Jonathan Licht

Professor Maus

CMPS 2301-03

February 17, 2021

Lab 3

**Section 2**

1)

Seed 1:

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) --> PA or segmentation violation?

VA 1: 0x00000105 (decimal: 261) --> PA or segmentation violation?

VA 2: 0x000001fb (decimal: 507) --> PA or segmentation violation?

VA 3: 0x000001cc (decimal: 460) --> PA or segmentation violation?

VA 4: 0x0000029b (decimal: 667) --> PA or segmentation violation?

VA 0: Segmentation Fault

VA 1: 13884 + 261 = 14145

VA 2: Segmentation Fault

VA 3: Segmentation Fault

VA 4: Segmentation Fault

Seed 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?

VA 0: 15529 + 57 = 15586

VA 1: 15529 + 86 = 15615

VA 2: Segmentation Fault

VA 3: Segmentation Fault

VA 4: Segmentation Fault

Seed 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?

VA 0: Segmentation Fault

VA 1: Segmentation Fault

VA 2: Segmentation Fault

VA 3: 8916 + 67 = 8983

VA 4: 8916 + 13 = 8929

2)

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?

To ensure that all virtual addresses are in bounds, the bounds register needs to accommodate for the highest virtual address, which is 929. Therefore the bounds register must be at least 930.

3)

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?

Total physical memory is 16384B for the simulation. The limit on the virtual address space is 100B. To calculate the maximum base value, it would simply be 16384B – 100B = 16284B.

**Section 3**

|  |  |  |
| --- | --- | --- |
| **Term** | **Explanation** | **Citation** |
| Policy | A set of rules that control how the user interacts with the computer. Policies are set in place to stop user programs/interaction from harming the hardware of the computer | [Linuxtopia](https://www.linuxtopia.org/online_books/writing_SELinux_policy_guide/about_policies_02.html) |
| Mechanism | Low level protocols that perform tasks that are essential for the OS to run. Mechanisms enforce policies. | [OSTEP](https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-intro.pdf) |
| System Call | A safe way for a process to access hardware. Any process that requires access to the hardware of the computer must use a system call to do it safely. | [Geeksforgeeks](https://www.geeksforgeeks.org/introduction-of-system-call/) |
| API | The commands that can be called by a user to trigger a system call. | [Techterms](https://techterms.com/definition/api) |
| File Descriptor | An integer that is used by the operating system to specify how a file should be accessed. | [Stackoverflow](https://stackoverflow.com/questions/5256599/what-are-file-descriptors-explained-in-simple-terms) |
| Interrupt | A mechanism that stops the current process from running so that OS can run and decide what to do next. They are triggered by a timer or by an I/O request. | [OSTEP](https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-mechanisms.pdf) |
| Trap | A set of instructions that are triggered when there is a system call. The OS ‘traps’ into a privileged state in which it has access to the hardware of the computer. It then does whatever process was requested by the system call. | [OSTEP](https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-mechanisms.pdf) |
| Trap Table | Links how to execute a trap to the system calls that trigger it. The table is created at boot time. | [ClassNotes](https://git.tulane.edu/amaus/cmps-2300-02-spring-2021/-/blob/master/lecture-notes/03-cpu-virualization.md) |
| Stack Pointer | Keeps track of the beginning of stack and used for managing the stack. | [OSTEP](https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-intro.pdf) |
| Kernel Mode | The privileged mode in which the OS is allowed to access the hardware of the computer. | [OSTEP](https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-mechanisms.pdf) |
| Kernel Stack | A stack kept in the memory for the OS used by the OS. It can only be used when the computer is in kernel mode. | [Baeldung](https://www.baeldung.com/linux/kernel-stack-and-user-space-stack#:~:text=The%20kernel%20stack%20is%20part,the%20running%20process%20is%20used) |
| Standard Output | The output that is displayed to the terminal by the OS or a user program. | [Linuxhint](https://linuxhint.com/bash_stdin_stderr_stdout/) |