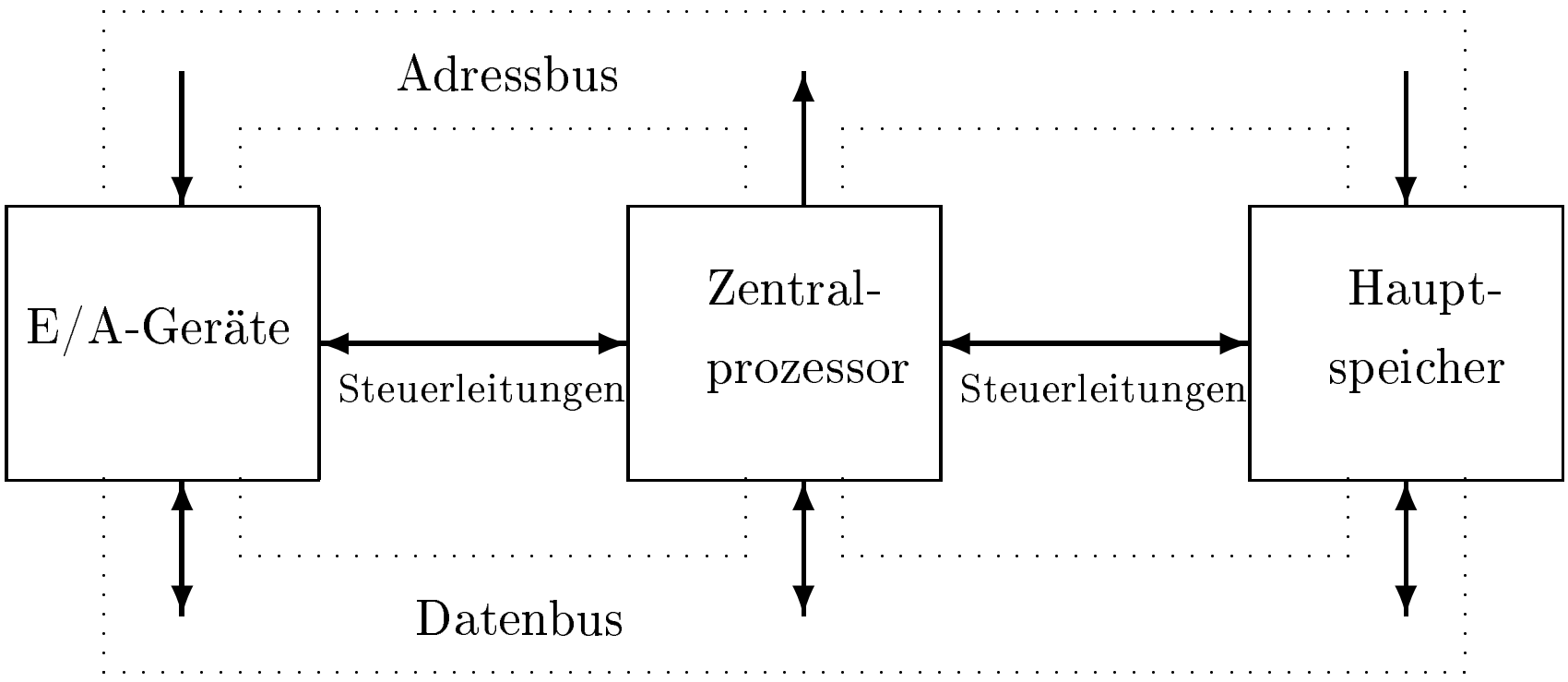
Informatik II für Verkehrsingenieure
AM_0 (Kapitel 14.1 + 14.2)

Janis Voigtländer

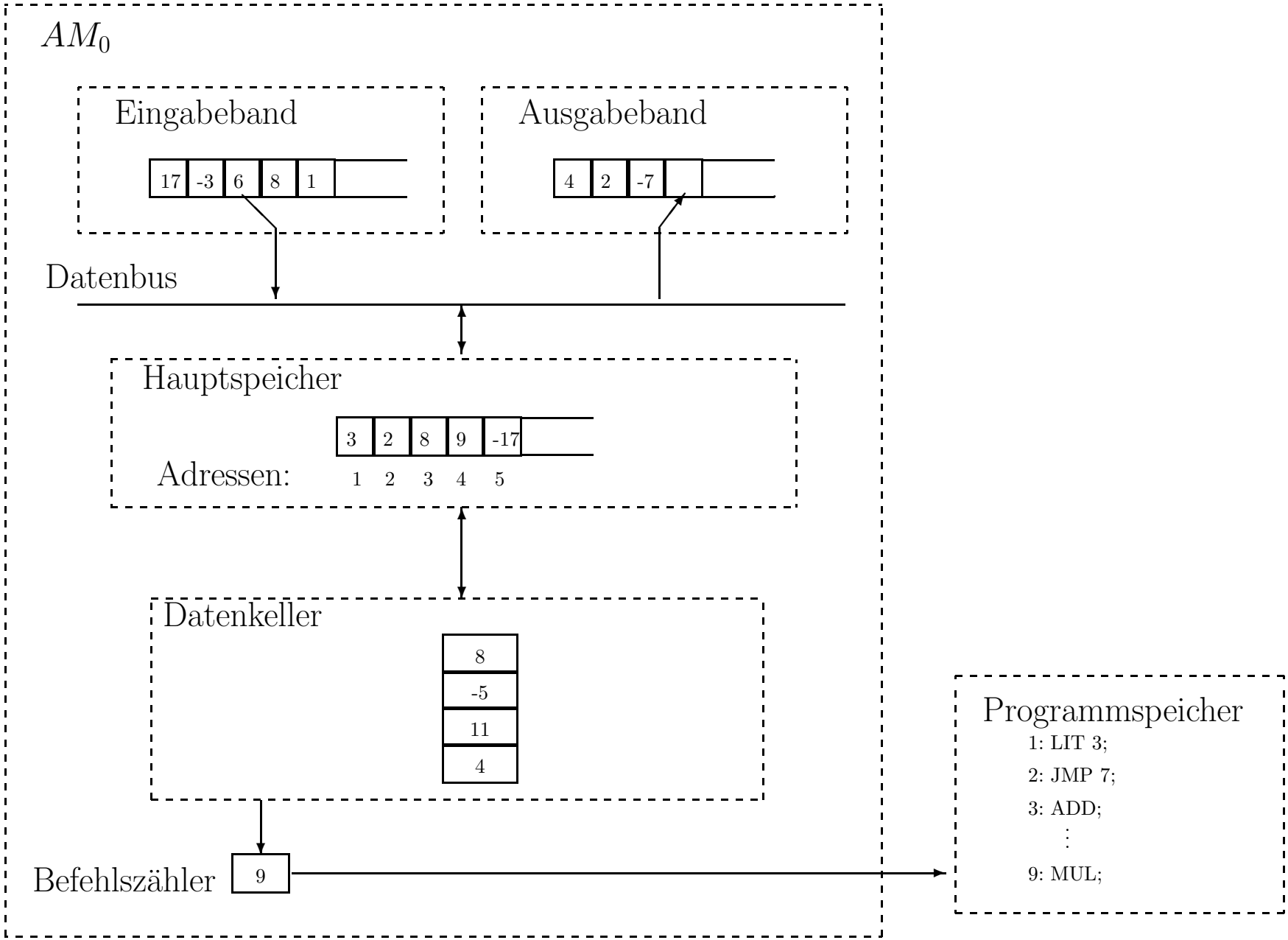
Technische Universität Dresden

Sommersemester 2007

Von-Neumann-Rechner



Abstrakte Maschine AM_0



Modellierung von C-Features

```
int main()
{ int i,n,s;
  scanf("%i",&n);
  i=1;
  s=0;
  while (i<=n)
   { s=s+i*i;
     i=i+1;
   }
  printf("%d",s);
  return 0;
}
```

Modellierung von C-Features

```
int main()
{ int i,n,s;
  scanf("%i",&n);
  i=1;
  s=0;
  while (i<=n)
   { s=s+i*i;
     i=i+1;
   }
  printf("%d",s);
  return 0;
}
```

Wir brauchen: Speicherplätze

Modellierung von C-Features

```
int main()
{ int i,n,s;
  scanf("%i",&n);
  i=1;
  s=0;
  while (i<=n)
   { s=s+i*i;
     i=i+1;
   }
  printf("%d",s);
  return 0;
}
```

Wir brauchen: Speicherplätze, Ein- und Ausgabe

Modellierung von C-Features

```
int main()
{ int i,n,s;
  scanf("%i",&n);
  i=1;
  s=0;
  while (i<=n)
   { s=s+i*i;
     i=i+1;
   }
  printf("%d",s);
  return 0;
}
```

Wir brauchen: Speicherplätze, Ein- und Ausgabe, Auswertung von Ausdrücken

Modellierung von C-Features

```
int main()
{ int i,n,s;
  scanf("%i",&n);
  i=1;
  s=0;
  while (i<=n)
   { s=s+i*i;
     i=i+1;
   }
  printf("%d",s);
  return 0;
}
```

Wir brauchen: Speicherplätze, Ein- und Ausgabe, Auswertung von Ausdrücken, Kontrollfluss

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte
- mathematisch: $\mathrm{HS} = \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte
- mathematisch: $\mathrm{HS} = \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$
- Notation: $h = [1/3,2/2,3/5]$

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte
- mathematisch: $\mathrm{HS} = \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$
- Notation: $h = [1/3,2/2,3/5]$
- Update: $h[n/d]$

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte
- mathematisch: $\mathrm{HS} = \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$
- Notation: $h = [1/3, 2/2, 3/5]$
- Update: $h[n/d]$;
  $h[n/d](n') = d$, falls $n' = n$, sonst $h[n/d](n') = h(n')$

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte
- mathematisch: $\mathrm{HS} = \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$
- Notation: $h = [1/3,2/2,3/5]$
- Update: $h[n/d]$;
  $h[n/d](n') = d$, falls $n' = n$, sonst $h[n/d](n') = h(n')$
- Beispiele für $h = [1/3,2/2,3/5]$:
  $h[5/7] = [1/3,2/2,3/5,5/7]$

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte
- mathematisch: $\mathrm{HS} = \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$
- Notation: $h = [1/3,2/2,3/5]$
- Update: $h[n/d]$;
  $h[n/d](n') = d$, falls $n' = n$, sonst $h[n/d](n') = h(n')$
- Beispiele für $h = [1/3,2/2,3/5]$:
  $h[5/7] = [1/3,2/2,3/5,5/7]$ und
  $h[2/7] = [1/3,2/7,3/5]$

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte
- mathematisch: $\mathrm{HS} = \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$
- Notation: $h = [1/3,2/2,3/5]$
- Update: $h[n/d]$;
  $h[n/d](n') = d$, falls $n' = n$, sonst $h[n/d](n') = h(n')$
- Beispiele für $h = [1/3,2/2,3/5]$:
  $h[5/7] = [1/3,2/2,3/5,5/7]$ und
  $h[2/7] = [1/3,2/7,3/5]$
- Befehle zur Kommunikation mit:
  - Ein- und Ausgabe

Modellierung des Hauptspeichers

- statt benannter Speicherplätze (Variablen) lediglich numerische Adressen
- einzig möglicher Speicherinhalt: ganze Zahl (`int`)
- konkreter Hauptspeicherinhalt modelliert als partielle Abbildung von Adressen auf Inhalte
- mathematisch: $\mathrm{HS} = \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$
- Notation: $h = [1/3,2/2,3/5]$
- Update: $h[n/d]$;
  $h[n/d](n') = d$, falls $n' = n$, sonst $h[n/d](n') = h(n')$
- Beispiele für $h = [1/3,2/2,3/5]$:
  $h[5/7] = [1/3,2/2,3/5,5/7]$ und
  $h[2/7] = [1/3,2/7,3/5]$
- Befehle zur Kommunikation mit:
  - Ein- und Ausgabe
  - „Berechnungseinheit“

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe; stattdessen: Eingabeband und Ausgabeband

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe; stattdessen: Eingabeband und Ausgabeband
- Bandinhalte modelliert als endliche Listen

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe; stattdessen: Eingabeband und Ausgabeband
- Bandinhalte modelliert als endliche Listen
- mathematisch: $\underline{\mathsf{Inp}} = \underline{\mathsf{Out}} = \mathbb{Z}^*$

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe; stattdessen: Eingabeband und Ausgabeband
- Bandinhalte modelliert als endliche Listen
- mathematisch: $\underline{\mathsf{Inp}} = \underline{\mathsf{Out}} = \mathbb{Z}^*$
- Notation: $\mathit{inp} = 1.13.5$ oder $\mathit{out} = \varepsilon$

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe; stattdessen: Eingabeband und Ausgabeband
- Bandinhalte modelliert als endliche Listen
- mathematisch: $\underline{\mathsf{Inp}} = \underline{\mathsf{Out}} = \mathbb{Z}^*$
- Notation: $\mathit{inp} = 1.13.5$ oder $\mathit{out} = \varepsilon$
- READ n: Einlesen erster Position von inp

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe;
  stattdessen: Eingabeband und Ausgabeband
- Bandinhalte modelliert als endliche Listen
- mathematisch: $\underline{\mathsf{Inp}} = \underline{\mathsf{Out}} = \mathbb{Z}^*$
- Notation: $\mathit{inp} = 1.13.5$ oder $\mathit{out} = \varepsilon$
- READ n: Einlesen erster Position von inp und
  entsprechendes Update der Position n des HS

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe;
  stattdessen: Eingabeband und Ausgabeband
- Bandinhalte modelliert als endliche Listen
- mathematisch: $\underline{\text{Inp}} = \underline{\text{Out}} = \mathbb{Z}^*$
- Notation: $inp = 1.13.5$ oder $out = \varepsilon$
- READ n: Einlesen erster Position von inp und
  entsprechendes Update der Position n des HS
- WRITE n: Ausgabe des HS-Inhalts an Position n

Modellierung von Ein- und Ausgabe

- lediglich Ein- und Ausgabe ganzer Zahlen (`int`)
- Abstraktion von interaktiver Ein- und Ausgabe;
  stattdessen: Eingabeband und Ausgabeband
- Bandinhalte modelliert als endliche Listen
- mathematisch: $\underline{\text{Inp}} = \underline{\text{Out}} = \mathbb{Z}^*$
- Notation: $inp = 1.13.5$ oder $out = \varepsilon$
- READ n: Einlesen erster Position von inp und
  entsprechendes Update der Position n des HS
- WRITE n: Ausgabe des HS-Inhalts an Position n
  am Ende von out

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Lösung:

- Einführung eines „Datenkellers“

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Lösung:

- Einführung eines „Datenkellers“
- Ablage und Entnahme jeweils nur an Kellerspitze

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Lösung:

- Einführung eines „Datenkellers“
- Ablage und Entnahme jeweils nur an Kellerspitze
- mathematisch: DK $= \mathbb{Z}^*$

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Lösung:

- Einführung eines „Datenkellers“
- Ablage und Entnahme jeweils nur an Kellerspitze
- mathematisch: DK $= \mathbb{Z}^*$
- Notation: $d = 8{:}7{:}2$

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Lösung:

- Einführung eines „Datenkellers“
- Ablage und Entnahme jeweils nur an Kellerspitze
- mathematisch: DK $= \mathbb{Z}^*$
- Notation: $d = 8{:}7{:}2$
- LIT z: Ablage einer Konstante

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Lösung:

- Einführung eines „Datenkellers“
- Ablage und Entnahme jeweils nur an Kellerspitze
- mathematisch: DK $= \mathbb{Z}^*$
- Notation: $d = 8{:}7{:}2$
- LIT z: Ablage einer Konstante
- LOAD n: Ablage des HS-Inhalts an Position n

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Lösung:

- Einführung eines „Datenkellers“
- Ablage und Entnahme jeweils nur an Kellerspitze
- mathematisch: DK $= \mathbb{Z}^*$
- Notation: $d = 8{:}7{:}2$
- LIT z: Ablage einer Konstante
- LOAD n: Ablage des HS-Inhalts an Position n
- ADD, MUL, . . . , EQ, NE ,LT, . . . : Berechnungen

Auswertung von Ausdrücken

Problem:

- Wir wollen strukturierte Ausdrücke wie `(3+5)*(7-2)` oder `j>2*i+1` auswerten.
- Dies soll jedoch in „linearisierter Form“ erfolgen, insbesondere eine Operation immer nur auf zwei Operanden wirken.
- Folglich müssen Zwischenergebnisse berechnet und in geeigneter Weise vorrätig gehalten werden.

