Cache 模拟器实验报告

姓名： 张煌昭
学号： 1400017707
院系： 元培学院
邮箱： zhang_hz@pku.edu.cn
手机： 17888838127

一．实验要求

本次 LAB 希望通过实现 Cache simulator，巩固对于 Cache 设计的理解，并通过实现多种 Cache 配置策略，加深同学们对于 Cache 体系结构的认识。

本次 LAB 分作如下两个部分。

1. Cache 模拟器

实现一个可以配置 Cache 容量（Cache size），块大小（Block size），组相联度（Set associativity）和命中延迟（Hit lantency）等参数的 Cache 模拟器，定替换策略（Replacement Policy）为 LRU。

要求该模拟器支持多层的 Cache 架构，并且对下层主存进行模拟。最终模拟结果需要统计访问总次数，Miss 总次数，Miss 率（Miss rate），替换总次数，对下层存储的访问次数，访问总延时周期等。

要求模拟器提供读取并模拟 trace 文件的接口。

规定主存延时为 100 各时钟周期，CPU 和主存频率默认为 2GHz。

2. Cache 与流水线模拟器联调

将 Cache 模拟器与流水线模拟器联调，并运行测试程序。

该 Cache 模拟器配置如下表：

表．与流水线联调的 Cache 配置

Level	Capacity	Associativity	Line size	Write-up policy	Hit latency
L1	32KB	8 ways	64	Write back	1 cpu cycle
L2	256KB	8 ways	64	Write back	8 cpu cycles
LLC	8MB	8 ways	64	Write back	20 cpu cycles
M	∞	--	--	--	100 cpu cycles

二．单层 Cache 模拟实现

参照提供的模版，由于各层 Cache 以及主存对于上一层的存储提供的操作均相同（读若干字节，或写若干字节），因而可以通过继承自一虚类来实现 Cache 和主存。又由于各层 Cache 独立地提供一组配置（包括大小，组相联方式，写回策略，换出策略等），且各层之间的策略互相独立，因此 Cache 的行为均可以在其类中描述。

1. 主存的行为

由于不论有几层 Cache，最下一层存储一定是主存（不考虑虚存的话），因此需要定义主存的行为。

主存提供读若干字节和写若干字节两种操作，并且由于主存是最下一层存储，因此对于主存的访问必然命中（Hit），不可能出现不命中（Miss），因此主存的行为如下。

主存对于读若干字节（Read bytes）的响应为读命中，该行为从存储中取出所需内容，通过总线传输至发出请求的上一级存储，需要经过总线延迟（Bus latency）和主存命中延迟（Hit latency）的延迟完成这一动作。

主存对于写若干字节（Write bytes）的响应为写命中，该行为通过总线传输从发出请求的上一级存储将待写入内容读入缓存，之后写入存储器中，需要经过总线演出（Bus latency）和主存命中延迟（Hit latency）的延迟完成这一动作。

对于主存的详细实现请见 code/cache/memory.h 和 code/cache/memory.cpp。

2. Cache 的行为

Cache 接收的请求为读若干字节和写若干字节，根据 Cache 当前的自身情况，可能出现的响应有——读命中（Read hit），读不命中（Read miss），写命中（Write hit）和写不命中（Write miss），由于配置中策略的不同，Cache 对于这四个响应的行为也不同，具体如下。

Read hit，所有策略的行为都相同，均为从存储中取出所需内容，通过总线传输至发出请求的上一级存储，需要经过总线延迟（Bus latency）和主存命中延迟（Hit latency）的延迟完成这一动作。

Read miss，首先找到空闲的块（valid=0）或待替换的块（LRU 计数最大），若写回策略为 Write back，则将被替换的块写回至下一层存储，之后向下一层存储发出读请求，读 Miss 的块至待替换的块中，之后再进行 Read hit 的操作，需要经过总线延迟（Bus latency）和主存命中延迟（Hit latency），再加上下一级的延迟来完成这一动作。

Write hit，若写回策略为 Write back，则设置 dirty 位为 1，之后通过总线传输从发出请求的上一级存储中读取待写内容至缓存，之后将其写入本级存储中。若写回策略为 Write back 则设置 dirty 位为 1，需要经过总线延迟（Bus latency）和主存命中延迟（Hit latency）来完成这一动作；否则写回策略为 Write through，需要向下一级存储也发出写请求，将待写内容也写入下一级存储之中，需要经过总线延迟（Bus latency）和主存命中延迟（Hit latency），再加上下一级的延迟来完成这一动作。

