****

**计算机组成与体系结构实践**

**Cache Simulator实验报告**



**学 院 智能与计算学部**

**专 业 计算机与科学技术**

**学 号 3019244140**

**姓 名 郭思齐**

# 1．实验目的

实验的目的是实现对Cache缓存的模拟。

成功通过验证之后，对每个benchmark trace分析：

1. 分析讨论各种体系结构参数对于Cache 缺失率的影响

①. L1 Cache size vs. miss rate

②. Associativity vs. miss rate

③. Block size vs. miss rate

1. 探索Cache 设计空间（design space），讨论其性能变化趋势，利用开发的仿真器，探索Cache 存储体系的设计空间，收集每种配置下的相关仿真结果（各种性能统计指标和AAT）。在实验报告中，讨论分析如下内容：

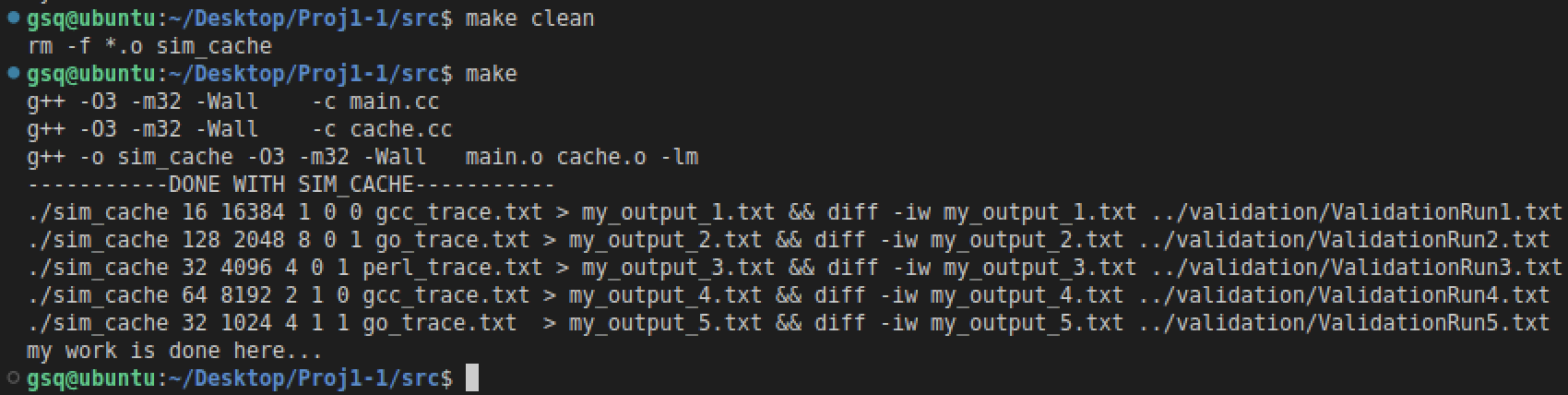
①.揭示随Cache 存储体系配置参数变化，性能指标（AAT）变化趋势。

②. 讨论性能指标的各种变化趋势如何受到各类配置参数变化的影响。

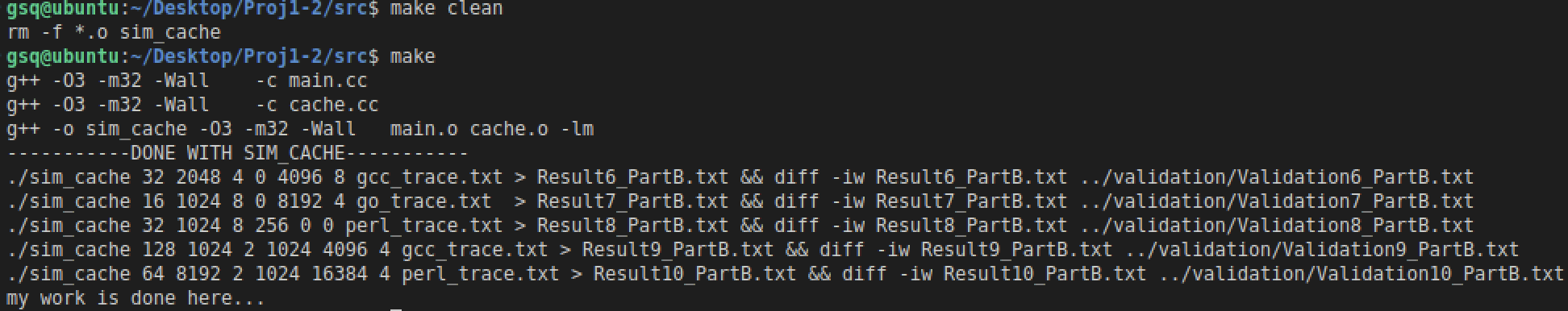
1. 寻找最优的Cache 存储体系配置方案
2. 通过设计空间探索，找出针对每种benchmark trace 的最优Cache 存储体系配置方案，即在满足面积约束下（Area <= Area Budget），AAT 最小的配置。

# 2. 实验结果

实验一实现对L1 Cache的模拟通过验证并成功得到以下结果：



实验二实现对L1 Cache，L2 Cache以及Victim Cache模拟，通过验证并成功得到以下结果：



# 3. 实验一 数据整理、分析以及可视化

## 3.1 分析讨论各种体系结构参数对于Cache 缺失率的影响

基于实验一，针对三个benchmark trace的不同配置参数进行比对分析Cache Size， Block Size 和 Associativity 对缺失率Miss Rate的影响。

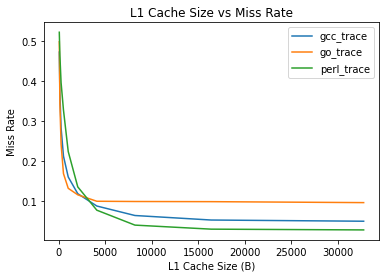
在策略选择上，3.1部分均与实验二保持一致，及WAWB（write-back + write-allocate）的Cache 写策略和基于LRU 的Cache 替换策略。

3.1.1 对比分析Cache Size对Miss Rate的影响：

Table 1 在Block Size = 16B， Associativity = 2的情况下Miss Rate受到Cache Size的影响

