8051-es mikrokontroller- és assembler tanfolyam

2. rész: Az első 8051-es utasítások

A bevezetés után konkrét tudnivalókra kerül sor: Bemutatjuk a 8051-es mikrokontroller-család első utasításait. Egyidejűleg foglalkoznunk kell a lehetséges címzési módokkal is. Mindent egybevetve, merész vállalkozásunk nem könnyű, ám amit az első pillanatban még nem értünk az a tanfolyam diszkettjén található programpéldákkal végzett próbálkozások és tesztek útján gyorsan világossá válik.

A Monitor-EPROM és az assembler teljes dokumentációja a DOC-fájlokban található. A diszkett valamennyi programpéldát is tartalmazza, úgyhogy saját ismereteink elmélyítése céljából a programok változtathatók és bővíthetők. Mindaz, amire egyébként szükségünk van, csupán egy egyszerű (ASCII) szövegszerkesztő. Hogy ez hogyan kapcsolható be az ugyancsak rendelkezésre álló MENUE-programba, szintén le van írva a diszketten. Ennél könnyebb tulajdonképpen már nem is lehet ...

Egy bővítőpanelt is bemutatunk a Compuboard-hoz, az összes szükséges hardverrel és a külső csatlakozásokhoz megfelelő csatlakozóhüvelyekkel (V24, MIDI) együtt. Így a tanfolyam valamennyi programpéldája a legegyszerűbb módon átültethető a gyakorlatba is.

ARITHMETIC OPERATIONS

Mnemonic		Description	Byte	Cyc
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	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 w Borrow	1	1
SUBB	A,#data	Subtract immed. data from A w 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
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 & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

LOGICAL OPERATIONS

Mnemonic		Destination	Byte	Cyc
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 A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Right through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

DATA TRANSFER

Mnemonic		Description	Byte	Cyc
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,direct	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	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,#data16	Load Data Pointer with a 16-bit constant	3	2

DATA TRANSFER (cont.)

Mnemonic		Description	Byte	Cyc
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A,@A+PC	Move Code byte relative to PC to A	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 A	1	1
XCHD	A,@Ri	Exchange low-order Digit ind. RAM w A	1	1

BOOLEAN VARIABLE MANIPULATION

Mnemonic		Description	Byte	Cyc
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

PROGRAM AND MACHINE CONTROL

Mnemonic		Description	Byte	Cyc
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 No Carry flag	2	2
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A,#data,rel	Comp. immed. to A & Jump if Not Equal	3	2
CJNE	Rn,#data,rel	Comp. immed. to reg. & Jump if Not Equal	3	2
CJNE	@Ri,#data,rel	Comp. immed. to ind. & Jump if Not Equal	3	2
DJNZ	Rn,rel	Decrement register & Jump if Not Zero	2	2
DJNZ	direct,rel	Decrement direct & Jump if Not Zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn	Working register R0-R7
direct	128 internal RAM locations, any I/O port, control or status register
@Ri	Indirect internal RAM location addressed by register R0 or R1
#data	8-bit constant included in instruction
#data16	16-bit constant included as bytes 2 & 3 of instruction
bit	128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

addr16	Destination address for LCALL & LJMP may be anywhere within the 64-Kilobyte program memory address space.
addr11	Destination address for ACALL & AJMP will be within the same 2-Kilobyte page of program memory as the first byte of the following instruction.
rel	SJMP and all conditional jumps include an 8-bit offset byte. Range is +127 -128 bytes relative to first byte of the following instruction.

A 8051-es Mikrokontroller utasításai

Mindazok az utasítások, amelyeket a 8051 végre tud hajtani, az *1. ábrán* fel vannak sorolva. Először csak néhány olyan egyszerű, de fontos utasítást mutatunk be, amelyek a *2. ábrán* látható programlista megértését lehetővé teszik.

A listában a **LINE** oszlop mutatja a forrásszöveg sorszámát. A **LOC** (Location) oszlopban az található, hogy a következő byte-ok a programmemóriában hol kerülnek tárolásra. Az OBJ oszlopban a generált tárgykód-byte-ok szerepelnek. A **T** oszlopban az egyes utasítások végrehajtásának mikroszekundumban kifejezett időtartama (12 MHz-es órafrekvencia esetén), a **SOURCE** oszlopban pedig a forráskód szerepel a kommentárral együtt.

Az assembler itt (a BSP2.A51 fájlból) a forrásszöveget ismétli, így a sorszám alapján pontosan felismerhető, melyik sor milyen byte-okat hoz létre. Így a hibák jól lokalizálhatók. A *6. ábrán* egyébként egy program forrásszövegét mutatjuk be. A forrásszöveg formátuma a tanfolyam diszkettjén le van írva. A forrásszöveg bármely szövegszerkesztővel előállítható. A programdiszketten BSP2.A51-ként szerepel, a program indításához pedig a BSP2.DOC-ban található útmutató.

A program indítása céljából a Compuboard-ot a PC V24-es interfészével kell összekötni. Ezután a Compuboard Reset-nyomógombját meg kell nyomni, ezzel a monitorprogram is kész az áttöltésre. A MOV-utasítással a különböző változatok közül a 8051 aktuális címzési módjait hívjuk be. A program végén aztán bemutatjuk az első minialkalmazást, nevezetesen a LED-ek villogtatását és az egyszerű hanggenerálást. Egy kis módosítással egy olyan „Original-english tea-timer"-hez (eredeti angol teafőző vekkerhez) jutunk, mely pontosan 2 perc és 50 másodperc eltelte után jelzőhangot bocsát ki.

