

OPERATING-SYSTEM CONSTRUCTION

Summary and Outlook

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

TILL SMEJKAL (slides by 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

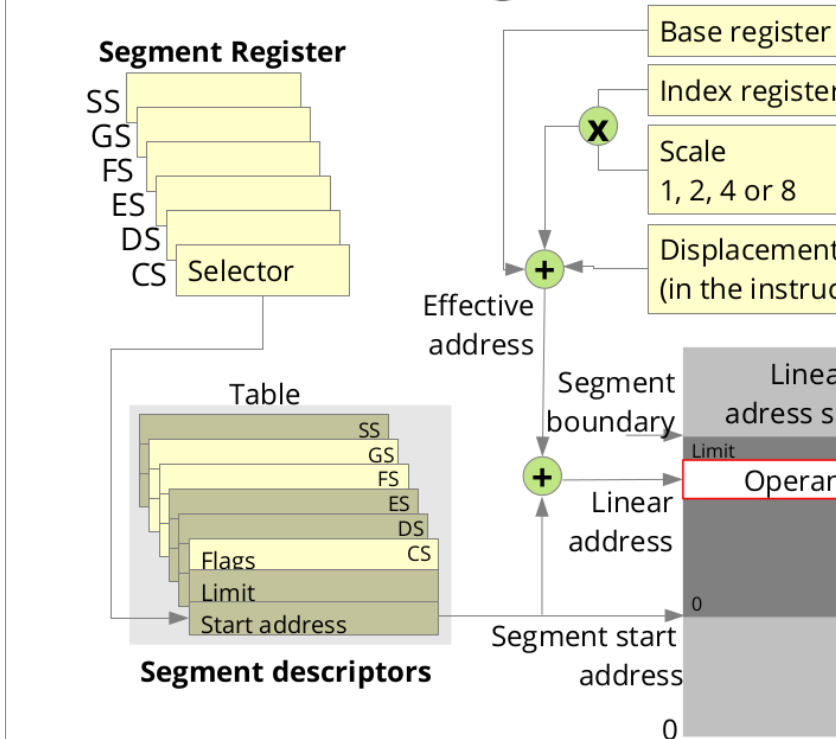
- L 1: Introduction
- L 2: Operating-System Development 101
- L 3: Interrupts – Hardware**
- L 4: Interrupts – Software
- L 5: Interrupts – Synchronization
- L 6: Intel®64: The 32/64-Bit Intel Architecture**
- L 7: Coroutines and Threads
- L 8: Scheduling
- L 9: Thread Synchronization
- L 10: Inter-process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

**1. An expedition through the
architecture of the x86 PC**

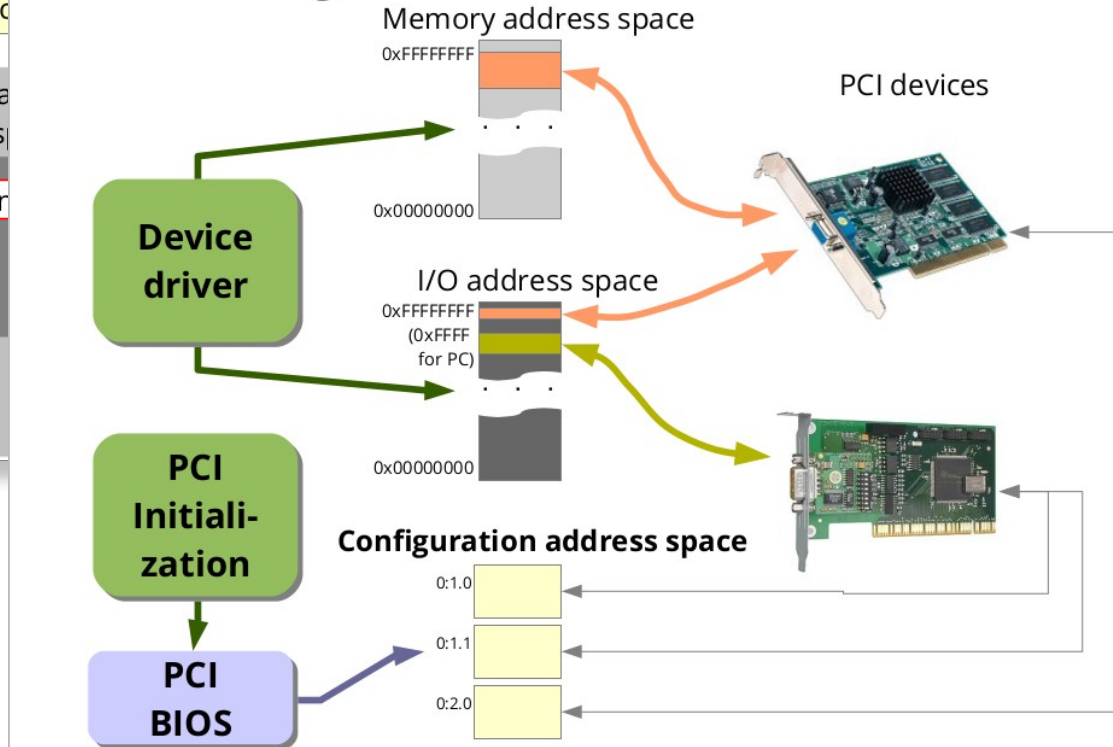
Three Core Areas

1. An expedition through the architecture of the x86 PC

IA-32 / x86-64: Segments



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

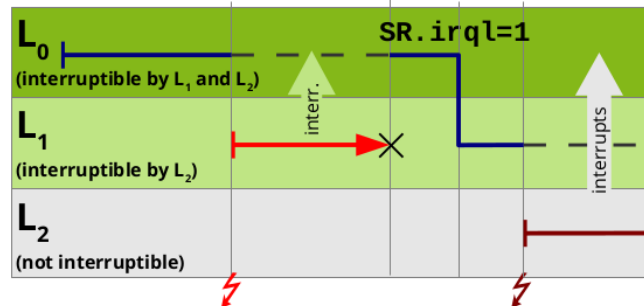
2. Control flows and their interactions

What We've Covered

2. Control flows and their interactions

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)