Lösung:

- Einführung eines „Datenkellers“
- Ablage und Entnahme jeweils nur an Kellerspitze
- mathematisch: DK $= \mathbb{Z}^*$
- Notation: $d = 8{:}7{:}2$
- LIT z: Ablage einer Konstante
- LOAD n: Ablage des HS-Inhalts an Position n
- ADD, MUL, . . . , EQ, NE ,LT, . . . : Berechnungen
- STORE n: Entnahme und entsprechendes Update der Position n des HS

Kontrollfluss — Abarbeitungsreihenfolge

- Durchnummerierung aller Befehle in einem Programm

Kontrollfluss — Abarbeitungsreihenfolge

- Durchnummerierung aller Befehle in einem Programm
- aktuelle Position im Befehlszähler gespeichert ($m \in \mathrm{BZ} = \mathbb{N}$)

Kontrollfluss — Abarbeitungsreihenfolge

- Durchnummerierung aller Befehle in einem Programm
- aktuelle Position im Befehlszähler gespeichert ($m \in \mathrm{BZ} = \mathbb{N}$)
- normalerweise einfache Erhöhung nach jeder Befehlsabarbeitung

Kontrollfluss — Abarbeitungsreihenfolge

- Durchnummerierung aller Befehle in einem Programm
- aktuelle Position im Befehlszähler gespeichert ($m \in \mathrm{BZ} = \mathbb{N}$)
- normalerweise einfache Erhöhung nach jeder Befehlsabarbeitung
- JMP n: Sprung an Position n

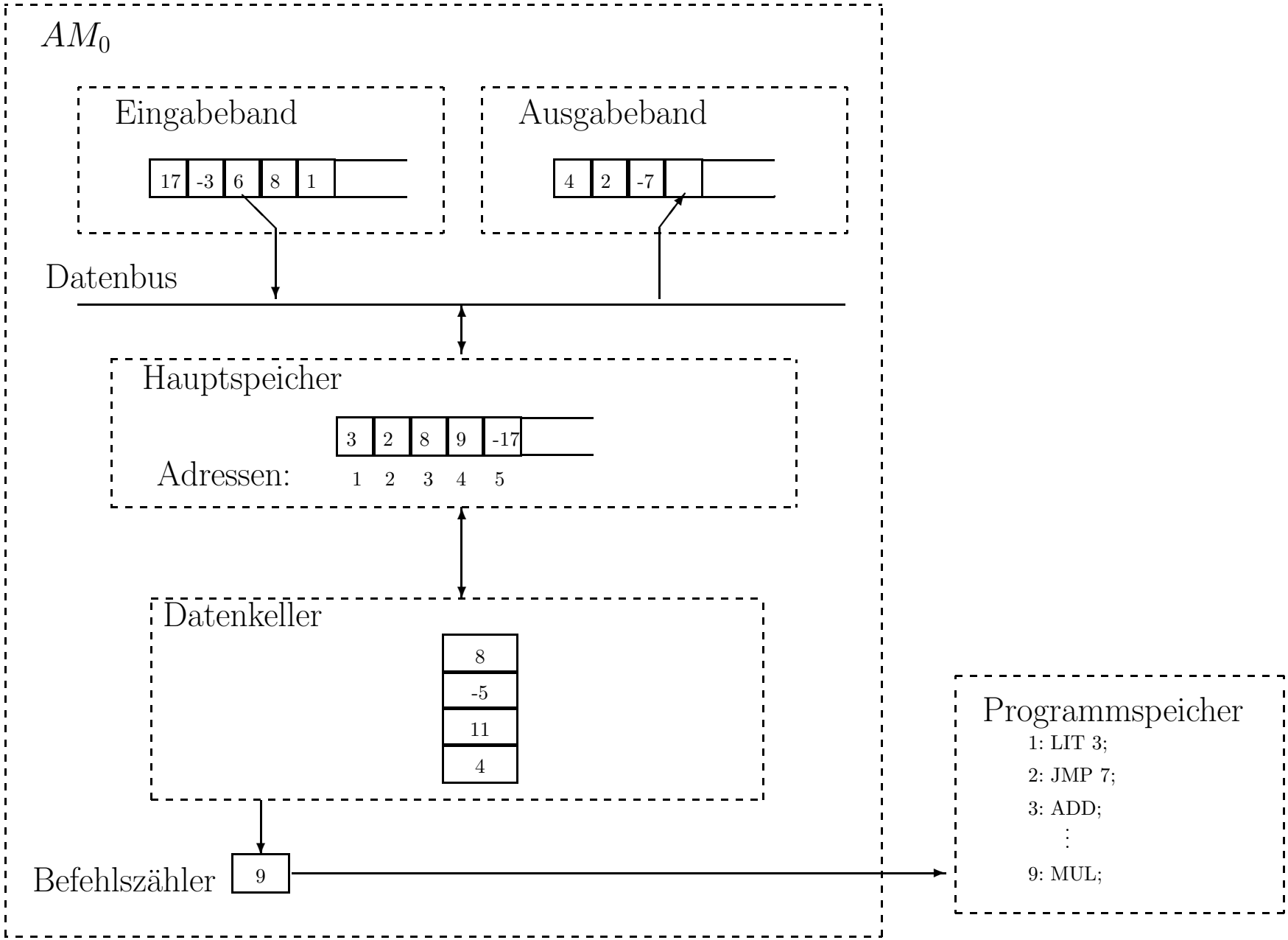
Kontrollfluss — Abarbeitungsreihenfolge

- Durchnummerierung aller Befehle in einem Programm
- aktuelle Position im Befehlszähler gespeichert ($m \in \mathrm{BZ} = \mathbb{N}$)
- normalerweise einfache Erhöhung nach jeder Befehlsabarbeitung
- JMP n: Sprung an Position n
- JMC n: bedingter Sprung an Position n

Kontrollfluss — Abarbeitungsreihenfolge

- Durchnummerierung aller Befehle in einem Programm
- aktuelle Position im Befehlszähler gespeichert ($m \in \mathrm{BZ} = \mathbb{N}$)
- normalerweise einfache Erhöhung nach jeder Befehlsabarbeitung
- JMP n: Sprung an Position n
- JMC n: bedingter Sprung an Position n;
  nur wenn Spitze des Datenkellers gleich 0

Abstrakte Maschine AM_0



Formale Modellierung

- $\mathsf{AM}_0 = \mathsf{BZ} \times \mathsf{DK} \times \mathsf{HS} \times \underline{\mathsf{Inp}} \times \underline{\mathsf{Out}}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{\mathsf{Inp}}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{\mathsf{Out}}$	$= \mathbb{Z}^*$	Ausgabeband

Formale Modellierung

- $\mathsf{AM}_0 = \mathsf{BZ} \times \mathsf{DK} \times \mathsf{HS} \times \underline{\mathsf{Inp}} \times \underline{\mathsf{Out}}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{\mathsf{Inp}}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{\mathsf{Out}}$	$= \mathbb{Z}^*$	Ausgabeband

- Maschine ist jederzeit in einem Zustand $s = (m, d, h, inp, out) \in \mathsf{AM}_0$

Formale Modellierung

- $\mathrm{AM}_0 = \mathrm{BZ} \times \mathrm{DK} \times \mathrm{HS} \times \underline{\mathrm{Inp}} \times \underline{\mathrm{Out}}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{\mathrm{Inp}}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{\mathrm{Out}}$	$= \mathbb{Z}^*$	Ausgabeband

- Maschine ist jederzeit in einem Zustand $s = (m, d, h, inp, out) \in \mathrm{AM}_0$
- Ein Programm ist eine partielle, endlich definierte Abbildung P von $\mathbb{N}$ auf die Menge Γ aller Befehle.

Formale Modellierung

- $\mathrm{AM}_0 = \mathrm{BZ} \times \mathrm{DK} \times \mathrm{HS} \times \underline{\mathrm{Inp}} \times \underline{\mathrm{Out}}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{\mathrm{Inp}}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{\mathrm{Out}}$	$= \mathbb{Z}^*$	Ausgabeband

- Maschine ist jederzeit in einem Zustand $s = (m, d, h, inp, out) \in \mathrm{AM}_0$
- Ein Programm ist eine partielle, endlich definierte Abbildung P von $\mathbb{N}$ auf die Menge Γ aller Befehle.
- Notation: 1: P(1);
  2: P(2);
  3: P(3);
  ...

Formale Modellierung

- $\mathsf{AM}_0 = \mathsf{BZ} \times \mathsf{DK} \times \mathsf{HS} \times \underline{\mathsf{Inp}} \times \underline{\mathsf{Out}}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{\mathsf{Inp}}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{\mathsf{Out}}$	$= \mathbb{Z}^*$	Ausgabeband

- Maschine ist jederzeit in einem Zustand $s = (m, d, h, inp, out) \in \mathsf{AM}_0$
- Ein Programm ist eine partielle, endlich definierte Abbildung P von $\mathbb{N}$ auf die Menge Γ aller Befehle.
- Notation: 1: P(1);
  2: P(2);
  3: P(3);
  ...
- Programmabarbeitung besteht aus wiederholter Anwendung von Befehlen auf einen Zustand.

Befehlssemantik (I)

$\mathcal{C}[\![.]\!]: \Gamma \longrightarrow (\mathsf{AM}_0 \rightharpoonup \mathsf{AM}_0)$

Befehlssemantik (I)

$\mathcal{C}[\![.]\!]: \Gamma \longrightarrow (\mathsf{AM}_0 \dashrightarrow \mathsf{AM}_0)$

$\mathcal{C}[\![\mathsf{READ}\ n]\!](m, d, h, inp, out) =$
wenn $inp = first(inp).rest(inp)$ mit $first(inp) \in \mathbb{Z}, rest(inp) \in \mathbb{Z}^*$,
dann $(m + 1, d, h[n/first(inp)], rest(inp), out)$

Befehlssemantik (I)

$\mathcal{C}[\![.]\!]: \Gamma \longrightarrow (\mathsf{AM}_0 \dashrightarrow \mathsf{AM}_0)$

$\mathcal{C}[\![\text{READ } n]\!](m, d, h, inp, out) =$
wenn $inp = first(inp).rest(inp)$ mit $first(inp) \in \mathbb{Z}, rest(inp) \in \mathbb{Z}^*$,
dann $(m+1, d, h[n/first(inp)], rest(inp), out)$

$\mathcal{C}[\![\text{WRITE } n]\!](m, d, h, inp, out) =$
wenn $h(n) \in \mathbb{Z}$, dann $(m+1, d, h, inp, out.h(n))$

Befehlssemantik (I)

$\mathcal{C}[\![.]\!]\colon \Gamma \longrightarrow (\mathsf{AM}_0 \dashrightarrow \mathsf{AM}_0)$

$\mathcal{C}[\![\text{READ } n]\!](m, d, h, inp, out) =$
wenn $inp = first(inp).rest(inp)$ mit $first(inp) \in \mathbb{Z}, rest(inp) \in \mathbb{Z}^*$,
dann $(m+1, d, h[n/first(inp)], rest(inp), out)$

$\mathcal{C}[\![\text{WRITE } n]\!](m, d, h, inp, out) =$
wenn $h(n) \in \mathbb{Z}$, dann $(m+1, d, h, inp, out.h(n))$

$\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =$
wenn $h(n) \in \mathbb{Z}$, dann $(m+1, h(n) : d, h, inp, out)$

Befehlssemantik (I)

$\mathcal{C}[\![.]\!]\colon \Gamma \longrightarrow (\mathsf{AM}_0 \dashrightarrow \mathsf{AM}_0)$

$\mathcal{C}[\![\mathsf{READ}\ n]\!](m, d, h, inp, out) =$
wenn $inp = first(inp).rest(inp)$ mit $first(inp) \in \mathbb{Z}, rest(inp) \in \mathbb{Z}^*$,
dann $(m+1, d, h[n/first(inp)], rest(inp), out)$

$\mathcal{C}[\![\mathsf{WRITE}\ n]\!](m, d, h, inp, out) =$
wenn $h(n) \in \mathbb{Z}$, dann $(m+1, d, h, inp, out.h(n))$

$\mathcal{C}[\![\mathsf{LOAD}\ n]\!](m, d, h, inp, out) =$
wenn $h(n) \in \mathbb{Z}$, dann $(m+1, h(n) : d, h, inp, out)$

$\mathcal{C}[\![\mathsf{STORE}\ n]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d'$, dann $(m+1, d', h[n/d.1], inp, out)$

Befehlssemantik (I)

$\mathcal{C}[\![.]\!]\colon \Gamma \longrightarrow (\mathsf{AM}_0 \dashrightarrow \mathsf{AM}_0)$

$\mathcal{C}[\![\text{READ } n]\!](m, d, h, inp, out) =$
wenn $inp = first(inp).rest(inp)$ mit $first(inp) \in \mathbb{Z}, rest(inp) \in \mathbb{Z}^*$,
dann $(m+1, d, h[n/first(inp)], rest(inp), out)$

$\mathcal{C}[\![\text{WRITE } n]\!](m, d, h, inp, out) =$
wenn $h(n) \in \mathbb{Z}$, dann $(m+1, d, h, inp, out.h(n))$

$\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =$
wenn $h(n) \in \mathbb{Z}$, dann $(m+1, h(n) : d, h, inp, out)$

$\mathcal{C}[\![\text{STORE } n]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d'$, dann $(m+1, d', h[n/d.1], inp, out)$

$\mathcal{C}[\![\text{LIT } z]\!](m, d, h, inp, out) = (m+1, z : d, h, inp, out)$

Befehlssemantik (II)

$$\mathcal{C}[\![\mathsf{ADD}]\!](m, d, h, inp, out) =$$
$$\quad \text{wenn } d = d.1 : d.2 : d', \text{ dann } (m + 1, (d.2 + d.1) : d', h, inp, out)$$

Befehlssemantik (II)

$\mathcal{C}[\![\text{ADD}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, (d.2 + d.1) : d', h, inp, out)$

für MUL, SUB, DIV und MOD analog

Befehlssemantik (II)

$\mathcal{C}[\![\text{ADD}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, (d.2 + d.1) : d', h, inp, out)$

für MUL, SUB, DIV und MOD analog

$\mathcal{C}[\![\text{LT}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, b : d', h, inp, out)$,
wobei $b = 1$, falls $d.2 < d.1$, sonst $b = 0$

Befehlssemantik (II)

$\mathcal{C}[\![\text{ADD}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, (d.2 + d.1) : d', h, inp, out)$

für MUL, SUB, DIV und MOD analog

$\mathcal{C}[\![\text{LT}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, b : d', h, inp, out)$,
wobei $b = 1$, falls $d.2 < d.1$, sonst $b = 0$

für EQ, NE, GT, LE und GE analog

Befehlssemantik (II)

$\mathcal{C}[\![\text{ADD}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, (d.2 + d.1) : d', h, inp, out)$

für MUL, SUB, DIV und MOD analog

$\mathcal{C}[\![\text{LT}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, b : d', h, inp, out)$,
wobei $b = 1$, falls $d.2 < d.1$, sonst $b = 0$

für EQ, NE, GT, LE und GE analog

$\mathcal{C}[\![\text{JMP } e]\!](m, d, h, inp, out) = (e, d, h, inp, out)$

Befehlssemantik (II)

$\mathcal{C}[\![\text{ADD}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, (d.2 + d.1) : d', h, inp, out)$

für MUL, SUB, DIV und MOD analog

$\mathcal{C}[\![\text{LT}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, b : d', h, inp, out)$,
wobei $b = 1$, falls $d.2 < d.1$, sonst $b = 0$

für EQ, NE, GT, LE und GE analog

$\mathcal{C}[\![\text{JMP } e]\!](m, d, h, inp, out) = (e, d, h, inp, out)$

$\mathcal{C}[\![\text{JMC } e]\!](m, d, h, inp, out) =$
wenn $d = 0 : d'$, dann (e, d', h, inp, out);

Befehlssemantik (II)

$\mathcal{C}[\![\text{ADD}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, (d.2 + d.1) : d', h, inp, out)$

für MUL, SUB, DIV und MOD analog

$\mathcal{C}[\![\text{LT}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, b : d', h, inp, out)$,
wobei $b = 1$, falls $d.2 < d.1$, sonst $b = 0$

für EQ, NE, GT, LE und GE analog

$\mathcal{C}[\![\text{JMP } e]\!](m, d, h, inp, out) = (e, d, h, inp, out)$

$\mathcal{C}[\![\text{JMC } e]\!](m, d, h, inp, out) =$
wenn $d = 0 : d'$, dann (e, d', h, inp, out);
wenn $d = 1 : d'$, dann $(m + 1, d', h, inp, out)$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$(\ 1\ ,\quad \varepsilon\ ,\ [] \qquad\qquad ,\ 2\ \ ,\ \varepsilon\ )$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;          15: STORE 3;
 2: LIT 1;       9: JMC 21;      16: LOAD 1;
 3: STORE 1;    10: LOAD 3;      17: LIT 1;
 4: LIT 0;      11: LOAD 1;      18: ADD;
 5: STORE 3;    12: LOAD 1;      19: STORE 1;
 6: LOAD 1;     13: MUL;         20: JMP 6;
 7: LOAD 2;     14: ADD;         21: WRITE 3;
```

$(\ 1\ ,\ \ \varepsilon\ ,\ [\ ]\ \ ,\ 2\ \ ,\ \varepsilon\ )$
$(\ 2\ ,\ \ \varepsilon\ ,\ [2/2]\ \ ,\ \varepsilon\ \ ,\ \varepsilon\ )$

$\mathcal{C}[\![\text{READ } n]\!](m, d, h, inp, out) =$
wenn $inp = first(inp).rest(inp)$ mit $first(inp) \in \mathbb{Z}, rest(inp) \in \mathbb{Z}^*$,
dann $(m+1, d, h[n/first(inp)], rest(inp), out)$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{lll}
(& 1 , & \varepsilon , [] \quad , 2 , \varepsilon) \\
(& 2 , & \varepsilon , [2/2] \quad , \varepsilon , \varepsilon) \\
(& 3 , & 1 , [2/2] \quad , \varepsilon , \varepsilon)
\end{array}
$$

$$\mathcal{C}[\![\text{LIT } z]\!](m, d, h, inp, out) = (m + 1, z : d, h, inp, out)$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{llll}
(& 1 , & \varepsilon , [] & , 2 , \varepsilon) \\
(& 2 , & \varepsilon , [2/2] & , \varepsilon , \varepsilon) \\
(& 3 , & 1 , [2/2] & , \varepsilon , \varepsilon) \\
(& 4 , & \varepsilon , [1/1,2/2] & , \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{STORE } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d', \text{ dann } (m+1, d', h[n/d.1], inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{llll}
(& 1 , & \varepsilon , [] & , 2 , \varepsilon) \\
(& 2 , & \varepsilon , [2/2] & , \varepsilon , \varepsilon) \\
(& 3 , & 1 , [2/2] & , \varepsilon , \varepsilon) \\
(& 4 , & \varepsilon , [1/1,2/2] & , \varepsilon , \varepsilon) \\
(& 5 , & 0 , [1/1,2/2] & , \varepsilon , \varepsilon)
\end{array}
$$

$$\mathcal{C}[\![\text{LIT } z]\!](m, d, h, inp, out) = (m + 1, z : d, h, inp, out)$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{llll}
(& 2 , & \varepsilon , [2/2] & , \varepsilon , \varepsilon) \\
(& 3 , & 1 , [2/2] & , \varepsilon , \varepsilon) \\
(& 4 , & \varepsilon , [1/1,2/2] & , \varepsilon , \varepsilon) \\
(& 5 , & 0 , [1/1,2/2] & , \varepsilon , \varepsilon) \\
(& 6 , & \varepsilon , [1/1,2/2,3/0] & , \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{STORE } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d', \text{ dann } (m + 1, d', h[n/d.1], inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{llll}
(& 3 , & 1 , [2/2] & , \varepsilon , \varepsilon) \\
(& 4 , & \varepsilon , [1/1,2/2] & , \varepsilon , \varepsilon) \\
(& 5 , & 0 , [1/1,2/2] & , \varepsilon , \varepsilon) \\
(& 6 , & \varepsilon , [1/1,2/2,3/0] & , \varepsilon , \varepsilon) \\
(& 7 , & 1 , [1/1,2/2,3/0] & , \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrlll}
(& 4 , & \varepsilon , & [1/1,2/2] & , \varepsilon , \varepsilon) \\
(& 5 , & 0 , & [1/1,2/2] & , \varepsilon , \varepsilon) \\
(& 6 , & \varepsilon , & [1/1,2/2,3/0] & , \varepsilon , \varepsilon) \\
(& 7 , & 1 , & [1/1,2/2,3/0] & , \varepsilon , \varepsilon) \\
(& 8 , & 2{:}1 , & [1/1,2/2,3/0] & , \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{llll}
(& 5 , & 0 , [1/1,2/2] & , \varepsilon , \varepsilon) \\
(& 6 , & \varepsilon , [1/1,2/2,3/0] & , \varepsilon , \varepsilon) \\
(& 7 , & 1 , [1/1,2/2,3/0] & , \varepsilon , \varepsilon) \\
(& 8 , & 2{:}1 , [1/1,2/2,3/0] & , \varepsilon , \varepsilon) \\
(& 9 , & 1 , [1/1,2/2,3/0] & , \varepsilon , \varepsilon)
\end{array}
$$

$\mathcal{C}[\![\mathsf{LE}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, b : d', h, inp, out)$,
wobei $b = 1$, falls $d.2 \leq d.1$, sonst $b = 0$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;        15: STORE 3;
 2: LIT 1;       9: JMC 21;    16: LOAD 1;
 3: STORE 1;    10: LOAD 3;    17: LIT 1;
 4: LIT 0;      11: LOAD 1;    18: ADD;
 5: STORE 3;    12: LOAD 1;    19: STORE 1;
 6: LOAD 1;     13: MUL;       20: JMP 6;
 7: LOAD 2;     14: ADD;       21: WRITE 3;
```

$$
\begin{array}{lrlll}
(& 6 , & \varepsilon , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon) \\
(& 7 , & 1 , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon) \\
(& 8 , & 2{:}1 , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon) \\
(& 9 , & 1 , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon) \\
(& 10 , & \varepsilon , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon)
\end{array}
$$

