

Technische Universität Ilmenau  
Fakultät IA  
Fachgebiet Rechnerarchitektur

Praktikum Rechnerarchitektur 1  
WS 2021/22

# Versuchsprotokoll

Versuche zur x86-Assemblerprogrammierung

16.11.2021

## A1: Lauflicht

Versuchsaufbau anhand einfacher Assemblerprogramme. Benutze parallele digitale Ein- und Ausgabebaugruppen, die mit den LED-Reihen, der Tasten- und der Schalterreihe sowie den Sieben-Segment-Anzeigen verbunden sind.

### Grundaufgabe a: Funktionen ermitteln

```

1  ; Programmbereich:
2  anf:  MOV   EDX,400000H ;Groesse der Verzoeigerung
3        MOV   [verzoe],EDX ;Verzoeigerung speichern
4
5  m1:   MOV   EDI,10      ;EDI=10
6        MOV   ESI,OFFSET ziff ;Adresse von ziff in ESI
7
8  m2:   MOV   AL,[ESI+EDI-1] ;AL=ziff+9
9        OUT   0B0H,AL     ;SiebenSegment schreibt AL
10       CALL  zeit        ;warten
11       DEC   EDI         ;EDI=EDI-1
12       JNZ   m2          ;if(EDI!=0) goto m2
13
14       MOV   AL,0FFH     ;AL=255 (dec)
15  m3:   OUT   5CH,AL     ;LED Reihe links schreiben
16       NOT   AL          ;AL negieren
17       OUT   5DH,AL     ;LED Reihe rechts schreiben
18       CALL  zeit        ;warten
19       MOV   BL,AL       ;Inhalt von AL wird noch gebraucht
20       IN   AL,59H       ;Tastenreihe rechts lesen auf AL
21       BT   EAX,7        ;Bit 7 von EAX in Carry Flag
22       MOV   AL,BL       ;AL bekommt alten Wert zurueck
23       JC   m1           ;if(m1==0) goto m1
24       JMP   m3          ;goto m3 (Loop)
25
26 ;zeit ist ein Unterprogramm, welches nur Zeit verbrauchen soll:
27 zeit:  MOV   ECX,[verzoe] ;Lade wartezeit
28  z1:   DEC   ECX         ;ECX=ECX-1
29       JNZ   z1          ;if(ECX!=0) goto z1
30       RET              ;zurueck zum Hauptprogramm
31
32 ; Datenbereich:
33 verzoe DD   ?           ;Eine Speicherzelle (Doppelwort)
34 ziff   DB   3FH,03H,6DH,67H,53H,76H,7EH,23H,7FH,77H
35

```

**anf** setzt die Länge der Wartezeit

**m1** Lädt Register

**m2** Zählt auf Sieben Segment Anzeige

**m3** schreibt auf LED Reihe links und invertierend rechts

**zeit** Verbraucht Zeit nach "verzoe"

## Grundaufgabe b

### einfaches Lauflicht

auf der rechten LED-Reihe soll ein sichtbarer Lichtpunkt von links nach rechts laufen und immer wieder von links beginnen

```

1 anf:    MOV     EDX,400000H
2        MOV     [verzoe],EDX
3
4        MOV     AL, 80H      ;Startwert fuer LED Reihe
5 lauf:   OUT     5CH, AL     ;Wert auf LED Reihe schreiben
6        CALL    zeit       ;warten
7        ROR     AL, 1       ;Bits um 1 nach rechts
8        JMP     lauf       ;Schleife wiederholen
9
10 zeit:  MOV     ECX,[verzoe]
11 z1:    DEC     ECX
12        JNZ    z1
13        RET

```

### Lauflicht mit Geschwindigkeitsumschalter

das Lauflicht soll durch den linken Schalter zwischen "schnell"(Schalter oben) und "langsam"(Schalter unten) umschalten

```

1 anf:    MOV     AL, 80H
2
3 lauf:   MOV     EDX, 400000H ; Wert fuer "langsam"
4        MOV     [verzoe], EDX ;"langsam" in Speicher
5        OUT     5CH, AL     ;LED Reihe schreiben
6        MOV     BL, AL     ;AL speichern
7        IN      AL, 58H     ;Schalter einlesen
8        BT     AL, 7       ;7. Bit von AL in Carry Flag
9        JNC    langsam    ;Carry Flag = 0, schalter unten
10       MOV     EDX, 200000H ; Wert fuer "schnell"
11       MOV     [verzoe], EDX ;"schnell" in Speicher
12       CMC     ;Carry Flag umschalten (0)
13
14 langsam: CALL    zeit     ;warten
15        MOV     AL, BL     ;AL aus speicher zurueck
16        ROR     AL,1       ;Bits um 1 nach rechts
17        JMP     anf       ;Schleife wiederholen
18
19 zeit:  MOV     ECX,[verzoe]
20 z1:    DEC     ECX
21        JNZ    z1
22        RET

