Mikrocontrollertechnik MC
Alte Klausuren 2010 bis 2012 zum Üben

12. 6. 2012

Automatisierungstechnik AU1

Klausur vom 16. 3. 2010

Hinweis: Die meisten der folgenden Aufgaben sind durch kurze Assemblerprogrammstücke zu lösen. Programmieren Sie nicht mehr, als wirklich verlangt ist. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

1. 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). Sie dürfen beliebig viele weitere Register verwenden. *Hinweis:* Die ASCII-Codes der Ziffern 0...9: 30H...39H, der Zeichen A...F: 41H...46H.

   (15 Punkte)

2. Erläutern Sie wenigstens drei Möglichkeiten der Parameterübergabe beim Unterprogrammruf.

   (6 Punkte)

3. Zu einem 24-Bit-Wort in den Registern r1, r2, r3 (Abb. 1) ist der Festwert 31H zu addieren Bei Bedarf dürfen weitere Register nach Belieben genutzt werden. Deren bisheriger Inhalt darf aber nicht verlorengehen. Mit welchem Befehl Sie den Festwert 31H in die Rechnung einbringen, ist gleichgültig.

   (10 Punkte)

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

Abb. 1

4. 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)

5. Im Programmspeicher steht die folgende Zeichenkette.

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

   Schreiben Sie ein Programmstück, das ermittelt, wie lang die Zeichenkette ist (ohne das abschließende Nullzeichen). Der Längenwert soll im Register r16 stehen. Es dürfen beliebig viele weitere Register belegt werden.

   (10 Punkte)

6. 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)

7. Wodurch unterscheidet sich das arithmetische vom gewöhnlichen Rechtsschieben? (Kurze Erläuterung.) Tragen Sie in Tabelle 1 ein, welche Werte sich ergeben, wenn die angegebenen Bytes um jeweils zwei Bits nach rechts verschoben werden. Tragen Sie in die Tabelle Befehlsfolgen ein, die die jeweilige Verschiebung bewirken. Da zu verschiebende Byte soll sich in Register r16 befinden.

(16 Punkte)

Byte	Gewöhnliche Verschiebung	Arithmetische Verschiebung
35H		
95H		
Befehlsfolgen		

Tabelle 1

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

(10 Punkte)

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

Abb. 2

9. Erläutern Sie wenigstens drei Möglichkeiten der Parameterübergabe beim Unterprogrammruf.

(6 Punkte)

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

(5 Punkte)

Z2. 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)

Z3. An einem Mikrocontroller Atmel AVR werden einige E-A-Anschlüsse nicht genutzt (Abb. 3).

a) Wie sind die Ports nach dem Rücksetzen eingestellt?

b) Was geschieht, wenn man nichts tut?

c) Wie kann man die E-A-Ports so initialisieren, dass es nicht schadet? (Geben Sie alle Möglichkeiten an, die Ihnen einfallen.)

Die Initialisierung der anderen (genutzten) Bitpositionen ist gleichgültig. Hier ist nichts zu programmieren, sondern nur zu beschreiben, was getan werden soll (Programmierabsicht).

(10 Punkte)

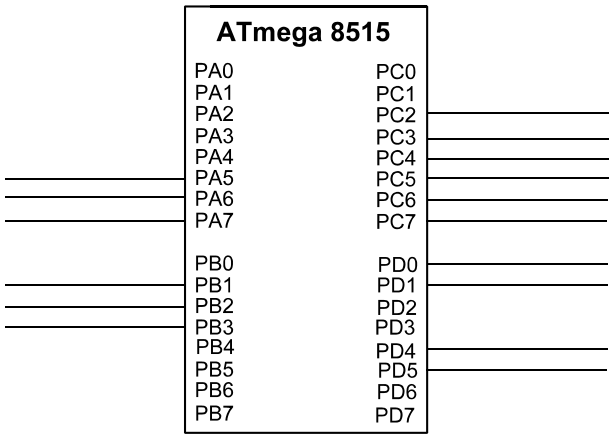

Abb. 3

Automatisierungstechnik AU1

Klausur vom 21. 9. 2010

Hinweis: Die meisten der folgenden Aufgaben sind durch kurze Assemblerprogrammstücke zu lösen. Programmieren Sie nicht mehr, als wirklich verlangt ist. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

1. Die Register r6, r7 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 Z2 speichert. Nach Ausführung der Rechnung müssen alle anderen Register außer r20, r21 den gleichen Inhalt haben wie vorher.

