（二）两级 **Cache** 仿真器

一. 实验目的：

- 设计一个可灵活配置的两级 Cache 存储体系仿真器。
- 基于该仿真器和 SPEC 标准测试程序对 Cache 存储体系的性能进行分析。

二. 实验环境和工具：

- 系统要求：Ubuntu XX.XX（32bit，x86）
- 编程工具：gcc
- 编程语言：C/C++/JAVA

三. 实验内容：

- 对实验内容（一）中单级 Cache 仿真模型进行扩展，设计一个灵活可配置的两级 Cache 存储体系仿真仿真器。
- 使用具有标准格式的访存地址流文件作为输入（该地址流已由 SPEC 标准测试程序产生），并将最终两级 Cache 中的存储内容和性能分析结果以标准格式输出到结果文件中。

(二). 两级 Cache 仿真器

该部分实验利用实验内容（一）中的单级 Cache 模型，例化 L1 和 L2 两级 Cache，构造两级 Cache 存储体系仿真器。其中 L1 Cache 将读/写请求发向 L2 Cache，而 L2 Cache 与主存储器进行交互，如图 1 所示。

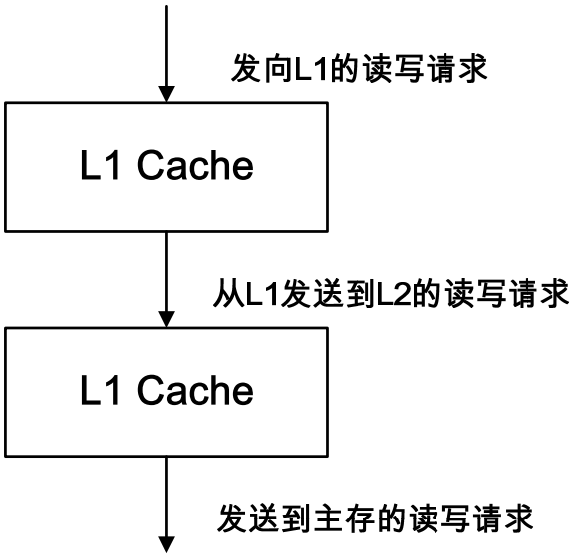

图 1. 两级 Cache 存储体系

L1 和 L2 Cache 将各自跟踪记录各自的性能计数器，如读次数，写次数、miss 次数等。在仿真结束后，仿真器可输出两级 Cache 的最终状态和原始性能指标。

L1 和 L2 可对各自的块大小，Cache 容量，相联度等体系结构参数进行配置（注：本实验中 L1 和 L2 具有相同的块大小），但替换策略和写策略不能配置。这些配置参数可通过命令行提供。

1. Cache 的配置

1.1 Cache 体系结构参数

参照实验（一）单级 Cache 的体系结构参数配置。

2. Cache 替换和写策略

2.1 Cache 替换策略

本实验中将实现基于 LRU 的 Cache 替换策略。

2.2 Cache 写策略

本实验将实现基于 WBWA（write-back + write-allocate）的 Cache 写策略。

3. 两级 Cache 的块分配：发送请求到存储体系的下一级

某级 Cache 在发生读或写缺失时，将势必与存储体系中的下一级（Cache 或主存储器）进行交互。此时在该 Cache 中，必定为此次缺失请求分配一个块 X。其分配过程包括两步：

(1). **为块 X 分配空间：**如果在被映射的组中有一个尚未分配的块（invalid），则直接将该块分配给块 X，掉转到步骤(2)。另一方面，如果被映射的组中所有的块都已经分配，则需要根据 LRU 替换策略挑选出某块 V（victim block）丢弃。对于 WBWA 策略，如果 V 是“脏”块，则需要将块 V 写回到存储体系的下一级。

(2). **将块 X 存入 Cache：**发射读 X 块的请求到存储体系的下一级，然后将块 X 写入 Cache 组中相应的位置（步骤（1）决定的位置）。

由此可见，当分配一个块时，Cache 将接连向存储体系的下一级发出一个写请求（如果需要替换一个块，并且此块为“脏”）和一个读请求。注意，这两个请求中的任意一个都有可能在存储体系的下一级发生 miss 缺失，从而将读写请求向存储体系的下一级进行传播，直到发送到主存储器。

