Karlsruher Institut für Technologie

Fakultät für Informatik
Prof. Dr. Tamim Asfour, Prof. Dr. Wolfgang Karl

Aufgabenblätter zur Prüfung

Digitaltechnik und Entwurfsverfahren & Rechnerorganisation

und

Technische Informatik I/II

am 28. Februar 2014, 14:00 – 16:00 Uhr

- Beschriften Sie bitte gleich zu Beginn jedes Lösungsblatt deutlich lesbar mit Ihrem Namen und Ihrer Matrikelnummer.
- Diese Aufgabenblätter werden nicht abgegeben. Tragen Sie Ihre Lösung deshalb ausschließlich in die für jede Aufgabe vorgesehenen Bereiche der Lösungsblätter ein. Lösungen auf separat abgegebenen Blättern werden nicht gewertet.
- Außer Schreibmaterial sind während der Klausur keine Hilfsmittel zugelassen. Täuschungsversuche durch Verwendung unzulässiger Hilfsmittel führen unmittelbar zum Ausschluss von der Klausur und zur Note „nicht bestanden“.
- Soweit in der Aufgabenstellung nichts anderes angegeben ist, tragen Sie in die Lösungsblätter bitte nur die Endergebnisse ein. Die Rückseiten der Aufgabenblätter können Sie als Konzeptpapier verwenden. Weiteres Konzeptpapier können Sie auf Anfrage während der Klausur erhalten.
- Halten Sie Begründungen oder Erklärungen bitte so kurz wie möglich. (Der auf den Lösungsblättern für eine Aufgabe vorgesehene Platz steht übrigens in keinem Zusammenhang mit dem Umfang einer korrekten Lösung!)
- Die Gesamtpunktzahl beträgt 90 Punkte. Zum Bestehen der Klausur sind mindestens 40 Punkte zu erreichen.

Viel Erfolg und viel Glück!

Aufgabe 1 *Schaltfunktionen* (10 Punkte)

1. Bestimmen Sie die konjunktive Normalform (KNF) der Funktion $f(w,x,y,z)$ 2 P.

$$f(w,x,y,z) = \overline{y}(x\overline{z} \vee z) \vee w\overline{x}(y \vee \overline{z}) \vee \overline{x}yz$$

2. Bestimmen Sie *algebraisch* die disjunktive Minimalform (DMF) der Funktion $g(c,b,a)$, die durch das Schaltnetz in Abblidung 1 realisiert ist. 3 P.

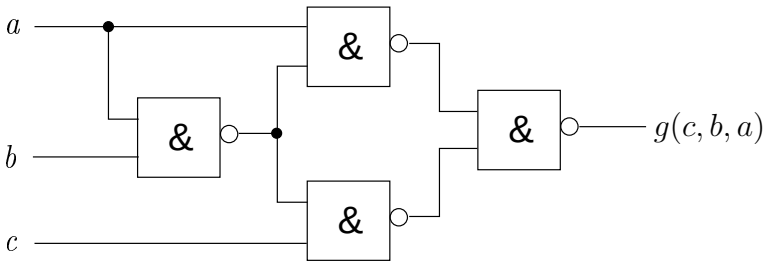


Abbildung 1: Schaltnetz der Funktion $g(c,b,a)$

Gegeben sei die Überdeckungstabelle einer weiteren Funktion z mit den Nullstellen a, b, c, d und e sowie allen Primimplikaten A, B, C, D und E der Funktion z (Siehe Tabelle 1). Die Realisierungskosten eines Primimplikaten sind umgekehrt proportional zur Anzahl der von ihm überdeckten Nullstellen.

Prim-implikate	Nullstellen a	b	c	d	e
A		×	×		
B	×				×
C	×			×	
D	×	×			×
E		×	×		×

Tabelle 1: Überdeckungstabelle der Funktion z

3. Vereinfachen Sie die Tabelle, indem Sie die Kern-Primimplikate ermitteln und die davon überdeckten Nullstellen in Tabelle 1 streichen. 1 P.

   Zeichnen Sie die so reduzierte Tabelle.

4. Vereinfachen Sie Ihre Tabelle aus Aufgabenteil 3 mit den Regeln der Spaltendominanz. 1 P.

   Zeichnen Sie die so reduzierte Tabelle.

