

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.

```
vladb@VladB ~/G/h/S/B/H/HW15-relocation (master)> ./relocation.py -s 1 -c

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

Base-and-Bounds register information:

Base   : 0x0000363c (decimal 13884)
Limit  : 290

Virtual Address Trace
VA 0: 0x0000030e (decimal: 782) --> SEGMENTATION VIOLATION
VA 1: 0x00000105 (decimal: 261) --> VALID: 0x00003741 (decimal: 14145)
VA 2: 0x000001fb (decimal: 507) --> SEGMENTATION VIOLATION
VA 3: 0x000001cc (decimal: 460) --> SEGMENTATION VIOLATION
VA 4: 0x0000029b (decimal: 667) --> SEGMENTATION VIOLATION
```

- Segmentation violation
- valid; 14145 (base + size virtuelle adresse)
- Segmentation violation
- Segmentation violation
- Segmentation violation

```
vladb@VladB ~/G/h/S/B/H/HW15-relocation (master)> ./relocation.py -s 2 -c

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) --> VALID: 0x00003ce2 (decimal: 15586)
VA 1: 0x00000056 (decimal: 86) --> VALID: 0x00003cff (decimal: 15615)
VA 2: 0x00000357 (decimal: 855) --> SEGMENTATION VIOLATION
VA 3: 0x000002f1 (decimal: 753) --> SEGMENTATION VIOLATION
VA 4: 0x000002ad (decimal: 685) --> SEGMENTATION VIOLATION
```

- valid; 15586
- valid; 15615

- segmentation violation
- segmentation violation
- segmentation violation

- 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?

```
vladb@VladB ~/G/h/S/B/H/HW15-relocation (master)> ./relocation.py -s 0 -n 10 -l 930 -c

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

Base-and-Bounds register information:

  Base   : 0x0000360b (decimal 13835)
  Limit  : 930

Virtual Address Trace
VA 0: 0x00000308 (decimal: 776) --> VALID: 0x00003913 (decimal: 14611)
VA 1: 0x000001ae (decimal: 430) --> VALID: 0x000037b9 (decimal: 14265)
VA 2: 0x00000109 (decimal: 265) --> VALID: 0x00003714 (decimal: 14100)
VA 3: 0x0000020b (decimal: 523) --> VALID: 0x00003816 (decimal: 14358)
VA 4: 0x0000019e (decimal: 414) --> VALID: 0x000037a9 (decimal: 14249)
VA 5: 0x00000322 (decimal: 802) --> VALID: 0x0000392d (decimal: 14637)
VA 6: 0x00000136 (decimal: 310) --> VALID: 0x00003741 (decimal: 14145)
VA 7: 0x000001e8 (decimal: 488) --> VALID: 0x000037f3 (decimal: 14323)
VA 8: 0x00000255 (decimal: 597) --> VALID: 0x00003860 (decimal: 14432)
VA 9: 0x000003a1 (decimal: 929) --> VALID: 0x000039ac (decimal: 14764)
```

- Base muss 930 sein, also Prozess mit größtem size + 1

- 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?

```
vladb@VladB ~/G/h/S/B/H/HW15-relocation (master)> ./relocation.py -s 1 -n 10 -l 100 -c

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) --> SEGMENTATION VIOLATION
VA 1: 0x0000030e (decimal: 782) --> SEGMENTATION VIOLATION
VA 2: 0x00000105 (decimal: 261) --> SEGMENTATION VIOLATION
VA 3: 0x000001fb (decimal: 507) --> SEGMENTATION VIOLATION
VA 4: 0x000001cc (decimal: 460) --> SEGMENTATION VIOLATION
VA 5: 0x0000029b (decimal: 667) --> SEGMENTATION VIOLATION
VA 6: 0x00000327 (decimal: 807) --> SEGMENTATION VIOLATION
VA 7: 0x00000060 (decimal: 96) --> VALID: 0x000008f9 (decimal: 2297)
VA 8: 0x0000001d (decimal: 29) --> VALID: 0x000008b6 (decimal: 2230)
VA 9: 0x00000357 (decimal: 855) --> SEGMENTATION VIOLATION
```

- Frage ist, base so maximal ändern, damit die zwei die valid sind, danach immer noch valid sind
- physischen Speicher ist $16 * 1024 \text{ Byte} = 16384 \text{ Byte}$
- $16384 - 100 = 16284$ → hier ist ab wo der erste Speicher anfangen soll

Maximales base = physical memory - limit

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

Limit so auswählen, dass alle virtuellen Adressen valid werden.

```
vladb@VladB ~/G/h/S/B/H/HW15-relocation (master)>
./relocation.py -s 0 -n 10 -a 7k -p 20k -l 6510 -c

ARG seed 0
ARG address space size 7k
ARG phys mem size 20k

Base-and-Bounds register information:

  Base   : 0x000021a5 (decimal 8613)
  Limit  : 6510

Virtual Address Trace
VA  0: 0x0000073f (decimal: 1855) --> VALID: 0x000028e4 (decimal: 10468)
VA  1: 0x00000e50 (decimal: 3664) --> VALID: 0x00002ff5 (decimal: 12277)
VA  2: 0x00000b56 (decimal: 2902) --> VALID: 0x00002cfb (decimal: 11515)
VA  3: 0x000015f2 (decimal: 5618) --> VALID: 0x00003797 (decimal: 14231)
VA  4: 0x0000087e (decimal: 2174) --> VALID: 0x00002a23 (decimal: 10787)
VA  5: 0x00000d58 (decimal: 3416) --> VALID: 0x00002efd (decimal: 12029)
VA  6: 0x00001055 (decimal: 4181) --> VALID: 0x000031fa (decimal: 12794)
VA  7: 0x0000196d (decimal: 6509) --> VALID: 0x00003b12 (decimal: 15122)
VA  8: 0x00000e21 (decimal: 3617) --> VALID: 0x00002fc6 (decimal: 12230)
VA  9: 0x000007e4 (decimal: 2020) --> VALID: 0x00002989 (decimal: 10633)
```

→ Es hat sich nichts geändert an der Formel, da der Adressbereich und der physikalische Speicher nicht den limit beeinflussen. Also $\max(\text{limit}) + 1 = 6510$

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.

