

Automatisierungstechnik AU1

Klausur vom 26. 9. 2007

Aufgaben

Hinweis:

Die meisten der folgenden Aufgaben sind durch kurze Assemblerprogrammstücke zu lösen. Programmieren Sie nicht mehr, als wirklich verlangt ist. Sollten Sie Arbeitsregister benötigen, so verwenden Sie dafür symbolische Bezeichnungen TEMP, TEMP1, TEMP2 o. ä. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

1. 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.

(5 Punkte)

2. 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?

(10 Punkte)

3. Erläutern Sie kurz (mit Skizze) den wesentlichen Unterschied zwischen der v. Neumann-Architektur und der Harvard-Architektur. Zu welchem Typ gehört Atmel AVR?

(10 Punkte)

4. In den Begriffen CISC und RISC stehen die Anfangsbuchstaben C für Complex und R für Reduced. Aber auch RISC-Maschinen können weit über einhundert Befehle haben. Erläutern Sie kurz, in welcher Hinsicht diese Maschinen wirklich als "reduziert" anzusprechen sind.

(10 Punkte)

5. Ein Mikrocontroller soll eine Zweifarben-LED (Abb. 1) ansteuern. Anschluß: Pin 1 an Port B, Bit 0, Pin 2 an Port B, 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 B 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 (haben wir im Praktikum ausprobiert). Stellen Sie das Blinkintervall auf 500 ms ein.

(15 Punkte)

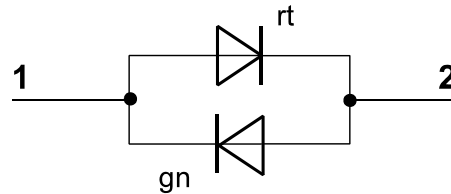


Abb. 1

6. Schreiben Sie eine Unterbrechungsbehandlungsroutine, die Zeichen auf eine LCD-Anzeige ausgibt. Die Zeichen kommen von der seriellen Schnittstelle. Jedes ankommende Zeichen löst die zu schreibende Behandlungsroutine aus. Es befindet sich im Register UDR der seriellen Schnittstelle. Zur Ausgabe auf die Anzeige ist ein fertiges Unterprogramm DISPLAY zu verwenden. Es erhält das darzustellende Zeichen im Register r16. Das Unterprogramm benötigt zudem die Register r18, r19, r24 und r25.

(10 Punkte)

7. Modifizieren Sie die Unterbrechungsbehandlungsroutine von Aufgabe 6 so, daß jedes ankommende Zeichen als zweistellige Hexadezimalzahl dargestellt wird (zuerst die höheren, dann die niederen vier Bits)¹. Geben Sie nach diesen beiden Zeichen ein Leerzeichen aus (ASCII 20H).

Hinweis: Die ASCII-Codes der Ziffern 0...9: 30H...39H, der Zeichen A...F: 41H...46H.

(10 Punkte)

8. Eine 16-Bit-Zahl in den Registern r1 und r2 (Abb. 2) ist um drei Bits nach links zu verschieben. Die frei werdenden Bitpositionen sind mit Nullen aufzufüllen.

(10 Punkte)

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

Abb. 2

9. Die Register r4 bis r6 und r16 bis r18 enthalten jeweils 24 Bits lange Binärzahlen Z1 und Z2. (Abb. 3). Schreiben Sie einen Programmablauf, der $Z1 + Z2$ berechnet und das Ergebnis in Z1 speichert.

(10 Punkte)

1 Geheimtip: Achten Sie darauf, welche Register Ihre HEX-Wandlung braucht ...

r4	Z1_0
r5	Z1_1
r6	Z1_2
r16	Z2_0
r17	Z2_1
r18	Z2_2

Abb. 3**Zusatzaufgaben:**

- Z1. Wie weisen Sie dem Register r21 den symbolischen Namen LIMIT zu? *(5 Punkte)*
- Z2. Wir beziehen uns auf Aufgabe 8. Beim Linksverschieben sind jetzt die frei werdenden Bitpositionen mit Einsen aufzufüllen. *(5 Punkte)*
- Z3. Wir beziehen uns auf Aufgabe 9. Es ist jetzt zu rechnen $Z3 = Z1 - Z2$, wobei Z3 die Register r23 bis r25 belegt. Die ursprünglichen Zahlen Z1 und Z2 sollen dabei erhalten bleiben. *(10 Punkte)*
- Z4. Welche Speichereinrichtungen hat ein Atmel-AVR-Mikrocontroller? *(5 Punkte)*

Automatisierungstechnik AU1

Klausur vom 18. 3. 2008

Aufgaben

1. Die Register r4 bis r6 und r16 bis r18 enthalten jeweils 24 Bits lange vorzeichenlose Binärzahlen Z1 und Z2 (Abb. 1). Schreiben Sie einen Programmablauf, der $Z1 + Z2$ nach den Regeln der Sättigungsarithmetik berechnet und das Ergebnis in Z1 speichert. Nach Ausführung der Rechnung müssen alle anderen Register außer r4, r5, r6 den gleichen Inhalt haben wie vorher.

