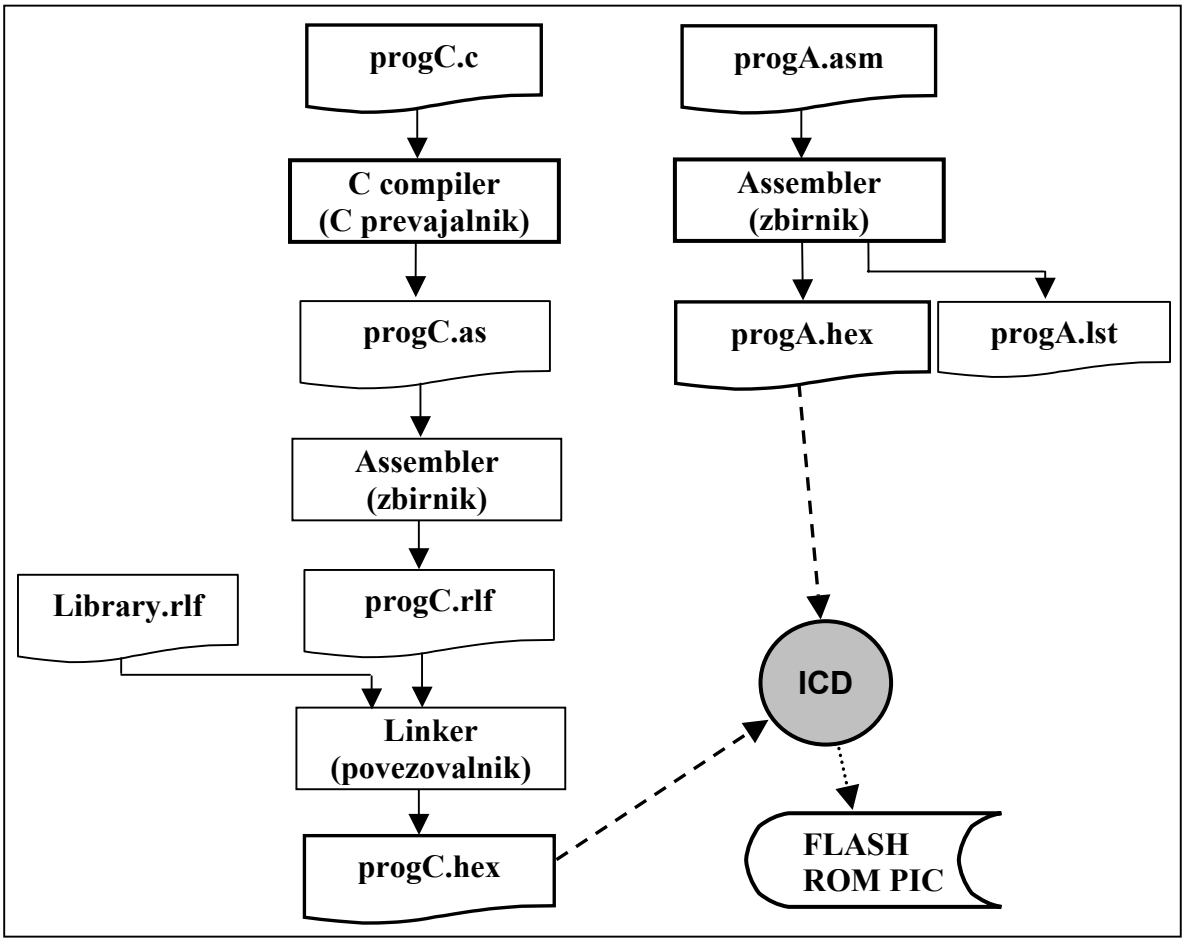
2. Programiranje PIC16F87x v zbirnem jeziku

Naučili se boste oblikovanja programa v zbirnem jeziku (splošni format) ter osnovne programske ukaze in njihov pomen. Ko boste obvladali nekatere podrobnosti, boste v nadaljevanju lahko uporabljali zgolj zgoščeno tabelo nabora vseh 35 ukazov.

