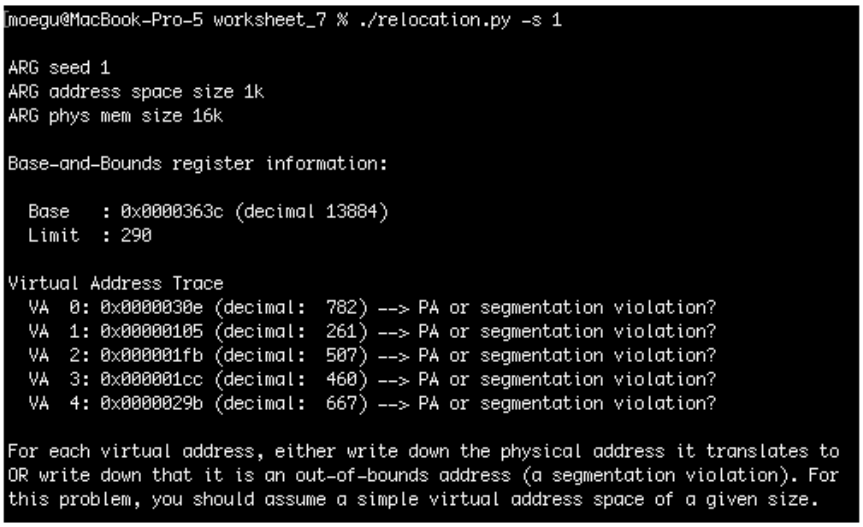
CSC 369 Worksheet 7 Solution

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

2. I need to run the command `./relocation.py -s 0 -n 10`, and determine which value of `-l` covers all the genereated virtual addresses

Running the command, I see

```
moegu@MacBook-Pro-5 worksheet_7 % ./relocation.py -s 0 -n 10

ARG seed 0
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00003082 (decimal 12418)
  Limit  : 472

Virtual Address Trace
  VA  0: 0x000001ae (decimal:  430) --> PA or segmentation violation?
  VA  1: 0x00000109 (decimal:  265) --> PA or segmentation violation?
  VA  2: 0x0000020b (decimal:  523) --> PA or segmentation violation?
  VA  3: 0x0000019e (decimal:  414) --> PA or segmentation violation?
  VA  4: 0x00000322 (decimal:  802) --> PA or segmentation violation?
  VA  5: 0x00000136 (decimal:  310) --> PA or segmentation violation?
  VA  6: 0x000001e8 (decimal:  488) --> PA or segmentation violation?
  VA  7: 0x00000255 (decimal:  597) --> PA or segmentation violation?
  VA  8: 0x000003a1 (decimal:  929) --> PA or segmentation violation?
  VA  9: 0x00000204 (decimal:  516) --> 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.
```

And the minimum value that covers all generated virtual addreses are: 930.

3. I need to run the command `./relocation.py -s 1 -n 10 -l 100` and write the maximum value of base such that all the address space fits into physical memory.

   Running the command, I see that

```
moegu@MacBook-Pro-5 worksheet_7 % ./relocation.py -s 1 -n 10 -l 100

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

Base-and-Bounds register information:

  Base   : 0x00000899 (decimal 2201)
  Limit  : 100

Virtual Address Trace
  VA  0: 0x00000363 (decimal:  867) --> PA or segmentation violation?
  VA  1: 0x0000030e (decimal:  782) --> PA or segmentation violation?
  VA  2: 0x00000105 (decimal:  261) --> PA or segmentation violation?
  VA  3: 0x000001fb (decimal:  507) --> PA or segmentation violation?
  VA  4: 0x000001cc (decimal:  460) --> PA or segmentation violation?
  VA  5: 0x0000029b (decimal:  667) --> PA or segmentation violation?
  VA  6: 0x00000327 (decimal:  807) --> PA or segmentation violation?
  VA  7: 0x00000060 (decimal:   96) --> PA or segmentation violation?
  VA  8: 0x0000001d (decimal:   29) --> PA or segmentation violation?
  VA  9: 0x00000357 (decimal:  855) --> 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.
```

   And the maximum value of base is:

$$\begin{aligned} \texttt{base} &= \texttt{physical address} - \texttt{virtual address} && (1)\\ &= 16000 - 100 && (2)\\ &= 15900 \texttt{ or } \texttt{15.9k} && (3) \end{aligned}$$

Correct Solution

I need to run the command `./relocation.py -s 1 -n 10 -l 100` and write the maximum value of base such that all the address space fits into physical memory.

Running the command, I see that

```
moegu@MacBook-Pro-5 worksheet_7 % ./relocation.py -s 1 -n 10 -l 100

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

Base-and-Bounds register information:

  Base   : 0x00000899 (decimal 2201)
  Limit  : 100

Virtual Address Trace
  VA  0: 0x00000363 (decimal:  867) --> PA or segmentation violation?
  VA  1: 0x0000030e (decimal:  782) --> PA or segmentation violation?
  VA  2: 0x00000105 (decimal:  261) --> PA or segmentation violation?
  VA  3: 0x000001fb (decimal:  507) --> PA or segmentation violation?
  VA  4: 0x000001cc (decimal:  460) --> PA or segmentation violation?
  VA  5: 0x0000029b (decimal:  667) --> PA or segmentation violation?
  VA  6: 0x00000327 (decimal:  807) --> PA or segmentation violation?
  VA  7: 0x00000060 (decimal:   96) --> PA or segmentation violation?
  VA  8: 0x0000001d (decimal:   29) --> PA or segmentation violation?
  VA  9: 0x00000357 (decimal:  855) --> 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.
```

And the maximum value of base is:

$$\begin{aligned} \texttt{base} &= \texttt{physical address} - \texttt{virtual address} && (4)\\ &= 16000 - 1000 && (5)\\ &= 15000 \text{ or } 15k && (6) \end{aligned}$$

Notes

- **Dynamic (Hardware-based) Relocation**
  - **Formula:** `physical addresss = virtual address + base`
- **Address space**
  - Is a range of valid addresses in memory that are available for a program or process.

