15-213/18-213，2012年秋季
缓存实验室：理解缓存记忆分配：星期二，2012年10月2日到期：星期四，10月11日，晚上11：59
最后可能的上交时间：星期日，10月14日，晚上11：59

1 后勤

这是一个单独的项目。 您必须在64位x86-64机器上运行这个实验室。

地点：在这里插入任何其他后勤项目，如如何寻求帮助。

2 概述

这个实验室将帮助您理解缓存内存对C程序性能的影响。

实验室由两部分组成.. 在第一部分中，您将编写一个小的C程序（大约200-300行），它模拟缓存内存的行为。 在第二部分中，您将优化一个小矩阵转置函数，目标是最小化缓存丢失的数量。

3 下载作业

SITE-SPECIFIC：在这里插入一个段落，解释教师将如何向学生分发cachelab-handout.tar文件。

首先，将cachelab-handout.tar复制到一个受保护的Linux目录中，您计划在该目录中完成您的工作。 然后下达命令

```
Linux>tar xvf Cachelab-handout.tar
```

这将创建一个名为cachelab的目录，其中包含许多文件。 您将修改两个文件：csim.c和trans.c。 若要编译这些文件，请键入：

```
Linux>制作干净的Linux>
```

警告：不要让Windows WinZip程序打开您的.tar文件(许多Web浏览器被设置为自动这样做)。 相反，将文件保存到Linux目录，并使用Linuxtar程序提取文件。 一般来说，对于这个类，您不应该使用Linux以外的任何平台来修改文件。 这样做会导致数据丢失（和重要工作）！)。

4 描述

实验室有两个部分。 在A部分中，您将实现缓存模拟器。 在B部分中，您将编写一个矩阵转置函数，该函数是为缓存性能优化的。

4.1 参考追踪文件

讲义目录的跟踪子目录包含一个引用跟踪文件的集合，我们将使用这些文件来评估您在A部分中编写的缓存模拟器的正确性。 跟踪文件由一个名为valgrind的Linux程序生成。 例如，打字

```
linux>valgrind-log-fd=1-工具=lackey-v-trace-mem=是ls-l
```

在命令行上运行可执行程序“ls-l”，按照发生的顺序捕获其每个内存访问的跟踪，并在stdout上打印它们。

刻划记忆痕迹有以下形式：.

```
我0400d7d4，
8M0421c7f0，
4L04f6b868，
8
 S7ff0005c8，8
```

每一行表示一个或两个内存访问。 每行的格式是.

【空间】操作地址，大小

操作字段表示内存访问的类型：“I”表示指令负载，“L”表示数据负载，“S”表示数据存储，“M”表示数据修改（即数据负载后面跟着数据存储）。 在每个“我”之前从来没有一个空间。 每个“M”、“L”和“S”之前总是有一个空格。 地址字段指定64位十六进制内存地址.. 大小字段指定操作访问的字节数。

4.2 A部分：编写缓存模拟器

在A部分中，您将在csim.c中编写一个缓存模拟器，该模拟器以Valgrind内存跟踪作为输入，模拟缓存内存在此跟踪上的命中/丢失行为，并输出命中、错过和驱逐的总数。

我们已经为您提供了一个称为csim-ref的引用缓存模拟器的二进制可执行文件，它模拟了在valgrind跟踪文件上具有任意大小和结合性的缓存的行为。 它使用LRU（最不经常使用的）替换策略时，选择哪个缓存线删除。

参考模拟器采用以下命令行参数：

用法：./csim-ref[-hv]-s<s>-E<E>-b<b>-t<跟踪文件>

- -h：打印使用信息的可选帮助标志
- -v：可选的显示跟踪信息的冗长标志
- <>：设置索引位数(S=2^s是集合的数量)
- -E<E>：联想（每套线数）
- -b<b>：块位数(B=2^b块大小)
- <跟踪文件>：要重播的valgrind跟踪的名称

命令行参数基于CS：APP2e教科书第597页的符号(s、E和b)。 例如：

```
 Linux>./csim-ref-s4-E1-b4-t跟踪/yi.trace点击：4错
过：5驱逐：3
```

啰嗦模式相同的例子：.

```
 Linux>./csim-ref-v-s4-E1-b4-t跟踪/yi.trace L10，1错过
M20，1失球命中
L22，1命中
 S18，1击中
L  110，1错过驱逐
L210，1错过驱逐
M  12,1被驱逐者命中：4被驱逐
者：5被驱逐者：3
```

