**齐鲁工业大学实验报告** 成绩

课程名称 计算机组成原理 指导教师 张睿超 实验日期 2021/11/9

院（系） 网络空间安全学院 专业班级 网安19-1 实验地点 网安实训中心

学生姓名 焦翔宇 学 号 201903150004 同组人 无

实验项目名称 Cache和虚拟存储软件仿真实验

1. **一、 实验目的**

理解cache三种不同的地址映射机制，理解不同访问序列对cache性能的影响，能通过cache仿真软件分析、验证cache数据加载的正确流程。熟悉虚拟存储器系统结构以及工作流程，能够利用内存访问可视化插件进行虚存页面访问流程的分析。

1. **二、 实验环境**

Camera.jar

Camera是利用Java开发的一款用于高速缓存cache和内存资源分配模拟的仿真软件，如图3.13所示，可以用于模拟仿真cache的不同映射策略方案和虚存管理，能够有效地帮助用户更好地理解cache、虚拟存储器相关概念。

**三、 实验任务**

(1)利用CAMERA仿真软件依次彷真1路组相联(直接相联映射),2路组相联映射,4路组相联映射,全相联映射,对于每一种映射方式,利用随机生成的访问序列进行数据入仿真,对于每一次内存访问,仔细观察内存地址如何分成标记位TAG ,索引Index,字地址word。另外,仔细观察除了请求数据外,还有哪些数据同时被装载到Cache中。注意, CAMERA仿真器中地址序列的地址是字地址,实际计算机系统中地址序列应该是字节地址。

(2)设计一组地址访问序列,当这组访问序列循环时,直接相联映射cachel会全部缺失,而全相联映射则会全部命中,在CAMERA仿真器中输入对应序列进行验证。

(3)设计一组地址访问序列,当这组访问序列循环时,全相邻映射cache会,全部缺失,但直接相联映射的缺失率不高,在CAMERA仿真器中输入对应序列进行验证。

(4)虚存载入过程观察:单击CAMERA软件界面右侧的自动生成虚拟地址访问序列按钮,系统将自动生成虚拟地址访问序列,假设这些虚拟地址都是由MIPS指令中的LW指令生成的,单击Next按钮开始虚存模拟仿真。注意,每单击一次Next按钮,都要仔细阅读PROGRESS UPDATE文本框的文字内容,尝试真正了解每一次虚存访问TIB表、页表以及物理内存的变化。仿真结束后,记下虚存访问序列,同时记下TLB以及页表中缺失的次数,记下的访问序列,是否发生命中,分析对应原因。.

(5)最糟糕的访问序列:设计一个包含10个地址的虚存访问序列,使得虚存系统会产生10次TLB缺失和10次页缺失,设计完成后,可以利用定制地址访回序列按钮一次性输入10个地址,然后进行仿真验证自己的设计。

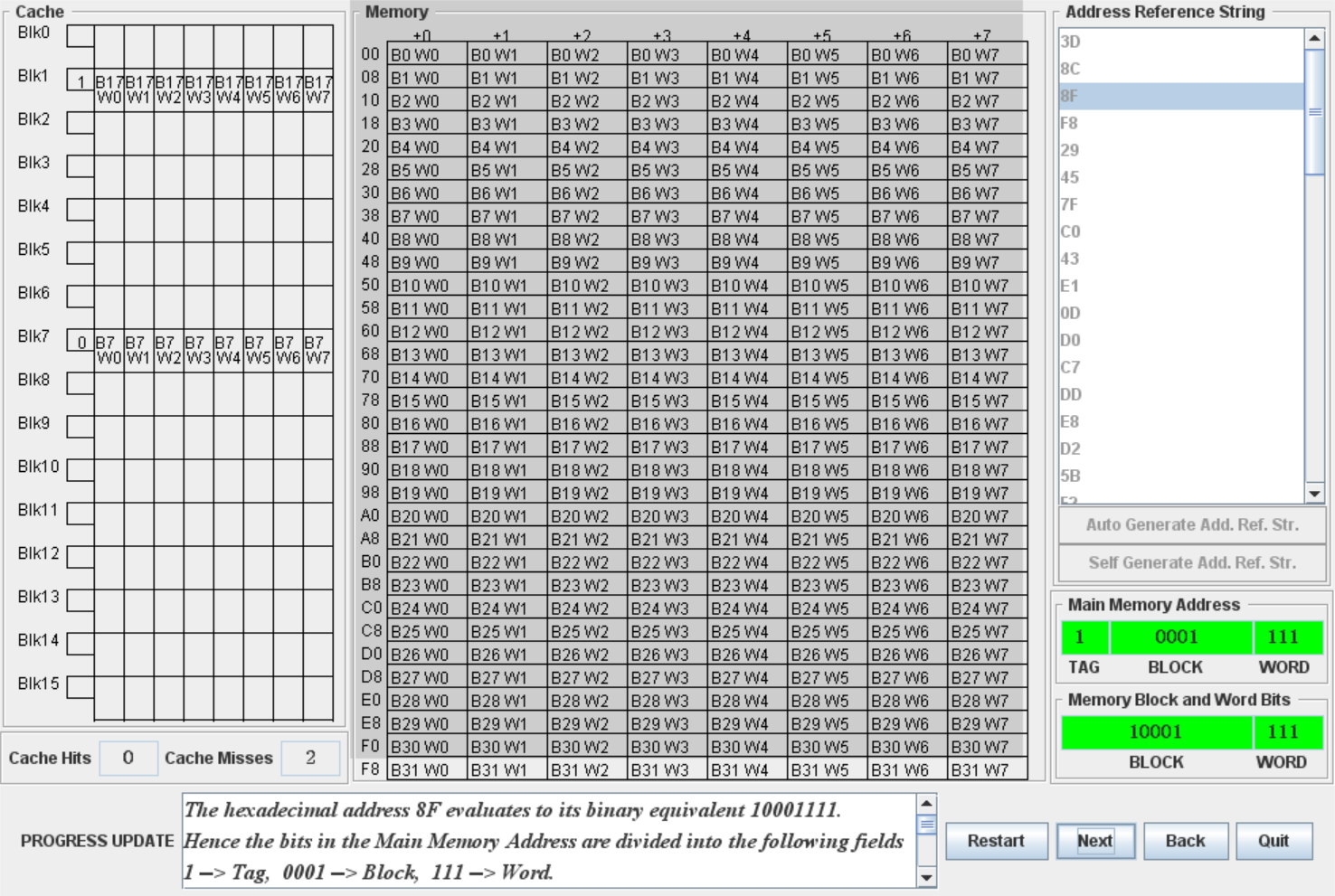
(6)存储系统优化：对于实验2中设计的能产生10次TLB缺失以及10次页缺失的访问序列,能否通过调整TLB大小,页表大小,内存大小等其中的一个参数,使得TLB缺失次数仍然为10,但页缺失次数小于10,并给出你的答案。

