

**Name:**

FH Dortmund

**Matr.-Nr.:**

FB Informations- und Elektrotechnik

## Automatisierungstechnik AP1

Klausur vom 29. 1. 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, z. B. TEMP, TEMP1, TEMP2 o. ä. Deuten Sie durch Kommentare an, wozu die einzelnen Befehle dienen.

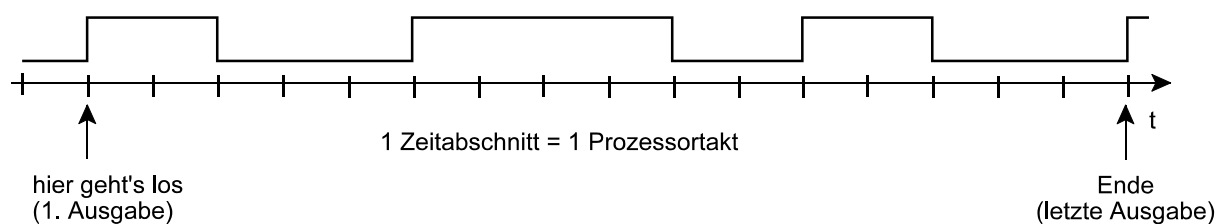
- Der E-A-Port A hat folgende symbolische Adressen: DDRA, PINA, PORTA. Welche Adresse verwenden Sie
  - um eine an ein E-A-Pin angeschlossene LED ein- oder auszuschalten?
  - um abzufragen, ob eine an den Port angeschlossene Taste betätigt wurde?

(4 Punkte)
- Erläutern Sie kurz den wesentlichen Unterschied zwischen der v. Neumann-Architektur und der Harvard-Architektur.
 

(6 Punkte)
- 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.
 

(8 Punkte)
- Schreiben Sie einen Programmablauf, der über Port B, Bitposition 3 das in Abb. 1 gezeigte Bitmuster ausgibt.
 

(12 Punkte)



**Abb. 1**

- Wie weisen Sie dem Register r21 den symbolischen Namen DIVISOR zu?
 

(5 Punkte)
- Abb. 2 zeigt einen Ausschnitt aus einer Belegung des Registerspeichers. Es ist wichtig, daß diese Registerinhalte erhalten bleiben. Jetzt sind Datenbytes aus dem Programmspeicher zu lesen, und zwar mit dem herkömmlichen LPM-Befehl. Die Aufgabe: Lesen eines Datenbytes ins Register r17. Die Adresse steht bereits fertig im Adreßregister Z. Geben Sie eine Befehlsfolge an, die diesen Datentransport ausführt.
 

(8 Punkte)

r0	PGM_STATE
r1	ARS_PTR_LO
r2	ADRS_PTR_HI

**Abb. 2**

7. Schreiben Sie einen Programmablauf, der eine Zeichenkette auf eine LCD-Anzeige ausgibt. Zur Zeichenausgabe ist ein Unterprogramm LCD\_OUT aufzurufen. Dabei ist das zu übertragende Zeichen in Register r16 zu übergeben. Die Zeichenkette steht im Programmspeicher und wird dort über die Marke TXT\_STRING angesprochen. Das Ende der Kette ist durch ein Zeichen 00H gekennzeichnet (vgl. Programmiersprache C). Dieses Endezeichen ist nicht auszugeben.

(12 Punkte)

8. Geben Sie an, wie Sie eine Zeichenkette gemäß Aufgabe 7 in den Quelltext des Assemblerprogramms eintragen. Textbeispiel: "Institut fuer Musikforschung".

3 Punkte)

9. Beim Aufruf eines Unterprogramms werden Register in den Stack gerettet und bei der Rückkehr wieder zurückgespeichert. Ist die nachstehende Befehlsfolge o.k.? Schlagen Sie ggf. eine entsprechende Änderung vor.

