计算机组织与体系结构实习报告 Lab3.2

学号：1500012752

姓名：薛犇

大班教师：程旭

cache管理策略优化（70分）

1. 根据Lab 3.2实习指导的要求，对默认配置下Cache进行优化。并使用附件中所给测试trace，对优化前后的cache进行比较。（20分）

(1) 请填写以下参数。（10分）

- 默认配置下，32nm工艺节点下，L1 Cache的 Hit Latency 为（ 1.47 ）ns，约等于（ 3 ）cycle
- 默认配置下，32nm工艺节点下，L2 Cache的 Hit Latency 为（ 1.92 ）ns，约等于（ 4 ）cycle

(2) 默认配置下，运行01-mcf-gem5-xcg.trace，结果如下：（10分）

- 运行trace共(10)遍
- L1 Cache： 平均 Miss Rate = （ 20.7% ）
- L2 Cache： 平均 Miss Rate = （ 34.2% ）
- AMAT =(11.37 ）

(3) 默认配置下，运行02-stream-gem5-xaa.trace，结果如下：（10分）

- 运行trace共(10)遍
- L1 Cache： 平均 Miss Rate = （ 11.3% ）
- L2 Cache： 平均 Miss Rate = （ 75.0% ）
- AMAT =(12.47 ）

2. 请填写最终确定的优化方案，并陈述理由。对于涉及到的算法，需要详细描述算法设计和实现思路，并给出优缺点分析。（40分）

- L1 Cache： prefetch + NRU
- L2 Cache：prefetch + NRU + bypass

(1) 替换策略：

- 最终方案: NRU : Not Recently used

这是一种放宽了的LRU策略，每个cache line会有一个标志位NRU_bit, 会周期性地更新标志位。在一个NRU周期T内，假如访问了某个cache line，就会把它的标志位置为1, 并把未访问的标志位置为0, 每次查询的时候，都会寻找标志位为0的cache line。程序会周期性地把所有标志位全部置0，本程序中把周期设成1000次cache 访问。

优点：极大得减少了替换策略所适用的时间，同时更加适合bypass策略的适用，因为bypass的目的就是为了保证某些有用的cache line不被替换出去，从而设计了很多技术让那些虽然最近不用，但是在未来仍有可能有用的cache line不被替换出去，而NRU的性质也迎合这一点。

- 备选方案: TREE-PLRU: TREE Pseudo LRU

对于8路组相连的cache，使用一个3层的二叉树，共8个叶节点，对应8路cache line。所以可以利用二叉树的遍历到哪个叶节点来表示替换哪个cache line。二叉树的节点有两个状态0和1，0表示向左子节点前进，1表示向右子节点前进，每次访问一个节点，就把这个节点的状态取反，这样可以利用二叉数的性质更高效地实现越早访问的越早被替换。

优点：减少替换策略使用的时间。需要的逻辑操作非常少，不需要加减操作，节能。

(2)预取策略：

- 基于stream buffer的Prefetch

采用8个stream，每个stream可以存放4个cache lined的数据。每一个stream是一个FIFO的队列，当取得某个地址时，和队首的地址做比较，如果相同，那么视为命中，直接从队首取出这个block并存入cache中，并更新队列的队首，当某个stream中没有元素时，将这个stream的valid置为0。当stream都满了的时候，使用FIFO策略选择一个victim stream。

(3)旁路策略：

- 采用Kharbutli在2013的工作SCIP: selective cache insertion and bypassing to improve the performance of last level caches

这是一个基于Reuse-Count的Bypass策略，它为每个block都设置了一个use bit和bypass count。use bit用来记录这个block最近一次是否被访问，bypass count用来记录这个block被bypass/access了几次。

策略如下： 当一个block被访问的时候，置它的use bit位为1，并把bypass count加1。然后判断这个bypass count是否小于3， 如果小于3，就视为这个block不怎么经常被访问，所以就bypass掉它，不给它在cache中分配空间。而当bypass count达到3的时候，就视为这个block是经常会被用到的了，所以把这个block分配到cache中。分配到cache中时，如果遇到该行已满的情况，就利用NRU算法挑出一个victim block。对于这个block的bypass/access信息，做如下的处理：如果它的use bit是1,那么设bypass count为3;反之，设成0。这相当于是对victim block的一种补偿，既然你被替换出去了，那么下一次你被访问的时候，就直接给你分配cache空间。

伪代码如下：

```
if cache hit on block B:
        B.use_bit = 1
```

```
else if cache miss on block B:
      if B.bypass_count < 3:
            B.bypass_count ++
            doBypass(B)
      else:
            find a victim block V using NRU
            V.bypass_count = V.use_bit * 3
            doInsert(B)
            B.use_bit = 0
```

3. 优化配置下，运行**01-mcf-gem5-xcg.trace（10次），结果如下：（10分）**

- L1 Cache： Miss Rate = （ 18.4% ）
- L2 Cache： Miss Rate = （ 25.6% ）
- AMAT =(8.77)

运行02-stream-gem5-xaa.trace （10次），结果如下：

- L1 Cache： Miss Rate = （ 2.3% ）
- L2 Cache： Miss Rate = （ 65.2% ）
- AMAT =(4.68)

可以看到，两个trace的AMAT显著下降。