Programovanie mikropočítačov

Vladimír Dronzek

Obsah

Jednočipové mikropočítače

Jednočipovémikropočítače sú integrované obvody, ktoré v sebe zahŕňajú mikroprocesor, pamäť (operačná pamäť – RAM a pevná pamäť – väčšinou EEPROM) a vstupno/výstupné obvody. Ďalej môžu obsahovať potrebné rozhrania pre riadenie periférií, napríklad sériové rozhranie, časovač, A/D resp. D/A prevodníky, ...

Keďže rozsah ich využitia je prakticky neobmedzený používajú sa takmer vo všetkých moderných elektronických zariadeniach ako sú napr. videorekordéry, práčky, šijacie stroje, mikrovlnné rúry, satelitné prijímače, automobily, tlačiarne, klávesnice, hračky, svetelné efekty, meteorologické stanice, riadenie križovatiek ...

1. Základné pojmy

Mikroprocesor – jadro mikropočítača, ktoré riadi celú jeho činnosť. Generuje riadiace signály potrebné pre súčasnú činnosť a spoluprácu všetkých súčastí mikropočítača, riadi výmenu dát medzi jednotlivými časťami a vykonáva jednotlivé inštrukcie. Je to vlastne počítač bez hlavnej pamäte a bez periférnych zariadení.

parametre procesorov:

- taktovacia frekvencia – rýchlosť jadra procesora
- šírka slova – maximálna veľkosť čísla, s ktorým je procesor schopný pracovať
- počet inštrukčných kanálov – maximálny počet inštrukcií, ktoré môže procesor spracovať počas jedného taktu
- efektivita strojového kódu – charakterizuje počet krokov potrebných na vykonanie určitej operácie
- veľkosť adresovacej pamäte – určuje s akou veľkou pamäťou dokáže procesor pracovať
- veľkosť vyrovnávacej pamäte

Pamäť – slúži na uchovávanie údajov. Je rozdelená na operačnú a pevnú pamäť a môže byť rozdelená na pamäť programu a pamäť dát.

Vstupno/výstupné obvody – spájajú mikropočítač s periférnymi zariadeniami.

Zbernica – skupiny vodičov, ktoré majú rovnakú funkciu.

- dátová – slúži na prenos dát
- adresová – slúži na adresovanie pamäte
- riadiaca – slúži na prenos riadiacich signálov

Inštrukcia – jeden riadok programu. Mikropočítač rieši dané úlohy tak, že vykonáva v určitom časovom slede jednotlivé kroky programu (inštrukcie). Každá inštrukcia vykoná jednu relatívne jednoduchú úlohu (spočíta dva čísla, skopíruje obsah pamäte, ...)

CISC –procesory s kompletnou sadou inštrukcií

RISC – procesory s redukovanou sadou inštrukcií

Bit – základná jednotka informácie, ktorá môže nadobúdať dve hodnoty - 0 a 1.

Byte – skupina ôsmich bitov, ktorá môže reprezentovať jedno pamäťové miesto v dátovej pamäti a má priradenú pevnú adresu. Bit s najnižšou váhou je vpravo.

Register – rýchla pamäť procesora, ktorá má definovanú svoju veľkosť (najčastejšie 8 alebo 16bitov). Medzi základné registre patria operačný register a čítač programu.

Čítač – register, do ktorého je možné údaje ukladať i čítať a pri zmene hodinového vstupu zvýši svoju hodnotu o 1.

Časovač – čítač, ktorého vstup je pripojený na vnútorný hodinový výstup oscilátora procesora.

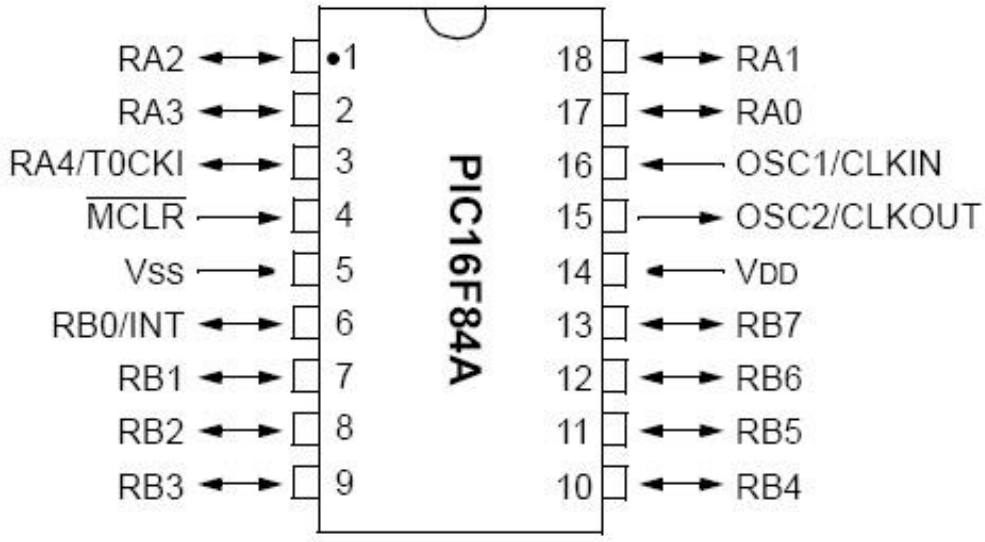
2. Mikrokontrolér PIC16F84

Všetky mikrokontroléry PIC obsahujú procesory typu RISC, teda procesory s redukovanou (obmedzenou) sadou inštrukcií. To znamená že poznajú len málo (35) pomerne jednoduchých inštrukcií, ktorých vykonanie trvá max. 2 strojové takty (cykly). Jeden strojový takt je rýchlosť, ktorou procesor dokáže spracovávať príkazy a je odvodená z frekvencie oscilátora, ktorý určuje rýchlosť procesora. Ak napr. na riadenie procesora využívame oscilátor s frekvenciou 4MHz, jeden takt trvá 1µs. Počas jedného taktu procesor vykoná inštrukciu a pripraví si ďalšiu.

2.1 Základné vlastnosti

- 35 inštrukcií
- 13 vstupno/výstupných pinov (zaťažiteľnosť 25mA)
- veľkosť programovej pamäte – 1024 slov (bytov)
- 68 bytov pamäte Data RAM
- 64 bytov pamäte Data EEPROM
- každá inštrukcia zaberá len jeden strojový cyklus, okrem inštrukcií, ktoré prerušujú beh programu (tie zaberajú dva cykly)
- doba strojového cyklu je rovná prevrátenej hodnote frekvencie oscilátora, ktorá je vydelená štyrmi (pri frekvencii 4MHz - 1/(4Mhz/4) = 1µs.)
- maximálna frekvencia 20MHz
- 1 časovač
- 8 úrovňový zásobník
- priame a relatívne adresovanie
- štyri zdroje prerušenia
  - externý pin RB0/INT
  - pretečenie časovača
  - zmena na pinoch 4-7 na porte B
  - dokončenie zápisu do EEPROM
- sleep mód
- ochrana programu pred prečítaním
- výdrž dát v pamäti > 40 rokov
- 10 000x prepísanie pamäte Flash
- 100 000x prepísanie pamäte EEPROM
- nastavenie oscilátora – RC, XT, HS, LP
- malá spotreba
- < 1µA – 2V standby
- 15µA – 2V, 32kHz
- < 2mA – 5V, 4MHz
- napájacie napätie 2,0 – 5,5V

2.2 Rozloženie vývodov a architektúra PIC16F84



- RA0 – RA4 - vývody portu A (5)
- RB0 – RB7 – vývody portu B (8)
- $\overline{MCLR}$ – reset
- V_{DD} – plus napájacieho napätia
- V_{SS}- zem napájacieho napätia
- OSC1, OSC2 – vývody pre pripojenie oscilátora

2.3 Pripájanievstupov a výstupov k mikrokontroléru

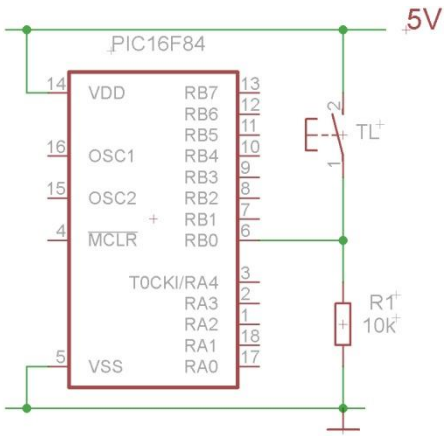
Ako už bolo spomenuté na začiatku, k mikrokontroléru PIC 16F84 je možné pripojiť 13 vstupov, alebo výstupov. Aby bola zabezpečená správna funkcia obvodu a nedošlo k poškodeniu súčiastok (najmä mikrokontroléra), je potrebné, aby boli dodržané podmienky pre pripájanie periférií k mikrokontroléru.

2.3.1 Pripájanie vstupov

Ak je pinmikrokontroléra nastavený ako vstup, hovoríme, že je vo vysokej impedancii. Obvody mikrokontroléra zabezpečia, aby prúd, ktorý pinom prechádza nebol väčší ako ±1µA. Znamienko plus znamená, že prúd do pinu vteká a znamienko mínus, že z neho vyteká. Keďže prúd je veľmi malý, pri výpočtoch sa väčšinou zanedbáva.

Vstupy mikrokontroléra je možné pripojiť dvoma spôsobmi. Pri prvom je pri uvoľnenom tlačidle na vstupe log. úroveň 0 a po stlačení tlačidla log. 1 (tlačidlo funguje ako spínací kontakt). Pri druhom spôsobe je to opačne (tlačidlo funguje ako rozpínací kontakt). Aby mikrokontrolér vyhodnotil stav na vstupe ako log. 0, je potrebné na vstupe zabezpečiť napätie v rozsahu 0 – 0,8V. Pre log. 1 je potrebné napätie 2,4 – 5V.

tlačidlo ako spínací kontakt

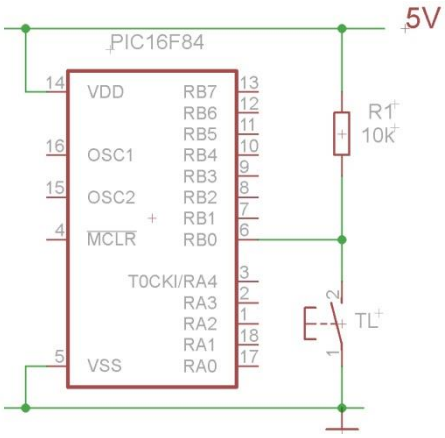


Ak je tlačidlo stlačené, napätie +5V je privedené priamo na pinmikrokontroléra a do pinu tečie prúd maximálne 1µA (dané vnútorným zapojením procesora). Cez odpor R bude tiecť prúd 5V/R. Odpor volíme zvyčajne tak, aby ním tiekol prúd aspoň 10x väčší, ako prúd, ktorý tečie pinom (je to z dôvodu obmedzenia vonkajších rušivých napätí). Z toho vyplýva, že veľkosť odporu nesmie prekročiť hodnotu 5V/(1µA*10) = 500kΩ.

Ak tlačidlo nie je stlačené, bude pin pripojený na zem cez odpor R. Aby napätie na odpore nepresiahlo hodnotu 0,8V (zaručená log. 0), nesmie byť veľkosť odporu väčšia ako 0,8V/1µA=800kΩ.

Na základe týchto dvoch podmienok je zrejmé, že pri návrhu veľkosti rezistora sme obmedzení len jeho hornou hranicou. Je však potrebné myslieť na to, že ak je tlačidlo zopnuté, tečie odporom prúd, ktorý zaťažuje napájací zdroj. Preto volíme hodnotu rezistora tak, aby zdroj nemusel dodávať zbytočne veľký prúd. Vhodná veľkosť rezistora je napríklad 10kΩ (zdroj je zaťažovaný prúdom 0,5mA).

tlačidlo ako rozpínací kontakt



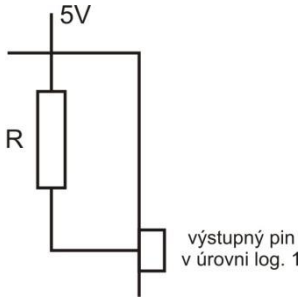
Ak je tlačidlo stlačené, vstupný pin bude pripojený priamo na zem, čo zodpovedá log. úrovni 0. Odporom bude tiecť prúd 5V/R.

Ak tlačidlo nebude stlačené, na pin je privedené napätie 5V cez rezistor R. Aby bola na vstupe zaistená log. 1, nesmie úbytok napätia na rezistore presiahnuť 2,6V (log. 1 je zaručená pre napätie 2,4 – 5V). Z toho vyplýva, že maximálna veľkosť odporu je 2,6V/1μA=2,6MΩ. Pre naše zapojenia môžeme zvoliť rezistor s odporom 10kΩ.

2.3.2 Pripájanie výstupov

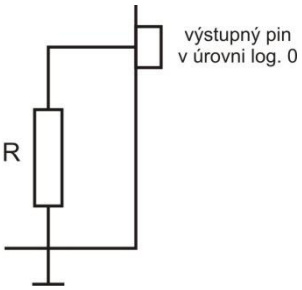
Rovnako ako pri vstupoch je možné aj pin nastavený ako výstupný zapojiť dvoma spôsobmi. Ak napríklad pripojíme LED diódu k mikrokontroléru, tá môže svietiť ak je na výstupe log. 0, alebo log. 1. Závisí to od pripojenia LED diódy. Keďže piny sú nastavené ako výstupy, prúd z nich vychádza a zapojenie musí byť navrhnuté tak, aby neprekročil hodnotu 25mA.

Ak je pin nastavený na úroveň log. 1, môžeme si zapojenie vnútri mikrokontroléra predstaviť takto:



Veľkosť odporu pre úroveň log. 1 je závislá na teplote mikrokontroléra. Pre teplotu 25˚C je jeho veľkosť 90Ω.

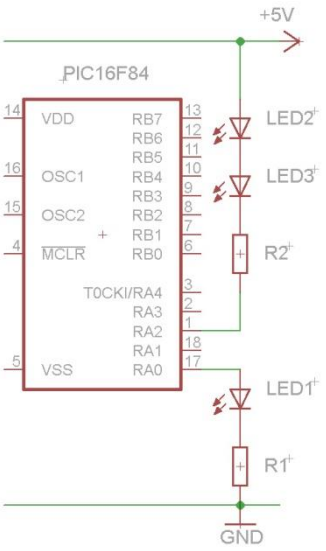
Ak je pin nastavený na úroveň log. 0, môžeme si zapojenie vnútri mikrokontroléra predstaviť takto:



Veľkosť odporu v stave log. 0 je tiež závislá na teplote mikrokontroléra a pre teplotu 25˚C je jeho veľkosť 26Ω.

Príklad:

Navrhnite zapojenie výstupov mikrokontroléra, ak na výstupe RA0 má byť pripojená dióda s úbytkom napätia 1,8V, ktorou má tiecť prúd 10mA a má svietiť ak je na výstupe log. 1. Na výstupe RA2 majú byť do série pripojené dve takéto diódy, ktoré majú svietiť ak je na výstupe log. 0.



Dióda LED1 má svietiť ak je na výstupe log. 1. Preto je potrebné ju pripojiť cez rezistor na zem. Pri log. 1 na výstupe má mikrokontrolér vnútorný odpor 90Ω, na ktorom taktiež vznikne úbytok napätia. Veľkosť odporu R_1 teda vypočítame ako:

$$R_1 = \frac{U - U_D - U_{RP}}{I_D}$$

U – napájacie napätie
U_D – napätie na dióde
U_{RP} – úbytok napätia na vnútornom odpore mikrokontroléra

$$U_{RP} = R_P * I_D$$

R_P – odpor mikrokontroléra
I_D – prúd tečúci diódou

$$R_1 = \frac{5V - 1{,}8V - 90\Omega * 10mA}{10mA} = 230\Omega$$

Keďže rezistory s hodnotou 230Ω sa nevyrábajú zvolíme najbližší vyšší.

Diódy LED2 a LED3 majú svietiť ak je na výstupe log. 0. Preto je potrebné ich pripojiť cez rezistor na +5V. Pri log. 0 na výstupe má mikrokontrolér vnútorný odpor 26Ω, na ktorom taktiež vznikne úbytok napätia. Veľkosť odporu R_2 teda vypočítame ako:

$$R_2 = \frac{U - 2 * U_D - U_{RP}}{I_D}$$

$$R_2 = \frac{5V - 2 * 1{,}8V - 26\Omega * 10mA}{10mA} = 114\Omega$$

Keďže rezistory s hodnotou 114Ω sa nevyrábajú zvolíme najbližší vyšší.

2.4 Oscilátor

Keďže mikrokontrolér patrí medzi synchrónne obvody, potrebuje pre svoju činnosť zdroj hodinových impulzov. Na výber máme jeden zo štyroch typov oscilátorov. Pri programovaní je potrebné zvolený typ nakonfigurovať. Dá sa to buď priamo pri písaní programu (v konfiguračnom slove), alebo je potrebné nastaviť zvolený typ oscilátora v programe, pomocou ktorého programujeme mikrokontrolér.

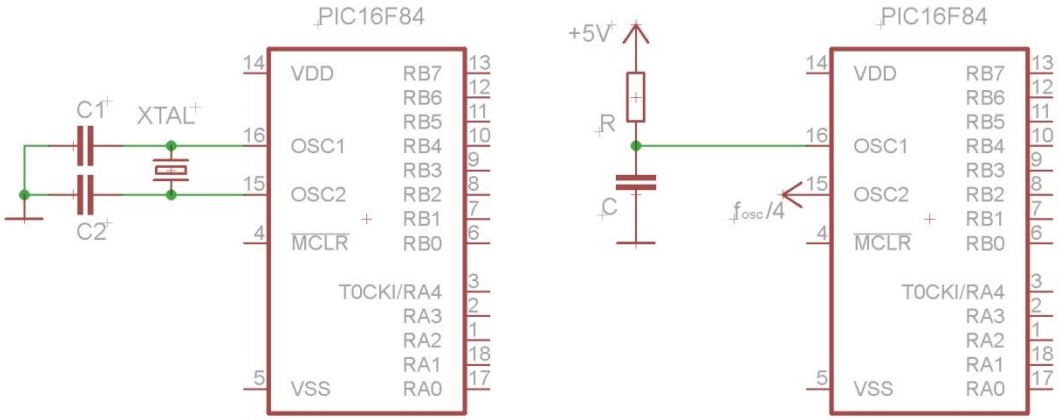
- LP – kryštálový oscilátor s nízkou spotrebou
- XT – kryštálový oscilátor, alebo keramický rezonátor (400 – 8MHz)
- HS – kryštálový oscilátor, alebo keramický rezonátor pre vyššie frekvencie (nad 4MHz)
- RC – oscilátor s RC článkom (tvorený jedným rezistorom a jedným kondenzátorom)

Voľba typu oscilátora závisí od danej aplikácie. Pri úlohách, ktoré nie sú náročné na presnosť je možné použiť lacnejší RC oscilátor. Výsledná frekvencia je závislá od odporu rezistora, kapacity kondenzátora, teploty i od veľkosti napájacieho napätia. Taktiež sa môžu líšiť vnútorné parametre mikrokontroléra, čo tiež ovplyvňuje frekvenciu oscilátora. RC oscilátor sa pripája na pin OSC1 a na pin OSC2 je frekvencia oscilátora vydelená 4.

