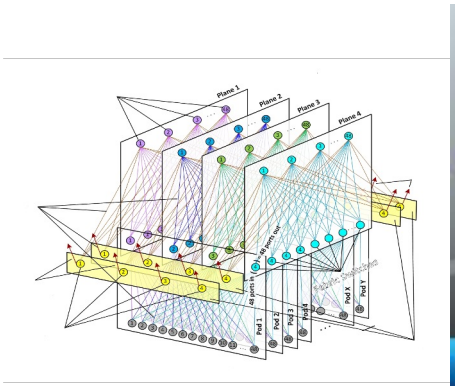
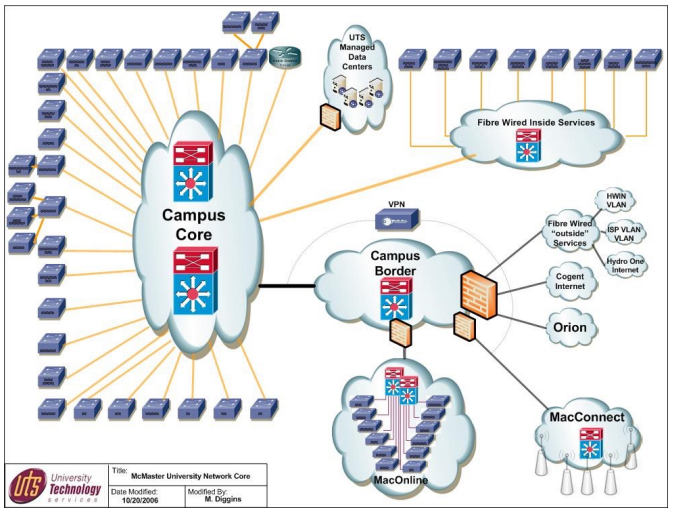
UNIVERSITEIT VAN AMSTERDAM



Large Systems:

Design + Implementation

2024-2025



➢ Week1-L1: Virtualization- Part 1

Shashikant Ilager
shashikantilager.com

12 december 2024

Virtualization

- What is Virtualization
  - *"a technique for hiding the physical characteristics of computing resources from the way you interact with those resources."*
  - Virtualization is the process of making things more abstract in order to make them easier to use.

Examples?

Storage virtualization

- Files
  - Linear sequence of bytes
  - Instead of blocks on a disk (or magnetic particles)
- Disk partitioning
  - One disk appears to be multiple parts
- Logical Block Addresses
  - Blocks addressed by a number, not physical Cylinder/Head/Sector
- RAID
  - redundant array of independent disks
- Logical Volume management
  - Combines disks and partitions into logical disks.

Network virtualization

- VLAN
  - Multiple logical networks on same physical wires
- Channel bonding
  - multiple links combined offered a single, higher-bandwidth link
- Computer clusters
  - multiple discrete computers into larger metacomputers
  - e.g. Hadoop
- Virtual NICs and bridges for VM communication

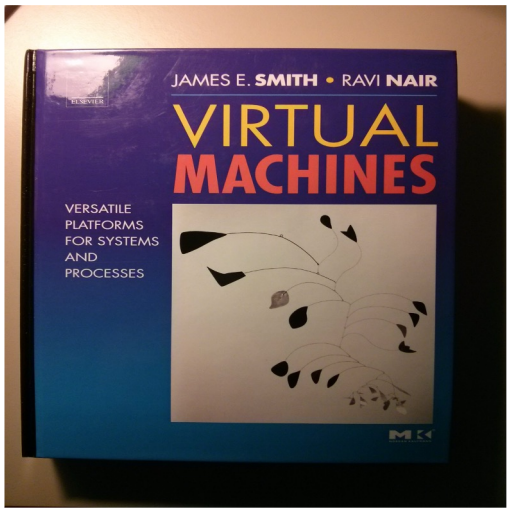
Resource virtualization

- Multiprogramming
  - Each process thinks it has a CPU to itself
- Virtual memory
  - Present linear address space composed of non-consecutive blocks of:
- Physical memory
  - Disk space

Credits

Slides largely based on:

- "Virtual Machines: Versatile Platforms for Systems and Processes"
  - James E. Smith
  - Ravi Nair
- Morgan Kaufmann Publishers -05



Why Virtual Machines?

