mobile business

Lecture 10

Concepts of Mobile
Operating Systems

Mobile Business I (WS 2015/16)

Prof. Dr. Kai Rannenberg

Deutsche Telekom Chair of Mobile Business & Multilateral Security
Johann Wolfgang Goethe University Frankfurt a. M.

- Functions
- Processes
    - States and elements
    - Scheduling
    - Inter-Process-Communication (IPC)
- Memory Management
    - Mapping
    - Paging
    - Segmentation
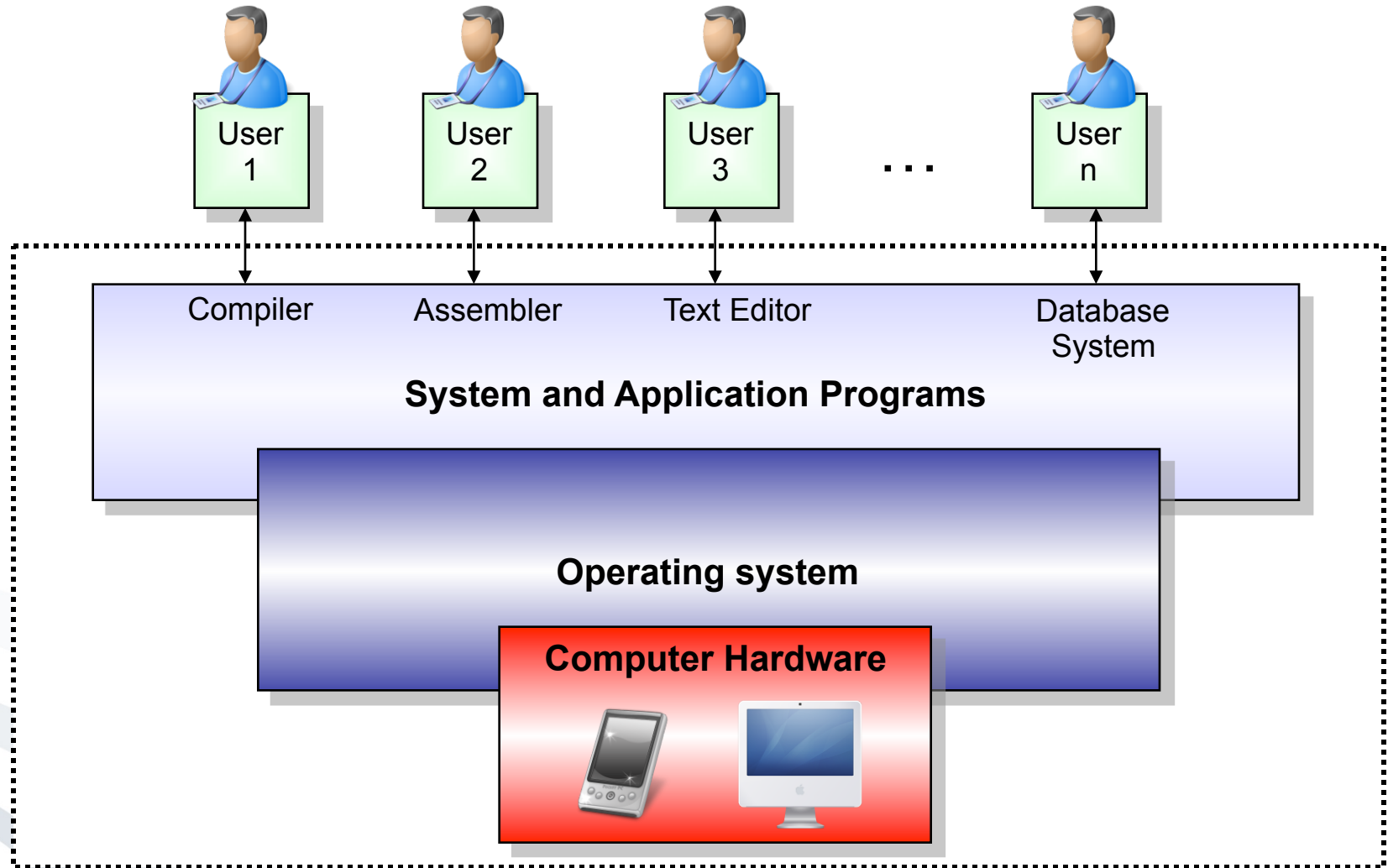   - Examples
- Security & Maintenance

What is an operating system (OS)?

- An OS is a program that serves as a mediator between the user and the hardware.

- It enables the users to execute programs

- *Other properties:* Multi-user, multi-thread, high availability, real-time, ...

- ***Primary goal of an OS***: Easy usage of the actual hardware

- ***Secondary goal of an OS***: Efficient usage of the hardware

[SilberGalvin1999]

[SilberGalvin1999]

- ## Functions

- ## Processes

  - ### States and elements

  - ### Scheduling

  - ### Inter-Process-Communication (IPC)

- ## Memory Management

  - ### Mapping

  - ### Paging

  - ### Segmentation

  - ### Examples

- ## Security & Maintenance

- # Controlling and sharing of resources
  - Computation time, real-time processing
    "Who is computing how much? How long does it take?"
  - Memory (RAM, Disk)
    "Who gets which part of the memory?"

- # Security functions
  - Protection of the data (memory, hard disk):
    "Who is allowed to access resources?"
  - Process protection (computation time, code, isolation):
    "Who is allowed to compute?"
  - Security module support

- # Communication
  - Allocation of I/O-Resources
  - Processing of the communication
  - User interface (UI)

- Functions
- Processes
  - States and elements
  - Scheduling
  - Inter-Process-Communication (IPC)
- Memory Management
  - Mapping
  - Paging
  - Segmentation
 - Examples
- Security & Maintenance

- Several programs (processes) can run simultaneously & concurrently on an OS:

    - *How are processes managed in a system with regard to processing time, memory, etc?*

    - *Which process is allowed to access resources when?*

    - *How are resources (I/O) shared among processes?*

    - *How do processes exchange data among each other?*

- A process is a program "in operation".

- A process uses resources, such as CPU time, memory, files, and I/O devices.

- The resources of a process are allocated while it is created or when it is running.

- The operating system has to manage the process (creation, resource distribution, etc.).

- More than simple code!

- Program counter: Indicates on which point in the code the process resides.

- Contents of the process registers:

  - *Stack:* Contains temporary data, such as subroutine parameters or return addresses, etc.

