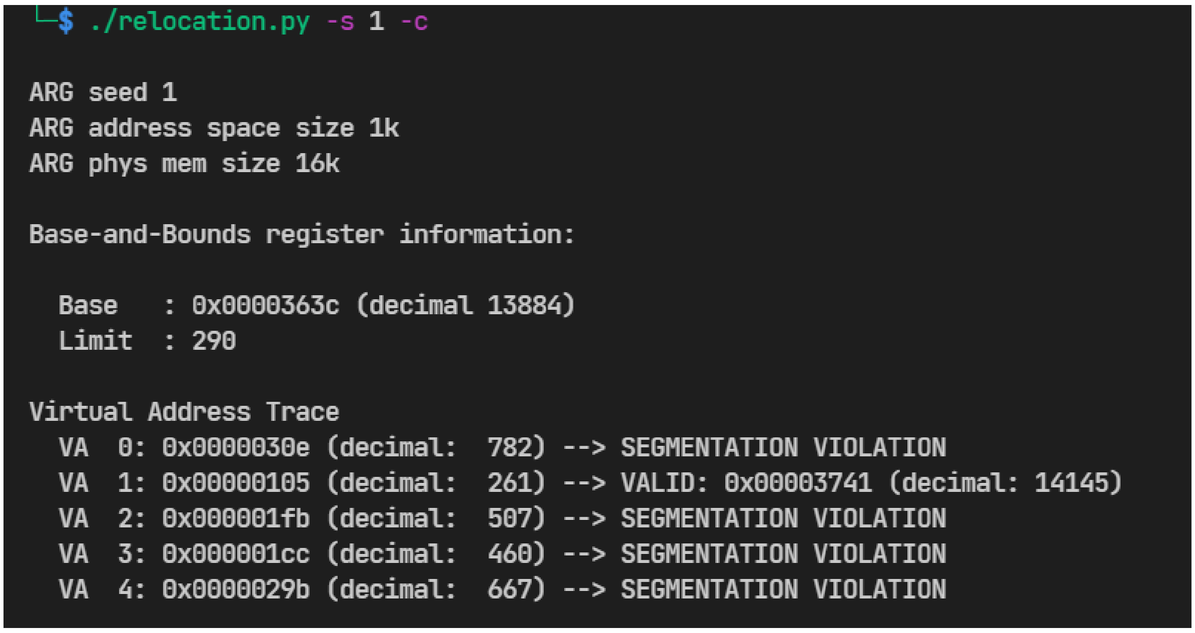
## 15.1

在seed1的条件下

VA1是有效的 转换为实地址为0x3741

其他都是不在界限内的地址



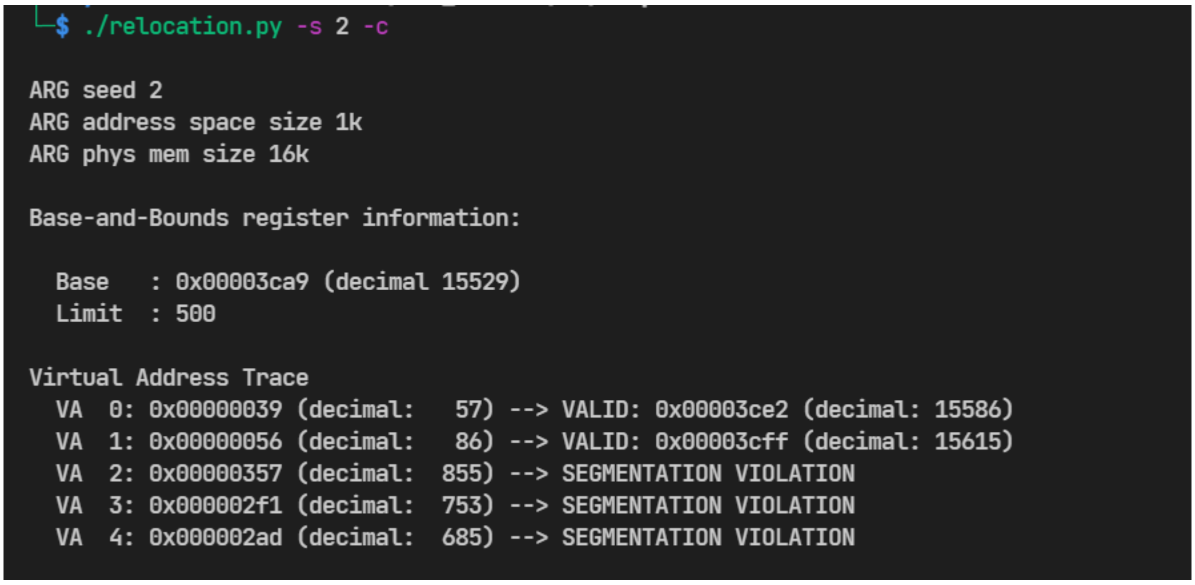
```
└─$ ./relocation.py -s 1 -c

ARG seed 1
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x0000363c (decimal 13884)
  Limit  : 290

Virtual Address Trace
  VA  0: 0x0000030e (decimal:  782) --> SEGMENTATION VIOLATION
  VA  1: 0x00000105 (decimal:  261) --> VALID: 0x00003741 (decimal: 14145)
  VA  2: 0x000001fb (decimal:  507) --> SEGMENTATION VIOLATION
  VA  3: 0x000001cc (decimal:  460) --> SEGMENTATION VIOLATION
  VA  4: 0x0000029b (decimal:  667) --> SEGMENTATION VIOLATION
```

在seed2的条件下

VA0是有效的 转换为实地址为0x3ce2

VA1是有效的 转换为实地址为0x3cff

其他都是不在界限内的地址



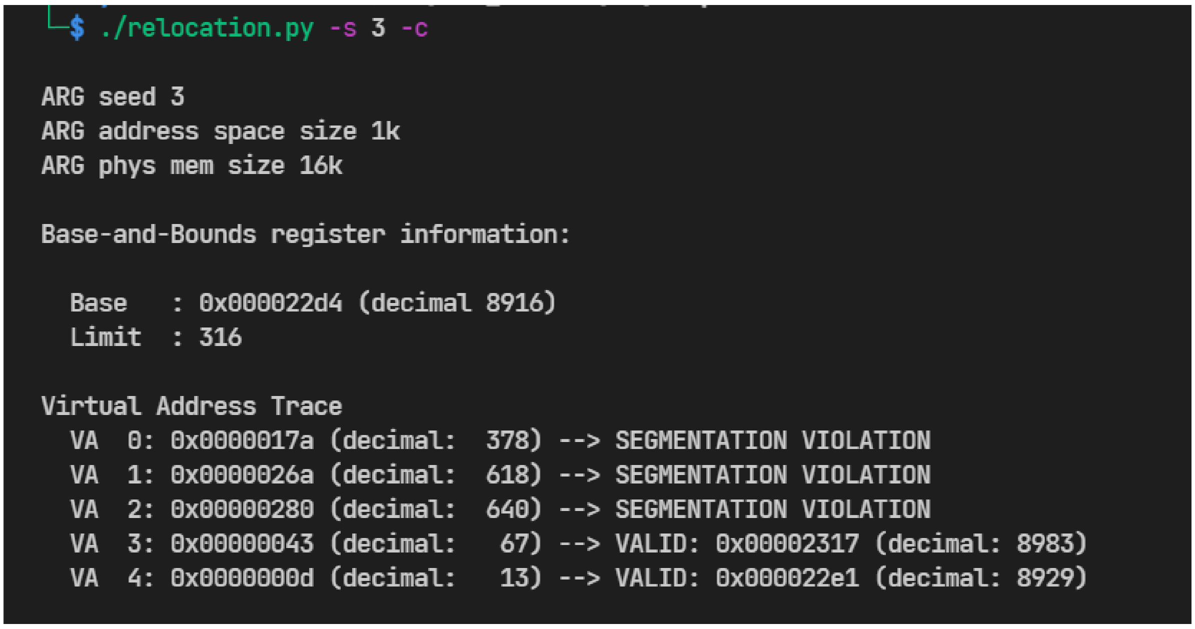
```
└─$ ./relocation.py -s 2 -c

ARG seed 2
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00003ca9 (decimal 15529)
  Limit  : 500

Virtual Address Trace
  VA  0: 0x00000039 (decimal:   57) --> VALID: 0x00003ce2 (decimal: 15586)
  VA  1: 0x00000056 (decimal:   86) --> VALID: 0x00003cff (decimal: 15615)
  VA  2: 0x00000357 (decimal:  855) --> SEGMENTATION VIOLATION
  VA  3: 0x000002f1 (decimal:  753) --> SEGMENTATION VIOLATION
  VA  4: 0x000002ad (decimal:  685) --> SEGMENTATION VIOLATION
```