Write miss，若策略为 Write alloc，则需要像 Read miss 一样找到待替换的块，并根据写回策略决定是否将被替换的块写回至下一层存储，之后向下一层存储发出读请求，读取 Miss 的块至待替换的块中，之后再进行 Write hit 的操作，需要经过总线延迟（Bus latency）和主存命中延迟（Hit latency），再加上下一级的延迟来完成这一动作；若策略为 Write non-alloc，则直接向下一级传递该写请求，需要经过总线延迟（Bus latency）和主存命中延迟（Hit latency），再加上下一级的延迟来完成这一动作。

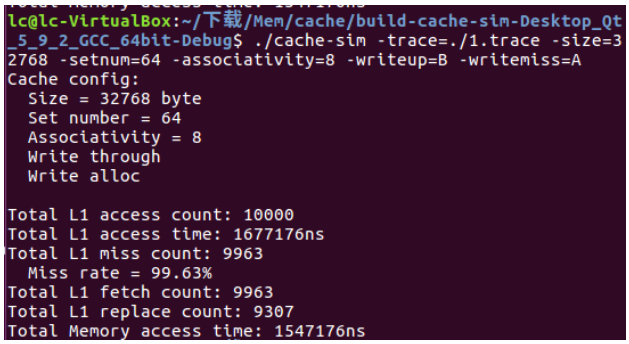
对于主存的详细实现请见 code/cache/ccache.h 和 code/cache/cache.cpp。

3. 单层 Cache 模拟器

联接 Cache 和主存，读取 trace 文件进行访存，通过命令行参数对 Cache 进行配置，详细代码请见 code/cache/main.cpp。

命令行参数有：-trace=，指定 trace 文件；-size=，指定 Cache size；-setnum=，指定 Cache set 数量；-associativity=，指定 Set associativity；-writeup=，指定 Write back / through 策略；-writemiss，指定 Write alloc / non-alloc 策略。

运行 1.trace 并配置 Cache，得到结果如下图所示。



三．单层 Cache 行为模拟

使用提供的两个 trace 进行如下实验，分别说明块大小（Block size），组相联度（Set associativity）和写策略对于 Cache 的不命中率（Miss rate）和访问延时的影响。

1. 固定 Cache size 时，Block size 对 Miss rate 的影响

固定 Cache size 为 32K，128K，512K，2M 和 8M，分别测试在 1.trace 和 2.trace 下，Block size 为 64，128，256，512 和 1024 时的 Miss rate，规定策略为 Write back，Write alloc 和 LRU 替换，规定 Set associativity 为 8，得到的结果如下表所示，绘制曲线如下图。

表．固定 Cache size 的条件下，使用 1.trace 测试得到的 Block size 对 Miss rate 的影响

Size = 32KB		Size = 128KB		Size = 512KB		Size = 2M		Size = 8M	
Block size	Miss rate	Block size	Miss rate	Block size	Miss rate	Block size	Miss rate	Block size	Miss rate
64	0.9963	64	0.9961	64	0.9960	64	0.9950	64	0.9950
128	0.5786	128	0.5629	128	0.5494	128	0.5475	128	0.5475
256	0.3518	256	0.3329	256	0.3082	256	0.3046	256	0.3046
512	0.2346	512	0.2112	512	0.1818	512	0.1749	512	0.1749
1024	0.1728	1024	0.1476	1024	0.1119	1024	0.1026	1024	0.1026

表．固定 Cache size 的条件下，使用 2.trace 测试得到的 Block size 对 Miss rate 的影响

Size = 32KB		Size = 128KB		Size = 512KB		Size = 2M		Size = 8M	
Block size	Miss rate	Block size	Miss rate	Block size	Miss rate	Block size	Miss rate	Block size	Miss rate
64	0.9998	64	1.0000	64	0.7969	64	0.6883	64	0.6883
128	0.5499	128	0.5487	128	0.4339	128	0.3496	128	0.3496
256	0.3041	256	0.3022	256	0.2565	256	0.1784	256	0.1784
		512	0.1668	512	0.1523	512	0.0913	512	0.0912
		1024	0.0932	1024	0.0898	1024	0.0472	1024	0.0472

