KIT-Fakultät für Informatik
Prof. Dr. Mehdi Tahoori, Prof. Dr. Wolfgang Karl

Aufgabenblätter zur Prüfung

Digitaltechnik und Entwurfsverfahren (TI-1)

und

Rechnerorganisation (TI-2)

am 25. Februar 2019, 13:30 – 15:30 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 Endergebnisse und Rechenweg 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 so kurz und präzise wie möglich. Der auf den Lösungsblättern für eine Aufgabe vorgesehene Platz lässt nicht auf den Umfang einer korrekten Lösung schließen.
- Die Gesamtpunktzahl beträgt 90 Punkte. Zum Bestehen der Klausur sind mindestens 40 Punkte zu erreichen.

Viel Erfolg und viel Glück!

Aufgabe 1 *Schaltfunktionen* (11 Punkte)

Eine unvollständig definierte Schaltfunktion $y = f(d, c, b, a)$ sei durch ihre Eins- und *don't care*-Stellen (Abkürzung d) gegeben:

$$y = \text{MINt}(0, 1, 7, 8, 15) \ \vee \ \text{d}(4, 9)$$

1. Tragen Sie alle Primimplikanten der Funktion ins KV-Diagramm im Lösungsblatt ein und geben Sie eine disjunktive Minimalform (DMF) der Funktion f an. [2 P.]

2. Tragen Sie alle Primimplikate der Funktion ins KV-Diagramm im Lösungsblatt ein und geben Sie eine konjunktive Minimalform (KMF) der Funktion f an. [3 P.]

Gegeben sei eine Schaltfunktion $z = g(d, c, b, a)$, von der man weiß, dass $\overline{c}\ b\ \overline{a}$ und $\overline{d}\ \overline{c}\ \overline{a}$ *Kernprimimplikanten* dieser Funktion sind.

3. Welche der im Lösungsblatt angegebenen Produktterme können definitiv **keine** Primimplikanten der Funktion z sein? Tragen Sie in diesem Fall ein **X** in der Tabelle im Lösungsblatt. Geben Sie jeweils eine Begründung Ihrer Antwort an. (Keine Punkte bei fehlender Begründung) [2 P.]

Die Funktionen $f_1,\ f_2,\ f_3$ und f_4 sollen mit Hilfe eines PLA-Bausteins realisiert werden.

$$\begin{aligned} f_1(c, b, a) &= \text{MINt}(3, 6, 7) \\ f_2(c, b, a) &= \text{MINt}(0, 1, 4, 5, 6) \\ f_3(c, b, a) &= \text{MINt}(2, 3, 4) \\ f_4(c, b, a) &= \text{MINt}(2, 3, 4, 7) \end{aligned}$$

4. Personalisieren Sie den im Lösungsblatt angegebenen PLA-Baustein, indem Sie geeignete Leitungskreuzungen der UND- und der ODER-Matrix markieren. [4 P.]

Aufgabe 2 *Minimierungsverfahren* (12 Punkte)

1. Bestimmen Sie die konjunktive Minimalform der durch das Multiplexer-Schaltnetz in Abbildung 1 realisierten Schaltfunktion $y = f(c, b, a)$. 3 P.