  - *Data section:* Contains the global variables

  - *Heap:* Dynamically allocated memory

[SilberGalvin1999]
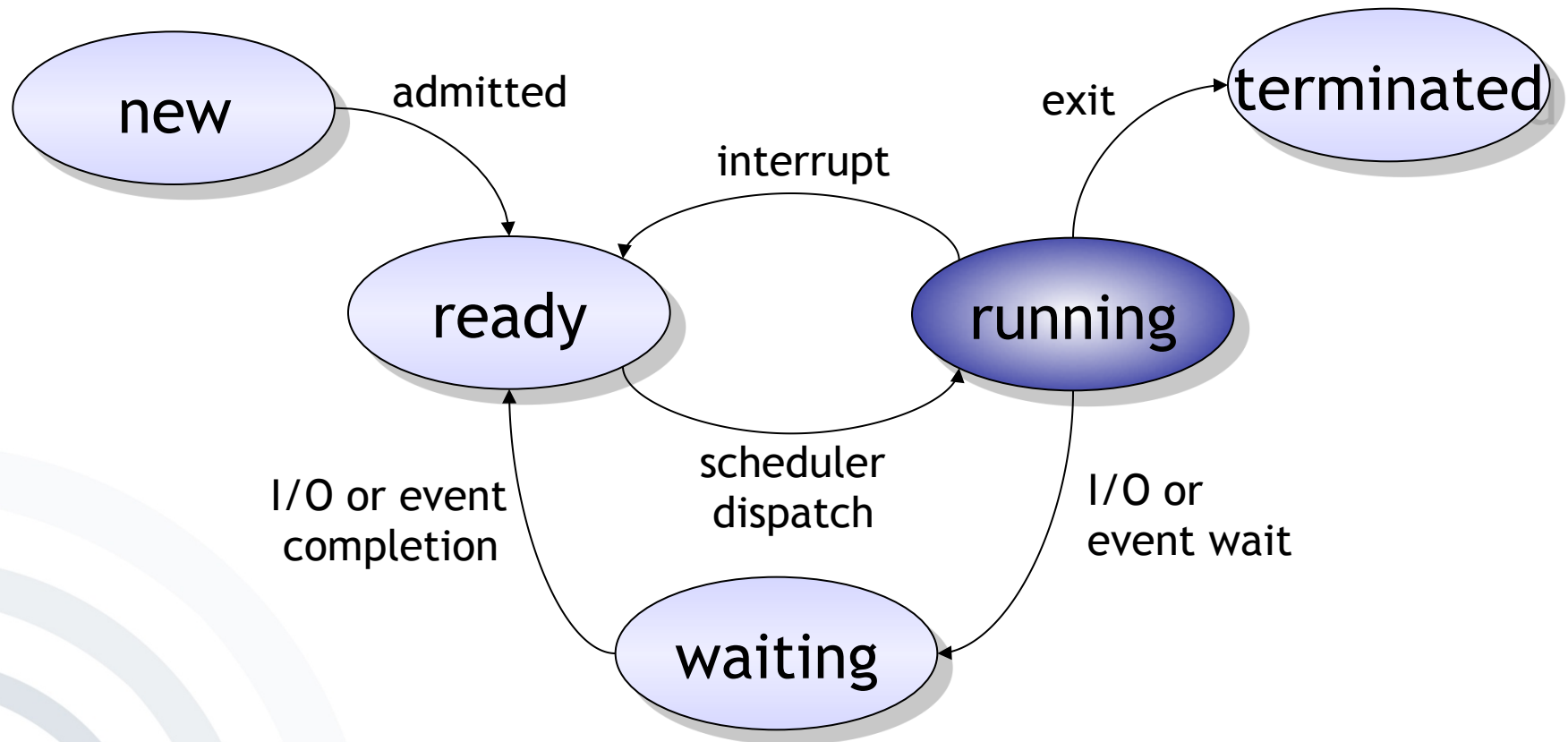
- **New:** Process is created.
- **Ready:** Process is waiting for being executed.
- **Running:** Process is running.
- **Waiting:** Process is waiting for results:
    - Completion of an I/O-operation
   - An event
- **Terminated:** Process is terminated.

- Abstracted representation of the contents of a process control block (PCB), needed by an operating system.

pointer	process state
process number	
program counter	
registers	
memory limits	
list of open files	
⋮	

[SilberGalvin1999]

13

- **Process State**: *new, ready, running, waiting, …*

- **Program Counter**: Address of the next command to be executed

- **CPU Registers:** Accumulator, Index Register, Stack Pointer and general registers

- **Information for:**

  - CPU-Scheduling

  - Memory-Management

  - Accounting

  - I/O Status

- **Multiprogramming**: Several processes are being run in parallel for:

    - Maximisation of the CPU usage

    - Enabling users to operate several programs simultaneously

    - Enabling several users to work on the same machine simultaneously

- On a CPU only one process is running at a time.

- The process switching must be fast, to enable the user to interact with all running programs.

- Queues are used to handle this task.

based on [SilberGalvin1999]

16
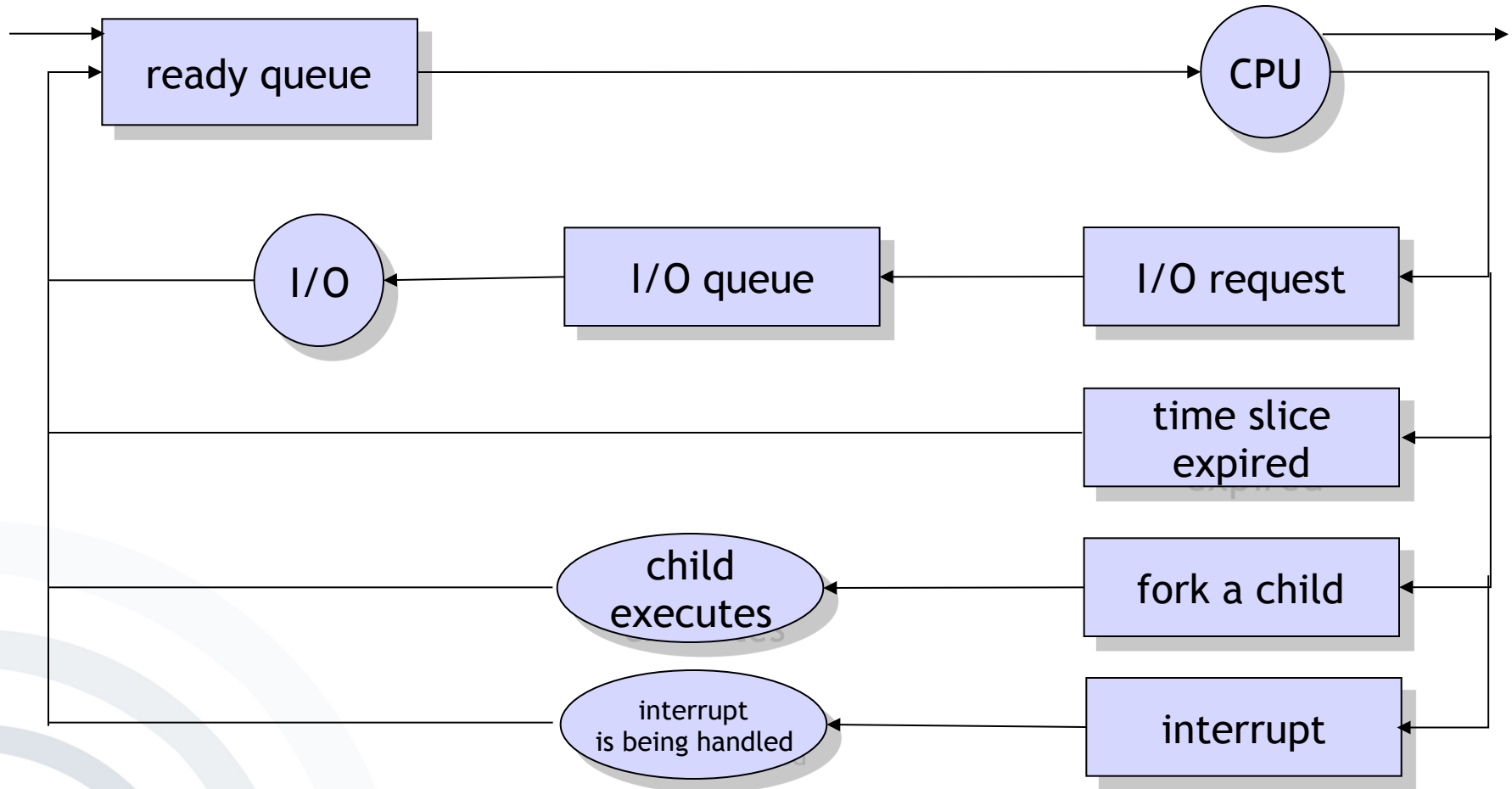
- If the CPU is idle (no process is running), the scheduler invokes a process from the ready-queue to be run on the CPU.

- There are different methods (algorithms) to make the choice, which process to invoke.

- Methods are optimised towards different criteria.

- **CPU utilisation:** The goal is to maximise the CPU usage.

- **Throughput:** Number of finished processes per time unit.

- **Turnaround-time**: Time interval between the beginning and the end of a process

- **Latency time**: Sum of all the waiting time of all processes in the queue.

- **Response time**: Time span of a process to answer a user's request and to generate the answer.

First Come, First Serve (FCFS)

- Processes are executed by the CPU one after another in order of their occurrence.

- FIFO-principles (First In First Out)

- ***Pros/Cons*:**

    - The throughput is not optimal.
    - Average response time is very high
    - No optimal utilisation of the CPU (Convoy-Effect)
    - Not appropriate for Time-Sharing-Systems

- The processes are executed in order of their execution time.

- Processes that can be finished fast are executed first.