(10 Punkte)

r6	Z1_0
r7	Z1_1
r20	Z2_0
r21	Z2_1

Abb. 1

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

(5 Punkte)

3. Zu einem 24-Bit-Wort in den Registern r1, r2, r3 (Abb. 2) ist der Festwert 1F53H zu addieren Bei Bedarf dürfen weitere Register nach Belieben genutzt werden. Deren bisheriger Inhalt darf aber nicht verlorengehen.

(10 Punkte)

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

Abb. 2

4. Erläutern Sie kurz, wodurch sich das arithmetische vom gewöhnlichen Rechtsschieben unterscheidet. Welche Befehle werden für beide Arten des Verschiebens verwendet?

(10 Punkte)

5. 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.

(10 Punkte)

6. Abbildung 3 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.)

(15 Punkte)

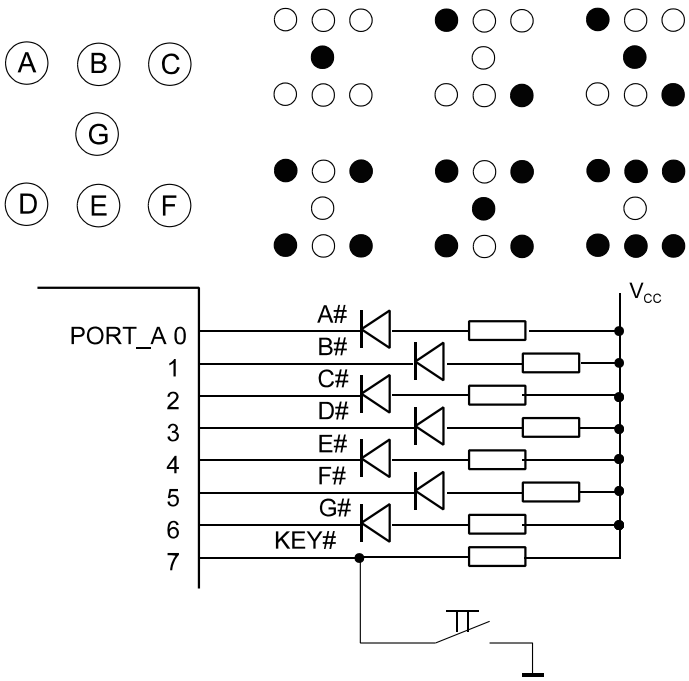

Abb. 3

7. 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 DPY zu verwenden. Es erhält das darzustellende Zeichen im Register r16. Um die Speicheradressierung usw. müssen Sie sich nicht kümmern. Das Unterprogramm benötigt zudem die Register r17, r18 und r19. Es sind Arbeitsregister, deren Inhalt bei der Rückkehr verworfen werden kann.

(10 Punkte)

8. 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.

(10 Punkte)

9. Im Programmspeicher steht die folgende Zeichenkette.

```
TEXT:        .db "Druckausgabe",0
```

Schreiben Sie ein Programmstück, das ermittelt, wie lang die Zeichenkette ist (ohne das abschließende Nullzeichen). Der Längenwert soll im Register r16 stehen. Es dürfen beliebig viele weitere Register genutzt werden.

(10 Punkte)

Zusatzaufgaben

Z1. Gegeben ist eine Stackmaschine mit folgender Befehlsliste: PUSH *variable*, ADD, SUB, MUL, DIV. Schreiben Sie ein Programm, das den Ausdruck ((A + B) * (C + D)) : (X – Y) berechnet. Das Ergebnis soll auf dem Stack hinterlassen werden (TOS).

(10 Punkte)

Z2. 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)

Z3. Beim Aufruf eines Unterprogramms werden Register in den Stack gerettet. Das Unterprogranm 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.