```

### Lauflicht verändert Richtung

zusätzlich zum oben implementierten soll die Bewegungsrichtung des Lichtpunktes durch den rechten Schalter der Schalterreihe zwischen "nach links" und "nach rechts" wechseln.

```

1 anf:    MOV     AL, 80H
2 lauf:   MOV     EDX, 400000H ; Wert fuer "langsam"
3        MOV     [verzoe], EDX ;"langsam" in Speicher

```

```

4      OUT      5CH, AL      ;LED Reihe schreiben
5      MOV      BL, AL      ;AL speichern
6      IN       AL, 58H     ;Schalter einlesen
7      BT       AL, 7       ;7. Bit von AL in Carry Flag
8      JNC      langsam    ;Carry Flag = 0, Schalter unten
9      MOV      EDX, 200000H ; Wert fuer "schnell"
10     MOV      [verzoe], EDX ;"schnell" in Speicher
11     CMC      ;Carry Flag umschalten
12 langsam: CALL   zeit     ;warten
13     MOV      AL, BL      ;AL aus speicher zurueck
14     BT       AL, 0       ;0. Bit von AL in Carry Flag
15     JNC      rechts     ;Carry Flag = 1; Schalter oben
16     ROL      AL,1        ;Bits um 1 nach links
17     CMC      ;Carry Flag umschalten (0)
18     JMP      anf        ;Schleife wiederholen
19 rechts: ROR    AL, 1     ;Bits um 1 nach rechts
20     JMP      anf        ;Schleife wiederholen
21 zeit:  MOV    ECX,[verzoe]
22 z1:    DEC    ECX
23     JNZ     z1
24     RET

```

### Lauflicht mit Invertierung

durch drücken einer beliebigen Taste der blauen Tastenreihe wird die Anzeige invertiert, d.h. der Lichtpunkt ist dunkel etc. Invertierung nur solange die Taste gedrückt wird.

```

1 anf:    MOV    A1, 80H
2 lauf:   MOV    EDX, 400000H ; Wert fuer "langsam"
3         MOV    [verzoe], EDX ;"langsam" in Speicher
4         MOV    BL, AL      ;Kopie von AL anlegen
5         IN     AL, 59H     ;Tastenreihe einlesen
6         AND    AL, FFH     ;UND Operation mit FF
7         JZ     nopress    ;kein Schalter gedrueckt
8         NOT    BL         ;BL invertieren
9         MOV    AL, BL     ;AL ueberschreiben
10 nopress: OUT   5CH, AL    ;LED Reihe schreiben
11        IN     AL, 58H     ;Schalter einlesen
12        BT     AL, 7       ;7. Bit von AL in Carry Flag
13        JNC    langsam    ;Carry Flag = 0, Schalter unten
14        MOV    EDX, 200000H ; Wert fuer "schnell"
15        MOV    [verzoe], EDX ;"schnell" in Speicher
16        CMC    ;Carry Flag umschalten
17 langsam: CALL   zeit     ;warten
18        MOV    AL, BL      ;AL aus speicher zurueck
19        BT     AL, 0       ;0. Bit von AL in Carry Flag
20        JNC    rechts     ;Carry Flag = 1; Schalter oben
21        ROL    AL,1        ;Bits um 1 nach links
22        CMC    ;Carry Flag umschalten (0)
23        JMP    anf        ;Schleife wiederholen
24 rechts: ROR    AL, 1     ;Bits um 1 nach rechts
25        JMP    anf        ;Schleife wiederholen
26 zeit:   MOV    ECX,[verzoe]
27 z1:     DEC    ECX
28        JNZ    z1
29        RET

```

## Zusatzaufgabe

Erweiterungen des Programms nach eigenen Ideen:

- symetrische LED Reihe zur Mitte
- Sieben Segment zählt 9 Schritte mit

```

1  anf:    MOV     AL, 80H
2         MOV     EDI, 0
3         MOV     ESI, OFFSET ziff
4  lauf:  MOV     EDX, 400000H    ; Wert fuer "langsam"
5         MOV     [verzoe], EDX ; "langsam" in Speicher
6         MOV     BL, AL         ; Kopie von AL anlegen
7         IN      AL, 59H        ; Tastenreihe einlesen
8         AND     AL, FFH        ; UND Operation mit FF
9         JZ      nopress       ; kein Schalter gedrueckt
10        NOT     BL             ; BL invertieren
11        MOV     AL, BL         ; AL ueberschreiben
12  nopress: OUT    5CH,AL        ; LED Reihe links schreiben
13        NOT     AL             ; AL negieren
14        OUT    5DH,AL        ; LED Reihe rechts schreiben
15        MOV     BH,[ESI+EDI-1] ; Sieben Segment berechnen
16        OUT    OBOH,BH       ; Sieben Segment schreiben
17        DEC     EDI           ; Sieben Segment runterzaehlen
18        JZ      timer        ; Timer auf 0 setzen
19        IN      AL, 58H       ; Schalter einlesen
20        BT     AL, 7          ; 7. Bit von AL in Carry Flag
21        JNC    langsam       ; Carry Flag = 0, Schalter unten
22        MOV     EDX, 200000H  ; Wert fuer "schnell"
23        MOV     [verzoe], EDX ; "schnell" in Speicher
24        CMC                     ; Carry Flag umschalten
25  langsam: CALL   zeit         ; warten
26        MOV     AL, BL         ; AL aus speicher zurueck
27        BT     AL, 0          ; 0. Bit von AL in Carry Flag
28        JNC    rechts        ; Carry Flag = 1; Schalter oben
29        ROL    AL,1           ; Bits um 1 nach links
30        CMC                     ; Carry Flag umschalten (0)
31        JMP     anf           ; Schleife wiederholen
32  rechts: ROR     AL, 1        ; Bits um 1 nach rechts
33        JMP     anf           ; Schleife wiederholen
34  timer: MOV     BH, OFFH
35        RET
36  zeit:  MOV     ECX,[verzoe]
37  z1:    DEC     ECX
38        JNZ    z1
39        RET

```

## A2: Timerbaustein

Arbeite mit einem programmierbaren Interfacebaustein, der über eigene Register angesprochen wird. Als Beispiel dient ein Programmierbarer Intervalltimer (PIT, auch als „Zähler-Zeitgeber-Baustein“ oder „Timerbaustein“ bezeichnet) vom Typ 8254.

Frequenzen der C-Dur Tonleiter								
Ton	c'	d'	e'	f'	g'	a'	h'	c''
f(Hz)	261,6	293,7	329,6	349,2	392,0	440,0	493,9	523,2
Zählkonstante	7662	6825	6079	5730	5102	4545	4056	3824
Freq(Hex)	1DEEH	1AA9H	17BFH	1662H	13EEH	11C1	FD8H	EF0H

### Grundaufgabe a

Der Kanal 0 des Timerbausteins soll als programmierbarer Frequenzgenerator benutzt werden. Dazu wird die Betriebsart „Mode 3“ verwendet (Frequenzteiler mit symmetrischer Rechteckschwingung am Output). Die Output-Frequenz soll 440 Hz betragen. Als Input benutzen Sie den eingebauten 2-MHz-Generator.

$$\text{Zählkonstante: } \frac{2\text{MHz}}{440\text{Hz}} = 4545,4545 = (11C1)_{16}$$

```

1     MOV AL, 36H; Steuerbyte 00110110
2     OUT 57H, AL
3     MOV AL, 0C1H; LSB
4     OUT 54H, AL
5     MOV AL, 011H; MSB
6     OUT 54H, AL

```

### Grundaufgabe b

Schalten Sie die Tonausgabe zunächst wieder ab und erweitern Sie das Programm um die Initialisierung der PIT-Kanäle 1 und 2. Die am Output des Kanals 2 angeschlossene LED soll mit einer Periodendauer von 0,5s blinken. Es ist wiederum Mode 3 zu benutzen. Da beide Kanäle hintereinander geschaltet (kaskadiert) sind, müssen Sie die benötigte Frequenzteilung auf beide Kanäle aufteilen. Außer der LED haben Sie diesmal keine weitere Kontrollmöglichkeit.