在seed3的条件下

VA3是有效的 转换为实地址为0x2317

VA4是有效的 转换为实地址为0x22e1

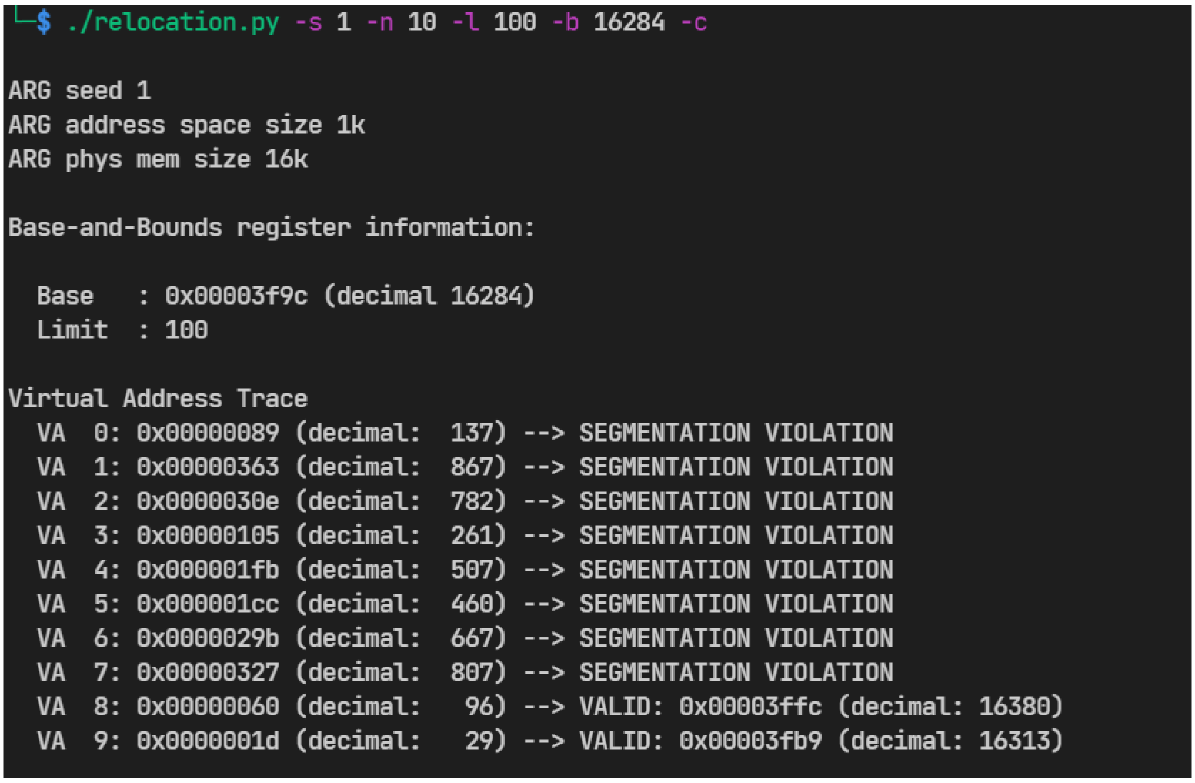
其他都是不在界限内的地址



```
  └─$ ./relocation.py -s 3 -c

  ARG seed 3
  ARG address space size 1k
  ARG phys mem size 16k

  Base-and-Bounds register information:

    Base   : 0x000022d4 (decimal 8916)
    Limit  : 316

  Virtual Address Trace
    VA  0: 0x0000017a (decimal:  378) --> SEGMENTATION VIOLATION
    VA  1: 0x0000026a (decimal:  618) --> SEGMENTATION VIOLATION
    VA  2: 0x00000280 (decimal:  640) --> SEGMENTATION VIOLATION
    VA  3: 0x00000043 (decimal:   67) --> VALID: 0x00002317 (decimal: 8983)
    VA  4: 0x0000000d (decimal:   13) --> VALID: 0x000022e1 (decimal: 8929)
```

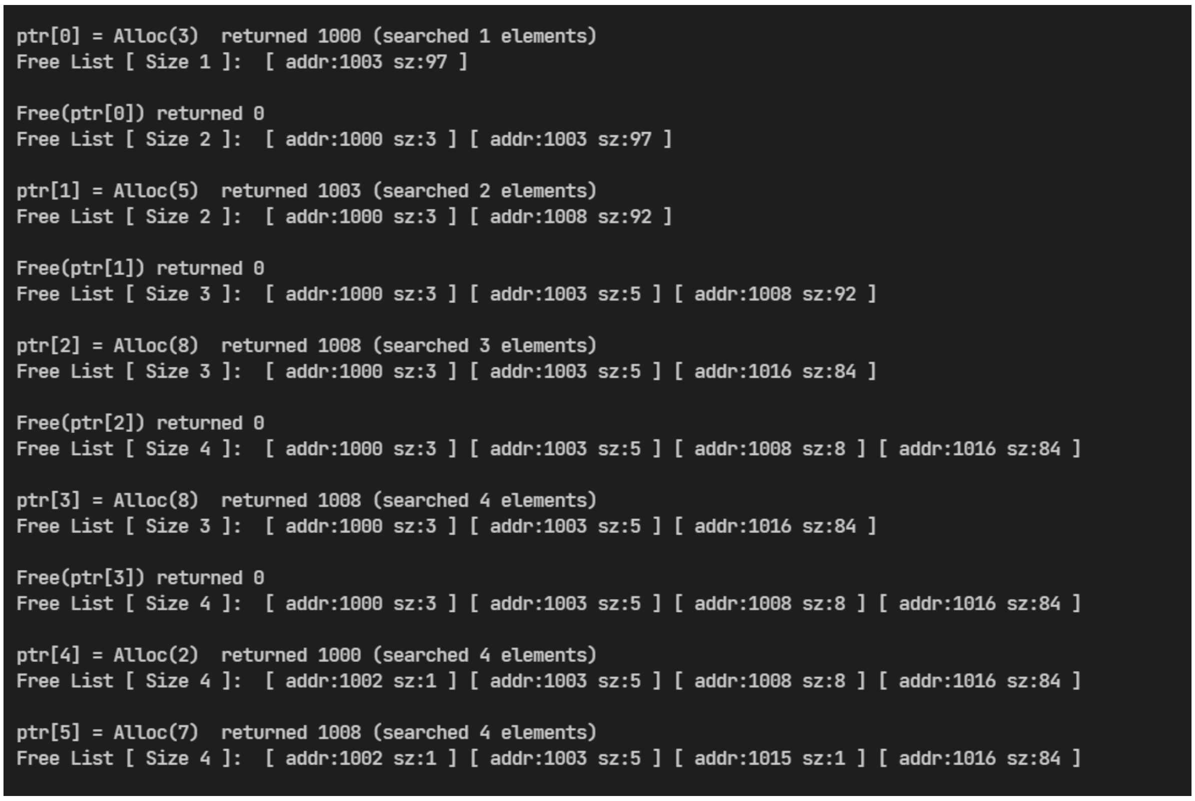
## 15.3

限制长度为100 物理内存大小为16kB 所以基址最大值是 16KB - 100 = 16284B



```
└─$ ./relocation.py -s 1 -n 10 -l 100 -b 16284 -c

ARG seed 1
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00003f9c (decimal 16284)
  Limit  : 100

Virtual Address Trace
  VA  0: 0x00000089 (decimal:  137) --> SEGMENTATION VIOLATION
  VA  1: 0x00000363 (decimal:  867) --> SEGMENTATION VIOLATION
  VA  2: 0x0000030e (decimal:  782) --> SEGMENTATION VIOLATION
  VA  3: 0x00000105 (decimal:  261) --> SEGMENTATION VIOLATION
  VA  4: 0x000001fb (decimal:  507) --> SEGMENTATION VIOLATION
  VA  5: 0x000001cc (decimal:  460) --> SEGMENTATION VIOLATION
  VA  6: 0x0000029b (decimal:  667) --> SEGMENTATION VIOLATION
  VA  7: 0x00000327 (decimal:  807) --> SEGMENTATION VIOLATION
  VA  8: 0x00000060 (decimal:   96) --> VALID: 0x00003ffc (decimal: 16380)
  VA  9: 0x0000001d (decimal:   29) --> VALID: 0x00003fb9 (decimal: 16313)
```

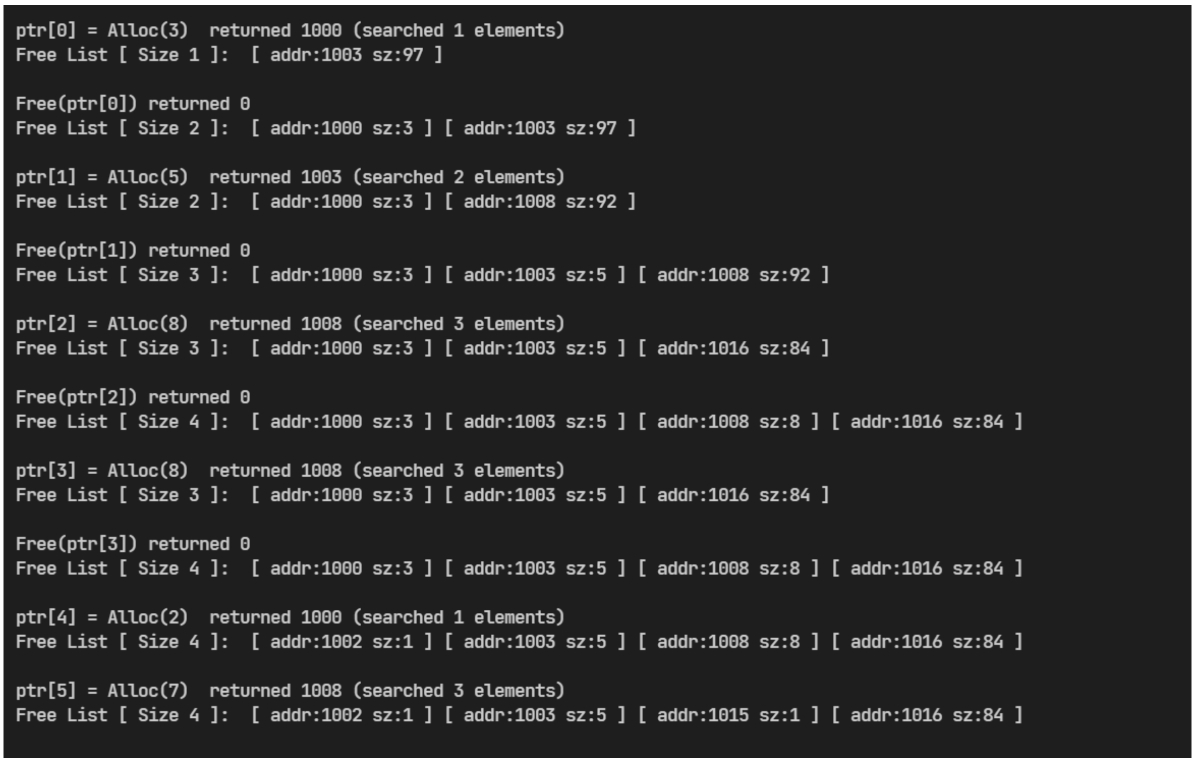
## 17.1


```
ptr[0] = Alloc(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:97 ]

ptr[1] = Alloc(5)  returned 1003 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1008 sz:92 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:92 ]

ptr[2] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [ addr:1016 sz:84 ]

ptr[3] = Alloc(8)  returned 1008 (searched 4 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[3]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [ addr:1016 sz:84 ]

ptr[4] = Alloc(2)  returned 1000 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1002 sz:1 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [ addr:1016 sz:84 ]

ptr[5] = Alloc(7)  returned 1008 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1002 sz:1 ] [ addr:1003 sz:5 ] [ addr:1015 sz:1 ] [ addr:1016 sz:84 ]
```