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cache Size (B) Benchmark   Miss Rate | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 |
| gcc\_trace Miss Rate | 0.4721 | 0.37 | 0.2829 | 0.2122 | 0.16 | 0.1183 | 0.0875 | 0.0638 | 0.0525 | 0.0495 |
| go\_trace Miss Rate | 0.4978 | 0.3428 | 0.2424 | 0.1685 | 0.1317 | 0.1155 | 0.0994 | 0.0988 | 0.0984 | 0.096 |
| perl\_trace Miss Rate | 0.5216 | 0.4634 | 0.3957 | 0.3275 | 0.2236 | 0.1353 | 0.0771 | 0.0398 | 0.0297 | 0.0278 |

结果可视化：



结果分析：

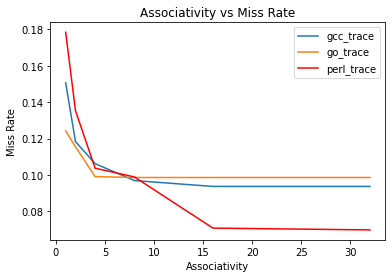
在Block Size和相联度一定的情况下，随着Cache Size的增大，能加载的数据越多，能在cache中找到数据的可能性越大，缺失率都会随之减小。但是随着Miss Rate趋近于某一个下限，缺失率下降得速度也会越来越慢。

3.1.2 对比分析Associativity对Miss Rate的影响：

Table 2在Block Size = 16B，Cache Size = 2048B的情况下Miss Rate受到Associativity的影响

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Associativity Benchmark   Miss Rate | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 |
| gcc\_trace Miss Rate | 0.1506 | 0.1183 | 0.1062 | 0.0969 | 0.0937 | 0.0937 | 0.0948 | 0.0954 | / | |
| go\_trace Miss Rate | 0.1242 | 0.1155 | 0.0991 | 0.0986 | 0.0986 | 0.0986 | 0.0986 | 0.0986 |
| perl\_trace Miss Rate | 0.1784 | 0.1353 | 0.1037 | 0.0989 | 0.0708 | 0.0698 | 0.0695 | 0.0697 |

