

4. fejezet

**A 8031/51 típusú mikrokontroller
család utasításkészlete**

A 4. fejezet artalomjegyzéke:

4.	A 8031/51 típusú mikrokontrollerek utasításai	68
4.1.	Az utasítások, és azok hossza	68
4.2.	Futási idő	68
4.3.	Címzési módok	69
4.1.1.	A címzésről általában	69
4.1.2.	Regiszter-címzés	70
4.1.3.	Direkt címzés	70
4.1.4.	Indirekt címzés	70
4.1.5.	Közvetlen címzés	71
4.1.6.	Indirekt regiszter-címzés	71
4.2.	A különböző utasítás fajták	71
4.2.1.	Az adatátviteli utasítások	72
4.2.2.	Az aritmetikai utasítások.	73
4.2.3.	Logikai és boole utasítások.	74
4.2.4.	Vezérlés átadó utasítások.	76
4.3.	Az utasítások hossza és végrehajtási idejük	79

4. A 8031/51 típusú mikrokontrollerek utasításai

A 8031/51 mikrokontroller család *mindegyik* egyedének az utasításkészlete *egyforma*. Az utasítások nagyon hasonlítanak a 8080 típusú mikroprocesszoréhoz. Lényeges *eltérés* azonban, hogy a 8051-t elsősorban különböző *vezérlési feladatokhoz* fejlesztették ki, és ezért *bit műveletek* programozása is lehetséges.

A *vezérlési* feladatok többségénél *kétállapotú* bemeneti jelek állapotát kell *lekérdezni*, és ezek alapján *logikai műveletek* elvégzése után kétállapotú *vezérlőjeleket* kell kiadni, *reléket, lámpákat* stb. *vezérvezérléséhez*. Az ilyen jellegű működés bitszervezést igényel. A *bitműveleteket* végzéséhez fejlesztették ki – a belső *memória egy szegmensének* - *bitenkénti címzési* lehetőségét, és a bitekre is értelme *logikai műveletek* (boole - utasítások).

4.1. Az utasítások, és azok hossza

A 8051 mikrokontrollernek összesen *111 utasítása* van. Az utasítások között *egy-, két- és három* bájt hosszúakat találunk. Az utasítás bájt-számban a *műveleti kód* és - rendszerint közvetetten - az *operandus* -ok is benne vannak. Az utasítások között:

- 49 egy-bájt
- 45 két-bájt és
- 17 három-bájt

hosszúságú van.

4.2. Futási idő

Egy program *futási idejének* pontos meghatározásához ismerni kell azt, hogy az egyes utasítások feldolgozása hány *gépi ciklus* alatt történik. A 8051 mikrokontroller utasításainak a végrehajtása - a MUL és DIV aritmetikai utasítások kivételével - *egy-, vagy két* gépi ciklus időtartamú. A két utasítás végrehajtásához *négy gépi ciklus* szükséges. A 2. fejezetben tárgyaltuk, hogy egy *műveleti ciklus 12 oszcillátor-periódus* hosszú. Az órajel frekvenciájának ismeretében a program végrehajtásához szükséges idő - a futási idő - a következők szerint számolható ki:

A program *összes utasításaihoz* tartozó *ciklusok* összegét *szorozni* kell a *quartz* periódusidejének 12-szeresével.

A 8051 utasításai - a végrehajtási ciklusszám alapján - az alábbiak szerint oszlanak meg:

- 63 egy-ciklusú,
- 46 két-ciklusú és
- 2 négy-ciklusú.

4.3. Címzési módok

A memóriákban tárolt *értékek*, illetve *tárolásuk helye* (címeik) különböző formában adhatók meg. Ezeket nevezzük *címzésnek*.

4.1.1. A címzésről általában

A 8051-nél az alábbi öt különböző címzési mód alkalmazható:

- regiszter-,
- direkt-,
- indirekt-,
- közvetlen-,
- indirekt regiszter-

címzés.

A mikrokontroller különböző memóriaterületekkel kommunikál. A fizikailag, vagy logikailag elkülönített területek más-más címzési móddal érhetőek el.