2

```
****** LISTING of EASM51 (BSP2) ******
LINE LOC  OBJ          T        SOURCE
   1 0000                ; ****** DATEI BSP2.A51 ****************************************
   2 0000                ;
   3 0000                ACC     EQU   0E0H          ; SFR Akkumulator Adresse ist 0E0H
   4 0000                P1      EQU   090H          ; SFR PORT1 Adresse ist 090H
   5 0000                xR3     EQU   3             ; Adresse vom Register R3 in Bank 0
   6 0000                WERT    EQU   100
   7 0000                ;
   8 0000                        ORG   4100H         ; Programm steht spaeter ab 4100H
   9 4100 90 41 44 [2] START   MOV   DPTR,#txt1
  10 4103 31 49    [2]         ACALL STXT
  11 4105 74 00    [1]         MOV   A,#0          ; Initialisiere einige Register
  12 4107 78 FF    [1]         MOV   R0,#255       ; ist OFFH hexadezimal
  13 4109 79 0A    [1]         MOV   R1,#10        ; 10 dezimal ist OAH hexadezimal
  14 410B 7A 10    [1]         MOV   R2,#10H       ; 10H hexadezimal ist 16 dezimal
  15 410D 7B 64    [1]         MOV   R3,#WERT      ; vgl. mit EQU Befehl oben
  16 410F 90 40 00 [2]         MOV   DPTR,#04000H  ; lade mit 16-bit Konstante
  17 4112 31 4F    [2]         ACALL SNAP          ; Ausgabe erster Schnappschuß
  18 4114 74 14    [1]         MOV   A,#20         ; Adressierung: immediate
  19 4116 F9       [1]         MOV   R1,A          ; Adressierung: register,A
  20 4117 31 4F    [2]         ACALL SNAP
  21 4119 75 E0 12 [2]         MOV   ACC,#12H      ; Adressierung: direct,immediate
  22 411C F5 03    [1]         MOV   xR3,A         ; Adressierung: direct,A
  23 411E 31 4F    [2]         ACALL SNAP
  24 4120 78 02    [1]         MOV   R0,#2         ; Adressierung: register,immediate
  25 4122 C6       [1]         XCH   A,@R0         ; Adressierung: A,indirect
  26 4123 75 90 55 [2]         MOV   P1,#01010101B ; binaere Konstante an Port senden
  27 4126 31 4F    [2]         ACALL SNAP
  28 4128 90 41 2B [2]         MOV   DPTR,#adr1    ; Zeige auf den MOV  A,#2 Befehl
  29 412B 74 02    [1] adr1    MOV   A,#2          ; Adressierungs-Offset
  30 412D 93       [2]         MOVC  A,@A+DPTR     ; Adressierung: A,code-byte relative
  31 412E 31 4F    [2]         ACALL SNAP
  32 4130 E0       [2]         MOVX  A,@DPTR       ; Adressierung: A,external RAM
  33 4131 A9 02    [2]         MOV   R1,2          ; Raetselfrage
  34 4133 31 4F    [2]         ACALL SNAP
  35 4135 90 00 00 [2]         MOV   DPTR,#0       ; Adressierung; DPTR,16-bit immediate
  36 4138 74 F2    [1]         MOV   A,#0F2H       ; externer Datensp. Adr 0 wird 0F2H
  37 413A F0       [2]         MOVX  @DPTR,A
  38 413B 74 00    [1]         MOV   A,#0          ; Programm Speicher Offset
  39 413D 93       [2]         MOVC  A,@A+DPTR     ; Adressierung: A,code-byte relative
  40 413E 31 4F    [2]         ACALL SNAP
  41 4140 E0       [2]         MOVX  A,@DPTR       ; Adressierung: A,external RAM
  42 4141 31 4F    [2]         ACALL SNAP
  43 4143 22       [2]         RET                 ; zurueck zum MONITOR
  44 4144 42 53 50     txt1    DB 'BSP2',0
          32 00
  45 4149               ;
  46 4149               ;  MONITOR INTERFACE
  47 4149               ccSTXT  EQU  2      ; MONITOR Kommando um Text zu senden
  48 4149               ccSNAP  EQU  020H   ; MONITOR Kommando um Schnappschuss auszugeben
  49 4149               COMMAND EQU  030H   ; MONITOR Kommando Speicherstelle
  50 4149               MON     EQU  0200H  ; MONITOR Einsprungadresse
  51 4149               ;
  52 4149 75 30 02 [2] STXT   MOV COMMAND,#ccSTXT  ; MONITOR Kommando setzen
  53 414C 02 02 00 [2]        LJMP MON             ; zum MONITOR springen (RET von da)
  54 414F 75 30 20 [2] SNAP   MOV COMMAND,#ccSNAP  ; MONITOR Kommando setzen
  55 4152 12 02 00 [2]        LCALL MON            ; Monitor aufrufen
  56 4155 22       [2]        RET                  ; zurueck zum Aufrufer
  57 4156                     END
********** SYMBOLTABLE (13 symbols) **********
      ACC :00E0       P1 :0090       xR3 :0003      WERT :0064
    START :4100     adr1 :412B      txt1 :4144    ccSTXT :0002
   ccSNAP :0020  COMMAND :0030       MON :0200      STXT :4149
     SNAP :414F
```

2. ábra. A 8051 valamennyi címzésfajtát szemléltető program

Ábrafeliratok:

2. ábra.
- SFR Akkumulator Adresse is OEOH = az SFR akkumulátor címe OEOH
- SFR PORT1 Adresse ist 090H = az SFR PORT1 címe 090H
- Adresse von Register R3 in Bank 0 = az R3 regiszter címe a 0. bankban
- Programm steht spaeter ab 4100H = a program később, 4100H-tól kezdődik
- Initialisiere einige Register = néhány regiszter inicializálása
- ist OFFH hexadezimal = OFFH hexadecimális
- 10 dezimal ist OAH hexadezimal = 10 decimális = OAH hexadecimális
- 10H hexadezimal ist 16 dezimal 10H hexadecimális = 16 decimális
- vg1. mit EQU Befehl oben = vesd össze a fenti EQU utasítással
- lade mit 16-bit Konstante = töltés 16-bites konstanssal
- Ausgabe erster Schnappschuß = első kapáslövés kiadása
- Adressierung: immediate = konstans címzés
- Adressierung: register, A = regiszter/akkumulátorcímzés
- Adressierung: direct, immediate = közvetlen/konstans címzés
- Adressierung: register, immediate = regiszter/konstans címzés
- Adressierung: A, indirect- akkumulátor/közvetett címzés
- binaere Konstante an Port senden = bináris konstans küldése a Port-ra
- Zeige auf den MOV A, #2 Befehl = címkéhez tartozó utasításcím betöltése

Alprogramok

Egy program rendszerint kisebb alprogramokból áll. Az egyszer megírt alprogram a későbbiekben többször újra felhasználható. BSP2 programunkban például szerepel egy STXT (küldj szöveget) elnevezésű program, amely a V24-interfész útján adja ki a szöveget. A 8051-nél az alprogramok a következő két utasítással hívhatók be:

```
ACALL addr11
LCALL addr16
```

- Adressierungs-Offset = címzés-offszet
- Adressierung: A, code-byte relative = akkumulátor code-byte relatív címzés
- Adressierung: A, external RAM = akkumulátor/külső RAM címzés
- Raetselfrage = találós kérdés
- Adressierung; DPTR, 126-bit immediate = DPTR/16-bites konstans címzés
- externer Datensp. Adr. 0 wird OF2H = a külső adattároló kezdőcíme OF2H lesz
- Program Speicher Offset = program memória offszet
- Adressierung: A, code-byte relative = akkumulátor/code-byte relatív címzés
- Adressierung: A, external RAM akkumulátor/külső RAM címzés
- zurueck zum MONITOR = vissza a MONITOR-hoz
- MONITOR Kommando um Text zu senden = szöveg küldésére vonatkozó MONITOR parancs
- MONITOR Kommando um Schnappschuss auszugeben = kapáslövés kiadására vonatkozó MONITOR parancs
- MONITOR Kommando Speicherstelle = MONITOR parancs tárolási címe
- MONITOR Einsprungadresse = MONITOR belépési címe
- MONITOR Kommando setzen = MONITOR parancs beírása
- zum MONITOR springen (RET von da) = ugrás a MONITOR-ra (visszatérés RET-tel)
- MONITOR Kommando setzen = MONITOR parancs beírása
- MONITOR aufrufen = MONITOR behívása
- zurueck zum Aufrufer = vissza a hívó programba

3

```
BSP2

 A  B PSW  SP DPTR R0 R1 R2 R3 R4 R5 R6 R7
00 00  00  0D 4000 FF 0A 10 64 00 00 00 00   Nach Zeile 17

14 00  00  0D 4000 FF 14 10 64 00 00 00 00   Nach Zeile 20

12 00  00  0D 4000 FF 14 10 12 00 00 00 00   Nach Zeile 23

10 00  01  0D 4000 02 14 12 12 00 00 00 00   Nach Zeile 27

93 00  00  0D 412B 02 14 12 12 00 00 00 00   Nach Zeile 31

74 00  00  0D 412B 02 12 12 12 00 00 00 00   Nach Zeile 34

02 00  01  0D 0000 02 12 12 12 00 00 00 00   Nach Zeile 40

F2 00  01  0D 0000 02 12 12 12 00 00 00 00   Nach Zeile 42
```

3. ábra. A 2. ábrán bemutatott program outputja lehetővé teszi valamennyi utasítás hatásának követését. A fontosabb regiszterek tartalma hexadecimális alakban szerepel

A két utasítás azonos hatású, nevezetesen a megadott címen (addr11, illetve addr16) található alprogramot indítja. Az alprogram végrehajtása után a processzor a CALL-t követő utasítással tovább folytatja a feldolgozást.

■ Ha az alprogram címe ugyanabban a 2k-s blokkban található, mint maga a CALL-utasítás, akkor az ACALL (Abszolut-CALL) utasítást kell használni. A 16-bites címek felső 5 bitjének tehát egymással meg kell egyeznie. Ha az **ACALL** utasítás tehát például a 9A12H címen áll, akkor a 9800H és a 9FFFH közötti valamennyi cím elérhető. Az ACALL utasításhoz két byte szükséges.

■ Az **LCALL** (Long-CALL) utasítással a 64k-s programmemória bármely címe alatt található alprogramra át lehet ugrani. Igen nagy programok esetében tehát a „távoli" alprogramok az LCALL-lal hívhatók be. Az LCALL-hoz három byte szükséges, nem olyan helytakarékos tehát, mint az ACALL. A közelben tárolt alprogramot ezért ACALL-lal célszerű behívni. Ha a megadott cím nem érhető el, akkor az assembler hibajelzést ad. Az LCALL használatára példa a *2. ábrán* az 55. sorban található. Itt egyébként (a 10. sorban található utasítással ellentétben) egy ACALL nem volna elegendő, mert a behívott, 0200H című program a mi (4100H-tól induló) programunktól túl messze esik.