结果可视化：



结果分析：

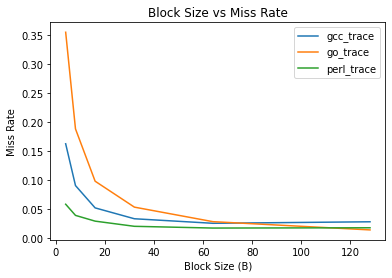
理论上，随着路数的增加，Cache Size和Block Size一定的情况下，组内块间的冲突可能降低，Miss Rate会先下降然后不变或稳定某一数值左右。在一路组相联也就是直接映射的情况下，Miss Rate最高。对比三个Benchmark，他们的情况也不尽相同，即Miss Rate达到最小值需要的组相联数不相同。Perl分支可以较好的接受比较大的组相联。

3.1.3 对比分析Block Size对Miss Rate的影响：

Table 3在Cache Size = 16384B， Associativity = 2的情况下Miss Rate受到Block Size的影响

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Block Size (B) Benchmark   Miss Rate | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 |
| gcc\_trace Miss Rate | 0.1628 | 0.0909 | 0.0525 | 0.0338 | 0.026 | 0.0286 | 0.0385 | 0.0505 |
| go\_trace Miss Rate | 0.3547 | 0.1884 | 0.0984 | 0.0538 | 0.0288 | 0.0146 | 0.0081 | 0.0128 |
| perl\_trace Miss Rate | 0.0588 | 0.0396 | 0.0297 | 0.0208 | 0.0178 | 0.0183 | 0.0299 | 0.0467 |

结果可视化：



结果分析：

理论上，随着Block Size的增大，Cache Size和Associativity一定的情况下，块容量增大使得可存数据量提升，减少替换次数的同时，降低冲突可能，Miss Rate同样会先逐渐下降然后稳定某一数值左右。三个分支都有体现出这样的趋势。

但是由于我设定的Cache Size是1638B，Block Size受到Cache Size的约束限制，后期随着Block Size的升高，块数量会降低，整体的Miss Rate会回升。

## 3.2 探索Cache 设计空间（design space），讨论其性能变化趋势

对于三个benchmark，探索Cache Size， Block Size和Associativity对AAT（Average Access Time (ns)）和通讯量（total memory traffic）两个性能指标的影响：

Table 4 Cache Size，Associativity，Block Size对性能指标AAT Average Access Time和TMT total memory traffic的影响

Benchmark #1: Block Size = 16B， Associativity = 2

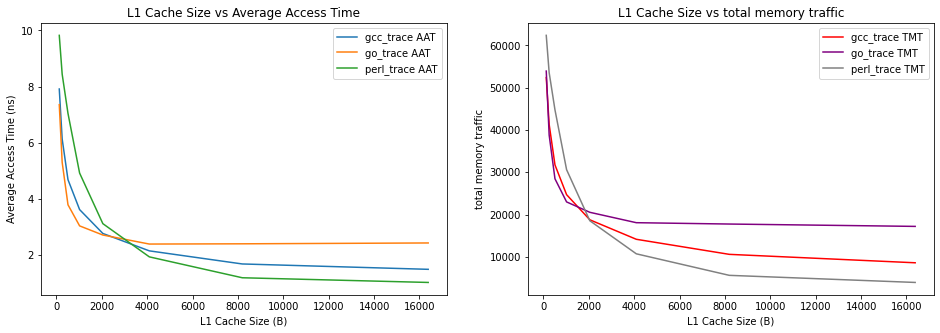
Benchmark #2: Block Size = 16B，Cache Size = 16384B

