CS304
Assignment 3

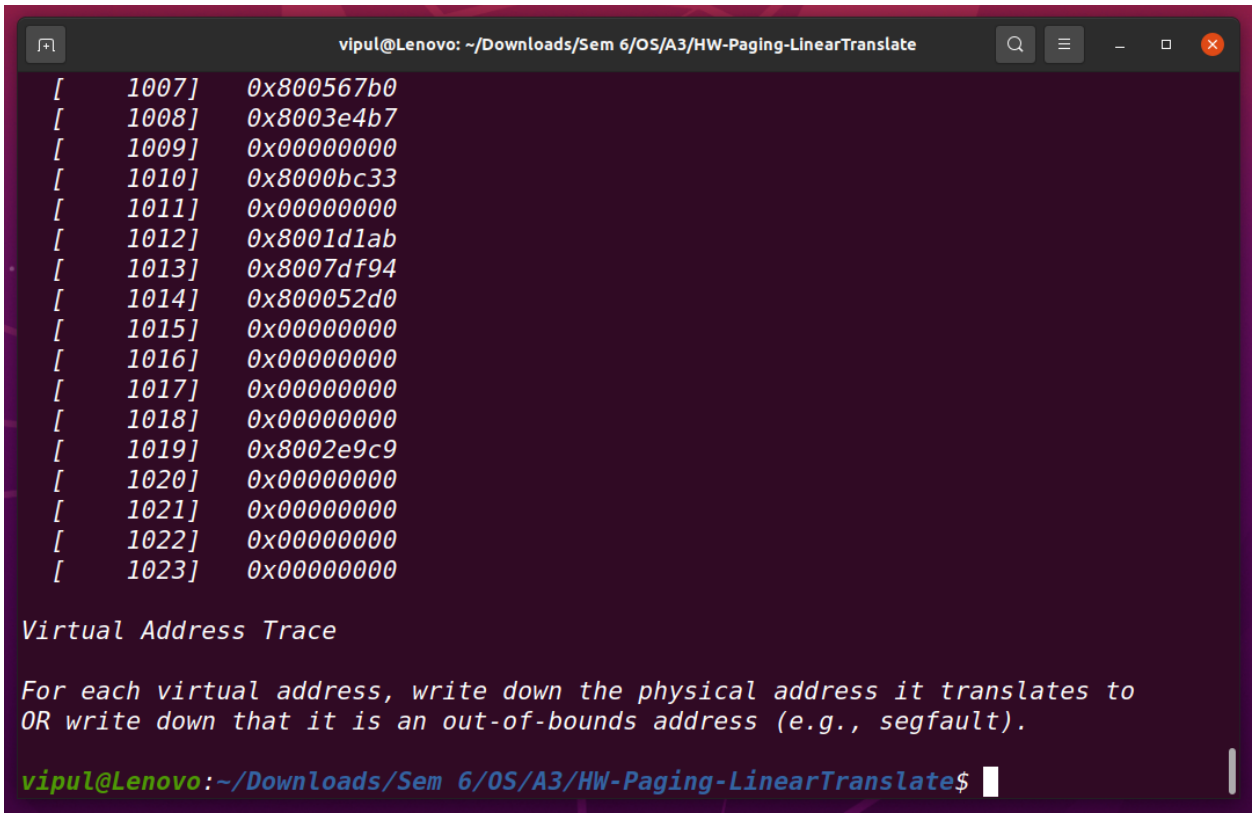
Vipul Yuvraj Nikam
180010041

LINEAR PAGE TABLE

Question 1:

a) How should the page table size change as the address space increases?

Answer: With the increase in the address space, the size of the page table also increases as more pages would be needed to cover the entire address space. as we see in the images as table size increases the address space grows because more pages are needed to cover the address space.



b) How should the page table resize as the page size increases?
Answer: When page size increases page table decreases because fewer pages are needed to cover the address space.

c) Why shouldn't we use really big pages in general?
Answer: Since most processes use little memory, it is a bad idea to use really large pages because it would waste a lot of memory. Big pages should not be used in general because they cause lots of internal fragmentation and lots of waste of memory.

