Aufbau

Die Bauteile

RAM	Zellen 0 bis 999: Programm **und** Daten
Akkumulator (ACC)	Rechenwerk: das aktuelle Ergebnis
Befehlsregister (INS)	der Befehl, der gerade läuft
Befehlszähler (PC)	Adresse des nächsten Befehls
Control Unit	Steuerwerk: enthält den Mikrocode
Adressbus (AB)	welche Zelle?
Datenbus (DB)	welcher Inhalt?

Gerechnet wird nur im Akkumulator. Johnny hat kein Ein- und Ausgabewerk: Zahlen werden direkt in den RAM geschrieben.

Zahlen im RAM

Jede Zelle enthält eine Zahl von 0 bis 19999. Johnny zeigt sie als `XX.YYY` an.

`XX`	Befehlscode (00 = Zahl)
`YYY`	Adresse
`00.128`	die Zahl 128
`01.014`	TAKE 14
`10.000`	HLT 0

Ob eine Zelle Zahl oder Befehl ist, entscheidet nur, wie sie benutzt wird. `[x]` bedeutet: die Zahl in Zelle `x`.
Johnny kennt **keine negativen Zahlen**: Bei `minus`, `SUB` und `DEC` bleibt es bei 0 (z. B. $23 - 42 = 0$). Nach oben ist bei 19999 Schluss.

Bedienung

linker Pfeil Zahl in eine Zelle schreiben

New RAM RAM leeren (oben links)

Reset PC und ACC auf 0, RAM bleibt

Control Unit abdecken Mikrobefehle an RAM, Akku und Bussen von Hand klicken

Control Unit aufdecken Mikrocode ansehen; Johnny läuft selbst

ManuellAb Zahl auf den Adressbus

ManuellDb Zahl auf den Datenbus

ManuellAb und ManuellDb sind Hilfsknöpfe und nicht im Mikrocode verwendbar. Die Busse behalten ihre Zahl, bis eine andere daraufgelegt wird.

Programmieren

Assemblerbefehle

`01 TAKE x`	ACC = [x]
`02 ADD x`	ACC = ACC + [x]
`03 SUB x`	ACC = ACC - [x]
`04 SAVE x`	[x] = ACC
`05 JMP x`	springe zu Zelle x
`06 TST x`	falls [x] = 0: nächsten Befehl überspringen
`07 INC x`	[x] = [x] + 1
`08 DEC x`	[x] = [x] - 1
`09 NULL x`	[x] = 0
`10 HLT x`	anhalten, x ist egal

Befehl in Zahl umrechnen

Befehlscode vorne, Adresse immer dreistellig hinten.

```
ADD 20  ->  02.020  ->  2020
SAVE 7  ->  04.007  ->  4007
HLT 0   ->  10.000  ->  10000
```

Programm eintragen

1. New RAM: alten Rest entfernen
2. Befehle ab Zelle 0 als Zahlen eintragen
3. Daten in die Zellen schreiben
4. Reset: PC und ACC auf 0
5. Programm im Einzelschritt oder ganz laufen lassen

Das Programm braucht am Ende einen `HLT`-Befehl.

Fehlersuche

Falsches Ergebnis: im Einzelschritt ablaufen lassen und dabei ACC und PC beobachten. Steht im Befehl die richtige Adresse?
Seltsame Anzeige: Zahl und Befehl verwechselt. Befehlscode und dreistellige Adresse prüfen.
Programm startet falsch: Reset vergessen, PC steht nicht auf 0.
Reste vom alten Programm: vorher New RAM klicken.

Mikrobefehle

RAM, Akkumulator, Busse

`ram->db`	[AB] auf den Datenbus
`db->ram`	Datenbus in Zelle [AB]
`db->acc`	Datenbus in den Akku
`acc->db`	Akku auf den Datenbus
`plus`	ACC = ACC + DB
`minus`	ACC = ACC - DB
`acc++`	ACC um 1 erhöhen
`acc--`	ACC um 1 erniedrigen
`acc=0`	ACC auf 0

`plus` und `minus` rechnen mit dem, was gerade auf dem Datenbus liegt.

Steuerwerk

`pc->ab`	PC auf den Adressbus
`pc++`	PC um 1 erhöhen
`=0:pc++`	falls ACC = 0: PC um 1 erhöhen
`db->ins`	Datenbus ins Befehlsregister
`ins->ab`	Adressteil YYY auf den Adressbus
`ins->pc`	Adressteil YYY in den PC
`ins->mc`	zum Mikroprogramm des Befehlscodes XX
`mc=0`	zurück zum Anfang (FETCH)
`stop`	Programm beenden

Der Befehlszyklus

Vor jedem Befehl steht der Teil FETCH (holen und entschlüsseln):

```
pc->ab
ram->db
db->ins
ins->mc
```

Danach folgt das Mikroprogramm des Befehls. Es endet fast immer mit `pc++` und `mc=0`.

Beispiele für Mikroprogramme

TAKE x: `acc=0, ins->ab, ram->db, plus, pc++, mc=0`
SAVE x: `ins->ab, acc->db, db->ram, pc++, mc=0`

Wert von Hand berechnen

Beispiel `[8] = [5] + [6]` bei abgedeckter Control Unit:

```
ManuellAb 5; ram->db
db->acc
ManuellAb 6; ram->db
plus
acc->db
ManuellAb 8; db->ram
```