Organisatorisches

- Nächste Vorlesung:
  Donnerstag, Oktober 26, 12:15–13:45 Uhr, Aula

- Achtung:
  In der Woche Oct-30 bis Nov-3 entfallen alle Tutorien!

- Webseite:
  http://algo.rwth-aachen.de/Lehre/WS1718/BuK.php
Wiederholung
Wdh.: k-Band- vs 1-Band-TM

Satz

Eine $k$-Band-TM $M$, die mit Rechenzeit $t(n)$ und Platz $s(n)$ auskommt, kann von einer (1-Band-)TM $M'$ mit Zeitbedarf $O(t^2(n))$ und Platzbedarf $O(s(n))$ simuliert werden.

Simuliert durch
Wdh.: Gödelnummer $\langle M \rangle$

$\Sigma = \{0, 1\}$  
$\Gamma = \{0, 1, \text{B}\}$  
$Q = \{q_1, q_2, q_3\}$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$(q_1, B, R)$</td>
<td>$(q_3, B, R)$</td>
<td>$(q_2, B, N)$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_1, 1, R)$</td>
<td>$(q_2, 0, R)$</td>
<td>$(q_1, B, L)$</td>
</tr>
</tbody>
</table>

$\Sigma = \{0, 1\}$  
$\Gamma = \{0, 1, \text{B}\}$  
$Q = \{q_1, q_2, q_3\}$

Start  Ende  Blank

$q_1$  $q_2$  B

$\Sigma = \{0, 1\}$  
$\Gamma = \{0, 1, \text{B}\}$  
$Q = \{q_1, q_2, q_3\}$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$(q_1, B, R)$</td>
<td>$(q_3, B, R)$</td>
<td>$(q_2, B, N)$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_1, 1, R)$</td>
<td>$(q_2, 0, R)$</td>
<td>$(q_1, B, L)$</td>
</tr>
</tbody>
</table>

$\Sigma = \{0, 1\}$  
$\Gamma = \{0, 1, \text{B}\}$  
$Q = \{q_1, q_2, q_3\}$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$(q_1, B, R)$</td>
<td>$(q_3, B, R)$</td>
<td>$(q_2, B, N)$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_1, 1, R)$</td>
<td>$(q_2, 0, R)$</td>
<td>$(q_1, B, L)$</td>
</tr>
</tbody>
</table>

$\Sigma = \{0, 1\}$  
$\Gamma = \{0, 1, \text{B}\}$  
$Q = \{q_1, q_2, q_3\}$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$(q_1, B, R)$</td>
<td>$(q_3, B, R)$</td>
<td>$(q_2, B, N)$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_1, 1, R)$</td>
<td>$(q_2, 0, R)$</td>
<td>$(q_1, B, L)$</td>
</tr>
</tbody>
</table>
Simulierte Turingmaschine $M$

Initialisierung der universellen Maschine $U$
Laufzeit der universellen TM

- Bei Eingabe $\langle M \rangle w$ simuliert $U$ die TM $M$ auf Wort $w$.
- Jeder Schritt von $M$ wird dabei von $U$ in $f(|\langle M \rangle|)$ Zeit simuliert.
- Wenn $|\langle M \rangle|$ als Konstante angesehen wird, so simuliert $U$ die TM $M$ mit einem konstanten Zeit- und Platzverlust.
Wikipedia: Alan Turing was an English computer scientist, mathematician, logician, cryptanalyst, philosopher and theoretical biologist.

Turing was highly influential in the development of theoretical computer science, providing a formalisation of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general purpose computer. Turing is widely considered to be the father of theoretical computer science and artificial intelligence.
Vorlesung VL-04
Registermaschinen

- Registermaschinen (RAMs)
- Simulation von RAM durch TM
- Simulation von TM durch RAM

- Church-Turing These
Registermaschinen (RAM)
Registermaschinen (RAM)

Befehlssatz:

LOAD, STORE, ADD, SUB, MULT, DIV
INDLOAD, INDSTORE, INDADD, INDSUB, INDMULT, INDDIV
CLOAD, CADD, CSUB, CMULT, CDIV
GOTO, IF c(0)\(\ ? x\) THEN GOTO j (mit ? aus \{=,<,\leq,>,\geq\}), END
LOAD $i$: $c(0) := c(i)$, $b := b + 1$;
RAM: Einige ausgewählte Befehle

LOAD \( i \): \( c(0) := c(i) \), \( b := b + 1 \);

INDLOAD \( i \): \( c(0) := c(c(i)) \), \( b := b + 1 \);
RAM: Einige ausgewählte Befehle

LOAD $i$: $c(0) := c(i)$, \quad b := b + 1$

INDLOAD $i$: $c(0) := c(c(i))$, \quad b := b + 1$

CLOAD $i$: $c(0) := i$, \quad b := b + 1$
LOAD $i$: \[c(0) := c(i), \quad b := b + 1;\]

INDLOAD $i$: \[c(0) := c(c(i)), \quad b := b + 1;\]

CLOAD $i$: \[c(0) := i, \quad b := b + 1;\]

STORE $i$: \[c(i) := c(0), \quad b := b + 1;\]
RAM: Einige ausgewählte Befehle

LOAD $i$: \[c(0) := c(i), \quad b := b + 1;\]
INDLOAD $i$: \[c(0) := c(c(i)), \quad b := b + 1;\]
CLOAD $i$: \[c(0) := i, \quad b := b + 1;\]
STORE $i$: \[c(i) := c(0), \quad b := b + 1;\]
INDSTORE $i$: \[c(c(i)) := c(0), \quad b := b + 1;\]
RAM: Einige ausgewählte Befehle

LOAD $i$: \[ c(0) := c(i), \quad b := b + 1; \]

INDLOAD $i$: \[ c(0) := c(c(i)), \quad b := b + 1; \]

CLOAD $i$: \[ c(0) := i, \quad b := b + 1; \]

STORE $i$: \[ c(i) := c(0), \quad b := b + 1; \]

INDSTORE $i$: \[ c(c(i)) := c(0), \quad b := b + 1; \]

ADD $i$: \[ c(0) := c(0) + c(i), \quad b := b + 1; \]
RAM: Einige ausgewählte Befehle

LOAD $i$:  
$c(0) := c(i)$,  
$b := b + 1$;

INDLOAD $i$:  
$c(0) := c(c(i))$,  
$b := b + 1$;

CLOAD $i$:  
$c(0) := i$,  
$b := b + 1$;

STORE $i$:  
$c(i) := c(0)$,  
$b := b + 1$;

INDSTORE $i$:  
$c(c(i)) := c(0)$,  
$b := b + 1$;

ADD $i$:  
$c(0) := c(0) + c(i)$,  
$b := b + 1$;