Programiranje PIC16F876 lahko poteka v **zbirnem jeziku** (Assembler, integriran v **MPLAB**) [6] ali v **programskem jeziku C** [5][12] (potrebno ga je dodatno naložiti, npr. [14]).

Izvorni program [1] (angl.: Source code programm) napišemo s pomočjo osebnega računalnika v enem od tekstovnih urejevalnikov (Beležnica, Notepad, UltraEdit, ...) ali kar v urejevalniku v **MPLAB** programskem okolju. Izvorni program v zbirnem jeziku ima obvezno končnico **.asm**, izvorni program v C-jeziku pa **.c**.

Diagram poteka (Slika 2-1) prikazuje postopek pretvarjanja (prevajanja) izvornega programa v C-jeziku [14] ali zbirnem jeziku [9] v datoteko izvršljive oblike (.hex), ki jo lahko s posredovanjem **ICD** modula (ali simulatorja) naložimo v pomnilnik mikrokrmilnika v ciljnem sistemu (ali kar zaženemo z **MPLAB Simulator** orodjem).



Slika 2-1: Postopek tvorjenja izvršljivega programa

V kolikor **prevajalnik** ali **zbirnik** zazna **sintaktično napako**, se postopek prekine, kar pomeni, da mora uporabnik odpraviti napake ter **ponovno sprožiti proces**

prevajanja (angl.: Build all) **in nalaganja** (angl.: Download) ter **programiranja** (angl.: Program) **FLASH ROM pomnilnika** v ciljnem mikrokrmilniku. Če se pri izvajanju programa na ciljnem sistemu pokažejo **logične napake**, je seveda **potrebno odpraviti le-te in ponoviti postopek prevajanja.**

2.1. Program v zbirnem jeziku

V splošnem imajo programi, ki jih pišemo (kodiramo) v zbirnem jeziku, dokaj strogo sintakso. Izvorni program je sestavljen iz zaporedja vrstic, ki (ločeno) predstavljajo:

- **programsko izvedljivo vrstico;**
- vrstico z psevdoukazom ali zbirniško smernico;
- komentarsko ali prazno vrstico.

Format prvih dveh tipov vrstic je natanko določen (Program 2-1). Priporočljivo je, da stolpce (polja) ločujemo z znakom 'TAB', sicer je po pravilih za ločevanje polj dovolj en presledek. Vsaka vrstica je razdeljena na več polj [1]:

- **Polje1** se imenuje področje **označb vrstice (Label)**. **Obvezno** se vpiše **v 1. stolpec** vrstice in se **začne s črko** (alfabetski znak brez šumnikov) ali s podčrtajem '_', nadaljuje se lahko tudi s številkami. Število znakov je omejeno na 32. Priporočljivo je uporabljati smiselna imena (npr.: `temperatura`), vendar se moramo izogibati rezerviranim imenom (imena registrov, imena ukazov). Program »**zbirnik«** ločuje velike in male črke.
- **Polje2** vsebuje **mnemonična imena ukazov procesorja** ali **makro ukaze** (za programsko izvedljive vrstice) oziroma **psevdoukazna imena** (za vrstice z zbirniškimi smernicami). **Mnemonična imena ukazov** definira proizvajalec procesorja (za Microchip mikrokrmilnike srednje kategorije PIC16Fxxx je potrebno poznati le 35 ukazov). Uporabljajo se lahko tako **velike kot male črke**.
- **Polje3** vsebuje **operande iz nabora načinov naslavljanja** (za programsko izvedljive vrstice) oziroma **definirana simbolna imena** ali **vrednosti** (za vrstice z zbirniškimi smernicami). Številčne vrednosti lahko zapišemo v različnih številskih formatih: desetiško `.39` ali `d'39'`, šestnajstiško `0x7E` ali `h'7E'` ali `7Eh`, dvojiško `b'10110001'`, ASCII znakovne konstante `'G'` ali `a'GcX'`.
- **Polje4** (ki ni obvezno) se začenja s podpičjem '`;`' in se nadaljuje s poljubnim komentarjem (vsekakor je **priporočljivo pisati smiselne komentarje** v kontekstu posledice izvršitve ukaza).

