Eemil Aspholm

CT30A3370 Käyttöjärjestelmät ja systeemiohjelmointi

Viikkotehtävä 6.

Tehtävä 1.

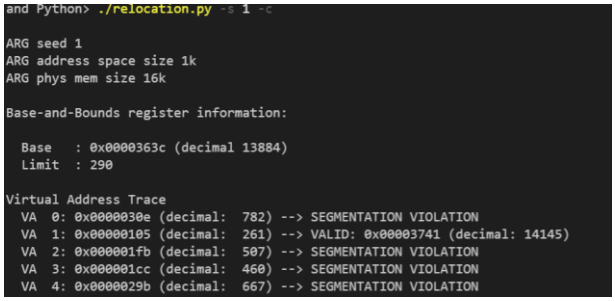
Kaksi prosessia A ja B voidaan suorittaa joko seuraavanlaisessa järjestyksessä ABAB tai BABA. Suoritusjärjestys vaikuttaa muuttujan x saamiin arvoihin, esim. tapaus ABAB, jossa x muuttuja alustetaan ensin numeroksi 1, jonka jälkeen prosessissa B y muuttuja alustetaan numeroksi 2. Tämän jälkeen x arvo muuttuu x = y+1 eli x = 2+1 =3. Lopuksi suoritetaan vielä y=y*2 eli y=2*2=4. Tapauksessa BABA x saa puolestaan arvon 5, sillä prosessin B y=y*2 suoritetaan ennen x arvon laskemista, jolloin x=(y*2)+1 => x=4+1=5.

Tehtävä 2.

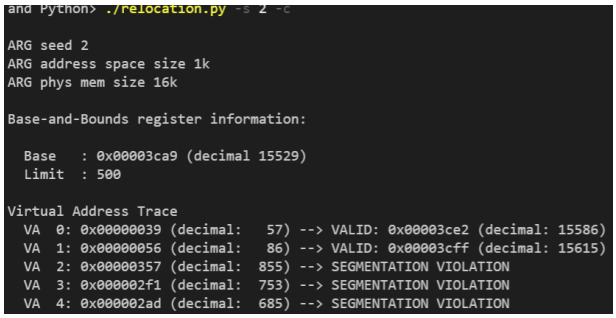
Jos suoritusjärjestys olisi AB AB AB BA, niin kone kirjoittaisi muistiin 0011, sillä kolme ensimmäistä lukua tulisi suoraan B muistiinkirjoituksesta, mutta viimeisin luku olisi järjestysmuutoksen vuoksi A:n lukusarjasta. Tällöin x saisi arvon 3. (binäärikoodi taulukosta).

Tehtävä 3.

Seed 1.



Seed 2.



Seed 3.

```
and Python> ./relocation.py -s 3 -c

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) --> SEGMENTATION VIOLATION
  VA  1: 0x0000026a (decimal:  618) --> SEGMENTATION VIOLATION
  VA  2: 0x00000280 (decimal:  640) --> SEGMENTATION VIOLATION
  VA  3: 0x00000043 (decimal:   67) --> VALID: 0x00002317 (decimal: 8983)
  VA  4: 0x0000000d (decimal:   13) --> VALID: 0x000022e1 (decimal: 8929)
```

Value of -l must be set at 930, to ensure that all the virtual addresses generated are within bounds.

```
and Python> ./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)
```

With trying different values, we can see that the Psize max is 16384. (this includes the base + limit) so when subtracting 100 from 16384 well get the maximum value that base can be set to. 16*1024 =16384

```
Error: address space does not fit into physical memory with those base/bounds values.
Base + Limit: 16484   Psize: 16384
```

