ELEKTROTEHNIČKI FAKULTET

UNIVERZITET U BEOGRADU

ARHITEKTURA I ORGANIZACIJA RAČUNARA 2

PROJEKAT

ZADATAK 3

MLADEN PANTIĆ 1/07
DRAGAN SLAVESKI 70/07
LAZAR DAVIDOVIĆ 78/07
MILAN BRANKOVIĆ 119/07

BEOGRAD 2010.

SADRŽAJ

SADRŽAJ ..... **I**

1 PROJEKTNI ZADATAK **1**

2 DIJAGRAM TOKA IZVRŠAVANJA OPERACIJA **5**

3 OPERACIONA JEDINICA **20**

3.1 OPERACIONA JEDINICA SA DVE MAGISTRALE 20

3.1.1 Struktura operacione jedinice *20*

3.1.2 Algoritam generisanja upravljačkih signala *23*

4 UPRAVLJAČKA JEDINICA **37**

4.1 OŽIČENA REALIZACIJA 37

4.1.1 Upravljačka jedinica bez spajanja koraka *37*

4.1.2 Upravljačka jedinica sa spajanjem koraka *47*

4.2 MIKROPROGRAMSKA REALIZACIJA 54

4.2.1 Mikroprogramska realizacija sa horizontalnim formatom mikroinstrukcija *54*

4.2.1.1 Mikroprogramska realizacija sa horizontalnim formatom mikroinstrukcija sa dva tipa instrukcija 54

4.2.1.2 Mikroprogramska realizacija sa horizontalnim formatom mikroinstrukcija sa jednim tipom instrukcija . 66

4.2.2 Mikroprogramska realizacija sa vertikalnim formatom mikroinstrukcija *73*

4.2.3 Mikroprogramska realizacija sa mešovitim formatom mikroinstrukcija *83*

5 PRILOZI **94**

OPERACIONA JEDINICA 94

5.1 BLOK INTERFEJS 96

5.2 BLOK REGISTRI 100

5.3 BLOK OPERACIJE 106

5.4 BLOK PREKIDI 108

6 MEMORIJA **111**

1 PROJEKTNI ZADATAK

Posmatra se deo računara koji čine memorija, procesor i magistrala.

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

Procesor je sa dvoadresnim formatom instrukcija. Podaci su celobrojne veličine bez i sa znakom u drugom komplementu dužine 2 bajta. Podaci u memoriji zauzimaju dve susedne memorijske lokacije, pri čemu se stariji bajt nalazi na nižoj a mlađi 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 4 bajta, prihvatni registri podataka X i Y dužine 2 bajta, registri opšte namene AX, BX, CX, DX, SP, BP, SI i DI dužine 2 bajta, programska statusna reč PSW dužine 1 bajt, registar maske IMR dužine 1 bajt i ukazivač na tabelu sa adresama prekidnih rutina IVTP dužine 2 bajta. Registri AX, BX, CX, DX, SP, BP, SI i DI se u zavisnosti od specificiranog načina adresiranja koriste kao registri podataka, adresni registri, bazni registri i indeksni registri, dok se registar SP u nekim instrukcijama implicitno koristi kao ukazivač na vrh steka. Instrukcije su dužine od 1 do 4 bajta.

Bit 7 prvog bajta instrukcije ima vrednost 0 za bezadresne instrukcije i instrukcije skoka, dok bit 6 prvog bajta instrukcije ima vrednost 0 za bezadresne instrukcije i vrednost 1 za instrukcije skoka. Bezadresne instrukcije su instrukcija povratka iz potprograma (RTS) i instrukcija povratka iz prekidne rutine (RTI). Bezadresne instrukcije su i instrukcije postavljanja indikatora I registra PSW na 1 i 0 (INTE i INTD) i indikatora T registra PSW na 1 i 0 (TRPE i TRPD). Bitovima 5 do 0 prvog bajta instrukcije specificira se kod operacije za bezadresne instrukcije. Dužina instrukcija je 1 bajt. U slučaju instrukcija skoka bit 5 prvog bajta instrukcije ima vrednost 0 za instrukcije bezuslovnog skoka i vrednost 1 za instrukcije uslovnog skoka. Instrukcije bezuslovnog skoka su instrukcija bezuslovnog skoka (JMP) i instrukcija skoka na potprogram (JSR). Instrukcije JMP i JSR se realizuju kao apsolutni skokovi, a adresa skoka je data 2. i 3. bajtom instrukcije, pri čemu je stariji bajt adrese skoka dat drugim a mlađi bajt trećim bajtom. Bitovima 4 do 0 prvog bajta instrukcije specificira se kod operacije za instrukcije bezuslovnog skoka. Dužina instrukcija je 3 bajta. Instrukcija uslovnog skoka je instrukcija skoka ukoliko rezultat nije nula (BNZ). 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. U ovu grupu spada i instrukcija prekida (INT). Broj ulaza u tabelu sa adresama prekidnih rutina je 8-mo bitna celobrojna veličina bez znaka data 2. bajtom instrukcije. Bitovima 4 do 0 prvog bajta instrukcije specificira se kod operacije za instrukcije uslovnog skoka. Dužina instrukcija je 2 bajta.

Bit 7 prvog bajta instrukcije ima vrednost 1 za adresne instrukcije. Adresne instrukcije obavezno imaju i drugi bajt, a u zavisnosti od specificiranog načina adresiranja koji je dat drugim bajtom instrukcije, dužina adresnih instrukcija može da bude 2, 3 ili 4 bajta. Adresne instrukcije mogu da budu jednoadresne i dvoadresne.

Jednoadresne instrukcije su instrukcija aritmetičkog pomeranja udesno (ASR), instrukcija prenosa na vrh steka (PUSH), instrukcija prenosa sa vrha steka (POP), instrukcija inkrementiranja (INC), instrukcija dekrementiranja (DEC) i instrukcija bezuslovnog indirektnog skoka (JMPIND). Instrukcije ASR, POP, INC i DEC postavljaju indikatore N, Z, C i V registra PSW. U instrukcijama ASR, POP, INC i DEC nije dozvoljeno neposredno adresiranje, a u instrukciji JMPIND nije dozvoljeno registarsko direktno i neposredno adresiranje, pa ukoliko se jave ova adresiranja u ovim instrukcijama, generiše se prekid zbog greške u adresiranju. Instrukcija ASR ima vrednost 0000000 na bitovima 6 do 0 prvog bajta

instrukcije, dok je kod operacije određen bitovima 5 do 3 drugog bajta instrukcije. Instrukcije PUSH, POP, INC, DEC i JMPIND imaju vrednost 1111111 na bitovima 6 do 0 prvog bajta instrukcije, dok je kod operacije određen bitovima 5 do 3 drugog bajta instrukcije. Operand može da bude registar opšte namene, neposredna veličina i memorijska lokacija u zavisnosti od specificiranog načina adresiranja. Načini adresiranja i registri opšte namene su određeni bitovima 7, 6, 2, 1 i 0 drugog bajta instrukcije. Dužina instrukcija je 2 ili 4 bajta i zavisi od specificiranog načina adresiranja.

Dvoadresne instrukcije su instrukcija prenosa (MOVS i MOVD), aritmetička instrukcija sabiranja (ADD) i logička instrukcija I (AND). Instrukcije MOVS, MOVD, ADD i AND postavljaju indikatore N, Z, C i V registra PSW. Kod operacije dvoadresnih instrukcija određen je vrednostima 0000001 do 1111110 bitova 6 do 0 prvog bajta instrukcije Prvi operand je registar opšte namene specificiran bitovima 5 do 3 drugog bajta instrukcije. Drugi operand može da bude registar opšte namene, neposredna veličina i memorijska lokacija u zavisnosti od specificiranog načina adresiranja. Načini adresiranja i registri opšte namene su određeni bitovima 7, 6, 2, 1 i 0 drugog bajta instrukcije. Za instrukciju MOVS izvorište je drugi operand a adredište prvi operand, za instrukciju MOVD izvorište je prvi operand a odredište drugi operand, dok je za instrukcije ADD i AND prvi operand istovremeno odredište i drugo izvorište, dok je drugi operand prvo izvorište. U odredištu instrukcije MOVD nije dozvoljeno neposredno adresiranje, pa ukoliko se neposredno adresiranje javi u odredištu, generiše se prekid zbog greške u adresiranju. Dužina instrukcija je 2, 3 ili 4 bajta i zavisi od specificiranog načina adresiranja.

Pri pojavi neiskorišćenih kodova operacija generiše se prekid zbog greške u kodu operacije.

Specifikacija operanda kod jednoadresnih instrukcija i drugog operanda kod dvoadresnih instrukcija je identična. Načini adresiranja i registri opšte namene su određeni bitovima 7, 6, 2, 1 i 0 drugog bajta instrukcije. Dužina instrukcija je 2, 3 ili 4 bajta i zavisi od specificiranog načina adresiranja.

Bitovima 7 i 6 drugog bajta instrukcije se specificiraju sledeća adresiranja: 00-registarsko direktno adresiranje, 01-registarsko indirektno adresiranje, 10-registarsko indirektno adresiranje sa pomerajem i 11-ostala adresiranja. Registarsko direktno adresiranje koristi neki od registara opšte namene AX, BX, CX, DX, SP, BP, SI i DI specificiran vrednostima 0 do 7, respektivno, bitova 2 do 0 drugog bajta instrukcije. Dužina instrukcije je 2 bajta. Registarsko indirektno adresiranje koristi ili pojedinačno (BX, BP, SI i DI) ili u kombinacijama (BX+SI, BX+DI, BP+SI, BP+DI) neke od registara opšte namene specificirane vrednostima 0 do 3 i 4 do 7, respektivno, bitova 2 do 0 drugog bajta instrukcije. Dužina instrukcije je 2 bajta. Registarsko indirektno adresiranje sa pomerajem koristi ili pojedinačno (BX, BP, SI i DI) ili u kombinacijama (BX+SI, BX+DI, BP+SI, BP+DI) neke od registara opšte namene specificirane vrednostima 0 do 3 i 4 do 7, respektivno, bitova 2 do 0 drugog bajta instrukcije i 8-mo bitni pomeraj koji se daje kao celobrojna veličina sa znakom 3. bajtom instrukcije. Dužina instrukcije je 3 bajta.

Ostala adresiranja se specificiraju bitovima 1 i 0 drugog bajta instrukcije i to na sledeći način: 00-memorijsko direktno adresiranje, 01-memorijsko indirektno adresiranje, 10-relativno adresiranje i 11-neposredno adresiranje. Bit 2 drugog bajta instrukcije se kod ovih adresiranja ne koristi. Memorijsko direktno adresiranje, memorijsko indirektno adresiranje i neposredno dugo adresiranje imaju i 3. i 4. bajt instrukcije, dok relativno adresiranje ima i 3. bajt instrukcije. Kod memorijskog direktnog i memorijskog indirektnog adresiranja 3. i 4. bajt instrukcije sadrže adresu memorijske lokacije, pri čemu je stariji bajt adrese memorijske lokacije dat 3. a mlađi bajt 4. bajtom instrukcije. Kod memorijskog indirektnog adresiranja adresa dužine 16 bita zauzima dve susedne memorijske lokacije, pri čemu se stariji bajt nalazi na nižoj a mlađi bajt na višoj adresi. Dužina instrukcije je 4. bajta. Kod relativnog adresiranja

3. bajt instrukcije sadrže 8-mo bitni pomeraj koji je dat kao celobrojna veličina sa znakom. Dužina instrukcije je 3 bajta. Kod neposrednog adresiranja 3. i 4. bajt instrukcije sadrže 16-to bitni podatak, pri čemu je stariji bajt podatka dat 3. a mlađi bajt 4. bajtom instrukcije. Dužina instrukcije je 4 bajta.

Stek raste prema nižim memorijskim lokacijama, a registar SP ukazuje na zadnju zauzetu memorijsku lokaciju.

Spoljašnji maskirajući zahtevi za prekid dolaze od 3 ulazno/izlazna uređaja po linijama označenim sa intr1, intr2 i intr3. Po liniji intr1 stiže zahtev za prekid najnižeg, a po liniji intr3 najvišeg prioriteta. Zahtevi za prekid intr1, intr2 i intr3 se mogu selektivno maskirati razredima IMR1, IMR2 i IMR3 registra maske IMR. Adrese prekidnih rutina 3 ulazno/izlazna uređaja koji po linijama intr1, intr2 i intr3 šalju zahteve za prekid nalaze se u ulazima 5, 6 i 7 tabele sa adresama prekidnih rutina. Brojevi ulaza su određeni fiksno. Zahtevi za prekid dolaze kao impulsi. Zahtevi za prekid se prihvataju ukoliko je njihov nivo viši od nivoa prioriteta tekućeg programa. Spoljašnji nekaskirajući zahtev za prekid dolazi po liniji inm i to kao impuls. Unutrašnji prekidi su prekid zbog greške u kodu operacije, prekid zbog greške u adresiranju i prekid zbog zadatog režima rada prekid posle svake instrukcije. Adrese prekidnih rutina za prekid zbog greške u kodu operacije, prekid zbog greške u adresiranju, spoljašnji nemaskirajući prekid i prekid zbog zadatog režima rada prekid posle svake instrukcije, nalaze se u ulazima 3, 2, 1 i 0 tabele sa adresama prekidnih rutina. Brojevi ulaza su određeni fiksno. Adrese dužine 16 bita zauzimaju po dve susedne memorijske lokacije, pri čemu se stariji bajt nalazi na nižoj a mlađi bajt na višoj adresi. Početna adresa tabele sa adresama prekidnih rutina se nalazi u registru IVTP dužine jedan 16 bita. U okviru hardverskog dela opsluživanja zahteva za prekid na stek sa stavljaju samo registri PC i PSW, u razrede I i T registra PSW upisuju 0 i u slučaju maskirajućih prekida u razrede L1 i L0 registra PSW upisuje nivo prioriteta prekidne rutine na koju se skače.

Magistrala je sinhrona sa atomskim ciklusima čitanja i upisa. Koristi se paralelni arbitrator sa 4 para linija za zahtev i potvrdu. Procesor, memorija i magistrala imaju zajednički signal takta. Postoji i signal zauzeća magistrale.

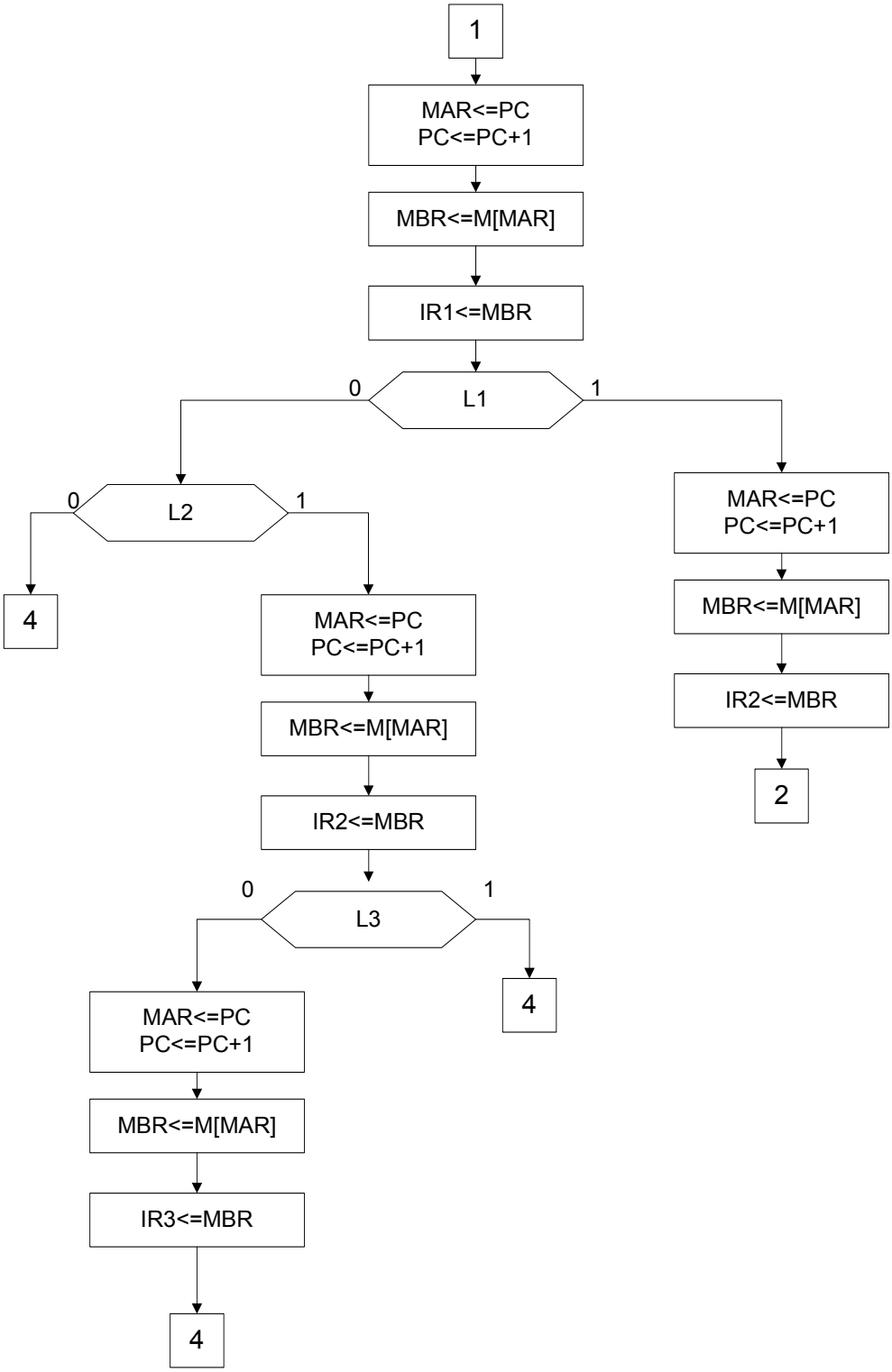
Isprojektovati procesor, memoriju i arbitrator magistrale. Operacionu jedinicu procesora realizovati sa dve magistrale. Upravljačke jedinice procesora realizovati u tehnikama 1. ožičene realizacije bez i sa spajanjem koraka, 2. mikroprogramske realizacije i horizontalnim kodiranjem upravljačkih signala sa dva tipa i jednim tipom mikroinstrukcija, 3. mikroprogramske realizacije i vertikalnim kodiranjem upravljačkih signala sa dva tipa i jednim tipom mikroinstrukcija i 4. mikroprogramske realizacije i mešovitim kodiranjem upravljačkih signala sa dva tipa i jednim tipom mikroinstrukcija.

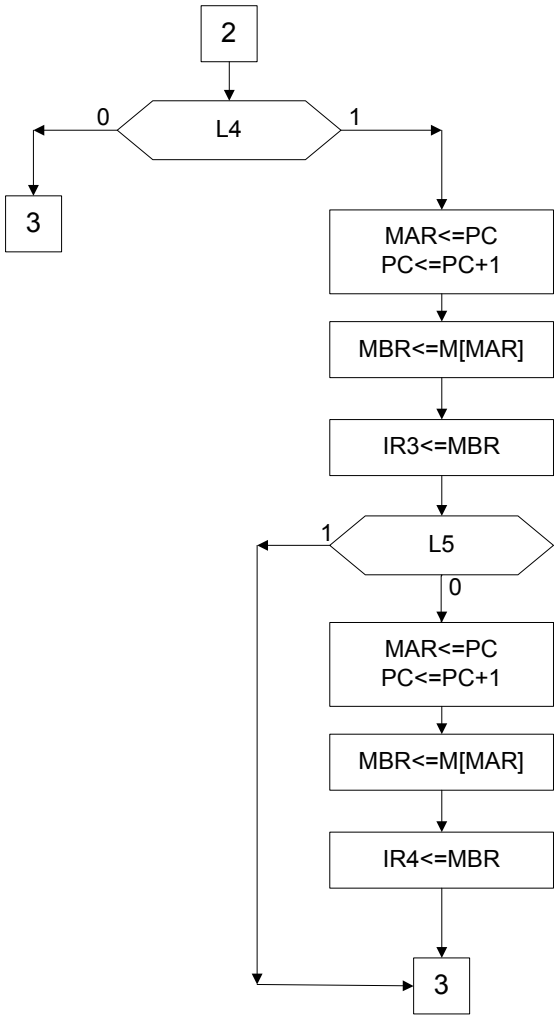
2 DIJAGRAM TOKA IZVRŠAVANJA OPERACIJA

Najpre treba nacrtati dijagram toka faza izvršavanja instrukcije i to: faze čitanja instrukcije, faze formiranja adrese i čitanja operanda, faza izvršavanja operacija MOVS, MOVD, ADD, AND, ASR, BNZ, JMP, JMPIND, JSR, RTS, RTI, PUSH, POP, INTE, INTD, TRPE, TRPD, INT, INC i DEC i faze opsluživanja zahteva za prekid. Dijagram toka izvršavanja instrukcije je dat na slikama 1.a, 1.b, 1.c i 1.d. Izvršavanje instrukcije se sastoji iz četiri faze: čitanje instrukcije (slika 1.a), formiranje adrese i čitanje operanda (slika 1.b), izvršavanje operacija (slika 1.c) i opsluživanje prekida (slika 1.d).

čitanje instrukcije (slika 1.*a)*

Instrukcija se čita iz memorije počev od adrese na koju ukazuju trenutka vrednost programskog brojača PC. Čita se reč po reč i posle svake pročitane reči vrednost PC se inkrementira. Pročitane reči instrukcije se smeštaju u registre IR1 do IR4 koji čine prihvatni registar instrukcije IR. Broj pročitanih reči zavisi od instrukcije. Bezadresne instrukcije sadrže samo kod operacije, pa zato treba pročitati samo jednu reč. Ove instrukcije ne prolaze kroz fazu *formiranje adrese i čitanje operanda*, pa zato posle čitanja jedne reči treba odmah preći na fazu *izvršavanje operacija*. Instrukcije skoka sadrže kod operacije i adresu skoka, pa zato treba pročitati nekoliko reči i to prvu reč koja sadrži kod operacije i drugu i nekoliko sledećih nekoliko reči specificiraju adresu skoka. Ove instrukcije, takođe, ne prolaze kroz fazu *formiranje adrese i čitanje operanda*, pa zato posle čitanja odgovarajućeg broja reči treba odmah preći na fazu *izvršavanje operacija*. Adresne instrukcije obavezno sadrže dve reči i to prvu reč koja specificira kod operacije i drugu reč koja specificira način adresiranja i adresu registra opšte namene. Kod registarskih adresiranja dužina instrukcije je dve ili tri reči, dok kod memorijskih adresiranja treća i četvrta reč specificiraju adresu, pomeraj ili neposrednu veličinu. Zbog toga kod ovih instrukcija posle čitanja dve reči kod registarskih adresiranja i treće i četvrte reči kod memorijskih adresiranja treba preći na fazu *formiranje adrese i čitanje operanda*. Prilikom čitanja reči instrukcije formiraju se signali dužine instrukcije L1, L2 do L5, koji predstavljaju signale logičkih uslova. Ovi signali označavaju da je dužina instrukcije jedna, dve do 4 reči. U slučaju bezadresnih instrukcija i instrukcija skoka posle čitanja prve reči koja sadrži kod operacije treba da se formira neaktivna vrednost signala L1. U slučaju bezadresnih instrukcija treba posle čitanja prve reči koja sadrži kod operacije, i provere uslova L1 da se formira neaktivna vrednost signala L2. U slučaju instrukcija skoka treba posle čitanja prve reči koja sadrži kod operacije, i provere uslova L1 da se formira aktivna vrednost signala L2. U slučaju adresnih instrukcija treba posle čitanja prve reči koja sadrži kod operacije da se formira aktivna vrednost signala L1. Na osnovu aktivne vrednosti signala L1 treba da se pročita druga reč instrukcije koja sadrži specifikaciju način adresiranja i adresu registra opšte namene. Na osnovu specifikacije načina adresiranja treba da se formira neaktivna vrednost signal L4 za registarska adresiranja i aktivna vrednost signala L4 i aktivna vrednost jednog od signala dužine instrukcije za memorijska adresiranja.


1
MAR<=PC
PC<=PC+1
MBR<=M[MAR]
IR1<=MBR
L1
0
1
L2
0
1
4
MAR<=PC
PC<=PC+1
MBR<=M[MAR]
IR2<=MBR
L3
0
1
MAR<=PC
PC<=PC+1
MBR<=M[MAR]
IR3<=MBR
4
4
MAR<=PC
PC<=PC+1
MBR<=M[MAR]
IR2<=MBR
2



Slika 1.a Dijagram toka – faza čitanje instrukcije

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

Sada se vrši provera signala logičkog uslova L1. Ukoliko se radi o bezadresnoj instrukciji ili instrukciji skoka signal L1 je neaktivan, pa se proverava logički uskov L2. Ukoliko je on neaktivan, radi se o bezadresnoj instrukciji, pa se prelazi na korak 4 i fazu *izvršavanje operacije* (slika 1.c). Ukoliko se radi o instrukcijama skoka L2 je aktivan, pa se prelazi na čitanje druge reči instrukcije. Čitanje druge reči instrukcije se realizuje na sličan način kao i čitanje prve reči instrukcije. Druga reč instrukcije se upisuje u registar IR2.

Sada se vrši provera signala logičkog uslova L3. Ukoliko se radi o instrukciji uslovnog skoka, signal L3 je aktivan, pa se prelazi na korak 4 i fazu *izvršavanje operacije* (slika 1.c). Ukoliko se radi o instrukciji bezuslovnog skoka čija je dužina, tri reči, signal L3 je neaktivan, pa se prelazi na čitanje treće reči instrukcije. Čitanje treće reči instrukcije se realizuje na sličan način kao i čitanje prve i druge reči instrukcije. Treća reč instrukcije se upisuje u registar IR3, pa se prelazi na korak 4 i fazu *izvršavanje operacije* (slika 1.c).

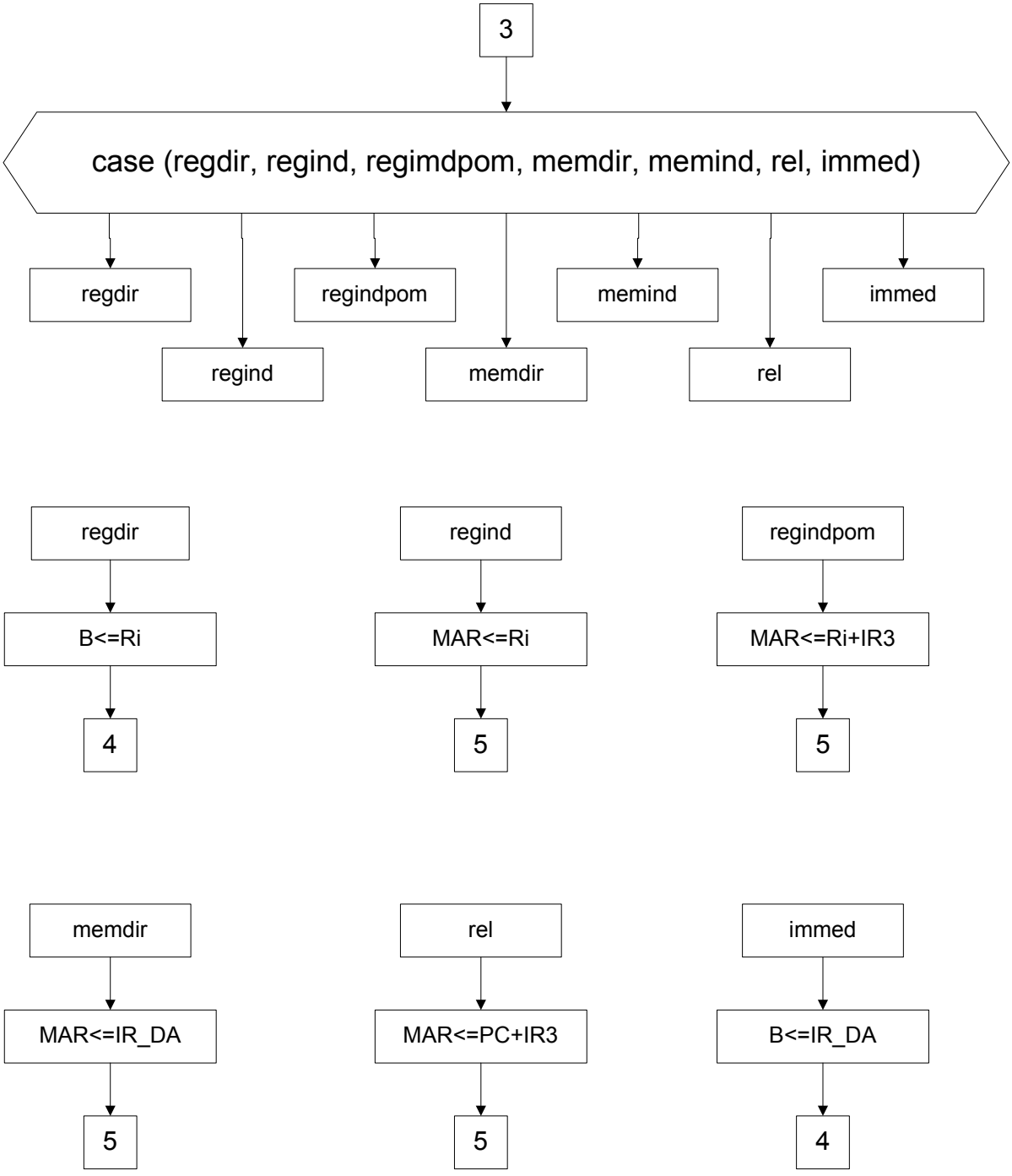
Ukoliko je signal logičkog uslova L1 aktivan, radi se on nekoj od adresnih instrukcija, pa se prelazi na čitanje druge reči instrukcije. Čitanje druge reči instrukcije se realizuje na sličan način kao i čitanje prve reči instrukcije. Druga reč instrukcije se upisuje u registar IR2. Logički uslovi L4 i L5 se generišu u zavisnosti od specificiranog načina adresiranja i oni

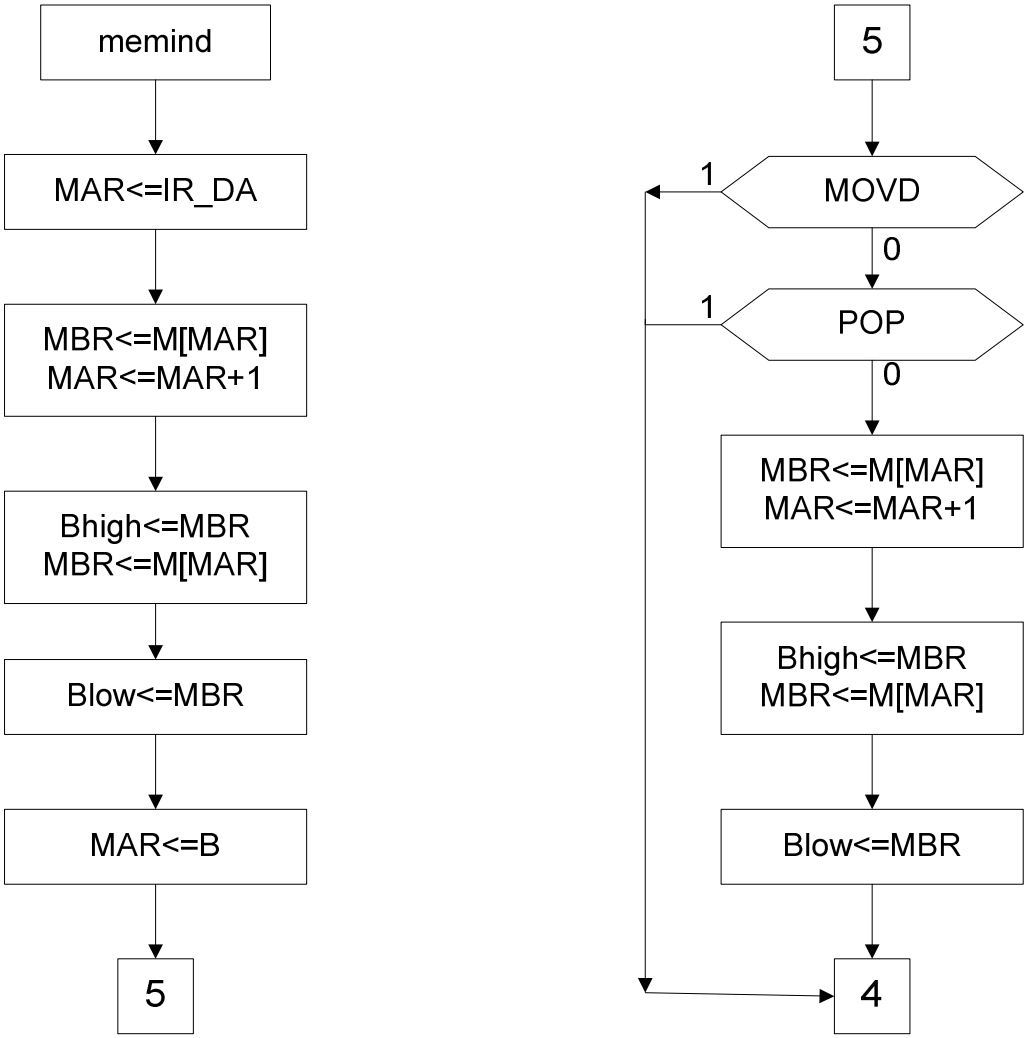
određuju dužinu odgovarajuće instrukcije. Prvo se vrši provera signala logičkog uslova L4. Ukoliko je signal L4 neaktivan tada se radi o registarskim adresiranjima bez pomeraja i tada je dužina instrukcije 2 bajta pa se prelazi na korak 3 i fazu *formiranje adrese i čitanje operanda* (slika 1.b). Ukoliko je signal L4 aktivan tada je dužina instrukcije 3 ili 4 bajta pa se čita 3 reč instrukcije. Čitanje treće reči instrukcije se realizuje na sličan način kao i čitanje prve ili druge reči instrukcije. Treća reč instrukcije se upisuje u registar IR3.

