SAARLAND UNIVERSITY
COMPUTER SCIENCE

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

Systemarchitektur SS 2021

Präsenzblatt 10 (Lösungsvorschläge)

Hinweis: **Dieses Aufgabenblatt wurde von Tutoren erstellt. Die Aufgaben sind für die Klausur weder relevant noch irrelevant, die Lösungsvorschlage weder korrekt noch inkorrekt.**

Aufgabe 10.1: Ersetzungsstrategien im Vergleich

Gegeben ein leerer, 4-fach vollassoziativer Cache. Geben Sie jeweils eine Zugriffssequenz an, so dass

1. LRU weniger Misses verursacht als FIFO
2. FIFO weniger Misses verursacht als LRU
3. PLRU weniger Misses verursacht als LRU
4. PLRU weniger Misses verursacht als FIFO.

Markieren Sie jeweils welche Zugriffe Misses verursachen.

Lösungsvorschlag:

1. abcdaea
2. abcdaeb
3. abcdcea
4. abcdaea

Aufgabe 10.2: Multi-Level Feedback Queue (MLFQ)

1. In Foliensatz 19 finden Sie auf Seite 23 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.
2. Wie müssen Sie die Parameter der MLFQ wählen, sodass diese sich wie ein Round Robin-Scheduler verhält?

Lösungsvorschlag:

1. Regeln für eine MLFQ:

Regel 1: *Priorität*$(A) >$ *Priorität*$(B) \rightarrow$ *führe* A *aus.*

Ermöglicht es, dass Prozesse aufgrund ihrer Prioritäten unterschiedlich behandelt werden. Durch das Setzen der Prioritäten kann entschieden werden, welcher Prozess ausgeführt werden soll.

Regel 2: *Priorität*$(A) =$ *Priorität*$(B) \rightarrow$ *Round Robin zwischen* A *und* B.

Prozesse mit gleicher Priorität werden gleichermaßen bearbeitet, verhindert „Verhungern“ von Prozessen bei gleicher Priorität.

Regel 3: *Neue Jobs erhalten höchste Priorität.*

Neue Jobs werden zuerst ausgeführt, damit sie, falls sie nur wenig Zeit beanspruchen, schnell abgearbeitet werden können. Lange Jobs werden so unterbrochen, eine Art STCF wird umgesetzt. So wird eine kurze Antwortzeit garantiert.

Regel 4: *Senke die Priorität eines Jobs, wenn er sein Budget auf einem Prioritätslevel erschöpft hat.*

Falls ein Job zu lange braucht, muss er nach kurzer Zeit warten. Dies realisiert den Teil von STCF, der garantiert, dass nur kurze Jobs einen langen Prozess dauerhaft unterbrechen können. Der Scheduler „lernt“, ob es sich um einen kurzen Job handelt.

Regel 5: *Setze die Priorität eines Jobs nach S Zeiteinheiten zurück auf die höchste Stufe.*

Sorgt dafür, dass lange Jobs nicht bei niedriger Prioriät verhungern, wenn immer wieder neue, kürzere Jobs auftauchen.

2. Die MLFQ degeneriert zu einem Round Robin-Scheduler, wenn immer alle Jobs die gleiche Priorität haben. Dies lässt sich dadurch erreichen, dass immer nur genau eine Prioritätsebene in der MLFQ genutzt wird, beispielsweise indem festgelegt wird, dass es nur eine Priorität gibt, oder indem Prioritäten niemals abgesenkt oder direkt wieder erhöht werden, sobald sie abgesenkt werden.

Aufgabe 10.3: Scheduling

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

1. Beschreiben Sie die Verfahren jeweils kurz und nennen Sie jeweils einen Vor- und einen Nachteil.
2. Gegeben die folgenden Jobs:

Job	Ankunftszeit	Ausführungszeit
A	0	12
B	2	6
C	4	3
D	0	9

   Planen Sie jeweils mit den obigen Verfahren die Ausführung dieser Jobs.
3. 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.
4. In der Vorlesung wurde Shortest Job First (SJF) als optimal bezüglich durchschnittlicher Umlaufzeit eingeführt, sofern alle Jobs gleichzeitig ankommen. Beweisen Sie, dass SJF nicht optimal ist, falls **nicht** alle Jobs gleichzeitig ankommen.

