2 Registermaschine

Lehrer Josef Simpel berechnet den Notendurchschnitt seiner Schüler und Schülerinnen immer noch mithilfe eines sehr einfachen Taschenrechners, der keine Klammerung kennt und nur zwei Speicherzellen Z1 und Z2 hat. Neben der aktuell angezeigten Zahl (Z1) wird in Z2 das Ergebnis der letzten Berechnung gespeichert.
Herr Simpel will damit für folgende Notenserie die Durchschnittsnote berechnen: 3, 5, 2, 1, 3. Geben Sie für alle Schritte dieser Berechnung jeweils die Belegungen der Zellen Z1 und Z2 an.

Zur Formalisierung von Aussagen über Berechenbarkeit und Komplexität von Algorithmen wurde eine Reihe theoretischer Maschinenmodelle entwickelt, z. B. endliche Automaten, Turing-Maschinen oder eben Registermaschinen. Letztere ähneln stark den heute üblichen Mikroprozessoren.

Aktuelle Mikroprozessoren basieren meist auf dem theoretischen Maschinenmodell der **Registermaschine**. Die aktuell benötigten Daten werden dabei in wenigen, speziellen, sehr schnell zugänglichen Speicherzellen innerhalb der Zentraleinheit (den Registern) zwischengespeichert.

Registermaschinen

Eine Registermaschine besteht aus einem Arbeitsspeicher für die Programme sowie einer Reihe von Registern. Der Einfachheit halber gehen wir davon aus, dass alle Daten in den Registern stehen. Die Register haben folgende Aufgaben:

- Der **Befehlszähler** BZ enthält die Speicheradresse des nächsten zu bearbeitenden Befehls.
- Das **Statusregister** SR enthält Informationen über das Ergebnis der letzten Operation.
- Die **Datenregister** A, R1, R2 … dienen zur Ablage der Daten.
  Das Datenregister A **(Akkumulator)** spielt dabei eine besondere Rolle: Es enthält einen Eingabewert für den folgenden Rechenbefehl und nimmt dessen Ergebnis auf.

*Neben den genannten Registern enthalten reale Mikroprozessoren auch ein **Befehlsregister**, das den Code des nächsten auszuführenden Befehls enthält (Seite 103).*

Assembler-Programme

Zur Programmierung von Registermaschinen verwendet man spezielle Programmiersprachen, die nur sehr elementare, auf die spezielle Struktur dieser Maschine bezogene Befehle zur Verfügung stellen. Dazu gehören:

- Transportbefehle zum Laden der Datenregister mit Werten (`LOAD`, `DLOAD`, `STORE`),
- arithmetische Befehle zum Rechnen (`ADD`, `SUB`, `MULT` …),
- Sprungbefehle mit und ohne Bedingung (`JGE`, `JUMP` …),
- logische Verknüpfungen (`AND`, `OR`, Negierung, Invertierung …),
- `END` zum Beenden des Programmlaufs.

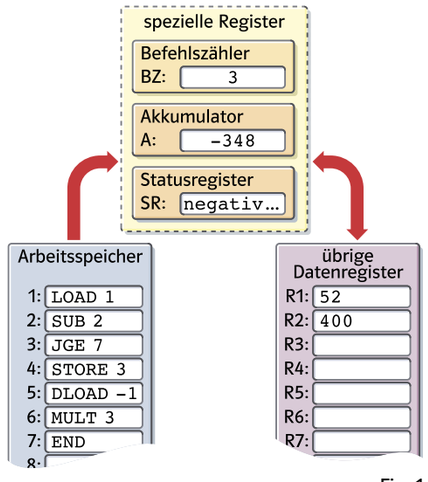

Fig. 1

Informatik 5 – Jahrgangsstufe 12
ISBN 978-3-12-731068-9

Eine *Assembler*-Sprache

Die *Assembler*-Sprache der hier dargestellten Registermaschine umfasst die folgenden Befehle:

Befehl	Auswirkung
`LOAD x`	kopiert den Wert in Rx nach A, erhöht den Wert in BZ um 1
`DLOAD i`	lädt unmittelbar die Zahl i in A, erhöht den Wert in BZ um 1
`STORE x`	kopiert den Wert in A nach Rx, erhöht den Wert in BZ um 1
`ADD x`	addiert den Wert in Rx zum Wert in A, legt das Ergebnis in A ab, erhöht den Wert in BZ um 1
`SUB x`	subtrahiert den Wert in Rx vom Wert in A, legt das Ergebnis in A ab, erhöht den Wert in BZ um 1
`MULT x`	multipliziert den Wert in Rx mit dem Wert in A, legt das Ergebnis in A ab, erhöht den Wert in BZ um 1
`DIV x`	dividiert den Wert in A durch den Wert in Rx (Ganzzahldivision ohne Rest), legt das Ergebnis in A ab, erhöht den Wert in BZ um 1
`JUMP n`	lädt die Zahl n in BZ, Programm wird mit Befehl in Speicherzelle n fortgesetzt Es handelt sich um einen unbedingten Sprung.
`JGE n`	lädt die Zahl n in BZ, falls der Wert in A größer oder gleich 0 ist, erhöht ansonsten den Wert in BZ um 1 Es handelt sich um einen bedingten Sprung. `JGE` steht für „Jump if greater or equal zero“.

`JGT n`
`JLE n`
`JLT n`
`JEQ n`
`JNE n`
	entsprechende bedingte Sprünge für die Fälle
	„größer als 0“ (greater than zero),
	„kleiner oder gleich 0“ (less or equal zero),
	„kleiner als 0“ (less than zero),
	„gleich 0“ (equal zero),
	„ungleich 0“ (not equal zero)

`END`	erhöht den Wert in BZ um 1 und beendet den Programmlauf

Mit dem Programm in Fig. 1 auf Seite 100 wird beispielsweise die absolute Differenz zweier Zahlen, die sich in den Registern R1 und R2 befinden, berechnet.

Zustände der Registermaschine

Registermaschinen kann man auch als Objekte auffassen. Die Klasse REGISTERMASCHINE besitzt dann jeweils ein Attribut für den Befehlszähler, den Akkumulator sowie für jedes ihrer Register (Fig. 1). Die Befehle werden jeweils durch eine Methode repräsentiert, z. B. *add(Rx)* für `ADD Rx`. Ein Zustand einer bestimmten Registermaschine *rm1* (d. h. des Objektes *rm1* dieser Klasse) wird durch die Gesamtheit der Werte aller Attribute von *rm1* festgelegt. Falls durch die Ausführung eines Befehls der Wert (zumindest) eines Attributes geändert wird, so geht das Objekt *rm1* in einen anderen Zustand über. Fig. 2 zeigt dies am Beispiel der Addition `ADD 2` bzw. der Methode *rm1.add(2)*.

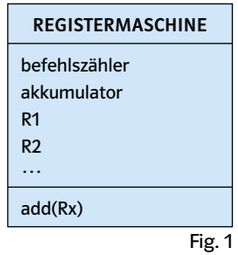

Fig. 1

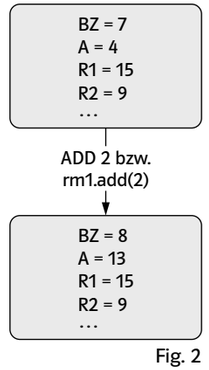

Fig. 2

Informatik 5 – Jahrgangsstufe 12
ISBN 978-3-12-731068-9