12 Der Befehlssatz der Controller-Familie 8051

Es wird hier der Befehlssatz vorgestellt, der als Zusammenfassung (Instruction Set Summary) im Handbuch (Users Manual) des Controllers 80515 angegeben ist. Da die Controllerbausteine, ausgehend vom Typ 8051, aufwärts kompatibel sind, gelten die Befehle für alle Controllerbausteine der 8051-Familie.

Um den Rahmen dieses Buches nicht zu sprengen, werden die Befehle nicht im einzelnen ausführlich erläutert, sondern zu Funktionsgruppen zusammengefasst vorgestellt. Die englische Kurzbeschreibung der einzelnen Befehle ist gut verständlich und beschreibt sie präzise und treffend.

12.1 Befehle zum Datentransfer

Mnemonic		Description	Byte	Cycle
Data Transfer				
MOV	A,Rn	Move register to accumulator	1	1
MOV	A,direct [1]	Move direct byte to accumulator	2	1
MOV	A,@Ri	Move indirect RAM to accumulator	1	1
MOV	A,#data	Move immediate data to accumulator	2	1
MOV	Rn,A	Move accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct byte	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data 16	Load data pointer with a 16-bit constant	3	2
MOVC	A,@A+DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC	A,@A+PC	Move code byte relative to PC to accumulator	1	2
MOVX	A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX	A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX	@Ri,A	Move A to external RAM (8-bit addr.)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with accumulator	1	1
XCH	A,direct	Exchange direct byte with accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with accumulator	1	1
XCHD	A,@Ri	Exchange low-order nibble indir. RAM with A	1	1

[1] MOV A,ACC is not a valid instruction

MOV-Befehle

(siehe Kapitel 4: „Speicherorganisation und Datentransfer")

Die MOV-Befehle beeinflussen nicht die Flags im Programmstatusword PSW.

Hinter der Transport-Operation MOV steht zuerst das Ziel und dann, durch ein Komma getrennt, die Quelle.

```
MOV direkt,A
MOV A,direkt
MOV direkt,direkt
```

A = Akkumulator

direkt = Adresse, symbolisch oder absolut, im direkt adressierbaren internen RAM-Speicher.

Beispiele:
```
MOV 20,A
MOV A,P1
MOV 21,P5
```

Rn = Register R0 bis R7 der aktuellen Registerbank

```
MOV Rn,A
MOV A,Rn
MOV Rn, direkt
```

Beispiele:
```
MOV R0,A
MOV A,R7
MOV R3,45
```

Ri = Register R0 oder R1
Diese Register werden als Zeiger für die indirekte Adressierung z. B. im internen RAM-Speicher genutzt. Der Zeiger enthält die 8-Bit-Adresse.

```
MOV A,@Ri
MOV @Ri,A
MOV @Ri,direkt
```

Beispiele:
```
MOV A,@R0
MOV @R1,P1
```

@ = Kennzeichnung für indirekte Adressierung. Es folgt das Register mit der Adresse.

= Kennzeichnung für eine folgende Konstante

```
MOV A,#data
MOV DPTR,#data 16
```

data = 8-Bit-Konstante
data 16 = 16-Bit-Konstante

Beispiele:
```
MOV A,#0F
MOV DPTR,#E100
```

MOVX kennzeichnet einen Datentransfer zwischen Akku und externem Datenspeicher. Der Transfer erfolgt immer über indirekte Adressierung. Bei einer 16-Bit-Adresse erfolgt die Adressierung über den Datapointer DPTR, bei einer 8-Bit-Adresse über Ri.

```
MOVX A,@DPTR
MOVX @DPTR,A
MOVX A,@R1
```

MOVC kennzeichnet einen Datentransfer vom externen Programmspeicher zum Akku. Die Adressierung erfolgt indirekt. Die Adresse wird gebildet aus der Summe der Inhalte von Akku plus Datapointer DPTR oder aus der Summe der Inhalte von Akku plus Programmcounter PC.

```
MOVC A,@A+DPTR
MOVC A,@A+PC
```

Die in diesem Buch vorgestellten Assemblerprogramme werden mit einem Makroassembler übersetzt, der Zahlen aus verschiedenen Zahlensystemen in Maschinensprache übersetzen kann. Deshalb muss das Zahlensystem hinter der Zahl angegeben werden. Ein H hinter der Zahl steht für Hexadezimalzahlen. Weiter erlaubt der Makroassembler auch symbolische Bezeichnungen von Adressen oder Zahlen. Um eine absolute Zahl von einer symbolischen Angabe unterscheiden zu können, ist bei allen Hexzahlen, die mit einem Buchstaben beginnen, zur Kennzeichnung eine Null voranzusetzen.

Alle Buchstaben dürfen groß- oder kleingeschrieben werden.