您在A部分的工作是填写csim.c文件，以便它接受相同的命令行参数，并产生与参考模拟器相同的输出。 请注意，这个文件几乎是完全空的。 你需要从头开始写。

A部分的编程规则

- 在csim.c的头注释中包含您的名称和登录ID。

- 您的csim.c文件必须在没有警告的情况下编译才能获得信用。
- 您的模拟器必须对任意s、E和b正确工作。 这意味着您需要使用malloc函数为模拟器的数据结构分配存储。 键入“manmalloc”以获得有关此函数的信息。
- 对于这个实验室，我们只对数据缓存性能感兴趣，所以您的模拟器应该忽略所有指令缓存访问(以“I”开头的行)。 回想一下，Valgrind总是将“I”放在第一列（没有前面的空间），而“M”、“L”和“S”放在第二列（有前面的空间）。 这可以帮助您解析跟踪。
- 要获得A部分的信用，您必须在主要功能的末尾调用函数打印摘要，其中包含命中、错过和驱逐的总数：

```
打印摘要(hit_count、miss_count、eviction_count);
```

- 对于这个实验室，您应该假设内存访问是正确对齐的，这样单个内存访问就不会跨越块边界。 通过做出这个假设，您可以忽略valgrind跟踪中的请求大小。

4.3 B部分：优化矩阵传递

在B部分中，您将在trans.c中编写一个转置函数，这将导致尽可能少的缓存丢失。

设A表示矩阵，$A._{ij}$ 表示第一行和JTH列上的组件。 A的转置，表示A^t是一个矩阵，使$A_{ij} = a^t$。

为了帮助您开始，我们给了您一个trans.c中的转换函数示例，该函数计算N×M矩阵A的转置，并将结果存储在M×N矩阵B：中

```
字符trans_desc[]=“简单的逐行扫描转置”；空转(intM，intN，
intA[N][M]，intB[M][N])
```

示例转置函数是正确的，但它效率低下，因为访问模式导致相对多的缓存丢失。

您在B部分中的工作是编写一个类似的函数，称为transpose_submit，它将不同大小的矩阵中的缓存丢失次数降到最小：

```
char transpose_submit_desc[]=“转呈”；
空洞transpose_submit(intM，intN，intA[N][M]，intB[M][N])；
```

不要更改transpose_submit函数的描述字符串(“Transpose提交”)。 自动记录器搜索此字符串以确定要评估信用的转置函数。

B部分的编程规则

- 在trans.c的头注释中包含您的名称和登录ID。
- 您在trans.c中的代码必须在没有警告的情况下编译才能获得信用。
- 允许每个转置函数最多定义12个int类型的局部变量。[1]
- 您不允许使用任何类型为long的变量或使用任何位技巧将多个值存储到单个变量，从而并行执行前面的规则。
- 转置函数可能不使用递归。
- 如果选择使用helper函数，则在helper函数和顶层转置函数之间，堆栈上的本地变量可能不超过12个。 例如，如果转置声明了8个变量，然后调用一个使用4个变量的函数，调用另一个变量
  函数使用2，您将在堆栈上有14个变量，并且您将违反规则。
- 转置函数可能不会修改数组A 但是，您可以对数组B的内容做任何您想做的事情。
- 不允许在代码中定义任何数组或使用malloc的任何变体。

5 评价

本节描述如何评估您的工作。 本实验室满分为60分：.

- A部分：27分
- B部分：26分
- 风格：7分

5.1 A部分的评价

对于A部分，我们将使用不同的缓存参数和跟踪来运行缓存模拟器。 有8个测试用例，每一个值3分，除了最后一个案例，这个值6分：.

```
Linux>     。   - 1 -e 1  - 1  -  痕迹/yi2.trace
        /csim  s         b    t
Linux>     。   - 4  - 2  - 4  -  痕迹
        /csim  s    e    b    t
Linux>     。   - 2  - 1  - 4  -  痕迹
        /csim  s    e    b    t
Linux>     。   - 2  - 1  - 3  -  跟踪/传输跟踪
        /csim  s    e    b    t
Linux>     。   - 2  - 2  - 3  -  跟踪/传输跟踪
        /csim  s    e    b    t
```

