Sistem prekida

Katedra za elektroniku

Sistem prekida

- Svaki procesor je u suštini **sekvencijalna mašina**
- Upravljačka jedinica **procesora prihvata, dekoduje i izvršava instrukcije** iz programske memorije prema redosledu koji je naveden u programu
- Imajući ovo u vidu jedan procesor u svakom trenutku može da izvršava samo jednu instrukciju
- Ukoliko se javi potreba za **servisiranjem periferija** od strane procesora, izvršavanje programa se mora prekinuti i programska kontrola se mora preneti na poseban blok instrukcija koji je napisan na takav način da na efikasan način može da opsluži posmatranu periferijsku jedinicu
- Kao primer, razmotrimo šta je potrebno da se opsluži zahtev izdat od strane tastature

Sistem prekida

- Tastatura se na najnižem nivou može posmatrati kao niz tastera, pri čemu sotfver daje odgovarajuće značenje svakom tasteru
- Tasteri su organizovani u **mrežnu strukturu** tako da svaki pritisnuti taster rezultuje u generisanju različitog koda na izlazu tasture
- Nakon svakog pritiska, procesor mora preuzeti generisani kod sa tastature
- Pod terminom *opsluživanje tastature* podrazumevamo akciju preuzimanja koda nakon svakog pritiska tastature i njegovo prosleđivanje programu koji će zatim interpretirani primljeni kod
- Kada govorimo o servisiranju periferijskih jedinica, postoje dva pristupa:
    - Servisiranje bazirano na prozivkama (engl. polling)
    - Servisiranje bazirano na prekidima (engl. interrupt handling)

Servisiranje bazirano na prozivkama

- Kod ovog načina servisiranja procesor neprekidno “ispituje” ili “proziva” status periferijske jedinice kako bi utvrdio da li ona zahteva servisiranje
- Kada utvrdi da periferijska jedinica zahteva servisiranje, procesor preduzima potrebne akcije kako bi servisirao periferiju
- Da bi ilustrovali princip servisiranja baziranog na prozivkama pretpostavimo da je naš zadatak da odgovaramo na telefonske pozive i preuzimamo poruke
- Pretpostavimo dalje da nemamo informaciju o tome kada će pozivi doći, i da na raspolaganju imamo telefon kome ne radi zvono, niti ima vibraciju tako da nemamo nikakav način da ustanovimo da je poziv stigao
- U ovom slučaju jedini način da saznamo da li je neki poziv stigao je da periodično podižemo slušalicu i proverimo da li je neko na liniji

Servisiranje bazirano na prozivkama

- Ovo bi bio prilično frustrirajući posao, pogotovo ako imamo i neke druge obaveze
- Međutim, ukoliko ne želimo da propustimo ni jedan poziv, moramo obustaviti sve ostale aktivnosti i posvetiti se neprekidnom ponavljanju sledeće sekvence:

  1) podizanju slušalice,
  2) prinošenju slušalice uvu,
  3) proveri da li je neko na liniji

- Obzirom da linija mora biti slobodna, kako bi poziv mogao da dođe, na kraju svake sekvence moramo spustiti slušalicu i ponoviti celu sekvencu od početka

- Kakvo gubljenje vremena! Ali to je prinicip rada sistema baziranog na prozivkama

Servisiranje bazirano na prekidima

- Kod ovog sistema, svaka periferijska jedinica koristi poseban signal da signalizira procesoru kada ima potrebu da bude servisirana
- Ovaj signal poznat je pod nazivom ***zahtev za prekidom*** (interrupt request, IRQ)
- **Procesor** može biti zauzet obavljanjem nekih drugih aktivnosti, ili može biti u režimu spavanja, međutim kada stigne neki od zahteva za prekidom, ukoliko su oni dozvoljeni, procesor će obustaviti aktivnost koju je trenutno izvodio kako bi se posvetio obradi prekidnog zahteva, izvršavajući specifičan blok instrukcija
- Ovaj događaj se zove ***prekid***
- Blok instrukcija koje se izvršavaju u cilju opsluživanja periferijske jedinice, kao odgovor na zahtev za prekidom, naziva se ***prekidni potprogram*** (**interrupt service routine, ISR**)

Servisiranje bazirano na prekidima

- Razmotrimo kako bi radio sistem za odgovaranje na pozive u slučaju da je baziran na sistemu prekida
- Pretpostavimo da naš telefon ima zvono pomoću kojega nam može signalizirati da je stigao novi poziv
- Dok čekamo na poziv, možemo obavljati neke druge aktivnosti, jer znamo da će nam zvono signalizirati dolazni poziv
- Kada se zvono oglasi, zaustavićemo tekuću aktivnost kako bismo podigli slušalicu i prihvatili poruku
- U ovom slučaju zvono predstavlja **naš indikator zahteva za prekidom**
- Jasno je da je ovaj sistem daleko efikasniji metod prijema poruka od prethodnog

Vrste prekida

- U opštem slučaju **prekidi unutar mikroračunarskog sistema** dele se u dve kategorije:

  - Maskirajući prekidi – ovi prekidi se mogu zabraniti (maskirati) od strane programera. Način maskiranja varira od procesora do procesora. Na ovaj način moguće je kontrolisati koji izvori prekida mogu prekinuti procesor u datom trenutku.

