Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 0
# Rechnerarchitektur im SoSe 2020

**Ankündigungen:** Herzlich willkommen zum Übungsbetrieb zur Vorlesung Rechnerarchitektur. Bitte melden Sie sich zu den Übungsgruppen im `Uni2Work` an. Beachten Sie dazu auch die Hinweise auf dem Merkblatt. Um kurzfristige Ankündigungen nicht zu verpassen, bitten wir Sie auch regelmäßig die Website zur Vorlesung zu besuchen:
`http://www.mobile.ifi.lmu.de/lehrveranstaltungen/rechnerarchitektur-sose20/`

## Aufgabe Ü1: Beitritt zu den Kleingruppenübungen über Zoom

(– Pkt.)

Für die Übungen kommt das Videokonferenz Tool Zoom zum Einsatz.
Um einer Videokonferenz beizutreten empfiehlt es sich, zunächst unter

`https://lmu-munich.zoom.us/`

auf "Sign in" zu klicken.

Dort kann man sich mit dem LMU-Benutzernamen oder der Campus-Kennung anmelden.

Danach kann man erneut unter

`https://lmu-munich.zoom.us/`

auf "Join" klicken, um einer Videokonferenz unter Angabe einer Meeting-ID (und ggf. eines Passworts) beizutreten.

Die Einwahldaten für Ihre spezifische Übung werden Ihnen rechtzeitig per E-Mail durch Ihren Tutor mitgeteilt (setzt eine Anmeldung zum Kurs und zu einer Übungsgruppe über Uni2work voraus).

Der Beitritt wurde erfolgreich im Chrome/Chromium Browser getestet. Es steht aber auch ein spezieller Zoom-Client zur Verfügung.
Es ist auch eine Einwahl per Telefon unter folgenden Nummern möglich:

+49 30 5679 5800 Deutschland
+49 695 050 2596 Deutschland
+49 69 7104 9922 Deutschland

Bei technischen Problemen können Sie sich per E-Mail an it-servicedesk@lmu.de wenden.

## **Aufgabe Ü2:** Einführung: Zahlensysteme

(– Pkt.)

In dieser Aufgabe sollen Sie sich die Grundlagen der Zahlensysteme nocheinmal bewusst machen. In jedem Zahlensystem muss eine Basis $b$ gewählt werden. Im Dezimalsystem gilt $b = 10$, im Binärsystem $b = 2$ und im Oktalsystem $b = 8$ und im Hexadezimalsystem $b = 16$. Beantworten Sie nun folgende Fragen zu Zahlensystemen:

a. Geben Sie sämtliche Ziffern des Hexadezimalsystems an.

b. Warum benutzt man für die Ziffern 10-15 Buchstaben und nicht die Zahlen 10-15?

c. Geben sie zu jeder der folgenden Dezimalzahlen ihre Binär-, Oktal- und Hexadezimaldarstellung an.

   (i) $(18)_{10}$
   (ii) $(53)_{10}$

d. Wandeln Sie folgende Dualzahlen in ihre Hexadezimaldarstellung um:

   (i) $(0101011001100110)_2$
   (ii) $(1111010000011111)_2$

e. Die Umrechnung einer Dualzahl in eine Hexadezimalzahl lässt sich durch folgendes Verfahren sehr schnell durchführen:

   – Die Dualzahl wird – beim niederwertigsten Bit beginnend – in 4er-Blöcke aufgeteilt.
   – Besteht die Dualzahl aus einer nicht durch 4 teilbaren Anzahl Bits, so wird vorne mit Nullen aufgefüllt. (11101 etwa wird somit zu 00011101 und dann aufgeteilt in die Blöcke 0001 und 1101.)
   – Anschließend wird jeder Block (bestehend aus vier Bits) separat in seine Hexadezimal-Darstellung transformiert.
   – In der ursprünglichen Reihenfolge der Blöcke ergibt die Konkatenation dieser Transformationen die Hexadezimaldarstellung der Ausgangszahl.

   (i) Wandeln Sie nun die Dualzahl $(101110101111)_2$, unter Anwendung des gerade gelernten Verfahrens, in das Hexadezimalsystem um.
   (ii) Welche kleine Änderung müssten Sie an diesem Algorithmus vornehmen, um nach dem prinzipiell gleichen Verfahren eine Dual-Oktal-Transformation zu erhalten? Eräutern Sie die Anpassung und führen Sie dann die Umwandlung der Dualzahl $(101110101111)_2$ in das Oktalsystem mit Ihrem angepassten Verfahren durch.

## **Aufgabe Ü3:** Adressdarstellung

(– Pkt.)

Viele Rechner besitzen einen Hauptspeicher, in dem 4-Byte-Worte gespeichert werden können. Das bedeutet, dass mit einer einzigen Operation 4 Bytes zwischen Speicher und Prozessor ausgetauscht werden können. Jedes Wort besitzt eine Adresse (wie eine Raumnummer). Adressen selbst sind binäre Zahlen, d.h. Bitfolgen einer gegebenen Länge. Die Bitfolge 0...00 adressiert das erste Wort, die Bitfolge 0...01 adressiert das zweite Wort, etc.

a. Nehmen Sie an, dass

   (i) für die Adressen eine Bitfolge von 1 Byte verwendet wird;

(ii) für die Adressen eine Bitfolge von 2 Byte verwendet wird;

Geben Sie für jeden Fall die Adresse des letzten Speicherwortes an, das mit der gegebenen Anzahl von Bytes adressiert werden kann:

a) in binärer Notation

b) in oktaler Notation[1]

c) in hexadezimaler Notation

d) in dezimaler Notation

(Hinweis: die Adresse des ersten Wortes ist 0)

b. Nehmen Sie an die Wortlänge sei 4 Bytes, und 2 Bytes werden für den Adressen-Bitstring verwendet.[2] Nehmen Sie weiterhin an, dass 380 Bytes im Speicher abgelegt werden sollen. Die ersten 4 Bytes werden an der Adresse $123A_{16}$ gespeichert. Der Rest wird ohne Lücken in den folgenden Speicherzellen abgelegt. Bestimmen Sie die Adresse des letzten Speicherwortes, das noch verwendet wird, um die 380 Bytes zu speichern.

Geben Sie die Antwort als i) binäre Zahl, ii) oktale Zahl, iii) dezimale Zahl und iv) als hexadezimale Zahl an.

[1] Das Oktalsystem ist ein Stellenwertsystem zur Basis 8 mit den Ziffern {0, 1, 2, 3, 4, 5, 6, 7} zur Darstellung einer Zahl.

[2] Wenn die Wortlänge 4 Bytes beträgt, dann ist die Adresse normalerweise auch 4 Bytes breit. Für diese Übung nehmen wir jedoch an, dass die Adresse 2 Bytes breit ist.

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 1
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen A, B

**Abgabetermin:** 28.04.2020, 23:59 Uhr

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 04. – 8. Mai 2020

**Ankündigungen:** Herzlich willkommen zum Übungsbetrieb zur Vorlesung Rechnerarchitektur. Bitte melden Sie sich zu den Übungsgruppen im `Uni2Work` an. Beachten Sie dazu auch die Hinweise auf dem Merkblatt. Um kurzfristige Ankündigungen nicht zu verpassen, bitten wir Sie auch regelmäßig die Website zur Vorlesung zu besuchen:
`http://www.mobile.ifi.lmu.de/lehrveranstaltungen/rechnerarchitektur-sose20/`

## Aufgabe Ü1: Übertragungsgeschwindigkeiten

(10 Pkt.)

Die meisten Laserdrucker können mit einer Auflösung von 1200 dpi (hier synonym zu Pixel pro Zoll, oder kurz ppi) drucken.

a. Wenn die Datenübertragung zum Drucker Pixel für Pixel bei 3*8-Bit Farben erfolgt, wie lange dauert dann die Übertragung einer DIN A4 Farbseite (21 cm x 29,7 cm) zum Drucker bei Verwendung folgender Übertragungsmöglichkeiten?