1. **四、 实验思路及结果分析**
2. 实验1

除了请求数据外，同块的数据也加载到缓存中。

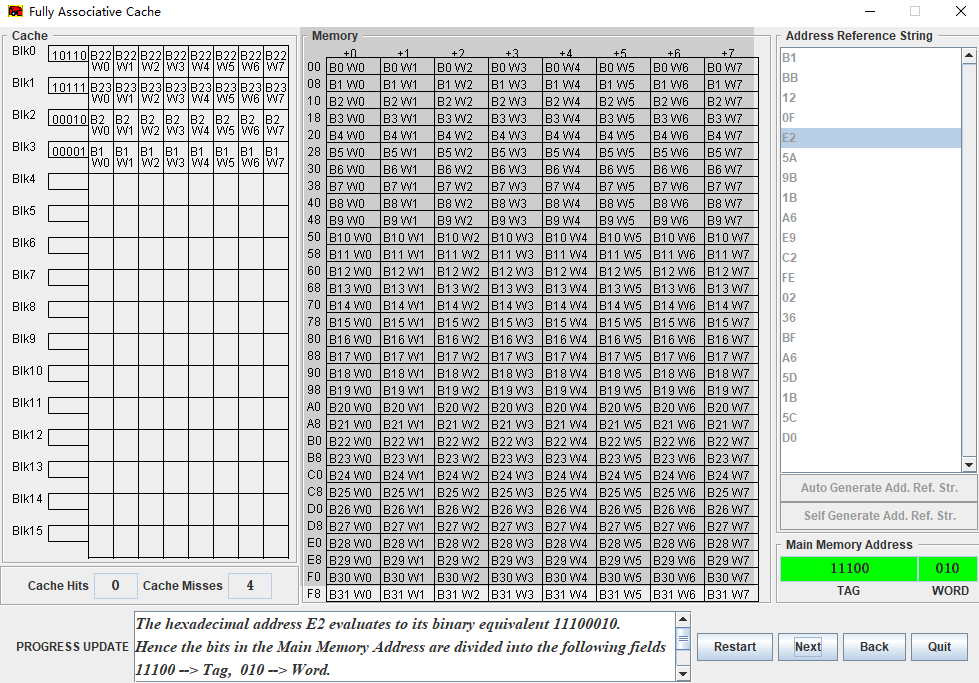
直接相联映射：

自动生成并仿真的图如下。块大小8个字，所以块内偏移为3位。缓存一共16个块，内存大小32个块。在直接相联映射中，内存被分为2个区，所以区号(TAG)占1位，Index占4位