(10 Punkte)

```
        PUSH    TEMP
        IN      TEMP, SREG
        PUSH    TEMP
        PUSH    r20
        PUSH    r21
        PUSH    r5
        PUSH    r3
          ...
LEAVE:
```

Automatisierungstechnik AU1

Klausur vom 8. 3. 2011

Hinweis: Die meisten der folgenden Aufgaben sind durch kurze Assemblerprogrammstücke zu lösen. Programmieren Sie nicht mehr, als wirklich verlangt ist. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

1. Schreiben Sie ein Assemblerprogrammstück, das folgenden Ablauf implementiert (Abb. 1).
(10 Punkte)

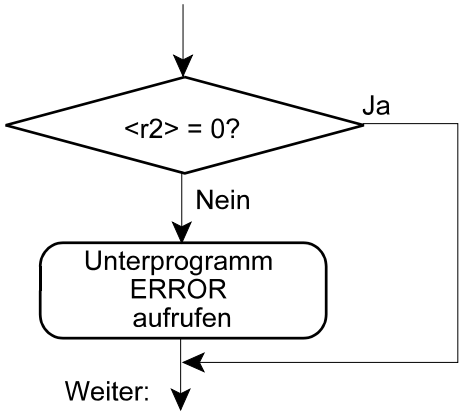

Abb. 1

2. Erläutern Sie wenigstens drei Möglichkeiten der Parameterübergabe beim Unterprogrammruf.
(6 Punkte)

3. Erläutern Sie kurz (Beschreibung, Skizze) den Begriff Stack Frame.
(10 Punkte)

4. Schreiben Sie eine Unterbrechungsbehandlungsroutine, die Zeichen ausgibt. Die Zeichen kommen von der seriellen Schnittstelle. Das jeweils empfangene Zeichen befindet sich im Register UDR. Zur Ausgabe auf die Anzeige ist ein fertiges Unterprogramm EMIT zu verwenden. Das darzustellende Zeichen wird im Register r20 übergeben. Das Unterprogramm benötigt zudem die Register r0, r1, r24 und r25. Es sind Arbeitsregister, deren Inhalt bei der Rückkehr verworfen werden kann.
(10 Punkte)

5. In den Registern r20 bis r22 steht eine 24 Bits lange vorzeichenlose Binärzahl. Schreiben Sie ein Programm, das diese Zahl mit Drei multipliziert. Sie dürfen hierzu beliebig viele weitere Register benutzen.
(10 Punkte)

6. Die Ports der Mikrocontroller werden nicht immer vollständig ausgenutzt. Worauf ist in diesem Fall zu achten? Erläutern Sie kurz wenigstens zwei Möglichkeiten, mit ungenutzten Bitpositionen korrekt umzugehen.
(10 Punkte)

7. Ein Mikrocontroller ist als Kabeltester einzusetzen (Abb. 2). Das maximal 8adrige Kabel (vgl. die typischen Netwerkkabel) ist an die Ports A und B angeschlossen. B dient zum Ausgeben des Prüfmusters, A zum Einlesen der Signalbelegung am anderen Ende des Kabels. An Port C sind Leuchtdioden angeschlossen, die zur Anzeige dienen. Sie wirken aktiv Low (vgl. Atmel-Starterkit).

Als Prüfmuster soll eine Eins durch alle Bitpositionen geschoben werden. Die eingelesene Signalbelegung soll auf die LEDs ausgegeben werden. Das Kabel ist dann fehlerfrei, wenn die gelesene Belegung dem Prüfmuster entspricht. Gibt es keinen Fehler, soll das Schieben, Vergleichen und Anzeigen endlos umlaufen. Dabei soll die LED an Port D grün leuchten. Wird ein Fehler erkannt, so soll der Ablauf anhalten, und die eingelesene Signalbelegung soll auf den LEDs an Port C dargestellt werden. Zudem soll die LED an Port D rot blinken. Schreiben Sie ein Programm, das alle erforderlichen Funktionen realisiert.

Hinweise:
- Denken Sie auch an die erforderlichen Initialisierungen.
- Zum Blinken steht ein fertiges Unterprogramm BLINK zur Verfügung.
- Die restlichen Bitpositionen des Ports D werden nicht genutzt.
- Zur seriellen Schnittstelle siehe Zusatzaufgabe Z3.

(25 Punkte)