■ Egy alprogram végét a **RET** (*Ret*urn) utasítással kell kiváltani. Ez az utasítás a program végrehajtásának a hívó programhoz való visszatérését váltja ki. A 43. sorban például a RET utasítással a teljes példaprogram kerül befejezésre és a monitorprogram végrehajtása folytatódik.

Ugróutasítások

Az ugróutasítások közül mindenekelőtt a következő utasításokat kell bemutatni:

```
SJMP rel
AJMP addr11
LJMP addr16
```

Ezek a programnak a megadot címnél (rel, addr11 és addr16) történő folytatását váltják ki. Itt úgynevezett feltétel nélküli ugrásokról van szó, mert ugrásra minden esetben sor kerül. A **JMP** *(JuMP)* elé tett A és L betűk (**AJMP** ill. **LJMP**) ugyanazt jelentik, mint a Call esetében. Van még ezenkívül egy Short-Jump **(SJMP)** utasítás is, amellyel csak a –128 és a +128 byte közötti relatív címek érhetők el. Ez az utasítás néhány byte átugrására szolgál. Az ugróutasításra példa a *2. ábra* 53. sora.

A feltétel nélküli ugrások mellett léteznek feltételes ugrások is, melyeknél feltételtől függ az, hogy az ugrás végrehajtásra kerül-e vagy egyszerűen a következő utasításnál folytatódik tovább a feldolgozás. A 8051 feltételes ugrásai kizárólag beszűkített ugrási tartományon belüli rövid ugrások, ezeket a tanfolyam következő részében tárgyaljuk.

Címek megadása

Az alprogram behívások célcímeit assembler-programozás esetén ugrócímkék, úgynevezett **LABEL**-ek segítségével lehet megadni (lásd 3. ábra.

– nach Zeile ... =
... sor után

EASM51.DOC). Az 54. sorban a **SNAP** (*SNAP*shot = kapáslövés) elnevezésű címkéhez az ott szereplő utasítás címét, azaz a 414FH értéket rendeltük hozzá. Az 50. sorban az **EQU** (*EQU*ATE = egyenlővé tenni) assembler-utasítással a **MON** (*MON*TOR) elnevezésű címkéhez a 0200H értéket rendeltük hozzá, ez ugyanis a monitor-alprogram indítócíme.

Az

```
LCALL MON
```

utasítás tehát a monitor alprogram behívását váltja ki. Az

```
ACALL SNAP
```

utasítás hatására megtörténik a „kapáslövés", azaz a 414FH címen induló alprogram behívása. Ez az alprogram a fontosabb processzorregiszterek tartalmát a V24-interfészen át hexadecimális írásmódban adja ki. Így a különböző utasítások hatása könnyen ellenőrizhető. A SNAP tehát számos tesztelési célra igen hasznos.

Az assembler-program olvashatóságát növeli az informatív nevekkel *(LABEL)* ellátott alprogramok használata. Példa erre a 49., 52. és 54. sorokban használt, **COMMAND** névvel ellátott LABEL. A LABEL-ek értékei (címei) abból a szimbólumtáblázatból vehetők ki, amelyet az EASM51 assembler a listához fűz hozzá. Számos assembler egyébként csak maximálisan hat betűből álló címkéket fogad el. Ezért az assembler-programozás során gyakran rövidített írásmódot szoktak használni, például a „V24-command" helyett „V24COM"-ot vagy a „Send Caharacter" helyett SNDCHR-t írnak. A mi EASM51-es assemblerünk maximálisan nyolc jelből álló Label-eket fogad el.

Címzésfajták

A *2. ábrán* bemutatott BSP2 program egyetlen feladata a 8051-es sorozat különböző címzésfajtáinak szemléltetése. Néhány bevezető assembler-utasítás és kommentár után maga a program a 9. sorban kezdődik. Legelőször néhány regiszter és az A akkumulátor értékadására kerül sor. A 17. sor a SNAP alprogramot hívja be, mely a V24 útján kiadja az első processzor-kapáslövést. A

program futásának ily módon (a regiszter-tartalmak időnkénti kiírásával) történő áttekinthetővé tétele a hibakeresés során fontos segédeszköz. A Compuboard egyébként a tanfolyam 1. részében ismertetett standardkonfigurációban üzemel.

A BSP2 program kimenetét a *3. ábrán* mutatjuk be. A 11-től 42-ig terjedő sorokban található utasítások hatása a közbeiktatott kiírások útján tehát minden esetben pontosan követhető. A 45. és az azt követő sorok csak a monitorprogrammal való logikai kapcsolatot állítják elő (szoftver-interfész) és egyelőre nem fontosak.

Térjünk azonban most rá az MOV-utasításra és a címzésfajtákra. A 8051, mint 8-bites mikrokontoller, főleg arra van felkészítve, hogy egy byte-os feldolgozási egységként működjék. Egy byte-nak egy memóriarekeszből (például program-, adat- vagy belső memóriából) egy regiszterbe vagy az output vezetékekre (PORT-ok) történő eljuttatását a MOV (MOVe) utasítással kezdeményezhetjük. A mozgatni vagy megváltoztatni kívánt byte-ot operandusként jelöljük ki. Attól függően, hogy hol van ez az operandus és hova kell átvinni, különböző címzésfajták alkalmazása szükséges. A MOV utasítás általános alakja a következő:

MOV cél, forrás

Célként például az A akkumulátor, az R0-tól R7-ig terjedő regiszterek vagy a különböző memóriarekeszek jönnek szóba. Forrásként az előbbiek mellett konstansok is szerepelhetnek. Mindazonáltal nem minden kombináció lehetséges, hanem csak azok, amelyek az utasításlistában *(1. ábra)* szerepelnek.