- ***Pros/Cons:***

  - ***Optimal*** with regard to the average latency time
  - Not fair ➲ Complex processes can "starve to death".

- Processes get an assigned priority number.

- Process execution in the order of the assigned priority.

- Deadlocks or "starvation" of processes with low priority numbers is possible.

➲ Aging: Gradually raising the priority of a process

- Especially used for Time-Sharing-Systems and one of the simplest scheduling algorithms

- Similar to FCFS, assigning time slices of a time interval to a process being held in the scheduling queue.

- After the time slice of a process is expired, the CPU is revoked from the process and the process is placed at the end of the scheduling queue.

- The efficiency of this method depends on the size of a time interval.

- Processes are able to interact with each other (data exchange)

- Several methods are possible

  - Direct or indirect

  - Symmetric or asymmetric

  - Automatic or by explicit buffering

  - Handover as copy or reference

  - Fixed or variable size of communication packets

- Functions
- Processes
    - States and elements
    - Scheduling
    - Inter-Process-Communication (IPC)
- Memory Management
    - Mapping
    - Paging
    - Segmentation
   - Examples
- Security & Maintenance

- The CPU retrieves instructions from the memory depending on the program counter.

- Thereby it might be necessary to read or write data from certain memory-addresses.

- The operating system has an address space where the data and the programs reside.

- The whereabouts of a process in the memory can be unknown by the time of the actual programming.

➲ Usage of symbolic addresses during programming that get mapped to physical addresses later on (*Mapping*)

➲ *Binding:* The conversion of symbolic addresses to logical addresses in the memory of an operating system.

25

- **Functions**
- **Processes**
  - States and elements
  - Scheduling
  - Inter-Process-Communication (IPC)
- **Memory Management**
  - Mapping
  - Paging
  - Segmentation
  - Examples
- **Security & Maintenance**

- **Logical Addresses:**
  Generated by the CPU

- **Physical Addresses:**
  Sent to the memory unit

- The mapping is done by a so called MMU (Memory Management Unit).

- Usage of a relocation register that contains the base address for a process.

- The base address is added to the logical address, resulting in the physical address.

relocation
register

14000

logical
address

CPU

346

+

MMU

physical
address

14346

memory

[SilberGalvin1999]
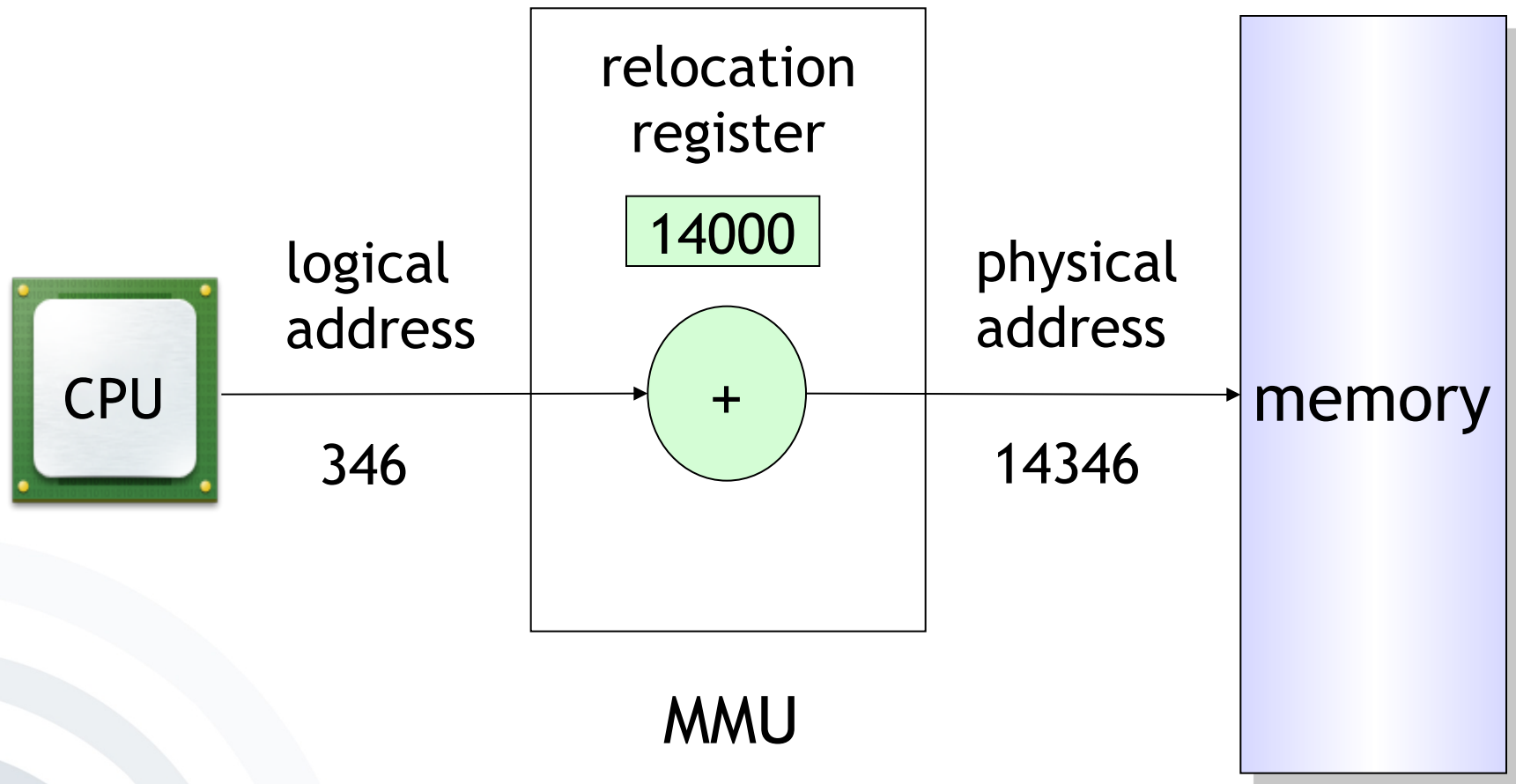
- The memory of a system also contains the actual operating system.

- The access of other processes onto the code of the operating systems needs to be prevented.

- Furthermore, the processes need to be protected against each other.

- Solution: Usage of so called „Limit Registers"
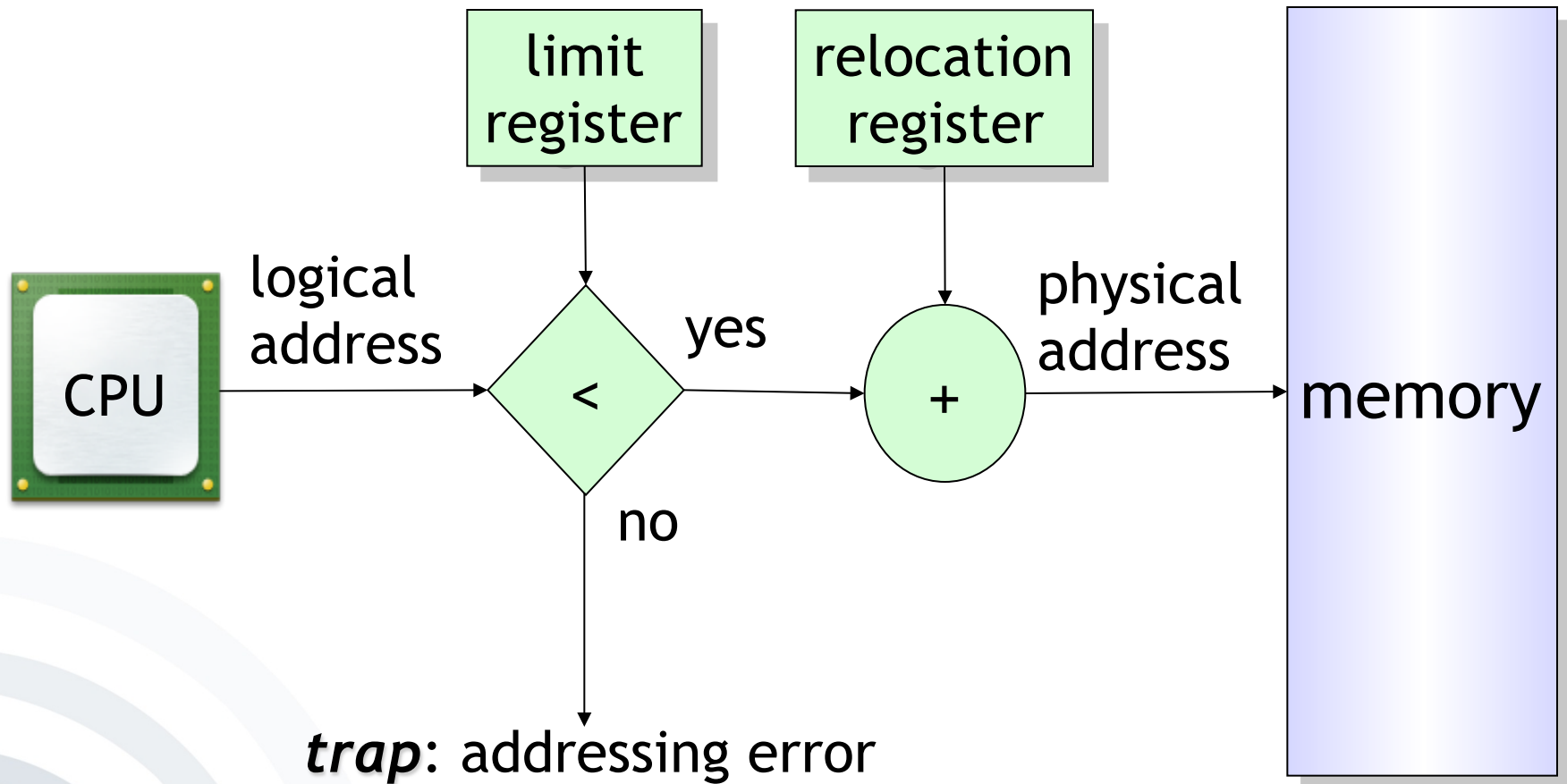
mobile business