Pri aplikáciách náročných na presnosť frekvencie hodinových impulzov (časovače, stopky, ...) je lepšie zvoliť kryštálový oscilátor, alebo keramický rezonátor, ktorý má stálu frekvenciu impulzov. Pre pripojenie týchto oscilátorov (rezonátorov) sa použijú piny OSC1 aj OSC2. Ak nastavíme v konfigurácii typ oscilátora LP, XT, alebo HS, je možné použiť aj vonkajší zdroj hodinových impulzov (OSC2 ostane nezapojený).

Doporučené hodnoty súčiastok je možné nájsť v datasheetoch k danému procesoru.

Zapojenia oscilátorov:



2.5Základné registre a ich popis

Registre sú 8 bitové časti pamäte, ktoré majú svoju adresu. Všetky registre, ktoré má mikrokontrolér PIC16F84 sú rozdelené na systémové (adresy 00h -0Bh a 80h – 8Bh) a užívateľské (adresy 0Ch – 4Fh). Systémové registre uchovávajú rôzne nastavenia a informácie o behu mikrokontroléra. Užívateľské registre sú plne k dispozícii pre užívateľa pre odkladanie si výsledkov a informácií. Registre je možné adresovať pomocou čísla, alebo pre jednoduchšie zapamätanie a sprehľadnenie programu pomocou mena. Systémové registre už majú mená definované a mená užívateľských registrov je možné nastaviť na začiatku programu.

Registre sú rozdelené do banky 0 a 1. Banky registrov sa prepínajú pomocou bitu RP0 v registri STATUS.

adresa	**banka 0**	**banka 1**
00h	INDF	
01h	TMR0	OPTION_REG
02h	PCL	
03h	STATUS	
04h	FSR	
05h	PORTA	TRISA
06h	PORTB	TRISB
07h	-	
08h	EEDATA	EECON1
09h	EEADR	EECON2
0Ah	PCLATH	
0Bh	INTCON	
0Ch až 4Fh	užívateľské registre	

Najpoužívanejšie registre:

PCL – v tomto registri je uložených 8 nižších bitov čítača programu. Je možné z neho údaje nielen čítať, ale aj zapisovať. Tým je možné presunúť procesor na akékoľvek miesto programu. V PCL je uložená adresa inštrukcie, ktorá sa má vykonať najbližšie.

STATUS – obsahuje informácie o behu procesora a bit na prepínanie sa medzi bankou 0 a 1.

- **RP0** – výber banky registrov
  - 0 – banka 0
  - 1 – banka 1
- **Z** – výsledok operácie rovný 0
  - 0 – nebol
  - 1 – bol

PORTA, PORTB – tieto registre sú napojené priamo na vstupno/výstupné porty a slúžia na zapísanie údajov na výstupné piny a prečítanie údajov zo vstupných pinov. Hodnoty zapísané do registrov tam ostanú, až kým ich neprepíšeme inou hodnotou (nie je potrebné obnovovať).

PORTA:

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	RA4	RA3	RA2	RA1	RA0

PORTB:

D7	D6	D5	D4	D3	D2	D1	D0
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

TRISA, TRISB – tieto registre slúžia na nastavenie vstupných a výstupných pinov mikrokontroléra. Bit, ktorý má hodnotu 0 je výstupný a bit s hodnotou 1 je vstupný. Nastavenie pinov je možné aj počas vykonávania programu. Rozloženie bitov je rovnaké ako pri registroch PORTA a PORTB.

Pr. Ak chceme, aby piny RB0 až RB3 boli nastavené ako vstupné a piny RB4 až RB7 ako výstupné, je potrebné register TRISB nastaviť podľa nasledujúcej tabuľky.

TRISB:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	1	1	1

2.6Inštrukčný súbor

Mikrokontrolér PIC16F84 má inštrukčný súbor, ktorý obsahuje 35 inštrukcií. Ako je vidieť z nasledujúcej tabuľky, väčšina z nich potrebuje na svoje vykonanie jeden strojový cyklus (pri frekvencii 4MHz to je 1µs). Výnimkou sú inštrukcie, ktoré spôsobia skok v programe, pretože procesor si pripravil pre vykonanie inštrukciu v ďalšom riadku, ktorá sa kvôli skoku nevykoná, a tak si musí pripraviť novú inštrukciu, čo potrvá jeden strojový cyklus.

Veľká väčšina inštrukcií sa vykonáva za pomoci registra W. Ten slúži ako pamäť pre vstupné údaje (pri inštrukciách, ktoré potrebujú dva operandy je jeden vždy uložený v reg. W), ale často i ako pamäť pre výsledok operácií.

Zoznam inštrukcií:

Kód	Popis	Cykly	Ovplyvňuje
Bytové inštrukcie			
ADDWF f,d	algebraický súčet W a f uložiť podľa d	1	C, DC, Z
ANDWF f,d	logický súčin W a f uložiť podľa d	1	Z
CLRF f	vynulovanie f	1	Z
CLRW	Vynulovanie registra W	1	Z
COMF f,d	vytvorí komplement (opačné bity) f a uloží podľa d	1	Z
DECF f,d	dekrementuje (odpočíta 1) f a uloží podľa d	1	Z
DECFSZ f,d	dekrementuje f a uloží podľa d - ak je výsledok 0, nasledujúca inštrukcia sa preskočí	1(2)	-
INCF f,d	inkrementuje (pripočíta 1) f a uloží podľa d	1	Z
INCFSZ f,d	inkrementuje f a uloží podľa d - ak je výsledok 0, nasledujúca inštrukcia sa preskočí	1(2)	-
IORWF f	logický súčet W a f	1	Z
MOVF f,d	presunie obsah registra f podľa d	1	Z
MOVWF f	presunie obsah registra W do registra f	1	-
MOVLW k	presunie číslo k do registra W	1	-
NOP	inštrukcia, ktorá nevykoná nič – čaká jeden cyklus (používa sa na vytvorenie oneskorenia prípadne pre neskoršie doplnenie inštrukcií)	1	-
RLF f,d	rotácia bitov registra f doľava cez C, výsledok sa uloží podľa d	1	C
RRF f,d	rotácia bitov registra f doprava cez C, výsledok sa uloží podľa d	1	C
SUBWF f,d	rozdiel f a W, výsledok sa uloží podľa d Ak je výsledok kladný C = 1 Ak je výsledok záporný C = 0	1	C, DC, Z

SWAPF f,d	vymení horné a dolné štyri bity registra f a výsledok uloží podľa d	1	-
XORWF f,d	exkluzívny súčet registrov W a f, výsledok sa uloží podľa d	1	Z
Bitové inštrukcie			
BCF f,b	nulovanie bitu b v registri f	1	-
BSF f,b	nastavenie bitu b v registri f	1	-
BTFSC f,b	testovanie bitu b v registri f, ak je 0 nasledujúca inštrukcia sa preskočí	1(2)	-
BTFSS f,b	testovanie bitu b v registri f, ak je 1 nasledujúca inštrukcia sa preskočí	1(2)	-
Ostatné inštrukcie			
ADDLW k	algebraický súčet registra W a konštanty k, výsledok sa uloží do W	1	C, DC, Z
ANDLW k	logický súčin registra W a konštanty k, výsledok sa uloží do W	1	Z
CALL k	volanie podprogramu	2	-
CLRWDT	vynulovanie časovača WDT	1	TO, PD
GOTO k	skok na návestie k	2	-
IORLW k	logický súčet registra W a konštanty k, výsledok sa uloží do W	1	Z
RETFIE	návrat z prerušenia	2	-
RETLW k	návrat z podprogramu, k sa uloží do W	2	-
RETURN	návrat z podprogramu	2	-
SLEEP	uspí procesor, návrat resetom, alebo prerušením	1	TO, PD
SUBLW k	rozdiel konštanty k a registra W, výsledok sa uloží do W ak je výsledok kladný C = 1 ak je výsledok záporný C = 0	1	C, DC, Z
XORLW k	exkluzívny súčet konštanty k a registra W, výsledok sa uloží do W	1	Z

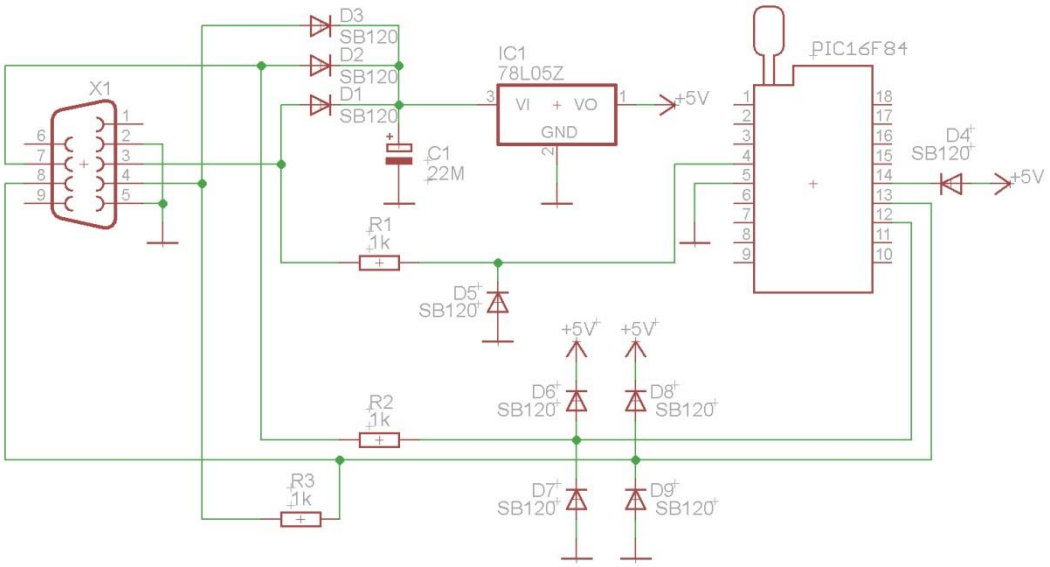
Vysvetlivky k tabuľke:
f – názov (adresa) registra
k – číslo, názov návestia
d – kam uložiť výsledok
0 – do registra W
1 – do registra f (druhý operand)
C, DC, Z, TO, PD – bity registra STATUS
b – pozícia bitu v registri (vpravo je najnižší bit 7, 6, 5 ,4, 3, 2, 1, 0)

3. Programátor

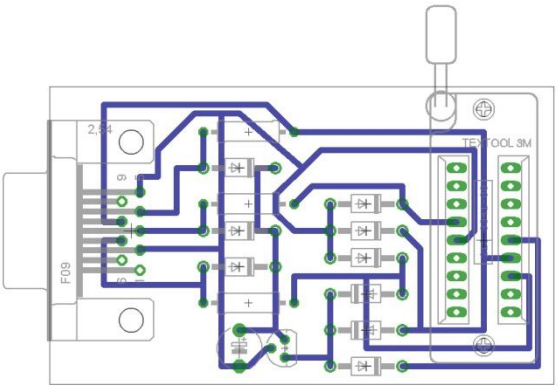
Pre naprogramovanie procesora budeme ešte potrebovať nejaký programátor. My budeme používať programátor *Presto* od firmy *ASIX*. Je to univerzálny programátor pre mikrokontroléry PIC, Atmel, EEPROM pamäte a ďalšie obvody.

Keďže cena programátorov je pomerne vysoká je možné si vytvoriť vlastný programátor určený pre niekoľko druhov súčiastok. Pomocou nižšie uvedenej schémy je možné si vytvoriť jednoduchý sériový programátor, na ktorý potrebujeme len niekoľko súčiastok.Jeho ďalšou výhodou je, že nepotrebuje externý zdroj napájania. Je možné ho použiť nielen pre mikrokontrolér PIC16F84, ale aj pre ostatné 18 pinové mikrokontroléry PIC. Pre zapísanie programu do mikrokontroléra je možné použiť takmer ľubovoľný program, ktorý je možné nájsť na internete. Pre naše potreby plne postačuje program IC-Prog. Ak používate Windows XP, je potrebné si stiahnuť aj ovládač. Keďže program sa do mikrokontroléra nenahráva v assembleri, je potrebné ho previesť do šestnástkovej sústavy. Na to nám slúži program MPASMWIN, ktorý nám zároveň skontroluje správnosť programu. Ak sa v programe vyskytujú chyby, ich výpis je v súbore s príponou err.

Schéma programátora:



Doska plošného spoja:



4. Začíname programovať

Najefektívnejšou metódou, ako sa naučiť programovať v ľubovoľnom programovacom jazyku, je pomocou príkladov. O to viac to platí pre programovací jazyk Assembler, ktorý sa používa pre programovanie mikrokontrolérov. Je to jazyk, ktorý je veľmi blízky jazyku mikrokontrolérov a zaraďujeme ho medzi jazyky nízkej úrovne. Ich hlavnými výhodami sú rýchlosť a malá veľkosť programov. Nevýhodou je o niečo zložitejšie programovanie, keďže tieto jazyky poznajú menej inštrukcií, z ktorých každá vykoná len jednu jednoduchú operáciu. Pre zložitejšie operácie je potrebné použiť vhodnú kombináciu jednoduchších úkonov, čo na prvý pohľad komplikuje programovanie.

Mikrokontroléry je možné programovať i pomocou jazyka C. Výhodou je jednoduchšie programovanie. Nevýhodou je dlhší a pomalší program.

Na prvom programe si ukážeme ako vyzerá samotný program napísaný v assembleri, čo všetko obsahuje a ako je možné ho okomentovať.

```
LIST P=16F84, R=DEC                 ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                 ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF       ;konfiguračné slovo = nastavenie parametrov pre programovanie

CISLO   equ     20h                 ;definovanie pamäti – priradenie mena konkrétnej pamäťovej bunke
CISLOA  equ     21h

#DEFINE TLAC    PORTA,0             ;definovanie názvu pre pin 0 portu A

        org     000                 ;začiatok programu je na adrese 0
        goto    START               ;telo programu

START   bsf     STATUS,RP0
        movlw   B'00000001'
        movwf   TRISA
        movlw   B'00000000'
        movwf   TRISB
        bcf     STATUS,RP0

        movlw   B'00000001'
        movwf   PORTB
ZNOVU   rlf     PORTB,1
        goto    ZNOVU
        end                         ;program je ukončený slovom END
```

Celý program (okrem úvodných troch riadkov, ktoré nemusia v programe byť – viď. ďalšia strana) je písaný v niekoľkých stĺpcoch. Tie pri písaní programu vytvárame pomocou tabelátorov (nie pomocou medzerníka). V prvom stĺpci sú názvy premenných a podprogramov, návestia a kľúčové slová (napr. #DEFINE). Druhý stĺpec obsahuje najmä príkazy. V treťom sú operandy jednotlivých príkazov a za nimi môže nasledovať komentár.

Takéto formátovanie nie je podmienkou, ale program je oveľa lepšie čitateľný, je prehľadnejší a ľahšie sa v ňom hľadajú prípadné chyby.

Ako vidieť z programu, komentáre je možné písať za bodkočiarku. Pri jednoduchších programoch nie sú veľmi potrebné, ale je dobré zvyknúť si ich písať. Je to dobré pre iných, aby lepšie a skôr pochopili

naše programy, ale aj pre nás, keď sa po dlhšom čase vrátime k programu. Komentáre nevplývajú na veľkosť potrebnej pamäte(do mikrokontroléra sa nenahrávajú).

V úvode programu je definovaný typ mikrokontroléra a typ číselnej sústavy. Táto číselná sústava je potom použitá pri všetkých číslach v programe, pri ktorých nie je definovaný typ sústavy. Konštanty v jednotlivých sústavách zapisujeme nasledovne:

- binárna sústava
  B'00111001'
- desiatková sústava
  D'100'
  .100
- šestnástkovásústava
  0x9f
  9fh

V druhom riadku je meno súboru, v ktorom sú informácie pre prekladač assembleru do šestnástkovej sústavy. Sú tam zadefinované napríklad názvy systémových registrov, rozdelenie pamäte ...

V ďalšom riadku je tzv. konfiguračné slovo. Slúži na nastavenie parametrov mikrokontroléra, ako sú napríklad typ oscilátora, oneskorenie programu po reštarte, ...

Tieto prvé riadky nemusia byť napísané v programe. Je možné ich nastaviť aj pri zapisovaní údajov do mikrokontroléra, ale je lepšie ak ich do programu napíšeme. Ak niekedy v budúcnosti budeme ešte tento program potrebovať nemusíme si pamätať ako musí byť mikrokontrolér nastavený aby správne pracoval.

Ďalšie riadky obsahujú definície symbolických mien. Je to výhodné najmä pri väčších programoch, kde využívame viac pamätí, alebo viac vstupov resp. výstupov. Je jednoduchšie si pamätať meno, ktoré vystihuje čo je v pamäti (na vstupe al. výstupe), ako si pamätať nejaké číslo v šestnástkovej sústave (najmä ak ich bude viac).

Za definíciou symbolických mien môžu byť umiestnené podprogramy, za ktorými nasleduje samotné telo programu, ktoré je ukončené slovom end.

Pre písanie programov nám postačuje poznámkový blok, prekladač a programátor. Rozšírené možnosti nám ponúka napr. vývojové prostredie MPLAB, ktoré je možné si stiahnuť zo stránok firmy Microchip.