**Hinweis:** Sie können davon ausgehen, dass die hier angegebenen Übertragungsraten verlustlos ohne Protokoll-Overhead ausgenutzt werden können. Rechnen Sie die cm zunächst in Zoll um und runden Sie dieses Ergebnis auf 2 Nachkommastellen! Der Rechenweg muss ersichtlich und nachvollziehbar sein!
Aus historischen Gründen wird bei der Angabe von Datenmengen in `KByte,MByte, ...` der Umrechnungsfaktor 1024 verwendet; bei Angaben in `KBit/s, MBit/s, ...` wird der Umrechnungsfaktor 1000 verwendet. Sie können die folgende Umrechnungstabelle benutzen:

| | | |
|---|---|---|
| 1 GBit/s = $10^9$ Bit/s | 1 MBit/s = $10^6$ Bit/s | 1 KBit/s = $10^3$ Bit/s |
| 1 GByte = $2^{30}$ Byte | 1 MByte = $2^{20}$ Byte | 1 KByte = $2^{10}$ Byte |

(i) Wireless LAN (IEEE 802.11n) mit 600 MBit/s

(ii) Ethernet mit 1 GBit/s

b. Nehmen Sie nun an, dass anstatt von Pixeln eine 16-Bit Codierung jedes Zeichens (ein Zeichen wird durch 16 Bit dargestellt) zusammen mit seinen Koordinaten auf der Seite übertragen wird. Dabei können die Zeichen frei auf der Seite positioniert werden indem der Ankerpunkt eines Zeichens eine Koordinate erhält.

(i) Wie viele Bits werden für die Koordinaten benötigt, wenn man die volle Auflösung von 1200 dpi ausnutzen will (d.h. wenn die horizontalen und vertikalen Koordinaten aller Pixel binär kodiert werden müssen)?

(ii) Wie lange dauert es jeweils, 100 Seiten mit 1800 Zeichen an den Drucker zu übertragen, wenn das Adressierungsschema aus Aufgabe bi) und die gleichen Übertragungsmöglichkeiten wie aus Aufgabe a) zur Verfügung stehen? Runden Sie ihr Ergebnis auf 4 Nachkommastellen!

## **Aufgabe Ü2:** Zahlensysteme

(9 Pkt.)

Bearbeiten Sie folgende Fragen zu Zahlensystemen:

a. Geben sie zu jeder der folgenden Dezimalzahlen ihre Binär-, Oktal- und Hexadezimaldarstellung an.

(i) $(17)_{10}$

(ii) $(42)_{10}$

(iii) $(255)_{10}$

b. Geben Sie zu folgenden Dualzahlen die Oktal-, Dezimal- und Hexadezimaldarstellung an:

(i) $(10001111)_2$

(ii) $(11010101)_2$

(iii) $(00011110)_2$

## **Aufgabe Ü3:** Einfachauswahlaufgabe: Einführung

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

| a) Wie viele Bit enthält ein Byte? | | | |
|---|---|---|---|
| (i) 32 | (ii) 64 | (iii) 8 | (iv) 16 |
| b) Welche Binärzahl entspricht dem hexadezimalen Wert `F`? | | | |
| (i) `1111` | (ii) `0111` | (iii)`1110` | (iv) `0110` |
| c) Welche Komponente ist gewöhnlich nicht an die South Bridge eines Mainboard-Chipsatzes angebunden? | | | |
| (i) Festplatten | (ii) Audio-Ausgang | (iii) Hauptspeicher | (iv) USB-Schnittstellen |
| d) Wie lautet die höchste Speicheradresse bei einer Adressbreite von $n$ Bit? | | | |
| (i)$2/n-1$ | (ii) $2*n-1$ | (iii) $2^n-1$ | (iv) $2+n-1$ |
| e) Was ist keine Komponente der Prozessorgrundstruktur? | | | |
| (i) Arithmetisch-logische Einheit | (ii) Befehlsregister | (iii) Operanden-register | (iv) Drucker |

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 2
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen C, D

**Abgabetermin:** 05.05.2020, 23:59 Uhr

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 11. – 15. Mai 2020

## Aufgabe Ü4: Funktionstabelle

(10 Pkt.)

Gegeben sei folgende Booleschen Funktion $g(a, b, c) = a \vee \overline{b} \vee (a \wedge c)$.

a. Füllen Sie folgende Funktionstabelle aus:

| $a$ | $b$ | $c$ | $g(a, b, c) = a \vee \overline{b} \vee (a \wedge c)$ |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

b. Gegeben sei die folgende Funktionstabelle von sechs dreistelligen Booleschen Funktion $f_1, \ldots, f_6$.

| A | B | C | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |

Schreiben Sie diese Funktionen als Boolesche Terme, wobei sie ausschließlich die Variablen $A, B$ und $C$ benutzen dürfen (insbesondere dürfen Sie die Werte $0$ und $1$ nicht verwenden)! Nicht in jedem der resultierenden Terme müssen alle Variablen vorhanden sein.

c. Stellen Sie die Funktion $h(a, b, c) = (a \wedge b) \vee c$ unter ausschließlicher Verwendung des NOR-Operators dar! Der Rechenweg muss klar ersichtlich sein!

## **Aufgabe Ü5:** Multiplexer

(6 Pkt.)

In dieser Aufgabe sollen logische Gatter durch 2-Eingaben Multiplexer dargestellt werden. Gehen Sie davon aus, dass S die Steuerleitung ist und für $S = 0$ der Eingang $In_0$ und für $S = 1$ der Eingang $In_1$ selektiert wird. So kann die Funktion $A \wedge B$ zum Beispiel durch Anlegen von 0 an den Eingang $In_0$, A an den Eingang $In_1$ und B an die Steuerleitung S durch einen Multiplexer realisiert werden.


Erstellen Sie zwei 2-Eingaben Multiplexer, welche folgende Eigenschaften erfüllen sollen:

a. Dieser Multiplexer soll den Term $A \vee B$ an seinem Ausgang erzeugen.

b. Dieser Multiplexer soll den Term $\overline{A}$ an seinem Ausgang erzeugen.

Für jeden der Eingänge des Multiplexers ($In_0$, $In_1$ sowie die Steuerleitung S) dürfen Sie ausschließlich die Werte A, B, $\overline{A}$, $\overline{B}$ sowie 0 und 1 benutzen. Sie dürfen keine weiteren Bausteine oder Gatter benutzen.

## **Aufgabe Ü6:** Einfachauswahlaufgabe: Boolesche Algebra

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

| | | | |
|---|---|---|---|
| a) Bei welcher Belegung $(a, b)$ ergibt der OR–Operator ($+$ oder $\vee$) den Wert 0? | | | |
| (i) $(0, 0)$ | (ii) $(0, 1)$ | (iii) $(1, 0)$ | (iv) $(1, 1)$ |
| b) Bei welcher Belegung $(a, b)$ ergibt der AND–Operator ($\cdot$ oder $\wedge$) den Wert 1? | | | |
| (i) $(0, 0)$ | (ii) $(0, 1)$ | (iii) $(1, 0)$ | (iv) $(1, 1)$ |
| c) Eine Funktion $f : B^n \to B$ heißt n–stellige Boolesche Funktion ($B = \{0, 1\}$). Wie viele n–stellige Boolesche Funktionen gibt es für jedes beliebige $n \in \mathbb{N}$ mit $n \geq 1$? | | | |
| (i) $2^n$ | (ii) $2^{2^n}$ | (iii) $2 \cdot 2^n$ | (iv) $2^{2 \cdot n}$ |
| d) Bei welcher Belegung $(x_1, x_2, x_3, x_4)$ ergibt die Boolesche Funktion $f(x_1, x_2, x_3, x_4) = (x_1 \cdot x_2 \cdot \overline{x_3}) + (x_3 \cdot x_4) + \overline{x_2}$ den Wert 1? | | | |
| (i) $(1, 1, 1, 0)$ | (ii) $(0, 1, 1, 0)$ | (iii) $(0, 1, 0, 1)$ | (iv) $(0, 0, 0, 0)$ |
| e) Wie wird die Anzahl der benötigten Steuereingänge s für einen n-Eingaben Multiplexer berechnet? | | | |
| (i) $s = n$ | (ii) $s = 2 * n$ | (iii) $s = \log_2 n$ | (iv) $s = \lceil \log_2 n \rceil$ |