可以看到 每次释放动态分配的内存都会产生一个新的Free List节点

这些节点越来越多 并且没有合并 最终可能会导致不能分配出特定大小的空闲空间

## 17.3


```
ptr[0] = Alloc(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:97 ]

ptr[1] = Alloc(5)  returned 1003 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1008 sz:92 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:92 ]

ptr[2] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [ addr:1016 sz:84 ]

ptr[3] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[3]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [ addr:1016 sz:84 ]

ptr[4] = Alloc(2)  returned 1000 (searched 1 elements)
Free List [ Size 4 ]:  [ addr:1002 sz:1 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [ addr:1016 sz:84 ]

ptr[5] = Alloc(7)  returned 1008 (searched 3 elements)
Free List [ Size 4 ]:  [ addr:1002 sz:1 ] [ addr:1003 sz:5 ] [ addr:1015 sz:1 ] [ addr:1016 sz:84 ]
```

可以看到 采用FIRST的方式寻找空闲空间

searched elements明显减少 只要找到第一个大于等于需求空间的空闲节点就会立刻分配出去

显然这种分配策略减少了空闲链表的搜索时间

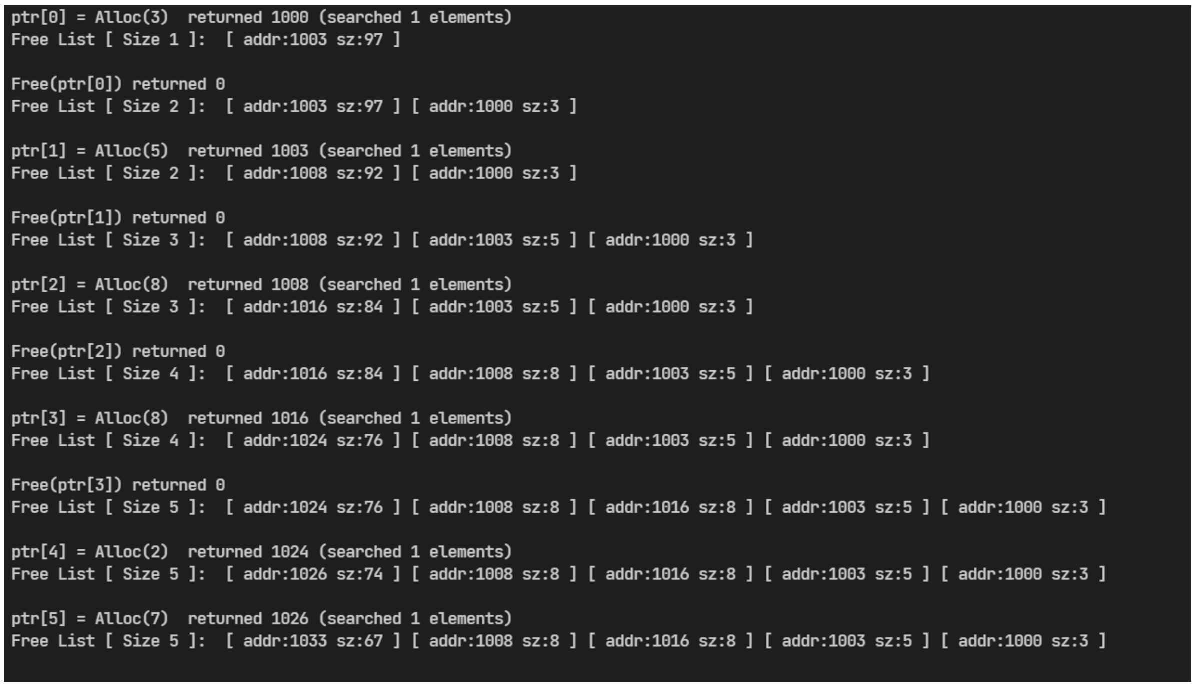
## 17.4

采用ADDRSORT和SIZESORT+方式进行排序:

基本和上面的几个性能差不多

采用SIZESORT-的方式进行排序:

由于空间最大的总是排在第一位 所以每次都只需要搜索一个元素就可以找到满足条件的节点



```
ptr[0] = Alloc(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1003 sz:97 ] [ addr:1000 sz:3 ]

ptr[1] = Alloc(5)  returned 1003 (searched 1 elements)
Free List [ Size 2 ]:  [ addr:1008 sz:92 ] [ addr:1000 sz:3 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1008 sz:92 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[2] = Alloc(8)  returned 1008 (searched 1 elements)
Free List [ Size 3 ]:  [ addr:1016 sz:84 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1016 sz:84 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[3] = Alloc(8)  returned 1016 (searched 1 elements)
Free List [ Size 4 ]:  [ addr:1024 sz:76 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

Free(ptr[3]) returned 0
Free List [ Size 5 ]:  [ addr:1024 sz:76 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[4] = Alloc(2)  returned 1024 (searched 1 elements)
Free List [ Size 5 ]:  [ addr:1026 sz:74 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[5] = Alloc(7)  returned 1026 (searched 1 elements)
Free List [ Size 5 ]:  [ addr:1033 sz:67 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]
```

## 16.1

地址空间大小 128

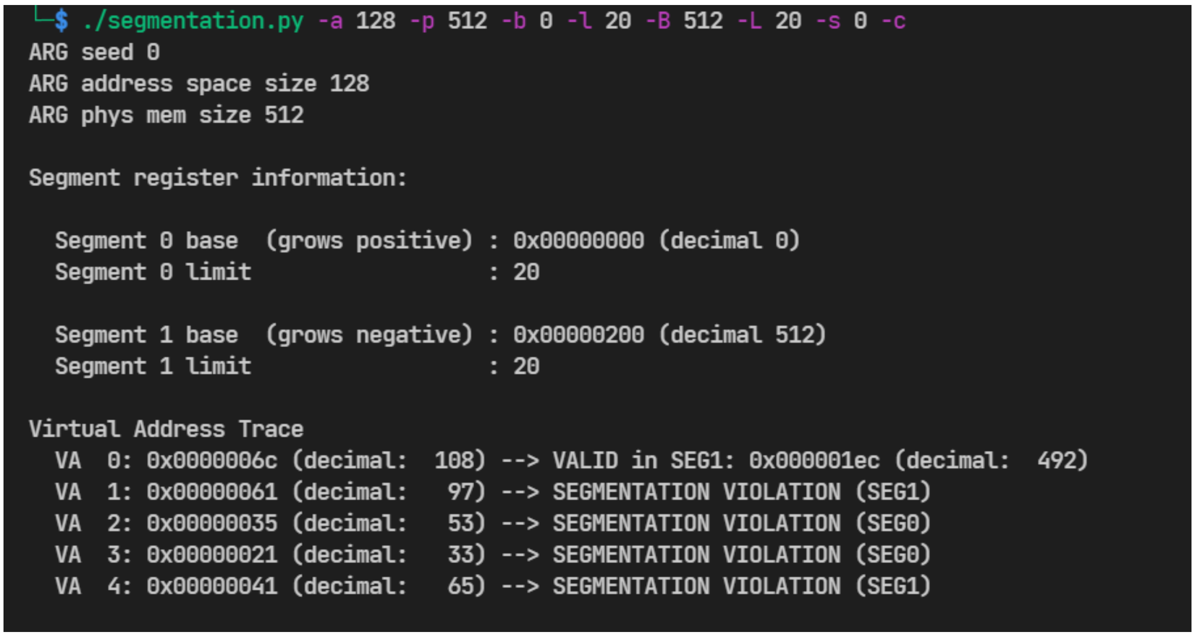
物理内存大小 512

段0基址 0x0 限制长度 20

段1基址 0x200 限制长度 20

VA0 在段1 物理地址为 0x1ec (512-(128-108))

计算可得其他VA都是非法的 不属于任何一个段



```
└─$ ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0 -c
ARG seed 0
ARG address space size 128
ARG phys mem size 512

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 20

  Segment 1 base  (grows negative) : 0x00000200 (decimal 512)
  Segment 1 limit                  : 20

Virtual Address Trace
  VA  0: 0x0000006c (decimal:  108) --> VALID in SEG1: 0x000001ec (decimal:  492)
  VA  1: 0x00000061 (decimal:   97) --> SEGMENTATION VIOLATION (SEG1)
  VA  2: 0x00000035 (decimal:   53) --> SEGMENTATION VIOLATION (SEG0)
  VA  3: 0x00000021 (decimal:   33) --> SEGMENTATION VIOLATION (SEG0)
  VA  4: 0x00000041 (decimal:   65) --> SEGMENTATION VIOLATION (SEG1)
```

地址空间大小 128

物理内存大小 512

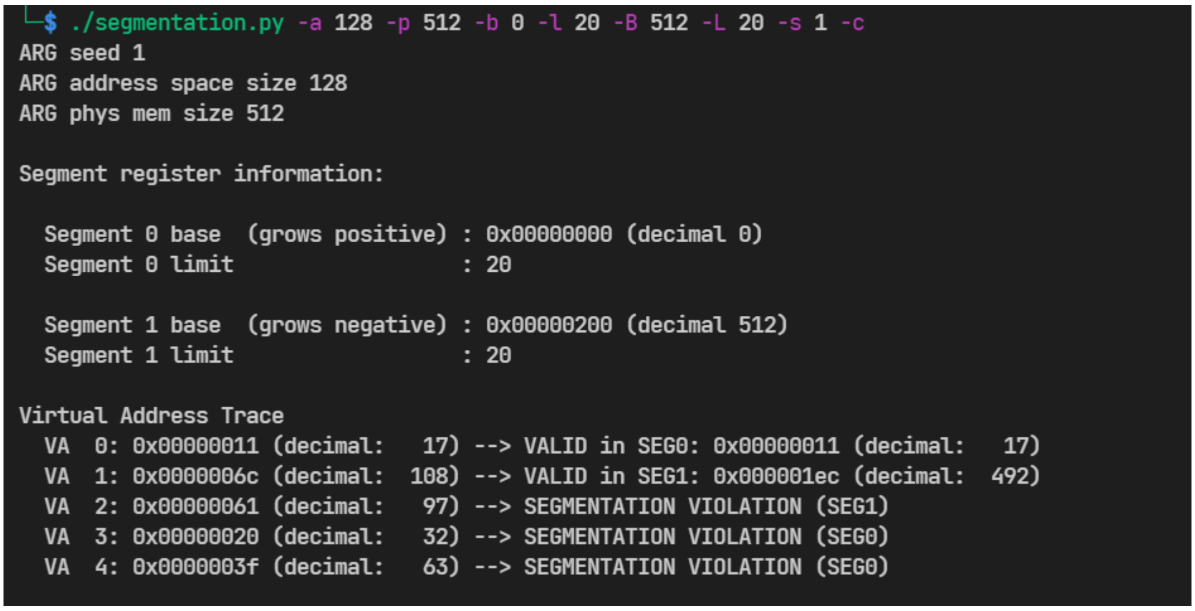
段0基址 0x0 限制长度 20

段1基址 0x200 限制长度 20

VA0 在段0 物理地址为 0x11

VA1 在段1 物理地址为 0x1ec

计算可得其他VA都是非法的 不属于任何一个段



```
└─$ ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 1 -c
ARG seed 1
ARG address space size 128
ARG phys mem size 512

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 20

  Segment 1 base  (grows negative) : 0x00000200 (decimal 512)
  Segment 1 limit                  : 20

Virtual Address Trace
  VA  0: 0x00000011 (decimal:   17) --> VALID in SEG0: 0x00000011 (decimal:   17)
  VA  1: 0x0000006c (decimal:  108) --> VALID in SEG1: 0x000001ec (decimal:  492)
  VA  2: 0x00000061 (decimal:   97) --> SEGMENTATION VIOLATION (SEG1)
  VA  3: 0x00000020 (decimal:   32) --> SEGMENTATION VIOLATION (SEG0)
  VA  4: 0x0000003f (decimal:   63) --> SEGMENTATION VIOLATION (SEG0)
```