Benchmark #3: Cache Size = 16384B， Associativity = 2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Benchmark #1** | **Cache Size (B)** | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 |
| gcc\_trace | AAT (ns) | 7.9112 | 6.1251 | 4.6773 | 3.6103 | 2.7591 | 2.1385 | 1.6713 | 1.48 |
| TMT | 52393 | 41304 | 31737 | 24698 | 18754 | 14145 | 10584 | 8590 |
| go\_trace | AAT (ns) | 7.353 | 5.2956 | 3.7819 | 3.0295 | 2.7035 | 2.3814 | 2.3901 | 2.4203 |
| TMT | 53922 | 38793 | 28453 | 22983 | 20527 | 18070 | 17752 | 17195 |
| perl\_trace | AAT (ns) | 9.8249 | 8.4383 | 7.0406 | 4.9129 | 3.1088 | 1.9243 | 1.1791 | 1.012 |
| TMT | 62374 | 53550 | 44757 | 30608 | 18532 | 10710 | 5620 | 3939 |
| **Benchmark #2** | **Associativity** | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| gcc\_trace | AAT (ns) | 1.7551 | 1.48 | 1.4418 | 1.5312 | 1.7287 | 2.1279 | 2.9277 | 4.5277 |
| TMT | 10656 | 8590 | 8023 | 7970 | 7954 | 7946 | 7945 | 7945 |
| go\_trace | AAT (ns) | 2.4956 | 2.4203 | 2.4697 | 2.5695 | 2.7695 | 3.1693 | 3.9693 | 8.7693 |
| TMT | 17863 | 17195 | 17192 | 17191 | 17191 | 17190 | 17190 | 17190 |
| perl\_trace | AAT (ns) | 1.4777 | 1.012 | 1.0138 | 1.1062 | 1.2997 | 1.6974 | 2.4968 | 4.0968 |
| TMT | 7177 | 3939 | 3636 | 3599 | 3559 | 3540 | 3538 | 3537 |
| **Benchmark #3** | **Block Size (B)** | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
| gcc\_trace | AAT (ns) | 3.8848 | 3.6609 | 2.2303 | 1.48 | 1.1388 | 1.0501 | 1.264 | 1.8553 |
| TMT | 28246 | 27085 | 15008 | 8590 | 5349 | 3991 | 4157 | 5564 |
| go\_trace | AAT (ns) | 7.497 | 7.5223 | 4.2057 | 2.4203 | 1.5577 | 1.1122 | 0.9292 | 1.0055 |
| TMT | 60217 | 60217 | 32209 | 17189 | 9437 | 5010 | 2541 | 1405 |
| perl\_trace | AAT (ns) | 1.7575 | 1.5687 | 1.1933 | 1.012 | 0.8641 | 0.8704 | 1.0183 | 1.6153 |
| TMT | 7923 | 7061 | 4953 | 3939 | 2753 | 2412 | 2516 | 4378 |

(#1)

结果可视化：



结果分析：

随着L1 Cache Size的增大，Block Size和Associativity一定的情况下，性能指标AAT即Average Access Time和主存与Cache 间的通讯量TMT均会下降。

在WBWA的策略下，主存与Cache 间的通讯量TMT满足

由此，Cache Size增大带来的Miss的减少使得TMT同样降低。

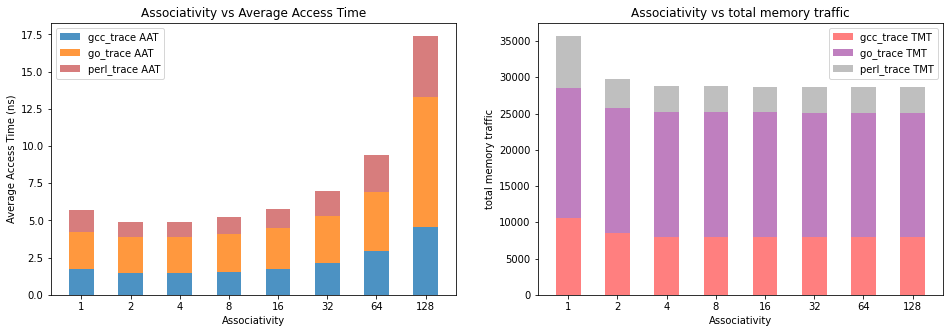
至于AAT，因为有公式：

因此前期受到Cache Size增大对Miss Rate降低带来的影响更大，导致AAT一直下降，但是最后Miss Rate受到Cache Size增大的影响不断减小，HTL1 会使得Miss Rate受到L1命中时间和L1缺失代价的影响更多。