- Isolate applications in separate VMs
- Sandbox applications for security
- Support different OSes concurrently (E.g., Running Windows, and Linux on the same physical machine-> Simultaneously)
- Legacy applications on legacy OSes
- Application testing using VMs with known state
- Testing OS upgrades, training
- OS development

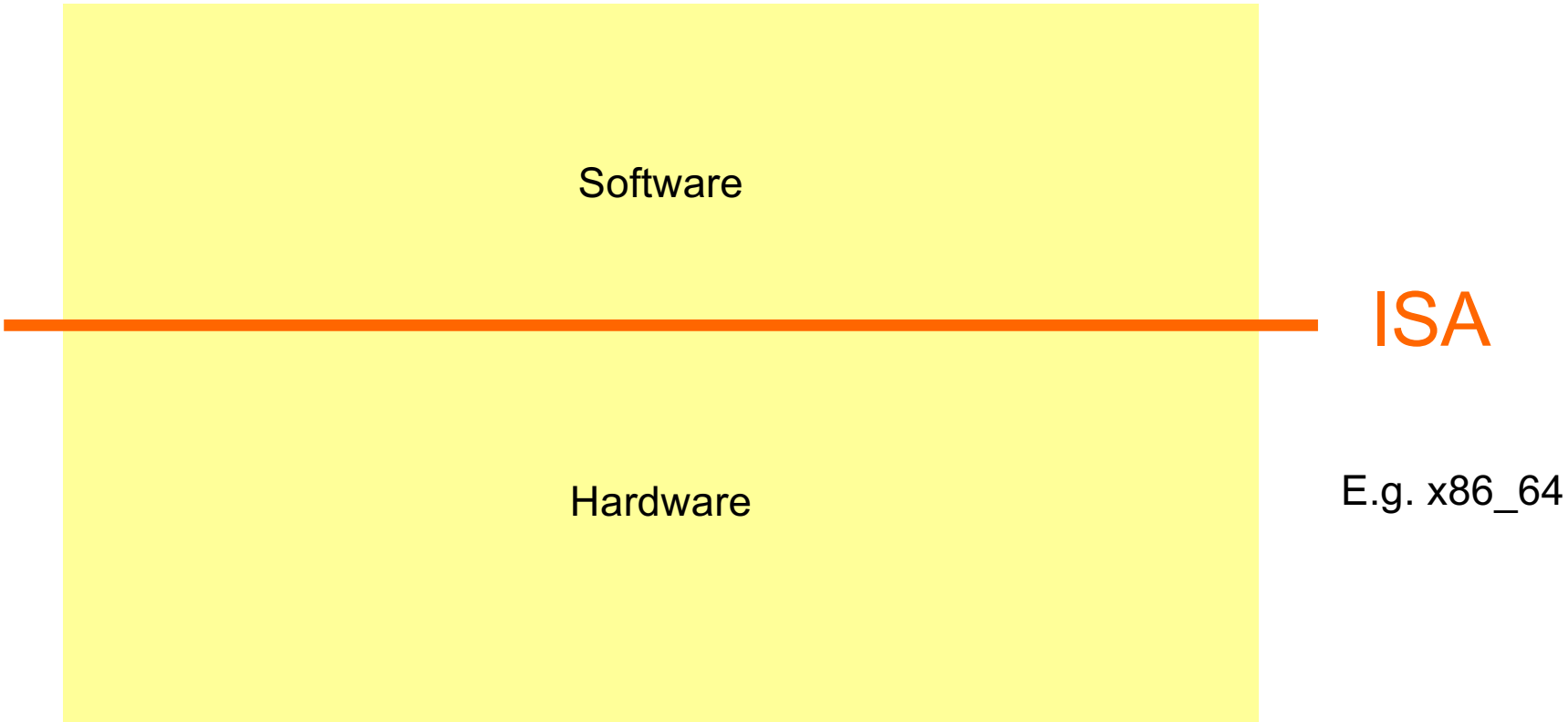
Why Virtual Machines?