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 3
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen E, F

**Abgabetermin:** 12.05.2020, 23:59 Uhr

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 18. – 22. Mai 2020

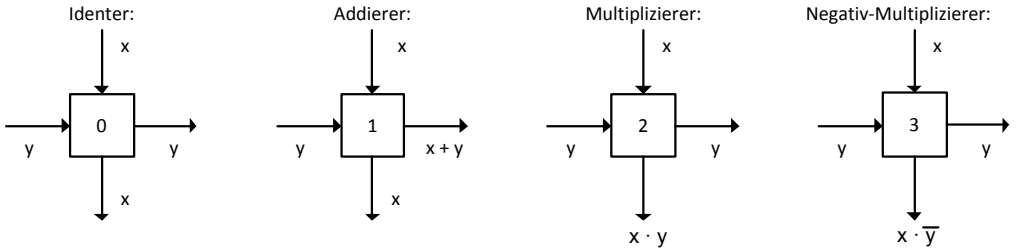
## Aufgabe Ü7: PLA-Entwurf

(10 Pkt.)

Gegeben sei die folgende Boolesche Funktion

$$f(x, y, z) = \left( x\bar{y} + \bar{x}y\bar{z},\ yz + \bar{z} \right)$$

Realisieren Sie diese Funktion durch ein normiertes PLA, welches aus der minimal möglichen Anzahl an Zeilen und Spalten besteht. Verwenden Sie ausschließlich die im Folgenden dargestellten Bausteine vom Typ 0 bis 3. Kennzeichnen Sie in Ihrer Skizze die Und- und die Oder-Ebene. Markieren Sie gesperrte und neutralisierte Eingänge. Beschriften Sie jeden Pfeil (sowohl ausgehende als auch die innerhalb des PLAs) mit der jeweils anliegenden logischen Funktion. Die zur Verfügung stehenden Bausteine sind:




## Aufgabe Ü8: Einfachauswahlaufgabe: Normalformen und PLA

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

a) Sei folgende Wahrheitstafel einer Booleschen Funktion gegeben. Was ist die Menge der einschlägigen Indizes?

| i | $x_1$ | $x_2$ | $f(x_1, x_2)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 |
| 3 | 1 | 1 | 0 |

| | | | |
|---|---|---|---|
| (i) $\{0\}$ | (ii) $\{0, 1, 3\}$ | (iii) $\{0, 1\}$ | (iv) $\{2\}$ |

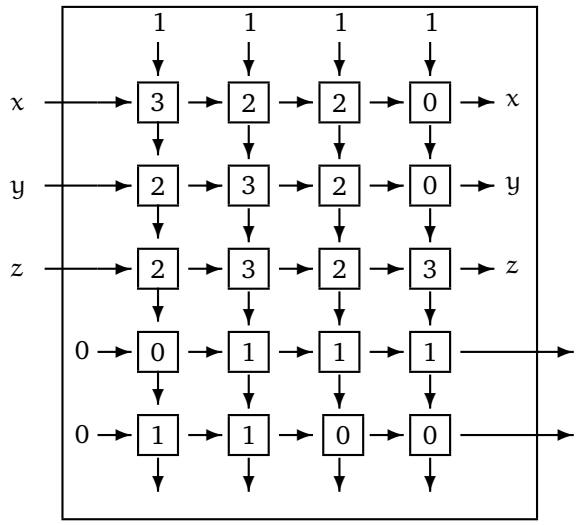
b) Welche der Aussage zu Normalformen von Schaltfunktionen ist korrekt?

| | | | |
|---|---|---|---|
| (i) Die DNF einer Funktion ist immer kürzer als die KNF | (ii) Die DNF einer Funktion ist immmer länger als die KNF | (iii) DNF und KNF einer Funktion sind immer gleich lang | (iv) Eine allgemeingültige Aussage darüber, ob die DNF oder die KNF einer Funktion länger ist, kann nicht getroffen werden |

c) Jede Boolesche Funktion $F : B^n \to B$ ist eindeutig darstellbar als...

| | | | |
|---|---|---|---|
| (i) ...Summe der Maxterme ihrer einschlägigen Indizes. | (ii) ...Produkt der Minterme ihrer einschlägigen Indizes. | (iii) ...Summe der Minterme ihrer nichteinschlägigen Indizes. | (iv) ...Summe der Minterme ihrer einschlägigen Indizes. |

d) Welche Boolesche Funktion realisiert folgendes PLA?




| | | | |
|---|---|---|---|
| (i) $f(x, y, z) = (\overline{y}\,\overline{z} + xyz + \overline{z}, \overline{x}yz + \overline{y}\,\overline{z})$ | (ii) $f(x, y, z) = (\overline{y}\,\overline{z}, xyz + \overline{y})$ | (iii) $f(x, y, z) = (\overline{y}\,\overline{z}, \overline{x}yz + \overline{y}z)$ | (iv) $f(x, y, z) = (x\overline{y}\,\overline{z} + xyz + \overline{z}, \overline{x}yz + x\overline{y}\,\overline{z})$ |

e) Sei folgende Wahrheitstafel einer Booleschen Funktion $f : B^2 \to B$ gegeben.
Welcher Ausdruck entspricht der disjunktiven Normalform (DNF)?

| $i$ | $x_1$ | $x_2$ | $f(x_1, x_2)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 |

| | | | |
|---|---|---|---|
| (i) $f(x_1, x_2) = \overline{x}_1 x_2 + x_1 \overline{x}_2 + \overline{x}_1 \overline{x}_2$ | (ii) $f(x_1, x_2) = \overline{x}_1 + x_2$ | (iii) $f(x_1, x_2) = x_1 + \overline{x}_2$ | (iv) $f(x_1, x_2) = \overline{x}_1 \overline{x}_2 + \overline{x}_1 x_2 + x_1 x_2$ |

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 4
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen G, H

**Abgabetermin:** 19.05.2020, 23:59 Uhr

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 25. – 29. Mai 2020

## Aufgabe Ü9: Optimierung von Schaltnetzen

(11 Pkt.)

a. Gegeben sei folgende Wahrheitstabelle einer Funktion $f(x_1, x_2, x_3, x_4)$.
Leiten Sie aus dieser Wahrheitstabelle die Schaltfunktion in ihrer vollständigen konjunktiven Normalform (**KNF**) her.

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $f(x_1, x_2, x_3, x_4)$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 0 |
| 12 | 1 | 1 | 0 | 0 | 1 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | 1 |

b. Im Folgenden ist die Wahrheitstabelle der Funktion $g(x_1, x_2, x_3, x_4)$ gegeben.
Minimieren Sie die Funktion g unter Verwendung eines Karnaugh-Diagramms grafisch. Kennzeichnen Sie **alle** Blöcke innerhalb Ihres Karnaugh-Diagramms, die Sie für Ihre Vereinfachung verwenden! Geben Sie abschließend die minimierte Funktion in disjunktiver Form an!

