SAARLAND UNIVERSITY
COMPUTER SCIENCE

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

# Systemarchitektur SS 2021

# Aufgabenblatt 12

Sie können Ihre Lösungen bis **Mittwoch, dem 14.07.2021, um 10:00 Uhr** im CMS abgeben.
Geben Sie auf Ihrer Lösung Ihr Tutorium sowie die Namen und Matrikelnummern aller Gruppenmitglieder an.

## Aufgabe 12.1: Scheduling

Wir betrachten die in der Vorlesung vorgestellten Planungsverfahren: FIFO/FCFS, SJF, STCF und Round Robin.

(a) Beschreiben Sie die Verfahren jeweils kurz und nennen Sie jeweils einen Vor- und einen Nachteil.

(b) Gegeben die folgenden Jobs:

| Job | Ankunftszeit | Ausführungszeit |
|---|---|---|
| A | 0 | 6 |
| B | 2 | 3 |
| C | 5 | 3 |
| D | 0 | 7 |

Planen Sie jeweils mit den obigen Verfahren die Ausführung dieser Jobs.

(c) Berechnen Sie je die durchschnittliche Umlauf- und Antwortzeit.

*Hinweis: (FIFO) Treffen zwei Jobs zur gleichen Zeit ein, betrachten wir den lexikographisch kleineren Job zuerst. (SJF, STCF) Haben zwei Jobs die gleich verbleibende Ausführungszeit, so dürfen Sie wählen.*

## Aufgabe 12.2: Scheduling (2)

(a) In der Vorlesung wurde Shortest Job First (SJF) als optimal bezüglich durchschnittlicher Umlaufzeit eingeführt, sofern alle Jobs gleichzeitig ankommen. Beweisen Sie diese Aussage formal mittels Widerspruch.

(b) Kommen nicht alle Jobs gleichzeitig an, ist SJF nicht optimal. Zeigen Sie auch dies.

## Aufgabe 12.3: Multi-Level Feedback Queue (MLFQ)

(a) In Foliensatz 16 finden Sie auf Seite 32 die fünf Regeln, nach denen eine MLFQ arbeitet. Erläutern Sie kurz in eigenen Worten, welchen Effekt jede dieser Regeln hat sowie die Motivation/Intention dahinter.

(b) Wie müssen Sie die Parameter der MLFQ wählen, sodass diese sich wie ein Round Robin-Scheduler verhält?

## Aufgabe 12.4: Segmentierung

Ein Segment $i$ ist beschrieben durch eine Basisadresse $B_i$ und ein Schranke $S_i$. Es erstreckt sich über den Adressbereich von $B_i, B_i + 1, \ldots, B_i + S_i - 1$. Der Stack eines Prozesses wächst jedoch kleineren Adressen entgegen. Ausgehend von der Stack-Basisadresse $B_s$ erstreckt sich der Stack bis zu einer Schranke $S_s$ also auf Adressen $B_s, B_s - 1, B_s - 2, \ldots, B_s - S_s + 1$. Wie lässt sich diese Stackorganisation mit Segmentierung verbinden? Wie würden Sie ggfls. die Segmenttabelle und die zugehörige MMU erweitern?

## Aufgabe 12.5: Speichervirtualisierung

Wir haben in der Vorlesung mehrere Verfahren zur Virtualisierung von Speicher kennen gelernt: Zeitmultiplexen, statische Relokation, dynamische Relokation, Segmentierung, Paging mit/ohne TLBs (mit und ohne ASID). Wir betrachten ein System mit 14-Bit virtuellen Adressen und 16-Bit physischen Adressen. Der Prozess-Adressraum besteht aus einem Code-Teil und einem Daten-Teil. Wir möchten zwei Prozesse des folgenden Programms ausführen:

```
0x0100: li $1, 0x1230
0x0104: lw $2, 0x4($1)
0x0108: addiu $2, $2, 5
0x010c: sw $2, 0x4($1)
```

Zunächst wird Prozess 1 ausgeführt. Prozess 1 wird aber nach Ausführung des Lade-Befehls durch Prozess 2 unterbrochen. Nachdem Prozess 2 seine Ausführung beendet hat, wird Prozess 1 fortgeführt.

(a) Beschreiben Sie für jedes der obigen Verfahren den jeweiligen Aufbau der Prozesskontrollblöcke sowie den Ablauf eines Prozesswechsels, d.h. was das Betriebssystem für einen Prozesswechsel tun muss.

(b) Betrachten Sie Segmentierung. Geben Sie die Sequenz aller physischen Speicherzugriffen der Prozesse an, d.h. unter Beachtung der zeitlichen Abläufe. Das Codesegment von Prozess 1 startet bei 0, das Codesegment von Prozess 2 startet bei 0x2000. Das Datensegment von Prozess 1 startet bei 0x3000, das von Prozess 2 bei 0x1000. Die Schranken sind jeweils 0x1000.