Sada se vrši provera signala logičkog uslova L5. Ukoliko se radi o adresnoj instrukciji sa nekim od memorijskih adresiranja ili neposrednim adresiranjem, signal L5 je neaktivan, pa se prelazi na čitanje četvrte reči instrukcije. Čitanje četvrte reči instrukcije se realizuje na sličan način kao i čitanje prve, druge ili treće reči instrukcije. Četvrta reč instrukcije se upisuje u registar IR4 i zatim se prelazi na korak 3 tj. fazu *formiranje adrese i čitanje operanda* (slika 1.b). Ukoliko se radi o adresnoj instrukciji sa relativnim adresiranjem ili registarskim indirektnim adresiranjem sa pomerajem, signal L5 je aktivan, pa se prelazi na korak 3 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 po posebnom algoritmu za svaki od načina adresiranja prolaskom kroz odgovarajuće korake. Na početku se realizuje višestruki uslovni skok na odgovarajući dijagram toka na osnovu toga koji je od signala logičkih uslova načina adresiranja aktivan. Aktivna vrednost jednog od signala načina adresiranja regdir, regind, regindpom, memdir, memind, rel i immed određuje da je specificirano registarsko direktno adresiranje, registarsko indirektno adresiranje, registarsko indirektno adresiranje sa pomerajem, memorijsko direktno adresiranje, memorijsko indirektno adresiranje, relativno i neposredno adresiranje, respektivno.





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

Ukoliko je signal regdir aktivan, radi se o registarskom direktnom adresiranju. Operand je tada u registru opšte namene R_i određenom vrednošću bitova 1 i 0 druge reči instrukcije iz registra IR2. Selektovani registar opšte namene R_i se prebacuje u registar B. Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 4 i fazu *izvršavanje operacija* (slika 1.c).

Ukoliko je signal regind aktivan, radi se o registarskom indirektnom adresiranju. Operand je tada u memoriji na adresi koja se nalazi u registru opšte namene R_i određenom vrednošću bitova 1 i 0 druge reči instrukcije iz registra IR2. Selektovani registar opšte namene R_i se prebacuje u registar registar MAR i prelazi se na korak 5.

Počev od koraka 5 se za sve operacije, sem operacije MOVD i POP, čita operand i smešta u registar B. Zbog toga se ovde vrši provera signala operacije MOVD, a zatim i POP. Ukoliko je njihova vrednost 0, iz memorije M se sa adrese određene sadržajem registra MAR čita reč i upisuje u registar MBR, registar MAR se uvećava za jedan, a zatim se sadržaj registra MBR prebacuje u viši razred registara B. Potom se ponavlja ova operacija, ali se sada pročitani podatak koji se nalazi u registru MBR upisuje u niži razred registra B. Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 4 i fazu *izvršavanje operacija* (slika 1.c). Ukoliko je vrednost signala MOVD ili POP 1, odmah se prelazi na korak 4 i fazu *izvršavanje operacija* (slika 1.c).

Ukoliko je signal regindpom aktivan, radi se o registarskom indirektnom adresiranju sa pomerajem. Operand je tada u memoriji na adresi koja se dobija sabiranjem sadržaja registra

R_i određenog vrednošću bitova 2, 1 i 0 druge reči instrukcije iz registra IR2 sa pomerajem, dok se pomeraj nalazi u registru IR3. Dobijena adresa se prebacuje u registar MAR i prelazi na korak 5 počev od koga se, na već opisani način, za sve operacije, sem operacije MOVD i POP, 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 memdir aktivan, radi se o memorijskom direktnom adresiranju. Operand je tada u memoriji na adresi koja se nalazi u razredima IR3 i IR4 označenim sa IR_DA čiji se sadržaj prebacuje u registar MAR i prelazi na korak 5 počev od koga se, na već opisani način, za sve operacije, sem operacije MOVD i POP, 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 memind aktivan, radi se o memorijskom indirektnom adresiranju. Operand je tada u memoriji na adresi određenoj sadržajem memorijske lokacije čija se adresa nalazi u registrima IR3 i IR4 označenim sa IR_DA. Stoga se, najpre, sadržaj ovih registara prebacuje u registar MAR, pa se iz memorije M sa adrese određene sadržajem registra MAR čita reč i upisuje u registar MBR, i registar MAR se uvećava za jedan. Potom se registar MBR prebacuje u viši razred registra B, i postupak se ponavlja samo što sada očitani podatak iz memorije upisujemo u niži razred registra B. Operand je u memoriji na adresi koja se nalazi u registru B čiji se sadržaj prebacuje u registar MAR i prelazi na korak 5 počev od koga se, na već opisani način, za sve operacije, sem operacije MOVD i POP, 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 immed aktivan, radi se o neposrednom adresiranju. Operand se tada nalazi u registrima IR2 i IR3 označenim sa IR_DA, čiji se sadržaj prebacuje u registar B. Time je završena faza *formiranje adrese i čitanje operanda* i prelazi se na korak 4 i fazu *izvršavanje operacija* (slika 1.c).

Ukoliko je signal rel aktivan, radi se o relativnom adresiranju. Operand je tada u memoriji na adresi koja se dobija sabiranjem sadržaja registra PC i pomeraja koji se nalazi u registru IR3. Dobijena adresa se prebacuje u registar MAR i prelazi na korak 5 počev od koga se, na već opisani način, za sve operacije, sem operacije MOVD i POP, 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).

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 MOVS, MOVD, ADD, AND, ASR, BNZ, JMP, JMPIND, JSR, RTS, RTI, PUSH, POP, INTE, INTD, TRPE, TRPD, INT, INC ili DEC određuje da je specificirana operacija prenosa u registar, operacija prenosa iz registra, aritmetička operacija sabiranja, logička operacija logički proizvod, operacija aritmetičkog pomeranja udesno za jedno mesto, operacija uslovnog skoka ukoliko je rezultat nije nula, operacija bezuslovnog skoka, instrukcija bezuslovnog indirektnog skoka, instrukcija skoka na potprograma, instrukcija povratka iz prekidne rutine, instrukcija povratka iz potprograma, operacija smeštanja na vrh steka, operacija skidanja sa vrha steka, operacija postavljanja indikatora I registra PSW na 1 i 0, operacija postavljanja indikatora T registra PSW na 1 i 0, instrukcija prekida, instrukcija inkrementiranja ili instrukcija dekrementiranja respektivno.

Ukoliko je signal MOVS aktivan, sadržaj registra B se prebacuje u neki od registara registarskog fajla, specificiran bitima 1 i 0 drugog bajta instrukcije. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal MOVD aktivan, najpre se proverava da li je aktivan signal immed, i ukoliko jeste prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d), jer u ovoj instrukciji nije dozvoljeno neposredno adresiranje. Ukoliko je signal immed neaktivan sadržaj jednog od registara registarskog fajla, određen bitovima 5,4 i 3 registra IR2, se upisuje u registar B. Time je završena faza *izvršavanje operacija* i prelazi se na korak 7 i fazu *vraćanje podatka* (slika 1.c).

Ukoliko je signal ADD aktivan, sabiraju se sadržaji jednog registara registarskog fajla specificiran bitima 1 i 0 registra IR2 i B i rezultat upisuje u jedan od registara registarskog fajla specificiran bitima 1 i 0 registra IR2. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal AND aktivan, logička I operacija se realizuje nad sadržajima jednog registara registarskog fajla specificiran bitima 1 i 0 registra IR2 i B i rezultat upisuje u jedan od registara registarskog fajla specificiran bitima 1 i 0 registra IR2. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal ASR aktivan, najpre se proverava da li je aktivan signal immed, i ukoliko jeste prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d), jer u ovoj instrukciji nije dozvoljeno neposredno adresiranje. Ukoliko je signal immed neaktivan sadržaj registra B se aritmetički pomera udesno za jedno mesto i upisuje u registar B. Time je završena faza *izvršavanje operacija* i prelazi se na korak 7 i fazu *vraćanje podatka* (slika 1.c).

Ukoliko je signal INC aktivan, najpre se proverava da li je aktivan signal immed, i ukoliko jeste prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d), jer u ovoj instrukciji nije dozvoljeno neposredno adresiranje. Ukoliko je signal immed neaktivan sadržaj registra B se inkrementira i upisuje u registar B. Time je završena faza *izvršavanje operacija* i prelazi se na korak 7 i fazu *vraćanje podatka* (slika 1.c).

Ukoliko je signal DEC aktivan, najpre se proverava da li je aktivan signal immed, i ukoliko jeste prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d), jer u ovoj instrukciji nije dozvoljeno neposredno adresiranje. Ukoliko je signal immed neaktivan sadržaj registra B se dekrementira i upisuje u registar B. Time je završena faza *izvršavanje operacija* i prelazi se na korak 7 i fazu *vraćanje podatka* (slika 1.c).

Ukoliko je signal INTE aktivan, bit I registra PSW se postavlja na jedan. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal INTD aktivan, bit I registra PSW se postavlja na nula. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal TRPE aktivan, bit T registra PSW se postavlja na jedan. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal TRPD aktivan, bit T registra PSW se postavlja na nula. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal BNZ aktivan, uslovni skok na osnovu vrednosti signala logičkog uslova rezultata operacija eql se realizuje. Signali logičkih uslova rezultata operacija eql (rezultat nula) se formiraju na osnovu vrednosti indikatora Z registra programske statusne reči PSW. Signal rezultata operacija eql ima vrednost 1 ukoliko je rezultat zadnje izvršene instrukcije 0 i vrednost 0 ukoliko rezultat zadnje izvršene instrukcije nije 0. Ukoliko je signal eql 1, uslov za skok nije ispunjen. Time je završena faza *izvršavanje operacija* i prelazi se na korak 5 i fazu *opsluživanje prekida* (slika 1.d). Ukoliko je signal eql 0, uslov za skok je ispunjen, pa se sadržaj registara PC uvećan za vrednost registra IR2, upisuje u registar PC. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

Ukoliko je signal JMP aktivan, bezuslovni skok se realizuje. Sadržaj registara IR2 i IR3 označeni sa IR_JA, koji predstavlja adresu skoka, upisuje se u registar PC. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

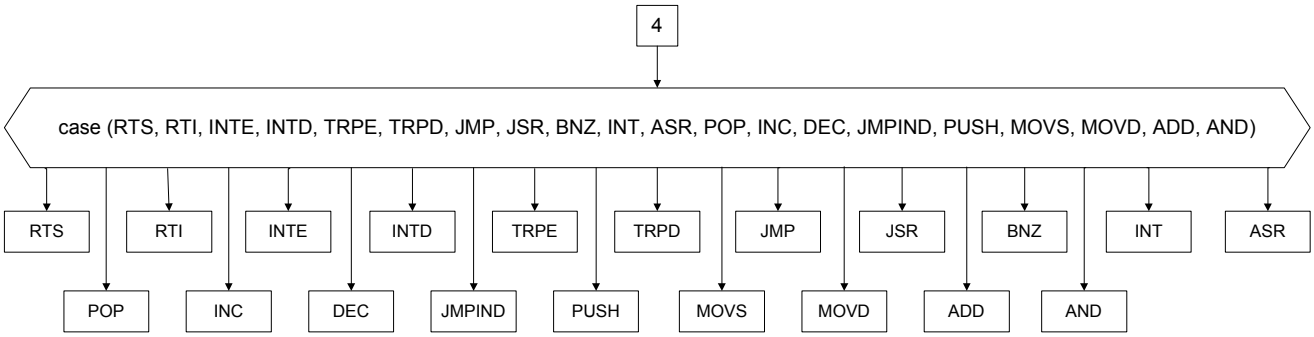
Ukoliko je signal JSR aktivan, skok na potprogram se realizuje. U okviru toga se sadržaj registra PC stavlja na stek. Najpre se registar SP dekrementira i nižih osam bita prebacuje u registar MBR. Zatim se sadržaj registra SP upisuje u registar MAR. Potom se registar MBR upisuje u memorijsku lokaciju određenu sadržajem registra MAR. Isti korak se ponavlja samo što se sada viših osam bita registra PC prebacuju u registar MBR. Na kraju se sadržaj registara IR2 i IR3 označeni sa IR_JA, koji predstavlja adresu skoka, upisuje se u registar PC. Treba uočiti da stek raste prema nižim lokacijama i da registar SP ukazuje na prvu popunjenu lokaciju. S toga se prilikom upisa na stek, prvo sadržaj registra SP dekrementira i posle toga upisuje u registar MAR. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

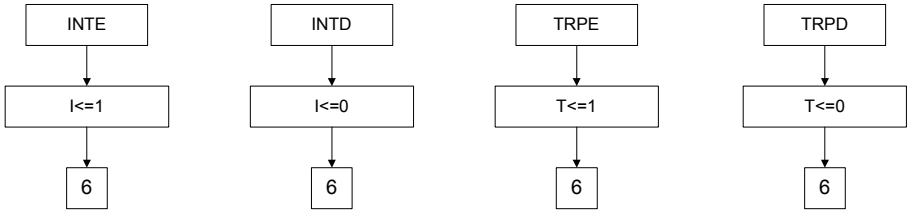
Ukoliko je signal JMPIND aktivan, bezuslovni indirektni skok se realizuje. Najpre se proveravaju uslovi immed i regind. Ukoliko je bilo koji od njih aktivan prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d), pošto u ovoj instrukciji nije dozvoljeno neposredno i registarsko indirektno adresiranje. Ukoliko ni jedan od ovih uslova nije ispunjen sadržaj registara B, koji predstavlja adresu skoka, upisuje se u registar PC. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

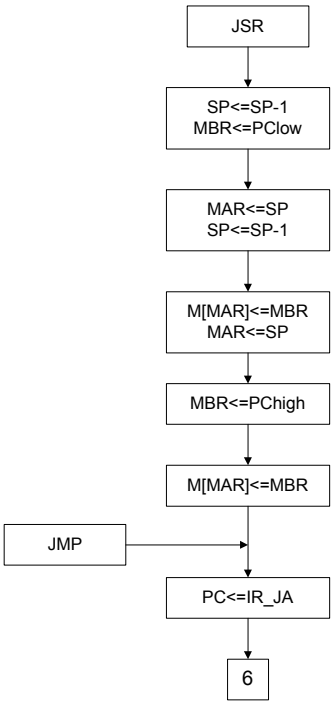
Ukoliko je signal RTI aktivan, povratak iz prekidne rutine se realizuje. U okviru toga se sadržajima sa steka restauriraju sadržaji registara PSW i PC. Najpre se sadržaj registra SP prebacuje u registar MAR i inkrementira. Zatim se iz memorijske lokacije određene sadržajem registra MAR čita sadržaj i 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 viših osam bita registara PC, a potom se na isti način sa steka čita još jedna reč i upisuje u nižih osam bita registara PC. Treba uočiti da stek raste prema nižim lokacijama i da registar SP ukazuje na prvu popunjenu lokaciju. S toga se prilikom čitanja sadržaja sa steka, prvo sadržaj registra SP prebacuje u registar MAR i posle toga inkrementira. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

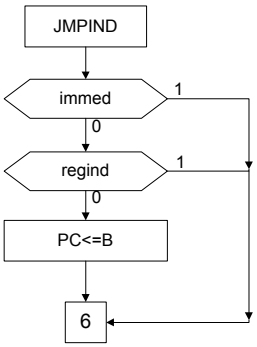
Ukoliko je signal RTS aktivan, povratak iz potprograma se realizuje. U okviru toga se sadržajem sa steka restaurira sadržaj registra PC. Ovo se realizuje na identičan kao i u slučaju instrukcije RTI. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

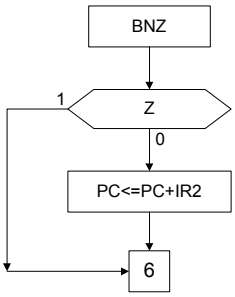
Ukoliko je signal INT aktivan, instrukcija prekida se realizuje. U okviru toga prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

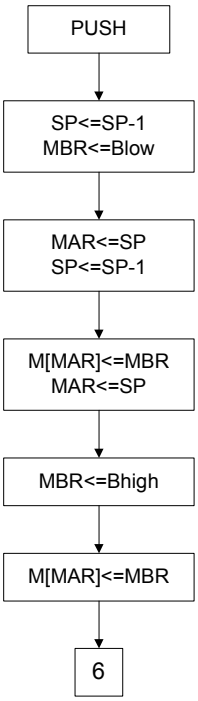

4
case (RTS, RTI, INTE, INTD, TRPE, TRPD, JMP, JSR, BNZ, INT, ASR, POP, INC, DEC, JMPIND, PUSH, MOVS, MOVD, ADD, AND)
RTS
RTI
INTE
INTD
TRPE
TRPD
JMP
JSR
BNZ
INT
ASR
POP
INC
DEC
JMPIND
PUSH
MOVS
MOVD
ADD
AND

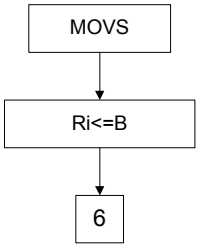

INTE
I<=1
6
INTD
I<=0
6
TRPE
T<=1
6
TRPD
T<=0
6

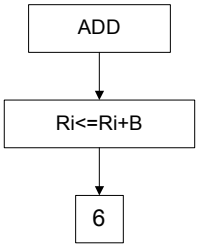

JSR
SP<=SP-1
MBR<=PClow
MAR<=SP
SP<=SP-1
M[MAR]<=MBR
MAR<=SP
MBR<=PChigh
M[MAR]<=MBR
JMP
PC<=IR_JA
6

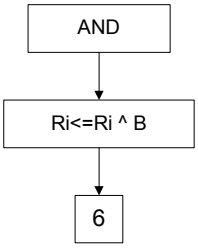

JMPIND
immed
1
0
regind
1
0
PC<=B
6

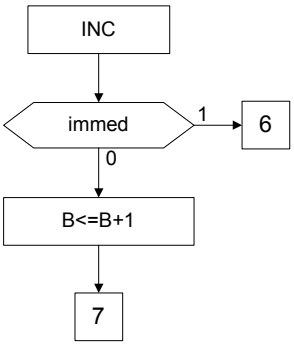

BNZ
1
Z
0
PC<=PC+IR2
6

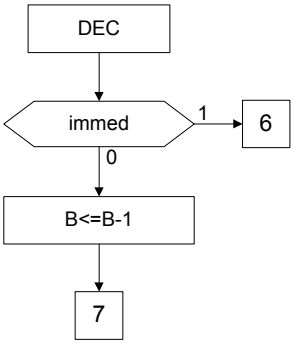

PUSH
SP<=SP-1
MBR<=Blow
MAR<=SP
SP<=SP-1
M[MAR]<=MBR
MAR<=SP
MBR<=Bhigh
M[MAR]<=MBR
6

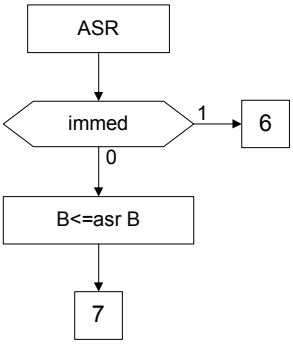

MOVS
Ri<=B
6

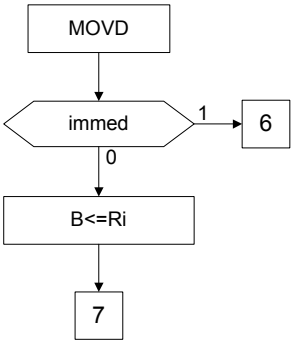

ADD
Ri<=Ri+B
6

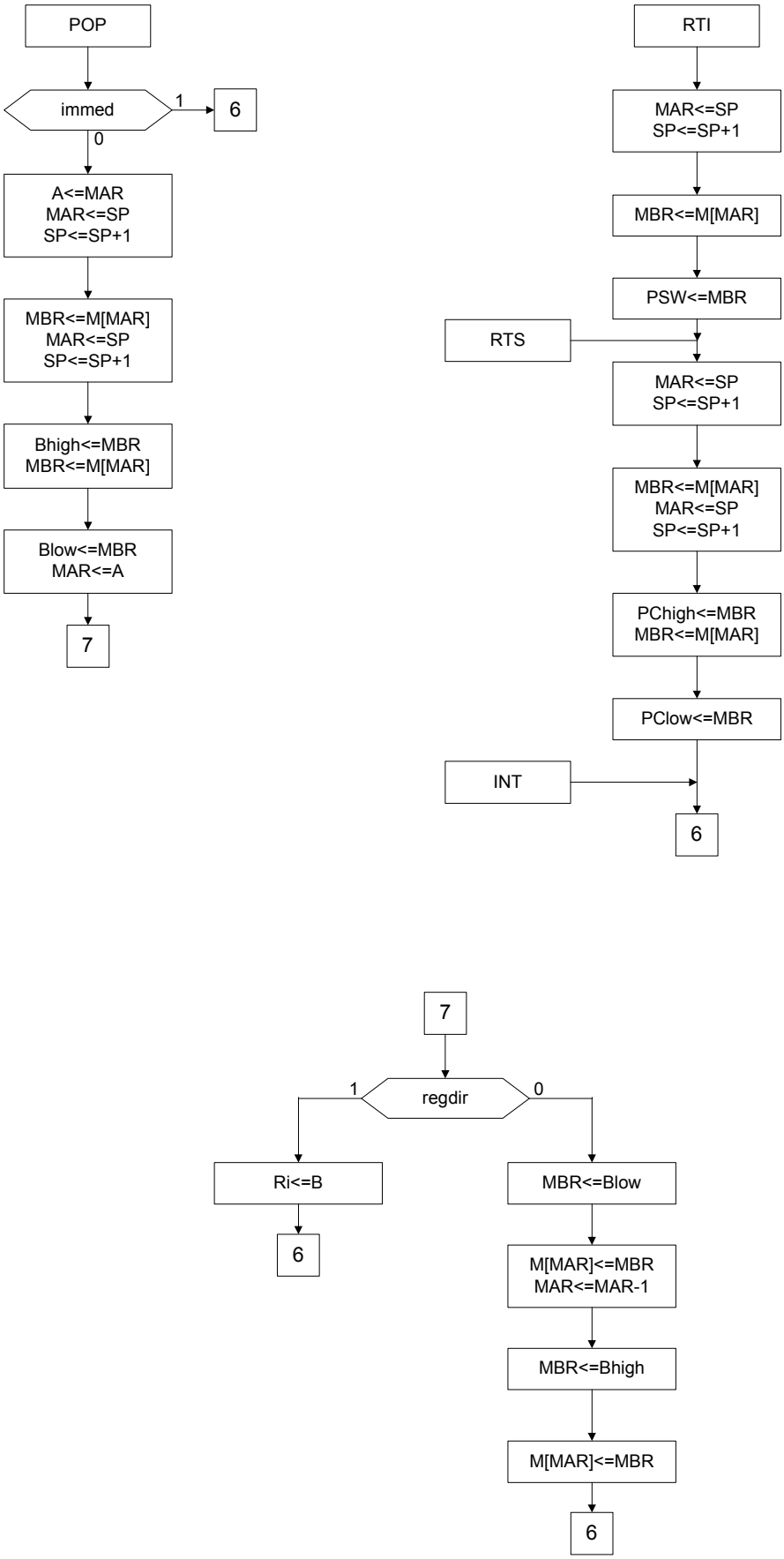

AND
Ri<=Ri ^ B
6


INC
immed
1
6
0
B<=B+1
7


DEC
immed
1
6
0
B<=B-1
7


ASR
immed
1
6
0
B<=asr B
7


MOVD
immed
1
6
0
B<=Ri
7



Slika 1.c Dijagram toka – faza izvršavanje operacija

Ukoliko je signal POP aktivan, skidanje sa steka se realizuje. U okviru toga se sadržaj sa steka upisuje u registar B. Najpre se proverava uslov immed. Ukoliko je on ispunjen prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d), jer u ovoj instrukciji neposredno adresiranje nije dozvoljeno. Ukoliko signal immed nije ispunjen sadržaj registra MAR prebacuje u registar A, sadržaj registra SP se prebacuje u registar MAR i sadržaj registra SP se inkrementira. Zatim se iz memorijske lokacije određene sadržajem registra MAR čita sadržaj i upisuje u registar MBR. Na kraju se sadržaj registra MBR upisuje u viši razred registara B. Na isti način se sa steka čita još jedna reč i upisuje u niži razred registara B. Na kraju se sadržaj registra A upisuje u registar MAR. 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 7 i fazu *vraćanje podatka* (slika 1.c).

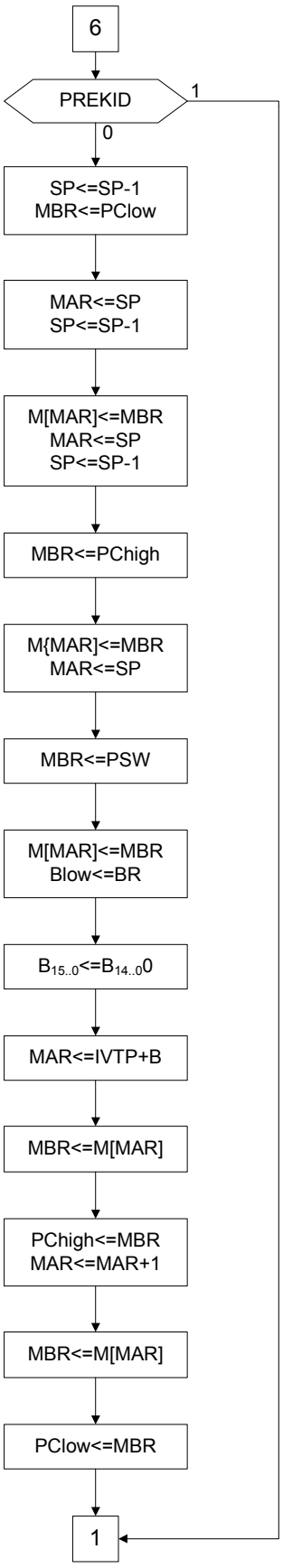
Ukoliko je signal PUSH aktivan, stavljanje na stek se realizuje. U okviru toga se sadržaj registra B stavlja na stek. Najpre se dekrementira sadržaj registra SP, prebacuje sadržaj nižeg razreda registara B u registar MBR. Potom se sadržaj registra SP prebacuje u registar MAR, i dekrementira se sadržaj registra SP. Zatim se sadržaj registra MBR upisuje u memorijsku lokaciju određenu sadržajem registra MAR. Na isti način se viši bajt registara B 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 dekrementira sadržaj registra SP i posle toga prebacuje u registar MAR. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

vraćanje podatka (slika 1.*c)*

Ukoliko je signal regdir aktivan, sadržaj registra B se prebacuje u jedan od registara registarskog fajla R_i određen vrednošću druge grupe bitova druge reči instrukcije iz registra IR2 i prelazi na korak 6 i fazu *opsluživanje prekida* (slika 1.d). Ukoliko je signal regdir neaktivan sadržaj registra B se upisuje u memoriju. Najpre se niži bajt registra B upisuje u registar MBR, a zatim se taj registar u sledećem koraku upisuje u memorijsku lokaciju određenu sadržajem registra MAR. Na isti način se viši bajt regitra B prebacuje u memoriju. Time je završena faza *izvršavanje operacija* i prelazi se na korak 6 i fazu *opsluživanje prekida* (slika 1.d).

opsluživanje prekida (slika 1.*d)*

Opsluživanje prekida se realizuje počev od koraka 6.



