第 16 章

1. 先让我们用一个小地址空间来转换一些地址。这里有一组简单的参数和几个不同的随机种子。你可以转换这些地址吗？

```
segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0

segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 1

segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2
```

①

```
[redacted] vm-segmentation % ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 1
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) --> PA or segmentation violation?
  VA  1: 0x0000006c (decimal:  108) --> PA or segmentation violation?
  VA  2: 0x00000061 (decimal:   97) --> PA or segmentation violation?
  VA  3: 0x00000020 (decimal:   32) --> PA or segmentation violation?
  VA  4: 0x0000003f (decimal:   63) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple address space with two segments: the top
bit of the virtual address can thus be used to check whether the virtual address
is in segment 0 (topbit=0) or segment 1 (topbit=1). Note that the base/limit pairs
given to you grow in different directions, depending on the segment, i.e., segment 0
grows in the positive direction, whereas segment 1 in the negative.
```

由设置的参数可知程序生成了 128 字节的地址空间的虚拟地址，512 字节的物理内存。

段 0 的基址（段顶）为 0，大小为 20 个字节，所以其虚拟地址范围为：0 ~ 19

段 1 的基址（段底）为 512，大小为 20 个字节，所以其虚拟地址范围为：108 ~ 127

则

VA 0：0x00000011 (decimal： 17) --> PA (seg0：17) (物理地址：0 + (17 - 0) = 17)

VA 1：0x0000006c (decimal： 108) --> PA (seg1：492) (物理地址：512 - (128 - 108) = 17)

VA 2：0x00000061 (decimal： 97) --> segmentation violation (seg1)

VA 3：0x00000020 (decimal： 32) --> segmentation violation (seg0)

VA 4：0x0000003f (decimal： 63) --> segmentation violation (seg0)

②

```
[redacted] ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2
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) --> PA or segmentation violation?
  VA  1: 0x00000079 (decimal:  121) --> PA or segmentation violation?
  VA  2: 0x00000007 (decimal:    7) --> PA or segmentation violation?
  VA  3: 0x0000000a (decimal:   10) --> PA or segmentation violation?
  VA  4: 0x0000006a (decimal:  106) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple address space with two segments: the top
bit of the virtual address can thus be used to check whether the virtual address
is in segment 0 (topbit=0) or segment 1 (topbit=1). Note that the base/limit pairs
given to you grow in different directions, depending on the segment, i.e., segment 0
grows in the positive direction, whereas segment 1 in the negative.
```

由设置的参数可知程序生成了 128 字节的地址空间的虚拟地址，512 字节的物理内存。

段 0 的基址（段顶）为 0，大小为 20 个字节，所以其虚拟地址范围为：0 ~ 19

段 1 的基址（段底）为 512，大小为 20 个字节，所以其虚拟地址范围为：108 ~ 127

则

VA 0：0x0000007a (decimal： 122) --> PA (seg1：506) (物理地址：512 - (128 - 122) = 506)

VA 1：0x00000079 (decimal： 121) --> PA (seg1：505) (物理地址：512 - (128 - 121) = 505)

VA 2：0x00000007 (decimal： 7) --> PA (seg0：7) (物理地址：0 + (7 - 0) = 7)

VA 3：0x0000000a (decimal： 10) --> PA (seg0：10) (物理地址：0 + (10 - 0) = 10)

VA 4：0x0000006a (decimal： 106) --> segmentation violation (seg1)

③

```
[redacted] vm-segmentation % ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 3
ARG seed 3
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: 0x0000001e (decimal:   30) --> PA or segmentation violation?
  VA  1: 0x00000045 (decimal:   69) --> PA or segmentation violation?
  VA  2: 0x0000002f (decimal:   47) --> PA or segmentation violation?
  VA  3: 0x0000004d (decimal:   77) --> PA or segmentation violation?
  VA  4: 0x00000050 (decimal:   80) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple address space with two segments: the top
bit of the virtual address can thus be used to check whether the virtual address
is in segment 0 (topbit=0) or segment 1 (topbit=1). Note that the base/limit pairs
given to you grow in different directions, depending on the segment, i.e., segment 0
grows in the positive direction, whereas segment 1 in the negative.
```

由设置的参数可知程序生成了 128 字节的地址空间的虚拟地址，512 字节的物理内存。

段 0 的基址（段顶）为 0，大小为 20 个字节，所以其虚拟地址范围为：0 ~ 19

段 1 的基址（段底）为 512，大小为 20 个字节，所以其虚拟地址范围为：108 ~ 127

则

VA 0：0x0000001e (decimal： 30) --> segmentation violation (seg0)

VA 1：0x00000045 (decimal： 69) --> segmentation violation (seg1)

VA 2：0x0000002f (decimal： 47) --> segmentation violation (seg0)

VA 3：0x0000004d (decimal： 77) --> segmentation violation (seg1)

VA 4：0x00000050 (decimal： 80) --> segmentation violation (seg1)

2. 现在，让我们看看是否理解了这个构建的小地址空间（使用上面问题的参数）。段 0 中最高的合法虚拟地址是什么？段 1 中最低的合法虚拟地址是什么？在整个地址空间中，最低和最高的非法地址是什么？最后，如何运行带有-A 标志的 segmentation.py 来测试你是否正确？

答：段 0 中最高的合法地址是：19；
段 1 中最低的合法地址是：108；
整个地址空间中最低和最高的非法地址分别是：20、107
根据 help 提示，-A 标志用于手动生成虚拟地址

```
                          vm-segmentation % ./segmentation.py -help
Usage: segmentation.py [options]

Options:
  -h, --help            show this help message and exit
  -s SEED, --seed=SEED  the random seed
  -A ADDRESSES, --addresses=ADDRESSES
                        a set of comma-separated pages to access; -1 means
                        randomly generate
  -a ASIZE, --asize=ASIZE
                        address space size (e.g., 16, 64k, 32m, 1g)
  -p PSIZE, --physmem=PSIZE
                        physical memory size (e.g., 16, 64k, 32m, 1g)
  -n NUM, --numaddrs=NUM
                        number of virtual addresses to generate
  -b BASE0, --b0=BASE0  value of segment 0 base register
  -l LEN0, --l0=LEN0    value of segment 0 limit register
  -B BASE1, --b1=BASE1  value of segment 1 base register
  -L LEN1, --l1=LEN1    value of segment 1 limit register
  -c                    compute answers for me
```

设置参数

segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -A 19,20,107,108 -c

```
    vm-segmentation % ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -A 19,20,107,108 -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: 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)
```

计算结果符合与给出答案符合

3. 假设我们在一个 128 字节的物理内存中有一个很小的 16 字节地址空间。你会设置什么样的基址和界限，以便让模拟器为指定的地址流生成以下转换结果：有效，有效，违规，……，违规，有效，有效？假设用以下参数：

```
segmentation.py -a 16 -p 128
```

```
-A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
--b0 ? --l0 ? --b1 ? --l1 ?
```

答：由设置的参数可知虚拟地址空间大小为 16 字节、物理内存大小为 128 字节
题目要求应是让前两个虚拟地址(0，1)和后两个虚拟地址(14，15)有效，其余地址无效，
所以整个地址空间中，最低和最高的非法地址是 2、13
段 0 的基址应为 0，大小为 2；
段 1 的基址范围为 4 ~ 128,大小为 2；
所以参数可设置为：
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 16 --l1 2