# *Übungen zu Rechnerarchitektur*

Sommersemester 2017

Prof. Jian-Jia Chen

Technische Universität Dortmund

Lehrstuhl Informatik 12

Entwurfsautomatisierung für Eingebettete Systeme

## 12.2 Multithreading (5 Punkte)

In dieser Aufgabe sollen Sie verschiedene Multithreading-Modelle exemplarisch anwenden. Gegeben sei dazu ein System mit

- einem Prozessor mit dynamischem Scheduling und
  - 2 Integer-Ausführungseinheiten (Latenz 1 Takt)
  - 1 Load/Store-Einheit (Latenz 1-4 Takte, je nach Cacheverhalten)
  - 1 Floating-Point-Einheit (Latenz 1 Takt)
- einem L1-Cache und einem L2-Cache. Die Latenzen der Zugriffe sind wie folgt:

| L1-Zugriff | L2-Zugriff | Latenz (Takte) |
|---|---|---|
| Hit | - | 1 |
| Miss | Hit | 2 |
| Miss | Miss | 4 |

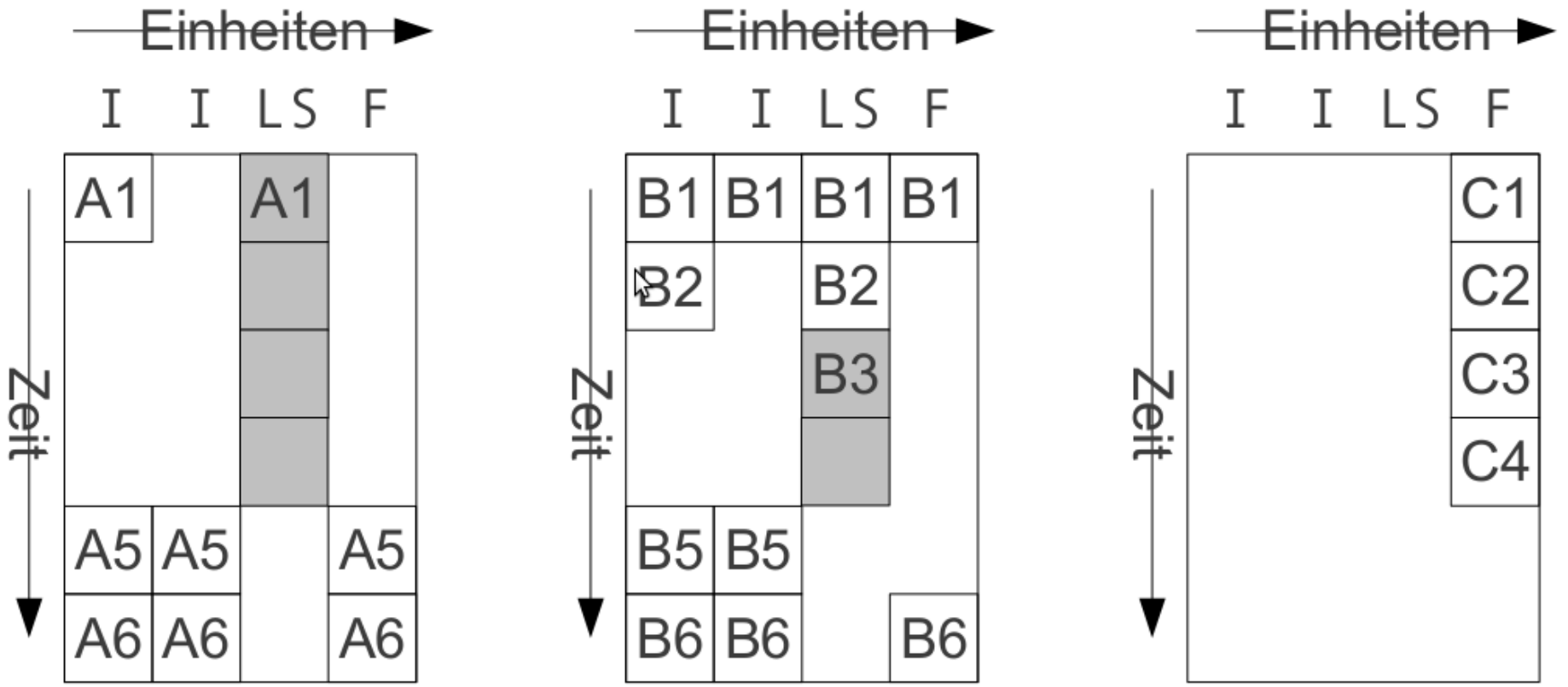
Beide Caches unterstützen *Hit-Under-1-Miss*, d.h. während ein Miss bearbeitet wird, können weitere Hit-Zugriffe abgewickelt werden. Ein Zugriff, der einen weiteren Miss produzieren würde, wird abgebrochen und muss vom Prozessor wiederholt werden.