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

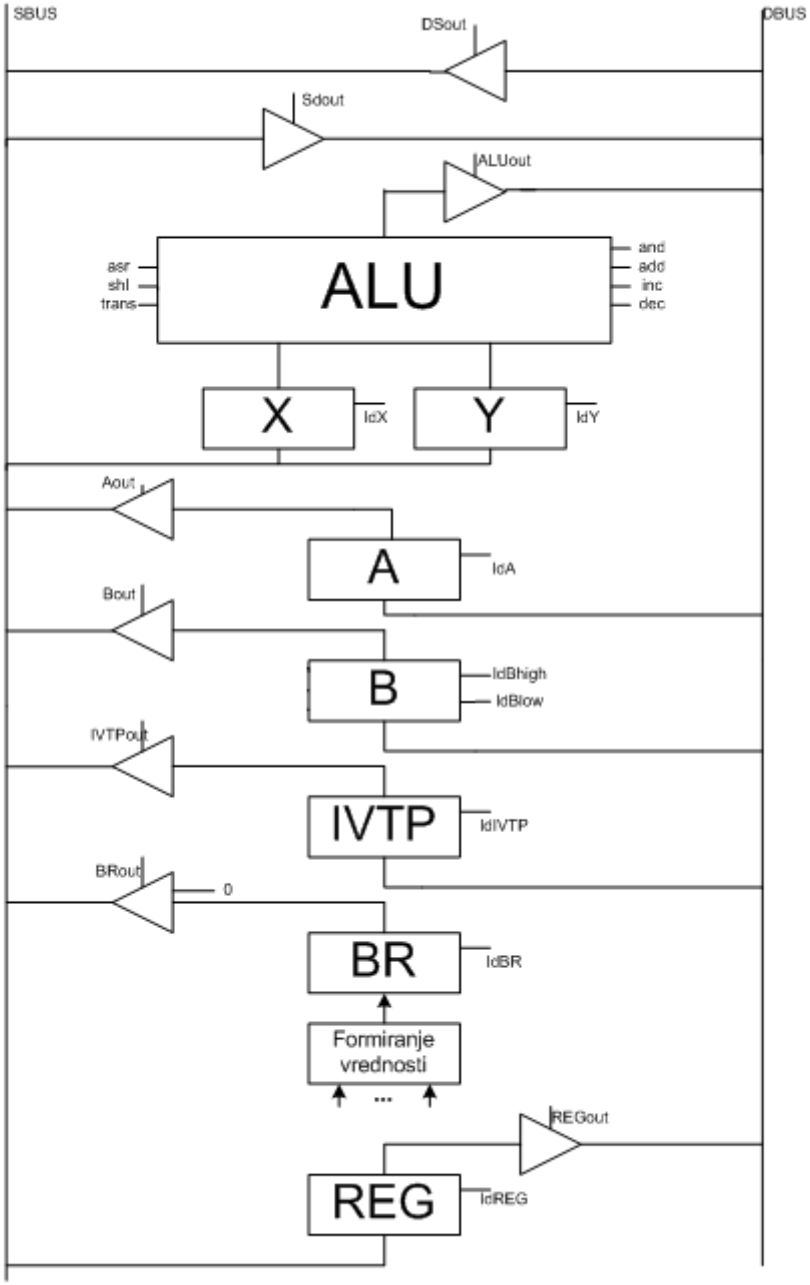
U ovom odeljku se razmatraju realizacije operacionih jedinica kod kojih su prekidačke mreže povezane direktno i pomoću jedne, dve i tri interne magistrale.

3.1 OPERACIONA JEDINICA SA DVE MAGISTRALE

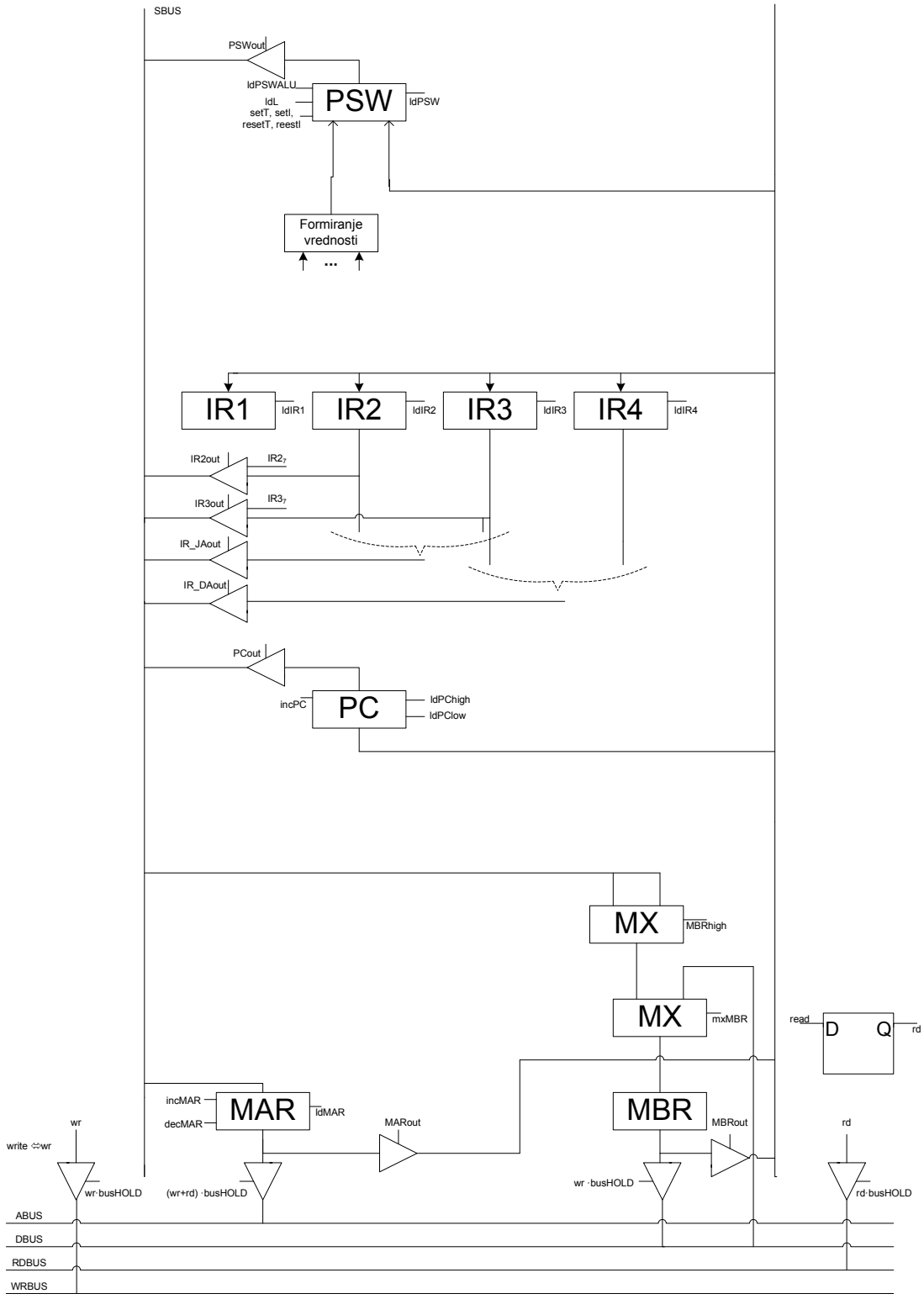
U ovom odeljku se razmatraju struktura upravljačke jedinice i algoritam generisanja upravljačkih signala.

3.1.1 Struktura operacione jedinice

Struktura operacione jedinice sa dve magistrale je data na slikama 3.a i 3.b.



Slika 3.a Operaciona jedinica sa dve magistrale



Slika 3.b Operaciona jedinica sa dve magistrale (nastavak)

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, prihvatni registri podatka A i B, registri opšte namene AX, BX, CX, DX, SI, DI, SP, BP (objedinjenih u registarski fajl REG), prihvatni registri podataka X i Y,

ukazivač na tabelu sa adresama prekidnih rutina IVTP, i 8-bitni registri: programska statusna reč PSW, prihvatni registar podatka memorije MBR, registar broja ulaza BR, registar maske IMR i četvorobajtni 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. U programski brojač se moze vršiti upis posebno nižeg, posebno višeg bajta.

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 adresnu magistralu, kao i na internu procesorsku magistralu DBUS, a u njega se vrši upis sa interne procesorske magistrale SBUS.

MBR je prihvatni registar podatka. U registar MBR se smešta sadržaj koji je pročitan iz memorijske lokacije ili sadržaj koji treba upisati na neku memorijsku lokaciju, a odabir se vrši multiplekserom na ulazu. Multiplekser služi za propustanje ili podataka sa sistemske magistrale DBUS, ili podataka sa interne procesorske magistrale SBUS, i to posebno nižeg, a posebno višeg bajta sa magistrale SBUS (to je realizovano pomoću još jednog multipleksera).Sadržaj registra MBR se vodi na sistemsku magistralu podataka DBUS ili na internu procesorsku magistralu DBUS.

A i **B** su prihvatni registri podataka. Oni imaju mogućnost upisa sa interne procesorske magistrale DBUS, a svoj sadržaj propuštaju na internu procesorsku magistralu SBUS.

PSW je registar koji predstavlja programsku statusnu reč. U njemu se na određenim pozicijama čuvaju indikatori statusa. Indikatori koje PSW sadrži su 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 da li je došlo do prenosa., kao i biti I, T, L_1 i L_0, da li su dozvoljeni prekidi, da li vršiti zaustavljenje nakon svake instrukcije i prioritet trenutno izvršavanog koda. U slučaju skoka na prekidnu rutinu njegova vrednost se čuva na steku jer ta vrednost zajedno sa vrednošću registra PC predstavlja trenutni kontekst rada procesora, i pri povratku prethodni kontekst mora biti restauriran. Stoga je moguće upisati u vrednost u sve razrede registra sa interne procesorske magistrale DBUS. Vrednost registra može biti 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 internu procesorsku magistralu SBUS.

IVTP registar čuva pokazivač na početak *Interupt Vector* tabele u kojoj se čuvaju adrese prekidnih rutina. Izlaz ovog registra se vodi internu procesorsku magistralu SBUS, sa koje će se dalje voditi na ulaz aritmeticko-logicke jedinice u kojoj će se u slučaju prekida, sabiranjem vrednosti ovog registra i broja ulaza, dobiti odgovarajuća adresa prekidne rutine. U ovaj registar moguće je upisati podatke sa interne procesorske magistrale DBUS.

IR je četvorobajtni prihvatni registar instrukcije. U sva četri bajta registra IR moguće je nezavisno smestiti vrednosti koje dolaze iz registra MBR preko interne procesorske magistrale DBUS, na taj način se iz memorije učita naredna instrukcija za izvršavanje. Binarnu reč u registru IR treba interpretirati saglasno formatu instrukcije. Registri IR1, IR2, IR3 i IR4 su 8-mo razredni registri koji formiraju razrede 32..24, 23…16, 15…8 i 7…0, respektivno, prihvatnog registra instrukcije $IR_{32...0}$. Instrukcije mogu, u zavisnosti od formata instrukcije, da budu dužine 1,2,3 ili 4 bajta. Razredi $IR_{32...24}$ 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 IR2 i IR3.

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 podatak 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. Ovaj registar čita podatke sa interne procesorske magistrale SBUS, a izbacuje podatke na internu procesorsku magistralu SBUS i nalazi se u okviru registarskog fajla REG.

BR je registar u koji se smešta broj ulaza u IV tabelu, na osnovu čega se određuje adresa prekidne rutine. Ulaz je određen fiksno, a za formiranje broja ulaza služi kombinaciona mreža označena kao "Formiranje vrednosti".

Detaljan opis operacione jedinice dat je u sekciji 5 PRILOZI

3.1.2 Algoritam generisanja upravljačkih signala

Sekvenca upravljačkih signala po koracima se formira na osnovu dijagrama toka izvršavanja instrukcije za operacionu jedinicu sa dve magistrale. 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 (tabela 1) 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 vrši resetovanje zahteva za prekid koji se ne prenose u sledeću instrukciju. U narednom koraku se startuje čitanje podatka iz memorije sa adrese upisane u registru MAR, nakon čega se podatak učitava u registar MBR i predstavlja prvi bajt instrukcije i inkrementira se registar PC,a zatim se u sledećem koraku taj podatak upisuje u IR1.U istom koraku se u slučaju potrebe čitanja sledeće reči instrukcije

adresa upisuje u MAR registrar. U naredna dva koraka vrši se provera uslova: **l1** i compl**2**. Ako je l1 neaktivan vrši se provera uslova compl2. Ukoliko je aktivan završava se čitanje instrukcije, a ako je compl2 neaktivan završeno je čitanje instrukcije i ide se na fazu izvršavanja instrukcije. Ukoliko je compl2 aktivan čita se još jedan bajt instrukcije na isti način kao i prvi bajt. Zatim se vrši provera uslova l3. Ukoliko je l3 aktivan završeno je čitanje instrukcije i ide se na fazu izvršavanja instrukcije. Ukoliko je l3 neaktivan, čita se još jedan reč instrukcije i ide se na fazu izvršavanja instrukcije.Ukoliko je vrednost signala l1 aktivna, čita se još jedan reč instrukcije. Zatim se proverava uslov l4. Ukoliko je on neaktivan završeno je čitanje instrukcije i ide se na fazu formiranje adrese i čitanje operanda, u suprotnom čita se još jedna reč instrukcije i proverava uslov l5. Ukoliko je l5 neaktivan završeno je čitanje instrukcije i ide se na fazu formiranje adrese i čitanje operanda, u suprotnom čita se još jedan bajt i prelazi se na fazu formiranje adrese i čitanje operanda !

$step_{00}$ **resetF, PCout**, **ldMAR**;
$step_{01}$ **read;**
$step_{02}$ **ldMBR**, **incPC**;
$step_{03}$ **MBRout**, **ldIR1**, **PCout**, **ldMAR**;
$step_{04}$ *br* (*if* **l1** *then* $step_{0E}$);
$step_{05}$ *br* (*if* compl**2** *then* $step_{3B}$);
$step_{06}$ **read;**
$step_{07}$ **ldMBR**, **incPC**;
$step_{08}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
$step_{09}$ *br* (*if* **l3** *then* $step_{3B}$);
$step_{0A}$ **read;**
$step_{0B}$ **ldMBR**, **incPC**;
$step_{0C}$ **MBRout**, **ldIR3**;
$step_{0D}$ *br* $step_{3B}$;

$step_{0E}$ **read;**
$step_{0F}$ **ldMBR**, **incPC**;
$step_{10}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
$step_{11}$ *br* (*if* compl**4** *then* $step_{19}$);
$step_{12}$ **read;**
$step_{13}$ **ldMBR**, **incPC**;
$step_{14}$ **MBRout**, **ldIR3**, **PCout**, **ldMAR**;
$step_{15}$ *br* (*if* **l5** *then* $step_{19}$);
$step_{16}$ **read;**
$step_{17}$ **ldMBR**, **incPC**;
$step_{18}$ **MBRout**, **ldIR4;**

! Formiranje adrese i čitanje operanda !

! U korak $step_{19}$ se dolazi iz koraka $step_{11}$, $step_{15}$ ili $step_{18}$ ukoliko se radi o instrukcijama dužine dva, tri ili četiri 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 koraku $step_{19}$ se realizuje višestruki uslovni skok na jedan od koraka $step_{1B}$, $step_{1D}$ … $step_{32}$ u zavisnosti od toga koji od signala adresiranja **regdir**, **regind**, **regindpom**, **memdir**, **memind**, **rel**, **immed** ima aktivnu vrednost, u slučaju da nijedan od navedenih signala nije aktivan došlo je do greške u adresiranju pa se u sledećem koraku skače na obradu prekida!

$step_{19}$ *br* (*case* (**regdir**, **regind**, **regindpom**, **memdir**, **memind**, **rel, immed**) *then*
(**regdir**, $step_{1B}$), (**regind**, $step_{1D}$), (**regindpom**, $step_{1F}$), (**memdir**, $step_{23}$),
(**memind**, $step_{25}$), (**rel**, $step_{2E}$), (**immed**, $step_{32}$));
$step_{1A}$ *br* $step_{9A}$;

Direktno registarsko !

! U korak $step_{1B}$ se dolazi iz $step_{19}$ ukoliko je signal za registarsko direktno adresiranje **regdir** aktivan. Prvo se operand učita u registar **B**, a zatim skače na fazu izvršavanja instrukcije !

$step_{1B}$ **REGout, ldBlow, ldBhigh, fdo**;
$step_{1C}$ *br* $step_{3B}$;

! Indirektno registarsko !

! U korak $step_{1D}$ se dolazi iz $step_{19}$ ukoliko je signal za registarsko indirektno adresiranje **regind** aktivan. Adresa operanda, koja se nalazi u nekom od registara registarskog fajla, se učita u registar **MAR** i skače se na fazu čitanje operanda za memorijska adresiranja!

$step_{1D}$ **REGout**,**DSout, ldMAR, fdo**;
$step_{1E}$ *br* $step_{34}$;

! Indirektno registarsko sa pomerajem !

! U korak $step_{1F}$ se dolazi iz $step_{19}$ ukoliko je signal za registarsko indirektno adresiranje sa pomerajem **regindpom** aktivan. Najpre se jedan od registara registarskog fajla upisuje u registar **X**, a zatim se pomeraj upisuje u registar **Y**. Potom se izvrši aritmetička operacija sabiranja, i rezultat upiše u registar MAR. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj registarsko indirektno adresiranje sa pomerajem !

$step_{1F}$ **REGout, ldX, DSout, fdo**;
$step_{20}$ **IR3out**, **ldY**;
$step_{21}$ **add**, **ldMAR, DSout, ALUout**;
$step_{22}$ *br* $step_{34}$;

! Direktno memorijsko !

! U korak $step_{23}$ se dolazi iz $step_{19}$ ukoliko je signal za memorijsko direktno adresiranje **memdir** aktivan. Adresa označena kao **IR_DA** se upisuje u registar **MAR**. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj memorijskog direktnog adresiranja !

$step_{23}$ **IR_DAout**, **ldMAR**;
$step_{24}$ *br* $step_{34}$;

! Indirektno memorijsko !

! U korak $step_{25}$ se dolazi iz $step_{19}$ ukoliko je signal za memorijsko indirektno adresiranje **memind** aktivan. Adresa označena kao **IR_DA** se upisuje u registar **MAR**. Zatim se sa memorijske lokacije određenom vrednošću registra **MAR** čita jedan bajt, koji se prebacuje u viših osam bitova registra B. Zatim se na isti način čita još jedna reč iz memorije i upisuje u nižih osam bitova registra B. Na kraju se sadržaj registra B prebacuje u regitar MAR. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj memorijskog direktnog adresiranja !

$step_{25}$ **IR_DAout**, **ldMAR**;
$step_{26}$ **read;**
$step_{27}$ **ldMBR, incMAR**;
$step_{28}$ **ldBhigh, MBRout**;
$step_{29}$ **read;**
$step_{2A}$ **ldMBR;**
$step_{2B}$ **ldBlow, MBRout**;
$step_{2C}$ **Bout**, **ldMAR**;
$step_{2D}$ *br* $step_{34}$;

! Relativno !

! U korak $step_{2E}$ se dolazi iz $step_{19}$ ukoliko je signal za relativno adresiranje **rel** aktivan. Najpre se programski projač **PC** upisuje u registar **X**, a zatim se pomeraj upisuje u registar **Y**. Potom se izvrši aritmetička operacija sabiranja, i rezultat upiše u registar MAR. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj relativnog adresiranja !

$step_{2E}$ **ldX, PCout**;
$step_{2F}$ **IR3out, ldY**;
$step_{30}$ **add**, **ALUout, DSout, ldMAR**;
$step_{31}$ *br* $step_{34}$;

! Neposredno !

! U korak $step_{32}$ se dolazi iz $step_{19}$ ukoliko je signal za neposredno adresiranje **immed** aktivan. Prvo se operand, označen kao **IR_DA** učita u registar **B**, a zatim se skače na fazu izvršavanja instrukcije !

step$_{32}$ **IR_DAout**, **SDout**, **ldBhigh, ldBlow**;
step$_{33}$ *br* step$_{3B}$;

! Čitanje operanda za memorijska adresiranja !
! Ako je signal MOVD ili POP aktivan ide se na korak *step*$_{3B}$ (faza izvršavanja instrukcije), u suprotnom se u registar MBR upisuje prvi bajt koji se nalazi na memorijskoj lokaciji određenoj sadržajem registra MAR i u istom taktu se inkrementira MAR registar da bi se pripremio za čitanje sledeceg bajta podatka. Zatim se pročitani podatak upisuju u viših osam bita registara B. Na isti način se čita još jedan podatak i upisuje u nižih osam bita registra B !

step$_{34}$ *br* (*if* **MOVD** or **POP** *then* step$_{3B}$);
step$_{35}$ **read;**
step$_{36}$ **ldMBR, incMAR**;
step$_{37}$ **MBRout**, **ldBhigh;**
step$_{38}$ **read;**
step$_{39}$ **ldMBR;**
step$_{3A}$ **MBRout**, **ldBlow**;

! Izvršavanje operacije !

! U korak *step*$_{3B}$ se dolazi iz koraka *step*$_{05}$, *step*$_{09}$,..., *step*$_{3A}$ radi izvršavanja operacije. U koraku *step*$_{3B}$ se realizuje višestruki uslovni skok na jedan od koraka *step*$_{3E}$, *step*$_{41}$, ..., *step*$_{90}$ u zavisnosti od toga koji od signala operacija **MOVS**, **MOVD**,..., **TRPD** ima aktivnu vrednost !

step$_{3B}$ *br* (*case* (**MOVS, MOVD, ADD**, **AND**, **ASR**, **BNZ**, **JSR**, **JMP**, **JMPIND, RTI**, **RTS, INT, PUSH, POP , INC, DEC, INTE, INTD, TRPE, TRPD**) *then*
(**MOVS,** step$_{3E}$), (**MOVD,** step$_{41}$), (**ADD**, step$_{45}$), (**AND**, step$_{49}$), (**ASR**, step$_{4D}$),
(**BNZ,** step$_{51}$), (**JSR**, step$_{56}$), (**JMP**, step$_{5C}$), (**JMPIND**, step$_{5E}$), (**RTI**, step$_{61}$),
(**RTS**, step$_{65}$), (**INT,** step$_{6D}$), (**PUSH**, step$_{6F}$), (**POP**, step$_{76}$), (**INC,** step$_{82}$),
(**DEC,** step$_{86}$), (**INTE,** step$_{8A}$), (**INTD,** step$_{8C}$), (**TRPE,** step$_{8E}$), (**TRPD,** step$_{90}$));

! Kod operacije !
! Ukoliko nije izabrana ni jedna od ovih instrukcija, to znači da je došlo do greške u kodu operacije, pa se stoga setuje flip-flop PRCOD, da bi smo znali prilikom obrade prekida da je došlo do greške u kodu operacije. Nakon toga se ide na korak *step*$_{9B}$ i fazu opsluživanja prekida !

step$_{3C}$ **setCOD**;
step$_{3D}$ *br* step$_{9B}$;

! MOVS !
! MOVS predstavlja instrukciju prenosa. Vrši se upis vrednosti iz registra B u jedan od registara registarskog fajla, ažuriraju se biti programske statusane reči **PSW** i skače se bezuslovno na korak step$_{9B}$ i fazu opsluživanja prekida !

step$_{3E}$ **Bout**, **daREG, ldREG, ldX**;
step$_{3F}$ **trans, ldPSWALU;**
step$_{40}$ *br* step$_{9B}$;

! MOVD !
! MOVD predstavlja instrukciju prenosa. U koraku step$_{41}$ se najpre vrši provera uslova immed, pošto ova instrukcija ne dozvoljava neposredno adresiranje. Ukoliko uslov nije ispinjen, vrši se upis vrednosti iz jednog od registara registarskog fajla u registar B, a zatim se ažuriraju biti programske statusane reči **PSW** i skače se bezuslovno na korak step$_{92}$ na fazu vraćanja podatka !

step$_{41}$ *br* (*if* **immed** *then* step$_{9A}$);
step$_{42}$ **REGout**, **daREG, ldBlow, ldBhigh, DSout, ldX**;
step$_{43}$ **trans, ldPSWALU**;
step$_{44}$ *br* step$_{92}$;

! ADD !

! ADD predstavlja aritmetičku operaciju sabiranja. Na ulaze aritmetičko-logičke jedinice se dovedu vrednosti jednog od registara registarskog fajla i registra B, koji se upisuju u registre X i Y respektivno. Zatim se izvrši operacija sabiranja, rezultat se upiše u jedan od registara registarskog fajla, i to onaj koji je bio prvi operand u instrukciji, i u istom taktu se ažuriraju biti N, Z, C i V registra PSW. Na kraju se bezuslovno skace na korak $step_{9B}$ na fazu opsluživanja prekida !

$step_{45}$ **REGout, daREG, DSout, ldX**;
$step_{46}$ **Bout**, **ldY**;
$step_{47}$ **add**, **ALUout, ldREG, daREG, ldPSWALU, DSout**;
$step_{48}$ *br* $step_{9B}$;

! AND !

! AND predstavlja operaciju logičko I. Na ulaze aritmetičko-logičke jedinice se dovedu vrednosti jednog od registara registarskog fajla i registra B, koji se upisuju u registre X i Y respektivno. Zatim se izvrši operacija bitsko I, rezultat se upiše u jedan od registara registarskog fajla, i to onaj koji je bio prvi operand u instrukciji i u istom taktu se ažuriraju biti N, Z, C i V registra PSW. Na kraju se bezuslovno skace na korak $step_{9B}$ na fazu opsluživanja prekida !

$step_{49}$ **REGout**, **daREG, DSout, ldX**;
$step_{4A}$ **Bout**, **ldY**;
$step_{4B}$ **and**, **ALUout, ldREG, daREG, ldPSWALU, DSout**;
$step_{4C}$ *br* $step_{9B}$;

! ASR !

! ASR predstavlja operaciju aritmetičkog pomeranja u desno. U koraku $step_{4D}$ se najpre vrši provera uslova immed, pošto ova instrukcija ne dozvoljava neposredno adresiranje. Ukoliko uslov nije ispinjen, vrši se prebacivanje vrednosti Iz registra B u registar X, zatim aritmetičko pomeranje u desno vrednosti iz registra X i ažuriraju se biti programske statusane reči **PSW,** i dobijena vrednost se upisuje u registar B i skače se bezuslovno na korak $step_{92}$ na fazu vraćanja podatka !

$step_{4D}$ *br* (*if* **immed** *then* $step_{9A}$);
$step_{4E}$ **Bout, ldX**;
$step_{4F}$ **asr, ALUout, ldBhigh, ldBlow, ldPSWALU**;
$step_{50}$ *br* $step_{92}$;

! BNZ !

! BNZ predstavlja operaciju branch if not zero (skoči ako rezultat nije nula, Z bit u registru PSW). U koraku $step_{51}$ se vrši provera bita Z i ako je njegova vrednost jedan ide se na korak $step_{9B}$, a u suprotnom se u PC upisuje trenutna vrednost PC-a povećana za osmobitni pomeraj, proširen znakom do šesnaestobitne vrednosti, koji se nalazi u registru IR2 i zatim se prelazi na korak $step_{9B}$ na fazu opsluživanja prekida !

$step_{51}$ *br* (*if* **Z** *then* $step_{9B}$);
$step_{52}$ **ldX, PCout**;
$step_{53}$ **ldY, IR2out**;
$step_{54}$ **add, ALUout**, **ldPChigh, ldPClow**;
$step_{55}$ *br* $step_{9B}$;

! JSR !

! Instrukcija JSR predstavlja skok u potprogram. Najpre se dekrementira vrednost registra SP. Podatak se smešta na stek tako što se u registar MAR upisuje vrednost stek pointer-a SP, a u registar MBR se smešta podatak, i na kraju se sadržaj registra MBR upisuje na memorijsku lokaciju određenu vrednošću registra MAR, odnosno na stek. Na stek se prvo smešta nižih osam bita registra PC, a zatim viših osam bita.!

$step_{56}$ **decSP, mxMBR**, **ldMBR, PCout**;
$step_{57}$ **upSPout**, **decSP, DSout**, **ldMAR**;
$step_{58}$ **write;**
$step_{59}$ **upSPout**, **DSout, ldMAR**;
$step_{5A}$ **MBRhigh, mxMBR**, **ldMBR, PCout**;
$step_{5B}$ **write;**

! JMP !

! JMP je operacija bezuslovnog skoka. U PC se upisuje vrednost, koja predstavlja adresu skoka, koja je data kao IR_JA**.** Na kraju se ide na korak $step_{9B}$ na fazu opsluživanja prekida !

step$_{5C}$ **IR_JAout**, **SDout**, **ldPChigh, ldPClow**;
step$_{5D}$ *br* step$_{9B}$;

! JMPIND !

! JMPIND je instrukcija bezuslovnog indirektnog skoka. U koraku step$_{5E}$ se najpre vrši proveravanje o kojem načinu adresiranja se radi. Ako je u pitanju neposredno ili registarsko direktno adresiranje skače se na korak step$_{9A}$ jer to predstavlja grešku u načinu adresiranja. Ukoliko nije specificirano nijedno od ovih adresiranja, upisuje se nova vrednost, koja se nalazi u registru B, u registar PC. Nakon toga se ide na korak step$_{9B}$ na fazu opsluživanja prekida !

step$_{5E}$ *br* (*if* **immed** or **regind** *then* step$_{9A}$);
step$_{5F}$ **Bout**, **SDout**, **ldPChigh, ldPClow**;
step$_{60}$ *br* step$_{9B}$;

! 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 upišemo u MAR sadržaj registra SP, a zatim i inkrementiramo SP, potom skidamo sa steka registar PSW. U nastavku se prelazi na instrukciju RTS !

step$_{61}$ **upSPout**, **DSout, ldMAR, incSP**;
step$_{62}$ **read;**
step$_{63}$ **ldMBR**;
step$_{64}$ **MBRout**, **ldPSW**;

! RTS !

! RTS predstavlja instrukciju povratka iz potprograma. Vrši se restauracija programskog brojača PC u suprotnom smeru u odnosu na smer kojim je stavljan na stek. Proces upisa stek pointera u registar MAR je identičan kao u prvom delu instrukcije RTI. Sada se pročitan podatak sa steka, koji predstavlja viših osam bita registra PC, iz registra MBR upisuje u viših osam bita registra PC. Na identičan način se sa steka skine nižih osam bita registra PC. Nakon toga se ide na korak step$_{9B}$ na fazu opsluživanja prekida !

step$_{65}$ **upSPout**, **DSout, ldMAR, incSP**;
step$_{66}$ **read;**
step$_{67}$ **ldMBR, upSPout**, **DSout, ldMAR incSP**;
step$_{68}$ **MBRout**, **ldPChigh**;
step$_{69}$ **read;**
step$_{6A}$ **ldMBR;**
step$_{6B}$ **MBRout**, **ldPClow;**
step$_{6C}$ *br* step$_{9B}$;

! INT !

! INT predstavlja instrukciju prekida. Kod ove instrukcije setujemo flip flpo PRINT I bezuslovno skačemo na korak step$_{9B}$, na fazu opsluživanja prekida !

step$_{6D}$ **setINT;**
step$_{6E}$ *br* step$_{9B}$;

! PUSH !

! Instrukcija PUSH izvršava operaciju stavljanja podatka na stek. Registar SP se najpre dekrementira, i niži bajt registra B upiše u registar MBR. Zatim se vrednost registra SP upiše u registar MAR, i sadržaj registra MBR upiše na memorijsku lokaciju određenu vrednošću registra MAR, odnosno na stek. Isti postupak se primeni i za viši bajt registra B. Nakon toga se ide na korak step$_{9B}$ na fazu opsluživanja prekida !

step$_{6F}$ **mxMBR**, **ldMBR, decSP, Bout**;
step$_{70}$ **ldMAR, upSPout, DSout, decSP**;
step$_{71}$ **write;**
step$_{72}$ **ldMAR, upSPout, DSout**;
step$_{73}$ **mxMBR, MBRhigh, Bout, ldMBR**;
step$_{74}$ **write;**
step$_{75}$ *br* step$_{9B}$;

! POP !