polje1	polje2	polje3	polje4
	list	P=16F876	; psevdoukaz za določitev tipa procesorja
sprem	equ	0x20	; psevdoukaz za prireditev naslova simbolu
	org	0	; psevdoukaz: določitev vrednosti programskega števca PC=0
	clrf	sprem	; ukaz za brisanje sprem
zanka	incf	sprem, f	; ukaz za povečanje sprem v registru f za 1
	goto	zanka	; ukaz za brezpogojni skok na označbo
	end		; psevdoukaz za konec programa
'TAB'	'TAB'	'TAB'	

Program 2-1: Splošni format izvornega programa v zbirnem jeziku

Če v 1. stolpcu (ali kasneje) vpišemo znak podpičje '`;`' , ki mu sledi poljuben tekst, dobimo vrstico z izključnim komentarjem. **Zaradi povečanja preglednosti in čitljivosti programa je smiselno vstavljati** tudi **prazne vrstice**.

Zbirnik (Assembler) (angl.: Assembling, »zbiranje«) pretvarja izvorno datoteko (npr.: progA.asm) sekvenčno – vrstico za vrstico. Vsako **programsko izvedljivo vrstico** pretvori v **14-bitno besedo (strojna koda),** pri čemer uporablja **vrstice s psevdoukazi (zbirniškimi smernicami)** kot navodilo za tvorjenje strojne kode. Postopek poteka v dveh korakih, kajti potrebno je upoštevati tudi vrednosti simbolov, ki so definirani kasneje. Če ni sintaktičnih napak, se tvori datoteka s šestnajstiško zapisanimi kodami (progA.hex), katere vsebino lahko prenesemo (programiramo) v FLASH ROM pomnilnik mikrokrmilnika v ciljnem sistemu.

Registrska struktura je (na prvi pogled) precej skromna, kajti v programskem modelu je na voljo **en sam 8-bitni splošno uporaben register z oznako W** (**W**ork). Vendar je potrebno upoštevati tudi, da je na voljo kar **368 datotečnih registrov** (**F**ile register), ki jih v ukazih lahko povečini enakovredno uporabljamo tako za izvorne kot ciljne operande.

Trajanje ukazov je tipično en strojni cikel (4 urini cikli, kar pomeni **200 ns** pri 20 MHz taktu), pri čemer je tudi nekaj ukazov, ki lahko pogojno trajajo dva cikla in nekaj, ki vedno trajajo dva cikla (zgoščen opis ukazov prikazuje Tabela 2-1).

2.2. Podrobnejši opis ukazov

Nabor ukazov zbirnega jezika obsega **35 mnemoničnih imen ukazov** v skladu z RISC arhitekturo [2][4][6][8]. Podrobno so opisani v nadaljevanju.

Vsak ukaz (Tabela 2-1) v **programsko izvedljivih vrsticah** se prične v **polju2** z zapisom mnemoničnega imena ukaza. Temu sledi **polje3**, kamor vpisujemo operande. Nekateri ukazi ne potrebujejo operandov in nekateri vsebujejo le en operand: izvorni operand (konstanta **k**) ali ciljni operand (datotečni register **f**).

Večina ukazov je takšnih, da je prvi operand izvorni (datotečni register **f**), drugi pa ciljni (bit **b** ali pa vrednost **d**, ki pomeni izbiro cilja med **W** in **f**). V splošnem je **»izvorni operand«** tisti, ki se **ne spreminja**, ampak samo bere, **»ciljni operand«** pa tisti, kamor se **shranjuje rezultat operacije**. Upoštevati je potrebno, da se v skladu z rezultatom operacije (predvsem pri aritmetičnih in logičnih ukazih) spreminjajo nekatere zastavice v registru **STATUS**.