LS Context:

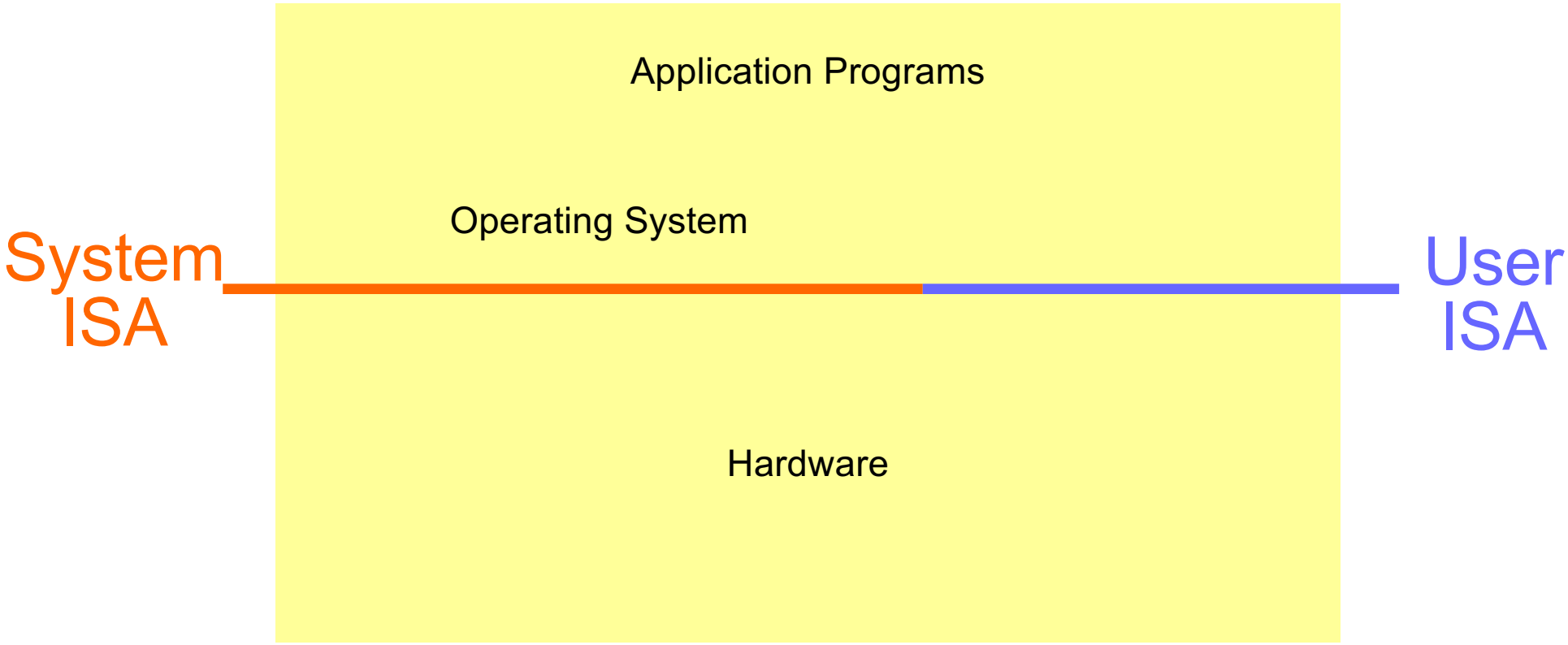
- Basis for cloud computing
  - Spin-up capacity on demand
- Resource utilization
  - "Server consolidation"
- Facilitate maintenance

Computer Architecture
Recap

Instruction Set Architecture



User + System ISA



User ISA

= For doing computations

- Simple Memory Instructions
  - Move data from memory to registers and v.v.
- Integer Instructions
- Floating-Point Instructions
- Branch instructions
  - Jump to address
  - Jump to address if ...