! Instrukcijom POP se skida podatak sa steka. Najpre sadržaj registra MAR sačuvamo u registu A. Posto stek ukazuje na poslednju zauzetu memorijsku lokaciju i raste ka nižim lokacijama prvo upišemo u MAR sadržaj registra SP, a zatim i inkrementiramo SP, potom skidamo sa steka podatak. Podatak se nalazi na memorijskoj lokaciji na koju ukazuje MAR, i njega upisujemo u MBR,a zatim u nižih osam bita registra B. Isto se uradi i za viši bajt podatka. Zatim se vrednost registra A upisuje u registar MAR. Na kraju se ažuriraju biti N, Z, C i V registra PSW. Nakon toga se ide na korak $step_{92}$ na fazu vraćanja podatka !

$step_{76}$ *br* (*if* **immed** *then* $step_{9A}$);
$step_{77}$ **MARout**, **ldA**;
$step_{78}$ **ldMAR, upSPout, DSout, incSP;**
$step_{79}$ **read;**
$step_{7A}$ **ldMBR, ldMAR, upSPout, DSout, incSP**;
$step_{7B}$ **MBRout, ldBhigh**;
$step_{7C}$ **read;**
$step_{7D}$ **ldMBR, ldMAR, Aout**;
$step_{7E}$ **MBRout, ldBlow;**
$step_{7F}$ **Bout, ldX;**
$step_{80}$ **trans, ldPSWALU**;
$step_{81}$ *br* $step_{92}$;

! INC !

! INC predstavlja instrukciju inkrementiranja. U koraku $step_{82}$ se najpre vrši provera uslova immed, pošto ova instrukcija ne dozvoljava neposredno adresiranje. Ukoliko uslov nije ispinjen, vrši se inkrementiranje vrednosti iz registra B i ažuriraju se biti programske statusane reči **PSW**, a zatim se skače bezuslovno na korak $step_{92}$ na fazu vraćanja podatka !

$step_{82}$ *br* (*if* **immed** *then* $step_{9A}$);
$step_{83}$ **Bout, ldX**;
$step_{84}$ **inc, ALUout, ldBhigh, ldBlow, ldPSWALU**;
$step_{85}$ *br* $step_{92}$;

! DEC !

! DEC predstavlja instrukciju dekrementiranja. U koraku $step_{86}$ se najpre vrši provera uslova immed, pošto ova instrukcija ne dozvoljava neposredno adresiranje. Ukoliko uslov nije ispinjen, vrši se dekrementiranje vrednosti iz registra B i ažuriraju se biti programske statusane reči **PSW**, a zatim se skače bezuslovno na korak $step_{92}$ na fazu vraćanja podatka !

$step_{86}$ *br* (*if* **immed** *then* $step_{9A}$);
$step_{87}$ **Bout, ldX**;
$step_{88}$ **dec, ALUout, ldBhigh, ldBlow, ldPSWALU**;
$step_{89}$ *br* $step_{92}$;

! INTE !

! INTE predstavlja instrukciju postavljanja indikatora I registra PSW. Ovde se samo postavlja bit I, i bezuslovno skače na korak $step_{9B}$ na fazu opsluživanja prekida !

$step_{8A}$ **setI**;
$step_{8B}$ *br* $step_{9B}$;

! INTD !

! INTD predstavlja instrukciju brisanja indikatora I registra PSW. Ovde se samo briše bit I, i bezuslovno skače na korak $step_{9B}$ na fazu opsluživanja prekida !

$step_{8C}$ **resetI**;
$step_{8D}$ *br* $step_{9B}$;

! TRPE !

! TRPE predstavlja instrukciju postavljanja indikatora T registra PSW. Ovde se samo postavlja bit T, i bezuslovno skače na korak $step_{9B}$ na fazu opsluživanja prekida !

$step_{8E}$ **setT**;
$step_{8F}$ *br* $step_{9B}$;

! TRPD !

! TRPD predstavlja instrukciju brisanja indikatora T registra PSW. Ovde se samo briše bit T, i bezuslovno skače na korak $step_{9B}$ na fazu opsluživanja prekida !

$step_{90}$ **resetT**;
$step_{91}$ *br* $step_{9B}$;

! Vraćanje podatka !

! U fazi vraćanja podatka najpre ispitujemo da li je registarsko direktno adresiranje u pitanju. Ukoliko nije, to znači da podatak moramo vratiti u memoriju.S obzirom na to, najpre nižih osam bita registra B upisujemo u registar MBR, zatim to upisujemo na memorijsku lokaciju određenu sadržajem registra MAR, i u istom taktu dekrementiramo sadržaj registra MAR. Isti postupak ponavljamo za viših osam bita registra B, samo sada ne dekrementiramo registar MAR na kraju. Ukoliko jeste registarsko direktno adresiranje onda samo vrednost registra B upisemo na odgovarajuće mesto u registrarskom bloku. Time je faza izvršavanje operacije je završena, podatak nam se vec nalazi u registru B, i prelazimo na korak $step_{9B}$ na fazu opsluživanja prekida!

$step_{92}$ *br* (*if* **regdir** *then* $step_{98}$);
$step_{93}$ **Bout**, **ldMBR, mxMBR**;
$step_{94}$ **write;**
$step_{95}$ **decMAR, Bout**, **ldMBR, mxMBR, MBRhigh**;
$step_{96}$ **write;**
$step_{97}$ *br* $step_{9B}$;
$step_{98}$ **Bout**, **ldREG, fvo**;
$step_{99}$ *br* $step_{9B}$;

! Opsluživanje prekida !

! U korak $step_{9B}$ se dolazi nakon svake izvršene instrukcije ili ukoliko je došlo do neke greške ili u kodu operacije, ili prilikom adresiranja. U zavisnosti od toga da li je signal **PREKID** neaktivan ili aktivan, ili završava izvršavanje tekuće instrukcije i prelaskom na korak $step_{00}$ počinje faza čitanje sledeće instrukcije ili se produžava izvršavanje tekuće instrukcije prelaskom na korak $step_{9C}$ produžava faza opsluživanje prekida tekuće instrukcije !

! 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 $step_{9B}$ do $step_{AF}$. Na stek se stavlja prvo niži, a zatim i viši bajt registra $PC_{15...0}$. Stoga se najpre u koraku $step_{9C}$ signalom **decSP** vrši dekrementiranje registra $SP_{15...0}$, a zatim se vrši upis nižih osam bita registra PC u registar $MBR_{7...0}$. Potom se sadržaj registra SP prebacuje u registar MAR, i na kraju se podatak iz registra MBR upisuje na memorijsku lokaciju određenu sadržajem registra MAR, odnosno na stek. Na isti način se na stek stavljaju i viši bajt registra PC, kao i registar PSW !

! Utvrđivanje broja ulaza i utvrđivanje adrese prekidne rutine !

! Počev od koraka $step_{A5}$ utvrđuje se broj ulaza u tabelu prekidnih rutina i adresa prekidne rutine. Najpre se sadržaj registra BR prebacuje u registar X. Sadržaj registra X se šiftuje u levo za jedno mesto, a potom se u registar MAR upisuje vrednost dobijena sabiranjem registra IVTP i registra X. U koracima koji slede se čitaju viši i niži bajt registra PC, sa memorijkse lokacije određene sadržajem registra MAR. Time je završena operacija opsluživanje prekida, i bezuslovno se prelazi na sledeću instrukciju!

$step_{9A}$ **setADR**;

$step_{9B}$ *br* (*if* $\overline{\text{PREKID}}$ *then* $step_{00}$);
$step_{9C}$ **decSP, PCout**, **mxMBR**, **ldMBR, ldBR, intack**;

$step_{9D}$ **upSPout**, **DSout, ldMAR, decSP**;
$step_{9E}$ **write;**
$step_{9F}$ **upSPout**, **DSout, ldMAR**, **decSP**;
$step_{A0}$ **PCout**, **mxMBR**, **ldMBR, MBRhigh**;
$step_{A1}$ **write;**
$step_{A2}$ **upSPout**, **DSout, ldMAR**;
$step_{A3}$ **PSWout**, **mxMBR**, **ldMBR**;
$step_{A4}$ **write;**
$step_{A5}$ **BRout**, **ldX**
$step_{A6}$ **shl, ALUout, DSout, ldX;**
$step_{A7}$ **IVTPout**, **ldY**;
$step_{A8}$ **add**, **ALUout, ldMAR, DSout ;**
$step_{A9}$ **read;**
$step_{AA}$ **ldMBR**;
$step_{AB}$ **MBRout**, **ldPChigh, incMAR**;
$step_{AC}$ **read;**
$step_{AD}$ **ldMBR**;
$step_{AE}$ **MBRout**, **ldPClow**;
$step_{AF}$ *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.

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. Na primer, u koraku $step_{03}$ 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 $step_{04}$ vrši provera signala **l1** i u zavisnosi od njegove vrednosti prelazi ili na korak $step_{0E}$ ili na korak $step_{05}$. Zbog toga koraci $step_{03}$ i $step_{04}$ ne mogu da se spoje. Takođe, u koraku $step_{08}$ se signalom **ldIR2** druga reč instrukcije upisuje u registar IR a na osnovu nje IR1 se formira vrednost signala logičkog uslova **l3**, dok se u koraku $step_{09}$ vrši provera signala **l3** i u zavisnosi od njegove vrednosti prelazi ili na korak $step_{10}$ ili na korak $step_{3B}$. Zbog toga koraci $step_{08}$ i $step_{09}$ mogu da se spoje.

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 $step_{18}$ **MBRout**, **ldIR4** i $step_{19}$ *br* (*case* (**regdir**, **regind**, **regindpom**, **memdir**, **memind**, **rel , immed**) *then* …), kao i koraci $step_{3A}$ **MBRout**, **ldBlow** i $step_{3B}$ *br* (*case* (**MOVS, MOVD, ADD**, **AND**, **ASR**, **BNZ**, **JSR**, **JMP**, **JMPIND, RTI**, **RTS, INT, PUSH, POP , INC, DEC, INTE, INTD, TRPE, TRPD**) *then* …). Koraci $step_{18}$ **MBRout**, **ldIR4** i $step_{19}$ *br*

(*case* (**regdir**, **regind**, **regindpom**, **memdir**, **memind**, **rel , immed**) *then* …) se ne mogu spojiti iako su signali logičkih uslova **regdir**, **regind**, ..., **immed**, koji se konsultuju u koraku $step_{13}$, formirani dosta ranije kada je, signalom **ldIR2**, druga reč instrukcije, koja sadrži polje način adresirana, upisana u registar IR i ne zavisi od mikrooperacija izvršenih na osnovu upravljačkih signala generisanih u koraku $step_{18}$, jer se na korak $step_{19}$ prelazi ne samo iz koraka $step_{18}$ već i iz drugih koraka. Takođe se koraci $step_{3A}$ **MBRout**, **ldBlow** i $step_{3B}$ *br* (*case* (**MOVS, MOVD, ADD**, **AND**, **ASR**, **BNZ**, **JSR**, **JMP**, **JMPIND, RTI**, **RTS, INT, PUSH, POP , INC, DEC, INTE, INTD, TRPE, TRPD**) *then* …) se ne mogu spojiti iako su signali logičkih uslova **MOVS**, **MOVD**, ..., **TRPD**, koji se konsultuju u koraku $step_{3B}$, formirani dosta ranije 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 $step_{3A}$, jer se na korak $step_{3B}$ prelazi ne samo iz koraka $step_{3A}$ već i iz drugih koraka.

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 npr. mogle da se jave u koracima $step_{01}$ **read** i $step_{58}$ **write** itd. ukoliko bi upis u neku memorijsku lokaciju, iniciran generisanjem signala **read** ili **write**, trajao više od jedne periode signala takta. Zbog toga se ovi koraci ne 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

! Čitanje instrukcije !

$step_{00}$ **resetF, PCout**, **ldMAR**;
$step_{01}$ **read;**
$step_{02}$ **ldMBR**, **incPC**;
$step_{03}$ **MBRout**, **ldIR1**, **PCout**, **ldMAR;**
$step_{04}$ *br* (*if* **l1** *then* $step_{0C}$);
$step_{05}$ *br* (*if* comp**l2** *then* $step_{31}$);
$step_{06}$ **read;**
$step_{07}$ **ldMBR**, **incPC**;
$step_{08}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR,** *br* (*if* **l3** *then* $step_{31}$);
$step_{09}$ **read;**
$step_{0A}$ **ldMBR**, **incPC**;
$step_{0B}$ **MBRout**, **ldIR3,** *br* $step_{31;}$

$step_{0C}$ **read;**
$step_{0D}$ **ldMBR**, **incPC**;
$step_{0E}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
$step_{0F}$ *br* (*if* comp**l4** *then* $step_{16}$);
$step_{10}$ **read;**
$step_{11}$ **ldMBR**, **incPC**;
$step_{12}$ **MBRout**, **ldIR3**, **PCout**, **ldMAR,** *br* (*if* **l5** *then* $step_{16}$);
$step_{13}$ **read;**

step$_{14}$ **ldMBR**, **incPC**;
step$_{15}$ **MBRout**, **ldIR4;**

! Formiranje adrese i čitanje operanda !

step$_{16}$ *br* (*case* (**regdir**, **regind**, **regindpom**, **memdir**, **memind**, **rel, immed**) *then* (**regdir**, step$_{18}$), (**regind**, step$_{19}$), (**regindpom**, step$_{1A}$), (**memdir**, step$_{1D}$), (**memind**, step$_{1E}$), (**rel**, step$_{26}$), (**immed**, step$_{29}$));
step$_{17}$ *br* step$_{7D}$;

! Direktno registarsko !

step$_{18}$ **REGout, ldBlow, ldBhigh, fdo,** *br* step$_{31}$;

! Indirektno registarsko !

step$_{19}$ **REGout**,**DSout, ldMAR, fdo,** *br* step$_{2A}$;

! Indirektno registarsko sa pomerajem !

step$_{1A}$ **REGout, ldX, DSout, fdo**;
step$_{1B}$ **IR3out**, **ldY**;
step$_{1C}$ **add**, **ldMAR, DSout, ALUout,** *br* step$_{2A}$;

! Direktno memorijsko !

step$_{1D}$ **IR_DAout**, **ldMAR,** *br* step$_{2A}$;

! Indirektno memorijsko !

step$_{1E}$ **IR_DAout**, **ldMAR**;
step$_{1F}$ **read;**
step$_{20}$ **ldMBR, incMAR**;
step$_{21}$ **ldBhigh, MBRout**;
step$_{22}$ **read;**
step$_{23}$ **ldMBR;**
step$_{24}$ **ldBlow, MBRout**;
step$_{25}$ **Bout**, **ldMAR,** *br* step$_{2A}$;

! Relativno !

step$_{26}$ **ldX, PCout**;
step$_{27}$ **IR3out, ldY**;
step$_{28}$ **add**, **ALUout, DSout, ldMAR,** *br* step$_{2A}$;

! Neposredno !

step$_{29}$ **IR_DAout**, **SDout**, **ldBhigh, ldBlow,** *br* step$_{31}$;

! Čitanje operanda za memorijska adresiranja !

step$_{2A}$ *br* (*if* **MOVD** or **POP** *then* step$_{31}$);
step$_{2B}$ **read;**
step$_{2C}$ **ldMBR, incMAR**;
step$_{2D}$ **MBRout**, **ldBhigh;**
step$_{2E}$ **read;**
step$_{2F}$ **ldMBR;**
step$_{30}$ **MBRout**, **ldBlow**;

! Izvršavanje operacije !

step$_{31}$ *br* (*case* (**MOVS, MOVD, ADD**, **AND**, **ASR**, **BNZ**, **JSR**, **JMP**, **JMPIND, RTI**, **RTS, INT, PUSH, POP , INC, DEC, INTE, INTD, TRPE, TRPD**) *then* (**MOVS,** step$_{33}$), (**MOVD,** step$_{35}$), (**ADD**, step$_{38}$), (**AND**, step$_{3B}$), (**ASR**, step$_{3E}$), (**BNZ,** step$_{41}$), (**JSR**, step$_{45}$), (**JMP**, step$_{4B}$), (**JMPIND**, step$_{4C}$), (**RTI**, step$_{4E}$), (**RTS**, step$_{52}$), (**INT,** step$_{59}$), (**PUSH**, step$_{5A}$), (**POP**, step$_{61}$), (**INC,** step$_{6C}$), (**DEC,** step$_{6F}$), (**INTE,** step$_{72}$), (**INTD,** step$_{73}$), (**TRPE,** step$_{74}$), (**TRPD,** step$_{75}$));

! Kod operacije !

step$_{32}$ **setCOD**, *br* step$_{7E}$;

! MOVS !

step$_{33}$ **Bout**, **daREG, ldREG, ldX**;

step$_{34}$ **trans, ldPSWALU,** *br* step$_{7E}$;

! MOVD !

step$_{35}$ *br* (*if* **immed** *then* step$_{7D}$);

step$_{36}$ **REGout**, **daREG, ldBlow, ldBhigh, DSout, ldX**;

step$_{37}$ **trans, ldPSWALU,** *br* step$_{76}$;

! ADD !

step$_{38}$ **REGout, daREG, DSout, ldX**;

step$_{39}$ **Bout**, **ldY**;

step$_{3A}$ **add**, **ALUout, ldREG, daREG, ldPSWALU,** *br* step$_{7E}$;

! AND !

step$_{3B}$ **REGout**, **daREG, DSout, ldX**;

step$_{3C}$ **Bout**, **ldY**;

step$_{3D}$ **and**, **ALUout, ldREG, daREG, ldPSWALU,** *br* step$_{7E}$;

! ASR !

step$_{3E}$ *br* (*if* **immed** *then* step$_{7D}$);

step$_{3F}$ **Bout, ldX**;

step$_{40}$ **asr, ALUout, ldBhigh, ldBlow, ldPSWALU,** *br* step$_{76}$;

! BNZ !

step$_{41}$ *br* (*if* **Z** *then* step$_{7E}$);

step$_{42}$ **ldX, PCout**;

step$_{43}$ **ldY, IR2out**;

step$_{44}$ **add, ALUout**, **ldPChigh, ldPClow,** *br* step$_{7E}$;

! JSR !

step$_{45}$ **decSP, mxMBR**, **ldMBR, PCout**;

step$_{46}$ **upSPout**, **decSP, DSout**, **ldMAR**;

step$_{47}$ **write;**

step$_{48}$ **upSPout**, **DSout, ldMAR**;

step$_{49}$ **MBRhigh, mxMBR**, **ldMBR, PCout**;

step$_{4A}$ **write;**

! JMP !

step$_{4B}$ **IR_JAout**, **SDout**, **ldPChigh, ldPClow,** *br* step$_{7E}$;

! JMPIND !

step$_{4C}$ *br* (*if* **immed** or **regind** *then* step$_{7D}$);

step$_{4D}$ **Bout**, **SDout**, **ldPChigh, ldPClow,** *br* step$_{7E}$;

! RTI !

step$_{4E}$ **upSPout**, **DSout, ldMAR, incSP**;

step$_{4F}$ **read;**

step$_{50}$ **ldMBR**;

step$_{51}$ **MBRout**, **ldPSW**;

! RTS !

step$_{52}$ **upSPout**, **DSout, ldMAR, incSP**;

step$_{53}$ **read;**

step$_{54}$ **ldMBR, upSPout**, **DSout, ldMAR incSP**;

step$_{55}$ **MBRout**, **ldPChigh**;

step$_{56}$ **read;**

step$_{57}$ **ldMBR;**

step$_{58}$ **MBRout**, **ldPClow,** *br* step$_{7E}$;

! INT !

step$_{59}$ **setINT,** *br* step$_{7E}$**;**

! PUSH !

step$_{5A}$ **mxMBR**, **ldMBR, decSP, Bout**;

step$_{5B}$ **ldMAR, upSPout, DSout, decSP**;

step$_{5C}$ **write;**

step$_{5D}$ **ldMAR, upSPout, DSout**;
step$_{5E}$ **mxMBR, MBRhigh, Bout, ldMBR**;
step$_{5F}$ **write;**
step$_{60}$ *br* step$_{7E}$;

! POP !

step$_{61}$ *br* (*if* **immed** *then* step$_{7D}$);
step$_{62}$ **MARout**, **ldA**;
step$_{63}$ **ldMAR, upSPout, DSout, incSP;**
step$_{64}$ **read;**
step$_{65}$ **ldMBR, ldMAR, upSPout, DSout, incSP**;
step$_{66}$ **MBRout, ldBhigh**;
step$_{67}$ **read;**
step$_{68}$ **ldMBR, ldMAR, Aout**;
step$_{69}$ **MBRout, ldBlow;**
step$_{6A}$ **Bout, ldX;**
step$_{6B}$ **trans, ldPSWALU,** *br* step$_{76}$;

! INC !

step$_{6C}$ *br* (*if* **immed** *then* step$_{7D}$);
step$_{6D}$ **Bout, ldX**;
step$_{6E}$ **inc, ALUout, ldBhigh, ldBlow, ldPSWALU,** *br* step$_{76}$;

! DEC !

step$_{6F}$ *br* (*if* **immed** *then* step$_{7D}$);
step$_{70}$ **Bout, ldX**;
step$_{71}$ **dec, ALUout, ldBhigh, ldBlow, ldPSWALU,** *br* step$_{76}$;

! INTE !

step$_{72}$ **setI,** *br* step$_{7E}$;

! INTD !

step$_{73}$ **resetI,** *br* step$_{7E}$;

! TRPE !

step$_{74}$ **setT,** *br* step$_{7E}$;

! TRPD !

step$_{75}$ **resetT,** *br* step$_{7E}$;

! Vraćanje podatka !

step$_{76}$ *br* (*if* **regdir** *then* step$_{7C}$);
step$_{77}$ **Bout**, **ldMBR, mxMBR**;
step$_{78}$ **write;**
step$_{79}$ **decMAR Bout**, **ldMBR, mxMBR, MBRhigh**;
step$_{7A}$ **write;**
step$_{7B}$ *br* step$_{7E}$;
step$_{7C}$ **Bout**, **ldREG, fvo,** br step$_{7E}$;

! Opsluživanje prekida !

step$_{7D}$ **setADR**;

step$_{7E}$ *br* (*if* $\overline{\text{PREKID}}$ *then* step$_{00}$);
step$_{7F}$ **decSP, PCout**, **mxMBR**, **ldMBR, ldBR, intack**;
step$_{80}$ **upSPout**, **DSout, ldMAR, decSP**;
step$_{81}$ **write;**
step$_{82}$ **upSPout**, **DSout, ldMAR**, **decSP**;
step$_{83}$ **PCout**, **mxMBR**, **ldMBR, MBRhigh**;
step$_{84}$ **write;**
step$_{85}$ **upSPout**, **DSout, ldMAR**;
step$_{86}$ **PSWout**, **mxMBR**, **ldMBR**;
step$_{87}$ **write;**

step$_{88}$ **BRout**, **ldX**
step$_{89}$ **shl, ALUout, DSout, ldX;**
step$_{8A}$ **IVTPout**, **ldY**;
step$_{8B}$ **add**, **ALUout, ldMAR, DSout ;**
step$_{8C}$ **read;**
step$_{8D}$ **ldMBR**;
step$_{8E}$ **MBRout**, **ldPChigh, incMAR**;
step$_{8F}$ **read;**
step$_{90}$ **ldMBR**;
step$_{91}$ **MBRout**, **ldPClow,** *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 ovom odeljku se razmatraju tehnike ožičene i mikroprogramske realizacije upravljačke jedinice i to za slučaj operacione jedinice sa direktnim vezama. Korišćenje ovih tehnika za operacione jedinice sa jednom, dve i tri magistrale je isto kao i za slučaj operacione jedinice sa direktnim vezama.

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 9.

Tabela 9 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
$step_{34}$	$\mathbf{val_{34}}$	34
$step_{3B}$	$\mathbf{val_{3B}}$	3B
$step_{92}$	$\mathbf{val_{92}}$	92
$step_{9A}$	$\mathbf{val_{9A}}$	9A
$step_{9B}$	$\mathbf{val_{9B}}$	9B

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 1. 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 2.

Tabela 1 Signali uslovnih skokova i signali uslova

signal uslovnog skoka	signal uslova
brl1	**l1**
brl2	compl**2**
brl3	**l3**
brl4	compl**4**
brl5	**l5**
brMOVD_POP	**MOVD or POP**
brimmed	**immed**
brregdir	**regdir**
brimm_regdir	**immed or regdir**
brZ	**Z**
brnotPREKID	$\overline{\text{PREKID}}$

Tabela 2 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
$step_{19}$	$\mathbf{val_{19}}$	19

$step_{3B}$	**val_{3B}**	3B
$step_{9A}$	**val_{9A}**	9A
$step_{9B}$	**val_{9B}**	9B
$step_{98}$	**val_{98}**	98

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 3. 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 4 i 5.

Tabela 3 Signali višestrukih uslovnih skokova

Korak	signal višestrukog uslovnog skoka
$step_{19}$	**bradr**
$step_{3B}$	**bropr**

Tabela 4 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{0F}$

signal uslova	vrednost
regdir	1B
indreg	1D
indregpom	1F
dirmem	23
indmem	25
rel	2E
immed	32

Tabela 5 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{31}$

signal uslova	vrednost	signal uslova	vrednost
MOVS	3E	**RTS**	65
MOVD	41	**INT**	6D
ADD	45	**PUSH**	6F
AND	49	**POP**	76
ASR	4D	**INC**	82
BNZ	51	**DEC**	86
JSR	56	**INTE**	8A
JMP	5C	**INTD**	8C
JMPIND	5E	**TRPE**	8E
RTI	61	**TRPD**	90

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela tabela 1), formirana sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 6). 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 1) i višestrukih uslovnih

skokova (tabela 3) i signali **val$_A$** za bezuslovne (tabela 9) i uslovne (tabela 2) 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 6 Sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije bez spajanja koraka

! Čitanje instrukcije !

T_{00} **resetF, PCout**, **ldMAR**;
T_{01} **read;**
T_{02} **ldMBR**, **incPC**;
T_{03} **MBRout**, **ldIR1**, **PCout**, **ldMAR**;
T_{04} **brl1, val$_{0E}$**;
T_{05} **brl2, val$_{3B}$**;
T_{06} **read;**
T_{07} **ldMBR**, **incPC**;
T_{08} **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
T_{09} **brl3, val$_{3B}$**;
T_{0A} **read;**
T_{0B} **ldMBR**, **incPC**;
T_{0C} **MBRout**, **ldIR3**;
T_{0D} **bruncnd, val$_{3B}$**;

T_{0E} **read;**
T_{0F} **ldMBR**, **incPC**;
T_{10} **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
T_{11} **brl4, val$_{19}$**;
T_{12} **read;**
T_{13} **ldMBR**, **incPC**;
T_{14} **MBRout**, **ldIR3**, **PCout**, **ldMAR**;
T_{15} **brl5, val$_{19}$**;
T_{16} **read;**
T_{17} **ldMBR**, **incPC**;
T_{18} **MBRout**, **ldIR4;**

! Formiranje adrese i čitanje operanda !

T_{19} **bradr**;
T_{1A} **bruncnd, val$_{9A}$**;

! Direktno registarsko !
T_{1B} **REGout, ldBlow, ldBhigh, fdo**;
T_{1C} **bruncnd, val$_{3B}$**;
! Indirektno registarsko !
T_{1D} **REGout**,**DSout, ldMAR, fdo**;
T_{1E} **bruncnd, val$_{34}$**;
! Indirektno registarsko sa pomerajem !
T_{1F} **REGout, ldX, DSout, fdo**;
T_{20} **IR3out**, **ldY**;
T_{21} **add**, **ldMAR, DSout, ALUout**;
T_{22} **bruncnd, val$_{34}$**;
! Direktno memorijsko !
T_{23} **IR_DAout**, **ldMAR**;
T_{24} **bruncnd, val$_{34}$**;
! Indirektno memorijsko !
T_{25} **IR_DAout**, **ldMAR**;

T_{26} **read;**
T_{27} **ldMBR, incMAR**;
T_{28} **ldBhigh, MBRout**;
T_{29} **read;**
T_{2A} **ldMBR;**
T_{2B} **ldBlow, MBRout**;
T_{2C} **Bout**, **ldMAR**;
T_{2D} **bruncnd, val$_{34}$**;

! Relativno !

T_{2E} **ldX, PCout**;
T_{2F} **IR3out, ldY**;
T_{30} **add**, **ALUout, DSout, ldMAR**;
T_{31} **bruncnd, val$_{34}$**;

! Neposredno !

T_{32} **IR_DAout**, **SDout**, **ldBhigh, ldBlow**;
T_{33} **bruncnd, val$_{3B}$**;

! Čitanje operanda za memorijska adresiranja !

T_{34} **brMOVD_POP, val$_{3B}$**;
T_{35} **read;**
T_{36} **ldMBR, incMAR**;
T_{37} **MBRout**, **ldBhigh;**
T_{38} **read;**
T_{39} **ldMBR;**
T_{3A} **MBRout**, **ldBlow**;

! Izvršavanje operacije !

T_{3B} **bropr**;

! Kod operacije !

T_{3C} **setCOD**;
T_{3D} **bruncnd, val$_{9B}$**;

! MOVS !

T_{3E} **Bout**, **daREG, ldREG, ldX**;
T_{3F} **trans, ldPSWALU;**
T_{40} **bruncnd, val$_{9B}$**;

! MOVD !

T_{41} **brimmed, val$_{9A}$**;
T_{42} **REGout**, **daREG, ldBlow, ldBhigh, DSout, ldX**;
T_{43} **trans, ldPSWALU**;
T_{44} **bruncnd, val$_{92}$**;

! ADD !

T_{45} **REGout, daREG, DSout, ldX**;
T_{46} **Bout**, **ldY**;
T_{47} **add**, **ALUout, ldREG, daREG, ldPSWALU, DSout**;
T_{48} **bruncnd, val$_{9B}$**;

! AND !

T_{49} **REGout**, **daREG, DSout, ldX,**;
T_{4A} **Bout**, **ldY**;
T_{4B} **and**, **ALUout, ldREG, daREG, ldPSWALU, DSout**;
T_{4C} **bruncnd, val$_{9B}$**;

! ASR !

T_{4D} **brimmed, val$_{9A}$**;
T_{4E} **Bout, ldX**;

T_{4F} **asr, ALUout, ldBhigh, ldBlow, ldPSWALU**;
T_{50} **bruncnd, val$_{92}$**;

! BNZ !

T_{51} **brZ, val$_{9B}$**;
T_{52} **ldX, PCout**;
T_{53} **ldY, IR2out**;
T_{54} **add, ALUout**, **ldPChigh, ldPClow**;
T_{55} **bruncnd, val$_{9B}$**;

! JSR !

T_{56} **decSP, mxMBR**, **ldMBR, PCout**;
T_{57} **upSPout**, **decSP, DSout**, **ldMAR**;
T_{58} **write;**
T_{59} **upSPout**, **DSout, ldMAR**;
T_{5A} **MBRhigh, mxMBR**, **ldMBR, PCout**;
T_{5B} **write;**

! JMP !

T_{5C} **IR_JAout**, **SDout**, **ldPChigh, ldPClow**;
T_{5D} **bruncnd, val$_{9B}$**;

! JMPIND !

T_{5E} **brimm_regind, val$_{9A}$**;
T_{5F} **Bout**, **SDout**, **ldPChigh, ldPClow**;
T_{60} **bruncnd, val$_{9B}$**;

! RTI !

