Befehlssatz der Mikrocontroller der 51er -Familie

Mikrocontrollerfamilie 8051

Befehlssatz

Abkürzungen:

A :	Akkumulator
Rn :	Register R0..R7
Ri :	R0 oder R1
dadr :	direkte Byte-Adresse im int. Speicher (auch SF-Register)
badr :	direkte Bit-Adresse im int. Speicher (auch SF-Flags)
adr16 :	absolute 16-bit Adresse
adr11 :	11-bit Adresse relative zu 2k-Seite (0 .. 2047 bzw. 0 .. 7FFH)
rel :	8-bit Adresse relative zu Befehl (-128 .. +127 bzw. 80H .. 7FH)
konst8 :	8-bit Konstante (0 .. 255 bzw. 0 .. 0FFH)
konst16 :	16-bit Konstante (0.. 65535 bzw. 0 .. 0FFFFH)
#B :	Anzahl der Bytes, die der Befehl belegt
#Z :	Anzahl der Prozessorzyklen für die Befehlsausführung

Adressierungsarten:

Registeradressierung: in Befehlen, die nur mit *A*, *B*, *Rn* und *DPTR* arbeiten
Bsp.: *MOV A,R0*

Direkte Adressierung: alle Adressen im internen Speicher (0..7FH und SFR)
Bsp.: *MOV dadr,80H*
MOV A,SCON

Indirekte Adressierung (int. Speicher): Adresse (0..FF) steht in *Ri*
Bsp.: *MOV A,@R0*

Indirekte Adressierung (ext. Speicher): Adresse (0..FFFF) steht in *DPTR* oder Adresse (0..0FFH) in *Ri*
Bsp.: *MOVX A,@DPTR*

Indirekte, indizierte Adressierung (Programm-Speicher): Adresse berechnet sich aus *A* + *DPTR* oder *A* + *PC*
Bsp.: *MOVC A,@A+DPTR*

Unmittelbare Adressierung (immediate): Die Adresse ist der Befehl selbst (Laden von Konstanten)
Bsp.: *MOV R7,#0FFH* *(8-bit Wert)*
MOV DPTR,#8000H (16-bit Wert)

Transfer-Befehle (Daten kopieren):

Kopieren eines Datenbytes aus dem und in den internen Speicher / SFR:

MOV	*A,Rn*	1 B	1 Z	*dadr,A*	2 B	1 Z
	A,dadr	2 B	1 Z	*dadr,Rn*	2 B	2 Z
	A,@Ri	1 B	1 Z	*dadr,dadr*	3 B	2 Z
	A,#konst8	2 B	1 Z	*dadr,@Ri*	2 B	2 Z
				dadr,#konst8	3 B	2 Z
	Rn,A	1 B	1 Z	*@Ri,A*	1 B	1 Z
	Rn,dadr	2 B	2 Z	*@Ri,dadr*	2 B	2 Z
	Rn,#konst8	2 B	1 Z	*@Ri,#konst8*	2 B	1 Z

Kopieren eines Datenbytes aus dem und in den externen Speicher:

MOVX	*A,@Ri*	1 B	2 Z	*@Ri,A*	1 B	2 Z
	A,@DPTR	1 B	2 Z	*@DPTR,A*	1 B	2 Z

Kopieren eines Datenbytes aus dem Programmspeicher:

MOVC	*A,@A+PC*	1 B	2 Z	
	A,@A+DPTR	1 B	2 Z	

Kopieren eines Datenbits von Carry in internen Datenspeicher und umgekehrt:

MOV	*C,badr*	2 B	1 Z	
	badr,C	2 B	2 Z	

Kopieren einer 16-bit Konstanten in das DPTR-Register:

MOV	*DPTR,#konst16*	3 B	2 Z	

© 05.09.05, J. Rathlev, IEAP-Kiel, Seite 4

Transfer-Befehle (Daten austauschen):

Austausch von Datenbytes zwischen Akkumulator und internem Speicher:

XCH	*A,Rr* *A,dadr* *A,@Ri*	1 B 2 B 1 B	1 Z 1 Z 1 Z	

Austausch der Datenbits 0..3 zwischen Akkumulator und Register:

XCHD	*A,@Ri*	1 B	1 Z	

Austausch der Datenbits 0..3 und 4..7 im Akkumulator:

SWAP	*A*	1 B	1 Z	

Transfer-Befehle (Daten von/nach Stack kopieren):