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 $f = 0$)

$L_0 \rightarrow$ Thread level

(interruptible, preemptible)

$L_1 \rightarrow$ Epilogue level

(interruptible, not preemptible)

$L_2 \rightarrow$ Interrupt level

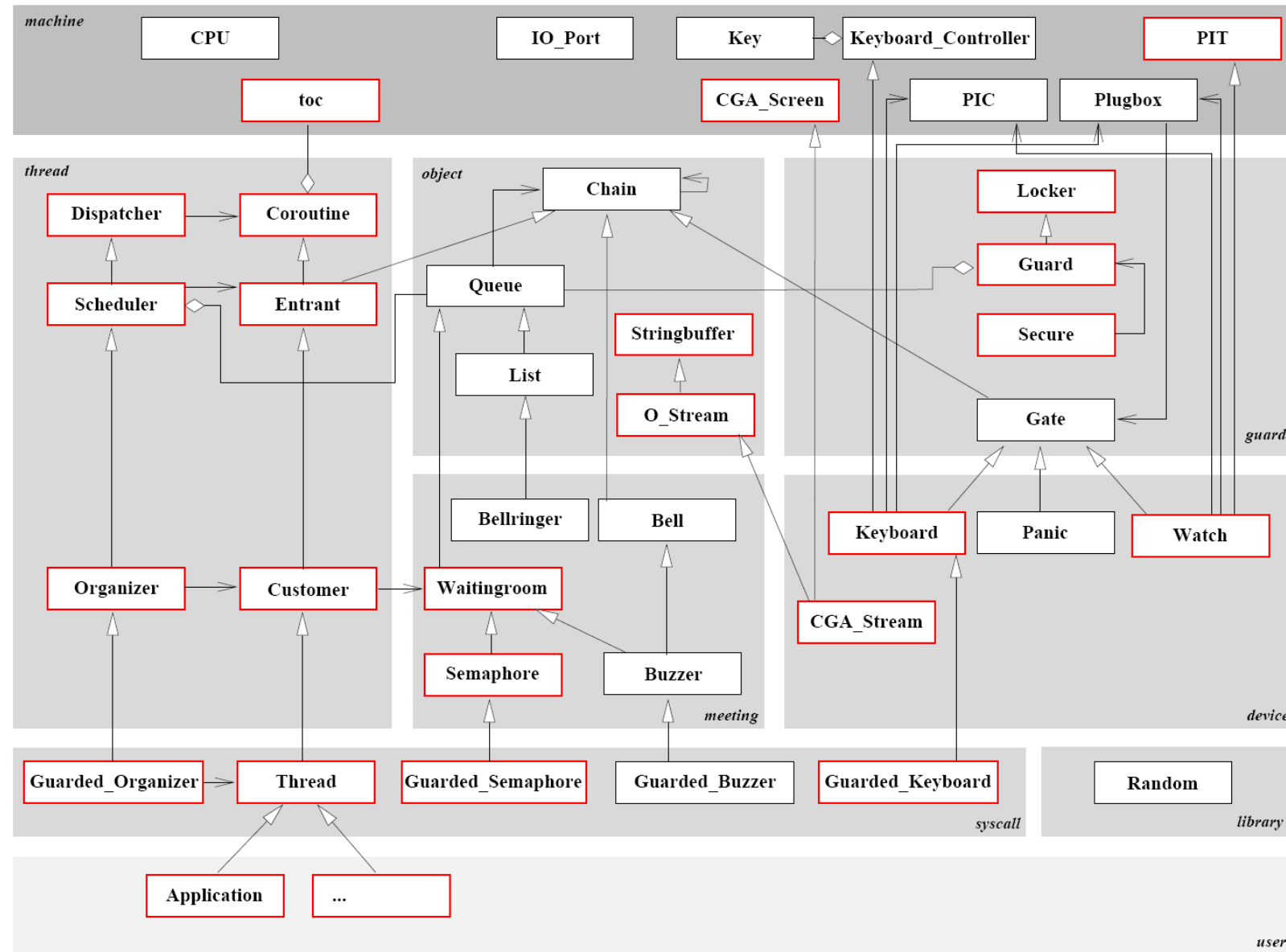
(not interruptible, not preemptible)