5. Ergänzen Sie Ihre Tabelle im Aufgabenteil 4 um eine Kostenspalte und vereinfachen Sie die Tabelle dann mit den Regeln der Zeilendominanz. 2 P.

   Zeichnen Sie die so reduzierte Tabelle.

6. Geben Sie die minimale Form der Funktion z an. 1 P.

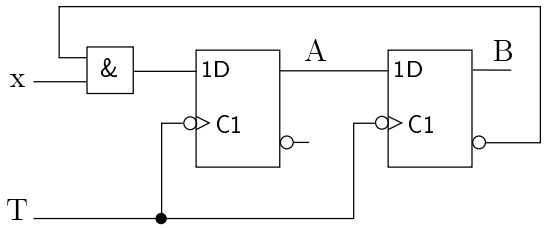
Aufgabe 2 *Spezielle Bausteine* (8 Punkte)

Analysieren Sie die in Abbildung 2 dargestellten Schaltwerke und geben Sie an, ob sie 8 P.

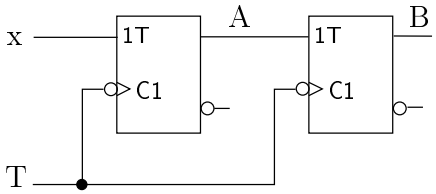
- vorwärts zählen
- rückwärts zählen
- synchron sind
- bei *jedem* Zählerstand mit Hilfe der Eingangsvariablen x angehalten werden können.

Verwenden Sie die im Lösungsblatt vorbereitete Tabelle und tragen Sie ein X an der richtigen Stelle ein. Für jede falsche Antwort wird ein halber Punkt abgezogen.

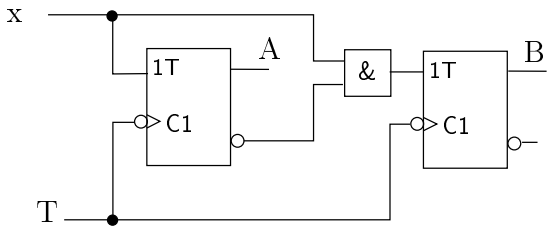
Hinweis: A stellt das niedrigstwertige Bit (LSB) und B das höchstwertige Bit (MSB) dar.



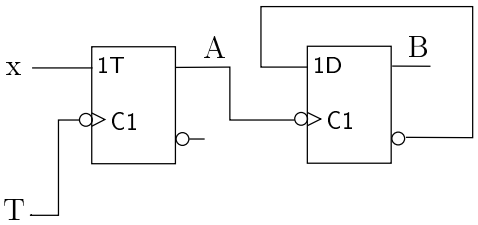
Schaltwerk 1



Schaltwerk 3



Schaltwerk 2



Schaltwerk 4

Abbildung 2: Schaltwerke

Aufgabe 3 *CMOS-Technologie* (7 Punkte)

1. Realisieren Sie die folgende Schaltfunktion y als CMOS-Schaltnetz [5 P.]

$$y = a\,b \;\vee\; c\,d \;\vee\; e\,f\,g$$

2. Welche Schaltfunktion wird durch das CMOS-Schaltnetz realisiert, dessen n-MOS-Netz in Abbildung 3 näher beschrieben ist? [2 P.]