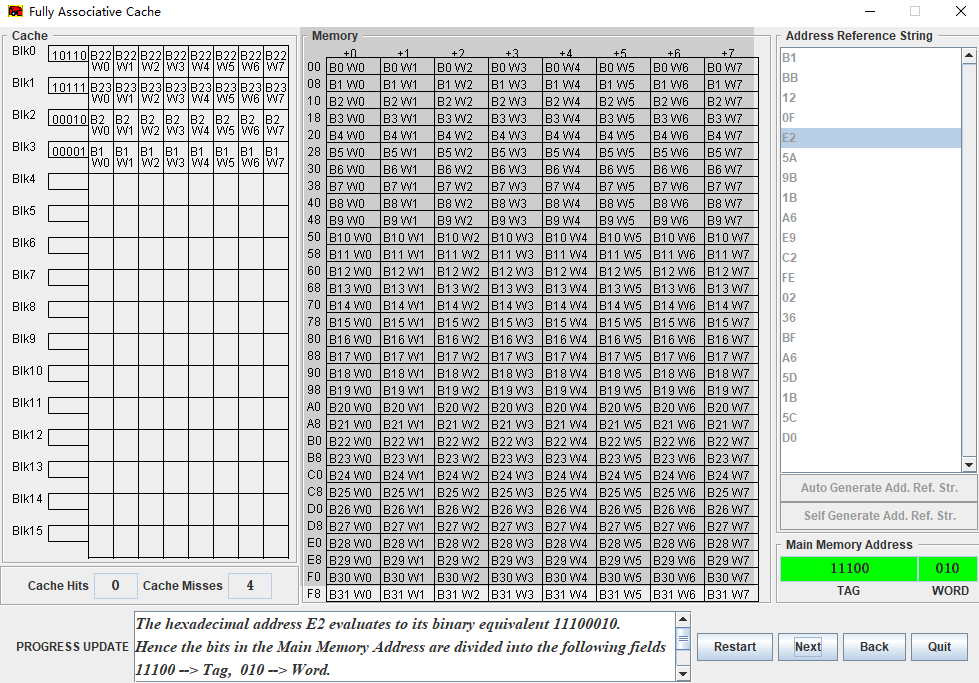
全相联映射：

自动生成地址并仿真结果如下。因为内存32个块，所以TAG占5位。因为块8字所以3位。



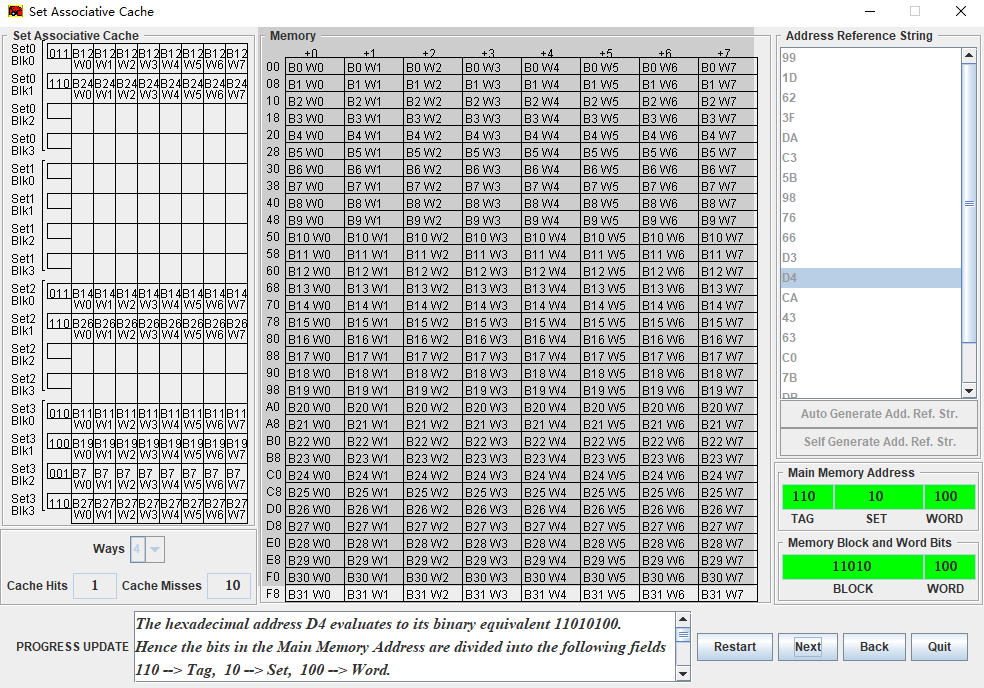
2路组相联映射：

自动生成地址并仿真结果如下。因为2路组相联，所以缓存以两块为1组，分为8组，所以set占3位。同时因为主存32块，缓存8组，所以主存被分为4组，占2位，所以TAG值2位。因为块8字，所以WORD占3位。