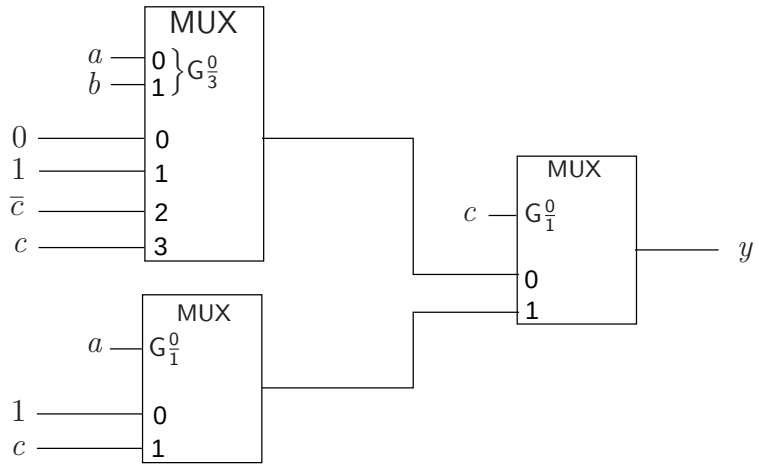


Abbildung 1: Multiplexer-Schaltnetz der Schaltfunktion $y = f(c, b, a)$

2. Bestimmen Sie alle Primimplikanten der Schaltfunktion 4 P.

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

mit Hilfe des Consensus-Verfahrens. Die prinzipielle Vorgehensweise bei der Anwendung dieses Verfahrens soll aus der Lösung ersichtlich sein. Verwenden Sie hierzu die im Lösungsblatt vorbereitete Tabelle. Geben Sie anschließend die Primimplikanten an.

3. Gegeben sei die Überdeckungstabelle einer Schaltfunktion $h(d, c, b, a)$ mit den Mintermen 4, 5, 6, 8, 9, 10, 13. Die Primimplikanten der Funktion seien A, B, C, D, E, F und G.

	4	5	6	8	9	10	13
A	×						
B				×			
C				×	×		
D				×		×	
E					×		×
F	×	×	×				
G		×					×

(a) Ist die Schaltfunktion $h(d, c, b, a)$ vollständig oder unvollständig definiert? Begründen Sie Ihre Antwort (Keine Punkte bei fehlender Begründung). 2 P.

(b) Im folgenden sei:

A	=	0	–	0	0	B	=	–	0	0	0	C	=	1	0	0	–
D	=	1	0	–	0	E	=	1	–	0	1	F	=	0	1	–	–
G	=	–	1	–	1												

Bestimmen Sie die disjunktive Minimalform (DMF) der Schaltfunktion $h(d, c, b, a)$. Beschreiben Sie Ihre Vorgehensweise. 3 P.

Aufgabe 3 *Spezielle Bausteine* (11 Punkte)

1. Die Schaltfunktion 4 P.

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

soll in der CMOS-Technologie realisiert werden. Es stehen Ihnen ein NOR-Gatter, ein NAND-Gatter und ein Inverter-Gatter zur Verfügung. Geben Sie das Transistor-Schaltbild an.

2. Für die Fehlererkennung in einem 4-Bit-Code wird ein Schaltnetz benötigt, welches die anliegenden Eingangsvariablen auf ungerade Parität überprüft, d.h die Funktion 4 P.

$$p = \mathrm{odd}(w, x, y, z) = w \oplus x \oplus y \oplus z$$

realisiert. Dabei bezeichnet $\oplus$ den Quersummen-Operator.

Realisieren Sie das Schaltnetz unter ausschließlicher Verwendung eines 8:1-Multiplexers und eines Inverters. Zeichnen Sie die Schaltung.

3. Entwerfen Sie ein 3-Bit Schieberegister aus taktflankengesteuerten D-Flipflops. Das Schieberegister soll asynchron rücksetzbar sein. 3 P.

Geben Sie die Schaltung des Schieberegisters an und kennzeichnen Sie die Daten- und Steuerleitungen.

Aufgabe 4 *Laufzeiteffekte* (6 Punkte)

Gegeben ist das in Abbildung 2 dargestellte Schaltnetz. NAND-, NOR- und OR-Gatter haben eine Totzeit von $5\,ns$, das XOR-Gatter von $7\,ns$ und der Inverter von $2\,ns$. Die Eingangsvariablen a, b und c wechseln zum Zeitpunkt $t = 0$ *gleichzeitig* von 0 auf 1.

