Weiterbildungslehrgang X "Informatik" für Gymnasien
Kurs 3

IFB-Veranstaltung Nr. 61 177 05 01
vom 8.5.2006 - 12.5.2006 in Speyer

Materialien zum Rahmenthema "Assembler/Rechnerarchitektur"

Klaus Merkert, Kaiserslautern

Über diese Materialien hinaus finden Sie unter der URL

hsg.region-kaiserslautern.de/faecher/inf/material/bonsai

Download-Möglichkeiten für das Simulationsprogramm, für das ausführliche Handbuch sowie eine Online-Fassung des Handbuchs, die immer wieder aktualisiert wird.

arithmetische Befehle:

INC adr Die Speicherzelle mit der Adresse «adr» wird um 1 erhöht (inkrementiert). Anschließend wird der Befehlszähler um 1 erhöht.

PASCAL:

Register[adr] := Register[adr]+1;
PC := PC+1

DEC adr Die Speicherzelle mit der Adresse «adr»wird um 1 erniedrigt (dekrementiert). Anschließend wird der Befehlszähler um 1 erhöht.

PASCAL:

Register[adr] := Register[adr]-1;
PC := PC+1

Steuerbefehle:

JMP adr Unbedingter Sprung (jump) zur Speicherzelle mit der Adresse «adr», dh. der Programmzähler (program counter, PC) wird mit «adr» geladen.

PASCAL:

PC := adr

TST adr Bedingter 'Sprung' in Abhängigkeit vom Inhalt der Speicherzelle mit der Adresse «adr», genauer: Teste (test) die Speicherzelle mit der Adresse «adr» auf 0, wenn nein, erhöhe den Pro-grammzähler (PC) um 1, wenn ja um 2.

PASCAL:

IF Register[adr] = 0
THEN PC := PC+2
ELSE PC := PC+1

sonstige Befehle:

HLT Halt, Programmende

PASCAL:

halt

Bemerkung : halt bricht in Turbo-Pascal die Ausführung eines Programms ab und führt einen Rücksprung ins Betriebssystem durch. Die Prozedur ist im ANSI-Pascal nicht definiert, kann aber durch einen Sprung zu einer Marke am Programmende ersetzt werden: (....goto Ende; .....Ende:end.).

Erarbeitung der Bonsai-Befehle und einiger Begriffe

Arbeitsauftrag 1

1. JMP, Programmzähler PC, Takt, Editieren

- Starten Sie das BONSAI-Programm mit bonsai ↵
- ❒ Wählen Sie *Editieren - Programm* (Alt E/P)

Es erscheint das Programmeditierfenster.

- Geben Sie folgendes Programm ein

```
1   JMP 2
2   JMP 1
```

- Beenden Sie die Eingabe mit Strg-↵ (Ctrl-↵.

Auf dem Bildschirm erscheint nun das Programm.

- Wählen Sie jetzt *Programmlauf - ohne CPU-Simulation* (Alt G/O)

Es erscheint ein Dreieck (<) auf dem Bildschirm, das immer auf den nächsten auszuführenden Befehl zeigt. Das Dreieck heißt **Befehlszähler (program counter, PC)**.

- Drücken Sie zur Ausführung des Befehls jeweils die Taste 'C' (**clock, Takt**) und verfolgen Sie das Programm.

Sie haben jetzt gesehen, was der JMP-Befehl bewirkt und den PC kennengelernt.

- Verlassen Sie nun die Option BONSAI-Lauf (unten links angezeigt) mit der Anwahl eines anderen Menüpunkts oder mit ESC.

2. INC, Datenregister

- Geben Sie nun folgendes Programm ein.
- INC 1
- JMP 1
- Lassen Sie es wie oben laufen.

Sie erhalten eine Fehlermeldung, weil Sie das Datenregister 1 nicht initialisiert haben, holen Sie das nach.

- Wählen Sie *Editieren - Daten editieren* (Alt E/D)

Es erscheint das Dateneditierfenster.

- Geben Sie z.B. im Datenregister 1 eine 0 ein !
- Beenden Sie die Eingabe mit Strg-↵ (Ctrl-↵.).
- Lassen Sie das Programm ausführen.

Sie wissen jetzt, was ein Datenregister ist und was der INC-Befehl bewirkt.

3. DEC

- Erforschen Sie analog, was der DEC-Befehl leistet.

4. HLT, Kommentar editieren

- Geben Sie das Programm

```
1    TST 1
2    DEC 1
3    JMP 1
```

ein, füllen Sie das Datenregister 1 mit einer Zahl <> 0 und führen Sie das Programm durch.

Was leistet das Programm anscheinend ?
Es hat wie alle bisher vorgestellten Programme einen Makel, es endet nicht.

- Verwenden sie den HLT-Befehl und verbessern Sie obiges Programm
- Wählen Sie *Editieren - Kommentar* editieren (Alt E/K) und beschreiben Sie das Programm knapp, vor allem die Registerbelegung

5. Bonsai-Programme laden und speichern

- Speichern Sie mit *Datei - BONSAI-Programm speichern* (Alt D/S) ab. Wählen Sie dabei einen eigenen Namen.
- Laden Sie Ihre Datei mit *Datei - BONSAI-Programm laden* (Alt D/L) wieder ein.

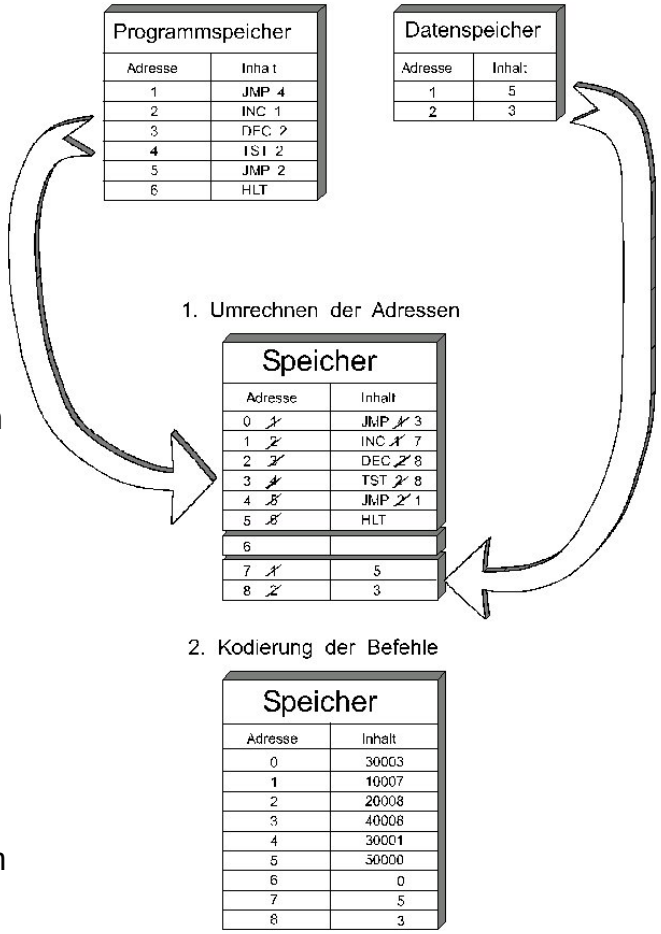
Vom BONSAI-Programm (Assembler-Sprache) zur Maschinensprache

Ein **Maschinenbefehl** besteht aus Operationscode und Adresscode.

Die Operation (der Befehl) wird der Einfachheit und Übersichtlichkeit wegen dezimal verschlüsselt:

INC	→	1. . . .
DEC	→	2. . . .
JMP	→	3. . . .
TST	→	4. . . .
HLT	→	50000

Programm- und Datenspeicher werden nun in einen Speicher vereinigt (von-Neumann-Prinzip). Die Anfangsadresse dieses Speichers ist willkürlich, man kann sie - den Hardware-Gepflogenheiten folgend - auf Null legen. Das Programm kann nun irgendwo im Speicher liegen, wir lassen es z.B. bei Null beginnen. Die Daten (die Lage des 'Datensegments' (DOS) ist ebenfalls willkürlich) haben wir an den Programmteil angehängt, dabei lassen wir zur optischen Trennung eine Speicherzelle frei (ebenfalls willkürlich!).
Die beschriebene dezimale Kodierung wird im BONSAI-Programm beim Assemblieren automatisch durchgeführt.
Ein Programm, das eine solche Übersetzung durchführt, heißt ebenfalls **Assembler.**



Handübersetzung in Maschinenprogramm

Arbeitsauftrag 2

- Geben Sie folgendes Programm Original-Know-How-Subtraktions-Programm (bis auf die Bonsai-Notation) auf der Bonsai-Programm-Ebene ein:

1 JMP 4 2 DEC 1 3 DEC 2 4 TST 1 5 JMP 7 6 JMP12 7 TST 2 8 JMP 2 9 HLT 10 INC 1 11 DEC 2 12 TST 2 13 JMP10 14 INC 2 15 HLT		Zitat aus der Know-How-Vorlage: "Und so arbeitet das Programm: Das Programm erwartet zwei (positive) Zahlen in Register 1 und in Register 2 und zieht die Zahl in Register 2 von der in Register 1 ab. Mit den Befehlen von 1 bis 8 werden die Register 1 und 2 zunächst herabgezählt. Wird Register 2 eher Null als Register 1, dann ist alles in Ordnung, in Register 1 bleibt die Differenz der beiden Zahlen stehen, die Maschine stoppt in 9. Tritt aber der Fall ein, dass der Subtrahend in Register 2 größer oder gleich dem Minuenden in Register 1 ist, dann führt der Befehl in Zeile 6 in den zweiten Programmteil. in dem nun Register 1 ebensooft wieder heraufgezählt wird wie Register 2 bis zu Null herabgezählt werden kann. Danach wird zur Kennzeichnung, dass der sich ergebende Differenzbetrag in Register 1 Null ist oder negativ, das Register 2 um Eins heraufgezählt. Das Programm merkt also in Register 2 das Vorzeichen an."

