JOVAN ĐORĐEVIĆ

ARHITEKTURA I ORGANIZACIJA RAČUNARA

Beograd 2006.

SADRŽAJ

1 PROCESOR

U ovoj glavi se razmatra jedan pristup organizacije procesora po kome se procesor sastoji iz dve jedinice i to operacione jedinice i upravljačke jedinice. Operaciona jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova upravljačke jedinice. Upravljačka jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala operacione jedinice na osnovu algoritama operacija i signala logičkih uslova. Najpre se daju osnovni elementi arhitekture i organizacije procesora koji se koristi za ilustraciju razmatranih pristupa realizacije operacione jedinice i upravljačke jedinice, a zatim posebno razmatraju neki pristupi realizacije operacione jedinice i upravljačke jedinice.

1.1 ARHITEKTURA I ORGANIZACIJA PROCESORA

U ovom odeljku se daju arhitektura i neki elementi organizacije procesora koji se koristi za ilustraciju razmatranih pristupa realizacije operacione jedinice i upravljačke jedinice.

Procesor je sa jednoadresnim formatom instrukcija. Dužina instrukcije je promenljiva i kreće se od 1, 2 do N memorijskih reči. Prva reč uvek specificira kod operacije. Bezadresne instrukcije su instrukcija povratka iz potprograma (RTS) i povratka iz prekidne rutine (RTI). Dužina instrukcija je jedna memorijska reč. Instrukcije skoka su instrukcija uslovnog skoka ukoliko je rezultat nula (JZ), bezuslovnog skoka (JMP) i skoka na potprograma (JSR). Kod ovih instrukcija druga i sledećih nekoliko reči specificiraju adresu skoka. Adresne instrukcije su instrukcija prenosa u akumulator (LOAD), instrukcija prenosa iz akumulaatora (STORE), aritmetička instrukcija sabiranja (ADD), logička instrukcija logički proizvod (AND) i instrukcija aritmetičkog pomeranja udesno za jedno mesto (ASR). Kod ovih instrukcija druga reč specificira način adresiranja i adresu registra opšte namene. Kod registarskih adresiranja dužina instrukcije je dve reči. Kod memorijskih adresiranja treća i sledećih nekoliko reči specificiraju adresu, pomeraj ili neposrednu veličinu. Podaci su celobrojne veličine bez znaka dužine koja odgovara širina memorijske reči.

Načini adresiranja su specificirani sa nekoliko bitova druge reči instrukcije i to: registarsko direktno adresiranje, registarsko indirektno adresiranje, registarsko indirektno adresiranje sa postdekrementiranjem, registarsko indirektno adresiranje sa preinkrementiranjem, memorijsko direktno adresiranje, memorijsko indirektno adresiranje, registarsko indirektno sa pomerajem i neposredno adresiranje. Kod adresiranja koja koriste neki od registara opšte namene R adresa registra se specificira preostalim bitovima druge reči instrukcije. Kod registarskog direktnog adresiranja, registarskog indirektnog adresiranja, registarskog indirektnog adresiranja sa postdekrementiranjem i registarsko indirektno adresiranje sa preinkrementiranjem dužina instrukcije je dve reči. Kod memorijskog direktnog i memorijskog indirektnog adresiranja treća i preostale reči instrukcije sadrže adresu memorijske lokacije. Bitovi druge reči koji sadrže adresu registra opšte namene se ne koriste. Kod registarskog indirektnog adresiranja sa pomerajem treća i preostale reči instrukcije sadrže sadrže pomeraj. Bitovi druge reči koji sadrže adresu registra opšte namene se koriste. Kod neposrednog adresiranja treća i preostale reči instrukcije sadrže podatak. Bitovi druge reči koji sadrže adresu registra opšte namene se ne koriste.

U procesoru postoji programski brojač PC, adresni registar memorije MAR, prihvatni registar podatka memorije MBR, prihvatni registar instrukcije IR dužine N memorijskih reči,

akumulator A, prihvatni registar podatka B, registri opšte name R, programska statusna reč PSW, ukazivač na vrh steka SP, registar broja ulaza u tabelu sa adresam prekidnih rutina BR i ukazivač na tabelu sa adresama prekidnih rutina IVTP. Svi registri sam registra IR su dužine koja odgovara širina memorijske reči, dok je registar IR dužine 1, 2 do N memorijskih reči.

Stek raste prema nižim memorijskim lokacijama, a registar SP ukazuje na prvu slobodnu memorijsku lokaciju.

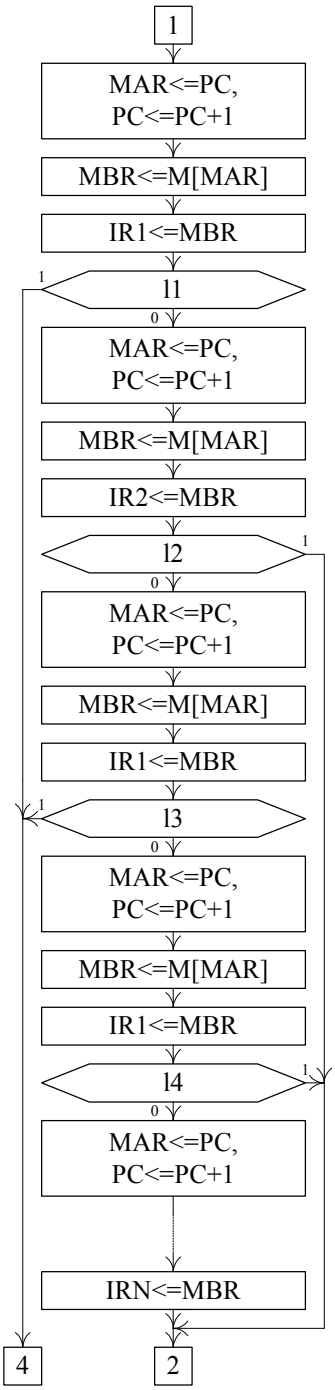
Zahtevi za prekid dolaze od 4 ulazno/izlazna uređaja po linijama označenim od 0 do 3. Po liniji 0 stiže zahtev za prekid najnižeg, a po liniji 3 najvišeg prioriteta. Broj linije najvišeg prioriteta po kojoj je stigao zahtev za prekid nalazi se u binarnom obliku u registru BR dužine 2 razreda. Adrese prekidnih rutina 4 ulazno/izlazna uređaja koji po linijama označenim od 0 do 3 šalju zahteve za prekid nalaze se u ulazima 0 do 3 tabele sa adresama prekidnih rutina. Sadržaj registra BR predstavlja broj ulaza u tabelu sa adresam pekidnih rutina. Početna adresa tabele sa adresama prekidnih rutina se nalazi u registru IVTP. U okviru hardverskog dela opsluživanja zahteva za prekid na stek sa stavljaju samo registri PC i PSW.

Najpre treba nacrtati dijagram toka faza izvršavanja instrukcije i to: faze čitanja instrukcije, faze formiranja adrese i čitanja operanda, faza izvršavanja operacija LOAD, STORE, ADD, AND, ASR, JZ, JMP, JSR, RTS i RTI i faze opsluživanja zahteva za prekid. Dijagram toka izvršavanja instrukcije je dat na slikama 1.a, 1.b, 1.c i 1.d. Izvršavanje instrukcije se sastoji iz četiri faze: čitanje instrukcije (slika 1.a), formiranje adrese i čitanje operanda (slika 1.b), izvršavanje operacija (slika 1.c) i opsluživanje prekida (slika 1.d).

čitanje instrukcije (slika 1.*a)*

Instrukcija se čita iz memorije počev od adrese na koju ukazuju trenutka vrednost programskog brojača PC. Čita se reč po reč i posle svake pročitane reči vrednost PC se inkrementira. Pročitane reči instrukcije se smeštaju u registre IR1, IR2 do IRN koji čine prihvatni registar instrukcije IR. Broj pročitanih reči zavisi od instrukcije. Bezadresne instrukcije sadrže samo kod operacije, pa zato treba pročitati samo jednu reč. Ove instrukcije ne prolaze kroz fazu *formiranje adrese i čitanje operanda*, pa zato posle čitanja jedne reči treba odmah preći na fazu *izvršavanje operacija*. Instrukcije skoka sadrže kod operacije i adresu skoka, pa zato treba pročitati nekoliko reči i to prvu reč koja sadrži kod operacije i drugu i nekoliko sledećih nekoliko reči specificiraju adresu skoka. Ove instrukcije, takođe, ne prolaze kroz fazu *formiranje adrese i čitanje operanda*, pa zato posle čitanja odgovarajućeg broja reči treba odmah preći na fazu *izvršavanje operacija*. Adresne instrukcije obavezno sadrže dve reči i to prvu reč koja specificira kod operacije i drugu reč koja specificira način adresiranja i adresu registra opšte namene. Kod registarskih adresiranja dužina instrukcije je dve reči, dok kod memorijskih adresiranja treća i sledećih nekoliko reči specificiraju adresu, pomeraj ili neposrednu veličinu. Zbog toga kod ovih instrukcija posle čitanja dve reči kod registarskih adresiranja i treće i nekoliko sledećih reči kod memorijskih adresiranja treba preći na fazu *formiranje adrese i čitanje operanda*. Prilikom čitanja reči instrukcije formiraju se signali dužine instrukcije l1, l2 do lN, koji predstavljaju signale logičkih uslova. Ovi signali označavaju da je dužina instrukcije jedna, dve do N reči i samo jedan od njih ima aktivnu vrednost. U slučaju bezadresnih instrukcija posle čitanja prve reči koja sadrži kod operacije treba da se formira aktivna vrednost signala l1. U slučaju instrukcija skoka treba posle čitanja prve reči koja sadrži kod operacije da se formira aktivna vrednost signala jednog od signala dužine instrukcije. Ako je, na primer, dužina instrukcije skoka tri reči signal l3 treba da bude aktivan a ostali signali dužina instrukcije neaktivni. U slučaju adresnih instrukcija treba posle čitanja prve reči koja sadrži kod operacije da se formira neaktivna vrednost signala l1. Na osnovu neaktivne vrednosti signala l1 treba da se pročita druga reč instrukcije koja sadrži specifikaciju način adresiranja i adresu registra opšte namene. Na osnovu specifikacije načina

adresiranja treba da se formira aktivna vrednost signal l2 za registarska adresiranja i neaktivna vrednost signala l2 i aktivna vrednost jednog od signala dužine instrukcije za memorijska adresiranja. Ako je, na primer, dužina adresnih instrukcija sa memorijskim adresiranjima četiri reči, signal l4 treba da bude aktivan a ostali signali dužina instrukcije neaktivni.



Slika 1.a Dijagram toka – faza čitanje instrukcije

Obavezno se čita prva reč instrukcije koja specificira kod operacije. U okviru toga se, najpre, PC prebacuje u MAR i inkrementira sadržaj PC. Iz memorije M se, zatim, sa adrese određene sadržajem registra MAR čita reč i upisuje u registar MBR. Sadržaj registra MBR se, na kraju, prebacuje u registar IR1.

Sada se vrši provera signala logičkog uslova l1. Ukoliko se radi o bezadresnoj instrukciji signal l1 je aktivan, pa se prelazi na korak 4 i fazu *izvršavanje operacije* (slika 1.c). Ukoliko se radi o instrukcijama skoka ili adresnim instrukcijama signal l1 je neaktivan, pa se prelazi na čitanje druge reči instrukcije. Čitanje druge reči instrukcije se realizuje na sličan način kao i čitanje prve reči instrukcije. Druga reč instrukcije se upisuje u registar IR2.

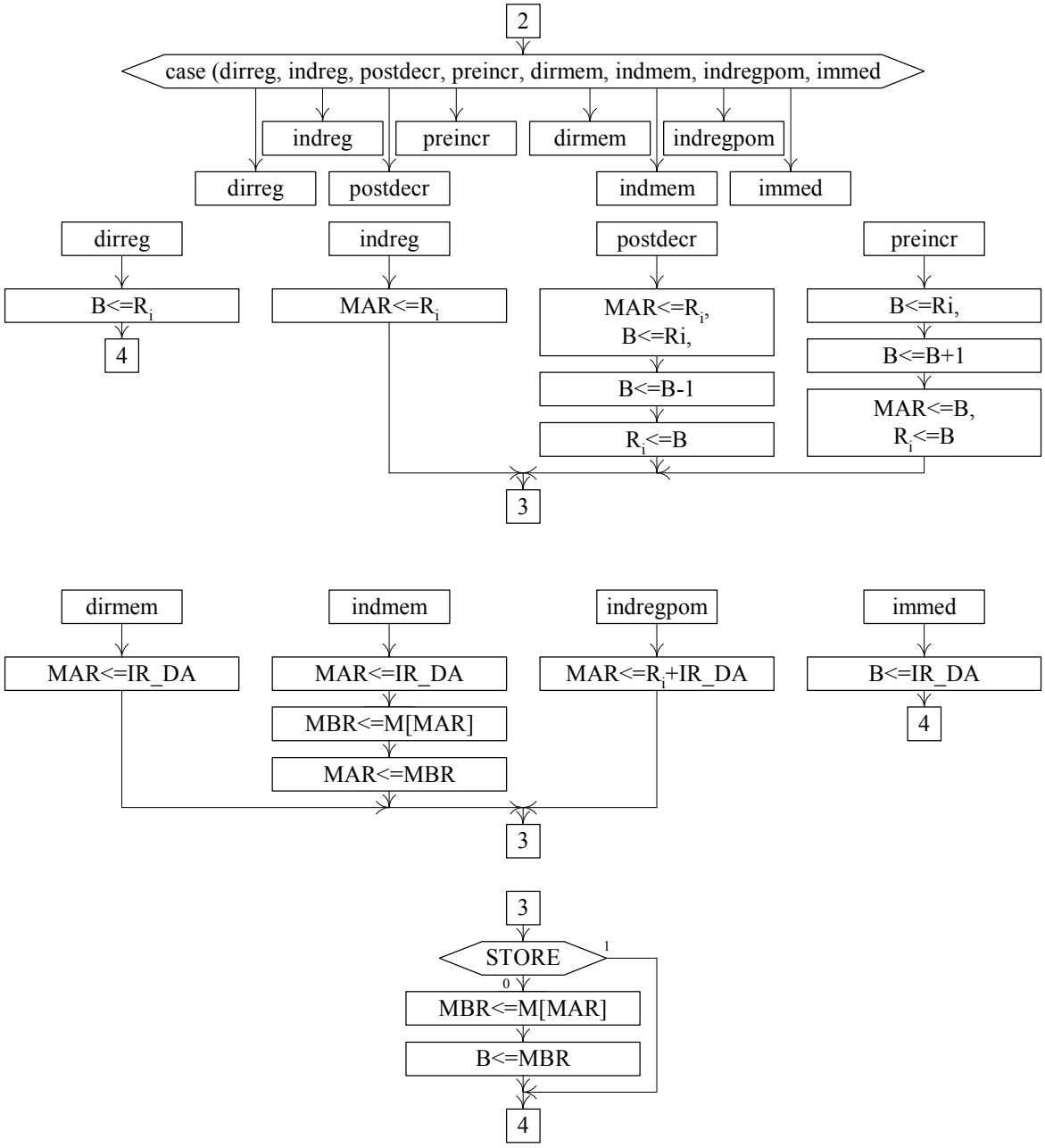
Sada se vrši provera signala logičkog uslova l2. Ukoliko se radi o adresnoj instrukciji sa nekim od registarskih adresiranja, signal l2 je aktivan, pa se prelazi na korak 2 i fazu *formiranje adrese i čitanje operanda* (slika 1.b). Ukoliko se radi o instrukciji skoka čija je dužina, na primer, tri reči ili o adresnoj instrukciji sa nekim od memorijskih adresiranja čija je dužina, na primer, četiri reči, signal l2 je neaktivan, pa se prelazi na čitanje treće reči instrukcije. Čitanje treće reči instrukcije se realizuje na sličan način kao i čitanje prve i druge reči instrukcije. Treća reč instrukcije se upisuje u registar IR3.

Sada se vrši provera signala logičkog uslova l3. Ukoliko se radi o instrukciji skoka čija je dužina, na primer, tri reči, signal l3 je aktivan, pa se prelazi na korak 4 i fazu *izvršavanje operacije* (slika 1.c). Ukoliko se radi o adresnoj instrukciji sa nekim od memorijskih adresiranja čija je dužina, na primer, četiri reči, signal l3 je neaktivan, pa se prelazi na čitanje četvrte reči instrukcije. Čitanje četvrte reči instrukcije se realizuje na sličan način kao i čitanje prve, druge ili treće reči instrukcije. Četvrta reč instrukcije se upisuje u registar IR4.

Sada se vrši provera signala logičkog uslova l4. Ukoliko se radi o adresnoj instrukciji sa nekim od memorijskih adresiranja čija je dužina, na primer, četiri reči, signal l4 je aktivan, pa se prelazi na korak korak 2 i fazu *formiranje adrese i čitanje operanda* (slika 1.b). U slučaju da se radi o instrukcijama čija je dužina veća od četiri reči, na sličan način bi se pročitale i upisale u registre IR5 do IRN i preostale reči instrukcije.

formiranje adrese i čitanje operanda (slika 1.*b)*

Formiranje adrese i čitanje operanda se realizuje samo za adresne instrukcije i to od koraka 2 po posebnom algoritmu za svaki od načina adresiranja prolaskom kroz odgovarajuće korake. Na početku se realizuje višestruki uslovni skok na jedan od dijagrama toka na osnovu toga koji je od signala logičkih uslova načina adresiranja aktivan. Aktivna vrednost jednog od signala načina adresiranja dirreg, indreg, postdecr, preincr, dirmem, indmem, indregpom i immed određuje da je specificirano registarsko direktno adresiranje, registarsko indirektno adresiranje, registarsko indirektno adresiranje sa postdekrementiranjem, registarsko indirektno adresiranje sa preinkrementiranjem, memorijsko direktno adresiranje, memorijsko indirektno adresiranje, registarsko indirektno sa pomerajem i neposredno adresiranje, respektivno.



Slika 1.b Dijagram toka – faza formiranje adrese i čitanje operanda

Ukoliko je signal dirreg aktivan, radi se o registarskom direktnom adresiranju. Operand je tada u registru opšte namene R_i određenom vrednošću druge grupe bitova druge reči instrukcije iz registra IR2. Selektovani registar opšte namene R_i se prebacuje u registar B.

Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 4 i fazu *izvršavanje operacija* (slika 1.c).

Ukoliko je signal indreg aktivan, radi se o registarskom indirektnom adresiranju. Operand je tada u memoriji na adresi koja se nalazi u registru opšte namene R_i određenom vrednošću druge grupe bitova druge reči instrukcije iz registra IR2. Selektovani registar opšte namene R_i se prebacuje u registar registar MAR i prelazi se na korak 3.

Počev od koraka 3 se za sve operacije, sem operacije STORE, čita operand i smešta u registar B. Zbog toga se ovde vrši provera signala operacije STORE. Ukoliko je njegova vrednost 0, iz memorije M se sa adrese određene sadržajem registra MAR čita reč i upisuje u registar MBR, a zatim se sadržaj registra MBR prebacuje u registar B. Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 4 i fazu *izvršavanje operacija* (slika 1.c). Ukoliko je vrednost signala STORE 0, odmah se prelazi na korak 4 i fazu *izvršavanje operacija* (slika 1.c).

Ukoliko je signal postdecr ili preincr aktivan, radi se o registarskom indirektnom adresiranju sa postdekrementiranjem ili registarskom indirektnom adresiranju sa preinkrementiranjem, respektivno. U oba slučaja je, kao i kod registarskog indirektnog adresiranja, operand u memoriji na adresi koja se nalazi u registru opšte namene R_i određenom vrednošću druge grupe bitova druge reči instrukcije iz registra IR2. Selektovani registar opšte namene R_i se prebacuje u registar MAR i prelazi se na korak 3 počev od koga se, na već opisani način, za sve operacije, sem operacije STORE, čita operand i smešta u registar B. Razlika u odnosu na registarsko indirektno adresiranje je samo u tome da se kod registarskog indirektnog adresiranja sa postdekrementiranjem sadržaj registra R_i najpre prebaci u registar MAR a zatim se dekrementiranjem umanji za 1, dok se kod registarskog indirektnog adresiranja sa preinkrementiranjem sadržaj registra R_i najpre inkrementiranjem uveća za 1 a zatim prebaci u registar MAR. Sadržaj registra R_i se umanjuje ili uvećava za 1, jer se to čini za dužinu operanda, čija je dužina, izražena u broju memorijskih reči, jedna reč. Treba uočiti da je uzeto da su registri opšte namene realizovani kao registarski fajl i da ne postoji mogućnost da se u okviru registarskog fajla vrši dekrementiranje i inkrementiranje pojedinačnih registara. S toga je uzeto da se registar R_i prebacuje u registar B, da se sadržaj registra B dekrementira ili inkrementira i da se dobijena vrednost upisuje u registar R_i.

Ukoliko je signal dirmem aktivan, radi se o memorijskom direktnom adresiranju. Operand je tada u memoriji na adresi koja se nalazi u razredima IR3 i IR4 označenim sa IR_DA čiji se sadržaj prebacuje u registar MAR i prelazi na korak 4 počev od koga se, na već opisani način, za sve operacije, sem operacije STORE, čita operand i smešta u registar B. Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 4 i fazu *izvršavanje operacija* (slika 1.c).

Ukoliko je signal indmem aktivan, radi se o memorijskom indirektnom adresiranju. Operand je tada u memoriji na adresi određenoj sadržajem memorijske lokacije čija se adresa nalazi u registrima IR3 i IR4 označenim sa IR_DA. Stoga se, najpre, sadržaj ovih registara prebacuje u registar MAR, pa se iz memorije M sa adrese određene sadržajem registra MAR čita reč i upisuje u registar MBR. Operand je u memoriji na adresi koja se nalazi u registru MBR čiji se sadržaj prebacuje u registar MAR i prelazi na korak 3 počev od koga se, na već opisani način, za sve operacije, sem operacije STORE, čita operand i smešta u registar B. Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 4 i fazu *izvršavanje operacija* (slika 1.c).

Ukoliko je signal indregpom aktivan, radi se o registarskom indirektnom adresiranju sa pomerajem. Operand je tada u memoriji na adresi koja se dobija sabiranjem sadržaja registra opšte namene R_i određenog vrednošću druge grupe bitova druge reči instrukcije iz registra IR2, dok se pomeraj nalazi u registrima IR3 i IR4 označenim sa IR_DA. Dobijena adresa se

prebacuje u registar MAR i prelazi na korak 4 počev od koga se, na već opisani način, za sve operacije, sem operacije STORE, čita operand i smešta u registar B. Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 5 i fazu *izvršavanje operacija* (slika 1.c).

Ukoliko je signal immed aktivan, radi se o neposrednom adresiranju. Operand se tada nalazi u registrima IR2 i IR3 označenim sa IR_DA, čiji se sadržaj prebacuje u registar B. Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 4 i fazu *izvršavanje operacija* (slika 1.c).

izvršavanje operacija (slika 1.*c)*

Izvršavanje operacija se realizuje počev od koraka 4 po posebnom algoritmu za svaku od navedenih operacija prolaskom kroz odgovarajuće korake. Na početku se realizuje višestruki uslovni skok na jedan od dijagrama toka na osnovu toga koji je od signala logičkih uslova operacija aktivan. Aktivna vrednost jednog od signala operacija LOAD, STORE, ADD, AND, ASR, JZ, JMP, JSR, RTI ili RTS određuje da je specificirana operacija prenosa u akumulator, operacija prenosa iz akumulatora, aritmetička operacija sabiranja, logička operacija logički proizvod, operacija aritmetičkog pomeranja udesno za jedno mesto, uslovnog skoka ukoliko je rezultat nula, bezuslovnog skoka, skoka na potprograma, povratka iz prekidne rutine i povratka iz potprograma, respektivno.

Ukoliko je signal LOAD aktivan, sadržaj registra B se prebacuje u registar akumulatora A. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal STORE aktivan, sadržaj registra akumulatora A se prebacuje u registar opšte namene R_i određen vrednošću druge grupe bitova druge reči instrukcije iz registra IR2, ukoliko je specificirano direktno registarsko adresiranje ili u memorijsku lokaciju, čija se adresa nalazi u registru MAR, ukoliko je specificirano neko od memorijskih adresiranja. Neposredno adresiranje nije dozvoljeno za odredišni operand i zato se uzima da se u slučaju da se u instrukciji STORE pojavi neposredno adresiranje, faza izvršavanja ove operacije preskače, čime se ova instrukcija pretvara u instrukciju bez dejstva. S toga se proverava da li je signal immed aktivan. Ukoliko jeste, specificirano je neposredno adresiranje, pa se faza *izvršavanja operacija* završava i prelazi na korak 5 i fazu *opsluživanje prekida* (slika 1.d). Ukoliko je signal immed neaktivan, proverava se da li je signal regdir aktivan. Ukoliko jeste, registarsko direktno adresiranje je specificirano, pa se sadržaj registra A upisuje u registar opšte namene R_i određen vrednošću druge grupe bitova druge reči instrukcije iz registra IR2 i prelazi na korak 5 i fazu *opsluživanje prekida* (slika 1.d). Ukoliko nije, neko od memorijskih adresiranja je specificirano, pa se sadržaj registra A prebacuje u registar MBR i upisuje u memorijsku lokaciju određenu sadržajem registra MAR. Time je završena faza *izvršavanje operacija* i prelazi se na korak 7 i fazu *opsluživanje prekida* (slika 1.d).

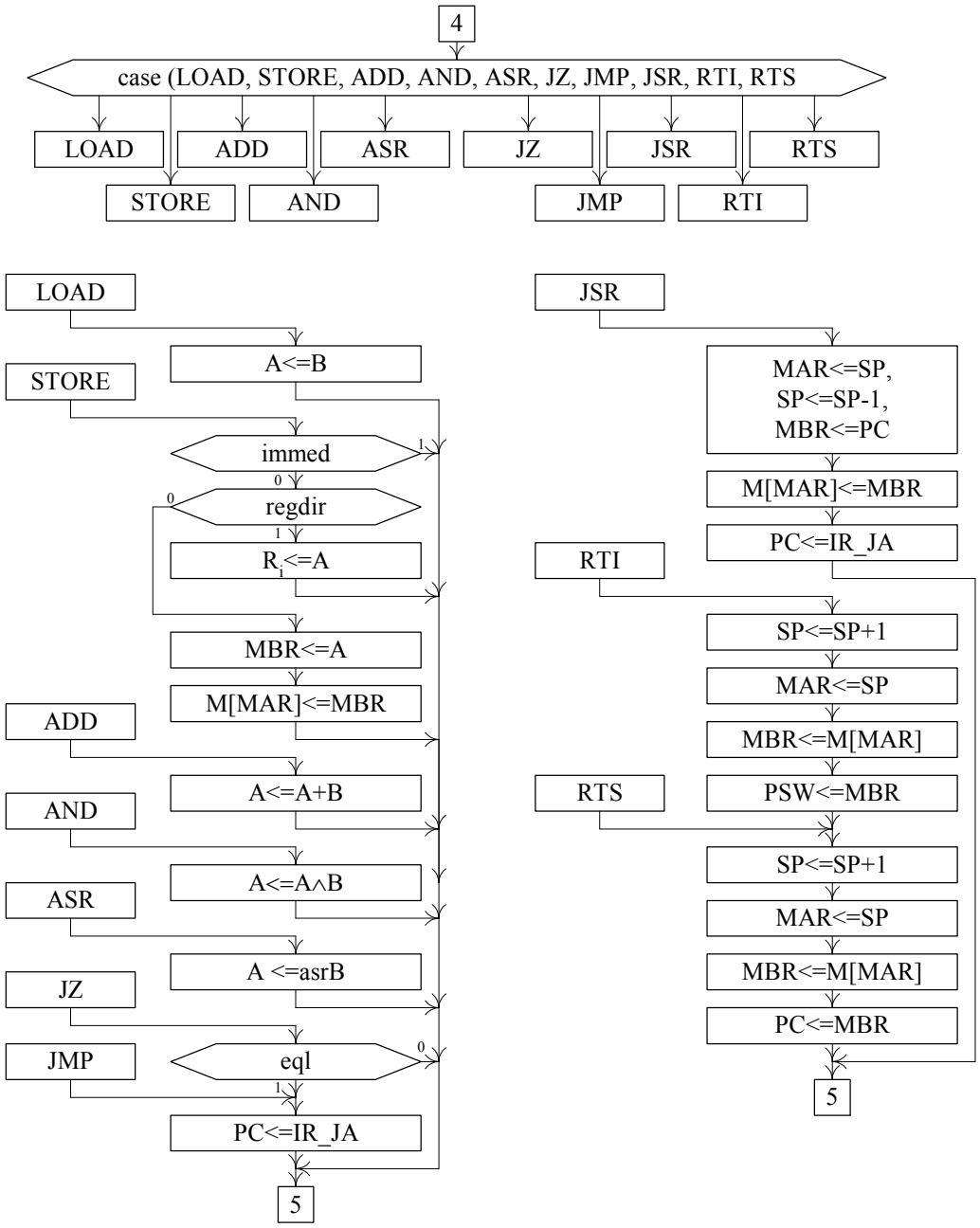
Ukoliko je signal ADD aktivan, sabiraju se sadržaji registara A i B i rezultat upisuje u registar A. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal AND aktivan, logička I operacija se realizuje nad sadržajima registara A i B i rezultat upisuje u registar A. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal ASR aktivan, sadržaj B se aritmetički pomera udesno za jedno mesto i upisuje u registar A. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal JZ aktivan, uslovni skok na osnovu vrednosti signala logičkog uslova rezultata operacija eql se realizuje. Signali logičkih uslova rezultata operacija eql (rezultat

nula), neq (rezultat nije nula), gtr (rezultat veći od nule), lss (rezultat manji od nule) itd. se formiraju na osnovu vrednosti indikatora N, Z, C i V registra programske statusne reči PSW. Signal rezultata operacija eql ima vrednost 1 ukoliko je rezultat zadnje izvršene instrukcije 0 i vrednost 0 ukoliko rezultat zadnje izvršene instrukcije nije 0. Ukoliko je signal eql 0, uslov za skok nije ispunjen. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d). Ukoliko je signal eql 1, uslov za skok je ispunjen, pa se sadržaj registara IR2 i IR3 označeni sa IR_JA, koji predstavlja adresu skoka, upisuje u registar PC. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).



Slika 1.c Dijagram toka – faza izvršavanje operacija

Ukoliko je signal JMP aktivan, bezuslovni skok se realizuje. Sadržaj registara IR2 i IR3 označeni sa IR_JA, koji predstavlja adresu skoka, upisuje se u registar PC. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

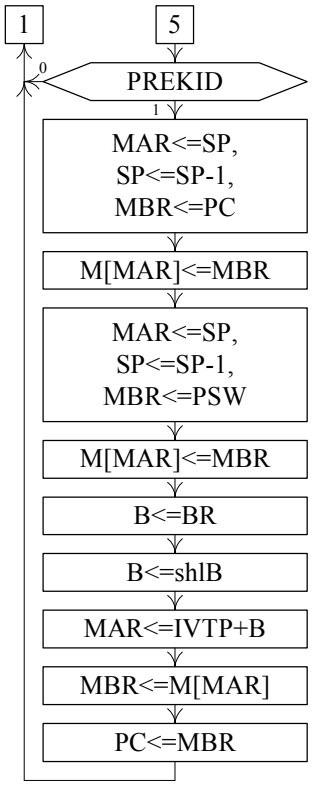
Ukoliko je signal JSR aktivan, skok na potprogram se realizuje. U okviru toga se sadržaj registra PC stavlja na stek. Najpre se prebacuje sadržaja registar SP u registar MAR, dekrementira sadržaj registra SP i prebacuje sadržaj registra PC u registar MBR. Zatim se sadržaj registra MBR upisuje u memorijsku lokaciju određenu sadržajem registra MAR. Na kraju se sadržaj registara IR2 i IR3 označeni sa IR_JA, koji predstavlja adresu skoka, upisuje se u registar PC. Treba uočiti da stek raste prema nižim lokacijama i da registar SP ukazuje na prvu slobodnu lokaciju. S toga se prilikom upisa na stek, prvo sadržaj registra SP prebacuje u registar MAR i posle toga inkrementira. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal RTI aktivan, povratak iz prekidne rutine se realizuje. U okviru toga se sadržajima sa steka restauriraju sadržaji registara PSW i PC. Najpre se sadržaj registra SP inkrementira i prebacuje u registar MAR. Zatim se iz memorijske lokacije određene sadržajem registra MAR čita sadržaj i upisuje u registar MBR. Na kraju se sadržaj registra MBR upisuje u registar PSW. Na isti način se sa steka čita još jedna reč i upisuje u registar PC. Treba uočiti da stek raste prema nižim lokacijama i da registar SP ukazuje na prvu slobodnu lokaciju. S toga se prilikom čitanja sadržaja sa steka, prvo inkrementira sadržaj registra SP i posle toga prebacuje u registar MAR. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal RTS aktivan, povratak iz potprograma se realizuje. U okviru toga se sadržajem sa steka restaurira sadržaj registra PC. Ovo se realizuje na identičan kao i u slučaju instrukcije RTI. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

opsluživanje prekida (slika 1.*d)*

Opsluživanje prekida se realizuje počev od koraka 5.



Slika 1.d Dijagram toka – faza opsluživanje prekida

Ukoliko je signal PREKID 0, u toku izvršavanja prethodnih faza nije došlo do generisanja signala prekida, pa se faza *opsluživanje prekida* završava i prelazi se na korak 1 i fazu *čitanje instrukcije* (slika 1.a).

Ukoliko je signal PREKID 1, u toku izvršavanja prethodnih faza došlo je do generisanja signala prekida, pa se prelazi na korake u okviru kojih se na steku najpre čuvaju sadržaji registara PC i PSW i potom utvrđuje adresa prekidne rutine i upisuje u registar PC. Čuvanje sadržaja registra PC na steku se realizuje na identičan način kao i u slučaju instrukcije JSR. Na isti način se na stek stavlja i sadržaj registra PSW. Adrese prekidnih rutina se nalaze u ulazima tabele sa adresama prekidnih rutina. Broj ulaza u tabelu je dat sadržajem registra BR, a početna adresa tabele sadržajem registra IVTP. Najpre se sadržaj registra BR prebacuje u registar B, pa se sadržaj registra B pomeranjem ulevo za jedno mesto množi sa dva. Time se broj ulaza pretvara u pomeraj. Potom se sabiranjem sadržaja registara IVTP i B i smeštanjem u registar MAR dobija adresa na kojoj se nalazi adresa prekidne rutine. Sa te i sledeće adrese iz memorije se čitaju dva bajta i upisuju u registar PC.

Faza *opsluživanje prekida* je time završena i prelazi se na korak 1 i *fazu čitanje instrukcije* (slika 1.a).

Treba uočiti da se javljaju dve situacije vezane za vrednost registra PC po završetku faze *opsluživanje prekida* i prelaska na korak 1 i fazu *čitanje instrukcije* (slika 1.a). Ukoliko je signal PREKID bio 0, u registru PC je adresa prve sledeće instrukcije posle instrukcije koja je izvršena. Ukoliko je signal PREKID bio 1, u registru PC je adresa prve instrukcije prekidne rutine.