地址空间大小 128

物理内存大小 512

段0基址 0x0 限制长度 20

段1基址 0x200 限制长度 20

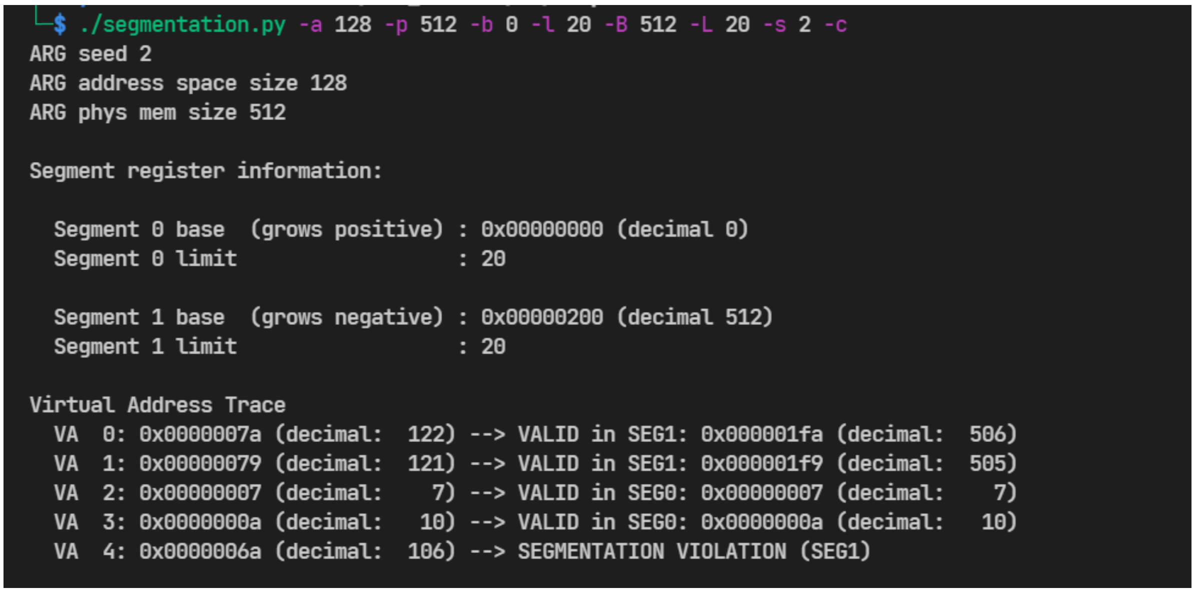
VA0 在段1 物理地址为 0x1fa

VA1 在段1 物理地址为 0x1f9

VA2 在段0 物理地址为 0x7

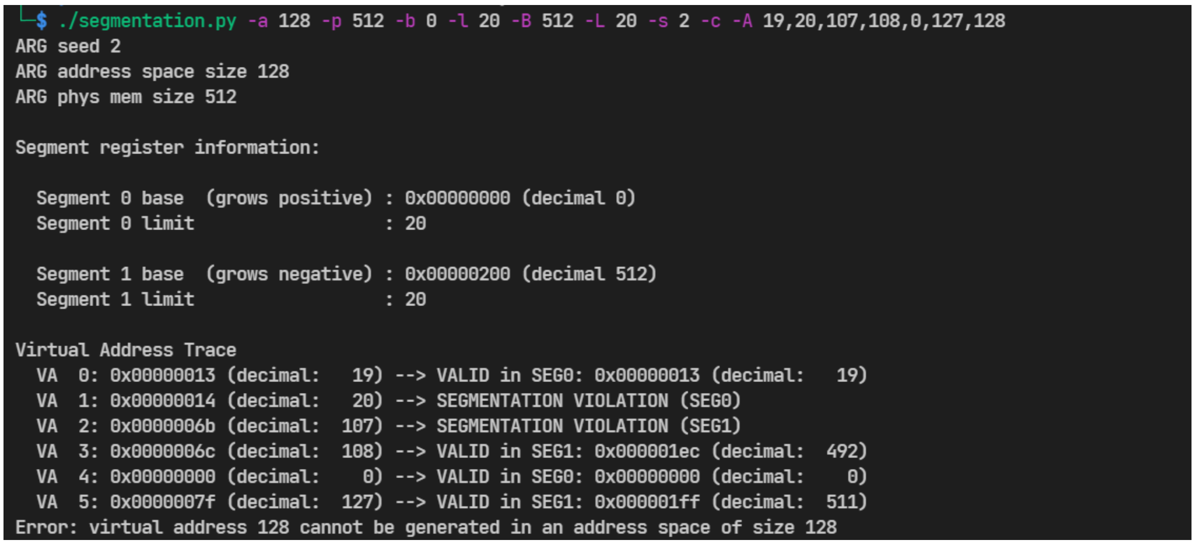
VA3 在段0 物理地址为 0xa

计算可得其他VA都是非法的 不属于任何一个段


```
└─$ ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2 -c
ARG seed 2
ARG address space size 128
ARG phys mem size 512

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 20

  Segment 1 base  (grows negative) : 0x00000200 (decimal 512)
  Segment 1 limit                  : 20

Virtual Address Trace
  VA  0: 0x0000007a (decimal:  122) --> VALID in SEG1: 0x000001fa (decimal:  506)
  VA  1: 0x00000079 (decimal:  121) --> VALID in SEG1: 0x000001f9 (decimal:  505)
  VA  2: 0x00000007 (decimal:    7) --> VALID in SEG0: 0x00000007 (decimal:    7)
  VA  3: 0x0000000a (decimal:   10) --> VALID in SEG0: 0x0000000a (decimal:   10)
  VA  4: 0x0000006a (decimal:  106) --> SEGMENTATION VIOLATION (SEG1)
```

## 16.2

段0最高的合法虚拟地址是19
段1最低的合法虚拟地址是108
整个地址空间中对子和最高的合法地址是0和127


```
└─$ ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2 -c -A 19,20,107,108,0,127,128
ARG seed 2
ARG address space size 128
ARG phys mem size 512

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 20

  Segment 1 base  (grows negative) : 0x00000200 (decimal 512)
  Segment 1 limit                  : 20

Virtual Address Trace
  VA  0: 0x00000013 (decimal:   19) --> VALID in SEG0: 0x00000013 (decimal:   19)
  VA  1: 0x00000014 (decimal:   20) --> SEGMENTATION VIOLATION (SEG0)
  VA  2: 0x0000006b (decimal:  107) --> SEGMENTATION VIOLATION (SEG1)
  VA  3: 0x0000006c (decimal:  108) --> VALID in SEG1: 0x000001ec (decimal:  492)
  VA  4: 0x00000000 (decimal:    0) --> VALID in SEG0: 0x00000000 (decimal:    0)
  VA  5: 0x0000007f (decimal:  127) --> VALID in SEG1: 0x000001ff (decimal:  511)
Error: virtual address 128 cannot be generated in an address space of size 128
```

