JOVAN ĐORĐEVIĆ

ARHITEKTURA I ORGANIZACIJA RAČUNARA

ARHITEKTURA RAČUNARA

BEOGRAD, 2012.

DJM

PREDGOVOR

Ova knjiga je napisana kao osnovni udžbenik iz arhitekture i organizacije računara i pokriva osnovne koncepte iz arhitekture i organizacije procesora, memorije, ulaza/izlaza i magistrale.

Sistemi.

Autor

Beograd

novembra 2011.

SADRŽAJ

1 ARHITEKTURA RAČUNARA

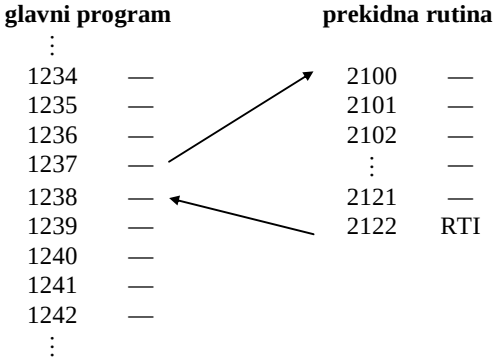
U ovoj glavi se razmatra arhitektura i organizacija mehanizma prekida.

1.1 ARHITEKTURA MEHANIZMA PREKIDA (KOMPLETAN)

Mehanizam prekida omogućuje da se, po generisanju zahteva za prekid od strane nekog modula računara van procesora ili samog procesora, prekine izvršavanje tekućeg programa, koji će se nazivati glavni program, i skoči na novi program, koji će se nazivati prekidna rutina. Poslednja instrukcija u prekidnoj rutini mora da bude instrukcija RTI. To je posebna instrukcija koja omogućuje povratak u glavni program i produžavanje izvršavanja glavnog programa sa onog mesta gde je bilo prekinuto.

Kada se javi zahtev za prekid, tekuća instrukcija se nalazi u nekoj od svojih faza izvršavanja. Mehanizam prekida se tako realizuje da se najpre završe faze *čitanje instrukcije*, *formiranje adrese i čitanje operanda*, i *izvršavanje operacije* tekuće instrukcije, pa da se tek onda pređe na prihvatanje zahteva za prekid tako što se izvršavanje tekuće instrukcije produži prelaskom na fazu *opsluživanje zahteva za prekid* u okviru koje se skače na prvu instrukciju prekidne rutine. Izuzetak od ovoga predstavljaju instrukcije nad nizovima alfanumeričkih znakova kod kojih se zahtev za prekid prihvata u nekom pogodnom trenutku izvršavanja faze *izvršavanje operacije*.

Efekti mehanizma prekida i instrukcije RTI na izvršavanje glavnog programa i prekidne rutine su prikazani na slici 1. Uzeto je da se u toku izvršavanja instrukcije glavnog programa sa adrese 1237 javlja zahtev za prekid. Ova instrukcija se najpre izvrši do kraja, pa se prelazi na izvršavanje instrukcije sa adrese 2100, na kojoj se nalazi prva instrukcija prekidne rutine, umesto instrukcije sa adrese 1238, na kojoj se nalazi prva sledeća instrukcija glavnog programa. Instrukcijom RTI sa adrese 2122 se obezbeđuje da procesor kao sledeću izvršava instrukciju glavnog programa sa adrese 1238. To je instrukcija glavnog programa koja bi se normalno i izvršavala posle instrukcije sa adrese 1237 da se u toku njenog izvršavanja nije javio zahtev za prekid.



Slika 1 Prekid i povratak iz prekidne rutine

Aktivnosti u procesoru kojima se prekida izvršavanje glavnog programa i skače na prekidnu rutinu nazivaju se opsluživanje zahteva za prekid, a aktivnosti kojima se obezbeđuje povratak iz prekidne rutine u glavni program i produžavanje izvršavanja glavnog programa sa mesta i pod uslovima koji su bili pre skoka na prekidnu rutinu nazivaju se povratak iz prekidne rutine.

Zahteve za prekid mogu da generišu:

① kontroleri periferija da bi procesoru signalizirali spremnost za prenos podataka (maskirajući prekidi),

② uređaji računara koji kontrolišu ispravnost napona napajanja, transfera na magistrali, rada memorije itd. (nemaskirajući prekidi),

③ procesor, kao rezultat otkrivene nekorektnosti u izvršavanju tekuće instrukcije (nelegalan kod operacije, nelegalno adresiranje, greška prilikom deljenja, itd.),

④ procesor, ako je prethodno posebnom instrukcijom zadat takav režim rada procesora, kroz postavljanje bita *prekid posle svake instrukcije* u programskoj statusnoj reči PSW na vrednost 1, da se posle svake instrukcije skače na određenu prekidnu rutinu i

⑤ procesor, ako je tekuća instrukcija koja se izvršavanja instrukcija prekida INT.

Prekidi pod ① i ② se nazivaju spoljašnji, a pod ③, ④ i ⑤ unutrašnji.

1.1.1 Opsluživanje zahteva za prekid i povratak iz prekidne rutine

Opsluživanje zahteva za prekid se realizuje delom hardverski i delom softverski, a povratak iz prekidne rutine softverski. Hardverska realizacija dela opsluživanja zahteva za prekid se ostvaruje izvršavanjem koraka faze *opsluživanje zahteva za prekid* na koju se prelazi po završetku izvršavanja koraka faza *čitanje instrukcije*, *formiranje adrese i čitanje operanda*, i *izvršavanje operacije*, jedino ukoliko se tokom izvršavanja ove tri faze javi neki od zahteva za prekid. Softverska realizacija dela opsluživanja zahteva za prekid i povratka iz prekidne rutine se ostvaruju izvršavanjem odgovarajućih instrukcija procesora na početku i kraju prekidne rutine.

1.1.1.1 Opsluživanje zahteva za prekid

Opsluživanje zahteva za prekid se sastoji iz:

- čuvanja konteksta procesora i
- utvrđivanja adrese prekidne rutine

Kontekst procesora čine programski brojač PC, programska statusna reč PSW i preostali programski dostupni registri, kao, na primer, registri podataka, adresni registri, indeksni registri, bazni registri, registri opšte namene itd. Kontekst procesora se čuva najčešće na steku i to:

- programski brojač PC da bi se po povratku iz prekidne rutine u glavni program omogućilo procesoru izvršavanje glavnog programa od instrukcije na kojoj se stalo i
- programska statusna reč PSW i preostali programski dostupni registri da bi se po povratku iz prekidne rutine u glavni program u procesoru obezbedilo isto stanje koje bi bilo da nije bilo prekida i skoka na prekidnu rutinu.

Programski brojač PC i programska statusna reč PSW se čuvaju hardverski. Preostali programski dostupni registri se čuvaju hardverski kod onih procesora kod kojih broj ovih registara nije veliki i softverski sa nekoliko instrukcija na početku prekidne rutine kod onih procesora kod kojih je broj ovih registara veliki. Kod onih procesora kod kojih broj ovih registara nije veliki postoji velika verovatnoća da će se sadržaji najvećeg broja registara menjati u prekidnoj rutini. Zato je opravdano da se ovi registri čuvaju hardverski, a ne da se prekidna rutina opterećuje instrukcijama za čuvanje registara. Međutim, kod onih procesora kod kojih je broj ovih registara veliki postoji velika verovatnoća da će se sadržaji samo manjeg broja registara menjati u prekidnoj rutini. Zato nije opravdano da se troši procesorsko vreme na hardversko čuvanje velikog broja registara čije se vrednosti ne menjaju u prekidnoj rutini, već je bolje da se to učini sa nekoliko instrukcija na početku prekidne rutine. Treba

imati na umu da ukoliko se registri čuvaju hardverski, svi registri se čuvaju. Međutim, ukoliko se registri čuvaju softverski, onda se čuvaju samo oni registri čije se vrednosti menjaju u prekidnoj rutini.

Utvrđivanje adrese prekidne rutine se realizuje hardverski. Utvrđivanje adrese prekidne rutine se realizuje na osnovu sadržaja tabele adresa prekidnih rutina (IV tabela) i broja ulaza u IV tabelu. Stoga se u memoriji, počev od adrese na koju ukazuje sadržaj registra procesora IVTP (*Interrupt Vector Table Pointer*), nalazi IV tabela sa adresama prekidnih rutina za sve vrste prekida. Brojevi ulaza u IV tabelu se dobijaju na više načina i to:

- procesoru ih šalju kontroleri periferija za prekide iz tačke ①, ako ulazi u IV tabelu za maskirajuće prekide nisu fiksni, što je određeno vrednošću 1 bita *promenljivi/fiksni brojevi ulaza* u programskoj statusnoj reči procesora PSW,
- procesor generiše fiksne vrednosti za prekide iz tačke ①, ako su ulazi u IV tabelu za maskirajuće prekide fiksni, što je određeno vrednošću 0 bita *promenljivi/fiksni brojevi ulaza* u programskoj statusnoj reči procesora PSW,
- procesor generiše fiksne vrednosti za prekide iz tačaka ②, ③ i ④ i
- procesor generiše vrednosti na osnovu adresnog dela instrukcije INT za prekid iz tačke ⑤.

Memorijska adresa na kojoj se nalazi adresa prekidne rutine dobija se sabiranjem broja ulaza u IV tabelu pretvorenog u pomeraj sa sadržajem registra IVTP. Sa ovako dobijene memorijske adrese se čita adresa prekidne rutine i upisuje u registar PC.

U okviru opsluživanja zahteva za prekid hardverski se još:

- briše zahtev za prekid u čiju prekidnu rutinu se skače,
- brišu biti *maskiranje svih maskirajućih prekida* i *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW kod prekida svih vrsta i
- upisuje u bite *tekući nivo prioriteta* u programskoj statusnoj reči procesora PSW nivo prioriteta prekidne rutine na koju se skače u slučaju maskirajućeg prekida.

Zahtev za prekid u čiju prekidnu rutinu se skače se briše, jer je prelaskom na prekidnu rutinu dati zahtev za prekid prihvaćen.

Brisanjem bita *maskiranje svih maskirajućih prekida* u programskoj statusnoj reči procesora PSW se obezbeđuje da procesor po ulasku u prekidnu rutinu ne reaguje na maskirajuće prekide, a brisanjem bita *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW se obezbeđuje da procesor po ulasku u prekidnu rutinu ne izvršava prekidnu rutinu u režimu prekid posle svake instrukcije. Time se omogućava obavljanje određenih aktivnosti na početku svake prekidne rutine. Posle toga moguće je u samoj prekidnoj rutini posebnim instrukcijama postaviti bit *maskiranje svih maskirajućih prekida* u programskoj statusnoj reči procesora PSW i time dozvoliti maskirajuće prekide i postaviti bit *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW i time zadati režim rada procesora prekid posle svake instrukcije.

Upisivanjem u bite *tekući nivo prioriteta* u statusnoj reči procesora PSW nivoa prioriteta prekidne rutine na koju se skače u slučaju maskirajućeg prekida obezbeđuje se da se u slučaju maskirajućih zahteva za prekid pristiglih u toku izvršavanja prekidne rutine prihvate samo oni koji su višeg nivoa prioriteta od nivoa prioriteta prekidne rutine.

Na osnovu prethodnih objašnjenja mogu se sumirati aktivnosti u procesoru tokom izvršavanja koraka faza *čitanje instrukcije*, *formiranje adrese i čitanje operanda*, *izvršavanje operacije* i *opsluživanje zahteva za prekid* instrukcije koje su vezane za prihvatanje zahteva za prekid. Zahtev za prekid koji se javi tokom izvršavanja faza *čitanje instrukcije*, *formiranje*

adrese i čitanje operanda, i *izvršavanje operacije* se u procesoru samo pamti, a na njega procesor reaguje tek po završetku ovih faza i prelasku na fazu *opsluživanje zahteva za prekid*.

Faza *opsluživanje zahteva za prekid* počinje na završetku izvršavanja koraka faza *čitanje instrukcije, formiranje adrese i čitanje operanda*, i *izvršavanje operacije*, ispitivanjem da li se tokom izvršavanja ove tri faze javio neki od zahteva za prekid. U slučaju da se nije javio ni jedan od zahteva za prekid, izvršavanje faze *opsluživanje zahteva za prekid* se završava i prelazi na prvi korak faze *čitanje instrukcije*. S obzirom da u fazi *opsluživanje zahteva za prekid* vrednost programskog brojača PC nije menjana, prelazi se na prvi korak faze *čitanje instrukcije* prve sledeće instrukcije u programu. U slučaju da se javio neki od zahteva za prekid, izvršavanje tekuće instrukcije se produžava izvršavanjem koraka faze *opsluživanje zahteva za prekid* u okviru kojih se:

- stavljaju na stek programski brojač PC i programska statusna reč PSW, a ukoliko se radi o procesorima kod kojih se hardverski čuvaju preostali programski dostupni registri, i preostali programski dostupni registri,
- upisuje u programski brojač PC početna adresa prekidne rutine,
- briše zahtev za prekid u čiju prekidnu rutinu se skače,
- brišu biti *maskiranje svih maskirajućih prekida* i *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW kod prekida svih vrsta,
- upisuje u bite *tekući nivo prioriteta* u programskoj statusnoj reči procesora PSW nivo prioriteta prekidne rutine na koju se skače samo u slučaju maskirajućih prekida i
- prelazi na prvi korak faze *čitanje instrukcije* prve instrukcije prekidne rutine.

S obzirom da je u fazi *opsluživanje zahteva za prekid* vrednost programskog brojača PC menjana, produžava se sa prvom instrukcijom prekidne rutine. Na početku prekidne rutine se samo oni preostali programski dostupni registri čije se vrednosti menjaju u prekidnoj rutini posebnim instrukcijama stavljaju na stek, ukoliko se radi o procesorima kod kojih se softverski čuvaju preostali programski dostupni registri.

1.1.1.2 Povratak iz prekidne rutine

Povratak iz prekidne rutine se realizuje tako što se, najpre, posebnim instrukcijama pri kraju prekidne rutine restauriraju vrednostima sa steka sadržaji onih programski dostupnih registara čije su vrednosti posebnim instrukcijama sačuvane na steku na početku prekidne rutine, ukoliko se radi o procesorima kod kojih se softverski čuvaju programski dostupni registri, a potom izvrši instrukcija RTI. Međutim, povratak iz prekidne rutine se realizuje samo izvršavanjem instrukcije RTI, ukoliko se radi o procesorima kod kojih se hardverski u okviru izvršavanja faze *opsluživanje zahteva za prekid* čuvaju programski dostupni registri.

Instrukcijom RTI se sa steka restauriraju sadržaji programske statusne reči procesora PSW i programskog brojača PC, ukoliko se radi o procesorima sa većim brojem programski dostupnih registara kod kojih se u okviru izvršavanja faze *opsluživanje zahteva za prekid* na steku čuvaju samo programski brojača PC i programska statusna reč PSW. Međutim, ukoliko se radi o procesorima sa manjim brojem programski dostupnih registara, kod kojih se u okviru izvršavanja faze *opsluživanje zahteva za prekid* na steku čuvaju ne samo programski brojač PC i programska statusna reč PSW, već i svi programski dostupni registri, instrukcijom RTI se sa steka restauriraju najpre sadržaji programski dostupnih registara, pa tek potom i sadržaji programske statusne reči procesora PSW i programskog brojača PC.

Po završetku izvršavanja instrukcije RTI, nastavlja se izvršavanje prekinutog glavnog programa od instrukcije koja bi se izvršavala i sa kontekstom procesora koji bi bio da nije bilo skoka na prekidnu rutinu.

1.1.2 Prioriteti prekida

U slučajevima kada se tokom izvršavanja koraka faza *čitanje instrukcije*, *formiranje adrese i čitanje operanda*, i *izvršavanje operacije*, generiše više prekida, prekidi se prihvataju u fazi *opsluživanje zahteva za prekid* i to po redosledu opadajućih prioriteta. Tako, na primer, za ranije pobrojane prekide taj redosled je sledeći: najviši je ⑤, zatim niži redom ③, ②, ① i najniži ④.

Prekidi ⑤ i ③ su unutrašnji prekidi koje procesor može da generiše prilikom izvršavanja koraka faza *čitanje instrukcije*, *formiranje adrese i čitanje operanda*, i *izvršavanje operacije*. Ovi prekidi su međusobno isključivi i tokom prolaska kroz korake ove tri faze instrukcije samo jedan od njih može da bude generisan. Tako se u koracima faze *čitanje instrukcije* proverava da li je pročitana neregularna vrednost koda operacije. Ukoliko jeste, generiše se prekid *greška u kodu operacije* i prelazi na prvi korak faze *opsluživanje zahteva za prekid*. Ukoliko nije, produžava se sa koracima faze *formiranje adrese i čitanje operanda* u kojima se proverava da li je specificirano neregularno adresiranje, kao, na primer, neposredno adresiranje za odredišni operand. Ukoliko jeste, generiše se prekid *greška u adresiranju* i prelazi na prvi korak faze *opsluživanje zahteva za prekid*. Ukoliko nije, produžava se sa koracima faze *izvršavanje operacije* u kojima se proverava da li se u polju za kod operacije pročitane instrukcije nalazi binarna vrednost koja odgovara instrukciji prekida. Ukoliko jeste, generiše se prekid *izvršavanje instrukcije prekida* i prelazi na prvi korak faze *opsluživanje zahteva za prekid*. Ukoliko nije, produžava se sa koracima faze *izvršavanje operacije* neke od ostalih instrukcija.

