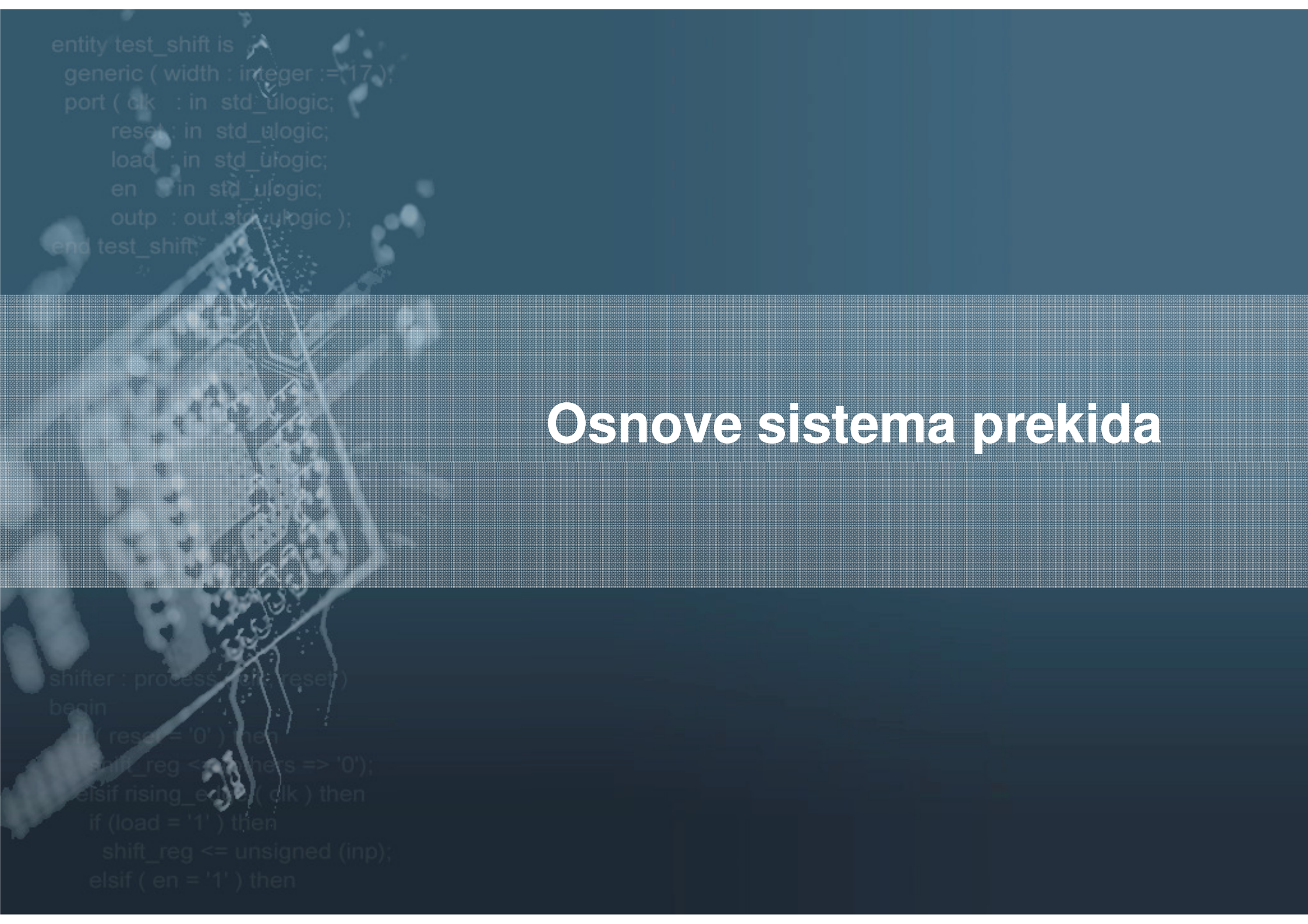
Mikroprocesorska elektronika

Predavanje VIII

Sadržaj predavanja

- Osnove sistema prekida
- Sistem prekida u mikrokontroleru Intel 8051
- Dizajn softvera za rad sa prekidima



Osnove sistema prekida

Prednosti embeded sistema baziranih na sistemu prekida

- Prekidi omogućavaju efikasno servisiranje zahteva za obradom generisanih u periferijskim jedinicama ili od strane okruženja embeded sistema
- Kada se uporedi sa sistemom prozivki, servisiranje zahteva bazirano na prekidima rezultuje u mnogo efikasnijem korišćenju centralnog procesora
- Pored ove prednosti, projektovanje embeded sistema baziranih na sistemu prekida ima i čitav niz dodatnih prednosti:
  - **Kompaktan i modularan softver** – prekidni programi (Interrupt Service Routine, ISR) nameću korišćenje modularizacije programa i njegovo ponovno korišćenje
  - **Smanjena potrošnja** – Kako korišćenje ISR rezultuje u izvršavanju manjeg broja CPU ciklusa, ovo ima direktan uticaj na količinu električne energije potrošenu od strane aplikacije
  - **Brže vreme odziva** – Kada u sistemu postoji veći broj različitih periferijskih jedinica, odnosno eksternih događaja, koje je potrebno servisirati, pažljivo projektovane ISR rutine obezbeđuju brzi odziv na zahteve za obradom. Pametno korišćenje sistema prioriteta prekida obezbeđuje brzo vreme odziva.

Osnovne komponente potrebne za implementaciju sistema prekida I

- Da bi mogao da koristi sistem prekida, projektant embeded sistema mora da uključi odgovarajuće hardverske i softverske komponente

- U zavisnosti od stepena integracije procesora na kome se bazira embeded sistem, hardverske komponente za podršku radu sa prekidima mogu se nalaziti:
  - Odvojene od CPU jedinice na posebnom integrisanom kolu, kao što je obično slučaj sa mikroprocesorima,
  - Zajedno sa CPU jedinicom, integrisane na istom čipu, kao što je obično slučaj kod mikrokontrolera

- Nezavisno od načina realizacije, kada postoji hardverska podrška radu sa sistemom prekida, potrebno je obezbediti korektno konfigurisanje hardvera i razviti odgovarajuće softverske module da bi se kompletirala implementacija sistema prekida u nekom embeded sistemu

Osnovne komponente potrebne za implementaciju sistema prekida II

- Zahtevi za prekidom najčešće su inicirani nekim hardverskim događajem
- Događaji kao što su pritisak tastera, dostizanje neke granične vrednosti ili isticanje nekog vremenskog intervala, su neki od primera događaja koji mogu da iniciraju pojavu zahteva za prekidom
- Kada se jednom neki hardverski događaj konfiguriše da generiše zahteve za prekidom, njihovo pojavljivanje je potpuno nepredvidivo i asinhrono
- Upravo iz ovog razloga softverska komponenta koja pruža podršku radu sa prekidima mora biti napisana imajući ovu činjenicu na umu

Načini signalizacije zahteva za prekidom

- Zahtev za prekidom uvek se sistemu za obradu prekida saopštava korišćenjem jednog električnog signala

- Generalno, postoje dva pristupa za inidikaciju postojanja zahteva za prekidom unutar embeded sistema:

  - Indikacija bazirana na korišćenju nivoa signala (level-sensitive)

  - Indikacija bazirana na korišćenju ivice signala (edge-sensitive)

Sistemi prekida bazirani na nivou signala

- Sistem za signalizaciju prekida baziran na nivou signala koristi nivo logičke nule ili logičke jedinice kao informaciju da postoji zahtev za prekidom
- Bez obzira na nivo signala koji se koristi za signalizaciju postojanja zahteva za prekidom, odgovarajući nivo signala mora da bude održavan sve dok se zahtev za prekidom ne primi od strane sistema za obradu prekida
- Na ovaj način se garantuje da će zahtev za prekidom biti „viđen“ od strane sistema za obradu prekida
- Neke periferijske jedinice su projektovane na takav način da automatski uklanjaju zahtev za prekidom kada im se pristupi
- Na primer, neki serijski komunikacioni kontroleri automatski uklanjaju indikaciju da je podatak primljen kada započne operacije čitanja podatka iz prijemnog registra

Sistemi prekida bazirani na ivici signala

- U sistemu za signalizaciju prekida baziranom na korišćenju ivice signala, prekid se signalizira pojavom rastuće ili opadajuće ivice
- U ovom sistemu sistem za obradu prekida mora da interno sačuva podatak o postojanju zahteva za prekidom, jer u protivnom zahtev za prekidom može proći neprimećen
- Većina savremenih mikrokontrolera dozvoljava da se za svaki ulazni port preko kojega se upućuje zahtev za prekidom specificira da li će taj zahtev biti saopšten pomoću nivoa ili ivice spojenog signala
- Obično unutar mikrokontrolera postoji poseban registar pomoću kojega je, konfigurisanjem individualnih bitova, moguće definisati način (nivo/ivica) saopštavanja zahteva za prekidom
- Kod nekih mikrokontrolera je u slučaju korišćenja ivice kao indikacije zahteva za prekidom moguće definisati da li će biti korišćena rastuća, opadajuća ili obe ivice

Softverski prekidi

- Zahtev za prekidom može biti iniciran od strane softvera, i u tom slučaju govorimo o softverskim prekidima
- Za razliku od hardverskih prekida, softverski prekidi su predvidljivi jer se generišu pod komandom programera
- Imajući ovo u vidu, softverski prekidi su ekvivalentni pozivu funkcije unutar programa

Maskirajući i nemaskirajući prekidi I

- Kao što je ranije rečeno, svi zahtevi za prekidom mogu se podeliti u dve grupe:
  - Maskirajuće zahteve za prekidom
  - 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 statusnog registra (Global Interrupt Enable, GIE) ili nekog posebnog registra namenjenog za tu svrhu (Interrupt Enable Register)

