第二十章

作业

这个有趣的小作业会测试你是否了解多级页表的工作原理。是的，前面句子中使用的“有趣”一词有一些争议。该程序叫作“可能不太怪：paging-multilevel-translate.py”。详情请参阅 README 文件。

页大小为 32 字节，虚拟地址空间为 1024 页，物理地址空间为 128 页。

所以虚拟地址需要 15 位，5 位偏移，10 位 VPN；物理地址需要 12 位，5 位偏移，7 位 PFN。

系统假设一个多级页表。页表项和页目录项都是 1 字节的。虚拟地址的前五位用来索引到页目录;页目录条目(PDE)如果有效，则指向页表中的一个页。每个页表页包含 32 个页表项(PTE)。每个 PTE(如果有效)都保存所讨论的虚拟页面的所需转换(PFN)。

The format of a PTE is thus:

```
VALID | PFN6 ... PFN0
```

一开始会得到两条信息。首先，给您页面目录基寄存器(PDBR)的值，它告诉您页面目录位于哪个页面上。其次，您将获得每个内存页的完整转储。

1. 对于线性页表，你需要一个寄存器来定位页表，假设硬件在 TLB 未命中时进行查找。你需要多少个寄存器才能找到两级页表？三级页表呢？

答：

对于二级页表，使用一个寄存器找到页目录的位置，然后从页目录中找到存放页表的位置，从页表中找到页表项。

三级页表与二级页表相同，也只需要一个寄存器找到页目录的位置，再通过一级页目录找到二级页目录的位置，通过二级页目录找到页表的位置，最后找到页表项。

若允许对于一个寄存器内的值进分段读取，则上面为正确答案。

如果寄存器的值只能被整体读取，每个起始地址都需要存储在它自己的寄存器中。这是因为当我们需要从页目录项中读取页表的起始地址时（页目录项的内容），我们需要另一个寄存器来存储这个地址。此时会出现二级页表两个寄存器、三级页表三个寄存器的情况。

2. 使用模拟器对随机种子 0、1 和 2 执行翻译，并使用-c 标志检查你的答案。需要多少内存引用来执行每次查找？

种子 0：

```
page 124:0000000000000000000000000000000000000000000000000000000000000000
page 125:0000000000000000000000000000000000000000000000000000000000000000
page 126:7f7f7f7f7f7f7f7f8ce6cf7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f967f7f7f7f7f
page 127:7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7fdf7f7f7f7f7f7f7f7f7f7f7f7f957f7f

PDBR: 108  (decimal) [This means the page directory is held in this page]

Virtual Address 611c: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 3da8: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 17f5: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 7f6c: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 0bad: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 6d60: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 2a5b: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 4c5e: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 2592: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 3e99: Translates To What Physical Address (And Fetches what Value)? Or Fault?
```

答案：

```
PDBR: 108  (decimal) [This means the page directory is held in this page]

Virtual Address 0x611c:
  --> pde index:0x18 [decimal 24] pde contents:0xa1 (valid 1, pfn 0x21 [decimal 33])
    --> pte index:0x8 [decimal 8] pte contents:0xb5 (valid 1, pfn 0x35 [decimal 53])
      --> Translates to Physical Address 0x6bc --> Value: 0x08
Virtual Address 0x3da8:
  --> pde index:0xf [decimal 15] pde contents:0xd6 (valid 1, pfn 0x56 [decimal 86])
    --> pte index:0xd [decimal 13] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x17f5:
  --> pde index:0x5 [decimal 5] pde contents:0xd4 (valid 1, pfn 0x54 [decimal 84])
    --> pte index:0x1f [decimal 31] pte contents:0xce (valid 1, pfn 0x4e [decimal 78])
      --> Translates to Physical Address 0x9d5 --> Value: 0x1c
Virtual Address 0x7f6c:
  --> pde index:0x1f [decimal 31] pde contents:0xff (valid 1, pfn 0x7f [decimal 127])
    --> pte index:0x1b [decimal 27] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x0bad:
  --> pde index:0x2 [decimal 2] pde contents:0xe0 (valid 1, pfn 0x60 [decimal 96])
    --> pte index:0x1d [decimal 29] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x6d60:
  --> pde index:0x1b [decimal 27] pde contents:0xc2 (valid 1, pfn 0x42 [decimal 66])
    --> pte index:0xb [decimal 11] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x2a5b:
  --> pde index:0xa [decimal 10] pde contents:0xd5 (valid 1, pfn 0x55 [decimal 85])
    --> pte index:0x12 [decimal 18] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x4c5e:
  --> pde index:0x13 [decimal 19] pde contents:0xf8 (valid 1, pfn 0x78 [decimal 120])
    --> pte index:0x2 [decimal 2] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x2592:
  --> pde index:0x9 [decimal 9] pde contents:0x9e (valid 1, pfn 0x1e [decimal 30])
    --> pte index:0xc [decimal 12] pte contents:0xbd (valid 1, pfn 0x3d [decimal 61])
      --> Translates to Physical Address 0x7b2 --> Value: 0x1b
Virtual Address 0x3e99:
  --> pde index:0xf [decimal 15] pde contents:0xd6 (valid 1, pfn 0x56 [decimal 86])
    --> pte index:0x14 [decimal 20] pte contents:0xca (valid 1, pfn 0x4a [decimal 74])
      --> Translates to Physical Address 0x959 --> Value: 0x1e
```