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

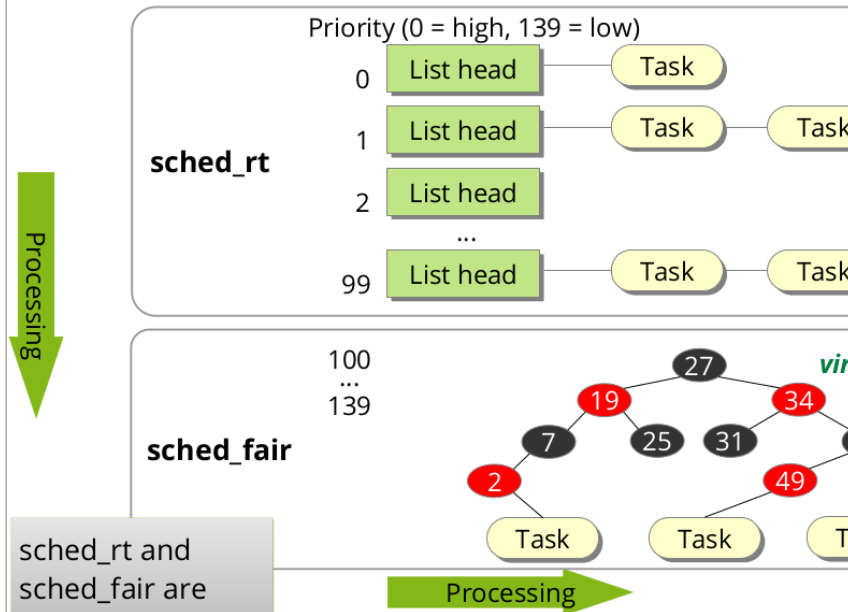
- L 1: Introduction
- L 2: Operating-System Development 101
- L 3: Interrupts – Hardware
- L 4: Interrupts – Software
- L 5: Interrupts – Synchronization
- L 6: Intel®64: The 32/64-Bit Intel Architecture
- L 7: Coroutines and Threads
- L 8: Scheduling**
- L 9: Thread Synchronization
- L 10: Inter-process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