CADD $i$:  
$c(0) := c(0) + i$,  
$b := b + 1$.;
LOAD \( i \): \( c(0) := c(i), \quad b := b + 1; \)
INDLOAD \( i \): \( c(0) := c(c(i)), \quad b := b + 1; \)
CLOAD \( i \): \( c(0) := i, \quad b := b + 1; \)
STORE \( i \): \( c(i) := c(0), \quad b := b + 1; \)
INDSTORE \( i \): \( c(c(i)) := c(0), \quad b := b + 1; \)
ADD \( i \): \( c(0) := c(0) + c(i), \quad b := b + 1; \)
CADD \( i \): \( c(0) := c(0) + i, \quad b := b + 1; \)
INDADD \( i \): \( c(0) := c(0) + c(c(i)), \quad b := b + 1; \)
RAM: Einige ausgewählte Befehle

\[
\begin{align*}
\text{LOAD } i: & \quad c(0) := c(i), \quad b := b + 1; \\
\text{INDLOAD } i: & \quad c(0) := c(c(i)), \quad b := b + 1; \\
\text{CLOAD } i: & \quad c(0) := i, \quad b := b + 1; \\
\text{STORE } i: & \quad c(i) := c(0), \quad b := b + 1; \\
\text{INDSTORE } i: & \quad c(c(i)) := c(0), \quad b := b + 1; \\
\text{ADD } i: & \quad c(0) := c(0) + c(i), \quad b := b + 1; \\
\text{CADD } i: & \quad c(0) := c(0) + i, \quad b := b + 1; \\
\text{INDADD } i: & \quad c(0) := c(0) + c(c(i)), \quad b := b + 1; \\
\end{align*}
\]
<table>
<thead>
<tr>
<th>Befehl</th>
<th>Anweisung</th>
<th>b := b + 1;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LOAD i:</strong></td>
<td>(c(0) := c(i),)</td>
<td></td>
</tr>
<tr>
<td><strong>INDLOAD i:</strong></td>
<td>(c(0) := c(c(i)),)</td>
<td></td>
</tr>
<tr>
<td><strong>CLOAD i:</strong></td>
<td>(c(0) := i,)</td>
<td></td>
</tr>
<tr>
<td><strong>STORE i:</strong></td>
<td>(c(i) := c(0),)</td>
<td></td>
</tr>
<tr>
<td><strong>INDSTORE i:</strong></td>
<td>(c(c(i)) := c(0),)</td>
<td></td>
</tr>
<tr>
<td><strong>ADD i:</strong></td>
<td>(c(0) := c(0) + c(i),)</td>
<td></td>
</tr>
<tr>
<td><strong>CADD i:</strong></td>
<td>(c(0) := c(0) + i,)</td>
<td></td>
</tr>
<tr>
<td><strong>INDADD i:</strong></td>
<td>(c(0) := c(0) + c(c(i)),)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SUB i:</strong></td>
<td>(c(0) := \max{0, c(0) - c(i)})</td>
<td></td>
</tr>
</tbody>
</table>
RAM: Einige ausgewählte Befehle

LOAD $i$: \[ c(0) := c(i), \quad b := b + 1; \]
INDLOAD $i$: \[ c(0) := c(c(i)), \quad b := b + 1; \]
CLOAD $i$: \[ c(0) := i, \quad b := b + 1; \]
STORE $i$: \[ c(i) := c(0), \quad b := b + 1; \]
INDSTORE $i$: \[ c(c(i)) := c(0), \quad b := b + 1; \]
ADD $i$: \[ c(0) := c(0) + c(i), \quad b := b + 1; \]
CADD $i$: \[ c(0) := c(0) + i, \quad b := b + 1; \]
INDADD $i$: \[ c(0) := c(0) + c(c(i)), \quad b := b + 1; \]
SUB $i$: \[ c(0) := \max\{0, c(0) - c(i)\} \quad b := b + 1; \]

...
RAM: Einige ausgewählte Befehle

LOAD $i$: $c(0) := c(i)$, $b := b + 1$
INDLOAD $i$: $c(0) := c(c(i))$, $b := b + 1$
CLOAD $i$: $c(0) := i$, $b := b + 1$
STORE $i$: $c(i) := c(0)$, $b := b + 1$
INDSTORE $i$: $c(c(i)) := c(0)$, $b := b + 1$
ADD $i$: $c(0) := c(0) + c(i)$, $b := b + 1$
CADD $i$: $c(0) := c(0) + i$, $b := b + 1$
INDADD $i$: $c(0) := c(0) + c(c(i))$, $b := b + 1$

SUB $i$: $c(0) := \max\{0, c(0) - c(i)\}$, $b := b + 1$

DIV $i$: $c(0) := \begin{cases} \lfloor c(0)/c(i) \rfloor & \text{falls } c(i) \neq 0, \\ 0 & \text{sonst} \end{cases}$, $b := b + 1$

GOTO $j$: $b := j$
LOAD $i$: $c(0) := c(i)$, $b := b + 1$

INDLOAD $i$: $c(0) := c(c(i))$, $b := b + 1$

CLOAD $i$: $c(0) := i$, $b := b + 1$

STORE $i$: $c(i) := c(0)$, $b := b + 1$

INDSTORE $i$: $c(c(i)) := c(0)$, $b := b + 1$

ADD $i$: $c(0) := c(0) + c(i)$, $b := b + 1$

CADD $i$: $c(0) := c(0) + i$, $b := b + 1$

INDADD $i$: $c(0) := c(0) + c(c(i))$, $b := b + 1$

SUB $i$: $c(0) := \max\{0, c(0) - c(i)\}$, $b := b + 1$

DIV $i$: $c(0) := \begin{cases} 
\lfloor c(0)/c(i) \rfloor & \text{falls } c(i) \neq 0, \\
0 & \text{sonst}
\end{cases}$, $b := b + 1$

GOTO $j$: $b := j$

IF $c(0) = x$ GOTO $j$: $b := j$ falls $c(0) = x$, sonst $b := b + 1$
LOAD $i$: \[ c(0) := c(i), \quad b := b + 1; \]
INDLOAD $i$: \[ c(0) := c(c(i)), \quad b := b + 1; \]
CLOAD $i$: \[ c(0) := i, \quad b := b + 1; \]
STORE $i$: \[ c(i) := c(0), \quad b := b + 1; \]
INDSTORE $i$: \[ c(c(i)) := c(0), \quad b := b + 1; \]
ADD $i$: \[ c(0) := c(0) + c(i), \quad b := b + 1; \]
CADD $i$: \[ c(0) := c(0) + i, \quad b := b + 1; \]
INDADD $i$: \[ c(0) := c(0) + c(c(i)), \quad b := b + 1; \]
SUB $i$: \[ c(0) := \max\{0, c(0) - c(i)\} \quad b := b + 1; \]
DIV $i$: \[ c(0) := \begin{cases} \left\lfloor \frac{c(0)}{c(i)} \right\rfloor & \text{falls } c(i) \neq 0, \\ 0 & \text{sonst} \end{cases}, \quad b := b + 1; \]
GOTO $j$: \[ b := j \]
IF $c(0) = x$ GOTO $j$: \[ b := j \quad \text{falls } c(0) = x, \quad \text{sonst } b := b + 1; \]
END.
RAM: Funktionsweise

- Der Speicher der RAM ist unbeschränkt und besteht aus dem Akkumulator \(c(0)\) und aus den Registern \(c(1), c(2), c(3), \ldots\)
- Die Inhalte der Register sind natürliche Zahlen, die beliebig gross werden können.
RAM: Funktionsweise

- Der Speicher der RAM ist unbeschränkt und besteht aus dem Akkumulator $c(0)$ und aus den Registern $c(1), c(2), c(3), \ldots$
- Die Inhalte der Register sind natürliche Zahlen, die beliebig gross werden können.

- Die Eingabe besteht ebenfalls aus natürlichen Zahlen, die zu Beginn in den ersten Registern abgespeichert sind.
- Alle anderen Register sind mit 0 initialisiert.
- Der Befehlszähler startet mit dem Wert 1. Ausgeführt wird jeweils der Befehl in derjenigen Zeile, auf die der Befehlszähler verweist.
Der Speicher der RAM ist unbeschränkt und besteht aus dem Akkumulator \( c(0) \) und aus den Registern \( c(1), c(2), c(3), \ldots \).

Die Inhalte der Register sind natürliche Zahlen, die beliebig gross werden können.

Die Eingabe besteht ebenfalls aus natürlichen Zahlen, die zu Beginn in den ersten Registern abgespeichert sind.

Alle anderen Register sind mit 0 initialisiert.

Der Befehlszähler startet mit dem Wert 1. Ausgeführt wird jeweils der Befehl in derjenigen Zeile, auf die der Befehlszähler verweist.

Die Rechnung stoppt, sobald der Befehl END erreicht ist.

