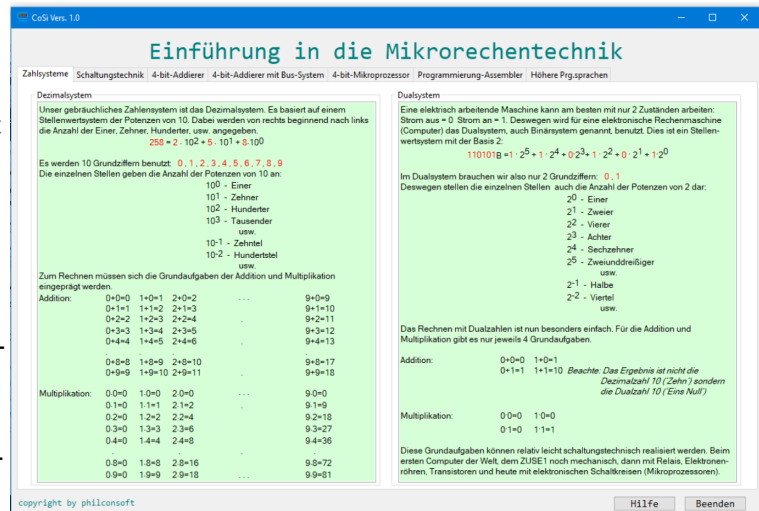


## Kurzbeschreibung des Programms ‚Cosima‘

Schon in den 80' er Jahren entstand die Idee, den Aufbau und die Wirkungsweise eines Computers auch graphisch programmtechnisch nachzubilden. Erste Ideen wurden mit dem Heimcomputer KC 85, dann mit dem Schulcomputer BIC realisiert, aber nie zu Ende geführt.

Erst jetzt, nach dem Ende des aktiven Berufslebens, habe ich mich wieder damit beschäftigt. Ziel war es, ein Programm zu entwickeln, das im Unterricht an allgemeinbildenden und beruflichen Schulen eingesetzt werden kann, um die Funktion einer automatisch arbeitenden, elektronischen Rechenmaschine zu demonstrieren. Es sollte modular aufgebaut sein und dem Nutzer eigene Aktivitäten ermöglichen. Entstanden ist die in Pascal mit Hilfe des Lazarus-Projektes geschriebene Software ‚Cosima‘ (‚ComputerSimulationsMaschine‘).



## 1. Aufruf und Bedienung

Das Programm sowie die dazugehörigen Beispiel-Dateien sollten in einem eigenen Ordner gespeichert sein. Durch Aufruf von ‚Cosima\_Vers\_1\_0.exe‘ wird es gestartet. Die Auswahl der einzelnen Kapitel erfolgt über die Reiter; durch Drücken der Hilfe-Taste werden kapitelspezifische Hinweise eingeblendet. Die Kapitel sind so konzipiert, dass man bei chronologischer Abarbeitung einen kurzen Einblick in die Anfänge der Mikrorechentechnik bekommt. Natürlich kann man einzelne auch überspringen.

## 2. Der Modellprozessor

Die Computertechnik wird am Beispiel eines 4-bit-Mikroprozessors (Modellprozessor) mit einem 8-bit breitem Adressbus erläutert. Aus diesem Grunde stehen lediglich 16 Elementarbefehle zur Verfügung und Programme können in maximal 256 Speicherzellen abgelegt werden. Die Vorgänge im Steuerwerk werden allerdings nur symbolisiert nachgebildet, obwohl die einzelnen Schritte bei der Ausführung eines Maschinenbefehls auch dargestellt werden. Da der Prozessor taktgesteuert arbeitet und man die Arbeitsweise eventuell auch als Lehrer erläutern oder beim selbständigen Erarbeiten durch Schüler nachvollziehen möchte, gibt es 3 Betriebsarten: Steuerung durch Einzeltakt, Steuerung durch einen Taktgenerator I, der mit geringer Frequenz arbeitet und Steuerung durch einen mit höherer Frequenz arbeitenden Taktgenerator II.

Die automatische Abarbeitung durch den Taktgenerator kann durch eine Stop-Taste unterbrochen werden. Durch Drücken der Weiter- und Takt-Taste wird das Programm fortgesetzt.

Als Eingabe-Gerät wird ein Tastenfeld zur Eingabe einer 4-stelligen Dualzahl benutzt, das erst bei Abarbeitung eines IN-Befehls eingeblendet wird; Ausgabegeräte sind einerseits 4 Leuchtdioden am Datenbus, andererseits ein symbolischer Monitor.