$$\text{Zählkonstante: } \frac{2\text{MHz}}{2\text{Hz}} / 2 = 1000000 / 2 = 500000$$

```

1     MOV AL, 0B6H; Kanal 2
2     OUT 57H, AL
3     MOV AL, 0FFH
4     OUT 56H, AL
5     MOV AL, 0FFH
6     OUT 56H, AL
7
8     MOV AL, 076H ;Kanal 1
9     OUT 57H, AL
10    MOV AL, 0FFH
11    OUT 55H, AL
12    MOV AL, 0FFH
13    OUT 56H, AL

```

## Grundaufgabe c

Die Tonausgabe von Kanal 0 wird wieder eingeschaltet. Sie soll jetzt aber nur noch dann aktiv sein, wenn gerade eine beliebige Taste in der blauen Tastenreihe gedrückt ist. Dazu müssen Sie in der Endlosschleife des Programms eine entsprechende Abfrage einbauen.

```

1 noton:  MOV AL, 59H
2          OUT 57H, AL
3 taste:  IN AL, 59H
4          AND AL, 0FFH
5          JZ noton      ;keine taste gedrueckt
6          JMP ton
7 ton:    MOV AL, 0C1H
8          OUT 54H, AL
9          MOV AL, 011H
10         OUT 54H, AL
11         JMP taste

```

## Fortgeschrittene Aufgabe d

Erweitern Sie das Programm dann so, dass den einzelnen Tasten unterschiedliche Frequenzen zugeordnet sind. Es wird angenommen, dass nicht mehrere Tasten gleichzeitig gedrückt werden. Das Blinken der LED von Aufgabe b) soll weiterhin funktionieren.

```

1 noton:  MOV AL, 59H
2          OUT 57H, AL
3 taste:  IN AL, 59H
4          MOV BL, AL
5          AND AL, 0FFH
6          JZ noton      ;keine taste gedrueckt
7          MOV AL, BL
8          AND AL, 001H   ; Taste A
9          JNZ tonA
10         MOV AL, BL
11         AND AL, 003H   ; Taste B
12         JNZ tonB
13         MOV AL, BL
14         AND AL, 004H   ;Taste C
15         JNZ tonC
16         JMP taste
17 tonA:  MOV AL, 0C1H
18         OUT 54H, AL
19         MOV AL, 011H
20         OUT 54H, AL
21         JMP taste
22 tonB:  MOV AL, 008H
23         OUT 54H, AL
24         MOV AL, 0FDH
25         OUT 54H, AL
26         JMP taste
27 tonC:  MOV AL, 000H
28         OUT 54H, AL
29         MOV AL, 0EFH
30         OUT 54H, AL
31         JMP taste

```

## A3: Matrixtastatur

Eine 4x4 Matrixtastatur ist über Zeilen- und Spaltenleitungen verbunden. Um eine gedrückte Taste zu erkennen müssen alle Zeilen nacheinander abgefragt werden. Bei jedem Abfrageschnitt erhält man die Information über die gedrückten Tasten jeweils einer Zeile.

### Grundaufgabe a

Alle Zeilen der Matrix je einmal abfragen und zurückkehren. Falls eine gedrückte Taste erkannt wurde, soll in einem gewählten Byteregister eine von Null verschiedene Tastennummer übergeben werden. Dies geschieht mit dem Unterprogramm *matr*.

Wie *matr* soll das Unterprogramm *wmatr* alle Zeilen der Matrix abfragen aber erst beim Erkennen eines Tastendrucks zurückkehren. Das bedeutet, dass es das Drücken einer Taste abwartet und dann deren Nummer übergibt.