Maskirajući i nemaskirajući prekidi II

- Nemaskirajući zahtevi za prekidom ne mogu se zabraniti i kada se pojave procesor će pristupiti procesu obrade prekida
- Ovaj tip zahteva za prekidom tipično je rezervisan za kritične događaje u embeded sistemu, čija se obrada ne može odložiti
- Primer kritičnog događaja u sistemu mogla bi biti indikacija niskog nivoa baterije u nekom prenosivom uređaju
- Nakon što se detektuje nizak nivo baterije pomoću naponskog komparatora, generisao bi se nemaskirajući zahtev za prekidom
- Procesor bi započeo obradu ovog zahteva za prekidom i pristupio bi snimanju trenutnog stanja CPU-a kao i svih ostalih kritičnih podataka smeštenih u registrima i memoriji sa gubitkom sadržaja kako bi se sprečio gubitak podataka, i zatim bi započeo postupak gašenja sistema

Potreba za maskiranjem zahteva za prekidom

- Onemogućavanje, odnosno maskiranje, zahteva za prekidom je neophodna mogućnost prilikom rada sa prekidima
- Postoje trenuci i situacija kada procesor nije spreman da obrađuje nove zahteve za prekidom
- Na primer, nakon sistemskog reseta pre nego što se završi konfigurisanje čitavog sistema, procesor ne može da prihvata zahteve za prekidom jer sistem za obradu prekida još uvek nije konfigurisan
- Drugi primer kada nije dozvoljena obrada novih zahteva za prekidom je kada procesor počinje da izvršava prekidni podprogram

Dozvola/zabrana zahteva za prekidom unutar periferijskih jedinica I

- Pored odgovarajućih bitova unutar procesora koji se mogu koristiti za kontrolu dozvole/zabrane zahteva za prekidima, same periferijske jedinice takođe mogu posedovati mogućnost kontrole njihove sposobnosti da generišu zahteve za prekidima

- Na primer, razmotrimo slučaj tastera čiji pritisak izaziva pojavu zahteva za prekidom koji je povezan sa procesorom preko nekog globalnog ulazno/izlaznog porta (GPIO port)

- U ovom slučaju GPIO može da sadrži bit dozvole prekida (IEF, Interrupt Enable Flag) koji omogućava dozvolu/zabranu pojave zahteva za prekidom

- U ovom sistemu, da bi za zaista došlo do prekida procesora potrebno je da budu podešeni odgovarajući bitovi za dozvolu prekida unutar samog procesora ali i IEF bit unutar GPIO porta

Dozvola/zabrana zahteva za prekidom unutar periferijskih jedinica II

- Postojanje bitova za dozvolu/zabranu prekida unutar periferijskih jedinica predstavlja zgodan način pojedinačne kontrole svakog od mogućih izvora prekida unutar embeded sistema bez uticaja na generisanje zahteva za prekidom u drugim delovima sistema

- Procesor, po pravilu, ne poseduje dovoljan broj unutrašnjih bitova za kontrolu dozvole/zabrane svih mogućih izvora prekida tako da na ovaj način nije moguća njihova pojedinačna kontrola

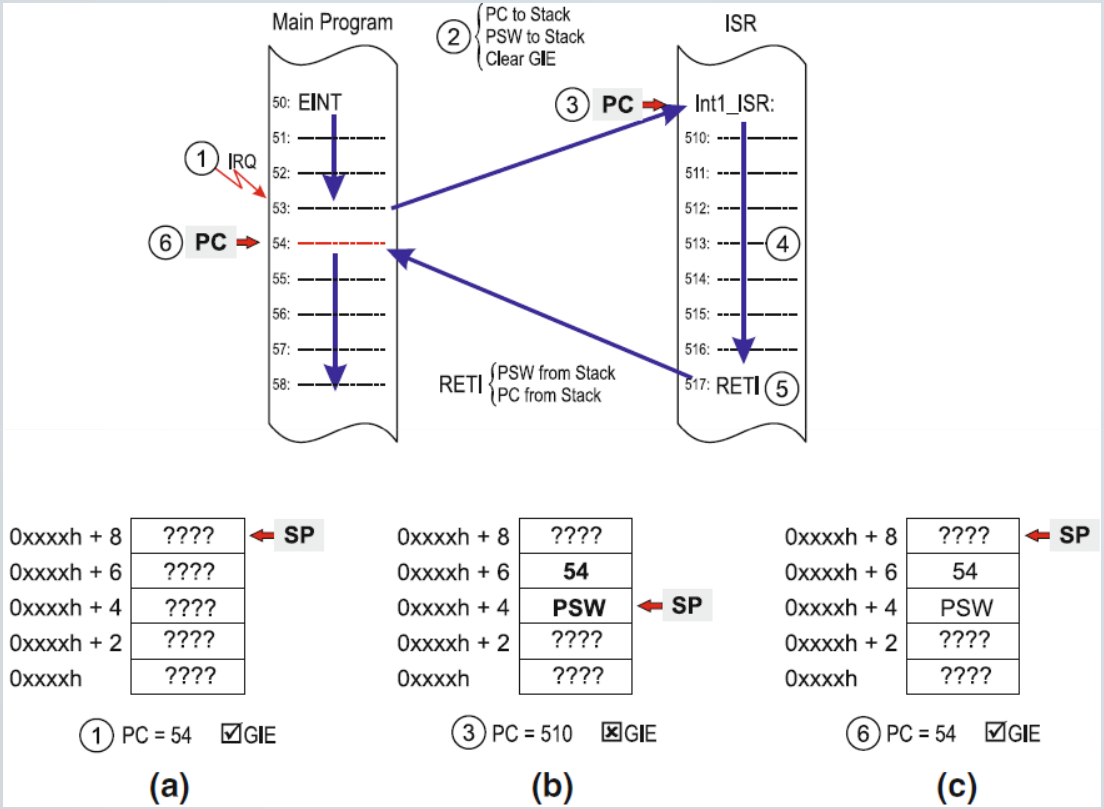
- U slučaju da procesor poseduje samo jedan, globalni, bit za dozvolu/zabranu prekida u sistemu možemo ili dozvoliti sve zahteve ili zabraniti sve zahteve za prekidom što u većini slučajeva ne pruža dovoljan nivo kontrole

Prozivka u sistemu prekida

- Vrlo često se i u sistemima baziranim na sistemu prekida javlja potreba za prozivkom pojedinačnih uređaja koji su mogli generisati zahtev za prekidom
- Ovo je redovno slučaj kada veći broj različitih događaja unutar sistema generiše jedan, zajednički, zahtev za prekidom
- Aktivacija jednog, zajedničkog, zahteva za prekidom obaveštava procesor da neka od periferijskih jedinica ima potrebu da komunicira sa procesorom ali procesor ne zna o kojoj konkretnoj jedinici je reč
- Da bi se rešio gore navedeni problem, većina periferijskih jedinica poseduje odgovarajuće statusne bite koji signaliziraju da je periferijska jedinica generisala odgovarajuće zahteve za prekidom
- Kada procesor dobije zajednički zahtev za prekidom, on redom proverava statusne bite u svim periferijskim jedinicama čiji su individulani zahtevi za prekidom povezani na zajednički zahtev za prekidom kako bi utvrdio koja je periferijska jedinica zahtevala komunikaciju sa procesorom

Sekvenca obrade zahteva za prekidom I

- Prilikom obrade zahteva za prekidom dešavaju se sledeći koraci:
  1. Zahtev za prekidom dolazi do procesora u trenutku kada on izvršava instrukciju iz glavnog programa na adresi 53. Nakon detekcije zahteva za prekidom procesor najpre kompletira izvršavanje instrukcije sa adrese 53.
  2. Procesor smešta trenutni sadržaj PC registra, koji pokazuje na adresu 54, sadržaj PSW registra na stek (vidi sliku b). Nakon toga procesor zabranjuje dozvolu svih novih prekida, brišući sadržaj GIE bita.
  3. U PC registar smešta se adresa prve instrukcije prekidnog podprograma (ISR) koji je asociran izvoru prekida preko kojega je stigao zahtev za prekidom. Način na koji se ovo određuje biće objašnjen kasnije.
  4. Procesor izvršava asocirani prekidni podprogram nailazeći na RETI instrukciju
  5. Izvršavanje RETI instrukcije izaziva postavljanje starih vrednosti PSW i PC registara, koristeći vrednosti sa steka
  6. Procesor nastavlja sa izvršavanjem glavnog programa, izvršavajući instrukciju sa adrese 54



Sekvenca obrade zahteva za prekidom II

- U prethodnoj sekvenci događaja, jedan aspekt je ostao neobjašnjen:

  Kako procesor zna koju početnu adresu prekidnog podprograma treba da smesti u PC registar kada naiđe zahtev za prekidom?