Tokom izvršavanja faza *čitanje instrukcije*, *formiranje adrese i čitanje operanda*, i *izvršavanje operacije* bilo koje instrukcije mogu da stignu spoljašni zahtevi za prekid ② i ①. Ovi zahtevi za prekid se samo pamte, a na njihovo eventualno prihvatanje prelazi tek po završetku faza *čitanje instrukcije*, *formiranje adrese i čitanje operanda*, i *izvršavanje operacije* tekuće instrukcije i prelasku na prvi korak faze *opsluživanje zahteva za prekid* u kome se proverava da li postoji neki od zahteva za prekid.

Zahtev za prekid ④ se generiše u slučaju posebne instrukcije kojom se u fazi *izvršavanje instrukcije* upisivanjem vrednosti 1 u bit *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW zadaje režim rada *prekid posle svake instrukcije*.

Svi zahtevi za prekid se prihvataju u fazi *opsluživanje zahteva za prekid* i to po redosledu opadajućih prioriteta, koji je najviši za ⑤, niži redom za ③, ②, ① i najniži za ④. Zbog toga se u fazi *opsluživanje zahteva za prekid* prilikom utvrđivanja broja ulaza u tabelu sa adresama prekidnih rutina proverava po opadajućim prioritetima ⑤, ③, ②, ① i ④ koji je zahtev za prekid najvišeg prioriteta prisutan, za njega formira broj ulaza, briše taj zahtev za prekid, osim zahteva za prekida pod ④, i prelazi na korake zajedničke za sve prekide u kojima se sabiranjem broja ulaza pretvorenog u pomeraj i registra čiji sadržaj predstavlja adresu tabele sa adresama prekidnih rutina dobija adresa sa koje se čita adresa prekidne rutine i upisuje u programski brojač PC.

Zahtevi za prekidi pod ① koji dolaze od kontrolera periferija (spoljašnji maskirajući prekidi) se prihvataju ukoliko nema zahteva za prekide pod ⑤, ③ i ②. Ovi zahtevi za prekid dolaze od više kontrolera periferija i to svaki po posebnoj liniji i mogu se javiti istovremeno. Stoga se i oni prihvataju po redosledu opadajućih prioriteta, pri čemu je nivo prioriteta zahteva za prekid određen pozicijama linija po kojima kontroleri periferija šalju zahteve za prekid.

Zahtev za prekid pod ④ se prihvata ukoliko nema ni jednog zahteva za prekid višeg nivoa proriteta.

1.1.3 Selektivno maskiranje maskirajućih prekida

Za maskirajuće prekide postoji u procesoru poseban programski dostupan registar IMR koji se naziva registar maske. Svakoj liniji po kojoj mogu da se šalju zahtevi za prekid od kontrolera periferija pridružen je poseban razred registra maske. Zahtev za prekid koji stiže po određenoj liniji u procesoru će biti prihvaćen jedino ukoliko se u odgovarajućem razredu registra maske nalazi vrednost 1. Posebnom instrukcijom prenosa se u razrede registra maske IMR upisuju vrednosti 1 ili 0. Time se programskim putem selektivno dozvoljava ili zabranjuje opsluživanje pojedinih maskirajućih zahteva za prekid.

1.1.4 Maskiranje svih maskirajućih prekida

Maskirajući zahtevi za prekid, bez obzira na to da li su selektivno maskirani sadržajem registra maske IMR ili ne, mogu se svi maskirati bitom *maskiranje svih maskirajućih prekida* u programskoj statusnoj reči procesora PSW. Posebnim instrukcijama u ovaj razred registra PSW upisuju se vrednosti 1 ili 0. Time se programski putem dozvoljava ili zabranjuje opsluživanje maskirajućih prekida. Pored toga u fazi *opsluživanje zahteva za prekid* svih instrukcija se hardverski u bit *maskiranje svih maskirajućih prekida* u programskoj statusnoj reči procesora PSW upisuje vrednost 0.

1.1.5 Prekid posle svake instrukcije

Postoji mogućnost da se zada takav režim rada procesora da se posle svake izvršene instrukcije skače na određenu prekidnu rutinu. Ovakav režim rada procesora se naziva *prekid posle svake instrukcije*. U njemu se procesor nalazi ukoliko bit *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW ima vrednost 1. Posebnim instrukcijama u ovaj bit programske statusne reči procesora PSW upisuju se vrednosti 1 ili 0. Time se programskim putem dozvoljava ili ne dozvoljava režim rada procesora *prekid posle svake instrukcije*. Pored toga u fazi *opsluživanje zahteva za prekid* svih instrukcija se hardverski u bit *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW upisuje vrednost 0.

1.1.6 Promenljivi/fiksni brojevi ulaza

Postoji mogućnost da se zadaju dva režima rada procesora prilikom utvrđivanja broja ulaza za spoljašnje maskirajuće prekide. U režimu rada *promenljivi ulazi* kontroler periferije šalje broj ulaza, a u režimu rada *fiksni ulazi* broj ulaza generiše fiksno procesor na osnovu pozicije linije po kojoj dolazi zahtev za prekid. Režim rada u kome se procesor nalazi određen je vrednošću bita *promenljivi/fiksni brojevi ulaza* u programskoj statusnoj reči procesora PSW. Posebnim instrukcijama u ovaj bit programske statusne reči procesora PSW upisuju se vrednosti 1 ili 0. Time se programskim putem zadaje jedan od ova dva režima određivanja broja ulaza za spoljašnje maskirajuće zahteve za prekid.

1.1.7 Instrukcija prekida

U skupu instrukcija postoji instrukcija INT kojom se može programskim putem izazvati prekid i skok na željenu prekidnu rutinu. Prekidna rutina na koju treba skočiti određuje se adresnim delom ove instrukcije koji sadrži broj ulaza u tabelu adresa prekidnih rutina. Izvršavanje ove instrukcije realizuje sve ono što je nabrojano u okviru hardverskog dela opsluživanja zahteva za prekid, s tim što je broj ulaza u tabeli adresa prekidnih rutina dat adresnim poljem same instrukcije.

1.1.8 Gneždenje prekida

Kada procesor izvršava prekidnu rutinu može stići novi zahtev za prekid. Na ovaj zahtev za prekid može se reagovati na sledeće načine:

- prekida se izvršavanje tekuće prekidne rutine i skače na novu prekidnu rutinu ili
- ne prekida se izvršavanje prekidne rutine, već se zahtev za prekid prihvata tek po povratku u glavni program.

Procesor reaguje na oba načina u zavisnosti od situacije u kojoj se nalazi. Ta situacija zavisi od čitavog niza elemenata kao što su:

- ima više tipova zahteva za prekid,
- kod ulaska u prekidnu rutinu brišu se biti *maskiranje svih maskirajućih prekida* i *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW kod prekida svih vrsta,
- programskim putem se može, upisivanjem vrednosti 0 ili 1 u bite *maskiranje svih maskirajućih prekida* i *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW, odrediti kada će se reagovati na maskirajuće prekide ili prekidati program posle svake instrukcije a kada ne,
- maskirajući prekidi se mogu selektivno maskirati odgovarajućim razredima registra maske IMR i
- maskirajući prekidi su uređeni po prioritetima.

Kao ilustracija tih situacija može se uzeti pojednostavljen primer da u procesor stižu samo maskirajući zahtevi za prekid koji nisu ni selektivno maskirani registrom maske IMR, ni svi zajedno bitom *maskiranje svih maskirajućih prekida* u programskoj statusnoj reči procesora PSW i da oni imaju prioritete. Pored toga, kada se uđe u prekidnu rutinu po nekom maskirajućem zahtevu za prekid, u procesoru se u bitima *tekući nivo prioriteta* u programskoj statusnoj reči procesora PSW nalazi nivo prioriteta te prekidne rutine. Kada stigne neki novi zahtev za prekid, a procesor se već nalazi u prekidnoj rutini, procesor će:

- prihvatiti novi zahtev za prekid, ako je on višeg prioriteta nego nivo prioriteta tekuće prekidne rutine ili
- ignorisati novi zahtev za prekid, ako je on nižeg ili istog nivoa prioriteta kao i nivo prioriteta tekuće prekidne rutine.

Prekidanje izvršavanja tekuće prekidne rutine i skok na novu prekidnu rutinu naziva se gneždenje prekida.

1.1.9 Prihvatanje zahteva za prekid

Zahtev za prekid može da bude prihvaćen i time skok na prekidnu rutinu realizovan ili u fazi *opsluživanje zahteva za prekid* instrukcije u toku čijeg izvršavanja je zahtev za prekid generisan ili kasnije u fazi *opsluživanje zahteva za prekid* neke od sledećih instrukcija. Kada će određeni zahtev za prekid biti opslužen zavisi od više faktora, kao što su: da li je reč o spoljašnjem ili unutrašnjem prekidu, da li su maskirajući prekidi maskirani, i to ili selektivno ili svi, da li je stigao samo jedan ili više zahteva za prekid, itd. Stoga za svaku vrstu zahteva za prekid određeni uslovi treba da budu ispunjeni da bi se prešlo na njegovo opsluživanje. Prelazak na opsluživanje određenog zahteva za prekid naziva se prihvatanje zahteva za prekid.

U slučaju da u toku izvršavanja neke instrukcije stigne više zahteva za prekid redosled njihovog prihvatanja definisan je međusobnim prioritetima različitih vrsta prekida. Najviši prioritet ima prekid izazvan izvršavanjem instrukcije prekida INT (⑤), pa redom slede unutrašnji procesorski prekidi (③), spoljašnji nemaskirajući prekid (②), spoljašnji maskirajući

prekidi (①) i prekid posle svake instrukcije (④) koji ima najniži prioritet. Detaljnije objašnjenje prihvatanja pojedinih tipova zahteva za prekid je dato u daljem tekstu.

Unutrašnji procesorski kao rezultat izvršavanja instrukcije prekida INT: Ovaj zahtev za prekid se prihvata u fazi *opsluživanje zahteva za prekid* instrukcije INT. Broj ulaza u IV tabelu je dat adresnim delom instrukcije INT i ima takve vrednosti da može da se uđe u bilo koji ulaz tabele sa adresama prekidnih rutina.

Unutrašnji procesorski prekid pri čitanju instrukcije sa nelegalnim kodom operacije. Ovaj zahtev za prekid se prihvata u fazi *opsluživanje zahteva za prekid* instrukcije za koju je otkriven nelegalan kod operacije. Ovaj zahtev za prekid i prethodni zahtev za prekid su međusobno isključivi i ne mogu da se jave istovremeno. Broj ulaza u IV tabelu za ovaj prekid je fiksan.

Unutrašnji procesorski prekidi pri čitanju instrukcije sa nelegalnim adresiranjem. Ovaj zahtev za prekid se prihvata u fazi *opsluživanje zahteva za prekid* instrukcije za koju je otkriveno nelegalno adresiranje. Ovaj zahtev za prekid i prethodna dva zahteva za prekid su međusobno isključivi i ne mogu da se jave istovremeno. Broj ulaza u IV tabelu za ovaj prekid je fiksan.

Spoljašnji nemaskirajući prekid: Ovaj zahtev za prekid se prihvata u fazi *opsluživanje zahteva za prekid* bilo koje instrukcije ukoliko ne postoji neki od prethodna tri zahteva za prekid koji su višeg prioriteta. Broj ulaza u IV tabelu za ovaj prekid je fiksan.

Spoljašnji maskirajući prekidi: Zahteve za maskirajućim prekidima postavljaju kontroleri periferija preko posebnih linija koje su označene rednim brojevima. Najveći prioritet, u slučaju simultanog pristizanja više od jednog zahteva, ima zahtev za prekid koji dolazi u procesor po liniji sa najvećim rednim brojem. Ovaj zahtev za prekid se prihvata u fazi *opsluživanje zahteva za prekid* bilo koje instrukcije ukoliko je ispunjen svaki od sledećih uslova:

- da je odgovarajući bit u registru maske IMR postavljen na 1,
- da je bit *maskiranje svih maskirajućih prekida* u programskoj statusnoj reči procesora PSW postavljen na 1,
- da je nivo prioriteta kontrolera periferije koji je uputio zahtev za prekid veći od nivoa prioriteta tekućeg programa i
- da ne postoje neki od prethodna četiri zahteva za prekid koji su višeg prioriteta.

Ako je bit *promenljivi/fiksni brojevi ulaza* u programskoj statusnoj reči procesora PSW jednak 1 pa ulazi u IV tabelu nisu fiksni, tada procesor dobija broj ulaza u IV tabelu od kontrolera periferije. Procesor o prihvatanju zahteva za prekid obaveštava kontroler periferije aktiviranjem odgovarajuće linije potvrde. Kontroler periferija tada šalje broj ulaza u IV tabelu koji procesor koristi za određivanje adrese prekidne rutine. Ako je bit *promenljivi/fiksni brojevi ulaza* u programskoj statusnoj reči procesora PSW jednak 0, pa su ulazi u IV tabelu fiksni, brojeve ulaza generiše sam procesor. Po čuvanju sadržaja programske statusne reči procesora PSW na steku, procesor u bite *tekući nivo prioriteta* u programskoj statusnoj reči procesora PSW upisuje nivo prioriteta prekidne rutine na koju se skače.

Unutrašnji procesorski prekid posle svake instrukcije: Zahtev za ovaj prekid se javlja posle izvršenja svake pojedine instrukcije procesora pod uslovom da je na 1 postavljen bit *prekid posle svake instrukcije* u programskoj statusnoj reči procesora PSW. Zahtev se prihvata ukoliko ne postoje neki od prethodnih pet zahteva za prekid koji su višeg prioriteta. Broj ulaza u IV tabelu je fiksan.

1.2 ORGANIZACIJA MEHANIZMA PREKIDA (KOMPLETAN)

(Ovo je proširenje materijala Organizacija (ORT2) i pored prekida od periferija postoje i svi ostali prekidi i proširenje procesora za laboratoriju za ORT2 time što postoje i promenljivi brojevi ulaza)

U ovom odeljku se razmatraju realizacija operacione i upravljačke jedinice dela procesora u kome se realizuje mehanizam prekida.

1.2.1 Operaciona jedinica

Blok *intr* sadrži logički ILI element za formiranje signala prekida **prekid**, kombinacione i sekvencijalne mreže za prihvatanje unutrašnjih prekida i spoljašnjih nemaskirajućih i maskirajućih prekida, registar $IMR_{15...0}$ (slika 2), kombinacione prekidačke mreže za formiranje signala spoljašnjih maskirajućih prekida **printr** (slika 3), kombinacione i sekvencijalne mreže za formiranje pomeraja za tabelu sa adresama prekidnih rutina $IVTDSP_{15...0}$ i registar $IVTP_{15...0}$ (slika 4).

Logički ILI element za formiranje signala prekida **prekid** daje vrednost 1 signala prekida **prekid** ukoliko barem jedan od signala prekida ima vrednost 1 (slika 2). To može da bude neki od signala unutrašnjih prekida i to **PRINS** za prekid zbog izvršavanja instrukcije prekida INT, **PRCOD** za prekid usled čitanja instrukcije sa nelegalnim kodom operacije i **PRADR** za prekid pri čitanju instrukcije sa nelegalnim adresiranjem, zatim signal spoljašnjeg prekida **PRINM** zbog generisanja nemaskirajućeg prekida, potom signal spoljašnjeg prekida **printr** zbog prisustva nekog od maskirajućih prekida i na kraju signal koji predstavlja I funkciju signala unutrašnjeg prekida **PSWT** usled prekida zbog zadatog režim rada procesora prekid posle svake instrukcije i komplementa signala operacije povratka iz prekidne rutine RTI. Treba uočiti da time što se uzima I funkciju signala **PSWT** i komplementa signala se obezbeđuje da režim rada prekid posle svake instrukcije nije dozvoljen za instrukciju povratka iz prekidne rutine RTI .

