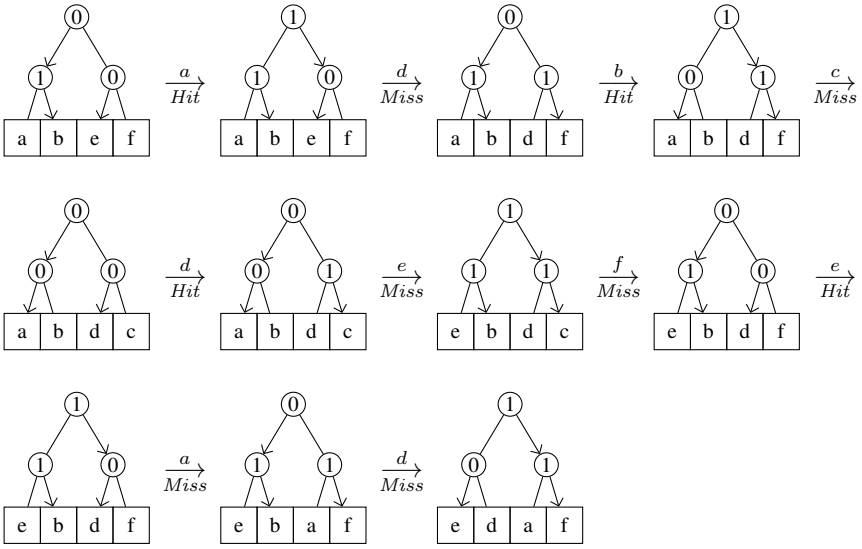
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

Systemarchitektur SS 2021

Lösungsskizze 11

Aufgabe 11.1: Pseudo-LRU



Aufgabe 11.2: Ersetzungsstrategien im Vergleich

Folgende Zugriffssequenzen verursachen bei erstgenannter Ersetzungsstrategie weniger Misses als bei zweitgenannter.

(a) *LRU|FIFO*: abcdaea

(b) *FIFO|LRU*: abcdaeb

(c) *PLRU|LRU*: abcdcea

(d) *PLRU|FIFO*: abcdaea

Aufgabe 11.3: Betriebssysteme

- **Virtualisierung**: Bestimmte physikalische Ressourcen werden dem Programm nicht direkt bereitgestellt, sondern vom Betriebssystem so verwaltet, dass Programme in einer abstrahierten Form darauf zugreifen. Beispielsweise:

  Speichervirtualisierung: Der Arbeitsspeicher wird soweit abstrahiert, dass ein Programm unter anderem nur die Bereiche des Speichers 'sehen' kann, die es vorher selber alloziert hat, und nicht die eines anderen Programms oder die Bereiche des Grafikspeichers. Dies wird erreicht, indem das Betriebssystem 'virtuelle' Speicheradressen des Programms vor Speicherzugriffen in echte Adressen übersetzt. Außerdem wird dem Programm dadurch die Illusion vermittelt, dass es alleine auf der Maschine ist und den gesamten Adressraum (d.h. den virtuellen Adressraum) sehen könnte.

  Prozessorvirtualisierung: Der Prozessor wird soweit abstrahiert, dass es für ein Programm so aussieht, dass es eine eigene, ihm zugewiesene (virtuelle) CPU hat, obwohl die physikalische CPU nebenläufig auch andere Programme oder Aufgaben berechnet. Mehrere Programme sollen 'gleichzeitig' ablaufen, indem der Prozessor abwechselnd den einzelnen Aufgaben Rechenzeit widmet. Dafür muss unter anderem sichergestellt werden, dass sich die verschiedenen Programme nicht gegenseitig stören und sie nichts von dem Programmwechsel mitbekommen.

- **Nebenläufigkeit**: Damit unterschiedliche Prozesse auch miteinander kommunizieren können, benötigen sie geteilte Ressourcen, zum Beispiel einen gemeinsamen Speicherbereich. Damit es beim Verwenden eines gemeinsamen Speichers nicht zu unerwünschten Effekten durch kritische Wettläufe (z.B. gleichzeitiges Lesen und Schreiben) kommt, benötigt man eine gewisse Synchronisation. Ein Beispiel sind sogenannte Locks mit denen verhindert werden kann, dass mehrere Prozesse gleichzeitig auf eine Variable zugreifen. Das Betriebssystem kann helfen eine effiziente und kontrollierte Interaktion mit geteilten Ressourcen zu erlauben.

