



TECHNISCHE
UNIVERSITÄT
DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

OPERATING-SYSTEM CONSTRUCTION

Material based on slides by Olaf
Spinczyk, Universität Osnabrück

Summary and Outlook

<https://tud.de/inf/os/studium/vorlesungen/betriebssystembau>

HORST SCHIRMEIER

Agenda

- Summary
- Evaluation
- Exam
- Outlook
- Get Involved

Agenda

- **Summary**
- Evaluation
- Exam
- Outlook
- Get Involved

What We've Covered

- L 1: Introduction
- L 2: Operating-System Development 101
- L 3: Interrupts – Hardware
- L 4: Interrupts – Software
- L 5: Interrupts – Synchronization
- L 6: Intel®64: The 32/64-Bit Intel Architecture
- L 7: Coroutines and Threads
- L 8: Scheduling
- L 9: Thread Synchronization
- L 10: Inter-process Communication
- L 11: Bus Systems
- L 12: Device Drivers

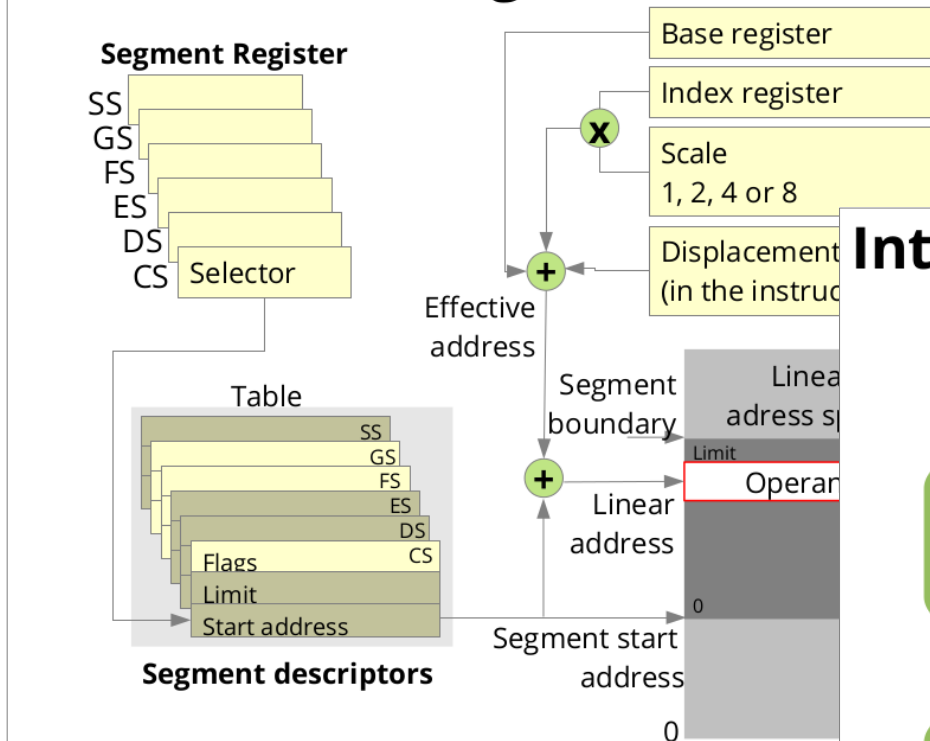
What We've Covered

- L 1: Introduction
- L 2: Operating-System Development 101
- L 3: Interrupts – Hardware**
- L 4: Interrupts – Software
- L 5: Interrupts – Synchronization
- L 6: Intel®64: The 32/64-Bit Intel Architecture**
- L 7: Coroutines and Threads
- L 8: Scheduling
- L 9: Thread Synchronization
- L 10: Inter-process Communication
- L 11: Bus Systems**
- L 12: Device Drivers