Die Ausgabe befindet sich nach dem Stoppen in den ersten Registern.
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

<table>
<thead>
<tr>
<th>Speicher</th>
<th>c(1)</th>
<th>c(2)</th>
<th>c(3)</th>
<th>c(4)</th>
<th>c(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Befehlszähler: 1
Akkumulator: 0
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

c(1) 2

Befehlszähler Akkumulator

1 0
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler

Akkumulator

\[c(1) \quad c(2) \quad c(3) \quad c(4) \quad c(5)\]
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If $c(0) = 0$ THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c(5)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>c(4)</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>3</td>
<td>c(3)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>c(2)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>c(1)</td>
</tr>
</tbody>
</table>
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler

Akkumulator
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<table>
<thead>
<tr>
<th>Index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>c(5)</td>
</tr>
<tr>
<td>1</td>
<td>c(4)</td>
</tr>
<tr>
<td>2</td>
<td>c(3)</td>
</tr>
<tr>
<td>3</td>
<td>c(2)</td>
</tr>
<tr>
<td>4</td>
<td>c(1)</td>
</tr>
</tbody>
</table>

Befehlszähler: 3
Akkumulator: 1
1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output: 2,0,8

Speicher:
- c(5)
- c(4)
- c(3)
- c(2)
- c(1)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

<table>
<thead>
<tr>
<th>Speicher</th>
<th>Akkumulator</th>
<th>Befehlszähler</th>
</tr>
</thead>
<tbody>
<tr>
<td>c(1)</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>c(2)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>c(3)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>c(4)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>c(5)</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Input: 2,3
Output:
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler
Akkumulator

5
3

0 0 2
1
3
4
5

Input: 2,3
Output:

Speicher

Befehlszähler
Akkumulator

5
3

0 0 2
1
3
4
5
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

-1

Befehlszähler

Akkumulator
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

\[ \begin{array}{|c|c|}
\hline
& c(1) \\
\hline
2 & 0 \\
\hline
3 & c(2) \\
\hline
1 & c(3) \\
\hline
0 & c(4) \\
\hline
0 & c(5) \\
\hline
\end{array} \]

Befehlszähler Akkumulator
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Befehlszähler

Akkumulator

Speicher

6

2

0 0 1 3 2

c(5) c(4) c(3) c(2) c(1)

Input: 2,3
Output:

Befehlszähler

Akkumulator

Speicher

6

2

0 0 1 3 2

c(5) c(4) c(3) c(2) c(1)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

|
---|--|--|--|--|--|
|   | 0 | 0 | 1 | c(3) | c(4) |
| 0 | 2 | 2 | 2 | c(2) | c(5) |

Befehlszähler  Akkumulator  Speicher
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:
- c(1) = 2
- c(2) = 2
- c(3) = 1
- c(4) = 0
- c(5) = 0

Akkumulator:
- 2

Befehlszähler:
- 7
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>c(5)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>c(4)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>c(3)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>c(2)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>c(1)</td>
</tr>
</tbody>
</table>
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

<table>
<thead>
<tr>
<th></th>
<th>c(1)</th>
<th>c(2)</th>
<th>c(3)</th>
<th>c(4)</th>
<th>c(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speicher</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Befehlszähler: 8
Akkumulator: 1

RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher: 
- c(5)
- c(4)
- c(3)
- c(2)
- c(1)

Befehlszähler

Akkumulator

9

2
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>c(1)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c(2)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c(3)</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c(4)</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c(5)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Befehlszähler

Akkumulator

c(1)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler

Akkumulator
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

\[
\begin{align*}
\text{Speicher:} & \\
& \begin{array}{c}
0 \\
0 \\
2 \\
2 \\
2 \\
\end{array} \\
& c(5) \\
& c(4) \\
& c(3) \\
& c(2) \\
& c(1)
\end{align*}
\]

Befehlszähler

Akkumulator

10

2
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

\[ \begin{array}{c}
0 \\
0 \\
2 \\
2 \\
2 \\
\end{array} \]

\[ \begin{array}{c}
c(5) \\
c(4) \\
c(3) \\
c(2) \\
c(1) \\
\end{array} \]

Befehlszähler

Akkumulator

Speicher
RAM: Beispielprogramm

Input: 2,3
Output:

<table>
<thead>
<tr>
<th>Input: 2,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>c(5)</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>c(4)</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>c(3)</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>c(2)</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>c(1)</td>
</tr>
</tbody>
</table>

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Befehlszähler

Akkumulator

Speicher
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:
- c(1) = 2
- c(2) = 2
- c(3) = 2
- c(4) = 0
- c(5) = 0

Input:
- c(0) = 0

Output:
- c(0) = 0
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<table>
<thead>
<tr>
<th></th>
<th>c(1)</th>
<th>c(2)</th>
<th>c(3)</th>
<th>c(4)</th>
<th>c(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Befehlszähler

Akkumulator

Akkumulator

Input: 2,3
Output:
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Befehlszähler

Akkumulator

Speicher

5
2

0
2
2

0
2
0

2
2

0
c(5)
c(4)
c(3)
c(2)
c(1)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2, 3
Output:

Speicher

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>c(5)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>c(4)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c(3)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c(2)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c(1)</td>
<td></td>
</tr>
</tbody>
</table>
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2, 3
Output:

<table>
<thead>
<tr>
<th>Speicher</th>
</tr>
</thead>
<tbody>
<tr>
<td>c(1)</td>
</tr>
<tr>
<td>c(2)</td>
</tr>
<tr>
<td>c(3)</td>
</tr>
<tr>
<td>c(4)</td>
</tr>
<tr>
<td>c(5)</td>
</tr>
</tbody>
</table>

Befehlszähler

Akkumulator
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler

Akkumulator

0
0
2
2
2
0
0
2
2
2

\[ c(1) \]
\[ c(2) \]
\[ c(3) \]
\[ c(4) \]
\[ c(5) \]
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler

Akkumulator

0
c(5)

0
c(4)

2
c(3)

1
c(2)

2
c(1)
RAM: Beispielprogramm

1:  CLOAD 1
2:  STORE 3
3:  LOAD 2
4:  If $c(0) = 0$ THEN GOTO 11
5:  CSUB 1
6:  STORE 2
7:  LOAD 3
8:  MULT 1
9:  STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

\[
\begin{array}{cccc}
\text{c(1)} & \text{c(2)} & \text{c(3)} & \text{c(4)} & \text{c(5)} \\
2 & 1 & 2 & 0 & 0 \\
\end{array}
\]
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler

Akkumulator

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>c(5)</td>
</tr>
<tr>
<td>0</td>
<td>c(4)</td>
</tr>
<tr>
<td>2</td>
<td>c(3)</td>
</tr>
<tr>
<td>1</td>
<td>c(2)</td>
</tr>
<tr>
<td>0</td>
<td>c(1)</td>
</tr>
</tbody>
</table>
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Input: 2,3
Output:

Befehlszähler

Akkumulator

Speicher

\[ c(1) \quad c(2) \quad c(3) \quad c(4) \quad c(5) \]
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

0  c(5)
0  c(4)
2  c(3)
1  c(2)
2  c(1)

Befehlszähler
Akkumulator
Speicher
1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Befehlszähler

Akkumulator

Speicher

9

4

0

2

1

2

0

c(5)
c(4)
c(3)
c(2)
c(1)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

0  c(5)
0  c(4)
4  c(3)
1  c(2)
2  c(1)

Befehlszähler

Akkumulator

Speicher
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler

Akkumulator

10

4

0 c(5)
0 c(4)
4 c(3)
1 c(2)
2 c(1)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>c(5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>c(4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>c(3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>c(2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Befehlszähler

Akkumulator

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3

Output:

... c(5)
    0
    0
    4
    c(4)
    c(3)
    1
    1
    2
    c(2)
    c(1)
RAM: Beispielprogramm

