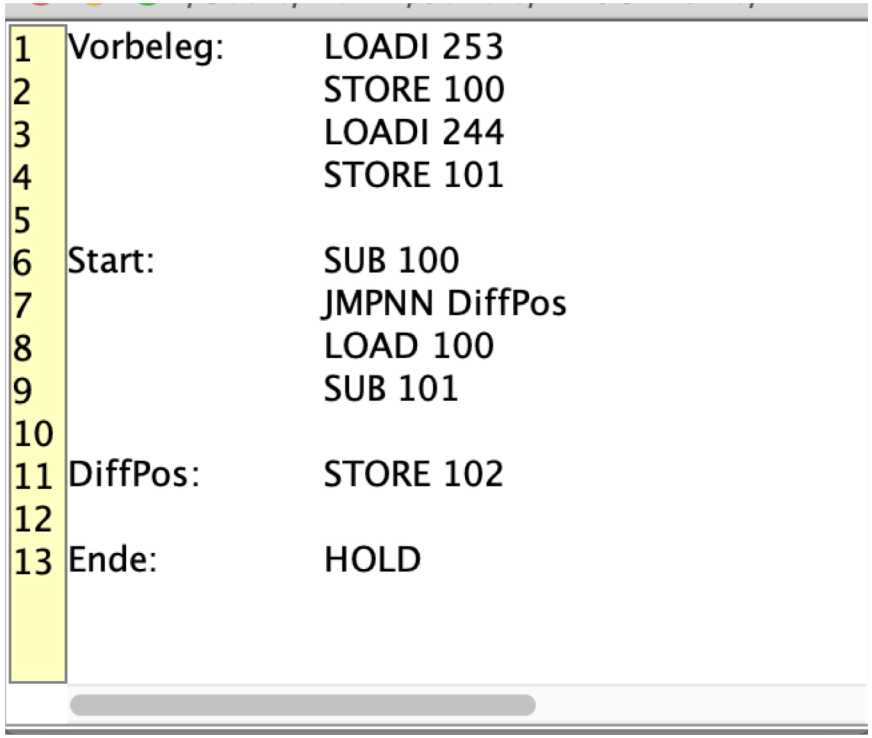
3. Funktionsweise eines Rechners

3.3 Systemnahe Programmierung

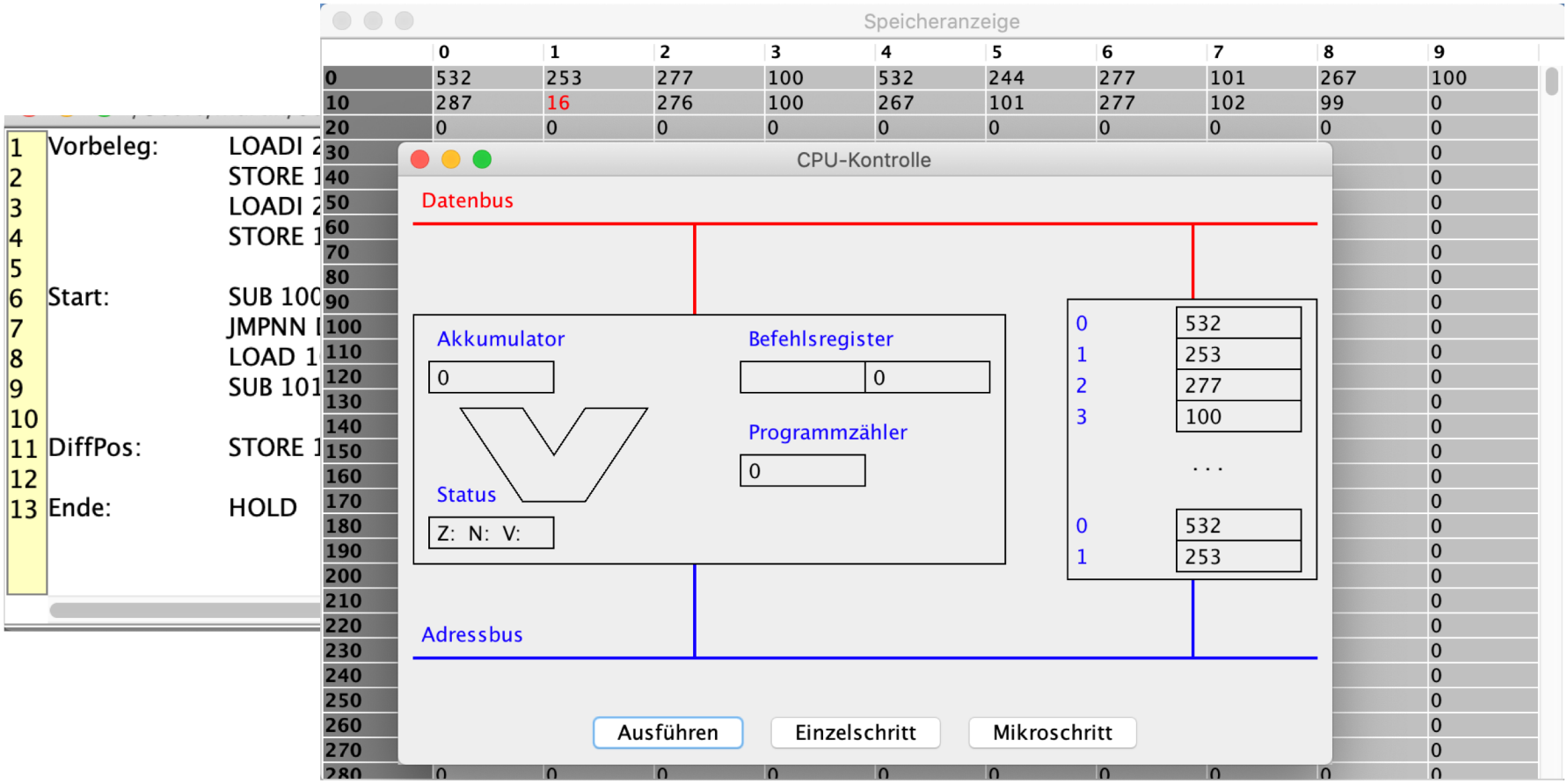
Als Simulationsprogramm verwenden wir die Minimaschine
(https://schule.awiedemann.de/minimaschine.html)



