KIT-Fakultät für Informatik
Prof. Dr. Tamim Asfour, Prof. Dr. Uwe D. Hanebeck, Prof. Dr. Jörg Henkel

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren & Rechnerorganisation

und

Technische Informatik I/II

am 24. Februar 2017, 14:00 – 16:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren/TI-1	
Aufgabe 1	9 von 9 Punkten
Aufgabe 2	6 von 6 Punkten
Aufgabe 3	10 von 10 Punkten
Aufgabe 4	8 von 8 Punkten
Aufgabe 5	12 von 12 Punkten

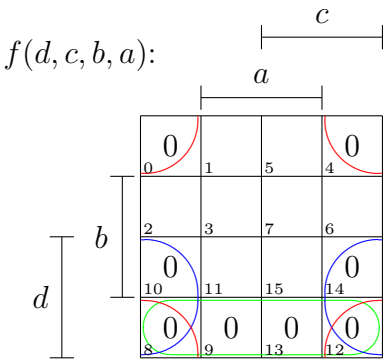
Rechnerorganisation/TI-2	
Aufgabe 6	7 von 7 Punkten
Aufgabe 7	12 von 12 Punkten
Aufgabe 8	8 von 8 Punkten
Aufgabe 9	10 von 10 Punkten
Aufgabe 10	8 von 8 Punkten

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1

1. 3 P.



Die Primimplikate sind:

1. $(\overline{d} \vee b)$ **2.** $(b \vee a)$ **3.** $(\overline{d} \vee a)$

2. Konjunktive Minimalformen: es existiert nur eine KMF und sie lautet: 1 P.

$$\begin{aligned} y_{KMF} &= \mathbf{1} \wedge \mathbf{2} \wedge \mathbf{3} \\ &= (\overline{d} \vee b) \wedge (b \vee a) \wedge (\overline{d} \vee a) \end{aligned}$$

3. Ausgangsgleichung für das Nelson-Verfahren: 1 P.
Die kürzeste Gleichung ist die KMF (siehe Aufgabenteil 1.2)

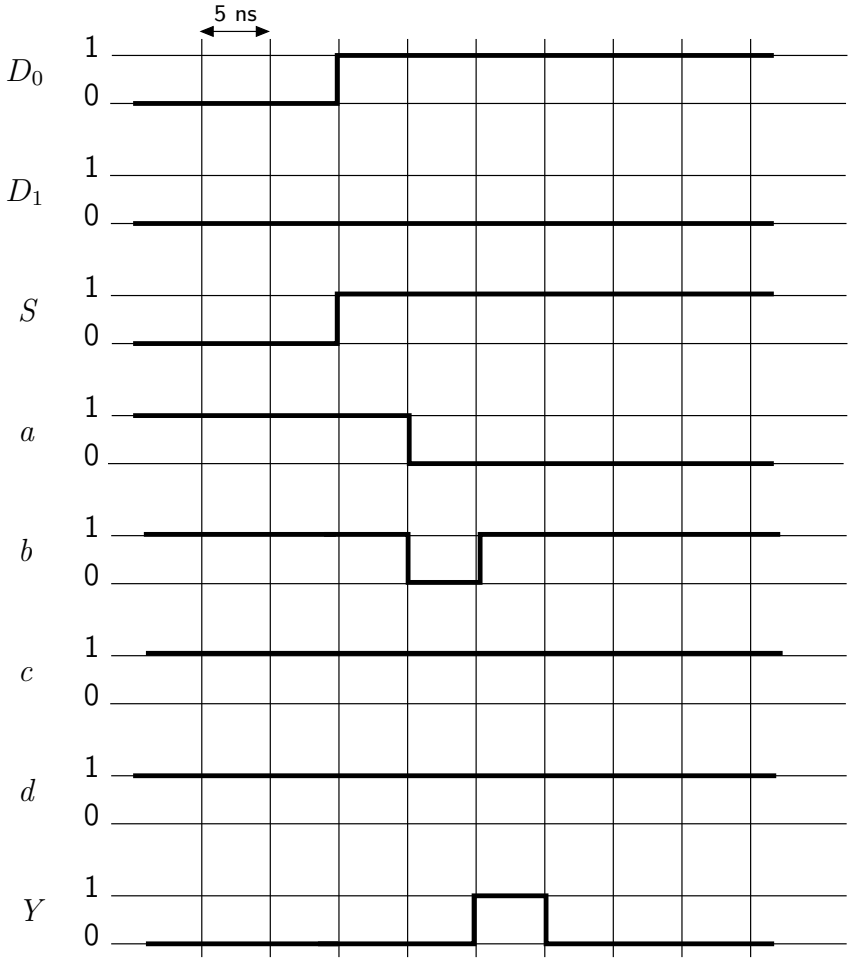
4. 4 P.

