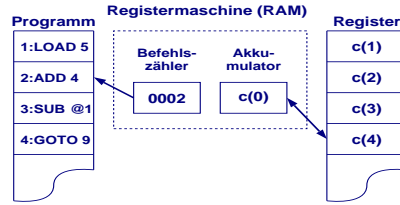


Registermaschine (RAM — random access machine)



$c(i)$ = Inhalt von Register i : beliebige ganze Zahlen

Programm in read-only Bereich, b : Inhalt des Befehlszählers (Start mit $b = 1$)

Adressierungsarten: unmittelbar, direkt, indirekt

RAM-Programm berechnet (partielle) Funktion $f : \mathbb{Z}^m \rightarrow \mathbb{Z}^n$

grundsätzliche Bemerkungen

- jeder Algorithmus lässt sich auf einer RAM implementieren, RAM und Turing-Maschinen können sich gegenseitig simulieren, alle Phänomene der Berechenbarkeit (Halteproblem, Unentscheidbarkeit, Universalität, ...) gelten genauso für RAM
- das RAM-Modell ist einigermaßen realistisch für Komplexitätsbetrachtungen, aber noch hinreichend einfach, um mathematisch handhabbar zu sein
- unrealistisch: beliebig grosse Zahlen in einem Register
man muss (oft) die Länge von Zahlen (= Anzahl der Ziffern der Zahldarstellung \rightarrow logarithmisches Kostenmass) mitberücksichtigen
- beachte: zwischen dem numerischen Wert einer Zahl n und ihrer Länge $\ell(n) = \lfloor \log(|n|) \rfloor + 1$ besteht ein exponentieller Sprung!

Operationen einer Registermaschine

Befehl	Operand		
	$\#i$	i	$@i$
LOAD	$c(0) := i; b++$	$c(0) := c(i); b++$	$c(0) := c(c(i)); b++$
STORE	—	$c(i) := c(0); b++$	$c(c(i)) := c(0); b++$
ADD	$c(0) := c(0) + i; b++$	$c(0) := c(0) + c(i); b++$	$c(0) := c(0) + c(c(i)); b++$
SUB	$c(0) := c(0) - i; b++$	$c(0) := c(0) - c(i); b++$	$c(0) := c(0) - c(c(i)); b++$
SHIFT	—	$c(0) := \lfloor c(i)/2 \rfloor; b++$	$c(0) := \lfloor c(c(i))/2 \rfloor; b++$
ODD	—	$c(0) := c(i) \bmod 2; b++$	$c(0) := c(c(i)) \bmod 2; b++$
GOTO	—	$b := i$	—
IF \circ 0 GOTO	—	$b := i$, wenn $c(0) \circ 0$ $b++$, sonst	—
END	Die Berechnung stoppt		
Hierbei ist $\circ \in \{=, \neq, <, \leq, >, \geq\}$			

RAM-Programm zur Multiplikation zweier ganzer Zahlen

01: LOAD 1	11: SUB 1	21: STORE 1
02: IF \geq 0 GOTO 16	12: STORE 1	22: LOAD 3
03: STORE 3	13: LOAD #0	23: ADD 2
04: LOAD 2	14: SUB 2	24: STORE 3
05: STORE 1	15: STORE 2	25: GOTO18
06: LOAD 3	16: LOAD #0	26: LOAD 3
07: STORE 2	17: STORE 3	27: STORE 1
08: LOAD 1	18: LOAD 1	28: END
09: IF \geq 0 GOTO 16	19: IF = 0 GOTO 26	
10: LOAD #0	20: SUB #1	

Hinweise:

- Argumente in Registern 1 und 2
- Positives Argument in Register 1 bringen — evtl. durch Vertauschen der Argumente oder beide Argumente durch Negativ ersetzen
[Zeilen 01-15]
- Eigentliche Multiplikation durch fortgesetzte Addition in Register 3
[Zeilen 16-25]
- Resultat von Register 3 in Register 1 kopieren
[Zeilen 26-27]

RAM-Programm zur Summation von n Zahlen (a_1, a_2, \dots, a_n)

01: LOAD 1	08: STORE 1	15: ADD @1
02: IF > 0 GOTO 6	09: SUB #2	16: STORE 2
03: LOAD #0	10: IF > 0 GOTO 14	17: LOAD 1
04: STORE 1	11: LOAD 2	18: SUB #1
05: END	12: STORE 1	19: STORE 1
06: LOAD 1	13: END	20: GOTO 9
07: ADD #1	14: LOAD 2	

Hinweise

- Beispiel verwendet indirekte Adressierung
- Anzahl n der Summanden in Register 1
Summand a_i in Register $i + 1$ ($1 \leq i \leq n$)
- Testen, ob Summe leer ist ($/n = 0$)
[Zeilen 1-5]
- Zwischenergebnisse der Summation im Register 2 akkumulieren
[Zeilen 14-20]
- Testen, ob noch weitere Summanden vorhanden; wenn nicht: Resultat von Register 2 nach Register 1 kopieren
[Zeilen 6-13]

Logarithmische Kosten von RAM-Operationen

Befehl	Operand		
	# i	i	@ i
LOAD	$\ell(i)$	$\ell(i) + \ell(c(i))$	$\ell(i) + \ell(c(i)) + \ell(c(c(i)))$
STORE	—	$\ell(i) + \ell(c(0))$	$\ell(i) + \ell(c(i)) + \ell(c(0))$
ADD	$\ell(c(0)) + \ell(i)$	$\ell(c(0)) + \ell(i) + \ell(c(i))$	$\ell(c(0)) + \ell(i) + \ell(c(i)) + \ell(c(c(i)))$
SUB	$\ell(c(0)) + \ell(i)$	$\ell(c(0)) + \ell(i) + \ell(c(i))$	$\ell(c(0)) + \ell(i) + \ell(c(i)) + \ell(c(c(i)))$
RAND	$\ell(i)$	$\ell(i) + \ell(c(i))$	$\ell(i) + \ell(c(i)) + \ell(c(c(i)))$
SHIFT	—	$\ell(i) + \ell(c(i))$	$\ell(i) + \ell(c(i)) + \ell(c(c(i)))$
ODD	—	$\ell(i)$	$\ell(i) + \ell(c(i))$
GOTO	—	1	—
IF \circ 0 GOTO	—	$\ell(c(0))$	—
END	1		
	$\ell(x) = \lfloor \log(x) \rfloor + 1$		