1. An expedition through the architecture of the x86 PC

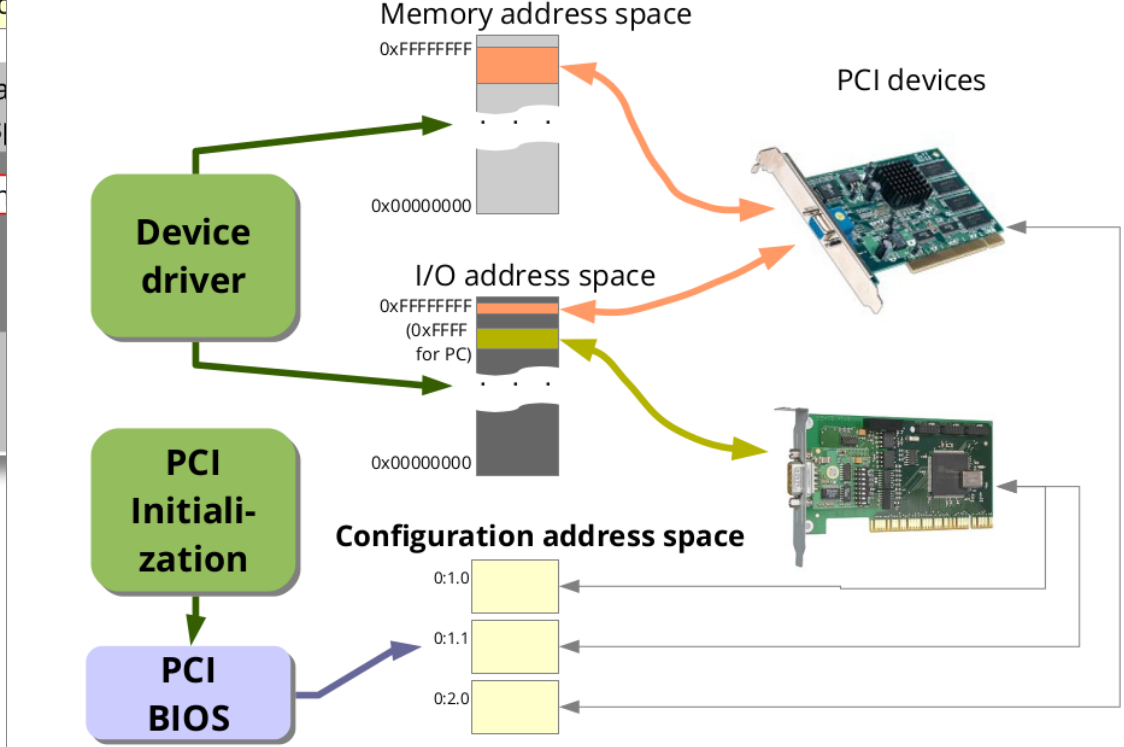
Three Core Areas

IA-32 / x86-64: Segments



1. An expedition through the architecture of the x86 PC

Interacting with PCI Devices



What We've Covered

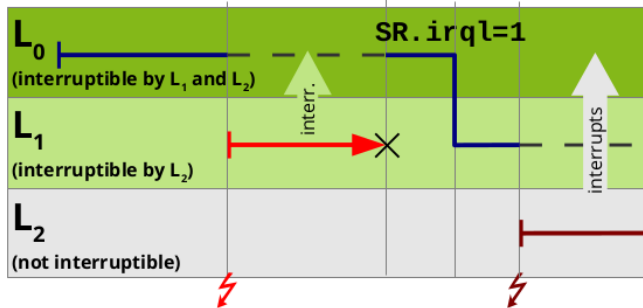
- L 1: Introduction
- L 2: Operating-System Development 101
- L 3: Interrupts – Hardware**
- L 4: Interrupts – Software**
- L 5: Interrupts – Synchronization**
- L 6: Intel®64: The 32/64-Bit Intel Architecture
- L 7: Coroutines and Threads**
- L 8: Scheduling**
- L 9: Thread Synchronization**
- L 10: Inter-process Communication**
- L 11: Bus Systems
- L 12: Device Drivers

2. Control flows and their interactions

What We've Covered

Control-Flow Level Model

- Generalization to multiple interrupt levels:
 - Control flows on L_f are
 - **interrupted anytime** by control flows on L_g (for $f < g$)
 - **never interrupted** by control flows on L_e (for $e \leq f$)
 - **sequentialized** with other control flows on L_f (for $f > 0$)
 - Control flows can switch levels
 - by special operations (here: modifying the state register `SR.irql=1`)



2. Control flows and their interactions

Control-Flow Level Model: **new**

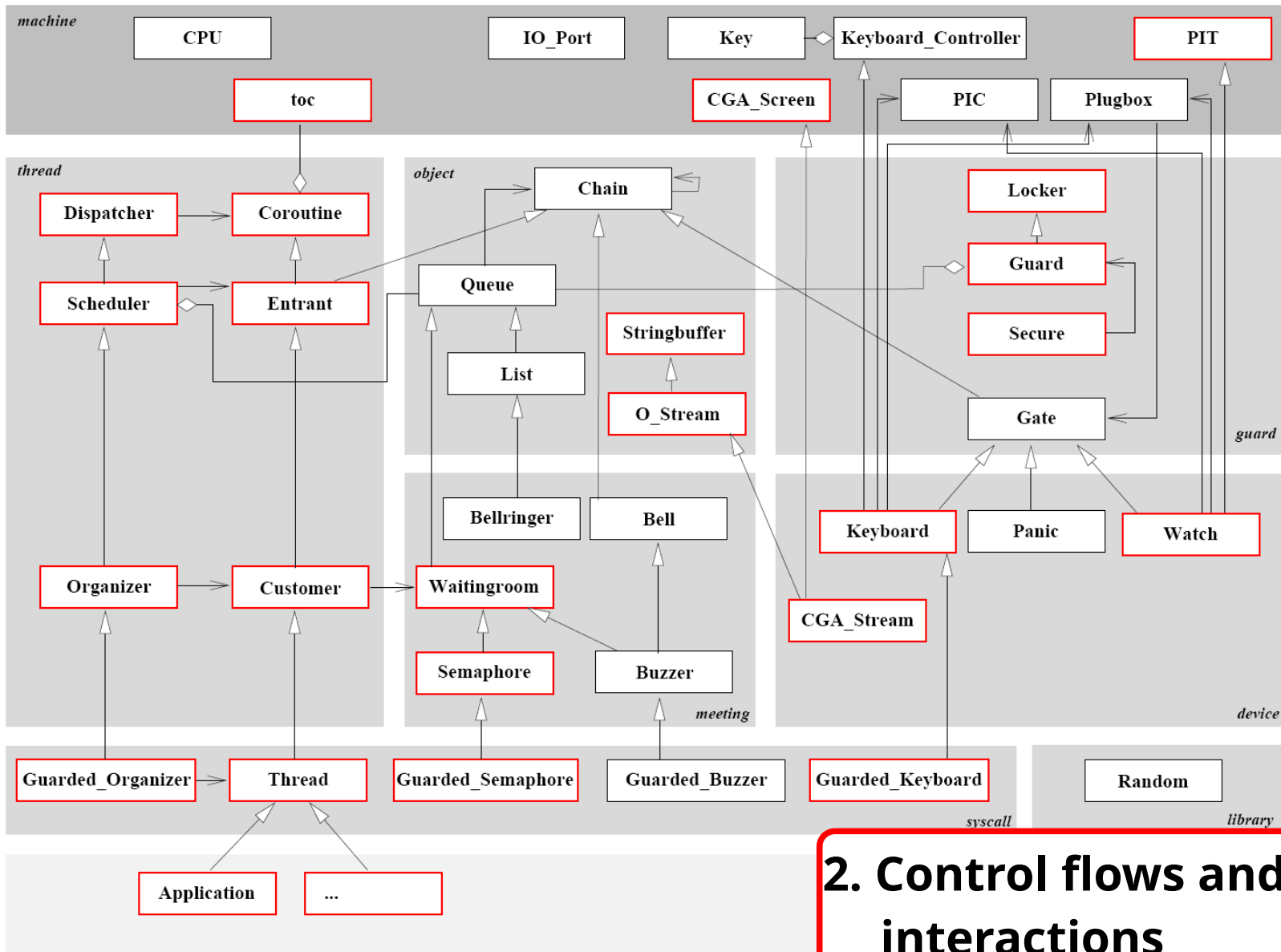
- Control flows on L_f are
 - **interrupted anytime** by control flows on L_g (for $f < g$)
 - **never interrupted** by control flows on L_e (for $e \leq f$)
 - **sequentialized** with other control flows on L_f (for $f > 0$)
 - **preempted** by other control flows on L_f (for $f = 0$)