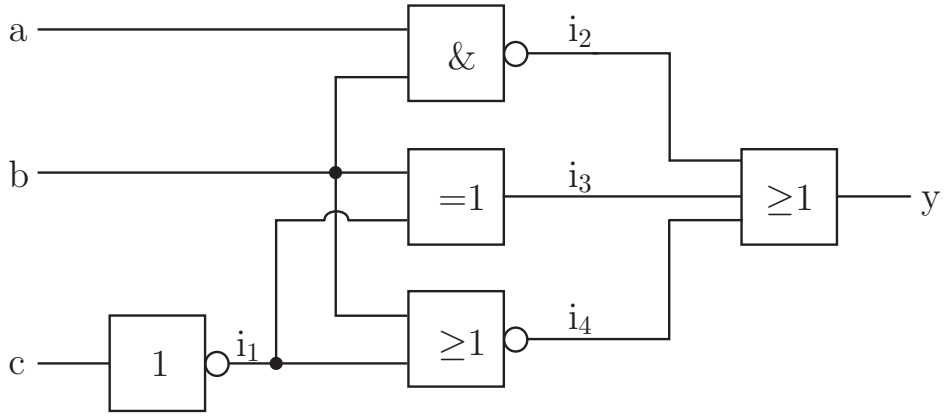


Abbildung 2: Schaltnetz

1. Geben Sie die Verläufe der Signale i_1, i_2, i_3, i_4 und y an, indem Sie das im Lösungsblatt angegebene Zeitdiagramm vervollständigen. 3 P.

2. Treten im Zeitverlauf Hasardfehler auf? Falls ja, um welchen Typ handelt es sich bei dem zu Grunde liegenden Hasard? Begründen Sie Ihre Antwort. 3 P.

Aufgabe 5 *Schaltwerke* (5 Punkte)

Es wurde ein Viren-Scanner als synchrones Schaltwerk entworfen, welcher einen binären Eingabestrom (Variable x) auf das Bitmuster *(virus signature)* 10-1 überprüft. Dabei steht - für eine 0 oder eine 1.
Beim Erkennen eines derartigen Musters wird eine 1 (Variable y) im nächsten Taktzyklus ausgegeben. Deshalb wurde das Schaltwerk als Moore-Automat realisiert. Ein Beispiel einer Eingabe-Ausgabe-Folge sieht folgendermaßen aus:

t:	1	2	3	4	5	6	7	8	9	10	11	...
x(t):	0	0	1	0	1	1	0	0	1	1	0	...
y(t):	0	0	0	0	0	0	1	0	0	1	0	...

1. Geben Sie den Automatengraphen des Schaltwerks mit minimaler Anzahl an Zuständen an. Bezeichnen Sie den Anfangszustand mit S und die restlichen Zustände mit $A, B, C, \ldots$ usw. Vergessen Sie nicht, die Kanten und Knoten Ihres Graphen zu beschriften. 5 P.

Aufgabe 6 *Mikroprozessor* (6 Punkte)

In Abbildung 3 ist der prinzipielle Aufbau eines Mikroprozessors mit dem internen Steuerbus dargestellt.