$$
\begin{aligned}
&\mathcal{C}[\![\text{JMC } e]\!](m, d, h, inp, out) = \\
&\quad \text{wenn } d = 0 : d', \text{ dann } (e, d', h, inp, out); \\
&\quad \text{wenn } d = 1 : d', \text{ dann } (m+1, d', h, inp, out)
\end{aligned}
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrlll}
(& 7 , & 1 , & [1/1,2/2,3/0] , & \varepsilon \ , \varepsilon) \\
(& 8 , & 2{:}1 , & [1/1,2/2,3/0] , & \varepsilon \ , \varepsilon) \\
(& 9 , & 1 , & [1/1,2/2,3/0] , & \varepsilon \ , \varepsilon) \\
(& 10 , & \varepsilon , & [1/1,2/2,3/0] , & \varepsilon \ , \varepsilon) \\
(& 11 , & 0 , & [1/1,2/2,3/0] , & \varepsilon \ , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;        15: STORE 3;
 2: LIT 1;       9: JMC 21;    16: LOAD 1;
 3: STORE 1;    10: LOAD 3;    17: LIT 1;
 4: LIT 0;      11: LOAD 1;    18: ADD;
 5: STORE 3;    12: LOAD 1;    19: STORE 1;
 6: LOAD 1;     13: MUL;       20: JMP 6;
 7: LOAD 2;     14: ADD;       21: WRITE 3;
```

$$
\begin{array}{lrl}
(& 8 , & 2{:}1 , [1/1,2/2,3/0] , \varepsilon , \varepsilon) \\
(& 9 , & 1 , [1/1,2/2,3/0] , \varepsilon , \varepsilon) \\
(& 10 , & \varepsilon , [1/1,2/2,3/0] , \varepsilon , \varepsilon) \\
(& 11 , & 0 , [1/1,2/2,3/0] , \varepsilon , \varepsilon) \\
(& 12 , & 1{:}0 , [1/1,2/2,3/0] , \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{llllll}
(& 9 , & 1 , & [1/1,2/2,3/0] , & \varepsilon , & \varepsilon) \\
(& 10 , & \varepsilon , & [1/1,2/2,3/0] , & \varepsilon , & \varepsilon) \\
(& 11 , & 0 , & [1/1,2/2,3/0] , & \varepsilon , & \varepsilon) \\
(& 12 , & 1{:}0 , & [1/1,2/2,3/0] , & \varepsilon , & \varepsilon) \\
(& 13 , & 1{:}1{:}0 , & [1/1,2/2,3/0] , & \varepsilon , & \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;          15: STORE 3;
 2: LIT 1;       9: JMC 21;      16: LOAD 1;
 3: STORE 1;    10: LOAD 3;      17: LIT 1;
 4: LIT 0;      11: LOAD 1;      18: ADD;
 5: STORE 3;    12: LOAD 1;      19: STORE 1;
 6: LOAD 1;     13: MUL;         20: JMP 6;
 7: LOAD 2;     14: ADD;         21: WRITE 3;
```

$$
\begin{array}{rrrrr}
(& 10 , & \varepsilon , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon) \\
(& 11 , & 0 , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon) \\
(& 12 , & 1{:}0 , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon) \\
(& 13 , & 1{:}1{:}0 , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon) \\
(& 14 , & 1{:}0 , & [1/1,2/2,3/0] , & \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\mathsf{MUL}]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d.2 : d', \text{ dann } (m+1, (d.2 * d.1) : d', h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{l}
(\ 11\ ,\ \ \ \ 0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 12\ ,\ \ \ 1{:}0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 13\ ,\ 1{:}1{:}0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 14\ ,\ \ \ 1{:}0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 15\ ,\ \ \ \ 1\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ )
\end{array}
$$

$$
\mathcal{C}[\![\mathsf{ADD}]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d.2 : d', \text{ dann } (m+1, (d.2 + d.1) : d', h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{l}
(\ 12\ ,\quad 1{:}0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 13\ ,\ 1{:}1{:}0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 14\ ,\quad 1{:}0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 15\ ,\qquad 1\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 16\ ,\qquad \varepsilon\ ,\ [1/1,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ )
\end{array}
$$

$$
\mathcal{C}[\![\text{STORE } n]\!](m, d, h, inp, out) = \\
\quad \text{wenn } d = d.1 : d', \text{ dann } (m+1, d', h[n/d.1], inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{l}
(\ 13\ ,\ 1{:}1{:}0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 14\ ,\ \ \ 1{:}0\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 15\ ,\ \ \ \ \ 1\ ,\ [1/1,2/2,3/0]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 16\ ,\ \ \ \ \ \varepsilon\ ,\ [1/1,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 17\ ,\ \ \ \ \ 1\ ,\ [1/1,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ )
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{llllll}
(& 14 , & 1{:}0 , & [1/1,2/2,3/0] , & \varepsilon , & \varepsilon) \\
(& 15 , & 1 , & [1/1,2/2,3/0] , & \varepsilon , & \varepsilon) \\
(& 16 , & \varepsilon , & [1/1,2/2,3/1] , & \varepsilon , & \varepsilon) \\
(& 17 , & 1 , & [1/1,2/2,3/1] , & \varepsilon , & \varepsilon) \\
(& 18 , & 1{:}1 , & [1/1,2/2,3/1] , & \varepsilon , & \varepsilon)
\end{array}
$$

$$\mathcal{C}[\![\text{LIT } z]\!](m, d, h, inp, out) = (m + 1, z : d, h, inp, out)$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{llllll}
(& 15 , & 1 & , [1/1,2/2,3/0] & , \varepsilon & , \varepsilon) \\
(& 16 , & \varepsilon & , [1/1,2/2,3/1] & , \varepsilon & , \varepsilon) \\
(& 17 , & 1 & , [1/1,2/2,3/1] & , \varepsilon & , \varepsilon) \\
(& 18 , & 1{:}1 & , [1/1,2/2,3/1] & , \varepsilon & , \varepsilon) \\
(& 19 , & 2 & , [1/1,2/2,3/1] & , \varepsilon & , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\mathsf{ADD}]\!](m, d, h, inp, out) =
$$

wenn $d = d.1 : d.2 : d'$, dann $(m + 1, (d.2 + d.1) : d', h, inp, out)$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;          15: STORE 3;
 2: LIT 1;       9: JMC 21;      16: LOAD 1;
 3: STORE 1;    10: LOAD 3;      17: LIT 1;
 4: LIT 0;      11: LOAD 1;      18: ADD;
 5: STORE 3;    12: LOAD 1;      19: STORE 1;
 6: LOAD 1;     13: MUL;         20: JMP 6;
 7: LOAD 2;     14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrl}
(& 16 , & \varepsilon , [1/1,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 17 , & 1 , [1/1,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 18 , & 1{:}1 , [1/1,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 19 , & 2 , [1/1,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 20 , & \varepsilon , [1/2,2/2,3/1] , \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{STORE } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d', \text{ dann } (m+1, d', h[n/d.1], inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{l}
(\ 17\ ,\quad\ \ 1\ ,\ [1/1,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 18\ ,\quad 1{:}1\ ,\ [1/1,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 19\ ,\quad\ \ 2\ ,\ [1/1,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 20\ ,\quad\ \ \varepsilon\ ,\ [1/2,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ \ 6\ ,\quad\ \ \varepsilon\ ,\ [1/2,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ )
\end{array}
$$

$$\mathcal{C}[\![\text{JMP } e]\!](m, d, h, inp, out) = (e, d, h, inp, out)$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{llll}
(& 18 , & 1{:}1 , [1/1,2/2,3/1] , \varepsilon & , \varepsilon) \\
(& 19 , & 2 , [1/1,2/2,3/1] , \varepsilon & , \varepsilon) \\
(& 20 , & \varepsilon , [1/2,2/2,3/1] , \varepsilon & , \varepsilon) \\
(& 6 , & \varepsilon , [1/2,2/2,3/1] , \varepsilon & , \varepsilon) \\
(& 7 , & 2 , [1/2,2/2,3/1] , \varepsilon & , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{llllll}
(& 19 , & 2 , & [1/1,2/2,3/1] , & \varepsilon , & \varepsilon) \\
(& 20 , & \varepsilon , & [1/2,2/2,3/1] , & \varepsilon , & \varepsilon) \\
(& 6 , & \varepsilon , & [1/2,2/2,3/1] , & \varepsilon , & \varepsilon) \\
(& 7 , & 2 , & [1/2,2/2,3/1] , & \varepsilon , & \varepsilon) \\
(& 8 , & 2{:}2 , & [1/2,2/2,3/1] , & \varepsilon , & \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;        15: STORE 3;
 2: LIT 1;       9: JMC 21;    16: LOAD 1;
 3: STORE 1;    10: LOAD 3;    17: LIT 1;
 4: LIT 0;      11: LOAD 1;    18: ADD;
 5: STORE 3;    12: LOAD 1;    19: STORE 1;
 6: LOAD 1;     13: MUL;       20: JMP 6;
 7: LOAD 2;     14: ADD;       21: WRITE 3;
```

$$
\begin{array}{lrlll}
(& 20 , & \varepsilon , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon) \\
(& 6 , & \varepsilon , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon) \\
(& 7 , & 2 , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon) \\
(& 8 , & 2{:}2 , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon) \\
(& 9 , & 1 , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon)
\end{array}
$$

$\mathcal{C}[\![\mathsf{LE}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, b : d', h, inp, out)$,
wobei $b = 1$, falls $d.2 \leq d.1$, sonst $b = 0$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{lrl}
(& 6 , & \varepsilon , [1/2,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 7 , & 2 , [1/2,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 8 , & 2{:}2 , [1/2,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 9 , & 1 , [1/2,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 10 , & \varepsilon , [1/2,2/2,3/1] , \varepsilon , \varepsilon)
\end{array}
$$

$$
\begin{aligned}
&\mathcal{C}[\![\text{JMC } e]\!](m, d, h, inp, out) = \\
&\quad \text{wenn } d = 0 : d', \text{ dann } (e, d', h, inp, out); \\
&\quad \text{wenn } d = 1 : d', \text{ dann } (m+1, d', h, inp, out)
\end{aligned}
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrlll}
(& 7 , & 2 , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon) \\
(& 8 , & 2{:}2 , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon) \\
(& 9 , & 1 , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon) \\
(& 10 , & \varepsilon , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon) \\
(& 11 , & 1 , & [1/2,2/2,3/1] , & \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrlll}
(& 8 , & 2{:}2 , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon) \\
(& 9 , & 1 , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon) \\
(& 10 , & \varepsilon , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon) \\
(& 11 , & 1 , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon) \\
(& 12 , & 2{:}1 , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;          15: STORE 3;
 2: LIT 1;       9: JMC 21;      16: LOAD 1;
 3: STORE 1;    10: LOAD 3;      17: LIT 1;
 4: LIT 0;      11: LOAD 1;      18: ADD;
 5: STORE 3;    12: LOAD 1;      19: STORE 1;
 6: LOAD 1;     13: MUL;         20: JMP 6;
 7: LOAD 2;     14: ADD;         21: WRITE 3;
```

$$
\begin{array}{rrlll}
(& 9 , & 1 , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon) \\
(& 10 , & \varepsilon , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon) \\
(& 11 , & 1 , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon) \\
(& 12 , & 2{:}1 , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon) \\
(& 13 , & 2{:}2{:}1 , & [1/2,2/2,3/1] , & \varepsilon \ , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;          15: STORE 3;
 2: LIT 1;       9: JMC 21;      16: LOAD 1;
 3: STORE 1;    10: LOAD 3;      17: LIT 1;
 4: LIT 0;      11: LOAD 1;      18: ADD;
 5: STORE 3;    12: LOAD 1;      19: STORE 1;
 6: LOAD 1;     13: MUL;         20: JMP 6;
 7: LOAD 2;     14: ADD;         21: WRITE 3;
```

$$
\begin{array}{l}
(\ 10\ ,\quad\ \varepsilon\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 11\ ,\quad\ \ 1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 12\ ,\quad 2{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 13\ ,\ 2{:}2{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 14\ ,\quad 4{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ )
\end{array}
$$

$$
\mathcal{C}[\![\mathsf{MUL}]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d.2 : d', \text{ dann } (m+1, (d.2 * d.1) : d', h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;        15: STORE 3;
 2: LIT 1;       9: JMC 21;    16: LOAD 1;
 3: STORE 1;    10: LOAD 3;    17: LIT 1;
 4: LIT 0;      11: LOAD 1;    18: ADD;
 5: STORE 3;    12: LOAD 1;    19: STORE 1;
 6: LOAD 1;     13: MUL;       20: JMP 6;
 7: LOAD 2;     14: ADD;       21: WRITE 3;
```

$$
\begin{array}{l}
(\ 11\ ,\ \ \ \ 1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 12\ ,\ \ 2{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 13\ ,\ 2{:}2{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 14\ ,\ \ 4{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 15\ ,\ \ \ \ 5\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ )
\end{array}
$$

$$
\mathcal{C}[\![\mathsf{ADD}]\!](m, d, h, inp, out) = \\
\quad \text{wenn } d = d.1 : d.2 : d', \text{ dann } (m+1, (d.2 + d.1) : d', h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;          15: STORE 3;
 2: LIT 1;       9: JMC 21;      16: LOAD 1;
 3: STORE 1;    10: LOAD 3;      17: LIT 1;
 4: LIT 0;      11: LOAD 1;      18: ADD;
 5: STORE 3;    12: LOAD 1;      19: STORE 1;
 6: LOAD 1;     13: MUL;         20: JMP 6;
 7: LOAD 2;     14: ADD;         21: WRITE 3;
```

$$
\begin{array}{l}
(\ 12\ ,\quad 2{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 13\ ,\ 2{:}2{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 14\ ,\quad 4{:}1\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 15\ ,\qquad 5\ ,\ [1/2{,}2/2{,}3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 16\ ,\qquad \varepsilon\ ,\ [1/2{,}2/2{,}3/5]\ ,\ \varepsilon\ \ ,\ \varepsilon\ )
\end{array}
$$

$$
\mathcal{C}[\![\text{STORE } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d', \text{ dann } (m+1, d', h[n/d.1], inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{l}
(\ 13\ ,\ 2{:}2{:}1\ ,\ [1/2,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 14\ ,\ \ \ 4{:}1\ ,\ [1/2,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 15\ ,\ \ \ \ \ 5\ ,\ [1/2,2/2,3/1]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 16\ ,\ \ \ \ \ \varepsilon\ ,\ [1/2,2/2,3/5]\ ,\ \varepsilon\ \ ,\ \varepsilon\ ) \\
(\ 17\ ,\ \ \ \ \ 2\ ,\ [1/2,2/2,3/5]\ ,\ \varepsilon\ \ ,\ \varepsilon\ )
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{llllll}
(& 14 , & 4{:}1 , & [1/2,2/2,3/1] , & \varepsilon , & \varepsilon) \\
(& 15 , & 5 , & [1/2,2/2,3/1] , & \varepsilon , & \varepsilon) \\
(& 16 , & \varepsilon , & [1/2,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 17 , & 2 , & [1/2,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 18 , & 1{:}2 , & [1/2,2/2,3/5] , & \varepsilon , & \varepsilon)
\end{array}
$$

$$\mathcal{C}[\![\text{LIT } z]\!](m, d, h, inp, out) = (m + 1, z : d, h, inp, out)$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;          15: STORE 3;
 2: LIT 1;       9: JMC 21;      16: LOAD 1;
 3: STORE 1;    10: LOAD 3;      17: LIT 1;
 4: LIT 0;      11: LOAD 1;      18: ADD;
 5: STORE 3;    12: LOAD 1;      19: STORE 1;
 6: LOAD 1;     13: MUL;         20: JMP 6;
 7: LOAD 2;     14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrl}
(& 15 , & 5 , [1/2,2/2,3/1] , \varepsilon , \varepsilon) \\
(& 16 , & \varepsilon , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(& 17 , & 2 , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(& 18 , & 1{:}2 , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(& 19 , & 3 , [1/2,2/2,3/5] , \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\mathsf{ADD}]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d.2 : d', \text{ dann } (m+1, (d.2 + d.1) : d', h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrl}
(& 16 , & \varepsilon , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(& 17 , & 2 , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(& 18 , & 1{:}2 , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(& 19 , & 3 , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(& 20 , & \varepsilon , [1/3,2/2,3/5] , \varepsilon , \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{STORE } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } d = d.1 : d', \text{ dann } (m+1, d', h[n/d.1], inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lr}
(17 , & 2 , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(18 , & 1{:}2 , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(19 , & 3 , [1/2,2/2,3/5] , \varepsilon , \varepsilon) \\
(20 , & \varepsilon , [1/3,2/2,3/5] , \varepsilon , \varepsilon) \\
(6 , & \varepsilon , [1/3,2/2,3/5] , \varepsilon , \varepsilon)
\end{array}
$$

$$\mathcal{C}[\![\text{JMP } e]\!](m, d, h, inp, out) = (e, d, h, inp, out)$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{llllll}
(& 18 , & 1{:}2 , & [1/2,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 19 , & 3 , & [1/2,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 20 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 6 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 7 , & 3 , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{llllll}
(& 19 , & 3 , & [1/2,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 20 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 6 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 7 , & 3 , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 8 , & 2{:}3 , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon)
\end{array}
$$

$$
\mathcal{C}[\![\text{LOAD } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, h(n) : d, h, inp, out)
$$

Programmsemantik — am Beispiel

```
 1: READ 2;      8: LE;        15: STORE 3;
 2: LIT 1;       9: JMC 21;    16: LOAD 1;
 3: STORE 1;    10: LOAD 3;    17: LIT 1;
 4: LIT 0;      11: LOAD 1;    18: ADD;
 5: STORE 3;    12: LOAD 1;    19: STORE 1;
 6: LOAD 1;     13: MUL;       20: JMP 6;
 7: LOAD 2;     14: ADD;       21: WRITE 3;
```

$$
\begin{array}{rrrrr}
(& 20 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon) \\
(& 6 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon) \\
(& 7 , & 3 , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon) \\
(& 8 , & 2{:}3 , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon)
\end{array}
$$

$\mathcal{C}[\![\mathsf{LE}]\!](m, d, h, inp, out) =$
wenn $d = d.1 : d.2 : d'$, dann $(m + 1, b : d', h, inp, out)$,
wobei $b = 1$, falls $d.2 \leq d.1$, sonst $b = 0$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrlll}
(& 6 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon) \\
(& 7 , & 3 , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon) \\
(& 8 , & 2{:}3 , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon) \\
(& 21 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , \varepsilon)
\end{array}
$$

$$
\begin{aligned}
&\mathcal{C}[\![\text{JMC } e]\!](m, d, h, inp, out) = \\
&\quad \text{wenn } d = 0 : d', \text{ dann } (e, d', h, inp, out); \\
&\quad \text{wenn } d = 1 : d', \text{ dann } (m+1, d', h, inp, out)
\end{aligned}
$$

Programmsemantik — am Beispiel