1.2 OPERACIONA JEDINICA

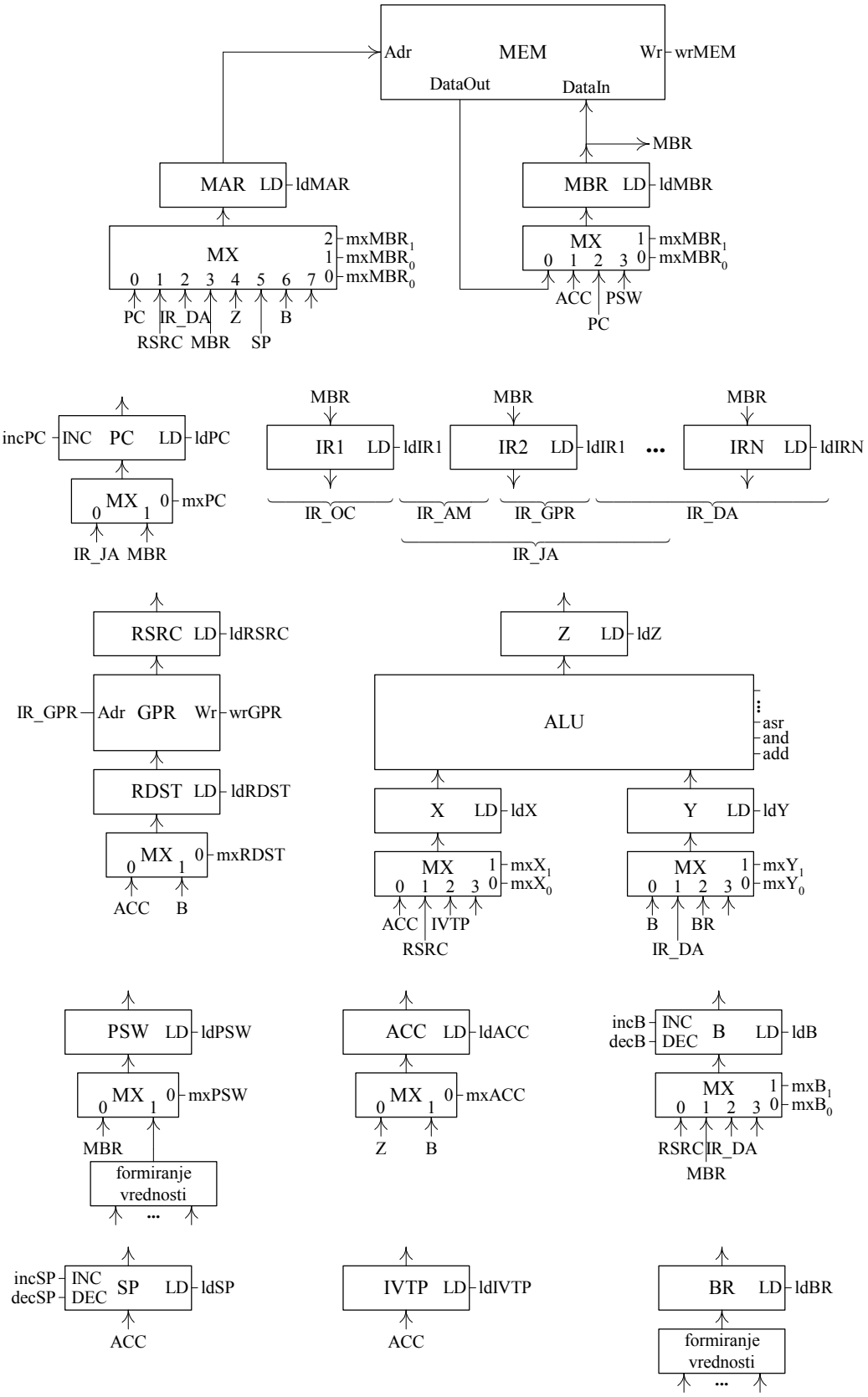
U ovom odeljku se razmatraju realizacije operacionih jedinica kod kojih su prekidačke mreže povezane direktno i pomoću jedne, dve i tri interne magistrale.

1.2.1 OPERACIONA JEDINICA SA DIREKTNIM VEZAMA

U ovom odeljku se razmatraju struktura upravljačke jedinice i algoritam generisanja upravljačkih signala.

1.2.1.1 Struktura operacione jedinice

Struktura operacione jedinice sa direktnim vezama je data na slici 2.



Slika 2 Operaciona jedinica sa direktnim vezama

1.2.1.2 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija (slika 1) i dat u obliku sekvence upravljačkih signala po koracima (tabela 1).

U sekvenci upravljačkih signala po koracima se koriste iskazi za signale i skokove. Iskazi za signale su oblika

signali.

Ovaj iskaz sadrži spisak upravljačkih signala operacione jedinice i određuje koji se signali bezuslovno generišu. Iskazi za skokove su oblika

br $step_A$,

br (*if* **uslov** *then* $step_A$) i

br (*case* (**$uslov_1$**, ..., **$uslov_n$**) *then* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$).

Prvi iskaz sadrži korak $step_A$ na koji treba bezuslovno preći i u daljem tekstu se referiše kao bezuslovni skok. Drugi iskaz sadrži signal **uslov** i korak $step_A$ i određuje korak $step_A$ na koji treba preći ukoliko signal **uslov** ima aktivnu vrednost i u daljem tekstu se referiše kao uslovni skok. Treći iskaz sadrži signale **$uslov_1$**, ..., **$uslov_n$** i korake $step_{A1}$, ..., $step_{An}$ i određuje na koji od koraka $step_{A1}$, ..., $step_{An}$ treba preći u zavisnosti od toga koji od signala **$uslov_1$**, ..., **$uslov_n$** ima aktivnu vrednost i u daljem tekstu se referiše kao višestruki uslovni skok.

Tabela 1 Sekvenca upravljačkih signala po koracima bez spajanja operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;
$step_{02}$ **ldIR1**;
$step_{03}$ *br* (*if* **l1** *then* $step_{31}$);
$step_{04}$ **ldMAR**, **incPC**;
$step_{05}$ **ldMBR**;
$step_{06}$ **ldIR2**;
$step_{07}$ *br* (*if* **l2** *then* $step_{0F}$);
$step_{08}$ **ldMAR**, **incPC**;
...
$step_{0E}$ **ldIRN**;

! Formiranje adrese i čitanje operanda !

$step_{0F}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, $step_{10}$), (**indreg**, $step_{13}$), (**postdec**, $step_{16}$), (**preinc**, $step_{1C}$),
(**dirmem**, $step_{22}$), (**indmem**, $step_{24}$), (**indregpom**, $step_{28}$), (**immed**, $step_{30}$));

! Direktno registarsko !

$step_{10}$ **ldRSRC**;
$step_{11}$ **ldB**;
$step_{12}$ *br* $step_{31}$;

! Indirektno registarsko !

$step_{13}$ **ldRSRC**;
$step_{14}$ **mxMAR$_0$**, **ldMAR**;
$step_{15}$ *br* $step_{2C}$;

! Postdekrement !

$step_{16}$ **ldRSRC**;
$step_{17}$ **mxMAR$_0$**, **ldMAR**, **ldB**;
$step_{18}$ **decB**;

step$_{19}$ **mxRDST**, **ldRDST**;
step$_{1A}$ **wrGPR**;
step$_{1B}$ *br* step$_{2C}$;

! Preinkrement !

step$_{1C}$ **ldRSRC**;
step$_{1D}$ **ldB**;
step$_{1E}$ **incB**;
step$_{1F}$ **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**, **mxRDST**, **ldRDST**;
step$_{20}$ **wrGPR**;
step$_{21}$ *br* step$_{2C}$;

! Direktno memorijsko !

step$_{22}$ **mxMAR$_1$**, **ldMAR**;
step$_{23}$ *br* step$_{2C}$;

! Indirektno memorijsko!

step$_{24}$ **mxMAR$_1$**, **ldMAR**;
step$_{25}$ **ldMBR**;
step$_{26}$ **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**;
step$_{27}$ *br* step$_{2C}$;

! Indirektno registarsko sa pomerajem !

step$_{28}$ **ldRSRC**;
step$_{29}$ **mxX$_0$**, **ldX**, **mxY$_0$**, **ldY**;
step$_{2A}$ **add**, **ldZ**;
step$_{2B}$ **mxMAR$_2$**, **ldMAR**;

! Čitanje operanda za memorijska adresiranja !

step$_{2C}$ *br* (*if* **STORE** *then* step$_{31}$);
step$_{2D}$ **ldMBR**;
step$_{2E}$ **mxB$_0$**, **ldB**;
step$_{2F}$ *br* step$_{31}$;

! Neposredno !

step$_{30}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

step$_{31}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, step$_{32}$), (**STORE**, step$_{34}$),
(**ADD**, step$_{3C}$), (**AND**, step$_{40}$), (**ASR**, step$_{44}$),
(**JZ**, step$_{48}$), (**JMP**, step$_{4C}$), (**JSR**, step$_{4A}$), (**RTI**, step$_{4E}$), (**RTS**, step$_{52}$));

! LOAD !

step$_{32}$ **mxACC**, **ldACC**;
step$_{33}$ *br* step$_{56}$;

! STORE !

step$_{34}$ *br* (*if* **immed** *then* step$_{56}$);
step$_{35}$ *br* (*if* **regdir** *then* step$_{39}$);
step$_{36}$ **mxMBR$_0$**, **ldMBR**;
step$_{37}$ **wrMEM**;
step$_{38}$ *br* step$_{56}$;
step$_{39}$ **ldRDST**;
step$_{3A}$ **wrGPR**;
step$_{3B}$ *br* step$_{56}$;

! ADD !

step$_{3C}$ **ldX**, **ldY**;
step$_{3D}$ **add**, **ldZ**;
step$_{3E}$ **ldACC**;
step$_{3F}$ *br* step$_{56}$;

! AND !

$step_{40}$ **ldX**, **ldY**;
$step_{41}$ **and**, **ldZ**;
$step_{42}$ **ldACC**;
$step_{43}$ *br* $step_{56}$;

! ASR !

$step_{44}$ **ldY**;
$step_{45}$ **asr**, **ldZ**;
$step_{46}$ **ldACC**;
$step_{47}$ *br* $step_{56}$;

! JZ !

$step_{48}$ *br* (*if* **eql** *then* $step_{4C}$);
$step_{49}$ *br* $step_{56}$;

! JSR !

$step_{4A}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
$step_{4B}$ **wrMEM**;

! JMP !

$step_{4C}$ **ldPC**;
$step_{4D}$ *br* $step_{56}$;

! RTI !

$step_{4E}$ **incSP**;
$step_{4F}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$step_{50}$ **ldMBR**;
$step_{51}$ **ldPSW**;

! RTS !

$step_{52}$ **incSP**;
$step_{53}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$step_{54}$ **ldMBR**;
$step_{55}$ **mxPC**, **ldPC**;

! Opsluživanje prekida !

$step_{56}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{57}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
$step_{58}$ **wrMEM**;
$step_{59}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **mxMBR$_0$**,**ldMBR**;
$step_{5A}$ **wrMEM**;
$step_{5B}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
$step_{5C}$ **add**, **ldZ**;
$step_{5D}$ **mxMAR$_2$**, **ldMAR**;
$step_{5E}$ **ldMBR**;
$step_{5F}$ **mxPC**, **ldPC**;
$step_{60}$ *br* $step_{00}$;

Operacioni korak i prvi sledeći upravljački korak u nekim situacijama mogu da se spoje u isti korak. Time se ukupan broj koraka neophodnih za izvršavanje instrukcije smanjuje, čime se povećava brzina izvršavanja instrukcija.

Ako je upravljački korak bezuslovni skok, tada se dati upravljački korak i prethodni korak koji je operacioni korak mogu spojiti ukoliko se na dati upravljački korak prelazi samo iz prethodnog koraka koji je operacioni korak a ne i iz još nekog koraka koji je upravljački korak. Primeri su koraci $step_{11}$ **ldB** i $step_{12}$ *br* $step_{31}$, zatim koraci $step_{14}$ **mxMAR$_0$**, **ldMAR** i $step_{15}$ *br* $step_{2C}$ itd.

Ako je upravljački korak uslovni skok, tada se dati upravljački korak i prethodni korak koji je operacioni korak mogu spojiti ukoliko signal logičkog uslova koji se konsultuje pri uslovnom skoku ne zavisi od mikrooperacija izvršenih na osnovu upravljačkih signala generisanih u operacionom koraku i ukoliko se na dati upravljački korak prelazi samo iz prethodnog koraka koji je operacioni korak a ne i iz još nekog koraka koji je upravljački korak. U suprotnom slučaju koraci se ne mogu spojiti. Primera kada to može da se učini nema u sekvenci u tabeli 1. Primeri kada to ne može da se učini su koraci $step_{02}$ **ldIR1** i $step_{03}$ *br* (*if* **l1** *then* $step_{31}$), zatim koraci $step_{06}$ **ldIR2** i $step_{07}$ *br* (*if* **l2** *then* $step_{0F}$) itd. Na primer, u koraku $step_{02}$ se signalom **ldIR1** prva reč instrukcije, koja sadrži polje koda operacije, upisuje u registar IR i na osnovu nje se formira vrednost signala logičkog uslova **l1**, dok se u koraku $step_{03}$ vrši provera signala **l1** i u zavisnosi od njegove vrednosti prelazi ili na korak $step_{31}$ ili na korak $step_{04}$. Zbog toga koraci $step_{02}$ i $step_{03}$ ne mogu da se spoje. Takođe, u koraku $step_{06}$ se signalom **ldIR2** druga reč instrukcije, koja sadrži polje sa načinima adresiranja, upisuje u registar IR i na osnovu nje se formira vrednost signala logičkog uslova **l2**, dok se u koraku $step_{07}$ vrši provera signala **l2** i u zavisnosi od njegove vrednosti prelazi ili na korak $step_{0F}$ ili na korak $step_{08}$. Zbog toga koraci $step_{07}$ i $step_{08}$ ne mogu da se spoje. Treba uočiti da se koraci $step_{2B}$ **mxMAR$_2$**, **ldMAR** i $step_{2C}$ *br* (*if* **STORE** *then* $step_{31}$) ne mogu spojiti iako je signal logičkog uslova **STORE**, koji se konsultuje u koraku $step_{2C}$, formiran još u koraku $step_{02}$ kada je, signalom **ldIR1**, prva reč instrukcije, koja sadrži polje koda operacije, upisana u registar IR i ne zavisi od mikrooperacija izvršenih na osnovu upravljačkih signala generisanih u koraku $step_{2B}$, jer se na korak $step_{2C}$ prelazi ne samo iz koraka $step_{2B}$ već i iz koraka $step_{15}$ *br* $step_{2C}$, $step_{1B}$ *br* $step_{2C}$ itd.

Ako je upravljački korak višestruki uslovni skok, tada se dati upravljački korak i prethodni korak koji je operacioni korak mogu spojiti ukoliko ni jedan od signala logičkih uslova koji se konsultuju pri višestrukom uslovnom skoku ne zavisi od mikrooperacija izvršenih na osnovu upravljačkih signala generisanih u operacionom koraku i ukoliko se na dati upravljački korak prelazi samo iz prethodnog koraka koji je operacioni korak a ne i iz još nekog koraka koji je upravljački korak. U suprotnom slučaju koraci se ne mogu spojiti. Primera kada to može da se učini nema u sekvenci u tabeli 1. Primeri kada to ne može da se učini su koraci $step_{0E}$ **ldIRN** i $step_{0F}$ *br* (*case* (**dirreg**, **indreg**, ..., **immed**) ...) i koraci $step_{30}$ **mxB$_1$**, **ldB** i $step_{31}$ *br* (*case* (**LOAD**, **STORE**, ..., **RTS**) ...). Koraci $step_{0E}$ **ldIRN** i $step_{0F}$ *br* (*case* (**dirreg**, **indreg**, ..., **immed**) ...) se ne mogu spojiti iako su signali logičkih uslova **dirreg**, **indreg**, ..., **immed**, koji se konsultuju u koraku $step_{0F}$, formirani još u koraku $step_{06}$ kada je, signalom **ldIR2**, druga reč instrukcije, koja sadrži polje načina adresirana, upisana u registar IR i ne zavisi od mikrooperacija izvršenih na osnovu upravljačkih signala generisanih u koraku $step_{0E}$, jer se na korak $step_{0F}$ prelazi ne samo iz koraka $step_{0E}$ već i iz koraka $step_{07}$ itd. Takođe se koraci $step_{30}$ **mxB$_1$**, **ldB** i $step_{31}$ *br* (*case* (**LOAD**, **STORE**, ..., **RTS**) ...) se ne mogu spojiti iako su signali logičkih uslova **LOAD**, **STORE**, ..., **RTS**, koji se konsultuju u koraku $step_{31}$, formirani još u koraku $step_{02}$ kada je, signalom **ldIR1**, prva reč instrukcije, koja sadrži polje koda operacije, upisana u registar IR i ne zavisi od mikrooperacija izvršenih na osnovu upravljačkih signala generisanih u koraku $step_{30}$, jer se na korak $step_{31}$ prelazi ne samo iz koraka $step_{30}$ već i iz koraka $step_{03}$, $step_{12}$, itd.

Operacioni korak i prvi sledeći upravljački korak ne bi mogli da se spoje u isti korak i u situacijama kada operacioni korak traje više od jedne periode signala takta. Takve situacije bi mogle da se jave u koracima $step_{1A}$ **wrGPR** i $step_{1B}$ *br* $step_{2C}$, koracima $step_{37}$ **wrMEM** i $step_{38}$ *br* $step_{56}$ itd. ukoliko bi upis ili u neki od registara opšte namene, iniciran generisanjem signala **wrGPR** u koraku $step_{1A}$, ili u neku memorijsku lokaciju, iniciran generisanjem signala **wrMEM** u koraku $step_{37}$, trajao više od jedne periode signala takta. Takvih primera nema u

sekvenci u tabeli 1, jer je pretpostavljeno da svi operacioni i upravljački koraci traju jednu periodu signala takta. Zbog toga se koraci $step_{1A}$ **wrGPR** i $step_{1B}$ *br* $step_{2C}$, koraci $step_{37}$ **wrMEM** i $step_{38}$ *br* $step_{56}$ itd. mogu spajati.

Kada se, saglasno prethodnim razmatranjima, izvrši spajanje operacionih i upravljačkih koraka iz tabele 1, dobija se sekvenca upravljačkih signala po koracima kao što je dato u tabeli 2.

Tabela 2 Sekvenca upravljačkih signala po koracima sa spajanjem operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;
$step_{02}$ **ldIR1**;
$step_{03}$ *br* (*if* **l1** *then* $step_{2A}$);
$step_{04}$ **ldMAR**, **incPC**;
$step_{05}$ **ldMBR**;
$step_{06}$ **ldIR2**;
$step_{07}$ *br* (*if* **l2** *then* $step_{0F}$);
$step_{08}$ **ldMAR**, **incPC**;
...
$step_{0E}$ **ldIRN**;

! Formiranje adrese i čitanje operanda !

$step_{0F}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, $step_{10}$), (**indreg**, $step_{12}$), (**postdec**, $step_{14}$), (**preinc**, $step_{19}$),
(**dirmem**, $step_{1E}$), (**indmem**, $step_{1F}$), (**indregpom**, $step_{22}$), (**immed**, $step_{29}$));

! Direktno registarsko !

$step_{10}$ **ldRSRC**;
$step_{11}$ **ldB**, *br* $step_{2A}$;

! Indirektno registarsko !

$step_{12}$ **ldRSRC**;
$step_{13}$ **mxMAR$_0$**, **ldMAR**, *br* $step_{26}$;

! Postdekrement !

$step_{14}$ **ldRSRC**;
$step_{15}$ **mxMAR$_0$**, **ldMAR**, **ldB**;
$step_{16}$ **decB**;
$step_{17}$ **mxRDST**, **ldRDST**;
$step_{18}$ **wrGPR**, *br* $step_{26}$;

! Preinkrement !

$step_{19}$ **ldRSRC**;
$step_{1A}$ **ldB**;
$step_{1B}$ **incB**;
$step_{1C}$ **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**, **mxRDST**, **ldRDST**;
$step_{1D}$ **wrGPR**, *br* $step_{26}$;

! Direktno memorijsko !

$step_{1E}$ **mxMAR$_1$**, **ldMAR**, *br* $step_{26}$;

! Indirektno memorijsko!

$step_{1F}$ **mxMAR$_1$**, **ldMAR**;
$step_{20}$ **ldMBR**;
$step_{21}$ **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**, *br* $step_{26}$;

! Indirektno registarsko sa pomerajem !

$step_{22}$ **ldRSRC**;
$step_{23}$ **mxX$_0$**, **ldX**, **mxY$_0$**, **ldY**;
$step_{24}$ **add**, **ldZ**;
$step_{25}$ **mxMAR$_2$**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
$step_{26}$ *br* (*if* **STORE** *then* $step_{2A}$);
$step_{27}$ **ldMBR**;
$step_{28}$ **mxB$_0$**, **ldB**, *br* $step_{2A}$;
! Neposredno !
$step_{29}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

$step_{2A}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, $step_{2B}$), (**STORE**, $step_{2C}$),
(**ADD**, $step_{32}$), (**AND**, $step_{35}$), (**ASR**, $step_{38}$),
(**JZ**, $step_{3B}$), (**JMP**, $step_{3F}$), (**JSR**, $step_{3D}$), (**RTI**, $step_{40}$), (**RTS**, $step_{44}$));
! LOAD !
$step_{2B}$ **mxACC**, **ldACC**, *br* $step_{48}$;
! STORE !
$step_{2C}$ *br* (*if* **immed** *then* $step_{48}$);
$step_{2D}$ *br* (*if* **regdir** *then* $step_{30}$);
$step_{2E}$ **mxMBR$_0$**, **ldMBR**;
$step_{2F}$ **wrMEM**, *br* $step_{48}$;
$step_{30}$ **ldRDST**;
$step_{31}$ **wrGPR**, *br* $step_{48}$;
! ADD !
$step_{32}$ **ldX**, **ldY**;
$step_{33}$ **add**, **ldZ**;
$step_{34}$ **ldACC**, *br* $step_{48}$;
! AND !
$step_{35}$ **ldX**, **ldY**;
$step_{36}$ **and**, **ldZ**;
$step_{37}$ **ldACC**, *br* $step_{48}$;
! ASR !
$step_{38}$ **ldY**;
$step_{39}$ **asr**, **ldZ**;
$step_{3A}$ **ldACC**, *br* $step_{48}$;
! JZ !
$step_{3B}$ *br* (*if* **Z** *then* $step_{3F}$);
$step_{3C}$ *br* $step_{48}$;
! JSR !
$step_{3D}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
$step_{3E}$ **wrMEM**;
! JMP !
$step_{3F}$ **ldPC**, *br* $step_{48}$;
! RTI !
$step_{40}$ **incSP**;
$step_{41}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$step_{42}$ **ldMBR**;
$step_{43}$ **ldPSW**;
! RTS !
$step_{44}$ **incSP**;
$step_{45}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;

$step_{46}$ **ldMBR**;
$step_{47}$ **mxPC**, **ldPC**;

! Opsluživanje prekida !

$step_{48}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{49}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
$step_{4A}$ **wrMEM**;
$step_{4B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **mxMBR$_0$**,**ldMBR**;
$step_{4C}$ **wrMEM**;
$step_{4D}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
$step_{4E}$ **add**, **ldZ**;
$step_{4F}$ **mxMAR$_2$**, **ldMAR**;
$step_{50}$ **ldMBR**;
$step_{51}$ **mxPC**, **ldPC**, *br* $step_{00}$;

Tabela sa spajanjem koraka (tabela 2) ima izgled veoma sličan tabeli bez spajanja koraka (tabela 1). Razlika je samo u tome da pored koraka u kojima se pojavljuju samo iskazi za signale ili samo iskazi za skokove, postoje i koraci u kojima se pojavljuju i iskazi za signale i iskazi za skokove. Koraci u tabeli 2 u kojima se pojavljuju i iskazi za signale i iskazi za skokove odgovaraju situacijama kada je bilo moguće spajanje operacionih koraka i upravljačkih koraka iz tabele 1.

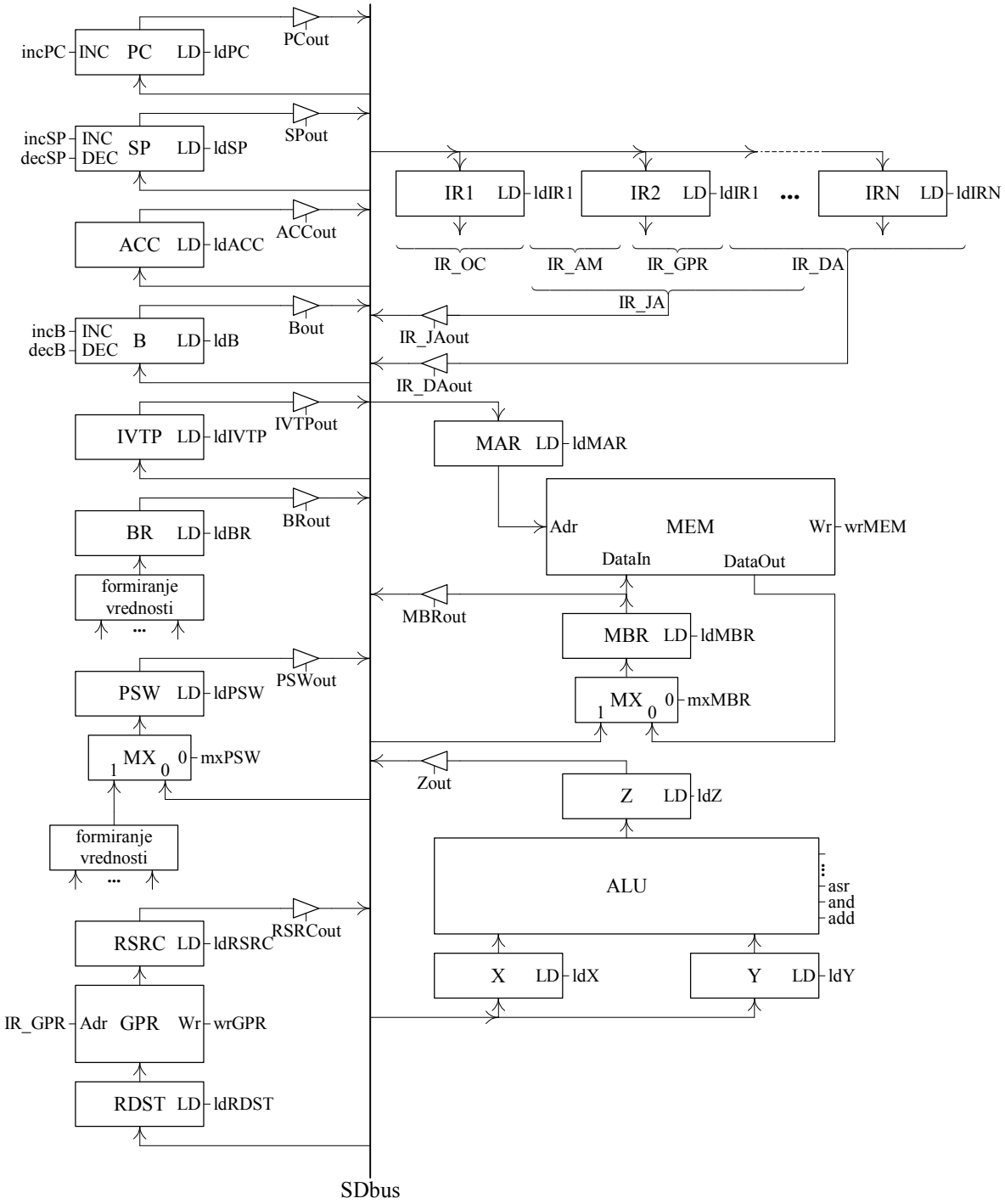
Kao rezultat spajanja koraka tabela 2 ima manji broj koraka od tabele 1. Iz istih razloga su u većini iskaza za skokove promenjene i vrednosti koraka na koje se skače.

1.2.2 OPERACIONA JEDINICA SA JEDNOM MAGISTRALOM

U ovom odeljku se razmatraju struktura upravljačke jedinice i algoritam generisanja upravljačkih signala.

1.2.2.1 Struktura operacione jedinice

Struktura operacione jedinice sa jednom magistralom je data na slici 3.



Slika 3 Operaciona jedinica sa jednom magistralom

1.2.2.2 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija (slika 1) i dat u obliku sekvence upravljačkih signala po koracima (tabela 3). U ovoj tabeli se pojavlju isti iskazi kao i u slučaju tabele 1 za operacionu jedinicu sa direktnim vezama.

Tabela 3 Sekvenca upravljačkih signala po koracima bez spajanja operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **PCout**, **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;

$step_{02}$ **MBRout**, **ldIR1**;
$step_{03}$ *br* (*if* **I1** *then* $step_{31}$);
$step_{04}$ **PCout**, **ldMAR**, **incPC**;
$step_{05}$ **ldMBR**;
$step_{06}$ **MBRout**, **ldIR2**;
$step_{07}$ *br* (*if* **I2** *then* $step_{0F}$);
$step_{08}$ **PCout**, **ldMAR**, **incPC**;
...
$step_{0E}$ **MBRout**, **ldIRN**;

! Formiranje adrese i čitanje operanda !

$step_{0F}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, $step_{10}$), (**indreg**, $step_{13}$), (**postdec**, $step_{16}$), (**preinc**, $step_{1C}$),
(**dirmem**, $step_{22}$), (**indmem**, $step_{24}$), (**indregpom**, $step_{28}$), (**immed**, $step_{30}$));

! Direktno registarsko !

$step_{10}$ **ldRSRC**;
$step_{11}$ **RSRCout**, **ldB**;
$step_{12}$ *br* $step_{31}$;

! Indirektno registarsko !

$step_{13}$ **ldRSRC**;
$step_{14}$ **RSRCout**, **ldMAR**;
$step_{15}$ *br* $step_{2C}$;

! Postdekrement !

$step_{16}$ **ldRSRC**;
$step_{17}$ **RSRCout**, **ldMAR**, **ldB**;
$step_{18}$ **decB**;
$step_{19}$ **Bout**, **ldRDST**;
$step_{1A}$ **wrGPR**;
$step_{1B}$ *br* $step_{2C}$;

! Preinkrement !

$step_{1C}$ **ldRSRC**;
$step_{1D}$ **RSRCout**, **ldB**;
$step_{1E}$ **incB**;
$step_{1F}$ **Bout**, **ldMAR**, **ldRDST**;
$step_{20}$ **wrGPR**;
$step_{21}$ *br* $step_{2C}$;

! Direktno memorijsko !

$step_{22}$ **IR_DAout**, **ldMAR**;
$step_{23}$ *br* $step_{2C}$;

! Indirektno memorijsko!

$step_{24}$ **IR_DAout**, **ldMAR**;
$step_{25}$ **ldMBR**;
$step_{26}$ **MBRout**, **ldMAR**;
$step_{27}$ *br* $step_{2C}$;

! Indirektno registarsko sa pomerajem !

$step_{28}$ **IR_DAout**, **ldY**, **ldRSRC**;
$step_{29}$ **RSRCout**, **ldX**;
$step_{2A}$ **add**, **ldZ**;
$step_{2B}$ **Zout**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
$step_{2C}$ *br* (*if* **STORE** *then* $step_{31}$);
$step_{2D}$ **ldMBR**;

step$_{2E}$ **MBRout**, **ldB**;
step$_{2F}$ *br* step$_{31}$;

! Neposredno !

step$_{30}$ **IR_DAout**, **ldB**;

! Izvršavanje operacije !

step$_{31}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, step$_{32}$), (**STORE**, step$_{34}$),
(**ADD**, step$_{3C}$), (**AND**, step$_{41}$), (**ASR**, step$_{46}$),
(**JZ**, step$_{4A}$), (**JMP**, step$_{4F}$), (**JSR**, step$_{4C}$), (**RTI**, step$_{51}$), (**RTS**, step$_{55}$));

! LOAD !

step$_{32}$ **Bout**, **ldACC**;
step$_{33}$ *br* step$_{59}$;

! STORE !

step$_{34}$ *br* (*if* **immed** *then* step$_{59}$);
step$_{35}$ *br* (*if* **regdir** *then* step$_{39}$);
step$_{36}$ **ACCout**, **ldMBR**;
step$_{37}$ **wrMEM**;
step$_{38}$ *br* step$_{59}$;
step$_{39}$ **ACCout**, **ldRDST**;
step$_{3A}$ **wrGPR**;
step$_{3B}$ *br* step$_{59}$;

! ADD !

step$_{3C}$ **ACCout**, **ldX**;
step$_{3D}$ **Bout**, **ldY**;
step$_{3E}$ **add**, **ldZ**;
step$_{3F}$ **Zout**, **ldACC**;
step$_{40}$ *br* step$_{59}$;

! AND !

step$_{41}$ **ACCout**, **ldX**;
step$_{42}$ **Bout**, **ldY**;
step$_{43}$ **and**, **ldZ**;
step$_{44}$ **Zout**, **ldACC**;
step$_{45}$ *br* step$_{59}$;

! ASR !

step$_{46}$ **Bout**, **ldY**;
step$_{47}$ **asr**, **ldZ**;
step$_{48}$ **Zout**, **ldACC**;
step$_{49}$ *br* step$_{59}$;

! JZ !

step$_{4A}$ *br* (*if* **Z** *then* step$_{4F}$);
step$_{4B}$ *br* step$_{59}$;

! JSR !

step$_{4C}$ **SPout**, **ldMAR**, **decSP**;
step$_{4D}$ **PCout**, **ldMBR**;
step$_{4E}$ **wrMEM**;

! JMP !

step$_{4F}$ **IR_JAout**, **ldPC**;
step$_{50}$ *br* step$_{59}$;

! RTI !

step$_{51}$ **incSP**;
step$_{52}$ **SPout**, **ldMAR**;
step$_{53}$ **ldMBR**;

$step_{54}$ **MBRout**, **ldPSW**;

! RTS !

$step_{55}$ **incSP**;
$step_{56}$ **SPout**, **ldMAR**;
$step_{57}$ **ldMBR**;
$step_{58}$ **MBRout**, **ldPC**;

! Opsluživanje prekida !

$step_{59}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{5A}$ **SPout**, **ldMAR**, **decSP**;
$step_{5B}$ **PCout**, **ldMBR**;
$step_{5C}$ **wrMEM**;
$step_{5D}$ **SPout**, **ldMAR**, **decSP**;
$step_{5E}$ **PSWout**, **ldMBR**;
$step_{5F}$ **wrMEM**;
$step_{60}$ **IVTPout**, **ldX**;
$step_{61}$ **BRout**, **ldY**;
$step_{62}$ **add**, **ldZ**;
$step_{63}$ **Zout**, **ldMAR**;
$step_{64}$ **ldMBR**;
$step_{65}$ **MBRout**, **ldPC**;
$step_{66}$ *br* $step_{00}$;

Operacioni korak i upravljački korak se, saglasno razmatranjima za operacionu jedinice sa direktnim vezama, mogu u nekim situacijama spojiti. Tada se dobija sekvenca upravljačkih signala po koracima kao što je dato u tabeli 4. U ovoj tabeli se pojavlju isti iskazi kao i u slučaju tabele 1 za operacionu jedinicu sa direktnim vezama.

Tabela 4 Sekvenca upravljačkih signala po koracima sa spajanjem operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **PCout**, **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;
$step_{02}$ **MBRout**, **ldIR1**;
$step_{03}$ *br* (*if* **I1** *then* $step_{2A}$);
$step_{04}$ **PCout**, **ldMAR**, **incPC**;
$step_{05}$ **ldMBR**;
$step_{06}$ **MBRout**, **ldIR2**;
$step_{07}$ *br* (*if* **I2** *then* $step_{0F}$);
$step_{08}$ **PCout**, **ldMAR**, **incPC**;
...
$step_{0E}$ **MBRout**, **ldIRN**;

! Formiranje adrese i čitanje operanda !

$step_{0F}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, $step_{10}$), (**indreg**, $step_{12}$), (**postdec**, $step_{14}$), (**preinc**, $step_{19}$),
(**dirmem**, $step_{1E)}$), (**indmem**, $step_{1F}$), (**indregpom**, $step_{22}$), (**immed**, $step_{29}$));

! Direktno registarsko !

$step_{10}$ **ldRSRC**;
$step_{11}$ **RSRCout**, **ldB**, *br* $step_{2A}$;

! Indirektno registarsko !

