Karlsruher Institut für Technologie

Fakultät für Informatik
Prof. Dr. Tamim Asfour, Prof. Dr. Wolfgang Karl, Prof. Dr. Mehdi B. Tahoori

Musterlösungen zur Klausur

Digitaltechnik und Entwurfsverfahren/Rechnerorganisation

Technische Informatik I/II

am 24. Juli 2014, 14:00 – 16:00 Uhr

Name:	Vorname:	Matrikelnummer:
Bond	James	007

Digitaltechnik und Entwurfsverfahren/TI-1	
Aufgabe 1	7 von 7 Punkten
Aufgabe 2	6 von 6 Punkten
Aufgabe 3	10 von 10 Punkten
Aufgabe 4	11 von 11 Punkten
Aufgabe 5	11 von 11 Punkten

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

Gesamtpunktzahl:	90 von 90 Punkten

	Note: **1,0**

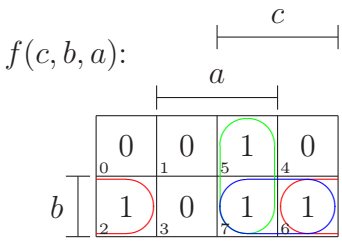
Aufgabe 1

1. KNF von $f(c,b,a)$:

   (Ablesen aus der Tabelle)

$$\begin{aligned} f(c,b,a) &= \text{MAXt}(0,1,3,4) \\ &= (c \vee b \vee a) \cdot (c \vee b \vee \overline{a}) \cdot (c \vee \overline{b} \vee \overline{a}) \cdot (\overline{c} \vee b \vee a) \end{aligned}$$

2. KV-Diagramm:



   Primimplikanten:

   - Kernprimimplikanten: $b \wedge \overline{a}$, $c \wedge a$
   - Entbehrliche Primimplikanten: $c \wedge b$

   DMF von $f(c,b,a)$: $(b \wedge \overline{a}) \vee (c \wedge a)$

3. Schaltnetz:

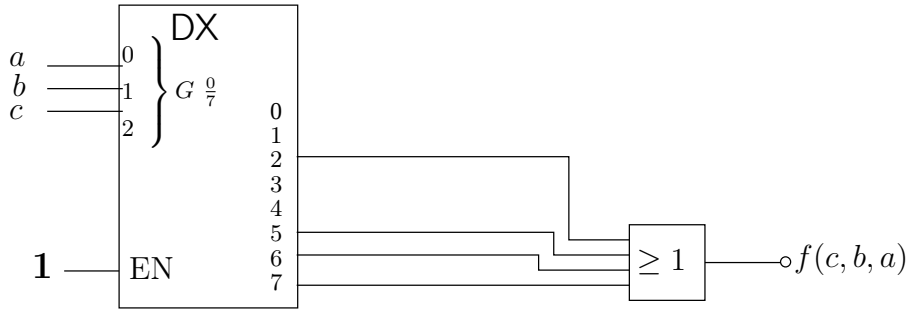


Abbildung 1: Schaltnetz

4. Schaltnetz:

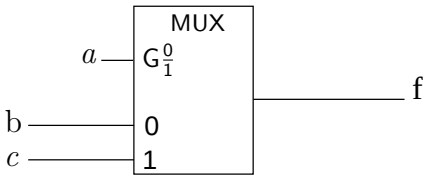


Abbildung 2: Schaltnetz

Aufgabe 2

1. Consensus-Verfahren:

Nr.	gebildet aus	Würfel d	c	b	a	gestrichen wegen
1		0	0	1	0	$\subset 7$
2		1	0	0	0	$\subset 6$
3		-	0	0	1	**Primimplikant** (A)
4		0	0	-	1	**Primimplikant** (B)
5		0	1	1	1	$\subset 8$
6	2,3	1	0	0	-	**Primimplikant** (C)
7	1,4	0	0	1	-	**Primimplikant** (D)
8	4,5	0	-	1	1	**Primimplikant** (E)

2. Überdeckungstabelle:

	1	2	3	7	8	9
A	×					×
B	×		×			
C					×	×
D		×	×			
E			×	×		

Vereinfachte Überdeckungstabelle (streiche Kernprimimplikanten C, D, E):

	1
A	×
B	×

Disjunktive Minimalform: $A \vee C \vee D \vee E$ oder $B \vee C \vee D \vee E$

Aufgabe 3

1. Schaltfunktion:

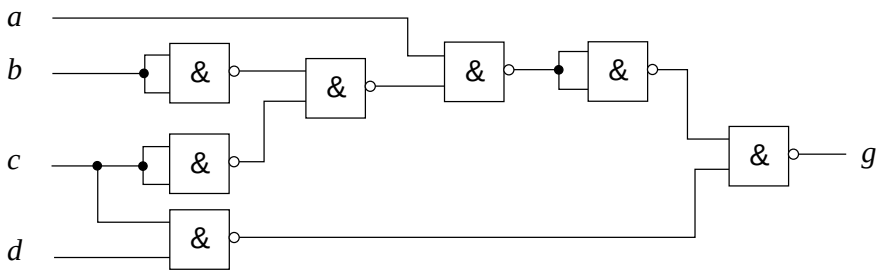
$$y \;=\; \Big((\overline{c} \leftrightarrow b)(\overline{b} \vee a)\Big) \vee \Big((\overline{c}\;\overline{a})(b \vee (a \nleftrightarrow b)\Big)$$

2. Zweistufige disjunktive Form von y

$$\begin{aligned}
y &= \Big((\overline{c} \leftrightarrow b)(\overline{b} \vee a)\Big) \vee \Big((\overline{c}\,\overline{a})(b \vee (a \nleftrightarrow b))\Big) \\
&= \Big((\overline{c}\,b \vee c\,\overline{b})(\overline{b} \vee a)\Big) \vee \Big((\overline{c}\,\overline{a})(b \vee (b\,\overline{a} \vee \overline{b}\,a))\Big) \\
&= \underbrace{\overline{c}\,b\,\overline{b}}_{=0} \vee \overline{c}\,b\,a \vee \underbrace{c\,\overline{b}\,\overline{b}}_{=c\,\overline{b}} \vee c\,\overline{b}\,a \vee \overline{c}\,\overline{a}\,(\underbrace{b \vee b\,\overline{a}}_{=b} \vee \overline{b}\,a) \\
&= \overline{c}\,b\,a \vee c\,\overline{b} \vee c\,\overline{b}\,a \vee \overline{c}\,\overline{a}\,b \vee \overline{c}\,\overline{a}\,\overline{b}\,a \\
&= \overline{c}\,b\,a \vee c\,\overline{b} \vee c\,\overline{b}\,a \vee \overline{c}\,b\,\overline{a} \\
&= c\,\overline{b} \vee \overline{c}\,b\,a \vee \overline{c}\,b\,\overline{a} = c\,\overline{b} \vee \overline{c}\,b\,(a \vee \overline{a}) \\
&= c\,\overline{b} \vee \overline{c}\,b
\end{aligned}$$

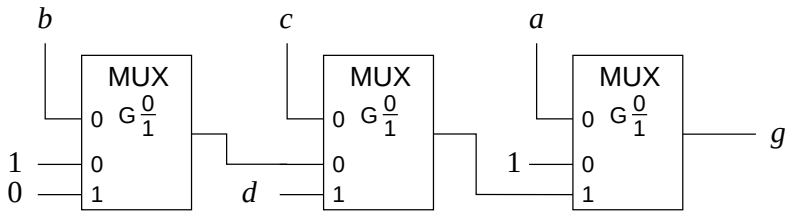
3. NAND-Schaltnetz:

$$\begin{aligned}
g(d,c,b,a) &= \overline{\overline{\overline{(\overline{a} \vee \overline{c}\,\overline{b}) \vee c\,d}}} = \overline{(a \wedge \overline{(\overline{c} \wedge \overline{b})) \wedge \overline{(c \wedge d)}}} \\
&= \overline{\overline{\overline{(a \wedge \overline{(\overline{c} \wedge \overline{b})) \wedge \overline{(c \wedge d)}}}}}
\end{aligned}$$



