

**Lab Task:**

In this part, you will use a simple program, which is known as paging-linear-translate.py, to see if you understand how simple virtual-to-physical address translation works with linear page tables. See the README for details.

**Tasks:**

1. Before doing any translations, let’s use the simulator 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?

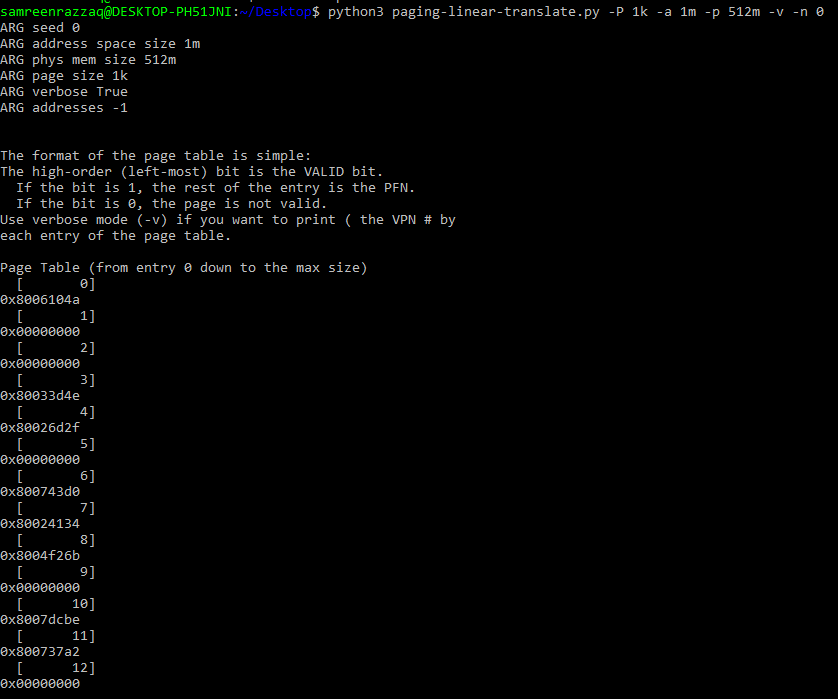
When we increase the address space size, the page grows more and more.

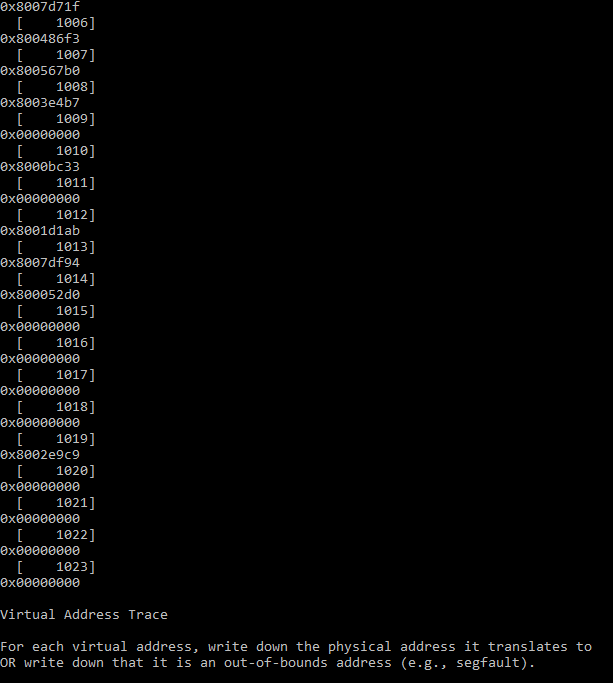
Firstly, it grows 1023 times, then 2047 times and lastly, it grows 4095 times.

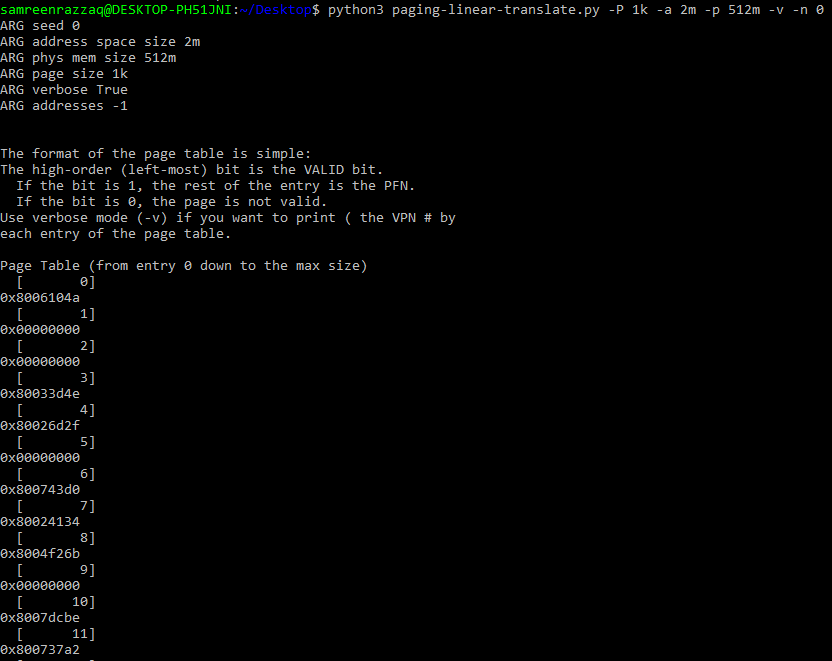
And big pages not use in general because it causes internal fragmentation and it is less flexible in term of memory allocation.

