计算机组织与体系结构实习 Lab 3.1

2017/10/29

该lab希望通过实现cache simulator，巩固同学对cache设计的理解。同时，通过实现多种cache的优化策略，加深同学对 cache 体系结构的认识。

1、Cache Simulator

1. 从网盘目录lab 3.1下载实习相关文件：https://pan.baidu.com/s/1o8wXJHg 密码: sann
2. 该lab给定模拟器框架，具体见Cache.tar.gz。请在给定的配置框架基础上，完成：

- 实现一个可配置Cache容量(Cache Size)、块大小(Block Size)、组相连度(Set Associativity)、替换策略(Replacement Policy，暂定为LRU)、命中延迟(Hit Latency)等参数的 Cache Simulator。
- 该 simulator 应提供读取 trace 文件的接口（格式参照样例 trace 文件）。
- 可以支持多层 cache 架构，比如 L2Cache
- 该 simulator 需要同时提供对下层 Memory 的仿真支持。比如能够体现末级缓存(last level cache, LLC)的 miss latency等（即主存访问平均延时）。
- 最终模拟结果需要正确统计访问总次数、Miss 总次数、 Miss Rate、 replacement 总次数、对下层存储的访问次数、访问总延时周期数等参数。
- 主存访问平均延时设置为 100 个时钟周期， CPU 和主存频率均默认为 2GHz。
- 各层 Cache 的自身延迟与各层存储介质之间的传输延时暂不考虑
- Replacement Policy 统一定为 LRU， 其他优化策略暂不考虑

2、检查要求

检查分成两部分：

要求1

使用附件中给定的 trace 通过单一层次 cache 测试。具体见trace.tar.gz。

1. 观察在不同的 Cache Size（ 32KB ~ 32MB ）的条件下， Miss Rate 随 Block Size变化的趋势，收集数据并绘制折线图。
2. 观察在不同的 Cache Size 的条件下， Miss Rate 随 Associativity（ 1-32 ）变化的趋势，收集数据并绘制折线图。
3. 比较 Write Through 和 Write Back、 Write Allocate 和 No-write allocate 的总访问延时的差异。

要求2

与Lab 2.2中的流水线模拟器联调，运行测试程序。

1. 该测试中cache的配置如下：

Level	Capacity	Associativity	Line size(Bytes)	WriteUp Polity	Hit Latency

L1	32 KB	8 ways	64	write Back	1 cpu cycle
L2	256 KB	8 ways	64	write Back	8 cpu cycle
LLC	8 MB	8 ways	64	write Back	20 cpu cycle

2. 测试程序自选

其他说明

1. cache.tar.gz是参考框架，可从网盘上下载。若使用该框架，请注意：

- 请自行添加/修改现有的类成员变量、成员函数和其他辅助函数，并在文档中详细描述其实现。
- 请尽量保留 Cache类的Handle Request 函数中已实现的控制流。若不采用该框架，也请尽量模仿该函数的大致流程。
- 这些框架**仅供参考**， 遇到不明确或不恰当的地方请根据需要修改。

2. trace.tar.gz 为测试样例。 样例中每行代表一次 cache 访问。
3. 实习报告参见pdf和md模板。

提交要求

每人需单独提交：

1. 实验报告1份。具体内容参照模板。
2. 要求1对应的实验代码。提交内容中还应包含 README 文档，简要描述 Simulator 的使用方法；主程序可以以命令行参数的形式指定 trace 文件，正确地加载并输出统计结果。
3. 要求2对应的实验代码。提交内容中应包含所有内容（Cache 部分、 CPU 部分和测试程序的ELF文件）。代码中应附上README文档，简要描述整个系统的使用方法。