2.2.1. Seznam in opisi ukazov v abecednem vrstnem redu:

Pomen uporabljenih simbolov v opisih posameznih ukazov:

k - konstanta
W - delovni register (**W**ork register)
f - pomnilniška lokacija (**f**ile register)
<b> - bit (mesto - pozicija v registru)

d - označba za izbiro cilja (W - 0 ali f - 1)
PC - programski števec
TOS - števec (vrh) sklada
WDT - Watchdog Timer
C, DC, Z - zastavice v **STATUS** registru (**C**arry, **D**ecimal **C**arry, **Z**ero)
[labela] - simbolna označba vrstice (začne se v 1. stolpcu polja1 in ni obvezna)

ADDLW Seštej konstanto in W

Sintaksa:	[labela] ADDLW k
Operandi:	$0 \le k \le 255$
Operacija:	(W) + k $\rightarrow$ (W)
Spreminja status:	C, DC, Z
Opis:	Vsebini registra W se prišteje osem bitna konstanta 'k', rezultat se shrani v register W.

ADDWF Seštej W in f

Sintaksa:	[labela] ADDWF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(W) + (f) $\rightarrow$ (cilj)
Spreminja status:	C, DC, Z
Opis:	Vsebini registra W se prišteje vsebina registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

ANDLW Konstanta IN W

Sintaksa:	[labela] ANDLW k
Operandi:	$0 \le k \le 255$
Operacija:	(W) .IN. k $\rightarrow$ (W)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden logični bitni IN (AND) z osem bitno konstanto 'k', rezultat se shrani v register W.

ANDWF W IN f

Sintaksa:	[labela] ANDWF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(W) .IN. (f) $\rightarrow$ (cilj)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni IN z vsebino registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

BCF Zbriši bit b v f

Sintaksa:	[labela] BCF f,b
Operandi:	$0 \le f \le 127$,
	$0 \le b \le 7$
Operacija:	0 $\rightarrow$ (f<b>)
Spreminja status:	Nobenega
Opis:	Bit 'b' v registru 'f' je zbrisan.

BSF Postavi bit b v f

Sintaksa:	[labela] BSF f,b
Operandi:	$0 \le f \le 127$,
	$0 \le b \le 7$
Operacija:	1 $\rightarrow$ (f<b>)
Spreminja status:	Nobenega
Opis:	Bit 'b' v registru 'f' je postavljen.

BTFSS	**Testiraj bit, preskoči če je '1'**
Sintaksa:	[labela] BTFSS f,b
Operandi:	$0 \le f \le 127$, $0 \le b \le 7$
Operacija:	Preskoči naslednji ukaz, če je (f<b>) = 1
Spreminja status:	Nobenega
Opis:	Če je bit 'b' v registru 'f' enak '0', se izvede naslednji ukaz. Če je bit 'b' enak '1', je naslednji ukaz preskočen in se namesto njega izvede NOP, kar povzroči, da se ukaz izvaja dva urina cikla.

BTFSC	**Testiraj bit, preskoči če je '0'**
Sintaksa:	[labela] BTFSC f,b
Operandi:	$0 \le f \le 127$, $0 \le b \le 7$
Operacija:	Preskoči naslednji ukaz, če je (f<b>) = 0
Spreminja status:	Nobenega
Opis:	Če je bit 'b' v registru 'f' enak '1', se izvede naslednji ukaz. Če je bit 'b' enak '0', je naslednji ukaz preskočen in se namesto njega izvede NOP, kar povzroči, da se ukaz izvaja dva urina cikla.

CALL	**Klic podprograma**
Sintaksa:	[labela] CALL k
Operandi:	$0 \le f \le 2047$
Operacija:	(PC) + 1 $\rightarrow$ TOS, k $\rightarrow$ PC<10:0>, (PCLATH<4:3>)$\rightarrow$ (PC<12:11>)
Spreminja status:	Nobenega
Opis:	Kliče podprogram. Najprej je naslov povratka iz podprograma (PC+1) shranjen v sklad. Enajst bitni naslov je naložen v bite PC <10:0>. Zgornja bita PC sta naložena iz PCLATH. Ukaz CALL traja dva cikla.