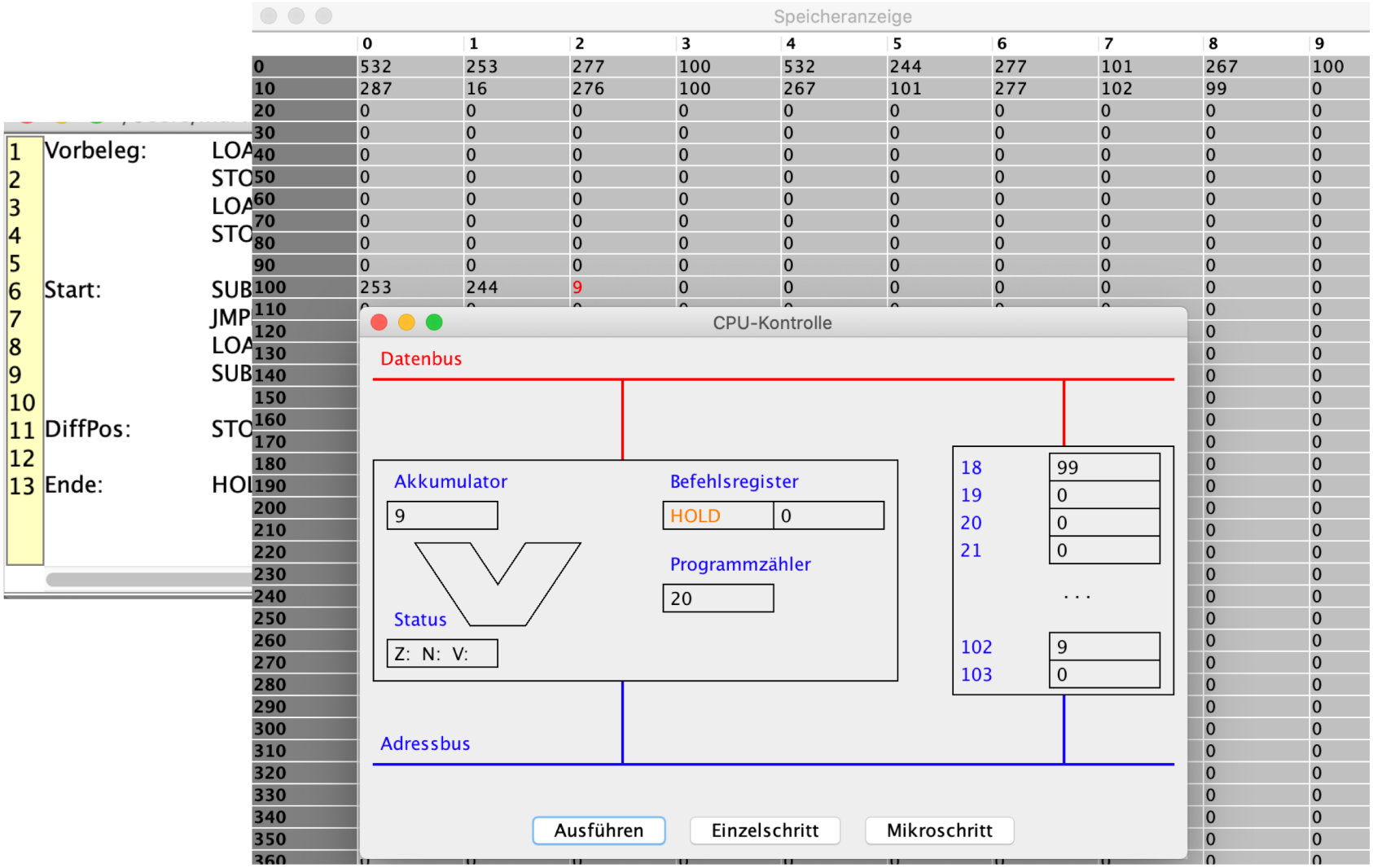
```
1   Vorbeleg:     LOADI 253
2                 STORE 100
3                 LOADI 244
4                 STORE 101
5
6   Start:        SUB 100
7                 JMPNN DiffPos
8                 LOAD 100
9                 SUB 101
10
11  DiffPos:      STORE 102
12
13  Ende:         HOLD
```