以计算 0x611c 为例：

转化为二进制为 110 0001 0001 1100，前五位为页目录索引，中间五位为页表索引，后五位为页偏移。页目录基地址为 108，页目录索引为 24，一个页目录条目为 1 字节（两个十六进制数），找到内容为 a1 即 1010 0001。有效位为 1，PFN 为 0100001 即 33，页表索引为 01000 即 8，找到内容为 b5 即 1011 0110，有效位为 1，内容为 011 0110。合成物理地址为 0110 1101 1100 即 0x6bc，页 53（不是 54）的 28，因为没有给出具体的物理地址，只给出了页，所以十进制 54 对应的是第 53 页，偏移量 28 没有错误，但要注意从 0 开始数。

```
page 108:83fee0da7fd47febbe9ed5ade4ac90d692d8c1f89fe1ede9a1e8c7c2a9d1dbff
page  33:7f7f7f7f7f7f7f7fb57f9d7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7ff6b17f7f7f7f
page  53:0f0c18090e121c0f081713071c1e191b09161b150e030d121c1d0e1a08181100
```

再以计算 0x3da8 为例：

转化为二进制为 011 1101 1010 1000，前五位为页目录索引，中间五位为页表索引，后五位为页偏移。页目录基地址为 108，页目录索引为 15，一个页目录条目为 1 字节（两个十六进制数），找到内容为 d6 即 1101 0110。有效位为 1，PFN 为 101 0110 即 86，页表索引为 01 101 即 13，找到内容为 7f 即 0111 1111，有效位为 0，不存在。

```
page 108:83fee0da7fd47febbe9ed5ade4ac90d692d8c1f89fe1ede9a1e8c7c2a9d1dbff
page  86:7f7f7f7f7f7f7fc57f7f7f7f7f7f7f7f7f7f7f7fca7f7fee7f7f7f7f7f7f7f7f
```

种子 1：

```
page 122:12181a0602120b16090d19020c0410161e17040d1013151e1d06041e041e0312
page 123:7ff67f7f7f7f7fef7f7fcd7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f
page 124:090f0b10000d0d090c18150f140b060008121b190f1e0e190c171e090513100b
page 125:0000000000000000000000000000000000000000000000000000000000000000
page 126:0000000000000000000000000000000000000000000000000000000000000000
page 127:110311001b0a0b1113120f131a0d0f1900040a061b0403090f191e1a12010113

PDBR: 17  (decimal) [This means the page directory is held in this page]

Virtual Address 6c74: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 6b22: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 03df: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 69dc: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 317a: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 4546: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 2c03: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 7fd7: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 390e: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 748b: Translates To What Physical Address (And Fetches what Value)? Or Fault?
```

答案：

```
PDBR: 17  (decimal) [This means the page directory is held in this page]

Virtual Address 0x6c74:
  --> pde index:0x1b [decimal 27] pde contents:0xa0 (valid 1, pfn 0x20 [decimal 32])
    --> pte index:0x3 [decimal 3] pte contents:0xe1 (valid 1, pfn 0x61 [decimal 97])
      --> Translates to Physical Address 0xc34 --> Value: 0x06
Virtual Address 0x6b22:
  --> pde index:0x1a [decimal 26] pde contents:0xd2 (valid 1, pfn 0x52 [decimal 82])
    --> pte index:0x19 [decimal 25] pte contents:0xc7 (valid 1, pfn 0x47 [decimal 71])
      --> Translates to Physical Address 0x8e2 --> Value: 0x1a
Virtual Address 0x03df:
  --> pde index:0x0 [decimal 0] pde contents:0xda (valid 1, pfn 0x5a [decimal 90])
    --> pte index:0x1e [decimal 30] pte contents:0x85 (valid 1, pfn 0x05 [decimal 5])
      --> Translates to Physical Address 0x0bf --> Value: 0x0f
Virtual Address 0x69dc:
  --> pde index:0x1a [decimal 26] pde contents:0xd2 (valid 1, pfn 0x52 [decimal 82])
    --> pte index:0xe [decimal 14] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x317a:
  --> pde index:0xc [decimal 12] pde contents:0x98 (valid 1, pfn 0x18 [decimal 24])
    --> pte index:0xb [decimal 11] pte contents:0xb5 (valid 1, pfn 0x35 [decimal 53])
      --> Translates to Physical Address 0x6ba --> Value: 0x1e
Virtual Address 0x4546:
  --> pde index:0x11 [decimal 17] pde contents:0xa1 (valid 1, pfn 0x21 [decimal 33])
    --> pte index:0xa [decimal 10] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x2c03:
  --> pde index:0xb [decimal 11] pde contents:0xc4 (valid 1, pfn 0x44 [decimal 68])
    --> pte index:0x0 [decimal 0] pte contents:0xd7 (valid 1, pfn 0x57 [decimal 87])
      --> Translates to Physical Address 0xae3 --> Value: 0x16
Virtual Address 0x7fd7:
  --> pde index:0x1f [decimal 31] pde contents:0x92 (valid 1, pfn 0x12 [decimal 18])
    --> pte index:0x1e [decimal 30] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x390e:
  --> pde index:0xe [decimal 14] pde contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page directory entry not valid)
Virtual Address 0x748b:
  --> pde index:0x1d [decimal 29] pde contents:0x80 (valid 1, pfn 0x00 [decimal 0])
    --> pte index:0x4 [decimal 4] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
```