Zur Durchführung sollen die Unterprogramme nacheinander in einem Hauptprogramm aufgerufen werden. Das Hauptprogramm selbst soll in einer ewigenSSchleife arbeiten und die erkannte Tastennummer binär auf einer der LED-Zeilen anzeigen.

```

1 main:
2     CALL wmatr           ; Tasten abfragen
3     MOV AL, BL          ; Ausgabe vorbereiten
4     OUT 5Dh, AL         ; Bits auf LED-Port
5     JMP main           ; Schleife
6 matr:
7     IN AL, 5Ah          ; Zeile einlesen
8     MOV BL, AL          ; in Register B
9     SHL BL, 4           ; Bits nach oben schieben
10    IN AL, 5Bh          ; Spalte einlesen
11    OR BL, AL           ; Spalten & Zeilen zusammenfassen
12    RET
13 wmatr:
14    CALL matr           ; Tasten abfragen
15    JNZ wmatr           ; weiter wenn keine Taste gedruickt
16 wloop: CALL matr       ; Taste abfragen
17    JZ wloop           ; weiter wenn Taste gedruickt
18    RET

```

Listing 1: aufgabe 3a

### Grundaufgabe b

Realisiere ein Programm, das die jeweils gedrückte Ziffer in lesbarer Darstellung auf der linken Stelle der Sieben-Segment-Anzeigen anzeigt. Tastaturbelegung:

0	1	2	3
4	5	6	7
8	9	-	-
-	-	-	-

```

1     MOV DX, 0Bh         ; Startwertpointer 7Seg.Anzeige
2 anf:
3     CALL wmatr         ; Tasten abfragen
4     CALL ziff          ; Ziffern abfragen
5     CALL anz           ; Ausgabe auf Anzeige
6     JMP anf

```

```
7 matr:
8     IN AL, 5Ah           ; Zeile einlesen
9     MOV BL, AL          ; in Register B sichern
10    SHL BL, 4           ; Bits nach oben schieben
11    IN AL, 5Bh          ; Spalte einlesen
12    OR BL, AL           ; Spalten & Zeilen in Byte zusammenfassen
13    RET
14 wmatr:
15    CALL matr           ; Tasten abfragen
16    JNZ wmatr           ; weiter wenn keine Taste gedrueckt
17 z2:  CALL matr           ; Taste abfragen
18    JZ w2               ; weiter wenn Taste gedrueckt
19    RET
20 ziff:
21    CMP BL, 17h
22    JNC z1
23    MOV BH, 3Fh
24    RET
25 z1:  CMP BL, 18h
26    JNC z2
27    MOV BH, 03h
28    RET
29 z2:  CMP BL, 20h
30    JNC z3
31    MOV BH, 6Dh
32    RET
33 z3:  CMP BL, 24h
34    JNC z4
35    MOV BH, 67h
36    RET
37 z4:  CMP BL, 33h
38    JNC z5
39    MOV BH, 53h
40    RET
41 z5:  CMP BL, 34h
42    JNC z6
43    MOV BH, 76h
44    RET
45 z6:  CMP BL, 36h
46    JNC z7
47    MOV BH, 7Eh
48    RET
49 z7:  CMP BL, 40h
50    JNC z8
51    MOV BH, 23h
52    RET
53 z8:  CMP BL, 65h
54    JNC z9
55    MOV BH, 7Fh
56    RET
57 z9:  CMP BL, 66h
58    JNC zq
59    MOV BH, 77h
60    RET
61 zq:  MOV BH, 04h
62    RET
63 anz:
64    MOV AL, BH
65    OUT DX, AL
```

66 RET

## Listing 2: aufgabe 3b

## Fortgeschrittene Aufgabe c

Erweitere das Programm so dass gedrückte Ziffern der Reihe nach nebeneinander angezeigt werden und beim Erreichen der letzten Stelle wieder links beginnt. Das Drücken einer nicht als Ziffer definierten Taste soll eine leere Stelle erzeugen.

```

1      MOV DX, 0BBh          ; Startwertpointer 7Seg.Anzeige
2  anf:
3      CALL wmatr           ; Tasten abfragen
4      CALL ziff           ; Ziffern abfragen
5      CALL anz            ; Ausgabe auf Anzeige
6      JMP anf
7  matr:
8      IN AL, 5Ah          ; Zeile einlesen
9      MOV BL, AL          ; in Register B sichern
10     SHL BL, 4           ; Bits nach oben schieben
11     IN AL, 5Bh          ; Spalte einlesen
12     OR BL, AL           ; Spalten & Zeilen in Byte zusammenfassen
13     RET
14  wmatr:
15     CALL matr           ; Tasten abfragen
16     JNZ wmatr           ; weiter wenn keine Taste gedruickt
17  w2:  CALL matr           ; Taste abfragen
18     JZ w2              ; weiter wenn Taste gedruickt
19     RET
20  ziff:
21     CMP BL, 17h
22     JNC z1
23     MOV BH, 3Fh
24     RET
25  z1:  CMP BL, 18h
26     JNC z2
27     MOV BH, 03h
28     RET
29  z2:  CMP BL, 20h
30     JNC z3
31     MOV BH, 6Dh
32     RET
33  z3:  CMP BL, 24h
34     JNC z4
35     MOV BH, 67h
36     RET
37  z4:  CMP BL, 33h
38     JNC z5
39     MOV BH, 53h
40     RET
41  z5:  CMP BL, 34h
42     JNC z6
43     MOV BH, 76h
44     RET
45  z6:  CMP BL, 36h
46     JNC z7
47     MOV BH, 7Eh
48     RET