Beispiele:

```
MOV A,0F5H
mov 0e1h,a
```

12.2 Befehle zu arithmetischen Operationen

Die Befehle zu arithmetischen und logischen Operationen beeinflussen die Flags im Programmstatuswort PSW. PSW ist ein Spezial-Funktions-Register.

Spezial-Funktions-Register PSW

PSW bitadressierbar Resetwert 00h

CY	AC	F0	RS1	RS0	OV	F1	P

Bit	Funktion
CY	Carry Flag, Übertragsbit
AC	Auxilary Carry Flag, Hilfs-Übertragsbit
F 0	Benutzerflag 0, zur freien Verfügung
RS1 RS0	Auswahl der Registerbank
0 0 0 1 1 0 1 1	Bank 0 1 2 3
OV	Overflow Flag, Überlaufbit
F1	Benutzerflag 1, zur freien Verfügung
P	Parit tätsbit

Erläuterung der Flags

Carry Flag CY

Wird bei einem Übertrag des Ergebnisses gesetzt. Der Übertrag tritt bei einer Addition auf, wenn das Ergebnis größer 8 Bit ist. Er tritt bei einer Subtraktion auf, wenn das Ergebnis negativ wird. (Borrow)

Auxiliary Flag AC

Wird bei einem Übertrag der niederwertigen vier Bits des Ergebnisses gesetzt. Wird beim Rechnen mit BCD-Zahlen benötigt.

Overflow Flag OV

Eine Kopie von Bit 7 des Ergebnisses. Wird benötigt beim Rechnen mit vorzeichenbehafteten Zahlen (Vorzeichenbit).

Parity Flag

Wird gesetzt, wenn sich eine ungerade Anzahl von 1-Signalen im Akku befindet.

Das B-Register

Das B-Register ist ein Spezial-Funktions-Register, welches bei der Multiplikation und Division benötigt wird.

Bei einer Multiplikation stehen in A und B die beiden 8-Bit-Zahlen. Nach Ausführung des Befehls befindet sich der niederwertige Teil des Ergebnisses in A und der höherwertige Teil in B. Ist der Teil in B > 0 (Ergebnis > 255), wird das Overflow Flag gesetzt.

Bei einer Division wird die 8-Bit-Zahl in A durch die 8-Bit-Zahl in B dividiert. Nach Ausführung des Befehls steht in A das Ergebnis als ganze Zahl. Der Rest steht in B.

Mnemonic		Description	Byte	Cycle
Arithmetic Operations				
ADD	A,Rn	Add register to accumulator	1	1
ADD	A,direct	Add direct byte to accumulator	2	1
ADD	A,@Ri	Add indirect RAM to accumulator	1	1
ADD	A,#data	Add immediate data to accumulator	2	1
ADDC	A,Rn	Add register to accumulator with carry flag	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1	1
ADDC	A,#data	Add immediate data to A with carry flag	2	1
SUBB	A,Rn	Subtract register from A with borrow	1	1
SUBB	A,direct	Subtract direct byte from A with borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A with borrow	1	1
SUBB	A,#data	Subtract immediate data from A with borrow	2	1
INC	A	Increment accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1

Mnemonic		Description	Byte	Cycle
Arithmetic Operations (cont'd)				
DEC	A	Decrement accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment data pointer	1	2
MUL	AB	Multiply A and B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal adjust accumulator	1	1

12.3 Befehle zu logischen Operationen

Mnemonic		Description	Byte	Cycle
Logic Operations				
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A,direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive OR immediate data to direct byte	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate accumulator right through carry	1	1
SWAP	A	Swap nibbles within the accumulator	1	1

12.4 Befehle zur Programm- und Maschinensteuerung

Mnemonic		Description	Byte	Cycle
Program and Machine Control				
ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long jump	3	2
SJMP	rel	Short jump (relative addr.)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if carry flag is not set	2	2
JB	bit,rel	Jump if direct bit is set	3	2
JNB	bit,rel	Jump if direct bit is not set	3	2
JBC	bit,rel	Jump if direct bit is set and clear bit	3	2
CJNE	A,direct,rel	Compare direct byte to A and jump if not equal	3	2
CJNE	A,#data,rel	Compare immediate to A and jump if not equal	3	2
CJNE	Rn,#data,rel	Compare immed. to reg. and jump if not equal	3	2
CJNE	@Ri,#data,rel	Compare immed. to ind. and jump if not equal	3	2
DJNZ	Rn,rel	Decrement register and jump if not zero	2	2
DJNZ	direct,rel	Decrement direct byte and jump if not zero	3	2
NOP		No operation	1	1

Sprungbefehle

Der Befehl ACALL beinhaltet eine 11-Bit-Adresse. Diese wird rechtsbündig im Programm Counter eingesetzt. Die oberen 5 Bit im PC bleiben erhalten.

Der Befehl LCALL beinhaltet eine 16-bit-Adresse. Das gleiche gilt für AJMP und LJMP.

Der Befehl SJMP enthält eine 8-Bit-Adresse. Damit sind relative Sprünge im Bereich von +127 bis −128 möglich.