Program č.1:

Napíšte program, ktorý bude kopírovať stav tlačidiel TL1 až TL4 pripojených na piny 0-3 portu A na diódy pripojené na port B. Tlačidla sú zapojené ako spínacie kontakty a diódy svietia, ak na výstup privedieme log. 1.

```
LIST P=16F84, R=DEC                 ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                 ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF       ;nastavenie procesora

;nastavenie vstupov a výstupov

        bsf     STATUS,RP0          ;nastavenie banky 1
        movlw   B'00001111'         ;uloženie konštanty do registra W
        movwf   TRISA               ;skopírovanie registra W do registra TRISA
        movlw   B'00000000'         ;uloženie konštanty do registra W
        movwf   TRISB               ;skopírovanie registra W do registra TRISB
        bcf     STATUS,RP0          ;prepnutie sa späť do banky 0

;tu začína samotné telo programu

START   movf    PORTA,0             ;skopíruj stav portu A do registra W
        movwf   PORTB               ;skopíruj register W do portu B
        goto    START               ;vráť sa na START - návestie

        end                         ;koniec programu
```

Program č.2:

Napíšte program, ktorý bude kopírovať negovaný stav tlačidiel TL1 až TL4 pripojených na piny 0-3 portu A na diódy pripojené na port B. Tlačidla sú zapojené ako spínacie kontakty a diódy svietia, ak na výstup privedieme log. 1.

```
LIST P=16F84, R=DEC                    ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                    ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF

;nastavenie vstupov a výstupov

        bsf     STATUS,RP0      ;nastavenie banky 1
        movlw   B'00001111'     ;uloženie konštanty do registra W
        movwf   TRISA           ;skopírovanie registra W do registra TRISA
        movlw   B'00000000'     ;uloženie konštanty do registra W
        movwf   TRISB           ;skopírovanie registra W do registra TRISB
        bcf     STATUS,RP0      ;prepnutie sa späť do banky 0

;tu začína samotné telo programu

START   comf    PORTA,0         ;skopíruj negovaný stav portu A do registra W
        movwf   PORTB           ;skopíruj register W do portu B
        goto    START           ;vráť sa na START – návestie

        end                     ;koniec programu
```

Program č.3:

Napíšte program, ktorý po stlačení tlačidla TL1 (pin 1 PORTA) rozsvieti všetky diódy pripojené na port B a po stlačení tlačidla TL2 (pin 2 PORTA) ich všetky zhasne. Tlačidlá sú zapojené ako spínacie kontakty a diódy svietia, ak na výstup privedieme log. 1.

```
LIST P=16F84, R=DEC                ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF

;nastavenie vstupov a výstupov
        bsf     STATUS,RP0         ;nastavenie banky 1
        movlw   B'00000110'        ;uloženie konštanty do registra W
        movwf   TRISA              ;skopírovanie registra W do registra TRISA
        movlw   B'00000000'        ;uloženie konštanty do registra W
        movwf   TRISB              ;skopírovanie registra W do registra TRISB
        bcf     STATUS,RP0         ;prepnutie sa späť do banky 0

;tu začína samotné telo programu
        clrw                       ;vynuluj register W
        clrf    PORTB              ;vynuluj register PORTB – zhasni diódy
START   btfsc   PORTA,1            ;zisti stav tlačidla TL1
        movlw   B'11111111'        ;ak je stlačené zapíš konštantu '11111111' do reg. W
        btfsc   PORTA,2            ;zisti stav tlačidla TL2
        movlw   B'00000000'        ;ak je stlačené zapíš konštantu '00000000' do reg. W
        movwf   PORTB              ;skopíruj reg. W na port B
        goto    START              ;vráť sa na START – návestie
        end                        ;koniec programu
```

Program č.4:

Napíšte program, ktorý pri stlačení tlačidla TL1 (pin 1 PORTA) rozsvieti diódu na pine 4 portu B, TL2 (pin 2 PORTA) rozsvieti diódu na pine 3 portu B, TL3 (pin 3 PORTA) rozsvieti diódu na pine 2 portu B, TL4 (pin 4 PORTA) rozsvieti diódu na pine 1 portu B. Ak tlačidlá nie sú stlačené, diódy nesvietia. Tlačidlá sú zapojené ako spínacie kontakty a diódy svietia, ak na výstup privedieme log. 1.

```
LIST P=16F84, R=DEC                 ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                 ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF

;nastavenie vstupov a výstupov
        bsf     STATUS,RP0      ;nastavenie banky 1
        movlw   B'00011110'     ;uloženie konštanty do registra W
        movwf   TRISA           ;skopírovanie registra W do registra TRISA
        movlw   B'00000000'     ;uloženie konštanty do registra W
        movwf   TRISB           ;skopírovanie registra W do registra TRISB
        bcf     STATUS,RP0      ;prepnutie sa späť do banky 0

;tu začína samotné telo programu
        movlw   B'00000000'     ;nastav všetky výstupy portu B na 0
        movwf   PORTB
START   bcf     PORTB,4         ;zhasni diódu na pine 4 portu B
        btfsc   PORTA,1         ;zisti stav tlačidla
        bsf     PORTB,4         ;ak je stlačené zapni diódu

        bcf     PORTB,3
        btfsc   PORTA,2
        bsf     PORTB,3

        bcf     PORTB,2
        btfsc   PORTA,3
        bsf     PORTB,2

        bcf     PORTB,1
        btfsc   PORTA,4
        bsf     PORTB,1

        goto    START           ;vráť sa na START – návestie
        end                     ;koniec programu
```

Program č.5:

Napíšte program, ktorý realizuje funkciu AND.

```
LIST P=16F84, R=DEC                ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF

;nastavenie vstupov a výstupov
        bsf     STATUS,RP0         ;nastavenie banky 1
        movlw   B'00000110'        ;uloženie konštanty do registra W
        movwf   TRISA              ;skopírovanie registra W do registra TRISA
        movlw   B'00000000'        ;uloženie konštanty do registra W
        movwf   TRISB              ;skopírovanie registra W do registra TRISB
        bcf     STATUS,RP0         ;prepnutie sa späť do banky 0

;tu začína samotné telo programu
        clrf    PORTB
TL1     btfsc   PORTA,1            ;test TL1
        goto    TL2                ;TL1 stlačené - potreba ešte otestovať TL2
        goto    ZHASNI             ;TL1 nie je stlačené = zhasni diódu
TL2     btfsc   PORTA,2            ;test TL2
        goto    SVIET              ;stlačené oba TL = rozsvietiť diódu
        goto    ZHASNI             ;TL2 nie je stlačené = zhasnúť diódu
SVIET   bsf     PORTB,0            ;zasvieť diódu
        goto    TL1
ZHASNI  bcf     PORTB,0            ;zahas diódu
        goto    TL1
        end                        ;koniec programu
```

NERIEŠENÉ PRÍKLADY – LOGICKÉ FUNKCIE:

Program č.6:

Napíšte program, ktorý realizuje funkciu OR.

Program č.7:

Napíšte program, ktorý realizuje funkciu NOR.

Program č.8:

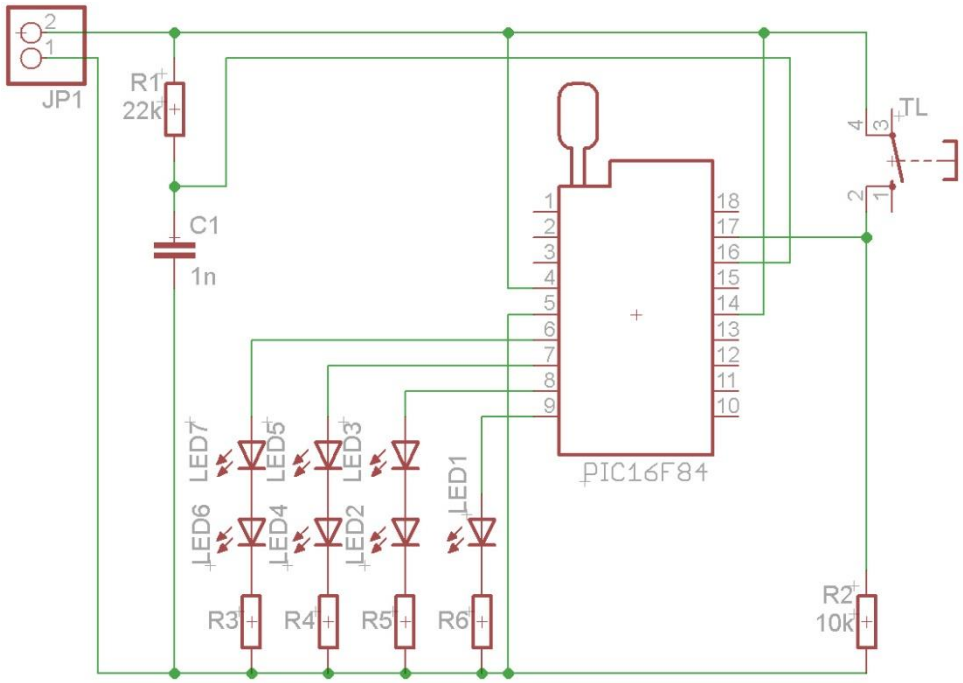
Napíšte program, ktorý realizuje funkciu NAND.

Program č.9:

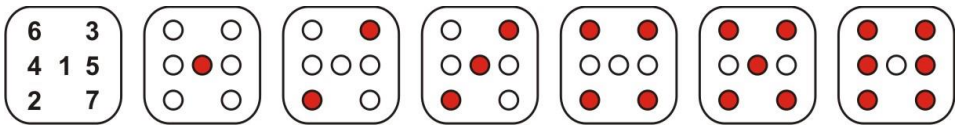
Napíšte program, ktorý realizuje funkciu $F = A + \bar{B}C$.

Program č.10:

Napíšte program, ktorý bude realizovať hraciu kocku (bude generovať 6 kombinácii). Navrhnite schému zapojenia. Vypočítajte a navrhnite veľkosti rezistorov, ak úbytok napätia na LED dióde je 1,9V a môže ňou prechádzať prúd 10mA.



Rozmiestnenie LED



Výpočet:

Rezistory R_3, R_4, R_5 budú mať rovnaké hodnoty. Ich veľkosť vypočítame ako:

$$R_3 = \frac{5V - 2 * U_D - U_{RP}}{I_D} = \frac{5V - 2 * 1{,}9V - 90\Omega * 10mA}{10mA} = 30\Omega - zvolíme\ 33\Omega$$

Rezistor R_6 vypočítame ako:

$$R_6 = \frac{5V - U_D - U_{RP}}{I_D} = \frac{5V - 1{,}9V - 90\Omega * 10mA}{10mA} = 220\Omega - zvolíme\ 220\Omega$$

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

#DEFINE  TL       PORTA,0

         bsf      STATUS,RP0      ;nastavenie banky 1
         movlw    B'00000001'     ;uloženie konštanty do registra W
         movwf    TRISA           ;skopírovanie registra W do registra TRISA
         movlw    B'00000000'     ;uloženie konštanty do registra W
         movwf    TRISB           ;skopírovanie registra W do registra TRISB
         bcf      STATUS,RP0      ;prepnutie sa späť do banky 0

         movlw    B'00001000'     ;na začiatku bude svietiť jednotka
         movwf    PORTB
START    btfss    TL              ;testovanie tlačidla
         goto     START           ;ak nie je stlačené vráť sa na START
         movlw    B'00001111'     ;ak je stlačené rozsvieť všetky diódy na porte B
         movwf    PORTB

MIES     movlw    B'00001000'     ;ulož hodnotu pre zobrazenie '1' do reg. W
         btfss    TL              ;test tlačidla
         goto     ZOBRAZ          ;ak tlačidlo nie je stlačené choď na ZOBRAZ
         movlw    B'00000100'
         btfss    TL
         goto     ZOBRAZ
         movlw    B'00001100'
         btfss    TL
         goto     ZOBRAZ
         movlw    B'00000101'
         btfss    TL
         goto     ZOBRAZ
         movlw    B'00001101'
         btfss    TL
         goto     ZOBRAZ
         movlw    B'00000111'
         btfss    TL
         goto     ZOBRAZ
         goto     MIES            ;tlačidlo je stlačené – vráť sa na MIES (miešanie)

ZOBRAZ   movwf    PORTB           ;prenes hodnotu uloženú v reg. W na port B
         goto     START           vráť sa na START
         end
```

Program č.11:

Napíšte program, ktorý rozbliká diódu pripojenú k pinu 0 na porte B. Dióda svieti ak je na výstupe log. 1.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

LED     equ     0               ;definovanie mena pinu pre diódu

;nastavenie vstupov a výstupov
        bsf     STATUS,RP0      ;nastavenie banky 1
        movlw   B'00000000'     ;uloženie konštanty do registra W
        movwf   TRISB           ;skopírovanie registra W do registra TRISB
        bcf     STATUS,RP0      ;prepnutie sa späť do banky 0

        clrf    PORTB           ;vypne všetky výstupy portu B
START   bsf     PORTB,LED       ;log. 0 na LED – zhasni LED
        bcf     PORTB,LED       ;log. 1 na LED – rozsvieť LED
        goto    START           ;vráť sa na START

        end
```

Po nahratí tohto programu do mikrokontroléra zistíme, že dióda pripojená k portu B nebliká, ale svieti. Aj keď trochu slabšie. Dôvodom je veľká rýchlosť vykonávania inštrukcií mikrokontrolérom. Ako už bolo spomenuté mikrokontrolér potrebuje na vykonanie jednej inštrukcie jeden, alebo dva strojové cykly, čo je pri frekvencii oscilátora 4MHz jedna, alebo dve mikrosekundy.

Ak sa pozrieme na náš program, je tam vytvorená nekonečná slučka. Mikrokontrolér zapne LED diódu, potom ju vypne a pomocou inštrukcie *goto* sa vráti a opäť ju zapne, vypne ...

Inštrukcie *bsf* a *bcf* potrebujú na svoje vykonanie jeden strojový cyklus a inštrukcia *goto* dva. Z toho vyplýva, že dióda sa zapne na 1µs a 3µs je vypnutá. Celý cyklus sa opakuje s periódou 4 µs. Ak si vypočítame frekvenciu blikanie, tak je to 1/4µs = 250 000Hz. Keďže naše oko nedokáže rozoznať takú veľkú frekvenciu blikania, zdá sa nám, akoby dióda svietila, aj keď trochu slabšie.

Ak by sme chceli rýchlosť blikania diódy trochu spomaliť môžeme využiť napríklad inštrukciu NOP, ktorá nič nevykoná, len spotrebuje trochu strojového času (pri f=4MHz trvá 1µs). Len inštrukcia *nop* je však pre väčšie oneskorenia nepoužiteľná. Ak by sme totiž potrebovali vytvoriť napr. oneskorenie 1s, potrebovali by sme 1 000 000 inštrukcií *nop*.

5. Časové oneskorenia

Pri riešení rôznych úloh a príkladov sa často stretávame s požiadavkou na určitý čas pozastaviť, resp. prerušiť vykonávanie behu programu. Keďže v assembleri nepoznáme žiadnu inštrukciu pre zastavenie vykonávania programu na určitý čas (ako napr. delay v pascale al. v C), je potrebné na tento čas zamestnať procesor. Pre vytváranie väčších oneskorení sa používajú čakacie cykly. Počas týchto cyklov procesor dookola vykonáva určité inštrukcie, z ktorých každá trvá jednu alebo dve µs. Ak napríklad vytvoríme cyklus na vykonanie ktorého bude procesor potrebovať 20µs a my potrebujeme zdržať vykonávanie operácii na 1ms, je potrebné, aby sa daný cyklus zopakoval 50-krát.

5.1 Použitie inštrukcie NOP

Inštrukciu *nop* je možné použiť len na vytvorenie veľmi krátkych oneskorení, prípadne na korekciu cyklov. V predošlom príklade sa na výstupe striedali hodnoty 0 a 1 v pomere 1:3. Ak by sme chceli vytvoriť generátor signálu, kde dĺžka signálu s hodnotou 0 a 1 je rovnaká, môžeme použiť posledný program, ktorý je potrebné doplniť o dve inštrukcie *nop*.

```
        clrf    PORTB           ;vypne všetky výstupy portu B
START   bsf     PORTB,LED       ;log. 0 na LED – zhasni LED
        nop
        nop
        bcf     PORTB,LED       ;log. 1 na LED – rozsvieť LED
        goto    START           ;vráť sa na START
```

Inštrukcie *bsf*, *bcf* a *nop* potrebujú na vykonanie jeden strojový cyklus a inštrukcia *goto* dva. Takto vytvoríme program, ktorý na výstupe vytvorí signál, kde hodnota bude (pri frekvencii oscilátora 4MHz) 3µs log. 0 a 3µs log. 1. Takže perióda signálu bude 6µs a frekvencia 167kHz.

5.2 Oneskorovacie cykly

Oneskorovacie cykly sa používajú pre vytvorenie časových intervalov, ktorých dĺžka môže byť niekoľko strojových cyklov (µs) až niekoľko minút. Cykly môžu byť jednoduché a zložené. Zložené cykly sa používajú pre dlhšie časové intervaly, ktoré vzniknú vnáraním jednoduchých cyklov. Tvorba samotných cyklov nie je zložitá, avšak trochu zložitejšie je to s ich počítaním. Pri rôznych svetelných efektoch, či elektronických hračkách je jedno či cyklus trvá 2357 alebo 2400 cyklov. Avšak pri konštrukcii programov, ktoré vyžadujú presné meranie času (hodiny, stopky, ...) je potrebné, aby každý oneskorovací cyklus obsahoval presne určený počet strojových cyklov.

5.2.1 Jednoduchý oneskorovací cyklus:

```
        movlw   1           ;ulož číslo 1 do registra CISLO
        movwf   CISLO

SLUCKA  decfsz  CISLO,1     ;dekrementuj hodnotu registra CISLO - ak je nula preskoč goto
        goto    SLUCKA      ;ak CISLO ≠0 skoč na SLUCKA
```

Inštrukcie movlw a movwf sú inštrukcie presunu, ktoré uložia konštantu (v našom prípade 1) do registra CISLO, ktorý je potrebné na začiatku programu zadefinovať. V ďalšom riadku je návestie SLUCKA, za ktorým je inštrukcia *decfsz*. Tá odpočíta z registra CISLO jednotku a výsledok uloží naspäť do tohto registra (jednotka na konci neznamená, že hodnota registra CISLO sa zníži o 1, ale určuje že číslo sa uloží do registra CISLO - ak dáme na konci inštrukcie 0, výsledok sa uloží do registra W). Ak je výsledok operácie 0, nasledujúca inštrukcia sa preskočí. Preto je inštrukcia *decfsz* inštrukciou podmieneného skoku. Ak výsledok operácie nebol nulový, vykoná sa inštrukcia *goto* a slučka sa zopakuje.