Part of SHA1 in assembly:

```
addl    %esi, %e;
movl    %c, %esi;
xorl    %d, %esi;
andl    %b, %esi;
xorl    %d, %esi;
```

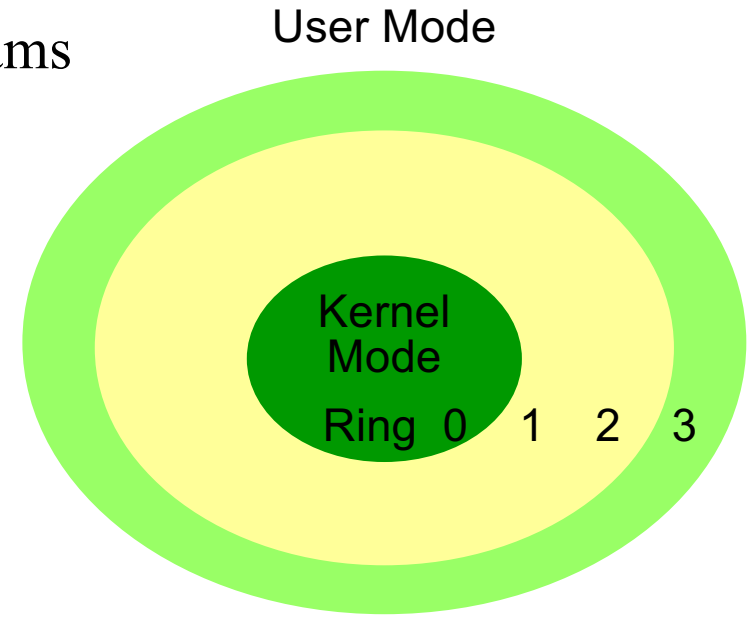
System ISA (1/2)

= Management of system resources

- System Resources:
  - Main memory
  - Storage
  - Other I/O devices
- Management:
  - Fair allocation between user programs
  - Prevent concurrent/unauthorized access
- Role of the Operating System

System ISA (2/2)

