Hardver alapok

(VIIIBA01)

Assembly alapok

Rácz György gyuriracz@iit.bme.hu

2022/23. tanév őszi félév

Assembly Bevezetés

- Assembly→összeszerelés
- Feladat: adott processzorhoz gépi kód előállítása jól olvasható szövegfájlból
- Változók, ugrási címek automatikus számítása
- Igény esetén több modulból álló program fordítása, esetenként eltérő programozási nyelveken íródott modulokból is

A fejlesztés folyamata

- Program írás általában PC-n →szövegszerkesztő a kód elkészítéséhez
- Szövegfájl fordítása (cross-assembler)→bináris tartalom előállítása
- [több programmodul esetén összefűzés (linkelés)]
- Beprogramozás („égetés”) a mikrokontrollerbe vagy a szimulátorba
- Hibakeresés, javítás

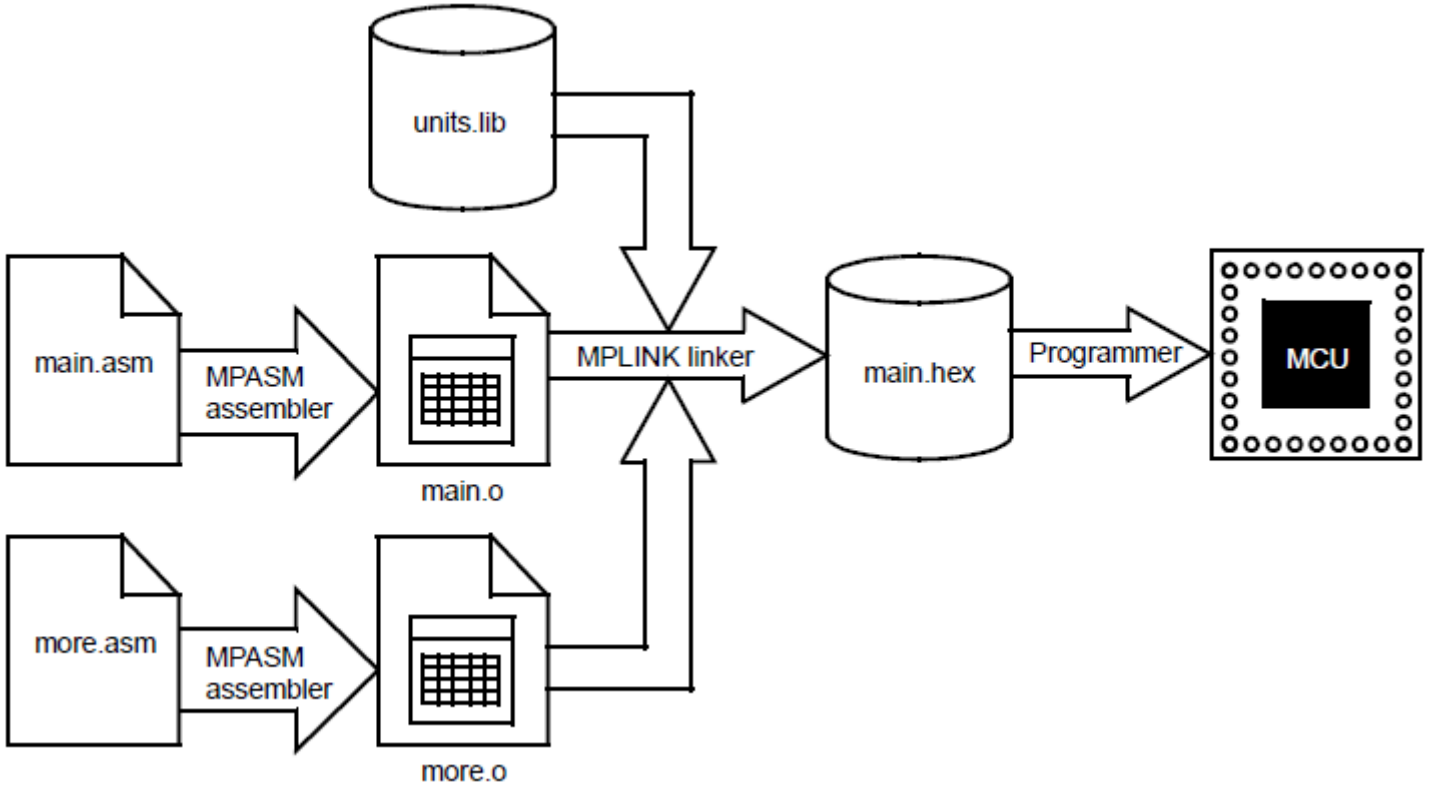
Fontosabb fogalmak

- Mnemonik→utasítás rövid neve
- Gépi kód, OP. kód
- Direktíva→fordítónak szóló *„utasítás”*
- Szimbólum→érték helyettesítés
  - Címke→belépési pont, hivatkozási pont
  - Konstans→állandó érték megadása
- Regiszter, SFR, RAM, FLAG
- Assembler, Linker
- Szintaxis→előírt formátum
  pl.:
  `címke: utasitás cim1,cél ; komment`
- Makró