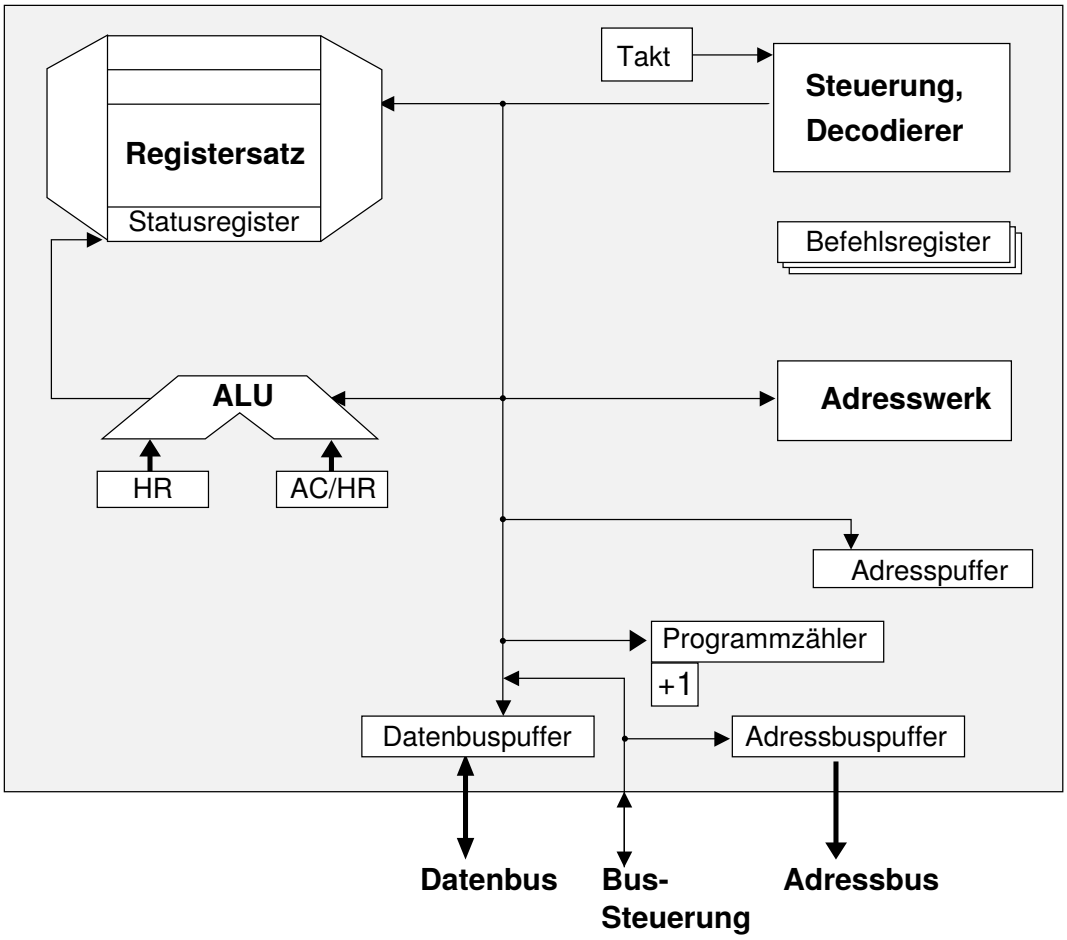


Abbildung 3: Aufbau eines Mikroprozessors und sein interner Steuerbus

Ihre Aufgabe besteht darin, die Architektur des internen Daten-Bussystems zu entwerfen, so dass eine hohe prozessorinterne Parallelität bei der Befehlsbearbeitung möglich ist, d. h.

- *OpCode Prefetching*.
- Gleichzeitiges Schreiben der ALU-Ergebnisse in den Registersatz und paralleles Laden der ALU-Eingänge mit Operanden aus dem Registersatz oder dem Datenbuspuffer (sofern nicht das gleiche Register hierfür benötigt wird).
- Direkter Zugriff des Adresswerks auf die Adressregister im Registersatz.
- Benötigter Datenaustausch zwischen den restlichen Komponenten bzw. Registern.

Ergänzen Sie die im Lösungsblatt angegebene Abbildung, sodass das interne Bussystem die obigen Anforderungen erfüllt. Die Richtung des Datenflusses muss aus Ihrer Zeichnung deutlich erkennbar sein.

Aufgabe 7 *C, MIPS–Assembler & MIMA* (11 Punkte)

1. Schreiben Sie die MIPS-Programmstücke in C-Sprache um. Verwenden Sie dabei die in den Kommentaren verwendeten Variablennamen. Berücksichtigen Sie dabei nicht die möglichen Pipelinekonflikte. 6 P.