Da der Programmierer bei Verwendung eines Makroassemblers mit symbolischen Sprungmarken arbeiten kann, berechnet der Makroassembler die absoluten Sprungadressen. Falls die Sprungweite überschritten wird, macht der Makroassembler ihn darauf aufmerksam. Zum Glück entfällt damit das Berechnen von Sprungweiten oder Sprungadressen.

12.5 Befehle zur Bitverarbeitung

Der Controller beinhaltet zur Bitverarbeitung einen Boole'schen Prozessor. Dieser verwendet das Carry C als „Akkumulator". „Direkt Bit" ist ein Bit im bitadressierbaren internen RAM-Speicher. (Bit 00h bis 7Fh ab Adresse 20h) Ein Makroassembler erlaubt auch eine symbolische Bezeichnung einzelner Bits. Einzelne Bits in den Spezial-Funktions-Registern lassen sich ebenfalls mit ihrer symbolischen Bezeichnung einsetzen.

In nachfolgender Liste sind die Sprungbefehle der „Programm- und Maschinensteuerung" wiederholt, die zur Bitverarbeitung benötigt werden.

Mnemonic		Description	Byte	Cycle
Boolean Variable Manipulation				
CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,/bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,/bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if carry flag is not set	2	2
JB	bit,rel	Jump if direct bit is set	3	2
JNB	bit,rel	Jump if direct bit is not set	3	2
JBC	bit,rel	Jump if direct bit is set and clear bit	3	2

Liste aller Befehle

Mnemonic		Description	Byte	Cycle
Data Transfer				
MOV	A,Rn	Move register to accumulator	1	1
MOV	A,direct [1)]	Move direct byte to accumulator	2	1
MOV	A,@Ri	Move indirect RAM to accumulator	1	1
MOV	A,#data	Move immediate data to accumulator	2	1
MOV	Rn,A	Move accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move accumulator to direct byte	2	1

[1)] MOV A,ACC is not a valid instruction

Mnemonic		Description	Byte	Cycle
Data Transfer (cont'd)				
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct byte	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data 16	Load data pointer with a 16-bit constant	3	2
MOVC	A,@A+DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC	A,@A+PC	Move code byte relative to PC to accumulator	1	2
MOVX	A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX	A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX	@Ri,A	Move A to external RAM (8-bit addr.)	1	2
MOVX	@DPTR,A	Move A to external RAM (16-bit addr.)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte from stack	2	2
XCH	A,Rn	Exchange register with accumulator	1	1
XCH	A,direct	Exchange direct byte with accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with accumulator	1	1
XCHD	A,@Ri	Exchange low-order nibble indir. RAM with A	1	1

Mnemonic		Description	Byte	Cycl
Arithmetic Operations				
ADD	A,Rn	Add register to accumulator	1	1
ADD	A,direct	Add direct byte to accumulator	2	1
ADD	A,@Ri	Add indirect RAM to accumulator	1	1
ADD	A,#data	Add immediate data to accumulator	2	1
ADDC	A,Rn	Add register to accumulator with carry flag	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1	1
ADDC	A,#data	Add immediate data to A with carry flag	2	1
SUBB	A,Rn	Subtract register from A with borrow	1	1
SUBB	A,direct	Subtract direct byte from A with borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A with borrow	1	1
SUBB	A,#data	Subtract immediate data from A with borrow	2	1
INC	A	Increment accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1

Mnemonic		Description	Byte	Cycle
Arithmetic Operations (cont'd)				
DEC	A	Decrement accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment data pointer	1	2
MUL	AB	Multiply A and B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal adjust accumulator	1	1

Mnemonic		Description	Byte	Cycle
Logic Operations				
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A,direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive OR immediate data to direct byte	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate accumulator right through carry	1	1
SWAP	A	Swap nibbles within the accumulator	1	1

Mnemonic		Description	Byte	Cycle
Program and Machine Control				
ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long jump	3	2
SJMP	rel	Short jump (relative addr.)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if carry flag is not set	2	2
JB	bit,rel	Jump if direct bit is set	3	2
JNB	bit,rel	Jump if direct bit is not set	3	2
JBC	bit,rel	Jump if direct bit is set and clear bit	3	2
CJNE	A,direct,rel	Compare direct byte to A and jump if not equal	3	2
CJNE	A,#data,rel	Compare immediate to A and jump if not equal	3	2
CJNE	Rn,#data,rel	Compare immed. to reg. and jump if not equal	3	2
CJNE	@Ri,#data,rel	Compare immed. to ind. and jump if not equal	3	2
DJNZ	Rn,rel	Decrement register and jump if not zero	2	2
DJNZ	direct,rel	Decrement direct byte and jump if not zero	3	2
NOP		No operation	1	1

Mnemonic		Description	Byte	Cycle
Boolean Variable Manipulation				
CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,/bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,/bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2