Question 2:
What happens as we increase the percentage of pages that are allocated in each address space?
Answer: With the increase in the percentage of allocated pages in each address space, more memory access operations would be valid. AS the percentage of pages that are allocated or usage of address space is increased more and more memory access operations become valid and free space decreases, we can see the changes in the attached ss too.

Question 3:
Which of these parameter combinations are not realistic? Why?

i) -P 8 -a 32 -p 1024 -v -s 1
ii) -P 8k -a 32k -p 1m -v -s 2
iii) -P 1m -a 256m -p 512m -v -s 3

Answer: i) The sizes are too small, with 8-byte pages there would be more overhead in the page table than usable memory, we usually use pages of at least 512 bytes. after seeing the outputs above we can see that the 1st variety is very unrealistic because of its small size and thus it is very unrealistic to have some combinations.

Question 4:
What are the limitations of the given program?
Answer: The program will not work if:

- The size of the address space is larger than the physical memory.
- Physical memory, address space size, and page size receives negative values.
- The page size is larger than the address space size.
- Address space size and page size must be powers of 2.

COMMANDS

```
cd HW-Paging-LinearTranslate/
# 1
python2 paging-linear-translate.py -P 1k -a 1m -p 512m -v -n 0
python2 paging-linear-translate.py -P 1k -a 2m -p 512m -v -n 0
python2 paging-linear-translate.py -P 1k -a 4m -p 512m -v -n 0
python2 paging-linear-translate.py -P 1k -a 1m -p 512m -v -n 0
python2 paging-linear-translate.py -P 2k -a 1m -p 512m -v -n 0
python2 paging-linear-translate.py -P 4k -a 1m -p 512m -v -n 0

# 2
python2 paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 0
python2 paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 25
python2 paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 50
python2 paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 75
python2 paging-linear-translate.py -P 1k -a 16k -p 32k -v -u 100

# 3
python2 paging-linear-translate.py -P 8 -a 32 -p 1024 -v -s 1
python2 paging-linear-translate.py -P 8k -a 32k -p 1m -v -s 2
python2 paging-linear-translate.py -P 1m -a 256m -p 512m -v -s 3

cd ..
```

Here are some commands… to see output run ./run.sh > output.txt

MULTILEVEL PAGE TABLE

Question 1:

How many records do we need to locate a two-level page table and a three-level page table?

Answer: Even for two-level and three-level page tables, the number of records required would still be 1. The hardware acquires the address from the table on the next page after it has been indexed into the table on the first page and the input is valid. So we can overwrite the single record, having one record for each page table would not be efficient. for the number of registers needed for both of the cases will still be 1. Because once the hardware has indexed into the first-page table it obtains the address of the nest page-table(assuming the first is a valid entry). Requiring a register for each possible page table in a 32-bit virtual address space would be very impractical and can't be justified.

Question 2:

How many memory references are needed to perform each search?

Answer: In the screenshot above, we translated the corresponding physical address to the virtual address (for VA: 611c) provided. We know from README that the virtual address needs 15 bits, of which 5 bits are for offset and 10 for VPN, and the physical address requires 12 bits, 5 for offset, and 7 for PFN. Also, the top 5 bits of the virtual address are used to index into a page directory, the page directory entry (PDE), if valid, points to a page in the page table. Each page in the page table contains 32-page table entries (PTE). Each PTE, if valid, contains the desired translation (physical frame number or PFN) of the virtual page in question. With this information, we manually translate a virtual address and compare it with the real response (Figure below).

A total of 3 memory references are needed to perform each search if the valid bit is 1 for each level, one for PDE, then for PTE, and the last to get the value.

Question 3:

How do you think the memory references to the cached page table will behave? Will they lead to a lot of cache hits (and therefore fast access) or a lot of misses (and therefore slow access)?

Answer: It should result in a lot of cache hits due to spatial and temporal characteristics location. The addresses in these exercises appear to be random, so the miss rate would be quite high. In the real world I'd expect a very high hit rate due to temporal and spatial locality (indeed, otherwise the TLB would be a useless idea!).

Note: All program outputs are included in their respective folder with question numbers as file names.

COMMANDS

```
cd HW-Paging-MultiLevelTranslate/
# 2
python2 paging-multilevel-translate.py -s 0 -n 4 | grep "page 108"
```

Here are some commands… to see output run ./run.sh > output.txt

XXX