Mikrocontroller-Grundlagen

Im achten Teil dieser Artikelserie beschreiben wir die Arithmetik- und Bitmanipulationsbefehle der MCS-51-Familie.

Teil 8

3.3 Arithmetik-Befehle

Eine wichtige Grundfunktion moderner Mikroprozessoren stellen die Arithmetik-Funktionen wie Addition und Multiplikation dar. Abbildung 70 zeigt eine Übersicht über die Arithmetik-Befehle der MCS-51-Familie mit einer jeweiligen Kurzbeschreibung. Neben den Inkrementier-, Dekrementier-, Additions- und Subtraktionsfunktionen beherrscht der Prozessor die 8-Bit-Multiplikation bzw. -Division.

Das Programm 15 in Abbildung 71 zeigt ein Beispiel für die Verwendung des Inkrement-Befehls, der den Inhalt des Akkumulators um 1 erhöht, wobei ein Wert von 0FFH zu 000H wird. Das Programm liest den Inhalt vom Port P1 aus, inkrementiert dessen Wert und gibt das Ergebnis am Port P3 aus. Interpretiert man die Schalterstellungen der Schalter S 0 bis S 7 als binären Zahlenwert, so ergibt sich an den Leuchtdioden D 0 bis D 7 der um 1 erhöhte Zahlenwert.

Der ADD-Befehl addiert den Inhalt des Akkumulators und des angegebenen Operanten. Das Ergebnis wird wiederum im Akku abgelegt, wobei auch das Aktualisieren der im Programmstatuswort PSW untergebrachten Flags OV, C und AC dazugehört. Das Hilfs-Carry-Flag AC wird gesetzt, wenn ein Übertrag vom 3. Bit erfolgt, während das Carry-Bit C bei einem Übertrag vom 7. Bit gesetzt wird. Wenn kein Übertrag erforderlich ist, werden die Flags entsprechend gelöscht.

Der ADD-Befehl setzt das OV-Bit, wenn ein Übertrag ins 6. Bit aber nicht ins 7. Bit oder ein Übertrag ins 7. Bit und nicht ins 6. Bit erfolgt. Anderenfalls wird es gelöscht. Bei der Addition von vorzeichenbehafteten Zahlen zeigt das OV-Bit auf ein scheinbar negatives Ergebnis bei der Addition zweier positiver Zahlen oder auf ein scheinbar positives Ergebnis bei der Addition zweier negativer Zahlen.

Das in Abbildung 72 gezeigte Programm 16 demonstriert die Funktionsweise des Additions-Befehls. Das Ergebnis der Addition wird wiederum in den Akkumulator geschrieben, während ein Überlauf (Ergebnis größer als 255) durch das Setzen des Carry-Flags (C) angezeigt wird.

Das Beispielprogramm liest den Inhalt des Ports P1 aus und addiert den konstanten Zahlenwert 020H hinzu. Anschließend wird durch den Rotierbefehl RLC A der Akkuinhalt um 1 nach links geschoben und der Inhalt des Carry-Flags in Bit 0 abgelegt. Praktisch bedeutet dies, daß der Wert der Schaltereinstellungen von S 0 bis S 7 in binärer Form übernommen wird, zu dessen Wert die Konstante 020H addiert und das Ergebnis über die Leuchtdioden D 1 bis D 7 auf der LED-Ausgabeplatine angezeigt wird.

Die Leuchtdiode D 0 zeigt den Zustand des Carry-Flags an. Diese LED ist aktiviert, sobald ein Überlauf bei der Addition des Akkumulators mit der Konstanten aufgetreten ist. Bei Eingangsbitkombinationen, deren Binärzahl größer als 0DFH ist, entsteht bei der anschließenden Addition ein Überlauf, wodurch die LED D 0 aktiviert wird.