种子 2：

```
page 124:0000000000000000000000000000000000000000000000000000000000000000
page 125:0000000000000000000000000000000000000000000000000000000000000000
page 126:0000000000000000000000000000000000000000000000000000000000000000
page 127:16181b1c11190f07080800130c1c0118060a091d140e05010b1e0d141b01090c

PDBR: 122  (decimal) [This means the page directory is held in this page]

Virtual Address 7570: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 7268: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 1f9f: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 0325: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 64c4: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 0cdf: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 2906: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 7a36: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 21e1: Translates To What Physical Address (And Fetches what Value)? Or Fault?
Virtual Address 5149: Translates To What Physical Address (And Fetches what Value)? Or Fault?
```

答案：

```
PDBR: 122  (decimal) [This means the page directory is held in this page]
Visual Studio Code x7570:
  --> pde index:0x1d [decimal 29] pde contents:0xb3 (valid 1, pfn 0x33 [decimal 51])
    --> pte index:0xb [decimal 11] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x7268:
  --> pde index:0x1c [decimal 28] pde contents:0xde (valid 1, pfn 0x5e [decimal 94])
    --> pte index:0x13 [decimal 19] pte contents:0xe5 (valid 1, pfn 0x65 [decimal 101])
      --> Translates to Physical Address 0xca8 --> Value: 0x16
Virtual Address 0x1f9f:
  --> pde index:0x7 [decimal 7] pde contents:0xaf (valid 1, pfn 0x2f [decimal 47])
    --> pte index:0x1c [decimal 28] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x0325:
  --> pde index:0x0 [decimal 0] pde contents:0x82 (valid 1, pfn 0x02 [decimal 2])
    --> pte index:0x19 [decimal 25] pte contents:0xdd (valid 1, pfn 0x5d [decimal 93])
      --> Translates to Physical Address 0xba5 --> Value: 0x0b
Virtual Address 0x64c4:
  --> pde index:0x19 [decimal 25] pde contents:0xb8 (valid 1, pfn 0x38 [decimal 56])
    --> pte index:0x6 [decimal 6] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
Virtual Address 0x0cdf:
  --> pde index:0x3 [decimal 3] pde contents:0x9d (valid 1, pfn 0x1d [decimal 29])
    --> pte index:0x6 [decimal 6] pte contents:0x97 (valid 1, pfn 0x17 [decimal 23])
      --> Translates to Physical Address 0x2ff --> Value: 0x00
Virtual Address 0x2906:
  --> pde index:0xa [decimal 10] pde contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page directory entry not valid)
Virtual Address 0x7a36:
  --> pde index:0x1e [decimal 30] pde contents:0x8a (valid 1, pfn 0x0a [decimal 10])
    --> pte index:0x11 [decimal 17] pte contents:0xe6 (valid 1, pfn 0x66 [decimal 102])
      --> Translates to Physical Address 0xcd6 --> Value: 0x09
Virtual Address 0x21e1:
  --> pde index:0x8 [decimal 8] pde contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page directory entry not valid)
Virtual Address 0x5149:
  --> pde index:0x14 [decimal 20] pde contents:0xbb (valid 1, pfn 0x3b [decimal 59])
    --> pte index:0xa [decimal 10] pte contents:0x81 (valid 1, pfn 0x01 [decimal 1])
      --> Translates to Physical Address 0x029 --> Value: 0x1b
```

3. 根据你对缓存内存的工作原理的理解，你认为对页表的内存引用如何在缓存中工作？它们是否会导致大量的缓存命中（并导致快速访问）或者很多未命中（并导致访问缓慢）？

答：

初次访问内存时会产生缓存不命中，这个不命中是必然的，被称为冷不命中。发生不命中时系统将访问页表，找到对应的物理页，并将该映射保存在 cache 中，由于大部分的程序访问具有时间局部性和空间局部性，访问将会在一页或者相邻页进行，接下来的访问可能会有大量的命中，从而达到快速访问的目的。

在一些情况下，比如程序访问的页数大于 cache 中的页数，可能会产生大量的缓存不命中。此外，缓存的存储方式和替换算法也会对命中率造成影响。

第二十二章

作业

这个模拟器 paging-policy.py 允许你使用不同的页替换策略。详情请参阅 README 文件。

默认策略是 FIFO，但其他策略也可用，包括 LRU、MRU、OPT（最佳更换策略，窥视未来，看看什么是最好替换）、UNOPT（这是悲观的替代品）、RAND（进行随机替换）和 CLOCK（执行时钟算法）。

1. 使用以下参数生成随机地址：-s 0 -n 10，-s 1 -n 10 和-s 2 -n 10。将策略从 FIFO 更改为 LRU，并将其更改为 OPT。计算所述地址追踪中的每个访问是否命中或未命中。

同样，以种子 0 为例解释。

种子 0：

FIFO:

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 0 -n 10 -p FIFO -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 8  MISS FirstIn ->          [8] <- Lastin  Replaced:- [Hits:0 Misses:1]
Access: 7  MISS FirstIn ->       [8, 7] <- Lastin  Replaced:- [Hits:0 Misses:2]
Access: 4  MISS FirstIn ->    [8, 7, 4] <- Lastin  Replaced:- [Hits:0 Misses:3]
Access: 2  MISS FirstIn ->    [7, 4, 2] <- Lastin  Replaced:8 [Hits:0 Misses:4]
Access: 5  MISS FirstIn ->    [4, 2, 5] <- Lastin  Replaced:7 [Hits:0 Misses:5]
Access: 4  HIT  FirstIn ->    [4, 2, 5] <- Lastin  Replaced:- [Hits:1 Misses:5]
Access: 7  MISS FirstIn ->    [2, 5, 7] <- Lastin  Replaced:4 [Hits:1 Misses:6]
Access: 3  MISS FirstIn ->    [5, 7, 3] <- Lastin  Replaced:2 [Hits:1 Misses:7]
Access: 4  MISS FirstIn ->    [7, 3, 4] <- Lastin  Replaced:5 [Hits:1 Misses:8]
Access: 5  MISS FirstIn ->    [3, 4, 5] <- Lastin  Replaced:7 [Hits:1 Misses:9]

FINALSTATS hits 1   misses 9   hitrate 10.00
```

一共有 10 个页，cache 的大小为 3，我们使用 FIFO 策略。

访问 8，未命中，加入；访问 7，未命中，加入；访问 4，未命中，加入；访问 2，未命中，将 8 移除 2 加入；访问 5，未命中，将 7 移除 5 加入；访问 4，命中；访问 7，未命中，将 4 移除 7 加入；访问 3，未命中，将 5 移除 3 加入；访问 4，未命中，将 5 移除 4 加入；访问 5，未命中，将 7 移除 5 加入。

先入先出策略，将最早的移除。

访问十次命中 1 次，命中率 10%。

LRU（最少最近使用）：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 0 -n 10 -p LRU -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 8  MISS LRU ->          [8] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 7  MISS LRU ->       [8, 7] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 4  MISS LRU ->    [8, 7, 4] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 2  MISS LRU ->    [7, 4, 2] <- MRU Replaced:8 [Hits:0 Misses:4]
Access: 5  MISS LRU ->    [4, 2, 5] <- MRU Replaced:7 [Hits:0 Misses:5]
Access: 4  HIT  LRU ->    [2, 5, 4] <- MRU Replaced:- [Hits:1 Misses:5]
Access: 7  MISS LRU ->    [5, 4, 7] <- MRU Replaced:2 [Hits:1 Misses:6]
Access: 3  MISS LRU ->    [4, 7, 3] <- MRU Replaced:5 [Hits:1 Misses:7]
Access: 4  HIT  LRU ->    [7, 3, 4] <- MRU Replaced:- [Hits:2 Misses:7]
Access: 5  MISS LRU ->    [3, 4, 5] <- MRU Replaced:7 [Hits:2 Misses:8]

FINALSTATS hits 2   misses 8   hitrate 20.00
```

一共有 10 个页，cache 的大小为 3，我们使用 LRU 策略。

访问 8，未命中，加入；访问 7，未命中，加入；访问 4，未命中，加入；访问 2，未命中，将 8 移除 2 加入；访问 5，未命中，将 7 移除 5 加入；访问 4，命中；访问 7，未命中，将 2 移除 7 加入；访问 3，未命中，将 5 移除 3 加入；访问 4，命中；访问 5，未命中，将 7 移除 5 加入。

最近最少访问策略，将最近未被访问的移除。

访问十次命中 2 次，命中率 20%。

OPT（最佳替换策略）：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 0 -n 10 -p OPT -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy OPT
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

Access: 8  MISS Left  ->          [8] <- Right Replaced:- [Hits:0 Misses:1]
Access: 7  MISS Left  ->       [8, 7] <- Right Replaced:- [Hits:0 Misses:2]
Access: 4  MISS Left  ->    [8, 7, 4] <- Right Replaced:- [Hits:0 Misses:3]
Access: 2  MISS Left  ->    [7, 4, 2] <- Right Replaced:8 [Hits:0 Misses:4]
Access: 5  MISS Left  ->    [7, 4, 5] <- Right Replaced:2 [Hits:0 Misses:5]
Access: 4  HIT  Left  ->    [7, 4, 5] <- Right Replaced:- [Hits:1 Misses:5]
Access: 7  HIT  Left  ->    [7, 4, 5] <- Right Replaced:- [Hits:2 Misses:5]
Access: 3  MISS Left  ->    [4, 5, 3] <- Right Replaced:7 [Hits:2 Misses:6]
Access: 4  HIT  Left  ->    [4, 5, 3] <- Right Replaced:- [Hits:3 Misses:6]
Access: 5  HIT  Left  ->    [4, 5, 3] <- Right Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

一共有 10 个页，cache 的大小为 3，我们使用 OPT 策略（查看最佳方式）。

访问 8，未命中，加入；访问 7，未命中，加入；访问 4，未命中，加入；访问 2，未命中，将 8 移除 2 加入；访问 5，未命中，将 2 移除 5 加入；访问 4，命中；访问 7，命中；访问 3，未命中，将 7 移除 3 加入；访问 4，命中；访问 5，命中。

最优解，是做不到的，只是为了查看最佳方案。

访问十次命中 4 次，命中率 40%。该命中率为最高命中率。

种子 1：

FIFO:

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 1 -n 10 -p FIFO -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 1
ARG notrace False

Solving...