(10 Punkte)

r4	Z1_0
r5	Z1_1
r6	Z1_2
r16	Z2_0
r17	Z2_1
r18	Z2_2

Abb. 1

2. Skizzieren Sie (in Form eines Schaltbildes für eine Bitposition) den Aufbau eines E-A-Ports.
(10 Punkte)
3. In den Registern r3 und r4 steht eine 16-Bit-Binärzahl INT_1 (Abb. 2). Schreiben Sie einen Programmablauf für folgende Berechnung:

$$\text{INT}_2 = \text{INT}_1 - 38\text{FAH}$$

INT_2 belegt dabei die Register r20 und r21.

(10 Punkte)

r3	INT_1_LO
r4	INT_1_HI
	...
r20	INT_2_LO
r21	INT_2_HI

Abb. 2

4. Es sind Zeichenketten auszugeben. Hierzu dient ein Unterprogramm CHAR_OUT (es ist nur aufzurufen, nicht aber selbst zu schreiben). Das auszugebende Zeichen ist im Register TEMP zu übergeben. Die Zeichenketten stehen im Programmspeicher (Flash). Geben Sie entsprechende Programmstücke für zwei Zeichenkettenformate an:

- a) Zeichenkette TEXT. Die Zeichenkette ist mit 00H abgeschlossen. Dieses Endezeichen ist nicht auszugeben.
- b) Zeichenkette CHARS. Das erste Byte der Zeichenkette gibt deren Länge an (1 = 1 Zeichen usw.). Dieses Byte ist nicht auszugeben.

TEXT und CHARS sind symbolische Adressen (Labels). Neben TEMP dürfen Sie beliebige weitere Register verwenden.

(20 Punkte)

5. An einem Mikrocontroller Atmel AVR werden einige E-A-Anschlüsse nicht genutzt (Abb. 3). Initialisieren Sie die E-A-Ports so, daß dies nicht schadet. Die Initialisierung der anderen (genutzten) Bitpositionen ist gleichgültig. Beschreiben Sie kurz, was Sie tun wollen (Programmierzweck). Es genügt die einfachere der beiden möglichen Lösungen.

(10 Punkte)

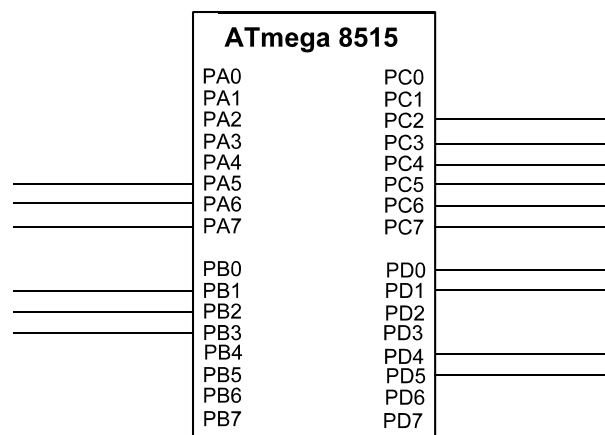


Abb. 3

6. Schreiben Sie eine Unterbrechungsbehandlungsroutine, die Zeichen in den RAM einträgt. Die Zeichen kommen von der seriellen Schnittstelle. Jedes ankommende Zeichen löst die Behandlungsroutine aus. Es befindet sich im Register UDR der seriellen Schnittstelle. Zur Ausgabe auf die Anzeige ist ein fertiges Unterprogramm DEPOSIT zu verwenden. Es erhält das darzustellende Zeichen im Register r20. Um die Speicheradressierung usw. müssen Sie sich nicht kümmern. Das Unterprogramm benötigt zudem die Register r18, r19, r24 und r25 (und zwar als Arbeitsregister, deren Inhalt bei der Rückkehr verworfen werden kann).

(10 Punkte)

7. Ein E-A-Port ist zu initialisieren. Dabei sollen einige Bits als Ausgänge eingerichtet werden und eine feste Anfangsbelegung erhalten. Was stellen Sie zuerst ein? – Das Datenmuster oder die Richtungssteuerung? Weshalb?