# Multithreading

Für dieses System sind die Ausführungsverläufe dreier Programme $A$-$C$ in Abbildung 1 skizziert, wie sie sich bei isolierter Ausführung auf dem System ergeben. Operationen, die mehr als einen Takt benötigen, sind in hellgrau markiert. Außerdem ist die Priorität jedes Threads angegeben (nur bei CGMT und SMT nötig), wobei niedrigere Zahlen höherer Priorität entsprechen.

a) Zeichnen Sie für eine Ausführung mit *Coarse-Grained Multithreading (CGMT)*, *Fine-Grained Multithreading (FGMT)* und *Simultaneous Multithreading (SMT)* jeweils den zeitlichen Verlauf der Abarbeitung auf dem gegebenen System. Richten Sie sich bei der Notation nach Abbildung 1, d.h. tragen Sie die Verarbeitungseinheiten horizontal und die Zeit vertikal auf. Der *Context-Switch-Overhead* sei $1$ Takt für CGMT und $0$ Takte für die anderen Modelle.

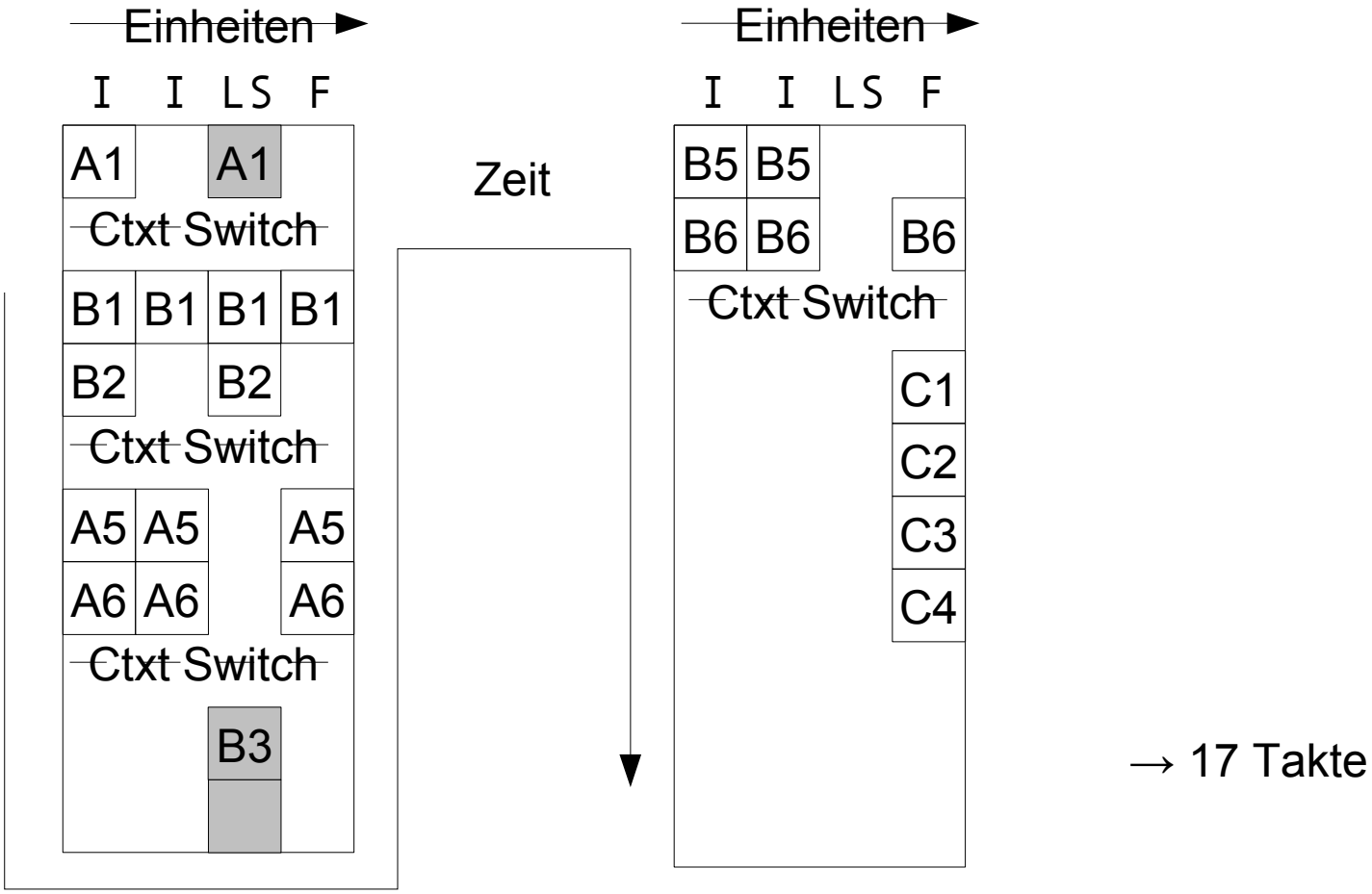
   Sie können davon ausgehen, dass die Speicherzugriffe der Threads auf disjunkte Cache-Bereiche abgebildet werden, d.h. dass diese Zugriffe interferenzfrei sind. Außerdem sei der verwendete Scheduler ein ASAP-Scheduler (*As Soon As Possible*). Falls nicht alle Instruktionen aus einem Zeitschritt aus den Abbildungen 1(a) bis 1(c) gleichzeitig gestartet werden können, führt der Scheduler die verbleidenden so bald wie möglich danach aus. Nehmen Sie in diesem Fall außerdem an, dass erst *alle* Instruktionen eines Zeitschritts $i$ aus Abbildung 1 ausgeführt worden sein müssen, bevor Instruktionen aus Schritt $i+1$ desselben Threads ausgeführt werden können.