limit register

relocation register

CPU

logical address

< yes

+ physical address

memory
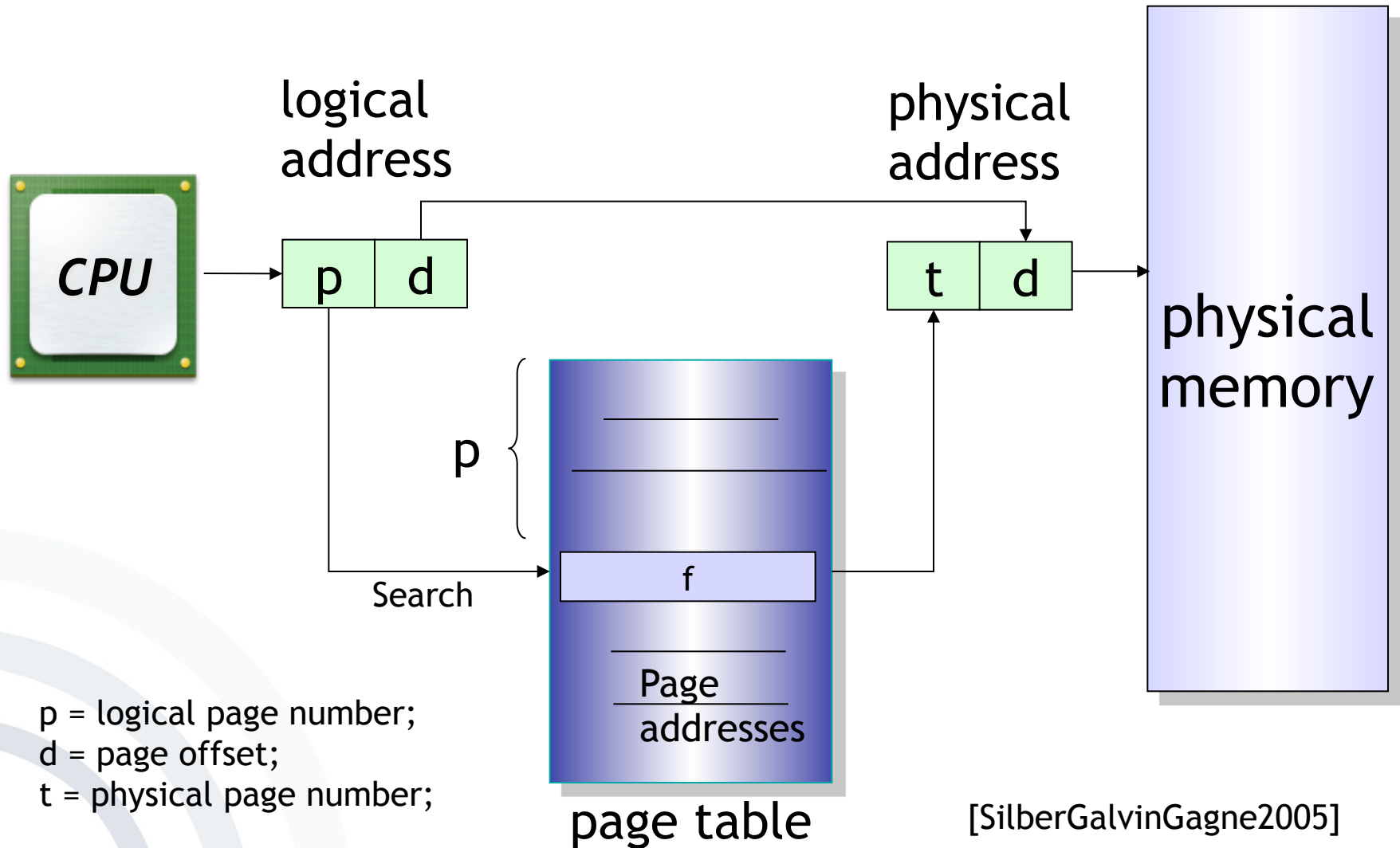
no

trap: addressing error

- **Functions**
- **Processes**
  - States and elements
  - Scheduling
  - Inter-Process-Communication (IPC)
- **Memory Management**
  - Mapping
  - Paging
  - Segmentation
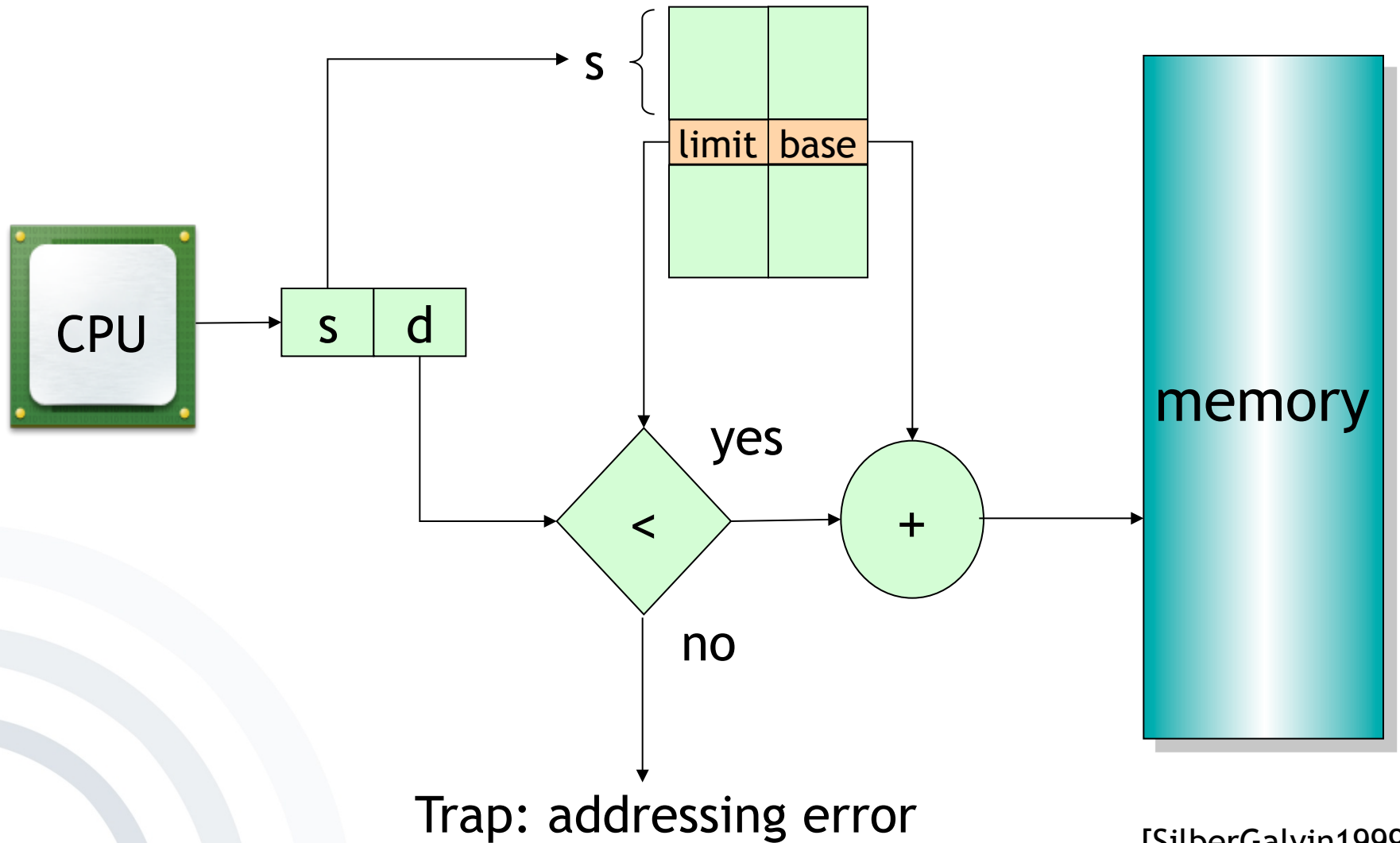 - Examples
- **Security & Maintenance**

