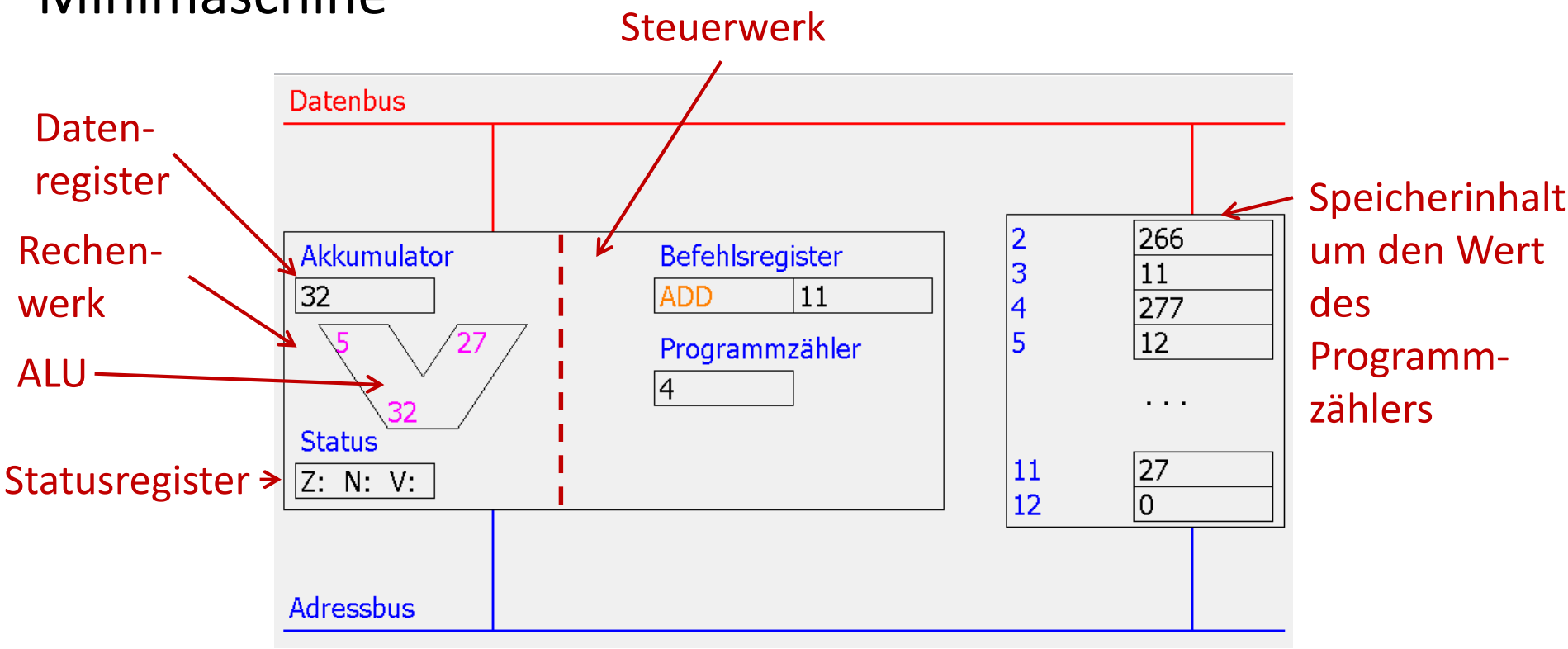
3 Die Registermaschine

Minimaschine



Speicher

	0	1	2	3	4	5	6	7	8	9
0	532	5	266	11	277	12	99	0	0	0
10	0	27	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0

Öffne die Minimaschine. Mit Ablage – Neu öffnet sich ein Editorfenster.
Gib dort das folgende Programm ein:

```
LOADI 5
ADD 11
STORE 12
HOLD
```

Trage im Speicher in Zelle 11 eine Zahl ein, z.B. 23.
(Dazu muss in Werkzeuge "Speicher editieren" ausgewählt sein.)
Wähle im Editor Werkzeuge – Assemblieren und beobachte den Speicherinhalt.
Führe das Programm im CPU-Kontrollfenster in Mikroschritten durch und beobachte sowohl CPU als auch Speicher.