L_0 → **Thread level**
 (interruptible, **preemptible**)
 L_1 → **Epilogue level**
 (interruptible, **not preemptible**)
 L_2 → **Interrupt level**
 (not interruptible, **not preemptible**)

Control flows on level L_0 (thread level) are **preemptible**.

To maintain consistency on this level, we need additional mechanisms for **thread synchronization**.

Three Core Areas



2. Control flows and their interactions

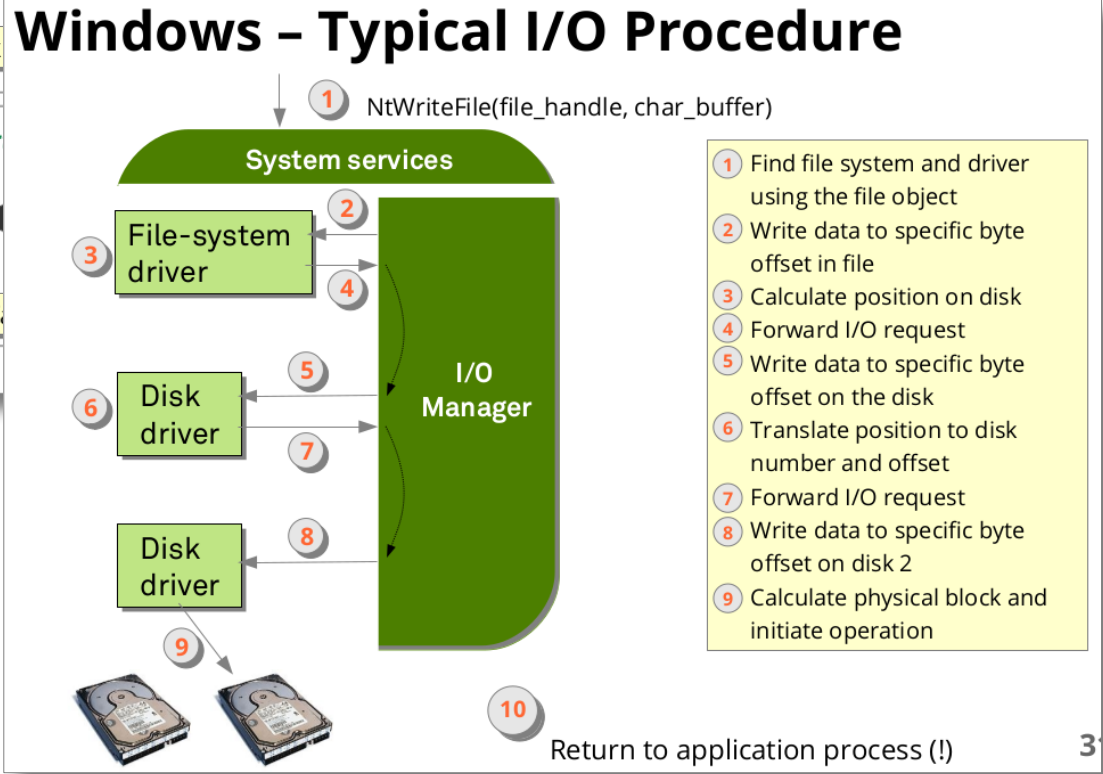
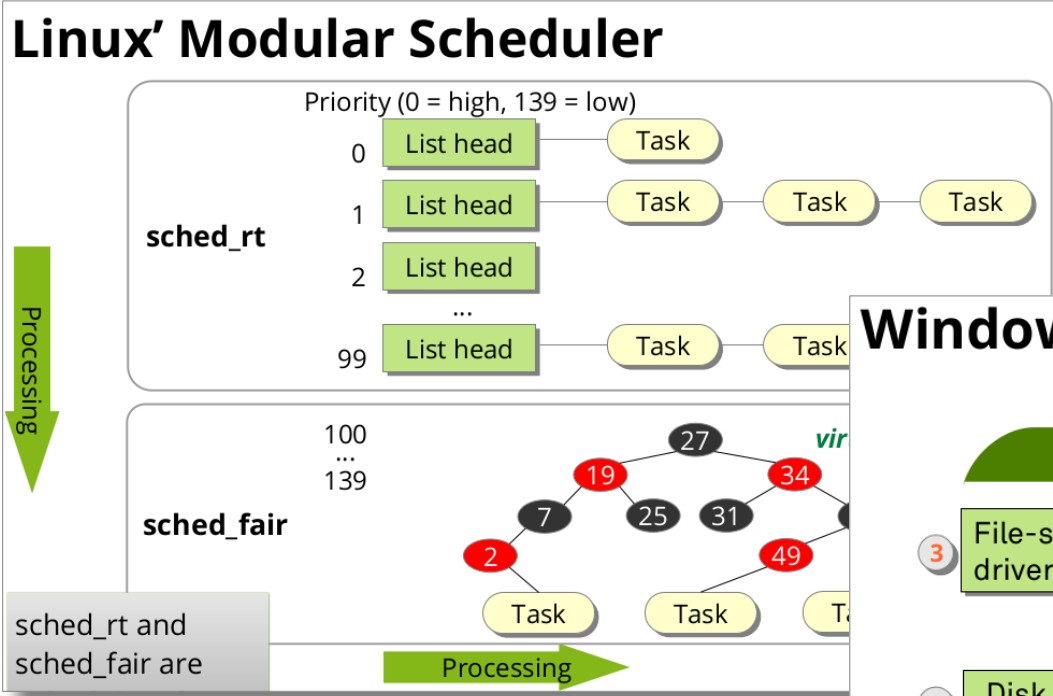
What We've Covered