同时，可以看出不同的分支AAT达到稳定的Cache Size各不相同。随着Cache Size增大，AAT降低的速率也不相同。

(#2)

结果可视化：



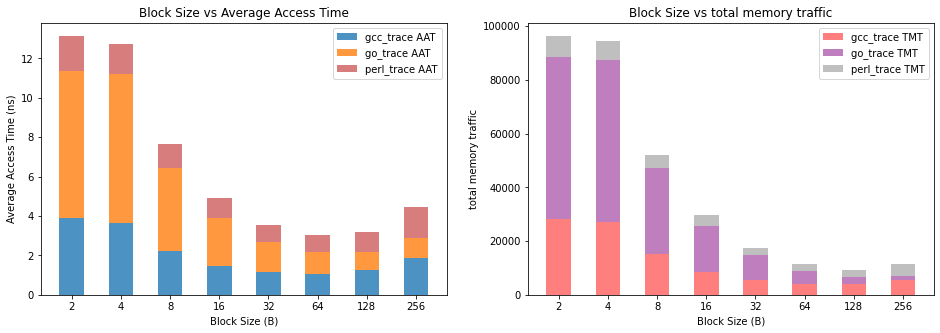
结果分析：

随着Associativity的增大，Block Size和Cache Size一定的情况下，性能指标AAT即Average Access Time会先小幅下降，之后上升；而主存与Cache 间的通讯量TMT会下降至趋于稳定。

主要分析AAT，对于Associativity而言，其增大对于Miss Rate的降低是有限的，因此前期AAT只有小幅降低，而后程 影响更大，因此后期AAT随路数增大在不断上升。

(#3)

结果可视化：



结果分析：

随着Block Size的增大，Associativity和Cache Size一定的情况下，性能指标AAT和TMT都会先下降，之后上升。关注AAT，分析过程与Associativity的分析过程类似。同样Block Size的增大开始会受到伴随Block Size增大而降低的Miss Rate的影响，但是后面Miss Rate的影响不再占据主导地位，根据公式中的 和 , 自身增大带来的影响会越来越大，使得AAT后面会有随Block Size增大的回升。

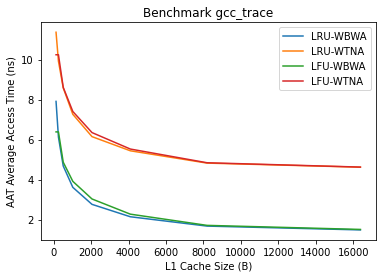
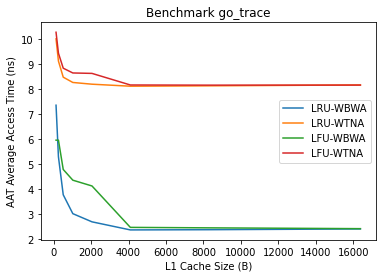
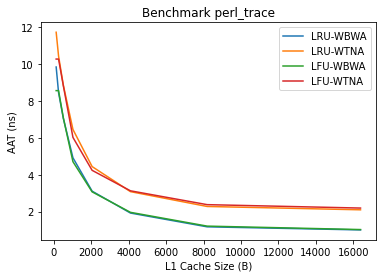
针对三个benchmark，探索不同的写策略和替换策略对结果指标的影响（以策略作为超参数，Cache Size作为自变量）：