- U slučaju da u sistemu imamo samo jedan izvor prekida, ovaj problem je trivijalan, ali u slučaju da u sistemu imamo veliki broj različitih izvora prekida, od kojih svaki ima svoj asocirani prekidni podprogram, izbor pravog prekidnog podprograma koji je potrebno izvršiti postaje ozbiljan

- Drugi problem koji se može pojaviti u sistemima sa većim brojem izvora prekida je istovremeno stizanje više od jednog zahteva za prekidom

- Obzirom da je procesor sekvencijalna mašina, on u jednom trenutku može da opsluži samo jedan zahtev

Sekvenca obrade zahteva za prekidom III

- Da bi se rešili navedeni problemi u embeded sistemu moraju biti prisutni sledeći mehanizmi za upravljanje prekidima:

- Identifikacija izvora koji je generisao zahtev za prekidom kako bi se mogao izvršiti njemu asocirani prekidni podprogram

- Razrešavanje konflikta do kojega dolazi kada veći broj periferijskih uređaja istovremeno generiše zahteve za prekidom

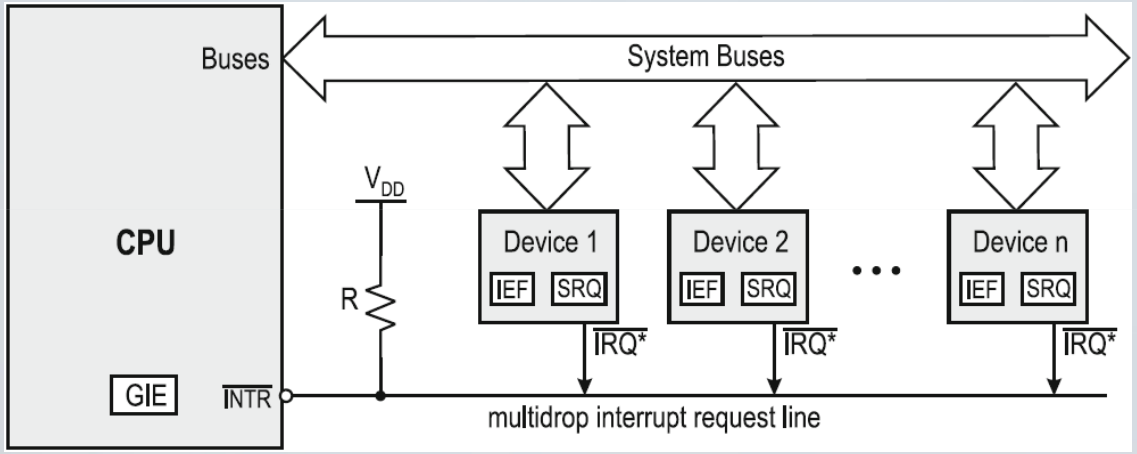
Postupci za identifikaciju izvora prekida

- Tokom evolucije embeded sistema, predloženi su različiti postupci za identifikaciju izvora zahteva za prekidom u slučaju kada u sistemu postoji više od jednog periferijskog uređaja koji može da generiše zahtev za prekidom

- U opštem slučaju, svi postupci mogu se klasifikovati u jednu od tri grupe:
  - Postupci bazirani na ne-vektorskim prekidima
  - Postupci bazirani na auto-vektorskim prekidima
  - Postupci bazirani na vektorskim prekidima

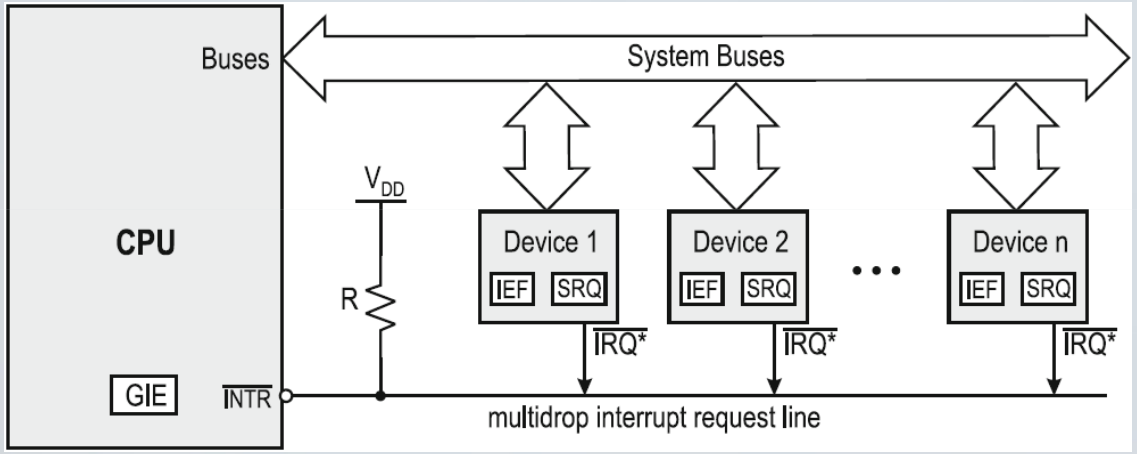
Ne-vektorski prekidi I

- Najjednostavniji način za obradu zahteva za prekidom je da sve periferijske jedinice upućuju zahteve preko jedne, zajedničke linije
- Kada bilo koja od periferija pošalje zahtev, INTR port procesora će se aktivirati i procesor će pristupiti obradi zahteva za prekidom
- Međutim, nadgledajući samo INTR port procesor nije u mogućnosti da sazna koja periferija je uputila zahtev za prekidom
- Stoga se unutar prekidnog podprograma mora pristupiti prozivci svih periferija koje su povezane na INTR liniju kako bi se, ispitujući njihove statusne bite (SRQ) utvrdilo koja od njih je uputila zahtev za prekidom
- Ovakvi sistemi se nazivaju ne-vektorski, jer svim mogućim zahtevima odgovara jedan prekidni podprogram



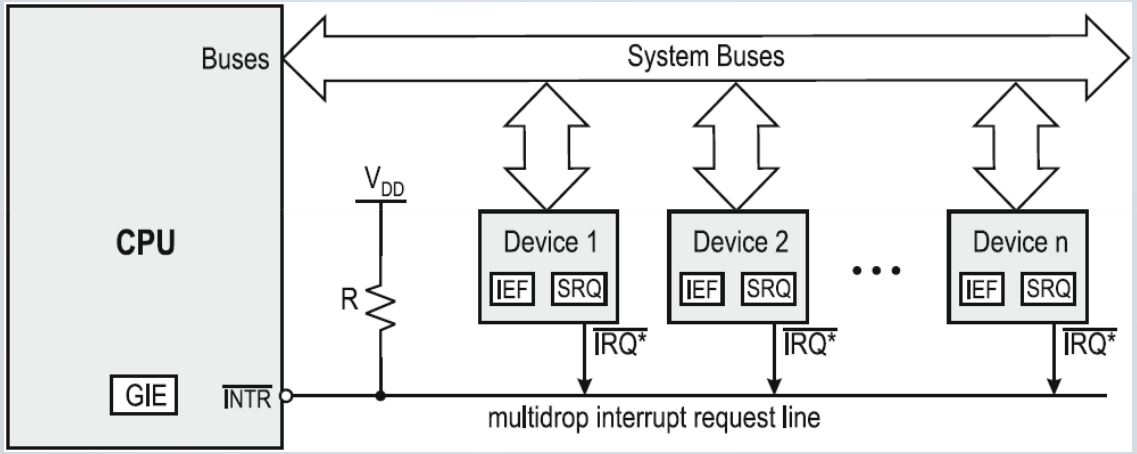
Ne-vektorski prekidi II

- Da bi sistem funkcionisao, svaka periferijska jedinica se mora povezati na zajedničku INTR liniju koristeći “open-drain” ili “open-collector” izlaz (IRQ*)
- Kada se unutar neke periferijske jedinice desi događaj koji treba da inicira generisanje zahteva za prekidom, ukoliko je njen IEF bit setovan, periferijska jedinica će aktivirati svoj IRQ* izlaz i setovati svoj SRQ bit
- Kada procesor detektuje da je INTR port aktivan, ukoliko je GIE bit setovan, prekinuće izvršavanje tekućeg programa i u PC registar smestiti početnu adresu prekidnog podprograma
- U sistemim sa ne-vektorskim prekidima ova adresa prekidnog programa je unapred utvrđena i fiksna, tako da se prilikom razvoja softvera prva instrukcija prekidnog podprograma mora smestiti na tu adresu



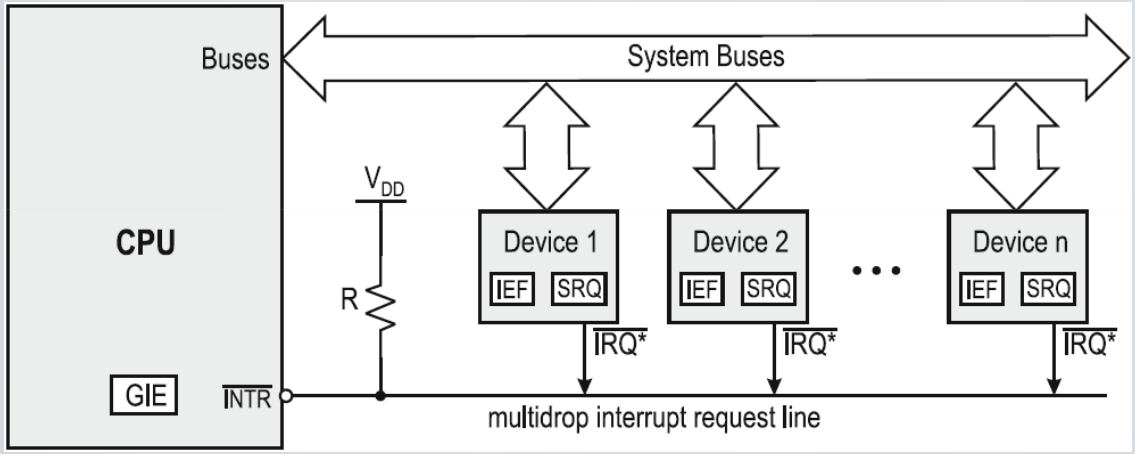
Ne-vektorski prekidi III

- Bez obzira na to koja je periferijska jedinica generisala zahtev za prekidom, procesor uvek izvršava isti prekidni podprogram
- Unutar prekidnog programa, procesor “proziva” svaku periferijsku jedinicu i proverava njen SRQ bit kako bi utvrdio koja od njih je generisala zahtev za prekidom
- Odsustvo hardverskog mehanizma koji bi procesoru omogućio da automatski identifikuje periferijsku jedinicu koja je generisala zahtev za prekidom je razlog zašto se ovakav sistem naziva ne-vektorski
- Neki od procesora umesto fiksne početne adrese prekidnog podprograma koriste fiksnu lokaciju na koju se smešta početna adresa
- Prilikom obrade prekida, procesor u PC registar upisu vrednost smeštenu na ovoj posebnoj lokaciji



Ne-vektorski prekidi IV

- Ukoliko se u sistemu pojavi veći broj istovremenih zahteva za prekidom, redosled “prozivanja” periferijskih jedinica određuje koji zahtev će prvi biti opslužen
- Na ovaj način, redosled “prozivanja” periferijskih jedinica određuje prioritet između različitih zahteva za prekidom
- Embeded sistemi bazirani na ne-vektorskim izvorima prekida, iako jednostavni za realizaciju, najčešće rezultuju u sporijoj obradi upućenih zahteva za prekidom
- Ovo je stoga što se prilikom svakog zahteva za prekidom prvo mora utvrditi koja periferijska jedinica je uputila zahtev