CLRF	**Zbriši f**
Sintaksa:	[labela] CLRF f
Operandi:	$0 \le f \le 127$
Operacija:	00h $\rightarrow$ (f), 1 $\rightarrow$ Z
Spreminja status:	Z
Opis:	Vsebina registra 'f' je zbrisana in bit (Z) v statusnem registru je postavljen na 1.

CLRW	**Zbriši W**
Sintaksa:	[labela] CLRW
Operandi:	Nima
Operacija:	00h $\rightarrow$ (W), 1 $\rightarrow$ Z
Spreminja status:	Z
Opis:	Vsebina registra W je zbrisana in bit (Z) v statusnem registru je postavljen na 1.

CLRWDT Zbriši Watchdog Timer

Sintaksa: [labela] CLRWDT

Operandi: Nima

Operacija: 00h → WDT,

0 → WDT prescaler,

1 → $\overline{\text{TO}}$

1 → $\overline{\text{PD}}$

Spreminja status: $\overline{\text{TO}}$, $\overline{\text{PD}}$

Opis: Ukaz CLRWDT resetira Watchdog Timer. Resetira tudi njegov prescaler. Statusna bita $\overline{\text{TO}}$ in $\overline{\text{PD}}$ sta postavljena.

COMF Komplement f

Sintaksa: [labela] COMF f,d

Operandi: $0 \le f \le 127$,

$d \in [0,1]$

Operacija: $(\overline{f})$ → (cilj)

Spreminja status: Z

Opis: Izračun eniškega komplementa registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

DECF Zmanjšaj f za 1

Sintaksa: [labela] DECF f,d

Operandi: $0 \le f \le 127$,

$d \in [0,1]$

Operacija: (f) - 1 → (cilj)

Spreminja status: Z

Opis: Zmanjšaj register 'f' za 1. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

DECFSZ Zmanjšaj f za 1, preskoči če je 0

Sintaksa: [labela] DECFSZ f,d

Operandi: $0 \le f \le 127$,

$d \in [0,1]$

Operacija: (f) - 1 → (cilj);

preskoči, če je rezultat enak 0

Spreminja status: Nobenega

Opis: Zmanjšaj register 'f' za 1. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

Če je rezultat enak 1, se izvede naslednji ukaz. Če je rezultat enak 0, je naslednji ukaz preskočen in se namesto njega izvede NOP, kar povzroči, da se ukaz izvaja dva urina cikla.

GOTO Brezpogojni skok

Sintaksa: [labela] GOTO k

Operandi: $0 \le f \le 2047$

Operacija: k → PC<10:0>,

(PCLATH<4:3>)→ (PC<12:11>)

Spreminja status: Nobenega

Opis: GOTO sproži brezpogojni skok (vejitev). Enajst bitni naslov je naložen v bite PC <10:0>. Zgornja bita PC sta naložena iz PCLATH. Ukaz GOTO traja dva cikla.

INCF	**Povečaj f za 1**
Sintaksa:	[labela] INCF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(f) + 1 $\rightarrow$ (cilj)
Spreminja status:	Z
Opis:	Povečaj register 'f' za 1. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

INCFSZ	**Povečaj f za 1, preskoči če je 0**
Sintaksa:	[labela] INCFSZ f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(f) + 1 $\rightarrow$ (cilj);
	preskoči, če je rezultat enak 0
Spreminja status:	Nobenega
Opis:	Poveča register 'f' za 1. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.
	Če je rezultat enak 1, se izvede naslednji ukaz. Če je rezultat enak 0, je naslednji ukaz preskočen in se namesto njega izvede NOP, kar povzroči, da se ukaz izvaja dva urina cikla.