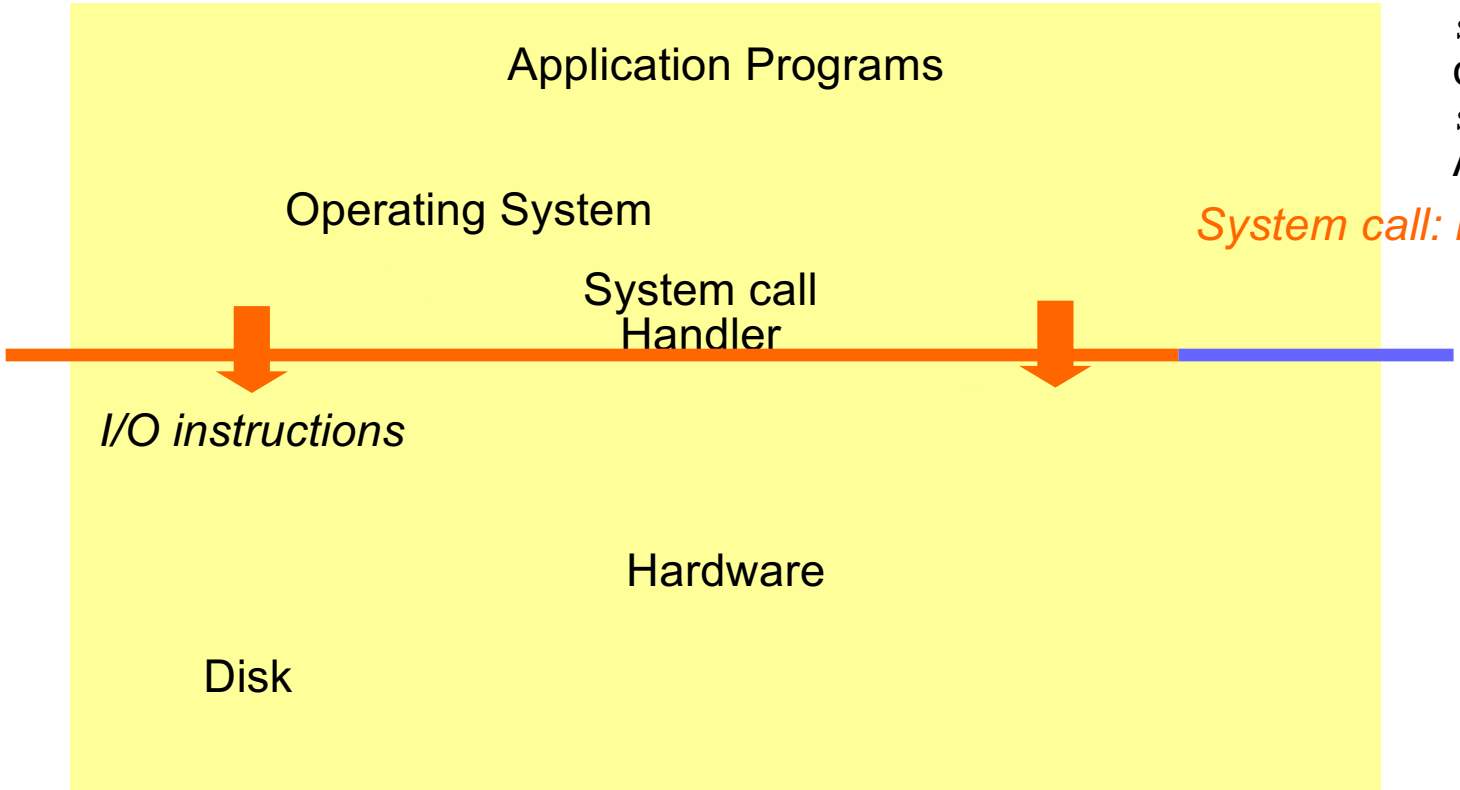
- OS requires special privileges over user programs
  - OS runs in CPU Kernel mode
  - Apps run in CPU User Mode
- x86: Implemented via
- 2 privilege levels/rings



System ISA Instructions

- Processor Management
  - "Change to user mode + run application"
  - Timer interrupt gives control back to the OS (Signals)
- Memory Management
  - Manage page table, TLB: virtual memory
- I/O Management
  - load and store to/from device
- Traps
  - "Change to kernel mode from application"
  - On purpose (system call) or on exception

System Call Instruction



E.g. `syscall` on x86 or `svc` on ARM.