```
 1: READ 2;        8: LE;          15: STORE 3;
 2: LIT 1;         9: JMC 21;      16: LOAD 1;
 3: STORE 1;      10: LOAD 3;      17: LIT 1;
 4: LIT 0;        11: LOAD 1;      18: ADD;
 5: STORE 3;      12: LOAD 1;      19: STORE 1;
 6: LOAD 1;       13: MUL;         20: JMP 6;
 7: LOAD 2;       14: ADD;         21: WRITE 3;
```

$$
\begin{array}{lrlll}
(& 7 , & 3 , & [1/3,2/2,3/5] , & \varepsilon \ , \varepsilon) \\
(& 8 , & 2{:}3 , & [1/3,2/2,3/5] , & \varepsilon \ , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/2,3/5] , & \varepsilon \ , \varepsilon) \\
(& 21 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon \ , \varepsilon) \\
(& 22 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon \ , 5)
\end{array}
$$

$$
\mathcal{C}[\![\text{WRITE } n]\!](m, d, h, inp, out) =
$$
$$
\quad \text{wenn } h(n) \in \mathbb{Z}, \text{ dann } (m+1, d, h, inp, out.h(n))
$$

Programmsemantik — am Beispiel

```
 1: READ 2;
 2: LIT 1;
 3: STORE 1;
 4: LIT 0;
 5: STORE 3;
 6: LOAD 1;
 7: LOAD 2;
 8: LE;
 9: JMC 21;
10: LOAD 3;
11: LOAD 1;
12: LOAD 1;
13: MUL;
14: ADD;
15: STORE 3;
16: LOAD 1;
17: LIT 1;
18: ADD;
19: STORE 1;
20: JMP 6;
21: WRITE 3;
```

$$
\begin{array}{llllll}
(& 7 , & 3 , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 8 , & 2{:}3 , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 9 , & 0 , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 21 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , & \varepsilon) \\
(& 22 , & \varepsilon , & [1/3,2/2,3/5] , & \varepsilon , & 5)
\end{array}
$$

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

- READ n: Lesen von Eingabeband in Hauptspeicher

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

- READ n: Lesen von Eingabeband in Hauptspeicher
- WRITE n: Ausgabe aus Hauptspeicher auf Ausgabeband

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

- READ n: Lesen von Eingabeband in Hauptspeicher
- WRITE n: Ausgabe aus Hauptspeicher auf Ausgabeband
- LOAD n: Ablage aus Hauptspeicher auf Datenkeller

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

- READ n: Lesen von Eingabeband in Hauptspeicher
- WRITE n: Ausgabe aus Hauptspeicher auf Ausgabeband
- LOAD n: Ablage aus Hauptspeicher auf Datenkeller
- STORE n: Entnahme aus Datenkeller in Hauptspeicher

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

- READ n: Lesen von Eingabeband in Hauptspeicher
- WRITE n: Ausgabe aus Hauptspeicher auf Ausgabeband
- LOAD n: Ablage aus Hauptspeicher auf Datenkeller
- STORE n: Entnahme aus Datenkeller in Hauptspeicher
- LIT z: Ablage einer Konstante auf Datenkeller

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

- READ n: Lesen von Eingabeband in Hauptspeicher
- WRITE n: Ausgabe aus Hauptspeicher auf Ausgabeband
- LOAD n: Ablage aus Hauptspeicher auf Datenkeller
- STORE n: Entnahme aus Datenkeller in Hauptspeicher
- LIT z: Ablage einer Konstante auf Datenkeller
- ADD, MUL, SUB, DIV, MOD, LT, EQ, NE, GT, LE, GE: Berechnungen und Vergleiche (auf Datenkeller)

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

- READ n: Lesen von Eingabeband in Hauptspeicher
- WRITE n: Ausgabe aus Hauptspeicher auf Ausgabeband
- LOAD n: Ablage aus Hauptspeicher auf Datenkeller
- STORE n: Entnahme aus Datenkeller in Hauptspeicher
- LIT z: Ablage einer Konstante auf Datenkeller
- ADD, MUL, SUB, DIV, MOD, LT, EQ, NE, GT, LE, GE: Berechnungen und Vergleiche (auf Datenkeller)
- JMP n: Sprung

Wiederholung — AM_0

$AM_0 = BZ \times DK \times HS \times \underline{Inp} \times \underline{Out}$ mit:

BZ	$= \mathbb{N}$	Befehlszähler
DK	$= \mathbb{Z}^*$	Datenkeller
HS	$= \{h \mid h : \mathbb{N} \rightharpoonup \mathbb{Z}\}$	Hauptspeicher
$\underline{Inp}$	$= \mathbb{Z}^*$	Eingabeband
$\underline{Out}$	$= \mathbb{Z}^*$	Ausgabeband

- READ n: Lesen von Eingabeband in Hauptspeicher
- WRITE n: Ausgabe aus Hauptspeicher auf Ausgabeband
- LOAD n: Ablage aus Hauptspeicher auf Datenkeller
- STORE n: Entnahme aus Datenkeller in Hauptspeicher
- LIT z: Ablage einer Konstante auf Datenkeller
- ADD, MUL, SUB, DIV, MOD, LT, EQ, NE, GT, LE, GE: Berechnungen und Vergleiche (auf Datenkeller)
- JMP n: Sprung
- JMC n: Sprung abhängig von Datenkeller

Programmsemantik — formal

(zur Erinnerung: Befehlssemantik $\mathcal{C}[\![.]\!]$: $\Gamma \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$)

Programmsemantik — formal

(zur Erinnerung: Befehlssemantik $\mathcal{C}[\![.]\!]$: $\Gamma \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$)

Sei $Prog_0$ die Menge aller Programme.

$\mathcal{I}[\![.]\!] : Prog_0 \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$

Programmsemantik — formal

(zur Erinnerung: Befehlssemantik $\mathcal{C}[\![.]\!]$: $\Gamma \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$)

Sei $Prog_0$ die Menge aller Programme.

$\mathcal{I}[\![.]\!] : Prog_0 \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$

$$\mathcal{I}[\![\mathrm{P}]\!](m, d, h, inp, out) =$$
$$\begin{cases} \mathcal{I}[\![\mathrm{P}]\!](\mathcal{C}[\![\mathrm{P(m)}]\!](m, d, h, inp, out)), & \text{falls } m \in def(\mathrm{P}) \\ (m, d, h, inp, out), & \text{falls } m \notin def(\mathrm{P}) \end{cases}$$

Programmsemantik — formal

(zur Erinnerung: Befehlssemantik $\mathcal{C}[\![.]\!]$: $\Gamma \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$)

Sei $Prog_0$ die Menge aller Programme.

$\mathcal{I}[\![.]\!] : Prog_0 \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$

$$\mathcal{I}[\![\mathrm{P}]\!](m, d, h, inp, out) = \begin{cases} \mathcal{I}[\![\mathrm{P}]\!](\mathcal{C}[\![\mathrm{P(m)}]\!](m, d, h, inp, out)), & \text{falls } m \in def(\mathrm{P}) \\ (m, d, h, inp, out), & \text{falls } m \notin def(\mathrm{P}) \end{cases}$$

$\mathcal{P}[\![.]\!] : Prog_0 \longrightarrow (\underline{\mathrm{Inp}} \dashrightarrow \underline{\mathrm{Out}})$

Programmsemantik — formal

(zur Erinnerung: Befehlssemantik $\mathcal{C}[\![.]\!]$: $\Gamma \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$)

Sei $Prog_0$ die Menge aller Programme.

$\mathcal{I}[\![.]\!] : Prog_0 \longrightarrow (\mathrm{AM}_0 \dashrightarrow \mathrm{AM}_0)$

$$\mathcal{I}[\![\mathrm{P}]\!](m, d, h, inp, out) = \begin{cases} \mathcal{I}[\![\mathrm{P}]\!](\mathcal{C}[\![\mathrm{P(m)}]\!](m, d, h, inp, out)), & \text{falls } m \in def(\mathrm{P}) \\ (m, d, h, inp, out), & \text{falls } m \notin def(\mathrm{P}) \end{cases}$$

$\mathcal{P}[\![.]\!] : Prog_0 \longrightarrow (\underline{\mathrm{Inp}} \dashrightarrow \underline{\mathrm{Out}})$

$\mathcal{P}[\![\mathrm{P}]\!](inp) = proj_5^{(5)}(\mathcal{I}[\![\mathrm{P}]\!](1, \varepsilon, [], inp, \varepsilon))$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(1 , ε , [] , 3.4.2.0 , ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(1 , ε , [] , 3.4.2.0 , ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;        6: NE;          11: STORE 2;
 2: STORE 2;      7: JMC 13;      12: JMP 3;
 3: READ 1;       8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;       9: LOAD 1;
 5: LIT 0;       10: ADD;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$(\ 1\ ,\ \ \varepsilon\ ,\ [] \quad ,\ 3.4.2.0\ ,\ \varepsilon\ )$

$(\ 2\ ,\ \ 0\ ,\ [] \quad ,\ 3.4.2.0\ ,\ \varepsilon\ )$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lll}
(& 1 , \quad \varepsilon , [] & , 3.4.2.0 , \varepsilon) \\
(& 2 , \quad 0 , [] & , 3.4.2.0 , \varepsilon) \\
(& 3 , \quad \varepsilon , [2/0] & , 3.4.2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lll}
(& 1 , \quad \varepsilon , [] & , 3.4.2.0 , \varepsilon) \\
(& 2 , \quad 0 , [] & , 3.4.2.0 , \varepsilon) \\
(& 3 , \quad \varepsilon , [2/0] & , 3.4.2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 1 , & \varepsilon , [] & , 3.4.2.0 , \varepsilon) \\
(& 2 , & 0 , [] & , 3.4.2.0 , \varepsilon) \\
(& 3 , & \varepsilon , [2/0] & , 3.4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , [1/3,2/0] & , 4.2.0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 1 , & \varepsilon , [] & , 3.4.2.0 , \varepsilon) \\
(& 2 , & 0 , [] & , 3.4.2.0 , \varepsilon) \\
(& 3 , & \varepsilon , [2/0] & , 3.4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , [1/3,2/0] & , 4.2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	ε , [1/3,2/0]	, 4.2.0 ,	ε)
(	5 ,	3 , [1/3,2/0]	, 4.2.0 ,	ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	ε , [1/3,2/0]	, 4.2.0 ,	ε)
(	5 ,	3 , [1/3,2/0]	, 4.2.0 ,	ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 1 , & \varepsilon , [] & , 3.4.2.0 , \varepsilon) \\
(& 2 , & 0 , [] & , 3.4.2.0 , \varepsilon) \\
(& 3 , & \varepsilon , [2/0] & , 3.4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 5 , & 3 , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 6 , & 0{:}3 , [1/3,2/0] & , 4.2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	ε , [1/3,2/0]	, 4.2.0 ,	ε)
(	5 ,	3 , [1/3,2/0]	, 4.2.0 ,	ε)
(	6 ,	0:3 , [1/3,2/0]	, 4.2.0 ,	ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

$$
\begin{array}{llll}
(& 1 , & \varepsilon , [] & , 3.4.2.0 , \varepsilon) \\
(& 2 , & 0 , [] & , 3.4.2.0 , \varepsilon) \\
(& 3 , & \varepsilon , [2/0] & , 3.4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 5 , & 3 , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 6 , & 0{:}3 , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 7 , & 1 , [1/3,2/0] & , 4.2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	ε , [1/3,2/0]	, 4.2.0 ,	ε)
(	5 ,	3 , [1/3,2/0]	, 4.2.0 ,	ε)
(	6 ,	0:3 , [1/3,2/0]	, 4.2.0 ,	ε)
(	7 ,	1 , [1/3,2/0]	, 4.2.0 ,	ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	ε , [1/3,2/0]	, 4.2.0 ,	ε)
(	5 ,	3 , [1/3,2/0]	, 4.2.0 ,	ε)
(	6 ,	0:3 , [1/3,2/0]	, 4.2.0 ,	ε)
(	7 ,	1 , [1/3,2/0]	, 4.2.0 ,	ε)
(	8 ,	ε , [1/3,2/0]	, 4.2.0 ,	ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	ε , [1/3,2/0]	, 4.2.0 ,	ε)
(	5 ,	3 , [1/3,2/0]	, 4.2.0 ,	ε)
(	6 ,	0:3 , [1/3,2/0]	, 4.2.0 ,	ε)
(	7 ,	1 , [1/3,2/0]	, 4.2.0 ,	ε)
(	8 ,	ε , [1/3,2/0]	, 4.2.0 ,	ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

(	2 ,	0 , []	, 3.4.2.0	, ε)
(	3 ,	ε , [2/0]	, 3.4.2.0	, ε)
(	4 ,	ε , [1/3,2/0]	, 4.2.0	, ε)
(	5 ,	3 , [1/3,2/0]	, 4.2.0	, ε)
(	6 ,	0:3 , [1/3,2/0]	, 4.2.0	, ε)
(	7 ,	1 , [1/3,2/0]	, 4.2.0	, ε)
(	8 ,	ε , [1/3,2/0]	, 4.2.0	, ε)
(	9 ,	0 , [1/3,2/0]	, 4.2.0	, ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

(	2 ,	0 , []	, 3.4.2.0	, ε)
(	3 ,	ε , [2/0]	, 3.4.2.0	, ε)
(	4 ,	ε , [1/3,2/0]	, 4.2.0	, ε)
(	5 ,	3 , [1/3,2/0]	, 4.2.0	, ε)
(	6 ,	0:3 , [1/3,2/0]	, 4.2.0	, ε)
(	7 ,	1 , [1/3,2/0]	, 4.2.0	, ε)
(	8 ,	ε , [1/3,2/0]	, 4.2.0	, ε)
(	9 ,	0 , [1/3,2/0]	, 4.2.0	, ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(	3 ,	ε ,	[2/0] ,	3.4.2.0 , ε)
(	4 ,	ε ,	[1/3,2/0] ,	4.2.0 , ε)
(	5 ,	3 ,	[1/3,2/0] ,	4.2.0 , ε)
(	6 ,	0:3 ,	[1/3,2/0] ,	4.2.0 , ε)
(	7 ,	1 ,	[1/3,2/0] ,	4.2.0 , ε)
(	8 ,	ε ,	[1/3,2/0] ,	4.2.0 , ε)
(	9 ,	0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	10 ,	3:0 ,	[1/3,2/0] ,	4.2.0 , ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 3 , & \varepsilon , [2/0] & , 3.4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 5 , & 3 , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 6 , & 0{:}3 , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 7 , & 1 , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 8 , & \varepsilon , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 9 , & 0 , [1/3,2/0] & , 4.2.0 , \varepsilon) \\
(& 10 , & 3{:}0 , [1/3,2/0] & , 4.2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 4 , & \varepsilon , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 6 , & 0{:}3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 7 , & 1 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 8 , & \varepsilon , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 4 , & \varepsilon , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 6 , & 0{:}3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 7 , & 1 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 8 , & \varepsilon , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

$$
\begin{array}{lrlll}
(& 5 , & 3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 6 , & 0{:}3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 7 , & 1 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 8 , & \varepsilon , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3,2/3] , & 4.2.0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

```
(  5 ,   3 , [1/3,2/0] , 4.2.0   , ε )
(  6 , 0:3 , [1/3,2/0] , 4.2.0   , ε )
(  7 ,   1 , [1/3,2/0] , 4.2.0   , ε )
(  8 ,   ε , [1/3,2/0] , 4.2.0   , ε )
(  9 ,   0 , [1/3,2/0] , 4.2.0   , ε )
( 10 , 3:0 , [1/3,2/0] , 4.2.0   , ε )
( 11 ,   3 , [1/3,2/0] , 4.2.0   , ε )
( 12 ,   ε , [1/3,2/3] , 4.2.0   , ε )
```

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llllll}
(& 6 , & 0{:}3 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 7 , & 1 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 9 , & 0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 11 , & 3 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3{,}2/3] , & 4.2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3{,}2/3] , & 4.2.0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llllll}
(& 6 , & 0{:}3 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 7 , & 1 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 9 , & 0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 11 , & 3 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3{,}2/3] , & 4.2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3{,}2/3] , & 4.2.0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 7 , & 1 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 8 , & \varepsilon , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 9 , & 0 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 10 , & 3{:}0 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & \varepsilon , [1/3,2/3] , & 4.2.0 \quad , \varepsilon) \\
(& 3 , & \varepsilon , [1/3,2/3] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & \varepsilon , [1/4,2/3] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llllll}
(& 7 , & 1 , & [1/3,2/0] , & 4.2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/3,2/0] , & 4.2.0 & , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/0] , & 4.2.0 & , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3,2/0] , & 4.2.0 & , \varepsilon) \\
(& 11 , & 3 , & [1/3,2/0] , & 4.2.0 & , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3,2/3] , & 4.2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3,2/3] , & 4.2.0 & , \varepsilon) \\
(& 4 , & \varepsilon , & [1/4,2/3] , & 2.0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 8 , & \varepsilon , & [1/3,2/0] , & 4.2.0 , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/0] , & 4.2.0 , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3,2/0] , & 4.2.0 , \varepsilon) \\
(& 11 , & 3 , & [1/3,2/0] , & 4.2.0 , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3,2/3] , & 4.2.0 , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3,2/3] , & 4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 5 , & 4 , & [1/4,2/3] , & 2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 8 , & \varepsilon , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 9 , & 0 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , & [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3,2/3] , & 4.2.0 \quad , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3,2/3] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & \varepsilon , & [1/4,2/3] , & 2.0 \quad , \varepsilon) \\
(& 5 , & 4 , & [1/4,2/3] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 9 , & 0 , & [1/3,2/0] , 4.2.0 , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3,2/0] , 4.2.0 , \varepsilon) \\
(& 11 , & 3 , & [1/3,2/0] , 4.2.0 , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3,2/3] , 4.2.0 , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3,2/3] , 4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , & [1/4,2/3] , 2.0 , \varepsilon) \\
(& 5 , & 4 , & [1/4,2/3] , 2.0 , \varepsilon) \\
(& 6 , & 0{:}4 , & [1/4,2/3] , 2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 9 , & 0 , & [1/3,2/0] , 4.2.0 , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3,2/0] , 4.2.0 , \varepsilon) \\
(& 11 , & 3 , & [1/3,2/0] , 4.2.0 , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3,2/3] , 4.2.0 , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3,2/3] , 4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , & [1/4,2/3] , 2.0 , \varepsilon) \\
(& 5 , & 4 , & [1/4,2/3] , 2.0 , \varepsilon) \\
(& 6 , & 0{:}4 , & [1/4,2/3] , 2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 10 , & 3{:}0 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & \varepsilon , [1/3,2/3] , & 4.2.0 \quad , \varepsilon) \\
(& 3 , & \varepsilon , [1/3,2/3] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & \varepsilon , [1/4,2/3] , & 2.0 \quad , \varepsilon) \\
(& 5 , & 4 , [1/4,2/3] , & 2.0 \quad , \varepsilon) \\
(& 6 , & 0{:}4 , [1/4,2/3] , & 2.0 \quad , \varepsilon) \\
(& 7 , & 1 , [1/4,2/3] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(10 ,	3:0 ,	[1/3,2/0] ,	4.2.0	, ε)
(11 ,	3 ,	[1/3,2/0] ,	4.2.0	, ε)
(12 ,	ε ,	[1/3,2/3] ,	4.2.0	, ε)
(3 ,	ε ,	[1/3,2/3] ,	4.2.0	, ε)
(4 ,	ε ,	[1/4,2/3] ,	2.0	, ε)
(5 ,	4 ,	[1/4,2/3] ,	2.0	, ε)
(6 ,	0:4 ,	[1/4,2/3] ,	2.0	, ε)
(7 ,	1 ,	[1/4,2/3] ,	2.0	, ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(11 ,	3 ,	[1/3,2/0] ,	4.2.0	, ε)
(12 ,	ε ,	[1/3,2/3] ,	4.2.0	, ε)
(3 ,	ε ,	[1/3,2/3] ,	4.2.0	, ε)
(4 ,	ε ,	[1/4,2/3] ,	2.0	, ε)
(5 ,	4 ,	[1/4,2/3] ,	2.0	, ε)
(6 ,	0:4 ,	[1/4,2/3] ,	2.0	, ε)
(7 ,	1 ,	[1/4,2/3] ,	2.0	, ε)
(8 ,	ε ,	[1/4,2/3] ,	2.0	, ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 11 , & 3 , & [1/3,2/0] , & 4.2.0 , \varepsilon) \\
(& 12 , & \varepsilon , & [1/3,2/3] , & 4.2.0 , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3,2/3] , & 4.2.0 , \varepsilon) \\
(& 4 , & \varepsilon , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 5 , & 4 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 6 , & 0{:}4 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 7 , & 1 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 8 , & \varepsilon , & [1/4,2/3] , & 2.0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llllll}
(& 12 , & \varepsilon , & [1/3,2/3] , & 4.2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3,2/3] , & 4.2.0 & , \varepsilon) \\
(& 4 , & \varepsilon , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 5 , & 4 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 6 , & 0{:}4 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 7 , & 1 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/4,2/3] , & 2.0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llllll}
(& 12 , & \varepsilon , & [1/3,2/3] , & 4.2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , & [1/3,2/3] , & 4.2.0 & , \varepsilon) \\
(& 4 , & \varepsilon , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 5 , & 4 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 6 , & 0{:}4 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 7 , & 1 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/4,2/3] , & 2.0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;        6: NE;          11: STORE 2;
 2: STORE 2;      7: JMC 13;      12: JMP 3;
 3: READ 1;       8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;       9: LOAD 1;
 5: LIT 0;       10: ADD;
```

```
(  3 ,   ε , [1/3,2/3] , 4.2.0   , ε )
(  4 ,   ε , [1/4,2/3] , 2.0     , ε )
(  5 ,   4 , [1/4,2/3] , 2.0     , ε )
(  6 , 0:4 , [1/4,2/3] , 2.0     , ε )
(  7 ,   1 , [1/4,2/3] , 2.0     , ε )
(  8 ,   ε , [1/4,2/3] , 2.0     , ε )
(  9 ,   3 , [1/4,2/3] , 2.0     , ε )
( 10 , 4:3 , [1/4,2/3] , 2.0     , ε )
```