A *2. ábrán* látható példák kapcsán most végigmegyünk a különböző utasításfajtákon. Az úgynevezett bit-címzést a processzor-Flag-ekkel együtt csak a tanfolyam harmadik részében mutatjuk be.

■ Regisztercímzés

Forrás (és cél) az akkumulátor vagy az R0-tól R7-ig terjedő, valamelyik regiszter lehet. A 11. sorban célként az akkumulátort, a 13. sorban az R1 regisztert használjuk. Regisztercímzésnél mindig a mindenkor kiválasztott bankban (tanfolymunkon rendszerint 0) levő regiszter kerül behívásra.

4 *Tabelle der Spezial Funktion Register*

```
Symbol  ADR  bit  binval    Kommentar  Name

ACC     0E0H  *  00000000B             Akkumulator
B       0F0H  *  00000000B             Hilfs-Akkumulator
PSW     0D0H  *  00000000B             Programm Status Wort
SP      081H     00000111B             Stapelzeiger (Stack-pointer)
DPTR                      16 Bit       Datenzeiger (DAta-Pointer)
  DPL   082H     00000000B             DPTR niederwertiges (LOW) Byte
  DPH   083H     00000000B             DPTR hoeherwertiges Byte
P0      080H  *  11111111B             Port0 bzw. Adress/Datenbus
P1      090H  *  11111111B             Port1
P2      0A0H  *  11111111B             Port2 bzw. Adressbus High-Byte
P3      0B0H  *  11111111B             Port3
IP      0B8H  *  xxx00000B  8051       Interrupt Prioritaets-register
                 xx000000B  8052
IE      0A8H  *  0xx00000B  8051       Interrupt Einschalt (Enable) Reg.
                 0x000000B  8052
TMOD    089H     00000000B             Timer-Modus Register
TCON    088H  *  00000000B             Timer-Control
T2CON   0C8H  *  00000000B  nur 8052   Timer2-Control
TH0     08CH     00000000B             Timer0 Hoeherwertiges (HIGH) Byte
TL0     08AH     00000000B                  0 niederwertiges (LOW) Byte
TH1     08DH     00000000B             Timer1 Hoeherwertiges Byte
TL1     08BH     00000000B                  1 niederwertiges (LOW) Byte
TH2     0CDH     00000000B  nur 8052   Timer2 Hoeherwertiges Byte
TL2     0CCH     00000000B  nur 8052        1 niederwertiges (LOW) Byte
RCAP2H  0CBH     00000000B  nur 8052   Capture-Register HIGH BYTE
RCAP2L  0CAH     00000000B  nur 8052   Capture-Register LOW Byte
SCON    098H  *  00000000B             Serial-Control  serielle Kontrolle
SBUF    099H     xxxxxxxxB             Serial Buffer   serieller Puffer
PCON    087H     0xxxxxxxB  HMOS       Processor Control
                 0xxx0000B  CHMOS

ADR:    Adresse unter der das SFR angesprochen wird.
bit:    * falls Bit-adressierbares SFR
binval: Inhalt (binaer) nach dem einschalten bzw. Reset., x=undefiniert.
```

4. ábra. Ez a táblázat a 8051 speciális funkcióregisztereinek jelöléseiről és címeiről nyújt felvilágosítást

■ Konstans (immediate) címzés

Ha (forrás) operandusként konstans számot akarunk választani, akkor a konstans címzést használjuk. Az assembler ezt a konstans elé tett kettőskereszttől ismeri fel. Az ezt követő konstans lehet decimális (12. sor), hexadecimális (14. sor, a konstans után írt H-val jelölve), bináris (26. sor, utána írt B-vel jelölve) vagy szimbólikus név is (LABEL), ahogy az a 15. sorban szerepel. Hogy az assembler a Label-t a hexadecimális számtól meg tudja különböztetni, olyan hexadecimális számok elé, amelyek betűvel kezdődnek, nullát kell írni. (E0H pl. egy címke, OE0H pedig hexadecimális szám, értéke decimálisan 224).

Így most már megérthetjük a 11-től 15-ig terjedő sorokban található utasítások hatását. Ezek a mindenkor megadott regisztert a bejegyzett értékkel töltik fel. Most már érdemes tüzetesebben is megnézni a *3. ábrán* bemutatott outputot.

■ Közvetlen (direct) címzés

Közvetlen címzés útján hozzá lehet férni a belső RAM alsó 128 byte-jához és a speciális funkcióregiszterekhez (SFR). Ha a megadott cím 128-nál kisebb, akkor a hozzáférés a belső RAM-hoz történik, egyébként pedig egy SFR-hez. Az akkumulátor SFR-ként is behívható a OE0H cím alatt. A 3. sorral adjuk ennek a memóriacímnek (mint SFR-nek) az ACC nevet. A 21. sor egy konstanst tölt be az akkumulátorba. Az utasításhoz azonban ez esetben 3 byte szükséges.