Access: 1  MISS FirstIn ->          [1] <- Lastin  Replaced:- [Hits:0 Misses:1]
Access: 8  MISS FirstIn ->       [1, 8] <- Lastin  Replaced:- [Hits:0 Misses:2]
Access: 7  MISS FirstIn ->    [1, 8, 7] <- Lastin  Replaced:- [Hits:0 Misses:3]
Access: 2  MISS FirstIn ->    [8, 7, 2] <- Lastin  Replaced:1 [Hits:0 Misses:4]
Access: 4  MISS FirstIn ->    [7, 2, 4] <- Lastin  Replaced:8 [Hits:0 Misses:5]
Access: 4  HIT  FirstIn ->    [7, 2, 4] <- Lastin  Replaced:- [Hits:1 Misses:5]
Access: 6  MISS FirstIn ->    [2, 4, 6] <- Lastin  Replaced:7 [Hits:1 Misses:6]
Access: 7  MISS FirstIn ->    [4, 6, 7] <- Lastin  Replaced:2 [Hits:1 Misses:7]
Access: 0  MISS FirstIn ->    [6, 7, 0] <- Lastin  Replaced:4 [Hits:1 Misses:8]
Access: 0  HIT  FirstIn ->    [6, 7, 0] <- Lastin  Replaced:- [Hits:2 Misses:8]

FINALSTATS hits 2   misses 8   hitrate 20.00
```

LRU：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 1 -n 10 -p LRU -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 1
ARG notrace False

Solving...

Access: 1  MISS LRU ->          [1] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 8  MISS LRU ->       [1, 8] <- MRU Replaced:- [Hits:0 Misses:2]
Access: 7  MISS LRU ->    [1, 8, 7] <- MRU Replaced:- [Hits:0 Misses:3]
Access: 2  MISS LRU ->    [8, 7, 2] <- MRU Replaced:1 [Hits:0 Misses:4]
Access: 4  MISS LRU ->    [7, 2, 4] <- MRU Replaced:8 [Hits:0 Misses:5]
Access: 4  HIT  LRU ->    [7, 2, 4] <- MRU Replaced:- [Hits:1 Misses:5]
Access: 6  MISS LRU ->    [2, 4, 6] <- MRU Replaced:7 [Hits:1 Misses:6]
Access: 7  MISS LRU ->    [4, 6, 7] <- MRU Replaced:2 [Hits:1 Misses:7]
Access: 0  MISS LRU ->    [6, 7, 0] <- MRU Replaced:4 [Hits:1 Misses:8]
Access: 0  HIT  LRU ->    [6, 7, 0] <- MRU Replaced:- [Hits:2 Misses:8]

FINALSTATS hits 2   misses 8   hitrate 20.00
```

OPT：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 1 -n 10 -p OPT -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy OPT
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 1
ARG notrace False

Solving...

Access: 1  MISS Left  ->          [1] <- Right Replaced:- [Hits:0 Misses:1]
Access: 8  MISS Left  ->       [1, 8] <- Right Replaced:- [Hits:0 Misses:2]
Access: 7  MISS Left  ->    [1, 8, 7] <- Right Replaced:- [Hits:0 Misses:3]
Access: 2  MISS Left  ->    [1, 7, 2] <- Right Replaced:8 [Hits:0 Misses:4]
Access: 4  MISS Left  ->    [1, 7, 4] <- Right Replaced:2 [Hits:0 Misses:5]
Access: 4  HIT  Left  ->    [1, 7, 4] <- Right Replaced:- [Hits:1 Misses:5]
Access: 6  MISS Left  ->    [1, 7, 6] <- Right Replaced:4 [Hits:1 Misses:6]
Access: 7  HIT  Left  ->    [1, 7, 6] <- Right Replaced:- [Hits:2 Misses:6]
Access: 0  MISS Left  ->    [1, 7, 0] <- Right Replaced:6 [Hits:2 Misses:7]
Access: 0  HIT  Left  ->    [1, 7, 0] <- Right Replaced:- [Hits:3 Misses:7]