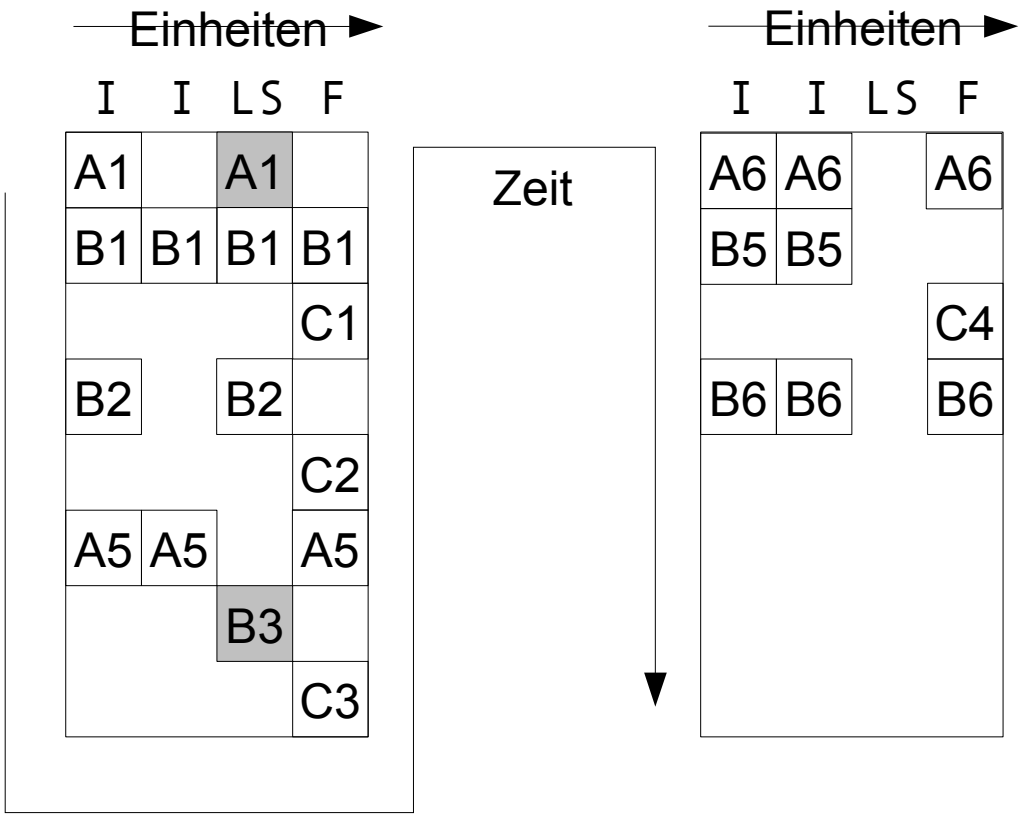

(a) Thread A (Priorität 0) (b) Thread B (Priorität 1) (c) Thread C (Priorität 2)