Abbildung 73 zeigt das Beispielprogramm 17, welches die Verwendungsmöglichkeit des Additionsbefehls vertieft. Zu-

Mnemonik		Beschreibung
INC	A	Increment Accumulator
INC	Rr	Increment Register
INC	direct	Increment direct (byte)
INC	@Ri	Increment indirect RAM
DEC	A	Decrement Accumulator
DEC	Rr	Decrement Register
DEC	direct	Decrement direct (byte)
DEC	@Ri	Decrement indirect RAM (16 Bit)
INC	DPTR	Increment Data Pointer
ADD	A,Rr	Add Register to Accumulator
ADD	A,direct	Add direct (byte) to Accumulator
ADD	A,@Ri	Add indirect RAM to Accumulator
ADD	A,#data	Add immediate data to Accumulator
ADDC	A,Rr	Add Register to Accumulator with Carry
ADDC	A,direct	Add direct (byte) to Accumulator with Carry
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry
ADDC	A,#data	Add immediate data to Accumulator with Carry
SUBB	A,Rr	Subtract Register from Accumulator with Borrow
SUBB	A,direct	Subtract direct (byte) from Accumulator with Borrow
SUBB	A,@Ri	Subtract indirect RAM from Accumulator with Borrow
SUBB	A,#data	Subtract immediate data from Accumulator with Borrow
MUL	AB	Multiply A with B
DIV	AB	Divide A by B
DA	A	Decimal Adjust Accumulator

Bild 70: Übersicht über die Arithmetik-Befehle der MCS-51-Familie

```
012D E590   L15:  MOV   A,P1     ; P1 lesen
012F 04           INC   A        ; Inhalt um 1 erhöhen
0130 F5B0         MOV   P3,A     ; Wert nach P3 schreiben
0132 80F9         SJMP  L15      ; Schleife
```

Bild 71: Testprogramm 15

```
0134 E590   L16:  MOV   A,P1     ; P1 lesen
0136 2420         ADD   A,#020H  ; Konstante addieren
0138 33           RLC   A        ; Linksschieben mit Carry
0139 F5B0         MOV   P3,A     ; Wert nach P3 schreiben
013B 80F7         SJMP  L16      ; Schleife
```

Bild 72: Testprogramm 16:

nächst wird der Inhalt vom Port P1 eingelesen und über die folgenden Befehle die unteren 4 Bits (S 0 bis S 3) zu den oberen 4 Bits (S 4 bis S 7) binär addiert und das Ergebnis auf dem Port P3 ausgegeben. Da die jeweils 4 zueinandergehörenden Bits (1 Nibble) den Zahlenumfang von 0 bis 15 darstellen, liegt das Ergebnis im Bereich zwischen 0 und 31, d. h. die Leuchtdioden D 5 bis D 7 sind immer erloschen.

Zur Verdeutlichung der Funktionsweise dieses Programmes zeigt Abbildung 74 die Zustände des Akkumulators und des Registers R 0 nach der Ausführung der Befehle.

Zusätzlich kann mit Hilfe des Befehls ADDC (Add with Carry) zu dem angegebenen Datum noch der Inhalt des Carry-Flags (0 oder 1) addiert werden. Diese Funktion findet vornehmlich Einsatz in der Fließkomma-, Arithmetik- oder auch String-Verarbeitung.

Beim SUBB-Befehl wird das Carry-Flag nicht als Überlauf, sondern als Unterlauf (Borrow)-Flag genutzt und somit gesetzt, wenn ein Übertrag für das 7. Bit benötigt wird, sonst wird es gelöscht.

Das Carry-Flag wird grundsätzlich mit dem Subtrahenden vom Inhalt des Akkumulators abgezogen. Es kann bei Mehrfach-Subtraktionen einen Untertrag kennzeichnen. Vor der ersten Subtraktion sollte daher das Carry-Flag gelöscht werden.

Der SUBB-Befehl setzt das AC-Flag, wenn ein Übertrag für das 3. Bit benötigt wird. Der Befehl setzt, wie auch bei der Addition das OV-Bit, wenn ein Übertrag ins 6. Bit aber nicht ins 7. Bit oder ein Übertrag ins 7. Bit und nicht ins 6. Bit benötigt wird, sonst wird es gelöscht. Bei der Subtraktion von vorzeichenbehafteten Zahlen macht es auf ein negatives Ergebnis aufmerksam, wenn ein negativer Wert von einem positiven abgezogen wird oder auf ein positives Ergebnis, wenn eine positive Zahl von einer negativen Zahl subtrahiert wird.

In dem in Abbildung 75 dargestellten Programm 18 wird zunächst der Inhalt des Ports P1 in den Akkumulator eingelesen und das Carry-Flag gesetzt, um anschließend die Konstante 03H und das gesetzte Carry-Flag vom Inhalt des Akkumulators abzuziehen. Das Ergebnis der Operation wird im Akkumulator abgelegt, wobei das Carry-Flag gesetzt ist, wenn die binäre Kombination der Schalter S 0 bis S 7 kleiner als 4 ist.