FINALSTATS hits 3   misses 7   hitrate 30.00
```

种子 2：

FIFO：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 2 -n 10 -p FIFO -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy FIFO
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 2
ARG notrace False

Solving...

Access: 9  MISS FirstIn ->          [9] <- Lastin  Replaced:- [Hits:0 Misses:1]
Access: 9  HIT  FirstIn ->          [9] <- Lastin  Replaced:- [Hits:1 Misses:1]
Access: 0  MISS FirstIn ->       [9, 0] <- Lastin  Replaced:- [Hits:1 Misses:2]
Access: 0  HIT  FirstIn ->       [9, 0] <- Lastin  Replaced:- [Hits:2 Misses:2]
Access: 8  MISS FirstIn ->    [9, 0, 8] <- Lastin  Replaced:- [Hits:2 Misses:3]
Access: 7  MISS FirstIn ->    [0, 8, 7] <- Lastin  Replaced:9 [Hits:2 Misses:4]
Access: 6  MISS FirstIn ->    [8, 7, 6] <- Lastin  Replaced:0 [Hits:2 Misses:5]
Access: 3  MISS FirstIn ->    [7, 6, 3] <- Lastin  Replaced:8 [Hits:2 Misses:6]
Access: 6  HIT  FirstIn ->    [7, 6, 3] <- Lastin  Replaced:- [Hits:3 Misses:6]
Access: 6  HIT  FirstIn ->    [7, 6, 3] <- Lastin  Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

LRU：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 2 -n 10 -p LRU -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 2
ARG notrace False

Solving...

Access: 9  MISS LRU ->          [9] <- MRU Replaced:- [Hits:0 Misses:1]
Access: 9  HIT  LRU ->          [9] <- MRU Replaced:- [Hits:1 Misses:1]
Access: 0  MISS LRU ->       [9, 0] <- MRU Replaced:- [Hits:1 Misses:2]
Access: 0  HIT  LRU ->       [9, 0] <- MRU Replaced:- [Hits:2 Misses:2]
Access: 8  MISS LRU ->    [9, 0, 8] <- MRU Replaced:- [Hits:2 Misses:3]
Access: 7  MISS LRU ->    [0, 8, 7] <- MRU Replaced:9 [Hits:2 Misses:4]
Access: 6  MISS LRU ->    [8, 7, 6] <- MRU Replaced:0 [Hits:2 Misses:5]
Access: 3  MISS LRU ->    [7, 6, 3] <- MRU Replaced:8 [Hits:2 Misses:6]
Access: 6  HIT  LRU ->    [7, 3, 6] <- MRU Replaced:- [Hits:3 Misses:6]
Access: 6  HIT  LRU ->    [7, 3, 6] <- MRU Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

OPT：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 2 -n 10 -p OPT -c
ARG addresses -1
ARG addressfile
ARG numaddrs 10
ARG policy OPT
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 2
ARG notrace False

Solving...

Access: 9  MISS Left  ->          [9] <- Right Replaced:- [Hits:0 Misses:1]
Access: 9  HIT  Left  ->          [9] <- Right Replaced:- [Hits:1 Misses:1]
Access: 0  MISS Left  ->       [9, 0] <- Right Replaced:- [Hits:1 Misses:2]
Access: 0  HIT  Left  ->       [9, 0] <- Right Replaced:- [Hits:2 Misses:2]
Access: 8  MISS Left  ->    [9, 0, 8] <- Right Replaced:- [Hits:2 Misses:3]
Access: 7  MISS Left  ->    [9, 0, 7] <- Right Replaced:8 [Hits:2 Misses:4]
Access: 6  MISS Left  ->    [9, 0, 6] <- Right Replaced:7 [Hits:2 Misses:5]
Access: 3  MISS Left  ->    [9, 6, 3] <- Right Replaced:0 [Hits:2 Misses:6]
Access: 6  HIT  Left  ->    [9, 6, 3] <- Right Replaced:- [Hits:3 Misses:6]
Access: 6  HIT  Left  ->    [9, 6, 3] <- Right Replaced:- [Hits:4 Misses:6]

FINALSTATS hits 4   misses 6   hitrate 40.00
```

2．对于大小为 5 的高速缓存，为以下每个策略生成最差情况的地址引用序列：FIFO、LRU 和 MRU（最差情况下的引用序列导致尽可能多的未命中）。对于最差情况下的引用序列，需要的缓存增大多少，才能大幅提高性能，并接近 OPT？

答：

FIFO：1,2,3,4,5,6,1,2,3,4,5,6……

LRU：1,2,3,4,5,6,1,2,3,4,5,6……

MRU:1,2,3,4,5,6,5,6,5,6,5……

对于 FIFO 和 LRU,使得高速缓存大小与引用序列大小相同就可大幅提高命中率。

对于 MRU，如果缓存已满，两页交替访问，将缓存增加 1 就可提高命中率，如果仍有其他页访问，缓存大小与序列大小相同同样可提高。

3．生成一个随机追踪序列（使用 Python 或 Perl）。你预计不同的策略在这样的追踪序列上的表现如何？

```python
import random
num=300
sequence=[]
for i in range(0,num):
    address=random.randint(0,9)
    sequence.append(address)
file=open("./sequence.txt","w")
for i in sequence:
    file.write(str(i)+",")
file.close()
```

使用 python 编写的随机序列生成器，生成的序列如下：

```
9,3,1,3,4,2,5,7,9,5,8,1,5,7,6,2,3,4,8,2,4,9,1,5,3,4,2,5,6,9,0,7,6,5,5,6,8,3,0,0,4,1,9,2,3,3,9,1
,6,1,3,2,9,6,3,5,5,1,4,3,9,7,3,0,4,0,1,4,8,5,1,7,3,0,8,7,0,5,8,5,1,8,3,1,3,3,5,9,5,3,6,1,0,8,7,
6,7,7,7,0,7,4,4,0,7,1,3,2,6,2,0,2,7,6,7,3,4,8,1,9,5,8,4,7,5,1,4,8,4,8,3,3,4,2,2,1,6,2,7,0,5,5,5
,4,2,5,2,5,4,7,0,2,9,5,5,8,2,4,9,1,4,2,5,0,1,9,0,0,1,3,1,2,9,6,7,7,6,0,8,2,9,5,1,3,0,2,6,2,0,4,
2,3,0,3,1,4,7,0,5,0,9,1,9,9,0,9,6,3,4,7,4,6,6,9,8,4,7,5,2,6,1,1,8,5,1,5,8,0,2,0,8,0,4,1,1,2,7,2
,7,9,5,3,1,8,9,3,3,5,0,7,6,0,1,5,3,7,3,1,3,7,6,2,1,9,3,1,8,8,5,6,1,5,1,1,9,9,0,6,7,7,4,8,4,9,8,
7,3,1,2,8,6,6,4,3,2,5,2,0,6,7,
```