- Übersetzen Sie das Programm von Hand in Maschinensprache. Lassen Sie das Programm ab Speicherzelle 3 beginnen. Legen Sie die Daten in die Zellen 20 und 21. Es soll ja die Adress-Umrechnung geübt werden!

- Wechseln Sie in die CPU-Ebene, geben Sie das Maschinenprogramm ein und speichern Sie es z.B. unter dem Namen khsub.bma ab.

Ausführen eines Maschinenprogramms

Eine Übersetzung in Maschinensprache hat wenig Sinn, wenn keine Maschine zur Verfügung steht, die diese Sprache "spricht".

Das Bonsai-Simulations-Programm stellt mit der "CPU-Ebene" eine solche "Maschine" dar.

Arbeitsauftrag 3

Lassen Sie diese Maschine nun arbeiten:

- Zuerst müssen Sie für ein funktionierendes Maschinenprogramm (*.bma) sorgen.

- Das kann man automatisch erzeugen lassen:

- Laden Sie ein Bonsai-Programm in der Programm-Ebene ein, z.B. addition.bon und wechseln Sie mit Programmlauf/mit CPU-Simulation zur CPU-Ebene.

- Hier können Sie mit CPU-Lauf/Assembler den eingebauten Assembler benutzen, um das Maschinenprogramm zu erzeugen.

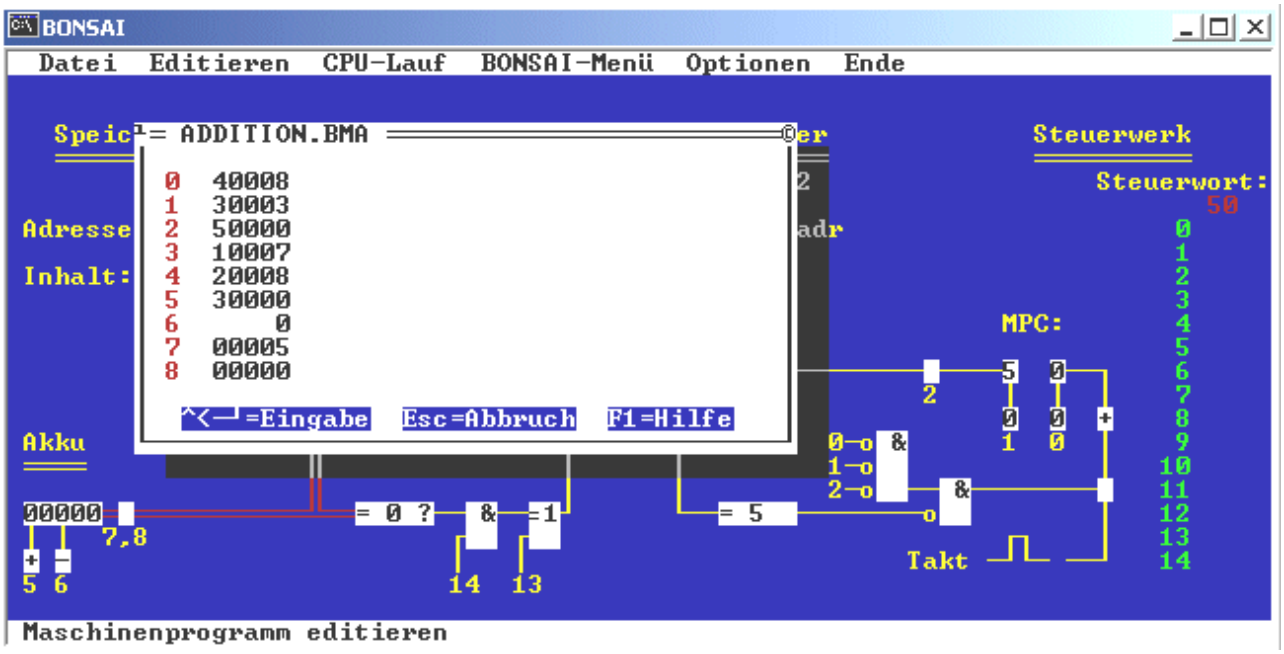
- Sehen Sie sich mit Editieren/Maschinensprache editieren das Programm an.

- Um das Maschinenprogramm durchführen zu können, müssen Sie auch für ein Mikroprogramm (*.bmi) sorgen. Laden Sie dazu mit Datei/Mikroprogramm laden das Standard-Mikroprogramm "mikropro.bmi" ein. Ein wirklicher Prozessor hat natürlich ein fest eingebautes Mikroprogramm, in der Simulation müssen Sie es erst laden.

- Mit CPU-Lauf/CPU_Lauf können Sie nun die Maschine laufenlassen.

- Wählen Sie ruhig Dauertakt d, denn Einzelheiten der CPU sollen erst später erläutert werden. Die Maschine bleibt stehen, wenn der Befehl 5.... zur Ausführung kommt.

- Schauen Sie mit Editieren/Maschinenprogramm editieren im Speicher nach, ob die Maschine richtig gerechnet hat.



- Kür: Lassen Sie das Programm khsub.bma laufen. Stellen Sie dazu nach dem CPU-Reset den 'richtigen' Wert für den PC ein.

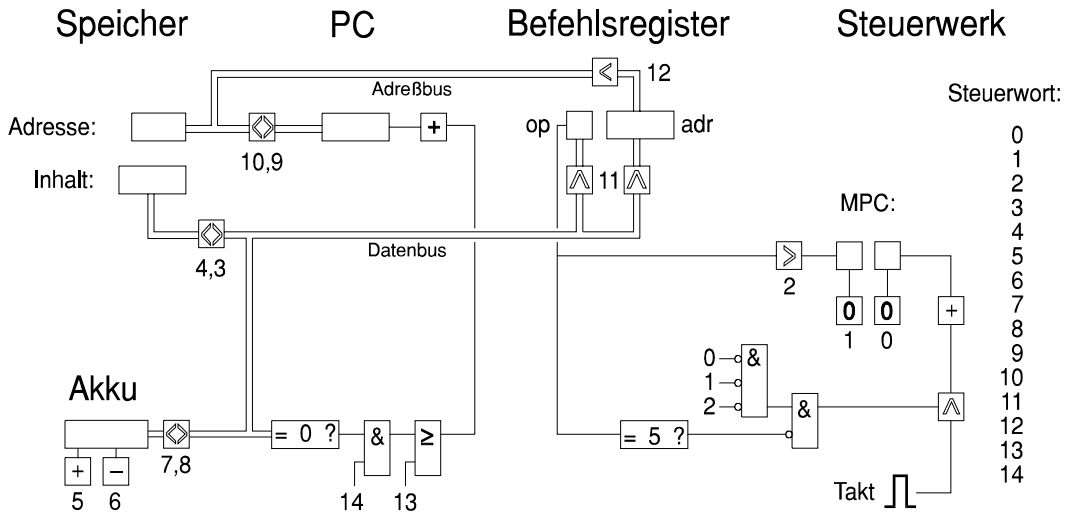
Die Architektur des BONSAI-Computers

Arbeitsauftrag 4

Die Architektur des BONSAI-Computers wird völlig durch den CPU-Bildschirm dargestellt:

- Schalten Sie mit ***Programmlauf - mit CPU-Simulation*** (Alt G/M) auf den CPU-Bildschirm um.

Sollte das nicht gelingen, so haben Sie das Programm ohne den Parameter 'e' gestartet, verlassen Sie BONSAI (Alt F4) und starten Sie mit 'bonsai e' bzw. 'start.bat' neu.
Sie sehen jetzt folgendes Bild:



Der beste Weg, Klarheit in dieses vielleicht doch zunächst komplexe CPU-Schema zu bringen, ist das Kennenlernen der einzelnen Komponenten im Direktbetrieb.

- schalten Sie mit ***CPU-Lauf - Direktbetrieb*** (Alt C/D) den Direktbetrieb an.

Wir wollen zunächst etwas in den Speicher schreiben.

- ***Editieren - Maschinenprogramm editieren*** (Alt E/A) öffnet ein Fenster, in dem Sie das Maschinenprogramm editieren können. Schreiben Sie ein kleines Maschinenprogramm (eventuell das dezimal kodierte Additionsprogramm) hinein.