4. 采用 Victim Cache 对 L1 Cache 进行改进

4.1 Victim Cache 的参数

Victim Cache 为全相联 Cache 结构，与 L1 Cache 具有相同的块大小。当 Victim Cache 的总容量设置为“0”的时候，Victim Cache 无效，即不工作。

4.2 Victim Cache 的操作流程

Victim Cache（VC）为一个小容量的全相联 Cache，采用 LRU 替换策略。Victim Cache 的操作流程分为以下几种情况：

(1). **L1 Cache 命中：**不对 VC 做任何操作。

(2). **L1 Cache 不命中，Victim Cache 命中：**将 VC 中命中的块与 L1 Cache 中的 LRU 块进行交换，并更新 VC 中各个块的 LRU 计数器。此种情况不需要访问存储体系的下一级。例如，当前的 L1 Cache 和 VC 如图 2 所示。当前某个读请求的块地址为“E”，该请求在 L1 Cache 中不命中，但在 VC 中命中。因为块“A”是 L1 Cache 中的 LRU 块，所以可将 VC 中的块“E”和 L1 Cache 中的块“A”进行交换，并修改相应的 LRU 计数器，如图 3 所示。

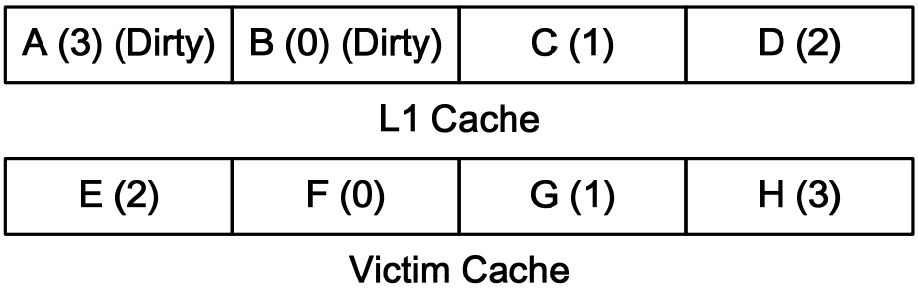

图 2. 访存请求到来前 L1 和 VC 中的情况

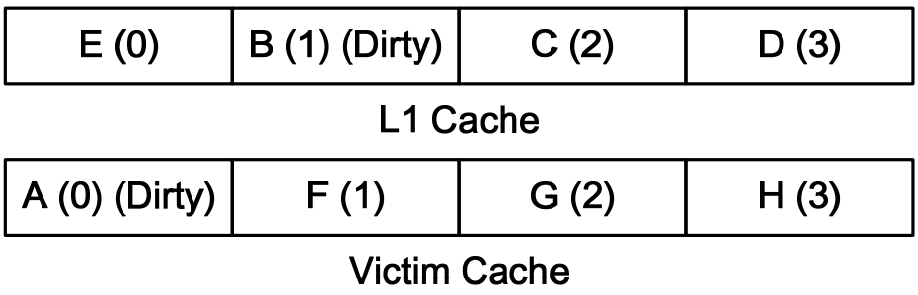

图 3. 访存后 L1 和 VC 中的情况

(3). **L1 Cache 和 Victim Cache 均不命中：**该种情况下，需要访问存储体系的下一级以获取所需的块。但在访问下一级存储器之前，又会出现两种情况：

①. **L1 Cache 不进行替换：**这种情况主要发生在“Cold”阶段，即在 L1 Cache

中还存在未分配的块。此时，仅需要将未分配的块分配给从存储体系的下一级传送来的块即可，VC 不需要进行任何操作。

②. **L1 Cache 发生替换：**此时，从 L1 Cache 中替换出来的块被存放到 VC 中，同时 VC 需要驱逐其中的 LRU 块。如果该 LRU 块是“脏”块，则将其写回下一级存储器，然后从下一级存储器读取所需的块，发送到 L1 Cache 中。例如，当前的 L1 Cache 和 VC 如图 4 所示。当前某个读请求的块地址为“K”，该请求在 L1 Cache 和 VC 中均不命中。根据 LRU 替换策略，L1 Cache 驱逐块“A”，并将其发送给 VC。VC 则驱逐块“H”，为块“A”腾出空间。因为块“H”为“脏”块，所以其将被写回下一级存储器，此时 L1 Cache 和 VC 如图 5 所示。

