Build an operating system from scratch

In this project, you will build your own mini-operating system from scratch. Doing so is not trivial. Operating systems are complex beasts that operate at the boundary between software and hardware. It means that you should not be afraid to look up the things that you may not understand initially, tinker with assembly, and cope with a dash of frustration now and then. On the other hand, once you have done this, you have, for the first time, developed something that boots on bare metal (or a VM) and is truly 'all yours'. This is both cool and very educational, as you will finally see what is going on under the hood.

To structure the projects, we will adhere to the timeline below. This will help us ensure supervision is available when you need it and also to guarantee that the project, which is after all quite challenging, finishes in bounded time.

In this document, we sometimes refer to the Simple Operating System as SOS.

Overall organization

The operating system you will build will consist of two parts. The first part is just a basic kernel with minimal functionality, and is the same for all projects. The second part is a specialization of this basic kernel in a direction that is unique for each student. A PhD student acting as daily supervisor will be available for questions and guidance at least during one fixed slot in the week (probably Wednesday at 10:00 in https://meet.google.com/htj-twbw-eeh).

Throughout the project, you will receive supervision by this PhD student, while the senior faculty operate more in the background. However, you will be discussing the first part of your project in a Google Meet session with faculty and PhD student, where you will have to show that you have really understood what you have built. Doing well on this part will lead to a passing grade (assuming you also do well on your thesis and presentation).

After completing the second part, you will start writing the thesis. Do not underestimate this part. Below you find a link to an example thesis. Finally, you will have to present your work in front of the research group.

Basic kernel

For the first part we expect you to bring up a basic kernel with

A. IDT/GDT initialization
B. 64bit mode
C. functionality to discover how much physical memory is available,
D. initialisation of virtual memory and page tables for these page,
E. basic memory allocation and free functions

As guidance, you should study, reproduce and *understand* the following tutorials:

A. http://ringzeroandlower.com/2017/08/08/x86-64-kernel-boot.html
B. https://www.cs.vu.nl/~herbertb/misc/basickernel.pdf
C. https://download.vusec.net/papers/kbook-0.1.bos.pdf

We will soon provide some test cases that your kernel needs to pass before you are eligible to go to the next step (Specialization) of your thesis project. These tests will check whether your kernel can successfully detect the available amount of physical memory and can successfully handle different sequences of memory allocations and deallocations (i.e., kmallocs, and kfrees).

Specialization

There are many possibilities. The ones below are suggestions, but maybe you have something else that is also interesting. In addition, it is of course possible to implement multiple of these.

- Implement user space processes. This means that you need support
  - iret/syscall (for returning from and trapping into your SOS kernel)
  - timer (to determine how much time a process has been active and schedule another task if needed)
  - scheduling (simplest: cooperative)
  - optionally: pre-emption
  - optionally: support for communication over serial ports
  - optionally: support basic side channel protection by making sure that sibling sibling hyperthreads of different processes cannot be both in userspace at the same time.
- Virtualization i: run your kernel in your SOS kernel. In this project, you have to turn on virtualization in such a way that you can run your basic kernel on top of the very same basic kernel.
- Virtualization ii: run SOS in the KVM hypervisor using the KWVM API. In this project, you also turn towards virtualization, but in this case, you will use the Linux KVM API to create an environment in which to run your SOS .
- File system. You have to create a simple file system. Starting by implementing primitives to read and write a byte at a specific location on disk, you will extend this to create a simple file system structure (with directories, files, etc.. At the least you will need to look at:
  - a basic I/O driver
  - a logic structure for a basic file system.
- Network driver : NIC ne2000. Implement a simple network driver to obtain efficient network communication.
- Port to RISC V. The name says it all: port what you have to RISC V.

- Investigate speculative execution attacks: play with different bits in the page table entries to investigate possible new attack vectors. You need to at least show the Foreshadow vulnerability in action (bypassing the present bit).

We will discuss what we expect you to do during the mid-term test (see Timeline).

Thesis

Students often wonder what is expected in a bachelor thesis. First, have a look at the official info with all the rules and regulations for bachelor projects can be found on the corresponding Canvas page and on the study guide. Second, an example of what *we* consider a successful bachelor thesis is available on our student projects page:

https://www.vusec.net/student-projects/

Timeline

The project officially starts on the first day of Period 5 and should finish after 10 weeks. The schedule is as follows:

Wk	Description	Additional comments
2	● basic set up: ○ reproduce: http://ringzeroandlower.com/2017/08/08/x86-64-kernel-boot.html ■ IDT/GDT ■ MMU ■ bring up 64b kernel with paging ● hands-on meeting with phd student to discuss progress	We will provide the test functions for the first part of your thesis project.
3	● figure out size of physical memory ● do address mappings for full memory	
4	● kmalloc/kfree ○ should run our test function ● Mid term test (to see if you understood things) ● Discussion on the specialization part of your thesis	The time/location of the midterm oral test will be announced in advance.
5-8	Specialization ● user space	

	● virtualization i: run your kernel in your kernel ● virtualization ii: run SOS in KVM hypervisor using KVM API ● file system ● network driver ● port to RISC V ● testing speculative execution attacks ● formal verification ● ...	
8	Start writing thesis + wrap up loose ends	
10	Submit thesis, present	