1: CLOAD 1  
2: STORE 3  
3: LOAD 2  
4: If c(0) = 0 THEN GOTO 11  
5: CSUB 1  
6: STORE 2  
7: LOAD 3  
8: MULT 1  
9: STORE 3  
10: GOTO 3  
11: END

Input: 2,3
Output:

Speicher:

<table>
<thead>
<tr>
<th></th>
<th>c(1)</th>
<th></th>
<th>c(2)</th>
<th></th>
<th>c(3)</th>
<th></th>
<th>c(4)</th>
<th></th>
<th>c(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Befehlszähler: 4
Akkumulator: 1
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<table>
<thead>
<tr>
<th></th>
<th>c(1)</th>
<th>c(2)</th>
<th>c(3)</th>
<th>c(4)</th>
<th>c(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Befehlszähler: 4
Akkumulator: 1
## RAM: Beispielprogramm

<table>
<thead>
<tr>
<th>Befehlszähler</th>
<th>Akkumulator</th>
<th>Speicher</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

1: CLOAD 1  
2: STORE 3  
3: LOAD 2  
4: If c(0) = 0 THEN GOTO 11  
5: CSUB 1  
6: STORE 2  
7: LOAD 3  
8: MULT 1  
9: STORE 3  
10: GOTO 3  
11: END

---

Input: 2,3  
Output: 2,0,8  

Input: ...  
Output: 0,0,4,1,2,0,0,0,0,0  

BuK/WS 2017 VL-04: Registermaschinen
1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:
- c(1) = 2
- c(2) = 1
- c(3) = 4
- c(4) = 0
- c(5) = 0

Befehlszähler: 5
Akkumulator: 1
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

```
  0
  4
  1
  2
c(1)
c(2)
c(3)
c(4)
c(5)
```

Befehlszähler

Akkumulator

Speicher
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:
0  
c(5)
0  
c(4)
4  
c(3)
1  
c(2)
2  
c(1)

Befehlszähler
Akkumulator

BuK/WS 2017 VL-04: Registermaschinen
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>c(1)</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>c(2)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>c(3)</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>c(4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c(5)</td>
</tr>
</tbody>
</table>

Befehlszähler

Akkumulator

7

0
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:
- c(1) = 2
- c(2) = 0
- c(3) = 0
- c(4) = 4
- c(5) = 0

Befehlszähler
Akkumulator
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>c(1)</td>
<td>c(2)</td>
<td>c(3)</td>
<td>c(4)</td>
<td>c(5)</td>
</tr>
</tbody>
</table>
1:  CLOAD 1
2:  STORE 3
3:  LOAD 2
4:  If c(0) = 0 THEN GOTO 11
5:  CSUB 1
6:  STORE 2
7:  LOAD 3
8:  MULT 1
9:  STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>c(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
<td>c(4)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>c(3)</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>c(2)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>c(1)</td>
</tr>
</tbody>
</table>
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher

Befehlszähler

Akkumulator

\( c(1) \)
\( c(2) \)
\( c(3) \)
\( c(4) \)
\( c(5) \)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>c(5)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>c(4)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>c(3)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>c(2)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c(1)</td>
<td></td>
</tr>
</tbody>
</table>

Befehlszähler:

Akkumulator:

9
8
RAM: Beispielprogramm

1:  CLOAD 1
2:  STORE 3
3:  LOAD 2
4:  If c(0) = 0 THEN GOTO 11
5:  CSUB 1
6:  STORE 2
7:  LOAD 3
8:  MULT 1
9:  STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

<table>
<thead>
<tr>
<th></th>
<th>c(5)</th>
<th>c(4)</th>
<th>c(3)</th>
<th>c(2)</th>
<th>c(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Befehlszähler: 10
Akkumulator: 8
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

10 8

Befehlszähler  Akkumulator  Speicher

0  c(5)
0  c(4)
8  c(3)
0  c(2)
2  c(1)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3

Output:

Befehlszähler

Akkumulator

Speicher

3

8

0

0

8

0

2

c(1)
c(2)
c(3)
c(4)
c(5)
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:
- c(1): 8
- c(2): 0
- c(3): 2
- c(4): 0
- c(5): 0

Befehlszähler
Akkumulator

3
8
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Speicher:

- c(1) = 2
- c(2) = 0
- c(3) = 8
- c(4) = 0
- c(5) = 0

Befehlszähler:

- 4

Akkumulator:

- 0
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output:

Input: 2,3
Output:

... | c(5) | c(4) | c(3) | c(2) | c(1)
---|-----|-----|-----|-----|-----
0   |     |     |     |     |     
0   |     |     |     |     |     
8   |     |     |     |     |     
0   |     |     |     |     |     
2   |     |     |     |     |     

Befehlszähler | Akkumulator | Speicher

4 | 0

yes
RAM: Beispielprogramm

1: CLOAD 1
2: STORE 3
3: LOAD 2
4: If c(0) = 0 THEN GOTO 11
5: CSUB 1
6: STORE 2
7: LOAD 3
8: MULT 1
9: STORE 3
10: GOTO 3
11: END

Input: 2,3
Output: 2,0,8

Speicher

Befehlszähler
Akkumulator

11
0

c(5)
c(4)
c(3)
c(2)
c(1)
Auf einer RAM können wir alle Befehle realisieren (wie beispielsweise Schleifen und Rekursionen), die wir von höheren Programmiersprachen her gewohnt sind,
Auf einer RAM können wir alle Befehle realisieren (wie beispielsweise Schleifen und Rekursionen), die wir von höheren Programmiersprachen her gewohnt sind.

Modelle für die Rechenzeit

- **Uniformes Kostenmaß**: Jeder Schritt zählt als eine Zeiteinheit.
- **Logarithmisches Kostenmaß**: Die Laufzeitkosten eines Schrittes sind proportional zur binären Länge der Zahlen in den angesprochenen Registern.
Simulation von RAM durch TM
Simulation RAM durch TM

Satz

Für jede im logarithmischen Kostenmass $t(n)$-zeitbeschränkte RAM $R$ gibt es ein Polynom $q$ und eine $q(n + t(n))$-zeitbeschränkte TM $M$, die $R$ simuliert.
Simulation RAM durch TM

Satz

Für jede im logarithmischen Kostenmass $t(n)$-zeitbeschränkte RAM $R$ gibt es ein Polynom $q$ und eine $q(n + t(n))$-zeitbeschränkte TM $M$, die $R$ simuliert.

Im Beweis können wir o.B.d.A. für die Simulation eine 2-Band-TM statt einer 1-Band-TM verwenden. Warum?
Vorbemerkungen (1)

Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.
Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.

Wir werden zeigen: Die Laufzeit der Simulation der RAM mit Laufzeitschranke $t(n)$ durch eine 2-Band-TM ist nach oben beschränkt durch $t'(n) = \alpha(n + t(n))^\beta$. 
Vorbemerkungen (1)

- Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.

- Wir werden zeigen: Die Laufzeit der Simulation der RAM mit Laufzeitschranke $t(n)$ durch eine 2-Band-TM ist nach oben beschränkt durch $t'(n) = \alpha(n + t(n))^\beta$.

- Die 2-Band-TM mit Laufzeitschranke $t'(n)$ kann nun wiederum mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden, also mit einer Laufzeitschranke der Form $t''(n) = \gamma(t'(n))^2$. 