(10 Punkte)

```

PUSH TEMP
IN TEMP, SREG
PUSH TEMP
PUSH r24
PUSH r25
...
POP r24
POP r25
POP TEMP
OUT SREG,TEMP
POP TEMP
RET
    
```

10. Der A/D-Wandler eines Mikrocontrollers liefert einen Wert von 10 Bits Länge. Formen Sie diesen Wert in einen 8-Bit-Wert um (Abb. 3).

(10 Punkte)

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. 3**

11. Schreiben Sie ein Assemblerprogrammstück, das folgenden Ablauf implementiert (Abb. 4).

(10 Punkte)

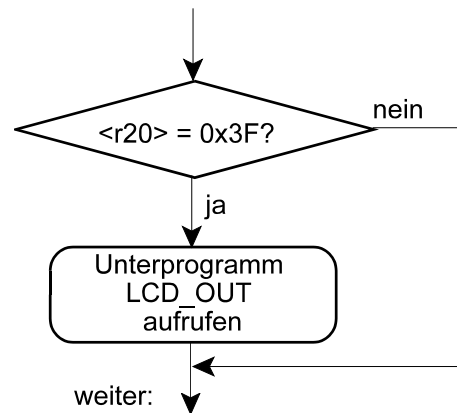


Abb. 4

12. Die Register r0 bis r5 enthalten jeweils 24 Bits lange Binärzahlen (Abb. 5). Schreiben Sie einen Programmablauf, der beide Zahlen so addiert, daß das Ergebnis in den Registern r3 bis r5 zu stehen kommt.

r0	LO_INT_0
r1	MID_INT_0
r2	HI_INT_0
r3	LO_INT_1
r4	MID_INT_1
r5	HI_INT_1

Abb. 5

### Zusatzaufgaben:

Z1. Erläutern Sie kurz (Aufzählung), aus welchen Teilen eine typische Anweisungszeile in einem Assemblerprogramm besteht.

(5 Punkte)

Z2. Erweitern Sie die Lösung der Aufgabe 10 durch eine Rundungsfunktion. Hierzu sind die weggeworfenen Bits 1 und 0 auszuwerten:

- Bitbelegungen 0,1 und 0,0: nichts tun.
- Bitbelegungen 1,0 und 1,1: Ergebnis um 1 erhöhen (Aufrunden), aber nur bis zum Endwert FFH (Sättigungsarithmetik).

(12 Punkte)

**Viel Erfolg!**

<b>Name:</b> <i>FH Dortmund</i>	<b>Matr.-Nr.:</b> <i>FB Informations- und Elektrotechnik</i>
------------------------------------	---

## Automatisierungstechnik AP1

Klausur vom 3. 7. 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. 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)

2. Ein Tastenfeld ist an Port A angeschlossen. Es soll abgefragt werden, ob die Taste an Bitposition 5 betätigt ist oder nicht (Tastenwirkung: aktiv low). Das Programmstück:

```
WAIT_5:
        SBIC     PORTA,5
        RJMP    WAIT_5
```

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

(5 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. 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. 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:

(10 Punkte)

6. 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.

(15 Punkte)

7. Initialisieren Sie Port C so, daß Bitposition 5 als Ausgang konfiguriert ist. Alle anderen Bitpositionen sollen als Eingänge wirken.

(5 Punkte)

8. Über Port C, Bitposition 5 ist das in Abb. 1 gezeigte Bitmuster auszugeben.