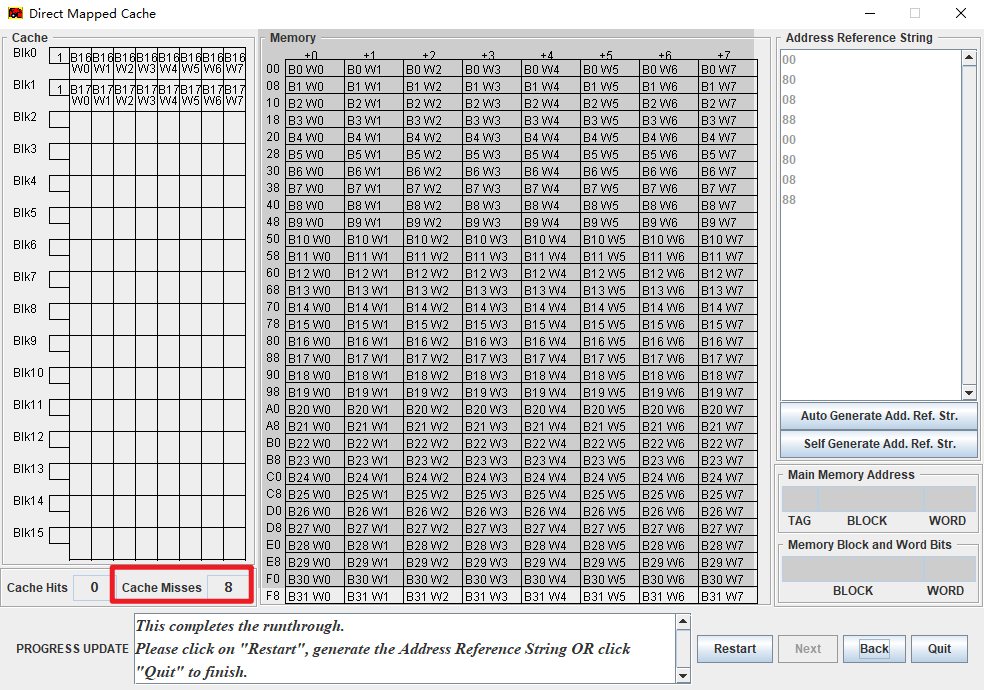
4路组相联映射：

自动生成地址并仿真结果如下。因为4路组相联，所以缓存以4块为1组，分为4组，所以set占2位。同时因为主存32块，缓存4组，所以主存被分为8组，占3位，所以TAG值3位。因为块8字，所以WORD占3位。



1. 实验2

想要达到这种效果很简单，根据直接相联和全相联映射的原理设计即可，直接相联会映射到特定的行，全相连是映射到任意一行（默认从第一行依次往后使用），那么只要设计的访问序列不超过cache的总行数，那么访问序列循环时，全相联自然会全部命中；现在只需要考虑让直接相联全部缺失就能达到题目要求，题目中的主存有32行，cache只有16行，自然主存中有2行是映射到cache中的同一行，那么只要将映射到同一行的两行的地址放入访问序列，自然就能做到使直接相联全部缺失。