IORLW	**Konstanta ALI W**
Sintaksa:	[labela] IORLW k
Operandi:	$0 \le k \le 255$
Operacija:	(W) .ALI. k $\rightarrow$ (W)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni logični ALI (OR) z osem bitno konstanto 'k', rezultat se shrani v register W.

IORWF	**W ALI f**
Sintaksa:	[labela] IORWF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(W) .ALI. (f) $\rightarrow$ (cilj)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni logični ALI (OR) z vsebino registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

MOVF	**Shrani f**
Sintaksa:	[labela] MOVF f,d
Operandi:	$0 \le f \le 127$,
	$d \in [0,1]$
Operacija:	(f) $\rightarrow$ (cilj)
Spreminja status:	Z
Opis:	Vsebina registra f je pomaknjena v ciljni register. Če je d=0, se rezultat shrani v register W, če pa je d=1, se rezultat shrani v register 'f'. Ukaz z d=1 je običajno uporabljen za testiranje registra f, saj vpliva na statusno zastavico Z.

MOVLW	**Shrani k v W**
Sintaksa:	[labela] MOVLW k
Operandi:	$0 \leq k \leq 255$
Operacija:	$k \rightarrow (W)$
Spreminja status:	Nobenega
Opis:	Osem bitna konstanta 'k' se shrani v register W.

MOVWF	**Shrani W v f**
Sintaksa:	[labela] MOVWF f
Operandi:	$0 \leq k \leq 127$
Operacija:	$(W) \rightarrow (f)$
Spreminja status:	Nobenega
Opis:	Vsebina registra W se shrani v register f.

NOP	**Ni operacije**
Sintaksa:	[labela] NOP
Operandi:	Nima
Operacija:	Ni operacije
Spreminja status:	Nobenega
Opis:	ni operacije, traja 1 cikel

RETFIE	**Vrni se iz interrupta**
Sintaksa:	[labela] RETFIE
Operandi:	Nima
Operacija:	TOS $\rightarrow$ PC, 1 $\rightarrow$ GIE
Spreminja status:	Nobenega
Opis:	Vrnitev iz prekinitvene rutine (interrupt).

RETLW	**RETURN s konstanto v W**
Sintaksa:	[labela] RETLW k
Operandi:	$0 \leq k \leq 255$
Operacija:	$k \rightarrow (W)$, TOS $\rightarrow$ PC
Spreminja status:	Nobenega
Opis:	V register W se naloži konstanta 'k'. V programski števec se naloži vrednost z vrha sklada (naslov za vrnitev iz podprograma). Operacija traja dva cikla.

RETURN	**Vrnitev iz podprograma**
Sintaksa:	[labela] RETURN
Operandi:	Nima
Operacija:	TOS $\rightarrow$ PC
Spreminja status:	Nobenega
Opis:	Vrnitev iz podprograma. V programski števec se naloži vrednost z vrha sklada (naslov za vrnitev iz podprograma). Operacija traja dva cikla.

RLF — Rotiraj f v levo skozi Carry

Sintaksa: [labela] RLF f,d

Operandi: $0 \le f \le 127$,

$d \in [0,1]$

Operacija:

Spreminja status: C

Opis: Vsebina registra f je pomaknjena za eno mesto v levo skozi zastavico statusnega registra Carry. Če je d=0, se rezultat shrani v register W, če pa je d=1, se rezultat shrani v register 'f'.

RRF — Rotiraj f v desno skozi Carry

Sintaksa: [labela] RRF f,d

Operandi: $0 \le f \le 127$,

$d \in [0,1]$

Operacija:

Spreminja status: C

Opis: Vsebina registra f je pomaknjena za eno mesto v desno skozi zastavico statusnega registra Carry. Če je d=0, se rezultat shrani v register W, če pa je d=1, se rezultat shrani v register 'f'.