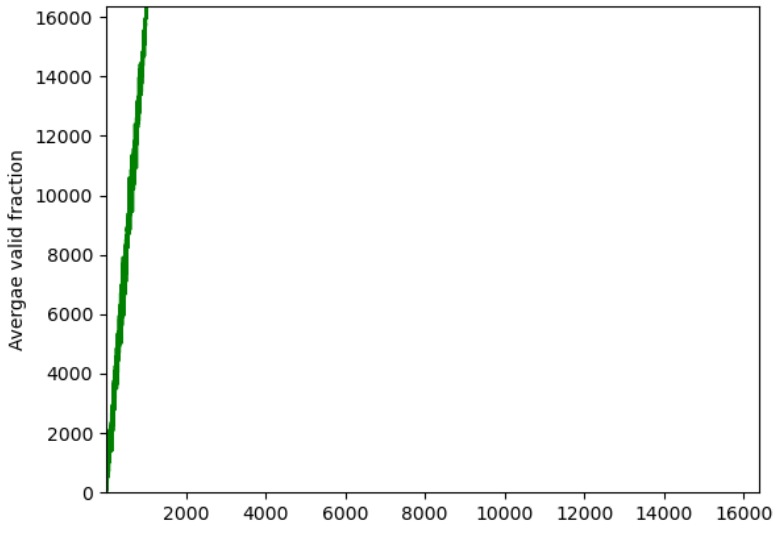
```
and Python> ./relocation.py -s 1 -n 10 -l 100 -b 16284 -c

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

Base-and-Bounds register information:

  Base   : 0x00003f9c (decimal 16284)
  Limit  : 100

Virtual Address Trace
  VA  0: 0x00000089 (decimal:  137) --> SEGMENTATION VIOLATION
  VA  1: 0x00000363 (decimal:  867) --> SEGMENTATION VIOLATION
  VA  2: 0x0000030e (decimal:  782) --> SEGMENTATION VIOLATION
  VA  3: 0x00000105 (decimal:  261) --> SEGMENTATION VIOLATION
  VA  4: 0x000001fb (decimal:  507) --> SEGMENTATION VIOLATION
  VA  5: 0x000001cc (decimal:  460) --> SEGMENTATION VIOLATION
  VA  6: 0x0000029b (decimal:  667) --> SEGMENTATION VIOLATION
  VA  7: 0x00000327 (decimal:  807) --> SEGMENTATION VIOLATION
  VA  8: 0x00000060 (decimal:   96) --> VALID: 0x00003ffc (decimal: 16380)
  VA  9: 0x0000001d (decimal:   29) --> VALID: 0x00003fb9 (decimal: 16313)
```

4. Running the same problems above with larger address spaces (-a) and physical memories (-p) values; For instance, 1024*1024 -100 = 1048576 and 1024*1024*1024 -100 = 1073741724.

```
and Python> ./relocation.py -s 1 -n 10 -l 100 -b 1048576 -a 16m -p 1g -

ARG seed 1
ARG address space size 16m
ARG phys mem size 1g

Base-and-Bounds register information:

  Base   : 0x00100000 (decimal 1048576)
  Limit  : 100

Virtual Address Trace
  VA  0: 0x002265b1 (decimal: 2254257) --> SEGMENTATION VIOLATION
  VA  1: 0x00d8f16a (decimal: 14217578) --> SEGMENTATION VIOLATION
  VA  2: 0x00c386bb (decimal: 12814011) --> SEGMENTATION VIOLATION
  VA  3: 0x00414c34 (decimal: 4279348) --> SEGMENTATION VIOLATION
  VA  4: 0x007ed4d5 (decimal: 8312021) --> SEGMENTATION VIOLATION
  VA  5: 0x007311d8 (decimal: 7541208) --> SEGMENTATION VIOLATION
  VA  6: 0x00a6cecc (decimal: 10931916) --> SEGMENTATION VIOLATION
  VA  7: 0x00c9e9c6 (decimal: 13232582) --> SEGMENTATION VIOLATION
  VA  8: 0x0018072e (decimal: 1574702) --> SEGMENTATION VIOLATION
  VA  9: 0x000741c7 (decimal: 475591) --> SEGMENTATION VIOLATION
```

```
and Python> ./relocation.py -s 1 -n 10 -l 100 -b 1073741724 -a 32m -p 1g -c

ARG seed 1
ARG address space size 32m
ARG phys mem size 1g

Base-and-Bounds register information:

  Base   : 0x3fffff9c (decimal 1073741724)
  Limit  : 100

Virtual Address Trace
  VA  0: 0x0044cb63 (decimal: 4508515) --> SEGMENTATION VIOLATION
  VA  1: 0x01b1e2d5 (decimal: 28435157) --> SEGMENTATION VIOLATION
  VA  2: 0x01870d77 (decimal: 25628023) --> SEGMENTATION VIOLATION
  VA  3: 0x00829868 (decimal: 8558696) --> SEGMENTATION VIOLATION
  VA  4: 0x00fda9aa (decimal: 16624042) --> SEGMENTATION VIOLATION
  VA  5: 0x00e623b1 (decimal: 15082417) --> SEGMENTATION VIOLATION
  VA  6: 0x014d9d98 (decimal: 21863832) --> SEGMENTATION VIOLATION
  VA  7: 0x0193d38c (decimal: 26465164) --> SEGMENTATION VIOLATION
  VA  8: 0x00300e5d (decimal: 3149405) --> SEGMENTATION VIOLATION
  VA  9: 0x000e838f (decimal: 951183) --> SEGMENTATION VIOLATION
```

The maximum size of address space is 16* 1024 as mentioned previously. Generating randomly 200 virtual addresses as a function of the value of the bounds register, we receive such a graph.



The source code that python program uses is applied implementation of xyzz python program for the same exact assignment. Source: https://github.com/xxyzz/ostep-hw/blob/master/15/plot.py