$step_{12}$ **ldRSRC**;

step$_{13}$ **RSRCout**, **ldMAR**, *br* step$_{26}$;
! Postdekrement !
step$_{14}$ **ldRSRC**;
step$_{15}$ **RSRCout**, **ldMAR**, **ldB**;
step$_{16}$ **decB**;
step$_{17}$ **Bout**, **ldRDST**;
step$_{18}$ **wrGPR**, *br* step$_{26}$;
! Preinkrement !
step$_{19}$ **ldRSRC**;
step$_{1A}$ **RSRCout**, **ldB**;
step$_{1B}$ **incB**;
step$_{1C}$ **Bout**, **ldMAR**, **ldRDST**;
step$_{1D}$ **wrGPR**, *br* step$_{26}$;
! Direktno memorijsko !
step$_{1E}$ **IR_DAout**, **ldMAR**, *br* step$_{26}$;
! Indirektno memorijsko!
step$_{1F}$ **IR_DAout**, **ldMAR**;
step$_{20}$ **ldMBR**;
step$_{21}$ **MBRout**, **ldMAR**, *br* step$_{26}$;
! Indirektno registarsko sa pomerajem !
step$_{22}$ **IR_DAout**, **ldY**, **ldRSRC**;
step$_{23}$ **RSRCout**, **ldX**;
step$_{24}$ **add**, **ldZ**;
step$_{25}$ **Zout**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
step$_{26}$ *br* (*if* **STORE** *then* step$_{2A}$);
step$_{27}$ **ldMBR**;
step$_{28}$ **MBRout**, **ldB**, *br* step$_{2A}$;
! Neposredno !
step$_{29}$ **IR_DAout**, **ldB**;

! Izvršavanje operacije !

step$_{2A}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, step$_{2B}$), (**STORE**, step$_{2C}$),
(**ADD**, step$_{32}$), (**AND**, step$_{36}$), (**ASR**, step$_{3A}$),
(**JZ**, step$_{3D}$), (**JMP**, step$_{42}$), (**JSR**, step$_{3F}$), (**RTI**, step$_{43}$), (**RTS**, step$_{47}$));
! LOAD !
step$_{2B}$ **Bout**, **ldACC**, *br* step$_{4B}$;
! STORE !
step$_{2C}$ *br* (*if* **immed** *then* step$_{4B}$);
step$_{2D}$ *br* (*if* **regdir** *then* step$_{30}$);
step$_{2E}$ **ACCout**, **ldMBR**;
step$_{2F}$ **wrMEM**, *br* step$_{4B}$;
step$_{30}$ **ACCout**, **ldRDST**;
step$_{31}$ **wrGPR**, *br* step$_{4B}$;
! ADD !
step$_{32}$ **ACCout**, **ldX**;
step$_{33}$ **Bout**, **ldY**;
step$_{34}$ **add**, **ldZ**;
step$_{35}$ **Zout**, **ldACC**, *br* step$_{4B}$;
! AND !
step$_{36}$ **ACCout**, **ldX**;
step$_{37}$ **Bout**, **ldY**;

$step_{38}$ **and**, **ldZ**;
$step_{39}$ **Zout**, **ldACC**, *br* $step_{4B}$;
! ASR !
$step_{3A}$ **Bout**, **ldY**;
$step_{3B}$ **asr**, **ldZ**;
$step_{3C}$ **Zout**, **ldACC**, *br* $step_{4B}$;
! JZ !
$step_{3D}$ *br* (*if* **Z** *then* $step_{42}$);
$step_{3E}$ *br* $step_{4B}$;
! JSR !
$step_{3F}$ **SPout**, **ldMAR**, **decSP**;
$step_{40}$ **PCout**, **ldMBR**;
$step_{41}$ **wrMEM**;
! JMP !
$step_{42}$ **IR_JAout**, **ldPC**, *br* $step_{4B}$;
! RTI !
$step_{43}$ **incSP**;
$step_{44}$ **SPout**, **ldMAR**;
$step_{45}$ **ldMBR**;
$step_{46}$ **MBRout**, **ldPSW**;
! RTS !
$step_{47}$ **incSP**;
$step_{48}$ **SPout**, **ldMAR**;
$step_{49}$ **ldMBR**;
$step_{4A}$ **MBRout**, **ldPC**;

! Opsluživanje prekida !

$step_{4B}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{4C}$ **SPout**, **ldMAR**, **decSP**;
$step_{4D}$ **PCout**, **ldMBR**;
$step_{4E}$ **wrMEM**;
$step_{4F}$ **SPout**, **ldMAR**, **decSP**;
$step_{50}$ **PSWout**, **ldMBR**;
$step_{51}$ **wrMEM**;
$step_{52}$ **IVTPout**, **ldX**;
$step_{53}$ **BRout**, **ldY**;
$step_{54}$ **add**, **ldZ**;
$step_{55}$ **Zout**, **ldMAR**;
$step_{56}$ **ldMBR**;
$step_{57}$ **MBRout**, **ldPC**, *br* $step_{00}$;

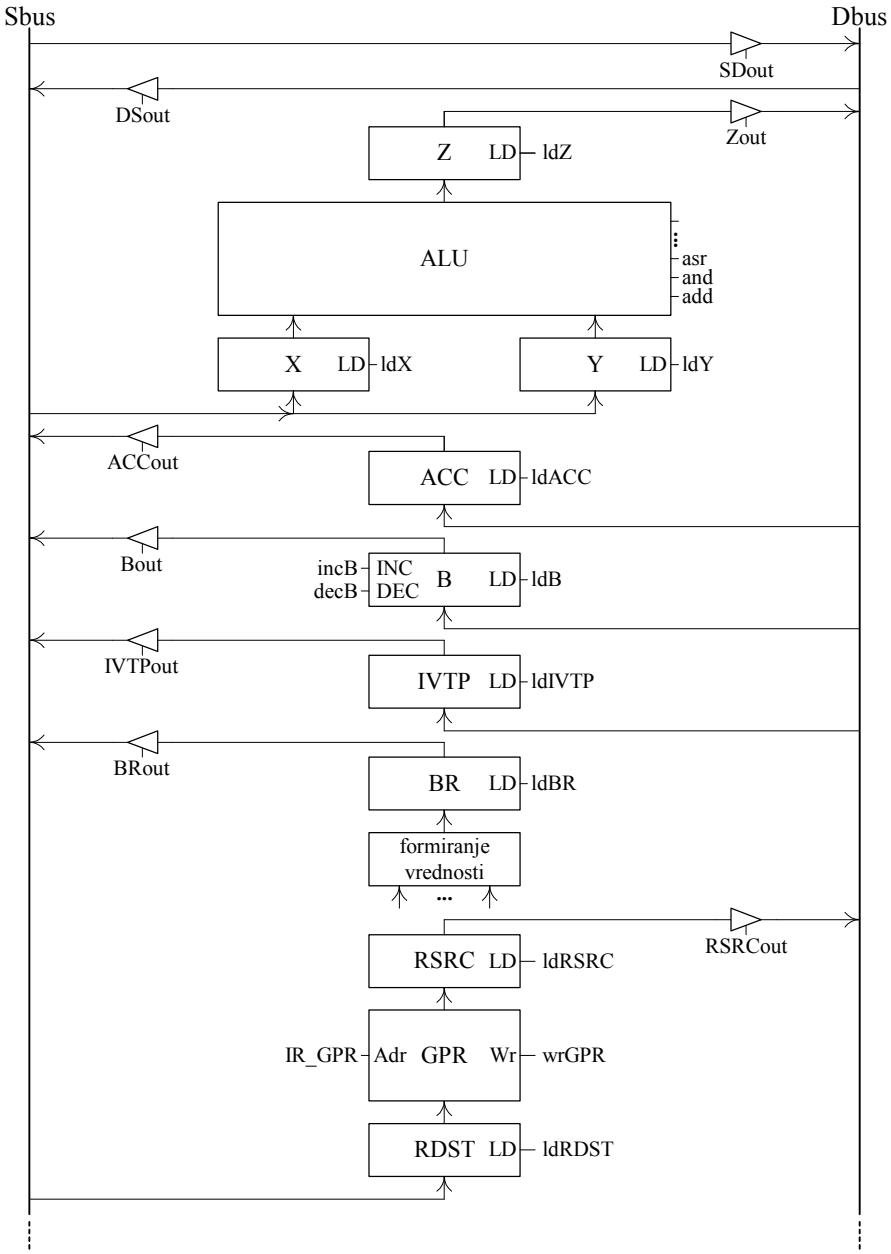
Kao rezultat spajanja koraka tabela 4 ima manji broj koraka od tabele 3. Iz istih razloga su u većini iskaza za skokove promenjene i vrednosti koraka na koje se skače.

1.2.3 OPERACIONA JEDINICA SA DVE MAGISTRALE

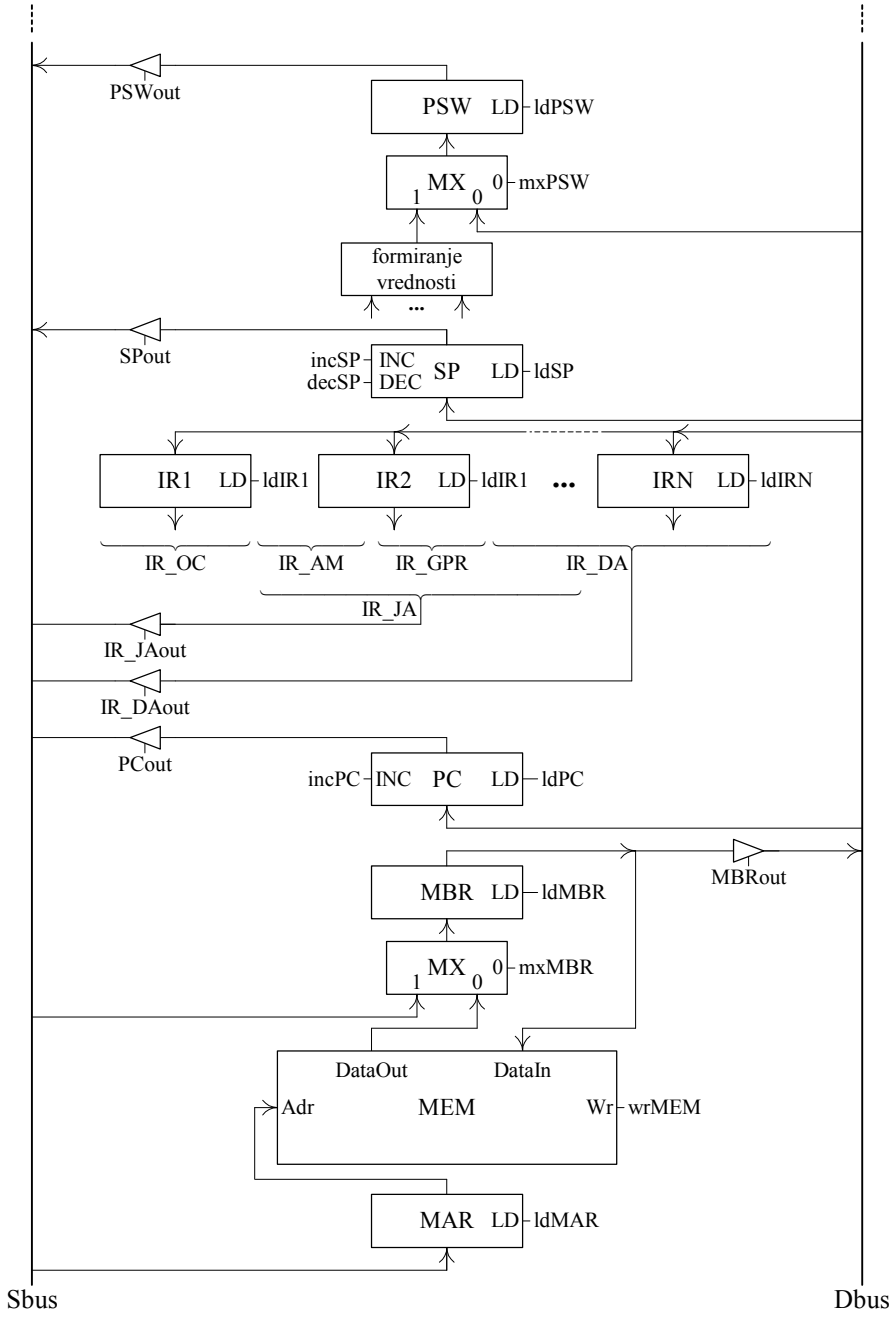
U ovom odeljku se razmatraju struktura upravljačke jedinice i algoritam generisanja upravljačkih signala.

1.2.3.1 Struktura operacione jedinice

Struktura operacione jedinice sa dve magistrale je data na slici 4.



Slika 4.a Operaciona jedinica sa dve magistrale



Slika 4.b Operaciona jedinica sa dve magistrale (nastavak)

1.2.3.2 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija (slika 1) i dat u obliku sekvence upravljačkih signala po koracima (tabela 5). U ovoj tabeli se pojavlju isti iskazi kao i u slučaju tabele 1 za operacionu jedinicu sa direktnim vezama.

Tabela 5 Sekvenca upravljačkih signala po koracima bez spajanja operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **PCout**, **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;
$step_{02}$ **MBRout**, **ldIR1**, **PCout**, **ldMAR**;
$step_{03}$ *br* (*if* **I1** *then* $step_{2E}$);
$step_{04}$ **ldMBR**, **incPC**;

step$_{05}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
step$_{06}$ *br* (*if* **I2** *then* step$_{0C}$);
step$_{07}$ **ldMBR**, **incPC**;
...
step$_{0B}$ **MBRout**, **ldIRN**;

! Formiranje adrese i čitanje operanda !

step$_{0C}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, step$_{0D}$), (**indreg**, step$_{10}$), (**postdec**, step$_{13}$), (**preinc**, step$_{19}$),
(**dirmem**, step$_{1F}$), (**indmem**, step$_{21}$), (**indregpom**, step$_{25}$), (**immed**, step$_{2D}$));

! Direktno registarsko !
step$_{0D}$ **ldRSRC**;
step$_{0E}$ **RSRCout**, **ldB**;
step$_{0F}$ *br* step$_{2E}$;

! Indirektno registarsko !
step$_{10}$ **ldRSRC**;
step$_{11}$ **RSRCout**, **DSout**, **ldMAR**;
step$_{12}$ *br* step$_{29}$;

! Postdekrement !
step$_{13}$ **ldRSRC**;
step$_{14}$ **RSRCout**, **DSout**, **ldMAR**, **ldB**;
step$_{15}$ **decB**;
step$_{16}$ **Bout**, **ldRDST**;
step$_{17}$ **wrGPR**;
step$_{18}$ *br* step$_{29}$;

! Preinkrement !
step$_{19}$ **ldRSRC**;
step$_{1A}$ **RSRCout**, **ldB**;
step$_{1B}$ **incB**;
step$_{1C}$ **Bout**, **ldMAR**, **ldRDST**;
step$_{1D}$ **wrGPR**;
step$_{1E}$ *br* step$_{29}$;

! Direktno memorijsko !
step$_{1F}$ **IR_DAout**, **ldMAR**;
step$_{20}$ *br* step$_{29}$;

! Indirektno memorijsko!
step$_{21}$ **IR_DAout**, **ldMAR**;
step$_{22}$ **ldMBR**;
step$_{23}$ **MBRout**, **DSout**, **ldMAR**;
step$_{24}$ *br* step$_{29}$;

! Indirektno registarsko sa pomerajem !
step$_{25}$ **IR_DAout**, **ldY**, **ldRSRC**;
step$_{26}$ **RSRCout**, **DSout**, **ldX**;
step$_{27}$ **add**, **ldZ**;
step$_{28}$ **Zout**, **DSout**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
step$_{29}$ *br* (*if* **STORE** *then* step$_{2E}$);
step$_{2A}$ **ldMBR**;
step$_{2B}$ **MBRout**, **ldB**;
step$_{2C}$ *br* step$_{2E}$;

! Neposredno !
step$_{2D}$ **IR_DAout**, **SDout**, **ldB**;

! Izvršavanje operacije !

step$_{2E}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, step$_{2F}$), (**STORE**, step$_{31}$),
(**ADD**, step$_{39}$), (**AND**, step$_{3E}$), (**ASR**, step$_{43}$),
(**JZ**, step$_{47}$), (**JMP**, step$_{4C}$), (**JSR**, step$_{49}$), (**RTI**, step$_{4E}$), (**RTS**, step$_{52}$));

! LOAD !

step$_{2F}$ **Bout**, **SDout**, **ldACC**;
step$_{30}$ *br* step$_{56}$;

! STORE !

step$_{31}$ *br* (*if* **immed** *then* step$_{56}$);
step$_{32}$ *br* (*if* **regdir** *then* step$_{36}$);
step$_{33}$ **ACCout**, **mxMBR**, **ldMBR**;
step$_{34}$ **wrMEM**;
step$_{35}$ *br* step$_{56}$;
step$_{36}$ **ACCout**, **ldRDST**;
step$_{37}$ **wrGPR**;
step$_{38}$ *br* step$_{56}$;

! ADD !

step$_{39}$ **ACCout**, **ldX**;
step$_{3A}$ **Bout**, **ldY**;
step$_{3B}$ **add**, **ldZ**;
step$_{3C}$ **Zout**, **ldACC**;
step$_{3D}$ *br* step$_{56}$;

! AND !

step$_{3E}$ **ACCout**, **ldX**;
step$_{3F}$ **Bout**, **ldY**;
step$_{40}$ **and**, **ldZ**;
step$_{41}$ **Zout**, **ldACC**;
step$_{42}$ *br* step$_{56}$;

! ASR !

step$_{43}$ **Bout**, **ldY**;
step$_{44}$ **asr**, **ldZ**;
step$_{45}$ **Zout**, **ldACC**;
step$_{46}$ *br* step$_{56}$;

! JZ !

step$_{47}$ *br* (*if* **eql** *then* step$_{4C}$);
step$_{48}$ *br* step$_{56}$;

! JSR !

step$_{49}$ **SPout**, **ldMAR**, **decSP**;
step$_{4A}$ **PCout**, **mxMBR**, **ldMBR**;
step$_{4B}$ **wrMEM**;

! JMP !

step$_{4C}$ **IR_JAout**, **SDout**, **ldPC**;
step$_{4D}$ *br* step$_{56}$;

! RTI !

step$_{4E}$ **incSP**;
step$_{4F}$ **SPout**, **ldMAR**;
step$_{50}$ **ldMBR**;
step$_{51}$ **MBRout**, **ldPSW**;

! RTS !

step$_{52}$ **incSP**;
step$_{53}$ **SPout**, **ldMAR**;

$step_{54}$ **ldMBR**;
$step_{55}$ **MBRout**, **ldPC**;

! Opsluživanje prekida !

$step_{56}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{57}$ **SPout**, **ldMAR**, **decSP**;
$step_{58}$ **PCout**, **mxMBR**, **ldMBR**;
$step_{59}$ **wrMEM**;
$step_{5A}$ **SPout**, **ldMAR**, **decSP**;
$step_{5B}$ **PSWout**, **mxMBR**, **ldMBR**;
$step_{5C}$ **wrMEM**;
$step_{5D}$ **IVTPout**, **ldX**;
$step_{5E}$ **BRout**, **ldY**;
$step_{5F}$ **add**, **ldZ**;
$step_{60}$ **Zout**, **DSout**, **ldMAR**;
$step_{61}$ **ldMBR**;
$step_{62}$ **MBRout**, **ldPC**;
$step_{63}$ *br* $step_{00}$;

Operacioni korak i upravljački korak se, saglasno razmatranjima za operacionu jedinice sa direktnim vezama, mogu u nekim situacijama mogu spojiti. Tada se dobija sekvenca upravljačkih signala po koracima kao što je dato u tabeli 6. U ovoj tabeli se pojavlju isti iskazi kao i u slučaju tabele 1 za operacionu jedinicu sa direktnim vezama.

Tabela 6 Sekvenca upravljačkih signala po koracima sa spajanjem operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **PCout**, **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;
$step_{02}$ **MBRout**, **ldIR1**, **PCout**, **ldMAR**;
$step_{03}$ *br* (*if* **I1** *then* $step_{27}$);
$step_{04}$ **ldMBR**, **incPC**;
$step_{05}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
$step_{06}$ *br* (*if* **I2** *then* $step_{0C}$);
$step_{07}$ **ldMBR**, **incPC**;
...
$step_{0B}$ **MBRout**, **ldIRN**;

! Formiranje adrese i čitanje operanda !

$step_{0C}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, ste_{0D}), (**indreg**, $step_{0F}$), (**postdec**, $step_{11}$), (**preinc**, $step_{16}$),
(**dirmem**, $step_{1B)}$), (**indmem**, $step_{1C}$), (**indregpom**, $step_{1F}$), (**immed**, $step_{26}$));

! Direktno registarsko !

$step_{0D}$ **ldRSRC**;
$step_{0E}$ **RSRCout**, **ldB**, *br* $step_{27}$;

! Indirektno registarsko !

$step_{0F}$ **ldRSRC**;
$step_{10}$ **RSRCout**, **DSout**, **ldMAR**, *br* $step_{23}$;

! Postdekrement !

$step_{11}$ **ldRSRC**;
$step_{12}$ **RSRCout**, **DSout**, **ldMAR**, **ldB**;
$step_{13}$ **decB**;

step$_{14}$ **Bout**, **ldRDST**;
step$_{15}$ **wrGPR**, *br* step$_{23}$;
! Preinkrement !
step$_{16}$ **ldRSRC**;
step$_{17}$ **RSRCout**, **ldB**;
step$_{18}$ **incB**;
step$_{19}$ **Bout**, **ldMAR**, **ldRDST**;
step$_{1A}$ **wrGPR**, *br* step$_{23}$;
! Direktno memorijsko !
step$_{1B}$ **IR_DAout**, **ldMAR**, *br* step$_{23}$;
! Indirektno memorijsko!
step$_{1C}$ **IR_DAout**, **ldMAR**;
step$_{1D}$ **ldMBR**;
step$_{1E}$ **MBRout**, **DSout**, **ldMAR**, *br* step$_{23}$;
! Indirektno registarsko sa pomerajem !
step$_{1F}$ **IR_DAout**, **ldY**, **ldRSRC**;
step$_{20}$ **RSRCout**, **DSout**, **ldX**;
step$_{21}$ **add**, **ldZ**;
step$_{22}$ **Zout**, **DSout**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
step$_{23}$ *br* (*if* **STORE** *then* step$_{27}$);
step$_{24}$ **ldMBR**;
step$_{25}$ **MBRout**, **ldB**, *br* step$_{27}$;
! Neposredno !
step$_{26}$ **IR_DAout**, **SDout**, **ldB**;

! Izvršavanje operacije !

step$_{27}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, step$_{28}$), (**STORE**, step$_{29}$),
(**ADD**, step$_{2F}$), (**AND**, step$_{33}$), (**ASR**, step$_{37}$),
(**JZ**, step$_{3A}$), (**JMP**, step$_{3F}$), (**JSR**, step$_{3C}$), (**RTI**, step$_{40}$), (**RTS**, step$_{44}$));
! LOAD !
step$_{28}$ **Bout**, **SDout**, **ldACC**, *br* step$_{48}$;
! STORE !
step$_{29}$ *br* (*if* **immed** *then* step$_{48}$);
step$_{2A}$ *br* (*if* **regdir** *then* step$_{2D}$);
step$_{2B}$ **ACCout**, **mxMBR**, **ldMBR**;
step$_{2C}$ **wrMEM**, *br* step$_{48}$;
step$_{2D}$ **ACCout**, **ldRDST**;
step$_{2E}$ **wrGPR**, *br* step$_{48}$;
! ADD !
step$_{2F}$ **ACCout**, **ldX**;
step$_{30}$ **Bout**, **ldY**;
step$_{31}$ **add**, **ldZ**;
step$_{32}$ **Zout**, **ldACC**, *br* step$_{48}$;
! AND !
step$_{33}$ **ACCout**, **ldX**;
step$_{34}$ **Bout**, **ldY**;
step$_{35}$ **and**, **ldZ**;
step$_{36}$ **Zout**, **ldACC**, *br* step$_{48}$;
! ASR !
step$_{37}$ **Bout**, **ldY**;
step$_{38}$ **asr**, **ldZ**;

$step_{39}$ **Zout**, **ldACC**, *br* $step_{48}$;
! JZ !
$step_{3A}$ *br* (*if* **eql** *then* $step_{3F}$);
$step_{3B}$ *br* $step_{48}$;
! JSR !
$step_{3C}$ **SPout**, **ldMAR**, **decSP**;
$step_{3D}$ **PCout**, **mxMBR**, **ldMBR**;
$step_{3E}$ **wrMEM**;
! JMP !
$step_{3F}$ **IR_JAout**, **SDout**, **ldPC**, *br* $step_{48}$;
! RTI !
$step_{40}$ **incSP**;
$step_{41}$ **SPout**, **ldMAR**;
$step_{42}$ **ldMBR**;
$step_{43}$ **MBRout**, **ldPSW**;
! RTS !
$step_{44}$ **incSP**;
$step_{45}$ **SPout**, **ldMAR**;
$step_{46}$ **ldMBR**;
$step_{47}$ **MBRout**, **ldPC**;

! Opsluživanje prekida !

$step_{48}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{49}$ **SPout**, **ldMAR**, **decSP**;
$step_{4A}$ **PCout**, **mxMBR**, **ldMBR**;
$step_{4B}$ **wrMEM**;
$step_{4C}$ **SPout**, **ldMAR**, **decSP**;
$step_{4D}$ **PSWout**, **mxMBR**, **ldMBR**;
$step_{4E}$ **wrMEM**;
$step_{4F}$ **IVTPout**, **ldX**;
$step_{50}$ **BRout**, **ldY**;
$step_{51}$ **add**, **ldZ**;
$step_{52}$ **Zout**, **ldMAR**;
$step_{53}$ **ldMBR**;
$step_{54}$ **MBRout**, **ldPC**, *br* $step_{00}$;

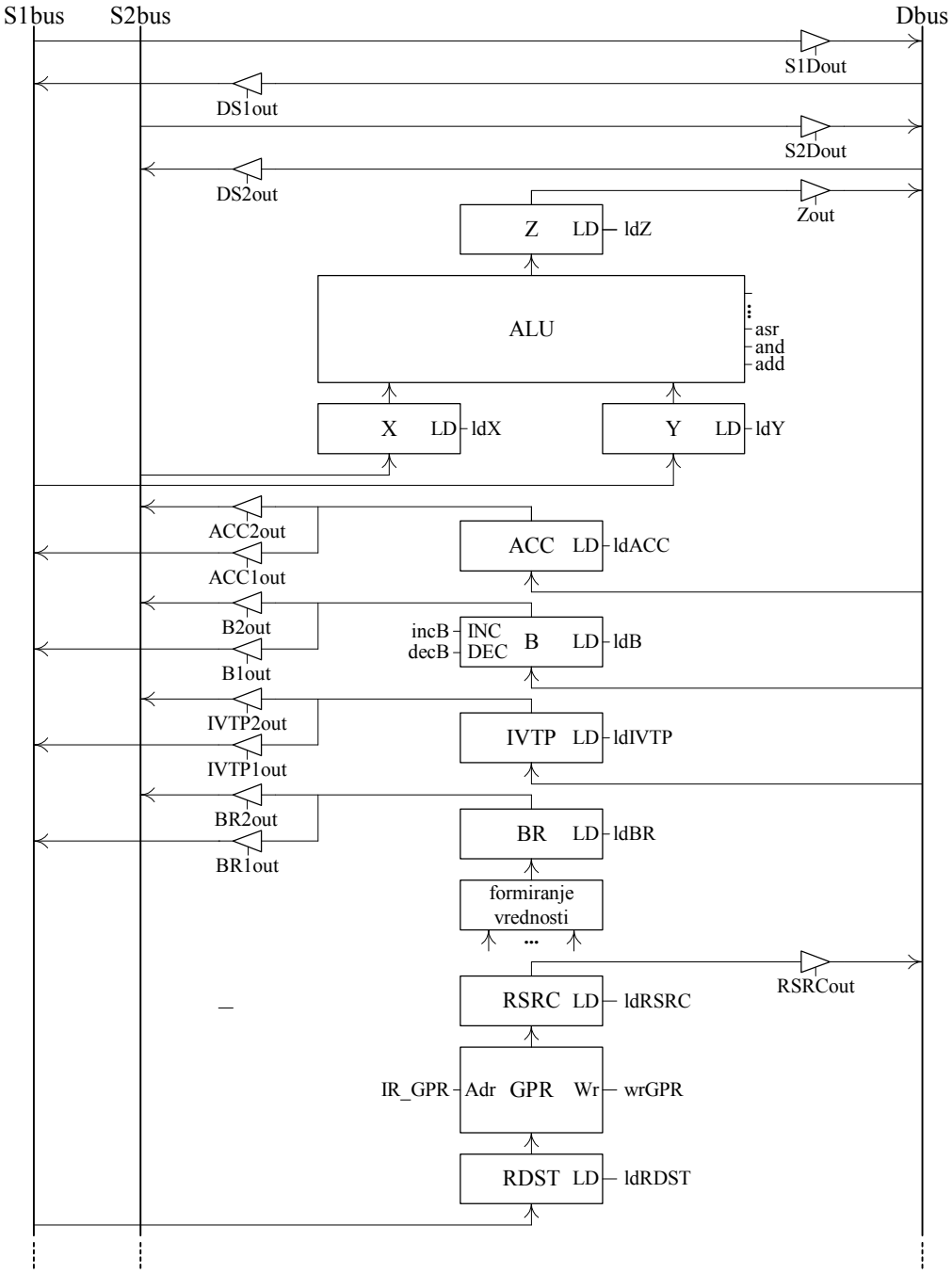
Kao rezultat spajanja koraka tabela 6 ima manji broj koraka od tabele 5. Iz istih razloga su u većini iskaza za skokove promenjene i vrednosti koraka na koje se skače.

1.2.4 OPERACIONA JEDINICA SA TRI MAGISTRALE

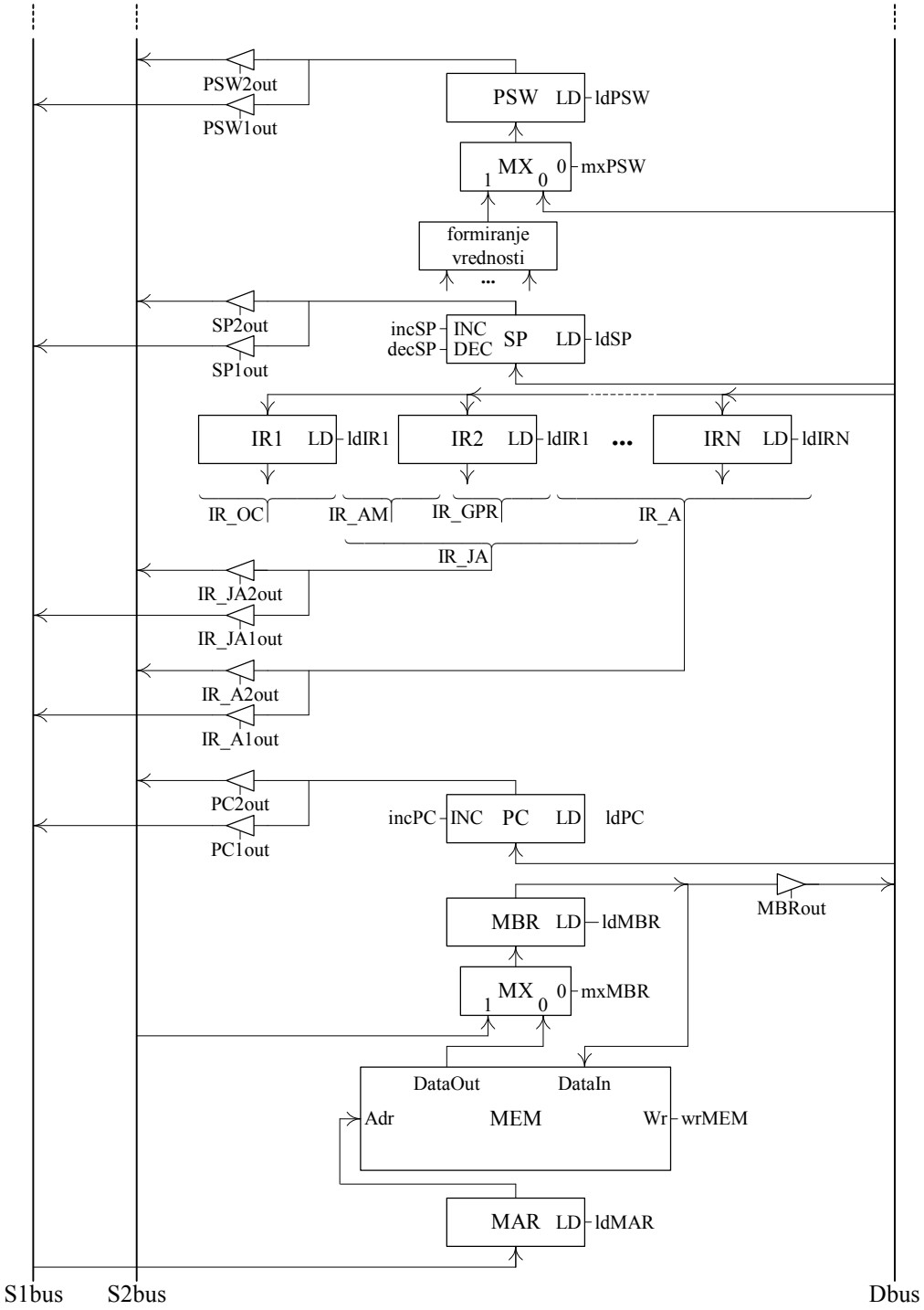
U ovom odeljku se razmatraju struktura upravljačke jedinice i algoritam generisanja upravljačkih signala.

1.2.4.1 Struktura operacione jedinice

Struktura operacione jedinice sa tri magistrale je data na slici 5.



Slika 5.a Operaciona jedinica sa tri magistrale



Slika 5.b Operaciona jedinica sa tri magistrale (nastavak)

1.2.4.2 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija (slika 1) i dat u obliku sekvence upravljačkih signala po koracima (tabela 7). U ovoj tabeli se pojavlju isti iskazi kao i u slučaju tabele 1 za operacionu jedinicu sa direktnim vezama.

Tabela 7 Sekvenca upravljačkih signala po koracima bez spajanja operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **PC1out**, **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;

$step_{02}$ **MBRout**, **ldIR1**, **PC1out**, **ldMAR**;
$step_{03}$ *br* (*if* **l1** *then* $step_{2E}$);
$step_{04}$ **ldMBR**, **incPC**;
$step_{05}$ **MBRout**, **ldIR2**, **PC1out**, **ldMAR**;
$step_{06}$ *br* (*if* **l2** *then* $step_{0C}$);
$step_{07}$ **ldMBR**, **incPC**;
...
$step_{0B}$ **MBRout**, **ldIRN**;

! Formiranje adrese i čitanje operanda !