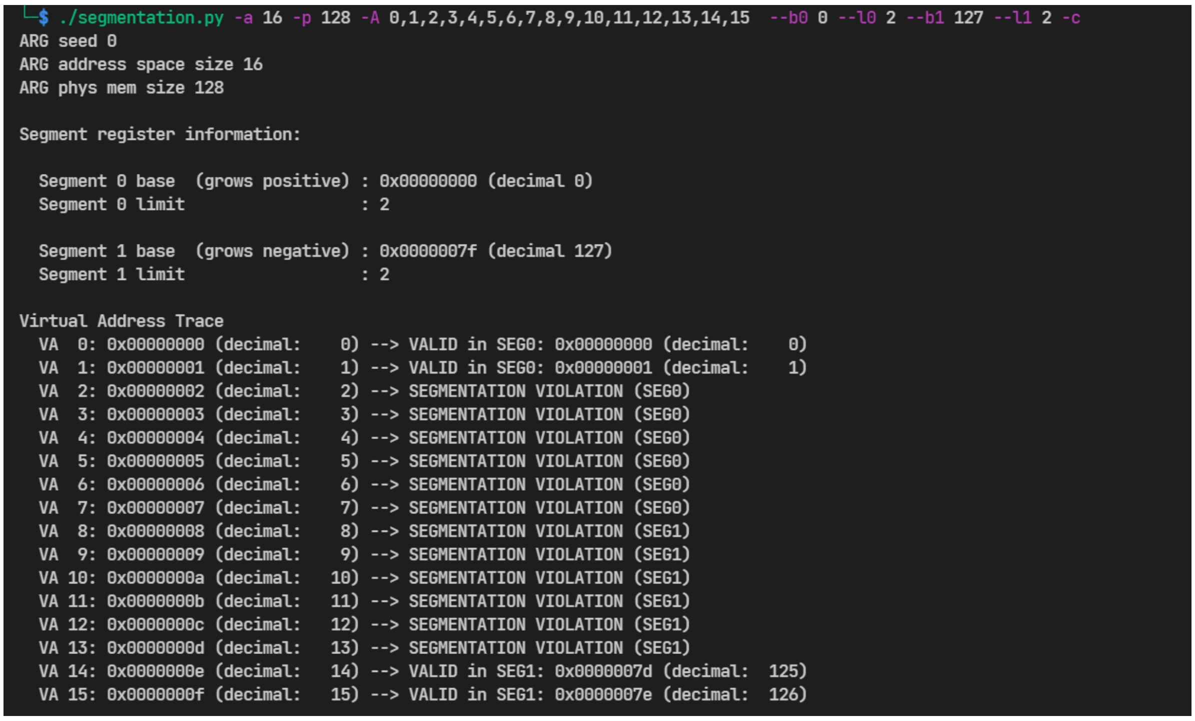
## 16.3

地址空间长度为16只有前两个和后两个虚拟地址有效

所以构造段0 起始地址0x0 长度为2 正方向增长

构造段1 起始地址0x7e 长度为2 负方向增长

运行结果如下



```
└─$ ./segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15  --b0 0 --l0 2 --b1 127 --l1 2 -c
ARG seed 0
ARG address space size 16
ARG phys mem size 128

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 2

  Segment 1 base  (grows negative) : 0x0000007f (decimal 127)
  Segment 1 limit                  : 2

Virtual Address Trace
  VA  0: 0x00000000 (decimal:    0) --> VALID in SEG0: 0x00000000 (decimal:    0)
  VA  1: 0x00000001 (decimal:    1) --> VALID in SEG0: 0x00000001 (decimal:    1)
  VA  2: 0x00000002 (decimal:    2) --> SEGMENTATION VIOLATION (SEG0)
  VA  3: 0x00000003 (decimal:    3) --> SEGMENTATION VIOLATION (SEG0)
  VA  4: 0x00000004 (decimal:    4) --> SEGMENTATION VIOLATION (SEG0)
  VA  5: 0x00000005 (decimal:    5) --> SEGMENTATION VIOLATION (SEG0)
  VA  6: 0x00000006 (decimal:    6) --> SEGMENTATION VIOLATION (SEG0)
  VA  7: 0x00000007 (decimal:    7) --> SEGMENTATION VIOLATION (SEG0)
  VA  8: 0x00000008 (decimal:    8) --> SEGMENTATION VIOLATION (SEG1)
  VA  9: 0x00000009 (decimal:    9) --> SEGMENTATION VIOLATION (SEG1)
  VA 10: 0x0000000a (decimal:   10) --> SEGMENTATION VIOLATION (SEG1)
  VA 11: 0x0000000b (decimal:   11) --> SEGMENTATION VIOLATION (SEG1)
  VA 12: 0x0000000c (decimal:   12) --> SEGMENTATION VIOLATION (SEG1)
  VA 13: 0x0000000d (decimal:   13) --> SEGMENTATION VIOLATION (SEG1)
  VA 14: 0x0000000e (decimal:   14) --> VALID in SEG1: 0x0000007d (decimal:  125)
  VA 15: 0x0000000f (decimal:   15) --> VALID in SEG1: 0x0000007e (decimal:  126)
```

## 18.1

页表项数统计如下:

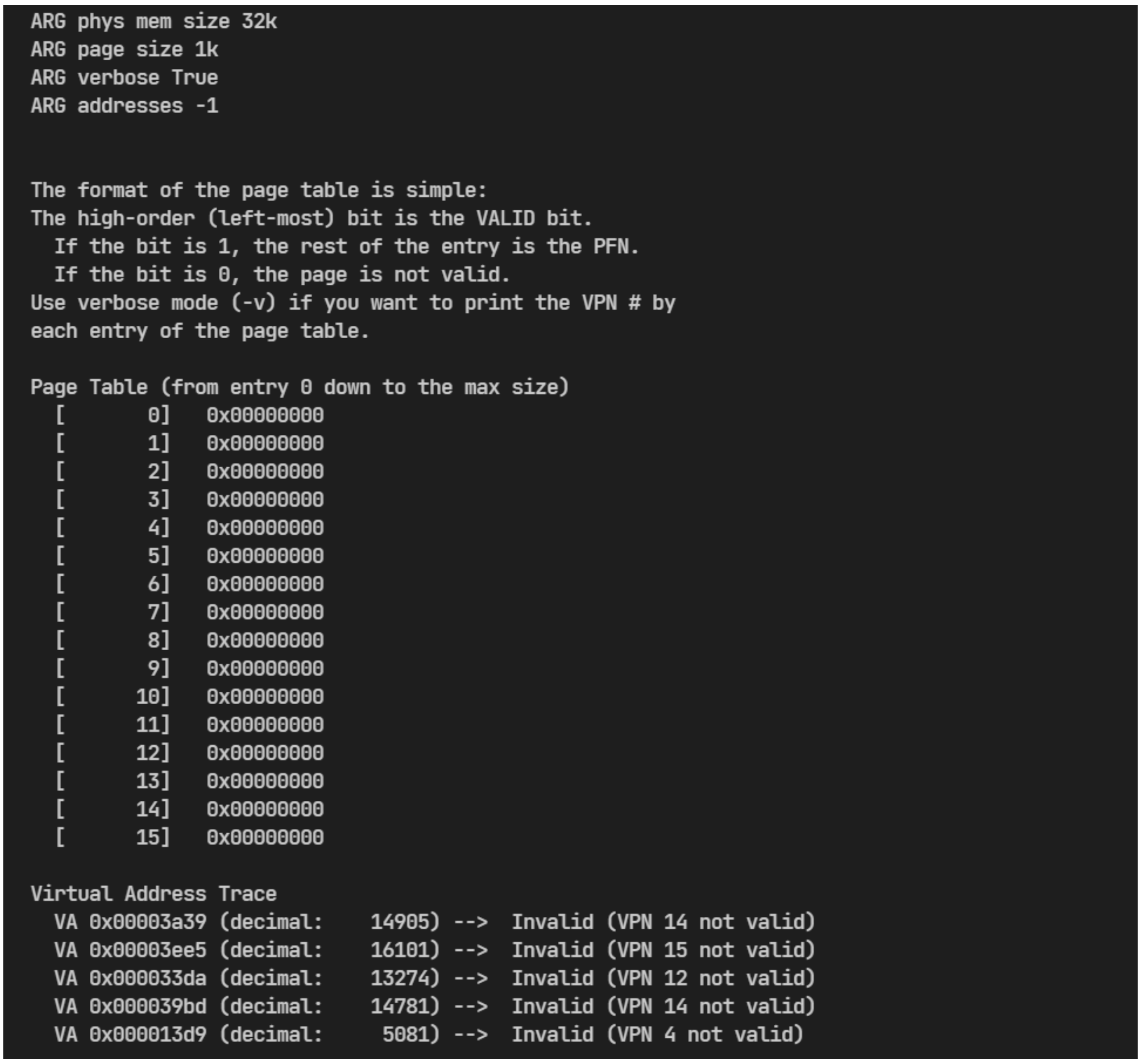
```
$ ./paging-linear-translate.py -P 1k -a 1m -p 512m -v -n 0
1024
$ ./paging-linear-translate.py -P 1k -a 2m -p 512m -v -n 0
2048
$ ./paging-linear-translate.py -P 1k -a 4m -p 512m -v -n 0
4096
$ ./paging-linear-translate.py -P 1k -a 1m -p 512m -v -n 0
1024
$ ./paging-linear-translate.py -P 2k -a 1m -p 512m -v -n 0
512
$ ./paging-linear-translate.py -P 4k -a 1m -p 512m -v -n 0
256
```