Folgende Maschinenbefehle sind implementiert:

Maschinen-Code	Mnemomik	Erläuterung
0000	HLT	Stoppt die Programmabarbeitung bis Button ‚Weiter‘ gedrückt
0001	IN A	Eingabe einer Zahl in das Register A; ein Eingabepaneel wird eingeblendet
0010	OUT A	Inhalt des Registers A wird auf das Ausgabepaneel und dem Monitor ausgegeben
0011	ADD B	Der Inhalt des Registers B wird zum Inhalt des Registers A addiert
0100	SUB B	Der Inhalt des Registers B wird vom Inhalt des Registers A subtrahiert
0101	LD A,n	Das Register A wird mit der Zahl n beschrieben, die in der nachfolgenden Speicherzelle stehen muss
0110	LD A,(nn)	Das Register A wird mit dem Inhalt der durch nn adressierten Speicherzelle geladen
0111	LD (nn),A	Die Speicherzelle mit der Adresse nn wird mit dem Inhalt des Registers A beschrieben
1000	MOV B,A	Das Register B wird mit dem Inhalt des Registers A geladen
1001	MOV A,B	Das Register A wird mit dem Inhalt des Registers B geladen
1010	JMP nn	Unbedingter Sprung zur Speicheradresse nn, d.h. der Befehlszähler wird mit nn geladen
1011	JNZ nn	Bedingter Sprung zur Speicheradresse nn, wenn das Zero-Flag nicht gesetzt ist
1100	JC nn	Bedingter Sprung zur Speicheradresse nn, wenn das Carry-Flag gesetzt ist
1101	AND B	Der Inhalt des Registers A wird mit dem Inhalt des Registers B bitweise AND-verknüpft; (A):=(A) and (B)
1110	RRC	Rotation der Bit-Stellen des Registers A um eine Stelle nach rechts; die erste Stelle wird auch in das C-Flag kopiert
1111	RLC	Rotation der Bit-Stellen des Registers A um eine Stelle nach links; die letzte Stelle wird auch in das C-Flag kopiert

Ein Maschinenprogramm (\*.MPR) steht beim Start des Modellrechners im Speicher. Es berechnet die Aufgabe  $(5+4)*2$ . Weitere Programme, auch selbst erstellte, können geladen werden. Maschinenprogramme können nur die 16 Elementarbefehle nutzen und müssen als Textdatei mit jeweils ei-

nem Befehl pro Zeile abgespeichert sein. Vergleiche dazu auch die Beispielprogramme. Maschinenprogramme werden am besten mit dem sich auf der Seite ‚Programmierung Assembler‘ befindlichen Assembler für den Modellcomputer erstellt. Die Übersetzung in den Maschinencode hat zwei Modi:

1. Die Argumente der Befehle werden neben den Befehlskode gestellt (Standard). Dies verbessert die Lesbarkeit.
2. Die Argumente der Befehle werden untereinander geschrieben, so wie es der Modellcomputer verlangt.

Auch der Assembler ist nur ein Modell-Assembler, der eigentlich nicht anderes macht, als die Maschinenbefehle zu übersetzen und die Dezimalzahlen in Dualzahlen umzuwandeln. Insbesondere werden keine Adressberechnungen vorgenommen, so dass keine symbolische Adressierung (mit Marken) benutzt werden kann.

### 3. Beispielprogramme

Um die universelle Nutzbarkeit eines Computers zu demonstrieren, wurden verschiedene Anwendungen realisiert. Dies sind ein Count-Down-Zähler, der von 10 bis 0 herunterzählt, ein Zahlenrate-spiel, ein Mittelwertbilder aus 4 einzugebenden Zahlen und aus dem Bereich Steuerungstechnik eine einfache Fahrstuhlsteuerung. Als Hauptanwendung früherer 4-bit-Prozessoren wurde ein einfaches Taschenrechnerprogramm realisiert, das mit BCD-Zahlen rechnet.

