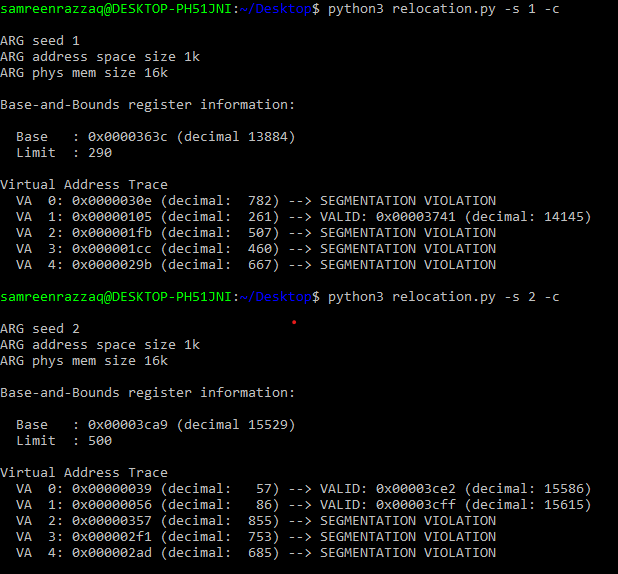


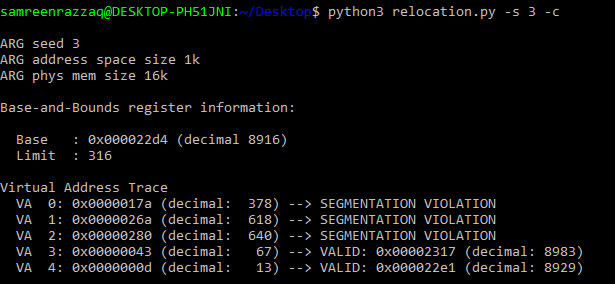
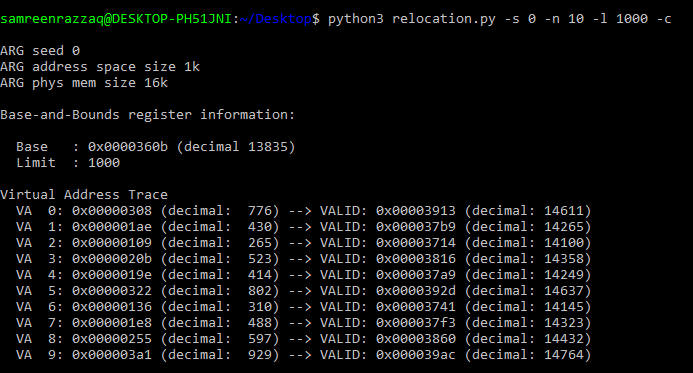
**Lab Task:**

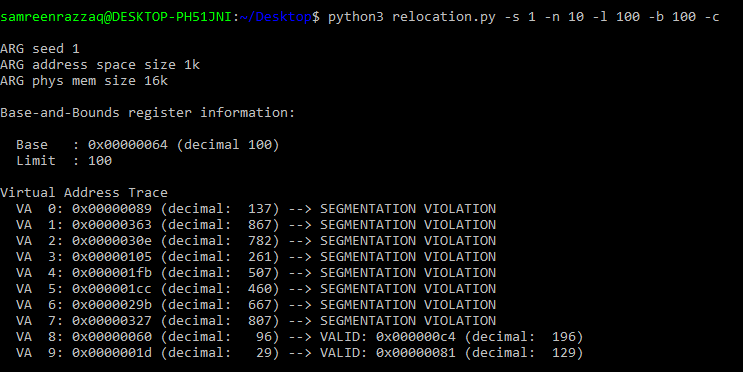
Let’s start with the program relocation.py that allows you to see how address translations are performed in a system with base and bounds registers. See the README for details and perform the following tasks.

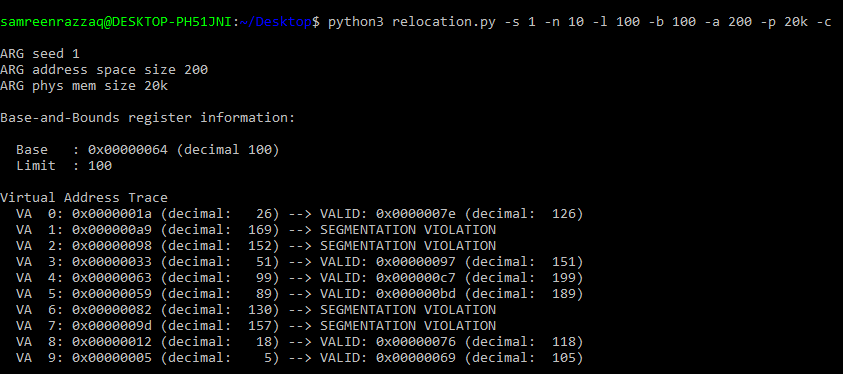
**Tasks:**

1. Run with seeds 1, 2, and 3, and compute whether each virtual address generated by the process is in or out of bounds. If in bounds, compute the translation.

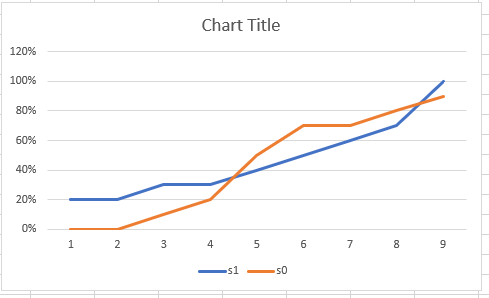


2. Run with these flags: -s 0 -n 10. What value do you have set -l (the bounds register) to in order to ensure that all the generated virtual addresses are within bounds?