- The memory contains several processes of varying size.

- When a process is loaded or removed from the memory, the free memory will be fragmented.

- One solution is the so called *paging*, putting the process into several separate memory chunks of a defined size, instead of putting it into the memory in one single piece.

- The *physical memory* is divided into blocks of a defined size, the so called *frames*.

- The *logical memory* gets divided into blocks of the same size, the so called (memory) *pages*.

- Every address created by a CPU is divided into a *page number* [p] and an *offset* [d].

- The page number is used as the index for the page table, containing the base address for all (memory) pages.

- The base address is combined with the offset resulting in the physical address.

logical address

physical address

CPU

| p | d |

| t | d |

physical memory

p {

| f |

Search

Page addresses

page table

p = logical page number;
d = page offset;
t = physical page number;

[SilberGalvinGagne2005]

- Functions
- Processes
  - States and elements
  - Scheduling
  - Inter-Process-Communication (IPC)
- Memory Management
  - Mapping
  - Paging
  - Segmentation
  - Examples
- Security & Maintenance

- The memory is partitioned into segments of variable length.

- Every segment has a name and a defined length.

- A segment table is used to store the base address and the limit of the segments.

- The logical address consists of a segment number [s] and the offset [d].

mobile business

CPU

s	d

s {

limit	base

<

yes

no

+

memory

Trap: addressing error

[SilberGalvin1999]

- Functions
- Processes
  - States and elements
  - Scheduling
  - Inter-Process-Communication (IPC)
- Memory Management
  - Mapping
  - Paging
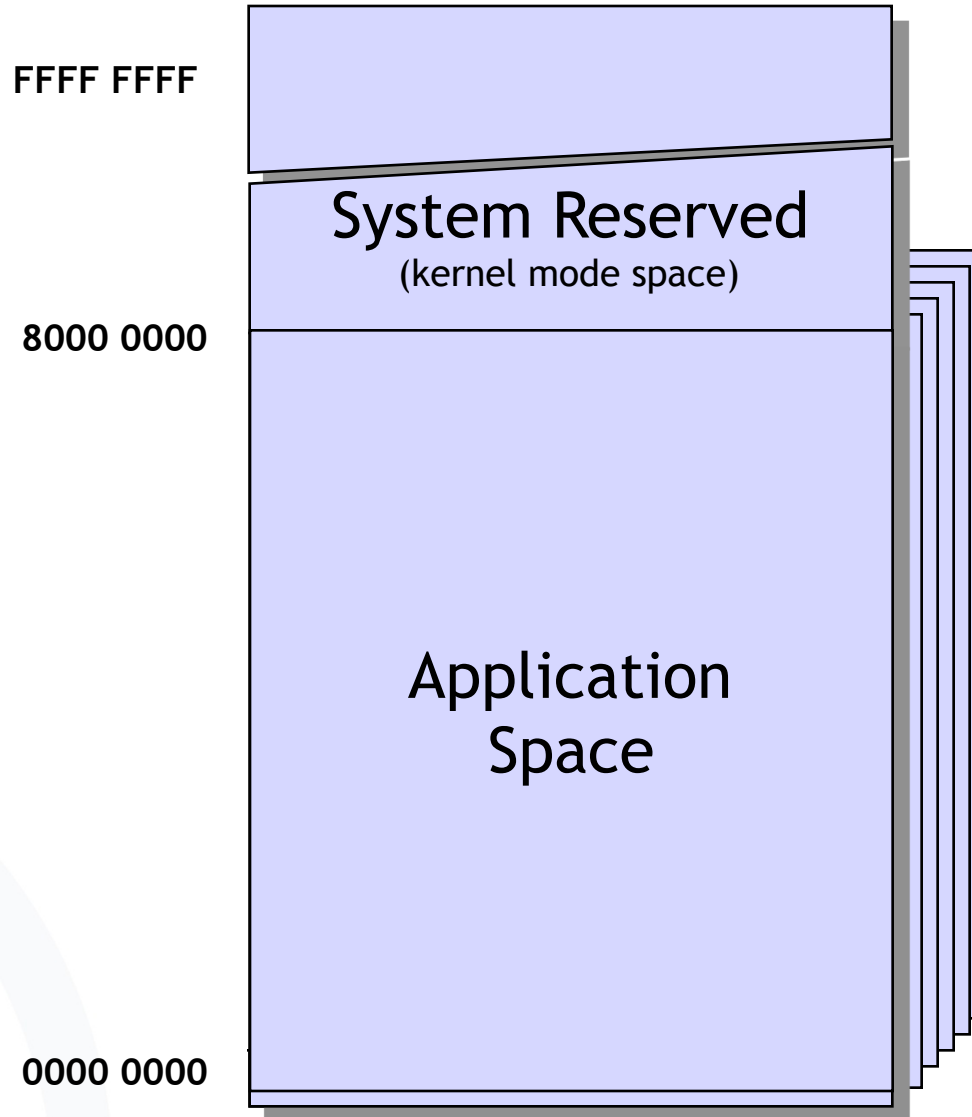 - Segmentation
  - Examples
- Security & Maintenance

mobile business

FFFF FFFF

System Reserved
(kernel mode space)

8000 0000

Application
Space

0000 0000

[Hall2002]

mobile
business

FFFF FFFF

System Reserved
(kernel mode space)

8000 0000

Large Memory Area
(memory mapped files)

4200 0000

Reserved

0200 0000
0000 0000   Application Space

[Hall2002]