#### 3.1. Count-Down-Zähler

In Pascal würde ein Count-Down-Zähler so formuliert werden:

```
Zähler := 10;
repeat
  Zähler := Zähler - 1 ;
  writeln(Zähler);
until Zähler = 0;
```

Eine Umsetzung für den Modellcomputer in der Assemblersprache wäre:

Adresse	Marken	Mnemonic	Erläuterung
0		LD A,1	Schrittweite
2		MOV B,A	retten
3		LD A,10	Zähler Anfangswert
5	M1:	SUB B	Zähler:=Zähler-1
6		OUT A	Ausgeben
7		JNZ M1	wenn nicht 0
10		HLT	
		END	

### 3.2. Zahlenraten

In Pascal:

```
const Computerzahl = 7;
var Ratezahl: integer;
begin
  repeat
    readln(Ratezahl);
    writeln(Ratezahl);
  until Ratezahl = Computerzahl;
end;
```

Assembler:

Adresse	Marken	Mnemonik	Erläuterung
0		JMP Start	
3		DB 7	zu ratende Zahl
4	Start:	LD A,(3)	
7		MOV B,A	in B retten
8	M1:	IN A	Rateversuch
9		OUT A	ausgeben
10		SUB B	testen
11		JNZ M1	nicht geraten
14		HLT	
		END	

Die Umsetzung in den Maschinencode würde so aussehen:

Adresse	Code	Mnemonik
00000000	1010	JMP 4
00000001	0100	
00000010	0000	
00000011	0111	DB 7
00000100	0110	LD A,(3)
00000101	0011	
00000110	0000	
00000111	1000	MOV B,A
00001000	0001	IN A
00001001	0010	OUT A
00001010	0100	SUB B
00001011	1011	JNZ 8
00001100	1000	
00001101	0000	
00001110	0000	HLT

### 3.3. Mittelwert aus 4 Zahlen

Adresse	Code	Mnemonik		Adresse	Code	Mnemonik
00000000	0001	IN A		00000111	1000	MOV B,A
00000001	1000	MOV B,A		00001000	0001	IN A
00000010	0001	IN A		00001001	0011	ADD A
00000011	0011	ADD A		00001010	1110	RRC
00000100	1000	MOV B,A		00001011	1110	RRC
00000101	0001	IN A		00001100	0010	OUT A
00000110	0011	ADD A		00001101	0000	HLT

### 3.4. Fahrstuhlsteuerung

Um mit den beschränkten Mitteln des Modellcomputers eine Fahrstuhlsteuerung zu simulieren, wurde von einem Haus mit 2 Stockwerken ausgegangen. Als Eingabeelemente wurde das Schalterpaneel und als Ausgabekanäle die Leuchtdioden am Datenbus benutzt.

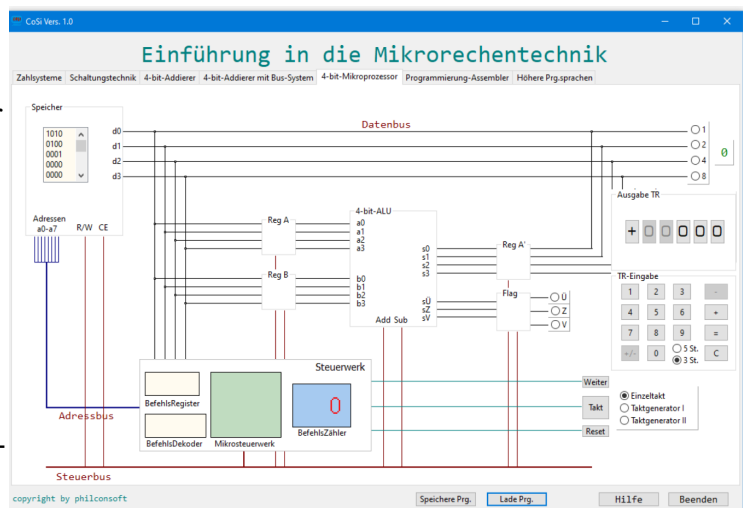
Vereinbarungen:			
Schalter	Bedeutung	LED	Bedeutung
1	Ruf ins 1. Stockwerk	1	befindet sich im 1. Stockwerk
2	Ruf ins 2. Stockwerk	2	befindet sich im 2. Stockwerk
4	angekommen im 1. Stock	4	Motor fährt hoch
8	angekommen im 2. Stock	8	Motor fährt runter