ASSEMBLY fordítás…

- Cél: gépi kód előállítása
- Több lépésben:
  - Szimbólumok értékeinek kiszámítása (szimbólum tábla)
  - Gépi kód előállítása
  - Listafájlok generálása
- Memória tartalom elhelyezéséhez mutató (Location pointer)
- Fordítást is kell tudnunk befolyásolni→direktívák

Fordítás…


units.lib
main.asm
MPASM
assembler
main.o
MPLINK linker
main.hex
Programmer
MCU
more.asm
MPASM
assembler
more.o

Listafájl - részlet

```
                        00001            list p=18f452
                        00002            #include p18f452.inc
                        00001          LIST
                        00002 ; P18F452.INC  Standard Header File, Version 1.4..
                        00845          LIST
  0000000B              00003 Dest  equ        0x0B
                        00004
000000                  00005            org    0x0000
000000 EF10 F000        00006            goto   Start
000020                  00007            org    0x0020
000020 0E0A             00008 Start movlw      0x0A
000022 6E0B             00009            movwf  Dest
000024 960B             00010            bcf    Dest, 3  ;This line uses 2 op..
000026 EF10 F000        00011            goto   Start
                        00012            end
```

Intelhex formátum

```
:BBAAAATTHHHH....HHHCC
```

BB: soron belüli adatbájtok száma

AAAA: adatrekord kezdőcíme

TT: rekord típusa (00:adat, 01:fájlvége, 02:szegmens cím, 04: lineáris cím)

HH…HH: adatbájtok

CC: ellenőrző összeg kettes komplemense

```
:100000000000000000000000000000000000000F0
:040010000000000EC
:10003200000028004000680 0A800E800C80028016D
:100042006801A9018901EA01280208026A02BF02C5
:10005200E002E80228036803BF03E803C8030804B8
:1000620008040804030443050306E807E807FF0839
:06007200FF08FF08190A57
:00000001FF
```

MPASM-X számábrázolás

- Támogatott számrendszerek: 2, 8, 10, 16

Számrendszer	**Lehetséges megadások**		
Bináris	B’00011001’	b’00011001’	0b11001
Decimális	D’25’	d’25’	.25
Hexadecimális	H’19’	h’19’	0x19

Alkalmazási példa:

```
MOVLW B'11010010'    ; W = d2h
```

MPASM-X direktívák

- Vezérlő direktívák
- Feltételes fordítás
- Adat megadások
- Listázás vezérlés
- Makró megadás
- Tárgykód hivatkozások

(nem foglalkozunk mindegyikkel, részletes leírás: MPASM_userguide_33014L.pdf)

Vezérlő direktívák

- ORG→location pointer beállítása
- EQU→konstans értékadás
- RES→helyfoglalás változónak
- END→forráskód vége jelzés
- BANKSEL→automatikus bankválasztás
- #define→szimbólum létrehozása
- #undefine→szimbólum törlése
- #include→külső forrásfájl beillesztése

Feltételes fordítás

- Fordítási időben bizonyos feltételektől függően befolyásolja a kódgenerálást
- #if
- #else
- #endif
- #ifdef
- #ifndef

Adat megadások

- __config – konfigurációs szó megadása
- cblock – konstans blokk definiálása
- endc – konstans blokk lezárása
- db – declare byte, 1 bájt definiálása
- de – declare EEPROM, adat definiálása
- dt – konstans táblázat definiálása (RETLW)
- dw – declare word, szó definiálása

Egyéb kiegészítések

- Processzor típus és az SFR-ek definicióit tartalmazó .inc fájlok megadása:

```
#include "p16f18875.inc"
```

- Konstansok elérése másik modulból
  - GLOBAL
  - EXTERN
- Változó definiálása BANK0-ban

```
Bank0vars        UDATA  0x20 ;változók ramban
  BITCNT         RES    1    ; 1 bájt foglalás
  DTMP1          RES    1    ; 1 bájt foglalás
```

- A fordítás alatt:
  - BITCNT címke értéke 0x20
  - DTMP1 címke értéke 0x21

Speciális regiszterek

- 12db „Core regiszter”
  - » Minden bankban mindig láthatók
- A három legfontosabb:
  - WREG vagy W
    - » A két operandusos utasítások egyik operandusa mindig itt van (mert az utasításkészlet egy címes, csak egy operandusnak van hely a kódban)
  - STATUS
    - » A legtöbb aritmetikai és mozgató utasítás a végeredményétől függően változtatja (később lesz még róla szó…)
  - BSR
    - » Az aktuálisan kiválasztott memóriabank számát tartalmazza (mert az utasításban csak egy bankon belüli címnek van hely)

Speciális regiszterek

- 12db „Core regiszter”
  - » Minden bankban mindig láthatók
  - – WREG
  - – STATUS
  - – BSR
  - – INTCON
  - – PCL
  - – PCLATH
  - – FSR0/1(H/L)
  - – INDF0/1(H/L)

TABLE 3-11: CORE FUNCTION REGISTERS SUMMARY (1)

Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
Bank 0-31											
x00h or x80h	INDF0	Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu
x01h or x81h	INDF1	Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu
x02h or x82h	PCL	Program Counter (PC) Least Significant Byte								0000 0000	0000 0000
x03h or x83h	STATUS	—	—	—	$\overline{TO}$	$\overline{PD}$	Z	DC	C	---1 1000	---q quuu
x04h or x84h	FSR0L	Indirect Data Memory Address 0 Low Pointer								0000 0000	uuuu uuuu
x05h or x85h	FSR0H	Indirect Data Memory Address 0 High Pointer								0000 0000	0000 0000
x06h or x86h	FSR1L	Indirect Data Memory Address 1 Low Pointer								0000 0000	uuuu uuuu
x07h or x87h	FSR1H	Indirect Data Memory Address 1 High Pointer								0000 0000	0000 0000
x08h or x88h	BSR	—	—	—	BSR4	BSR3	BSR2	BSR1	BSR0	---0 0000	---0 0000
x09h or x89h	WREG	Working Register								0000 0000	uuuu uuuu
x0Ah or x8Ah	PCLATH	—	Write Buffer for the upper 7 bits of the Program Counter							-000 0000	-000 0000
x0Bh or x8Bh	INTCON	GIE	PEIE	TMR0IE	INTE	IOCIE	TMR0IF	INTF	IOCIF	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: These registers can be addressed from any bank.

Utasításkészletben használt rövidítések

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1 .
n	FSR or INDF number. (0-1)
mm	Pre-post increment-decrement mode selection

Utasításkészlet - 1

TABLE 34-3: ENHANCED MID-RANGE INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	2
ADDWFC	f, d	Add with Carry W and f	1	11	1101	dfff	ffff	C, DC, Z	2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	2
ASRF	f, d	Arithmetic Right Shift	1	11	0111	dfff	ffff	C, Z	2
LSLF	f, d	Logical Left Shift	1	11	0101	dfff	ffff	C, Z	2
LSRF	f, d	Logical Right Shift	1	11	0110	dfff	ffff	C, Z	2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRW	–	Clear W	1	00	0001	0000	00xx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	2
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	2
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		2
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z	2
SUBWFB	f, d	Subtract with Borrow W from f	1	11	1011	dfff	ffff	C, DC, Z	2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	2
BYTE ORIENTED SKIP OPERATIONS									
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1, 2

Utasításkészlet - 2

BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		2
BIT-ORIENTED SKIP OPERATIONS									
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		1, 2
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		1, 2
LITERAL OPERATIONS									
ADDLW	k	Add literal and W	1	11	1110	kkkk	kkkk	C, DC, Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLB	k	Move literal to BSR	1	00	0000	001k	kkkk		
MOVLP	k	Move literal to PCLATH	1	11	0001	1kkk	kkkk		
MOVLW	k	Move literal to W	1	11	0000	kkkk	kkkk		
SUBLW	k	Subtract W from literal	1	11	1100	kkkk	kkkk	C, DC, Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

Utasításkészlet - 3

CONTROL OPERATIONS									
BRA	k	Relative Branch	2	11	001k	kkkk	kkkk		
BRW	–	Relative Branch with W	2	00	0000	0000	1011		
CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CALLW	–	Call Subroutine with W	2	00	0000	0000	1010		
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
RETFIE	k	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	0100	kkkk	kkkk		
RETURN	–	Return from Subroutine	2	00	0000	0000	1000		
INHERENT OPERATIONS									
CLRWDT	–	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$, $\overline{PD}$	
NOP	–	No Operation	1	00	0000	0000	0000		
OPTION	–	Load OPTION_REG register with W	1	00	0000	0110	0010		
RESET	–	Software device Reset	1	00	0000	0000	0001		
SLEEP	–	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$, $\overline{PD}$	
TRIS	f	Load TRIS register with W	1	00	0000	0110	0fff		
C-COMPILER OPTIMIZED									
ADDFSR	n, k	Add Literal k to FSRn	1	11	0001	0nkk	kkkk		
MOVIW	n mm	Move Indirect FSRn to W with pre/post inc/dec modifier, mm	1	00	0000	0001	0nmm kkkk	Z	**2, 3**
	k[n]	Move INDFn to W, Indexed Indirect.	1	11	1111	0nkk	1nmm	Z	**2**
MOVWI	n mm	Move W to Indirect FSRn with pre/post inc/dec modifier, mm	1	00	0000	0001	kkkk		**2, 3**
	k[n]	Move W to INDFn, Indexed Indirect.	1	11	1111	1nkk			**2**

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

3: See Table in the MOVIW and MOVWI instruction descriptions.