Abbildung 1: Ausführungsverlauf dreier Threads auf der gegebenen Plattform

# CGMT

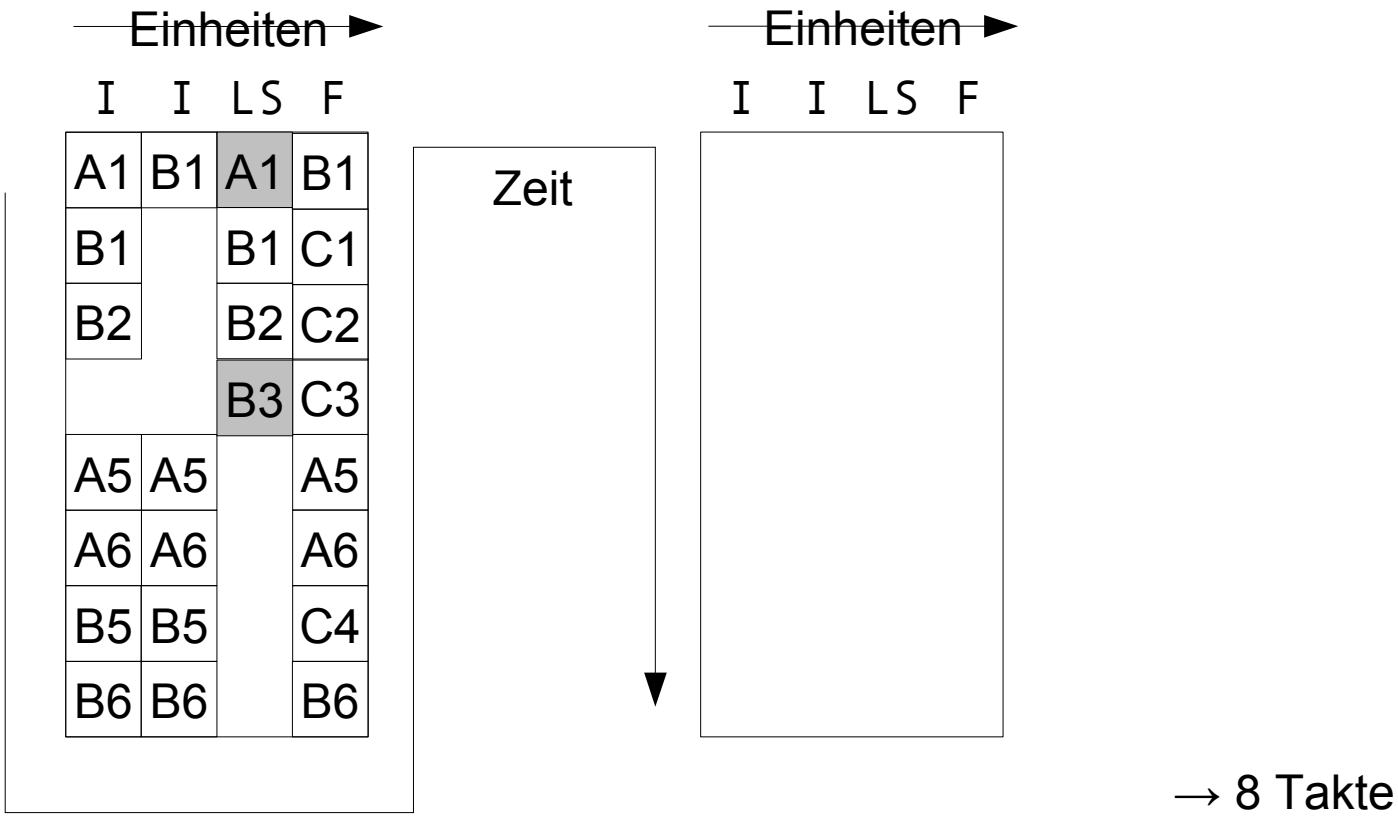

Einheiten
I I LS F
A1 A1
Ctxt Switch
B1 B1 B1 B1
B2 B2
Ctxt Switch
A5 A5 A5
A6 A6 A6
Ctxt Switch
B3
Zeit
Einheiten
I I LS F
B5 B5
B6 B6 B6
Ctxt Switch
C1
C2
C3
C4
→ 17 Takte