Lösungsvorschlag:

(a) Übersicht der Scheduling Planungsverfahren:

	Beschreibung	Vorteile	Nachteile
FIFO/ FCFS	Prozesse kriegen Betriebsmittel nacheinander zugeteilt. Wenn zwei zeitgleich → lexikographisch.	einfach	keinerlei Berücksichtigung anderer Faktoren
SJF	Prozesse nach steigender Ausführungszeit geordnet. (bei Gleichheit egal)	optimal bzgl. ø–Umlaufzeit (wenn Jobs glztg. ankommen)	längere Prozesse müssen ggf. lange warten
STCF	Wähle Prozess mit niedrgst. Restlaufzeit. Wenn neuer Prozess eintritt ggf. laufenden unterbrechen	optm. bzgl. ø–Umlaufzeit	Verhungern (Starvation) sehr langer Jobs
Round Robin	Führe Prozesse abwechselnd für eine Zeitscheibe fester Länge aus	fair kein Verhungern gut bzgl. Antwortzeit	Prozesswechsel kosten Zeit schlecht bzgl. Umlaufzeit unflexible wg. fester Zeitscheibenlänge

(b) Die Ausführungen sind in Abbildung 2 dargestellt. Zu beachten ist, dass es sich dabei nur jeweils um eine mögliche Ausführung handelt, andere Ausführungen mit anderen Werten für Umlauf- und Antwortzeit sind möglich.

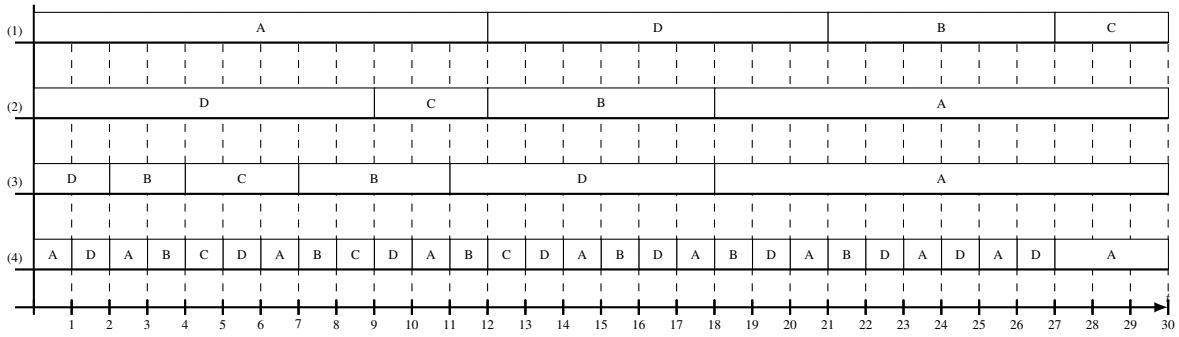

Abbildung 1: Ausführungen mit den Verschiedenen Planungsverfahren (1-4 entsprechen FIFO, SJF, STCF und Round Robin in dieser Reihenfolge)

(c) Übersicht über die sich ergebenden Umlauf– und Antwortzeiten:

Verfahren	ø–Umlaufzeit	ø–Antwortzeit
FIFO	$\frac{12+(27-2)+(30-4)+21)}{4} = 21$	$\frac{0+(21-2)+(27-4)+12}{4} = 13,5$
SJF	$\frac{30+(18-2)+(12-4)+9}{4} = 15.75$	$\frac{0+(12-2)+(9-4)+0}{4} = 3,75$
STCF	$\frac{30+(11-2)+(7-4)+18}{4} = 15$	$\frac{18+(2-2)+(4-4)+0}{4} = 4,5$
Round Robin (L1)	$\frac{30+(22-2)+(13-4)+27}{4} = 21.5$	$\frac{0+(3-2)+(4-4)+1}{4} = 0.5$

(d) Gegenbeispiel:

Job	Ankunftszeit	Ausführungszeit
A	0	10
B	1	2
C	2	3

In diesem Fall ergibt sich für die ø–Umlaufzeit mit SJF–Verfahren:

$$\frac{(10-0)+(12-1)+(15-2)}{3} = \frac{34}{3} \approx 11,33$$

Für das STCF–Verfahren ergibt sich jedoch:

$$\frac{(15-0)+(3-1)+(6-2)}{3} = \frac{21}{3} = 7$$

Demnach ist SJF hier nicht optimal. Dieser Effekt findet sich im Foliensatz zu Scheduling auf Folie 10 unter dem Begriff „Konvoieffekt“ und ist dadurch bedingt, dass die wesentlich kürzeren Jobs kurz nach dem deutlich längeren Job A ankommen, SJF jedoch im Gegensatz zu STCF Prozesse nicht unterbricht.