```

```

49 z7:    CMP BL, 40h
50        JNC z8
51        MOV BH, 23h
52        RET
53 z8:    CMP BL, 65h
54        JNC z9
55        MOV BH, 7Fh
56        RET
57 z9:    CMP BL, 66h
58        JNC zq
59        MOV BH, 77h
60        RET
61 zq:    CMP BL, 77h
62        JNC z1
63        MOV BH, 04h
64        RET
65 anz:
66        MOV AL, BH
67        OUT DX, AL
68        CMP DX, 0B0h
69        JC rst
70        DEC DX
71        RET
72 rst:   MOV DX, 0BBh
73        RET

```

Listing 3: aufgabe 3c

## Fortgeschrittene Aufgabe d

Verhindere das Prellen der Tasten durch Software um doppelte Tastendrucke zu vermeiden.

```

1      MOV DX, 0BBh           ; Startwertpointer 7Seg.Anzeige
2  anf:
3      CALL wmatr           ; Tasten abfragen
4      CALL ziff           ; Ziffern abfragen
5      JMP anf
6  wmatr:
7      CALL matr
8      CMP BH, 0h
9      JNZ wmatr
10 lp:   CALL matr
11      CMP BH, 0h           ; Entprelltimer
12      JZ lp
13      RET
14  matr:
15      MOV ECX, 5000h
16  t:    DEC ECX
17      JNZ t
18      MOV AL, 01h
19      OUT 5Ah, AL
20      IN AL, 5Bh
21      MOV BL, AL
22  z0:
23      CMP BL, 01h
24      JNC z1
25      MOV BH, 3Fh
26      RET
27  z1:   CMP BL, 02h

```

```
28     JNC z2
29     MOV BH, 03h
30     RET
31 z2:   CMP BL, 04h
32     JNC z3
33     MOV BH, 6Dh
34     RET
35 z3:   CMP BL, 08h
36     JNC w2
37     MOV BH, 67h
38     RET
39 w2:   MOV AL, 02h
40     OUT 5Ah, AL
41     IN AL, 5Bh
42     MOV BL, AL
43 z4:   CMP BL, 01h
44     JNC z5
45     MOV BH, 53h
46     RET
47 z5:   CMP BL, 02h
48     JNC z6
49     MOV BH, 76h
50     RET
51 z6:   CMP BL, 04h
52     JNC z7
53     MOV BH, 7Eh
54     RET
55 z7:   CMP BL, 08h
56     JNC w3
57     MOV BH, 23h
58     RET
59 w3:   MOV AL, 04h
60     OUT 5Ah, AL
61     IN AL, 5Bh
62     MOV BL, AL
63 z8:   CMP BL, 01h
64     JNC z9
65     MOV BH, 7Fh
66     RET
67 z9:   CMP BL, 02h
68     JNC zq
69     MOV BH, 77h
70     RET
71 zq:   MOV BH, 0h
72     RET
73 anz:
74     MOV AL, BH
75     OUT DX, AL
76     CMP DX, 0B0h
77     JC rst
78     DEC DX
79     RET
80 rst:  MOV DX, 0BBh
81     RET
```

Listing 4: aufgabe 3d

## Zusatzaufgabe

Realisiere einen einfachen Taschenrechner der einstellige nichtnegative Dezimalzahlen addiert und das ein- bis zweistellige Ergebnis auf der Sieben-Segment-Anzeige anzeigt. Definiere dafür Tasten für „+“ und „=“.

```
1 | 2 | 3 | 4
5 | 6 | 7 | 8
0 | 0 | 0 | 0
+ | 0 | 0 | =
```

Listing 5: addition