Vektorski prekidi I

- Efikasniji način za identifikaciju izvora prekida postignut je u sistemima koji koriste vektorske izvore prekida
- U vektorskom sistemu postoji hardverski mehanizam koji omogućava automatsku identifikaciju izvora prekida bez potrebe za “prozivkom” potencijalnih izvora
- Najčešće korišćeni mehanizam uključuje postojanje dodatne linije (INTA), preko koje procesor saopštava periferijskim jedinicama da je primio zahtev za prekidom
- Kada periferijska jedinica detektuje da je INTA signal aktivan, u slučaju da je ona generisala zahtev za prekidom, šalje identifikacioni kod (ID) nazad ka procesoru
- Svakoj periferijskoj jedinici pridružen je jedinstven ID kod

Vektorski prekidi II

- Analizirajući primljeni ID kod, procesor može odrediti koja periferijska jedinica je uputila zahtev za prekidom i početi izvršavanje odgovarajućeg prekidnog programa

- Zbog toga se ovi ID kodovi često nazivaju i prekidni vektori, a otuda i ime čitavom sistemu

- Kod vektorskih sistema, u sistemskoj memoriji postoji posebna zona u koju se smešta tabela u kojoj se nalaze asocijacije između svakog prekidnog vektora i početnih adresa prekidnih podprograma koji su namenjeni za njihovu obradu

- Ova memorijska zona se često naziva i sistemska vektor tabela

Vektorski prekidi III

- Vektorski pristup identifikaciji izvora prekida često se sreće kod mikroprocesora, gde se od periferijskih jedinica koje su generisale zahtev za prekidom očekuje da pošalju svoj ID kod procesoru preko magistrale podataka u trenutku kada detektuju aktivaciju INTA signala

- U većini slučajeva ID kod ne predstavlja vrednost početne adrese prekidnog podprograma koji je potrebno izvršiti

- ID kod se obično koristi da se pomoću njega izračuna vrednost početne adrese prekidnog podprograma, najčešće korišćenjem sistemske vektor tabele

- Intel x86 procesori koriste ovaj mehanizam

Auto-vektorski prekidi I

- Kod mikrokontrolera, gde veliki broj različitih periferijskih jedinica na čipu (tajmera, I/O portova, serijskih portova, A/D i D/A konvertora) može generisati zahteve za prekidom, često se koristi mehanizam koji se zove auto-vektorski
- Kod ovog mehanizma, svaka periferijska jedinica ima unapred pridružen vektor prekida, odnosno fiksnu adresu koja upućuje na početnu instrukciju prekidnog podprograma
- U ovom slučaju, procesor nema potrebe da generiše INTA signal niti periferijska jedinica mora da šalje svoj ID kod procesoru
- Kada se detektuje neki od zahteva za prekidom, procesor u PC registar upisuje početnu adresu prekidnog podprograma

Auto-vektorski prekidi II

- U zavisnosti da li je moguće modifikovati početne adrese prekidnih podprograma kod auto-vektorskih prekida postoje dve vrste mikrokontrolera:

- **Mikrokontroleri sa fiksnim početnim adresama prekidnih podprograma** - u PC registar upisuje se predefinisana početna adresa prekidnog podprograma. Početna adresa prekidnog podprograma ne može se menjati od strane programera.

- **Mikrokontroleri sa fiksnim vektorima prekida** – u PC registar se upisuje početna adresa prekidnog podprograma iz sistemske vektor tabele na osnovu vektora pridruženog dotičnom prekidu. Programer ne može menjati vektore prekida ali može menjati sistem vektor tabelu. Ovo pruža mogućnost programeru da proizvoljno organizuje pozivije prekidnih podprograma unutar memorijskog adresnog prostora mikrokontrolera

Prioriteti zahteva za prekidom I

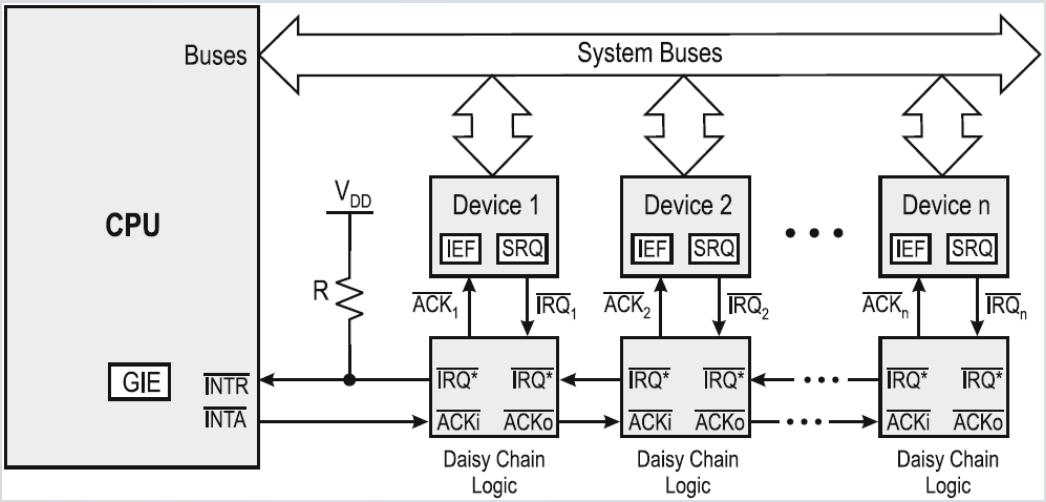
- U slučaju da postoji više periferijskih jedinica sa mogućnošću generisanja zahteva za prekidom unutar embeded sistema, moguće su situacije da dve ili više periferijskih jedinica istovremeno generišu zahteve za prekidom
- U ovom slučaju, procesor mora da poseduje neki mehanizam pomoću kojega će odrediti redosled obrade prekida, odnoso mora da utvrdi prioritet obrade pristiglih zahteva
- Ovo je neophodno obzirom da u svakom trenutku procesor može da izvršava samo jedan zadatak
- Postoje razni načini da se obezbedi sistem prioriteta unutar mikroračunarskog sistema, pri čemu mogu biti bazirani na softverskoj ili hardverskoj realizaciji

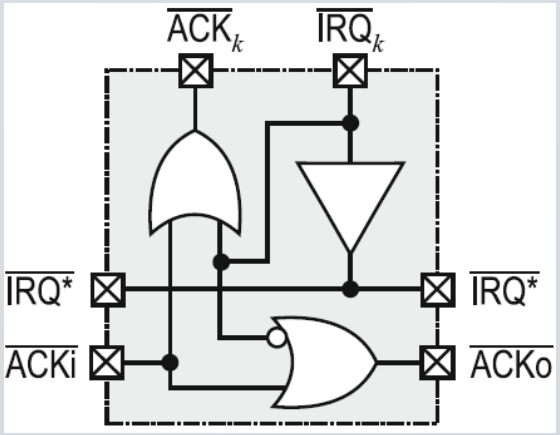
Prioriteti zahteva za prekidom II

- U slučaju ne-vektorskih sistema, prioritetizacija izvora prekida vrši se softverski
- Redosled obrade pristiglih zahteva za prekidom implicitno je određen redosledom “prozivke” periferijskih jedinica na početku izvršavanja prekidnog podprograma
- U ovakvim sistemima nepohodno je odrediti prioritet periferijskih jedinica pre nego što se odredi redosled “prozivke” kako bi se obezbedilo da periferijske jedinice sa “urgentnijim zahtevima” budu obrađene prve
- Isti koncept koristio bi se i u sistemima baziranim na prozivci
- Vektorski i auto-vektorski sistemi zahtevaju postojanje odgovarajuće hardverske podrške koja vrši arbitraciju (razrešavanje) prioriteta. U praksi postoje dva metoda arbitracije:
  - “Daisy Chain” arbitracija
  - Arbitracija bazirana na kontroleru prekida

“Daisy Chain” arbitracija I

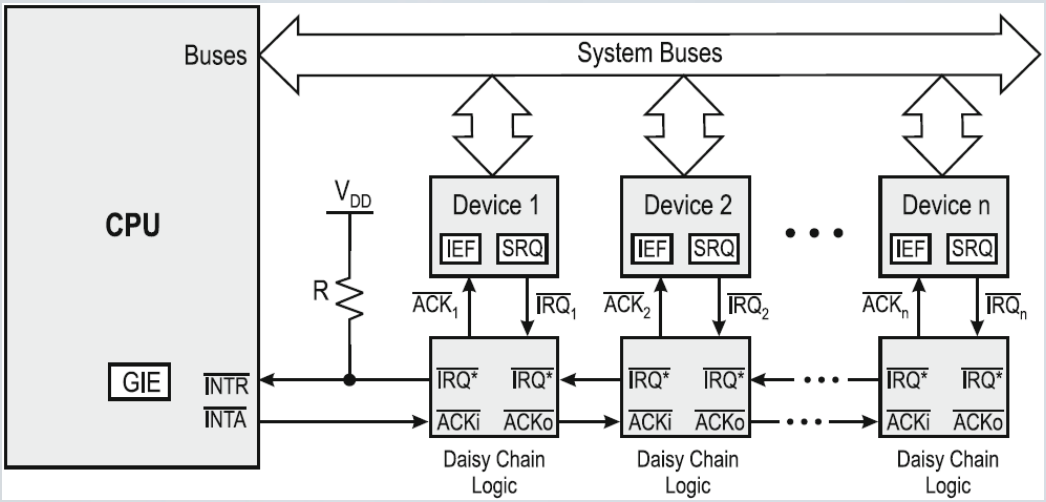
- Ovo je najjednostavniji metod za obezbeđivanje hardverske arbitracije prioriteta u embeded sistemu
- Pretpostavka je da sve periferijske jedinice koriste jednu, zajedničku, liniju za upućivanje zahteva za prekidom (INTR)
- Takođe sve periferijske jedinice povezane su na jednu, zajedničku, liniju potvrde prijema zahteva za prekidom (INTA)
- IRQ i ACK portovi svih periferijskih jedinica povezani su u jedan dugački lanac koji je na kraju povezan sa INTR i INTA portovima procesora
- Svai čvor u lancu sadrži logiku koja omogućava da se svaki zahtev za prekidom prosledi procesoru, ali i da se limitira propagacija INTA signala samo do periferije sa najvećim nivoom prioriteta koja je generisala zahtev

