Writing an operating system in 2.5 years

Yunhao Zhang
But first, writing an OS in 0.5 years

• P0: understand C and user-level instructions
• P1: understand context-switch and multi-threading
• P2: understand exception and privilege levels
• P3: understand file system abstractions
• P4: understand file system implementation
• P5: understand bus and I/O devices

Why 2.5 years? An overview

- **Motivation**: Summer 2020
- **Obstacles**: Fall 2020
- **Ideas**: Fall 2021
- **Implementation**: Spring 2022
- **Evaluation**: Fall 2022

- **0.5 years**
In June 2020, we created egos

- ~20K lines of code
- run on Intel/Arm CPU
- run as a Linux / MacOS process
20K lines of code

Students read a very small portion

2K lines of code

Students read a large portion
Intel x86 (1987)
CPU document has **several thousands** of pages

RISC-V (2010)
CPU document has **<100** of pages
User-mode OS
Easier to deploy and run

OS on hardware
More realistic and fun
The motivation

~20K → ~2K

x86 → RISC-V

Linux / MacOS → hardware
Lesson

Good motivations should convince non-experts why the work is important.

non-experts like students, my friend doing ML theory, etc.
Yet, ideal $\neq$ possible

Obstacles

Summer 2020  Fall 2020
Obstacles & Hope

- The hardware had only 24KB memory
- There was no disk
- The hardware supports timer interrupt
- The hardware supports privilege levels and exceptions
Overcome obstacles with ideas!

- Idea #1: Increase memory
  - Jan 2021

- Idea #2: SD card extension
  - Nov 2021
Open-source hardware
Running open-source hardware

A binary file encoding the hardware design (clocks, registers, circuits, etc.)

FPGA: emulate the hardware design
Idea #1: Increase memory

```scala
  86  dcache = Some(DCacheParams(
  87       rowBits = site(SystemBusKey).beatBits,
  88           nSets = 256, // 16Kb scratchpad
  89           nWays = 1,
  90           nTLBEntries = 4,
  91           nMSHRs = 0,
  92       blockBytes = site(CacheBlockBytes),
  93       scratch = Some(0x80000000L)),
```
Background of the SPI bus

<table>
<thead>
<tr>
<th>Instance</th>
<th>Flash Controller</th>
<th>Address</th>
<th>cs_width</th>
<th>div_width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSPI 0</td>
<td>Y</td>
<td>0x10014000</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>SPI 1</td>
<td>N</td>
<td>0x10024000</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>SPI 2</td>
<td>N</td>
<td>0x10034000</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 64: SPI Instances

Chapter 19 of Sifive FE310 manual, v19p04
Remap SPI1 to microSD card

<table>
<thead>
<tr>
<th>Instance</th>
<th>Flash Controller</th>
<th>Address</th>
<th>cs_width</th>
<th>div_width</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSPI 0</td>
<td>Y</td>
<td>0x10014000</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>SPI 1</td>
<td>N</td>
<td>0x10024000</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>SPI 2</td>
<td>N</td>
<td>0x10034000</td>
<td>1</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 64: SPI Instances

new SPI1

old SPI1
Remap SPI1 to microSD card

Old SPI1 mapping

New SPI1 mapping

6 pins
GND + VCC + SPI (4)
Idea #2: Using microSD card as disk

## ChipKit SPI

```python
set_property -dict { PACKAGE_PIN G1 IOSTANDARD LVCMOS33 } [get_ports { ck_miso }];
set_property -dict { PACKAGE_PIN H1 IOSTANDARD LVCMOS33 } [get_ports { ck_mosi }];
set_property -dict { PACKAGE_PIN F1 IOSTANDARD LVCMOS33 } [get_ports { ck_sck }];
set_property -dict { PACKAGE_PIN C1 IOSTANDARD LVCMOS33 } [get_ports { ck_ss }];
```

## Pmod Header JA

```python
set_property -dict { PACKAGE_PIN G13 IOSTANDARD LVCMOS33 } [get_ports { ja_0 }];
set_property -dict { PACKAGE_PIN B11 IOSTANDARD LVCMOS33 } [get_ports { ja_1 }];
set_property -dict { PACKAGE_PIN A11 IOSTANDARD LVCMOS33 } [get_ports { ja_2 }];
set_property -dict { PACKAGE_PIN D12 IOSTANDARD LVCMOS33 } [get_ports { ja_3 }];
```

Find these 4 wires in the repo and replace them

The road to ideas is difficult

No concrete progress for >1 year

Not sure whether this can succeed at all

Only person working on this project
Take-away:
Ideas are difficult to come up with and there is no guarantee of success 😅
The bug taking me >1 day to fix

core = RocketCoreParams(
    useVM = false,
    fpu = None,
    mulDiv = Some(MulDivParams(mulUnroll = 8)),
    btb = None,
    dcache = Some(DCacheParams(
        rowBits = site(SystemBusKey).beatBits,
        nSets = 256, // 16Kb scratchpad
        nWays = 1,
        nTLBEntries = 4,
        nMSHRs = 0,
        blockBytes = site(CacheBlockBytes),
        scratch = Some(0x8000000L)),
    icache = Some(ICacheParams(
        rowBits = site(SystemBusKey).beatBits,
        nSets = 64,
        nWays = 1,
        nTLBEntries = 4,
        blockBytes = site(CacheBlockBytes)))
)
Lesson
Implementing a system is non-trivial and requires determination and hard work
A 4.5-year research process

2 years: Becoming familiar with OS education

Then, challenge state-of-the-art

Motivation  Obstacles  Ideas  Implementation  Evaluation

Summer 2018  Summer 2020

Summer 2020  Fall 2022
Next step: **Publish** the research

**Fighting for a world where every college student can read all the code of an operating system**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>199</td>
<td>Boot Loader &amp; TTY Driver</td>
<td>336</td>
<td>File System</td>
</tr>
<tr>
<td>182</td>
<td>SD Card Driver</td>
<td>264</td>
<td>Applications &amp; Daemons</td>
</tr>
<tr>
<td>32</td>
<td>Interrupt &amp; Exception Handling</td>
<td>269</td>
<td>Library &amp; <strong>Networking (TBA)</strong></td>
</tr>
<tr>
<td>137</td>
<td>Page Table &amp; Software Translation</td>
<td>64</td>
<td>Makefile</td>
</tr>
<tr>
<td>345</td>
<td>Timer, Scheduler &amp; System Call</td>
<td>172</td>
<td>RISC–V Emulator &amp; Board Tools</td>
</tr>
</tbody>
</table>