Vorlesung 4
Registermaschine (RAM), Church-Turing-These

Wdh.: k-Band- vs 1-Band-TM

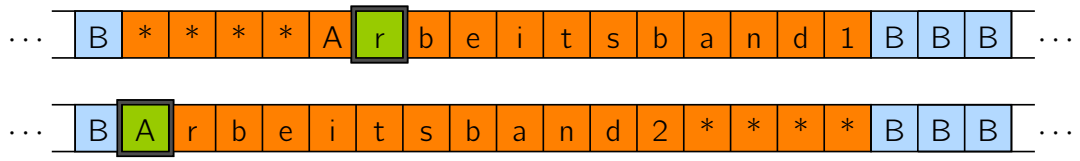
Satz

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.

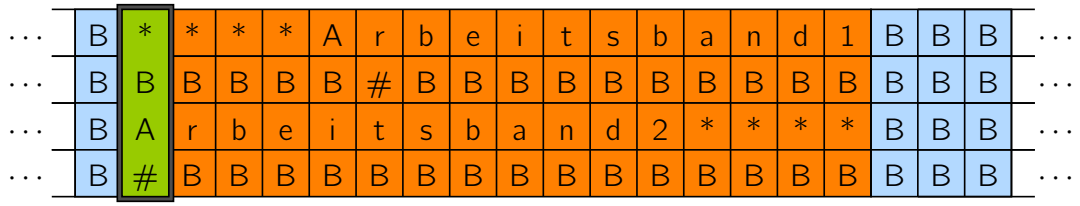
Wdh.: k-Band- vs 1-Band-TM

Satz

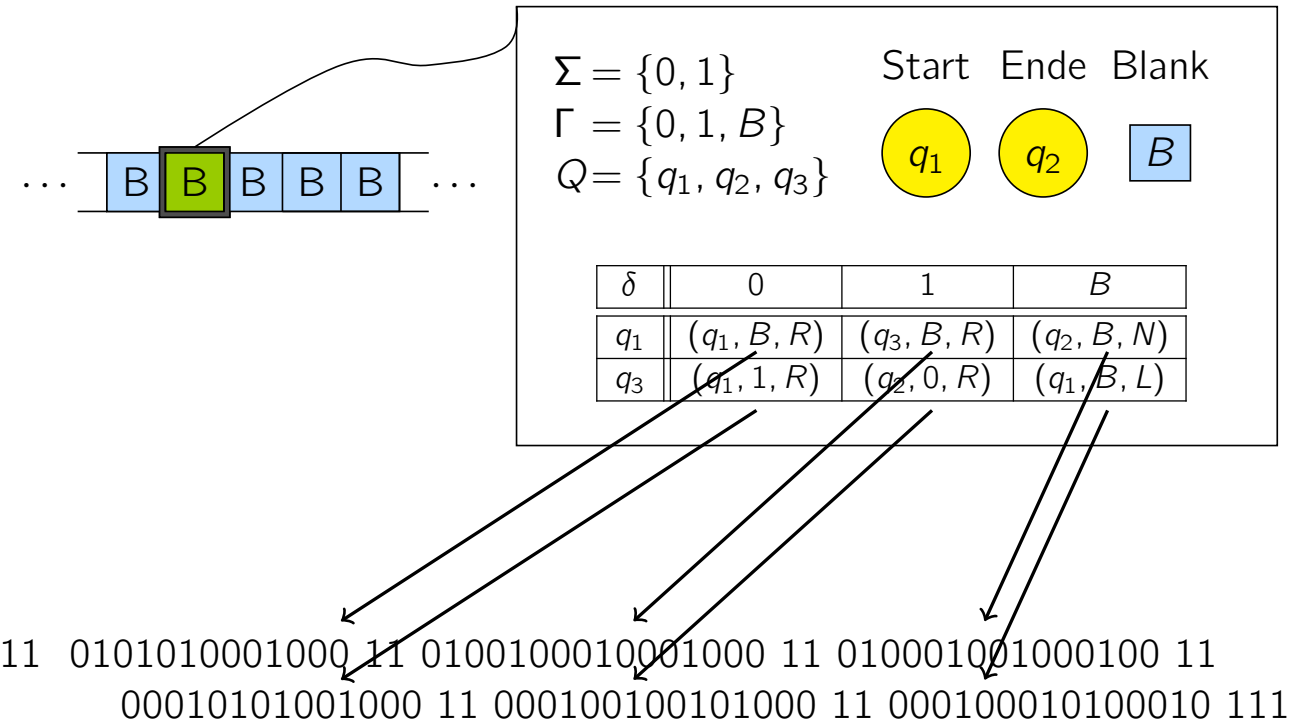
Eine k-Band-TM M, die mit Rechenzeit $t(n)$ und Platz $s(n)$ auskommt, kann von einer (1-Band-)TM M' mit Zeitbedarf $O(t^2(n))$ und Platzbedarf $O(s(n))$ simuliert werden.



Simuliert durch

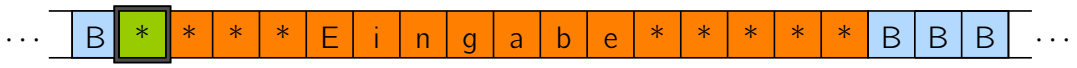


Wdh.: Gödelnummer $\langle M \rangle$

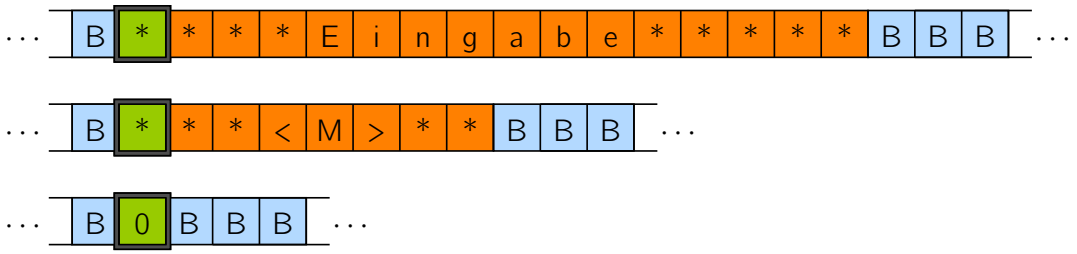


Wdh.: Universelle TM

simulierte Turingmaschine M



Initialisierung der universellen Maschine U



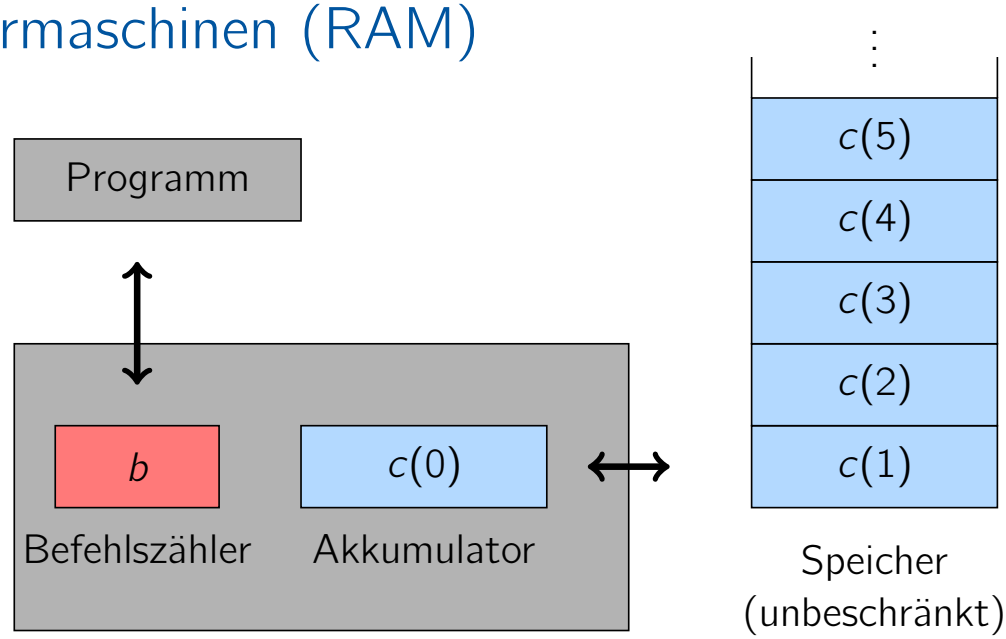
Wdh.: Universelle TM

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.

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 (wobei ? aus {=,<,<=,>,>=} ist),
END

Erläuterung einiger ausgewählter RAM-Befehle

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

Erläuterung einiger ausgewählter RAM-Befehle

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

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

Vorlesung BuK im WS 22/23, M. Grohe **Seite 94** Version 12. Oktober 2022

Erläuterung einiger ausgewählter RAM-Befehle

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

Erläuterung einiger ausgewählter RAM-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:	$c(i) := c(0)$,	$b := b + 1$;

Erläuterung einiger ausgewählter RAM-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:	$c(c(i)) := c(0)$,	$b := b + 1$;

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(i)$,	$b := b + 1$;

Vorlesung BuK im WS 22/23, M. Grohe Seite 94 Version 12. Oktober 2022

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + i$,	$b := b + 1$;

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(c(i))$,	$b := b + 1$;

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(c(i))$,	$b := b + 1$;
	$\vdots$	

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(c(i))$,	$b := b + 1$;
	$\vdots$	
SUB i:	$c(0) := \max\{0, c(0) - c(i)\}$	$b := b + 1$;

Vorlesung BuK im WS 22/23, M. Grohe **Seite 94** Version 12. Oktober 2022

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(c(i))$,	$b := b+1$;
	$\vdots$	
SUB i:	$c(0) := \max\{0, c(0) - c(i)\}$	$b := b+1$;
	$\vdots$	

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(c(i))$,	$b := b+1$;
	$\vdots$	
SUB i:	$c(0) := \max\{0, c(0) - c(i)\}$	$b := b+1$;
	$\vdots$	
DIV i:	$c(0) := \begin{cases} \lfloor c(0)/c(i) \rfloor & \text{falls } c(i) \neq 0, \\ 0 & \text{sonst} \end{cases}$,	$b := b+1$;
	$\vdots$	

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(c(i))$,	$b := b+1$;
	$\vdots$	
SUB i:	$c(0) := \max\{0, c(0) - c(i)\}$	$b := b+1$;
	$\vdots$	
DIV i:	$c(0) := \begin{cases} \lfloor c(0)/c(i) \rfloor & \text{falls } c(i) \neq 0, \\ 0 & \text{sonst} \end{cases}$,	$b := b+1$;
	$\vdots$	
GOTO j:		$b := j$

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(c(i))$,	$b := b+1$;
	$\vdots$	
SUB i:	$c(0) := \max\{0, c(0) - c(i)\}$	$b := b+1$;
	$\vdots$	
DIV i:	$c(0) := \begin{cases} \lfloor c(0)/c(i) \rfloor & \text{falls } c(i) \neq 0, \\ 0 & \text{sonst} \end{cases}$,	$b := b+1$;
	$\vdots$	
GOTO j:		$b := j$
IF $c(0) = x$ GOTO j:	$b := j$ falls $c(0) = x$, sonst $b := b+1$;	

Erläuterung einiger ausgewählter RAM-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:	$c(0) := c(0) + c(c(i))$,	$b := b + 1$;
	$\vdots$	
SUB i:	$c(0) := \max\{0, c(0) - c(i)\}$	$b := b + 1$;
	$\vdots$	
DIV i:	$c(0) := \begin{cases} \lfloor c(0)/c(i) \rfloor & \text{falls } c(i) \neq 0, \\ 0 & \text{sonst} \end{cases}$,	$b := b + 1$;
	$\vdots$	
GOTO j:		$b := j$

IF $c(0) = x$ GOTO j: $b := j$ falls $c(0) = x$, sonst $b := b + 1$;

END.

Funktionsweise der RAM

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

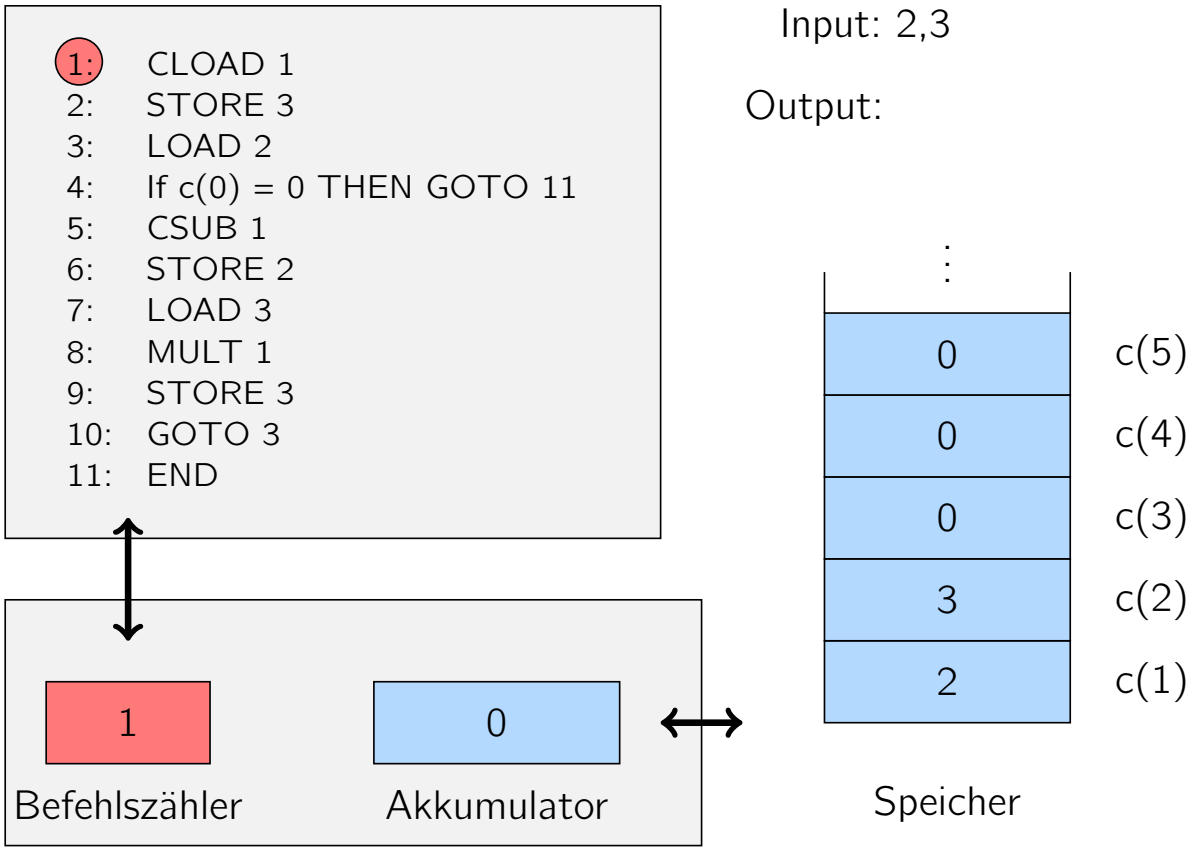
Funktionsweise der RAM

- Der Speicher der RAM ist unbeschränkt und besteht aus dem Akkumulator $c(0)$ und den Registern $c(1), c(2), c(3), \ldots$
- Die Inhalte der Register sind natürliche Zahlen, die beliebig groß sein können.
- Die Eingabe besteht ebenfalls aus natürlichen Zahlen, die initial in den ersten Registern abgespeichert sind.
- Der Befehlszähler startet mit dem Wert 1. Ausgeführt wird jeweils der Befehl in derjenigen Zeile, auf die der Befehlszähler verweist.

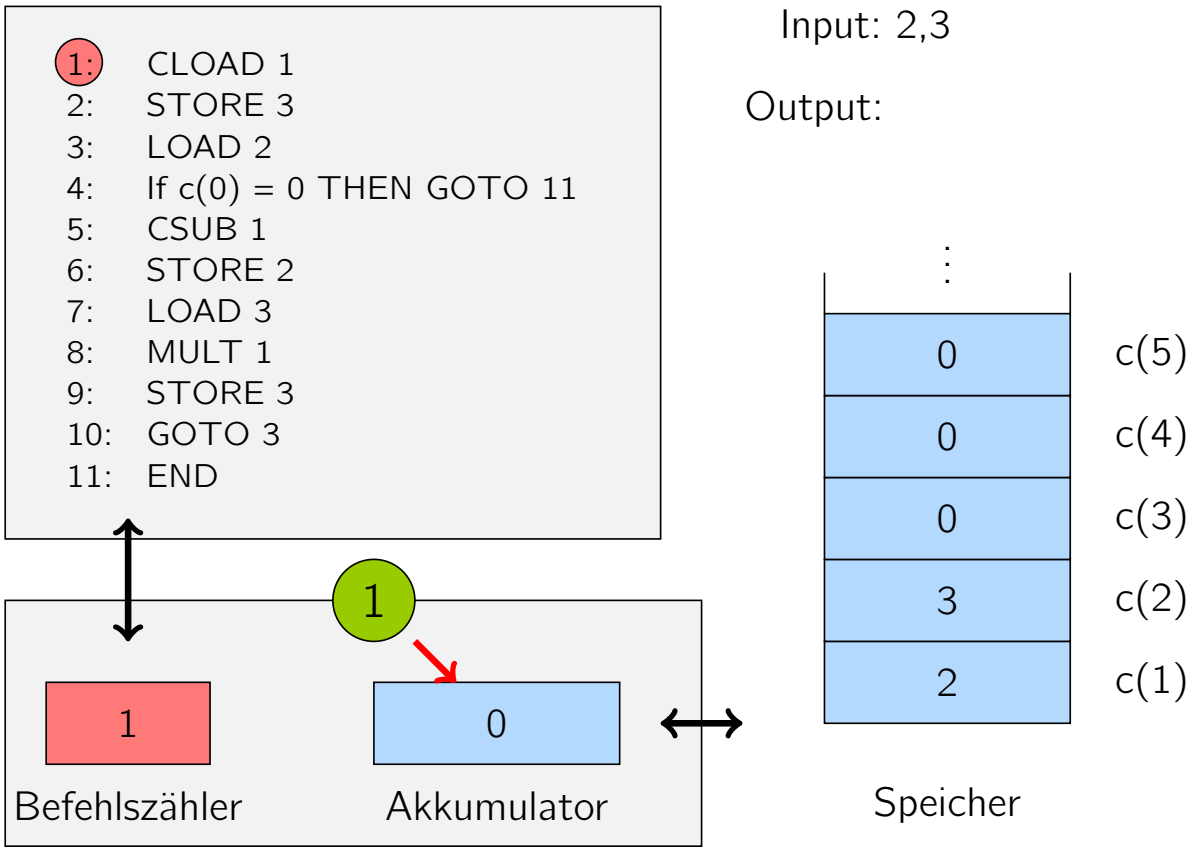
Funktionsweise der RAM

- Der Speicher der RAM ist unbeschränkt und besteht aus dem Akkumulator $c(0)$ und den Registern $c(1), c(2), c(3), \ldots$
- Die Inhalte der Register sind natürliche Zahlen, die beliebig groß sein können.
- Die Eingabe besteht ebenfalls aus natürlichen Zahlen, die initial in den ersten Registern abgespeichert sind.
- 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.