T_{61} **upSPout**, **DSout, ldMAR, incSP**;
T_{62} **read;**
T_{63} **ldMBR**;
T_{64} **MBRout**, **ldPSW**;

! RTS !

T_{65} **upSPout**, **DSout, ldMAR, incSP**;
T_{66} **read;**
T_{67} **ldMBR, upSPout**, **DSout, ldMAR incSP**;
T_{68} **MBRout**, **ldPChigh**;
T_{69} **read;**
T_{6A} **ldMBR;**
T_{6B} **MBRout**, **ldPClow;**
T_{6C} **bruncnd, val$_{9B}$**;

! INT !

T_{6D} **setINT;**
T_{6E} **bruncnd, val$_{9B}$**;

! PUSH !

T_{6F} **mxMBR**, **ldMBR, decSP, Bout**;
T_{70} **ldMAR, upSPout, DSout, decSP**;
T_{71} **write;**
T_{72} **ldMAR, upSPout, DSout**;
T_{73} **mxMBR, MBRhigh, Bout, ldMBR**;
T_{74} **write;**
T_{75} **bruncnd, val$_{9B}$**;

! POP !

T_{76} **brimmed,val$_{9A}$**;
T_{77} **MARout**, **ldA**;
T_{78} **ldMAR, upSPout, DSout, incSP;**
T_{79} **read;**
T_{7A} **ldMBR, ldMAR, upSPout, DSout, incSP**;
T_{7B} **MBRout, ldBhigh**;
T_{7C} **read;**

T_{7D} **ldMBR, ldMAR, Aout**;
T_{7E} **MBRout, ldBlow;**
T_{7F} **Bout, ldX;**
T_{80} **trans, ldPSWALU**;
T_{81} **bruncnd, val$_{92}$**;

! INC !

T_{82} **brimmed, val$_{9A}$**;
T_{83} **Bout, ldX**;
T_{84} **inc, ALUout, ldBhigh, ldBlow, ldPSWALU**;
T_{85} **bruncnd, val$_{92}$**;

! DEC !

T_{86} **brimmed, val$_{9A}$**;
T_{87} **Bout, ldX**;
T_{88} **dec, ALUout, ldBhigh, ldBlow, ldPSWALU**;
T_{89} **bruncnd, val$_{92}$**;

! INTE !

T_{8A} **setI**;
T_{8B} **bruncnd, val$_{9B}$**;

! INTD !

T_{8C} **resetI**;
T_{8D} **bruncnd, val$_{9B}$**;

! TRPE !

T_{8E} **setT**;
T_{8F} **bruncnd, val$_{9B}$**;

! TRPD !

T_{90} **resetT**;
T_{91} **bruncnd, val$_{9B}$**;

! Vraćanje podatka !

T_{92} **brregdir,val$_{98}$**;
T_{93} **Bout**, **ldMBR, mxMBR**;
T_{94} **write;**
T_{95} **decMAR, Bout**, **ldMBR, mxMBR, MBRhigh**;
T_{96} **write;**
T_{97} **bruncnd, val$_{9B}$**;
T_{98} **Bout**, **ldREG, fvo**;
T_{99} **bruncnd, val$_{9B}$**;

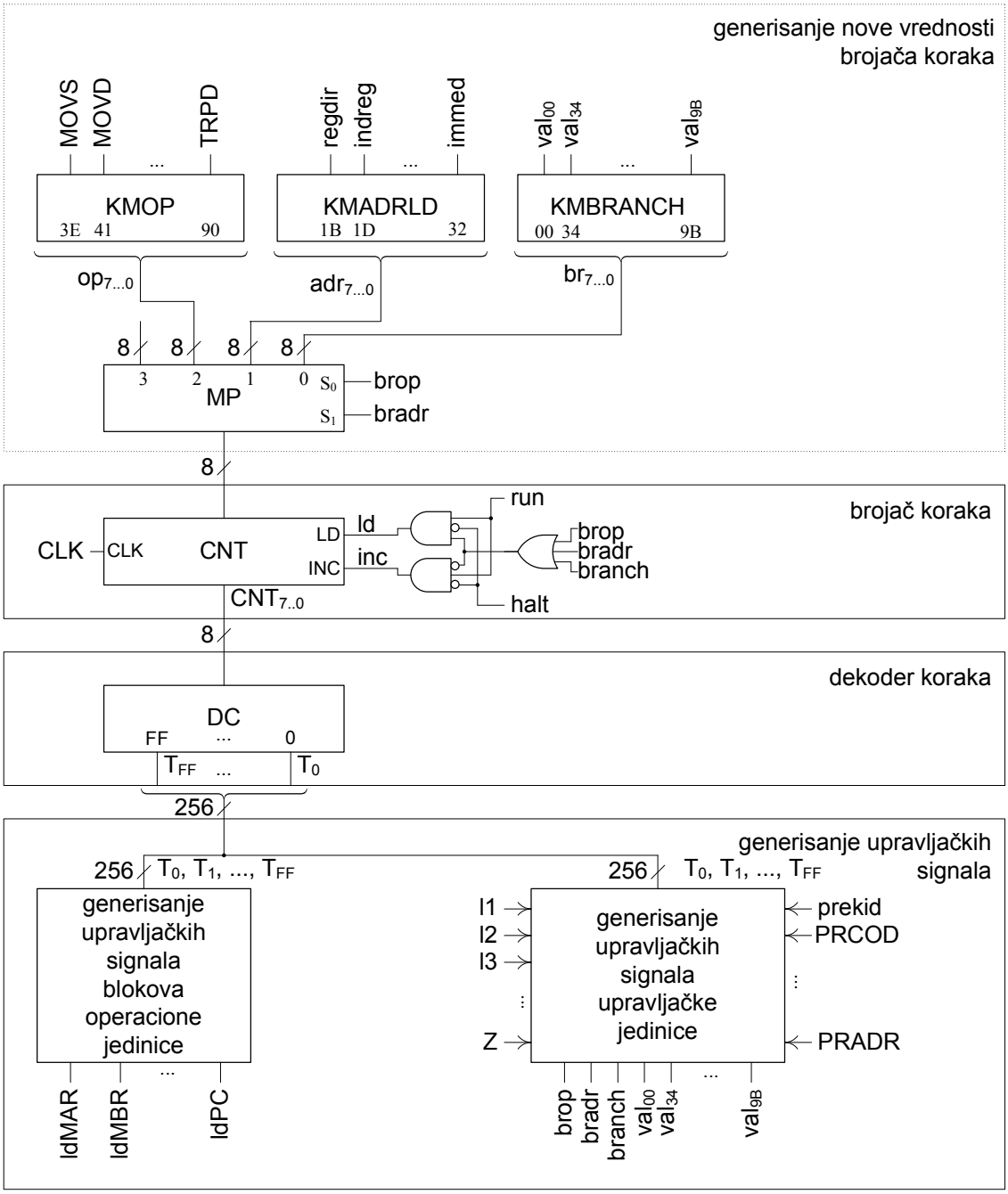
! Opsluživanje prekida !

T_{9A} **setADR**;
T_{9B} **brnotPrekid, val$_{00}$**;
T_{9C} **decSP, PCout**, **mxMBR**, **ldMBR, ldBR, intack**;
T_{9D} **upSPout**, **DSout, ldMAR, decSP**;
T_{9E} **write;**
T_{9F} **upSPout**, **DSout, ldMAR**, **decSP**;
T_{A0} **PCout**, **mxMBR**, **ldMBR, MBRhigh**;
T_{A1} **write;**
T_{A2} **upSPout**, **DSout, ldMAR**;
T_{A3} **PSWout**, **mxMBR**, **ldMBR**;
T_{A4} **write;**
T_{A5} **BRout**, **ldX**
T_{A6} **shl, ALUout, DSout, ldX;**
T_{A7} **IVTPout**, **ldY**;

T_{A8} **add**, **ALUout, ldMAR, DSout ;**
T_{A9} **read;**
T_{AA} **ldMBR**;
T_{AB} **MBRout**, **ldPChigh, incMAR**;
T_{AC} **read;**
T_{AD} **ldMBR**;
T_{AE} **MBRout**, **ldPClow**;
T_{AF} **bruncnd, val$_{00}$**;

Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 2. 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 **opr$_{7...0}$**, kombinacione mreže KMADR koja formira signale **adr$_{7...0}$** i kombinacione mreže KMBR koja formira signale **br$_{7...0}$**. Selekcija jedne od tri grupe signala koji daju novu vrednost brojača koraka obezbeđuje se signalima **bropr** i **bradr** i to: signali **opr$_{7...0}$** ako je aktivan signal **bropr**, signali **adr$_{7...0}$** ako je aktivan signal **bradr** i signali **br$_{7...0}$** ako su neaktivni signali **bropr** i **bradr**.



Slika 2 Struktura upravljačke jedinice

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 5) za realizaciju višestrukog uslovnog skoka u koraku $step_{3B}$ sekvence upravljačkih signala po koracima. U zavisnosti od toga koji od signala **MOVS**, **MOVD**, …, **TRPD** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 5 da se pojavi tada na linijama $\mathbf{opr_{7...0}}$. S obzirom da stanje brojača koraka T_{3B} 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 4) za realizaciju višestrukog uslovnog skoka u koraku $step_{19}$ sekvence upravljačkih signala po koracima. U zavisnosti od toga koji od signala **regdir**, **indreg**,..., **immed** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 4 da se pojavi tada na linijama $\mathbf{adr_{7...0}}$. S obzirom da stanje brojača

koraka T_{19} daje aktivnu vrednost signala višestrukog uslovnog skoka **bradr**, vrednost na linijama $\mathbf{adr_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima brojača koraka $CNT_{7...0}$.

Kombinacionom mrežom KMBR generišu se vrednosti za upis u brojač koraka za bezuslovne skokove (tabela 9) i uslovne skokove (tabela 2) u sekvenci upravljačkih signala po koracima. U zavisnosti od toga koji od signala $\mathbf{val_{00}}$, $\mathbf{val_{04}}$, ..., $\mathbf{val_{98}}$ ima aktivnu vrednost zavisi koja će od vrednosti iz tabela 9 i 2 tada da se pojavi na linijama $\mathbf{br_{7...0}}$. Signali višestrukih uslovnih skokova **bradr** i **bropr** su aktivni samo u stanjima T_{19} i T_{3B} brojača koraka, a u svim ostalim neaktivni. S obzirom da nijedan od ova dva signala nije aktivan u stanjima brojača koraka kada treba realizovati bezuslovni ili neki od uslovnih skokova, vrednost na linijama $\mathbf{br_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima brojača koraka $CNT_{7...0}$.

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

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača $CNT_{7...0}$ za jedan čime se obezbeđuje sekvencijalno generisanje upravljačkih signala iz sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 6). 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 6). 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 6). S obzirom da se upravljački signali svih faza izvršavanja instrukcija realizuju u opsegu od koraka T_{00} do koraka T_{AF} 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 6).

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala $\mathbf{T_0}$, $\mathbf{T_1}$, ..., $\mathbf{T_{FF}}$ koji dolaze sa bloka *dekoder koraka*, signala logičkih uslova **l1**, **l2**, ..., **PREKID** koji dolaze iz operacione jedinice i saglasno sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 6) 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:

- $\mathbf{ldMAR = T_{00} + T_{03} + T_{08} + T_{10} + T_{14} + T_{1D} + T_{21} + T_{23} + T_{25} + T_{2C} + T_{30} + T_{57} + T_{59} + T_{61} + T_{65} + T_{67} + T_{70} + T_{72} + T_{78} + T_{7A} + T_{7D} + T_{9D} + T_{9F} + T_{A2} + T_{A8}}$
- $\mathbf{REGout = T_{1B} + T_{1D} + T_{1F} + T_{42} + T_{45} + T_{49}}$

- **setI** = T_{8A}

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

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr** = T_{3B}
- **bradr** = T_{19}
- **branch** = **bruncnd** + **brl1*****l1** + **brl2*****l2** + **brl3*****l3** + **brl4*****l4** + **brl5*****l5** + **brMOVD_POP***(**MOVD or POP**) + **brimmed*****immed** + **brregdir*****regdir** + **brimm_regdir***(**immed or regdir**) + **brZ*****Z**+ **brnotPREKID*** $\overline{\text{PREKID}}$
- $\mathbf{val_{00}} = T_{9B} + T_{AF}$
- $\mathbf{val_{0E}} = T_{04}$
- $\mathbf{val_{19}} = T_{11} + T_{15}$
- $\mathbf{val_{3B}} = T_{05} + T_{09} + T_{0D} + T_{1C} + T_{33} + T_{34}$
- $\mathbf{val_{98}} = T_{92}$
- $\mathbf{val_{9A}} = T_{1A} + T_{41} + T_{4D} + T_{5E} + T_{76} + T_{82} + T_{86}$
- $\mathbf{val_{9B}} = T_{3D} + T_{40} + T_{48} + T_{4C} + T_{51} + T_{55} + T_{5D} + T_{60} + T_{6C} + T_{6E} + T_{75} + T_{8B} + T_{8D} + T_{8F} + T_{91} + T_{97} + T_{99}$

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

- **bruncnd** = $T_{0D} + T_{1A} + T_{1C} + T_{1E} + T_{22} + T_{24} + T_{2D} + T_{31} + T_{33} + T_{3D} + T_{40} + T_{44} + T_{48} + T_{4C} + T_{50} + T_{55} + T_{5D} + T_{60} + T_{6C} + T_{6E} + T_{75} + T_{81} + T_{85} + T_{89} + T_{8B} + T_{8D} + T_{8F} + T_{91} + T_{97} + T_{99} + T_{AF}$
- **brl1** = T_{04}
- **brl2** = T_{05}
- **brl3** = T_{09}
- **brl4** = T_{11}
- **brl5** = T_{15}
- **brMOVD_POP** = T_{34}
- **brimmed** = $T_{41} + T_{4D} + T_{76} + T_{82} + T_{86}$
- **brregdir** = T_{92}
- **brimm_regdir** = T_{5E}
- **brZ** = T_{51}
- **brnotPREKID** = T_{9B}

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **l3**, **l4**, **l5**, **MOVD, POP**, **immed**, **regdir, Z** 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* $step_A$, *br* (*if* **uslov** *then* $step_A$) i *br* (*case* (**$uslov_1$**, ..., **$uslov_n$**) *then* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$).

Umesto iskaza *br* $step_A$ treba staviti signal bezuslovnog skoka i signal **val_A**. Simbolička oznaka signala bezuslovnog skoka je **bruncnd**. Koraci $step_A$ na koje treba bezuslovno preći, simboličke oznake signala **val_A** i vrednosti A koje treba upisati u brojač koraka, dati su u tabeli 7.

Tabela 7 Koraci $step_A$, signali **val_A** i vrednosti A za bezuslovne skokove

$step_A$	**val_A**	A
$step_{00}$	**val_{00}**	00
$step_{2A}$	**val_{2A}**	2A
$step_{31}$	**val_{31}**	31
$step_{76}$	**val_{76}**	76
$step_{7D}$	**val_{7D}**	7D
$step_{7E}$	**val_{7E}**	7E

Umesto iskaza *br* (*if* **uslov** *then* $step_A$) treba staviti signal uslovnog skoka koji određuje signal uslova **uslov** na koji se vrši provera i signal **val_A**. Simboličke oznake signala uslovnih skokova i signala uslova dati su u tabeli 8. Koraci $step_A$ na koje treba preći ukoliko je signal **uslov** aktivan, simboličke oznake signala **val_A** i vrednosti A koje treba tada upisati u brojač koraka, dati su u tabeli 9.

Tabela 8 Signali uslovnih skokova i signali uslova

signal uslovnog skoka	signal uslova
brl1	**l1**
brl2	$\overline{l2}$
brl3	**l3**
brl4	$\overline{l4}$
brl5	**l5**
brMOVD_POP	**MOVD or POP**
brimmed	**immed**
brregdir	**regdir**
brimm_regdir	**immed or regdir**
brZ	**Z**
brnotPREKID	$\overline{\text{PREKID}}$

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

$step_A$	**val_A**	A
$step_{00}$	**val_{00}**	00
$step_{0C}$	**val_{0C}**	0C
$step_{16}$	**val_{16}**	16
$step_{31}$	**val_{31}**	31
$step_{7C}$	**val_{7C}**	7C
$step_{7D}$	**val_{7D}**	7D
$step_{7E}$	**val_{7E}**	7E

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 **$uslov_1$**, **$uslov_2$**,..., **$uslov_n$** na koje se vrši provera. Simboličke oznake signala višestrukog uslovnog skoka date su u tabeli 10. 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 11 i 12.

Tabela 10 Signali višestrukih uslovnih skokova

Korak	signal višestrukog uslovnog skoka
$step_{16}$	**bradr**
$step_{31}$	**bropr**

Tabela 11 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{0F}$

signal uslova	vrednost
regdir	18
indreg	19
indregpom	1A
dirmem	1D
indmem	1E
rel	26
immed	29

Tabela 12 Signali uslova i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{2A}$

signal uslova	vrednost	signal uslova	vrednost
MOVS	33	**RTS**	52
MOVD	35	**INT**	59
ADD	38	**PUSH**	5A
AND	3B	**POP**	61
ASR	3E	**INC**	6C
BNZ	41	**DEC**	6F
JSR	45	**INTE**	72
JMP	4B	**INTD**	73
JMPIND	4C	**TRPE**	74
RTI	4E	**TRPD**	75

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 13). Ona ima istu formu kao i tabela 6 za upravljačku jedinicu bez spajanja koraka.

Tabela 13 Sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije sa spajanjem koraka

! Čitanje instrukcije !

T_{00} **resetF, PCout**, **ldMAR**;
T_{01} **read;**
T_{02} **ldMBR**, **incPC**;
T_{03} **MBRout**, **ldIR1**, **PCout**, **ldMAR;**
T_{04} **brl1, val$_{0C}$**;
T_{05} **brl2, val$_{31}$**;
T_{06} **read;**
T_{07} **ldMBR**, **incPC**;
T_{08} **MBRout**, **ldIR2**, **PCout**, **ldMAR, brl3, val$_{31}$**;
T_{09} **read;**
T_{0A} **ldMBR**, **incPC**;
T_{0B} **MBRout**, **ldIR3, bruncnd, val$_{31}$**;

T_{0C} **read;**
T_{0D} **ldMBR**, **incPC**;
T_{0E} **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
T_{0F} **brl4, val$_{16}$**;
T_{10} **read;**
T_{11} **ldMBR**, **incPC**;
T_{12} **MBRout**, **ldIR3**, **PCout**, **ldMAR, brl5, val$_{16}$**;
T_{13} **read;**
T_{14} **ldMBR**, **incPC**;
T_{15} **MBRout**, **ldIR4;**

! Formiranje adrese i čitanje operanda !

T_{16} **bradr**;
T_{17} **bruncnd, val$_{7D}$**;

! Direktno registarsko !

T_{18} **REGout, ldBlow, ldBhigh, fdo, bruncnd, val$_{31}$**;

! Indirektno registarsko !

T_{19} **REGout**,**DSout, ldMAR, fdo, bruncnd, val$_{2A}$**;

! Indirektno registarsko sa pomerajem !

T_{1A} **REGout, ldX, DSout, fdo**;
T_{1B} **IR3out**, **ldY**;
T_{1C} **add**, **ldMAR, DSout, ALUout, bruncnd, val$_{2A}$**;

! Direktno memorijsko !

T_{1D} **IR_DAout**, **ldMAR, bruncnd, val$_{2A}$**;

! Indirektno memorijsko !

T_{1E} **IR_DAout**, **ldMAR**;
T_{1F} **read;**
T_{20} **ldMBR, incMAR**;
T_{21} **ldBhigh, MBRout**;
T_{22} **read;**
T_{23} **ldMBR;**
T_{24} **ldBlow, MBRout**;
T_{25} **Bout**, **ldMAR, bruncnd, val$_{2A}$**;

! Relativno !

T_{26} **ldX, PCout**;
T_{27} **IR3out, ldY**;
T_{28} **add**, **ALUout, DSout, ldMAR, bruncnd, val$_{2A}$**;

! Neposredno !

T_{29} **IR_DAout**, **SDout**, **ldBhigh, ldBlow, bruncnd, val$_{31}$**;

! Čitanje operanda za memorijska adresiranja !

T_{2A} **brMOVD_POP, val$_{31}$**;
T_{2B} **read;**
T_{2C} **ldMBR, incMAR**;
T_{2D} **MBRout**, **ldBhigh;**
T_{2E} **read;**
T_{2F} **ldMBR;**
T_{30} **MBRout**, **ldBlow**;

! Izvršavanje operacije !

T_{31} **bropr**;

! Kod operacije !

```
        T32     setCOD,bruncnd, val7E;
! MOVS !
        T33     Bout, daREG, ldREG, ldX;
        T34     trans, ldPSWALU, bruncnd, val7E;
! MOVD !
        T35     brimmed, val7D;
        T36     REGout, daREG, ldBlow, ldBhigh, DSout, ldX;
        T37     trans, ldPSWALU, bruncnd, val76;
! ADD !
        T38     REGout, daREG, DSout, ldX;
        T39     Bout, ldY;
        T3A     add, ALUout, ldREG, daREG, ldPSWALU, DSout, bruncnd, val7E;
! AND !
        T3B     REGout, daREG, DSout, ldX;
        T3C     Bout, ldY;
        T3D     and, ALUout, ldREG, daREG, ldPSWALU, DSout, bruncnd, val7E;
! ASR !
        T3E     brimmed, val7D;
        T3F     Bout, ldX;
        T40     asr, ALUout, ldBhigh, ldBlow, ldPSWALU, bruncnd, val76;
! BNZ !
        T41     brZ, val7E;
        T42     ldX, PCout;
        T43     ldY, IR2out;
        T44     add, ALUout, ldPChigh, ldPClow, bruncnd, val7E;
! JSR !
        T45     decSP, mxMBR, ldMBR, PCout;
        T46     upSPout, decSP, DSout, ldMAR;
        T47     write;
        T48     upSPout, DSout, ldMAR;
        T49     MBRhigh, mxMBR, ldMBR, PCout;
        T4A     write;
! JMP !
        T4B     IR_JAout, SDout, ldPChigh, ldPClow, bruncnd, val7E;
! JMPIND !
        T4C     brimm_regind, val7D;
        T4D     Bout, SDout, ldPChigh, ldPClow, bruncnd, val7E;
! RTI !
        T4E     upSPout, DSout, ldMAR, incSP;
        T4F     read;
        T50     ldMBR;
        T51     MBRout, ldPSW;
! RTS !
        T52     upSPout, DSout, ldMAR, incSP;
        T53     read;
        T54     ldMBR, upSPout, DSout, ldMAR incSP;
        T55     MBRout, ldPChigh;
        T56     read;
        T57     ldMBR;
        T58     MBRout, ldPClow, bruncnd, val7E;
! INT !
        T59     setINT, bruncnd, val7E;
! PUSH !
        T5A     mxMBR, ldMBR, decSP, Bout;
        T5B     ldMAR, upSPout, DSout, decSP;
```

T_{5C} **write;**
T_{5D} **ldMAR, upSPout, DSout**;
T_{5E} **mxMBR, MBRhigh, Bout, ldMBR**;
T_{5F} **write;**
T_{60} **bruncnd, val$_{7E}$**;

! POP !

T_{61} **brimmed, val$_{7D}$**;
T_{62} **MARout**, **ldA**;
T_{63} **ldMAR, upSPout, DSout, incSP;**
T_{64} **read;**
T_{65} **ldMBR, ldMAR, upSPout, DSout, incSP**;
T_{66} **MBRout, ldBhigh**;
T_{67} **read;**
T_{68} **ldMBR, ldMAR, Aout**;
T_{69} **MBRout, ldBlow;**
T_{6A} **Bout, ldX;**
T_{6B} **trans, ldPSWALU, bruncnd, val$_{76}$**;

! INC !

T_{6C} **brimmed, val$_{7D}$**;
T_{6D} **Bout, ldX**;
T_{6E} **inc, ALUout, ldBhigh, ldBlow, ldPSWALU, bruncnd, val$_{76}$**;

! DEC !

T_{6F} **brimmed, val$_{7D}$**;
T_{70} **Bout, ldX**;
T_{71} **dec, ALUout, ldBhigh, ldBlow, ldPSWALU, bruncnd, val$_{76}$**;

! INTE !

T_{72} **setI, bruncnd, val$_{7E}$**;

! INTD !

T_{73} **resetI, bruncnd, val$_{7E}$**;

! TRPE !

T_{74} **setT, bruncnd, val$_{7E}$**;

! TRPD !

T_{75} **resetT, bruncnd, val$_{7E}$**;

! Vraćanje podatka !

T_{76} **brregdir, val$_{7C}$**;
T_{77} **Bout**, **ldMBR, mxMBR**;
T_{78} **write;**
T_{79} **decMAR Bout**, **ldMBR, mxMBR, MBRhigh**;
T_{7A} **write;**
T_{7B} **bruncnd, val$_{7E}$**;
T_{7C} **Bout**, **ldREG, fvo, bruncnd, val$_{7E}$**;

! Opsluživanje prekida !

T_{7D} **setADR**;

T_{7E} **brnotPrekid, val$_{00}$**;
T_{7F} **decSP, PCout**, **mxMBR**, **ldMBR, ldBR, intack**;
T_{80} **upSPout**, **DSout, ldMAR, decSP**;
T_{81} **write;**
T_{82} **upSPout**, **DSout, ldMAR**, **decSP**;
T_{83} **PCout**, **mxMBR**, **ldMBR, MBRhigh**;
T_{84} **write;**
T_{85} **upSPout**, **DSout, ldMAR**;
T_{86} **PSWout**, **mxMBR**, **ldMBR**;

T_{87} **write;**
T_{88} **BRout**, **ldX**
T_{89} **shl, ALUout, DSout, ldX;**
T_{8A} **IVTPout**, **ldY**;
T_{8B} **add**, **ALUout, ldMAR, DSout ;**
T_{8C} **read;**
T_{8D} **ldMBR**;
T_{8E} **MBRout**, **ldPChigh, incMAR**;
T_{8F} **read;**
T_{90} **ldMBR**;
T_{91} **MBRout**, **ldPClow, bruncnd, val$_{00}$**;

Struktura upravljačke jedinice sa spajanjem koraka je ista kao i za slučaj bez spajanja koraka (slika 2). 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_{91}. 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 3E, 41, ..., 90 pri aktivnim vrednostima signala **MOVS**, **MOVD**, ..., **TRPD**, respektivno. Kombinaciona mreža KMADR generiše vrednosti 1B, 1D, ..., 32 pri aktivnim vrednostima signala **regdir**, **indreg**, ..., **immed**, respektivno. Kombinaciona mreža KMBR generiše vrednosti 00, 0C, ..., 7E pri aktivnim vrednostima signala **val$_{00}$**, **val$_{0C}$**, ..., **val$_{7E}$**, respektivno. Zbog drugih vrednosti koje generiše kombinaciona mreža KMBR javljaju se drugi signali **val$_A$**. Tako se umesto signala **val$_{31}$**, javlja signal **val$_{2A}$**, jer u brojač koraka umesto vrednosti 31 treba upisati 2A itd.

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

Upravljački signali operacione jedinice se generišu na sledeći način:

- **ldMAR** = $T_{00} + T_{03} + T_{08} + T_{0E} + T_{12} + T_{19} + T_{1C} + T_{1D} + T_{1E} + T_{25} + T_{28} + T_{46} + T_{48} + T_{4E} + T_{52} + T_{54} + T_{5B} + T_{5D} + T_{63} + T_{65} + T_{68} + T_{80} + T_{82} + T_{85} + T_{8B}$
- **REGout** = $T_{18} + T_{19} + T_{1A} + T_{36} + T_{38} + T_{3B} + T_{49}$
- **setI** = T_{72}

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

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr** = T_{31}
- **bradr** = T_{16}
- **branch** = **bruncnd** + **brl1*l1** + **brl2*l2** + **brl3*l3** + **brl4*l4** + **brl5*l5** + **brMOVD_POP***(**MOVD or POP**) + **brimmed*immed** + **brregdir*regdir** + **brimm_regdir***(**immed or regdir**) + **brZ*Z**+ **brnotPREKID*** $\overline{\text{PREKID}}$
- **val$_{00}$** = $T_{7E} + T_{91}$
- **val$_{0C}$** = T_{04}
- **val$_{16}$** = $T_{0F} + T_{12}$
- **val$_{31}$** = $T_{05} + T_{08} + T_{0B} + T_{18} + T_{29} + T_{2A}$
- **val$_{7C}$** = T_{76}
- **val$_{7D}$** = $T_{17} + T_{35} + T_{3E} + T_{4C} + T_{61} + T_{6C} + T_{6F}$
- **val$_{7E}$** = $T_{32} + T_{34} + T_{3A} + T_{3D} + T_{41} + T_{44} + T_{4B} + T_{4D} + T_{58} + T_{59} + T_{60} + T_{72} + T_{73} + T_{74} + T_{75} + T_{7B} + T_{7C}$

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

- **bruncnd** = $T_{0B} + T_{17} + T_{18} + T_{19} + T_{1C} + T_{1D} + T_{25} + T_{28} + T_{29} + T_{32} + T_{34} + T_{37} + T_{3A} + T_{3D} + T_{40} + T_{44} + T_{4B} + T_{4D} + T_{58} + T_{59} + T_{60} + T_{6B} + T_{6E} + T_{71} + T_{72} + T_{73} + T_{74} + T_{75} + T_{7B} + T_{7C} + T_{91}$
- **brl1** = T_{04}
- **brl2** = T_{05}
- **brl3** = T_{08}
- **brl4** = T_{0F}
- **brl5** = T_{12}
- **brMOVD_POP** = T_{2A}
- **brimmed** = $T_{35} + T_{3E} + T_{61} + T_{6C} + T_{6F}$
- **brregdir** = T_{76}
- **brimm_regdir** = T_{4C}
- **brZ** = T_{41}
- **brnotPREKID** = T_{7E}

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **l3**, **l4**, **l5**, **MOVD, POP**, **immed**, **regdir, Z** i **PREKID**.

4.2 MIKROPROGRAMSKA REALIZACIJA

4.2.1 Mikroprogramska realizacija sa horizontalnim formatom mikroinstrukcija

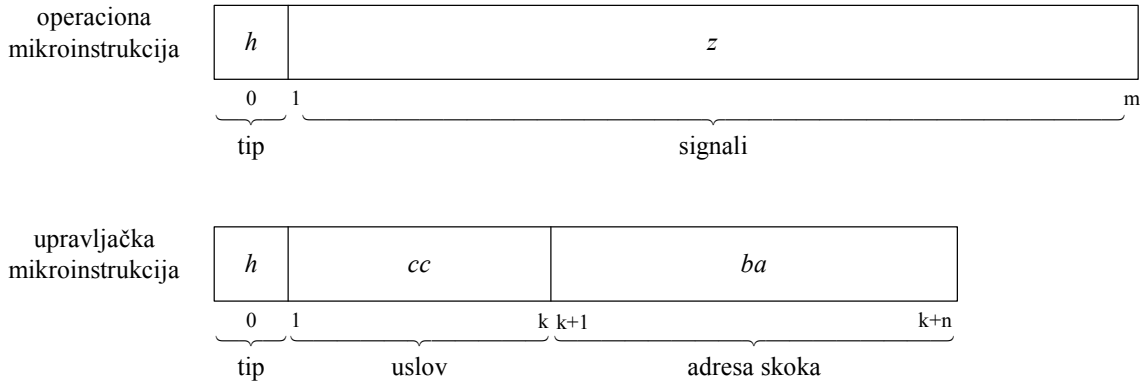
4.2.1.1 Mikroprogramska realizacija sa horizontalnim formatom mikroinstrukcija sa dva tipa instrukcija

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 3 i svakom upravljačkom koraku, u kome se realizuju skokovi, pridružuje binarna reč čiji je format dat na slici 3. 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 3 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 3) i to operaciona mikroinstrukcija i upravljačka mikroinstrukcija.

Format operacione mikroinstrukcije je dat na slici 4. Polje *h* je 0. Bitovi polja *z* dodeljeni su upravljačkim signalima operacione jedinice.

0	1	2	3	4	5	6	7
0	resetF	PCout	ldMAR	read	write	ldMBR	incPC

8	9	10	11	12	13	14	15
MBRout	ldIR1	ldIR2	ldIR3	ldIR4	REGout	ldBlow	ldBhigh

16	17	18	19	20	21	22	23
fdo	DSout	SDout	IR3out	ldX	ldY	add	and

24	25	26	27	28	29	30	31
inc	dec	asr	shl	trans	ALUout	IR_DAout	incMAR

32	33	34	35	36	37	38	39
decMAR	Bout	Aout	setCOD	setADR	setINT	daREG	ldREG

40	41	42	43	44	45	46	47
ldPSWALU	IR2out	decSP	incSP	mxMBR	upSPout	MBRhigh	IR_JAout

48	49	50	51	52	53	54	55
ldPChigh	ldPClow	MARout	ldA	setI	resetI	setT	reset

56	57	58	59	60	61	62	63
fvo	intack	PSWout	ldPSW	BRout	IVTPout	ldBR	/

Slika 4 Operaciona mikroinstrukcija

Format upravljačke mikroinstrukcije je dat na slici 5. Polje *h* je 1.

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

8	9	10	11	12	13	14	15
ba							

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

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

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

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

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

56	57	58	59	60	61	62	63
/	/	/	/	/	/	/	/

Slika 5 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 14.

Tabela 14 Signal bezuslovnog skoka

cc	signal bezuslovnog skoka
01	**bruncond**

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 15.

Tabela 15 Signali uslovnih skokova

cc	signal uslovnog skoka	signal uslova
02	**brl1**	**l1**
03	**brnotl2**	$\overline{\text{l2}}$

04	**brl3**	**l3**
05	**brnotl4**	$\overline{l4}$
06	**brl5**	**l5**
07	**brMOVDorPOP**	**MOVD or POP**
08	**brimmed**	**immed**
09	**brZero**	**Z**
0A	**brIMMEDorREGIND**	**immed or regind**
0B	**brREGDIR**	regdir
0C	**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 16.

Tabela 16 Signali višestrukih uslovnih skokova

cc	korak	signal višestrukog uslovnog skoka
0D	step$_{0F}$	**bradr**
0E	step$_{31}$	**bropr**

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

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

Operacioina mikroinstrukcija je duža od upravljačke mikroinstrukcije, pa je dužina mikroinstrukcije određena dužinom operacione mikroinstrukcije i iznosi 64 bita.

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 14. 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 17.

Tabela 17 Koraci $step_A$, adrese $madr_A$ i vrednosti A za bezuslovne skokove

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

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

Simboličke oznake signala uslovnog skoka, način njihovog kodiranja poljem *cc* i signali **uslov** za sve iskaze ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima dati su u tabeli 15. 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 18.

Tabela 18 Koraci $step_A$, adrese $madr_A$ i vrednosti A za uslovne skokove

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

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

Simboličke oznake signala višestrukih uslovnih skokova, način njihovog kodiranja poljem *cc* i koraci u sekvenci upravljačkih signala po koracima u kojima se javljaju iskazi ovog tipa dati su u tabeli 16. 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 19 i 20.

Tabela 19 Signali uslova i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{0F}$

signal uslova	vrednost
dirreg	1B
regind	1D
regindpom	1F
memdir	23
memind	25
rel	2E
immed	32

Tabela 20 Signali uslova i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{31}$

signal uslova	vrednost	signal uslova	vrednost
MOVS	3E	**RTS**	65
MOVD	41	**INT**	6D
ADD	45	**PUSH**	6F
AND	49	**POP**	76
ASR	4D	**INC**	82
BNZ	51	**DEC**	86
JSR	56	**INTE**	8A
JMP	5C	**INTD**	8C
JMPIND	5E	**TRPE**	8E
RTI	61	**TRPD**	90

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 1) formiran mikroprogram (tabela 21). 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 21 Mikroprogram

! Čitanje instrukcije !

$madr_{00}$ **resetF, PCout**, **ldMAR**;
$madr_{01}$ **read;**
$madr_{02}$ **ldMBR, incPC**;
$madr_{03}$ **MBRout, ldIR1, PCout, ldMAR;**
$madr_{04}$ **cnt, brl1, madr**$_{0E}$
$madr_{05}$ **cnt, brnotl2, madr**$_{3B}$
$madr_{06}$ **read;**
$madr_{07}$ **ldMBR, incPC**
$madr_{08}$ **MBRout, ldIR2, PCout, ldMAR;**
$madr_{09}$ **cnt, brl3, madr**$_{3B}$
$madr_{0A}$ **read;**
$madr_{0B}$ **ldMBR**, **incPC**;
$madr_{0C}$ **MBRout**, **ldIR3**;
$madr_{0D}$ **cnt, bruncond, madr**$_{3B}$;

$madr_{0E}$ **read;**
$madr_{0F}$ **ldMBR**, **incPC**;
$madr_{10}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
$madr_{11}$ **cnt, brnotl4, madr**$_{19}$**;**

madr$_{12}$ **read;**
madr$_{13}$ **ldMBR**, **incPC**;
madr$_{14}$ **MBRout**, **ldIR3**, **PCout**, **ldMAR**;
madr$_{15}$ **cnt, brl5, madr**$_{19}$;
madr$_{16}$ **read;**
madr$_{17}$ **ldMBR**, **incPC**;
madr$_{18}$ **MBRout**, **ldIR4;**

! Formiranje adrese i čitanje operanda !

madr$_{19}$ **cnt, bradr;**
madr$_{1A}$ **cnt, bruncond, madr**$_{9A}$;
! Direktno registarsko !
madr$_{1B}$ **REGout, ldBlow, ldBhigh, fdo**;
madr$_{1C}$ **cnt, bruncond, madr**$_{3B}$;
! Indirektno registarsko !
madr$_{1D}$ **REGout**,**DSout, ldMAR, fdo**;
madr$_{1E}$ **cnt, bruncond, madr**$_{34}$;
! Indirektno registarsko sa pomerajem !
madr$_{1F}$ **REGout, ldX, DSout, fdo**;
madr$_{20}$ **IR3out**, **ldY**;
madr$_{21}$ **add**, **ldMAR, DSout, ALUout**;
madr$_{22}$ **cnt, bruncond, madr**$_{34}$;
! Direktno memorijsko !
madr$_{23}$ **IR_DAout**, **ldMAR**;
madr$_{24}$ **cnt, bruncond, madr**$_{34}$;
! Indirektno memorijsko !
madr$_{25}$ **IR_DAout**, **ldMAR**;
madr$_{26}$ **read;**
madr$_{27}$ **ldMBR, incMAR**;
madr$_{28}$ **ldBhigh, MBRout**;
madr$_{29}$ **read;**
madr$_{2A}$ **ldMBR;**
madr$_{2B}$ **ldBlow, MBRout**;
madr$_{2C}$ **Bout**, **ldMAR**;
madr$_{2D}$ **cnt, bruncond, madr**$_{34}$;
! Relativno !
madr$_{2E}$ **ldX, PCout**;
madr$_{2F}$ **IR3out, ldY**;
madr$_{30}$ **add**, **ALUout, DSout, ldMAR**;
madr$_{31}$ **cnt, bruncond, madr**$_{34}$;
! Neposredno !
madr$_{32}$ **IR_DAout**, **SDout**, **ldBhigh, ldBlow**;
madr$_{33}$ **cnt, bruncond, madr**$_{3B}$;
! Čitanje operanda za memorijska adresiranja !
madr$_{34}$ **cnt, brMOVDorPOP, madr**$_{3B}$;
madr$_{35}$ **read;**
madr$_{36}$ **ldMBR, incMAR**;
madr$_{37}$ **MBRout**, **ldBhigh;**
madr$_{38}$ **read;**
madr$_{39}$ **ldMBR;**
madr$_{3A}$ **MBRout**, **ldBlow**;

! Izvršavanje operacije !

madr$_{3B}$ **cnt**, **bropr**;

```
        madr3C  setCOD;
        madr3D  cnt, bruncond, madr9B;

! MOVS !
        madr3E  Bout, daREG, ldREG, ldX;
        madr3F  trans, ldPSWALU;
        madr40  cnt, bruncond, madr9B;
! MOVD !
        madr41  cnt, brimmed, madr9A;
        madr42  REGout, daREG, ldBlow, ldBhigh, DSout, ldX;
        madr43  trans, ldPSWALU;
        madr44  cnt, bruncond, madr92;
! ADD !
        madr45  REGout, daREG, DSout, ldX;
        madr46  Bout, ldY;
        madr47  add, ALUout, ldREG, daREG, ldPSWALU, DSout;
        madr48  cnt, bruncond, madr9B;
! AND !
        madr49  REGout, daREG, DSout, ldX;
        madr4A  Bout, ldY;
        madr4B  and, ALUout, ldREG, daREG, ldPSWALU, DSout;
        madr4C  cnt, bruncond, madr9B;
! ASR !
        madr4D  cnt, brimmed, madr9A;
        madr4E  Bout, ldX;
        madr4F  asr, ALUout, ldBhigh, ldBlow, ldPSWALU;
        madr50  cnt, bruncond, madr92;
! BNZ!
        madr51  cnt, brZero, madr9B;
        madr52  ldX, PCout;
        madr53  ldY, IR2out;
        madr54  add, ALUout, ldPChigh, ldPClow;
        madr55  cnt, bruncond, madr9B;
! JSR !
        madr56  decSP, mxMBR, ldMBR, PCout;
        madr57  upSPout, decSP, DSout, ldMAR;
        madr58  write;
        madr59  upSPout, DSout, ldMAR;
        madr5A  MBRhigh, mxMBR, ldMBR, PCout;
        madr5B  write;
! JMP !
        madr5C  IR_JAout, SDout, ldPChigh, ldPClow;
        madr5D  cnt, bruncond, madr9B;
! JMPIND !
        madr5E  cnt, brIMMEDorREGIND, madr9A;
        madr5F  Bout, SDout, ldPChigh, ldPClow;
        madr60  cnt, bruncond, madr9B;
! RTI !
        madr61  upSPout, DSout, ldMAR, incSP;
        madr62  read;
        madr63  ldMBR;
        madr64  MBRout, ldPSW;
! RTS !
        madr65  upSPout, DSout, ldMAR, incSP;
        madr66  read;
```

madr$_{67}$ **ldMBR, upSPout**, **DSout, ldMAR incSP**;
madr$_{68}$ **MBRout**, **ldPChigh**;
madr$_{69}$ **read;**
madr$_{6A}$ **ldMBR;**
madr$_{6B}$ **MBRout**, **ldPClow;**
madr$_{6C}$ **cnt, bruncond, madr**$_{9B}$;

! INT !

madr$_{6D}$ **setINT;**
madr$_{6E}$ **cnt, bruncond, madr**$_{9B}$;

! PUSH !

madr$_{6F}$ **mxMBR**, **ldMBR, decSP, Bout**;
madr$_{70}$ **ldMAR, upSPout, DSout, decSP**;
madr$_{71}$ **write;**
madr$_{72}$ **ldMAR, upSPout, DSout**;
madr$_{73}$ **mxMBR, MBRhigh, Bout, ldMBR**;
madr$_{74}$ **write;**
madr$_{75}$ **cnt, bruncond, madr**$_{9B}$;

! POP !

madr$_{76}$ **cnt, brimmed, madr**$_{9A}$;
madr$_{77}$ **MARout**, **ldA**;
madr$_{78}$ **ldMAR, upSPout, DSout, incSP;**
madr$_{79}$ **read;**
madr$_{7A}$ **ldMBR, ldMAR, upSPout, DSout, incSP**;
madr$_{7B}$ **MBRout, ldBhigh**;
madr$_{7C}$ **read;**
madr$_{7D}$ **ldMBR, ldMAR, Aout**;
madr$_{7E}$ **MBRout, ldBlow;**
madr$_{7F}$ **Bout, ldX;**
madr$_{80}$ **trans, ldPSWALU**;
madr$_{81}$ **cnt, bruncond, madr**$_{92}$;

! INC !

madr$_{82}$ **cnt, brimmed, madr**$_{9A}$;
madr$_{83}$ **Bout, ldX**;
madr$_{84}$ **inc, ALUout, ldBhigh, ldBlow, ldPSWALU**;
madr$_{85}$ **cnt, bruncond, madr**$_{92}$;

! DEC !

madr$_{86}$ **cnt, brimmed, madr**$_{9A}$;
madr$_{87}$ **Bout, ldX**;
madr$_{88}$ **dec, ALUout, ldBhigh, ldBlow, ldPSWALU**;
madr$_{89}$ **cnt, bruncond, madr**$_{92}$;

! INTE !

madr$_{8A}$ **setI**;
madr$_{8B}$ **cnt, bruncond, madr**$_{9B}$;

! INTD !

madr$_{8C}$ **resetI**;
madr$_{8D}$ **cnt, bruncond, madr**$_{9B}$;

! TRPE !

madr$_{8E}$ **setT**;
madr$_{8F}$ **cnt, bruncond, madr**$_{9B}$;

! TRPD !

madr$_{90}$ **resetT**;
madr$_{91}$ **cnt, bruncond, madr**$_{9B}$;

! Vraćanje podatka !

madr$_{92}$ **cnt, brregdir, madr**$_{98}$;
madr$_{93}$ **Bout**, **ldMBR, mxMBR**;

$madr_{94}$ **write;**
$madr_{95}$ **decMAR, Bout**, **ldMBR, mxMBR, MBRhigh**;
$madr_{96}$ **write;**
$madr_{97}$ **cnt, bruncond, madr**$_{9B}$;
$madr_{98}$ **Bout**, **ldREG, fvo**;
$madr_{99}$ **cnt, bruncond, madr**$_{9B}$;

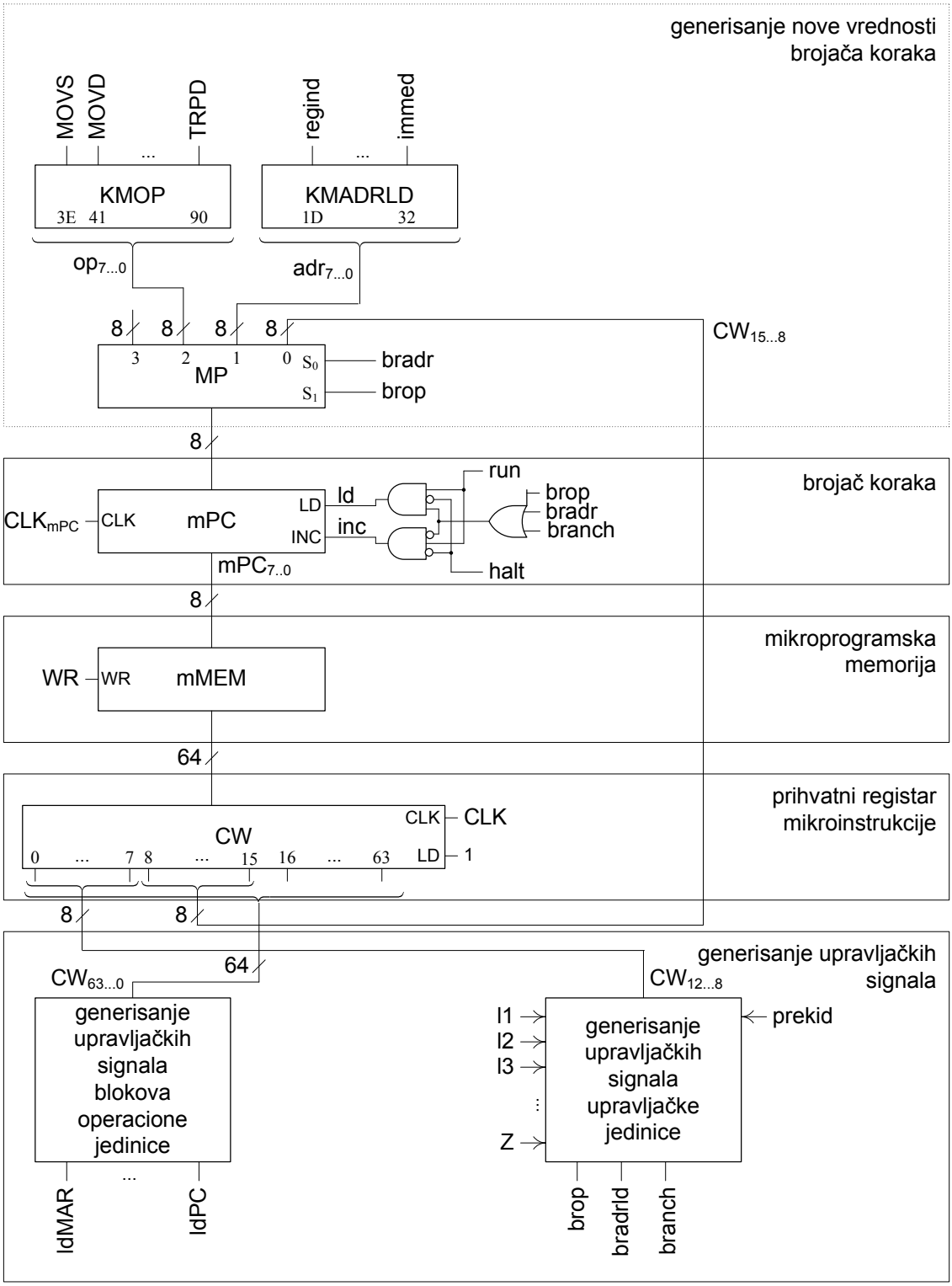
! Opsluživanje prekida !

$madr_{9A}$ **setADR**;

$madr_{9B}$ **cnt, brnotPREKID, madr**$_{00}$;
$madr_{9C}$ **decSP, PCout**, **mxMBR**, **ldMBR, ldBR, intack**;
$madr_{9D}$ **upSPout**, **DSout, ldMAR, decSP**;
$madr_{9E}$ **write;**
$madr_{9F}$ **upSPout**, **DSout, ldMAR**, **decSP**;
$madr_{A0}$ **PCout**, **mxMBR**, **ldMBR, MBRhigh**;
$madr_{A1}$ **write;**
$madr_{A2}$ **upSPout**, **DSout, ldMAR**;
$madr_{A3}$ **PSWout**, **mxMBR**, **ldMBR**;
$madr_{A4}$ **write;**
$madr_{A5}$ **BRout**, **ldX**
$madr_{A6}$ **shl, ALUout, DSout, ldX;**
$madr_{A7}$ **IVTPout**, **ldY**;
$madr_{A8}$ **add**, **ALUout, ldMAR, DSout ;**
$madr_{A9}$ **read;**
$madr_{AA}$ **ldMBR**;
$madr_{AB}$ **MBRout**, **ldPChigh, incMAR**;
$madr_{AC}$ **read;**
$madr_{AD}$ **ldMBR**;
$madr_{AE}$ **MBRout**, **ldPClow**;
$madr_{AF}$ **cnt, bruncond, madr**$_{00}$;

Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 6. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti mikroprogramskog brojača*, blok *mikroprogramski brojač*, blok *mikroprogramska memorija*, blok *prihvatni registar mikroinstrukcije* i blok *generisanje upravljačkih signala*.

Blok *generisanje nove vrednosti mikroprogramskog brojača* se sastoji od kombinacionih mreža KMOPR i KMADR sa multiplekserom MP i služi za generisanje i selekciju vrednosti koju treba upisati u mikroprogramski brojač. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikroprograma. Vrednosti koje treba upisati u mikroprogramski brojač generišu se na tri načina i to pomoću: kombinacione mreže KMOPR koja formira signale $\mathbf{opr_{7...0}}$, 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**.



Slika 6 Struktura upravljačke jedinice mikroprogramske realizacije sa horizontalnim formatom mikroinstrukcije

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 20) za realizaciju višestrukog uslovnog skoka na adresi 3B mikroprograma (tabela 21). U zavisnosti od toga koji od signala **MOVS**, **MOVD**, …, **RTS** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 20 da se pojavi na linijama $\mathbf{opr_{7...0}}$. S obzirom da se na adresi 3B 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 19) za realizaciju višestrukog uslovnog skoka na adresi 19 mikroprograma (tabela 21). U zavisnosti od toga koji od signala **dirreg**, **indreg**,..., **immed** ima aktivnu vrednost zavisi koja će od vrednosti iz tabele 19 da se pojavi tada na linijama $\mathbf{adr_{7...0}}$. S obzirom da se na adresi 19 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 17) i uslovne skokove (tabela 18). Signali višestrukih uslovnin skokova **bropr** i **bradr** su aktivni samo prilikom izvršavanja mikroinstrukcija na adresama 19 i 3B 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 21). 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 21). 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 21). S obzirom da se mikroprogram svih faza izvršavanja instrukcija nalazi u opsegu od adrese 00 do adrese AF, 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 64 bita, a kapacitet veličinom mikroprograma svih instrukcija procesora (tabela 21) i iznosi 256 lokacija. Adresiranje mikroprogramske memorije se realizuje sadržajem mikroprogramskog brojača $mPC_{n-1...0}$.

Blok *prihvatni registar mikroinstrukcije* sadrži prihvatni registar mikroinstrukcije $CW_{0...k+n}$. Prihvatni registar mikroinstrukcije $CW_{0...k+n}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...m}$ i $CW_{0...k}$ se koriste u bloku *generisanje upravljačkih*

signala za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{k+1...k+n}$ koriste u bloku *generisanje nove vrednosti mikroprogramskog brojača* kao adresa skoka u mikrorogramu u slučaju bezuslovnih i uslovnih skokova. Upis u ovaj registar se realizuje signalom takta **CLK**. Signal takta **CLK** kasni za signalom takta $\mathbf{CLK_{mPC}}$ onoliko koliko je potrebno da se pročita sadržaj sa odgovarajuće adrese mikroprogramske memorije.

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

Upravljački signali operacione jedinice se generišu na sledeći način:

- **PCout** = $\overline{\mathbf{CW}_0} \cdot \mathbf{CW_2}$
- **read** = $\overline{\mathbf{CW}_0} \cdot \mathbf{CW_4}$
- **incSP** = $\overline{\mathbf{CW}_0} \cdot \mathbf{CW_{43}}$

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

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr** = $\mathbf{CW_0 \cdot CW_4 \cdot CW_5 \cdot} \overline{\mathrm{CW}_6} \mathbf{\cdot CW_7}$
- **bradr** = $\mathbf{CW_0 \cdot CW_4 \cdot CW_5 \cdot CW_6 \cdot} \overline{\mathrm{CW}_7}$
- **branch** = **bruncnd**
  + **brl1*l1** + **brl2*** $\overline{\mathrm{l2}}$ + **brl3*l3** + **brnotl4*** $\overline{\mathrm{l4}}$ + **brl5*l5** + **brMOVDorPOP*(MOVD+POP)** + **brimmed*immed** + **brZero*Z** + **brIMMEDorREGIND*(immed+regind)** + **brREGDIR*regdir** + **brnotPREKID*** $\overline{\mathrm{PREKID}}$

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

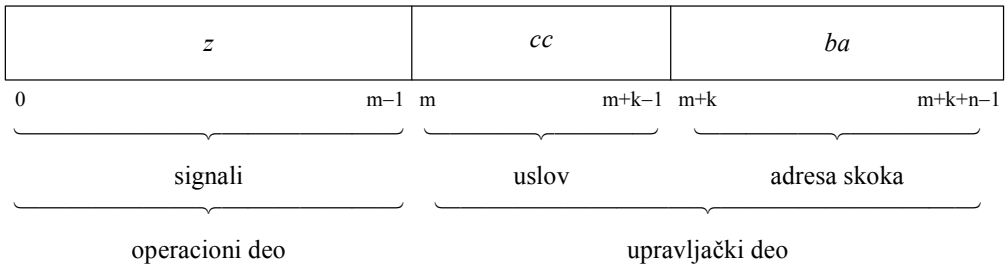
- **bruncond** = $\mathbf{CW_0 \cdot} \overline{\mathrm{CW}_4} \cdot \overline{\mathrm{CW}_5} \cdot \overline{\mathrm{CW}_6} \mathbf{\cdot CW_7}$
- **brl1** = $\mathbf{CW_0 \cdot} \overline{\mathrm{CW}_4} \cdot \overline{\mathrm{CW}_5} \mathbf{\cdot CW_6 \cdot} \overline{\mathrm{CW}_7}$
- **brnotl2** = $\mathbf{CW_0 \cdot} \overline{\mathrm{CW}_4} \cdot \overline{\mathrm{CW}_5} \mathbf{\cdot CW_6 \cdot CW_7}$
- **brl3** = $\mathbf{CW_0 \cdot} \overline{\mathrm{CW}_4} \mathbf{\cdot CW_5 \cdot} \overline{\mathrm{CW}_6} \cdot \overline{\mathrm{CW}_7}$
- **brnotl4** = $\mathbf{CW_0 \cdot} \overline{\mathrm{CW}_4} \mathbf{\cdot CW_5 \cdot} \overline{\mathrm{CW}_6} \mathbf{\cdot CW_7}$
- **brl5** = $\mathbf{CW_0 \cdot} \overline{\mathrm{CW}_4} \mathbf{\cdot CW_5 \cdot CW_6 \cdot} \overline{\mathrm{CW}_7}$
- **brMOVDorPOP** = $\mathbf{CW_0 \cdot} \overline{\mathrm{CW}_4} \mathbf{\cdot CW_5 \cdot CW_6 \cdot CW_7}$
- **brimmed** = $\mathbf{CW_0 \cdot CW_4 \cdot} \overline{\mathrm{CW}_5} \cdot \overline{\mathrm{CW}_6} \cdot \overline{\mathrm{CW}_7}$
- **brZero** = $\mathbf{CW_0 \cdot CW_4 \cdot} \overline{\mathrm{CW}_5} \cdot \overline{\mathrm{CW}_6} \mathbf{\cdot CW_7}$
- **brIMMEDorREGIND** = $\mathbf{CW_0 \cdot CW_4 \cdot} \overline{\mathrm{CW}_5} \mathbf{\cdot CW_6 \cdot} \overline{\mathrm{CW}_7}$
- **brREGDIR** = $\mathbf{CW_0 \cdot CW_4 \cdot} \overline{\mathrm{CW}_5} \mathbf{\cdot CW_6 \cdot CW_7}$

  brnotPREKID = $\mathbf{CW_0 \cdot CW_4 \cdot CW_5 \cdot} \overline{\mathrm{CW}_6} \cdot \overline{\mathrm{CW}_7}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **l3, l4, l5, MOVD, POP, immed, Z, immed, regind, regdir, PREKID**.

4.2.1.2 Mikroprogramska realizacija sa horizontalnim formatom mikroinstrukcija sa jednim tipom instrukcija

U slučaju spajanja koraka postoji samo jedan tip mikroinstrukcije (slika 7).



Slika 7 Format mikroinstrukcije za horizontalni način kodiranja upravljačkih signala

Kodiranje operacionog i upravljačkog dela mikroinstrukcije je dato na slici 8.

0	1	2	3	4	5	6	7
0	resetF	PCout	ldMAR	read	write	ldMBR	incPC

8	9	10	11	12	13	14	15
MBRout	ldIR1	ldIR2	ldIR3	ldIR4	REGout	ldBlow	ldBhigh

16	17	18	19	20	21	22	23
fdo	DSout	SDout	IR3out	ldX	ldY	add	and

24	25	26	27	28	29	30	31
inc	dec	asr	shl	trans	ALUout	IR_DAout	incMAR

32	33	34	35	36	37	38	39
decMAR	Bout	Aout	setCOD	setADR	setINT	daREG	ldREG

40	41	42	43	44	45	46	47
ldPSWALU	IR2out	decSP	incSP	mxMBR	upSPout	MBRhigh	IR_JAout

48	49	50	51	52	53	54	55
ldPChigh	ldPClow	MARout	ldA	setI	resetI	setT	reset

56	57	58	59	60	61	62	63
fvo	intack	PSWout	ldPSW	BRout	IVTPout	ldBR	/

64	65	66	67	68	69	70	71
/	/	/	/	*cc*			

72	73	74	75	76	77	78	79
ba							

Slika 8 Mikroinstrukcija

Mikroprogram je dat u tabeli 22 .

Tabela 22 Mikroprogram (jedan tip mikroinstrukcije)

! Čitanje instrukcije !

$madr_{00}$ **resetF, PCout**, **ldMAR**;
$madr_{01}$ **read;**
$madr_{02}$ **ldMBR**, **incPC**;
$madr_{03}$ **MBRout**, **ldIR1**, **PCout**, **ldMAR;**

madr$_{04}$ **brl1, madr**$_{0C}$;
madr$_{05}$ **brnotl2, madr**$_{31}$;
madr$_{06}$ **read;**
madr$_{07}$ **ldMBR**, **incPC**;
madr$_{08}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR, brl3, madr**$_{31}$;
madr$_{09}$ **read;**
madr$_{0A}$ **ldMBR**, **incPC**;
madr$_{0B}$ **MBRout**, **ldIR3, bruncond, madr**$_{31}$;

madr$_{0C}$ **read;**
madr$_{0D}$ **ldMBR**, **incPC**;
madr$_{0E}$ **MBRout**, **ldIR2**, **PCout**, **ldMAR**;
madr$_{0F}$ **cnt, brnotl4, madr**$_{16}$;
madr$_{10}$ **read;**
madr$_{11}$ **ldMBR**, **incPC**;
madr$_{12}$ **MBRout**, **ldIR3**, **PCout**, **ldMAR, brl5, madr**$_{16}$;
madr$_{13}$ **read;**
madr$_{14}$ **ldMBR**, **incPC**;
madr$_{15}$ **MBRout**, **ldIR4;**

! Formiranje adrese i čitanje operanda !

madr$_{16}$ **bradr;**
madr$_{17}$ **bruncond, madr**$_{7D}$;
! Direktno registarsko !
madr$_{18}$ **REGout, ldBlow, ldBhigh, fdo, bruncond, madr**$_{31}$;
! Indirektno registarsko !
madr$_{19}$ **REGout**,**DSout, ldMAR, fdo, bruncond, madr**$_{2A}$;
! Indirektno registarsko sa pomerajem !
madr$_{1A}$ **REGout, ldX, DSout, fdo**;
madr$_{1B}$ **IR3out**, **ldY**;
madr$_{1C}$ **add**, **ldMAR, DSout, ALUout, bruncond, madr**$_{2A}$;
! Direktno memorijsko !
madr$_{1D}$ **IR_DAout**, **ldMAR, bruncond, madr**$_{2A}$;
! Indirektno memorijsko !
madr$_{1E}$ **IR_DAout**, **ldMAR**;
madr$_{1F}$ **read;**
madr$_{20}$ **ldMBR, incMAR**;
madr$_{21}$ **ldBhigh, MBRout**;
madr$_{22}$ **read;**
madr$_{23}$ **ldMBR;**
madr$_{24}$ **ldBlow, MBRout**;
madr$_{25}$ **Bout**, **ldMAR, bruncond, madr**$_{2A}$;
! Relativno !
madr$_{26}$ **ldX, PCout**;
madr$_{27}$ **IR3out, ldY**;
madr$_{28}$ **add**, **ALUout, DSout, ldMAR, bruncond, madr**$_{2A}$;
! Neposredno !
madr$_{29}$ **IR_DAout**, **SDout**, **ldBhigh, ldBlow, bruncond, madr**$_{31}$;

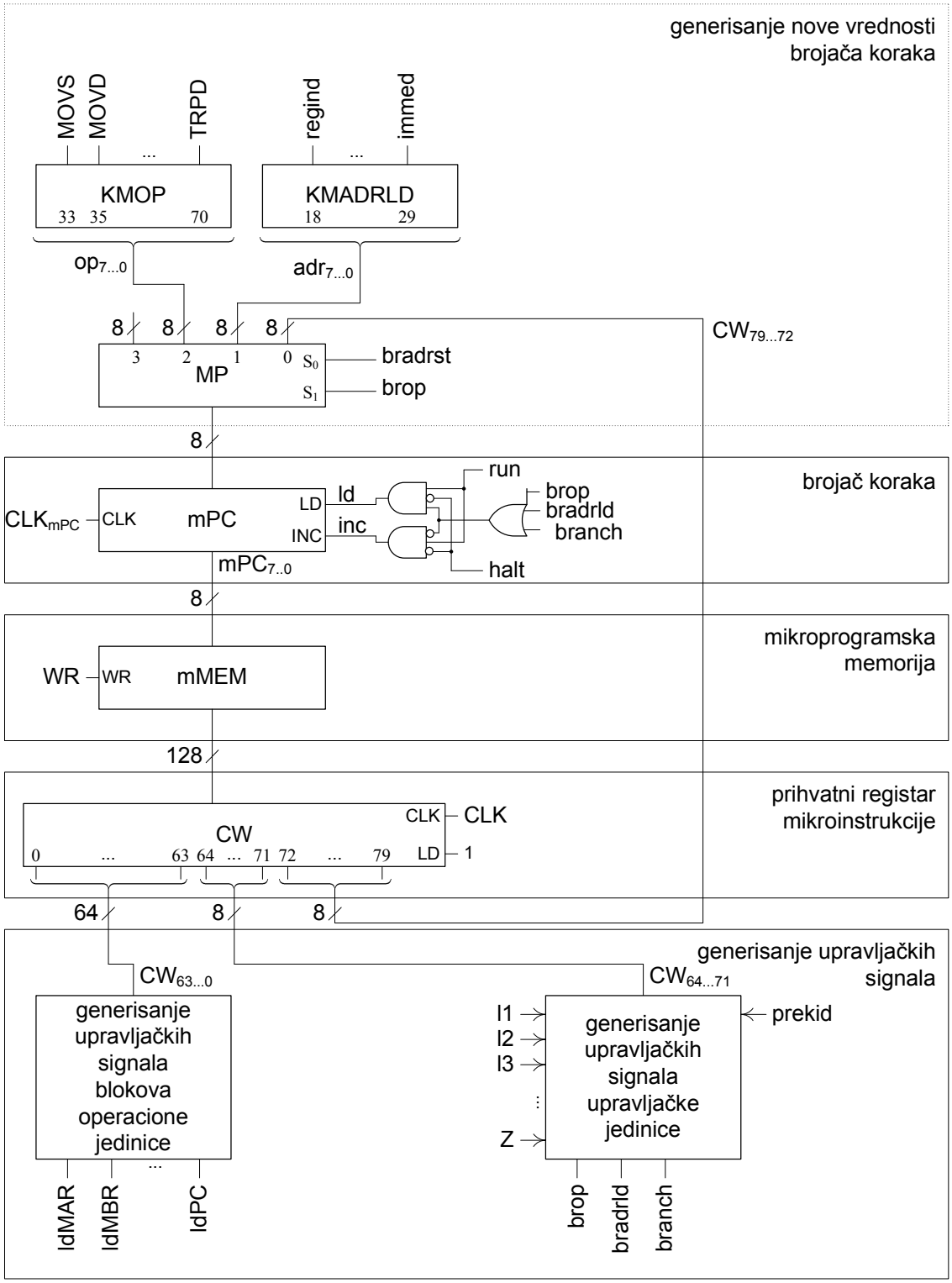
! Čitanje operanda za memorijska adresiranja !
madr$_{2A}$ **brMOVDorPOP, madr**$_{3B}$;
madr$_{2B}$ **read;**
madr$_{2C}$ **ldMBR, incMAR**;
madr$_{2D}$ **MBRout**, **ldBhigh;**

```
        madr2E  read;
        madr2F  ldMBR;
        madr30  MBRout, ldBlow;

! Izvršavanje operacije !
        madr31  bropr;

! Kod operacije !
        madr32  setCOD, bruncond, madr7E;
! MOVS !
        madr33  Bout, daREG, ldREG, ldX;
        madr34  trans, ldPSWALU, bruncond, madr7E;
! MOVD !
        madr35  brimmed, madr7D;
        madr36  REGout, daREG, ldBlow, ldBhigh, DSout, ldX;
        madr37  trans, ldPSWALU, bruncond, madr76;
! ADD !
        madr38  REGout, daREG, DSout, ldX;
        madr39  Bout, ldY;
        madr3A  add, ALUout, ldREG, daREG, ldPSWALU, DSout, bruncond, madr7E;
! AND !
        madr3B  REGout, daREG, DSout, ldX;
        madr3C  Bout, ldY;
        madr3D  and, ALUout, ldREG, daREG, ldPSWALU, DSout, bruncond, madr7E;
! ASR !
        madr3E  brimmed, madr7D;
        madr3F  Bout, ldX;
        madr40  asr, ALUout, ldBhigh, ldBlow, ldPSWALU, bruncond, madr76;
! BNZ !
        madr41  brZero, madr7E;
        madr42  ldX, PCout;
        madr43  ldY, IR2out;
        madr44  add, ALUout, ldPChigh, ldPClow, bruncond, madr7E;
! JSR !
        madr45  decSP, mxMBR, ldMBR, PCout;
        madr46  upSPout, decSP, DSout, ldMAR;
        madr47  write;
        madr48  upSPout, DSout, ldMAR;
        madr49  MBRhigh, mxMBR, ldMBR, PCout;
        madr4A  write;
! JMP !
        madr4B  IR_JAout, SDout, ldPChigh, ldPClow, bruncond, madr7E;
! JMPIND !
        madr4C  brIMMEDorREGIND, madr7D;
        madr4D  Bout, SDout, ldPChigh, ldPClow, bruncond, madr7E;
! RTI !
        madr4E  upSPout, DSout, ldMAR, incSP;
        madr4F  read;
        madr50  ldMBR;
        madr51  MBRout, ldPSW;
! RTS !
        madr52  upSPout, DSout, ldMAR, incSP;
        madr53  read;
        madr54  ldMBR, upSPout, DSout, ldMAR incSP;
```

```
        madr55  MBRout, ldPChigh;
        madr56  read;
        madr57  ldMBR;
        madr58  MBRout, ldPClow, bruncond, madr7E;
! INT !
        madr59  setINT, bruncond, madr7E;
! PUSH !
        madr5A  mxMBR, ldMBR, decSP, Bout;
        madr5B  ldMAR, upSPout, DSout, decSP;
        madr5C  write;
        madr5D  ldMAR, upSPout, DSout;
        madr5E  mxMBR, MBRhigh, Bout, ldMBR;
        madr5F  write;
        madr60  bruncond, madr7E;
! POP !
        madr61  brimmed, madr7D;
        madr62  MARout, ldA;
        madr63  ldMAR, upSPout, DSout, incSP;
        madr64  read;
        madr65  ldMBR, ldMAR, upSPout, DSout,  incSP;
        madr66  MBRout, ldBhigh;
        madr67  read;
        madr68  ldMBR, ldMAR, Aout;
        madr69  MBRout, ldBlow;
        madr6A  Bout, ldX;
        madr6B  trans, ldPSWALU, bruncond, madr76;
! INC !
        madr6C  brimmed, madr7D;
        madr6D  Bout, ldX;
        madr6E  inc, ALUout, ldBhigh, ldBlow, ldPSWALU, bruncond, madr76;
! DEC !
        madr6F  brimmed, madr7D;
        madr70  Bout, ldX;
        madr71  dec, ALUout, ldBhigh, ldBlow, ldPSWALU, bruncond, madr76;
! INTE !
        madr72  setI, bruncond, madr7E;
! INTD !
        madr73  resetI, bruncond, madr7E;
! TRPE !
        madr74  setT, bruncond, madr7E;
! TRPD !
        madr75  resetT, bruncond, madr7E;

! Vraćanje podatka !
        madr76  brregdir, madr7C;
        madr77  Bout, ldMBR, mxMBR;
        madr78  write;
        madr79  decMAR Bout, ldMBR, mxMBR, MBRhigh;
        madr7A  write;
        madr7B  bruncond, madr7E;
        madr7C  Bout, ldREG, fvo, bruncond, madr7E;

! Opsluživanje prekida !

        madr7D  setADR;
```

$madr_{7E}$ **brnotPREKID, madr**$_{00}$;
$madr_{7F}$ **decSP, PCout**, **mxMBR**, **ldMBR, ldBR, intack**;
$madr_{80}$ **upSPout**, **DSout, ldMAR, decSP**;
$madr_{81}$ **write;**
$madr_{82}$ **upSPout**, **DSout, ldMAR**, **decSP**;
$madr_{83}$ **PCout**, **mxMBR**, **ldMBR, MBRhigh**;
$madr_{84}$ **write;**
$madr_{85}$ **upSPout**, **DSout, ldMAR**;
$madr_{86}$ **PSWout**, **mxMBR**, **ldMBR**;
$madr_{87}$ **write;**
$madr_{88}$ **BRout**, **ldX**
$madr_{89}$ **shl, ALUout, DSout, ldX;**
$madr_{8A}$ **IVTPout**, **ldY**;
$madr_{8B}$ **add**, **ALUout, ldMAR, DSout ;**
$madr_{8C}$ **read;**
$madr_{8D}$ **ldMBR**;
$madr_{8E}$ **MBRout**, **ldPChigh, incMAR**;
$madr_{8F}$ **read;**
$madr_{90}$ **ldMBR**;
$madr_{91}$ **MBRout**, **ldPClow, bruncond, madr**$_{00}$;

Struktura upravljačke jedinice je data na slici 9.



Slika 9 Struktura upravljačke jedinice sa jednim tipom mikroinstrukcije

Upravljački signali operacione jedinice se generišu na sledeći način:

- **PCout** = $\mathbf{CW_2}$
- **read** = $\mathbf{CW_4}$
- **incSP** = $\mathbf{CW_{43}}$

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

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bropr** = $\mathbf{CW_{68} \cdot CW_{69} \cdot \overline{CW}_{70} \cdot CW_{71}}$
- **bradr** = $\mathbf{CW_{68} \cdot CW_{69} \cdot CW_{70} \cdot \overline{CW}_{71}}$
- **branch** = **bruncnd**
  + brl1*l1 + brl2* $\overline{\text{l2}}$ **+ brl3*l3 + brnotl4*** $\overline{\text{l4}}$ **+ brl5*l5 +**
  brMOVDorPOP*(MOVD+POP) + brimmed*immed + brZero*Z +
  brIMMEDorREGIND*(immed+regind) + brREGDIR*regdir +
  brnotPREKID* $\overline{\text{PREKID}}$

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

- **bruncond** = $\mathbf{\overline{CW}_{68} \cdot \overline{CW}_{69} \cdot \overline{CW}_{70} \cdot CW_{71}}$
- **brl1** = $\mathbf{\overline{CW}_{68} \cdot \overline{CW}_{69} \cdot CW_{70} \cdot \overline{CW}_{71}}$
- **brnotl2** = $\mathbf{\overline{CW}_{68} \cdot \overline{CW}_{69} \cdot CW_{70} \cdot CW_{71}}$
- **brl3** = $\mathbf{\overline{CW}_{68} \cdot CW_{69} \cdot \overline{CW}_{70} \cdot \overline{CW}_{71}}$
- **brnotl4** = $\mathbf{\overline{CW}_{68} \cdot CW_{69} \cdot \overline{CW}_{70} \cdot CW_{71}}$
- **brl5** = $\mathbf{\overline{CW}_{68} \cdot CW_{69} \cdot CW_{70} \cdot \overline{CW}_{71}}$
- **brMOVDorPOP** = $\mathbf{\overline{CW}_{68} \cdot CW_{69} \cdot CW_{70} \cdot CW_{71}}$
- **brimmed** = $\mathbf{CW_{68} \cdot \overline{CW}_{69} \cdot \overline{CW}_{70} \cdot \overline{CW}_{71}}$
- **brZero** = $\mathbf{CW_{68} \cdot \overline{CW}_{69} \cdot \overline{CW}_{70} \cdot CW_{71}}$
- **brIMMEDorREGIND** = $\mathbf{CW_{68} \cdot \overline{CW}_{69} \cdot CW_{70} \cdot \overline{CW}_{71}}$
- **brREGDIR** = $\mathbf{CW_{68} \cdot \overline{CW}_{69} \cdot CW_{70} \cdot CW_{71}}$
- **brnotPREKID** = $\mathbf{CW_{68} \cdot CW_{69} \cdot \overline{CW}_{70} \cdot \overline{CW}_{71}}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **l3, l4, l5, MOVD, POP, immed, Z, immed, regind, regdir, PREKID**.

4.2.2 Mikroprogramska realizacija sa vertikalnim formatom mikroinstrukcija

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

Kombinacije upravljačkih signala operacione jedinice i simboličke oznake usvojenih kodova dati su u tabeli 23. Usvojeni kodovi su simbolički označeni sa V_{xx}, pri čemu xx odgovara heksadekadnoj vrednosti usvojenog koda. Na primer, sa V_{00} je simbolički označena heksadekadna vrednost 00 ovog polja. Svakom usvojenom kodu odgovara neka kombinacija upravljačkih signala. Na primer, kodu V_{01} odgovarju signali **resetF, PCout** i **ldMAR**, kodu V_{01} signal **read**, itd. Kombinacije upravljačkih signala su tako odabrane da njima mogu da se pokriju sve situacije iz sekvence upravljačkih signala po koracima (tabele 1 i 2). Iz tabele 23 se vidi da je za kodiranje kombinacija upravljačkih signala operacione jedinice potrebno 70 kodova, pa je za kodiranje polja *kombinacija upravljačkih signala* operacionih mikroinstrukcija dovoljno 7 bitova.

Tabela 23 Kombinacije upravljačkih signala operacione jedinice i simboličke oznake kodova

Kombinacija signala	Oznaka koda
/	V_{00}
resetF, PCout, **ldMAR**	V_{01}
read	V_{02}
ldMBR, incPC	V_{03}
MBRout, ldIR1, PCout, ldMAR	V_{04}
MBRout, ldIR2, PCout, ldMAR	V_{05}
MBRout, **ldIR3**	V_{06}
MBRout, **ldIR3**, **PCout**, **ldMAR**	V_{07}
MBRout, **ldIR4**	V_{08}
REGout, ldBlow, ldBhigh, fdo	V_{09}
REGout,**DSout, ldMAR, fdo**	V_{0A}
REGout, ldX, DSout, fdo	V_{0B}
IR3out, **ldY**	V_{0C}
add, **ldMAR, DSout, ALUout**	V_{0D}
IR_DAout, **ldMAR**	V_{0E}
ldMBR, incMAR	V_{0F}
ldBhigh, MBRout	V_{10}
ldMBR	V_{11}
ldBlow, MBRout	V_{12}
Bout, **ldMAR**	V_{13}
ldX, PCout	V_{14}
IR_DAout, **SDout**, **ldBhigh, ldBlow**	V_{15}
setCOD	V_{16}
Bout, **daREG, ldREG, ldX**	V_{17}
trans, ldPSWALU	V_{18}
REGout, **daREG, ldBlow, ldBhigh, DSout, ldX**	V_{19}
REGout, daREG, DSout, ldX	V_{1A}
Bout, **ldY**	V_{1B}
add, **ALUout, ldREG, daREG, ldPSWALU, DSout**	V_{1C}
and, **ALUout, ldREG, daREG, ldPSWALU, DSout**	V_{1D}
Bout, ldX	V_{1E}
asr, ALUout, ldBhigh, ldBlow, ldPSWALU	V_{1F}
ldY, IR2out	V_{20}
add, ALUout, **ldPChigh, ldPClow**	V_{21}
decSP, mxMBR, **ldMBR, PCout**;	V_{22}
upSPout, **decSP, DSout**, **ldMAR**	V_{23}
write	V_{24}
upSPout, **DSout, ldMAR**	V_{25}
MBRhigh, mxMBR, **ldMBR, PCout**	V_{26}
IR_JAout, **SDout**, **ldPChigh, ldPClow**	V_{27}
Bout, **SDout**, **ldPChigh, ldPClow**	V_{28}
upSPout, **DSout, ldMAR, incSP**	V_{29}
MBRout, **ldPSW**	V_{2A}
ldMBR, upSPout, **DSout, ldMAR, incSP**	V_{2B}
MBRout, **ldPChigh**	V_{2C}

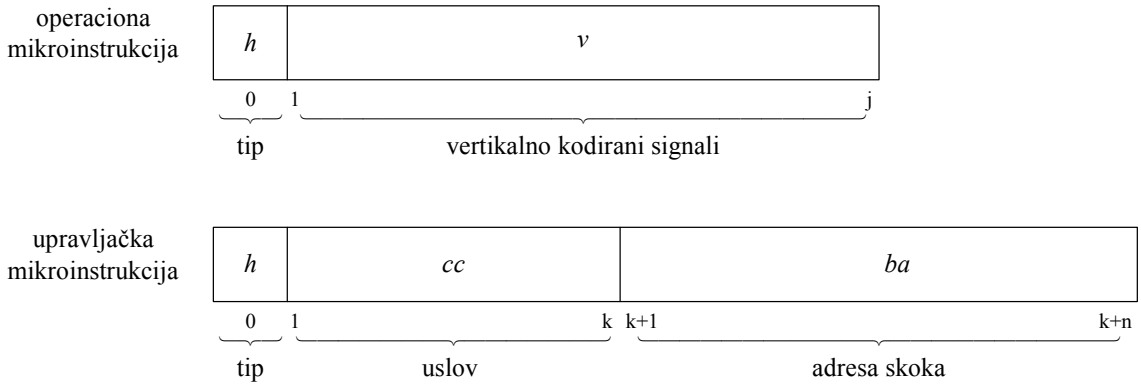
MBRout, **ldPClow**	V_{2D}
setINT	V_{2E}
mxMBR, **ldMBR, decSP, Bout**	V_{2F}
ldMAR, upSPout, DSout, decSP	V_{30}
ldMAR, upSPout, DSout	V_{31}
mxMBR, MBRhigh, Bout, ldMBR	V_{32}
MARout, **ldA**	V_{33}
ldMBR, ldMAR, Aout	V_{34}
inc, ALUout, ldBhigh, ldBlow, ldPSWALU	V_{35}
dec, ALUout, ldBhigh, ldBlow, ldPSWALU	V_{36}
setI	V_{37}
resetI	V_{38}
setT	V_{39}
resetT	V_{3A}
Bout, **ldMBR, mxMBR**	V_{3B}
decMAR, Bout, **ldMBR, mxMBR, MBRhigh**	V_{3C}
Bout, **ldREG, fvo**	V_{3D}
setADR	V_{3E}
decSP, PCout, **mxMBR**, **ldMBR, ldBR, intack**	V_{3F}
PCout, **mxMBR**, **ldMBR, MBRhigh**	V_{40}
PSWout, **mxMBR**, **ldMBR**	V_{41}
BRout, **ldX**	V_{42}
shl, ALUout, DSout, ldX	V_{43}
IVTPout, **ldY**	V_{44}
add, **ALUout, ldMAR, DSout**	V_{45}
MBRout, **ldPChigh, incMAR**	V_{46}

Kodom V_{00} se definiše da upravljačka jedinica ne generiše ni jedan signal čime se u operacionoj jedinici ne realizacije ni jedna mikrooperacija. Kodom V_{01} se definiše da upravljačka jedinica generiše signale **resetF, PCout** i **ldMAR** čime se u operacionoj jedinici realizuje mikrooperacija upisa sadržaja registra PC u registar MAR (slika 2). Kodom V_{13} se definiše da upravljačka jedinica generiše signale **Bout** i **ldMAR** čime se u operacionoj jedinici realizacije mikrooperacija upisa sadržaja registra B u registar MAR (slika 2). Na sličan način se odgovarajućim kodovima specificiraju i sve ostale mikrooperacije upisa u registre operacione jedinice. Kodom V_{24} se definiše da upravljačka jedinica generiše signale **write** čime se u operacionoj jedinici realizacije mikrooperacija upisa sadržaja registra MBR u memorijsku lokaciju adresiranu sadržajem registra MAR. Kodom V_{29} se definiše da upravljačka jedinica generiše signale **add**, **ALUout, ldREG, daREG, ldPSWALU** i **DSout** čime se u operacionoj jedinici realizacije mikrooperacija sabiranja sadržaja registara X i Y u ALU i upis rezultata u registar naveden u instrukciji. Na sličan način se odgovarajućim kodovima specificiraju i sve ostale aritmetičke, logičke i pomeračke mikrooperacije u ALU i upis rezultata u odgovarajuci registar.

U slučaju vertikalnog formata mikroinstrukcija u jednom koraku se omogućava izvršavanje samo jedne mikrooperacije. Pošto je u ovom slučaju tabela mikrooperacija obezbeđuje sve mikrooperacije iz ranije navedene sekvence upravljačkih signala nema potreba za izmenom iste (tabela 1).

Formati mikroinstrukcija su dati na slici 10. Polja *h*, *cc* i *ba* imaju isto značenje kao i u slučaju upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem

upravljačkih signala operacione jedinice (slika 3), pri čemu su i kodovi polja *cc* za bezuslovne, uslovne i višestruke uslovne skokove isti (tabele 14, 15 i 16). Razlika je samo u poljima *z* i *v* operacionih mikroinstrukcija. U slučaju horizontalnog kodiranja upravljačkih signala operacione jedinice poseban bit polja *z* je dodeljen svakom signalu operacione jedinice, dok su u slučaju vertikalnog kodiranja upravljačkih signala operacione jedinice u polju *v* pojavljuju kodovi koji određuju kombinacije aktivnih vrednosti upravljačkih signala operacione jedinice neophodnih da se u jednom koraku realizuje jedna mikrooperacija. Kodovi polja *v* su dati u tabeli 23.



Slika 10 Formati operacionih i upravljačkih mikroinstrukcija za vertikalno kodiranje upravljačkih signala

U slučaju razmatrane operacione jedinice sa dve interne magistrale formati operacione i upravljačke mikroinstrukcije su dati na slikama 11 i 12. Format operacione mikroinstrukcije je dat na slici 4. Polje *h* je 0. Format upravljačke mikroinstrukcije je dat na slici 5. Polje *h* je 1.

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

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

Slika 11 Operaciona mikroinstrukcija

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

8	9	10	11	12	13	14	15
ba							

Slika 12 Upravljačka mikroinstrukcija

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

Kod formiranja operacionih mikroinstrukcija polazi se od sekvence upravljačkih signala po koracima za upravljačku jedinicu sa vertikalnim formatom mikroinstrukcija i traže koraci u kojima se javljaju upravljački signali operacione jedinice. Za svaki takav korak formira se jedna operaciona mikroinstrukcija koja, saglasno kombinaciji upravljačkih signala operacione jedinice koja se javlja u datom koraku i tabeli 23, dobija odgovarajuću vrednost za polje *kombinacija upravljačkih signala*. Na primer, za korake $step_{00}$, $step_{01}$ i $step_{03}$

$step_{00}$ **resetF, PCout**, **ldMAR**;
$step_{01}$ **read**;

$step_{02}$ **ldMBR, incPC**;

formiraju se tri operacione mikroinstrukcije

$madr_{00}$ V_{01}; ! **ldMAR** !
$madr_{01}$ V_{02}; ! **incPC** !
$madr_{02}$ V_{03}; ! **ldMBR** !

sa vrednostima polja *kombinacija upravljačkih signala* V_{01}, $V0_2$ i V_{03}, respektivno.

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

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 1) formiran mikroprogram (tabela 24). On ima sledeću formu:

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

Tabela 24 Mikroprogram

! Čitanje instrukcije !

$madr_{00}$	V_{01};	**! resetF, PCout**, **ldMAR !**
$madr_{01}$	V_{02};**! read !**	
$madr_{02}$	V_{03};	**! ldMBR, incPC !**
$madr_{03}$	V_{04};	**! MBRout, ldIR1, PCout, ldMAR !**
$madr_{04}$	**cnt, brl1, madr**$_{0E}$	
$madr_{05}$	**cnt, brnotl2, madr**$_{3B}$	
$madr_{06}$	V_{02};	**! read !**
$madr_{07}$	V_{03};	**! ldMBR, incPC !**
$madr_{08}$	V_{05}**;**	**! MBRout, ldIR2, PCout, ldMAR !**
$madr_{09}$	**cnt, brl3, madr**$_{3B}$	
$madr_{0A}$	V_{02};	**! read !**
$madr_{0B}$	V_{03};	**! ldMBR, incPC !**
$madr_{0C}$	V_{06};	**! MBRout**, **ldIR3 !**
$madr_{0D}$	**cnt, bruncond, madr**$_{3B}$;	
$madr_{0E}$	V_{02};	**! read !**
$madr_{0F}$	V_{03};	**! ldMBR, incPC !**
$madr_{10}$	V_{05};	**! MBRout**, **ldIR2**, **PCout**, **ldMAR !**
$madr_{11}$	**cnt, brnotl4, madr**$_{19}$**;**	
$madr_{12}$	V_{02};	**! read !**
$madr_{13}$	V_{03};	**! ldMBR, incPC !**
$madr_{14}$	V_{07};	**! MBRout**, **ldIR3**, **PCout**, **ldMAR !**
$madr_{15}$	**cnt, brl5, madr**$_{19}$;	
$madr_{16}$	V_{02};	**! read !**
$madr_{17}$	V_{03};	**! ldMBR, incPC !**
$madr_{18}$	V_{08};	**! MBRout**, **ldIR4 !**

```
! Formiranje adrese i čitanje operanda !
        madr19  cnt, bradr;
        madr1A  cnt, bruncond, madr9A;
! Direktno registarsko !
        madr1B  V09;                        ! REGout, ldBlow, ldBhigh, fdo !
        madr1C  cnt, bruncond, madr3B;
! Indirektno registarsko !
        madr1D  V0A;                        ! REGout,DSout, ldMAR, fdo !
        madr1E  cnt, bruncond, madr34;
! Indirektno registarsko sa pomerajem !
        madr1F  V0B;                        ! REGout, ldX, DSout, fdo !
        madr20  V0C;                        ! IR3out, ldY !
        madr21  V0D;                        ! add, ldMAR, DSout, ALUout !
        madr22  cnt, bruncond, madr34;
! Direktno memorijsko !
        madr23  V0E;                        ! IR_DAout, ldMAR !
        madr24  cnt, bruncond, madr34;
! Indirektno memorijsko !
        madr25  V0E;                        ! IR_DAout, ldMAR !
        madr26  V02;                        ! read !
        madr27  V0F;                        ! ldMBR, incMAR !
        madr28  V10;                        ! ldBhigh, MBRout !
        madr29  V02;                        ! read !
        madr2A  V11;                        ! ldMBR !
        madr2B  V12;                        ! ldBlow, MBRout !
        madr2C  V13;                        ! Bout, ldMAR !
        madr2D  cnt, bruncond, madr34;
! Relativno !
        madr2E  V14;                        ! ldX, PCout !
        madr2F  V0C;                        ! IR3out, ldY !
        madr30  V0D;                        ! add, ALUout, DSout, ldMAR !
        madr31  cnt, bruncond, madr34;
! Neposredno !
        madr32  V15;                        ! IR _DAout, SDout, ldBhigh, ldBlow !
        madr33  cnt, bruncond, madr3B;
! Čitanje operanda za memorijska adresiranja !
        madr34  cnt, brMOVDorPOP, madr3B;
        madr35  V02;                        ! read !
        madr36  V0F;                        ! ldMBR, incMAR !
        madr37  V10;                        ! MBRout, ldBhigh !
        madr38  V02;                        ! read !
        madr39  V11;                        ! ldMBR !
        madr3A  V12;                        ! MBRout, ldBlow !
! Izvršavanje operacije !
        madr3B  cnt, bropr;

        madr3C  V16;                        ! setCOD !
        madr3D  cnt, bruncond, madr9B;

! MOVS !
        madr3E  V17;                        ! Bout, daREG, ldREG, ldX !
        madr3F  V18;                        ! trans, ldPSWALU !
        madr40  cnt, bruncond, madr9B;
```

```
! MOVD !
        madr41  cnt, brimmed, madr9A;
        madr42  V19;                        ! REGout, daREG, ldBlow, ldBhigh, DSout, ldX !
        madr43  V18;                        ! trans, ldPSWALU !
        madr44  cnt, bruncond, madr92;
! ADD !
        madr45  V1A;                        ! REGout, daREG, DSout, ldX !
        madr46  V1B;                        ! Bout, ldY !
        madr47  V1C;                  ! add, ALUout, ldREG, daREG, ldPSWALU, DSout !
        madr48  cnt, bruncond, madr9B;
! AND !
        madr49  V1A;                        ! REGout, daREG, DSout, ldX !
        madr4A  V1B;                        ! Bout, ldY !
        madr4B  V1D;                  ! and, ALUout, ldREG, daREG, ldPSWALU, DSout !
        madr4C  cnt, bruncond, madr9B;
! ASR !
        madr4D  cnt, brimmed, madr9A;
        madr4E  V1E;                        ! Bout, ldX !
        madr4F  V1F;                        ! asr, ALUout, ldBhigh, ldBlow, ldPSWALU !
        madr50  cnt, bruncond, madr92;
! BNZ!
        madr51  cnt, brZero, madr9B;
        madr52  V14;                        ! ldX, PCout !
        madr53  V20;                        ! ldY, IR2out !
        madr54  V21;                        ! add, ALUout, ldPChigh, ldPClow !
        madr55  cnt, bruncond, madr9B;
! JSR !
        madr56  V22;                        ! decSP, mxMBR, ldMBR, PCout !
        madr57  V23;                        ! upSPout, decSP, DSout, ldMAR !
        madr58  V24                         ! write !
        madr59  V25;                        ! upSPout, DSout, ldMAR !
        madr5A  V26;                        ! MBRhigh, mxMBR, ldMBR, PCout !
        madr5B  V24;                        ! write !
! JMP !
        madr5C  V27;                        ! IR_JAout, SDout, ldPChigh, ldPClow !
        madr5D  cnt, bruncond, madr9B;
! JMPIND !
        madr5E  cnt, brIMMEDorREGIND, madr9A;
        madr5F  V28;                        ! Bout, SDout, ldPChigh, ldPClow !
        madr60  cnt, bruncond, madr9B;
! RTI !
        madr61  V29;                        ! upSPout, DSout, ldMAR, incSP !
        madr62  V02;                        ! read !
        madr63  V11;                        ! ldMBR !
        madr64  V2A;                        ! MBRout, ldPSW !
! RTS !
        madr65  V29;                        ! upSPout, DSout, ldMAR, incSP !
        madr66  V02;                        ! read !
        madr67  V2B;                        ! ldMBR, upSPout, DSout, ldMAR incSP !
        madr68  V2C;                        ! MBRout, ldPChigh !
        madr69  V02;                              ! read !
        madr6A  V11;                              ! ldMBR !
        madr6B  V2D;                        ! MBRout, ldPClow !
        madr6C  cnt, bruncond, madr9B;
! INT !
```

```
        madr6D  V2E;      ! setINT !
        madr6E  cnt, bruncond, madr9B;
! PUSH !
        madr6F  V2F;                    ! mxMBR, ldMBR, decSP, Bout !
        madr70  V30;                    ! ldMAR, upSPout, DSout, decSP !
        madr71  V24;                    ! write !
        madr72  V31;                    ! ldMAR, upSPout, DSout !
        madr73  V32;                    ! mxMBR, MBRhigh, Bout, ldMBR !
        madr74  V24;                    ! write !
        madr75  cnt, bruncond, madr9B;
! POP !
        madr76  cnt, brimmed, madr9A;
        madr77  V33;                    ! MARout, ldA !
        madr78  V29;                    ! ldMAR, upSPout, DSout, incSP !
        madr79  V02;                    ! read !
        madr7A  V2b;                    ! ldMBR, ldMAR, upSPout, DSout,  incSP !
        madr7B  V10;                    ! MBRout, ldBhigh !
        madr7C  V30;                    ! read !
        madr7D  V34;                    ! ldMBR, ldMAR, Aout !
        madr7E  V12;                    ! MBRout, ldBlow !
        madr7F  V1E;                    ! Bout, ldX !
        madr80  V18;                    ! trans, ldPSWALU !
        madr81  cnt, bruncond, madr92;
! INC !
        madr82  cnt, brimmed, madr9A;
        madr83  V1E;                    ! Bout, ldX !
        madr84  V35;                    ! inc, ALUout, ldBhigh, ldBlow, ldPSWALU !
        madr85  cnt, bruncond, madr92;
! DEC !
        madr86  cnt, brimmed, madr9A;
        madr87  V1E;                    ! Bout, ldX !
        madr88  V36;                    ! dec, ALUout, ldBhigh, ldBlow, ldPSWALU !
        madr89  cnt, bruncond, madr92;
! INTE !
        madr8A  V37;                    ! setI !
        madr8B  cnt, bruncond, madr9B;
! INTD !
        madr8C  V38;                    ! resetI !
        madr8D  cnt, bruncond, madr9B;
! TRPE !
        madr8E  V39;                    ! setT !
        madr8F  cnt, bruncond, madr9B;
! TRPD !
        madr90  V3A;                    ! resetT !
        madr91  cnt, bruncond, madr9B;
! Vraćanje podatka !
        madr92  cnt, brregdir, madr98;
        madr93  V3B;                    ! Bout, ldMBR, mxMBR !
        madr94  V24;                    ! write !
        madr95  V3C;                    ! decMAR, Bout, ldMBR, mxMBR, MBRhigh !
        madr96  V24;                    ! write !
        madr97  cnt, bruncond, madr9B;
        madr98  V3D;                    ! Bout, ldREG, fvo !
        madr99  cnt, bruncond, madr9B;
```

! Opsluživanje prekida !

