分两个部分

* 第一个部分：写一个小型C程序模拟存储
* 第二部分：优化一个矩阵转置函数，最小化缓存不命中。

更改csim.c，trans.c

Make clean

Make

一定要在linux系统下解压

**引用跟踪文件**

Traces子目录包含引用追踪文件，被用来评估第一部分写的C程序模拟器的正确性

安装valgrind这个linux程序，指令如下

linux> sudo apt install valgrind

linux> valgrind --log-fd=1 --tool=lackey -v --trace-mem=yes ls -l

“ls -l”后缀 —— 按访问顺序捕获每个内存访问的跟踪，并在stdout上打印它们。

有如下形式

I 0400d7d4,8

M 0421c7f0,4

L 04f6b868,8

S 7ff0005c8,8

它的普适格式是

[space]operation address,size

Operation域：内存访问的类型，“I”是指令加载，“L”是数据加载，“S”是数据存储，“M”是数据更改（数据加载伴随着数据存储），除了I，其它都有一个空格前缀。

Address域：指定了一个六十四位的十六进制地址。

Size域：操作所能访及的字节数

**Part A 写内存模拟器**

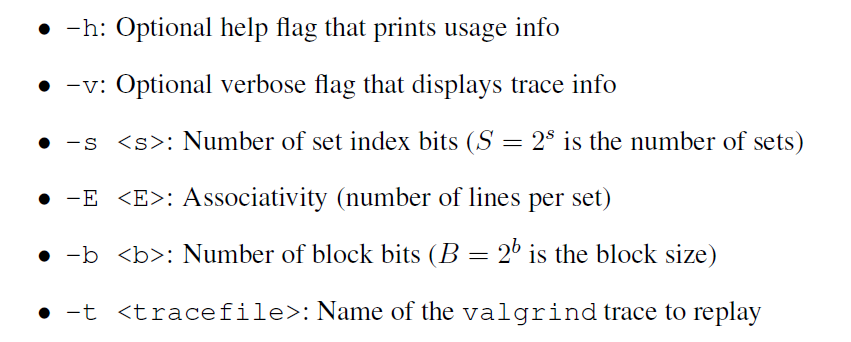
在csim.c内编写，以valgrind内存跟踪作为输入，在这个跟踪中模拟中/失行为，输出是命中、失、驱逐的总次数。

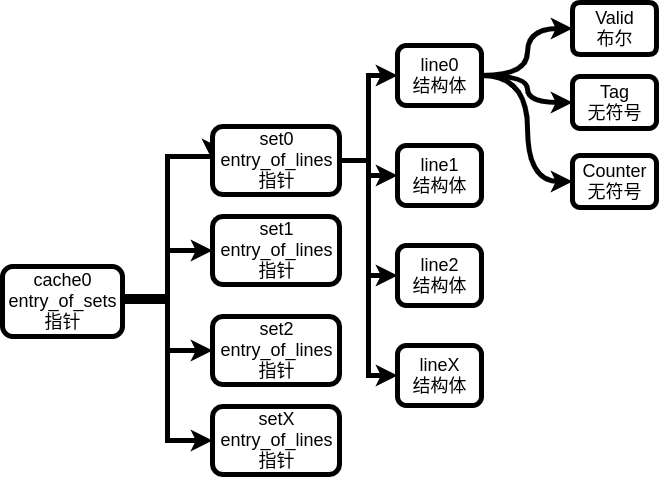
提供了一个参考存储模拟器的二进制可执行文件，叫csim-ref，模拟了任意尺寸缓存的行为和与valgrind跟踪文件的关联性，它用了LFU替换策略，

<https://en.wikipedia.org/wiki/Cache_replacement_policies>

参考模拟器的用法

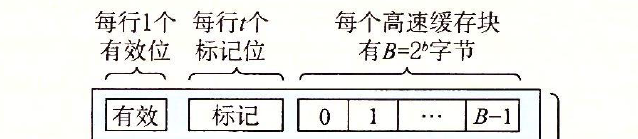
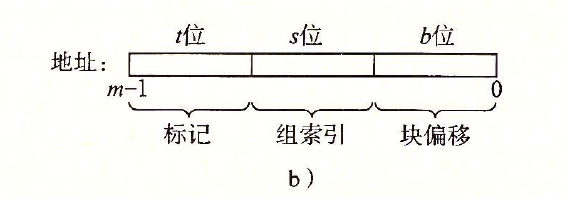
Usage: ./csim-ref [-hv] -s <s> -E <E> -b <b> -t <tracefile>





malloc得到的指针到底定义成什么类型。如果是C++ 我可以用 int(\*)[M]类型，然后用auto传参给其他函数，但是在C里这样是不行的，所以只能退而求其次，定义成int\*\*类型，然后给每行申请空间。（记得最后先按行释放，再释放二级指针！）

fgets函数，先统一读一行到buffer字符串中，再用sscanf进行分析。





将原始地址分为三个部分：CT, CI, CO

**Part B 最优化矩阵转置**

在trans.c中写一个转置函数，要引起尽可能少的缓存不命中。

A表示矩阵，Aij表示第i行第j列的元素，AT定义同转置矩阵。

示例程序效率不高，因为有很多缓存不命中。

transpose\_submit函数，在不同大小的矩阵中最小化缓存不命中

最多定义12个整型本地变量

不能用递归

栈中不能超过12个

不能更改数组A，数组B随喜

不能定义任何数组，使用malloc的变体

cache规模为：32组，直接映射，每行32字节数据

**32\*32**

我们发现A和B中对应的元素都是映射到同一行的。

我们在加载A的第二行之后，在将A[2][2]传送给B[2][2]的时候，就发生了冲突——A的第二行被替换掉了。当然，过了一会，A的第二行又回来了。进一步观察会发现，这种冲突只会发生在处理对角线的块上。

空间换效率——把一行全部读到临时变量里，达到一次性读完的效果。这样就不存在对角线时的冲突访问情况（a[i][i]=b[i][i]的时候发生的冲突）