- L 1: Introduction
- L 2: Operating-System Development 101
- L 3: Interrupts – Hardware
- L 4: Interrupts – Software
- L 5: Interrupts – Synchronization
- L 6: Intel®64: The 32/64-Bit Intel Architecture
- L 7: Coroutines and Threads
- L 8: Scheduling**
- L 9: Thread Synchronization
- L 10: Inter-process Communication**
- L 11: Bus Systems
- L 12: Device Drivers**

**3. OS concepts in general
and in Linux/Windows**

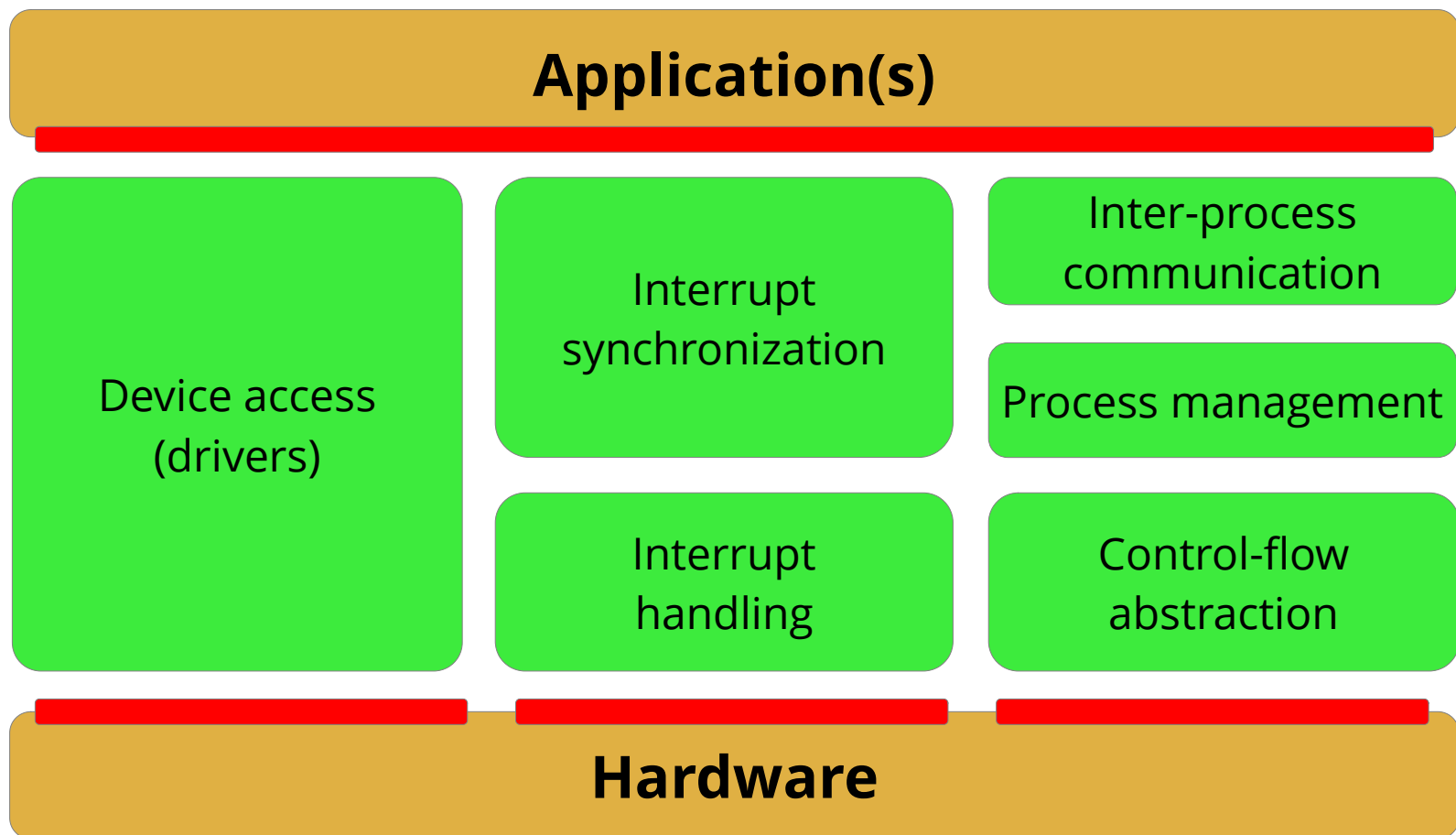
Three Core Areas

3. OS concepts in general and in Linux/Windows



... Altogether Quite A Lot!

