分布情况。

实验内容

分析与模分析与模拟：步骤II

- 在这一步中，我们将编写一个配置脚本，使你可以使用 HW2MinorCPU 模拟 DAXPYWorkload。请自己了解如何实例化 HW2MinorCPU 的对象。请注意：虽然可以调用其构造函数（init）而不传递任何输入参数，但在实验中你需要设置这些值。请阅读 HW2MinorCPU 的文档，并了解传递给 init 的每个输入参数的含义。

- MinorCPU 是 gem5 的CPU 模型之一，它模拟了一个按顺序流水线处理的 CPU。HW2MinorCPU 基于 MinorCPU。MinorCPU 的默认功能单元池包括两个整数运算单元、一个浮点运算单元和一个 SIMD 单元。

- 修改你的配置脚本以更改发射延迟和浮点操作延迟。发射延迟是指在流水线中插入两个指令之间的周期数。发射延迟为 3 个周期意味着每 3 个周期插入一个指令到流水线中。浮点操作延迟是指完成浮点指令执行所需的周期数。

- 接下来，针对这两个延迟的不同组合，测试程序性能。为简单起见，从发射延迟为 3 个周期和浮点操作延迟为 2 个周期的初始值开始。假设发射延迟和浮点操作延迟的乘积始终保持为 6 的常数。评估以下不同配置组合的性能。

实验内容

分析与模分析与模拟：步骤II

#	发射延迟	浮点操作延迟
1	3	2
2	2	3
3	6	1

注意：确保记录所有模拟运行的输出以供后续分析。

- 在报告中，根据运行结果数据回答以下问题。这一步的完整模拟数据应包括 3 个配置（发射延迟和浮点操作延迟的 3 种可能组合）。

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？

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？

实验内容

分析与模分析与模拟：步骤II

- 在这一步中，我们将编写一个配置脚本，使你可以使用 HW2MinorCPU 模拟 DAXPYWorkload。请自己了解如何实例化 HW2MinorCPU 的对象。请注意：虽然可以调用其构造函数（init）而不传递任何输入参数，但在实验中你需要设置这些值。请阅读 HW2MinorCPU 的文档，并了解传递给 init 的每个输入参数的含义。

- MinorCPU 是 gem5 的CPU 模型之一，它模拟了一个按顺序流水线处理的 CPU。HW2MinorCPU 基于 MinorCPU。MinorCPU 的默认功能单元池包括两个整数运算单元、一个浮点运算单元和一个 SIMD 单元。

- 修改你的配置脚本以更改发射延迟和浮点操作延迟。发射延迟是指在流水线中插入两个指令之间的周期数。发射延迟为 3 个周期意味着每 3 个周期插入一个指令到流水线中。浮点操作延迟是指完成浮点指令执行所需的周期数。

- 接下来，针对这两个延迟的不同组合，测试程序性能。为简单起见，从发射延迟为 3 个周期和浮点操作延迟为 2 个周期的初始值开始。假设发射延迟和浮点操作延迟的乘积始终保持为 6 的常数。评估以下不同配置组合的性能。

实验内容

分析与模分析与模拟：步骤II

#	发射延迟	浮点操作延迟
1	3	2
2	2	3
3	6	1

注意：确保记录所有模拟运行的输出以供后续分析。

- 在报告中，根据运行结果数据回答以下问题。这一步的完整模拟数据应包括 3 个配置（发射延迟和浮点操作延迟的 3 种可能组合）。

- 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？
- 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？

实验内容

分析与模分析与模拟：步骤III

- 在这一步中，修改你的配置脚本以更改整数操作延迟和浮点操作延迟。假设我们的处理器具有非常快的解码速度，可以在 1 个周期内发射整数和浮点指令。接下来，我们将关注整数操作延迟和浮点操作延迟。
- 假设我们设定baseline整数操作延迟为 4 个周期，浮点操作延迟为 8 个周期。你只能将这些延迟中的一个减少为原来的一半。这意味着你可以构建一个整数操作延迟为 2 个周期，浮点操作延迟为 8 个周期的处理器，或者整数操作延迟为 4 个周期，浮点操作延迟为 4 个周期的处理器。
- 模拟baseline和两种可能的改进情况。以下是你需要进行实验的所有可能延迟组合的表格。

#	整数发射延迟	整数操作延迟	浮点发射延迟	浮点操作延迟
1	1	4	1	8
2	1	2	1	8
3	1	4	1	4

实验内容

分析与模分析与模拟：步骤Ⅲ

- 在报告中回答以下问题。

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。

2） 每种改进情况相对于baseline的加速比是多少？

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？

实验内容

提交

- 在报告中回答所有问题。使用清晰的推理和可视化来支持你的结论。请确保包含以下代码/脚本。

- Instruction.md： 包含有关如何运行模拟的说明。
- Automation： 用于运行模拟的代码/脚本。
- Configuration： 配置系统以进行模拟的 Python 文件。

谢谢！