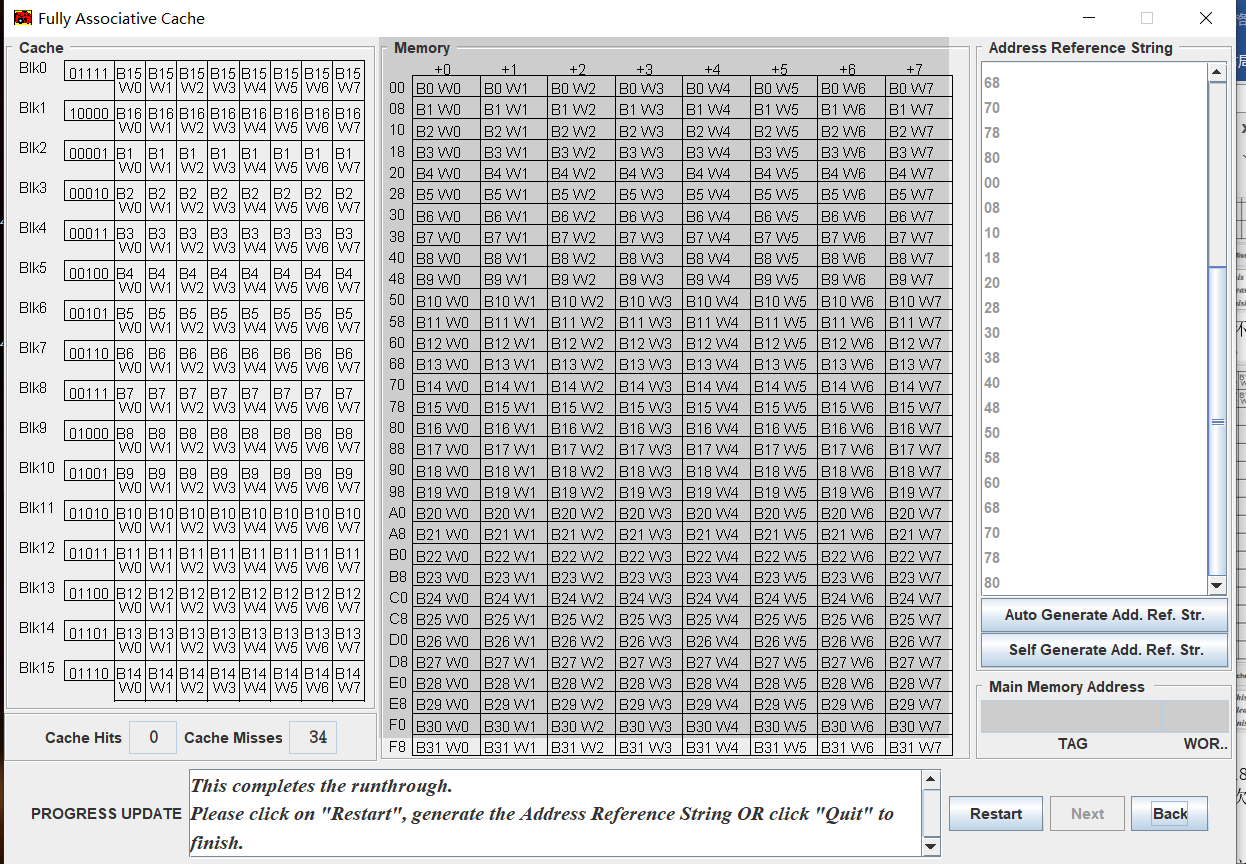


1. 实验3

只需要设计一个序列是的它超过cache的行数一行，这样在循环访问时，序列的最后一行会缺失将第一个添加进cache的块替换掉，但接着循环到开始的第一行时，会再次缺失，因为上一次循环的最后一块将其在cache中的块替换掉了，那么根据策略cache会再次替换掉一个块，这样就能造成全相联全部缺失。而直接相联序列中只有一对访问地址会映射到同一行中导致缺失，其他的块会常驻cache中，因此缺失率不高。

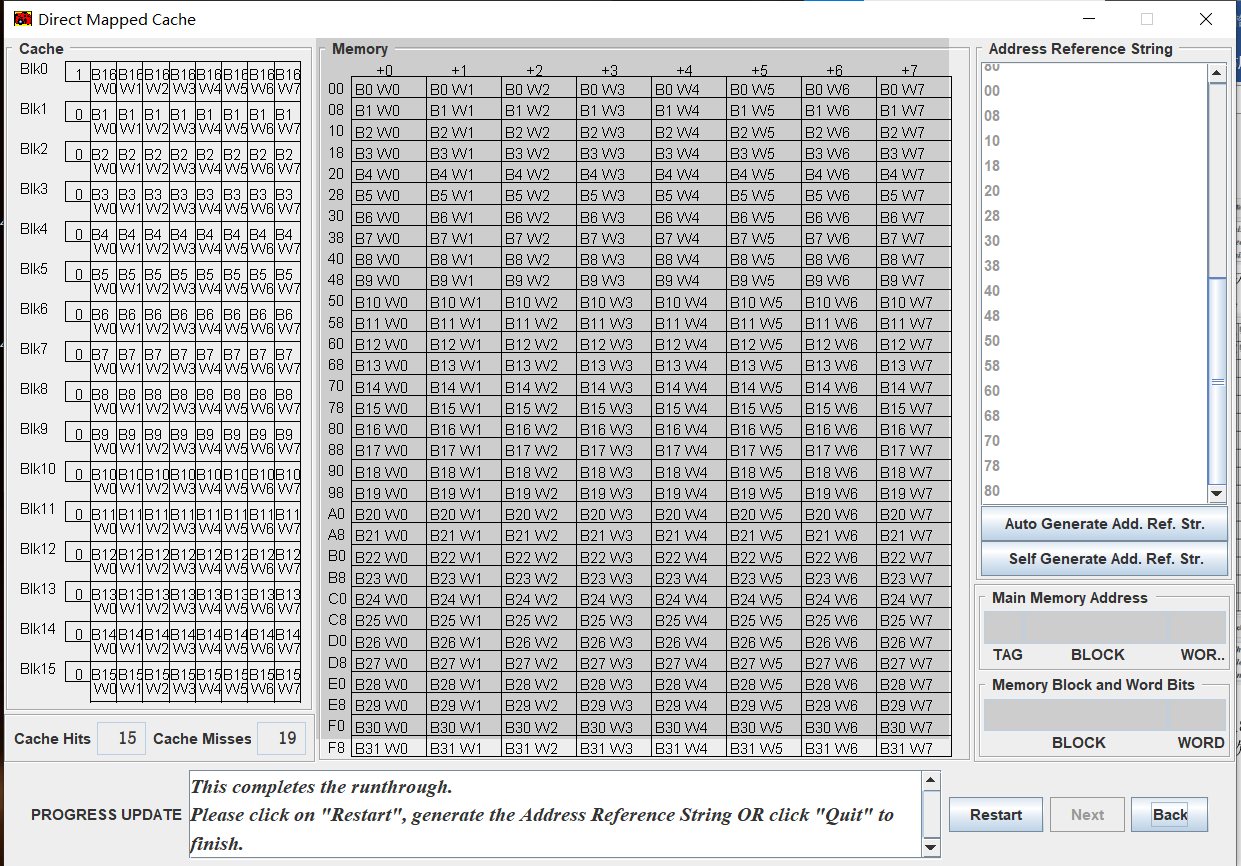
选择00 08 10 18 20 28 30 38 40 48 50 58 60 68 70 78 80