Structure of the "OO-StuBS" operating system:



Agenda

- Summary
- **Evaluation**
- Exam
- Outlook
- Get Involved

Evaluation Results

- [switch to evaluation PDF]

You have more to say?

→ Contact me

→ Use our “Anonymer Briefkasten”

Agenda

- Summary
- Evaluation
- **Exam**
- Outlook
- Get Involved

Exam

- **Contents**

- All three core areas
- Exercises + lab tasks are also relevant
 - e.g. explain concepts from core areas 1 + 2 with implementation in OOSTuBS
 - C++, x86 assembler
- Concepts are more important than learning stuff by heart

- **Exam appointments**

- Until Jul 21st; from Aug 14th
- Appointments: e-mail to sandy.seifarth-haupold@tu-dresden.de including module name and preferred appointment time frame
- (Withdrawal: until 14 days before the appointment)

- Language: German or English (or, if necessary, a mix)

Agenda

- Summary
- Evaluation
- Exam
- **Outlook**
- Get Involved

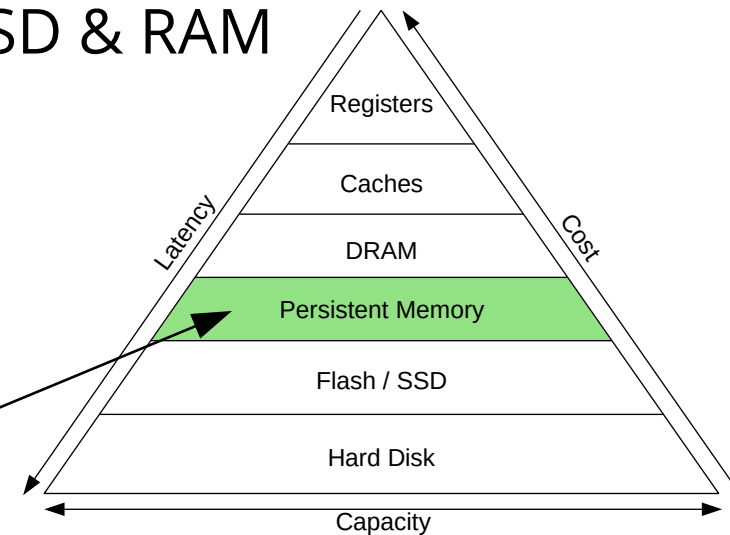
Challenge: New Memory Technologies

NVRAM: New class of memory between SSD & RAM

- Almost as fast as DRAM
- Maintains its state if turned off
- Available for servers since 2019
 - Optane DCPMMs



Source: Intel



Research Questions:

- File abstraction vs. direct, byte-wise access?
- Persistent data structures, processes, systems?
- Reliability? Reboot doesn't „fix“ the system anymore!
- 1D memory hierarchy? *Demand Paging?*

Other emerging technologies:
Processing-in-memory (PIM)
High-bandwidth Memory (HBM)

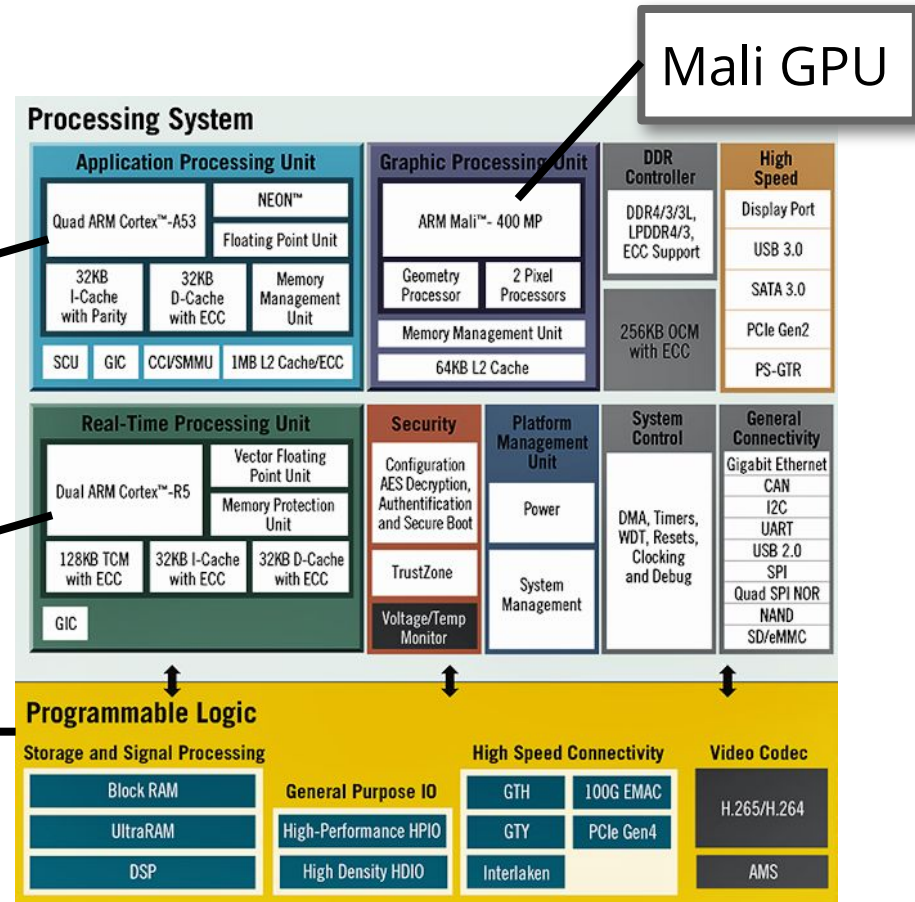
Challenge: Hardware Heterogeneity

- Example: Xilinx Ultrascale+

4 ARM Cortex A53 processor cores

2 ARM Cortex R5 real-time processors

Programmable logic (FPGA)



