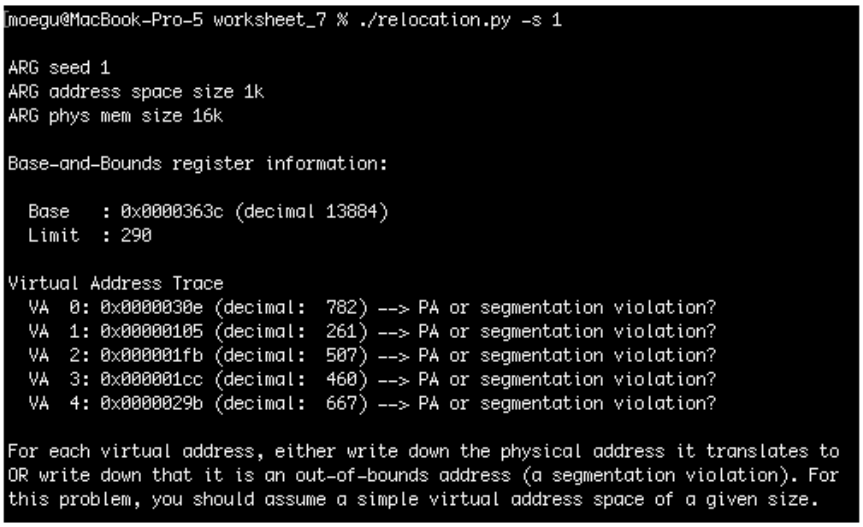
CSC 369 Worksheet 7 Solution

August 23, 2020

1. First, I need to run the program with seeds 1,2, and 3, and compute whether each virtual address generated by the process is in or out of bounds.

   By running the following commands, we see the corresponding base bounds register information (see images under each command).

   - `./relocation.py -s 1`



   Here, the following is inbound:

   – VA 1: 0x00000105 (decimal: 261) –¿ PA 0x00003741 (decimal 14145)

   and the following is out of bound:

   – VA 0: 0x0000030e (decimal: 782)
   – VA 2: 0x000001fb (decimal: 507)
   – VA 3: 0x000001cc (decimal: 460)

  - VA 4: 0x0000029b (decimal: 667)

- `./relocation.py -s 2`

```
moegu@MacBook-Pro-5 worksheet_7 % ./relocation.py -s 2

ARG seed 2
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00003ca9 (decimal 15529)
  Limit  : 500

Virtual Address Trace
  VA  0: 0x00000039 (decimal:   57) --> PA or segmentation violation?
  VA  1: 0x00000056 (decimal:   86) --> PA or segmentation violation?
  VA  2: 0x00000357 (decimal:  855) --> PA or segmentation violation?
  VA  3: 0x000002f1 (decimal:  753) --> PA or segmentation violation?
  VA  4: 0x000002ad (decimal:  685) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple virtual address space of a given size.
```

  Here, the following is inbound:

  - VA 0: 0x00000039 (decimal: 57) –¿ PA 0x00003CE2 (decimal 15586)
  - VA 1: 0x00000056 (decimal: 86) –¿ PA 0x00003CFF (decimal 15615)

  and the following is out of bound:

  - VA 2: 0x00000357 (decimal: 855)
  - VA 3: 0x000002f1 (decimal: 753)
  - VA 4: 0x000002ad (decimal: 685)

- `./relocation.py -s 3`

```
moegu@MacBook-Pro-5 worksheet_7 % ./relocation.py -s 3

ARG seed 3
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x000022d4 (decimal 8916)
  Limit  : 316

Virtual Address Trace
  VA  0: 0x0000017a (decimal:  378) --> PA or segmentation violation?
  VA  1: 0x0000026a (decimal:  618) --> PA or segmentation violation?
  VA  2: 0x00000280 (decimal:  640) --> PA or segmentation violation?
  VA  3: 0x00000043 (decimal:   67) --> PA or segmentation violation?
  VA  4: 0x0000000d (decimal:   13) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple virtual address space of a given size.
```

Here, the following is inbound:

- VA 3: 0x00000043 (decimal: 67) –¿ PA 0x00002317 (decimal 8983)
- VA 4: 0x0000000d (decimal: 13) –¿ PA 0x000022E1 (decimal 8929)

and the following is out of bound:

- VA 0: 0x0000017a (decimal: 378)
- VA 1: 0x0000026a (decimal: 618)
- VA 2: 0x00000280 (decimal: 640)

Second, for the ones that are in bounds, I need to calculate the translation.

Notes

1. Decimal to hexadecimal link here
2. The prefix `0x` in `0x00003ca9` is to indicate that the number is written in hex
3. Simulation is run with command
4. Reality → many programs share memory at the same time
5. Virtualization → creates illution that program has its own private memory, where its own code and data inside
6. Virtualization allows us turn into something useful, powerful and easy to use
7. **Dynamic (Hardware-based) Relocation**
   -