Veľkosť časového oneskorenia v takomto cykle závisí od veľkosti čísla, ktoré uložíme do registra CISLO. Ak v ňom máme uložené číslo 0, celý cyklus bude trvať 4 strojové cykly (SC), čo je pri oscilátore s frekvenciou 4MHz 4 µs. Inštrukcie movlw a movwf trvajú 1SC, inštrukcia dekrementuje hodnotu registra CISLO, takže sa späť uloží hodnota 0 a nasledujúca inštrukcia (*goto*) sa preskočí. Keďže však došlo v programe ku skoku, inštrukcia *decfsz* bude trvať 2SC (celý cyklus: 1SC + 1SC + 2SC = 4SC). Pri vložení konštanty 1 do registra CISLO sa slučka v cykle vykoná len raz a minimálna dĺžka oneskorovacieho cyklu je 4SC, alebo 4µs (ak nebude uvedené ináč, uvažujeme s oscilátorom s frekvenciou 4MHz).

Ak nastavíme do registra CISLO hodnotu 2, vytvoríme časové oneskorenie 7SC (7µs).

```
        movlw   2           ;ulož číslo 2 do registra CISLO
        movwf   CISLO

SLUCKA  decfsz  CISLO,1     ;dekrementuj hodnotu registra CISLO - ak je nula preskoč goto
        goto    SLUCKA      ;ak CISLO ≠0 skoč na SLUCKA
```

Inštrukcie sa budú vykonávať nasledovne:

```
        movlw   2           ;1SC
        movwf   CISLO       ;1SC
SLUCKA  decfsz  CISLO,1     ;1SC – CISLO bude mat hodnotu 1 – nevykoná sa skok
        goto    SLUCKA      ;2SC – skok na SLUCKA
SLUCKA  decfsz  CISLO,1     ;2SC – CISLO bude mat hodnotu 0 – vykoná sa skok
```

Ak by sme do registra CISLO vložili hodnotu 3, slučka sa v cykle zopakuje dvakrát pomocou inštrukcie *goto* a raz bude ukončená inštrukciou *decfsz*, ktorá cyklus ukončí.

Keďže v mikrokontroléri majú registre veľkosť 8 bitov, môžeme do nich uložiť najväčšie číslo 255. V takomto prípade sa bude cyklus opakovať 254-krát pomocou inštrukcie *goto* a poslednýkrát sa ukončí inštrukciou *decfsz*. Najväčšie časové oneskorenie, ktoré je možné spraviť pomocou jednoduchého cyklu dosiahneme, ak do registra CISLO uložíme 0. Cyklus s *goto* sa takto zopakuje 255-krát a potom sa ukončí podmieneným skokom *decfsz*.

Celkový čas (počet strojových cyklov) môžeme vypočítať podľa vzťahu:

$$T = 3 * (n - 1) + 2 + 2 = 3n - 3 + 4 = 3n + 1$$

- kde n je konštanta uložená do registra CISLO

Tento vzťah sa nedá použiť pre číslo 0. Ak chceme vedieť aké časové oneskorenie by nám vzniklo, musíme do vzorca dosadiť 256.

n	**SC**	**T pri f=4MHz**	**T pri f=10kHz**
1	4	4µs	1,6ms
2	7	7µs	2,8ms
...	...	...	
150	451	451µs	60ms
...	...	...	
255	766	766µs	102ms
0 (256)	769	769µs	102,4ms

Z tabuľky môžeme vidieť, že pri oscilátore s frekvenciou 4MHz, môžeme pomocou jednoduchého oneskorovacieho cyklu dosiahnuť oneskorenie 4-769µs. Toto oneskorenie zodpovedá frekvenciám 250kHz-1300Hz. Keďže ľudské oko nie je schopné rozoznať takúto frekvenciu, je potrebné pri aplikáciách so svetelnými efektmi, ale aj pri iných častokrát použiť zložitejšie oneskorovacie cykly (zložené), pomocou ktorých je možné vytvárať väčšie oneskorenia. Zložené oneskorovacie cykly vznikajú vnáraním jednoduchých.

5.2.2 Dvojitý oneskorovací cyklus:

```
        movlw   1           ;ulož číslo 1 do registra CISLO2
        movwf   CISLO2

SLUCKA2 movlw   1           ;ulož číslo 1 do registra CISLO1
        movwf   CISLO1

SLUCKA1 decfsz  CISLO1,1    ;dekrementuj hodnotu registra CISLO1-ak je nula preskoč goto
        goto    SLUCKA1     ;ak CISLO1 ≠0 skoč na SLUCKA1
        decfsz  CISLO2,1    ;dekrementuj hodnotu registra CISLO2-ak je nula preskoč goto
        goto    SLUCKA2     ;ak CISLO2 ≠0 skoč na SLUCKA2
```

Dvojitý oneskorovací cyklus je cyklus s jedným vnorením. Uvedený cyklus má hodnoty registrov CISLO1 a CISLO2 nastavené na 1, čiže nám vytvorí najmenšie možné oneskorenie. Pri dvojitom cykle to je 8µs. Keďže takéto oneskorenie je možné vytvoriť aj s jednoduchým cyklom, je pre nás výhodnejšie oneskorenia kratšie ako 770µs vytvárať pomocou jednoduchých cyklov. Maximálne oneskorenie, ktoré môžeme pomocou dvojitého cyklu vytvoriť dosiahneme nastavením registrov na 0. Jeho hodnota bude necelých 200ms.

Celkový čas (počet strojových cyklov) môžeme vypočítať podľa vzťahu:

$$T = 6 + 5(C_2 - 1) + C_2\big(2 + 3(C_1 - 1)\big) = 4C_2 + 3C_1C_2 + 1$$

- kde C1 je konštanta uložená do registra CISLO1
- kde C2 je konštanta uložená do registra CISLO2

Tento vzťah sa opäť nedá použiť pre číslo 0. Ak chceme vedieť aké časové oneskorenie by nám vzniklo, musíme do vzorca dosadiť 256.

C_1	**C_2**	**SC**	**T pri f=4MHz**
1	1	8	8µs
2	2	21	21µs
...		...	...
150	255	115 771	115 771µs
...		...	...
255	255	196 096	196 096µs
0 (256)	0(256)	196 633	196 633µs

Z tabuľky vidieť, že pri frekvencii oscilátora môžeme s dvojitým cyklom vytvoriť oneskorenia necelých 200ms. Pre väčšie oneskorenia sa ešte využívajú trojité cykly, ktoré obsahujú dva vnorenia. S ich pomocou vieme vytvoriť oneskorenia, s dĺžkou 50s (pre oscilátor s f=4MHz), čo pre väčšinu aplikácii postačuje. Zložitejšie cykly sa využívajú len málokedy.

5.2.3 Trojitý oneskorovací cyklus:

```
                movlw   1               ;ulož číslo 1 do registra CISLO3
                movwf   CISLO3

SLUCKA3         movlw   1       ;ulož číslo 1 do registra CISLO2
                movwf   CISLO2

SLUCKA2         movlw   1
                movwf   CISLO1

SLUCKA1         decfsz  CISLO1,1        ;dekrementuj hodnotu registra CISLO1-ak je nula preskoč goto
                goto    SLUCKA1         ;ak CISLO1 ≠0 skoč na SLUCKA1
                decfsz  CISLO2,1        ;dekrementuj hodnotu registra CISLO2-ak je nula preskoč goto
                goto    SLUCKA2         ;ak CISLO2 ≠0 skoč na SLUCKA2
                decfsz  CISLO3,1        ;dekrementuj hodnotu registra CISLO3-ak je nula preskoč goto
                goto    SLUCKA3         ;ak CISLO3 ≠0 skoč na SLUCKA3
```

Cyklus je opäť nastavený na najkratšiu dobu, ktorú je možné s trojitým cyklom vytvoriť. Keďže jeho dĺžka je 12µs, je výhodnejšie pre kratšie oneskorenia využívať jednoduchšie cykly. Tak ako u predošlých cyklov, najdlhšiu dobu oneskorenia dosiahneme, ak registre CISLO nastavíme na 0.

Celkový čas (počet strojových cyklov) môžeme vypočítať podľa vzťahu:

$$T = 3(C_3 - 1) + 4 + C_3\left(6 + 5(C_2 - 1) + C_2\left(2 + 3(C_1 - 1)\right)\right) = 4C_3 + 4C_2C_3 + 3C_1C_2C_3 + 1$$

- kde C1 je konštanta uložená do registra CISLO1
- kde C2 je konštanta uložená do registra CISLO2
- kde C3 je konštanta uložená do registra CISLO3

C_1	**C_2**	**C_3**	**SC**	**T pri f=4MHz**
1	1	1	12	12µs
2	2	2	49	49µs
...			...	...
150	200	100	9 080 401	9 080 401µs
...			...	...
255	255	255	50 005 246	50 005 246µs
0 (256)	0(256)	0(256)	50 594 817	50 594 817µs

Program č.12:

Napíšte program, ktorý rozbliká diódu pripojenú k pinu 0 na porte B. Použite jednoduchý oneskorovací cyklus, ktorým nastavte najnižšiu frekvenciu blikania. Dióda svieti ak je na výstupe log. 1. Zmerajte frekvenciu blikania diódy.

```
LIST P=16F84, R=DEC                 ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                 ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF

CISLO    equ     20h                ;definovanie mena pre register na adrese 20h
#DEFINE  LED     PORTB,0            ;definovanie symbolického mena pre výstupnú LED

;nastavenie vstupov a výstupov
         bsf     STATUS,RP0         ;nastavenie banky 1
         movlw   B'00000000'        ;uloženie konštanty do registra W
         movwf   TRISB              ;skopírovanie registra W do registra TRISB
         bcf     STATUS,RP0         ;prepnutie sa späť do banky 0

;tu začína samotné telo programu
         movlw   B'00000000'        ;nastav všetky výstupy portu B na 0
         movwf   PORTB
START    bsf     LED                ;rozsvieť LED
         movlw   0                  ;ulož číslo 1 do registra CISLO
         movwf   CISLO
SLUCKA1  decfsz  CISLO,1            ;dekrementuj hodnotu registra CISLO - ak je nula preskoč goto
         goto    SLUCKA1            ;ak CISLO ≠0 skoč na SLUCKA1

         bcf     LED                ;zhasni LED
         movlw   0                  ;ulož číslo 1 do registra CISLO
         movwf   CISLO
SLUCKA2  decfsz  CISLO,1            ;dekrementuj hodnotu registra CISLO - ak je nula preskoč goto
         goto    SLUCKA2            ;ak CISLO ≠0 skoč na SLUCKA2

         goto    START              ;vráť sa na START
         end                        ;koniec programu
```

Program č.13:

Napíšte program, ktorý rozbliká diódu pripojenú k pinu 0 na porte B. Použite dvojitý oneskorovací cyklus, ktorým nastavte najnižšiu frekvenciu blikania. Dióda svieti ak je na výstupe log. 1. Zmerajte frekvenciu blikania diódy. Skúste meniť rýchlosť blikania diódy.

```
LIST P=16F84, R=DEC                ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF

CISLO1   equ     20h               ;definovanie mena pre register na adrese 20h
CISLO2   equ     21h               ;definovanie mena pre register na adrese 21h

#DEFINE  LED     PORTB,0           ;definovanie symbolického mena pre výstupnú LED

;nastavenie vstupov a výstupov
         bsf     STATUS,RP0        ;nastavenie banky 1
         movlw   B'00000000'       ;uloženie konštanty do registra W
         movwf   TRISB             ;skopírovanie registra W do registra TRISB
         bcf     STATUS,RP0        ;prepnutie sa späť do banky 0

;tu začína samotné telo programu
         movlw   B'00000000'       ;nastav všetky výstupy portu B na 0
         movwf   PORTB
START    bsf     LED               ;rozsvieť LED
         movlw   0                 ;ulož číslo 0 do registra CISLO2
         movwf   CISLO2

SLUCKA2  movlw   0                 ;ulož číslo 0 do registra CISLO1
         movwf   CISLO1

SLUCKA1  decfsz  CISLO1,1          ;dekrementuj hodnotu registra CISLO1-ak je nula preskoč goto
         goto    SLUCKA1           ;ak CISLO1 ≠0 skoč na SLUCKA1
         decfsz  CISLO2,1          ;dekrementuj hodnotu registra CISLO2-ak je nula preskoč goto
         goto    SLUCKA2           ;ak CISLO2 ≠0 skoč na SLUCKA2

         bcf     LED               ;zhasni LED
         movlw   0                 ;ulož číslo 0 do registra CISLO2
         movwf   CISLO2

SLUCKA4  movlw   0                 ;ulož číslo 0 do registra CISLO1
         movwf   CISLO1

SLUCKA3  decfsz  CISLO1,1          ;dekrementuj hodnotu registra CISLO1-ak je nula preskoč goto
         goto    SLUCKA3           ;ak CISLO1 ≠0 skoč na SLUCKA3
         decfsz  CISLO2,1          ;dekrementuj hodnotu registra CISLO2-ak je nula preskoč goto
         goto    SLUCKA4           ;ak CISLO2 ≠0 skoč na SLUCKA4

         goto    START             ;vráť sa na START
         end                       ;koniec programu
```

Program č.14:

Napíšte program, ktorý rozbliká diódu pripojenú k pinu 0 na porte B. Použite trojitý oneskorovací cyklus, ktorým nastavte frekvenciu blikania. Dióda svieti ak je na výstupe log. 1. Zmerajte frekvenciu blikania diódy. Skúste meniť rýchlosť blikania diódy.

```
LIST P=16F84, R=DEC                ;definovanie typu mikrokontroléra a číselnej sústavy
INCLUDE<P16F84.INC>                ;meno súboru, kde sú informácie o mikrokontroléri
__CONFIG _PWRTE_ON & _WDT_OFF

CISLO1   equ     20h               ;definovanie mena pre register na adrese 20h
CISLO2   equ     21h               ;definovanie mena pre register na adrese 21h
CISLO3   equ     22h               ;definovanie mena pre register na adrese 22h

#DEFINE  LED     PORTB,0           ;definovanie symbolického mena pre výstupnú LED
;nastavenie vstupov a výstupov
         bsf     STATUS,RP0        ;nastavenie banky 1
         movlw   B'00000000'       ;uloženie konštanty do registra W
         movwf   TRISB             ;skopírovanie registra W do registra TRISB
         bcf     STATUS,RP0        ;prepnutie sa späť do banky 0
;tu začína samotné telo programu
         movlw   B'00000000'       ;nastav všetky výstupy portu B na 0
         movwf   PORTB
START    bsf     LED               ;rozsvieť LED
         movlw   100               ;ulož číslo 100 do registra CISLO3
         movwf   CISLO3
SLUCKA3  movlw   100               ;ulož číslo 100 do registra CISLO2
         movwf   CISLO2
SLUCKA2  movlw   100               ;ulož číslo 100 do registra CISLO1
         movwf   CISLO1
SLUCKA1  decfsz  CISLO1,1          ;dekrementuj hodnotu registra CISLO1-ak je nula preskoč goto
         goto    SLUCKA1           ;ak CISLO1 ≠0 skoč na SLUCKA1
         decfsz  CISLO2,1          ;dekrementuj hodnotu registra CISLO2-ak je nula preskoč goto
         goto    SLUCKA2           ;ak CISLO2 ≠0 skoč na SLUCKA2
         decfsz  CISLO3,1          ;dekrementuj hodnotu registra CISLO3-ak je nula preskoč goto
         goto    SLUCKA3           ;ak CISLO3 ≠0 skoč na SLUCKA3

         bcf     LED               ;zhasni LED
         movlw   1                 ;ulož číslo 1 do registra CISLO3
         movwf   CISLO3
SLUCKA6  movlw   1                 ;ulož číslo 1 do registra CISLO2
         movwf   CISLO2
SLUCKA5  movlw   1                 ;ulož číslo 100 do registra CISLO1
         movwf   CISLO1
SLUCKA4  decfsz  CISLO1,1          ;dekrementuj hodnotu registra CISLO1-ak je nula preskoč goto
         goto    SLUCKA4           ;ak CISLO1 ≠0 skoč na SLUCKA4
         decfsz  CISLO2,1          ;dekrementuj hodnotu registra CISLO2-ak je nula preskoč goto
         goto    SLUCKA5           ;ak CISLO2 ≠0 skoč na SLUCKA5
         decfsz  CISLO3,1          ;dekrementuj hodnotu registra CISLO3-ak je nula preskoč goto
         goto    SLUCKA6           ;ak CISLO3 ≠0 skoč na SLUCKA6
         goto    START             ;vráť sa na START
         end                       ;koniec programu
```

6. Podprogramy

Podprogram je časť kódu vnútri väčšieho programu, ktorý vykonáva špecifickú úlohu, ktorá sa v programe opakuje, resp. sa využíva viackrát. Je to vlastne časť programu, ktorá je v hlavnom programe nahradená jednou inštrukciou – volaním podprogramu. Medzi základné výhody podprogramov patria:

- redukuje sa duplicita kódu (možnosť opakovania) – šetrí sa miesto v pamäti
- rozloženie zložitých problémov na jednoduchšie
- lepšia čitateľnosť programu
- ľahšie ladenie programov – jednoduchšia oprava chýb

Podprogramy sa väčšinou píšu na začiatku programu za definície konštánt a mien registrov, alebo na konci programu. Pre ich volanie sa využíva inštrukcia *call* a pre návrat z podprogramu sa používajú inštrukcie *return* a *retlw*. Inštrukcia *return* vykoná len návrat z podprogramu. Pomocou inštrukcie *retlw* je možné vykonať návrat z podprogramu a zároveň do registra W uložiť konštantu.

Mikrokontrolér vždy začína vykonávať program od adresy 0h. Ak vložíme podprogram na začiatok celého programu, je potrebné mikrokontroléru určiť na ktorej adrese v pamäti sa začína hlavný program.

```
org     000         ;začiatok programu je na adrese 000
goto    START       ;hneď choď na návestie START
```

Za týmito inštrukciami nasledujú podprogramy a za nimi hlavný program, ktorý začína návestím START.

Podprogram sa začína menom (návestím), potom nasleduje samotné telo podprogramu a ukončený je príkazom *return*, alebo *retlw*.

Podprogramy je možné aj vnorovať. To znamená, že jeden podprogram môže vyvolať ďalší a ten zasa ďalší. Vnorenie podprogramov je možné do maximálne ôsmej úrovne. To preto, že vždy pri volaní podprogramu je potrebné uložiť adresu, kde sa má mikrokontrolér pri skončení podprogramu vrátiť a na tento účel sa využíva zásobník, ktorého veľkosť je 8 pamäťových buniek. Počet vnorení si musí programátor ustrážiť sám, keďže pretečenie zásobníka nie je nikde signalizované.