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

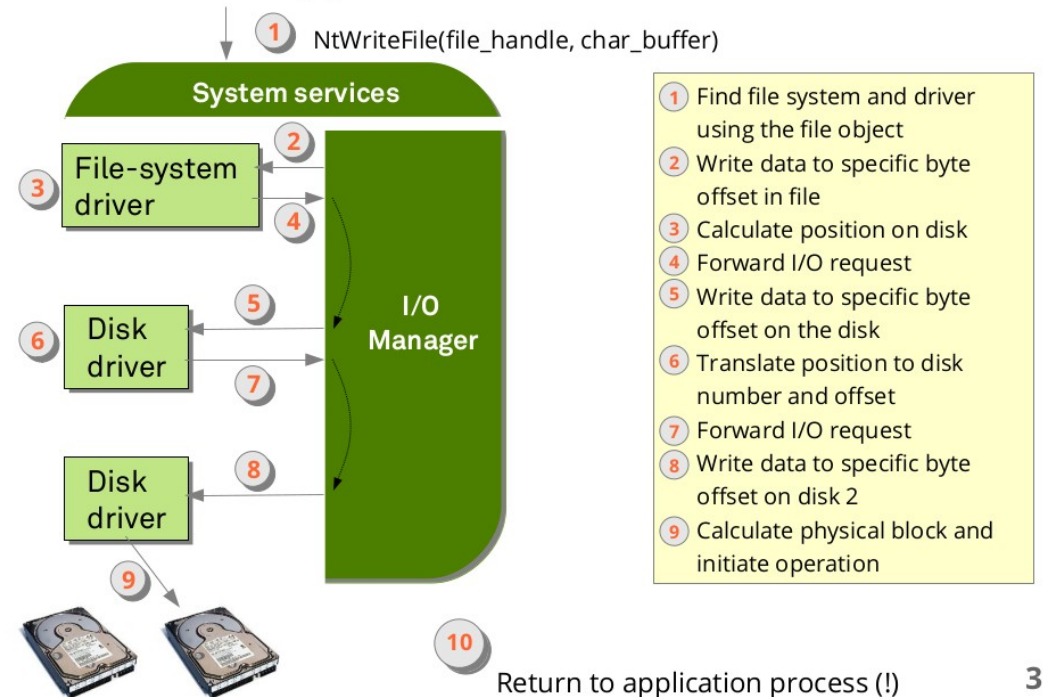
Three Core Areas

3. OS concepts in general and in Linux/Windows

Linux' Modular Scheduler

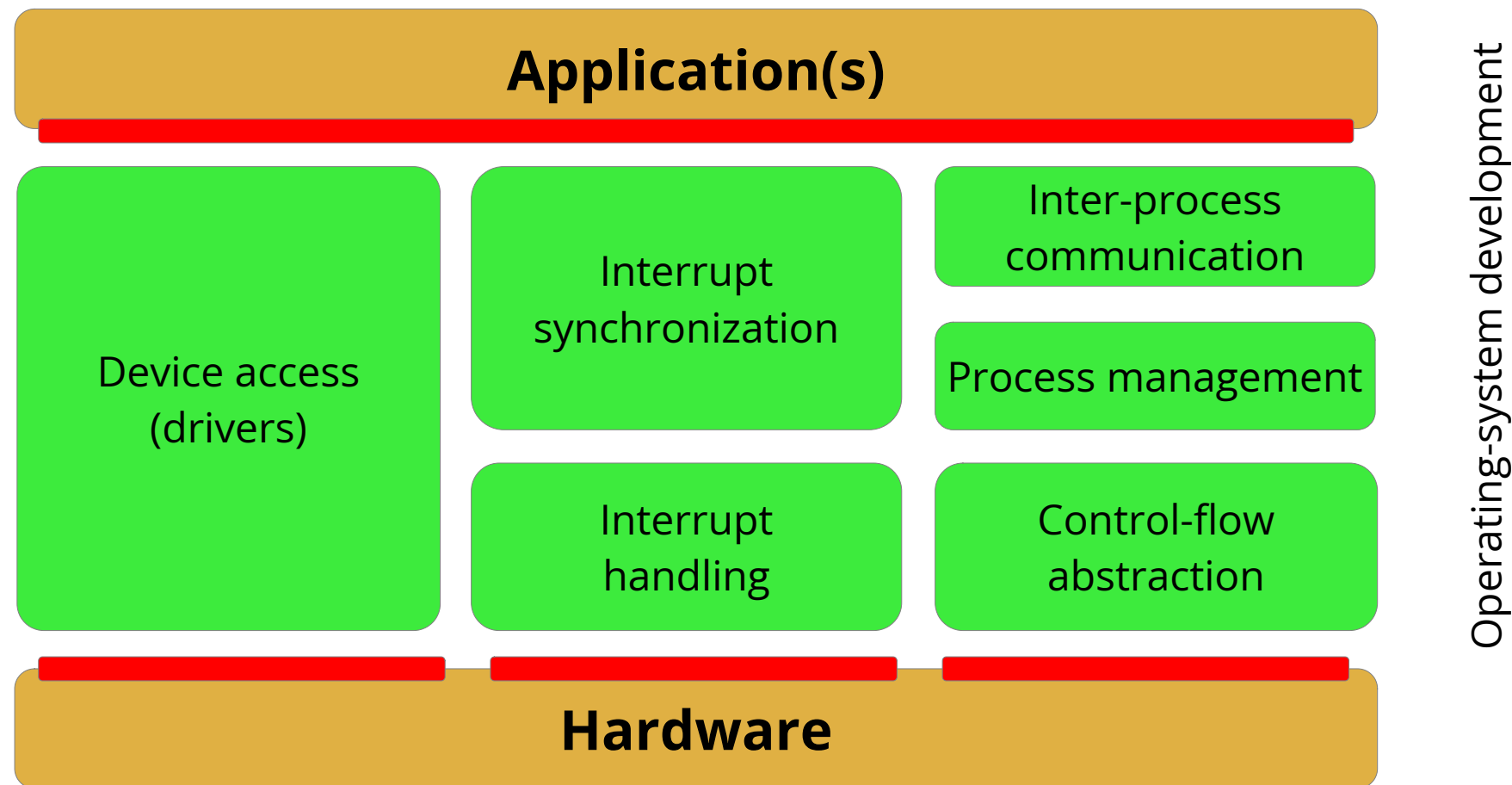


Windows – Typical I/O Procedure



... 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 (1)

- **Contents / Preparation**

- 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 (2)

- **Exam appointments**

- From August 17th
- Appointments: e-mail to os@mailbox.tu-dresden.de, include:
 - Matriculation number
 - Degree program (Master CS, Bachelor/Master Informatik, DSE, IST, ...)
 - Module name (e.g., INF-B-510)
 - “OSC w/ Prof. Schirmeier”
 - Date wishes + constraints (e.g., “between X and Y”), don’t be too restrictive
 - (for INF-BAS4 / INF-VERT4: list other lectures + examiners)
- Expect delays due to second-examiner sync. (*Komplexprüfung*), vacation period, etc.
- Withdrawal until 3 days before the appointment –
make sure to deregister with selma **AND** to notify os@mailbox.tu-dresden.de!

Exam (3)

- **Exam Procedure**

- Closely **listen** to the question (ask if it was unclear!)
- Answer the question as **completely and precisely** as possible
- Feel free to **anticipate** follow-up questions
- If applicable: Use pen & paper (provided by us), make examples, refer to your OOSTuBS implementation, ...
- 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

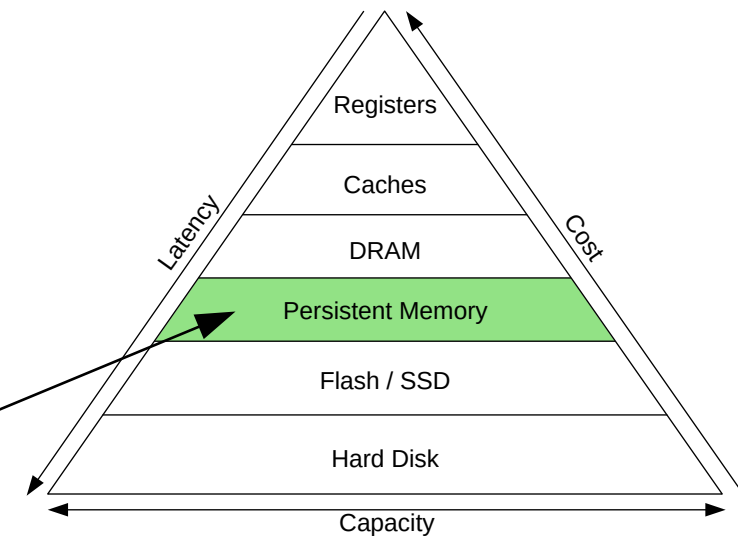
- Almost as fast as DRAM
- *Maintains its state if turned off*
- Available for servers since 2019
 - Optane DCPMMs (discontinued)

