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

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren (TI-1)

und

Rechnerorganisation (TI-2)

am 27. März 2021, 09:00 – 11:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren (TI-1)	
Aufgabe 1	11 von 11 Punkten
Aufgabe 2	10 von 10 Punkten
Aufgabe 3	6 von 6 Punkten
Aufgabe 4	10 von 10 Punkten
Aufgabe 5	8 von 8 Punkten

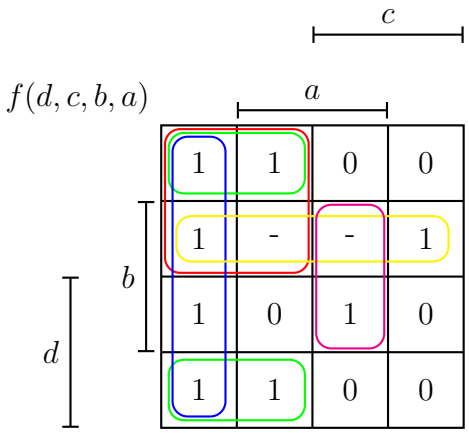
Rechnerorganisation (TI-2)	
Aufgabe 6	9 von 9 Punkten
Aufgabe 7	11 von 11 Punkten
Aufgabe 8	9 von 9 Punkten
Aufgabe 9	12 von 12 Punkten
Aufgabe 10	4 von 4 Punkten

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

Aufgabe 1 *Schaltfunktionen* (11 Punkte)

1. 4 P.



Primimplikanten:

$A: (\overline{d}\,\overline{c})$ $B: (\overline{c}\,\overline{b})$ $C: (\overline{c}\,\overline{a})$

$D: (\overline{d}\,b)$ $E: (c\,b\,a)$

2. Disjunktive Minimalform von $f(d, c, b, a)$: 1 P.

$$\begin{aligned} f(d,c,b,a) &= B \vee C \vee D \vee E \\ (&= \overline{c}\,\overline{b} \vee \overline{c}\,\overline{a} \vee \overline{d}\,b \vee c\,b\,a) \end{aligned}$$

3. Die Schaltfunktion ist unvollständig definiert, da 2 P.

   - Primimplikanten bei vollständig definierten Funktionen aus 2^n Mintermen bestehen. Der Primimplikanten A überdeckt 3 Minterme, d. h. er muss noch eine Freistelle enthalten. ODER
   - Primimplikant D ist im Primimplikanten A enthalten. Somit wäre A kein Primimplikant $\rightarrow$ Widerspruch.

4. Kernprimimplikanten: B und C 1 P.

5. Überdeckungsfunktion: 3 P.