Vorbemerkungen (1)

- Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.
- Wir werden zeigen: Die Laufzeit der Simulation der RAM mit Laufzeitschranke $t(n)$ durch eine 2-Band-TM ist nach oben beschränkt durch $t'(n) = \alpha(n + t(n))^\beta$.
- Die 2-Band-TM mit Laufzeitschranke $t'(n)$ kann nun wiederum mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden, also mit einer Laufzeitschranke der Form $t''(n) = \gamma(t'(n))^2$.
- Für die Simulation der RAM auf der (1-Band-)TM ergibt sich somit eine Laufzeitschranke von

$$t''(n) = \gamma(t'(n))^2$$
Vorbemerkungen (1)

- Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.
- Wir werden zeigen: Die Laufzeit der Simulation der RAM mit Laufzeitschranke $t(n)$ durch eine 2-Band-TM ist nach oben beschränkt durch $t'(n) = \alpha(n + t(n))^\beta$.
- Die 2-Band-TM mit Laufzeitschranke $t'(n)$ kann nun wiederum mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden, also mit einer Laufzeitschranke der Form $t''(n) = \gamma(t'(n))^2$.
- Für die Simulation der RAM auf der (1-Band-)TM ergibt sich somit eine Laufzeitschranke von

$$t''(n) = \gamma(t'(n))^2 = \gamma (\alpha(n + t(n))^\beta)^2$$
Vorbemerkungen (1)

- Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.
- Wir werden zeigen: Die Laufzeit der Simulation der RAM mit Laufzeitschranke $t(n)$ durch eine 2-Band-TM ist nach oben beschränkt durch $t'(n) = \alpha(n + t(n))^\beta$.
- Die 2-Band-TM mit Laufzeitschranke $t'(n)$ kann nun wiederum mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden, also mit einer Laufzeitschranke der Form $t''(n) = \gamma(t'(n))^2$.
- Für die Simulation der RAM auf der (1-Band-)TM ergibt sich somit eine Laufzeitschranke von

$$t''(n) = \gamma(t'(n))^2 = \gamma(\alpha(n + t(n))^\beta)^2 = \gamma\alpha^2 \cdot (n + t(n))^{2\beta}.$$
Seien $\alpha, \beta, \gamma \in \mathbb{N}$ geeignet gewählte Konstanten.

Wir werden zeigen: Die Laufzeit der Simulation der RAM mit Laufzeitschranke $t(n)$ durch eine 2-Band-TM ist nach oben beschränkt durch $t'(n) = \alpha(n + t(n))^\beta$.

Die 2-Band-TM mit Laufzeitschranke $t'(n)$ kann nun wiederum mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden, also mit einer Laufzeitschranke der Form $t''(n) = \gamma(t'(n))^2$.

Für die Simulation der RAM auf der (1-Band-)TM ergibt sich somit eine Laufzeitschranke von

$$t''(n) = \gamma(t'(n))^2 = \gamma \left(\alpha(n + t(n))^\beta\right)^2 = \gamma \alpha^2 \cdot (n + t(n))^{2\beta}.$$ 

Diese Laufzeitschranke ist polynomiell in $n + t(n)$, weil sowohl der Term $\gamma \alpha^2$ als auch der Term $2\beta$ konstant sind.
Beobachtung

Die Klasse der Polynome ist unter Hintereinanderausführung abgeschlossen.

Mit anderen Worten:
Wenn sowohl die Abbildung $x \mapsto p(x)$ als auch die Abbildung $x \mapsto q(x)$ Polynome sind, dann ist auch die Abbildung $x \mapsto q(p(x))$ ein Polynom.
Beobachtung

Die Klasse der Polynome ist unter Hintereinanderausführung abgeschlossen.

Mit anderen Worten:
Wenn sowohl die Abbildung $x \mapsto p(x)$ als auch die Abbildung $x \mapsto q(x)$ Polynome sind, dann ist auch die Abbildung $x \mapsto q(p(x))$ ein Polynom.

Deshalb können wir eine konstante Anzahl von Simulationen, deren Zeitverlust jeweils polynomiell nach oben beschränkt ist, ineinander schachteln und erhalten dadurch wiederum eine Simulation mit polynomiell beschränktem Zeitverlust.
Beweis des Satzes

Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert. Auf Band 1 werden die einzelnen Befehle simuliert, und auf Band 2 wird der Inhalt aller verwendeten Register abgespeichert.
Beweis (1)

Beweis des Satzes

Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert. Auf Band 1 werden die einzelnen Befehle simuliert, und auf Band 2 wird der Inhalt aller verwendeten Register abgespeichert.

Das RAM-Programm $P$ bestehe aus $p$ Programmzeilen.
Beweis des Satzes

Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert. Auf Band 1 werden die einzelnen Befehle simuliert, und auf Band 2 wird der Inhalt aller verwendeten Register abgespeichert.

Das RAM-Programm $P$ bestehe aus $p$ Programmzeilen.

Für jede Programmzeile schreiben wir ein TM-Unterprogramm. Sei $M_i$ das Unterprogramm für Programmzeile $i$, $1 \leq i \leq p$. 
Beweis des Satzes

- Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert. Auf Band 1 werden die einzelnen Befehle simuliert, und auf Band 2 wird der Inhalt aller verwendeten Register abgespeichert.
- Das RAM-Programm $P$ bestehe aus $p$ Programmzeilen.
- Für jede Programmzeile schreiben wir ein TM-Unterprogramm. Sei $M_i$ das Unterprogramm für Programmzeile $i$, $1 \leq i \leq p$.
- Ausserdem spezifizieren wir ein Unterprogramm $M_0$ für die Initialisierung der TM und $M_{p+1}$ für die Aufbereitung der Ausgabe des Ergebnisses.
Abspeichern der *RAM-Konfiguration* auf der TM:
Beweis (2)

Abspeichern der *RAM-Konfiguration* auf der TM:

- Den Befehlszähler kann die TM im Zustand abspeichern, da die Länge des RAM-Programms konstant ist.
Abspeichern der RAM-Konfiguration auf der TM:

- Den Befehlszähler kann die TM im Zustand abspeichern, da die Länge des RAM-Programms konstant ist.
- Die Registerinhalte werden wie folgt auf Band 2 abgespeichert:

\[
\# \bin(c(0)) \# \bin(i_1) \# \bin(c(i_1)) \# \ldots \# \bin(i_m) \# \bin(c(i_m))
\]

wobei 0, i_1, \ldots, i_m die Indizes der benutzten Register sind.

Beobachtung

Der Platzbedarf auf Band 2 ist durch \(O(n + t(n))\) beschränkt, weil die RAM für jedes neue Bit, das sie erzeugt, mindestens eine Zeiteinheit benötigt.
Beweis (2)

Abspeichern der \textit{RAM-Konfiguration} auf der TM:

- Den Befehlszähler kann die TM im Zustand abspeichern, da die Länge des RAM-Programms konstant ist.
- Die Registerinhalte werden wie folgt auf Band 2 abgespeichert:

\[
\#\#0\# \text{bin}(c(0))\#\# \text{bin}(i_1)\# \text{bin}(c(i_1))\#\# \ldots
\]

\[
\ldots\#\# \text{bin}(i_m)\# \text{bin}(c(i_m))\#\#,\]

wobei $0, i_1, \ldots, i_m$ die Indizes der benutzten Register sind.

Beobachtung

\textit{Der Platzbedarf auf Band 2 ist durch $O(n + t(n))$ beschränkt, weil die RAM für jedes neue Bit, das sie erzeugt, mindestens eine Zeiteinheit benötigt.}
Beweis (3)

Rechenschritt für Rechenschritt simuliert die TM nun die Konfigurationsveränderungen der RAM.
Beweis (3)

Rechenschritt für Rechenschritt simuliert die TM nun die Konfigurationsveränderungen der RAM.

Dazu ruft die TM das im Programmzähler $b$ angegebene Unterprogramm $M_b$ auf.

Das Unterprogramm $M_b$

- kopiert den Inhalt der in Programmzeile $b$ angesprochenen Register auf Band 1,
- führt die notwendigen Operationen auf diesen Registerinhalten durch,
- kopiert dann das Ergebnis in das in Zeile $b$ angegebene Register auf Band 2 zurück, und
- aktualisiert zuletzt den Programmzähler $b$. 
Beweis (4)

Laufzeitanalyse:

Die Initialisierung erfordert Zeit $O(n)$. 