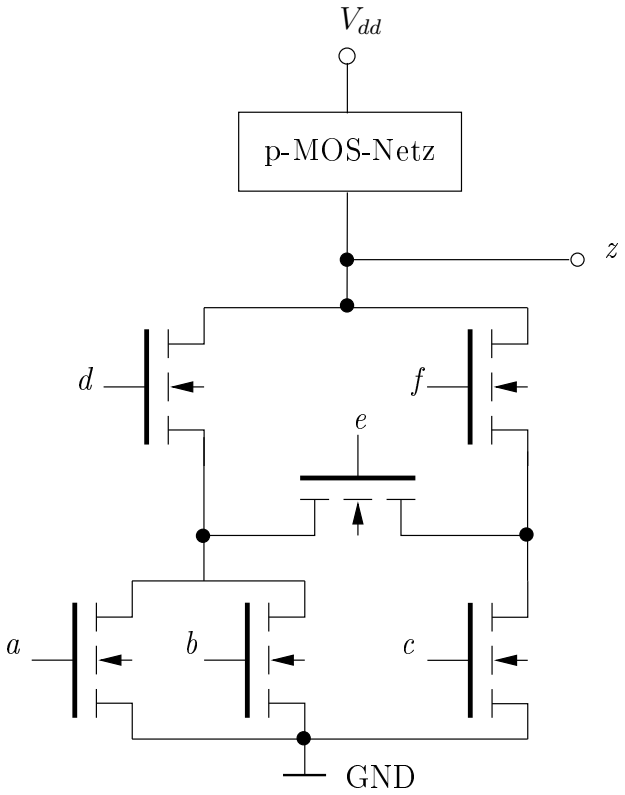


Abbildung 3: CMOS-Schaltnetz

Aufgabe 4 *Schaltwerke* (10 Punkte)

1. Gegeben ist das in Abbildung 4 dargestellte Schaltwerk.

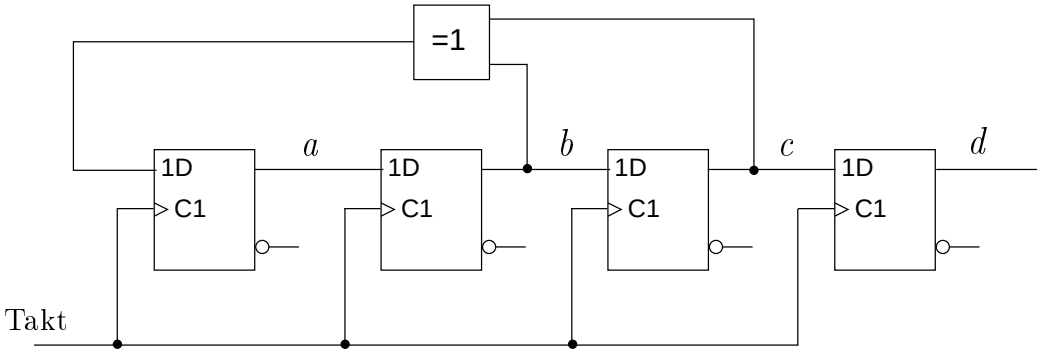


Abbildung 4: Schaltwerk

   (a) Ist das Schaltwerk synchron oder asynchron? 1 P.

   (b) Wie viele Zustände kann das Schaltwerk maximal annehmen? 1 P.

   (c) Vervollständigen Sie die Verläufe der Signale a, b, c und d im Lösungsblatt. 4 P.

2. Es soll ein synchroner modulo-8-Rückwärtszähler mit flankengesteuerten T-Flipflops entworfen werden.

   (a) Geben Sie den Automatengraphen des Zählers an. 2 P.

   (b) Stellen Sie die kodierte Ablauftabelle des Zählers auf. Verwenden Sie hierzu die im Lösungsblatt vorbereitete Tabelle. Die Zustände des Zählers seien mit Hilfe der Zustandsvariablen q_2, q_1 und q_0 dual kodiert. 1 P.

   (c) Geben Sie die Ansteuerfunktionen der verwendeten Flipflops in minimaler Form an. 1 P.

Aufgabe 5 *Rechnerarithmetik* (10 Punkte)

Hinweis: Geben Sie in dieser Aufgabe *immer* den Rechenweg an.

1. Geben Sie die Gleitkommadarstellung der folgenden Zahlen im IEEE-754 Standard in einfacher Genauigkeit an. [2 P.]

   (a) $29,375_{10}$

   (b) $-0,75_{10}$

2. Was repräsentieren die folgenden Bitmuster, wenn man sie als Zweierkomplement und als Gleitkommazahl im IEEE-754 Standard in einfacher Genauigkeit interpretiert? [2 P.]

   (a) `0011 1111 1100 1000 0000 0000 0000 0000`

   (b) `1001 0000 0110 1000 0000 0000 0000 0000`

   Hinweis: Sie brauchen die Zweier-Potenzen nicht explizit auszurechnen.

3. Warum wird zur Darstellung der Zahl `0` als normalisierte Zahl im IEEE-754 Standard eine Ausnahmeregel benötigt? [1 P.]

4. Der Hauptspeicher eines Rechners mit der Datenwortbreite von 8 Bit unterstützt eine Einzelfehler-Korrektur mit Hilfe des Hamming-Codes. Aus dem Speicher erhält man das Codewort `1 1 0 1 0 1 1 1 0 1 1 1`. [2 P.]

   Prüfen Sie das Codewort auf Einzelbit-Fehler, die beim Übertragen oder Speichern entstanden sein könnten und korrigieren Sie diese gegebenenfalls. Geben Sie das zugehörige Datenwort an.

5. Die PPS-Methode (*partial product sum*) zur seriellen Multiplikation zweier n-Bit Zahlen kann mit Hilfe eines n-Bit Addierers, eines n-Bit Registers B, zweier n-Bit Schieberegister A und P und eines Flipflops E implementiert werden. [3 P.]

   Skizzieren Sie den Aufbau eines solchen seriellen Multiplizierers. Aus Ihrer Zeichnung soll der Datenfluss erkennbar sein.

Aufgabe 6 *Allgemeines* (6 Punkte)

1. Was versteht man unter einem *von-Neumann-Flaschenhals*? [1 P.]

2. Das Statusregister enthält Bits (*Flags*), die das Ergebnis einer arithmetischen Operation widerspiegeln. Nennen Sie 4 verschiedene Bits und erklären Sie Ihre Bedeutung. [2 P.]

3. Skizzieren Sie den Aufbau einer statischen CMOS-Speicherzelle. Aus Ihrer Zeichnung muss die Ankoppelung der Speicherzelle an die Daten- und Adressleitungen erkennbar sein. [3 P.]

Aufgabe 7 *MIPS-Assembler* (12 Punkte)

1. Geben Sie für das folgende MIPS-Programmstück den Inhalt des Zielregisters in hexadezimaler Schreibweise nach der Ausführung des jeweiligen Befehls an. 3 P.

```
andi  $s1, $zero, 40
sra   $s2, $s1, 2
slt   $s3, $zero, $s2
sub   $s4, $s3, $s2
xori  $s5, $s4, -1
```

2. Gegeben sei das folgende MIPS-Programmstück:

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

        .text
main:   lw $t1, vec
        lw $t2, vec+4
        lw $t3, vec($t1)
        lw $t4, vec+4($t1)
```

   (a) Geben Sie die Inhalte der Register `$t1`, `$t2`, `$t3` und `$t4` in hexadezimaler Schreibweise nach der Ausführung des obigen Programmcodes an. 2 P.

   (b) Geben Sie den MIPS-Code an, mit dem man die Adresse von `vec` im Register `$s0` speichert. 1 P.

   (c) Schreiben Sie eine Programmschleife, welche für jedes Element aus `vec` die Summe aller vorhergehenden Elemente ausgibt. Die ausgegebenen Zahlen sollen dabei durch ein Leerzeichen getrennt sein. 4 P.

   Hinweis: In Tabelle 2 sind die Nummern und die Argumente der benötigten Systemaufrufe angegeben.

Funktion	Systemaufruf	Argumente
`print_int`	1	Integer in `$a0`
`print_string`	4	Adresse einer mit Null terminierten Zeichenkette in `$a0`

Tabelle 2: Ausgabe-Systemaufrufe *(system calls)*

3. Was ist ein Pseudobefehl? Und was ist eine Assemblerdirektive? 1 P.

4. Wie ist die Trennung von Programmen und Daten bei der Ihnen bekannten MIPS-R2000-Architektur realisiert? 1 P.

Aufgabe 8 *Pipelining* (12 Punkte)

Das folgende Programmstück soll in der DLX-Pipeline abgearbeitet werden.