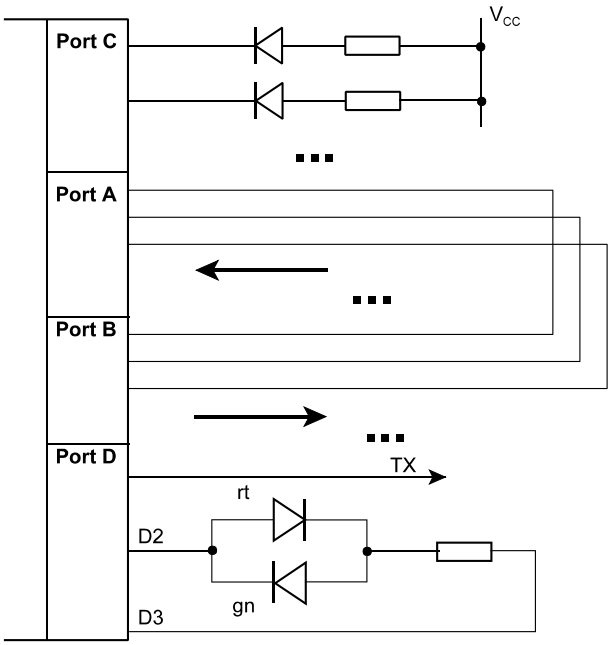

Abb. 2

8. Von einem 24-Bit-Wort in den Registern r1, r2, r3 (Abb. 3) ist der Festwert 55H zu subtrahieren. Bei Bedarf dürfen weitere Register nach Belieben genutzt werden. Deren bisheriger Inhalt darf aber nicht verlorengehen.

(10 Punkte)

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

Abb. 3

9. Gegeben ist eine Stackmaschine mit folgender Befehlsliste: PUSH *variable*, ADD, SUB, MUL, DIV. Schreiben Sie ein Programm, das den Ausdruck ((A * B) + (C – D)) : (E * F) berechnet. Das Ergebnis soll auf dem Stack hinterlassen werden (TOS).

(10 Punkte)

Zusatzaufgaben

Z1. Erläutern Sie kurz den Fachbegriff CISC. Nennen Sie wenigstens ein Beispiel einer CISC-Maschine.
(5 Punkte)

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

(5 Punkte)

Z3. Wir gehen nochmals zu Aufgabe 7 und Abbildung 2. Im Fehlerfall soll die Nummer des aktuellen Prüfschritts (0...7) als Hexadezimalzahl über die serielle Schnittstelle gesendet werden. Hierfür steht ein Unterprogramm XMIT zur Verfügung, dem das zu übertragende Zeichen in Register r16 übergeben wird. Um die Initialisierung der Schnittstelle müssen Sie sich nicht kümmern.
(10 Punkte)

Automatisierungstechnik AU1/ Hard- und Software-Engineering HS1

Klausur vom 28. 9. 2011

Hinweis: Die meisten der folgenden Aufgaben sind durch kurze Assemblerprogrammstücke zu lösen. Programmieren Sie nicht mehr, als wirklich verlangt ist. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

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

   (5 Punkte)

2. Erläutern Sie kurz die Fachbegriffe CISC und RISC. Gehen Sie dabei vor allem auf die typischen Unterschiede zwischen CISC- und RISC-Maschinen ein.

   (10 Punkte)

3. Erläutern Sie kurz (Beschreibung, Skizze) den Begriff Stack Frame.

   (10 Punkte)

4. Eine 16-Bit-Zahl in den Registern r22 und r23 (Abb. 1) ist um drei Bits nach links zu verschieben. Die frei gewordenene Bitpositionen sind mit Einsen aufzufüllen.

   (10 Punkte)

r22	Bits 7...0
r23	Bits 15...8

Abb. 1

5. Wir bleiben bei Abb. 1. Die Zahl in den Registern ist jetzt um 2 Bits arithmetisch nach rechts zu verschieben.

   (10 Punkte)

6. Die Ports der Mikrocontroller werden nicht immer vollständig ausgenutzt. Worauf ist in diesem Fall zu achten? Erläutern Sie kurz wenigstens zwei Möglichkeiten, mit ungenutzten Bitpositionen korrekt umzugehen.

   (10 Punkte)

7. Schreiben Sie eine Unterbrechungsbehandlungsroutine, die Zeichen zu einer LCD-Anzeige überträgt. 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 DISPLAY zu verwenden. Es erhält das darzustellende Zeichen im Register r16. Um die Einzelheiten müssen Sie sich nicht kümmern. Das Unterprogramm benötigt zudem die Register r17, r24 und r25 (und zwar als Arbeitsregister, deren Inhalt bei der Rückkehr verworfen werden kann).

