Rückblick

Von-Neumann-Rechner
Das theoretische Modell eines universellen, programmierbaren digitalen Rechners besteht aus Steuerwerk, Rechenwerk, Speicherwerk und Ein-/Ausgabewerk. Die Werke sind über ein getaktetes Bus-System miteinander verbunden. Daten und Befehle sind in binärer Form in den fortlaufend nummerierten Zellen des Speicherwerks abgelegt.

Registermaschine
Eine theoretische Registermaschine besteht aus einem Programm im Speicher, aus Registern als Speicherzellen für Daten sowie einem Befehlszähler, der die Speicheradresse des nächsten zu bearbeitenden Befehls enthält. Der Akkumulator ist ein spezielles Register zur Speicherung des letzten Berechnungsergebnisses.
Zur Programmierung verwendet man *Assembler*-Sprachen mit sehr kleinem Befehlsvorrat. Dieser umfasst
- Transportbefehle (`LOAD`, `DLOAD`, `STORE`),
- arithmetische Befehle (`ADD`, `SUB`, `MULT` …),
- Sprungbefehle mit und ohne Bedingung (`JGE`, `JUMP` …),
- logische Verknüpfungen (`AND`, `OR`, Negierung, Invertierung …).

Reale Maschinen
Reale Mikroprozessoren rechnen ebenfalls in Registern. In der Regel sind hierbei die Register der Operanden und des Ergebnisses frei wählbar.
Die gesamte Abarbeitung eines Maschinenbefehls (der Befehlszyklus) besteht dabei grob aus zwei Phasen: Befehl holen und Befehl ausführen.
- In der Befehlsholphase erfolgt der Speicherzugriff auf die vom Befehlszähler angezeigte Adresse. Der Befehl wird in das Befehlsregister des Prozessors geladen und der Befehlszähler aktualisiert.
- Der Befehl wird decodiert. Anschließend werden die notwendigen Operanden ermittelt und aus dem Speicher geladen. Der Befehl wird ausgeführt. Nötigenfalls werden Ergebnisse im Speicher abgelegt.

Algorithmische Strukturelemente
Algorithmen, die bedingte Anweisungen, Alternativen oder Wiederholungen enthalten, können allein mithilfe von Vergleichen und Sprungbefehlen implementiert werden.
Bei bedingten Anweisungen und Alternativen wird eine Sequenz übersprungen.
Bei Wiederholungen erfolgt ein Rücksprung zur erneuten Prüfung der Bedingung.

Das folgende Assembler*-Programm berechnet die dritte Potenz von 8.*

Assembler

```
1:  DLOAD 8       --Basis 8
2:  STORE 1       --Potenz p in R1
3:  STORE 2       --Basis b in R2
4:  DLOAD 1
5:  STORE 3       --Konstante 1 in R3
6:  DLOAD 3
7:  STORE 4       --Exponent e = 3 in R4
8:  JEQ 21        --falls e = 0
9:  SUB 3
10: JEQ 19        --falls e = 1
11: STORE 4       --e = e - 1
12: LOAD 1
13: MULT 2
14: STORE 1       --p = p * b
15: LOAD 4
16: SUB 3
17: STORE 4       --e = e - 1
18: JGT 12        --falls e > 0
19: LOAD 1        --Ergebnis in A
20: JUMP 22
21: LOAD 3        --1 in A für e = 0
22: END
```

Die folgende Tabelle zeigt, wie der Akkumulator und die einzelnen Register belegt werden:

	A	R1	R2	R3	R4	BZ
						1
1: DLOAD 8	8					2
2: STORE 1		8				3
3: STORE 2			8			4
4: DLOAD 1	1					5
5: STORE 3				1		6
6: DLOAD 3	3					7
7: STORE 4					3	8
8: JEQ 21						9
9: SUB 3	2					10
10: JEQ 19						11
11: STORE 4					2	12
12: LOAD 1	8					13
13: MULT 2	64					14
14: STORE 1		64				15
15: LOAD 4	2					16
16: SUB 3	1					17
17: STORE 4					1	18
18: JGT 12						12
12: LOAD 1	64					13
13: MULT 2	512					14
14: STORE 1		512				15
15: LOAD 4	1					16
16: SUB 3	0					17
17: STORE 4					0	18
18: JGT 12						19
19: LOAD 1	512					20
20: JUMP 22						22
22: END						23

© Ernst Klett Verlag GmbH, Stuttgart 2010 | www.klett.de | Alle Rechte vorbehalten.
Informatik 5 – Jahrgangsstufe 12
ISBN 978-3-12-731068-9

