## Aufgabe 1

a) Es ist die Anzahl aller möglichen Kombinationen der Register $\alpha$, $\beta$ und $\eta$ sowie der Schalter $\omega_1$ und $\omega_2$ zu bilden:

$\alpha \in [-Q:+Q]$
$\beta \in W$, wobei $W = Op \times [0:K]$ und $K = \max(N,M)$
$\eta \in [0:N]$
$\omega_1, \omega_2 \in \{0,1\}$

$\#Z_s = \#[-Q:+Q] \cdot \#W \cdot \#[0:N] \cdot \#\{0,1\} \cdot \#\{0,1\}$
$\quad = (2Q+1) \cdot (\#Op \cdot \max(N,M)) \cdot (N+1) \cdot 2 \cdot 2$

b) Der Zustand des Rechners ist die Kombination der Zustände der Steuerwerks, des Programm- und Datenspeichers:

$Z_R = (\pi, \rho, z_s)$ wobei $\pi$ - Programmspeicherzustand
$\rho$ - Datenspeicherzustand
$z_s$ - Schaltwerkszustand

$\pi$: $\Pi[0:N] \rightarrow W$
$\#(\Pi[0:N] \rightarrow W) = (\#W)^{N+1}$
$\quad = (\#Op \cdot (\max(N,M)+1))^{N+1}$

$\rho$: $P[0:M] \rightarrow [-Q:+Q]$
$\#( P[0:M] \rightarrow [-Q:+Q]) = (2Q+1)^{M+1}$

Somit ist $Z_M = \pi \times \rho \times z_s$, was selbst bei wenig Speicher und Befehlen zu sehr schnell wachsenden Zahlen führt, so dass es schon unmöglich ist, nur für kleine Computer alle Zustände anzugeben.

c) Zwar wird die Anzahl der Zustände bei einem Computer enorm groß, jedoch haben alle Einflussgrößen einen endlichen Wertebereich. Somit ist es zumindest theoretisch möglich, alle Zustände aufzuzählen und auch alle Zustandsänderungen zu erfassen.

## Aufgabe 2

$k = (\pi, \rho, \alpha, \beta, \gamma, \eta, \omega_1, \omega_2)$

a) `do α:=α-ρ(i)`
für $0 <= i <= M$ und $0 <= \eta <= N$ und $\alpha\text{-}\rho(i) \in [-Q:+Q]$:
$k'_1 = (\pi, \rho, \alpha'=\alpha-\rho(i), \beta'=\pi(\eta'), \gamma, \eta'=\eta+1, \omega_1'=1, \omega_2'=0)$
sonst:
$k_2' = (\pi, \rho, \alpha, \beta, \gamma, \eta, \omega_1'=1, \omega_2'=1)$

Da ich im folgenden nicht extra alle Sonderfälle bzw. mögliche Fehler derart ausführlich beschreiben möchte, unterstriche ich sie, was heißt, dass diese Register den Wertebereich eventuell verlassen und somit $\omega_2'=1$ setzen

b) `do α:=α+1`
$k' = (\pi, \rho, \underline{\alpha'=\alpha+1}, \beta'=\pi(\eta'), \gamma, \underline{\eta'=\eta+1}, \omega_1'=1, \omega_2'=0)$

c) `if α>0 goto j`
falls Bedingung $\alpha>0$ erfüllt ist:
$k_1' = (\pi, \rho, \alpha, \underline{\beta'=\pi(j)}, \gamma, \underline{\eta'=j}, \omega_1'=1, \omega_2'=0)$
falls Bedingung $\alpha>0$ nicht erfüllt ist:
$k_2' = (\pi, \rho, \alpha, \beta'=\pi(\eta'), \gamma, \underline{\eta'=\eta+1}, \omega_1'=1, \omega_2'=0)$

d) `do α:=ρ(i)/α`
$k' = (\pi, \rho, \underline{\alpha'=\rho(i)/\alpha}, \beta'=\pi(\eta'), \gamma, \underline{\eta'=\eta+1}, \omega_1'=1, \omega_2'=0)$

Insbesondere ist bei der Division auf a≠0 zu achten. Außerdem dürfen Rundungsfehler (z.B. Division nur für ganze Zahlen definiert) nicht vergessen werden.

e) `do π(i):=α`
$k' = (\pi'(j)= \pi(j)$ für $i\neq j$ bzw. $\alpha$ für $i=j, \rho, \alpha, \beta'=\pi(\eta'), \gamma, \underline{\eta'=\eta+1}, \omega_1'=1, \omega_2'=0)$

Es muss noch $0 <= i <= M$ gelten.

f) `do α:=ρ(ρ(γ+i))`
$k' = (\pi, \rho, \underline{\alpha'=\rho(\rho(\gamma+i))}, \beta'=\pi(\eta'), \underline{\gamma'=\gamma+i}, \underline{\eta'=\eta+1}, \omega_1'=1, \omega_2'=0)$

Beim indizierten Zugriff sind zwei Fehlerquellen möglich, es muss daher eingeschränkt werden:
$0 <= \gamma+i <= M$ und $0 <=\rho(\gamma+i) <= M$
Ich bin mir aufgrund fehlenden Detailwissens dieser Maschine nicht sicher, ob $\gamma$ zweimal manipuliert wird. Neben der oben angegebenen Gleichung $\gamma'=\gamma+i$ *könnte* noch der Seiteneffekt $\gamma''=\rho(\gamma')=\rho(\gamma+i)$ auftreten.

g) do $\alpha:=\rho(\gamma+\rho(i))$
$k' = (\pi, \rho, \underline{\alpha'=\rho(\gamma+\rho(i))}, \beta'=\pi(\eta'), \underline{\gamma'=\gamma+\rho(i)}, \underline{\eta'=\eta+1}, \omega_1'=1, \omega_2'=0)$

Auch hier muss ergänzt werden:
$0 <= \gamma+\rho(i) <= M$ und $0 <= i <= M$

**<u>Aufgabe 3</u>**

Als Algorithmus nutze ich die Definition der Multiplikation in N:
I. $m \cdot 1 = m$
II. $(m+1) \cdot n = m \cdot n + n$
Im Endeffekt bedeutet dies, dass m genau n-mal aufaddiert wird.

a)
```
0: begin: goto 1;
1: do α:=ρ(1);
2: if α=0 goto 9;
3: do α:=α+ρ(3);
4: do ρ(1):=α;
5: do α:=ρ(2);
6: do α:=α+ρ(0);
7: do ρ(2):=α;
8: goto 1;
9: end.
```

Das einzige für mich nicht lösbare Problem ist, dass ρ(2) mit 0 initialisiert sein muss (gute Datenspeicher machen das alleine ...).

b)
```
0,1,2: anfang: if n=0 then goto ende;
3,4:   n=n-1;
5,6,7: p=p+m;
8:     goto anfang;
9:     ende: stop.
```

Wobei die Variablen der Bedeutung $p=m\cdot n$ folgen. Wiederum müsste auch p mit 0 initialisiert werden.

## Aufgabe 4

a) Das Indexregister ermöglicht den Zugriff auf ein Datum, dessen Adresse erst zur Laufzeit ermittelt wird. Sehr sinnvoll ist dies z.B. bei der Bearbeitung eines Feldes in einer Schleife, wobei auf jedes Feldelement die gleiche Operation (d.h. Befehlsfolge) angewendet wird. Zwar kann man das auch über den Akkumulator erreichen, jedoch benötigt man dann viel Code zum „Puffern" einzelner Zwischenergebnisse (da man sich das Indexregister quasi im Datenspeicher schafft). Zusätzlich ermöglicht das Indexregister noch viele Adressberechnungen, z.B. durch diverse Offset-Einflussmöglichkeiten.

b) Es kann vorkommen, dass in einem Befehlstakt sowohl auf $\alpha$ als auch auf $\gamma$ zugegriffen werden muss, z.B. als Seiteneffekt von $\alpha' = \alpha + \rho(\gamma+3)$ ist $\gamma' = \gamma + 3$.

c) Die Anzahl der Zustände $Z_{S+\gamma}$ ist:

$$\begin{aligned} \#Z_{S+\gamma} &= \#Z_S \cdot \#\gamma \\ &= \#Z_S \cdot \#[0:M] \\ &= \#Z_S \cdot (M+1) \end{aligned}$$

Lässt man Zugriffe auf den Programmspeicher auch zu, so ist:

$$\#Z_{S+\gamma} = \#Z_S \cdot (K+1)$$

(siehe auch Aufgabe 1)