Nr.	gebildet aus	Würfel				gestrichen wegen
1		0	0	–	1	$\subset 6$
2		0	0	–	0	$\subset 6$
3		0	1	–	0	$\subset 10$
4		1	0	0	–	$\subset 9$
5		1	1	0	0	$\subset 8$
6	2,1	0	0	–	–	
7	5,4	1	–	0	0	$\subset 11$
8	5,3	–	1	0	0	$\subset 11$
9	6,4	–	0	0	–	
10	6,3	0	–	–	0	
	7,6	–	0	0	0	$\subset 11$
11	9,8	–	–	0	0	

Die Menge der Primimplikanten: $\{\overline{d}\overline{c},\quad \overline{c}\overline{b},\quad \overline{d}\overline{a},\quad \overline{b}\overline{a}\}$

Aufgabe 2

1. Zeitdiagramm:



2. Hasardfehler und Analyse:

$$F \;=\; (D_0 \barwedge \overline{S}) \barwedge (D_0 \barwedge D_1) \barwedge (S \barwedge D_1) \;=\; D_0\,\overline{S} \;\vee\; D_0\,D_1 \;\vee\; S\,D_1$$

F:

D_1 \ D_0		1		
	0	1	0	0
D_1	0	1	1	1
				S

Das Schaltnetz ist als Disjunktion aller Primimplikanten der Funktion F realisiert $\Rightarrow$ Schaltnetz ist frei von allen statischen Strukturhasards.

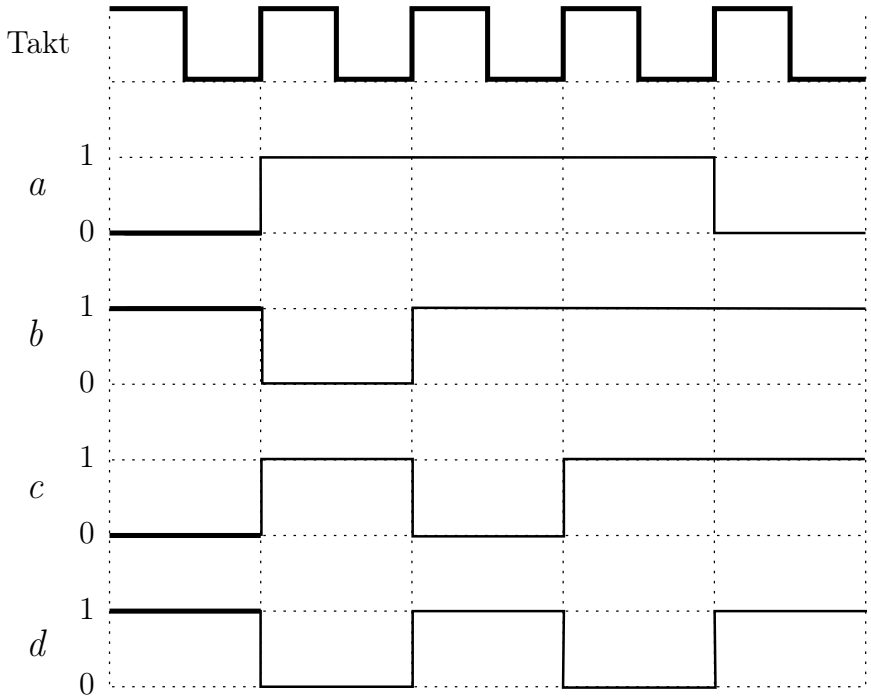
Der Übergang erfolgt über den Weg $B_0 \to B_1 \to B_5$, da $\tau_{D_0} < \tau_S$ ist ($\tau_{D_0} = 10ns,\ \tau_S = 15ns$) $\Rightarrow$ nicht monotone Folge der Funktionswerte ($0 \to 1 \to 0$)
$\Rightarrow$ Es liegt ein statischer 0-Funktionshasard vor.