Wie kann man nun den Speicher ansprechen ? Zuerst muss eine Speicherzelle durch Anlegen einer Adresse ausgewählt werden. Das können Sie beispielsweise dadurch erreichen, dass Sie den **PC (program counter)** auf den Adressbus schreiben lassen:

- Drücken Sie die Taste A (Steuersignal 10)

Das Steuersignal '10' hat das 'Tor' vom PC hin zum Adressbus geöffnet, der PC 'schreibt' auf den Adressbus (Hardware: Der PC legt Spannungen an die Leitungen des Adressbusses). Das Feld, das mit 'Adresse' beschriftet ist, zeigt jeweils den momentanen Zustand des Adressbusses an. Der Adressbus liegt beständig am Speicher an, es gibt kein Adressregister. Verwechseln Sie also dieses Feld nicht mit einem Register. In einem Register können Sie etwas speichern, auf dem Adressbus nicht. Die Simulation macht es möglich, dass Sie in den Speicher hineinsehen können, ohne dass er explizit (mit den Steuersignalen 3 oder 4) angesprochen worden ist. In unserem Fall sehen Sie den Inhalt der angesprochenen Speicherzelle. Nehmen Sie die Adresse vom Adressbus, 'weiß' der Speicher nicht mehr, welche Zelle angesprochen ist und die Anzeige verschwindet. Probieren Sie das durch wiederholtes Drücken der Taste aus.
Wollen Sie eine andere Speicherzelle sehen, müssen Sie die Adresse ändern:

- Wählen Sie ***Editieren - Programmzähler PC editieren*** (Alt E/Z) und setzen Sie den PC nacheinander auf die Werte 1,2 und 3.

Falls das 'Tor' 10 zum Adressbus geöffnet ist, sehen Sie die Inhalte der entsprechenden Speicherzellen.Jetzt wollen wir den Inhalt der Speicherzelle 3 in den Akku transferieren:

- Legen Sie die richtige Adresse an.
- Lassen Sie den Speicher durch Setzen des Steuersignals 3 (Taste) auf den "Datenbus" schreiben.
- Lassen Sie den Akku durch Setzen des Steuersignals 7 (Taste) vom Datenbus lesen.

Sie haben gerade durch Benutzung gewisser Steuersignale einen "**Registertransfer**" durchgeführt.
Zeigen Sie nun, dass Sie etwas gelernt haben:

- Löschen Sie alle Steuersignale.
- Setzen Sie den Akku auf einen Wert z.B. 17 und transferieren Sie den Akkuinhalt in das Befehlsregister.

Sie bemerken, es gibt Register, die lesen und schreiben können, wie z.B. der Akku, solche, die nur lesen können, wie das Befehlsregister vom Datenbus, und solche, die nur schreiben können, wie der Adressteil des Befehlsregisters auf den Adressbus.
Es soll nun der erste Befehl des Maschinenprogramms aus dem Speicher in das Befehlsregister übertragen werden:

- Stellen Sie den PC auf Null und schreiben Sie seinen Inhalt als Adresse auf den Adressbus.
- Lassen Sie den Speicher auf den Datenbus schreiben und das Befehlsregister vom Datenbus lesen.

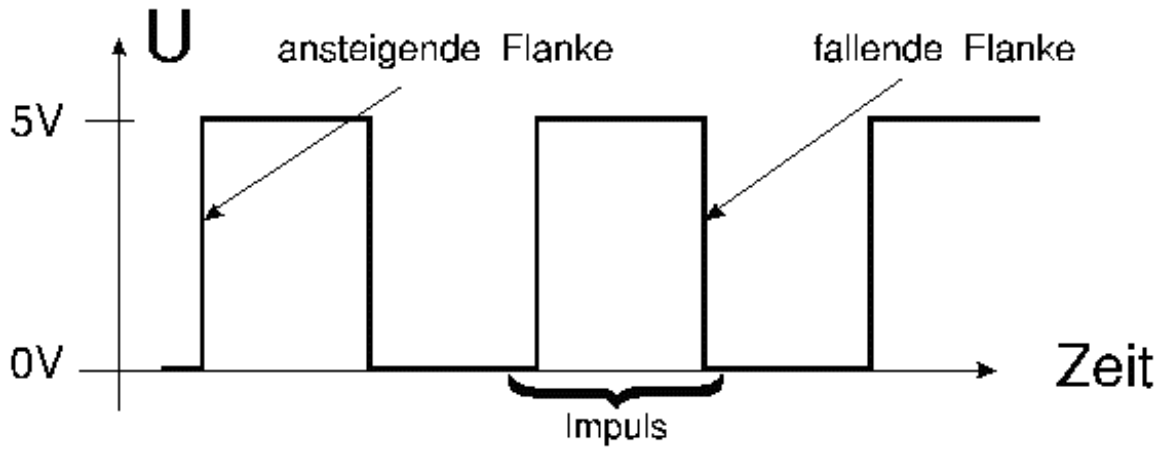
Der gerade durchgeführte Vorgang heißt "Befehl-Hole-Phase" (engl. **fetch-cycle**) und muss immer am Beginn der Abarbeitung eines Befehls stehen.
Das Befehlsregister enthält in seiner ersten mit "op" bezeichneten Stelle den operationscode (opcode), die restlichen Stellen enthalten eine Adresse. Nur diese Adresse kann mit dem Steuersignal 12 auf den Adressbus gelegt werden, der opcode-Teil des Befehlsregisters läßt sich über Tor 2 in die Zehnerstelle des Mikroprogrammzählers (MPC) transferieren.

- Öffnen Sie das Tor 2 (Anzeige: >) mit der Taste 2.
- Schließen Sie das Tor 2 (Anzeige:) mit der Taste 2.

Der **Mikroprogrammzähler** hat nicht von ungefähr einen ähnlichen Namen wie der PC, auch er zeigt auf den nächsten auszuführenden Mikrobefehl im Mikroprogramm. Eine Kombination von Steuersignalen nennt man "**Mikrobefehl**" oder - hier in der BONSAI-CPU - "**Steuerwort**". Das gerade aktuelle Steuerwort sehen Sie im CPU-Schema rechts. ACHTUNG: Im Direktbetrieb wird **nicht** wie eben beschrieben das Steuerwort aus dem Mikroprogramm geladen, sondern Sie setzen das Steuerwort selbst.

Sie sind das Steuerwerk, das das Operationswerk bedient.

Die Abfolge der Steuerworte wird durch den Takt gesteuert. Der Takt ist ein (Spannungs-) Signal, das zwei verschiedene Werte in regelmäßigen Abständen annimmt.



- Legen Sie mit T ein paar Taktimpulse an und sehen Sie, was passiert.

In der BONSAI-CPU ereignet sich mit ansteigender Flanke des Taktes zweierlei:

- Das neue Steuerwort wird aus dem Mikroprogrammspeicher übernommen. Diese Option ist im Direktbetrieb abgeschaltet.
- Der Mikroprogrammzähler wird um Eins hochgezählt. Jedoch nur, wenn der Taktimpuls das Tor zu dem -Symbol passieren kann.

Mit den beiden Funktionseinheiten 0, die mit den Steuersignalen 0 und 1 aktiviert werden, können Einer- und Zehnerstelle des Mikroprogrammzählers getrennt auf Null gesetzt werden.

- Benutzen Sie die Tasten 0, 1 und 2 in beliebigen Kombinationen. Achten Sie dabei auch auf das Tor für den Takt.

Sie werden bemerken, dass keines der Signale 0,1 und/oder 2 gesetzt sein darf, wenn der Takt zum MPC vordringen soll. Das ist aber gut verständlich, wenn man bedenkt, dass es wenig sinnvoll wäre, den MPC gleichzeitig auf bestimmte Werte zu setzen und hochzuzählen. Die beiden UND-Gatter mit den teilweise vorhandenen Invertern an den Eingängen haben also den Sinn, das Hochzählen des MPC im Setzbetrieb zu sperren und im Zählbetrieb zuzulassen. Außerdem sperren Sie im Verein mit der "5-Entdeckungslogik" = 5 ? das Zählen, wenn der HLT-Befehl (kodiert mit 5) im Befehlsregister steht. Auf diese Weise wird der BONSAI-Computer angehalten.

- Schreiben Sie mit Hilfe der Editierfunktion eine "5" in den opcode-Teil des Befehlsregisters und beobachten Sie die Auswirkung auf das oben genannte Tor.

Es ist an dieser Stelle keinesfalls notwendig, diese Logik in mehr als ihrer äußeren Funktion zu verstehen.
Als Letztes soll nun das bedingte und das unbedingte Hochzählen des PCs untersucht werden:

- Legen Sie mehrfach die Steuersignale 13 und 14 an.

Sie werden bemerken, dass jeweils mit der ansteigenden Flanke des Signals 13 der PC inkrementiert wird, das ist das unbedingte Hochzählen. Das Steuersignal 14 scheint jedoch ohne Wirkung zu bleiben.

- Schreiben Sie mit der Editierfunktion Null in den Akku und lassen Sie den Akku auf den Datenbus schreiben.

Sie sehen an dem Blinken der Anzeige [= 0 ?] , dass die Nullentdeckungslogik die Null auf dem Bus entdeckt hat. Was nicht zu sehen ist, der Ausgang der Nullentdeckungslogik hat jetzt den Wert 1 , so dass das UND-Gatter das Signal 14 jetzt durchlässt.

- Legen Sie mehrfach das Steuersignal 14 an.

Sie sehen jetzt, dass unter der Bedingung, dass eine Null auf dem Datenbus liegt, auch das Signal 14 den PC inkrementiert. Ebenso wenig wie bei der Steuerung des Takttors ist es an dieser Stelle nötig, sich über die verwendeten Logikbausteine Gedanken zu machen.