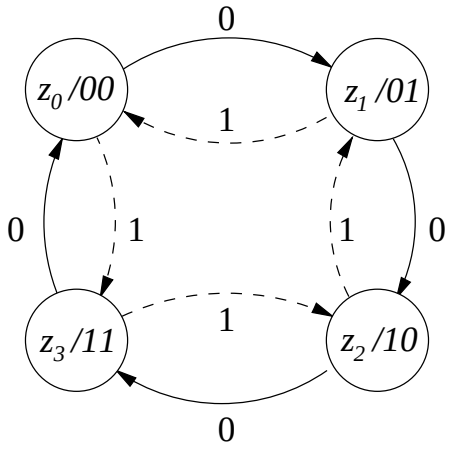
4. Multiplexer-Schaltnetz:

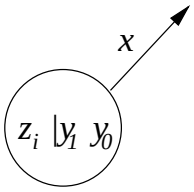
$$\begin{aligned}
g(d,c,b,a) &= \overline{a} \vee \overline{c}\,\overline{b} \vee c\,d \\
&= a\left[\overline{c}\,\overline{b} \vee c\,d\right] \vee \overline{a}\left[1\right] \\
&= a\left[\overline{c}\,(\overline{b}) \vee c\,(d)\right] \vee \overline{a}\left[1\right] \\
&= a\left[\overline{c}\left(b(0) \vee \overline{b}(1)\right) \vee c\left(d(1) \vee \overline{d}(0)\right)\right] \vee \overline{a}\left[1\right]
\end{aligned}$$



Aufgabe 4

1. Automatengraph:





   Anzahl der erforderlichen Flipflops: 2

2. Kodierte Ablauftabelle:

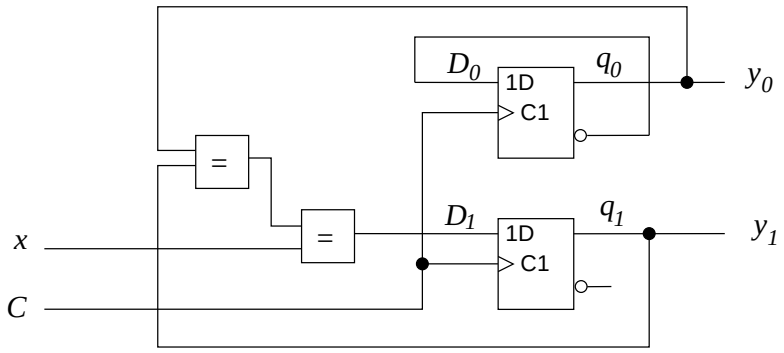
Eingabe	Zustand		Folgezustand		Ausgang		FF-Ansteuersignale	
x^t	q_1^t	q_0^t	q_1^{t+1}	q_0^{t+1}	y_1^t	y_0^t	D_1^t	D_0^t
0	0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	1	0
0	1	0	1	1	1	0	1	1
0	1	1	0	0	1	1	0	0
1	0	0	1	1	0	0	1	1
1	0	1	0	0	0	1	0	0
1	1	0	0	1	1	0	0	1
1	1	1	1	0	1	1	1	0

3. Ansteuerfunktionen der Flipflops:

$$
\begin{aligned}
D_1 &= \overline{x}\,\overline{q}_1 q_0 \vee \overline{x}\, q_1 \overline{q}_0 \vee x\,\overline{q}_1\overline{q}_0 \vee x\, q_1 q_0 \\
&= \overline{x}(\overline{q}_1 q_0 \vee q_1\overline{q}_0) \vee x(\overline{q}_1\overline{q}_0 \vee q_1 q_0) \\
&= \overline{x}(q_1 \not\leftrightarrow q_0) \vee x(q_1 \leftrightarrow q_0) \\
&= x \leftrightarrow (q_1 \leftrightarrow q_0)
\end{aligned}
$$