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $g(x_1, x_2, x_3, x_4)$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 1 |

## **Aufgabe Ü10:** Einfachauswahlaufgabe: Karnaugh und Quine-McCluskey

( 5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

| a) Das Karnaugh-Diagramm basiert auf.. | | | |
|---|---|---|---|
| (i) der Resolutionsregel. | (ii) dem Verschmelzungsgesetz. | (iii) dem Identitätsgesetz. | (iv) den de-morganschen Regeln. |
| b) Die Reihenfolge der Beschriftung eines Karnaugh-Diagramms erfolgt so, dass sich zwei zyklisch benachbarte Spalten oder Zeilen in... | | | |
| (i) keiner Komponente (Variable) unterscheiden. | (ii) zwei Komponenten (Variablen) unterscheiden. | (iii) in allen Komponenten (Variablen) unterscheiden. | (iv) genau einer Komponente (Variable) unterscheiden. |
| c) Es kann sein, dass nicht alle $2^n$ Argumente einer Booleschen Funktion $f : B^n \to B$ ($n \geq 1$) auftreten können. Wie bezeichnet man die Argumente einer solchen partiellen Funktion $f$, für die der Funktionswert nicht festgelegt ist? | | | |
| (i) Don't Panic | (ii) Don't Worry | (iii) Don't Cares | (iv) Don't Know |
| d) Wie viele Felder enthält das Karnaugh-Diagramm einer Booleschen Funktion $f : B^3 \to B$? | | | |
| (i) 1 | (ii) 4 | (iii) 8 | (iv) 2 |
| e) Welche Aussage zum Algorithmus von QuineMcCluskey stimmmt nicht? | | | |
| (i) Die minimierte Funktion ist immer die Summe aller Primimplikanten | (ii) Terme werden anhand der Anzahl der negierten Variablen gruppiert. | (iii) Terme, die nicht mehr zusammengefasst werden können, nennt man Primimplikanten | (iv) Terme, die sich in der Verneinung von genau einer Variablen unterscheiden, werden zusammengefasst. |

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 5
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen C, D, H

**Abgabetermin:** 26.05.2020, 23:59 Uhr

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 01. – 05. Juni 2020

## Aufgabe Ü11: Quine-McCluskey

(10 Pkt.)

Gegeben ist die folgende boolesche Funkion:

$$f(x) = x_1x_2x_3x_4 + x_1\overline{x}_2x_3x_4 + x_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2x_3x_4 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + \overline{x}_1\overline{x}_2\overline{x}_3x_4$$

Die Funktion $f(x)$ soll mit Hilfe des Algorithmus von Quine-McCluskey vereinfacht werden. Der Beginn der Vereinfachung durch den Algorithmus von Quine-McCluskey ist bereits wie folgt gegeben:

Bestimmung der Implikanten

| Gruppe | Minterm | Einschl. Index | | |
|---|---|---|---|---|
| 0 | $x_1x_2x_3x_4$ | 1111 | = | 15 |
| 1 | $x_1\overline{x}_2x_3x_4$ | 1011 | = | 11 |
| 2 | $x_1\overline{x}_2\overline{x}_3x_4$ | 1001 | = | 09 |
| | $\overline{x}_1\overline{x}_2x_3x_4$ | 0011 | = | 03 |
| 3 | $\overline{x}_1x_2\overline{x}_3\overline{x}_4$ | 0100 | = | 04 |
| | $\overline{x}_1\overline{x}_2x_3\overline{x}_4$ | 0010 | = | 02 |
| | $\overline{x}_1\overline{x}_2\overline{x}_3x_4$ | 0001 | = | 01 |

Verkürzung der Implikanten

| Gruppe | Implikant | Einschl. Index | | |
|---|---|---|---|---|
| 0 | $x_1\square x_3x_4$ | 1*11 | = | 11, 15 |
| 1 | $x_1\overline{x}_2\square x_4$ | 10*1 | = | 9, 11 |
| | $\square\overline{x}_2x_3x_4$ | *011 | = | 3, 11 |
| 2 | $\square\overline{x}_2\overline{x}_3x_4$ | *001 | = | 9, 1 |
| | $\overline{x}_1\overline{x}_2x_3\square$ | 001* | = | 2, 3 |
| | $\overline{x}_1\overline{x}_2\square x_4$ | 00*1 | = | 1, 3 |
| 3 | $\overline{x}_1x_2\overline{x}_3\overline{x}_4$ | 0100 | = | 4 |

a. Geben Sie die noch fehlenden Schritte des Algorithmus von Quine-McCluskey an! Jeder Ihrer Schritte muss klar nachvollziehbar sein! Geben Sie insbesondere auch die resultierende Primimplikantentabelle an.

b. Geben Sie die minimierte Funktion an.

c. Bestimmen Sie die Kosten der Realisierung dieser Funktion vor und nach der Optimierung.

## **Aufgabe Ü12:** Einfachauswahlaufgabe: Boolesche Algebra und Multiplexer

( 5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

<table>
<tr><td colspan="4">a) Welche der folgenden Mengen an Booleschen Funktion ist nicht funktional vollständig?</td></tr>
<tr><td>(i) $\{\text{AND}, \text{NOT}\}$</td><td>(ii) $\{\text{AND}, \text{OR}\}$</td><td>(iii) $\{\text{OR}, \text{NOT}\}$</td><td>(iv) $\{\text{NAND}\}$</td></tr>
<tr><td colspan="4">b) Angenommen ein Multiplexer hat <code>512</code> (Nutz-)Eingänge. Wie viele Steuereingänge werden benötigt, um die (Nutz-)Eingänge einzeln selektieren zu können?</td></tr>
<tr><td>(i) <code>9</code></td><td>(ii) <code>512</code></td><td>(iii) <code>3</code></td><td>(iv) <code>256</code></td></tr>
<tr><td colspan="4">c) Wie lautet das Komplementärgesetz zur Manipulation logischer Gleichungen?</td></tr>
<tr><td>(i) $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$</td><td>(ii) $a + b = b + a$</td><td>(iii) $(a + b) + c = a + (b + c)$</td><td>(iv) $a + \overline{a} = 1$</td></tr>
<tr><td colspan="4">d) Welcher der folgenden Booleschen Terme ist äquivalent zu $(x_1 \cdot x_2) + x_1 + x_3$?</td></tr>
<tr><td>(i) $(x_1x_2x_3) \cdot (x_1\overline{x_2}x_3)$</td><td>(ii) $(x_1 + x_2 + x_3) + (x_1\overline{x_2}x_3)$</td><td>(iii) $(\overline{x_1}x_2x_3) + (x_1\overline{x_2}x_3)$</td><td>(iv) $(x_1 + x_2 + x_3) \cdot (x_1 + \overline{x_2} + x_3)$</td></tr>
<tr><td colspan="4">e) Sei folgende Wahrheitstafel einer Booleschen Funktion $f : B^2 \to B$ gegeben.<br>Welcher Ausdruck entspricht nicht dieser Funktion?<br><br>
<table>
<tr><th>i</th><th>$x_1$</th><th>$x_2$</th><th>$f(x_1, x_2)$</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>(i) $f(x_1, x_2) = (x_1 + \overline{x}_2) \cdot (\overline{x}_1 + \overline{x}_2)$</td><td>(ii) $f(x_1, x_2) = \overline{(\overline{x}_1 \cdot x_2)}$</td><td>(iii) $f(x_1, x_2) = \overline{\overline{(x_1 \cdot x_1)} \cdot x_2}$</td><td>(iv) $f(x_1, x_2) = x_1 + \overline{x}_2$</td></tr>
</table>

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 6
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen I, J

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 08. – 12. Juni 2020

## **Aufgabe Ü13:** Addition von Dualzahlen

(– Pkt.)

Beantworten Sie folgende Fragen im Bezug auf die 2er-Komplement-Darstellung ganzer Zahlen:

a. Geben Sie die größte und die kleinste darstellbare Zahl, sowie die Null bei Verwendung von 8 Bits an.

b. Folgende Dualzahlen in 2er-Komplement-Darstellung sind gegeben: 10010001 und 10011011.

   (i) Addieren Sie die beiden Zahlen.

   (ii) Hat bei der Addition ein Überlauf (Overflow) stattgefunden? Begründen Sie kurz Ihre Antwort.

c. Folgende Dualzahlen in 2er-Komplement-Darstellung sind gegeben: 10010001 und 01110011. Ohne Rechnung: Wird bei der Addition dieser Zahlen ein Überlauf oder ein Übertrag stattfinden? Bitte begründen Sie Ihre Antwort. Erklären Sie auch den Unterschied.