Podprogramy nemusia byť umiestnené na začiatku programu a nemusia byť ani všetky umiestnené na jednom mieste. Umiestnenie všetkých podprogramov spolu sprehľadňuje zdrojový kód programu.

Program č.15:

Napíšte program, ktorý rozbliká diódu pripojenú k pinu 0 na porte B. Použite dvojitý oneskorovací cyklus, ktorým nastavte najnižšiu frekvenciu blikania. Dióda svieti ak je na výstupe log. 1. Použite podprogram.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

;*************************************
CISLO1    equ       20h
CISLO2    equ       21h

#DEFINE   LED       PORTB,0

;*************************************
          org       000
          goto      START
;*************************************
CAKAJ     movlw     0                 ;podprogram CAKAJ
          movwf     CISLO2
SLUCKA2   movlw     0
          movwf     CISLO1
SLUCKA1   decfsz    CISLO1,1
          goto      SLUCKA1
          decfsz    CISLO2,1
          goto      SLUCKA2
          return                      ;návrat z podprogramu
;*************************************
START     bsf       STATUS,RP0        ; hlavný program
          movlw     B'00000000'
          movwf     TRISB
          bcf       STATUS,RP0
          movlw     B'00000000'       ;nastav všetky výstupy portu B na 0
          movwf     PORTB
ZNOVU     bsf       LED               ;rozsvieť LED
          call      CAKAJ             ;zavolaj podprogram CAKAJ
          bcf       LED               ;zhasni LED
          call      CAKAJ             ;zavolaj podprogram CAKAJ
          goto      ZNOVU             ;vráť sa na ZNOVU

          end                         ;koniec programu
```

Program č.16:

Napíšte program, ktorý vytvorí svetelného hada z diód pripojených na PORTB. Diódy svietia ak je na výstupe log. 1.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

;*************************************
CISLO1   equ      20h
CISLO2   equ      21h

;*************************************
         org      000
         goto     START

;*************************************
CAKAJ    movlw    0              ;podprogram pre vytvorenie oneskorenia
         movwf    CISLO2
SLUCKA2  movlw    0
         movwf    CISLO1
SLUCKA1  decfsz   CISLO1,1
         goto     SLUCKA1
         decfsz   CISLO2,1
         goto     SLUCKA2
         return

;*************************************
START    bsf      STATUS,RP0     ;nastavenie vstupov a výstupov
         movlw    B'00000000'
         movwf    TRISB
         bcf      STATUS,RP0

ZNOVU    movlw    B'00000001'    ;ulož na port B danú kombináciu
         movwf    PORTB
         call     CAKAJ          ;zavolaj podprogram
         movlw    B'00000010'
         movwf    PORTB
         call     CAKAJ
         movlw    B'00000100'
         movwf    PORTB
         call     CAKAJ
         movlw    B'00001000'
         movwf    PORTB
         call     CAKAJ
         movlw    B'00010000'
         movwf    PORTB
         call     CAKAJ
         movlw    B'00100000'
         movwf    PORTB
         call     CAKAJ
         movlw    B'01000000'
```

```
movwf   PORTB
call    CAKAJ
movlw   B'10000000'
movwf   PORTB
call    CAKAJ
goto    ZNOVU          ;vrať sa na ZNOVU
end
```

Program č.17:

Napíšte program, ktorý vytvorí svetelného hada z diód pripojených na PORTB. Diódy svietia ak je na výstupe log. 1. Diódy majú blikať takto: 1+8, 2+7, 3+6, 4+5, 3+6, 2+7.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF
;*************************************
CISLO1    equ       20h
CISLO2    equ       21h
;*************************************
          org       000
          goto      START
;*************************************
CAKAJ     movlw     0              ;podprogram pre vytvorenie oneskorenia
          movwf     CISLO2
SLUCKA2   movlw     0
          movwf     CISLO1
SLUCKA1   decfsz    CISLO1,1
          goto      SLUCKA1
          decfsz    CISLO2,1
          goto      SLUCKA2
          return
;*************************************
START     bsf       STATUS,RP0     ;nastavenie vstupov a výstupov
          movlw     B'00000000'
          movwf     TRISB
          bcf       STATUS,RP0

ZNOVU     movlw     B'10000001'    ;ulož na port B danú kombináciu
          movwf     PORTB
          call      CAKAJ          ;zavolaj podprogram
          movlw     B'01000010'
          movwf     PORTB
          call      CAKAJ
          movlw     B'00100100'
          movwf     PORTB
          call      CAKAJ
          movlw     B'00011000'
          movwf     PORTB
          call      CAKAJ
          movlw     B'00100100'
          movwf     PORTB
          call      CAKAJ
          movlw     B'01000010'
          movwf     PORTB
          call      CAKAJ
          goto      ZNOVU          ;vráť sa na ZNOVU
          end
```

Program č.18:

Napíšte program, ktorý vytvorí svetelného hada z diód pripojených na PORTB. Diódy svietia ak je na výstupe log. 1. Diódy sa budú blikať postupne po jednej. Použite inštrukciu *rlf* a *rrf*.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF
;*************************************
CISLO1    equ       20h
CISLO2    equ       21h
;*************************************
          org       000
          goto      START
;*************************************
CAKAJ     movlw     0                 ;podprogram pre vytvorenie oneskorenia
          movwf     CISLO2
SLUCKA2   movlw     0
          movwf     CISLO1
SLUCKA1   decfsz    CISLO1,1
          goto      SLUCKA1
          decfsz    CISLO2,1
          goto      SLUCKA2
          return
;*************************************
START     bsf       STATUS,RP0        ;nastavenie vstupov a výstupov
          movlw     B'00000000'
          movwf     TRISB
          bcf       STATUS,RP0

          movlw     B'00000001'       ;ulož hodnotu do registra PORTB
          movwf     PORTB
ZNOVU     call      CAKAJ             ;zavolaj podprogram
          rlf       PORTB,1           ;vykonaj rotáciu bitov doľava v registri PORTB
          goto      ZNOVU             ;vráť sa na ZNOVU
          end
```

Program č.19:

Napíšte program, ktorý vytvorí svetelného hada z diód pripojených na PORTB. Diódy budú blikať len ak je stlačené tlačidlo na RA1. Ak nie je tlačidlo stlačené, nemá svietiť žiadna dióda.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF
;*************************************
CISLO1    equ     20h
CISLO2    equ     21h
#DEFINE   TL      PORTA,1
;*************************************
          org     0
          goto    START
CAKAJ     movlw   0               ;podprogram pre vytvorenie oneskorenia
          movwf   CISLO2
SLUCKA2   movlw   0
          movwf   CISLO1
SLUCKA1   decfsz  CISLO1,1
          goto    SLUCKA1
          decfsz  CISLO2,1
          goto    SLUCKA2
          return
;*************************************
START     bsf     STATUS,RP0      ;nastavenie vstupov a výstupov
          movlw   B'00000010'
          movwf   TRISA
          movlw   B'00000000'
          movwf   TRISB
          bcf     STATUS,RP0
TEST      movlw   B'00000000'     ;ulož na PORTB hodnotu 0
          movwf   PORTB
          btfss   TL              ;test stlačenia TL
          goto    TEST            ;ak nie je stlačené, vráť sa na TEST
          movlw   B'00000001'     ;ak bolo stlačené, ulož do registra PORTB hodnotu 1
          movwf   PORTB
ZNOVU     call    CAKAJ           ;zavolaj podprogram
          rlf     PORTB,1         ;vykonaj rotáciu bitov doľava v registri PORTB
          btfss   TL              ;testuj stlačenie TL
          goto    TEST            ;ak nie je vráť sa na TEST
          goto    ZNOVU           ;ak je stlačené, vráť sa na ZNOVU
          end
```

Program č.20:

Napíšte program, ktorý vytvorí svetelného hada z diód pripojených na PORTB. Diódy budú blikať v jednom smere a pri stlačenom tlačidle v druhom smere.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF
;*************************************
CISLO1    equ       20h
CISLO2    equ       21h
#DEFINE   TL        PORTA,1
;*************************************
          org       000
          goto      START
;*************************************
CAKAJ     movlw     0               ;podprogram pre vytvorenie oneskorenia
          movwf     CISLO2
SLUCKA2   movlw     0
          movwf     CISLO1
SLUCKA1   decfsz    CISLO1,1
          goto      SLUCKA1
          decfsz    CISLO2,1
          goto      SLUCKA2
          return
;*************************************
START     bsf       STATUS,RP0      ;nastavenie vstupov a výstupov
          movlw     B'00000010'
          movwf     TRISA
          movlw     B'00000000'
          movwf     TRISB
          bcf       STATUS,RP0

          movlw     B'00000001'     ;do registra PORTB ulož hodnotu 1
          movwf     PORTB

SMER1     rlf       PORTB,1         ;vykonaj rotáciu bitov doľava v registri PORTB
          call      CAKAJ           ;zavolaj podprogram
          btfss     TL              ;test tlačidla TL
          goto      SMER2           ;ak nie je stlačené choď na SMER2
          goto      SMER1           ;ak je stlačené choď na SMER1

SMER2     rrf       PORTB,1         ;vykonaj rotáciu bitov doprava
          call      CAKAJ           ;zavolaj podprogram
          btfss     TL              ;test tlačidla TL
          goto      SMER2           ;ak je stlačené choď na SMER2
          goto      SMER1           ;ak nie je stlačené choď na SMER1

          end
```

Program č.21:

Napíšte program, ktorý vytvorí svetelného hada z diód pripojených na PORTB. Pomocou tlačidla sa má meniť rýchlosť. Pri stlačenom tlačidle má byť rýchlosť blikania väčšia.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

;*************************************
CISLO1   equ     20h
CISLO2   equ     21h
CAS      equ     22h

;*************************************
         org     000
         goto    START

;*************************************
CAKAJ    movf    CAS,0           ;CAS->W
         movwf   CISLO2
SLUCKA2  movlw   0
         movwf   CISLO1
SLUCKA1  decfsz  CISLO1,1
         goto    SLUCKA1
         decfsz  CISLO2,1
         goto    SLUCKA2
         return
R1       movlw   250             ;podprogram pre zmenu rýchlosti - pomalšia
         movwf   CAS
         return
R2       movlw   50              ;podprogram pre zmenu rýchlosti - rýchlejšia
         movwf   CAS
         return
;*************************************
START    bsf     STATUS,RP0      ;nastavenie vstupov a výstupov
         movlw   B'00000001'
         movwf   TRISA
         movlw   B'00000000'
         movwf   TRISB
         bcf     STATUS,RP0
         movlw   B'00000001'     ;nastavenie hodnoty 1 na port B
         movwf   PORTB
ZNOVU    rlf     PORTB,1         ;rotácia bitov doľava
         btfss   PORTA,0         ;test stlačenia tlačidla
         call    R1              ;ak je stlačené nastav rýchlosť R1
         btfsc   PORTA,0         ;test stlačenia tlačidla
         call    R2              ;ak nie je stlačené nastav rýchlosť R2
         call    CAKAJ
         goto    ZNOVU
         end
```

NERIEŠENÉ PRÍKLADY – SVETELNÉ EFEKTY:

Program č.22:

Napíšte program, ktorý vytvorí svetelného hada z diód pripojených na PORTB. Pomocou tlačidla TL1 sa má meniť rýchlosť a pomocou tlačidla TL2 smer blikania.

Program č.23:

Napíšte program, ktorý vytvorí svetelného hada z diód pripojených na PORTB. Pomocou tlačidla TL1 sa dajúzapnúť/vypnúť diódy (tlačidlo nie je potrebné držať stlačené) a pomocou TL2 sa mení rýchlosť blikania.

Program č.24:

Napíšte program, ktorý vytvorí po stlačení tlačidla 20x zabliká diódou.

Program č.25:

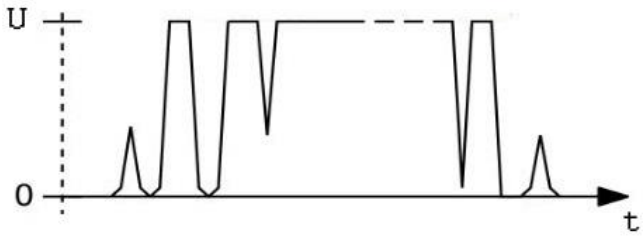
Napíšte program, ktorý bude ovládať diódu jedným tlačidlom (START/STOP).

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

;*************************************
#DEFINE  TL      PORTA,1
#DEFINE  LED     PORTB,0

;*************************************
         bsf     STATUS,RP0
         movlw   B'00000010'
         movwf   TRISA
         movlw   B'00000000'
         movwf   TRISB
         bcf     STATUS,RP0

T1       btfss   TL
         goto    T1              ;možno použiť inštrukciu goto $-1 ( návrat o 1 riadok)
         bsf     LED
T2       btfsc   TL
         goto    T2

T3       btfss   TL
         goto    T3
         bcf     LED
T4       btfsc   TL
         goto    T4

         goto    T1

         end
```

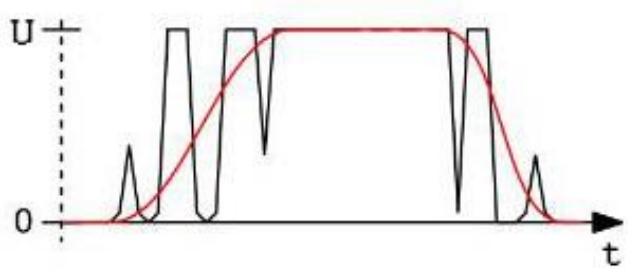
Po nahratí programu do mikrokontroléra zistíme, že nie vždy po stlačení tlačidla sa zmení stav LED diódy. Dôvodom sú zákmity kontaktov tlačidiel.

7. Ošetrenie zákmitov

Zákmity sú krátke impulzy na kontaktoch, ktoré vznikajú z dôvodu pružnosti kontaktova nečistôt. Pri stlačení tlačidla nevznikne len jeden impulz, ale aj niekoľko ďalších, ktoré ovplyvnia správnu funkciu programu. Tieto zákmity je možné ošetriť hardvérovo alebo softvérovo.



Hardvérovo je to možné pridaním kondenzátora k tlačidlu,



alebo použitím RS klopného obvodu, či bezzákmitového tlačidla.

Pre softvérové ošetrenie zákmitov môžeme použiť oneskorovacie slučky. Po zistení stlačenia tlačidla zavoláme podprogram s krátkym časovým oneskorením a po jeho skončení opäť otestujeme stav tlačidla. Ak je opäť signalizované stlačenie, pokračujeme ďalej v programe. Ak mikrokontrolér nezistí stlačené tlačidlo vrátime sa späť.

Program č.26:

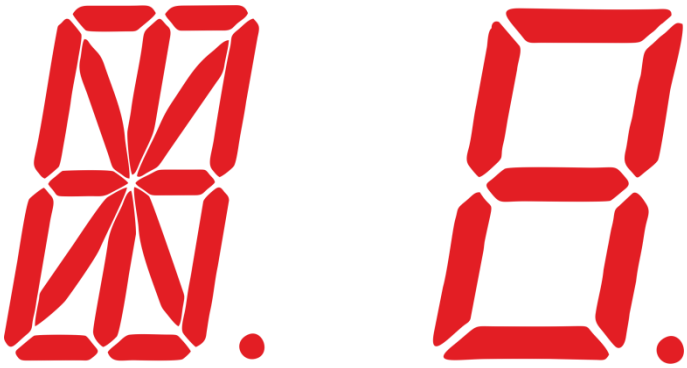
Napíšte program, ktorý bude ovládať diódu jedným tlačidlom (START/STOP). Ošetrite zákmity tlačidla.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

;*****************************************************
CISLO1    equ       20h
CISLO2    equ       21h
#DEFINE   LED       PORTB,0
#DEFINE   TL        PORTA,1

;*****************************************************
          org       000
          goto      START


;*****************************************************
```

```
CAKAJ   movlw   5               ;podprogram oneskorenia
        movwf   CISLO2
SLUCKA2 movlw   0
        movwf   CISLO1
SLUCKA1 decfsz  CISLO1,1
        goto    SLUCKA1
        decfsz  CISLO2,1
        goto    SLUCKA2
        return
;*****************************************************
START   bsf     STATUS,RP0      ;nastavenie vstupov a výstupov
        movlw   B'00000010'
        movwf   TRISA
        movlw   B'00000000'
        movwf   TRISB
        bcf     STATUS,RP0

        clrf    PORTB           ;zhasnutie všetkých diód na porte B
TEST1   btfss   TL              ;čakanie na stlačenie tlačidla
        goto    TEST1
        call    CAKAJ           ;ak je stlačené zavolaj podprogram
        btfss   TL              ;a znovu otestuj tlačidlo
        goto    TEST1           ;ak nie je stlačené vráť sa na TEST1
        bsf     LED             ;ak je stlačené rozsvieť diódu
TEST2   btfsc   TL              ;čakaj na pustenie tlačidla
        goto    TEST2
        call    CAKAJ           ;ak je pustené zavolaj podprogram
        btfsc   TL              ;opäť otestuj
        goto    TEST2           ;ak je stlačené vráť sa na TEST2

