MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

Razvoj programske podrške za mikroupravljače Microchip PIC16

Upute za laboratorijske vježbe

Mihael Kukec
Ivan Šumiga
Željko Knok
Nenad Breslauer

Čakovec
2016

Autori:

dr.sc. Mihael Kukec, v. pred.
mr.sc. Ivan Šumiga, v. pred.
mr.sc. Željko Knok, v. pred.
Nenad Breslauer, pred.

Recenzenti:

dr. sc. Oliver Jukić, prof. v.š.
dr. sc. Goran Đambić

Nakladnik:

Međimursko veleučilište u Čakovcu

Za nakladnika:

doc.dr.sc. Nevenka Breslauer, prof.v.š.

ISBN 978-953-8095-03-0

PREDGOVOR

Inženjeri računarstva i elektrotehnike moraju dobro poznavati sklopovlje računala u smislu razumijevanja procesa na razini izvršavanja strojnih naredbi. Kako bi se postigao taj cilj, potrebno je prikupiti mnoga znanja o samom sklopovlju i načinu na koji se razvija programska podrška za to sklopovlje.

Poseban naglasak treba staviti na mikroupravljače jer se oni nameću kao rješenja za izgradnju ugrađenih računalnih sustava za čije je oblikovanje potrebno sinergijsko spajanje znanja iz područja računarstva, elektronike i elektrotehnike.

Ove laboratorijske vježbe sastavni su dio gradiva obveznog predmeta Arhitektura računala na stručnom studiju Računarstvo Međimurskog veleučilištu u Čakovcu te predmeta Građa računala na stručnom studiju Elektrotehnika Sveučilišta Sjever.

Vježbe su po sadržaju i redosljedu izvođenja usklađene sa sadržajem i redoslijedom izvođenja predavanja i auditornih vježbi iz navedenih predmeta. Svrha laboratorijskih vježbi je da se studentima omogući svladavanje osnovnih znanja iz područja sklopovlja računala, s posebnim naglaskom na mikroupravljače.

dr.sc. Mihael Kukec, v. pred.
mr.sc. Ivan Šumiga, v. pred.
mr.sc. Željko Knok, v. pred.
Nenad Breslauer, pred.

UPUTE ZA RAD U LABORATORIJU

Budući da se student prvi puta susreće s radom u laboratoriju za razvoj programske podrške za mikroupravljače ovdje se navode osnovne upute kojih se treba pridržavati kako bi se vježbe mogle uspješno obaviti.

Na laboratorijske vježbe je potrebno, osim uputa i programa iz pripreme, donijeti i pribor za računanje i pisanje. Posebne upute za vježbu, ako ih ima, bit će dane na početku vježbe.

Uspješno obavljene laboratorijske vježbe uvjet su za dobivanje potpisa. Na vježbama se dva ili tri puta tokom semestra obavlja provjera znanja, ovisno o izvedbenom planu predmeta za pojedinu akademsku godinu. Dobiveni bodovi ulaze u ocjenu pismenog ispita.

Materija s laboratorijskih vježbi zastupljena je kako na pismenom, tako i na usmenom dijelu ispita.

SADRŽAJ

Vježba 1. UPOZNAVANJE S PROGRAMSKIM PAKETOM MPLAB 1
Vježba 2. LOGIČKO OBLIKOVANJE PROGRAMA 17
Vježba 3. ČITANJE I PISANJE PORTOVA 24
Vježba 4. PALJENJE I GAŠENJE LED DIODA 26
Vježba 5. ŠETAJUĆE SVJETLO 28
Vježba 6. BROJENJE PRITISKOM NA TIPKU 31
Vježba 7. BINARNI BROJAČ 34
Vježba 8. BROJAČ IMPULSA 37
Vježba 9. POVEZIVANJE LCD ZASLONA 40
PROGRAMSKI PRIMJERI S PREDAVANJA 42
Primjer 1 – Prvi program, aritmetičke operacije 42
Primjer 2 – Petlje za kašnjenje 46
Primjer 3 – Potprogrami, mehanizam poziva potprograma 48
Primjer 4 – Prekidi 52
Primjer 5 – Modul Timer 54
Primjer 6 – Timer pomoću prekida 57
Primjer 7 – Rad s 16 bitnim podacima 58
Primjer 8 – Upravljanje s LCD zaslonom HD44780 60
Primjer 9 – Upravljanje tipkovnicom i LCD zaslonom 65
Dodatak A. LABORATORIJSKA OPREMA 71
Dodatak B. REGISTRI MIKROKONTROLERA 76
Dodatak C. SKUP INSTRUKCIJA MCU PIC16F84 84
Dodatak D. LITERATURA 106

Vježba 1. UPOZNAVANJE S PROGRAMSKIM PAKETOM MPLAB

1.1. Cilj vježbe: *Upoznavanje s programskim paketom MPLAB, njegovim funkcioniranjem i radnom okolinom, te osnovnim elementima.*

1.2. Opis vježbe:

Na ovoj vježbi upoznati ćete se s MPLAB-ovom radnom okolinom i s osnovnim elementima MPLAB-a kao što su:

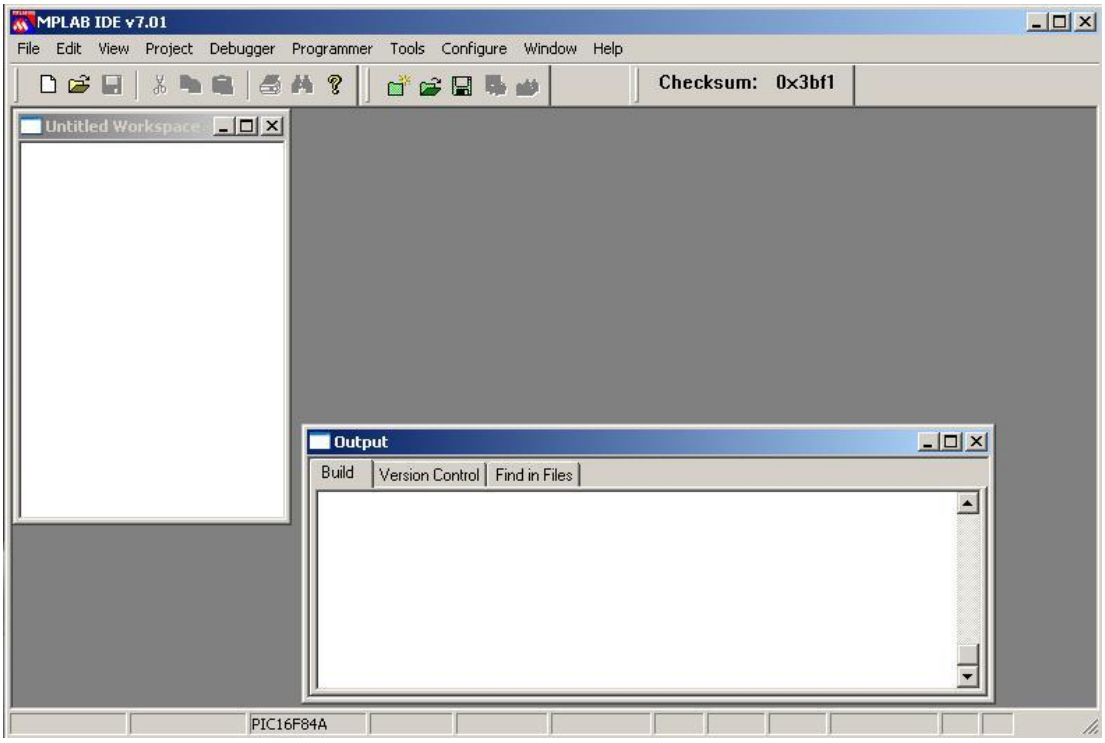
- Odabir razvojnog moda rada;
- Stvaranje projekta;
- Stvaranje datoteka za glavni program;
- Pisanje elementarnog programa u asemblerskom jeziku;
- Prevođenje programa u izvršni kod;
- Pokretanje programa;
- Otvaranje prozora za simulator;
- Otvaranje prozora za varijable čije vrijednosti promatramo (engl. *Watch Window*); Pohranjivanje prozora s varijablama čije vrijednosti promatramo;
- Podešavanje prekidnih točaka u simulatoru (engl. *Break point*).

1.3. Priprema za vježbu:

Proučiti materijale za rad na laboratorijskoj vježbi.

1.4. Rad na vježbi:

Pokretanjem programa "mplab.exe" otvara se sučelje prikazano na slici 1.1.



Sl.1.1. *Sučelje programskog paketa MPLAB.*

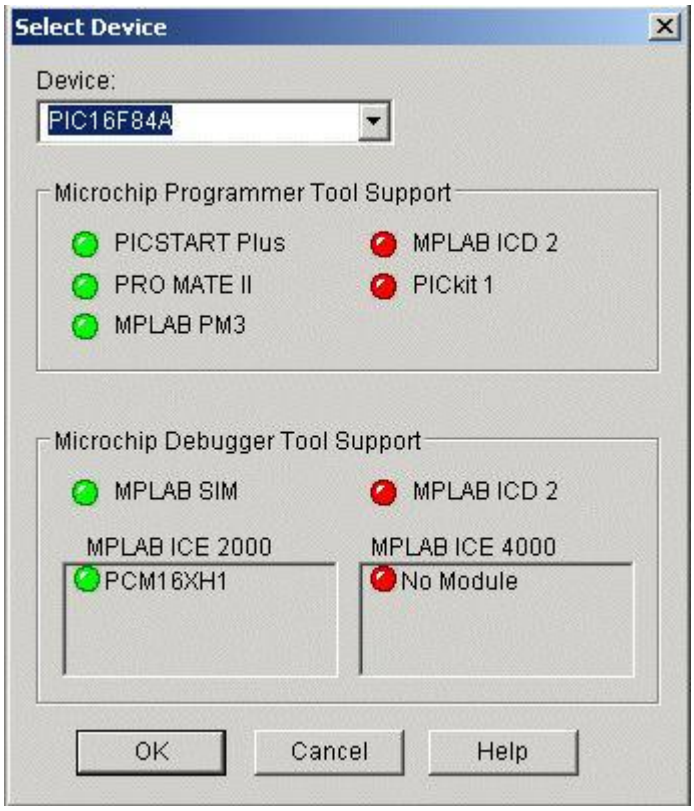
Programski paket MPLAB izgleda kao i većina Windows programa. Sastoji se od izbornika (ispod nazivne trake) s opcijama File, Edit, itd., trake s alatima (područje s ikonama), radnog područja i status trake koja se nalazi na dnu prozora.

Stvaranje projekta sastoji se od sljedećih koraka:

1. Odabir uređaja za koji se piše program
2. Stvaranje projekta
3. Odabira alata
4. Dodavanje datoteka projektu
5. Pisanje kôda
6. Prevođenje kôda u izvršni oblik

ODABIR UREĐAJA

Odabir uređaja za koji se piše program obavezan je zbog toga kako bi MPLAB znao koji će alati biti korišteni za izvršavanje napisanog programa. Pritiskom na *Configure->Select Device* otvara se prozor kao na slici 1.2



Sl.1.2. *Odabir razvojnog moda rada.*

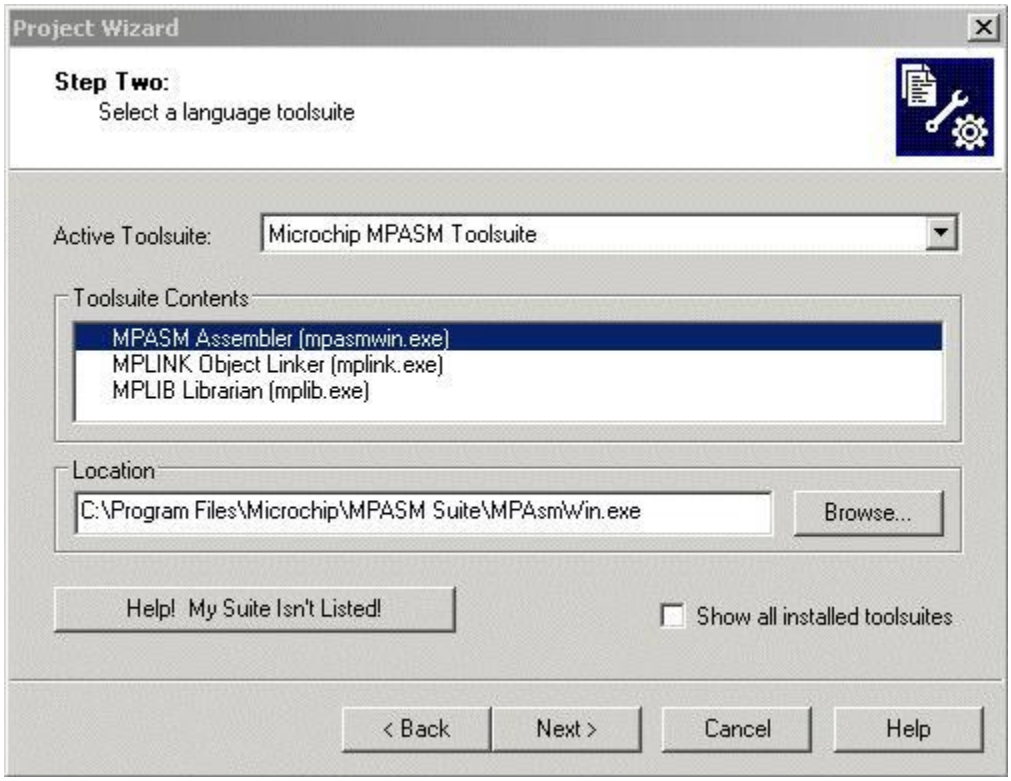
Na vježbama će se koristiti mikrokontroler PIC16F84A pa je pod opcijom *Device* potrebno je odabrati "PIC16F84A".

STVARANJE PROJEKTA

Sljedeći korak je stvaranje projekta pomoću Čarobnjaka. Projektom se organiziraju datoteke za prevođenje u izvršni oblik i povezivanje. Za pokretanje Čarobnjaka potrebno je odabrati *Project->Project Wizard*. U pozdravnom prozoru pritisnuti tipku *Next>* za sljedeći prozor koji omogućava odabir uređaja. Kako smo mi već odabrali uređaj ovdje nije potrebno išta mijenjati, no ipak, ako pod *Device* u prozoru *Project Wizard - Step One* nije odabrano "PIC16F84A", odabrati navedenu opciju. Pritisnuti tipku *Next>* za sljedeći prozor. Otvara se prozor *Project Wizard – Step two* kako je prikazano na slici 1.3.

ODABIR ALATA

U ovom koraku odabiru se jezični alati koji se koriste za prevođenje kôda u izvršni oblik. Prozor. Prozor u kojem se odabiru jezični alati prikazan je na slici 1.3.



Sl.1.3. *Odabir jezičnih alata projekta*

Odabrati “Microchip MPASM Toolsuite” kao *Active Toolsuite*. U popisu *Toolsuite Contents* vidljivi su "MPASM" i "MPLINK". Pritiskom na svaku od tih opcija u *Location* može se vidjeti njihova staza. U slučaju kada je MPLAB IDE instaliran u podrazumijevani direktorij staze do datoteka su sljedeće:

MPASM assembler:
C:\Program Files\Microchip\MPASM Suite\MPAsmWin.exe

MPLINK linker:
C:\Program Files\Microchip\MPASM Suite\MPLink.exe

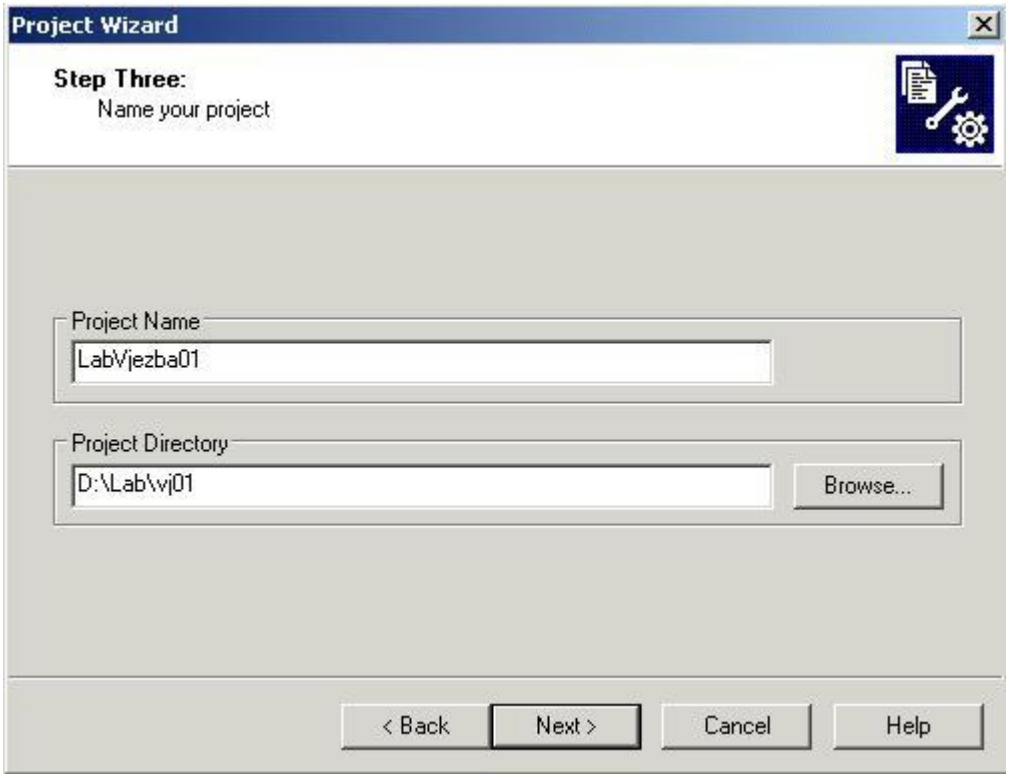
MPLIB librarian:
C:\Program Files\Microchip\MPASM Suite\MPLIB.EXE

U slučaju kada staze do datoteka nisu točne ispravite ih pritiskom na tipku *Browse* odabirom datoteka.

Kada ste završili pritisnite tipku *Next>* za prelazak u sljedeći prozor.

IMENOVANJE PROJEKTA

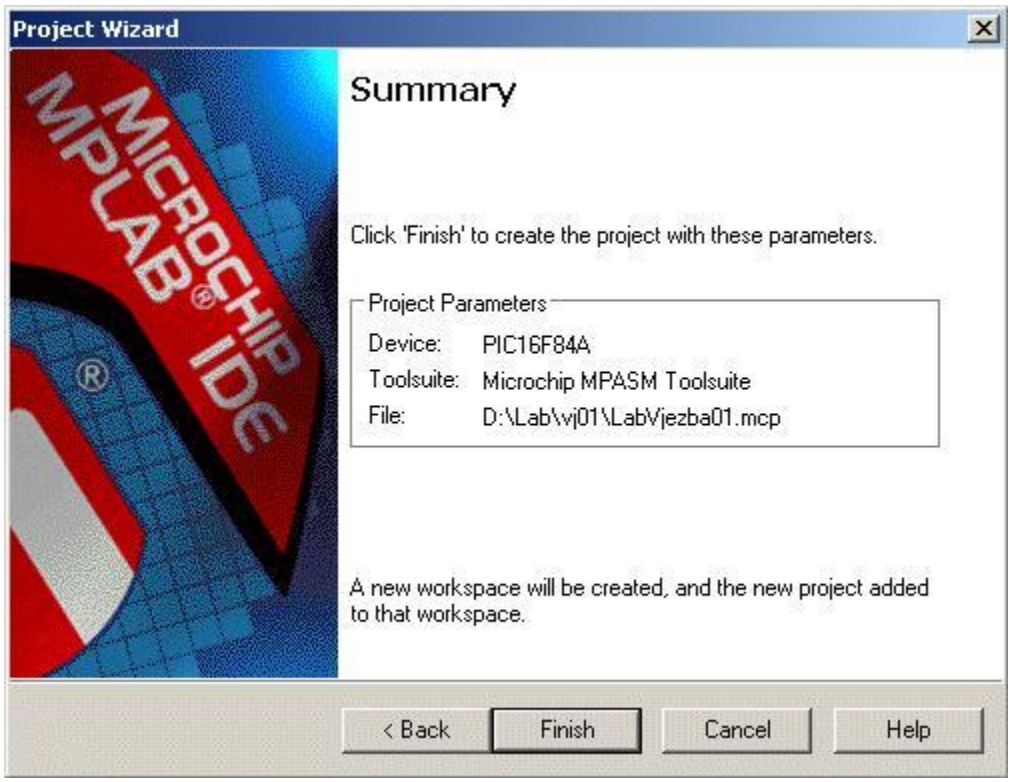
U trećem koraku stvaranja projekta otvara se prozor kao što je prikazano na slici 1.4. Projektu se daje ime i položaj na čvrstom disku. Ime projekta upisuje se u polje *Project Name* a staza na čvrstom disku postavlja se pritiskom na tipku *Browse*.



Sl.1.4. *Imenovanje projekta*

Kada ste završili pritisnite tipku *Next>* za prelazak u sljedeći prozor.

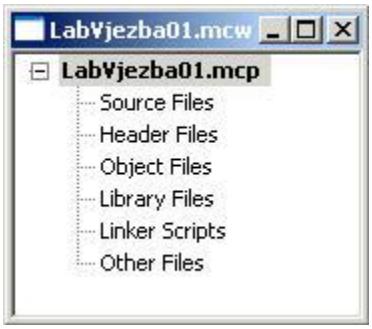
Četvrti korak (Project Wizard – Step Four) omogućava dodavanje datoteka projektu. Datoteke će biti kasnije dodane projektu pa u prozoru Project Wizard – Step Four nije potrebno išta mijenjati. Pritisnuti tipku *Next>* za prelazak u sljedeći prozor koji je prikazan na slici 1.5



Sl. 1.5 *Završetak stvaranja projekta*

U prozoru prikazanom na slici 1.5 pregledati da li su ispisane opcije ispravne i pritisnuti tipku *Finish* za završetak stvaranja projekta.

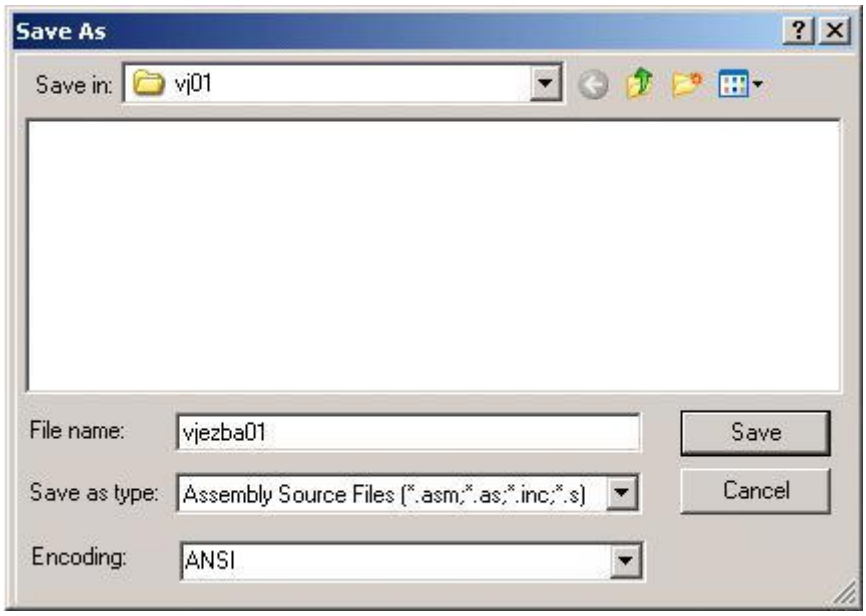
Ovim je otvoren novi prazan projekt. Na radnom površini pojavljuje se prozor projekta kao što je prikazano na slici 1.6. Vidljivo je da je projekt prazan, tj. ne sadrža ikakve datoteke.



Sl. 1.6 *Novi projekt*

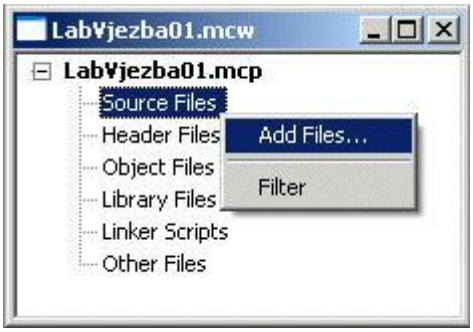
DODAVANJE DATOTEKA PROJEKTU

Kako bi se datoteka s asemblerskim kôdom mogla dodati projektu, treba je stvoriti i pohraniti na čvrsti disk računala. Datoteka se stvara opcijom *File->New* kojom se otvara prazan prozor uređivača teksta s nazivom "Untitled". U ovom trenutku nije potrebo upisivati asemblerski kôd nego je dovoljno pohraniti datoteku pomoću *File->Save* As na čvrsti disk računala (bez obzira što će datoteka biti prazna). Odabirom opcije *File-Save As* otvara se prozor za pohranjivanje datoteke kako je pokazano na slici 1.7. Direktorij u koji se sprema datoteka mora biti direktorij projekta. U gornjem primjeru na slici 1.4 direktorij projekta je "D:\LAB\vj01" pa je pod opcijom *Save in* prozora na slici 1.7 potrebno odabrati taj direktorij. U polje *File name* upisuje se ime datoteke.



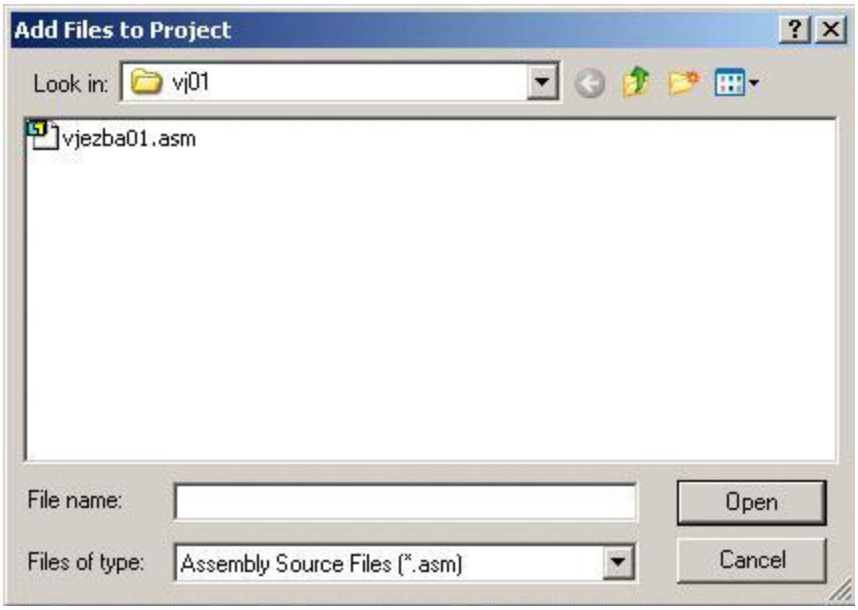
Sl. 1.7 *Pohranjivanje datoteke*

Kada je ime datoteke upisano i direktorij ispravno odabran pritisnuti tipku *Save* za pohranjivanje datoteke. Ovim je stvorena prazna datoteka u koju će se upisivati asemblerski kôd. U nazivnoj traci uređivača asemblerskog kôda sada se nalazi puna staza datoteke. Kako bi mogli raditi s novostvorenom datotekom potrebno je razvojnoj okolini (MPLAB-u) reći da je upravo stvorena datoteka dio našeg projekta.



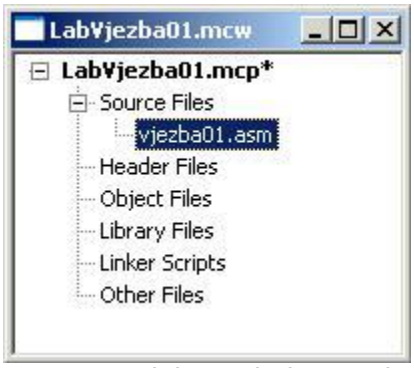
Sl. 1.8 *Dodavanje datoteke projektu*

Novostvorena datoteka dodaje se projektu pritiskom na *Source Files* desnom tipkom miša u prozoru projekta kako je pokazano na slici 1.8. U lebdećem izborniku koji se otvara pritisnuti na opciju *Add Files*.



Sl. 1.9 *Odabir datoteke koja se dodaje projektu*

U prozoru za odabir datoteke koji se otvara (slika 1.9) odabrati novostvorenu datoteku. U ovom primjeru je to datoteka "D:\LAB\vj01\vjezba01.asm[1]". Kada je datoteka odabrana pritisnuti tipku *Open* za dodavanje datoteke projektu.



Sl. 1.10 *Prozor projekta s dodanom datotekom*

Slika 1.10 pokazuje prozor projekta "LabVjezba01" kojem je dodana datoteka "vjezba01.asm". Stvorene promjene pohraniti pomoću opcije *File->Save Workspace*.

[1] Datoteke s asemblerskim kôdom imaju ekstenziju "asm"

PISANJE KÔDA (PISANJE PROGRAMA)

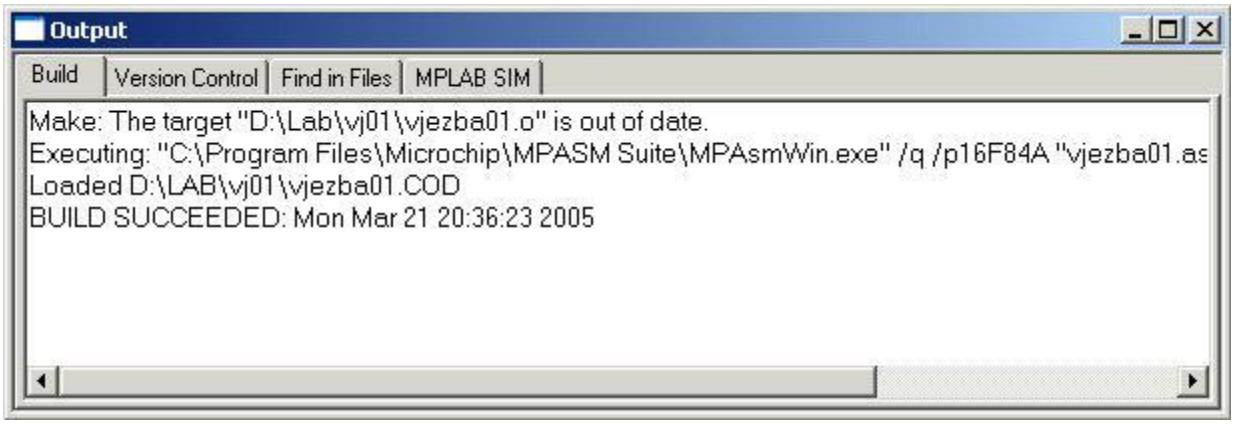
Tek nakon što obavimo sve prije navedene radnje možemo početi pisati program. Napisati ćemo jednostavan program koji je dan u nastavku.

```
  ================================================
  Visoka Elektrotehnicka Skola s pravom javnosti
  Kolegij: Gradja Racunala, laboratorijske vjezbe

  Program početno pali diode na RB1 i RB3 a gasi na RB2
  Crvena dioda svijetli uvijek, dok se stanja zelene i žute
  diode izmjenjuju ovisno o tome je li tipkalo pritisnuto ili nije.
  Početno stanje je da zelena dioda ne svijetli a žuta svijetli.
  Pritiskom na tipkalo zelena dioda svijetli a žuta je ugašena.
  ================================================

  ================================================
  Postavljnje MCU i kofiguracijska rijec
  ================================================
              PROCESSOR 16F84A
              #include "p16f84a.inc"
              ERRORLEVEL -224
              __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF

              org 0x0        ; pocetak na pocetak :)

  ================================================
  Postavljnje PORTAB i PORTA
  ================================================
              clrf    PORTB          ; postavi 0x0 na PORTB
              bsf     STATUS, RP0    ; odabir Bank 1
              movlw   B'11110001'    ; RB0 je ulaz, ostali su izlaz
              movwf   TRISB ^ 0x80   ; postavi 0xF1 u TRISB
              movlw   B'00011111'    ; 0x1F u W
              movwf   TRISA ^ 0x80   ; svi pinovi na PORTA ulazni

  ================================================
  Pali LED diode na RB1 i RB3, pocetno stanje
  ================================================
              bcf     STATUS, RP0    ; odabir Bank 0
              bcf     PORTB, 1       ; upali diodu na RB1 (CRVENA svijetli uvijek)
              bsf     PORTB,  2      ; ugasi diodu na RB2 (ZELENA ne svijetli)
              bcf     PORTB,  3      ; upali diodu na RB3 (ZUTA svijetli)

  ================================================
  Ukljuci pull-upove na PORTB
  ================================================
              bsf     STATUS, RP0            ; odabir Bank 1
              bcf     OPTION REG, NOT RBPU   ; omoguci interni pull-up

; ================================================
; Kada je tipkalo pritisnuto   : pali RB2, gasi RB3
  Kada tipkalo NIJE pritisnuto : pali RB3, gasi RB2
  ================================================

              bcf     STATUS, RP0    ; odabir Bank 0

petlja        nop
              btfsc   PORTB, 0       ; test tipkala
              goto    p_zut          ; idi na p_zut
              nop
```

```
p_zel   nop
                bsf             PORTB, 3                ; gasi ZUTU
                bcf             PORTB, 2                ; pali ZELENU
                goto            petlja                  ; idi na petlja

p_zut           nop
                bsf             PORTB, 2                ; ugasi ZELENU
                bcf             PORTB, 3                ; upali ZUTU
                goto            petlja                  ; idi na petlja

                end
```

Sl. 1.11 *Kôd programa*

Nakon što se program upiše u prozoru uređivača kôda, naredbom *Project->Build All* asemblerski kôd programa prevodi se u izvršni oblik. Rezultati prevođenja ispisuju se u *Output* prozoru pod jahačem *Build* kao što je pokazano na slici 1.12.



