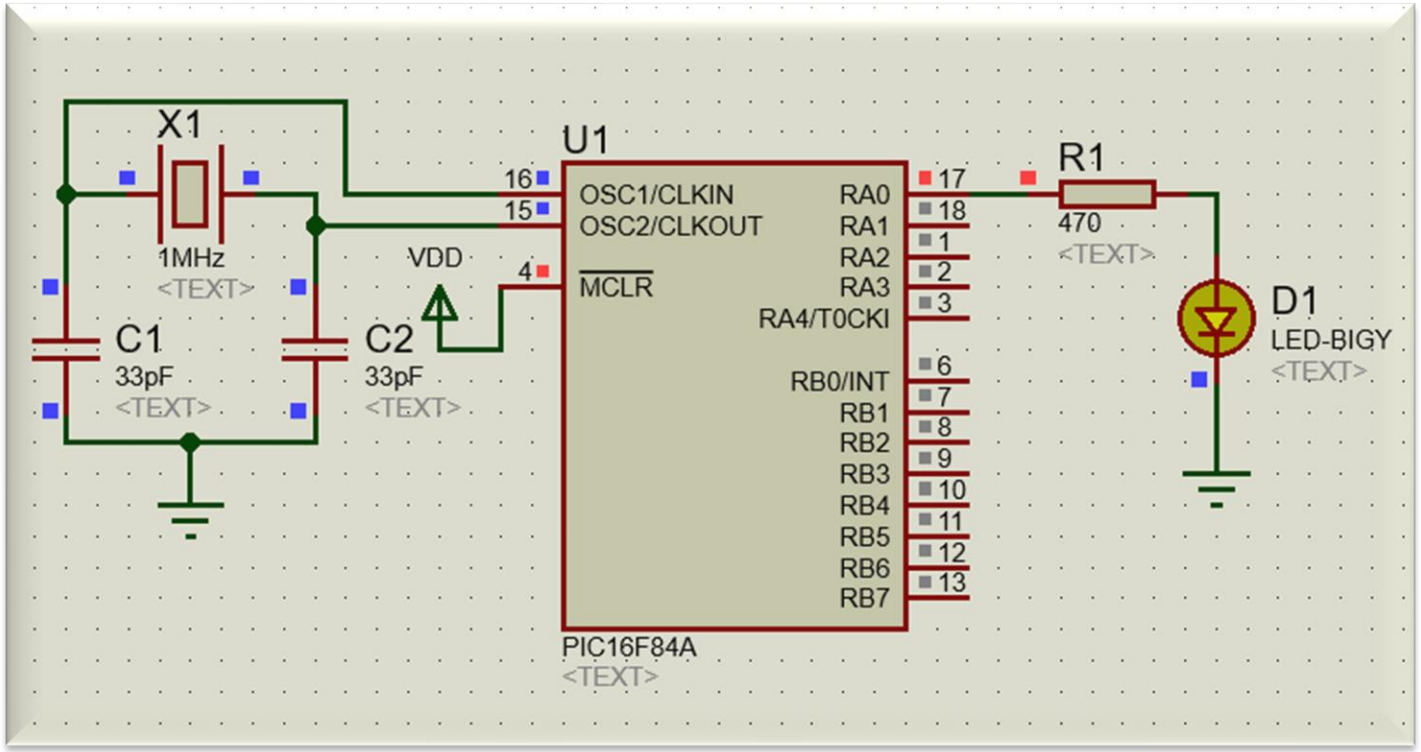
MIKRORAČUNARSKI SISTEMI (2021)
II deo računskih vežbi
PIC16 mikrokontroleri

Nenad Petrović

nenad.petrovic@elfak.ni.ac.rs



Uvod

- PIC (***P**eripheral **I**nterface **C**ontroller*) je familija mikrokontrolera razvijena od strane kompanije Microchip Technology
- Naziv su dobili po modelu PIC1650, razvijenog sredinom sedamdesetih godina proslog veka
- Kasnije, promenjeno značenje u ***P**rogrammable **I**ntelligent **C**omputer*
- Karakteristike:
  - niska cena
  - dobra podrška
  - velika zajednica korisnika

Istorija

- Nastao iz potrebe da se odvoje I/O operacije od CPU (1975. godine)
- Originalno je bio 8-bit
- Kreiran da radi zajedno sa 16-bit CPU – CP1600
- Koristio ROM memoriju za čuvanje koda
- Jedan od prvih RISC-baziranih uređaja
- 1993. se javlja PIC16C84, prvi sa EEPROM
- 2001. se javljaja PIC sa flash memorijom
- Do 2013. godine je u upotrebi preko 12 milijardi primeraka mikrokontrolera iz ove familije

Familije PIC

- Različte širine instrukcija i memorije podataka
- Memorija podataka
  - 8, 16, 32-bit
- Instrukcije
  - 12, 14, 16 i 24-bit
- Različite dubine poziva steka
- Različit broj U/I portova
- Različit broj registara i memorijskih reči
- Naprednije verzije poseduju dodatne module
  - Serijska komunikacija: USART, I2C, SPI, USB
  - ADC/DAC
  - Ethernet

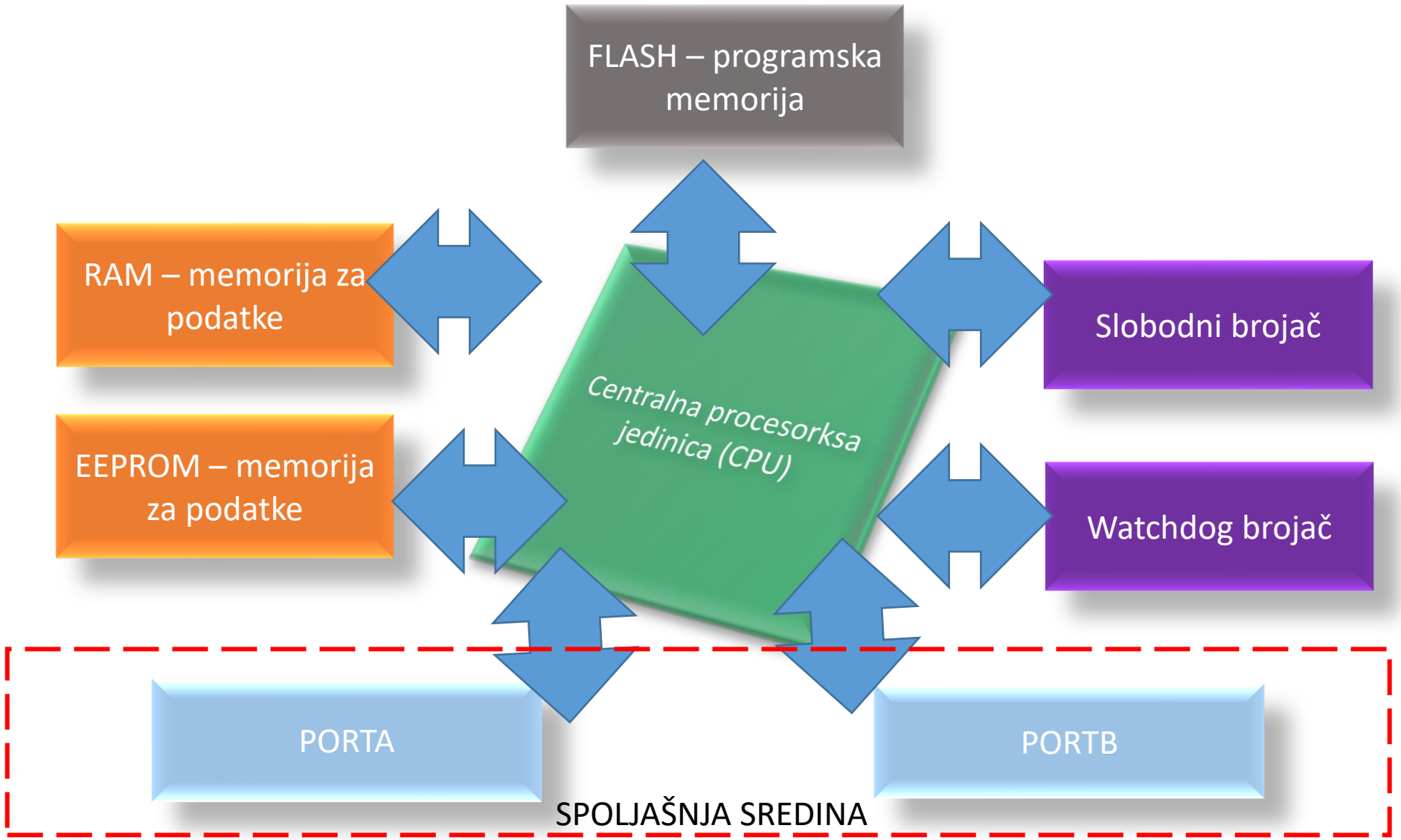
Pregled PIC familija

Familija	Glavne karakteristike	Dodaci
PIC10 i PIC12	12bit instrukcije, 2 nivoa steka, nema HW mul/div	-
PIC16	14bit instrukcije, 8 nivoa steka	-
PIC17	16bit instrukcije, 16 nivoa steka, HW množenje (8b x 8b)	-
PIC18	16bit instrukcije, 31 nivo steka, rad sa USB	USB podrška
PIC24	24bit Hardversko množenje u 1 ciklusu (16b x 16b) Deljenje u 19 ciklusa (32b / 16b)	DSP operacije, DMA pristup
PIC32	32bit, MIPS-bazirani, namenjen IoT primenama	USB OTG, Ethernet

Razvojna okruženja za PIC

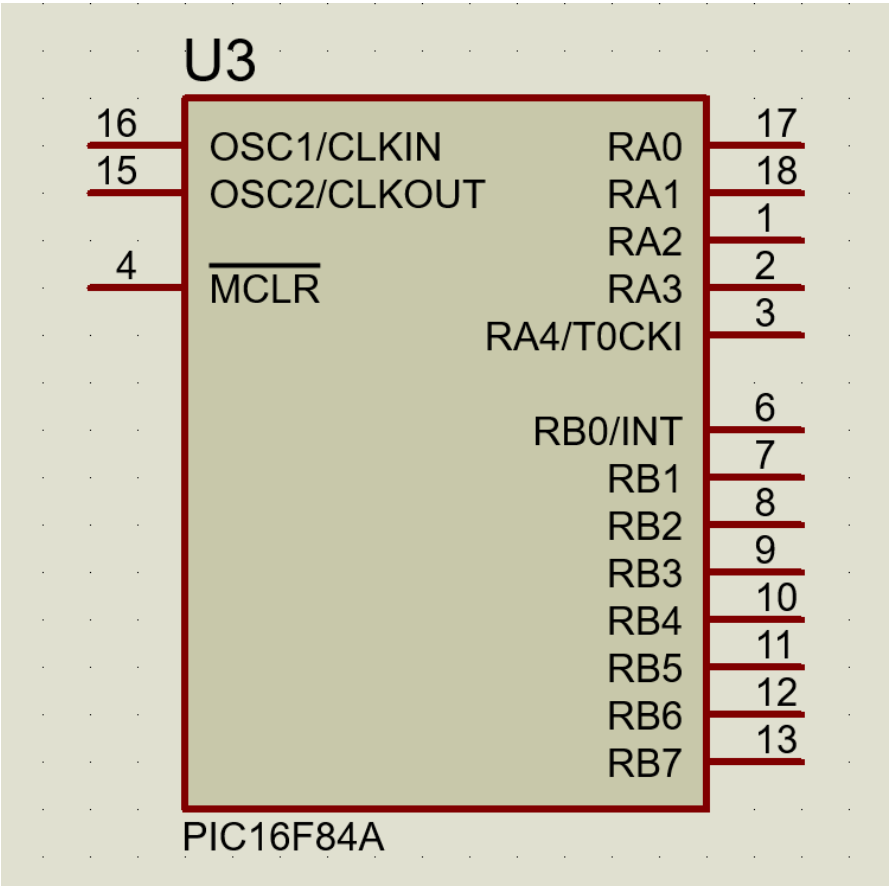
- Freeware MPLAB X razvojno okruženje
  - Razvijeno od strane proizvođača
  - Integriše assembler, linker, softverski simulator i debugger
- Različite mogućnosti pisanja koda
  - asembler
  - C (XC8, MikroC)
  - C++
  - Basic
  - Pascal
  - PICBASIC
- Proteus 8.4 SP0
  - Podrška za simulaciju
  - XC8 i ASM

Blok šema PIC16F84



Pinovi PIC16F84

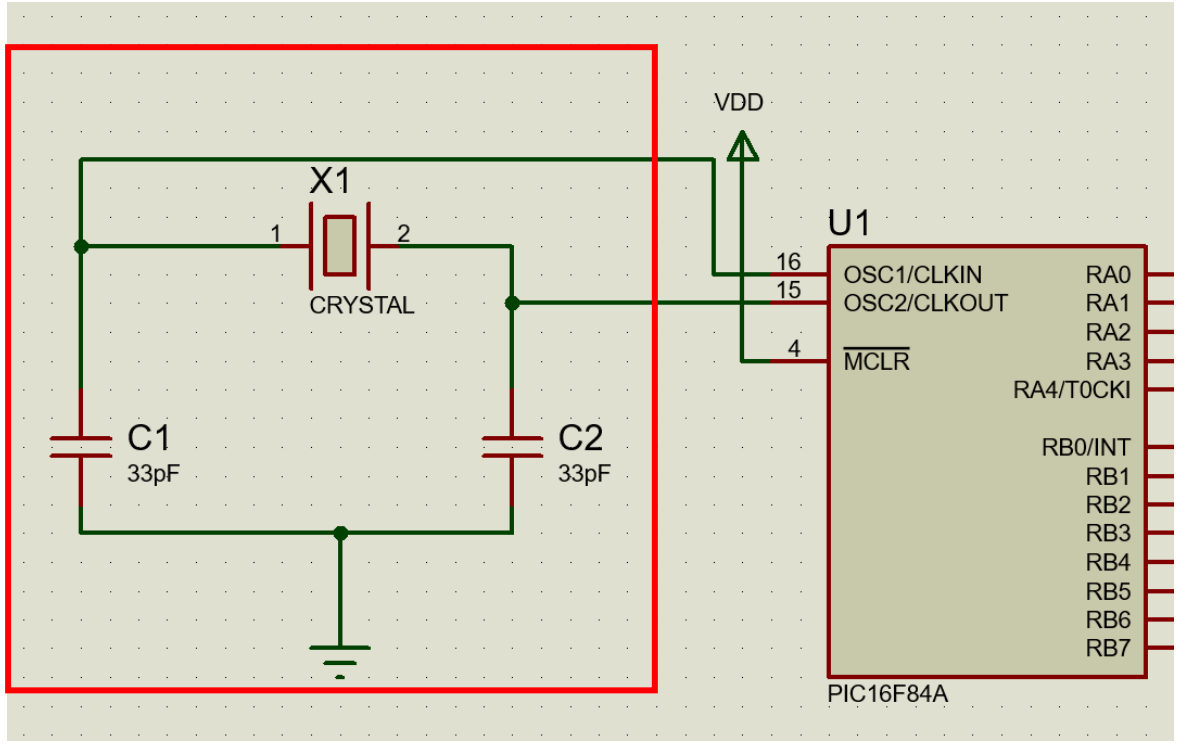
- Dolazi u 18-pin DIP formatu



Pinovi	Uloga
RA0-RA4	Pinovi porta A **RA4 – TOCK1 za tajmersku funkciju – spoljašnji takt**
RB0-RB7	Pinovi porta B **RB0/INT : (spoljašnji) interptni ulaz** **RB4-RB7 : interpatni ulaz (svi zajedno)**
Vdd	Pozitivan pol napajanja
Vss	Napajanje, masa
MCLR	Reset ulaz
OSC1 i OSC2	Pinovi namenjeni spajanju sa oscilatorom

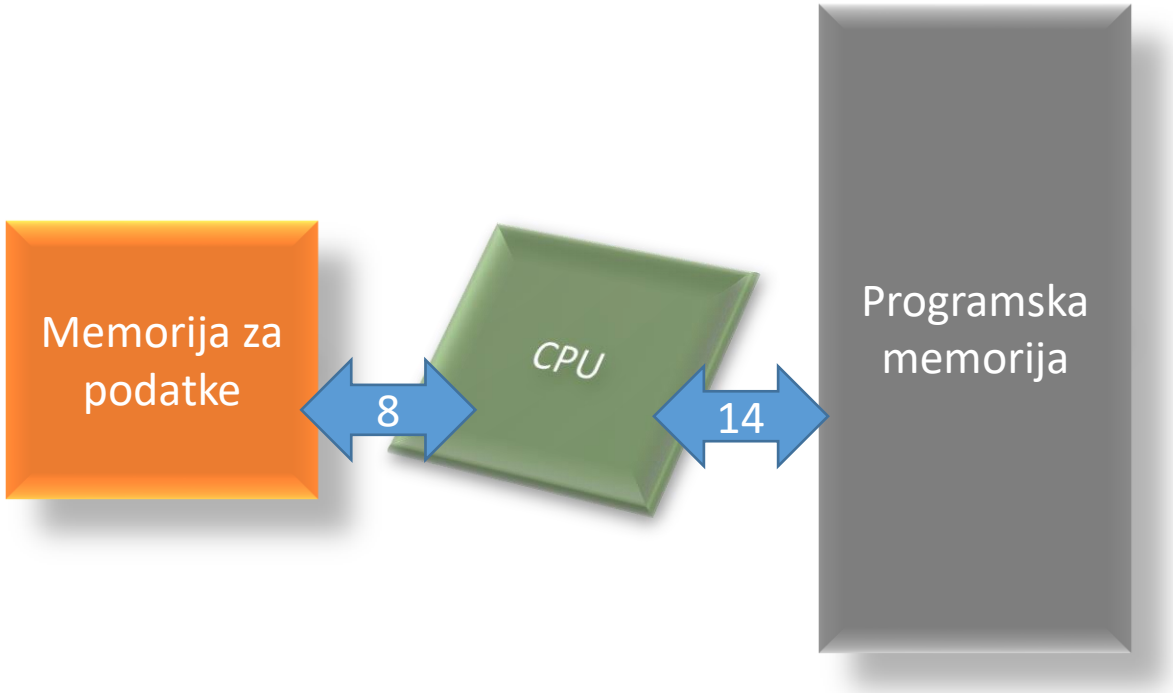
Oscilatori

- Procesorski takt predstavlja ¼ spoljašnjeg takta oscilatora
- Najčešće se koriste dve vrste oscilatora
  - Kristalni oscilator (XT, HS, LP)
  - Sa otpornikom i kondenzatorom (RC)

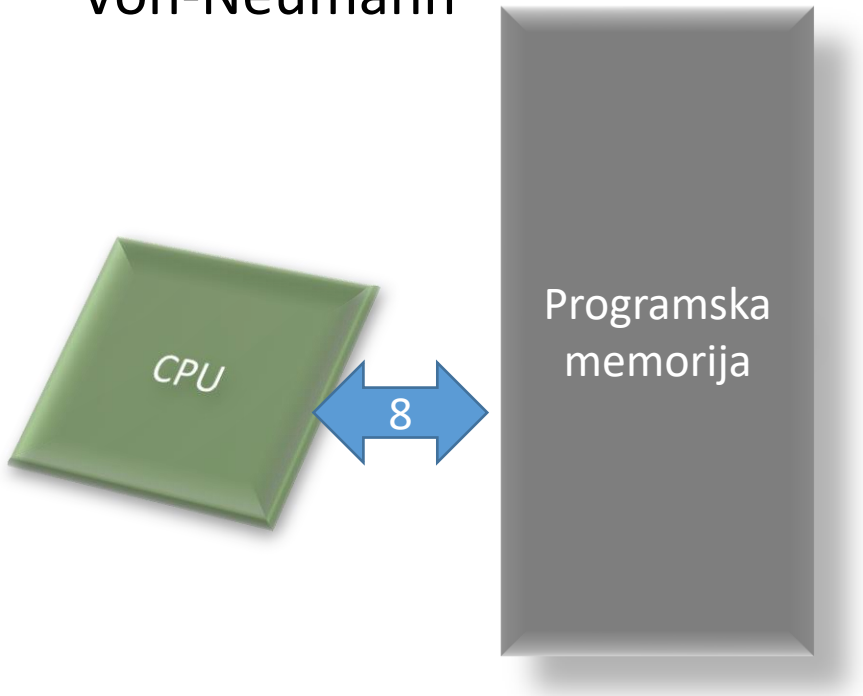


Harvard RISC arhitektura

- Harvard arhitektura omougćava istovremeno čitanje instrukcije i čitanje/upis podataka
  - Harvard



Von-Neumann



Primene PIC16F84

- Elektronske brave
- Sigurnosni uređaji
- Automobilska industrija
- Kućni aparati
- Svetleće reklame
- Energetski sistemi

Organizacija memorije

- Programski blok
  - FLASH :1Kx14b
- Blok za podatke
  - EEPROM: 64x8b
  - RAM
    - SFR (registri specijalne namene):
      - Dve memorijske banke
      - prvih 12 lokacija u banci 0 i 1
    - GPR (registri opšte namene):
      - 68x8b lokacija, pristup nezavisno od banke

00h 80h

BANK0	BANK1
INDF	INDF
TMR0	OPTION
PCL	PCL
STATUS	STATUS
FSR	FSR
PORTA	TRISA
PORTB	TRISB
EEDATA	EECON1
EEADR	EECON2
PCLATH	PCLATH
INTCON	INTCON
68 bajtova GPR – zajedničkih za obe banke	

12 SFR

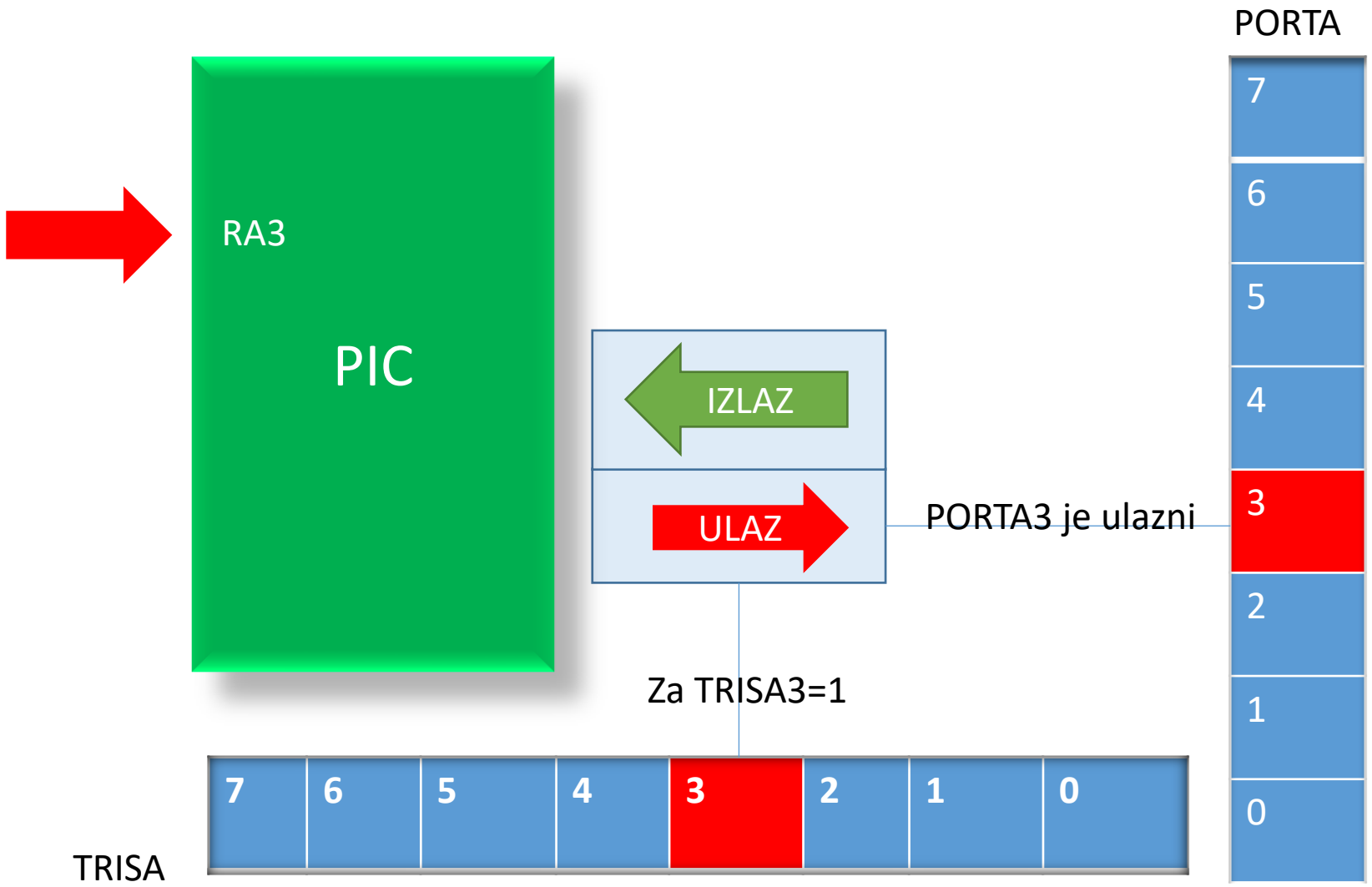
Portovi

- Fizička veza centralne procesorske jedinice sa spoljnim svetom.
- Svi pinovi portova mogu se definisati kao ulazni ili kao izlazni, već prema potrebama uređaja koji se razvija.
- PIC16F84 ima dva porta
  - PORTA (5 bit)
  - PORTB (8 bit)
- Da bi definisali pin kao ulazni ili kao izlazni mora se u registar TRIS upisati odgovarajuća kombinacija nula i jedinica. Ako je na odgovarajućem mestu u TRIS registru upisana logička jedinica ''1'' onda je taj pin ulazni, u obratnom slučaju pin je izlazni.
- Svaki port ima svoj odgovarajući TRIS registar. Tako port A ima TRISA a port B TRISB. Promena smera pinova se može vršiti u toku rada što je posebno pogodno za slučaj komunikacije preko jedne linije gde se menja smer prenosa podataka.
- Registri stanja portova PORTA i PORTB se nalaze u banci 0 dok se registri za smer pinova TRISA i TRISB nalaze u banci 1.

PORTA

- Ima 5 pinova koji su mu pridruženi
- Odgovarajući registar za smer podataka je TRISA
- RA4 pin
  - Na tom pinu se još nalazi i spoljni ulaz za brojač TMR0.
  - Da li se ovaj pin bira kao standardni ulazni pin ili kao ulaz brojača koji se vodi na TMR0 zavisi od bita T0CS (TMR0 Clock Source Select bit).
  - Ovaj bit omogućava da brojač TMR0 uvećava svoje stanje ili iz internog oscilatora ili preko spoljnih impulsa na pinu RA4/TOCKI.

Odnos TRIS i PORT registara



PORTB

- PORTB ima 8 pinova koji su mu pridruženi.
- Odgovarajući registar za smer podataka je TRISB
- Svaki pin na PORTB ima slabi interni pull-up otpornik (otpornik koji definiše liniju na logičku jedinicu) koji se može aktivirati resetovanjem sedmog bita RBPU u registru OPTION.
- Ovi ''pull-up'' otpornoci se automatski isključuče kada je pin porta konfigurisan kao izlaz.
  - Pri uključenju mikrokontrolera pull-up -ovi su onemogućeni.
- RB7:RB4 mogu izazvati prekid (interapt) koji se dešava kada se stanje na njima promeni iz logičke jedinice u logičku nulu i obratno.
- RB0 se koristi takođe kao izvor prekida (rastuća ili opadajuća ivica)

STATUS registar

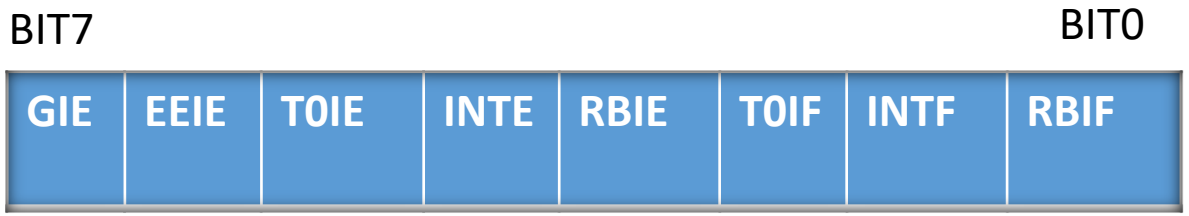
Bit	Uloga
7: IRP	Ne koristi se
6, 5: RP1 i RP0	Selekcija memorijske banke kod direktnog adresiranja **RP0=0 -> BANKA0 (00h-7Fh)** **RP0=1 BANKA1 (80h-FFh)** **RP1 se ne koristi kod PIC16F84A, ima samo 2 banke**
4: TO- (time out)	1 – Nakon dovođenja napajanja, CLRWDT ili SLEEP 0 – Dogodio se time out reset od WDT
3: PD- (power down)	1 – Nakon dovođenja napajanja ili CLRWDT 0 – nakon SLEEP naredbe
2: Z (zero)	1- rezultat aritmetičke ili logičke operacije je bio 0 0 – rezultat nije bio 0
1: DC (digit carry)	1- došlo do prenosa iz niže u višu tetradu 0 – nije bilo prenosa iz niže u višu tetradu
0: C (carry/borrow)	1-Dogodio se prenos iz bita najviše pozicije 0-Nije se dogodio prenos iz bita najviše pozicije **Napomena: kod oduzimanja je vrednost pozajmice komplementirana, bit 1 je kad nema pozajmice!**

Vrste prekida kod PIC16F84

PORTB

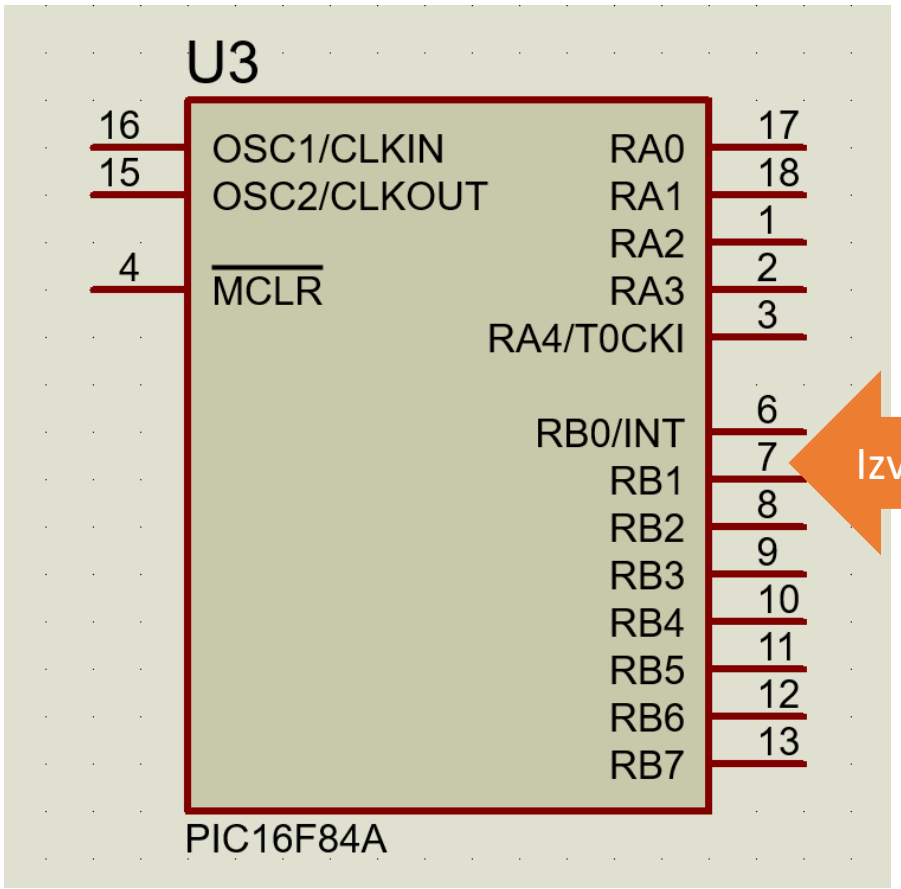
- Spoljašnji prekid na RB0/INT pinu mikrokontrolera
- Prekid prilikom promene na pinovima 4, 5, 6 i 7 porta B
- Prekid prilikom prekoračenja TMR0 brojača
- Prekid prilikom završetka upisivanja u EEPROM

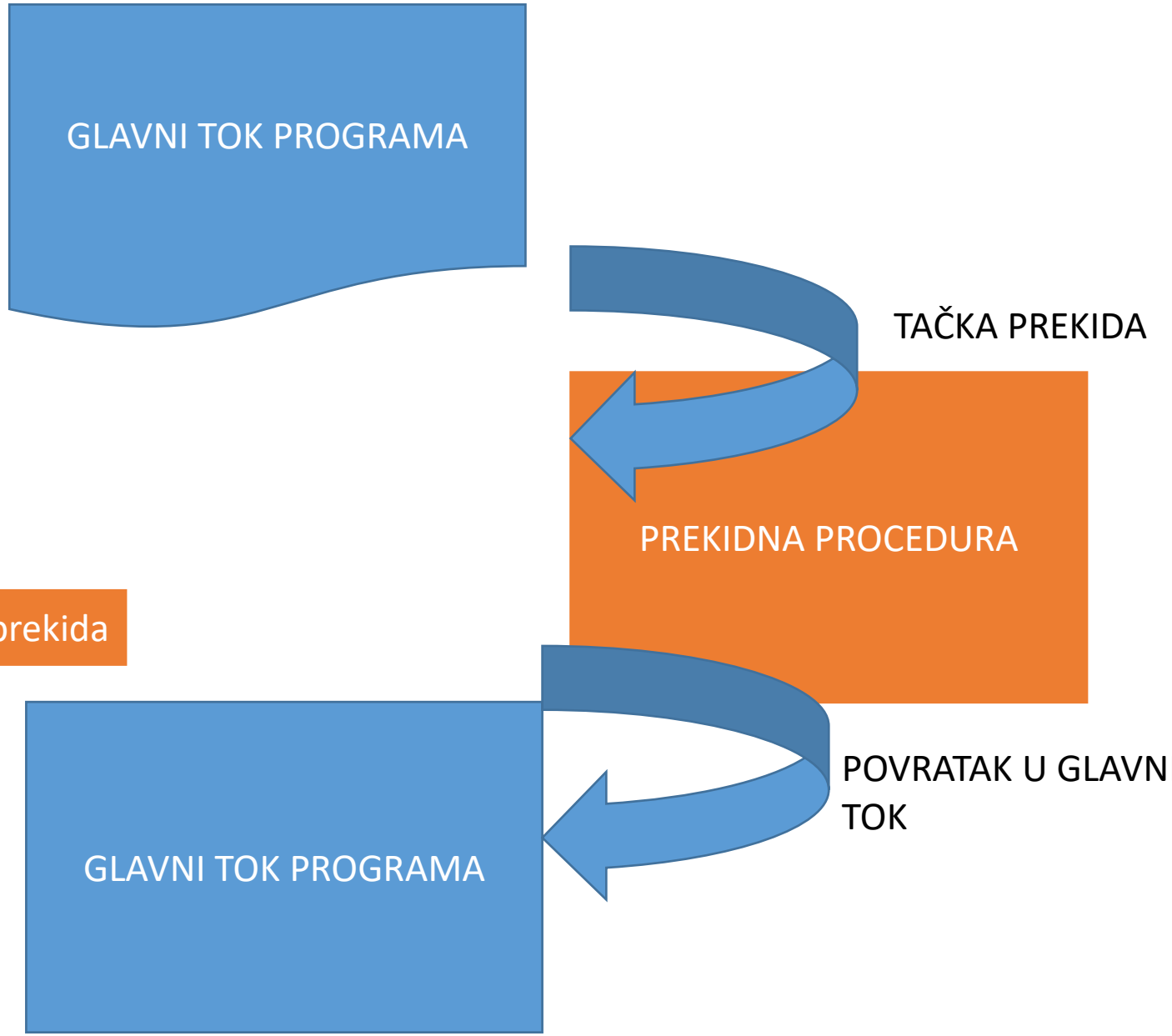
INTCON registar



Bit	Uloga
7: GIE	Bit globalne dozvole prekida: **1- svi prekidi omogućeni 0 – svi prekidi onemogućeni**
6: EEIE	Dozvoljava prekid nakon završetka upisa u EEPROM: **1- omogućen 0- nije omogućen**
5: T0IE	Dozvoljava prekid prekoračenja TMR0: **1- omogućen 0- nije omogućen**
4: INTE	Dozvoljava prekid na RB0/INT priključku: **1- omogućen 0- nije omogućen**
3: RBIE	Dozvoljava prekid po promeni RB4-RB7: **1- omogućen 0- nije omogućen**
2: T0IF	Indikator prekida TMR0: **1- došlo je do prekida 0- nije došlo do prekida**
1: INTF	Indikator prekida RB0/INT: **1- došlo je do prekida 0- nije došlo do prekida**
0: RBIF	Indikator prekida RB4-RB7: **1- došlo je do prekida 0- nije došlo do prekida**

Prekidi


U3
16
15
4
OSC1/CLKIN
OSC2/CLKOUT
MCLR
RA0
RA1
RA2
RA3
RA4/T0CKI
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6
RB7
17
18
1
2
3
6
7
8
9
10
11
12
13
PIC16F84A


GLAVNI TOK PROGRAMA
TAČKA PREKIDA
PREKIDNA PROCEDURA
Izvor prekida
GLAVNI TOK PROGRAMA
POVRATAK U GLAVNI TOK

Struktura prekidne rutine

- 1. Čuva se W radni registar bez obzira na tekuću banku
- 2. Čuva se STATUS registar u bank0.
- 3. Čuvaju se ostali registri
- [Test interrput flegova]
- [Izvršava se prekidna rutina za obradu detektovanog prekida (ISR)]
- [Čišćenje flega obrađenog prekida]
- 4. Restaurišu se ostali registri
- 5. Restauriše se STATUS registar
- 6. Restauriše se W registar

OPTION registar

BIT7 BIT0

RBPU-	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Bit	Uloga
7: Not RBPU	Bit globalne dozvole prekida: **1- PORTB otpornici isključeni 0 – PORTB otpornici uključeni**
6: INTEDG	Ivica za spoljašnji prekid RB0: **1- rastuća 0- opadajuća**
5: T0CS	Izbor takta zaTMR0: **1-spoljašni – RA4 0- interni CLK**
4: T0SE	Izbor ivice takta za TMR0 na RA4: **1- Prelaz sa visokog na nisko uvećava TMR0 0- Prelaz sa niskog na visoko uvećava TMR0**
3: PSA	Dodela preskalera: **1-WDT tajmeru 0-TMR0**
2, 1, 0: PS2, PS1, PS0	**Bitovi za izbor odnosa deljenja preskalera**

Preskaler

- Preskaler je naziv za deo mikrokontrolera koji deli instrukcijski ciklus pre nego što on dođe do logike koja povećava stanje brojača. Ovim se dobija mogućnost merenja dužih vremenskih perioda
- Preskaler se može pridružiti jednom od dva brojača pomoću PSA bita
  - TMR0
  - Watchdog timer
- PS2,PS1,PS0 definišu vrednost preskalera

Bitovi	TMR0	WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:18
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

TMR0 (Slobodni brojač)

- Pomoću njih se stvara relacije između realne veličine kao što je vreme, sa promenljivom koja predstavlja stanje brojača unutar mikrokontrolera
- PIC16F84 ima 8-bitni brojač
- Nakon svakog odbrojavnja do 255 brojač resetuje svoju vrednost na nulu i kreće sa novim ciklusom brojanja do 255. Prilikom svakog prelaska sa 255 na nulu, setuje se bit T0IF u INTCON registru. Ukoliko je dozvoljena pojava prekida, ovo se može iskoristiti za generisanje prekida i obradu prekidne rutine.
- Na programeru je da resetuje bit T0IF u prekidnoj rutini, kako bi se mogao detektovati novi prekid, tj. novo prekoračenje

Watchdog timer (Sigurnosni brojač)

- Mehanizam mikrokontrolera kojim se mikrokontroler brani od zaglavljivanja programa. Kao i kod svakog elektronskog kola, i kod mikrokontrolera može doći do kvara ili nepravilnog rada.
- Kada se to desi mikrokontroler će stati sa radom i ostati u tom stanju sve dok ga neko ne resetuje.
- Zbog toga je uveden mehanizam watchdog koji, nakon određenog vremena, resetuje mikorokontroler (mikrokontroler sam sebe resetuje).
- Watchdog radi na jednostavnom principu: ako doše do prekoračenja brojača mikrokontroler se resetuje i kreće sa izvršavanjem programa iz početka.
- Sledeći korak je sprečavanje reseta u slučaju pravilnog rada, što se postiže upisom nule u WDT registar (instrukcijom CLRWDT) svaki put kada se približi svom prekoračenju, čime će program sprečiti reset sve dok se pravilno izvršava.
- Jednom, kada dođe do zaglavljivanja, nula neće biti upisana, doći će do prekoračenja WDT brojača i desiće se reset koji će mikorokontroler ponovo vratiti u pravilan rad.

Primer podešavanja preskalera

- Neka je radni takt mikrokontrolera 3.2768Mhz. Potrebno je podesiti vrednost OPTION registra tako da se svaki od displej osvežava frekvencijom od po 200Hz, pri čemu imamo 2 displeja.

$$\frac{2^{15} x\ 100\ Hz}{2^2 x 2^8 x 2^{n+1}} = 400\ Hz$$

2^{13} / 2^{n+11} / 1 / 2^2

```
n=2
OPTION_REG=1XX0 0010= 1000 0010 (nisu uključeni otpornici na PORTB)
MOVLW 0x82 (učitamo konstanu u radni registar)
MOVWF OPTION_REG (prebacimo u OPTION registar iz radnog)
```

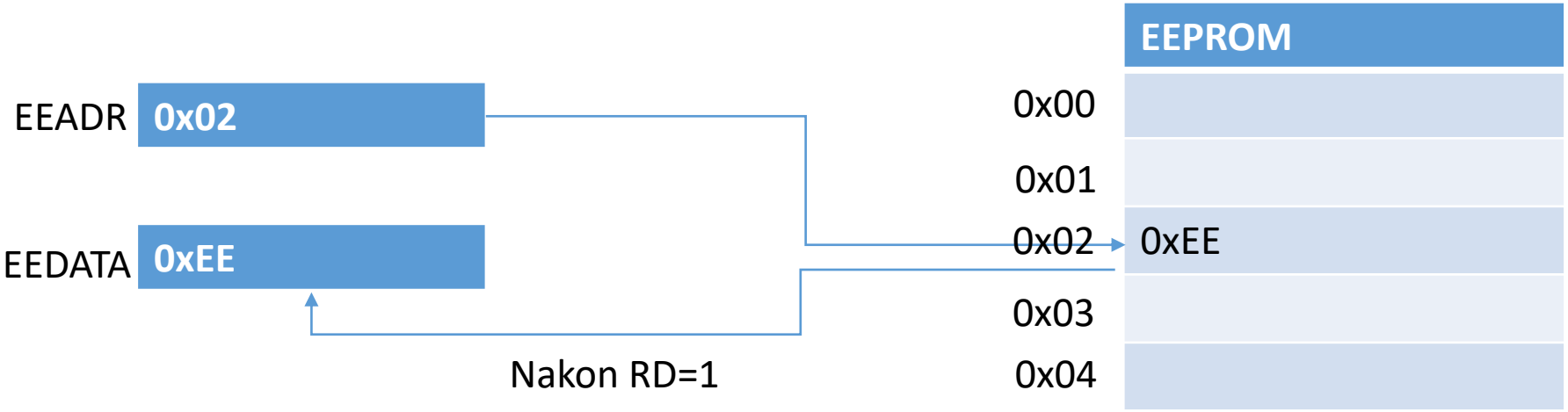
EEPROM memorija

- EEPROM memorija se nalazi u posebnom memorijskom prostoru i pristupa joj se preko specijalnih registara.
- Registri za rad sa EEPROM memorijom su:
  - EEDATA
    - sadrži podatak koji je pročitan ili koga treba upisati.
  - EEADR
    - sadrži adresu EEPROM lokacije kojoj se pristupa.
  - EECON1
    - sadrži kontrolne bitove, statuse i intertupt flag
  - EECON2
    - ovaj registar ne postoji fizički i služi da zaštiti EEPROM od slučajnog upisa.

EECON1 registar

- EEIF
  - Interrupt flag za detekciju prekid nakon upisa u EEPROM
- WRERR
  - Greška pri upisu
- WREN
  - Omogućiti upis
- WR
  - Izvršiti upis
- RD
  - Izvršiti čitanje

Čitanje EEPROM memorije



Upis u EEPROM memoriju

Prekid prilikom završetka upisa u EEPROM

- Prekid može biti uključen/isključen setovanjem/resetovanjem EEIE bita u INTCON registru.
- Ovaj prekid je čisto praktične prirode. Kako upis u jednu lokaciju EEPROM-a traje oko 10ms (što je za pojmove mikrokontrolera veoma dugo), to se mikrokontroleru ne isplati da čeka da se taj upis završi, već je dodat mehanizam prekida po kome on može da nastavi sa izvršenjem glavnog programa, dok se u pozadini vrši upis u EEPROM.
- Kada se upis završi, prekid obaveštava mikrokontroler da je upis gotov. Bit EEIF, kojim se ovo obaveštenje vrši, nalazi se registru EECON1. Pojava prekida može biti onemogućena resetovanjem EEIE bita u INTCOM registru.

Direktno adresiranje

5. Bit iz STATUS registra

RP1	RP0
-	0 – BANK0
-	1 – BANK1

Adresni bitovi iz instrukcije

7	6	5	4	3	2	1	0

Selektovana banka

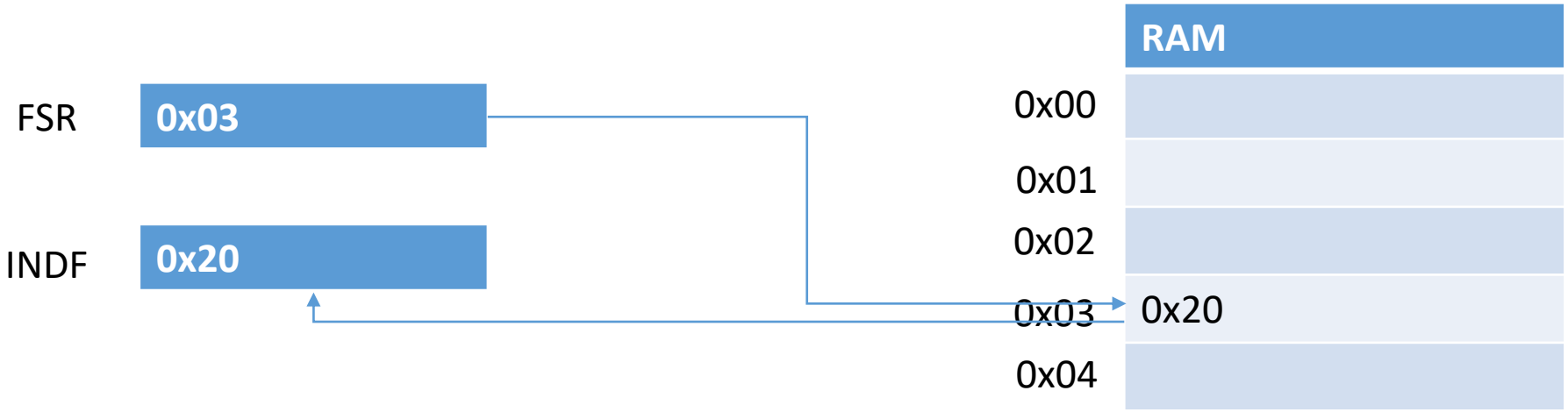
Selektovana lokacija

BANK0	BANK1
...	...
...	...
...	...

FSR i INDF registri

- Indirektno adresiranje se ostvaruje pomoću INDF i FSR registra
- FSR čuva adresu
- Korišćenjem INDF dobijamo vrednost na adresi iz FSR
- Primer: Ako na adresi 0Fh imamo vrednost 20, upisom vrednosti 0Fh u registar FSR dobićemo pokazivač na registar na adresi 0Fh, a čitanjem iz registra INDF dobijamo vrednost 20, što znači da smo iz prvog registra pročitali njegovu vrednost a da mu nismo direktno pristupili (već preko FSR i INDF).
- Indirektno adresiranje je veoma pogodno kada se vrše operacijama sa nizovima podataka koji su smešteni u okviru GPR registra. U tom slučaju je na početku potrebno inicijalizovati registar FSR na vrednost adrese prvog člana niza, a zatim se narednim članovima niza pristupa uvećanjem registra FSR.

Indirektno adresiranje



Mašinski jezik mikrokontrolera PIC16F84

- RISC, redukovan skup instrukcija
- 14-bit instrukcije
- U zavisnosti od tipa, mogu sadržati sledeće elemente
  - f-Memorijski (fajl) registar (7-bit)
  - d-Destinacioni bit (1-bit)
    - d=0 :destinacija akumulator (w)
    - d=1 :destinacija memorijski registar
  - b-kod odgovarajućeg bita u registru (3-bit)
  - k-konstanta (literal)
    - 8-bit za operacije za rad sa memorijskim registrima
    - 11-bit za adrese bezulsovnih skokova

Tipovi mašinskih istrukcija PIC16F84

- Tri grupe:
  - Bajt-orijentisane za rad sa memorijskim registrima
  - Bit-orijentisane za rad sa memorijskim registrima
  - Kontrolne instrukcije i rad sa literalima
- Sve instrukcije se izvršavaju u jednom taktu procesora, izuzev
  - Uslovnih i bezuslovnih skokova
  - Poziva i povratka iz potprograma
  - Ove instrukcije se izvršavaju u dva takta

Bajt-orijentisane naredbe za rad sa memorijskim registrima

13 8	7	6 0
Operacioni kod	Destinacija (d)	Fajl registar (f)

- MOVF f,d
  - d=0->w:=f
  - d=1->f:=f (korisno za testiranje ZERO FLAG)
- SWAPF f,d
  - Niža i viša četvorka osmobitnog REGISTRA menjaju mesta
- ADDWF, ANDWF, SUBWF, IORWF, XORWF
  - Za SUBWF f,d recimo, kada je d=0 efekat je w:=f-w, a za d=1 f:=f-w
- COMF REGISTAR,d
  - d=0->w:=komplement(REGISTAR)
  - d=1->REGISTAR:=komplement(REGISTAR)
- INCF f,d/DECF f,d
- RRF f,d
  - Sadržaj registra se rotira za jedan bit udesno kroz CARRY FLAG.
- RLF f,d
  - Sadržaj registra se rotira za jedan bit ulevo kroz CARRY FLAG.

Bit-orijentisane naredbe za rad sa memorijskim registrima

13 10	9 7	6 0
Operacioni kod	Bit kod (b)	Fajl registar (f)

- BTFSS f,b
  - Testira bit b u registru f. Preskače narednu instrukciju ako je f[b]==1.
- BTFSC f,b
  - Testira bit b u registru f. Preskače narednu instrukciju ako je f[b]==0.
- BSF f,b
  - Postavlja b-ti bit registra na vrednost 1.
- BCF f,b
  - Postavlja b-ti bit registra na vrednost 0.

Rad sa literalima

13	8	7	0
Operacioni kod		Literal (k)	

- k je u ovom slučaju 8-bit konstanta
- MOVLW k
  - w:=k
- ADDLW k, ANDLW k, SUBLW k, IORLW k, XORLW k
  - w:=w+k
  - w:=w and k
  - w:=k-w
  - …
- RETLW k
  - Povratak iz interapta uz w:=k

Rad sa literalima

13 11	10 0
Operacioni kod	Apsolutna adresa skoka (k)

- k je u ovom slučaju 11-bit konstanta
- GOTO k
  - w:=k
  - Bezuslovni skok
- CALL k
  - Poziv potprograma

CONFIGURATION WORD

Bit	Uloga
13-4: CP Code protection	**1- žaštita isključena 0 – programska memorija zaštićena**
3: PWRTE-	**1- isključen power-up timer 0- omogućen power-up timer**
2: WDTE	**1- watchdog uključen 0- watchdog isključen**
1-0: FOSC1 i FOSC0	Selekcija tipa oscilatora **00 - RC** **01 - HS** **10 - XT** **11 - LP**

Stek

- 13-bitni stek (Stack) sa 8 nivoa ili drugim rečima, grupisanih 8 memorijskih lokacija širine 13 bita sa posebnom namenom.
- Njegova osnovna uloga je da sačuva vrednost programskog brojača nakon što se iz glavnog programa skoči na adresu podprograma koji se izvršava.
- Da bi program znao da se vrati na mesto odakle je pošao, mora sa steka da vrati vrednost programskog brojača. Pri prelasku iz programa u podprogram, programski brojač se potiskuje na stek (Primer je instrukcija CALL), a pri izvršenju instrukcija kao što su RETURN, RETLW ili RETFIE koje se izvršavaju na kraju podprograma, vraća sa steka da bi program mogao da nastavi tamo gde je stao pre nego što je bio prekinut.
- Ove operacije stavljanja i vraćanja sa steka programskog brojača u žargonu se nazivaju PUSH i POP po instrukcijama koje pod istim imenom postoje na nekim većim mikrokontrolerima.

Programski brojač

- Programski brojač (PC) je 13-to bitni registar koji sadrži adresu instrukcije koja se izvršava.
- Fizički se realizuje pomoću petobitnog registra PCLATH koji predstavlja pet viših bitova adrese i osmobitnog registra PCL koji predstavlja nižih osam bita adrese.
- Njegovim uvaćanjem ili promenom (npr. u slučaju skoka) mikrokontroler izvršava jednu po jednu instrukciju programa.

Selekcija memorijske banke

- Za pristup specijalnim registrima potrebno je pored navođenja imena registra u okviru instrukcije prethodno selektovati odgovarajuću banku.
- Selekcija banke se može vršiti i pomoću direktive banksel posle koje se navodi ime registra kome se pristupa. Na ovaj način nema potrebe da se pamti koji je registar u kojoj banci.
  - banksel TRISB ; Pistupi banci u kojoj je TRISB clrf TRISB ; Izvrsi operaciju nad registrom TRISB
  - banksel POTRB ; Pristupi banci u kojo je PORTB clrf PORTB ; Izvrsi operaciju nad registrom PORTB
- Primer: bcf STATUS,RPO
  - Efekat: Instrukcija BCF resetuje bit RPO (RP0=0) u STATUS registru i time selektuje banku 0, koja ostaje selektovana sve dok se RP= ne postavi na jedan.
- Primer: bsf STATUS,RPO Instrukcija BSF setuje bit RP0
  - Efekat: Instrukcija BSF setuje bit RP0 (RP0=1) u STATUS registru i time selektuje adresiranje registara iz banke1.

Čitanje EEPROM memorije

- Setovanje bita RD inicira prenos podataka sa adrese koja se nalazi u registru EEADR u EDATA registar.
- Kako za čitanje podataka nije potrebno vreme kao za upis, preuzeti podatak iz EEDATA registra može se već u narednoj instrukciji koristiti dalje.
- Primer čitanja sadržaja EEPROM memorije

```
bcf   STATUS, RPO  ;bank0, jer je EEDAR na 09h
movlw   0x00     ;adresa lokacije koja se čita
movwf   EEADR    ;adresa se prebacuje u EEADR
bsf   STATUS, RPO  ;bank1 jer je EECON1 na 88h
bsf   EECON1, RD  ;čitanje uz EEPROM-a
bcf   STATUS, RPO  ;Bank0 jer je EEDATA na 08h
movf   EEDATA, W  ;W <-- EEDATA
```

- Nakon poslednje programske instrukcije, sadržaj sa adrese nula EEPROM se nalazi u radnom registru w.

Incijalizacija prekida

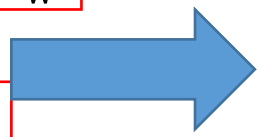
```
clrf    INTCON  ; svi prekidi onemogućeni
movlw   B'00010000' ; omogućen samo spoljni prekid
movwf   INTCON
bsf   INTCON,  GIE ; dozvoljena pojava prekida
```

Čuvanje sadržaja važnih registara

- Za vreme prekida, samo se povratna vrednost programskog brojača čuva na steku (pod povratnom vrednošću programskog brojača podrazumeva se adresa instrukcije koja je trebala da se izvrši, ali nije se izvršila jer se prekid desio)
- Procedura snimanja važnih registara pre odlaska u prekidnu rutinu u žargonu se naziva ''puš'' (PUSH), dok se procedura vraćanja snimljenih vrednosti naziva ''pop'' (POP). PUSH i POP su instrukcije kod nekih drugih mikorokontrolera (Intel), ali su toliko prihvaćene da se po njima naziva čitava operacija. PIC16F84 nema instrukcija kao što su PUSH i POP i one se moraju programski napraviti
- Za razmenu podataka između registara koristi se instrukcija SWAPF umesto MOVF jer ona ne utiče na stanje bitova STATUS registra
- SWAP f,d : zamena više i niže tetrade u registru f

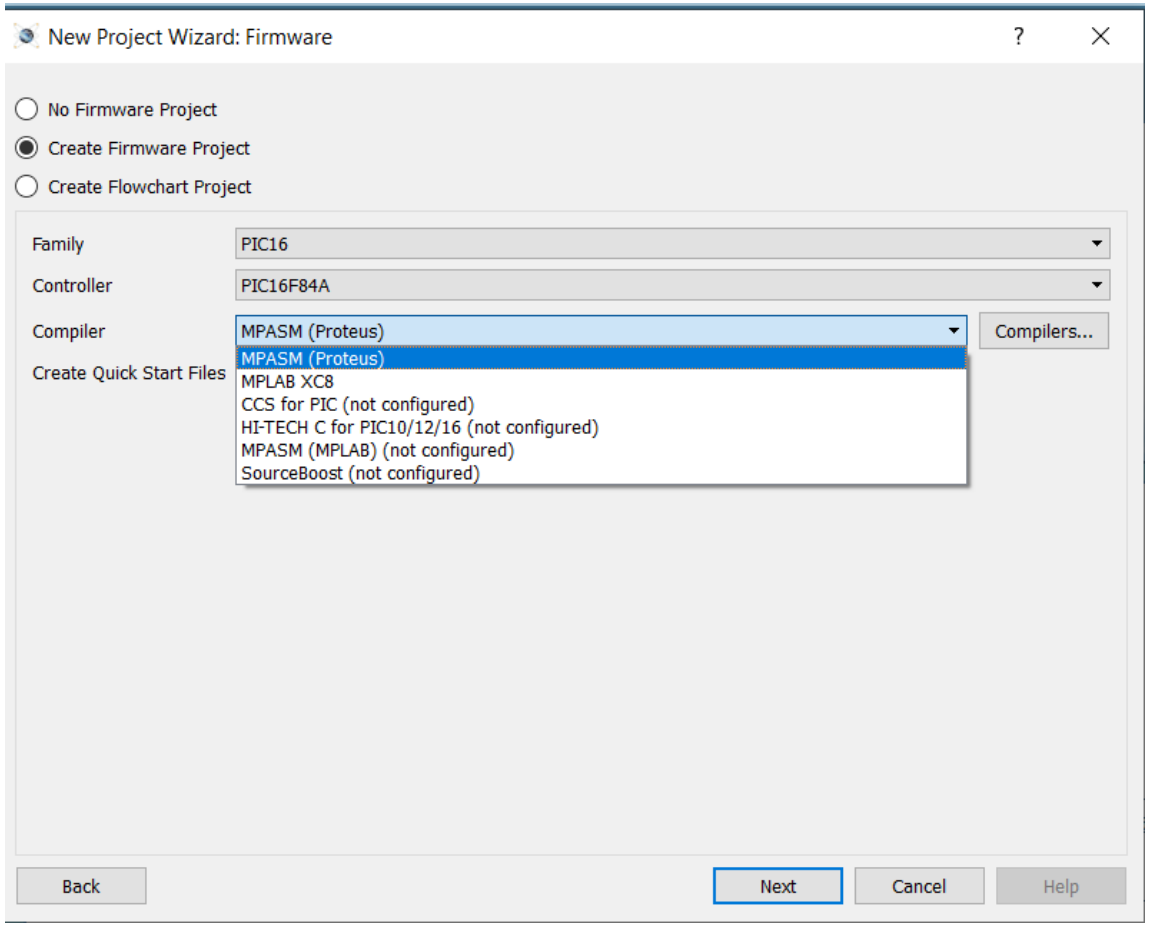
Šablon prekidne procedure

- Čuvanje konteksta
  - `movwf WREG_TEMP`
  - `swapf STATUS, w`
  - `clrf status`
  - `movwf STATUS_TEMP`
- Testuratu flegove interaptova
- Izvršiti obradu
- Restauracija konteksta
  - `swapf STATUS_TEMP, w`
  - `movwf STATUS`
  - `swapf WREG_TEMP, f`
  - `swapf WREG_TEMP, w`
  - `retfie`



SWAPF ne utiče na FLAG-ove u STATUSU!

PIC16F84A u Proteusu



- Family
  - PIC16
- Controller
  - PIC16F84A
- Compiler
  - Instalirati MPLAB XC kompajler
    - https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers
  - MPASM – PIC assembler
  - XC8 – C programski jezik

Primer 1 – ASM

- Sistem zasnovan na PIC16F84A
- Žuta dioda povezana na pin RA0
- Kod u asembleru

```
#include p16f84a.inc
; RESET and INTERRUPT VECTORS

; Reset Vector
RST   code  0x0
      goto  Start

; CODE SEGMENT

PGM   code
Start
        banksel PORTA
        movlw 0x00
        movwf   PORTA
        movwf   PORTB
        banksel TRISA
        movlw   0x00
        movwf   TRISA
        movlw   0x00
        movwf   TRISB
        bcf     STATUS,RP0
        movlw   0x01
        movwf PORTA
        Loop
            goto  Loop

END
```

Primer 1 – XC8

- Sistem zasnovan na PIC16F84A
- Žuta dioda povezana na pin RA0
  - Svakih 100ms se naizmenično pali i gasi
- Kod u XC8

```
#include<htc.h>
// Config word
__CONFIG(FOSC_HS & WDTE_OFF & PWRTE_ON & CP_OFF);

// Define LED pin
#define LED  RA0

//CPU takt
//Mora da se definise ako se koristi  __delay_ms()
#define _XTAL_FREQ   20000000


// Main function
void main()
{
      TRISA0  = 0;// RA0 pin izlazni
      LED    = 0; // Poslati signal 0 da se ugasi

      while(1)
      {
            __delay_ms(100);   // 100ms pauza
            LED = 0;           // Ugasi diodu
            __delay_ms(100); // 100ms pauza
            LED = 1;         // Upali diodu
      }
}
```

Šema povezivanja