[1] 这种限制的原因是我们的测试代码无法计数对堆栈的引用。 我们希望您限制对堆栈的引用，并关注源数组和目标数组的访问模式。

```
 linux>./csim-s2-E4-b3-t            跟            踪
/trans.tracelinux>./csim-s5-E1-b5-t      跟      踪
/trans.tracelinux>./csim-s5-E1-b5-t跟踪/long.trace
```

您可以使用参考模拟器csim-ref获得这些测试用例的正确答案。 在调试过程中，使用-v选项详细记录每次命中和错过。

对于每个测试用例，输出正确数量的缓存命中、错误和驱逐将给您该测试用例的全部信用。 你所报告的每一次点击、错过和驱逐的次数都是该测试用例的三分之一。 也就是说，如果一个特定的测试用例值3分，并且您的模拟器输出正确的命中和错过次数，但是报告错误的驱逐次数，那么您将获得2分。

5.2 B部分的评价

对于B部分，我们将评估您的transpose_submit函数在三个不同大小的输出矩阵上的正确性和性能：

- 32×32(米=32，n=32)
- 64×64(米=64，n=64)
- 61×67(米=61，n=67)

5.2.1 性能(26pts).

对于每个矩阵大小，使用valgrind提取函数的地址跟踪来评估transpose_submit函数的性能，然后使用参考模拟器在具有参数(s=5、E=1、b=5)的缓存上重放此跟踪。

您对每个矩阵大小的性能评分与错过次数m成线性关系，达到某个阈值：

- 32×32：如果米<300，8分，如果米>600，0分
- 64×64：8分如果米<1300，0分如果米>2000
- 61×67：10分如果米<2000，0分如果米>3000.

您的代码必须是正确的，才能接收特定大小的任何性能点。 您的代码只需要在这三种情况下是正确的，您可以特别为这三种情况优化它。 特别是，您的函数可以显式地检查输入大小，并实现针对每个情况优化的单独代码。

5.3 风格评价

编码样式有7分.. 这些将由课程工作人员手动分配。 样式指南可在课程网站上找到。

课程人员将检查B部分中的代码是否有非法数组和过多的本地变量。

6 在实验室工作

6.1 在A部分工作

我们已经为您提供了一个自动评分程序，称为test-csim，它在参考跟踪上测试缓存模拟器的正确性。 在运行测试之前，一定要编译模拟器：

```
linux>制造
Linux>./test-csim
                        你的模拟器              参考模拟器
  要点  (s，E，          窝      小姐      驱逐      窝      小姐      驱逐
      b)
    3 (1,1,1)           9         8         6         9         8         6   痕迹/yi2.trace
    3 (4,2,4)           4         5         2         4         5         2   痕迹
    3 (2,1,4)           2         3         1         2         3         1   痕迹
    3 (2,1,3)         167        71        67       167        71        67   跟踪/传输跟踪
    3 (2,2,3)         201        37        29       201        37        29   跟踪/传输跟踪
    3 (2,4,3)         212        26        10       212        26        10   跟踪/传输跟踪
    3 (5,1,5)         231         7         0       231         7         0   跟踪/传输跟踪
    6 (5,1,5)      265189     21775     21743    265189     21775     21743   痕迹/长痕迹
   27
```

对于每个测试，它显示您获得的点数、缓存参数、输入跟踪文件以及来自模拟器和参考模拟器的结果的比较。

以下是关于A部分工作的一些提示和建议：

- 在小跟踪上进行初始调试，例如跟踪/Dave.Trace。
- 参考模拟器采用一个可选的-v参数，它允许冗长的输出，显示每次内存访问所产生的命中、错过和驱逐。 您不需要在csim.c代码中实现此功能，但我们强烈建议您这样做。 它将帮助您调试，允许您直接比较模拟器的行为与参考跟踪文件上的参考模拟器。
- 我们建议您使用getopt函数解析命令行参数。 您将需要以下头文件：

```
 包括<getopt.h>#包括
<stdlib.h>#   包   括
unistd.h>
```

