Übungsaufgaben 1

1. Lauflicht mit Port C. Diese Aufgabe dient dazu, mit dem Simulator vertraut zu werden. Die Programme werden im Schrittbetrieb ausgeführt, so daß Verzögerungsroutinen nicht erforderlich sind.

   a) Eine Eins läuft von rechts nach links.
   b) Die links außen angekommene Eins läuft zurück.
   c) Es laufen zwei Einsen von außen nach innen und dann wieder von innen nach außen.

2. Einfache Rechenaufgaben mit längeren Binärzahlen. Operand 1 in den Registern 1 bis 3, Operand 2 in den Registern 10 bis 12. Werteingabe im Simulator.

   a) 3E26 + 65AB
   b) Ergebnis – 65AB
   c) 35*31 (vorzeichenlos)
   d) -35*-31 (beide mit Vorzeichen)
   e) -35*31 (beide mit Vorzeichen)
   f) -35*31 (31 vorzeichenlos)

3. Schreiben Sie ein Programm, das zwei 16stellige Binärzahlen miteinander multipliziert. Nutzen Sie dazu die Multiplikationsbefehle. Operand 1 in den Registern 2 und 3, Operand 2 in den Registern 10 und 11. Es dürfen beliebige weitere Register verwendet werden.

4. In den Registern r16 bis r18 steht eine 24 Bits lange vorzeichenlose Binärzahl. Schreiben Sie ein Programm, das diese Zahl mit Drei multipliziert (eine Denksportaufgabe ...). Sie dürfen hierzu beliebig viele weitere Register benutzen.

5. Zu einem 24-Bit-Wort in den Registern r1, r2, r3 (Abb. 1) ist der Festwert 55H zu subtrahieren. Bei Bedarf dürfen weitere Register nach Belieben genutzt werden. Deren bisheriger Inhalt darf aber nicht verlorengehen.

r1	Bits 7...0
r2	Bits 15...8
r3	Bits 23...16

Abb. 1

6. Die Register r6, r7 und r20, r21 enthalten jeweils 16 Bits lange vorzeichenlose Binärzahlen Z1 und Z2 (Abb. 2). Schreiben Sie einen Programmablauf, der Z1 – Z2 nach den Regeln der Sättigungsarithmetik berechnet und das Ergebnis in Z2 speichert. Nach Ausführung der Rechnung müssen alle anderen Register außer r20, r21 den gleichen Inhalt haben wie vorher.

r6	Z1_0
r7	Z1_1
r20	Z2_0
r21	Z2_1

Abb. 2

7. Eine 16-Bit-Zahl in den Registern r1 und r2 (Abb. 3) ist um drei Bits nach links zu verschieben. Die frei werdenden Bitpositionen sind mit Nullen aufzufüllen.

r1	Bits 7...0
r2	Bits 15...8

Abb. 3

8. Eine 16-Bit-Zahl in den Registern r20 und r21 (Abb. 4) ist um zwei Bits arithmetisch nach rechts zu verschieben.

r20	Bits 7...0
r21	Bits 15...8

Abb. 4

9. Eine 16-Bit-Zahl in den Registern r20 und r21 (Abb. 5) ist um drei Bits nach links zu verschieben. Die frei gewordenene Bitpositionen sind mit Einsen aufzufüllen.

r20	Bits 7...0
r21	Bits 15...8

Abb. 5

10. Schreiben Sie ein Programmstück, das den Inhalt der Bits 3...0 des Registers TEMP in eine Hexadezimalziffer wandelt, die als ASCII-Zeichen angegeben wird. Das Zeichen soll im Register HEX abgelegt werden (alle Registeradressen > 15). Sie dürfen beliebig viele weitere Register verwenden. Die Bits 7...4 des Registers TEMP können beliebige Werte enthalten. Der gesamte Inhalt von TEMP soll erhalten bleiben. *Hinweis:* Die ASCII-Codes der Ziffern 0...9: 30H...39H, der Zeichen A...F: 41H...46H.

11. Der A/D-Wandler eines Mikrocontrollers liefert einen Wert von 10 Bits Länge. Formen Sie diesen Wert in einen 8-Bit-Wert um (Abb. 6). Dienstvorschrift: Erst ADCL lesen, dann ADCH.

ADC Data Register ADCL und ADCH:

Register	7	6	5	4	3	2	1	0
ADCH	-	-	-	-	-	.	ADC9	ADC8
ADCL	ADC7							ADC0

Das gewünschte Ergebnis:

7	6	5	4	3	2	1	0
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2

Abb. 6

12. Eine 16-Bit-Binärzahl ist zu runden. Hierzu sind die niedrigstwertigen Bits 1 und 0 auszuwerten:

    - Bitbelegung 0,0: nichts tun.
    - Bitbelegung 1,0: durch 0, 0 ersetzen (abrunden).
    - Bitbelegungen 1,0 und 1,1: durch 0,0 ersetzen und den Rest der Zahl (von Bit 2 an) um Eins erhöhen (aufrunden). Die Zahl darf aber nicht größer werden als der Endwert FFFCH (Sättigungsarithmetik).

    Die Binärzahl steht in den Registern ZL (Bits 7...0) und ZH (Bits 15...8). Sie dürfen beliebig viele Arbeitsregister verwenden.

13. Abb. 7 veranschaulicht einen elektronischen Würfel. So lange die Taste (KEY#) betätigt ist, sollen die Bitmuster der Zahlen 1 bis 6 in schneller Folge zyklisch ausgegeben werden. Ist die Taste losgelassen, bleibt das letzte Bitmuster stehen. Schreiben Sie ein Programm, das diese Funktion ausführt. Es soll mit der Initialisierung des Ports A beginnen. Das erste Bitmuster ist die Eins. (Zur Erinnerung: # bedeutet "aktiv Low.)