TEST3   btfss   TL              ;to isté len dióda sa zhasne
        goto    TEST3
        call    CAKAJ
        btfss   TL
        goto    TEST3
        bcf     LED
TEST4   btfsc   TL
        goto    TEST4
        call    CAKAJ
        btfsc   TL
        goto    TEST4
        goto    TEST1           ;vráť sa na TEST1

        end
```

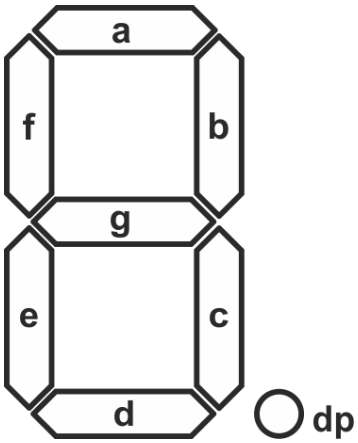
8. Segmentový displej

Je to zobrazovacia jednotka s malou hustotou zobrazovanej informácie. Segmentové displeje sa využívajú prevažne pre zobrazenie číslic, môžu však zobraziť i niektoré špeciálne symboly. Príslušný znak sa zobrazí rozsvietením príslušného segmentu. Medzi najrozšírenejšie segmentové displeje patria sedem a šestnásť segmentové displeje.

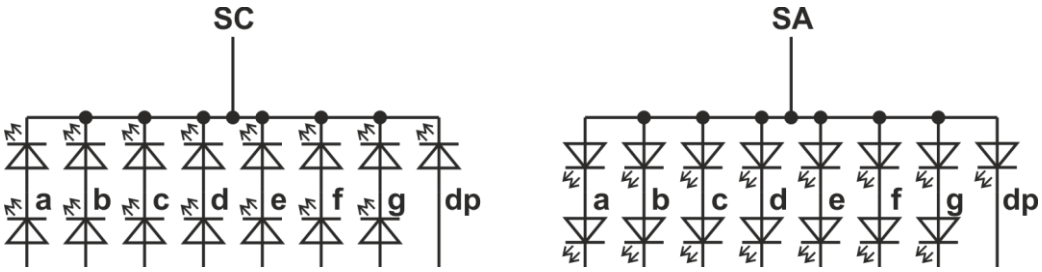


Pomocou sedemsegmentovky je možné napríklad zobraziť všetky číslice šestnástkovej sústavy.

Každý segment má svoje označenie, ktoré je dané a je pre všetky sedemsegmentovky rovnaké. Jednotlivé segmenty sú označené písmenami **a-g** a bodka má označenie **dp**.



Každý segment displeja je tvorený jednou alebo viacerými LED diódami. Podľa ich zapojenia môžeme segmentové displeje rozdeliť na displeje so spoločnou anódou a so spoločnou katódou.



Pri zapojení SA ide o negatívnu logiku (je potrebné pre rozsvietenie daného segmentu priviesť log. 0) a pri SC o pozitívnu logiku (je potrebné pre rozsvietenie daného segmentu priviesť log. 1).

Pri napájaní 5V je k jednotlivým segmentom potrebné predradiť rezistor!

Program č. 27:

Napíšte program, ktorý bude počítať dookola od 0 do F. Čísla sa budú zobrazovať na sedemsegmentovke v zapojení so spoločnou anódou.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

;*************************************
CISLO1   equ      20h
CISLO2   equ      21h

;*************************************
         org      000
         goto     START

;*************************************
CAKAJ    movlw    150
         movwf    CISLO2
SLUCKA2  movlw    150
         movwf    CISLO1
SLUCKA1  decfsz   CISLO1,1
         goto     SLUCKA1
         decfsz   CISLO2,1
         goto     SLUCKA2
         return

;*************************************
START    bsf      STATUS,RP0
         movlw    B'00000000'
         movwf    TRISB
         bcf      STATUS,RP0

ZNOVU    movlw    B'01000000'      ;0
         movwf    PORTB
         call     CAKAJ
         movlw    B'01111100'      ;1
         movwf    PORTB
         call     CAKAJ
         movlw    B'00100100'      ;2
         movwf    PORTB
         call     CAKAJ
         movlw    B'00110000'      ;3
         movwf    PORTB
         call     CAKAJ
         movlw    B'00011001'      ;4
         movwf    PORTB
         call     CAKAJ
         movlw    B'00010010'      ;5
         movwf    PORTB
         call     CAKAJ
```

```
movlw   B'00000010'     ;6
movwf   PORTB
call    CAKAJ
movlw   B'11111000'     ;7
movwf   PORTB
call    CAKAJ
movlw   B'00000000'     ;8
movwf   PORTB
call    CAKAJ
movlw   B'00010000'     ;9
movwf   PORTB
call    CAKAJ
movlw   B'00001000'     ;A
movwf   PORTB
call    CAKAJ
movlw   B'00000011'     ;b
movwf   PORTB
call    CAKAJ
movlw   B'01000110'     ;C
movwf   PORTB
call    CAKAJ
movlw   B'00100001'     ;d
movwf   PORTB
call    CAKAJ
movlw   B'00000110'     ;E
movwf   PORTB
call    CAKAJ
movlw   B'00001110'     ;F
movwf   PORTB
call    CAKAJ

goto    ZNOVU

end
```

NERIEŠENÉ PRÍKLADY - SEDEMSEGMENTOVKA:

Program č. 28:

Napíšte program, ktorý bude ovládať sedemsegmentový displej pomocou tlačidiel PLUS, MINUS a RESET. Na začiatku má na displeji svietiť „0“. Maximálne číslo má byť 5.

Program č. 29:

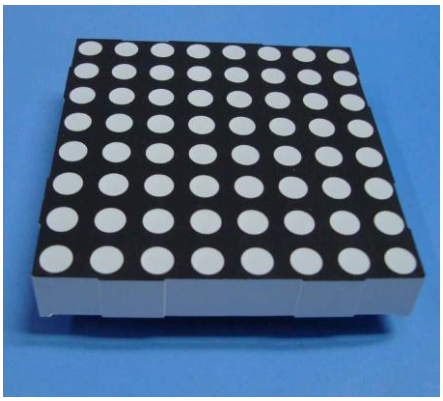
Napíšte program, ktorý vytvorí hraciu kocku pomocou sedemsegmentovky. Na začiatku má svietiť pomlčka (segment g). Po stlačení TL sa na displeji zobrazí číslo „8“ a majú sa miešať čísla 0-6. Po pustení TL sa na displeji zobrazí vyžrebované číslo.

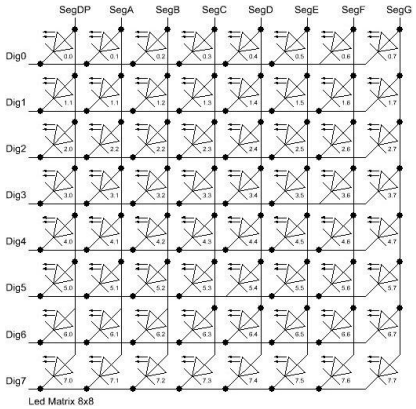
Program č. 30:

Príklad 29 + vyžrebované číslo 5x zabliká a zostane svieti.

9. Maticový displej

Medzi výstupné zariadenia , ktoré je možné pripojiť k mikrokontrolérom patrí aj maticový displej. Je to vlastne skupina diód, ktoré sú uložené do matice. Týmto spôsobom je možné pripojiť k mikrokontroléru viac diód ako máme k dispozícii výstupov (napr. na pripojenie 25 diód nám postačuje 10 výstupov mikrokontroléra). Ak použijeme zložitejšiu schému obsahujúcu napr. posuvný register, je možné počet výstupov ešte skorigovať.

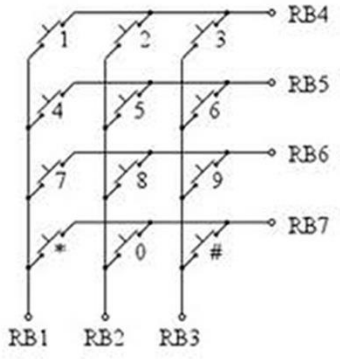




Aj keď maticový displej obsahuje menej výstupov, ako má diód, je možné rozsvietiť každú diódu osobitne. Pre rozsvietenie diódy je potrebné priviesť na daný stĺpec log. 1 (5V) a na daný riadok log. 0 (0V). Zapojenie je potrebné doplniť o rezistory, prípadne tranzistory.

10. Maticová klávesnica

Pri aplikáciách s mikrokontrolérmi je často potrebné mikroprocesoru zadávať rôzne informácie ako sú napríklad čísla. Ak by sme tlačidla pripojili k mikrokontroléru bežným spôsobom (každá číslica potrebuje 1 pin), neostali by nám takmer žiadne voľné piny pre výstupy. Jednou z možností ako pripojiť väčší počet tlačidiel k mikrokontroléru je maticová klávesnica. Tá nám umožní pripojiť k mikrokontroléru 12 (16) tlačidiel pomocou 7 (8) pinov. Maticová klávesnica je vlastne skupina 12 (16) tlačidiel, ktoré sú usporiadané do matice 3x4 (4x4).



Testovanie stlačenia niektorého z tlačidiel na maticovej klávesnici je postupné. Ak máme napríklad klávesnicu zapojenú podľa predošlého obrázku, privádzame postupne kladné napätie na piny RB4 až RB7 (vždy len na jeden) a testujeme prítomnosť napätia na pinoch RB1 až RB3. Ak zistíme na niektorom z daných pinov napätie, vieme, že došlo k stlačeniu príslušného tlačidla.

Pre testovanie klávesnice je vhodné použiť podprogram, ktorý sa spustí ak dôjde k stlačenie nejakého tlačidla. Ušetrí sa tým čas vykonávania programu (v prípade že nie je stlačené tlačidlo, nemusíme testovať všetky možnosti, stačí len testovať prítomnosť napätia na niektorom zo vstupov) a tiež pamäť (v prípade, že by sme testovanie potrebovali použiť v programe niekoľkokrát). Na zistenie stlačenia je tiež možné použiť prerušenie.

Program č.23:

Napíšte program, ktorý bude ovládať diódu pomocou maticovej klávesnice. Dióda má zablikať toľkokrát, aké číslo bolo stlačené.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF
;*****************************************************
CISLO1    equ     20h
CISLO2    equ     21h
CISLO     equ     22h
#DEFINE   LED     PORTA,0
;*****************************************************
          org     000
          goto    START

;*****************************************************
CAKAJ     movlw   0               ;podprogram oneskorenia
          movwf   CISLO2
SLUCKA2   movlw   0
          movwf   CISLO1
SLUCKA1   decfsz  CISLO1,1
          goto    SLUCKA1
          decfsz  CISLO2,1
          goto    SLUCKA2
          return
;*****************************************************
START     bsf     STATUS,RP0    ;nastavenie vstupov a výstupov
          movlw   B'00000000'
          movwf   TRISA
          movlw   B'11110000'
          movwf   TRISB
          bcf     STATUS,RP0

TEST      clrf    PORTA         ;zhasne všetky diódy
          clrf    PORTB         ;testovanie stlačenia
          movlw   0
          bsf     PORTB,1       ;test 1. stĺpca
          btfsc   PORTB,4
          movlw   1
          btfsc   PORTB,5
          movlw   4
          btfsc   PORTB,6
          movlw   7
          btfsc   PORTB,7
          movlw   0
          bcf     PORTB,1
          bsf     PORTB,0       ;test 2. stĺpca
          btfsc   PORTB,4
          movlw   2
          btfsc   PORTB,5
          movlw   5
```

```
        btfsc   PORTB,6
        movlw   8
        btfsc   PORTB,7
        movlw   0
        bcf     PORTB,0
        bsf     PORTB,2         ;test 3. stĺpca
        btfsc   PORTB,4
        movlw   3
        btfsc   PORTB,5
        movlw   6
        btfsc   PORTB,6
        movlw   9
        btfsc   PORTB,7
        movlw   0
        bcf     PORTB,2

        movwf   CISLO           ;reg. W ulož do CISLO
        btfsc   CISLO,0         ;kontrola stlačenia
        goto    BLIK
        btfsc   CISLO,1
        goto    BLIK
        btfsc   CISLO,2
        goto    BLIK
        btfsc   CISLO,3
        goto    BLIK            ;ak bolo stlačene 1-9, chod na BLIK
        goto    TEST            ;ak nebolo stlačene 1-9 vráť sa na TEST
BLIK    bsf     LED             ;rozblikanie LED
        call    CAKAJ
        bcf     LED
        call    CAKAJ
        decfsz  CISLO,1
        goto    BLIK
        goto    TEST
        end
```

Program č.24:

Napíšte program, ktorý bude ovládať diódu pomocou maticovej klávesnice. K mikrokontroléru sú pripojené 4 diódy na porte A (piny 0-3).Po stlačení tlačidla sa rozsvieti príslušný počet diód. Ak stlačíme číslo väčšie ako 4, rozsvietia sa všetky diódy.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF
;*****************************************************
CISLO1   equ      20h
CISLO2   equ      21h
CISLO    equ      22h
#DEFINE  LED      PORTA,0
;*****************************************************
         org      000
         goto     START
;*****************************************************
START    bsf      STATUS,RP0        ;nastavenie vstupov a výstupov
         movlw    B'00000000'
         movwf    TRISA
         movlw    B'00000000'
         movwf    TRISB
         bcf      STATUS,RP0
         movlw    B'11111111'
         movwf    PORTA
         clrf     PORTB             ;testovanie stlačenia
         movlw    0
TEST     bsf      PORTB,0           ;test 2. stĺpca
         btfsc    PORTB,4
         movlw    B'00000011'
         btfsc    PORTB,5
         movlw    B'00001111'
         btfsc    PORTB,6
         movlw    B'00001111'
         btfsc    PORTB,7
         movlw    B'00000000'
         bcf      PORTB,0
         bsf      PORTB,1           ;test 1. stĺpca
         btfsc    PORTB,4
         movlw    B'00000001'
         btfsc    PORTB,5
         movlw    B'00001111'
         btfsc    PORTB,6
         movlw    B'00001111'
         btfsc    PORTB,7
         movlw    B'00000000'
         bcf      PORTB,1
         bsf      PORTB,2           ;test 3. stĺpca
         btfsc    PORTB,4
         movlw    B'00000111'
         btfsc    PORTB,5
```

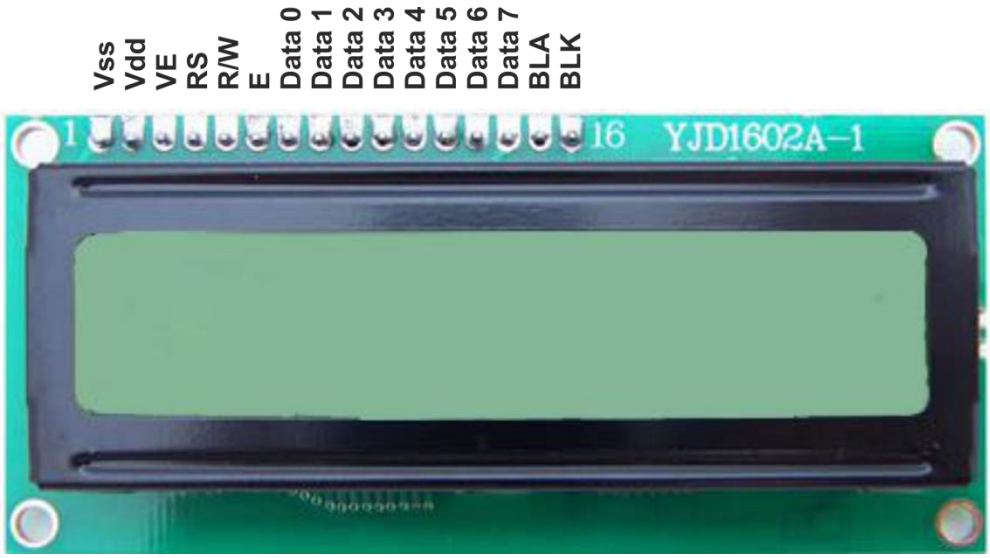
```
movlw   B'00001111'
btfsc   PORTB,6
movlw   B'00001111'
btfsc   PORTB,7
movlw   B'00000000'
bcf     PORTB,2
movwf   PORTA
goto    TEST
end
```

11. LCD znakový displej

Je to zobrazovacia jednotka, ktorá umožňuje zobrazovať vopred definované znaky. Väčšina LCD znakových displejov obsahuje riadiaci obvod (radič) HD44780 firmy HITACHI. Je to vlastne mikropočítač, ktorý je súčasťou LCD. Pomocou tohto obvodu je možné komunikovať s displejom. Tento radič obsahuje znakovú sadu a inštrukcie pre ovládanie displeja. Zobraziteľná plocha môže obsahovať jeden až štyri riadky a každý riadok môže obsahovať 8 až 40 znakov.

11.1 Vývody LCD displeja

LCD displej obsahuje 16 vývodov. Vývody 15 a 16 sú voliteľné a je možné ich použiť ak displej obsahuje podsvietenie. Ak displej neobsahuje podsvietenie, väčšinou piny 15 a 16 nie sú vôbec vyvedené.



- V_{SS}–zem napájacieho napätia
- V_{DD} – plus napájacieho napätia
- VE – kontrast
- RS - určuje prenos dát, alebo inštrukcií
  - 0 – prenos inštrukcií
  - 1 – prenos dát
- R/W – voľba zápisu/čítania
  - 0 – zápis do LCD
  - 1 – čítanie z LCD
- E – potvrdzovací signál
  - kladný signál štartuje komunikáciu medzi zariadeniami
- Data 0-7 – dátové piny
- BLA – podsvietenie – anóda
- BLK – podsvietenie - katóda

11.2 Základné zapojenie

Zapojenie základných vývodov LCD modulu (napájanie, kontrast, podsvietenie) je zobrazené v nasledujúcej schéme. Okrem napájania samotného displeja (vývody 1 a 2), je potrebné zapojiť aj kontrast (3), ktorý sa nastavuje potenciometrom (trimrom).

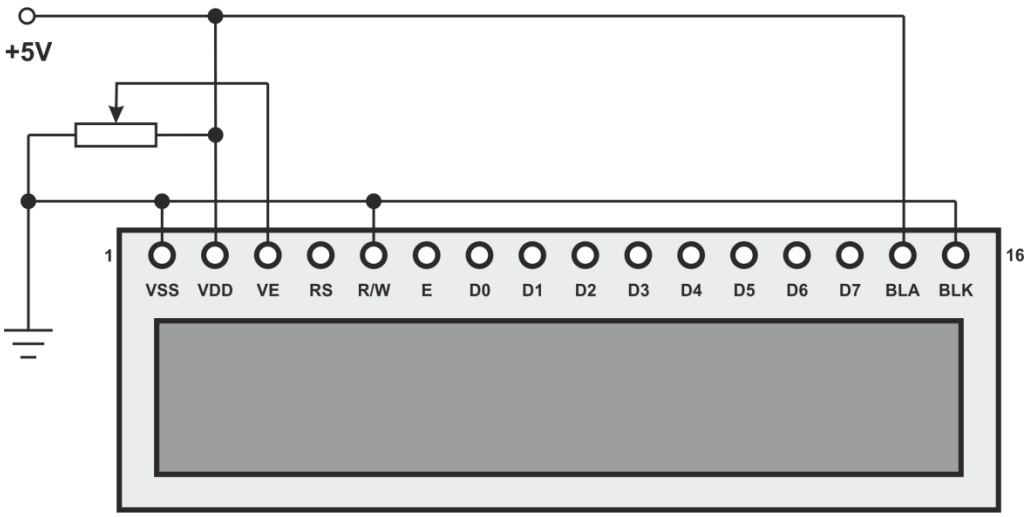
Vývod R/W (5) je väčšinou pripojený k zemi. Týmto vývodom sa nastavuje, či chceme na displej zapisovať alebo z displeja čítať (z/do pamäte radiča v displeji). Pretože čítanie z displeja je veľmi zriedkavé, obvykle sa tento vývod pripojí k zemi a na displej je umožnené iba zapisovať, čo pre väčšinu aplikácií postačuje a zároveň tým ušetríme jeden vývod mikrokontroléra.