全相联映射以2次循环为例子，结果如下，全部缺失。



直接相联映射，以2次循环为例，结果如下图。

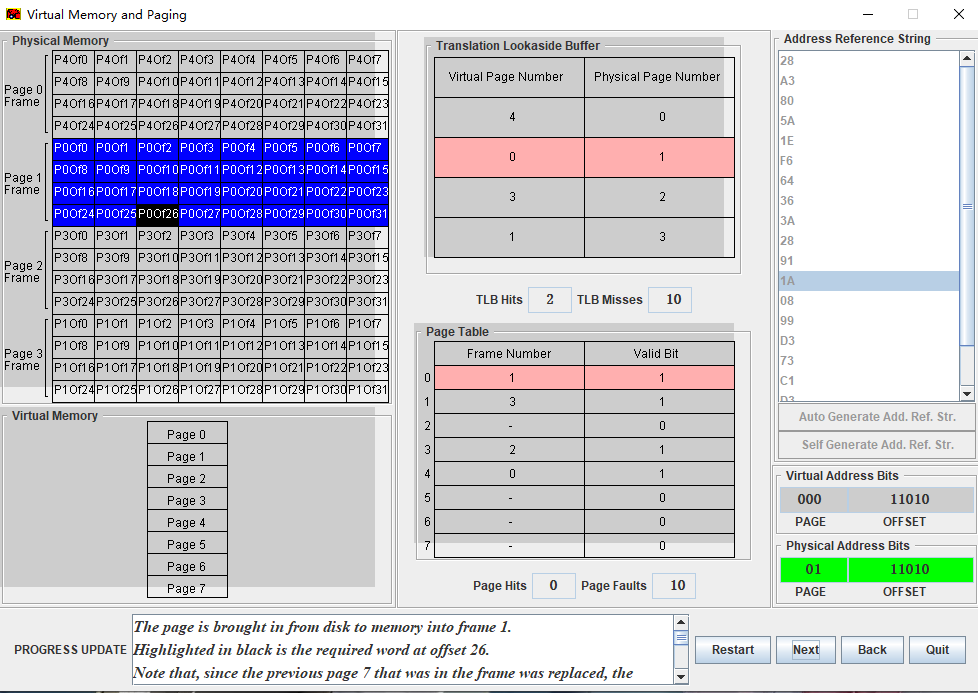
当循环次数足够多，Cache Hits结果趋近于15/17。



1. 实验4

物理页大小32B，字节寻址，虚拟页大小和物理页大小相同，所以虚拟页大小也为32B，即需要5位表示。虚拟系统有8个虚拟页，需要用3位表示，所以虚拟地址一共8位，其中高3位表示虚拟页号，低5位表示偏移地址。

TLB表4行，只能缓存4个物理页，需要通过PT表进行寻找，PT表的Valid Bit表示该页是否在TLB表中，为1则在TLB表中，为0则表示不在。PT表的FN表示的是TLB表的序号，TLB表的VPN表示虚拟页号，PPN表示物理地址页号，和偏移地址共同确定物理内存地址。