Aufgabe 3

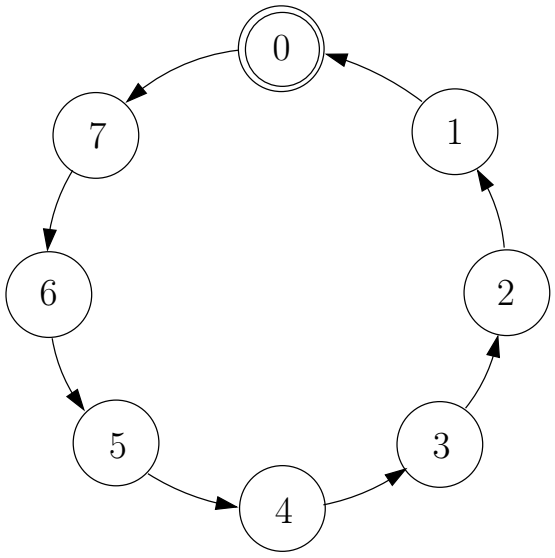
1. (a) Das Schaltwerk ist *synchron*

   (b) Maximale Anzahl der Zustände ist: $2^4 = 16$ Zustände

   (c) Verläufe der Signale a, b, c und d:



2. (a) Automatengraph:



(b) Kodierte Ablauftabelle:

q_2^t	q_1^t	q_0^t	q_2^{t+1}	q_1^{t+1}	q_0^{t+1}	d_2^t	d_1^t	d_0^t
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	1
0	1	1	0	1	0	0	1	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	0	1	0	0
1	1	0	1	0	1	1	0	1
1	1	1	1	1	0	1	1	0

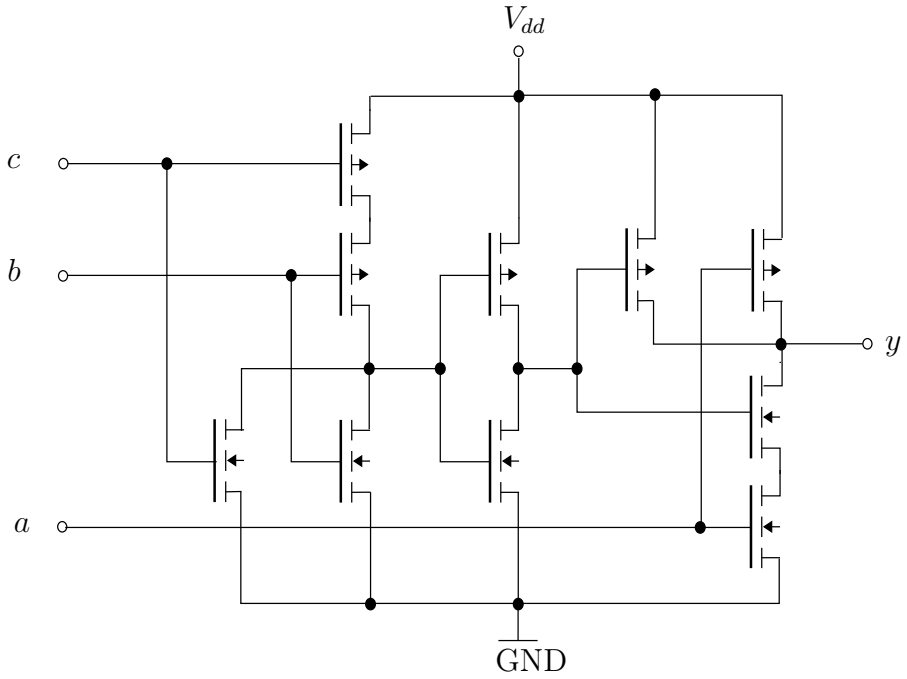
(c) Minimalformen der Ansteuerfunktionen der Flipflops: Aus der Ablauftabelle ablesbar oder mit Hilfe von KV-Diagrammen ermittelbar.

