HW4

姓名: 陈锐林, 学号:21307130148

2023 年 9 月 30 日

Chapter16

Question1

这题中三行命令的前面都是相同的。address space size 是 128，所以需要 7 位来标识。在 7 位的虚拟地址中，大于 64(0x40) 的是 seg1，小于的是 seg0。再根据界限寄存器，所以合理的虚拟地址应该在 (0∼19)，(108∼127)。

(1) 生成的四条 VA 为：108/97/53/33/65，显然只有第一条是可取的，其他都是 segmentation violation；再由 512-20=492，得到其真实物理地址 (seg1 中)。

(2) 生成的四条 VA 为：17/108/97/32/63，前两条是有效的，剩下三条是 segmentation violation；0+17=17，512-20=492，得物理地址为 17(seg0)，492(seg1)。

(3) 生成的四条 VA 为：122/121/7/10/106，前四条是有效的，第五条是 segmentation violation；512-6=506，512-7=505，0+7=7，0+10=10，得物理地址为 506(seg1)，505(seg1)，7(seg0)，10(seg0)。

Question2

这题在上面已经指出了，seg0 最高有效：19，seg1 最高有效：108；最低和最高非法：20/127。验证如下：

```
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)
```

Question3

调用命令:python3 ./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 -c 就可以了，能得到如下结果：

```
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: 0x0000000e (decimal:   14)
  VA 15: 0x0000000f (decimal:   15) --> VALID in SEG1: 0x0000000f (decimal:   15)
```

Question4

为了让大约 90% 的地址都是有效的，借鉴 3 中的思路可以知道，应该满足：(limit0+limit1)/address_space_size $\geq$ 90%。

Question5

只要让界限长度为 0，所有的 VA 就都无效了。

Chapter18

Question1

根据观察，随着地址空间的增大，页表会变大；而随着页大小的增加，页表会变小。很大的页可能会导致开销增大，而且利用率会下降。

Question2

对于命令："python3 ./paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 0"，意思即：虚拟地址空间 16k，物理空间 32k，页大小 1k。以生成的虚拟地址"0x00003a39"为例计算，转为二进制为 1110 1000111001，1110 即指向页项 VPN=14，紧接着到页表查询结果为 0x00000000，即这项是无效的。而取-u 0 时，即所有页表项都失效，所以生成的四个都是无效的。如果我们增加-u 的值，有效的页项会更多。以-u 25 为例，生成虚拟地址"0x00002bc6"，转为二进制是 1010 1111000110，查询页表项 VPN=10，结果为 0x80000013；此时首位是 1，即有效；将虚拟地址的偏移 1111000110 + 物理页号 19 * 1k，得到物理地址：0x4fc6。而其他的地址转换也类似，结果如下表。而随着-u 的值增加，也让有效的页增多。

-u	VA	PA
0	0x00003a39	Invalid (VPN 14 not valid)
0	0x00003ee5	Invalid (VPN 15 not valid)
0	0x000033da	Invalid (VPN 12 not valid)
0	0x000039bd	Invalid (VPN 14 not valid)
0	0x000013d9	Invalid (VPN 4 not valid)
25	0x00003986	Invalid (VPN 14 not valid)
25	0x00002bc6	00004fc6 [VPN 10]
25	0x00001e37	Invalid (VPN 7 not valid)
25	0x00000671	Invalid (VPN 1 not valid)
25	0x00001bc9	Invalid (VPN 6 not valid)
50	0x00003385	00003f85 [VPN 12]
50	0x0000231d	Invalid (VPN 8 not valid)
50	0x000000e6	000060e6 [VPN 0]
50	0x00002e0f	Invalid (VPN 11 not valid)
50	0x00001986	00007586 [VPN 6]
75	0x00002e0f	00004e0f [VPN 11]
75	0x00001986	00007d86 [VPN 6]
75	0x000034ca	00006cca [VPN 13]
75	0x00002ac3	00000ec3 [VPN 10]
75	0x00000012	00006012 [VPN 0]
100	0x00002e0f	00004e0f [VPN 11]
100	0x00001986	00007d86 [VPN 6]
100	0x000034ca	00006cca [VPN 13]
100	0x00002ac3	00000ec3 [VPN 10]
100	0x00000012	00006012 [VPN 0]

Question3

可以看到，前两个组合的参数其实在比例上是一样的；而第三个组合的页太小了，虚拟地址有 256 页，物理地址有 512 页，会导致通过虚拟地址寻找页表中的项开销很大。

Question4

在这个程序里修改命令使得物理内存小于虚拟内存，会直接报错。而即使正常运行，也可能有的虚拟内存无法导入、或者也页无法正确寻找。