A 8051 öt féle címzési módja alapvetően két nagy csoportot alkot, mégpedig a direkt és az indirekt címzésűeket.

A direkt címzésnél az utasítás része a megfelelő tárlócella címe (hexadecimális szám). Az indirekt címzésnél a tárlócella címét egy regiszterben vagy egy másik tároló-cellában van. Az utasítás ez utóbbiak címét tartalmazza.

Az indirekt címzés kissé nehezkesebb, mivel a programozónak először arról kell gondoskodnia, hogy a szükséges cím a megfelelő tároló-egységbe kerüljön. Ugyanakkor rugalmasabb programozást tesz lehetővé. Az indirekt címzés egy másik lehetősége az ún. adatmutató, vagyis a Data-Pointer használata. Az adatmutatóba egy 16 bites szám írható, amely 64 Kbájt kapacitású adatmemória címzését teszi lehetővé. Az adott értékhez **relatív** címzés is megoldható.

A relatív címzés a strukturált programozást teszi könnyebbé. Sok esetben előnyös ez a módszer.

4.1.2. Regiszter-címzés

Az aktuális bank R0 és R7 regisztereire, valamint az un. processzor regiszterekre - ACC, B, PSW, DPTR - közvetlenül lehet hivatkozni. Az ilyen utasításoknál az utasítás-kód három legalacsonyabb helyiértékű bitje határozza meg az aktuális regisztert.

4.1.3. Direkt címzés

A direkt címzésnél az utasítás része az elérendő memória címe. A cím az utasítás műveleti kódja utáni hexadecimális szám.

Például: **MOV A,32H** utasítás a **32H** című belső memória tartalmát az Akkumulátorba (A) viszi.

A speciális funkcióregiszterek (SFR-k) csak a direkt címzéssel érhetők el. A belső RAM alsó 128 bájtja direkt módon is címezhető.

4.1.4. Indirekt címzés

Az indirekt címzésnél egy regiszterben van az a cím, amellyel a memória valamelyikébe írni, vagy olvasni akarunk.

E címzésnél tehát nem az utasítás, hanem - az aktuális regiszterbank - **R0**, vagy **R1** regisztere, illetve a **DPTR** tartalmazza az elérendő memóriacella címét. Az R0, R1 címzésnél a 8 bites tartalommal 256 bájt széles RAM terület címezhető. E terület lehet a teljes belső-, vagy a külső **RAM** egy lapja (page). A 16 bites DPTR segítségével csak külső memória (adat, vagy program) címezhető.

Az indirekt címzésre példa a **MOVX A,@Ri** utasítás (**i** értéke **0**, vagy **1**), amely az akkumulátorba viszi a külső RAM egy cellájának tartalmát. A cella címe az Ri regiszterben van. A magasabb helyiértékű 8 címbit ez alatt nem változik meg. Az utasítás segítségével relatív címzést is meg tudunk valósítani. Először a **P2** SFR-be kell beírni a cím magasabb bájtját, s az Ri-be pedig az alacsonyabb bájtot.

Az indirekt címzések egy bájtosak. Segítségükkel a külső RAM 256 bájtos területe érhető el. A teljes memóriaterület - 64 Kbájt - címzése a DPTR adatmutató regiszterrel történhet, pl. **MOVX A,@DPTR** .

A stack - terület a PUSH és POP utasításoknál is indirekt a címzés. A címe itt a stack - mutatóban (SP) van. Ezen az alapon a stack - terület is a RAM tetszőleges területére helyezhető. A programozónak kell biztosítani a Stack kezdő címének megfelelő beállítását.

4.1.5. Közvetlen címzés

A közvetlen címzésnél az utasításhoz tartozó adat, vagy cím az utasítás része.

Ez azt jelenti, hogy a kívánt értéket - az utasítás részeként - a programmemóriába (ROM, vagy EPROM) kell beírni.

Például: a **MOV A,#23H** utasítás 23 hexadecimális értéket ír az akkumulátorba.