Table 不同写策略和替换策略对性能指标AAT随Cache Size变化的影响

(Block Size = 16B，Associativity =2)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Benchmark** | | **Cache Size (B)** | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 |
| gcc\_trace AAT (ns) | **LRU-WBWA** | 7.9112 | 6.1251 | 4.6773 | 3.6103 | 2.7591 | 2.1385 | 1.6713 | 1.48 |
| **LRU-WTNA** | 11.3712 | 9.9204 | 8.587 | 7.2648 | 6.1475 | 5.4392 | 4.823 | 4.6187 |
| **LFU-WBWA** | core dumped | 6.385 | 4.8682 | 3.91 | 3.0285 | 2.2685 | 1.7076 | 1.5042 |
| **LFU-WTNA** | core dumped | 10.2454 | 8.6129 | 7.4059 | 6.3505 | 5.5269 | 4.8406 | 4.6237 |
| go\_trace AAT (ns) | **LRU-WBWA** | 7.353 | 5.2956 | 3.7819 | 3.0295 | 2.7035 | 2.3814 | 2.3901 | 2.4203 |
| **LRU-WTNA** | 9.9992 | 9.1119 | 8.4707 | 8.2597 | 8.1908 | 8.111 | 8.1258 | 8.1601 |
| **LFU-WBWA** | core dumped | 5.9596 | 4.7915 | 4.3641 | 4.1332 | 2.4827 | 2.4643 | 2.4316 |
| **LFU-WTNA** | 10.262 | 9.4264 | 8.8282 | 8.6385 | 8.6213 | 8.1595 | 8.153 | 8.1552 |
| perl\_trace AAT (ns) | **LRU-WBWA** | 9.8249 | 8.4383 | 7.0406 | 4.9129 | 3.1088 | 1.9243 | 1.1791 | 1.012 |
| **LRU-WTNA** | 11.705 | 10.4157 | 8.836 | 6.4465 | 4.438 | 3.0821 | 2.2804 | 2.0999 |
| **LFU-WBWA** | core dumped | 8.5467 | 7.0689 | 4.7021 | 3.0707 | 1.9628 | 1.2177 | 1.0312 |
| **LFU-WTNA** | core dumped | 10.2542 | 8.8052 | 6.0166 | 4.2353 | 3.1272 | 2.3788 | 2.194 |

结果可视化：



结果分析：

对于两种写策略两种替换策略两两组合，共有四种策略。因为Cache Size增大对AAT的影响最为直观，依次作图分析。可以看出WBWA情况下比WBNA的AAT更小。

而比较LRU和LFU，在实验一中，LRU也会略优一些。因此在之前的其他Cache 存储体系配置的比较中，使用的LRU-WBWA是合理的。

## 3.3 寻找最优的Cache 存储体系配置方案

综上，结合各种参数配置的规律，经验证及更多组数据尝试，在Area <= Area Budget = 512KB 条件下最优的Cache存储体系配置方案如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Benchmark | Cache Size (B) | Associativity | Block Size (B) | AAT (ns) |
| gcc\_trace | 32768 | 4 | 64 | 0.949 |
| go\_trace | 16384 | 2 | 128 | 0.9292 |
| perl\_trace | 32768 | 4 | 64 | 0.8065 |

后期，Cache Size的增大，其实为AAT性能提升带来了负担。

# 4. 实验总结

在本次Cache Simulator实验一中，我们实现了对L1级缓存的一个简单模拟。通过基于C语言实现缓存模拟器这一个过程，加深了对课件中概念和讲过的Cache知识的理解。

实验指导书在开始有很详细的引导，实验一在设计Cache的部分难度不是很大，但是实现两两组和四种策略情况，后期程序的复杂度比较高，遇到结果与预期不一致，通过Validation和Bug文档修复Bug的过程比较考验耐心。

在实验二后期，在设计Cache L2的过程中，因为用到了链表似的结构，设计有一些难度，但是实现的逻辑因为有实验一的预热，要容易一些。

实验整体下来，既让我对缓存地址，缓存方式，访问命中的计算，以及诸多存储体系配置如Cache Size，Associativity，Block Size，各种策略等等对Cache性能AAT和缺失率Miss Rate的影响有了更全面的认识，也又一次锻炼了C语言编程编写类、链表、结构体、调用各种API的基本功，