- **Persistenz**: Daten sollen nicht nur im Arbeitsspeicher abgelegt werden, sondern auch auf beispielsweise einer Festplatte, damit sie auch nach einem erneuten Start des Systems noch zur Verfügung stehen. Dafür verwaltet das Betriebssystem ein *Dateisystem*, dass die Ansteuerung der Massenspeicher (Festplatten, SSDs,...) abstrahiert und es Programmen durch einfache Systemaufrufe ermöglicht, auf Dateien zuzugreifen.

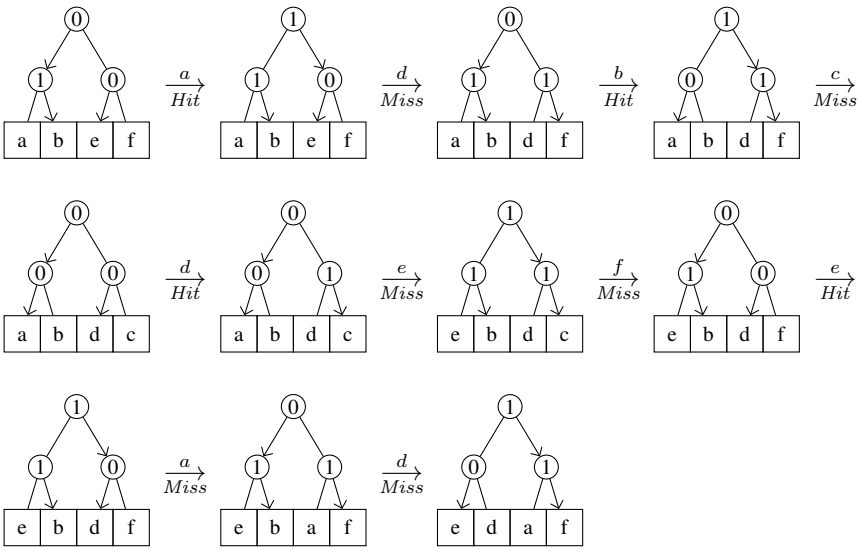
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Jan Reineke
Dr. Andreas Abel
Lucas Biehl

System Architecture SS 2021

Solution Sketch 11

Problem 11.1: Pseudo-LRU



Problem 11.2: Comparison of Replacement Policies

The following access sequences cause fewer misses with the first replacement policy than with the second.

(a) *LRU*|*FIFO*: abcdaea

(b) *FIFO*|*LRU*: abcdaeb

(c) *PLRU*|*LRU*: abcdcea

(d) *PLRU*|*FIFO*: abcdaea

Problem 11.3: Operating Systems

- **Virtualization**: Certain physical resources are not provided directly to the program, but are managed by the operating system so that programs access them in an abstracted form. For example:

  Memory virtualization: Memory is abstracted to the extent that, among other things, a program can 'see' only the areas of memory that it has previously allocated itself, and not those of another program or the areas of graphics memory. This is achieved by the operating system translating 'virtual' memory addresses of the program into physical addresses before memory accesses. It also gives the program the illusion that it is alone on the machine and could see the entire (virtual) address space.

  Processor virtualization: The processor is abstracted to the point where it appears to a program that it has its own assigned (virtual) CPU, even though the physical CPU is concurrently executing other programs or tasks. The processor runs several programs 'simultaneously' by providing computing time to each task in turn. For this, it must be ensured among other things that the different programs do not interfere with each other, and that they do not notice the task switch.

- **Concurrency**: In order for different processes to be able to communicate with each other, they need shared resources, for example, a shared memory area. To avoid undesired effects caused by critical race conditions (e.g. simultaneous reading and writing) when using a shared memory, we need a certain synchronization. An example are so-called locks, which can prevent that several processes access a variable at the same time. The operating system can help to allow for an efficient and controlled interaction with shared resources.

- **Persistence**: Data should not only be stored in the main memory, but also on, e.g., a hard disk, so that they are still available after a reboot of the system. For this purpose, the operating system manages a *file system* that abstracts the control of the mass storage (hard disks, SSDs,...) and enables programs to access files by basic system calls.