4.1.6. Indirekt regiszter-címzés

Az indirekt regiszter-címzésnél a tényleges fizikai címet két regiszter tartalmának az összege adja. A cím tehát egy **bázis-**, és egy **eltolási** (offset) címrészből áll. A bázis cím vagy az adatmutatóban (DPTR), vagy a programszámlálóban (PC) van. E címhez adja hozzá egy belső regiszter - az akkumulátor (A) tartalmát.

Például: **MOVC A, @A+ DPTR.**

Az indirekt regiszter-címzési formát rendszerint a programtárolóban lévő táblázat kezeléséhez használjuk.

4.2. A különböző utasítás fajták

A 8051-nek 111 különböző utasítása van. Ezek a következő négy csoportba sorolhatók:

- adatátviteli utasítások,
- aritmetikai utasítások,
- logikai, ill. bit műveleti utasítások,
- vezérlés átadó utasítások.

A felhasználói könyvek, katalógusok, és itt is az alábbi rövidítéseket használjuk:

Rn az R0 - R7 munkaregiszterek valamelyikét jelöli. Az utasítás e regiszter tartalmára vonatkozik.

direct a belső RAM-ban egy cím. Az utasításban hexadecimálisan kell megadni.

@Ri	az R0, vagy R1 regiszterekkel történő indirect címzés.
#data	az utasításban megadott 8 bites adat.
#data 16	az utasításban megadott 16 bites adat (a 2. és a 3. bájt).
rel	egy relatív cím. A következő utasítás címéhez viszonyítottan -128 és +127 területen belülre mutathat.
bit	jelentheti a 128 "softver-flag" valamelyikét, egy I/O bitet illetve vezérlő vagy státuszbitet.

4.2.1. Az adatátviteli utasítások

Az adatátviteli - az un. **MOV** - utasítások regiszter vagy memória cella tartalmát viszik át egy másik regiszterbe, vagy memória cellába.

Az utasítás típus három csoportra bontható.

⇒ Az általános adatátviteli utasítások

A csoportot az alábbi utasítások alkotják:

MOV **cél, forrás**

egy bitet, vagy bájtot visz át a belső memóriában lévő **forrás**-helyről a **cél**-helyre.

PUSH **reg**

inkrementálja a Stack-Pointer tartalmát majd a vonatkozó regiszter tartalmát az SP által címzett memóriába írja.

POP **reg**

az SP által címzett memória tartalmát átírja a vonatkozó regiszterbe, majd dekrementálja az SP tartalmát.

Az utasítások a címzett helyről az adatot átmásolják a cél helyre, de közben a forrás tartalmát nem változtatják meg.

⇒ **Akkumulátor**-utasítások

Ezeknél az utasításoknál az ACC-regiszter a célja, vagy a forrása az adatátvitelnek. Az akkumulátor önmaga lehet a cél is és a forrás is.

XCH A,mem

(exchange) az akkumulátor és a címzett memória tartalmát cseréli fel.

XCHD A,mem

hasonló az XCH utasításhoz, de csak az akkumulátor és a címzett memória tartalmának az alsó **négy bitjét** cseréli meg.

MOVX cél,forrás

a külső adatmemória és az akkumulátor között végez adatátvitelt, vagyis az egyik memóiahely mindig az Akkumulátor.

MOVC A,forrás

programtárolóból visz egy bájtot az akkumulátorba. A címezésnél a DPTR-ben, vagy PC-ben van a báziscím.

⇒ ***Data-Pointer utasítás***

MOV DPTR, # áll

a megadott 16 bites állandóval tölti fel a **Data-Pointer** -t (DPTR).

4.2.2. Az aritmetikai utasítások.

A 8051 mikrokontroller matematikai műveletei korlátozottak. Csupán 8 bites előjel nélküli számokkal lehet műveleteket végezni. Az Overflow-flag (OV) segíti a felhasználót az előjeles számok összeadásánál és kivonásánál. A 8051 aritmetikai utasításai hasonlítanak a 8080 és a 8085 mikroprocesszorok azonos utasításaihoz.