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?*



Source: Intel



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

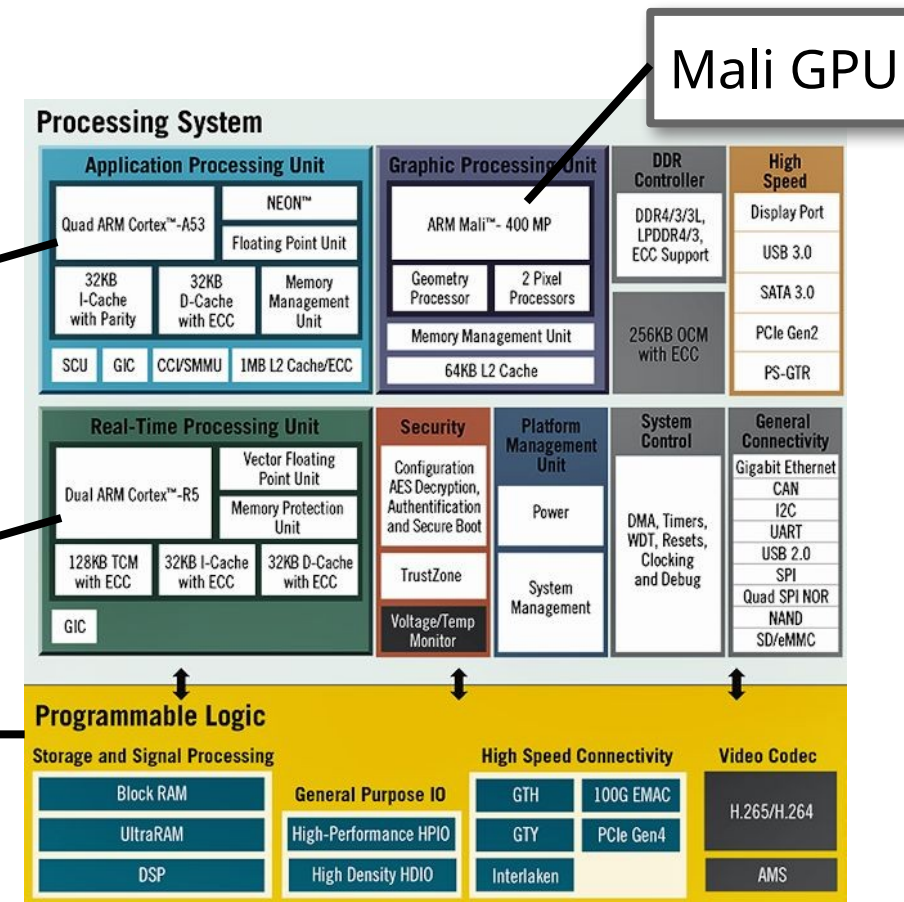
Challenge: Hardware Heterogeneity

- 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

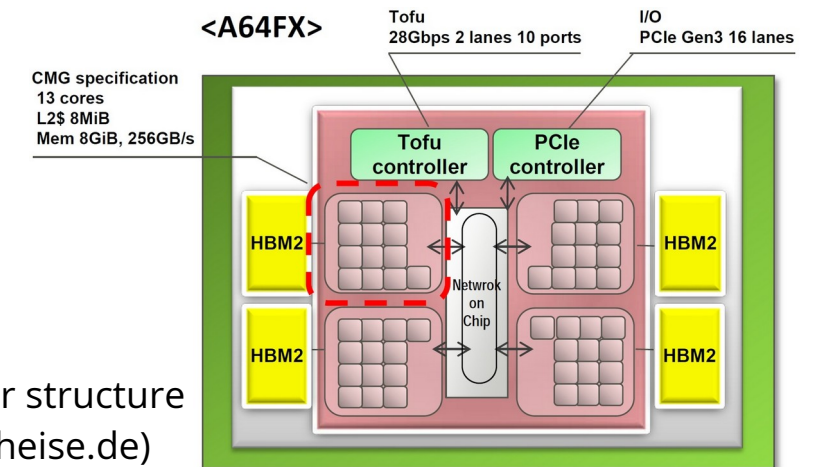
- Example: Fujitsu A64FX (#1 top500.org)
 - 48+4 cores (2.7 Tflops) per chip
 - 4 High-Bandwidth Memories (1 TB/s); 4 NUMA regions
 - 7.299.072 cores in the supercomputer
 - *Remote DMA* (RDMA) for communication



Fugaku supercomputer (Source: Fujitsu)

Research Questions:

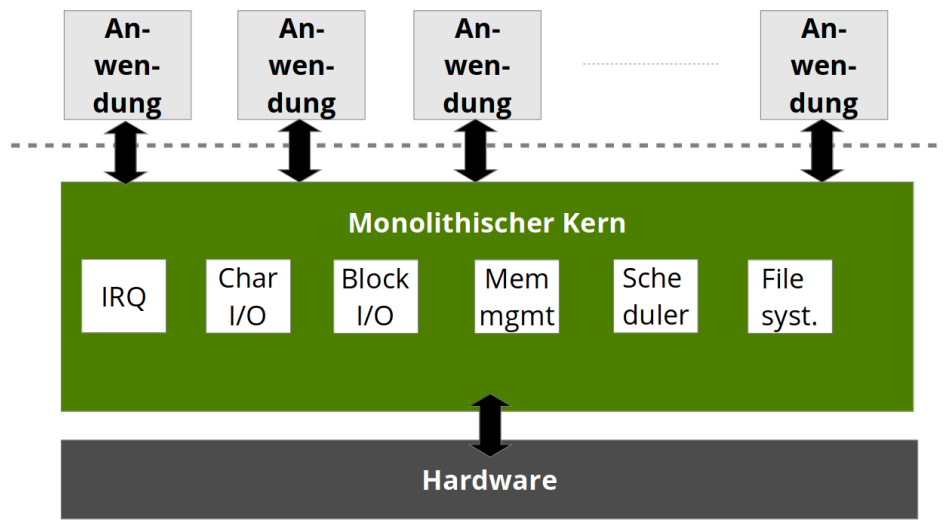
- Do we still need CPU multiplexing?
- How to place control flows and data objects?