Ak displej obsahuje podsvietenie zapojíme aj piny 15 a 16. Ostatné vývody slúžia na programovanie, pripájajú sa na mikrokontrolér a sú nastavené ako výstupy.

Pri komunikácii mikrokontroléra s displejom je možné využiť dva základné spôsoby komunikácie:

- 8–bitová komunikácia – vyžaduje 11(10) vstupno/výstupných pinov mikrokontroléra
- 4–bitová komunikácia – vyžaduje 7(6) vstupno/výstupných pinov mikrokontroléra
  - na LCD displeji sa pripojí horná polovica dátových pinov DB7 – DB4
  - dolné dátové piny displeja sa uzemnia
  - rozdelené dátové slová sa posielajú postupne (najprv sa pošlú horné 4 bity a potom dolné 4)

Základné zapojenie LCD displeja:



11.3 Znaková sada displeja

Každý znak je zobrazovaný ako matica 5x8 bodov. Definícia jednotlivých znakov je napevno uložená vo vnútornej pamäti ROM. Ôsmy riadok každého znaku je prázdny. Je to z dôvodu možnosti zobrazenia kurzora práve v tomto riadku. Keďže takmer každý jazyk má aj vlastné špecifické symboly, je možné dodefinovať si osem vlastných symbolov. Pri ich definícii je možné využívať celú maticu 5x8.

11.4 Inštrukčná sada displeja

Inštrukčná sada displeja obsahuje 11 inštrukcií, ktoré nám umožňujú pracovať s displejom. S ich pomocou je možné nastaviť základné vlastnosti displeja, jeho správanie sa a umožňujú nám vytvárať nové znaky.

Inštrukcia	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	dĺžka	Popis
Zmazať displej	0	0	0	0	0	0	0	0	0	1	1,64 ms	Zmaže displej a nastaví DDRAM na 0
Návrat na pozíciu 0	0	0	0	0	0	0	0	0	1	*	1,64 ms	Nastaví DDRAM na 0 – kurzor na pozícii 0. Obsah DDRAM sa nemení
Nastavenie módu	0	0	0	0	0	0	0	1	I/D	S	40 us	smer posuvu kurzora I/D (0=vľavo, 1=vpravo), posuv textu S (0=nie, 1=áno)
Kontrola displeja	0	0	0	0	0	0	1	D	C	B	40 us	D - zapne displej, C - zapne kurzor, B - zapne blikanie kurzora
Posun kurzora, displeja	0	0	0	0	0	1	S/C	R/L	*	*	40 us	1x posunie (S/C=0 kurzor, S/C=1 text) smerom (R/L=0 vľavo, R/L=1 vpravo)
Nastavenie funkcie	0	0	0	0	1	DL	N	F	*	*	40 us	DL=0 4-bit, DL=1 8-bit mód N=0 jednoriadkový, N=1 dvojriadkový disp. F=0 font 5x8, F=1 font 5x10
Nastavenie adresy CGRAM	0	0	0	1	CGRAM adresa						40 us	Nastavenie adresy CGRAM. Dáta sú prenášané po tomto nastavení

Nastavenie adresy DDRAM	0	0	1	DDRAM adresa	40 us	Nastavenie adresy DDRAM. Dáta sú prenášané po tomto nastavení
Čítanie príznaku BUSY Flag a adresy	0	1	BF	CGRAM/DDRAM adresa	0 us	Čítanie príznaku BF a čítanie adresy BF=1 radič zaneprázdnený
Zápis dát do CGRAM alebo DDRAM	1	0	Dáta		40 us	
Čítanie dát z CGRAM alebo DDRAM	1	1	Dáta		40 us	

11.5 Pamäť pre zobrazovanie znakov

Dáta, ktoré sa majú zobraziť na displeji sú uložené v pamäti DDRAM (Display Data Random Acces Memory). Každý zobrazovaný znak má svoju adresu.

Rozsah pamäte DDRAM:

- jednoriadkový displej

pozícia znaku (dec)	0	1	2	3	4	.....	34	35	36	37	38	39
adresa znaku (hex)	00	01	02	03	04	.....	22	23	24	25	26	27

- dvojriadkový displej

pozícia znaku (dec)	0	1	2	3	4	.....	34	35	36	37	38	39
adresa znaku (hex) – 1. riadok	00	01	02	03	04	.....	22	23	24	25	26	27
adresa znaku (hex) – 2. riadok	40	41	42	43	44	.....	62	63	64	65	66	67

- štvorriadkový displej

pozícia znaku (dec)	0	1	2	3	4	.....	14	15	16	17	18	19
adresa znaku (hex) – 1. riadok	00	01	02	03	04	.....	0E	0F	10	11	12	13
adresa znaku (hex) – 2. riadok	40	41	42	43	44	.....	4E	4F	50	51	52	53
adresa znaku (hex) – 3. riadok	14	15	16	17	18	.....	22	23	24	25	26	27
adresa znaku (hex) – 4. riadok	54	55	56	57	58	.....	62	63	64	65	66	67

Adresy zobrazovaných znakov:

Jednoriadkové displeje

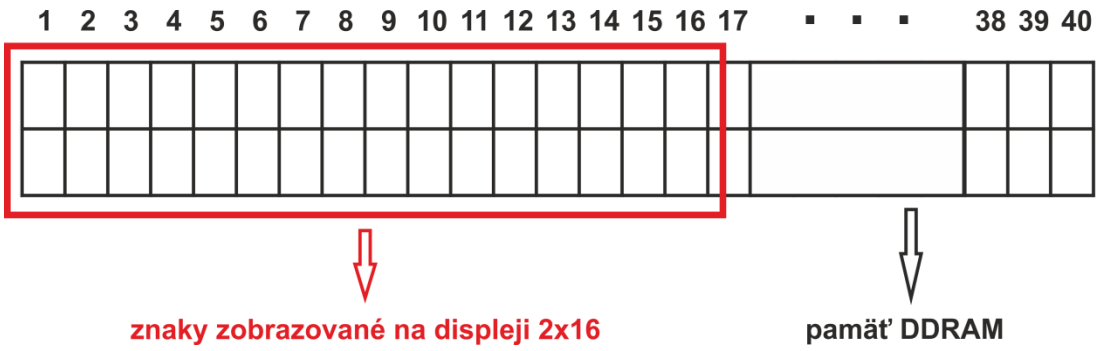
počet znakov	pozícia v DDRAM
1 x 16	00h – 0Fh
1 x 24	00h – 17h

Dvojriadkové displeje

počet znakov	pozícia v DDRAM
2 x 8	00h – 07h 40h – 47h
2 x 16	00h – 0Fh 40h – 4Fh
2 x 20	00h – 13h 40h – 53h
2 x 24	00h – 17h 40h – 57h
2 x 40	00h – 27h 40h – 67h

Štvorriadkové displeje

počet znakov	pozícia v DDRAM
4 x 16	00h – 0Fh 40h – 4Fh 10h – 1Fh 50h – 5Fh
4 x 20	00h – 13h 40h – 53h 14h – 27h 54h – 67h
4 x 24	00h – 17h 40h – 57h 18h – 2Fh 58h – 6Fh



11.6 Posielanie príkazov a dát

Príkazy a dáta sú do displeja posielané po ôsmych dátových vodičoch (pri 8-bitovej komunikácii). Vodičom RS (vývod 4) posielame displeju informáciu o tom, či sa na dátových vodičoch nachádza príkaz alebo dáta. Ak RS = 0, znamená to, že radič displeja si interpretuje informáciu, ktorá sa nachádza na dátových vodičoch ako príkaz. Ak RS = 1, radič si v tomto prípade interpretuje informáciu na dátových vodičoch ako dáta. Príkaz alebo dáta sú do radiča zapísané pri dobežnej hrane signálu na vodiči E (vývod 6), týmto sa myslí prechod z vysokej úrovne (log.1) na nízku úroveň (log.0).

Takže, ak chcete zapísať príkaz alebo dáta do radiča displeja, potrebujete vykonať nasledujúcu sekvenciu:

1. zvoľte typ posielaných údajov pomocou vodiča RS
2. nastavte na vodiči E vysokú log. úroveň (log.1)
3. nastavte na dátových vodičoch požadované hodnoty (log.1 alebo log.0)

   Pri 4-bitovej komunikácii sa posielajú najprv horné 4 bity a potom dolné. Posielanie je dvakrát pomalšie.

4. nastavte na vodiči E nízku log. úroveň (log.0)

Poradie prvých dvoch krokov je možné vymeniť.

Medzi jednotlivými vyššie spomínanými operáciami existujú isté minimálne hodnoty časov, ktoré musia byť dodržané pre korektné fungovanie zápisu dát na displej.

11.7 Inicializácia displeja

Po zapnutí displeja sa vždy vykoná tzv. vnútorný reset:

- vymaže sa displej
- nastaví sa funkcia
  - 8 - bitová komunikácia
  - 1 – riadkový displej
  - veľkosť znakov 5x8
- kontrola displeja
  - displej – vypnutý
  - kurzor – vypnutý
  - blikanie kurzora – vypnuté
- nastavenie módu
  - I/D – zvýšenie
  - posun displeja – vypnutý

Keďže pre väčšinu aplikácií nie je toto nastavenie vhodné, je potrebné vykonať inicializáciu LCD displeja po každom zapnutí.

Inicializácia displeja pre 8-bitovú komunikáciu

Zapnutie napájania										Poznámka
Čakaj minimálne 15 ms po nábehu U_{CC} nad 4,5V										
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	8 - bitová komunikácia
0	0	0	0	1	1	-	-	-	-	
Čakaj min. 4,1 ms										
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	
0	0	0	0	1	1	-	-	-	-	
Čakaj min. 100us										
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	
0	0	0	0	1	1	-	-	-	-	
RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	Vždy čakaj 40us
0	0	0	0	1	1	N(1)	F(0)	-	-	2 riadky, znaky 5x8 bodov
0	0	0	0	0	0	1	0	0	0	Vypnúť displej, kurzor i blikanie kurzora
0	0	0	0	0	0	0	0	0	1	Zmazať displej a nastaviť DDRAM na 0
0	0	0	0	0	0	0	1	I/D(1)	S(0)	Zapnutý pohyb kurzora
0	0	0	0	0	0	1	1	0	0	Zapnutie displeja

11.8 Programovanie LCD displeja – výpis textov na LCD (8-bit)

Pri programovaní LCD displeja sa v podstate využíva len niekoľko inštrukcií, ktoré sa opakujú. Preto je vhodné vytvoriť si niekoľko podprogramov, ktoré nám uľahčia prácu a tiež ušetria miesto v pamäti mikrokontroléra. Je dobré vytvoriť si podprogramy na všetky potrebné časové oneskorenia, podprogram na inicializáciu displeja, jeho vymazanie, výpis reťazca, znaku či posun displeja.

Program č.25:

Napíšte program, ktorý vypíše do prvého riadku LCD displeja „Moj prvy program“

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _RC_OSC & _PWRTE_ON & _WDT_OFF & _CP_ON

RAM     equ     20h
CISLO1  equ     RAM+1
CISLO2  equ     RAM+2
CAS1    equ     RAM+3
CAS2    equ     RAM+4
TEXT    equ     RAM+5
ZNAK    equ     RAM+6

#DEFINE E       PORTA,0
#DEFINE RS      PORTA,1

        org     0
        goto    START
;***********************************
;*** oneskorovacie podprogramy
CAK15m  movlw   21
        movwf   CAS1
        movlw   0
        movwf   CAS2
        call    CAKAJ
        return
CAK4m1  movlw   6
        movwf   CAS1
        movlw   0
        movwf   CAS2
        call    CAKAJ
        return
CAK100u movlw   1
        movwf   CAS1
        movlw   35
        movwf   CAS2
        call    CAKAJ
        return
CAK40u  movlw   1
        movwf   CAS1
```

```
            movlw   15
            movwf   CAS2
            call    CAKAJ
            return
CAKAJ       movf    CAS1
            movwf   CISLO2
SLUCKA2     movf    CAS2
            movwf   CISLO1
SLUCKA1     decfsz  CISLO1,1
            goto    SLUCKA1
            decfsz  CISLO2,1
            goto    SLUCKA2
            return
;***********************************
;*** inicializácia displeja
LCD_INI     call    CAK15m
            bcf     RS
            bsf     E
            movlw   B'00110000'
            movwf   PORTB
            bcf     E
            call    CAK4m1
            movlw   B'00111000'
            call    PIS_IN
            movlw   B'00001000'
            call    PIS_IN
            movlw   B'00000001'
            call    PIS_IN
            movlw   B'00000110'
            call    PIS_IN
            movlw   B'00001100'
            call    PIS_IN
            return
;***********************************
;*** tabuľka textov
TAB_TXT     addwf   PCL,1
            retlw   'M'             ;0
            retlw   'o'
            retlw   'j'
            retlw   ' '
            retlw   'p'
            retlw   'r'
            retlw   'v'
            retlw   'y'
            retlw   ' '
            retlw   'p'             ;9
            retlw   'r'
            retlw   'o'
            retlw   'g'
            retlw   'r'
            retlw   'a'
            retlw   'm'
            retlw   80h
```

```
;**********************************
;*** zapísanie textu
PIS_TXT   movwf    TEXT
          call     TAB_TXT
          movwf    ZNAK
          sublw    0x80
          btfsc    STATUS,Z
          return
          movf     ZNAK,0
          call     PIS_ZNAK
          incf     TEXT,0
          goto     PIS_TXT
;**********************************
;*** zapísanie znaku
PIS_ZNAK  bsf      RS
          bsf      E
          movwf    PORTB
          bcf      E
          call     CAK40u
          return
;**********************************
;*** zapísanie inštrukcie
PIS_IN    bcf      RS
          bsf      E
          movwf    PORTB
          bcf      E
          call     CAK40u
          return
;**********************************
;*** nastavenie vstupov a výstupov
;*** začiatok programu
START     bsf      STATUS,RP0

          clrf     TRISA
          clrf     TRISB
          bcf      STATUS,RP0
          clrf     PORTA
          clrf     PORTB

          call     LCD_INI

;**********************************
;*** hlavný program
          movlw    0
          call     PIS_TXT
          end
```

11.9 Programovanie LCD displeja – presun kurzora a vymazanie displeja

Ak programujeme viacriadkový displej, je potrebné sa presúvať medzi jednotlivými riadkami. Je to možné vykonať pomocou nastavenia adresy v pamäti DDRAM. Týmto spôsobom je možné presunúť kurzor na ľubovoľnú pozíciu v pamäti.

Inštrukcia	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Nastavenie adresy DDRAM	0	0	1	DDRAM adresa						

Napríklad druhý riadok displeja sa začína adresou 40h. Ak to prevedieme do binárnej sústavy, dostaneme číslo „1000000“. Ak ho pridáme do kódu inštrukcia, získame číslo „11000000“.

```
;**********************************
;*** presun kurzora na druhy riadok
RIADOK2 movlw   B'11000000'
        call    PIS_IN
        return
```

Vymazať displej je možné takmer rovnakým podprogramom. Jedinou zmenou je kód inštrukcie, ktorý má pre vymazanie displeja hodnotu „00000001“. Okrem vymazania dôjde i k návratu kurzora na prvý znak prvého riadku.

```
;**********************************
;*** vymazanie displeja
VYMAZ   movlw   B'00000001'
        call    PIS_IN
        return
```

Program č.26:

Doplňte predošlý program tak, aby na prvom riadku vypísal „Môj prvý program“ a na druhom „s druhým riadkom“.

Program 24 doplníme o podprogram pre presun kurzora do druhého riadku.

```
;***********************************
;*** presun kurzora na druhy riadok
RIADOK2  movlw    B'11000000'
         call     PIS_IN
         return
```

Je potrebné tiež doplniť tabuľku textov

```
;***********************************
;*** tabuľka textov
TAB_TXT  addwf    PCL,1
         retlw    'M'              ;0
         retlw    'o'
         retlw    'j'
         retlw    ' '
         retlw    'p'
         retlw    'r'
         retlw    'v'
         retlw    'y'
         retlw    ' '
         retlw    'p'
         retlw    'r'
         retlw    'o'
         retlw    'g'
         retlw    'r'
         retlw    'a'
         retlw    'm'
         retlw    80h
         retlw    's'              ;17
         retlw    ' '
         retlw    'd'
         retlw    'r'
         retlw    'u'
         retlw    'h'
         retlw    'y'
         retlw    'm'
         retlw    ' '
         retlw    'r'
         retlw    'i'
         retlw    'a'
         retlw    'd'
         retlw    'k'
         retlw    'o'
         retlw    'm'
         retlw    80h
```

a doplniť telo programu.

```
;**********************************
;*** hlavný program
        movlw   0
        call    PIS_TXT
        call    RIADOK2
        movlw   17
        call    PIS_TXT
        end
```

Program č.27:

Napíšte program, ktorý po zapnutí mikrokontroléra vypíše na LCD „Technicka“ a po 2s „akademia“. Vždy po 2s sa budú nápisy opakovať.

Program č.28:

Napíšte program, ktorý po stlačení TL1 vypíše „Programovanie“ a po stlačení TL2 „mam rád“.

11.10 Programovanie LCD displeja – posúvanie textov

Pri programovaní LCD sa často využíva posúvanie textov. Je možné ho použiť pri postupnom zobrazovaní dlhších textov, alebo na postupné vymazanie textov.Na posúvanie textov nám slúži inštrukcia posunu, s ktorou je možné posúvať aj kurzor o jedno miesto vpravo, alebo vľavo.

Inštrukcia	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
Posun kurzora, displeja	0	0	0	0	0	1	S/C	R/L	*	*

- S/C – nastavenie pohybu
  - 0 – kurzora
  - 1 - textu
- R/L – smer pohybu
  - 0 – doľava
  - 1 - doprava

```
;**********************************
;*** posun textu doľava
POSUNL  movlw   B'00011000'
        call    PIS_IN
        return

;**********************************
;*** posun textu doprava
POSUNR  movlw   B'00011100'
        call    PIS_IN
        return
```

Program č.29:

Napíšte program, ktorý do prvého riadku vypíše „Technicka akademia, Spisska Nova Ves“ a potom bude posúvať text stále doľava.

V programe je potrebné zmeniť tabuľku textov, aby obsahovala požadovaný text. Ďalej je potrebné doplniť podprogram pre posunutie textu doľava.