详情见“man3getopt”。

- 每个数据加载(L)或存储(S)操作最多会导致一个缓存丢失。 的数据修改操作.
  (M)被视为负载，然后是存储到相同地址。 因此，M操作会导致

两次缓存命中，或错过和命中加上可能的驱逐。

- 如果您想使用15-122年的C0样式合同，您可以包括Contracts.h，为了您的方便，我们在讲义目录中提供了这些合同。

6.2 在B部分工作

我们已经为您提供了一个称为test-trans.c的自动评分程序，它测试您在自动评分器上注册的每个转置函数的正确性和性能。

您可以在trans.c文件中注册多达100个版本的转置函数。 每个转置版本有以下形式：.

```
/ * 标题评论*/
 char trans_simple_desc[]=“简单的转置”;
空洞trans_simple(intM，intN，intA[N][M]，intB[M][N])
{
    / * 你的转置代码在这里*/
}
```

通过对表单进行调用，向自动记录器注册特定的转置函数：

```
寄存器传输函数(trans_simple，trans_simple_desc);
```

在trans.c中的寄存器函数例程中。 在运行时，自动记录器将评估每个注册的转置函数并打印结果。 当然，注册功能之一必须是您提交的transpose_submit功能：

```
寄存器传输函数(transpose_submit，transpose_submit_desc);
```

请参阅默认的trans.c函数，以了解如何工作。

自动评分器以矩阵大小作为输入。 它使用valgrind生成每个注册转置函数的跟踪。 然后，它通过在具有参数(s=5、E=1、b=5)的缓存上运行参考模拟器来评估每个跟踪。

例如，要在32×32矩阵上测试注册的转置函数，重建test-trans，然后使用M和N：的适当值运行它

```
linux>制造
Linux>./test-trans-M32-N32
步骤1：评估注册转置Funcs的正确性：Func0（转置提交）：正确性：1
```

功能1（简单的行扫描转置)：正确性：1功能2(列扫描转置)：正确性：1
功能3(使用锯齿访问模式）：正确性：1

步骤2：生成注册转置功能的内存跟踪。

第三步：评估注册转置Funcs(S=5，E=1，b=5)Func0（转置提交）的性能：点击：1766，错过：287，驱逐：255
功能1（简单的行扫描转置)：点击：870，错过：1183，驱逐：151功能2(列扫描转置)：点击：870，错过：1183，驱逐：151功能3(使用锯齿形访问模式）：点击：1076，错过：977，驱逐：945

官方提交摘要(Func0)：正确性=1错过=287

在本例中，我们在trans.c中注册了四个不同的转置函数。 测试-trans程序测试每个注册函数，显示每个函数的结果，并为正式提交提取结果。

以下是关于B部分工作的一些提示和建议。

- test-trans程序保存文件trace.fi中函数i的跟踪。[2] 这些跟踪文件是非常宝贵的调试工具，可以帮助您准确地理解每个转置函数的命中和错过位置。 要调试特定的函数，只需使用详细选项：通过引用模拟器运行其跟踪

```
 Linux>./csim-ref-v-s5-E1-b5-t
trace.f0S68312c，1错过
 L683140，8小姐
 L683124，4击中
 升683120，4击中.
 L603124，4错过驱逐
S6431a0，4错过
...
```

- 由于您的转置函数是在直接映射的缓存上进行评估的，冲突丢失是一个潜在的问题。 考虑代码中冲突遗漏的可能性，特别是沿对角线。 试着考虑访问模式，这将减少这些冲突错过的数量。

- 阻塞是减少缓存丢失的有用技术。 你看

  http://csapp.cs.cmu.edu/public/waside/waside-blocking.pdf

  以获得更多信息。

[2] 因为valgrind引入了许多与代码无关的堆栈访问，所以我们从跟踪中过滤掉了所有堆栈访问。 这就是为什么我们禁止本地数组并限制本地变量的数量。

6.3 把所有的都放在一起

我们为您提供了一个名为./driver.py的驱动程序，它对您的模拟器和转置代码进行了完整的评估。这是你的老师用来评估你的手的程序。 驱动程序使用test-csim来评估您的模拟器，它使用test-trans来评估您提交的三个矩阵大小的转置函数。 然后它打印出你的结果和你所获得的分数的摘要。

要运行驱动程序，键入：

```
linux>./driver.py
```

7 交你的工作

每次在cachelab-handout目录中键入make时，Makefile都会创建一个tarball，称为userid-handin.tar，包含当前csim.c和trans.c文件。

地点-专业：在这里插入文本，告诉每个学生如何在你的学校上交他们的useridhandin.tar文件。

重要事项：不要在Windows或Mac机器上创建HandinTarball，也不要以任何其他存档格式提交文件，如.zip、.gzip或.tgz文件。