

# Automatisierungstechnik AU1

## – Merkblatt zur Klausur –

7. 9. 2007

### 1. Themen:

- Grundbegriffe der Rechnerarchitektur
- Grundlagen der Assemblerprogrammierung
- Ausdrucksmittel der Assemblersprache
- Aufbau eines Assemblerprogramms
- Architektur und Befehlsliste der Atmel-AVR-Mikrocontroller
- Die E-A-Ports der Atmel-AVR-Mikrocontroller
- Elementare Programmabläufe

Die Mikrocontrollerprogrammierung in C kommt nicht dran.

Wichtige Punkte:

- Initialisierung
- Elementare Ein- und Ausgabe
- Zugriffe auf Daten im Programmspeicher
- Der Stack
- Programmablaufsteuerung
- Elementare Informationswandlungen und Verknüpfungen (wie beispielsweise Verschiebung oder Addition)
- Unterprogrammrufer
- elementare Unterbrechungsbehandlung (vgl. Praktikum)

### 2. Informationsquellen:

- Automatisierungstechnik AU1: Übersicht
- Einführung in die Mikrocontroller-Programmierung am Beispiel Atmel AVR
- Elementare Programmieretechniken, Teil 1
- Die Musterprogramme zum Praktikum
- Die Original-Befehlsbeschreibung der Fa. Atmel (8 Bit AVR Instruction Set). Im Grunde genügt die Kurzübersicht der Seiten 10 bis 15 (Instruction Set Summary)
- Beliebige Lehrbücher zur AVR-Programmierung
- Einschlägiges Material aus dem Internet

### 3. Die Klausuraufgaben umfassen:

- Wissensfragen zur Rechnerarchitektur, zu den E-A-Ports und zur Assemblerprogrammierung (beschränkt auf den Stoff, wie er in der Vorlesung und im Praktikum behandelt wurde),
- vor allem aber das Schreiben kleiner Programmstücke.

Eine besondere Einarbeitung in die Atmel-Peripherie (A/D-Wandler, serielle Schnittstelle usw.) ist nicht erforderlich. Falls in den Aufgaben periphere Einheiten angesprochen werden, werden die einschlägigen Angaben (Registerbelegungen usw.) mitgeliefert.

## Übungsaufgaben

1. Erläutern Sie kurz, worin sich CISC- und RISC-Architekturen voneinander unterscheiden. Zu welchem Typ gehört die AVR-Architektur?
2. Wozu dienen **byte**-Anweisungen?
3. Skizzieren Sie (in Form eines Schaltbildes für eine Bitposition) den Aufbau eines E-A-Ports.
4. Die Register r2 bis r5 enthalten jeweils 16 Bits lange Binärzahlen X und Y. Schreiben Sie einen Programmablauf für die Subtraktion X-Y. Hierbei soll das Ergebnis in den Registern r2 und r3 zu stehen kommen.

r2	LO_X
r3	HI_X
r4	LO_Y
r5	HI_Y

5. Wir beziehen uns auf Aufgabe 4. Jetzt ist aber die Subtraktion X-Y so auszuführen, daß das Ergebnis in den Registern r18 und r19 zu stehen kommt. Die Inhalte der Register r2 bis r5 sollen unverändert erhalten bleiben.
6. Initialisieren Sie Port B so, daß Bitposition 3 als Ausgang konfiguriert ist und anfänglich mit dem Wert 1 belegt wird. Alle anderen Bitpositionen sollen als Eingänge wirken.
7. Schreiben Sie einen Programmablauf, der über Port B, Bitposition 3 das in Abb. 1 gezeigte Bitmuster ausgibt.

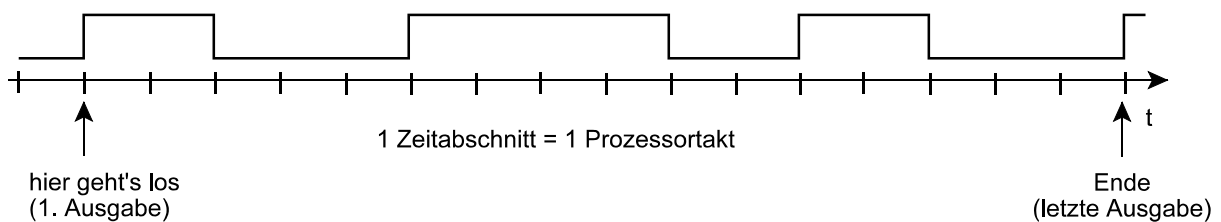


Abb. 1

8. Schreiben Sie eine einfache Zeitschleife in Form eines Unterprogramms TIME\_COUNT. Dessen Funktionen:
  - Laden des Registers r16 mit dem Festwert 0x38,
  - Vermindern des Registerinhaltes um 1,
  - Rückkehr, wenn der Registerinhalt den Wert Null erreicht hat.