通过上面的数据可以看到

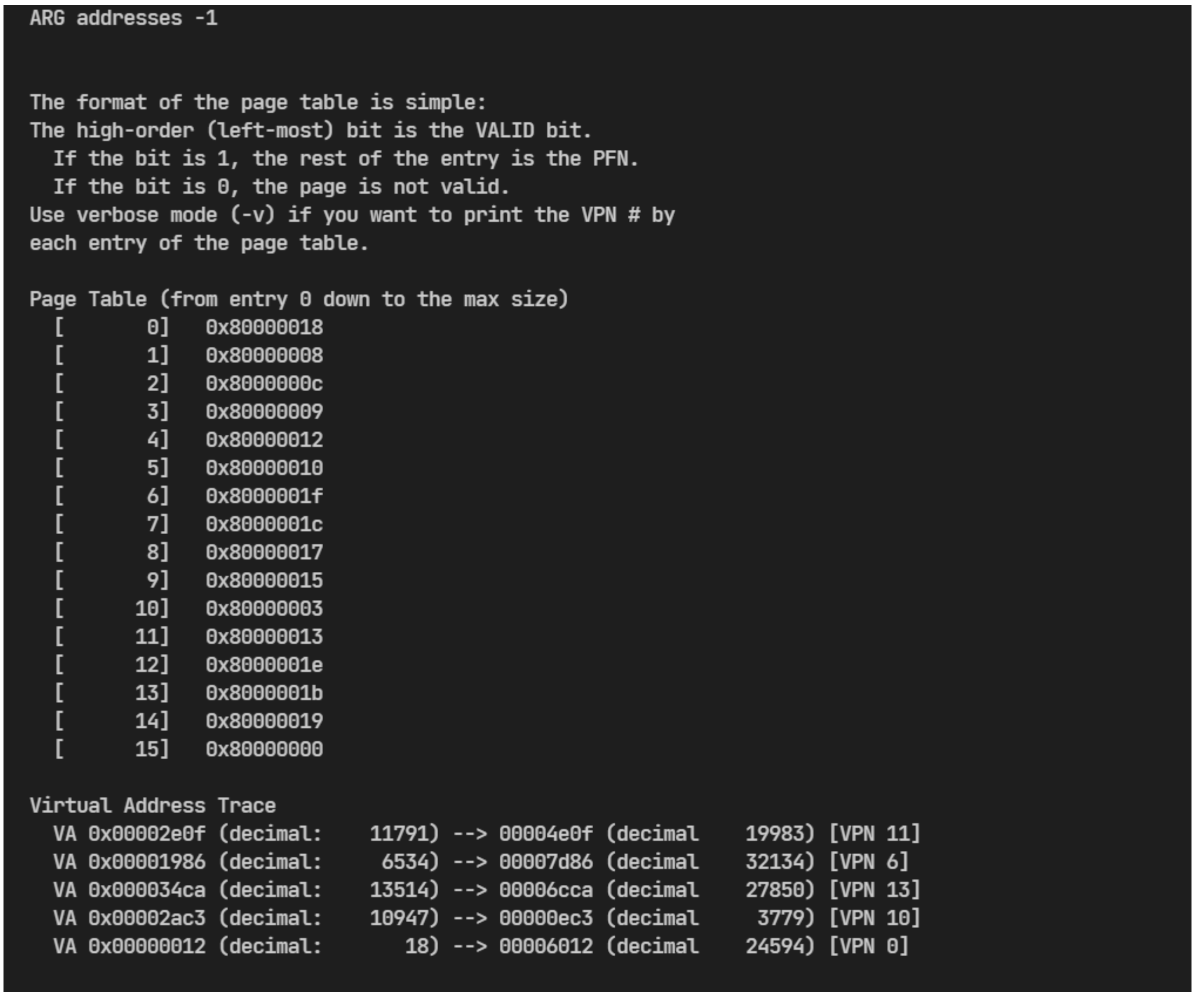
页表的大小随地址空间的增大而增大 随页大小的增大而减小

页面很大 会导致很多内部碎片 浪费空间

## 18.2



```
ARG phys mem size 32k
ARG page size 1k
ARG verbose True
ARG addresses -1


The format of the page table is simple:
The high-order (left-most) bit is the VALID bit.
  If the bit is 1, the rest of the entry is the PFN.
  If the bit is 0, the page is not valid.
Use verbose mode (-v) if you want to print the VPN # by
each entry of the page table.

Page Table (from entry 0 down to the max size)
  [       0]   0x00000000
  [       1]   0x00000000
  [       2]   0x00000000
  [       3]   0x00000000
  [       4]   0x00000000
  [       5]   0x00000000
  [       6]   0x00000000
  [       7]   0x00000000
  [       8]   0x00000000
  [       9]   0x00000000
  [      10]   0x00000000
  [      11]   0x00000000
  [      12]   0x00000000
  [      13]   0x00000000
  [      14]   0x00000000
  [      15]   0x00000000

Virtual Address Trace
  VA 0x00003a39 (decimal:    14905) -->  Invalid (VPN 14 not valid)
  VA 0x00003ee5 (decimal:    16101) -->  Invalid (VPN 15 not valid)
  VA 0x000033da (decimal:    13274) -->  Invalid (VPN 12 not valid)
  VA 0x000039bd (decimal:    14781) -->  Invalid (VPN 14 not valid)
  VA 0x000013d9 (decimal:     5081) -->  Invalid (VPN 4 not valid)
```

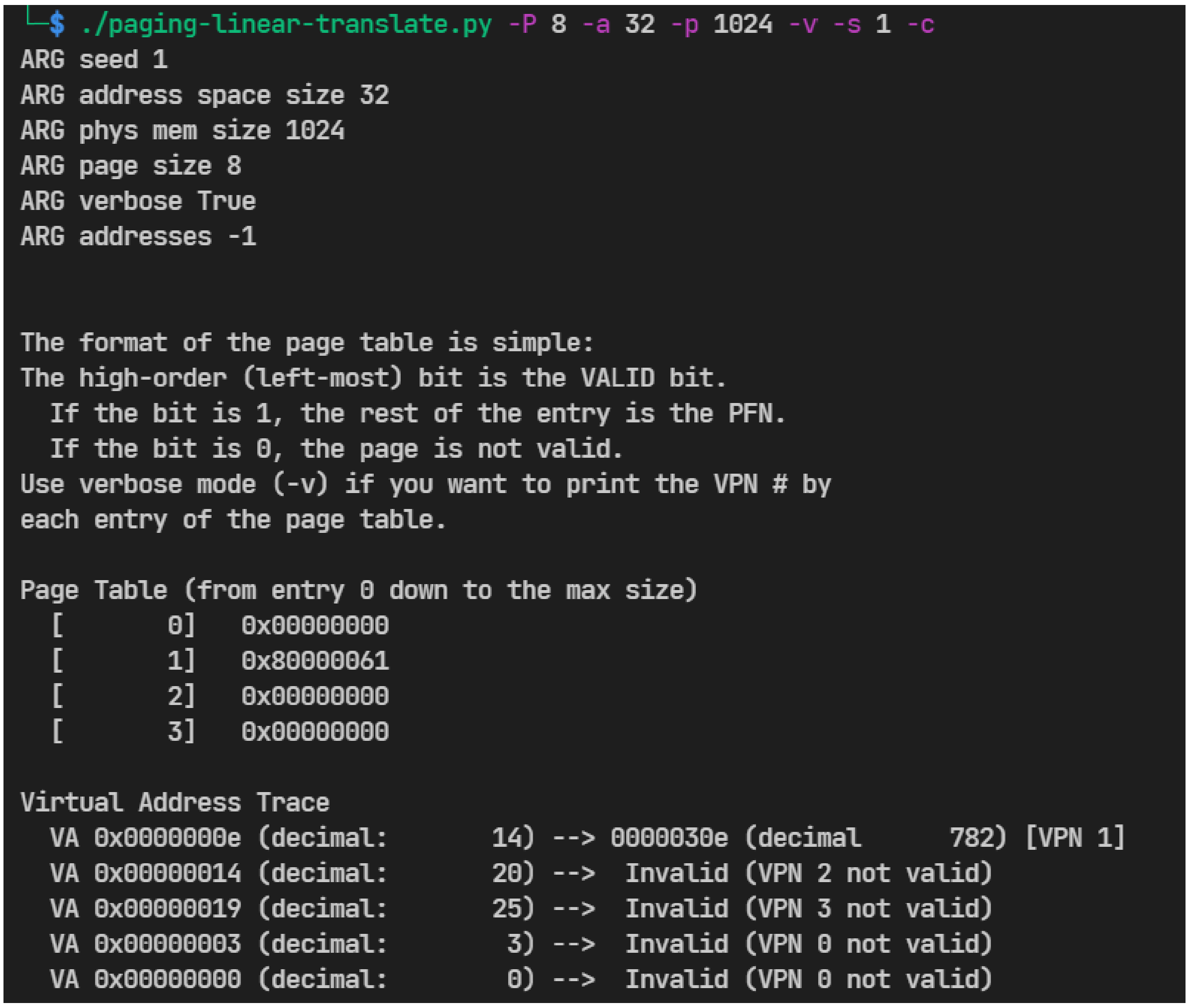


```
ARG addresses -1


The format of the page table is simple:
The high-order (left-most) bit is the VALID bit.
  If the bit is 1, the rest of the entry is the PFN.
  If the bit is 0, the page is not valid.
Use verbose mode (-v) if you want to print the VPN # by
each entry of the page table.

Page Table (from entry 0 down to the max size)
  [       0]   0x80000018
  [       1]   0x80000008
  [       2]   0x8000000c
  [       3]   0x80000009
  [       4]   0x80000012
  [       5]   0x80000010
  [       6]   0x8000001f
  [       7]   0x8000001c
  [       8]   0x80000017
  [       9]   0x80000015
  [      10]   0x80000003
  [      11]   0x80000013
  [      12]   0x8000001e
  [      13]   0x8000001b
  [      14]   0x80000019
  [      15]   0x80000000

Virtual Address Trace
  VA 0x00002e0f (decimal:    11791) --> 00004e0f (decimal    19983) [VPN 11]
  VA 0x00001986 (decimal:     6534) --> 00007d86 (decimal    32134) [VPN 6]
  VA 0x000034ca (decimal:    13514) --> 00006cca (decimal    27850) [VPN 13]
  VA 0x00002ac3 (decimal:    10947) --> 00000ec3 (decimal     3779) [VPN 10]
  VA 0x00000012 (decimal:       18) --> 00006012 (decimal    24594) [VPN 0]
```