Kopieren eines Datenbytes vom internen Datenspeicher / SFR in den Stack:

PUSH	*dadr*	2 B	2 Z	

Kopieren eines Datenbytes aus dem Stack in den internen Datenspeicher / SFR:

POP	*dadr*	2 B	2 Z	

Sonstige Befehle:

Leerbefehl (No operation):

NOP		1 B	1 Z	

© 05.09.05, J. Rathlev, IEAP-Kiel, Seite 5

Logik-Befehle mit Bytes:

Das Ergebnis der logischen Operation steht immer im linken Operanden*:

ANL ***ORL*** ***XRL***	*A,Rn* *A,dadr* *A,@Ri* *A,#konst8*	1 B 2 B 1 B 2 B	1 Z 1 Z 1 Z 1 Z	*dadr,A* *dadr,#konst8*	2 B 3 B	1 Z 2 Z

Invertieren des Akkumulator-Inhalts:

CPL	*A*	1 B	1 Z	

Löschen des Akkumulator-Inhalts*:

CLR	*A*	1 B	1 Z	

*: Wenn das Ziel der Akkumulator ist, wird im PSW das *P*-Bit beeinflusst.

Logik-Befehle mit Bits:

Das Ergebnis der logischen Operation steht immer Carry:

ANL ***ORL***	*C,badr* *C,/badr*	2 B 2 B	2 Z 2 Z	

Invertieren eines Bits:

CPL	*C* *badr*	1 B 2 B	1 Z 1 Z	

Löschen eines Bits

CLR	*C* *badr*	1 B 2 B	1 Z 1 Z	

Setzen eines Bits

SETB	*C* *badr*	1 B 2 B	1 Z 1 Z	

© 05.09.05, J. Rathlev, IEAP-Kiel, Seite 6

Arithmetik-Befehle:

Das Ergebnis der arithmetischen Operation steht immer im Akkumulator*:

ADD ***ADDC*** ***SUBB***	*A,Rn* *A,dadr* *A,@Ri* *A,#konst8*	1 B 2 B 1 B 2 B	1 Z 1 Z 1 Z 1 Z	

*: Im PSW werden entsprechend dem Ergebnis die Bits *C,OV,AC* und *P* angepasst.

ADD(C): C = 1: Ergebnis >255

SUBB: C = 1: [Dest-Byte] < [Src-Byte] (z.B. [A] < konst8)

C = 0: [Dest-Byte] ≥ [Src-Byte] (z.B. [A] ≥ konst8)

Das jeweilig adressierte Datenbyte wird um 1 erhöht oder erniedrigt:

INC ***DEC***	*A* *Rn* *@Ri* *dadr*	1 B 1 B 1 B 2 B	1 Z 1 Z 1 Z 1 Z	

Das DPTR-Register (16 Bit) wird um 1 erhöht:

INC	*DPTR*	1 B	1 Z	

Der Akkumulator *A* wird mit Register *B* muliplizierт - Ergebnis in *B,A***:*

MUL	*AB*	1 B	4 Z	

Der Akkumulator *A* wird durch Register *B* dividiert - Quotient in *A*, Rest in *B***:

DIV	*AB*	1 B	4 Z	

**: Im PSW wird *OV* gesetzt, wenn das Ergebnis größer als 255 ist, *C* ist immer 0.

Dezimalkorrekur bei einer Addition von BCD-Zahlen in *A* ***:*

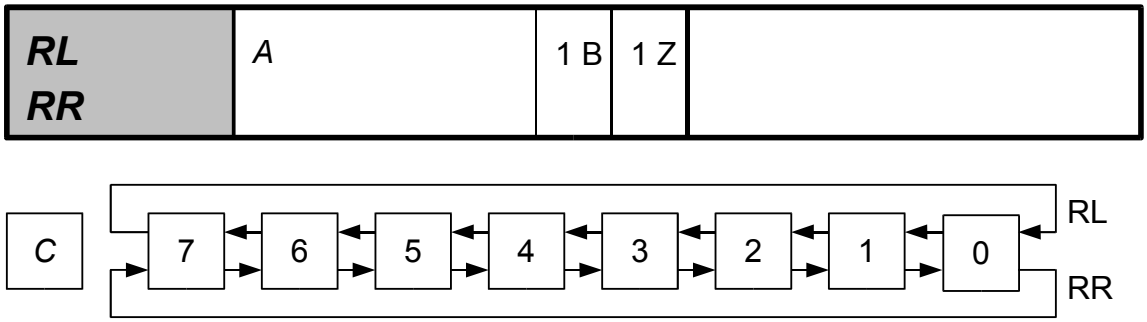
DA	*A*	1 B	1 Z	

***: Im PSW wird *C*-Bit (Carry) angepasst.

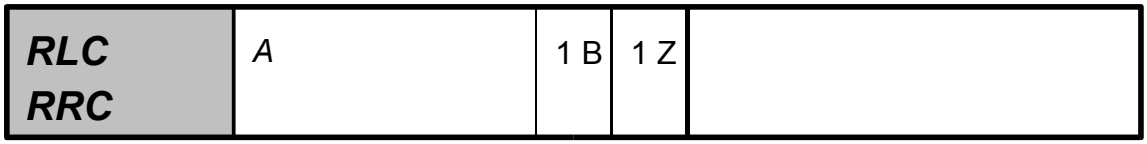
© 05.09.05, J. Rathlev, IEAP-Kiel, Seite 7

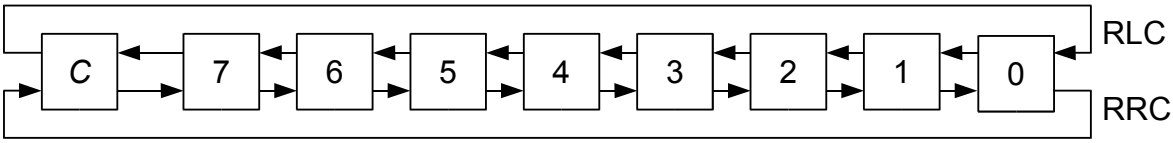
Schiebe-Befehle

Rotiert alle Bits des Akkumulators um eins nach links oder rechts (ohne *C*):



Rotiert alle Bits des Akkumulators mit Carry um eins nach links oder rechts:





Unbedingte Sprung-Befehle:

Sprung an eine absolute 16-bit Adresse:

LJMP	*adr16*	3 B	2 Z	

Sprung an eine relative 11-bit Adresse innerhalb einer 2k-Seite:

AJMP	*adr11*	2 B	2 Z	

Sprung an eine zum Befehl relative Adresse (-128 .. +127):

SJMP	*rel*	2 B	2 Z	

Sprung an eine indirekte und indizierte Adresse:

JMP	*@A+DPTR*	1 B	2 Z	

Unbedingter Sprung in eine Unterroutine:

Sprung an eine absolute 16-bit Adresse:

LCALL	*adr16*	3 B	2 Z	

Sprung an eine relative 11-bit Adresse innerhalb einer 2k-Seite:

ACALL	*adr11*	2 B	2 Z	

Rücksprung aus einer Unterroutine:

Rücksprung aus einer normalen Unterroutine:

RET		1 B	2 Z	

Rücksprung aus einer Interruptroutine:

RETI		1 B	2 Z	

© 05.09.05, J. Rathlev, IEAP-Kiel, Seite 9

Bedingte Sprung-Befehle:

Bedingter relativer Sprung in Abhängigkeit des Carry-Bits:

JC ***JNC***	*rel*	2 B	2 Z	

Bedingter relativer Sprung in Abhängigkeit vom Akkumulatorinhalt (=0 oder ≠0):

JZ ***JNZ***	*rel*	2 B	2 Z	

Bedingter relativer Sprung in Abhängigkeit von einem Bit:

JB ***JNB*** ***JBC***	*badr,rel*	3 B	2 Z	

Bedingter relativer Sprung, wenn Akkumulator ungleich Datenbyte oder Akkumulator/Register/Datenbyte ungleich Konstante ist:

CJNE	*A,dadr,rel* *A,#konst8,rel* *Rn,#konst8,rel* *@Ri,#konst8,rel*	3 B 3 B 3 B 3 B	2 Z 2 Z 2 Z 2 Z	

C = 1: [Dest-Byte] < [Src-Byte] (z.B. [A] < konst8)
C = 0: [Dest-Byte] ≥ [Src-Byte] (z.B. [A] ≥ konst8)

Das Register/Datenbyte wird um 1 erniedrigt, bedingter relativer Sprung, wenn ≠0:

DJNZ	*Rn,rel* *dadr,rel*	2 B 3 B	2 Z 2 Z	