(a)

```
                addi $t0, $zero, 0      # a = 0
                addi $t1, $zero, 0      # b = 0
                addi $t2, $zero, 0      # c = 0

                bge  $s0, 5, marke      # if (i >=5)  goto marke
                addi $t0, $zero, 1      # a = 1
                addi $t1, $zero, 2      # b = 2
                addi $t2, $zero, 3      # c = 3
        marke:  addi $t3, $zero, 5      # d = 5
```

(b)

```
                addi $t0, $zero, 0      # a = 0
                addi $t1, $zero, 0      # b = 0
                addi $t2, $zero, 0      # c = 0

                bge  $s0, 5, marke1     # if (i >=5)  goto marke1
                addi $t0, $zero, 1      # a = 1
                addi $t1, $zero, 2      # b = 2
                addi $t2, $zero, 3      # c = 3
                j    marke2             # goto marke2
        marke1: addi $t0, $zero, 4      # a = 4
                addi $t1, $zero, 5      # b = 5
                addi $t2, $zero, 6      # c = 6
        marke2: addi $t3, $zero, 5      # d = 5
```

(c)

```
                la   $t0, a             # $t0 = &a[0]
                addi $a0, $zero, 0      # sum=0
                addi $a1, $zero, 0      # i=0
                addi $a3, $zero, 100    # $a3 = 100
         loop:  mult $a2, $a1, 4        # $a2=i*4
                addu $t1,$t0,$a2        # $t1=&a[i]
                lw   $t2,0($t1)         # $t2=a[i]
                add  $a0,$a0,$t2        # sum=sum+a[i]
                addi $a1,$a1,1          # i++
                blt  $a1,$a3,loop       # if i<100  goto loop
```

2. Vervollständigen Sie das unten angegebene MIMA-Mikroprogramm für den Befehl `ADD a` (Akku + <a> → Akku) in Register-Transfer-Schreibweise: 5 P.

   1. Takt: IAR → SAR; IAR → X; R = 1
   2. Takt: Eins → Y; R = 1
   3. Takt: ...

   Hinweis: Geben Sie alle nötigen Phasen des Befehls mit an. Auf der letzten Seite der Aufgabenblätter finden Sie das aus der Übung bekannte Beiblatt zur MIMA.

Aufgabe 8 *Pipelining* (10 Punkte)

Gegeben sei eine Pipeline mit den folgenden Stufen:

- 1. Stufe: Befehl holen
- 2. Stufe: Befehl dekodieren und Operanden aus Registern bereitstellen
- 3. Stufe: Befehl ausführen
- 4. Stufe: Zugriff auf Speicher
- 5. Stufe: Ergebnis im Register speichern

Bei Lade/Speicher-Befehle wird in der Stufe 3 die Adresse berechnet und in der Stufe 4 auf den Speicher zugegriffen. Alle anderen Befehlsarten führen in der Stufe 4 keine Operation durch. Das Laden von Operanden in der Stufe 2 und Speichern von Ergebnissen in der Stufe 5 ist erst am Ende des Taktes abgeschlossen.

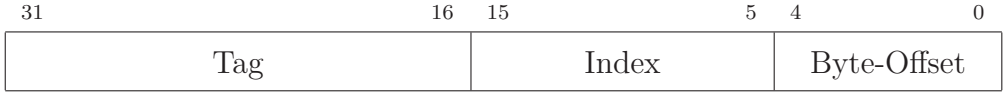
Betrachten Sie die folgende Programmsequenz:

```
S1:   mult    R5, R2, R2  ;    R5 = R2 * R2
S2:   mult    R1, R4, R2  ;    R1 = R4 * R2
S3:   add     R5, R5, R3  ;    R5 = R5 + R3
S4:   add     R1, R1, R5  ;    R1 = R1 + R5
```

1. Bestimmen Sie alle Datenabhängigkeiten in der gegebenen Programmsequenz. 3 P.
2. Welche der gefundenen Abhängigkeiten führen in der obigen Programmsequenz zu Pipelinekonflikten? 3 P.
3. Beseitigen Sie alle Pipelinekonflikte durch Einfügen von möglichst wenigen NOP-Befehlen. 2 P.
4. Welches Problem tritt auf, wenn bei der gegebenen Pipeline ein gemeinsamer Daten- und Befehls-Cache benutzt wird, auf den nur ein Lesezugriff pro Takt möglich ist? 2 P.