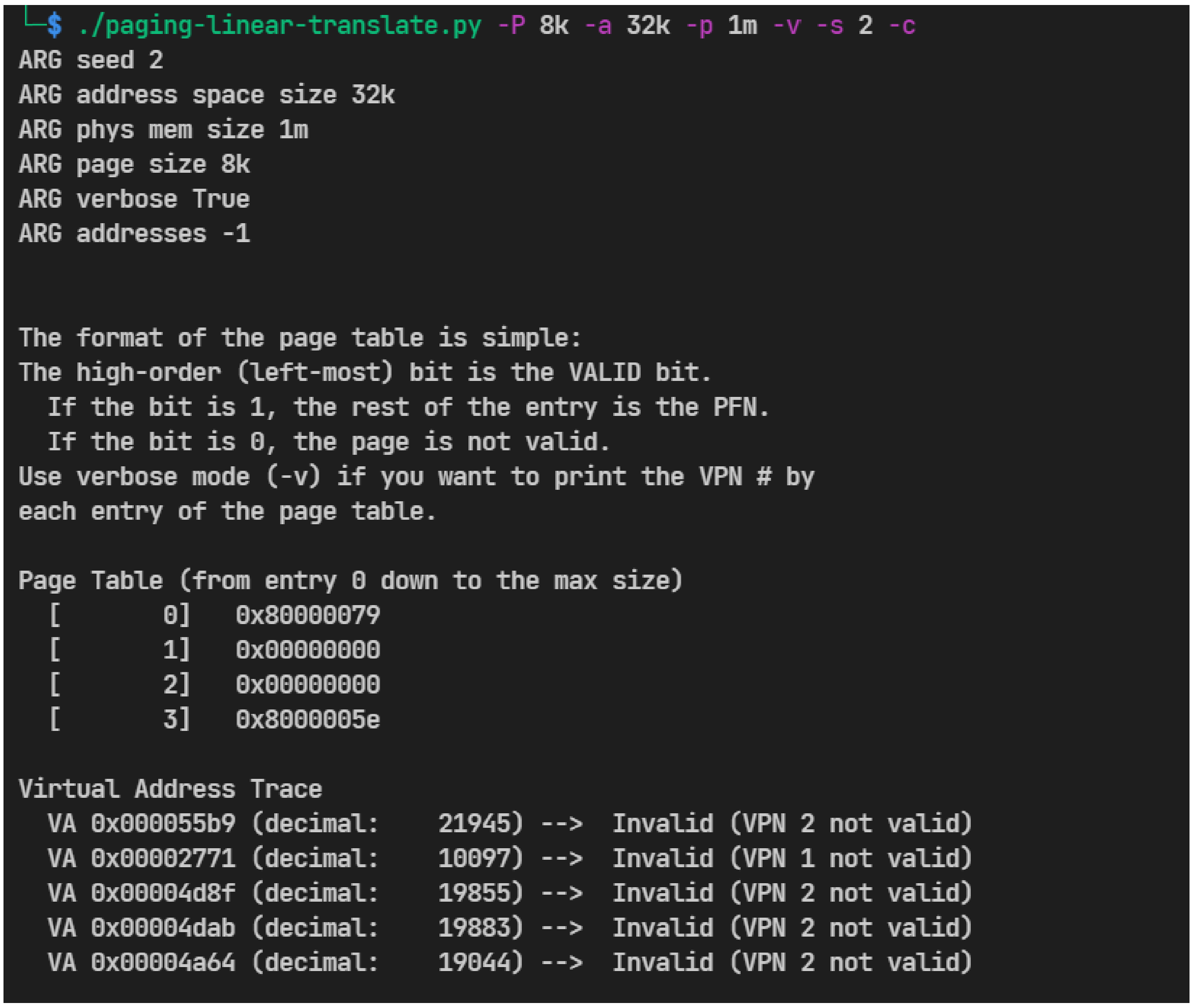
结果过多 只列出了开头和结尾

通过过程可以看到增加每个地址空间中页的百分比 可以提高虚拟地址转换的成功率

## 18.3



```
└─$ ./paging-linear-translate.py -P 8 -a 32 -p 1024 -v -s 1 -c
ARG seed 1
ARG address space size 32
ARG phys mem size 1024
ARG page size 8
ARG verbose True
ARG addresses -1


The format of the page table is simple:
The high-order (left-most) bit is the VALID bit.
  If the bit is 1, the rest of the entry is the PFN.
  If the bit is 0, the page is not valid.
Use verbose mode (-v) if you want to print the VPN # by
each entry of the page table.

Page Table (from entry 0 down to the max size)
  [       0]   0x00000000
  [       1]   0x80000061
  [       2]   0x00000000
  [       3]   0x00000000

Virtual Address Trace
  VA 0x0000000e (decimal:       14) --> 0000030e (decimal      782) [VPN 1]
  VA 0x00000014 (decimal:       20) -->  Invalid (VPN 2 not valid)
  VA 0x00000019 (decimal:       25) -->  Invalid (VPN 3 not valid)
  VA 0x00000003 (decimal:        3) -->  Invalid (VPN 0 not valid)
  VA 0x00000000 (decimal:        0) -->  Invalid (VPN 0 not valid)
```



```
└─$ ./paging-linear-translate.py -P 8k -a 32k -p 1m -v -s 2 -c
ARG seed 2
ARG address space size 32k
ARG phys mem size 1m
ARG page size 8k
ARG verbose True
ARG addresses -1


The format of the page table is simple:
The high-order (left-most) bit is the VALID bit.
  If the bit is 1, the rest of the entry is the PFN.
  If the bit is 0, the page is not valid.
Use verbose mode (-v) if you want to print the VPN # by
each entry of the page table.

Page Table (from entry 0 down to the max size)
  [       0]   0x80000079
  [       1]   0x00000000
  [       2]   0x00000000
  [       3]   0x8000005e

Virtual Address Trace
  VA 0x000055b9 (decimal:    21945) -->  Invalid (VPN 2 not valid)
  VA 0x00002771 (decimal:    10097) -->  Invalid (VPN 1 not valid)
  VA 0x00004d8f (decimal:    19855) -->  Invalid (VPN 2 not valid)
  VA 0x00004dab (decimal:    19883) -->  Invalid (VPN 2 not valid)
  VA 0x00004a64 (decimal:    19044) -->  Invalid (VPN 2 not valid)
```

```
Virtual Address Trace
  VA 0x0308b24d (decimal: 50901581) --> 1f68b24d (decimal 526955085) [VPN 48]
  VA 0x042351e6 (decimal: 69423590) -->  Invalid (VPN 66 not valid)
  VA 0x02feb67b (decimal: 50247291) --> 0a9eb67b (decimal 178173563) [VPN 47]
  VA 0x0b46977d (decimal: 189175677) -->  Invalid (VPN 180 not valid)
  VA 0x0dbcceb4 (decimal: 230477492) --> 1f2cceb4 (decimal 523030196) [VPN 219]
```