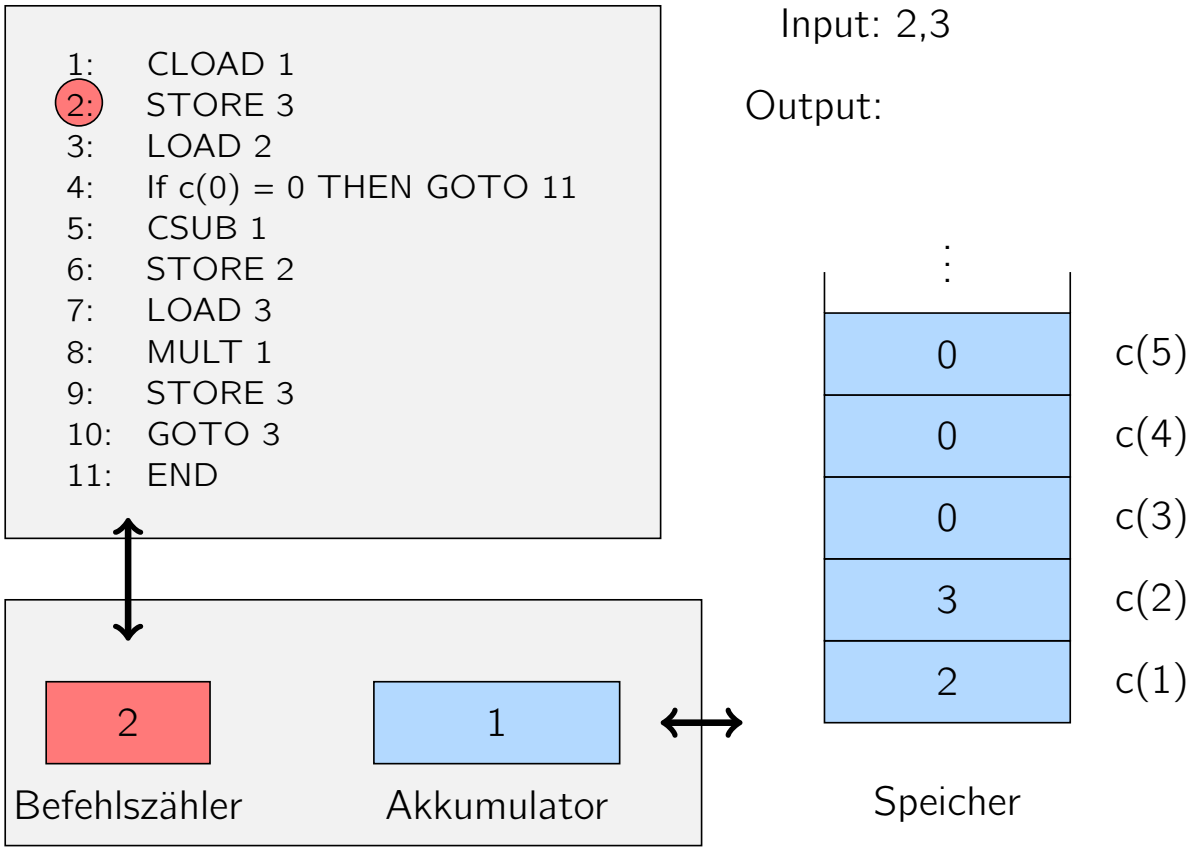
Beispielprogramm für die RAM



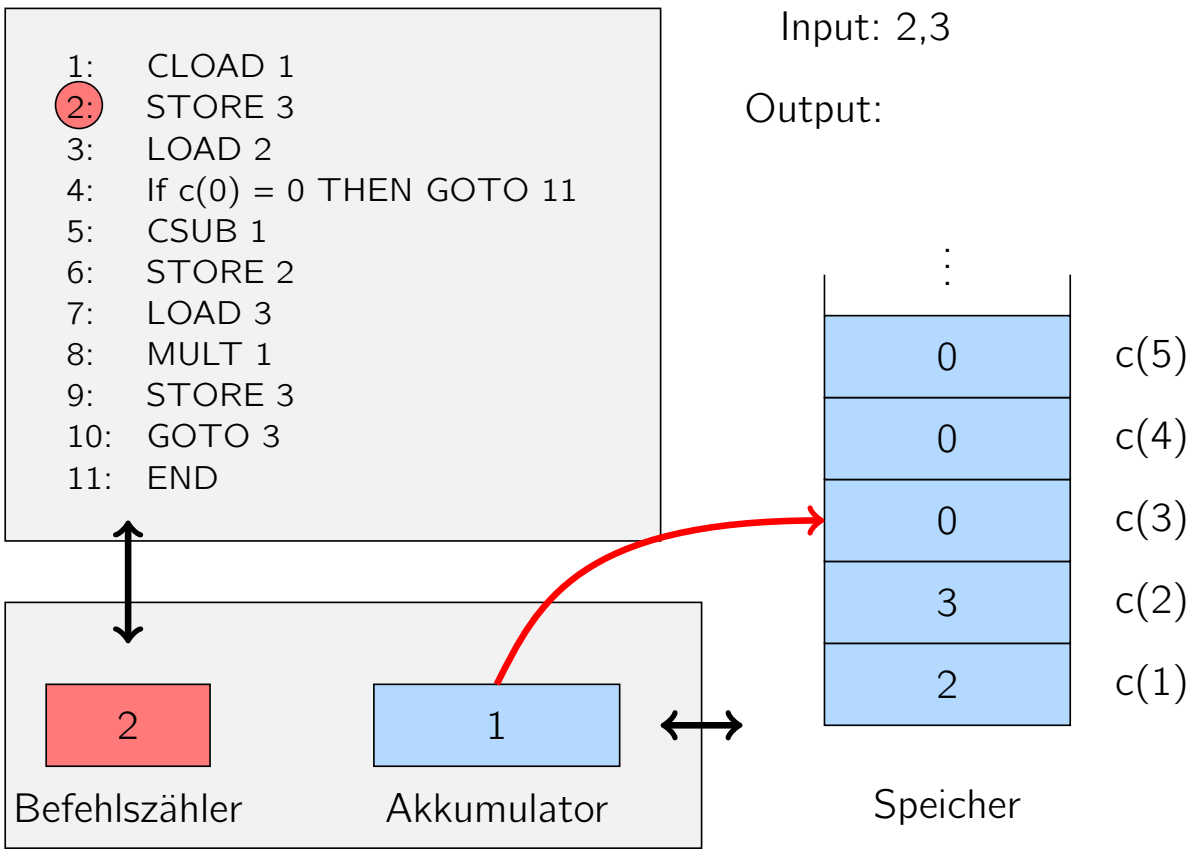
Beispielprogramm für die RAM



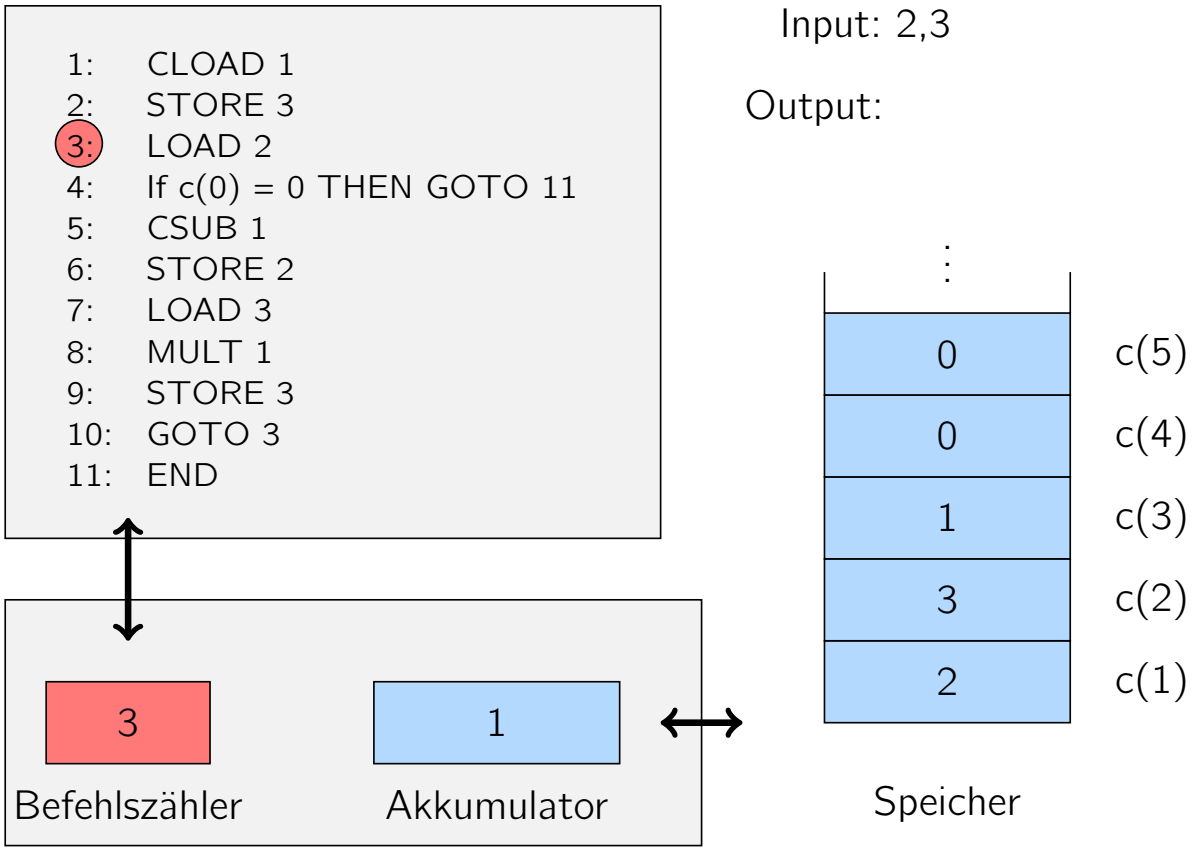
Beispielprogramm für die RAM



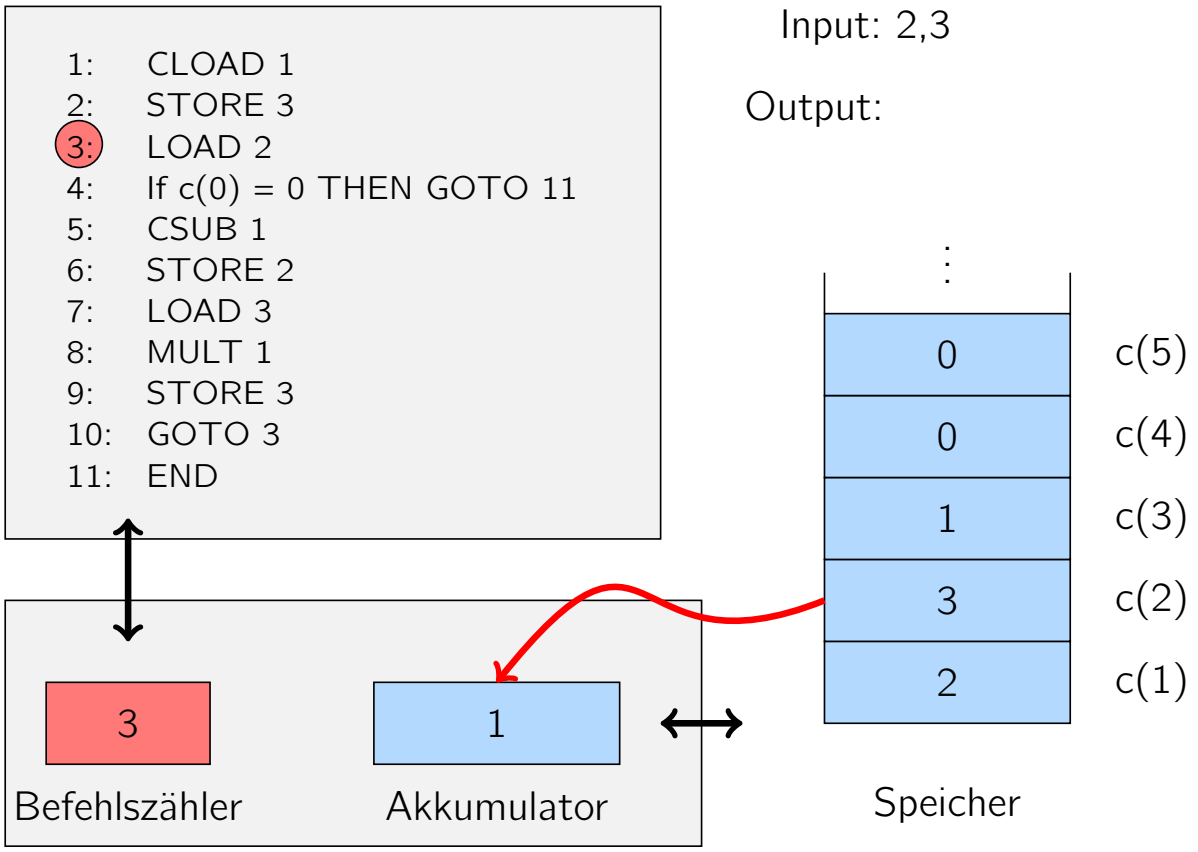
Beispielprogramm für die RAM



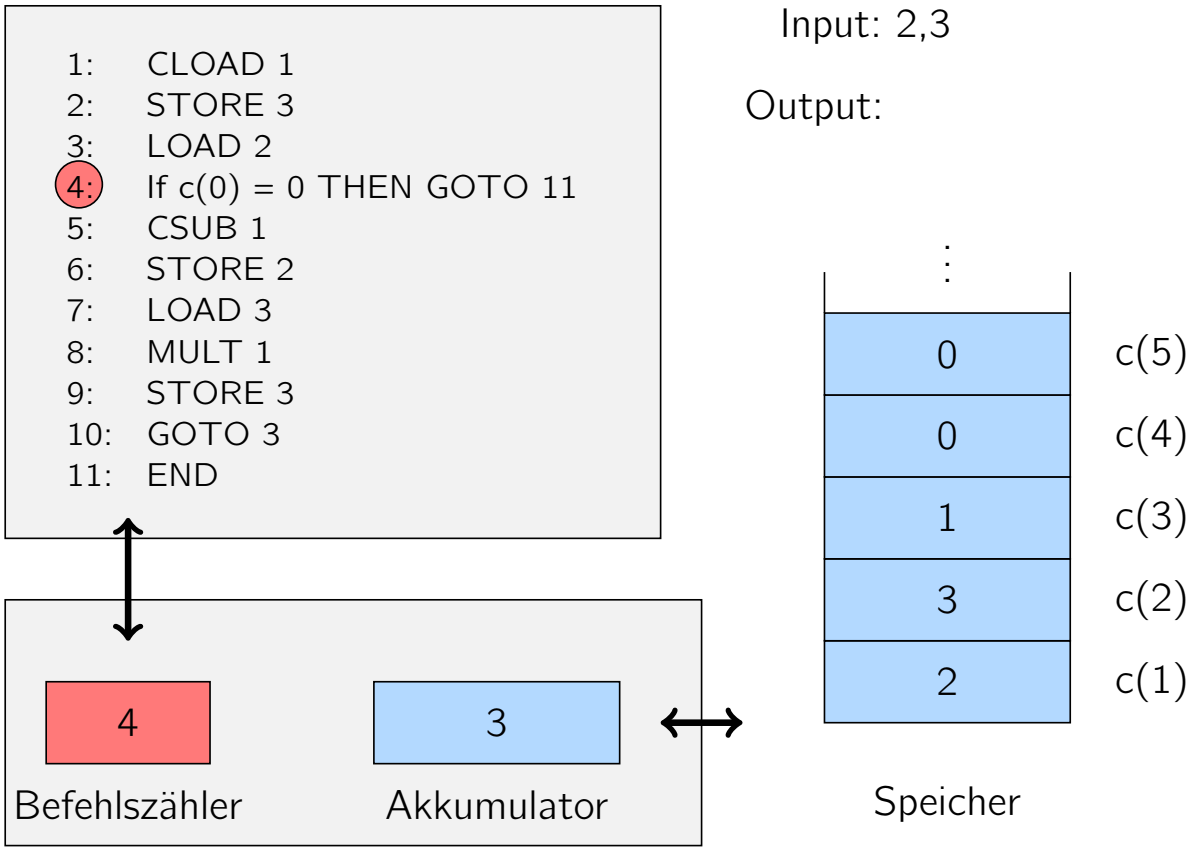
Beispielprogramm für die RAM



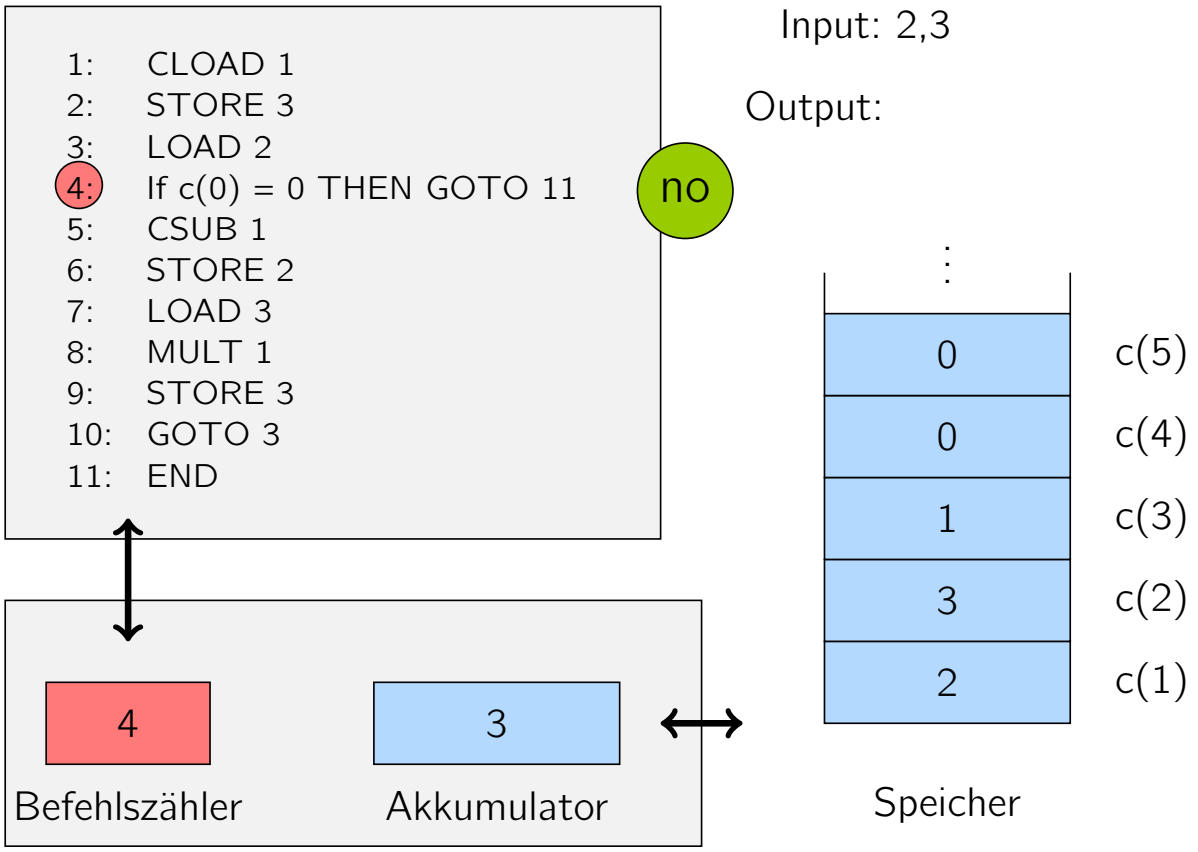
Beispielprogramm für die RAM



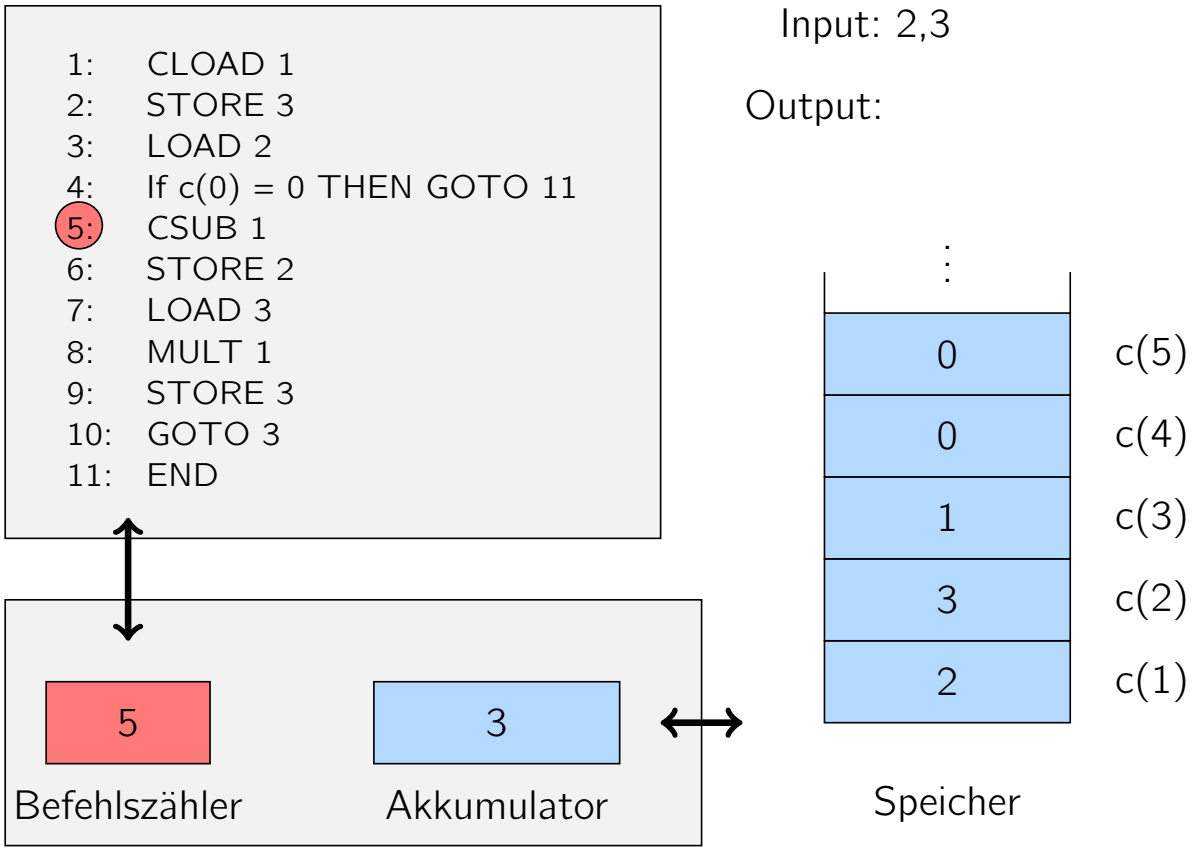
Beispielprogramm für die RAM



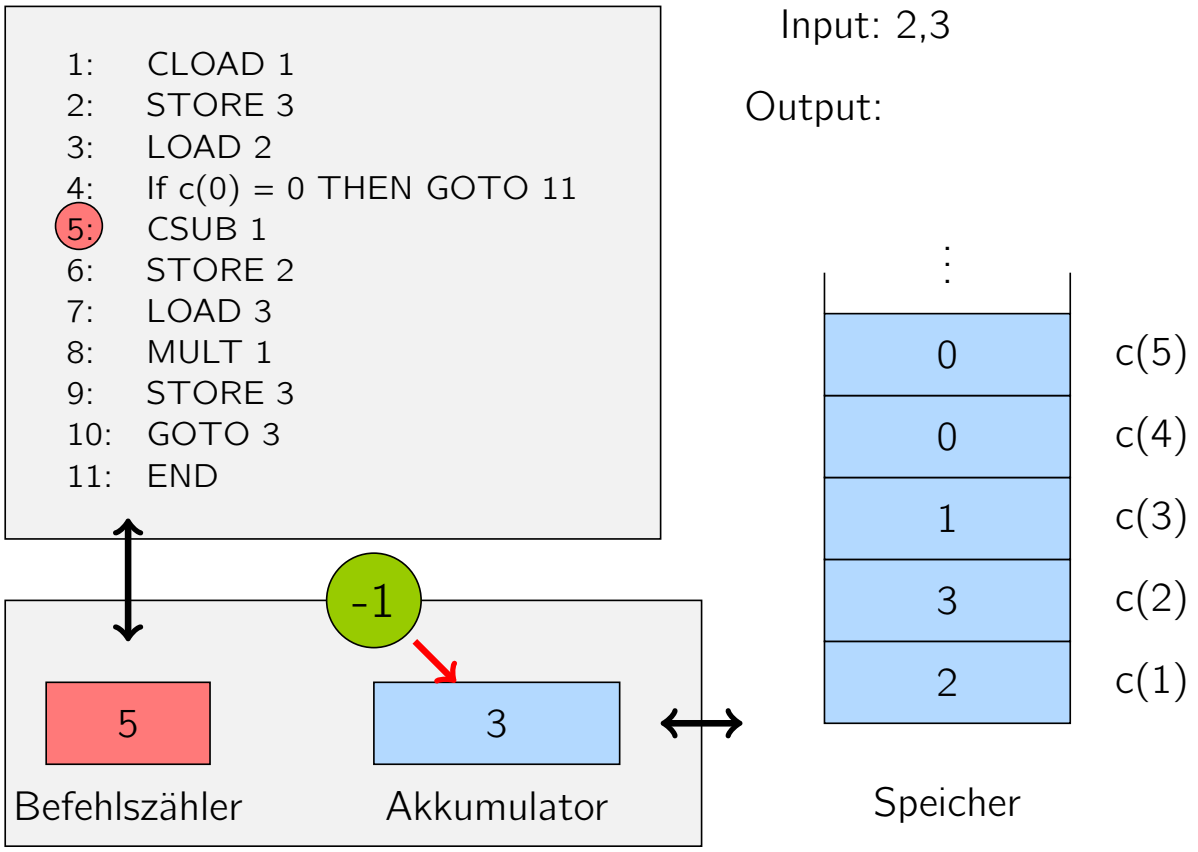
Beispielprogramm für die RAM



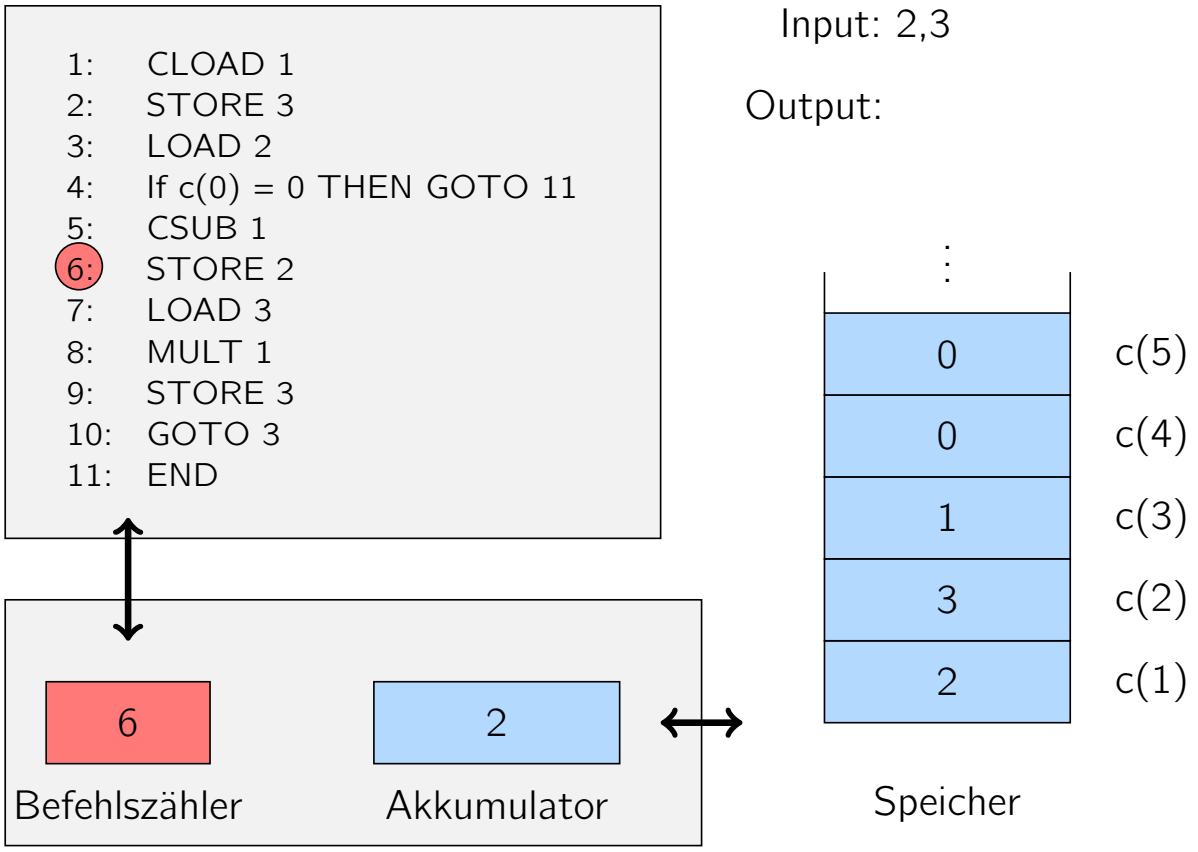
Beispielprogramm für die RAM



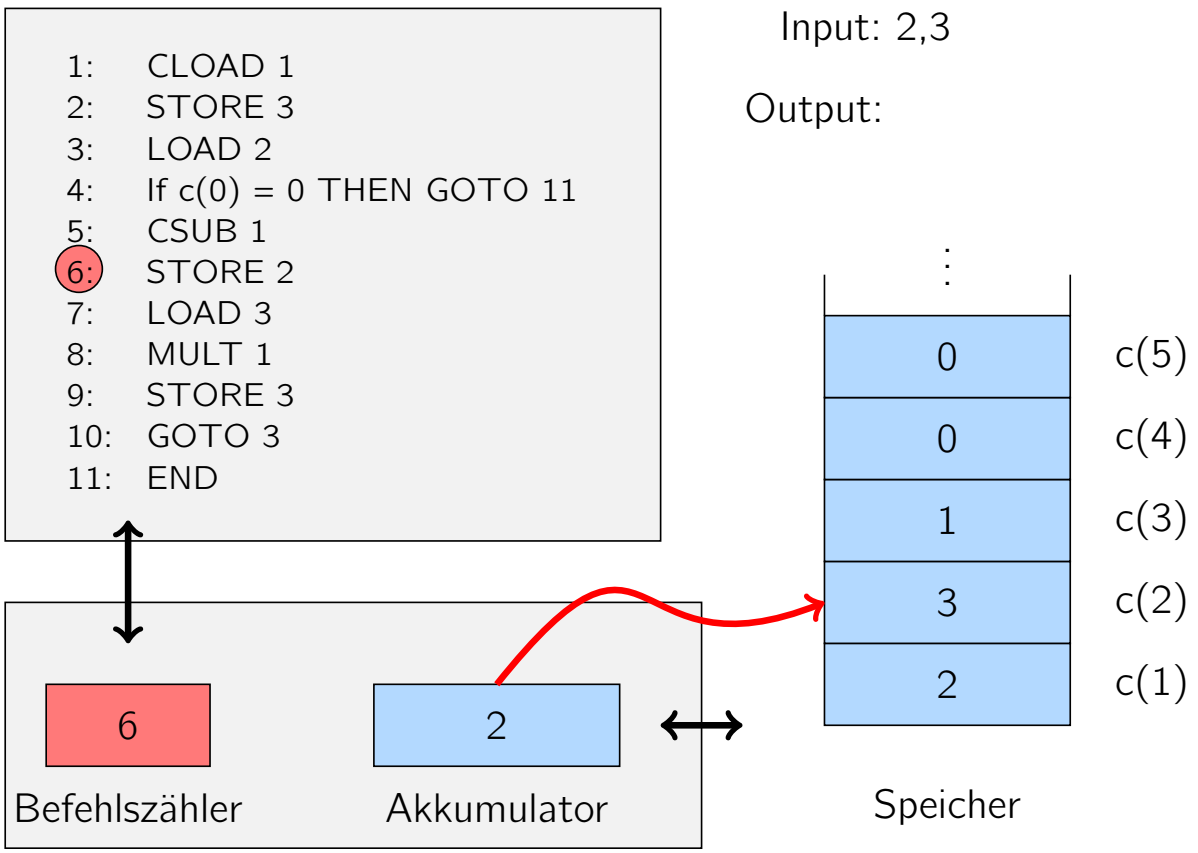
Beispielprogramm für die RAM



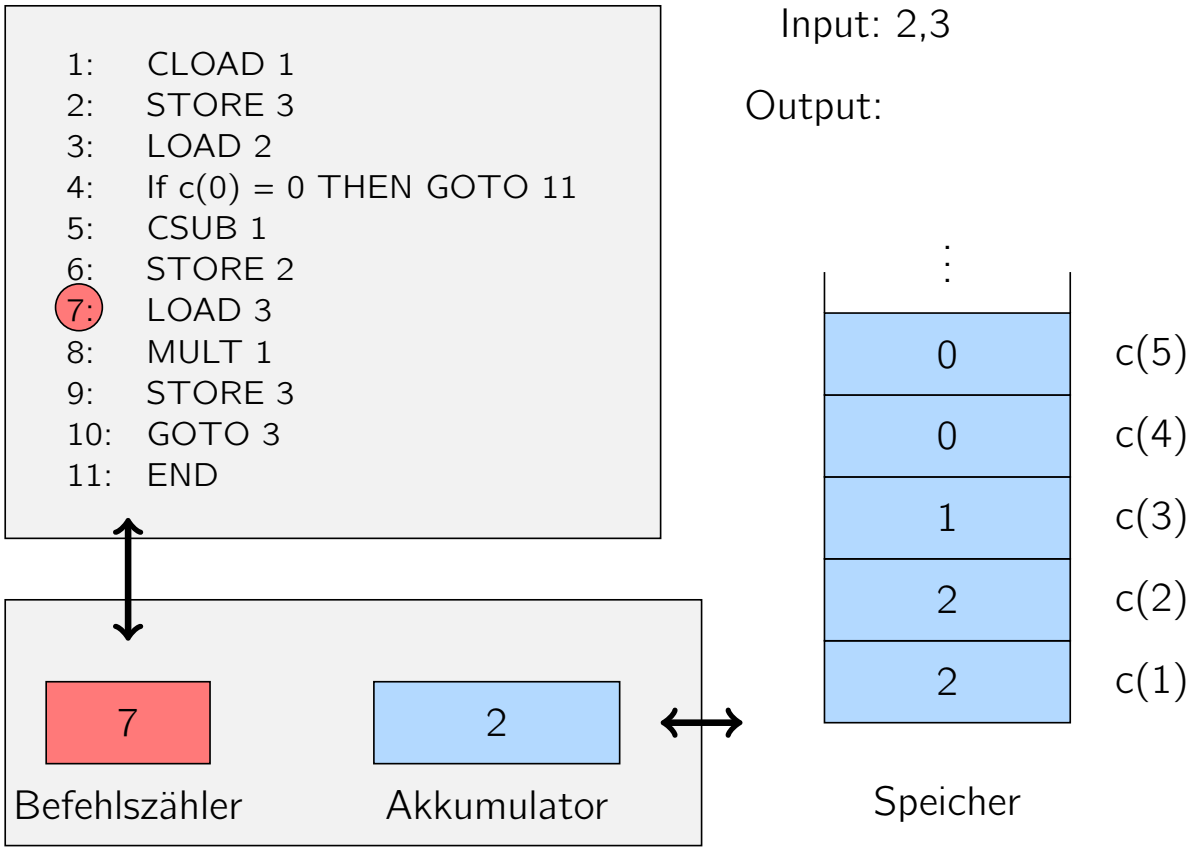
Beispielprogramm für die RAM



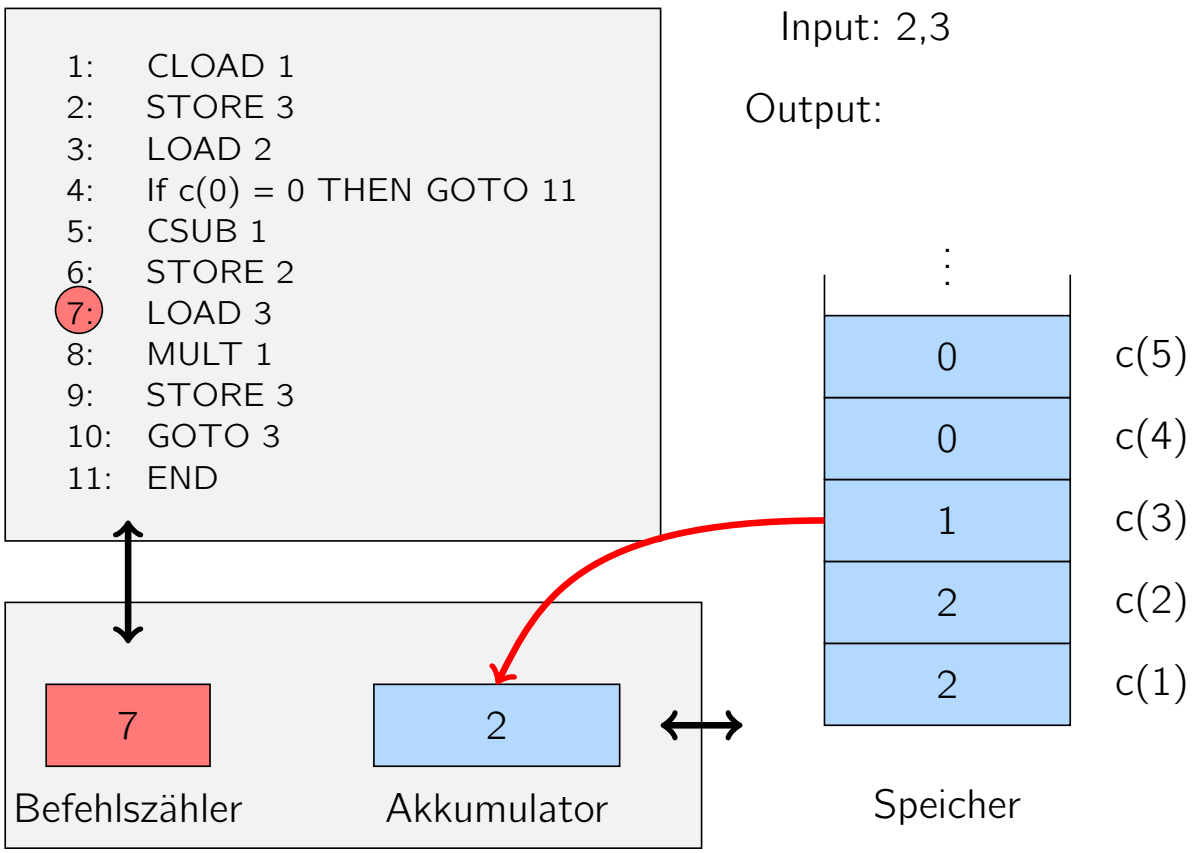
Beispielprogramm für die RAM



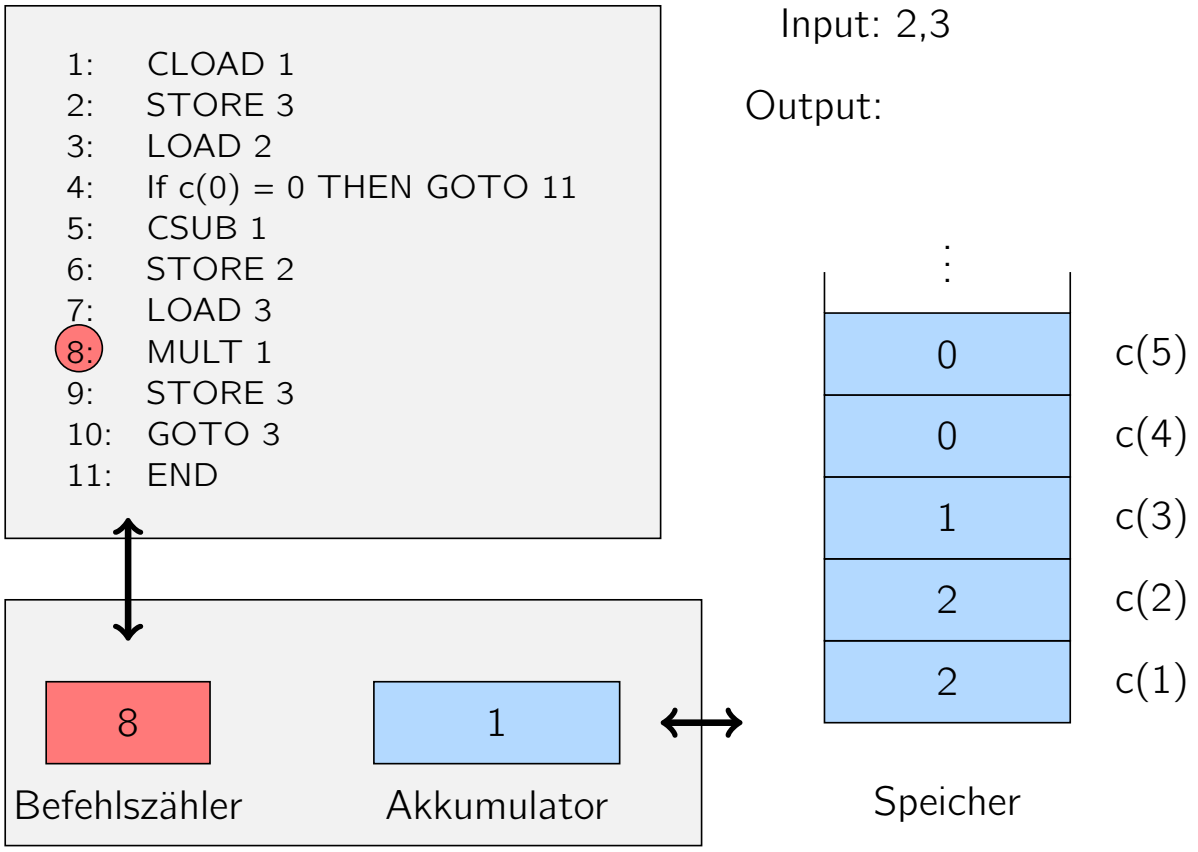
Beispielprogramm für die RAM



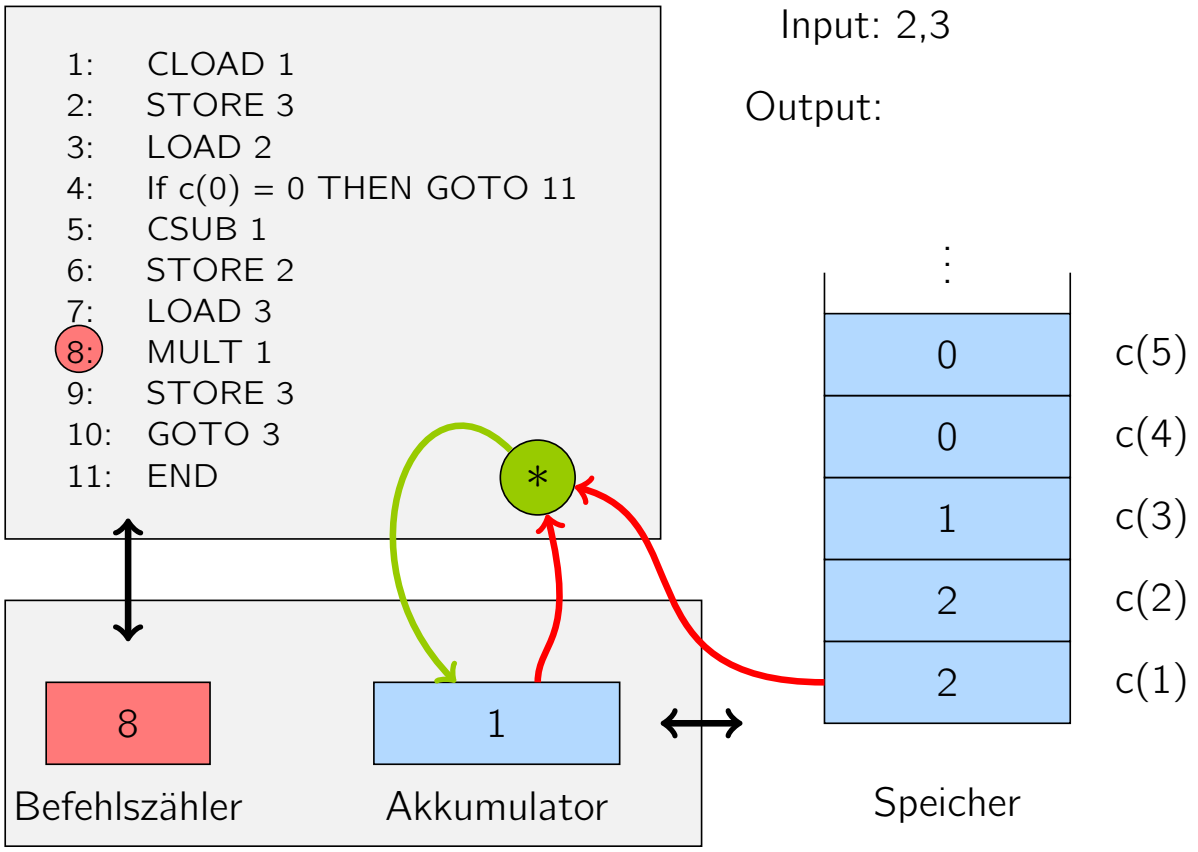
Beispielprogramm für die RAM



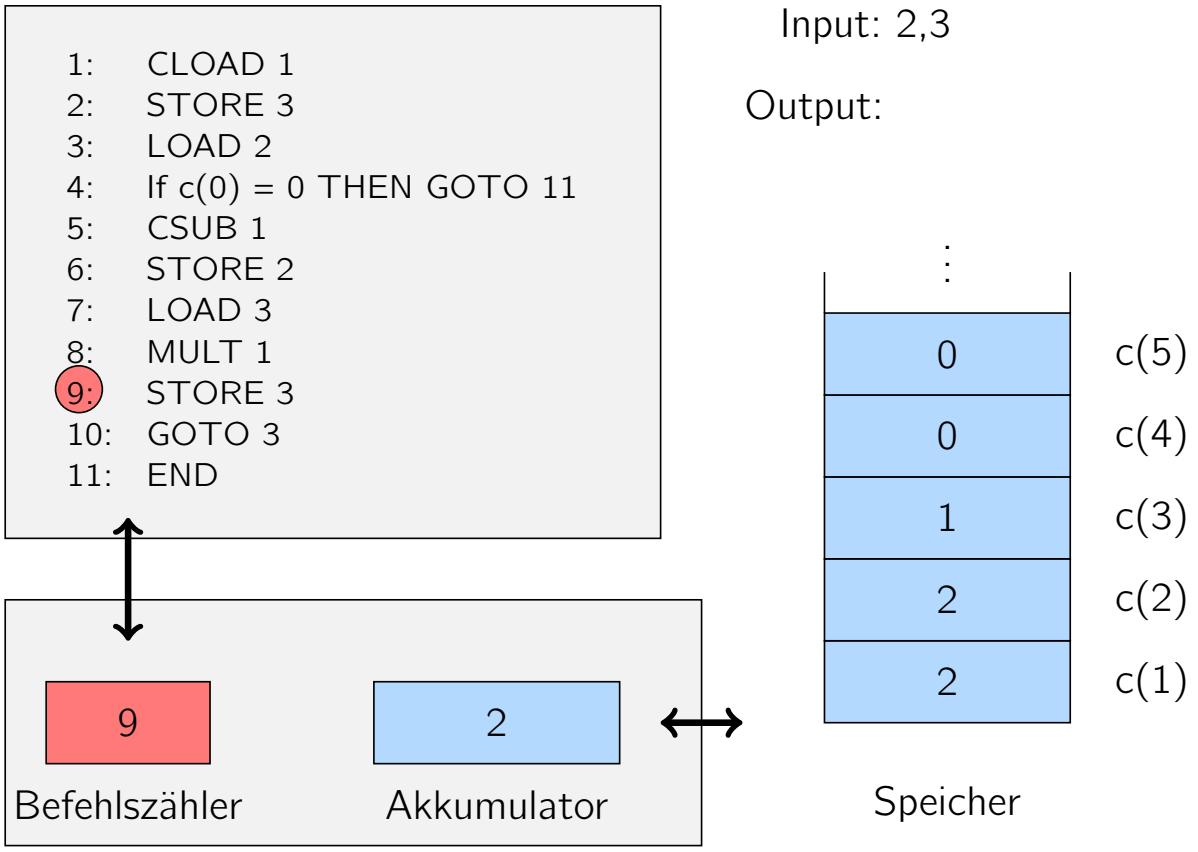
Beispielprogramm für die RAM



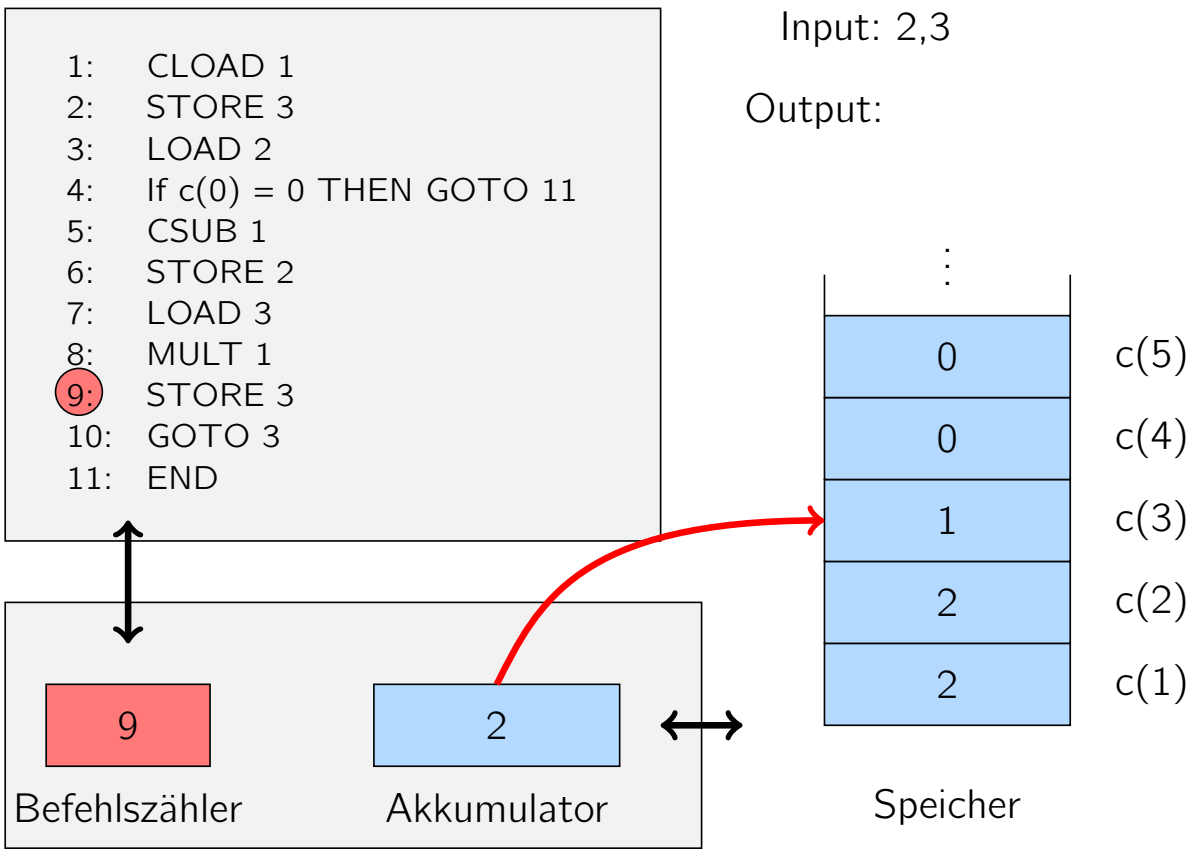
Beispielprogramm für die RAM



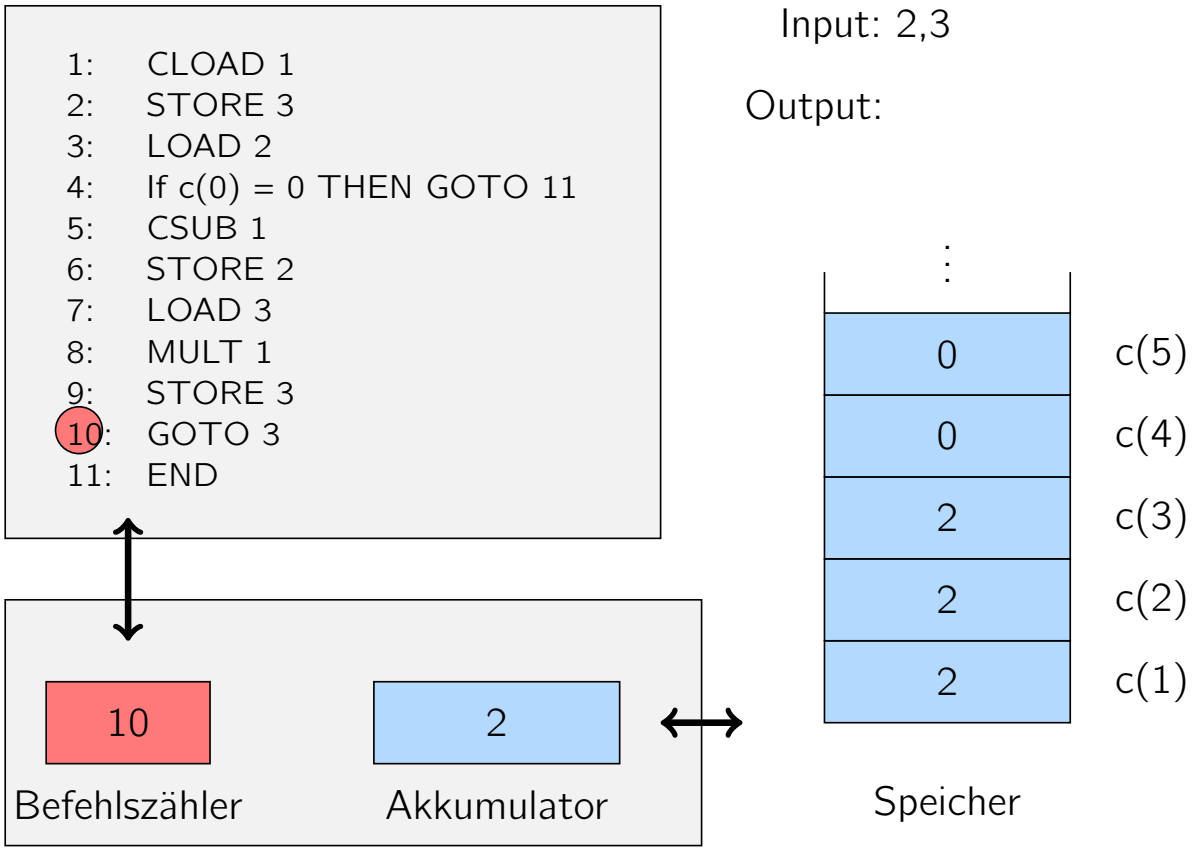
Beispielprogramm für die RAM



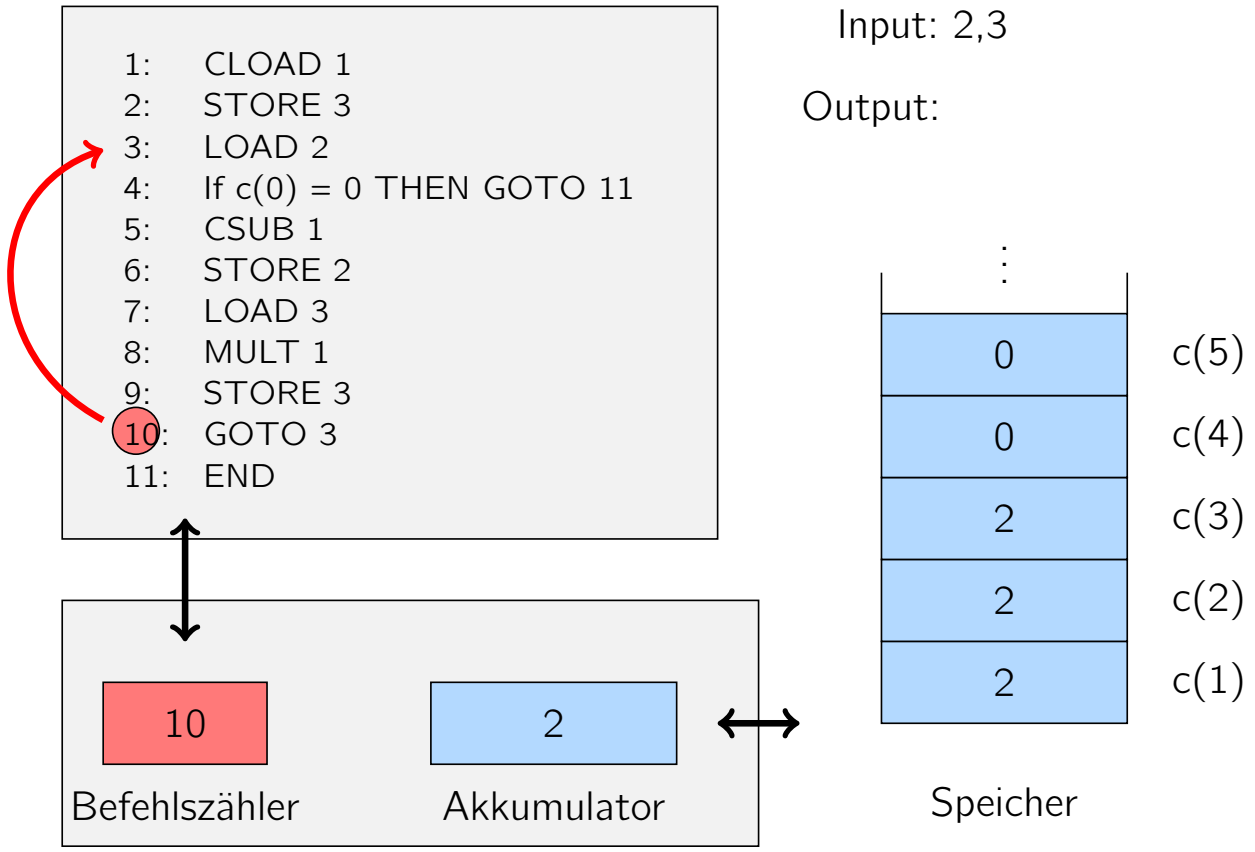
Beispielprogramm für die RAM



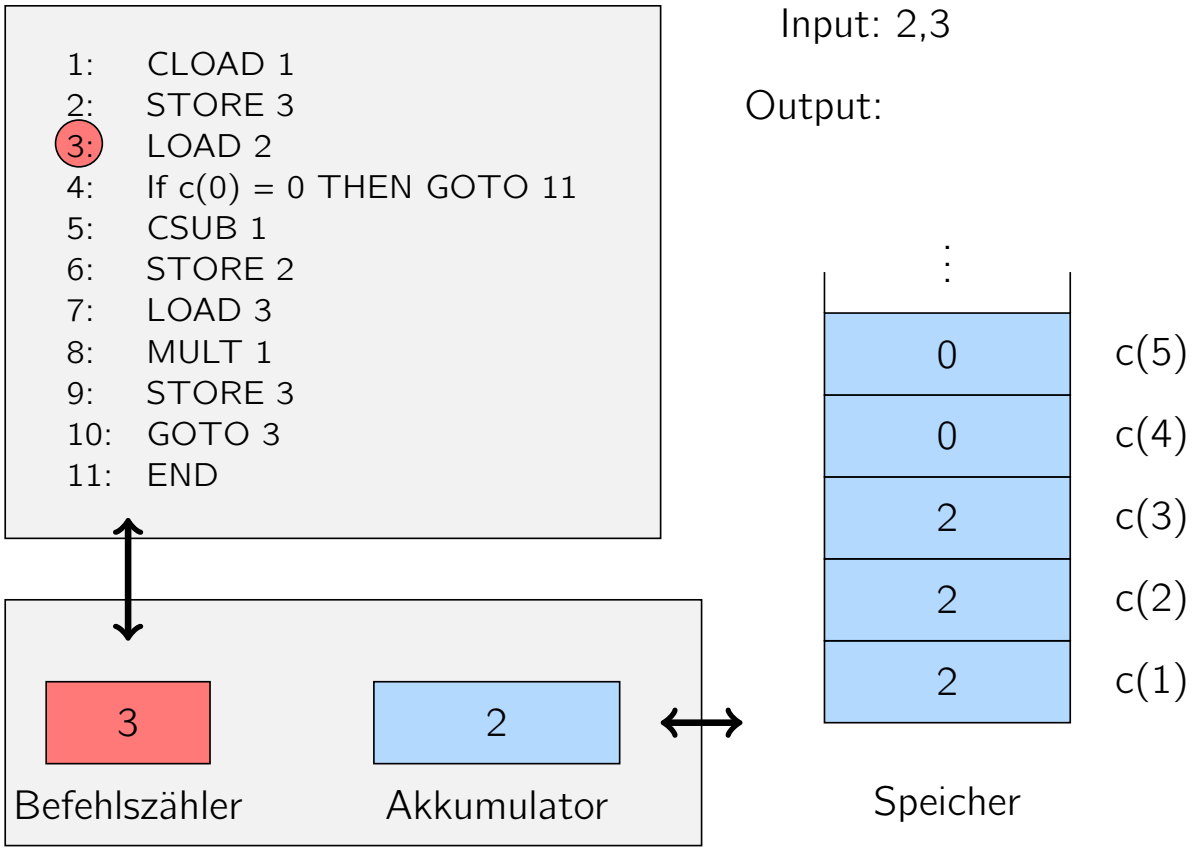
Beispielprogramm für die RAM



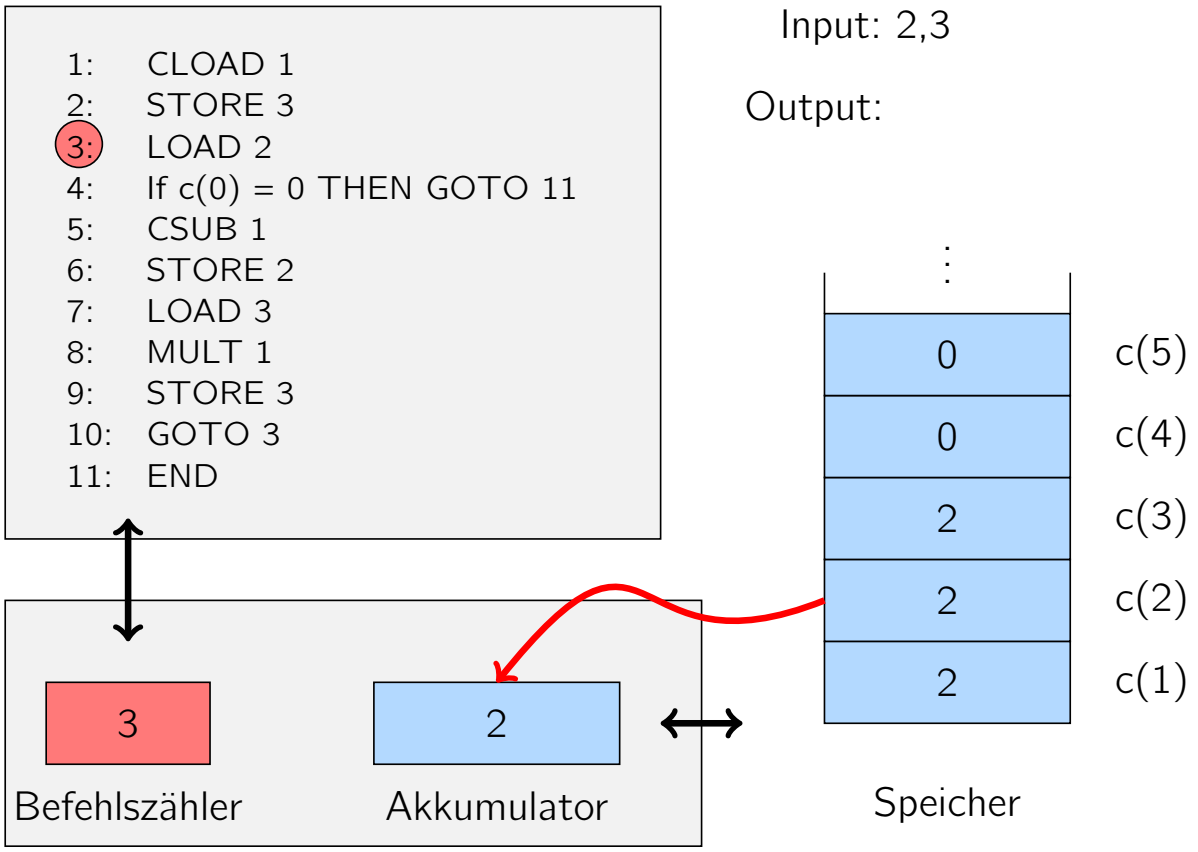
Beispielprogramm für die RAM



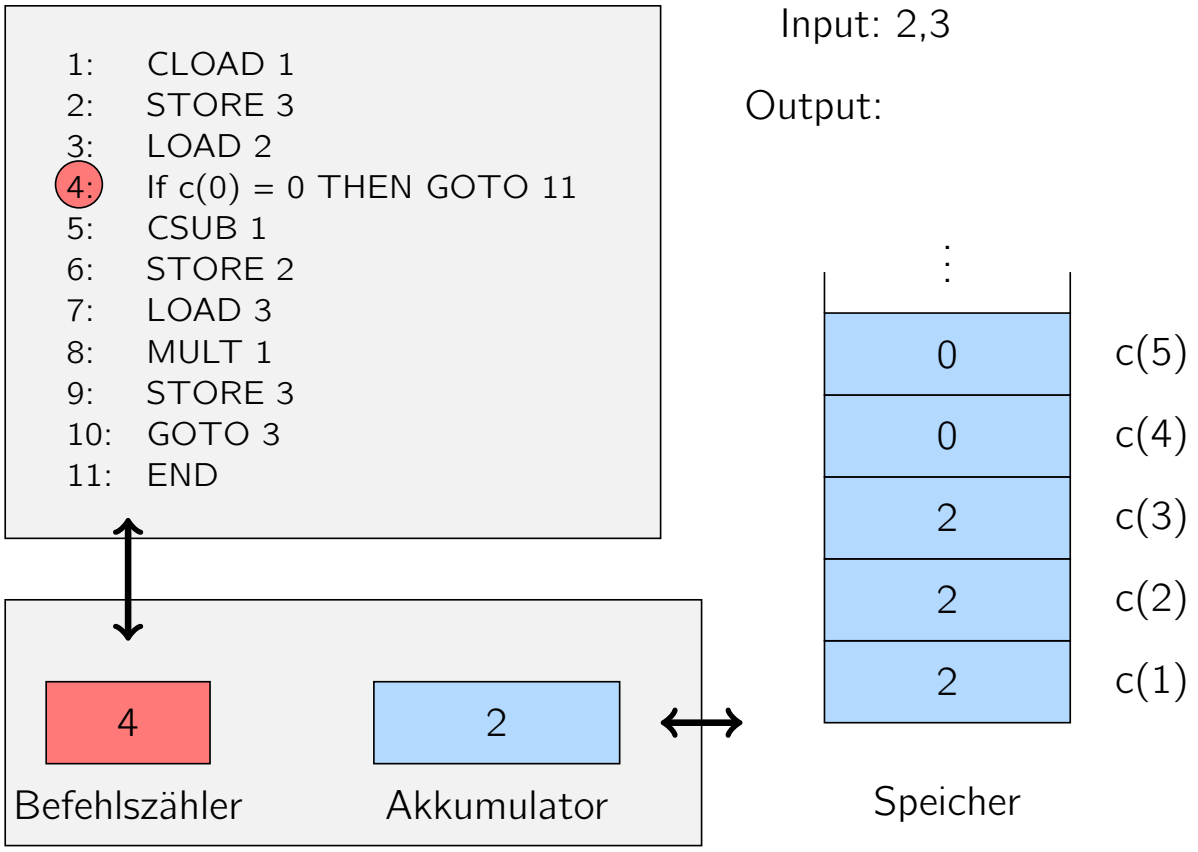
Beispielprogramm für die RAM



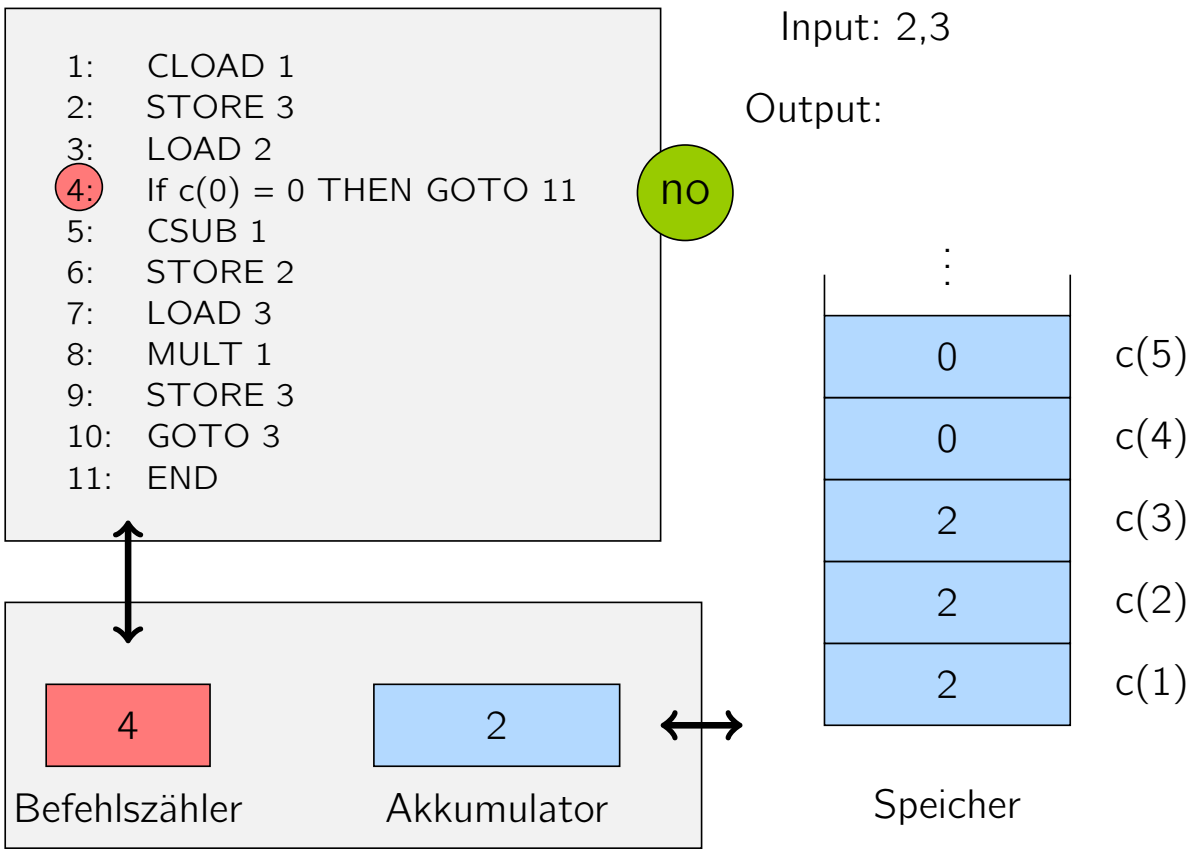
Beispielprogramm für die RAM



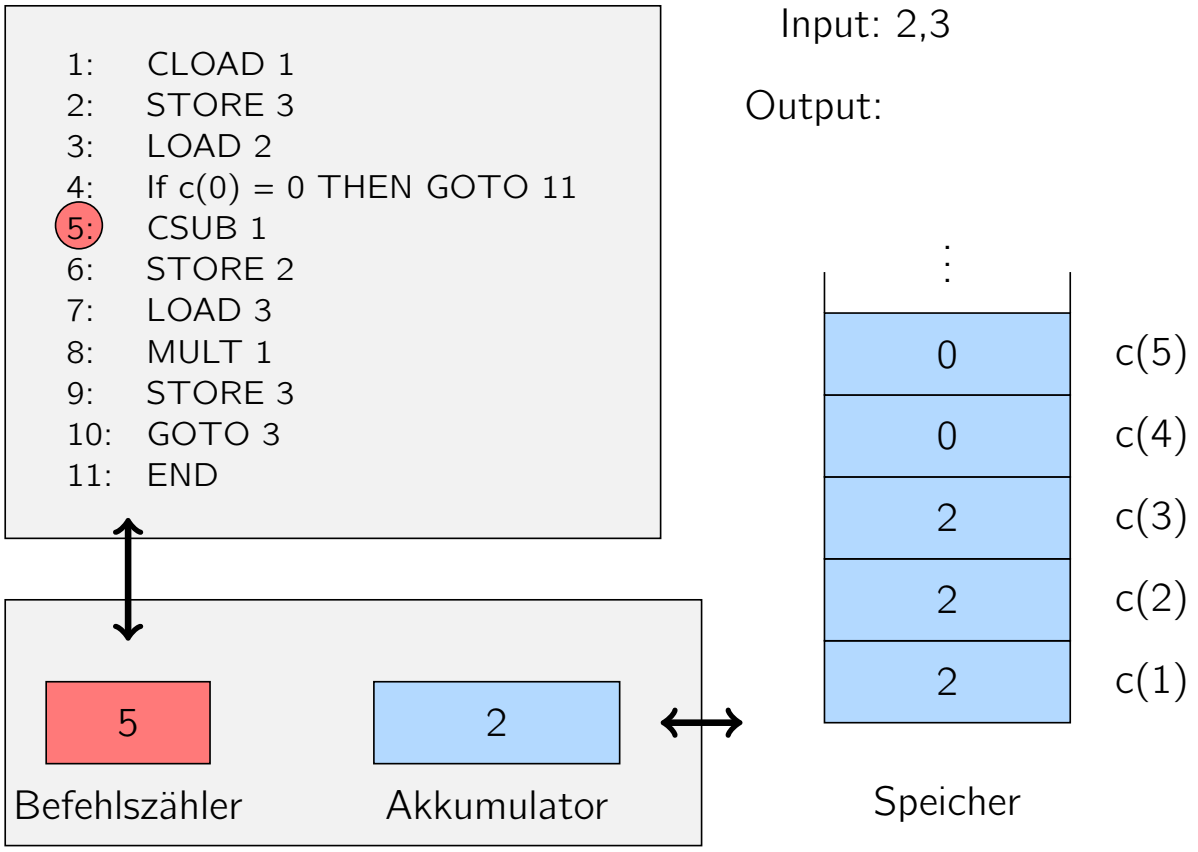
Beispielprogramm für die RAM



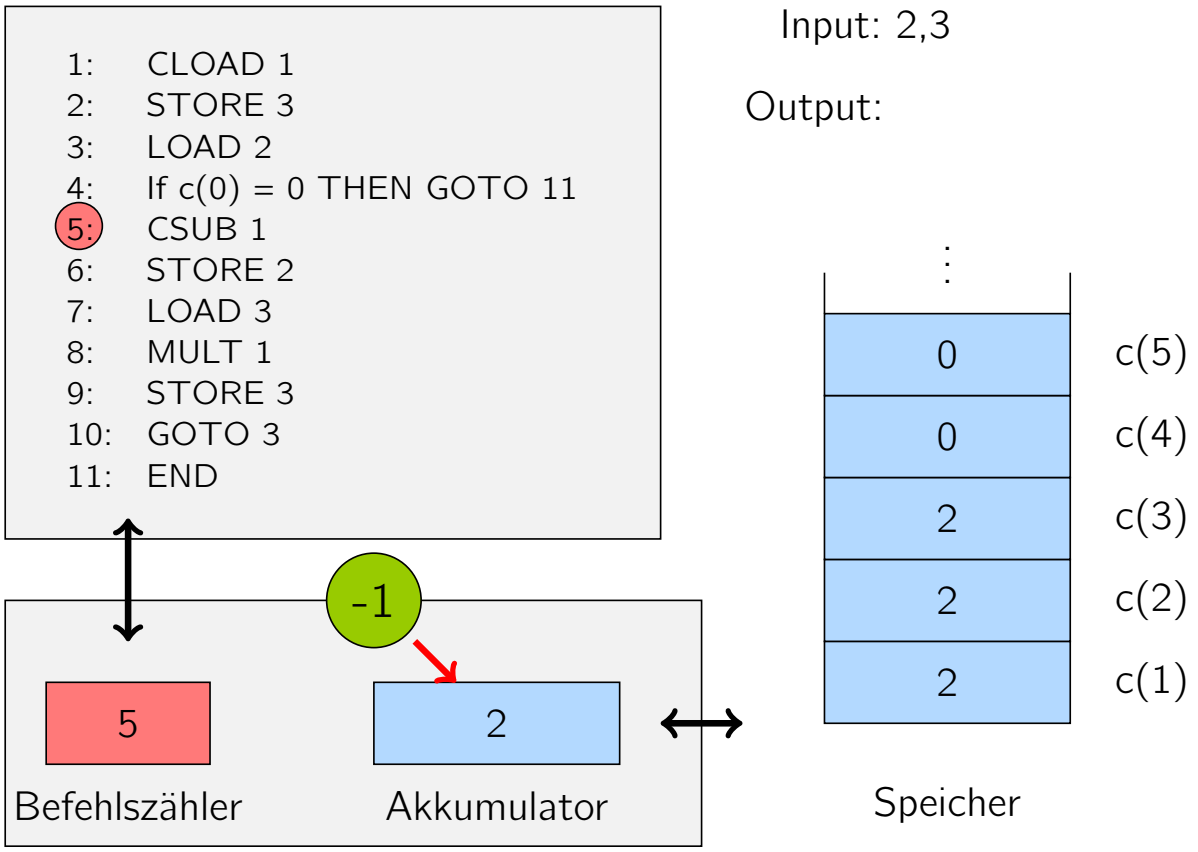
Beispielprogramm für die RAM



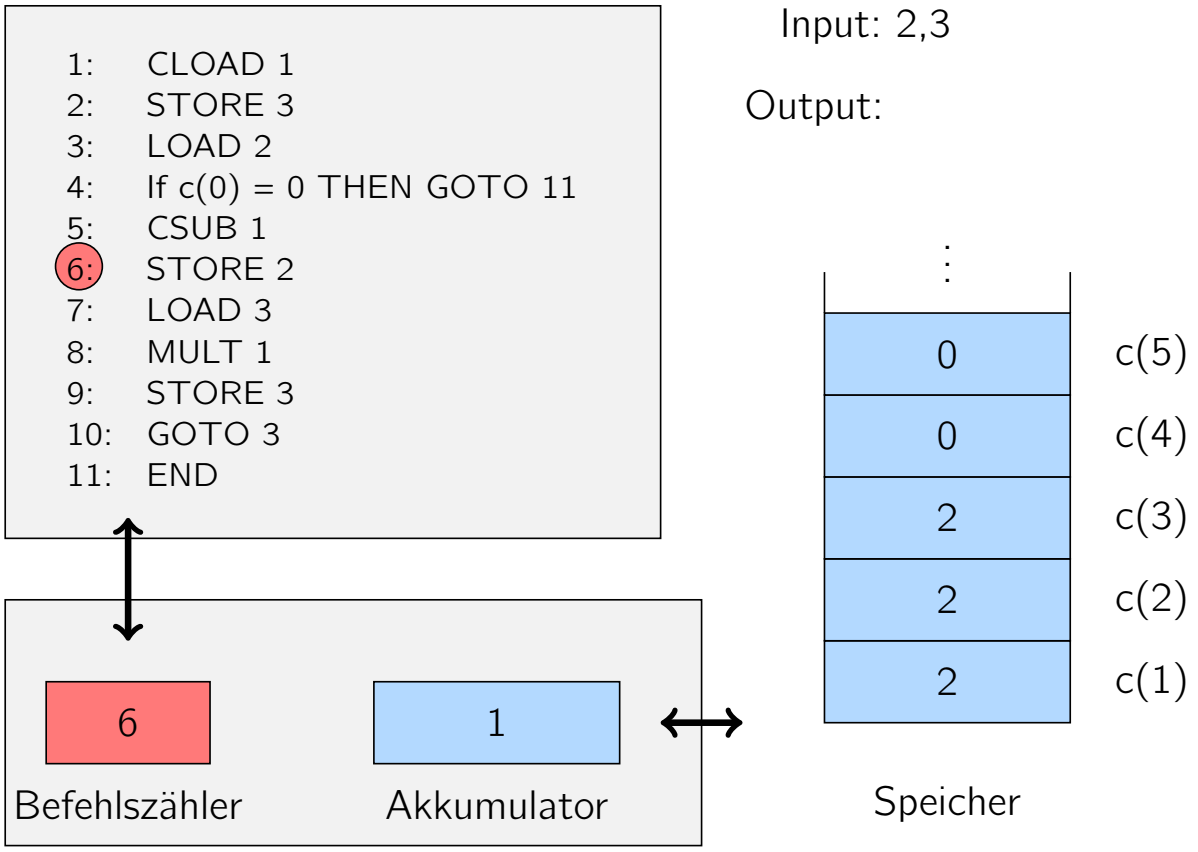
Beispielprogramm für die RAM



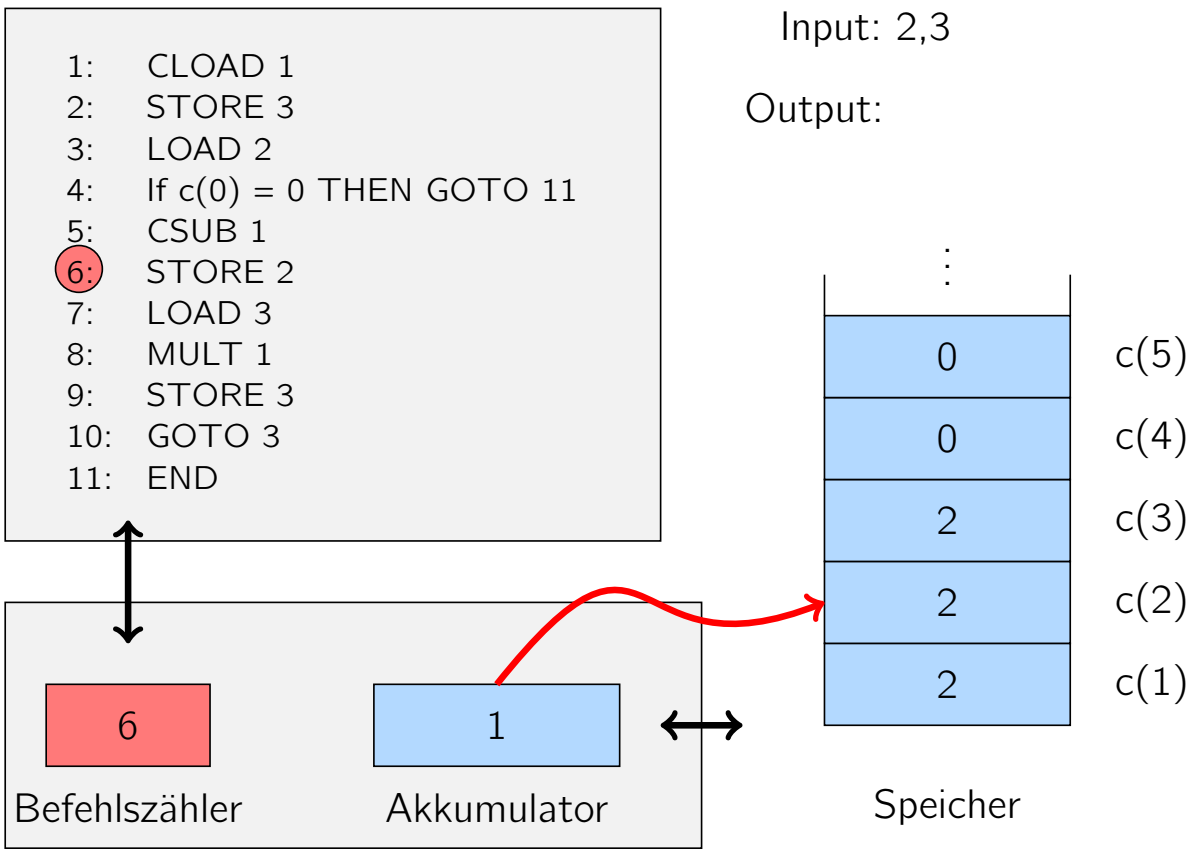
Beispielprogramm für die RAM



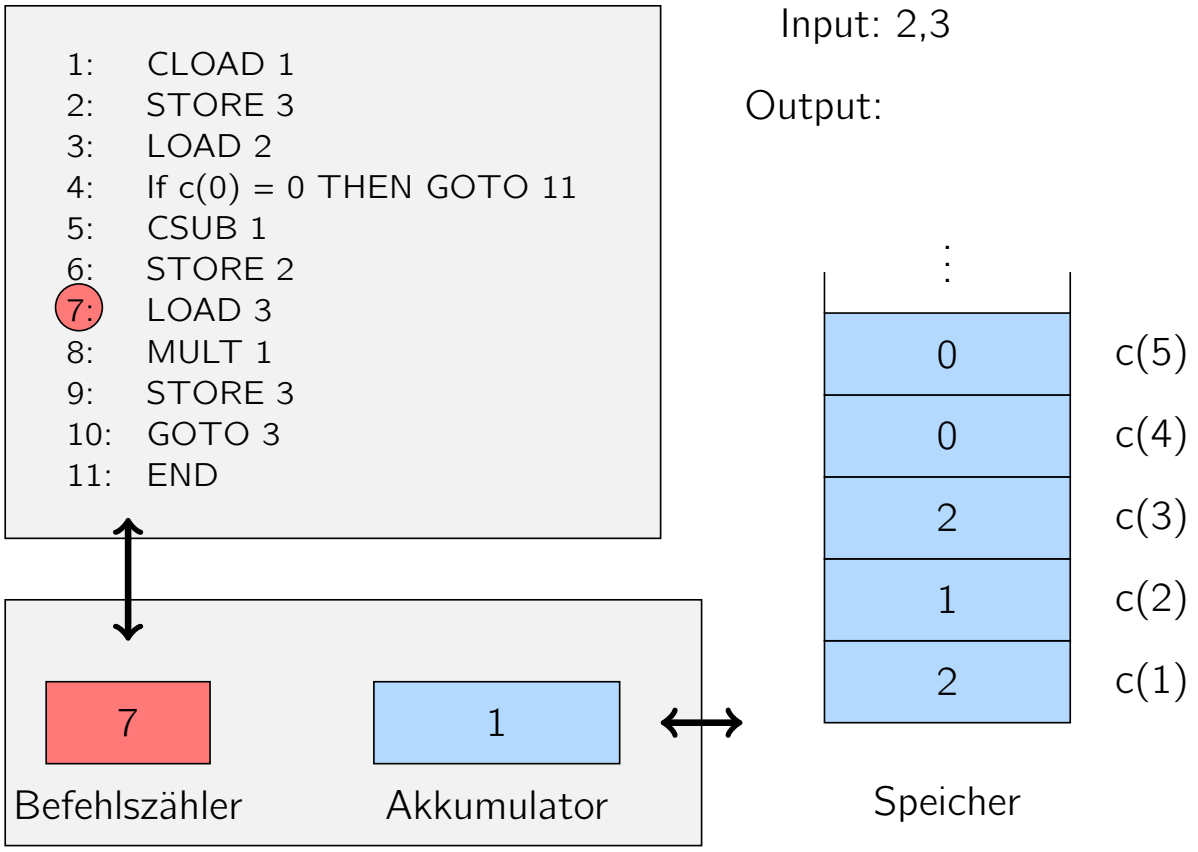
Beispielprogramm für die RAM



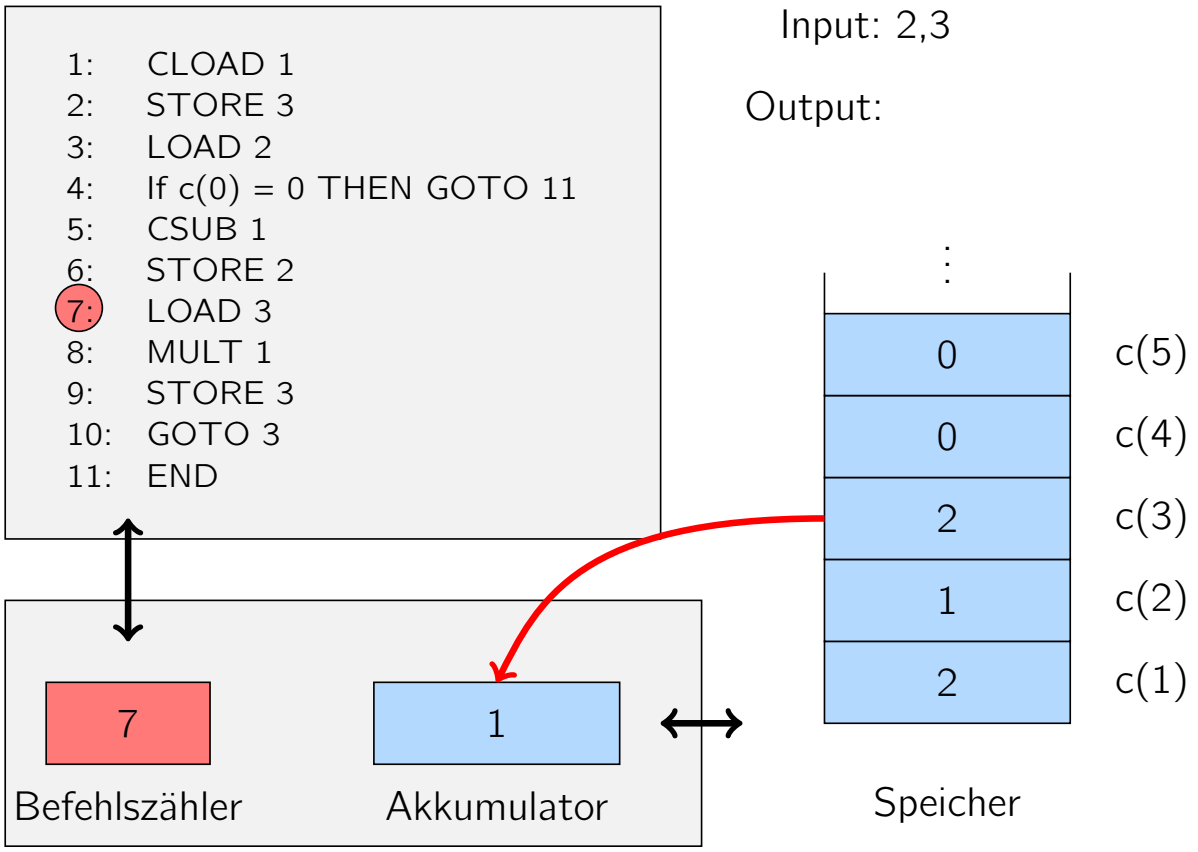
Beispielprogramm für die RAM



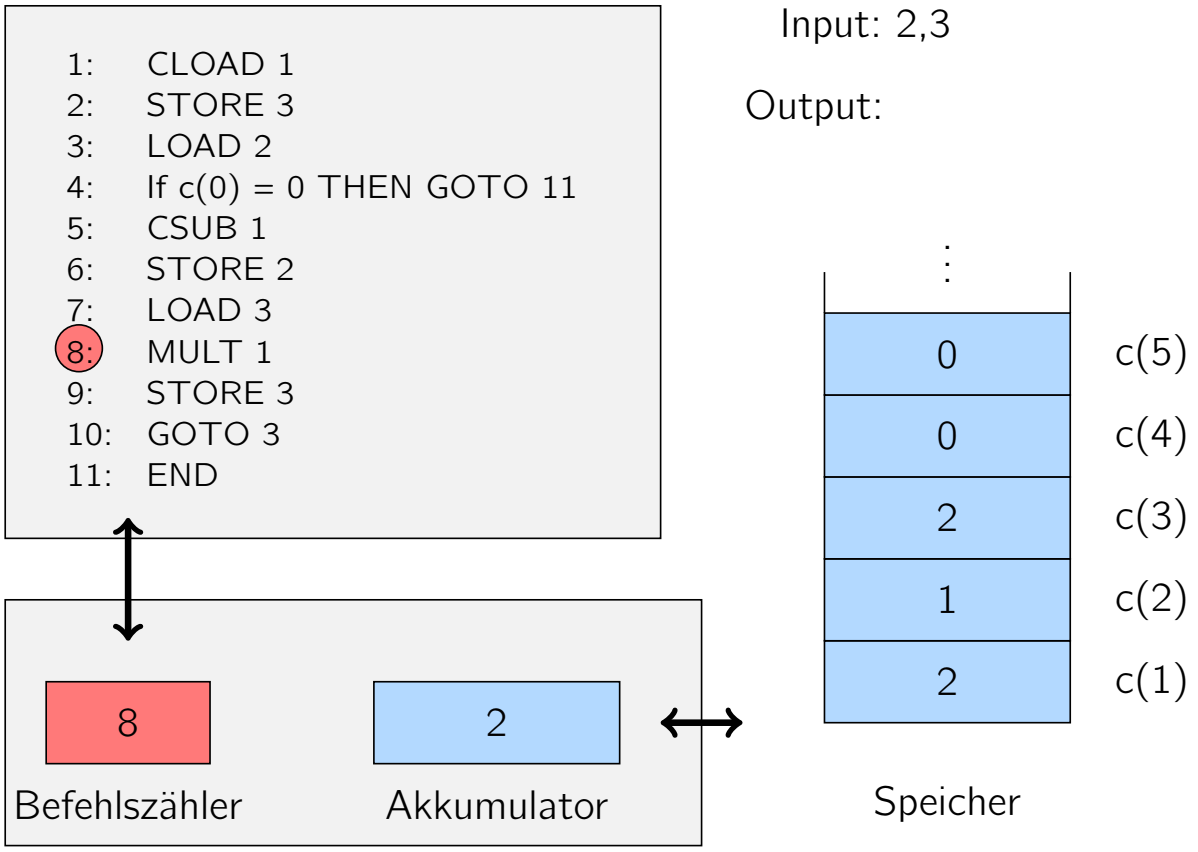
Beispielprogramm für die RAM