Sorgen Sie dabei dafür, daß der ursprüngliche Inhalt des Registers r16 erhalten bleibt (mit anderen Worten, das rufende Programm darf gar nicht merken, daß r16 im Unterprogramm benutzt wurde).

9. Im Programmspeicher stehen mehrere Zeichenketten, die Warnungsnachrichten darstellen. Die Anfangsadresse der betreffenden Nachricht wird im Register Z übergeben.

Beispiel:

```
TEXT_1:    .db "Temperaturwarnung",0
TEXT_2:    .db "Kühlwasserkreislauf",0
```

usw.

- a) Wie laden Sie die Adresse einer bestimmten Nachricht (Beispiel: TEXT\_2) ins Register Z?  
 b) Schreiben Sie ein Unterprogramm MSG\_OUT, das den derart adressierten Text anzeigt. Zur eigentlichen Ausgabe dient ein fertiges Unterprogramm BYTE\_OUT, dem das jeweils auszugebende Datenbyte im Register TEMP übergeben wird. Mit Ausnahme der Register TEMP und Z sollen alle anderen Registerinhalte erhalten bleiben.
10. Beim Aufruf eines Unterprogramms werden Register in den Stack gerettet. Das Unterprogramm beginnt mit den folgenden Befehlen. Geben Sie eine dazu passende Befehlsfolge an (im Anschluß an die Marke LEAVE), mit der das Unterprogramm verlassen werden kann.

```
PUSH      TEMP
IN        TEMP, SREG
PUSH      TEMP
PUSH      r5
PUSH      r6
PUSH      r24
PUSH      r25
...

```

LEAVE:

11. Der E-A-Port A hat folgende symbolische Adressen: DDRA, PINA, PORTA. Welche Adresse verwenden Sie
- a) um eine an ein E-A-Pin angeschlossene LED ein- oder auszuschalten?  
 b) um abzufragen, ob eine an den Port angeschlossene Taste betätigt wurde?
12. Schreiben Sie ein Assemblerprogrammstück, das folgenden Ablauf implementiert (Abb. 2).