Adresse	Marken	Mnemonic	Erläuterung
0	Init:	LD A,4	Anfangsbedingung: FS im 1. Stock
2		MOV B,A	
3	M1:	IN A	
4		AND B	FS im 1. Stock?
5		JNZ 14 (M2)	ja
8		LD A,8	nein
10		OUT A	Motor runter
11		JMP 3 (M1)	bis angekommen
14	M2:	LD A,1	im 1.Stock
16		OUT A	LED 1 an
17	M0:	IN A	Abfrage Tasten
18		MOV B,A	retten
19		LD A,6	FS im 1. Stock+ Ruf 2.Stock
21		SUB B	
22		JNZ 28 (M3)	nein
25		LD A,4	ja
27		OUT A	Motor hoch
28	M3:	LD A,9	im 2.Stock+ Ruf 1.Stock
30		SUB B	
31		JNZ 37 (M4)	nein
34		LD A,8	ja
36		OUT A	Motor runter
37	M4:	LD A,4	nur im 1.Stock?
39		SUB B	
40		JNZ 46 (M5)	nein
43		LD A,1	ja
45		OUT A	LED 1 an
46	M5:	LD A,8	nur im 2.Stock?
48		SUB B	
49		JNZ 55 (M6)	nein
52		LD A,2	ja
54		OUT A	LED 2 an
55	M6:	JMP 17 (M0)	wieder von vorn
58		HLT	
		END	

### 3.5. Einfacher Taschenrechner

Eigentlich sollte zur Demonstration der Möglichkeiten dieses einfachen Modellprozessors ein Taschenrechnerprogramm, das die Addition und Subtraktion von 5-stelligen ganzen Dezimalzahlen realisiert, entstehen. Das wäre vom Befehlsumfang her auch möglich gewesen, scheiterte aber an der begrenzten Anzahl von adressierbaren Speicherplätzen (256). Aus diesem Grund wurde im Maschinenprogramm nur die Addition von 3-stelligen positiven ganzen Dezimalzahlen realisiert. Die ursprüngliche Aufgabenstellung ist dennoch implementiert, allerdings in Pascal.

Nach dem Laden des Taschenrechnerprogramms (TR\_Vers\_1.MPR) wird der Modellrechner durch ein Taschenrechnerpaneel, das die Eingabetastatur und die Ziffernanzeige enthält, ergänzt. Wird die 3-stellige Variante gewählt (Standard), werden alle nicht nutzbaren Elemente ausgeblendet. Es sollte der Taktgenerator II eingestellt sein. Die Abarbeitungsgeschwindigkeit ist dennoch gering. Wenn Sie also eine Taste drücken, müssen Sie solange warten, bis das Programm sich wieder in der Eingabeschleife befindet, bevor Sie die nächste Taste betätigen.



Erkennbar ist dies daran, dass in den Registern A und B ein Wechsel von '15' und '0' stattfindet.

Um überhaupt in den 256 Speicherzellen das Grundprinzip eines Taschenrechnerprogramms demonstrieren zu können, wurden der IN- und der OUT-Befehl beim Laden des TR-Programms modifiziert. Die Tastatur des Taschenrechners besitzt eine eigene Logik, die jeder Taste einen Code zuordnet (Ziffern: ,0'–,9'; Funktionstasten: ,10'–,14'). Dieser wird beim IN-Befehl übergeben. Ist keine Taste gedrückt, übergibt die Tastatur den Code '1111' also ,15'.

Der OUT-Befehl geht ebenfalls von einer eigenen Logik in der Ansteuerung der Ziffernanzeige aus. Diese liest die Ziffern im Speicher je nach Zustand für die ZahlA, die ZahlB oder das Ergebnis aus und gibt sie stellenrichtig an die Ziffernanzeige. (Eigentlich hätte dazu nur der Anzeigespeicher dienen sollen, aber aus Kapazitätsgründen des Speichers konnte die dann notwendige Umspeicherung nicht vorgenommen werden.)

Auch der Löschbefehl ,C' ist auf diese Weise realisiert: In der Speicherzelle 17 wird der Funktionscode abgelegt. Ist dieser ,14' werden alle Zahlenspeicher gelöscht, also die Anfangsbedingungen wieder hergestellt und angezeigt.

Adresse	Marken	Mnemonic	Erläuterung
		REM TaschenrechnerDemo 3 Digit	
		REM c.by philconsoft	
0		JMP Start	
3		DB 0000	Speicher ZahlA
4		DB 0000	
5		DB 0000	
6		DB 0000	Speicher ZahlB
7		DB 0000	
8		DB 0000	
9		DB 0000	Speicher Ergebnis
10		DB 0000	
11		DB 0000	