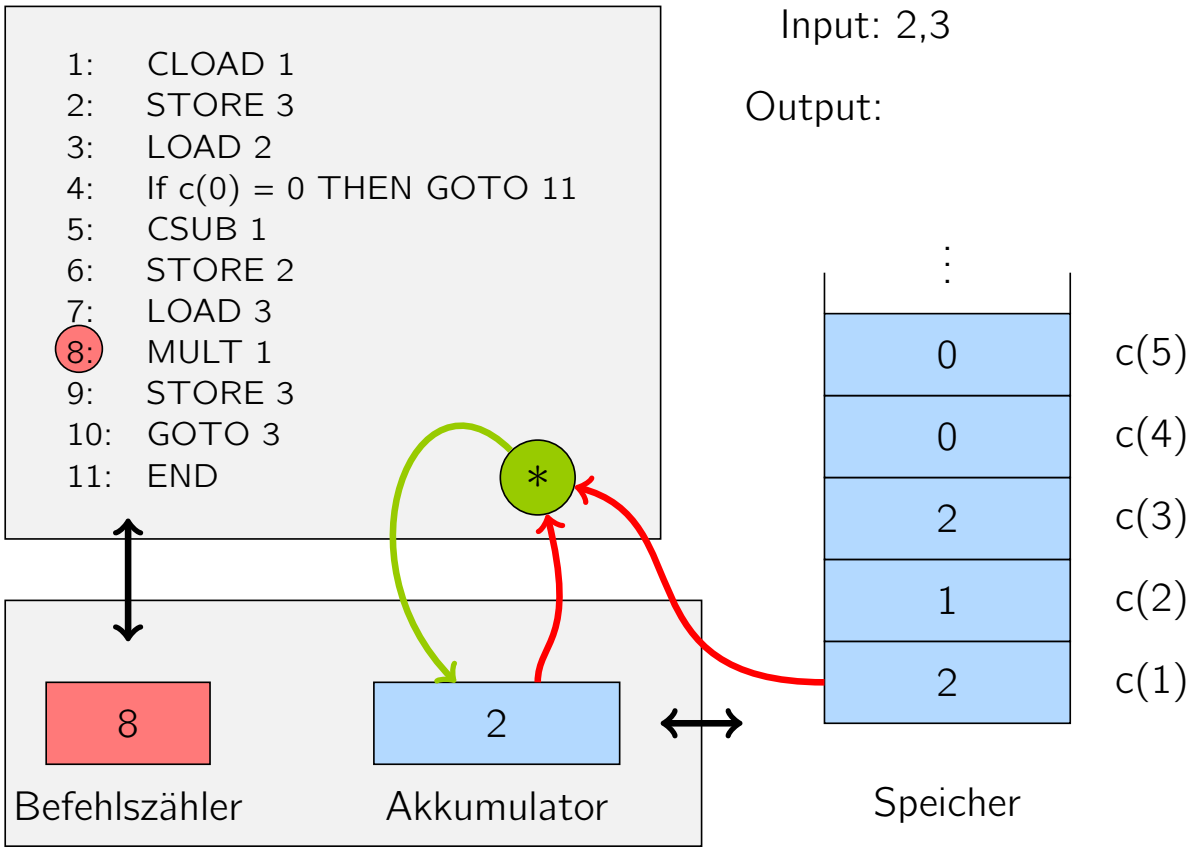
Beispielprogramm für die RAM



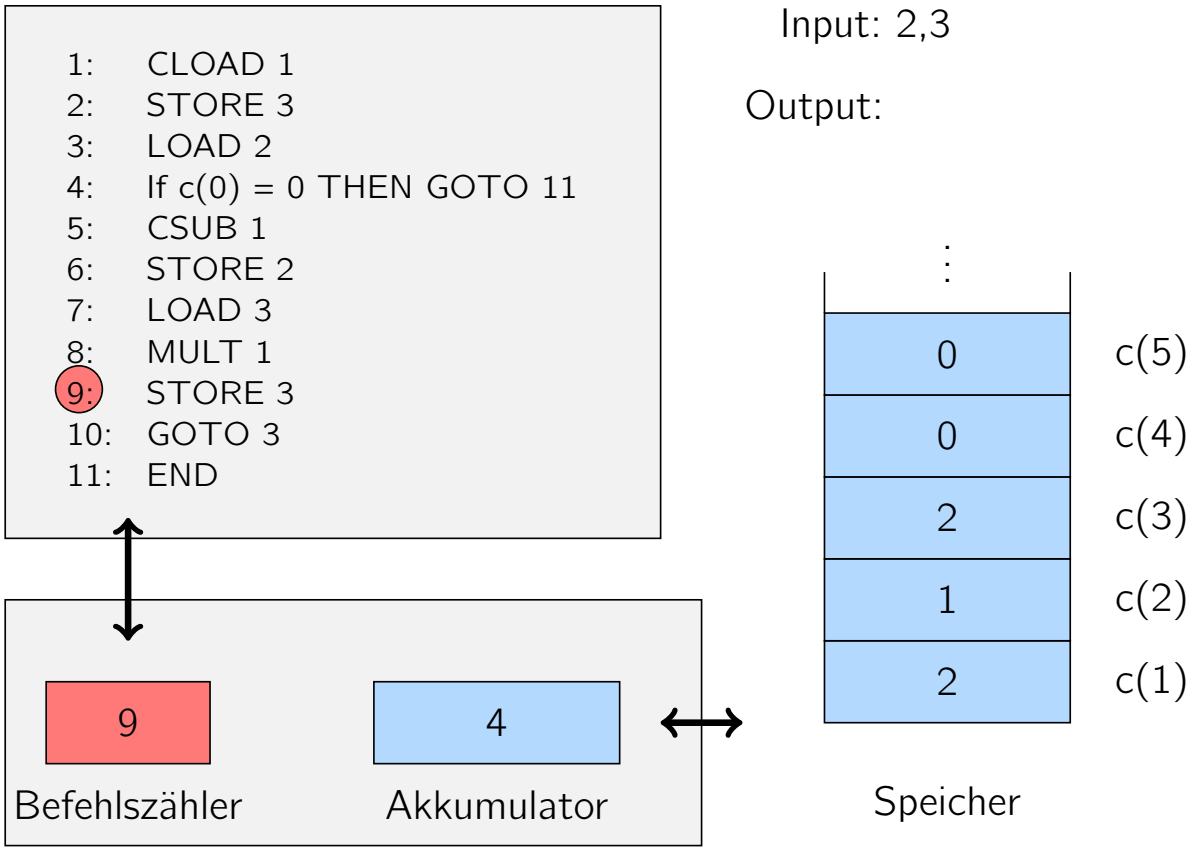
Beispielprogramm für die RAM



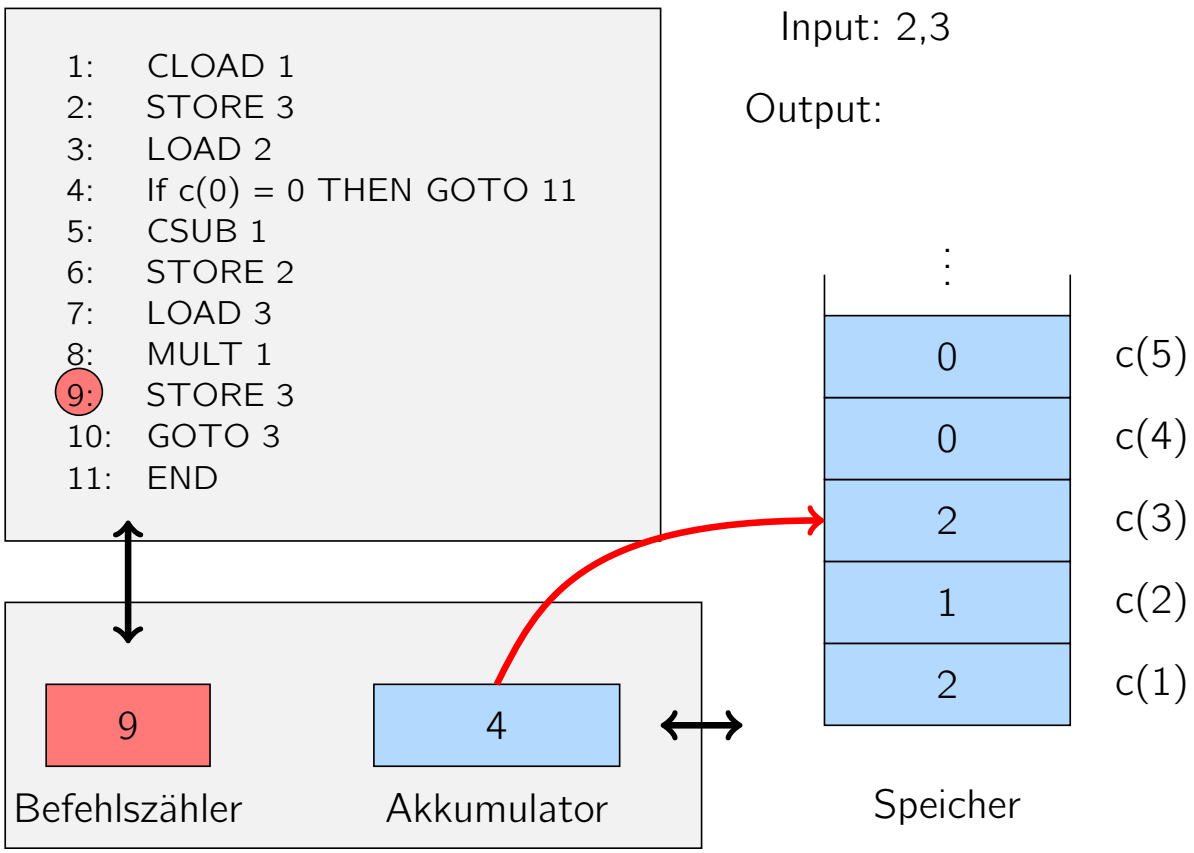
Beispielprogramm für die RAM



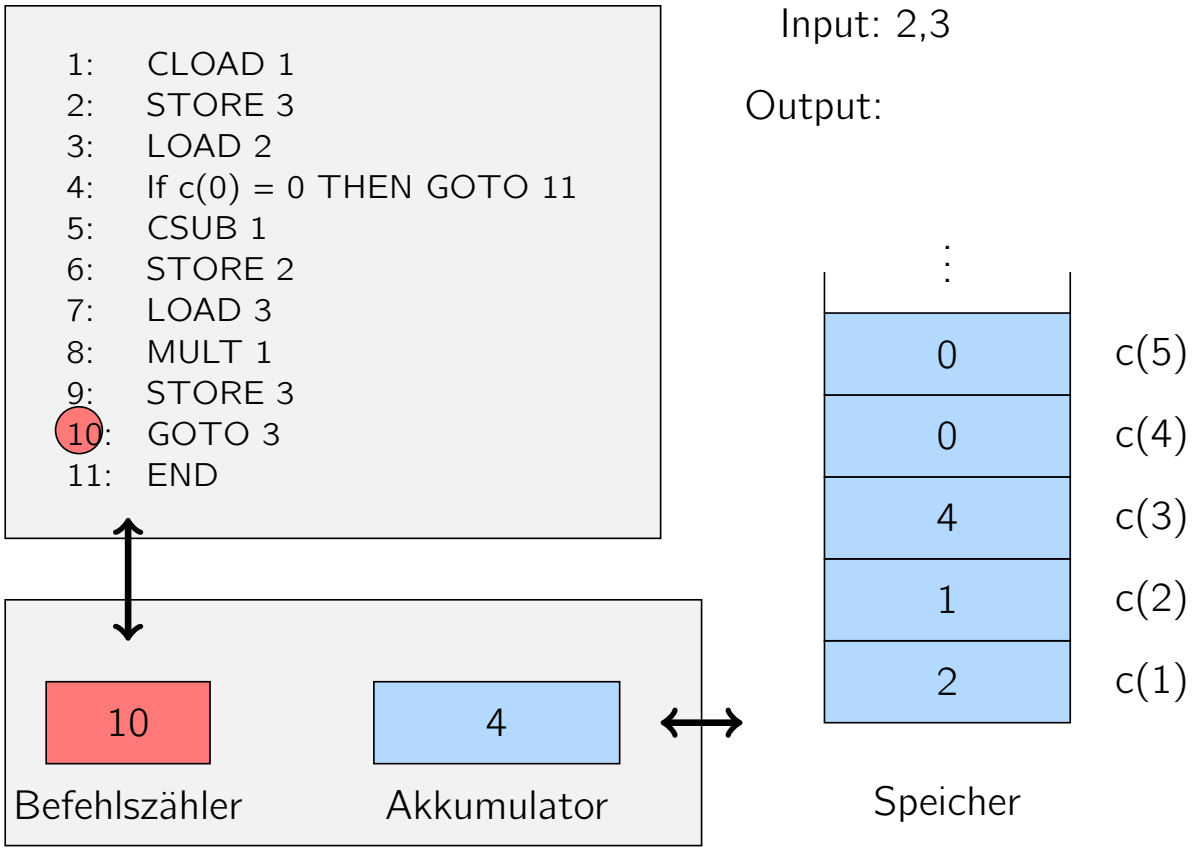
Beispielprogramm für die RAM



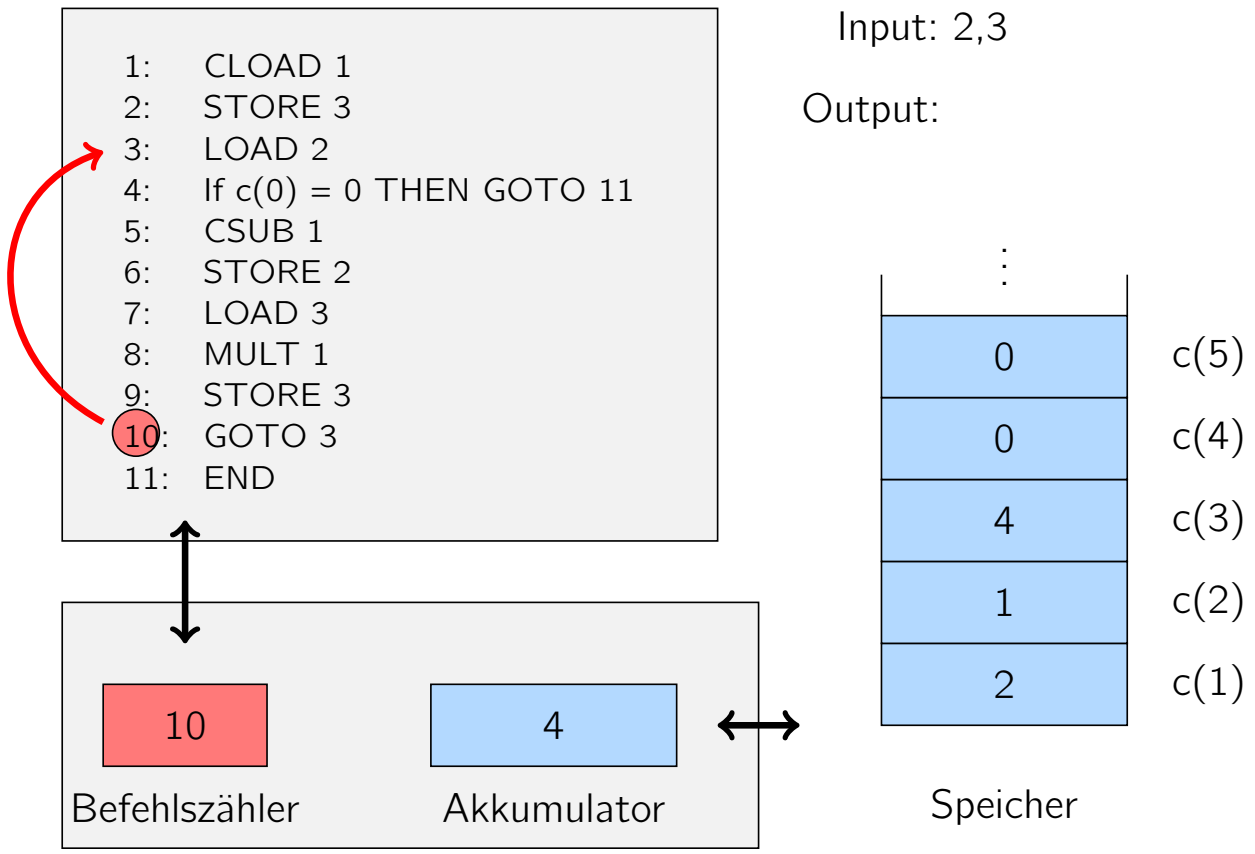
Beispielprogramm für die RAM



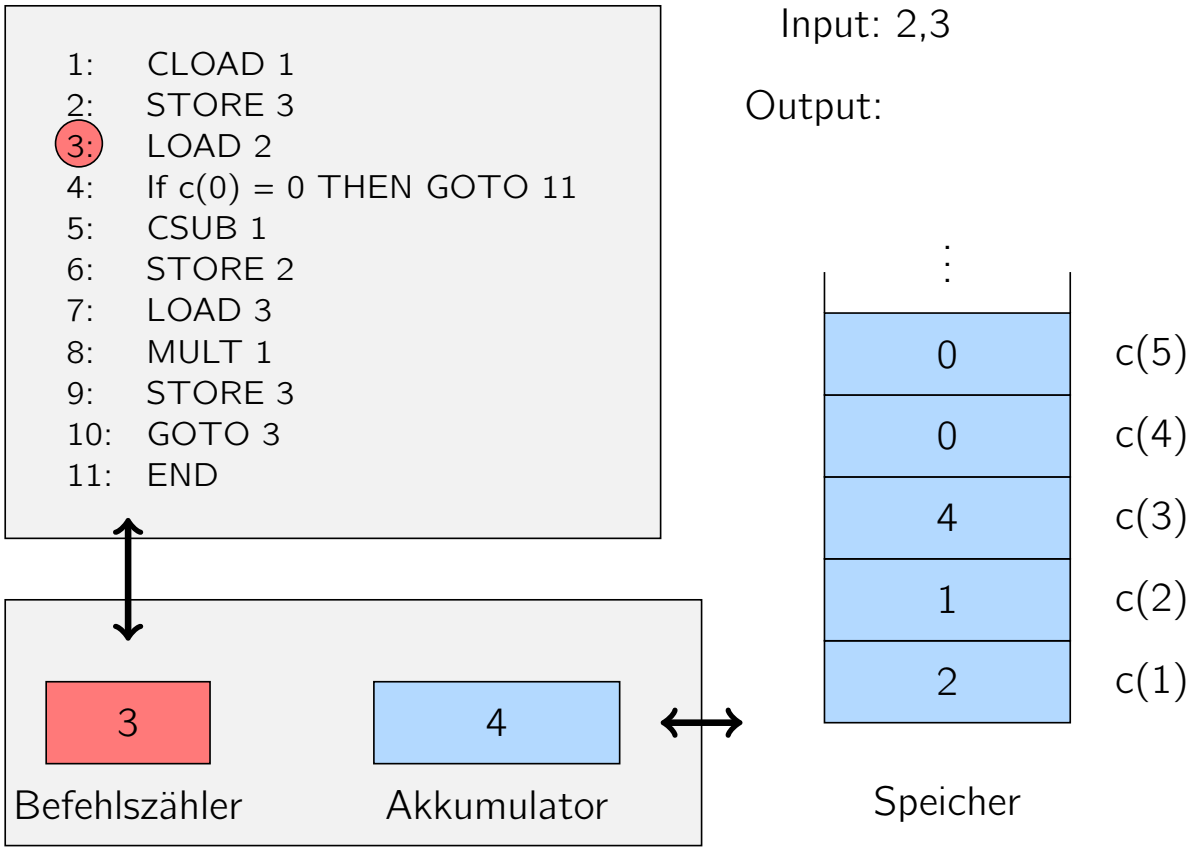
Beispielprogramm für die RAM



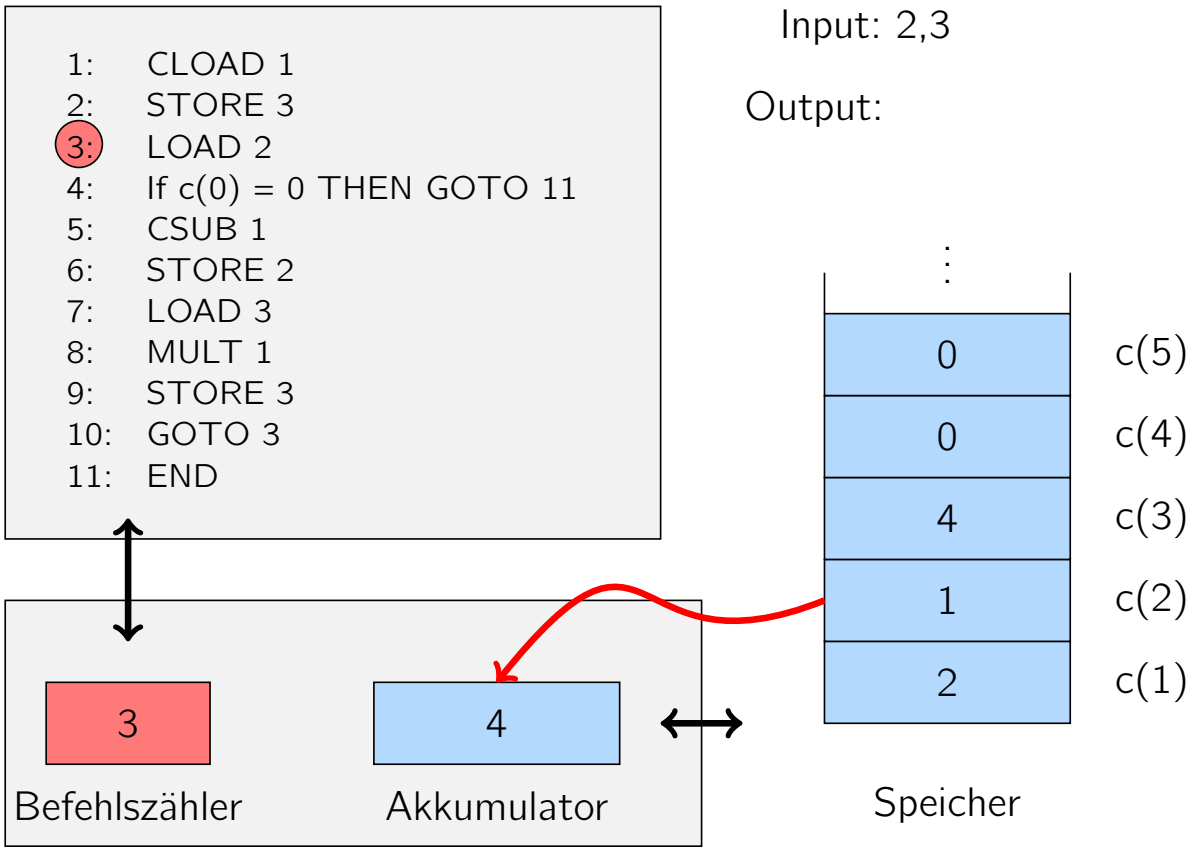
Beispielprogramm für die RAM



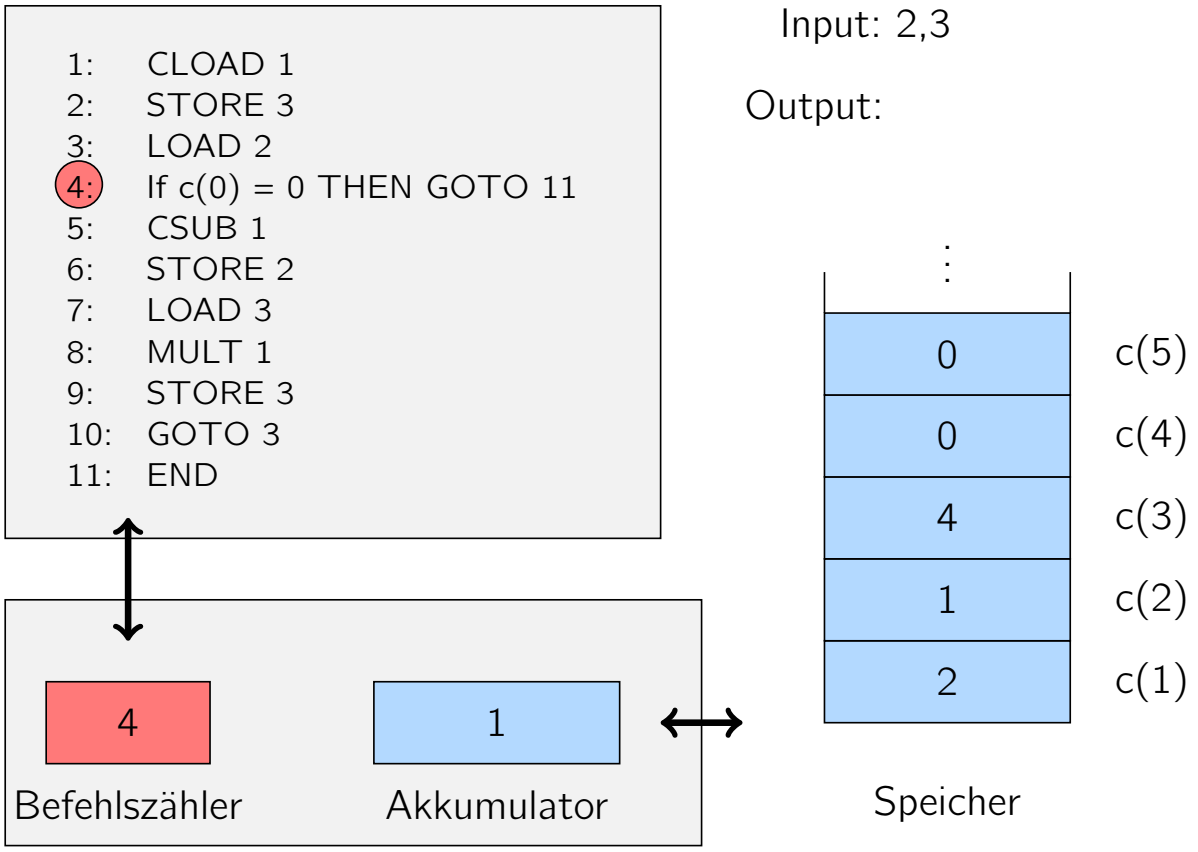
Beispielprogramm für die RAM



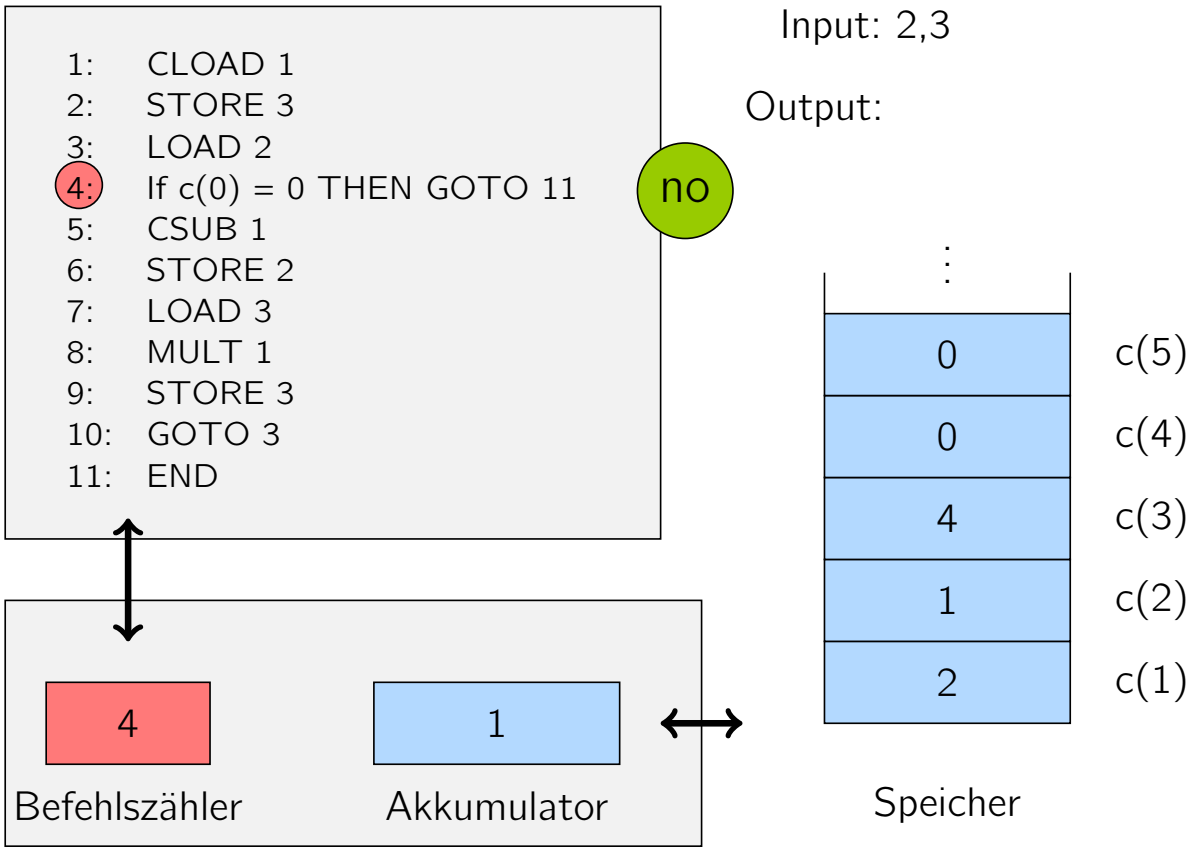
Beispielprogramm für die RAM



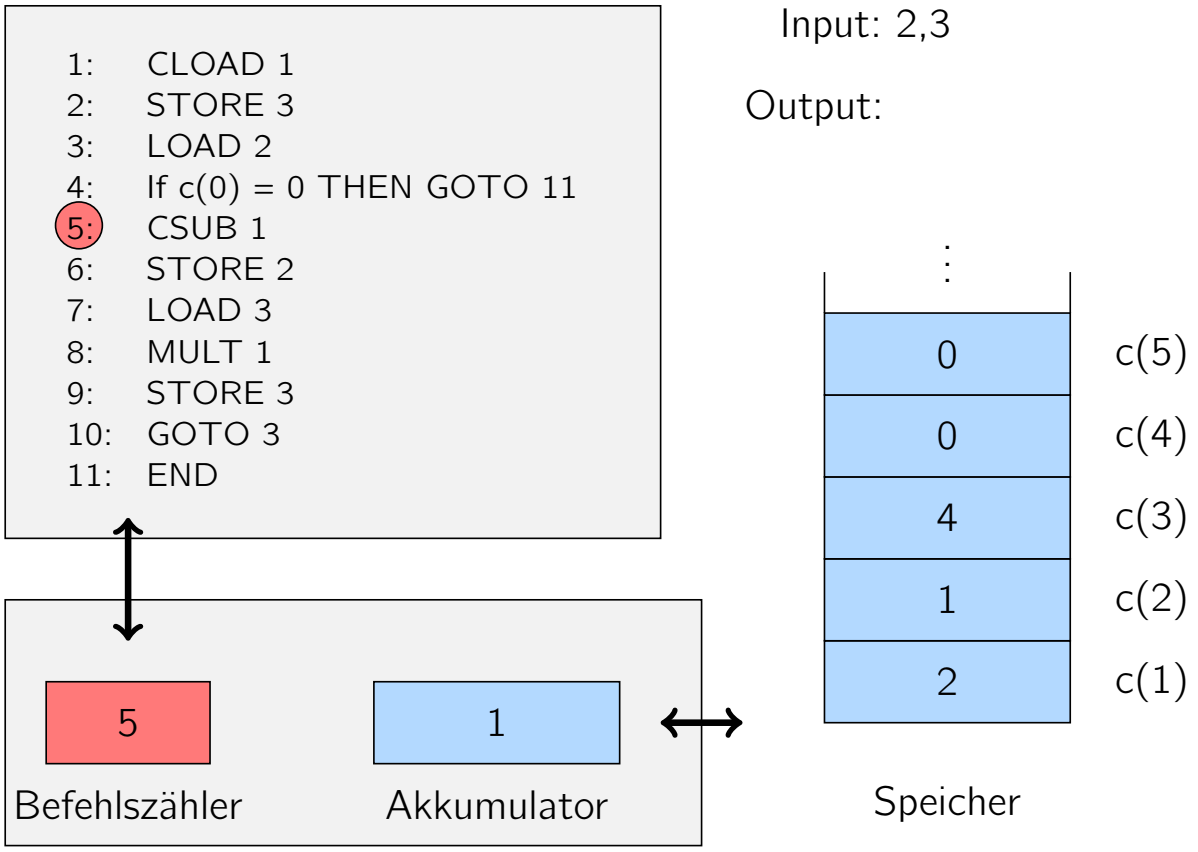
Beispielprogramm für die RAM



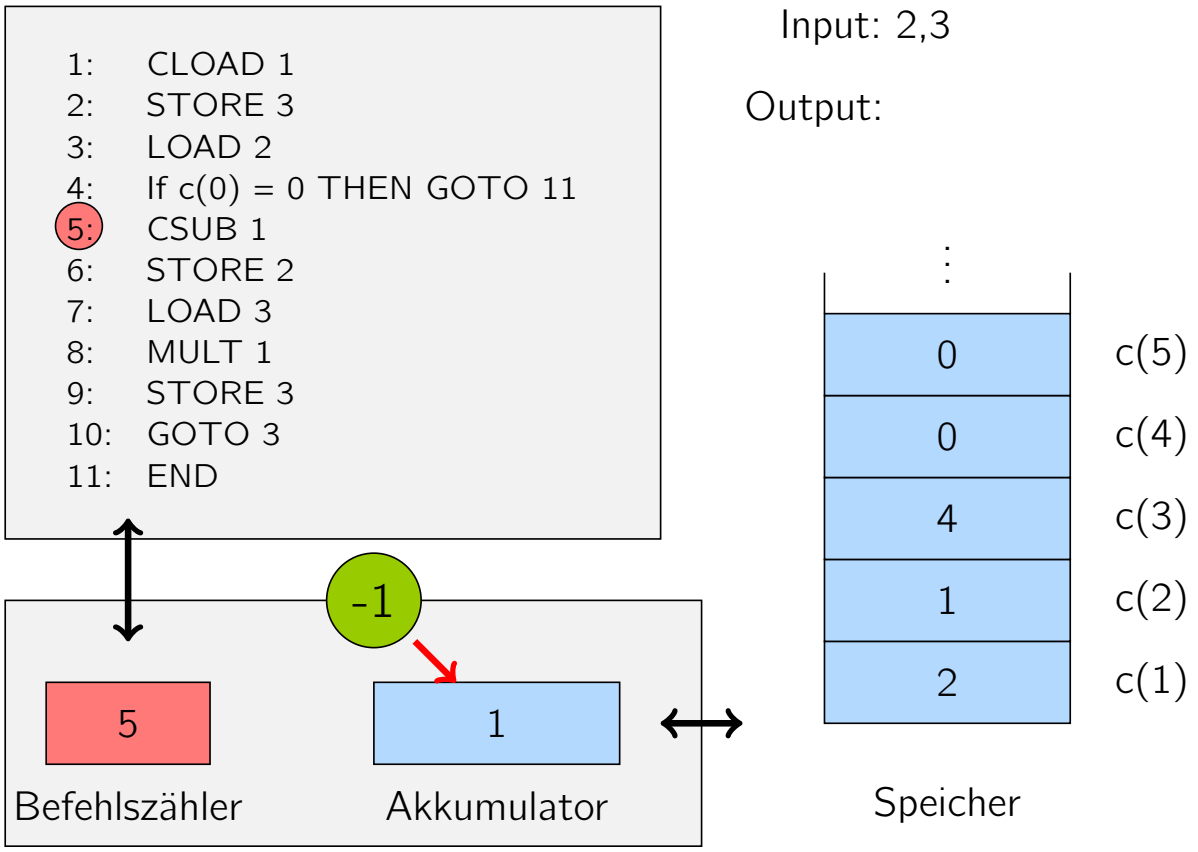
Beispielprogramm für die RAM



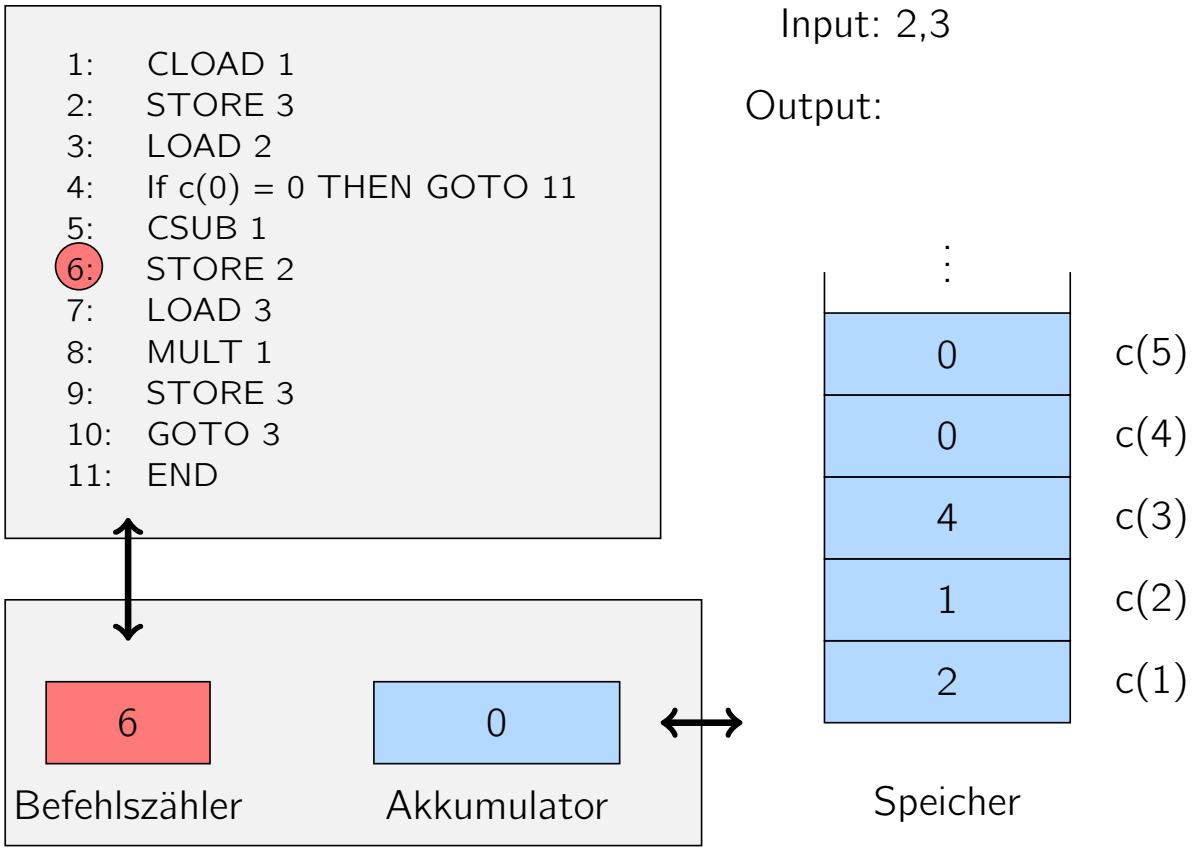
Beispielprogramm für die RAM



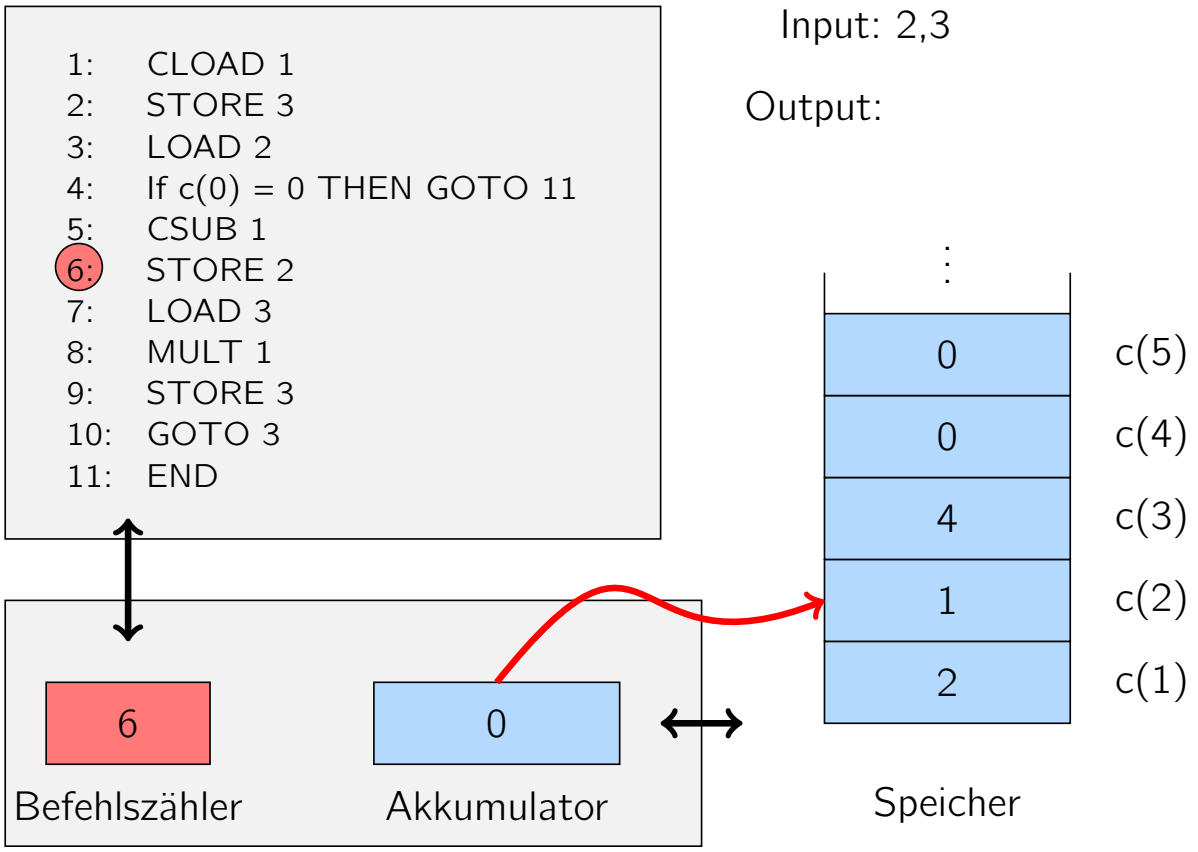
Beispielprogramm für die RAM



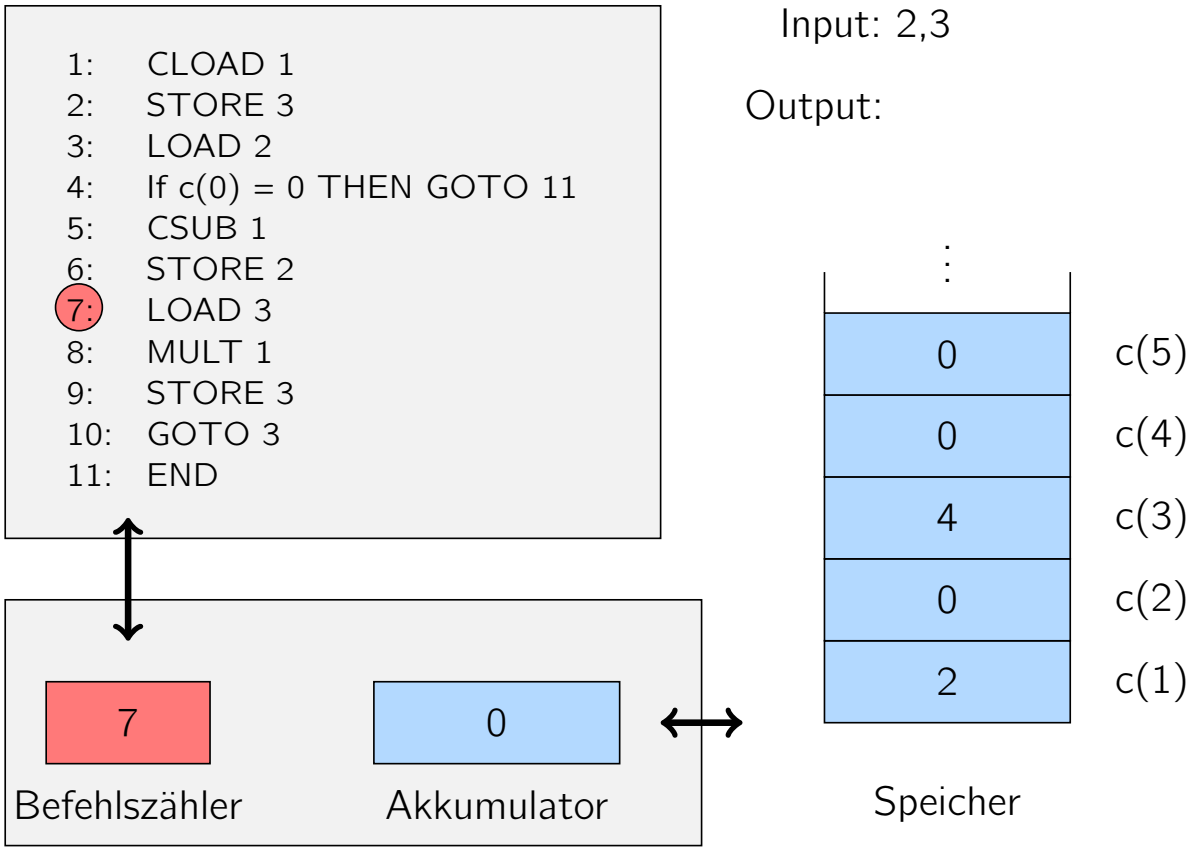
Beispielprogramm für die RAM



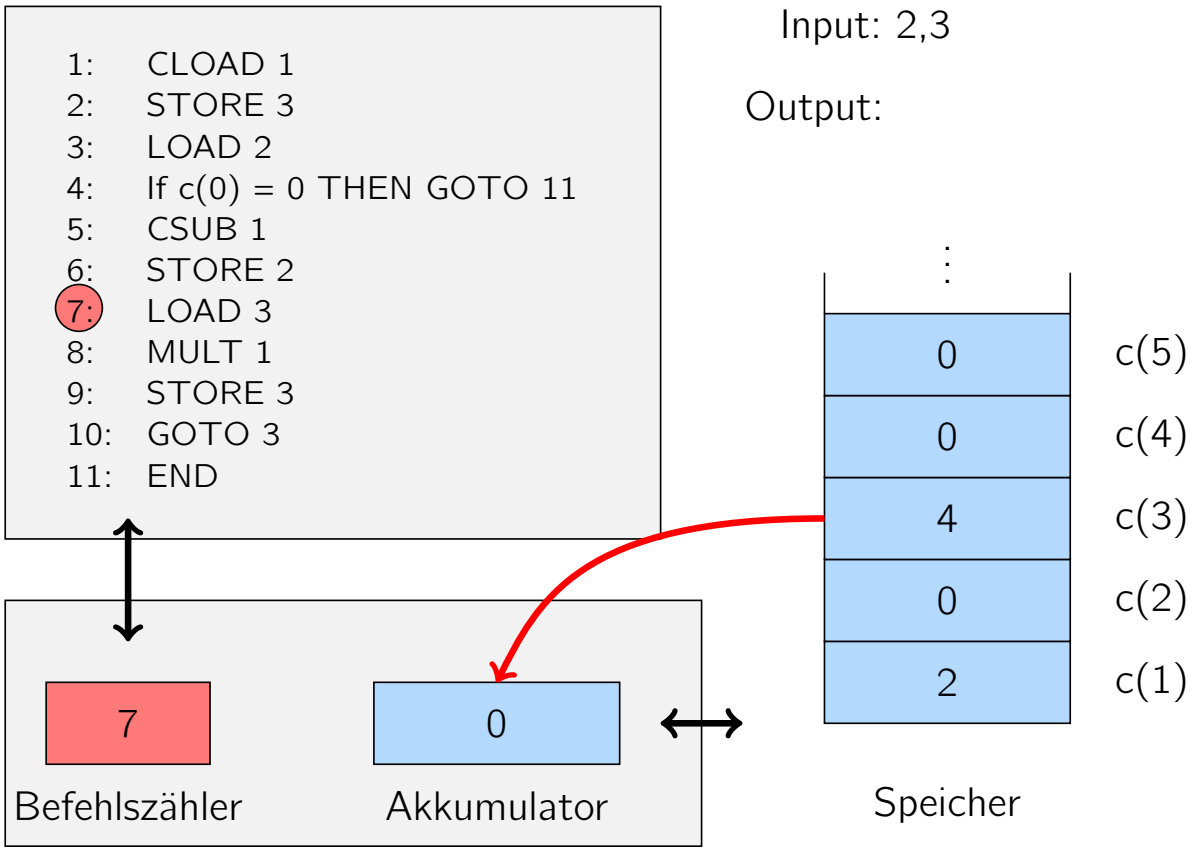
Beispielprogramm für die RAM



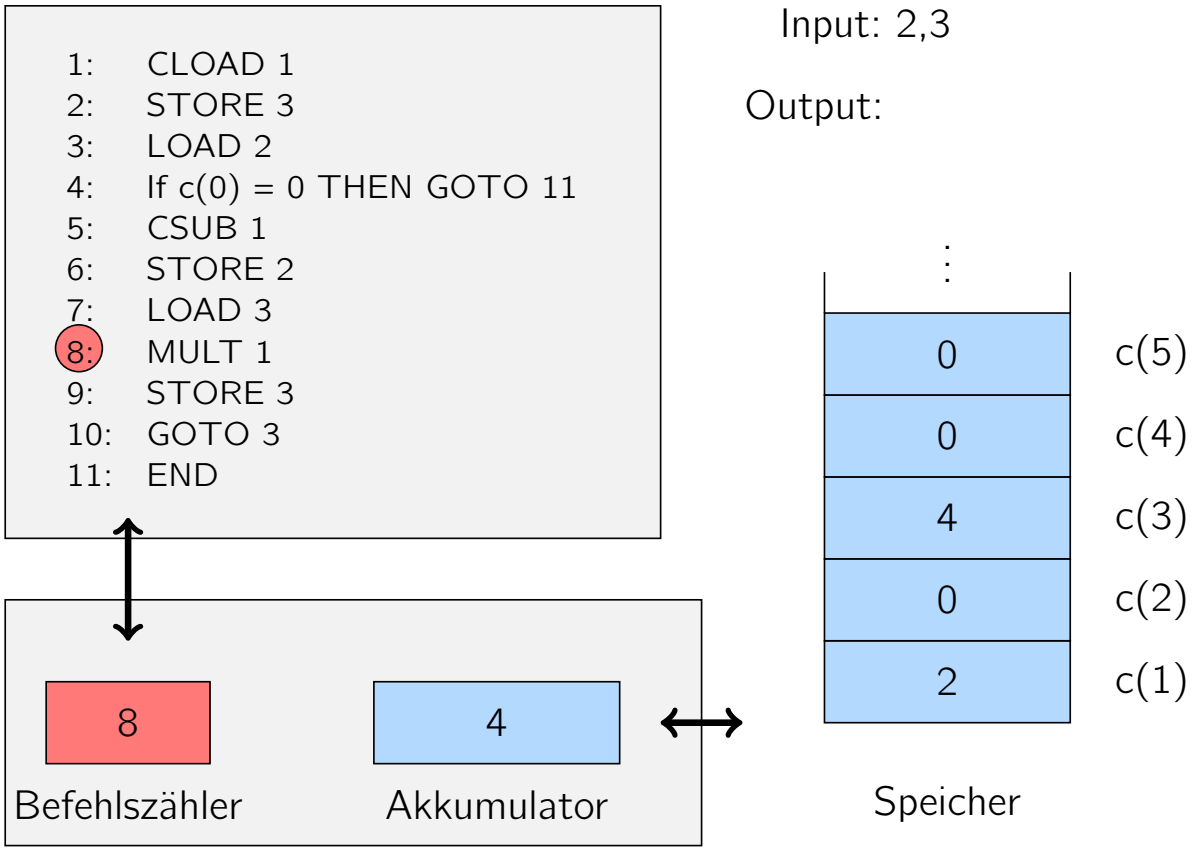
Beispielprogramm für die RAM



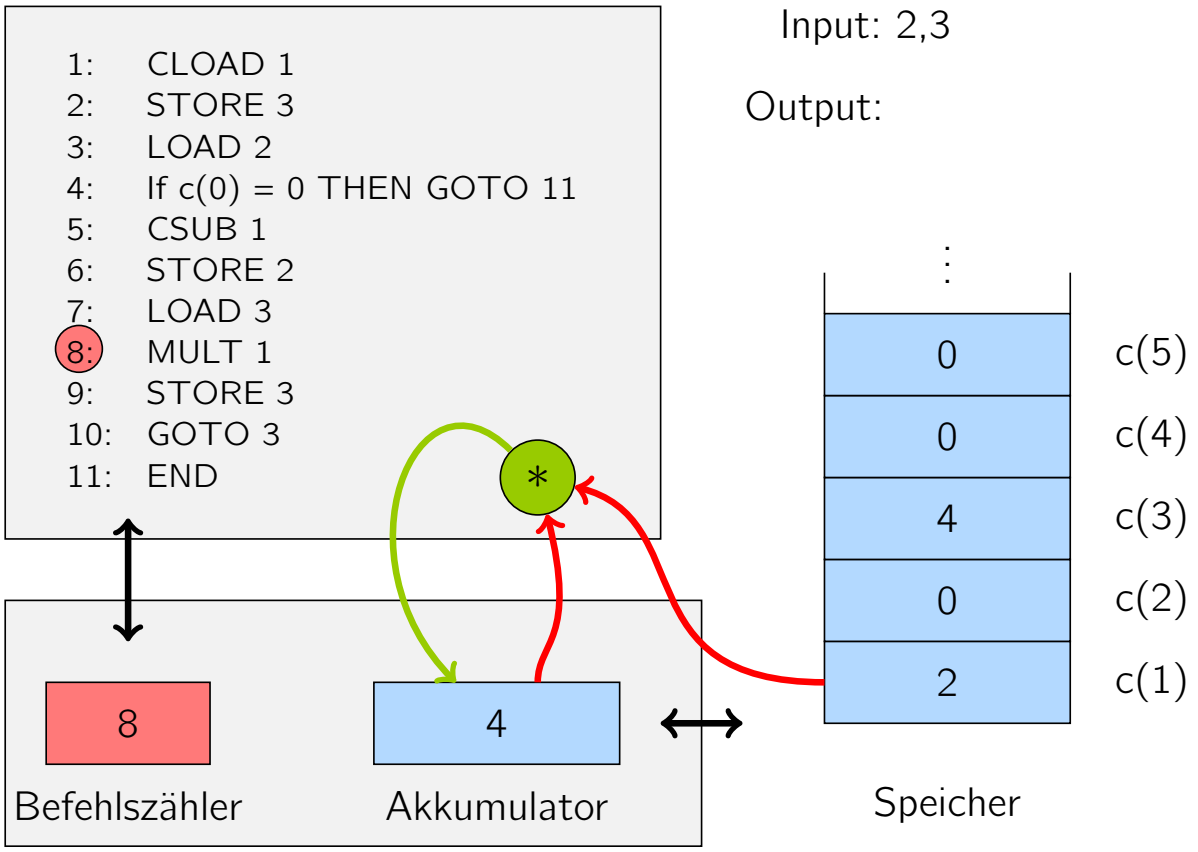
Beispielprogramm für die RAM



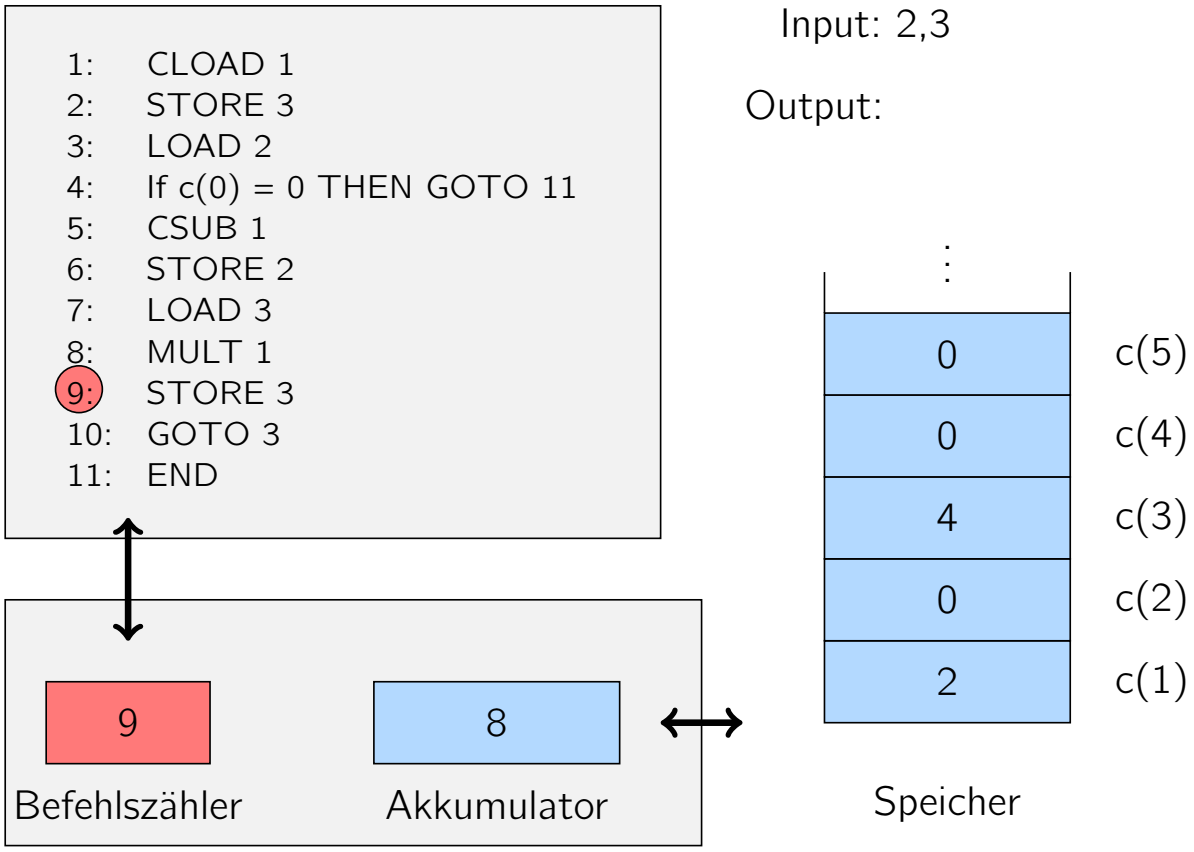
Beispielprogramm für die RAM



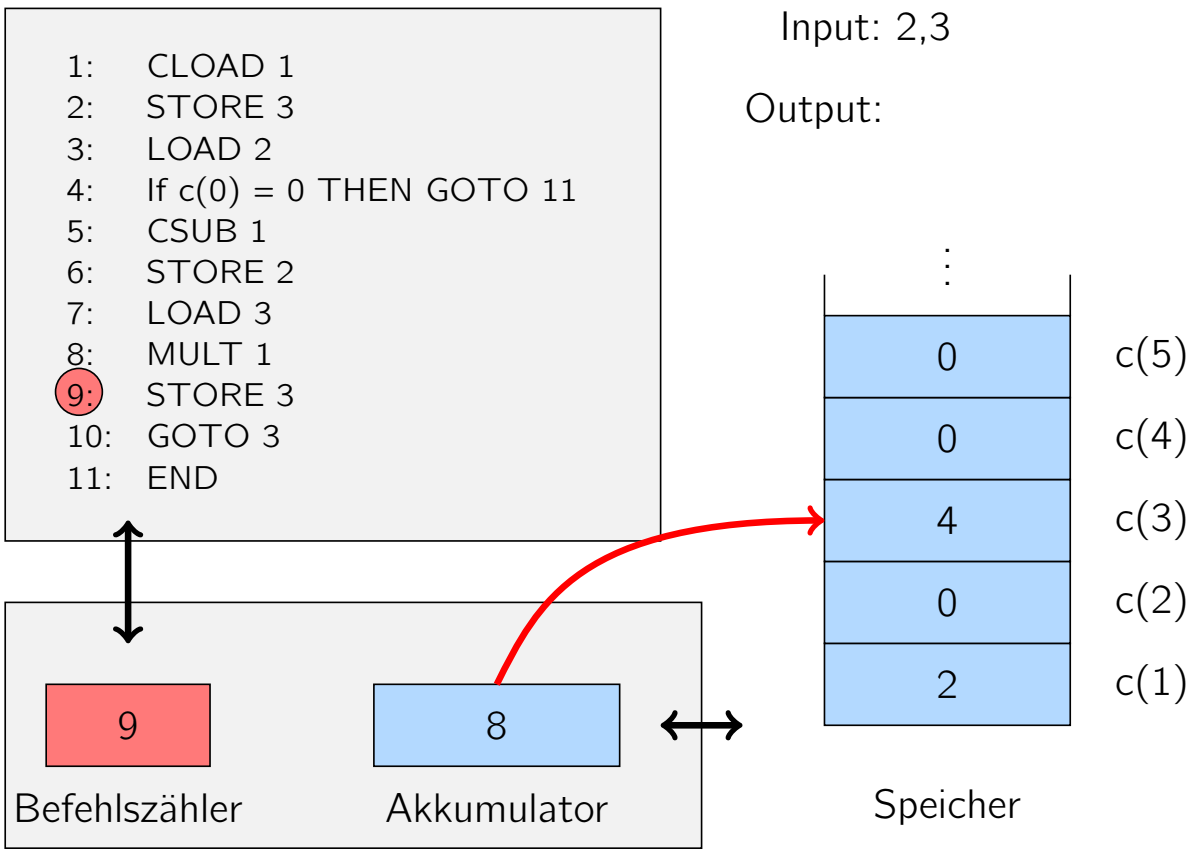
Beispielprogramm für die RAM



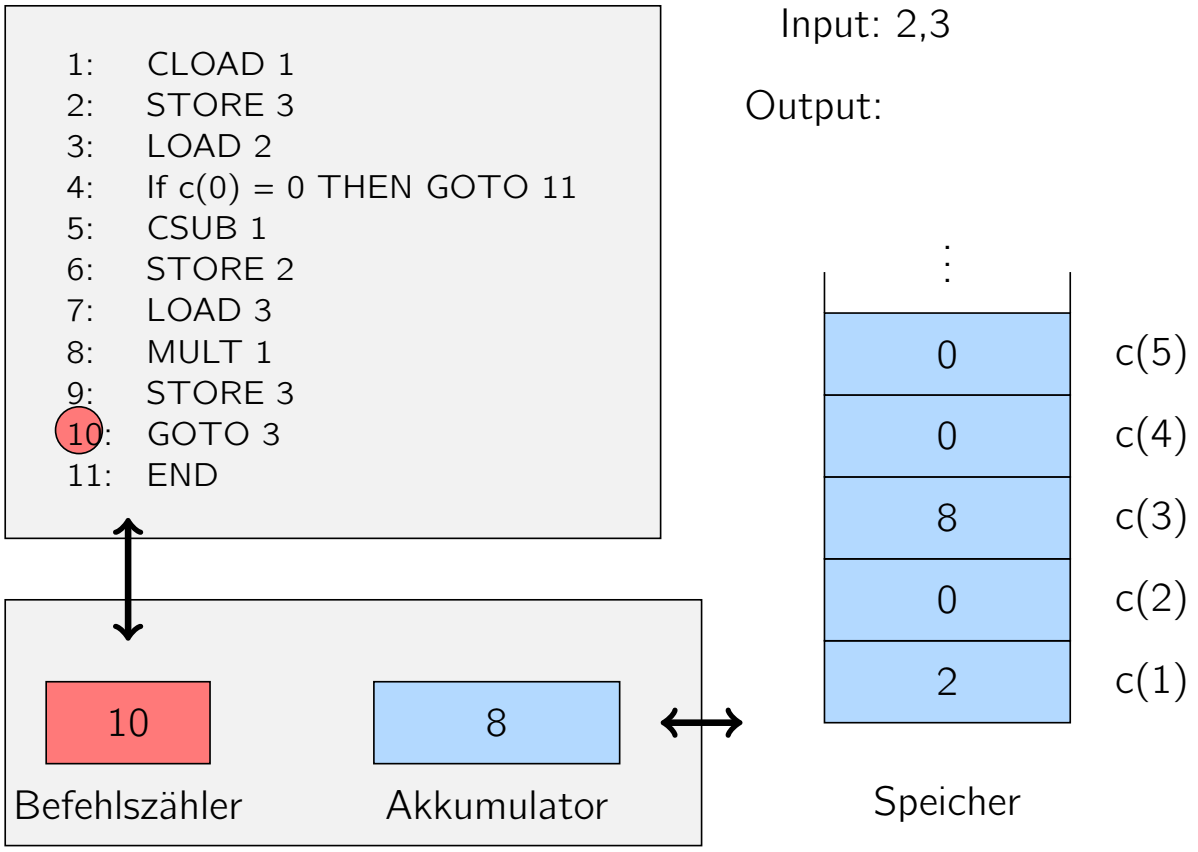
Beispielprogramm für die RAM



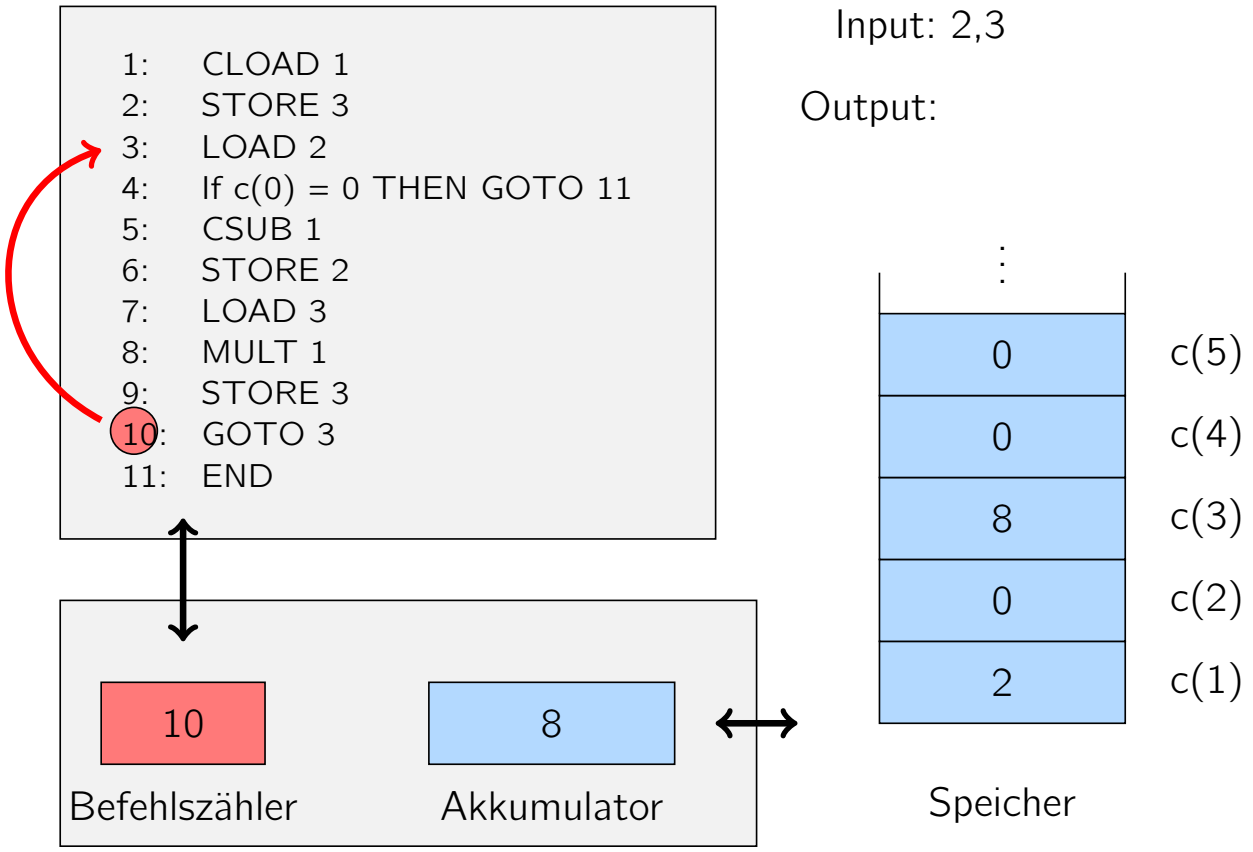
Beispielprogramm für die RAM



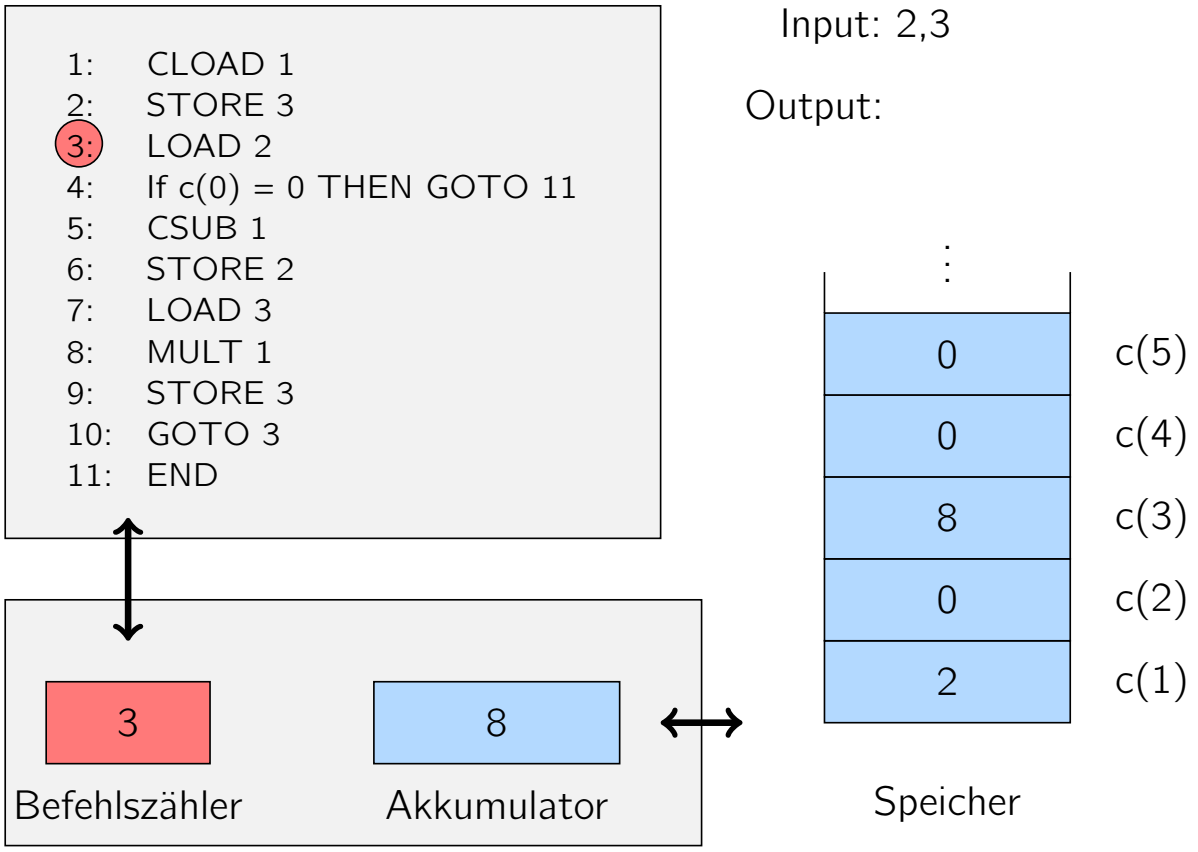
Beispielprogramm für die RAM



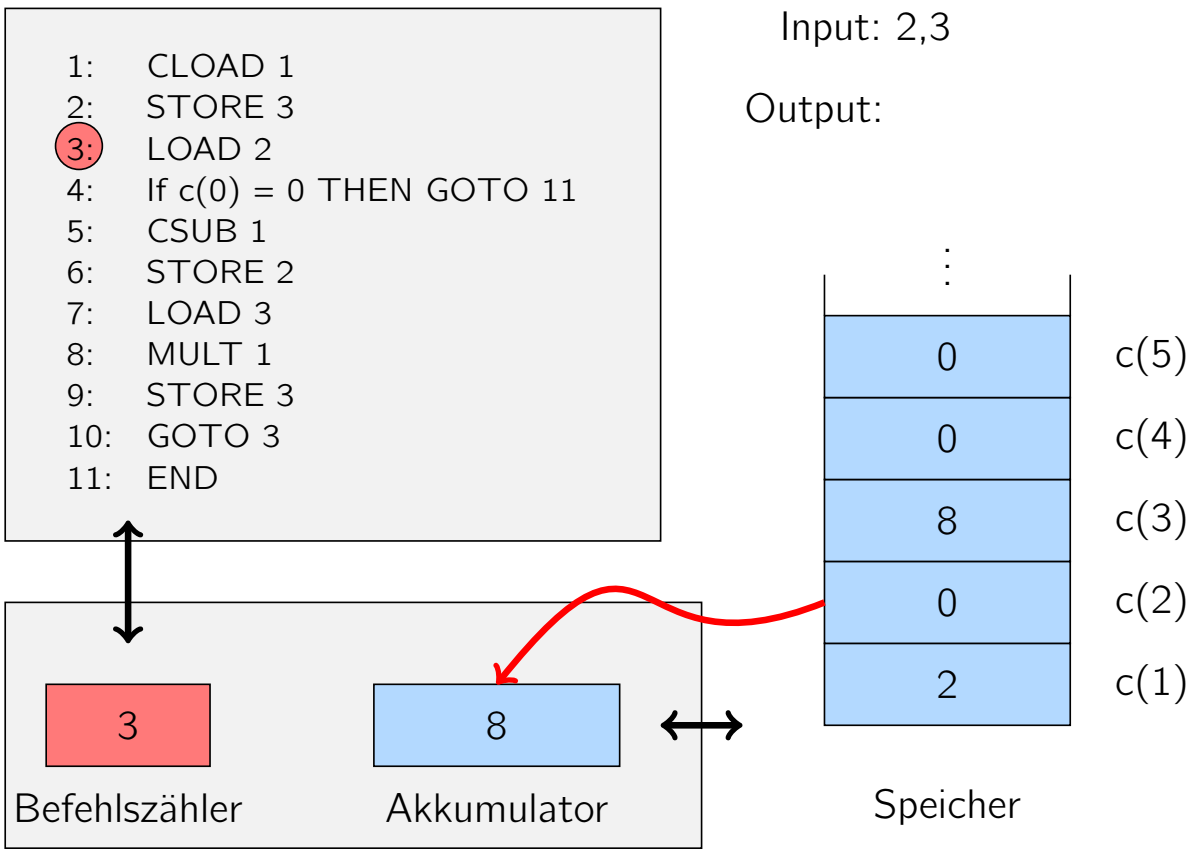
Beispielprogramm für die RAM



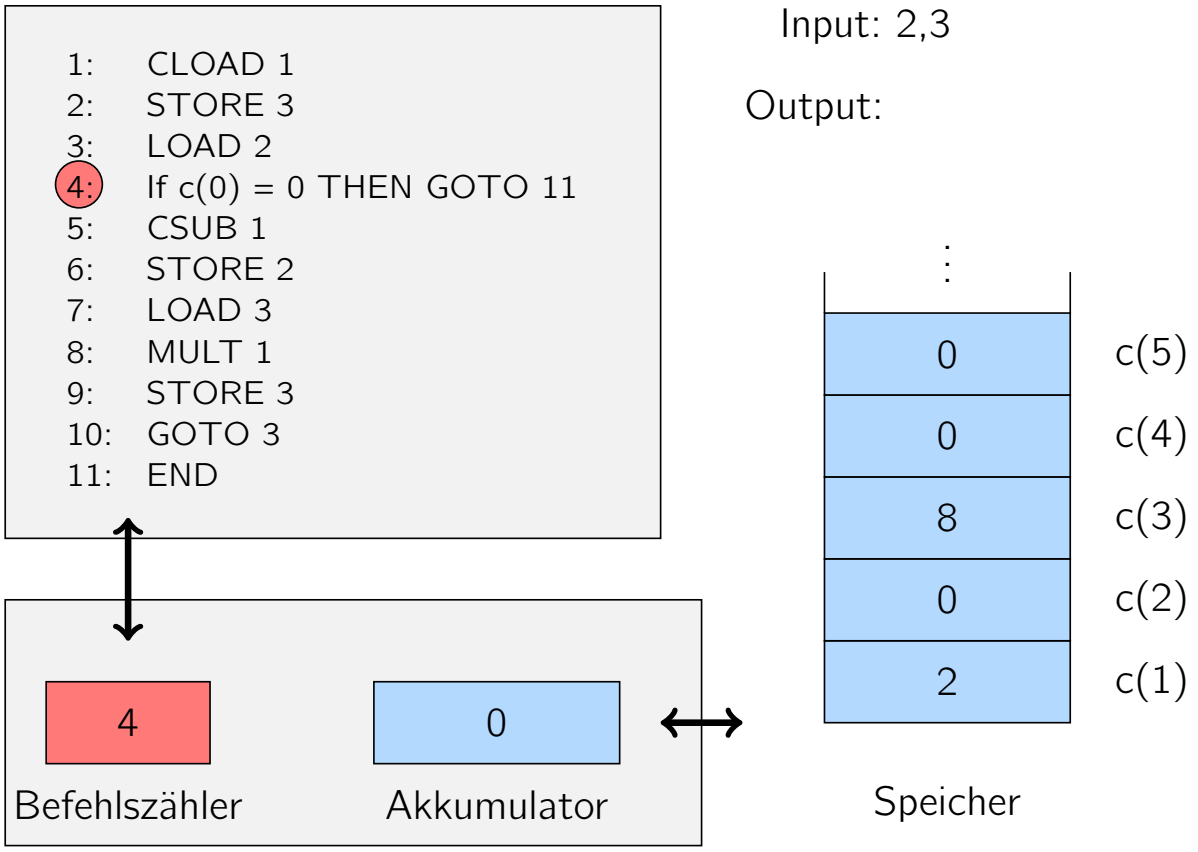
Beispielprogramm für die RAM



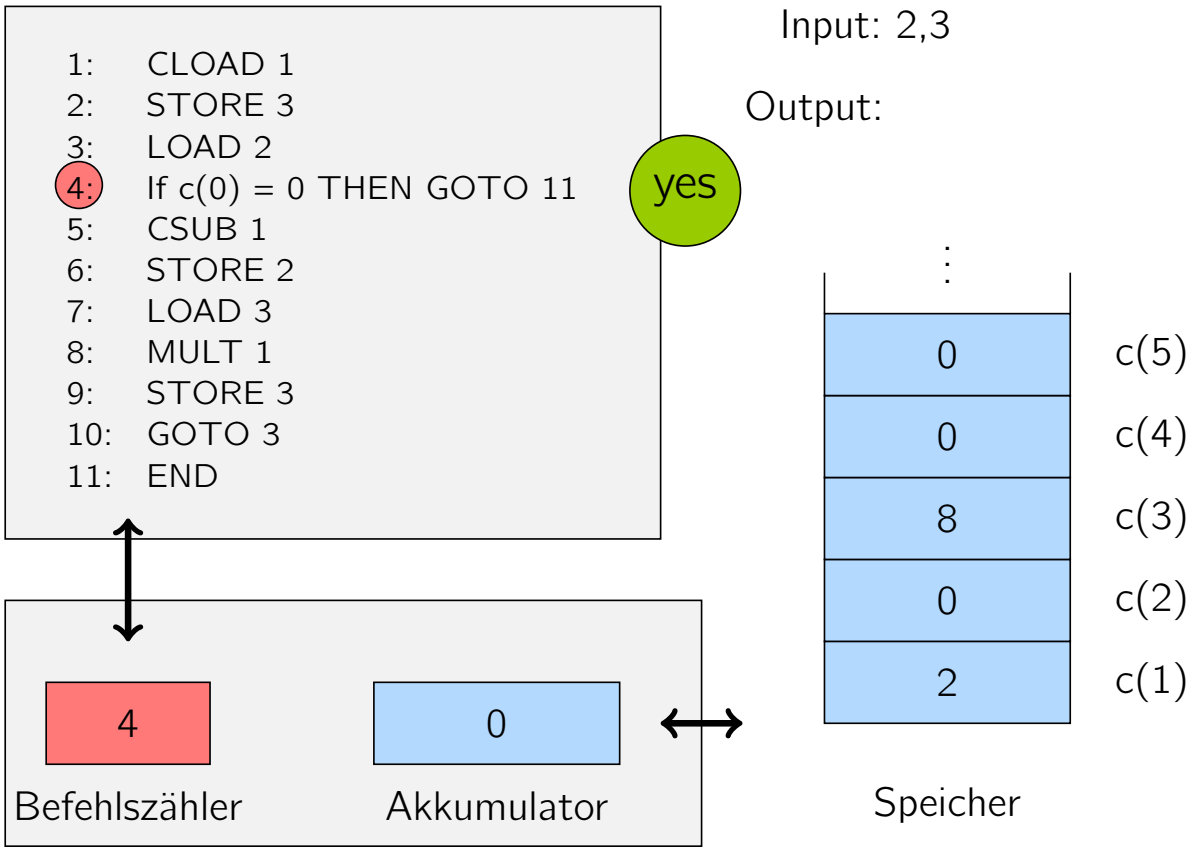
Beispielprogramm für die RAM



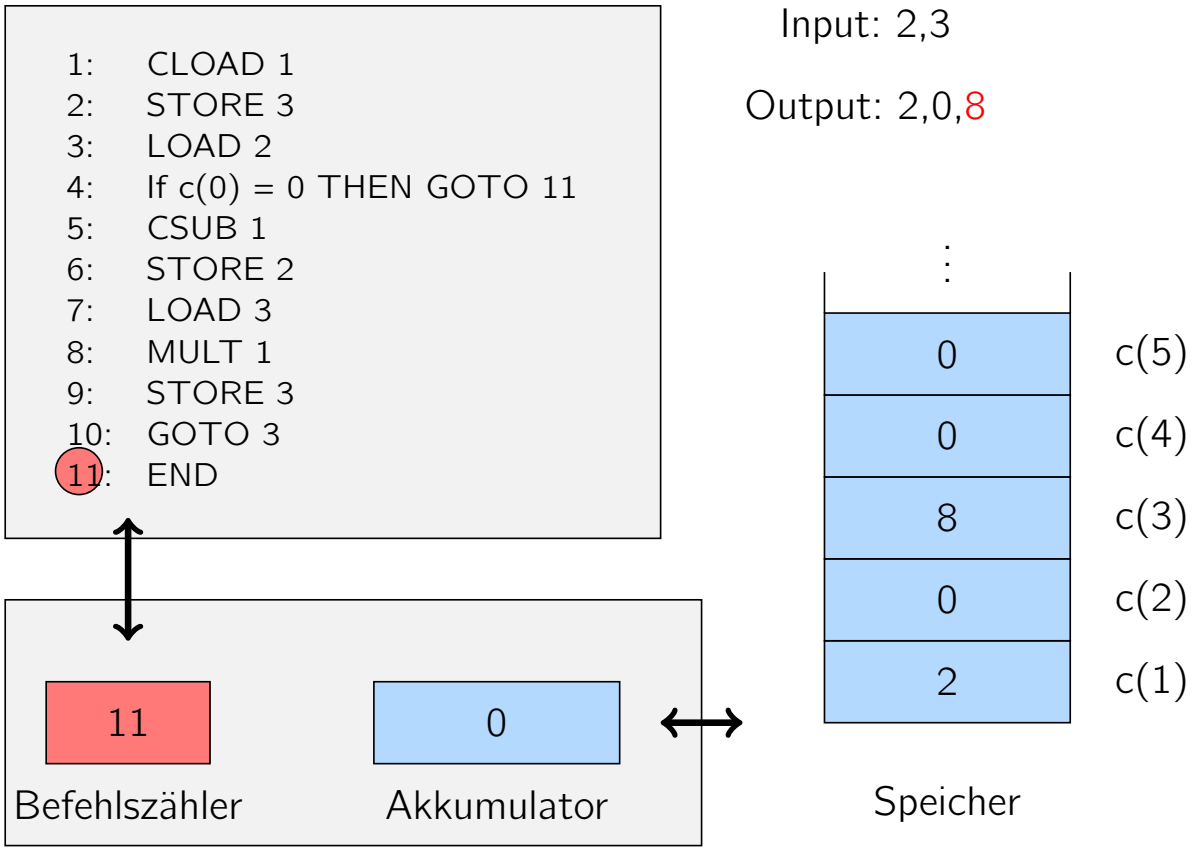
Beispielprogramm für die RAM



Beispielprogramm für die RAM



Beispielprogramm für die RAM



Bemerkungen zur RAM

Auf einer RAM können wir alle Befehle wie beispielsweise Schleifen und Rekursionen, die wir von höheren Programmiersprachen gewohnt sind, realisieren.

