Technische Universität Ilmenau
Fakulät 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  ;Groessee der Verzoegerung
3           MOV   [verzoe],EDX ;Verzoegerung 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

```
anf:    MOV     EDX,400000H
        MOV     [verzoe],EDX

        MOV     AL, 80H     ;Startwert fuer LED Reihe
lauf:   OUT     5CH, AL     ;Wert auf LED Reihe schreiben
        CALL    zeit        ;warten
        ROR     AL, 1       ;Bits um 1 nach rechts
        JMP     lauf        ;Schleife wiederholen

zeit:   MOV     ECX,[verzoe]
z1:     DEC     ECX
        JNZ     z1
        RET
```

Lauflicht mit Geschwindigkeitsumschalter

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

```
anf:    MOV     Al, 80H

lauf:   MOV     EDX, 400000H    ; Wert fuer "langsam"
        MOV     [verzoe], EDX   ;"langsam" in Speicher
        OUT     5CH, AL         ;LED Reihe schreiben
        MOV     BL, AL          ;AL speichern
        IN      AL, 58H         ;Schalter einlesen
        BT      AL, 7           ;7. Bit von AL in Carry Flag
        JNC     langsam         ;Carry Flag = 0, schalter unten
        MOV     EDX, 200000H    ; Wert fuer "schnell"
        MOV     [verzoe], EDX   ;"schnell" in Speicher
        CMC                     ;Carry Flag umschalten (0)

langsam: CALL   zeit            ;warten
        MOV     AL, BL          ;AL aus speicher zurueck
        ROR     AL,1            ;Bits um 1 nach rechts
        JMP     anf             ;Schleife wiederholen

zeit:   MOV     ECX,[verzoe]
z1:     DEC     ECX
        JNZ     z1
        RET
```

Lauflicht verändert Richtung

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

```
anf:    MOV     Al, 80H
lauf:   MOV     EDX, 400000H    ; Wert fuer "langsam"
        MOV     [verzoe], EDX   ;"langsam" in Speicher
```

```
        OUT     5CH, AL         ;LED Reihe schreiben
        MOV     BL, AL          ;AL speichern
        IN      AL, 58H         ;Schalter einlesen
        BT      AL, 7           ;7. Bit von AL in Carry Flag
        JNC     langsam         ;Carry Flag = 0, Schalter unten
        MOV     EDX, 200000H    ; Wert fuer "schnell"
        MOV     [verzoe], EDX   ;"schnell" in Speicher
        CMC                     ;Carry Flag umschalten
langsam: CALL   zeit            ;warten
        MOV     AL, BL          ;AL aus speicher zurueck
        BT      AL, 0           ;0. Bit von AL in Carry Flag
        JNC     rechts          ;Carry Flag = 1; Schalter oben
        ROL     AL,1            ;Bits um 1 nach links
        CMC                     ;Carry Flag umschalten (0)
        JMP     anf             ;Schleife wiederholen
rechts: ROR     AL, 1           ;Bits um 1 nach rechts
        JMP     anf             ;Schleife wiederholen
zeit:   MOV     ECX,[verzoe]
z1:     DEC     ECX
        JNZ     z1
        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.

```
anf:    MOV     Al, 80H
lauf:   MOV     EDX, 400000H    ; Wert fuer "langsam"
        MOV     [verzoe], EDX   ;"langsam" in Speicher
        MOV     BL, AL          ;Kopie von AL anlegen
        IN      AL, 59H         ;Tastenreihe einlesen
        AND     AL, FFH         ;UND Operation mit FF
        JZ      nopress         ;kein Schalter gedrueckt
        NOT     BL              ;BL invertieren
        MOV     AL, BL          ;AL ueberschreiben
nopress: OUT    5CH, AL         ;LED Reihe schreiben
        IN      AL, 58H         ;Schalter einlesen
        BT      AL, 7           ;7. Bit von AL in Carry Flag
        JNC     langsam         ;Carry Flag = 0, Schalter unten
        MOV     EDX, 200000H    ; Wert fuer "schnell"
        MOV     [verzoe], EDX   ;"schnell" in Speicher
        CMC                     ;Carry Flag umschalten
langsam: CALL   zeit            ;warten
        MOV     AL, BL          ;AL aus speicher zurueck
        BT      AL, 0           ;0. Bit von AL in Carry Flag
        JNC     rechts          ;Carry Flag = 1; Schalter oben
        ROL     AL,1            ;Bits um 1 nach links
        CMC                     ;Carry Flag umschalten (0)
        JMP     anf             ;Schleife wiederholen
rechts: ROR     AL, 1           ;Bits um 1 nach rechts
        JMP     anf             ;Schleife wiederholen
zeit:   MOV     ECX,[verzoe]
z1:     DEC     ECX
        JNZ     z1
        RET
```

Zusatzaufgabe

Erweiterungen des Programms nach eigenen Ideen:

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

