Registermaschinen

Maschinenmodelle

In der theoretischen Informatik verwendet man verschiedene Maschinenmodelle, um den Algorithmenbegriff genauer spezifizieren zu können. Wenn es um die grundlegende Definition von Berechenbarkeit oder die Charakterisierung der Klasse **P** der in Polynomialzeit berechenbaren Probleme geht, wird in der Regel das Modell der Turing-Maschine gewählt. Es hat aber den Nachteil, dass es sich sehr stark von realen Computern unterscheidet. Insbesondere kann die Simulation eines einzelnen Schritts eines realen Computers einen linearen oder sogar quadratischen Mehraufwand auf einer Turing-Maschine kosten, so dass eine genauere Analyse der Laufzeit von Turing-Maschinen (unterhalb von polynomiell) wenig Bedeutung für praktische Anwendungen hat.

Deshalb werden wird in dieser Vorlesung das Modell der Registermaschine bevorzugen. Diese Maschinen sind im Englischen als Random Access Machines, abgekürzt RAM bekannt. Sie sind stark verwandt zu Von-Neumann-Rechnern und unterscheiden sich von einem realen Computer nur dadurch, dass man von einem unbegrenzten Speichervolumen ausgeht und auch jedes einzelne Register beliebig große natürliche Zahlen halten kann. Solange man diesen Unterschied nicht ausnutzt, kann man davon ausgehen, dass die auf einer Registermaschine analysierte Laufzeit bis auf einen konstanten Faktor auf reale Rechener übertragen werden kann.

Modell der Registermaschine

Im Gegensatz zum Von-Neumann-Modell trennt man in einer Registermaschine den Speicher in einen nummerierten Befehlsspeicher und einen Datenspeicher, der aus den Registern $c(1), c(2), c(3), \ldots$ gebildet wird.
Im Prozessor einer Registermaschine spielen nur zwei Register eine spezielle Rolle, nämlich der Befehlzähler b, der die Adresse des nächsten auszuführenden Befehls hält, und der Akkumulator, den man mit $c(0)$ bezeichnet.
Der Befehlssatz erlaubt das Laden von Speicherinhalten in den Akkumulator, das Abspeichern des Akkumulatorinhalts in andere Zellen, die Verknüpfung des Akkumulatorinhalts mit dem Inhalt von anderen Registern oder mit konstanten Werten duch die arithmetischen Grundoperationen und unbedingte sowie bedingte Sprunganweisungen wobei die Bedingung durch Vergleich des Akkumulatorinhalts mit Zahlen ausgewertet wird. Die folgenden Tabellen listen die Befehle auf und stellen die Auswirkung ihrer Ausführung auf die betroffenen Registerinhalte dar. Die Parameter j, k stehen für beliebige natürliche Zahlen, i für eine beliebige positive, natürliche Zahl und das Fragezeichen für eine beliebige Vergleichoperation, d.h. $? \in \{<, \leq, >, \geq, =\}$.
Die etwas ungewöhnlichen Definitionen von Subtraktion und Division ergeben sich daraus, dass wir uns die ganze Zeit im Bereich der natürlichen Zahlen bewegen wollen: Wenn die Differenz $c(0) - c(i)$ negativ ist, wird das Ergebnis auf 0 gesetzt. Zur Divi-

sion wird das ganzzahlige Teilen verwendet, d.h. der rationale Quotient $\frac{c(0)}{c(i)}$ wird auf den nächsten ganzzahligen Wert abgerundet.

$LOAD\ i$	$c(0) = c(i)$	$b = b + 1$
$STORE\ i$	$c(i) = c(0)$	$b = b + 1$
$ADD\ i$	$c(0) = c(0) + c(i)$	$b = b + 1$
$MULT\ i$	$c(0) = c(0) \cdot c(i)$	$b = b + 1$
$SUB\ i$	$c(0) = max\,(\,c(0) - c(i), 0\,)$	$b = b + 1$
$DIV\ i$	$c(0) = \left\lfloor \frac{c(0)}{c(i)} \right\rfloor$	$b = b + 1$
$GOTO\ j$		$b = j$
$IF\ c(0)?k\ GOTO\ j$		$\begin{cases} \text{b=j} & \text{falls } c(0)?k \\ \text{b=b+1} & \text{sonst} \end{cases}$
END		

Diese Aufstellung wird ergänzt durch Befehle zum Laden einer Konstante j und Verknüpfen mit dem konstanten Wert j (an Stelle des Registerinhalts $c(i)$) sowie einem Befehlssatz mit indirekter Adressierung. Dabei wird $c(i)$ durch $c(c(i))$ ersetzt, d.h. der Parameter i gibt die Speicherzelle an, deren Inhalt die Adresse des zu ladenden oder zu verknüpfenden Registers beschreibt.