41

mobile business

Address	Region	
FFFF FFFF		
	System Reserved (kernel mode space)	
8000 0000		
	Large Memory Area (memory mapped files)	
4200 0000	Slot 32	Process 31
4000 0000	Slot 31	Process 30
3E00 0000		
0800 0000	Slot 3	Process 2
0600 0000	Slot 2	Process 1
0400 0000	Slot 1	Kernel (NK.exe)
0200 0000	Slot 0	Act. Process
0000 0000		
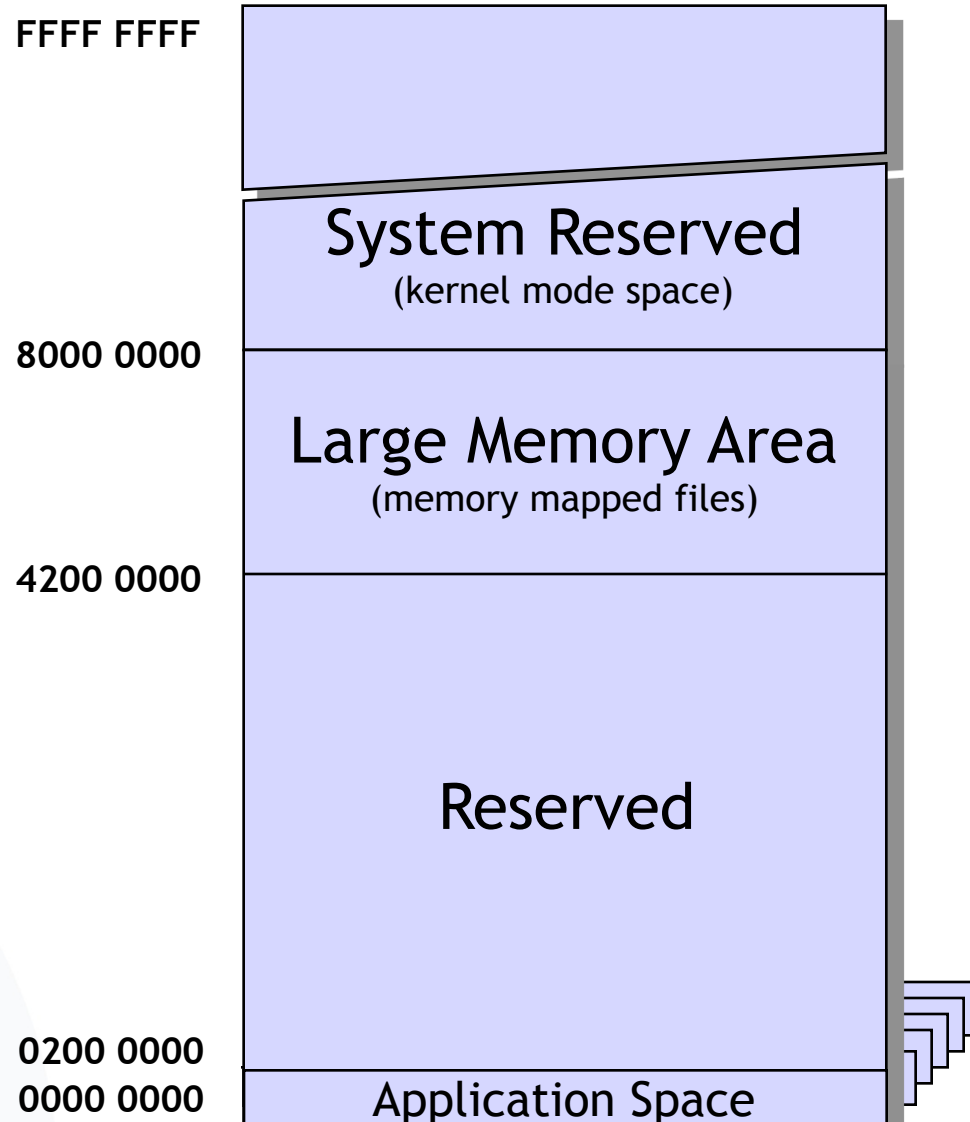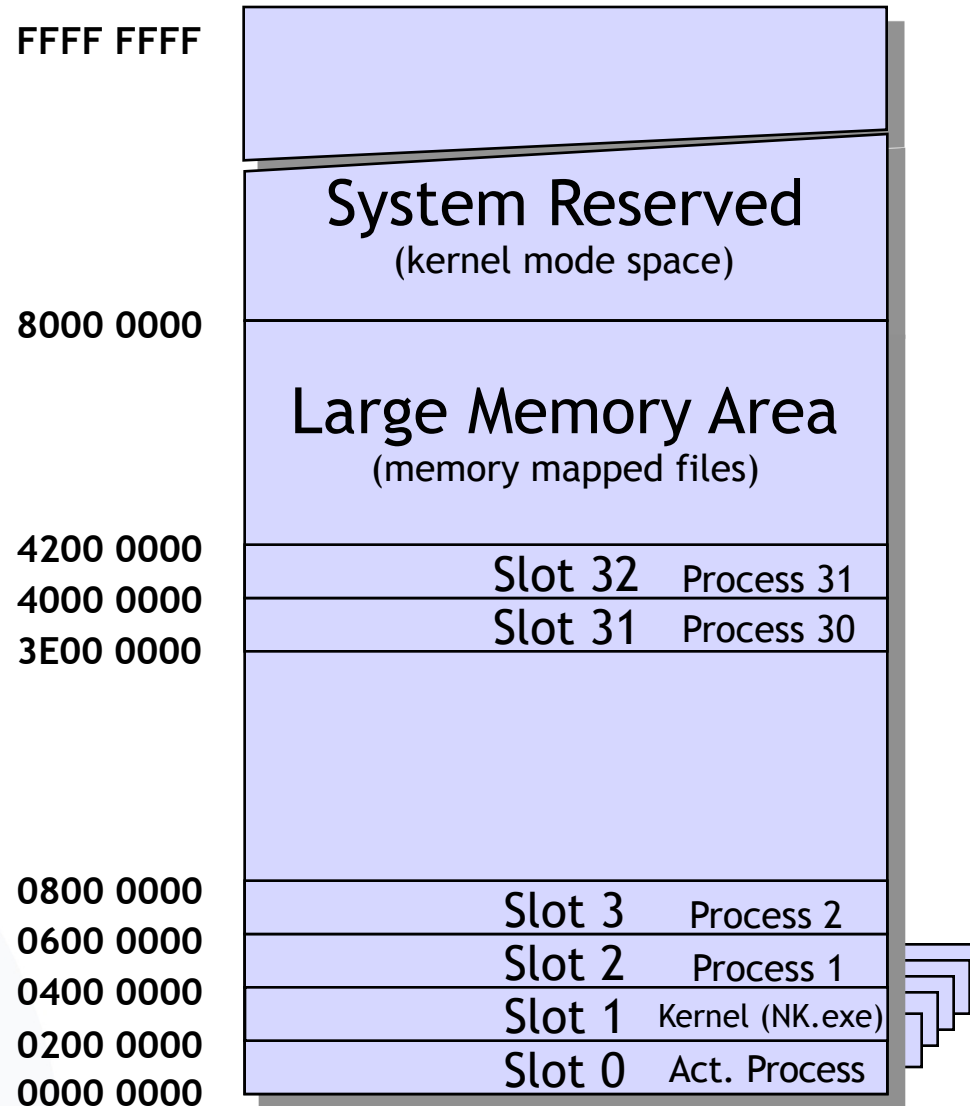
Detailed View

[Hall2002]

42

- Memory (RAM) is divided into 33 slots.