```
anf:     MOV     Al, 80H
         MOV     EDI, 0
         MOV     ESI, OFFSET ziff
lauf:    MOV     EDX, 400000H    ; Wert fuer "langsam"
         MOV     [verzoe], EDX   ;"langsam" in Speicher
         MOV     BL, AL          ;Kopie von AL anlegen
         IN      AL, 59H         ;Tastenreihe einlesen
         AND     AL, FFH         ;UND Operation mit FF
         JZ      nopress         ;kein Schalter gedrueckt
         NOT     BL              ;BL invertieren
         MOV     AL, BL          ;AL ueberschreiben
nopress: OUT     5CH,AL           ;LED Reihe links schreiben
         NOT     AL              ;AL negieren
         OUT     5DH,AL          ;LED Reihe rechts schreiben
         MOV     BH,[ESI+EDI-1]  ;Sieben Segment berechnen
         OUT     0B0H,BH         ;Sieben Segment schreiben
         DEC     EDI             ;Sieben Segment runterzaehlen
         JZ      timer           ;Timer auf 0 setzen
         IN      AL, 58H         ;Schalter einlesen
         BT      AL, 7           ;7. Bit von AL in Carry Flag
         JNC     langsam         ;Carry Flag = 0, Schalter unten
         MOV     EDX, 200000H    ; Wert fuer "schnell"
         MOV     [verzoe], EDX   ;"schnell" in Speicher
         CMC                     ;Carry Flag umschalten
langsam: CALL    zeit            ;warten
         MOV     AL, BL          ;AL aus speicher zurueck
         BT      AL, 0           ;0. Bit von AL in Carry Flag
         JNC     rechts          ;Carry Flag = 1; Schalter oben
         ROL     AL,1            ;Bits um 1 nach links
         CMC                     ;Carry Flag umschalten (0)
         JMP     anf             ;Schleife wiederholen
rechts:  ROR     AL, 1           ;Bits um 1 nach rechts
         JMP     anf             ;Schleife wiederholen
timer:   MOV     BH, 0FFH
         RET
zeit:    MOV     ECX,[verzoe]
z1:      DEC     ECX
         JNZ     z1
         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.

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

```
        MOV AL, 36H; Steuerbyte 00110110
        OUT 57H, AL
        MOV AL, 0C1H; LSB
        OUT 54H, Al
        MOV AL, 011H; MSB
        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.

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

```
        MOV AL, 0B6H; Kanal 2
        OUT 57H, AL
        MOV AL, 0FFH
        OUT 56H, AL
        MOV AL, 0FFH
        OUT 56H, AL

        MOV AL, 076H ;Kanal 1
        OUT 57H, AL
        MOV AL, 0FFH
        OUT 55H, AL
        MOV AL, 0FFH
        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.

```
noton:  MOV AL, 59H
        OUT 57H, AL
taste:  IN AL, 59H
        AND AL, 0FFH
        JZ noton        ;keine taste gedrueckt
        JMP ton
ton:    MOV AL, 0C1H
        OUT 54H, AL
        MOV AL, 011H
        OUT 54H, AL
        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.

