Übungen zu Vorlesung
Praktische Informatik I
Wintersemester 2002/03

Lektüre 1

Aussagenlogik, Mengen, Relationen, Funktionen

Lektion 1: Aussagenlogik

Stichworte:

- Aussagen
- Operatoren $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\neg$
- Gleichheit von Aussagen
- Wahrheitswerttafeln zur Definition von Operatoren
- Wahrheitswerttafeln zur Auswertung von Aussagen
- Quantoren $\forall$, $\exists$

Beispiel:

$$A \wedge B \Leftrightarrow B \wedge A$$

Online-Übung: Zeigen Sie mit Hilfe von Wahrheitswerttafeln, daß die folgenden Aussagen allgemeingültig sind:

$$\begin{array}{rcl} A \wedge (B \vee C) & \Leftrightarrow & (A \wedge B) \vee (A \wedge C) \\ \neg(A \wedge B) & \Leftrightarrow & \neg A \vee \neg B \\ A \wedge (A \Rightarrow B) & \Rightarrow & B \end{array}$$

Lektion 2: Mengenlehre

Stichworte:

- Menge
  - Aufzählungen
  - $\{x \in A \mid P(x)\}$
- Mengenoperationen $\cup$, $\cap$, Mengenkomplement, $\subseteq$, $\setminus$, Mengengleichheit
  - Erläuterung mit Venn-Diagrammen
  - Zurückführen auf Aussagenlogik
- Teilmenge, Potenzmenge
- Kreuzprodukt von Mengen (als Vorbereitung auf Funktionen)

Beispiele:

- $A = \{1, 2, 3, 4\}$
- $B = \{x \in \{1, \ldots, 10\} \mid x \; gerade\}$
- $C = A \cap B$
- $D = A \backslash B$
- $E = (A \cup B) \backslash C$

Online-Übung: Seien A, B und C beliebige Mengen. Beweisen Sie

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

Hinweis: zurückführen auf Aussagenlogik

Lektion 3: Relationen und Funktionen

Stichworte:

- binäre Relation als Menge von Paaren ("Pfeil-Diagramme")
- Funktion informell (Abbildung mit Urbild und Bildbereich)
- Funktionen $M_1 \times \ldots \times M_n \rightarrow M_{n+1}$; Operationen sind Funktionen (gemeinsames Erarbeiten der Funktionalität der Logik- und Mengenoperatoren)
- Komposition von Funktionen ("Pfeil-Diagramme")

Beispiele:

- $\leq$ auf $\{1, \ldots, 5\}$
- Teilbarkeitsrelation auf der Menge der Zahlen $\{1, \ldots, 16\}$

Rechneraufbau und Rechnerstrukturen

nach Oberschelp/Vossen

Lektion 4: Arbeitsweise der CPU

Wiederholung und Ergänzung zur Vorlesung
Stichworte:

- CPU-Komponenten (Bild S. 232)
  - Datenprozessor - Befehlsprozessor
  - Adressbus - Datenbus
  - Decodierer/Steuerwerk
  - IR

  - MAR
  - PC
  - MR - L - A
  - ALU
  - MBR

- "fetch-execution" Zyklus
- Maschinencode - Mikroprogrammierung

Dazu soll folgender Mikrocode eingeführt werden:

MAR	$\leftarrow$	PC	MAR erhält Inhalt von PC (Adresse)
MBR	$\leftarrow$	<MAR>	MBR erhält Inhalt der Speicheradresse, die im MAR steht
MBR	$\leftarrow$	A	lade Inhalt des Akku in MBR
MBR	$\leftarrow$	MAR	lade Inhalt des MAR (Adresse) in MBR
<MAR>	$\leftarrow$	MBR	Speichere Inhalt von MBR in die Adresse auf die MAR zeigt
IR	$\leftarrow$	MBR	Instruktions-Register erhält Befehlt aus MBR
PC	$\leftarrow$	PC+n	PC erhöhen um n
PC	$\leftarrow$	MBR	PC auf Adresse im MBR setzen (Absolutadressierung)
PC	$\leftarrow$	MAR	PC auf Adresse im MAR setzen
MAR	$\leftarrow$	MAR+n	MAR auf Inhalt $+n$ setzen
A	$\leftarrow$	A + MBR	Akku-Berechnung, Addition
A	$\leftarrow$	-A	unäres Minus
A	$\leftarrow$	$\neg$A	logische Negation
MR	$\leftarrow$	A * MR	Multiplikation
A	$\leftarrow$	MBR	Konstante in Akku laden
A	$\leftarrow$	A < 0	logischer Vergleich mit Konstanten, Ergebnis im Akku
MAR	$\leftarrow$	MBR	Umladen von Adressen

Anmerkung: MR ist üblicherweise eine Erweiterung des Akku, d.h. ein Schreiben auf MR zerstört den Inhalt von L und A.

Beispiel:

Die Umsetzung des Maschinenbefehls `LOAD x`, mit `x=20` in die Mikroinstruktionen; PC soll auf 0 stehen;

Maschinenprogramm:

0	LOAD
1	20
...	
20	10

MAR	←	PC	MAR == 0
MBR	←	<MAR>	MBR == Bitcode von LOAD
IR	←	MBR	IR == Bitcode von LOAD
Dekodieren			führt zu:
MAR	←	MAR + 1	MAR == 1
MBR	←	<MAR>	MBR == 20
MAR	←	MBR	MAR == 20
MBR	←	<MAR>	MBR == 10
A	←	MBR	A == 10
PC	←	PC + 2	PC == 2

dabei bezeichnet <X> den Inhalt der Zelle an der Adresse, die in X steht.

Maschinencode:
Wie in der Vorlesung eingeführt.

LOAD adr	n	LOAD	Lade den Inhalt der Speicherdresse adr in den Akku A
	n+1	<adr>	
STORE adr	n	STORE	Speichere den Inhalt des Akku A im RAM an Adresse adr
	n+1	<adr>	
NEG	n	NEG	Negiere den Inhalt des Akku A
ADD adr	n	ADD	Addiere den Speicherinhalt der Adresse adr zum Inhalt des Akku A und platziere das Ergebnis wieder in A.
	n+1	<adr>	
MULT adr	n	MULT	Multipliziere den Speicherinhalt der Adresse adr zum Inhalt des Akku A und platziere das Ergebnis im Multiplikationsregister MR
	n+1	<adr>	
JMPNEG adr	n	JMPNEG	Wenn der Inhalt dess Akku A negativ ist, springe zur Adresse adr und lies den dort gespeicherten BEFEHL (da dürfen keine Daten stehen!) Andernfalls mache weiter mit dem nächsten Befehl (der 2 Worte hinter dem JMPNEG-Befehl steht)
	n+1	<adr>	
	...		
	<adr>	BEFEHL	
HALT	n	HALT	Programm hält an

Online-Übung:

a) Implementiere ADD x im Mikrocode,
Wirkung von ADD x: A == A + <x>

b) Implementiere SWAP x y im Maschinencode (mittels load und store),
Wirkung von SWAP x y: <x>,<y> := <y>,<x>