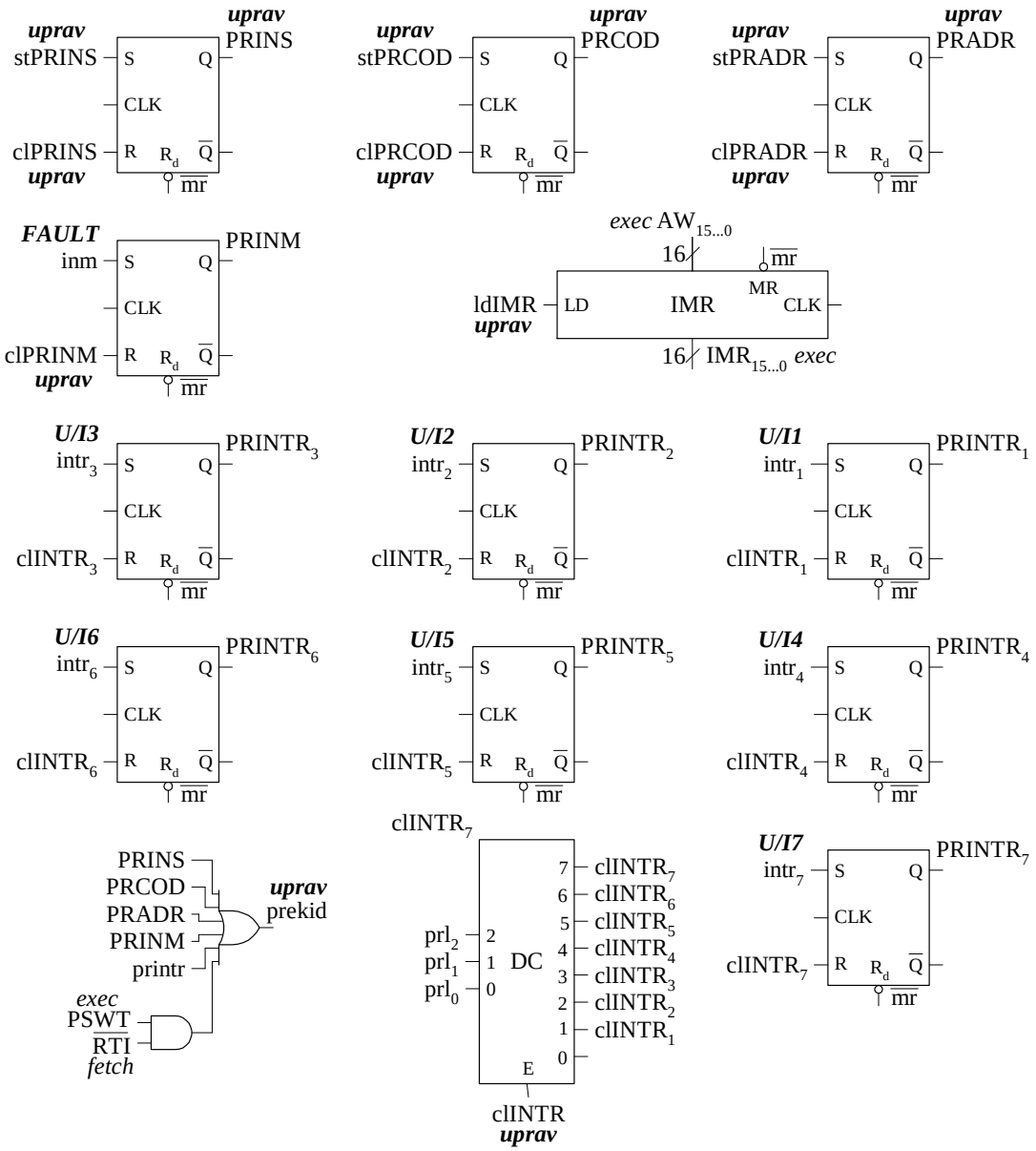
Kombinacione i sekvencijalne mreže za prihvatanje unutrašnjih zahteva za prekid (slika 2) se sastoje od flip-flopova PRINS, PRCOD i PRADR.

Flip-flop PRINS pamti unutrašnji zahtev za prekid izazvan izvršavanjem instrukcije prekida INT. Ovaj flip-flop se postavlja na vrednost 1 vrednošću 1 signala **stPRINS** u fazi *izvršavanje operacije* instrukcije prekida INT i na vrednost 0 vrednošću 1 signala **clPRINS** u fazi *opsluživanje zahteva za prekid* onda kada se ovaj zahtev za prekid prihvata.

Flip-flop PRCOD pamti unutrašnji zahtev za prekid zbog čitanja instrukcije sa nelegalnim kodom operacije. Flip-flop PRCOD se postavlja na vrednost 1 vrednošću 1 signala **stPRCOD** koji se generiše u fazi *čitanje instrukcije* ukoliko je u bloku *fetch* otkriven nepostojeći kod operacije i formirana vrednost 1 signala **grop**. Flip-flop PRCOD se postavlja na vrednost 0 vrednošću 1 signala **clPRCOD** u fazi *opsluživanje zahteva za prekid* onda kada se ovaj zahtev za prekid prihvata.

Flip-flop PRADR pamti unutrašnji zahtev za prekid zbog nelegalnog adresiranja kao na primer korišćenja neposrednog adresiranja za odredišni operand. Flip-flop PRADR se postavlja na vrednost 1 vrednošću 1 signala **stPRADR** koji se generiše u fazi *čitanje instrukcije* ukoliko je u bloku *fetch* otkriveno korišćenja neposrednog adresiranja za odredišni operand i formirana vrednost 1 signala **gradr**. Flip-flop PRADR se postavlja na vrednost 0 vrednošću 1 signala **clPRADR** u fazi *opsluživanje zahteva za prekid* onda kada se ovaj zahtev za prekid prihvata.

Signali **PRINS**, **PRCOD** i **PRADR** se koriste u upravljačkoj jedinici ***uprav*** kao signali logičkih uslova prilikom opsluživanja prekida.



Slika 2 Blok intr (prvi deo)

Kombinacione i sekvencijalne mreže za prihvatanje spoljašnjih nemaskirajućih i maskirajućih prekida se sastoje od flip-flopova PRINM, $PRINTR_1$ do $PRINTR_7$ i dekodera DC (slika 2).

Flip-flop PRINM služi za prihvatanje spoljašnjeg nemaskirajućeg zahteva za prekid **inm**. Zahtev za prekid **inm** dolazi od uređaja ***FAULT*** kao impuls i pamti se u flip-flopu PRINM na prvi signal takta. Flip-flop PRINM se postavlja na vrednost 0 vrednošću 1 signala **clPRINM** u fazi *opsluživanje zahteva za prekid* onda kada se ovaj zahtev za prekid prihvata.

Flip-flopovi $PRINTR_1$ do $PRINTR_7$ služe za prihvatanje spoljašnjih maskirajućih zahteva za prekid $intr_1$ do $intr_7$. Zahtevi za prekid $intr_1$ do $intr_7$ koji dolaze od ulazno/izlaznih uređaja ***U/I*** kao impuls pamte se u flip-flopovima za prihvatanje prekida $PRINTR_1$ do $PRINTR_7$, respektivno. U obradi prekida koriste se sadržaji flip-flopova $PRINTR_1$ do $PRINTR_7$. Kada se

u okviru opsluživanja zahteva za prekid prihvati neki od maskirajućih zahteva za prekid, odgovarajući flip-flop $PRINTR_1$ do $PRINTR_7$ se postavlja na vrednost 0. Postavljanje flip-flopova $PRINTR_1$ do $PRINTR_7$ na vrednost 0 se realizuje u okviru opsluživanja spoljašnjeg maskirajućeg zahteva za prekid vrednošću 1 odgovarajućeg signala **clINTR$_1$** do **clINTR$_7$**, respektivno. Postavljanje odgovarajućeg flip-flopa $PRINTR_1$ do $PRINTR_7$ na vrednost 0 se realizuje tako što se u okviru opsluživanja spoljašnjeg maskirajućeg zahteva za prekid generiše vrednost 1 signala **clINTR**. S obzirom da signali **prl$_2$** do **prl$_0$** daju binarnu vrednost jedne od linija intr$_1$ do intr$_7$ po kojoj je prihvaćeni zahtev za prekid stigao, ovi signali se koriste da selektuju odgovarajući flip-flop $PRINTR_1$ do $PRINTR_7$ koji se vrednošću 1 signala **clINTR** postavlja na vrednost 0.

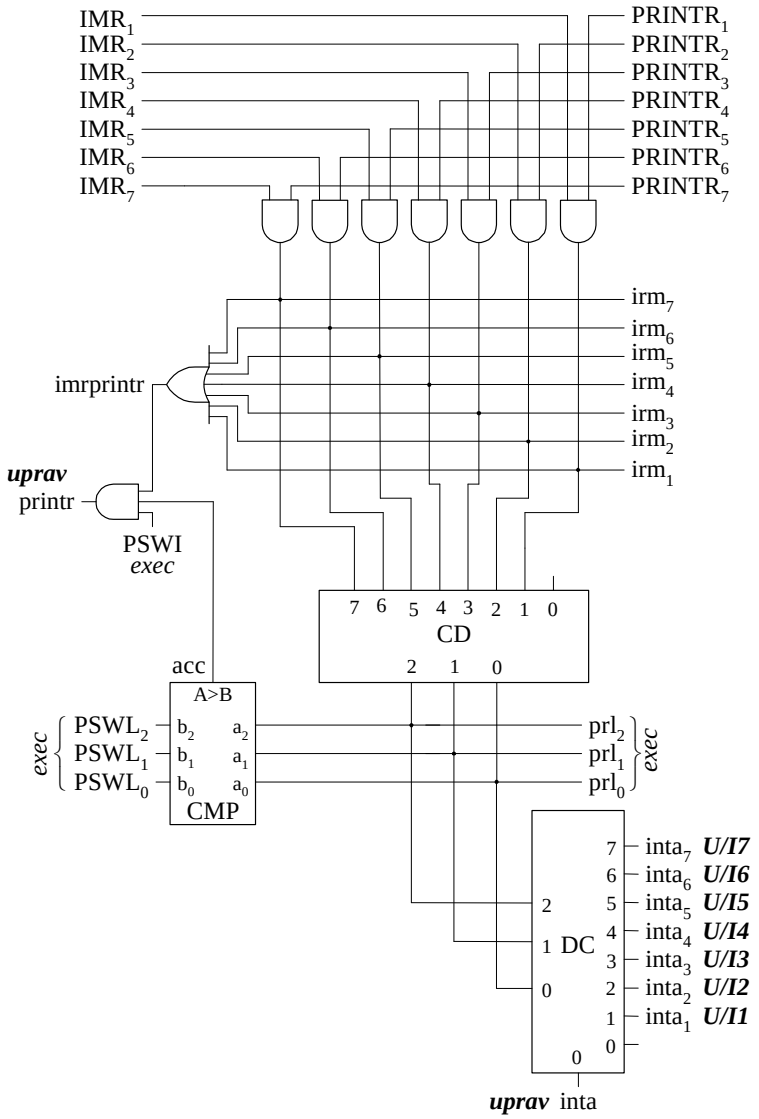
Dekoder DC u okviru opsluživanja spoljašnjeg maskirajućeg zahteva za prekid pri vrednosti 1 signala **clINTR** daje vrednost 1 jednog od signala **clINTR$_1$** do **clINTR$_7$**. Koji će od signala **clINTR$_1$** do **clINTR$_7$** tada imati vrednost 1 zavisi od vrednosti signala **prl$_2$** do **prl$_0$** koji daju binarnu vrednost linije intr$_1$ do intr$_7$ po kojoj je prihvaćeni zahtev za prekid stigao.

Registar maske $IMR_{15...0}$ služi za pojedinačno maskiranje spoljašnjih maskirajućih zahteva za prekid intr$_1$ do intr$_7$. Zahtevima za prekid intr$_1$ do intr$_7$ odgovaraju razredi IMR_1 do IMR_7 registra maske $IMR_{15...0}$, dok se preostali razredi ne koriste. Upis sadržaja sa linija $AW_{15...0}$ se obavlja ukoliko signal **ldIMR** ima vrednost 1. Upis u registar $IMR_{15...0}$ se realizuje samo kod izvršavanja instrukcije STIMR, koja služi za upis sadržaja akumulatora $AW_{15...0}$ u registar procesora $IMR_{15...0}$.

Kombinacione prekidačke mreže za formiranje signala spoljašnjih maskirajućih prekida **printr** se sastoji od logičkih ILI i I elemenata, kodera CD i komparatora CMP (slika 3).

Signal maskirajućeg prekida **printr** ima vrednost 1 ukoliko signali **imrprintr**, **acc** i **PSWI** imaju vrednosti 1. Signal **imrprintr** ima vrednost 1 ukoliko barem jedan od signala **irm$_1$** do **irm$_7$** ima vrednost 1. Ovo će se desiti ukoliko se barem u jednom od flip-flopova $PRINTR_1$ do $PRINTR_7$ (slika 2) nalazi vrednost 1 i ukoliko je u odgovarajućem razredu IMR_1 do IMR_7 registra maske $IMR_{15...0}$ takođe vrednost 1. Signal **acc** na izlazu komparatora CMP ima vrednost 1 ukoliko je prioritet pristiglog spoljašnjeg maskirajućeg zahteva za prekid najvišeg prioriteta koji nije maskiran viši od prioriteta tekućeg programa. Prioritet pristiglog spoljašnjeg maskirajućeg zahteva za prekid najvišeg prioriteta koji nije maskiran određen je signalima **prl$_2$** do **prl$_0$** na izlazu kodera prioriteta CD, dok je prioritet tekućeg programa određenog signalima **PSWL$_2$** do **PSWL$_0$** registra $PSW_{15...0}$ (slika 3). Signal **PSWI** dolazi sa izlaza odgovarajućeg razreda registra $PSW_{15...0}$ i njegovim vrednostima 1 i 0 se dozvoljavaju i maskiraju svi maskirajući prekidi, respektivno (slika 5).

Dekoder DC služi za generisanje signala potvrda inta$_7$ do inta$_1$ koji se šalju kontrolerima periferija kao potvrda da se prihvata zahtev za prekid. U toku faze *opsluživanje zahteva za prekid* procesor onda kada je spreman da prihvati broj ulaza od kontrolera periferije najvišeg prioriteta koji nije selektivno maskiran generiše vrednost 1 signala potvrde inta. Ovaj signal se preko dekodera DC prosleđuje po jednoj od linija inta$_7$ do inta$_1$ na osnovu vrednosti signala **prl$_2$** do **prl$_0$** na izlazu kodera prioriteta CD.



Slika 3 Blok intr (drugi deo)

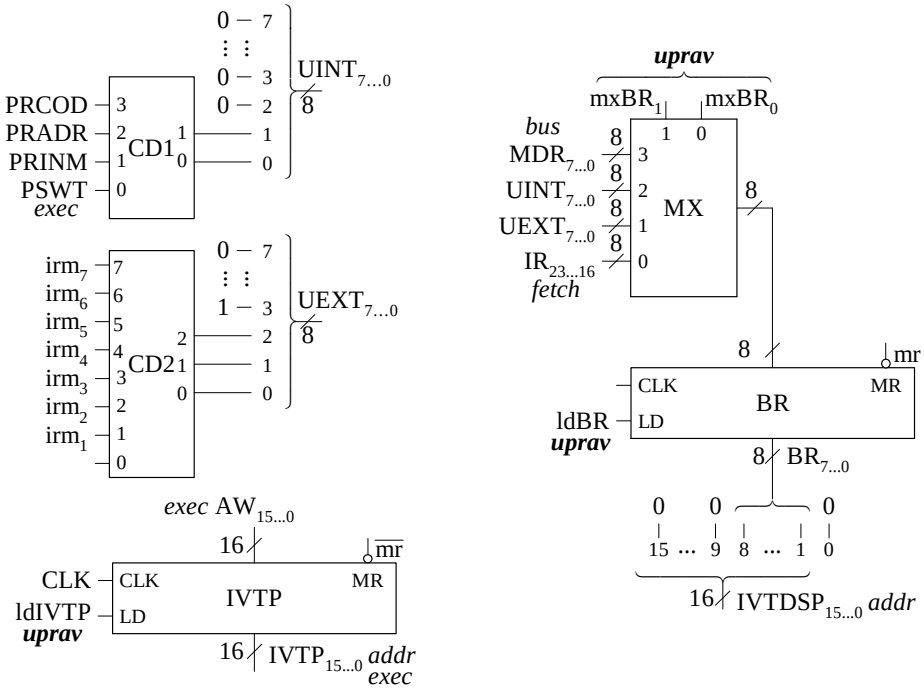
Kombinacione i sekvencijalne mreže za formiranje pomeraja $IVTDSP_{15...0}$ za tabelu sa adresama prekidnih rutina se sastoje od kodera CD1 i CD2, i registra $BR_{7...0}$ sa multiplekserom MX (slika 4).

Koder CD1 služi za formiranje broja ulaza u tabelu sa adresama prekidnih rutina za unutrašnje prekide, spoljašnji nemaskirajući prekid i unutrašnji prekid zbog zadatog režima rada prekid posle svake instrukcije. Ovi prekidi su određeni signalima na izlazima flip-flopova PSWT za unutrašnji prekid usled zadatog režima rada prekid posle svake instrukcije, PRINM za spoljašnji nemaskirajući prekid, PRADR za unutrašnji prekid zbog nelegalnog adresiranja i PRCOD za unutrašnji prekid zbog nelegalnog koda operacije. Signali sa izlaza flip-flopova PSWT, PRINM, PRADR i PRCOD dovedeni su na ulaze 0 do 3 kodera prioriteta CD1 po rastućim prioritetima. Na izlazu kodera dobija se broj zahteva za prekid najvišeg prioriteta binarno kodiran sa dva bita koji predstavljaju dva najniža bita broja ulaza. Bitovi 2 do 7 broja ulaza imaju vrednost 0 i sa dva bita na izlazu kodera CD1 formiraju 8-mo bitnu vrednost broja ulaza $UINT_{7...0}$. Ovim se dobijaju brojevi ulaza od 0 do 3. Ovako formirane vrednosti brojeva ulaza za ove prekide su u skladu sa odlukom da adrese prekidnih rutina za ova četiri prekida budu smeštene u ulaze 0 do 3 tabele sa adresama prekidnih rutina.