Research Questions:

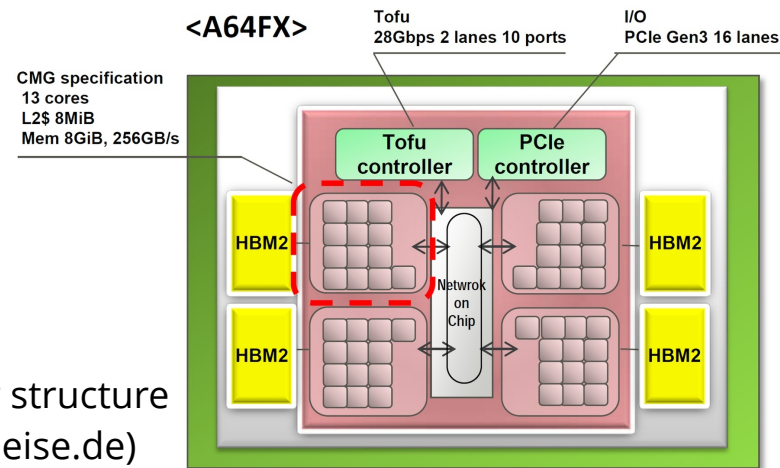
- Common control-flow abstraction?
- How does scheduling work here?

Challenge: Manycore Hardware

- Example: Fujitsu A64FX (#1 top500.org)
 - 48+4 cores (2.7 Tflops) per chip
 - 4 High-Bandwidth Memories (1 TB/s); 4 NUMA regions
 - 7.299.072 cores in the supercomputer
 - *Remote DMA (RDMA)* for communication



Fugaku supercomputer (Source: Fujitsu)



Research Questions:

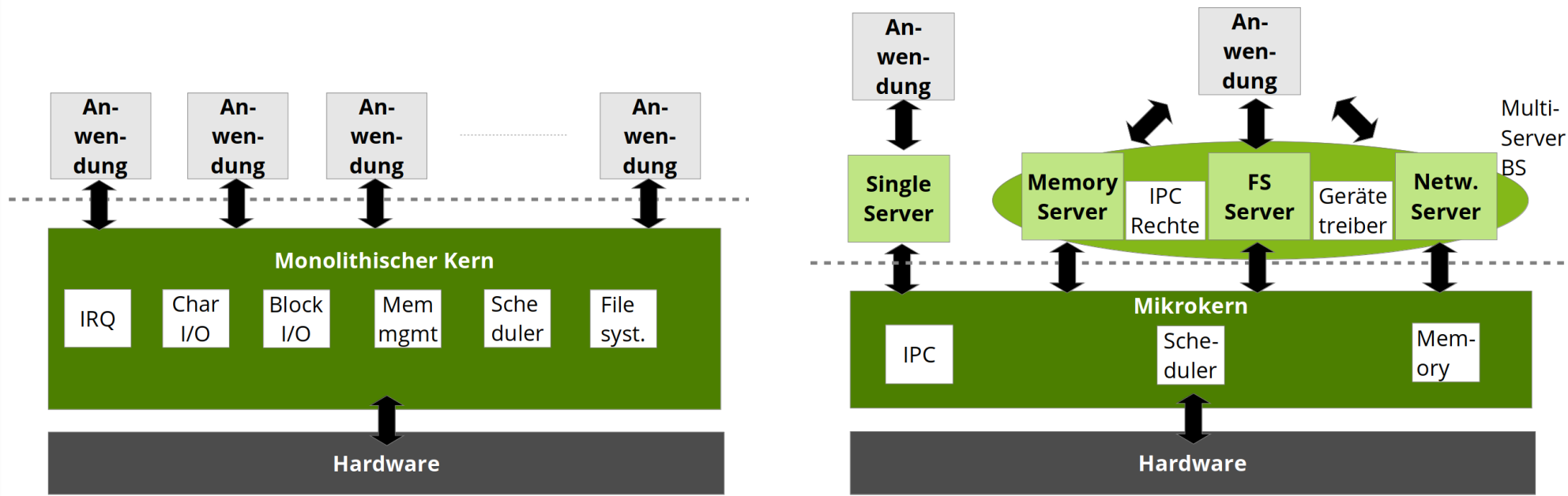
- Do we still need CPU multiplexing?
- How to place control flows and data objects?