## **Aufgabe Ü14:** Gleitkommazahlen

(– Pkt.)

a. Geben Sie die Dezimaldarstellung der folgenden Gleitkommazahlen an. Interpretieren Sie die Kommazahl und den Exponenten jeweils als Sign/Magnitude Darstellung. Also das jeweils erste Bit von Mantisse und Exponent gilt als Vorzeichenbit.

   (i) $(011,01)_2 \cdot 2^{(0101)_2}$

   (ii) $(110,11)_2 \cdot 2^{(0011)_2}$

   (iii) $(111,01)_2 \cdot 2^{(1011)_2}$

b. Geben Sie die Darstellung folgender Zahlen als Gleitkommazahl nach IEEE 754 in einfacher (32-Bit) Genauigkeit an. Hinweis: nach dem IEEE 754 Standard gilt folgendes:

$$(-1)^S \cdot (1 + \text{Signifikant}) \cdot 2^{(\text{Exponent}-\text{Bias})}$$

   wobei der Standard

- für das Vorzeichen $S$ ein Bit,
- für den Signifikanten (Mantisse) 23 Bit bei einfacher und 52 Bit bei doppelter Genauigkeit,
- für den Exponenten 8 Bit bei einfacher und 11 Bit bei doppelter Genauigkeit

reserviert und den Bias auf $127 = 2^{8-1} - 1$ bei einfacher bzw. auf $1023 = 2^{11-1} - 1$ bei doppelter Genauigkeit setzt.

(i) $(15,75)_{10}$

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

c. Wandeln Sie folgende Zahl, die in Gleitkommadarstellung (IEEE 754) gegeben ist, in ihre Dezimaldarstellung um.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| S | Exponent | | | | | | | | Significand | | | | | | | | | | | | | | | | | | | | | | |

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 7
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen K

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 15. – 19. Juni 2020

## **Aufgabe Ü15:** Assemblerprogrammierung unter SPIM

(– Pkt.)

Im Folgenden soll ein MIPS-Assembler Programm vervollständigt werden, welches als Nutzereingabe eine 8-stellige Binärzahl von der Konsole entgegennimmt, sie in ihre dezimale Repräsentation umwandelt und diese auf der Konsole ausgibt. Die Eingabe wird vom Programm als String entgegengenommen. Danach soll in einer Schleife über die Zeichen diese Strings iteriert und geprüft werden, ob es sich beim aktuellen Zeichen um eine „0“ oder eine „1“ handelt und die entsprechende Wertigkeit aufsummiert werden. Zudem muss der Fall behandelt werden, dass das Ende des Strings, welches durch ein Byte mit dem Zahlenwert `0` markiert ist, erreicht wurde. Beachten Sie, dass der Sting nach 8 von der Konsole gelesen Zeichen automatisch übernommen und Null-terminiert wird. Dementsprechend ist kein Zeilenumbruch enthalten.
Laden sie sich die `binarytodecimal.s` von der Homepage herunter und ergänzen Sie den dort angegebenen Coderahmen um insgesamt **6 Zeilen Code**, so dass das Programm wie beschrieben funktioniert. Tragen Sie Ihre Lösung unter den mit “`# Ihre Loesung:`” markierten Stellen direkt in den Coderahmen der heruntergeladenen Datei ein.

## **Aufgabe Ü16:** Interaktion mit dem Stack

(– Pkt.)

Sei folgendes MIPS-Codefragment gegeben:

```
1        li $t0, 30
2        li $t1, 5
3        li $t2, 15
4        li $t3, 10
5        addi $sp, $sp, -12
6        sw $t2, 12($sp)
7        sw $t3, 12($sp)
8        sw $t0, 8($sp)
9        sw $t1, 4($sp)
```

$sp = stack pointer

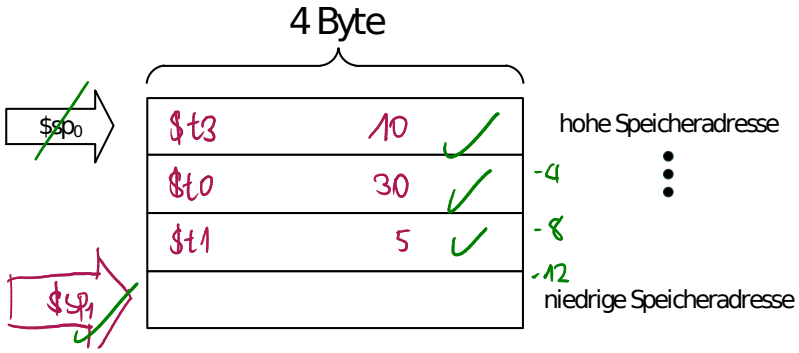
verschoben ↑

(nun $sp 3 down) // um 12 heruntergesetzt.

15
10
30
5

Skizzieren Sie in folgenden Vorlage den Inhalt des Stacks nachdem alle Programmzeilen ausgeführt wurden. Kennzeichnen Sie auch die neue Position des Stackpointers!

12($sp) = (12 + $sp) → Address Operator




Stackpointer : zeigt im Stack die Add der nächste freie Speicherzelle.

Program Counter : Next Instruction to execute


Stack : Super Easy to free up memory !
Just move the $sp up !

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 7
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen K

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 15. – 19. Juni 2020

## **Aufgabe Ü15:** Assemblerprogrammierung unter SPIM

(– Pkt.)

Im Folgenden soll ein MIPS-Assembler Programm vervollständigt werden, welches als Nutzereingabe eine 8-stellige Binärzahl von der Konsole entgegennimmt, sie in ihre dezimale Repräsentation umwandelt und diese auf der Konsole ausgibt. Die Eingabe wird vom Programm als String entgegengenommen. Danach soll in einer Schleife über die Zeichen diese Strings iteriert und geprüft werden, ob es sich beim aktuellen Zeichen um eine „0“ oder eine „1“ handelt und die entsprechende Wertigkeit aufsummiert werden. Zudem muss der Fall behandelt werden, dass das Ende des Strings, welches durch ein Byte mit dem Zahlenwert `0` markiert ist, erreicht wurde. Beachten Sie, dass der Sting nach 8 von der Konsole gelesen Zeichen automatisch übernommen und Null-terminiert wird. Dementsprechend ist kein Zeilenumbruch enthalten.
Laden sie sich die `binarytodecimal.s` von der Homepage herunter und ergänzen Sie den dort angegebenen Coderahmen um insgesamt **6 Zeilen Code**, so dass das Programm wie beschrieben funktioniert. Tragen Sie Ihre Lösung unter den mit “`# Ihre Loesung:`” markierten Stellen direkt in den Coderahmen der heruntergeladenen Datei ein.

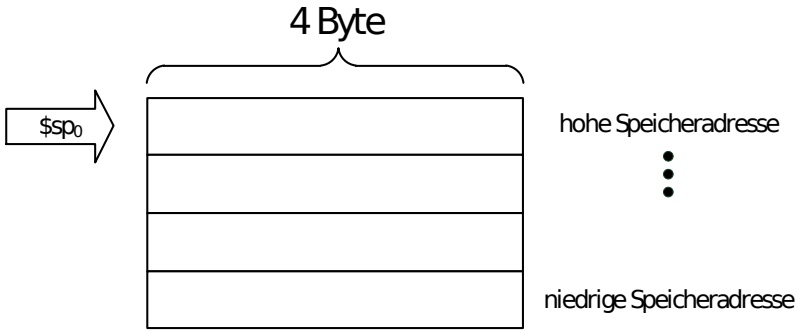
## **Aufgabe Ü16:** Interaktion mit dem Stack

(– Pkt.)