SLEEP — SLEEP način delovanja

Sintaksa: [labela] SLEEP

Operandi: Nima

Operacija: $00h \rightarrow WDT$,

$0 \rightarrow$ WDT prescaler,

$1 \rightarrow \overline{TO}$

$0 \rightarrow \overline{PD}$

Spreminja status: $\overline{TO}$, $\overline{PD}$

Opis: Statusni bit $\overline{PD}$, ki označuje status napajanja je zbrisan, statusni bit $\overline{TO}$, ki označuje pretek časa, je postavljen. Watchdog Timer in njegov »prescaler« (preddelilnik) sta zbrisana. Procesor se preklopi v način delovanja SLEEP z zaustavljenim oscilatorjem.

SUBLW — Odštej W od konstante

Sintaksa: [labela] SUBLW k

Operandi: $0 \le k \le 255$

Operacija: $k - (W) \rightarrow (W)$

Spreminja status: C, DC, Z

Opis: Vsebina registra W se odšteje (po metodi dvojiškega komplementa) od osem bitne konstante 'k', rezultat se shrani v register W.

SUBWF	**Odštej W od f**
Sintaksa:	[labela] SUBWF f,d
Operandi:	$0 \le f \le 127$, $d \in [0,1]$
Operacija:	(f) - (W) → (cilj)
Spreminja status:	C, DC, Z
Opis:	Vsebina registra W se odšteje (po metodi dvojiškega komplementa) od vsebine registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

SWAPF	**Zamenjaj »nibbla« v f**
Sintaksa:	[labela] SWAPF f,d
Operandi:	$0 \le f \le 127$, $d \in [0,1]$
Operacija:	(f<3:0>) → (cilj<7:4>), (f<7:4>) → (cilj<3:0>)
Spreminja status:	-
Opis:	Zgornji in spodnji »nibble« (štirje biti) sta zamenjana. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

XORLW	**Konstanta XOR W**
Sintaksa:	[labela] XORLW k
Operandi:	$0 \le k \le 255$
Operacija:	(W) .XOR. k → (W)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni logični ekskluzivni ALI (XOR) z osem bitno konstanto 'k', rezultat se shrani v register W.

XORWF	**W IN f**
Sintaksa:	[labela] XORWF f,d
Operandi:	$0 \le f \le 127$, $d \in [0,1]$
Operacija:	(W) .XOR. (f) → (cilj)
Spreminja status:	Z
Opis:	Na vsebini registra W je izveden bitni bitni logični ekskluzivni ALI (XOR) z vsebino registra 'f'. Če je 'd' 0, se rezultat shrani v register W, če pa je 'd' 1, se rezultat shrani v register 'f'.

Za razumevanje tematike je potrebno tudi predznanje iz osnov digitalne elektronike (številski sistemi, logične operacije) [3] in osnov programiranja v zbirnem jeziku: [2] in [8]. Dobri opisi programskih ukazov PIC mikrokrmilnikov srednje kategorije so na voljo v [2] in [6] ter na spletnem portalu: http://www.interq.or.jp/japan/se-inoue/e_pic.htm.

Vprašanja za utrjevanje:

S katerim ukazom postavimo (na 1) bite 0, 1 in 2 v delovnem registru W ?

Rešitev:

```
IORLW b'00000111' ;'ALI' log. oper. med istoleznimi biti, visjih 5 bitov v W se ohrani,
                  ;spodnji trije biti v W se postavijo na 1
```

1. *Koliko različnih ukazov je na voljo ?*
2. *Kakšna polja vsebuje programska vrstica v zbirnem jeziku ?*
3. *S katerim ukazom zbrišemo bit 3 v registru f na naslovu 0x2F ?*
4. *S katerim ukazom opravimo logični IN (AND) med registrom W in f na naslovu 0x22 ?*
5. *S katerim ukazom zaključimo podprogram ?*
6. *S katerim ukazom zbrišem bite 5, 6, in 7 v registru W ?*