$step_{0C}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, $step_{0D}$), (**indreg**, $step_{10}$), (**postdec**, $step_{13}$), (**preinc**, $step_{19}$),
(**dirmem**, $step_{1F}$), (**indmem**, $step_{21}$), (**indregpom**, $step_{25}$), (**immed**, $step_{2D}$));

! Direktno registarsko !
$step_{0D}$ **ldRSRC**;
$step_{0E}$ **RSRCout**, **ldB**;
$step_{0F}$ *br* $step_{2E}$;

! Indirektno registarsko !
$step_{10}$ **ldRSRC**;
$step_{11}$ **RSRCout**, **DS1out**, **ldMAR**;
$step_{12}$ *br* $step_{29}$;

! Postdekrement !
$step_{13}$ **ldRSRC**;
$step_{14}$ **RSRCout**, **DS1out**, **ldMAR**, **ldB**;
$step_{15}$ **decB**;
$step_{16}$ **B1out**, **ldRDST**;
$step_{17}$ **wrGPR**;
$step_{18}$ *br* $step_{29}$;

! Preinkrement !
$step_{19}$ **ldRSRC**;
$step_{1A}$ **RSRCout**, **ldB**;
$step_{1B}$ **incB**;
$step_{1C}$ **B1out**, **ldMAR**, **ldRDST**;
$step_{1D}$ **wrGPR**;
$step_{1E}$ *br* $step_{29}$;

! Direktno memorijsko !
$step_{1F}$ **IR_DA1out**, **ldMAR**;
$step_{20}$ *br* $step_{29}$;

! Indirektno memorijsko!
$step_{21}$ **IR_DA1out**, **ldMAR**;
$step_{22}$ **ldMBR**;
$step_{23}$ **MBRout**, **DS1out**, **ldMAR**;
$step_{24}$ *br* $step_{29}$;

! Indirektno registarsko sa pomerajem !
$step_{25}$ **ldRSRC**;
$step_{26}$ **RSRCout**, **DS2out**, **ldX**, **IR_DA1out**, **ldY**;
$step_{27}$ **add**, **ldZ**;
$step_{28}$ **Zout**, **DS1out**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
$step_{29}$ *br* (*if* **STORE** *then* $step_{2E}$);
$step_{2A}$ **ldMBR**;
$step_{2B}$ **MBRout**, **ldB**;

step$_{2C}$ *br* step$_{2E}$;

! Neposredno !

step$_{2D}$ **IR_DA1out**, **S1Dout**, **ldB**;

! Izvršavanje operacije !

step$_{2E}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, step$_{2F}$), (**STORE**, step$_{31}$),
(**ADD**, step$_{39}$), (**AND**, step$_{3D}$), (**ASR**, step$_{41}$),
(**JZ**, step$_{45}$), (**JMP**, step$_{49}$), (**JSR**, step$_{47}$), (**RTI**, step$_{4B}$), (**RTS**, step$_{4F}$));

! LOAD !

step$_{2F}$ **B1out**, **S1Dout**, **ldACC**;
step$_{30}$ *br* step$_{53}$;

! STORE !

step$_{31}$ *br* (*if* **immed** *then* step$_{53}$);
step$_{32}$ *br* (*if* **regdir** *then* step$_{36}$);
step$_{33}$ **ACC2out**, **mxMBR**, **ldMBR**;
step$_{34}$ **wrMEM**;
step$_{35}$ *br* step$_{53}$;
step$_{36}$ **ACC1out**, **ldRDST**;
step$_{37}$ **wrGPR**;
step$_{38}$ *br* step$_{53}$;

! ADD !

step$_{39}$ **ACC2out**, **ldX**, **B1out**, **ldY**;
step$_{3A}$ **add**, **ldZ**;
step$_{3B}$ **Zout**, **ldACC**;
step$_{3C}$ *br* step$_{53}$;

! AND !

step$_{3D}$ **ACC2out**, **ldX**, **B1out**, **ldY**;
step$_{3E}$ **and**, **ldZ**;
step$_{3F}$ **Zout**, **ldACC**;
step$_{40}$ *br* step$_{53}$;

! ASR !

step$_{41}$ **B1out**, **ldY**;
step$_{42}$ **asr**, **ldZ**;
step$_{43}$ **Zout**, **ldACC**;
step$_{44}$ *br* step$_{53}$;

! JZ !

step$_{45}$ *br* (*if* **eql** *then* step$_{49}$);
step$_{46}$ *br* step$_{53}$;

! JSR !

step$_{47}$ **SP1out**, **ldMAR**, **decSP**, **PC2out**, **mxMBR**, **ldMBR**;
step$_{48}$ **wrMEM**;

! JMP !

step$_{49}$ **IR_JA1out**, **S1Dout**, **ldPC**;
step$_{4A}$ *br* step$_{53}$;

! RTI !

step$_{4B}$ **incSP**;
step$_{4C}$ **SP1out**, **ldMAR**;
step$_{4D}$ **ldMBR**;
step$_{4E}$ **MBRout**, **ldPSW**;

! RTS !

step$_{4F}$ **incSP**;
step$_{50}$ **SP1out**, **ldMAR**;

$step_{51}$ **ldMBR**;
$step_{52}$ **MBRout**, **ldPC**;

! Opsluživanje prekida !

$step_{53}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{54}$ **SP1out**, **ldMAR**, **decSP**, **PC2out**, **mxMBR**, **ldMBR**;
$step_{55}$ **wrMEM**;
$step_{56}$ **SP1out**, **ldMAR**, **decSP**, **PSW2out**, **mxMBR**, **ldMBR**;
$step_{57}$ **wrMEM**;
$step_{58}$ **IVTP2out**, **ldX**, **BR1out**, **ldY**;
$step_{59}$ **add**, **ldZ**;
$step_{5A}$ **Zout**, **DS1out**, **ldMAR**;
$step_{5B}$ **ldMBR**;
$step_{5C}$ **MBRout**, **ldPC**;
$step_{5D}$ *br* $step_{00}$;

Operacioni korak i upravljački korak se, saglasno razmatranjima za operacionu jedinice sa direktnim vezama, mogu u nekim situacijama spojiti. Tada se dobija sekvenca upravljačkih signala po koracima kao što je dato u tabeli 8. U ovoj tabeli se pojavlju isti iskazi kao i u slučaju tabele 1 za operacionu jedinicu sa direktnim vezama.

Tabela 8 Sekvenca upravljačkih signala po koracima sa spajanjem operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **PC1out**, **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;
$step_{02}$ **MBRout**, **ldIR1**, **PC1out**, **ldMAR**;
$step_{03}$ *br* (*if* **l1** *then* $step_{27}$);
$step_{04}$ **ldMBR**, **incPC**;
$step_{05}$ **MBRout**, **ldIR2**, **PC1out**, **ldMAR**;
$step_{06}$ *br* (*if* **l2** *then* $step_{0C}$);
$step_{07}$ **ldMBR**, **incPC**;
...
$step_{0B}$ **MBRout**, **ldIRN**;

! Formiranje adrese i čitanje operanda !

$step_{0C}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, ste_{0D}), (**indreg**, $step_{0F}$), (**postdec**, $step_{11}$), (**preinc**, $step_{16}$),
(**dirmem**, $step_{1B}$), (**indmem**, $step_{1C}$), (**indregpom**, $step_{1F}$), (**immed**, $step_{26}$));

! Direktno registarsko !

$step_{0D}$ **ldRSRC**;
$step_{0E}$ **RSRCout**, **ldB**, *br* $step_{27}$;

! Indirektno registarsko !

$step_{0F}$ **ldRSRC**;
$step_{10}$ **RSRCout**, **DS1out**, **ldMAR**, *br* $step_{23}$;

! Postdekrement !

$step_{11}$ **ldRSRC**;
$step_{12}$ **RSRCout**, **DS1out**, **ldMAR**, **ldB**;
$step_{13}$ **decB**;
$step_{14}$ **B1out**, **ldRDST**;
$step_{15}$ **wrGPR**; *br* $step_{23}$;

! Preinkrement !

step$_{16}$ **ldRSRC**;
step$_{17}$ **RSRCout**, **ldB**;
step$_{18}$ **incB**;
step$_{19}$ **B1out**, **ldMAR**, **ldRDST**;
step$_{1A}$ **wrGPR**, *br* step$_{23}$;

! Direktno memorijsko !

step$_{1B}$ **IR_DA1out**, **ldMAR**. *br* step$_{23}$;

! Indirektno memorijsko!

step$_{1C}$ **IR_DA1out**, **ldMAR**;
step$_{1D}$ **ldMBR**;
step$_{1E}$ **MBRout**, **DS1out**, **ldMAR**; *br* step$_{23}$;

! Indirektno registarsko sa pomerajem !

step$_{1F}$ **ldRSRC**;
step$_{20}$ **RSRCout**, **DS2out**, **ldX**, **IR_DA1out**, **ldY**;
step$_{21}$ **add**, **ldZ**;
step$_{22}$ **Zout**, **DS1out**, **ldMAR**;

! Čitanje operanda za memorijska adresiranja !

step$_{23}$ *br* (*if* **STORE** *then* step$_{27}$);
step$_{24}$ **ldMBR**;
step$_{25}$ **MBRout**, **ldB**; *br* step$_{27}$;

! Neposredno !

step$_{26}$ **IR_DA1out**, **S1Dout**, **ldB**;

! Izvršavanje operacije !

step$_{27}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, step$_{28}$), (**STORE**, step$_{29}$),
(**ADD**, step$_{2F}$), (**AND**, step$_{32}$), (**ASR**, step$_{35}$),
(**JZ**, step$_{38}$), (**JMP**, step$_{3C}$), (**JSR**, step$_{3A}$), (**RTI**, step$_{3D}$), (**RTS**, step$_{41}$));

! LOAD !

step$_{28}$ **B1out**, **S1Dout**, **ldACC**, *br* step$_{45}$;

! STORE !

step$_{29}$ *br* (*if* **immed** *then* step$_{45}$);
step$_{2A}$ *br* (*if* **regdir** *then* step$_{2D}$);
step$_{2B}$ **ACC2out**, **mxMBR**, **ldMBR**;
step$_{2C}$ **wrMEM**, *br* step$_{45}$;
step$_{2D}$ **ACC1out**, **ldRDST**;
step$_{2E}$ **wrGPR**, *br* step$_{45}$;

! ADD !

step$_{2F}$ **ACC2out**, **ldX**, **B1out**, **ldY**;
step$_{30}$ **add**, **ldZ**;
step$_{31}$ **Zout**, **ldACC**, *br* step$_{45}$;

! AND !

step$_{32}$ **ACC2out**, **ldX**, **B1out**, **ldY**;
step$_{33}$ **and**, **ldZ**;
step$_{34}$ **Zout**, **ldACC**, *br* step$_{45}$;

! ASR !

step$_{35}$ **B1out**, **ldY**;
step$_{36}$ **asr**, **ldZ**;
step$_{37}$ **Zout**, **ldACC**, *br* step$_{45}$;

! JZ !

step$_{38}$ *br* (*if* **eql** *then* step$_{3C}$);

$step_{39}$ *br* $step_{45}$;

! JSR !

$step_{3A}$ **SP1out**, **ldMAR**, **decSP**, **PC2out**, **mxMBR**, **ldMBR**;

$step_{3B}$ **wrMEM**;

! JMP !

$step_{3C}$ **IR_JA1out**, **S1Dout**, **ldPC**; *br* $step_{45}$;

! RTI !

$step_{3D}$ **incSP**;

$step_{3E}$ **SP1out**, **ldMAR**;

$step_{3F}$ **ldMBR**;

$step_{40}$ **MBRout**, **ldPSW**;

! RTS !

$step_{41}$ **incSP**;

$step_{42}$ **SP1out**, **ldMAR**;

$step_{43}$ **ldMBR**;

$step_{44}$ **MBRout**, **ldPC**;

! Opsluživanje prekida !

$step_{45}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);

$step_{46}$ **SP1out**, **ldMAR**, **decSP**, **PC2out**, **mxMBR**, **ldMBR**;

$step_{47}$ **wrMEM**;

$step_{48}$ **SP1out**, **ldMAR**, **decSP**, **PSW2out**, **mxMBR**, **ldMBR**;

$step_{4C}$ **wrMEM**;

$step_{49}$ **IVTP2out**, **ldX**, **BR1out**, **ldY**;

$step_{4A}$ **add**, **ldZ**;

$step_{4B}$ **Zout**, **DS1out**, **ldMAR**;

$step_{4C}$ **ldMBR**;

$step_{4D}$ **MBRout**, **ldPC**, *br* $step_{00}$;

Kao rezultat spajanja koraka tabela 8 ima manji broj koraka od tabele 7. Iz istih razloga su u većini iskaza za skokove promenjene i vrednosti koraka na koje se skače.

1.3 UPRAVLJAČKA JEDINICA

Upravljačke jedinica se u opštem slučaju realizuje kao sekvencijalna mreža sa onoliko stanja koliko ima koraka u sekvenci upravljačkih signala po koracima. Svakom koraku se dodeljuje posebno stanje. Stanja dodeljena operacionim koracima se koriste za generisanje upravljačkih signala operacione jedinice, a stanja dodeljena upravljačkim koracima se koriste za realizaciju skokova. U zavisnosti od toga kako se stanja sekvencijalne mreže koriste za generisanje upravljačkih signala operacione jedinice i realizaciju skokova u sekvenci upravljačkih signala po koracima, razlikuju se dve osnovne tehnike realizacija upravljačke jedinice i to ožičena realizacija upravljačke jedinice i mikroprogramska realizacija upravljačke jedinice.

U ovom odeljku se razmatraju tehnike ožičene i mikroprogramske realizacije upravljačke jedinice i to za slučaj operacione jedinice sa direktnim vezama. Korišćenje ovih tehnika za operacione jedinice sa jednom, dve i tri magistrale je isto kao i za slučaj operacione jedinice sa direktnim vezama.

1.3.1 Ožičena realizacija

Upravljačka jedinica se sastoji iz brojača koraka, dekodera stanja, kombinacione mreže za generisanje upravljačkih signala i kombinacione mreže za generisanje nove vrednosti brojača

koraka. Posebno stanje brojača koraka se dodeljuje svakom od koraka u sekvenci upravljačkih signala po koracima. Na osnovu vrednosti brojača koraka na izlazima dekodera koraka se dobija aktivna vrednost jednog signala koraka. Kombinaciona mreža za generisanje upravljačkih signala na osnovu signala koraka generiše dve grupe signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice. Upravljački signali operacione jedinice obezbeđuju izvršavanje odgovarajućih mikrooperacija u operacionoj jedinici. Upravljački signali upravljačke jedinice obezbeđuju da se sadržaj brojača koraka ili inkrementira ili da se preko kombinacione mreže za generisanje nove vrednosti brojača koraka generiše nova vrednost i upiše u brojač koraka i time realizuje skok u sekvenci upravljačkih signala po koracima. Upravljački signali se generišu kao unija signala dekodovanih stanja brojača koraka dodeljenih koracima u kojima se odgovarajući upravljački signali operacione jedinice pojavljuju i koracima u kojima upravljački signali upravljačke jedinice treba da realizuju bezuslovne, uslovne i višestruke uslovne skokove.

U ovom odeljku se razmatraju dve tehnike upravljačke jedinice ožičene realizacije i to upravljačka jedinica bez spajanja koraka i upravljačka jedinica sa spajanjem koraka.

1.3.1.1 Upravljačka jedinica bez spajanja koraka

Upravljački signali operacione jedinice se mogu generisati na osnovu sekvence upravljačkih signala po koracima (tabela 1). Za svaki upravljački signal operacione jedinice treba krenuti kroz sekvencu upravljačkih signala po koracima i tražiti korake sa iskazima za signale u kojima se pojavljuje dati signal. Za svaki takav korak treba uzeti signal dekodovanog stanja brojača koraka i formirati njihovu uniju.

Upravljački signali upravljačke jedinice se ne mogu generisati na osnovu sekvence upravljačkih signala po koracima (tabela 1), jer se u njoj ne pojavljuju upravljački signali upravljačke jedinice, već samo iskazi za skokove. Zbog toga je potrebno na osnovu sekvence upravljačkih signala po koracima formirati sekvencu upravljačkih signala za upravljačku jedinicu ožičene realizacije. U njoj treba da se pored upravljačkih signala operacione jedinice pojave i upravljački signali upravljačke jedinice neophodni za realizaciju bezuslovnih, uslovnih i višestruih uslovnih skokova specificiranih iskazima za skokove. Prilikom njenog formiranja primenjuje se različiti postupak za upravljačke signale operacione jedinice i za upravljačke signale upravljačke jedinice.

Za upravljačke signale operacione jedinice treba staviti iskaze za signale onako kako se javljaju u sekvenci upravljačkih signala po koracima.

Za upravljačke signale upravljačke jedinice treba u sekvenci upravljačkih signala po koracima tražiti iskaze: *br* $step_A$, *br* (*if* **uslov** *then* $step_A$) i *br* (*case* (**$uslov_1$**, ..., **$uslov_n$**) *then* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$).

Umesto iskaza *br* $step_A$ treba staviti signal bezuslovnog skoka koji određuje da se bezuslovno prelazi na korak $step_A$ i signal **val_A** koji određuje da treba formirati binarnu vrednost A za upis u brojač koraka. Simbolička oznaka signala bezuslovnog skoka je **bruncnd**. Koraci $step_A$, simboličke oznake signala **val_A** i vrednosti A za sve korake ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima, dati su u tabeli 9.

Tabela 9 Koraci $step_A$, signali **val_A** i vrednosti A za bezuslovne skokove

$step_A$	**val_A**	A
$step_{00}$	**val_{00}**	00
$step_{2C}$	**val_{2C}**	2C
$step_{31}$	**val_{31}**	31

$step_{56}$	**val_{56}**	56

Umesto iskaza *br* (*if* **uslov** *then* $step_A$) treba staviti signal uslovnog skoka koji određuje signal **uslov** koji treba da bude aktivan da bi se realizovao prelaz na korak $step_A$ i signal **val_A** koji određuje da treba formirati binarnu vrednost A za upis u brojač koraka u slučaju da je signal **uslov** aktivan. Simboličke oznake signala uslovnih skokova i signala uslova za sve iskaze ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima, dati su u tabeli 10. Koraci $step_A$, simboličke oznake signala **val_A** i vrednosti A za sve korake ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima dati su u tabeli 11.

Tabela 10 Signali uslovnih skokova i signali uslova

signal uslovnog skoka	signal uslova
brl1	**l1**
brl2	**l2**
brSTORE	**STORE**
brimmed	**immed**
brregdir	**regdir**
breql	**eql**
brnotPREKID	$\overline{\text{PREKID}}$

Tabela 11 Koraci $step_A$, signali **val_A** i vrednosti A za uslovne skokove

$step_A$	**val_A**	A
$step_{00}$	**val_{00}**	00
$step_{0F}$	**val_{0F}**	0F
$step_{31}$	**val_{31}**	31
$step_{39}$	**val_{39}**	39
$step_{4C}$	**val_{4C}**	4C
$step_{56}$	**val_{56}**	56

Umesto iskaza *br* (*case* (**$uslov_1$**, ..., **$uslov_n$**) *then* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$) treba staviti signal višestrukog uslovnog skoka koji određuje signale **$uslov_1$**, **$uslov_2$**,..., **$uslov_n$** od kojih jedan treba da bude aktivan da bi se realizovao prelazak na jedan od koraka $step_{A1}$, $step_{A2}$, ..., $step_{An}$. Simboličke oznake signala višestrukog uslovnog skoka za sve iskaze ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima, date su u tabeli 12. Vrednosti koje treba upisati u brojač koraka i signali uslova koji određuju koju od tih vrednosti treba upisati u brojač koraka za dva iskaza ovog tipa koji se javljaju u koracima $step_{0F}$ i $step_{31}$, dati su u tabelama 13 i 14.

Tabela 12 Signali višestrukih uslovnih skokova

korak	signal višestrukog uslovnog skoka
$step_{0F}$	**bradr**
$step_{31}$	**bropr**

Tabela 13 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{0F}$

signal uslova	vrednost
dirreg	10
indreg	13
postdec	16
preinc	1C
dirmem	22
indmem	24

indregpom	28
immed	30

Tabela 14 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{31}$

signal uslova	vrednost	signal uslova	vrednost
LOAD	32	**JZ**	48
STORE	34	**JMP**	4A
ADD	3C	**JSR**	4C
AND	40	**RTI**	4E
ASR	44	**RTS**	52

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela tabela 1), formirana sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 15). Ona ima sledeću formu: na levoj strani nalaze se dekodovani signali stanja brojača koraka, u sredini je niz upravljačkih signala operacione i upravljačke jedinice koji su aktivni pri datoj vrednosti brojača koraka, dok komentar, u koracima gde se to radi lakšeg razumevanja smatralo korisnim, uvek počinje uskličnikom (!) i proteže se do sledećeg uskličnika (!).

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu ožičene realizacije signal bezuslovnog skoka **bruncnd**, signali uslovnih skokova (tabela 10) i višestrukih uslovnih skokova (tabela 12) i signali $\mathbf{val_A}$ za bezuslovne (tabela 9) i uslovne (tabela 11) skokove. Oni se formiraju na osnovu dobijene sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije na identičan način kao i upravljački signali operacione jedinice.

Tabela 15 Sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije bez spajanja koraka

! Čitanje instrukcije !

T_{00} **ldMAR**, **incPC**;
T_{01} **ldMBR**;
T_{02} **ldIR1**;
T_{03} **brl1**, $\mathbf{val_{31}}$;
T_{04} **ldMAR**, **incPC**;
T_{05} **ldMBR**;
T_{06} **ldIR2**;
T_{07} **brl2**, $\mathbf{val_{0F}}$;
T_{08} **ldMAR**, **incPC**;
...
T_{0E} **ldIRN**;

! Formiranje adrese i čitanje operanda !

T_{0F} **bradr**;

! Direktno registarsko !

T_{10} **ldRSRC**;
T_{11} **ldB**;
T_{12} **bruncnd**, $\mathbf{val_{31}}$;

! Indirektno registarsko !

T_{13} **ldRSRC**;
T_{14} $\mathbf{mxMAR_0}$, **ldMAR**;
T_{15} **bruncnd**, $\mathbf{val_{2C}}$;

! Postdekrement !

T_{16} **ldRSRC**;

T_{17} **mxMAR$_0$**, **ldMAR**, **ldB**;
T_{18} **decB**;
T_{19} **mxRDST**, **ldRDST**;
T_{1A} **wrGPR**;
T_{1B} **bruncnd**, **val$_{2C}$**;

! Preinkrement !

T_{1C} **ldRSRC**;
T_{1D} **ldB**;
T_{1E} **incB**;
T_{1F} **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**, **mxRDST**, **ldRDST**;
T_{20} **wrGPR**;
T_{21} **bruncnd**, **val$_{2C}$**;

! Direktno memorijsko !

T_{22} **mxMAR$_1$**, **ldMAR**;
T_{23} **bruncnd**, **val$_{2C}$**;

! Indirektno memorijsko!

T_{24} **mxMAR$_1$**, **ldMAR**;
T_{25} **ldMBR**;
T_{26} **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**;
T_{27} **bruncnd**, **val$_{2C}$**;

! Indirektno registarsko sa pomerajem !

T_{28} **ldRSRC**;
T_{29} **mxX$_0$**, **ldX**, **mxY$_0$**, **ldY**;
T_{2A} **add**, **ldZ**;
T_{2B} **mxMAR$_2$**, **ldMAR**;

! Čitanje operanda za memorijska adresiranja !

T_{2C} **brSTORE**, val_{31};
T_{2D} **ldMBR**;
T_{2E} **mxB$_0$**, **ldB**;
T_{2F} **bruncnd**, **val$_{31}$**;

! Neposredno !

T_{30} **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

T_{31} **bropr**;

! LOAD !

T_{32} **mxACC**, **ldACC**;
T_{33} **bruncnd**, **val$_{56}$**;

! STORE !

T_{34} **brimmed**, **val$_{56}$**;
T_{35} **brregdir**, **val$_{39}$**;
T_{36} **mxMBR$_0$**, **ldMBR**;
T_{37} **wrMEM**;
T_{38} **bruncnd**, **val$_{56}$**;
T_{39} **ldRDST**;
T_{3A} **wrGPR**;
T_{3B} **bruncnd**, **val$_{56}$**;

! ADD !

T_{3C} **ldX**, **ldY**;
T_{3D} **add**, **ldZ**;
T_{3E} **ldACC**;
T_{3F} **bruncnd**, **val$_{56}$**;

! AND !

T_{40} **ldX**, **ldY**;
T_{41} **and**, **ldZ**;
T_{42} **ldACC**;
T_{43} **bruncnd**, $\mathbf{val_{56}}$;

! ASR !

T_{44} **ldY**;
T_{45} **asr**, **ldZ**;
T_{46} **ldACC**;
T_{47} **bruncnd**, $\mathbf{val_{56}}$;

! JZ !

T_{48} **breql**, $\mathbf{val_{4C}}$;
T_{49} **bruncnd**, $\mathbf{val_{56}}$;

! JSR !

T_{4A} $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**, **decSP**, $\mathbf{mxMBR_1}$, **ldMBR**;
T_{4B} **wrMEM**;

! JMP !

T_{4C} **ldPC**;
T_{4D} **bruncnd**, $\mathbf{val_{56}}$;

! RTI !

T_{4E} **incSP**;
T_{4F} $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**;
T_{50} **ldMBR**;
T_{51} **ldPSW**;

! RTS !

T_{52} **incSP**;
T_{53} $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**;
T_{54} **ldMBR**;
T_{55} **mxPC**, **ldPC**;

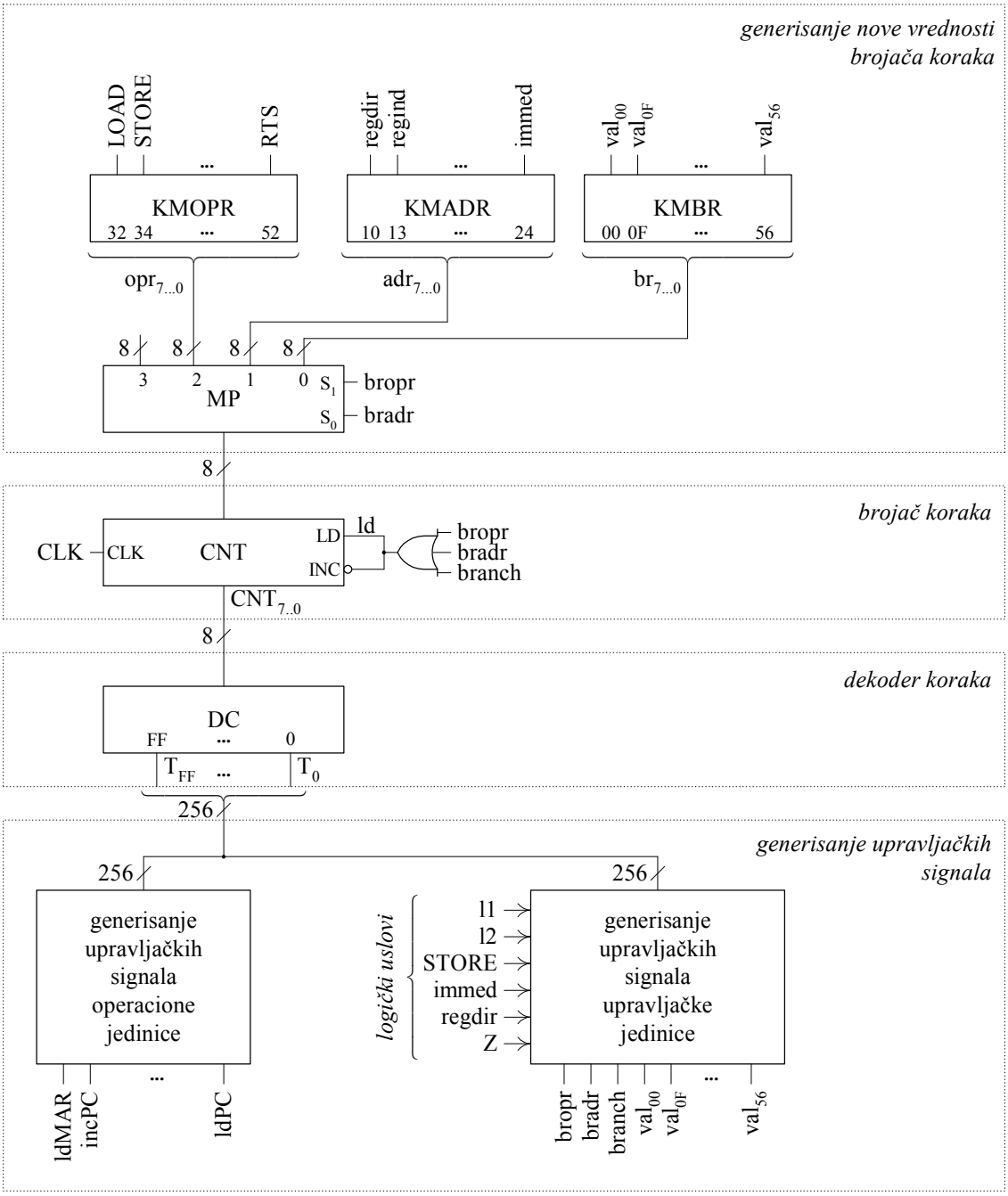
! Opsluživanje prekida !

T_{56} **brnotPREKID**, $\mathbf{val_{00}}$;
T_{57} $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**, **decSP**, $\mathbf{mxMBR_1}$, **ldMBR**;
T_{58} **wrMEM**;
T_{59} $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**, **decSP**, $\mathbf{mxMBR_1}$, $\mathbf{mxMBR_0}$,**ldMBR**;
T_{5A} **wrMEM**;
T_{5B} $\mathbf{mxX_1}$, **ldX**, $\mathbf{mxY_1}$, **ldY**;
T_{5C} **add**, **ldZ**;
T_{5D} $\mathbf{mxMAR_2}$, **ldMAR**;
T_{5E} **ldMBR**;
T_{5F} **mxPC**, **ldPC**;
T_{60} **bruncnd**, $\mathbf{val_{00}}$;

Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 6. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti brojača koraka*, blok *brojač koraka*, blok *dekoder koraka* i blok *generisanje upravljačkih signala*.

Blok *generisanje nove vrednosti brojača koraka* se sastoji od kombinacionih mreža KMOPR, KMADR i KMBR sa multiplekserom MP i služi za generisanje i selekciju vrednosti koju treba upisati u brojač koraka. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikrooperacija. Vrednosti koje treba upisati u brojač koraka generišu se na tri načina i to pomoću: kombinacione mreže KMOPR koja formira signale $\mathbf{opr_{7...0}}$, kombinacione mreže KMADR koja formira signale $\mathbf{adr_{7...0}}$ i kombinacione mreže KMBR koja formira signale $\mathbf{br_{7...0}}$. Selekcija jedne od tri grupe signala koji daju novu

vrednost brojača koraka obezbeđuje se signalima **bropr** i **bradr** i to: signali $\mathbf{opr_{7...0}}$ ako je aktivan signal **bropr**, signali $\mathbf{adr_{7...0}}$ ako je aktivan signal **bradr** i signali $\mathbf{br_{7...0}}$ ako su neaktivni signali **bropr** i **bradr**.



Slika 6 Struktura upravljačke jedinice

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 14) za realizaciju višestrukog uslovnog skoka u koraku $step_{31}$ sekvence upravljačkih signala po koracima. U zavisnosti od toga koji od signala **LOAD**, **STORE**, …, **RTS** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 14 da se pojavi tada na linijama $\mathbf{opr_{7...0}}$. S obzirom da stanje brojača koraka T_{31} daje aktivnu vrednost signala višestrukog uslovnog skoka **bropr**, vrednost na linijama $\mathbf{opr_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima brojača koraka $CNT_{7...0}$.

Kombinacionom mrežom KMADR generišu se vrednosti (tabela 13) za realizaciju višestrukog uslovnog skoka u koraku $step_{0F}$ sekvence upravljačkih signala po koracima. U zavisnosti od toga koji od signala **dirreg**, **indreg**,..., **immed** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 13 da se pojavi tada na linijama $\mathbf{adr_{7...0}}$. S obzirom da stanje brojača koraka T_{0F} daje aktivnu vrednost signala višestrukog uslovnog skoka **bradr**, vrednost na linijama $\mathbf{adr_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima brojača koraka $CNT_{7...0}$.

Kombinacionom mrežom KMBR generišu se vrednosti za upis u brojač koraka za bezuslovne skokove (tabela 9) i uslovne skokove (tabela 11) u sekvenci upravljačkih signala po koracima. U zavisnosti od toga koji od signala $\mathbf{val_{00}}$, $\mathbf{val_{07}}$, ..., $\mathbf{val_{56}}$ ima aktivnu vrednost zavisi koja će od vrednosti iz tabela 9 i 11 tada da se pojavi na linijama $\mathbf{br_{7...0}}$. Signali višestrukih uslovnih skokova **bradr** i **bropr** su aktivni samo u stanjima T_{0F} i T_{31} brojača koraka, a u svim ostalim neaktivni. S obzirom da nijedan od ova dva signala nije aktivan u stanjima brojača koraka kada treba realizovati bezuslovni ili neki od uslovnih skokova, vrednost na linijama $\mathbf{br_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima brojača koraka $CNT_{7...0}$.

Blok *brojač koraka* sadrži brojač $CNT_{7...0}$. Brojač $CNT_{7...0}$ svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač $CNT_{7...0}$ može da radi u sledećim režimima: režim inkrementiranja i režim skoka.

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača $CNT_{7...0}$ za jedan čime se obezbeđuje sekvencijalno generisanje upravljačkih signala iz sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 15). Ovaj režim rada se obezbeđuje neaktivnom vrednošću signala **ld**. Signal **ld** je neaktivan ako su svi signali **bropr**, **bradr** i **branch** neaktivni. Signali **bropr**, **bradr** i **branch** su uvek neaktivni sem kada treba obezbediti režim skoka.

U režimu skoka pri pojavi signala takta vrši se upis nove vrednosti u brojač $CNT_{7...0}$ čime se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 15). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ld**. Signal **ld** je aktivan ako je jedan od signala **bropr**, **bradr** i **branch** aktivan. Jedan od signala **bropr**, **bradr** i **branch** je aktivan samo u stanjima brojača koraka koja se koriste da daju aktivnu vrednost nekog od signala viišestrukog uslovnog skoka, bezuslovnog skoka ili nekog od uslovnih skokova sa ispunjenim uslovom skoka.

Brojač koraka $CNT_{7...0}$ je dimenzionisan prema broju koraka u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 15). S obzirom da se upravljački signali svih faza izvršavanja instrukcija realizuju u opsegu od koraka T_{00} do koraka T_{60} usvojena je dužina brojača koraka $CNT_{7...0}$ od 8 bita.

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača $CNT_{7...0}$. Dekodovana stanja brojača $CNT_{7...0}$ pojavljuju se kao signali $\mathbf{T_0}$, $\mathbf{T_1}$, ..., $\mathbf{T_{FF}}$ na izlazima dekodera DC. Svakom koraku iz sekvence upravljačkih signala po koracima (tabela 1) dodeljeno je po jedno stanje brojača $CNT_{7...0}$ određeno vrednošću signala $\mathbf{T_0}$ do $\mathbf{T_{FF}}$ i to koraku $step_0$ signal $\mathbf{T_0}$, koraku $step_1$ signal $\mathbf{T_1}$, itd. (tabela 15).

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala $\mathbf{T_0}$, $\mathbf{T_1}$, ..., $\mathbf{T_{FF}}$ koji dolaze sa bloka *dekoder koraka*, signala logičkih uslova **l1**, **l2**, ..., **PREKID** koji dolaze iz operacione jedinice i saglasno sekvenci upravljačkih signala za upravljačku

jedinicu ožičene realizacije (tabela 15) generišu dve grupe upravljačkih signala i to: upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice se generišu na sledeći način:

- **ldMAR** = $\mathbf{T_{00}} + \mathbf{T_{04}} + \mathbf{T_{08}} + \mathbf{T_{14}} + \mathbf{T_{17}} + \mathbf{T_{1F}} + \mathbf{T_{22}} + \mathbf{T_{24}} + \mathbf{T_{26}} + \mathbf{T_{2B}} + \mathbf{T_{4A}} + \mathbf{T_{4F}} + \mathbf{T_{53}} + \mathbf{T_{57}} + \mathbf{T_{59}} + \mathbf{T_{5D}}$
- **mxMAR**$_0$ = $\mathbf{T_{14}} + \mathbf{T_{17}} + \mathbf{T_{1F}} + \mathbf{T_{26}} + \mathbf{T_{4A}} + \mathbf{T_{4F}} + \mathbf{T_{53}} + \mathbf{T_{57}} + \mathbf{T_{59}}$
- **wrGPR** = $\mathbf{T_{1A}} + \mathbf{T_{20}} + \mathbf{T_{3A}}$

Na identičan način se generišu i preostali upravljački signali operacione jedinice.