Bescheibung der Einzelheiten des CPU-Schemas

Die verschiedenen Busse kann man an den Farben erkennen:

- Adressbus (grün, oben, doppelt)
- Datenbus (rot, unten, doppelt)
- Steuerbus (gelbe Zahlen, einfach, nur z.T. eingezeichnet)

Die Steuerleitungen (Steuerbus) vom Steuerwerk zu den einzelnen Toren bzw. Funktionseinheiten sind nicht eingezeichnet, sondern nur durch Zahlen am Zielort symbolisiert. An Funktionseinheiten gibt es:

[+],[-]	bewirken mit steigender Taktflanke ein Auf- oder Abwärtszählen,
[0]	bewirkt ein 'auf Null setzen',
[= 0 ?]	gibt eine logische 1 aus, wenn auf dem Bus eine 0 liegt,
[= 5 ?]	gibt eine logische 1 aus, wenn auf dem Bus eine 5 liegt.

Die Richtung der Toröffnungen wird dabei durch die Lage der Zahlen bestimmt. Wenn das entsprechende Steuersignal gesetzt ist, fließen die Daten 'in Richtung der Zahl'.
Setzen/löschen Sie mit den Tasten 0..9,A..E (wir verwenden hexadezimale Ziffern, da wir so mit einer Taste auskommen, die Zuordnung können Sie rechts unten im Steuerwort sehen) oder mit der Maus die Steuersignale 0..14.

Sie werden sehen, dass viele Kombinationen von Steuersignalen zu Fehlern führen. Diese Fehler werden einschließlich eines Kommentars angezeigt und führen zu einem Löschen aller Signale.
Das Menü ***Editieren*** gibt Ihnen die Gelegenheit, sowohl Speicher- als auch Registerinhalte zu verändern.
Das Mikroprogramm sehen Sie nicht, sondern nur das gerade aktuelle Steuerwort. Der Mikroprogrammzähler MPC zeigt die Nummer des nächsten Steuerwortes an, das mit dem Takt aus dem Mikroprogrammspeicher in das Steuerwortregister übernommen wird.
Im Direktbetrieb wird kein Mikroprogramm benutzt, sondern das Steuerwort setzen Sie selbst.

Im Direktbetrieb sind Sie das Steuerwerk, das das Operationswerk bedient.

Arbeitsblatt zum CPU-Schema

Bekannte Bausteine eines Computers sind zu erkennen:

(1) _____

(a) _____

(b) _____

(c) _____

(d) _____

(2) _____

(e) _____

(f) _____

(g) _____

Verbindungen zwischen den Bausteinen:

(1) _____

(2) _____

(3) _____

Einige Aufgaben:

1. Der PC soll den Wert 0012 erhalten. (Zählen)
2. IR soll den Wert 4 0034 erhalten. (Editieren)
3. Laden Sie den Adressteil in den PC. (Torsteuerung)
4. Laden Sie den Akku mit 50003. (Editieren)
5. Ändern Sie den Inhalt des Akku auf 50002. (Dekrementieren)
6. Transferieren Sie den Akku-Inhalt in Speicherzelle 34!
7. Ändern Sie den Inhalt von Zelle 34 auf den Wert 50006!
8. Schreiben Sie in Zelle 30 den Wert 23456!
9. Laden Sie den Inhalt von Zelle 34 in das Befehlsregister!
10. Wann wird der PC um 1 erhöht? (1) _____
    (2) _____
11. Wie wird der MPC gesteuert?

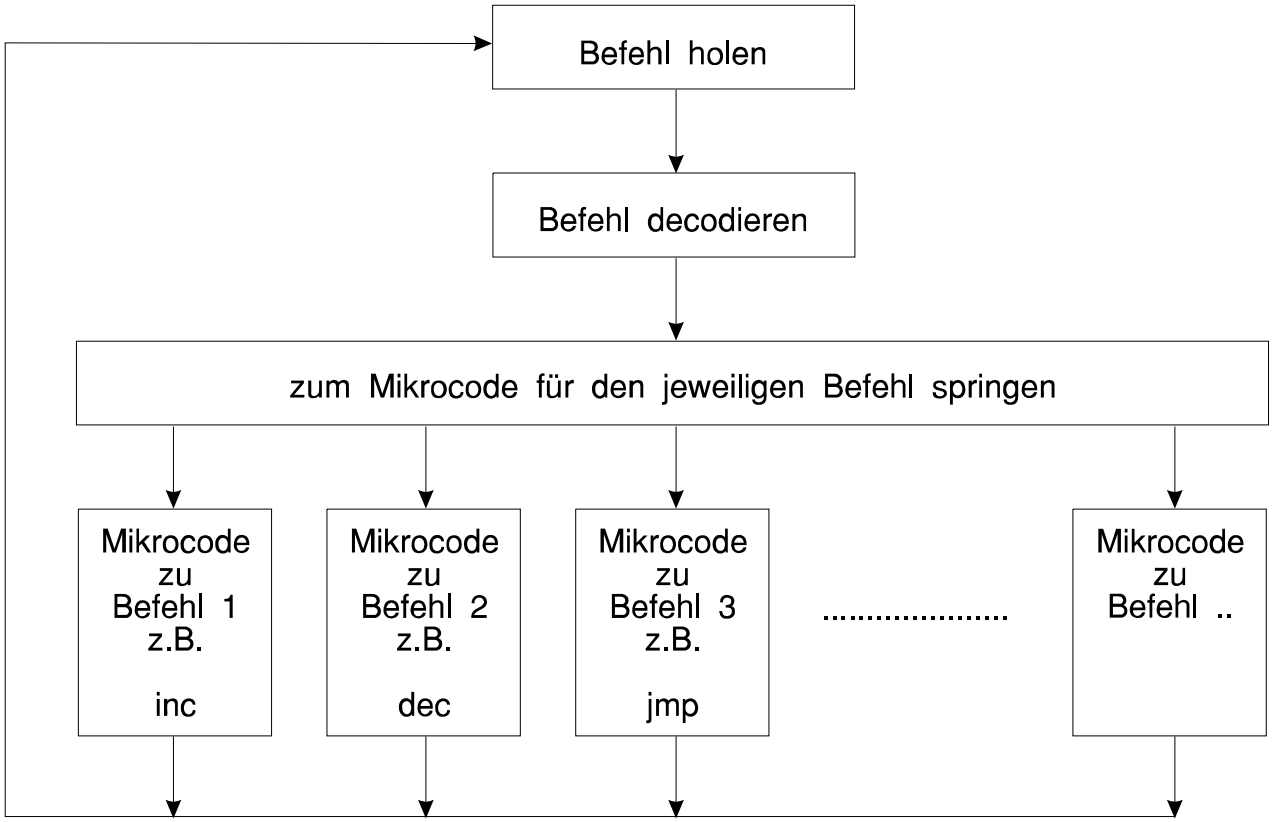
    die Zehnerstelle (1) _____
    (2) _____
    (3) Übertrag von der Einerstelle

    die Einerstelle (1) _____
    (2) _____

Mikroprogrammierung mit dem BONSAI-Programm

Ein Computer durchläuft endlos den

von-Neumann-Zyklus (Fundamentalzyklus):



"Befehl holen" bedeutet, dass der Inhalt der Speicherzelle, auf die der PC deutet, in das Befehlsregister (engl. instruction register IR) transportiert wird.
"Befehl decodieren" heißt im Rechnermodell, aus dem Opcode des Befehls die richtige Einsprungstelle für den Mikrocode des Befehls zu berechnen. Im BONSAI-Programm ist das so gelöst, dass beispielsweise die Mikrobefehle des Befehls 3 ab Speicherstelle 30 im Mikroprogramm stehen, die des Befehls 4 ab 40 und so weiter. Da der Mikroprogrammzähler MPC immer auf das nächste Steuerwort (Mikrobefehl) zeigt, muss zur Realisierung des Sprungs nur der Opcode des Befehls als Zehnerstelle genommen werden und die Einerstelle auf Null gesetzt werden.
Es darf nach der der Abarbeitung eines Befehls nicht vergessen werden, den Mikroprogrammzähler MPC auf Null zu setzen. So leitet dann das Steuerwerk einen neuen Befehl-Hole-Zyklus (engl. fetch-cycle) ein.
Beim Ansprechen des Speichers muss darauf geachtet werden, dass eine gültige Adresse anliegt. Das wird sicher erreicht , wenn man die Adresse einen Takt vor dem Ansprechen des Speichers anlegt.

Das Mikroprogramm 'mikropro.bmi'

Arbeitsauftrag 5

- Laden Sie ein Maschinenprogramm, z.B. 'addition.bma'
- Laden Sie das Mikroprogramm 'mikropro.bmi'
- Verfolgen Sie nach einem 'Reset' (CPU-Lauf - Reset) Takt für Takt (CPU-Lauf - CPU-Lauf) den Ablauf des Mikroprogramms für einzelne Befehle.
- Verlassen Sie das BONSAI-Programm und erstellen Sie nur mit dem CPU-Schema als Hilfe die Folge der Mikrobefehle für die einzelnen BONSAI-Befehle etwa nach folgendem Muster (| bezeichnet eine Taktgrenze):

  tst-Befehl:

  10 | 3 , 10 , 11 | 0 , 2 | 12 | 3 , 12 | 3 , 12 , 14 | 13 | 0 , 1

  | | | |
  Befehl holen | Befehl decodieren | Befehl ausführen | Rücksprung

  inc-Befehl:

  _____

  | | | |
  Befehl holen | Befehl decodieren | Befehl ausführen | Rücksprung

  dec-Befehl:

  _____

  | | | |
  Befehl holen | Befehl decodieren | Befehl ausführen | Rücksprung

  jmp-Befehl:

  _____

  | | | |
  Befehl holen | Befehl decodieren | Befehl ausführen | Rücksprung

  hlt-Befehl:

  _____

  | | | |
  Befehl holen | Befehl decodieren | Befehl ausführen | Rücksprung

Auftrag 5b

Schreiben Sie ein Mikroprogramm z.B. m.bmi , das das Maschinenprogramm

```
0    1 0007
1    5 0000
.......
7        13
```

erfolgreich durchführen kann. Testen Sie die Kombination.