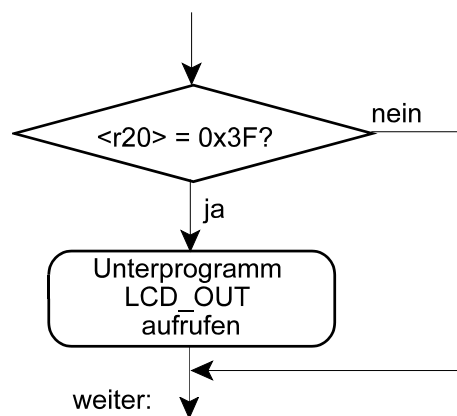


Abb. 2

## Wichtige Maschinenbefehle

Befehl	Wirkung
<b>LDI rd, imm</b>	Lädt einen Direktwert in ein Register. Beispiel: <i>LDI r17, 0x22</i>
<b>MOV rd, rr</b>	Transportiert Inhalt des Registers rr in das Register rd. Beispiel: <i>MOV r2, r22</i>
<b>LPM</b>	Lädt Byte aus Programmspeicher in Register r0. Adresse in Register Z (r31, r30)
<b>OUT port, rr</b>	Ausgabe eines Registerinhaltes. Beispiel: <i>OUT portd, r17</i>
<b>IN rd, port</b>	Eingabe in ein Register. Beispiel: <i>IN r17, pind</i>
<b>SBI port, bit</b>	Setzen Bit in E-A-Port. Beispiel: <i>SBI porta, 3</i>
<b>CBI port, bit</b>	Löschen Bit in E-A-Port. Beispiel: <i>CBI porta, 3</i>
<b>SBIC port, bit</b>	Bit in E-A-Port abfragen. Folgebefehl überspringen, wenn = 0. Beispiel: <i>SBIC porta, 3</i>
<b>SBIS port, bit</b>	Bit in E-A-Port abfragen. Folgebefehl überspringen, wenn = 1. Beispiel: <i>SBIS porta, 3</i>
<b>NOP</b>	keine Wirkung (Leerbefehl). Es wird nur eine Taktperiode verbraucht
<b>JMP label</b>	Unbedingte Verzweigung. Beispiel: <i>JMP anfang</i>
<b>BRNE label</b>	Verzweigen, wenn Ergebnis ungleich Null. Beispiel: <i>BRNE anfang</i>
<b>BREQ label</b>	Verzweigen, wenn Ergebnis gleich Null. Beispiel: <i>BREQ ende</i>
<b>CALL label</b>	Unterprogrammrufer. Beispiel: <i>CALL warten</i>
<b>RET</b>	Rückkehr aus Unterprogramm
<b>PUSH rr</b>	Register rr in Stack retten. Beispiel: <i>PUSH r20</i>
<b>POP rd</b>	Register rd aus Stack laden. Beispiel: <i>POP r20</i>
<b>AND rd, rs</b>	Register konjunktiv verknüpfen $\langle rd \rangle := \langle rd \rangle \text{ AND } \langle rs \rangle$ . Beispiel: <i>AND r12, r3</i>
<b>ANDI rd, imm</b>	konjunktive Verknüpfung mit Direktwert. Beispiel: <i>ANDI r12, 0b11110111</i>
<b>OR rd, rs</b>	Register disjunktiv verknüpfen $\langle rd \rangle := \langle rd \rangle \text{ OR } \langle rs \rangle$ . Beispiel: <i>AND r12, r3</i>
<b>ORI rd, imm</b>	disjunktive Verknüpfung mit Direktwert. Beispiel: <i>ORI r12, 0b11110111</i>
<b>COM rd</b>	Bitweise Negation (Einerkomplement). Beispiel: <i>COM r17</i>
<b>ADD rd, rs</b>	Registerinhalte zueinander addieren. $\langle rd \rangle := \langle rd \rangle + \langle rs \rangle$ . Beispiel: <i>ADD r12, r4</i>
<b>ADC rd, rs</b>	Addieren mit Eingangsübertrag. $\langle rd \rangle := \langle rd \rangle + \langle rs \rangle + \text{CF}$ . Beispiel: <i>ADC r13, r5</i>
<b>ADIW rd, imm</b>	Addieren Direktwert zu Wort ( Register 24, 26, 28, 30). Beispiel: <i>ADC r24, 12</i>
<b>SUBI rd, imm</b>	Subtrahieren Direktwert. Beispiel: <i>SBI r16, 0x04</i>
<b>SUB rd, rs</b>	Registerinhalte voneinander subtrahieren. $\langle rd \rangle := \langle rd \rangle - \langle rs \rangle$ . Beispiel: <i>SUB r12, r4</i>
<b>SBC rd, rs</b>	Subtrahieren mit Eingangsübertrag. $\langle rd \rangle := \langle rd \rangle - \langle rs \rangle - \text{CF}$ . Beispiel: <i>SBC r13, r5</i>
<b>SBIW rd, imm</b>	Subtrahieren Direktwert von Wort ( Register 24, 26, 28, 30). Beispiel: <i>SBIW r26, 10</i>
<b>CP rd, rs</b>	Registerinhalte vergleichen (Subtraktion $\langle rd \rangle - \langle rs \rangle$ ). Beispiel: <i>CP r12, r22</i>
<b>CPI rd, imm</b>	Vergleichen mit Direktwert (Subtraktion $\langle rd \rangle - \text{imm}$ ). Beispiel: <i>CPI r16, 0x33</i>
<b>ROL rd</b>	Linksrotieren über CF (CF => Bit 0, Bit 7 => CF): Beispiel: <i>ROL r17</i>
<b>ROR</b>	Rechtsrotieren über CF (CF => Bit 7, Bit 0 => CF). Beispiel: <i>ROR r17</i>
<b>LSL</b>	Linksverschieben. Bit 7 nach CF, Bit 0 wird mit 0 gefüllt. Beispiel: <i>LSL r17</i>
<b>LSR</b>	Rechtsverschieben. Bit 0 nach CF, Bit 7 wird mit 0 gefüllt. Beispiel: <i>LSR r17</i>

Achtung: Direktwertverknüpfungen nur mit Registern r16...r31. Weitere Einzelheiten s. Befehlsbeschreibung.