(10 Punkte)

8. Ein Mikrocontroller ist als Kabeltester einzusetzen. Das maximal 8adrige Kabel (vgl. die typischen Netwerkkabel) ist an die Ports C und D angeschlossen. C dient zur Ausgabe der Prüfmuster, D zum Einlesen der Signalbelegung am anderen Ende des Kabels. Als Prüfmuster soll eine Eins durch alle Bitpositionen geschoben werden. Schreiben Sie ein Programm, das alle acht Prüfmuster ausgibt und mit den ankommenden Signalbelegungen vergleicht. Zur Fehleranzeige ist ein Register ERROR zu verwenden, das am Schluß des Prüfablaufs die Anzahl der Fehler enthält (0 = kein Fehler, 1 = ein Fehler usw.). Neben ERROR dürfen Sie beliebige weitere Register verwenden.

(16 Punkte)

9. Wir beziehen uns auf Aufgabe 8. In jedem Prüfschritt ist das Prüfmuster auszugeben und die resultierende Signalbelegung einzulesen. Dürfen hierbei Ausgabe und Eingabe unmittelbar aufeinander folgen? (Kurze Diskussion des Problems.)

(10 Punkte)

Zusatzaufgaben:

- Z1. Wir beziehen uns auf Aufgabe 3. Es ist die gleiche Berechnung auszuführen. INT_2 belegt jetzt aber die Register r7 und r8. Nach Ausführung der Rechnung müssen alle anderen Register außer r7, r8 den gleichen Inhalt haben wie vorher.

(10 Punkte)

- Z2. Wozu dienen **org**-Anweisungen?

(5 Punkte)

- Z3. Eine 16-Bit-Zahl in den Registern r20 und r21 (Abb. 2) ist um zwei Bits arithmetisch nach rechts zu verschieben.

(8 Punkte)

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

Abb. 2

Automatisierungstechnik AU1

Klausur vom 23. 9. 2008

Aufgaben

Hinweis: Die meisten der folgenden Aufgaben sind durch kurze Assemblerprogrammstücke zu lösen. Programmieren Sie nicht mehr, als wirklich verlangt ist. Sollten Sie Arbeitsregister benötigen, so verwenden Sie dafür symbolische Bezeichnungen TEMP, TEMP1, TEMP2 o. ä. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

1. Zu einem 24-Bit-Wort in den Registern r1, r2, r3 ist ein Byte zu addieren, das in Register r6 steht (Abb. 1). Bei Bedarf dürfen weitere Register nach Belieben genutzt werden. Deren bisheriger Inhalt darf aber nicht verlorengehen. Es sind zwei Programme zu schreiben:
 - a) wenn das Byte in r6 eine vorzeichenlose Binärzahl enthält,
 - b) wenn das Byte in r6 eine vorzeichenbehaftete Binärzahl enthält.

(Zusammen 14 Punkte)

r1	Bits 7...0
r2	Bits 15...8
r3	Bits 23...16
r6	Bits 7...0

Abb. 1

2. Es geht um Zeichenketten, wie beispielsweise "Borussia Dortmund", "Schalke Nullvier", "Netzausfall", "Papier einlegen" usw., die im Mikrocontroller gespeichert werden sollen. Erläutern Sie kurz wenigstens zwei Möglichkeiten, die Länge einer Zeichenkette anzugeben. (10 Punkte)
3. Erläutern Sie kurz den Fachbegriff RISC. In welcher Hinsicht weichen die tatsächlich am Markt befindlichen RISC-Maschinen von den ursprünglichen akademischen Prinzipien ab? (10 Punkte)
4. Erläutern Sie wenigstens drei Möglichkeiten der Parameterübergabe beim Unterprogrammruft. (6 Punkte)
5. Wozu dienen **byte**-Anweisungen? (4 Punkte)
6. In den Registern r3 und r4 sowie r6 und r7 stehen zwei 16-Bit-Binärzahlen INT_1 und INT_2 (Abb. 2). Schreiben Sie einen Programmablauf für folgende Berechnung:

$$\text{INT_2} = \text{INT_1} + 38\text{FAH}$$

Kein anderer Registerinhalt darf dabei verlorengehen.