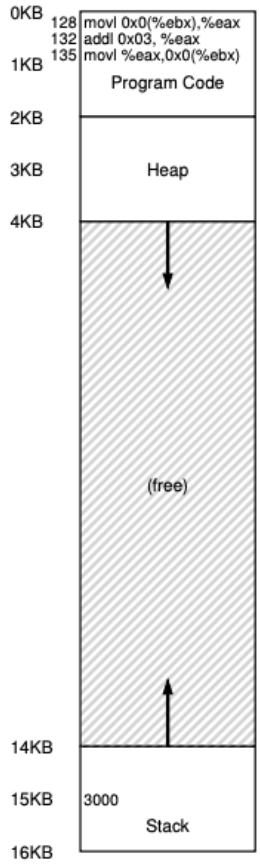


Figure 15.1: **A Process And Its Address Space**

4. I need to run the same problem as above, but with larger address space `-a` and physical memories `-p`.

   Running the command `./relocation.py -s 1 -n 10 -l 100 -a 2k -p 20k`, we see the following

```
moegu@MacBook-Pro-5 worksheet_7 % ./relocation.py -s 1 -n 10 -l 100 -a 2k -p 20k

ARG seed 1
ARG address space size 2k
ARG phys mem size 20k

Base-and-Bounds register information:

  Base   : 0x00000abf (decimal 2751)
  Limit  : 100

Virtual Address Trace
  VA  0: 0x000006c7 (decimal: 1735) --> PA or segmentation violation?
  VA  1: 0x0000061c (decimal: 1564) --> PA or segmentation violation?
  VA  2: 0x0000020a (decimal:  522) --> PA or segmentation violation?
  VA  3: 0x000003f6 (decimal: 1014) --> PA or segmentation violation?
  VA  4: 0x00000398 (decimal:  920) --> PA or segmentation violation?
  VA  5: 0x00000536 (decimal: 1334) --> PA or segmentation violation?
  VA  6: 0x0000064f (decimal: 1615) --> PA or segmentation violation?
  VA  7: 0x000000c0 (decimal:  192) --> PA or segmentation violation?
  VA  8: 0x0000003a (decimal:   58) --> PA or segmentation violation?
  VA  9: 0x000006af (decimal: 1711) --> 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.
```

Using this information, the maximum value of base is:

$$\begin{aligned} \texttt{base} &= \texttt{physical address} - \texttt{virtual address} && (7)\\ &= 20000 - 2000 && (8)\\ &= 18000 \texttt{ or } \texttt{18k} && (9) \end{aligned}$$

5. I need to make a graph concerning what fraction of virtual addresses are valid with graph running with different random seeds with limits ranging from 0 to maximum size of the address space.

   Using the following commands, we can write

   - `./relocation.py -s 1 -n 10 -a 500 -l 0` [Valid Value: 0]
   - `./relocation.py -s 2 -n 10 -a 500 -l 0` [Valid Value: 0]
   - `./relocation.py -s 3 -n 10 -a 500 -l 0` [Valid Value: 0]
   - `./relocation.py -s 1 -n 10 -a 500 -l 100` [Valid Value: 0.2]
   - `./relocation.py -s 2 -n 10 -a 500 -l 100` [Valid Value: 0.2]
   - `./relocation.py -s 3 -n 10 -a 500 -l 100` [Valid Value: 0.2]
   - `./relocation.py -s 1 -n 10 -a 500 -l 200` [Valid Value: 0.3333]
   - `./relocation.py -s 2 -n 10 -a 500 -l 200` [Valid Value: 0.3333]

- `./relocation.py -s 3 -n 10 -a 500 -l 200` [Valid Value: 0.5]
- `./relocation.py -s 1 -n 10 -a 500 -l 300` [Valid Value: 0.5]
- `./relocation.py -s 2 -n 10 -a 500 -l 300` [Valid Value: 0.4]
- `./relocation.py -s 3 -n 10 -a 500 -l 300` [Valid Value: 0.6]
- `./relocation.py -s 1 -n 10 -a 500 -l 400` [Valid Value: 0.8]
- `./relocation.py -s 2 -n 10 -a 500 -l 400` [Valid Value: 0.8]
- `./relocation.py -s 3 -n 10 -a 500 -l 400` [Valid Value: 0.8]
- `./relocation.py -s 1 -n 10 -a 500 -l 500` [Valid Value: 1]
- `./relocation.py -s 2 -n 10 -a 500 -l 500` [Valid Value: 1]
- `./relocation.py -s 3 -n 10 -a 500 -l 500` [Valid Value: 1]

Using above information, we have