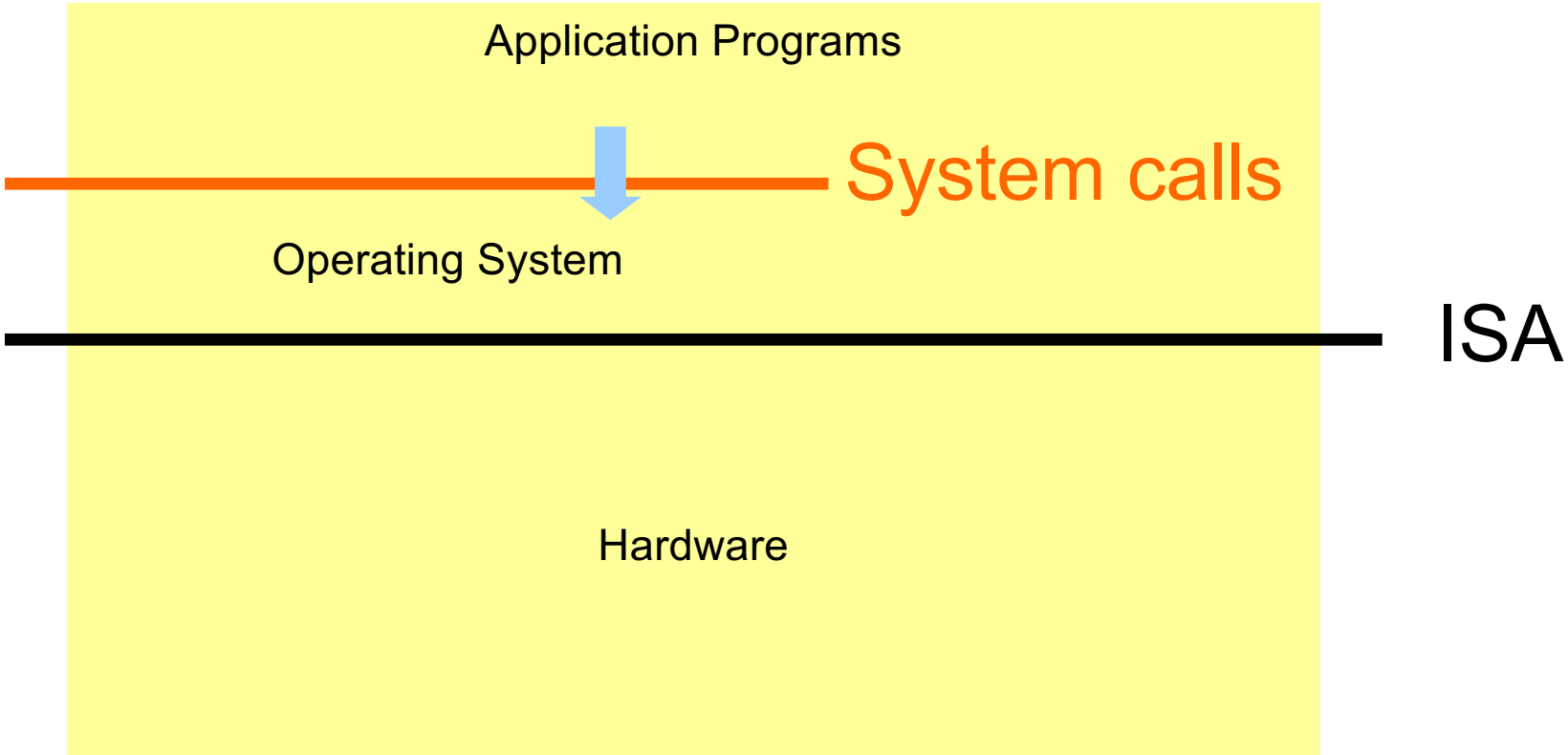
System call: read(file)

System Call

= Mechanism for User Mode code to request services from Kernel Mode

- What services?
  - Read/write to files and devices
  - Create processes
- i.e. use Operating System abstractions:
  - “File” abstraction for storing blocks on disk
  - “Process” abstraction for running different code in parallel

System Call Interface (1/2)



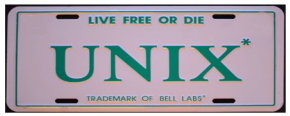
System Call Interface (2/2)

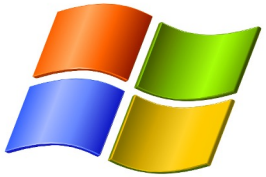
Read/write files or devices:

-open(filename, …)	CreateFile(...)
-read(fd, data, …)	ReadFile(...)
-write(fd, data, …)	WriteFile(...)
-ioctl()	SetConsoleMode(...)