(c) Betrachten Sie Paging ohne TLB. Geben Sie die Sequenz aller physischen Speicherzugriffen der Prozesse an, d.h. unter Beachtung der zeitlichen Abläufe. Als Seitengröße wählen wir 4 kB, ein Seitentabelleneintrag ist 4 Byte groß. Die Seitentabelle von Prozess 1 enthält von Adresse 0x0000 aufsteigend die Werte 0x9, 0x7, 0x8, 0x6. Die Seitentabelle von Prozess 2 enthält von Adresse 0x0800 aufsteigend die Werte 0x3, 0x4, 0x2, 0x1.

(d) Wie ändert sich die Sequenz durch den Einsatz eines TLBs ohne ASID, bzw. eines TLBs mit ASID?

SAARLAND UNIVERSITY
COMPUTER SCIENCE

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

# System Architecture SS 2021

# Assignment 12

You may submit your solutions via the CMS until **10:00 a.m. on Wednesday, July 14, 2021**.
Please state on your solutions your tutorial, and the names and matriculation numbers of all team members.

## Problem 12.1: Scheduling

We consider the scheduling algorithms discussed in the lecture: FIFO/FCFS, SJF, STCF, and Round Robin.

(a) Briefly describe each of the algorithms and name one advantage and one disadvantage of each.

(b) Given the following jobs:

| Job | Arrival time | Run-time |
|---|---|---|
| A | 0 | 6 |
| B | 2 | 3 |
| C | 5 | 3 |
| D | 0 | 7 |

Schedule the execution of these jobs with each of the algorithms.

(c) Compute the average turnaround and response times for each.

*Note: (FIFO) If two jobs arrive at the same time, we consider the lexicographically smaller job first. (SJF, STCF) If two jobs have the same remaining execution time, you may choose.*

## Problem 12.2: Scheduling (2)

(a) In the lecture, Shortest Job First (SJF) was introduced as optimal with respect to the average turnaround time, provided that all jobs arrive at the same time. Prove this statement formally by contradiction.

(b) If not all jobs arrive at the same time, SJF is not optimal. Also prove this.

## Problem 12.3: Multi-Level Feedback Queue (MLFQ)

(a) In slide deck 19, you can find on page 33 the five rules that MLFQs are based on. Briefly explain in your own words the effect of each of these rules and the motivation/intention behind them.

(b) How would you choose the parameters of the MLFQ so that it behaves like a Round Robin scheduler?

## Problem 12.4: Segmentation

A segment $i$ is described by a base address $B_i$ and a bound $S_i$. It spans the address range of $B_i, B_i + 1, \ldots, B_i + S_i - 1$. However, the stack of a process grows toward smaller addresses. Starting from the stack base address $B_s$, the stack thus extends up to a bound $S_s$, i.e., to the addresses $B_s, B_s - 1, B_s - 2, \ldots, B_s - S_s + 1$. How can this stack organization be combined with segmentation? How would you extend the segment table and the corresponding MMU, if necessary?

## Problem 12.5: Memory Virtualization

We have seen several methods for virtualizing memory in the lecture: time division multiplexing, static relocation, dynamic relocation, segmentation, paging with/without TLBs (with and without ASID). We consider a system with 14-bit virtual addresses and 16-bit physical addresses. The process address space consists of a code part and a data part. We would like to execute two processes of the following program:

```
0x0100: li $1, 0x1230
0x0104: lw $2, 0x4($1)
0x0108: addiu $2, $2, 5
0x010c: sw $2, 0x4($1)
```

First, process 1 is executed. However, process 1 is interrupted by process 2 after executing the load instruction. After process 2 finishes its execution, process 1 continues.

(a) For each of the above methods, describe the respective structure of the process control blocks as well as the workflow of a process switch, i.e., what the operating system has to do for a process switch.

(b) Consider segmentation. Specify the sequence of all physical memory accesses of the processes, i.e., considering the timing. The code segment of process 1 starts at 0, the code segment of process 2 starts at 0x2000. The data segment of process 1 starts at 0x3000, that of process 2 at 0x1000. The bounds are 0x1000 in each case.

(c) Consider paging without a TLB. Specify the sequence of all physical memory accesses of the processes, i.e., considering the timing. We choose 4 kB as the page size, and a page table entry is 4 bytes in size. The page table of process 1 contains from address 0x0000 ascending the values 0x9, 0x7, 0x8, 0x6. The page table of process 2 contains from address 0x0800 ascending the values 0x3, 0x4, 0x2, 0x1.

(d) How does the sequence change when using a TLB without ASID, or a TLB with ASID, respectively?