- One process per slot
  - Slot 2 to Slot 32
  - A process only has access to his own slot
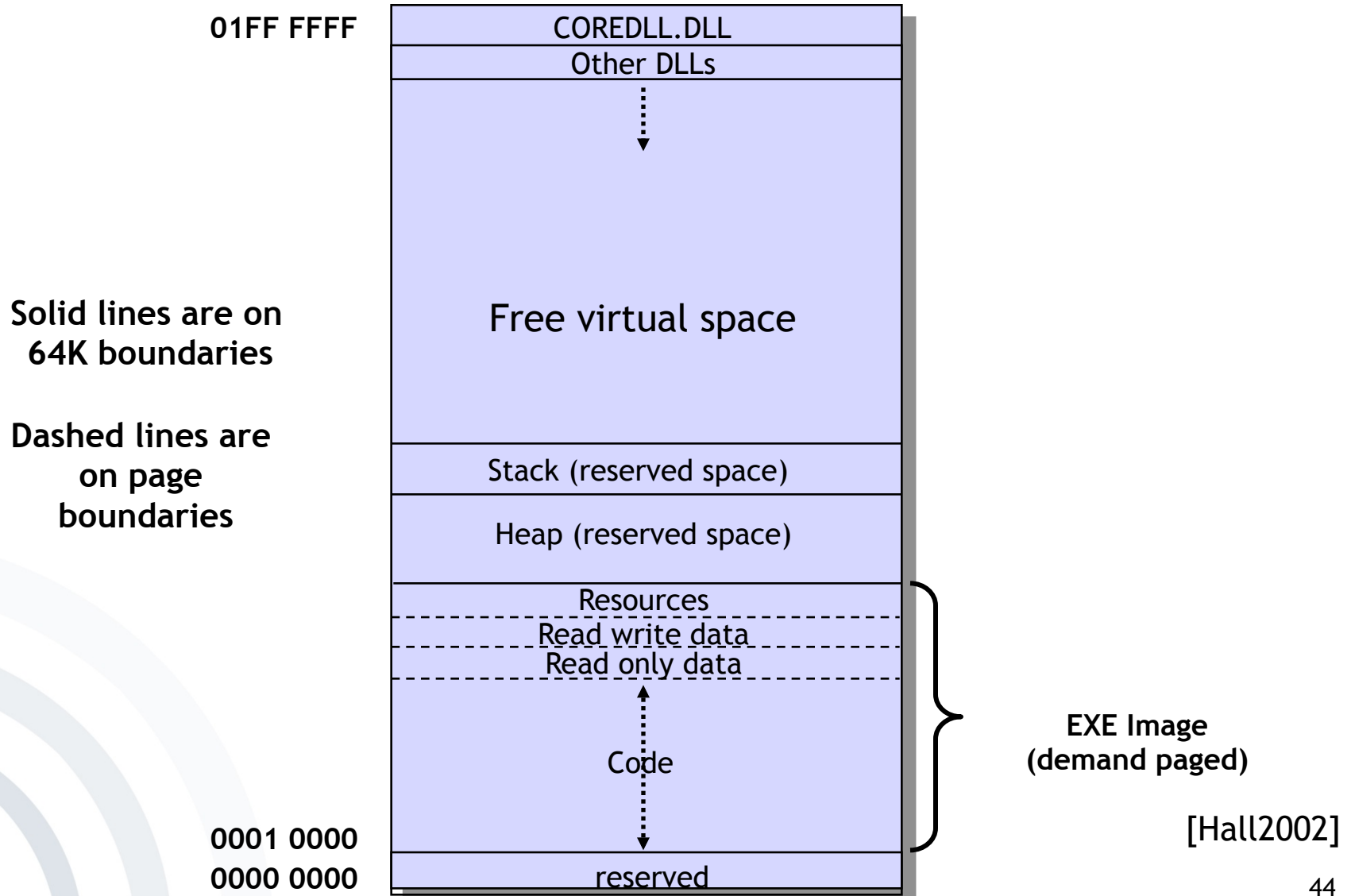   - ... and to slot 0, when it is active.

- Active process is placed into Slot 0.

- Kernel (NK.exe) is placed into Slot 1.

- Remaining memory is shared.

mobile business

01FF FFFF

COREDLL.DLL
Other DLLs
Free virtual space
Stack (reserved space)
Heap (reserved space)
Resources
Read write data
Read only data
Code
reserved

Solid lines are on 64K boundaries

Dashed lines are on page boundaries

EXE Image (demand paged)

0001 0000
0000 0000

[Hall2002]

44

- ## Maximum of 32 MB for virtual memory
  - ### Virtual memory is used for the code and the data

- ## Memory is:
  - ### Allocated on the basis of pages
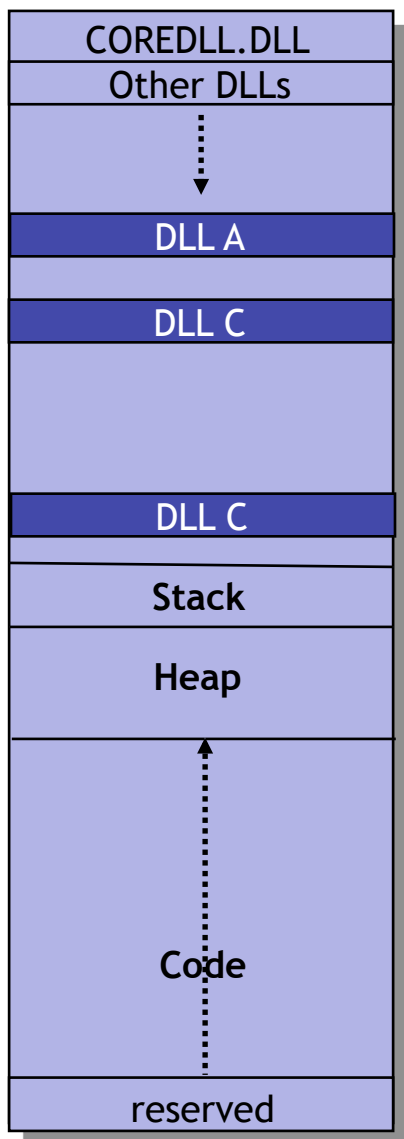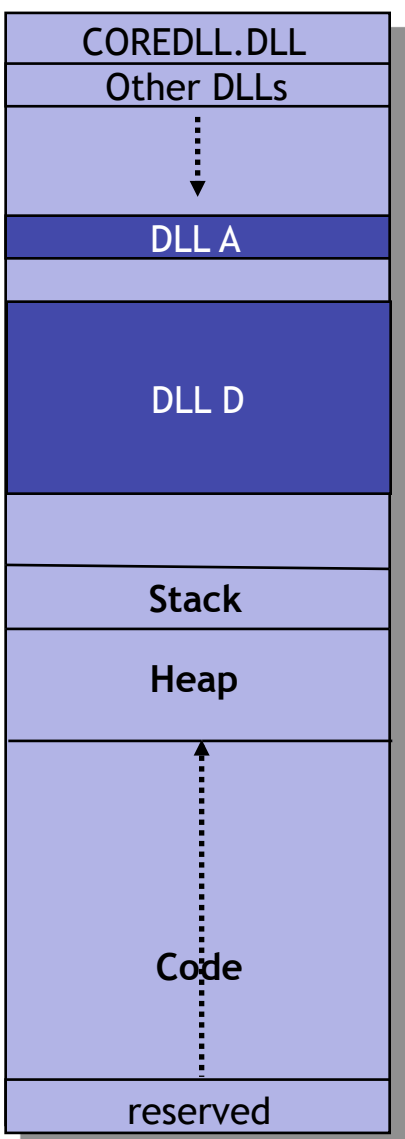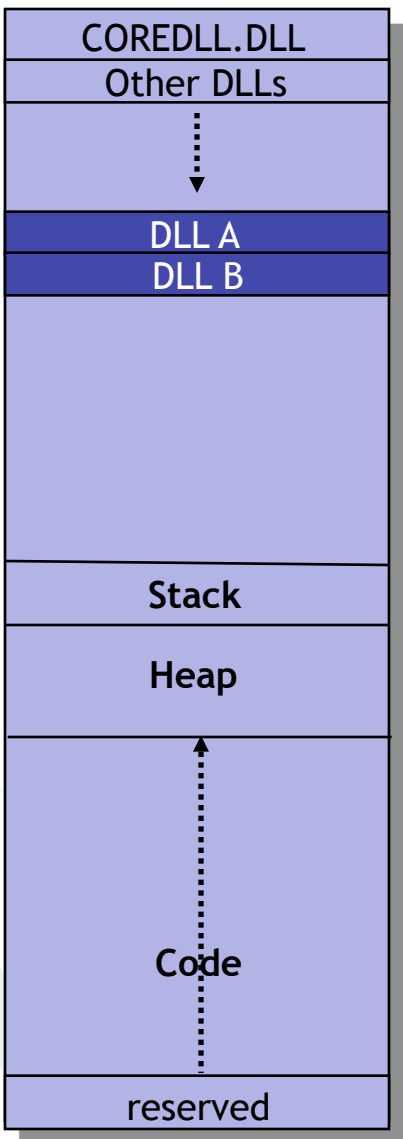 - ### Reserved in blocks of 64 KB

- Software library, containing a collection of functions and sub-programs that can be used by other independent programs.