  - **Nemaskirajuće prekide** – ovi prekidi ne mogu biti zabranjeni od strane programera i svaki put kada se generiše neki prekid ovog tipa procesor će zaustaviti izvršavanje tekuće akcije i preći na opsluživanje prekida. Ovi prekidi su obično rezervisani za signalizaciju kritičnih događaja u sistemu koji se moraju opslužiti odmah.

Maskirajući i nemaskirajući prekidi

- Kao što je ranije rečeno, svi zahtevi za prekidom mogu se podeliti u dve grupe:

  1) Maskirajuće zahteve za prekidom
  2) Nemaskirajuće zahteve za prekidom

- Većina zahteva za prekidom u sistemu spadaju u klasu maskirajućih zahteva i prosto se nazivaju prekidima

- U slučaju maskirajućih zahteva za prekidom postoji odgovarajući mehanizam pomoću kojega programer može da zabrani (maskira) odgovarajući zahtev za prekidom

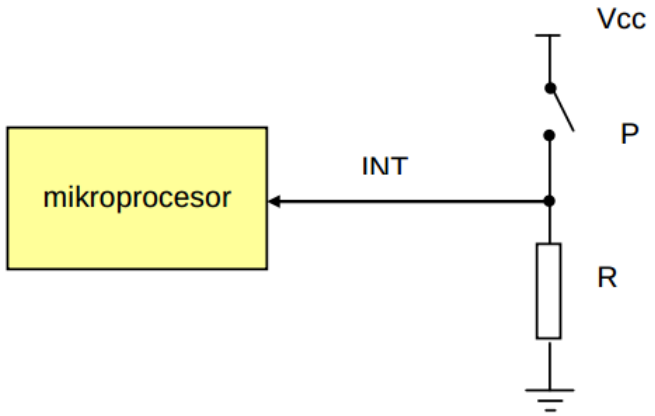
- Najčešće se to postiže postavljanjem ili brisanjem odgovarajućih bitova koji se nalaze unutar registra (Global Interrupt Enable, GIE) ili nekog posebnog registra namenjenog za tu svrhu (Interrupt Enable Register)

Sistem prekida

- Prekidi predstavljaju mehanizam pomoću kojeg mikroprocesor, ili mikrokontroler može da **odreaguje na pojedine događaje** koji zahtevaju hitnu obradu. Ovo je pogotovo od značaja u vremenski kritičnim aplikacijama, gde vremena odziva moraju biti strikno ispoštovana.
- Podsistem prekida omogućava mikroprocesoru da pod dejstvom spoljnog ili unutrašnjeg dogadaja, prekine izvršavanje tekućeg programa i pređe na izvršavanje specijalizovanog potprograma namenjenog obradi tog dogadaja.
- Prekidi mogu biti:
    - Hardverski (eksterni) - Izazvani od strane periferijskog uređaja sa kojim je procesor povezan. Tipično, procesor se obaveštava o zahtevu za prekidom putem specijalizovanog **elektronskog signala (INT, engl. Interrupt).** Primeri događaja koji izazivaju zahtev za prekidom: pritisak tastera na tastaturi, pomeranje miša i sl.
    - Softverski (interni) - Izazvan specifičnim stanjem procesora (npr. deljenjem nulom), ili specijalizovanom instrukcijom koja dovodi do pojave prekida (engl. trap).
- Svakom izvoru prekida dodeljuje se zaseban potprogram za obradu. Broj hardverskih prekida ograničen je brojem linija za zahteve za prekidom (engl. **IRQ = Interrupt Request**).
- Prekidi su uobičajeno korišćena tehnika u multitaskingu, pogotovo u aplikacijama koje rade u realnom vremenu.

Načini reagovanja na eksterni događaj

PRIMER: Neka je mikroprocesor povezan sa prekidačem, kao što je prikazano na slici. Potrebno je momentalno detektovati uključenje prekidača i u tom slučaju izvršiti odgovarajuću proceduru.