Sei folgendes MIPS-Codefragment gegeben:

```
1          li $t0, 30
2          li $t1, 5
3          li $t2, 15
4          li $t3, 10
5          addi $sp, $sp, -12
6          sw $t2, 12($sp)
7          sw $t3, 12($sp)
8          sw $t0, 8($sp)
9          sw $t1, 4($sp)
```

Skizzieren Sie in folgenden Vorlage den Inhalt des Stacks nachdem alle Programmzeilen ausgeführt wurden. Kennzeichnen Sie auch die neue Position des Stackpointers!


4 Byte
$sp_0$
hohe Speicheradresse
niedrige Speicheradresse

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 8
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen K

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 22. – 26. Juni 2020

## **Aufgabe Ü15:** Binomialkoeffizient

(– Pkt.)

Der Binomialkoeffizient dient dazu, die Anzahl $k$-elementiger Teilmengen einer $n$-elementigen Menge zu berechenen. Die etwas vereinfachte Definition des Binomialkoeffizienten laute folgendermaßen:

$$\binom{n}{k} := \frac{n!}{k!(n-k)!} = \begin{cases} \textit{Fehler} & \text{wenn } k > n \\ 1 & \text{wenn } k = 0 \\ \frac{n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot(n-k+1)}{k\cdot(k-1)\cdot(k-2)\cdot\ldots\cdot 1} & \text{sonst} \end{cases}$$

Schreiben Sie ein MIPS-Assembler-Programm, welches den Binomialkoeffizienten entsprechend der obigen Definition berechnet. Das Programm soll dazu die zwei Werte $n$ und $k$ über die Konsole einlesen und das Ergebnis der Berechnung wieder auf der Konsole ausgeben.
Halten Sie sich an die folgenden Vorgaben:
Berechnen Sie den Zähler und den Nenner getrennt, aber in ein und derselben Schleife (Hinweis: Verwenden Sie dazu die ausgeschriebene Form und nicht die mit den Fakultätstermen!) Führen Sie die Division als letzten Schritt durch! Werden zwei Zahlen mit $n < k$ eingegeben, so soll eine Fehlermeldung generiert werden. Das Programm soll auch Grenzfälle, z.B. $\binom{0}{0}$, richtig berechnen.
**Kommentieren Sie Ihr Programm ausführlich!**

## **Aufgabe Ü16:** Addition mit doppelter Genauigkeit

(– Pkt.)

Ein MIPS-Register hat eine Breite von 32 Bit. Dennoch ist es möglich die Addition von Integerzahlen so zu realisieren, dass man eine doppelte Genauigkeit erreichen kann.
Skizzieren Sie die beiden *minimalen* Abfolgen von MIPS-Befehlen, die notwendig sind die Integeraddition mit doppelter Genauigkeit zu realisieren. Die eine Abfolge soll Überläufe ignorieren, die andere soll im Falle eines Überlaufs eine Unterbrechung provozieren. Nehmen Sie dazu an, dass die eine 64-Bit Ganzzahl (im Zweierkomplement) in den Registern `$t4` und `$t5`, die andere in den Registern `$t6` und `$t7` abgelegt sind. Die Summe der beiden Zahlen soll in die Register `$t2` und `$t3` geschrieben werden. Nehmen Sie weiterhin an, dass das höchstwertige Wort der 64-Bit Ganzzahl in den geraden, das niedrigstwertige Wort in den ungeraden Registern liegt.
Kommentieren Sie jede Zeile ihrer Programmfragmente ausführlich!
Hinweis: Es genügen vier Instruktionen.

| **Überblick über die wichtigsten SPIM Assemblerbefehle** | | |
|---|---|---|
| Befehl | Argumente | Wirkung |
| `add` | `Rd, Rs1, Rs2` | Rd := Rs1 + Rs2 (mit Überlauf) |
| `sub` | `Rd, Rs1, Rs2` | Rd := Rs1 - Rs2 (mit Überlauf) |
| `addu` | `Rd, Rs1, Rs2` | Rd := Rs1 + Rs2 (ohne Überlauf) |
| `subu` | `Rd, Rs1, Rs2` | Rd := Rs1 - Rs2 (ohne Überlauf) |
| `addi` | `Rd, Rs1, Imm` | Rd := Rs1 + Imm |
| `addiu` | `Rd, Rs1, Imm` | Rd := Rs1 + Imm (ohne Überlauf) |
| `div` | `Rd, Rs1, Rs2` | Rd := Rs1 DIV Rs2 |
| `rem` | `Rd, Rs1, Rs2` | Rd := Rs1 MOD Rs2 |
| `mul` | `Rd, Rs1, Rs2` | Rd := Rs1 $\times$ Rs2 |
| `sltu` | `Rd, Rs1, Rs2` | Rd := 1 if Rs1 < Rs2 else 0; Rs1, RS2 sind unsigned integers |
| `b` | `label` | unbedingter Sprung nach `label` |
| `j` | `label` | unbedingter Sprung nach `label` |
| `jal` | `label` | unbed.Sprung nach `label`, Adresse des nächsten Befehls in $ra |
| `jr` | `Rs` | unbedingter Sprung an die Adresse in `Rs` |
| `beq` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $=$ `Rs2` |
| `beqz` | `Rs, label` | Sprung, falls `Rs` $=$ `0` |
| `bne` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $\neq$ `Rs2` |
| `bnez` | `Rs1, label` | Sprung, falls `Rs1` $\neq$ `0` |
| `bge` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $\geq$ `Rs2` |
| `bgeu` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $\geq$ `Rs2` |
| `bgez` | `Rs, label` | Sprung, falls `Rs` $\geq$ `0` |
| `bgt` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $>$ `Rs2` |
| `bgtu` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $>$ `Rs2` |
| `bgtz` | `Rs, label` | Sprung, falls `Rs` $>$ `0` |
| `ble` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $\leq$ `Rs2` |
| `bleu` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $\leq$ `Rs2` |
| `blez` | `Rs, label` | Sprung, falls `Rs` $\leq$ `0` |
| `blt` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $<$ `Rs2` |
| `bltu` | `Rs1, Rs2, label` | Sprung, falls `Rs1` $<$ `Rs2` |
| `bltz` | `Rs, label` | Sprung, falls `Rs` $<$ `0` |
| `not` | `Rd, Rs1` | Rd := $\neg$ Rs1 (bitweise Negation) |
| `and` | `Rd, Rs1, Rs2` | Rd := Rs1 & Rs2 (bitweises UND) |
| `or` | `Rd, Rs1, Rs2` | Rd := Rs1 \| Rs2 (bitweises ODER) |
| `syscall` | | führt Systemfunktion aus |
| `move` | `Rd, Rs` | Rd := Rs |
| `la` | `Rd, label` | Adresse des Labels wird in Rd geladen |
| `lb` | `Rd, Adr` | Rd := MEM[Adr] |
| `lw` | `Rd, Adr` | Rd := MEM[Adr] |
| `li` | `Rd, Imm` | Rd := Imm |
| `sw` | `Rs, Adr` | MEM[Adr] := Rs (Speichere ein Wort) |
| `sh` | `Rs, Adr` | MEM[Adr] MOD $2^{16}$ := Rs (Speichere ein Halbwort) |
| `sb` | `Rs, Adr` | MEM[Adr] MOD 256 := Rs (Speichere ein Byte) |

| Funktion | Code in `$v0` | Funktion | Code in `$v0` |
|---|---|---|---|
| `print_int` | 1 | `read_float` | 6 |
| `print_float` | 2 | `read_double` | 7 |
| `print_double` | 3 | `read_string` | 8 |
| `print_string` | 4 | `sbrk` | 9 |
| `read_int` | 5 | `exit` | 10 |

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 9
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen L

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 29. Juni – 3. Juli 2020

## **Aufgabe H19:** Addiernetze in arithmetisch-logischen Einheiten

(11 Pkt.)