Brojevi ulaza u tabelu sa adresama prekidnih rutina za spoljašnje maskirajuće prekide $intr_1$ do i $intr_7$ se generišu na dva načina i to ili fiksno na osnovu pozicija linija $intr_1$ do i $intr_7$ po kojima u procesor dolaze zahtevi za prekid od kontrolera periferija ili promenljivo na osnovu brojeva ulaza koje šalju kontroleri periferija. Razred PSWP registra $PSW_{15...0}$ sadrži indikator *promenljivi broj ulaza* koji vrednostima 0 i 1 određuje da li se broj ulaza generiše fiksno ili promenljivo, respektivno.

Koder CD2 služi za fiksno generisanje broja ulaza u tabelu sa adresama prekidnih rutina za spoljašnje maskirajuće prekide $intr_1$ do i $intr_7$. Signali spoljašnjih maskirajućih prekida $intr_1$ do $intr_7$ posle eventualnog maskiranja razredima IMR_1 do IMR_7 registra maske $IMR_{15...0}$ pojavljuju se kao signali $\mathbf{irm_1}$ do $\mathbf{irm_7}$ (slika 3). Ovi signali se vode na ulaze 1 do 7 kodera prioriteta CD2 po rastućim prioritetima. Na izlazu kodera dobija se broj zahteva za prekid najvišeg prioriteta binarno kodiran sa tri bita koji predstavljaju tri najniža bita broja ulaza. Bit 3, koji ima vrednost 1 i bitovi 4 do 7, koji imaju vrednost 0, sa tri bita na izlazu kodera CD2 formiraju 8-mo bitnu vrednost broja ulaza $UEXT_{7...0}$. Ovim se dobijaju brojevi ulaza od 9 do 15. Ovako formirane vrednosti brojeva ulaza za ove prekide su u skladu sa odlukom da adrese prekidnih rutina za ovih sedam prekida budu smeštene u ulaze 9 do 15 tabele sa adresama prekidnih rutina.

U slučaju promenljivog generisanja broja ulaza procesor šalje po jednoj od linija $inta_7$ do $inta_1$ vrednost 1 kontroleru periferije najvišeg prioriteta koji je generisao zahtev za prekid a čiji zahtev za prekid nije selektivno maskiran kao potvrdu da se zahtev za prekid prihvata i da kontroler periferije treba da pošalje broj ulaza po linijama podataka magistrale. Ovde se pretpostavlja da su tokom inicijalizacije sistema određeni ulazi tabele sa adresama prekidnih rutina popunjeni adresama prekidnih rutina periferija i da su ti brojevi ulaza upisani u posebne registre svih kontrolera periferija.



Slika 4 Blok intr (treći deo)

Registar $BR_{7...0}$ sa multiplekserom MX služi za selekciju i pamćenje broja ulaza u tabelu sa adresama prekidnih rutina (slika 4). Upis sadržaja sa izlaza multipleksera MX u registar

$BR_{7...0}$ se obavlja vrednošću 1 signala **ldBR**. Sadržaj registra $BR_{7...0}$ se u okviru opsluživanja zahteva za prekid koristi za formiranje pomeraja $IVTDSP_{15...0}$ za ulazak u tabelu sa adresama prekidnoh rutina. Kako adresa prekidne rutine zauzima dve susedne memorijske lokacije, to se pomeraj $IVTDSP_{15...0}$ dobija pomeranjem sadržaja registra $BR_{7...0}$ za jedno mesto ulevo.

Multiplekser MX je 8-mo razredni multiplekser sa 4 ulaza. Na ulaze 0 do 3 multipleksera se vode sadržaji $IR_{23...16}$, $UEXT_{7...0}$, $UINT_{7...0}$ i $MDR_{7...0}$ koji predstavljaju brojeve ulaza u tabelu sa adresama prekidnih rutina za instrukciju prekida INT, za spoljašnje maskirajuće prekide ukoliko su fiksni, za unutrašnje prekide i spoljašnji nemaskirajući prekid, i za spoljašnje maskirajuće prekide ukoliko su promenljivi, respektivno. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 00 do 11 upravljačkih signala $\mathbf{mxBR_1}$ i $\mathbf{mxBR_0}$. Sadržaj $IR_{23...16}$ se propušta kroz multiplekser za instrukciju prekida INT, $UEXT_{7...0}$ za spoljašnje maskirajuće prekide ukoliko su ulazi fiksni, $UINT_{7...0}$ za unutrašnje prekide i spoljašnji nemaskirajući prekid, i $MDR_{7...0}$ za spoljašnje maskirajuće prekide ukoliko su promenljivi. Sadržaj sa izlaza multipleksera MX se vodi na ulaze registra $BR_{7...0}$.

Registar $IVTP_{15...0}$ je ukazivač na tabelu sa adresama prekidnih rutina i sadrži početnu adresu tabele. Upis sadržaja sa linija $AW_{15...0}$ u registar $IVTP_{15...0}$ se obavlja vrednošću 1 signala **ldIVTP**. Upis u registar $IMR_{15...0}$ se realizuje samo kod izvršavanja instrukcije STIVTP, koja služi za upis sadržaja akumulatora $AW_{15...0}$ u registar procesora $IVTP_{15...0}$.

Registar $PSW_{15...0}$ je 16-to razredni registar koji sadrži indikatore programske statusne reči procesora (slika 5). Registar $PSW_{15...0}$ se sastoji od flip-flopova PSWI, PSWT, PSWP, $PSWL_2$, $PSWL_1$, $PSWL_0$, PSWV, PSWC, PSWZ i PSWN, koji predstavljaju razrede $PSW_{15...13}$ i $PSW_{6...0}$, respektivno, dok razredi $PSW_{12...7}$ ne postoje.

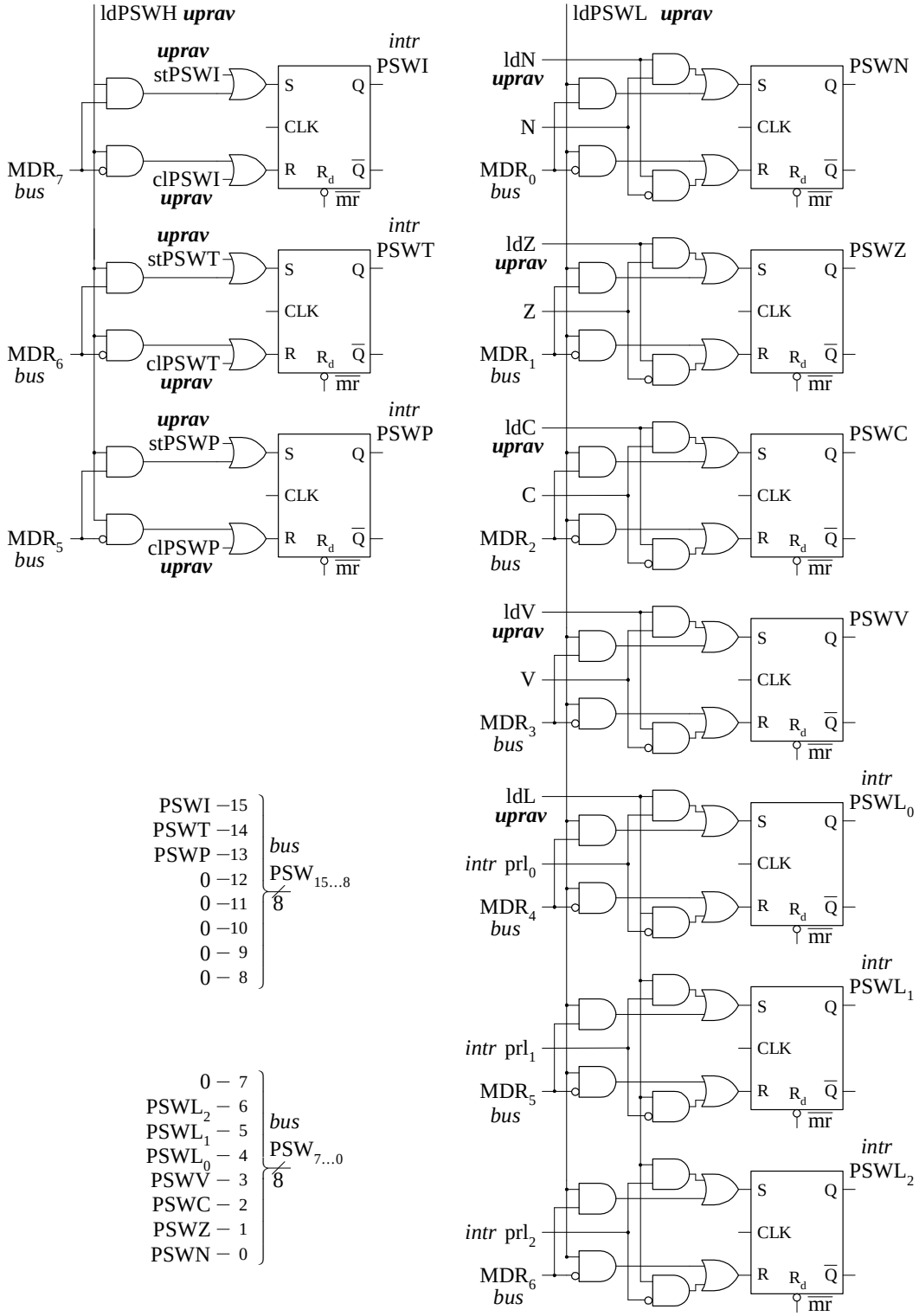
Flip-flop PSWI sadrži indikator *maskiranje svih maskirajućih prekida* (*interrupt*). Flip-flop PSWI se postavlja na vrednost 1 vrednošću 1 signala **stPSWI** u okviru faze *izvršavanje instrukcije* INTE i na vrednost 0 vrednošću 1 signala **clPSWI** u okviru faze *izvršavanje instrukcije* INTD kao i u okviru faze *opsluživanje zahteva za prekid* svih instrukcija ukoliko je tokom izvršavanja instrukcije stigao zahtev za prekid pa se prolazi kroz ovu fazu.

Flip-flop PSWT sadrži indikator *prekid posle svake instrukcije* (*trap*). Flip-flop PSWT se postavlja na vrednost 1 vrednošću 1 signala **stPSWT** u okviru faze *izvršavanje instrukcije* TRPE i na vrednost 0 vrednošću 1 signala **clPSWT** u okviru faze *izvršavanje instrukcije* TRPD kao i u okviru faze *opsluživanje zahteva za prekid* svih instrukcija ukoliko je tokom izvršavanja instrukcije stigao zahtev za prekid pa se prolazi kroz ovu fazu.

Flip-flop PSWP sadrži indikator *promenljivi/fiksni broj ulaza*. Flip-flop PSWP se postavlja na vrednost 1 vrednošću 1 signala **stPSWP** u okviru faze *izvršavanje instrukcije* PRME i na vrednost 0 vrednošću 1 signala **clPSWP** u okviru faze *izvršavanje instrukcije* PRMD.

Flip-flopovi $PSWL_2$ do $PSWL_0$ sadrže indikatore *tekući nivo prioriteta* (*level*). U flip-flopove $PSWL_2$ do $PSWL_0$ se vrednošću 1 signala **ldL** u okviru faze *opsluživanje zahteva za prekid* od ulazno/izlaznih uređaja upisuju vrednosti signala $\mathbf{prl_2}$ do $\mathbf{prl_0}$, čime se ovi flip-flopovi postavljaju na vrednost nivoa prioriteta ulazno/izlaznog uređaja na čiju se prekidnu rutinu prelazi.

Flip-flop PSWV sadrži indikator *overflow*. Flip-flop PSWC sadrži indikator *carry/borrow*. Flip-flop PSWZ sadrži indikator *zero*. Flip-flop PSWN sadrži indikator *negative*.



Slika 5 Blok exec (drugi deo)

U razrede $PSW_{15...13}$ koji predstavljaju 3 najstarija razreda registra $PSW_{15...0}$ se vrednošću 1 signala **ldPSWH** upisuje sadržaj razreda $MDR_{7...5}$ prihvatnog registra podatka $MDR_{7...0}$ bloka *bus*, dok se u razrede $PSW_{6...0}$ koji predstavljaju 7 najmlađih razreda registra $PSW_{15...0}$ vrednošću 1 signala **ldPSWL** upisuje sadržaj razreda $MDR_{6...0}$ prihvatnog registra podatka $MDR_{7...0}$. Ovo se koristi prilikom izvršavanja instrukcije RTI da se sadržajem sa vrha steka

restaurira sadržaj registra $PSW_{15...0}$. Sadržaji 3 najstarija razreda $PSW_{15...13}$, 7 najmlađih razreda $PSW_{6...0}$ i vrednosti 0 za nepostojeće razrede $PSW_{12...7}$ registra $PSW_{15...0}$ se vode posebno kao 8 starijih razreda $PSW_{15...8}$ i 8 mlađih razreda $PSW_{7...0}$ registra $PSW_{15...0}$ u blok *bus* u kome se upisuju u prihvatni registar podatka $MDR_{7...0}$. Ovo se koristi prilikom opsluživanja zahteva za prekida da se sadržaj registra $PSW_{15...0}$ stavi na vrh steka.

1.2.2 Upravljačka jedinica

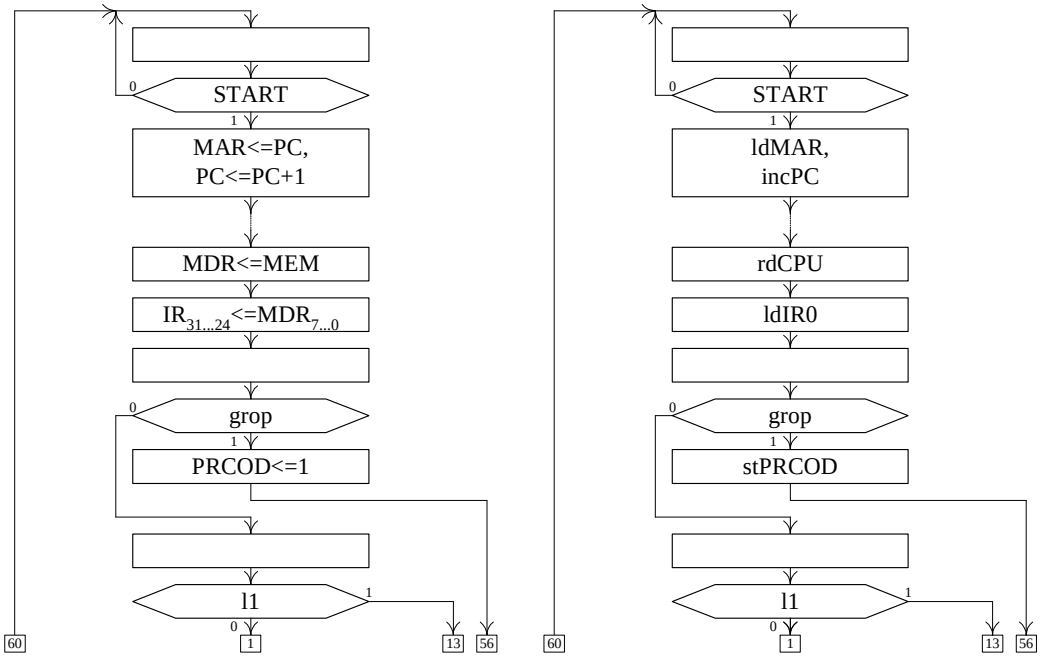
U ovom poglavlju se najpre daju delovi sekvence upravljačkih signala relevantni za mehanizam prekida i zatim razmatraju uslovi po kojima se generiše signal **prekid**.

1.2.2.1 Sekvenca upravljačkih signala

U tabeli 1 se daju delovi sekvence upravljačkih signala faza izvršavanja instrukcija relevantni za mehanizam prekida.

Tabela 1 Sekvenca upravljačkih signala

! Čitanje instrukcije !



Slika 6 Čitanje instrukcije (prvi deo)

! U koraku $step_{00}$ se proverava vrednost signala **START** bloka *exec* koji vrednostima 0 i 1 ukazuje da li je processor neaktivan ili aktivan, respektivno. Ukoliko je procesor neaktivan ostaje se u koraku $step_{00}$, dok se u suprotnom slučaju prelazi na korak $step_{01}$. !

$step_{00}$ *br* (*if* $\overline{\textbf{START}}$ *then* $step_{00}$);

! U koracima $step_{01}$ do $step_{04}$ se čita prvi bajt instrukcije i smešta u razrede $IR_{31...24}$ prihvatnog registra instrukcije $IR_{31...0}$.

$step_{01}$ **ldMAR**, **incPC**;
...
$step_{03}$ **rdCPU**,
$step_{04}$ **ldIR0**;

!U koraku $step_{05}$ se proverava vrednost signala **gropr** bloka *fetch* da bi se utvrdilo da li prvi bajt instrukcije sadrži korektnu vrednost koda operacije. U zavisnosti od toga da li signal **gropr** ima vrednost 1 ili 0, prelazi se na korak $step_{06}$ ili $step_{07}$, respektivno. Ukoliko prvi bajt instrukcije sadrži nekorektnu vrednost koda operacije, u koraku $step_{06}$ se vrednošću 1 signala **stPRCOD** bloka *intr* u flip-flop **PRCOD** upisuje vrednost 1 i bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida.!

