

Aufgabe 1: Speicherverwaltung

(14 Punkte)

Es wird das Innenleben einer *Memory Management Unit* (MMU) mit Seitenadressierung betrachtet, wobei eine Seitentabelle mit 16 Seiten verwendet wird, virtuelle Adressen sind 16 Bit, die physikalischen Adressen sind 15 Bit lang und die Wortlänge sei 1 Byte¹.

- a) Wie groß ist eine Seite?
- b) (optional) Woraus besteht ein typischer Seitentableneintrag und welche Bedeutung hat insbesondere das *Present/Absent*-Bit? Bei höchstens wievielen Einträgen der Seitentabelle kann im gegebenen Fall das *Present/Absent*-Bit auf 1 („Present“) gesetzt sein?

- c) Gegeben sei ein Ausschnitt der Seitentabelle zu einem Zeitpunkt (vgl. Abbildung rechts). Welche physikalische Adresse ergibt sich für folgende virtuelle Adressen?

- i) 0x5fe8
ii) 0xfcee
iii) 0xa470
iv) 0x0101

	Seiten- rahmen- nr.	Present/ Absent- Bit
15	000	0
14	000	0
13
12
11
10	000	1
9	000	0
8	110	1
7
6
5	001	1
4
3
2	111	1
1	010	1
0	101	1

- d) Um eine möglichst optimale Seitengröße zu wählen, muss man zwischen mehreren zum Teil gegenläufigen Faktoren abwägen. Bitte nennen Sie mindestens 2 Faktoren, die für eher kleine Seiten sprechen, und mindestens einen Grund, welcher für eher große Seiten spricht.
- e) Sei $p = 4$ MiB die durchschnittliche Prozessgröße im Speicher und $L = 8$ B die Länge eines Seitentableneintrags. Erläutern Sie kurz den Zusammenhang zwischen der Seitentabellengröße im Speicher und der internen Fragmentierung bei verschiedenen Seitengrößen und ermitteln Sie die optimale Seitengröße s in KiB unter der Annahme, dass im Durchschnitt die Hälfte der letzten Seite eines Prozesses leer bleibt. Geben Sie dabei den vollständigen Rechenweg an und erklären Sie Ihre Annahmen.

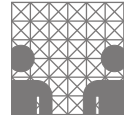
Aufgabe 2: Seitenersetzungsalgorithmen

(14 Punkte)

- a) Gegeben sei ein Seitenspeicher der Größe 3. Zum Zeitpunkt $t = 0$ sei der Speicher leer. Illustrieren Sie anhand von Tabelle 1 den Zustand des Seitenspeichers bei Verwendung des (i) NRU-Algorithmus (*Not Recently Used*) und (ii) *Second-Chance-Algorithmus* zu den Zeitpunkten 1 bis 12 für die gegebene Referenzkette (siehe Zeile *Angeforderte Seite*) unter Berücksichtigung der jeweiligen Zugriffsart (Lesen (r) bzw. Schreiben (w)). Geben Sie für jeden Zeitpunkt an, ob ein Seitenalarm (*page fault*) aufgetreten ist (j/n).

Bitte verwenden Sie als Grundlage das Buch *Moderne Betriebssysteme* von Andrew S. Tanenbaum. Das Löschen der R-Bits beim NRU-Algorithmus soll jeweils **nach** $t = 5$ und $t = 10$ erfolgen (Timerinterrupt).

¹Hilfe zur Beantwortung dieser Fragen finden Sie in weiterführender Literatur, bspw. in Andrew S. Tanenbaum, *Moderne Betriebssysteme*, 3. Auflage, Abschnitte 3.3.1ff (*Paging*) und 3.5.3ff (*Seitengröße*)



t	1	2	3	4	5	6	7	8	9	10	11	12
Angeforderte Seite	1	2	3	4	2	1	2	5	6	2	6	3
Zugriffsart	r	w	w	r	r	r	r	w	r	w	w	r
Seitenalarm	j
Seiten im Speicher	1

Tabelle 1: Zustand des Seitenspeichers für $t = 1$ bis $t = 15$.

- b) (optional) Der optimale Seitenersetzungsalgorithmus wird gerne für theoretische Betrachtungen herangezogen. Warum kommt er nicht in realen Betriebssystemen zum Einsatz (max. 3 Sätze)?
- c) (optional) Ergibt „LRU“ als Seitenersetzungstrategie Sinn, sofern das „L“ in „LRU“ für „Last“ und nicht, wie korrekterweise, für „Least“ stehen würde ?

Aufgabe 3: Synchronisation

(12 Punkte)

Der Gebrauch von Semaphoren kann anhand der Lösung des klassischen Reader-Writer-Problems anschaulich demonstriert werden. Bei dem Reader-Writer-Problem gibt es zwei verschiedene Klassen, die beide auf einen gemeinsamen Datenbereich zugreifen. Instanzen der Klasse `Writer` schreiben in ihrer Hauptmethode (`processWriter()`) in den gemeinsamen Datenbereich und benötigen daher den exklusiven Zugriff. Instanzen der Klasse `Reader` hingegen greifen in ihrer Hauptmethode (`processReader()`) nur lesend auf den Datenbereich zu und können daher auch parallel auf dem gemeinsamen Datenbereich agieren.

- a) Entwerfen Sie eine Lösung für das Reader-Writer-Problem, indem Sie die Hauptmethoden (`processWriter()` und `processReader()`) der beiden Klassen in Pseudocode angeben. Der Zugriff auf den gemeinsamen Datenbereich erfolgt durch die Methoden `readData()` (im Falle eines Readers) und durch die Methode `writeData()` (im Falle eines Writers). Verwenden Sie in Ihrer Lösung folgende Hilfsvariablen und geben Sie deren Initialisierungswerte mit an:
- ein binäres Semaphor `W`, um den kritischen Abschnitt zu schützen, in dem die eigentlichen Datenzugriffe (durch `writeData()` bzw. `readData()`) erfolgen
 - eine Zähler-Variable `NumberOfActiveReaders`, die die Anzahl der aktiven Reader angibt, die gerade auf den gemeinsamen Datenbereich zugreifen
 - ein binäres Semaphor `Mutex`, welches dafür sorgt, dass die Zähler-Variable `NumberOfActiveReaders` nur durch einen Reader zur Zeit modifiziert werden kann
- b) Werden bei Ihrer Lösung bestimmte Prozesse benachteiligt? Wenn ja, deuten Sie bitte kurz (max. 5 Sätze) einen möglichen Lösungsweg an.