a. Nehmen Sie einen Carry-Look-Ahead-Addierer mit einer Größe der Bit-Gruppen von $g = 3$ an. Leiten Sie den logischen Ausdruck her, mit dem der ausgehende Übertrag $U_{out}$ bereits vor Abarbeitung des Addiernetzes bestimmt werden kann. Bezeichnen Sie dabei die beiden eingehenden Binärzahlen als $x_2x_1x_0$ und $y_2y_1y_0$ und den eingehenden Übertrag als $U_{in}$.

b. Zeichnen Sie das Schaltnetz für einen Carry-Look-Ahead-Addierer für eine Größe von Bit-Gruppen von $g = 3$. Vorkommende Volladdierer können dabei durch ihr entsprechendes Schaltsymbol dargestellt werden. Hierbei können Sie annehmen, dass AND-Gatter und OR-Gatter zur Verfügung stehen, die mehr als zwei Eingaben gleichzeitig verarbeiten können. Achten Sie darauf, die Verbindung von zwei Leitungen explizit zu kennzeichnen.

c. Gehen Sie nun von der Addition zweier Dualzahlen der Länge 6-Bit aus. Berechnen Sie die Ausführungsdauer der Addition für Carry-Look-Ahead-Addierer mit einer Größe der Bit-Gruppen von $g = 3$, d.h. es werden zwei Carry-Look-Ahead-Addierer aus den vorhergehenden Aufgabenteilen hintereinander geschaltet. Berechnen Sie zudem die Ausführungsdauer für ein angenommenes Ripple-Carry-Addiernetz, das zwei 6-stellige Dualzahlen addieren kann. Nehmen Sie hierbei an, dass ein Volladdierer eine Verzögerung von 70 psec, ein AND-Gatter und OR-Gatter jeweils eine Verzögerung von 10 psec verursachen. AND-Gatter und OR-Gatter mit mehr als zwei Eingängen sollen ebenfalls mit einer Verzögerung von 10 psec veranschlagt werden.

## **Aufgabe H20:** Schaltung für Successor-Funktion

(8 Pkt.)

In dieser Aufgabe sollen Sie das Schaltnetz für eine Binärschaltung entwerfen, welche die Successor-Funktion für 2-Bit-Zahlen realisiert.

Sei $i \in \{0, \ldots, 3\}$ eine Dezimalzahl und $d(i)$ die zweistellige Dualdarstellung von $i$. Die Successor-Funktion soll die Funktion

$$f_{succ}(d(i)) = d(i+1) \bmod 4 \qquad \textbf{mit} \qquad f_{succ} : B^2 \to B^2$$

realisieren.

Neben zwei Dateneingänge $x_0$ und $x_1$ sowie zwei Datenausgänge $y_0$ und $y_1$ soll die Schaltung einen Steuereingang $s$ besitzen. Nur wenn gilt $s = 1$ soll die Schaltung den Wert von $f_{succ}(d(i))$ als Ergebnis an den Datenausgängen liefern. Für $s = 0$ soll die Schaltung die Dateneingänge unverändert an die Datenausgängen weiterleiten.

a. Stellen Sie zunächst die Funktionstafel für die gewünschte Schaltung auf.

b. Leiten Sie von der Funktionstafel die beiden Schaltfunktionen $f_{y_0}$ und $f_{y_1}$ für die Datenausgänge $y_0$ und $y_1$ ab. Minimieren Sie beide Funktion so weit wie möglich.

c. Oft ist es billiger eine Schaltung aus komplexeren schon existierenden Bausteinen zusammen zu setzen, anstatt sie von Grund auf neu zu konstruieren.

   Gehen Sie davon aus, dass Ihnen aus Kostengründen nur Halbaddierer zur Verfügung stehen und konstruieren Sie die Schaltung mittels Halbaddierer.

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 10
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen M

**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 6. – 10. Juli 2020

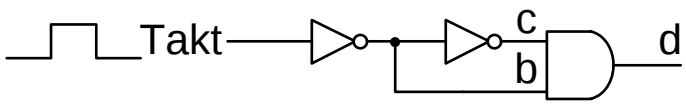
## **Aufgabe Ü21:** Latch- bzw. Flip-Flop-Schaltungen

(– Pkt.)

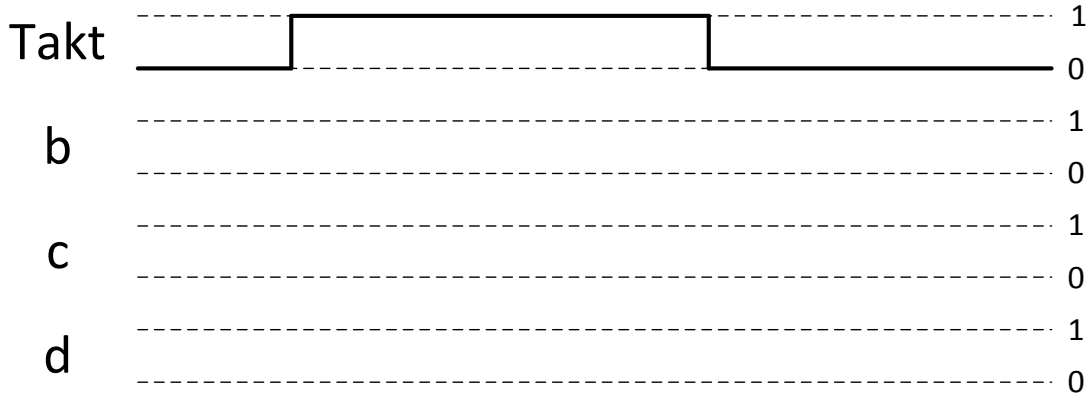
Bearbeiten Sie die folgenden Teilaufgaben zum Thema Schaltwerke:

a. Zeichnen Sie das Schaltnetz eines getakteten SR-Latch, indem Sie folgende Vorlage ergänzen. Verwenden Sie dabei ausschließlich **NOR-Gatter** und Leitungen.

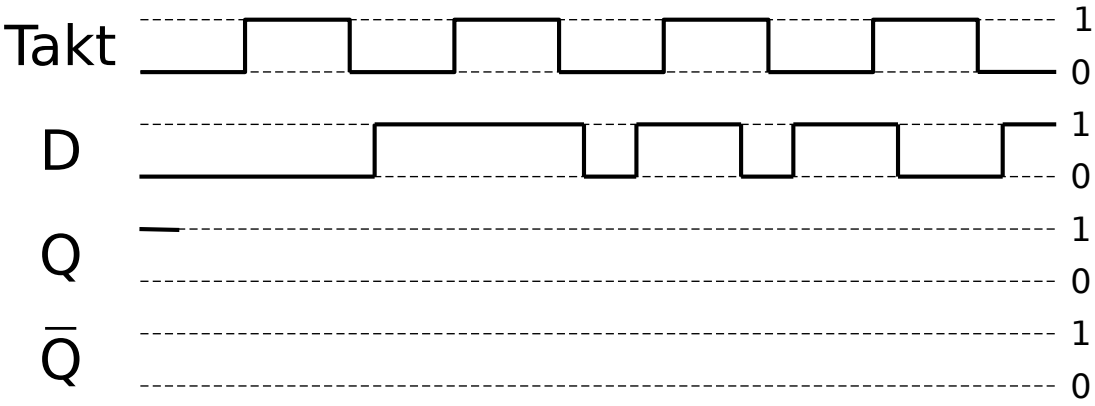

b. Gegeben sei folgendes Schaltnetz eine Impulsgenerators.




Ergänzen Sie folgende Vorlage zu einem Impulsdiagramm für die Ausschnitte b, c, d basierend auf dem eingezeichneten Takt. Dabei verursacht jedes Gatter eine kurze aber nicht vernachlässigbare Verzögerung des Signals. Insbesondere ist die Verzögerung der `NOT`-Gatter größer als die des `AND`-Gatters.