   (10 Punkte)

8. Abb. 2 veranschaulicht eine Anwendungsbeschaltung des E-A-Ports D. Der E-A-Port hat folgende symbolische Adressen: DDRD, PORTD, PIND.

a) An Bit D7 hängt eine Leistungsstufe, die aktiv Low angesteuert wird. Wird Bit D7 Null, läuft die angeschlossene Kreissäge los... Was initialisieren Sie zuerst – das Richtungssteuerregister oder das Datenregister?

b) Initialisieren Sie Port D so, daß die Kreissäge nicht anläuft, daß LED3 rot leuchtet und daß der Lautsprecher mit einem High-Pegel angesteuert wird.

c) Schreiben Sie ein Programm, das die LED 1 blinken läßt, solange Taste Ta 1 gedrückt wird. Zum Blinken steht ein Unterprogramm BLNK zur Verfügung (muß nur aufgerufen werden).

(15 Punkte)

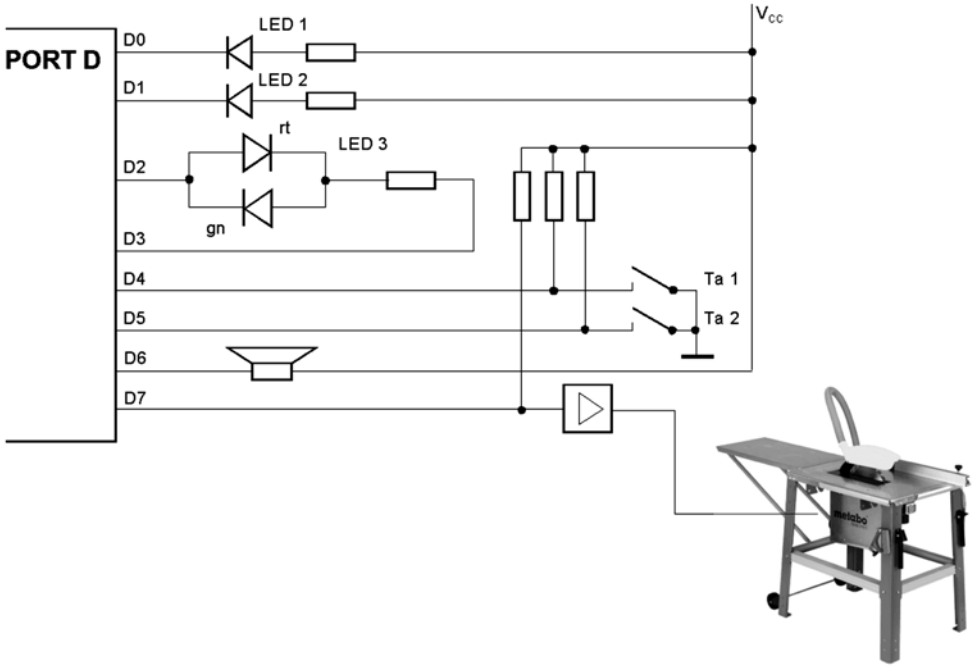

Abb. 2

9. In den Registern r3 und r4 steht eine 16-Bit-Binärzahl INT_A (Abb. 3). Schreiben Sie einen Programmablauf für folgende Berechnung:

$$INT\_B = INT\_A + 5CH$$

INT_B belegt dabei die Register r22 und r23.

(10 Punkte)

r3	INT_A_LO
r4	INT_A_HI
	...
r22	INT_B_LO
r23	INT_B_HI

Abb. 3

10. Erläutern Sie wenigstens drei Möglichkeiten der Parameterübergabe beim Unterprogrammruf.

(6 Punkte)

Zusatzaufgaben

Z1. In der Werbung wird gern herausgestellt, daß eine bestimmte Maschine (wie beispielsweise auch Atmel AVR) besonders viele Register aufweist. Diskutieren Sie kurz die typischen Vor- und Nachteile dieser Auslegung.

(10 Punkte)

Z2. Wir beziehen uns nochmals auf Abb. 2. Schreiben Sie einen Programmablauf, der folgendermaßen wirkt:

a) Wenn Ta 1 betätigt wird, soll die Kreissäge laufen – und zwar dauernd (also auch, nachdem Ta 1 wieder losgelassen wurde). LED 3 soll dabei grün leuchten.

b) Wenn Ta 2 betätigt wird, soll die Kreissäge anhalten (und zwar auch dann, wenn Ta 1 noch gedrückt ist). LED 3 soll dann rot leuchten. Achtung: Wird währenddessen Ta 1 immer noch niedergehalten, darf die Kreissäge keineswegs wieder anlaufen (Arbeitsschutz). Es ist also folgendes Verhalten zu gewährleisten: Ta 2 wird betätigt – Kreissäge geht aus. Wenn jetzt Ta 1 noch betätigt sein sollte, passiert nichts. Erst nachdem Ta 1 losgelassen ist, geht es wieder zu Punkt a.