# FGMT




→ 12 Takte

# SMT


Einheiten
I I LS F
A1 B1 A1 B1
B1 B1 C1
B2 B2 C2
B3 C3
A5 A5 A5
A6 A6 A6
B5 B5 C4
B6 B6 B6
Zeit
Einheiten
I I LS F
→ 8 Takte

# Grobgranulares Multithreading

## Grobgranulares Multithreading (2 Punkte)

Für die Ausführungszeiten der Instruktionen aller Threads einer Anwendung gilt diese Verteilung:

- 40% der Taktzyklen entfallen auf die Ausführung der Instruktionen im Prozessor.
- 30% der Taktzyklen sind Warte-Zyklen, die durch L1-Cache-Misses (aber L2-Cache-Hits) entstehen. Die L1-Miss-Bearbeitung benötigt 10 Taktzyklen.
- 30% der Taktzyklen sind Warte-Zyklen, die durch L2-Cache-Misses hervorgerufen werden (jeweils 30 Taktzyklen).

Im Durchschnitt tritt pro Thread nach jeweils 10 Rechen-Takten ein Cache-Miss auf. Beantworten Sie folgende Fragen unter der Annahme, dass ausreichend viele Threads sowohl von der Hardware als auch durch die Anwendung unterstützt werden.

a) Wie hoch ist die maximale Auslastung, wenn ein Threadwechsel 5 Taktzyklen dauert?

b) Wie lange darf ein Threadwechsel höchstens dauern, so dass die Auslastung mindestens 50% beträgt?

# Annahme

- Thread arbeitet *R=10* Zyklen zwischen zwei Cache-Misses.

→Threadwechsel lohnt bei jedem Cache-Miss
(weil Ziel-Thread Instruktionen zum Ausführen hat)

→Maximal zu überbrückende Wartezeit: 30 Takte

→Bei mindestens 3 Threads (5 Takte Context-Switch) gibt es immer einen Thread, der nicht wegen eines Cache-Misses wartet.

# Threads

| | |
|---|---|
| Instruktionsausführung | 10 Taktzyklen |
| L1-Cache-Miss | 10 Taktzyklen |
| Instruktionsausführung | 10 Taktzyklen |
| L1-Cache-Miss | 10 Taktzyklen |
| Instruktionsausführung | 10 Taktzyklen |
| L1-Cache-Miss | 10 Taktzyklen |
| Instruktionsausführung | 10 Taktzyklen |
| L2-Cache-Miss | 30 Taktzyklen |
| | 100 Zyklen |

# Verbesserte Auslastung

- **In der Wartezeit kann anderer Thread ausgeführt werden**
  - Wartezeiten werden versteckt.
  - Auslastung wird erhöht.

Ohne Threadwechsel

80 Zyklen
davon 40 Zyklen für Instruktionen

Mit Threadwechsel

80 Zyklen
davon 50 Zyklen für Instruktionen

10 Zyklen Instruktionsausführung
10 Warte-Zyklen wegen L1-Cache-Miss
5 Zyklen Threadwechsel

# Auslastung

a) Wie hoch ist die maximale Auslastung, wenn ein Threadwechsel 5 Taktzyklen dauert?

- **Auslastung *U*:**

$$U = \frac{\text{busy}}{\text{busy+context switch+waiting}}$$

  - Abwechselnd: 10 Zyklen Busy, 5 Zyklen Threadwechel

$$U = \frac{10}{10+5+0} = \frac{2}{3}$$

# Latenz für Threadwechsel

b) Wie lange darf ein Threadwechsel höchstens dauern, so dass die Auslastung mindestens 50% beträgt?

- Weiterhin keine aktiven Wartezeiten durch Cache-Misses.

$$U = \frac{10}{10+X+0} = \frac{1}{2}$$

- $X = 10$
- Threadwechsel darf höchstens 10 Zyklen dauern