前两个地址空间太小了

第三个页面太大了

## 20.1

也是一个寄存器

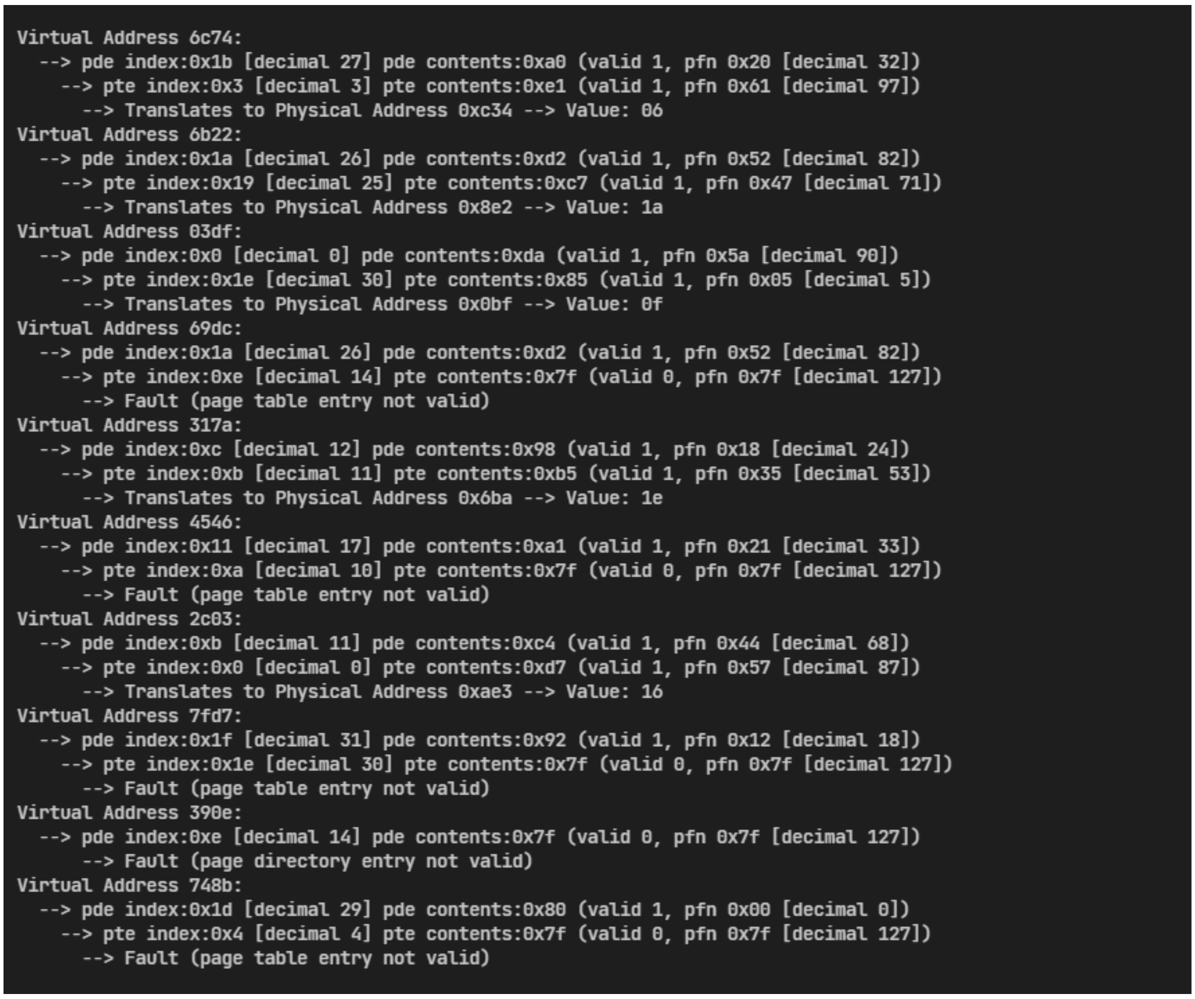
因为多级页表是通过分段虚拟地址来进行定位页表 所以几级页表都是一个寄存器

## 20.2

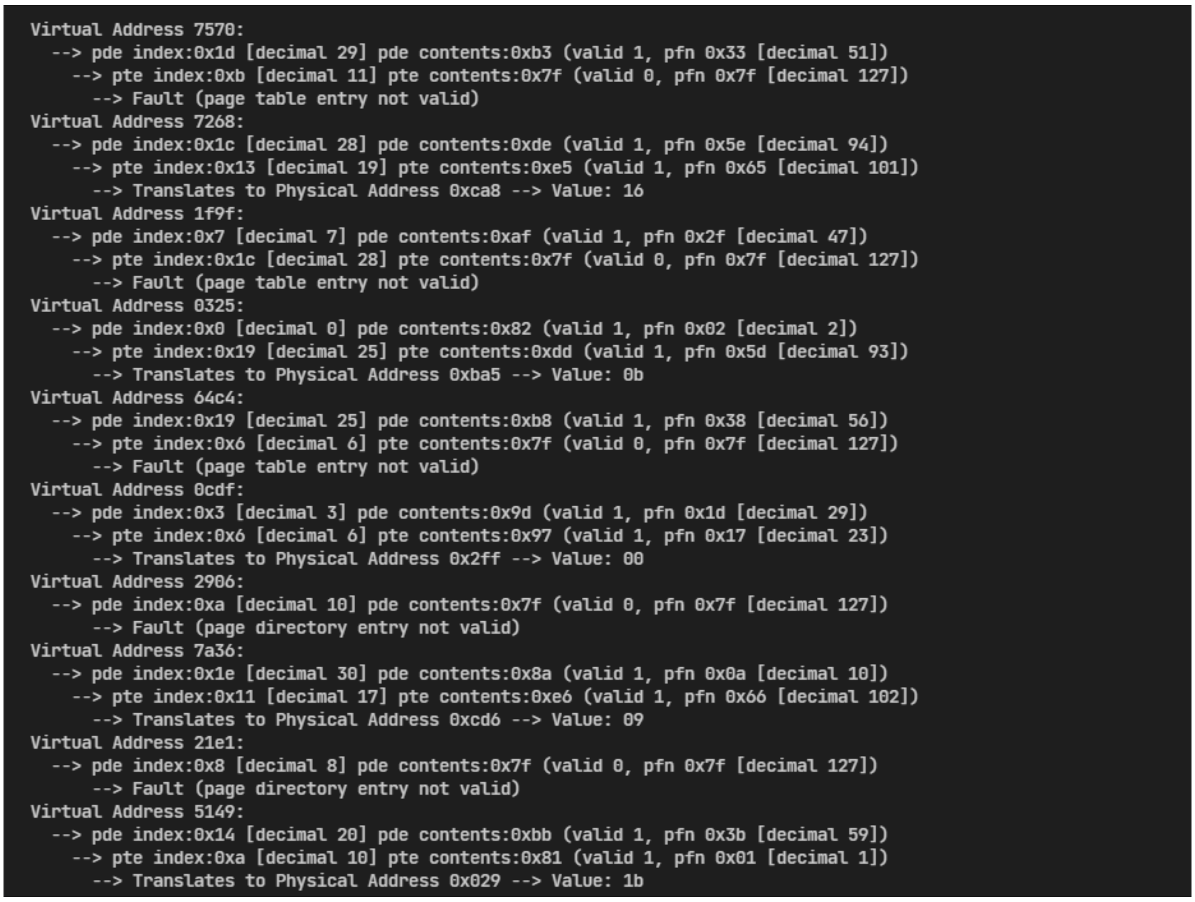
通过查看VA中的pde找到pte然后找到物理地址

计算结果如下:

```
Virtual Address 611c:
  --> 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: 08
Virtual Address 3da8:
  --> 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 17f5:
  --> 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: 1c
Virtual Address 7f6c:
  --> 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 0bad:
  --> 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 6d60:
  --> 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 2a5b:
  --> 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 4c5e:
  --> 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 2592:
  --> 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: 1b
Virtual Address 3e99:
  --> 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: 1e
```



```
Virtual Address 6c74:
  --> 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: 06
Virtual Address 6b22:
  --> 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: 1a
Virtual Address 03df:
  --> 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: 0f
Virtual Address 69dc:
  --> 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 317a:
  --> 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: 1e
Virtual Address 4546:
  --> 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 2c03:
  --> 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: 16
Virtual Address 7fd7:
  --> 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 390e:
  --> pde index:0xe [decimal 14] pde contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page directory entry not valid)
Virtual Address 748b:
  --> 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)
```



```
Virtual Address 7570:
  --> 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 7268:
  --> 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: 16
Virtual Address 1f9f:
  --> 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 0325:
  --> 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: 0b
Virtual Address 64c4:
  --> 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 0cdf:
  --> 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: 00
Virtual Address 2906:
  --> pde index:0xa [decimal 10] pde contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page directory entry not valid)
Virtual Address 7a36:
  --> 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: 09
Virtual Address 21e1:
  --> pde index:0x8 [decimal 8] pde contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page directory entry not valid)
Virtual Address 5149:
  --> 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: 1b
```

## 20.3

程序使用内存是会经常访问页表的

所以页表应该在缓存中 可以保证大量命中

很多不命中应该只会存在程序刚刚启动时(冷不命中)

## 22.1

统计结果如下:

```
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

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

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

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
```

```
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

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

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

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

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

## 22.2

6 5 4 3 2 1

全不命中

## 22.3

使用python脚本生成随机数

```python
#! /usr/bin/python3
import random
Max =  10
num =  10

arr = []
for i in range(0, num):
    t = random.randint(0, Max - 1)
    arr.append(t)
print(arr)

file = open('./test.txt', 'w')

for i in arr:
    file.write(str(i) + '\n')

file.close()
```

ARG addresses -1

ARG addressfile test.txt

ARG numaddrs 10

ARG policy FIFO

ARG clockbits 2

ARG cachesize 3

ARG maxpage 10

ARG seed 0

```
ARG notrace False

Solving...

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

FINALSTATS hits 5  misses 5  hitrate 50.00

ARG addresses -1
ARG addressfile test.txt
ARG numaddrs 10
ARG policy LRU
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

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

```
Access: 2  HIT  LRU ->   [9, 3, 2] <- MRU Replaced:- [Hits:2 Misses:4]

Access: 3  HIT  LRU ->   [9, 2, 3] <- MRU Replaced:- [Hits:3 Misses:4]

Access: 5  MISS LRU ->   [2, 3, 5] <- MRU Replaced:9 [Hits:3 Misses:5]

Access: 9  MISS LRU ->   [3, 5, 9] <- MRU Replaced:2 [Hits:3 Misses:6]

Access: 2  MISS LRU ->   [5, 9, 2] <- MRU Replaced:3 [Hits:3 Misses:7]


FINALSTATS hits 3  misses 7  hitrate 30.00


ARG addresses -1

ARG addressfile test.txt

ARG numaddrs 10

ARG policy OPT

ARG clockbits 2

ARG cachesize 3

ARG maxpage 10

ARG seed 0

ARG notrace False


Solving...


Access: 4  MISS Left  ->      [4] <- Right Replaced:- [Hits:0 Misses:1]

Access: 3  MISS Left  ->   [4, 3] <- Right Replaced:- [Hits:0 Misses:2]

Access: 9  MISS Left  ->  [4, 3, 9] <- Right Replaced:- [Hits:0 Misses:3]

Access: 3  HIT  Left  ->  [4, 3, 9] <- Right Replaced:- [Hits:1 Misses:3]

Access: 2  MISS Left  ->  [3, 9, 2] <- Right Replaced:4 [Hits:1 Misses:4]

Access: 2  HIT  Left  ->  [3, 9, 2] <- Right Replaced:- [Hits:2 Misses:4]

Access: 3  HIT  Left  ->  [3, 9, 2] <- Right Replaced:- [Hits:3 Misses:4]

Access: 5  MISS Left  ->  [9, 2, 5] <- Right Replaced:3 [Hits:3 Misses:5]

Access: 9  HIT  Left  ->  [9, 2, 5] <- Right Replaced:- [Hits:4 Misses:5]

Access: 2  HIT  Left  ->  [9, 2, 5] <- Right Replaced:- [Hits:5 Misses:5]


FINALSTATS hits 5  misses 5  hitrate 50.00
```

## 22.4

```
#! /usr/bin/python3
```

```python
import random
Max =  10
num =  10
randCite = 0.8
randomHotPage = 0.2

arr = []

numlist = set(range(0, Max))
hotPage = set()
coldPage = set()

while len(hotPage) < int(Max * randomHotPage):
    i = random.choice(list(numlist))
    hotPage.add(i)
coldPage = numlist - hotPage
print(hotPage)
print(coldPage)
hotPage = list(hotPage)
coldPage = list(coldPage)

for i in range(0, num):
    if(random.random() > randCite):
        t = random.choice(coldPage)
    else:
        t = random.choice(hotPage)
    arr.append(t)
print(arr)

file = open('./1.txt', 'w')

for i in arr:
    file.write(str(i) + '\n')

file.close()
```

ARG addresses -1

ARG addressfile test.txt

ARG numaddrs 10

ARG policy FIFO

ARG clockbits 2

ARG cachesize 3

ARG maxpage 10

ARG seed 0

ARG notrace False

Solving...

```
Access: 4  MISS FirstIn ->      [4] <- Lastin  Replaced:- [Hits:0 Misses:1]

Access: 3  MISS FirstIn ->    [4, 3] <- Lastin  Replaced:- [Hits:0 Misses:2]

Access: 9  MISS FirstIn ->   [4, 3, 9] <- Lastin  Replaced:- [Hits:0 Misses:3]

Access: 3  HIT  FirstIn ->   [4, 3, 9] <- Lastin  Replaced:- [Hits:1 Misses:3]

Access: 2  MISS FirstIn ->   [3, 9, 2] <- Lastin  Replaced:4 [Hits:1 Misses:4]

Access: 2  HIT  FirstIn ->   [3, 9, 2] <- Lastin  Replaced:- [Hits:2 Misses:4]

Access: 3  HIT  FirstIn ->   [3, 9, 2] <- Lastin  Replaced:- [Hits:3 Misses:4]

Access: 5  MISS FirstIn ->   [9, 2, 5] <- Lastin  Replaced:3 [Hits:3 Misses:5]

Access: 9  HIT  FirstIn ->   [9, 2, 5] <- Lastin  Replaced:- [Hits:4 Misses:5]

Access: 2  HIT  FirstIn ->   [9, 2, 5] <- Lastin  Replaced:- [Hits:5 Misses:5]


FINALSTATS hits 5  misses 5  hitrate 50.00


ARG addresses -1

ARG addressfile test.txt

ARG numaddrs 10

ARG policy LRU

ARG clockbits 2

ARG cachesize 3

ARG maxpage 10

ARG seed 0

ARG notrace False


Solving...


Access: 4  MISS LRU ->      [4] <- MRU Replaced:- [Hits:0 Misses:1]

Access: 3  MISS LRU ->    [4, 3] <- MRU Replaced:- [Hits:0 Misses:2]

Access: 9  MISS LRU ->   [4, 3, 9] <- MRU Replaced:- [Hits:0 Misses:3]

Access: 3  HIT  LRU ->   [4, 9, 3] <- MRU Replaced:- [Hits:1 Misses:3]

Access: 2  MISS LRU ->   [9, 3, 2] <- MRU Replaced:4 [Hits:1 Misses:4]

Access: 2  HIT  LRU ->   [9, 3, 2] <- MRU Replaced:- [Hits:2 Misses:4]

Access: 3  HIT  LRU ->   [9, 2, 3] <- MRU Replaced:- [Hits:3 Misses:4]

Access: 5  MISS LRU ->   [2, 3, 5] <- MRU Replaced:9 [Hits:3 Misses:5]

Access: 9  MISS LRU ->   [3, 5, 9] <- MRU Replaced:2 [Hits:3 Misses:6]
```

```
Access: 2  MISS LRU ->   [5, 9, 2] <- MRU Replaced:3 [Hits:3 Misses:7]

FINALSTATS hits 3  misses 7  hitrate 30.00

ARG addresses -1
ARG addressfile test.txt
ARG numaddrs 10
ARG policy OPT
ARG clockbits 2
ARG cachesize 3
ARG maxpage 10
ARG seed 0
ARG notrace False

Solving...

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

FINALSTATS hits 5  misses 5  hitrate 50.00
```

CLOCK bit为1时表现差于LRU，提高bit时出现好于LRU的情况，且在一定数值内bit越高表现越好