Um das Carry-Flag sichtbar zu machen, wurde ähnlich wie in Abbildung 72 der Inhalt des Akkumulators um 1 nach links geschoben und auf dem Port P3 ausgegeben. Das Ergebnis dieser Subtraktion spiegelt sich somit in den Bits 1 bis 7 wider, während das Bit 0 den Inhalt des Carry-Flags anzeigt.

Abbildung 76 zeigt das Beispielprogramm 19, welches die Verwendungsmöglichkeiten des Subtraktionsbefehls verdeutlicht. Zunächst wird der Inhalt von Port P1 eingelesen. Über die folgenden Befehle wird nun erreicht, daß mit der SUBB-Anweisung der Inhalt der Bits 0 bis 3 von den Bits 4 bis 7 abgezogen wird. Letztere stellen jeweils ein Nibble dar, die über die Schalter S 0 bis S 7 vorgegeben werden. Das 4-Bit-Ergebnis wird anschließend in den Port P3 geschrieben, wobei ein negatives Ergebnis in diesem Fall durch die gesetzten Bits 4 bis 7 dargestellt wird.

Zur Verdeutlichung der Funktionsweise dieses Programmes zeigt Abbildung 77 die Zustände des Akkumulators und RAM-Inhaltes nach der Ausführung der jeweiligen Befehle. Anzumerken ist in diesem Zusammenhang, daß für die Auftrennung der 4-Bit-Informationen ein etwas anderer Weg als im Testprogramm 17 gewählt wurde, um die Flexibilität dieser Mikrocontrollerarchitektur zu demonstrieren.

Zu den wichtigsten Arithmetik-Befehlen gehören die Multiplikations- und Divi-

```
013D E590   L17:  MOV   A,P1          ; P1 lesen
013F F8           MOV   R0, A         ; A nach R0 kopieren
0140 540F         ANL   A, #00001111B ; Bit 4..7 löschen
0142 C8           XCH   A, R0         ; A und R0 tauschen
0143 C4           SWAP  A             ; Nibble tauschen
0144 540F         ANL   A, #00001111B ; Bit 4..7 löschen
0146 28           ADD   A, R0         ; Inhalt A und R0 addieren
0147 F5B0         MOV   P3,A          ; Wert nach P3 schreiben
0149 80F2         SJMP  L17           ; Schleife
```

Bild 73: Testprogramm 17

Befehl		Accumulator								R0							
nach MOV	A, P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	x	x	x	x	x	x	x	x
nach MOV	R0, A	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
nach ANL	A, # 00FH	0	0	0	0	P1.3	P1.2	P1.1	P1.0	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
nach XCH	A, R0	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach SWAP	A	P1.3	P1.2	P1.1	P1.0	P1.7	P1.6	P1.5	P1.4	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach ANL	A, # 00FH	0	0	0	0	P1.7	P1.6	P1.5	P1.4	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach ADD	A, R0	(P1.7...P1.4) + (P1.3...P1.0)								0	0	0	0	P1.3	P1.2	P1.1	P1.0

Bild 74: Zustände des Akkumulators und R 0 nach der Befehlsausführung

```
014B E590   L18:  MOV   A,P1          ; P1 lesen
014D D3           SETB  C             ; Carry Bit setzen
014E 9403         SUBB  A, #003H      ; Konstante subtrahieren
0150 33           RLC   A             ; Linksschieben mit Carry
0151 F5B0         MOV   P3,A          ; Wert nach P3 schreiben
0153 80F6         SJMP  L18           ; Schleife
```

Bild 75: Testprogramm 18

```
0155 7820         MOV   R0, #020H     ; RAM-Zeiger laden
0157 E590   L19:  MOV   A,P1          ; P1 lesen
0159 7600         MOV   @R0, #000H    ; Inhalt auf 0 setzen
015B D6           XCHD  A, @R0        ; nur die unteren 4 Bit tauschen
015C C4           SWAP  A             ; Nibble tauschen
015D C3           CLR   C             ; Carry Bit löschen
015E 96           SUBB  A, @R0        ; RAM-Inhalt subtrahieren
015F F5B0         MOV   P3,A          ; Wert nach P3 schreiben
0161 80F4         SJMP  L19           ; Schleife
```

