**PHILIPPS-UNIVERSITÄT MARBURG**
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen

D-35032 Marburg
Hans-Meerwein-Straße
Lahnberge

# Klausur zur „Technischen Informatik I“, WS 2000/01

**29. Januar 2001**

---

**Hinweise:**

- **Bearbeitungszeit:** 2 Stunden
  **Gesamtpunktzahl:** 100 Punkte
  Zum Bestehen der Klausur sind **40 Punkte** erforderlich.
- Hilfsmittel sind nicht erlaubt.
- Jede Aufgabe ist auf einem eigenen Blatt zu bearbeiten.
- Alle ausgehändigten Blätter sind zurückzugeben und, soweit beschrieben, mit Namen zu versehen. Auch der Klausurtext ist abzugeben.

**Viel Erfolg!**

---

**Name: .....**

**Mat.-Nr.: .....Studienfach: .....**

| Aufgabe | max. Punktzahl | erreichte Punktzahl | korrigiert von |
|---|---|---|---|
| 1 | 8 | | |
| 2 | 15 | | |
| 3 | 10 | | |
| 4 | 12 | | |
| 5 | 12 | | |
| 6 | 12 | | |
| 7 | 13 | | |
| 8 | 8 | | |
| 9 | 10 | | |
| Summe | 100 | | |

| Note: |
|---|
| |

1. von Neumannsches Rechnerkonzept — 8 Punkte

   (a) Nennen Sie die Basiskomponenten eines von Neumann-Rechners. 2

   (b) Was versteht man unter dem „von Neumannschen Flaschenhals“? Begründen Sie Ihre Antwort. 2

   (c) Erläutern Sie eine Maßnahme, die zur Behebung des Flaschenhalses getroffen wurde. Begründen Sie, weshalb diese Maßnahme wirkt. 4

2. Schaltfunktionen — 15 Punkte

   Gegeben sei die folgende Schaltfunktion

   $$f(x_1, x_2, x_3, x_4) = x_1' x_2' x_3' x_4' + x_1 x_2 x_3 x_4 + x_2 x_3' x_4 + x_1 x_2' x_3 x_4' + x_1' x_2 x_3 x_4$$

   (a) Ist dies eine disjunktive Normalform? Falls nein, erklären Sie, warum nicht und geben Sie die korrekte disjunktive Normalform dieser Funktion an. 3

   (b) Minimieren Sie die obige Darstellung so weit wie möglich. 3

   (c) Ist es für die Funktion $f$ günstiger, die disjunktive oder die konjunktive Normalform anzugeben? Begründen Sie Ihre Antwort. 2

   (d) Stellen Sie die Funktion als PLA dar. 3

   (e) Bestimmen Sie alle Primimplikanten der zu $f$ komplementären Funktion. 4

3. Boolesche Algebra — 10 Punkte

   Sei $\mathcal{B} = \langle B; +, *, '; 0, 1\rangle$ eine beliebige Boolesche Algebra.

   Beweisen oder widerlegen Sie, dass für beliebige $x, y, z \in B$ die folgenden Gleichungen gelten. Benennen Sie gegebenenfalls die verwendeten Gesetze der Booleschen Algebra.

   (a) $x' * y + y' * x = (x' * y' + x * y)'$ 5

   (b) $x' * z + y * z' + x * y' = x * z' + y' * z + x' * y$ 5

4. Vollsubtrahierer — 12 Punkte

   Ein Vollsubtrahierer ist eine Schaltung, die eine 1-Bit-Subtraktion durchführt. Dabei soll ein Übertrag aus der vorigen Stelle als Eingabe verarbeitet und neben dem Subtraktionsergebnis der Übertrag für die nachfolgende Stelle bestimmt werden.

   (a) Geben Sie die Wertetabelle eines Vollsubtrahierers an. 2

   (b) Geben Sie Termdarstellungen für die Ausgänge des Vollsubtrahierers an und zeichnen Sie eine entsprechende Schaltung mit Und-, Oder-, Negations- oder XOR-Gattern als Grundbausteinen. 6

   (c) Erweitern Sie Ihre Schaltung durch Hinzunahme möglichst weniger Grundbausteine zu einem integrierten Volladdierer/-subtrahierer-Baustein mit einem zusätzlichen Moduseingang $m$. Falls $m = 0$ soll der Baustein einen Volladdierer realisieren, falls $m = 1$ einen Vollsubtrahierer. 4

5. T-Flipflop — 12 Punkte

Das T-Flipflop ('Toggle'-Flipflop) hat einen einzelnen Dateneingang $T$ und ist dadurch charakterisiert, daß $T = 1$ dazu führt, dass das Flipflop seinen Zustand $Q$ beim Taktübergang $0 \rightarrow 1$ wechselt, während $T = 0$ den Zustand unverändert läßt.