由书中无局部性工作负载图可知，OPT 明显好于其他方法，且其他方法表现大致相同。

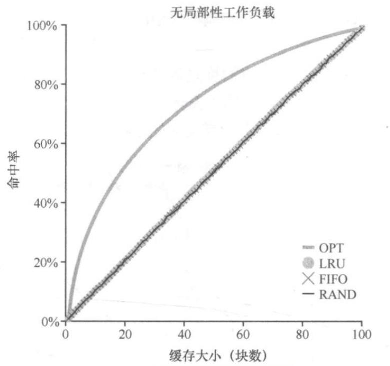

图 22.2 无局部性工作负载

使用程序检查一下。指令举例如下：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -s 2 -n 10 -p OPT -c -a 9,
3,1,3,4,2,5,7,9,5,8,1,5,7,6,2,3,4,8,2,4,9,1,5,3,4,2,5,6,9,0,7,6,5,5,6,8,3,0,0,4,1,9,2,3,3,9,1,6,1,3
,2,9,6,3,5,5,1,4,3,9,7,3,0,4,0,1,4,8,5,1,7,3,0,8,7,0,5,8,5,1,8,3,1,3,3,5,9,5,3,6,1,0,8,7,6,7,7,7,0,
7,4,4,0,7,1,3,2,6,2,0,2,7,6,7,3,4,8,1,9,5,8,4,7,5,1,4,8,4,8,3,3,4,2,2,1,6,2,7,0,5,5,5,4,2,5,2,5,4,7
,0,2,9,5,5,8,2,4,9,1,4,2,5,0,1,9,0,0,1,3,1,2,9,6,7,7,6,0,8,2,9,5,1,3,0,2,6,2,0,4,2,3,0,3,1,4,7,0,5,
0,9,1,9,9,0,9,6,3,4,7,4,6,6,9,8,4,7,5,2,6,1,1,8,5,1,5,8,0,2,0,8,0,4,1,1,2,7,2,7,9,5,3,1,8,9,3,3,5,0
,7,6,0,1,5,3,7,3,1,3,7,6,2,1,9,3,1,8,8,5,6,1,5,1,1,9,9,0,6,7,7,4,8,4,9,8,7,3,1,2,8,6,6,4,3,2,5,2,0,
6,7
```

策略	命中率
OPT	51%
LRU	29.97%
FIFO	30.64%
CLOCK	30.98%
MRU	31.65%
UNOPT	8.75%
RAND	29.63%

4. 现在生成一些局部性追踪序列。如何能够产生这样的追踪序列？LRU 表现如何？RAND 比 LRU 好多少？CLOCK 表现如何？CLOCK 使用不同数量的时钟位，表现如何？

编写程序生成时间局部性和空间局部性序列，-s 生成具有空间局部性的序列，-t 生成具有时间局部性的序列。

```python
#tool.py
import random
import sys

numAddr = 10
# 空间局部性
def generate_spatial_locality_trace():
    trace = [random.randint(0, numAddr)]
    for i in range(1000):
        l = trace[-1]
        rand = [l, (l + 1) % numAddr, (l - 1) % numAddr, random.randint(0, numAddr)]
        trace.append(random.choice(rand))
    # 问题给的paging-policy.py -a参数里，逗号后不能空格,因此拼接再打印
    print(','.join([str(i) for i in trace]))


# 时间局部性
def generate_temporal_locality_trace():
    trace = [random.randint(0, numAddr)]
    for i in range(1000):
        rand = [random.randint(0, numAddr), random.choice(trace)]
        trace.append(random.choice(rand))
    print(','.join([str(i) for i in trace]))


if len(sys.argv) != 1:
    if sys.argv[1] == '-t':
        generate_temporal_locality_trace()
    elif sys.argv[1] == '-s':
        generate_spatial_locality_trace()

#用法如下
#python3 tool.py -s #产生具有空间局部性序列
#python3 tool.py -t #产生具有时间局部性序列
```

指令举例如下：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -c -N -a  $(python3 location.py -s) -p LRU
```

```
ARG addressfile
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 0
ARG notrace True


FINALSTATS hits 572   misses 429   hitrate 57.14
```

	时间局部性命中率	空间局部性命中率
LRU	28.87%	56.44%
RAND	28.47%	48.85%
CLOCK	31.07%	56.04%
OPT	50.55%	65.83%