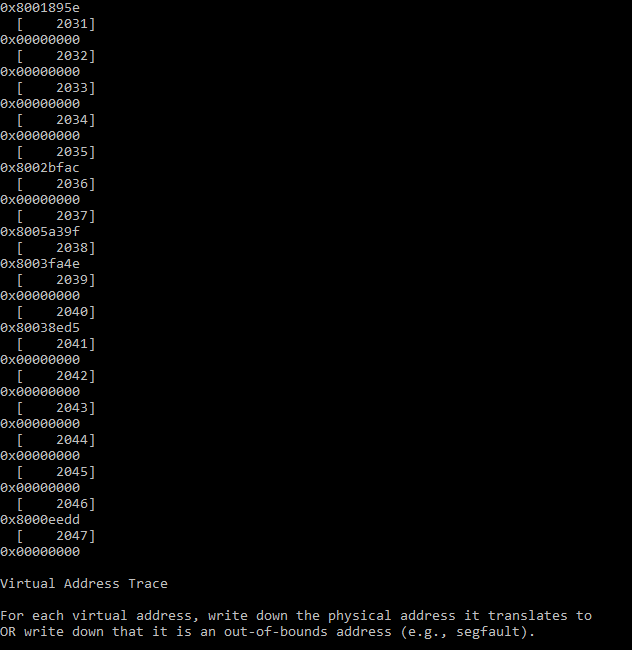
When we increase page size (-P), the page growth is become lesser and lesser. Firstly, it is 1023, then 511 and lastly, it is 255.

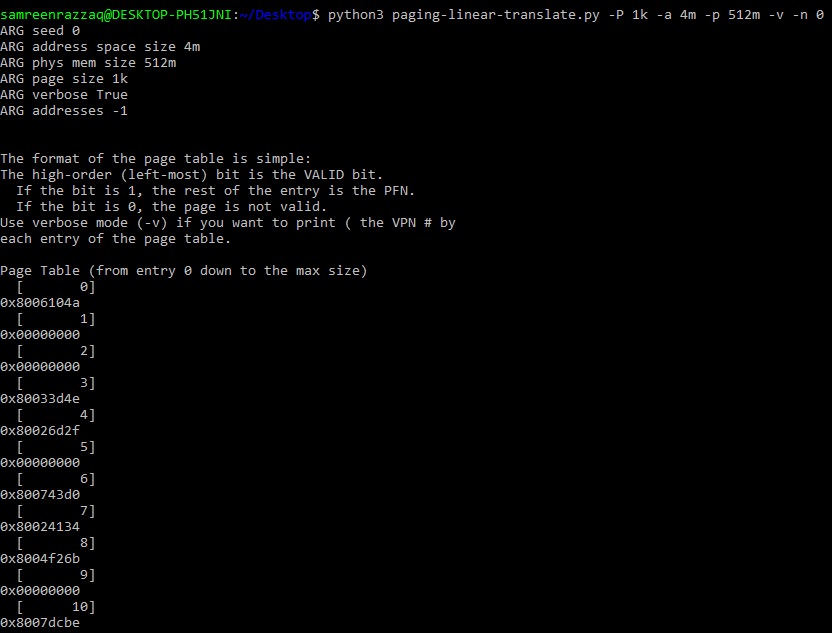
* -P 1k -a 1m -p 512m -v -n 0:

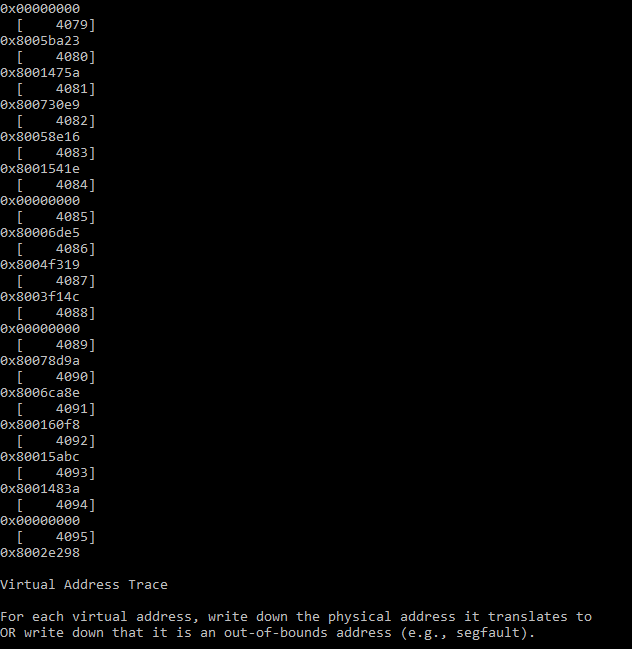


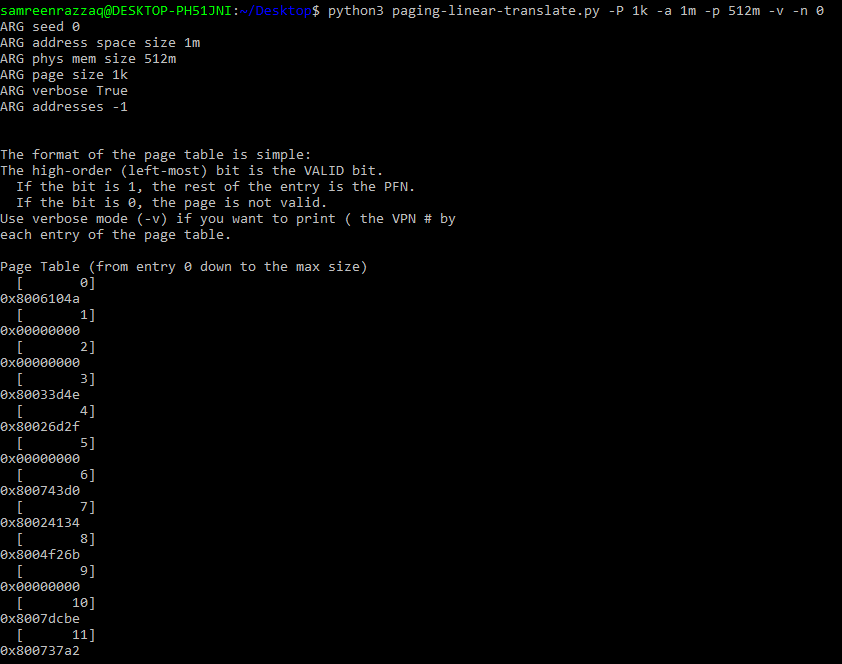


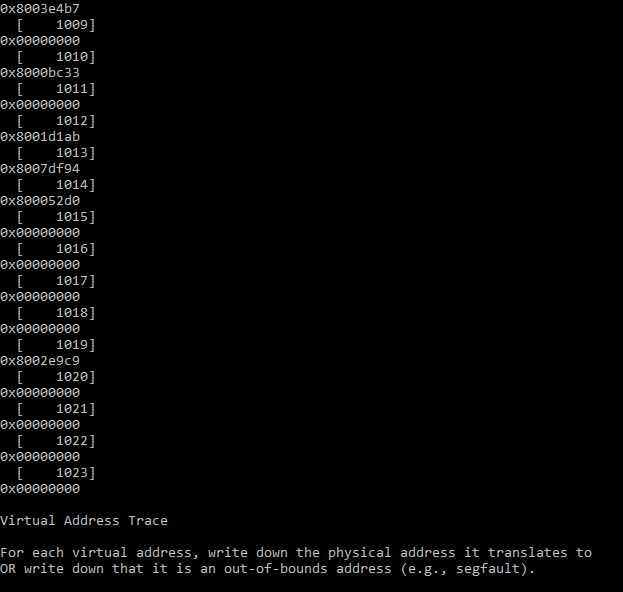
* -P 1k -a 2m -p 512m -v -n 0:

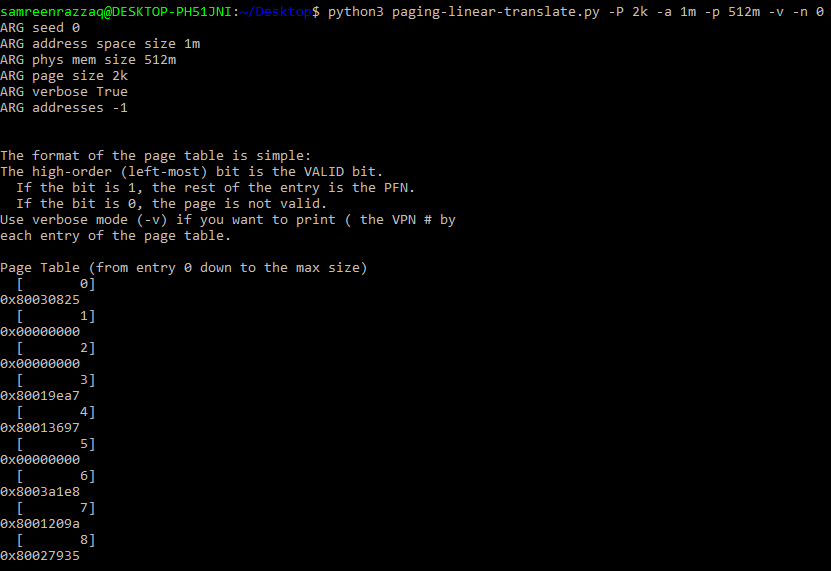


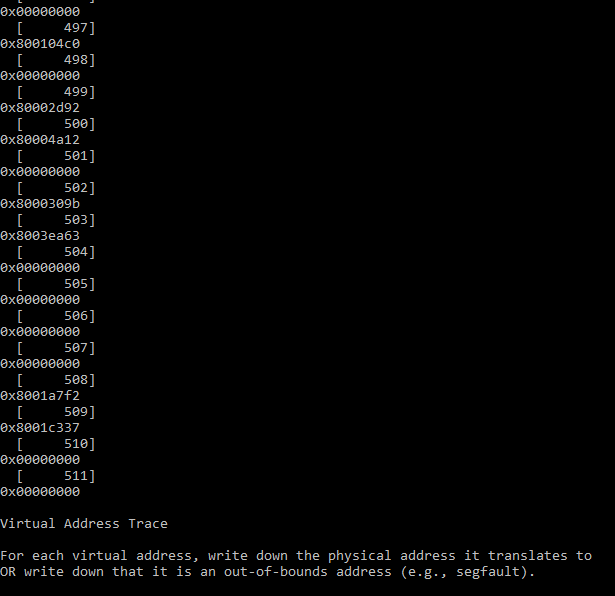
* -P 1k -a 4m -p 512m -v -n 0:



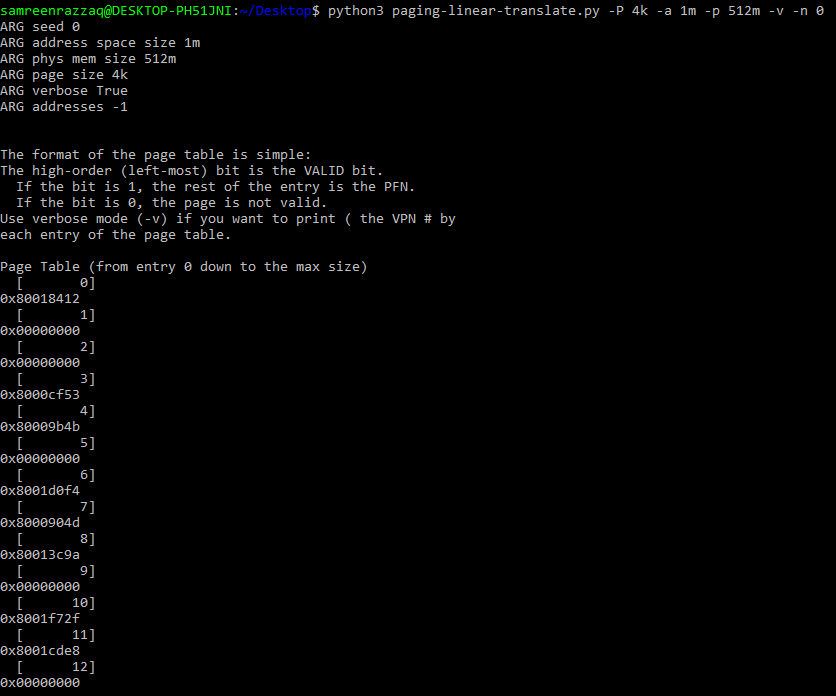
* -P 1k -a 1m -p 512m -v -n 0:

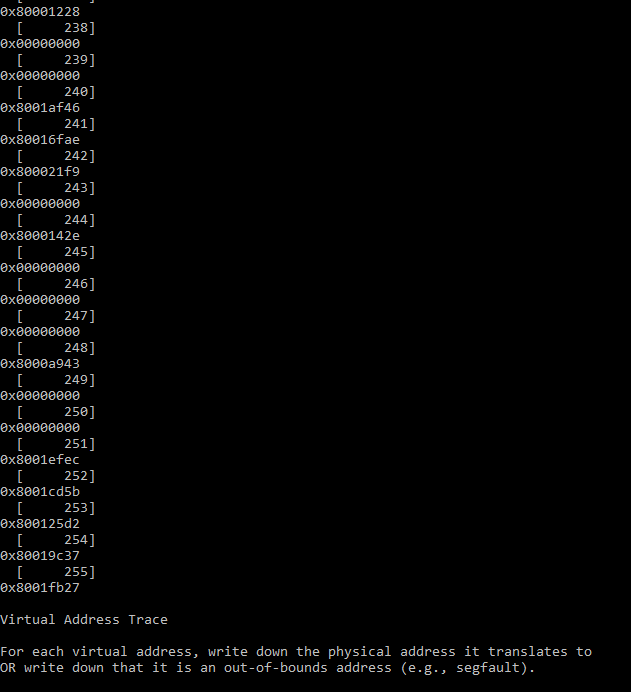


* -P 2k -a 1m -p 512m -v -n 0:



* -P 4k -a 1m -p 512m -v -n 0:





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

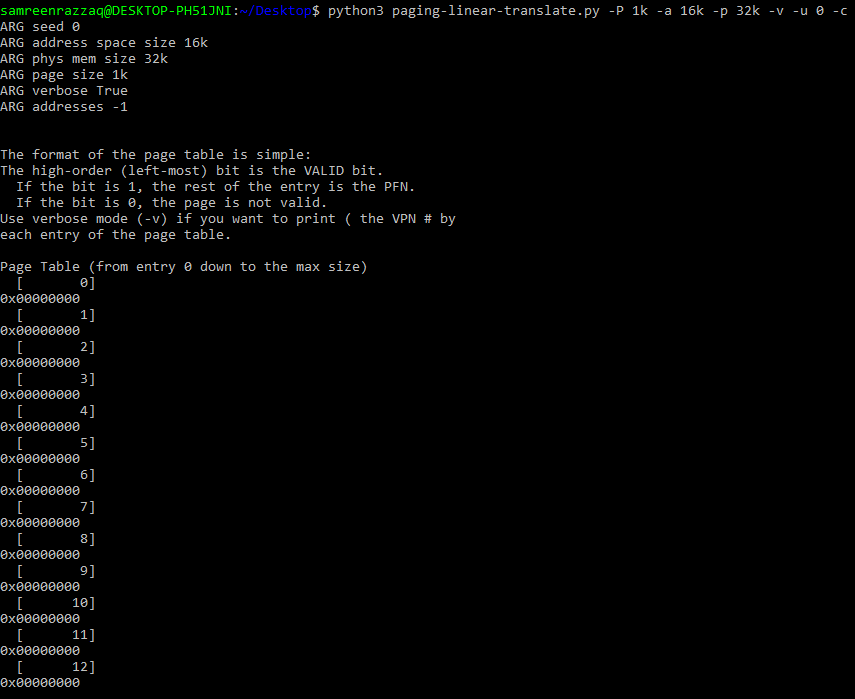
-P 1k -a 16k -p 32k -v -u 75

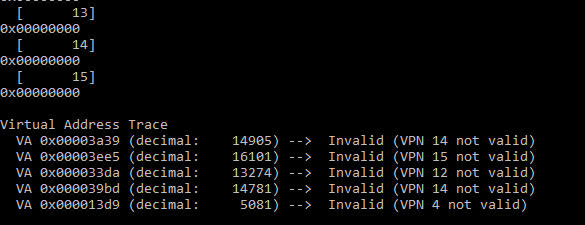
-P 1k -a 16k -p 32k -v -u 100

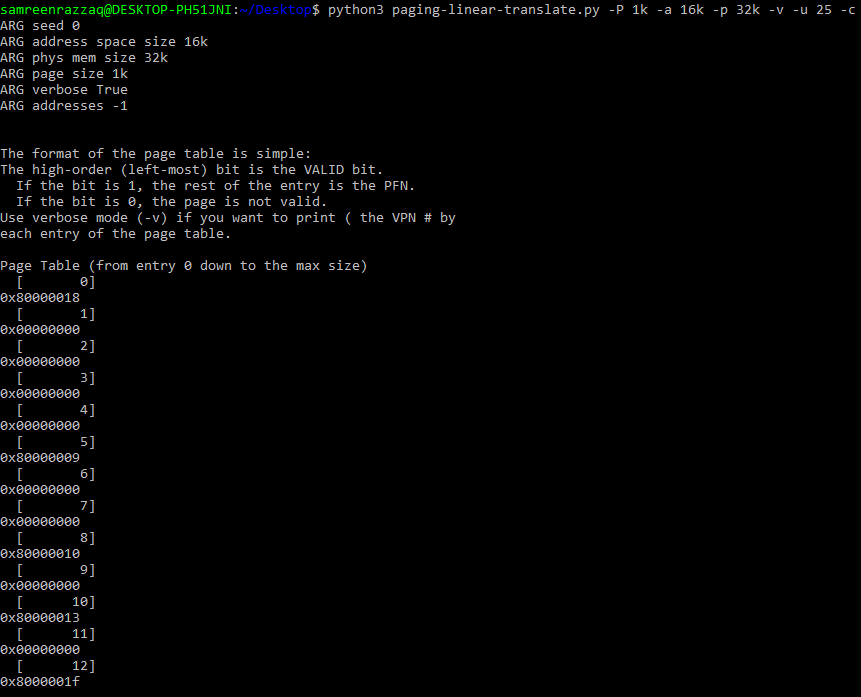
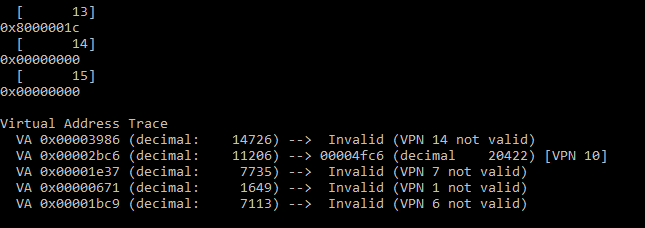
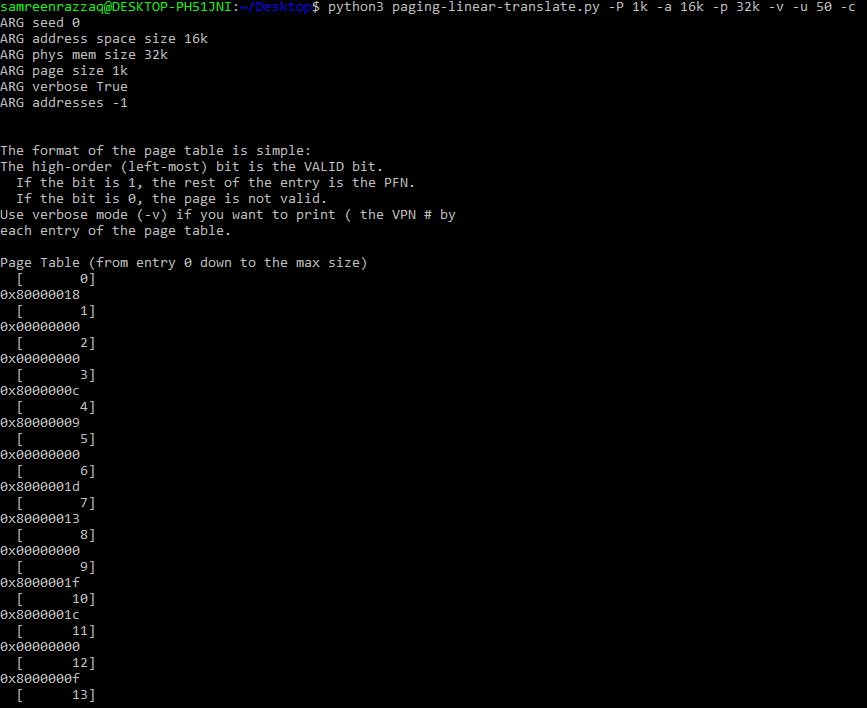