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr** = $\mathbf{T_{31}}$
- **bradr** = $\mathbf{T_{0F}}$
- **branch** = **bruncnd** + **brl1*l1** + **brl2*l2** + **brSTORE*STORE** + **brimmed*immed** + **brregdir*regdir** + **breql*eql**+ **brnotPREKID*** $\overline{\text{PREKID}}$
- **val**$_{00}$ = $\mathbf{T_{56}} + \mathbf{T_{60}}$
- **val**$_{0F}$ = $\mathbf{T_{07}}$
- **val**$_{2C}$ = $\mathbf{T_{15}} + \mathbf{T_{1B}} + \mathbf{T_{21}} + \mathbf{T_{23}} + \mathbf{T_{27}}$
- **val**$_{31}$ = $\mathbf{T_{03}} + \mathbf{T_{12}} + \mathbf{T_{2C}} + \mathbf{T_{2F}}$
- **val**$_{39}$ = $\mathbf{T_{35}}$
- **val**$_{4C}$ = $\mathbf{T_{48}}$
- **val**$_{56}$ = $\mathbf{T_{33}} + \mathbf{T_{34}} + \mathbf{T_{38}} + \mathbf{T_{3B}} + \mathbf{T_{3F}} + \mathbf{T_{43}} + \mathbf{T_{47}} + \mathbf{T_{49}} + \mathbf{T_{4D}}$

Signali koji se javljaju u izrazu za signal **branch** se generišu na sledeći način:

- **bruncnd** = $\mathbf{T_{12}} + \mathbf{T_{15}} + \mathbf{T_{1B}} + \mathbf{T_{21}} + \mathbf{T_{23}} + \mathbf{T_{27}} + \mathbf{T_{2F}} + \mathbf{T_{33}} + \mathbf{T_{38}} + \mathbf{T_{3B}} + \mathbf{T_{3F}} + \mathbf{T_{43}} + \mathbf{T_{47}} + \mathbf{T_{49}} + \mathbf{T_{4D}} + \mathbf{T_{60}}$
- **brl1** = $\mathbf{T_{03}}$
- **brl2** = $\mathbf{T_{07}}$
- **brSTORE** = $\mathbf{T_{2C}}$
- **brimmed** = $\mathbf{T_{34}}$
- **breql** = $\mathbf{T_{48}}$
- **brnotPREKID** = $\mathbf{T_{56}}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **STORE**, **immed**, **regdir**, **eql** i **PREKID**.

1.3.1.2 Upravljačka jedinica sa spajanjem koraka

Upravljačka jedinica sa spajanjem koraka se realizuje istim postupkom kao i upravljačka jedinica bez spajanja koraka. Najpre se na osnovu sekvence upravljačkih signala po koracima sa spajanjem koraka (tabela 2) formira sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije sa spajanjem koraka. Prilikom njenog formiranja primenjuje se različiti postupak za upravljačke signale operacione jedinice i za upravljačke signale upravljačke jedinice.

Za upravljačke signale operacione jedinice treba staviti iskaze za signale onako kako se javljaju u sekvenci upravljačkih signala po koracima.

Za upravljačke signale upravljačke jedinice treba u sekvenci upravljačkih signala po koracima tražiti iskaze: *br* step$_A$, *br* (*if* **uslov** *then* step$_A$) i *br* (*case* (**uslov**$_1$, ..., **uslov**$_n$) *then* (**uslov**$_1$, step$_{A1}$), ..., (**uslov**$_n$, step$_{An}$).

Umesto iskaza *br* $step_A$ treba staviti signal bezuslovnog skoka i signal $\mathbf{val_A}$. Simbolička oznaka signala bezuslovnog skoka je **bruncnd**. Koraci $step_A$ na koje treba bezuslovno preći, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A koje treba upisati u brojač koraka, dati su u tabeli 16.

Tabela 16 Koraci $step_A$, signali $\mathbf{val_A}$ i vrednosti A za bezuslovne skokove

$step_A$	$\mathbf{val_A}$	A
$step_{00}$	$\mathbf{val_{00}}$	00
$step_{26}$	$\mathbf{val_{26}}$	26
$step_{2A}$	$\mathbf{val_{2A}}$	2A
$step_{48}$	$\mathbf{val_{48}}$	48

Umesto iskaza *br* (*if* **uslov** *then* $step_A$) treba staviti signal uslovnog skoka koji određuje signal uslova **uslov** na koji se vrši provera i signal $\mathbf{val_A}$. Simboličke oznake signala uslovnih skokova i signala uslova dati su u tabeli 17. Koraci $step_A$ na koje treba preći ukoliko je signal **uslov** aktivan, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A koje treba tada upisati u brojač koraka, dati su u tabeli 18.

Tabela 17 Signali uslovnih skokova i signali uslova

signal uslovnog skoka	signal uslova
brl1	**l1**
brl2	**l2**
brSTORE	**STORE**
brimmed	**immed**
brregdir	**regdir**
breql	**eql**
brnotPREKID	$\overline{\text{PREKID}}$

Tabela 18 Koraci $step_A$, signali $\mathbf{val_A}$ i vrednosti A za uslovne skokove

$step_A$	$\mathbf{val_A}$	A
$step_{00}$	$\mathbf{val_{00}}$	00
$step_{0F}$	$\mathbf{val_{0F}}$	0F
$step_{2A}$	$\mathbf{val_{2A}}$	2A
$step_{30}$	$\mathbf{val_{30}}$	30
$step_{3F}$	$\mathbf{val_{3F}}$	3F
$step_{48}$	$\mathbf{val_{48}}$	48

Umesto iskaza *br* (*case* ($\mathbf{uslov_1}$, ..., $\mathbf{uslov_n}$) *then* ($\mathbf{uslov_1}$, $step_{A1}$), ..., ($\mathbf{uslov_n}$, $step_{An}$) treba staviti signal višestrukog uslovnog skoka koji određuje signale $\mathbf{uslov_1}$, $\mathbf{uslov_2}$,..., $\mathbf{uslov_n}$ na koje se vrši provera. Simboličke oznake signala višestrukog uslovnog skoka date su u tabeli 19. Signali uslova na koje se vrši provera za dva iskaza ovog i vrednosti koje treba upisati u brojač koraka u zavisnosti od toga koji od signala uslova je aktivan, dati su u tabelama 20 i 21.

Tabela 19 Signali višestrukih uslovnih skokova

korak	signal višestrukog uslovnog skoka
$step_{0F}$	**bradr**
$step_{2A}$	**bropr**

Tabela 20 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{0F}$

signal uslova	vrednost

dirreg	10
indreg	12
postdec	14
preinc	19
dirmem	1E
indmem	1F
indregpom	22
immed	29

Tabela 21 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{2A}$

signal uslova	vrednost	signal uslova	vrednost
LOAD	2B	**JZ**	3B
STORE	2C	**JMP**	3F
ADD	32	**JSR**	3D
AND	35	**RTI**	40
ASR	38	**RTS**	44

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima sa spajanjem koraka (tabela 2), formirana sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije sa spajanjem koraka (tabela 22). Ona ima istu formu kao i tabela 15 za upravljačku jedinicu bez spajanja koraka.

Tabela 22 Sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije sa spajanjem koraka

! Čitanje instrukcije !

T_{00} **ldMAR**, **incPC**;
T_{01} **ldMBR**;
T_{02} **ldIR1**;
T_{03} **brl1**, $\mathbf{val_{2A}}$;
T_{04} **ldMAR**, **incPC**;
T_{05} **ldMBR**;
T_{06} **ldIR2**;
T_{07} **brl2**, $\mathbf{val_{0F}}$;
T_{08} **ldMAR**, **incPC**;
...
T_{0E} **ldIRN**;

! Formiranje adrese i čitanje operanda !

T_{0F} **bradr**;

! Direktno registarsko !

T_{10} **ldRSRC**;
T_{11} **ldB**, **bruncnd**, $\mathbf{val_{2A}}$;

! Indirektno registarsko !

T_{12} **ldRSRC**;
T_{13} $\mathbf{mxMAR_0}$, **ldMAR**, **bruncnd**, $\mathbf{val_{26}}$;

! Postdekrement !

T_{14} **ldRSRC**;
T_{15} $\mathbf{mxMAR_0}$, **ldMAR**, **ldB**;
T_{16} **decB**;
T_{17} **mxRDST**, **ldRDST**;
T_{18} **wrGPR**, **bruncnd**, $\mathbf{val_{26}}$;

! Preinkrement !

T_{19} **ldRSRC**;
T_{1A} **ldB**;
T_{1B} **incB**;
T_{1C} **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**, **mxRDST**, **ldRDST**;
T_{1D} **wrGPR**, **bruncnd**, **val$_{26}$**;

! Direktno memorijsko !

T_{1E} **mxMAR$_1$**, **ldMAR**, **bruncnd**, **val$_{26}$**;

! Indirektno memorijsko!

T_{1F} **mxMAR$_1$**, **ldMAR**;
T_{20} **ldMBR**;
T_{21} **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**, **bruncnd**, **val$_{26}$**;

! Indirektno registarsko sa pomerajem !

T_{22} **ldRSRC**;
T_{23} **mxX$_0$**, **ldX**, **mxY$_0$**, **ldY**;
T_{24} **add**, **ldZ**;
T_{25} **mxMAR$_2$**, **ldMAR**;

! Čitanje operanda za memorijska adresiranja !

T_{26} **brSTORE**, **val$_{2A}$**;
T_{27} **ldMBR**;
T_{28} **mxB$_0$**, **ldB**, **bruncnd**, **val$_{2A}$**;

! Neposredno !

T_{29} **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

T_{2A} **bropr**;

! LOAD !

T_{2B} **mxACC**, **ldACC**, **bruncnd**, **val$_{48}$**;

! STORE !

T_{2C} **brimmed**, **val$_{48}$**;
T_{2D} **brregdir**, **val$_{30}$**;
T_{2E} **mxMBR$_0$**, **ldMBR**;
T_{2F} **wrMEM**, **bruncnd**, **val$_{48}$**;
T_{30} **ldRDST**;
T_{31} **wrGPR**, **bruncnd**, **val$_{48}$**;

! ADD !

T_{32} **ldX**, **ldY**;
T_{33} **add**, **ldZ**;
T_{34} **ldACC**, **bruncnd**, **val$_{48}$**;

! AND !

T_{35} **ldX**, **ldY**;
T_{36} **and**, **ldZ**;
T_{37} **ldACC**, **bruncnd**, **val$_{48}$**;

! ASR !

T_{38} **ldY**;
T_{39} **asr**, **ldZ**;
T_{3A} **ldACC**, **bruncnd**, **val$_{48}$**;

! JZ !

T_{3B} **breql**, **val$_{3F}$**;
T_{3C} **bruncnd**, **val$_{48}$**;

! JSR !

T_{3D} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
T_{3E} **wrMEM**;

! JMP !

T_{3F} **ldPC**, **bruncnd**, **val$_{48}$**;

! RTI !

T_{40} **incSP**;
T_{41} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
T_{42} **ldMBR**;
T_{43} **ldPSW**;

! RTS !

T_{44} **incSP**;
T_{45} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
T_{46} **ldMBR**;
T_{47} **mxPC**, **ldPC**;

! Opsluživanje prekida !

T_{48} **brnotPREKID**, **val$_{00}$**;
T_{49} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
T_{4A} **wrMEM**;
T_{4B} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **mxMBR$_0$**,**ldMBR**;
T_{4C} **wrMEM**;
T_{4D} **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
T_{4E} **add**, **ldZ**;
T_{4F} **mxMAR$_2$**, **ldMAR**;
T_{50} **ldMBR**;
T_{51} **mxPC**, **ldPC**, **bruncnd**, **val$_{00}$**;

Struktura upravljačke jedinice sa spajanjem koraka je ista kao i za slučaj bez spajanja koraka (slika 6). Brojač koraka se na isti način inkrementira i u brojač koraka se na isti način upisuje nova vrednost, pri čemu brojač koraka prolazi kroz manji broj stanja. Stoga je i manji broj signala dekodovanih stanja brojača koraka i to T_{00} do T_{51}. Na isti način se generišu i upravljački signali operacione i upravljačke jedinice jedino se druge vrednosti signala dekodovanih vrednosti stanja brojača koraka koriste. Druge su i vrednosti koje generišu kombinacione mreže KMOPR, KMADR i KMBR. Kombinaciona mreža KMOPR generiše vrednosti 2B, 2C, ..., 44 pri aktivnim vrednostima signala **LOAD**, **STORE**, ..., **RTS**, respektivno. Kombinaciona mreža KMADR generiše vrednosti 10, 12, ..., 29 pri aktivnim vrednostima signala **dirreg**, **indreg**, ..., **immed**, respektivno. Kombinaciona mreža KMBR generiše vrednosti 00, 0F, ..., 56 pri aktivnim vrednostima signala **val$_{00}$**, **val$_{0F}$**, ..., **val$_{56}$**, respektivno. Zbog drugih vrednosti koje generiše kombinaciona mreža KMBR javljaju se drugi signali **val$_A$**. Tako se umesto signala **val$_{31}$**, javlja signal **val$_{2A}$**, jer u brojač koraka umesto vrednosti 31 treba upisati 2A itd.

Istim postupkom kao i u slučaju upravljačke jedinice bez spajanja koraka dobijaju se izrazi za upravljačke signale operacione i upravljačke jedinice.

Upravljački signali operacione jedinice se generišu na sledeći način:

- **ldMAR = T_{00} + T_{04} + T_{08} + T_{13} + T_{15} + T_{1C} + T_{1E} + T_{1F} + T_{21} + T_{25} + T_{3D} + T_{41} + T_{45} + T_{49} + T_{4B} + T_{4F}**
- **mxMAR$_0$ = T_{13} + T_{15} + T_{21} + T_{3D} + T_{41} + T_{45} + T_{49} + T_{4B}**
- **wrGPR = T_{18} + T_{1D} + T_{31}**

Na identičan način se generišu i preostali upravljački signali operacione jedinice.

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr = T_{2A}**
- **bradr = T_{0F}**
- **branch = bruncnd**

$$+ \mathbf{brl1}*\mathbf{l1} + \mathbf{brl2}*\mathbf{l2} + \mathbf{brSTORE}*\mathbf{STORE} + \mathbf{brimmed}*\mathbf{immed}$$
$$+ \mathbf{brregdir}*\mathbf{regdir} + \mathbf{breql}*\mathbf{eql} + \mathbf{brnotPREKID}*\overline{\text{PREKID}}$$

- $\mathbf{val_{00}} = \mathbf{T_{48}} + \mathbf{T_{51}}$
- $\mathbf{val_{0F}} = \mathbf{T_{07}}$
- $\mathbf{val_{26}} = \mathbf{T_{13}} + \mathbf{T_{18}} + {}_{1D} + \mathbf{T_{1E}} + \mathbf{T_{21}}$
- $\mathbf{val_{2A}} = \mathbf{T_{03}} + \mathbf{T_{11}} + \mathbf{T_{26}} + \mathbf{T_{28}}$
- $\mathbf{val_{30}} = \mathbf{T_{2D}}$
- $\mathbf{val_{3F}} = \mathbf{T_{3B}}$
- $\mathbf{val_{48}} = \mathbf{T_{2B}} + \mathbf{T_{2C}} + \mathbf{T_{2F}} + \mathbf{T_{31}} + \mathbf{T_{34}} + \mathbf{T_{37}} + \mathbf{T_{3A}} + \mathbf{T_{3C}} + \mathbf{T_{3F}}$

Signali koji se javljaju u izrazu za signal **branch** se generišu na sledeći način:

- $\mathbf{bruncnd} = \mathbf{T_{11}} + \mathbf{T_{13}} + \mathbf{T_{18}} + \mathbf{T_{1D}} + \mathbf{T_{1E}} + \mathbf{T_{21}} + \mathbf{T_{28}} + \mathbf{T_{2B}} + \mathbf{T_{2F}} + \mathbf{T_{31}} + \mathbf{T_{34}} + \mathbf{T_{37}} + \mathbf{T_{3A}} + \mathbf{T_{3C}} + \mathbf{T_{3F}} + \mathbf{T_{51}}$
- $\mathbf{brl1} = \mathbf{T_{03}}$
- $\mathbf{brl2} = \mathbf{T_{07}}$
- $\mathbf{brSTORE} = \mathbf{T_{26}}$
- $\mathbf{brimmed} = \mathbf{T_{2C}}$
- $\mathbf{brregdir} = \mathbf{T_{2D}}$
- $\mathbf{breql} = \mathbf{T_{3B}}$
- $\mathbf{brnotPREKID} = \mathbf{T_{48}}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **STORE**, **immed**, **regdir**, **eql** i **PREKID**.

1.3.2 Mikroprogramska realizacija

U ovom poglavlju se razmatraju mikroprogramska upravljačka jedinica sa dva tipa mikroinstrukcija, mikroprogramska upravljačka jedinica sa jednim tipom mikroinstrukcija, kodiranje upravljačkih signala operacione jedinice i mikroprogramska upravljačke jedinica sa nanoprogramiranjem.

1.3.2.1 Mikroprogramska realizacija sa dva tipa mikroinstrukcija

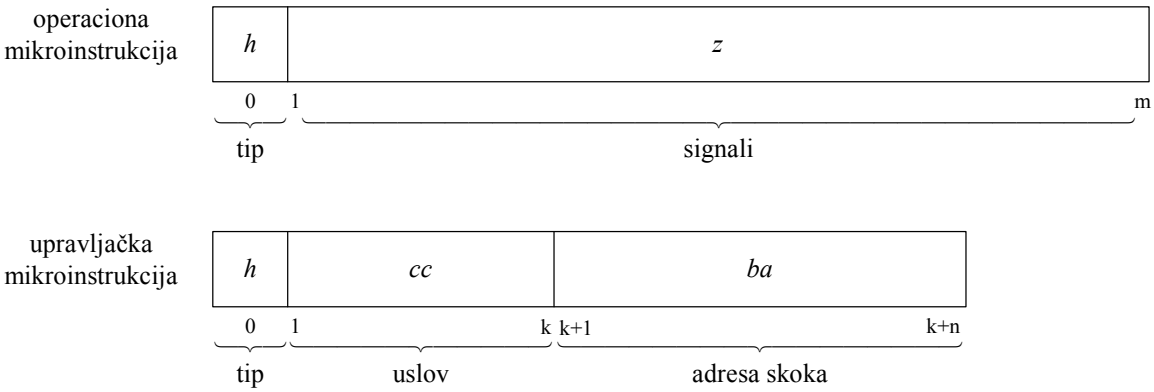
U sekvenci upravljačkih signala po koracima bez spajanja koraka (tabela 1) se svakom operacionom koraku, u kome se generišu upravljački signali operacione jedinice, pridružuje binarna reč čiji je format dat na slici 7 i svakom upravljačkom koraku, u kome se realizuju skokovi, pridružuje binarna reč čiji je format dat na slici 7. Te binarne reči se nazivaju mikroinstrukcijama, mikronaredbama ili mikrokomandama. Mikroinstrukcije pridružene operacionim koracima nazivaju se operacione mikroinstrukcije, dok se mikroinstrukcije pridružene upravljačkim koracima nazivaju upravljačke mikroinstrukcije. Uređeni niz mikroinstrukcija pridruženih operacioinim koracima i upravljačkim koracima, naziva se mikroprogram.

Poljem *h* dužine 1 bit određuje se da li se radi o operacionoj ili upravljačkoj mikroinstrukciji.

Poljem *z* dužine m bita operacione mikroinstrukcije određuju se vrednosti svih upravljačkih signala. Uzeto je da svakom upravljačkom signalu odgovara poseban bit. Ovakav način kodiranja upravljačkih signala se naziva horizontalni način kodiranja.

Poljem *cc* dužine k bita upravljačke mikroinstrukcije specificira se bezuslovni skok, uslovni skokovi na osnovu vrednosti svakog od signala logičkog uslova i višestruki uslovni skokovi. Polje *ba* dužine n bita upravljačke mikroinstrukcije predstalja adresu

mikroinstrukcije u mikroprogramu na koju se skače u slučaju bezuslovnog skoka i u slučaju uslovnog skoka ukoliko ukoliko je vrednost odgovarajućeg signala logičkog uslova aktivna.



Slika 7 Formati operacionih i upravljačkih mikroinstrukcija

Upravljačka jedinice se sastoji iz mikroprogramske memorije, mikroprogramskog brojača, prihvatnog registra mikroinstrukcije, kombinacione mreže za generisanje upravljačkih signala i kombinacione mreže za generisanje nove vrednosti mikroprogramskog brojača. Mikroprogramska memorije služi za smeštanje mikroprograma. Mikroprogramski brojač određuje adresu mikroinstrukcije u mikroprogramskoj memoriji. Prihvatni registar mikroinstrukcije služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije. Kombinaciona mreža za generisanje upravljačkih signala generiše dve grupe signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice. Upravljački signali operacione jedinice se generišu na osnovu vrednosti bitova polja *z* ukoliko je polje *h* 0. Upravljački signali upravljačke jedinice se generišu na osnovu vrednosti bitova polja *cc* i signala logičkih uslova ukoliko je polje *h* 1. Njima se sadržaj mikroprogramskog brojača ili inkrementira ili se u mikroprogramski brojač preko kombinacione mreže za generisanje nove vrednosti mikroprogramskog brojača upisuje vrednost određena poljem *ba* i time realizuje skok u mikroprogramskoj memoriji.

Postoje dva tipa mikroinstrukcije (slika 7) i to operaciona mikroinstrukcija i upravljačka mikroinstrukcija.

Format operacione mikroinstrukcije je dat na slici 8. Polje *h* je 0. Bitovi polja *z* dodeljeni su upravljačkim signalima operacione jedinice.

0	1	2	3	4	5	6	7
0	ldMAR	$mxMAR_2$	$mxMAR_1$	$mxMAR_0$	wrMEM	ldMBR	$mxMBR_1$

8	9	10	11	12	13	14	15
$mxMBR_0$	ldPC	incPC	mxPC	incSP	decSP	$ldIR_1$	$ldIR_2$

16	17	18	19	20	21	22	23
ldACC	mxACC	incB	decB	ldB	mxB_1	mxB_0	ldPSW

24	25	26	27	28	29	30	31
mxPSW	ldX	mxX_1	mxX_0	ldY	mxY_1	mxY_0	add

32	33	34	35	36	37	38	39
and	asr	ldZ	ldRSRC	wrGPR	ldRDST	mxRDST	/

Slika 8 Operaciona mikroinstrukcija

Format upravljačke mikroinstrukcije je dat na slici 9. Polje *h* je 1.

0	1	2	3	4	5	6	7
1	/	/	/	*cc*			

8	9	10	11	12	13	14	15
ba							

16	17	18	19	20	21	22	23
/	/	/	/	/	/	/	/

24	25	26	27	28	29	30	31
/	/	/	/				

32	33	34	35	36	37	38	39
/	/	/	/	/	/	/	/

Slika 9 Upravljačka mikroinstrukcija

Bitovi polja *cc* mikroinstrukcije koriste se za kodiranje upravljačkih signala kojima se određuje da li treba realizovati skok u mikroprogramu i to: bezuslovni skok, uslovni skok i višestruki uslovni skok ili preći na sledeću mikroinstrukciju.

Bezuslovni skok se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 1) u kojima se pojavljuju iskazi tipa *br* $step_A$. Simbolička oznaka signala bezuslovnog skoka koji za svaki od njih treba generisati i način njegovog kodiranja bitovima polja *cc* mikroinstrukcije dati su u tabeli 23.

Tabela 23 Signal bezuslovnog skoka

cc	signal bezuslovnog skoka
01	**bruncnd**

Uslovni skokovi se realizuju u onim koracima sekvence upravljačkih signala po koracima u kojima se pojavljuju iskazi tipa *br* (*if* **uslov** *then* $step_A$). Način kodiranja signala uslovnih skokova bitovima polja *cc* mikroinstrukcije, simboličke oznake signala uslovnih skokova i signal uslova koji treba da je aktivan da bi se realizovao skok dati su u tabeli 24.

Tabela 24 Signali uslovnih skokova

cc	signal uslovnog skoka	signal uslova
02	**brl1**	**l1**
03	**brl2**	**l2**
04	**brSTORE**	**STORE**
05	**brimmed**	**immed**
06	**breql**	**eql**
07	**brnotPREKID**	$\overline{\text{PREKID}}$

Višestruki uslovni skokovi se realizuju u onim koracima sekvence upravljačkih signala po koracima u kojima se pojavljuju iskazi tipa *br* (*case* (**uslov$_1$**, ..., **uslov$_n$**) *then* (**uslov$_1$**, $step_{A1}$), ..., (**uslov$_n$**, $step_{An}$). Način kodiranja signala višestrukih uslovnih skokova bitovima polja *cc* mikroinstrukcije, koraci u sekvenci upravljačkih signala po koracima u kojima se pojavljuju iskazi ovog tipa i simboličke oznake signala višestrukih uslovnih skokova dati su u tabeli 25.

Tabela 25 Signali višestrukih uslovnih skokova

cc	korak	signal višestrukog uslovnog skoka
08	$step_{0F}$	**bradr**
09	$step_{31}$	**bropr**

Vrednost polja *cc* 00 i sve ostale vrednosti koje nisu dodeljene signalu bezuslovnog skoka, signalima uslovnih skokova i signalima višestrukih uslovnih skokova određuje da treba preći na sledeću mikroinstrukciju.

Bitovi *ba* mikroinstrukcije koriste se za specificiranje adrese mikroinstrukcije na koju treba skočiti kod uslovnih i bezuslovnih skokova u sekvenci upravljačkih signala po koracima (tabela 1). Ovi bitovi sadrže vrednost koju treba upisati u mikroprogramski brojač u slučaju bezuslovnih skokova i ukoliko je signal uslova aktivan u slučaju uslovnih skokova. Kod pisanja mikroprograma ovo polje se simbolički označava sa $madr_{xx}$, pri čemu xx odgovara heksadekadnoj vrednosti ovog polja. Na primer, sa $madr_{56}$ je simbolički označena heksadekadna vrednost 56 ovog polja. Za kodiranje polja *adresa skoka* usvojeno je 8 bitova, jer je za kompletan mikroprogram dovoljan kapacitet mikroprogramske memorije od 256 reči.

Operacioina mikroinstrukcija je duža od upravljačke mikroinstrukcije, pa je dužina mikroinstrukcije određena dužinom operacione mikroinstrukcije i iznosi 40 bitova.

Mikroprogram se za razmatrani slučaj mikroprogramske realizacije formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima (tabela 1) formira jedna mikroinstrukcija i to operaciona ili upravljačka.

Kod formiranja operacionih mikroinstrukcija polazi se od sekvence upravljačkih signala po koracima i traže koraci u kojima se javljaju upravljački signali operacione jedinice. Za takve korake se bit polja *h* postavlja na 0, bitovi polja *z* koji odgovaraju upravljačkim signalima operacione koji se javljaju u datom koraku postavljaju na 1 i bitovi polja *z* koji odgovaraju upravljačkim signalima operacione koji se ne javljaju u datom koraku postavljaju na 0.

Kod formiranja upravljačkih mikroinstrukcija polazi se od sekvence upravljačkih signala po koracima i traže koraci u kojima se javlja neki od iskaza *br* $step_A$, *br* (*if* **uslov** *then* $step_A$) i *br* (*case* (**$uslov_1$**, ..., **$uslov_n$**) *then* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$)). Za takve korake se bit polja *h* postavlja na 1, što se u mikroprogramu označava signalom **cnt**, dok se bitovi polja *cc* i *ba* kodiraju u zavisnosti od toga koji se od ova tri iskaza javlja u datom koraku.

Za iskaz *br* $step_A$ se upravljačka mikroinstrukcija kodira tako što se za polje *cc* uzima kod dodeljen signalu bezuslovnog skoka koji određuje da se bezuslovno prelazi na korak $step_A$ i za polje *ba* binarna vrednosti A koju treba upisati u mikroprogramski brojač.

Simbolička oznaka signala bezuslovnog skoka i način njegovog kodiranja poljem *cc* dati su u tabeli 23. Korak $step_A$ na koji treba preći u sekvenci upravljačkih signala po koracima, simbolička oznaka vrednosti $madr_A$ koju treba upisati u mikroprogramski brojač i sama vrednost A za sve korake u sekvenci upravljačkih signala po koracima u kojima se javljaju iskazi ovog tipa dati su u tabeli 26.

Tabela 26 Koraci $step_A$, adrese $madr_A$ i vrednosti A za bezuslovne skokove

$step_A$	$madr_A$	A
$step_{00}$	$madr_{00}$	00
$step_{2C}$	$madr_{2C}$	2C
$step_{31}$	$madr_{31}$	31
$step_{56}$	$madr_{56}$	56

Za iskaz *br* (*if* **uslov** *then* $step_A$) se upravljačka mikroinstrukcija kodira tako što se za polje *cc* uzima kod dodeljen signalu uslovnog skoka koji određuje signal **uslov** koji treba da bude aktivan da bi se realizovao prelaz na korak $step_A$ i za polje *bb* binarna vrednosti A koju treba upisati u mikroprogramski brojač u slučaju da je signal **uslov** aktivan.

Simboličke oznake signala uslovnog skoka, način njihovog kodiranja poljem *cc* i signali **uslov** za sve iskaze ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima dati su u tabeli 24. Korak $step_A$ na koji treba preći u sekvenci upravljačkih signala po koracima, simbolička oznaka vrednosti $madr_A$ koju treba upisati u mikroprogramski brojač u slučaju da je signal **uslov** aktivan i sama vrednost A za sve korake u sekvenci upravljačkih signala po koracima u kojima se javljaju iskazi ovog tipa dati su u tabeli 27.

Tabela 27 Koraci $step_A$, adrese $madr_A$ i vrednosti A za uslovne skokove

$step_A$	$madr_A$	A
$step_{00}$	$madr_{00}$	00
$step_{0F}$	$madr_{0F}$	0F
$step_{31}$	$madr_{31}$	31
$step_{39}$	$madr_{39}$	39
$step_{56}$	$madr_{56}$	56

Za iskaz *br* (*case* (**uslov$_1$**, ..., **uslov$_n$**) *then* (**uslov$_1$**, $step_{A1}$), ..., (**uslov$_n$**, $step_{An}$)) se upravljačka mikroinstrukcija kodira tako što se za polje *cc* uzima kod dodeljen signalu višestrukog uslovnog skoka koji određuje signale **uslov$_1$**, ..., **uslov$_n$** za koje treba izvršiti proveru koji je od njih aktivan da bi se na osnovu toga realizovao prelaz na jedan od koraka $step_{A1}$, ..., $step_{An}$ i za polje *bb* nule jer njegova vrednost nije bitna. Upravljačka jedinica mora da bude tako realizovana da za svaki višestruki uslovni skok generiše vrednosti A1,..., An koje treba upisati u mikroprogramski brojač. Ona mora da obezbedi i selekciju jedne od vrednosti A1,..., An u zavisnosti od toga koji od signala uslova **uslov$_1$**, ..., **uslov$_n$** ima aktivnu vrednost.

Simboličke oznake signala višestrukih uslovnih skokova, način njihovog kodiranja poljem *cc* i koraci u sekvenci upravljačkih signala po koracima u kojima se javljaju iskazi ovog tipa dati su u tabeli 25. Vrednosti A1,..., An koje treba upisati u mikroprogramski brojač i signali uslova **uslov$_1$**, ..., **uslov$_n$** za koje treba izvršiti proveru koji je od njih aktivan da bi se na osnovu toga realizovao prelaz na jedan od koraka $step_{A1}$, ..., $step_{An}$ za dva iskaza ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima dati su u tabelama 28 i 29.

Tabela 28 Signali uslova i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{0F}$

signal uslova	vrednost
dirreg	10
indreg	13
postdec	16
preinc	1C
dirmem	22
indmem	24
indregpom	28
immed	30

Tabela 29 Signali uslova i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{31}$

signal uslova	vrednost	signal uslova	vrednost
LOAD	32	**JZ**	48
STORE	34	**JMP**	4A
ADD	3C	**JSR**	4C
AND	40	**RTI**	4E
ASR	44	**RTS**	52

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 1) formiran mikroprogram (tabela 30). On ima sledeću formu:

- na levoj strani se nalaze adrese mikroinstrukcija u mikroprogramskoj memoriji u heksadekadnom obliku,
- u sredini su mikroinstrukcije predstavljene nizom simboličkih oznaka samo upravljačkih signala operacione i/ili upravljačke jedinice koji treba da budu aktivni i koji su razdvojeni zapetama,
- dok komentar, u koracima gde se to radi lakšeg razumevanja smatralo korisnim, uvek počinje uskličnikom (!) i proteže se do sledećeg uskličnika (!).

Tabela 30 Mikroprogram

! Čitanje instrukcije !

$madr_{00}$ **ldMAR**, **incPC**;
$madr_{01}$ **ldMBR**;
$madr_{02}$ **ldIR1**;
$madr_{03}$ **cnt**, **brl1**, **madr$_{31}$**;
$madr_{04}$ **ldMAR**, **incPC**;
$madr_{05}$ **ldMBR**;
$madr_{06}$ **ldIR2**;
$madr_{07}$ **cnt**, **brl2**, **madr$_{0F}$**;
$madr_{08}$ **ldMAR**, **incPC**;
...
$madr_{0E}$ **ldIRN**;

! Formiranje adrese i čitanje operanda !

$madr_{0F}$ **cnt**, **bradr**;

! Direktno registarsko !

$madr_{10}$ **ldRSRC**;
$madr_{11}$ **ldB**;
$madr_{12}$ **cnt**, **bruncnd**, **madr$_{31}$**;

! Indirektno registarsko !

$madr_{13}$ **ldRSRC**;
$madr_{14}$ **mxMAR$_0$**, **ldMAR**;
$madr_{15}$ **cnt**, **bruncnd**, **madr$_{2C}$**;

! Postdekrement !

$madr_{16}$ **ldRSRC**;
$madr_{17}$ **mxMAR$_0$**, **ldMAR**, **ldB**;
$madr_{18}$ **decB**;
$madr_{19}$ **mxRDST**, **ldRDST**;
$madr_{1A}$ **wrGPR**;
$madr_{1B}$ **cnt**, **bruncnd**, **madr$_{2C}$**;

! Preinkrement !

$madr_{1C}$ **ldRSRC**;
$madr_{1D}$ **ldB**;
$madr_{1E}$ **incB**;
$madr_{1F}$ **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**, **mxRDST**, **ldRDST**;
$madr_{20}$ **wrGPR**;
$madr_{21}$ **cnt**, **bruncnd**, **madr$_{2C}$**;

! Direktno memorijsko !

$madr_{22}$ **mxMAR$_1$**, **ldMAR**;
$madr_{23}$ **cnt**, **bruncnd**, **madr$_{2C}$**;

! Indirektno memorijsko!

madr$_{24}$ **mxMAR$_1$**, **ldMAR**;
madr$_{25}$ **ldMBR**;
madr$_{26}$ **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**;
madr$_{27}$ **cnt**, **bruncnd**, **madr$_{2C}$**;

! Indirektno registarsko sa pomerajem !

madr$_{28}$ **ldRSRC**;
madr$_{29}$ **mxX$_0$**, **ldX**, **mxY$_0$**, **ldY**;
madr$_{2A}$ **add**, **ldZ**;
madr$_{2B}$ **mxMAR$_2$**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
madr$_{2C}$ **brSTORE**, **madr$_{31}$**;
madr$_{2D}$ **ldMBR**;
madr$_{2E}$ **mxB$_0$**, **ldB**;
madr$_{2F}$ **cnt**, **bruncnd**, **madr$_{31}$**;

! Neposredno !

madr$_{30}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

madr$_{31}$ **cnt**, **bropr**;

! LOAD !

madr$_{32}$ **mxACC**, **ldACC**;
madr$_{33}$ **cnt**, **bruncnd**, **madr$_{56}$**;

! STORE !

madr$_{34}$ **cnt**, **brimmed**, **madr$_{56}$**;
madr$_{35}$ **cnt**, **brregdir**, **madr$_{39}$**;
madr$_{36}$ **mxMBR$_0$**, **ldMBR**;
madr$_{37}$ **wrMEM**;
madr$_{38}$ **cnt**, **bruncnd**, **madr$_{56}$**;
madr$_{39}$ **ldRDST**;
madr$_{3A}$ **wrGPR**;
madr$_{3B}$ **cnt**, **bruncnd**, **madr$_{56}$**;

! ADD !

madr$_{3C}$ **ldX**, **ldY**;
madr$_{3D}$ **add**, **ldZ**;
madr$_{3E}$ **ldACC**;
madr$_{3F}$ **cnt**, **bruncnd**, **madr$_{56}$**;

! AND !

madr$_{40}$ **ldX**, **ldY**;
madr$_{41}$ **and**, **ldZ**;
madr$_{42}$ **ldACC**;
madr$_{43}$ **cnt**, **bruncnd**, **madr$_{56}$**;

! ASR !

madr$_{44}$ **ldY**;
madr$_{45}$ **asr**, **ldZ**;
madr$_{46}$ **ldACC**;
madr$_{47}$ **cnt**, **bruncnd**, **madr$_{56}$**;

! JZ !

madr$_{48}$ **cnt**, **breql**, **madr$_{4C}$**;
madr$_{49}$ **cnt**, **bruncnd**, **madr$_{56}$**;

! JSR !

madr$_{4A}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
madr$_{4B}$ **wrMEM**;

! JMP !