Címzési módok

Címzés: a művelet operandusának kijelölése

- Bit címzés (BSF LATA,7)
- Direkt címzés (MOVF PORTA,W)
- Indirekt címzés (MOVWF INDF0)
- Bázisrelatív címzés konstans eltolással (MOVIW [5]FSR0)
- Indirekt címzés automatikus cím módosítás (pre/post inkremens, dekremens) (MOVIW FSR0++)
- Relatív címzés (BRW)
- Implicit címzés (CLRW, RETURN)
- Immediate címzés (közvetlen adat)

Bit címzés

- Bármely memóriarekesz elérhető bitenként is
- Meg kell adni a regiszter címét és a bitpozíciót

```
BCF   LATA,2 ; LATA.2=0
BSF   LATA,3 ; LATA.3=1
BTFSC LATA,2 ; skip, ha LATA.2==0
BTFSS LATA,3 ; skip, ha LATA.3==1
```

Bit-oriented file register operations

13 … 10	9 … 7	6 … 0
OPCODE	b (BIT #)	f (FILE #)

b = 3-bit bit address
f = 7-bit file register address

Direkt címzés

- Az OPkód közvetlenül tartalmazza az operandus címének alsó 7 bitjét
- A felső címbitek a BSR regiszterben
- Eredmény kerülhet a W-be, vagy a megcímzett rekeszbe (F)
- `MOVLB 0  ; 0.bank kiválasztása`
- `MOVF  PORTA,W ; W=PORTA`

Byte-oriented file register operations

13 … 8	7	6 … 0
OPCODE	d	f (FILE #)

d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

Indirekt címzés

- Cím: FSR0 vagy FSR1 címtároló regiszterben
- Utasítás INDF0 vagy INDF1 pszeudó regiszterre hivatkozik

```
MOVLW    HIGH regaddr
MOVWF    FSR0H       ; cím felső bájt
MOVLW    LOW  regaddr
MOVWF    FSR0L       ; cím alsó bájt
MOVF     INDF0,W     ; W=M[FSR0H:FSR0L]
```

Byte-oriented file register operations

13 – 8	7	6 – 0
OPCODE	d	f (FILE #)

d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

INDF0 vagy INDF1

Bázisrelatív címzés

- A cím bázisát az FSR0 vagy FSR1 tárolja,
- ehhez adódik egy, az utasításban tárolt konstans
- A művelet végén nem változik az FSR értéke
- Csak a W-be kerülhet az adat
- Összetett adatszerkezetek kezelésekor előnyös

```
MOVIW  [3]FSR0  ; W=M[3+FSR0H:FSR0L]
```

FSR Offset instructions

13 … 7	6	5 … 0
OPCODE	n	k (literal)

n = appropriate FSR
k = 6-bit immediate value

Indirekt címzés automatikus cím módosítással

- FSR0 vagy FSR1 értéke módosul a hozzáférés során
- Predekremens vagy preinkremens esetén a hozzáférés az új cím szerinti

```
MOVIW    FSR0++     ; W=M[FSR0H:FSR0L], FSR0=FSR0+1
MOVIW    ++FSR0     ; FSR0=FSR0+1, W=M[FSR0H:FSR0L]
MOVIW    FSR0--     ; W=M[FSR0H:FSR0L], FSR0=FSR0-1
MOVIW    --FSR0     ; FSR0=FSR0-1 W=M[FSR0H:FSR0L]
```

FSR Increment instructions

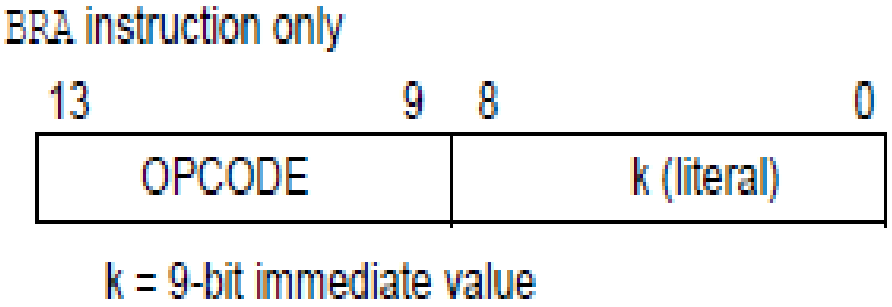
13 … 3	2	1 … 0
OPCODE	n	m (mode)

n = appropriate FSR
m = 2-bit mode value

Relatív címzés

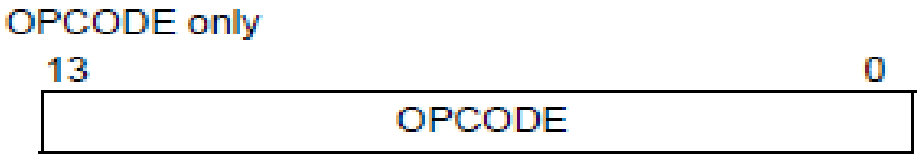
- A cím a PC pillanatnyi állapotához képest előjeles összeadással

```
BRA     címke
```



- A cím a PC pillanatnyi állapotához képest előjel nélküli összeadással

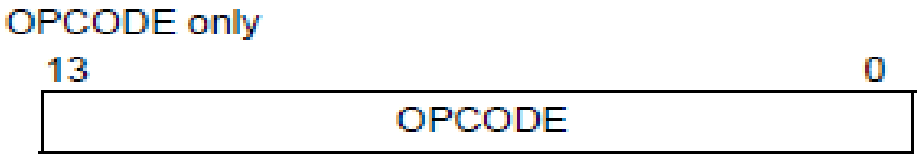
```
BRW     ; cím=PC+W+1
```



Implicit címzés

- A cím a műveletből következik, nem az OPkód cím részéből

```
RETURN ;  PC=[STACK]
CLRW   ;  W=0
CALLW  ;  PC=PCLATH:W
```