Bemerkungen zur RAM

Auf einer RAM können wir alle Befehle wie beispielsweise Schleifen und Rekursionen, die wir von höheren Programmiersprachen gewohnt sind, realisieren.

Modelle für die Rechenzeit

- Uniformes Kostenmaß: Jeder Schritt zählt eine Zeiteinheit.
- Logarithmisches Kostenmaß: Die Laufzeitkosten eines Schrittes sind proportional zur binären Länge der Zahlen in den angesprochenen Registern.

Simulation RAM durch TM

Satz

Für jede im logarithmischen Kostenmaß $t(n)$-zeitbeschränkte RAM R gibt es ein Polynom q und zu diesem eine $O\big(q(n + t(n))\big)$-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 zu diesem eine $O\big(q(n+t(n))\big)$-TM M, die R simuliert.

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

Simulation RAM durch TM – Vorbemerkung zum Beweis

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

Simulation RAM durch TM – Vorbemerkung zum Beweis

- 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 $t'(n) = \alpha(n + t(n))^\beta$.

Simulation RAM durch TM – Vorbemerkung zum Beweis

- 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 $t''(n) = \gamma(t'(n))^2$.

Simulation RAM durch TM – Vorbemerkung zum Beweis

- 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

$$t''(n) \;=\; \gamma(t'(n))^2$$

Simulation RAM durch TM – Vorbemerkung zum Beweis

- 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

$$t''(n) \;=\; \gamma(t'(n))^2 \;=\; \gamma\left(\alpha(n + t(n))^\beta\right)^2$$

Simulation RAM durch TM – Vorbemerkung zum Beweis

- 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

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

Simulation RAM durch TM – Vorbemerkung zum Beweis

- 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 2β konstant sind.

Simulation RAM durch TM – Vorbemerkung zum Beweis

Beobachtung

Die Klasse der Polynome ist unter Hintereinanderausführung abgeschlossen.

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

Simulation RAM durch TM – Vorbemerkung zum Beweis

Beobachtung

Die Klasse der Polynome ist unter Hintereinanderausführung abgeschlossen.