- Programm öffnen oder schreiben
- Werkzeuge - Assemblieren

Das Programm und die Startwerte werden durch das Assemblieren in den Speicher geladen:



Ausführen des Programms:



Befehlssatz:

Speicherbefehle	
LOAD adresse	Lädt den Wert von der angegebenen Adresse in den Akkumulator.
LOADI zahl	Lädt die angegebenen Zahl in den Akkumulator.
STORE adresse	Speichert den Wert im Akkumulator an der angegebenen Adresse.

Artihmetikbefehle	
ADD adresse	Addiert den Wert von der angegebenen Adresse zum Akkumulator.
SUB adresse	Subtrahiert den Wert der angegebenen Adresse vom Akkumulator.
MUL adresse	Multipliziert den Wert von der angegebenen Adresse zum Akkumulator.
DIV adresse	Dividiert den Wert im Akkumulator durch den Wert der angegebenen Adresse.
MOD adresse	Dividiert den Wert im Akkumulator durch den Wert der angegebenen Adresse und speichert den Rest im Akkumulator.
CMP adresse	Vergleicht den Wert der angegebenen Adresse mit dem Akkumulator und setzt Null- und Negativflag entsprechend. Dazu wird berechnet Akku - Wert von adresse
ADDI zahl	Addiert den angegebenen Wert zum Akkumulator.
SUBI zahl	Subtrahiert den angegebenen Wert vom Akkumulator.
MULI zahl	Multipliziert den angegebenen Wert zum Akkumulator.
DIVI zahl	Dividiert den Wert im Akkumulator durch den angegebenen Wert.
MODI zahl	Dividiert den Wert im Akkumulator durch den angegebenen Wert und speichert den Rest im Akkumulator.
CMPI zahl	Vergleicht den angegebenen Wert mit dem Akkumulator und setzt Null- und Negativflag entsprechend. Dazu wird berechnet Akku - zahl

Sprungbefehle	
JMPP adresse	Springt zur angegebenen Adresse, wenn das Ergebnis der letzen Operation positiv (> 0) war, d. h. weder N noch Z-Flag sind gesetzt.
JMPNN adresse	Springt zur angegebenen Adresse, wenn das Ergebnis der letzen Operation nicht negativ (≥ 0) war, d. h. das N-Flag ist nicht gesetzt.
JMPN adresse	Springt zur angegebenen Adresse, wenn das Ergebnis der letzen Operation negativ (< 0) war, d. h. das N-Flag ist gesetzt.
JMPNP adresse	Springt zur angegebenen Adresse, wenn das Ergebnis der letzen Operation nicht positiv (≤ 0) war, d. h. das N-Flag oder as Z-Flag ist gesetzt.
JMPZ adresse	Springt zur angegebenen Adresse, wenn das Ergebnis der letzen Operation zero (= 0) war, d. h. das Z-Flag ist gesetzt.
JMPNZ adresse	Springt zur angegebenen Adresse, wenn das Ergebnis der letzen Operation nicht zerol (≠ 0) war, d. h. das Z-Flag ist nicht gesetzt.
JMPV adresse	Springt zur angegebenen Adresse, wenn die letzte Operation einen Überlauf verursacht hat, d. h. das V-Flag ist gesetzt.
JMP adresse	Springt zur angegebenen Adresse.