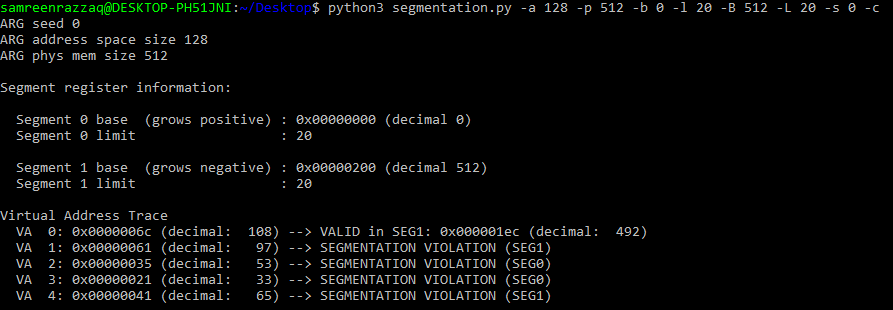
3. Run with these flags: -s 1 -n 10 -l 100. What is the maximum value that base can be set to, such that the address space still fits into physical memory in its entirety?

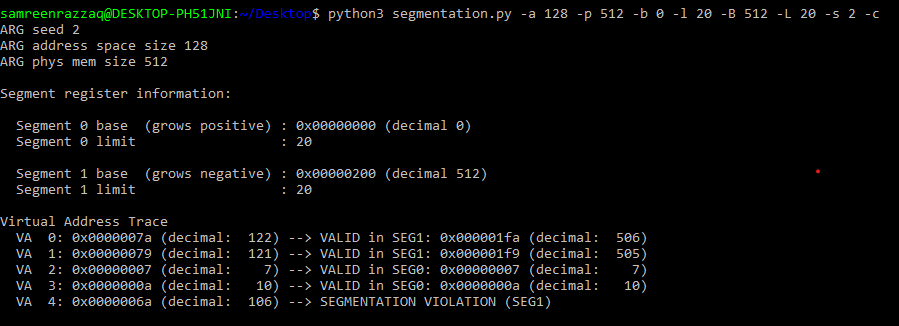
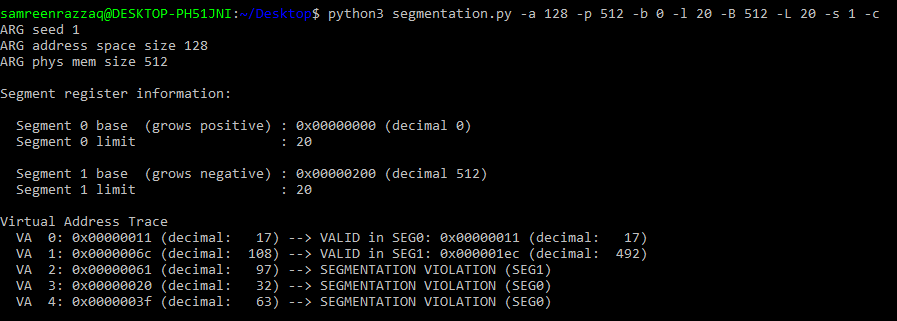
4. Run some of the same problems above, but with larger address spaces (-a) and physical memories (-p).

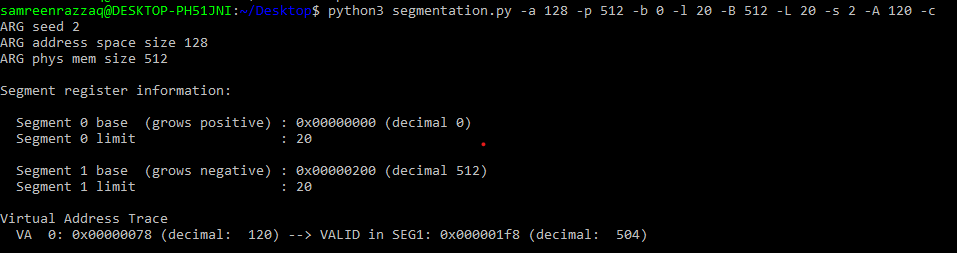
5. What fraction of randomly-generated virtual addresses are valid, as a function of the value of the bounds register? Make a graph from running with different random seeds, with limit values ranging from 0 up to the maximum size of the address space Let’s work with segmentation.py. This program allows you to see how address translations are performed in a system with segmentation. See the README for details and perform the following tasks.



6. First let’s use a tiny address space to translate some addresses. Here’s a simple set of parameters with a few different random seeds; can you translate the addresses?

segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 1 segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 2



7. Now, let’s see if we understand this tiny address space we’ve constructed (using the parameters from the question above). What is the highest legal virtual address in segment 0? What about the lowest legal virtual address in segment 1? What are the lowest and highest illegal addresses in this entire address space? Finally, how would you run segmentation.py with the -A flag to test if you are right?

8. Let’s say we have a tiny 16-byte address space in a 128-byte physical memory. What base and bounds would you set up so as to get the simulator to generate the following translation results for the specified address stream: valid, valid, violation, ..., violation, valid, valid? Assume the following parameters:

segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 --b0 ? --l0 ? --b1 ? --l1 ?

9. Assume we want to generate a problem where roughly 90% of the randomly-generated virtual addresses are valid (not segmentation violations). How should you configure the simulator to do so? Which parameters are important to getting this outcome?

10. Can you run the simulator such that no virtual addresses are valid? How?