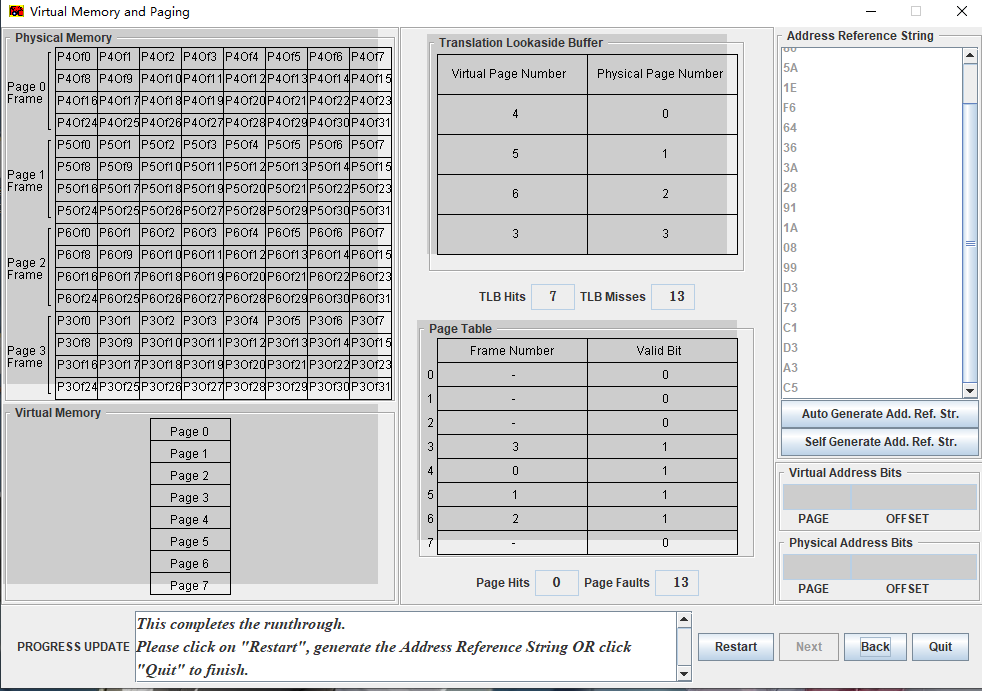


随机生成的访问序列



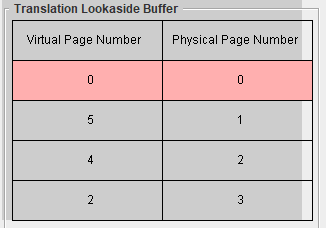
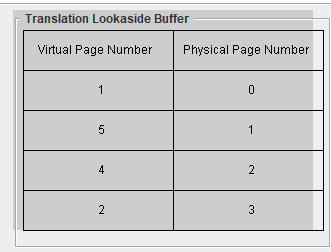
对应虚拟页号为：1、5、4、2、0、7、3、1、1、1、4、0、0、4、6、3、6、6、5、6

命中次数7，缺页次数13

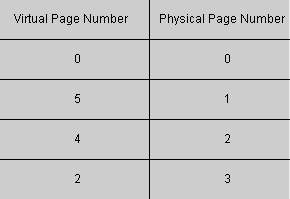
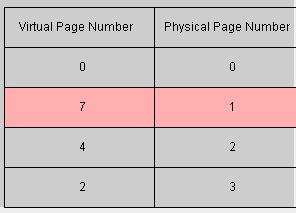


由于访问序列过于巧合，发生13次缺页，去除最初4次缺页，共发生了9次替换，前8次的替换完全符合FIFO算法，本来以为就是采用的FIFO算法，结果第9次替换，替换掉的不是先进的，根据第9次的替换结果分析，再结合前8次替换，最终确定是LFU算法。

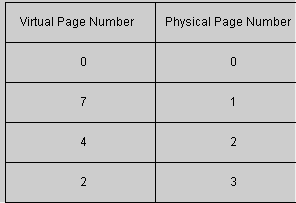
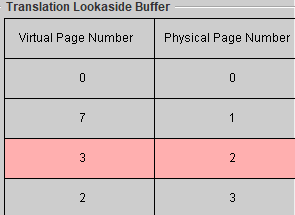
第1次替换前后（1E）



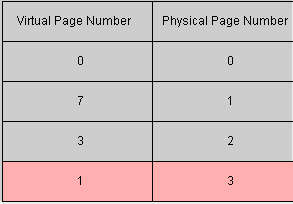
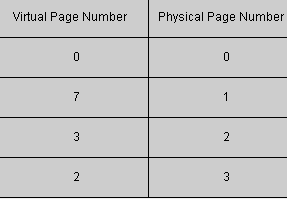
第2次替换前后（F6）

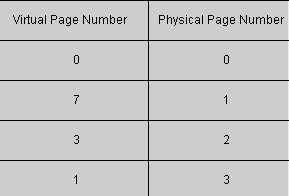
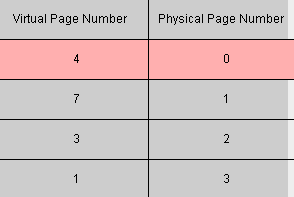
第3次替换前后（64）

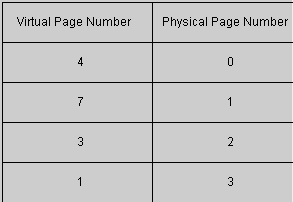
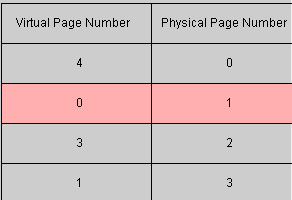
第4次替换前后（36）



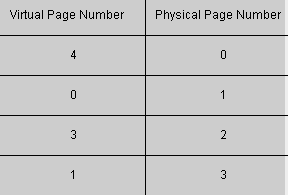
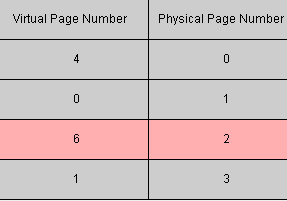
第5次替换前后（91）