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(3 ,	ε ,	[1/3,2/3] ,	4.2.0	, ε)
(4 ,	ε ,	[1/4,2/3] ,	2.0	, ε)
(5 ,	4 ,	[1/4,2/3] ,	2.0	, ε)
(6 ,	0:4 ,	[1/4,2/3] ,	2.0	, ε)
(7 ,	1 ,	[1/4,2/3] ,	2.0	, ε)
(8 ,	ε ,	[1/4,2/3] ,	2.0	, ε)
(9 ,	3 ,	[1/4,2/3] ,	2.0	, ε)
(10 ,	4:3 ,	[1/4,2/3] ,	2.0	, ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 4 , & \varepsilon , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 5 , & 4 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 6 , & 0{:}4 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 7 , & 1 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 9 , & 3 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 10 , & 4{:}3 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 11 , & 7 , [1/4,2/3] , 2.0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

$$
\begin{array}{lrlll}
(& 4 , & \varepsilon , & [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 5 , & 4 , & [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 6 , & 0{:}4 , & [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 7 , & 1 , & [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 10 , & 4{:}3 , & [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 11 , & 7 , & [1/4,2/3] , 2.0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(5 ,	4 ,	[1/4,2/3] ,	2.0	, ε)
(6 ,	0:4 ,	[1/4,2/3] ,	2.0	, ε)
(7 ,	1 ,	[1/4,2/3] ,	2.0	, ε)
(8 ,	ε ,	[1/4,2/3] ,	2.0	, ε)
(9 ,	3 ,	[1/4,2/3] ,	2.0	, ε)
(10 ,	4:3 ,	[1/4,2/3] ,	2.0	, ε)
(11 ,	7 ,	[1/4,2/3] ,	2.0	, ε)
(12 ,	ε ,	[1/4,2/7] ,	2.0	, ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llllll}
(& 5 , & 4 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 6 , & 0{:}4 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 7 , & 1 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 10 , & 4{:}3 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 11 , & 7 , & [1/4,2/3] , & 2.0 & , \varepsilon) \\
(& 12 , & \varepsilon , & [1/4,2/7] , & 2.0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

```
(  6 , 0:4 , [1/4,2/3] , 2.0    , ε )
(  7 ,   1 , [1/4,2/3] , 2.0    , ε )
(  8 ,   ε , [1/4,2/3] , 2.0    , ε )
(  9 ,   3 , [1/4,2/3] , 2.0    , ε )
( 10 , 4:3 , [1/4,2/3] , 2.0    , ε )
( 11 ,   7 , [1/4,2/3] , 2.0    , ε )
( 12 ,   ε , [1/4,2/7] , 2.0    , ε )
(  3 ,   ε , [1/4,2/7] , 2.0    , ε )
```

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 6 , & 0{:}4 , [1/4{,}2/3] , & 2.0 \quad , \varepsilon) \\
(& 7 , & 1 , [1/4{,}2/3] , & 2.0 \quad , \varepsilon) \\
(& 8 , & \varepsilon , [1/4{,}2/3] , & 2.0 \quad , \varepsilon) \\
(& 9 , & 3 , [1/4{,}2/3] , & 2.0 \quad , \varepsilon) \\
(& 10 , & 4{:}3 , [1/4{,}2/3] , & 2.0 \quad , \varepsilon) \\
(& 11 , & 7 , [1/4{,}2/3] , & 2.0 \quad , \varepsilon) \\
(& 12 , & \varepsilon , [1/4{,}2/7] , & 2.0 \quad , \varepsilon) \\
(& 3 , & \varepsilon , [1/4{,}2/7] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 7 , & 1 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 9 , & 3 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 10 , & 4{:}3 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 11 , & 7 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/4,2/7] , 2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/4,2/7] , 2.0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 7 , & 1 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 9 , & 3 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 10 , & 4{:}3 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 11 , & 7 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/4,2/7] , 2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/4,2/7] , 2.0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 8 , & \varepsilon , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 9 , & 3 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 10 , & 4{:}3 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 11 , & 7 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 12 , & \varepsilon , & [1/4,2/7] , & 2.0 , \varepsilon) \\
(& 3 , & \varepsilon , & [1/4,2/7] , & 2.0 , \varepsilon) \\
(& 4 , & \varepsilon , & [1/2,2/7] , & 0 , \varepsilon) \\
(& 5 , & 2 , & [1/2,2/7] , & 0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 8 , & \varepsilon , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 9 , & 3 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 10 , & 4{:}3 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 11 , & 7 , & [1/4,2/3] , & 2.0 , \varepsilon) \\
(& 12 , & \varepsilon , & [1/4,2/7] , & 2.0 , \varepsilon) \\
(& 3 , & \varepsilon , & [1/4,2/7] , & 2.0 , \varepsilon) \\
(& 4 , & \varepsilon , & [1/2,2/7] , & 0 , \varepsilon) \\
(& 5 , & 2 , & [1/2,2/7] , & 0 , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

```
(  9 ,   3 , [1/4,2/3] , 2.0    , ε )
( 10 , 4:3 , [1/4,2/3] , 2.0    , ε )
( 11 ,   7 , [1/4,2/3] , 2.0    , ε )
( 12 ,   ε , [1/4,2/7] , 2.0    , ε )
(  3 ,   ε , [1/4,2/7] , 2.0    , ε )
(  4 ,   ε , [1/2,2/7] , 0      , ε )
(  5 ,   2 , [1/2,2/7] , 0      , ε )
(  6 , 0:2 , [1/2,2/7] , 0      , ε )
```

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(9 ,	3 ,	[1/4,2/3] ,	2.0	, ε)
(10 ,	4:3 ,	[1/4,2/3] ,	2.0	, ε)
(11 ,	7 ,	[1/4,2/3] ,	2.0	, ε)
(12 ,	ε ,	[1/4,2/7] ,	2.0	, ε)
(3 ,	ε ,	[1/4,2/7] ,	2.0	, ε)
(4 ,	ε ,	[1/2,2/7] ,	0	, ε)
(5 ,	2 ,	[1/2,2/7] ,	0	, ε)
(6 ,	0:2 ,	[1/2,2/7] ,	0	, ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 10 , & 4{:}3 , [1/4{,}2/3] , & 2.0 \quad , \varepsilon) \\
(& 11 , & 7 , [1/4{,}2/3] , & 2.0 \quad , \varepsilon) \\
(& 12 , & \varepsilon , [1/4{,}2/7] , & 2.0 \quad , \varepsilon) \\
(& 3 , & \varepsilon , [1/4{,}2/7] , & 2.0 \quad , \varepsilon) \\
(& 4 , & \varepsilon , [1/2{,}2/7] , & 0 \quad , \varepsilon) \\
(& 5 , & 2 , [1/2{,}2/7] , & 0 \quad , \varepsilon) \\
(& 6 , & 0{:}2 , [1/2{,}2/7] , & 0 \quad , \varepsilon) \\
(& 7 , & 1 , [1/2{,}2/7] , & 0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(10 ,	4:3 ,	[1/4,2/3] ,	2.0	, ε)
(11 ,	7 ,	[1/4,2/3] ,	2.0	, ε)
(12 ,	ε ,	[1/4,2/7] ,	2.0	, ε)
(3 ,	ε ,	[1/4,2/7] ,	2.0	, ε)
(4 ,	ε ,	[1/2,2/7] ,	0	, ε)
(5 ,	2 ,	[1/2,2/7] ,	0	, ε)
(6 ,	0:2 ,	[1/2,2/7] ,	0	, ε)
(7 ,	1 ,	[1/2,2/7] ,	0	, ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;          6: NE;          11: STORE 2;
 2: STORE 2;        7: JMC 13;      12: JMP 3;
 3: READ 1;         8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;         9: LOAD 1;
 5: LIT 0;         10: ADD;
```

$$
\begin{array}{llll}
(& 11 , & 7 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/4,2/7] , 2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/4,2/7] , 2.0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 5 , & 2 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 6 , & 0{:}2 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 7 , & 1 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 11 , & 7 , [1/4,2/3] , 2.0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/4,2/7] , 2.0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/4,2/7] , 2.0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 5 , & 2 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 6 , & 0{:}2 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 7 , & 1 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 12 , & \varepsilon , [1/4,2/7] , & 2.0 \quad , \varepsilon) \\
(& 3 , & \varepsilon , [1/4,2/7] , & 2.0 \quad , \varepsilon) \\
(& 4 , & \varepsilon , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 5 , & 2 , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 6 , & 0{:}2 , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 7 , & 1 , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 8 , & \varepsilon , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 9 , & 7 , [1/2,2/7] , & 0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 12 , & \varepsilon , [1/4,2/7] , & 2.0 \quad , \varepsilon) \\
(& 3 , & \varepsilon , [1/4,2/7] , & 2.0 \quad , \varepsilon) \\
(& 4 , & \varepsilon , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 5 , & 2 , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 6 , & 0{:}2 , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 7 , & 1 , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 8 , & \varepsilon , [1/2,2/7] , & 0 \quad , \varepsilon) \\
(& 9 , & 7 , [1/2,2/7] , & 0 \quad , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(3 ,	ε ,	[1/4,2/7] ,	2.0	, ε)
(4 ,	ε ,	[1/2,2/7] ,	0	, ε)
(5 ,	2 ,	[1/2,2/7] ,	0	, ε)
(6 ,	0:2 ,	[1/2,2/7] ,	0	, ε)
(7 ,	1 ,	[1/2,2/7] ,	0	, ε)
(8 ,	ε ,	[1/2,2/7] ,	0	, ε)
(9 ,	7 ,	[1/2,2/7] ,	0	, ε)
(10 ,	2:7 ,	[1/2,2/7] ,	0	, ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(3 ,	ε ,	[1/4,2/7] ,	2.0	, ε)
(4 ,	ε ,	[1/2,2/7] ,	0	, ε)
(5 ,	2 ,	[1/2,2/7] ,	0	, ε)
(6 ,	0:2 ,	[1/2,2/7] ,	0	, ε)
(7 ,	1 ,	[1/2,2/7] ,	0	, ε)
(8 ,	ε ,	[1/2,2/7] ,	0	, ε)
(9 ,	7 ,	[1/2,2/7] ,	0	, ε)
(10 ,	2:7 ,	[1/2,2/7] ,	0	, ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 4 , & \varepsilon , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 5 , & 2 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 6 , & 0{:}2 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 7 , & 1 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , & [1/2,2/7] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 4 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 5 , & 2 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 6 , & 0{:}2 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 7 , & 1 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2,2/7] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 5 , & 2 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 6 , & 0{:}2 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 7 , & 1 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{lrlll}
(& 5 , & 2 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 6 , & 0{:}2 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 7 , & 1 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , & [1/2,2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , & [1/2,2/9] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 6 , & 0{:}2 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 7 , & 1 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/2{,}2/9] , 0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/2{,}2/9] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 6 , & 0{:}2 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 7 , & 1 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2{,}2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/2{,}2/9] , 0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/2{,}2/9] , 0 & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;
 2: STORE 2;
 3: READ 1;
 4: LOAD 1;
 5: LIT 0;
 6: NE;
 7: JMC 13;
 8: LOAD 2;
 9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 7 , & 1 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/0,2/9] , \varepsilon & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 7 , & 1 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 8 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/0,2/9] , \varepsilon & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 8 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/0,2/9] , \varepsilon & , \varepsilon) \\
(& 5 , & 0 , [1/0,2/9] , \varepsilon & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 8 , & \varepsilon , [1/2,2/7] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/0,2/9] , \varepsilon & , \varepsilon) \\
(& 5 , & 0 , [1/0,2/9] , \varepsilon & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(9 ,	7 ,	[1/2,2/7] ,	0	, ε)
(10 ,	2:7 ,	[1/2,2/7] ,	0	, ε)
(11 ,	9 ,	[1/2,2/7] ,	0	, ε)
(12 ,	ε ,	[1/2,2/9] ,	0	, ε)
(3 ,	ε ,	[1/2,2/9] ,	0	, ε)
(4 ,	ε ,	[1/0,2/9] ,	ε	, ε)
(5 ,	0 ,	[1/0,2/9] ,	ε	, ε)
(6 ,	0:0 ,	[1/0,2/9] ,	ε	, ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 9 , & 7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2,2/7] , 0 & , \varepsilon) \\
(& 12 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 3 , & \varepsilon , [1/2,2/9] , 0 & , \varepsilon) \\
(& 4 , & \varepsilon , [1/0,2/9] , \varepsilon & , \varepsilon) \\
(& 5 , & 0 , [1/0,2/9] , \varepsilon & , \varepsilon) \\
(& 6 , & 0{:}0 , [1/0,2/9] , \varepsilon & , \varepsilon)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
 1: LIT 0;        6: NE;          11: STORE 2;
 2: STORE 2;      7: JMC 13;      12: JMP 3;
 3: READ 1;       8: LOAD 2;      13: WRITE 2;
 4: LOAD 1;       9: LOAD 1;
 5: LIT 0;       10: ADD;
```

(10 ,	2:7 ,	[1/2,2/7] ,	0	, ε)
(11 ,	9 ,	[1/2,2/7] ,	0	, ε)
(12 ,	ε ,	[1/2,2/9] ,	0	, ε)
(3 ,	ε ,	[1/2,2/9] ,	0	, ε)
(4 ,	ε ,	[1/0,2/9] ,	ε	, ε)
(5 ,	0 ,	[1/0,2/9] ,	ε	, ε)
(6 ,	0:0 ,	[1/0,2/9] ,	ε	, ε)
(7 ,	0 ,	[1/0,2/9] ,	ε	, ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(10 ,	2:7 ,	[1/2,2/7] ,	0	, ε)
(11 ,	9 ,	[1/2,2/7] ,	0	, ε)
(12 ,	ε ,	[1/2,2/9] ,	0	, ε)
(3 ,	ε ,	[1/2,2/9] ,	0	, ε)
(4 ,	ε ,	[1/0,2/9] ,	ε	, ε)
(5 ,	0 ,	[1/0,2/9] ,	ε	, ε)
(6 ,	0:0 ,	[1/0,2/9] ,	ε	, ε)
(7 ,	0 ,	[1/0,2/9] ,	ε	, ε)

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(11 ,	9 ,	[1/2,2/7] ,	0	, ε)
(12 ,	ε ,	[1/2,2/9] ,	0	, ε)
(3 ,	ε ,	[1/2,2/9] ,	0	, ε)
(4 ,	ε ,	[1/0,2/9] ,	ε	, ε)
(5 ,	0 ,	[1/0,2/9] ,	ε	, ε)
(6 ,	0:0 ,	[1/0,2/9] ,	ε	, ε)
(7 ,	0 ,	[1/0,2/9] ,	ε	, ε)
(13 ,	ε ,	[1/0,2/9] ,	ε	, ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

(11 ,	9 ,	[1/2,2/7] ,	0	, ε)
(12 ,	ε ,	[1/2,2/9] ,	0	, ε)
(3 ,	ε ,	[1/2,2/9] ,	0	, ε)
(4 ,	ε ,	[1/0,2/9] ,	ε	, ε)
(5 ,	0 ,	[1/0,2/9] ,	ε	, ε)
(6 ,	0:0 ,	[1/0,2/9] ,	ε	, ε)
(7 ,	0 ,	[1/0,2/9] ,	ε	, ε)
(13 ,	ε ,	[1/0,2/9] ,	ε	, ε)

Weiteres Beispiel

Aufgabe:
- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llll}
(& 12 , & \varepsilon , [1/2,2/9] , & 0 , \varepsilon) \\
(& 3 , & \varepsilon , [1/2,2/9] , & 0 , \varepsilon) \\
(& 4 , & \varepsilon , [1/0,2/9] , & \varepsilon , \varepsilon) \\
(& 5 , & 0 , [1/0,2/9] , & \varepsilon , \varepsilon) \\
(& 6 , & 0{:}0 , [1/0,2/9] , & \varepsilon , \varepsilon) \\
(& 7 , & 0 , [1/0,2/9] , & \varepsilon , \varepsilon) \\
(& 13 , & \varepsilon , [1/0,2/9] , & \varepsilon , \varepsilon) \\
(& 14 , & \varepsilon , [1/0,2/9] , & \varepsilon , 9)
\end{array}
$$

Weiteres Beispiel

Aufgabe:

- Einlesen einer Folge (Ende: 0) vom Eingabeband
- Ausgabe der Gesamtsumme auf Ausgabeband

```
1: LIT 0;
2: STORE 2;
3: READ 1;
4: LOAD 1;
5: LIT 0;
6: NE;
7: JMC 13;
8: LOAD 2;
9: LOAD 1;
10: ADD;
11: STORE 2;
12: JMP 3;
13: WRITE 2;
```

$$
\begin{array}{llllll}
(& 12 , & \varepsilon , & [1/2,2/9] , & 0 & , \varepsilon) \\
(& 3 , & \varepsilon , & [1/2,2/9] , & 0 & , \varepsilon) \\
(& 4 , & \varepsilon , & [1/0,2/9] , & \varepsilon & , \varepsilon) \\
(& 5 , & 0 , & [1/0,2/9] , & \varepsilon & , \varepsilon) \\
(& 6 , & 0{:}0 , & [1/0,2/9] , & \varepsilon & , \varepsilon) \\
(& 7 , & 0 , & [1/0,2/9] , & \varepsilon & , \varepsilon) \\
(& 13 , & \varepsilon , & [1/0,2/9] , & \varepsilon & , \varepsilon) \\
(& 14 , & \varepsilon , & [1/0,2/9] , & \varepsilon & , 9)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;
2: READ 1;
3: LOAD 1;
4: LIT 0;
5: NE;
6: JMC 10;
7: LOAD 1;
8: ADD;
9: JMP 2;
10: STORE 1;
11: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(1 , ε , [] , 3.4.2.0 , ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(	1 ,	ε , []	, 3.4.2.0	, ε)
(	2 ,	0 , []	, 3.4.2.0	, ε)
(	3 ,	0 , [1/3]	, 4.2.0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

1: LIT 0;	5: NE;	9: JMP 2;
2: READ 1;	6: JMC 10;	10: STORE 1;
3: LOAD 1;	7: LOAD 1;	11: WRITE 1;
4: LIT 0;	8: ADD;	

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)
(3 , 0 , [1/3] , 4.2.0 , ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(	1 ,	ε , []	, 3.4.2.0	, ε)
(	2 ,	0 , []	, 3.4.2.0	, ε)
(	3 ,	0 , [1/3]	, 4.2.0	, ε)
(	4 ,	3:0 , [1/3]	, 4.2.0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(8 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(8 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(8 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(9 ,	3 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε	, []	, 3.4.2.0	, ε)
(2 ,	0	, []	, 3.4.2.0	, ε)
(3 ,	0	, [1/3]	, 4.2.0	, ε)
(4 ,	3:0	, [1/3]	, 4.2.0	, ε)
(5 ,	0:3:0	, [1/3]	, 4.2.0	, ε)
(6 ,	1:0	, [1/3]	, 4.2.0	, ε)
(7 ,	0	, [1/3]	, 4.2.0	, ε)
(8 ,	3:0	, [1/3]	, 4.2.0	, ε)
(9 ,	3	, [1/3]	, 4.2.0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(8 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(9 ,	3 ,	[1/3]	, 4.2.0 ,	ε)
(2 ,	3 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(1 ,	ε ,	[]	, 3.4.2.0 ,	ε)
(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(8 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(9 ,	3 ,	[1/3]	, 4.2.0 ,	ε)
(2 ,	3 ,	[1/3]	, 4.2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(8 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(9 ,	3 ,	[1/3]	, 4.2.0 ,	ε)
(2 ,	3 ,	[1/3]	, 4.2.0 ,	ε)
(3 ,	3 ,	[1/4]	, 2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(2 ,	0 ,	[]	, 3.4.2.0 ,	ε)
(3 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(4 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(5 ,	0:3:0 ,	[1/3]	, 4.2.0 ,	ε)
(6 ,	1:0 ,	[1/3]	, 4.2.0 ,	ε)
(7 ,	0 ,	[1/3]	, 4.2.0 ,	ε)
(8 ,	3:0 ,	[1/3]	, 4.2.0 ,	ε)
(9 ,	3 ,	[1/3]	, 4.2.0 ,	ε)
(2 ,	3 ,	[1/3]	, 4.2.0 ,	ε)
(3 ,	3 ,	[1/4]	, 2.0 ,	ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(3 ,	0	, [1/3],	4.2.0	, ε)
(4 ,	3:0	, [1/3],	4.2.0	, ε)
(5 ,	0:3:0	, [1/3],	4.2.0	, ε)
(6 ,	1:0	, [1/3],	4.2.0	, ε)
(7 ,	0	, [1/3],	4.2.0	, ε)
(8 ,	3:0	, [1/3],	4.2.0	, ε)
(9 ,	3	, [1/3],	4.2.0	, ε)
(2 ,	3	, [1/3],	4.2.0	, ε)
(3 ,	3	, [1/4],	2.0	, ε)
(4 ,	4:3	, [1/4],	2.0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{lrlll}
(& 3 , & 0 , & [1/3], 4.2.0 & , \varepsilon) \\
(& 4 , & 3{:}0 , & [1/3], 4.2.0 & , \varepsilon) \\
(& 5 , & 0{:}3{:}0 , & [1/3], 4.2.0 & , \varepsilon) \\
(& 6 , & 1{:}0 , & [1/3], 4.2.0 & , \varepsilon) \\
(& 7 , & 0 , & [1/3], 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 4 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 5 , & 0{:}3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 6 , & 1{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 7 , & 0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 4 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 5 , & 0{:}3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 6 , & 1{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 7 , & 0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 5 , & 0{:}3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 6 , & 1{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 7 , & 0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;
2: READ 1;
3: LOAD 1;
4: LIT 0;
5: NE;
6: JMC 10;
7: LOAD 1;
8: ADD;
9: JMP 2;
10: STORE 1;
11: WRITE 1;
```

$$
\begin{array}{llllll}
(& 5 , & 0{:}3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 6 , & 1{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 7 , & 0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;
2: READ 1;
3: LOAD 1;
4: LIT 0;
5: NE;
6: JMC 10;
7: LOAD 1;
8: ADD;
9: JMP 2;
10: STORE 1;
11: WRITE 1;
```

$$
\begin{array}{llllll}
(& 6 , & 1{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 7 , & 0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 6 , & 1{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 7 , & 0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 7 , & 0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 7 , & 0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 8 , & 3{:}0 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 9 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 2 , & 3 , & [1/3], & 4.2.0 & , \varepsilon) \\
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(9 ,	3 ,	[1/3],	4.2.0	, ε)
(2 ,	3 ,	[1/3],	4.2.0	, ε)
(3 ,	3 ,	[1/4],	2.0	, ε)
(4 ,	4:3 ,	[1/4],	2.0	, ε)
(5 ,	0:4:3 ,	[1/4],	2.0	, ε)
(6 ,	1:3 ,	[1/4],	2.0	, ε)
(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llll}
(& 9 , & 3 , [1/3], & 4.2.0 , \varepsilon) \\
(& 2 , & 3 , [1/3], & 4.2.0 , \varepsilon) \\
(& 3 , & 3 , [1/4], & 2.0 , \varepsilon) \\
(& 4 , & 4{:}3 , [1/4], & 2.0 , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , [1/4], & 2.0 , \varepsilon) \\
(& 6 , & 1{:}3 , [1/4], & 2.0 , \varepsilon) \\
(& 7 , & 3 , [1/4], & 2.0 , \varepsilon) \\
(& 8 , & 4{:}3 , [1/4], & 2.0 , \varepsilon) \\
(& 9 , & 7 , [1/4], & 2.0 , \varepsilon) \\
(& 2 , & 7 , [1/4], & 2.0 , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(2 ,	3 ,	[1/3],	4.2.0	, ε)
(3 ,	3 ,	[1/4],	2.0	, ε)
(4 ,	4:3 ,	[1/4],	2.0	, ε)
(5 ,	0:4:3 ,	[1/4],	2.0	, ε)
(6 ,	1:3 ,	[1/4],	2.0	, ε)
(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(2 ,	3 ,	[1/3],	4.2.0	, ε)
(3 ,	3 ,	[1/4],	2.0	, ε)
(4 ,	4:3 ,	[1/4],	2.0	, ε)
(5 ,	0:4:3 ,	[1/4],	2.0	, ε)
(6 ,	1:3 ,	[1/4],	2.0	, ε)
(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(3 ,	3 ,	[1/4],	2.0	, ε)
(4 ,	4:3 ,	[1/4],	2.0	, ε)
(5 ,	0:4:3 ,	[1/4],	2.0	, ε)
(6 ,	1:3 ,	[1/4],	2.0	, ε)
(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)
(4 ,	2:7 ,	[1/2],	0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 3 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 4 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 2 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

1: LIT 0;	5: NE;	9: JMP 2;
2: READ 1;	6: JMC 10;	10: STORE 1;
3: LOAD 1;	7: LOAD 1;	11: WRITE 1;
4: LIT 0;	8: ADD;	

(4 ,	4:3 ,	[1/4],	2.0	, ε)
(5 ,	0:4:3 ,	[1/4],	2.0	, ε)
(6 ,	1:3 ,	[1/4],	2.0	, ε)
(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)
(4 ,	2:7 ,	[1/2],	0	, ε)
(5 ,	0:2:7 ,	[1/2],	0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(4 ,	4:3 ,	[1/4],	2.0	, ε)
(5 ,	0:4:3 ,	[1/4],	2.0	, ε)
(6 ,	1:3 ,	[1/4],	2.0	, ε)
(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)
(4 ,	2:7 ,	[1/2],	0	, ε)
(5 ,	0:2:7 ,	[1/2],	0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 5 , & 0{:}4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 2 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(5 ,	0:4:3 ,	[1/4],	2.0	, ε)
(6 ,	1:3 ,	[1/4],	2.0	, ε)
(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)
(4 ,	2:7 ,	[1/2],	0	, ε)
(5 ,	0:2:7 ,	[1/2],	0	, ε)
(6 ,	1:7 ,	[1/2],	0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 2 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 6 , & 1{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 7 , & 3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 2 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)
(4 ,	2:7 ,	[1/2],	0	, ε)
(5 ,	0:2:7 ,	[1/2],	0	, ε)
(6 ,	1:7 ,	[1/2],	0	, ε)
(7 ,	7 ,	[1/2],	0	, ε)
(8 ,	2:7 ,	[1/2],	0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(7 ,	3 ,	[1/4],	2.0	, ε)
(8 ,	4:3 ,	[1/4],	2.0	, ε)
(9 ,	7 ,	[1/4],	2.0	, ε)
(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)
(4 ,	2:7 ,	[1/2],	0	, ε)
(5 ,	0:2:7 ,	[1/2],	0	, ε)
(6 ,	1:7 ,	[1/2],	0	, ε)
(7 ,	7 ,	[1/2],	0	, ε)
(8 ,	2:7 ,	[1/2],	0	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 8 , & 4{:}3 , & [1/4], & 2.0 & , \varepsilon) \\
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 2 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], & 0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{lrlll}
(& 8 , & 4{:}3 , & [1/4], 2.0 & , \varepsilon) \\
(& 9 , & 7 , & [1/4], 2.0 & , \varepsilon) \\
(& 2 , & 7 , & [1/4], 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], 0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 2 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 2 , & 9 , & [1/2], & 0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 9 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 2 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 2 , & 9 , & [1/2], & 0 & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llllll}
(& 2 , & 7 , & [1/4], & 2.0 & , \varepsilon) \\
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 2 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 3 , & 9 , & [1/0], & \varepsilon & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(2 ,	7 ,	[1/4],	2.0	, ε)
(3 ,	7 ,	[1/2],	0	, ε)
(4 ,	2:7 ,	[1/2],	0	, ε)
(5 ,	0:2:7 ,	[1/2],	0	, ε)
(6 ,	1:7 ,	[1/2],	0	, ε)
(7 ,	7 ,	[1/2],	0	, ε)
(8 ,	2:7 ,	[1/2],	0	, ε)
(9 ,	9 ,	[1/2],	0	, ε)
(2 ,	9 ,	[1/2],	0	, ε)
(3 ,	9 ,	[1/0],	ε	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(3 ,	7 ,	[1/2],	0	, ε)
(4 ,	2:7 ,	[1/2],	0	, ε)
(5 ,	0:2:7 ,	[1/2],	0	, ε)
(6 ,	1:7 ,	[1/2],	0	, ε)
(7 ,	7 ,	[1/2],	0	, ε)
(8 ,	2:7 ,	[1/2],	0	, ε)
(9 ,	9 ,	[1/2],	0	, ε)
(2 ,	9 ,	[1/2],	0	, ε)
(3 ,	9 ,	[1/0],	ε	, ε)
(4 ,	0:9 ,	[1/0],	ε	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

1: LIT 0;	5: NE;	9: JMP 2;
2: READ 1;	6: JMC 10;	10: STORE 1;
3: LOAD 1;	7: LOAD 1;	11: WRITE 1;
4: LIT 0;	8: ADD;	

$$
\begin{array}{llllll}
(& 3 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 2 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 3 , & 9 , & [1/0], & \varepsilon & , \varepsilon) \\
(& 4 , & 0{:}9 , & [1/0], & \varepsilon & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 2 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 3 , & 9 , & [1/0], & \varepsilon & , \varepsilon) \\
(& 4 , & 0{:}9 , & [1/0], & \varepsilon & , \varepsilon) \\
(& 5 , & 0{:}0{:}9 , & [1/0], & \varepsilon & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 4 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 2 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 3 , & 9 , & [1/0], & \varepsilon & , \varepsilon) \\
(& 4 , & 0{:}9 , & [1/0], & \varepsilon & , \varepsilon) \\
(& 5 , & 0{:}0{:}9 , & [1/0], & \varepsilon & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

$$
\begin{array}{llllll}
(& 5 , & 0{:}2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 6 , & 1{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 7 , & 7 , & [1/2], & 0 & , \varepsilon) \\
(& 8 , & 2{:}7 , & [1/2], & 0 & , \varepsilon) \\
(& 9 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 2 , & 9 , & [1/2], & 0 & , \varepsilon) \\
(& 3 , & 9 , & [1/0], & \varepsilon & , \varepsilon) \\
(& 4 , & 0{:}9 , & [1/0], & \varepsilon & , \varepsilon) \\
(& 5 , & 0{:}0{:}9 , & [1/0], & \varepsilon & , \varepsilon) \\
(& 6 , & 0{:}9 , & [1/0], & \varepsilon & , \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(5 ,	0:2:7 ,	[1/2],	0	, ε)
(6 ,	1:7 ,	[1/2],	0	, ε)
(7 ,	7 ,	[1/2],	0	, ε)
(8 ,	2:7 ,	[1/2],	0	, ε)
(9 ,	9 ,	[1/2],	0	, ε)
(2 ,	9 ,	[1/2],	0	, ε)
(3 ,	9 ,	[1/0],	ε	, ε)
(4 ,	0:9 ,	[1/0],	ε	, ε)
(5 ,	0:0:9 ,	[1/0],	ε	, ε)
(6 ,	0:9 ,	[1/0],	ε	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(6 ,	1:7 ,	[1/2],	0	, ε)
(7 ,	7 ,	[1/2],	0	, ε)
(8 ,	2:7 ,	[1/2],	0	, ε)
(9 ,	9 ,	[1/2],	0	, ε)
(2 ,	9 ,	[1/2],	0	, ε)
(3 ,	9 ,	[1/0],	ε	, ε)
(4 ,	0:9 ,	[1/0],	ε	, ε)
(5 ,	0:0:9 ,	[1/0],	ε	, ε)
(6 ,	0:9 ,	[1/0],	ε	, ε)
(10 ,	9 ,	[1/0],	ε	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(6 ,	1:7 ,	[1/2],	0	, ε)
(7 ,	7 ,	[1/2],	0	, ε)
(8 ,	2:7 ,	[1/2],	0	, ε)
(9 ,	9 ,	[1/2],	0	, ε)
(2 ,	9 ,	[1/2],	0	, ε)
(3 ,	9 ,	[1/0],	ε	, ε)
(4 ,	0:9 ,	[1/0],	ε	, ε)
(5 ,	0:0:9 ,	[1/0],	ε	, ε)
(6 ,	0:9 ,	[1/0],	ε	, ε)
(10 ,	9 ,	[1/0],	ε	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

$$
\begin{array}{llll}
(& 7, & 7, & [1/2], 0, \varepsilon) \\
(& 8, & 2{:}7, & [1/2], 0, \varepsilon) \\
(& 9, & 9, & [1/2], 0, \varepsilon) \\
(& 2, & 9, & [1/2], 0, \varepsilon) \\
(& 3, & 9, & [1/0], \varepsilon, \varepsilon) \\
(& 4, & 0{:}9, & [1/0], \varepsilon, \varepsilon) \\
(& 5, & 0{:}0{:}9, & [1/0], \varepsilon, \varepsilon) \\
(& 6, & 0{:}9, & [1/0], \varepsilon, \varepsilon) \\
(& 10, & 9, & [1/0], \varepsilon, \varepsilon) \\
(& 11, & \varepsilon, & [1/9], \varepsilon, \varepsilon)
\end{array}
$$

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(7 ,	7 ,	[1/2],	0	, ε)
(8 ,	2:7 ,	[1/2],	0	, ε)
(9 ,	9 ,	[1/2],	0	, ε)
(2 ,	9 ,	[1/2],	0	, ε)
(3 ,	9 ,	[1/0],	ε	, ε)
(4 ,	0:9 ,	[1/0],	ε	, ε)
(5 ,	0:0:9 ,	[1/0],	ε	, ε)
(6 ,	0:9 ,	[1/0],	ε	, ε)
(10 ,	9 ,	[1/0],	ε	, ε)
(11 ,	ε ,	[1/9],	ε	, ε)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;          5: NE;          9: JMP 2;
2: READ 1;         6: JMC 10;     10: STORE 1;
3: LOAD 1;         7: LOAD 1;     11: WRITE 1;
4: LIT 0;          8: ADD;
```

(8 ,	2:7 ,	[1/2],	0	, ε)
(9 ,	9 ,	[1/2],	0	, ε)
(2 ,	9 ,	[1/2],	0	, ε)
(3 ,	9 ,	[1/0],	ε	, ε)
(4 ,	0:9 ,	[1/0],	ε	, ε)
(5 ,	0:0:9 ,	[1/0],	ε	, ε)
(6 ,	0:9 ,	[1/0],	ε	, ε)
(10 ,	9 ,	[1/0],	ε	, ε)
(11 ,	ε ,	[1/9],	ε	, ε)
(12 ,	ε ,	[1/9],	ε	, 9)

Optimierung

Idee: statt Ablage im HS, Zwischensumme auf DK lassen