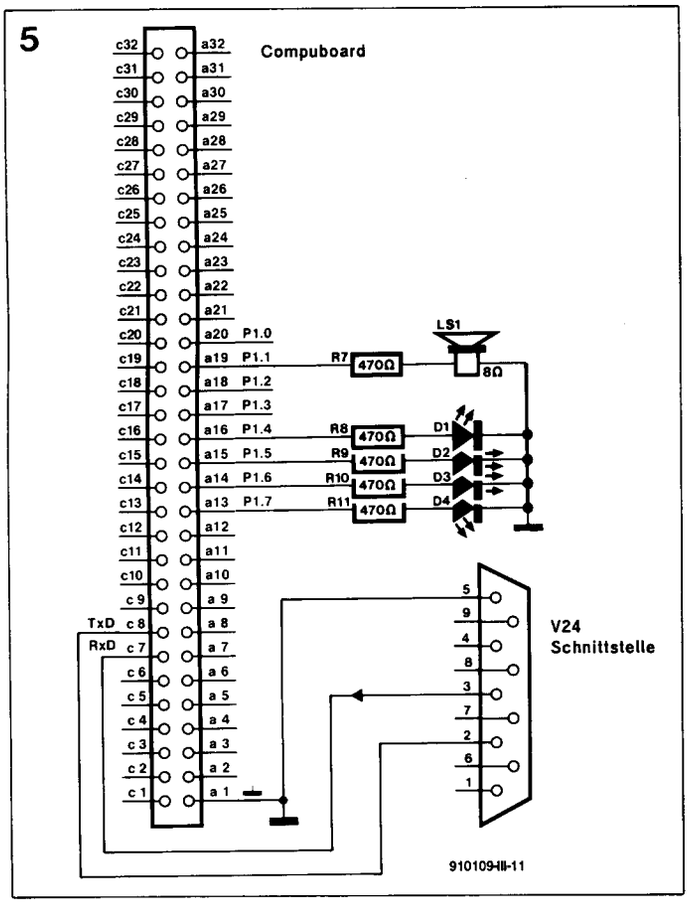
A Compuboard-on a 8051 P1 Port-ja ki van vezetve. Belsőleg ennek az SFR-címe O9OH (4. sor). A 26. sorban található utasítás kiadja a P1 Port-ra a 01010101B bit-mintát. Előállítottuk tehát első kapcsolatunkat a külvilággal!

A teljes 64K-s adat- és programtároló címezhetőségét biztosítja a **DPTR** (*DATA-POINTER*) 16-bites adatpointer. Ezt két 8 bites SFR valósítja meg: címük DPL = 082H (DPTR Low Byte, azaz alsó byte) és DPH = 083H (DPTR High Byte, azaz felső byte). A DPTR-nek egy 16 bites értékkel való betölthetősége céljából ugyancsak egy MOV-utasítás áll rendelkezésre. Ezt a 9. sorban használjuk a pointernek a szöveg első bytejára, txt1-re való állítása céljából. A 4. ábrán a 8051-es sorozat valamennyi SFR-je látható, címével együtt felsorolva.

4. ábra.
- Hilfs-Akkumulator = segédakkumulátor
- Programm Status Wort = program állapot-szó
- Stapelzeiger (Stack-pointer) = verem-mutató (Stack-pointer)
- Datenzeiger (Data-pointer) = adatmutató (Data-pointer)
- DPTR niederwertiges (LOW) Byte = DPTR alsó (low) byte-ja
- DPTR hoeherwertiges Byte = DPTR felső (high) byte-ja
- Port0, bzw. Adress/Datenbus = Port0, ill. cím/adatbusz
- Port2 bzw. Adressbus High-Byte = Port2. ill. címbusz felső byte-ja
- Interrupt Prioritaets-register = megszakítási piroritás-regiszter
- Interrupt Einschalt (Enable) Reg. = megszakítás engedélyezési (Enable) regiszter
- Timer-Modus Register = Timer-módus regiszter
- Timer0 Hoehewertiges (HIGH) Byte = Timer0 felső (high) byte
- 0 niederwertiges (LOW) Byte = Timer0 alsó (low) byte és ugyanúgy a következő négy sorban 0 helyett 1-et ill. 2-t írva
- Serial-Control serielle Kontrolle = Serial-Control (soros vezérlés)
- Serial Buffer serieller Puffer = Serial Buffer (soros puffer)
- Adresse unter der das SFR engesprochen virdt = az SFR hivatkozási címe
- falls Bit-adressierbares SFR = ha SFR bit-címezhető
- Inhalt (binaer) nach dem einschalten bzw. Reset., x = undefiniert = (Bináris) tartalom a bekapcsolás, ill. Reset után, x = nem definiált

■ Közvetett (indirect) címzés

A **@R0**-val vagy **@R1**-gyel jelzett közvetett címzésnél a megadott R0 és R1 regiszterekben a belső RAM alkalmazandó byte-jának címe áll. Az R2-től R7-ig terjedő regiszterek közvetett címzésre sajnos nem használhatók. Ha pl. R0-ban a 43H érték áll, akkor @R0-lal a belső



5. ábra. LED-didóda- és hangszóró-output vezérlése a 8051-es P1 Portján keresztül

RAM 43H című byte-ja kerül lehívásra. Példaként a 25. sort említjük meg. R0 itt a belső RAM (24. sorban kijelölt) 2. memóriacellájára mutat, ahol azonban (lásd az 1. rész 5. ábráját) pontosan a 0 BANK R2 regisztere is tárolásra került. Így az utasítás által valójában az R2 regiszterben álló érték változik meg. Az XCH utasítás a cél- és a forrás-byte-ot felcseréli.