□
Laufzeitanalyse:

Die Initialisierung erfordert Zeit $O(n)$.

Alle Unterprogramme haben eine Laufzeit, die polynomiell in der Länge des aktuellen Wortes auf Band 2 beschränkt ist. Also: Eine Laufzeit polynomiell in $n + t(n)$. 
Laufzeitanalyse:

Die Initialisierung erfordert Zeit $O(n)$.

Alle Unterprogramme haben eine Laufzeit, die polynomiell in der Länge des aktuellen Wortes auf Band 2 beschränkt ist. Also: Eine Laufzeit polynomiell in $n + t(n)$.

Somit ist auch die Gesamtlaufzeit der Simulation polynomiell in $n + t(n)$ beschränkt.
Laufzeitanalyse:

Die Initialisierung erfordert Zeit $O(n)$.

Alle Unterprogramme haben eine Laufzeit, die polynomiell in der Länge des aktuellen Wortes auf Band 2 beschränkt ist. Also: Eine Laufzeit polynomiell in $n + t(n)$.

Somit ist auch die Gesamtlaufzeit der Simulation polynomiell in $n + t(n)$ beschränkt.

□
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

7: LOAD 3
8: MULT 1
9: STORE 3

Simulierte Turingmaschine

0: c(5)
1: c(4)
2: c(3)
3: c(2)
4: c(1)
Simulierte Registermaschine $M$

Simulierende Turingmaschine

Programm $M_8$: Multiplikation
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Kopiere $c(0)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

8: MULT 1
7: LOAD 3
9: STORE 3

Simulierende Turingmaschine

Kopiere $c(0)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Kopiere $c(0)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Kopiere $c(0)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Kopiere $c(0)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Kopiere $c(1)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Kopiere $c(1)$
Simulierende Turingmaschine

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Kopiere $c(1)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine \( M \)

Simulierende Turingmaschine

Kopiere \( c(1) \)
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Kopiere $c(1)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Multiplizieren
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

BuK/WS 2017 VL-04: Registermaschinen
Simuliertes Registermaschinen $M$

Simulierende Turingmaschine

Simuliertes Registermaschinen $M$

Simulierende Turingmaschine
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Suche $c(0)$
Simulierende Turingmaschine

Simulierte Registermaschine $M$

Update $c(0)$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Update $c(0)$
Simulierende Turingmaschine $M$

Simulierte Registermaschine $M$

Simulierende Turingmaschine

$\ldots$ B 1 0 0 B B B B B $\ldots$

$\ldots$ B # # 0 # 1 0 # # 1 0 # # 0 1 0 # # # B B B $\ldots$
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

Lücke erzeugen
Simulierende Turingmaschine

Simulierte Registermaschine $M$

---

Lücke erzeugen
Simulation RAM durch TM: Illustration

Simulierte Registermaschine $M$

Simulierende Turingmaschine

BuK/WS 2017
VL-04: Registermaschinen
Simulation von TM durch RAM
Simulation TM durch RAM

Satz

Jede $t(n)$-zeitbeschränkte TM kann durch eine RAM simuliert werden, die zeitbeschränkt ist durch

- $O(t(n) + n)$ im uniformen Kostenmass und
- $O((t(n) + n) \cdot \log(t(n) + n))$ im logarithmischen Kostenmass.
O.B.d.A. nehmen wir an, es handelt sich um eine TM mit einseitig beschränktem Band, deren Zellen mit 0, 1, 2, 3, ... durchnummeriert sind. (Vgl. Übungsaufgabe)
O.B.d.A. nehmen wir an, es handelt sich um eine TM mit einseitig beschränktem Band, deren Zellen mit 0, 1, 2, 3, ... durchnummeriert sind. (Vgl. Übungsaufgabe)

Die Zustände und Zeichen werden ebenfalls durchnummeriert und mit ihren Nummern identifiziert, so dass sie in den Registern abgespeichert werden können.
O.B.d.A. nehmen wir an, es handelt sich um eine TM mit einseitig beschränktem Band, deren Zellen mit 0, 1, 2, 3, ... durchnummeriert sind. (Vgl. Übungsaufgabe)

Die Zustände und Zeichen werden ebenfalls durchnummeriert und mit ihren Nummern identifiziert, so dass sie in den Registern abgespeichert werden können.

Register 1 speichert den Index der Kopfposition.
Beweis (1)

- O.B.d.A. nehmen wir an, es handelt sich um eine TM mit einseitig beschränktem Band, deren Zellen mit 0, 1, 2, 3, ... durchnummeriert sind. (Vgl. Übungsaufgabe)

- Die Zustände und Zeichen werden ebenfalls durchnummeriert und mit ihren Nummern identifiziert, so dass sie in den Registern abgespeichert werden können.

- Register 1 speichert den Index der Kopfposition.

- Register 2 speichert den aktuellen Zustand.
O.B.d.A. nehmen wir an, es handelt sich um eine TM mit einseitig beschränktem Band, deren Zellen mit 0, 1, 2, 3, ... durchnummersiert sind. (Vgl. Übungsaufgabe)

Die Zustände und Zeichen werden ebenfalls durchnummersiert und mit ihren Nummern identifiziert, so dass sie in den Registern abgespeichert werden können.

Register 1 speichert den Index der Kopfposition.

Register 2 speichert den aktuellen Zustand.

Die Register 3, 4, 5, 6, ... speichern die Inhalte der besuchten Bandpositionen 0, 1, 2, 3, ...
Beweis (2)

Die TM wird nun Schritt für Schritt durch die RAM simu- liert.

**Auswahl des richtigen TM-Übergangs**

Die RAM verwendet eine zweistufige if-Abfrage:

- Auf einer ersten Stufe von $|Q|$ vielen if-Abfragen wird der aktuelle Zustand selektiert.

Führt die if-Abfrage zu einem bestimmten Zustand, wird die zweite Stufe der Abfrage durchgeführt, bei der das gelesene Zeichen selektiert wird.

Je nach Ausgang der if-Abfragen aktualisiert die RAM den TM-Zustand in Register 2, die TM-Bandinschrift in Register $c(1)$ und die TM-Bandposition in Register 1.
Die TM wird nun Schritt für Schritt durch die RAM simuliert.

**Auswahl des richtigen TM-Übergangs**

Die RAM verwendet eine zweistufige *if*-Abfrage:

- Auf einer ersten Stufe von $|Q|$ vielen *if*-Abfragen wird der aktuelle Zustand selektiert.
- Für jeden möglichen Zustand gibt es dann eine zweite Stufe von $|\Gamma|$ vielen *if*-Abfragen, die das gelesene Zeichen selektieren.
Beweis (2)

Die TM wird nun Schritt für Schritt durch die RAM simuliert.

**Auswahl des richtigen TM-Übergangs**

Die RAM verwendet eine zweistufige \( if \)-Abfrage:

- Auf einer ersten Stufe von \(|Q|\) vielen \( if \)-Abfragen wird der aktuelle Zustand selektiert.
- Für jeden möglichen Zustand gibt es dann eine zweite Stufe von \(|Γ|\) vielen \( if \)-Abfragen, die das gelesene Zeichen selektieren.

**Durchführung des TM-Übergangs**

Je nach Ausgang der \( if \)-Abfragen aktualisiert die RAM

- den TM-Zustand in Register 2,
- die TM-Bandinschrift in Register \( c(1) \) und
- die TM-Bandposition in Register 1.
Simulierte Turingmaschine \( M \)

\[
\begin{array}{cccc}
\text{B} & 0 & 0 & 1 \\
\end{array} \quad \ldots
\]

\[
\begin{array}{c|cc|c}
\delta & 0 & 1 & B \\
\hline
q_1 & & & \\
q_2 & & & \\
q_3 & (q_2, 0, R)
\end{array}
\]

Simulierende Registermaschine

6: LOAD 2
7: IF \( c(0) = 1 \) THEN GOTO 23
8: IF \( c(0) = 2 \) THEN GOTO 32
9: IF \( c(0) = 3 \) THEN GOTO 41

41: INDLOAD 1
42: IF \( c(0) = 0 \) THEN GOTO 77
43: IF \( c(0) = 1 \) THEN GOTO 85

Zustand

Kopfposition
Simulierte Turingmaschine \( M \)

\[
\begin{array}{c}
\text{B} \quad 0 \quad 0 \quad 1 \quad 0 \quad B \quad B \quad B \quad \ldots
\end{array}
\]

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>0</th>
<th>1</th>
<th>( B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_2 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_3 )</td>
<td></td>
<td></td>
<td>((q_2, 0, R))</td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

6: LOAD 2
7: IF \( c(0) = 1 \) THEN GOTO 23
8: IF \( c(0) = 2 \) THEN GOTO 32
9: IF \( c(0) = 3 \) THEN GOTO 41
41: INDLOAD 1
42: IF \( c(0) = 0 \) THEN GOTO 77
43: IF \( c(0) = 1 \) THEN GOTO 85

Zustand

Hier ist ein Diagramm der Registermaschine mit den Zuständen und Kopfpositionen.
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_2, 0, R)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

6: LOAD 2
7: IF $c(0) = 1$ THEN GOTO 23
8: IF $c(0) = 2$ THEN GOTO 32
9: IF $c(0) = 3$ THEN GOTO 41
41: INDLOAD 1
42: IF $c(0) = 0$ THEN GOTO 77
43: IF $c(0) = 1$ THEN GOTO 85

Zustand
Kopfposition
Simulierte Turingmaschine $M$

1. **Simulierende Registermaschine**

   - **Zustandsübergangstable:**
     
     | $\delta$ | 0 | 1 | $B$ |
     |----------|---|---|-----|
     | $q_1$    |   |   |     |
     | $q_2$    |   |   |     |
     | $q_3$    |   |   | $(q_2, 0, R)$ |

   - **Programm:**
     
     6: LOAD 2
     7: IF $c(0) = 1$ THEN GOTO 23
     8: IF $c(0) = 2$ THEN GOTO 32
     9: IF $c(0) = 3$ THEN GOTO 41
     41: INDLOAD 1
     42: IF $c(0) = 0$ THEN GOTO 77
     43: IF $c(0) = 1$ THEN GOTO 85

   - **Registerdarstellung:**

     | Zustand | Kopfposition |
     |---------|--------------|
     | 8       | 3            |

BuK/WS 2017
 VL-04: Registermaschinen
BuK/WS 2017
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>(q&lt;sub&gt;2&lt;/sub&gt;, 0, R)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

6: LOAD 2
7: IF $c(0) = 1$ THEN GOTO 23
8: IF $c(0) = 2$ THEN GOTO 32
9: IF $c(0) = 3$ THEN GOTO 41

41: INDLOAD 1
42: IF $c(0) = 0$ THEN GOTO 77
43: IF $c(0) = 1$ THEN GOTO 85

Zustand

Kopfposition

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

BuK/WS 2017
VL-04: Registermaschinen

29/38
**Simulierte Turingmaschine** \( M \)

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>0</th>
<th>1</th>
<th>( B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td></td>
<td></td>
<td>( q_2 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td></td>
<td></td>
<td>( q_3 )</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( q_2 ), ( 0 ), ( R )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Simulierende Registermaschine**

6: LOAD 2  
7: IF \( c(0) = 1 \) THEN GOTO 23  
8: IF \( c(0) = 2 \) THEN GOTO 32  
9: IF \( c(0) = 3 \) THEN GOTO 41  
41: INDLOAD 1  
42: IF \( c(0) = 0 \) THEN GOTO 77  
43: IF \( c(0) = 1 \) THEN GOTO 85

- \( \text{Zustand} \):
  - \( 0 \) c(6)  
  - \( 1 \) c(5)  
  - \( 0 \) c(4)  
  - \( 0 \) c(3)  
  - \( 3 \) c(2)  
  - \( 5 \) c(1)

- \( \text{Kopfposition} \):
  - \( 41 \)  
  - \( 3 \)
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

6: LOAD 2
7: IF $c(0) = 1$ THEN GOTO 23
8: IF $c(0) = 2$ THEN GOTO 32
9: IF $c(0) = 3$ THEN GOTO 41
41: INDLOAD 1
42: IF $c(0) = 0$ THEN GOTO 77
43: IF $c(0) = 1$ THEN GOTO 85

Zustand

Kopfposition
Simulierte Turingmaschine $M$

\[
\begin{array}{c}
\text{B} \ 0 \ 0 \ 1 \ 0 \ B \ B \ B \ldots
\end{array}
\]

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$0$</th>
<th>$1$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_2, 0, R)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

6: LOAD 2
7: IF $c(0) = 1$ THEN GOTO 23
8: IF $c(0) = 2$ THEN GOTO 32
9: IF $c(0) = 3$ THEN GOTO 41

41: INDLOAD 1
42: IF $c(0) = 0$ THEN GOTO 77
43: IF $c(0) = 1$ THEN GOTO 85

Zustand
Kopfposition
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

6: LOAD 2
7: IF $c(0) = 1$ THEN GOTO 23
8: IF $c(0) = 2$ THEN GOTO 32
9: IF $c(0) = 3$ THEN GOTO 41

41: INDLOAD 1
42: IF $c(0) = 0$ THEN GOTO 77
43: IF $c(0) = 1$ THEN GOTO 85
Simulierte Turingmaschine $M$

$$
\begin{array}{cccccc}
B & 0 & 0 & 1 & 0 & B \ B \ B \ \cdots
\end{array}
$$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

\[\begin{array}{ccc}
0 & c(6) \\
1 & c(5) \\
0 & c(4) \\
0 & c(3) \\
3 & c(2) \\
5 & c(1)
\end{array}\]

Zustand

Kopfposition
Simulierende Turingmaschine $M$

Simulierende Registermaschine

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>($q_2, 0, R$)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

86: CLOAD 2
2: STORE 2
3: CLOAD 0
5: INDSTORE 1
1: LOAD 1
0: CADD 1
0: STORE 1
0: GOTO 6

Zustand
Kopfposition
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2  
86: STORE 2  
87: CLOAD 0  
88: INDSTORE 1  
89: LOAD 1  
90: CADD 1  
91: STORE 1  
92: GOTO 6

Kopfposition

Zustand

$c(1)$  
$c(2)$  
$c(3)$  
$c(4)$  
$c(5)$  
$c(6)$
### Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>

#### Simulierende Registermaschine

85: CLOAD 2  
86: STORE 2  
87: CLOAD 0  
88: INDSTORE 1  
89: LOAD 1  
90: CADD 1  
91: STORE 1  
92: GOTO 6

- **87**: CLOAD 0  
- **2**:  
- **5**:  
- **Zustand**:  
- **Kopfposition**: 
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

87: 2

$0$

0 2 5

Zustand

Kopfposition
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>(0, 0, $R$)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

Zustand
Kopfposition

BuK/WS 2017
VL-04: Registermaschinen
Simulierte Turingmaschine $M$

$$\begin{array}{c}
\text{B} 0 0 1 0 \text{B} \text{B} \text{B} \ldots \\
\hline
\hline
\delta & 0 & 1 & \text{B} \\
\hline
q_1 & & & \\
q_2 & & & \\
q_3 & (q_2, 0, R) & & \\
\end{array}$$

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

<table>
<thead>
<tr>
<th>Zustand</th>
<th>Kopfposition</th>
</tr>
</thead>
</table>
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>δ</th>
<th>0</th>
<th>1</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_2, 0, R)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6
Simulierende Turingmaschine $M$

\[
\begin{array}{c}
\text{B 0 0 0 0 B B B \ldots}
\end{array}
\]