```
noton:  MOV AL, 59H
        OUT 57H, AL
taste:  IN AL, 59H
        MOV BL, AL
        AND AL, 0FFH
        JZ noton        ;keine taste gedrueckt
        MOV AL, BL
        AND AL, 001H    ; Taste A
        JNZ tonA
        MOV AL, BL
        AND AL, 003H    ; Taste B
        JNZ tonB
        MOV AL, BL
        AND AL, 004H    ;Taste C
        JNZ tonC
        JMP taste
tonA:   MOV AL, 0C1H
        OUT 54H, AL
        MOV AL, 011H
        OUT 54H, AL
        JMP taste
tonB:   MOV AL, 008H
        OUT 54H, AL
        MOV AL, 0FDH
        OUT 54H, AL
        JMP taste
tonC:   MOV AL, 000H
        OUT 54H, AL
        MOV AL, 0EFH
        OUT 54H, AL
        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 ëwigenSSchleife arbeiten und die erkannte Tastennummer binär auf einer der LED-Zeilen anzeigen.

```
main:
        CALL wmatr              ; Tasten abfragen
        MOV AL, BL              ; Ausgabe vorbereiten
        OUT 5Dh, AL             ; Bits auf LED-Port
        JMP main                ; Schleife
matr:
        IN AL, 5Ah              ; Zeile einlesen
        MOV BL, AL              ; in Register B
        SHL BL, 4               ; Bits nach oben schieben
        IN AL, 5Bh              ; Spalte einlesen
        OR BL, AL               ; Spalten & Zeilen zusammenfassen
        RET
wmatr:
        CALL matr               ; Tasten abfragen
        JNZ wmatr               ; weiter wenn keine Taste gedrueckt
wloop:  CALL matr               ; Taste abfragen
        JZ wloop                ; weiter wenn Taste gedrueckt
        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	-	-
-	-	-	-

```
        MOV DX, 0BBh            ; Startwertpointer 7Seg.Anzeige
anf:
        CALL wmatr              ; Tasten abfragen
        CALL ziff               ; Ziffern abfragen
        CALL anz                ; Ausgabe auf Anzeige
        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 w2:     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 gedrueckt
17  w2:     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
```

```
z7:     CMP BL, 40h
        JNC z8
        MOV BH, 23h
        RET
z8:     CMP BL, 65h
        JNC z9
        MOV BH, 7Fh
        RET
z9:     CMP BL, 66h
        JNC zq
        MOV BH, 77h
        RET
zq:     CMP BL, 77h
        JNC z1
        MOV BH, 04h
        RET
anz:
        MOV AL, BH
        OUT DX, AL
        CMP DX, 0B0h
        JC rst
        DEC DX
        RET
rst:    MOV DX, 0BBh
        RET
```

Listing 3: aufgabe 3c

Fortgeschrittene Aufgabe d

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

```
        MOV DX, 0BBh            ; Startwertpointer 7Seg.Anzeige
anf:
        CALL wmatr              ; Tasten abfragen
        CALL ziff               ; Ziffern abfragen
        CALL anz                ; Ausgabe auf Anzeige
        JMP anf
matr:
        MOV ECX, 50000h         ; Entprelltimer
t:      DEC ECX
        JNZ t
        IN AL, 5Ah              ; Zeile einlesen
        MOV BL, AL              ; in Register B sichern
        SHL BL, 4               ; Bits nach oben schieben
        IN AL, 5Bh              ; Spalte einlesen
        OR BL, AL               ; Spalten & Zeilen in Byte zusammenfassen
        RET
wmatr:
        CALL matr               ; Tasten abfragen
        JNZ wmatr               ; weiter wenn keine Taste gedrueckt
w2:     CALL matr               ; Taste abfragen
        JZ w2                   ; weiter wenn Taste gedrueckt
        RET
ziff:
        CMP BL, 17h
        JNC z1
        MOV BH, 3Fh
        RET
```

```
z1:     CMP BL, 18h
        JNC z2
        MOV BH, 03h
        RET
z2:     CMP BL, 20h
        JNC z3
        MOV BH, 6Dh
        RET
z3:     CMP BL, 24h
        JNC z4
        MOV BH, 67h
        RET
z4:     CMP BL, 33h
        JNC z5
        MOV BH, 53h
        RET
z5:     CMP BL, 34h
        JNC z6
        MOV BH, 76h
        RET
z6:     CMP BL, 36h
        JNC z7
        MOV BH, 7Eh
        RET
z7:     CMP BL, 40h
        JNC z8
        MOV BH, 23h
        RET
z8:     CMP BL, 65h
        JNC z9
        MOV BH, 7Fh
        RET
z9:     CMP BL, 66h
        JNC zq
        MOV BH, 77h
        RET
zq:     CMP BL, 77h
        JNC z1
        MOV BH, 04h
        RET
anz:
        MOV AL, BH
        OUT DX, AL
        CMP DX, 0B0h
        JC rst
        DEC DX
        RET
rst:    MOV DX, 0BBh
        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