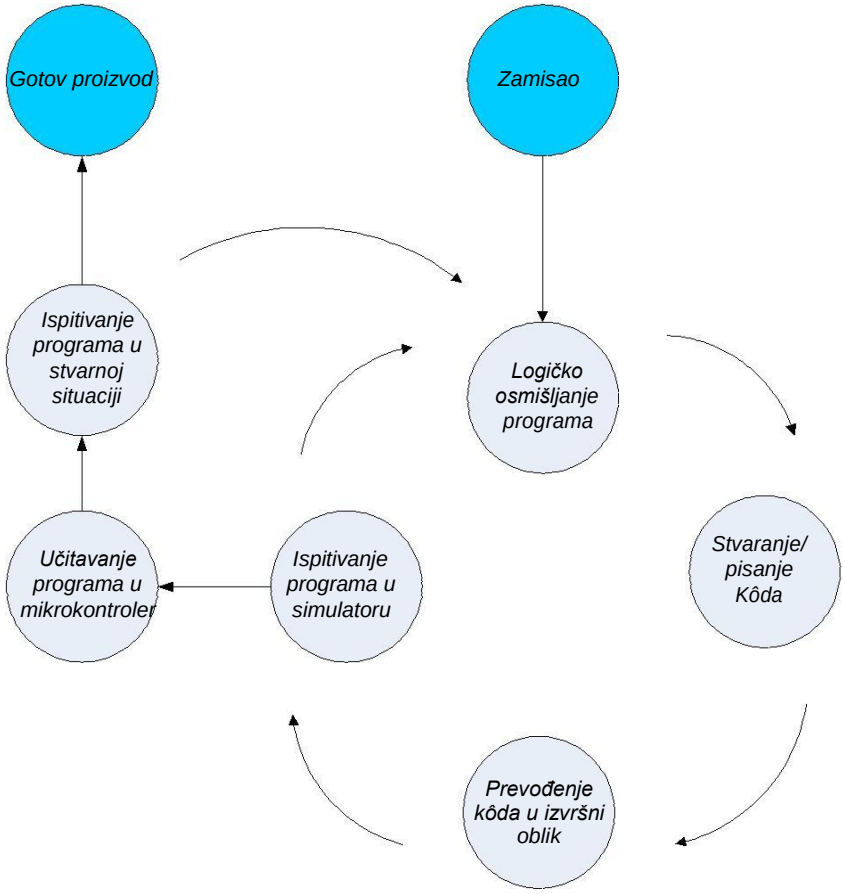
Sl.1.12. *Prozor s porukom o prijevodu asemblerskog programa.*

Posljednja rečenica u prozoru je najvažnija jer pokazuje je li prevađanje bilo uspješno ili nije. "BUILD SUCCEEDED" je poruka koja govori o uspješnosti prevađanja, te nam govori da nije bilo pogrešaka.

U slučaju kada se pojavi pogreška, dvostruki pritisak na poruku o pogrješci u "Build" prozoru automatski vas prebacuje u asemblerski program u redak za koji prevodioc misli da sadržava grješku.

MPSIM SIMULATOR

Proces pisanja programa često se naziva razvojnim ciklusom. Rijetko se sve faze razvoja od zamisli do implementacije izvrše bez grješke. Češće, kôd se piše, ispituje i mijenja kako bi se dobila aplikacija koja zadovoljava početne zahtijeve. Slika 1.13 pokazuje razvojni ciklus aplikacije od zamisli do gotovog proizvoda.



Sl. 1.13. *Razvojni ciklus programa*

Simulator je dio MPLAB-ove radne okoline koji omogućava bolji uvid u rad mikrokontrolera. Kroz simulator možemo promatrati trenutne vrijednosti varijabli, vrijednosti u registrima i status na pinovima portova. U slučaju kada je program jednostavan (kao u danom primjeru), simulator nije od prevelike važnosti no ispravljanje logičkih progrješaka bez simulatora bilo bi znatno teže ili gotovo nemoguće. Simulator nam može biti od velike pomoći sa složenijim programima koji uključuju mjerače vremena, različita stanja gdje se nešto događa i drugi slični zadaci (osobito s matematičkim operacijama). Simulacija, kao što joj i samo ime govori označava "simulaciju rada mikrokontrolera". Kao što mikrokontroler obavlja instrukcije jednu po jednu tako je i simulator zamišljen – programer se kreće kroz program korak-po-korak (redak-po-

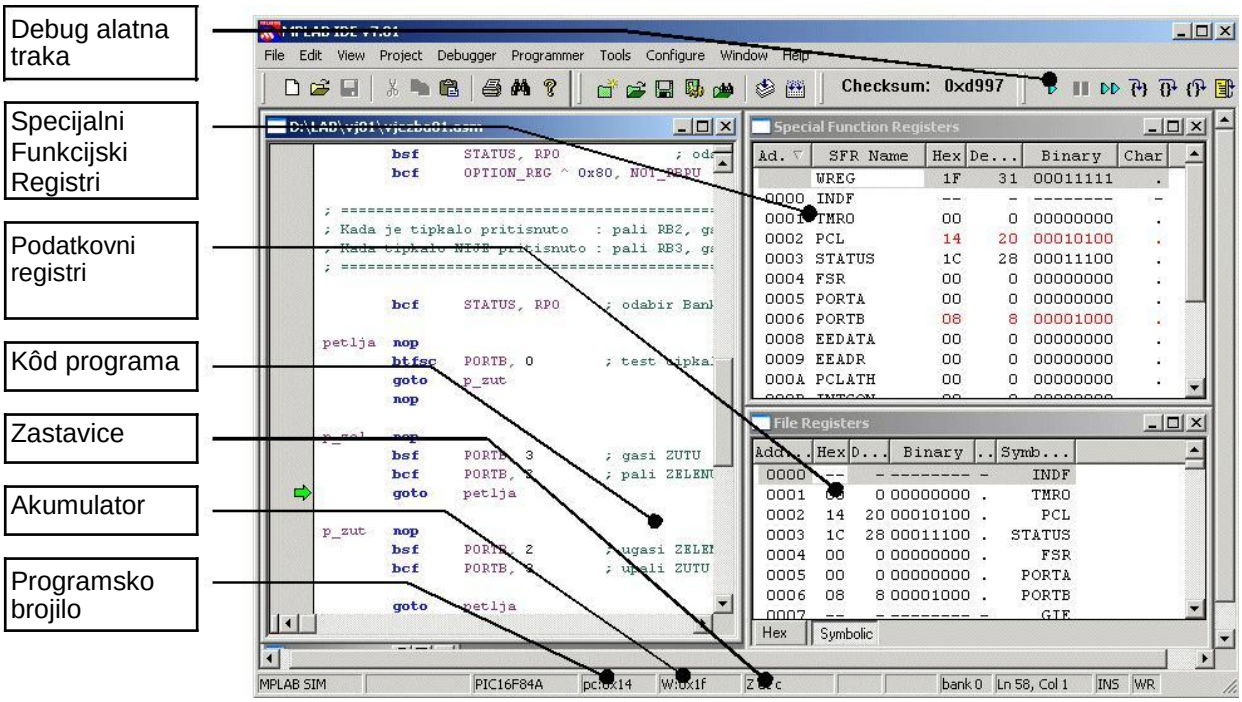
redak) i promatra što se događa s podacima unutar mikrokontrolera. Kad se završi s pisanjem programa potrebno je pomoću simulatora provjeriti da li program obavlja zadani zadatak. Potrebno je provjeriti da li program radi upravo ono što je određeno zahtijevima tj. zadatkom. Tek kada se pomoću simulatora uvjerimo da program radi upravo ono što je određeno u zadatku tada se prelazi na ispitivanje programa u stvarnoj situaciji.

Kako bi se mogao koristiti simulator, razvojnoj okolini (MPLAB-u) je potrebno reći koji simulator mislimo koristiti. Na vježbama će se koristiti simulator ugrađen u razvojnu okolinu MPLAB. Odabir simulatora vrši se u izborniku *Debugger->Select Tool.* U izborniku koji se otvara potrebno je odabrati *MPLAB SIM (Debugger->Select Tool->MPLAB SIM)*.

Nakon prevođenja kôda u izvršni oblik, prvu stvar koju koju je potrebno učiniti, kao u pravoj situaciji, je resetiranje mikrokontrolera opcijom izbornika *Debugger->Reset->Processor Reset* ili pritiskom na tipku F6. Resetiranje mikrokontrolera rezultira postavljanjem postavljanje programskog brojila na nulu što se može vidjeti u statusnoj liniji (pc:0x00). U uređivaču kôda, zelena strelica koja pokazuje instrukciju koja će se izvršiti, postavlja se na prvu instrukciju programa.

Jedna od glavnih karakteristika simulatora je mogućnost pregleda vrijednosti registara mikrokontrolera tijekom izvođenja programa. Ti registri se još nazivaju specijalni funkcijski registri, ili SFR registri. Prozor sa SFR registrima otvara se opcijom *View->Special Function Registers*. Osim SFR registara, korisno je imati uvid u podatkovne registre. Prozor sa podatkovnim registrima može se otvoriti odabirom *View->File Registers*.

U slučaju kada se u programu koriste varijable, moguće je promatrati promjenu njihovih vrijednosti tijekom izvođenja programa pomoću prozora koji se otvara opcijom *View->Watch*.



Sl. 1.14. *Ispitivanje ispravnosti rada programa*

Slika 1.14 pokazuje proces ispitivanje programa koji se izvodi. Ispitivanje programa je zahtijevan proces jer iziskuje dobro poznavanje rada mikrokontrolera. Program se ispituje izvođenjem instrukciju po instrukciju opcijom *Debugger->Step Into* (ili tipka F7). Po izvođenju svake instrukcije u prozorima specijalnih funkcijskih registara, podatkovnih registara i kôda programa provjerava se da li se mikrokontroler nalazi u željenom stanju, što znači da vrijednosti svih registara moraju biti jednake onima koje programer očekuje nakon izvršavanja instrukcije. Analiza programa koja se provodi na ovakav način može otkriti pogrješke u pisanju samog kôda koje su zadovoljile sintaksnu analizu no logički su krive te daju pogrješne rezultate, no osim toga, ovakvom analizom otkrivaju se i grješke u logičkom osmišljanju programa, što znači da slijed izvođenja instrukcija nije dobro osmišljen i da neće dati željeni rezultat.

Kao što je rečeno u gornjem dijelu teksta, naredba simulatora koja omogućava pokretanje programa korak po korak je *Debugger->Step Into*. Tu istu naredbu mogli bi aktivirati pomoću tipkovnice s F7 tipkom (drugim riječima svaka značajna naredba ima svoju odgovarajuću tipku na tipkovnici). Koristeći tipku F7, program se izvršava korak-po-korak. U prozoru SFR registara možemo promatrati kako W registar (akumulator) dobiva vrijednost 0xF1 nakon izvođenja treće instrukcije kôda sa slike 1.11.

ALATNE TRAKE

Alatne trake MPLAB IDE su:

1. Standard (slika 1.15)
2. Project Manager (slika 1.16)
3. Debug

Sl. 1.15 *Alatna traka standard*

Alati alatne trake Standard su (redom s lijeva na desno)

1. New File Otvaranje novog prozora uređivača kôda
2. Open File Otvaranje datoteke u uređivaču kôda
3. Save File Pohranjivanje kôda trenutno aktivnog prozora uređivača
4. Cut Izreži
5. Copy Kopiraj
6. Paste Zalijepi
7. Print File Ispiši datoteku
8. Find Traži
9. Help Pomoć

Sl. 1.16 *Alatna traka Project Manager*

Alati alatne trake Project Manager su (redom s lijeva na desno):

1. New project — Stvaranje novog projekta
2. Open project — Otvaranje prije stvorenog projekta
3. Save Workspace — Pohranjivanje radne površine
4. Build options — Opcije prevođenja programa u izvršni oblik
5. Find In Project Files — Traži po datotekama projekta
6. Make — Prevođenje u izvršni oblik
7. Build All — Prevođenje u izvršni oblik

Sl. 1.17 *Alatna traka Debug*

U alatnoj traci Debug nalaze se alati za ispitivanje rada programa. Alati (redom s lijeva na desno) su:

1. Run — Pokreće program. Prije pokretanja program je potrebno prevesti u izvršni oblik
2. Halt — Zaustavlja izvođenje programa
3. Animate — Izvodi program automatski instrukciju po instrukciju brzinom koja omogućava praćenje izvođenja programa
4. Step Into — Izvodi jednu instrukciju sljedeći pozive potprograma
5. Step Over — Izvodi jednu instrukciju preskačući pozive potprograma. Cijeli potprogram tretira kao jednu instrukciju.
6. Reset — Postavlja programsko brojilo na prvu instrukciju tj. vraća izvođenje programa na početak

OSTALE OPCIJE MPLAB RAZVOJNE OKOLINE

Disasemblirani ispis programa (Izbornik: *View->Disassembly Listing*)

Disasemblirani ispis pomaže pri ispitivanju ispravnosti rada programa te otklanjanju pogrješaka. U disasmebliranom ispisu sva simbolička imena u kôdu i labele zamijenjeni su njihovim brojčanim ekvivalentima i adresama. Uz disasemblirani kôd ispisani su i izvornik komentari koji olakšavaju praćenje kôda.

Pregled programske memorije (Izbornik: *View->Program Memory*)

Pregled programske memorije daje uvid u kôd programa na način na koji ga vidi mikrokontroler. Moguć je uvid u izvšni oblik programa u heksadecimalnom zapisu (*Opcode Hex*), strojni kôd popraćen disasembliranim kôdom (*Machine*) i simbolički kôd (*Symbolic*)

Pregled podatkovnih registara mikrokontrolera (Izbornik: *View->File registers*)

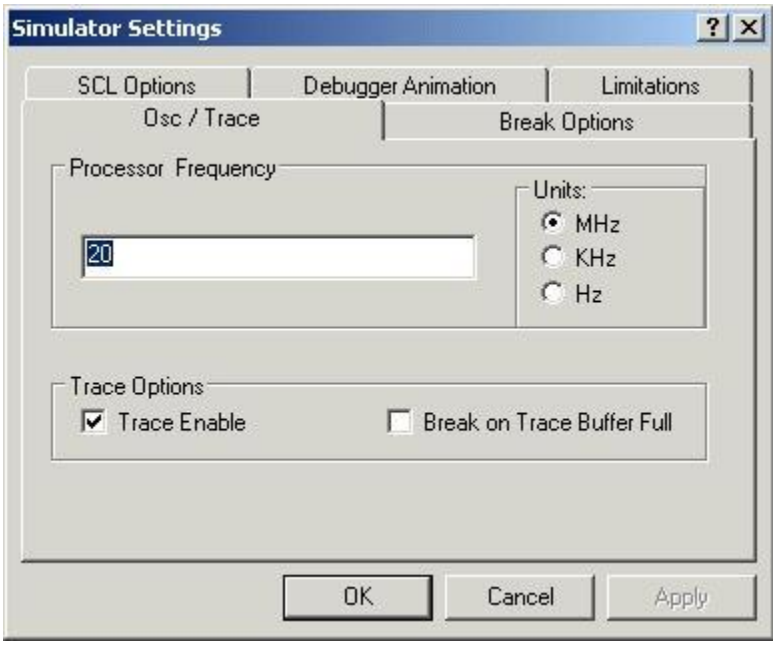
Podatkovni registri su radna memorija mikrokontrolera, mjesto gdje su pohranjene sve varijable programa. Pregled podatkovnih registara je nužan za ispitivanje ispravnosti rada programa i pronalaženje pogrješaka. Pregled je moguć u heksadecimalnom zapisu (*Hex*) i simbolički (*Symbolic*).

Pregled specijalnih funkcijskih registara (Izbornik: *View->Special Function Registers*)

Specijalni funkcijski registri su jednako važni kao i podatkovni registri. Vrijednosti u njima određuju ponašanje mikrokontrolera te uvelike utječu na izvođenje programa. Važno je osigurati da vrijednosti u njima u svakom trenutku budu ispravne.

Podešavanje radnog takta mikrokontrolera

Radni takt mikrokontrolera podešava se u prozoru koji se otvara pomoću izbornika *Debugger->Settings*. Prozor koji se otvara nalazi se na slici 1.18.

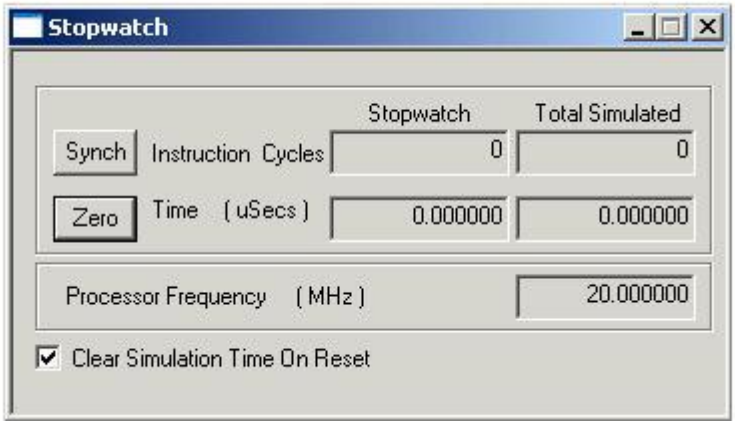


Sl. 1.18 *Podešavanje radnog takta mikrokontrolera*

Željeni radni takt potrebno je upisati u polje *Processor Frequency* te odabrati jedinicu pod *Units*. Na slici 1.18 radni takt procesora je 20MHz.

Mjerenje vremena

Mnoge primjene mikrokontrolera uključuju vremenski generirane signale. Kako bi se uvjerili da je određeni dio kôda za izvođenje potrošio neku količinu vremena koristimo štopericu (engl. stopwatch) ugrađenu u MPLAB IDE. Štoperica se pokreće odabirom opcije *Debugger>StopWatch*. Prozor koji se otvara odabirom ove opcije pokrazan je na slici 1.19.



Sl. 1.19 *Mjerenje vremena*

Vježba 2. LOGIČKO OBLIKOVANJE PROGRAMA

2.1. Cilj vježbe: *Upoznavanje s logičkim oblikovanjem programa, uvođenje algoritamske razrade zadatka kroz blok dijagrame.*

2.2. Opis vježbe:

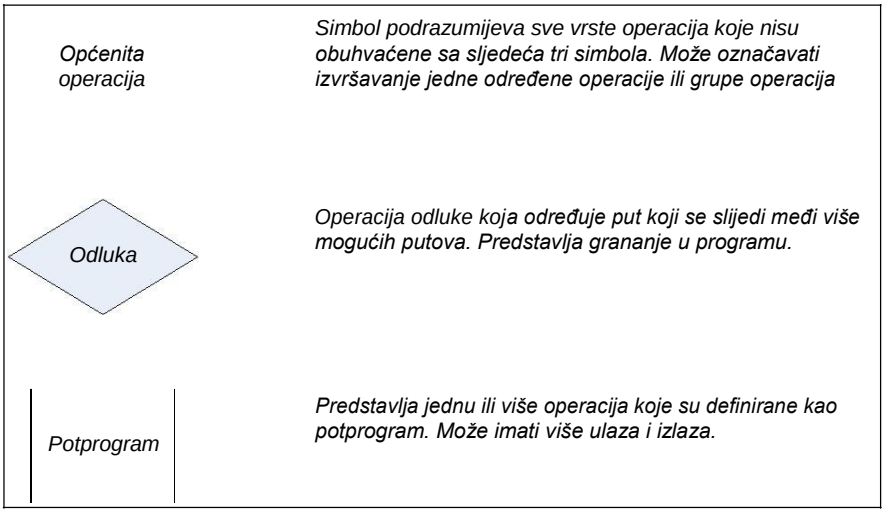
Vježba upoznaje studente s algoritamskim načinom razmišljanja. Pokazuje dekompoziciju zadanog problema na manje problemske cjeline, identificiranje potrebnih alata i postupaka kako bi se pojedina problemska cjelina razriješila te određivanje ispravnog slijeda primjene alata i postupaka kako bi se dobio zadovoljavajući rezultat te kompozicija dobivenih dijelova u cjelovito rješenje problema. Za prikazivanje algoritama koristiti će se blok dijagrami.

1.3. Priprema za vježbu:

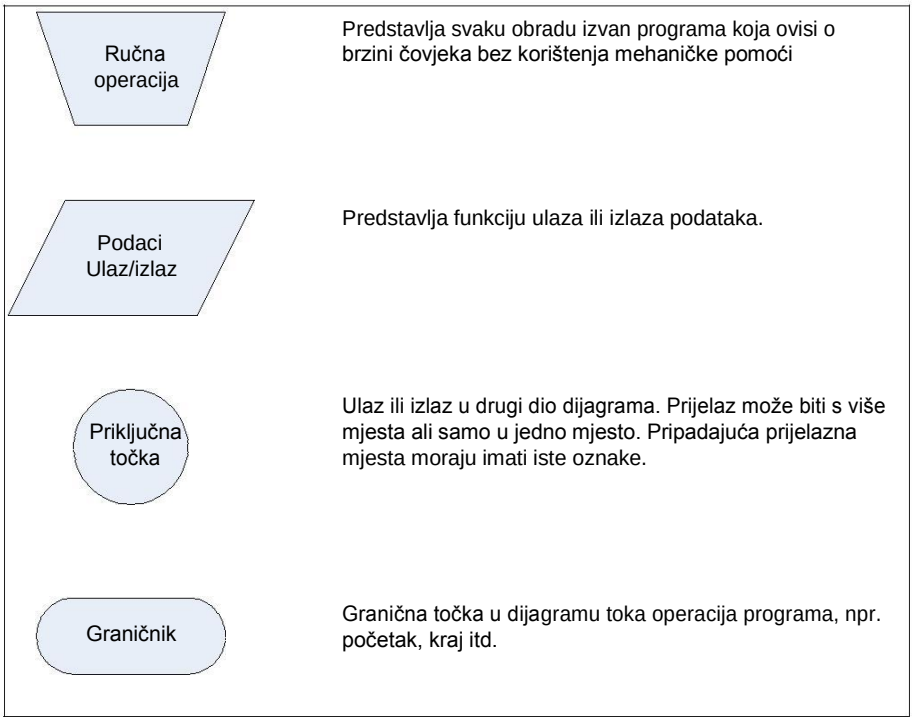
Proučiti materijale za rad na laboratorijskoj vježbi.

1.4. Rad na vježbi:

SIMBOLI ZA DIJAGRAME ODVIJANJA OPERACIJA PROGRAMA



Sl. 2.1 *Simboli dijagrama odvijanja operacija programa*



Sl. 2.2 *Simboli dijagrama odvijanja operacija programa*

PRIMJER 1 : Postupak pisanja programa u razvojnoj okolini MPLAB.

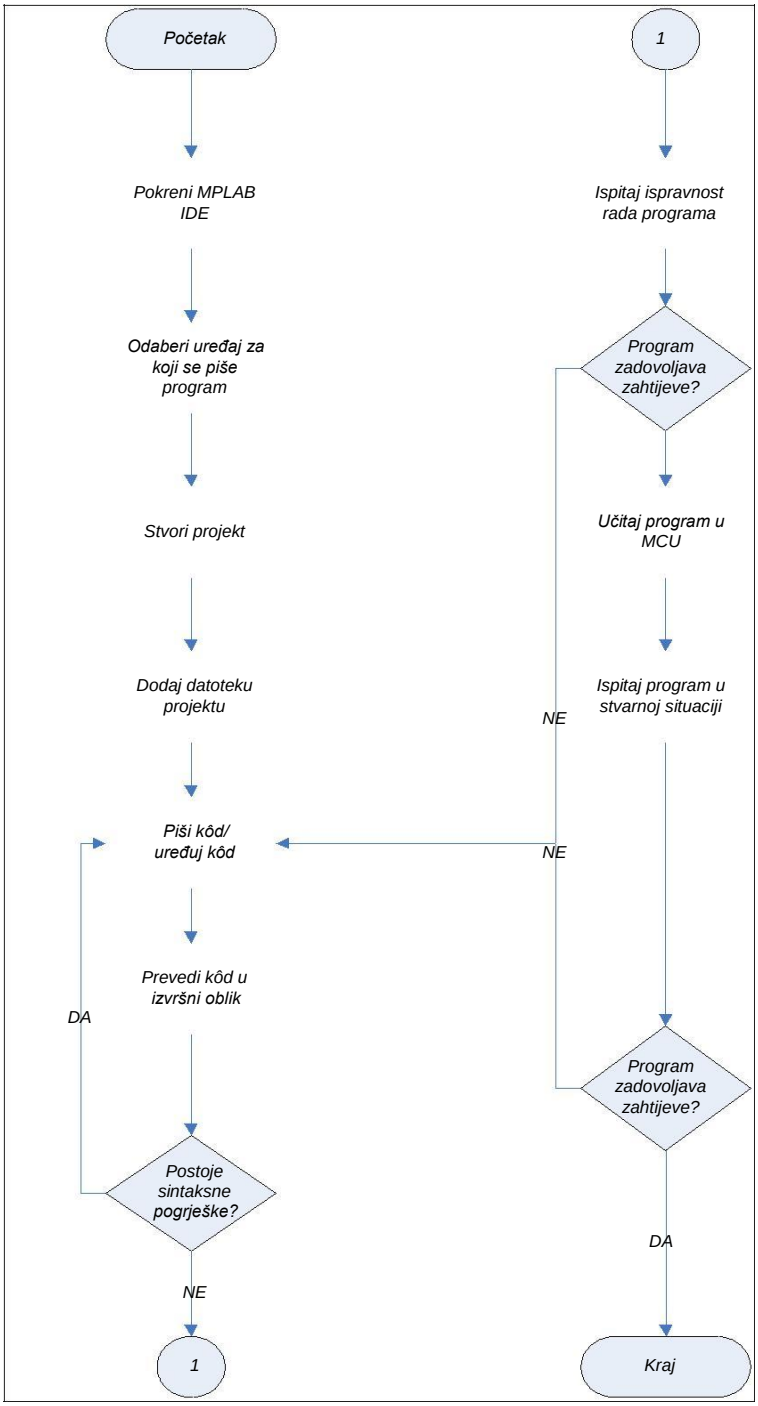
Rastavljanje problema na cjeline (podijeli pa vladaj):

1. Pokretanje razvojne okoline MPLAB
2. Odabir uređaja za koji se piše program
3. Stvaranje projekta
4. Dodavanje datoteka projektu
5. Pisanje kôda
6. Prevođenje kôda u izvršni oblik
7. Otklanjanje pogrješaka u programu i ispitivanje ispravnosti rada
8. Učitavanje programa u mikrokontroler
9. Ispitivanje ispravnosti rada

Identificiranje alata i postupaka

Problem	Alat/sredstvo	Postupak
Pokretanje razvojne okoline MPLAB	GUI sučelje MS Windows OS-a	Pritisak lijevom tipkom miša na ikonu MPLAB IDE
Odabir uređaja za koji se piše program	MPLAB IDE: *Configure->Select Device* izbornik	Pritisak lijevom tipkom miša na opciju izbornika
Stvaranje projekta	MPLAB IDE: Čarobnjak za stvaranje projekta	Pokrenuti Čarobnjak za stvaranje projekta. Postupati po uputama Čarobnjaka.
Dodavanje datoteke projektu	MPLAB IDE: Opcije izbornika *File->New* i *File->Save As*	Pritisak lijevom tipkom miša na opcije izbornika te upisivanje potrebnih podataka
Pisanje kôda	MPLAB IDE: uređivač teksta	Pisanje kôda unošenjem s tipkovnice
Prevođenje u izvršni oblik	MPLAB IDE: prevodilac i povezivač	Pokretanje opcije *Project->Build All*
Otklanjanje sintaksnih pogrješaka	MPLAB IDE: Izlaz procesa prevođenja i uređivač kôda.	Uređivačem kôda ispraviti pogrješke prijavljene u izlazu procesa prevođenja
Ispitivanje ispravnosti rada programa	MPLAB IDE: Debugger	Pregledom programa instrukciju po instrukciju utvrditi ispravnost rada
Učitavanje programa u MCU	Program za učitavanje izvršnog kôda u MCU	Učitati izvršni kôd u MCU
Ispitivanje ispravnosti rada u stvarnoj situaciji	Stvarna okolina u kojoj će MCU raditi	Ispitivanjem rada MCU unutar sustava odrediti ispravnost rada

Dijagram toka operacija:



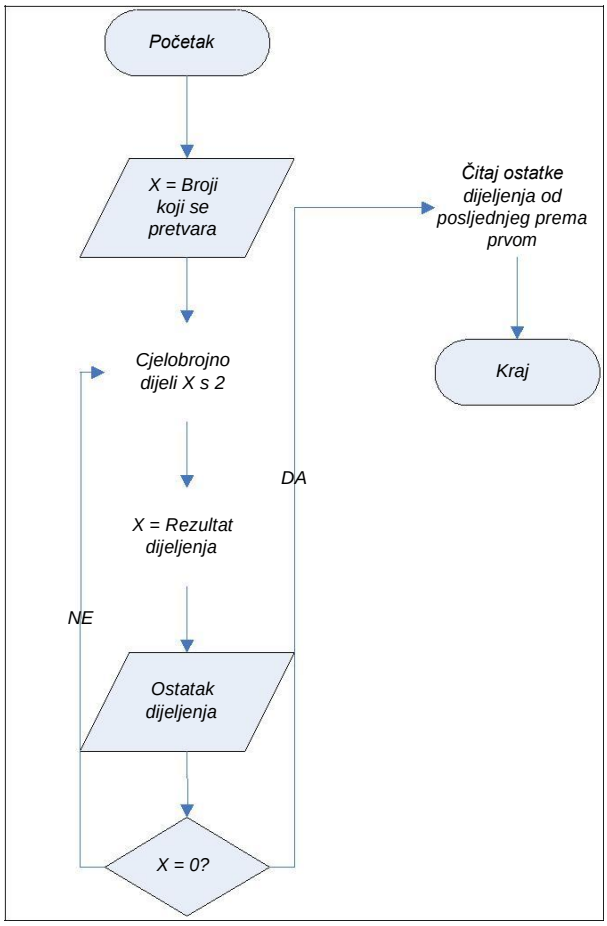
Sl. 2.3 *Dijagram toka pisanja programa u MPLAB IDE*

PRIMJER 2 : Pretvaranje decimalnog broja u binarni

Identificiranje alata i postupaka

Problem	Alat/sredstvo	Postupak
Pretvaranje decimalnog broja u binarni	Dijeljenje	Dijeljenje s 2 dok se ne dobije rezultat 0. Zapisivanje ostataka. Rezultat pretvorbe su zapisani ostaci pročitani redoslijedom obrnutim od zapisivanja.