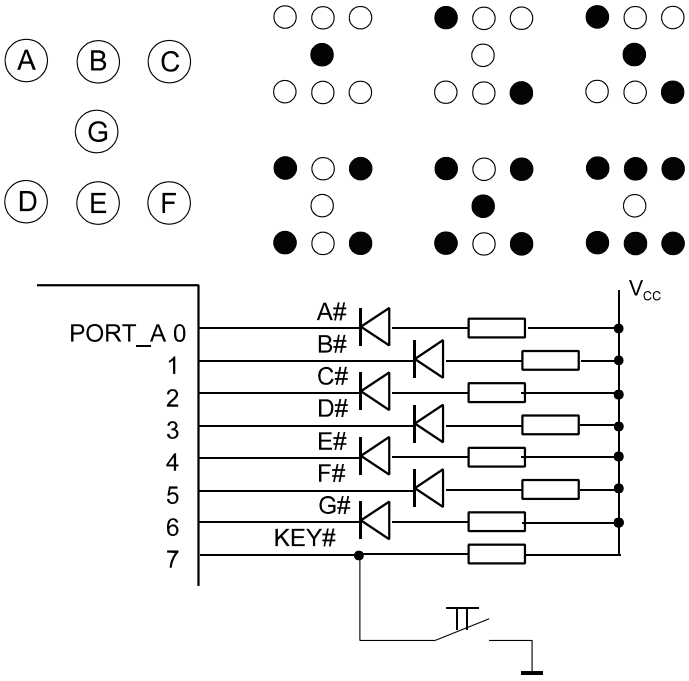

Abb. 7

14. Schreiben Sie ein Assemblerprogrammstück, das folgenden Ablauf implementiert (Abb. 8).

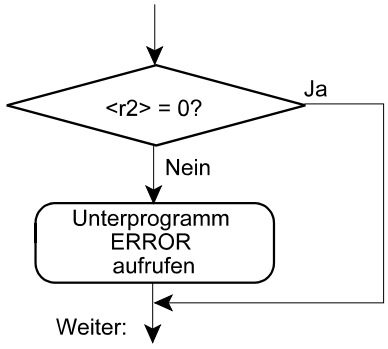

Abb. 8

15. Ein Tastenfeld ist an Port B angeschlossen. Es soll abgefragt werden, ob die Taste an Bitposition 3 betätigt ist oder nicht (Tastenwirkung: aktiv low). Das Programmstück:

```
WAIT_3:
        SBIC    PORTB,3
        RJMP    WAIT_3
```

Wird das so funktionieren? Geben Sie ggf. eine korrigierte Befehlsfolge an.

16. Ihr Programm enthält diese Befehlsfolge:

```
CP      TEMP, LIMIT
BRNE    BEGIN
IN      TEMP, PIND
```

Bisher lief alles. Nachdem jedoch an verschiedenen (anderen) Stellen im Programm geändert wurde, wird beim Assemblieren der BRNE-Befehl als fehlerhaft gekennzeichnet:

```
xyz.asm(80): error: Relative branch out of reach
```

a) Weshalb?
b) Wie helfen Sie sich?

17. Schreiben Sie einen Programmablauf, der über Port A, Bitpositionen 1 und 0 das nachfolgend gezeigte Bitmuster ausgibt (Abb. 9).

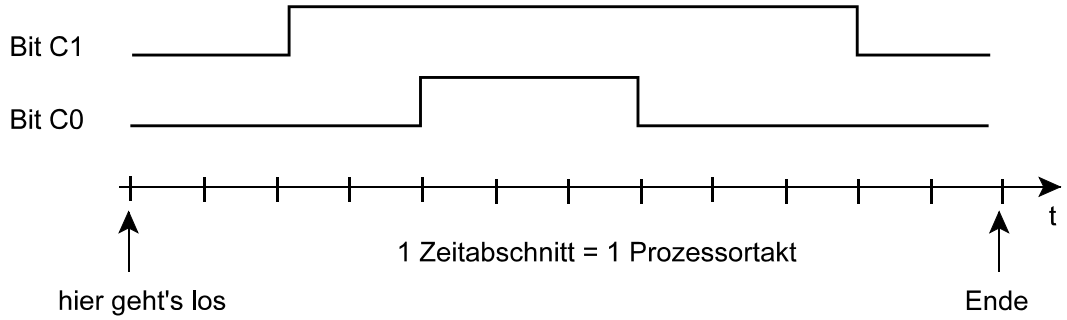

Abb. 9

18. Ein Mikrocontroller soll eine Zweifarben-LED ansteuern (Abb. 10). Anschluß: Pin 1 an Port C, Bit 0, Pin 2 an Port C, Bit 1 (natürlich über Serienwiderstand...). Die weiteren Bits 7...2 sollen zu Ausgabezwecken verwendet werden.

a) wie ist die LED anzusteuern, damit sie (1) grün, (2) rot und (3) gar nicht leuchtet? Geben Sie die Belegungen mit Einsen und Nullen an, z. B. Pin 1 = 1, Pin 2 = 0.
b) Initialisieren Sie die Port C so, daß die LED anfänglich nicht leuchtet.
c) Schreiben Sie ein Programm, das die LED zyklisch (ewig) rot – grün – rot ... blinken läßt. Die Blinkzeit erzeugen Sie mit einem (fertigen) Unterprogramm MILLISEC, dem die Zeit in Millisekunden in den Registern r24 und r25 übergeben wird. Stellen Sie das Blinkintervall auf 500 ms ein.

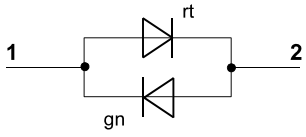

Abb. 10