$CLOAD\ j$	$c(0) = j$	$b = b + 1$
$CADD\ j$	$c(0) = c(0) + j$	$b = b + 1$
$CMULT\ j$	$c(0) = c(0) \cdot j$	$b = b + 1$
$CSUB\ j$	$c(0) = max\,(\,c(0) - j, 0\,)$	$b = b + 1$
$CDIV\ i$	$c(0) = \left\lfloor \frac{c(0)}{i} \right\rfloor$	$b = b + 1$
$INDLOAD\ i$	$c(0) = c(c(i))$	$b = b + 1$
$INDSTORE\ i$	$c(c(i)) = c(0)$	$b = b + 1$
$INDADD\ i$	$c(0) = c(0) + c(c(i))$	$b = b + 1$
$INDMULT\ i$	$c(0) = c(0) \cdot c(c(i))$	$b = b + 1$
$INDSUB\ j$	$c(0) = max\,(\,c(0) - c(c(i)), 0\,)$	$b = b + 1$
$INDDIV\ i$	$c(0) = \left\lfloor \frac{c(0)}{c(c(i))} \right\rfloor$	$b = b + 1$

Viele einfache Anweisungen aus einem Pseudocode-Programm erfordern einigen Aufwand, um sie auf einer Registermaschine zu implementieren, aber Registermaschinen sind unversell, d.h. man kann prinzipiell jeden Algorithmus auf einer solchen Maschine realisieren. Wir demonstrieren das an zwei kleinen Beispielen. Kommentare stehen auf der rechten Seite

Beispiel 1: Die Zahlen aus den Registern $c(1)$ und $c(2)$ sollen sortiert werden. Ziel ist es, den kleineren Wert in $c(3)$ und den größeren in $c(4)$ zu speichern. Da ein direkter Vergleich von zwei Zahlen nicht zur Verfügung steht, muss man die Differenz mit 0 vergleichen:

0 :	$LOAD\ 1$	
1 :	$SUB\ 2$	$c(1) > c(2) \iff c(0) > 0$
2 :	$IF\ c(0) > 0\ GOTO\ 8$	
3 :	$LOAD\ 1$	Anweisungen 3 bis 7 für den Fall $c(1) \leq c(2)$
4 :	$STORE\ 3$	
5 :	$LOAD\ 2$	
6 :	$STORE\ 4$	
7 :	$GOTO\ 12$	Man könnte auch gleich den END-Befehl verwenden
8 :	$LOAD\ 1$	Anweisungen 8 bis 11 für der Fall $c(1) > c(2)$
9 :	$STORE\ 4$	
10 :	$LOAD\ 2$	
11 :	$STORE\ 3$	
12 :	END	

Beispiel 2: Hier geht es um die Realisierung einer einfachen Schleifenanweisung. Die Summe der Registerinhalte $c(10), c(11), \ldots, c(10 + c(1))$ soll gebildet und in $c(2)$ gespeichert werden. Der Inhalt $c(1)$ erhöht um 1 beschreibt also die Anzahl der Summanden. Wir werden $c(3)$ für die Adresse des nächsten Summanden verwenden. Mit jedem verarbeiteten Summanden wird $c(1)$ um 1 verkleinert.

0 :	$CLOAD\ 0$	Initialisierung in Anweisungen 0 bis 3
1 :	$STORE\ 2$	
2 :	$CLOAD\ 10$	
3 :	$STORE\ 3$	
4 :	$LOAD\ 2$	Hier beginnt die Schleife
5 :	$INDADD\ 3$	Addition des nächsten Summanden
6 :	$STORE\ 2$	
7 :	$LOAD\ 1$	Testen, ob Summenberechnung abgeschlossen ist
8 :	$IF\ c(0) > 0\ GOTO\ 10$	weitermachen
9 :	END	sonst fertig
10 :	$CSUB\ 1$	
11 :	$STORE\ 1$	
12 :	$LOAD\ 3$	
13 :	$CADD\ 1$	
14 :	$STORE\ 3$	
15 :	$GOTO\ 4$	Rücksprung an Schleifenanfang