Das Prellen der Tasten muß nicht berücksichtig werden.

(15 Punkte)

Z3. 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 PRINT_INIT (int DVC_ADRS, int DVC_TYPE, int BUFF_ADRS, int BUFF_LEN);
{
int V, W;
...

return ();
}
```

Jetzt wird die Funktion aufgerufen:

```
...

PRINT_INIT (X, ALPHA, Y, Z);
...
```

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

Z4. In den Registern r20 bis r22 steht eine 24 Bits lange vorzeichenlose Binärzahl. Schreiben Sie ein Programm, das diese Zahl mit Vier multipliziert. Sie dürfen hierzu beliebig viele weitere Register benutzen.

(10 Punkte)

Automatisierungstechnik AU1/ Hard- und Software-Engineering HS1/ Mikrocontrollertechnik MC

Klausur vom 13. 3. 2012

Hinweis: Die meisten der folgenden Aufgaben sind durch kurze Assemblerprogrammstücke zu lösen. Programmieren Sie nicht mehr, als wirklich verlangt ist. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

1. Eine 16-Bit-Zahl in den Registern r22 und r23 (Abb. 1) ist um drei Bits nach links zu verschieben.
(10 Punkte)

r22	Bits 7...0
r23	Bits 15...8

Abb. 1

2. Eine 16-Bit-Zahl in den Registern r22 und r23 (vgl. Abb. 1) ist um drei Bits nach links zu rotieren.
(10 Punkte)

   Hinweis: Überlegen Sie sich bitte, was Rotieren eigentlich bedeutet. Sie dürfen weitere Register nach Belieben verwenden. Deren ursprünglicher Inhalt darf aber nicht verlorengehen.

3. Erläutern Sie kurz (Beschreibung, Skizze) den Begriff Stack Frame.
(10 Punkte)

4. Erläutern Sie kurz den Unterschied zwischen dem gewöhnlichen (logischen) und dem arithmetischen Rechtsverschieben. Welche Befehle verwendet man dafür (nur nennen, nicht näher beschreiben). Was kommt heraus, wenn man die folgenden Werte logisch und arithmetisch um 1 Bit nach rechts verschiebt? (Lösung hexadezimal in die Tabelle eintragen.)
(10 Punkte)

Anfangswert	Logische Verschiebung	Arithmetische Verschiebung
3BH		
B3H		

5. Ein Mikrocontroller soll eine Zweifarben-LED (Abb. 2) ansteuern. Anschluß: Pin 1 an Port C, Bit 0, Pin 2 an Port C, Bit 1 (natürlich über Serienwiderstand...). Der Port ist schon richtg initialisiert.

   a) Als Vorübung: 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) Schreiben Sie ein Programmstück, das die LED ansteuert. Die Art der Anzeige ergibt sich aus den Signalen an Port A1 und A0:

- A1 = 0, A0 = 0: LED aus.
- A1 = 0, A0 = 1: rot.
- A1 = 1, A0 = 0: grün.
- A1 = 1, A0 = 1: Blinken rot – grün – rot usw.

Wenn sich die Belegung der Eingänge ändert, soll die LED entsprechend umschalten. Dabei soll ggf. die jeweilig Blinkphase noch zu Ende kommen.

c) Schreiben Sie ein Unterprogramm LED_BLNK, das die LED blinken läßt. Parameter:

- Register r20: Anzahl der Blinkzyklen (0 bis 255). 0 = gar nicht blinken (sofort zurück).
- Register r21: Farbe. 0 = gar nicht (nur Zeit verbrauchen), 1 = Blinken grün – aus, 2 = Blinken rot – aus; 3 = Blinken Rot – Grün.

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 300 ms ein.

(15 Punkte)

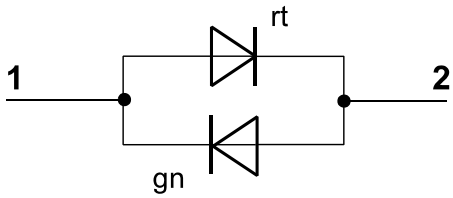

Abb. 2

6. Beim Aufruf eines Unterprogramms werden Register in den Stack gerettet. Das Unterprogranm 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.

(10 Punkte)