Aufgabe 9 *Cache-Speicher* (14 Punkte)

1. Gegeben sei ein 2-fach-satzassoziativer Cache-Speicher (*2-way-set-associative cache*) mit der folgenden Unterteilung der Hauptspeicheradresse



   (a) Wie viele Bytes enthält ein Cache-Block? 1 P.

   (b) Wie groß ist die Kapazität des Cache-Speichers? 1 P.

   (c) Bestimmen Sie den insgesamt erforderlichen Speicherbedarf für die Realisierung des Cache-Speichers? Nehmen Sie dabei an, dass zwei Statusbits (*Valid* und *Dirty*) zur Verwaltung eines Cacheblocks verwendet werden. 2 P.

   (d) Der Prozessor greift auf die Speicheradresse `0x00EF1A34` zu. Mit wie vielen und welchen Zeilen im Cache wird ein Vergleich durchgeführt, um herauszufinden, ob ein Cache-Hit vorliegt? 2 P.

2. Gegeben sei ein direkt abgebildeter Cache *(direct mapped)* mit einer Speicherkapazität von 128 Byte und einer Blockgröße von 16 Bytes. Als Aktualisierungsstrategie wird das Rückschreib-Verfahren *(write back)* verwendet. Die Hauptspeicheradresse ist 32 Bit breit. Zur Verwaltung eines Cacheblocks werden zwei Statusbits verwendet: ein *Valid*-Bit (Abkürzung: V) und ein *Dirty*-Bit (Abkürzung: D). 4 P.

   Der Zustand des Cache-Speichers sei durch Tabelle 1 angegeben. Dabei kennzeichnet $V = 1$ einen gültigen Eintrag im Cache und $D = 1$ einen Eintrag im Cache, der gegenüber seiner Originalkopie verändert wurde.

Cache-Speicher			
Zeile	D-Bit	V-Bit	Tag
0	0	1	1
1	0	1	1
2	0	0	4
3	0	1	5
4	1	1	0
5	0	1	3
6	1	1	0
7	0	0	1

Tabelle 1: Anfangsbelegung des Cache-Speichers

   Betrachten Sie die folgende Sequenz von Lese- und Schreibzugriffen auf die folgenden Hauptspeicheradressen:

Adresse	**0x44**	**0xA0**	**0xC3**	**0x9E**	**0x66**	**0x2D**	**0x6B**	**0x49**
read/write	w	r	w	r	r	w	r	w

Geben Sie an, ob es sich beim Zugriff auf die jeweiligen Adressen um einen Cache-Miss oder einen Cache-Hit handelt. Verwenden Sie dabei „—“ für Cache-Miss und „$\times$“ für Cache-Hit. Geben Sie an, ob der entsprechende Cacheblock in den Hauptspeicher zurückkopiert werden muss (**ja**) oder nicht (**nein**).