$madr_{4C}$ **ldPC**;
$madr_{4D}$ **cnt**, **bruncnd**, **madr$_{56}$**;

! RTI !

$madr_{4E}$ **incSP**;
$madr_{4F}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$madr_{50}$ **ldMBR**;
$madr_{51}$ **ldPSW**;

! RTS !

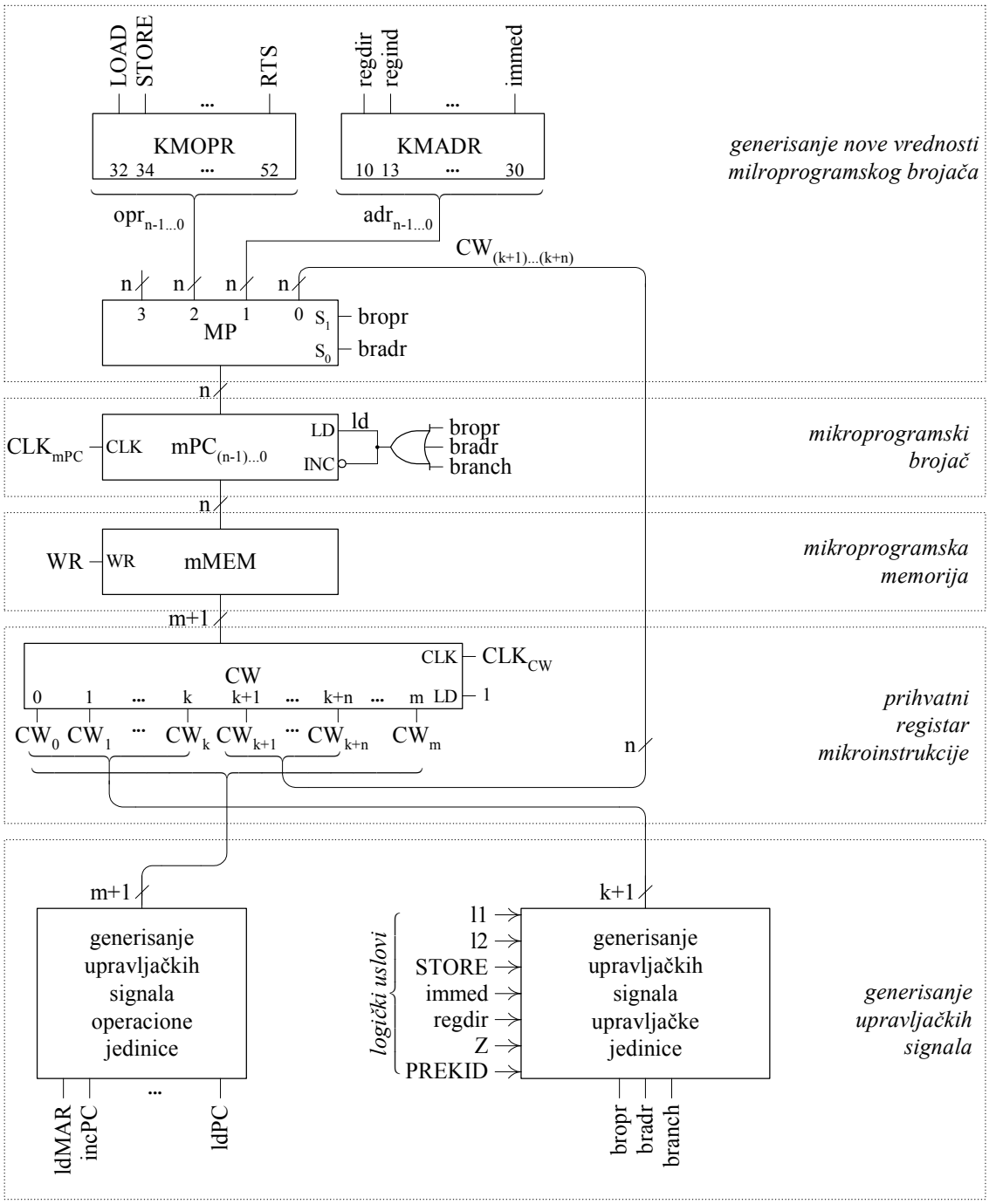
$madr_{52}$ **incSP**;
$madr_{53}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$madr_{54}$ **ldMBR**;
$madr_{55}$ **mxPC**, **ldPC**;

! Opsluživanje prekida !

$madr_{56}$ **cnt**, **brnotPREKID**, **madr$_{00}$**;
$madr_{57}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
$madr_{58}$ **wrMEM**;
$madr_{59}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **mxMBR$_0$**,**ldMBR**;
$madr_{5A}$ **wrMEM**;
$madr_{5B}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
$madr_{5C}$ **add**, **ldZ**;
$madr_{5D}$ **mxMAR$_2$**, **ldMAR**;
$madr_{5E}$ **ldMBR**;
$madr_{5F}$ **mxPC**, **ldPC**;
$madr_{60}$ **cnt**, **bruncnd**, **madr$_{00}$**;

Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 10. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti mikroprogramskog brojača*, blok *mikroprogramski brojač*, blok *mikroprogramska memorija*, blok *prihvatni registar mikroinstrukcije* i blok *generisanje upravljačkih signala*.

Blok *generisanje nove vrednosti mikroprogramskog brojača* se sastoji od kombinacionih mreža KMOPR i KMADR sa multiplekserom MP i služi za generisanje i selekciju vrednosti koju treba upisati u mikroprogramski brojač. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikroprograma. Vrednosti koje treba upisati u mikroprogramski brojač generišu se na tri načina i to pomoću: kombinacione mreže KMOPR koja formira signale **opr$_{7...0}$**, kombinacione mreže KMADR koja formira signale **adr$_{7...0}$** i razreda $CW_{k+1...k+n}$ prihvatnog registra mikroinstrukcije CW. Selekcija jedne od tri grupe signala koje daju novu vrednost mikroprogramskog brojača obezbeđuje se signalima **brop** i **bradr** i to: signali **opr$_{7...0}$** ako je aktivan signal **bropr**, signali **adr$_{7...0}$** ako je aktivan signal **bradr** i signali **CW$_{k+1...k+n}$** ako su neaktivni signali **bropr** i **bradr**.



Slika 10 Struktura upravljačke jedinice mikroprogramske realizacije sa horizontalnim formatom mikroinstrukcije

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 29) za realizaciju višestrukog uslovnog skoka na adresi 31 mikroprograma (tabela 30). U zavisnosti od toga koji od signala **LOAD**, **STORE**, …, **RTS** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 29 da se pojavi na linijama $\mathbf{opr_{7...0}}$. S obzirom da se na adresi 31 mikroprograma nalazi upravljačka mikroinstrukcija sa tako kodiranim poljem *cc* da njeno izvršavanje daje aktivnu vrednost signala višestrukog uslovnog skoka **bropr**, vrednost na linijama $\mathbf{opr_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača mPC.

Kombinacionom mrežom KMADR generišu se vrednosti (tabela 28) za realizaciju višestrukog uslovnog skoka na adresi 0F mikroprograma (tabela 30). U zavisnosti od toga koji

od signala **dirreg**, **indreg**,..., **immed** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 28 da se pojavi tada na linijama $\mathbf{adr_{7...0}}$. S obzirom da se na adresi 0F mikroprograma nalazi mikroinstrukcija sa tako kodiranim poljem *cc* da njeno izvršavanje daje aktivnu vrednost signala višestrukog uslovnog skoka **bradr**, vrednost na linijama $\mathbf{adr_{7...0}}$ prolazi kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača mPC.

Prihvatni registar mikroinstrukcije CW u svojim razredima $CW_{k+1...k+n}$ sadrži vrednost za upis u mikroprogramski brojač $mPC_{7...0}$ za bezuslovne skokove (tabela 26) i uslovne skokove (tabela 27). Signali višestrukih uslovnin skokova **bropr** i **bradr** su aktivni samo prilikom izvršavanja mikroinstrukcija na adresama 0F i 31 mikroprograma, respektivno, a u svim ostalim situacijama neaktivni. S obzirom da nijedan od ova dva signala nije aktivan prilikom izvršavanja mikroinstrukcija kojima se realizuju bezuslovni ili uslovni skokovi u mikroprogramu, vrednost određena razredima $CW_{k+1...k+n}$ prolazi kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača $mPC_{7...0}$.

Blok *mikroprogramski brojač* sadrži mikroprogramski brojač $mPC_{n-1...0}$. Mikroprogramski brojač $mPC_{n-1...0}$ svojom trenutnom vrednošću određuje adresu mikroprogramske memorije mMEM sa koje treba očitati mikroinstrukciju. Mikroprogramski brojač $mPC_{n-1...0}$ može da radi u sledećim režimima: režim inkrementiranja i režim skoka.

U režimu inkrementiranja pri pojavi signala takta $\mathbf{CLK_{mPC}}$ vrši se uvećavanje sadržaja mikroprogramskog brojača $mPC_{n-1...0}$ za jedan čime se obezbeđuje sekvencijalno očitavanje mikroinstrukcija iz mikroprogramske memorije (tabela 30). Ovaj režim rada se obezbeđuje neaktivnom vrednošću signala **ld**. Signal **ld** je neaktivan ako su svi signali **bropr**, **bradr** i **branch** neaktivni. Signali **bropr**, **bradr** i **branch** su uvek neaktivni sem kada treba obezbediti režim skoka.

U režimu skoka pri pojavi signala takta $\mathbf{CLK_{mPC}}$ vrši se upis nove vrednosti u mikroprogramski brojač $mPC_{n-1...0}$ čime se obezbeđuje odstupanje od sekvencijalnog očitavanja mikroinstrukcija iz mikroprogramske memorije (tabela 30). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ld**. Signal **ld** je aktivan ako je jedan od signala **bropr**, **bradr** i **branch** aktivan. Jedan od signala **bropr**, **bradr** i **branch** je aktivan samo prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran neki višestruki uslovni skok, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen.

Mikroprogramski brojač $mPC_{n-1...0}$ je dimenzionisan prema veličini mikroprograma (tabela 30). S obzirom da se mikroprogram svih faza izvršavanja instrukcija nalazi u opsegu od adrese 00 do adrese 60, usvojena je dužina mikroprogramskog brojača $mPC_{n-1...0}$ od 8 bita.

Blok *mikroprogramski memorija* sadrži mikroprogramsku memoriju mMEM, koja služi za smeštanje mikroprograma. Širina reči mikroprogramske memorije je određena dužinom mikroinstrukcija i iznosi 40 bita, a kapacitet veličinom mikroprograma svih instrukcija procesora (tabela 30) i iznosi 256 lokacija. Adresiranje mikroprogramske memorije se realizuje sadržajem mikroprogramskog brojača $mPC_{n-1...0}$.

Blok *prihvatni registar mikroinstrukcije* sadrži prihvatni registar mikroinstrukcije $CW_{0...k+n}$. Prihvatni registar mikroinstrukcije $CW_{0...k+n}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...m}$ i $CW_{0...k}$ se koriste u bloku *generisanje upravljačkih signala* za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{k+1...k+n}$ koriste u bloku *generisanje nove vrednosti mikroprogramskog brojača* kao adresa skoka u mikrorogramu u slučaju bezuslovnih i uslovnih skokova. Upis u ovaj registar se realizuje signalom takta **CLK**. Signal takta **CLK**

kasni za signalom takta $\mathbf{CLK_{mPC}}$ onoliko koliko je potrebno da se pročita sadržaj sa odgovarajuće adrese mikroprogramske memorije.

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje na osnovu sadržaja razreda $CW_{0...m}$ prihvatnog registra mikroinstrukcije generišu upravljačke signale operacione jedinice i na osnovu sadržaja razreda $CW_{0...k}$ prihvatnog registra mikroinstrukcije i signala logičkih uslova **l1**, **l2**, ..., **PREKID** koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice se generišu na sledeći način:

- $\mathbf{ldMAR} = \overline{\mathbf{CW}}_0 \cdot \mathbf{CW_1}$
- $\mathbf{mxMAR_0} = \overline{\mathbf{CW}}_0 \cdot \mathbf{CW_4}$
- $\mathbf{wrGPR} = \overline{\mathbf{CW}}_0 \cdot \mathbf{CW_{36}}$

Na identičan način se generišu i preostali upravljački signali operacione jedinice.

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- $\mathbf{bropr} = \mathbf{CW_0 \cdot CW_4 \cdot} \overline{\mathbf{CW}}_5 \cdot \overline{\mathbf{CW}}_6 \cdot \mathbf{CW_7}$
- $\mathbf{bradr} = \mathbf{CW_0 \cdot CW_4 \cdot} \overline{\mathbf{CW}}_5 \cdot \overline{\mathbf{CW}}_6 \cdot \overline{\mathbf{CW}}_7$
- $\mathbf{branch} = \mathbf{bruncnd} + \mathbf{brl1 * l1} + \mathbf{brl2 * l2} + \mathbf{brSTORE * STORE} + \mathbf{brimmed * immed} + \mathbf{brregdir * regdir} + \mathbf{breql * eql} + \mathbf{brnotPREKID *} \overline{\mathrm{PREKID}}$

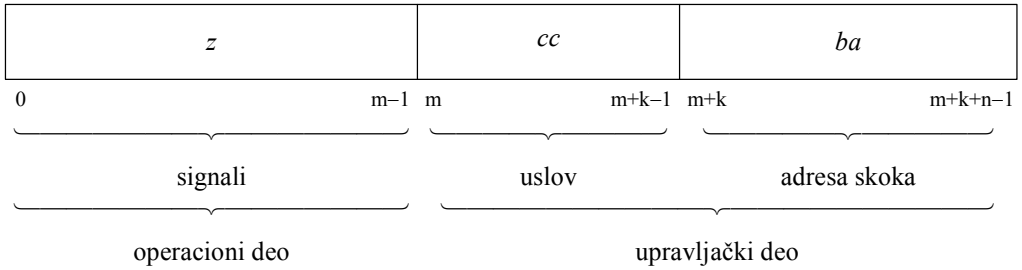
Signali koji se javljaju u izrazu za signal **branch** se generišu na sledeći način:

- $\mathbf{bruncnd} = \mathbf{CW_0 \cdot} \overline{\mathbf{CW}}_4 \cdot \overline{\mathbf{CW}}_5 \cdot \overline{\mathbf{CW}}_6 \cdot \mathbf{CW_7}$
- $\mathbf{brl1} = \mathbf{CW_0 \cdot} \overline{\mathbf{CW}}_4 \cdot \overline{\mathbf{CW}}_5 \cdot \mathbf{CW_6 \cdot} \overline{\mathbf{CW}}_7$
- $\mathbf{brl2} = \mathbf{CW_0 \cdot} \overline{\mathbf{CW}}_4 \cdot \overline{\mathbf{CW}}_5 \cdot \mathbf{CW_6 \cdot CW_7}$
- $\mathbf{brSTORE} = \mathbf{CW_0 \cdot} \overline{\mathbf{CW}}_4 \cdot \mathbf{CW_5 \cdot} \overline{\mathbf{CW}}_6 \cdot \overline{\mathbf{CW}}_7$
- $\mathbf{brimmed} = \mathbf{CW_0 \cdot} \overline{\mathbf{CW}}_4 \cdot \mathbf{CW_5 \cdot} \overline{\mathbf{CW}}_6 \cdot \mathbf{CW_7}$
- $\mathbf{breql} = \mathbf{CW_0 \cdot} \overline{\mathbf{CW}}_4 \cdot \mathbf{CW_5 \cdot CW_6 \cdot} \overline{\mathbf{CW}}_7$
- $\mathbf{brnotPREKID} = \mathbf{CW_0 \cdot} \overline{\mathbf{CW}}_4 \cdot \mathbf{CW_5 \cdot CW_6 \cdot CW_7}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **STORE**, **immed**, **regdir**, **eql** i **PREKID**.

1.3.2.2 Mikroprogramska realizacija sa jednim tipom mikroinstrukcija

U slučaju spajanja koraka postoji samo jedan tip mikroinstrukcije (slika 11).



Slika 11 Format mikroinstrukcije za horizontalni način kodiranja upravljačkih signala

Kodiranje operacionog i upravljačkog dela mikroinstrukcije je dato na slici 12.

0	1	2	3	4	5	6	7
/	ldMAR	$mxMAR_2$	$mxMAR_1$	$mxMAR_0$	wrMEM	ldMBR	$mxMBR_1$

8	9	10	11	12	13	14	15
$mxMBR_0$	ldPC	incPC	mxPC	incSP	decSP	$ldIR_1$	$ldIR_2$

16	17	18	19	20	21	22	23
ldACC	mxACC	incB	decB	ldB	mxB_1	mxB_0	ldPSW

24	25	26	27	28	29	30	31
mxPSW	ldX	mxX_1	mxX_0	ldY	mxY_1	mxY_0	add

32	33	34	35	36	37	38	39
and	asr	ldZ	ldRSRC	wrGPR	ldRDST	mxRDST	/

40	41	42	43	44	45	46	47
/	/	/	/	*cc*			

48	49	50	51	52	53	54	55
ba							

Slika 12 Mikroinstrukcija

Mikroprogram je dat u tabeli 31 .

Tabela 31 Mikroprogram (jedan tip mikroinstrukcije)

! Čitanje instrukcije !

$madr_{00}$ **ldMAR**, **incPC**;
$madr_{01}$ **ldMBR**;
$madr_{02}$ **ldIR1**;
$madr_{03}$ **brl1**, $\mathbf{madr_{2A}}$;
$madr_{04}$ **ldMAR**, **incPC**;
$madr_{05}$ **ldMBR**;
$madr_{06}$ **ldIR2**;
$madr_{07}$ **brl2**, $\mathbf{madr_{0F}}$;
$madr_{08}$ **ldMAR**, **incPC**;
...
$madr_{0E}$ **ldIRN**;

! Formiranje adrese i čitanje operanda !

$madr_{0F}$ **bradr**;

! Direktno registarsko !

$madr_{10}$ **ldRSRC**;
$madr_{11}$ **ldB**, **bruncnd**, $\mathbf{madr_{2A}}$;

! Indirektno registarsko !

$madr_{12}$ **ldRSRC**;
$madr_{13}$ $\mathbf{mxMAR_0}$, **ldMAR**, **bruncnd**, $\mathbf{madr_{26}}$;

! Postdekrement !

$madr_{14}$ **ldRSRC**;
$madr_{15}$ $\mathbf{mxMAR_0}$, **ldMAR**, **ldB**;
$madr_{16}$ **decB**;
$madr_{17}$ **mxRDST**, **ldRDST**;
$madr_{18}$ **wrGPR**, **bruncnd**, $\mathbf{madr_{26}}$;

! Preinkrement !

madr$_{19}$ **ldRSRC**;
madr$_{1A}$ **ldB**;
madr$_{1B}$ **incB**;
madr$_{1C}$ **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**, **mxRDST**, **ldRDST**;
madr$_{1D}$ **wrGPR**, **bruncnd**, **madr$_{26}$**;

! Direktno memorijsko !

madr$_{1E}$ **mxMAR$_1$**, **ldMAR**, **bruncnd**, **madr$_{26}$**;

! Indirektno memorijsko!

madr$_{1F}$ **mxMAR$_1$**, **ldMAR**;
madr$_{20}$ **ldMBR**;
madr$_{21}$ **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**, **bruncnd**, **madr$_{26}$**;

! Indirektno registarsko sa pomerajem !

madr$_{22}$ **ldRSRC**;
madr$_{23}$ **mxX$_0$**, **ldX**, **mxY$_0$**, **ldY**;
madr$_{24}$ **add**, **ldZ**;
madr$_{25}$ **mxMAR$_2$**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
madr$_{26}$ **brSTORE**, **madr$_{2A}$**;
madr$_{27}$ **ldMBR**;
madr$_{28}$ **mxB$_0$**, **ldB**, **bruncnd**, **madr$_{2A}$**;

! Neposredno !

madr$_{29}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

madr$_{2A}$ **bropr**;

! LOAD !

madr$_{2B}$ **mxACC**, **ldACC**, **bruncnd**, **madr$_{48}$**;

! STORE !

madr$_{2C}$ **brimmed**, **madr$_{48}$**;
madr$_{2D}$ **brregdir**, **madr$_{30}$**;
madr$_{2E}$ **mxMBR$_0$**, **ldMBR**;
madr$_{2F}$ **wrMEM**, **bruncnd**, **madr$_{48}$**;
madr$_{30}$ **ldRDST**;
madr$_{31}$ **wrGPR**, **bruncnd**, **madr$_{48}$**;

! ADD !

madr$_{32}$ **ldX**, **ldY**;
madr$_{33}$ **add**, **ldZ**;
madr$_{34}$ **ldACC**, **bruncnd**, **madr$_{48}$**;

! AND !

madr$_{35}$ **ldX**, **ldY**;
madr$_{36}$ **and**, **ldZ**;
madr$_{37}$ **ldACC**, **bruncnd**, **madr$_{48}$**;

! ASR !

madr$_{38}$ **ldY**;
madr$_{39}$ **asr**, **ldZ**;
madr$_{3A}$ **ldACC**, **bruncnd**, **madr$_{48}$**;

! JZ !

madr$_{3B}$ **breql**, **madr $_{3F}$**;
madr$_{3C}$ **bruncnd**, **madr$_{48}$**;

! JSR !

madr$_{3D}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;
madr$_{3E}$ **wrMEM**;

! JMP !

$madr_{3F}$ **ldPC**, **bruncnd**, $\mathbf{madr_{48}}$;

! RTI !

$madr_{40}$ **incSP**;
$madr_{41}$ $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**;
$madr_{42}$ **ldMBR**;
$madr_{43}$ **ldPSW**;

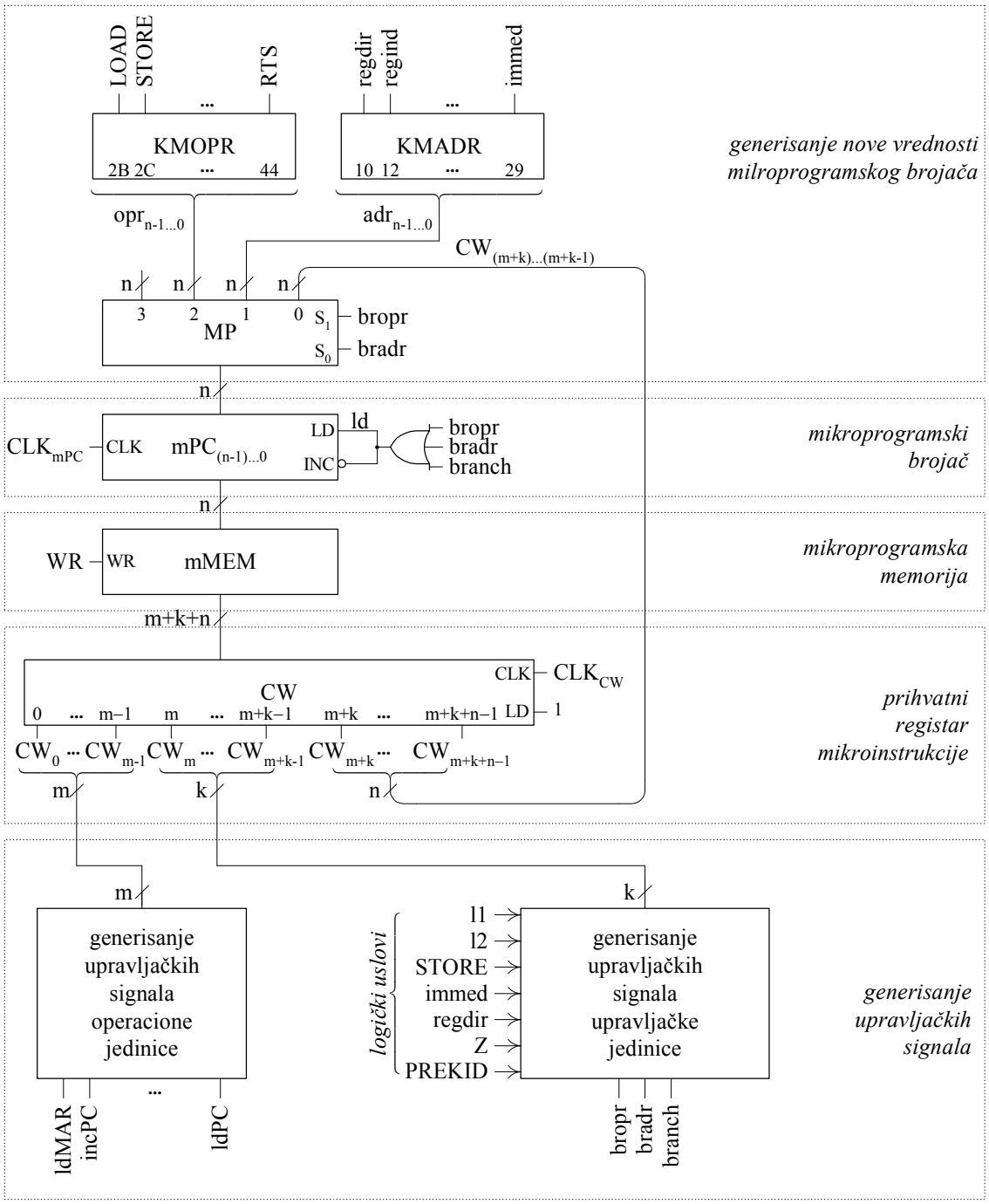
! RTS !

$madr_{44}$ **incSP**;
$madr_{45}$ $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**;
$madr_{46}$ **ldMBR**;
$madr_{47}$ **mxPC**, **ldPC**;

! Opsluživanje prekida !

$madr_{48}$ **brnotPREKID**, $\mathbf{madr_{00}}$;
$madr_{49}$ $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**, **decSP**, $\mathbf{mxMBR_1}$, **ldMBR**;
$madr_{4A}$ **wrMEM**;
$madr_{4B}$ $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**, **decSP**, $\mathbf{mxMBR_1}$, $\mathbf{mxMBR_0}$,**ldMBR**;
$madr_{4C}$ **wrMEM**;
$madr_{4D}$ $\mathbf{mxX_1}$, **ldX**, $\mathbf{mxY_1}$, **ldY**;
$madr_{4E}$ **add**, **ldZ**;
$madr_{4F}$ $\mathbf{mxMAR_2}$, **ldMAR**;
$madr_{50}$ **ldMBR**;
$madr_{51}$ **mxPC**, **ldPC**, **bruncnd**, $\mathbf{madr_{00}}$;

Struktura upravljačke jedinice je data na slici 13.



Slika 13 Struktura upravljačke jedinice sa jednim tipom mikroinstrukcije

Upravljački signali operacione jedinice se generišu na sledeći način:

- **ldMAR** = $\mathbf{CW_1}$
- **mxMAR**$_0$ = $\mathbf{CW_4}$
- **wrGPR** = $\mathbf{CW_{36}}$

Na identičan način se generišu i preostali upravljački signali operacione jedinice.

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr** = $\mathbf{CW_{44}} \cdot \overline{\mathbf{CW}}_{45} \cdot \overline{\mathbf{CW}}_{46} \cdot \mathbf{CW_{47}}$
- **bradr** = $\mathbf{CW_{44}} \cdot \overline{\mathbf{CW}}_{45} \cdot \overline{\mathbf{CW}}_{46} \cdot \overline{\mathbf{CW}}_{47}$
- **branch** = **bruncnd**
  + **brl1*l1** + **brl2*l2** + **brSTORE*STORE** + **brimmed*immed**

$$+ \textbf{brregdir}*\textbf{regdir} + \textbf{breql}*\textbf{eql} + \textbf{brnotPREKID}*\overline{\text{PREKID}}$$

Signali koji se javljaju u izrazu za signal **branch** se generišu na sledeći način:

- **bruncnd** = $\overline{\mathbf{CW}}_{44} \cdot \overline{\mathbf{CW}}_{45} \cdot \overline{\mathbf{CW}}_{46} \cdot \mathbf{CW}_{7}$
- **brl1** = $\overline{\mathbf{CW}}_{44} \cdot \overline{\mathbf{CW}}_{45} \cdot \mathbf{CW}_{46} \cdot \overline{\mathbf{CW}}_{47}$
- **brl2** = $\overline{\mathbf{CW}}_{44} \cdot \overline{\mathbf{CW}}_{45} \cdot \mathbf{CW}_{46} \cdot \mathbf{CW}_{47}$
- **brSTORE** = $\overline{\mathbf{CW}}_{44} \cdot \mathbf{CW}_{45} \cdot \overline{\mathbf{CW}}_{46} \cdot \overline{\mathbf{CW}}_{47}$
- **brimmed** = $\overline{\mathbf{CW}}_{44} \cdot \mathbf{CW}_{45} \cdot \overline{\mathbf{CW}}_{46} \cdot \mathbf{CW}_{47}$
- **breql** = $\overline{\mathbf{CW}}_{44} \cdot \mathbf{CW}_{45} \cdot \mathbf{CW}_{46} \cdot \overline{\mathbf{CW}}_{47}$
- **brnotPREKID** = $\overline{\mathbf{CW}}_{44} \cdot \mathbf{CW}_{45} \cdot \mathbf{CW}_{46} \cdot \mathbf{CW}_{47}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **STORE**, **immed**, **regdir**, **eql** i **PREKID**.

1.3.2.3 Kodiranje upravljačkih signala operacione jedinice

Kodiranje upravljačkih signala operacione jedinice se može realizovati na više različitih načina. Do sada je bio poseban bit za svaki upravljački signal operacione jedinice. U tom slučaju se kaže da je format instrukcije horizontalan. Pored toga postoje i vertikalan i mešoviti format.

Ova razmatranja važe za oba formata mikroinstrukcija i to dva tipa mikroinstrukcija sa posebnim operacionim i upravljačkim mikroinstrukcijama i jedan tip mikroinstrukcije sa operacionim i upravljačkim delom.

1.3.2.3.1 Mikroprogramska realizacija sa vertikalnim formatom mikroinstrukcija

U slučaju vertikalnog kodiranja upravljačkih signala operacione jedinice jedna binarna vrednost se dodeljuje određenoj kombinaciji upravljačkih signala operacione jedinice neophodnoj da se u jednom koraku realizuje jedna mikrooperacija. Vertikalno kodiranje upravljačkih signala operacione jedinice se realizuje na isti način bez obzira na to da li su oni specificirani posebnim operacionim mikroinstrukcijama ili operacionim delom mikroinstrukcije. U ovom odeljku se daje jedan mogući način kodiranja upravljačkih signala operacione jedinice i realizacija upravljačkih jedinica sa dva i jednim tipom mikroinstrukcija.

Kombinacije upravljačkih signala operacione jedinice i simboličke oznake usvojenih kodova dati su u tabeli 32. Usvojeni kodovi su simbolički označeni sa V_{xx}, pri čemu xx odgovara heksadekadnoj vrednosti usvojenog koda. Na primer, sa V_{00} je simbolički označena heksadekadna vrednost 00 ovog polja. Svakom usvojenom kodu odgovara neka kombinacija upravljačkih signala. Na primer, kodu V_{01} odgovara signal **ldMAR**, kodu V_{01} signali **mxMAR0** i **ldMAR**, itd. Kombinacije upravljačkih signala su tako odabrane da njima mogu da se pokriju sve situacije iz sekvence upravljačkih signala po koracima (tabele 1 i 2). Iz tabele 32 se vidi da je za kodiranje kombinacija upravljačkih signala operacione jedinice potrebno 44 koda, pa je za kodiranje polja *kombinacija upravljačkih signala* operacionih mikroinstrukcija dovoljno 6 bitova.

Tabela 32 Kombinacije upravljačkih signala operacione jedinice i simboličke oznake kodova

Kombinacija signala	Oznaka koda
/	V_{00}
ldMAR	V_{01}
$mxMAR_0$, ldMAR	V_{02}
$mxMAR_1$, ldMAR	V_{03}
$mxMAR_1$, $mxMAR_0$, ldMAR	V_{04}
$mxMAR_2$, ldMAR	V_{05}
$mxMAR_2$, $mxMAR_0$, ldMAR	V_{06}
$mxMAR_2$, $mxMAR_1$, ldMAR	V_{07}
ldMBR	V_{08}
$mxMBR_0$, ldMBR	V_{09}
$mxMBR_1$, ldMBR	V_{0A}
$mxMBR_1$, $mxMBR_0$, ldMBR	V_{0B}
wrMEM	V_{0C}
ldIR1	V_{0D}
ldIR2	V_{0E}
ldIRN	V_{0F}
ldPC	V_{10}
mxPC, ldPC	V_{11}
incPC	V_{12}
ldRDST	V_{13}
mxRDST, ldRDST	V_{14}
wrGPR	V_{15}
ldRSRC	V_{16}
ldPSW	V_{17}
mxPSW, ldPSW	V_{18}
ldSP	V_{19}
incSP	V_{1A}
decSP	V_{1B}
ldIVTP	V_{1C}
ldBR	V_{1D}
ldACC	V_{1E}
mxACC, ldACC	V_{1F}
ldB	V_{20}
incB	V_{21}
decB	V_{22}
mxB_0, ldB	V_{23}
mxB_1, ldB	V_{24}
ldX	V_{25}
mxX_0, ldX	V_{26}
mxX_1, ldX	V_{27}
ldY	V_{28}
mxY_0, ldY	V_{29}
mxY_1, ldY	V_{2A}
add, ldZ	V_{2B}
and, ldZ	V_{2C}
asr, ldZ	V_{2D}

Kodom V_{00} se definiše da upravljačka jedinica ne generiše ni jedan signal čime se u operacionoj jedinici ne realizacije ni jedna mikrooperacija. Kodom V_{01} se definiše da upravljačka jedinica generiše signal **ldMAR** čime se u operacionoj jedinici realizacije mikrooperacija upisa sadržaja registra PC u registar MAR (slika 2). Kodom V_{02} se definiše da upravljačka jedinica generiše signale **$mxMAR_0$** i **ldMAR** čime se u operacionoj jedinici

realizacije mikrooperacija upisa sadržaja registra RSRC u registar MAR (slika 2). Na sličan način se odgovarajućim kodovima specificiraju i sve ostale mikrooperacije upisa u registre operacione jedinice. Kodom V_{15} se definiše da upravljačka jedinica generiše signale **wrGPR** čime se u operacionoj jedinici realizacije mikrooperacija upisa sadržaja registra RDST u registar opšte namene adresiran poljem IR_GPR prihvatnog registra instrukcije IR. Kodom V_{0C} se definiše da upravljačka jedinica generiše signale **wrMEM** čime se u operacionoj jedinici realizacije mikrooperacija upisa sadržaja registra MBR u memorijsku lokaciju adresiranu sadržajem registra MAR. Kodom V_{29} se definiše da upravljačka jedinica generiše signale **add** i **ldZ** čime se u operacionoj jedinici realizacije mikrooperacija sabiranja sadržaja registara X i Y u ALU i upis rezultata u registar Z. Na sličan način se odgovarajućim kodovima specificiraju i sve ostale aritmetičke, logičke i pomeračke mikrooperacije u ALU i upis rezultata u registar Z.

Sekvence upravljačkih signala po koracima (tabele 1 i 2) su tako formirane da se njima omogućava izvršavanje mikrooperacija u najmanjem broju koraka, pa se u nekim koracima generišu upravljački signali operacione jedinice koji omogućavaju izvršavanje više mikrooperacija u istom koraku. U slučaju vertikalnog formatom mikroinstrukcija u jednom koraku se omogućava izvršavanje samo jedne mikrooperacije. Zbog toga se neki od koraka iz sekvence upravljačkih signala po koracima u kojima se javlja više upravljačkih signala operacione jedinice mora se zameniti sa više koraka, tako da se u svakom koraku realizuje samo jedna mikrooperacija.