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

Simulation RAM durch TM – Beweis

Beweis des Satzes

- Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert.

Simulation RAM durch TM – Beweis

Beweis des Satzes

- Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert.
- Das RAM-Programm P bestehe aus p Programmzeilen.

Simulation RAM durch TM – Beweis

Beweis des Satzes

- Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert.
- Das RAM-Programm P bestehe aus p Programmzeilen.
- Für jede Programmzeile schreiben wir ein TM-Unterprogramm. Sei M_i das Unterprogramm für Programmzeile i, $1 \leq i \leq p$.

Simulation RAM durch TM – Beweis

Beweis des Satzes

- Wir verwenden eine 2-Band-TM, die die RAM schrittweise simuliert.
- 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.

Simulation RAM durch TM – Beweis

Abspeichern der *RAM-Konfiguration* auf der TM:

Simulation RAM durch TM – Beweis

Abspeichern der *RAM-Konfiguration* auf der TM:

- Den Befehlszähler kann die TM im Zustand abspeichern, da die Länge des RAM-Programms konstant ist.

Simulation RAM durch TM – Beweis

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:

$$\#\#0\#\,\mathsf{bin}(c(0))\#\#\,\mathsf{bin}(i_1)\#\,\mathsf{bin}(c(i_1))\#\#\ldots$$

$$\ldots\#\#\,\mathsf{bin}(i_m)\#\,\mathsf{bin}(c(i_m))\#\#\#,$$

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

Simulation RAM durch TM – Beweis

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:

$$\#\#0\#\,\mathsf{bin}(c(0))\#\#\,\mathsf{bin}(i_1)\#\,\mathsf{bin}(c(i_1))\#\#\ldots$$

$$\ldots\#\#\,\mathsf{bin}(i_m)\#\,\mathsf{bin}(c(i_m))\#\#\#,$$

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.

Simulation RAM durch TM – Beweis

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

Simulation RAM durch TM – Beweis

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.

Simulation RAM durch TM – Beweis

Laufzeitanalyse:

Die Initialisierung erfordert Zeit $O(n)$.

Simulation RAM durch TM – Beweis

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 $n + t(n)$.

Simulation RAM durch TM – Beweis

Laufzeitanalyse:

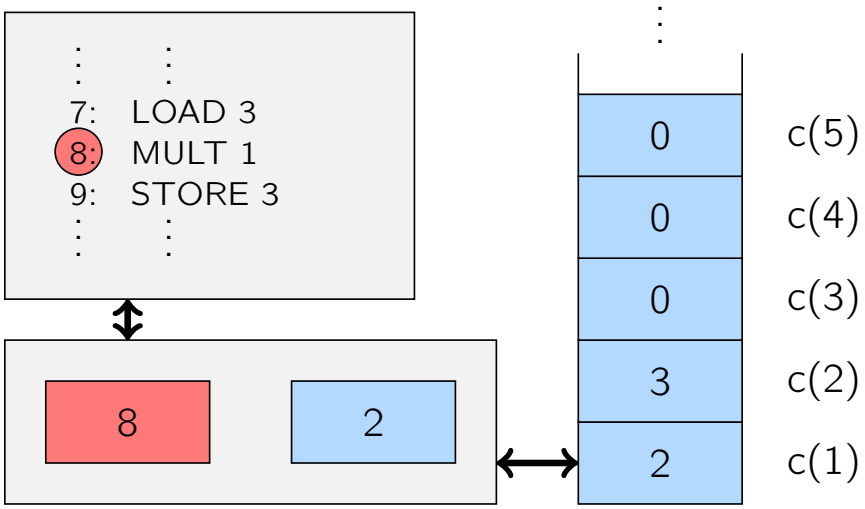
Die Initialisierung erfordert Zeit $O(n)$.

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. □

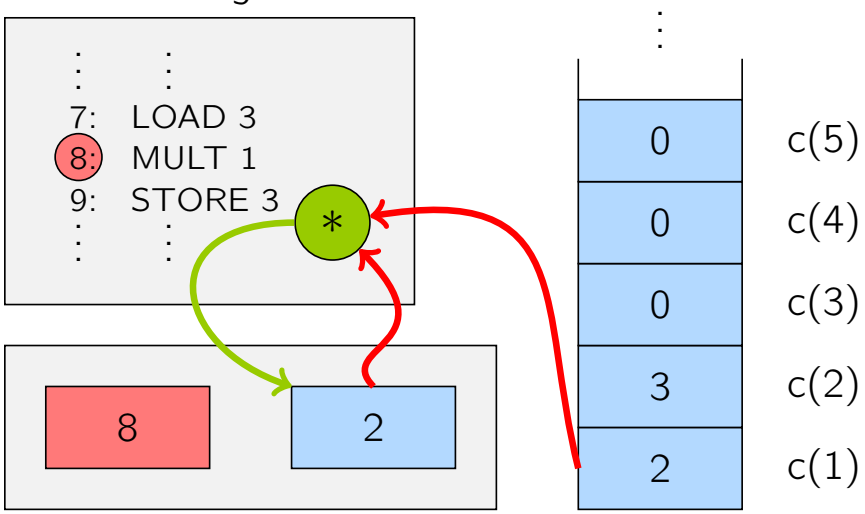
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



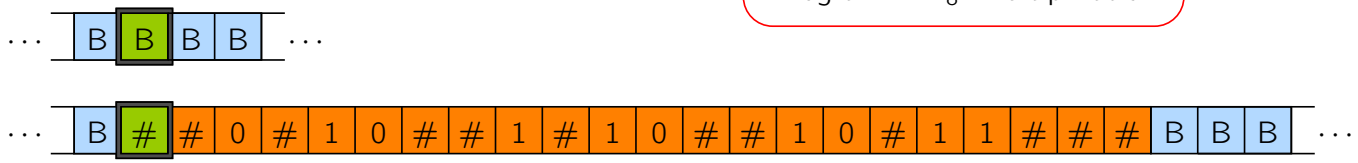
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



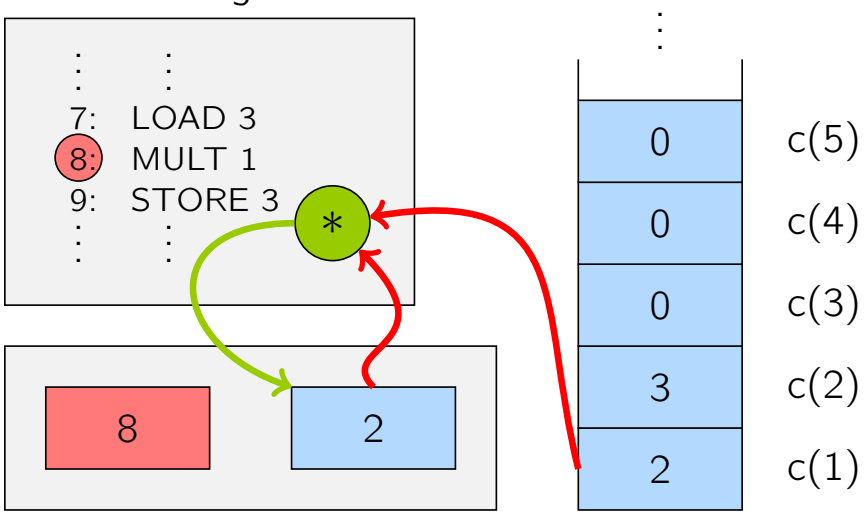
simulierende Turingmaschine

Programm M_8: Multiplikation



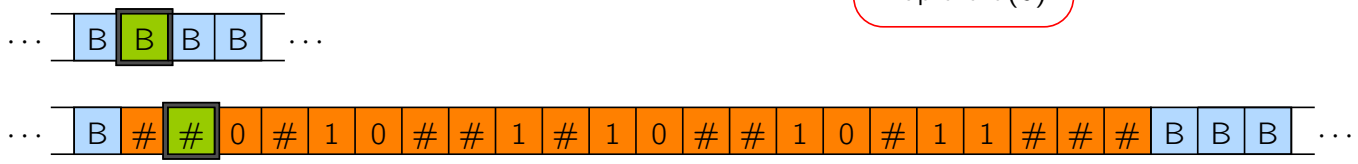
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



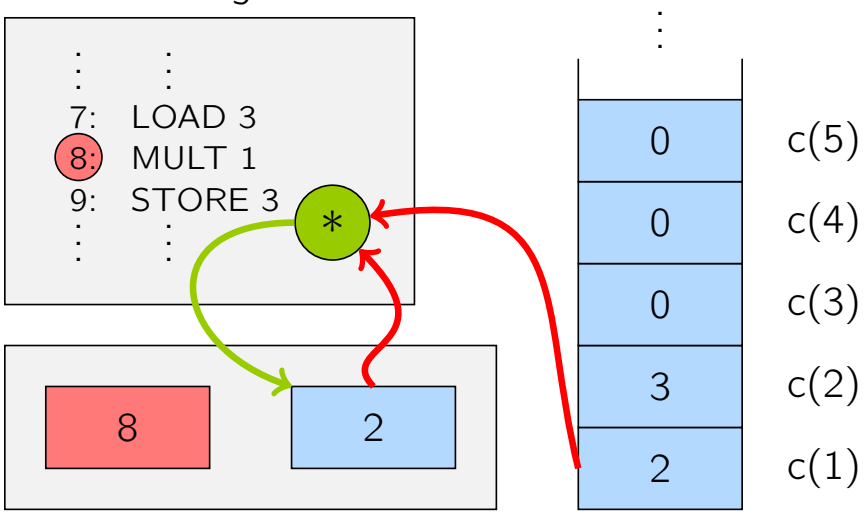
simulierende Turingmaschine

Kopiere $c(0)$

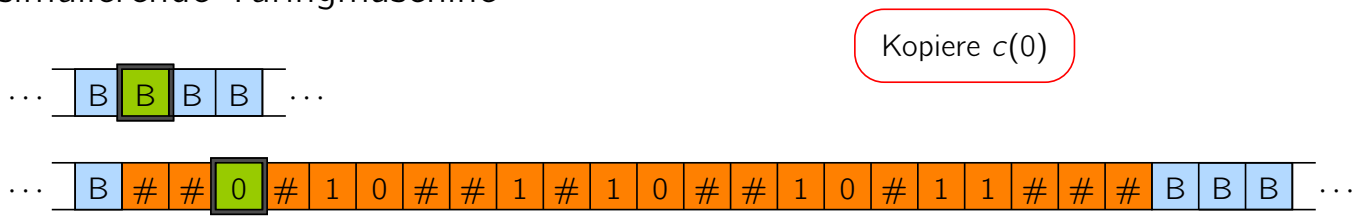


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

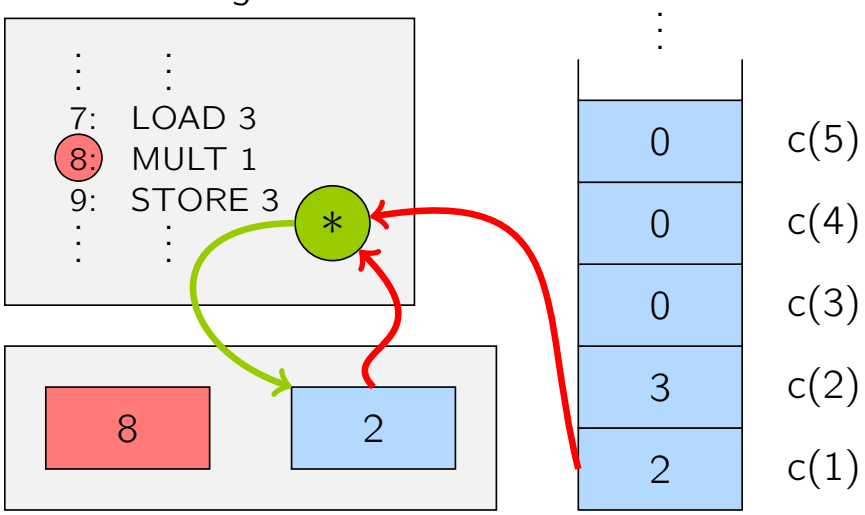


simulierende Turingmaschine

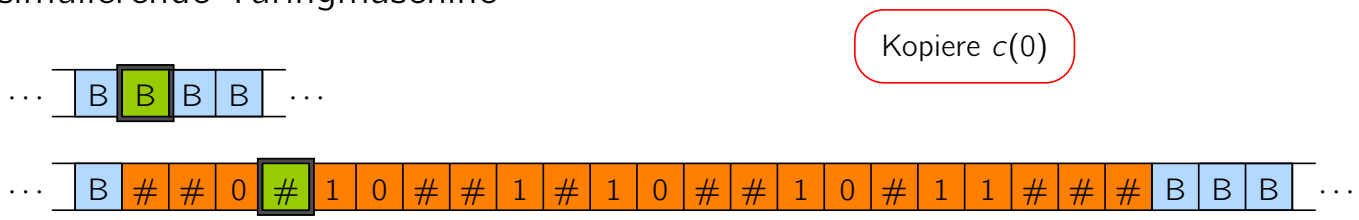


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

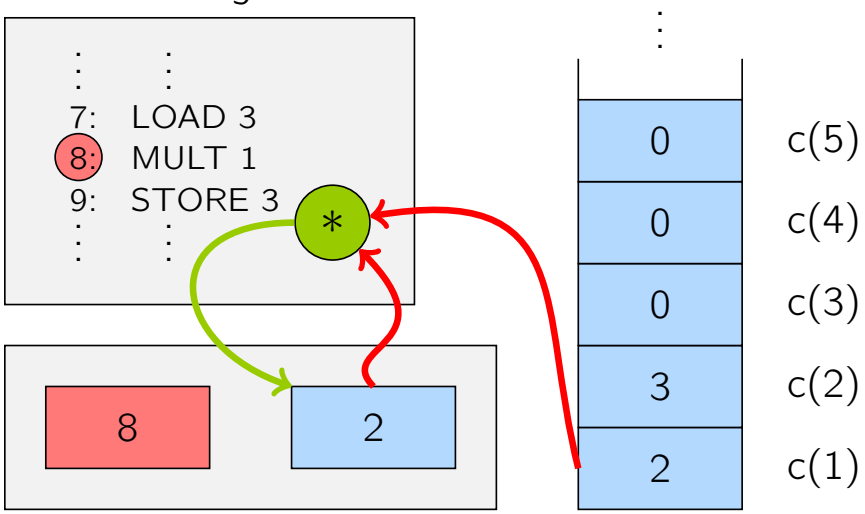


simulierende Turingmaschine



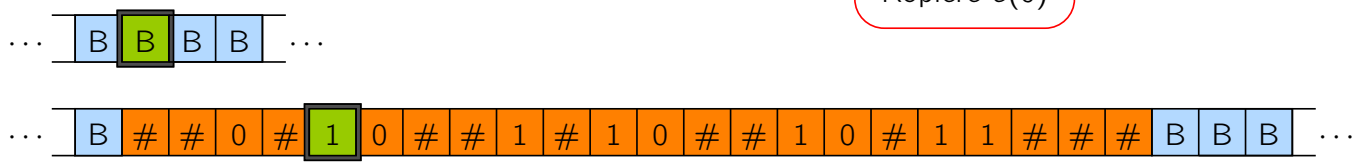
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



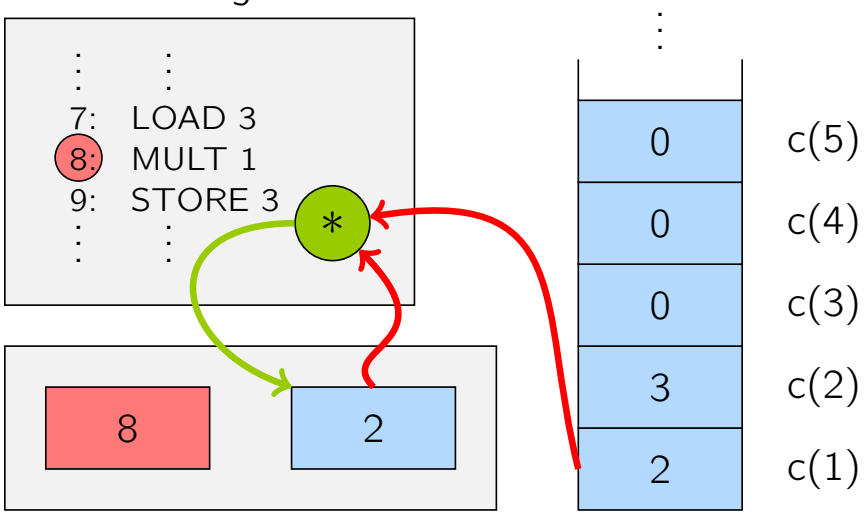
simulierende Turingmaschine

Kopiere $c(0)$



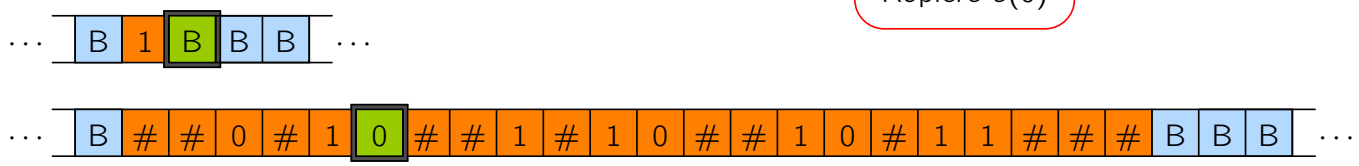
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



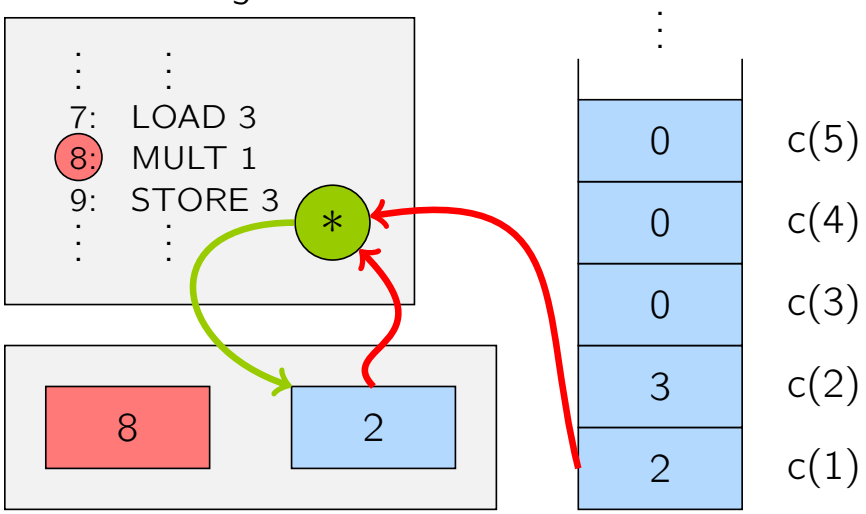
simulierende Turingmaschine

Kopiere $c(0)$

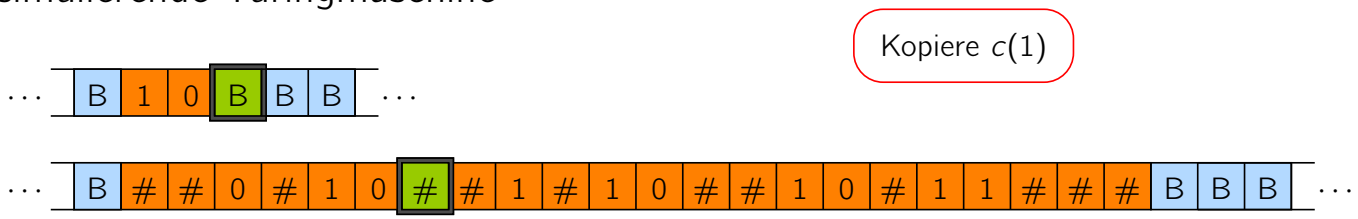


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

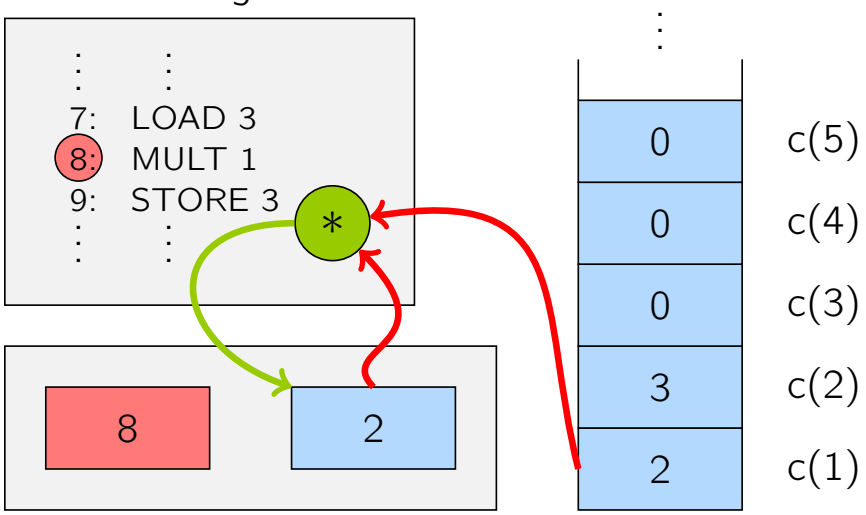


simulierende Turingmaschine

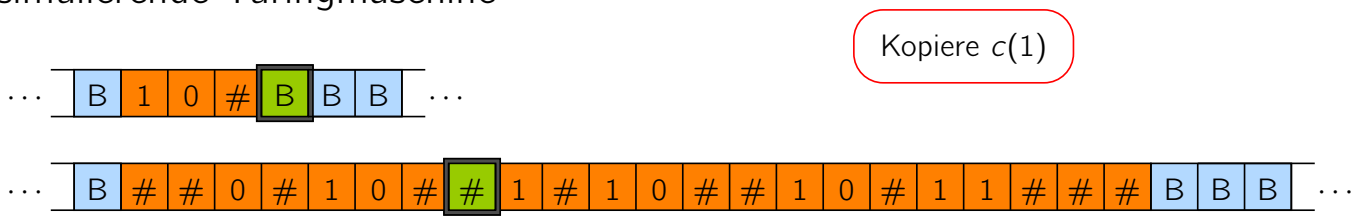


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

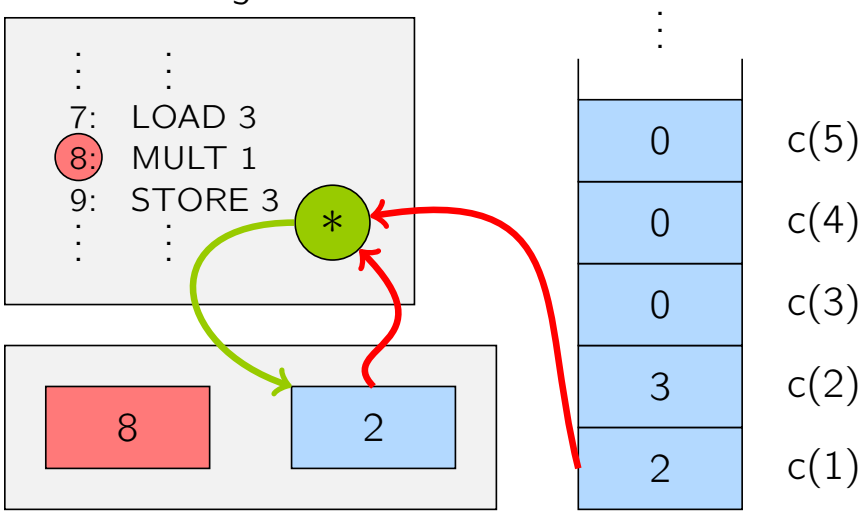


simulierende Turingmaschine



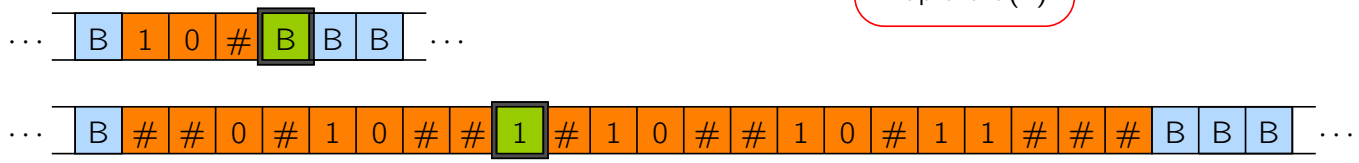
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



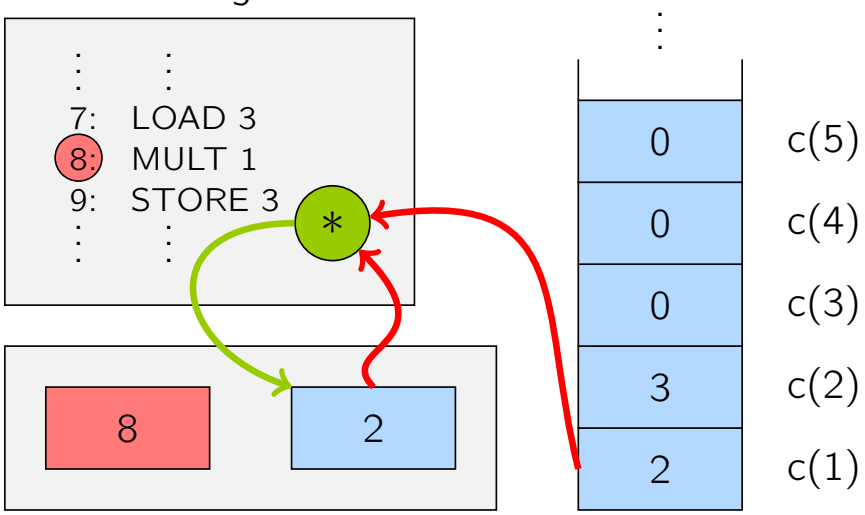
simulierende Turingmaschine

Kopiere $c(1)$



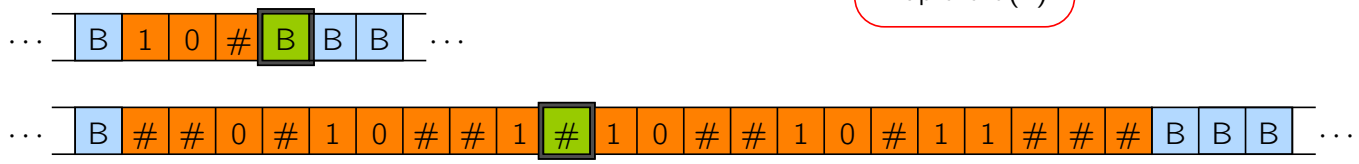
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



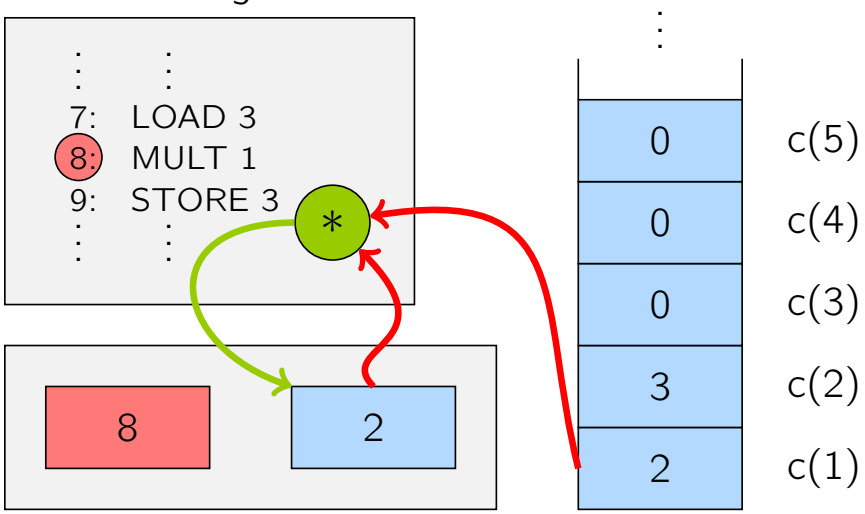
simulierende Turingmaschine

Kopiere $c(1)$



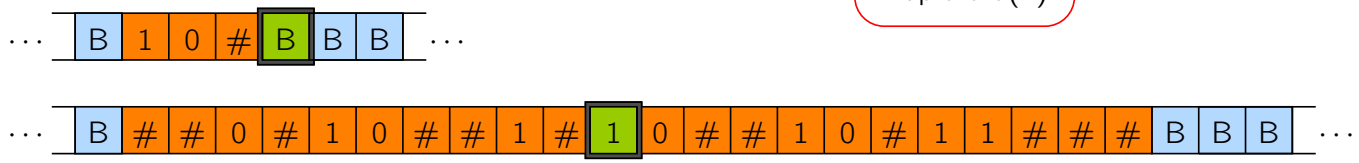
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



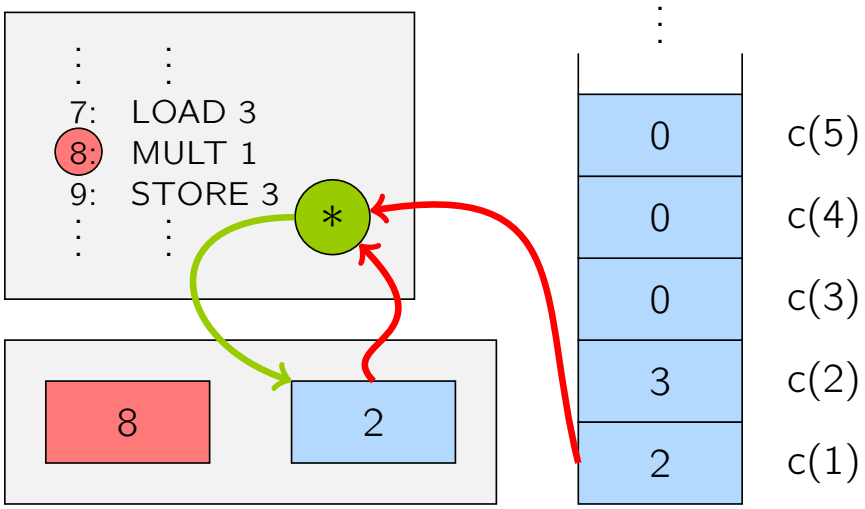
simulierende Turingmaschine

Kopiere $c(1)$

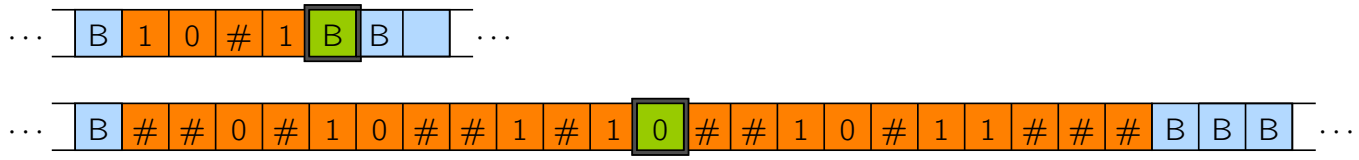


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

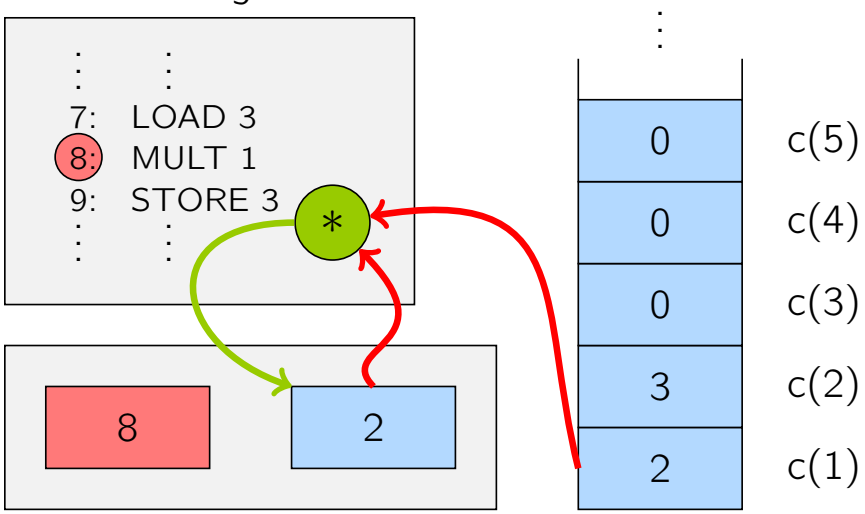


simulierende Turingmaschine

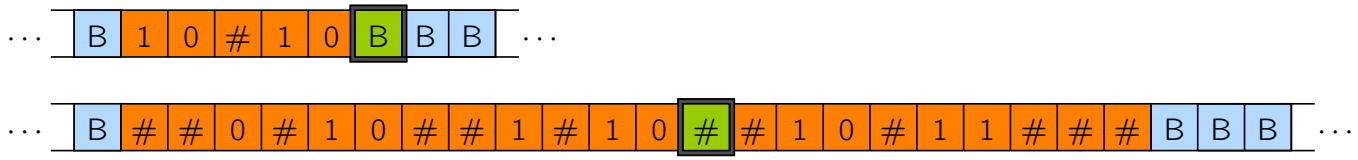


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

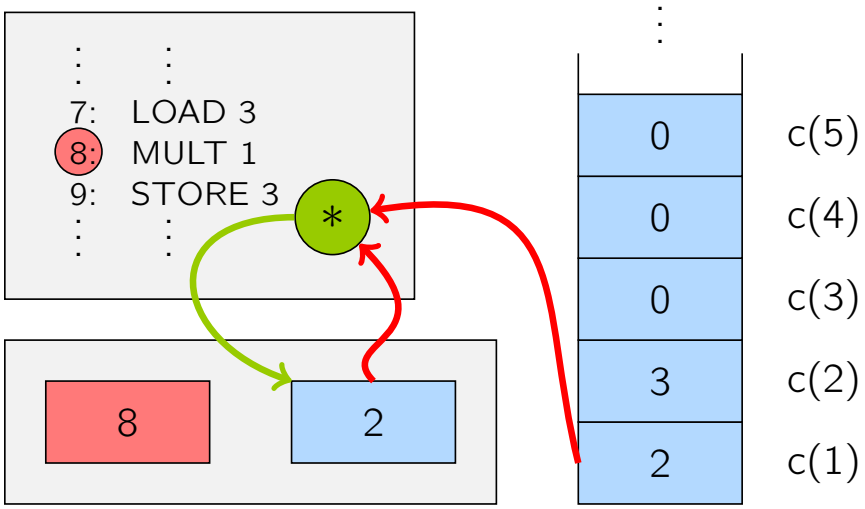


simulierende Turingmaschine



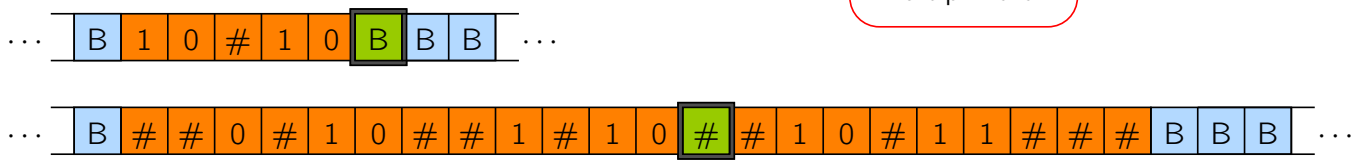
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



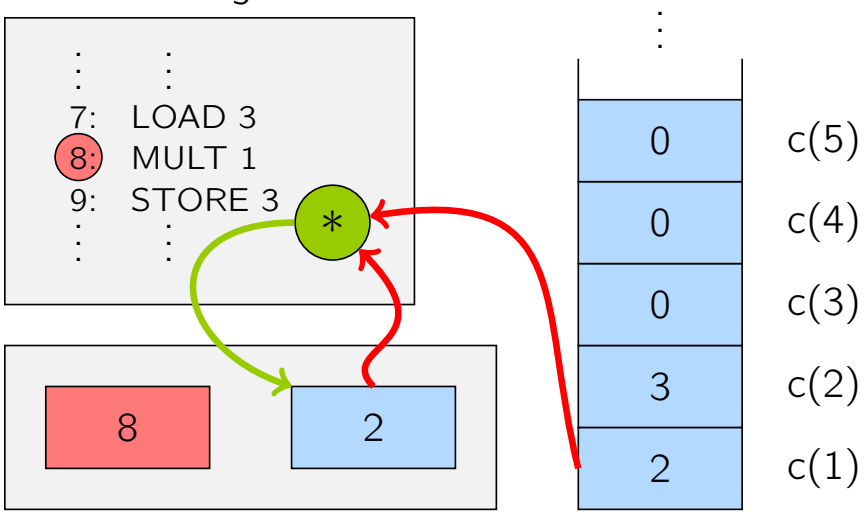
simulierende Turingmaschine

Multiplizieren

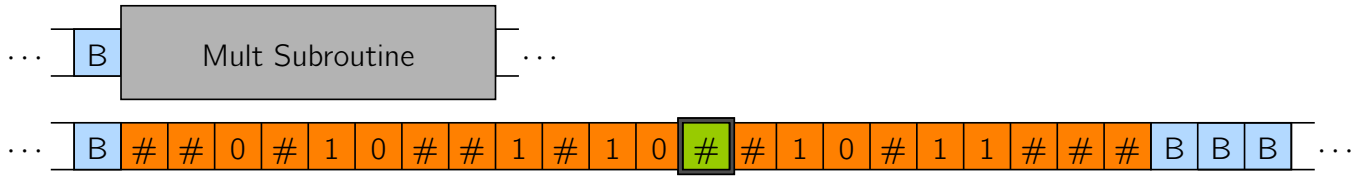


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

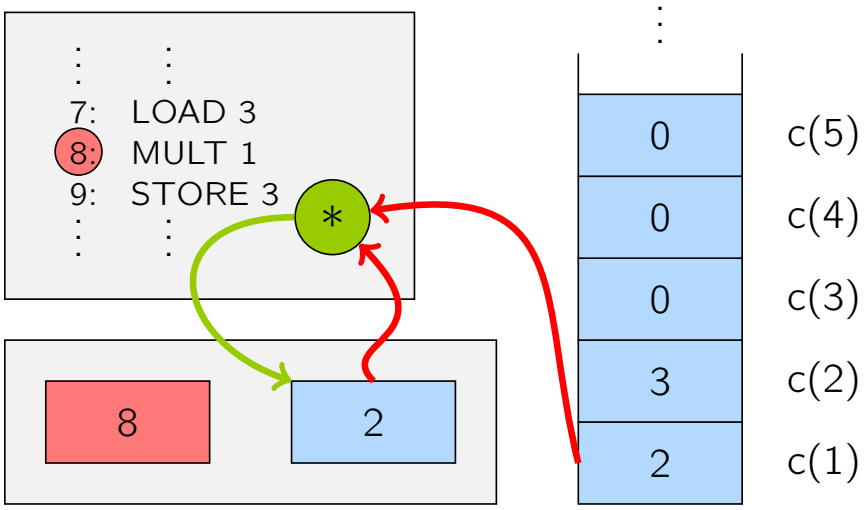


simulierende Turingmaschine

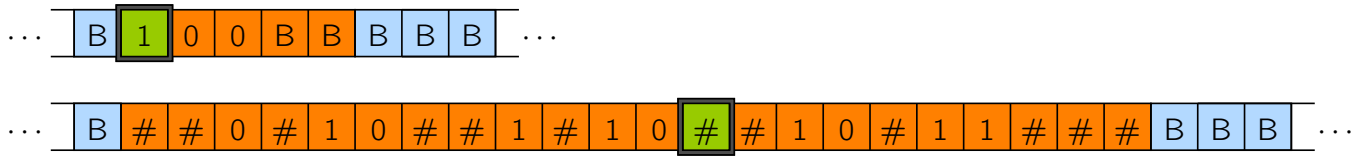


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

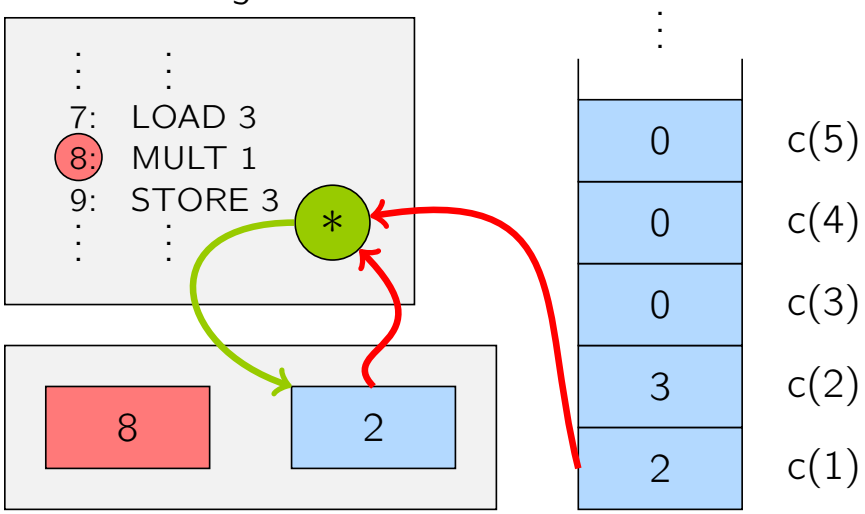


simulierende Turingmaschine

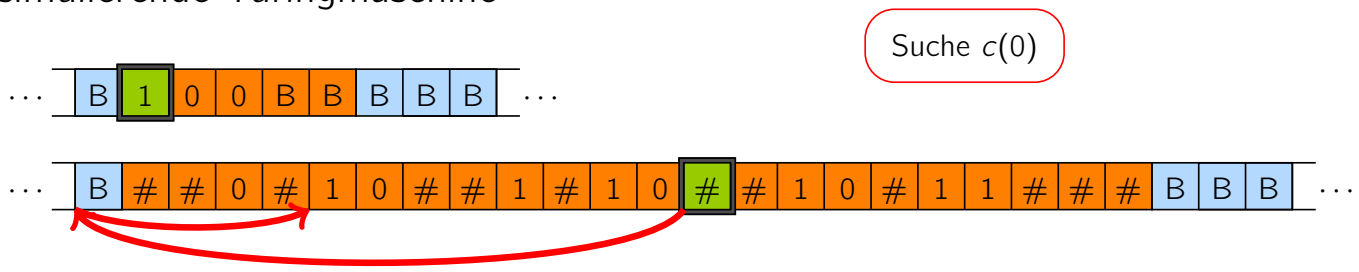


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

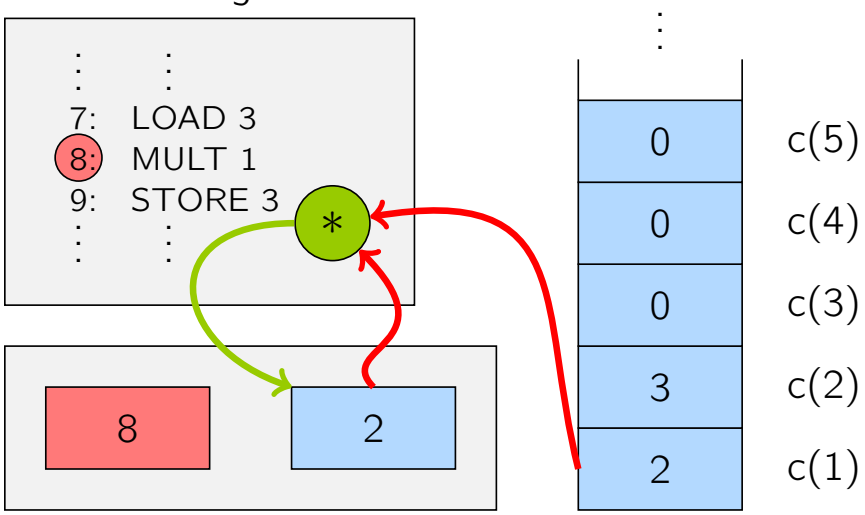


simulierende Turingmaschine

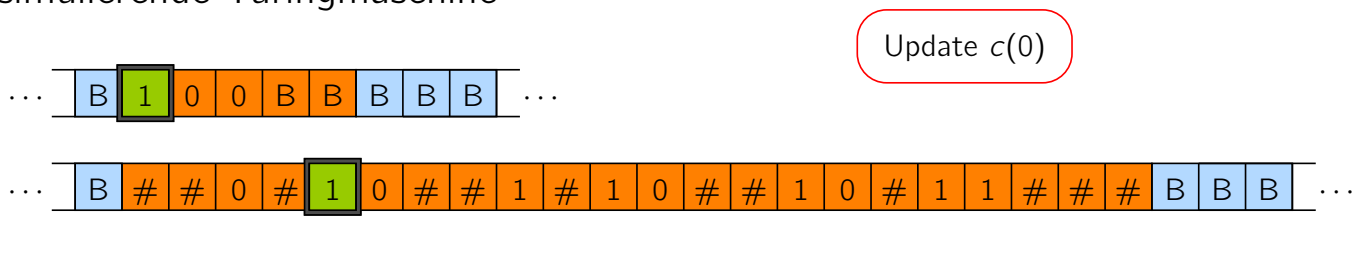


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

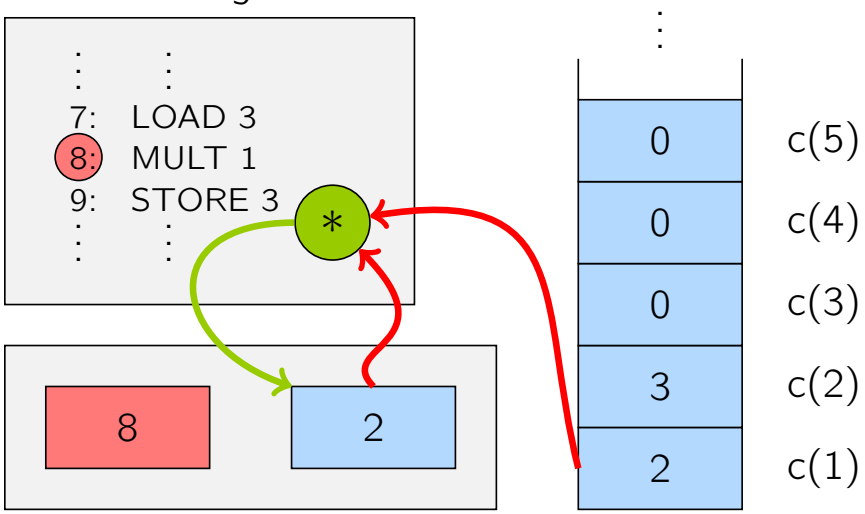


simulierende Turingmaschine



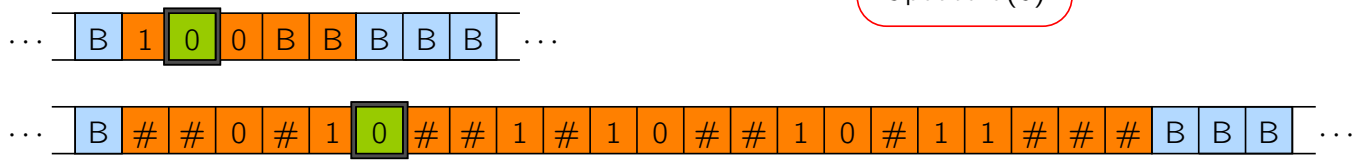
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



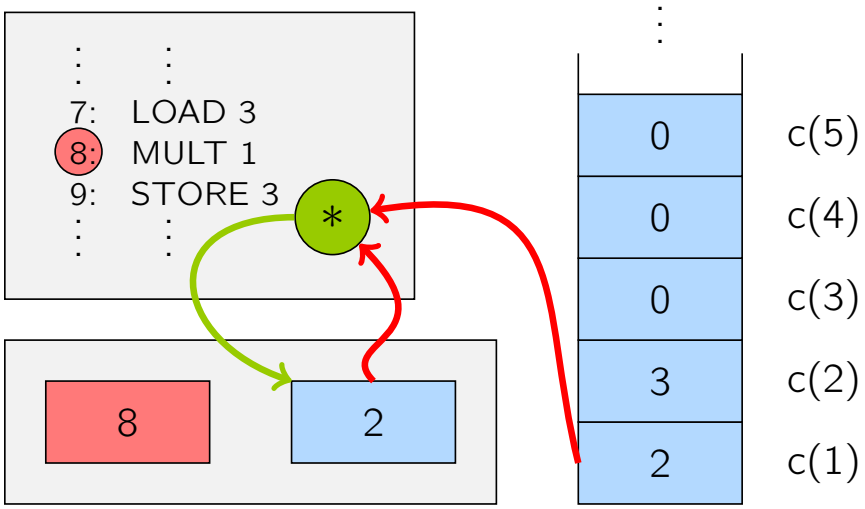
simulierende Turingmaschine

Update $c(0)$

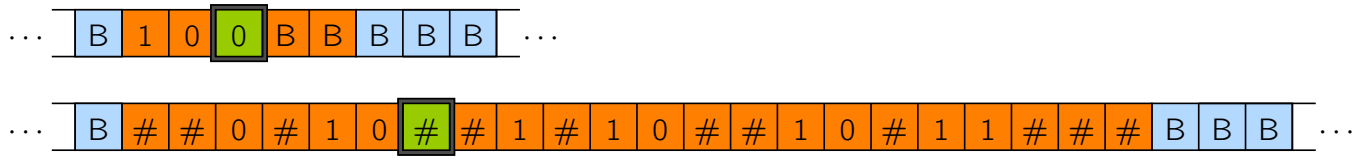


Simulation RAM durch TM – Illustration

simulierte Registermaschine M

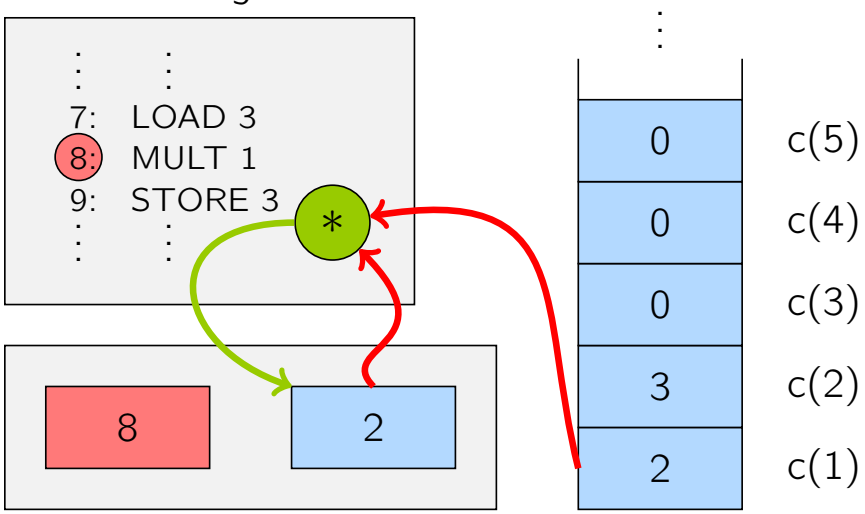


simulierende Turingmaschine



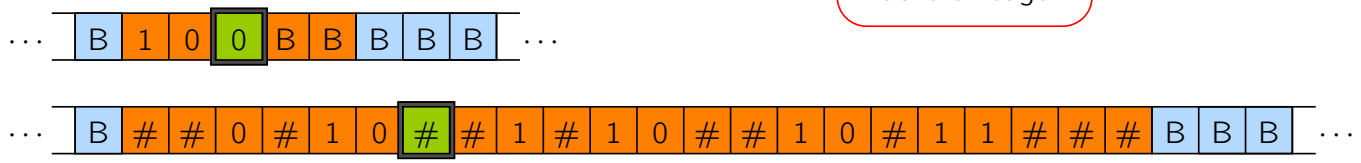
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



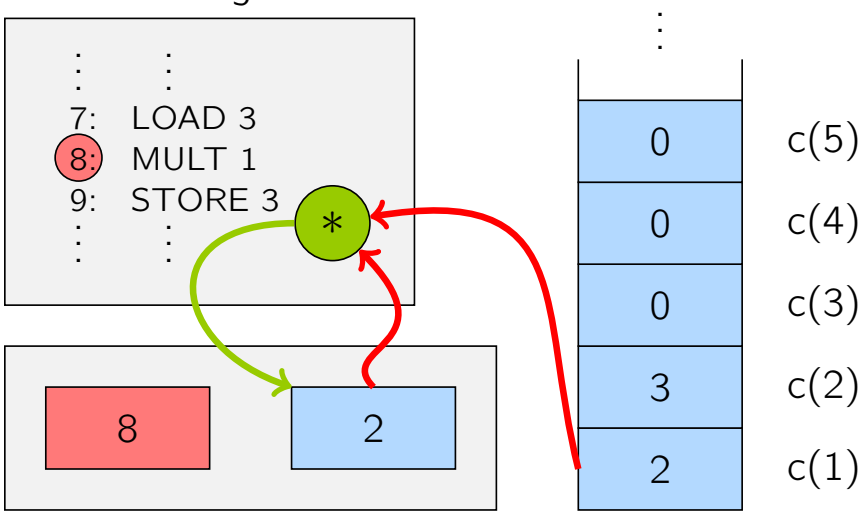
simulierende Turingmaschine

Lücke erzeugen



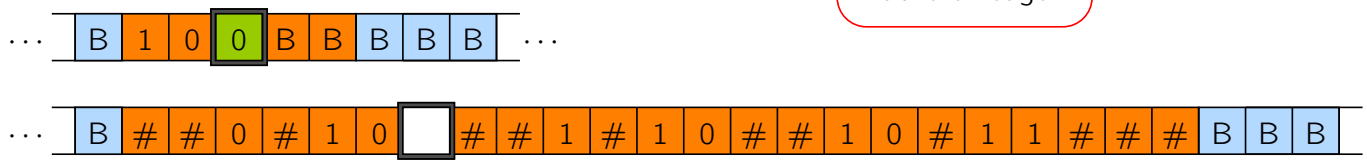
Simulation RAM durch TM – Illustration

simulierte Registermaschine M



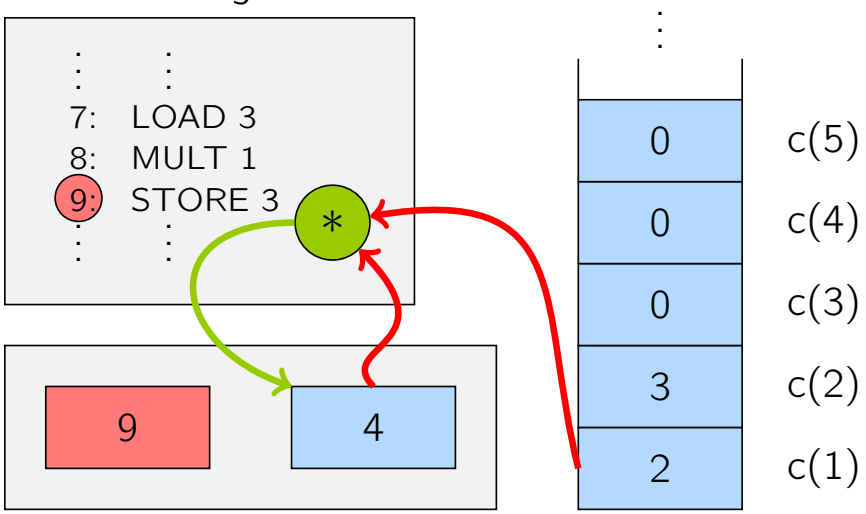
simulierende Turingmaschine

Lücke erzeugen

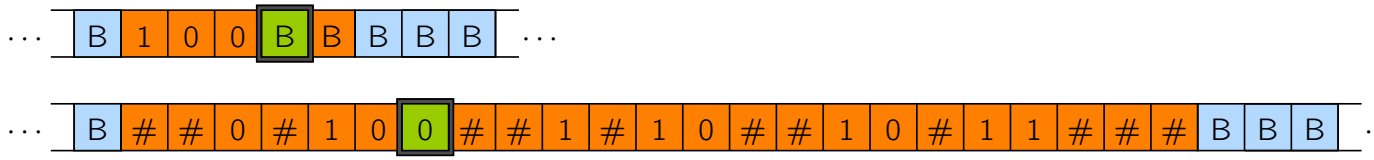


Simulation RAM durch TM – Illustration

simulierte Registermaschine M



simulierende Turingmaschine



Simulation 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.*

Simulation TM durch RAM – Beweis

Beweis des Satzes

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

Simulation TM durch RAM – Beweis

Beweis des Satzes

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

Simulation TM durch RAM – Beweis

Beweis des Satzes

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

Simulation TM durch RAM – Beweis

Beweis des Satzes

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

Simulation TM durch RAM – Beweis

Beweis des Satzes

- O.B.d.A. nehmen wir an, es handelt sich um eine TM mit einseitig beschränktem Band, deren Zellen mit $0, 1, 2, 3, \ldots$ durchnummeriert sind. (Vgl. Übung)
- 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.
- Die Register $3, 4, 5, 6, \ldots$ speichern die Inhalte der besuchten Bandpositionen $0, 1, 2, 3, \ldots$

Simulation TM durch RAM – Beweis

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.

Simulation TM durch RAM – Beweis

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.

Simulation TM durch RAM – Beweis

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.

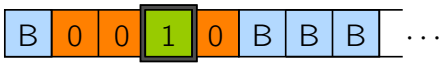
Durchführung des TM-Übergangs

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.

Simulation TM durch RAM – Illustration

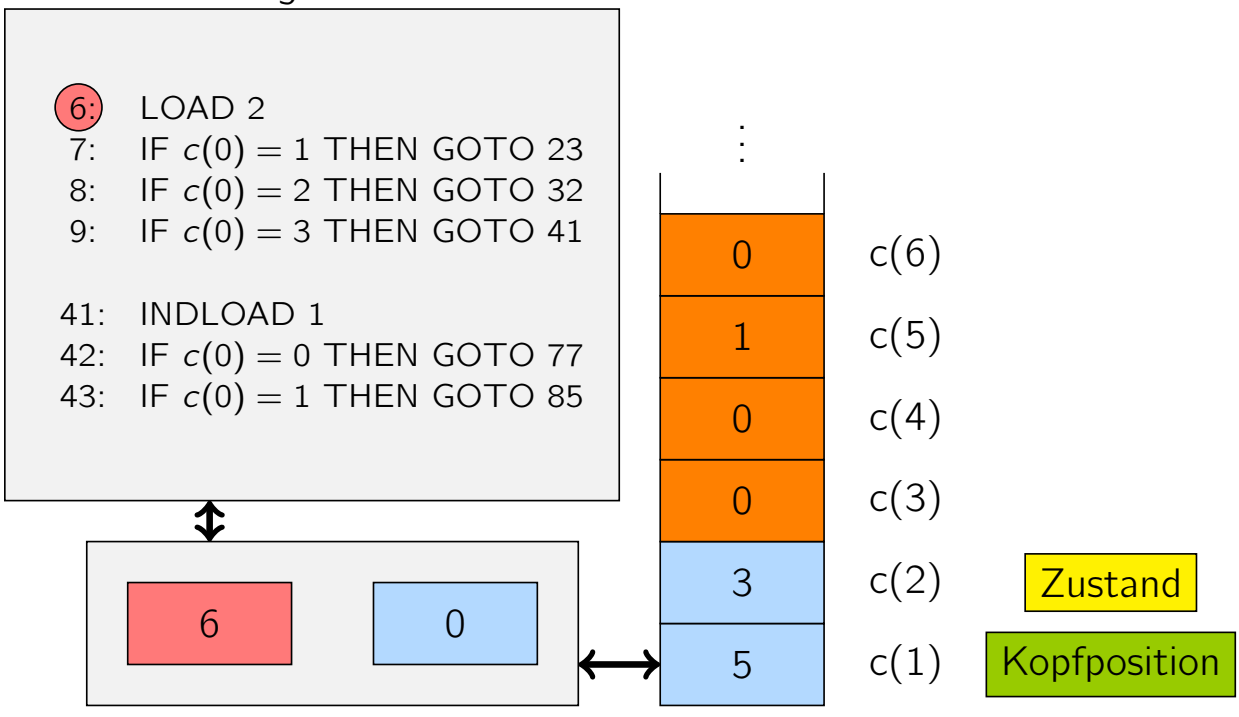
simulierte Turingmaschine M



δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

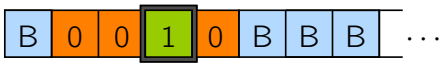
q_3

simulierende Registermaschine



Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

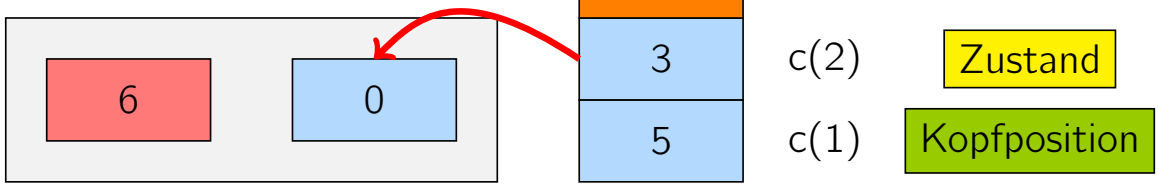


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

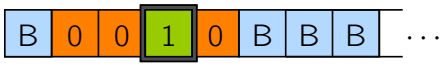
simulierende Registermaschine

```
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
```



Simulation TM durch RAM – Illustration

simulierte Turingmaschine M



δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

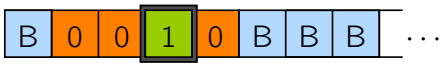
q_3

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
```

Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

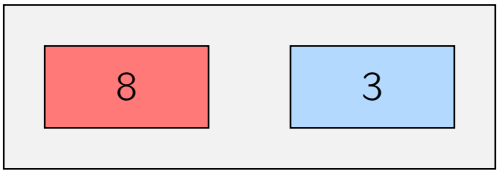


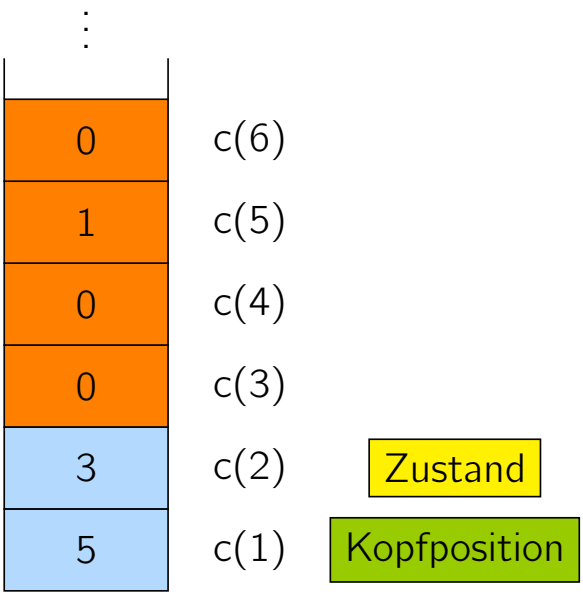
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

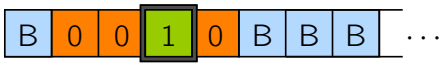
```
 6:   LOAD 2
 7:   IF c(0) = 1 THEN GOTO 23
 8:   IF c(0) = 2 THEN GOTO 32
 9:   IF c(0) = 3 THEN GOTO 41

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





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

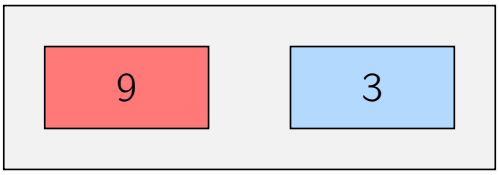


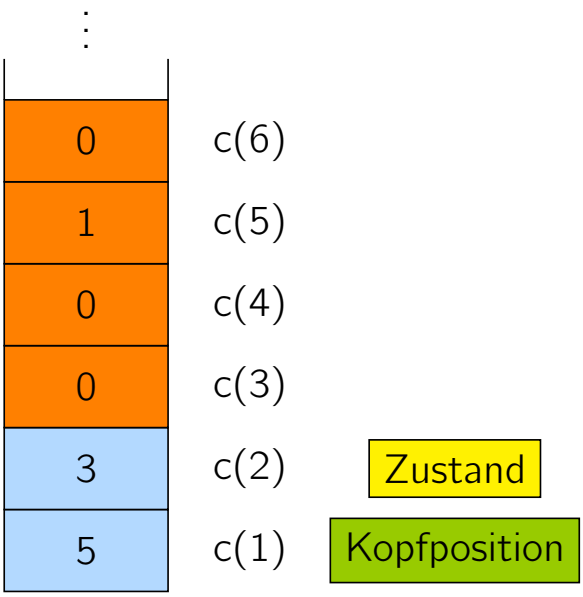
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

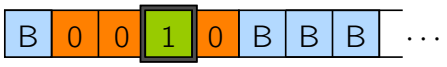
```
 6:   LOAD 2
 7:   IF c(0) = 1 THEN GOTO 23
 8:   IF c(0) = 2 THEN GOTO 32
 9:   IF c(0) = 3 THEN GOTO 41

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





Simulation TM durch RAM – Illustration

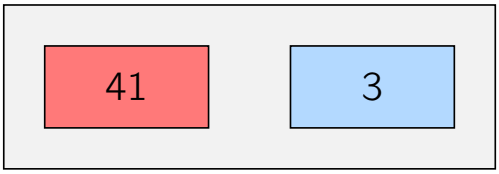
simulierte Turingmaschine M

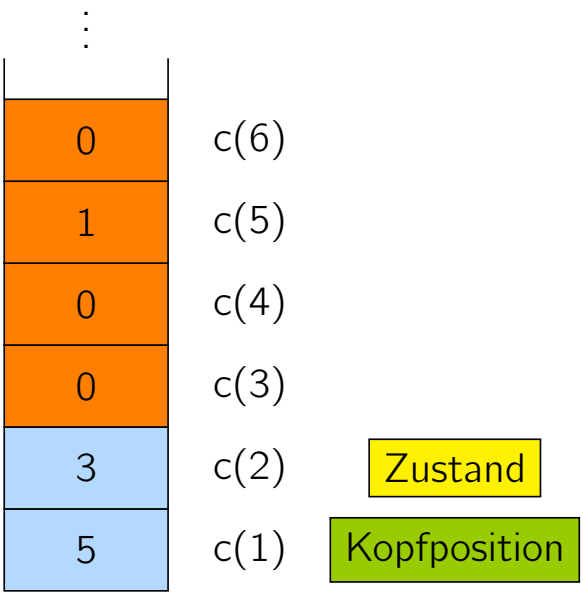


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

simulierende Registermaschine

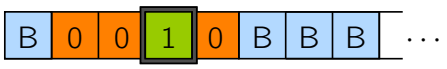
```
6:   LOAD 2
7:   IF c(0) = 1 THEN GOTO 23
8:   IF c(0) = 2 THEN GOTO 32
9:   IF c(0) = 3 THEN GOTO 41

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





Simulation TM durch RAM – Illustration

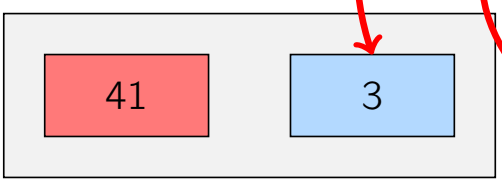
simulierte Turingmaschine M

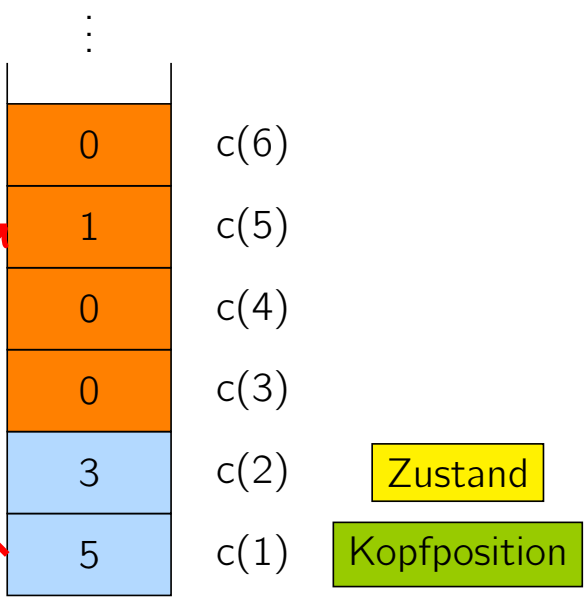


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

simulierende Registermaschine

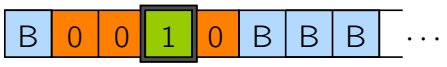
```
6:   LOAD 2
7:   IF c(0) = 1 THEN GOTO 23
8:   IF c(0) = 2 THEN GOTO 32
9:   IF c(0) = 3 THEN GOTO 41

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





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

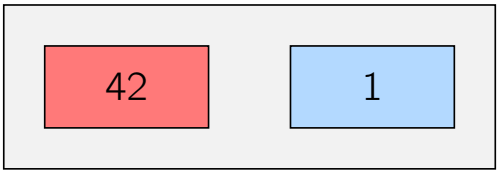


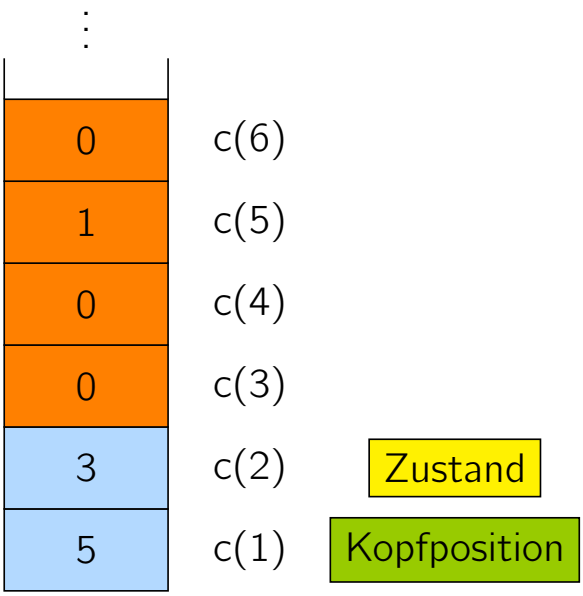
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

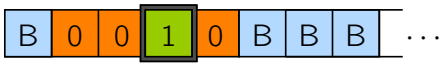
```
 6:  LOAD 2
 7:  IF c(0) = 1 THEN GOTO 23
 8:  IF c(0) = 2 THEN GOTO 32
 9:  IF c(0) = 3 THEN GOTO 41

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





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

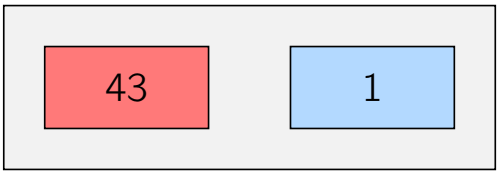


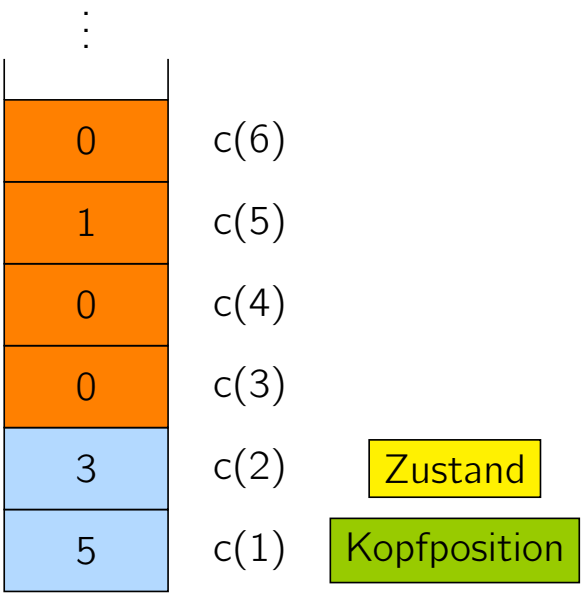
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

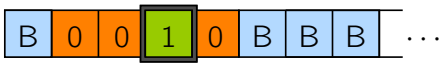
```
 6:  LOAD 2
 7:  IF c(0) = 1 THEN GOTO 23
 8:  IF c(0) = 2 THEN GOTO 32
 9:  IF c(0) = 3 THEN GOTO 41

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





Simulation TM durch RAM – Illustration

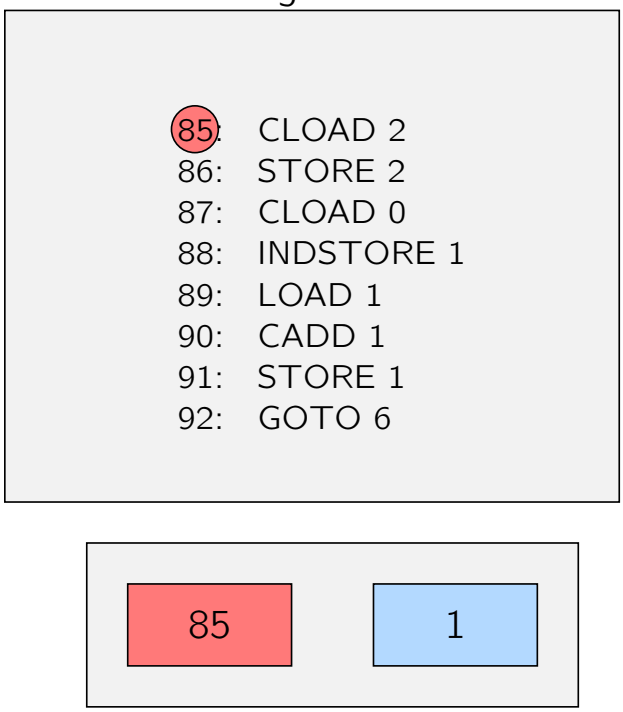
simulierte Turingmaschine M

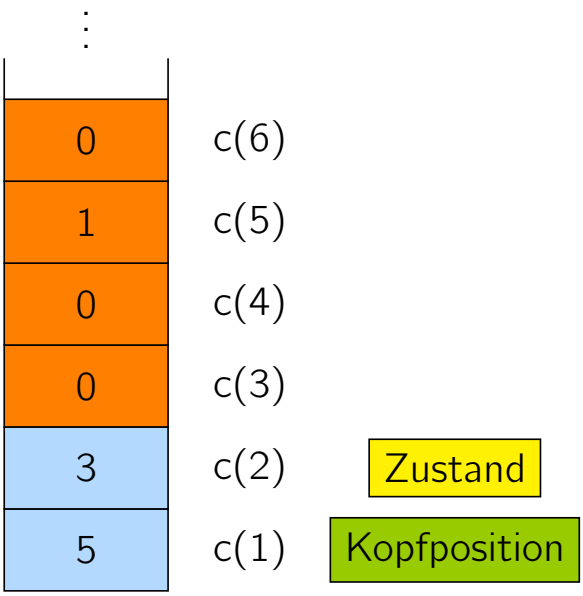


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

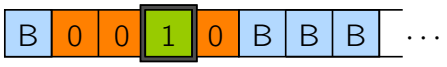
simulierende Registermaschine





Simulation TM durch RAM – Illustration

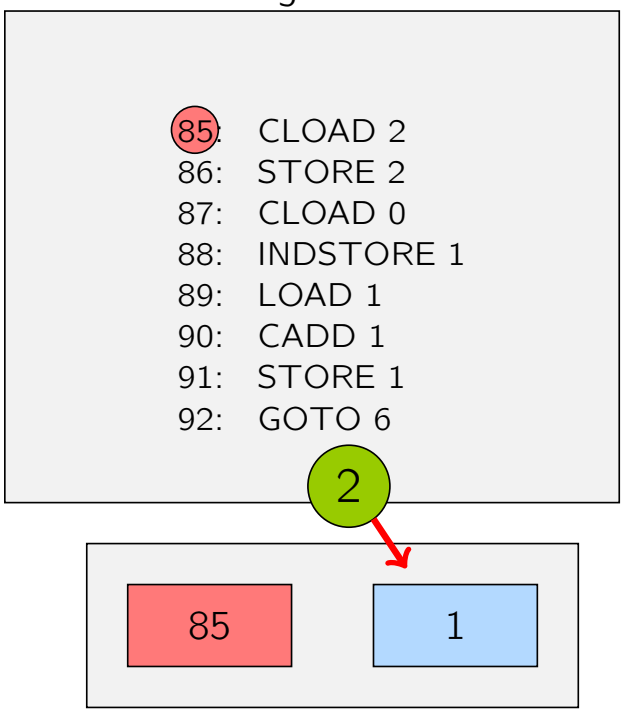
simulierte Turingmaschine M

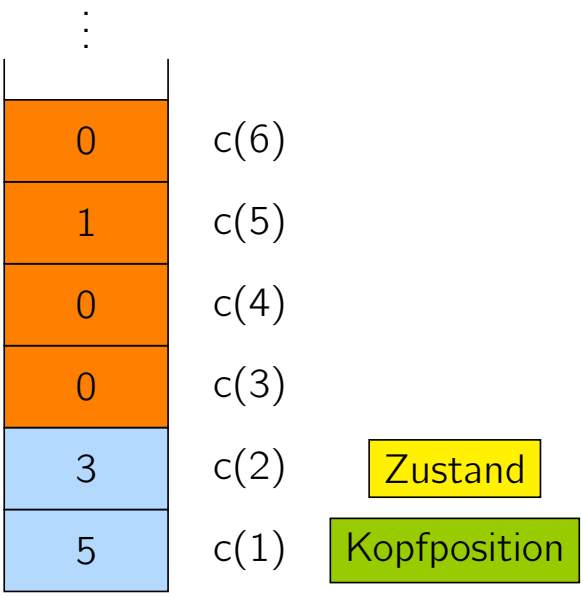


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

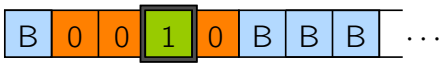
simulierende Registermaschine





Simulation TM durch RAM – Illustration

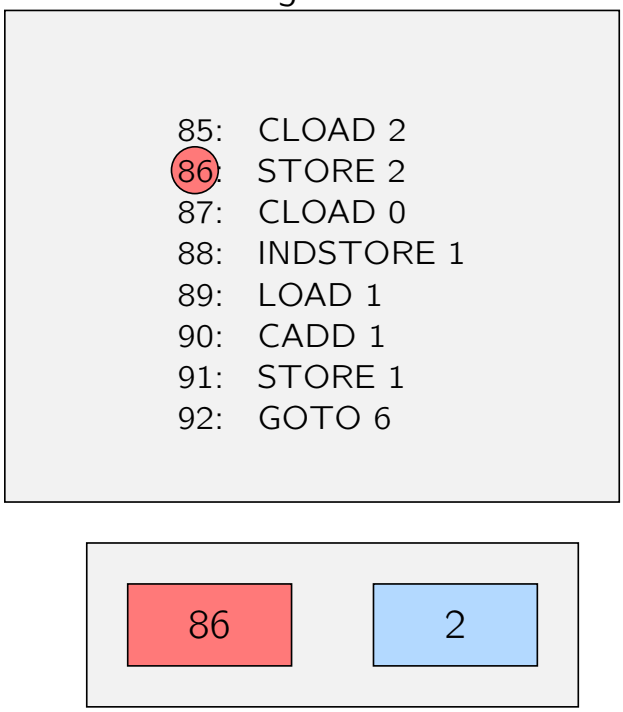
simulierte Turingmaschine M

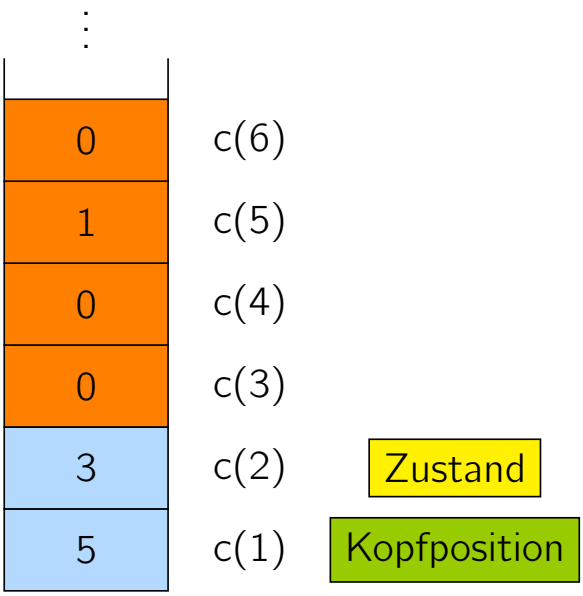


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

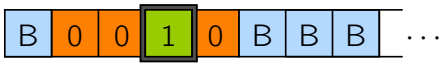
simulierende Registermaschine





Simulation TM durch RAM – Illustration

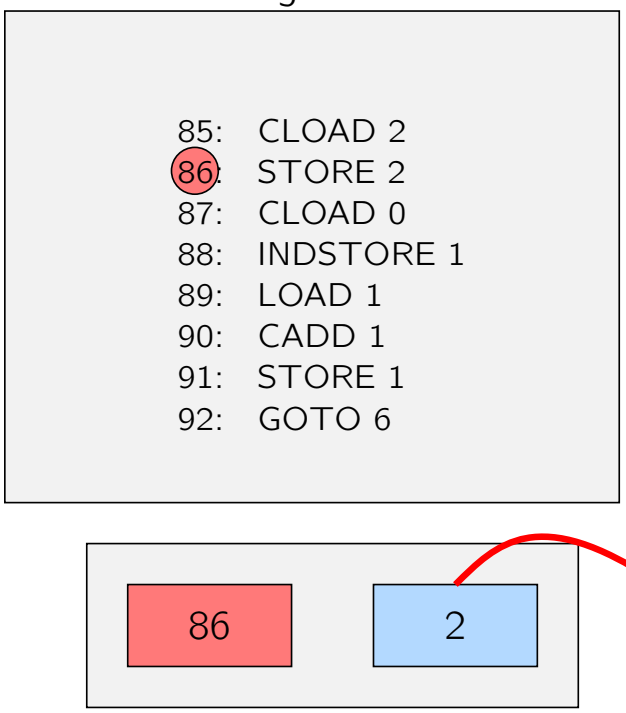
simulierte Turingmaschine M

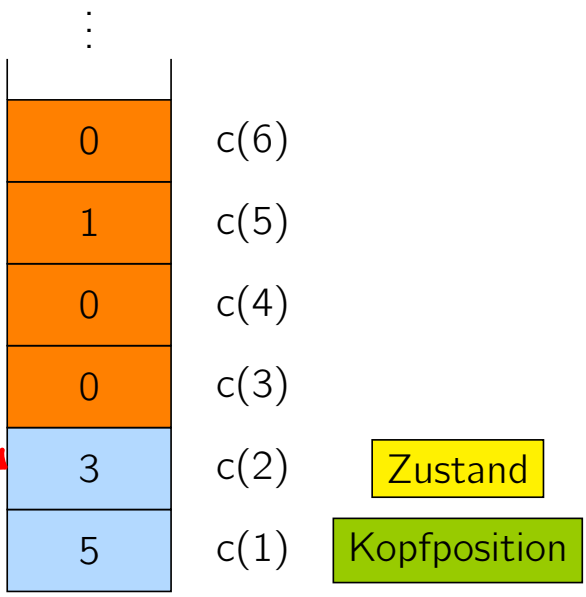


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

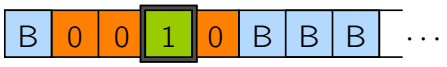
simulierende Registermaschine





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

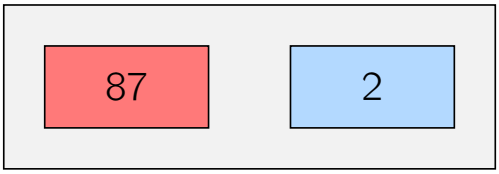


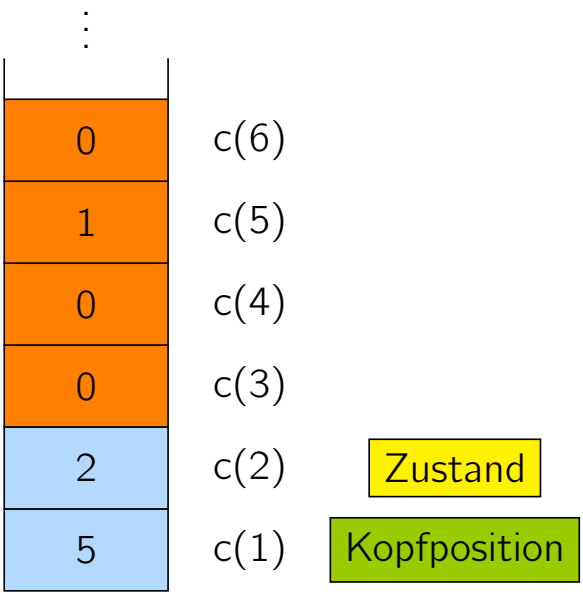
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

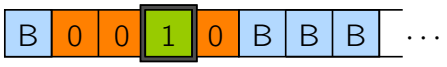
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

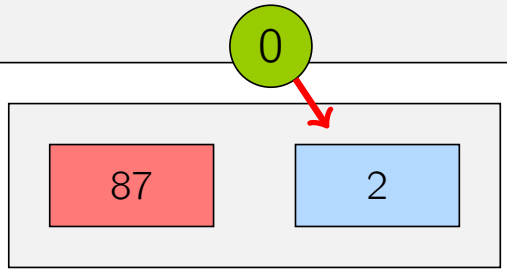


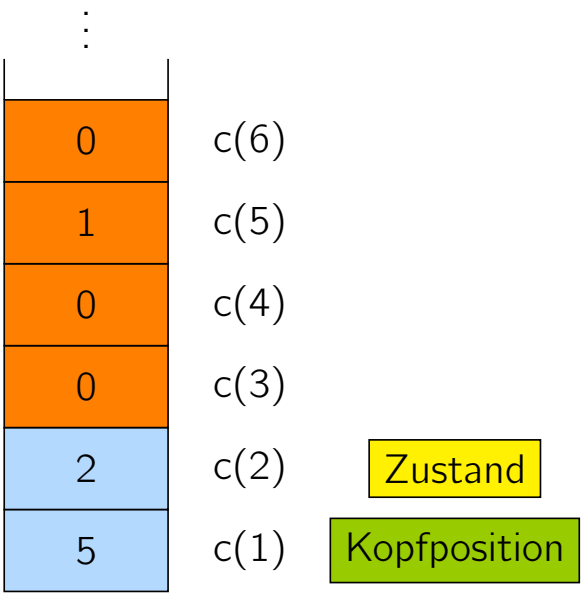
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

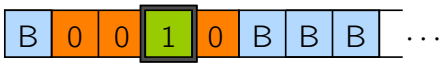
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

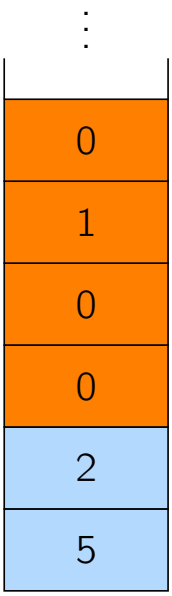


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

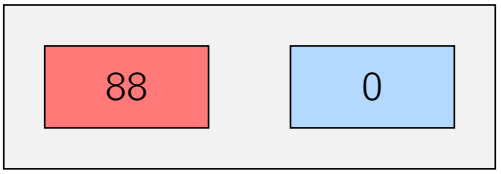
q_3

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
```



c(6) c(5) c(4) c(3) c(2) c(1)

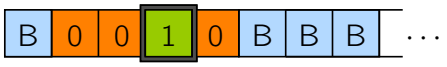


Zustand

Kopfposition

Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

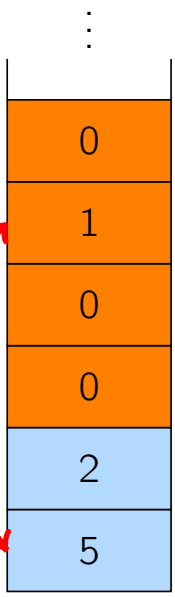


δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

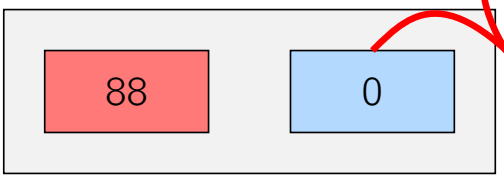
q_3

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
```



c(6) c(5) c(4) c(3) c(2) c(1)



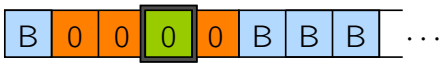
Zustand

Kopfposition

Vorlesung BuK im WS 22/23, M. Grohe Seite 109 Version 12. Oktober 2022

Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

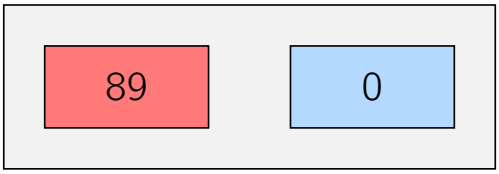


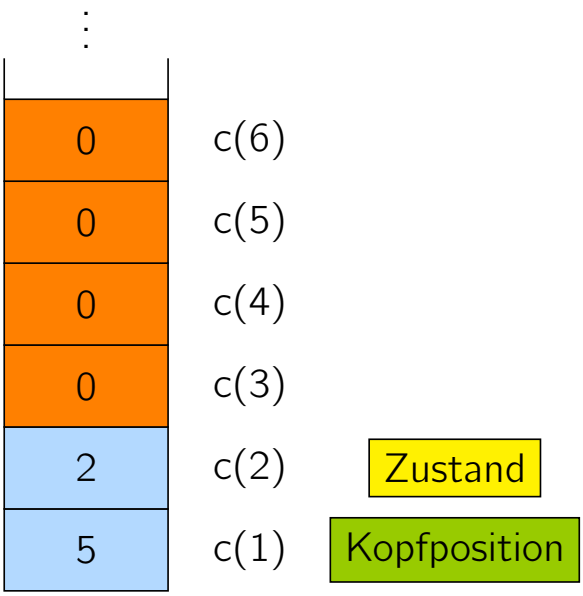
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

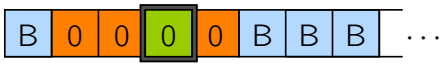
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

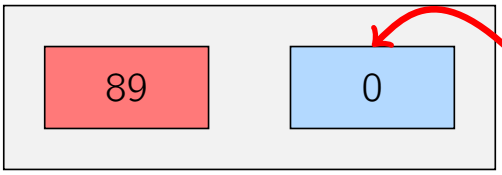


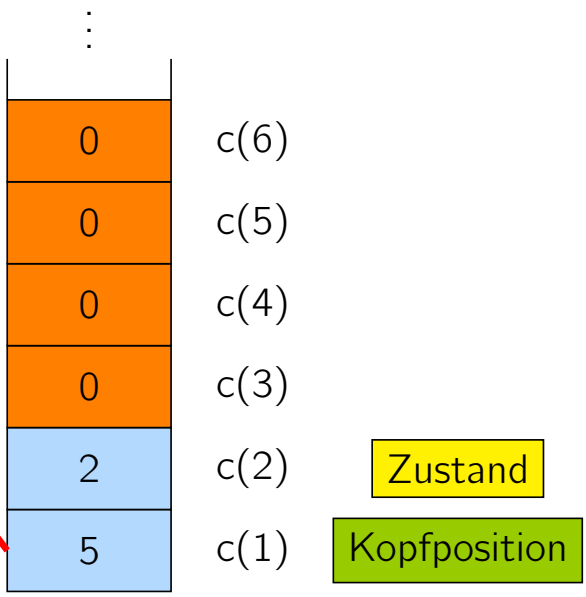
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

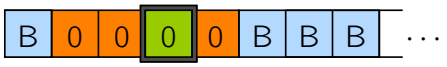
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

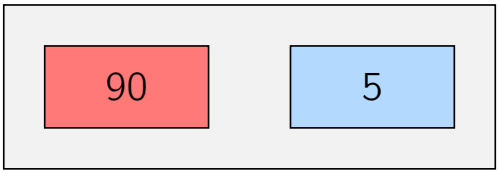


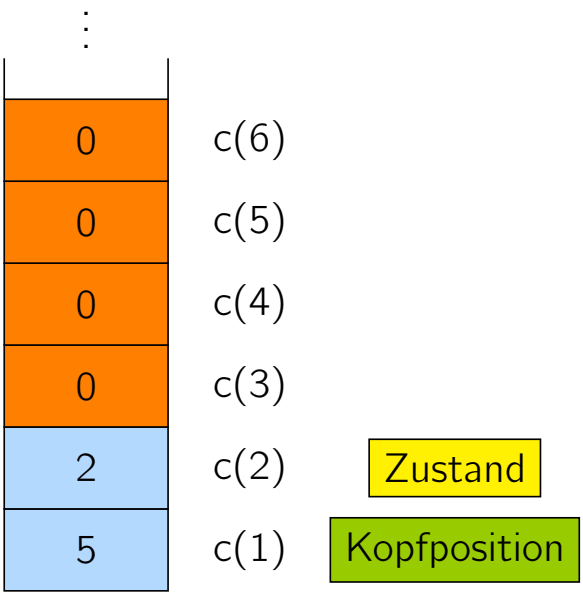
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

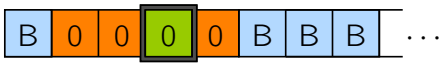
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

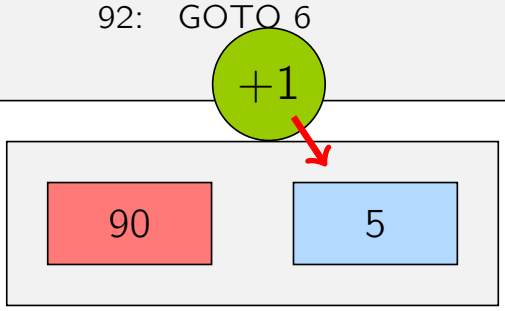


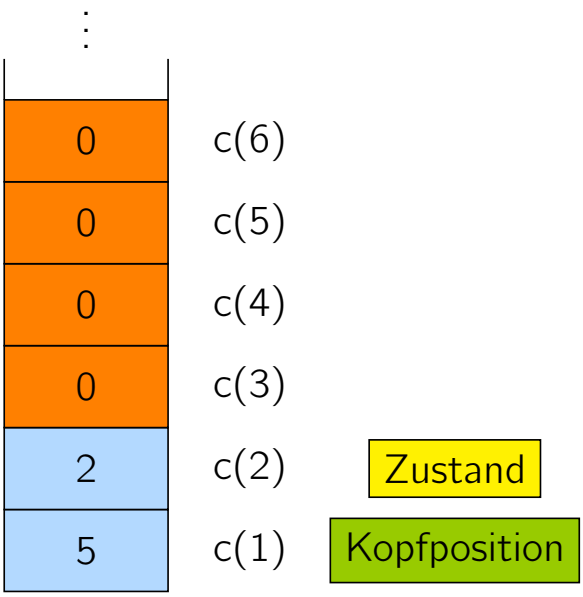
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

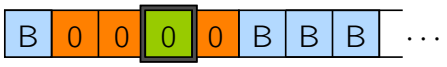
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

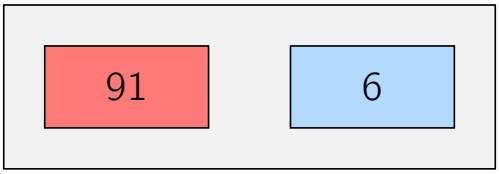


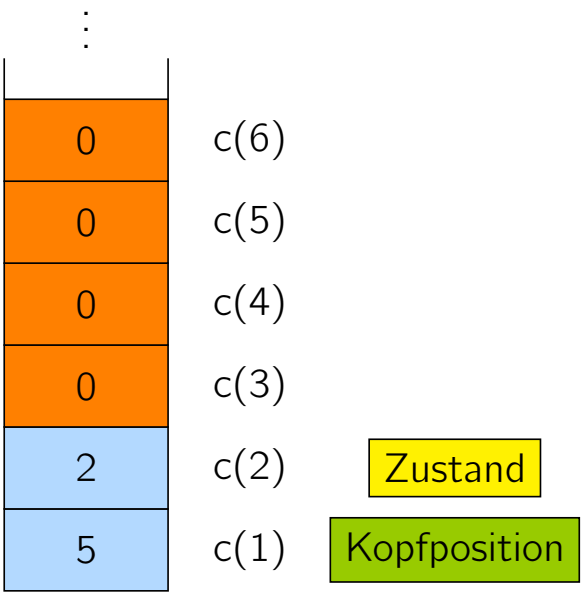
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

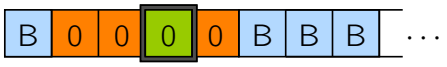
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

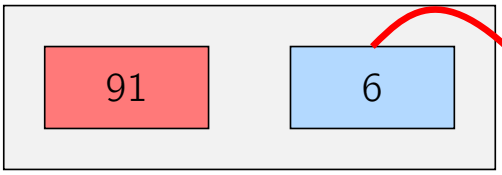


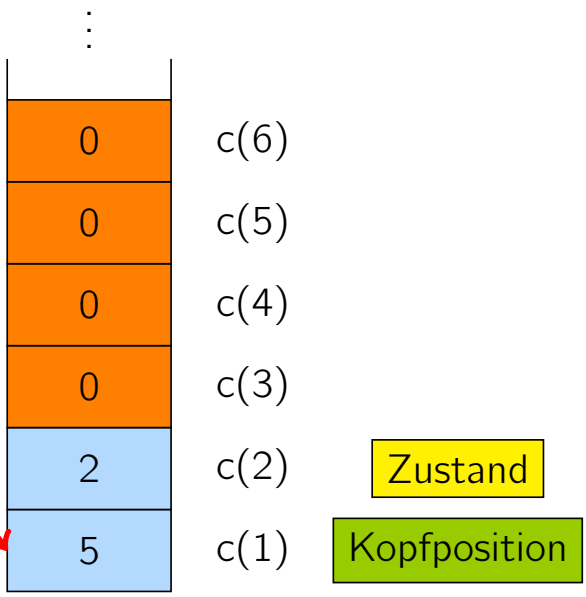
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

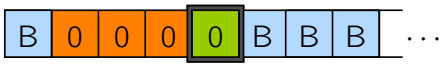
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

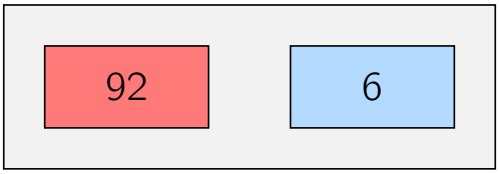


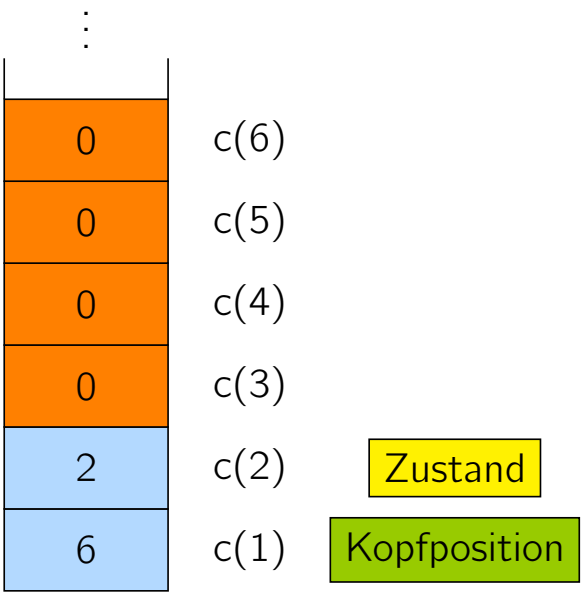
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_3

simulierende Registermaschine

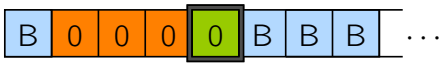
```
85:  CLOAD 2
86:  STORE 2
87:  CLOAD 0
88:  INDSTORE 1
89:  LOAD 1
90:  CADD 1
91:  STORE 1
92:  GOTO 6
```





Simulation TM durch RAM – Illustration

simulierte Turingmaschine M

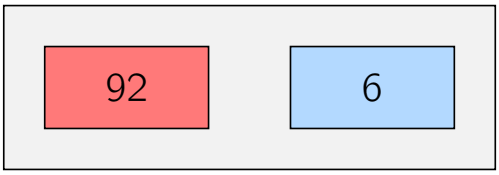


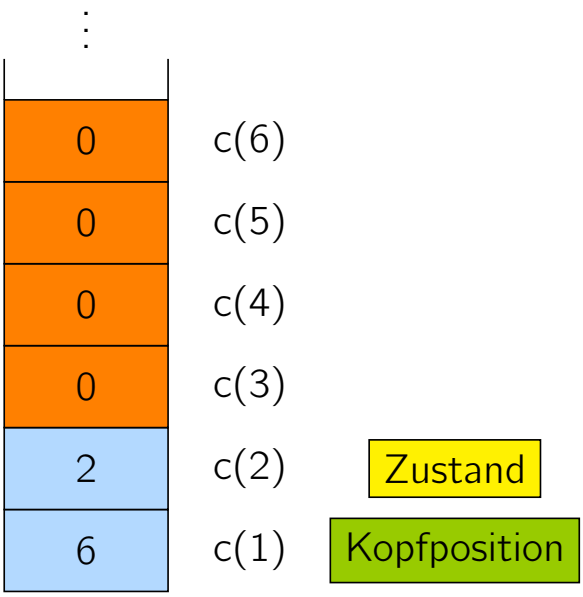
δ	0	1	B
q_1			
q_2			
q_3		$(q_2, 0, R)$	

q_2

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
```





Simulation TM durch RAM – Beweis

Laufzeitanalyse im uniformen Kostenmodell:

- Die Initialisierung kann in Zeit $O(n)$ durchgeführt werden.

Simulation TM durch RAM – Beweis

Laufzeitanalyse im uniformen Kostenmodell:

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

Simulation TM durch RAM – Beweis

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 $O(n + t(n))$.

Simulation TM durch RAM – Beweis

Laufzeitanalyse im logarithmischen Kostenmodell:

- Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen.

Simulation TM durch RAM – Beweis

Laufzeitanalyse im logarithmischen Kostenmodell:

- Die in den Registern gespeicherten Zahlen repräsentieren Zustände, Zeichen und Bandpositionen.
- Zustände und Zeichen haben eine konstante Kodierungslänge.

Simulation TM durch RAM – Beweis

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 $O(\log(t(n) + n))$.

Simulation TM durch RAM – Beweis

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.

Simulation TM durch RAM – Beweis

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 logarithmischen Kostenmodell) können sich gegenseitig mit polynomiellem Zeitverlust simulieren.

Zusammenfassung

- 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

Kein jemals bisher vorgeschlagenes „vernünftiges“ Rechnermodell ist mächtiger als die TM.

Die Church-Turing-These

Kein jemals bisher vorgeschlagenes „vernünftiges“ Rechnermodell ist mächtiger als die TM.

Diese Einsicht hat Church zur Formulierung der folgenden These veranlasst.

Church-Turing-These

Die Klassen der TM-berechenbaren Funktionen und TM-entscheidbaren Sprachen stimmen mit den Klassen der “intuitiv berechenbaren” Funktionen bzw. “intuitiv entscheidbaren” Sprachen überein.

Die Church-Turing-These

Kein jemals bisher vorgeschlagenes „vernünftiges“ Rechnermodell ist mächtiger als die TM.

Diese Einsicht hat Church zur Formulierung der folgenden These veranlasst.

Church-Turing-These

Die Klassen der TM-berechenbaren Funktionen und TM-entscheidbaren Sprachen stimmen mit den Klassen der “intuitiv berechenbaren” Funktionen bzw. “intuitiv entscheidbaren” Sprachen überein.

Wir werden deshalb nicht mehr von TM-berechenbaren Funktionen oder TM-entscheidbaren Sprachen sprechen, sondern allgemein von berechenbaren Funktionen bzw. entscheidbaren Sprachen.

Gleichbedeutend wird auch der Begriff rekursive Funktion bzw. Sprache verwendet.

Erweiterung: Partielle Funktionen

Wenn eine Turingmaschine bei einer Eingabe anhält, liefert sie eine wohldefinierte Ausgabe. Aber Turingmaschinen halten nicht immer an und liefern deswegen zu manchen Eingaben keine Ausgabe.

Erweiterung: Partielle Funktionen

Wenn eine Turingmaschine bei einer Eingabe anhält, liefert sie eine wohldefinierte Ausgabe. Aber Turingmaschinen halten nicht immer an und liefern deswegen zu manchen Eingaben keine Ausgabe.

Im allgemeinen berechnet eine Turingmaschine M mit Ein- und Ausgabealphabet Σ also nur eine partielle Funktion f_M von Σ^* nach Σ^*, die definiert ist durch

$$f_M(x) = \begin{cases} y & \text{falls } M \text{ bei Eingabe } x \text{ mit Ausgabe } y \text{ anhält,} \\ \perp \text{ (undefiniert)} & \text{falls } M \text{ bei Eingabe } x \text{ nicht anhält.} \end{cases}$$

Erweiterung: Partielle Funktionen

Wenn eine Turingmaschine bei einer Eingabe anhält, liefert sie eine wohldefinierte Ausgabe. Aber Turingmaschinen halten nicht immer an und liefern deswegen zu manchen Eingaben keine Ausgabe.

Im allgemeinen berechnet eine Turingmaschine M mit Ein- und Ausgabealphabet Σ also nur eine partielle Funktion f_M von Σ^* nach Σ^*, die definiert ist durch

$$f_M(x) = \begin{cases} y & \text{falls } M \text{ bei Eingabe } x \text{ mit Ausgabe } y \text{ anhält,} \\ \perp \text{ (undefiniert)} & \text{falls } M \text{ bei Eingabe } x \text{ nicht anhält.} \end{cases}$$

Wir nennen eine partielle Funktion f von Σ^* nach Σ^* berechenbar, wenn es eine Turingmaschine M gibt, so dass $f = f_M$.

Zum Inhalt der Vorlesung

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

In der *Berechenbarkeitstheorie* ...

... wird untersucht, welche (partiellen) Funktionen berechenbar und welche Sprachen entscheidbar sind, d.h. welche Berechnungsprobleme durch einen Algorithmus – ohne jegliche Einschränkungen der Rechenzeit oder des Speicherplatzes – gelöst werden können.