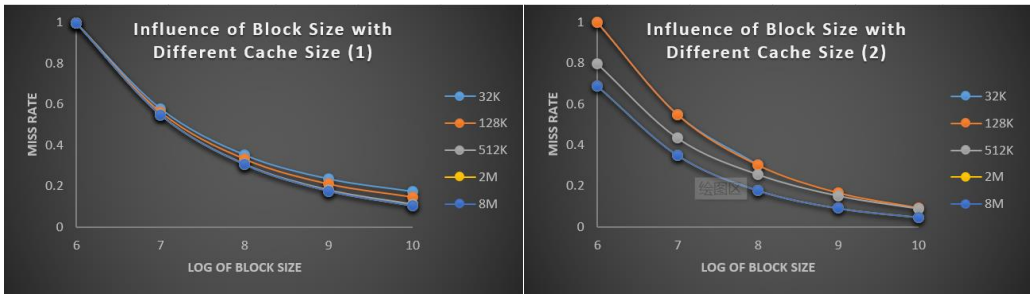


图. 固定 Cache size 和 trace 条件下，Block size 对 Miss rate 的影响曲线

根据曲线，可以发现固定 Cache size 时，随着 Block size 增大，Miss rate 会逐渐降低，并不断逼近某值。对其原因的分析如下。

只考虑一个块的内部，根据局部性原理，假定每一次访存的存储器地址，相对于上一次访存的地址的差值，符合正态分布，即有下式。

$$\Delta\text{Addr} = Addr_{curr} - Addr_{last} \sim N(\mu, \sigma^2)$$

由于上一次访存必然使得所考虑的 Cache 块中存放包含该地址的内存片段，那么本次访存命中的概率即与 Block size 相关，且满足下式。

$$\int_{block\ begin}^{block\ end} p(\Delta Addr)\, d(\Delta Addr), \text{其中 block end} - \text{block begin} = \text{block size}$$

上式略显粗糙，且只考虑一个 Block 并不完善，但足以说明结论。由于概率恒非负，因而 Block size 越大，该积分值越大；并且由于互斥概率和恒小于等于 1，因而随着 Block size 增大，该积分趋紧于 1。因此，固定 Cache size 时，随着 Block size 增大，Miss rate 会逐渐降低，并趋近于某定值。

2. 固定 Cache size 时，Associativity 对 Miss rate 的影响

固定 Cache size 为 32K，128K，512K，2M 和 8M，分别测试在 1.trace 和 2.trace 下，Associativity 为 4，8，16 和 32 时的 Miss rate，规定策略为 Write back，Write alloc 和 LRU 替换，规定 Set num 为 64，得到的结果如下表所示，绘制曲线如下图。

表. 固定 Cache size 的条件下使用 1.trace 测试得到的 Associativity 对 Miss rate 的影响

Size = 32KB		Size = 128KB		Size = 512KB		Size = 2M		Size = 8M	
Ways	Miss rate	Ways	Miss rate	Ways	Miss rate	Ways	Miss rate	Ways	Miss rate
32	0.9961	16	0.5623	32	0.3085	32	0.1031	32	0.0391
16	0.9965	8	0.3329	16	0.1799	16	0.0621	16	0.0251
8	0.9963	4	0.2138	8	0.1119	8	0.0391	8	0.0157
4	0.5805			4	0.0830	4	0.0261		

表. 固定 Cache size 的条件下使用 2.trace 测试得到的 Associativity 对 Miss rate 的影响

Size = 32KB		Size = 128KB		Size = 512KB		Size = 2M		Size = 8M	
Ways	Miss rate	Ways	Miss rate	Ways	Miss rate	Ways	Miss rate	Ways	Miss rate
32	0.9961	16	0.5488	32	0.2981	32	0.0479	32	0.0129
16	1.0000	8	0.3022	16	0.1632	16	0.0252	16	0.0069
8	0.9998	4	0.1696	8	0.0898	8	0.0129	8	0.0037

4	0.5506	4	0.0476	4	0.0007

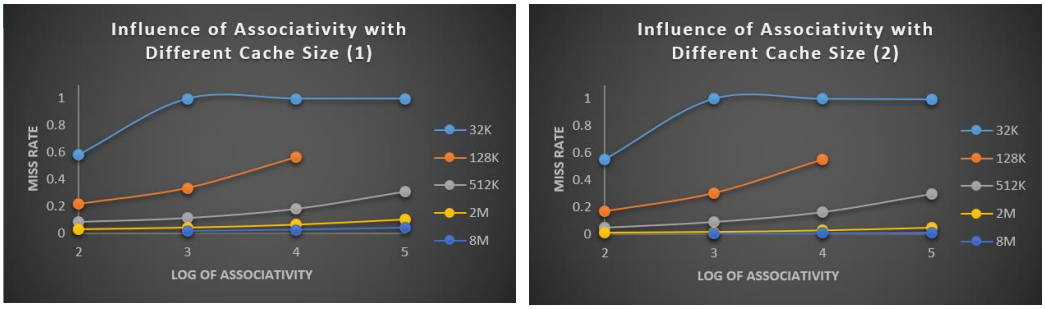


图. 固定 Cache size 和 trace 条件下，Associativity 对 Miss rate 的影响曲线

根据曲线，可以发现固定 Cache size 时，随着 Associativity 增大，Miss rate 会逐渐升高，并不断逼近 1。对其原因的分析如下。

由于试验中固定了 Cache size 和 Set num，因而 Associativity 增大，会间接地引起 Block size 减小，因而由于与上一部分相同的原因，Miss rate 升高。