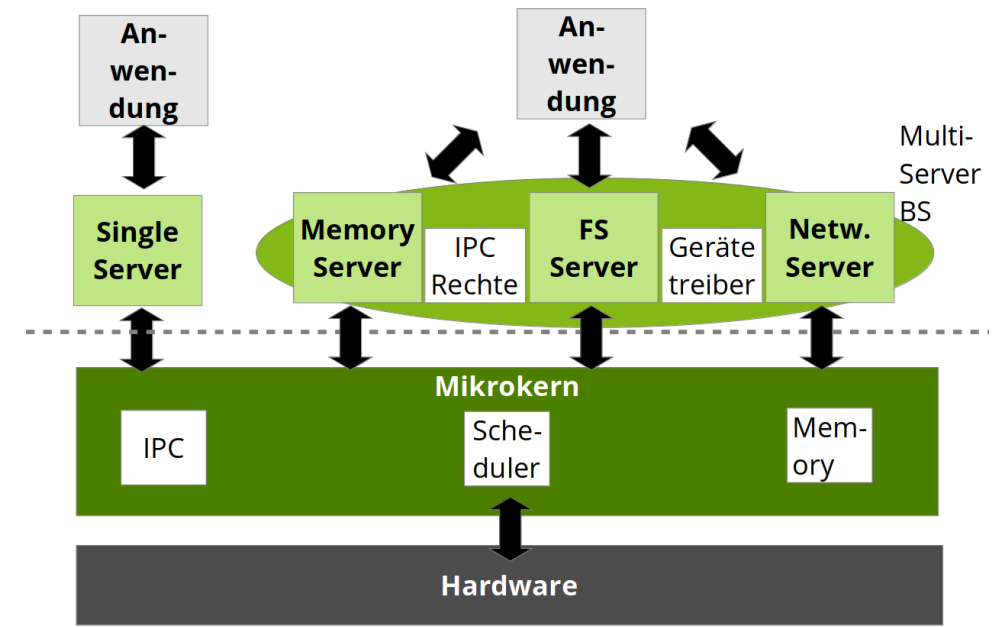
Research Topics at the OS Chair

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

Complexity: Monolith vs. Microkernel



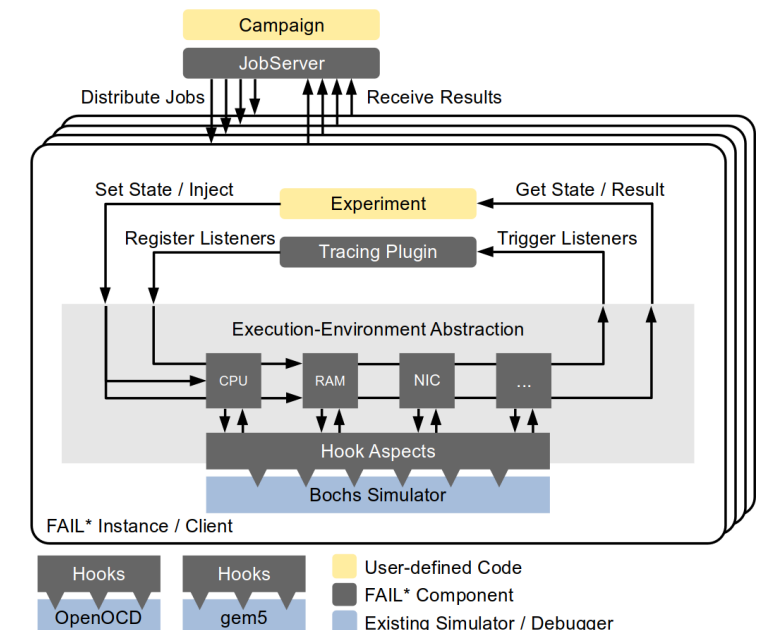
- Fine-grained locking is error-prone: **LockDoc** project
- *Security*: Take over a kernel component = Game Over
- but: Performance, lots of legacy code