Dijagram toka operacija:



Sl. 2.4 *Diagram toka pretvorbe dekadskog broja u binarni*

PRIMJER 3 : Logičko oblikovanje programa sa slike 1.11.

Zadatak:

Program početno pali diode na RB1 i RB3 a gasi na RB2. Crvena dioda svijetli uvijek, dok se stanja zelene i žute diode izmjenjuju ovisno o tome je li tipkalo pritisnuto ili nije. Početno stanje je da zelena dioda ne svijetli a žuta svijetli. Pritiskom na tipkalo zelena dioda svijetli a žuta je ugašena. Diode su spojene na sljedeći način:

- Crvena: RB1
- Zelena : RB2
- Žuta : RB3

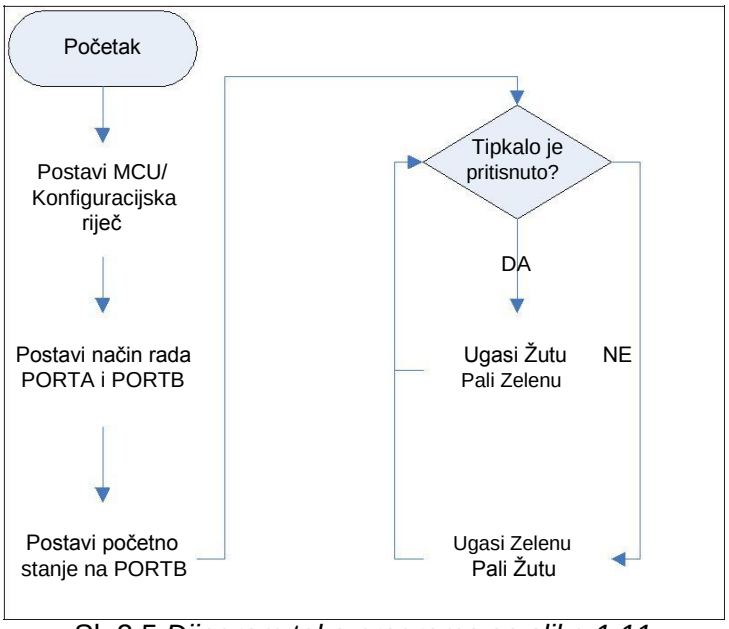
Rastavljanje problema na manje problemske cjeline:

1. Postavljanje MCU (konfiguracijska riječ)
2. Postavljanje načina rada PORTAB i PORTA
3. Postavljanje početnog stanja (Pali diode na RB1 i RB3)
4. Određivanje stanja tipkala (pritisnuto/nije pritisnuto)
5. Promjena stanja na diodama spojenim na RB2 i RB3

Identificiranje alata i postupaka:

Problem	Alat/sredstvo	Postupak
Postavljanje MCU	Konfiguracijska riječ	Konfiguracijskom riječi postaviti MCU
Postavljanje načina rada PORTA i PORTB	TRISA i TRISB registar određuju način rada PORTA i PORTB, respektivno.	Postaviti TRISA i TRISB tako da pinovi portova odgovaraju namjeni prema shemi (RB1, RB2 i RB3 su izlazni, RB0 je ulazni)
Postavljanje početnog stanja	PORTB	RB1 i RB3 postavi na 0, RB2 postavi na 1.
Određivanje stanja tipkala	PORTB, pin 0; **btfsc** instrukcija	Ispitati stanje RB0 pomoću **btfsc** instrukcije
Promjena stanja na dididama spojenim na RB2 i RB3.	PORTB, pin 2 i 3 (RB2 i RB3)	Instrukcijama **bsf** i **bsf** promjeniti stanje na RB2 i RB3

Dijagram toka operacija:



Sl. 2.5 *Dijagram toka programa sa slike 1.11*

Vježba 3. ČITANJE I PISANJE PORTOVA

3.1. Cilj vježbe: *Upoznati se s postupkom programiranja mikrokontrolera PIC16F84, te načinom uporabe njegovih ulaza i izlaza.*

3.2. Opis vježbe:

Bilo što da se radi s mikrokontrolerom koristi se njegove U/I (engl. *Input/Output (I/O)*) portove, bilo za prikupljanje podataka ili za upravljanje nekim drugim uređajima, tj. neki od pinova koriste se kao ulazi, a neki kao izlazi. Microchipov PIC16F84 posjeduje dva U/I porta s ukupno 13 U/I linija [port A (5 pinova) i port B (8 pinova)]. Funkcija pojedinih U/I pinova određuje se

"TrisA" i "TrisB" registrima. Nakon reseta mikrokontrolera svi pinovi su podešeni kao ulazni tj. svi bitovi u tris registru su postavljeni na "1". Ukoliko se želi neki pin na portu proglasiti za izlazni potrebno je u tris registru na tom mjestu izbrisati bit (postaviti na "0"). U memorijskoj mapi U/I registri (PortA i PortB) nalaze se u nultoj banci (engl. *Bank0*) na lokacijama 0x05 i 0x06, a registri smjera (TrisA i TrisB) u prvoj banci (engl. *Bank1*) na lokacijama 0x85 i 0x86.

Odabir banke kojoj se želi pristupiti vrši se bitovima RP1 (šesti bit) i RP0 (peti bit) u STATUS registru. Omogućavanje internog pull-up-a vrši se brisanjem sedmog bita u OPTION_REG registru u banci 1.

3.3. Priprema za vježbu:

Napisati program koji će u sklopu sa slike 3.1. raditi sljedeće:

- postaviti pinove porta A RA0, RA1, RA2 i RA3 kao izlazne, a pin RA4 kao ulazni;
- postaviti pinove porta B RB2, RB3, RB4 i RB5 kao ulazne, a pinove RB0, RB1, RB6 i RB7 kao izlazne;
- provjeravati stanja tipkala na pinovima RB2, RB3, RB4 i RB5, te prenositi ta stanja na pinove RA0, RA1, RA2 i RA3, a komplementirana stanja na pinove RB0, RB1, RB6 i RB7;
- ponavljati cjelokupni postupak.
-

Napomena: Stanja pina RB2 prenose se na RA0 i RB0; stanja pina RB3 prenose se na RA1 i RB1; Stanja pina RB4 prenose se na RA2 i RB6; Stanja pina RB5 prenose se na RA3 i RB7. Potrebno je omogućiti interni pull-up.

3.4. Rad na vježbi:

Pokrenuti program MPLAB i u njemu napraviti projekt pod nazivom "vj3.pjt".
Program iz pripreme spremiti u datoteku "vj3.asm" i uključiti ju u projekt.

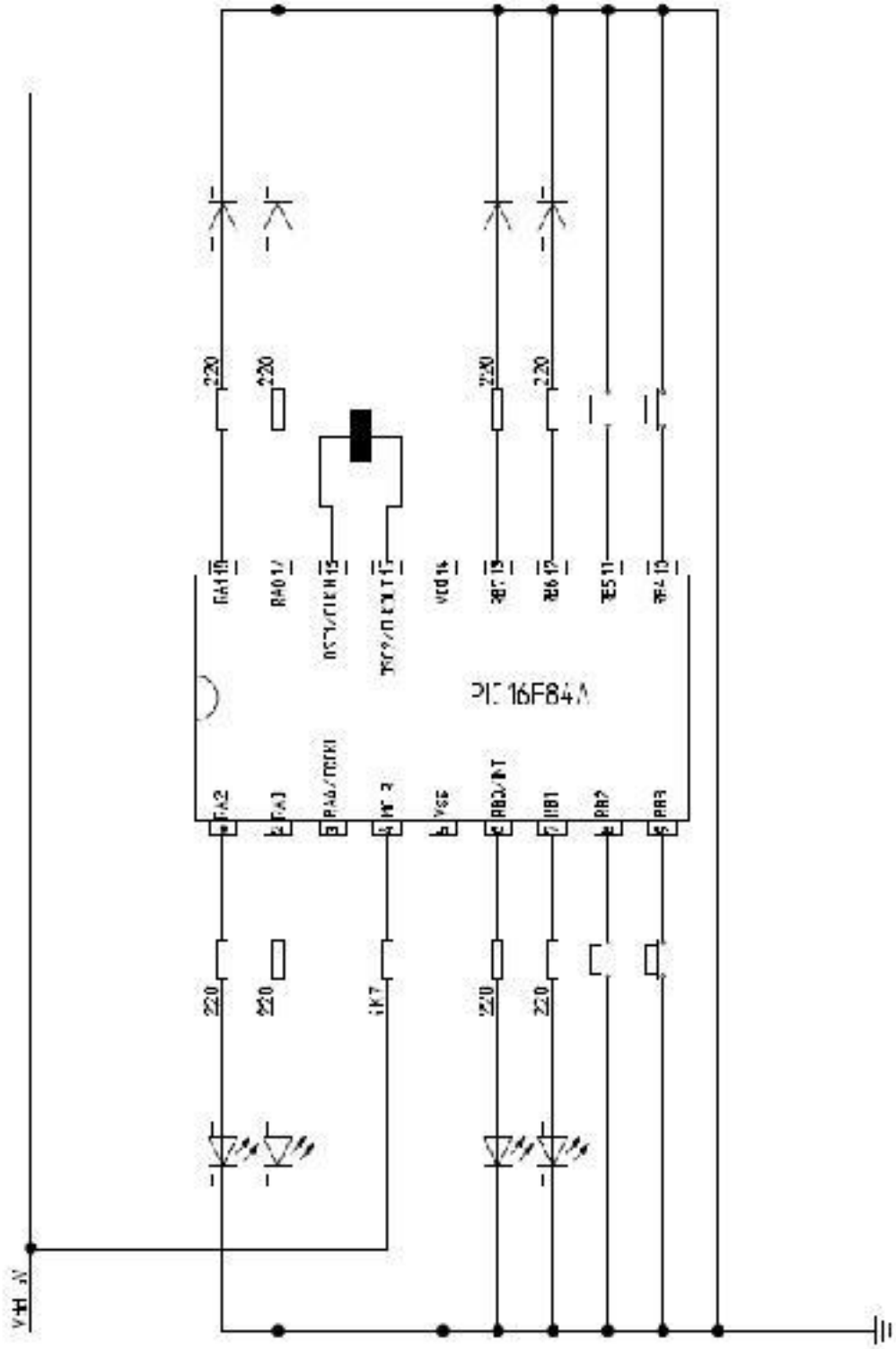
Izgraditi projekt i pokrenuti proces prevođenja projekta, te otkloniti pogreške u slučaju da se pojave.

Nakon uspješnog prevođenja asemblerske datoteke potrebno je pokrenuti simulator.

U simulatoru provjeriti rad programa i ustanoviti je li rad ispravan.

Kad se ustanovi ispravan rad potrebno je prijeći na prebacivanje programa u mikrokontroler pomoću programiralice.

Nakon programiranja mikrokontrolera potrebno je spojiti shemu sa slike 3.1. i provjeriti rad sklopa.



Slika 3.1 Čitanje i pisanje portova

Vježba 4. PALJENJE I GAŠENJE LED DIODA

4.1. Cilj vježbe: *Upoznati se s postupkom programiranja mikrokontrolera PIC16F84, te načinom uporabe njegovih portova i spajanja LED dioda.*

4.2. Opis vježbe:

Na ovoj vježbi upoznat ćete se s načinom programiranja mikrokontrolera PIC16F84, te načinom uporabe portova mikrokontrolera. Također ćete se upoznati s tipičnim načinom spajanja elemenata na ulaz i izlaz portova mikrokontrolera, te konfiguracijom oscilatora. Diode su spojene tako da svijetle kad je na pinovima niska razina. Ovo je napravljeno ovako iz razloga što je struja poniranja po pinu veća nego struja izvora po pinu. Pinovi koriste unutarnji pull-up koji je slabiji nego onaj koji bi bio izveden izvana (npr. Vdd preko otpornika 4k7), ali koji u ovom slučaju nije potreban. Koristi se oscilator od 4MHZ, no mogao bi i neki drugi jer brzina u ovom slučaju nije bitna. Crvena dioda svijetli uvijek, dok se stanja zelene i žute diode izmjenjuju ovisno o tome je li tipkalo pritisnuto ili nije. Početno stanje je da zelena dioda ne svijetli a žuta svijetli. Pritiskom na tipkalo zelena dioda svijetli a žuta je ugašena.

4.3. Priprema za vježbu:

Napisati program koji će u sklopu sa slike 4.1. raditi sljedeće:

- postaviti pinove RB4, RB5 i RB6 porta B kao izlazne, a sve ostale kao ulazne;
- upaliti LED diode na pinovima RB4 i RB6;
- omogućiti interni pull-up;
- napraviti petlju koja će provjeravati je li tipkalo pritisnuto:
    - ako je pritisnuto onda je potrebno ugasiti LED diodu na pinu RB6 i upaliti diodu na pinu RB5;
    - ako nije pritisnuto onda je potrebno ugasiti LED diodu na pinu RB5 i upaliti diodu na pinu RB6.

4.4. Rad na vježbi:

Pokrenuti program MPLAB i u njemu napraviti projekt pod nazivom "vj4.pjt".
Program iz pripreme spremiti u datoteku "vj4.asm" i uključiti ju u projekt.

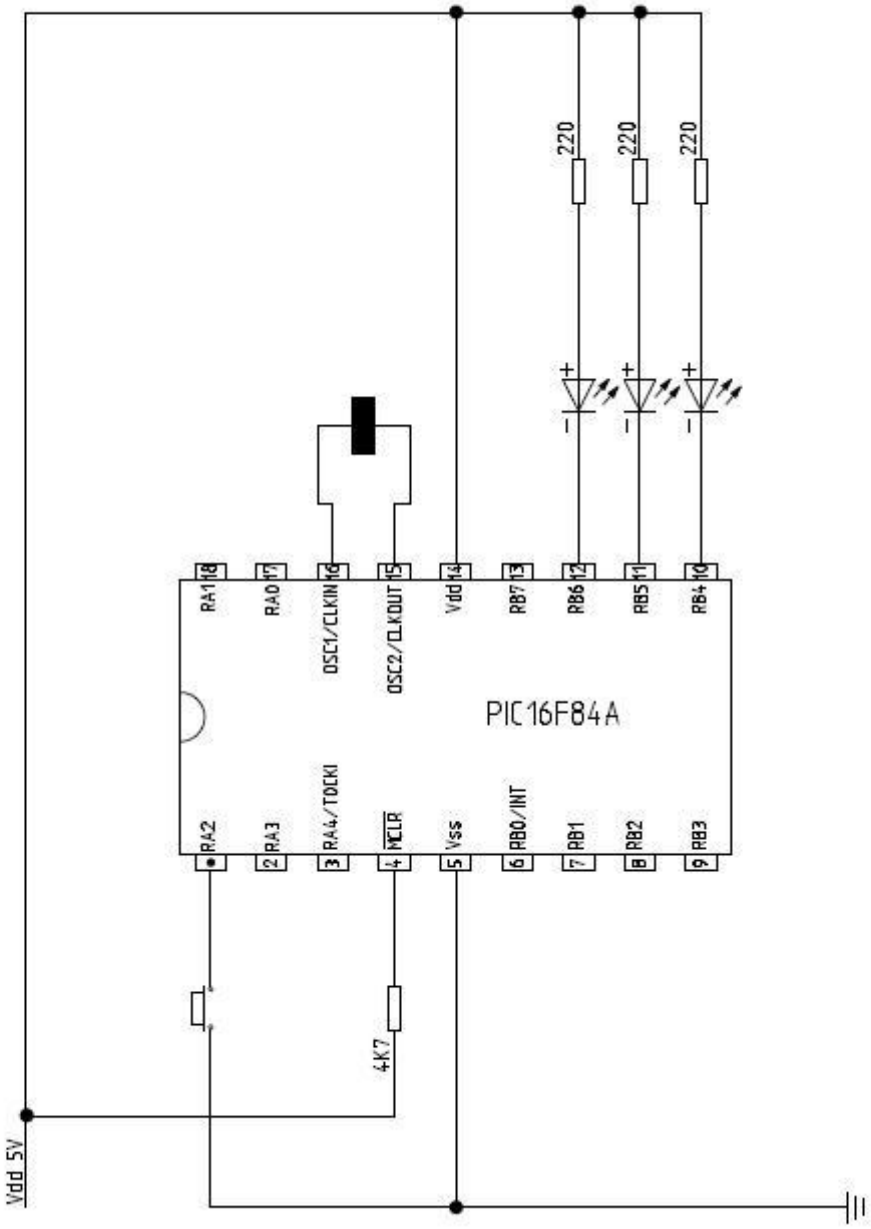
Izgraditi projekt i pokrenuti proces prevođenja projekta, te otkloniti pogreške u slučaju da se pojave.

Nakon uspješnog prevođenja asemblerske datoteke potrebno je pokrenuti simulator.

U simulatoru provjeriti rad programa i ustanoviti je li rad ispravan.

Kad se ustanovi ispravan rad potrebno je prijeći na prebacivanje programa u mikrokontroler pomoću programiralice.

Nakon programiranja mikrokontrolera potrebno je spojiti shemu sa slike 4.1. i provjeriti rad sklopa.



Slika 4.1 Sklop za paljenje i gašenje svijetlećih dioda

Vježba 5. ŠETAJUĆE SVJETLO

5.1. Cilj vježbe: *Upoznati se s postupkom programiranja mikrokontrolera PIC16F84, te načinom uporabe njegovih portova, kao i izradom potprograma za čekanje.*

5.2. Opis vježbe:

Na ovoj vježbi upoznat ćete se s načinom programiranja mikrokontrolera PIC16F84, te načinom uporabe portova mikrokontrolera. Također ćete se upoznati s tipičnim načinom spajanja elemenata na ulaz i izlaz portova mikrokontrolera, te konfiguracijom oscilatora. Diode su spojene tako da svijetle kad je na pinovima visoka razina. Koristi se oscilator od 4MHZ, što znači da vrijeme jednog instrukcijskog ciklusa iznosi 1µs. Svjetlo se prenosi s jedne diode na drugu, počevši od diode na pinu RB0 pa sve do one na pinu RB7, nakon čega se vraća opet do one na pinu RB0. Kako bi se prenošenje svjetla moglo vidjeti potrebno je programski izvesti potprogram za čekanje koji će pojedinu diodu držati upaljenom određeni broj milisekundi.

5.3. Priprema za vježbu:

Napisati program koji će u sklopu sa slike 5.1. raditi sljedeće:

- postaviti pinove RB1, RB2, RB3, RB4, RB5, RB6 i RB7 porta B kao izlazne;
- postaviti pin RA2 kao ulazni;
- upaliti LED diodu na pinu RB0;
- pričekati oko **V** ms i ugasiti diodu na pinu RB0 te upaliti diodu na pinu RB1;
- pričekati oko **V** ms i ugasiti diodu na pinu RB1 te upaliti diodu na pinu RB2;
- ponoviti postupak za sve diode do one na pinu RB7;
- pričekati oko **V** ms i ugasiti diodu na pinu RB7 te upaliti diodu na pinu RB6;
- pričekati oko **V** ms i ugasiti diodu na pinu RB6 te upaliti diodu na pinu RB5;
- ponoviti postupak za sve diode do one na pinu RB0;
- ponavljati cjelokupni postupak.
-

NAPOMENA: Ovo je samo primjer jednog mogućeg načina prenošenja svjetla. Svjetlo se ovdje prenosi s diode na diodu od diode RB0 do diode RB7 te natrag na RB0. **Svaki student u grupi izraditi će pomicanje svjetla na način kako je određeno u njegovom zadatku.**

5.4. Rad na vježbi:

Pokrenuti program MPLAB i u njemu napraviti projekt pod nazivom "vj5.pjt".
Program iz pripreme spremiti u datoteku "vj5.asm" i uključiti ju u projekt.

Izgraditi projekt i pokrenuti proces prevođenja projekta, te otkloniti pogreške u slučaju da se pojave.

Nakon uspješnog prevođenja asemblerske datoteke potrebno je pokrenuti simulator.

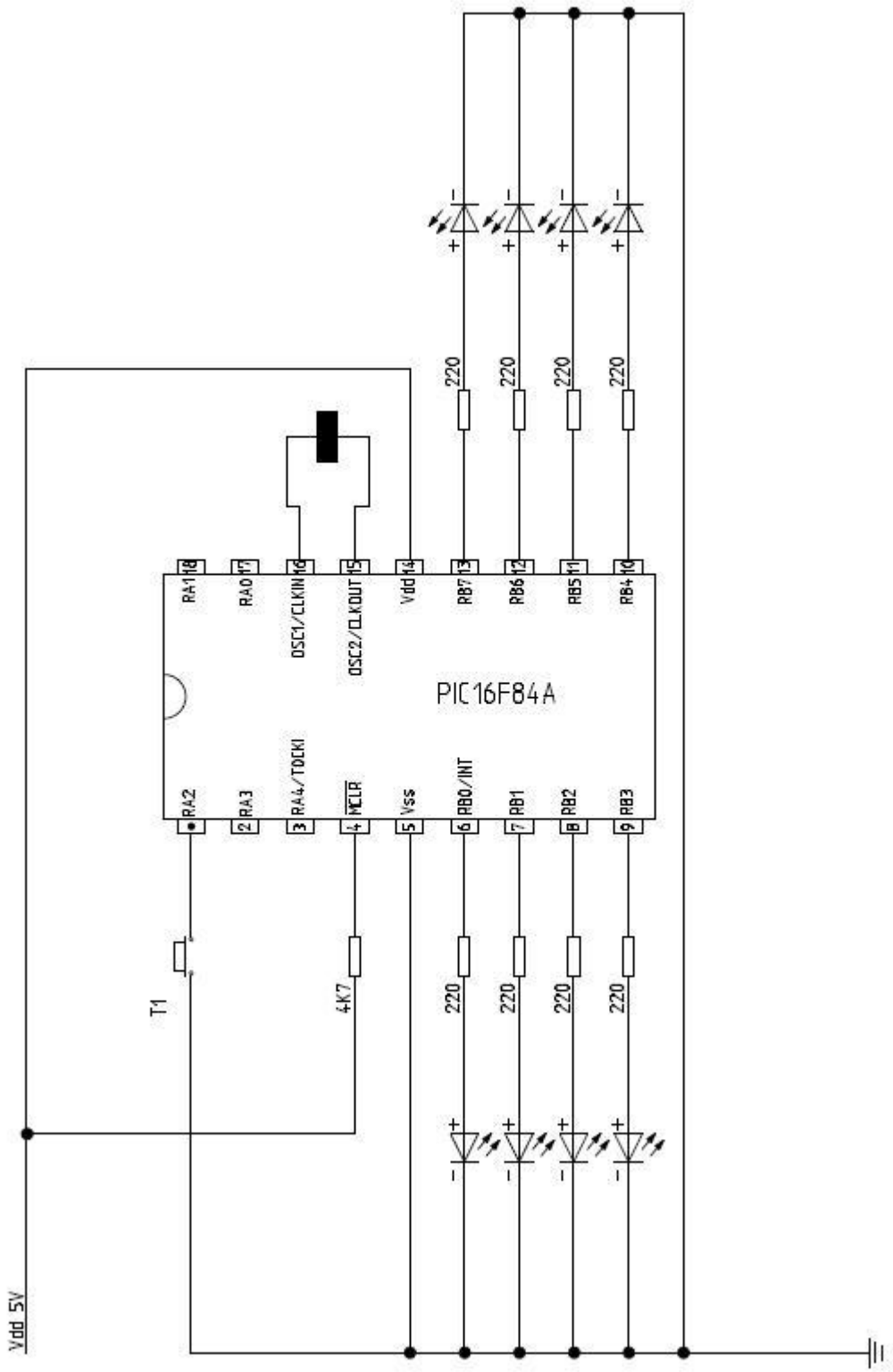
U simulatoru provjeriti rad programa i ustanoviti je li rad ispravan.

Kad se ustanovi ispravan rad potrebno je prijeći na prebacivanje programa u mikrokontroler pomoću programiralice.

Nakon programiranja mikrokontrolera potrebno je spojiti shemu sa slike 5.1. i provjeriti rad sklopa.

Zadaci za 5. laboratorijsku vježbu iz Građe Računala

1. Svjetlo se prenosi tako da u svakom trenutku svijetli samo jedna dioda počevši od diode spojene na RB7 do diode spojene na RB0. Nakon završenog ciklusa, svjetlo se postavlja opet na RB7 i ide do RB0. Redoslijed paljenja dioda je RB7, RB6, …RB1, RB0, RB7, RB6,…RB0,… V = 100ms
2. Svjetlo se prenosi tako da u svakom trenutku svijetli samo jedna dioda počevši od diode spojene na RB0 do diode spojene na RB7. Nakon završenog ciklusa, svjetlo se postavlja opet na RB0 i ide do RB7. Redoslijed paljenja dioda je RB0, RB1, RB2,…RB6, RB7, RB0, RB1, …, RB7, RB0,.. V = 100ms
3. Svjetlo se prenosi na način da se postupno pale sve diode redom od RB0 do RB7 s vremenskim razmakom od V ms. Kada se upali posljednja dioda RB7, pricekati V ms, ugasiti sve diode, pričekati V ms i ponoviti ciklus. V = 75ms
4. Svjetlo se prenosi na način da se postupno pale sve diode redom na RB7 do RB0 s vremenskim rzamakom od V ms. Kada se upali posljednja dioda RB0, pričekati V ms, ugasiti sve diode, pričekati V ms i ponoviti ciklus. V = 50ms
5. Svjetlo se prenosi tako da u svakom trenutku svijetli samo jedna dioda na način da se pale prvo diode na parnim pinovima MCU, pa nakon toga na neparnim pinovima MCU. Red paljenja dioda je RB0, RB2, RB4, RB6, RB1, RB3, RB5, RB7, RB0, RB2, … V = 30ms
6. Svjetlo se prenosi tako da u svakom trenutku svijetli samo jedna dioda na način da se pale prvo diode na neparnim pinovima MCU, pa nakon toga na parnim pinovima MCU. Red paljenja dioda je RB1, RB3, RB5, RB7, RB0, RB2, RB4, RB6, RB1, RB3, RB5, RB7, RB0, RB2, … V = 50ms
7. Upaliti sve diode. Gasiti redom jednu po jednu diodu počevši od RB0 do RB7. Redoslijed gašenja dioda je RB0, RB1, RB2, …, RB7, nakon toga, pričekati V ms, upaliti sve diode, pričekati V ms te ponoviti postupak. V = 60ms
8. Upaliti sve diode. Gasiti redom jednu po jednu diodu počevši od RB7 do RB0. Redoslijed gašenja dioda je RB7, RB6, RB5, …, RB0, nakon toga, pričekati V ms, upaliti sve diode, pričekati V ms te ponoviti postupak. V = 70ms
9. Ugasiti diode RB0 do RB3, upaliti diode RB4 do RB7. Redom, istovremeno upaliti diodu RB0 i ugasiti diodu RB4, pričekati V ms, upaliti diodu RB1, ugasiti diodu RB5, pričekati V ms, upaliti diodu RB2, ugasiti diodu RB6, pričekati V ms, upaliti diodu RB3, ugasiti diodu RB7. Pričekati V ms, te ponoviti cijelokupni postupak. V = 80ms
10. Ugasiti diode RB0, RB2, RB4, RB6, sve ostale upaliti. Istovremeno upaliti diodu RB0 i ugasiti RB1, pričekati V ms, upaliti diodu RB2 i ugasiti diodu RB3, pričekati V ms, nastaviti postupak do dioda RB6 i RB7, pričekati V ms, te ponoviti cijelokupni postupak. V = 90ms
11. Svjetlo se prenosi na način da su u svakom trenutku upaljene 2 susjedne diode (RB0 i RB1, RB1 i RB2, RB2 i RB3, itd.) Na ovaj način prenijeti svjetlo s diode RB0 na diodu RB7. Nakon svakog prenošenja svjetla pričekati V ms. Ponavljati postupak. V = 85ms
12. Svjetlo se prenosi na način da su u svakom trenutku upaljene 3 susjedne diode (RB0, RB1, RB2; RB1, RB2, RB3; RB2, RB3, RB4, …) Na ovaj način prenijeti svjetlo s diode RB0 na diodu RB7. Nakon svakog prenošenja svjetla pričekati V ms. Ponavljati postupak. . V = 105ms
13. Ugasiti sve diode. Istovremeno paliti RB0 i RB7, pričekati V ms, ugasiti sve diode, paliti RB1 i RB6, pričekati V ms, ugasiti sve diode itd. sve do RB3 i RB4. Ugasiti sve diode. Pričekati V ms. Ponoviti cijeli postupak. V = 130ms
14. Ugasiti sve diode. Istovremeno paliti RB0 i RB4, pričekati V ms, ugasiti sve diode, upaliti RB1 i RB5, pričekati V ms, ugasiti sve diode itd. sve do RB4 i RB7. Pričekati V ms. Ponoviti cijeli postupak. V = 140ms
15. 10. invertirani zadatak. Izraditi zadatak na način da se diode pale i gase suprotno od onog u 10. zadatku. V = 170ms
16. 11. invertirani zadatak. Izraditi zadatak na način da se diode pale i gase suprotno od onog u 11. zadatku. V = 180ms
17. 12. invertirani zadatak. Izraditi zadatak na način da se diode pale i gase suprotno od onog u 12. zadatku. V = 120ms
18. 13. invertirani zadatak. Izraditi zadatak na način da se diode pale i gase suprotno od onog u 13. zadatku. V = 195ms



Slika 5.1 Sklop za šetajuće svjetlo

Vježba 6. BROJENJE PRITISKOM NA TIPKU

6.1. Cilj vježbe: *Upoznati se s postupkom programiranja mikrokontrolera PIC16F84, te načinom kontroliranja rada mikrokontrolera vanjskim djelovanjem, kao i osnovnim matematičkim operacijama.*

6.2. Opis vježbe:

Mikrokontroler se često koristi u sklopovima kao brojač. Jedan jednostavan primjer je brojenje pritiskom na tipku. Na sličan način se može implementirati i brojenje na neki drugi način. Primjerice, brojenje impulsa iz nekog davača. Na ovoj vježbi bit će prikazano brojenje do 15, za što su potrebna četiri bita. Implementacija za brojenje većih vrijednosti je slična (koristi se samo više izlaza). Rezultat u će se prikazivati na četiri diode koje su spojene na port B prema slici 6.1. Diode moraju odražavati stanje brojača na način da upaljena dioda označava "1" a ugašena "0". Tako bi se broj 10 na didodama prikazao na način da su diode RB5 i RB7 upaljene, a sve ostale ugašene.

Potrebno je voditi računa i o tome da tipkalo treba neko vrijeme za preklapanje, te da se ne može promatrati kao ON-OFF sklopka koja bi trenutno preklopila. Većina tipkala ima vrijeme potrebno za preklapanje ispod 10 ms. Tipkalo T1 povećava vrijednost brojača za 1. Kada vrijednost brojača postane veća od 15 postaviti brojač na 0. Tipkalo T2 smanjuje vrijednost brojača za 1. Kada vrijednost brojača postane manja od 0 postaviti vrijednost brojača na 15.

6.3. Priprema za vježbu:

Napisati program koji će u sklopu sa slike 6.1. raditi sljedeće:

- postaviti pinove porta B RB4, RB5, RB6 i RB7 kao izlazne, a pinove RB2 i RB3 kao ulazne;
- omogućiti interni pull-up;
- provjeravati stanje tipkala na pinu RB2 i RB3;
- ako nije pritisnuto onda provjeravaj kada bude pritisnuto;
- ako je tipkalo pritisnuto onda treba uvećati/smanjiti stanje na portu B za jedan i pričekati 10ms;
- provjeravati je li tipkalo na pinu RB2/RB3 otpušteno;
- ako nije onda čekati da se otpusti tipkalo;
- ako je otpušteno onda pričekati 10 ms i provjeriti je li još uvijek otpušteno;
- ako nije onda pričekati da bude otpušteno;
- ako je otpušteno onda se vratiti na provjeru kada će biti ponovo pritisnuto.

NAPOMENA: Ovo je samo primjer načina brojenja. **Svaki student u grupi izraditi će vježbu na način kako je određeno u njegovom zadatku.** Pojedinačni zadaci za studente definiraju **razliku** s obzirom na izvorni zadatak. Sve što nije definirano u pojedinačnom zadatku izvodi se prema izvornom zadatku.

6.4. Rad na vježbi:

Pokrenuti program MPLAB i u njemu napraviti projekt pod nazivom "vj6.pjt".

Program iz pripreme spremiti u datoteku "vj6.asm" i uključiti ju u projekt.

Izgraditi projekt i pokrenuti proces prevođenja projekta, te otkloniti pogreške u slučaju da se pojave.

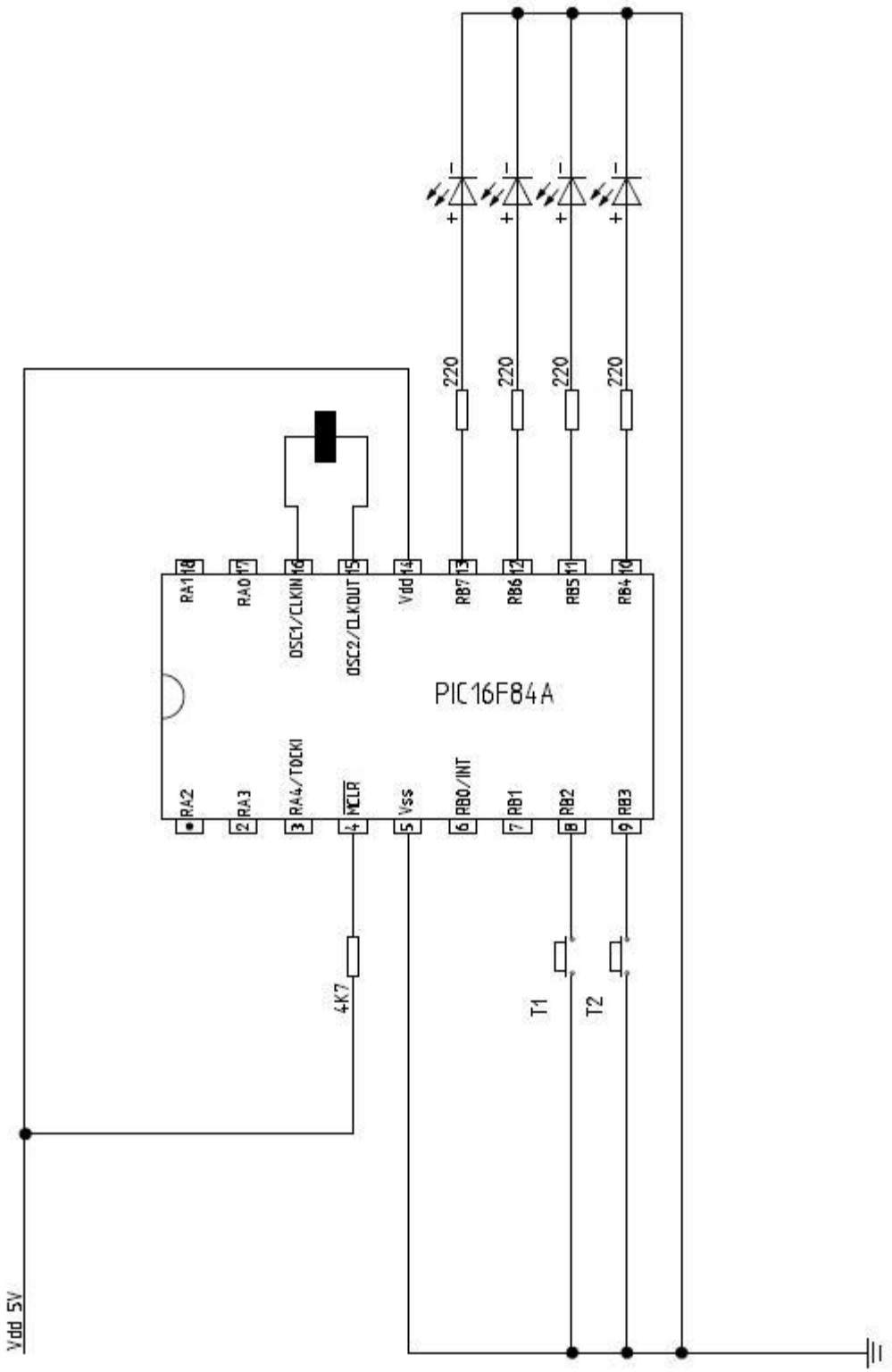
Nakon uspješnog prevođenja asemblerske datoteke potrebno je pokrenuti simulator.

U simulatoru provjeriti rad programa i ustanoviti je li rad ispravan.

Kad se ustanovi ispravan rad potrebno je prijeći na prebacivanje programa u mikrokontroler pomoću programiralice.

Nakon programiranja mikrokontrolera potrebno je spojiti shemu sa slike 6.1. i provjeriti rad sklopa.

1. Tipkalo T1 uvećava vrijednost brojača za 1. Tipkalo T2 smanjuje vrijednost brojača za 1. (zadatak jednak izvornom zadatku)
2. Tipkalo T1 uvećava vrijednost brojača za 2. Tipkalo T2 smanjuje vrijednost brojača za 2.
3. Tipkalo T1 uvećava vrijednost brojača za 1. Tipkalo T2 smanjuje vrijednost brojača za 2.
4. Tipkalo T1 uvećava vrijednost brojača za 2. Tipkalo T2 smanjuje vrijednost brojača za 1.
5. Broji se od 0 do 4. Kada se u brojaču nalazi vrijednost 0 sve su diode ugašene; kada se u brojaču nalazi vrijednost 1 upaljena je jedna dioda, 2 -> upaljene su 2 diode, itd..
6. Tipkalo T1 uvećava vrijednost brojača za 2. Tipkalo T2 koristi se za postavljanje brojača na 0.
7. Tipkalo T2 koristi se za uvećavanje brojača za vrijednost 5. U slučaju kada vrijednost postane veća od 15, oduzeti 15 od brojača (u brojaču je vrijenost 14, pritisne se tipka T2 : 14 + 5 = 19 (preljev), 19-15 = 4, u brojač staviti 4 i nastaviti program.
8. Tipkalo T2 koristi se za smanjivanje brojača za vrijednost 5. U slučaju kada vrijednost postane manja od 0, dodati 15 brojaču (u brojaču je vrijenost 3, pritisne se tipka T2 : 3 - 5 = -2 (preljev), -2+15 = 13, u brojač staviti 13 i nastaviti program.
9. Na diodama prikazati samo parne brojeve. Kada je u brojaču neparni broj, ugasiti sve diode.
10. Na diodama prikazati samo neparne brojeve. Kada je u brojaču parni broj, upaliti sve diode.



Slika 6.1 Sklop za brojenje pritiskom na tipku

Vježba 7. BINARNI BROJAČ

7.1. Cilj vježbe: *Upoznati se s postupkom programiranja mikrokontrolera PIC16F84, te načinom brojenja i prikazom rezultata brojenja.*

7.2. Opis vježbe:

Jedan od načina korištenja mikrokontrolera je korištenje u svrhu brojenja. Jednostavan primjer je binarno brojenje. Na sličan način se može implementirati i brojenje na neki drugi način. Na ovoj vježbi bit će prikazano brojenje do 15, za što su potrebna četiri bita. Implementacija za brojenje većih vrijednosti je slična (koristi se samo više izlaza). Rezultat će se prikazivati na četiri diode koje su spojene na port B prema slici 7.1. Vrijednost brojača uvećavat će se za jedan svaki interval vremena. Izraditi potprograme koji će imati kašnjenje od 100ms. Kašnjenje se izvodi pomoću mjerača vremena (engl. timer) TMR0.

7.3. Priprema za vježbu:

Napisati program koji će u sklopu sa slike 7.1. raditi sljedeće:

- postaviti pinove porta B RB0, RB1, RB2 i RB3 kao izlazne;
- omogućiti interni pull-up i postaviti predjelilo za TMR0;
- svaki vremenski interval uvećavati stanje na portu B za jedan.
- Sav kôd smjestiti u jednu .asm datoteku. Koristiti programske direktive za odabir dijela kôda koji se koristi

NAPOMENA: Ovo je samo primjer načina brojenja. **Svaki student u grupi izraditi će vježbu na način kako je određeno u njegovom zadatku.** Pojedinačni zadaci za studente definiraju **razliku** s obzirom na izvorni zadatak. Sve što nije definirano u pojedinačnom zadatku izvodi se prema izvornom zadatku.

7.4. Rad na vježbi:

Pokrenuti program MPLAB i u njemu napraviti projekt pod nazivom "vj7.pjt".
Program iz pripreme spremiti u datoteku "vj7.asm" i uključiti ju u projekt.

Izgraditi projekt i pokrenuti proces prevođenja projekta, te otkloniti pogreške u slučaju da se pojave.

Nakon uspješnog prevođenja asemblerske datoteke potrebno je pokrenuti simulator.

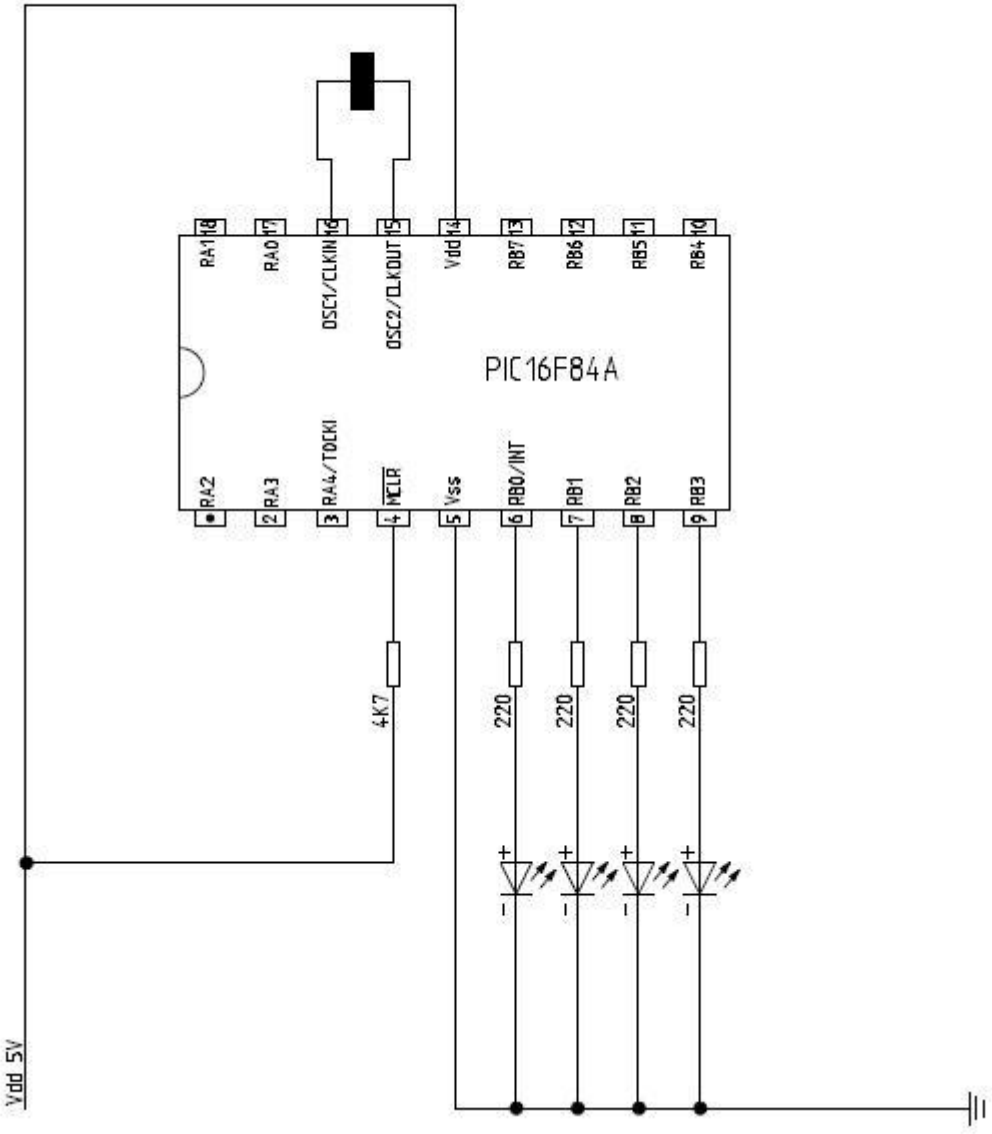
U simulatoru provjeriti rad programa i ustanoviti je li rad ispravan.

Kad se ustanovi ispravan rad potrebno je prijeći na prebacivanje programa u mikrokontroler pomoću programiralice.

Nakon programiranja mikrokontrolera potrebno je spojiti shemu sa slike 7.1. i provjeriti rad sklopa.

Zadaci za 7. laboratorijsku vježbu iz Građe Računala

1. Broji se od 0 do 4 i unatrag (od 4 do 0). Kašnjenje za pozitivan smjer je 80ms a za negativan 150ms. Kada se u brojaču nalazi vrijednost 0 sve su diode ugašene; kada se u brojaču nalazi vrijednost 1 upaljena je jedna dioda, 2 -> upaljene su 2 diode, itd..
2. Broji se od 0 do 15. Za brojenje do 8 koristiti kašnjenje od 100ms, a dalje koristiti kašnjenje od 200ms.
3. Korak brojenja u pozitivnom smjeru je 4 (broji se 0, 4, 8, 12) a u negativnom 2 (12, 10, 8, …, 0). Postupak brojenja stalno ponavljati.
4. Na diodama prikazati samo parne brojeve. Kada je u brojaču neparni broj, ugasiti sve diode.
5. Na diodama prikazati samo neparne brojeve. Kada je u brojaču parni broj, upaliti sve diode.
6. Broji se od 0 do 8. Početi od 100ms kašnjenja. Vrijeme kašnjenja povećavati za 25ms za vrijednost brojača od 0 do 4 za svaku promjenu brojača. Vrijeme kašnjenja smanjivati za 25ms za vrijednost brojača od 5 do 8 za svaku promjenu brojača.
7. Broji se od 0 do 4 i unatrag (od 4 do 0). Početi od kašnjenja 40ms. Kašnjenje za pozitivan smjer uvećavati za 40ms na svaku promjenu brojača. Kašnjenje za negativan smjer smanjivati za 40ms na svaku promjenu brojača. Kada se u brojaču nalazi vrijednost 0 sve su diode ugašene; kada se u brojaču nalazi vrijednost 1 upaljena je jedna dioda, 2 -> upaljene su 2 diode, itd..
8. Jednako kao i 5. zadatak s razlikom da ovdje stanje dioda u svakom trenutku mora biti invertirano, tj. kada u 5. zadataku dioda svijetli u ovom ne svijetli.
9. Jednako kao i 6. zadatak s razlikom da ovdje stanje dioda u svakom trenutku mora biti invertirano, tj. kada u 6. zadataku dioda svijetli u ovom ne svijetli.
10. Jednako kao i 7. zadatak s razlikom da ovdje stanje dioda u svakom trenutku mora biti invertirano, tj. kada u 7. zadataku dioda svijetli u ovom ne svijetli.
11. Jednako kao i 1. zadatak s razlikom da ovdje stanje dioda u svakom trenutku mora biti invertirano, tj. kada u 1. zadataku dioda svijetli u ovom ne svijetli.
12. Jednako kao i 2. zadatak s razlikom da ovdje stanje dioda u svakom trenutku mora biti invertirano, tj. kada u 2. zadataku dioda svijetli u ovom ne svijetli.
13. Jednako kao i 3. zadatak s razlikom da ovdje stanje dioda u svakom trenutku mora biti invertirano, tj. kada u 3. zadataku dioda svijetli u ovom ne svijetli.
14. Jednako kao i 4. zadatak s razlikom da ovdje stanje dioda u svakom trenutku mora biti invertirano, tj. kada u 4. zadataku dioda svijetli u ovom ne svijetli.



Slika 7.1 Sklop za binarni sat

Vježba 8. BROJAČ IMPULSA

8.1. Cilj vježbe: *Upoznati se s postupkom programiranja mikrokontrolera PIC16F84, te načinom korištenja prekida u svrhu brojenja impulsa.*

8.2. Opis vježbe:

U sve više aplikacija u životu koriste se brojači impulsa. Od primjene za brojenje broja putnika, broja posjetilaca, broja automobila, … U industriji je također nezamisliv rad bez brojača impulsa. Najočitiji primjer je brojenje komada određenih artikala kako bi se imao uvid u proizvodnju odnosno potrošnju. Sljedeći primjer je brojenje impulsa u telefonskim centralama. Uglavnom, primjera ima svuda oko nas i svakim danom ih je sve više. Sam brojač impulsa može se promatrati kroz tri dijela. Prvi dio je davač impulsa, a zatim slijedi dio za brojenje, pamćenje i za pripremu za prikazivanje (PIC16F84), te na kraju dio za prikaz informacije (7-segmentni pokaznici). Za prikaz rezultata brojenja koriste se dva 7-segmentna pokazivača koji su spojeni paralelno na port B, a jedino su zajedničke katode spojene na port A (svaka na drugi pin).

8.3. Priprema za vježbu:

Napisati program koji će u sklopu sa slike 8.1. raditi sljedeće:

- postaviti pinove porta A i porta B prema slici 8.1.;
- brojiti impulse na pinu RB7 (na rastući brid);
- u prekidnoj rutini povećati sadržaj registra u kojem su spremljene jedinice za jedan, provjeriti je li vrijednost dostigla iznos 10, te u slučaju da je dostigla postaviti registar koji sadrži jedinice na nulu a vrijednost registra koji sadrži desetice povećati za jedan;
- kada vrijednost broja u registru desetica dostigne vrijednost 10 onda se on postavlja na vrijednost nula i brojanje počinje ispočetka;
- tokom izvršavanja glavnog programa potrebno je naizmjence prikazivati vrijednosti jedinica i desetica na 7-segmentnim pokazivačima.

NAPOMENA: Ovo je samo primjer načina brojenja. **Svaki student u grupi izraditi će vježbu na način kako je određeno u njegovom zadatku.** Pojedinačni zadaci za studente definiraju **razliku** s obzirom na izvorni zadatak. Sve što nije definirano u pojedinačnom zadatku izvodi se prema izvornom zadatku.

8.4. Rad na vježbi:

Pokrenuti program MPLAB i u njemu napraviti projekt pod nazivom "vj8.pjt".

Program iz pripreme spremiti u datoteku "vj8.asm" i uključiti ju u projekt.

Izgraditi projekt i pokrenuti proces prevođenja projekta, te otkloniti pogreške u slučaju da se pojave.

Nakon uspješnog prevođenja asemblerske datoteke potrebno je pokrenuti simulator.

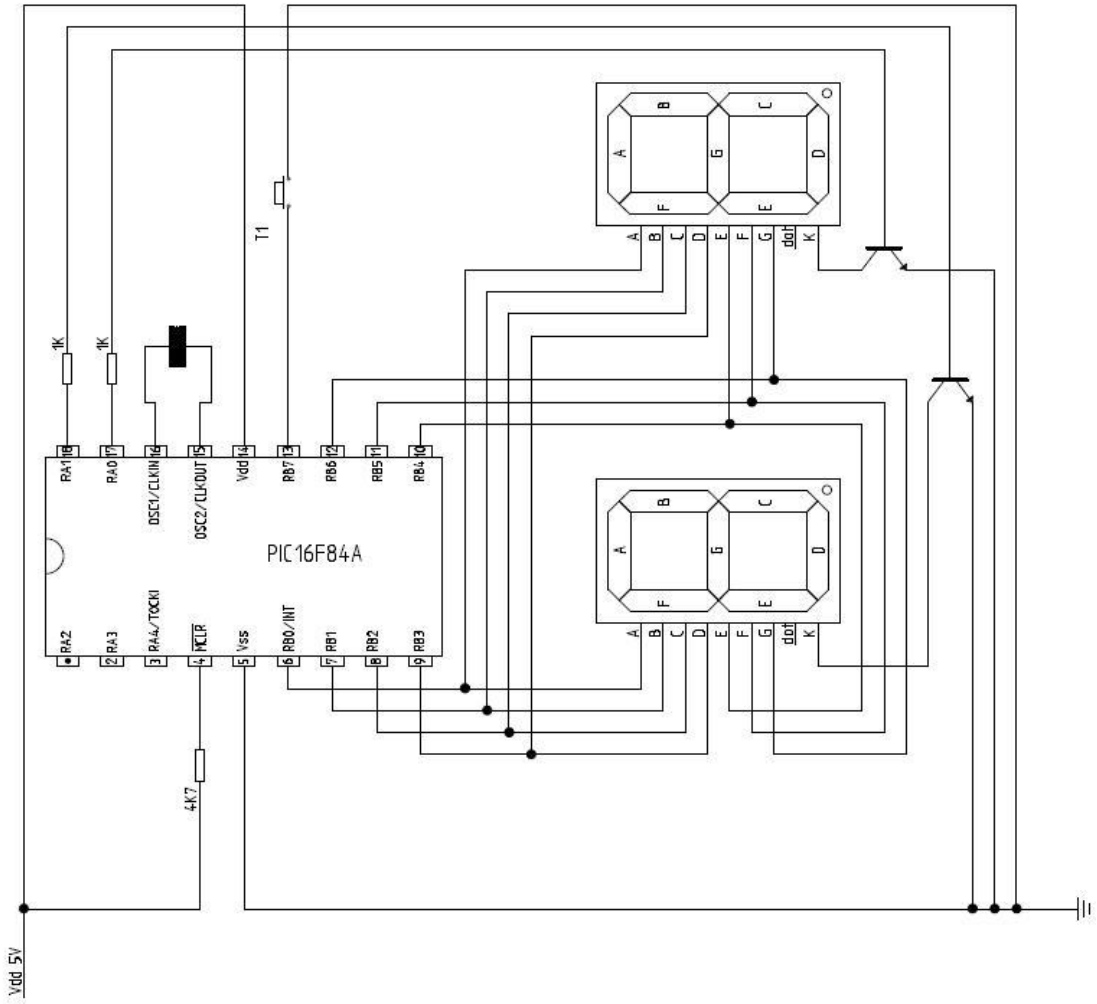
U simulatoru provjeriti rad programa i ustanoviti je li rad ispravan.

Kad se ustanovi ispravan rad potrebno je prijeći na prebacivanje programa u mikrokontroler pomoću programiralice.

Nakon programiranja mikrokontrolera potrebno je spojiti shemu sa slike 8.1. i provjeriti rad sklopa.

Zadaci za 8. laboratorijsku vježbu iz Građe Računala

1. Broji se heksadecimalno od 0 do FF.
2. Broji se heksadecimalno od FF do 0.
3. Izraditi program za testiranje 7 segmentnog pokaznika. Lijevi pokaznik neka je P1 a desni P2.
   Na pritisak tipkala T1 s čekanjem od 190 ms izvesti sljedeće: Upali segmente P1-A, P2-A, čekaj, ugasi sve, Upali P2-B, P2-C, čekaj, ugasi sve, Upali P2-D, P1-D, čekaj, gasi sve, upali P1-E, P1-F, čekaj, ugasi sve, upali P1-G, P2-G, čekaj, ugasi sve, ponovi prethodnu proceduru.
4. Broji se decimalno od 0 do 9. Na oba pokaznika je u svakom trenutku ista vrijednost.
5. Broji de od 0 do 99. Pritisak na tipkalo koji traje do 200ms broji se kao jedan, a na svaki duzi pritisak brojac se jednokratno povecava za 10.
6. Broji de od 99 do 0. Pritisak na tipkalo koji traje do 200ms broji se kao jedan, a na svaki duzi pritisak brojac se jednokratno smanjuje za 10.
7. Broji se decimalno od 9 do 0. Na oba pokaznika je u svakom trenutku ista vrijednost.
8. Na jednom pokazniku brojiti od 0 do 9 a na drugom od 9 do 0. Kada je na prvom vrijednost 0 na drugom je 9, kada je na prvom vrijednost 1 na drugom je 8, kada je na prvom vrijednost 2 na drugom je 7, …
9. Izraditi program za testiranje 7 segmentnog pokaznika. Na pritisak tipkala T1 s čekanjem od 190 ms izvesti paljenje svih segmenata od A do G na oba pokaznika na način: upali segmente A, čekaj, ugasi sve, upali segmente B, čekaj, ugasi sve, upali segmente C, čekaj, ugasi sve, …
10. Broji se od -9 do 9.
11. Nakon pritiska tipkala T1, pokrenuti brojač koji će brojiti od 0 do FF, vrijednost brojača s čekanjem od 150ms prikazati na pokaznicima. Nakon prvog pritiska tipkala sklop bez daljnjih pritisaka na tipkalo broji do FF. Svaka vrijednost se prikazuje, čeka se 150 ms, te se prelazi na prikaz nove vrijednosti.
12. Po uključivanju sklopa bez pritisaka na tipkalo započinje se brojenje od 0 do FF. Vrijednost brojača s čekanjem od 100ms prikazati na pokaznicima. Pritiskom na T1 brojenje se zaustavlja.
    Ponovnim pritiskom na T1 brojenje se nastavlja.
13. Broji se heksadecimalno od -F do F.
14. Broji se od 0 do 99. Svaki pritisak na tipkalo broji se kao 5.
15. Broji se od 99 do 0. Svaki pritisak na tipkalo broji se kao -5.



Slika 8.1 Sklop za brojač impulsa

Vježba 9. POVEZIVANJE LCD ZASLONA

9.1. Cilj vježbe: *Upoznati se s postupkom programiranja mikrokontrolera PIC16F84, te načinom spajanja LCD zaslona i prikazom podataka na njemu.*

9.2. Opis vježbe:

Informacije dobivene mjerenjem ili nekim drugim putem moraju se prikazati kako bi ih mogli očitati. Za prikazivanje tih informacija postoji više načina koji se danas koriste. Jedan od njih je prikaz pomoću LCD zaslona. Na LCD zaslonu mogu se prikazivati i slova i brojevi što ga čini pogodnim za primjenu u raznim aplikacijama. Druga karakteristika koja čini LCD pogodnim za upotrebu je niska potrošnja. Sljedeća karakteristika je pouzdanost, a mogli bismo ih nabrojati i još više. Gotovo da i nema elektroničkih uređaja koji se danas mogu nabaviti a da u njima nema barem jedan ugrađen LCD zaslon. Sve ovo nam pokazuje važnost LCD zaslona. Zbog toga će se na ovoj vježbi naučiti kako se LCD zaslon spaja na mikrokontroler, i to na dva načina (jedan radi uštede pinova na mikrokontroleru), te kako se prikazuju podaci na njemu.

9.3. Priprema za vježbu:

Napisati program koji će u sklopu sa slike 9.1. raditi sljedeće:

- postaviti pinove porta B prema slici 9.1.;
- izraditi potprograme za inicijalizaciju, slanje naredaba i znakova na LCD
- koristeći potprograme ispisati svoje ime u prvom retku LCD zaslona, te svoje prezime u drugom retku LCD zaslona.

9.4. Rad na vježbi:

Pokrenuti program MPLAB i u njemu napraviti projekt pod nazivom "vj9.pjt". Program iz pripreme spremiti u datoteku "vj9.asm" i uključiti ju u projekt.

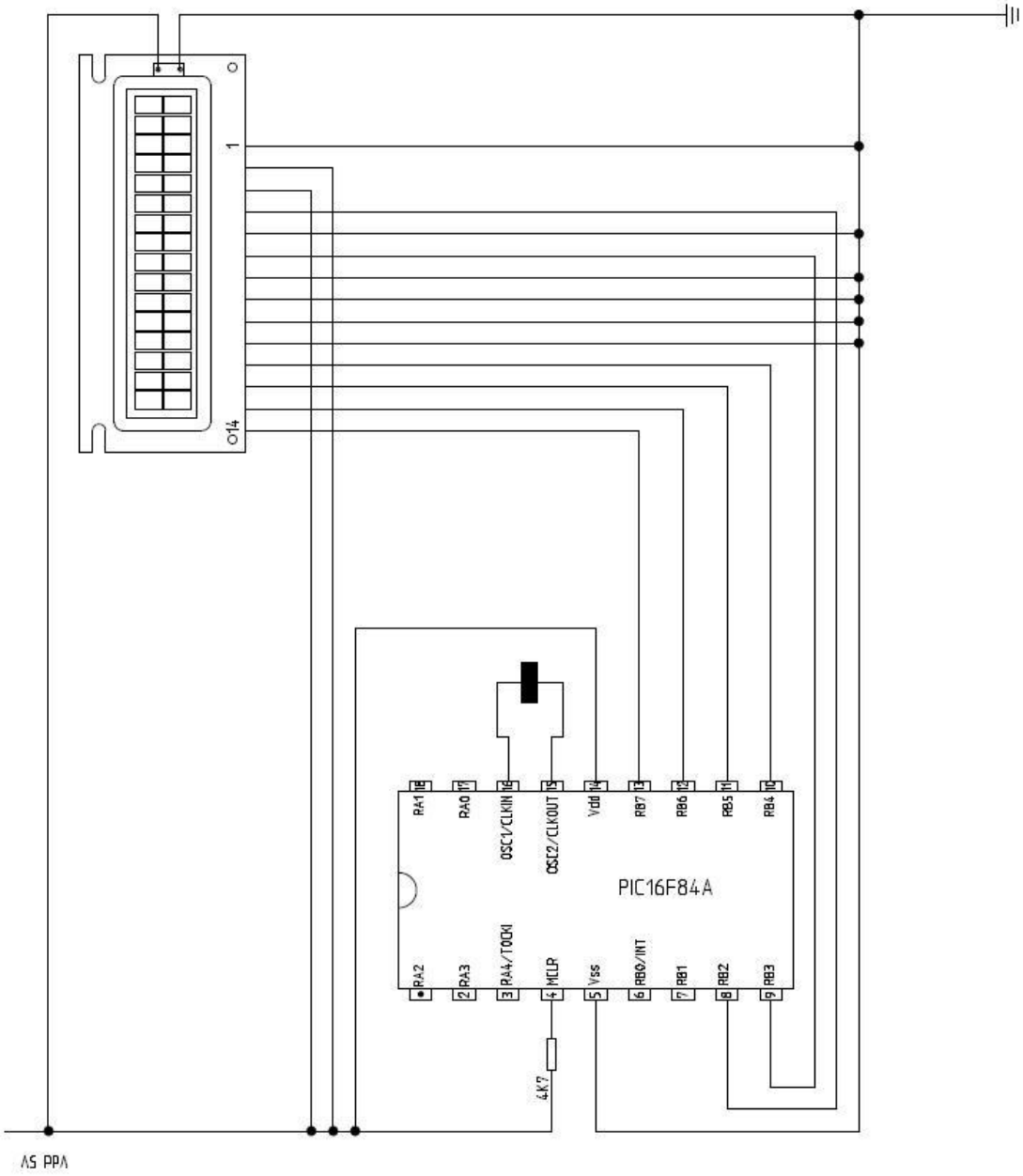
Izgraditi projekt i pokrenuti proces prevođenja projekta, te otkloniti pogreške u slučaju da se pojave.