3. 不同写策略对于总访问延时的影响

固定 Cache size 为 32KB，Associativity 为 8 路，在不同的写策略组合下（Write back+Write Alloc，Write back+Write non-alloc，Write through+Write alloc 和 Write through+Write non-alloc）使用 1.trace 测试 Block size 从 64 变化至 1024 带来的影响，得到的结果如下表所示，绘制曲线如下图。

表. 使用 1.trace 测试得到写策略对 Miss rate 和 Access time 的影响

Write back + Write alloc			Write through + Write alloc			Write back + Write non-alloc			Write through + Write non-alloc		
Block size	Miss rate	Time /ms	Block size	Miss rate	Time /ms	Block size	Miss rate	Time /ms	Block size	Miss rate	Time /ms
64	0.996	1.58	64	0.996	1.62	64	0.998	1.09	64	0.998	1.09
128	0.579	0.99	128	0.579	1.21	128	0.774	0.87	128	0.774	0.87
256	0.352	0.66	256	0.352	1.00	256	0.658	0.76	256	0.658	0.76
512	0.235	0.49	512	0.235	0.88	512	0.597	0.70	512	0.597	0.70
1024	0.173	0.40	1024	0.173	0.82	1024	0.567	0.67	1024	0.567	0.68

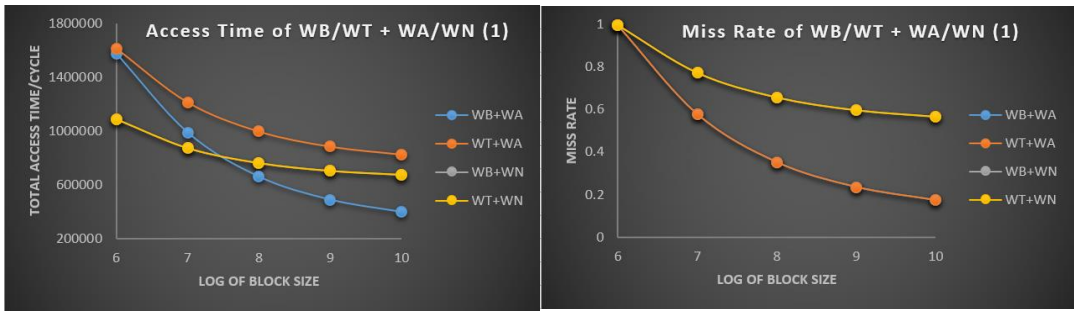


图. 不同写策略对 Miss rate 和 Access time 的影响

根据曲线，发现 Write back / Write through 对于 Miss rate 没有影响，但 Write back 策略使得 Access time 降低；Write alloc / Write non-alloc 对于 Miss rate 有影响，Write alloc 使得 Miss rate 降低。综合二者，Write back + Write alloc 策略在 Block size 足够时，是更优的策略。

四．与 Cache 联调的流水线模拟器

将 Cache 模拟按要求中所述的配置加入流水线模拟之中，简单起见，并未将数据也放入 Cache 模拟之中，而是由 Cache 模拟向流水线模拟提供访存的周期数，而流水线模拟仍然从一维线性的 Memory 数组中获取数据，仅仅根据访存周期对行为进行调整。

若有访存行为，则需要对 IF，ID，EX，EX2，以及 MEM 阶段进行 Stall，直至 MEM 阶段获得数据或将数据写入存储器中，向流水线模拟器中添加 wait_cache 函数进行上述操作，并使用宏定义 USE_CACHE 来选择使用 Cache 进行联调模拟。