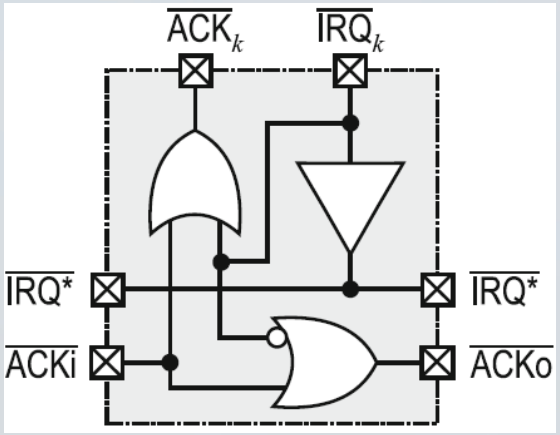




“Daisy Chain” arbitracija II

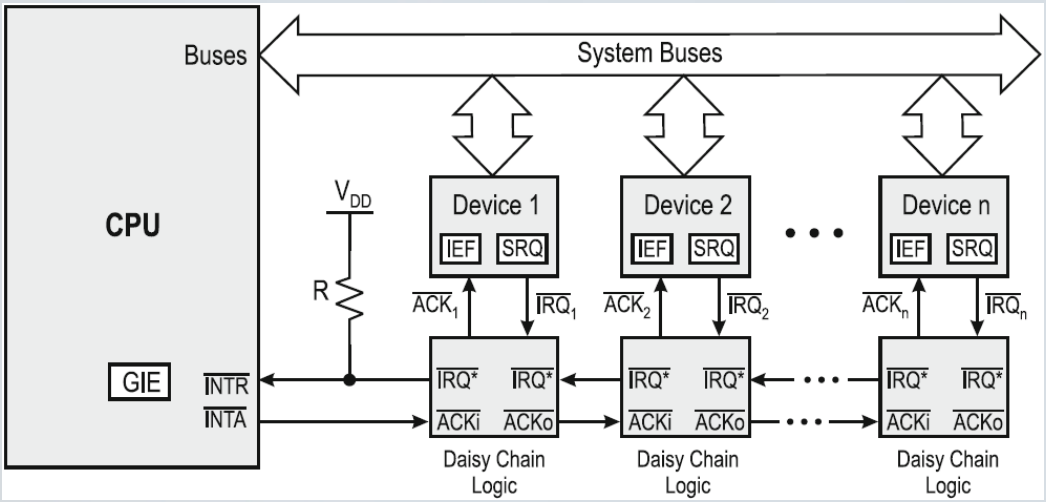
- Usled blokirajuće sposobnosti linka, prioriteti dodeljeni periferijskim jedinicama u direktnoj su vezi sa njihovom pozicijom u linku
- Periferije “bliže” procesoru imaju veći priorite od periferija koje su “udaljenije” od procesora
- U slučaju da dve periferijske jedinice, recimo periferije 1 i 2, istovremeno upute zahtev za prekidom, INTR linija će se aktivirati (obratite pažnju da su INTR i INTA linije aktivne na **NISKOM LOGIČKOM NIVOU**)
- Kao erzultat aktiviranja INTR linije procesor će aktivirati INTA liniju
- Međutim, INTA signal stići će samo do Periferije 1, jer aktivacija IRQ1 signala blokira propagaciju INTA signala dalje od Periferije 1 (vidi sliku dole) pa on ne može stići do Periferije 2

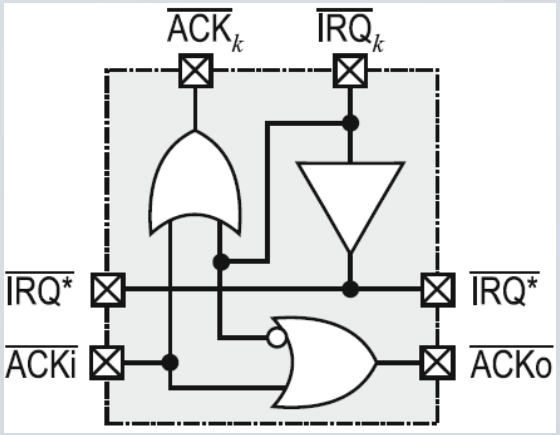




“Daisy Chain” arbitracija III

- Stoga će samo Periferija 1 proslediti svoj ID kod procesoru, koristeći sistemske magistrale
- Na osnovu prosleđenog ID koda, procesor će započeti obradu zahteva za prekidom generisanog od strane Periferije 1, izvršavajući njoj asocirani prekidni podprogram
- Nakon završetka obrade zahteva za prekidom Periferije 1, ona deaktivira svoju IRQ1 liniju, ali INTR signal je još uvek aktivan jer Periferija 2 još uvek generiše svoj zahtev za prekidom (IRQ2 linija je još uvek aktivna)
- Obzirom da je INTR port aktivan, procesor će ponovo aktivirati INTA signal koji će ovog puta stići do Periferije 2, koja će zatim poslati svoj ID kod ka procesoru, na taj način započinjući fazu obrade prekida Periferije 2



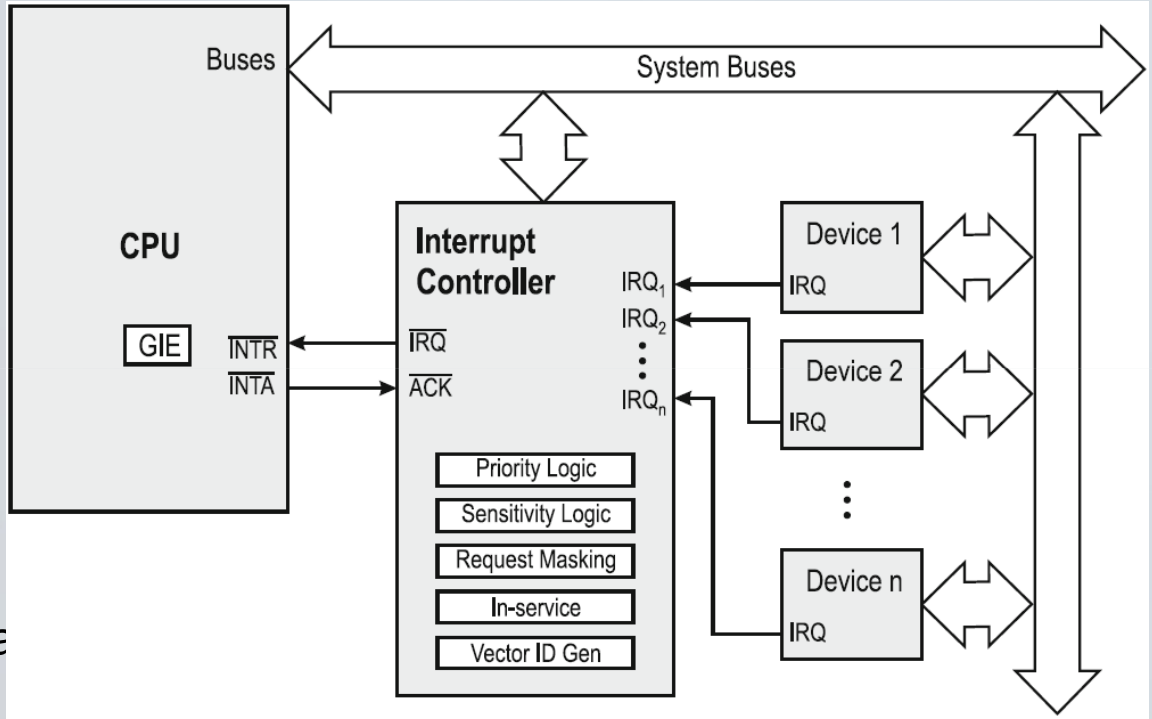


“Daisy Chain” arbitracija IV

- “Daisy Chain” mehanizam je vrlo jednostavan za implementaciju ali ima nekoliko nedostataka:
  - Prioriteti periferijskih jedinica su “ožičeni” (hardwired) i ne mogu se menjati tokom rada sistema
  - Periferije koje se nalaze “niže duž lanca” imaju duže vreme obrade prekida, jer je potrebno više vremena da se INTA signal propagira do njih
  - “Daisy Chain” mehanizam može da radi samo sa prekidima koji se koriste nivo signala da iniciraju zahtev. U slučaju zahteva za prekidom koji koriste ivicu signala, dodatni hardver je neophodan za memorisanje svih pristiglih zahteva u sistemu kako ne bi došlo do gubitka nekog od njih