(12 Punkte)

r3	INT_1_LO
r4	INT_1_HI
r6	INT_2_LO
r7	INT_2_HI

Abb. 2

7. Es ist ein Programmstück zu schreiben, das die Funktion eines Testers ausführt. Im Flash befinden sich zwei gleich lange Zeichenketten: STIM und SOLL. Die Länge steht in Register r16 (Wertebereich: 1...255). Die Fehleranzahl ist in Register r17 darzustellen (0 Fehler = alles o.k.). Die E-A-Ports sind bereits initialisiert. Ablauf:

1. Ein Zeichen von STIM auf Port C ausgeben.
2. Die Belegung von Port D zurücklesen und mit dem zugehörigen Zeichen von SOLL vergleichen.
3. Im Fehlerfall ist der Fehlerzähler (r17) um Eins zu erhöhen.
4. Die Schritte 1 bis 3 so oft wiederholen, bis alle Zeichen abgearbeitet sind.

Hierbei dürfen beliebige weitere Register belegt werden. *Hinweis:* Achten Sie bei den Ein- und Ausgaben auf die jeweiligen Spitzfindigkeiten (ggf. kurz kommentieren).

(20 Punkte)

8. Eine 16-Bit-Zahl in den Registern r20 und r21 (Abb. 3) ist um drei Bits nach links zu verschieben. Die frei gewordenene Bitpositionen sind mit Einsen aufzufüllen.

(10 Punkte)

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

Abb. 3

9. Schreiben Sie eine Unterbrechungsbehandlungsroutine, die Zeichen in den RAM einträgt. Die Zeichen kommen von der seriellen Schnittstelle. Jedes ankommende Zeichen löst die Behandlungsroutine aus. Es befindet sich im Register UDR der seriellen Schnittstelle. Zur Ausgabe auf die Anzeige ist ein fertiges Unterprogramm DEPOSIT zu verwenden. Es erhält das darzustellende Zeichen im Register r20. Um die Speicheradressierung usw. müssen Sie sich nicht kümmern. Das Unterprogramm benötigt zudem die Register r18, r19, r24 und r25 (und zwar als Arbeitsregister, deren Inhalt bei der Rückkehr verworfen werden kann).

(14 Punkte)

Zusatzaufgaben

- Z1. Geben Sie an, wie Sie eine (feste) Zeichenkette in den Quelltext eines Assemblerprogramms eintragen. Textbeispiel: "Institut fuer Interruptforschung".

(4 Punkte)

- Z2. Es folgt ein Ausschnitt aus einem C-Programm. Skizzieren Sie die Belegung des Stack Frame unmittelbar vor dem Eintritt in den eigentlichen Funktionskörper. Geben Sie an, worauf Frame Pointer und Stackpointer zeigen. Zugriffsbreite des Stacks: 16 Bits. Datentyp int = 16 Bits.

(10 Punkte)

Deklaration einer Funktion:

```
void FRAME_UPDATE (int HOR, int VERT, int PIXEL);
{
int A, B, C;
...

return ();
}
```

Jetzt wird die Funktion aufgerufen:

...

```
FRAME_UPDATE (X, Y, GREEN);
```

...

Darstellung: wie im Script (niedere Adressen oben, höhere unten; Stack wächst in Richtung niederer Adressen).

- Z3. Um Parameter P1...Pn an ein Unterprogramm zu übergeben, soll wie folgt programmiert werden:

```
PUSH P1
PUSH P2
... usw.
PUSH Pn
CALL UP
```

- a) Welche Voraussetzungen müssen seitens des Prozessors erfüllt sein, um so programmieren zu können? (Kurze Erläuterung.)
- b) Wenn diese Voraussetzungen nicht gegeben sind: wie behelfen Sie sich? – Schlagen Sie eine alternative Variante der Parameterübergabe vor. Sie sollte auch dann funktionieren, wenn n ziemlich groß ist (Anhaltswert: > 8).

(10 Punkte)

Automatisierungstechnik AU1

Klausur vom 17. 3. 2009

Aufgaben

Hinweis: Die meisten der folgenden Aufgaben sind durch kurze Assemblerprogrammstücke zu lösen. Programmieren Sie nicht mehr, als wirklich verlangt ist. Sollten Sie Arbeitsregister benötigen, so verwenden Sie dafür symbolische Bezeichnungen TEMP, TEMP1, TEMP2 o. ä. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

1. Die Register r16, r17 und r20, r21 enthalten jeweils 16 Bits lange vorzeichenlose Binärzahlen Z1 und Z2 (Abb. 1). Schreiben Sie einen Programmablauf, der Z1 – Z2 nach den Regeln der Sättigungsarithmetik berechnet und das Ergebnis in Z1 speichert. Nach Ausführung der Rechnung müssen alle anderen Register außer r16, r17 den gleichen Inhalt haben wie vorher.