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>(q_2, 0, R)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

Zustand

Kopfposition
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td>(q_2, 0, R)</td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

- 85: CLOAD 2
- 86: STORE 2
- 87: CLOAD 0
- 88: INDSTORE 1
- 89: LOAD 1
- 90: CADD 1 (Aktiver Command)
- 91: STORE 1
- 92: GOTO 6

Zustand

Kopfposition
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2  
86: STORE 2  
87: CLOAD 0  
88: INDSTORE 1  
89: LOAD 1  
90: CADD 1  
91: STORE 1  
92: GOTO 6

Zustand  
Kopfposition

BuK/WS 2017  
VL-04: Registermaschinen
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td></td>
<td>$(q_2, 0, R)$</td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

Zustand

Kopfposition

BuK/WS 2017

VL-04: Registermaschinen

29/38
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_2, 0, R)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6

Zustand

Kopfposition
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(q_2, 0, R)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2  
86: STORE 2  
87: CLOAD 0  
88: INDSTORE 1  
89: LOAD 1  
90: CADD 1  
91: STORE 1  
92: GOTO 6

Zustand  
Kopfposition
Simulierte Turingmaschine $M$

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>0</th>
<th>1</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$q_2, 0, R$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Simulierende Registermaschine

85: CLOAD 2
86: STORE 2
87: CLOAD 0
88: INDSTORE 1
89: LOAD 1
90: CADD 1
91: STORE 1
92: GOTO 6
**Beweis (3a)**

Laufzeitanalyse im uniformen Kostenmodell:

- Die Initialisierung kann in Zeit $O(n)$ durchgeführt werden.
Laufzeitanalyse im uniformen Kostenmodell:

- Die Initialisierung kann in Zeit $O(n)$ durchgeführt werden.
- Die Simulation jedes einzelnen TM-Schrittes hat konstante Laufzeit.
Laufzeitanalyse im uniformen Kostenmodell:

- Die Initialisierung kann in Zeit $O(n)$ durchgeführt werden.
- Die Simulation jedes einzelnen TM-Schrittes hat konstante Laufzeit.
- Insgesamt ist die Simulationszeit somit $O(n + t(n))$. 
Beweis (3b)

Laufzeitanalyse im logarithmischen Kostenmodell:

- Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen.
Beweis (3b)

Laufzeitanalyse im logarithmischen Kostenmodell:

- Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen.
- Zustände und Zeichen haben eine konstante Kodierungslänge.
Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen. Zustände und Zeichen haben eine konstante Kodierungslänge. Die Bandpositionen, die während der Simulation angesprochen werden, sind durch \( \max\{n, t(n)\} \leq n + t(n) \) beschränkt. Die Kodierungslänge dieser Positionen ist also \( O(\log(t(n) + n)) \).
Beweis (3b)

Laufzeitanalyse im logarithmischen Kostenmodell:

- Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen.
- Zustände und Zeichen haben eine konstante Kodierungslänge.
- Die Bandpositionen, die während der Simulation angesprochen werden, sind durch $\max\{n, t(n)\} \leq n + t(n)$ beschränkt. Die Kodierungslänge dieser Positionen ist also $O(\log(t(n) + n))$.
- Damit kann die Simulation jedes einzelnen TM-Schrittes in Zeit $O(\log(t(n) + n))$ durchgeführt werden.
Beweis (3b)

Laufzeitanalyse im logarithmischen Kostenmodell:

- Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen.
- Zustände und Zeichen haben eine konstante Kodierungslänge.
- Die Bandpositionen, die während der Simulation angesprochen werden, sind durch $\max\{n, t(n)\} \leq n + t(n)$ beschränkt. Die Kodierungslänge dieser Positionen ist also $O(\log(t(n) + n))$.
- Damit kann die Simulation jedes einzelnen TM-Schrittes in Zeit $O(\log(t(n) + n))$ durchgeführt werden.
- Insgesamt ergibt sich somit eine Simulationszeit von $O((t(n) + n)\log(t(n) + n))$. 

□
Zusammenfassung

Die Mehrband-TM kann mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden.
Zusammenfassung

- Die Mehrband-TM kann mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden.
- TM und RAM (im logarithmischem Kostenmodell) können sich gegenseitig mit polynomiellem Zeitverlust simulieren.
Die Mehrband-TM kann mit quadratischem Zeitverlust durch eine (1-Band-)TM simuliert werden.

TM und RAM (im logarithmischen Kostenmodell) können sich gegenseitig mit polynomiellem Zeitverlust simulieren.

Wenn es uns also „nur“ um Fragen der Berechenbarkeit von Problemen (oder um ihre Lösbarkeit in polynomieller Zeit) geht, können wir wahlweise auf die TM, die Mehrband-TM oder die RAM zurückgreifen.
Die Church-Turing These
Wikipedia: Alonzo Church was an American mathematician and logician who made major contributions to mathematical logic and the foundations of theoretical computer science.

He is best known for the lambda calculus, Church-Turing thesis, proving the undecidability of the Entscheidungsproblem, Frege-Church ontology, and the Church-Rosser theorem.
Kein bekanntes “vernünftiges” Rechnermodell ist mächtiger als die TM:
• Lambda-Kalkül von Alonzo Church: gleich mächtig wie TM
• Kanonische Systeme von Emil Post: gleich mächtig wie TM
• $\mu$-rekursive Funktionen von Kurt Gödel: gleich mächtig wie TM
Kein bekanntes “vernünftiges” Rechnermodell ist mächtiger als die TM:
• Lambda-Kalkül von Alonzo Church: gleich mächtig wie TM
• Kanonische Systeme von Emil Post: gleich mächtig wie TM
• $\mu$-rekursive Funktionen von Kurt Gödel: gleich mächtig wie TM

Alonzo Church und Alan Turing formulierten die folgende These in den 1930er Jahren:

**Church-Turing These**

*Die Klasse der TM-berechenbaren Funktionen stimmt mit der Klasse der “intuitiv berechenbaren” Funktionen überein.*
Kein bekanntes „vernünftiges“ Rechnermodell ist mächtiger als die TM:
• Lambda-Kalkül von Alonzo Church: gleich mächtig wie TM
• Kanonische Systeme von Emil Post: gleich mächtig wie TM
• $\mu$-rekursive Funktionen von Kurt Gödel: gleich mächtig wie TM

Alonzo Church und Alan Turing formulierten die folgende These in den 1930er Jahren:

**Church-Turing These**

*Die Klasse der TM-berechenbaren Funktionen stimmt mit der Klasse der “intuitiv berechenbaren” Funktionen überein.*

Wir werden deshalb nicht mehr von TM-berechenbaren Funktionen sprechen, sondern allgemein von berechenbaren Funktionen.
Gleichbedeutend verwenden wir den Begriff rekursive Funktion bzw. rekursive oder auch entscheidbare Sprache.
Zum folgenden Inhalt der Vorlesung

Jetzt sind wir endlich bereit, die zentrale Fragestellung des ersten Teils dieser Vorlesung formal zu fassen.

In der Berechenbarkeitstheorie ...
...
... wird untersucht, welche Probleme entscheidbar sind, d.h. welche Probleme durch einen Algorithmus – ohne jegliche Einschränkungen der Rechenzeit oder des Speicherplatzes – gelöst werden können.
Harvard Mark I

Users: Waldir, Topory/Wikimedia Commons/CC-BY-SA-3.0
Harvard Mark I
Organisatorisches

• Nächste Vorlesung:
  Donnerstag, Oktober 26, 12:15–13:45 Uhr, Aula

• Achtung:
  In der Woche Oct-30 bis Nov-3 entfallen alle Tutorien!

• Webseite:
  http://algo.rwth-aachen.de/Lehre/WS1718/BuK.php