A közvetlen címzéssel ellentétben a 127 feletti címek itt nem vezetnek SFR-címzéshez, hanem a mikrokontroller belső RAM-jának felső 128 Byte-jához nyúlnak. Sőt, ezek a byte-ok kizárólag közvetett címzés segítségével érhetők el.

■ A programmemória címzése

A programmemóriából, mely rendszerint egy ROM vagy EPROM, a processzor csak olvasni tud. Erre szolgál a **MOVC** utasítás, melynek célja egyébként mindig az akkumulátor. Az utasítás nevének végén a C a Code-Memory-t jelenti. A tényleges címet az akkumulátor tartalmának a DPTR adatpointerhez való hozzáadásával (MOVC A, @A+DPTR), illetve az utasításszámláló aktuális állásához való hozzáadásával (MOVC A, @A+PC) kell képezni. A 39. sorban a tényleges cím 0, mert a DPTR és az A is egyaránt 0-t tartalmaz.

5. ábra.
– V24 Schnittstelle = V24-es interfész

A programmemóriában fix táblázatok vagy szövegek tárolhatók, melyek aztán a DPTR útján olvashatók ki. Például az EMON51.A51 monitorban utána lehet nézni, hogyan van az STXT rutin programozva, mely egyébként a programmemóriában álló szövegek kiadására szolgál.

■ A (külső) adatmemória címzése

A rendszerint RAM-ból álló külső adatmemóriához való hozzáférésre az MOVX utasítás szolgál. Az X ebben az utasításnévben az „eXternal" helyett áll. @DPTR megadása esetén, mint pl. a 37. vagy a 43. sorban, a DPTR adatpointer tartalma kerül 16-bites címként felhasználásra. @R0 vagy @R1 megadása esetén a cím alsó byte-jaként R0 vagy R1 tartalma, felső byte-jaként pedig a P2 Port tartalma (OAOH SFR-cím) kerül alkalmazásra. A Compuboard-nál a OCOOH-tól OFFFFH-ig terjedő „adatmemória"-címtartomány input/output célokra van előirányozva (memory mapped I/O). Ha tehát e címek alatti címekről akarunk behívni, akkor a MOVX utasítást használjuk.

6

```
; ****** DATEI BSP5.A51 ****************************************
;
P1      EQU   090H            ; SFR PORT1 Adresse ist 090H
;
        ORG   4100H           ; 1. Programm steht spaeter ab 4100H
START   MOV   P1,#00010000B   ; LED D1 an, Rest aus
        MOV   DPTR,#500       ; 500 Millisekunden
        ACALL ZEIT            ; Warten
        MOV   P1,#11100000B   ; LED D2,D3,D4 an D1 aus
        MOV   DPTR,#100       ; 100 Millisekunden
        ACALL ZEIT            ; Warten
        SJMP  START           ; Wiederholen
;
        ORG   4200H           ; 2. Programm steht spaeter ab 4100H
START2  MOV   P1,#010B        ; Lautsprecher +5 Volt
        MOV   DPTR,#1         ; 1 Millisekunde
        ACALL ZEIT            ; Warten
        MOV   P1,#000B        ; Lautsprecher 0 Volt
        MOV   DPTR,#1         ; 1 Millisekunde
        ACALL ZEIT            ; Warten
        SJMP  START2          ; von vorne los
;
;  MONITOR INTERFACE
ccLTIME EQU  021H   ; MONITOR Kommando , DPTR Millisekunden Verzoegerung
COMMAND EQU  030H   ; MONITOR Kommando Speicherstelle
MON     EQU  0200H  ; MONITOR Einsprungadresse
;
ZEIT    MOV   COMMAND,#ccLTIME
        LJMP MON
        END
```

6. ábra. Két program egybeépítve: A 4100H címtől a LED-diódák villogni kezdenek. A 4200H startcímtől hang generálására kerül sor

Monitor-behívások

Miután már néhány fontos utasítást és címzésfajtát megismertünk, rátérünk arra a kérdésre, hogy hogyan tudjuk használhatóvá tenni a Monitor-Eprom-ban tárolt segédrutinokat. Ez a következő módon történik: Legelőször a belső RAM-ba a O3H címre egy úgynevezett parancs-byte-ot (command byte) kell beírni (52. vagy 54. sor). E byte alapján fogja tudni a Monitor-Epromban levő program, hogy melyik rutin kerül behívásra. A monitorprogram által rendelkezésre bocsátott programok a tanfolyam diszkettjén az EMON51.DOC fájlban szerepelnek. A mindenkor használandó parancs-byte is ott van megadva. Ezek a rutinok sok munkát vesznek át tőlünk. Érdemes tehát alaposabban átnézni az így rendelkezésre álló lehetőségeket, annál is inkább, mert a monitor

6. ábra.
– SFR PORT1 Adresse ist 090H = SFR PORT1 címe 090H
– 1. Programm steht spaeter ab 4100H = az 1. program később, 4100H-tól kezdődik
– LED D1 an, Rest aus = D1 LED be, többi ki
– 500 Millisekunden = 500 milliszekundum
– Warten = várni
– Led D2, D3, D4 and D1 aus = D2, D3, D4 be, D1 ki
– 100 Millisekunden = 100 milliszekundum
– Wiederholen = ismételni
– 2. Programm steht spaeter ab 4200H = a 2. program később, 4200H-tól kezdődően
– Lautsprecher 0 Volt = hangszóró 0 Volt
– von vorne los = indulj előlről
– MONITOR Kommando, DPTR Millisekunden Verzoegerung = MONITOR parancs, DPTR milliszekundumnyi késleltetés
– MONITOR Kommando Speicherstelle = MONITOR parancs tárolási címe
– MONITOR Einsprungadresse = MONITOR belépési címe
– Zeit = idő

(EMON51.A51) forrásprogramjából kivehető, hogy hogyan vannak programozva az egyes rutinok.