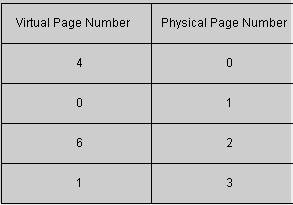
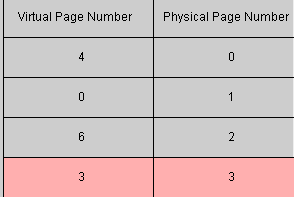
第6次替换前后（1A）

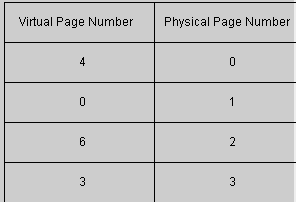
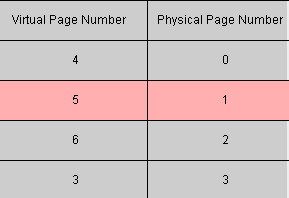
第7次替换前后（D3）

第8次替换前后（73）

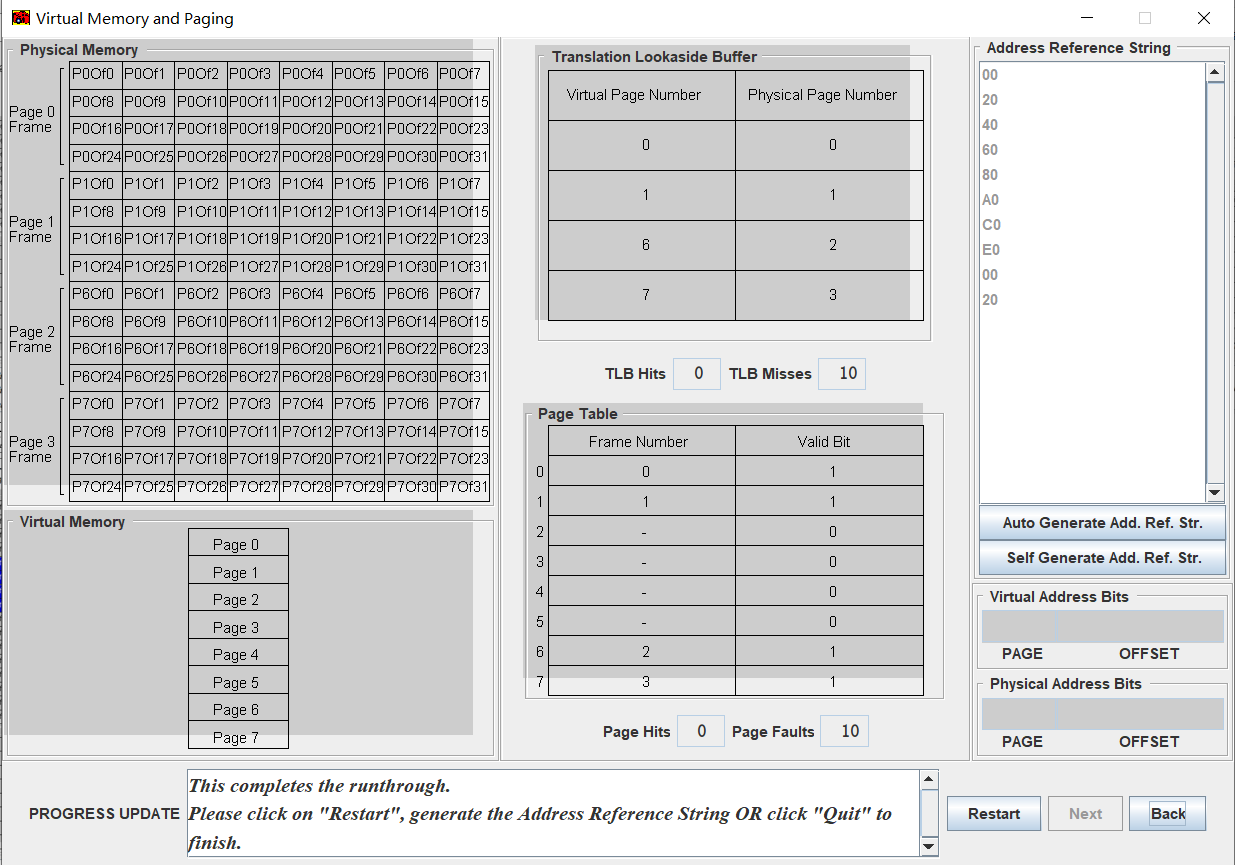
 

第9次替换前后（1A）

1. 实验5

这里我们使用00 20 40 60 80 A0 C0 E0 00 20



1. 实验6

可以通过调整内存大小，将内存大小调至8页。那么此时TLB Misses10,页面缺失为8。

**五、实验思考**