Arbitracija pomoću kontrolera prekida I

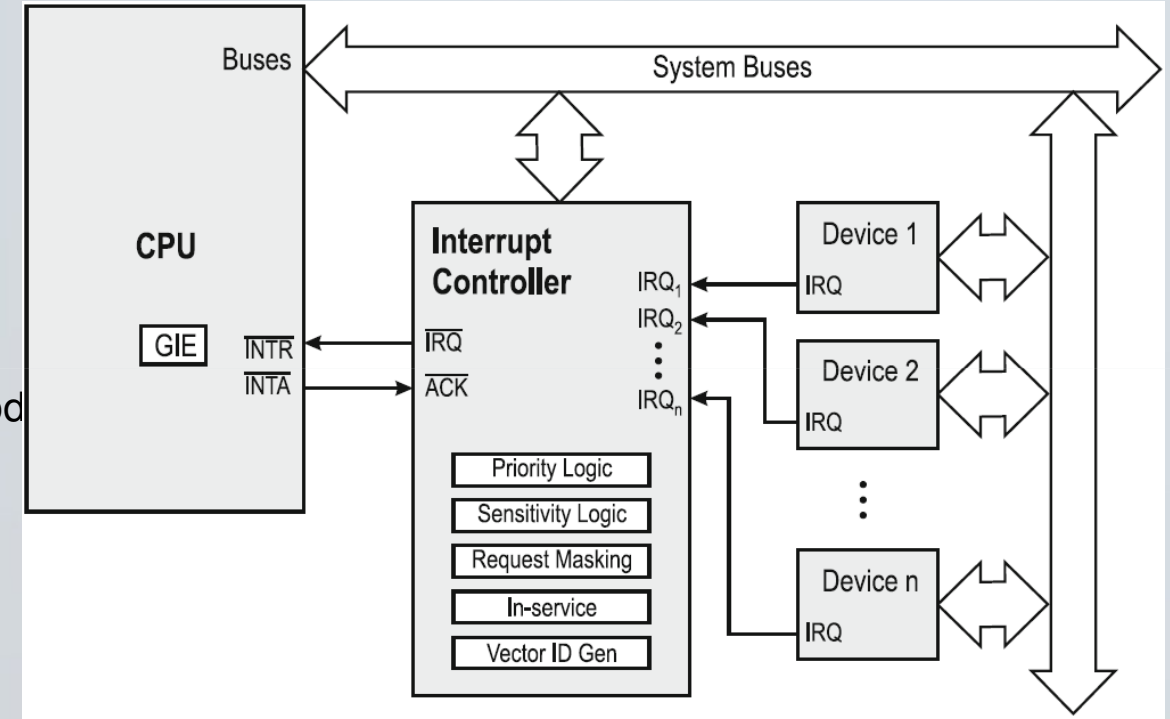
- Arbitracija pomoću kontrolera prekida zahteva postojanje posebnog hardverskog modula (kontrolera prekida) koji će prihvatati sve pristigle prekide i izvršiti odgovarajuću arbitraciju
- Kontroler prekida je programabilni arbiter, koji se može programirati od strane procesora tako da je moguće definisati redosled obrade (šemu obrade prekida) pristiglih zahteva za prekidom
- Kontroleri prekida se obavezno koriste u sistemima sa vektorskim prekidima jer pojednostavljuju dizajn periferijskih jedinica
- Kontroler prekida postavlja se između procesora i periferijskih jedinica koje mogu generisati zahteve za prekidom
- Pojedinačni zahtevi za prekidom se prihvataju u prekidnom kontroleru, koji ih zatim prosleđuje ka procesoru na osnovu definisane šeme prioriteta



Arbitracija pomoću kontrolera prekida II

- Kontroler prekida, pored obrade pristiglih zahteva za prekidom, obezbeđuje i niz drugih funkcija koje bi inače zahtevale dodatne hardverske i softverske resurse za njihovu implementaciju.

- Neke od dodatnih funkcija koje obezbeđuje kontroler prekida su:
  - **Slanje ID koda tokom INTA ciklusa** – nakon što dobije INTA zahtev od procesora kontroler, na osnovu konfigurisanog režima prioriteta, bira periferiju koju je potrebno opslužiti i šalje njen ID kod procesoru
  - **Mogućnost konfigurisanja načina signalizacije zahteva za prekidom** – za svaku periferijsku jedinicu moguće je izabrati način signalizacije: nivoom, ivicom (kao i kojom ivicom ili sa obe)
  - **Mogućnost definisanje šeme prioriteta** – ovo pruža mogućnost izmene šeme prioriteta u toku rada sistema kao odgovor na novonastale okolnosti
  - **Mogućnost maskiranja individualnih zahteva za prekidom** – na ovaj način moguće je zabraniti zahteve za prekidom pojedinih periferijskih jedinica



Arbitracija pomoću kontrolera prekida III

- Funkcije koje će obezbeđivati kontroler prekida zavise od sveukupne arhitekture sistema u kojem će se on koristiti
- U slučaju mikropocesora, programabilni kontroleri prekida se projektuju tako da odgovaraju arhitekturi i komunikacionim protokolima koje koristi ciljni mikroprocesor
- Rani sistemi posedovali su posebna integrisana kola, poput kontrolera prekida 8259A kompanije Intel koji je bio projektovan za korišćenje sa ranim familijama x86 kompatibilnih procesora
- Savremeni mikroprocesori po pravilu poseduju integrisane kontrolere prekida koji se nalaze na istom čipu sa procesorskim jezgrom ili u nekom od sistemskih kontrolera

Arbitracija pomoću kontrolera prekida IV

- Funkcije koje će obezbeđivati kontroler prekida zavise od sveukupne arhitekture sistema u kojem će se on koristiti
- U slučaju mikropocesora, programabilni kontroleri prekida se projektuju tako da odgovaraju arhitekturi i komunikacionim protokolima koje koristi ciljni mikroprocesor
- Rani sistemi posedovali su posebna integrisana kola, poput kontrolera prekida 8259A kompanije Intel koji je bio projektovan za korišćenje sa ranim familijama x86 kompatibilnih procesora
- Savremeni mikroprocesori po pravilu poseduju integrisane kontrolere prekida koji se nalaze na istom čipu sa procesorskim jezgrom ili u nekom od sistemskih kontrolera

Arbitracija pomoću kontrolera prekida V

- U slučaju mikrokontrolera, obrada zahteva za prekidom obično je poverena posebnom modulu koji se gotovo uvek nalazi na istom integrisanom kolu sa procesorom i ostalim modulima koji čine odgovarajući mikrokontroler

- Zbog toga je jedan od prvih koraka prilikom započinjanja razvoja embeded sistema koji je baziran na odabranom mikrokontroleru, analiza mogućnosti i načina programiranja pridruženog kontrolera prekida

- Prvi korak je da se konsultuje dokumentacija proizvođača mikrokontrolera

Sistem prekida u mikrokontroleru Intel 8051

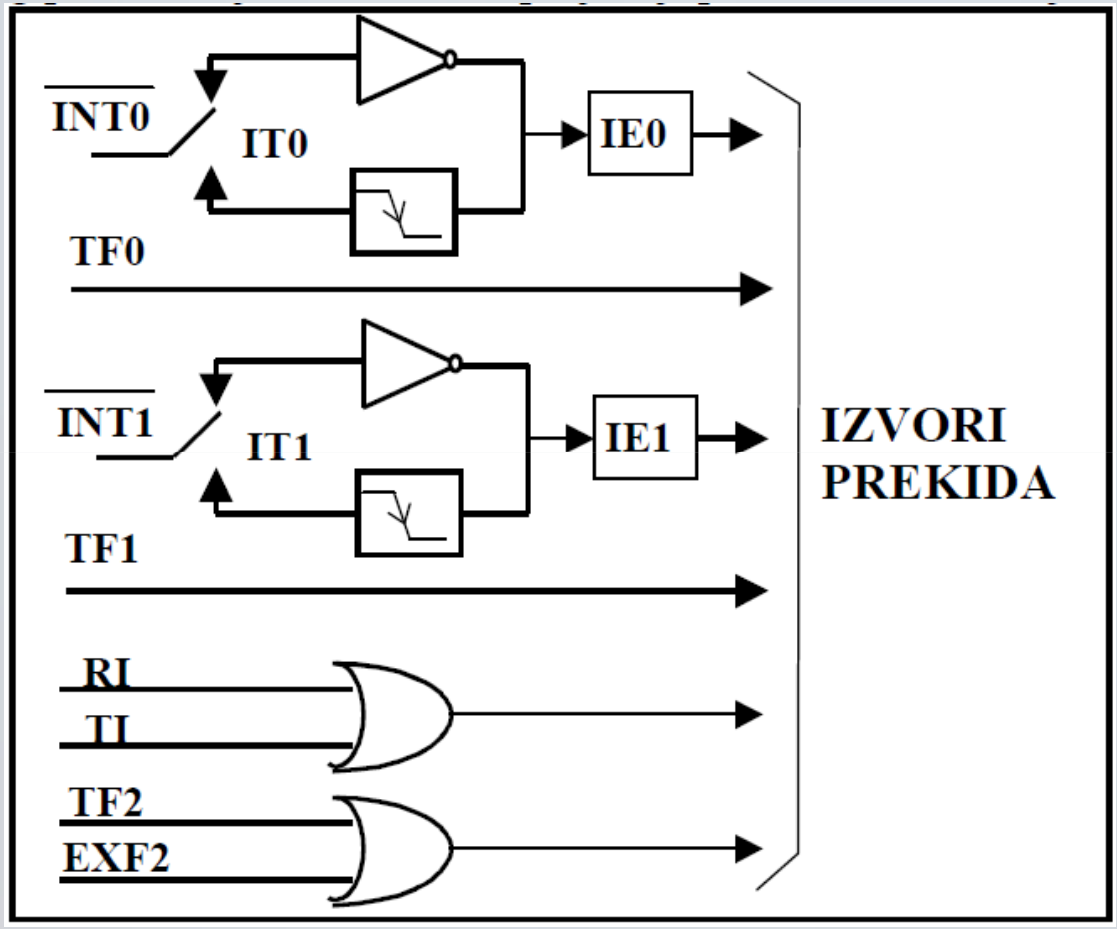
Sistem prekida u mikrokontroleru Intel 8051 I

- Mikrokontroler 8051 poseduje integrisani kontroler prekida pomoću kojega je moguće dozvoliti/zabraniti odgovarajuće prekide kao i uvesti sistem prioriteta

- Unutar mikrokontrolera 8051 postoji ukupno pet izvora prekida:
  - Dva spoljašnja prekida (**INT0** i **INT1**)
  - Prekidi tajmera 0 i 1 (**TF0** i **TF1**)
  - Prekid serijskog porta (**RI** i **TI** objedinjeni u jedan zahtev za prekidom)