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

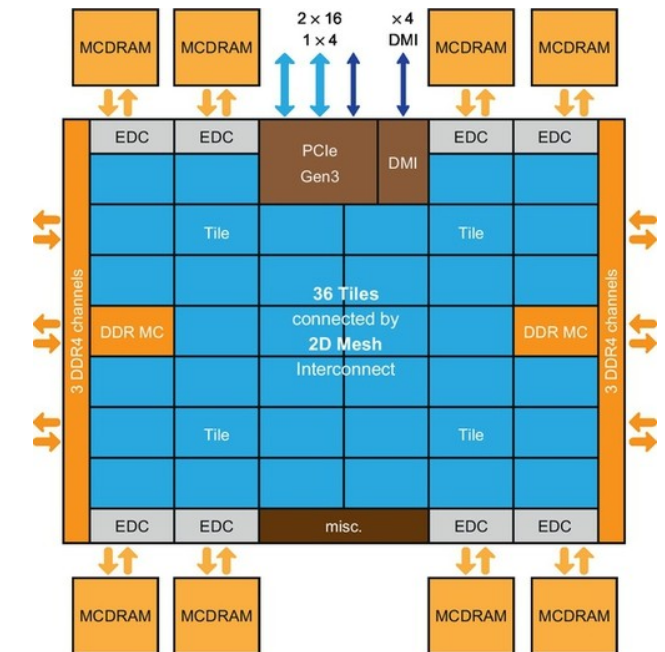
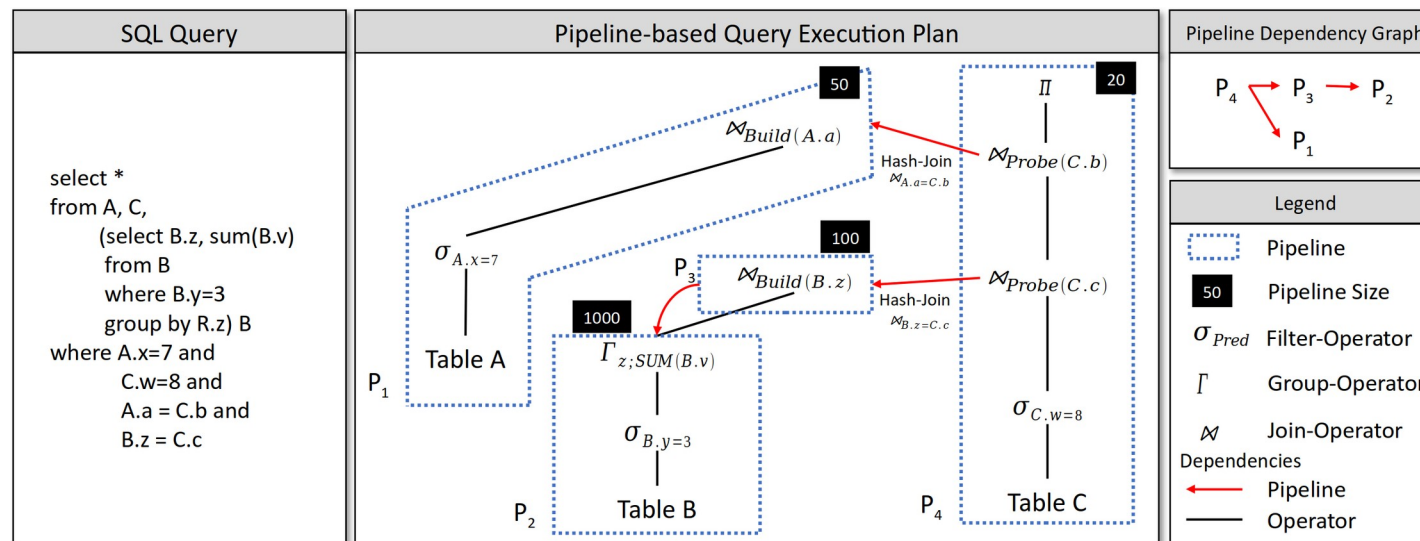
Fault-Tolerant Operating Systems