```
PUSH    TEMP
IN      TEMP, SREG
PUSH    TEMP
PUSH    r20
PUSH    r21
PUSH    r7
PUSH    r8
        . . .
LEAVE:
```

7. Die Ports der Mikrocontroller werden nicht immer vollständig ausgenutzt. Worauf ist in diesem Fall zu achten? Erläutern Sie kurz wenigstens zwei Möglichkeiten, mit ungenutzten Bitpositionen korrekt umzugehen.

(10 Punkte)

8. Erläutern Sie kurz die Fachbegriffe v.Neumann-Architektur und Harvard-Architektur. Zu welcher Art gehört Atmel AVR?

(10 Punkte)

9. In den Registern r20 und r21 steht eine 16 Bits lange vorzeichenlose Binärzahl. Schreiben Sie ein Programm, das diese Zahl mit Vier multipliziert. Sie dürfen hierzu beliebig viele weitere Register benutzen.

(10 Punkte)

10. Abb. 3 zeigt eine Anwendungsbeschaltung des E-A-Ports D. Der E-A-Port hat folgende symbolische Adressen: DDRD, PORTD, PIND. Wird Bit D7 Null, läuft die angeschlossene Kreissäge los. Initialisieren Sie Port D so, daß die Kreissäge nicht anläuft, daß LED 1 leuchtet, LED 3 grün leuchtet und der Lautsprecher mit einem High-Pegel angesteuert wird.

(8 Punkte)

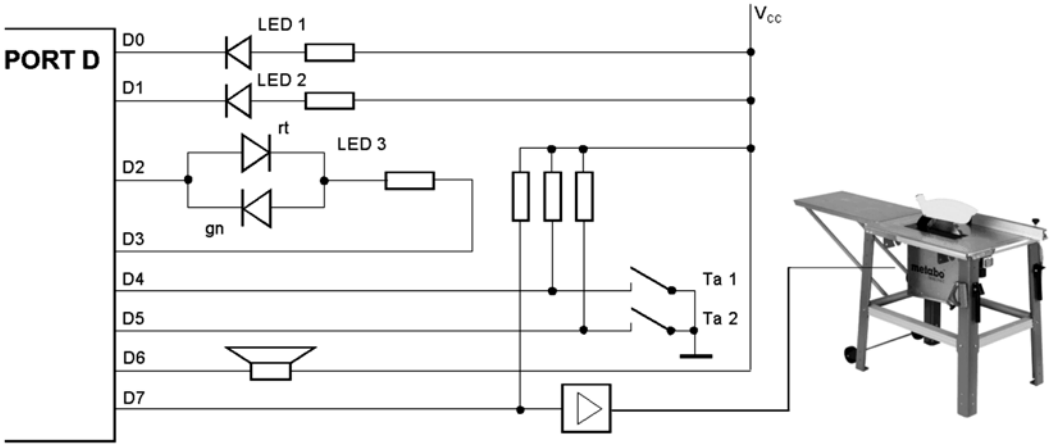

Abb. 3

Zusatzaufgaben

Z1. In der Werbung wird gern herausgestellt, daß eine bestimmte Maschine (wie beispielsweise auch Atmel AVR) besonders viele Register aufweist. Diskutieren Sie kurz die typischen Vor- und Nachteile dieser Auslegung.

(10 Punkte)

Z2. Gegeben ist eine Stackmaschine mit folgender Befehlsliste: PUSH *variable*, ADD, SUB, MUL, DIV. Schreiben Sie ein Programm, das den Ausdruck (A * (X – Y)): (C + D) berechnet. Das Ergebnis soll auf dem Stack hinterlassen werden (TOS).

(10 Punkte)

Z3. Wir gehen nochmals zu Aufgabe 6. Wie müßten Sie Ihre Lösung abändern, wenn es sich nicht um ein Unterprogramm handelt, sondern um eine Interruptserviceroutine (ISR)?

(5 Punkte)

Befehl	Wirkung
LDI *rd, imm*	Lädt einen Direktwert in ein Register. Beispiel: *LDI r17, 0x22*
MOV *rd, rs*	Transportiert Inhalt des Registers rs in das Register rd. Beispiel: *MOV r2, r22*
LPM	Lädt Byte aus Programmspeicher in Register r0. Adresse in Register Z (r31, r30). Eweiterungen: LPM rd, Z sowie LPM rd, Z+ (Adreßerhöhung). Beispiel: *LPM r20, Z+*
LDS *rd, addrs*	Lädt ein Byte aus dem SRAM in das Register rd. Beispiel: *LDS r2, BYTE_ADRS*
STS *addrs, rs*	Transportiert den Inhalt des Registers rs in den SRAM. Beispiel: *STS BYTE_ADRS, r2*
OUT *port, rs*	Ausgabe eines Registerinhaltes. Beispiel: *OUT portd, r17*
IN *rd, port*	Eingabe in ein Register. Beispiel: *IN r17, pind*
SBI *port, bit*	Setzen Bit in E-A-Port. Beispiel: *SBI porta, 3*