几种策略都在空间局部性好的序列上表现出了较好的效果，其中 OPT 效果最好，CLOCK 和 LRU 基本相同，RAND 的执行看运气。

对于 CLOCK：

```
guoruiling@guoruiling-virtual-machine:~/os/hw3$ python3 paging-policy.py -c -N -a  8,0,1,0,8,7,7,8,8,7,7,8,7,7,6,5,4,5,4,5,5,4,4,3,3
,3,4,5,4,9,9,9,0,10,9,5,4,4,4,5,5,6,6,4,3,8,10,3,8,3,4,4,3,3,2,0,2,2,7,7,8,7,8,7,7,8,1,2,3,4,5,6,5,6,6,6,3,4,5,1,0,9,4,8,5,5,4,3,6,7
,6,7,8,9,0,0,1,1,2,3,4,4,4,5,6,5,4,3,10,9,0,1,2,9,0,0,0,4,4,4,3,4,10,1,2,2,5,6,9,8,4,5,6,7,8,7,0,9,0,9,0,8,8,5,5,4,5,4,4,4,3,2,1,2,1
,1,1,2,1,0,5,6,5,6,5,7,6,9,5,6,1,1,2,3,3,8,8,8,2,1,0,9,8,8,8,8,9,8,8,7,10,1,2,1,1,2,6,6,6,7,8,9,0,1,2,2,2,5,5,5,5,6,7,8,8,8,7,8,7,7,
8,7,7,6,2,3,3,2,2,2,2,3,2,10,1,2,1,0,1,4,1,2,1,1,0,0,1,2,2,3,3,3,3,1,0,0,9,9,6,6,7,8,9,8,7,6,2,1,0,2,2,7,7,5,6,9,2,2,3,10,1,1,10,3,9
,8,4,5,4,3,3,2,1,0,9,9,8,8,9,8,8,8,8,8,9,8,9,0,9,0,1,0,9,9,0,0,0,9,8,9,8,7,7,7,6,6,6,7,8,5,5,5,0,9,0,0,0,9,0,9,9,0,1,0,7,8,9,9,0,10,
9,0,0,1,0,2,3,3,4,2,3,1,0,1,1,2,3,2,3,2,3,2,1,1,9,9,8,7,8,5,6,6,3,3,2,3,7,8,7,7,3,2,1,7,2,1,2,1,2,1,2,1,1,0,1,1,4,5,8,7,8,0,0,9,0,1,
0,6,6,5,6,6,7,2,1,2,2,1,0,0,1,1,0,0,9,0,0,9,0,0,1,0,9,0,1,0,0,1,7,6,7,7,6,5,6,6,6,6,5,5,5,6,0,1,0,9,9,0,9,8,1,1,9,0,1,7,1,7,6,5,5,4,
5,6,7,8,9,4,4,4,3,4,5,4,3,3,1,0,1,0,1,2,3,2,2,1,0,4,5,6,3,4,9,8,9,0,0,1,1,0,0,9,8,8,5,1,2,9,9,1,1,2,1,0,1,2,0,9,8,9,7,8,7,6,6,7,6,5,
6,0,0,9,10,2,2,2,2,1,0,0,1,8,5,6,7,7,4,5,6,7,7,7,8,8,8,9,0,8,8,3,3,3,4,7,6,7,8,9,1,2,3,3,3,4,5,2,3,4,4,4,4,3,2,3,2,3,3,4,5,5,6,6,5,2
,5,8,0,1,7,7,8,4,4,8,8,1,2,1,2,8,7,7,10,9,8,8,8,7,8,9,1,0,9,9,0,1,2,2,2,2,2,2,7,6,5,4,0,9,9,0,8,9,0,1,8,7,8,7,8,8,9,9,0,0,9,9,5,5,6,
7,6,7,0,9,9,8,7,7,6,6,7,6,7,6,6,7,8,7,10,1,7,8,9,8,0,9,9,8,9,0,9,0,3,4,4,3,4,3,3,2,1,0,1,0,0,7,8,8,9,0,9,2,1,1,2,1,0,9,9,2,10,9,9,0,
0,1,2,3,6,7,6,6,1,0,6,7,6,7,8,9,9,9,0,0,9,8,7,7,7,3,4,3,6,5,10,10,9,0,1,0,2,2,3,4,5,6,6,6,6,10,10,10,10,1,0,9,3,4,5,6,6,7,7,7,6,7,7,
8,7,7,3,2,2,8,8,3,2,10,9,1,2,3,2,3,3,2,1,1,0,0,1,0,0,9,9,8,7,7,6,7,8,9,4,4,3,2,2,3,3,8,8,9,0,1,2,4,5,6,5,0,1,2,3,2,3,4,5,4,3,4,5,4,1
,1,2,2,6,7,7,8,9,9,9,5,5,4,3,4,4,0,1,1,5,6,5,6,7,9,9,5,10,1,2,2,3,4,4,4,8,8,9,9,8,10,10,1,1,7,6,5,5,5,5,6,6,6,6,6,6,5,6,7,8,2,1,0,4,
3,8,9,0,9,0,9,8,9,0,1,1,1,0,9,0,9,9,9,8,7,7,6,6,0,9,0,9,8,7,6,5,4,3,3,4,4,5,6,2,2,1,2,3,3,3,7,8,9,9,8,9,8,8,6,5,4,3,2,2,2,7,6,7 -p C
LOCK -b 1
```

时钟位	时间局部性命中率	空间局部性命中率
1	30.41	51.35%
2	31.82	52.95%
3	30.51	52.05%
4	31.82	53.25%
5	32.33	51.05%
6	31.42	50.75%
7	31.52	51.15%
8	31.62	51.45%
9	31.62	50.55%
10	31.62	50.55%

在一定的范围内，随着时钟位的增加，CLOCK 策略的效果逐步提升。但是超过一定范围效果会减弱直至保持不变，甚至会回落。