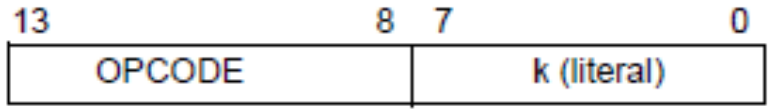
Immediate címzés

- Az OPkód tartalmazza a konstans operandust

```
GOTO    címke ; PC=cimke
MOVLW   12    ; W=12
MOVLB   12    ; BSR=12
MOVLP   12    ; PCLATH=12
```

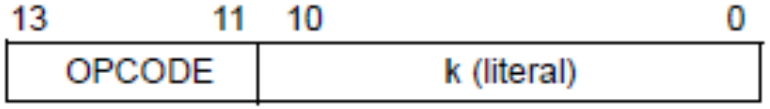
Literal and control operations

General



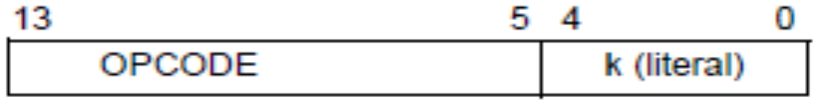
k = 8-bit immediate value

CALL and GOTO instructions only



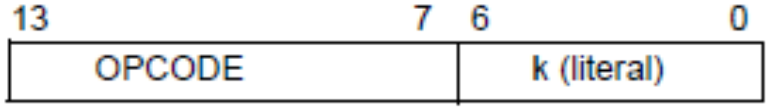
k = 11-bit immediate value

MOVLB instruction only



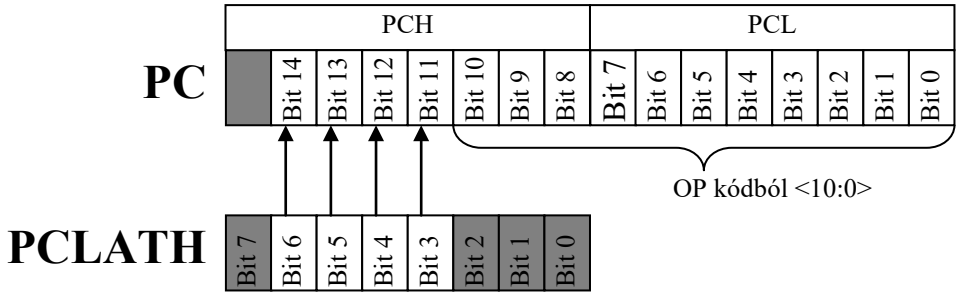
k = 5-bit immediate value

MOVLP instruction only

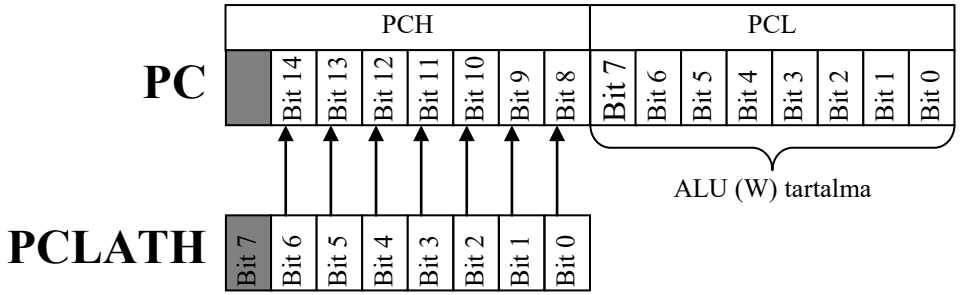


k = 7-bit immediate value

PC módosítása



CALL, GOTO 11bites cím
→ többi a PCLATH-ban



CALLW,

PCL írásakor 8bit a W-ben
→ többi a PCLATH-ban

Utasítások csoportosítása

- Adatmozgató utasítások
- Aritmetikai műveletek
- Logikai műveletek
- Bitműveletek
- Vezérlés átadás
- Vezérlő műveletek

Adatmozgató utasítások

- **MOVF**→F másolása W-be, vagy önmagára Z
- **MOVWF**→W másolása f-be -
- SWAPF→alsó és felső 4 bit felcserélése -
- MOVLB→Konstans a BSR-be -
- MOVLP→konstans a PCLATH-ba -
- **MOVLW**→konstans a W-be -
- MOVIW→indirekt adatmozgatás W-be FSR0 vagy FSR1 szerint Z
- MOVWI→indirekt adatmozgatás W-ből FSR0 vagy FSR1 szerint Z

Aritmetikai műveletek

• ADDWF→d=W+f	C,DC,Z
• ADDWFC→d=W+f+C	C,DC,Z
• ASRF→d=f>>1, előjel kiterjesztéssel	C,Z
• DECF→f=f-1	Z
• INCF→f=f+1	Z
• SUBWF→f=f-W	C,DC,Z
• SUBWFB→f=f-W-C	C,DC,Z
• ADDLW→W=W+k	C,DC,Z
• SUBLW→W=k-W	C,DC,Z