Bild 76: Testprogramm 19

sionsanweisungen. Diese ermöglichen den Aufbau von umfangreichen Fließkomma-, Arithmetik- und Tabellenverarbeitungsroutinen. Der Multiplikationsbefehl MUL AB multipliziert den vorzeichenlosen Inhalt des Akkumulators mit dem vom Register B. Das Low-Byte des 16-Bit-Produktes steht im Akkumulator, während das High-Byte im Register B abgelegt wird. Wenn das Ergebnis größer als 0FFH ist, setzt dieser Multiplikationsbefehl das OV-Bit, anderenfalls wird es gelöscht. Das Carry-Flag (C) wird dabei immer gelöscht.

Abbildung 78 zeigt das Programm 20, welches die binären Schaltereinstellungen von S 0 bis S 3 mit denen von S 4 bis S 7 multipliziert und über die Leuchtdioden D 0 bis D 7, die an den Port P3 angeschlossen sind, ausgibt. Das größtmögliche Ergebnis in unserem Beispielprogramm ergibt sich aus der Multiplikation der beiden Zahlen 0FH miteinander, woraus sich das Ergebnis 11100001B ergibt.

Zur Verdeutlichung der Funktionsweise des Programmes zeigt Abbildung 79 die Zustände des Akkumulators und des Registers B nach der Ausführung der einzelnen Befehle.

Der 8-Bit-Divisions-Befehl DIV AB teilt den vorzeichenlosen Wert im Akkumulator durch den vorzeichenlosen Wert im B-Register. Nach der Division enthält der Akkumulator das ganzzahlige Ergebnis, während im Register B der ganzzahlige Rest der Division steht. Bei der Division durch 0 sind die Werte im Akkumulator und im Register B unbestimmt und das OV-Bit gesetzt, anderenfalls ist es gelöscht. Das Carry-Bit (C) wird bei der Division immer gelöscht.

Der Divisions-Befehl findet vornehmlich bei der Fließkomma-Arithmetik- und BCD-Konvertierung eines 8-Bit-Wertes Einsatz.

Abbildung 80 zeigt das Beispielprogramm 21, welches die Funktion des DIV-

Befehl	Accumulator								RAM (020 H)							
nach MOV A, P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	x	x	x	x	x	x	x	x
nach MOV @R0, #000 H	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	0	0	0	0	0	0	0	0
nach XCHD A, @ R0	P1.7	P1.6	P1.5	P1.4	0	0	0	0	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach SWAP A	0	0	0	0	P1.7	P1.6	P1.5	P1.4	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach CLR C	0	0	0	0	P1.7	P1.6	P1.5	P1.4	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach SUBB A, @ R0	(P1.7...P1.4) – (P1.3...P1.0)								0	0	0	0	P1.3	P1.2	P1.1	P1.0

Bild 77: Zustände des Akkumulators und des RAM-Speicherplatzes 20 H nach der Befehlsausführung

```
0163 8590F0  L20:  MOV   B,P1            ; P1 lesen
0166 E5F0          MOV   A,B             ; B nach A kopieren
0168 540F          ANL   A, #00001111B   ; Bit 4..7 löschen
016A C5F0          XCH   A,B             ; A und B tauschen
016C C4            SWAP  A               ; Nibble tauschen
016D 540F          ANL   A, #00001111B   ; Bit 4..7 löschen
016F A4            MUL   AB              ; Inhalt A und B multiplizieren
0170 F5B0          MOV   P3,A            ; Wert nach P3 schreiben
0172 80EF          SJMP  L20             ; Schleife
```

Bild 78: Testprogramm 20

Befehl	Accumulator								B							
nach MOV B, P1	x	x	x	x	x	x	x	x	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
nach MOV A, B	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
nach ANL A, #00FH	0	0	0	0	P1.3	P1.2	P1.1	P1.0	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
nach XCH A, B	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach SWAP A	P1.3	P1.2	P1.1	P1.0	P1.7	P1.6	P1.5	P1.4	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach ANL A, #00FH	0	0	0	0	P1.7	P1.6	P1.5	P1.4	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach MUL A, B	(P1.7...P1.4) • (P1.3...P1.0)								0	0	0	0	P1.3	P1.2	P1.1	P1.0