What happens as you increase the percentage of pages that are allocated in each address space?

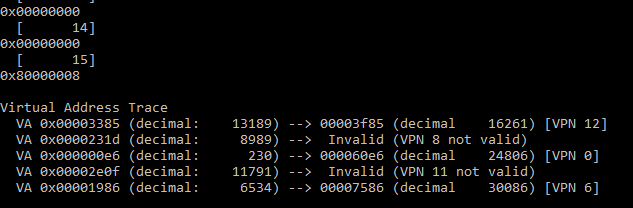
As you increase the percentages of pages allocated to an address space, you are effectively increasing the memory usage for each address space and it consumes more memory and it also may lead to potential fragmentation issues.

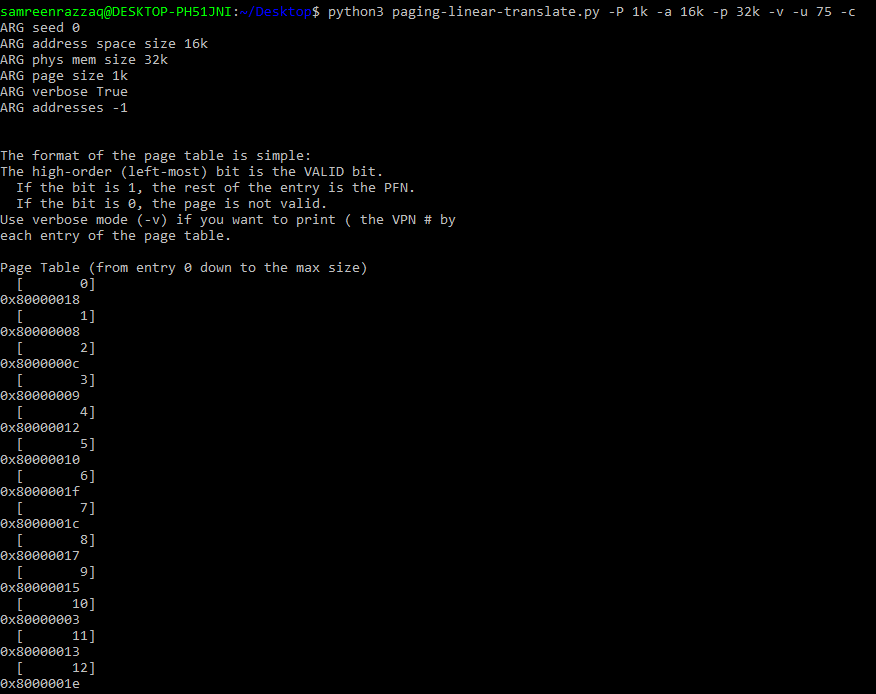
* -P 1k -a 16k -p 32k -v -u 0 -c:

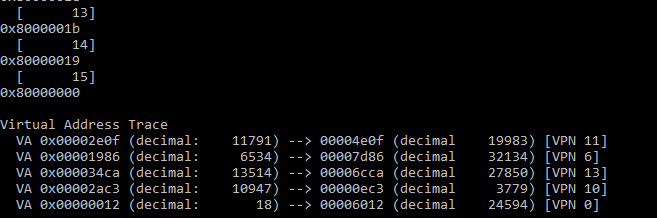


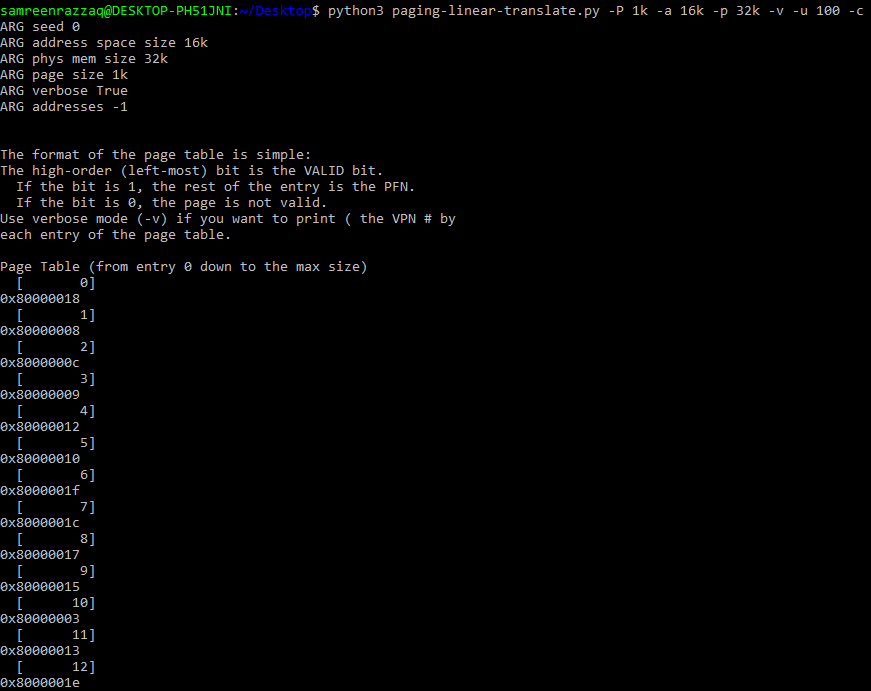


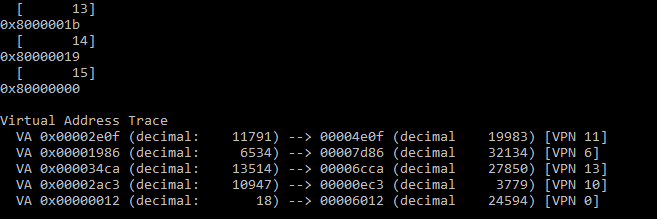
* -P 1k -a 16k -p 32k -v -u 25 -c :
* -P 1k -a 16k -p 32k -v -u 50 -c:

-P 1k -a 16k -p 32k -v -u 75 -c:



* -P 1k -a 16k -p 32k -v -u 100 :