Logikai műveletek

• ANDWF→f=W&F	Z
• LSLF→ C←f←0	C,Z
• LSRF→ 0→f→C	C,Z
• COMF→f=~f	Z
• IORWF→f=f|W	Z
• RLF→C←f←C (körbeforgatás C-n keresztül)	C
• RRF→C→f→C (körbeforgatás C-n keresztül)	C
• XORWF→f=F^W	Z
• ANDLW→W=W&k	Z
• IORLW→W=W|k	Z
• XORLW→W=W^k	Z

Bitműveletek

- BCF→f.i=0 -
- BSF→f.i=1 -
- BTFSC→skip, ha f.i==0 -
- BTFSS→skip, ha f.i==1 -

f: memóriaváltozó vagy periféria regiszter 7 bites címe

i: 0..7, bit kiválasztás, 0. bit LSB

Vezérlés átadás

- DECFSZ→d=f-1, skip, ha nulla -
- INCFSZ→d=f+1, skip, ha nulla -
- BTFSC→skip, ha f.i==0 -
- BTFSS→skip, ha f.i==1 -
- BRA→9bites előjeles relatív ugrás konstans távolságra -
- BRW→8bites PC relatív ugrás, PC=PC+W -
- CALL→szubrutin hívás, PC→stack -
- CALLW→szubrutin hívás, PC=PCLATH:W -
- GOTO→ugrás -
- RETFIE→visszatérés+IT eng., stack→PC -
- RETLW→visszatérés, W=k, stack→PC -
- RETURN→visszatérés, stack→PC -

Vezérlő utasítások

• CLRWDT→WDT időzítő törlése	TO,PD
• NOP→egy út. ciklus késleltetés	-
• RESET→újraindítás	-
• SLEEP→kis fogyasztású módba lépés	TO,PD

TO: time out jelzőbit

PD: power down jelzőbit

Vezérlési szerkezetek

- Aritmetikai és logikai műveletek mellékhatásként jelző biteket (flag) is állítanak→
- Státusz regiszter (STATUS)

U-0	U-0	U-0	R-1/q	R-1/q	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	$\overline{TO}$	$\overline{PD}$	Z	DC(1)	C(1)
bit 7							bit 0

- STATUS regiszter, FLAG bitek ↑
- Feltételes elágazások (`BTFSC, BTFSS`)
- Ugrótáblák (`BRW`)
- Iteráció (`DECFSZ, INCFSZ`)
- Utasítás táblában ellenőrizni kell, hogy melyik utasítás milyen flagbiteket állít!

Elágazás megvalósítása 1

if VAR_A==VAR_B:

egyik

else:

masik

```
   MOVF          VAR_A,W
   SUBWF         VAR_B,W ; W=VAR_B-VAR_A
   BTFSC         STATUS,Z ; Z=1, ha egyenlő
   GOTO          ide; Z=1 ág
   GOTO          oda; Z=0 ág
ide:
      egyik..
   GOTO vége
oda:
      masik..
vége:
```

Elágazás megvalósítása 2

if VAR_A>VAR_B :

egyik

else:

masik

```
MOVF        VAR_A,W
SUBWF       VAR_B,W ; W=VAR_B-VAR_A
BTFSC       STATUS,C ; C=1, ha A>B
GOTO        ide; C=1, azaz egyik ág
GOTO        oda; C=0, azaz másik ág
```

…….

Elágazás megvalósítása 3

switch(VAR_A) { *(ilyen nincs Pythonban, ami kár mert hatékony…)*
case 0: Goto Rut0;
case 1: Goto Rut1;
} ;→ assemblyben:

```
MOVF     VAR_A,W
BRW      ; PC=PC+1+W
GOTO          Rut0; W=0 ág
GOTO          Rut1; W=1 ág
```

Ügyelni kell a túlcímzés elkerülésére! Pl: maszkolással:

```
MOVF     VAR_A,W
ANDLW    B'00000001'     ; Csak egy bitet hagyunk
BRW      ; PC=PC+1+W
```

Máshogy is számolható az ugrás helye (pl /4, …)