Training

1 Anhalteweg

In der Fahrschule lernt man folgende zwei Faustregeln:
- „Reaktionsweg (in m) = Geschwindigkeit $\left(\text{in } \frac{\text{km}}{\text{h}}\right)$ geteilt durch 10 mal 3“
- „Bremsweg (in m) = Geschwindigkeit $\left(\text{in } \frac{\text{km}}{\text{h}}\right)$ geteilt durch 10 und das Ergebnis mit sich selbst malgenommen“

a) Schreiben Sie ein *Assembler*-Programm, das aus der Geschwindigkeit in $\frac{\text{km}}{\text{h}}$ den Anhalteweg (in m) berechnet. Da es in dieser Aufgabe um sicherheitsrelevante Berechnungen geht und Ganzzahldivisionen immer zu Abrundungen führen, sollten Sie Divisionen möglichst spät ausführen und das Ergebnis auf ganze Meter aufrunden lassen.
b) Ist die Regelung „höchstens 50 $\frac{\text{km}}{\text{h}}$ bei 50-m-Sicht“ gerechtfertigt? Wie schnell darf man bei einer Sichtweite von 100 m fahren?

2 Tilgungsplan

Zur Tilgung eines Darlehens bezahlt man über die gesamte Laufzeit monatlich einen festen Betrag, die sogenannte Annuität, zurück. Da unmittelbar davor der monatlich fällige Darlehenszins verbucht wird und dieser mit abnehmendem Darlehensbetrag immer geringer wird, tilgt man zu Beginn der Laufzeit effektiv weniger als zu deren Ende.
Schreiben Sie ein *Assembler*-Programm, das berechnet, nach wie vielen Monaten ein Darlehen von 5000 € bei einer Annuität von 70 € und einem effektiven Jahreszins von 6 % getilgt ist. Die Berechnungen sollen mit positiven Beträgen für die Schulden in ganzen Cent und ohne Rundung erfolgen.

3 Quersumme

Bei der Frage nach der Teilbarkeit natürlicher Zahlen wird häufig die Quersumme benötigt. Testen Sie die Programme, die Sie in den folgenden Teilaufgaben schreiben, jeweils an selbst gewählten Beispielen, die auch Spezialfälle beinhalten.
a) Schreiben Sie ein *Assembler*-Programm, das von einer natürlichen Zahl n die letzte Ziffer abspaltet. Diese soll am Ende im Akkumulator stehen. Die um die letzte Ziffer verkürzte Zahl (0, falls n einstellig ist) soll zur weiteren Verwendung in einem Register gespeichert werden.
Hinweis: Ganzzahldivision durch 10 und anschließende Multiplikation
b) Erweitern Sie Ihr Programm so, dass am Ende die Quersumme von n im Akkumulator steht.
c) Das Programm soll durch rekursive Berechnung der Quersumme prüfen, ob die eingegebene Zahl durch 9 teilbar ist. Wahrheitswerte werden wie üblich mithilfe der Zahlen 0 bzw. 1 dargestellt.

4 Eine etwas andere Registermaschine

Nicht alle Registermaschinen verwenden die *Assembler*-Sprache mit den auf Seite 101 angegebenen Befehlen. Die *Assembler*-Sprache einer anderen Registermaschine kommt mit dem in Fig. 1 dargestellten Befehlssatz aus, es fehlt jedoch beispielsweise ein Befehl zur Multiplikation.
Schreiben Sie ein Programm, welches zwei natürliche Zahlen, die in den Registern R2 und R3 abgelegt sind, miteinander multipliziert und das Ergebnis in R1 schreibt.

`load x`	lädt den Wert aus Rx in den Akkumulator
`store x`	schreibt den Wert aus dem Akkumulator in Rx
`add x`	addiert den Wert aus Rx zum Wert im Akkumulator
`sub x`	subtrahiert vom Wert im Akkumulator den Wert aus Rx
`inc`	vergrößert den Wert im Akkumulator um 1
`dec`	verringert den Wert im Akkumulator um 1
`jgt n`	springt zum Befehl in Zeile n, falls der Wert im Akkumulator positiv ist
`jump n`	springt zum Befehl in Zeile n

Fig. 1

Lösungen auf den Seiten 198 – 200

© Ernst Klett Verlag GmbH, Stuttgart 2010 | www.klett.de | Alle Rechte vorbehalten.
Informatik 5 – Jahrgangsstufe 12
ISBN 978-3-12-731068-9