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

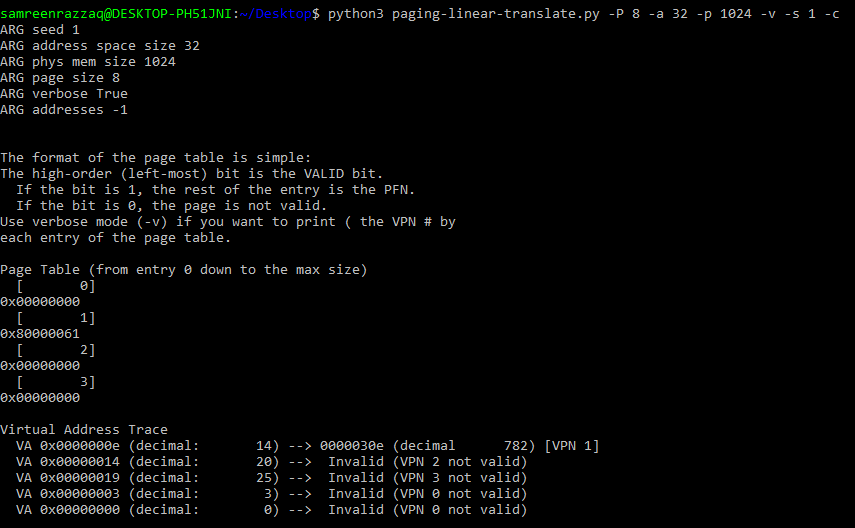
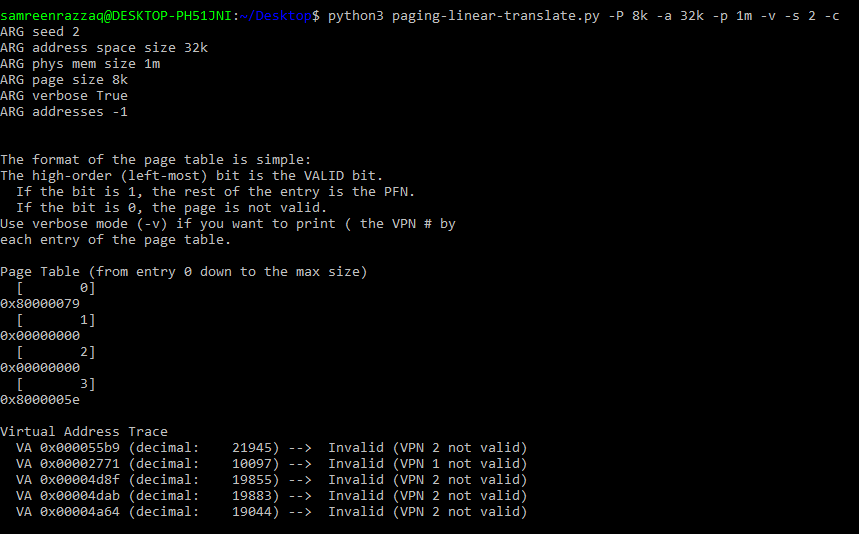
-P 8 -a 32 -p 1024 -v -s 1

-P 8k -a 32k -p 1m -v -s 2

-P 1m -a 256m -p 512m -v -s 3

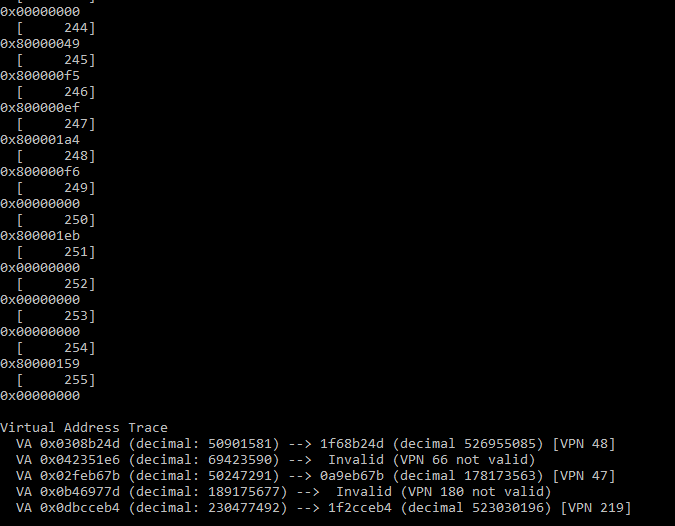
Which of these parameter combinations are unrealistic? Why?

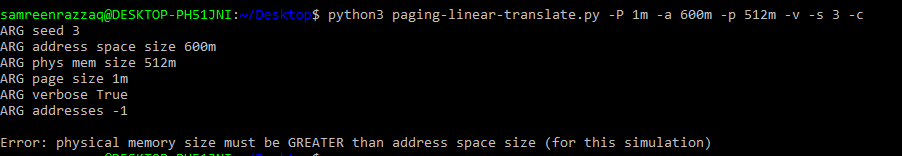
In the first parameter, its combination with extremely small sizes for page size, address space, and process size is unrealistic. The other two combinations have more reasonable sizes, although the specific values might still not align with common real-world scenarios, but they are not as extreme as the first one. Realistic values for these parameters will depend on the context and the specific hardware and software being used.

* -P 8 -a 32 -p 1024 -v -s 1 -c:
* -P 8k -a 32k -p 1m -v -s 2 -c: 

* -P 1m -a 256m -p 512m -v -s 3 -c:





4. Use the program to try out some other problems. Can you find the limits of where the program doesn’t work anymore? For example, what happens if the address-space size is bigger than physical memory?

We see that when we increase the address space size from physical memory than we get an error.