Ciklus megvalósítása

```
for x in range(2,0,-1):
    dolgozik();
```

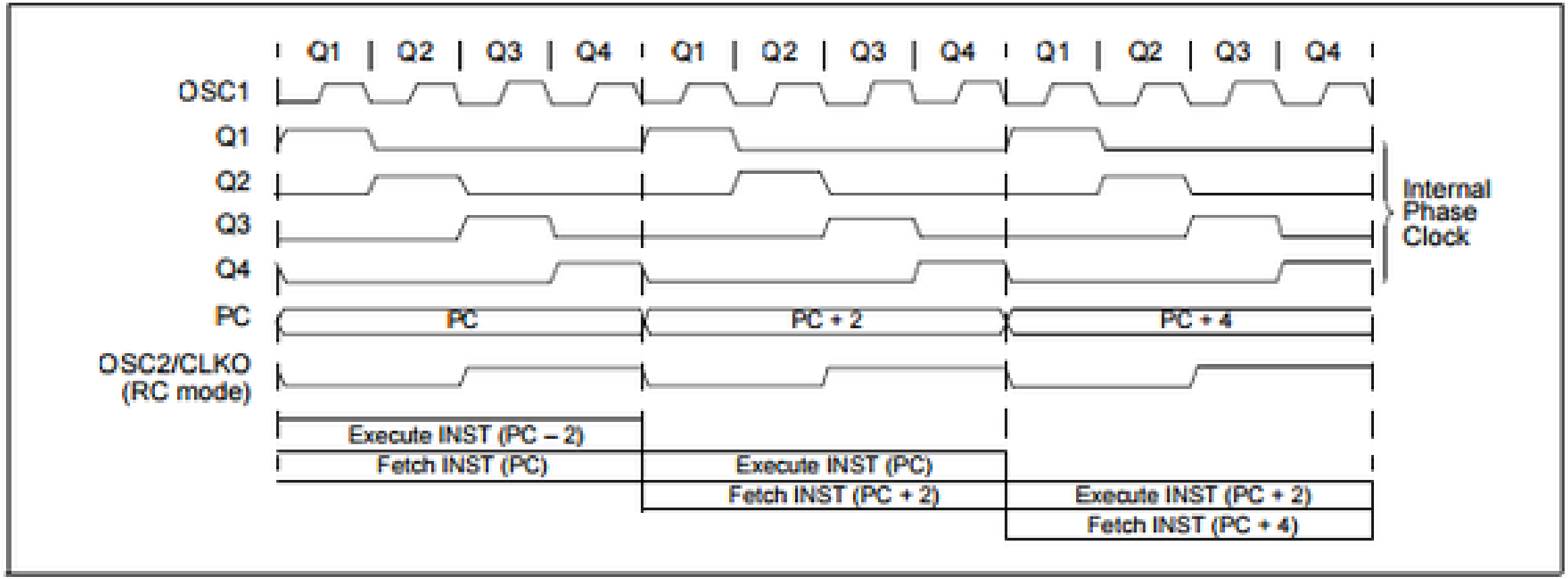
```
    MOVLW       D'02'
    MOVWF       zz; ZZ=2
címke:
    CALL        dolgozik
    DECFSZ      zz,F ; zz--, skip ha zz==0
    GOTO        címke
    …folytatás…
```

Szubrutin készítése

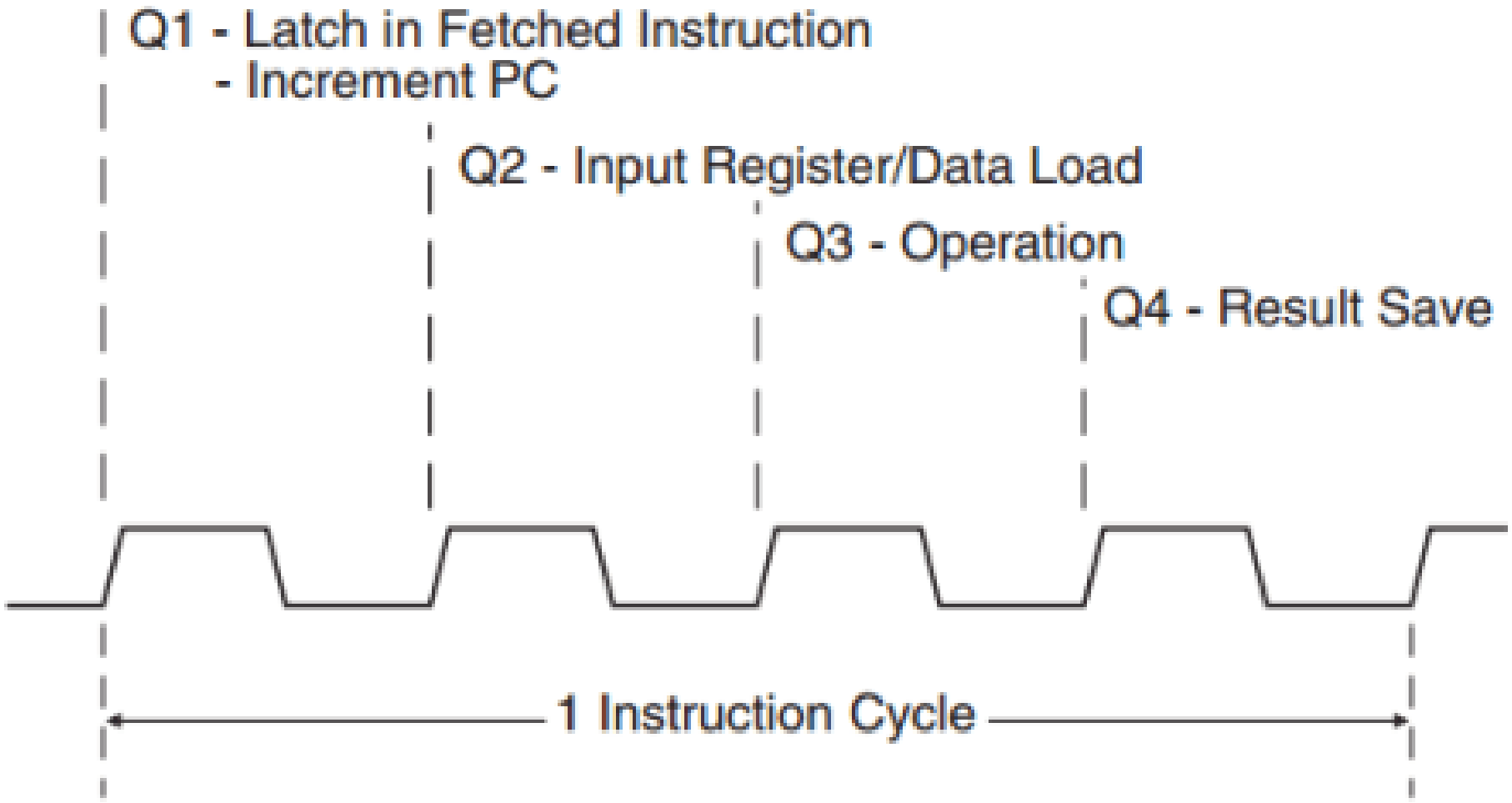
- Több helyről is meghívható eljárások készítése
- Meghívás: CALL utasítással
- Aktuális PC→STACK
- Korlátozott egymásba ágyazás <16
- Meghívott szubrutin a futás végeztével visszatér→RETURN vagy RETLW
- Paraméter átadás W-ben, vagy memóriában