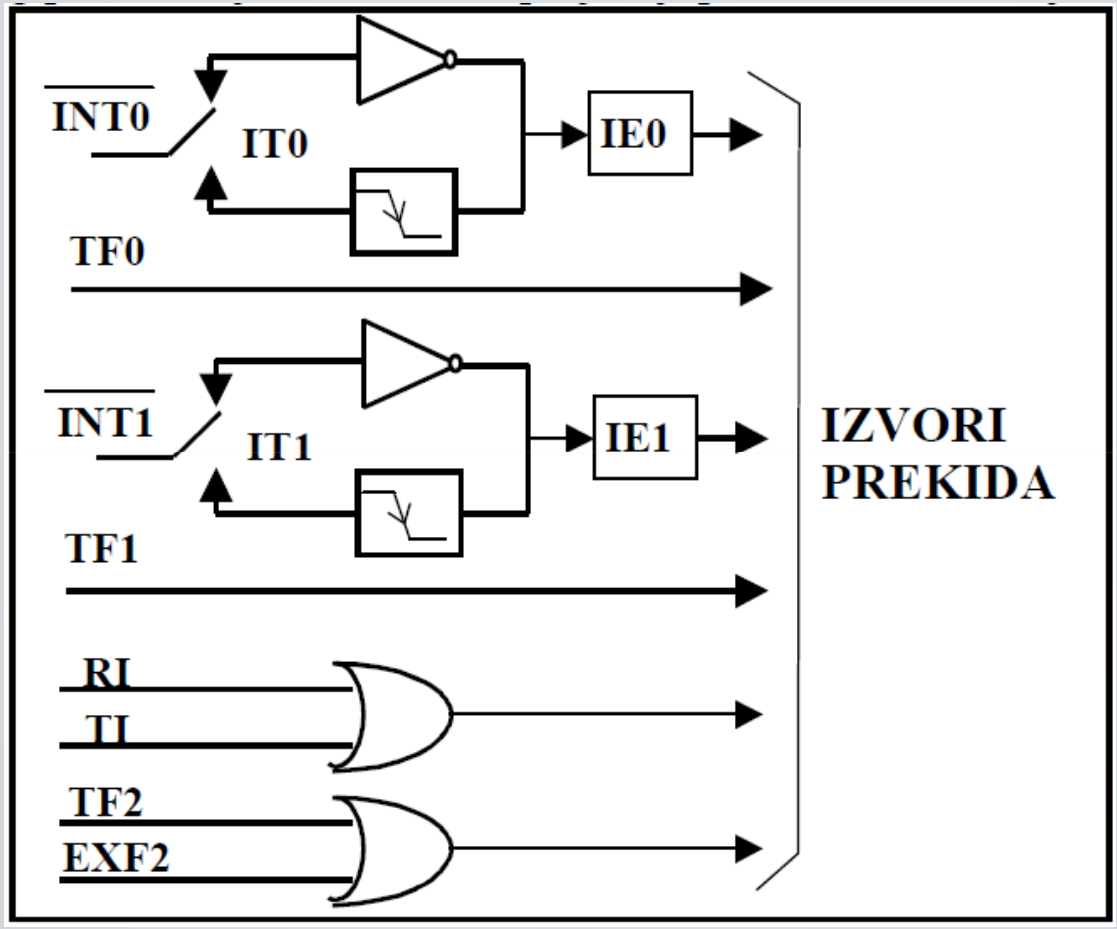
- Mikrokontroler poseduje dva ulaza – **INT0** i **INT1** (deo su porta 3) preko kojih je moguće dovesti signale za zahtev za prekidom sa bilo kog uređaja iz okruženja mikrokontrolera

- Dva bita, **IT0 i IT1** registra **TCON**, određuju hoće li ulazi INT0 i INT1 biti osetljivi na nizak nivo ili opadajuću ivicu signala zahteva za prekidom.



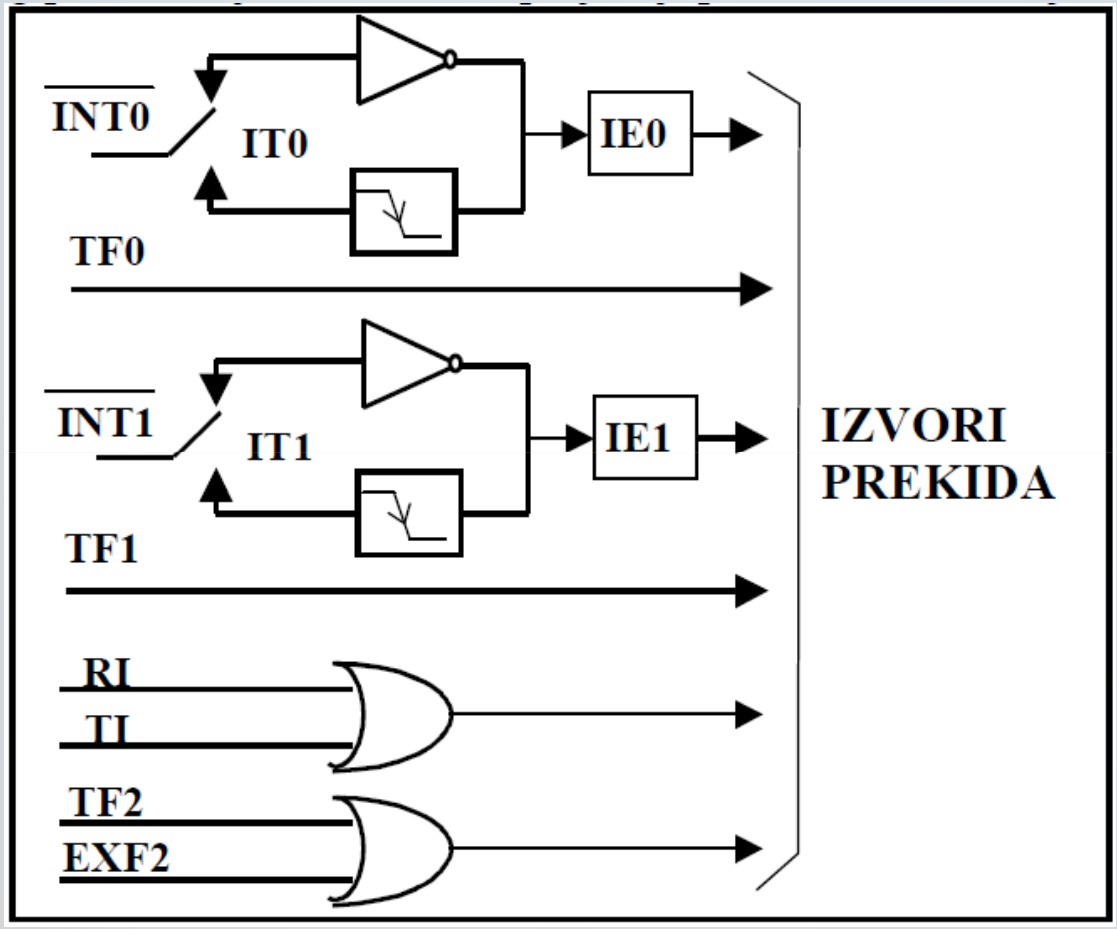
Sistem prekida u mikrokontroleru Intel 8051 II

- Ako je IT0=0 pojava niskog nivoa na INT0 ulazu će se smatrati zahtevom za prekidom
- Ako je IT0=1 samo opadajuća ivica signala na INT0 izaziva prekid, a sledeći zahtev može se desiti se tek kada se INT0 vrati na visok nivo i ponovo padne na nizak
- U oba slučaja važeći zahtev za prekidom postavlja odgovarajući bit IE registra TCON na jedinicu (zahtev na INT0 bit IE0, a na INT1 bit IE1)
- Tek postavljanje ovih bita izaziva prelazak na odgovarajući potprogram za opsluživanje prekida
- Bitovi IE0 i IE1 se automatski vraćaju u stanje 0 kada mikrokontroler prene na potprogram za opsluživanje odgovarajućeg prekida.



Sistem prekida u mikrokontroleru Intel 8051 III

- Preticanje tajmera 0 ili 1, u bilo kom modu, može predstavljati zahtev za prekidom
- Preticanje tajmera 0 ili 1 postavlja bite **TF0** ili **TF1** registra **TCON**, označavajući da je došlo do preticanja
- Ako je odgovarajući prekid dozvoljen, tj. nije maskiran, prelazi se na potprogram za opsluživanje prekida, a time se automatski bit **TF0** ili **TF1**, u zavisnosti od opsluženog prekida, vraća na vrednost 0
- Ukoliko je dozvoljen prekid serijskog interfejsa, zahtev za prekidom će se desiti svaki put kada je završeno slanje reči preko serijske veze (**TI** bit je aktivan), kao i svaki put kada je primljena cela reč (**RI** bit je aktivan)



Maskiranje prekida

- Svaki od pomenutih prekida se može dozvoliti ili zabraniti tj. maskirati
- Kada je prekid maskiran, odgovarajući biti (**TF0, TF1, IE0 i IE1)** se postavljaju na 1, ali to ne izaziva prelazak na odgovarajući servisni potprogram
- Kod 8051 svaki od izvora prekida se može maskirati, što se kontroliše preko registra **IE** (Interrupt Enable, vidi sliku desno)
- Postavljanjem odrenenog bita na 1 odgovarajući prekid biva dozvoljen, u suprotnom je maskiran
- Najviši bit **EA** (Enable All) kontroliše maskiranje svih prekida
- Ukoliko je on 0, svi prekidi su maskirani bez obzira na stanje bitova istog registra koji kontrolišu maskiranost pojedinačnih izvora prekida.