Kapáslövés kiadása céljából a O2OH parancs-byte-ot kell használni, majd ezután a O2OOH cím alatti monitorprogramot kell behívni. A programozás tehát a következő módon történhet:

```
MOV O3OH, #O2OH
LCALL O2OOH
```

Hogy a programozás e módja nem a legolvasmányosabb, arról a BSP2 példában az 52-től 56-ig terjedő sorokkal való összehasonlítás tanúskodik. A szimbólikus nevek használata tehát a programot érthetőbbé teszi.

A tanfolyam diszkettje a parancs-byte-ok céljára használt valamennyi szimbólumot tartalmazza. Az 1. részben bemutatott egyik rutin *(3. ábra)* egy karakterlánc kiadásáról gondoskodik. A hozzá tartozó parancsbyte értéke 2 és a neve nálunk „ccSTXT", ami „Command Code Send TeXT"-ből származik. A kiadandó karaktereknek a programmemóriában kell tartózkodniok. Az utolsó karakter helyén nullának (OH) kell állnia, hogy az alprogram tudja, hol a szöveg vége (44 sor). A szöveg kezdési címét DPTR-ben kell rendelkezésre bocsátani. Az egész eljárást a 9. és 10., valamint az 52 és 53 sorok valósítják meg.

Egyszerű Port-output

Mindeddig a Compuboard-ot mintegy fekete dobozként kezeltük és még egyáltalán nem foglalkoztunk olyan alkalmazással, melyben a mikrokontroller PORT-jai útján jeleket adunk ki és azokkal vezérlési- vagy egyéb feladatokat hajtunk végre. A Compuboard-nál a 8051 külső kapcsolása következtében nem használható valamennyi Port tetszőlegesen, mivel a P0 és a P2 Portok már cím- és adatbuszként a külső EPROM és a RAM számára ki vannak használva. A P1 Port nyolc vezetéke azonban a Compuboard-on található kétirányú meghajtó útján az a20-tól a13-ig terjedő pontokon át ki van vezetve. A kétirányú meghajtó irányát az a21-re adott jel határozza meg. Ha ezt a kivezetést nem kötjük be, akkor P1 (a byte-onkénti nyolc bitnek megfelelően) nyolc jel outputjára használható. Az 5. ábra szerinti kapcsolással (mely megfelel a bővítő-kártyának) 4 LED be- és kikapcsolását tudjuk elvégezni és egyszerű jeleket tudunk a hangszóróra bocsátani.

Az egy byte-on belüli bitek, így egy Port vagy regiszter bitjei is rendszerint 0-val kezdődően kerünek beszámozásra. A legalacsonyabb helyi értékű (bináris írásmódban a jobb oldalon álló) bit a 0, a legmagasabb helyi értékű a 7-es számot kapja. P1. 3-mal tehát P1 negyedik bitjét kell jelölni. Például a D3 LED bekapcsolását a következők szerint kell programozni:

```
MOV P1, #01000000B;
LED 4 ki 3 be 2 ki 1 ki
jobb oldali bit mind 0
```

A 26. sorban ilyen output programozása szerepel a Reset után (P1 tartalma = 11111111B, az összes LED be), a 26. sor végrehajtása során a LED-ek által alkotott minta megváltozik.

Ezt az egyszerű output-lehetőséget használjuk ki most az egyszerű BSP5 program bemutatására, amely LED-diódák villogtatására szolgál (startcíme 4100H) vagy 500 Hz-es hangjelet hoz létre (a 4200H címtől kezdődően). A jelek generálása egyszerűen azzal történik, hogy a P1 Porton megfelelő bitminta output jelenik meg. Szolgálatkész szellemként megint egy olyan monitor-rutin siet segítségünkre, mely a DPTR-ben előre megadott számú milliszekundumig tartó várakozást tesz lehetővé. A programot a *6. ábrán* mutatjuk be.

A programozást nem zöld asztal mellett szokták megtanulni. A tanfolyamunkon szerzett ismeretek elmélyítése céljából a Compuboard-dal és a rendelkezésre álló programozási segédletekkel feltétlenül el kell játszadozni és saját ötletekkel kell kipróbálni. Itt is van mindjárt az első feladat: Módosítsuk a BSP5 programot úgy, hogy 2 perc és 50 másodperc kezdeti várakozási idő eltelte után jel hangozzék fel. Ezzel a Compuboard Teatimer-ként való alkalmazásra kész. Egy módosított BSP5 program egyszerű négyszöggenerátorként is alkalmazható. ◘

Videókamera-időzítő

G. Hagl

Néhány videókamera rendelkezik ugyan távkezelő csatlakozóval és annak használatára a kezelési útmutatóban utalások is találhatók, ha azonban e csatlakozó útján időzsugorítást, feliratos felvételeket vagy hasonlókat kívánunk vezérelni, akkor hamarosan problémák merülnek fel.

A Balupunkt 8010 típusnál például a vezérléshez egy olyan kapcsoló szükséges, amely az érintkezőt mintegy 40 ... 60 ms időtartamra zárja. A funkció kikapcsolást egy második, ugyanilyen impulzus végzi. Csupán az ujjak ügyességével bizony elég nehéz ezzel a megoldással előírt időtartamú be-

1