Beispiele für Mikroprogrammierung:

Programmierung des Befehls 6 , lda adr , "lade den Akku mit dem Inhalt der Speicherzelle adr und erhöhe anschließend den PC":

- Editieren des Mikroprogramms:
  60:12 61: 12,3,7 62: 12,13,1,0
- Erstellen eines kleinen Test-Maschinenprogramms:

  teste6.bma:

  0 60003 lda 3
  1 50000 hlt
  2 00000 Lücke
  3 00027 Datum, das in den Akku geladen wird
- Austesten mit der Option 'CPU-Lauf' (vorher Reset)

Arbeitsauftrag 5c

Programmieren und testen Sie analog den Befehls 7 , sta adr , "speichere den Akkuinhalt in der Speicherzelle adr und erhöhe anschließend den PC":

Der BONSAI-Computer kann auch indirekt adressieren !

Programmierung des Befehls 6, lda (adr) , "lade den Akku mit dem Inhalt der Speicherzelle, deren Adresse in der Speicherzelle adr steht, und erhöhe anschließend den PC "

- Editieren des Mikroprogramms:

  60:12
  61:12,3,7
  62:8,11
  63:12,13
  64:12,7,3
  65:1,0
- Erstellen eines kleinen Test-Maschinenprogramms:

  indirekt.bma:

  0 60003 lda (3)

1	50000	hlt
2	00000	Lücke
3	00004	Adresse der Datums
4	00027	Datum, das in den Akku geladen wird

- Austesten mit der Option 'CPU-Lauf' (vorher Reset)

Arbeitsauftrag 5d

- Programmieren und testen Sie analog den Befehls 7 , INC (adr) , "inkrementiere den Inhalt der Speicherzelle, deren Adresse in der Speicherzelle adr steht, und erhöhe anschließend den PC".

Die Bonsai-Hardware

Binäre Kodierung

Der Einfachheit halber werden alle Befehle des BONSAI-Lehrcomputers nur mit einem Byte kodiert. Für vier Befehle (der HLT-Befehl wurde "wegrationalisiert", mehr dazu weiter unten) werden 2 Bit benötigt, so dass für den Adressteil 6 Bit übrigbleiben. Es können also Adressen zwischen $0_{10} = 000000_2$ und $63_{10} = 111111_2$ verwendet werden. Mehr-Wort-Befehle (ein 'Wort' ist in diesem Rechner ein Byte lang) werden nicht verwendet. Die Befehle sind einfach durchnummeriert:

$$\textbf{INC} \mathrel{\hat{=}} \textbf{00}_2,\ \textbf{DEC} \mathrel{\hat{=}} \textbf{01}_2,\ \textbf{JMP} \mathrel{\hat{=}} \textbf{10}_2,\ \textbf{TST} \mathrel{\hat{=}} \textbf{11}_2$$

Operationscode, opcode
|
«Befehl» = «op» «adr»
|
Adresse

zum Beispiel: 10 000101 bedeutet : JMP 5 (opcode=2, adr=5)

Ein kleines Beispiel für die Kodierung (Additionsprogramm):

Assembler:				Maschinenprogramm:	
				Adresse	Datum
0	JMP 3		Befehle	0000 0000	10 000011
1	INC 7			0000 0001	00 000111
2	DEC 8			0000 0010	01 001000
3	TST 8			0000 0011	11 001000
4	JMP 1			0000 0100	10 000001
5	JMP 5	*		0000 0101	10 000101
6	0		Lücke	0000 0110	00000000
7	3		Daten	0000 0111	0000 0011
8	2			0000 1000	0000 0010

Der * weist auf eine Besonderheit hin:
Der entfallene HLT-Befehl wird durch einen Sprung "auf sich selbst" ersetzt. Der Rechner wird also durch eine Endlosschleife abgefangen, deren Auftreten an einem stabilen Bitmuster im Be-

fehlsregister (hier: 10 000101) zu erkennen ist. Außerdem sieht man im "leuchtenden" Mikroprogramm, dass immer wieder nach dem fetch-cycle nur noch der JMP-Befehl abgearbeitet wird. Als weiteres Beispiel für die Binärkodierung sei die Übersetzung des weiter vorne systematisch aus PASCAL entwickelten Multiplikationsprogramms angeführt:

MULTIPLI.BMA → **MULTIPLI.BIN**

dezimal			binär		
0	40020		00000000	11 010100	← Befehle
1	30003		00000001	10 000011	
2	3 2	*	00000010	10 000010	
3	40019		00000011	11 010011	
4	30006		00000100	10 000110	
5	30010		00000101	10 001010	
6	20019		00000110	01 010011	
7	10021		00000111	00 010101	
8	10022		00001000	00 010110	
9	30003		00001001	10 000011	
10	40022		00001010	11 010110	
11	30013		00001011	10 001101	
12	30016		00001100	10 010000	
13	20022		00001101	01 010110	
14	10019		00001110	00 010011	
15	30010		00001111	10 001010	
16	20020		00010000	01 010100	
17	30000		00010001	10 000000	
18	0		00010010	00000000	← Lücke
19	13		00010011	00001101	← Daten
20	17		00010100	00010001	
21	0		00010101	00000000	
22	0		00010110	00000000	

Das kleine Hilfsprogramm **bma2bin** (bma to bin) hat obige Ausgabe (in der Datei **multipli.bin**) erzeugt, die sich zur direkten Eingabe in den BONSAI-Lehrcomputer eignet.

Syntax: **bma2bin multipli** ,dh. der Name der Datei mit dem BONSAI-Maschinenprogramm muss ohne die Erweiterung .bma angegeben werden. Bei Unklarheiten bitte den Quelltext inspizieren.

Der HLT-Befehl (50000) wird durch einen Sprung auf sich selbst ersetzt, siehe oben an der mit einem '*' markierten Stelle.

Die ersten Schritte mit dem Lehrcomputer

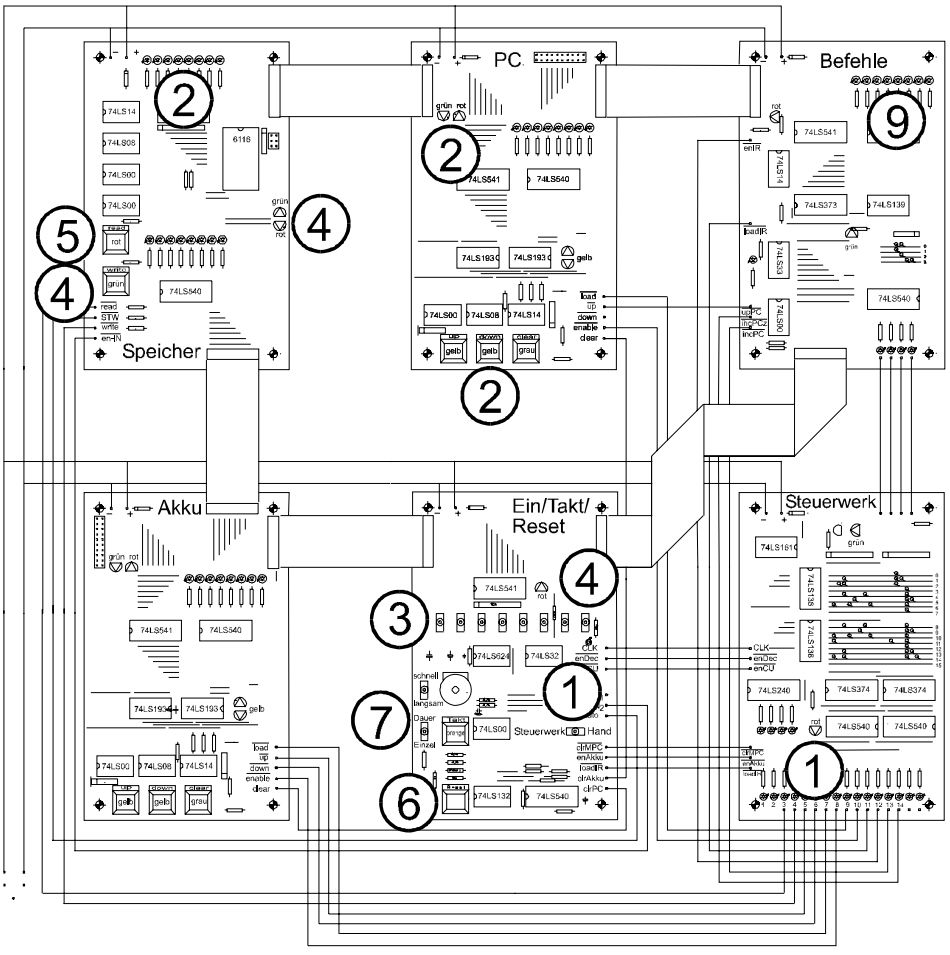
Arbeitsauftrag 6

Um den Umgang mit dem Rechner zu illustrieren, soll das kleine Additionsprogramm

				Adresse	Datum
0	JMP	3	Befehle	0000 0000	10 000011
1	INC	7		0000 0001	00 000111
2	DEC	8		0000 0010	01 001000
3	TST	8		0000 0011	11 001000
4	JMP	1		0000 0100	10 000001
5	JMP	5	*	0000 0101	10 000101
6		0	Lücke	0000 0110	00000000
7		3	Daten	0000 0111	0000 0011
8		2		0000 1000	0000 0010

in den Speicher gebracht und ausgeführt werden.

Jeden von-Neumann-Rechner kann man in Operationswerk und Steuerwerk aufteilen. Das Operationswerk ermöglicht vielerlei 'Operationen' wie Registertransfers, Rechnungen u.s.w., das Steuerwerk steuert diese Operationen. Mit dem Lehrcomputer ist es möglich, die Rolle des Steuerwerks selbst zu übernehmen und das eingebaute Steuerwerk abzuschalten. Genau das müssen Sie bei Ein-und Ausgaben auch tun.
Achten Sie bei den folgenden Ausführungen auf die Zahlen ($\in, \notin$,...), sie ermöglichen Ihnen mit Hilfe der Übersichtszeichnung eine einfachere Orientierung:



∈ Stellen Sie den Schalter zur Steuerwerkskontrolle auf 'Hand' (Knebel nach rechts), die rote Leuchtdiode, die das Schreiben des Steuerwerks auf den Steuerbus anzeigt, erlischt.

∉ Wählen Sie auf der PC-Platine die richtige Adresse. Benutzen Sie dazu die gelben Tasten 'up' und 'down' bzw. die graue 'clear'-Taste zum 'Auf-Null-Setzen'. Beachten Sie die Anzeige des Adressbusses auf der Speicherplatine, sie folgt der Anzeige des PC. Die rotleuchtende Diode auf der PC-Platine zeigt, dass der PC auf den Datenbus schreibt.

∠ Stellen Sie mit den acht Kippschaltern der Eingabe das richtige Datum ein ('1' ≙ Knebel nach oben).

∇ Drücken Sie die grüne 'write'-Taste auf der Speicherplatine, um das eingestellte Datum in den Speicher zu schreiben Wenn Sie die Taste etwas gedrückt halten, können Sie am Aufleuchten der Leuchtdioden den Weg der Eingabedaten verfolgen. Der Eingabemodul schreibt auf den Datenbus (rote Leuchtdiode), der Speicher liest vom Datenbus (grüne Leuchtdiode).Sie können auch während des Schreibens ('write'-Taste gedrückt) das Datum ändern (die Eingabeschalter betätigen).

® Mit Hilfe der roten 'read'-Taste können Sie das in den Speicher geschriebene Datum wieder auslesen. Da Sie während des Lesens die Adresse ändern dürfen, ist es ein leichtes, z.B. zu Kontrollzwecken den Speicher durchzublättern.

© Haben Sie durch mehrmaliges Wiederholen der Schritte ∉ bis ∇ das Programm und seine Daten in den Speicher gebracht, können Sie das Programm starten. Durch Drücken der blauen Reset-Taste auf der Platine EIN/TAKT/RESET wird ein 'Hardware-Reset' durchgeführt, der im einzelnen folgendes bewirkt:

- der PC wird auf Null gesetzt
- der Mikroprogrammzähler (MPC) im Steuerwerk wird auf Null gesetzt.
- der AKKU wird auf Null gesetzt
- das Befehlsregister (instruction register, IR) wird mit Nullen gefüllt
- das Steuerwortregister im Steuerwerk wird mit lauter '1'en (alle Signale auf inaktiv) gefüllt

™ Überprüfen Sie die Stellung der beiden Kippschalter am Taktgeber:
Mit dem oberen Schalter können Sie zwischen dem hohen und dem niederen Frequenzbereich umschalten. Es ist an sich gleich wie dieser Schalter steht, aber wenn Sie von der Ausführung des Programms etwas sehen wollen, stellen Sie ihn auf 'langsam' (Knebel nach unten). Der untere Kippschalter schaltet zwischen Einzel- und Dauertakt um, stellen Sie ihn zunächst auf Einzeltakt (Knebel nach unten). Sollte der Taktgeber gelaufen sein (die rote Leuchtdiode 'Takt' auf der gleichen Platine war in Betrieb), so stellen Sie durch einen neuerlichen RESET sicher, dass der Mikroprogrammzähler auf Null steht (Die vier grünen Leuchtdioden auf der Steuerwerksplatine müssen *aus* sein.).

∏ Übergeben Sie nun dem Steuerwerk durch Betätigen des Schalters **Hand/ Steuerwerk** (∈) die Kontrolle. Das Steuerwerk schreibt auf den Steuerbus, wenn die dreieckige rote Leuchtdiode (∈) aufleuchtet. Jetzt starten Sie das Programm, indem Sie entweder den Dauertakt einschalten oder einzelne Taktimpulse durch Betätigen der 'Takt'-Taste (©) auslösen.

√ Wie bereits bei der binären Kodierung erwähnt fehlt ein HLT-Befehl. Der Rechner wird durch eine Endlosschleife abgefangen, deren Auftreten an einem stabilen Bitmuster (im Beispielprogramm: 10 000101) im Befehlsregister zu erkennen ist.

· Auslesen der Ergebnisse:
Wenn nicht schon geschehen, so wird der Rechner durch Schalten auf Einzeltakt angehalten. Jetzt wird das Steuerwerk abgeschaltet und am PC die Adresse der interessierenden Speicherzelle (hier: z.B. 7_{10} = 0000 0111_2) eingestellt. Drücken der

READ-Taste auf der Speicherplatine läßt nun das Datum (hier: z.B. 5_{10} = 0000 0101_2) auf dem Datenbus erscheinen.

Aspekte eines Betriebssystems

Abstraktion des Maschinenbegriffes (nach Coy):

- Reale Maschine = Zentraleinheit + Geräte (Hardware)
- Abstrakte Maschine = Reale Maschine + Betriebssystem
- Benutzermaschine = Abstrakte Maschine + Anwendungsprogramm

In wikipedia heißt es

> *"Ein Betriebssystem ist die Software, die die Verwendung (den Betrieb) eines Computers ermöglicht. Es verwaltet Betriebsmittel wie Speicher, Ein- und Ausgabegeräte und steuert die Ausführung von Programmen."*

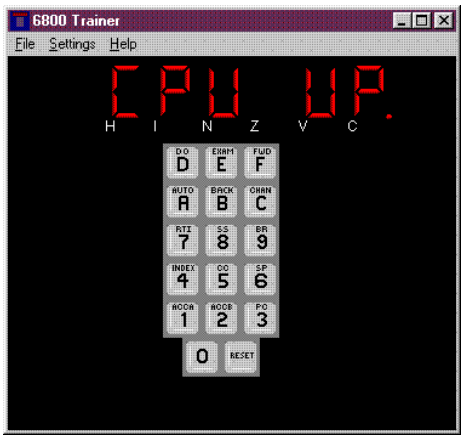
Beim Bonsai gibt es keine Ein- und Ausgabegeräte, da ist also nichts zu verwalten. Natürlich will man Programme eingeben, starten und ihre Ergebnisse auslesen können. Das wird alles durch den Handbetrieb möglich.

Ein realer Prozessor lässt sich nicht auf Handbetrieb umschalten. Hier muss eine Software-Lösung gefunden werden. Um das Grundprinzip zu verstehen, reicht ein einfacher Ein-Platinen-Computer, wie z.B. der Heathkit ET-3400 aus. Mini-Betriebssysteme für sehr kleine Computer nennt man Monitor.

In wikipedia heißt es

> *"Ein Monitor kann auch ein primitives Betriebssystem- oder Debuggerprogramm sein, typischerweise sehr klein und mit sehr kurzen Kommandozeilen-Kommandos. Solche Monitore wurden/werden typischerweise als komplette Betriebssysteme für Kleinstcomputer, Debugger oder als BIOS-Äquivalent auf Workstations benutzt."*

Der Monitor des Heathkit ET-3400



Der mc6800-Einplatinencomputer ET-3400 hat eine Eingabe-Tastatur mit 17 Tasten und eine Ausgabe mit 6 7-Segment-Anzeigen. Weitere Peripherie lässt sich anschließen. Mit der Tastatur lässt sich ein Reset auslösen und alle 16 Hex-Ziffern eingeben. Mit Ausnahme der Null sind alle Hex-Ziffern-Tasten mit einer zusätzlichen Funktion belegt.

Bei 'Daves Old Computers' (www.classiccmp.org/dunfield/heath/index.htm) findet man die Handbücher und - besonders reizvoll - ein Simulationsprogramm.

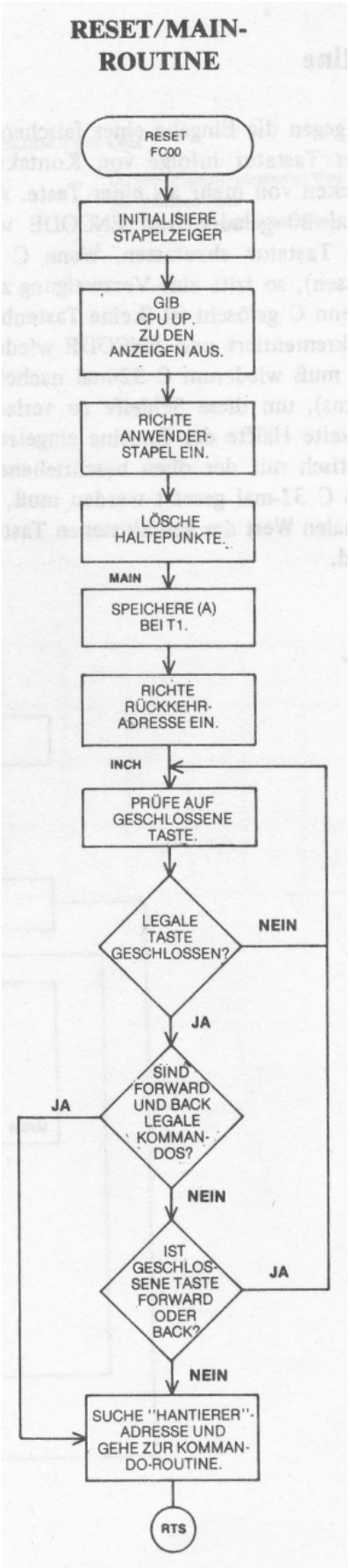
Betriebssystemroutinen, die über die Tastatur aufrufbar sind

- reset führt CPU-Reset durch und springt in die Monitor-Hauptschleife
- do startet nach Eingabe von 4 Hex-Ziffern ein Programm mit der angegebenen PC-Belegung.
- exam zeigt den Speicherinhalt an der angegebenen Stelle
- fwd schaltet eine Speicherzelle weiter, back eine Speicherzelle zurück
- chan ermöglicht ein Ändern der Speicherzelle
- auto ermöglicht ein schnelles Beschreiben des Speichers, weil nach Eingabe eines Bytes sofort eine Speicherzelle weitergerückt wird.
- acca zeigt den Inhalt des Akkumulators a, accb entsprechend, mit chan ändern
- pc zeigt den Inhalt des PCs an, mit chan ändern
- index zeigt den Inhalt des Index-Registers an, mit chan ändern

- cc zeigt den Inhalt des Condition-Codes-Registers an, mit chan ändern
- sp zeigt den Inhalt des Stack-Pointer-Registers an
- rti Resume User's Program
- ss Einzelschritt-Betrieb
- br setzt Breakpoint

Weitere Monitor-Routinen

- addr - accept address value with 'ad' prompt
- outsta - output string for address prompt
- redis - reset displays
- baddr - build address
- bkpt - break point return
- mem - display address and data
- replace - replace displayed value
- prompt - prompt and input bytes
- register display functions
- display - display indexed bytes
- encode - scan and encode keyboard
- inch - input character from keyboard
- ihb - input hex byte und display on leds
- outbyt - output two hex digits
- outhex - output hex digit
- outch - output character to display
- outstr - output imbedded charakter string
- swive1 - set up breakpoint return and jump to user code



Das komplette Monitor-Programm (ca. 1kB)

```
8E00EBBDFD8D4E673E003EE7CE00CBDFF286FFC608365A26FC97EE86193686FC
36BDFDF47D00EE2708810F27F4810B27F0DFECCEFFB408084A2AFBA60136A600
36DEEC96EE39CE00E286FFC6040808A1002604A101270E5A26F3BDFD8D00473E
0E0EA04C39DFEE8D1D1F858D084C39DFEE8D133D9D8D45DEEEC6027EFD25DFEE
8D0477BD20EFCEC12F7EFE50DEF20808080808088DD98D244FC606BDFE3A5A26
FA8D19BDFE6BC60430EE08A6003636863FA7005A26F2CEFCCE7EFEFCDFECCEC1
6FDFF0DEEC39CE00EE8DB3DEEE39309FF2A60626026A054AA706E605D7EC97ED
0C8E00D9C604323230EE089CEC26010DA7005A26F124ACDEEC8DC1DFEECE00EE
C6028D03EE005A7EFD7B8DBA8DEB8D0B0820F98DB10908080920DE5D2706368D
228D02323937860858BDFE3A5A26FA338D1137BDFE09A700085A26F73317094A
26FC393796F18B205A26FB97F133398D3B309520168D35770D0DFD20108D2D77
0D0D9F20098D25678D4C4C5C4C4C5C8B02DEF2084A26FC8D024C3937A600BDFE
20085A26F73317094A26FC395FCEC16F7EFE50BDFCBCDEF2C6204FE50127014C
BDFE285626F44C398DE25BE7D6F3CB0799F28D6C175F8D6886013937F6C003B6
C0064848485948594859485937F6C005C41F1B334353DFECCEFFA511271124063617
33CEFFAD5D2606084822FC27010CA600DEEC333937C6208DC225FA5A26F9C620
8DB924FA5A26F933398DE98D1B4848484837168DDF8D111B33368D9F25FC3239
36444444448D013236840FDFECCEFF95084A2AFCA6008D043239DFECDEF03749
49C61049A700095A26F9DFF0DEEC3339DFF030EE003131A6008DDF084D2AF84F
6E008D07DEF2EE067EFCF99FEEDEF2A60736A60636EE06863F3636A60236A601
36A6003616CEFF7508C00824FBA600465C26FC3236251E813024048120241481
602511818D270C84BD818C270484308130C2FF5C5C2770302502E7018601C102
2E062702A701A7024FEB06A905A705E706DEF2A706E707C606323684CF818D32
2748816E275B817E275E81392762813B276C813F276EAF0636CEFF05867E97F4
DFF59EF23B30EE05084F5F9CEE260C09EE0009E6002A014330EE05EB01A90030
A705E70609DFF29EEE39818D2602865F803F360909DFF2A603A701085A2AF820
90334FEB05A9048C3233A706E70720D50808DFF2A603A705095A2EF820C9085A
2AFC20C1A607A700095A2AF88A10A701C6FA860020D49C003CAF400000AC6412
64126410641011011004100010001l0D100C100C100C7E306D79335B5F707F7B
```

```
771F4E3D4F47070A0D0205080B0E0306090C0F000104FC45FD55FD5DFD65FD4F
FD93FDA8FC96FE62FC46FD0AFD18FD1BFC8CFD13FD16FFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00F700F400FDFC00
```

Auszug aus dem Lehrplanentwurf zum LK Informatik:

Lerninhalte – Technische Grundlagen

Die Wirkungsweise eines Computers erschließt sich als das Zusammenspiel verschiedener Ebenen. Eine Hochsprache wird in eine Maschinensprache übersetzt. Dies wird typischerweise von einem Compiler erledigt. Das Maschinensprachenprogramm bedient sich vorhandener Betriebssystemsroutinen. Maschinenbefehle werden von einem Prozessor ausgeführt. Dazu verfügt der Prozessor über passende Komponenten. Das Zusammenspiel dieser Komponenten wird von einem Steuerwerk kontrolliert.
Die Verwendung eines Simulationsprogramms erscheint unerlässlich.

Ziele / Inhalte	**Bemerkungen**
Elemente eines einfachen Modell-Assemblers kennen und anwenden	Es genügt eine sehr einfache 1-Adress-Sprache.
Einfache Algorithmen aus einer Hochsprache über die Assemblersprache in eine Maschinensprache übersetzen	Hierbei ist eine systematische Übersetzung elementarer Kontrollstrukturen vorzunehmen. Die Wahl einer Zwischensprache mit Sprungbefehlen ist vorteilhaft.
Wirkungsweise eines Compilers verstehen	Zu behandeln sind die Definition der verwendeten Quell- und Zielsprache (z.B. durch Syntaxdiagramme) und die Funktionsweise von Scanner, Parser, Codeerzeuger
Komponenten eines von-Neumann-Rechners kennen und ihr Zusammenwirken verstehen	Besprochen werden müssen: • die Komponenten eines Operationswerks (Speicher, Programmzähler, Befehlsregister, ALU, Adressbus, Datenbus, Steuerleitungen, Tore) • der Fundamental- / Befehlszyklus (Befehl holen, dekodieren, ausführen, Rücksprung) • die Organisation eines Steuerwerks z.B. durch Mikroprogrammierung Ein fundiertes Verständnis lässt sich durch den Einsatz eines geeigneten Simulationsprogramms/Modellrechners erreichen.
Parameter für die Arbeitsgeschwindigkeit eines Computers beurteilen	Taktfrequenz und Taktanzahl zur Ausführung eines Befehls
Das gewählte Rechnerkonzept bewerten und Erweiterungen bzw. Verbesserungen angeben	Unterprogrammtechnik, RISC, CISC, Parallelisierung...
Grundlegende Funktionen eines Betriebssystems angeben	z.B. BIOS, Dateisystem, Prozesse

Beispiel bzw. Vorschlag für einen Unterrichtsverlauf in Leistungs- und Grundkurs:

Thema	Std. im Lk (Beispiel)	Std. im Gk (Vorschlag)
Bonsai-Assembler	1	1
Übersetzungsschablonen	3	2
systematische Programmentwicklung	3	1
Compiler	1	1
Maschinensprache	1	1
Architektur	2	2
Mikroprogrammierung	3	1
Binäre Kodierung	1	1
Arbeit mit Hardware	3	1
Betriebssystem	2	1
	20	12

Das Thema erstreckt sich bei Einhaltung der Stundenansätze auf genau vier Wochen sowohl im Grund - wie im Leistungskurs.