(10 Punkte)

r16	Z1_0
r17	Z1_1
r20	Z2_0
r21	Z2_1

Abb. 1

2. Schreiben Sie ein Programmstück, das den Inhalt des Registers TEMP in zwei ASCII-Zeichen wandelt, die das Bitmuster in hexadezimaler Form darstellen. Das Zeichen der niederwertigen Tetrade soll im Register LO_HEX abgelegt werden, das der höherwertigen Tetrade im Register HI_HEX (alle Registeradressen > 15). Am Schluß des Programms müssen alle anderen Register (einschließlich TEMP) sowie die Flagbits den gleichen Inhalt haben wie zu Beginn. *Hinweis:* Die ASCII-Codes der Ziffern 0...9: 30H...39H, der Zeichen A...F: 41H...46H.

(15 Punkte)

3. Erläutern Sie kurz (Aufzählung), aus welchen Teilen eine typische Anweisungszeile in einem Assemblerprogramm besteht.

(5 Punkte)

4. 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.

(15 Punkte)

5. Wir beziehen uns auf die Aufgabe 4. Die Zeichenketten haben jetzt aber das Format der Programmiersprache Pascal (Längenangabe am Anfang):

```
TEXT_1:    .db 17, "Temperaturwarnung"  
TEXT_2:    .db 19, "Kühlwasserkreislauf"
```

Schreiben Sie ein Programmstück, das die Zeichenkette TEXT_1 anzeigt. Zur eigentlichen Ausgabe ist wie in Aufgabe 4 das Unterprogramm BYTE_OUT zu verwenden. Es dürfen beliebige weitere Register verwendet werden.

(15 Punkte)

6. Es folgt ein Ausschnitt aus einem C-Programm. Skizzieren Sie die Belegung des Stack Frame unmittelbar vor dem Eintritt in den eigentlichen Funktionskörper. Geben Sie an, worauf Frame Pointer und Stackpointer zeigen. Zugriffsbreite des Stacks: 32 Bits. Datentyp int = 32 Bits.

(10 Punkte)

Deklaration einer Funktion:

```
void PAINT (int HOR, int VERT, int COLOR);  
{  
int U, V W, X, Y, Z;  
...  
  
return ();  
}
```

Jetzt wird die Funktion aufgerufen:

```
...  
  
PAINT (A, B, GREEN);  
...
```

Darstellung: wie im Skript (niedere Adressen oben, höhere unten; Stack wächst in Richtung niederer Adressen).

7. Eine 24-Bit-Binärzahl ist in den Registern r16 bis r18 untergebracht. Schreiben Sie ein Programmstück, das eine arithmetische Rechtsverschiebung um drei Bits ausführt.

(10 Punkte)

8. Es ist ein Programmstück zu schreiben, das die Funktion eines Testers ausführt. Die Sollwerte stehen in den Registern r16, r17 und r18 Die Fehleranzahl ist in Register r20 darzustellen (0 Fehler = alles o.k.). Die E-A-Ports sind bereits initialisiert. Abauf:

1. Einen Sollwert auf Port C ausgeben.
2. Die Belegung von Port D zurücklesen und mit dem Sollwert vergleichen. Sie ist dann o.k. wenn der gelesene Wert das invertierte Bitmuster des Sollwerts aufweist.
3. Im Fehlerfall ist der Fehlerzähler (r20) um Eins zu erhöhen.
4. Die Schritte 1 bis 3 so oft wiederholen, bis alle Sollwerte abgearbeitet sind.

Hierbei dürfen beliebige weitere Register belegt werden. Erläutern Sie kurz, worauf Sie bei der Aus- und Eingabe achten müssen.

(15 Punkte)

Zusatzaufgaben:

- Z1. Wie weisen Sie dem Register r12 den symbolischen Namen XMIT zu? (5 Punkte)
- Z2. Erläutern Sie kurz den Fachbegriff CISC. Nennen Sie wenigstens ein Beispiel einer CISC-Maschine. (5 Punkte)
- Z3. Der E-A-Port C hat folgende symbolische Adressen: DDRC, PORTC, PINC. 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?
- (4 Punkte)
- Z4. Erläutern Sie wenigstens drei Möglichkeiten der Parameterübergabe beim Unterprogrammruf. (6 Punkte)
- Z5. Eine Interruptserviceroutine (ISR) beginnt mit den folgenden Befehlen. Geben Sie eine dazu passende Befehlsfolge an (im Anschluß an die Marke LEAVE), mit der die ISR verlassen werden kann. (10 Punkte)

```
PUSH    TEMP
IN      TEMP, SREG
PUSH    TEMP
PUSH    r24
PUSH    r25
PUSH    r3
PUSH    r4
```

...

LEAVE: