Tutorial 7 Solution

CS3103 Operating Systems

Student Name:
Student No. : | | | | | | | | |

Day: ☐ Monday ☐ Tuesday
Time: ☐ 10:00 - 10:50 ☐ 11:00 - 11:50 ☐ 14:00 - 14:50 ☐ 16:00 - 16:50 ☐ 18:00 - 18:50

Paging

Important Note: The tutorial solutions are given for reference only. They should not be used as "model answers" to help solve the tutorial problems. For current and former CS3103 students, **DO NOT** give the solutions to current CS3103 students. **DO NOT** post electronic copies of the solutions online.

Answers

1. (1) The size of the page table depends upon the number of entries in the table and the bytes stored in one entry. The number of page table entries increases linearly as the address size grows. (2) The number of page table entries is proportional to the inverse of page size: doubling the page size halves the number of entries. (3) Using large pages increases the risk of wasting actual memory through internal fragmentation.

2. More pages are valid. As the percentage of pages that are allocated in each address space increases, the percentage of random virtual addresses that trigger page faults decreases proportionally.

3. In a way, each is unrealistic. The first is unrealistically tiny in every way. The second is very small. The third has an incredibly large page size, given the average OS page size is 4KB. Note that the second and third parameter combination are possible. The first parameters are unrealistic because each page table would have to hold 128 entries, which could not possibly fit in a 32 bit address space. The second set of parameters, while not impossible (each page table entry would have to hold 4 PTEs, and each PTE would have to be at least 7bits in size, easily fitting into a 32k address space), would only result in 4 pages per process. This risks wasting a lot of space due to internal fragmentation. The third set of parameters are not as unrealistic as the first two: each page table would be composed of 256 entries and each entry would need to address one of 2^9 physical pages, yielding a minimum size for a flat page table of around 2k bits.

4. Virtual Address 5a23:

   --> pde index:0x16 [decimal 22] pde contents:0xb5 (valid 1, pfn 0x35 [decimal 53])

   --> pte index:0x11 [decimal 17] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])

   --> Fault (page table entry not valid)

   Virtual Address 14ab:

   --> pde index:0x5 [decimal 5] pde contents:0x7f (valid 0, pfn 0x7f [decimal 127])

   --> Fault (page directory entry not valid)

   Virtual Address 257e:

   --> pde index:0x9 [decimal 9] pde contents:0xc4 (valid 1, pfn 0x44 [decimal 68])

--> pte index:0xb [decimal 11] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])

--> Fault (page table entry not valid)

Virtual Address 7988:

--> pde index:0x1e [decimal 30] pde contents:0xbb (valid 1, pfn 0x3b [decimal 59])

--> pte index:0xc [decimal 12] pte contents:0x9d (valid 1, pfn 0x1d [decimal 29])

--> Translates to Physical Address 0x3a8 --> Value: 19

Virtual Address 75cf:

--> pde index:0x1d [decimal 29] pde contents:0xc6 (valid 1, pfn 0x46 [decimal 70])

--> pte index:0xe [decimal 14] pte contents:0x81 (valid 1, pfn 0x01 [decimal 1])

--> Translates to Physical Address 0x02f --> Value: 07

Virtual Address 3350:

--> pde index:0xc [decimal 12] pde contents:0xd4 (valid 1, pfn 0x54 [decimal 84])

--> pte index:0x1a [decimal 26] pte contents:0xc0 (valid 1, pfn 0x40 [decimal 64])

--> Translates to Physical Address 0x810 --> Value: 04

Virtual Address 0a70:

--> pde index:0x2 [decimal 2] pde contents:0xba (valid 1, pfn 0x3a [decimal 58])

--> pte index:0x13 [decimal 19] pte contents:0x87 (valid 1, pfn 0x07 [decimal 7])

--> Translates to Physical Address 0x0f0 --> Value: 1e

Virtual Address 55f9:

--> pde index:0x15 [decimal 21] pde contents:0xd5 (valid 1, pfn 0x55 [decimal 85])

--> pte index:0xf [decimal 15] pte contents:0x86 (valid 1, pfn 0x06 [decimal 6])

--> Translates to Physical Address 0x0d9 --> Value: 1d