Nakon uspješnog prevođenja asemblerske datoteke potrebno je pokrenuti simulator.

U simulatoru provjeriti rad programa i ustanoviti je li rad ispravan.

Kad se ustanovi ispravan rad potrebno je prijeći na prebacivanje programa u mikrokontroler pomoću programiralice.

Nakon programiranja mikrokontrolera potrebno je spojiti shemu sa slike 9.1. i provjeriti rad sklopa.



Slika 9.1 Sklop za povezivanje LCD zaslona

PROGRAMSKI PRIMJERI S PREDAVANJA

U ovom poglavlju se nalaze programski primjeri koji se koriste na predavnjima. Studentima se preporuča temeljito proučavanje ovih programskih primjera u svrhu pripreme za laboartorijske vježbe te prije izlaženja na kolokvije i ispite. Razina znanja koju je potrebno postići kako bi se sa punim samopouzdanjem moglo izaći na ispite je razumijevanje svake linije programskog kôda te sposobnost rješavanja zadataka koji se zadaju na temelju ovih primjera.

Primjer 1 – Prvi program, aritmetičke operacije

```
  ============================================================
  Primjeri 1
  ============================================================


  ============================================================
  Pretprocesorske naredbe
  ============================================================
                  PROCESSOR 16F84A
                  #include "p16f84a.inc"
                  ERRORLEVEL -224
                  __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF

                  org          0x0;
  ============================================================
  EQU - Pridjeljuje adresnoj lokaciji identifikator
  ============================================================
VARA            EQU        0x0C          ; VARA je mem. lokacija 0x0C
VARB            EQU        0x0D          ; VARB je mem. lokacija 0x0D
VARC            EQU        0x0E          ; VARC je mem. lokacija 0x0E
VARD            EQU        0x0F          ; VARD je mem. lokacija 0x0F
  ============================================================
  Pregled instrukcija
  movlw, baza konstante
  ============================================================
                nop                          ; No OPeration
                nop                          ; No OPeration
                nop                          ; No OPeration
                movlw      5                 ; 5 -> W
                movlw      9                 ; 9 -> W
                movlw      10                ; 16 -> W
                movlw      D'25'             ; 25 -> W
                movlw      O'20'             ; 16 -> W
                movlw      B'00001111'       ; 15 -> W
                movlw      0xF0              ; 240 -> W
; ============================================================
                movlw      D'300'            ; 300 -> W?
                                             ; Koliko je u W nakon ove naredbe?
                                             ; Zasto? 256 + 44 = 300
                                             ; 300(10) = 100101100(2)
                                             ;             |------|
                                             ;                44
  ============================================================
  Pregled instrukcija (1)
  ============================================================
                movlw      D'100'            ; 100 -> W
                clrw                                ; 0 -> W
                addlw      D'25'             ; W + 25 -> W
                addlw      0x19              ; W + 25 -> W
                movwf      VARA              ; W -> VARA
                movwf      VARB              ; W -> VARB
                addwf      VARB,W            ; W + VARB -> W
                addwf      VARB,F            ; W + VARB -> VARB
```

```
                clrf    VARA                ; 0 -> VARA
                clrf    VARB                ; 0 -> VARB
  ============================================================
  Pregled instrukcija (2)
  ============================================================
                    bsf         VARA,0              ; 1 -> 0. bit VARA
                bsf     VARA,1              ; 1 -> 1.  bit VARA
                bsf     VARA,2              ; 1 -> 2.  bit VARA
                bsf     VARA,3              ; 1 -> 3.  bit VARA
                bsf     VARA,4              ; 1 -> 4.  bit VARA
                bcf     VARA,4              ; 0 -> 4.  bit VARA
                comf    VARA,W              ; Komplementiraj VARA -> W
                movwf   VARC                ; W -> VARC
                decf    VARC,F              ; VARC - 1  -> VARC
                decf    VARC,F              ; VARC - 1  -> VARC
                incf    VARC,W              ; VARC + 1  -> W
                movwf   VARC                ; W -> VARC
                incf    VARC,F              ; VARC + 1  -> VARC
  ============================================================
  Pregled instrukcija (3), oduzimanje
  ============================================================
                movlw   D'50'         ; 50 -> W
                sublw   D'75'         ; 75 - W -> W
                nop                   ; ------------------
                movlw   D'10'         ; --\
                nop                   ;    ) 10 -> VARA
                movwf   VARA          ; --/
                movlw   D'30'         ; --\
                nop                   ;    ) 30 -> VARB
                movwf   VARB          ; --/
                nop                   ;..
                clrw                  ;....
                nop                   ;..
                nop                   ; VARC = VARB - VARA ?
                movf    VARA,W        ; VARA -> W
                subwf   VARB,W        ; VARB - W -> W
                movwf   VARC          ; W -> VARC
                nop                   ; ------------------
                nop
  ============================================================
  Pregled instrukcija (4), oduzimanje
  Interpretacija vrijednosti -> Dvojni komplement
  ============================================================
                movlw   D'5'                ; 10 -> W
                sublw   D'2'                ; 2 - W -> W (W == 253)
                nop                         ;
                movlw   D'253'              ; 253 -> W
                addlw   D'5'                ; 5 + W -> W (W == 2)
                nop                         ;
  ============================================================
  Pregled instrukcija (5), labela i goto
  ============================================================
tu1             nop                         ;
                goto    tu3                 ; idi na "tu3"
                nop                         ;
tu2             nop                         ;
                nop                         ;
                goto    tu4                 ; idi na tu4
tu3             nop                         ;
                goto    tu2                 ; idi na tu2
tu4             nop                         ;
  ============================================================
  Pregled instrukcija (6), grananje
  ============================================================
                clrf    VARA                ; 0 -> VARA
```

```
                bsf     VARA,3          ; 1 -> 3. bit VARA
                bcf     VARA,3          ; 0 -> 3. bit VARA
                nop                     ;
                nop                     ;
                btfsc   VARA, 3         ; 3. bit VARA je 0?
                goto    s1              ; idi na s1
                goto    s2              ; idi na s2
s1              nop                     ; s1
                nop                     ;
                nop                     ;
                goto    kr1             ; idi na kr1 (kraj)
s2              nop                     ; s2
                nop                     ;
                nop                     ;
kr1             nop                     ; kraj
  ============================================================
  Pregled instrukcija (6), grananje
  Na RB0 dolaze signali (0/1), reagirati u ovisnosti.
  Debugger -> Stimulus controller -> New scenario
  RB0 Set high & RB0 Set low
  ============================================================
poc1            nop                     ;
                btfss   PORTB,0         ; 0. bit PORTB je 1?
                goto    p2              ; idi na p2
p1              nop                     ; p1
                nop                     ;
                goto    kr2             ; idi na kr2 (kraj)
p2              nop                     ; p2
                nop                     ;
kr2             nop                     ; kr2 (kraj)
;               goto    poc1            ; idi na pocetak
  ============================================================
  Pregled instrukcija (7), brojenje od 5 do 0
  ============================================================
                movlw   D'5'            ; 5 -> W
                movwf   VARA            ; W -> VARA
opet1           nop                     ;
                decfsz  VARA,F          ; VARA - 1 -> VARA, da li je VARA 0?
                goto    opet1           ; idi na opet1
                nop                     ;
kr3             nop                     ; kr3 (kraj)
  ============================================================
  Operacija =, da li su brojevi u VARA i VARB jednaki?
  ako da, postavi "1" u VARC inace "0"
  ============================================================
                movlw   D'21'           ; x -> W
                movwf   VARA            ; W -> VARA
                movlw   D'20'           ; y -> W
                movwf   VARB            ; W -> VARB
                nop                     ;
                movf    VARA,W          ; VARA -> W
                subwf   VARB,W          ; VARB - W -> W
                btfsc   STATUS, Z       ; provjeri Z zastavicu
                goto    st2             ; Z = 1, idi na st2
st1             clrf    VARC            ; 0 -> VARC
                goto    kr4             ; idi na kr4 (kraj)
st2             movlw   D'1'            ; 1 -> W
                movwf   VARC            ; W -> VARC
kr4             nop                     ; kr4 (kraj)
  ============================================================
  Broji od VARA do VARB
  ============================================================
                movlw   D'5'            ; postavi 5 u VARA
                movwf   VARA            ; -||-
                movlw   D'12'           ; postavi 12 u VARB
```

```
                movwf      VARB                ; -||-
                nop                      ;
kreni           nop                      ;
                movf       VARA,W              ; VARA -> W
                subwf      VARB,W              ; VARB - W -> W
                btfsc      STATUS,Z            ; rezultat je 0?
                goto       kr5                 ; da, idi na kraj
                nop                            ; ne
                nop                            ;
                incf       VARA,F              ; VARA + 1 -> VARA
                goto       kreni               ; idi na kreni
kr5             nop                            ; kr5 (kraj)
  ============================================================


  ====================================================================
  Za podatke A=100, B=2, C=5, D=10, E=4 Treba izracunati sljedeci
  izraz: Y = A-B*(C-D*E) = 100-2*(5-10*4)
  PRVI NAČIN
  ====================================================================

prim4           nop

REZ             EQU                 0x0c


                movlw       D'10'          ; ------------------
                addlw       D'10'          ; 10 * 4 BLOK
                addlw       D'10'          ;
                addlw       D'10'          ; ------------------

                movwf       REZ                  ; 40 -> REZ
                comf        REZ, F
                incf        REZ, F         ; REZ = -40
                movlw       D'5'
                addwf       REZ, F         ; REZ = 5-40
                movf        REZ, W
                addwf       REZ, F         ; REZ = 2 * (5-40)
                comf        REZ, F
                incf        REZ, F         ; REZ = -2*(5-40)
                movlw       D'100'
                addwf       REZ, F

  ====================================================================
  Za podatke A=100, B=2, C=5, D=10, E=4 Treba izracunati sljedeci
  izraz: Y = A-B*(C-D*E) = 100-2*(5-10*4)
  DRUGI NAČIN
  ====================================================================
prim5  nop

                movlw       D'10'
                addlw       D'10'
                addlw       D'10'
                addlw       D'10'
                movwf       REZ                  ; 40 -> REZ
                movlw       D'5'           ; 5  -> W
                subwf       REZ, F         ; 40-5
                movf        REZ, W
                addwf       REZ, F         ; REZ = 2*(40-5)
                movlw       D'100'
                addwf       REZ, F         ; REZ = 100 + 2 * (40-5)
                end
```

Primjer 2 – Petlje za kašnjenje

```
  ============================================================
  Primjeri 2
  ============================================================

  ============================================================
  PETLJE ZA KAŠNJENJE
  ============================================================

                    PROCESSOR 16F84A
                    #include "p16f84a.inc"
                    ERRORLEVEL -224
                    __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF

  ================================================
  Jednostruka petlja
  Najvece kasnjenje = 768 ciklusa
  768 ciklusa uz Fosc=4 MHz traje 768 us.
  ================================================

                nop                     ; demonstracija izvođenja jednog ciklusa
                nop                     ; demonstracija izvođenja jednog ciklusa
                nop                     ; demonstracija izvođenja jednog ciklusa

;               call      cekanje

COUNT           equ       0x0c          ; definirati konstantu za brojac (registar)
                                        ;
                movlw     D'0'          ; L -> W
                movwf     COUNT         ; W -> COUNT
                nop                     ; 1 Ciklus
petlja          decfsz    COUNT         ; (L-1) * 1 Ciklus + 2 Ciklusa
                goto      petlja        ; (L-1) * 2 Ciklusa
                nop                     ; ====================
                                          3*L Ciklusa
  ================================================
  Dvostruka petlja
  Najvece kasnjenje = 197632 ciklusa
  197632 ciklusa uz Fosc=4 MHz traje 197,632 us
  197632 ciklusa uz Fosc=4 MHz traje ~ 200 ms
  K = 255, L = 255
  ================================================

                nop
cekanje         nop

COUNT1          equ       0xd           ;
COUNT2          equ       0xe           ;
                movlw     D'255'        ;                         K -> W
                movwf     COUNT2        ;                         W -> COUNT2
                nop                     ; 1C
petlja1         movlw     D'255'        ; K * 1C                  L -> W
                movwf     COUNT1        ; K * 1C                  W -> COUNT1
petlja2         decfsz    COUNT1        ; [(L-1) * 1C + 2C] * K
                goto      petlja2       ; [(L-1) * 2C] * K
                decfsz    COUNT2        ; (K-1) * 1C + 2C
                goto      petlja1       ; (K-1) * 2C
                nop                     ;============================
                nop                     ; K*L*3C + K*4C - 1C


  ================================================
  Trostruka petlja za kasnjenje
```

```
  Najvece kasnjenje na Fosc=4MHz je 49.87s
  ================================================

BROJAC1         equ       0x10
BROJAC2         equ       0x11
BROJAC3         equ       0x12

                nop
kasni           movlw     D'0'          ; J
                movwf     BROJAC1
p1              movlw     D'0'          ; K
                movwf     BROJAC2
p2              movlw     D'0'          ; L
                movwf     BROJAC3
p3              decfsz    BROJAC3, F
                goto      p3
                decfsz    BROJAC2, F
                goto      p2
                decfsz    BROJAC1, F
                goto      p1
                nop
                end
```

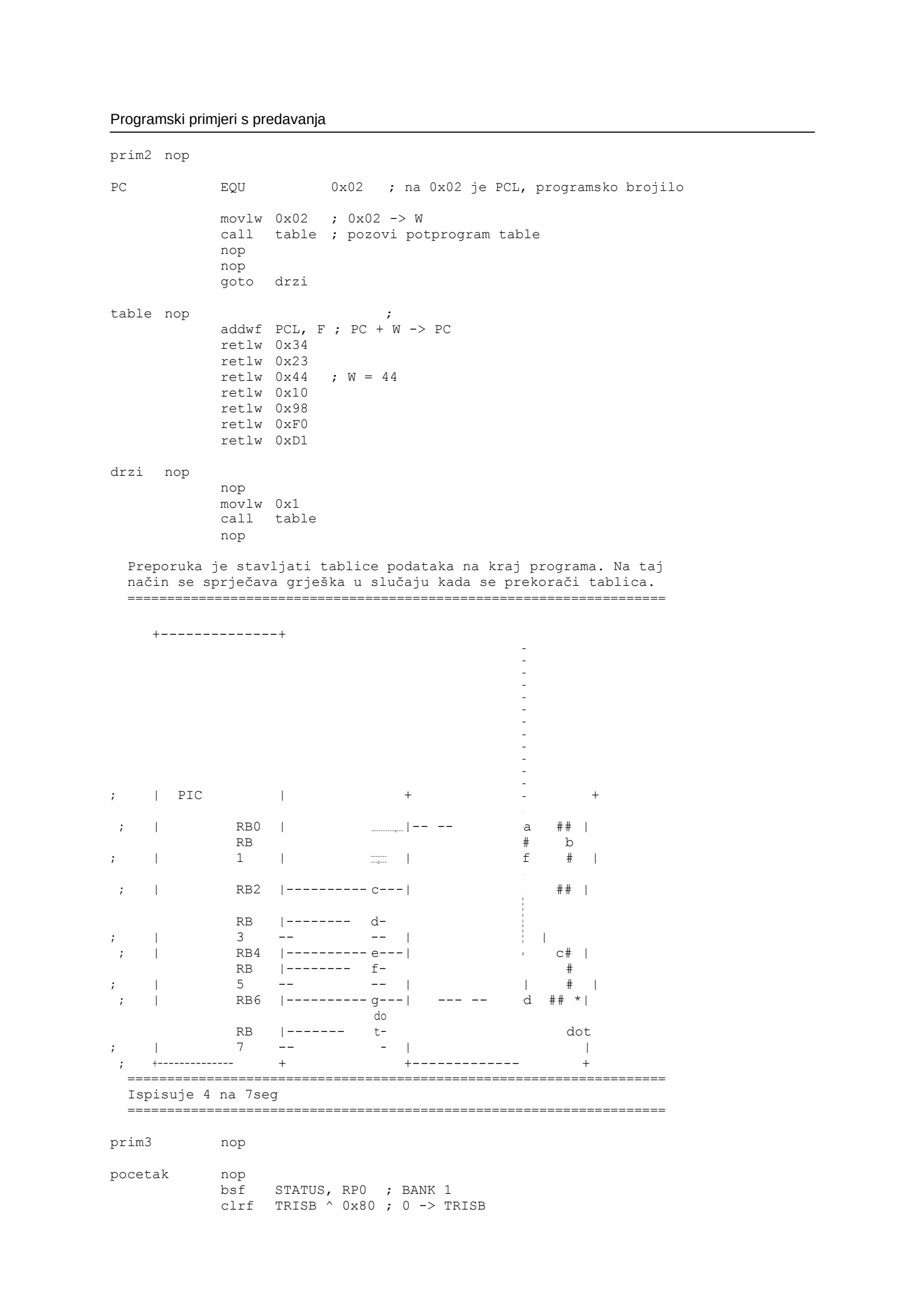
Primjer 3 – Potprogrami, mehanizam poziva potprograma

```
  ============================================================
  Primjeri 3
  ============================================================

  ============================================================
  POTPROGRAMI
  ============================================================
              PROCESSOR 16F84A
              #include "p16f84a.inc"
              ERRORLEVEL -224
              __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF

  ====================================================================
  MACRO
  --------------------------------------------------------------------
  Makro je *niz* instrukcija koje se umeću u izvorni kôd pomoću
  samo *jednog* poziva. Makro je potrebno prvo definirati kako bi
  ga se moglo pozvati u kôdu koji slijedi.
  ====================================================================

BANK0  macro
       nop
       bcf STATUS, RP0 nop

       endm

BANK1  macro
       nop
       bsf STATUS, RP0 nop

       endm

              org          0x0;

;             goto   prim1;
              goto   prim2;
              goto prim3;
              goto   prim4;
 ;            goto   prim5;
  ================================================
  Postavljaju se 4 prva pina PORTB kao izlazni
  a druga 4 pina kao ulazni. Na prva dva pina
  salje se "1"
  Izlazni pin :0, ulazni pin:1
  ================================================
              bsf    STATUS, RP0         ; odabir Bank 1
              movlw  0xF0                ; OxF0 == B'11110000'
              movwf  TRISB ^ 0x80        ; postavi 0xF0 u TRISB

              bcf    STATUS, RP0         ; odabir Bank 0 radi PORTB
              movlw  0x03                ; postavi B'00000011' u W
              movwf  PORTB               ; postavi 0x03 na PORTB
  ================================================
  (Isto kao i gore. Koristi se macro)
  Postavljaju se 4 prva pina PORTB kao izlazni
  a druga 4 pina kao ulazni. Na prva dva pina
  salje se "1"
  Izlazni pin :0, ulazni pin:1
  ================================================
              nop
              BANK1
              movlw  0xF0                ; OxF0 == B'11110000'
```

```
              movwf  TRISB ^ 0x80        ; postavi 0xF0 u TRISB
              nop
              BANK0
              movlw  0x03                ; postavi B'00000011' u W
              movwf  PORTB               ; postavi 0x03 na PORTB

  ============================================
  View -> Disassembly Listing !!!
  View -> Program Memory !!!
  View -> Hardware Stack !!!
  ============================================

  ========================================================================
  POTPROGRAMI
  ------------------------------------------------------------------------
  CALL, RETURN, RETLW
  Cemu sluze i kako rade? Stog (Stack). Na koji nacin se izvrsavaju?
  Goto naredba i potprogram.
;
        |
 ;      \|/            /   |
 ;       |       /   \|/
       call _ _ / potprogram
 ;       |           \       \|/
 ;      \|/            \
        |
  ========================================================================
prim1         nop
              nop
              nop
              ; radi nesto
              call   cekanje
              nop
              nop
              nop
              call   cekanje
              nop

cekanje       nop
              ; odradi cekanje
              nop
              call   cekanje2
              nop
              return

cekanje2      nop
              ; odradi cekanje
              nop
              call   cekanje3
              nop
              return

cekanje3      nop
              ; odradi cekanje
              nop
              nop
              return


  ========================================================================
  TABLICE PODATAKA
  - liste podataka, podaci su smješteni jedan iza drugoga
  - prikladno za prikaz podataka na 7-segmentnom pokazniku.
  - razmotrimo primjer
  ========================================================================
```

```
prim2  nop

PC            EQU          0x02   ; na 0x02 je PCL, programsko brojilo

              movlw  0x02   ; 0x02 -> W
              call   table  ; pozovi potprogram table
              nop
              nop
              goto   drzi

table  nop                          ;
              addwf  PCL, F ; PC + W -> PC
              retlw  0x34
              retlw  0x23
              retlw  0x44   ; W = 44
              retlw  0x10
              retlw  0x98
              retlw  0xF0
              retlw  0xD1

drzi   nop
              nop
              movlw  0x1
              call   table
              nop
```

Preporuka je stavljati tablice podataka na kraj programa. Na taj način se sprječava grješka u slučaju kada se prekorači tablica.



Ispisuje 4 na 7seg

```
prim3         nop

pocetak       nop
              bsf    STATUS, RP0  ; BANK 1
              clrf   TRISB ^ 0x80 ; 0 -> TRISB
```

```
            bcf    STATUS, RP0  ; BANK 0
            clrf   PORTB        ; 0 -> PORTB
glavni      movlw  0x00         ; 0x09 -> W
            call   sedseg       ; idi na sedseg
            movwf  PORTB        ; W -> PORTB
pettlja     goto   pettlja      ; zavrti se tu

sedseg      nop                 ;
            addwf  PCL, F       ; W + PCL -> PCL
            retlw  0x3F         ; 0 0x3F = B'00111111'
            retlw  0x06         ; 1 0x06 = B'00000110'
            retlw  0x5B         ; 2 0x5B = B'01011011'
            retlw  0x4F         ; 3 0x4F = B'01001111'
            retlw  0x66         ; 4 0x66 = B'01100110'
```

```
        retlw  0x6D         ; 5 ...
        retlw  0x7D         ; 6 ..
        retlw  0x07         ; 7 .
        retlw  0x7F         ; 8
        retlw  0x6F         ; 9
        retlw  0x77         ; A
        retlw  0x7C         ; B
        retlw  0x39         ; C
        retlw  0x5E         ; D
        retlw  0x79         ; E
        retlw  0x71         ; F
        retlw  0x80         ; .
```

Primjer 4 – Prekidi

```
  ============================================================
  Primjeri 4
  ============================================================

  ============================================================
  PREKIDI
  ============================================================
              PROCESSOR 16F84A
              #include "p16f84a.inc"
              ERRORLEVEL -224
              __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF
  ================================================================
  Prekid je proces koji zaustavlja mikrokontroler u njegovom radu
  kako bi se obavilo nešto drugo, tj. kako bi neki sklop signalizirao
  procesoru da se s njim nesto zbiva i da je potrebno jedan dio procesorskog
  vremena i njemu dati. (Sklop TIMER0 stvara prekid pri preljevu 0xFF -> 0x00)


       ...
 ;     GLAVNI PROGRAM      --- PREKID se dogodi ==>>--+
 ;     ...                 \                          |
 ;     ...           <-------                         |
 ;     ...           <-----povratak-iz-prekida---------+--+
 ;   PREKIDNA RUTINA                                 + |
 ;     ...                                                                 |
 ;     ...                                                                 |
 ;     RETFIE ==>>-------------------------------------                     +
 ;
  4 izvora prekida; 2 izvora su vanjski prekidi, 2 izvora su unutrasnji
  prekidi (TIMER0 i WDT)
  RB0/INT - može biti izvor vanjskog prekida
  RB4 - RB7 - isto se može koristiti za prekide
  1. Kod koristenja prekida PICu moramo dati do znanja da ćemo koristiti prekid
  2. Odrediti koji pin porta B će se koristiti za prekide
  Registar za kontrolu prekida je INTCON
;
;  7     6       5      4      3     2        1     0
; +-----+-----+------   +------+------+------+------+------+
; | GIE | EEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF |
; +-----+-----+------   +------+------+------+------+------+
; GIE   ::   Global Interrupt Enable bit; 1 = Enable, 0 = Disable
;                   Omogućavanje prekida
  T0IE :: TMR0 Owerflow Interrupt bit; 1 = Enable TMR0 interrupt, 0 = Disable
  INTE :: RB0/INT Interrupt Enable bit; 1 = Enable, 0 = Disable
;                   RB0 pin je odabran kao prekidni ulaz
; RBIE ::    RB Port Change Interrupt bit; 1 = Enable, 0 = Disable
;                   Omogućavanje prekida na promjenu stanja
; T0IF   ::  TMR0 Overflow interrupt flag bit; 1 = Preljev, 0 = nema preljeva
; INTF ::    RB0/INTInterrupt Flag bit
;                   1 kod pojave prekida na RB0/INT pinu
; RBIF   ::  RB Port Change Interrupt Flag; 1 = neki od RB4-RB7 je promijenio stanje
;                   kod promjene stanja na nekom od pinova RB4-RB7
;
; OPTION_REG ::
;  7       6         5       4      3     2     1     0
; +-------   +--------       +------+------+-----+-----+-----+-----+
; | *RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 | PS0 |
; +-------   +--------       +------+------+-----+-----+-----+-----+
; INTEDG::   Interrupt Edge Select bit
;                   1 - prekid na rastući brid na pinu RB0/INT
;                   0 - prekid na padajući brid na pinu RB0/INT
;
; Zastavica prekida RBIF i INTF postavljaju se automatski u "1" kod pojave
```

```
  prekida, te se ne postavljaju automatski u "0" već se to mora riješiti programski

   ================================================
    | Vrijednost PC-a se pri pojavi prekida postavlja|
 ;  | na 0x04 što je adresa gdje počinje prekidna    |
 ;  | servisna rutina                                |
   ================================================

  Iz te servisne rutine vraća se naredbom RETFIE
             org   0x00
             goto  glavni

             org   0x04
             ; prekidna rutina
             nop
             nop
             nop
             retfie


glavni       nop
             nop
             goto  glavni       ; glavni dio programa


  kod pozivanja prekida treba paziti na dvije stvari
  1. U slučaju kada se koriste isti registri u glavnom programu
     i u prekidnoj rutini, vrlo vjerojatno će se sadržaj registara
     promijeniti u prekidnoj rutini, što može dovesti do pogrešnog
     rada glavnog programa.
  2. Postoji odeđeno kašnjenje između dva prekida koji se mogu primijetiti
     Između dva susjedna prekida mogu proći 3 do 4 instrukcijska ciklusa
     (zbog skoka na prekidnu adresu, postavljanja zastvica prekida i povratka
      iz prekidne rutine)
;
  Kako prilikom prekida jedino sadržaj PC-a biva sačuvan, korisnik se sam mora
  pobrinuti da sačuva sve vrijednosti registara koje su mu važni za rad
  programa (npr. W i STATUS). Tako bi servisna rutina morala sadržavati
  sljedeće korake
       a) zapamtiti W registar
       b) zapamtiti STATUS registar
       c) izvršiti kôd servisne rutine
       d) obnoviti STATUS registar
       e) obnoviti W registar
```

Primjer 5 – Modul Timer

```
  ============================================================
  Primjeri 5
  ============================================================

  ============================================================
  TIMER0 Modul
  ============================================================
            PROCESSOR 16F84A
            #include "p16f84a.inc"
            ERRORLEVEL -224
            __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON &
            _WDT_OFF org 0x0
  ====================================================================
  TIMER0 Modul
  Mogucnosti: 8 bitni timer/brojac; TMR0 se moze pisati i citati,
  8 bitno predjelilo, unutarnji ili vanjski podrazaji za brojilo,
; prekid pri preljevu  0xFF -> 0x00, odabir ruba za vanjski podrazaj
;
  Dva načina rada: Timer i Brojač. Nacin rada odabire se pomocu TOCS bita
  (OPTION_REG<5>). Kada predjelilo nije odabrano u timer načinu rada
  TMR0 registar povećava se za 1 svaki ciklus. Brojač način rada povećava
  za jedan na rastući ili padajući brid pina RA4/T0CKI. Rastući/padajući
  brid se odabire itom T0SE (OPTION_REG<4>)
;
  Predjelilo je izvedeno kao 8 bitni brojač. Postoji samo jedno predjelilo
  kojeg Timer0 ili WDT mogu koristiti.
;
       4MHz osc :: MAX kasnjenje: 65.53s
  32.768KHz osc :: MAX kasnjenje: 8s
  Kasnjenje se moze izvesti koristenjem prekidne rutine ili bez
  prekida koristeci T0IF bit INTCON<2> registra. Pri preljevu registra
  TMR0 (0xFF -> 0x00) postavlja se T0IF bit (potrebno ga je 'rucno' obrisati
  u programu) i dolazi do prekida. Prekid se moze maskirati brisanjem T0IE
  bita INTCON<5> registra.
;
              (256-INIT_TMR0) * PREDJELILO
  KASNJENJE = ----------------------------
;                 FREKVENCIJA / 4
;
;                     KASNJENJE * FREKVENCIJA
; INIT_TMR0 = 256 - -------------------------
;                         4 * PREDJELILO
; OPTION_REG ::
;  7       6        5       4      3     2     1     0
; +-------+--------+------      +------+-----+-----+-----+-----+
; | *RBPU | INTEDG | T0CS | T0SE | PSA | PS2 | PS1 |  PS0 |
; +-------+--------+------      +------+-----+-----+-----+-----+
; TOCS    :: TMR0 Clock Source Select Bit; 1 = RA4/T0CKI, 0 = CLKOUT interni clock
; PSA     :: Prescaler assignment; 1 = WDT, 0 = TMR0
; PS2-PS0 :: bitovi predjelila
;
; +-------+--------+------      +------+-----+-----+-----+-----+
; |   X   |   X    |   0  |  X   |  0  |     |     |     |
; +-------+--------+------      +------+-----+-----+-----+-----+
;                                          \    |    /
;                                           \   |   /
; 1:1 se postize bez predjelila               000 1 : 2
; tj. predjelilo za taj slucaj mora biti      001 1 : 4
; pridjeljeno WDT-u                           010 1 : 8
;                                             011 1 : 16
;                                             100 1 : 32
;                                             101 1 : 64
```

```
;                                                   110 1 : 128
;                                                   111 1 : 256
; INTCON ::
;   7     6       5       4       3      2      1      0
  +-----+------+------+------+------+------+------+------+
  | GIE | EEIE | T0IE | INTE | RBIE | T0IF | INTF | RBIF |
  +-----+------+------+------+------+------+------+------+
; GIE   :: Global Interrupt Enable bit; 1 = Enable, 0 = Disable
; T0IE :: TMR0 Owerflow Interrupt bit; 1 = Enable, 0 = Disable
; T0IF  :: TMR0 Overflow interrupt flag bit; 1 = Preljev, 0 = nema preljeva
;
  Timeru se vrijednost poveća za 1 svakih: 1/[(Fosc/4)/predjelilo]


  ====================================================================
  Vrijeme kasnjenja u potprogramu za kasnjenje je 32.8 ms
  128 * 256 us = 32.768
  128 upravo zato jer se "opet" testira da li je TMR0 dosao
  do 128

TMR0   EQU   0x01

             clrf   TMR0                        ; 0->TMR0
             bsf    STATUS, RP0                 ; BANK 1
             movlw  B'11011000'                 ; 0->T0IE, 0->T0IF, predjelilo 256
             movwf  OPTION_REG ^ 0x80           ; W -> OPTION_REG
             bcf    STATUS, RP0                 ; BANK 0
             nop
             nop
             nop
             nop
             movlw  D'253'
             movwf  TMR0
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop
             nop                                ; neki posao se tu obavlja
             call   cekaj                       ; pozovi potprogram za cekanje
             nop
             call   cekaj2
             nop
             call   cekaj2
             nop                                ; opet neki posao

  ====================================================================
  Primjer s direktnom provjerom TMR0 registra
  ====================================================================
                                      Potprogram za cekanje
cekaj  clrf   TMR0                   ; 0->TMR0, ocisti TMR0
             movlw  D'120'
             movwf  TMR0

opet   btfss  TMR0,7                 ; da li je TMR0 = 128?
             goto   opet             ; vrati se na "opet"
             return                  ; povratak iz potprograma

  ====================================================================
  Provjera T0IF zastavice, INTCON<2>
  ====================================================================
```

```
cekaj2 nop
             movlw        D'250'
             movwf        TMR0
             bcf          INTCON, T0IF        ; !!!!!!!!
             bcf          INTCON, T0IF        ; !!!!!!!!
opet2        btfss        INTCON, T0IF
             goto   opet2
             nop
             bcf          INTCON, T0IF        ; !!!!!!!!
             return
             end
```

Primjer 6 – Timer pomoću prekida

```
  ============================================================
  Primjeri 6
  ============================================================

  ============================================================
  PREKIDI + TIMER0 (čin treći)
  ============================================================
            PROCESSOR 16F84A
            #include "p16f84a.inc"
            ERRORLEVEL -224
            __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF
; ====================================================================

TMR0        EQU         0x01        ; TMR0 registar
BROJI       EQU         0x0C        ; BROJI registar

            org         0x00        ; idi na glavni program
            goto        program     ; kako bi se preskocila prekidna rutina

            org         0x04        ; prekidna rutina
            bcf         STATUS, RP0         ; BANK0
            bcf         INTCON, T0IF        ; ocisti zastavicu prekida
            clrf        TMR0                ; 0->TMR0, ocisti TMR0 iako je cist
            incf        BROJI, F            ; povecaj broji
            retfie                          ; vrati se iz prekida

  ====================================================================
  Postavljanje TIMER0 sklopa
  ====================================================================
program     nop
            bsf         STATUS,  RP0        ; BANK1
            movlw        B'11010000'        ; 0->T0IE, 0->T0IF, predjelilo 2
            movwf       OPTION_REG ^ 0x80   ; W -> OPTION_REG
            bcf         STATUS,  RP0        ; BANK0
  ====================================================================
  Postavljanje prekida
  ====================================================================
            bcf         STATUS, RP0         ; BANK0
            bsf         INTCON, GIE         ; omoguci prekide
            bsf         INTCON, T0IE        ; TIMER0 Interrupt Enable, omoguci
                                            ; prekid pri preljevu
            bcf         INTCON, INTE        ; onemoguci prekida na RB0/INT
            bcf         INTCON, INTF        ; ocisti zastavicu prekida
            bcf         INTCON, T0IF        ; ocisti zastavicu T0IF
  ====================================================================
  Glavni dio programa
  ====================================================================

            clrf   BROJI                    ; ocisti pregistar broji
            movlw  D'200'
            movwf  TMR0
opet        nop
            nop
            nop
            goto   opet                     ; idi na opet, idi na opet, idi ...
            end
```

Primjer 7 – Rad s 16 bitnim podacima

```
  ============================================================
  Primjeri 7
  ============================================================

  ============================================================
  Aritmeticke operacije, STATUS registar
  ============================================================
            PROCESSOR 16F84A
            #include "p16f84a.inc"
            ERRORLEVEL -224
            __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF

  ====================================================================
  Napisati program koji će zbrojiti dva heksadecimalna broja:
  0xFF i 0x01. Rezultat prikazati na LED diodama koje su spojene
  na PORTB. Status zastavice C, DC i Z prikazati na LED diodama koje su
  spojene na pinovima RA0, RA1 i RA2
  ====================================================================
  0xFF + 0x01 = 0x0; Z=1, DC=1, C=1
;
    11111111 255
   +00000001 + 1
    -------- ----
   100000000 256
   |
   |-> C = 1; Prekoračenje opsega broja

  Status registar
  +-----+-----+-----+-----+-----+-----+-----+-----+
; |     |     |     |     |     |  Z  |  DC |  C  |
; +-----+-----+-----+-----+-----+-----+-----+-----+
;
            org 0x0
            nop
            bsf          STATUS, RP0         ; bank1
            clrf   TRISA ^ 0x80 ; Svi pinovi PORTA izlazni
            clrf   TRISB ^ 0x80 ; Svi pinovi PORTB izlazni
            bcf          STATUS, RP0         ; bank0
            clrf   PORTA               ; ocisti PORTA
            clrf   PORTB               ; ocisti PORTB
            bcf          STATUS, C           ; ocisti C zastavicu
            bcf          STATUS, DC          ; ocisti DC zastavicu
            bcf    STATUS, Z           ; ocisti Z zastavicu
            movlw  0xFF          ; 0xFF -> W
            addlw  0x01          ; + 0x01 -> W
            nop                              ;
  ====================================================================
  Primijetiti promjenu zastavicu Z,DC i C STATUS registra !!!!
  ====================================================================
            nop                              ;
            movwf  PORTB         ; PORTB
            movf   STATUS, W     ; STATUS -> W
            movwf  PORTA         ; W -> PORTA
            nop                  ;

  ====================================================================
  16-bitno zbrajanje u 8-bitnom mikrokontroleru
  ====================================================================
  Najprije se zbroje niži bajtovi, onda se zbroje viši bajtovi i CARRY
  bit ako se pojavio kod zbrajanja nižih bajtova.
  Primjer 1. :: 0x2056 + 0x12EC = 0x3342
;
```

```
 ;
 ;    56         20
 ;   +EC        +12
 ;  ----    |--> 01
 ;  142     |   ----
 ;  |       |    33
    |------
```

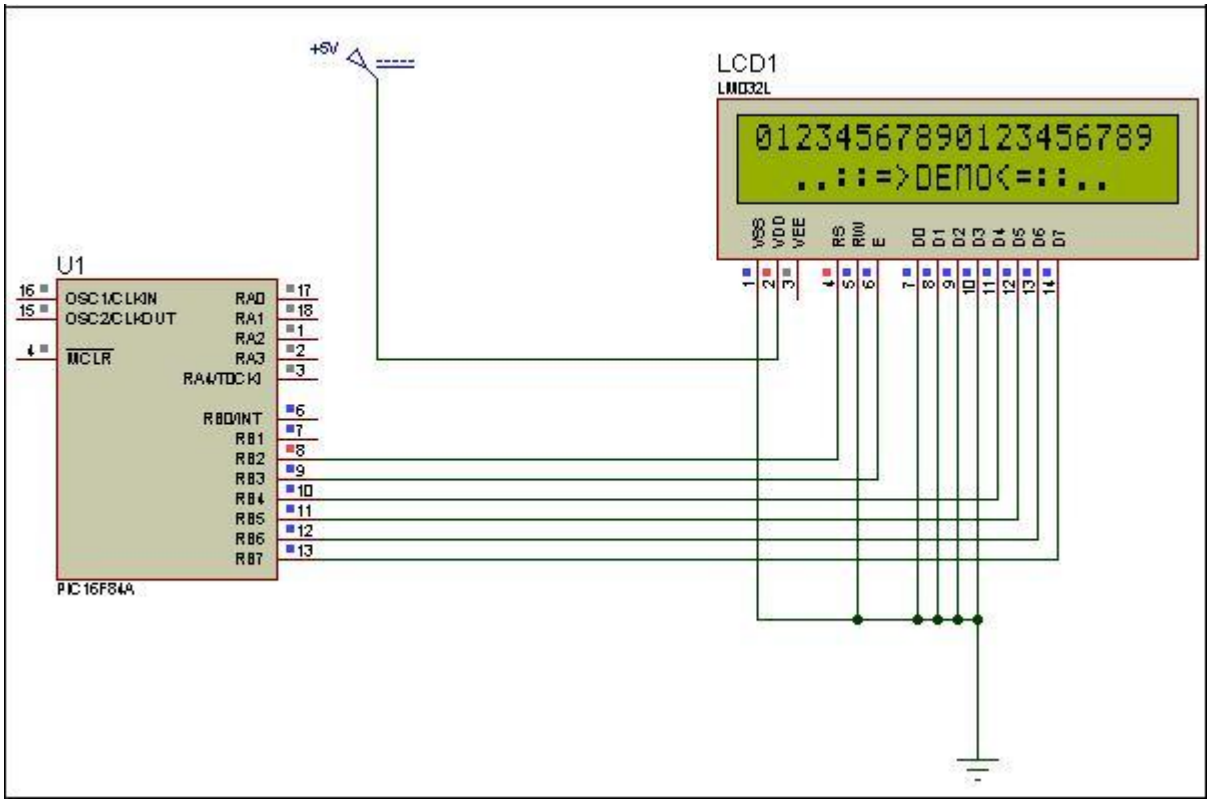
Primjer 2. :: 0xFFFF + 0xFFFF = 0x1FFFF

```
 ;    FF          FF          00
 ;   +FF          FF          00
 ;  ----   |-- > +01   |-- > +01
 ;  1FE    |    ----   |    ----
 ;  |      |    1FF    |      01
 ;  |------     |------
```

```
  ====================================================================
  Napisati program koji će zbrojiti dva 16-bitna broja : 0xA98B i
  0x7C48. Bitove 0 do 7 prikazati na PORTB koristeći LED diode a
  pritiskom na tipalo spojeno na pin RA4 prikazati bitove 8 do 15 na
  PORTB. te 16 i 17 bit na pinovima RA0 i RA1. Najveći mogući rezultat
  je 0x01FFFF (18 bitova)
  ====================================================================
  0xA98B + 0x7C48 = 0x125D3
;
      org 0x20
XL    EQU   0x0C
XH    EQU   0x0D
YL    EQU   0x0E
YH    EQU   0x0F

RESULTL      EQU   0x10
RESULTH      EQU   0x11
RESULTM      EQU   0x12
TCARRY EQU   0x13

             bsf   STATUS, RP0         ; BANK 1
             clrf  TRISB ^ 0x80        ; PORTB svi pinovi izlazni
             movlw 0x10                ; RA4 ulazni, ostali izlazni
             movwf TRISA ^ 0x80        ; W -> TRISA
             bcf   STATUS, RP0         ; BANK 0
             clrf  PORTB               ; ocisti PORTB
             clrf  PORTA               ; ocisti PORTA
             clrf  TCARRY              ; ocisti TCARRY
             clrf  RESULTM             ; ocisti RESULTM

             movlw 0xA9
             movwf XH                  ; 0xA9 -> XH
             movlw 0x8B
             movwf XL                  ; 0x8B -> XL
             movlw 0x7C
             movwf YH                  ; 0x7C -> YH
             movlw 0x48
             movwf YL                  ; 0x48 -> YL

ADD          nop                       ; Zbroji LOW dijelove
             movf  XL, W               ; XL -> W
             addwf YL, W               ; W + YL = W
             movwf RESULTL             ; XL + YL = RESULTL
             btfsc STATUS, 0           ; CARRY?
             incf  TCARRY              ; postoji CARRY

                                       ; Zbroji HIGH dijelove
             movf  XH, W               ; XH -> W
             addwf YH, W               ; W + YH = W
```

```
                movwf  RESULTH              ; XH + YH = RESULTH
                btfsc  STATUS, 0            ; CARRY?
                incf   RESULTM              ; postoji CARRY

                                            ; Zbroji TCARRY i RESULTH
                movf   TCARRY, W            ; TCARRY -> W
                addwf  RESULTH, F           ; RESULTH + CARRY -> RESULTH
                btfsc  STATUS, 0
                incf   RESULTM              ; Postoji CARRY

LOBYTE          nop
                movf   RESULTL, W
                movwf  PORTB

TIPKA           btfsc  PORTA, 4
                goto   TIPKA

HIBYTE          movf   RESULTH, 0
                movwf  PORTB
                movf   RESULTM, 0
                movwf  PORTA

kraj            goto   kraj

                end
```

Primjer 8 – Upravljanje s LCD zaslonom HD44780



Shema spajanja LCD pokaznika

```
  ============================================================
  Upravljanje s LCD zaslonom HD44780 pomocu PIC16F84A
  ============================================================
            PROCESSOR 16F84
            #include "p16f84a.inc"
            ERRORLEVEL -224
            __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF
  ================================================
  Definicija memorijskih lokacija
  ================================================
BROJAC      EQU 0x0C
CHAR        EQU 0x0D
TEMP        EQU 0x0E
COUNT1      EQU 0x10            ; prvi (vanjski) brojac
COUNT2      EQU   0x11          ; drugi (unutarnji) brojac
  ================================================
  RS i E linije prema LCD modulu
  Spojene su na RB2(RS) i RB3(E)
  ================================================
RS          EQU 2
E           EQU 3
  ================================================
  Instruction set za Hitachi HD44780
  (http://www.doc.ic.ac.uk/~ih/doc/lcd/instruct.html)
  Ovo su kontrolni okteti za LCD tj. naredbe za LCD
  kojima se određuje način njegovor rada
  ================================================
INIT4BIT1   EQU B'00110011'
INIT4BIT2   EQU B'00110010'

CLEAR       EQU B'00000001'           ; Brise ekran
RETHOME     EQU B'00000010'           ; Ispisivanje se vraca na pocetak
ENTRYMODE   EQU B'00000110'           ; Nacin rada. Inc/Dec = 1, Shift = 0
DISPONOFF   EQU B'00001110'           ; Display ON/OFF, Disp=1, Curs=0, Blink=1
;FUNCSET    EQU B'00101000'           ; Function set, DataLength=0, NF=1*
FUNCSET           EQU B'00101100'     ; Function set, DataLength=0, NF=1*

                  org 0x00
  ================================================
  Postavljanje portova
  ================================================
            bsf         STATUS, RP0  ; banka 1
            clrf        TRISB ^ 0x80 ; PORTB je izlazni
            bcf         STATUS, RP0  ; banka 0
            clrf        PORTB        ; ocisti PORTB
            clrf        BROJAC       ; ocisti BROJAC
  ================================================
  Inicijalizacija LCD-a
  ================================================
            call  cek15
            call  LCDINIT            ; Inicijalizacija LCD-a
  ================================================
  Glavni dio programa
  ================================================
            movlw       0x0    ; 0 u W, uzmi 0. znak iz tablice za pocetak
glavni1     movwf       BROJAC ; Sačuvaj sadržaj W-registra (brojac)
            call  string1      ; Uzima znak iz tablice (string1)
            andlw 0xFF         ; 0 oznacava kraj znakovnog niza
            btfsc STATUS, Z    ; ako je 0 niz je ispisan do kraja
            goto  dalje        ; ako je 0 idi na kraj
            call  SNDCH        ; salje znak na LCD
            call  cek1         ; kasnjenje
L1          movf  BROJAC,W     ; Sadrzaj brojaca u W registar
            addlw D'1'         ; Uvecaj ga za 1
            goto  glavni1      ; Idi na pocetak glavnog programa
```

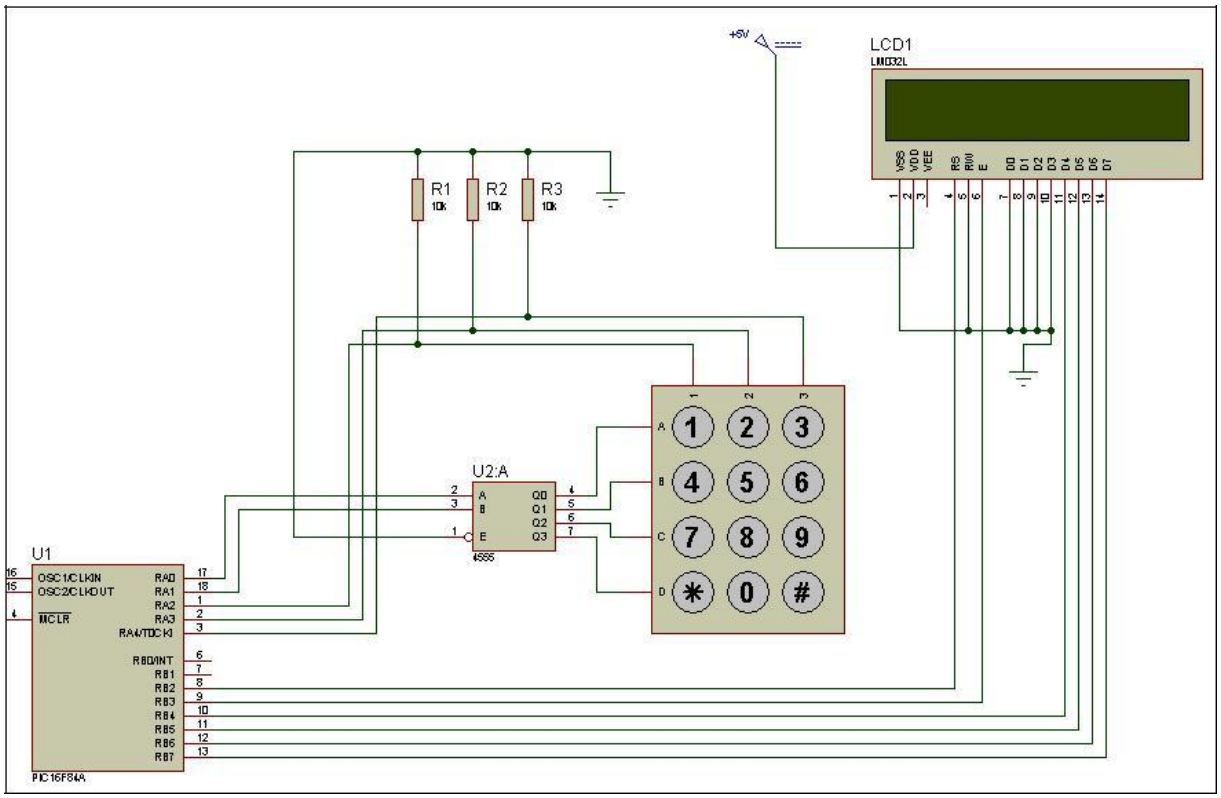
```
                                ;
dalje           nop             ;
                movlw  0xC0     ; naredba za prelezak u drugi red
                call   SNDCMD   ;
                                ;
                movlw       0x0 ; 0 u W, uzmi 0. znak iz tablice za pocetak
glavni2         movwf  BROJAC   ; Sačuvaj sadržaj W-registra (brojac)
                call   string2  ; Uzima znak iz tablice (string2)
                andlw  0xFF     ; 0 oznacava kraj znakovnog niza
                btfsc  STATUS, Z ; ako je 0 niz je ispisan do kraja
                goto   kraj     ; ako je 0 idi na kraj
                call   SNDCH    ; salje znak na LCD
                call   cek1     ; kasnjenje
L11             movf   BROJAC,W ; Sadrzaj brojaca u W registar
                addlw  D'1'     ; Uvecaj ga za 1
                goto   glavni2  ; Idi na pocetak glavnog programa
                                ;
kraj            nop             ;
                goto   kraj     ;
                                ;
                                ; Podprogram za slanje znakova na LCD (RS=1)
                                ; Znak koji se salje na LCD nalazi se u W registru
SNDCH           nop             ;
                movwf  CHAR     ; W -> CHAR
                call   cek1     ; cek1
                call   cek1     ; cek1
                call   cek1     ; cek1
                movf   CHAR, W  ; CHAR -> W
                andlw  0xF0     ; od W uzmi samo gornja 4 bita, ostalo na 0
                movwf  PORTB    ; Postavi gornja 4 bita W na PORTB (RB4-RB7 = D4-D7)
                bsf    PORTB,RS ; Odaberi podatkovni registar
                bsf    PORTB,E  ; Postavi ENABLE signal
                nop
                bcf    PORTB,E  ; ocisti ENABLE signal
                swapf  CHAR, W  ; Zamjeni gornja i donja 4 bita CHAR-a rezultat u W
                andlw  0xF0     ; od W uzmi samo gornja 4 bita, ostalo na 0
                movwf  PORTB    ; Postavi donja 4 bita podatka za LCD na RB4-RB7
                bsf    PORTB,RS ; Odaberi podatkvoni registar
                bsf    PORTB,E  ; Postavi ENABLE signal
                nop
      bcf       PORTB,E         ; Ocisti ENABLE signal
                return          ; Povratak iz potprograma

SNDCMD          movwf  CHAR     ; W -> CHAR
                call   cek1     ; pozovi cek1
                call   cek1     ; pozovi cek1
                call   cek1     ; pozovi cek1
                movf   CHAR,W           ; CHAR -> W
                andlw  0xF0     ; od W uzmi samo gornja 4 bita, ostalo na 0
                movwf  PORTB    ; postavi gornja 4 bita na RB4-RB7 = D4-D7 LCDa
                bcf    PORTB,RS ; odaberi instrukcijski registar LCDa
                bsf    PORTB,E  ; postavi ENABLE signal
                nop
                bcf    PORTB,E  ; ocisti ENABLE signal
                call   cek15
                swapf  CHAR,W   ; zamijeni gornja i donja 4 bita CHAR, rezultat u W
                andlw  0xF0     ; od W uzmi samo gornja 4 bita, ostalo na 0
                movwf  PORTB    ; postavi donja 4 bita naredbe za LCD na RB4-RB7
                bcf    PORTB,RS ; odaberi instrukcijski registar
                bsf    PORTB,E  ; postavi ENABLE signal
                nop
                bcf    PORTB,E  ; ocisti ENABLE signal
                call   cek15
                nop
                return
```

```
LCDINIT      nop
             goto   init1

             bcf          PORTB, 7
             bcf          PORTB, 6
             bsf          PORTB, 5
             bsf          PORTB, 4
             bsf          PORTB, E
             bcf          PORTB, E
             call         cek15

             bsf          PORTB, E
             bcf          PORTB, E
             call         cek15

             bsf          PORTB, E
             bcf          PORTB, E
             call         cek15

             bcf          PORTB, 7
             bcf          PORTB, 6
             bsf          PORTB, 5
             bcf          PORTB, 4
             bsf          PORTB, E
             bcf          PORTB, E
             call   cek15

             goto   init2

init1        nop
             movlw  INIT4BIT1
             call   SNDCMD

             movlw  INIT4BIT2
             call   SNDCMD

init2        nop

             movlw  FUNCSET      ; Function set, DataLength=0, NF=1*
             call   SNDCMD

             movlw  DISPONOFF    ; Display ON/OFF, Disp=1, Curs=0, Blink=1
             call   SNDCMD

             movlw  CLEAR        ; Brise ekran
             call   SNDCMD

             movlw  ENTRYMODE    ; Nacin rada. Inc/Dec = 1, Shift = 0
             call   SNDCMD
             return

  ============================================================
  Kašnjenje od 1ms (Fosc = 4MHz)
  ============================================================
cek1         nop
             goto   cek15
             nop
             movlw  0xFF
             movwf  TEMP
L2           nop
             decfsz       TEMP,F
             goto   L2
             return
```

```
  ============================================================
  Kašnjenje od 15ms (Fosc = 4MHz) K=50, L=100
  ============================================================
cek15  nop
             movlw  D'15'        ;            K -> W
             movwf  COUNT2       ;            W -> COUNT2
             nop                 ; 1C
petlja1      movlw  D'100'       ; K * 1C     L -> W
             movwf  COUNT1       ; K * 1C     W -> COUNT1
petlja2      decfsz COUNT1       ; [(L-1) * 1C + 2C] * K
             goto   petlja2      ; [(L-1) * 2C] * K
             decfsz COUNT2       ; (K-1) * 1C + 2C
             goto   petlja1      ; (K-1) * 2C
             nop                 ;============================
             nop                 ; K*L*3C + K*4C - 1C
             return

  ============================================================
  Znakovni niz za prvi red LCD-a
  ============================================================
string1      addwf  PCL, F
             retlw  '0'
             retlw  '1'
             retlw  '2'
             retlw  '3'
             retlw  '4'
             retlw  '5'
             retlw  '6'
             retlw  '7'
             retlw  '8'
             retlw  '9'
             retlw  '0'
             retlw  '1'
             retlw  '2'
             retlw  '3'
             retlw  '4'
             retlw  '5'
             retlw  '6'
             retlw  '7'
             retlw  '8'
             retlw  '9'
             retlw  0x0

  ============================================================
  Znakovni niz za drugi red LCD-a
  ============================================================
string2      addwf  PCL, F
             retlw  ' '
             retlw  ' '
             retlw  '.'
             retlw  '.'
             retlw  ':'
             retlw  ':'
             retlw  '='
             retlw  '>'
             retlw  'D'
             retlw  'E'
             retlw  'M'
             retlw  'O'
             retlw  '<'
             retlw  '='
             retlw  ':'
             retlw  ':'
             retlw  '.'
             retlw  '.'
```

```
            retlw   ' '
            retlw   ' '
            retlw   0x0         ; Kraj poruke

            end
```

Primjer 9 – Upravljanje tipkovnicom i LCD zaslonom



Shema spajanja tipkovnice i LCD pokaznika

```
  ============================================================
  Upravljanje s tipkovnicom i LCD zaslonom pomocu PIC16F84A
  ============================================================
            PROCESSOR 16F84
            #include "p16f84a.inc"
            ERRORLEVEL -224
            __CONFIG _CP_OFF & _XT_OSC & _PWRTE_ON & _WDT_OFF
  ================================================
  Definicija memorijskih lokacija
  ================================================
BROJAC      EQU 0x0C
CHAR        EQU 0x0D
TEMP        EQU 0x0E
COUNT1      EQU 0x10            ; prvi (vanjski) brojac
COUNT2      EQU   0x11          ; drugi (unutarnji) brojac
  ================================================
  RS i E linije prema LCD modulu
  Spojene su na RB2(RS) i RB3(E)
  ================================================
RS          EQU 2
E           EQU 3
  ================================================
  Instruction set za Hitachi HD44780
  (http://www.doc.ic.ac.uk/~ih/doc/lcd/instruct.html)
```

```
  Ovo su kontrolni okteti za LCD tj. naredbe za LCD
  koje mu govore na koji nacin da radi
  ================================================
INIT4BIT1     EQU B'00110011'           ; Init 1
INIT4BIT2     EQU B'00110010'           ; Init 2
CLEAR         EQU B'00000001'           ; Brise ekran
RETHOME       EQU B'00000010'           ; Ispisivanje se vraca na pocetak
ENTRYMODE     EQU B'00000110'           ; Nacin rada. Inc/Dec = 1, Shift = 0
DISPONOFF     EQU B'00001110'           ; Display ON/OFF, Disp, Curs, Blink
FUNCSET       EQU B'00101100'           ; Function set, DataLength, NF

              org 0x00
  ================================================
  Postavljanje portova
  ================================================
              bsf    STATUS, RP0  ; banka 1
              clrf   TRISB ^ 0x80 ; PORTB je izlazni
              movlw  b'11100'     ; RA0-RB1 izlazi, RA2-RB4 izlazi
              movwf  TRISA        ; RA0-RB1 izlazi, RA2-RB4 izlazi
              bcf    STATUS, RP0  ; banka 0
              clrf   PORTB        ; ocisti PORTB
              clrf   BROJAC       ; ocisti BROJAC
  ================================================
  Inicijalizacija LCD-a
  ================================================
              call   cek5ms
              call   LCDINIT      ; Inicijalizacija LCD-a
  ================================================
  Glavni dio programa
  ================================================

  ============================================================
; test 1. reda      1.segment - provjera tipkala tipkovnice
; ============================================================
              movlw  0x00
              movwf  BROJAC
pocetak       clrf   PORTA        ; niska naponska razina na RA0,RA1      - red 1 HIGH
              btfsc  PORTA, 2     ; Provjera tipkala na RA2               - stupac 1
              goto   tipka1       ;
              btfsc  PORTA, 3     ; Provjera tipkala na RA3               - stupac 2
              goto   tipka2       ;
              btfsc  PORTA, 4     ; Provjera tipkala na RA4               - stupac 3
              goto   tipka3       ;

  ============================================================
  test 2. reda
  ============================================================
              bsf    PORTA, 0     ; visoka naponska razina na RB0         - red 2 HIGH
              btfsc  PORTA, 2     ; Provjera tipkala na RB5               - stupac 1
              goto   tipka4       ;
              btfsc  PORTA, 3     ; Provjera tipkala na RB6               - stupac 2
              goto   tipka5       ;
              btfsc  PORTA, 4     ; Provjera tipkala na RB7               - stupac 3
              goto   tipka6       ;
              bcf    PORTA, 0     ;

  ============================================================
  test 3. reda
  ============================================================
              bsf    PORTA, 1     ; visoka naponska razina na RB2         - red 3 HIGH
              btfsc  PORTA, 2     ; Provjera tipkala na RB5               - stupac 1
              goto   tipka7       ;
              btfsc  PORTA, 3     ; Provjera tipkala na RB6               - stupac 2
              goto   tipka8       ;
              btfsc  PORTA, 4     ; Provjera tipkala na RB7               - stupac 3
```

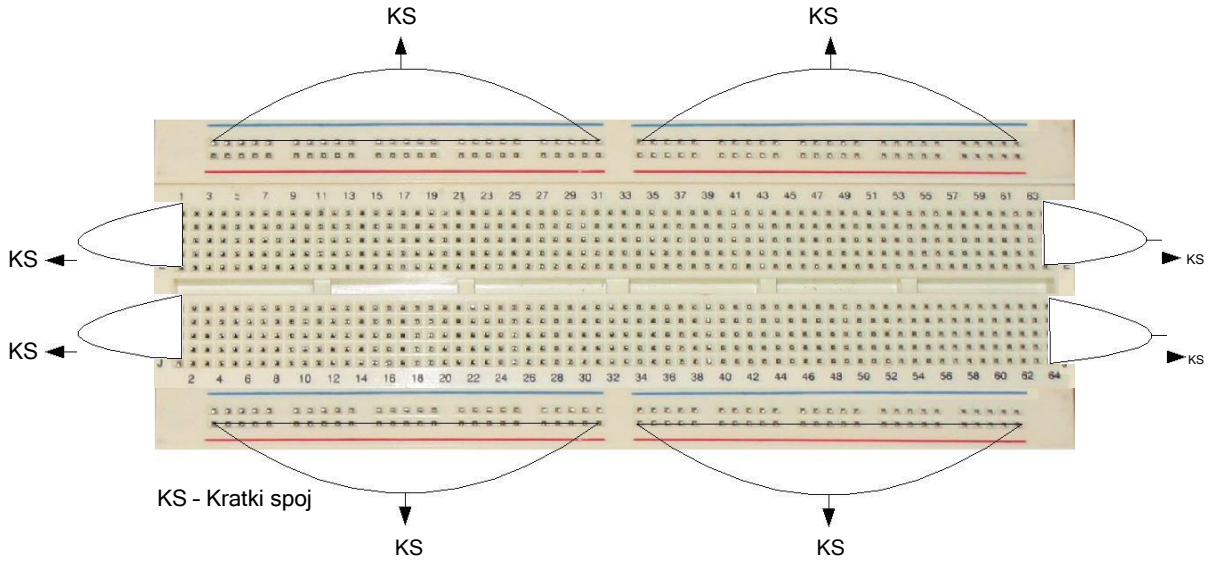
```
            goto   tipka9       ;

  ============================================================
  test 4. reda
  ============================================================
            bsf    PORTA, 0     ; visoka naponska razina na RB3        - red 4 HIGH
            btfsc  PORTA, 2     ; Provjera tipkala na RB5              - stupac 1
            goto   tipkaA       ;
            btfsc  PORTA, 3     ; Provjera tipkala na RB6              - stupac 2
            goto   tipka0       ;
            btfsc  PORTA, 4     ; Provjera tipkala na RB7              - stupac 3
            goto   tipkaB       ;
            goto   pocetak      ; vrati se na pocetak - vrti se u krug

  ============================================================
  2. segment - provjera tipkala tipkovnice
  ============================================================
tipka1      movlw  0x01
            btfsc  PORTA, 2
            goto   tipka1
            goto   glavni1

tipka2      movlw  0x02
            btfsc  PORTA, 3
            goto   tipka2
            goto   glavni1

tipka3      movlw  0x03
            btfsc  PORTA, 4
            goto   tipka3
            goto   glavni1

tipka4      movlw  0x04
            btfsc  PORTA, 2
            goto   tipka4
            goto   glavni1

tipka5      movlw  0x05
            btfsc  PORTA, 3
            goto   tipka5
            goto   glavni1

tipka6      movlw  0x06
            btfsc  PORTA, 4
            goto   tipka6
            goto   glavni1

tipka7      movlw  0x07
            btfsc  PORTA, 2
            goto   tipka7
            goto   glavni1

tipka8      movlw  0x08
            btfsc  PORTA, 3
            goto   tipka8
            goto   glavni1

tipka9      movlw  0x09
            btfsc  PORTA, 4
            goto   tipka9
            goto   glavni1

tipkaA      movlw  0x0A
            btfsc  PORTA, 2
            goto   tipkaA
```

```
            goto   glavni1

tipka0      movlw  0x00
            btfsc  PORTA, 3
            goto   tipka0
            goto   glavni1

tipkaB      movlw  0x0B
            btfsc  PORTA, 4
            goto   tipkaB
            goto   glavni1


glavni1     movwf  BROJAC       ; Sačuvaj sadržaj W-registra (brojac)
            call   string1      ; Uzima znak iz tablice (string1)
            andlw  0xFF         ; 0 oznacava kraj znakovnog niza
            btfsc  STATUS, Z    ; ako je 0 niz je ispisan do kraja
            goto   dalje        ; ako je 0 idi na kraj
            call   SNDCH        ; salje znak na LCD
            call   cek1         ; kasnjenje
L1          movf   BROJAC,W     ; Sadrzaj brojaca u W registar
            addlw  D'1'         ; Uvecaj ga za 1
            goto   pocetak      ; Idi na pocetak glavnog programa
                                ;
dalje       nop                 ;
            movlw  0xC0         ; naredba za prelezak u drugi red
            call   SNDCMD       ;
                                ;
            movlw        0x0    ; 0 u W, uzmi 0. znak iz tablice za pocetak
glavni2     movwf  BROJAC       ; Sačuvaj sadržaj W-registra (brojac)
            call   string2      ; Uzima znak iz tablice (string2)
            andlw  0xFF         ; 0 oznacava kraj znakovnog niza
            btfsc  STATUS, Z    ; ako je 0 niz je ispisan do kraja
            goto   kraj         ; ako je 0 idi na kraj
            call   SNDCH        ; salje znak na LCD
            call   cek1         ; kasnjenje
L11         movf   BROJAC,W     ; Sadrzaj brojaca u W registar
            addlw  D'1'         ; Uvecaj ga za 1
            goto   glavni2      ; Idi na pocetak glavnog programa
                                ;
kraj        nop                 ;
            goto   kraj         ;
                                ;
                                ; Podprogram za slanje znakova na LCD (RS=1)
                                ; Znak koji se salje na LCD nalazi se u W registru
SNDCH       nop                 ;
            movwf  CHAR         ; W -> CHAR
            call   cek1         ; cek1
            call   cek1         ; cek1
            call   cek1         ; cek1
            movf   CHAR,W       ; CHAR -> W
            andlw  0xF0         ; od W uzmi samo gornja 4 bita, ostalo na 0
            movwf  PORTB        ; Postavi gornja 4 bita W na PORTB (RB4-RB7 = D4-D7)
            bsf    PORTB,RS     ; Odaberi podatkovni registar
            bsf    PORTB,E      ; Postavi ENABLE signal
            nop
            bcf    PORTB,E      ; ocisti ENABLE signal
            swapf  CHAR,W       ; Zamjeni gornja i donja 4 bita CHAR-a i rez. u W
            andlw  0xF0         ; od W uzmi samo gornja 4 bita, ostalo na 0
            movwf  PORTB        ; Postavi donja 4 bita podatka za LCD na RB4-RB7
            bsf    PORTB,RS     ; Odaberi podatkvoni registar
            bsf    PORTB,E      ; Postavi ENABLE signal
            nop
            bcf    PORTB,E      ; Ocisti ENABLE signal
            return              ; Povratak iz potprograma
```

```
SNDCMD      movwf  CHAR         ; W -> CHAR
            call   cek1         ; pozovi cek1
            call   cek1         ; pozovi cek1
            call   cek1         ; pozovi cek1
            movf   CHAR,W       ; CHAR -> W
            andlw  0xF0         ; od W uzmi samo gornja 4 bita, ostalo na 0
            movwf  PORTB        ; postavi gornja 4 bita na RB4-RB7 = D4-D7 LCDa
            bcf    PORTB,RS     ; odaberi instrukcijski registar LCDa
            bsf    PORTB,E      ; postavi ENABLE signal
            nop                 ;
            bcf    PORTB,E      ; ocisti ENABLE signal
            call   cek5ms       ;
            swapf  CHAR,W       ; zamijeni gornja i donja 4 bita CHAR, rezultat u W
            andlw  0xF0         ; od W uzmi samo gornja 4 bita, ostalo na 0
            movwf  PORTB        ; postavi donja 4 bita naredbe za LCD na RB4-RB7
            bcf    PORTB,RS     ; odaberi instrukcijski registar
            bsf    PORTB,E      ; postavi ENABLE signal
            nop                 ;
            bcf    PORTB,E      ; ocisti ENABLE signal
            call   cek5ms
            nop
            return

LCDINIT     nop
            movlw  INIT4BIT1
            call   SNDCMD

            movlw  INIT4BIT2
            call   SNDCMD

            movlw  FUNCSET      ; Function set
            call   SNDCMD

            movlw  DISPONOFF    ; Display ON/OFF
            call   SNDCMD

            movlw  CLEAR        ; Brise ekran
            call   SNDCMD

            movlw  ENTRYMODE    ; Nacin rada.
            call   SNDCMD
            return

  ============================================================
  Kašnjenje od 1ms (Fosc = 4MHz)
  ============================================================
cek1        nop
            goto   cek5ms
            nop
            movlw  0xFF
            movwf  TEMP
L2          nop
            decfsz TEMP,F
            goto   L2
            return

  ============================================================
  Kašnjenje od 5ms (~4.5 ms) (Fosc = 4MHz) K=15, L=100
  ============================================================
cek5ms nop
            movlw  D'15'        ;            K -> W
            movwf  COUNT2       ;            W -> COUNT2
            nop                 ; 1C
```

```
petlja1      movlw  D'100'       ; K * 1C            L -> W
             movwf  COUNT1       ; K * 1C            W -> COUNT1
petlja2      decfsz COUNT1       ; [(L-1) * 1C + 2C] * K
             goto   petlja2      ; [(L-1) * 2C] * K
             decfsz COUNT2       ; (K-1) * 1C + 2C
             goto   petlja1      ; (K-1) * 2C
             nop                 ;============================
             nop                  ; K*L*3C + K*4C - 1C
             return
```

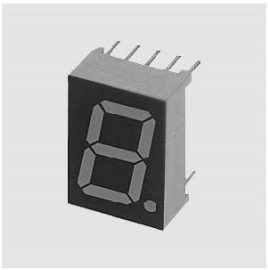
Dodatak A. LABORATORIJSKA OPREMA

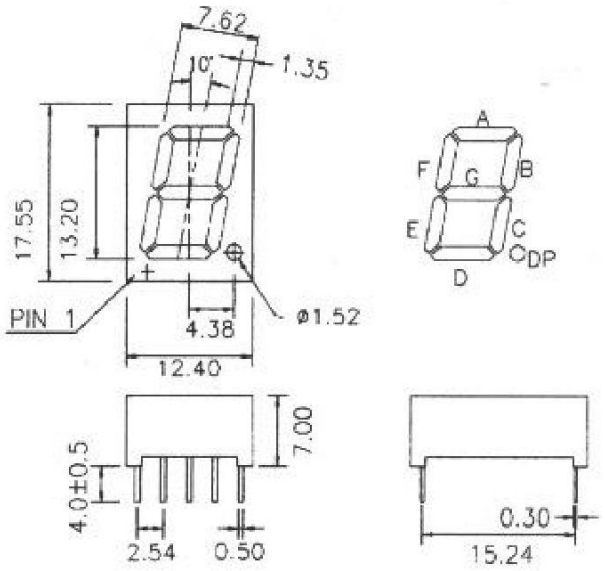


Sl.A.1. *Laboratorijski panel.*

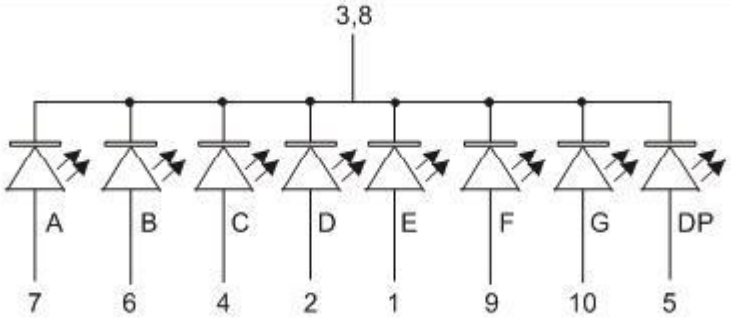
7-segmentni pokaznici S547AP

Služe za prikaz podataka iz mikrokontrolera.





Sl.A.2 *7-segmentni pokaznik.*



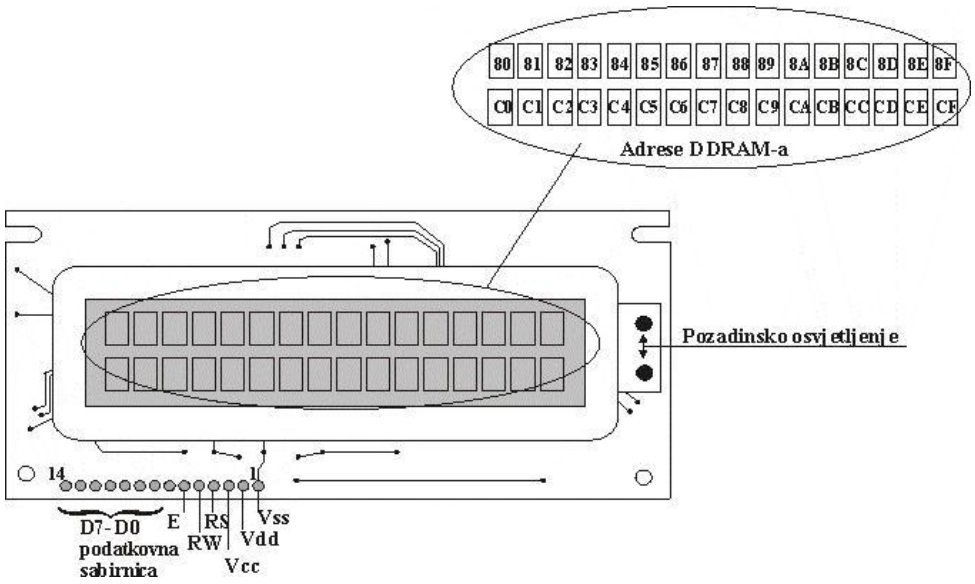
Sl.A.3. *Raspored pinova pokaznika.*

Pin br.	Spoj
1	Anoda E
2	Anoda D
3	Zajednička katoda
4	Anoda C
5	Anoda DP
6	Anoda B
7	Anoda A
8	Zajednička katoda
9	Anoda F
10	Anoda G

Tab.A.1. Značenja pinova pokaznika.

LCD zaslon PVC160203PYL01 (HD44780

Služi za prikaz poruka i podataka iz mikrokontrolera.



Sl.A.4. *LCD zaslon.*

Pin broj	Simbol	U/I	Funkcija
1	Vss	U	Signal mase
2	VDD	U	Signal napajanja za logiku
3	Vo	I	Signal napajanja za LCD
4	RS	U	Odabir registra: H : Podatkovni, L : Instrukcijski.
5	R/W	U	Čitanje/Pisanje
6	E	U	Signal omogućavanja
7~14	DB0 ~ DB7	U	Podatkovna sabirnica
15	A	U	Nije spojeno
16	K	U	Nije spojeno

Tab.A.2. *Značenja pinova LCD zaslona.*

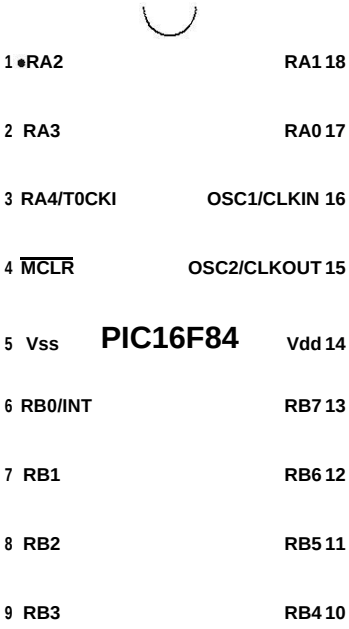
Instrukcija	Kod										Funkcija
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Čišćenje zaslona	0	0	0	0	0	0	0	0	0	1	Čisti cijeli zaslon, vraća zaslon na početak, i učitava DD RAM adresu 80H u adresno brojilo.
Zaslon/Kursor na početak	0	0	0	0	0	0	0	0	1		Vraća zaslon/kursor na početak i učitava DD RAM adresu 80H u adresno brojilo.
Postavljanje moda unosa	0	0	0	0	0	0	0	1	I/D	S	Određuje smjer pomaka kursora i zaslona. Ova instrukcija dolazi nakon svake instrukcije čitanja/pisanja podatka
Zaslon ON/OFF	0	0	0	0	0	0	1	D	C	B	Određuje hoće li zaslon i kursor biti prikazani, te da li će kursor treperiti.
Pomak Zaslona/Kursora	0	0	0	0	0	1	S/C	R/L			Pomak zaslona ili kursora za jedno mjesto.
Postavljanje moda rada	0	0	0	0	1	DL	N	F			Odabir dužine podataka, broja prikazanih redaka, i veličineprikazanih znakova.
Postavljanje adrese RAM-a	0	0	0	1	ACG						Učitavanje CG RAM adrese u adresno brojilo. Kasniji pristup podatku je za CG RAM podatak.
Postavljanje adrese DD RAM-a	0	0	1	ADD							Učitavanje DD RAM adrese u adresno brojilo. Kasniji pristup podatku je za DD RAM podatak.
Zastavica zauzeto/ Čitanje adresnog brojila	0	1	BF	AC							Čitanje zastavice zauzetosti (BF) i sadržaja adresnog brojila.
CG RAM/ DD RAM upis podatka	1	0	Upis podatka								Upis podatka u CG RAM ili DD RAM.
CG RAM/ DD RAM čitanje podatka	1	1	Čitanje podatka								Čitanje podatka iz CG RAM-a ili DD RAM-a.
	I/D = 1 : Pomak kursora ili zaslona udesno I/D = 0 : Pomak kursora ili zaslona ulijevo S = 1 : Uključen pomak zaslona D = 1 : Uključen zaslon C = 1 : Uključen prikaz kursora B = 1 : Uključeno treperenje kursora										DD RAM : RAM podataka zaslona CG RAM : RAM generatora znakova

	S/C = 1 : Pomak zaslona S/C = 0 : Pomak kursora R/L = 1 : Pomak udesno R/L = 0 : Pomak ulijevo DL = 1 : 8 bitni mod rada DL = 0 : 4 bitni mod rada N = 1 : Mod rada s dva retka na zaslonu N = 0 : Mod rada s jednim retkom na zaslonu F = 1 : Veličina znaka od 5x10 točkica F = 0 : Veličina znaka od 5x8 točkica BF = 1 : Unutarnje izvršavanje instrukcije (zauzeto) BF = 0 : Spreman za instrukciju	ACG:RAMadresa generatora znakova ADD : RAM adresa podataka zaslona AC : adresno brojilo

Napomena: Simbol " " na mjestu bita znači da je vrijednost dotičnog bita nevažna.

Tab.A.3. *Skup instrukcija LCD zaslona.*

Mikrokontroler PIC16F84



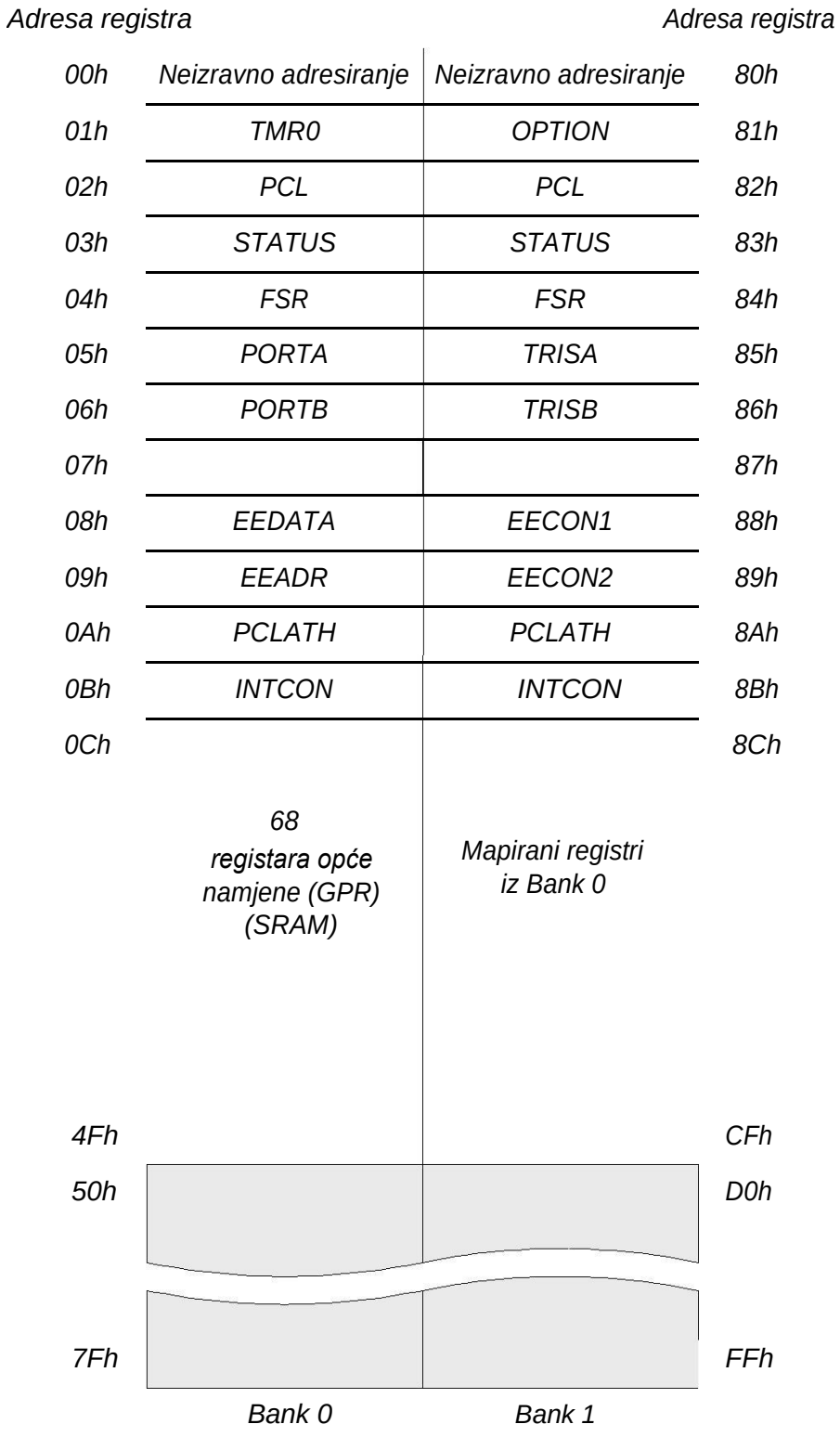
Sl.A.5. *Dijagram pinova.*

Oznaka pina	DIP br.	SOIC br.	U/I tip	Tip međuspremnika	Opis
OSC1/CLKIN	16	16	U	ST/CMOS	Ulaz kristalnog oscilatora/ulaz vanjskog izvora takta.
OSC2/CLKOUT	15	15	I	—	Izlaz kristalnog oscilatora. Spaja se na kristal ili rezonator u modu rada kristalnog oscilatora. U RC modu, OSC2 pin daje na izlazu CLKOUT koji je 1/4 frekvencije od OSC1, i predstavlja mjeru instrukcijskog ciklusa.
$\overline{MCLR}$	4	4	U	ST	"Master clear" (reset) ulaz/naponski ulaz za programiranje. Ovaj pin je aktivan na nisku razinu i služi za reset uređaja.
RA0	17	17	U/I	TTL	PORTA je dvosmjerni U/I port.
RA1	18	18	U/I	TTL	
RA2	1	1	U/I	TTL	
RA3	2	2	U/I	TTL	
RA4/T0CKI	3	3	U/I	ST	Može biti odabran da bude ulazni takt za TMR0 timer/brojilo. Izlaz je tipa otvorenog odvoda.
RB0/INT	6	6	U/I	TTL/ST	

RB1	7	7	U/I	TTL	PORTB je dvosmjerni U/I port. Može biti softverski programiran za unutarnji "slabi pull-up" na svim ulazima. RB0/INT može biti odabran kao vanjski prekidni pin. RB4 do RB7 su pinovi koji mogu generirati prekid kod promjene stanja na pinu. Takt za serijsko programiranje. Podaci za serijsko programiranje.
RB2	8	8	U/I	TTL	
RB3	9	9	U/I	TTL	
RB4	10	10	U/I	TTL	
RB5	11	11	U/I	TTL	
RB6	12	12	U/I	TTL/ST	
RB7	13	13	U/I	TTL/ST	
VSS	5	5	—	—	Signal mase.
VDD	14	14	—	—	Signal pozitivnog napajanja.

Tab.A.4. *Značenja pinova.*

Dodatak B. REGISTRI MIKROKONTROLERA



Adresa	Naziv	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bank 0									
00h	INDF	Koristi sadržaj FSR registra za adresiranje podatkovne memorije (nije fizički registar)							
01h	TMR0	8-bitni sat/brojilo u realnom vremenu							
02h	PCL	Donjih 8 bitova programskog brojila (PC)							
03h	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
04h	FSR	Pokazivač za neizravno adresiranje podatkovne memorije							
05h	PORTA	-	-	-	RA4/ T0CKI	RA3	RA2	RA1	RA0
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT
07h		Nije implementiran (čita se kao "0'")							
08h	EEDATA	Registar podataka EEPROM-a							
09h	EEADR	Adresni registar EPROM-a							
0Ah	PCLATH	-	-	-	Gornjih 5 bitova programskog brojila				
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	TOIF	INTF	RBIF
Bank 1									
80h	INDF	Koristi sadržaj FSR registra za adresiranje podatkovne memorije (nije fizički registar)							
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0
82h	PCL	Donjih 8 bitova programskog brojila							
83h	STATUS	IRP	RP1	RP0	IO	PD	Z	DC	C
84h	FSR	Pokazivač za neizravno adresiranje podatkovne memorije							
85h	TRISA	-	-	-	Registar smjera porta A				
86h	TRISB	Registar smjera porta B							
87h		Nije implementiran (čita se kao "0")							
88h	EECON1	-	-	-	EEIE	WRERR	WREN	WR	RD
89h	EECON2	Drugi kontrolni registar EEPROM-a							
8Ah	PCLATH	-	-	-	Gornjih 5 bitova programskog brojila				
8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	TOIF	INTF	RBIF

STATUS REGISTAR (ADRESA 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit7							bit0

R = bit koji se može čitati
W = bit koji se može pisati
U = bit koji nije implementiran, čita se kao "0"
- n = Vrijednost na POR reset

bit 7:

IRP: Bit za odabir bank-a (koristi se za neizravno adresiranje)
0 = Bank 0, 1 (00h - FFh)
1 = Bank 2, 3 (100h - 1FFh)
IRP bit se ne koristi kod PIC16F8X. Ovaj bit treba biti obrisan.

bit 6-5:

RP1:RP0: Bit za odabir bank-a (koristi se za izravno adresiranje)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
10 = Bank 2 (100h - 17Fh)
11 = Bank 3 (180h - 1FFh)
Svaki bank je veličine 128 bajta. Samo bit RP0 se koristi kod PIC16F8X. Bit RP1 treba biti obrisan

bit 4:

$\overline{TO}$: Bit isteka vremena (engl. Time-out bit)
1 = Nakon dovođenja pod napon, CLRWDT instrukcije, ili SLEEP instrukcije
0 = Kod WDT isteka vremena

bit 3:

$\overline{PD}$: Bit signalizacije napajanja (engl. Power-down bit)
1= Nakon dovođenja pod napon ili kod CLRWDT instrukcije
0= Kod izvršenja SLEEP instrukcije

bit 2:

Z: Bit nule (engl. Zero bit)
1 = Rezultat aritmetičke ili logičke operacije je nula 0
= Rezultat aritmetičke ili logičke operacije nije nula

bit 1:

DC: Bit prijenosa/posuđivanja kod donjih 4 bita (engl. Digit carry/borrow bit) (kod ADDWF i ADDLW instrukcija)
1 = Pojavio se prijenos kod 4 niža bita u rezultatu
0 = Nije se pojavio prijenos kod 4 niža bita u rezultatu

bit 0:

C: Bit prijenosa/posuđivanja (engl. Carry/borrow bit) (kod ADDWF i ADDLW instrukcija) 1 = Pojavio se prijenos kod najvažnijeg bita u rezultatu
0 = Nije se pojavio prijenos kod najvažnijeg bita u rezultatu

OPTION_REG REGISTAR (ADRESA 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7							bit0

R = bit koji se može čitati
W = bit koji se može pisati
U = bit koji nije implementiran, čita se kao
"0" - n = Vrijednost na POR reset

bit 7:

$\overline{\text{RBPU}}$: PORTB "Pull-up" bit omogućavanja
1= PORTB "pull-up"- onemogućeni
0= PORTB "pull-up"-i omogućeni

bit 6:

INTEDG: Bit odabira brida okidanja prekida (engl. Interrupt Edge Select bit) 1 = Prekid na rastući brid na RB0/INT pin
0 = Prekid na padajući brid na RB0/INT pin

bit 5:

T0CS: TMR0 bit odabira izvora takta (engl. TMR0 Clock Source Select bit) 1 = Promjena na RA4/T0CKI pinu
0 = Takt na unutarnji instrukcijski ciklus (CLKOUT)

bit 4:

T0SE: TMR0 bit odabira brida izvora (engl. TMR0 Source Edge Select bit) 1 = Povećanje na promjenu s "visokog" u "nisko" na RA4/T0CKI pinu
0 = Povećanje na promjenu s "niskog" u "visoko" na RA4/T0CKI pinu

bit 3:

PSA: Bit pridruživanja predjelila (engl. Prescaler Assignment bit) 1 = Predjelilo dodijeljeno WDT-u
0 = Predjelilo dodijeljeno TMR0-u

bit 2-0:

PS2:PS0: Bitovi odabira vrijednosti predjelila (engl. Prescaler Rate Select bits)

Vrijednosti bitova	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

INTCON REGISTAR (ADRESA 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

R = bit koji se može čitati
W = bit koji se može pisati
U = bit koji nije implementiran, čita se kao "0"
- n = Vrijednost na POR reset

bit 7:

GIE: Bit globalnog omogućavanja prekida (engl. Global Interrupt Enable bit) 1 = Omogućavanje svih nemaskiranih prekida
0 = Onemogućavanje svih prekida

bit 6:

EEIE: Bit omogućavanja prekida kod završetka pisanja u EEPROM (engl. EE Write Complete Interrupt Enable bit)
1 = Omogućavanje prekida kod završetka pisanja u EEPROM
0 = Onemogućavanje prekida kod završetka pisanja u EEPROM

bit 5:

T0IE: Bit omogućavanja prekida kod preljeva TMR0-a (engl. TMR0 Overflow Interrupt Enable bit) 1 = Omogućavanje prekida kod preljeva TMR0-a
0 = Onemogućavanje prekida kod preljeva TMR0-a

bit 4:

INTE: Bit omogućavanja prekida na RB0/INT (engl. RB0/INT Interrupt Enable bit) 1 = Omogućavanje prekida na RB0/INT
0 = Onemogućavanje prekida na RB0/INT

bit 3:

RBIE: Bit omogućavanja prekida kod promjene na pinu na portu B (engl. RB Port Change Interrupt Enable bit)
1 = Omogućavanje prekida kod promjene na pinu na portu B
0 = Onemogućavanje prekida kod promjene na pinu na portu B

bit 2:

T0IF: Bit prekida kod preljeva TMR0-a (engl. TMR0 overflow interrupt flag bit) 1 = Preljev kod TMR0-a (mora biti obrisan softverski)
0 = Nema preljeva kod TMR0-a

bit 1:

INTF: Bit zastavice prekida na RB0/INT (engl. RB0/INT Interrupt Flag bit) 1 = Primjećen prekid na RB0/INT
0 = Nije primjećen prekid na RB0/INT

bit 0:

RBIF: Bit zastavice prekida kod promjene na pinovima porta B (engl. RB Port Change Interrupt Flag bit) 1 = Kad je najmanje na jednom od pina RB7:RB4 stanje promijenjeno (mora biti obrisan softverski)
0 = Nema promjene stanja na nijednom od pina RB7:RB4

EECON1 REGISTAR (ADRESA 88h)