(a) Beschreiben Sie das Verhalten eines T-Flipflops durch eine Zustandstabelle. 2

(b) Leiten Sie aus der Zustandstabelle eine Termdarstellung des Flipflop-Verhaltens ab. 3

(c) Welches Phänomen tritt bei einer direkten Umsetzung der Termdarstellung in eine Schaltung mit Und-, Oder- und Negationsgattern auf? 1

(d) Entwickeln Sie eine stabile Schaltung für das T-Flipflop unter Verwendung von RS-Flipflops. Der Zustandswechsel soll beim Taktwechsel $0 \rightarrow 1$ erfolgen. 6

6. Ringzähler — 12 Punkte

Entwerfen Sie ein Schaltwerk für einen 3-Bit-Ringzähler mit einem *increment*-Eingang. Wenn an diesem Eingang eine Eins anliegt, soll der gespeicherte Wert (3-Bit-Zahl) ums Eins erhöht werden, wobei der Wert 111 zu 000 geändert wird. Wenn am increment-Eingang eine Null anliegt, soll der gespeicherte Wert konstant bleiben. Die Schaltung soll sowohl mit einem PLA als auch mit einzelnen Gattern realisiert werden.

7. Zahldarstellungen und Arithmetik — 13 Punkte

(a) Führen Sie folgende Operationen im Zweierkomplement aus: 6

$$10100 + 01111, 00111 + 01110, 11010 + 10101$$

Geben Sie jeweils an, ob ein Übertrag (carry), Überlauf (Overflow) oder beides auftritt. Führen Sie zur Kontrolle die entsprechenden Berechnungen auch im Dezimalsystem durch. Interpretieren Sie die Ergebnisse.

(b) Sei $n \geq 1$. Für die Multiplikation zweier $n$-Bit langer vorzeichenloser Binärzahlen steht üblicherweise ein Ergebnisregister der Länge $2n$ zur Verfügung. Diskutieren Sie die Überlaufproblematik. 2

(c) Wie groß ist der betragsgrößte Wert bei der Multiplikation von zwei Zweierkomplementzahlen der Länge $n + 1$? 2

(d) Erläutern Sie, weshalb der IEEE-Gleitkommastandard eine verschobene Exponentdarstellung (biased notation) verwendet. 3

8. Mikroprogrammierte CPU — 8 Punkte

Gegeben sei der folgende Mikrobefehl zur Realisierung des unbedingten Sprungbefehls `JMP <n>` im Rahmen des Load-Increment-Execute-Zyklus (LIE):

```
Segment, Opcode: 02H
Mnemonik: JMP <n>
008H: 00 00 0001 0000100 00000000 00000000 10000000 000100 01 00
Bemerkung:
JMP <n>

Phase 1: Lesen, MDR -> Y
Phase 2: Z := Y
Phase 3: Z -> R0
Go LIE
```

(a) Zerlegen Sie den Mikrobefehl in seine Komponenten und ordnen Sie die in der Befehlserklärung genannten Datenbewegungen und Aktionen diesen Komponenten zu. 4

(b) Welche Schritte sind in einem LIE-Zyklus notwendig, in dem der obige Befehl ausgeführt werden kann? Genügt hierzu ein einzelner Mikrobefehl? 4

## 9. Assemblerprogrammierung

10 Punkte

(a) Wie kann man das nachfolgende (Pseudo-)Assemblercodestück in Pascal-ähnlicher Syntax ausdrücken? Was wird in Register `dx` berechnet? 3

```
        mov cx, 1
        mov dx, 0
        jmp marke2
marke1: inc dx
        div bx, 2
marke2: cmp bx, cx
        jg marke1;  jump greater
marke3: ...
```

(b) Realisieren Sie das folgende Pascal-Konstrukt in Assembler: 3

```
if ((a OR b) AND c)
then <Anweisung1>
else <Anweisung2>
```

`a`, `b` und `c` seien Variablen mit den Speicheradressen `ma`, `mb` und `mc`.

(c) Schreiben Sie ein Assembler-Unterprogramm, welches die maximale Komponente eines Vektors bestimmt. 4

Der Vektor soll aus 100 vorzeichenlosen Zahlen der Länge 1 Byte bestehen. Seine Anfangsadresse soll durch den symbolischen Namen VEKTOR bezeichnet sein. Das Ergebnis soll im Speicher unter der symbolischen Adresse MAXELEM abgelegt werden. Geben Sie die Deklaration der Speicherplätze an.

Nach dem Aufruf des Unterprogramms sollen alle Register dieselben Werte wie vor dem Aufruf enthalten. Versehen Sie Ihr Programm mit Kommentaren. Beachten Sie, dass zwei Speicheroperanden in Assemblerbefehlen nicht erlaubt sind.