实验报告-LAB4

甘晨 181240014

2020 年 1 月 11 日

1 实验进度:

1.1 实现 cache

完成了 cache 的实现。

1.2 cache 性能评估

实现了评估 cache 性能的模型。

1.3 不同 cache 比较

实现了 4 种不同大小的 cache 的性能比较。

2 实验过程

2.1 实现 cache

cache 的实现起来在思维上没有很高的难度，只是要注意地址划分和各种情况的考虑，以及对于 dirty 位，tag 的更新不要忘记就可以了。另外就是对于重复代码的封装问题，我的代码中用了很多 copy-paste，这导致很难维护，事实上可以另外设计为独立的函数，再调用效果应该会好不少。

这里对于框架代码有点疑问，在 common.h 文件中，MEM_SIZE 的注释明明是 1MB，可却左移了 25 位，且 main.c 中的结构体里的 addr 也是 28 位，很不一致。于是我把这些统统改为了 20，统一按照 20 位的主存地址来做的。

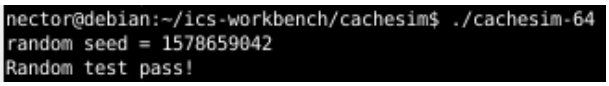

图 1: 测试通过

2.2 cache 性能评估

对于 cache 的性能评估，我采取了计算整个 test 过程中访问 cache 和访问内存各自的平均时间来比较。主要是通过静态变量记录 cache 访问次数和内存访问次数，再记录下各自总共的访问时间，做除法计算。对于 cache 访问时间，是从判断出 hit 开始记录，直到返回前的这段时间；对于内存访问时间，是从判断出 miss 开始记录，到返回前的这对时间。对于 256 行，4 路组相联的 cache，算得的平均访问时间是如下：

$$\begin{aligned} cache \quad &---42.20ns \\ mem \quad &---304.13ns \end{aligned}$$

2.3 不同 cache 比较

表 1: 时间对比

cache 种类	命中率 (%)	访问 cache 时间 (ns)	访问 mem 时间 (ns)	平均访问时间 (ns)
256 行 4 路组相连	20.72	26.48	61.21	75.01
4 行 2 路组相连	19.49	26.67	58.11	73.45
64 行 4 路组相连	19.78	26.81	60.61	75.43
512 行 4 路组相连	21.97	29.33	62.30	77.94

基于以上数据，可以发现，cache 行数越多一定程度上反映着命中率越高，但 cache 行数的增加之后，也可以看出平均的 cache 访问时间和平均访存时间都有所增加，这一点在 512 行 4 路组相联上体现的最为明显，但这似乎不是线性关系，从 64 行 4 路组相联到 256 行 4 路组相联上表现出这两个指标的下降，而平均访问时间也体现了这一规律。结合表格结果观察分析，256 行 4 路组相联的效果是最好的，命中率较高，且平均访问时间较短，相比之下 512 行 4 路组相联和 4 行 2 路组相联都只在一个指标上性能较好，另一个指标则表现很差。

3 后记

这次实验让我深深的感受到了 copy-paste 编程习惯的坏处，由于 cache_write 大部分时 copy-paste 的 cache_read 的内容，而在后来改变 cache 设计时，又是把之前写的整体 copy-paste，之后再修改参数，要改的地方很多，而且代码很冗余。正如前面所提到的，如果把 cache 里面常用的方法，如划分主存地址，回写等，封装成函数，在 cache_write 和 cache_read 中调用，这样实现起来应该会更加简洁，而修改 cache 设计时，修改参数应该也很方便。