- This methodology offers the following advantages:

  - Reutilisation of existing code

  - Distribution of the development process

  - Etc.

mobile
business

01FF FFFF

| COREDLL.DLL |
| Other DLLs |
| |
| DLL A |
| DLL B |
| |
| |
| Stack |
| Heap |
| |
| Code |
| reserved |

| COREDLL.DLL |
| Other DLLs |
| |
| DLL A |
| |
| DLL D |
| |
| Stack |
| Heap |
| |
| Code |
| reserved |

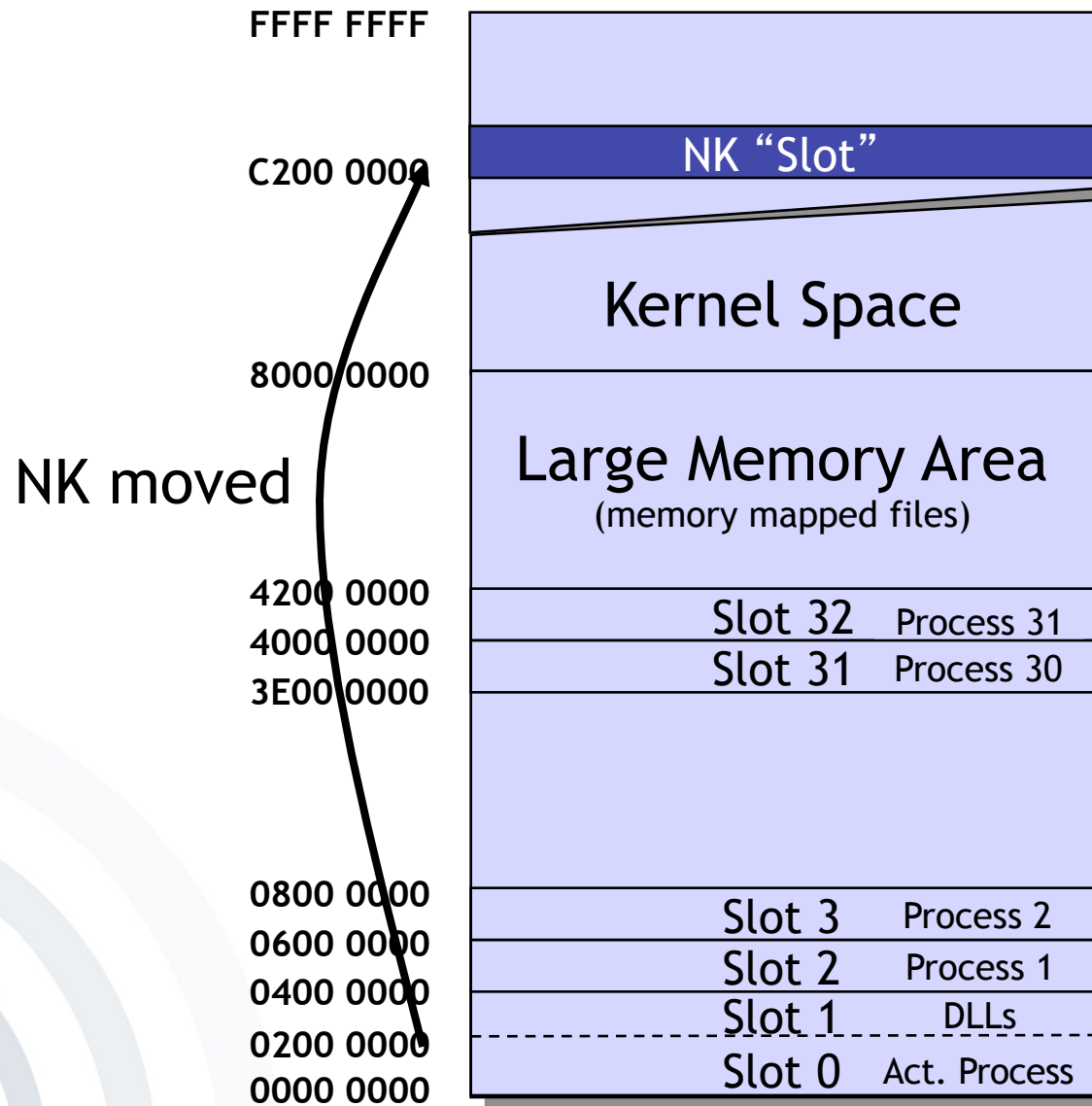| COREDLL.DLL |
| Other DLLs |
| |
| DLL A |
| |
| DLL C |
| |
| DLL C |
| Stack |
| Heap |
| |
| Code |
| reserved |

0001 0000
0000 0000

47

[Hall02]

- Any DLL being loaded by any process allocates memory of other processes, regardless if the DLL is used by other processes or not.

- The address, the DLL is loaded to, is dependent on the other DLLs being loaded by other processes.

- All DLLs are loaded/stored into memory blocks of 64K.

⮑ The more DLLs are loaded, the bigger is the problem.

- Windows CE .NET solves the DLL load problem by modifying the memory map.

- The kernel (NK.EXE) is relocated from Slot 1 into the kernel space starting from address 0xC200 0000.

- Slot 1 is used for the DLLs:

  - Is connected with all applications for Slot 0

Address	Region	
FFFF FFFF		
C200 0000	NK "Slot"	
	Kernel Space	
8000 0000		
	Large Memory Area (memory mapped files)	
4200 0000	Slot 32	Process 31
4000 0000	Slot 31	Process 30
3E00 0000		
0800 0000	Slot 3	Process 2
0600 0000	Slot 2	Process 1
0400 0000	Slot 1	DLLs
0200 0000	Slot 0	Act. Process
0000 0000		

NK moved

[Hall2002]

50

mobile
business

03FF FFFF	COREDLL.DLL	DLL Space (Constant for all applications)
	Other XIP DLLs	
	XIP DLL space	
01FF FFFF		
	RAM based DLLs	Application Slot (Cloned into slot 0)
	Free virtual space	
	Stack (reserved space)	
	Heap (reserved space)	
	Resources	
	Read write data	
	Read only data	
	Code	
0001 0000		
0000 0000	reserved	

[Hall2002]

51

- Windows CE .NET Application Memory Map

  - Application memory is now extended to 64 MB (from 0000 0000 up to 03FF FFFF).

  - DLLs are loaded into the upper 32 MB (from 0200 0000 up to 03FF FFFF).

  - Executable (EXE) code, heaps and stacks are using the lower 32 MB (from 0000 0000 up to 01FF FFFF).

  - There is no possibility for loaded applications to allocate memory above 32 MB.

- Functions
- Processes
    - States and elements
    - Scheduling
    - Inter-Process-Communication (IPC)
- Memory Management
    - Mapping
    - Paging
    - Segmentation
    - Examples
- Security & Maintenance

- Security mechanism provided by all mobile operating systems

  - Separation of running programs

  - Memory Management allocates well-defined memory areas for every sandboxed application at runtime.

  - Protection of device's resources from mobile applications in the sandbox

  - Untested (program) code cannot cause damage from within the sandbox.

- Examples

  - Network-access restrictions

  - Restricted file system access

Mobile Device Management

- Software to secure, monitor, manage and support mobile devices

- Over-the-air distribution of
    - Applications
    - Data
    - Configuration settings

➲ Higher security level, lower cost and fewer downtimes

Bibliography

- [Burkhardt2001] Burckhardt J. et al.: Pervasive Computing, München, 2001

- [Hall2002] Hall, Mike: Windows CE .NET Advanced Memory Management. Microsoft Embedded Crash Course for Faculty and PhD's, 2002

- [Microsoft2006] http://research.microsoft.com/collaboration/university/europe/events/dotnetcc/Version3/DVD/, accessed 2006-10-20

- [SilberGalvin1999] Silberschatz; Galvin: Operating System Concepts, 5th Edition, John Wiley & Sons, Inc, 1999

- [SilberGalvinGagne2005] Silberschatz; Galvin; Gagne: Operating System Concepts, 7th Edition, John Wiley & Sons, Inc. 2005

- [Stallings2003] Stallings, W.: Betriebssysteme – Prinzipien und Umsetzung, 4th Edition, Pearson Studium, München, 2003.

- [Zobel2001] Zobel, J.: Mobile Business und M-Commerce, München, 2001