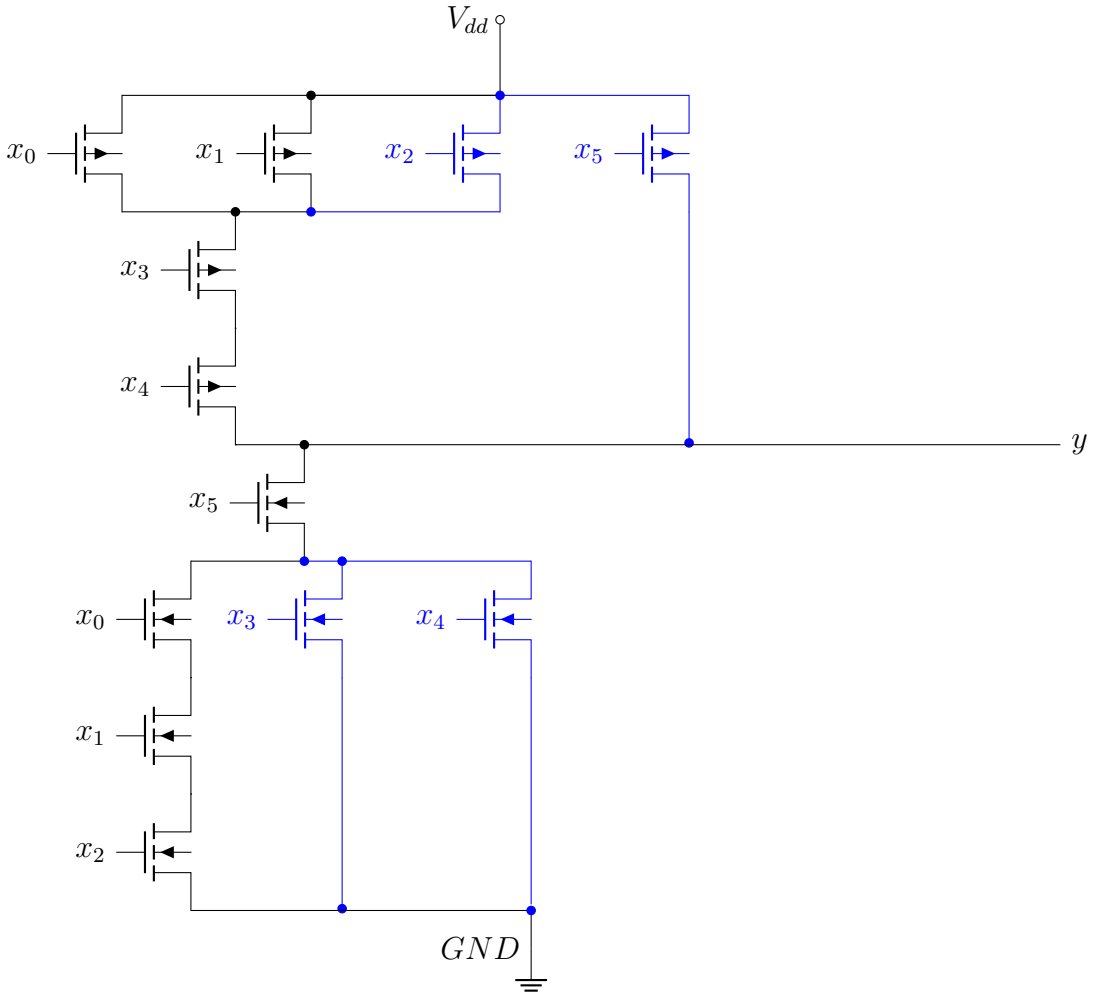
$$\begin{aligned} \ddot{u}_g &= (A \vee D)\,BC\,(A \vee D)\,(A \vee B) \\ &= (A \vee D)\,BC\,(A \vee B) \\ &= (ABC \vee BCD)(A \vee B) \\ &= ABC \vee ABC \vee ABCD \vee BCD \\ &= ABC \vee DBC \end{aligned}$$

Alternativ: A ist zeilendominant gegenüber D. Somit kann D gestrichen werden. Es folgt:

$$\ddot{u}_g = ABC$$

Aufgabe 2 *CMOS, Spezielle Bausteine* (10 Punkte)

1. 4 P.

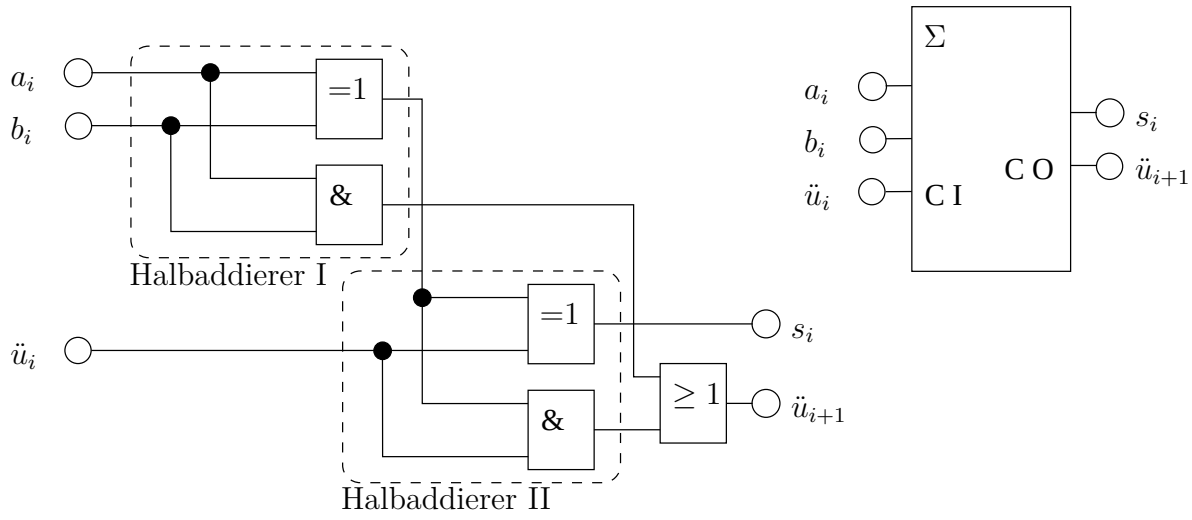


2. Realisierte Schaltfunktion: 2 P.

$$\begin{aligned} y &= \overline{x_5\,((\,x_0\;x_1\;x_2\,)\;\vee\;x_3\;\vee\;x_4\,)} \\ &= \overline{x}_5\;\vee\;\overline{x}_0\,\overline{x}_3\,\overline{x}_4\;\vee\;\overline{x}_1\,\overline{x}_3\,\overline{x}_4\;\vee\;\overline{x}_2\,\overline{x}_3\,\overline{x}_4 \end{aligned}$$

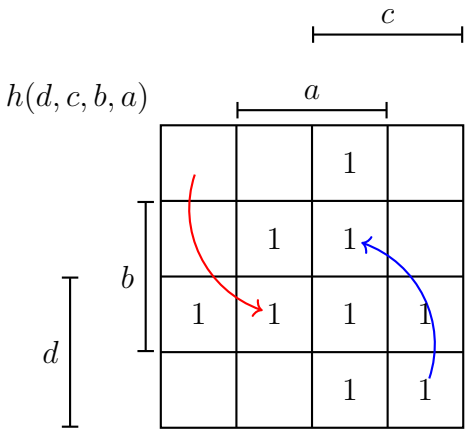
3. Unterschied zwischen Halbaddierer und Volladdierer: 1 P.
Ein Volladdierer berücksichtigt den Übertrag der vorhergehenden Stellen, deshalb besitzt er, zusätzlich zu den zwei Eingänge für die zu addierenden Dualziffern, einen Eingang für den Übertrag.

4. Schaltbild eines 1-Bit-Volladdierers: 3 P.



Aufgabe 3 *Laufzeiteffekte* (6 Punkte)

1. 2 P.



Übergang $(0,0,0,0) \rightarrow (1,0,1,1)$:

Übergang mit 3-Variablen-Wechsel $\Rightarrow$ $3! = 6$ Wege. Alle zugehörigen Folgen der Funktionswerte sind monoton $\Rightarrow$ Übergang ist frei von Funktionshasards.

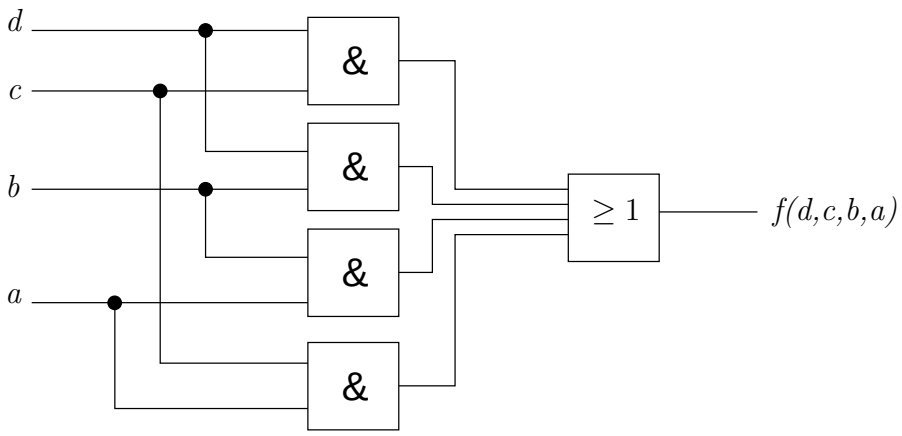
Übergang $(1,1,0,0) \rightarrow (0,1,1,1)$:

Übergang mit 3-Variablen-Wechsel $\Rightarrow 3! = 6$ Wege. Es existiert mindestens eine nicht-monotone Folge der Funktionswerte (z.B. $B_{12} \rightarrow B_{14} \rightarrow B_6 \rightarrow B_7$) $\Rightarrow$ Übergang ist mit einem statischen 1-Funktionshasard behaftet.