```
1: LIT 0;        5: NE;          9: JMP 2;
2: READ 1;       6: JMC 10;     10: STORE 1;
3: LOAD 1;       7: LOAD 1;     11: WRITE 1;
4: LIT 0;        8: ADD;
```

(8 ,	2:7 ,	[1/2],	0	, ε)
(9 ,	9 ,	[1/2],	0	, ε)
(2 ,	9 ,	[1/2],	0	, ε)
(3 ,	9 ,	[1/0],	ε	, ε)
(4 ,	0:9 ,	[1/0],	ε	, ε)
(5 ,	0:0:9 ,	[1/0],	ε	, ε)
(6 ,	0:9 ,	[1/0],	ε	, ε)
(10 ,	9 ,	[1/0],	ε	, ε)
(11 ,	ε ,	[1/9],	ε	, ε)
(12 ,	ε ,	[1/9],	ε	, 9)

Erweiterung

Aufgabe:

- statt Summe, nun Durchschnitt zu berechnen

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
1: LIT 0;
2: STORE 2;
3: LIT 0;
4: READ 1;
5: LOAD 1;
6: LIT 0;
7: NE;
8: JMC 16;
9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)
(3 , ε , [2/0] , 3.4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)
(3 , ε , [2/0] , 3.4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)
(3 , ε , [2/0] , 3.4.2.0 , ε)
(4 , 0 , [2/0] , 3.4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 , ε , [] , 3.4.2.0 , ε)
(2 , 0 , [] , 3.4.2.0 , ε)
(3 , ε , [2/0] , 3.4.2.0 , ε)
(4 , 0 , [2/0] , 3.4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	0 , [2/0]	, 3.4.2.0 ,	ε)
(	5 ,	0 , [1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	0 , [2/0]	, 3.4.2.0 ,	ε)
(	5 ,	0 , [1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	0 , [2/0]	, 3.4.2.0 ,	ε)
(	5 ,	0 , [1/3,2/0]	, 4.2.0 ,	ε)
(	6 ,	3:0 , [1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	1 ,	ε , []	, 3.4.2.0 ,	ε)
(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	0 , [2/0]	, 3.4.2.0 ,	ε)
(	5 ,	0 , [1/3,2/0]	, 4.2.0 ,	ε)
(	6 ,	3:0 , [1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 ,	ε , []	, 3.4.2.0 ,	ε)
(2 ,	0 , []	, 3.4.2.0 ,	ε)
(3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(4 ,	0 , [2/0]	, 3.4.2.0 ,	ε)
(5 ,	0 , [1/3,2/0]	, 4.2.0 ,	ε)
(6 ,	3:0 , [1/3,2/0]	, 4.2.0 ,	ε)
(7 ,	0:3:0 , [1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
1: LIT 0;
2: STORE 2;
3: LIT 0;
4: READ 1;
5: LOAD 1;
6: LIT 0;
7: NE;
8: JMC 16;
9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(1 ,	ε , []	, 3.4.2.0 ,	ε)
(2 ,	0 , []	, 3.4.2.0 ,	ε)
(3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(4 ,	0 , [2/0]	, 3.4.2.0 ,	ε)
(5 ,	0 , [1/3,2/0]	, 4.2.0 ,	ε)
(6 ,	3:0 , [1/3,2/0]	, 4.2.0 ,	ε)
(7 ,	0:3:0 , [1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	0 , [2/0]	, 3.4.2.0 ,	ε)
(	5 ,	0 , [1/3,2/0]	, 4.2.0 ,	ε)
(	6 ,	3:0 , [1/3,2/0]	, 4.2.0 ,	ε)
(	7 ,	0:3:0 , [1/3,2/0]	, 4.2.0 ,	ε)
(	8 ,	1:0 , [1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	2 ,	0 , []	, 3.4.2.0 ,	ε)
(	3 ,	ε , [2/0]	, 3.4.2.0 ,	ε)
(	4 ,	0 , [2/0]	, 3.4.2.0 ,	ε)
(	5 ,	0 , [1/3,2/0]	, 4.2.0 ,	ε)
(	6 ,	3:0 , [1/3,2/0]	, 4.2.0 ,	ε)
(	7 ,	0:3:0 , [1/3,2/0]	, 4.2.0 ,	ε)
(	8 ,	1:0 , [1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	3 ,	ε ,	[2/0] ,	3.4.2.0 , ε)
(	4 ,	0 ,	[2/0] ,	3.4.2.0 , ε)
(	5 ,	0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	6 ,	3:0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	7 ,	0:3:0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	8 ,	1:0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	9 ,	0 ,	[1/3,2/0] ,	4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	3 ,	ε ,	[2/0] ,	3.4.2.0 , ε)
(	4 ,	0 ,	[2/0] ,	3.4.2.0 , ε)
(	5 ,	0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	6 ,	3:0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	7 ,	0:3:0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	8 ,	1:0 ,	[1/3,2/0] ,	4.2.0 , ε)
(	9 ,	0 ,	[1/3,2/0] ,	4.2.0 , ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(4 ,	0 ,	[2/0]	, 3.4.2.0 ,	ε)
(5 ,	0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(6 ,	3:0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(7 ,	0:3:0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(8 ,	1:0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(9 ,	0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(10 ,	3:0 ,	[1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(4 ,	0 ,	[2/0]	, 3.4.2.0 ,	ε)
(5 ,	0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(6 ,	3:0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(7 ,	0:3:0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(8 ,	1:0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(9 ,	0 ,	[1/3,2/0]	, 4.2.0 ,	ε)
(10 ,	3:0 ,	[1/3,2/0]	, 4.2.0 ,	ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	5 ,	0 ,	[1/3,2/0] , 4.2.0	, ε)
(	6 ,	3:0 ,	[1/3,2/0] , 4.2.0	, ε)
(	7 ,	0:3:0 ,	[1/3,2/0] , 4.2.0	, ε)
(	8 ,	1:0 ,	[1/3,2/0] , 4.2.0	, ε)
(	9 ,	0 ,	[1/3,2/0] , 4.2.0	, ε)
(	10 ,	3:0 ,	[1/3,2/0] , 4.2.0	, ε)
(	11 ,	3 ,	[1/3,2/0] , 4.2.0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	5 ,	0 ,	[1/3,2/0] , 4.2.0	, ε)
(	6 ,	3:0 ,	[1/3,2/0] , 4.2.0	, ε)
(	7 ,	0:3:0 ,	[1/3,2/0] , 4.2.0	, ε)
(	8 ,	1:0 ,	[1/3,2/0] , 4.2.0	, ε)
(	9 ,	0 ,	[1/3,2/0] , 4.2.0	, ε)
(	10 ,	3:0 ,	[1/3,2/0] , 4.2.0	, ε)
(	11 ,	3 ,	[1/3,2/0] , 4.2.0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llllll}
(& 6 , & 3{:}0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 7 , & 0{:}3{:}0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 8 , & 1{:}0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 9 , & 0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 10 , & 3{:}0 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 11 , & 3 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon) \\
(& 12 , & 0{:}3 , & [1/3{,}2/0] , & 4.2.0 & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
(  6 ,   3:0 , [1/3,2/0] , 4.2.0   , ε )
(  7 , 0:3:0 , [1/3,2/0] , 4.2.0   , ε )
(  8 ,   1:0 , [1/3,2/0] , 4.2.0   , ε )
(  9 ,     0 , [1/3,2/0] , 4.2.0   , ε )
( 10 ,   3:0 , [1/3,2/0] , 4.2.0   , ε )
( 11 ,     3 , [1/3,2/0] , 4.2.0   , ε )
( 12 ,   0:3 , [1/3,2/0] , 4.2.0   , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(7 , 0:3:0 , [1/3,2/0] , 4.2.0 , ε)
(8 , 1:0 , [1/3,2/0] , 4.2.0 , ε)
(9 , 0 , [1/3,2/0] , 4.2.0 , ε)
(10 , 3:0 , [1/3,2/0] , 4.2.0 , ε)
(11 , 3 , [1/3,2/0] , 4.2.0 , ε)
(12 , 0:3 , [1/3,2/0] , 4.2.0 , ε)
(13 , 1:0:3 , [1/3,2/0] , 4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(7 , 0:3:0 , [1/3,2/0] , 4.2.0 , ε)
(8 , 1:0 , [1/3,2/0] , 4.2.0 , ε)
(9 , 0 , [1/3,2/0] , 4.2.0 , ε)
(10 , 3:0 , [1/3,2/0] , 4.2.0 , ε)
(11 , 3 , [1/3,2/0] , 4.2.0 , ε)
(12 , 0:3 , [1/3,2/0] , 4.2.0 , ε)
(13 , 1:0:3 , [1/3,2/0] , 4.2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 8 , & 1{:}0 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 9 , & 0 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 10 , & 3{:}0 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & 0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
(  8 ,   1:0 , [1/3,2/0] , 4.2.0   , ε )
(  9 ,     0 , [1/3,2/0] , 4.2.0   , ε )
( 10 ,   3:0 , [1/3,2/0] , 4.2.0   , ε )
( 11 ,     3 , [1/3,2/0] , 4.2.0   , ε )
( 12 ,   0:3 , [1/3,2/0] , 4.2.0   , ε )
( 13 , 1:0:3 , [1/3,2/0] , 4.2.0   , ε )
( 14 ,   1:3 , [1/3,2/0] , 4.2.0   , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 9 , & 0 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 10 , & 3{:}0 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & 0{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3,2/1] , & 4.2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 9 , & 0 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 10 , & 3{:}0 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & 0{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3,2/1] , & 4.2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 10 , & 3{:}0 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & 0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 10 , & 3{:}0 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 11 , & 3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & 0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 11 , & 3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & 0{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3,2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3,2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , [1/4,2/1] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 11 , & 3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 12 , & 0{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3,2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3,2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3,2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , [1/4,2/1] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 12 , & 0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon) \\
(& 6 , & 4{:}3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 12 , & 0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 13 , & 1{:}0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon) \\
(& 6 , & 4{:}3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 13 , & 1{:}0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon) \\
(& 6 , & 4{:}3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon) \\
(& 7 , & 0{:}4{:}3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
1: LIT 0;
2: STORE 2;
3: LIT 0;
4: READ 1;
5: LOAD 1;
6: LIT 0;
7: NE;
8: JMC 16;
9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 13 , & 1{:}0{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 14 , & 1{:}3 , [1/3{,}2/0] , & 4.2.0 \quad , \varepsilon) \\
(& 15 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3{,}2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon) \\
(& 6 , & 4{:}3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon) \\
(& 7 , & 0{:}4{:}3 , [1/4{,}2/1] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(14 , 1:3 , [1/3,2/0] , 4.2.0 , ε)
(15 , 3 , [1/3,2/1] , 4.2.0 , ε)
(4 , 3 , [1/3,2/1] , 4.2.0 , ε)
(5 , 3 , [1/4,2/1] , 2.0 , ε)
(6 , 4:3 , [1/4,2/1] , 2.0 , ε)
(7 , 0:4:3 , [1/4,2/1] , 2.0 , ε)
(8 , 1:3 , [1/4,2/1] , 2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(14 ,	1:3 ,	[1/3,2/0] ,	4.2.0	, ε)
(15 ,	3 ,	[1/3,2/1] ,	4.2.0	, ε)
(4 ,	3 ,	[1/3,2/1] ,	4.2.0	, ε)
(5 ,	3 ,	[1/4,2/1] ,	2.0	, ε)
(6 ,	4:3 ,	[1/4,2/1] ,	2.0	, ε)
(7 ,	0:4:3 ,	[1/4,2/1] ,	2.0	, ε)
(8 ,	1:3 ,	[1/4,2/1] ,	2.0	, ε)

Erweiterung

Aufgabe:

- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 15 , & 3 , [1/3,2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 4 , & 3 , [1/3,2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 6 , & 4{:}3 , [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 7 , & 0{:}4{:}3 , [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 8 , & 1{:}3 , [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 9 , & 3 , [1/4,2/1] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 15 , & 3 , [1/3,2/1] , & 4.2.0 , \varepsilon) \\
(& 4 , & 3 , [1/3,2/1] , & 4.2.0 , \varepsilon) \\
(& 5 , & 3 , [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 6 , & 4:3 , [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 7 , & 0:4:3 , [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 8 , & 1:3 , [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 9 , & 3 , [1/4,2/1] , & 2.0 , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{lrlll}
(& 4 , & 3 , & [1/3,2/1] , & 4.2.0 \quad , \varepsilon) \\
(& 5 , & 3 , & [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 6 , & 4{:}3 , & [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 7 , & 0{:}4{:}3 , & [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 8 , & 1{:}3 , & [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 9 , & 3 , & [1/4,2/1] , & 2.0 \quad , \varepsilon) \\
(& 10 , & 4{:}3 , & [1/4,2/1] , & 2.0 \quad , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	4 ,	3 ,	[1/3,2/1] , 4.2.0	, ε)
(	5 ,	3 ,	[1/4,2/1] , 2.0	, ε)
(	6 ,	4:3 ,	[1/4,2/1] , 2.0	, ε)
(	7 ,	0:4:3 ,	[1/4,2/1] , 2.0	, ε)
(	8 ,	1:3 ,	[1/4,2/1] , 2.0	, ε)
(	9 ,	3 ,	[1/4,2/1] , 2.0	, ε)
(	10 ,	4:3 ,	[1/4,2/1] , 2.0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{lrlll}
(& 5 , & 3 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 6 , & 4:3 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 7 , & 0:4:3 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 8 , & 1:3 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 9 , & 3 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 10 , & 4:3 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 11 , & 7 , & [1/4,2/1] , & 2.0 , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	5 ,	3 ,	[1/4,2/1] , 2.0	, ε)
(	6 ,	4:3 ,	[1/4,2/1] , 2.0	, ε)
(	7 ,	0:4:3 ,	[1/4,2/1] , 2.0	, ε)
(	8 ,	1:3 ,	[1/4,2/1] , 2.0	, ε)
(	9 ,	3 ,	[1/4,2/1] , 2.0	, ε)
(	10 ,	4:3 ,	[1/4,2/1] , 2.0	, ε)
(	11 ,	7 ,	[1/4,2/1] , 2.0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	6 ,	4:3 ,	[1/4,2/1] , 2.0	, ε)
(	7 ,	0:4:3 ,	[1/4,2/1] , 2.0	, ε)
(	8 ,	1:3 ,	[1/4,2/1] , 2.0	, ε)
(	9 ,	3 ,	[1/4,2/1] , 2.0	, ε)
(	10 ,	4:3 ,	[1/4,2/1] , 2.0	, ε)
(	11 ,	7 ,	[1/4,2/1] , 2.0	, ε)
(	12 ,	1:7 ,	[1/4,2/1] , 2.0	, ε)

Erweiterung

Aufgabe:

- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
(  6 ,   4:3 , [1/4,2/1] , 2.0    , ε )
(  7 , 0:4:3 , [1/4,2/1] , 2.0    , ε )
(  8 ,   1:3 , [1/4,2/1] , 2.0    , ε )
(  9 ,     3 , [1/4,2/1] , 2.0    , ε )
( 10 ,   4:3 , [1/4,2/1] , 2.0    , ε )
( 11 ,     7 , [1/4,2/1] , 2.0    , ε )
( 12 ,   1:7 , [1/4,2/1] , 2.0    , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
(  7 , 0:4:3 , [1/4,2/1] , 2.0    , ε )
(  8 ,   1:3 , [1/4,2/1] , 2.0    , ε )
(  9 ,     3 , [1/4,2/1] , 2.0    , ε )
( 10 ,   4:3 , [1/4,2/1] , 2.0    , ε )
( 11 ,     7 , [1/4,2/1] , 2.0    , ε )
( 12 ,   1:7 , [1/4,2/1] , 2.0    , ε )
( 13 , 1:1:7 , [1/4,2/1] , 2.0    , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
(  7 , 0:4:3 , [1/4,2/1] , 2.0    , ε )
(  8 ,   1:3 , [1/4,2/1] , 2.0    , ε )
(  9 ,     3 , [1/4,2/1] , 2.0    , ε )
( 10 ,   4:3 , [1/4,2/1] , 2.0    , ε )
( 11 ,     7 , [1/4,2/1] , 2.0    , ε )
( 12 ,   1:7 , [1/4,2/1] , 2.0    , ε )
( 13 , 1:1:7 , [1/4,2/1] , 2.0    , ε )
```

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
(  8 ,   1:3 , [1/4,2/1] , 2.0    , ε )
(  9 ,     3 , [1/4,2/1] , 2.0    , ε )
( 10 ,   4:3 , [1/4,2/1] , 2.0    , ε )
( 11 ,     7 , [1/4,2/1] , 2.0    , ε )
( 12 ,   1:7 , [1/4,2/1] , 2.0    , ε )
( 13 , 1:1:7 , [1/4,2/1] , 2.0    , ε )
( 14 ,   2:7 , [1/4,2/1] , 2.0    , ε )
```

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
(  8 ,   1:3 , [1/4,2/1] , 2.0    , ε )
(  9 ,     3 , [1/4,2/1] , 2.0    , ε )
( 10 ,   4:3 , [1/4,2/1] , 2.0    , ε )
( 11 ,     7 , [1/4,2/1] , 2.0    , ε )
( 12 ,   1:7 , [1/4,2/1] , 2.0    , ε )
( 13 , 1:1:7 , [1/4,2/1] , 2.0    , ε )
( 14 ,   2:7 , [1/4,2/1] , 2.0    , ε )
```

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{lrlll}
(& 9 , & 3 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 10 , & 4:3 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 11 , & 7 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 12 , & 1:7 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 13 , & 1:1:7 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 14 , & 2:7 , & [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 15 , & 7 , & [1/4,2/2] , & 2.0 , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	9 ,	3 ,	[1/4,2/1] ,	2.0 , ε)
(	10 ,	4:3 ,	[1/4,2/1] ,	2.0 , ε)
(	11 ,	7 ,	[1/4,2/1] ,	2.0 , ε)
(	12 ,	1:7 ,	[1/4,2/1] ,	2.0 , ε)
(	13 ,	1:1:7 ,	[1/4,2/1] ,	2.0 , ε)
(	14 ,	2:7 ,	[1/4,2/1] ,	2.0 , ε)
(	15 ,	7 ,	[1/4,2/2] ,	2.0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(10 ,	4:3 ,	[1/4,2/1] ,	2.0	, ε)
(11 ,	7 ,	[1/4,2/1] ,	2.0	, ε)
(12 ,	1:7 ,	[1/4,2/1] ,	2.0	, ε)
(13 ,	1:1:7 ,	[1/4,2/1] ,	2.0	, ε)
(14 ,	2:7 ,	[1/4,2/1] ,	2.0	, ε)
(15 ,	7 ,	[1/4,2/2] ,	2.0	, ε)
(4 ,	7 ,	[1/4,2/2] ,	2.0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(10 ,	4:3 ,	[1/4,2/1] ,	2.0	, ε)
(11 ,	7 ,	[1/4,2/1] ,	2.0	, ε)
(12 ,	1:7 ,	[1/4,2/1] ,	2.0	, ε)
(13 ,	1:1:7 ,	[1/4,2/1] ,	2.0	, ε)
(14 ,	2:7 ,	[1/4,2/1] ,	2.0	, ε)
(15 ,	7 ,	[1/4,2/2] ,	2.0	, ε)
(4 ,	7 ,	[1/4,2/2] ,	2.0	, ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 11 , & 7 , & [1/4,2/1] , 2.0 , \varepsilon) \\
(& 12 , & 1{:}7 , & [1/4,2/1] , 2.0 , \varepsilon) \\
(& 13 , & 1{:}1{:}7 , & [1/4,2/1] , 2.0 , \varepsilon) \\
(& 14 , & 2{:}7 , & [1/4,2/1] , 2.0 , \varepsilon) \\
(& 15 , & 7 , & [1/4,2/2] , 2.0 , \varepsilon) \\
(& 4 , & 7 , & [1/4,2/2] , 2.0 , \varepsilon) \\
(& 5 , & 7 , & [1/2,2/2] , 0 , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 11 , & 7 , [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 12 , & 1{:}7 , [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 13 , & 1{:}1{:}7 , [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 14 , & 2{:}7 , [1/4,2/1] , & 2.0 , \varepsilon) \\
(& 15 , & 7 , [1/4,2/2] , & 2.0 , \varepsilon) \\
(& 4 , & 7 , [1/4,2/2] , & 2.0 , \varepsilon) \\
(& 5 , & 7 , [1/2,2/2] , & 0 , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 12 ,   1:7 , [1/4,2/1] , 2.0    , ε )
( 13 , 1:1:7 , [1/4,2/1] , 2.0    , ε )
( 14 ,   2:7 , [1/4,2/1] , 2.0    , ε )
( 15 ,     7 , [1/4,2/2] , 2.0    , ε )
(  4 ,     7 , [1/4,2/2] , 2.0    , ε )
(  5 ,     7 , [1/2,2/2] , 0      , ε )
(  6 ,   2:7 , [1/2,2/2] , 0      , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 12 ,   1:7 , [1/4,2/1] , 2.0     , ε )
( 13 , 1:1:7 , [1/4,2/1] , 2.0     , ε )
( 14 ,   2:7 , [1/4,2/1] , 2.0     , ε )
( 15 ,     7 , [1/4,2/2] , 2.0     , ε )
(  4 ,     7 , [1/4,2/2] , 2.0     , ε )
(  5 ,     7 , [1/2,2/2] , 0       , ε )
(  6 ,   2:7 , [1/2,2/2] , 0       , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 13 , 1:1:7 , [1/4,2/1] , 2.0     , ε )
( 14 ,   2:7 , [1/4,2/1] , 2.0     , ε )
( 15 ,     7 , [1/4,2/2] , 2.0     , ε )
(  4 ,     7 , [1/4,2/2] , 2.0     , ε )
(  5 ,     7 , [1/2,2/2] , 0       , ε )
(  6 ,   2:7 , [1/2,2/2] , 0       , ε )
(  7 , 0:2:7 , [1/2,2/2] , 0       , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 13 , 1:1:7 , [1/4,2/1] , 2.0     , ε )
( 14 ,   2:7 , [1/4,2/1] , 2.0     , ε )
( 15 ,     7 , [1/4,2/2] , 2.0     , ε )
(  4 ,     7 , [1/4,2/2] , 2.0     , ε )
(  5 ,     7 , [1/2,2/2] , 0       , ε )
(  6 ,   2:7 , [1/2,2/2] , 0       , ε )
(  7 , 0:2:7 , [1/2,2/2] , 0       , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 14 ,   2:7 , [1/4,2/1] , 2.0     , ε )
( 15 ,     7 , [1/4,2/2] , 2.0     , ε )
(  4 ,     7 , [1/4,2/2] , 2.0     , ε )
(  5 ,     7 , [1/2,2/2] , 0       , ε )
(  6 ,   2:7 , [1/2,2/2] , 0       , ε )
(  7 , 0:2:7 , [1/2,2/2] , 0       , ε )
(  8 ,   1:7 , [1/2,2/2] , 0       , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(14 ,	2:7 ,	[1/4,2/1] ,	2.0	, ε)
(15 ,	7 ,	[1/4,2/2] ,	2.0	, ε)
(4 ,	7 ,	[1/4,2/2] ,	2.0	, ε)
(5 ,	7 ,	[1/2,2/2] ,	0	, ε)
(6 ,	2:7 ,	[1/2,2/2] ,	0	, ε)
(7 ,	0:2:7 ,	[1/2,2/2] ,	0	, ε)
(8 ,	1:7 ,	[1/2,2/2] ,	0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(15 ,	7 ,	[1/4,2/2] ,	2.0	, ε)
(4 ,	7 ,	[1/4,2/2] ,	2.0	, ε)
(5 ,	7 ,	[1/2,2/2] ,	0	, ε)
(6 ,	2:7 ,	[1/2,2/2] ,	0	, ε)
(7 ,	0:2:7 ,	[1/2,2/2] ,	0	, ε)
(8 ,	1:7 ,	[1/2,2/2] ,	0	, ε)
(9 ,	7 ,	[1/2,2/2] ,	0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(15 ,	7 ,	[1/4,2/2] ,	2.0	, ε)
(4 ,	7 ,	[1/4,2/2] ,	2.0	, ε)
(5 ,	7 ,	[1/2,2/2] ,	0	, ε)
(6 ,	2:7 ,	[1/2,2/2] ,	0	, ε)
(7 ,	0:2:7 ,	[1/2,2/2] ,	0	, ε)
(8 ,	1:7 ,	[1/2,2/2] ,	0	, ε)
(9 ,	7 ,	[1/2,2/2] ,	0	, ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	4 ,	7 ,	[1/4,2/2] , 2.0	, ε)
(	5 ,	7 ,	[1/2,2/2] , 0	, ε)
(	6 ,	2:7 ,	[1/2,2/2] , 0	, ε)
(	7 ,	0:2:7 ,	[1/2,2/2] , 0	, ε)
(	8 ,	1:7 ,	[1/2,2/2] , 0	, ε)
(	9 ,	7 ,	[1/2,2/2] , 0	, ε)
(	10 ,	2:7 ,	[1/2,2/2] , 0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	4 ,	7 ,	[1/4,2/2] , 2.0	, ε)
(	5 ,	7 ,	[1/2,2/2] , 0	, ε)
(	6 ,	2:7 ,	[1/2,2/2] , 0	, ε)
(	7 ,	0:2:7 ,	[1/2,2/2] , 0	, ε)
(	8 ,	1:7 ,	[1/2,2/2] , 0	, ε)
(	9 ,	7 ,	[1/2,2/2] , 0	, ε)
(	10 ,	2:7 ,	[1/2,2/2] , 0	, ε)

Erweiterung

Aufgabe:

- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(5 ,	7 ,	[1/2,2/2] ,	0	, ε)
(6 ,	2:7 ,	[1/2,2/2] ,	0	, ε)
(7 ,	0:2:7 ,	[1/2,2/2] ,	0	, ε)
(8 ,	1:7 ,	[1/2,2/2] ,	0	, ε)
(9 ,	7 ,	[1/2,2/2] ,	0	, ε)
(10 ,	2:7 ,	[1/2,2/2] ,	0	, ε)
(11 ,	9 ,	[1/2,2/2] ,	0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(	5 ,	7 ,	[1/2,2/2] , 0	, ε)
(	6 ,	2:7 ,	[1/2,2/2] , 0	, ε)
(	7 ,	0:2:7 ,	[1/2,2/2] , 0	, ε)
(	8 ,	1:7 ,	[1/2,2/2] , 0	, ε)
(	9 ,	7 ,	[1/2,2/2] , 0	, ε)
(	10 ,	2:7 ,	[1/2,2/2] , 0	, ε)
(	11 ,	9 ,	[1/2,2/2] , 0	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 6 , & 2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 7 , & 0{:}2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 8 , & 1{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 12 , & 2{:}9 , [1/2{,}2/2] , 0 & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 6 , & 2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 7 , & 0{:}2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 8 , & 1{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 12 , & 2{:}9 , [1/2{,}2/2] , 0 & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 7 , & 0{:}2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 8 , & 1{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 12 , & 2{:}9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 13 , & 1{:}2{:}9 , [1/2{,}2/2] , 0 & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
(  7 , 0:2:7 , [1/2,2/2] , 0     , ε )
(  8 ,   1:7 , [1/2,2/2] , 0     , ε )
(  9 ,     7 , [1/2,2/2] , 0     , ε )
( 10 ,   2:7 , [1/2,2/2] , 0     , ε )
( 11 ,     9 , [1/2,2/2] , 0     , ε )
( 12 ,   2:9 , [1/2,2/2] , 0     , ε )
( 13 , 1:2:9 , [1/2,2/2] , 0     , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(8 , 1:7 , [1/2,2/2] , 0 , ε)
(9 , 7 , [1/2,2/2] , 0 , ε)
(10 , 2:7 , [1/2,2/2] , 0 , ε)
(11 , 9 , [1/2,2/2] , 0 , ε)
(12 , 2:9 , [1/2,2/2] , 0 , ε)
(13 , 1:2:9 , [1/2,2/2] , 0 , ε)
(14 , 3:9 , [1/2,2/2] , 0 , ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 8 , & 1{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 9 , & 7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 12 , & 2{:}9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 13 , & 1{:}2{:}9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 14 , & 3{:}9 , [1/2{,}2/2] , 0 & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 9 , & 7 , [1/2,2/2] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , [1/2,2/2] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2,2/2] , 0 & , \varepsilon) \\
(& 12 , & 2{:}9 , [1/2,2/2] , 0 & , \varepsilon) \\
(& 13 , & 1{:}2{:}9 , [1/2,2/2] , 0 & , \varepsilon) \\
(& 14 , & 3{:}9 , [1/2,2/2] , 0 & , \varepsilon) \\
(& 15 , & 9 , [1/2,2/3] , 0 & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{lrlll}
(& 9 , & 7 , & [1/2,2/2] , 0 & , \varepsilon) \\
(& 10 , & 2{:}7 , & [1/2,2/2] , 0 & , \varepsilon) \\
(& 11 , & 9 , & [1/2,2/2] , 0 & , \varepsilon) \\
(& 12 , & 2{:}9 , & [1/2,2/2] , 0 & , \varepsilon) \\
(& 13 , & 1{:}2{:}9 , & [1/2,2/2] , 0 & , \varepsilon) \\
(& 14 , & 3{:}9 , & [1/2,2/2] , 0 & , \varepsilon) \\
(& 15 , & 9 , & [1/2,2/3] , 0 & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 10 ,   2:7 , [1/2,2/2] , 0      , ε )
( 11 ,     9 , [1/2,2/2] , 0      , ε )
( 12 ,   2:9 , [1/2,2/2] , 0      , ε )
( 13 , 1:2:9 , [1/2,2/2] , 0      , ε )
( 14 ,   3:9 , [1/2,2/2] , 0      , ε )
( 15 ,     9 , [1/2,2/3] , 0      , ε )
(  4 ,     9 , [1/2,2/3] , 0      , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 10 , & 2{:}7 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 11 , & 9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 12 , & 2{:}9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 13 , & 1{:}2{:}9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 14 , & 3{:}9 , [1/2{,}2/2] , 0 & , \varepsilon) \\
(& 15 , & 9 , [1/2{,}2/3] , 0 & , \varepsilon) \\
(& 4 , & 9 , [1/2{,}2/3] , 0 & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(11 ,	9	, [1/2,2/2] ,	0	, ε)
(12 ,	2:9	, [1/2,2/2] ,	0	, ε)
(13 ,	1:2:9	, [1/2,2/2] ,	0	, ε)
(14 ,	3:9	, [1/2,2/2] ,	0	, ε)
(15 ,	9	, [1/2,2/3] ,	0	, ε)
(4 ,	9	, [1/2,2/3] ,	0	, ε)
(5 ,	9	, [1/0,2/3] ,	ε	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(11 ,	9	, [1/2,2/2] , 0		, ε)
(12 ,	2:9	, [1/2,2/2] , 0		, ε)
(13 ,	1:2:9	, [1/2,2/2] , 0		, ε)
(14 ,	3:9	, [1/2,2/2] , 0		, ε)
(15 ,	9	, [1/2,2/3] , 0		, ε)
(4 ,	9	, [1/2,2/3] , 0		, ε)
(5 ,	9	, [1/0,2/3] , ε		, ε)

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 12 ,   2:9 , [1/2,2/2] , 0       , ε )
( 13 , 1:2:9 , [1/2,2/2] , 0       , ε )
( 14 ,   3:9 , [1/2,2/2] , 0       , ε )
( 15 ,     9 , [1/2,2/3] , 0       , ε )
(  4 ,     9 , [1/2,2/3] , 0       , ε )
(  5 ,     9 , [1/0,2/3] , ε       , ε )
(  6 ,   0:9 , [1/0,2/3] , ε       , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 12 ,   2:9 , [1/2,2/2] , 0       , ε )
( 13 , 1:2:9 , [1/2,2/2] , 0       , ε )
( 14 ,   3:9 , [1/2,2/2] , 0       , ε )
( 15 ,     9 , [1/2,2/3] , 0       , ε )
(  4 ,     9 , [1/2,2/3] , 0       , ε )
(  5 ,     9 , [1/0,2/3] , ε       , ε )
(  6 ,   0:9 , [1/0,2/3] , ε       , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 13 , 1:2:9 , [1/2,2/2] , 0       , ε )
( 14 ,   3:9 , [1/2,2/2] , 0       , ε )
( 15 ,     9 , [1/2,2/3] , 0       , ε )
(  4 ,     9 , [1/2,2/3] , 0       , ε )
(  5 ,     9 , [1/0,2/3] , ε       , ε )
(  6 ,   0:9 , [1/0,2/3] , ε       , ε )
(  7 , 0:0:9 , [1/0,2/3] , ε       , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

```
( 13 , 1:2:9 , [1/2,2/2] , 0       , ε )
( 14 ,   3:9 , [1/2,2/2] , 0       , ε )
( 15 ,     9 , [1/2,2/3] , 0       , ε )
(  4 ,     9 , [1/2,2/3] , 0       , ε )
(  5 ,     9 , [1/0,2/3] , ε       , ε )
(  6 ,   0:9 , [1/0,2/3] , ε       , ε )
(  7 , 0:0:9 , [1/0,2/3] , ε       , ε )
```

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(14 ,	3:9 ,	[1/2,2/2] ,	0	, ε)
(15 ,	9 ,	[1/2,2/3] ,	0	, ε)
(4 ,	9 ,	[1/2,2/3] ,	0	, ε)
(5 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(6 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(7 ,	0:0:9 ,	[1/0,2/3] ,	ε	, ε)
(8 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(14 ,	3:9 ,	[1/2,2/2] ,	0	, ε)
(15 ,	9 ,	[1/2,2/3] ,	0	, ε)
(4 ,	9 ,	[1/2,2/3] ,	0	, ε)
(5 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(6 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(7 ,	0:0:9 ,	[1/0,2/3] ,	ε	, ε)
(8 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(15 ,	9	, [1/2,2/3] ,	0	, ε)
(4 ,	9	, [1/2,2/3] ,	0	, ε)
(5 ,	9	, [1/0,2/3] ,	ε	, ε)
(6 ,	0:9	, [1/0,2/3] ,	ε	, ε)
(7 ,	0:0:9	, [1/0,2/3] ,	ε	, ε)
(8 ,	0:9	, [1/0,2/3] ,	ε	, ε)
(16 ,	9	, [1/0,2/3] ,	ε	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(15 ,	9	, [1/2,2/3] ,	0	, ε)
(4 ,	9	, [1/2,2/3] ,	0	, ε)
(5 ,	9	, [1/0,2/3] ,	ε	, ε)
(6 ,	0:9	, [1/0,2/3] ,	ε	, ε)
(7 ,	0:0:9	, [1/0,2/3] ,	ε	, ε)
(8 ,	0:9	, [1/0,2/3] ,	ε	, ε)
(16 ,	9	, [1/0,2/3] ,	ε	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(4 ,	9 ,	[1/2,2/3] ,	0	, ε)
(5 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(6 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(7 ,	0:0:9 ,	[1/0,2/3] ,	ε	, ε)
(8 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(16 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(17 ,	3:9 ,	[1/0,2/3] ,	ε	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(4 ,	9 ,	[1/2,2/3] ,	0	, ε)
(5 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(6 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(7 ,	0:0:9 ,	[1/0,2/3] ,	ε	, ε)
(8 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(16 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(17 ,	3:9 ,	[1/0,2/3] ,	ε	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(5 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(6 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(7 ,	0:0:9 ,	[1/0,2/3] ,	ε	, ε)
(8 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(16 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(17 ,	3:9 ,	[1/0,2/3] ,	ε	, ε)
(18 ,	3 ,	[1/0,2/3] ,	ε	, ε)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{lrlll}
(& 5 , & 9 , & [1/0,2/3] , & \varepsilon , \varepsilon) \\
(& 6 , & 0{:}9 , & [1/0,2/3] , & \varepsilon , \varepsilon) \\
(& 7 , & 0{:}0{:}9 , & [1/0,2/3] , & \varepsilon , \varepsilon) \\
(& 8 , & 0{:}9 , & [1/0,2/3] , & \varepsilon , \varepsilon) \\
(& 16 , & 9 , & [1/0,2/3] , & \varepsilon , \varepsilon) \\
(& 17 , & 3{:}9 , & [1/0,2/3] , & \varepsilon , \varepsilon) \\
(& 18 , & 3 , & [1/0,2/3] , & \varepsilon , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe: ▶ statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llll}
(& 6 , & 0{:}9 , [1/0{,}2/3] , \varepsilon & , \varepsilon) \\
(& 7 , & 0{:}0{:}9 , [1/0{,}2/3] , \varepsilon & , \varepsilon) \\
(& 8 , & 0{:}9 , [1/0{,}2/3] , \varepsilon & , \varepsilon) \\
(& 16 , & 9 , [1/0{,}2/3] , \varepsilon & , \varepsilon) \\
(& 17 , & 3{:}9 , [1/0{,}2/3] , \varepsilon & , \varepsilon) \\
(& 18 , & 3 , [1/0{,}2/3] , \varepsilon & , \varepsilon) \\
(& 19 , & \varepsilon , [1/3{,}2/3] , \varepsilon & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

$$
\begin{array}{llllll}
(& 6 , & 0{:}9 , & [1/0{,}2/3] , & \varepsilon & , \varepsilon) \\
(& 7 , & 0{:}0{:}9 , & [1/0{,}2/3] , & \varepsilon & , \varepsilon) \\
(& 8 , & 0{:}9 , & [1/0{,}2/3] , & \varepsilon & , \varepsilon) \\
(& 16 , & 9 , & [1/0{,}2/3] , & \varepsilon & , \varepsilon) \\
(& 17 , & 3{:}9 , & [1/0{,}2/3] , & \varepsilon & , \varepsilon) \\
(& 18 , & 3 , & [1/0{,}2/3] , & \varepsilon & , \varepsilon) \\
(& 19 , & \varepsilon , & [1/3{,}2/3] , & \varepsilon & , \varepsilon)
\end{array}
$$

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(7 ,	0:0:9 ,	[1/0,2/3] ,	ε	, ε)
(8 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(16 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(17 ,	3:9 ,	[1/0,2/3] ,	ε	, ε)
(18 ,	3 ,	[1/0,2/3] ,	ε	, ε)
(19 ,	ε ,	[1/3,2/3] ,	ε	, ε)
(20 ,	ε ,	[1/3,2/3] ,	ε	, 3)

Erweiterung

Aufgabe:
- statt Summe, nun Durchschnitt zu berechnen

```
 1: LIT 0;
 2: STORE 2;
 3: LIT 0;
 4: READ 1;
 5: LOAD 1;
 6: LIT 0;
 7: NE;
 8: JMC 16;
 9: LOAD 1;
10: ADD;
11: LOAD 2;
12: LIT 1;
13: ADD;
14: STORE 2;
15: JMP 4;
16: LOAD 2;
17: DIV;
18: STORE 1;
19: WRITE 1;
```

(7 ,	0:0:9 ,	[1/0,2/3] ,	ε	, ε)
(8 ,	0:9 ,	[1/0,2/3] ,	ε	, ε)
(16 ,	9 ,	[1/0,2/3] ,	ε	, ε)
(17 ,	3:9 ,	[1/0,2/3] ,	ε	, ε)
(18 ,	3 ,	[1/0,2/3] ,	ε	, ε)
(19 ,	ε ,	[1/3,2/3] ,	ε	, ε)
(20 ,	ε ,	[1/3,2/3] ,	ε	, 3)