⇒ ***Összeadó utasítások***

ADD A , op2

a 2. Operandus és az akkumulátor tartalmát (1. operandus) adja össze. Az eredmény az akkumulátorba kerül.

ADDC A , op2

mint az ADD utasítás, de még a **CY** flag értékét is az eredményhez adja.

DA A

a BCD számok összeadása után alakítja az eredményt BCD alakúra.

INC bájt

a címzett memória tartalmát inkrementálja (1-el növeli).

⇒ *Kivonó utasítások*

DEC bájt

A címzett memória tartalmát dekrementálja (1-el csökkenti). Az INC utasítás ellentettje.

SUBB A , op2

az akkumulátor tartalmából (1. operandus) levonja a címzett 2.operanduszt. Ha a CY=1, akkor az eredményből még 1-t levon. A művelet eredménye az akkumulátorba kerül.

⇒ *Szorzó és osztó utasítások*

A B regisztert (az SFR -ben) kizárólag csak ezeknél az utasításoknál használja közvetlenül a kontroller.

MUL AB

két előjel nélküli 8 bites számot szoroz össze. A szorzandókat az **ACC** és a **B** regiszterekbe kell vinni. A szorzat kétbájtos lesz. Az eredmény alacsonyabb helyiértékű bájtja az **ACC** -be, míg a magasabb helyiértékű pedig a **B**-be kerül. A 256-nál, nagyobb eredménynél az **OV** 1-be íródik.

DIV AB

az **ACC** tartalmát osztja a **B** tartalmával. Előjelet nem vesz figyelembe! Az osztás egészrészét az **ACC** fogja tartalmazni. A maradék kerül a **B** regiszterbe. Ha a hányados 0, akkor az **OV** flag 1-be íródik.

4.2.3. Logikai és boole utasítások.

A logikai utasítások formailag nagyon hasonlóak a mikroprocesszorok azonos jellegű utasításaihoz. E csoport viszont - az aritmetikai műveletekkel ellentétben - sokkal bővebb alkalmazási lehetőséget nyújt. A 8051-es mikrokontrollert elsődlegesen vezérlésekhez fejlesztették és ezért mind bitekkel, mind pedig bájtokkal tud logikai műveleteket végrehajtani.

ANL op1 , op2

logikai **ÉS** művelet az operandus -ok azonos helyiértékű bitjei között. Az eredmény az első operandus (op1) helyére íródik, miig a második nem változik meg.

ORL op1 , op2

az ANL -hez hasonlóan végez logikai **VAGY** műveletet.

XRL op1 , op2

az **EXKLUSIV-OR** (kizáró-vagy) művelet két operandus azonos helyiértékű bitjei között. Az eredmény - az ANL és ORL műveletekhez hasonlóan - az első operandus helyére kerül.

Az ANL, ORL műveletek egyes bitek között is alkalmazhatóak. A hagyományos diszkrét logikai hálózatok kiválthatók a 8051 bázisú rendszerrel (például a különböző tárolt programú vezérlések).

SETB bit

A direkt címzett bitet 1-be írja.

CLR bit

Törli a direkt címzett bitet.

⇒ *Forgatás (rotáció)*

RL A

az akkumulátor tartalmát egy hellyel balra forgatja.

RLC A

az akkumulátor tartalmát a CY közbeiktatásával forgatja egy hellyel balra.

RR A

az akkumulátor tartalmát egy hellyel jobbra forgatja.

RRC A

az akkumulátor tartalmát a CY közbeiktatásával forgatja egy hellyel jobbra.

Az RLC és RRC utasításoknál a **CY** az akkumulátor **kilencedik** bitjének tekinthető.

SWAP A

felcseréli az akkumulátor felső- és alsó négy bitjét.

4.2.4. Vezérlés átadó utasítások.

A vezérlés átadó utasítások alkalmazhatók a programokon belüli különböző ugrások végrehajtására. Ilyen lehet egy szubrutin hívása, vagy egy feltételtől függő program-elágazás. A utasítások az alábbi három csoportba sorolhatók:

- feltétel nélküli programelágazás,
- feltételes programelágazás ,
- megszakítás kiszolgálás.

Ezen utasítások közös jellemzője, hogy a programszámláló tartalmát változtatják meg. A PC határozza meg, hogy a kontroller mely címről hiv. beutasítást. Ennek megváltoztatásával vezérelhető egy programelágazás.

A program-elágazási utasítások megtörik a program tiszta sorrendi (lineáris) lefutását.

Először meg kell ismerni az ugrás (JUMP), a szubrutinhívás (CALL), és a megszakítást (Interrupt) kiszolgáló rutinhívás közötti alapvető különbségeket.

A különböző **JUMP cím** utasítások (esetleg egy meghatározott feltételtől függően) a program meghatározott helyére történő ugrást vezérlik. A program végrehajtása e helyről fog folytatódni.

A **CALL cím** utasítás egy szubrutin kezdetére adja át a következő utasítás hívását. Ugyanakkor a Stack-be automatikusan eltárolja a főprogram következő utasításának címét. A rutin feldolgozása egy **RET** utasításig tart. A RET hatására a főprogram - a Stack-ben tárolt címről - fut tovább. Ugyanaz a szubrutin a főprogram tetszőleges helyéről és többször is hívható. Elsődlegesen a különböző összetettebb, de ismétlődő műveletekhez használjuk a szubrutinokat.

Egy szubrutin CALL utasítással történő hívása |kizárólag szoftverből történhet.

A **megszakítási** programelágazás, hasonlóan a CALL-hoz egy szubrutin hívását jelenti. Lényeges eltérés, az hogy ezt az ugrást egy hardver-esemény váltja ki. A szubrutin hívá-

sa a megszakítás után azonnal megtörténik. A CALL-al való szubrutinhívásra addig kell várni, amíg a program az adott CALL-hoz nem ér.

⇒ *Feltétel nélküli vezérlésátadás*

A feltétel nélküli ugrás, mint ahogyan a nevében is benne van, nem függvénye valamilyen eredménynek. Az ugrás minden esetben bekövetkezik, amikor a program egy ilyen utasításhoz ér. Szigorúan véve a RETURN utasítás is e csoportba tartozik. A CALL végrehajtása előtt a főprogram következő utasításának címe a stack-be íródik, s közben a Stack-pointer értéke kétszer inkrementálódik (16 bites cím kerül be a stack-be). A RET utasítás hatására ez a cím visszaíródik a PC-be, miközben az SP tartalma kettővel csökken.

ACALL cím11

két bájtos szubrutinhívó utasítás. 2 K-bájtos szegmensben belüli programugrást hajt végre. Az ACALL-hoz 11 bites cím tartozik. A PC-ben lévő legnagyobb helyiértékű öt bit érvényes marad (együtt adják a 16 bites címet). Az ACALL hívásakor a PC tartalma inkrementálódik. Ha az ACALL egy 256-bájtos szegmens utolsó két bájta, akkor a PC inkrementálása miatt az a következő szegmensbe kerül.

LCALL cím16

három bájtos szubrutinhívó utasítás. Alkalmas a teljes 64-Kbájton belüli tetszőleges című rutin hívására a 16 bites címezés miatt.

AJMP cím11

két bájtos - 2 K-bájtos szegmensben belüli - ugrást vezérlő utasítás.

LJMP cím16

három bájtos ugrást vezérlő utasítás. Alkalmas a teljes 64-Kbájton belüli tetszőleges címre történő ugrásra a 16 bites címezés miatt.

SJMP rel

relatív ugrást vezérlő utasítás. A SHORT JUMP használatával csak 256 bájt területen belüli ugrás oldható meg.

JMP @A+DPTR

az ugrás címét a DPTR és az akkumulátor tartalmának összege adja. A 8 bites akkumulátor-tartalom egy lapot fog át. A DPTR a teljes 64 Kbájt -os programmemória tetszőleges helyére mutathat.

Megjegyzés: a legtöbb assembler elfogadja a JMP, illetve a CALL utasítás mnemonic -ot is. Az ugrás távolságától függően helyettesít a megfelelő utasítással.

RET

az ACALL vagy az LCALL utasításokkal meghívott szubrutinból való visszatérést vezérli. Hatására a PC-be íródik a szubrutinhívást követő utasítás címe.

RETI

megszakítás (interrupt) után hívott szubrutinból való visszatérés utasítása. A visszatérésen kívül felszabadítja a megszakítástiltást.

⇒ ***Feltételes ugrás***

Az előzőekkel ellentétben az ugrás csak akkor következik, ha meghatározott feltétel teljesül.

A feltételes ugrás mindig relatív. Az éppen aktuális helytől számítottan 8 bites címtávolságon belül lehet a cél-hely, vagyis az utasítás helyétől számítva **-128**, vagy **+127** bájt területen belül lehet a cél cím.

A 8051-nek a következő feltételes ugró-utasításai vannak:

A program a megadott címre ugrik, ha teljesül a megadott feltétel

JZ **rel**

ha az akkumulátor tartalma **0**.

JNZ **rel**

ha az akkumulátor tartalma **nem 0**.

JC **rel**

ha a Carry - Flag értéke **1**.

JNC **rel**

ha a Carry - Flag **0**.

JB **bit, rel**

ha a direkt címzett bit **1**.

JNB **bit, rel**

ha a direkt címzett bit **0**.

JBC **bit, rel**

ha a direkt címzett bit **1**, majd **törli** a bitet.

CJNE **adat1, adat2, rel**

összehasonlítja a megadott (címzett) két adatot. Akkor következik az ugrás, ha a két tartalom **nem egyforma**. Amikor az első adat (regiszter tartalom) a **kisebb**, akkor a **CY** is 1-be íródik. Ellenkező esetben törlődik.

DJNZ **mem , rel**

először dekrementálja az adott címen lévő értéket. Majd ellenőrzi, hogy a csökkentett érték 0 vagy nem és az utóbbi esetben hajtja végre az adott címre az ugrást.

⇒ *A FLAG -ket befolyásoló utasítások*

A 8051 utasításai közül csak nagyon kevés változtatja a mikrokontroller **flag** -jeit. A 4.5. táblázatban foglaltuk össze, hogy az egyes flag -ekre melyik utasítás hat. Az **X** azt jelenti, hogy az adott flag -t az utasítás változtatja. A 0 ill. az 1 jelöli azt a konkrét értéket, amelyre a flag mindig beáll a művelet hatására.

⇒ *READ-MODIFY-WRITE utasítások*

A READ-MODIFY-WRITE utasítások egy port tartalmát kiolvassák, a kívánt értékre változtatják, és azonnal visszaírják a port - latchbe. A READ-MODIFY-WRITE utasításokhoz mindig egy port címe tartozik.

A csoportba tartozó utasítások a 4.6. táblázatban láthatók

4.3. Az utasítások hossza és végrehajtási idejük

Az 4.1 - 4.4 táblázatokban az egyes utasítások hosszát és az oszcillátor periódusában megadott végrehajtási idejét adtuk meg. A felhasználó ezek alapján kiszámíthatja a szükséges memória-területet és a futás időt.

MOV	@Ri,direct	2	24
-----	------------	---	----

4.1.táblázat Adatátviteli utasítások

Utasítás	Hossz bájtban	Oszcil- látor periódus
MOV A,Rn	1	12
MOV A,direct	2	12
MOV A,@Ri	1	12
MOV A,#data	2	12
MOV Rn,A	1	12
MOV Rn,direct	1	24
MOV Rn,#data	2	12
MOV direct,A	2	12
MOV direct,Rn	2	24
MOV direct,direct	3	24
MOV direct,@Ri	2	24
MOV direct,#data	3	24
MOV @Ri,A	1	12

Utasítás	Hossz bájtban	Oszcil- látor periódus
MOV @Ri,#data	2	12
MOV DPTR,#data16	3	24
MOVC A,@A+DPTR	1	24
MOVC A,@A+PC	1	24
MOVX A,@Ri	1	24
MOVX A,@DPTR	1	24
MOVX @Ri,A	1	24
MOVX @DPTR,A	1	24
PUSH direct	2	24
POP direct	2	24
XCH A,Rn	1	12
XCH A,direct	2	12
XCH A,@Ri	1	12
XCHD A,@Ri	1	12

4.2.táblázat Aritmetikai utasítások

				Utasítás		Hossz bájtban	Oscil- látor periódus
Utasítás							
				INC	A	1	12
ADD	A,Rn	1	12	INC	Rn	1	12
ADD	A,direct	2	12	INC	direct	2	12
ADD	A,@Ri	1	12	INC	@Ri	1	12
ADD	A,#data	2	12	INC	DPTR	1	24
ADDC	A,Rn	1	12	DEC	A	1	12
ADDC	A,direct	2	12	DEC	Rn	1	12
ADDC	A,@Ri	1	12	DEC	direct	2	12
ADDC	A,#data	2	12	DEC	@Ri	1	12
SUBB	A,Rn	1	12	MUL	AB	1	48
SUBB	A,direct	2	12	DIV	AB	1	48
SUBB	A,@Ri	1	12	DA		1	12
SUBB	A,#data	2	12				

4.3.táblázat Logikai - utasítások

				RLC	A	1	12
Utasítás		Hossz bájtban	Oszcil- látor periódus	Utasítás		Hossz bájtban	Oszcil- látor periódus
ANL	A,Rn	1	12	RR	A	1	12
ANL	A,direct	2	12	RRC	A	1	12
ANL	A,@Ri	1	12	SWAP	A	1	12
ANL	A,#data	2	12	CLR	C	1	12
ANL	direct,A	2	12	CLR	bit	2	12
ANL	direct,#data	3	24	SETB	C	1	12
ORL	A,Rn	1	12	SETB	bit	2	12
ORL	A,direct	2	12	CPL	C	1	12
ORL	A,@Ri	1	12	CPL	bit	2	12
ORL	A,#data	2	12	ANL	C,bit	2	24
ORL	direct,A	2	12	ANL	C,/bit	2	24
ORL	direct,#data	3	24	ORL	C,bit	2	24
XRL	A,Rn	1	12	ORL	C,/bit	2	24
XRL	A,direct	2	12	MOV	C,bit	2	12
XRL	A,@Ri	1	12	MOV	bit,C	2	24
XRL	A,#data	2	12	JC	rel	2	24
XRL	direct,A	2	12	JNC	rel	2	24
XRL	direct,#data	3	24	JB	bit,rel	3	24
CLR	A	1	12	JNB	bit,rel	3	24
CPL	A	1	12	JBC	bit,rel	3	24
RL	A	1	12				

4.4.táblázat Vezérlő utasítások

				Utasítás		Hossz bájtban	Oszcil -látor periódus
			JNZ	rel		2	24
			CJNE	A,direct,rel		3	24
ACALL	addr11	2	24	CJNE	A,#data,rel	3	24
LCALL	addr16	3	24	CJNE	Rn,#data,rel	3	24
RET		1	24	CJNE	@Ri,#data,rel	3	24
AJMP	addr11	2	24	DJNZ	Rn,rel	3	24
LJMP	addr16	3	24	DJNZ	direct,rel	3	24
SJMP	rel	2	24	NOP		1	12
JMP		1	24	RETI		1	12
JZ	rel	2	24				

4.5.táblázat. A flag -ket állító utasítások

Utasítás	CY	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		
SETB C	X		
CLR C	X		
CPL C	X		
ANL C,bit	X		
ANL C,/bit	X		
ORL C,bit	X		
ORL C,/bit	X		
MOV C,bit	X		
CJNE	X		

Utasítás	Példa
ANL	ANL P2,A
ORL	ORL P1,A
XRL	XRL P1,A
JBC	JBC P2.2,re
CPL	CPL P1.1
INC	INC P1
DEC	DEC P1
DJNZ	DJNZ P1,rel
MOV	MOV P2.1,C
CLR	CLR P1.0
SETB	SETB P1.0