```
;***********************************
;*** posun textu doľava
POSUNL   movlw    B'00011000'
    call    PIS_IN
    return
```

Poslednú zmenu je potrebné spraviť v tele hlavného programu.

```
;***********************************
;*** hlavný program
         movlw    0
         call     PIS_TXT
OPAKUJ   call     CAK15m
         call     POSUNL
         goto     OPAKUJ
         end
```

Program č.30:

Napíšte program, ktorý bude ovládať LCD displej pomocou troch tlačidiel. Po stlačení prvého tlačidla sa do prvého riadku vypíše „Technicka akademia“. Po stlačení druhého tlačidla sa vypíše do druhého riadku „Spisska Nova Ves“. Po stlačení tretieho text odíde doľava a ostane prázdny displej. Tlačidlá má byť možné stláčať v ľubovoľnom poradí.

Program č.31:

Napíšte program pre vytvorenie hracej kocky na LCD displeji. Na začiatku bude na displeji napísané „stlac tlacidlo“. Po jeho stlačení sa na displeji vypíše „zrebujem“. Po pustení tlačidla sa zobrazí v prvom riadku vyžrebované číslo napr. „cislo: 6“ a v druhom riadku „stlac tlacidlo“.

11.11 4-bitová komunikácia

Pri programovaní displeja pomocou štyroch dátových vodičov je potrebné (okrem zapojenia) zmeniť len základné podprogramy a inicializáciu.

```
INI_LCD call   CAK15m
        movlw  30h
        movwf  PORTB
        call   CAK4m1
        movlw  30h
        movwf  PORTB
        bsf    E
        bcf    E
        call   CAK4m1
        bsf    E
        bcf    E
        call   CAK40u
        bsf    E
        bcf    E
        call   CAK40u
        movlw  20h
        movwf  PORTB
        bsf    E
        bcf    E
        call   CAK40u
        movlw  28h
        call   PIS_IN
        movlw  01h
        call   PIS_IN
        call   CAK4m1
        movlw  0x06
        call   PIS_IN
        movlw  0x0C
        call   PIS_IN
        return
```

Z podprogramov je potrebné zmeniť len PIS_IN a PIS_ZNAK, pretože údaje sa budú posielať postupne (po 4 bitoch). Keďže na displeji môžu byť použité ostatné piny portu B, je potrebné ich maskovať. Pre maskovanie sa využíva register TEMP (pre uloženie konštanty) a logický súčin AND v inštrukcii *andlw*, ktorý zamaskuje dolné štyri bity.

```
PIS_IN  movwf  TEMP
        andlw  0xF0
        movwf  PORTB
        bsf    E
        bcf    E
        swapf  TEMP,0
        andlw  0xF0
        movwf  PORTB
        bsf    E
        bcf    E
        call   CAK40u
        return
```

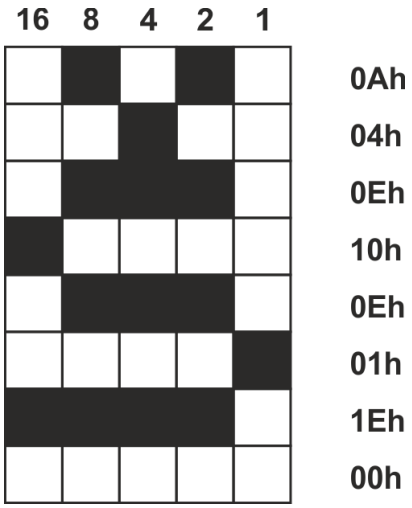
```
PIS_ZNAK movwf  TEMP
         andlw  0xF0
         movwf  PORTB
         bsf    RS
         bsf    E
         bcf    E
         movfw  TEMP
         swapf  TEMP,0
         andlw  0xF0
         movwf  PORTB
         bsf    RS
         bsf    E
         bcf    E
         call   CAK40u
         return
```

11.12 Vytváranie vlastných znakov

LCD displeje sú vyrábané pre celý svet rovnaké. Obsahujú rovnaký radič i znaky. Takmer každý jazyk však obsahuje vlastné špecifické znaky. U nás sú to najmä dĺžne a mäkčene. V LCD displeji je možné naraz zadefinovať osem rôznych znakov, každý v rastri 5x8 bodov.

Pre definovanie vlastných znakov nám slúži pamäť CGRAM s veľkosťou 64 bytov.Adresa prvého znaku sa začína na 40h. Jednotlivé znaky sú v nej adresované adresami 00h až 07h (prípadne 08h až 0Fh).

Pri vytváraní znakov je dobré si nakresliť znak do mriežky a vypísať kódy jednotlivých riadkov.



Pre zadefinovanie znaku je potrebné poslať kódy jednotlivých riadkov do pamäte CGRAM. Pre znak 0 sú adresy 00h – 08h.

Samotné znaky je najlepšie do LCD displeja nahrať hneď po inicializácii. Keďže zadefinovanie znakov sa skladá z opakovaného vkladania kódov jednotlivých riadkov je vhodné vytvoriť si podprogramy, ktoré tento proces sprehľadnia a ušetria miesto v pamäti.

Príklady podprogramov:

```
;***********************************
;*** nahranie znakov do LCD

SK_INI   clrf    ADRESA
         movlw   0x40              ; prva adresa znaku v LCD displeji
         call    PIS_IN
SK_INC   movf    ADRESA,0
         call    TAB_SK
         call    PIS_ZNAK
         incf    ADRESA,1
         movlw   0x40              ; ked ADRESA dosiahne 40h tak je koniec
         subwf   ADRESA,0
         btfss   STATUS,Z
         goto    SK_INC
         return
```

Prvý podprogram nám slúži na postupné zapisovanie kódov jednotlivých znakov do pamäti CGRAM. Postupne zvyšuje adresu v pamäti od 40h a zapisuje do nej 64 kódov (8 znakov - každý má 8 riadkov=kódov). Druhý podprogram slúži ako tabuľka znakov, z ktorej prvý podprogram postupne číta kódy.

```
;***********************************
;*** tabulka znakov
TAB_SK  addwf   PCL,1                   ; definovanie slovenských znakov pre LCD
;       ------------------------
        retlw   0x02                    ; znak0 = 40h     á
        retlw   0x04
        retlw   0x1E
        retlw   0x01
        retlw   0x0F
        retlw   0x11
        retlw   0x0F
        retlw   0x00
;       ------------------------
        retlw   0x02                    ; znak1 = 48h     í
        retlw   0x04
        retlw   0x0C
        retlw   0x04
        retlw   0x04
        retlw   0x04
        retlw   0x0E
        retlw   0x00
;       ------------------------
        retlw   0x02                    ; znak2   é
        retlw   0x04
        retlw   0x0E
        retlw   0x11
        retlw   0x1F
        retlw   0x10
        retlw   0x0E
        retlw   0x00
;       ------------------------
        retlw   0x0A                    ; znak3   š
        retlw   0x04
        retlw   0x0E
        retlw   0x10
        retlw   0x0E
        retlw   0x01
        retlw   0x1E
        retlw   0x00
;       ------------------------
        retlw   0x0A                    ; znak4 = 60h     č
        retlw   0x04
        retlw   0x0E
        retlw   0x10
        retlw   0x10
        retlw   0x11
        retlw   0x0E
        retlw   0x00
;       ------------------------
        retlw   0x02                    ; znak5 = 68h     ú
        retlw   0x04
        retlw   0x11
        retlw   0x11
```

```
        retlw   0x11
        retlw   0x13
        retlw   0x0D
        retlw   0x00
;       ------------------------
        retlw   0x0A                    ; znak6 = 70h       ž
        retlw   0x04
        retlw   0x1F
        retlw   0x02
        retlw   0x04
        retlw   0x08
        retlw   0x1F
        retlw   0x00
;       ------------------------
        retlw   0x02                    ; znak7 = 78h       ý
        retlw   0x04
        retlw   0x11
        retlw   0x11
        retlw   0x0F
        retlw   0x01
        retlw   0x0E
        retlw   0x00
```

Ak chceme vypísať napr. písmeno š do tabuľky textov vložíme riadok

```
        retlw   03h
```

Program č.32

Napíšte program, ktorý vypíše do prvého riadku „Slovenské znaky:“ a do druhého riadku vypíše 8 slovenských znakov.

Program č.33

Vytvorte aspoň dva vlastné znaky a použite ich v programe.

12. Prerušenia

Prerušenie je funkcia, ktorá po dopredu nastavenej akcii preruší aktuálnu činnosť procesora a vykoná obslužný program. Po dokončení tohto podprogramu sa vykonávanie programu vráti na miesto, kde bol prerušený.

Mikrokontrolér PIC 16F84 pozná štyri druhy prerušení:

- externý pin RB0/INT
- pretečenie časovača
- zmena na pinoch 4-7 na porte B
- dokončenie zápisu do EEPROM

Po prijatí prerušenia mikrokontrolér skočí na adresu 04h. Keďže po resete alebo zapnutí napájania mikrokontrolér začína na adrese 00h, zostávajú štyri voľné miesta v pamäti medzi začiatkom programu a začiatkom obsluhy prerušenia. Tie sa väčšinou využívajú pre inštrukciu skoku na hlavný program a pre inštrukcie volania inicializačných podprogramov. Pred ošetrením samotného prerušenia je potrebné zakázať ostatné prerušenia, prípadne uložiť obsahy registrov, ktoré sú potrebné pre bezchybné pokračovanie programu. Po ošetrení prerušenia sa vrátime späť k vykonávaniu hlavného programu pomocou inštrukcie *retfie*.

Pri ošetrovaní prerušení sa využíva register INTCON. Štyri horné bity povoľujú a zakazujú prerušenia a štyri spodné bity sú príznaky jednotlivých typov prerušení.

INTCON – je to posledný systémový register, ktorý slúži na nastavenie prerušení a tiež indikuje typ prerušenia.

D7	D6	D5	D4	D3	D2	D1	D0
GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

- **GIE** – povoľuje akékoľvek prerušenie
  - 0 – zakázané
  - 1 – povolené
- **EEIE** – prerušenie po dokončení zápisu do EEPROM
  - 0 – zakázané
  - 1 – povolené
- **TOIE** – prerušenie po pretečení časovača TMR0
  - 0 – zakázané
  - 1 – povolené
- **RBIE** – prerušenie zmenou na pinoch 4-7 na porte B
  - 0 – zakázané
  - 1 – povolené
- **TOIF** – príznak prerušenia TMR0 (je potrebné ručne vynulovať)
  - 0 – nebol
  - 1 – bol
- **INTF** – príznak vnútorného prerušenia (je potrebné ručne vynulovať)
  - 0 – nebol
  - 1 – bol
- **RBIF**– príznak prerušenia od portu B (je potrebné ručne vynulovať)
  - 0 – nebol
  - 1 – bol

Program č.34

Napíšte program, ktorý po stlačení tlačidla rozsvieti diódu pripojenú na RB0 a po opätovnom stlačení ju zhasne (START/STOP tlačidlo). Tlačidlo je pripojene na RB7. Použite prerušenie.

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF
;*************************************
            org     0                   ;zaciatok programu
            goto    START               ;skok na START

            org     4                   ;pri prerusenidojde k skoku na adr. 4
            bcf     INTCON,GIE          ;zakazprerusenie
            bcf     INTCON,RBIF         ;vypni priznakprerusenia
            goto    PRERUS              ;chod na osetrenieprerusenia - PRERUS – podprogram
;*************************************
;*** osetrenieprerusenia
PRERUS      btfsc   PORTB,7             ;ak je stlacene TL vrat sa na PRERUS - caka na pustenie
            goto    PRERUS
            btfsc   PORTB,0
            goto    ZHASNI
            goto    SVIET
SVIET       bsf     PORTB,0
            goto    POVOL
ZHASNI      bcf     PORTB,0
POVOL       movlw   B'10001000'         ;povolenie preruseni na pinoch 4-7 na PORTB
            movwf   INTCON
            retfie                      ;navrat z prerusenia
;*************************************
;*** hlavny program
START       bsf     STATUS,RP0          ;nastavenie vstupov/vystupov
            clrf    TRISA               ;PORTA výstupy/nepouzite
            clrf    TRISB               ;PORTB vyspupy/nepouzite
            bsf     TRISB,7             ;PORTB7 - vstup
            bcf     STATUS,RP0
            movlw   B'10001000'         ;povolenie preruseni na pinoch 4-7 na PORTB
            movwf   INTCON
            clrf    PORTB
OPAKUJ      nop                         ;nekonečná slučka
            goto    $-1

            end
```

Program č.35:

Napíšte program, ktorý bude ovládať rýchlosť blikania LED diódy pomocou troch tlačidiel. Dióda má blikať už po pripojení napájania. Použite prerušenie.

Program č.36:

Napíšte program, ktorý bude ovládať počet svietiacich LED diód pomocou troch tlačidiel PLUS (rozsvieti sa o 1 diódu viac), MINUS (rozsvieti sa o 1 diódu menej), RESET (všetky diódy zhasnú). Použite prerušenie.

13. Časovač Timer0

Timer0 je modul, ktorý môže pracovať v režime časovača aj čítača (závisí od zdroja signálu pre tento modul). Pre prácu s modulom Timer0 sa využívajú systémové registre OPTION_REG a TMR0. Register OPTION_REG slúži na nastavenie funkcie modulu a register TMR0 obsahuje hodnotu čítača/časovača.

OPTION_REG

D7	D6	D5	D4	D3	D2	D1	D0
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

- **RBPU** – povoľuje pripojenie pull-up rezistorov na všetky vývody PORTB, konfigurované ako vstupné
  - 1 – zakázané – potrebné vonkajšie rezistory
  - 0 – pripojené
- **INTEDG** – voľba aktívnej hrany pre vonkajšie prerušenie
  - 1 – nábežná hrana na RB0/INT
  - 0 – zostupná hrana na RB0/INT
- **T0CS** – voľba zdroja hodinového signálu pre modul Timer0
  - 1 – vonkajší zdroj na vývode RA4/T0CKI - čítač
  - 0 – vnútorný zdroj závislý na taktovacej frekvencii - časovač
- **T0SE** – voľba aktívnej hrany pri vonkajšom zdroji hodinového signálu
  - 1 – inkrementácia so zostupnou hranou na RA4/T0CKI
  - 0 – inkrementácia s nábežnou hranou na RA4/T0CKI
- **PSA** – voľba použitia preddeličky
  - 1 – preddelička použitá pre časovač Watchdog (WDT)
  - 0 – preddelička použitá pre časovač TMR0
- **PS2:PS0** – voľba deliaceho pomeru preddeličky

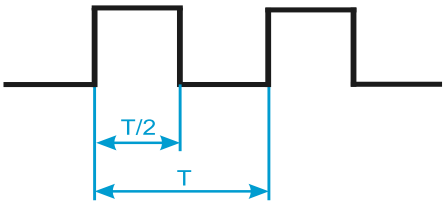
	TMR0	WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

V režime časovača je register TMR0 inkrementovaný v každom inštrukčnom cykle (pokiaľ nie je zaradená preddelička). Pri pretečení registra TMR0 (255 -> 0) môže dôjsť k prerušeniu. Je potrebné ho zapnúť v registri INTCON – bit T0IE. Potrebné nulovať príznak prerušenia T0IF.

Program č.35

Napíšte program, ktorý vytvorí na bzučiaku (PORTB,3) tón s frekvenciou 800Hz. Využite časovač Timer0.

Pre nastavenie časovača na danú frekvenciu potrebujeme vypočítať potrebnú preddeličku a začiatočnú hodnotu časovača TMR0. Začneme výpočtom periódy a polperiódy signálu (doba po ktorej sa mení hodnota signálu).



$$T = \frac{1}{f} = \frac{1}{800Hz} = 0{,}00125s = 1250\mu s$$

$$\frac{T}{2} = \frac{1250\mu s}{2} = 625\mu s$$

Register TMR0 môže nadobúdať len hodnoty 0-255. Pri frekvencii 4Mhz je dĺžka strojového cyklu 1us, preto nemôže bez preddeličky napočítať 625us. Veľkosť potrebnej preddeličky vypočítame ako:

$$\frac{625\mu s}{256\mu s} = 2{,}44$$

Najbližšia vyššia preddelička je 1:4, ktorú použijeme v našom programe. Je možné použiť aj väčšie preddeličky, ale môže dôjsť k odchýlke výslednej frekvencie. Pri použití preddeličky 1:4 potrebujeme, aby sa obsah registra TMR0 inkrementoval:

$$\frac{625}{4} = 156{,}25 \cong 156\ krát$$

Keďže k prerušeniu dôjde pri pretečení obsahu registra, potrebujeme register TMR0 nastaviť na:

$$256 - 156 = 100$$

```
LIST P=16F84, R=DEC
INCLUDE<P16F84.INC>
__CONFIG _PWRTE_ON & _WDT_OFF

#DEFINE  BZUCIAK  PORTB,3

         org      0              ;začiatok programu
         goto     START          ;skok na START

         org      4              ;pri prerušeni dôjde k skoku na adr. 4
         bcf      INTCON,GIE     ;zakaz prerusenie
         bcf      INTCON,T0IF    ;vypni priznak prerusenia

         btfsc    BZUCIAK
         goto     ZAPNI
         goto     VYPNI

ZAPNI    bsf      BZUCIAK
         goto     POVOL
```

```
VYPNI   bcf     BZUCIAK
        goto    POVOL
POVOL   movlw   100
        movwf   TMR0
        movlw   B'10100000'         ;povolenie prerusenia od TMR0
        movwf   INTCON
        retfie                      ;navrat z prerusenia
;************************************
;*** hlavny program
START   bsf     STATUS,RP0          ;nastavenie vstupov/vystupov
        clrf    TRISA               ;PORTA výstupy/nepouzite
        clrf    TRISB               ;PORTB vyspupy/nepouzite
        movlw   B‘10000001‘         ;nastavenie preddelicky 1:4
        movwf   OPTION_REG
        bcf     STATUS,RP0

        movlw   100                 ;nastavenie TMR0 pre f=800Hz
        movwf   TMR0

        molvw   B‘10100000‘         ;zapnutie preruseni pre TMR0
        movwf   INTCON

        nop
        goto    $-1
        end
```

Zoznam príkladov: