Praktikum Mikrorechner 1 (Einführung)

G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

November 5, 2014

Einführung

1.1 Wie funktioniert ein Prozessor?

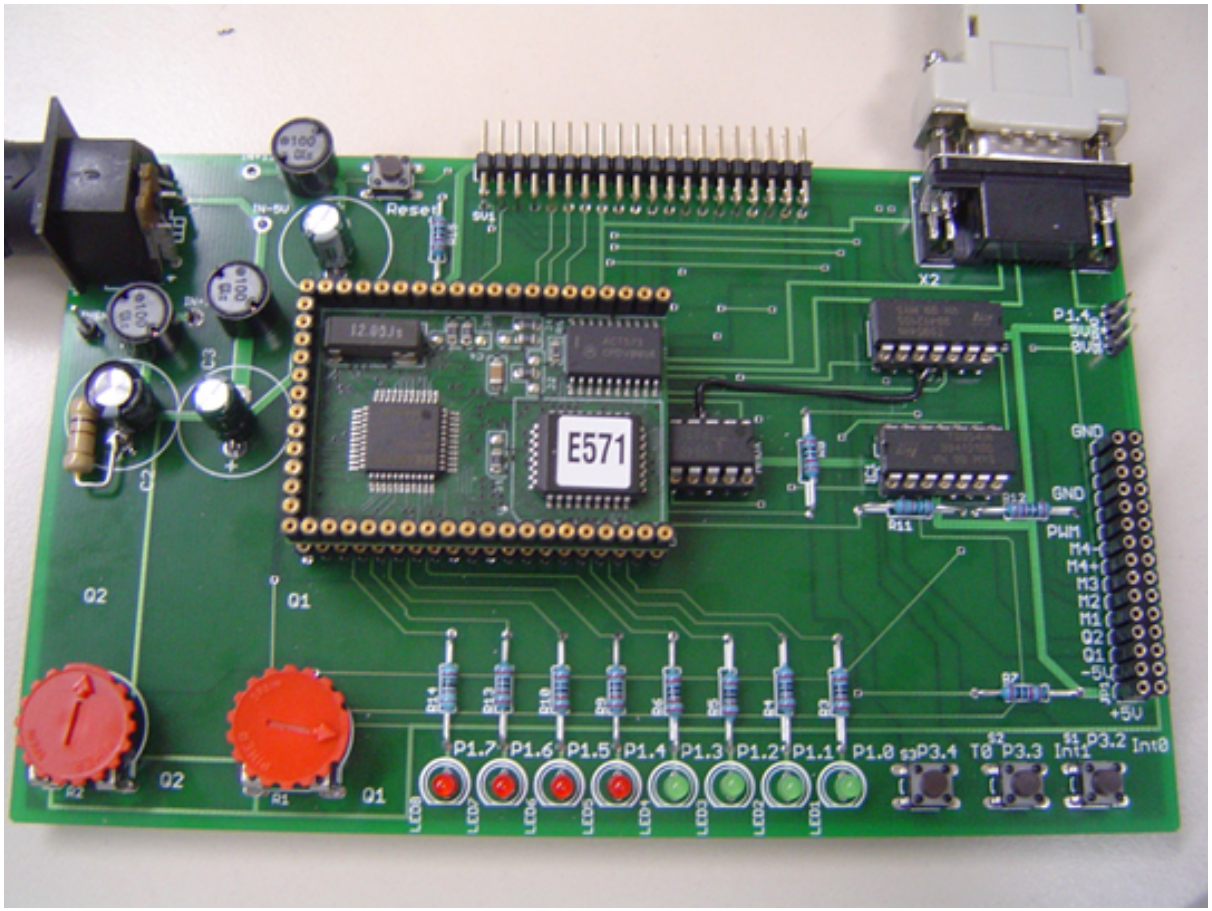
1.2 Entwicklungs- und Testumgebung

Einführung

Die überwiegende Mehrheit der heute eingesetzten Prozessoren sind kleine preiswerte 8- und 16-Bit-Mikrocontroller. Der beispielhaft benutzte SAB-C504 von Siemens ist ein 1-Chip-Rechner mit:

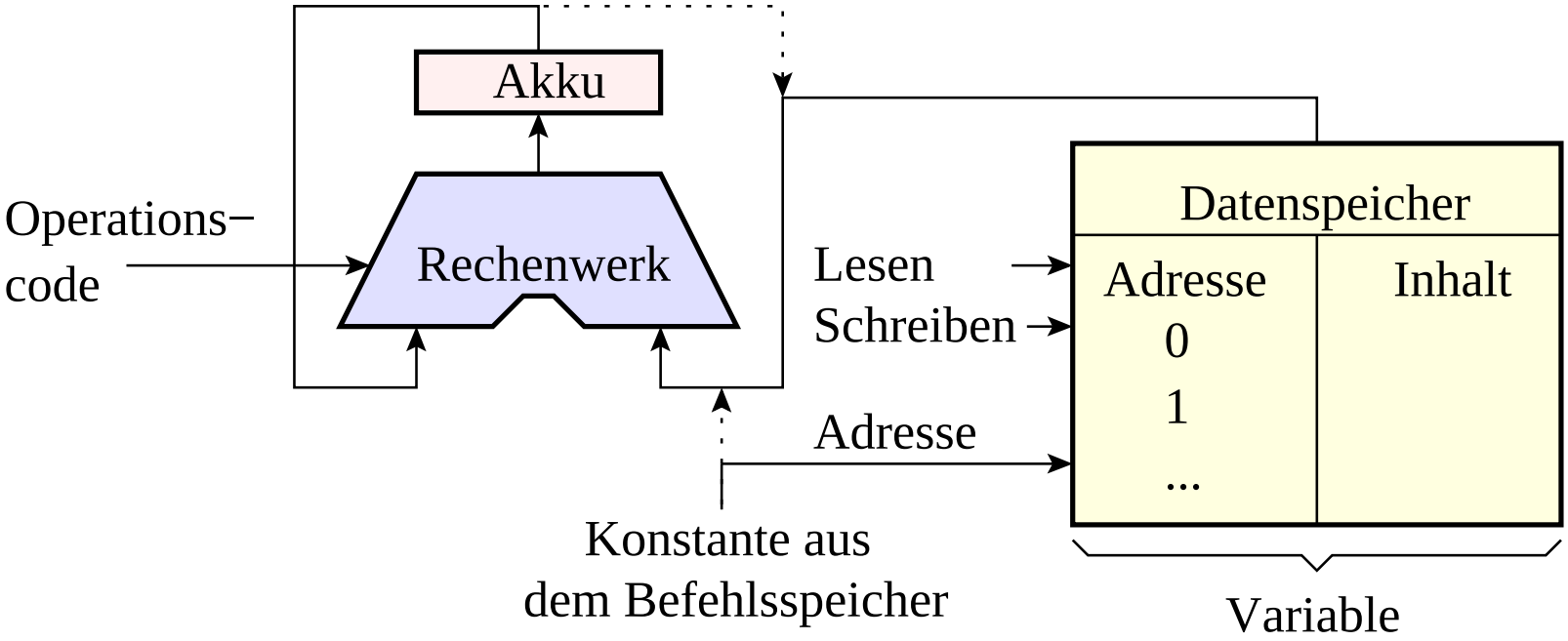
- 8051 kompatiblem Prozessor
- Befehls- und Datenspeicher
- paraller und serieller Ein- und Ausgabe
- Timern
- Analog-Digital-Wandler etc..

Die Versuchsbaugruppe



Wie funktioniert ein Prozessor?

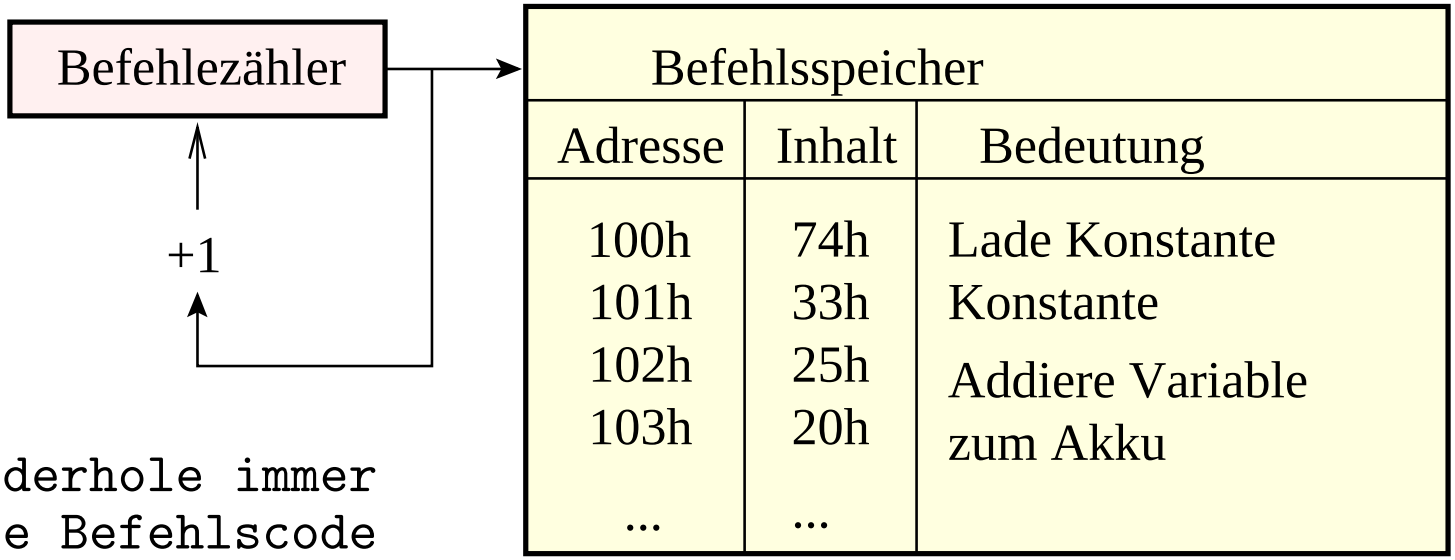
Datenfluss



`Grundalgorithmus für die Verarbeitung von Daten`

- `Lade Wert einer Variablen oder Konstante in den Akku`
- `Akku = Op(Akku, Variable/Konstante)`
- `Kopieren des Ergebnisses aus dem Akku in den Speicher`

Kontrollfluss




```
wiederhole immer
hole Befehlscode
PC = PC + 1
Fallunterscheidung
  wenn "Lade Konstante"
    Akku = Befehlsspeicher(PC)
    PC = PC + 1
  wenn ...
```

Befehlssatz

- Transportbefehle

```
Semantik: Ziel = Quelle
Syntax:   OpCode Ziel, Quelle
Ziel/Quelle: Akku, Register, Speicherplatz
```

- Verarbeitungsbefehle

```
Semantik: Akku = Akku Op Operand2
Syntax:   OpCode A, Operand2
Op: Addition, Subtraktion, ...
```

- Steuerbefehle

```
PC = PS(PC, PC+1) [wenn Bedingung]
PC = PC+PS(PC) [wenn Bedingung]
...
```

(absoluter/relativer/berechneter, bedingter/unbedingter Sprung mit/ohne Speichern der aktuellen Adresse)

Entwicklungs- und Testumgebung

Erstes Programm

- Zielfunktion

```
a, b, c:    16-Bit-
            Variablen
a = b + c
```

- Adresskonstanten

```
Start equ 100h
Monitor equ 0e300h
a0 equ 60h
a1 equ 61h
b0 equ 62h
b1 equ 63h
c0 equ 64h
c1 equ 65h
```