2. Strukturhasardfreie Realisierung: 4 P.

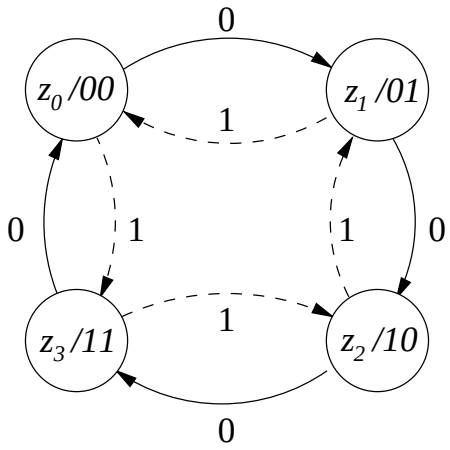
Disjunktion aller Primimplikanten oder Konjunktion aller Primimplikate (Satz von Eichelberger)

$$f(d,c,b,a) \;=\; d\,c \;\vee\; d\,b \;\vee b\,a \;\vee\; c\,a$$



Aufgabe 4 *Schaltwerke* (10 Punkte)

1. Automatengraph: 2 P.



Anzahl der erforderlichen Flipflops: 2

2. Kodierte Ablauftabelle: 3 P.

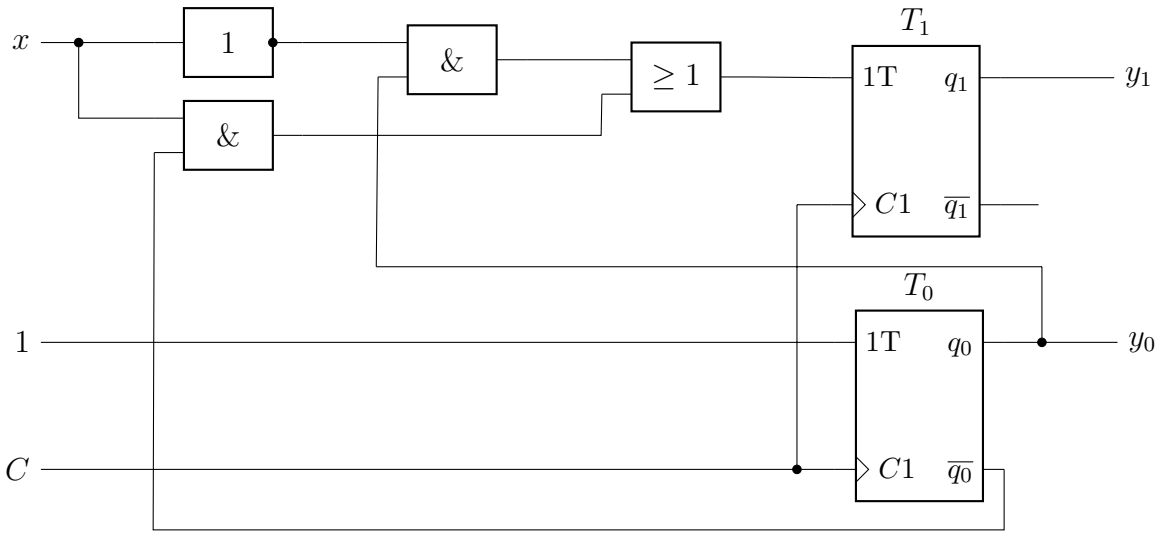
Eingabe	Zustand		Folgezustand		Ausgang		FF-Ansteuersignale	
x^t	q_1^t	q_0^t	q_1^{t+1}	q_0^{t+1}	y_1^t	y_0^t	T_1^t	T_0^t
0	0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	1	1
0	1	0	1	1	1	0	0	1
0	1	1	0	0	1	1	1	1
1	0	0	1	1	0	0	1	1
1	0	1	0	0	0	1	0	1
1	1	0	0	1	1	0	1	1
1	1	1	1	0	1	1	0	1

3. Ansteuerfunktionen der Flipflops: 2 P.

$$
\begin{aligned}
T_1 &= \overline{x}\ \overline{q_1}\ q_0 \ \vee\ \overline{x}\ q_1\ q_0 \ \vee\ x\ \overline{q_1}\ \overline{q_0} \ \vee\ x\ q_1\ \overline{q_0} \\
&= \overline{x}\ (\overline{q_1}\ q_0 \ \vee\ q_1\ q_0) \ \vee\ x\ (\overline{q_1}\ \overline{q_0} \ \vee\ q_1\ \overline{q_0}) \\
&= \overline{x}\ (q_0\ (\overline{q_1} \ \vee\ q_1)) \ \vee\ x\ (\overline{q_0}\ (\overline{q_1} \ \vee\ q_1)) \\
&= \overline{x}\ q_0 \ \vee\ x\overline{q_0}
\end{aligned}
$$

$$
T_0 = 1
$$

4. Schaltung des Schaltwerks: 3 P.



Aufgabe 5 *Rechnerarithmetik & Codes* (8 Punkte)

3 P.

1. $2021_{10} \;=\; 111\ 1110\ 0101_2$

   - 32-Bit Zweierkomplement-Format:

     $$0000\ 0000\ 0000\ 0000\ \ 0000\ 0111\ 1110\ 0101$$

   - 32-Bit IEEE-754-Gleitkomma-Format:

     $$111\ 1110\ 0101_2 \;=\; 1,11\ 1110\ 0101 \cdot 2^{10}$$

     $$Exp = 10 \Rightarrow Char \;=\; Exp + 127 = 137_{10} = 1000\ 1001_2$$

31	30 23	22 0
0	1000 1001	1111 1001 0100 0000 ... 000

3 P.

2. Datenwörter:

Position	11	10	9	8	7	6	5	4	3	2	1
	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1
Codewort 1:	1	0	0	0	1	1	0	1	0	1	0
Codewort 2:	0	1	1	0	0	1	0	1	0	1	1

   Die Prüfbits lassen sich nach den folgenden Regeln berechnen:

$$\begin{aligned} k_1 &= k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 \\ k_2 &= k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 \\ k_3 &= k_3 \oplus m_2 \oplus m_3 \oplus m_4 \\ k_4 &= k_4 \oplus m_5 \oplus m_6 \oplus m_7 \end{aligned}$$

   - Codewort 1: **1 0 0 0 1 1 0 1 0 1 0** $\Rightarrow$ $k_4\ k_3\ k_2\ k_1 = 1\ 1\ 0\ 0$ $\Rightarrow$ Es liegt angeblich ein Ein-Bit-Fehler an Position 12 vor. Es gibt aber keine Position 12 $\Rightarrow$ es liegt ein Mehrbit-Fehler vor $\Rightarrow$ Datenwort 1 kann nicht ermittelt werden.
   - Codewort 2: **0 1 1 0 0 1 0 1 0 1 1** $\Rightarrow$ $k_4\ k_3\ k_2\ k_1 = 0\ 0\ 1\ 0$ $\Rightarrow$ Es liegt ein Fehler an der 2. Postion vor $\Rightarrow$ Datenwort 2 = **0 1 1 0 1 0 0**.

3. BCD-Addition (Zahlen des Aufgabenblatts): 1 P.

		1001	0110
+		0111	0101
	0001	0000	1011
		0110	0110
	0001	0111	0001

ODER (Zahlen des Lösungsblatts):

		0011	1000
+		0110	0100
		1001	1100
+			0110
		1010	0010
+		0110	
	0001	0000	0010

4. Vor- und Nachteile der BCD-Arithmetik: 1 P.

- 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.

Aufgabe 6 *Die Programmiersprache C* (9 Punkte)

1. (a) Ausgabe: 1 P.

```
C-Teil - 1: Ausgabe lautet 18
```

(b) Ausgabe: 1 P.

```
C-Teil - 2: Ausgabe lautet 3
```

(c) Ausgabe: 2 P.

```
C-Teil - 3: Ausgabe lautet 50
```

(d) Ausgabe: 3 P.

```
C-Teil - 4: Ausgabe lautet MhjqpProzessor
```

2. Der Codeausschnitt konvertiert die Bitrepräsentation von z im IEEE-Format zu einem long. Dabei wird nicht die Zahl an sich konvertiert sondern lediglich die Bits: 2 P.
Beispiel ist nicht gefordert.

$$y = 1234.0_{10} = 0x449a4000 \quad \Big| \quad i = 0x449a4000 = 1150959616_{10}$$

Normaler Cast von Float nach long sähe so aus:

$$y = 1234.0_{10} = 0x449a4000 \quad \Big| \quad i = 1234.0_{10} = 0x4d2$$

Mit dieser Umwandlung ist es möglich zum Beispiel Shift-Operationen auf der Zahl anzuwenden. (Somit sind verschiedene Optimierungen möglich. Ein Beispiel hierfür ist die schnelle Berechnung der Invertierung der Quadratwurzel (`https://en.wikipedia.org/wiki/Fast_inverse_square_root`))

Aufgabe 7 *MIPS-Assembler* (11 Punkte)

1. Inhalte der Zielregister: 3 P.

Befehl	Zielregister = (z. B. `$s6 = 0x0000 F00A`)
`subi $s1, $zero, 0x4`	`$s1 = 0x0000 0004`
`sll $s2, $s1, 4`	`$s2 = 0x0000 0040`
`slti $s3, $s2, 100`	`$s3 = 0x0000 0001`
`lui $s4, 0x40`	`$s4 = 0x0040 0000`
`xor $s5, $s1, $s4`	`$s5 = 0x0040 0004`

2. Register- und Speicherinhalte nach der Ausführung: 6 P.

Registersatz

Register	Inhalt
`$t0`	`0x12`
`$t1`	**0x18**
`$t2`	`0x1E`
`$t3`	**0x24**
`$t4`	**0x10**

Hauptspeicher

Adresse	Inhalt
`$0x20`	`0x10`
`$0x24`	**0x10**
`$0x28`	`0x12`
`$0x2C`	`0xCF`
`$0x30`	**0x67**

3. (a) Little-Endian: 1 P.

Register	Wert = (z. B. `0x0000 F00A`)
`$t1`	`0x0000 0045`
`$t2`	`0x0000 0045`

(b) Big-Endian: 1 P.

Register	Wert = (z. B. `0x0000 F00A`)
`$t1`	`0x0000 00AB`
`$t2`	`0xFFFF FFAB`

Aufgabe 8 *Pipelining* (9 Punkte)

1. Datenabhängigkeiten:

   - Echte Abhängigkeiten: 3 P.

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

   - Gegen-Abhängigkeiten: 1 P.

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

   - Ausgabe- Abhängigkeiten: 1 P.

   $$S_1 \rightarrow S_4 \ (\texttt{\$t1}) \qquad S_5 \rightarrow S_6 \ (\texttt{\$t4})$$

2. Beseitigung der Datenkonflikte: 4 P.

```
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 *Cache- und Speicherverwaltung* (12 Punkte)

1. (a) Blockgröße in Bytes: 4 Bit Byte Offset ⇒ Blockgröße = $2^4 = 16$ Byte — 1 P.

   (b) Anzahl der Einträge: — 1 P.

$$\text{Anzahl der Einträge} = \frac{\text{Kapazität}}{\text{Blockgröße}} = \frac{1024\ KByte}{16\ Byte} = 64\ K \text{ Einträge}$$

   (c) Cache-Organisation: — 2 P.

   12 Bit Index-Feld ⇒ Es lassen sich $2^{12} = 4$ K Sätze im Cache adressieren

$$\text{Assoziativität} = \frac{64\ K}{4\ K} = 16$$

   Der Cache ist als 16-fach satzassoziativer Speicher (*16-way set associative*) organisiert.

2. — 4 P.

Adresse	**0x000**	**0xA29**	**0xA39**	**0xC26**	**0xA34**	**0x021**	**0x041**	**0xB11**
read/write	r	r	r	w	r	r	r	w
Index	0	2	3	2	3	2	4	1
Tag	0	A	A	C	A	0	0	B
Hit/Miss	M	M	M	M	H	M	M	M

3. Physikalische Adresse von 1444: — 4 P.

   $1444/256 = 5 +$ Rest 164 ⇒ virtuelle Seitennummer 5

   Aus der Tabelle ⇒ physikalische Seitennummer 7

   ⇒ physikalische Adresse ist: $7 * 256 + 164 = 1956$ oder $1444 + 2 * 256 = 1956$

   Physikalische Adresse von 789:

   $789/256 = 3 +$ Rest 21 ⇒ virtuelle Seitennummer 3

   Aus der Tabelle ⇒ physikalische Seitennummer 2

   ⇒ physikalische Adresse ist: $2 * 256 + 21 = 533$ oder $789 + (-1) * 256 = 533$

Aufgabe 10 *Allgemeines* (4 Punkte)

1. Das Y-Diagramm von D.D.Gajski enthält 3 Sichten und 5 Entwurfsebenen. 1 P.
2. Die Übersetzung der logischen Adresse zu einer physikalischen Adresse nimmt die Speicherverwaltungseinheit (*memory management unit*, MMU) vor. 1 P.
3. (a) Einheitliche Befehlslänge (und einheitliches Befehlsformat) 2 P.
   (b) Getrennte Speicher und Busse für Befehle und Daten