PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen, Dipl.-Inform. J. Beringer

D-35032 Marburg
Hans Meerwein Straße
Lahnberge
4. Januar 2005

Übungen zur „Technischen Informatik I“, WS 2004/05

Nr. 10, Abgabe: Dienstag, 11. Januar vor der Vorlesung

A. Hausaufgabe

52. Mikroprogrammiertes Maschinenspracheprogramm — 12 Punkte

Entwickeln Sie analog zur Vorgehensweise der Vorlesung ein mikroprogrammiertes Maschinenspracheprogramm für das folgende Programm (in Pascal-Syntax):

```
VAR n, m: Integer; VAR b, i, aux: Integer;
BEGIN
  read(n);    read(m);  b := n;    i := 2;   aux := n-1;
  WHILE i <= m DO
    BEGIN
       b := (b*aux) div i;  i := i+1;  aux := aux-1;
    END;  write(b)
END.
```

(a) Was berechnet das Programm? / 1

(b) Übersetzen Sie das Programm unter Verwendung der in der folgenden Tabelle angegebenen Maschinenbefehle in ein Maschinenprogramm. / 3

	Mnemonik	Beschreibung	Opcode
⋆	CMP_IND_B	Vergleiche Register R3 (INDEX) und R2 (B)	0Eh
	DEC_AUX	Dekrementiere Inhalt von R4 (AUX) um 1	0Ah
	DIV_A_IND	Dividiere Inhalt von R1 (A) durch R3 (INDEX)	14h
	INC_INDEX	Erhöhe Inhalt von R3 (INDEX) um 1	0Ch
	JMP marke	Springe zu Adresse marke	0Fh
⋆	JGREATER marke	Springe zu Adresse marke, falls Flag > gesetzt	10h
	MOV_A_TO_AUX	Kopiere Inhalt von R1 (A) nach R4 (AUX)	08h
	MUL_A_AUX	Multipliziere Inhalt von R1 (A) mit R4 (AUX)	12h
	READ_A adr	Lade Inhalt der Adresse adr in Register R1 (A)	02h
	READ_B adr	Lade Inhalt der Adresse adr in Register R2 (B)	03h
	LIT_IND n	Lade Parameter n in Register R3 (IND)	06h
	STOP	Stoppe die Ausführung	FFh
⋆	STORE_A adr	Speichere Inhalt von R1 (A) an Adresse adr	04h

(c) Implementieren Sie die in der Tabelle mit ⋆ markierten Befehle im Mikrocodesimulator. / 6

Auf der Vorlesungsseite liegt eine Datei Aufgabe52.ROM, in der die übrigen Befehle und der Load-Increment-Execute-Zyklus bereits implementiert sind. Ergänzen Sie diese Datei entsprechend.

(d) Übertragen Sie Ihr Maschinenspracheprogramm in eine Hexadezimalziffernfolge, die in den Hauptspeicher geladen und im Mikrocodesimulator ausgeführt werden kann. Den Variablen n, m seien die Speicheradressen F0h und F4h zugeordnet. Der Ausgabewert b soll in Adresse F8h geschrieben werden. / 2

B. Mündliche Aufgaben

53. Ein-Befehlsrechner

Gegeben sei ein Rechner mit dem Namen SIC (Single Instruction Computer), der mit einem einzigen Befehl auskommt. Dieser Befehl heißt `sbn` (subtract and branch if negative). Er erhält drei Operanden, in denen jeweils Speicheradressen stehen:

```
sbn a, b, c          # a := a-b; if (a < 0) goto c
```

Das folgende Beispiel zeigt, wie eine Zahl von a nach b kopiert werden kann. Dabei steht `.+1` für die jeweils nachfolgende Adresse, so dass die Abarbeitung hier immer mit der nächsten Instruktion weitergeht, unabhängig davon, ob das Ergebnis negativ ist oder nicht. Die Adresse temp wird für Zwischenergebnisse verwendet.

```
start : sbn temp, temp, .+1          # temp := 0
        sbn temp, a, .+1             # temp := -a
        sbn b, b, .+1                # b := 0
        sbn b, temp, .+1             # b := -(-a)
```

(a) Entwickeln Sie ein SIC-Programm, das die Inhalte von `a` und `b` vertauscht.

(b) Schreiben Sie ein SIC-Programm zur Addition der Inhalte von `a` und `b`, das die Summe in `a` speichert und den Inhalt von `b` unverändert lässt.

54. Kellermaschine

Betrachten Sie eine Kellermaschine, in der folgende Befehle zur Verfügung stehen:

`ADD` `AND`
`SUB` `OR`
`MULT` `NOT`
`DIV`
	`PUSH`< *adresse* >
	`POP`< *adresse* >
		`JT`< *label* >
		`JF`< *label* >
		`COMP`

Die beiden oberen (bzw. bei `NOT` nur das oberste) Kellerelement(e) werden verknüpft und durch das Ergebnis ersetzt.	`PUSH` schreibt ein Speicherelement auf den Keller. `POP` nimmt das oberste Element vom Keller und speichert es.	`JT`/`JF` führt zu einem Sprung, falls das oberste Kellerelement `true`/`false` ist. Dieses wird dabei entfernt. `COMP` vergleicht die beiden oberen Kellerelemente und ersetzt diese durch `true`/`false` je nach Vergleichsergebnis.

Entwickeln Sie für die folgenden Anweisungen möglichst kurzen Stackcode:

(a) `Z:= (A + B*C) - D`

(b) `if A XOR B` $\neq$ `C then C := D else goto E`