Kao primer ovoga se može iz sekvence upravljačkih signala po koracima (tabela 1) uzeti korak $step_{00}$:

ldMAR, **incPC**

koji se mora razbiti na sledeća dva koraka:

ldMAR;
incPC;

ili $korak_{4A}$:

mxMAR$_2$, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;

koji se mora razbiti na sledeća tri koraka:

mxMAR$_2$, **mxMAR$_0$**, **ldMAR**;
decSP;
mxMBR$_1$, **ldMBR**;

Pored toga, kao rezultat povećanja broja koraka potrebno je u iskazu

br (*if* **l1** *then* $step_{31}$)

zameniti $step_{31}$ odgovarajućom vrednošću. To treba uraditi i u svim ostalim iskazima za bezuslovne, uslovne i višestruke uslovne skokove.

Po ovom postupku je na osnovu sekvence upravljačkih signala po koracima (tabela 1) formirana sekvenca upravljačkih signala po koracima za upravljačku jedinicu sa vertikalnim formatom mikroinstrukcija (tabela 33).

Tabela 33 Sekvenca upravljačkih signala po koracima za upravljačku jedinicu sa vertikalnim formatom mikroinstrukcija bez spajanja operacionih i upravljačkih koraka

! Čitanje instrukcije !

$step_{00}$ **ldMAR**;
$step_{01}$ **incPC**;
$step_{02}$ **ldMBR**;
$step_{03}$ **ldIR1**;
$step_{04}$ *br* (*if* **l1** *then* $step_{38}$);
$step_{05}$ **ldMAR**;

$step_{06}$ **incPC**;
$step_{07}$ **ldMBR**;
$step_{08}$ **ldIR2**;
$step_{09}$ *br* (*if* **l2** *then* $step_{13}$);
$step_{0A}$ **ldMAR**
$step_{0B}$ **incPC**;
...
$step_{12}$ **ldIRN**;

! Formiranje adrese i čitanje operanda !

$step_{13}$ *br* (*case* (**dirreg**, **indreg**, **postdec**, **preinc**,
dirmem, **indmem**, **indregpom**, **immed**) *then*
(**dirreg**, $step_{14}$), (**indreg**, $step_{17}$), (**postdec**, $step_{1A}$), (**preinc**, $step_{21}$),
(**dirmem**, $step_{28}$), (**indmem**, $step_{2A}$), (**indregpom**, $step_{2E}$), (**immed**, $step_{37}$));

! Direktno registarsko !
$step_{14}$ **ldRSRC**;
$step_{15}$ **ldB**;
$step_{16}$ *br* $step_{38}$;

! Indirektno registarsko !
$step_{17}$ **ldRSRC**;
$step_{18}$ **mxMAR$_0$**, **ldMAR**;
$step_{19}$ *br* $step_{33}$;

! Postdekrement !
$step_{1A}$ **ldRSRC**;
$step_{1B}$ **mxMAR$_0$**, **ldMAR**;
$step_{1C}$ **ldB**;
$step_{1D}$ **decB**;
$step_{1E}$ **mxRDST**, **ldRDST**;
$step_{1F}$ **wrGPR**;
$step_{20}$ *br* $step_{33}$;

! Preinkrement !
$step_{21}$ **ldRSRC**;
$step_{22}$ **ldB**;
$step_{23}$ **incB**;
$step_{24}$ **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**;
$step_{25}$ **mxRDST**, **ldRDST**;
$step_{26}$ **wrGPR**;
$step_{27}$ *br* $step_{33}$;

! Direktno memorijsko !
$step_{28}$ **mxMAR$_1$**, **ldMAR**;
$step_{29}$ *br* $step_{33}$;

! Indirektno memorijsko!
$step_{2A}$ **mxMAR$_1$**, **ldMAR**;
$step_{2B}$ **ldMBR**;
$step_{2C}$ **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**;
$step_{2D}$ *br* $step_{33}$;

! Indirektno registarsko sa pomerajem !
$step_{2E}$ **ldRSRC**;
$step_{2F}$ **mxX$_0$**, **ldX**;
$step_{30}$ **mxY$_0$**, **ldY**;
$step_{31}$ **add**, **ldZ**;
$step_{32}$ **mxMAR$_2$**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !

step$_{33}$ *br* (*if* **STORE** *then* step$_{38}$);
step$_{34}$ **ldMBR**;
step$_{35}$ **mxB$_0$**, **ldB**;
step$_{36}$ *br* step$_{38}$;
! Neposredno !
step$_{37}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

step$_{38}$ *br* (*case* (**LOAD**, **STORE**, **ADD**, **AND**, **ASR**, **JZ**, **JMP**, **JSR**, **RTI**, **RTS**) *then*
(**LOAD**, step$_{39}$), (**STORE**, step$_{3B}$),
(**ADD**, step$_{43}$), (**AND**, step$_{48}$), (**ASR**, step$_{4D}$),
(**JZ**, step$_{51}$), (**JMP**, step$_{57}$), (**JSR**, step$_{53}$), (**RTI**, step$_{59}$), (**RTS**, step$_{5D}$));
! LOAD !
step$_{39}$ **mxACC**, **ldACC**;
step$_{3A}$ *br* step$_{61}$;
! STORE !
step$_{3B}$ *br* (*if* **immed** *then* step$_{61}$);
step$_{3C}$ *br* (*if* **regdir** *then* step$_{40}$);
step$_{3D}$ **mxMBR$_0$**, **ldMBR**;
step$_{3E}$ **wrMEM**;
step$_{3F}$ *br* step$_{61}$;
step$_{40}$ **ldRDST**;
step$_{41}$ **wrGPR**;
step$_{42}$ *br* step$_{61}$;
! ADD !
step$_{43}$ **ldX**;
step$_{44}$ **ldY**;
step$_{45}$ **add**, **ldZ**;
step$_{46}$ **ldACC**;
step$_{47}$ *br* step$_{61}$;
! AND !
step$_{48}$ **ldX**;
step$_{49}$ **ldY**;
step$_{4A}$ **and**, **ldZ**;
step$_{4B}$ **ldACC**;
step$_{4C}$ *br* step$_{61}$;
! ASR !
step$_{4D}$ **ldY**;
step$_{4E}$ **asr**, **ldZ**;
step$_{4F}$ **ldACC**;
step$_{50}$ *br* step$_{61}$;
! JZ !
step$_{51}$ *br* (*if* **eql** *then* step$_{57}$);
step$_{52}$ *br* step$_{61}$;
! JSR !
step$_{53}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
step$_{54}$ **decSP**;
step$_{55}$ **mxMBR$_1$**, **ldMBR**;
step$_{56}$ **wrMEM**;
! JMP !
step$_{57}$ **ldPC**;
step$_{58}$ *br* step$_{61}$;
! RTI !

$step_{59}$ **incSP**;
$step_{5A}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$step_{5B}$ **ldMBR**;
$step_{5C}$ **ldPSW**;

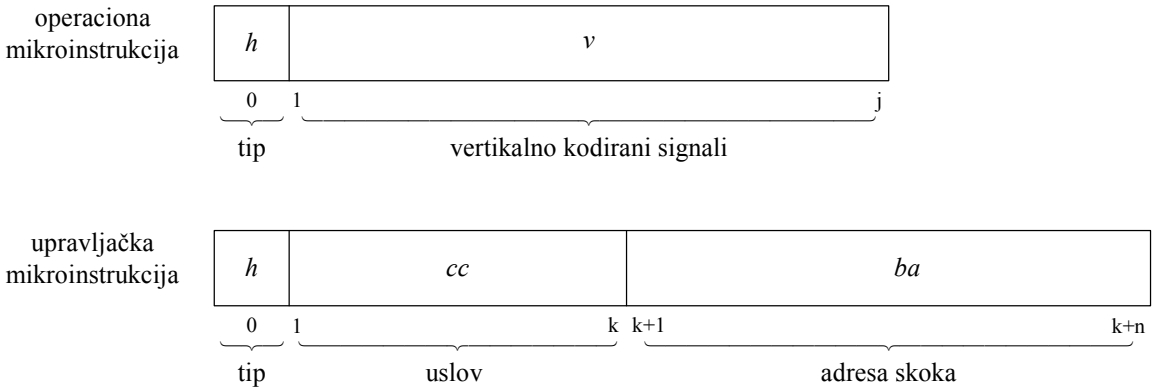
! RTS !

$step_{5D}$ **incSP**;
$step_{5E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$step_{5F}$ **ldMBR**;
$step_{60}$ **mxPC**, **ldPC**;

! Opsluživanje prekida !

$step_{61}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{62}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$step_{63}$ **decSP**;
$step_{64}$ **mxMBR$_1$**, **ldMBR**;
$step_{65}$ **wrMEM**;
$step_{66}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$step_{67}$ **decSP**;
$step_{68}$ **mxMBR$_1$**, **mxMBR$_0$**, **ldMBR**;
$step_{69}$ **wrMEM**;
$step_{6A}$ **mxX$_1$**, **ldX**;
$step_{6B}$ **mxY$_1$**, **ldY**;
$step_{6C}$ **add**, **ldZ**;
$step_{6D}$ **mxMAR$_2$**, **ldMAR**;
$step_{6E}$ **ldMBR**;
$step_{6F}$ **mxPC**, **ldPC**;
$step_{70}$ *br* $step_{00}$;

Formati mikroinstrukcija su dati na slici 14. Polja *h*, *cc* i *ba* imaju isto značenje kao i u slučaju upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice (slika 7), pri čemu su i kodovi polja *cc* za bezuslovne, uslovne i višestruke uslovne skokove isti (tabele 23, 24 i 25). Razlika je samo u poljima *z* i *v* operacionih mikroinstrukcija. U slučaju horizontalnog kodiranja upravljačkih signala operacione jedinice poseban bit polja *z* je dodeljen svakom signalu operacione jedinice, dok su u slučaju vertikalnog kodiranja upravljačkih signala operacione jedinice u polju *v* pojavljuju kodovi koji određuju kombinacije aktivnih vrednosti upravljačkih signala operacione jedinice neophodnih da se u jednom koraku realizuje jedna mikrooperacija. Kodovi polja *v* su dati u tabeli 32.



Slika 14 Formati operacionih i upravljačkih mikroinstrukcija za vertikalno kodiranje upravljačkih signala

U slučaju razmatrane operacione jedinice sa direktnim vezama formati operacione i upravljačke mikroinstrukcije su dati na slikama 15 i 16. Format operacione mikroinstrukcije je dat na slici 8. Polje *h* je 0. Format upravljačke mikroinstrukcije je dat na slici 9. Polje *h* je 1.

0	1	2	3	4	5	6	7
0	*v*						

8	9	10	11	12	13	14	15
/	/	/	/	/	/	/	/

Slika 15 Operaciona mikroinstrukcija

0	1	2	3	4	5	6	7
1	/	/	/	*cc*			

8	9	10	11	12	13	14	15
ba							

Slika 16 Upravljačka mikroinstrukcija

Mikroprogram se formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima za upravljačku jedinicu sa vertikalnim formatom mikroinstrukcija (tabela 33) formira jedna mikroinstrukcija i to operaciona ili upravljačka.

Kod formiranja operacionih mikroinstrukcija polazi se od sekvence upravljačkih signala po koracima za upravljačku jedinicu sa vertikalnim formatom mikroinstrukcija i traže koraci u kojima se javljaju upravljački signali operacione jedinice. Za svaki takav korak formira se jedna operaciona mikroinstrukcija koja, saglasno kombinaciji upravljačkih signala operacione jedinice koja se javlja u datom koraku i tabeli 32, dobija odgovarajuću vrednost za polje *kombinacija upravljačkih signala*. Na primer, za korake $step_{00}$, $step_{01}$ i $step_{06}$

$step_{00}$ **ldMAR**;
$step_{01}$ **incPC**;
$step_{02}$ **ldMBR**;

formiraju se tri operacione mikroinstrukcije

$madr_{00}$ V_{01}; ! **ldMAR** !
$madr_{01}$ V_{12}; ! **incPC** !
$madr_{02}$ V_{08}; ! **ldMBR** !

sa vrednostima polja *kombinacija upravljačkih signala* V_{01}, V_{12} i V_{08}, respektivno.

Kod formiranja upravljačkih mikroinstrukcija primenjuje se identičan postupak kao i u slučaju upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice.

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 1) formiran mikroprogram (tabela 34). On ima sledeću formu:

- na levoj strani se nalaze adrese mikroinstrukcija u mikroprogramskoj memoriji u heksadekadnom obliku,
- u sredini su ili operacione mikroinstrukcije, predstavljene simboličkim oznaka polja *v* za kombinacije upravljačkih signala operacione jedinice, ili upravljačke mikroinstrukcije, predstavljene nizom simboličkih oznaka signala za tip mikroinstrukcije cnt, bezuslovne, uslovne i višestruke uslovne skokove koji treba da budu aktivni i adresu skoka razdvojenih zapetama,
- dok komentar, u koracima gde se to radi lakšeg razumevanja smatralo korisnim, uvek počinje uskličnikom (!) i proteže se do sledećeg uskličnika (!).

Tabela 34 Mikroprogram

! Čitanje instrukcije !

madr$_{00}$ V$_{01}$; ! **ldMAR** !
madr$_{01}$ V$_{12}$; ! **incPC** !
madr$_{02}$ V$_{08}$; ! **ldMBR** !
madr$_{03}$ V$_{0D}$; ! **ldIR1** !
madr$_{04}$ **cnt**, **brl1**, madr$_{38}$;
madr$_{05}$ V$_{01}$; ! **ldMAR** !
madr$_{06}$ V$_{12}$; ! **incPC** !
madr$_{07}$ V$_{08}$; ! **ldMBR** !
madr$_{08}$ V$_{0E}$! **ldIR2** !
madr$_{09}$ **cnt**, **brl2**, madr$_{13}$;
madr$_{0A}$ V$_{01}$; ! **ldMAR** !
madr$_{0B}$ V$_{12}$; ! **incPC** !
...
madr$_{12}$ V$_{0F}$; ! **ldIRN** !

! Formiranje adrese i čitanje operanda !

madr$_{13}$ **cnt**, **bradr**;

! Direktno registarsko !

madr$_{14}$ V$_{16}$; ! **ldRSRC** !
madr$_{15}$ V$_{20}$; ! **ldB** !
madr$_{16}$ **cnt**, **bruncnd**, madr$_{38}$;

! Indirektno registarsko !

madr$_{17}$ V$_{16}$; ! **ldRSRC** !
madr$_{18}$ V$_{02}$; ! **mxMAR$_0$**, **ldMAR** !
madr$_{19}$ **cnt**, **bruncnd**, madr$_{33}$;

! Postdekrement !

madr$_{1A}$ V$_{16}$; ! **ldRSRC** !
madr$_{1B}$ V$_{02}$; ! **mxMAR$_0$**, **ldMAR** !
madr$_{1C}$ V$_{20}$; ! **ldB** !
madr$_{1D}$ V$_{22}$; ! **decB** !
madr$_{1E}$ V$_{14}$; ! **mxRDST**, **ldRDST** !
madr$_{1F}$ V$_{15}$; ! **wrGPR** !
madr$_{20}$ **cnt**, **bruncnd**, madr$_{33}$;

! Preinkrement !

madr$_{21}$ V$_{16}$; ! **ldRSRC** !
madr$_{22}$ V$_{20}$; ! **ldB** !
madr$_{23}$ V$_{21}$; ! **incB** !
madr$_{24}$ V$_{07}$; ! **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR** !
madr$_{25}$ V$_{14}$; ! **mxRDST**, **ldRDST** !
madr$_{26}$ V$_{15}$; !**wrGPR** !
madr$_{27}$ **cnt**, **bruncnd**, madr$_{33}$;

! Direktno memorijsko !

madr$_{28}$ V$_{03}$; ! **mxMAR$_1$**, **ldMAR** !
madr$_{29}$ **cnt**, **bruncnd**, madr$_{33}$;

! Indirektno memorijsko!

madr$_{2A}$ V$_{03}$; ! **mxMAR$_1$**, **ldMAR** !
madr$_{2B}$ V$_{08}$; ! **ldMBR** !
madr$_{2C}$ V$_{04}$; ! **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR** !
madr$_{2D}$ **cnt**, **bruncnd**, madr$_{33}$;

! Indirektno registarsko sa pomerajem !

$madr_{2E}$ V_{16}; ! **ldRSRC** !
$madr_{2F}$ V_{26}; ! **mxX_0**, **ldX** !
$madr_{30}$ V_{29}; ! **mxY_0**, **ldY** !
$madr_{31}$ V_{2B}; ! **add**, **ldZ** !
$madr_{32}$ V_{05}; !**mxMAR_2**, **ldMAR** !
! Čitanje operanda za memorijska adresiranja !
$madr_{33}$ **cnt**, **brSTORE**, $madr_{38}$;
$madr_{34}$ V_{08}; ! **ldMBR** !
$madr_{35}$ V_{23}; ! **mxB_0**, **ldB** !
$madr_{36}$ **cnt**, **bruncnd**, $madr_{38}$;
! Neposredno !
$madr_{37}$ V_{24}; ! **mxB_1**, **ldB** !

! Izvršavanje operacije !

$madr_{38}$ **bropr**;
! LOAD !
$madr_{39}$ V_{1F}; ! **mxACC**, **ldACC** !
$madr_{3A}$ **cnt**, **bruncnd**, $madr_{61}$;
! STORE !
$madr_{3B}$ **cnt**, **brimmed**, $madr_{61}$;
$madr_{3C}$ **cnt**, **brregdir**, $madr_{40}$;
$madr_{3D}$ V_{09}; ! **mxMBR_0**, **ldMBR** !
$madr_{3E}$ V_{0C}; ! **wrMEM** !
$madr_{3F}$ **cnt**, **bruncnd**, $madr_{61}$;
$madr_{40}$ V_{13}; ! **ldRDST** !
$madr_{41}$ V_{15}; ! **wrGPR** !
$madr_{42}$ **cnt**, **bruncnd**, $madr_{61}$;
! ADD !
$madr_{43}$ V_{25}; ! **ldX** !
$madr_{44}$ V_{28}; ! **ldY** !
$madr_{45}$ V_{2B}; ! **add**, **ldZ** !
$madr_{46}$ V_{1E}; ! **ldACC** !
$madr_{47}$ **cnt**, **bruncnd**, $madr_{61}$;
! AND !
$madr_{48}$ V_{25}; ! **ldX** !
$madr_{49}$ V_{28}; ! **ldY** !
$madr_{4A}$ V_{2B}; ! **and**, **ldZ** !
$madr_{4B}$ V_{1E}; ! **ldACC** !
$madr_{4C}$ **cnt**, **bruncnd**, $madr_{61}$;
! ASR !
$madr_{4D}$ V_{28}; ! **ldY** !
$madr_{4E}$ V_{2D}; ! **asr**, **ldZ** !
$madr_{4F}$ V_{1E}; ! **ldACC** !
$madr_{50}$ **cnt**, **bruncnd**, $madr_{61}$;
! JZ !
$madr_{51}$ **cnt**, **breql**, $madr_{57}$;
$madr_{52}$ **cnt**, **bruncnd**, $madr_{61}$;
! JSR !
$madr_{53}$ V_{06}; ! **mxMAR_2**, **mxMAR_0**, **ldMAR** !
$madr_{54}$ V_{1B}; ! **decSP** !
$madr_{55}$ V_{0A}; ! **mxMBR_1**, **ldMBR** !
$madr_{56}$ V_{0C}; ! **wrMEM** !
! JMP !

$madr_{57}$ V_{10}; ! **ldPC** !
$madr_{58}$ **cnt**, **bruncnd**, $madr_{61}$;
! RTI !
$madr_{59}$ V_{1A}; ! **incSP** !
$madr_{5A}$ V_{06}; ! **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR** !
$madr_{5B}$ V_{08}; ! **ldMBR** !
$madr_{5C}$ V_{17}; ! **ldPSW** !
! RTS !
$madr_{5D}$ V_{1A}; ! **incSP** !
$madr_{5E}$ V_{06}; ! **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR** !
$madr_{5F}$ V_{08}; ! **ldMBR** !
$madr_{60}$ V_{11}; ! **mxPC**, **ldPC** !

! Opsluživanje prekida !

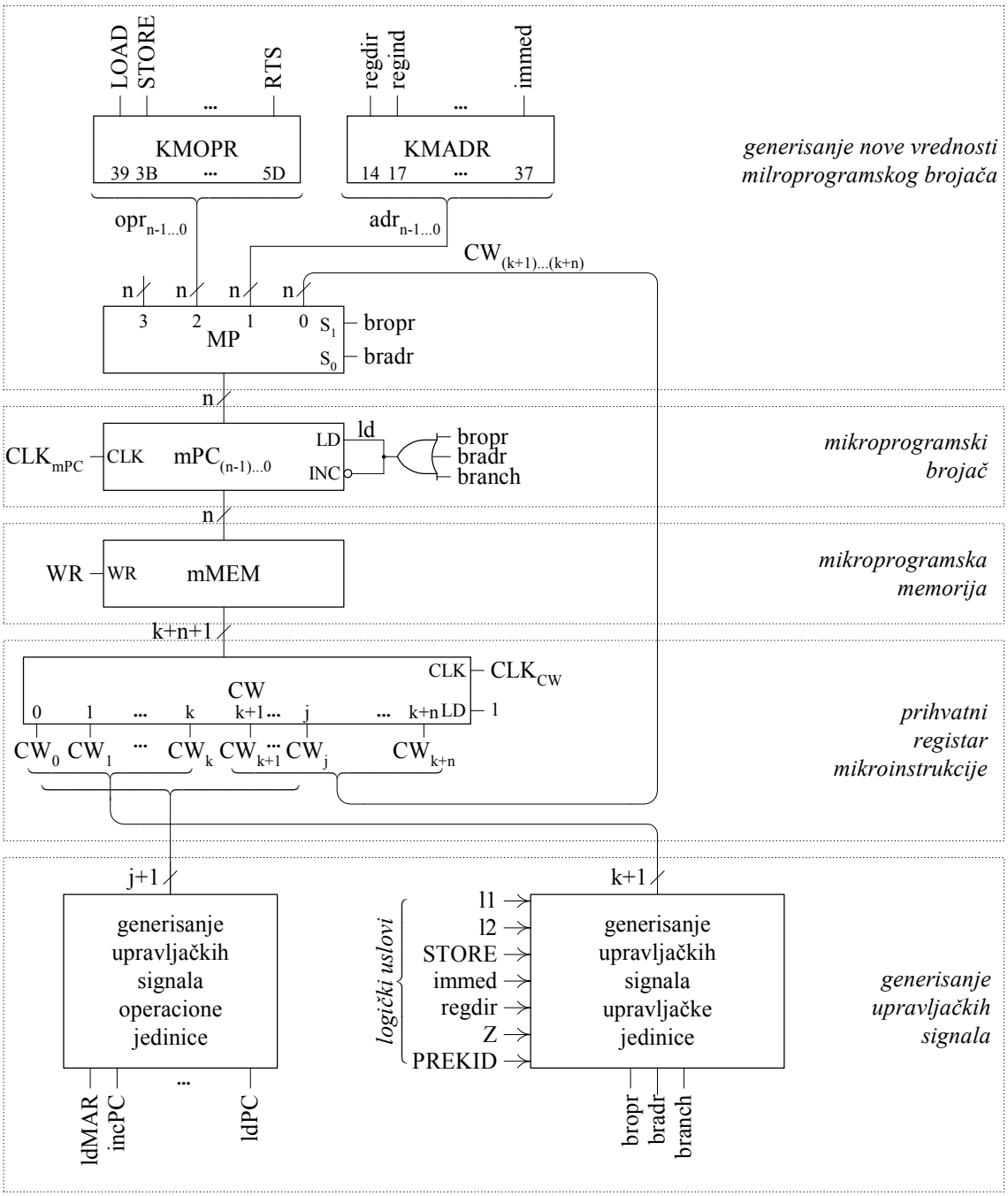
$madr_{61}$ **cnt**, **brnotPREKID**, $madr_{00}$;
$madr_{62}$ V_{06}; ! **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR** !
$madr_{63}$ V_{1B}; ! **decSP** !
$madr_{64}$ V_{0A}; ! **mxMBR$_1$**, **ldMBR** !
$madr_{65}$ V_{0C}; !**wrMEM** !
$madr_{66}$ V_{06}; ! **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR** !
$madr_{67}$ V_{1B}; ! **decSP** !
$madr_{68}$ V_{0B}; ! **mxMBR$_1$**, **mxMBR$_0$**, **ldMBR** !
$madr_{69}$ V_{0C}; ! **wrMEM** !
$madr_{6A}$ V_{25}; ! **mxX$_1$**, **ldX** !
$madr_{6B}$ V_{2A}; ! **mxY$_1$**, **ldY** !
$madr_{6C}$ V_{2B}; ! **add**, **ldZ** !
$madr_{6D}$ V_{05}; ! **mxMAR$_2$**, **ldMAR** !
$madr_{6E}$ V_{08}; ! **ldMBR** !
$madr_{6F}$ V_{11}; ! **mxPC**, **ldPC** !
$madr_{70}$ **cnt**, **bruncnd**, $madr_{00}$;

Struktura upravljačke jedinice je slična strukturi upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice i data je na slici 17. Mikroprogramski brojač se na isti način inkrementira i u mikroprogramski brojač se na isti način upisuje nova vrednost, pri čemu mikroprogramski brojač koraka prolazi kroz veći broj stanja. Druge su jedino vrednosti koje generišu kombinacione mreže KMOPR i KMADR i vrednosti $CW_{k+1...k+n}$ koje se upisuju iz upravljačke mikroinstrukcije. Kombinaciona mreža KMOPR generiše vrednosti 39, 3B, ..., 5D pri aktivnim vrednostima signala **LOAD**, **STORE**, ..., **RTS**, respektivno. Kombinaciona mreža KMADR generiše vrednosti 14, 17, ..., 37 pri aktivnim vrednostima signala **dirreg**, **indreg**, ..., **immed**, respektivno.

Upravljački signali operacione jedinice se generišu ukoliko je $\mathbf{CW_0} = 0$, jer se tada u prihvatnom registru mikroinstrukcije $CW_{0...k+n}$ nalazi operaciona mikroinstrukcija. Najpre se dekodovanjem na osnovu signala $\mathbf{CW_{0...7}}$ generišu signali kombinacija upravljačkih signala $\mathbf{V_{00}}$ do $\mathbf{V_{2D}}$. (tabela 32) i to:

- $\mathbf{V_{00}} = \overline{\mathbf{CW}}_0\ \overline{\mathbf{CW}}_1 \cdot \overline{\mathbf{CW}}_2 \cdot \overline{\mathbf{CW}}_3 \cdot \overline{\mathbf{CW}}_4 \cdot \overline{\mathbf{CW}}_5 \cdot \overline{\mathbf{CW}}_6 \cdot \overline{\mathbf{CW}}_7$
- $\mathbf{V_{01}} = \overline{\mathbf{CW}}_0\ \overline{\mathbf{CW}}_1 \cdot \overline{\mathbf{CW}}_2 \cdot \overline{\mathbf{CW}}_3 \cdot \overline{\mathbf{CW}}_4 \cdot \overline{\mathbf{CW}}_5 \cdot \overline{\mathbf{CW}}_6 \cdot \mathbf{CW}_7$
- $\mathbf{V_{02}} = \overline{\mathbf{CW}}_0\ \overline{\mathbf{CW}}_1 \cdot \overline{\mathbf{CW}}_2 \cdot \overline{\mathbf{CW}}_3 \cdot \overline{\mathbf{CW}}_4 \cdot \overline{\mathbf{CW}}_5\ \mathbf{CW}_6 \cdot \overline{\mathbf{CW}}_7$ itd.

Potom, se za svaki upravljački signal operacione jedinice posmatra u kojim kombinacijama upravljačkih signala se signal pojavljuje (tabela 32), pa se dati signal dobija kao njihova unija.



Slika 17 Struktura upravljačke jedinice mikroprogramske realizije sa dva tipa mikroinstrukcija i vertikalnim formatom mikroinstrukcija

Upravljački signali operacione jedinice se generišu na sledeći način:

- **ldMAR** = $\mathbf{V_{01} + V_{02} + V_{03} + V_{04} + V_{05} + V_{06} + V_{07}}$
- **mxMAR$_0$** = $\mathbf{V_{02} + V_{04} + V_{06}}$
- **wrGPR** = $\mathbf{V_{15}}$

Na identičan način se generišu i preostali upravljački signali operacione jedinice.

Upravljački signali upravljačke jedinice se generišu na identičan način kao i u slučaju upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice, jer je format upravljačkih mikroinstrukcija identičan, i to:

- **bropr** = $\mathbf{CW_0 \cdot CW_4 \cdot \overline{CW}_5 \cdot \overline{CW}_6 \cdot CW_7}$

- **bradr** = $\mathbf{CW_0 \cdot CW_4 \cdot \overline{CW}_5 \cdot \overline{CW}_6 \cdot \overline{CW}_7}$
- **branch** = **bruncnd**
  + **brl1*l1** + **brl2*l2** + **brSTORE*STORE** + **brimmed*immed**
  + **brregdir*regdir** + **breql*eql**+ **brnotPREKID*** $\overline{\text{PREKID}}$

Signali koji se javljaju u izrazu za signal **branch** se generišu na sledeći način:

- **bruncnd** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot \overline{CW}_6 \cdot CW_7}$
- **brl1** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot CW_6 \cdot \overline{CW}_7}$
- **brl2** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot CW_6 \cdot CW_7}$
- **brSTORE** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot \overline{CW}_7}$
- **brimmed** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot CW_7}$
- **breql** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot CW_6 \cdot \overline{CW}_7}$
- **brnotPREKID** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot CW_6 \cdot CW_7}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **STORE**, **immed**, **regdir**, **eql** i **PREKID**.

$$$$

Moguće je vertikalno kodiranje signala i za slučaj kada postoji spajanje koraka i samo jedan tip mikroinstrukcije.

$$$$

1.3.2.3.2 Mikroprogramska realizacija sa mešovitim formatom mikroinstrukcija

U slučaju mešovitog kodiranja upravljačkih signala operacione jedinice postoji više grupa signala, pri čemu je unutar grupe binarno kodiranje signala. Time su kombinovani pozitivni efekti horizontalnog i vertikalnog kodiranja. Na nivou grupa je horizontalno, a unutar grupe vertikalno kodiranje. Prilikom određivanja broja grupa i raspoređivanja signala po grupama treba omogućiti da se svi signali koji se u sekvenci upravljačkih signala po koracima javljaju u istom koraku mogu da pojave u istom koraku. Mešovito kodiranje upravljačkih signala operacione jedinice se realizuje na isti način bez obzira na to da li su oni specificirani posebnim operacionim mikroinstrukcijama ili operacionim delom mikroinstrukcije. U ovom odeljku se daje jedan mogući način kodiranja upravljačkih signala operacione jedinice i realizacija upravljačkih jedinica sa dva i jednim tipom mikroinstrukcija.

Usvojeno kodiranje upravljačkih signala operacione jedinice je dato u tabeli 35. Upravljački signali operacione jedinice kojih ima 41 grupisani su u osam polja označena sa M1, M2, M3, M4, M5, M6, M7 i M8. Vrednost 0 svih polja se koristi za specificiranje da ni jedan od signala iz date grupe ne treba da bude aktivan. Polje M1 se koristi za kodiranje četiri signala, pa je njegova dužina 3 bita. Polja M2, M3, M4 i M5 se koristi za kodiranje po sedam signala, pa je njihova dužina, takođe, po 3 bita. Polja M6, M7 i M8 se koristi za kodiranje po tri signala, pa je njihova dužina po dva bita. Kombinacije upravljačkih signala su tako odabrane da njima mogu da se pokriju sve situacije iz sekvence upravljačkih signala po koracima (tabele 1 i 2). Ukupan broj bitova potrebnih za kodiranje kombinacija upravljačkih signala operacione jedinice je 21. U odnosu na horizontalan način kodiranja upravljačkih signala operacione jedinice potreban je dvostuko manji broj bitova, uz isti ukupan broja koraka.

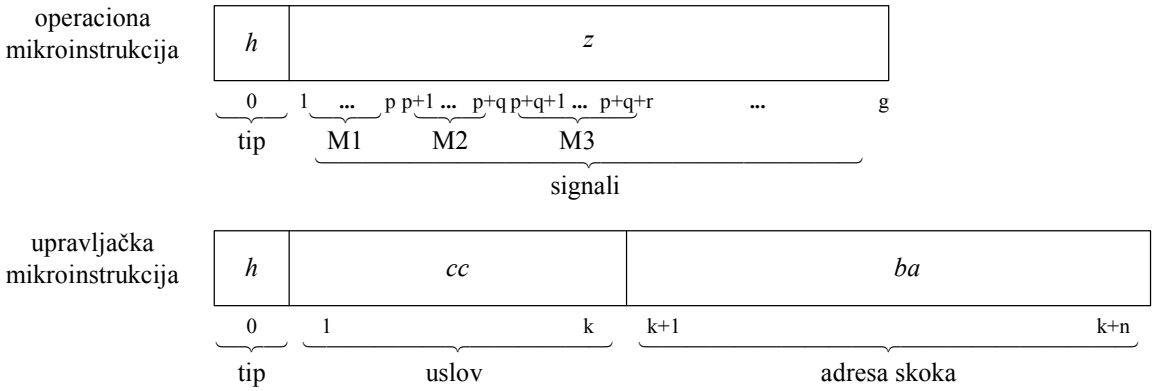
Tabela 35 Kodiranje upravljačkih signala operacione jedinice i simboličke oznake kodova

polje M1		polje M2		polje M3		polje M4	
$M1_0$	–	$M2_0$	–	$M3_0$	–	$M4_0$	–
$M1_1$	$mxMAR_2$	$M2_1$	$mxMAR_1$	$M3_1$	$mxMAR_0$	$M4_1$	lMAR
$M1_2$	add	$M2_2$	mxB_1	$M3_2$	mxB_0	$M4_2$	ldB
$M1_3$	and	$M2_3$	mxX_1	$M3_3$	mxX_0	$M4_3$	ldX
$M1_4$	asr	$M2_4$	mxPC	$M3_4$	wrGPR	$M4_4$	ldZ
$M1_5$	/	$M2_5$	ldIR1	$M3_5$	mxACC	$M4_5$	incB
$M1_6$	/	$M2_6$	ldIR2	$M3_6$	mxPSW	$M4_6$	decB
$M1_7$	/	$M2_7$	ldIRN	$M3_7$	wrMEM	$M4_7$	ldIVTP

polje M5		polje M6		polje M7		polje M8	
$M1_0$	–	$M2_0$	–	$M3_0$	–	$M4_0$	–
$M1_1$	decSP	$M2_1$	$mxMBR_1$	$M3_1$	$mxMBR_0$	$M4_1$	ldMBR
$M1_2$	incSP	$M2_2$	ldRSRC	$M3_2$	mxRDST	$M4_2$	ldRDST
$M1_3$	ldSP	$M2_3$	mxY_1	$M3_3$	mxY_0	$M4_3$	ldY
$M1_4$	incPC						
$M1_5$	ldPC						
$M1_6$	ldACC						
$M1_7$	ldPSW						

Kodom $M1_1$ se definiše da je vrednost polja M1 jedan i da je, iz grupe signala $mxMAR_2$, add, and i asr koji mogu da se specificiraju poljem M1, signal $mxMAR_2$ aktivan a ostali su neaktivni. Na sličan način se označavaju i vrednosti preostalih polja i time određuje po jedan signal iz svih preostalih grupa koji može da bude aktivan sa signalom $mxMAR_2$ iz grupe M1. Ukoliko za neke od grupa tada ni jedan od signala iz date grupe signala ne treba da bude aktivan, vrednost vrednost polja date grupe treba da bude nula.

U slučaju realizacije upravljačke jedinice sa dva tipa mikroinstrukcija i mešovitim kodiranjem upravljačkih signala operacione jedinice, formati mikroinstrukcija su dati na slici 18. Polja *h*, *cc* i *ba* imaju isto značenje kao i u slučaju upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice (slika 7), pri čemu su i kodovi polja *cc* za bezuslovne, uslovne i višestruke uslovne skokove isti (tabele 23, 24 i 25). Razlika je samo u poljima *z* operacionih mikroinstrukcija. U slučaju horizontalnog kodiranja upravljačkih signala operacione jedinice poseban bit polja *z* je dodeljen svakom signalu operacione jedinice, dok je u slučaju mešovitog kodiranja upravljačkih signala operacione jedinice polje *z* dužine g bita podeljeno na onoliko potpolja M1, M2, M3 itd. dužine p, q, r itd. bita, koliko ima grupa upravljačkih signala operacione jedinice. Binarnim vrednostima potpolja kodiraju se signali iz svake od grupa signala.