12		DB 0000	AnzeigeSpeicher
13		DB 0000	
14		DB 0000	
15		DB 15	TastenSpeicher
16		DB 1	ZahlZähler
17		DB 0000	OperationSpeicher
18		DB 0000	OperationSpeicherAlt
19		DB 0000	Überlauf
20	Start:	IN A	
21		MOV B,A	Tastencode retten
22		LD A,15	
24		SUB B	
25		JNZ M1	Taste gedrückt
28		JMP Ausgabe1	Taste nicht gedrückt
31	M1:	LD A,9	Ziffer?
33		SUB B	
34		JC M3	keine Ziffer
37		LD A,(16)	ZahlZähler?
40		MOV B,A	
41		LD A,1	
43		SUB B	
44		JNZ M2	in ZahlB
47		LD A,(4)	in Zahl A
50		LD (5),A	
53		LD A,(3)	
56		LD (4),A	
59		LD A,(15)	hole Tastencode
62		LD (3),A	
65		JMP Ausgabe	
68	M2:	LD A,(7)	in ZahlB
71		LD (8),A	
74		LD A,(6)	
77		LD (7),A	
80		LD A,(15)	hole Tastencode
83		LD (6),A	
86		JMP Ausgabe	
89	M3:	LD A,11	Taste '+' gedrückt?
91		SUB B	
92		JNZ M4	nein
95		LD A,11	ja
97		LD (17),A	Operation sichern
100		LD A,(16)	ZahlZähler?
103		MOV B,A	
104		LD A,1	
106		SUB B	
107		JNZ Add0	Addition ausführen
110		LD A,2	ZahlZähler:=2
112		LD (16),A	
115		JMP Ausgabe	
118	Add0:	LD A,(6)	ZB(1)
121		MOV B,A	
122		LD A,(3)	ZA(1)
125		ADD B	
126		LD (9),A	ZE(1)
129		LD A,6	
131		MOV B,A	
132		LD A,(9)	ZE(1)
135		ADD B	ZE+6 (DAA-Nachbildung)
136		JC Add1	>9, Übertrag in 2.St
139		JMP Add2	
142	Add1:	LD (9),A	ZE:=ZE+6
145		LD A,1	
147		MOV B,A	

148		LD A,(4)	ZA(2)
151		ADD B	
152		LD (4),A	
155	Add2:	LD A,(7)	ZB(2)
158		MOV B,A	
159		LD A,(4)	ZA(2)
162		ADD B	
163		LD (10),A	ZE(2)
166		LD A,6	
168		MOV B,A	
169		LD A,(10)	
172		ADD B	ZE+6 (DAA?)
173		JC Add21	>9, Übertrag in 3.St
176		JMP Add3	
179	Add21:	LD (10),A	ZE:=ZE+6
182		LD A,1	
184		MOV B,A	
185		LD A,(5)	ZA(3)
188		ADD B	
189		LD (5),A	
192	Add3:	LD A,(8)	ZB(3)
195		MOV B,A	
196		LD A,(5)	ZA(3)
199		ADD B	
200		LD (11),A	ZE(3)
203		LD A,6	
205		MOV B,A	
206		LD A,(11)	
209		ADD B	ZE+6 (DAA?)
210		JC Add31	>9, Überlauf!!
213		JMP Ausgabe	
216	Add31:	LD A,1	ZE:=ZE+6
218		LD (19),A	Überlauf:=1
221		JMP Ausgabe	
224	M4:	LD A,12	Taste '=' gedrückt?
226		SUB B	
227		JNZ M5	nein
230		JMP Add0	ja, Addieren
233	M5:	LD A,14	Taste 'C' gedrückt?
235		SUB B	
236		JNZ Ausgabe	nein
239		LD A,14	ja, alles löschen
241		LD (17),A	
244	Ausgabe:	LD A,15	
246		LD (15),A	
249	Ausgabe1:	OUT A	
250		JMP Start	
253			



#### **4. Anmerkungen**

Da das Programm den allgemeinbildenden und beruflichen staatlichen Schulen frei als ausführbare Datei (\*.EXE) überlassen wird, kann es unter normalen Umständen nicht geändert, bzw. an eigene Bedürfnisse angepasst werden. Insbesondere ist es so nicht möglich den Befehlssatz zu ändern oder z.B. andere Ein-, Ausgabegeräte zu verwenden. Wenn dies dennoch gewünscht ist, kann der Quellcode angefordert werden. Es ist natürlich auch möglich, mir die Änderungswünsche zu beschreiben und ich erstelle die entsprechenden Korrekturen. Entsprechende Anfragen sind an [philconsoft@gmx.de](mailto:philconsoft@gmx.de) zu richten.