```
S1:  and   $t1, $t2, $t5
S2:  add   $t3, $t1, $t4
S3:  sll   $t4, $t3, 1
S4:  sub   $t1, $0,  $t1
S5:  add   $t2, $t3, $t4
```

1. Bestimmen Sie alle Datenabhängigkeiten innerhalb dieses Programmstücks. 3 P.

2. Zu Beginn des Programmstücks seien die Register folgendermaßen belegt: 3 P.

`$t1`	`$t2`	`$t3`	`$t4`	`$t5`
4	8	0	-2	15

   Geben Sie die Registerbelegung nach Ablauf des Programmstücks an. Tragen Sie hierzu in die Tabelle auf dem Lösungsblatt den Zustand der Pipeline und der Register (am Ende des Taktes) für jeden Takt ein.

   Wie viele Takte werden benötigt, um das Programm abzuarbeiten?

3. Wie wäre die Belegung der Register, wenn der Prozessor keine Pipeline besäße und die Befehle rein sequentiell abarbeitete? 1 P.

4. Die einzige Methode, die Pipelinekonflikte bei diesem Prozessor zu beheben, sei das Einfügen von `NOP`-Befehlen (*No Operation*) in den Befehlsstrom. 2 P.

   Fügen Sie möglichst wenige `NOP`-Befehle in das Programmstück ein, so dass es zu keinen Konflikten mehr kommt, und das Ergebnis dem der sequentiellen Ausführung entspricht. Geben Sie das modifizierte Programmstück an.

   Wie viele Takte werden nun benötigt?

5. Warum stellen bedingte Sprünge ein Problem für die Pipelineimplementierung dar? Geben Sie zwei Möglichkeiten zur Lösung dieses Problems an. 2 P.

6. Ist es möglich, einen Befehlssatz mit Befehlen variabler Länge in der DLX-Pipeline auszuführen? Begründen Sie Ihre Antwort. 1 P.

Aufgabe 9 *Cache-Speicher* (11 Punkte)

Ein 32-Bit-Prozessor besitzt einen Cache-Speicher mit einer Speicherkapazität von 128 Byte und einer Blockgröße von 16 Byte (4 Wörter). Es werden einzelne Bytes adressiert. Der Cache-Speicher ist als 2-fach assoziativer Cache *(2-way set associative cache)* organisiert. Die Hauptspeicheradresse ist 32 Bit breit. Zur Verwaltung eines Cacheblocks werden zwei zusätzliche Statusbits (*Valid*-Bit: V und *Dirty*-Bit: D) verwendet. Falls notwendig, wird die „Least Recently Used“-Ersetzungsstrategie (LRU) verwendet.

1. Welche Bits der 32-Bit-Adresse bilden Byte-Offset, Wort-Offset, Index und Tag? Skizzieren Sie die Unterteilung der Hauptspeicheradresse für diesen Cache-Speicher. 2 P.

2. Ermitteln Sie den insgesamt erforderlichen Speicherbedarf zur Realisierung dieses Cache-Speichers. 2 P.

3. Nehmen Sie an, der Cache-Speicher sei zu Beginn leer und betrachten Sie die Folge der Lesezugriffe auf die folgenden Hauptspeicheradressen: 6 P.

   ```
   0x743, 0x2A1, 0x183, 0x2A9, 0x06F, 0x69C,
   0x2A3, 0x26A, 0x18C, 0x3C7, 0x2A8, 0x475
   ```

   Kennzeichnen Sie in der vorbereiteten Tabelle im Lösungsblatt, ob es sich beim Lesezugriff auf die jeweiligen Adressen um einen Treffer (Cache-Hit) oder um keinen Treffer (Cache-Miss) handelt. Verwenden Sie dabei „**M**“ für Cache-Miss und „**H**“ für Cache-Hit.