Aufgabe 10.4: System Calls and Stack

In der Vorlesung haben wir gesehen, dass man mit System Calls in den Kernel Modus wechseln kann und so auch auf Ein- und Ausgabegeräte benutzen kann. Register v0 wird dabei für die Interrupt Codes benutzt. Finden Sie (online) heraus, welche Codes für welche Events stehen.

- Schreiben Sie “Hello World!” in die Ausgabe und beenden Sie das Programm ordnungsgemäß.
- Lesen Sie eine Zahl ein, addieren Sie 42 und geben Sie das Ergebnis auf der Ausgabe aus.
- Lesen Sie Zahlen ein, bis 0 eingegeben wird und schreiben Sie diese auf den Stack.
- Geben Sie die Zahlen in reversierter Reihenfolge aus.

Lösungsvorschlag:

code	command	arguments
1	print int	a0
2	print float	f12
3	print double	f12
4	print string	a0 (adress)
5	read int	v0
6	read float	f0
7	read double	f0
8	read string	a0, a1
9	sbrk	a0, v0
10	exit	

```
.data
 msg: .asciiz "Hello World!"
.text
j ex3

ex1:
 li $v0 4
 la $a0 msg
 syscall
 li $v0 10
 syscall

ex2:
 li $v0 5
 syscall
 addiu $a0 $v0 42
 li $v0 1
 syscall
 li $v0 10
 syscall

ex3:
 li $v0 0
 store:
 #store
 sw $v0 0($sp)
 subi $sp $sp 4
 # read integer
 li $v0 5
 syscall
 bnez $v0 store

 li $v0 1
 printer:
 addiu $sp $sp 4
 lw $a0 0($sp)
```

```
beqz $a0 end
syscall
j printer

end:
li $v0 10
syscall
```

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

System Architecture SS 2021

Tutorial Sheet 10 (Suggested Solutions)

Note: **This task sheet was created by tutors. The tasks are neither relevant nor irrelevant for the exam, the suggested solutions are neither correct nor incorrect.**

Problem 10.1: Replacement policies in comparison

Assuming an initially empty 4 slot fully-associative cache, give an access sequence for each of the following situations:

1. LRU leads to less misses than FIFO
2. FIFO leads to less misses than LRU
3. PLRU leads to less misses than LRU
4. PLRU leads to less misses than FIFO.

Mark all misses.

Suggested solution:

1. abcdaea
2. abcdaeb
3. abcdcea
4. abcdaea

Problem 10.2: Multi-Level Feedback Queue (MLFQ)

1. In Slides 19 at page 23 you will find the five rules, that define how a MLFQ works. Please explain each rule in your own words and motivate it.
2. How should a MLFQ be configured so that it behaves like a Round Robin Scheduler?

Suggested solution:

1. Rules for a MLFQ:

Rule 1: $\mathit{priority}(A) > \mathit{priority}(B) \rightarrow \mathit{execute}\ A$.

Allows proccesses to be treated in repect to their priority. throu different prioritys the proccess to be executed can be defined.

Rule 2: $\mathit{priority}(A) = \mathit{priority}(B) \rightarrow \mathit{Round\ Robin\ between}\ A\ \mathit{and}\ B$.

proccess with the same priority are treated equaly. Prevents ßtarvingöf proccess with equal priority.

Rule 3: *new jobs get the highest priority.*

New Jobs are executed imidiatly. This reducess the latency.

Rule 4: *Reduce the priority of a proccess if it has used up its time buget on a priority level.*

If a job takes to long its priority is reduced. This lets short jobs interrupt longer jobs.

Rule 5: *Set the priority of a proccess to th maximum after a time S.*

Prevents ßtarvingöf long jobs.

2. If a MLFQ only has one priority level only rule 2 applies at we have Round Robin.

Problem 10.3: Scheduling

Lets look at the schedulers from the lecture: FIFO/FCFS, SJF, STCF and Round Robin.

1. Describe each scheduler and give and name a benefit and weakness for each.
2. Schedule the following Jobs with each scheduler:

Job	Arivall	duration
A	0	12
B	2	6
C	4	3
D	0	9

3. Calculate the average turnaround and response time.

4. We have seen that SJF is optimal in regards to average turnaround time if all jobs come in at the same time. Show that this is not the case if the jobs come in at different times.