U	U	U	R/W-0	R/W-xR/W-0		R/S-0	R/S-x
—	—	—	EEIF	WRERR	WREN	WR	RD
bit7							bit0

R = bit koji se može čitati
W = bit koji se može pisati
S = bit koji se može postavljati
U = bit koji nije implementiran, čita se kao "0"
- n = Vrijednost na POR reset

bit 7:5:
Nisu implementirani: Čitaju se kao "0"

bit 4:
EEIF: Bit zastavice prekida kod pisanja u EEPROM (engl. EEPROM Write Operation Interrupt Flag bit) 1 = Operacija pisanja završena (mora biti obrisan softverski)
0 = Operacija pisanja nije završena ili nije počela

bit 3:
WRERR: Bit zastavice greške kod pisanja u EEPROM (engl. EEPROM Error Flag bit)
1 = Operacija pisanja prijevremeno završila (svaki $\overline{\text{MCLR}}$ reset ili svaki WDT reset tijekom normalnog rada)
0 = Operacija pisanja završila

bit 2:
WREN: Bit omogućavanja pisanja u EEPROM (engl. EEPROM Write Enable bit) 1 = Omogućen ciklus pisanja
0 = Onemogućen ciklus pisanja

bit 1:
WR: Bit kontrole pisanja (engl. Write Control bit)
1 = Pokretanje ciklusa pisanja EEPROM-a (ovaj bit je obrisan hardverski kad završi pisanje; WR bit može biti samo postavljen (ne obrisan) softverski)
0 = Ciklus pisanja u EEPROM je završen

bit 0:
RD: Bit kontrole čitanja (engl. Read Control bit)
1 = Pokretanje ciklusa čitanja EEPROM-a (čitanje traje jedan ciklus; RD je obrisan hardverski; RD bit može biti samo postavljen (ne obrisan) softverski)
0 = Ne pokreće se čitanje EEPROM-a

KONFIGURACIJSKA RIJEČ - PIC16F83 I PIC16F84

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	*PWRTE*	WDTE	FOSC1	FOSC0
bit13													bit0

R = bit koji se može čitati
P = bit koji se može programirati
- n = Vrijednost na POR reset
u = bit koji se ne može promijeniti

bit 13:4:

CP: Bitovi zaštite koda (engl. Code Protection
bit) 1 = Zaštita koda je isključena
0 = Cijela memorija je zaštićena

bit 3:

$\overline{\text{PWRTE}}$: Bit omogućavanja mjerača vremena kod dovođenja pod napon (engl. Power-up Timer Enable
bit)
1 = Mjerač vremena je onemogućen
0 = Mjerač vremena je omogućen

bit 2:

WDTE: Bit omogućavanja "Watchdog" mjerača vremena (engl. Watchdog Timer Enable
bit) 1 = WDT omogućen
0 = WDT onemogućen

bit 1:0:

FOSC1:FOSC0: Bitovi odabira oscilatora (engl. Oscillator Selection
bits) 11 = RC oscilator
10 = HS oscilator
01 = XT oscilator
00 = LP oscilator

RESET STANJA ZA SVE REGISTRE

Registar	Adresa	Reset kod dovođenja pod napon	$\overline{MCLR}$ reset tijekom: - normalnog rada - SLEEP-a WDT reset tijekom normalnog rada	"Buđenje" iz SLEEP-a: - pomoću prekida - pomoću WDT isteka vremena
W	—	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	00h	---- ----	---- ----	---- ----
TMR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000h	0000h	PC + 1
STATUS	03h	0001 1xxx	000q quuu	uuuq quuu
FSR	04h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	05h	---x xxxx	---u uuuu	---u uuuu
PORTB	06h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATA	08h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEADR	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCLATH	0Ah	---0 0000	---0 0000	---u uuuu
INTCON	0Bh	0000 000x	0000 000u	uuuu uuuu
INDF	80h	---- ----	---- ----	---- ----
OPTION_REG	81h	1111 1111	1111 1111	uuuu uuuu
PCL	82h	0000h	0000h	PC + 1
STATUS	83h	0001 1xxx	000q quuu	uuuq quuu
FSR	84h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TRISA	85h	---1 1111	---1 1111	---u uuuu
TRISB	86h	1111 1111	1111 1111	uuuu uuuu
EECON1	88h	---0 x000	---0 q000	---0 uuuu
EECON2	89h	---- ----	---- ----	---- ----
PCLATH	8Ah	---0 0000	---0 0000	---u uuuu
INTCON	8Bh	0000 000x	0000 000u	uuuu uuuu

Dodatak C. SKUP INSTRUKCIJA MCU PIC16F84

Instrukcija		Opis	trajanje u strojnim ciklusima	utječe na STATUS bitove:
Instrukcije s bajtovima				
addwf	f,d	Zbrajanje sadržaja akumulatora i registra f	1	C,DC,Z
andwf	f,d	Logički "I" sadržaja akumulatora i registra f	1	Z
clrf	f	Brisanje sadržaja registra f	1	Z
clrw		Brisanje sadržaja akumulatora	1	Z
comf	f,d	Komplement sadržaja registra f	1	Z
decf	f,d	Smanjenje sadržaja registra f za 1	1	Z
decfsz	f,d	Smanjenje sadržaja registra f za 1, preskakanje sljedeće instrukcije ako je rezultat 0	1(2)	nijedan
incf	f,d	Povećanje sadržaja registra f za 1	1	Z
incfz	f,d	Povećanje sadržaja registra f za 1, preskakanje sljedeće instrukcije ako je rezultat 0	1(2)	nijedan
iorwf	f,d	Uključivi "ILI" sadržaja akumulatora i registra f	1	Z
movf	f,d	Premještanje sadržaja registra f	1	Z
movwf	f	Premještanje sadržaja akumulatora u registar f	1	nijedan
nop		Bez djelovanja	1	nijedan
rlf	f,d	Rotiranje sadržaja registra f ulijevo kroz "Carry"	1	C
rrf	f,d	Rotiranje sadržaja registra f udesno kroz "Carry"	1	C
subwf	f,d	Oduzimanje sadržaja akumulatora od registra f	1	C,DC,Z
swapf	f,d	Zamjena donja i gornja 4 bita registra f	1	nijedan
xorwf	f,d	Isključivo "ILI" sadržaja akumulatora i registra f	1	Z
Instrukcije s bitovima				
bcf	f,b	Brisanje bita b registra f	1	nijedan
bsf	f,b	Postavljanje bita b registra f	1	nijedan
btfsc	f,b	Testiranje bita b registra f, preskakanje sljedeće instrukcije ako je bit obrisan	1(2)	nijedan
btfss	f,b	Testiranje bita b registra f, preskakanje sljedeće instrukcije ako je bit postavljen	1(2)	nijedan
Instrukcije s konstantama i kontrole instrukcije				
addlw	k	Zbrajanje sadržaja akumulatora i konstante k	1	C,DC,Z
andlw	k	Logički "I" sadržaja akumulatora i konstante k	1	Z
call	k	Poziv potprograma	2	
clwrdt		Brisanje "Watchdog timer"-a	1	$\overline{TO}$, $\overline{PD}$
goto	k	Bezuvjetno grananje programa	2	nijedan
iorlw	k	Uključivi "ILI" sadržaja akumulatora i konstante k	1	Z
movlw	k	Upisivanje konstante k u akumulator	1	nijedan
retfie		Povratak iz prekidne rutine	2	nijedan
retlw	k	Povratak iz potprograma s vrijednošću k u akumulatoru	2	nijedan
return		Povratak iz potprograma	2	nijedan
sleep		Postavljanje procesora u "standby" mod	1	$\overline{TO}$, $\overline{PD}$
sublw	k	Oduzimanje sadržaja akumulatora od konstante k	1	C,DC,Z
xorlw	k	Isključivo "ILI" sadržaja akumulatora i konstante k	1	Z

ADDLW Dodavanje konstante u W

Sintaksa: [labela] ADDLW k
Operand: 0 k 255
Djelovanje: (W) + k (W)
Zastavica: C, DC, Z

Kodiranje:

11	111x	kkkk	kkkk

Opis: Sadržaj W registra dodaje se osam bitnoj konstanti 'k',
a rezultat se sprema u W registar.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje konstante 'k'	Obrada podataka	Spremanje u W

Primjer: ADDLW 0x15
Prije instrukcije W = 0x10
Nakon instrukcije W = 0x25

ANDLW Logički "I" konstante s W

Sintaksa: [labela] ANDLW k
Operand: 0 k 255
Djelovanje: (W) .AND. (k)(W)
Zastavica: Z

Kodiranje:

11	1001	kkkk	kkkk

Opis: Izvodi se logička "I" operacija između
sadržaja registra W i osam bitne konstante 'k'.
Rezultat se sprema u W registar.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje konstante 'k'	Obrada podataka	Spremanje u W

Primjer ANDLW 0x5F
Prije instrukcije W = 0xA3
Nakon instrukcije W = 0x03

ADDWF Zbrajanje W i f

Sintaksa: [labela] ADDWF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (W) + (f)(odredište)
Zastavica: C, DC, Z

Kodiranje:

00	0111	dfff	ffff

Opis: Dodaje sadržaj W registra registru 'f'.
Ako je 'd' jednak 0 rezultat se sprema u W registar,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer ADDWF FSR, 0
Prije instrukcije W = 0x17 FSR = 0xC2
Nakon instrukcije W = 0xD9 FSR = 0xC2

ANDWF Logički "I" W s f

Sintaksa: [labela] ANDWF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (W) .AND. (f)(odredište)
Zastavica: Z

Kodiranje:

00	0101	dfff	ffff

Opis: Izvodi se logička "I" operacija između W registra i registra 'f'.
Ako je 'd' jednak 0 rezultat se sprema u W registar,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer ANDWF FSR, 1
Prije instrukcije W = 0x17 FSR = 0xC2
Nakon instrukcije W = 0x17 FSR = 0x02

BCF **Brisanje bita u f**

Sintaksa: [labela] BCF f,b
Operand: 0 f 127
0 b 7
Djelovanje: 0 (f<b>)
Zastavica: Nijedna

Kodiranje:

01	00bb	bfff	ffff

Opis: Bit ’b’ u registru 'f' je obrisan.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u registar 'f'

Primjer BCF FLAG_REG, 7
Prije instrukcije FLAG_REG = 0xC7
Nakon instrukcije FLAG_REG = 0x47

BSF **Postavljanje bita u f**

Sintaksa: [labela] BSF f,b
Operand: 0 f 127
0 b 7
Djelovanje: 1 (f<b>)
Zastavica: Nijedna

Kodiranje:

01	01bb	bfff	ffff

Opis: Bit ’b’ u registru 'f' je postavljen.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u registar 'f'

Primjer BSF FLAG_REG, 7
Prije instrukcije FLAG_REG = 0x0A
Nakon instrukcije FLAG_REG = 0x8A

BTFSC **Testiranje bita u f, preskakanje ako je obrisan**

Sintaksa: [labela] BTFSC f,b
Operand: 0 f 127
0 b 7
Djelovanje: preskakanje ako je (f<b>) = 0
Zastavica: Nijedna

Kodiranje:

01	10bb	bfff	ffff

Opis: Ako je bit ’b’ u registru 'f' jednak 1 tada se sljedeća instrukcija izvodi.
Ako je bit ’b’ u registru 'f' jednak 0 tada se sljedeća instrukcija preskače,
i umjesto nje se izvodi NOP instrukcija (što čini ovu instrukciju dvociklusnom).

Broj riječi: 1
Broj ciklusa: 1(2)
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Bez djelovanja

Ako se preskače: (2. Ciklus)

Q1	Q2	Q3	Q4
Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer OVDJE BTFSC ZASTAVICA,1

ISTINA
•
•
•

Prije instrukcije PC = adresa od OVDJE
Nakon instrukcije ako je ZASTAVICA<1> = 0, PC = adresa od ISTINA
ako je ZASTAVICA<1> = 1, PC = adresa od LAZ

BTFSS **Testiranje bita u f, preskakanje ako je postavljen**

Sintaksa: [labela] BTFSS f,b
Operand: 0 f 127
0 b 7
Djelovanje: preskakanje ako je (f<b>) = 1
Zastavica: Nijedna

Kodiranje:

01	11bb	bfff	ffff

Opis: Ako je bit ’b’ u registru 'f' jednak 0 tada se sljedeća instrukcija izvodi.
Ako je bit ’b’ u registru 'f' jednak 1 tada se sljedeća instrukcija preskače,
i umjesto nje se izvodi NOP instrukcija (što čini ovu instrukciju dvociklusnom).

Broj riječi: 1
Broj ciklusa: 1(2)

Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Bez djelovanja

Ako se preskače: (2. Ciklus)

Q1	Q2	Q3	Q4
Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer OVDJE BTFSS ZASTAVICA,1

ISTINA
•
•
•

Prije instrukcije PC = adresa od OVDJE
Nakon instrukcije ako je ZASTAVICA<1> = 0, PC = adresa od LAZ
ako je ZASTAVICA<1> = 1, PC = adresa od ISTINA

CALL **Poziv potprograma**

Sintaksa: [labela] CALL k
Operand: 0 k 2047
Djelovanje: (PC) + 1TOS, kPC<10:0>,
(PCLATH<4:3>)PC<12:11>
Zastavica: Nijedna

Kodiranje:

10	0kkk	kkkk	kkkk

Opis: Pozivanje potrograma. Prvo se na stog sprema adresa povratka (PC+1). Jedanaest bitova neposrednog adresiranja učitava se u PC bitove <10:0>. Gornji bitovi PC-a učitavaju se iz PCLATH-a. CALL je dvociklusna instrukcija.

Broj riječi: 1
Broj ciklusa: 2
Q taktovi:

	Q1	Q2	Q3	Q4
1. Ciklus	Dekodiranje	Čitanje konstante 'k', Spremanje PC-a na stog	Obrada podataka	Spremanje u PC
2. Ciklus	Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer OVDJE CALL TAMO
Prije instrukcije PC = adresa od OVDJE
Nakon instrukcije PC = adresa od TAMO TOS = adresa od OVDJE+1

CLRF **Brisanje registra f**

Sintaksa: [labela] CLRF f
Operand: 0 f 127
Djelovanje: 00h(f)
1 Z
Zastavica: Z

Kodiranje:

00	0001	1fff	ffff

Opis: Sadržaj registra 'f' je obrisan i postavljen je bit Z.

Broj riječi: 1
Broj ciklusa: 1

Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u registar 'f'

Primjer CLRF FLAG_REG
Prije instrukcije FLAG_REG = 0x5A
Nakon instrukcije FLAG_REG = 0x00 Z =1

CLRW Brisanje registra W

Sintaksa: [labela] CLRW
Operand: Nijedan
Djelovanje: 00h(W)
1 Z
Zastavica: Z

Kodiranje:

00	0001	0xxx	xxxx

Opis: W registar je obrisan. Bit nule (Z) je postavljen.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Bez djelovanja	Obrada podataka	Spremanje u W

Primjer CLRW
Prije instrukcije W = 0x5A
Nakon instrukcije W = 0x00 Z = 1

CLRWDT Brisanje "Watchdog Timer"-a

Sintaksa: [labela] CLRWDT
Operand: Nijedan
Djelovanje: 00h WDT
0 WDT prescaler,
1 $\overline{TO}$
1 $\overline{PD}$
Zastavica: $\overline{TO}$, $\overline{PD}$

Kodiranje:

00	0000	0110	0100

Opis: CLRWDT instrukcija resetira "Watchdog Timer".
Također resetira i predjelilo WDT-a. Status bitovi $\overline{TO}$ i $\overline{PD}$ su postavljeni.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Bez djelovanja	Obrada podataka	Brisanje WDT brojača

Primjer CLRWDT
Prije instrukcije WDT brojač = ?
Nakon instrukcije WDT brojač = 0x00 WDT predjelilo= 0
$\overline{TO}$ = 1 $\overline{PD}$ = 1

COMF **Komplement f**

Sintaksa: [labela] COMF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (f)(odredište)
Zastavica: Z

Kodiranje:

00	1001	dfff	ffff

Opis: Sadržaj registra 'f' se komplementira.
Ako je 'd' jednak 0 rezultat se sprema u W,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer COMF REG1,0
Prije instrukcije REG1 = 0x13
Nakon instrukcije REG1 = 0x13 W = 0xEC

DECF **Smanjenje za 1 registra f**

Sintaksa: [labela] DECF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (f) – 1 (odredište)
Zastavica: Z

Kodiranje:

00 0011	dfff	ffff	Opis: Smanjenje

sadržaja registra 'f' za 1.
Ako je 'd' jednak 0 rezultat se sprema u W,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer DECF CNT, 1
Prije instrukcije CNT = 0x01 Z = 0
Nakon instrukcije CNT = 0x00 Z = 1

DECFSZ **Smanjenje za 1 registra f, preskakanje ako je 0**

Sintaksa: [labela] DECFSZ f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (f) – 1 (odredište);
preskakanje ako je rezultat = 0
Zastavica: Nijedna

Kodiranje:

00	1011	dfff	ffff

Opis: Sadržaj registra 'f' se smanjuje za 1.
Ako je 'd' jednak 0 rezultat se sprema u W,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.
Ako je rezultat različit od 0, sljedeća instrukcija se izvodi.
Ako je rezultat jednak 0, tada se umjesto sljedeće izvodi
NOP instrukcija (što čini ovu instrukciju dvociklusnom).

Broj riječi: 1
Broj ciklusa: 1(2)
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Ako se preskače: (2. Ciklus)

Q1	Q2	Q3	Q4
Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer
```
OVDJE     DECFSZ CNT, 1
          GOTO PETLJA
NASTAVI
•
•
•
```
Prije instrukcije PC = adresa od OVDJE
Nakon instrukcije CNT = CNT - 1
ako je CNT = 0, PC = adresa od NASTAVI
ako je CNT 0, PC = adresa od OVDJE+1

GOTO **Bezuvjetno grananje**

Sintaksa: [labela] GOTO k
Operand: 0 k 2047
Djelovanje: kPC<10:0>
PCLATH<4:3>PC<12:11>
Zastavica: Nijedna

Kodiranje:

10	1kkk	kkkk	kkkk

Opis: GOTO izvodi bezuvjetno grananje.
Jedanaest bitova neposrednog adresiranja učitava se u PC bitove <10:0>.
Gornji bitovi PC-a učitavaju se iz PCLATH<4:3>.
GOTO je dvociklusna instrkcija.

Broj riječi: 1
Broj ciklusa: 2
Q taktovi:

	Q1	Q2	Q3	Q4
1. Ciklus	Dekodiranje	Čitanje konstante 'k'	Obrada podataka	Spremanje u PC
2. Ciklus	Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer GOTO TAMO
Nakon instrukcije PC = adresa od TAMO

INCF **Povećanje za 1 registra f**

Sintaksa: [labela] INCF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (f) + 1(odredište)
Zastavica: Z

Kodiranje:

00	1010	dfff	ffff

Opis: Sadržaj registra 'f' se uvećava za 1.
Ako je 'd' jednak 0 rezultat se sprema u W,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer INCF CNT, 1
Prije instrukcije CNT = 0xFF Z = 0
Nakon instrukcije CNT = 0x00 Z = 1

INCFSZ **Povećanje za 1 registra f, preskakanje ako je 0**

Sintaksa: [labela] INCFSZ f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (f) + 1 (odredište),
preskakanje ako je rezultat = 0
Zastavica: Nijedna

Kodiranje:

00	1111	dfff	ffff

Opis: Sadržaj registra 'f' se povećava za 1.
Ako je 'd' jednak 0 rezultat se sprema u W,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.
Ako je rezultat različit od 0, sljedeća instrukcija se izvodi.
Ako je rezultat jednak 0, tada se umjesto sljedeće izvodi
NOP instrukcija (što čini ovu instrukciju dvociklusnom).

Broj riječi: 1
Broj ciklusa: 1(2)
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Ako se preskače: (2. Ciklus)

Q1	Q2	Q3	Q4
Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer

```
OVDJE     INCFSZ CNT, 1
          GOTO PETLJA
NASTAVI
•
•
•
```

Prije instrukcije PC = adresa od OVDJE
Nakon instrukcije CNT = CNT + 1
ako je CNT = 0, PC = adresa od NASTAVI
ako je CNT 0, PC = adresa od OVDJE +1

IORLW Uključivi "ILI" konstante s W

Sintaksa: [labela] IORLW k
Operand: 0 k 255
Djelovanje: (W) .OR. k(W)
Zastavica: Z

Kodiranje:

11	1000	kkkk	kkkk

Opis: Vrši se logčka operacija uključivi "ILI" između sadržaja W registra i osam bitne konstante 'k'. Rezultat se sprema u W registar.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje konstante 'k'	Obrada podataka	Spremanje u W

Primjer IORLW 0x35
Prije instrukcije W = 0x9A
Nakon instrukcije W = 0xBF Z = 1

IORWF Uključivi "ILI" W s f

Sintaksa: [labela] IORWF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (W) .OR. (f)(odredište)
Zastavica: Z

Kodiranje:

00	0100	dfff	ffff

Opis: Uključivi "ILI" W i 'f' registra.
Ako je 'd' jednak 0 rezultat se sprema u W,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer IORWF REZULTAT, 0
Prije instrukcije REZULTAT = 0x13 W = 0x91
Nakon instrukcije REZULTAT = 0x13 W = 0x93 Z = 1

MOVLW Premjesti konstantu u W

Sintaksa: [labela] MOVLW k
Operand: 0 k 255
Djelovanje: k(W)
Zastavica: Nijedna

Kodiranje:

11	00xx	kkkk	kkkk

Opis: Osam bitna konstanta 'k' se učitava u W registar.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje konstante 'k'	Obrada podataka	Spremanje u W

Primjer MOVLW 0x5A
Nakon instrukcije W = 0x5A

MOVF Premjesti f

Sintaksa: [labela] MOVF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (f)(odredište)
Zastavica: Z

Kodiranje:

00	1000	dfff	ffff

Opis: Sadržaj registra 'f' premješta se na odredište ovisno o
'd'. Ako je 'd' jednak 0 rezultat se sprema u W,
a ako je 'd' jednak 1 rezultat se sprema u sam registar 'f'.
'd' = 1 je korisno za testiranje 'f' registra jer se utječe na Z zastavicu.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer MOVF FSR, 0
Nakon instrukcije W = vrijednost iz FSR registra Z = 1

MOVWF Premjesti W u f

Sintaksa: [labela] MOVWF f
Operand: 0 f 127
Djelovanje: (W) (f)
Zastavica: Nijedna

Kodiranje:

00	0000	1fff	ffff

Opis: Premještanje podatka iz W registra u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u registar 'f'

Primjer MOVWF OPTION_REG
Prije instrukcije OPTION_REG = 0xFF W = 0x4F
Nakon instrukcije OPTION_REG = 0x4F W = 0x4F

NOP Bez djelovanja

Sintaksa: [labela] NOP
Operand: Nijedan
Djelovanje: Bez djelovanja
Zastavica: Nijedna

Kodiranje:

00 0000	0xx0 0000	Opis:	Bez djelovanja.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer NOP

RETFIE **Povratak iz prekidne rutine**

Sintaksa: [labela] RETFIE
Operand: Nijedna
Djelovanje: TOS PC,
1 GIE
Zastavica: Nijedna

Kodiranje:

00	0000	0000	1001

Opis: Povratak iz prekidne rutine. Sa stoga se skida vrh (TOS) i učitava se u PC. Prekidi su omogućeni postavljanjem GIE bita (INTCON<7>). Ovo je dvociklusna instrukcija.

Broj riječi: 1
Broj ciklusa: 2
Q taktovi:

	Q1	Q2	Q3	Q4
1. Ciklus	Dekodiranje	Bez djelovanja	Postavljanje GIE bita	Skidanje sa stoga
2. Ciklus	Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer RETFIE
Nakon prekida PC = TOS GIE = 1

RETLW Povratak s konstantom u W

Sintaksa: [labela] RETLW k
Operand: 0 k 255
Djelovanje: k(W);
TOS PC
Zastavica: Nijedna

Kodiranje:

11	01xx	kkkk	kkkk

Opis: U W registar se učitava osam bitna konstanta 'k'.
U PC se učitava vrh stoga (adresa povratka).
Ovo je dvociklusna instrukcija.

Broj riječi: 1
Broj ciklusa: 2
Q taktovi:

	Q1	Q2	Q3	Q4
1. Ciklus	Dekodiranje	Čitanje konstante 'k'	Bez djelovanja	Spremanje u W, Skidanje sa stoga
2. Ciklus	Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

Primjer
```
CALL TABLICA        ;W sadrži vrijednost pomaka u tablici
•
•
•
TABLICA   ADDWF PC  ;W = pomak
RETLW k1  ;početak tablice
RETLW k2  ;
RETLW kn  ;kraj tablice
```
Prije instrukcije W = 0x07
Nakon instrukcije W = vrijednost k8

RETURN **Povratak iz potprograma**

Sintaksa: [labela] RETURN
Operand: Nijedan
Djelovanje: TOS PC
Zastavica: Nijedna

Kodiranje:

00 0000 0000 1000	Opis:

Povratak iz potprograma.
U PC se skida adresa s vrha stoga (TOS).
Ovo je dvociklusna instrukcija.

Broj riječi: 1
Broj ciklusa: 2
Q taktovi:

Br. ciklusa	Q1	Q2	Q3	Q4
1st Cycle	Dekodiranje	Bez djelovanja	Bez djelovanja	Skidanje sa stoga
2nd Cycle	Bez djelovanja	Bez djelovanja	Bez djelovanja	Bez djelovanja

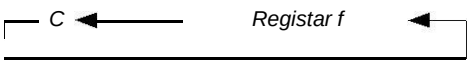
Primjer RETURN
Nakon prekida PC = TOS

RLF **Rotiranje registra f ulijevo kroz "Carry"**
Sintaksa: [labela] RLF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: Vidi opis
Zastavica: C

Kodiranje:

00	1101	dfff	ffff

Opis: Sadržaj registra 'f' rotira se jedan bit ulijevo kroz "Carry" zastavicu.
Ako je 'd' jednak 0 rezultat se sprema u W registar,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.



Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer RLF REG1,0
Prije instrukcije REG1 = 1110 0110 C = 0
Nakon instrukcije REG1 = 1110 0110 W = 1100 1100 C = 1

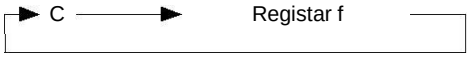
RRF **Rotiranje registra f udesno kroz "Carry"**

Sintaksa: [labela] RRF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: Vidi opis
Zastavica: C

Kodiranje:

00	1100	dfff	ffff

Opis: Sadržaj registra 'f' rotira se jedan bit udesno kroz "Carry" zastavicu.
Ako je 'd' jednak 0 rezultat se sprema u W registar,
a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.



Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer RRF REG1,0
Prije instrukcije REG1 = 1110 0110 C = 0
Nakon instrukcije REG1 = 1110 0110 W = 0111 0011 C = 0

SLEEP

Sintaksa: [labela] SLEEP
Operand: Nijedan
Djelovanje: 00h WDT,
0 WDT predjelilo,
1 $\overline{TO}$,
0 $\overline{PD}$
Zastavica: $\overline{TO}$, $\overline{PD}$

Kodiranje:

00	0000	0110	0011

Opis: "Power-down" status bit (PD) je obrisan.
"Time-out" status bit (TO) je postavljen. "Watchdog
Timer" i njegovo predjelilo su obrisani.
Procesor odlazi u SLEEP mod sa zaustavljenim oscilatorom.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Bez djelovanja	Bez djelovanja	Odlazak u "Sleep"

Primjer: SLEEP

SUBLW Oduzimanje W od konstante

Sintaksa: [labela] SUBLW k
Operand: 0 k 255
Djelovanje: k - (W) (W)
Zastavica: C, DC, Z

Kodiranje:

11	110x	kkkk	kkkk

Opis: Sadržaj W registra se oduzima (metoda dvojnog komplementa) od osam bitne konstante 'k'. Rezultat se sprema u W registar.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje konstante 'k'	Obrada podataka	Spremanje u W

Primjer 1:	SUBLW 0x02		
Prije instrukcije	W = 1	C = ?	Z = ?
Nakon instrukcije	W = 1	C = 1; rezultat je pozitivan	Z = 0
Primjer 2:			
Prije instrukcije	W = 2	C = ?	Z = ?
Nakon instrukcije	W = 0	C = 1; rezultat je nula	Z = 1
Primjer 3:			
Prije instrukcije	W = 3	C = ?	Z = ?
Nakon instrukcije	W = 0xFF	C = 0; rezultat je negativan	Z = 0

SUBWF Oduzimanje W od f

Sintaksa: [labela] SUBWF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (f) - (W) (odredište)
Zastavica: C, DC, Z

Kodiranje:

00	0010	dfff	ffff

Opis: Oduzimanje sadržaja (metodom dvojnog komplementa) W registra od sadržaja registra 'f'.
Ako je 'd' jednak 0 rezultat se sprema u W registar, a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1

Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer 1: SUBWF REG1,1
Prije instrukcije REG1 = 3 W = 2 C = ? Z = ?
Nakon instrukcije REG1 = 1 W = 2 C = 1; rezultat je pozitivan Z = 0

Primjer 2:
Prije instrukcije REG1 = 2 W = 2 C = ? Z = ?
Nakon instrukcije REG1 = 0 W = 2 C = 1; rezultat je nula Z = 1

Primjer 3:
Prije instrukcije REG1 = 1 W = 2 C = ? Z = ?
Nakon instrukcije REG1 = 0xFF W = 2 C = 0; rezultat je negativan Z = 0

SWAPF Zamjena bitova u f
Sintaksa: [labela] SWAPF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (f<3:0>)(odredište<7:4>),
(f<7:4>)(odredište<3:0>)
Zastavica: Nijedna

Kodiranje:

00	1110	dfff	ffff

Opis: Gornja i donja četiri bita registra 'f' mijenjaju mjesta.
Ako je 'd' jednak 0 rezultat se sprema u W registar, a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer SWAPF REG, 0
Prije instrukcije REG1 = 0xA5
Nakon instrukcije REG1 = 0xA5 W = 0x5A

XORLW Isključivo ili konstante s W

Sintaksa: [labela] XORLW k
Operand: 0 k 255
Djelovanje: (W) .XOR. k (W)
Zastavica: Z

Kodiranje:

11	1010	kkkk	kkkk

Opis: Isključivo "ILI" operacija između sadržaja W registra i osam bitne konstante 'k'. Rezultat se sprema u W registar.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje konstante 'k'	Obrada podataka	Spremanje u W

Primjer: XORLW 0xAF
Prije instrukcije W = 0xB5
Nakon instrukcije W = 0x1A

XORWF Isključivo "ILI" W s f
Sintaksa: [labela] XORWF f,d
Operand: 0 f 127
d Є [0,1]
Djelovanje: (W) .XOR. (f)(odredište)
Zastavica: Z

Kodiranje:

00	0110	dfff	ffff

Opis: Isključivo "ILI" između W i 'f' registra. Ako je 'd' jednak 0 rezultat se sprema u W registara, a ako je 'd' jednak 1 rezultat se sprema u registar 'f'.

Broj riječi: 1
Broj ciklusa: 1
Q taktovi:

Q1	Q2	Q3	Q4
Dekodiranje	Čitanje registra 'f'	Obrada podataka	Spremanje u odredište

Primjer XORWF REG 1
Prije instrukcije REG = 0xAF W = 0xB5
Nakon instrukcije REG = 0x1A W = 0xB5

Dodatak D. LITERATURA

[1] PIC16F84 Data Sheet, Microchip Technology Inc., 1998.

[2] MPLAB User's Guide, Microchip Technology Inc., 2000.

[3] PCF8591P Data Sheet, Philips Semiconductors, 2003.

[4] I2C Manual, Philips Semiconductors, 2003.

[5] PVC160203PYL01 Data Sheet, Picvue Electronics Co., Ltd.

[6] S547AP Data Sheet, Lite-on Electronics Inc.