Az utasítás végrehajtás időzítése

FIGURE 5-3: CLOCK/INSTRUCTION CYCLE



1 utasításciklus összetétele (4 fázis)



Szoftveres késleltetés

- Minden utasításnak fix végrehajtási ideje van (pl.: 8MHz instruction clock esetén Tcy=125ns
- Szervezzünk visszaszámoló ciklust

```
Delay15us:
  MOVLW    D'38'   ; 1+2
D15CK:
  DECFSZ   WREG,W  ; 1 /2
  GOTO     D15CK   ; 2
  RETURN           ; 2
```

```
Delay= (5+N*3)*125ns
15us/125ns=120 ciklus
N=(120-5)/3=38.3
```

Szoftveres késleltetés

- Hosszabb időzítéshez több egymásba ágyazott ciklus

```
Delay1ms:
        MOVLW   D`66'
        MOVWF   DTMP1
D1MSCK:
        CALL    Delay15us
        DECFSZ  DTMP1,F
        GOTO    D1MSCK
        RETURN
```

Összehasonlítás konstanssal

```
;  ALARM szubrutin hívása, ha ERTEK>8
CHECK_ALARM:
  MOVF     ERTEK,W    ; W = ERTEK
  SUBLW    D'08'      ; W = 8-ERTEK
  BTFSC    STATUS,C   ; C = 1, ha negatív
  CALL     ALARM      ; Hívás, ha C=1
  RETURN
```

SUBLW D'08' lehetne akár ADDLW -D'08'

Memória másolása

```
MEMCOPY:
    ; FSR0: forrás mutató
    ; FSR1: cél mutató
    ; W: darabszám
    MOVWF   ByteCNT
MCPCK:
    MOVIW   FSR0++  ; M[FSR0H:FSR0L]→W
    MOVWI   FSR1++  ; W→M[FSR1H:FSR1L]
    DECFSZ  ByteCNT,F
    GOTO    MCPCK
    RETURN
```

PORT kezelés

```
BCF TRISA,5 ; RA5 kimenet
BSF LATA,5  ; RA5=1
NOP
BCF LATA,5  ; RA5=0
```

Várakozás gombnyomásra

PORTC.4=0→lenyomták→visszatér

```
VAR_0:
  BTFSC  PORTC,4
  BRA    VAR_0
  RETURN
```

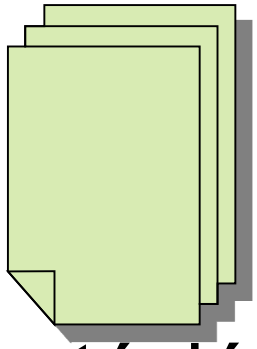
Táblázat tárolása programmemóriában

```
; indexelő szubrutin, INDEX_TABLA [W] értéket adja vissza
INDEX_TABLA:
  BRW
  RETLW   D'12'; INDEX_TABLA [0] érték = 12
  RETLW   H'22'; INDEX_TABLA [1] érték = 34
……                ; további értékek…
```

Táblázat létrehozható a DT direktívával is

```
INDEX_TABLA:
  BRW
  DT D'12,H'22'
```

A témához kapcsolódó anyagok



- Microchip weboldala

  http://www.microchip.com
- MPLABX fejlesztői környezet és dokumentáció letöltése

  https://www.microchip.com/mplab/mplab-x-ide
- Online help a mikrokontrollerhez és a fejlesztői környezethez

  http://microchipdeveloper.com/mcu1102:start
- Tárgy honlapján

  https://www.iit.bme.hu