Manipulate processes

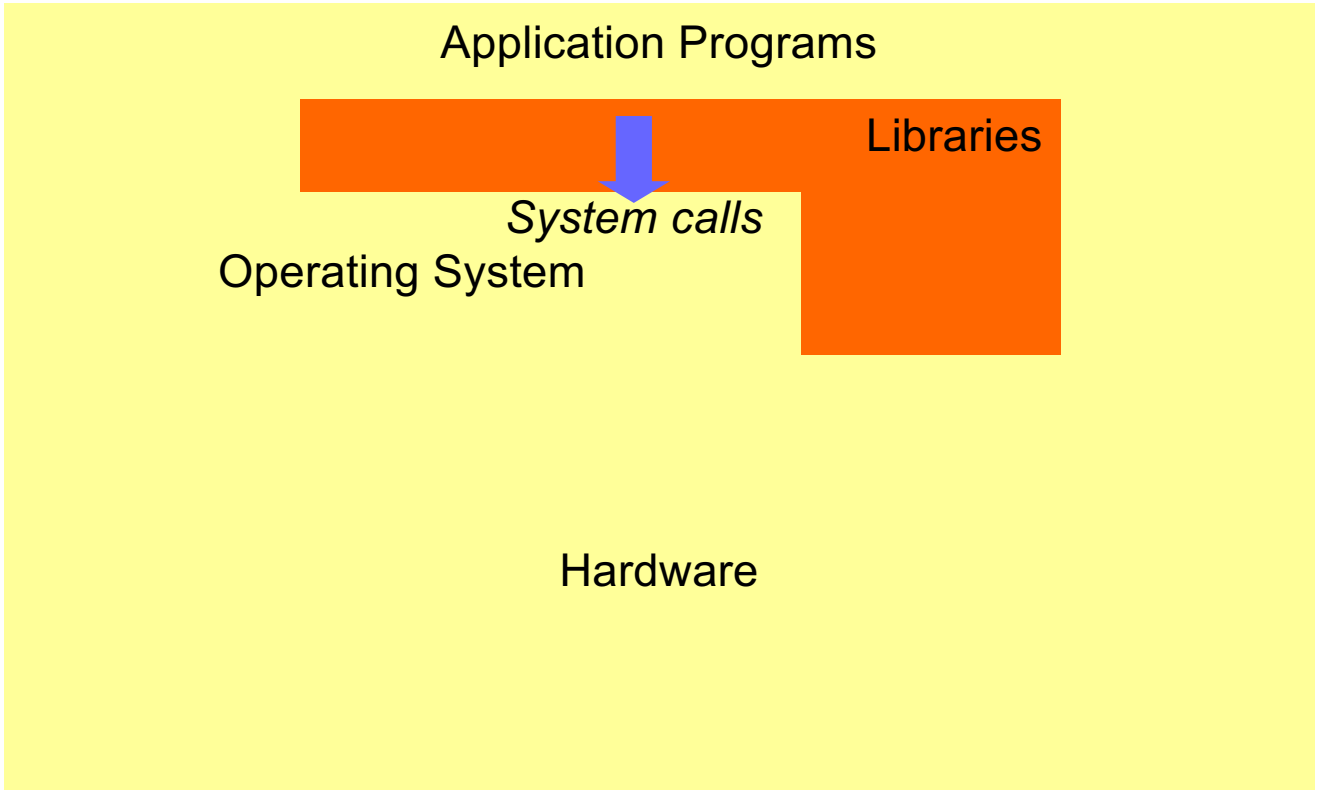
-fork()	CreateProcess(…)
-exit()	ExitProcess(…)