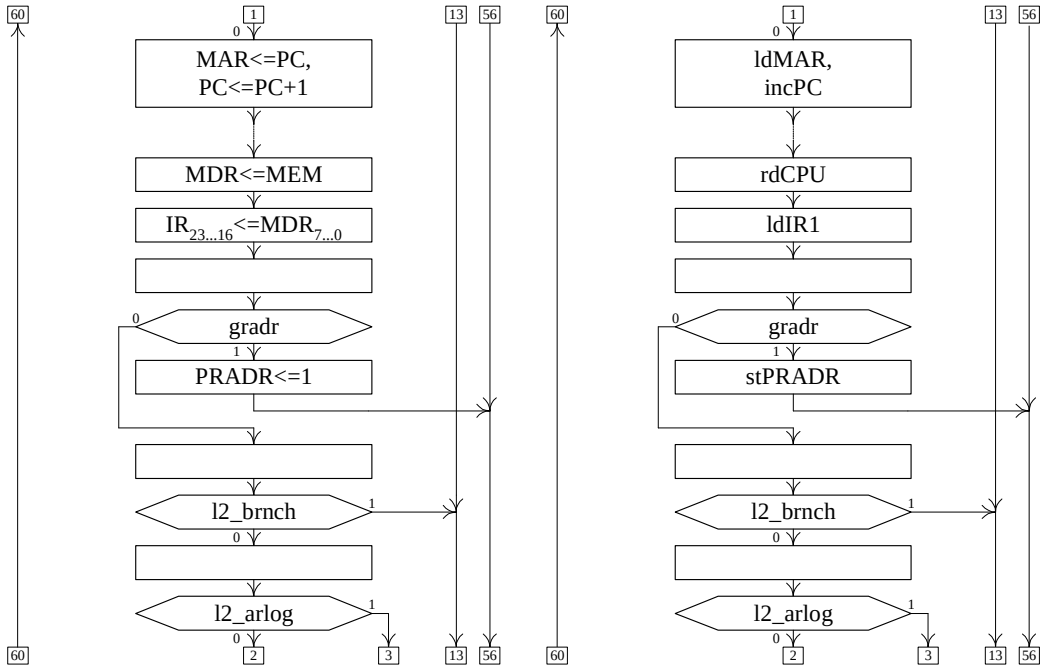
$step_{05}$ *br* (*if* $\overline{\textbf{gropr}}$ *then* $step_{07}$);
$step_{06}$ **stPRCOD**,
br $step_{C0}$;

! U koraku $step_{07}$ se proverava vrednost signala **l1** bloka *fetch* da bi se utvrdilo da li je dužina instrukcije jedan bajt ili više bajtova. U zavisnosti od toga da li signal **l1** ima vrednost 1 ili 0, prelazi se na korak $step_{50}$ i fazu izvršavanje operacije ili $step_{07}$ i produžava sa čitanjem drugog bajta instrukcije. !

$step_{07}$ *br* (*if* **l1** *then* $step_{50}$);

! U koracima $step_{08}$ do $step_{0B}$ se čita drugi bajt instrukcije i smešta u razrede $IR_{23...16}$ prihvatnog registra instrukcije $IR_{31...0}$.!

$step_{08}$ **ldMAR**, **incPC**;
...
$step_{0A}$ **rdCPU**,
$step_{0B}$ **ldIR1**;



Slika 7 Čitanje instrukcije (drugi deo)

!U koraku $step_{0C}$ se proverava vrednost signala **gradr** bloka *fetch* da bi se utvrdilo da li je nekorektno adresiranje specificirano. Ovo se dešava ukoliko se za instrukcije STB i STW specificira neposredno adresiranje. U zavisnosti od toga da li signal **gradr** ima vrednost 1 ili 0, prelazi se na korak $step_{0D}$ ili $step_{0E}$, respektivno. Ukoliko je nekorektno adresiranje specificirano, u koraku $step_{0D}$ se vrednošću 1 signala **stPRADR** bloka *intr* u flip-flop **PRADR** upisuje vrednost 1 i bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida.!

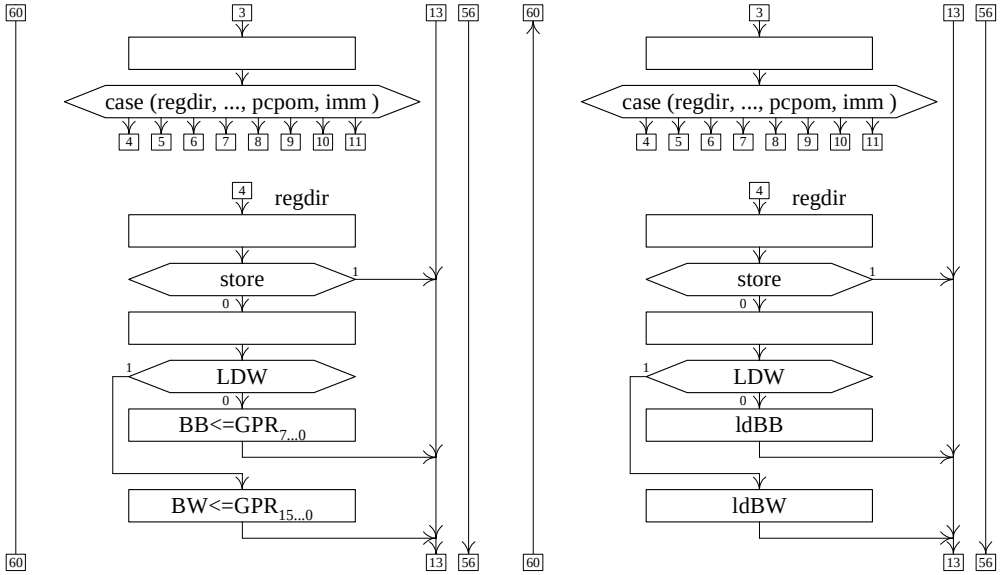
$step_{0C}$ *br* (*if* $\overline{\textbf{gradr}}$ *then* $step_{0E}$);
$step_{0D}$ **stPRADR**,
br $step_{C0}$;

! U koracima $step_{0E}$ i $step_{0F}$ se proverava vrednost signala **l2_brnch** i **l2_arlog** bloka *fetch* da bi se utvrdilo da li je dužina instrukcije dva bajta ili više bajtova. Iz koraka $step_{0E}$ se u zavisnosti od toga da li signal **l2_brnch** ima vrednost 1 ili 0, prelazi na korak $step_{50}$ i fazu izvršavanje operacije ili $step_{0F}$ i proveru vrednosti signala **l2_arlog**. Iz koraka $step_{0F}$ se u zavisnosti od toga da li signal **l2_arlog** ima vrednost 1 ili 0, prelazi na korak $step_{20}$ i fazu formiranje adrese i čitanje operanda ili $step_{10}$ i produžava sa čitanjem bajtova instrukcije.!

$step_{0E}$ *br* (*if* **l2_brnch** *then* $step_{50}$);
$step_{0F}$ *br* (*if* **l2_arlog** *then* $step_{20}$);
...

! U sledećim koracima se čita treći i četvrti bajt instrukcije i smešta u razrede $IR_{15...8}$ i $IR_{7...0}$ prihvatnog registra instrukcije $IR_{31...0}$. U slučaju instrukcija bezuslovnog skoka i skoka na potprogram posle čitanja trećeg bajta prelazi se na fazu izvršavanje operacije, dok se u slučaju aritmetičkih i logičkih instrukcija prelazi na fazu formiranje adrese i čitanje operanda i to u zavisnosti od načina adresiranja ili posle trećeg ili posle četvrtog bajta. !

! Formiranje adrese i čitanje operanda !



Slika 8 Formiranje adrese i čitanje operanda (prvi deo)

! U korak $step_{20}$ se dolazi radi izvršavanja faze formiranje adrese i čitanje operanda. U koraku $step_{20}$ se realizuje višestruki uslovni skok na jedan od koraka $step_{21}$, $step_{25}$, …, $step_{41}$ u zavisnosti od toga koji od signala adresiranja **regdir**, **...**, **pcpom**, **imm** bloka *addr* ima vrednost 1. Za sve instrukcije, sem instrukcija STB i STW, operand se smešta u prihvatni registar podatka $BB_{7...0}$ ili $BW_{15...0}$. Na kraju se prelazi na $korak_{50}$ i fazu izvršavanje operacije !

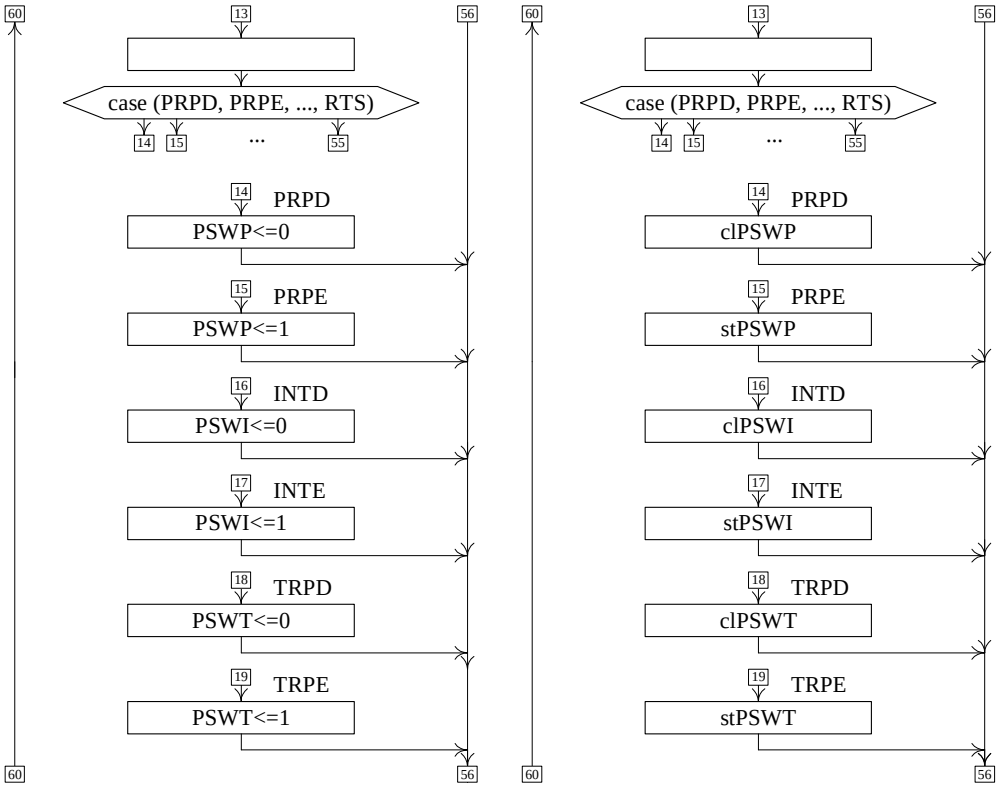
$step_{20}$ *br* (*case* (**regdir**, **...**, **pcpom**, **imm**) *then*
(**regdir**, $step_{21}$), ..., (**pcpom**, $step_{37}$), (**imm**, $step_{41}$));

...

! Izvršavanje operacije !

! U korak $step_{50}$ se dolazi radi izvršavanja operacije. U koraku $step_{50}$ se realizuje višestruki uslovni skok na jedan od koraka $step_{51}$, $step_{52}$, ..., $step_{B6}$ u zavisnosti od toga koji od signala operacija **PRPD**, **PRPE**, ..., **RTS** ima vrednost 1. !

$step_{50}$ *br* (*case* (**PRPD**, **PRPE**,
INTD, **INTE**, **TRPD**, **TRPE**,
LDW, **STIMR**,
JMP, **INT**, **RTI**)
then
(**PRPD**, $step_{51}$), (**PRPE**, $step_{52}$),
(**INTD**, $step_{53}$), (**INTE**, $step_{54}$), (**TRPD**, $step_{55}$), (**TRPE**, $step_{56}$),
(**LDW**, $step_{59}$), (**STIMR**, $step_{8A}$),
(**JMP**, $step_{32}$), (**INT**, $step_{A4}$), (**RTI**, $step_{AE}$));



Slika 9 Izvršavanje operacije (prvi deo)

! PRPD !

! U korak $step_{51}$ se dolazi iz $step_{50}$ ukoliko signal operacije PRPD ima vrednost 1. Vrednošću 1 signala **clPSWP** se razred PSWP bloka *exec* postavlja na vrednost 0. Iz koraka $step_{51}$ se bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida. !

$step_{51}$ **clPSWP**,
br $step_{C0}$;

! PRPE !

! U korak $step_{52}$ se dolazi iz $step_{50}$ ukoliko signal operacije PRPE ima vrednost 1. Vrednošću 1 signala **stPSWP** se razred PSWP bloka *exec* postavlja na vrednost 1. Iz koraka $step_{52}$ se bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida!

$step_{52}$ **stPSWP**,
br $step_{C0}$;

! INTD !

! U korak $step_{53}$ se dolazi iz $step_{50}$ ukoliko signal operacije **INTD** ima vrednost 1. Vrednošću 1 signala **clPSWI** se razred PSWI bloka *exec* postavlja na vrednost 0. Iz koraka $step_{53}$ se bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida. !

$step_{53}$ **clPSWI**,
br $step_{C0}$;

! INTE !

! U korak $step_{54}$ se dolazi iz $step_{50}$ ukoliko signal operacije **INTE** ima vrednost 1. Vrednošću 1 signala **stPSWI** se razred PSWI bloka *exec* postavlja na vrednost 1. Iz koraka $step_{54}$ se bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida.!

$step_{54}$ **stPSWI**,
br $step_{C0}$;

! TRPD !

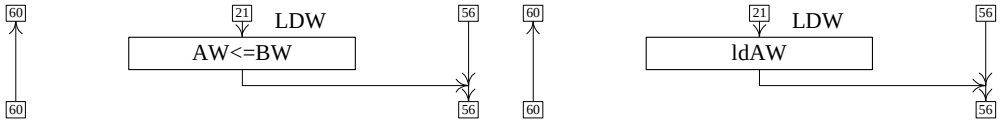
! U korak $step_{55}$ se dolazi iz $step_{50}$ ukoliko signal operacije **TRPD** ima vrednost 1. Vrednošću 1 signala **clPSWT** se razred PSWT bloka *exec* postavlja na vrednost 0. Iz koraka $step_{55}$ se bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida. !

$step_{55}$ **clPSWT**,
br $step_{C0}$;

! TRPE !

! U korak $step_{56}$ se dolazi iz $step_{50}$ ukoliko signal operacije **TRPE** ima vrednost 1. Vrednošću 1 signala **stPSWT** se razred PSWT bloka *exec* postavlja na vrednost 1. Iz koraka $step_{56}$ se bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida.!

$step_{56}$ **stPSWT**,
br $step_{C0}$;

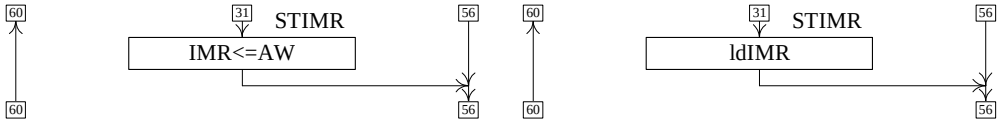


Slika 10 Izvršavanje operacije (drugi deo)

! LDW !

! U korak $step_{59}$ se dolazi iz $step_{50}$ ukoliko signal operacije **LDW** ima vrednost 1. U ovom koraku se vrednošću 1 signala **ldAW** bloka *exec* se sadržaj registra $BW_{15...0}$ i upisuje u registar $AW_{15...0}$ i prelazi na korak $step_{C0}$ i fazu opsluživanje prekida. !

$step_{59}$ **ldAW**,
br $step_{C0}$;

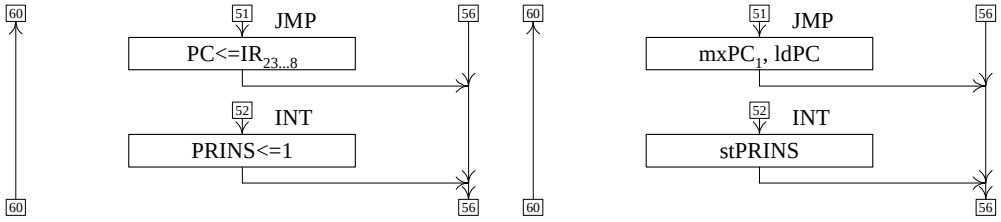


Slika 11 Izvršavanje operacije (deveti deo)

! STIMR !

! U korak $step_{8A}$ se dolazi iz $step_{50}$ ukoliko signal operacije STIMR ima vrednost 1. U fazi izvršavanja ove instrukcije se sadržaj registra $AW_{15...0}$ bloka *exec* upisuje u registar $IMR_{15...0}$ bloka *intr*. Stoga se vrednošću 1 signala **ldIMR** sadržaj registra $AW_{15...0}$ upisuje u registar $IMR_{15...0}$. Na kraju se iz koraka $step_{8A}$ bezuslovno prelazi na korak $step_{C0}$ i fazu opsluživanje prekida. !