详细代码请见 code/riscv-sim-modified 中相关文件。

使用联调模拟器运行 Lab 2.1 中的 add 和 simple-function 测试程序，得到的结果如下图，完全正确。

```
lc@lc-VirtualBox:~/下载/RISCV/risc
$ ./simulator ./add result

Pipeline-sim Mode Done!
Bypass technique applied.
Do nothing about control hazard.
Cache sim included.

Machine halt!

CPI = 2.441558
  Instructions = 231
  Cycles = 564
Ctrl hazard = 13
  Branch caused = 12
  Jalr caused = 1
  Wrong pred = 11
Data hazard = 52
L1 Cache stats:
  Total access time = 333 cycle
  Miss rate = 2.60%
    Access num = 77
    Miss num = 2
  Fetch num = 2
  Replace num = 0
L2 Cache stats:
  Total access time = 256 cycle
  Miss rate = 100.00%
    Access num = 2
    Miss num = 2
  Fetch num = 2
  Replace num = 0
LLC Cache stats:
  Total access time = 240 cycle
  Miss rate = 100.00%
    Access num = 2
    Miss num = 2
  Fetch num = 2
  Replace num = 0
memory block 0, result:
0x11010 - 0x11013: 0b 00 00 00
0x11014 - 0x11017: 0c 00 00 00
0x11018 - 0x1101b: 0d 00 00 00
0x1101c - 0x1101f: 0e 00 00 00
0x11020 - 0x11023: 0f 00 00 00
0x11024 - 0x11027: 01 00 00 00
0x11028 - 0x1102b: 02 00 00 00
0x1102c - 0x1102f: 03 00 00 00
0x11030 - 0x11033: 04 00 00 00
0x11034 - 0x11037: 05 00 00 00
```

```
lc@lc-VirtualBox:~/下载/RISCV/riscv-si
$ ./simulator ./simple-function result

Pipeline-sim Mode Done!
Bypass technique applied.
Do nothing about control hazard.
Cache sim included.

Machine halt!

CPI = 2.392562
  Instructions = 242
  Cycles = 579
Ctrl hazard = 15
  Branch caused = 12
  Jalr caused = 3
  Wrong pred = 13
Data hazard = 52
L1 Cache stats:
  Total access time = 337 cycle
  Miss rate = 2.47%
    Access num = 81
    Miss num = 2
  Fetch num = 2
  Replace num = 0
L2 Cache stats:
  Total access time = 256 cycle
  Miss rate = 100.00%
    Access num = 2
    Miss num = 2
  Fetch num = 2
  Replace num = 0
LLC Cache stats:
  Total access time = 240 cycle
  Miss rate = 100.00%
    Access num = 2
    Miss num = 2
  Fetch num = 2
  Replace num = 0
memory block 0, result:
0x11010 - 0x11013: 0b 00 00 00
0x11014 - 0x11017: 0c 00 00 00
0x11018 - 0x1101b: 0d 00 00 00
0x1101c - 0x1101f: 0e 00 00 00
0x11020 - 0x11023: 0f 00 00 00
0x11024 - 0x11027: 01 00 00 00
0x11028 - 0x1102b: 02 00 00 00
0x1102c - 0x1102f: 03 00 00 00
0x11030 - 0x11033: 04 00 00 00
0x11034 - 0x11037: 05 00 00 00
```

图．联调模拟器对 add 和 simple-function 测试程序的模拟结果

该模拟器中使用了三层 Cache 架构，自上而下分别为 L1，L2 和 LLC，配置与要求中完全一致。使用该模拟器对 Lab 2.2 中的十个测试程序进行测试，所有测试程序均由源码编译得到，测试结果如下组图。所有运行结果均正确。

图. 联调模拟器对 Lab 2.2 测试程序的模拟结果

统计 CPI 并与 Lab 2.2 中旁路优化后的 CPI 结果进行比较，如下表所示。

表. 联调模拟器 CPI 与流水线模拟器 CPI 的比较

Program	Instruction	Access	L1 Miss	Access Time	CPI of Ideal Memory	CPI of Cache hierarchy
1	39	17	2	273	1.2821	7.8461
2	39	17	2	273	1.2821	7.8461
3	128	45	2	301	1.2578	3.2578
4	178	65	3	449	1.2416	3.3989
5	319	45	2	301	1.1034	1.9060
6	274	35	2	291	1.0839	2.0182
7	158	54	3	438	1.2405	3.6709
8	81	22	3	406	1.1728	5.9136
9	79	18	2	274	1.1646	4.4051
10	68	14	2	270	1.1324	4.8971

发现所有测试程序 CPI 均有约 2 倍以上的提升。由于有流水线模拟器中理想存储器假设的存在，其瓶颈并不在于访存，而是在于数据相关和控制相关；而联调模拟器中取消了理想

模拟器假设，因而该实验中瓶颈在于访存——只有在 L1 hit 的情况下才可以与理想存储器的一周期访存一致，其他任何情况下，都会有 8-128 周期的损失。

因此在一个较短的程序中（例如测试程序，均在 1K 条指令以下），任何一次 LLC miss 都会导致 100 周期以上的损失，即相当于损失大于 10%的指令的周期数，这对于程序运行的 CPU time 的影响，是非常大的，因此如何减小访存延迟，是提升 CPU time 的重要的，甚至是最重要的一环。