2.2.2. Kratek opis ukazov mikrokrmilnika PIC 16F87x

ukaz, operand	opis ukaza	STATUS	cikli
UKAZI, KI SO POVEZANI Z ZLOGI (BYTE)			
ADDWF f, d	seštej **W** in register **f**	C, DC, Z	1
ANDWF f, d	»logična in« operacija med **W** in registrom **f**	Z	1
CLRF f	register **f** naj dobi vrednost 0	Z	1
CLRW -	register **W** naj dobi vrednost 0	Z	1
COMF f, d	eniški komplement registra **f** (zamenjava 0 in 1)	Z	1
DECF f, d	zmanjšaj za ena vsebino registra **f**	Z	1
DECFSZ f, d	zmanjšaj vsebino reg. **f** za ena in preskoči naslednji ukaz, če **f**=0	-	1(2)
INCF f, d	povečaj za ena vsebino registra **f**	Z	1
INCFSZ f, d	povečaj vsebino reg. **f** za ena in preskoči naslednji ukaz, če **f**=0	-	1(2)
IORWF f, d	»logična ali« operacija med **W** in registrom f	Z	1
MOVF f, d	kopiraj vsebino registra **f** (d=0 ->**W**; d=1 ->**f**)	Z	1
MOVWF f	kopiraj vsebino **W** v register **f**	-	1
NOP -	ne naredi ničesar, potroši 1 urin cikel	-	1
RLF f, d	krožno premakni vse bite v registru **f** za eno mesto v levo	C	1
RRF f, d	krožno premakni vse bite v registru **f** za eno mesto v desno	C	1
SUBWF f, d	odštej vsebino reg. **W** od vsebine v reg. **f** (d=**f**-**W**)	C, DC, Z	1
SWAPF f, d	zamenjaj zgornje in spodnje 4 bite v registru **f**	-	1
XORWF f, d	»izključni ali« logična operacija med **W** in reg. **f**	Z	1
UKAZI, KI SO POVEZANI Z BITI			
BCF f, b	briši bit b v registru **f**	-	1
BSF f, b	postavi bit b v registru **f**	-	1
BTFSC f, b	testiraj bit b v registru **f** in preskoči naslednji ukaz, če je bit b=0	-	1(2)
BTFSS f, b	testiraj bit b v registru **f** in preskoči naslednji ukaz, če je bit b=1	-	1(2)
SISTEMSKI IN UKAZI, KI SO POVEZANI S KONSTANTAMI			
ADDLW k	registru **W** prištej konstanto **k**	C, DC, Z	1
ANDLW k	»logična in« operacija med **W** in konstanto **k**	Z	1
CALL k	klic podprograma	-	2
CLRWDT -	vsebina »časovnega stražnika« (**W**atch **D**og **T**imer) naj bo enaka nič	TO, PD	1
GOTO k	skoči na naslov **k** (11-bitni naslov)	-	2
IORLW k	»logična ali« operacija med **W** in konstanto **k**	Z	1
MOVLW k	vpiši konstantno vrednost k v **W** (**W**=k)	-	1
RETFIE -	vrni se iz podprograma za strežbo prekinitvene zahteve	-	2
RETLW k	vrni se iz podprograma s konstantno vrednostjo **k** v registru **W**	-	2
RETURN -	vrni se iz podprograma	-	2
SLEEP -	postavi mikrokrmilnik v stanje mirovanja - »sleep« način	TO, PD	1
SUBLW k	odštej vsebino **W** od konstantne vrednosti **k** (**W**=k-**W**)	C, DC, Z	1
XORLW k	»izključno ali« logična operacija med **W** in konstanto **k**	Z	1

Tabela 2-1: Tabela ukazov s kratkimi opisi

f - ime ali naslov datotečnega registra (**f**ile register)

k - konstantna (8-bitna) vrednost med 0 in 255

d - cilj hrambe rezultata operacije (d=0 - shrani se v **W**, d=1 – shrani se v **f**)

b – pozicija ali oznaka bita med 0 in 7

1(2) – trajanje ukaza je 1 (ali 2) ukazna cikla (pri 20 MHz taktu je trajanje enega ukaznega cikla (4*50) ns = 0,2 µs)