Bild 79: Zustände des Akkumulators und des Registers B nach der Befehlsausführung

```
0174 8590F0  L21:  MOV   B,P1            ; P1 lesen
0177 E5F0          MOV   A,B             ; B nach A kopieren
0179 540F          ANL   A, #00001111B   ; Bit 4..7 löschen
017B C5F0          XCH   A,B             ; A und B tauschen
017D 54F0          ANL   A, #11110000B   ; Bit 0..3 löschen
017F 84            DIV   AB              ; Inhalt A durch B dividieren
0180 F5B0          MOV   P3,A            ; Wert nach P3 schreiben
0182 80F0          SJMP  L21             ; Schleife
```

Bild 80: Testprogramm 21

Befehl	Accumulator								B							
nach MOV B, P1	x	x	x	x	x	x	x	x	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
nach MOV A, B	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
nach ANL A, #00FH	0	0	0	0	P1.3	P1.2	P1.1	P1.0	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
nach XCH A, B	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach ANL A, #0F0H	P1.7	P1.6	P1.5	P1.4	0	0	0	0	0	0	0	0	P1.3	P1.2	P1.1	P1.0
nach DIV A, B	(P1.7...P1.4) • 16/ (P1.3...P1.0)								0	0	0	0	P1.3	P1.2	P1.1	P1.0

Bild 81: Zustände des Akkumulators und des Registers B nach der Befehlsausführung

Befehls vertieft. Das Testprogramm verarbeitet hierzu die logischen Pegel der Schaltereinstellungen S 0 bis S 3 und S 4 bis S 7 nibbleweise. Es teilt das mit 16 multiplizierte höherwertige Nibble (S 4 bis S 7) durch den Wert des niederwertigen Nibbles (S 0 bis S 3). Das Ergebnis, welches in diesem Beispielprogramm im 8-Bit-Bereich liegt, wird über die Leuchtdioden D 0 bis D 7 ausgegeben.

Zur Verdeutlichung der Funktionsweise dieses Programmes zeigt Abbildung 81 die Zustände des Akkumulators und Registers B nach der Ausführung der jeweiligen Befehle.

Die Prozessoren der MCS-51-Familie sind für die binäre Datenverarbeitung ausgelegt. Soll eine BCD-Addition erfolgen, muß nach jeder einzelnen Addition der DA A-Befehl (Decimal-Adjust-Accumulator for Addition) zur BCD-Korrektur eingesetzt werden. Der DA A-Befehl sollte ausschließlich nach einer vorausgegangenen Addition zweier BCD-Zahlen eingesetzt werden.

Die Additionsbefehle setzen oder löschen die für den DA A-Befehl wichtigen Flags C und AC. Zunächst addiert der DA A-Befehl den Wert 6 zum Akkumulator, wenn die Bits 0 bis 3 des Akkus größer als 9 sind (xxxx1010B bis xxxx1111B) oder wenn das Hilfs-Carry-Flag AC gesetzt ist, zum Erhalt des richtigen BCD-Wertes im Low-Nibble des Akkumulators. Wenn bei dieser Addition ein Übertrag über das 7. Bit hinausgeht, wird das Carry-Flag gesetzt. Im Gegensatz zu anderen Befehlen wird ein eventuell gesetztes Carry-Flag nicht gelöscht, wenn kein Übertrag vom 7. Bit vorliegt.

Im zweiten Schritt wird auch zum High-Nibble der Wert 6 addiert, wenn das Carry-Flag gesetzt oder der Wert des High-Nibbles größer als 9 (1010xxxxB bis 1111xxxxB) ist, um auch hier die BCD-Korrektur vorzunehmen. Auch dieser zweite Schritt kann das Carry-Flag setzen aber nicht löschen.

Das gesetzte Carry-Flag zeigt nun für den folgenden Programmablauf an, daß die Summe der addierten BCD-Zahlen größer als 99 geworden ist. Der DA A-Befehl addiert zum Akkumulatorinhalt je nach Additionsergebnis die Werte 00H, 06H, 60H oder 66H. Zu beachten ist, daß dieser Befehl keine beliebig großen 8-Bit-Binärwerte in BCD-Ziffern umwandeln kann.