Research Topics at the OS Chair

- Dealing with **complexity**
 - constructively (projects L4, M³)
 - analytically (project LockDoc)
- **Non-functional properties**
 - *Security*
 - *Safety/fault tolerance* (Projekts DanceOS, FAIL*)
 - Timing behavior
 - Energy
- **Hardware** developments
 - Disruptive memory technologies (projects VAMPIR, FOSSIL)




Complexity: Monolith vs. Microkernel

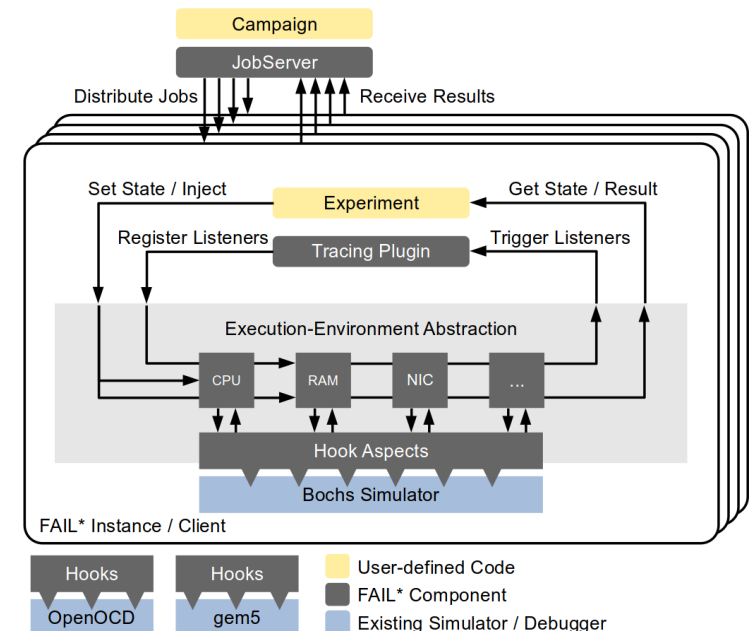


- Fine-grained locking is error-prone: **LockDoc** project 
- *Security*: Take over a kernel component = Game Over
- but: Performance, lots of legacy code

- **L4Re** project
- Minimal, application specific **Trusted Computing Base**
- Constructive complexity control (*divide & conquer*)

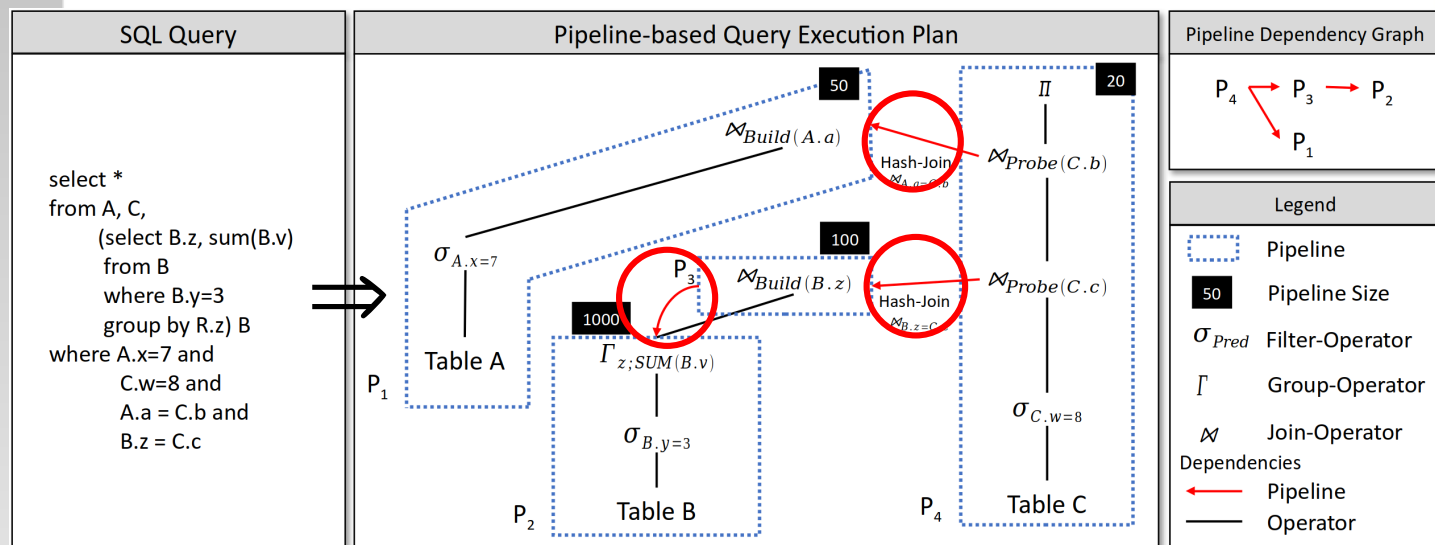
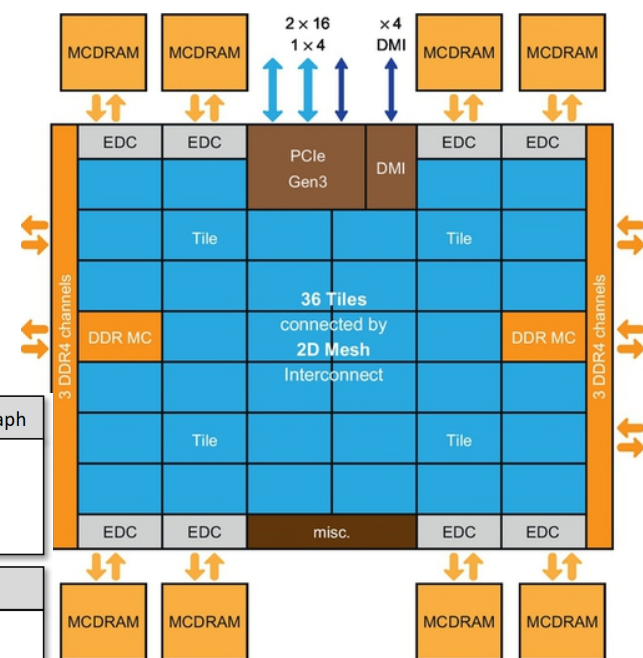
Fault-Tolerant Operating Systems