4. Wozu wird das *Dirty*-Bit in einem Cache-Speicher verwendet? 1 P.

Aufgabe 10 *Multiple Choice* (4 Punkte)

Kreuzen Sie bitte für jede der Behauptungen im Lösungsblatt an, ob sie Ihrer Meinung nach richtig oder falsch ist. Nicht angekreuzte Behauptungen zählen nicht und gehen somit nicht in die Bewertung ein. Zur Ermittlung der Punktzahl werden von den richtig angekreuzten Behauptungen die falsch angekreuzten Behauptungen abgezogen; ein negativer Übertrag in andere Teilaufgaben erfolgt nicht.

Karlsruher Institut für Technologie

Fakultät für Informatik
Prof. Dr. Tamim Asfour, Prof. Dr. Wolfgang Karl

Lösungsblätter zur Klausur

Digitaltechnik und Entwurfsverfahren & Rechnerorganisation

und

Technische Informatik I/II

am 28. Februar 2014, 14:00 – 16:00 Uhr

Name:	Vorname:	Matrikelnummer:

Digitaltechnik und Entwurfsverfahren/TI-1	
Aufgabe 1	von 10 Punkten
Aufgabe 2	von 8 Punkten
Aufgabe 3	von 7 Punkten
Aufgabe 4	von 10 Punkten
Aufgabe 5	von 10 Punkten
Rechnerorganisation/TI-2	
Aufgabe 6	von 6 Punkten
Aufgabe 7	von 12 Punkten
Aufgabe 8	von 12 Punkten
Aufgabe 9	von 11 Punkten
Aufgabe 10	von 4 Punkten

Gesamtpunktzahl:	

	Note:

Aufgabe 1

1. KNF von $f(w, x, y, z)$:

2. DMF von $g(c, b, a)$:

3. Kern-Primimplikate:

   Reduzierte Tabelle:

4. Dominierte Maxterme:

   Reduzierte Tabelle:

5. Dominierende Primimplikate:

   Reduzierte Tabelle:

6. Minimalform der Funktion z:

Aufgabe 2

Schaltwerk	**1**	**2**	**3**	**4**
zählt vorwärts				
zählt rückwärts				
ist synchron				
kann bei *jedem* Zählerstand mit Hilfe von x angehalten werden.				

Aufgabe 3

1. CMOS-Schaltnetz:

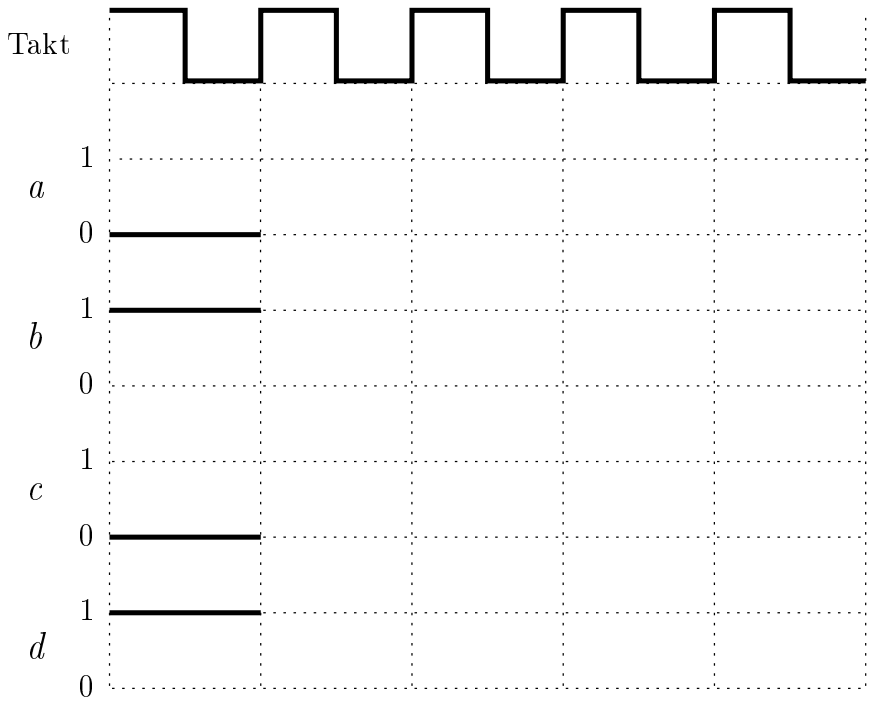
2. Realisierte Schaltfunktion:

Aufgabe 4

1. (a) Das Schaltwerk ist

   (b) Maximale Anzahl der Zustände ist:

   (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}	
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

(c) Minimalformen der Ansteuerfunktionen der Flipflops:

Aufgabe 5

1. Gleitkommadarstellung von

   (a) $29,375_{10}$

   (b) $-0,75_{10}$

2. (a) `0011 1111 1000 0000 0000 0000 0000 0000` als

   - Zweierkomplement:

   - Gleitkommazahl (IEEE-754):

(b) `1001 0000 0110 1000 0000 0000 0000 0000` als

- Zweierkomplement:

- Gleitkommazahl (IEEE-754):

3. Ausnahmeregel für die Null im IEEE-754-Standard:

4. Datenwort:

5. Serieller Multiplizierer nach der PPS-Methode:

Aufgabe 6

1. *von-Neumann-Flaschenhals:*

2. Bits im Statusregister:
   -
   -
   -
   -

3. Statische CMOS-Speicherzelle:

Aufgabe 7

1. Inhalte der Zielregister:

Befehl	Zielregister = (z. B. `$s6 = 0x0000 F00A`)
`andi $s1, $zero, 40`	
`sra  $s2, $s1, 2`	
`slt  $s3, $zero, $s2`	
`sub  $s4, $s3, $s2`	
`xori $s5, $s4, -1`	

2. (a) Registerinhalte:

Register	Inhalt
`$t1`	
`$t2`	
`$t3`	
`$t4`	

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

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

3. Pseudobefehl:

   Assemblerdirektive:

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

Aufgabe 8

1. Datenabhängigkeiten:

2. Belegung der Register nach Ablauf des Programms und Zustand der Pipeline:

Takt	IF	ID/RF	EX	MEM	WB	$t1	$t2	$t3	$t4	$t5
1	S1	—	—	—	—	4	8	0	-2	15

   Anzahl der Takte:

3. Belegung der Register bei sequentieller Bearbeitung des Programms:

$t1	$t2	$t3	$t4	$t5

4. Behebung der Pipelinekonflikte durch Einfügen von NOP-Befehlen:

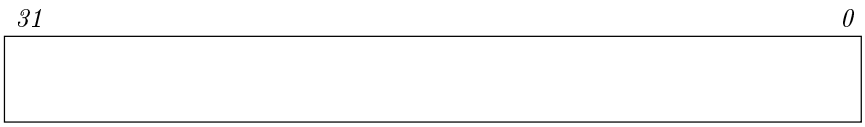
   Anzahl der Takte:

5. Bedingte Sprünge (Problem und zwei Lösungsmöglichkeiten):

6. Befehlssatz mit Befehlen variabler Länge in der DLX-Pipeline:

Aufgabe 9

1. Unterteilung der Hauptspeicheradresse:



2. Speicherbedarf:

3. Cache-Miss (**M**) oder Cache-Hit(**H**):

Adresse	Hit/Miss
0x743	
0x2A1	
0x183	
0x2A9	
0x06F	
0x69C	
0x2A3	
0x26A	
0x18C	
0x3C7	
0x2A8	
0x475	

4. *Dirty*-Bit:

Aufgabe 10

1.

Speicher-Bausteine	*richtig*	*falsch*
Der Hauptspeicher heutiger Rechner wird aus statischen RAM-Bausteinen realisiert.		
Statische RAM-Bausteine lassen sich schlechter als dynamische RAM-Bausteine integrieren; sie besitzen jedoch eine kürzere Zugriffszeit.		
Statische RAM-Bausteine müssen ständig „refresht“ werden.		
Statische RAM-Bausteine speichern die Information in Kondensatoren.		

2.

Programmiersprache C	*richtig*	*falsch*
Globale Variablen werden als Variablen der Speicherklasse *static* definiert.		
Ein *Pointer* ist ein Zeiger auf einen Speicherbereich.		
In C sind rekursive Funktionsaufrufe nicht zulässig.		
Eine Variable, auf die schnell zugegriffen werden muss, wird als eine Variable der Speicherklasse *register* definiert.		

3.

Adressierung	richtig	falsch
Bei einem von-Neumann-Rechner kann anhand der Adresse eines Datums erkannt werden, ob es sich bei diesem Datum um einen Operanden oder um einen Befehl handelt.		
Beim Little-Endian-Datenformat ist das niedrigstwertige Byte eines Wortes an seiner niedrigsten Adresse.		
Bei einer speicherbezogenen Adressierung *(memory mapping)* von Peripherie-Bausteinen existieren zwei getrennte Adreßräume für Speicher und Peripherie.		
Durch das Multiplexen bestimmter Signalgruppen kann die Anzahl der Anschlüsse am Prozessorgehäuse verringert werden.		

4.

Virtuelle Speicherverwaltung	richtig	falsch
Externe Fragmentierung ist ein typisches Problem der Segmentierung.		
Das Betriebsystem ist allein zuständig für die Umsetzung virtueller in physikalische Adressen.		
Seitengrößen können je nach Anforderung dynamisch wachsen.		
Die Umsetzung virtueller in physikalische Adressen wird durch die MMU beschleunigt.		