$$
\begin{aligned}
d_2^t &= q_2^t q_0^t \ \vee \ q_2^t q_1^t \vee \overline{q}_2^t \overline{q}_1^t \overline{q}_0^t \\
d_1^t &= \overline{q}_1^t \overline{q}_0^t \ \vee \ q_1^t q_0^t \\
d_0^t &= \overline{q}_0^t
\end{aligned}
$$

Aufgabe 4

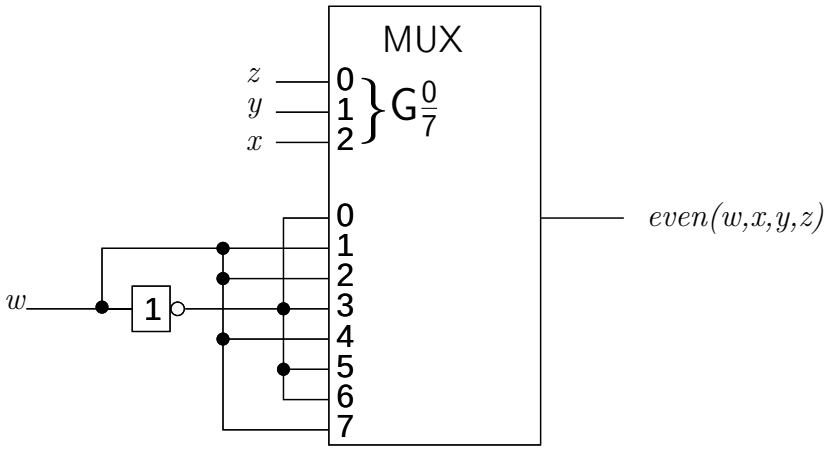
1. CMOS-Transistor-Schaltbild von $f(c, b, a)$:

$$f(c,b,a) \;=\; \overline{(c \;\vee\; b) \wedge\; a} \;=\; (c \;\vee\; b)\; \overline{\wedge}\; a \;=\; \overline{(b\; \overline{\vee}\; c)} \overline{\wedge}\; a$$



2. Schaltnetz von $p = \text{even}(w, x, y, z)$:

$$p = \text{even}(w,x,y,z) = \text{MINt}(0, 3, 5, 6, 9, 10, 12, 15)$$



Aufgabe 5

1. BCD-Addition:

$$\begin{array}{rrr} & 0011 & 1000 \\ + & 0110 & 0100 \\ \hline & 1001 & 1100 \\ & 0110 & 0110 \\ \hline 0001 & 0000 & 0010 \end{array}$$

2. Vor- und Nachteile der BCD-Arithmetik:

   - Die BCD-Arithmetik ist genauer. Zahlen, die im Dezimalsystem exakt dargestellt werden können (z.B. 0,1) lassen sich auch als BCD-Zahl exakt darstellen.
   - BCD-Berechnungen sind langsamer und Zahlen benötigen mehr Speicher.

3. Anzahl Prüfbits für Hamming-Code bei 32-Bit-Datenwort:

   Für $m = 32$ Datenbits kann gemäß der Bedingung $2^k \geq m + k + 1$ die notwendige Anzahl an Korrekturbits bestimmt werden. Es ergibt sich $k = 6$.

4.a) $-1,375_{10}$ im angegebenen Format:

$$-1,375_{10} \;=\; (-1,011)_2 \;=\; (-1,011)_2 \cdot 2^0$$
$$Vz = 1,\; Exp = 0 \;\Rightarrow\; Char \;=\; Exp + 15 \;=\; 15_{10} \;=\; 01111_2$$

$-1,375_{10}$ =

1	0	1	1	1	1	0	1	1	0	0	0

4.b) Kleinste darstellbare normalisierte positive Zahl:

$Char = 1 \;\Rightarrow\; Exp = -14_{10} \qquad (1, Mantisse)_2 = (1,000000)_2$

0	0	0	0	0	1	0	0	0	0	0	0

Dezimalwert:

$$(-1)^0 \cdot (1,000000)_2 \cdot 2^{-14} \;=\; 1,0 \cdot 2^{-14} \;\; (\approx\; 6,103515 \cdot 10^{-5})$$