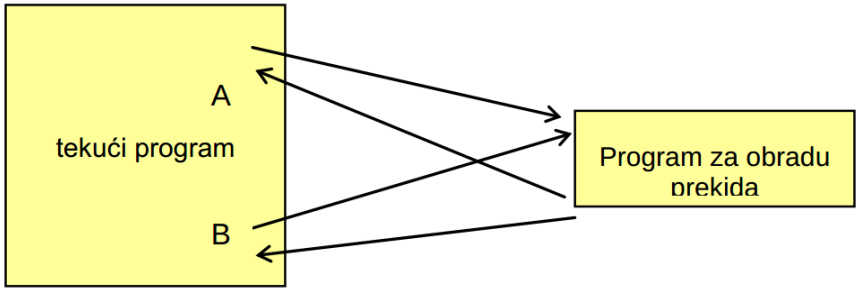


- Dva tipična pristupa rešavanju navedenog problema su:

1. Program se "vrti" u petlji tokom koje kontinualno proverava stanje ulazne linije i u slučaju detekcije logičke jedinice poziva specijalizovanu proceduru. Tokom izvršenja petlje, procesor nije u mogućnosti da obavlja neki drugi zadatak, što može znatno da umanji efikasnost sistema. Ovakav pristup se u programerskom žargonu naziva "prozivanje" (engl. polling).
2. Pritisak tastera detektuje specijalizovana hardverska jedinica - **kontroler prekida**. U međuvremenu, procesor izvršava neki zadatak manje hitnosti. Kada se pojavi zahtev za prekidom, procesor automatski prekida **trenutnu aktivnost, odrađuje potprogram za obradu prekida,** nakon čega nastavlja sa izvršenjem programa od istog mesta gde se nalazio u trenutku pojave prekida.

Potprogram za obradu prekida

- **Potprogram za obradu prekida** je deo programskog koda koji se automatski poziva po nastanku zahteva za prekidom. Naziva se još i prekidna rutina (engl. *Interrupt Service Routine* (ISR), *Interrupt Handler*).
- Program koji procesor izvršava onda kada nema prekida, u ovom kontekstu se naziva **tekući program**. Do prekida može doći tokom različitih faza izvršenja tekućeg programa. Instrukcija tokom čijeg izvršenja je nastao zahtev za prekidom naziva se **tačka prekida**.



- Potrebno je obezbediti mehanizam koji omogućava procesoru povratak u tačku prekida nakon izvršenja **prekidne rutine**. Ovo se postiže tako što se neposredno pre skoka na početak prekidne rutine trenutna vrednost **programskog brojača (PC)** koja predstavlja povratnu adresu, postavlja na **stek**.
- Na kraju prekidne rutine, povratna adresa se automatski očitava sa steka i smešta u programski brojač.