$step_{8A}$ **ldIMR**,
br $step_{C0}$;



Slika 12 Izvršavanje operacije (četrnaesti deo)

! JMP !

! U korak $step_{A3}$ se dolazi iz $step_{50}$ ukoliko signal operacije **JMP** ima vrednost 1. U fazi izvršavanja ove instrukcije se realizuje bezuslovni skok na adresu koja je data u samoj instrukciji. Stoga se sadržaj registra $IR_{23...8}$ bloka *fetch* koji predstavlja adresu skoka vrednošću 1 signala **ldPC** upisuje u registar $PC_{15...0}$. Pored toga iz koraka $step_{A3}$ se bezuslovno prelazi na $step_{C0}$ i fazu opsluživanje prekida. !

$step_{A3}$ **$mxPC_1$, ldPC**,
br $step_{C0}$;

! INT !

! U korak $step_{A4}$ se dolazi iz $step_{50}$ ukoliko signal operacije **INT** ima vrednost 1. U fazi izvršavanja ove instrukcije se samo evidentira da se radi o instrukciji prekida i prelazi na fazu opsluživanje prekida iz koje se zatim skače na prekidnu rutinu na osnovu broja ulaza u tabelu sa adresama

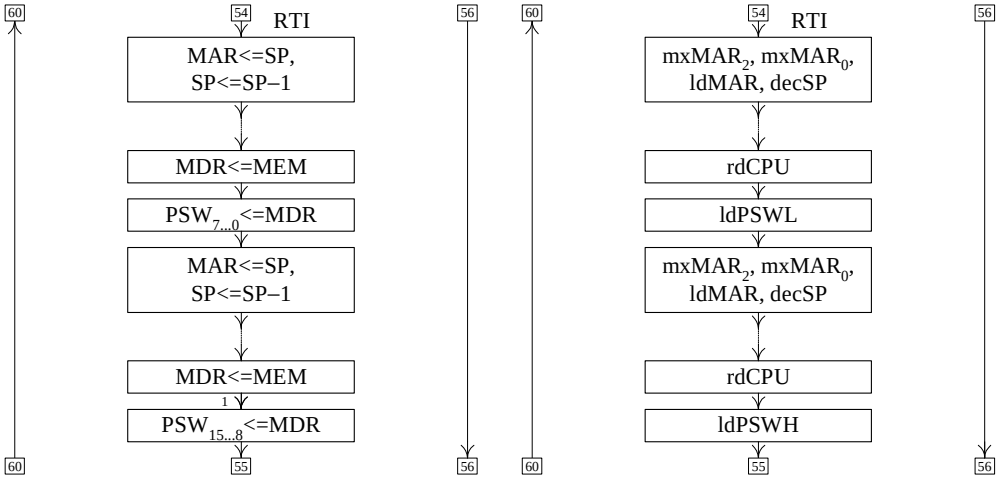
prekidnih rutina koji je dat u samoj instrukciji. Stoga se vrednošću 1 signala **stPRINS** flip-flop PRINS bloka *intr* postavlja na vrednost 1 i bezuslovno prelazi na $step_{C0}$ i fazu opsluživanje prekida. !

$step_{A4}$ **stPRINS**,
br $step_{C0}$;

! RTI !

! U korak $step_{AE}$ se dolazi iz $step_{50}$ ukoliko signal operacije **RTI** ima vrednost 1. U fazi izvršavanja ove instrukcije vrednostima sa steka se restauriraju programska statusna reč $PSW_{15..0}$ i programski brojač $PC_{15...0}$. U koracima $step_{AE}$ do $step_{B5}$ se najpre vrednošću sa steka restaurira programska statusna reč $PSW_{15...0}$ bloka *exec*. Sa steka se najpre skida niži a zatim i viši bajt registra $PSW_{15...0}$. Stoga se u koraku $step_{AE}$ vrednošću 1 signala **ldMAR** bloka *bus* sadržaj registra $SP_{15...0}$ bloka *addr* upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koracima $step_{AF}$ i $step_{B0}$. Na kraju se u koraku $step_{B1}$ vrednošću 1 signala **ldPSWL** sadržaj registra $MDR_{7...0}$ bloka *bus* upisuje u niži bajt registra $PSW_{7...0}$ bloka *exec*. Potom se u koraku $step_{B2}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koracima $step_{B3}$ i $step_{B4}$. Na kraju se u koraku $step_{B5}$ vrednošću 1 signala **ldPSWH** sadržaj registra $MDR_{7...0}$ bloka *bus* upisuje u viši bajt registra $PSW_{15...8}$. Time je 16-to bitna vrednost skinuta sa steka i smeštena u registar $PSW_{15...0}$, pa se prelazi na sledeći korak počev od koga se sa steka skida 16-to bitna vrednost i smešta u registar $PC_{15...0}$. !

$step_{AE}$ **..., ldMAR**, **decSP**;
$step_{AF}$ ...
$step_{B0}$ **rdCPU**;
$step_{B1}$ **ldPSWL**;
$step_{B2}$ **..., ldMAR**, **decSP**;
$step_{B3}$ ...
$step_{B4}$ **rdCPU**;
$step_{B5}$ **ldPSWH**;

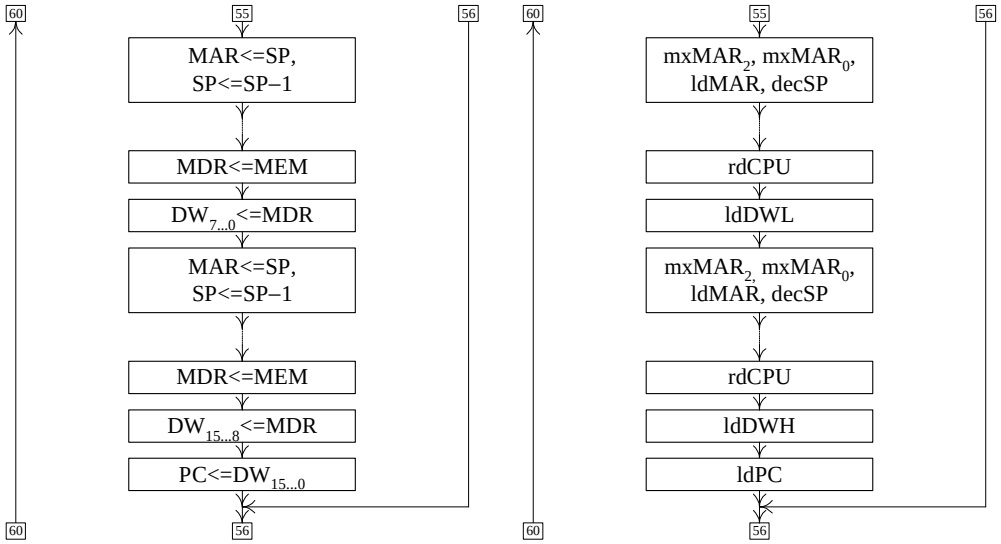


Slika 13 Izvršavanje operacije (šesnaesti deo)

! U koracima $step_{B6}$ do $step_{BE}$ se vrednošću sa steka restaurira programski brojač $PC_{15...0}$. Sa steka se najpre skida niži a zatim i viši bajt registra $PC_{15...0}$. Stoga se u koraku $step_{B6}$ vrednošću 1 signala **ldMAR** bloka *bus* sadržaj registra $SP_{15...0}$ bloka *addr* upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koracima $step_{B7}$ i $step_{B8}$. Na kraju se u koraku $step_{B9}$ vrednošću 1 signala **ldDWL** sadržaj registra $MDR_{7...0}$ bloka *bus* upisuje u niži bajt registra $DW_{7...0}$ bloka *bus*. Potom se u koraku $step_{BA}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koracima $step_{BB}$ i $step_{BC}$. Na kraju se u koraku $step_{BD}$ vrednošću 1 signala **ldDWH** sadržaj registra $MDR_{7...0}$ upisuje u viši bajt registar $DW_{15...8}$. Time je 16-to bitna vrednost skinuta sa steka i smeštena u registar $DW_{15...0}$. Konačno se u

koraku $step_{BE}$ sadržaj registra $DW_{15...0}$ vrednošću 1 signala **ldPC** upisuje u registar $PC_{15...0}$ i bezuslovno prelazi na $step_{C0}$ i fazu opsluživanje prekida. !

$step_{B6}$ **..., ldMAR**, **decSP**;
$step_{B7}$ ...
$step_{B8}$ **rdCPU**;
$step_{B9}$ **ldDWL**;
$step_{BA}$ **..., ldMAR**, **decSP**;
$step_{BB}$ ...
$step_{BC}$ **rdCPU**;
$step_{BD}$ **ldDWH**;
$step_{BE}$ **ldPC**,
br $step_{C0}$**;**



Slika 14 Izvršavanje operacije (sedamnaesti deo)

! Opsluživanje prekida !

! U korak $step_{C0}$ se dolazi koraka $step_{06}$ ukoliko signal **PRCOD** ima vrednost 1, koraka $step_{0D}$ ukoliko signal **PRADR** ima vrednost 1 i koraka $step_{51}$, $step_{52}$, ..., $step_{BE}$ na završetku faze izvršavanje instrukcije. U koraku $step_{C0}$ se, u zavisnosti od toga da li signal **prekid** bloka *intr* ima vrednost 0 ili 1, ili završava izvršavanje tekuće instrukcije i prelaskom na korak $step_{00}$ započinje faza čitanje instrukcije sledeće instrukcije ili se produžava izvršavanje tekuće instrukcije i prelaskom na korak $step_{C1}$ produžava faza opsluživanje prekida tekuće instrukcije. !

$step_{C0}$ br (*if* $\overline{\textbf{prekid}}$ *then* $step_{00}$);

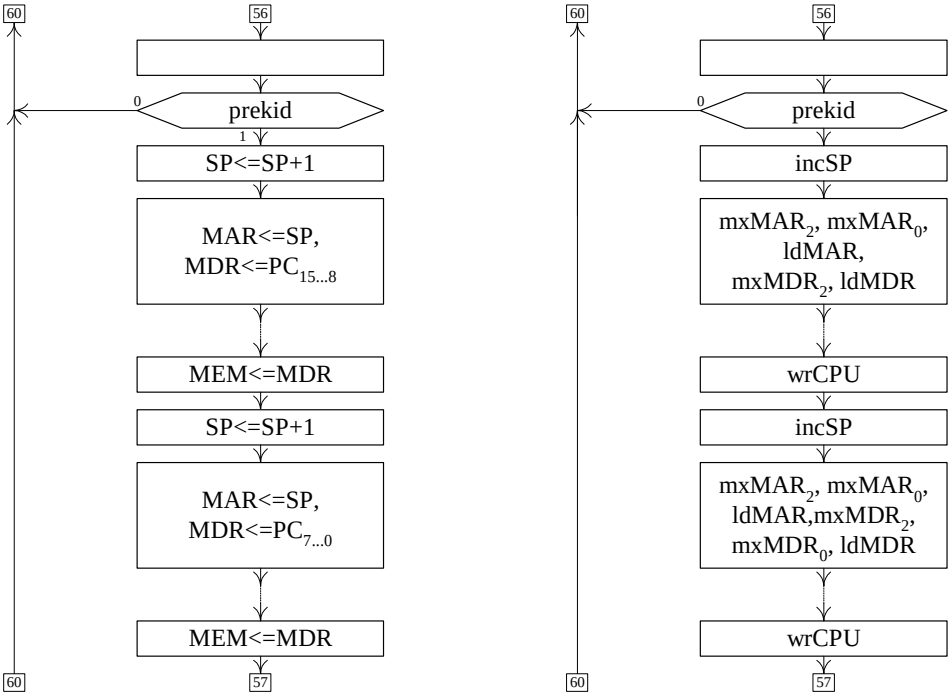
! Opsluživanje prekida se sastoji iz tri grupe koraka u kojima se realizuje čuvanje konteksta procesora, utvrđivanje broja ulaza i utvrđivanje adrese prekidne rutine. !

! Čuvanje konteksta procesora !

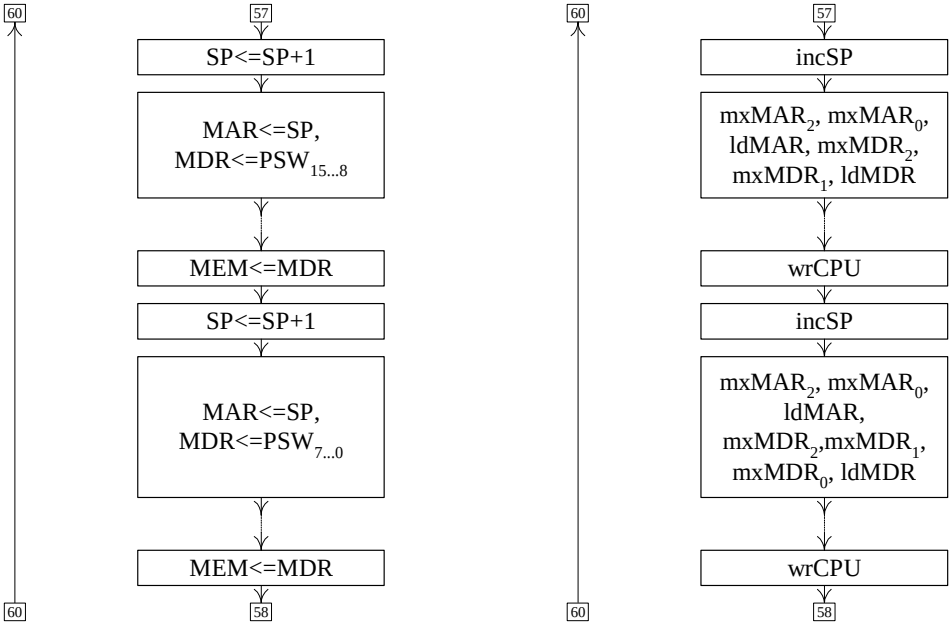
! Kontekst procesora i to $PC_{15...0}$ i $PSW_{15...0}$ se čuva u koracima $step_{C1}$ do $step_{D0}$. U koracima $step_{C1}$ do $step_{C8}$ se na stek stavlja programski brojač $PC_{15...0}$. Na stek se stavlja prvo viši a zatim i niži bajt registra $PC_{15...0}$. Stoga se najpre u koraku $step_{C1}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$ bloka *addr*. Zatim se u koraku $step_{C2}$ vrednošću 1 signala **ldMAR**, sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$ i vrednošću 1 signala **ldMDR** sadržaj višeg bajta registra $PC_{15...8}$ upisuje u registar $MDR_{7...0}$. Upis se realizuje u koracima $step_{C3}$ i $step_{C4}$. Potom se u koraku $step_{C5}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{C6}$ vrednošću 1 signala **ldMAR**, sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$ i vrednošću 1 signala **ldMDR** sadržaj nižeg bajta registra $PC_{7...0}$ upisuje u registar $MDR_{7...0}$. Upis se realizuje u koracima $step_{C7}$ i $step_{C8}$.!

$step_{C1}$ **incSP**;
$step_{C2}$ **..., ldMAR**, **..., ldMDR**;

$step_{C3}$ ...
$step_{C4}$ **wrCPU**;
$step_{C5}$ **incSP**;
$step_{C6}$ **..., ldMAR**, **..., ldMDR**;
$step_{C7}$ ...
$step_{C8}$ **wrCPU**;



Slika 15 Opsluživanje prekida (prvi deo)



Slika 16 Opsluživanje prekida (drugi deo)

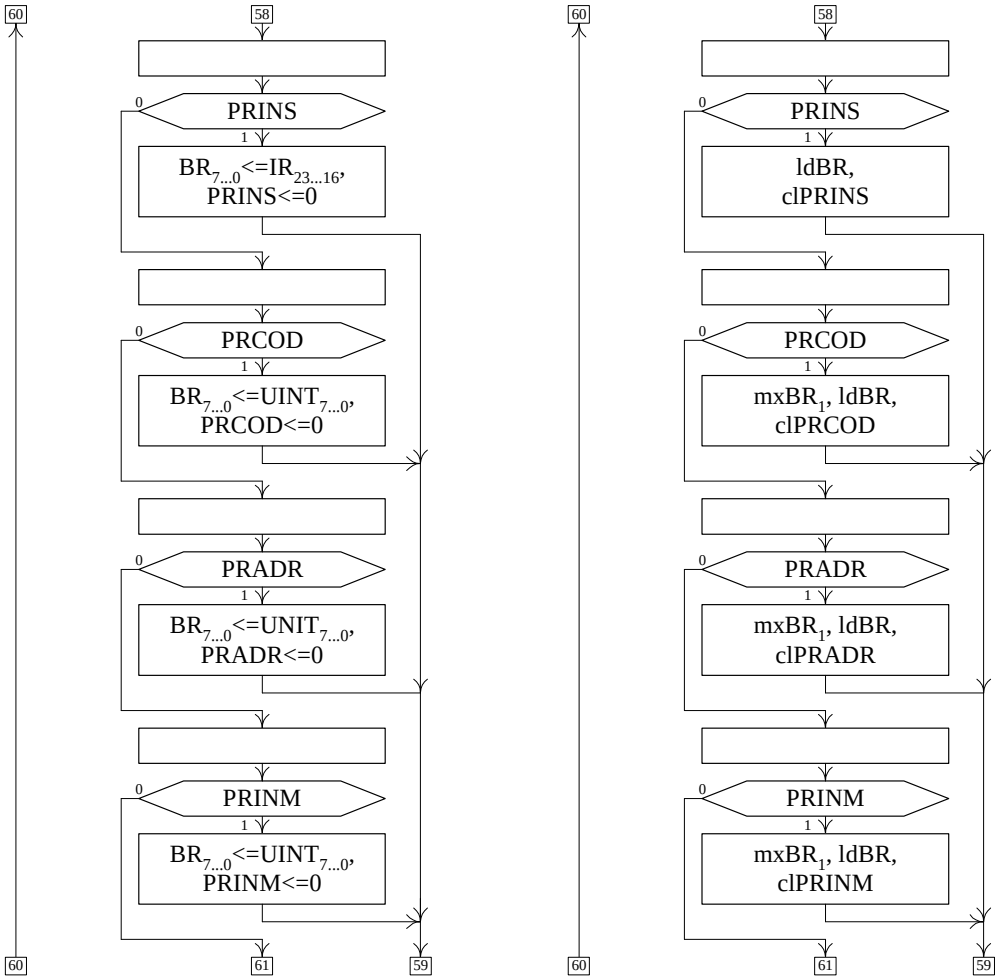
! U koracima $step_{C9}$ do $step_{D0}$ se na stek stavlja programska statusna reč $PSW_{15...0}$ bloka *exec*. Na stek se stavlja prvo viši a zatim i niži bajt registra $PSW_{15...0}$. Stoga se najpre u koraku $step_{C9}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$ bloka *addr*. Zatim se u koraku $step_{CA}$ vrednošću 1 signala **ldMAR**, sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$ i vrednošću 1 signala **ldMDR**

sadržaj višeg bajta registra $PSW_{15...8}$ upisuje u registar $MDR_{7...0}$. Upis se realizuje u koracima $step_{CB}$ i $step_{CC}$. Potom se u koraku $step_{CD}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{CE}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$ i vrednošću 1 signala **ldMDR** sadržaj nižeg bajta registra $PSW_{7...0}$ upisuje u registar $MDR_{7...0}$. Upis se realizuje u koracima $step_{CF}$ i $step_{D0}$. !