4.c) (Betragsmäßig) größte darstellbare normalisierte negative Zahl:

$Char = 30 \;\Rightarrow\; Exp = 15_{10} \qquad (1, Mantisse)_2 = (1,111111)_2$

1	1	1	1	1	0	1	1	1	1	1	1

Dezimalwert:

$$\begin{aligned} (-1)^1 \cdot (1,111111)_2 \cdot 2^{15} &= -\left(1 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6}\right) \cdot 2^{15} \\ &= -\left(2^{16} - 2^9\right) = -\,65024 \end{aligned}$$

4.d) Verschiedene Darstellungen der Null:

$Char = 0$, Vz und $Mantisse$ beliebig $\Rightarrow\ 2 \cdot 2^6 \ = \ 2^7$ verschiedene Darstellungen.

4.e) $Char_A = 17 \ \rightarrow \ Exp_A = 2 \ \Rightarrow \ A = +(1,011011)_2 \cdot 2^2$

$Char_B = 15 \ \rightarrow \ Exp_B = 0 \ \Rightarrow \ B = -(1,101010)_2 \cdot 2^0$

$$A + B \ = \ (1,011011 - 0,011010)_2 \cdot 2^2 \ = \ (1,000001)_2 \cdot 2^2$$

$$\begin{aligned} A * B \ &= \ (1,011011 * 1,101010)_2 \cdot 2^{2+0} \ = \ (10,0010110101110)_2 \cdot 2^2 \\ &= \ (1,0010110101110)_2 \cdot 2^3 \end{aligned}$$

A + B =

0	1	0	0	0	1	0	0	0	0	0	1

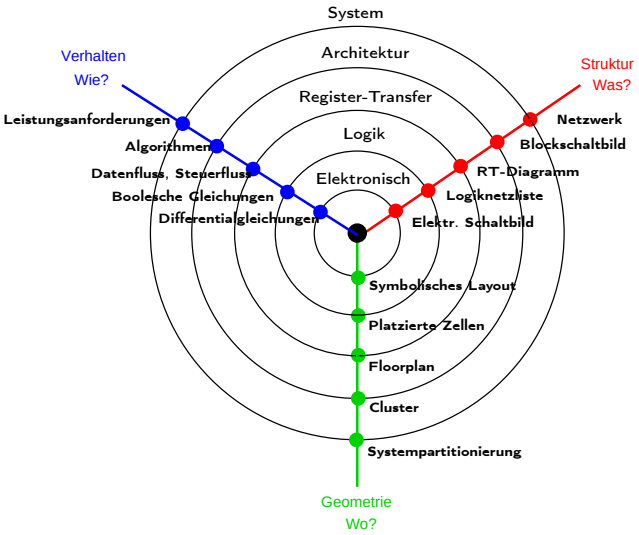
A * B =

1	1	0	0	1	0	0	0	1	0	1	1

Aufgabe 6

1. (a) Befehlsregister
   (b) Statusregister
   (c) Programmzähler
   (d) Rechenwerk (ALU)
2. (a) Einheitliche Befehlslänge (und einheitliches Befehlsformat)
   (b) Der Zugriff auf den Speicher erfolgt nur über *Load-Store-Befehle*
   (c) festverdrahtet
   (d) Getrennte Speicher und Busse für Befehle und Daten
3. Y-Diagramm:

   Das Y-Diagramm ist eine Darstellung der unterschiedlichen Ebenen des Entwurfs. Dabei wird der Abstraktionsgrad eines Entwurfs in so genannten Entwurfsebenen dargestellt. Die verschiedenen Repräsentationen eines Entwurfs auf einer Ebene sind als drei prinzipiell unterschiedliche Sichten (funktionell, strukturell, physikalisch) dargestellt.


System
Architektur
Register-Transfer
Logik
Elektronisch
Verhalten
Wie?
Struktur
Was?
Leistungsanforderungen
Algorithmen
Datenfluss, Steuerfluss
Boolesche Gleichungen
Differentialgleichungen
Netzwerk
Blockschaltbild
RT-Diagramm
Logiknetzliste
Elektr. Schaltbild
Symbolisches Layout
Platzierte Zellen
Floorplan
Cluster
Systempartitionierung
Geometrie
Wo?