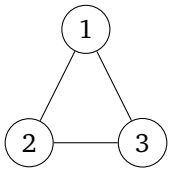

c. Ergänzen Sie nun die folgende Vorlage zum Impulsdiagramm eines D-Flip-Flops mit dem Impulsgenerator aus der vorherigen Teilaufgabe b). Das D-Flip-Flop verfügt über die Ausgänge Q und $\bar{Q}$. Gehen Sie zur Vereinfachung davon aus, dass sich die Pegel von Q und $\bar{Q}$ des Bausteins ohne Zeitverzögerung in Abhängigkeit vom Takt und dem Signal D ändern.




## **Aufgabe Ü22:** Graph Coloring mittels Quantenannealing

(– Pkt.)

Sei folgender Graph gegeben, dessen Knoten $\{1, 2, 3\}$ mit den Farben Rot, Grün und Blau $\{R, G, B\}$ gefärbt werden sollen, so dass keine zwei benachbarten Knoten (mit einer Kante verbunden) die gleiche Farbe tragen.




Füllen Sie folgenden Matrix mit den Zahlenwerten 0 und 5, je nachdem, wie günstig eine Zustandskombination zu bewerten ist, so dass die Optimierung (Minimierung) mittels Quantenannealing stattfinden kann.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1R | 1G | 1B | 2R | 2G | 2B | 3R | 3G | 3B |
| | 1R | −1 | | | | | | | | |
| | 1G | | −1 | | | | | | | |
| | 1B | | | −1 | | | | | | |
| | 2R | | | | −1 | | | | | |
| | 2G | | | | | −1 | | | | |
| | 2B | | | | | | −1 | | | |
| | 3R | | | | | | | −1 | | |
| | 3G | | | | | | | | −1 | |
| | 3B | | | | | | | | | −1 |

Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

# Übungsblatt 11
# Rechnerarchitektur im SoSe 2020

### Zu den Modulen N

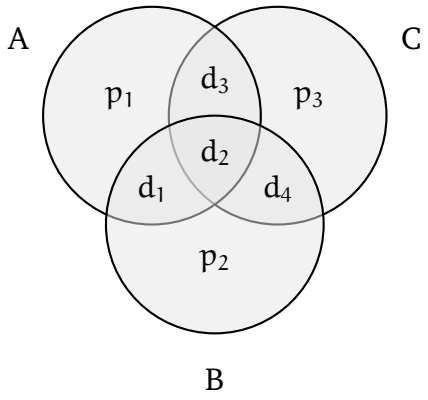
**Besprechung:** Besprechung der Übungsaufgaben in den Übungsgruppen vom 13. – 17. Juli 2020

## **Aufgabe Ü23:** Fehlererkennung und -korrektur

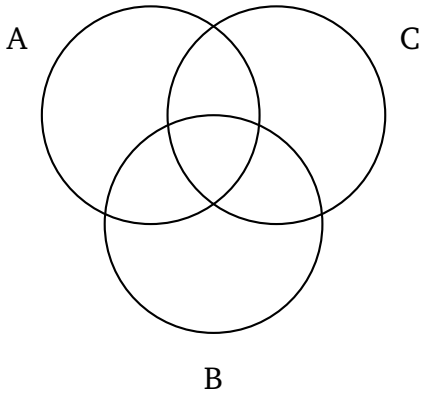
(– Pkt.)

Zum Schutz vor Speicherfehlern werden sogenannte Codes zur Fehlererkennung und zur Fehlerkorrektur eingesetzt. Bearbeiten Sie die folgenden Teilaufgaben:

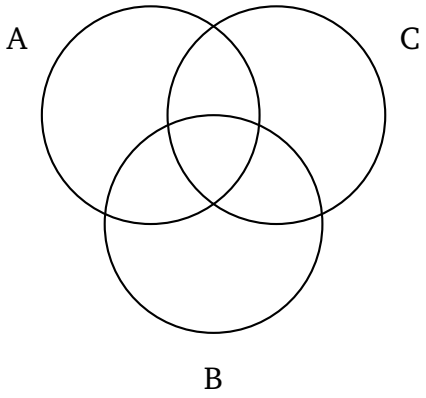
a. Wir gehen von folgender Struktur der Code-Wörter $d_1 d_2 d_3 d_4 p_1 p_2 p_3$ aus. Wobei $d_i (i \in \{1,2,3,4\})$ für das jeweilige Datenbit und $p_j (j \in \{1,2,3\})$ für das jeweilige Prüf- bzw. Paritätsbit steht. Die Paritätsbits zur Fehlererkennung bzw. Fehlerkorrektur für ein Datenwort $d_1 d_2 d_3 d_4$ können anschaulich mit Hilfe eines Venn-Diagramms berechnet werden, in welchem die Bits wie folgt angeordnet sind:




(i) Berechnen Sie unter Verwendung des folgenden Venn-Diagramms die Prüfbits für das Datenwort **`1110`**. Verwenden Sie dazu **gerade Parität**. Tragen Sie zunächst die Datenbits in die für die Berechnung sinnvollen (Schnitt-)Mengen ein.




(ii) Gehen Sie nun davon aus, dass Sie ein mit dem zuvor beschriebenen Code codiertes Code-Wort `0011010` empfangen haben. Es wurde **gerade Parität** verwendet. Handelt es sich um ein gültiges Codewort? Falls nein, treffen Sie eine Aussage darüber, an welcher/welchen Stelle/Stellen mutmaßlich (ein) Bitfehler aufgetreten ist/sind. Verwenden Sie zur Berechnung das folgende Venn-Diagramm. Korrigieren Sie (falls möglich/nötig) den/die Fehler **innerhalb** des Venn-Diagramms und **geben Sie das** (ggf. korrigierte) **4-Bit Datenwort an**.

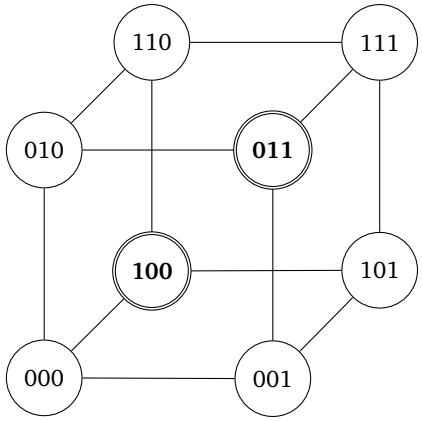



(iii) Wie ist beim vorliegenden Fehlererkennungs- bzw. Fehlerkorrekturcode der Overhead der Prüfbits bzgl. der Speicherbits in Prozent (%)?

b. In dieser Teilaufgabe sei ein Fehlerkorrekturcode gegeben, der nur aus **zwei gültigen Codewörtern** besteht:

- `100`
- `011`

In der Abbildung rechts sind alle möglichen Kombinationen von 3 Bit eingezeichnet und die gültigen Codewörter markiert.




Bearbeiten Sie dazu die folgenden Aufgaben:

(i) Wie viele Einzelbitfehler können mit dem vorliegenden Code in jedem Fall **erkannt** werden?

(ii) Wie viele Einzelbitfehler können mit dem vorliegenden Code in jedem Fall **korrigiert** werden?

## **Aufgabe Ü24:** Hamming Codes

(– Pkt.)

Übertragung von Daten über physische Kanäle (Kabel etc.) ist fehleranfällig. Als Schutz vor solchen Fehlern setzen die meisten Speicher Codes für die Fehlererkennung und ggf. auch zur Fehlerkorrektur ein. Lesen Sie sich im Vorlesungsskript das Kapitel 14.4 zur „Fehlererkennung und -korrektur“ (S. 173-176) aufmerksam durch und bearbeiten Sie die folgenden Aufgaben:

a. Kodieren Sie die folgenden 16-Bit Daten in 21-Bit Hamming Code. Verwenden Sie dazu gerade Parität.

(i) 1100 1010 0000 0100

b. Dekodieren Sie das folgende 21-Bit Codewort. Wenn Sie Fehler enthalten, identifizieren Sie das fehlerhafte Bit und korrigieren Sie den Fehler. Das Ergebnis muss ein 16-Bit Datenwort sein.

(i) 0110 1011 0110 1100 1101 1