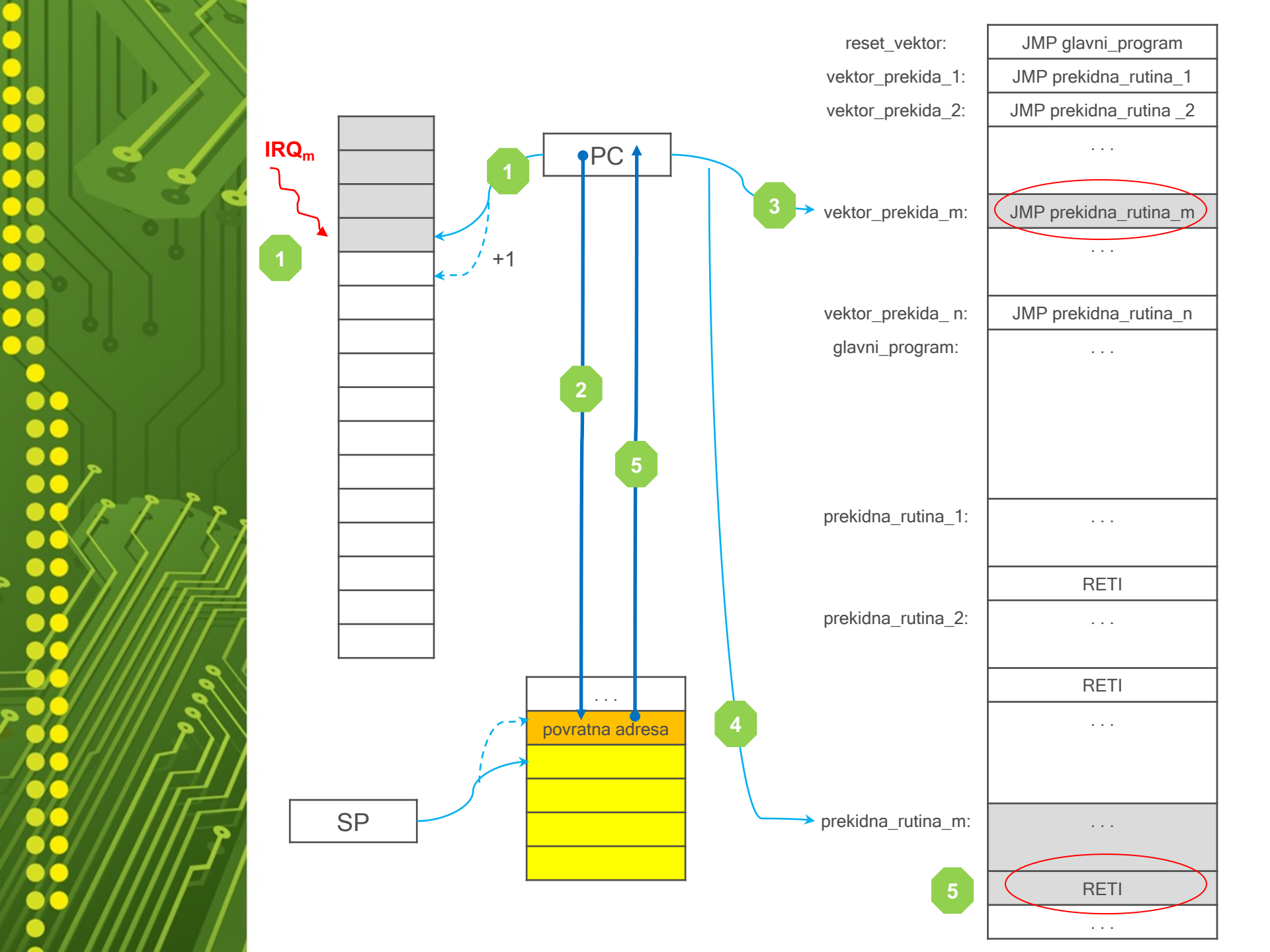
Višestruki izvori prekida

- U praksi je čest slučaj da postoje višestruki izvori prekida kao što su eksterni signali, serijski port, tajmeri i sl.
- U slučaju višestrukih izvora prekida, podsistem za prekide mora da obezbedi:
  - Potprograme (rutine) za obradu različitih prekida
  - Prepoznavanje izvora prekida
  - Rešavanje prioriteta prekida u slučaju istovremenog aktiviranja više prekidnih signala
  - Prelazak na rutinu koja odgovara izvoru prekida
  - Postupak u slučaju da se tokom obrade jednog prekida aktivira neki drugi signal prekida
- Svakom izvoru prekida pridružena su 2 bita posebne namene, koji pripadaju specijalizovanim kontrolnim registrima:
  - **Bit za maskiranje (dozvolu) prekida**, (engl. *Interrupt Enable*) - da bi prekid bio dozvoljen, ovaj bit mora biti setovan. Pored toga, uobičajeno je da postoji bit za globalnu zabranu prekida, čijim aktiviranjem se svi prekidi istovremeno zabranjuju.
  - **Indikator zahteva za prekidom** (engl. *Interrupt Flag*) - Ovaj bit se automatski setuje kada se desi događaj koji zahteva prekid.

Pozivanje potprograma za obradu prekida

- Na najnižim adresama u programskoj memoriji nalaze se tzv. vektori prekida.
- Po pojavi zahteva za prekidom **m** (u pitanju je jedan od **n** mogućih izvora prekida za dati procesor), automatski se obavljaju sledeće akcije:

1. U trenutku pojave zahteva za prekidom (IRQ_m), procesor izvršava instrukciju koja se nalazi na adresi na koju pokazuje programski brojač (**PC**). Procesor automatski inkrementira PC i proverava da li je prekid m dozvoljen. Ukoliko nije, izvršenje se nastavlja kao da se ništa nije desilo, a ukoliko jeste, prelazi na korak br. 2.
2. **Indikator zahteva za prekidom se automatski resetuje**, a istovremeno se setuje bit za **globalnu zabranu prekida**, čime je onemogućena obrada novog prekida sve do povratka iz prekidne rutine. Trenutna vrednost PC predstavlja **povratnu adresu**, koja se postavlja na stek.
3. U PC se upisuje **adresa vektora prekida** m.
4. Vrši se **bezuslovan skok** na početak prekidne rutine.
5. Izvršava se prekidna rutina, koja se završava instrukcijom **RETI**. Po nailasku na ovu instrukciju, povratna adresa se skida sa steka i upisuje u PC, čime se vrši povratak u glavni program. Pored toga, RETI instrukcija resetuje bit za **globalnu zabranu prekida**, čime je omogućena obrada novog prekida.


IRQm
1
1
+1
PC
2
3
4
5
5
. . .
povratna adresa
SP
reset_vektor:
JMP glavni_program
vektor_prekida_1:
JMP prekidna_rutina_1
vektor_prekida_2:
JMP prekidna_rutina _2
. . .
vektor_prekida_m:
JMP prekidna_rutina_m
. . .
vektor_prekida_ n:
JMP prekidna_rutina_n
glavni_program:
. . .
prekidna_rutina_1:
. . .
RETI
prekidna_rutina_2:
. . .
RETI
. . .
prekidna_rutina_m:
. . .
RETI
. . .