**ELEKTROTEHNIČKI FAKULTET**

**UNIVERZITET U BEOGRADU**

# OSNOVI RAČUNARSKE TEHNIKE 2

## PROJEKAT

## ZADATAK 2

BOJANA BELOJEVIĆ 28/07
ANA JELIČIĆ 89/07
MILAN BRANKOVIĆ 119/07
ALEKSANDAR MARJANOVIĆ 211/07

**BEOGRAD 2009.**

# SADRŽAJ

# 1 PROJEKTNI ZADATAK

Posmatra se deo računara koji čine memorija i procesor.

Memorija je kapaciteta $2^{16}$ bajtova. Širina memorijske reči je 1 bajt.

Procesor je sa jednoadresnim formatom instrukcija. Podaci su celobrojne veličine bez znaka dužine 2 bajta. Podaci u memoriji zauzimaju dve susedne memorijske lokacije, pri čemu se mlađi bajt nalazi na nižoj a stariji bajt na višoj adresi.

U procesoru postoji programski brojač PC dužine 2 bajta, adresni registar memorije MAR dužine 2 bajta, prihvatni registar podatka memorije MBR dužine 1 bajt, prihvatni registar instrukcije IR dužine 3 bajta, akumulator A dužine 2 bajta, prihvatni registar podatka B dužine 2 bajta, registri opšte namene R0…R3 dužine 2 bajta, programska statusna reč PSW dužine 1 bajt, ukazivač na vrh steka SP dužine 2 bajta, registar broja ulaza u tabelu sa adresam prekidnih rutina BRU dužine 2 bita i ukazivač na tabelu sa adresama prekidnih rutina IVTP dužine 2 bajta. Instrukcije su dužine 1 ili 3 bajta.

Bitovi 7, 6, 5 i 4 prvog bajta instrukcije su 0000 za sve instrukcije skoka, dok se bitovima 3 do 0 prvog bajta instrukcija specificira kod operacije za instrukcije skoka. Instrukcije skoka su instrukcija uslovnog skoka ukoliko je rezultat nije nula (BNZ), bezuslovnog skoka (JMP) i skoka na potprograma (JSR). Instrukcija BNZ se realizuje kao relativni skok u odnosu na tekuću vrednost programskog brojača PC, a pomeraj je 8-mo bitna celobrojna veličina sa znakom data 2. bajtom instrukcije. Dužina instrukcije je 2 bajta. Instrukcije JMP i JSR se realizuju kao apsolutni skokovi, a adresa skoka je data 2 i 3 bajtom instrukcije, pri čemu je mlađi bajt adrese skoka dat drugim a stariji bajt trećim bajtom. Dužina instrukcija je 3 bajta.

Bitovi 7, 6, 5 i 4 prvog bajta instrukcije su 1111 za bezadresne instrukcije, dok se bitovima 3 do 0 prvog bajta instrukcija specificira kod operacije za bezadresne instrukcije. Bezadresne instrukcije su instrukcije povratka iz potprograma (RTS), povratka iz prekidne rutine (RTI), stavljanja sadržaja akumulatora na stek (PUSH) i skidanja sadržaja sa steka i punjenje akumulatora (POP). Dužina instrukcija je 1 bajt.

Bitovi 7, 6, 5 i 4 prvog bajta instrukcije u opsegu vrednosti 0001 do 1110 specificiraju kod operacije za adresne instrukcije. Adresne instrukcije su instrukcije prenosa u akumulator (LOAD), instrukcije prenosa iz akumulatora (STORE), aritmetička instrukcija oduzimanja (SUB), logička instrukcija logička suma (OR), instrukcija aritmetičkog pomeranja ulevo za jedno mesto (ASL) i instrukcija bezuslovnog indirektnog skoka (JMPIND). U instrukciji STORE nije dozvoljeno neposredno adresiranje, a u instrukciji JMPIND nije dozvoljeno registarsko direktno i neposredno adresiranje, pa ukoliko se jave ova adresiranja u ovim instrukcijama, instrukcije treba da budu bez dejstva. Dužina instrukcija je 1, 2, ili 3 bajta i zavisi od specificiranog načina adresiranja.

Načini adresiranja su specificirani bitovima 3 i 2 prvog bajta instrukcije i to na sledeći način: 00-neposredno adresiranje, 01-memorijsko direktno adresiranje, 10-registarsko indirektno adresiranje sa pomerajem i 11- registarsko direktno adresiranje. Kod neposrednog adresiranja 16-to bitni operand je dat drugim i trećim bajtom instrukcije, pri čemu je mlađi bajt operanda dat drugim a stariji bajt trećim bajtom. Bitovi 1 i 0 prvog bajta instrukcije se ne koriste. Dužina instrukcija je 3 bajta. Kod memorijskog direktnog adresiranja 16-to bitna adresa memorijske lokacije je data drugim i trećim bajtom instrukcije, pri čemu je mlađi bajt adrese dat drugim a stariji bajt trećim bajtom. Bitovi 1 i 0 prvog bajta instrukcije se ne koriste. Dužina instrukcija je 3 bajta. Kod registarskog indirektnog adresiranja sa pomerajem 8-mo bitni pomeraj je celobrojna veličina sa znakom u drugom komplementu data drugim bajtom

instrukcije. Bitovi 1 i 0 prvog bajta instrukcije se koriste za adresiranje jednog od registara opšte namene R0 do R3. Dužina instrukcija je 2 bajta. Kod registarskog direktnog adresiranja bitovi 1 i 0 prvog bajta instrukcije se koriste za adresiranje jednog od registara opšte namene R0 do R3. Dužina instrukcija je 1 bajt.

Stek raste prema nižim memorijskim lokacijama, a registar SP ukazuje na zadnu zauzetu 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 BRU 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. Adrese dužine 16 bita zauzimaju po dve susedne memorijske lokacije, pri čemu se mlađi bajt nalazi na nižoj a stariji bajt na višoj adresi Sadržaj registra BRU predstavlja broj ulaza u tabelu sa adresam pekidnih rutina. Početna adresa tabele sa adresama prekidnih rutina se nalazi u registru IVTP dužine 2 bajta. U okviru hardverskog dela opsluživanja zahteva za prekid na stek sa stavljaju samo registri PC i PSW.

ZADATAK

Nacrtati i objasniti dijagram toka faza izvršavanja instrukcije i to: faze čitanja instrukcije, faze formiranja adrese i čitanja operanda, faza izvršavanja operacija LOAD, STORE, PUSH, POP, SUB, OR, ASL, BNZ, JMP, JMPIND, JSR, RTS i RTI i faze opsluživanja zahteva za prekid.

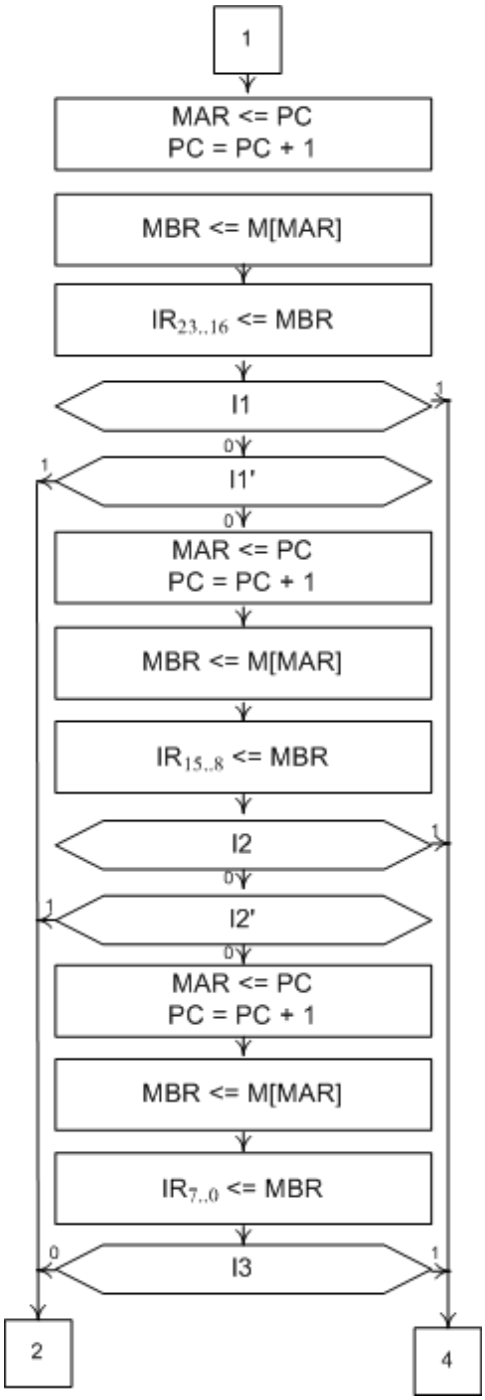
Isprojektovati za dati procesor operacionu jedinicu realizovanu sa direktnim vezama i upravljačke jedinice ožičene realizacije bez i sa spajanjem koraka i mikroprogramske realizacije i horizontalnim kodiranjem upravljačkih signala sa dva tipa i jednim tipom mikroinstrukcija.

# 2 DIJAGRAM TOKA IZVRŠAVANJA INSTRUKCIJA

U ovoj glavi se daje dijagram toka faza izvršavanja instrukcije i to: 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 i IR3 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 jednu, dve ili tri reči i to prvu reč koja specificira kod operacije i specificira način adresiranja i adresu registra opšte namene. Kod registarskog indirektnog adresiranja sa pomerajem dužina instrukcije je dve reči (dva bajta), kod registarskog direktnog adresiranja dužina instrukcija je 1 bajt, kod memorijskog direktnog adresiranja duzina instrukcije je 3 bajta, kao i kod neposrednog adresiranja. Posle čitanja dovoljnog broja reči kod registarskih i memorijskih adresiranja moze se preći na *formiranje adrese i čitanje operanda*. Prilikom čitanja reči instrukcije formiraju se signali dužine instrukcije l1,l1',l2',l2 i l3, koji predstavljaju signale logičkih uslova. Ovi signali označavaju da je dužina instrukcije jedna, dve ili tri 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, l2 i l3. Nakon pročitane prve reči zna se koji način adresiranja se koristi. Ako je u pitanju registarsko adresiranje zna se koji registar se koristi. Ako je, na primer, dužina adresnih instrukcija sa memorijskim adresiranjima tri reči, svi signali dužine instrukcije treba da budu neaktivani.




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 se, zatim, sa adrese određene sadržajem registra MAR čita reč i upisuje u registar MBR. Sadržaj registra MBR se, na kraju, prebacuje u registar $IR_{23...16}$.

Sada se vrši provera signala logičkog uslova **BEZADRESNE ILI REGISTARSKO DIREKTNO** $l1 = IR_{23} \cdot IR_{22} \cdot IR_{21} \cdot IR_{20}$ i $l1' = IR_{19} \cdot IR_{18}$.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 registarsko direktnom adresiranju signal l1' je aktivan, pa se prelazi na korak 2 i fazu *formiranje adrese i čitanje operanda* (slika 1.b).Ukoliko se radi o instrukcijama skoka ili adresnim instrukcijama signali l1 i l1' su neaktivni, 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 $IR_{15...8}$.

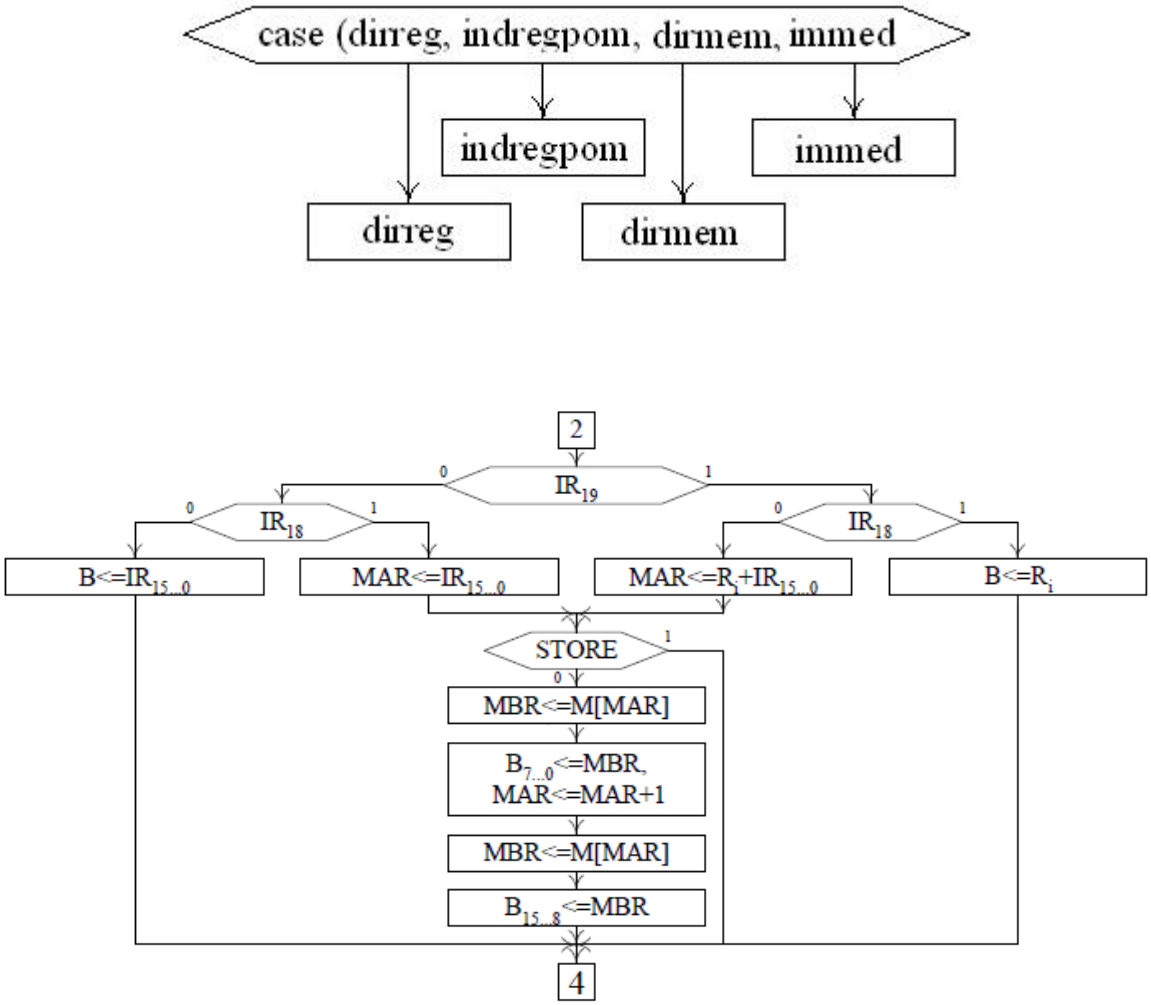
Sada se vrši provera signala logičkog uslova **BNZ ILI REG IND POM** $l2 = IR_{23} \cdot IR_{22} \cdot IR_{21} \cdot IR_{20}$ i $l2' = IR_{19} \cdot \overline{IR}_{18}$.Ukoliko se radi o instrukciji uslovnog skoka ukoliko je rezultat nije nula signal l2 je aktivan, pa se prelazi na korak 4 i fazu *izvršavanje operacije* (slika 1.c). Ukoliko se radi o registarsko indirektnom adresiranju sa pomerajem signal l2' je aktivan, pa se prelazi na korak 2 i fazu *formiranje adrese i čitanje operanda* (slika 1.b). Ukoliko se radi o instrukcijama skoka ili adresnim instrukcijama signali l2 i l2' su neaktivni, pa se prelazi na čitanje treće reči instrukcije. Čitanje treće reči instrukcije se realizuje na isti način kao i čitanje prve i druge reči instrukcije. Treća reč instrukcije se upisuje u registar $IR_{7...0}$.

Sada se vrši provera signala logičkog uslova **JUMP** $l3 = \overline{IR}_{23} \cdot \overline{IR}_{22} \cdot \overline{IR}_{21} \cdot \overline{IR}_{20}$.Ukoliko se radi o instrukciji skoka 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 signal l3 je neaktivan, pa se prelazi na korak 2 i fazu *formiranje adrese i čitanje operanda* (slika 1.b).

*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, dirmem, indregpom i immed određuje da je specificirano registarsko direktno adresiranje, memorijsko direktno adresiranje, registarsko indirektno adresiranje sa pomerajem i neposredno adresiranje respektivno. $_{7..0}$




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

Ukoliko je signal dirreg = $IR_{19}\cdot IR_{18}$ aktivan, radi se o registarskom direktnom adresiranju. Operand je tada u registru opšte namene $R_i$ određenom vrednošću bitova 1 i 0 prve reči instrkcijskog registra IR. 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 dirmem = $\overline{IR}_{19}\cdot IR_{18}$ aktivan, radi se o memorijskom direktnom adresiranju. Operand je tada u memoriji na adresi koja se nalazi u razredima IR2 i IR3 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 indregpom = $IR_{19} \cdot \overline{IR}_{18}$ 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 i 8-mo bitnog pomeraja koji je celobrojna veličina sa znakom u drugom komplementu data drugim bajtom instrukcije. 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 = $\overline{IR}_{19} \cdot \overline{IR}_{18}$ 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, SUB, OR, ASL, BNZ, JMP, JSR, JMPIND, RTI, RTS, PUSH ili POP određuje da je specificirana operacija prenosa u akumulator, operacija prenosa iz akumulatora, aritmetička operacija oduzimanja, logička operacija logičkog sabiranja, operacija aritmetičkog pomeranja ulevo za jedno mesto, operacija uslovnog skoka ukoliko rezultat nije nula, operacija bezuslovnog skoka, operacija skoka na potprograma, operacija bezuslovnog skoka u odnosu na pomeraj, operacija povratka iz prekidne rutine, operacija povratka iz potprograma, operacija stavljanja na stek i operacija skidanja sa steka, 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).

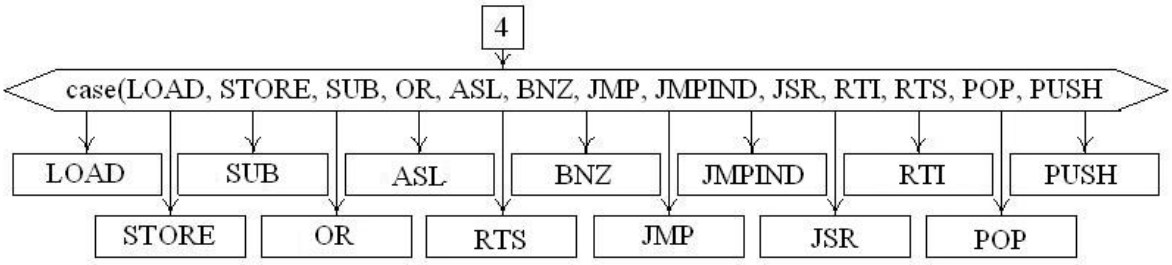
Ukoliko je signal STORE aktivan, sadržaj registra akumulatora A se prebacuje u registar opšte namene $R_i$ određen vrednošću bitova 1 i 0 prve reči instrukcije iz registra IR1, ukoliko je specificirano direktno registarsko adresiranje ili u memorijsku lokaciju, čija se adresa nalazi u registru MAR, ukoliko je specificirano memorijsko direktno adresiranje. 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 bitova 1 i 0 prve reči instrukcije iz registra IR1 i prelazi na korak 5 i fazu *opsluživanje prekida* (slika 1.d). Ukoliko nije, memorijsko direktno adresiranje 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 5 i fazu *opsluživanje prekida* (slika 1.d).

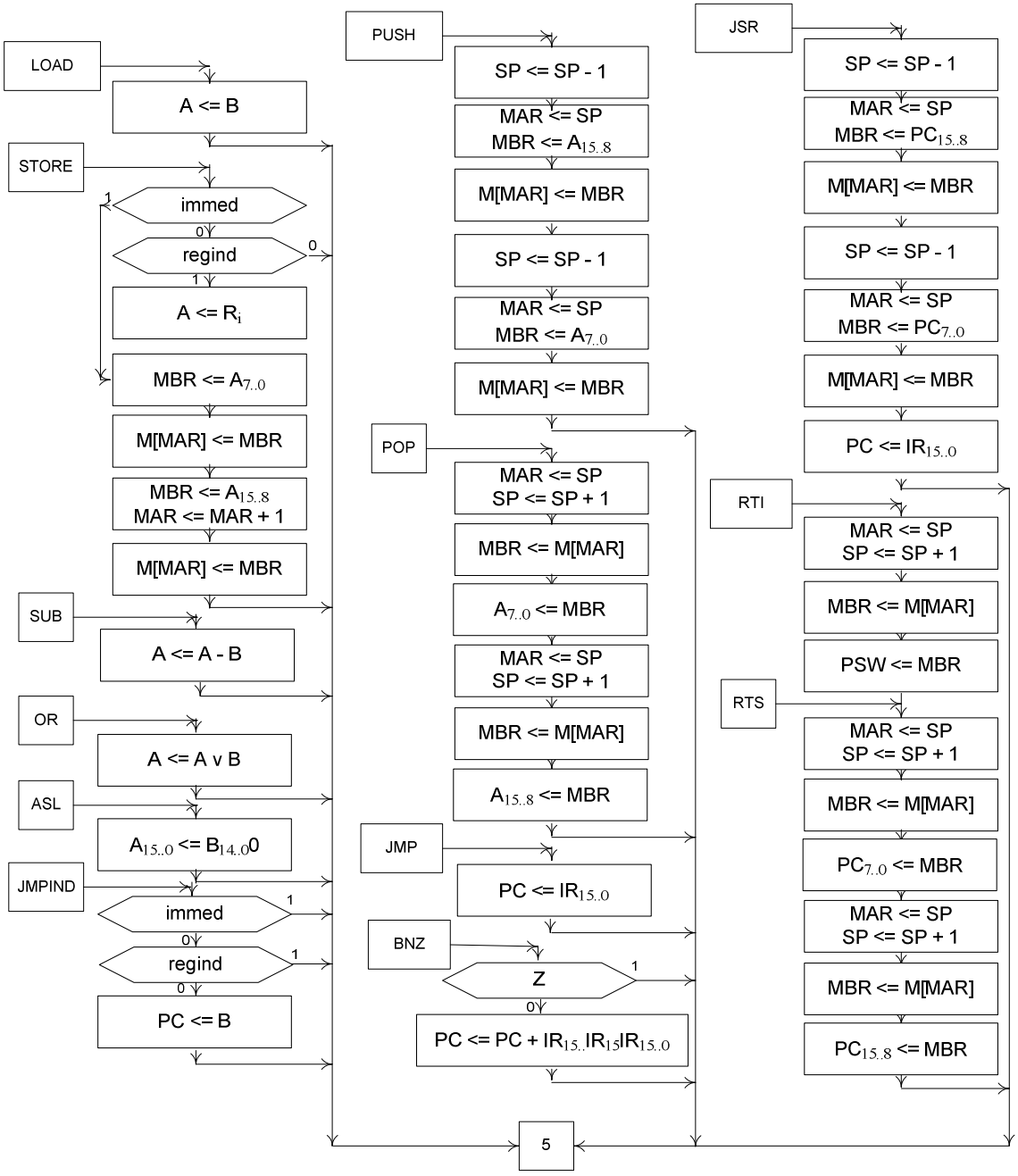
Ukoliko je signal SUB aktivan, oduzimaju se sadržaji registara A i B i rezultat upisuje u registar A. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal OR aktivan, logička ILI 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 ASL aktivan, sadržaj B se aritmetički pomera ulevo 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 BNZ aktivan, uslovni skok na osnovu vrednosti signala logičkog uslova rezultata operacija neq 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 neq ima vrednost 1 ukoliko je rezultat zadnje izvršene instrukcije je različit od 0 i vrednost 0 ukoliko je rezultat zadnje izvršene instrukcije jednak 0. Ukoliko je signal neq 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 neq 1, uslov za skok je ispunjen, najpre se vrsi prosirivanje jer se 16bitni podatak(sadrzaj PC) sabira sa 8bitnim pomerajem datim drugim bajtom instrukcije.U tu svhu vrsimo proveru bita IR15.Ako je on 1 PC sabiramo sa 11111111IR15..8,a ako je 0 onda sa 00000000IR15..8. U fazi dohvatanja operanda smo u registar B dovukli adresu sa koje treba procitati adresu sledece instrukcije.Zato registar B saljemo u MAR i zatim sadrzaj memorijskih lokacije M[MAR] i M[MAR+1] preko pomocnog registra MBR upisujemo u PC kao adresu sledece instrukcije. 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 $IR_{15...0}$, 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 dekrementira registar SP, a zatim se prebacuje sadržaj tog registra u registar MAR i prebacuje sadržaj registra PC (stariji bajt) u registar MBR. Zatim se sadržaj registra MBR upisuje u memorijsku lokaciju određenu sadržajem registra MAR. Potom se isto uradi sa mlađim bajtom registra PC. Na kraju se sadržaj registara IR2 i IR3 označeni sa $IR_{15..8}$, koji predstavlja adresu skoka, upisuje se u registar PC. Treba uočiti da stek raste prema nižim lokacijama i da registar SP ukazuje na poslednju popunjenu lokaciju. S toga se prilikom upisa na stek, prvo sadržaj registra SP dekrementira 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 JMPIND aktivan, instrukcija bezuslovnog indirektnog skoka se realizuje. Sadržaj registra B, koji predstavlja adresu skoka, upisuje se u registar PC.Neposredno adresiranje nije dozvoljeno za odredišni operand i zato se uzima da se u slučaju da se u instrukciji JMPIND 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. Registarsko direktno adresiranje nije dozvoljeno za odredišni operand i zato se uzima da se u slučaju da se u instrukciji JMPIND pojavi registarsko direktno adresiranje, faza izvršavanja ove operacije preskače, čime se ova instrukcija pretvara u instrukciju bez dejstva. S toga se proverava da li je signal regdir aktivan. Ukoliko jeste, specificirano je registarsko direktno adresiranje, pa se faza *izvršavanja operacija* završava i prelazi 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 prebacuje u registar MAR i inkrementira sadržaj registra SP. 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 poslednju popunjenu lokaciju. S toga se prilikom čitanja sadržaja sa steka, prvo prebacuje sadržaj registra SP u registar MAR i posle toga inkrementira sadržaj registra SP. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

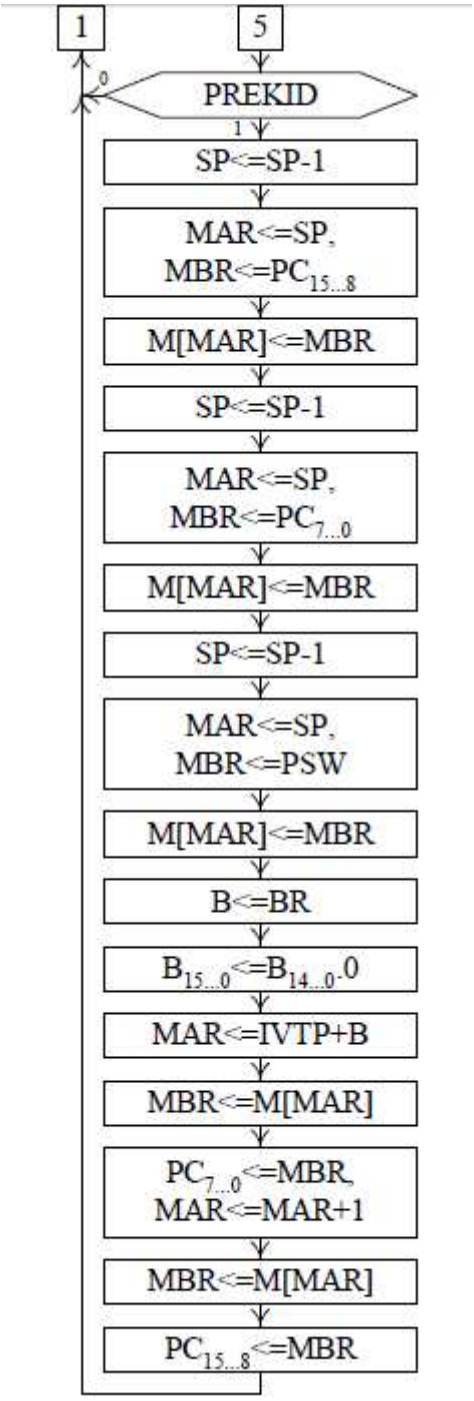
Ukoliko je signal RTS aktivan, povratak iz potprograma se realizuje. U okviru toga se sadržajem sa steka restaurira sadržaj registra PC. Ovo se realizuje na identičan kao i u slučaju instrukcije RTI. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika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

Ukoliko je signal POP aktivan, skidanje sa steka se realizuje. U okviru toga se sadržaj sa steka upisuje u akumulator. Najpre se sadržaj registra SP prebacuje u registar MAR. Zatim se iz memorijske lokacije određene sadržajem registra MAR čita sadržaj i upisuje u registar MBR, pa se inkrementira sadrzaj registra SP. Na kraju se sadržaj registra MBR upisuje u niži razred registara ACC. Na isti način se sa steka čita još jedna reč i upisuje u visi razred registara ACC. Treba uočiti da stek raste prema nižim lokacijama i da registar SP ukazuje na poslednju zauzetu lokaciju. S toga se prilikom čitanja sadržaja sa steka, prvo prebacuje u registar MAR i posle toga inkrementira sadržaj registra SP. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal PUSH aktivan, stavljanje na stek se realizuje. U okviru toga se sadržaj registra ACC stavlja na stek. Najpre se dekrementira sadržaj registra SP , prebacuje sadržaja registar SP u registar MAR, i prebacuje sadržaj visi bajt registra ACC u registar MBR. Zatim se sadržaj registra MBR upisuje u memorijsku lokaciju određenu sadržajem registra MAR. Na isti način se nizi bajt registara ACC stavlja na stek. Treba uočiti da stek raste prema nižim lokacijama i da registar SP ukazuje na poslednju zauzetu lokaciju. S toga se prilikom čitanja sadržaja sa steka, prvo prebacuje u registar MAR i posle toga inkrementira sadržaj registra SP. 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.

# 3 OPERACIONA JEDINICA

Operacione jedinice procesora identične arhitekture mogu da budu realizovane na više različitih načina. Najčešće se sreću realizacije operacionih jedinica kod kojih su prekidačke mreže povezane direktno i pomoću jedne, dve i tri interne magistrale. U ovom odeljku se daju realizacija operacione jedinice sa direktnim vezama kroz prikaz strukturne šeme i sekvenca upravljačkih signala po koracima.

## 3.1 STRUKTURA OPERACIONE JEDINICE

---

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

Operaciona jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova upravljačke jedinice.

Pri realizaciji operacione jedinice koriste se sledeci 16-bitni registri: programski brojač PC adresni registar memorije MAR, akumulator ACC, prihvatni registar podatka B, registri opšte name R0, R1, R2 i R3, ukazivač na vrh steka SP, ukazivač na tabelu sa adresama prekidnih rutina IVTP, pomocni registrar D, izlaz iz ALU Z i 8-bitni registri: programska statusna reč PSW, prihvatni registar podatka memorije MBR, pomocni registri POM1, POM2. Dvobitni registar BR i trobajtni registar IR.

PC je brojački registar koji se naziva programski brojač. Njegov sadržaj se koristi kao adresa memorijske lokacije sa koje treba čitati binarnu reč koja se interpretira kao instrukcija. S obzirom da su binarne reči koje se interpretiraju kao instrukcije smeštene jedna iza druge u memorijskim lokacijama i da stoga treba da se čitaj sekvencijalno, sadržaj programskog brojača PC se, najpre, koristi kao adresa memorijske lokacije sa koje se čita binarn reč, pa se, zatim, njegov sadržaj inkrementira. Registar $PC_{15...0}$ je 16-to razredni programski brojač čiji sadržaj predstavlja adresu memorijske lokacije počev od koje treba pročitati jedan do četiri bajta instrukcije. Sadržaj registra $PC_{15...0}$ se inkrementira generisanjem aktivne vrednosti signala **incPC**. Ovo se koristi prilikom čitanja svakog bajta instrukcije koji se nalaze u susednim 8-mo bitnim lokacijama. Sadržaj registra $PC_{15...0}$ se koristi i za formiranje adrese memorijske lokacije kada se za adresiranje operanda koristi PC relativno adresiranje. U programski brojač se moze vršiti upis kako iz memorije preko registara POM1 i POM2 ( odnosno spoja dve reci registra MBR), tako i iz izlaza aritmeticko-logičke jedinice tj. njenog prihvatnog registra Z i direktno čitanjem instrukcije,odnosno, odredišta u instrukcijam skoka.

MAR je adresni registar memorije. U registar MAR se smešta sadržaj koji predstavlja adresu memorijske lokacije sa koje treba pročitati ili u koju treba upisati binarnu reč. Sadržaj registra MAR se vodi na adresne linije memorije, a u njega se vrsi upis iz registara B, Z, PC, MBR, IR_DA itd.

MBR je prihvatni registar podatka memorije. U registar MBR se smešta sadržaj koji je pročitan iz memorijske lokacije i koji dolazi sa izlaznih linija podataka memorije. U registar MBR se smešta sadržaj koji treba upisati u memorijsku lokaciju. Sadržaj registra MBR se vodi na ulazne linije podataka memorije, a na njegovom ulazu se nalazi multiplekser za propuštanje određenih podataka.

Na ulazne linije aritmeticko logicke jedinice dovodi se izlazi registara: ACC, RS_RC, ITVP, PC, B, IR_DA, BR, DA i na osnovu multipleksera MX i MY se u registre X i Y respektivno se propusta odgovarajuca vrednost.

ACC - Na ulazu u akumulator ACC ima multiplekser na onovu koga se u akumulator propusta izlaz iz aritmeticko logicke jedinice Z, prihvatni registar B i pomocni registri POM1 i POM2.

R0, R1, R2, R3 su pomocni registri opste namene.

B - Na ulazu u prihvatni registrar B takodje postoji multiplekser i na osnovu njega se propustaju vrednosti RS_RC, MBR, IR_DA, POM1 i POM2.

RS_RC je registar koji sluzi da se na osnovu bitova iz IR ako se radi o registarskom adresiranju odredi koji registar se koristi, i na osnovu multipleksera na njegovom ulazu moze da se upise vrednost iz ACC ili B.

PSW je registar programske statusne reci. U njemu se na određenim pozicijama čuvaju biti indikatori statusa. Neki od njih su i biti N, Z, V, C koji sadrže informaciju o prethodno izvršenoj operaciji: da li je dobijena negativna vrednost pri izracunavanju, da li je dobijena nulta vrednost kao rezultat, da li je došlo do prekoračenja i sl. U slučaju skoka na prekidnu rutinu njegova vrednost se čuva na steku jer ta vrednost zajedno sa vrednošcu registra PC pretstavlja trenutni kontekst rada, i pri povratku prethodni kontekst mora biti restauriran. Stoga se na njegov ulaz dovodi vrednost MBR prihvatnog registra podataka iz memorije. Vrednost registra moze bit i rezultat nekog izracunavanja, kao što je ranije navedeno, te se mora formirati u kombinacionoj mreži označenoj kao “Formiranje vrednosti”. Izlaz registra se vodi na ulaz registra MBR, odnosno, na multiplekser MX1.

POM1 i POM2 služe da bi se formirao šesnaestobitni ulaz u određene multipleksere, jer je MBR osmobitan, a adresa je šesnaestobitna pa se na ulaz POM1 dovede viših osam bita, a na POM2 nižih osam bita.

D registar je šesnaestobitni pomeraj. Na njegov ulaz se dovodi osmobitni pomeraj IR2 i preostalih osam viših bita se popunjava bitom IR15 jer je upitanju pomeraj sa znakom. Pomoćni regi-birnim registar D se koristi za prihvatanje 8-bitnog pomeraja u slucaju registarskog indirektnog adresiranja sa 8-bitnim pomerajem. Kako se taj pomeraj treba sabrati sa vrednošću specificiranog registra, operacija sabiranja se obavlja u aritmeticko-logickoj jedinici. Pošto su ulazi ALU jedinice 16-bitne veličine, potrebno je vrednost pomeraja proširiti, vodeći računa o algebarskom znaku pomeraja. Zbog toga je registar D 16-bitni i prvih osam bitova ulaza predstavljaju znak pomeraja, odnosno bit $IR_{15}$ posto je pomeraj određen drugom rečju prihvatnog registra IR.

IVTP registar čuva pokazivač na početak *Interupt Vector* tabele u kojoj se čuvaju adrese prekidnih rutina. Izlaz ovog registra se vodi na ulaz aritmeticko-logicke jedinice u kojoj će se u slučaju prekida, sabiranjem vrednosti ovog registra i broja ulaza, dobiti odgovarajuća adresa prekidne rutine. Na ulaz IVT registra dovodi se vrednost akumulatora.

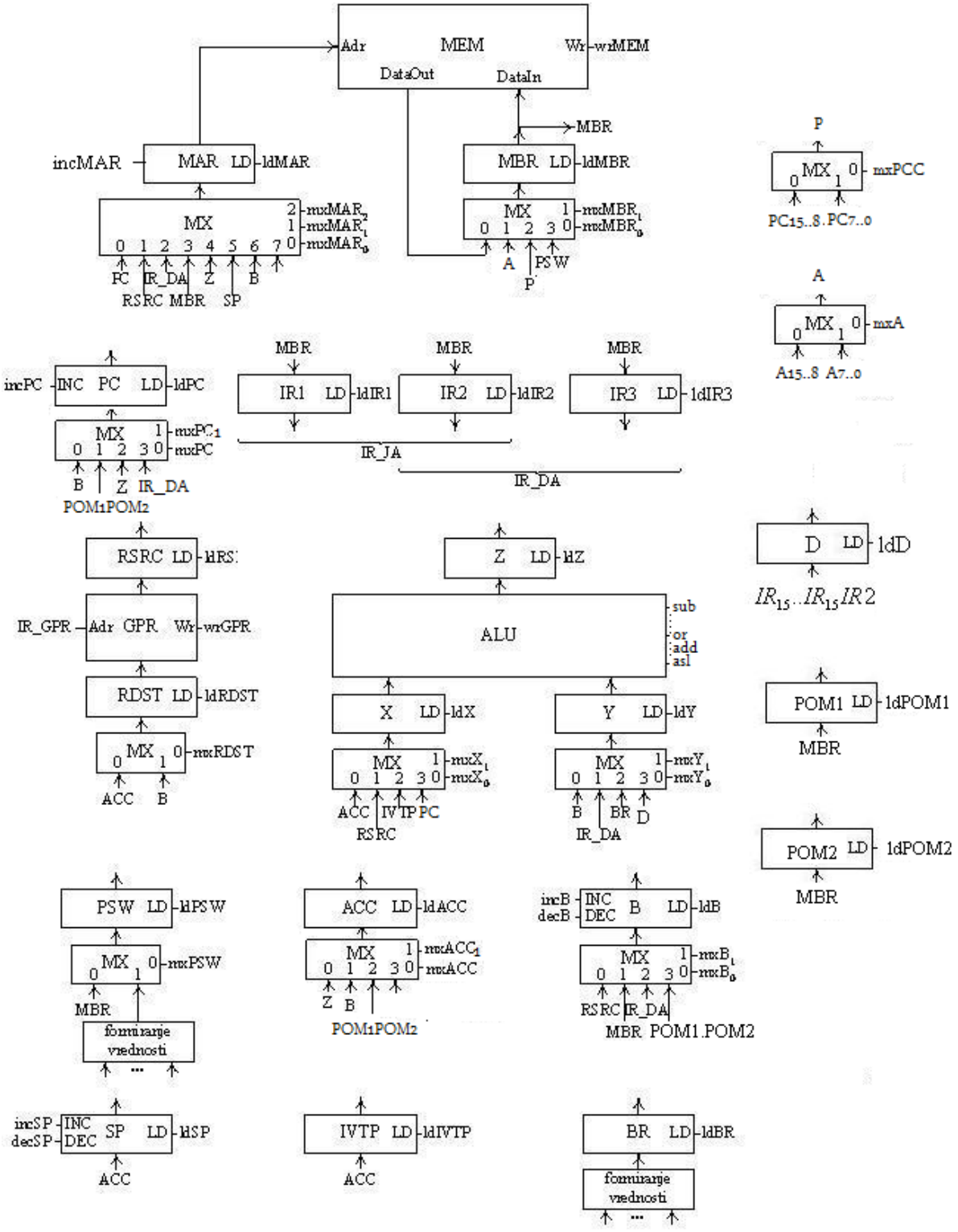
Sa IR je trobajtni prihvatni registar instrukcije. U registar IR se smešta sadržaj registra MBR pročitan sa memorijske lokacije čija je adresa određena sadržajem registra PC, na taj način je iz memorije učitana naredna instrukcija za izvršavanje. Binarnu reč u registru IR treba interpretirati saglasno formatu instrukcije. Registri IR0, IR1, IR2 su 8-mo razredni registri koji formiraju razrede 23…16, 15…8 i 7…0, respektivno, prihvatnog registra instrukcije $IR_{23...0}$. Instrukcije mogu, u zavisnosti od formata instrukcije, da budu dužine 1,2 ili 3 bajta. Razredi $IR_{23...16}$ se uvek čitaju i njihov sadržaj predstavlja kod operacije. Broj preostalih razreda koji se čita zavisi od koda operacije, a u slučaju aritmetičkih i logičkih operacija, i od načina adresiranja i njihov sadržaj ima različito značenje.

IR_JA (Jump Address) predstavlja adresu skoka koja se formira na osnovu registara IR1 i IR2.

IR_DA (Data) Podatak koji se koristi pri izvršavalju određenih instrukcija.

SP registar je pokazivač na poslednju popunjenu memorijsku lokaciju na vrhu steka. Inkrementira se ili dekrementira u zavisnosti da li se stavlja ili skida sa steka. Pošto stek pokazuje na poslednju zauzetu memorijsku lokaciju i raste prema nižim memorijskim lokacijama nakon dekrementiranja podatak se smešta na stek. Pri skidanju sa steka scenario je suprotan, odnosno prvo skidamo podatak sa steka, a potom vrsimo inkrementiranje registra SP. Na ulaz registra se dovodi vrednost čuvana u akumulatoru ACC, a njegov izlaz se naravno dovodi na multiplekser i potom u registar MAR, gde će se sačuvati adresa memorijske lokacije sa koje se čita ili na koju se vrši upis.

BR je registar u koji se smešta broj ulaza u IV tabelu, na osnovu čega se određuje adresa prekidne rutine.




Slika 2 Operaciona jedinica sa direktnim vezama

## 3.2 SEKVENCA UPRAVLJAČKIH SIGNALA PO KORACIMA

Sekvenca upravljačkih signala po koracima se formira na osnovu dijagrama toka izvršavanja instrukcije za operacionu jedinicu sa direktnim vezama. Sekvenca upravljačkih signala po koracima sadrži korake u kojima se generišu upravljački signali operacione jedinice radi realizacije mikrooperacija predstavljenih operacionim blokovima u dijagramu toka i korake u kojima se realizuju grananja predstavljena uslovnim blokovima u dijagramu toka. Koraci u kojima se generišu upravljački signali operacione jedinice nazivaju se operacioni koraci, dok se koraci u kojima se realizuju grananja nazivaju upravljački koraci. U ovom odeljku su date dve sekvence upravljačkih signala po koracima i to sekvenca bez spajanja operacionih i upravljačkih koraka i sekvenca sa spajanjem operacionih i upravljačkih koraka(tabela 2).

Sekvenca upravljačkih signala operacione jedinice bez spajanja operacionih i upravljačkih koraka data je u tabeli 1. U sekvenci 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 !

! U koraku $step_{00}$ radi se učitavanje adrese u MAR sa koje se čita prvi bajt instrukcije i u istom taktu se radi povećavanje brojača PC za 1. U narednom koraku se iz memorije na osnovu registra MAR učitava u registar MBR podatak koji predstavlja prvi bajt instrukcije, zatim se taj podatak upisuje u IR1 (korak $step_{02}$). U naredna dva koraka vrši se provera uslova: l1 i l1'. Ako je l1 aktivan završeno je čitanje instrukcije i ide se na fazu izvršavanja instrukcije. Ukoliko je l1' aktivan završeno je čitanje instrukcije i ide se na fazu formiranje adrese i čitanje operanda. Ukoliko nijedan od ova dva signala nije aktivan čita se još jedan bajt. U naredna dva koraka vrši se provera uslova: l2 i l2'. Ako je l2 aktivan završeno je čitanje instrukcije i ide se na fazu izvršavanja instrukcije. Ukoliko je l2' aktivan završeno je čitanje instrukcije i ide se na fazu formiranje adrese i čitanje operanda. Ukoliko nijedan od ova dva signala nije aktivan čita se još jedan bajt.Zatim se proverava signal logičkog uslova l3 I ukoliko je on aktivan prelazi se na fazu izvršavanja instrukcije, a ukoliko je neaktivan ide se na fazu formiranje adrese i čitanje operanda !

$step_{00}$ **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;
$step_{02}$ **ldIR1**;
$step_{03}$ *br* (*if* **l1** *then* $step_{20}$);
$step_{04}$ *br* (*if* **l1'** *then* $step_{0E}$);
$step_{05}$ **ldMAR**, **incPC**;
$step_{06}$ **ldMBR**;
$step_{07}$ **ldIR2**;
$step_{08}$ *br* (*if* **l2** *then* $step_{20}$);

$step_{09}$ *br* (*if* **I2'** *then* $step_{0E}$);
$step_{0A}$ **ldMAR**, **incPC**;
$step_{0B}$ **ldMBR**;
$step_{0C}$ **ldIR3**;
$step_{0D}$ *br* (*if* **I3** *then* $step_{20}$);

! Formiranje adrese i čitanje operanda !

! U korak $step_{0E}$ se dolazi iz koraka $step_{04}$ , $step_{09}$ ili $step_{0D}$ ukoliko se radi o instrukcijama dužine jedan, dva ili tri bajta koje zahtevaju da se do operanda dođe saglasno specificiranom načinu adresiranja. U slučaju adresiranja kod kojih se operand nalazi u nekom od registara opšte namene ili u samoj instrukciji, ova faza se svodi na prebacivanje operanda u odgovarajuci registar. U slučaju adresiranja kod kojih se operand nalazi u memoriji, ova faza se sastoji od koraka u kojima se prvo formira adresa operanda u memoriji i zatim čita operand. U slučaju nekog od adresiranja kod kojih se operand upisuje u memoriju, u ovoj fazi se samo formira adresa operanda u registru $MAR_{15...0}$, pa se prelazi na fazu izvršavanje operacije u kojoj se operand upisuje u memoriju na formiranoj adresi. U koraku $step_{0E}$ se realizuje višestruki uslovni skok na jedan od koraka $step_{0F}$, $step_{12}$, $step_{14}$ i $step_{1F}$ u zavisnosti od toga koji od signala adresiranja **dirreg**, **dirmem**, **indregpom**, **immed** ima aktivnu vrednost. !

$step_{0E}$ *br* (*case* (**dirreg**, **dirmem**, **indregpom**, **immed**) *then*
(**dirreg**, $step_{0F}$), (**dirmem**, $step_{12}$),), (**indregpom**, $step_{14}$), (**immed**, $step_{1F}$));

! Direktno registarsko !
! U korak $step_{0F}$ se dolazi iz $step_{0E}$ ukoliko je signal za registarsko direktno adresiranje **dirreg** aktivan. Prvo se operand učita u **RSRC** (prihvatni registar), a zatim se upiše u registar B i skače se na fazu izvršavanja instrukcije !
$step_{0F}$ **ldRSRC**;
$step_{10}$ **ldB**;
$step_{11}$ *br* $step_{20}$;

! Direktno memorijsko !
! U korak $step_{12}$ se dolazi iz $step_{0E}$ ukoliko je signal za memorijsko direktno adresiranje **dirmem** aktivan. Signalom $\mathbf{mxMAR_1}$ se sadržaj registra $IRL_{15...0}$ propušta kroz multiplekser i signalom **ldMAR** upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj memorijskog direktnog adresiranja !
$step_{12}$ $\mathbf{mxMAR_1}$, **ldMAR**;
$step_{13}$ *br* $step_{18}$;

! Indirektno registarsko sa pomerajem !
! U korak $step_{14}$ se dolazi iz $step_{0E}$ ukoliko je signal za registarsko indirektno adresiranje sa pomerajem **indregpom** aktivan. Signalima $\mathbf{mxX_0}$, $\mathbf{mxY_0}$ i $\mathbf{mxY_1}$ se najpre kroz multipleksere i na ulaze aritmetičko logucike jedinice propuštaju adresirani registar opšte namene i osmobitni pomeraj iz D registra, zatim se signalom $\mathbf{mxMAR_2}$ sadržaj dobijeni sadrţaja sa izlaza ALU propušta kroz multiplekser i na kraju signalom **ldMAR** upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj registarsko indirektno adresiranje sa pomerajem !
$step_{14}$ **ldRSRC**;
$step_{15}$ $\mathbf{mxX_0}$, **ldX**, $\mathbf{mxY_0}$, $\mathbf{mxY_1}$ ,**ldY**;
$step_{16}$ **add**, **ldZ**;
$step_{17}$ $\mathbf{mxMAR_2}$, **ldMAR**;

! Čitanje operanda za memorijska adresiranja !
! Ako je signal STORE aktivan ide se na korak $step_{20}$ (faze izvršavanja instrukcije), u suprotnom se u registar MBR upisuje prvi bajt koji se prosleđuje pomoćnom registru **POM1** i u istom taktu se inkrementira

MAR registar da bi se pripremio za čitanje sledeceg bajta podatka. Zatim se aktivnim vrednostima signala **mxB$_1$, mxB$_0$** i **ldB** propuštaju podaci sa ulaza multipleksera i upisuju u B !

step$_{18}$ *br* (*if* **STORE** *then* step$_{20}$);
step$_{19}$ **ldMBR**;
step$_{1A}$ **ldPOM1, incMAR**;
step$_{1B}$ **ldMBR**;
step$_{1C}$ **ldPOM2**;
step$_{1D}$ **mxB$_1$, mxB$_0$,ldB**;
step$_{1E}$ *br* step$_{20}$;

! Neposredno !
! U korak step$_{1F}$ se dolazi iz step$_{0E}$ ukoliko je signal za neposredno adresiranje **immed** aktivan. U registar **B** se propusta podatak koji se selektovan na osnovu aktivne vrednosti signala **mxB1** sa ulaza multipleksera !

step$_{1F}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

! U korak step$_{20}$ se dolazi iz koraka step$_{03}$, step$_{08}$, step$_{0D}$, step$_{11}$, step$_{1E}$ i step$_{1F}$ radi izvršavanja operacije. U koraku step$_{20}$ se realizuje višestruki uslovni skok na jedan od koraka step$_{21}$, step$_{23}$, ..., step$_{62}$ u zavisnosti od toga koji od signala operacija **LOAD**, **STORE**,...,**JMPIND** ima aktivnu vrednost !

step$_{20}$ *br* (*case* (**LOAD**, **STORE**, **SUB**, **OR**, **ASL**, **BNZ**, **JMP**, **JSR**, **RTI**, **RTS, PUSH, POP, JMPIND** ) *then*
(**LOAD**, step$_{21}$), (**STORE**, step$_{23}$),
(**SUB**, step$_{2D}$), (**OR**, step$_{31}$), (**ASL**, step$_{35}$),
(**BNZ**, step$_{39}$), (**JSR**, step$_{3E}$), (**JMP**, step$_{46}$), (**RTI**, step$_{48}$), (**RTS**, step$_{4B}$) , (**PUSH**, step$_{53}$) ,
(**POP**, step$_{5A}$), (**JMPIND**, step$_{62}$));

! LOAD !
! U koraku step$_{21}$ se vrši upis vrednosti iz registra B u registar ACC pomocu aktivnih signala **mxACC**, **ldACC** i skače se bezuslovno na korak step$_{66}$ na fazu opsluživanja prekida !

step$_{21}$ **mxACC**, **ldACC**;
step$_{22}$ *br* step$_{66}$;

! STORE !
! U koraku step$_{23}$ se najpre vrši proveravanje o kojem načinu adresiranja se radi. Ako je u pitanju neposredno adresiranje skače se na korak step$_{66}$ jer to predstavlja grešku instrukcije. Zatim se proverava da li je specificirano registarsko direktno adresiranje, ako jeste onda se skače na korak step$_{2A}$ i u $R_i$ se upisuje sadržaj akumulatora. U suprotnom se prelazi na naredni korak i na memorijsku lokaciju na koju ukazuje registar MAR se upisuje sadržaj akulumatora. Najpre niži bajt pa zatim i viši bajt !

step$_{23}$ *br* (*if* **immed** *then* step$_{66}$);
step$_{24}$ *br* (*if* **regdir** *then* step$_{2A}$);
step$_{25}$ **mxMBR$_0$**, **ldMBR**;
step$_{26}$ **wrMEM**;
step$_{27}$ **mxMBR$_0$**, **mxA**, **ldMBR,incMAR**;
step$_{28}$ **wrMEM**;
step$_{29}$ *br* step$_{66}$;
step$_{2A}$ **ldRDST**;
step$_{2B}$ **wrGPR**;
step$_{2C}$ *br* step$_{66}$;

! SUB !

! SUB predstavlja operaciju oduzimanja. Na ulaze aritmetičko logičke jedinice se dovedu vrednosti registara ACC i B i upisu u registre X i Y respektivno. Zatim se izvrši operacija oduzimanja, pa se izvrši prenos podatka iz prihvatnog registra Z u akumulator (ACC). Na kraju se bezuslovno skace na korak step$_{66}$ !

step$_{2D}$ **ldX**, **ldY**;
step$_{2E}$ **sub**, **ldZ**;
step$_{2F}$ **ldACC**;
step$_{30}$ *br* step$_{66}$;

! OR !
! OR predstavlja operaciju logičko ILI. Na ulaze aritmetičko-logičke jedinice se dovedu vrednosti registara ACC i B i upišu u registre X i Y respektivno. Zatim se izvrši operacija logičko ili, pa se izvrši prenos podatka iz prihvatnog registra Z u akumulator (ACC). Na kraju se bezuslovno skače na korak step$_{66}$ !

step$_{31}$ **ldX**, **ldY**;
step$_{32}$ **or**, **ldZ**;
step$_{33}$ **ldACC**;
step$_{34}$ *br* step$_{66}$;

! ASL !
! ASL predstavlja operaciju aritmetičkog pomeranja u levo. Na ulaz aritmetičko-logičke jedinice se dovede vrednost registra B i upiše u registar Y. Zatim se izvrši operacija aritmetičkog pomeranja u levo, pa se izvrši prenos podatka iz prihvatnog registra Z u akumulator (ACC). Na kraju se bezuslovno skače na korak step$_{66}$ !

step$_{35}$ **ldY**;
step$_{36}$ **asl**, **ldZ**;
step$_{37}$ **ldACC**;
step$_{38}$ *br* step$_{66}$;

! BNZ !
! BNZ predstavlja operaciju branch if not zero (skoči ako rezultat nije nula, Z bit u registru PSW). U koraku step$_{39}$ se vrsi provera bita Z i ako je njegova vrednost jedan ide se na korak step$_{67}$, a u suprotnom se u PC upisuje trenutna vrednost PC-a povecana za osmobitni pomeraj koji se nalazi u registru IR2 i zatim se prelazi na korak step$_{66}$ !

step$_{39}$ *br* (*if* **neq** *then* step$_{66}$);
step$_{3A}$ **mxX$_1$, mxX$_2$, ldX, mxY$_1$, mxY$_0$, ldY**;
step$_{3B}$ **add**, **ldZ**;
step$_{3C}$ **mxPC$_1$, ldPC;**
step$_{3D}$ *br* step$_{66}$;

! JSR !
! Instrukcija JSR predstavlja skok u potprogram. Na stek se prvo smešta viših osam bita registra PC, a zatim nižih osam bita. Nakon toga se na isti način smešta na stek i registar PSW. Najpre se dekrementira vrednost registra SP. Podatak se smešta na stek tako što se aktiviranjem odgovarajućih signala multipleksera u registar MAR upisuje vrednost stek pointera, a u registar MBR se smesta podatak. i na identican način se obavlja stavljanje ostalih podataka na stek. Nakon toga se u PC upise adresa pocetka potprograma. Na kraju se ide na korak step$_{66}$ !

step$_{3E}$ **decSP;**
step$_{3F}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
step$_{40}$ **wrMEM;**
step$_{41}$ **decSP;**
step$_{42}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
step$_{43}$ **wrMEM;**
step$_{44}$ **mxPC, mxPC$_1$, ldPC**;
step$_{45}$ *br* step$_{66}$;

! JMP !
! JMP je operacija bezuslovnog skoka. U PC se upiše vrednost koja se nalazi u $IR_{15..0}$ na osnovu aktivnih signala **mxPC$_1$** i **mxPC.** Na kraju se ide na korak step$_{66}$ !

step$_{46}$ **mxPC$_1$** , **mxPC ,ldPC**;
step$_{47}$ *br* step$_{66}$;

! RTI !
! RTI je instrukcija povratka iz prekidne rutine. U ovoj instrukciji se vrši restauracija podataka sa steka. Posto stek ukazuje na poslednju zauzetu memorijsku lokaciju i raste ka nižim lokacijama prvo upisemo u MAR sadržaj registra SP, a zatim i inkrementiramo SP, potom skidamo sa steka bajtove registra PSW u suprotnom poretku kako su stavljani na stek. Vrednost stek pointera se upiše u MAR pomocu kontrolnih signala **mxMAR$_2$**, **mxMAR$_0$** multipleksera koji se nalazi na ulazu u MAR. Pročita se memorijska lokacija na koju ukazuje MAR u MBR, a se ta vrednost upise u PSW. U nastavku se prelazi na instrukciju RTS !

step$_{48}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
step$_{49}$ **ldMBR**;
step$_{4A}$ **ldPSW**;

! RTS !
! RTS predstavlja instrukciju povratka iz potprograma. Vrši se restauracija programskog brojača PC u suprotnom pravcu u odnosu na pravac kojim je stavljan na stek. Proces upisa stek pointera u registar MAR je identičan kao u prvom delu instrukcije RTI. Sada se procita podatak sa steka koji predstavlja nižih osam bita registra PC i upise u registar POM2, a zatim podatak koji predstavlja viših osam bita i upise u registar POM1. Nakon toga se pomoću signala **mxPC** odgovarajuća vrednost upiše u PC. Na kraju se ide na korak step$_{66}$ !

step$_{4B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR ,incSP**;
step$_{4C}$ **ldMBR**;
step$_{4D}$ **ldPOM2**;
step$_{4E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
step$_{4F}$ **ldMBR**;
step$_{50}$ **ldPOM1**;
step$_{51}$ **mxPC, ldPC;**
step$_{52}$ *br* step$_{66}$;

! PUSH !
! Instrukcija PUSH izvršava operaciju stavljanja akumulatora na stek. Registar SP se najpre dekrementira, a zatim se pomocu upravljačkih signala **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, mxMBR$_0$** , **ldMBR** vrednost registra SP upiše u registar MAR, i viši bajt akumulatora upiše u registar MBR , a zatim se ta vrednost upiše u memoriju **wrMEM.** Isti postupak se primeni i za niži bajt akumulatora.Na kraju se ide na korak step$_{66}$ !

step$_{53}$ **decSP**;
step$_{54}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, mxMBR$_0$** , **ldMBR**;
step$_{55}$ **wrMEM**;
step$_{56}$ **decSP**;
step$_{57}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, ldMBR**;
step$_{58}$ **wrMEM**;
step$_{59}$ *br* step$_{66}$;

! POP !
! Instrukcijom POP se skida podatak sa steka i smesta u akumulator (ACC). Posto stek ukazuje na poslednju zauzetu memorijsku lokaciju i raste ka nižim lokacijama prvo upisemo u MAR sadržaj registra SP, a zatim i inkrementiramo SP, potom skidamo sa steka podatak u suprotnom poretku od kojeg je stavljan na stek. Vrednost stek pointera se upiše u MAR pomocu kontrolnih signala **mxMAR$_2$**, **mxMAR$_0$** , **ldMAR** multipleksera koji se nalazi na ulazu u MAR. Pročita se memorijska lokacija na koju ukazuje MAR u

MBR,a zatim u pomoćni registar POM2.Isto se uradi i za viši bajt podatka. Na kraju se vrednost podatka upiše u akumulator upravljačkim signalima **ldACC, mxACC**$_1$. Nakon toga se ide na korak step$_{66}$ !

step$_{5A}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
step$_{5B}$ **ldMBR**;
step$_{5C}$ **ldPOM2**;
step$_{5D}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
step$_{5E}$ **ldMBR**;
step$_{5F}$ **ldPOM1**;
step$_{60}$ **ldACC, mxACC**$_1$;
step$_{61}$ *br* step$_{66}$ ;

! JMPIND !
! JMPIND je instrukcija bezuslovnog indirektnog skoka. U koraku step$_{62}$ se najpre vrši proveravanje o kojem načinu adresiranja se radi. Ako je u pitanju neposredno adresiranje skače se na korak step$_{67}$ jer to predstavlja grešku instrukcije. Zatim se proverava da li je specificirano registarsko direktno adresiranje, ako jeste onda se skače na korak step$_{67}$ jer to predstavlja grešku instrukcije. Ukoliko nije specificirano nijedno od ovih adresiranja, upisuje se nova vrednos u registar PC. Nakon toga se ide na korak step$_{66}$ !

step$_{62}$ *br* (*if* **immed** *then* step$_{66}$);
step$_{63}$ *br* (*if* **regdir** *then* step$_{66}$);
step$_{64}$ **ldPC**;
step$_{65}$ *br* step$_{66}$;

! Opsluživanje prekida !

! U korak step$_{66}$ se dolazi nakon svake izvrsene instrukcije ukoliko je signal **PREKID** aktivan, na završetku faze izvršavanje instrukcije. U koraku step$_{66}$ se, u zavisnosti od toga da li je signal **prekid** bloka *intr* neaktivan ili aktivan, ili završava izvršavanje tekuće instrukcije i prelaskom na korak step$_{00}$ započinje faza čitanje instrukcije sledeće instrukcije ili se produžava izvršavanje tekuće instrukcije i prelaskom na korak step$_{67}$ produžava faza opsluživanje prekida tekuće instrukcije !

step$_{66}$ *br* (*if* $\overline{\text{PREKID}}$ *then* step$_{00}$);

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

! Čuvanje konteksta procesora !

! Kontekst procesora i to PC$_{15...0}$ i PSW$_{7...0}$ se čuva u koracima step$_{67}$ do step$_{6C}$. U koracima step$_{67}$ do step$_{6C}$ se na stek stavlja programski brojač PC$_{15...0}$. Na stek se stavlja prvo viši a zatim i niži bajt registra PC$_{15...0}$. Stoga se najpre u koraku step$_{67}$ signalom **decSP** vrši dekrementiranje registra SP$_{15...0}$,a zatim signalima **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR**, sadržaj registra SP$_{15...0}$ propušta kroz multiplekser MX i upisuje u registar MAR$_{15...0}$ ,a signalima **mxMBR$_2$** i **ldMBR** sadržaj višeg bajta registra PC$_{15...8}$ propušta kroz multiplekser MX i upisuje u registar MDR$_{7...0}$. Upis se realizuje u koracima step$_{69}$ i step$_{6C}$. Potom se u koraku step$_{6A}$ signalom **decSP** vrši dekrementiranje registra SP$_{15...0}$, a zatim signalima **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR**, sadržaj registra SP$_{15...0}$ propušta kroz multiplekser MX i upisuje u registar MAR$_{15...0}$ i signalima **mxMBR2**, **mxMBR0**, **ldMBR** sadržaj nižeg bajta registra PC$_{7...0}$ propušta kroz multiplekser MX i upisuje u registar MDR$_{7...0}$ .U koracima step$_{6D}$ do step$_{6F}$ se na stek stavlja programska statusna reč PSW. Stoga se najpre u koraku step$_{6D}$ signalom **decSP** vrši dekrementiranje registra SP$_{15...0}$. i signalima **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR**, sadržaj registra SP$_{15...0}$ propušta kroz multiplekser MX i upisuje u registar MAR$_{15...0}$ i signalima **mxMDR$_2$** i **mxMDR$_0$**, **ldMDR** sadržaj registra PSW propušta kroz multiplekser MX i upisuje u registar MBR$_{7...0}$. Zatim sledi upis u memoriju !

step$_{67}$ **decSP**;
step$_{68}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
step$_{69}$ **wrMEM**;
step$_{6A}$ **decSP**;
step$_{6B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
step$_{6C}$ **wrMEM**;
step$_{6D}$ **decSP**;
step$_{6E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
step$_{6F}$ **wrMEM**;

! Utvrđivanje broja ulaza i utvrđivanje adrese prekidne rutine !

! U korak step$_{75}$ se dolazi iz step$_{74}$. U koracima step$_{70}$ do step$_{7A}$ se utvrđuje broj ulaza u tabelu sa adresama prekidnih rutina i upisuje u registar BR$_{7...0}$. U ovim koracima se po opadajućim prioritetima utvrđuje zahtev za prekid najvišeg prioriteta i za njega određuje broj ulaza u tabelu sa adresama prekidnih rutina !

step$_{70}$ **mxB$_2$, ldB**;
step$_{71}$ **slB**;
step$_{72}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
step$_{73}$ **add**, **ldZ**;
step$_{74}$ **mxMAR$_2$**, **ldMAR**;
step$_{75}$ **ldMBR**;
step$_{76}$ **ldPOM$_2$, incMAR**;
step$_{77}$ **ldMBR**;
step$_{78}$ **ldPOM$_1$**;
step$_{79}$ **mxPC**, **ldPC**;
step$_{7A}$ *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$_{10}$ **ldB** i step$_{11}$ *br* step$_{31}$, zatim koraci step$_{12}$ **mxMAR$_0$**, **ldMAR** i step$_{13}$ *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$_{20}$), zatim koraci step$_{07}$ **ldIR2** i step$_{08}$ *br* (*if* **l2** *then* step$_{20}$) 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$_{20}$ ili na korak step$_{04}$. Zbog toga koraci step$_{02}$ i step$_{03}$ ne mogu da se spoje. Takođe, u koraku step$_{07}$ 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$_{08}$ vrši provera signala **l2** i u zavisnosi od njegove vrednosti prelazi ili na korak step$_{09}$ ili na korak

$step_{20}$. Zbog toga koraci $step_{07}$ i $step_{08}$ ne mogu da se spoje. Treba uočiti da se koraci $step_{17}$ **mxMAR$_2$**, **ldMAR** i $step_{18}$ *br* (*if* **STORE** *then* $step_{20}$) ne mogu spojiti iako je signal logičkog uslova **STORE**, koji se konsultuje u koraku $step_{18}$, 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 $step_{17}$, jer se na korak $step_{18}$ prelazi ne samo iz koraka $step_{17}$ već i iz koraka $step_{13}$ *br* $step_{18}$.

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 $step_{0D}$ *br* (*if* **l3** *then* $step_{20}$) i $step_{0E}$ *br* (*case* (**dirreg**, **indreg**, ..., **immed**) ...) i koraci $step_{1F}$ **mxB$_1$**, **ldB** i $step_{20}$ *br* (*case* (**LOAD**, **STORE**, ..., **RTS,**…) ...). Koraci $step_{0D}$ *br* (*if* **l3** *then* $step_{20}$) i $step_{0E}$ *br* (*case* (**dirreg**, **indreg**, ..., **immed**) ...) se ne mogu spojiti iako su signali logičkih uslova **dirreg**, **indreg**, ..., **immed**, koji se konsultuju u koraku $step_{0E}$, formirani još u koraku $step_{07}$ 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 $step_{0D}$, jer se na korak $step_{0E}$ prelazi ne samo iz koraka $step_{0D}$ već i iz koraka $step_{04}$ i $step_{07}$. Takođe se koraci **mxB$_1$**, **ldB** i $step_{20}$ *br* (*case* (**LOAD**, **STORE**, ..., **RTS,**…) ...) se ne mogu spojiti iako su signali logičkih uslova **LOAD**, **STORE**, ..., **RTS**, koji se konsultuju u koraku $step_{20}$, 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 $step_{1F}$, jer se na korak $step_{20}$ prelazi ne samo iz koraka $step_{1F}$ već i iz koraka $step_{03}$, $step_{08}$, 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 $step_{28}$ **wrMEM** i $step_{29}$ *br* $step_{66}$ itd. ukoliko bi upis u neku memorijsku lokaciju, iniciran generisanjem signala **wrMEM** u koraku $step_{28}$, 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 $step_{28}$ **wrMEM** i $step_{29}$ *br* $step_{66}$ itd. mogu spajati.

Kada se, saglasno prethodnim razmatranjima, izvrši spajanje operacionih i upravljačkih koraka iz tabele 1, dobija se sekvenca upravljačkih signala po koracima data u tabeli 2. U sekvenci se koriste iskazi za signale, iskazi za skokove i kombinacija iskaza za signale i nekog od iskaza za skokove. Iskazi za signale i skokove su istog oblika u sekvenci sa spajanjem koraka kao i u sekvenci bez spajanja koraka. 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, dok koraci u kojima se pojavljuju samo iskazi za signale ili iskazi za skokove odgovaraju situacijama kada to nije bilo moguće.

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

! Nije moguće spajanje koraka jer logički uslovi se formiraju od reči koja je prethodno učitana !

! Čitanje instrukcije !

$step_{00}$ **ldMAR**, **incPC**;
$step_{01}$ **ldMBR**;
$step_{02}$ **ldIR1**;
$step_{03}$ *br* (*if* **l1** *then* $step_{1D}$);
$step_{04}$ *br* (*if* **l1'** *then* $step_{0E}$);
$step_{05}$ **ldMAR**, **incPC**;

$step_{06}$ **ldMBR**;
$step_{07}$ **ldIR2**;
$step_{08}$ *br* (*if* **l2** *then* $step_{1D}$);
$step_{09}$ *br* (*if* **l2'** *then* $step_{0E}$);
$step_{0A}$ **ldMAR**, **incPC**;
$step_{0B}$ **ldMBR**;
$step_{0C}$ **ldIR3**;
$step_{0D}$ *br* (*if* **l3** *then* $step_{1D}$);

! Formiranje adrese i čitanje operanda !

$step_{0E}$ *br* (*case* (**dirreg**, **dirmem**, **indregpom**, **immed**) *then*
(**dirreg**, $step_{0F}$), (**dirmem**, $step_{11}$),), (**indregpom**, $step_{12}$), (**immed**, $step_{1C}$));

! Za sva adresiranje moguće je spajanje koraka jer se bezuslovni skok realizuje bez obzira na instrukciju koja je prethodno izvršena !

! Direktno registarsko !
$step_{0F}$ **ldRSRC**;
$step_{10}$ **ldB**,*br* $step_{1D}$;
! Direktno memorijsko !
$step_{11}$ **mxMAR$_1$**, **ldMAR**, *br* $step_{16}$;
! Indirektno registarsko sa pomerajem !
$step_{12}$ **ldRSRC**;
$step_{13}$ **mxX$_0$**, **ldX**, **mxY$_0$**, **mxY$_1$** ,**ldY**;
$step_{14}$ **add**, **ldZ**;
$step_{15}$ **mxMAR$_2$**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
$step_{16}$ *br* (*if* **STORE** *then* $step_{1D}$);
$step_{17}$ **ldMBR**;
$step_{18}$ **ldPOM1, incMAR**;
$step_{19}$ **ldMBR**;
$step_{1A}$ **ldPOM2**;
$step_{1B}$ **mxB$_1$, mxB$_0$,ldB**, *br* $step_{1D}$;
! Neposredno !
$step_{1C}$ **mxB$_1$**, **ldB**;

! Za svaku instrukciju moguće je spajanje koraka jer se bezuslovni skok realizuje bez obzira na poslednju instrukcju !

! Izvršavanje operacije !

$step_{1D}$ *br* (*case* (**LOAD**, **STORE**, **SUB**, **OR**, **ASL**, **BNZ**, **JMP**, **JSR**, **RTI**, **RTS, PUSH, POP, JMPIND** ) *then*
(**LOAD**, $step_{1E}$), (**STORE**, $step_{1F}$),
(**SUB**, $step_{27}$), (**OR**, $step_{2A}$), (**ASL**, $step_{2D}$),
(**BNZ**, $step_{30}$), (**JSR**, $step_{34}$), (**JMP**, $step_{3B}$), (**RTI**, $step_{3C}$), (**RTS**, $step_{3F}$) , (**PUSH**, $step_{46}$) ,
(**POP**, $step_{4C}$), (**JMPIND**, $step_{53}$));

! LOAD !
$step_{1E}$ **mxACC**, **ldACC**, *br* $step_{56}$;
! STORE !
$step_{1F}$ *br* (*if* **immed** *then* $step_{56}$);

step$_{20}$ *br* (*if* **regdir** *then* step$_{25}$);
step$_{21}$ **mxMBR$_0$, ldMBR**;
step$_{22}$ **wrMEM**;
step$_{23}$ **mxMBR$_0$, mxA, ldMBR,incMAR**;
step$_{24}$ **wrMEM**, *br* step$_{56}$;
step$_{25}$ **ldRDST**;
step$_{26}$ **wrGPR**, *br* step$_{56}$;

! SUB !

step$_{27}$ **ldX, ldY**;
step$_{28}$ **sub, ldZ**;
step$_{29}$ **ldACC**, *br* step$_{56}$;

! OR !

step$_{2A}$ **ldX, ldY**;
step$_{2B}$ **or, ldZ**;
step$_{2C}$ **ldACC**, *br* step$_{56}$;

! ASL !

step$_{2D}$ **ldY**;
step$_{2E}$ **asl, ldZ**;
step$_{2F}$ **ldACC**, *br* step$_{56}$;

! BNZ !

step$_{30}$ *br* (*if* **neq** *then* step$_{56}$);
step$_{31}$ **mxX$_1$, mxX$_2$, ldX, mxY$_1$, mxY$_0$, ldY**;
step$_{32}$ **add, ldZ**;
step$_{33}$ **mxPC$_1$, ldPC,** *br* step$_{56}$;

! JSR !

step$_{34}$ **decSP;**
step$_{35}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
step$_{36}$ **wrMEM;**
step$_{37}$ **decSP;**
step$_{38}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
step$_{39}$ **wrMEM;**
step$_{3A}$ **mxPC, mxPC$_1$, ldPC,** *br* step$_{56}$;

! JMP !

step$_{3B}$ **mxPC$_1$ , mxPC ,ldPC**, *br* step$_{56}$;

! RTI !

step$_{3C}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, incSP**;
step$_{3D}$ **ldMBR**;
step$_{3E}$ **ldPSW**;

! RTS !

step$_{3F}$ **mxMAR$_2$, mxMAR$_0$, ldMAR ,incSP**;
step$_{40}$ **ldMBR**;
step$_{41}$ **ldPOM2**;
step$_{42}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, incSP**;
step$_{43}$ **ldMBR**;
step$_{44}$ **ldPOM1**;
step$_{45}$ **mxPC, ldPC,** *br* step$_{56}$;

! PUSH !

step$_{46}$ **decSP**;
step$_{47}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_0$ , ldMBR**;
step$_{48}$ **wrMEM**;
step$_{49}$ **decSP**;
step$_{4A}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, ldMBR**;

step$_{4B}$ **wrMEM**, *br* step$_{56}$;

! POP !

step$_{4C}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
step$_{4D}$ **ldMBR**;
step$_{4E}$ **ldPOM2**;
step$_{4F}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
step$_{50}$ **ldMBR**;
step$_{51}$ **ldPOM1**;
step$_{52}$ **ldACC, mxACC$_1$**, *br* step$_{56}$ ;

! JMPIND !

step$_{53}$ *br* (*if* **immed** *then* step$_{56}$);
step$_{54}$ *br* (*if* **regdir** *then* step$_{56}$);
step$_{55}$ **ldPC**, *br* step$_{56}$;

! Nije moguće spajanje koraka jer skok zavisi od vrednosti PC-a !

! Opsluživanje prekida !

step$_{56}$ *br* (*if* $\overline{\text{PREKID}}$ *then* step$_{00}$);
step$_{57}$ **decSP**;
step$_{58}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
step$_{59}$ **wrMEM**;
step$_{5A}$ **decSP**;
step$_{5B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
step$_{5C}$ **wrMEM**;
step$_{5D}$ **decSP**;
step$_{5E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
step$_{5F}$ **wrMEM**;
step$_{60}$ **mxB$_2$, ldB**;
step$_{61}$ **slB**;
step$_{62}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
step$_{63}$ **add**, **ldZ**;
step$_{64}$ **mxMAR$_2$**, **ldMAR**;
step$_{65}$ **ldMBR**;
step$_{66}$ **ldPOM$_2$, decMAR**;
step$_{67}$ **ldMBR**;
step$_{68}$ **ldPOM$_1$**;
step$_{69}$ **mxPC**, **ldPC**;
step$_{6A}$ *br* step$_{00}$;

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.

# 4 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 ovoj glavi je data realizacija upravljačke jedinice 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.

## 4.1 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.

### 4.1.1 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 $\mathbf{val_A}$ 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 $\mathbf{val_A}$ i vrednosti A za sve korake ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima, dati su u tabeli 3.

Tabela 3 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 |
| $step_{18}$ | $\mathbf{val_{18}}$ | 18 |
| $step_{20}$ | $\mathbf{val_{20}}$ | 20 |
| $step_{66}$ | $\mathbf{val_{66}}$ | 66 |

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 $\mathbf{val_A}$ 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 4. Koraci $step_A$, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A za sve korake ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima dati su u tabeli 5.

Tabela 4 Signali uslovnih skokova i signali uslova

| signal uslovnog skoka | signal uslova |
|---|---|
| **brl1** | **l1** |
| **brl1'** | **l1'** |
| **brl2** | **l2** |
| **brl2'** | **l2'** |
| **brl3** | **l3** |
| **brSTORE** | **STORE** |
| **brimmed** | **immed** |
| **brregdir** | **regdir** |
| **brneq** | **neq** |
| **brnotPREKID** | $\overline{\text{PREKID}}$ |

Tabela 5 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 |
| $step_{0E}$ | $\mathbf{val_{0E}}$ | 0E |
| $step_{20}$ | $\mathbf{val_{20}}$ | 20 |

| | | |
|---|---|---|
| $step_{29}$ | **$val_{29}$** | 29 |
| $step_{66}$ | **$val_{66}$** | 66 |

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 6. 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 7 i 8.

Tabela 6 Signali višestrukih uslovnih skokova

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

Tabela 7 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{0F}$

| signal uslova | vrednost |
|---|---|
| **dirreg** | 0F |
| **dirmem** | 12 |
| **indregpom** | 14 |
| **immed** | 1F |

Tabela 8 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{31}$

| signal uslova | vrednost | signal uslova | vrednost |
|---|---|---|---|
| **LOAD** | 21 | **BNZ** | 39 |
| **STORE** | 23 | **JMP** | 46 |
| **SUB** | 2D | **JSR** | 3E |
| **OR** | 31 | **RTI** | 48 |
| **ASL** | 35 | **RTS** | 4B |
| **PUSH** | 53 | **POP** | 5A |
| **JMPIND** | 62 | | |

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela tabela 1), formirana sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 9). 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 4) i višestrukih uslovnih skokova (tabela 6) i signali **$val_A$** za bezuslovne (tabela 3) i uslovne (tabela 5) 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 9 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**, **val$_{20}$**;
$T_{04}$ **brl1'**, **val$_{0E}$**;
$T_{05}$ **ldMAR**, **incPC**;
$T_{06}$ **ldMBR**;
$T_{07}$ **ldIR2**;
$T_{08}$ **brl2**, **val$_{20}$**;
$T_{09}$ **brl2'**, **val$_{0E}$**;
$T_{0A}$ **ldMAR**, **incPC**;
$T_{0B}$ **ldMBR**;
$T_{0C}$ **ldIR3**;
$T_{0D}$ **brl3**, **val$_{20}$**;

! Formiranje adrese i čitanje operanda !

$T_{0E}$ **bradr**;

! Direktno registarsko !

$T_{0F}$ **ldRSRC**;
$T_{10}$ **ldB**;
$T_{11}$ **bruncnd**, **val$_{20}$**;

! Direktno memorijsko !

$T_{12}$ **mxMAR$_1$**, **ldMAR**;
$T_{13}$ **bruncnd**, **val$_{18}$**;

! Indirektno registarsko sa pomerajem !

$T_{14}$ **ldRSRC**;
$T_{15}$ **mxX$_0$**, **ldX**, **mxY$_0$**, **mxY$_1$** ,**ldY**;
$T_{16}$ **add**, **ldZ**;
$T_{17}$ **mxMAR$_2$**, **ldMAR**;

! Čitanje operanda za memorijska adresiranja !

$T_{18}$ **brSTORE**, **val$_{20}$;**
$T_{19}$ **ldMBR**;
$T_{1A}$ **ldPOM1, incMAR**;
$T_{1B}$ **ldMBR**;
$T_{1C}$ **ldPOM2**;
$T_{1D}$ **mxB$_1$, mxB$_0$,ldB**;
$T_{1E}$ **bruncnd**, **val$_{20}$**;

! Neposredno !

$T_{1F}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

$T_{20}$ **bropr**;

! LOAD !

$T_{21}$ **mxACC**, **ldACC**;
$T_{22}$ **bruncnd**, **val$_{66}$**;

! STORE !

$T_{23}$ **brimmed**, **val$_{66}$**;
$T_{24}$ **brregdir**, **val$_{29}$**;
$T_{25}$ **mxMBR$_0$**, **ldMBR**;

$T_{26}$ **wrMEM**;
$T_{27}$ **mxMBR$_0$**, **mxA**, **ldMBR,incMAR**;
$T_{28}$ **wrMEM**;
$T_{29}$ **bruncnd**, **val$_{66}$**;
$T_{2A}$ **ldRDST**;
$T_{2B}$ **wrGPR**;
$T_{2C}$ **bruncnd**, **val$_{66}$**;

! SUB !

$T_{2D}$ **ldX**, **ldY**;
$T_{2E}$ **sub**, **ldZ**;
$T_{2F}$ **ldACC**;
$T_{30}$ **bruncnd**, **val$_{66}$**;

! OR !

$T_{31}$ **ldX**, **ldY**;
$T_{32}$ **or**, **ldZ**;
$T_{33}$ **ldACC**;
$T_{34}$ **bruncnd**, **val$_{66}$**;

! ASL !

$T_{35}$ **ldY**;
$T_{36}$ **asl**, **ldZ**;
$T_{37}$ **ldACC**;
$T_{38}$ **bruncnd**, **val$_{66}$**;

! BNZ !

$T_{39}$ **brneq**, **val$_{66}$**;
$T_{3A}$ **mxX$_1$, mxX$_2$, ldX, mxY$_1$, mxY$_0$, ldY**;
$T_{3B}$ **add**, **ldZ**;
$T_{3C}$ **mxPC$_1$, ldPC;**
$T_{3D}$ **bruncnd**, **val$_{66}$**;

! JSR !

$T_{3E}$ **decSP;**
$T_{3F}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
$T_{40}$ **wrMEM;**
$T_{41}$ **decSP;**
$T_{42}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
$T_{43}$ **wrMEM;**
$T_{44}$ **mxPC, mxPC$_1$, ldPC**;
$T_{45}$ **bruncnd**, **val$_{66}$**;

! JMP !

$T_{46}$ **mxPC$_1$** , **mxPC ,ldPC**;
$T_{47}$ **bruncnd**, **val$_{66}$**;

! RTI !

$T_{48}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
$T_{49}$ **ldMBR**;
$T_{4A}$ **ldPSW**;

! RTS !

$T_{4B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR ,incSP**;
$T_{4C}$ **ldMBR**;
$T_{4D}$ **ldPOM2**;
$T_{4E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
$T_{4F}$ **ldMBR**;
$T_{50}$ **ldPOM1**;
$T_{51}$ **mxPC, ldPC;**

$T_{52}$ **bruncnd**, **val$_{66}$**;

! PUSH !

$T_{53}$ **decSP**;
$T_{54}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, mxMBR$_0$**, **ldMBR**;
$T_{55}$ **wrMEM**;
$T_{56}$ **decSP**;
$T_{57}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, ldMBR**;
$T_{58}$ **wrMEM**;
$T_{59}$ **bruncnd**, **val$_{66}$**;

! POP !

$T_{5A}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
$T_{5B}$ **ldMBR**;
$T_{5C}$ **ldPOM2**;
$T_{5D}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
$T_{5E}$ **ldMBR**;
$T_{5F}$ **ldPOM1**;
$T_{60}$ **ldACC, mxACC$_1$**;
$T_{61}$ **bruncnd**, **val$_{66}$**;

! JMPIND !

$T_{62}$ **brimmed , val$_{66}$**;
$T_{63}$ **brregdir, val$_{66}$**;
$T_{64}$ **ldPC**;
$T_{65}$ **bruncnd**, **val$_{66}$**;

! Opsluživanje prekida !

$T_{66}$ **brnotPREKID, val$_{00}$**;
$T_{67}$ **decSP**;
$T_{68}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
$T_{69}$ **wrMEM**;
$T_{6A}$ **decSP**;
$T_{6B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
$T_{6C}$ **wrMEM**;
$T_{6D}$ **decSP**;
$T_{6E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
$T_{6F}$ **wrMEM**;
$T_{70}$ **mxB$_2$, ldB**;
$T_{71}$ **slB**;
$T_{72}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
$T_{73}$ **add**, **ldZ**;
$T_{74}$ **mxMAR$_2$**, **ldMAR**;
$T_{75}$ **ldMBR**;
$T_{76}$ **ldPOM$_2$, incMAR**;
$T_{77}$ **ldMBR**;
$T_{78}$ **ldPOM$_1$**;
$T_{79}$ **mxPC**, **ldPC**;
$T_{7A}$ **bruncnd**, **val$_{00}$**;

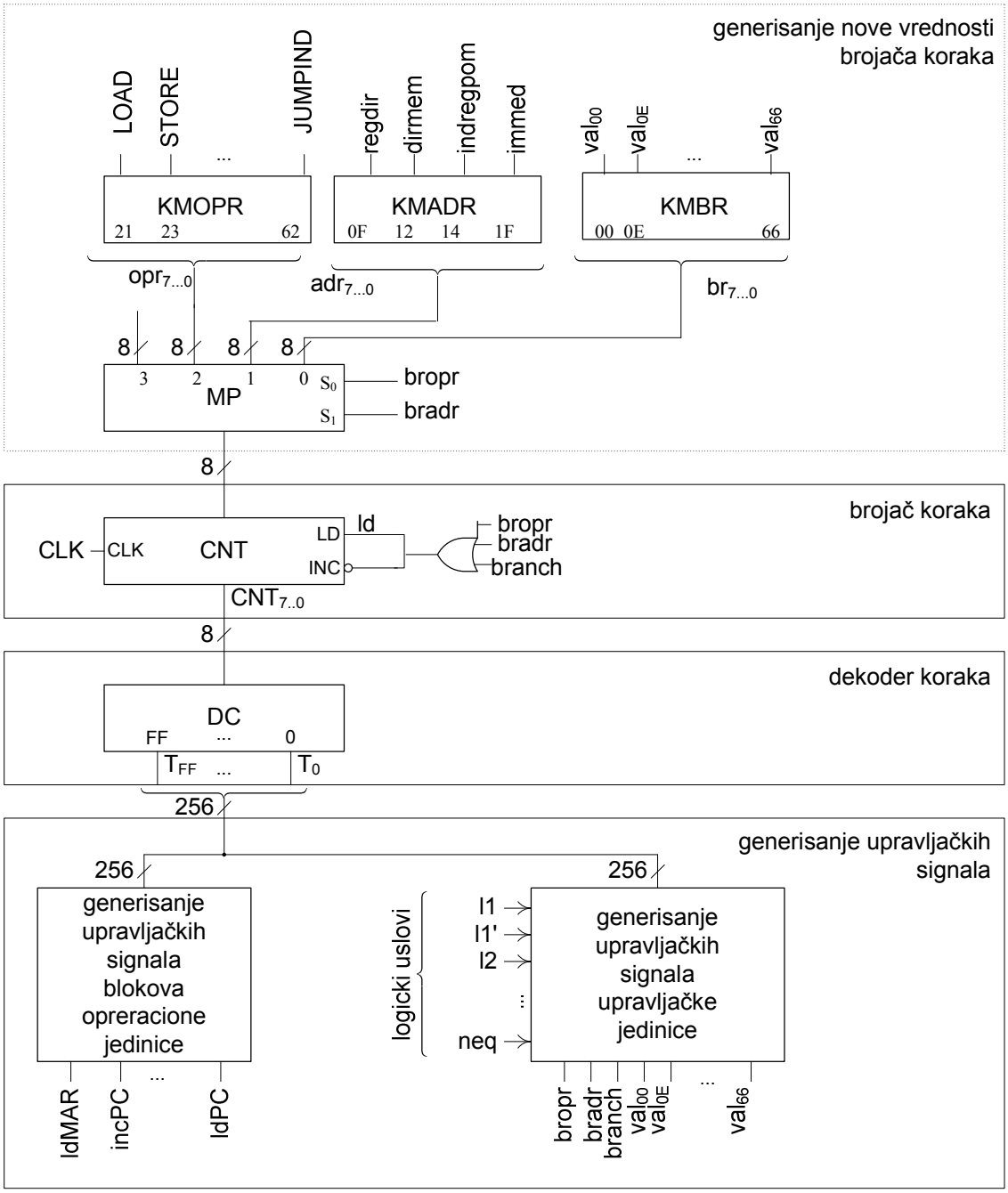
Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 3. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti brojača koraka*, blok *brojač koraka*, blok *dekoder koraka* i blok *generisanje upravljačkih signala*.

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**.

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 8) za realizaciju višestrukog uslovnog skoka u koraku $step_{20}$ 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 8 da se pojavi tada na linijama $\mathbf{opr_{7...0}}$. S obzirom da stanje brojača koraka $T_{20}$ 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 7) za realizaciju višestrukog uslovnog skoka u koraku $step_{0E}$ sekvence upravljačkih signala po koracima. U zavisnosti od toga koji od signala **dirreg**, **dirmem**, **indregpom**, **immed** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 7 da se pojavi tada na linijama $\mathbf{adr_{7...0}}$. S obzirom da stanje brojača koraka $T_{0E}$ 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 3) i uslovne skokove (tabela 5) u sekvenci upravljačkih signala po koracima. U zavisnosti od toga koji od signala $\mathbf{val_{00}}$, $\mathbf{val_{0E}}$, ..., $\mathbf{val_{66}}$ ima aktivnu vrednost zavisi koja će od vrednosti iz tabela 3 i 5 tada da se pojavi na linijama $\mathbf{br_{7...0}}$. Signali višestrukih uslovnih skokova **bradr** i **bropr** su aktivni samo u stanjima $T_{0E}$ i $T_{20}$ 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 $CNT_{7...0}$.




Slika 3 Struktura upravljačke jedinice

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 9). 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 9). 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 9). S obzirom da se upravljački signali svih faza izvršavanja instrukcija realizuju u opsegu od koraka $T_{00}$ do koraka $T_{7A}$ 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 9).

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, l1'**, **l2**, ..., **PREKID** koji dolaze iz operacione jedinice i saglasno sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 9) 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:

- $\mathbf{ldMAR = T_{00} + T_{05} + T_{12} + T_{17} + T_{3F} + T_{42} + T_{48} + T_{4B} + T_{4E} + T_{54} + T_{57} + T_{5A} + T_{5D} + T_{68} + T_{6B} + T_{6E} + T_{74}}$
- $\mathbf{mxMAR_0 = T_{3F} + T_{42} + T_{48} + T_{4B} + T_{4E} + T_{54} + T_{57} + T_{5A} + T_{5D} + T_{68} + T_{6B} + T_{6E}}$
- $\mathbf{mxMAR_1 = T_{12}}$
- $\mathbf{mxMAR_2 = T_{17} + T_{3F} + T_{42} + T_{48} + T_{4B} + T_{4E} + T_{54} + T_{57} + T_{5A} + T_{5D} + T_{68} + T_{6B} + T_{6E} + T_{74}}$
- $\mathbf{wrGPR = T_{2B}}$

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

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- $\mathbf{bropr = T_{20}}$
- $\mathbf{bradr = T_{0E}}$
- $\mathbf{branch = bruncnd + brl1 \cdot l1 + brl1' \cdot l1' + brl2 \cdot l2 + brl2' \cdot l2' + brl3 \cdot l3 + brSTORE \cdot STORE + brimmed \cdot immed + brregdir \cdot regdir + brneq \cdot neq + brnotPREKID \cdot \overline{PREKID}}$
- $\mathbf{val_{00} = T_{66} + T_{7A}}$
- $\mathbf{val_{0E} = T_{04} + T_{09}}$
- $\mathbf{val_{18} = T_{13}}$
- $\mathbf{val_{20} = T_{04} + T_{08} + T_{0D} + T_{11} + T_{18} + T_{1E}}$
- $\mathbf{val_{29} = T_{24}}$
- $\mathbf{val_{66} = T_{22} + T_{23} + T_{29} + T_{2C} + T_{30} + T_{34} + T_{38} + T_{39} + T_{3D} + T_{45} + T_{47} + T_{52} + T_{59} + T_{61} + T_{62} + T_{63} + T_{65}}$

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

- **bruncnd** = $\mathbf{T_{11}} + \mathbf{T_{13}} + \mathbf{T_{1E}} + \mathbf{T_{22}} + \mathbf{T_{29}} + \mathbf{T_{2C}} + \mathbf{T_{30}} + \mathbf{T_{34}} + \mathbf{T_{38}} + \mathbf{T_{3D}} + \mathbf{T_{45}} + \mathbf{T_{47}} + \mathbf{T_{52}} + \mathbf{T_{59}} + \mathbf{T_{61}} + \mathbf{T_{65}} + \mathbf{T_{7A}}$
- **brl1** = $\mathbf{T_{03}}$
- **brl1'** = $\mathbf{T_{04}}$
- **brl2** = $\mathbf{T_{08}}$
- **brl2'** = $\mathbf{T_{09}}$
- **brl3** = $\mathbf{T_{0D}}$
- **brSTORE** = $\mathbf{T_{18}}$
- **brimmed** = $\mathbf{T_{23}} + \mathbf{T_{62}}$
- **brregdir** = $\mathbf{T_{24}} + \mathbf{T_{63}}$
- **brneq** = $\mathbf{T_{39}}$
- **brnotPREKID** = $\mathbf{T_{66}}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l1'**, **l2**, **l2'**, **l3**, **STORE**, **immed**, **regdir**, **neq** i **PREKID**.

### 4.1.2 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* $\text{step}_A$, *br* (*if* **uslov** *then* $\text{step}_A$) i *br* (*case* ($\mathbf{uslov_1}$, ..., $\mathbf{uslov_n}$) *then* ($\mathbf{uslov_1}$, $\text{step}_{A1}$), ..., ($\mathbf{uslov_n}$, $\text{step}_{An}$).

Umesto iskaza *br* $\text{step}_A$ treba staviti signal bezuslovnog skoka i signal $\mathbf{val_A}$. Simbolička oznaka signala bezuslovnog skoka je **bruncnd**. Koraci $\text{step}_A$ na koje treba bezuslovno preći, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A koje treba upisati u brojač koraka, dati su u tabeli 10.

Tabela 10 Koraci $\text{step}_A$, signali $\mathbf{val_A}$ i vrednosti A za bezuslovne skokove

| $\text{step}_A$ | $\mathbf{val_A}$ | A |
|---|---|---|
| $\text{step}_{00}$ | $\mathbf{val_{00}}$ | 00 |
| $\text{step}_{16}$ | $\mathbf{val_{16}}$ | 16 |
| $\text{step}_{1D}$ | $\mathbf{val_{1D}}$ | 1D |
| $\text{step}_{56}$ | $\mathbf{val_{56}}$ | 56 |

Umesto iskaza *br* (*if* **uslov** *then* $\text{step}_A$) 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 11. Koraci $\text{step}_A$ 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 12.

Tabela 11 Signali uslovnih skokova i signali uslova

| signal uslovnog skoka | signal uslova |
|---|---|
| **brl1** | **l1** |

| | |
|---|---|
| **brl1'** | **l1'** |
| **brl2** | **l2** |
| **brl2'** | **l2'** |
| **brl3** | **l3** |
| **brSTORE** | **STORE** |
| **brimmed** | **immed** |
| **brregdir** | **regdir** |
| **brneq** | **neq** |
| **brnotPREKID** | $\overline{\text{PREKID}}$ |

Tabela 12 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 |
| $step_{0E}$ | $\mathbf{val_{0E}}$ | 0E |
| $step_{1D}$ | $\mathbf{val_{1D}}$ | 1D |
| $step_{25}$ | $\mathbf{val_{25}}$ | 25 |
| $step_{56}$ | $\mathbf{val_{56}}$ | 56 |

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 13. 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 14 i 15.

Tabela 13 Signali višestrukih uslovnih skokova

| korak | signal višestrukog uslovnog skoka |
|---|---|
| $step_{0E}$ | **bradr** |
| $step_{1D}$ | **bropr** |

Tabela 14 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{0F}$

| signal uslova | vrednost |
|---|---|
| **dirreg** | 0F |
| **dirmem** | 11 |
| **indregpom** | 12 |
| **immed** | 1C |

Tabela 15 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{2A}$

| signal uslova | vrednost | signal uslova | vrednost |
|---|---|---|---|
| **LOAD** | 1E | **BNZ** | 30 |
| **STORE** | 1F | **JMP** | 34 |
| **SUB** | 27 | **JSR** | 3B |
| **OR** | 2A | **RTI** | 3C |
| **ASL** | 2D | **RTS** | 3F |
| **PUSH** | 46 | **POP** | 4C |
| **JMPIND** | 53 | | |

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 16). Ona ima istu formu kao i tabela 9 za upravljačku jedinicu bez spajanja koraka.

Tabela 16 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, val$_{1D}$**;
$T_{04}$ **brl1', val$_{0E}$**;
$T_{05}$ **ldMAR**, **incPC**;
$T_{06}$ **ldMBR**;
$T_{07}$ **ldIR2**;
$T_{08}$ **brl2, val$_{1D}$**;
$T_{09}$ **brl2', val$_{0E}$**;
$T_{0A}$ **ldMAR**, **incPC**;
$T_{0B}$ **ldMBR**;
$T_{0C}$ **ldIR3**;
$T_{0D}$ **brl3, val$_{1D}$**;

! Formiranje adrese i čitanje operanda !

$T_{0E}$ **bradr;**

! Direktno registarsko !

$T_{0F}$ **ldRSRC**;
$T_{10}$ **ldB**, **bruncnd**, **val$_{1D}$**;

! Direktno memorijsko !

$T_{11}$ **mxMAR$_1$**, **ldMAR**, **bruncnd**, **val$_{16}$**;

! Indirektno registarsko sa pomerajem !

$T_{12}$ **ldRSRC**;
$T_{13}$ **mxX$_0$**, **ldX**, **mxY$_0$**, **mxY$_1$** ,**ldY**;
$T_{14}$ **add**, **ldZ**;
$T_{15}$ **mxMAR$_2$**, **ldMAR**;

! Čitanje operanda za memorijska adresiranja !

$T_{16}$ **brSTORE**, **val$_{1D}$**;
$T_{17}$ **ldMBR**;
$T_{18}$ **ldPOM1, incMAR**;
$T_{19}$ **ldMBR**;
$T_{1A}$ **ldPOM2**;
$T_{1B}$ **mxB$_1$, mxB$_0$,ldB**, **bruncnd**, **val$_{1D}$**;

! Neposredno !

$T_{1C}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

$T_{1D}$ **bropr**;

! LOAD !

$T_{1E}$ **mxACC**, **ldACC**, **bruncnd**, **val$_{56}$**;

! STORE !

$T_{1F}$ **brimmed**, **val$_{56}$**;
$T_{20}$ **brregdir**, **val$_{25}$**;
$T_{21}$ **mxMBR$_0$**, **ldMBR**;
$T_{22}$ **wrMEM**;

$T_{23}$ **mxMBR$_0$**, **mxA**, **ldMBR,incMAR**;
$T_{24}$ **wrMEM**, **bruncnd**, **val$_{56}$**;
$T_{25}$ **ldRDST**;
$T_{26}$ **wrGPR**, **bruncnd**, **val$_{56}$**;

! SUB !

$T_{27}$ **ldX**, **ldY**;
$T_{28}$ **sub**, **ldZ**;
$T_{29}$ **ldACC**, **bruncnd**, **val$_{56}$**;

! OR !

$T_{2A}$ **ldX**, **ldY**;
$T_{2B}$ **or**, **ldZ**;
$T_{2C}$ **ldACC**, **bruncnd**, **val$_{56}$**;

! ASL !

$T_{2D}$ **ldY**;
$T_{2E}$ **asl**, **ldZ**;
$T_{2F}$ **ldACC**, **bruncnd**, **val$_{56}$**;

! BNZ !

$T_{30}$ **brneq**, **val$_{56}$**;
$T_{31}$ **mxX$_1$, mxX$_2$, ldX, mxY$_1$, mxY$_0$, ldY**;
$T_{32}$ **add**, **ldZ**;
$T_{33}$ **mxPC$_1$, ldPC, bruncnd**, **val$_{56}$**;

! JSR !

$T_{34}$ **decSP;**
$T_{35}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
$T_{36}$ **wrMEM;**
$T_{37}$ **decSP;**
$T_{38}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
$T_{39}$ **wrMEM;**
$T_{3A}$ **mxPC, mxPC$_1$, ldPC, bruncnd**, **val$_{56}$**;

! JMP !

$T_{3B}$ **mxPC$_1$** , **mxPC ,ldPC**, **bruncnd**, **val$_{56}$**;

! RTI !

$T_{3C}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
$T_{3D}$ **ldMBR**;
$T_{3E}$ **ldPSW**;

! RTS !

$T_{3F}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR ,incSP**;
$T_{40}$ **ldMBR**;
$T_{41}$ **ldPOM2**;
$T_{42}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
$T_{43}$ **ldMBR**;
$T_{44}$ **ldPOM1**;
$T_{45}$ **mxPC, ldPC, bruncnd**, **val$_{56}$**;

! PUSH !

$T_{46}$ **decSP**;
$T_{47}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, mxMBR$_0$** , **ldMBR**;
$T_{48}$ **wrMEM**;
$T_{49}$ **decSP**;
$T_{4A}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, ldMBR**;
$T_{4B}$ **wrMEM**, **bruncnd**, **val$_{56}$**;

! POP !

$T_{4C}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;

$T_{4D}$ **ldMBR**;
$T_{4E}$ **ldPOM2**;
$T_{4F}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
$T_{50}$ **ldMBR**;
$T_{51}$ **ldPOM1**;
$T_{52}$ **ldACC, mxACC$_1$**, **bruncnd**, **val$_{56}$**;

! JMPIND !

$T_{53}$ **brimmed**, **val$_{56}$**;
$T_{54}$ **brregdir**, **val$_{56}$**;
$T_{55}$ **ldPC**, **bruncnd**, **val$_{56}$**;

! Opsluživanje prekida !

$T_{56}$ **brnotPREKID**, **val$_{00}$**;
$T_{57}$ **decSP**;
$T_{58}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
$T_{59}$ **wrMEM**;
$T_{5A}$ **decSP**;
$T_{5B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
$T_{5C}$ **wrMEM**;
$T_{5D}$ **decSP**;
$T_{5E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
$T_{5F}$ **wrMEM**;
$T_{60}$ **mxB$_2$, ldB**;
$T_{61}$ **slB**;
$T_{62}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
$T_{63}$ **add**, **ldZ**;
$T_{64}$ **mxMAR$_2$**, **ldMAR**;
$T_{65}$ **ldMBR**;
$T_{66}$ **ldPOM$_2$, decMAR**;
$T_{67}$ **ldMBR**;
$T_{68}$ **ldPOM$_1$**;
$T_{69}$ **mxPC**, **ldPC**;
$T_{6A}$ **bruncnd**, **val$_{00}$**;

Struktura upravljačke jedinice sa spajanjem koraka je ista kao i za slučaj bez spajanja koraka (slika 3). 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 $T_{6A}$. 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 1E, 1F, ..., 53 pri aktivnim vrednostima signala **LOAD**, **STORE**,…, **JMPIND** respektivno. Kombinaciona mreža KMADR generiše vrednosti 0F, 11, 12, 1C pri aktivnim vrednostima signala **dirreg**, **dirmem**, **indregpom**, **immed** respektivno. Kombinaciona mreža KMBR generiše vrednosti 00, 0E, ..., 56 pri aktivnim vrednostima signala **val$_{00}$**, **val$_{0E}$**, ..., **val$_{56}$**, respektivno. Zbog drugih vrednosti koje generiše kombinaciona mreža KMBR javljaju se drugi signali **val$_A$**. Tako se umesto signala **val$_{20}$**, javlja signal **val$_{1D}$**, jer u brojač koraka umesto vrednosti 20 treba upisati 1D 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** = $T_{00} + T_{05} + T_{0A} + T_{11} + T_{15} + T_{35} + T_{38} + T_{3C} + T_{3F} + T_{42} + T_{47} + T_{4A} + T_{4C} + T_{4F} + T_{58} + T_{5B} + T_{5E}$
- **mxMAR**$_0$ = $T_{35} + T_{38} + T_{3C} + T_{3F} + T_{42} + T_{47} + T_{4A} + T_{4C} + T_{4F} + T_{58} + T_{5B} + T_{5E}$
- **mxMAR**$_1$ = $T_{11}$
- **mxMAR**$_2$ = $T_{15} + T_{35} + T_{38} + T_{3C} + T_{3F} + T_{42} + T_{47} + T_{4A} + T_{4C} + T_{4F} + T_{58} + T_{5B} + T_{5E} + T_{64}$
- **wrGPR** = $T_{26}$

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

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr** = $T_{1D}$
- **bradr** = $T_{0E}$
- **branch** = **bruncnd** + **brl1**·**l1** + **brl1'**·**l1'** + **brl2**·**l2** + **brl2'**·**l2'** + **brl3**·**l3** + **brSTORE**·**STORE** + **brimmed**·**immed** + **brregdir**·**regdir** + **brneq**·**neq**+ **brnotPREKID**·$\overline{\text{PREKID}}$
- **val**$_{00}$ = $T_{56} + T_{6A}$
- **val**$_{0E}$ = $T_{04} + T_{09}$
- **val**$_{18}$ = $T_{11}$
- **val**$_{1D}$ = $T_{03} + T_{08} + T_{0D} + T_{10} + T_{16} + T_{1B}$
- **val**$_{25}$ = $T_{20}$
- **val**$_{56}$ = $T_{1E} + T_{1F} + T_{24} + T_{26} + T_{29} + T_{2C} + T_{2F} + T_{30} + T_{33} + T_{3A} + T_{3B} + T_{45} + T_{4B} + T_{52} + T_{53} + T_{54} + T_{55}$

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

- **bruncnd** = $T_{10} + T_{11} + T_{1B} + T_{1E} + T_{24} + T_{26} + T_{29} + T_{2C} + T_{2F} + T_{33} + T_{3A} + T_{3B} + T_{45} + T_{4B} + T_{52} + T_{55} + T_{6A}$
- **brl1** = $T_{03}$
- **brl1'** = $T_{04}$
- **brl2** = $T_{08}$
- **brl2'** = $T_{09}$
- **brl3** = $T_{0D}$
- **brSTORE** = $T_{16}$
- **brimmed** = $T_{1F} + T_{53}$
- **brregdir** = $T_{20} + T_{54}$
- **brneq** = $T_{30}$
- **brnotPREKID** = $T_{56}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l1'**, **l2**, **l2'**, **l3**, **STORE**, **immed**, **regdir**, **neq** i **PREKID**.

## 4.2 MIKROPROGRAMSKA REALIZACIJA

U ovom poglavlju se razmatraju mikroprogramska upravljačka jedinica sa dva tipa mikroinstrukcija i mikroprogramska upravljačka jedinica sa jednim tipom mikroinstrukcija.

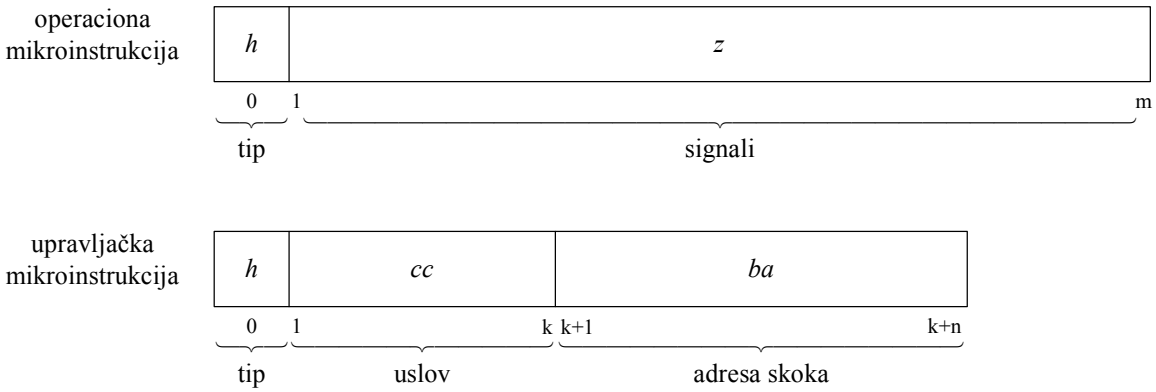
### 4.2.1 Mikroprogramska realizacija sa dva tipa mikroinstrukcija

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 4 i svakom upravljačkom koraku, u kome se realizuju skokovi, pridružuje binarna reč čiji je format dat na slici 4. 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 4 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 4) i to operaciona mikroinstrukcija i upravljačka mikroinstrukcija.

Format operacione mikroinstrukcije je dat na slici 5. 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 |
|---|---|---|---|---|---|---|---|
| $mxMBR_0$ | incMAR | ldMAR | mxPCC | ldPC | incPC | mxPC | $mxPC_1$ |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| $ldR_1$ | $ldR_2$ | $ldR_3$ | mxA | ldRSRC | wrGPR | ldZ | ldD |

| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|
| mxRDST | ldRDST | ldX | ldY | $ldPOM_1$ | $ldPOM_2$ | $mxX_1$ | $mxX_0$ |

| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
|---|---|---|---|---|---|---|---|
| $mxY_1$ | $mxY_0$ | ldPSW | mxPSW | incSP | decSP | ldSP | ldACC |

| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|---|---|---|---|---|---|---|---|
| $mxACC_1$ | mxACC | ldIVTP | ldB | incB | decB | slB | $mxB_2$ |

| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
|---|---|---|---|---|---|---|---|
| $mxB_1$ | $mxB_0$ | ldBR | / | / | / | / | / |

Slika 5 Operaciona mikroinstrukcija

Format upravljačke mikroinstrukcije je dat na slici 6. Polje *h* je 1.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | *cc* | | | | *ba* | | |

| 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 |
|---|---|---|---|---|---|---|---|
| / | / | / | / | / | / | / | / |

| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|---|---|---|---|---|---|---|---|
| / | / | / | / | / | / | / | / |

| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
|---|---|---|---|---|---|---|---|
| / | / | / | / | / | / | / | / |

Slika 6 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 17.

Tabela 17 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 18.

Tabela 18 Signali uslovnih skokova

| *cc* | signal uslovnog skoka | signal uslova |
|---|---|---|
| 02 | **brl1** | **l1** |
| 03 | **brl1'** | **l1'** |
| 04 | **brl2** | **l2** |
| 05 | **brl2'** | **l2'** |
| 06 | **brl3** | **l3** |
| 07 | **brSTORE** | **STORE** |
| 08 | **brimmed** | **immed** |
| 09 | **brregdir** | **regdir** |
| 10 | **brneq** | **neq** |
| 11 | **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 19.

Tabela 19 Signali višestrukih uslovnih skokova

| *cc* | korak | signal višestrukog uslovnog skoka |
|---|---|---|
| 12 | $step_{0E}$ | **bradr** |
| 13 | $step_{1D}$ | **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 17. 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 20.

Tabela 20 Koraci $step_A$, adrese $madr_A$ i vrednosti A za bezuslovne skokove

| $step_A$ | $madr_A$ | A |
|---|---|---|
| $step_{00}$ | $madr_{00}$ | 00 |
| $step_{0E}$ | $madr_{0E}$ | 0E |
| $step_{20}$ | $madr_{20}$ | 20 |
| $step_{66}$ | $madr_{66}$ | 66 |

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 18. 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 21.

Tabela 21 Koraci $step_A$, adrese $madr_A$ i vrednosti A za uslovne skokove

| $step_A$ | $madr_A$ | A |
|---|---|---|
| $step_{00}$ | $madr_{00}$ | 00 |
| $step_{0E}$ | $madr_{0E}$ | 0E |
| $step_{20}$ | $madr_{20}$ | 20 |
| $step_{2A}$ | $madr_{2A}$ | 2A |

| | | |
|---|---|---|
| $step_{66}$ | $madr_{66}$ | 66 |

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 19. 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 22 i 23.

Tabela 22 Signali uslova i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{0F}$

| signal uslova | vrednost |
|---|---|
| **dirreg** | 0F |
| **dirmem** | 12 |
| **indregpom** | 14 |
| **immed** | 1F |

Tabela 23 Signali uslova i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{31}$

| signal uslova | vrednost | signal uslova | vrednost |
|---|---|---|---|
| **LOAD** | 21 | **JSR** | 3E |
| **STORE** | 23 | **RTI** | 48 |
| **SUB** | 2D | **RTS** | 4B |
| **OR** | 31 | **PUSH** | 53 |
| **ASL** | 35 | **POP** | 5B |
| **BNZ** | 39 | **JUMPIND** | 62 |
| **JMP** | 46 | | |

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 1) formiran mikroprogram (tabela 24). 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 24 Mikroprogram

! Čitanje instrukcije !

$madr_{00}$ **ldMAR**, **incPC**;
$madr_{01}$ **ldMBR**;
$madr_{02}$ **ldIR1**;

madr$_{03}$ **cnt**, **brl1**, **madr$_{20}$**;
madr$_{04}$ **cnt**, **brl1'**, **madr$_{0E}$**;
madr$_{05}$ **ldMAR, incPC**;
madr$_{06}$ **ldMBR**;
madr$_{07}$ **ldIR2**;
madr$_{08}$ **cnt,brl2,madr$_{20}$**;
madr$_{09}$ **cnt,brl2',madr$_{0E}$**;
madr$_{0A}$ **ldMAR, incPC**;
madr$_{0B}$ **ldMBR**;
madr$_{0C}$ **ldIR3**;
madr$_{0D}$ **cnt,brl3,madr$_{20}$**;

! Formiranje adrese i čitanje operanda !

madr$_{0E}$ **cnt, bradr;**

! Direktno registarsko !

madr$_{0F}$ **ldRSRC**;
madr$_{10}$ **ldB**;
madr$_{11}$ **cnt**, **bruncnd**, **madr$_{20}$**;

! Direktno memorijsko !

madr$_{12}$ **mxMAR$_1$**, **ldMAR**;
madr$_{13}$ **cnt**, **bruncnd**, **madr$_{18}$**;

! Indirektno registarsko sa pomerajem !

madr$_{14}$ **ldRSRC**;
madr$_{15}$ **mxX$_0$**, **ldX**, **mxY$_0$**, **ldY**;
madr$_{16}$ **add**, **ldZ**;
madr$_{17}$ **mxMAR$_2$**, **ldMAR**;
! Čitanje operanda za memorijska adresiranja !
madr$_{18}$ **brSTORE**, **madr$_{20}$**;
madr$_{19}$ **ldMBR**;
madr$_{1A}$ **ldPOM1, incMAR;**
madr$_{1B}$ **ldMBR**;
madr$_{1C}$ **ldPOM2**;
madr$_{1D}$ **mxB$_1$, mxB$_0$,ldB**;
madr$_{1E}$ **cnt**, **bruncnd**, **madr$_{20}$**;

! Neposredno !

madr$_{1F}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

madr$_{20}$ **cnt**, **bropr**;

! LOAD !

madr$_{21}$ **mxACC**, **ldACC**;
madr$_{22}$ **cnt**, **bruncnd**, **madr$_{66}$**;

! STORE !

madr$_{23}$ **cnt**, **brimmed**, **madr$_{66}$**;
madr$_{24}$ **cnt**, **brregdir**, **madr$_{2A}$**;
madr$_{25}$ **mxMBR$_0$**, **ldMBR**;
madr$_{26}$ **wrMEM**;
madr$_{27}$ **mxMBR$_0$**, **mxA**, **ldMBR,incMAR**;
madr$_{28}$ **wrMEM**;
madr$_{29}$ **cnt**, **bruncnd**, **madr$_{66}$**;
madr$_{2A}$ **ldRDST**;
madr$_{2B}$ **wrGPR**;
madr$_{2C}$ **cnt**, **bruncnd**, **madr$_{66}$**;

! SUB !

madr$_{2D}$ **ldX**, **ldY**;

madr$_{2E}$ **sub**, **ldZ**;

madr$_{2F}$ **ldACC**;

madr$_{30}$ **cnt**, **bruncnd**, **madr$_{66}$**;

! OR !

madr$_{31}$ **ldX**, **ldY**;

madr$_{32}$ **or**, **ldZ**;

madr$_{33}$ **ldACC**;

madr$_{34}$ **cnt**, **bruncnd**, **madr$_{66}$**;

! ASR !

madr$_{35}$ **ldY**;

madr$_{36}$ **asl**, **ldZ**;

madr$_{37}$ **ldACC**;

madr$_{38}$ **cnt**, **bruncnd**, **madr$_{66}$**;

! BNZ !

madr$_{39}$ **brZ**, **madr$_{56}$**;

madr$_{3A}$ **mxX$_1$, mxX$_2$, ldX, mxY$_1$, mxY$_0$, ldY**;

madr$_{3B}$ **add**, **ldZ**;

madr$_{3C}$ **mxPC$_1$, ldPC;**

madr$_{3D}$ **cnt, bruncnd, madr$_{66}$**

! JSR !

madr$_{3E}$ **decSP**;

madr$_{3F}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**

madr$_{40}$ **wrMEM;**

madr$_{41}$ **decSP;**

madr$_{42}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**

madr$_{43}$ **wrMEM;**

madr$_{44}$ **mxPC, mxPC$_1$, ldPC**;

madr$_{45}$ **cnt, bruncnd, madr$_{66}$**;

! JMP !

madr$_{46}$ **mxPC$_1$** , **mxPC ,ldPC**;

madr$_{47}$ **cnt**, **bruncnd**, **madr$_{66}$**;

! RTI !

madr$_{48}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;

madr$_{49}$ **ldMBR**;

madr$_{4A}$ **ldPSW**;

! RTS !

madr$_{4B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR ,incSP**;

madr$_{4C}$ **ldMBR**;

madr$_{4D}$ **ldPOM2**;

madr$_{4E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;

madr$_{4F}$ **ldMBR**;

madr$_{50}$ **ldPOM1**;

madr$_{51}$ **mxPC, ldPC;**

madr$_{52}$ **cnt**, **bruncnd**, **madr$_{66}$**;

! PUSH !

madr$_{53}$ **decSP**;

madr$_{54}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, mxMBR$_0$** , **ldMBR**;

madr$_{55}$ **wrMEM**;

madr$_{56}$ **decSP**;

madr$_{57}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, ldMBR**;
madr$_{58}$ **wrMEM**;
madr$_{59}$ **cnt**, **bruncnd**, **madr$_{66}$**;

! POP !

madr$_{5A}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
madr$_{5B}$ **ldMBR**;
madr$_{5C}$ **ldPOM2**;
madr$_{5D}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
madr$_{5E}$ **ldMBR**;
madr$_{5F}$ **ldPOM1**;
madr$_{60}$ **ldACC, mxACC$_1$**;
madr$_{61}$ **cnt**, **bruncnd**, **madr$_{66}$**;

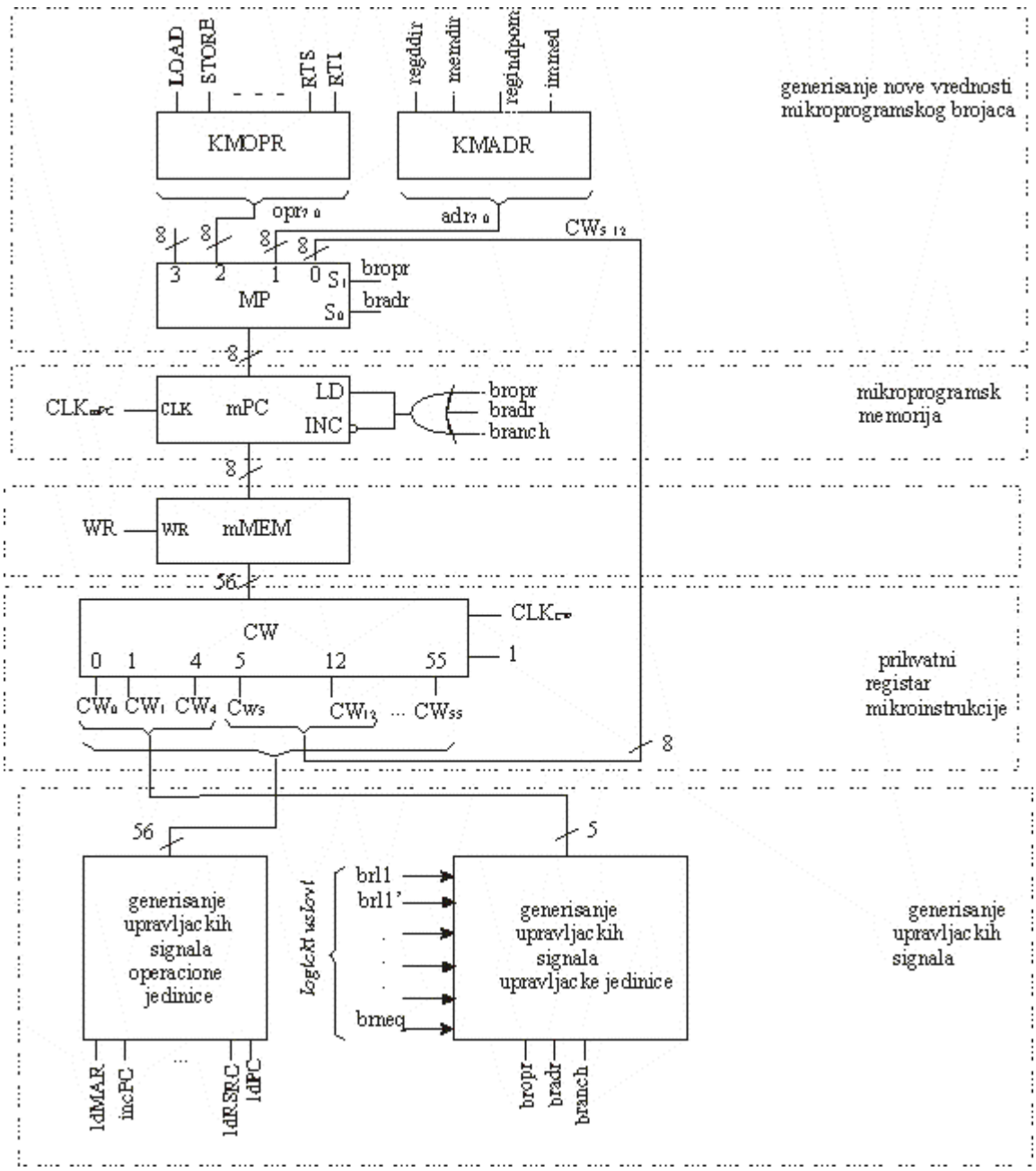
! JMPIND !

madr$_{62}$ **cnt**, **brimmed**, **madr$_{66}$**;
madr$_{63}$ **cnt**, **brregdir**, **madr$_{66}$**;
madr$_{64}$ **ldPC**;
madr$_{65}$ *br* step$_{66}$;

! Opsluživanje prekida !

madr$_{66}$ **cnt**, **brnotPREKID**, **madr$_{00}$**;
madr$_{67}$ **decSP**;
madr$_{68}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
madr$_{69}$ **wrMEM**;
madr$_{6A}$ **decSP**;
madr$_{6B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
madr$_{6C}$ **wrMEM**;
madr$_{6D}$ **decSP**;
madr$_{6E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
madr$_{6F}$ **wrMEM**;
madr$_{71}$ **slB**;
madr$_{72}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
madr$_{73}$ **add**, **ldZ**;
madr$_{74}$ **mxMAR$_2$**, **ldMAR**;
madr$_{75}$ **ldMBR**;
madr$_{76}$ **ldPOM$_2$, incMAR**;
madr$_{77}$ **ldMBR**;
madr$_{78}$ **ldPOM$_1$**;
madr$_{79}$ **mxPC**, **ldPC**;
madr$_{7A}$ **cnt**, **bruncnd**, **madr$_{00}$**;

Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 7. 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*.




Slika 7 Struktura upravljačke jedinice mikroprogramske realizacije sa horizontalnim formatom mikroinstrukcije

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 $\mathbf{opr_{7...0}}$, kombinacione mreže KMADR koja formira signale $\mathbf{adr_{7...0}}$ 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 $\mathbf{opr_{7...0}}$ ako je aktivan signal **bropr**, signali $\mathbf{adr_{7...0}}$ ako je aktivan signal **bradr** i signali $\mathbf{CW_{k+1...k+n}}$ ako su neaktivni signali **bropr** i **bradr**.

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 23) za realizaciju višestrukog uslovnog skoka na adresi 31 mikroprograma (tabela 24). U zavisnosti od toga koji od signala **LOAD**, **STORE**, …, **RTS** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 23 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 22) za realizaciju višestrukog uslovnog skoka na adresi 0F mikroprograma (tabela 24). U zavisnosti od toga koji od signala **dirreg**, **indreg**,..., **immed** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 22 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 20) i uslovne skokove (tabela 21). 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 24). 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 24). 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 24). S obzirom da se mikroprogram svih faza izvršavanja instrukcija nalazi u opsegu od adrese 00 do adrese 7A, 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 51 bita, a kapacitet veličinom mikroprograma svih instrukcija

procesora (tabela 24) 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 **brl1**,**brl1'**,**brl2**,**brl2'**,**brl3**,**brSTORE**,**brimmed**,**brregdir**,**brneq**,**brnotPREKID** koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice se generišu na sledeći način:

- $\mathbf{ldMAR} = \overline{\mathbf{CW}_0} \cdot \mathbf{CW_1}$
- $\mathbf{mxMAR_0} = \overline{\mathbf{CW}_0} \cdot \mathbf{CW_4}$
- Na identičan način se generišu i preostali upravljački signali operacione jedinice.

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- $\mathbf{bropr} = \mathbf{CW_0} \cdot \mathbf{CW_4} \cdot \overline{\mathbf{CW}_5} \cdot \overline{\mathbf{CW}_6} \cdot \mathbf{CW_7}$
- $\mathbf{bradr} = \mathbf{CW_0} \cdot \mathbf{CW_4} \cdot \overline{\mathbf{CW}_5} \cdot \overline{\mathbf{CW}_6} \cdot \overline{\mathbf{CW}_7}$
- **branch** = **bruncnd**+ **brl1***l1 + **brl1'***l1' + **brl2***l2 + **brl2'***l2' + **brl3***l3 + **brSTORE***STORE + **brimmed***immed + **brregdir***regdir + **brneq***eql+ **brnotPREKID*** $\overline{\text{PREKID}}$

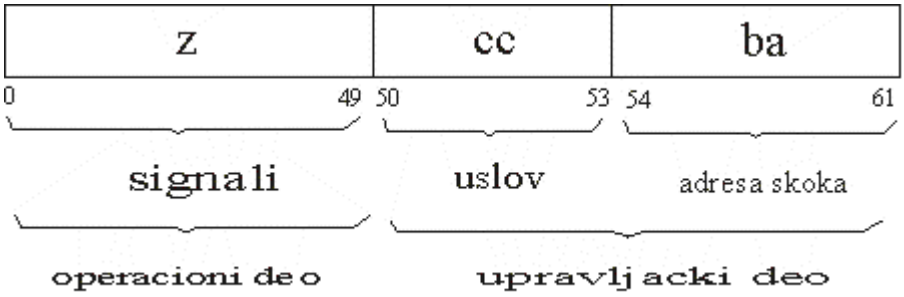
Signali koji se javljaju u izrazu za signal **branch** se generišu na sledeći način:

- $\mathbf{bruncnd} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \overline{\mathbf{CW}_5} \cdot \overline{\mathbf{CW}_6} \cdot \mathbf{CW_7}$
- $\mathbf{brl1} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \overline{\mathbf{CW}_5} \cdot \mathbf{CW_6} \cdot \overline{\mathbf{CW}_7}$
- $\mathbf{brl1'} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \overline{\mathbf{CW}_5} \cdot \mathbf{CW_6} \cdot \mathbf{CW_7}$
- $\mathbf{brl2} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \mathbf{CW_5} \cdot \overline{\mathbf{CW}_6} \cdot \overline{\mathbf{CW}_7}$
- $\mathbf{brl2'} = \mathbf{CW_0} \cdot \mathbf{CW_4} \cdot \mathbf{CW_5} \cdot \overline{\mathbf{CW}_6} \cdot \mathbf{CW_7}$
- $\mathbf{brl3} = \mathbf{CW_0} \cdot \mathbf{CW_4} \cdot \mathbf{CW_5} \cdot \overline{\mathbf{CW}_6} \cdot \overline{\mathbf{CW}_7}$
- $\mathbf{brneq} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \mathbf{CW_5} \cdot \mathbf{CW_6} \cdot \overline{\mathbf{CW}_7}$
- $\mathbf{brSTORE} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \mathbf{CW_5} \cdot \overline{\mathbf{CW}_6} \cdot \overline{\mathbf{CW}_7}$
- $\mathbf{brregdir} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \overline{\mathbf{CW}_5} \cdot \overline{\mathbf{CW}_6} \cdot \mathbf{CW_7}$
- $\mathbf{brimmed} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \mathbf{CW_5} \cdot \overline{\mathbf{CW}_6} \cdot \mathbf{CW_7}$
- $\mathbf{brnotPREKID} = \mathbf{CW_0} \cdot \overline{\mathbf{CW}_4} \cdot \mathbf{CW_5} \cdot \mathbf{CW_6} \cdot \mathbf{CW_7}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l1'**, **l2, l2', l3**, **STORE**, **immed**, **regdir**, **eql** i $\overline{\text{PREKID}}$.

### 4.2.2 Mikroprogramska realizacija sa jednim tipom mikroinstrukcija

U slučaju spajanja koraka postoji samo jedan tip mikroinstrukcije (slika 8).




Slika 8 Format mikroinstrukcije za horizontalni način kodiranja upravljačkih signala

Kodiranje operacionog i upravljačkog dela mikroinstrukcije je dato na slici 9.

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

| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|
| $mxMBR_0$ | incMAR | ldMAR | mxPCC | ldPC | incPC | mxPC | $mxPC_1$ |

| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|
| $ldR_1$ | $ldR_2$ | $ldR_3$ | mxA | ldRSRC | wrGPR | ldZ | ldD |

| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|
| mxRDST | ldRDST | ldX | ldY | $ldPOM_1$ | $ldPOM_2$ | $mxX_1$ | $mxX_0$ |

| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
|---|---|---|---|---|---|---|---|
| $mxY_1$ | $mxY_0$ | ldPSW | mxPSW | incSP | decSP | ldSP | ldACC |

| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
|---|---|---|---|---|---|---|---|
| $mxACC_1$ | mxACC | ldIVTP | ldB | incB | decB | slB | $mxB_2$ |

| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
|---|---|---|---|---|---|---|---|
| $mxB_1$ | $mxB_0$ | *cc* | | | | *ba* | |

| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
|---|---|---|---|---|---|---|---|
| *ba* | | | | | | / | / |

Slika 9 Mikroinstrukcija

Mikroprogram je dat u tabeli 25 .

Tabela 25 Mikroprogram (jedan tip mikroinstrukcije)

! Čitanje instrukcije !

$madr_{00}$ **ldMAR**, **incPC**;
$madr_{01}$ **ldMBR**;
$madr_{02}$ **ldIR1**;
$madr_{03}$ **brl1**, $\mathbf{madr_{1D}}$;

$madr_{04}$ **brl1', madr$_{0E}$**;
$madr_{05}$ **ldMAR, incPC**;
$madr_{06}$ **ldMBR**;
$madr_{07}$ **ldIR2**;
$madr_{08}$ **brl2, madr$_{1D}$**;
$madr_{09}$ **brl2', madr$_{0E}$**;
$madr_{0A}$ **ldMAR, incPC**;
$madr_{0B}$ **ldMBR;**
$madr_{0C}$ **ldIR3;**
$madr_{0D}$ **brl3, madr$_{1D}$;**

! Formiranje adrese i čitanje operanda !

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

! Direktno registarsko !

$madr_{0F}$ **ldRSRC**;
$madr_{10}$ **ldB**, **bruncnd**, **madr$_{1D}$**;

! Direktno memorijsko !

$madr_{11}$ **mxMAR$_1$**, **ldMAR**, **bruncnd,** $madr_{16}$;

! Indirektno registarsko sa pomerajem !

$madr_{12}$ **ldRSRC**;
$madr_{13}$ **mxX$_0$**, **ldX**,**mxY$_0$**, **mxY$_1$**, **ldY** ;
$madr_{14}$ **add**, **ldZ**;
$madr_{15}$ **mxMAR$_2$**, **ldMAR**;

! Čitanje operanda za memorijska adresiranja !

$madr_{16}$ **brSTORE,** $madr_{1D}$ ;
$madr_{17}$ **ldMBR**;
$madr_{18}$ **ldPOM1, incMAR**;
$madr_{19}$ **ldMBR**;
$madr_{1A}$ **ldPOM2**;
$madr_{1B}$ **mxB$_1$, mxB$_0$,ldB**, **bruncnd madr$_{1D}$**;

! Neposredno !

$madr_{1C}$ **mxB$_1$**, **ldB**;

! Izvršavanje operacije !

$madr_{1D}$ **bropr**;

! LOAD !

$madr_{1E}$ **mxACC**, **ldACC**, **bruncnd**, **madr$_{48}$**;

! STORE !

$madr_{1F}$ **brimmed**, **madr$_{56}$**;
$madr_{20}$ **brregdir**, **madr$_{25}$**;
$madr_{21}$ **mxMBR$_0$**, **ldMBR**;
$madr_{22}$ **wrMEM**;
$madr_{23}$ **mxMBR$_0$**, **mxA**, **ldMBR,incMAR**;
$madr_{24}$ **wrMEM**, **bruncnd, madr$_{56}$** ;
$madr_{25}$ **ldRDST**;
$madr_{26}$ **wrGPR**, **bruncnd**, **madr$_{56}$**;

! SUB !

madr$_{27}$ **ldX**, **ldY**;
madr$_{28}$ **sub**, **ldZ**;
madr$_{29}$ **ldACC**, **bruncd madr$_{56}$**;

! OR !

madr$_{2A}$ **ldX**, **ldY**;
madr$_{2B}$ **or**, **ldZ**;
madr$_{2C}$ **ldACC**, **bruncd madr$_{56}$**;

! ASL !

madr$_{2D}$ **ldY**;
madr$_{2E}$ **asl**, **ldZ**;
madr$_{2F}$ **ldACC**, **bruncd madr$_{56}$**;

! BNZ !

madr$_{30}$ **brneq, madr$_{56}$**;
madr$_{31}$ **mxX$_1$, mxX$_2$, ldX, mxY$_1$, mxY$_0$, ldY**;
madr$_{32}$ **add**, **ldZ**;
madr$_{33}$ **mxPC$_1$, ldPC, bruncd madr$_{56}$**;

! JSR !

madr$_{34}$ **decSP;**
madr$_{35}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
madr$_{36}$ **wrMEM;**
madr$_{37}$ **decSP;**
madr$_{38}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, mxMBR$_1$, ldMBR;**
madr$_{39}$ **wrMEM;**
madr$_{3A}$ **mxPC, mxPC$_1$, ldPC, bruncd madr$_{56}$**;

! JMP !

madr$_{3B}$ **mxPC$_1$** , **mxPC ,ldPC**, **bruncd madr$_{56}$**;

! RTI !

madr$_{3C}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
madr$_{3D}$ **ldMBR**;
madr$_{3E}$ **ldPSW**;

! RTS !

madr$_{3F}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR ,incSP**;
madr$_{40}$ **ldMBR**;
madr$_{41}$ **ldPOM2**;
madr$_{42}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
madr$_{43}$ **ldMBR**;
madr$_{44}$ **ldPOM1**;
madr$_{45}$ **mxPC, ldPC, bruncd madr$_{56}$**;

! PUSH !

madr$_{46}$ **decSP**;
madr$_{47}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, mxMBR$_0$** , **ldMBR**;
madr$_{48}$ **wrMEM**;
madr$_{49}$ **decSP**;
madr$_{4A}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, ldMBR**;
madr$_{4B}$ **wrMEM**, **bruncd madr$_{56}$**;

! POP !

madr$_{4C}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
madr$_{4D}$ **ldMBR**;
madr$_{4E}$ **ldPOM2**;

madr$_{4F}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR, incSP**;
madr$_{50}$ **ldMBR**;
madr$_{51}$ **ldPOM1**;
madr $_{52}$ **ldACC, mxACC$_1$**, **bruncd madr$_{56}$**;

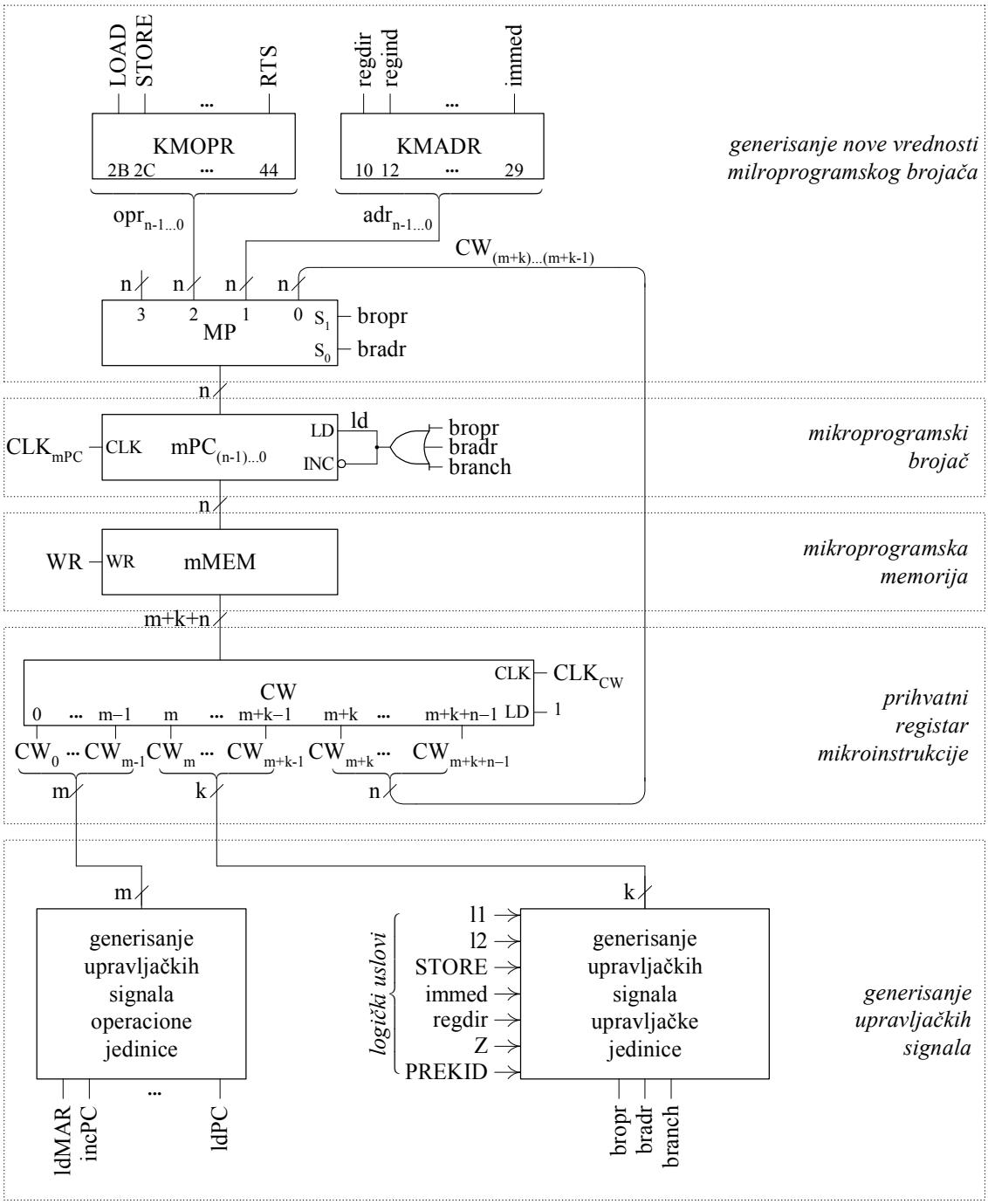
! JMPIND !

madr$_{53}$ **brimmed, madr$_{56}$**;
madr$_{54}$ **brregdir, madr$_{56}$**;
madr$_{55}$ **ldPC**, **bruncd madr$_{56}$**;

! Opsluživanje prekida !

madr$_{56}$ **brnotPREKID, madr$_{00}$**;
madr$_{57}$ **decSP**;
madr$_{58}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
madr$_{59}$ **wrMEM**;
madr$_{5A}$ **decSP**;
madr$_{5B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
madr$_{5C}$ **wrMEM**;
madr$_{5D}$ **decSP**;
madr$_{5E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMBR$_1$**, **ldMBR**;
madr$_{5F}$ **wrMEM**;
madr$_{60}$ **mxB$_2$, ldB**;
madr$_{61}$ **slB**;
madr$_{62}$ **mxX$_1$**, **ldX**, **mxY$_1$**, **ldY**;
madr$_{63}$ **add**, **ldZ**;
madr$_{64}$ **mxMAR$_2$**, **ldMAR**;
madr$_{65}$ **ldMBR**;
madr$_{66}$ **ldPOM$_2$, decMAR**;
madr$_{67}$ **ldMBR**;
madr$_{68}$ **ldPOM$_1$**;
madr$_{69}$ **mxPC**, **ldPC**;
madr$_{6A}$ **bruncd madr$_{00}$;**

Struktura upravljačke jedinice je data na slici 10.




Slika 10 Struktura upravljačke jedinice sa jednim tipom mikroinstrukcije

Upravljački signali operacione jedinice se generišu na sledeći način:

- **ldMAR = CW$_1$**
- **mxMAR$_0$ = CW$_4$**

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

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr = CW$_{60}$**$\cdot \overline{CW}_{61} \cdot \overline{CW}_{62} \cdot$**CW$_{63}$**
- **bradr = CW$_{60}$**$\cdot \overline{CW}_{61} \cdot \overline{CW}_{62} \cdot \overline{CW}_{63}$
- **branch = bruncnd**+ **brl1*l1** + **brl1´*l1´** + **brl2*l2** + **brl2´*l2´**+ **brl3*l3** + **brSTORE*STORE** + **brimmed*immed** + **brregdir*regdir** + **brneq*neq**+ **brnotPREKID*** $\overline{\text{PREKID}}$

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

- **bruncnd** = $\overline{CW}_{60} \cdot \overline{CW}_{61} \cdot \overline{CW}_{62} \cdot \mathbf{CW_{63}}$
- **brl1** = $\overline{CW}_{60} \cdot \overline{CW}_{61} \cdot \mathbf{CW_{62}} \cdot \overline{CW}_{63}$
- **brl1´** = $\overline{CW}_{60} \cdot \overline{CW}_{61} \cdot \mathbf{CW_{62}} \cdot \mathbf{CW_{63}}$
- **brl2=** $\overline{CW}_{60} \cdot \mathbf{CW_{61}} \cdot \overline{CW}_{62} \cdot \overline{CW}_{63}$
- **brl2´=** $\overline{CW}_{60} \cdot \mathbf{CW_{61}} \cdot \overline{CW}_{62} \cdot \mathbf{CW_{63}}$
- **brl3=** $\overline{CW}_{60} \cdot \mathbf{CW_{61}} \cdot \mathbf{CW_{62}} \cdot \overline{CW}_{63}$
- **brSTORE** = $\overline{CW}_{60} \cdot \mathbf{CW_{61}} \cdot \mathbf{CW_{62}} \cdot \mathbf{CW_{63}}$
- **brimmed** = $CW_{60} \cdot \overline{CW}_{61} \cdot \overline{CW}_{62} \cdot \overline{CW}_{63}$
- **brneq** = $CW_{60} \cdot \overline{CW}_{61} \cdot \overline{CW}_{62} \cdot \mathbf{CW_{63}}$
- **brregdir=** $\mathbf{CW_{60}} \cdot \overline{CW}_{61} \cdot \mathbf{CW_{62}} \cdot \overline{CW}_{63}$
- **brnotPREKID** = $CW_{60} \cdot \overline{CW}_{61} \cdot \mathbf{CW_{62}} \cdot \mathbf{CW_{63}}$

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**.