Aufgabe 7

1. Inhalte der Zielregister:

Befehl	Zielregister = (z. B. `$s6 = 0x0000 F00A`)
`andi $s1, $zero, 40`	`$s1 = 0x0000 0000` = 0
`sra $s2, $t1, 2`	`$s2 = 0x0000 0010` = 16
`slt $s3, $zero, $t2`	`$s3 = 0x0000 0001` = 1
`sub $s4, $t3, $t2`	`$s4 = 0x0000 0002` = 2
`xori $s5, $t4, -1`	`$s5 = 0x0000 0004` = 4

2. Pseudobefehl:

   Pseudobefehle stellen eine Erweiterung der Assemblerbefehle dar. Sie werden vom Assembler in eine Folge von Maschinenbefehlen umgesetzt. Für den Assemblerprogrammierer unterscheiden sie sich nicht von den echten Maschinenbefehlen.

   Assemberdirektive:

   Assemblerdirektiven sind Anweisungen an den Assembler, die z. B. der Steuerung des Assembliervorgangs, der Platzierung vom Programmcode und Daten, der statischen Allokierung von Speicherplatz, der Wertzuweisung an Operanden und der Erzeugung von Konatanten dienen.

3. Trennung von Programmen und Daten bei der MIPS-R2000-Architektur:

   Der Hauptspeicher wird in mehreren Segementen unterteilt. Es existiert ein Datensegment für die Daten und ein Textsegment für Programme.

4. (a) Registerinhalte:

Register	Inhalt
`$t1`	`$t1 = 0x0000 0014`
`$t2`	`$t2 = 0x0000 0010`
`$t3`	`$t3 = 0x0000 0000`
`$t4`	`$t4 = 0xFFFF FFFC`

   (b) MIPS-Code zur Speicherung der Adresse von `vec` im Register `$s0`:

```
la $s0, vec
```

(c) Programmschleife zur Ausgabe von Partialsummen aus `vec`:

```
        .data
vec:    .word 20, 16, 12, 0, -3, 0, -4, 16, 20, 2014
leer:   .asciiz " "

        .text

        la   $s0, vec          # Adresse von vec in $s0
        addi $t0, $0, 0        # Zähler = 0
        addi $t1, $0, 10       # Zum Beenden der Schleife
        add  $t2, $0, $s0      # Zeiger auf aktuelles Element
        addi $t3, $0, 0        # Summenvariable
loop:   add  $a0, $0, $t3      # Auszugebende Zahl nach $a0 kopieren
        li   $v0, 1            # print_int
        syscall
        la   $a0, leer
        li   $v0, 4            # print-str (Leerzeichen)
        syscall
        lw   $t4, 0($t2)       # Element aus vec in $t4 laden
        add  $t3, $t3, $t4     # Auf Summenvariable addieren
        addi $t0, $t0, 1       # Zähler inkrementieren
        addi $t2, $t2, 4       # Zeiger auf vec inkrementieren
        bne  $t0, $t1, loop
```

Aufgabe 8

1. Datenabhängigkeiten:

   - Echte Abhängigkeiten:

   $$S_1 \rightarrow S_2 \ (\texttt{\$t1}) \qquad S_1 \rightarrow S_3 \ (\texttt{\$t1})$$
   $$S_2 \rightarrow S_3 \ (\texttt{\$t2}) \qquad S_2 \rightarrow S_5 \ (\texttt{\$t2})$$
   $$S_3 \rightarrow S_4 \ (\texttt{\$t3}) \qquad S_3 \rightarrow S_5 \ (\texttt{\$t3})$$

   - Gegen-Abhängigkeiten:

   $$S_2 \rightarrow S_4 \ (\texttt{\$t1}) \qquad S_3 \rightarrow S_4 \ (\texttt{\$t1})$$

   - Ausgabe- Abhängigkeiten:

   $$S_1 \rightarrow S_4 \ (\texttt{\$t1})$$

2. Beseitigung der Datenkonflikte:

```
S1:     anfang:  andi $t2, $t1, 1
                 NOP
                 NOP
S2:              beqz $t2, weiter
                 NOP
                 NOP
                 NOP
S3:              subi $t1, $t1, 1
S4:              j anfang
                 NOP
                 NOP
                 NOP
S5:     weiter:  srl  $t1, $t1, 1
S6:              j anfang
                 NOP
                 NOP
                 NOP
S7:              addi $t3, $t0,1
```