Slika 18 Formati operacionih i upravljačkih mikroinstrukcija za mešovito kodiranje upravljačkih signala

U slučaju razmatrane operacione jedinice sa direktnim vezama formati operacione i upravljačke mikroinstrukcije su dati na slikama 19 i 20. Format operacione mikroinstrukcije je dat na slici 19. Polje *h* je 0. Format upravljačke mikroinstrukcije je dat na slici 20. Polje *h* je 1.

0	1	2	3	4	5	6	7
0	M1	M1	M1	M2	M2	M2	M3

8	9	10	11	12	13	14	15
M3	M3	M4	M4	M4	M5	M5	M5

16	17	18	19	20	21	22	23
M6	M6	M7	M7	M8	M8	/	/

Slika 19 Operaciona mikroinstrukcija

0	1	2	3	4	5	6	7
1	/	/	/	*cc*			

8	9	10	11	12	13	14	15
ba							

Slika 20 Upravljačka mikroinstrukcija

Mikroprogram se formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima (tabela 1) formira jedna mikroinstrukcija i to operaciona ili upravljačka.

Kod formiranja operacionih mikroinstrukcija polazi se od sekvence upravljačkih signala po koracima (tabela 1) i traže koraci u kojima se javljaju upravljački signali operacione jedinice. Za svaki takav korak formira se jedna operaciona mikroinstrukcija tako što se, saglasno kombinaciji upravljačkih signala operacione jedinice koja se javlja u datom koraku i načinu njihovog kodiranja potpoljima M1 do M8 (tabela 35), formiraju vrednosti potpolja M1 do M8. Na primer, za korake $step_{00}$, $step_{01}$ i $step_{02}$

$step_{00}$ **ldMAR**, **incPC**;
$step_{01}$ **dMBR**;
$step_{02}$ **ldIR1**;

formiraju se tri operacione mikroinstrukcije

$madr_{00}$ $M4_1$, $M5_4$; ! **ldMAR**, **incPC** !
$madr_{01}$ $M8_1$; ! **ldMBR** !
$madr_{02}$ $M2_5$; ! **ldIR1** !

gde u mikroinstrukciji sa adrese 00 potpolja M4 i M5 imaju vrednosti 1 i 4, respektivno, a sva ostala potpolja vrednost 0, mikroinstrukciji sa adrese 01 potpolje M8 ima vrednosti 1, a sva ostala potpolja vrednost 0, mikroinstrukciji sa adrese 02 potpolje M2 ima vrednosti 5, a sva ostala potpolja vrednost 0, itd.

Kod formiranja upravljačkih mikroinstrukcija primenjuje se identičan postupak kao i u slučaju upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice.

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 1), formiran mikroprogram (tabela 36). On ima sledeću formu:

- na levoj strani se nalaze adrese mikroinstrukcija u mikroprogramskoj memoriji u heksadekadnom obliku,
- u sredini su ili operacione mikroinstrukcije, predstavljene simboličkim oznaka polja M1 do M8 za kombinacije upravljačkih signala operacione jedinice, ili upravljačke mikroinstrukcije, predstavljene nizom simboličkih oznaka signala za tip mikroinstrukcije cnt, bezuslovne, uslovne i višestruke uslovne skokove koji treba da budu aktivni i adresu skoka razdvojenih zapetama,
- dok komentar, u koracima gde se to radi lakšeg razumevanja smatralo korisnim, uvek počinje uskličnikom (!) i proteže se do sledećeg uskličnika (!).

Tabela 36 Mikroprogram

```
! Čitanje instrukcije !
        madr00  M41, M54;           ! ldMAR, incPC !
        madr01  M81;                ! ldMBR !
        madr02  M25;                ! ldIR1 !
        madr03  cnt, brl1, madr31;
        madr04  M41, M54;           ! ldMAR, incPC !
        madr05  M81;                ! ldMBR !
        madr06  M26;                ! ldIR2 !
        madr07  cnt, brl2, madr0F;
        madr08  M41, M54;           ! ldMAR, incPC !
        ...
        madr0E  M27;                ! ldIRN !
! Formiranje adrese i čitanje operanda !
        madr0F  cnt, bradr;
! Direktno registarsko !
        madr10  M62;                ! ldRSRC !
        madr11  M42;                ! ldB !
        madr12  cnt, bruncnd, madr31;
! Indirektno registarsko !
        madr13  M62;                ! ldRSRC !
        madr14  M31, M41;           ! mxMAR0, ldMAR !
        madr15  cnt, bruncnd, madr2C;
! Postdekrement !
        madr16  M62;                ! ldRSRC !
        madr17  M31, M41, M42;      ! mxMAR0, ldMAR, ldB !
        madr18  M46;                ! decB !
        madr19  M72, M82;           ! mxRDST, ldRDST !
        madr1A  M34;                ! wrGPR !
```

```
        madr1B  cnt, bruncnd, madr2C;
! Preinkrement !
        madr1C  M62;                    ! ldRSRC !
        madr1D  M42;                    ! ldB !
        madr1E  M45;                    ! incB !
        madr1F  M11, M21, M41, M72, M82;
                        ! mxMAR2, mxMAR1, ldMAR, mxRDST, ldRDST !
        madr20  M34;                            ! wrGPR !
        madr21  cnt, bruncnd, madr2C;
! Direktno memorijsko !
        madr22  M21, M41;                       ! mxMAR1, ldMAR !
        madr23  cnt, bruncnd, madr2C;
! Indirektno memorijsko!
        madr24  M21, M41;                       ! mxMAR1, ldMAR !
        madr25  M81;                            ! ldMBR !
        madr26  M21, M31, M41;                  ! mxMAR1, mxMAR0, ldMAR !
        madr27  cnt, bruncnd, madr2C;
! Indirektno registarsko sa pomerajem !
        madr28  M62;                            ! ldRSRC !
        madr29  M33, M43,M73, M83;              ! mxX0, ldX, mxY0, ldY !
        madr2A  M12, M44;                       ! add, ldZ !
        madr2B  M11, M41;                       ! mxMAR2, ldMAR !
        ! Čitanje operanda za memorijska adresiranja !
        madr2C  brSTORE, madr31;
        madr2D  M81;                            ! ldMBR !
        madr2E  M32, M42;                       ! mxB0, ldB !
        madr2F  cnt, bruncnd, madr31;
! Neposredno !
        madr30  M22, M42;                       ! mxB1, ldB !

! Izvršavanje operacije !

        madr31  cnt, bropr;
! LOAD !
        madr32  M35, M56;                       ! mxACC, ldACC !
        madr33  cnt, bruncnd, madr56;
! STORE !
        madr34  cnt, brimmed, madr56;
        madr35  cnt, brregdir, madr39;
        madr36  M71, M81;                       ! mxMBR0, ldMBR !
        madr37  M37;                            ! wrMEM !
        madr38  cnt, bruncnd, madr56;
        madr39  M82;                            ! ldRDST !
        madr3A  M34;                            ! wrGPR !
        madr3B  cnt, bruncnd, madr56;
! ADD !
        madr3C  M43, M83;                       ! ldX, ldY !
        madr3D  M12, M44;                       ! add, ldZ !
        madr3E  M56;                            ! ldACC !
        madr3F  cnt, bruncnd, madr56;
! AND !
        madr40  M43, M83;                       ! ldX, ldY !
        madr41  M13, M44;                       ! and, ldZ !
        madr42  M56;                            ! ldACC !
```

```
        madr43  cnt, bruncnd, madr56;
! ASR !
        madr44  M83;                        ! ldY !
        madr45  M11, M44;                   ! asr, ldZ !
        madr46  M56;                        ! ldACC !
        madr47  cnt, bruncnd, madr56;
! JZ !
        madr48  cnt, breql, madr4C;
        madr49  cnt, bruncnd, madr56;
! JSR !
        madr4A  M11, M31, M41, M51, M61, M81;
                        ! mxMAR2, mxMAR0, ldMAR, decSP, mxMBR1, ldMBR !
        madr4B  M37;                        ! wrMEM !
! JMP !
        madr4C  M55;                        ! ldPC !
        madr4D  cnt, bruncnd, madr56;
! RTI !
        madr4E  M52;                        ! incSP !
        madr4F  M11, M31, M41;              ! mxMAR2, mxMAR0, ldMAR !
        madr50  M81;                        ! ldMBR !
        madr51  M57;                        ! ldPSW !
! RTS !
        madr52  M52;                        ! incSP !
        madr53  M11, M31, M41;              ! mxMAR2, mxMAR0, ldMAR !
        madr54  M81;                        ! ldMBR !
        madr55  M24, M55;                   ! mxPC, ldPC !

! Opsluživanje prekida !

        madr56  cnt, brnotPREKID, madr00;
        madr57  M11, M31, M41, M51, M61, M81;
                        ! mxMAR2, mxMAR0, ldMAR, decSP, mxMBR1, ldMBR !
        madr58  M37;                        ! wrMEM !
        madr59  M11, M31, M41, M51, M61, M71, M81;
                        ! mxMAR2, mxMAR0, ldMAR, decSP, mxMBR1, mxMBR0, ldMBR !
        madr5A  M37;                        ! wrMEM !
        madr5B  M23, M43, M63, M83;         ! mxX1, ldX, mxY1, ldY !
        madr5C  M12, M44;                   ! add, ldZ !
        madr5D  M11, M41;                   ! mxMAR2, ldMAR !
        madr5E  M81;                        ! ldMBR !
        madr5F  M24, M55;                   ! mxPC, ldPC !
        madr60  cnt, bruncnd, madr00;
```

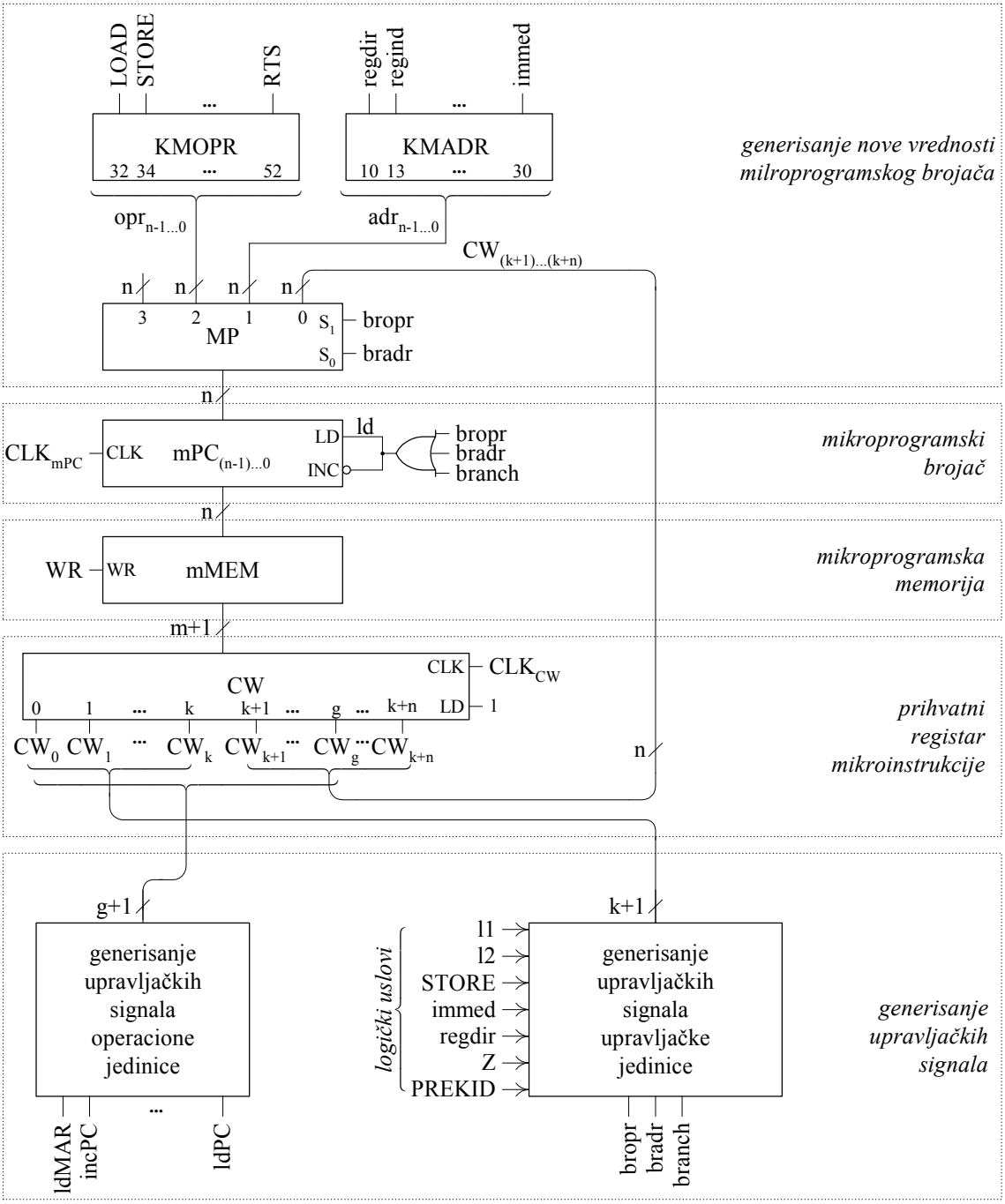
Struktura upravljačke jedinice je slična strukturi upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice i data je na slici 21. Mikroprogramski brojač se na isti način inkrementira, u mikroprogramski brojač se na isti način upisuje nova vrednost i mikroprogramski brojač prolazi kroz isti broj stanja. Iste su i vrednosti koje generišu kombinacione mreže KMOPR i KMADR i vrednosti $CW_{k+1...k+n}$ koje se upisuju iz upravljačke mikroinstrukcije.

Upravljački signali operacione jedinice se generišu ukoliko je $\mathbf{CW_0} = 0$, jer se tada u prihvatnom registru mikroinstrukcije $CW_{0...k+n}$ nalazi operaciona mikroinstrukcija. Upravljački signali operacione jedinice se generišu na sledeći način (tabela 35):

- $\mathbf{ldMAR} = \overline{\mathbf{CW}}_0 \cdot \overline{\mathbf{CW}}_{10} \cdot \overline{\mathbf{CW}}_{11} \cdot \mathbf{CW_{12}}$

- **mxMAR$_0$** = $\overline{\mathbf{CW}}_0 \cdot \overline{\mathbf{CW}}_7 \cdot \overline{\mathbf{CW}}_8 \cdot \mathbf{CW_9}$
- **wrGPR** = $\overline{\mathbf{CW}}_0 \cdot \mathbf{CW_7} \cdot \overline{\mathbf{CW}}_8 \cdot \overline{\mathbf{CW}}_9$ itd.

Na identičan način se generišu i preostali upravljački signali operacione jedinice.



Slika 21 Struktura upravljačke jedinice mikroprogramske realizije sa dva tipa mikroinstrukcija i vertikalnim formatom mikroinstrukcija

Upravljački signali upravljačke jedinice se generišu na identičan način kao i u slučaju upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice, jer je format upravljačkih mikroinstrukcija identičan, i to:

- **bropr** = $\mathbf{CW_0} \cdot \mathbf{CW_4} \cdot \overline{\mathbf{CW}}_5 \cdot \overline{\mathbf{CW}}_6 \cdot \mathbf{CW_7}$
- **bradr** = $\mathbf{CW_0} \cdot \mathbf{CW_4} \cdot \overline{\mathbf{CW}}_5 \cdot \overline{\mathbf{CW}}_6 \cdot \overline{\mathbf{CW}}_7$

- **branch** = **bruncnd**
  + **brl1*l1** + **brl2*l2** + **brSTORE*STORE** + **brimmed*immed**
  + **brregdir*regdir** + **brZ*Z**+ **brnotPREKID*** $\overline{\text{PREKID}}$

Signali koji se javljaju u izrazu za signal **branch** se generišu na sledeći način:

- **bruncnd** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot \overline{CW}_6 \cdot CW_7}$
- **brl1** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot CW_6 \cdot \overline{CW}_7}$
- **brl2** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot CW_6 \cdot CW_7}$
- **brSTORE** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot \overline{CW}_7}$
- **brimmed** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot CW_7}$
- **breql** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot CW_6 \cdot \overline{CW}_7}$
- **brnotPREKID** = $\mathbf{CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot CW_6 \cdot CW_7}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **STORE**, **immed**, **regdir**, **eql** i **PREKID**.

$$$$

Moguće je mešovito kodiranje signala i za slučaj kada postoji spajanje koraka i samo jedan tip mikroinstrukcije.

$$$$

1.3.2.3.3 Mikroprogramska realizacija sa nanoprogramiranjem

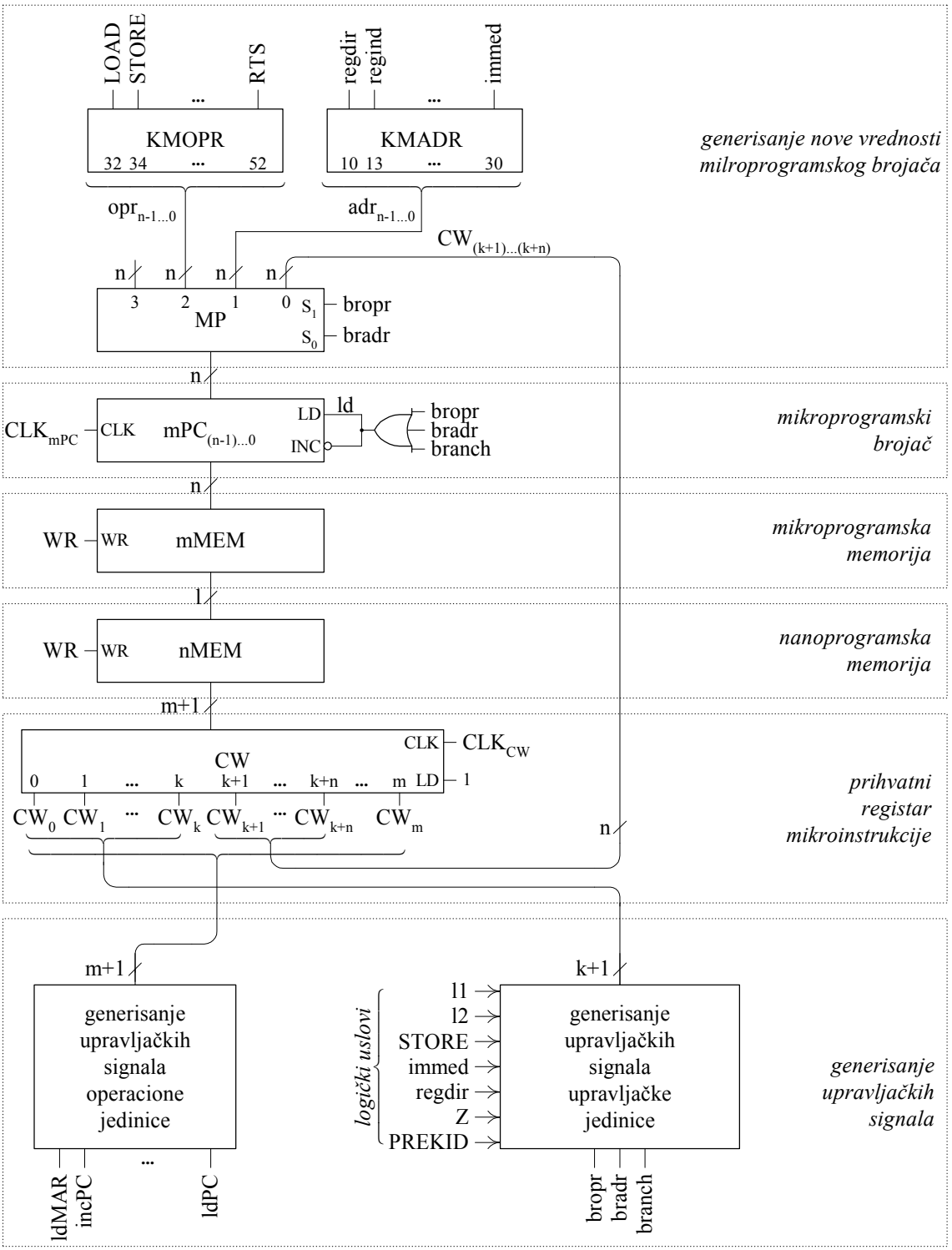
U ovom odeljku se razmatra mikroprogramska realizacija upravljačke jedinice sa nanoprogramiranjem za slučaj da postoje dva tipa mikroinstrukcija i da se koristi horizontalno kodiranje upravljačkih signala operacione jedinice. Do strukture šeme se dolazi tako što se u strukturnoj šemi upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice (slika 10) između mikroprogramske memorije mMEM i prihvatnog registra mikroinstrukcije CW ubacuje nanoprogramska memorija nMEM (slika 22).

Na osnovu mikroprograma za upravljačku jedinicu sa horizontalnim kodiranjem (tabela 30) formiraju se nanoprogram i mikroprogram za upravljačku jedinicu sa horizontalnim kodiranjem mikroinstrukcija i nanoprogramiranjem. Oni se smeštaju u nanoprogramsku memoriju i mikroprogramsku memoriju upravljačke jedinice sa horizontalnim kodiranjem mikroinstrukcija i nanoprogramiranjem (slika 22).

Nanoprogram se formira od mikroinstrukcija iz mikroprograma za upravljačku jedinicu sa horizontalnim formatom mikroinstrukcija (tabela 30) tako što se svaka mikroinstrukcija koja se u njemu javlja stavi u nanoprogram ali samo jedanput. Tako formiran nanoprogram za upravljačku jedinicu sa horizontalnim formatom mikroinstrukcija i nanoprogramiranjem je dat u tabeli 37.

Mikroprogram se formira na osnovu mikroprograma za upravljačku jedinicu sa horizontalnim formatom mikroinstrukcija (tabela 30) i nanoprograma za upravljačku jedinicu sa horizontalnim formatom mikroinstrukcija i nanoprogramiranjem (tabela 37) tako što se svaka mikroinstrukcija iz tabele 30 zameni adresom te mikroinstrukcije u tabeli 37. Tako formiran mikroprogram za upravljačku jedinicu sa horizontalnim formatom mikroinstrukcija i

nanoprogramiranjem je dat u tabeli 38. U njemu je sa $nadr_j$ simbolički označena adresa j nanoprogramske memorije.



Slika 22 Struktura upravljačke jedinice sa horizontalnim formatom mikroinstrukcija i nanoprogramiranjem

Strukture upravljačke jedinice sa horizontalnim formatom mikroinstrukcija i upravljačke jedinice sa horizontalnim formatom mikroinstrukcija i nanoprogramiranjem su veoma slične. Razlike su samo u delu gde se u prvom slučaju nalazi mikroprogramska memorija (slika 10), a u drugom mikroprogramska memorija i nanoprogramska memorija (slika 22). U prvom slučaju se na osnovu vrednosti mikroprogramskog brojača mPC iz mikroprogramske

memorije čita mikroinstrukcija i upisuje u prihvatni registar mikroinstrukcije CW. U drugom slučaju se, najpre, na osnovu vrednosti mikroprogramskog brojača mPC iz mikroprogramske memorije čita adresa sa koje se iz nanoprogramske memorije čita mikroinstrukcija i upisuje u prihvatni registar mikroinstrukcije CW. Pošto je u oba slučaja struktura mikroinstrukcija identična, dalje se na identičan način u oba slučaja generišu i upravljački signali operacione jedinice i upravljački signali upravljačke jedinice. Sve ovo ima dalje za posledicu da se na identičan način upravlja radom blokova *generisanje nove vrednosti mikroprogramskog brojača* i *mikroprogramski brojač*.

Tabela 37 Nanoprogram

$nadr_{00}$	**ldMAR**, **incPC**;
$nadr_{01}$	**ldMBR**;
$nadr_{02}$	**ldIR1**;
$nadr_{03}$	**cnt**, **brl1**, **madr$_{31}$**;
$nadr_{04}$	**ldIR2**;
$nadr_{05}$	**cnt**, **brl2**, **madr$_{0F}$**;
$nadr_{06}$	**ldIRN**;
$nadr_{07}$	**cnt**, **bradr**;
$nadr_{08}$	**ldRSRC**;
$nadr_{09}$	**ldB**;
$nadr_{0A}$	**cnt**, **bruncnd**, **madr$_{31}$**;
$nadr_{0B}$	**mxMAR$_0$**, **ldMAR**;
$nadr_{0C}$	**cnt**, **bruncnd**, **madr$_{2C}$**;
$nadr_{0D}$	**mxMAR$_0$**, **ldMAR**, **ldB**;
$nadr_{0E}$	**decB**;
$nadr_{0F}$	**mxRDST**, **ldRDST**;
$nadr_{10}$	**wrGPR**;
$nadr_{11}$	**incB**;
$nadr_{12}$	**mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**, **mxRDST**, **ldRDST**;
$nadr_{13}$	**mxMAR$_1$**, **ldMAR**;
$nadr_{14}$	**mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**;
$nadr_{15}$	**mxX$_0$**, **ldX**, **mxY$_0$**, **ldY**;
$nadr_{16}$	**add**, **ldZ**;
$nadr_{17}$	**mxMAR$_2$**, **ldMAR**;
$nadr_{18}$	**brSTORE**, **madr$_{31}$**;
$nadr_{19}$	**mxB$_0$**, **ldB**;
$nadr_{1A}$	**mxB$_1$**, **ldB**;
$nadr_{1B}$	**cnt**, **bropr**;
$nadr_{1C}$	**mxACC**, **ldACC**;
$nadr_{1D}$	**cnt**, **bruncnd**, **madr$_{56}$**;
$nadr_{1E}$	**cnt**, **brimmed**, **madr$_{56}$**;
$nadr_{1F}$	**cnt**, **brregdir**, **madr$_{39}$**;
$nadr_{20}$	**mxMBR$_0$**, **ldMBR**;
$nadr_{21}$	**wrMEM**;
$nadr_{22}$	**ldRDST**;
$nadr_{23}$	**ldX**, **ldY**;
$nadr_{24}$	**ldACC**;
$nadr_{25}$	**and**, **ldZ**;
$nadr_{26}$	**ldY**;
$nadr_{27}$	**asr**, **ldZ**;
$nadr_{28}$	**cnt**, **breql**, **madr$_{4C}$**;
$nadr_{29}$	**mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR**;

$nadr_{2A}$ **ldPC**;
$nadr_{2B}$ **incSP**;
$nadr_{2C}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**;
$nadr_{2D}$ **ldPSW**;
$nadr_{2E}$ **mxPC**, **ldPC**;
$nadr_{2F}$ **cnt**, **brnotPREKID**, **madr$_{00}$**;
$nadr_{30}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **mxMBR$_0$**,**ldMBR**;
$nadr_{31}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
$nadr_{32}$ **cnt**, **bruncnd**, **madr$_{00}$**;

Tabela 38 Mikroprogram

! Čitanje instrukcije !

$madr_{00}$ $nadr_{00}$; ! **ldMAR**, **incPC** !
$madr_{01}$ $nadr_{01}$; ! **ldMBR** !
$madr_{02}$ $nadr_{02}$; ! **ldIR1** !
$madr_{03}$ $nadr_{03}$; ! **cnt**, **brl1**, **madr$_{31}$** !
$madr_{04}$ $nadr_{00}$; ! **ldMAR**, **incPC** !
$madr_{05}$ $nadr_{01}$; ! **ldMBR** !
$madr_{06}$ $nadr_{04}$; ! **ldIR2** !
$madr_{07}$ $nadr_{05}$; ! **cnt**, **brl2**, **madr$_{0F}$** !
$madr_{08}$ $nadr_{00}$; ! **ldMAR**, **incPC** !
...
$madr_{0E}$ $nadr_{06}$; ! **ldIRN** !

! Formiranje adrese i čitanje operanda !

$madr_{0F}$ $nadr_{07}$; ! **cnt**, **bradr** !

! Direktno registarsko !

$madr_{10}$ $nadr_{08}$; ! **ldRSRC** !
$madr_{11}$ $nadr_{09}$; ! **ldB** !
$madr_{12}$ $nadr_{0A}$; ! **cnt**, **bruncnd**, **madr$_{31}$** !

! Indirektno registarsko !

$madr_{13}$ $nadr_{08}$; ! **ldRSRC** !
$madr_{14}$ $nadr_{0B}$; ! **mxMAR$_0$**, **ldMAR** !
$madr_{15}$ $nadr_{0C}$; ! **cnt**, **bruncnd**, **madr$_{2C}$** !

! Postdekrement !

$madr_{16}$ $nadr_{08}$; ! **ldRSRC** !
$madr_{17}$ $nadr_{0D}$; ! **mxMAR$_0$**, **ldMAR**, **ldB** !
$madr_{18}$ $nadr_{0E}$; ! **decB** !
$madr_{19}$ $nadr_{0F}$; ! **mxRDST**, **ldRDST** !
$madr_{1A}$ $nadr_{10}$; ! **wrGPR** !
$madr_{1B}$ $nadr_{0C}$; ! **cnt**, **bruncnd**, **madr$_{2C}$** !

! Preinkrement !

$madr_{1C}$ $nadr_{0B}$; ! **ldRSRC** !
$madr_{1D}$ $nadr_{09}$; ! **ldB** !
$madr_{1E}$ $nadr_{11}$; ! **incB** !
$madr_{1F}$ $nadr_{12}$; ! **mxMAR$_2$**, **mxMAR$_1$**, **ldMAR**, **mxRDST**, **ldRDST** !
$madr_{20}$ $nadr_{10}$; ! **wrGPR** !
$madr_{21}$ $nadr_{0C}$; ! **cnt**, **bruncnd**, **madr$_{2C}$** !

! Direktno memorijsko !

$madr_{22}$ $nadr_{13}$; ! **mxMAR$_1$**, **ldMAR** !
$madr_{23}$ $nadr_{0C}$; ! **cnt**, **bruncnd**, **madr$_{2C}$** !

! Indirektno memorijsko!

$madr_{24}$ $nadr_{13}$; ! **mxMAR$_1$**, **ldMAR** !
$madr_{25}$ $nadr_{01}$; ! **ldMBR** !
$madr_{26}$ $nadr_{14}$; ! **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR** !
$madr_{27}$ $nadr_{0C}$; ! **cnt**, **bruncnd**, **madr$_{2C}$** !
! Indirektno registarsko sa pomerajem !
$madr_{28}$ $nadr_{08}$; ! **ldRSRC** !
$madr_{29}$ $nadr_{15}$; ! **mxX$_0$**, **ldX**, **mxY$_0$**, **ldY** !
$madr_{2A}$ $nadr_{16}$; ! **add**, **ldZ** !
$madr_{2B}$ $nadr_{17}$; ! **mxMAR$_2$**, **ldMAR** !
! Čitanje operanda za memorijska adresiranja !
$madr_{2C}$ $nadr_{18}$; ! **brSTORE**, **madr$_{31}$** !
$madr_{2D}$ $nadr_{01}$; ! **ldMBR** !
$madr_{2E}$ $nadr_{19}$; ! **mxB$_0$**, **ldB** !
$madr_{2F}$ $nadr_{0A}$; ! **cnt**, **bruncnd**, **madr$_{31}$** !
! Neposredno !
$madr_{30}$ $nadr_{1A}$; ! **mxB$_1$**, **ldB** !

! Izvršavanje operacije !

$madr_{31}$ $nadr_{1B}$; ! **cnt**, **bropr** !
! LOAD !
$madr_{32}$ $nadr_{1C}$; ! **mxACC**, **ldACC** !
$madr_{33}$ $nadr_{1D}$; ! **cnt**, **bruncnd**, **madr$_{56}$** !
! STORE !
$madr_{34}$ $nadr_{1E}$; ! **cnt**, **brimmed**, **madr$_{56}$** !
$madr_{35}$ $nadr_{1F}$; ! **cnt**, **brregdir**, **madr$_{39}$** !
$madr_{36}$ $nadr_{20}$; ! **mxMBR$_0$**, **ldMBR** !
$madr_{37}$ $nadr_{21}$; ! **wrMEM** !
$madr_{38}$ $nadr_{1D}$; ! **cnt**, **bruncnd**, **madr$_{56}$** !
$madr_{39}$ $nadr_{22}$; ! **ldRDST** !
$madr_{3A}$ $nadr_{10}$; ! **wrGPR** !
$madr_{3B}$ $nadr_{1D}$; ! **cnt**, **bruncnd**, **madr$_{56}$** !
! ADD !
$madr_{3C}$ $nadr_{23}$; ! **ldX**, **ldY** !
$madr_{3D}$ $nadr_{16}$; ! **add**, **ldZ** !
$madr_{3E}$ $nadr_{24}$; ! **ldACC** !
$madr_{3F}$ $nadr_{1D}$; ! **cnt**, **bruncnd**, **madr$_{56}$** !
! AND !
$madr_{40}$ $nadr_{23}$; ! **ldX**, **ldY** !
$madr_{41}$ $nadr_{25}$; ! **and**, **ldZ** !
$madr_{42}$ $nadr_{24}$; ! **ldACC** !
$madr_{43}$ $nadr_{1D}$; ! **cnt**, **bruncnd**, **madr$_{56}$** !
! ASR !
$madr_{44}$ $nadr_{26}$; ! **ldY** !
$madr_{45}$ $nadr_{27}$; ! **asr**, **ldZ** !
$madr_{46}$ $nadr_{24}$; ! **ldACC** !
$madr_{47}$ $nadr_{1D}$; ! **cnt**, **bruncnd**, **madr$_{56}$** !
! JZ !
$madr_{48}$ $nadr_{28}$; ! **cnt**, **breql**, **madr$_{4C}$** !
$madr_{49}$ $nadr_{1D}$; ! **cnt**, **bruncnd**, **madr$_{56}$** !
! JSR !
$madr_{4A}$ $nadr_{29}$; ! **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**, **mxMBR$_1$**, **ldMBR** !
$madr_{4B}$ $nadr_{21}$; ! **wrMEM** !
! JMP !

$madr_{4C}$ $nadr_{2A}$; ! **ldPC** !
$madr_{4D}$ $nadr_{1D}$; ! **cnt**, **bruncnd**, $\mathbf{madr_{56}}$!
! RTI !
$madr_{4E}$ $nadr_{2B}$; ! **incSP** !
$madr_{4F}$ $nadr_{2C}$; ! $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR** !
$madr_{50}$ $nadr_{01}$; ! **ldMBR** !
$madr_{51}$ $nadr_{2D}$; ! **ldPSW** !
! RTS !
$madr_{52}$ $nadr_{2B}$; ! **incSP** !
$madr_{53}$ $nadr_{2C}$; ! $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR** !
$madr_{54}$ $nadr_{01}$; ! **ldMBR** !
$madr_{55}$ $nadr_{2E}$; ! **mxPC**, **ldPC** !

! Opsluživanje prekida !

$madr_{56}$ $nadr_{2F}$; ! **cnt**, **brnotPREKID**, $\mathbf{madr_{00}}$!
$madr_{57}$ $nadr_{29}$; ! $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**, **decSP**, $\mathbf{mxMBR_1}$, **ldMBR** !
$madr_{58}$ $nadr_{21}$; ! **wrMEM** !
$madr_{59}$ $nadr_{30}$; ! $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_0}$, **ldMAR**, **decSP**, $\mathbf{mxMBR_1}$, $\mathbf{mxMBR_0}$,**ldMBR** !
$madr_{5A}$ $nadr_{21}$; ! **wrMEM** !
$madr_{5B}$ $nadr_{31}$; ! $\mathbf{mxX_1}$, **ldX**, $\mathbf{mxY_1}$, **ldY** !
$madr_{5C}$ $nadr_{16}$; ! **add**, **ldZ** !
$madr_{5D}$ $nadr_{17}$; ! $\mathbf{mxMAR_2}$, **ldMAR** !
$madr_{5E}$ $nadr_{01}$; ! **ldMBR** !
$madr_{5F}$ $nadr_{2E}$; ! **mxPC**, **ldPC** !
$madr_{60}$ $nadr_{32}$; ! **cnt**, **bruncnd**, $\mathbf{madr_{00}}$!!

2