Abbildung 82 zeigt das Beispielprogramm 22, welches die Funktion des DA A-Befehls verdeutlicht. Zunächst wird der Inhalt von Port P1 gelesen und der Zahlenwert 000H addiert. Diese Addition löscht gleichzeitig das Carry-Flag C und das Hilfs-Carry-Flag AC. Der nachfolgende DA A-Befehl nimmt nun im Rahmen seiner Möglichkeiten eine BCD-Korrektur vor, mit der anschließenden Ausgabe auf dem Port P3, wo die Leuchtdioden D 0 bis D 7 den Inhalt des Akkumulators anzeigen.

```
0184 E590     L22:  MOV   A,P1       ; P1 lesen
0186 2400           ADD   A,#000H    ; AC- und C-Flags löschen
0188 D4             DA    A          ; Dezimal-Anpassung
0189 F5B0           MOV   P3,A       ; Wert nach P3 schreiben
018B 80F7           SJMP  L22        ; Schleife
```

Bild 82: Testprogramm 22

Mnemonik		Beschreibung
CLR	C	Clear Carry flag
CLR	bit	Clear direct bit
SETB	C	Set Carry flag
SETB	bit	Set direct bit
CPL	C	Complement Carry flag
CPL	bit	Complement direct bit
ANL	C,bit	AND direct bit to Carry flag
ANL	C,/bit	AND complement of direct bit to Carry flag
ORL	C,bit	OR direct bit to Carry flag
ORL	C,/bit	OR complement of direct bit to Carry flag
MOV	C,bit	Move direct bit to Carry flag
MOV	bit,C	Move Carry flag to direct bit

Bild 83: Übersicht über die Bitmanipulations-Befehle der MCS-51-Familie

```
018D 75B000         MOV   P3,#000H   ; alle LEDs löschen
0190 A290     L23:  MOV   C,P1.0     ; P1.0 lesen
0192 8291           ANL   C,P1.1     ; UND-Verknüpfung mit P1.1
0194 92B0           MOV   P3.0,C     ; Ausgabe auf Bit 0 von P3
0196 80F8           SJMP  L23        ; Schleife
```

Bild 84: Testprogramm 23

```
0198 75B000         MOV   P3,#000H   ; alle LEDs löschen
019B A291     L24:  MOV   C,P1.1     ; P1.0 lesen
019D 7290           ORL   C,P1.0     ; ODER-Verknüpfung mit P1.0
019F 92B7           MOV   P3.7,C     ; Ausgabe auf Bit 7 von P3
01A1 80F8           SJMP  L24        ; Schleife
```

Bild 85: Testprogramm 24

3.4 Bitmanipulations-Befehle

Der 8051-Mikroprozessor wird auch als Boolscher-Prozessor bezeichnet, da er umfangreiche Bitverarbeitungs-Befehle zur Verfügung stellt. Abbildung 83 zeigt eine Übersicht über die Bitmanipulationsbefehle. Die Verknüpfungen lassen sich jeweils mit dem Carry-Flag C oder auch mit einer bitadressierbaren Speicherzelle vornehmen.

Die Befehle CLR, SETB und CPL ermöglichen das Löschen, Setzen und Invertieren einer Bit-Information. Mit den ANL- und ORL-Befehlen lassen sich Bitvariable gezielt mit dem Carry-Flag UND oder ODER verknüpfen. Ein Kopieren der Bitinformation von und zum Carry-Flag ist über den Transportbefehl MOV möglich.

Das Programm 23 aus Abbildung 84 zeigt die Verwendung der bitweisen UND-Verknüpfung. Zunächst wird der Inhalt des Ports P3 auf 0 gesetzt, woraufhin die Leuchtdioden D 0 bis D 7 erlöschen. In der folgenden Schleife wird nun permanent der Inhalt des Ports P1.0 (Schalter S 0) mit dem Inhalt von dem Port P1.1 (Schalter S 1) UND-verknüpft und das Ergebnis auf dem Port P3.0 (LED D 0) ausgegeben, d. h. die Leuchtdiode D 0 wird nur aktiviert, wenn die logischen Pegel der Schalter S 0 und S 1 auf High liegen.

Abbildung 85 zeigt das Testprogramm 24, welches die ODER-Verknüpfung der Schalter S 0 und S 1 vornimmt und das Ergebnis auf dem Port P 3.7 (LED D 7) ausgibt. Daraus ergibt sich, daß die Leuchtdiode D 7 nur aktiv ist, wenn mindestens einer der Schalter auf aktiv high geschaltet ist.

Im nächsten Teil dieser Artikelserie beschreiben wir ausführlich die Programm-Steuerbefehle. ELV