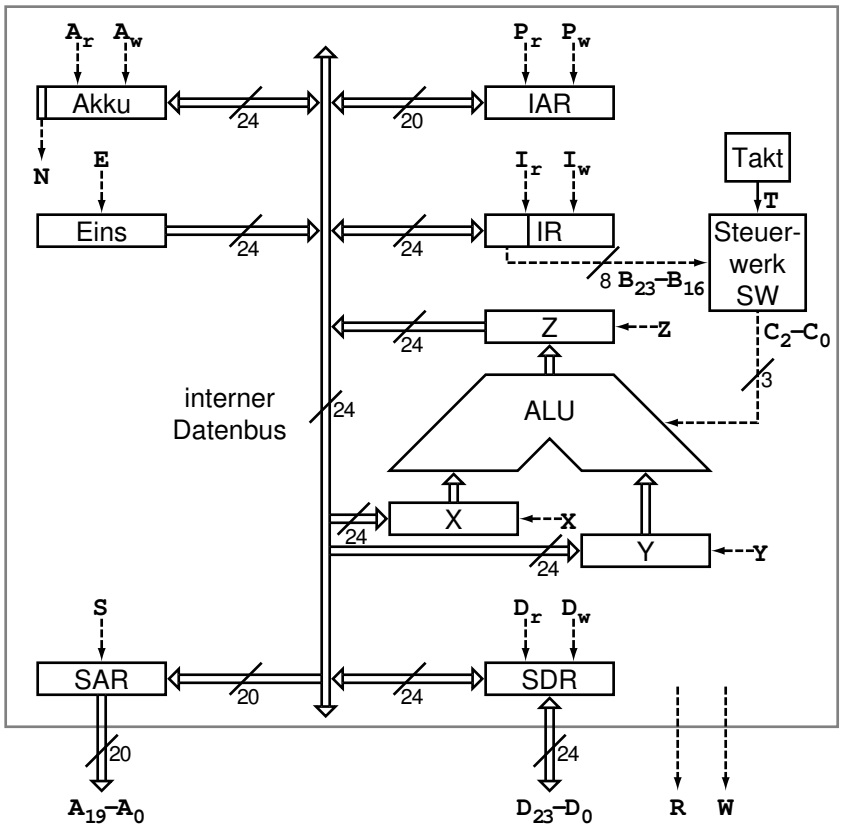
3. Beweisen oder widerlegen Sie folgende Behauptung: 4 P.
   Eine Erhöhung der Assoziativität eines Caches zieht immer eine Verringerung der Miss-Rate nach sich.

   Hinweis: Gehen Sie in ihren Überlegungen davon aus, dass die Caches gemäß Least-Recently-Used-(LRU)-Strategie verdrängen.

Aufgabe 10 *Allgemeines* (4 Punkte)

1. Erklären Sie *arithmetisches Pipelining*? 1 P.
2. Nennen Sie zwei Eigenschaften einer superskalaren Pipeline. 2 P.
3. Was besagt das *Mooresche Gesetz*? 1 P.

Architektur der MIMA



Register

Akku: Akkumulator
X: 1. ALU Operand
Y: 2. ALU Operand
Z: ALU Ergebnis
Eins: Konstante 1
IAR: Instruktionsadreßregister
IR: Instruktionsregister
SAR: Speicheradreßregister
SDR: Speicherdatenregister

Steuersignale vom SW

– für den internen Datenbus

A_r: Akku liest
A_w: Akku schreibt
X: X-Register liest
Y: Y-Register liest
Z: Z-Register schreibt
E: Eins-Register schreibt
P_r: IAR liest
P_w: IAR schreibt
I_r: IR liest
I_w: IR schreibt
D_r: SDR liest
D_w: SDR schreibt
S: SAR liest

– für die ALU

c_2–c_0: Operation auswählen

– für den Speicher

R: Leseanforderung
W: Schreibanforderung

Meldesignale zum SW

T: Takteingang
N: Vorzeichen des Akku
B_{23}–B_{16}: OpCode-Feld im IR

$c_2 c_1 c_0$	*ALU Operation*
0 0 0	tue nichts (d.h. Z -> Z)
0 0 1	X + Y -> Z
0 1 0	rotiere X nach rechts -> Z
0 1 1	X AND Y -> Z
1 0 0	X OR Y -> Z
1 0 1	X XOR Y -> Z
1 1 0	Eins-Komplement von X -> Z
1 1 1	falls X = Y, -1 -> Z, sonst 0 -> Z

OpCode	*Mnemonik*	*Beschreibung*
0	LDC c	c -> Akku
1	LDV a	<a> -> Akku
2	STV a	Akku -> <a>
3	ADD a	Akku + <a> -> Akku
4	AND a	Akku AND <a> -> Akku
5	OR a	Akku OR <a> -> Akku
6	XOR a	Akku XOR <a> -> Akku
7	EQL a	falls Akku = <a>:-1 -> Akku
		sonst : 0 -> Akku
8	JMP a	a -> IAR
9	JMN a	falls Akku < 0 : a -> IAR
F0	HALT	stoppt die MIMA
F1	NOT	bilde Eins-Komplement von Akku -> Akku
F2	RAR	rotiere Akku eins nach rechts -> Akku

Befehlsformate