CBI *port, bit*	Löschen Bit in E-A-Port. Beispiel: *CBI porta, 3*
SBIC *port, bit*	Bit in E-A-Port abfragen. Folgebefehl überspringen, wenn = 0. Beispiel: *SBIC porta, 3*
SBIS *port, bit*	Bit in E-A-Port abfragen. Folgebefehl überspringen, wenn = 1. Beispiel: *SBIS porta, 3*
NOP	keine Wirkung (Leerbefehl). Nur Verbrauchen einer Taktperiode.
JMP *label*	Unbedinge Verzweigung. Beispiel: *JMP anfang*
BRNE *label*	Verzweigen, wenn Ergebnis ungleich Null. Beispiel: *BRNE anfang*
BREQ *label*	Verzweigen, wenn Ergebnis gleich Null. Beispiel: *BREQ ende*
CALL *label*	Unterprogrammruf. Beispiel: *CALL warten*
RET	Rückkehr aus Unterprogramm
RETI	Rückkehr aus Unterbrechungsbehandlung
PUSH *rs*	Register rs in Stack retten. Beispiel: *PUSH r20*
POP *rd*	Register rd aus Stack laden. Beispiel: *POP r20*
AND *rd, rs*	Register konjunktiv verknüpfen <rd> := <rd> AND <rs>. Beispiel: *AND r12, r3*
ANDI *rd, imm*	konjunktive Verknüpfung mit Direktwert. Beispiel: *ANDI r12*, 0b11110111
OR *rd, rs*	Register disjunktiv verknüpfen <rd> := <rd> OR <rs>. Beispiel: *AND r12, r3*
ORI *rd, imm*	disjunktive Verknüpfung mit Direktwert. Beispiel: *ORI r12*, 0b11110111
COM *rd*	Bitweise Negation (Einerkomplement). Beispiel: *COM r17*
ADD *rd, rs*	Registerinhalte zueinander addieren. <rd> := <rd> + <rs>. Beispiel: *ADD r12, r4*
ADC *rd, rs*	Addieren mit Eingangsübertrag. <rd> := <rd> + <rs>. + CF. Beispiel: *ADC r13, r5*
ADIW *rd, imm*	Addieren Direktwert zu Wort (Register 24, 26, 28, 30). Beispiel: *ADIW r24, 12*
SUBI *rd, imm*	Subtrahieren Direktwert. Beispiel: *SUBI r16, 0x04*. Addieren durch Subtrahieren des negativen Wertes. Statt ADDI rd,imm (gibt es nicht) also SUBI rd,–imm
SUB *rd, rs*	Registerinhalte voneinander subtrahieren. <rd> := <rd> – <rs>. Beispiel: *SUB r12, r4*
SBC *rd, rs*	Subtrahieren mit Eingangsübertrag. <rd> := <rd> – <rs> – CF. Beispiel: *SBC r13, r5*
SBIW *rd, imm*	Subtrahieren Direktwert von Wort (Register 24, 26, 28, 30). Beispiel*: SBIW r26, 10*
CP *rd, rs*	Registerinhalte vergleichen (Subtraktion <rd> – <rs>). Beispiel: *CP r12, r22*
CPI *rd, imm*	Vergleichen mit Direktwert (Subtraktion <rd> – imm). Beispiel: *CPI r16, 0x33*
ROL *rd*	Linksrotieren über CF (CF => Bit 0, Bit 7 => CF): Beispiel: *ROL r17*
ROR *rd*	Rechtsrotieren über CF (CF => Bit 7, Bit 0 => CF). Beispiel: *ROR r17*
LSL *rd*	Linksverschieben. Bit 7 nach CF, Bit 0 wird mit 0 gefüllt. Beispiel: *LSL r17*
LSR *rd*	Rechtsverschieben. Bit 0 nach CF, Bit 7 wird mit 0 gefüllt. Beispiel: *LSR r17*

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