Aufgabe 9

1. (a) Größe eines Cache-Blocks in Byte:

   5 Bits Byte-Offset → Blockgröße: $2^5 = 32$ Byte

   (b) Kapazität des Cache-Speichers:

   11 Bits Index → 2^{11} Sätze

   A3-Cache $3 \times 2^{11} = 6144(6K)$ Cache-Blöcke mit je 32 Bytes

   Cache-Kapzität: $3 \times 2^{11} \times 32 = 3 \times 2^{11} \times 2^5$ Byte $= 3 \times 2^{16}$ Byte $= 192$ KByte

   (c) Der insgesamt erforderliche Speicherbedarf:

   Kapazität + (Tag-Länge + 2 Statusbits) × (Anzahl der Cache-Blöcke)
   192 KByte $+ (16 + 2) \cdot 3 \cdot 2^{11}$ Bit $=$
   192 KByte $+ 18 \cdot 3 \cdot 2^8$ Byte $=$
   192 KByte $+ 13,5 \cdot 2^{10}$ Byte $=$
   192 KByte + 13,5 KByte = 205,5 KByte

   (d) Zugriff auf die Adresse `0x00FE13A4`:

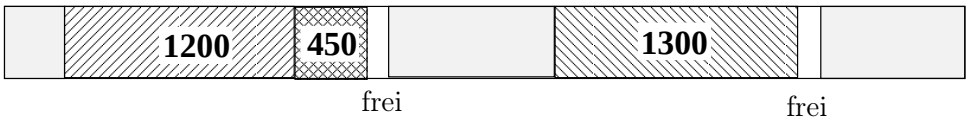
   A3-Cache → Es wird ein Vergleich mit 3 Zeilen im Cache durchgeführt.
   Satz+Byte-Index = 0001 0011 1010 0100 →
   Satz-Index = $0001\ 0011\ 101_2 = 1001\ 1101_2 = 128 + 16 + 8 + 4 + 1 = 157_{10}$
   → Der Vergleich wird mit den Zeilen 471 $(3 \cdot 157)$, 472 und 473 durchgeführt.

2.

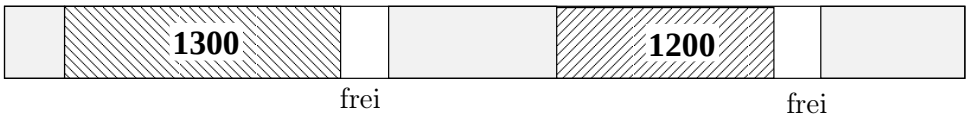
Adresse	**0x44**	**0xA0**	**0xC3**	**0x9E**	**0x66**	**0x2D**	**0x6B**	**0x49**
read/write	w	w	r	r	r	r	r	r
Hit/Miss	×	—	—	×	×	—	×	—
write back?	nein	ja	ja	nein	nein	ja	nein	nein

Aufgabe 10

1. Anzahl der Bits der virtuellen Adresse: 32 Bits
2. Anzahl der Bits der physikalischen Adresse: 25 Bits
3. Anzahl der Bits der virtuellen Seitennummer: 20 Bits
4. Anzahl der Bits der physikalische Seitennummer: 13 Bits
5. *first-fit*-Strategie:



best-fit-Strategie:



Für die Speicheranforderung 450 ist bei der *best-fit*-Strategie kein Platz mehr vorhanden

6. *best-fit*-Strategie:

   Vorteil: Geringere Fragmentierung und damit bessere Speicherausnutzung

   Nachteil: Höherer Suchaufwand gegenüber der *first-fit*-Strategie zum Finden einer freien Lücke.

   first-fit-Strategie:

   Kleine Lücken im Hauptspeicher sammeln sich an den unteren Adressen, während am Speicherende die größeren Lücken zu finden sind. Damit erhöht sich der mittlere Anzahl der erforderlichen Suchschritte zum Auffinden der ersten passenden Lücke.