Name _____

1. Verwende die Marken B (Bedingung), S (Schleifenkörper) und E (Ende) und stelle folgende repeat-Schleife mit Hilfe von 'if-then-else'- und 'goto'-Anweisungen dar.

```
repeat
   Dec(r1); Inc(r2);
until r1 = 0;
```

2. a. Übersetze folgendes Programm von Hand zunächst noch unter Verwendung der Marken in BONSAI-Assembler.

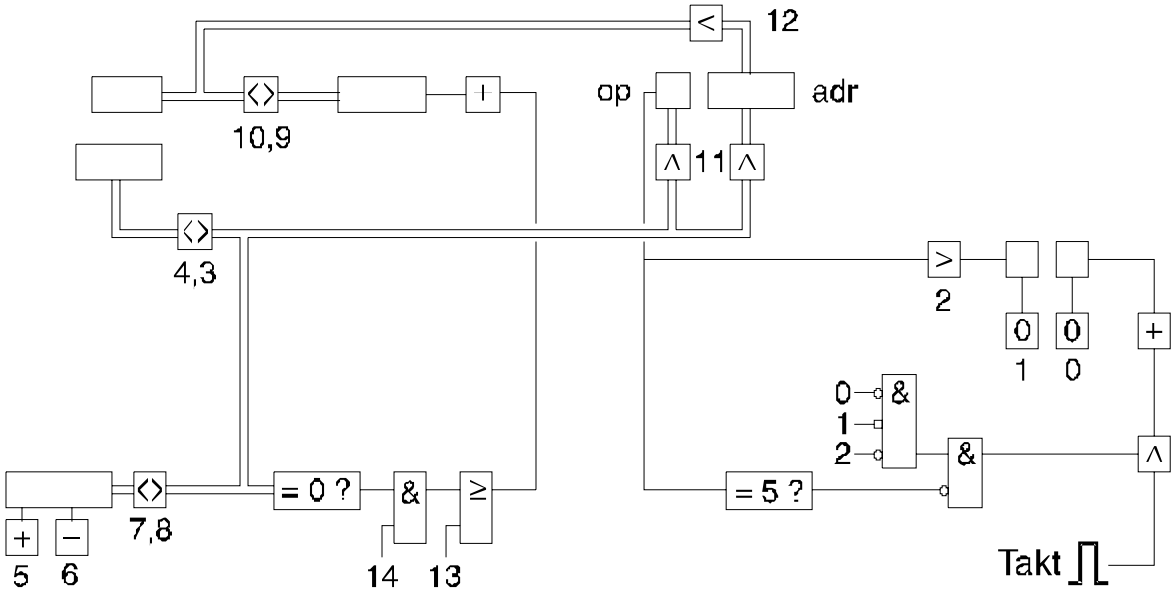
```
program sumn;
label
  B1,S1,E1,B2,S2,E2,B3,S3,E3;
var
  r1,r2,r3 : integer;
begin
  writeLn; write('r1 = '); readLn(r1); r2 := 0; r3 := 0;
  writeLn;

B1:  if r1 = 0 then goto E1 else goto S1;
S1:  S2:  Dec(r1); Inc(r2); Inc(r3);
     B2:  if r1 = 0 then goto E2 else goto S2;
     E2:
     S3:  Dec(r3); Inc(r1);
     B3:  if r3 = 0 then goto E3 else goto S3;
     E3:
     Dec(r1);
     goto B1;
E1:

  write('r2 = ',r2);
  readLn
end.
```

   b. Ersetze die Marken durch Nummern von Speicherzellen.

3. Gib ein Dokument an, das ein RePas-Scanner ohne Fehler verarbeitet, ein RePas-Parser jedoch als fehlerhaft erkennt. Erläutere am Beispiel den Unterschied zwischen lexikalischer und syntaktischer Analyse.

4. a) Einen Computer kann man grob durch Operationswerk und Steuerwerk strukturieren. Erläutere kurz die Begriffe und begründe, wo der Taktgeber einzuordnen ist.

   b) Benenne die Bestandteile des „Bonsai“–Computers (Register, Busse, usw). Beschrifte dazu die weiter unten stehende Zeichnung.



5. Folgende Zeile gibt die Mikrobefehle in den einzelnen Takten für den tst-Befehl an. Beschrifte die Folge mit den einzelnen Phasen des „Fundamentalzyklus“ ?

   10 | 3 , 10 , 11 | 0 , 2 | 12 | 3 , 12 | 3 , 12 , 14 | 13 | 0 , 1

---

6. Folgendes Programm verwendet einen neuen Bonsai-Assembler-Befehl 7, Inc (adr) „inkrementiere den Inhalt der Speicherzelle, deren Adresse in der Speicherzelle adr steht, und erhöhe den PC“. Es wird mit PC = 0 gestartet.

	Maschinensprache	Assembler
0		jmp 7
1	1	
2	0	
3	1	
4	0	
5	3	
6	5	
7		tst 6
8		jmp 10
9		hlt
10		inc (6)
11		dec 6
12		jmp 7

   a) Wie nennt man die Art der Adressierung in Befehl 7?

   b) Gib den Zustand der Daten nach dem Programmlauf an.
   Streiche dazu die Anfangswerte der Daten in der Tabelle durch und schreibe die Endwerte daneben.

   c) Was verändert sich, wenn das gleiche Programm mit einer 2 in Speicherzelle 6 gestartet wird?

d) Ergänze die zweite Spalte der Tabelle zu einem dezimal kodierten Maschinenprogramm.

e) Gib die Folge der Mikrobefehle für den inc (adr)-Befehl an.

f)* Implementiere den Befehl 7 im Mikroprogramm "mikropro.bmi". Speichere das neue Mikroprogramm unter mikro7.bmi im Ordner <familienname>. Teste das Mikroprogramm mit Hilfe des Maschinenprogramms aus Aufgabe 6. Speichere das Maschinenprogramm unter dem Namen inc7.bma im Ordner <familienname>.

Aufgabe f) nur bei genügender Zeit bearbeiten!

praktische Aufgabe:

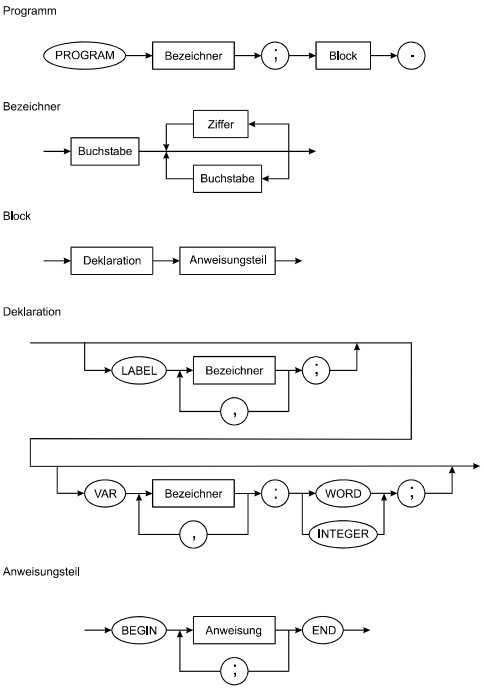
7. In dem Delphi-Projekt "Subtraktion" (im Ordner i:\Lk12\subtraktion, Ordner nach h: kopieren) wird systematisch ein Pascal-Programm in ein RePas-Programm (Syntax-Diagramme liegen bei) übersetzt. Diese Übersetzung ist schon weit fortgeschritten, aber noch nicht fertig.

   a) Bringe die Übersetzung zu Ende.
   Kopiere deinen Projekt-Ordner "subtraktion"am Ende in i:\ka12_031010\<familienname>.

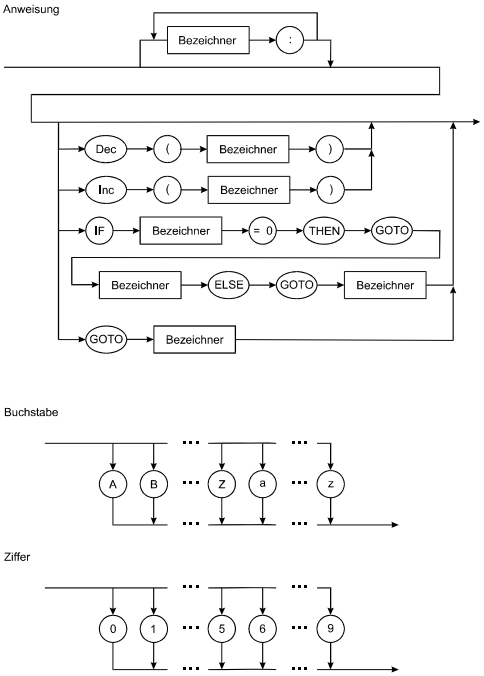
   b) Erzeuge aus deinem Projekt in der Datei "subtrakt.pas" ein korrektes RePas-Programm als Eingabe für den Compiler. Lege die Datei im Ordner <familienname> an.

   c) Erzeuge aus "subtrakt.pas" mit Hilfe des Compilers (Ordner "compiler" aus i:\lk12 nach h: kopieren) ein lauffähiges Bonsai-Programm "subtrakt.bon". Kopiere "subtrakt.bon" ebenfalls in den Ordner <familienname>.

RePas



RePas



Viel Erfolg!