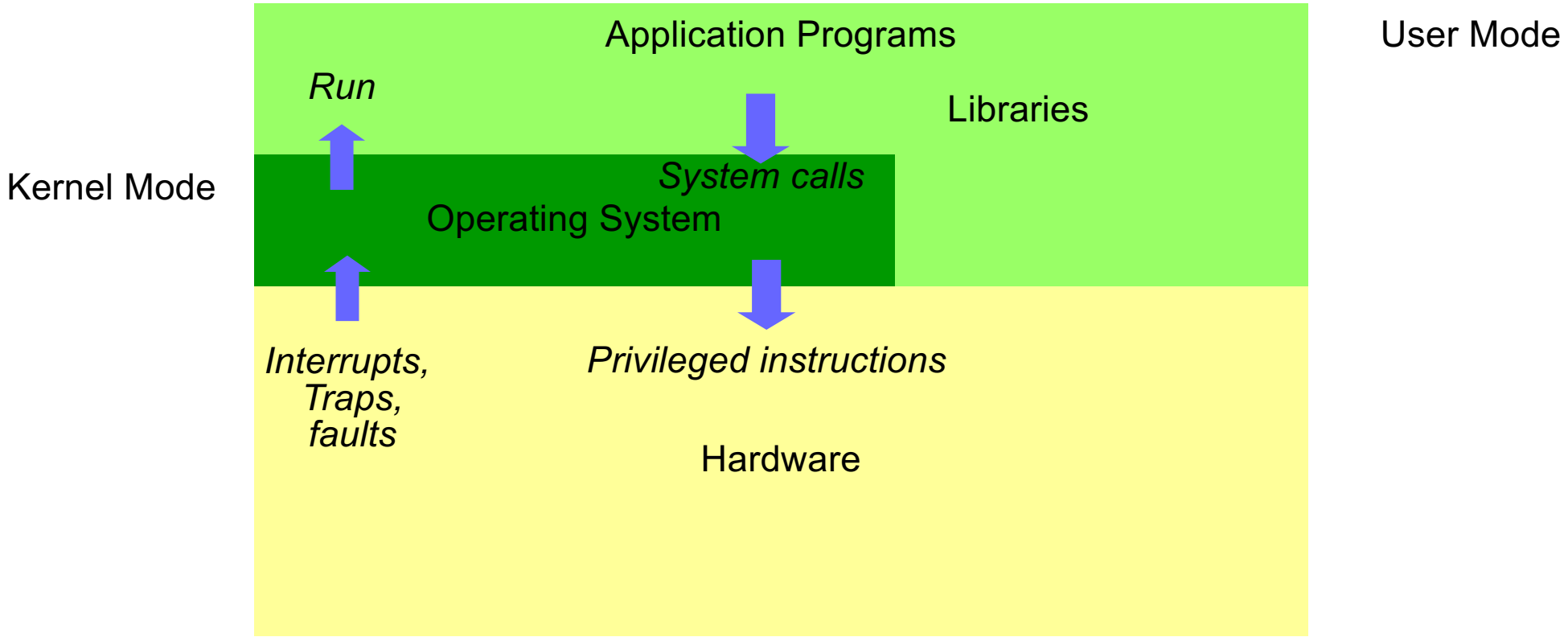
- More...

System calls via libraries

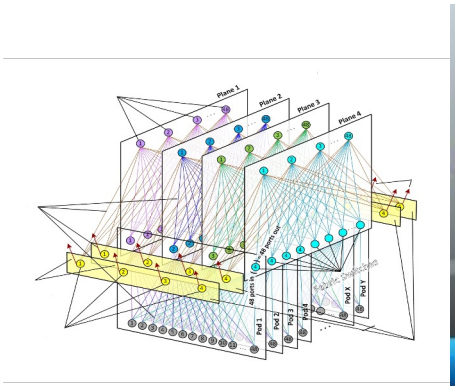


E.g. libc

Architecture Model



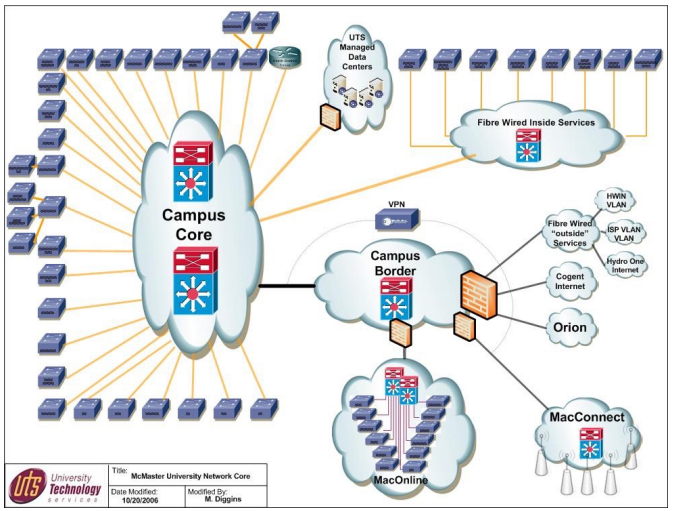
UNIVERSITEIT VAN AMSTERDAM



Large Systems:

Design + Implementation

2024-2025



- Week1-L2: Virtualization- Part 2

Shashikant Ilager
shashikantilager.com