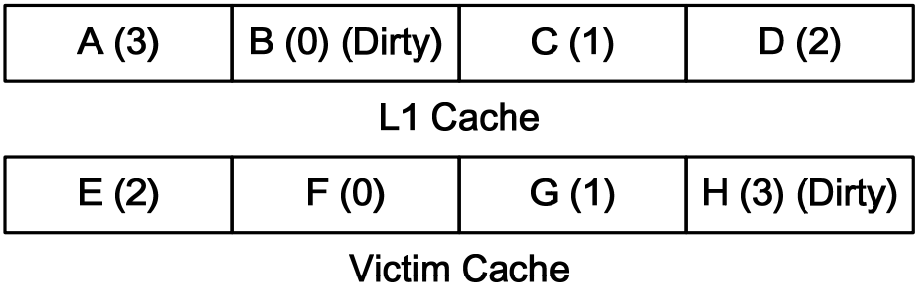

图 4. 访存请求到来前 L1 和 VC 中的情况

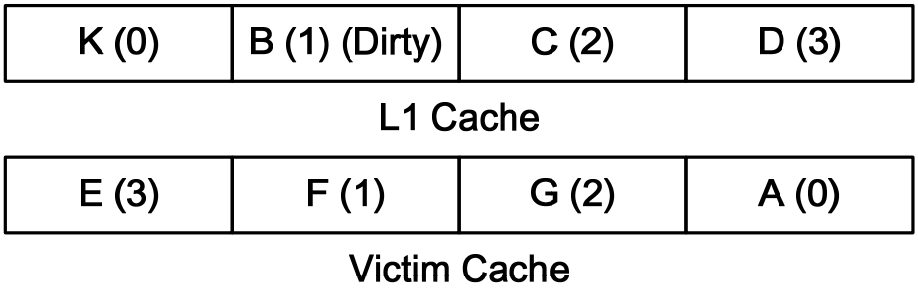

图 5. 访存后 L1 和 VC 中的情况

5. 所需进行的测试

L1 + L2 Cache，L1 cache + victim cache 和 L1 cache + victim cache + L2 cache。

6. 仿真结果的验证

该部分实验的验证与实验内容（一）类似。两级 Cache 仿真器需要从命令行接受如下参数（顺序不能改变）：

sim_cache <BLOCKSIZE> <L1_SIZE> <L1_ASSOC> <Victim_Cache_SIZE> <L2_SIZE> <L2_ASSOC> <trace_file>

命令行参数	解释
<BLOCKSIZE>	块大小（单位：B），2 的幂次，正数
<L1_SIZE> <L2_SIZE>	L1 Cache 的容量（单位：B），正数 L2 Cache 的容量（单位：B），正数
<L1_ASSOC> <L2_ASSOC>	L1 Cache 的相联度，至少为“1” L2 Cache 的相联度，至少为“1”
<Victim_Cache_SIZE>	Victim Cache 的容量（单位：B），当取“0”时，表示 Victim Cache 无效
<trace_file>	地址流文件名

例如：

$./sim_cache 64 1024 2 128 4096 8 gcc_trace.txt

两级 Cache 存储体系仿真器的编译、运行及验证要求与实验内容(一)相同。

7. 仿真结果打印的要求

本实验中要求在打印 Cache 的 tag 部分时，以 LRU 计数器的大小顺序进行打印，首先打印某组中 LRU = 0 的块，然后打印 LRU = 1 的块，以此类推，即最近最少使用的块（LRU 块）在最后打印，以保证和验证文件一致。**Victim Cache 中的内容也按同样的方式进行打印**。