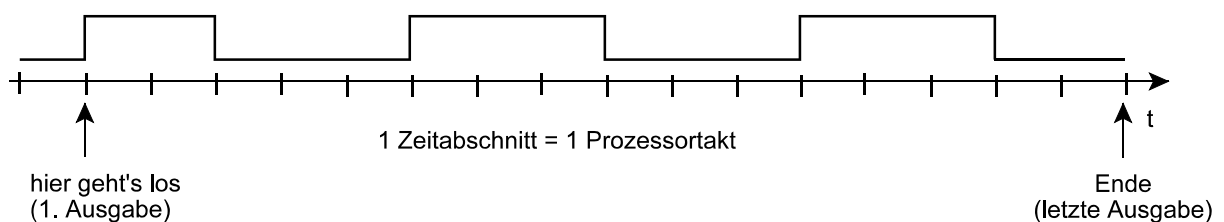


Abb. 1

10 Punkte)

9. 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
```

- Weshalb?
- Wie helfen Sie sich?

(10 Punkte)

10. Die Register r4 bis r7 und r16 bis r19 enthalten jeweils 32 Bits lange Binärzahlen Z1 und Z2. (Abb. 2). Schreiben Sie einen Programmablauf, der  $Z1 - Z2$  berechnet und das Ergebnis in Z1 speichert.

r4	Z1_0
r5	Z1_1
r6	Z1_2
r7	Z1_3
r16	Z2_0
r17	Z2_1
r18	Z2_2
r19	Z2_3

**Abb. 2**

(10 Punkte)

### Zusatzaufgaben:

Z1. Wie weisen Sie dem Register r15 den symbolischen Namen TEMP\_LIMIT zu?

(5 Punkte)

Z2. Wir beziehen uns auf Aufgabe 10. Es ist jetzt zu rechnen  $Z3 = Z1 - Z2$ , wobei Z3 die Register r22 bis r25 belegt. Die ursprünglichen Zahlen Z1 und Z2 sollen dabei erhalten bleiben.

(10 Punkte)

**Viel Erfolg!**

<b>Name:</b>	<b>Matr.-Nr.:</b>
<i>FH Dortmund</i>	<i>FB Informations- und Elektrotechnik</i>

## Automatisierungstechnik AP1

Klausur vom 28. 1. 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. In den Registern r3 und r4 steht eine 16-Bit-Binärzahl INT\_1 (Abb. 1). Schreiben Sie einen Programmablauf für folgende Berechnung:

$$\text{INT\_2} = \text{INT\_1} - 25\text{BFH}$$

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. 1**

2. Skizzieren Sie (in Form eines Schaltbildes für eine Bitposition) den Aufbau eines E-A-Ports.  
(10 Punkte)
3. 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:
- Zeichenkette TEXT. Die Zeichenkette ist mit 00H abgeschlossen. Dieses Endezeichen ist nicht auszugeben.
  - 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)

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

(10 Punkte)

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

**Abb. 2**

5. Welche Speichereinrichtungen hat ein Atmel-AVR-Mikrocontroller?

(5 Punkte)

6. Initialisieren Sie Port D so, daß Bitpositionen 3 und 8 als Eingänge konfiguriert sind. Alle anderen Bitpositionen sollen als Ausgänge wirken.

(5 Punkte)

7. Beim Aufruf eines Unterprogramms werden Registerinhalte in den Stack gerettet und bei der Rückkehr wieder zurückgespeichert. Ist die nachstehende Befehlsfolge o.k.? Schlagen Sie ggf. eine entsprechende Änderung vor.

(6 Punkte)

```

IN TEMP, SREG
PUSH TEMP
PUSH TEMP
PUSH r12
PUSH r13
PUSH r14
...
POP r14
POP r12
POP r13
POP TEMP
OUT SREG,TEMP
POP TEMP
RET

```

8. 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)

9. Ein Mikrocontroller ist als Kabeltester einzusetzen. Das maximal 8adrige Kabel (vgl. die typischen Netzkabel) ist an die Ports C und D angeschlossen. C dient zur Ausgabe die Prüfmusters, 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)

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

(8 Punkte)

**Zusatzaufgaben:**

Z1. Wir beziehen uns auf Aufgabe 1. 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. Die Register r4 bis r6 und r16 bis r18 enthalten jeweils 24 Bits lange vorzeichenlose Binärzahlen Z1 und Z2 (Abb. 3). 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. 3**

**Viel Erfolg!**

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