$step_{C9}$ **incSP**;
$step_{CA}$ **..., ldMAR**, **..., ldMDR**;
$step_{CB}$ **...**
$step_{CC}$ **wrCPU**;
$step_{CD}$ **incSP**;
$step_{CE}$ **..., ldMAR**, **..., ldMDR**;
$step_{CF}$ **...**
$step_{D0}$ **wrCPU**;

! Utvrđivanje broja ulaza !

! U korak $step_{D1}$ se dolazi iz $step_{D0}$. U koracima $step_{D1}$ do $step_{DB}$ se utvrđuje broj ulaza u tabelu sa adresama prekidnih rutina i upisuje u registar $BR_{7...0}$ bloka *intr*. U ovim koracima se po opadajućim prioritetima utvrđuje zahtev za prekid najvišeg prioriteta i za njega određuje broj ulaza u tabelu sa adresama prekidnih rutina. !



Slika 17 Opsluživanje prekida (treći deo)

! Provera da li postoji zahtev za prekid zbog izvršavanja instrukcije prekida INT. !

! U koraku $step_{D1}$ se vrši provera da li signal **PRINS** bloka *intr* ima vrednost 1 ili 0 i prelazi na korak $step_{D2}$ ili $step_{D3}$, respektivno. Ukoliko signal **PRINS** ima vrednost 1 jer postoji ovaj zahtev za prekid,

u koraku $step_{D2}$ se vrednostima 0 signala **mxBR$_1$** i **mxBR$_0$** bloka *intr* sadržaj razreda $IR_{23...16}$, koji sadrži broj ulaza, propušta kroz multiplekser MX bloka *intr* i vrednošću 1 signala **ldBR** upisuje u registar $BR_{7...0}$. Istovremeno se vrednošću 1 signala **clPRINS** bloka *intr* se flip-flop PRINS postavlja na vrednost 0. Iz koraka $step_{D2}$ se prelazi na korak $step_{EC}$ radi utvrđivanja adrese prekidne rutine. !

$step_{D1}$ *br* (*if* $\overline{\textbf{PRINS}}$ *then* $step_{D3}$);
$step_{D2}$ **ldBR**, **clPRINS**,
br $step_{EC}$;

! Provera da li postoji zahtev za prekid zbog greške u kodu operacije. !

! U koraku $step_{D3}$ se vrši provera da li signal **PRCOD** bloka *intr* ima vrednost 1 ili 0 i prelazi na korak $step_{D4}$ ili $step_{D5}$, respektivno. Ukoliko signal **PRCOD** ima vrednost 1 jer postoji ovaj zahtev za prekid, u koraku $step_{D4}$ se vrednošću 1 signala **mxBR$_1$** bloka *intr* sadržaj $UINT_{7...0}$ sa izlaza kodera CD2, koji sadrži broj ulaza, propušta kroz multiplekser MX bloka *intr* i vrednošću 1 signala **ldBR** upisuje u registar $BR_{7...0}$. Istovremeno se vrednošću 1 signala **clPRCOD** bloka *intr* flip-flop PRCOD postavlja na vrednost 0. Iz koraka $step_{D4}$ se prelazi na korak $step_{EC}$ radi utvrđivanja adrese prekidne rutine. !

$step_{D3}$ *br* (*if* $\overline{\textbf{PRCOD}}$ *then* $step_{D5}$);
$step_{D4}$ **mxBR$_1$**, **ldBR**, **clPRCOD**,
br $step_{EC}$;

! Provera da li postoji zahtev za prekid zbog greške u adresiranju. !

! U koraku $step_{D5}$ se vrši provera da li signal **PRADR** bloka *intr* ima vrednost 1 ili 0 i prelazi na korak $step_{D6}$ ili $step_{D7}$, respektivno. Ukoliko signal **PRADR** ima vrednost 1 jer postoji ovaj zahtev za prekid, u koraku $step_{D6}$ se vrednošću 1 signala **mxBR$_1$** bloka *intr* sadržaj $UINT_{7...0}$ sa izlaza kodera CD2, koji sadrži broj ulaza, propušta kroz multiplekser MX bloka *intr* i vrednošću 1 signala **ldBR** upisuje u registar $BR_{7...0}$. Istovremeno se vrednošću 1 signala **clPRADR** bloka *intr* flip-flop PRADR postavlja na vrednost 0. Iz koraka $step_{D6}$ se prelazi na korak $step_{EC}$ radi utvrđivanja adrese prekidne rutine. !

$step_{D5}$ *br* (*if* $\overline{\textbf{PRADR}}$ *then* $step_{D7}$);
$step_{D6}$ **mxBR$_1$**, **ldBR**, **clPRADR**,
br $step_{EC}$;

! Provera da li postoji spoljašnji nemaskirajući zahtev za prekid. !

! U koraku $step_{D7}$ se vrši provera da li signal **PRINM** bloka *intr* ima vrednost 1 ili 0 i prelazi na korak $step_{D8}$ ili $step_{D9}$, respektivno. Ukoliko signal **PRINM** ima vrednost 1 jer postoji ovaj zahtev za prekid, u koraku $step_{D8}$ se vrednošću 1 signala **mxBR$_1$** bloka *intr* sadržaj $UINT_{7...0}$ sa izlaza kodera CD2, koji sadrži broj ulaza, propušta kroz multiplekser MX bloka *intr* i vrednošću 1 signala **ldBR** upisuje u registar $BR_{7...0}$. Istovremeno se vrednošću 1 signala **clPRINM** bloka *intr* flip-flop PRINM postavlja na vrednost 0. Iz koraka $step_{D8}$ se prelazi na korak $step_{EC}$ radi utvrđivanja adrese prekidne rutine. !

$step_{D7}$ *br* (*if* $\overline{\textbf{PRINM}}$ *then* $step_{D9}$);
$step_{D8}$ **mxBR$_1$**, **ldBR**, **clPRINM**,
br $step_{EC}$;

! Provera da li postoji spoljašnji maskirajući zahtev za prekid. !

! U koraku $step_{D9}$ se vrši provera da li signal **printr** bloka *intr* ima vrednost 1 ili 0 i prelazi na korak $step_{DA}$ ili $step_{EB}$, respektivno..!

$step_{D9}$ *br* (*if* $\overline{\textbf{printr}}$ *then* $step_{EB}$);

! U koraku $step_{DA}$ se vrši provera da li signal PSWP bloka *exec* ima vrednost 1 ili 0 i prelazi na korak $step_{DC}$ ili $step_{DB}$, respektivno.!

$step_{DA}$ *br* (*if* PSWP *then* $step_{DC}$);

! Fiksni ulazi !

Ukoliko signal PSWP ima vrednost 0 jer se fiksno određuju brojevi ulaza u koraku $step_{DB}$ se vrednošću 1 signala **mxBR$_0$** bloka *intr* sadržaj $UEXT_{7...0}$ sa izlaza kodera CD3, koji sadrži broj ulaza, propušta kroz multiplekser MX bloka *intr* i vrednošću 1 signala **ldBR** upisuje u registar $BR_{7...0}$.!

$step_{DB}$ **mxBR$_0$**, **ldBR**,
br $step_{DF}$;

! Promenljivi ulazi !

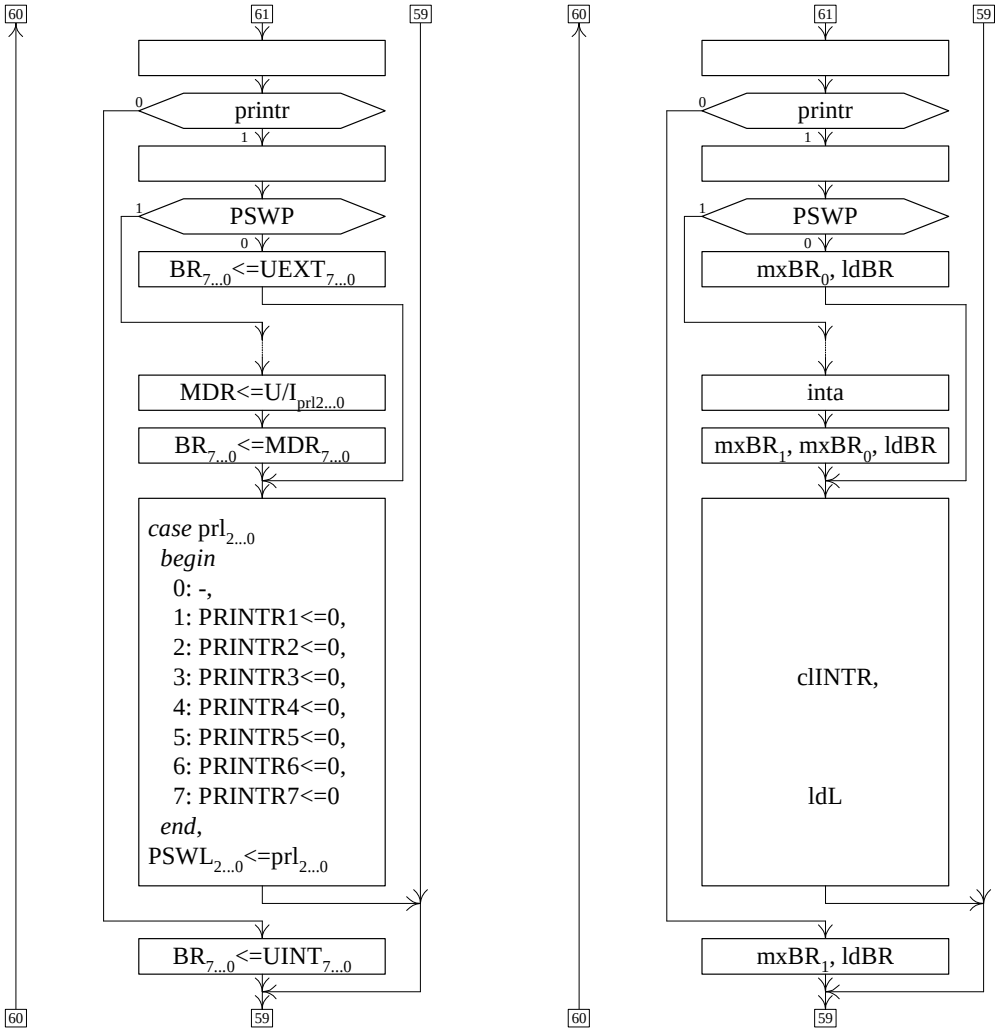
Ukoliko signal PSWP ima vrednost 1, jer kontroleri periferija šalju brojeve ulaza, u koraku $step_{DD}$ se kontroleru periferiji najvišeg prioriteta šalje signal **inta** na koji ona po linijama podataka magistrale šalje broj ulaza. Procesor prihvata broj ulaza u registru $MDR_{7...0}$. Vrednostima 1 signala **$mxBR_1$** i **$mxBR_0$** bloka *intr* sadržaj registra $MDR_{7...0}$, koji sadrži broj ulaza, se propušta kroz multiplekser MX bloka *intr* i vrednošću 1 signala **ldBR** upisuje u registar $BR_{7...0}$.!

$step_{DC}$ ...
$step_{DD}$ **inta**;
$step_{DE}$ **$mxBR_1$**, **$mxBR_0$**, **ldBR**;

! Brisanje zahteva i pamćenje nivoa prioriteta !

! Vrednošću 1 signala **clINTR** bloka *intr* jedan od flip-flopova $PRINTR_7$ do $PRINTR_1$ postavlja na vrednost 0. Flip-flop $PRINTR_7$ do $PRINTR_1$ koji se postavlja na vrednost 0 je selektovan binarnim vrednostima 7 do 1, respektivno, signala **prl_2** do **prl_0** bloka *intr* i odgovara liniji najvišeg nivoa prioriteta po kojoj je stigao spoljašnji maskirajući zahtev za prekid koji nije selektivno maskiran odgovarajućim razredom registra maske IMR. Pored toga vrednošću 1 signala **ldL** bloka *exec* se signali **prl_2** do **prl_0** , koji predstavljaju binarnu vrednost nivoa prioriteta prekidne rutine na koju se skače, upisuju u razrede $PSWL_2$ do $PSWL_0$ programske statusne reči $PSW_{15...0}$ bloka *exec*. Iz koraka $step_{DA}$ se prelazi na korak $step_{DC}$ radi utvrđivanja adrese prekidne rutine. !

$step_{DF}$ **clINTR**, **ldL**,
br $step_{EC}$;



Slika 18 Opsluživanje prekida (treći deo)

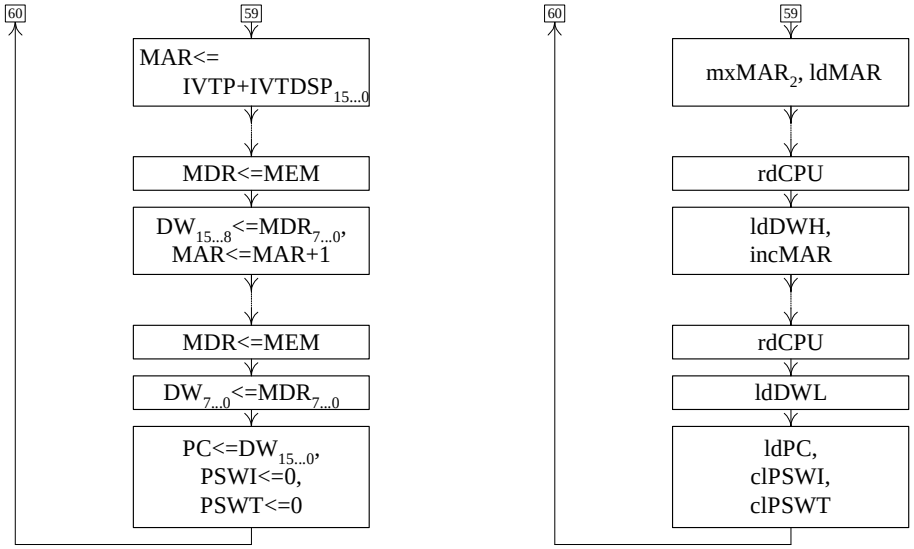
! Prekid posle svake instrukcije !

! U korak $step_{EB}$ se dolazi iz $step_{D9}$ ukoliko signal **printr** ima vrednost 0. Kako se u ovaj korak dolazi jedino ukoliko se proverom signala **prekid** u koraku $step_{C0}$ utvrđuje da postoji barem jedan zahtev za prekid i proverom signala **PRINS, PRCOD, PRADR**, **PRINM** i **printr** u koracima $step_{D1}$, $step_{D3}$, $step_{D5}$, $step_{D7}$ i $step_{D9}$ utvrđuje da ovi signali imaju vrednost 0 i da odgovarajućih zahteva za prekid nema, to znači da postoji zahtev za prekid zbog zadatog režima rada prekid posle svake instrukcije. Stoga se vrednošću 1 signala $\mathbf{mxBR_1}$ bloka *intr* sadržaj $UINT_{7...0}$ sa izlaza kodera CD2, koji sadrži broj ulaza, propušta kroz multiplekser MX i vrednošću 1 signala **ldBR** upisuje u registar $BR_{7...0}$. Iz koraka $step_{DB}$ se prelazi na korak $step_{DC}$ radi utvrđivanja adrese prekidne rutine. !