8. 原始统计数据

本部分实验需要仿真器收集如下统计数据：

a. L1 读次数

b. L1 读缺失次数，不包括 L1 Cache 不命中，但 Victim Cache 命中的情况

c. L1 写次数

d. L1 写缺失次数，不包括 L1 Cache 不命中，但 Victim Cache 命中的情况

e. L1 缺失率 = MR_{L1} = (b + d)/(a + c)

f. 交换次数，即 L1 Cache 不命中，但 Victim Cache 命中

g. 从 Victim Cache 写回下一级存储器的次数

h. L2 读次数

i. L2 读缺失次数

j. L2 写次数

k. L2 写缺失次数

l. L2 缺失率（从延迟 CPU 操作来看） = MR_{L2} = i/h

m. 从 L2 写回主存储器的次数

n. 主存与 Cache 间的通讯量 = (i + k + m)——存在 L2 Cache

= (b + d +g)——不存在 L2 Cache

仿真器在仿真结束后以规定的格式打印存储体系的配置参数、原始统计数据以及各级 Cache 中的内容。

9. 性能分析

通过仿真器收集的原始性能指标，可对各级 Cache 的性能进行分析。

Cache 平均访问时间（Average Access Time, AAT）

Cache 平均访问时间 AAT 是指 Cache 服务一次从 CPU 发出的读/写请求的平均时间。对于 L1 和 L2 两级存储体系，AAT 可通过如下公式计算。

Average Access Time = HT_{L1} + (MR_{L1} *(HT_{L2}+MR_{L2}*Miss $Penalty_{L2}$))

其中，HT_{L1}（L1 命中时间）和 L1 缺失代价如下所示：

L2 命中时间 HT_{L2}（以 ns 为单位）= 2.5ns + 2.5ns × (L2_Cache Size / 512KB) + 0.025ns × (L2_BLOCKSIZE / 16B) + 0.025ns × L2_SET_ASSOCIATIVITY

L2 缺失代价 $MissPenalty_{L2}$（以 ns 为单位）= 20ns + 0.5 × (L2_BLOCKSIZE / 16B/ns)

四. 实验报告：10分

实验报告只包括实验内容（一）。L1 Cache 的面积必须满足面积约束 Area Budget（Area <= Area Budget = 512KB）实验报告中应当包括实验结果、分析和讨论几部分。实验报告应当对每个 benchmark trace 进行如下分析（gcc_trace, perl_trace, go_trace）：

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

2. 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. 采用图形或表格方式，揭示随着 Cache 存储体系配置参数的变化，其性能指标（AAT）的变化趋势。

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。

3. 寻找最优的 Cache 存储体系配置方案

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。

五. 实验工程的提交：

学生需要将设计完成的仿真器提交到课程 FTP 之上，以 ZIP 压缩包的形式提交，文件名为学号_姓名_ project1-2.zip。该压缩包中应该包含如下文件：

- 源代码
- Makefile 文件
- 实验报告，包含所有的数据，图和分析。

在 Linux 下创建 ZIP 压缩包的命令如下，假设源文件为.c 和.h 文件：

$$ zip 学号_姓名_ project1-2 *.c *.h Makefile

注意，如果在 windows 系统下打包，请直接把文件进行打包，不要将所有文件放在一个文件夹下，再进行打包。

六. 评分标准：

评分要求：

- 所有性能指标正确
- Cache 最终的存储内容正确

实验内容(二)满分 20 分，分值分布如下：

仿真器能够编译通过并运行		5
L1 + L2 Cache no Victim Cache	Validation Run # 5	3
	Validation Run #6	3
L1 with victim cache and no L2	Validation Run #7	3
L1, L2 with victim cache	Validation Run #8	3
	Validation Run #9	3

分数扣除（满分 60）：

- **根据 FTP 时间戳，每晚提交 1 小时扣除“2”分。**
- **仿真器无法编译通过将判为“0”分。**
- **上表中某相应项仿真发生错误或无法完成，扣除相应分数。**
- **使用比对工具对源代码检测，若发现抄袭现象，双方都计为“0”分。**

注：通过实验一工程路径下的 checklist.pdf 文件完成自查，确保符合所有评分要求。