Suggested solution:

(a) Overview of scheduling algorithms:

	description	advantages	disadvantages
FIFO/ FCFS	Jobs are scheduled in arrival order.	simple	ignoring other factors
SJF	shortest jobs are executed first	optimal in regards to average turnaround time if all jobs arrive at the same time	long processes may wait for a long time
STCF	execute process with least time left.	optimal in regards to average turnaround time	starvation of long jobs
Round Robin	execute processes alternatingly for a set duration	fair no starvation good latency	switching processes takes time bad in regards to average turnaround time unflexible

(b)

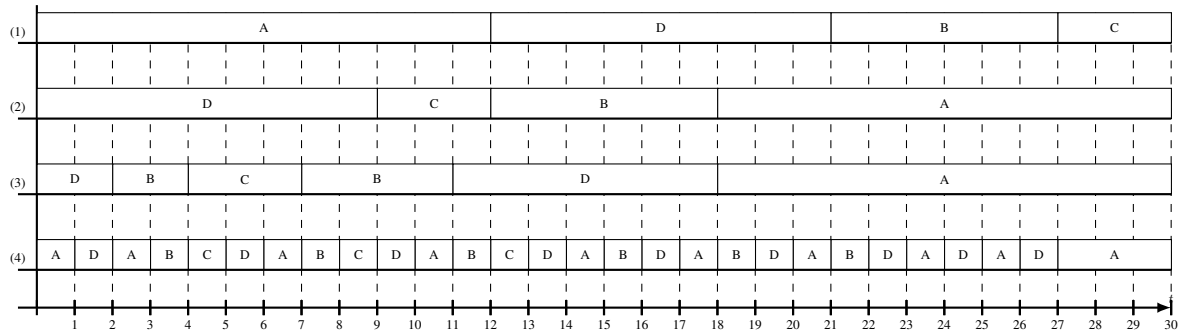

Abbildung 2: schedules with differing scheduling algorithms (1-4 entsprechen FIFO, SJF, STCF and Round Robin in this order)

(c)

scheduler	ø–turnaround time	ø–response time
FIFO	$\frac{12+(27-2)+(30-4)+21)}{4} = 21$	$\frac{0+(21-2)+(27-4)+12}{4} = 13,5$
SJF	$\frac{30+(18-2)+(12-4)+9}{4} = 15.75$	$\frac{0+(12-2)+(9-4)+0}{4} = 3,75$
STCF	$\frac{30+(11-2)+(7-4)+18}{4} = 15$	$\frac{18+(2-2)+(4-4)+0}{4} = 4,5$
Round Robin (L1)	$\frac{30+(22-2)+(13-4)+27}{4} = 21.5$	$\frac{0+(3-2)+(4-4)+1}{4} = 0.5$

(d) counter example:

Job	arrival time	processing time
A	0	10
B	1	2
C	2	3

in this case, the ø–turnaround time with SJF is:

$$\frac{(10-0)+(12-1)+(15-2)}{3} = \frac{34}{3} \approx 11,33$$

for STCF it is:

$$\frac{(15-0)+(3-1)+(6-2)}{3} = \frac{21}{3} = 7$$

SJF is not optimal here.

Problem 10.4: System Calls and Stack

In the lecture we learned that we can switch to kernel mode with syscalls and that we can use that to talk to IO devices. Register v0 is used for interrupt codes. Search online for the relevant codes and write the following programs

- write “hello world” and terminate the program.
- read a number, then add 42, output the number.
- read in numbers and store them onto the stack until 0 is read.
- output numbers in reversed order.

Suggested solution:

code	command	arguments
1	print int	a0
2	print float	f12
3	print double	f12
4	print string	a0 (adress)
5	read int	v0
6	read float	f0
7	read double	f0
8	read string	a0, a1
9	sbrk	a0, v0
10	exit	

```
.data
 msg: .asciiz "Hello World!"
.text
j ex3

ex1:
 li $v0 4
 la $a0 msg
 syscall
 li $v0 10
 syscall

ex2:
 li $v0 5
 syscall
 addiu $a0 $v0 42
 li $v0 1
 syscall
 li $v0 10
 syscall

ex3:
 li $v0 0
 store:
 #store
 sw $v0 0($sp)
 subi $sp $sp 4
 # read integer
 li $v0 5
 syscall
 bnez $v0 store

 li $v0 1
 printer:
 addiu $sp $sp 4
 lw $a0 0($sp)
 beqz $a0 end
 syscall
 j printer

 end:
 li $v0 10
 syscall
```