- Programm

```
org Start     ;Organisation
mov a, b0     ;Kopie DS(b0)=>Acc
add a, c0     ;Acc=Acc+DS(c0)
mov a0, a     ;Kopie Acc=>DS(a0)
mov a, b1
addc a, c1
mov a1, a
ljmp Monitor
end           ;Organisation
```

Erstellen und Übersetzen

- Editieren: gedit <dateiname.asm>&
- Übersetzen: asem <dateiname>.asm
- Übersetzungergebnis in lesbarer Form: less <dateiname>.lst

```
8:   ..
9:    N       0100      org 100h
10:  0100  E5 62              mov a, b0
11:  0102  25 64              add a, c0
12:  0104  F5 60              mov a0, a
13:  0106  E5 63              mov a, b1
14:  0108  35 65              addc a, c1
15:  010A  F5 61              mov a1, a
16:  010C  02 E3 00           ljmp Monitor
17:                     end ...
```

Laden des Programms

- Rechner und Baugruppe mit seriellem Kabel verbinden
- Terminalprogramm (kermit) starten und konfigurieren:

```
...$ kermit
... C-Kermit> set line /dev/ttyS0
... C-Kermit> set baud 9600
... C-Kermit> set carrier-watch off
... C-Kermit> set flow-control none
... C-Kermit> c
```

Danach muss eine Meldung der Form erscheinen:

```
Connecting to /dev/ttyS0, speed 9600
```

- Mikrorechner initialisieren und Konsolenprogramm wecken:
  - auf der Baugruppe Reset-Taste drücken
    => in Kermit erscheint eine Meldung
  - mit der Maus die Konsole von Kerit auswählen und CTRL-C eingeben
    => Statusausgabe des Mikrorechners
- Download

```
upload31 <dateiname>.hex
```

- Tastatureingaben in Kermit zur Kontrolle, dass richtig geladen

```
CTRL-C
=> Statusausgabe des Mikrorechners
cb 100 to 10E
=> Anzeige des Bytecodes: E5622564F560E5...
```

Kommunikation

go [from 100] [till 10C]	Programm ausführen [von Adresse] [bis Adresse]
step [from 100]	Einzelbefehl abarbeiten [ab Adresse]
db <adresse>=<Wert>	Wertzuweisung an eine Variable im Speicher
db <adresse> to <adresse>	Lesen eines Datenspeicherbereichs
cb <adresse> to <adresse>	Lesen eines Befehlsspeicherbereichs

Siehe Web-Seite zur Lehrveranstaltung, Datei:

»Handout Toolchain«

Test

```
db 60=0                    Eingabe des Wertes für a0
   ...                              ...
db 65=72                   Eingabe des Wertes für c1
db 60 to 65          ⇒     Speicherzustand:  00003f154372
```

```
cb 100 to 102        ⇒ 1.  Befehl:           e562 (mov a, b0)
step from 100        ⇒ Befehlszähler:        0102
                       Folgebefehl:          2564 (add a, c0)
                       Akku:                 3f
step                 ⇒ Befehlszähler:        0104
                       Folgebefehl:          2564 (mov a0, a)
                       Akku:                 82
step                   Übertrag (psw.7):     psf>7f, d.h.  cy=1
                       Befehlszähler:        0106
                       Folgebefehl:          e563 (mov a, b1)
db 60 to 65          ⇒ Speicherzustand:      82003f154372
```

Aufgaben

1. Test zu Ende führen.
2. Variablenanordnung so ändern, dass mit db das höchstwertige Byte einer Zahl links und das niederwertigste Byte rechts angezeigt wird.
3. Test mit anderen Beispielzahlen.
4. Änderung der Addition in eine Subtraktion durch folgende Ersetzungen:

add a, c0	clr c subb a, c0	Übertrag löschen acc = acc - c0 - cy
addc a, c1	subb a, c1	acc = acc - c1 - cy

und Test der Subtraktion mit einem Zahlenbeispiel.