Mit Werkzeuge-CPU zurücksetzen wird alles wieder in den Ausgangszustand versetzt, so dass das Programm nochmals durchlaufen werden kann.

- Verändere den Inhalt von Speicherzelle 11 in -18 und -5 und beobachte das Statusregister.
- Verändere das Programm wie folgt:

  LOAD 5 (statt LOAD**I**)
  ADD**I** 11 (statt ADD)
  STORE 12
  HOLD

  Versuche, aufgrund des Speichers vorherzusagen, welches Ergebnis am Ende im Akkumulator steht.

Die Speicherzellen haben eine Länge von 16 Bit und können ganze Zahlen darstellen. Berechne, welcher Zahlbereich dadurch abgedeckt wird und teste deine Vermutung.

-32768 bis 32767

Teste, wie sich der Speicher verhält, wenn der Zahlbereich überschritten wird und beobachte das Statusregister.

```
LOADI 5
ADD 11
STORE 12
HOLD
```

Addiere zur Zahl 5 die Zahl in Speicherzelle 11 und speichere das Ergebnis in Speicherzelle 12.
kurz: [12] = 5 + [11]

- Das Befehlsregister enthält den momentan auszuführenden Befehl.
- Der Befehlszähler (Program Counter = PC) enthält die Speicheradresse des nächsten Maschinenbefehls.
- Datenregister sind spezielle Speichereinheiten im Prozessor, auf die schneller als auf den Arbeitsspeicher zugegriffen werden kann.
- Die Minimaschine ist eine 1-Registermaschine. Ein Operand kommt aus dem Datenregister, der andere aus dem Arbeitsspeicher. Das Ergebnis wird im Datenregister (Akkumulator) abgelegt.
- Das Statusregister besteht aus mehreren einzelnen Bits (Flags), z.B. Z = zero, N = negativ, V = overflow
- Maschinenbefehl: z.B. LOAD 5

Operationskennung ↗ Opcode

↖ Operandenteil Adressteil

Setze folgende Terme mit der Minimaschine um

- Beispiel a: [103] = [100] + [101] · [102]
- Beispiel b: [104] = [100] + [101] + [102] + [103]
- Beispiel c: [104] = [100] + 2 · [101] + 3 · [102] + 4 ·[103]
- Umgekehrt: Stelle den passenden Term auf!

```
LOAD 100
DIV 101
MUL 101
STORE 103
LOAD 100
SUB 103
STORE 103
HOLD
```

- Beispiel d:
  Erstelle ein Assemblerprogramm, das den Inhalt zweier Speicherplätze vertauscht.

Setze folgende Terme mit der Minimaschine um (Beispiel a):

- [103] = [100] + [101] · [102]

```
#Speicherbelegung:
LOADI 12
STORE 100
LOADI 10
STORE 101
LOADI 5
STORE 102
#Termberechnung
LOAD 101
MUL 102
ADD 100
STORE 103
HOLD
```

- Beispiel b: [104] = [100] + [101] + [102] + [103]

```
#Speicherbelegung:
LOADI 12
STORE 100
LOADI 10
STORE 101
LOADI 5
STORE 102
LOADI 23
STORE 103
#Termberechnung
LOAD 100
ADD 101
ADD 102
ADD 103
STORE 104
HOLD
```

- Beispiel c: [104] = [100] + 2 · [101] + 3 · [102] + 4 ·[103]

```
#Speicherbelegung: …
#Termberechnung
LOAD 100
STORE 104
LOAD 101
MULI 2
ADD 104
STORE 104
LOAD 102
MULI 3
ADD 104
STORE 104
LOAD 103
MULI 4
ADD 104
STORE 104
HOLD
```

- Umgekehrt: Stelle den passenden Term auf!

```
LOAD 100
DIV 101
MUL 101
STORE 103
LOAD 100
SUB 103
STORE 103
HOLD
```

$[103] = [100] – [100] / [101] \cdot [101]$

/ Ganzzahldivision

In [103]: Rest bei Division

- Beispiel d:
  Erstelle ein Assemblerprogramm, das den Inhalt zweier Speicherplätze vertauscht.

```
LOAD 100
STORE 102
LOAD 101
STORE 100
LOAD 102
STORE 101
HOLD
```