- *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



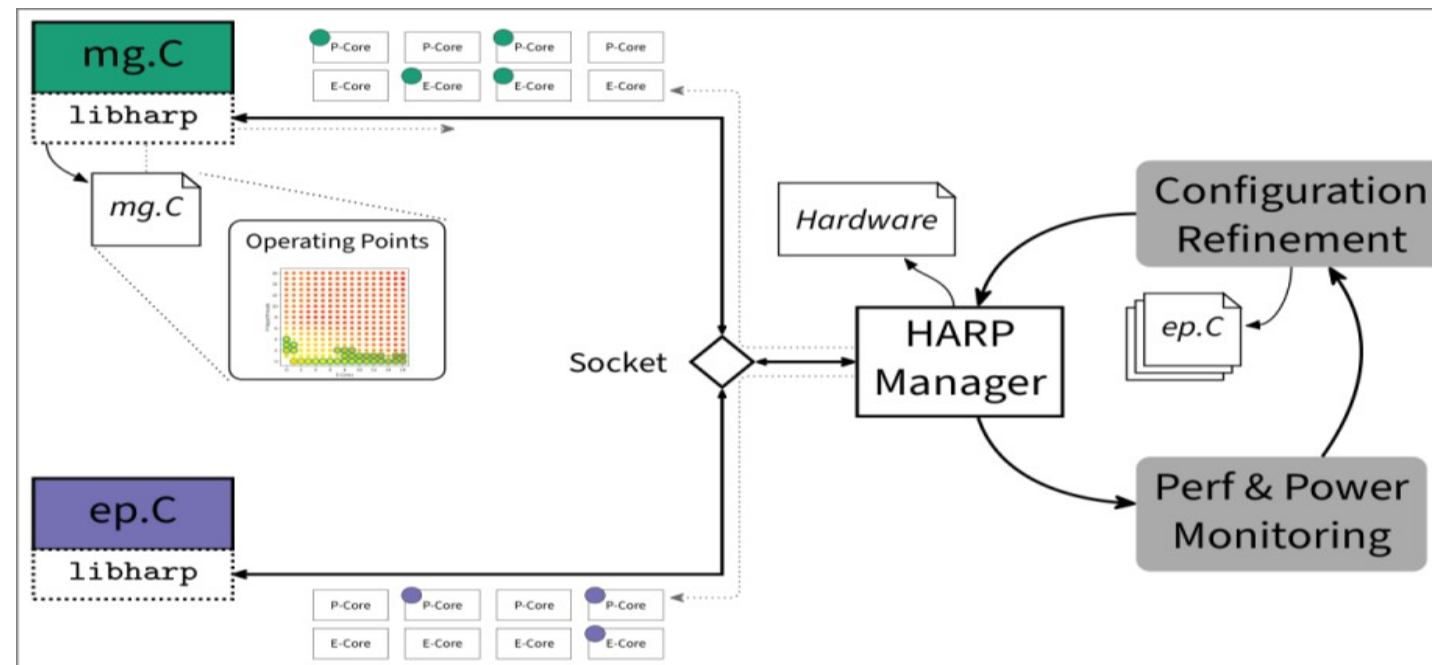
Disruptive Memory Technologies

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



Energy-Aware Resource Management

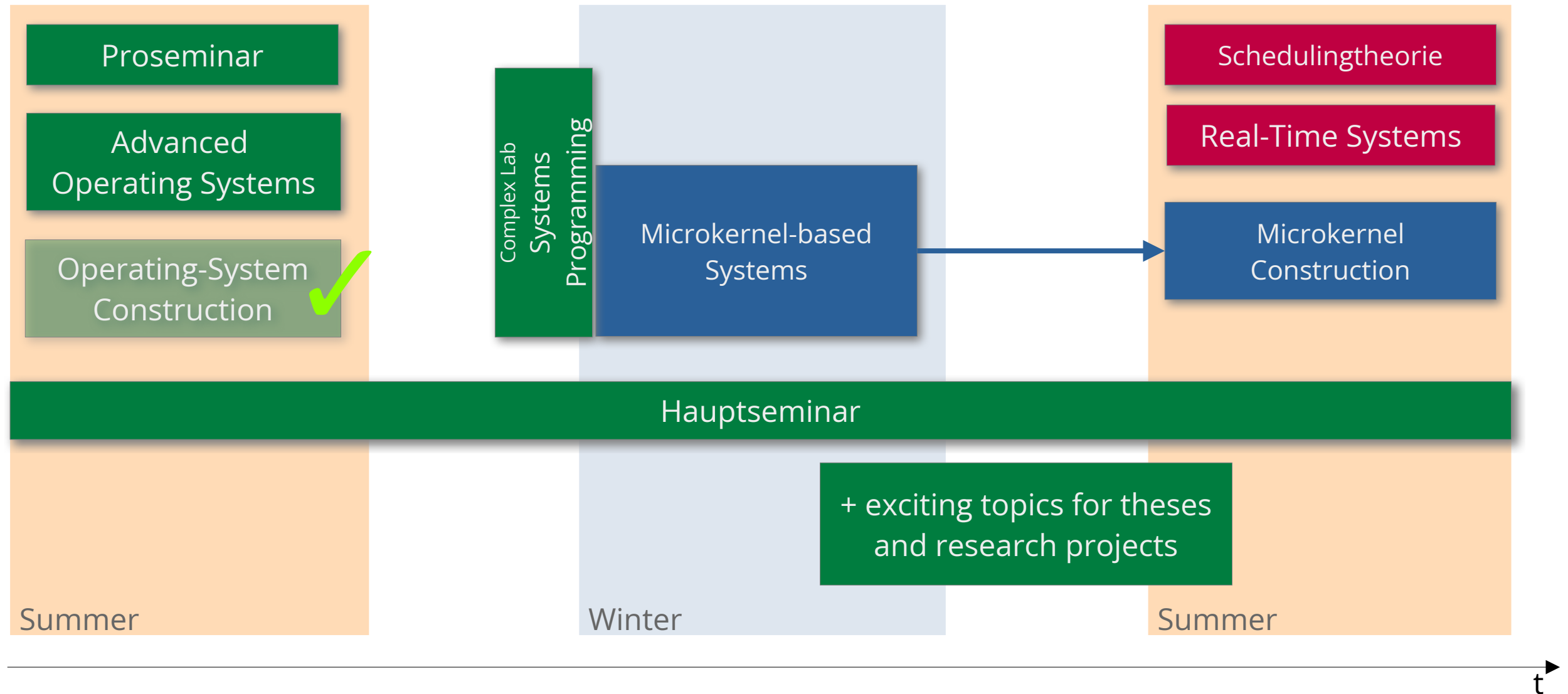
- How to distribute tasks efficiently in a modern system?
 - Understand and integrate hardware properties → e.g. core types (P-cores vs. E-cores)
 - Understand and integrate application behavior → e.g. core preferences, resource sharing, ...
- Find optimal solution at runtime (or at least as close as possible)



Agenda

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

Other Lectures



Thesis Topics

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

- **Fl4mer**: Application Tracing via Intel PT on L4Re
- **TEECore**: Core Isolation via Cache Separation
- **NUMA-HARP**: Dynamic Congestion Detection via PEBS
- Dynamic Adaptive Polling in MPI
- Distributed Gang Scheduling for HPC Applications



More on our website

<https://tu-dresden.de/ing/informatik/sya/professur-fuer-betriebssysteme/studium/abschlussarbeiten>

That's It!



Thank you!
I hope we'll meet again.



Don't forget:
Task #7 Contest
tomorrow, Wednesday 2026-07-21 09:20 (E023)
Voting only possible in presence.
(Possibly w/ live stream, but no recording this time.)