Tutorial 7 Answer Sheet

CS3103 Operating Systems

Student Name: LIU Hengche

Student No. : 5 7 8 5 4 3 2 9

Day: ☐Tuesday ☑Wednesday ☐Friday

Time: ☐ 10:00 - 10:50 ☐ 11:00 - 11:50 ☐ 16:00 - 16:50 ☑ 18:00 - 18:50

Paging

Submission:

- Deadline: Sunday, April 14, 2024, 23:59 pm HKT.
- Answers are allowed in text only. Any form of image/snapshot is not allowed.
- Submit this answer sheet via Canvas->Assignments->Tutorials->Tutorial 7.

Questions

Question 1: Before doing any translations, let's use the simulator `paging-linear-translate.py` to study how linear page tables change size given different parameters. Compute the size of linear page tables as different parameters change. Some suggested inputs are below; by using the `-v` flag, you can see how many page-table entries are filled. First, to understand how linear page table size changes as the address space grows, run with these flags:

```
-P 1k -a 1m -p 512m -v -n 0
-P 1k -a 2m -p 512m -v -n 0
-P 1k -a 4m -p 512m -v -n 0
```

Then, to understand how linear page table size changes as page size grows:

```
-P 1k -a 1m -p 512m -v -n 0
-P 2k -a 1m -p 512m -v -n 0
-P 4k -a 1m -p 512m -v -n 0
```

Before running any of these, try to think about the expected trends. How should page-table size change as the address space grows? As the page size grows? Why not use big pages in general?

Answer:

As the address space grows, the page-table size should also increase linearly accordingly as the pages should cover the entire address space.

As the page size grows, the page-table size should decrease. According to "page_table_size = address_space / page_size", the times that page size increases should be proportional to the times that page-table size shrinks.

When we have bigger page size, then it indicates larger internal fragmentation, which is a large overhead that takes up a lot of memory space.

Question 2: Now let's do some translations. Start with some small examples and change the number of pages that are allocated to the address space with the `-u` flag. For example:

```
-P 1k -a 16k -p 32k -v -u 0
-P 1k -a 16k -p 32k -v -u 25
-P 1k -a 16k -p 32k -v -u 50
-P 1k -a 16k -p 32k -v -u 75
-P 1k -a 16k -p 32k -v -u 100
```

What happens as you increase the percentage of pages that are allocated in each address space?

Answer:

As we increase the percentage of the pages allocated, more pages are valid (i.e. more pages begin with the valid bit of "1"). The flag "U" represents the "percent of virtual address space that is used", so with bigger U, comes more valid pages. Some output as follow:

```
-P 1k -a 16k -p 32k -v -u 0
```

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

```
-P 1k -a 16k -p 32k -v -u 25
```

Page Table (from entry 0 down to the max size)

[0] 0x80000018

[1] 0x00000000

[2] 0x00000000

[3] 0x00000000

[4] 0x00000000

[5] 0x80000009

[6] 0x00000000

[7] 0x00000000

[8] 0x80000010

[9] 0x00000000

[10] 0x80000013

[11] 0x00000000

[12] 0x8000001f

[13] 0x8000001c

[14] 0x00000000

[15] 0x00000000

Question 3: Now let's try some different random seeds, and some different (and sometimes quite crazy) address-space parameters, for variety:

```
-P 8 -a 32 -p 1024 -v -s 1
-P 8k -a 32k -p 1m -v -s 2
-P 1m -a 256m -p 512m -v -s 3
```

Which of these parameter combinations are unrealistic? Why?

Answer:

They are all unrealistic.

For the first combination, the physical memory, the address space and the size of the page are unrealistically small, making it unrealistic. For the second combination, still the corresponding space size for each flag is quite small, given that normal page size should be 4KB. For the third combination, the sizes are too large. Thus, the three combinations are all not quite realistic.

However, for the second set of flags, it would result in large internal fragmentation and wasting a lot memory space, but it is not impossible to implement. Same for the third one, each entry of page table would need to record the address of one of the 512 frames, thus it would require a large size for the page table, still not impossible to implement.

Question 4: Use the simulator `paging-multilevel-translate.py` to perform translations. Run:

```
$ ./paging-multilevel-translate.py -s 3103
```

You are given the value of the PDBR, a complete dump of each page of memory, and a list of virtual addresses to translate. Solutions to the first two addresses are given below for reference:

- Virtual Address 4a14: Translates to what Physical Address (and fetches what value)? Or Fault?

```
Virtual Address 4a14:
  --> pde index:0x12 [decimal 18] pde contents:0x9a (valid 1, pfn 0x1a [decimal 26])
    --> pte index:0x10 [decimal 16] pte contents:0xd8 (valid 1, pfn 0x58 [decimal 88])
      --> Translates to Physical Address 0xb14 --> Value: 17
```

- Virtual Address 685e: Translates to what Physical Address (and fetches what value)? Or Fault?

```
Virtual Address 685e:
  --> pde index:0x1a [decimal 26] pde contents:0xbf (valid 1, pfn 0x3f [decimal 63])
    --> pte index:0x2 [decimal 2] pte contents:0x7f (valid 0, pfn 0x7f [decimal 127])
      --> Fault (page table entry not valid)
```

For each of the following virtual addresses, write down the physical address it translates to or write down that it is a fault (e.g., page directory entry not valid, page table entry not valid).

- Virtual Address 5a23
- Virtual Address 14ab
- Virtual Address 257e
- Virtual Address 7988
- Virtual Address 75cf
- Virtual Address 3350
- Virtual Address 0a70
- Virtual Address 55f9

Answer:

Virtual Address 5a23: this is fault.
Virtual Address 14ab: this is fault.
Virtual Address 257e: this is fault.
Virtual Address 7988: the physical address it translates to is 19.
Virtual Address 75cf: the physical address it translates to is 07.
Virtual Address 3350: the physical address it translates to is 04.
Virtual Address 0a70: the physical address it translates to is 1e.
Virtual Address 55f9: the physical address it translates to is 1d.