$step_{EB}$ $\mathbf{mxBR_1}$, **ldBR**;

! Utvrđivanje adrese prekidne rutine !

! U koracima $step_{DC}$ do $step_{E3}$ se na osnovu dobijenog broja ulaza i sadržaja registra koji ukazuje na početnu adresu tabele sa adresama pekidnih rutina, iz odgovarajućeg ulaza čita adresa prekidne rutine i upisuje u programski brojač $PC_{15...0}$ bloka *fetch*. U koraku $step_{DC}$ se na ulaze sabirača ADD propuštaju sadržaj registra $IVTP_{15...0}$ i sadržaj $IVTDSP_{15...0}$ koji predstavlja sadržaj registra $BR_{7...0}$ pomeren ulevo za jedno mesto i proširen nulama do dužine 16 bita. Vrednošću 1 signala **ldMAR** bloka *bus* se sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa memorijske lokacije počev od koje treba pročitati dva bajta koji predstavljaju viši i niži bajt adrese prekidne rutine. Čitanje prvog bajta se realizuje u koracima $step_{DD}$ i $step_{DE}$, a drugog bajta u koracima $step_{E0}$ i $step_{E1}$. U koraku $step_{DF}$ se prvi bajt vrednošću 1 signala **ldDWH** upisuje u viši bajt registra $DW_{15...8}$ bloka *bus*, a vrednošću 1 signala **incMAR** adresni registar $MAR_{15...0}$ inkrementira na adresu sledećeg bajta. U koraku $step_{E2}$ se drugi bajt vrednošću 1 signala **ldDWL** upisuje u niži bajt registra $DW_{7...0}$. Time se u registru $DW_{15...0}$ nalazi adresa prekidne rutine. Na kraju se u koraku $step_{E3}$ sadržaj registra $DW_{15...0}$ vrednošću 1 signala **ldPC** upisuje u registar $PC_{15...0}$. Time se u registru $PC_{15...0}$ nalazi adresa prve instrukcije prekidne rutine. Vrednostima 1 signala **clPSWI** i **clPSWT** se u razrede PSWI i PSWT bloka *exec* upisuju vrednosti 0. Time se u prekidnu rutinu ulazi sa režimom rada u kome su maskirani svi maskirajući prekidi i u kome nema prekida posle svake instrukcije. Iz koraka $step_{E3}$ se bezuslovno prelazi na $step_{00}$. !



Slika 19 Opsluživanje prekida (četvrti deo)

$step_{EC}$ **..., ldMAR**;

$step_{ED}$ ...

$step_{EE}$ **rdCPU**;

$step_{EF}$ **ldDWH**, **incMAR**;

$step_{F0}$ ...

$step_{F1}$ **rdCPU**;

$step_{F2}$ **ldDWL**;

$step_{F3}$ **ldPC**, **clPSWI**, **clPSWT**,
br $step_{00}$;

1.2.2.2 Generisanje signala prekid

Prelazak na neku prekidnu rutinu se realizuje samo kada se u koraku $step_{C0}$ koji je prvi korak faze *opsluživanje zahteva za preki* utvrdi da signal **prekid** ima vrednost 1. Signal **prekid** ima vrednost 1 ukoliko bar jedan od signala **PRINS**, **PRCOD**, **PRADR**, **PRINM**, **printr** ili **PSWT** ima vrednost 1 (slika 2). U daljem tekstu se razmatra u kom trenutku i pod kojim uslovima ovi signali dobijaju vrednosti 1 i 0.

1.2.2.2.1 PRINS

Signal **PRINS** (slika 2) postaje 1 kada se u fazi *izvršavanje operacije* utvrdi da je instrukcija koja se trenutno izvršava instrukcija prekida INT.

U ovom slučaju se u fazi *čitanje instrukcije* čitaju dva bajta instrukcije. Posle prvog bajta se u koraku $step_{05}$ utvrđuje da nema greške u kodu operacije, a posle drugog bajta u koraku $step_{0C}$ da nema greške u načinu adresiranja. U slučaju instrukcije prekida čija je dužina dva bajta signal **l2_brnch** ima vrednost 1 pa se iz koraka $step_{0E}$ prelazi na korak $step_{50}$ koji je prvi korak faze *izvršavanje operacije*. U koraku $step_{50}$ se utvrđuje da signal operacije INT ima vrednost 1 pa se prelazi na korak $step_{A4}$. Faza izvršavanja instrukcije INT se svodi na generisanje vrednosti 1 signala **stPRINS** kojim se upisuje vrednost 1 u flip-flop PRINS i prelazak na korak $step_{C0}$ i fazu *opsluživanje zahteva za prekid*. Upisivanjem vrednosti 1 u flip-flop PRINS i signal **prekid** postaje 1. Signal PRINS postaju 0 pri pojavi vrednosti 1 signala clPRINS koji se generišu u procesoru u koraku $step_{D2}$ faze *opsluživanje zahteva za prekid* kada se broj ulaza za dati prekid dobija i prelazi na izračunavanje adrese prekidne rutine.

1.2.2.2.2 PRCOD

Signal **PRCOD** (slika 2) postaje 1 kada se u fazi *čitanje instrukcije* posle pročitanog prvog bajta instrukcije u koraku $step_{05}$ utvrdi da postoji greška u kodu operacije i pređe na korak $step_{06}$.

Tada se u koraku $step_{06}$ vrednošću 1 signala **stPRCOD** upisuje vrednost 1 u flip-flop **PRCOD** i prelazi na korak $step_{C0}$ i fazu *opsluživanje zahteva za prekid*. Upisivanjem vrednosti 1 u flip-flop PRCOD i signal **prekid** postaje 1.

Signal PRCOD postaju 0 pri pojavi vrednosti 1 signala clPRCOD koji se generišu u procesoru u koraku $step_{D4}$ faze *opsluživanje zahteva za prekid* kada se broj ulaza za dati prekid dobija i prelazi na izračunavanje adrese prekidne rutine.

1.2.2.2.3 PRADR

Signal **PRADR** (slika 2) postaje 1 kada se u fazi *čitanje instrukcije* posle pročitanog drugog bajta instrukcije u koraku $step_{0C}$ utvrdi da postoji greška u načinu adresiranja i pređe na korak $step_{0D}$.

Tada se u koraku $step_{0D}$ vrednošću 1 signala **stPRCOD** upisuje vrednost 1 u flip-flop PRADR i prelazi na korak $step_{C0}$ i fazu *opsluživanje zahteva za prekid*. Upisivanjem vrednosti 1 u flip-flop PRCOD i signal **prekid** postaje 1.

Signal PRADR postaju 0 pri pojavi vrednosti 1 signala clPRCOD koji se generišu u procesoru u koraku $step_{D6}$ faze *opsluživanje zahteva za prekid* kada se broj ulaza za dati prekid dobija i prelazi na izračunavanje adrese prekidne rutine.

Prekidi zbog instrukcije prekida, greške u kodu operacije i greške u načinu adresiranja su unutrašnji prekidi jer ih procesor generiše tokom izvršavanja koraka tekuće instrukcije. Ovi prekidi su međusobno isključivi i tokom izvršavanja koraka tekuće instrukcije samo jedan od ova tri prekida može da se generiše. Prva prilika da se generiše neki od ova tri unutrašnja prekida je u koraku $step_{05}$ kada se proverava da li postoji greška u kodu operacije i ukoliko postoji prelazi na *fazu opsluživanje prekida*. Ako se u koraku $step_{05}$ utvrdi da ne postoji greška u kodu operacije može da se stigne do koraka $step_{0C}$ u kome se proverava da li postoji greška u načinu adresiranja i ukoliko postoji pređe na *fazu opsluživanje prekida*. Jedino ukoliko se prvo u koraku $step_{05}$ utvrdi da ne postoji greška u kodu operacije i zatim u koraku $step_{0C}$ utvrdi da ne postoji greška u načinu adresiranja, može se stići do faze *izvršavanje operacije* tekuće instrukcije. Ukoliko se tada u koraku $step_{50}$ utvrdi da se radi o instrukciji prekida, prelazi se na *fazu opsluživanje prekida*.

1.2.2.2.4 PRINM

Signal **PRINM** (slika 2) postaje 1, čime se generiše spoljašnji nemaskirajući prekid, kada uređaj ***FAULT*** koji kontroliše ispravnost napona napajanja vrednošću 1 signala **inm** upiše vrednost 1 u flip-flop PRINM. Upisivanjem vrednosti 1 u flip-flop PRINM i signal **prekid** postaje 1. Ovaj prekid je spoljašnji prekid jer se generiše izvan procesora i nezavisno od toga u kojoj fazi izvršavanja tekuće instrukcije se procesor nalazi. Tekuća instrukcija tokom čijeg izvršavanja se generiše ovaj prekid može da bude bilo koja instrukcija uključujući i instrukciju za koju se generiše prekid zbog greške u kodu operacije ili greške u načinu adresiranja ili instrukcija prekida. Za razliku od prekida zbog greške u kodu operacije ili greške u načinu adresiranja ili instrukcije prekida kod kojih se odmah po generisanju nekog od ova tri prekida prelazi na korak $step_{C0}$ i fazu *opsluživanje zahteva za prekid* i time reaguje na dati prekid, u slučaju spoljašnjeg nemaskirajućeg prekida na ovaj prekid će procesor moći da reaguje tek kada tekuća instrukcija dođe u korak $step_{C0}$ faze *opsluživanje zahteva za prekid*. U fazi *opsluživanje zahteva za prekid* procesor će da reaguje na spoljašnji nemaskirajući zahtev za prekid ukoliko tokom izvršavanja tekuće instrukcije nije generisan prekid zbog greške u kodu operacije ili načina adresiranja ili zbog instrukcije prekida. Međutim, ukoliko je tokom izvršavanja tekuće instrukcije generisan prekid zbog greške u kodu operacije ili načinu adresiranja ili zbog instrukcije prekida, najpre se skače u jednu od tri prekidne rutine za ove tri vrste prekida, pa se tek iz nje skače u prekidnu rutine za spoljašnji nemaskirajući prekid.

Signal PRINM postaju 0 pri pojavi vrednosti 1 signala clPRINM koji se generišu u procesoru u koraku $step_{D8}$ faze *opsluživanje zahteva za prekid* kada se broj ulaza za dati prekid dobija i prelazi na izračunavanje adrese prekidne rutine.

1.2.2.2.5 printr

Da bi signal **printr** (slika 3) bio 1 potrebno je da,

① je generisan neki od spoljašnjih maskirajućih zahteva za prekida tako što je neki od ulazno/izlaznih uređaja ***U/I1*** do ***U/I7*** vrednošću 1 jednog od signala $\mathbf{intr_1}$ **do** $\mathbf{intr_7}$ (slika 2) upisao vrednost 1 u jedan od flip-flopova $PRINTR_1$ do $PRINTR_7$, respektivno,

② maskirajući zahtev za prekid $PRINTR_1$ do $PRINTR_7$ nije selektivno maskiran jer se u odgovarajućem razredu IMR_1 do IMR_7 registra maske nalazi vrednost 1, pa jedan od signala $\mathbf{irm_1}$ do $\mathbf{irm_7}$ ima vrednost 1, što daje vrednost 1 signala **imrprintr** (slika 3),

③ je nivo prioriteta maskirajućeg zahteva za prekid prl_2 do prl_0 (slika 3) formiran na osnovu vrednosti signal $\mathbf{irm_1}$ do $\mathbf{irm_7}$ viši od nivoa prioriteta tekućeg programa određenog sadržajem razreda $PSWL_2$ do $PSWL_0$ programske statusne reči PSW procesora, što daje vrednost 1 signala **acc** i

④ maskirajući zahtevi za prekid nisu kompletno maskirani jer se u razredu PSWI programske statusne reči PSW procesora nalazi vrednost 1.
Ukoliko bilo koji od ovih uslova nije ispunjen, signal **printr** ima vrednost 0.

Promena vrednosti bilo kog od signala $PRINTR_1$ do $PRINTR_7$, IMR_1 do IMR_7, $PSWL_2$ do $PSWL_0$ i PSWI utiče na to da li će vrednost signala **printr** da bude 1 ili 0. Signali $PRINTR_1$ do $PRINTR_7$ postaju 1 pri pojavi vrednosti 1 signala **$intr_1$ do $intr_7$** koji se generišu izvan procesora i nezavisno od toga u kojoj fazi izvršavanja tekuće instrukcije se procesor nalazi. Signali $PRINTR_1$ do $PRINTR_7$ postaju 0 pri pojavi vrednosti 1 signala $clINTR_1$ do $clINTR_7$ koji se generišu u procesoru u fazi *opsluživanje zahteva za prekid* kada se broj ulaza za dati prekid dobija i prelazi na izračunavanje adrese prekidne rutine. Tada se u koraku $step_{DF}$ generiše vrednost 1 signala clINTR koja na osnovu binarne vrednosti signala prl_2 do prl_0 koja predstavlja nivoa prirititeta prekidne rutine na koju se skače daje vrednost 1 jednog od signala $clINTR_1$ do $clINTR_7$.

Promene vrednosti signala IMR_1 do IMR_7, $PSWL_2$ do $PSWL_0$ i PSWI se ostvaruju u procesoru u fazi *izvršavanje operacije* posebnih instrukcija koje služe da se programskim putem ovi signali postavljaju na odgovarajuće vrednosti. Pored toga promene vrednosti signala $PSWL_2$ do $PSWL_0$ i PSWI se u određenim slučajevima realizuju i hardverskim putem u fazi *opsluživanje zahteva za prekid* svih instrukcija.

Promene vrednosti signala IMR_1 do IMR_7 se ostvaruju u procesoru u fazi *izvršavanje operacije* instrukcije STIMR. Instrukcija STIMR je bezadresna instrukcija prenosa koja u koraku $step_{8A}$ prebacuje sadržaj registra akumulatora $AW_{15...0}$ u registar maske $IMR_{15...0}$. Pretpostavlja se da je pre instrukcije STIMR instrukcijom prenosa LDW u registar $AW_{15...0}$ upisana vrednost koja se instrukcijiom STIMR prebacuje iz registra akumulatora $AW_{15...0}$ u registar maske $IMR_{15...0}$.

Promene vrednosti signala $PSWL_2$ do $PSWL_0$ se realizuju hardverskim putem u koraku $step_{DF}$ faze *opsluživanje zahteva za prekid* svih instrukcija ukoliko se prihvata neki od maskirajućih zahteva za prekid tako što se vrednošću 1 signala **ldL** u razrede $PSWL_2$ do $PSWL_0$ upisuju vrednosti prl_2 do prl_0 nivoa prirititeta prekidne rutine na koju se skače. Pre toga su vrednost $PSWL_2$ do $PSWL_0$ nivoa prioriteta programa čije se izvršavanje prekida stavljene na stek sa ostalim razredima registra PSW. Promene vrednosti signala $PSWL_2$ do $PSWL_0$ se ostvaruju u procesoru programskim putem u koraku $step_{B1}$ faze *izvršavanje operacije* instrukcije RTI. Tada se vrednošću sa steka restauriraju vrednosti signala $PSWL_2$ do $PSWL_0$ na nivo prioriteta programa u koji se vraća.

Promene vrednosti signala PSWI se realizuju hardverskim putem u koraku $step_{F3}$ faze *opsluživanje zahteva za prekid* svih instrukcija tako što se u razred PSWI upisuje vrednost 0. Pre toga je vrednost PSWI programa čije se izvršavanje prekida stavljena na stek sa ostalim razredima registra PSW. Promene vrednosti signala PSWI se ostvaruje u procesoru programskim putem u koraku $step_{B5}$ faze *izvršavanje operacije* instrukcije RTI. Tada se vrednošću sa steka restaurira vrednost signala PSWI na vrednost programa u koji se vraća. Pored toga promene vrednosti signala PSWI se ostvaruje u procesoru programskim putem u koraku $step_{54}$ faze *izvršavanje operacije* instrukcije INTE kada se u PSWI upisuje vrednost 1 i u koraku $step_{53}$ faze *izvršavanje operacije* instrukcije INTD kada se u PSWI upisuje vrednost 0.

1.2.2.2.6 PSWT

Promene vrednosti signala PSWT (slika 2) se realizuju hardverskim putem u koraku $step_{F3}$ faze *opsluživanje zahteva za prekid* svih instrukcija tako što se u razred PSWT upisuje

vrednost 0. Pre toga je vrednost PSWT programa čije se izvršavanje prekida stavljena na stek sa ostalim razredima registra PSW. Promene vrednosti signala PSWT se ostvaruje u procesoru programskim putem u koraku $step_{B5}$ faze *izvršavanje operacije* instrukcije RTI. Tada se vrednošću sa steka restaurira vrednost signala PSWT na vrednost programa u koji se vraća. Pored toga promene vrednosti signala PSWT se ostvaruje u procesoru programskim putem u koraku $step_{56}$ faze *izvršavanje operacije* instrukcije TRPE kada se u PSWT upisuje vrednost 1 i u koraku $step_{55}$ faze *izvršavanje operacije* instrukcije TRPD kada se u PSWT upisuje vrednost 0.