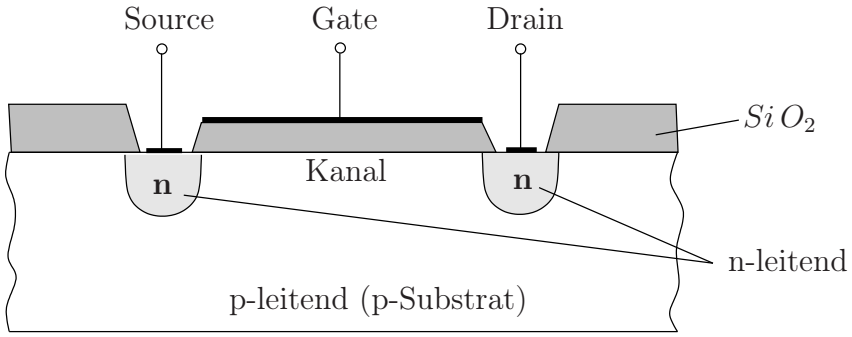
$$
\begin{aligned}
D_0 &= \overline{x}\,\overline{q}_1\overline{q}_0 \vee \overline{x}\, q_1\overline{q}_0 \vee x\,\overline{q}_1\overline{q}_0 \vee x\, q_1\overline{q}_0 \\
&= \overline{q}_1\overline{q}_0(\overline{x} \vee x) \vee q_1\overline{q}_0(\overline{x} \vee x) \;=\; (q_1 \vee \overline{q}_1)\overline{q}_0 = \overline{q}_0
\end{aligned}
$$

4. Schaltung des Schaltwerks:



Aufgabe 5

1. Aufbau eines nMOS-Transistors:

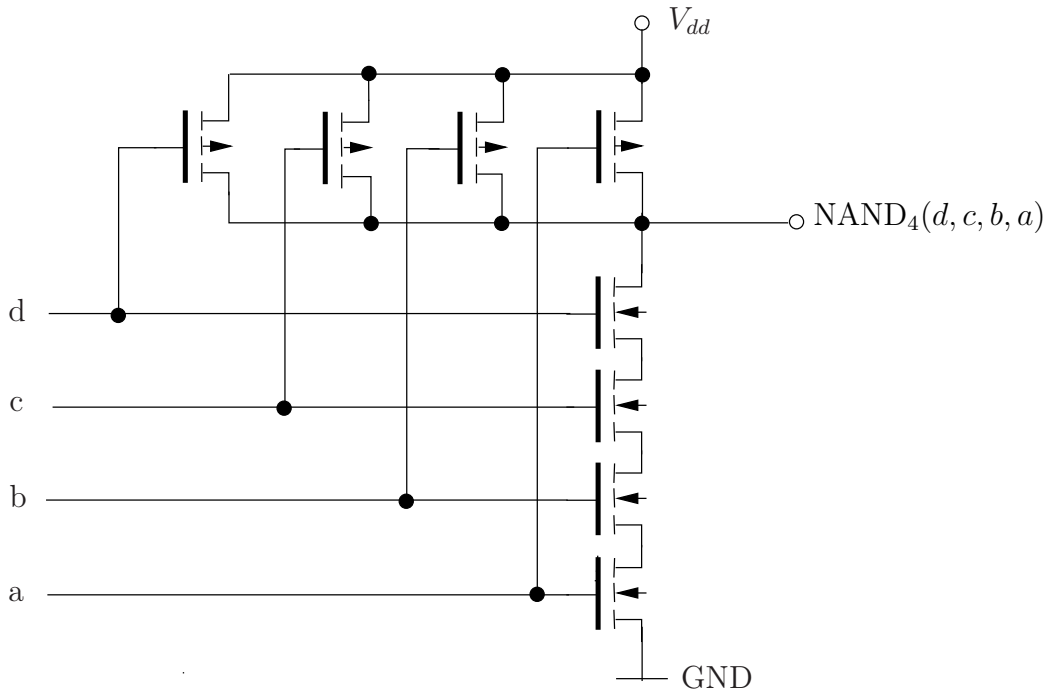


Selbstsperrender n-Kanal-MOSFET

2. Schaltfunktion $g(d, c, b, a)$:

$$\begin{aligned} g(d,c,b,a) &= \overline{\overline{b\,c \;\vee\; \overline{d}\,(\overline{b} \vee\; \overline{c}\,a)}} \;=\; b\,c \;\vee\; \overline{d}\,(\overline{b} \vee\; \overline{c}\,a) \\ &= b\,c \;\vee\; \overline{d}\,\overline{b} \;\vee\; \overline{d}\,\overline{c}\,a \end{aligned}$$

3. Transistorschaltbild:



4. Datenwörter: (Man beachte: $k_i = QS_{i-1}$)

Position	12	11	10	9	8	7	6	5	4	3	2	1
	m_8	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	1	0	1	1	0	1	0	1	0	0	1
Codewort 2:	0	1	1	0	0	0	1	0	1	1	0	1

Die Prüfbits lassen sich nach den folgenden Regeln berechnen:

$$\begin{aligned} k_1 &= k_1 \oplus m_1 \oplus m_2 \oplus m_4 \oplus m_5 \oplus m_7 \\ k_2 &= k_2 \oplus m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 \\ k_3 &= k_3 \oplus m_2 \oplus m_3 \oplus m_4 \oplus m_8 \\ k_4 &= k_4 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8 \end{aligned}$$

- Codewort 1: **1 1 0 1 1 0 1 0 1 0 0 1** $\Rightarrow$ $k_4\ k_3\ k_2\ k_1 = 0\ 1\ 0\ 1 \Rightarrow$ Es liegt ein Fehler an der 5. Position vor $\Rightarrow$ Datenwort 1: **1 1 0 1 0 1 1 0**
- Codewort 2: **0 1 1 0 0 0 1 0 1 1 0 1** $\Rightarrow$ $k_4\ k_3\ k_2\ k_1 = 0\ 0\ 0\ 1 \Rightarrow$ Es liegt ein Fehler an der 1. Postion vor $\Rightarrow$ Datenwort 2: **0 1 1 0 0 1 0 1**

5. Gray-Code: einschrittig, zyklisch

   Ausführung arithmetischer Operationen im Gray-Code ist schwierig, weil die Stellen des Codes keine feste Stellenwertigkeit besitzen.

Aufgabe 6

1. Repräsentation von 2014_6: `010 000 001 100`

2. Pseudokombinationen: `110` und `111`

3. Korrektur einer Pseudokombination:
   Es muss ein Übertrag erzeugt werden und $+6$ abgezogen werden $\Rightarrow +2$ addieren.

4. Korrektur bei einem Übertrag:
   Ein Übertrag bedeutet, daß das Ergebnis der Addition y von binär codierten Ziffern größer als 8_{10} ist. Übrig bleibt die Differenz $y - 8_{10}$. Es müsste jedoch die Differenz zu 6_{10} übrig bleiben $\Rightarrow +2$ ist zum Rest zu addieren.

5. Addition $2152_6 + 2014_6$:

	2152_6		010	001	101	010
$+$	2014_6	$+$	010	000	001	100
			1 0 0		0 1 0	
			100	001	110	110
					010	010
				0 1 1	1 0 1	1 0 0
$=$	4210_6	$=$	100	010	001	000

6. Dezimaler Wert des Ausdrucks:

$$\begin{aligned}(-1,01010) \cdot (10)^{101} &= -\left((2^0 + 2^{-2} + 2^{-4}) \cdot 2^5\right)_{10} \\ &= -(2^5 + 2^3 + 2^1)_{10} \\ &= -(32 + 8 + 2)_{10} = -42_{10}\end{aligned}$$

7. Ergebnis der Operation $x - y$:
   Es gilt:

$$\begin{aligned}x &= -(\texttt{011 1010 0111 1100}) \\ y &= +(\texttt{010 0100 0001 1010})\end{aligned}$$

   Subtraktion:

$$\begin{aligned}x - y &= -(y - x) \\ &= -(\texttt{101 1110 1001 0110})\end{aligned}$$

   Vorzeichen-Betrag-Form (Vorzeichenbit $= 1$):

$$\begin{aligned}x - y &= \texttt{1101 1110 1001 0110} \\ &= DE96_{16}\end{aligned}$$

Aufgabe 7

1. Mathematische Berechnung und Ausgabe des Programms:
   Das Programm berechnet x^y. Für $x = 2$ und $y = 3$ ist die Ausgabe daher $2^3 = 8$.

2. Implementierung von `printresult`:

```
printresult: li $v0, 1
             syscall
             jr $ra
```

3. MIPS-Instruktionen:

   (a) `move $s0, $s1`:

```
add $s0, $s1, $zero
```

   (b) `li $t0, 0x12345678`:

```
lui $t0, 0x1234
ori $t0, $t0, 0x5678
```

4. Assemblerdirektive `.align 2`:
   `.align 2` bewirkt, dass die folgenden Daten an der nächstmöglichen Adresse gespeichert werden, die durch $2^2 = 4$ teilbar ist.

Aufgabe 8

1. Daten- und Steuerflussabhängigkeiten:

   - Echte Datenabhängigkeiten (δ^t):

     S1-S2 S3-S4 S3-S5

   - Gegenabhängigkeiten (δ^a):

     S4-S5

   - Ausgabeabhängigkeiten (δ^o):

     S3-S5

   - Steuerflussabhängigkeiten:

     S4-S5

2. Keine Pipelinekonflikte bei:

   - S3-S5 (δ^t): Bei der Ausführung von S4 muss das Ergebnis von S3 bereits bekannt sein. Da S5 wiederum von S4 abhängt, kann zwischen S3 und S5 kein Datenkonflikt auftreten.
   - S4-S5 (δ^a): Da in der Pipeline alle möglichen Schreibvorgänge in späteren Stufen als alle möglichen Lesevorgänge erfolgen, können prinzipiell keine Konflikte aus Gegenabhängigkeiten auftreten.
   - S3-S5 (δ^o): Da in der Pipeline sämtliche Schreibvorgänge nur in einer Stufe (Stufe 5) erfolgen, können prinzipiell keine Konflikte aus Ausgabeabhängigkeiten auftreten.

3. Beseitigung aller Pipelinekonflikte durch NOP-Befehle:

```
S1:     SUB   R1, R2, 4         ; R1 = R2 - 4
        NOP
        NOP
S2:     SUB   R4, R1, R6        ; R4 = R1 - R6
S3:     MULT  R3, R5, 3         ; R3 = R5 * 3
        NOP
        NOP
S4:     BEQ   R2, R3, MARKE1    ; if R2 == R3 then PC := MARKE1
        NOP
        NOP
        NOP
        (NOP)
S5:     ADD   R3, R3, 2         ; R3 = R3 + 2
MARKE1:
```

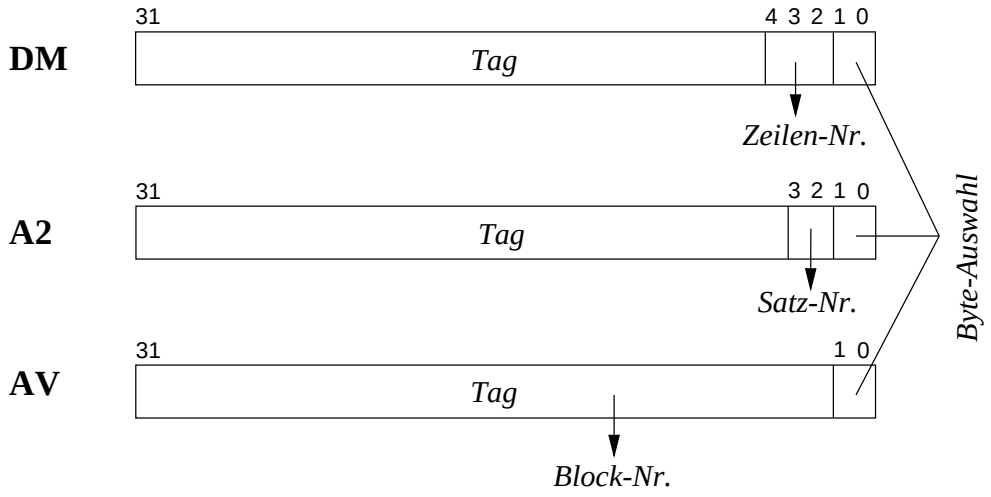
   Die Anzahl der NOP-Befehle nach S4 und S5 (drei oder vier) hängt davon ab, ob bei einem erfolgten Sprung der Befehlszähler in der vierten oder fünften Pipeline-Stufe geschrieben wird.

4. Konflikte, die durch Result-Forwarding vermieden werden können:

   - Der Konflikt S1-S2 (δ^t) wird durch Result Forwarding verhindert, indem das Ergebnis von Stufe 4 der Pipeline nach Ausführung von S1 direkt zum Eingang der Stufe 4 für die Ausführung von S2 zurückgeleitet wird.
   - Der Konflikt S3-S4 (δ^t) wird durch Result Forwarding verhindert, indem das Ergebnis von Stufe 4 der Pipeline nach Ausführung von S3 direkt zum Eingang der Stufe 4 für die Ausführung von S4 zurückgeleitet wird.
   - Der Pipeline-Konflikt aus der Steuerflussabhängigkeit kann durch Result-Forwarding nicht verhindert werden.

Aufgabe 9

1. Unterteilung der Hauptspeicheradresse:



2. Anzahl der Vergleicher:

Cache	Anzahl der Vergleicher
DM	1
A2	2
AV	8

3. »×« für Cache-Hit und »-« für Cache-Miss:

Adresse:	0x0B	0x2B	0x07	0x0C	0x1E	0x0A	0x1A	0x05	0x04	0x29
DM	-	-	-	-	-	-	-	×	×	-
A2	-	-	-	-	-	×	-	×	×	-
AV	-	-	-	-	-	×	-	×	×	×

Aufgabe 10

1.

Speicher-Bausteine	*richtig*	*falsch*
SRAM wird vorwiegend für schnelle Zwischenspeicher wie Register und Caches eingesetzt.	×	
Eine DRAM-Speicherzelle besteht aus zwei rückgekoppelten Invertern.		×
SDRAM arbeitet synchron zum Systemtakt und Datenpakete werden sowohl bei steigender als auch bei fallender Taktflanke übertragen.		×
32-Bit-Prozessoren mit 32-Bit-Adressbus können bei byteweiser Adressierung maximal 4 GByte Hauptspeicher verwenden.	×	

2.

Virtuelle Speicherverwaltung	*richtig*	*falsch*
Ein Nachteil einer virtuellen Speicherverwaltung ist, dass Programme dann nicht mehr im Hauptspeicher verschoben werden können.		×
Die Übersetzung von virtuellen zu physikalischen Adressen ist einfach und wird durch das Betriebssystem vorgenommen.		×
Bei einem virtuellen Cache werden die niederwertigen Bits der logischen Adresse als Tag verwendet.		×
Segmente sind in der Regel deutlich größer als Seiten.	×	
Es gibt vier Schutzebenen für Speicherzugriffe bei der segmentbasierten Speicherverwaltung der x86-Architektur.	×	

3.

Systembusse	*richtig*	*falsch*
Bus Snooping in Mehrkernprozessoren ist nur bei Schreibzugriffen notwendig.		×
Ein semi-synchroner Systembus arbeitet synchron zum Systemtakt.	×	
Die langsamste Komponente an einem synchronen Systembus bestimmt den Systemtakt.	×	
Bei einem asynchronen Bus werden Daten ohne Verwendung eines Taktsignals übertragen.	×	
Ein semi-synchroner Systembus kann nur mit DRAM verwendet werden.		×