- *Soft Errors* can cause e.g. bit-flips in memory or the CPU
- How can we **extend** operating systems – or **design** them ground-up – so that they still work?
 - **DanceOS** project 
- How can we (systematically) determine whether we were successful?
 - Fault injection: **FAIL*** project



Disruptive Memory Technologies

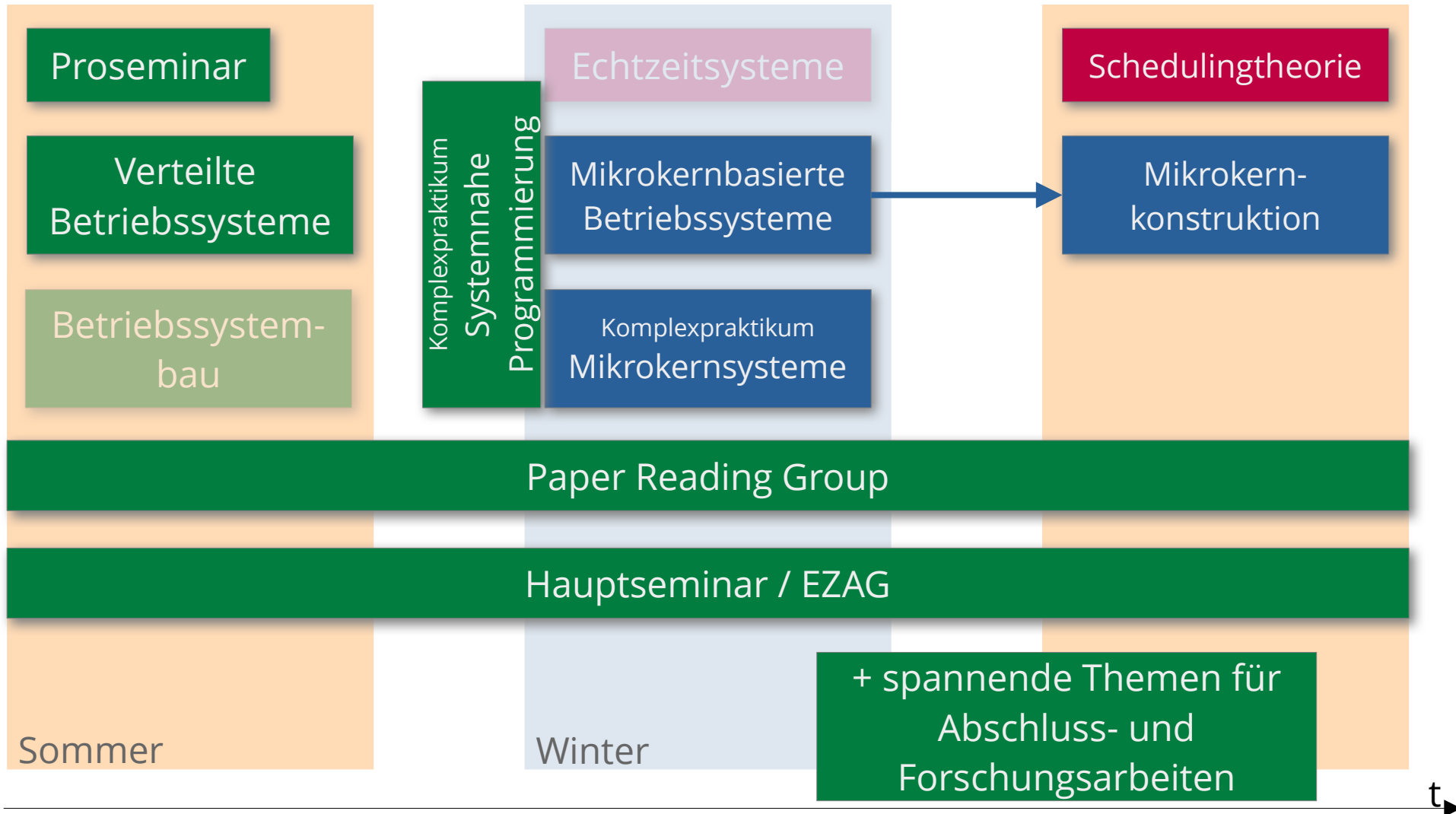
- Dealing with heterogeneous memories: **VAMPIR** project
 - Latency, throughput, persistency, fault tolerance, wearout, energy consumption, PIM capabilities, ...
- Use case: databases
 - Predictions much easier!



Agenda

- Summary
- Evaluation
- Exam
- Outlook
- **Get Involved**

Other Lectures



Thesis Topics

{Bachelor, Master, Diploma} theses, Beleg, Forschungsprojekt, ...

- **Empirical work** → Build, measure, evaluate

We Need: Student Assistants

- **Tutors** for “Betriebssysteme und Sicherheit” (WS 23/24) and “Operating-System Construction” (SS 23)
 - Exercises: Discuss + collectively solve work sheets
 - Support students in the PC pool (C/C++)
- Assistant in a **Research Project**
 - As required: Programming, literature research, measurements, etc.

That's It!

Thank you!
I hope we'll meet again.



Don't forget:
Task #7 Contest
next week, Wednesday 2023-07-19 11:10!
Voting only possible in presence.