(MSB)							(LSB)
$\overline{EA}$	X	(ET2)	ES	ET1	EX1	ET0	EX0

$\overline{EA}$	ako je ovaj bit 0, svi prekidi su maskirani. Ako je 1, svaki prekid se pojedinačno maskira pomoću ostalih bita ovog registra	ET1	ako je ovaj bit jednak jedinici, prekid zbog preticanja tajmera 1 je dozvoljen, u suprotnom je maskiran
ET2	ako je ovaj bit jednak jedinici, prekid zbog preticanja tajmera 2 je dozvoljen, u suprotnom je maskiran. Bit postoji samo u 8052.	EX1	ako je ovaj bit jedinica, spoljašnji prekid 1 je dozvoljen, u suprotnom je maskiran
		ET0	ako je ovaj bit jednak jedinici, prekid zbog preticanja tajmera 0 je dozvoljen, u suprotnom je maskiran
ES	ako je ovaj bit jednak jedinici, prekid serijskog porta je dozvoljen, inače je maskiran	EX0	ako je ovaj bit jedinica, spoljašnji prekid 0 je dozvoljen, u suprotnom je maskiran

Prioriteti prekida u mikrokontroleru 8051

- Kod 8051 svaki prekid može da ima dva nivoa prioriteta
- Ako se tokom opsluživanja nekog prekida desi novi prekid, opsluživanje prethodnog će se prekinuti samo ako je novi prekid višeg prioriteta
- Prioritet prekida određuje stanje registra **IP**
- Ako je neki bit 0, njemu pripadajući prekid je nižeg prioriteta, a u suprotnom višeg
- Na taj način se svi prekidi dele u dve grupe po prioritetu
- U slučaju da dva (ili više) prekida nastupe istovremeno, prvo se opslužuje prekid sa višim, a zatim sa nižim prioritetom
- Ako su oba prekida istog prioriteta, tada je redosled prioriteta fiksan i sledeći: spoljašnji prekid 0, tajmer 0, spoljašnji prekid 1, tajmer 1 i na kraju prekid serijskog interfejsa.

(MSB)							(LSB)
X	X	(PT2)	PS	PT1	PX1	PT0	PX0

PT2	postavljanjem ovog bita na jedinicu, prekid tajmera 2 se programira kao prekid višeg prioriteta (samo u 8052)	**PX1**	postavljanjem ovog bita na jedinicu, spoljašnji prekid 1 se programira kao prekid višeg prioriteta
PS	postavljanjem ovog bita na jedinicu, prekid serijskog porta se programira kao prekid višeg prioriteta	**PT0**	postavljanjem ovog bita na jedinicu, prekid tajmera 0 se programira kao prekid višeg prioriteta
PT1	postavljanjem ovog bita na jedinicu, prekid tajmera 1 se programira kao prekid višeg prioriteta	**PX0**	postavljanjem ovog bita na jedinicu, spoljašnji prekid 0 se programira kao prekid višeg prioriteta

Obrada prekida u mikrokontroleru 8051

- Mikrokontroler 8051 koristi sistem sa auto-vektorskim prekidima i fiksnim početnim adresama prekidnih podprograma

- Procedura obrade prekida unutar mikrokontrolera 8051 je sledeća:
  - Nakon izvršenja svake pojedine instrukcije proverava se stanje svih bita koji označavaju validan zahtev za prekidom (**IE0**, **TF0**, **IE1**, **TF1**, **TI** i **RI**). Svaki od ovih bita može biti postavljen automatski, validnim zahtevom za prekidom (uobičajena situacija), ili softverski, upisivanjem u odgovarajući registar (softversko izazivanje prekida) – efekat je isti.

  - Ako se utvrdi da postoji zahtev za prekidom, a pri tome:
    - Taj prekid nije maskiran.
    - Nije u toku opsluživanje prekida istog ili višeg prioriteta.
    - Instrukcija koja je sledeća po redu nije **RETI** ili instrukcija upisa u registre IE ili IP.

  - Na stek se stavlja sadržaj programskog brojača (PC – 16 bita), a sam programski brojač se puni adresom početka servisnog potprograma tog konkretnog prekida (registar stanja, **PSW**, se **NE ČUVA** na steku)

vrsta prekida	početna adresa servisnog potprograma
spoljašnji prekid 0	0003H
prekid tajmera 0	000BH
spoljašnji prekid 1	0013H
prekid tajmera 1	001BH
prekid serijskog porta	0023H
prekid tajmera 2	002BH

Dizajn softvera za rad sa prekidima

Osnove razvoja softvera za rad sa prekidima

- Da bi se prekidi mogli uspešno koristiti u nekoj aplikaciji potrebno je rešiti sledeća četiri problema:
  - Izvršiti alokaciju i organizaciju steka
  - Pripremiti sistemsku vektor tabelu
  - Razviti prekidne podprograme za svaki od mogućih izvora prekida
  - Dozvoliti pojavu odgovarajućih zahteva za prekidom

Alokacija steka

- Pravilna alokacija steka je od fundamentalnog značaja sa pravilan rad sistema baziranog na prekidima
- Stek se koristi za smeštanje tekućih vrednosti programskog brojača (PC), statusnog registra (SR) i niza drugih registara nakon prihvatanja zahteva za prekidom
- U slučaju asemblera, alokacija steka mora se eksplicitno izvršiti prilikom pisanja programa, u slučaju C-a ovaj posao će odraditi kompajler
- Prilikom alokacije steka mora se pravilno proceniti njegova potrebna veličina, koja će biti dovoljna da opsluži potrebe prekidnih podprograma, funkcijskih poziva, privremenog smeštaja podataka i prosleđivanja parametara
- U slučaju rekurzivnih i ugnježđenih funkcijskih poziva dubina rekurzije mora se strogo kontrolisati kako ne bi došlo do prekoračenja opsega steka (stack overflow)

Priprema sistemske vektor tabele

- Procesor koristi sadržaj sistemske vektor tabele prilikom određivanja početne adrese prekidnog podprograma koji je potrebno izvršiti prilikom obrade zahteva za prekidom

- U slučaju da se sadržaj ove tabele može definisati i modifikovati od strane programera, prilikom inicijalizacije sistema potrebno je izvršiti inicijalizaciju sistemske vektor tabele kako bi se svakom aktivnom izvoru prekida u sistemu pridružio odgovarajući prekidni podprogram koji će ga servisirati

Razvoj prekidnih podprograma I

- Prekidni podprogrami, kao i svaki drugi vid programa moraju da poštuju pravila dobre programerske prakse

- Međutim, zbog toga što prekidni podprogrami mogu biti “pozvani” u bilo kom trenutku prilikom izvršavanja glavnog programa, moraju se poštovati i neka dodatna pravila:
  - **Prekidni podprogrami bi trebali da budu što brži i što kraći**

    U većini slučajeva, dok se izvršava jedan prekidni podprogram nije moguće primati nove zahteve za prekidom.

    Ukoliko se u sistemu pojavi neki događaj koji zahteva hitnu akciju, u slučaju izvršavanja dugačkog prekidnog podprograma to neće biti moguće što može rezultovati u neželjenim posledicama po rad čitavog sistema ili će usporiti vreme reakcije sistema na neke eksterne događaje.

    Da bi se ovo postiglo, u prekidnim podprogramima treba izbegavati dugačke postupke izračunavanja, petlje i pozive funkcija. Ukoliko su ovakve stvari neophodne da bi se obradio zahtev za prekidom treba ih realizovati unutar glavnog programa,

Razvoj prekidnih podprograma II

- **Prekidni podprogram ne sme da izmeni sadržaj unutrašnjih registara procesora**

  Nakon što se završi izvršavanje prekidnog podprograma, sadržaj svih unutrašnjih registara procesora mora biti jednak sadržaju koji je bio zatečen prilikom primanja zahteva za prekidom

  Ovo pravilo je od posebnog značaja ukoliko se softver razvija korišćenjem asemblerskog jezika. U slučaju viših programskih jezika kompajler vodi računa o ovome.

  Da bi se ispunio ovaj zahtev, na početku svakog prekidnog podprograma sadržaj svakog unutrašnjeg registra procesora mora se smestiti na stek

  Neposredno pre povratka iz svakog prekidnog podprograma sadržaj svakog unutrašnjeg registra mora se postaviti na staru vrednost, koristeći vrednosti su smeštene na steku

Razvoj prekidnih podprograma III

- **Zabranjeno je prosleđivanje parametara i vraćanje povratnih vrednosti**

  Obzirom da je poziv prekidnog podprograma nepredvidiv, ne postoji siguran način da se prekidnom podprogramu proslede parametri ili da on vrati povratnu vrednost preko nekog od unutrašnjih registara ili steka

  Ukoliko je ovo neophodno, moraju se koristiti globalne promenljive

- **Prekidni podprogram mora se završiti korišćenjem posebne instrukcije za povratak iz prekidnog podprograma (RETI)**

  Iako ovaj zahtev zvuči logično, česta greška je da se za povratak iz prekidnog podprograma koristi instrukcija za povratak iz običnog podprograma (RET) umesto specijalizovane instrukcije RETI

  Ova greška neće biti detektovana od strane asemblera, jer je RET instrukcija takođe legalna instrukcija, ali će njenog neadekvatno korišćenje izazvati gubitka integriteta steka i unutrašnjih registara procesora i narušiti očekivano ponašanje sistema

  Kako bi se olakšao razvoj aplikacije i otklanjanje grešaka, poželjno je da svaki prekidni podprogram ima samo jednu izlaznu tačku

Dozvola pojave zahteva za prekidom

- Da bi se prekidi mogli obrađivati, potrebno ih je dozvoliti

- Ovo se obezbeđuje postavljanjem odgovarajućih vrednosti bitovima za dozvolu prekida na različitim nivoima unutar embeded sistema (unutar procesora, unutar kontrolera prekida, unutar periferijskih jedinica)

- Minimalno, potrebno je dozvoliti prekide barem na dva mesta:
  - Unutar procesora, postavljanjem bita za globalnu dozvolu prekida
  - Unutar svake od periferijskih jedinica, postavljanjem odgovarajućeg bita koji će omogućiti generisanje zahteva za prekidom


entity test_shift is
generic (width : integer := 7
shift_reg <= unsigned (inp);
elsif (en = '1') then