Sonstige Befehle	
HOLD	Hält den Prozessor an.
RESET	Setzt den Prozessor auf den Startzustand zurück.
NOOP	Tut einfach nichts (NO OPeration).

Speicherorganisation	
WORD zahl	Besetzt eine Speicherzelle mit der angegebenen Zahl.

Umsetzung von einfachen Algorithmen:

Sequenz:
Lade drei Werte in die Zellen [100], [101] und [102], berechne den Termwert ([100]+[101])*[102] und speichere das Ergebnis in [103].

Vorbelegung:	**LOADI 502**
	STORE 100
	LOADI 18
	STORE 101
	LOADI 25
	STORE 102
Start:	**LOAD 100**
	ADD 101
	MUL 102
	STORE 103
Ende:	**HOLD**

Einseitige bedingte Anweisung:
Berechne von zwei Zahlen den Betrag der Differenz.

Algorithmus:

erg =[101]-[100]

wenn erg <0
erg=[100]-[101]
ende wenn

[102]=erg

Vorbelegung:	**LOADI 253**	
	STORE 100	
	LOADI 244	
	STORE 101	
Start:	**SUB 100**	berechnet [101]-[100]
	JMPNN DiffPos	ist das Ergebnis nicht negativ, kann gleich zu DiffPos gesprungen werden
	LOAD 100	
	SUB 101	berechnet [100]-[101]
DiffPos:	**STORE 102**	
Ende:	**HOLD**	

Zweiseitig bedingte Anweisung:

Ist eine Zahl gerade, so halbiere sie. Ist sie ungerade verdopple sie.

Algorithmus:

wenn [100] mod 2 = 0
[101] = [100]/2
ende wenn

sonst
[101] = [100]*2
ende sonst

Vorbelegung:	**LOADI 24**	
	STORE 100	
Start:	**MODI 2**	berechnet denRest bei der Division durch 2
	JMPNZ Ungerade	ist das Ergebnis ungleich 0, ist die Zahl ungerade. Springe zum sonst-Teil (Ungerade)
	LOAD 100	Wenn-Teil (halbiere die Zahl)
	DIVI 2	
	JMP Ergebnis	Überspringe den sonst-Teil!
Ungerade:	**LOAD 100**	Sonst-Teil (verdopple die Zahl)
	MULI 2	
Ergebnis:	**STORE 101**	
Ende:	**HOLD**	

Wiederholung mit Anfangsbedingung

Berechne die Summe aller ungeraden Zahlen 15 + 13 + 11 + ... + 3 + 1 .

Algorithmus:

ergebnis = 15
summand = ergebnis - 2
wiederhole solange summand >0
ergebnis = ergebnis + summand
summand = summand - 2
ende wiederhole

Vorbelegung:	**LOADI 15**	
Ergebnis:	**STORE 102**	erster Summand und Ergebnis
	SUBI 2	
Summand:	**STORE 101**	nächster Summand
Wiederhole:	**LOAD 101**	lädt den aktualisierten nächsten Summanden
	JMPN Ende	ist der Summand negativ, wird die Wiederholung übersprungen
	LOAD 102	Ergebnis laden
	ADD 101	nächsten Summanden addieren
	STORE 102	Ergebnis speichern
	LOAD 101	wieder den Summanden laden
	SUBI 2	diesen um zwei verkleinern
	STORE 101	speichern
	JMP Wiederhole	zum Anfang der Schleife springen
Ende:	**LOAD 102**	
	HOLD	

Wiederholung mit fester Anzahl :

Verdopple n (z.B. 3) mal eine gegebene Zahl (z.B. 5) ; (5*2*2*2)

Algorithmus:

anzahl = n
zahl = 5
ergebnis = zahl
wiederhole anzahl mal
ergebnis = ergebnis * 2
ende wiederhole

Vorbelegung:	**LOADI 5**	
	STORE 101	Zahl und Ergebnis speichern
	LOADI 3	
	STORE 102	Anzahl der Durchläufe
	LOADI 1	
	STORE 103	Zählvariable
Wiederhole:	**CMP 102**	vergleiche die Zählvariable mit der Anzahl der Durchläufe; berechnet [103] - [102]
	JMPP Ende	erreicht die Zählvariable den Wert 3, ergibt der Vergleich den Wert 0; die Schleife wird das letzte Mal durchlaufen.
	LOAD 101	Ergebnis laden
	MULI 2	verdoppeln
	STORE 101	Ergebnis speichern
	LOAD 103	Zählvariable laden
	ADDI 1	diese um 1 erhöhen
	STORE 103	speichern
	JMP Wiederhole	zum Anfang der Schleife springen
Ende:	**LOAD 101**	
	HOLD	