$madr_{9A}$ V_{3E}; **! setADR !**

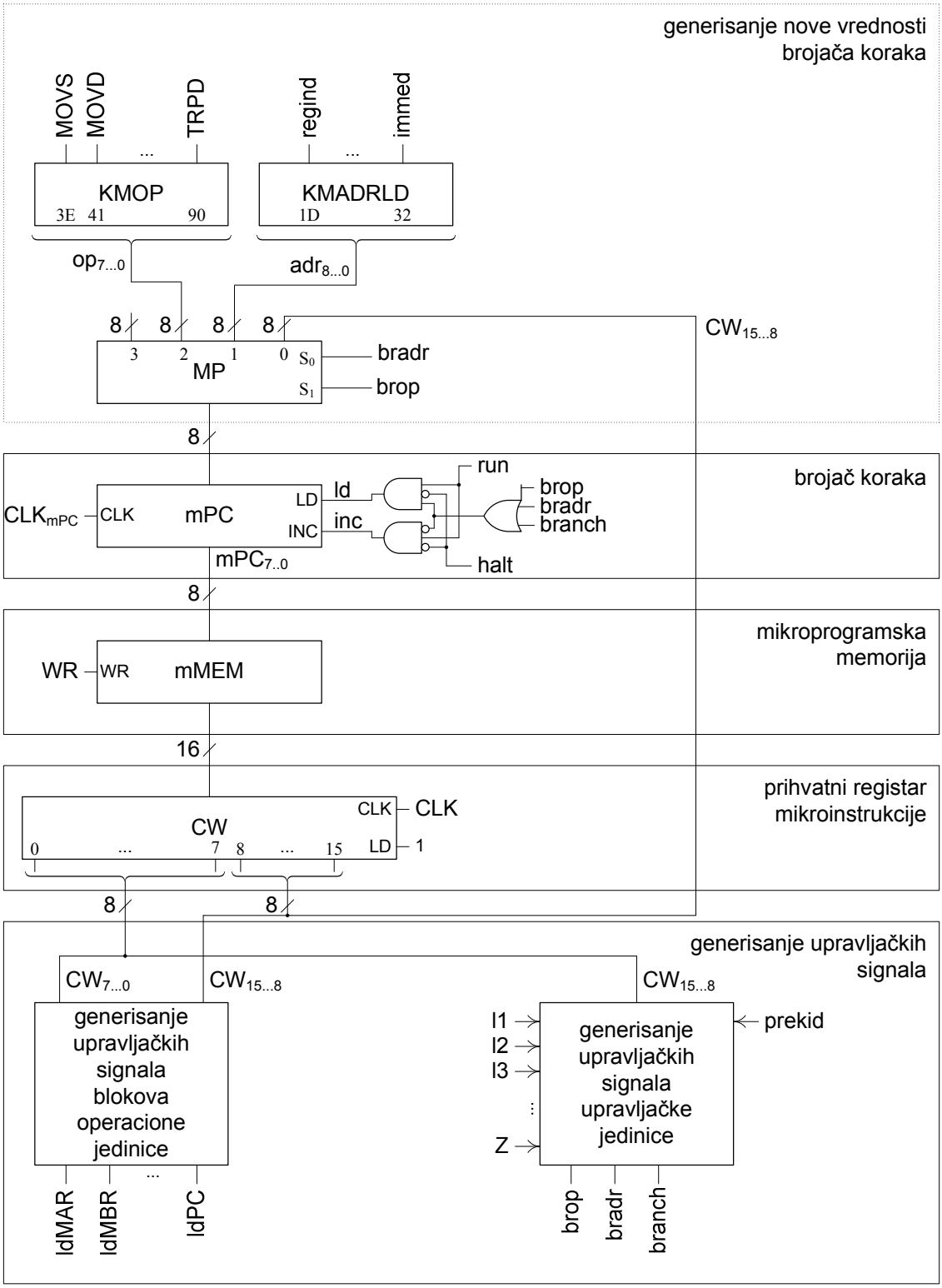
$madr_{9B}$ **cnt, brnotPREKID, madr**$_{00}$;
$madr_{9C}$ V_{3F}; **! decSP, PCout**, **mxMBR**, **ldMBR, ldBR, intack !**
$madr_{9D}$ V_{30}; **! upSPout**, **DSout, ldMAR, decSP !**
$madr_{9E}$ V_{24}**;** **! write !**
$madr_{9F}$ V_{30}; **! upSPout**, **DSout, ldMAR**, **decSP !**
$madr_{A0}$ V_{40}; **! PCout**, **mxMBR**, **ldMBR, MBRhigh !**
$madr_{A1}$ V_{24}**;** **! write !**
$madr_{A2}$ V_{25}; **! upSPout**, **DSout, ldMAR !**
$madr_{A3}$ V_{41}; **! PSWout**, **mxMBR**, **ldMBR !**
$madr_{A4}$ V_{24}**;** **! write !**
$madr_{A5}$ V_{42}**;** **! BRout**, **ldX !**
$madr_{A6}$ V_{43}**;** **! shl, ALUout, DSout, ldX !**
$madr_{A7}$ V_{44}; **! IVTPout**, **ldY !**
$madr_{A8}$ V_{45}**;** **! add**, **ALUout, ldMAR, DSout !**
$madr_{A9}$ V_{02}**;** **! read !**
$madr_{AA}$ V_{11}; **! ldMBR !**
$madr_{AB}$ V_{46}; **! MBRout**, **ldPChigh, incMAR !**
$madr_{AC}$ V_{02}**;** **! read !**
$madr_{AD}$ V_{11}; **! ldMBR !**
$madr_{AE}$ V_{2D}; **! MBRout**, **ldPClow !**
$madr_{AF}$ **cnt, bruncond, madr**$_{00}$;

Struktura upravljačke jedinice je slična strukturi upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice i data je na slici 13. Mikroprogramski brojač se na isti način inkrementira i u mikroprogramski brojač se na isti način upisuje nova vrednost, pri čemu mikroprogramski brojač koraka prolazi kroz veći broj stanja. Druge su jedino vrednosti koje generišu kombinacione mreže KMOPR i KMADR i vrednosti $CW_{k+1...k+n}$ koje se upisuju iz upravljačke mikroinstrukcije. Kombinaciona mreža KMOPR generiše vrednosti 3E, 41, ..., 90 pri aktivnim vrednostima signala **MOVS**, **MOVD**, ..., **TRPD**, respektivno. Kombinaciona mreža KMADR generiše vrednosti 1B, 1D, ..., 32 pri aktivnim vrednostima signala **dirreg**, **indreg**, ..., **immed**, respektivno.

Upravljački signali operacione jedinice se generišu ukoliko je $\mathbf{CW_0} = 0$, jer se tada u prihvatnom registru mikroinstrukcije $CW_{0...k+n}$ nalazi operaciona mikroinstrukcija. Najpre se dekodovanjem na osnovu signala $\mathbf{CW_{0...7}}$ generišu signali kombinacija upravljačkih signala $\mathbf{V_{00}}$ do $\mathbf{V_{46}}$. (tabela 23) i to:

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

Potom, se za svaki upravljački signal operacione jedinice posmatra u kojim kombinacijama upravljačkih signala se signal pojavljuje (tabela 23), pa se dati signal dobija kao njihova unija.



Slika 13 Struktura upravljačke jedinice mikroprogramske realizije sa dva tipa mikroinstrukcija i vertikalnim formatom mikroinstrukcija

Upravljački signali operacione jedinice se generišu na sledeći način:

- **ldMAR** = $\mathbf{V_{01} + V_{04} + V_{05} + V_{07} + V_{0A} + V_{0D} + V_{0E} + V_{13} + V_{23} + V_{25} + V_{29} + V_{2B} + V_{30} + V_{31} + V_{34} + V_{45}}$
- **mxMBR** = $\mathbf{V_{22} + V_{26} + V_{2F} + V_{32} + V_{3B} + V_{3C} + V_{3F} + V_{40} + V_{41}}$

- **write** = $\mathbf{V_{24}}$

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

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

- $\mathbf{bropr = CW_0 \cdot CW_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot CW_7}$
- $\mathbf{bradr = CW_0 \cdot CW_4 \cdot CW_5 \cdot CW_6 \cdot \overline{CW}_7}$
- **branch = branch** = **bruncnd**

  + brl1*l1 + brl2* $\overline{\text{l2}}$ **+ brl3*l3 + brnotl4*** $\overline{\text{l4}}$ **+ brl5*l5 +**
  brMOVDorPOP*(MOVD+POP) + brimmed*immed + brZero*Z +
  brIMMEDorREGIND*(immed+regind) + brREGDIR*regdir +
  brnotPREKID* $\overline{\text{PREKID}}$

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

- $\mathbf{bruncond = CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot \overline{CW}_6 \cdot CW_7}$
- $\mathbf{brl1 = CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot CW_6 \cdot \overline{CW}_7}$
- $\mathbf{brnotl2 = CW_0 \cdot \overline{CW}_4 \cdot \overline{CW}_5 \cdot CW_6 \cdot CW_7}$
- $\mathbf{brl3 = CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot \overline{CW}_7}$
- $\mathbf{brnotl4 = CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot CW_7}$
- $\mathbf{brl5 = CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot CW_6 \cdot \overline{CW}_7}$
- $\mathbf{brMOVDorPOP = CW_0 \cdot \overline{CW}_4 \cdot CW_5 \cdot CW_6 \cdot CW_7}$
- $\mathbf{brimmed = CW_0 \cdot CW_4 \cdot \overline{CW}_5 \cdot \overline{CW}_6 \cdot \overline{CW}_7}$
- $\mathbf{brZero = CW_0 \cdot CW_4 \cdot \overline{CW}_5 \cdot \overline{CW}_6 \cdot CW_7}$
- $\mathbf{brIMMEDorREGIND = CW_0 \cdot CW_4 \cdot \overline{CW}_5 \cdot CW_6 \cdot \overline{CW}_7}$
- $\mathbf{brREGDIR = CW_0 \cdot CW_4 \cdot \overline{CW}_5 \cdot CW_6 \cdot CW_7}$
- $\mathbf{brnotPREKID = CW_0 \cdot CW_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot \overline{CW}_7}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **l3, l4, l5, MOVD, POP, immed, Z, immed, regind, regdir, PREKID**.

4.2.3 Mikroprogramska realizacija sa mešovitim formatom mikroinstrukcija

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

Usvojeno kodiranje upravljačkih signala operacione jedinice je dato u tabeli 25. Upravljački signali operacione jedinice kojih ima 62 grupisani su u osam polja označena sa M1, M2, M3, M4, M5, M6, M7 i M8. Vrednost 0 svih polja se koristi za specificiranje da ni jedan od signala iz date grupe ne treba da bude aktivan. Polja M1,M2, M3, M4, M5, M6 i M7 se koriste za kodiranje po sedam signala, pa je njihova dužina po 3 bita. Polja M8 se koristi za kodiranje trinaest signala, pa je njegova dužina četiri bita. Kombinacije upravljačkih signala su tako odabrane da njima mogu da se pokriju sve situacije iz sekvence upravljačkih signala po koracima (tabele 1 i 2). Ukupan broj bitova potrebnih za kodiranje kombinacija upravljačkih signala operacione jedinice je 21. U odnosu na horizontalan način kodiranja upravljačkih signala operacione jedinice potreban je dvostuko manji broj bitova, uz isti ukupan broja koraka.

Tabela 25 Kodiranje upravljačkih signala operacione jedinice i simboličke oznake kodova

polje M1		polje M2		polje M3		polje M4	
$M1_0$	–	$M2_0$	–	$M3_0$	–	$M4_0$	–
$M1_1$	resetF	$M2_1$	MARout	$M3_1$	ldMAR	$M4_1$	REGout
$M1_2$	read	$M2_2$	ALUout	$M3_2$	ldBlow	$M4_2$	IR_DAout
$M1_3$	write	$M2_3$	PCout	$M3_3$	ldA	$M4_3$	IR_JAout
$M1_4$	MBRout	$M2_4$	Aout	$M3_4$	incMAR	$M4_4$	upSPout
$M1_5$	ldMBR	$M2_5$	incPC	$M3_5$	mxMBR	$M4_5$	ldREG
$M1_6$	daREG	$M2_6$	decMAR	$M3_6$	fvo	$M4_6$	MBRhigh
$M1_7$	setT	$M2_7$	fdo	$M3_7$	resetT	$M4_7$	ldBR

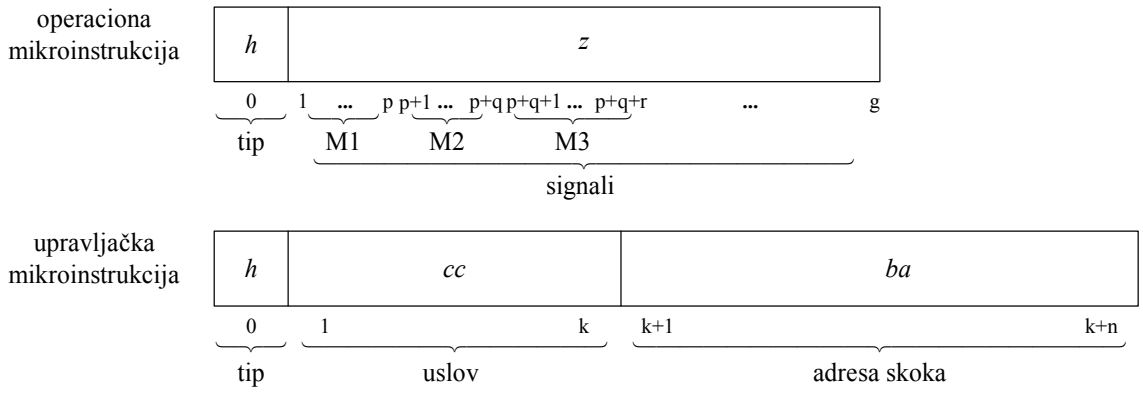
polje M5		polje M6		polje M7	
$M5_0$	–	$M6_0$	–	$M7_0$	–
$M5_1$	ldIR1	$M6_1$	Bout	$M7_1$	ldX
$M5_2$	ldIR2	$M6_2$	PSWout	$M7_2$	ldY
$M5_3$	ldIR3	$M6_3$	BRout	$M7_3$	ldPSW
$M5_4$	ldIR4	$M6_4$	IVTPout	$M7_4$	ldPSWALU
$M5_5$	IR2out	$M6_5$	DSout	$M7_5$	ldPChigh
$M5_6$	IR3out	$M6_6$	SDout	$M7_6$	incSP
$M5_7$	ldBhigh	$M6_7$	intack	$M7_7$	decSP

polje M8	
$M8_0$	–
$M8_1$	ldPClow
$M8_2$	add
$M8_3$	and
$M8_4$	asr
$M8_5$	inc
$M8_6$	dec

$M8_7$	shl
$M8_8$	trans
$M8_9$	setCOD
$M8_A$	setINT
$M8_B$	setADR
$M8_C$	setI
$M8_D$	resetI
$M8_E$	/
$M8_F$	/

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

U slučaju realizacije upravljačke jedinice sa dva tipa mikroinstrukcija i mešovitim kodiranjem upravljačkih signala operacione jedinice, formati mikroinstrukcija su dati na slici 14. Polja *h*, *cc* i *ba* imaju isto značenje kao i u slučaju upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice (slika 3), pri čemu su i kodovi polja *cc* za bezuslovne, uslovne i višestruke uslovne skokove isti (tabele 14, 15 i 16). Razlika je samo u poljima *z* operacionih mikroinstrukcija. U slučaju horizontalnog kodiranja upravljačkih signala operacione jedinice poseban bit polja *z* je dodeljen svakom signalu operacione jedinice, dok je u slučaju mešovitog kodiranja upravljačkih signala operacione jedinice polje *z* dužine g bita podeljeno na onoliko potpolja M1, M2, M3 itd. dužine p, q, r itd. bita, koliko ima grupa upravljačkih signala operacione jedinice. Binarnim vrednostima potpolja kodiraju se signali iz svake od grupa signala.



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

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

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

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

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

24	25	26	27	28	29	30	31
M8	M8	/	/	/	/	/	/

Slika 15 Operaciona mikroinstrukcija

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

8	9	10	11	12	13	14	15
ba							

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

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

Slika 16 Upravljačka mikroinstrukcija

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

Kod formiranja operacionih mikroinstrukcija polazi se od sekvence upravljačkih signala po koracima (tabela 1) i traže koraci u kojima se javljaju upravljački signali operacione jedinice. Za svaki takav korak formira se jedna operaciona mikroinstrukcija tako što se, saglasno kombinaciji upravljačkih signala operacione jedinice koja se javlja u datom koraku i načinu njihovog kodiranja potpoljima M1 do M8 (tabela 25), formiraju vrednosti potpolja M1 do M8. Na primer, za korake $step_{00}$, $step_{01}$ i $step_{02}$

$step_{00}$ **resetF, PCout**, **ldMAR**;
$step_{01}$ **read;**
$step_{02}$ **ldMBR**, **incPC**;

formiraju se tri operacione mikroinstrukcije

$madr_{00}$ $M1_1$, $M2_3$, $M3_1$; ! **resetF, PCout**, **ldMAR**!
$madr_{01}$ $M1_2$; ! **read** !
$madr_{02}$ $M1_5$, $M2_5$; ! **ldMBR, incPC** !

gde u mikroinstrukciji sa adrese 00 potpolja M1,M2 i M3 imaju vrednosti 1,3 i 1, respektivno, a sva ostala potpolja vrednost 0, u mikroinstrukciji sa adrese 01 potpolje M1 ima vrednosti 2, a sva ostala potpolja vrednost 0, u mikroinstrukciji sa adrese 02 potpolja M1 i M2 imaju vrednosti 5, a sva ostala potpolja vrednost 0, itd.

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

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 1), formiran mikroprogram (tabela 26). On ima sledeću formu:

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

Tabela 26 Mikroprogram

! Čitanje instrukcije !		
$madr_{00}$	$M1_1$, $M2_3$, $M3_1$;	**! resetF, PCout**, **ldMAR !**
$madr_{01}$	$M1_2$**;**	**! read !**
$madr_{02}$	$M1_5$**,** $M2_5$;	**! ldMBR, incPC !**
$madr_{03}$	$M1_4$**,** $M5_1$**,** $M2_3$**,** $M3_1$**;**	**! MBRout, ldIR1, PCout, ldMAR !**
$madr_{04}$	**cnt, brl1, madr**$_{0E}$	
$madr_{05}$	**cnt, brnotl2, madr**$_{3B}$	
$madr_{06}$	$M1_2$**;**	**! read !**
$madr_{07}$	$M1_5$**,** $M2_5$	**! ldMBR, incPC !**
$madr_{08}$	$M1_4$**,** $M5_2$**,** $M2_3$**,** $M3_1$**;**	**! MBRout, ldIR2, PCout, ldMAR !**
$madr_{09}$	**cnt, brl3, madr**$_{3B}$	
$madr_{0A}$	$M1_2$**;**	**! read !**
$madr_{0B}$	$M1_5$, $M2_5$;	**! ldMBR, incPC !**
$madr_{0C}$	$M1_4$, $M5_3$;	**! MBRout**, **ldIR3 !**
$madr_{0D}$	**cnt, bruncond, madr**$_{3B}$;	
$madr_{0E}$	$M1_2$**;**	**! read !**
$madr_{0F}$	$M1_5$, $M2_5$;	**! ldMBR, incPC !**
$madr_{10}$	$M1_4$, $M5_2$, $M2_3$, $M3_1$;	**! MBRout**, **ldIR2**, **PCout**, **ldMAR !**
$madr_{11}$	**cnt, brnotl4, madr**$_{19}$**;**	
$madr_{12}$	$M1_2$**;**	**! read !**
$madr_{13}$	$M1_5$, $M2_5$;	**! ldMBR, incPC !**
$madr_{14}$	$M1_4$, $M5_3$, $M2_3$, $M3_1$;	**! MBRout**, **ldIR3**, **PCout**, **ldMAR !**
$madr_{15}$	**cnt, brl5, madr**$_{19}$;	
$madr_{16}$	$M1_2$**;**	**! read !**
$madr_{17}$	$M1_5$, $M2_5$;	**! ldMBR, incPC !**
$madr_{18}$	$M1_4$, $M5_4$**;**	**! MBRout**, **ldIR4 !**
! Formiranje adrese i čitanje operanda !		
$madr_{19}$	**cnt, bradr;**	
$madr_{1A}$	**cnt, bruncond, madr**$_{9A}$;	
! Direktno registarsko !		
$madr_{1B}$	$M4_1$**,** $M3_2$**,** $M5_7$**,** $M2_7$;	**! REGout, ldBlow, ldBhigh, fdo !**
$madr_{1C}$	**cnt, bruncond, madr**$_{3B}$;	
! Indirektno registarsko !		
$madr_{1D}$	$M4_1$,$M6_5$**,** $M3_1$**,** $M2_7$;	**! REGout**,**DSout, ldMAR, fdo !**
$madr_{1E}$	**cnt, bruncond, madr**$_{34}$;	
! Indirektno registarsko sa pomerajem !		
$madr_{1F}$	$M4_1$**,** $M7_1$**,** $M6_5$**,** $M2_7$;	**! REGout, ldX, DSout, fdo !**
$madr_{20}$	$M5_6$, $M7_2$;	**! IR3out**, **ldY !**

```
        madr21  M82, M31, M65, M22;          ! add, ldMAR, DSout, ALUout !
        madr22  cnt, bruncond, madr34;
! Direktno memorijsko !
        madr23  M42, M31;                    ! IR_DAout, ldMAR !
        madr24  cnt, bruncond, madr34;
! Indirektno memorijsko !
        madr25  M42, M31;                    ! IR_DAout, ldMAR !
        madr26  M12;                         ! read !
        madr27  M15, M34;                    ! ldMBR, incMAR !
        madr28  M57, M14;                    ! ldBhigh, MBRout !
        madr29  M12;                         ! read !
        madr2A  M15;                         ! ldMBR !
        madr2B  M32, M14;                    ! ldBlow, MBRout !
        madr2C  M61, M31;                    ! Bout, ldMAR !
        madr2D  cnt, bruncond, madr34;
! Relativno !
        madr2E  M71, M23;                    ! ldX, PCout !
        madr2F  M56, M72;                    ! IR3out, ldY !
        madr30  M82, M22, M65, M31;          ! add, ALUout, DSout, ldMAR !
        madr31  cnt, bruncond, madr34;
! Neposredno !
        madr32  M42, M66, M57, M32;          ! IR _DAout, SDout, ldBhigh, ldBlow !
        madr33  cnt, bruncond, madr3B;
! Čitanje operanda za memorijska adresiranja !
        madr34  cnt, brMOVDorPOP, madr3B;
        madr35  M12;                         ! read !
        madr36  M15, M34;                    ! ldMBR, incMAR !
        madr37  M14, M57;                    ! MBRout, ldBhigh !
        madr38  M12;                         ! read !
        madr39  M15;                         ! ldMBR !
        madr3A  M14, M32;                    ! MBRout, ldBlow !

! Izvršavanje operacije !

        madr3B  cnt, bropr;

        madr3C  M89;                         ! setCOD !
        madr3D  cnt, bruncond, madr9B;

! MOVS !
        madr3E  M61, M16, M45, M71;          ! Bout, daREG, ldREG, ldX !
        madr3F  M88, M74;                    ! trans, ldPSWALU !
        madr40  cnt, bruncond, madr9B;
! MOVD !
        madr41  cnt, brimmed, madr9A;
        madr42  M41, M16, M32, M57, M65, M71;
                    ! REGout, daREG, ldBlow, ldBhigh, DSout, ldX !
        madr43  M88, M74;                    ! trans, ldPSWALU !
        madr44  cnt, bruncond, madr92;
! ADD !
        madr45  M41, M16, M65, M71;          ! REGout, daREG, DSout, ldX !
        madr46  M61, M72;                    ! Bout, ldY !
        madr47  M82, M22, M45, M16, M74, M65;
                    ! add, ALUout, ldREG, daREG, ldPSWALU, DSout !
        madr48  cnt, bruncond, madr9B;
! AND !
```

```
        madr49  M41, M16, M65, M71;          ! REGout, daREG, DSout, ldX !
        madr4A  M61, M72;                    ! Bout, ldY !
        madr4B  M83, M22, M45, M16, M74, M65;
                        ! and, ALUout, ldREG, daREG, ldPSWALU, DSout !
        madr4C  cnt, bruncond, madr9B;
! ASR !
        madr4D  cnt, brimmed, madr9A;
        madr4E  M61, M71;                    ! Bout, ldX !
        madr4F  M84, M22, M57, M32, M74;     ! asr, ALUout, ldBhigh, ldBlow, ldPSWALU !
        madr50  cnt, bruncond, madr92;
! BNZ!
        madr51  cnt, brZero, madr9B;
        madr52  M71, M23;                    ! ldX, PCout !
        madr53  M72, M55;                    ! ldY, IR2out !
        madr54  M82, M22, M75, M81;          ! add, ALUout, ldPChigh, ldPClow !
        madr55  cnt, bruncond, madr9B;
! JSR !
        madr56  M77, M35, M15, M23;                ! decSP, mxMBR, ldMBR, PCout !
        madr57  M44, M77, M65, M31;                ! upSPout, decSP, DSout, ldMAR !
        madr58  M13;                               ! write !
        madr59  M44, M65, M31;                     ! upSPout, DSout, ldMAR !
        madr5A  M46, M35, M15, M23;                ! MBRhigh, mxMBR, ldMBR, PCout !
        madr5B  M13;                               ! write !
! JMP !
        madr5C  M43, M66, M75, M81;                ! IR_JAout, SDout, ldPChigh, ldPClow !
        madr5D  cnt, bruncond, madr9B;
! JMPIND !
        madr5E  cnt, brIMMEDorREGIND, madr9A;
        madr5F  M61, M66, M75, M81;                ! Bout, SDout, ldPChigh, ldPClow !
        madr60  cnt, bruncond, madr9B;
! RTI !
        madr61  M44,
        M65, M31, M76;                             ! upSPout, DSout, ldMAR, incSP !
        madr62  M12;                               ! read !
        madr63  M15;                               ! ldMBR !
        madr64  M14, M73;                          ! MBRout, ldPSW !
! RTS !
        madr65  M44, M65, M31, M76;                ! upSPout, DSout, ldMAR, incSP !
        madr66  M12;                               ! read !
        madr67  M15, M44, M65, M31 M76;            ! ldMBR, upSPout, DSout, ldMAR incSP !
        madr68  M14, M75;                          ! MBRout, ldPChigh !
        madr69  M12;                               ! read !
        madr6A  M15;                               ! ldMBR !
        madr6B  M14, M81;                          ! MBRout, ldPClow !
        madr6C  cnt, bruncond, madr9B;
! INT !
        madr6D  M8A;    ! setINT !
         madr6E cnt, bruncond, madr9B;
! PUSH !
        madr6F  M35, M15, M77, M61;                ! mxMBR, ldMBR, decSP, Bout !
        madr70  M31, M44, M65, M77;                ! ldMAR, upSPout, DSout, decSP !
        madr71  M13;                               ! write !
        madr72  M31, M44, M65;                     ! ldMAR, upSPout, DSout !
        madr73  M35, M46, M61, M15;                ! mxMBR, MBRhigh, Bout, ldMBR !
        madr74  M13;                               ! write !
```

```
        madr75  cnt, bruncond, madr9B;
! POP !
        madr76  cnt, brimmed, madr9A;
        madr77  M21, M33;                       ! MARout, ldA !
        madr78  M31, M44, M65, M76;             ! ldMAR, upSPout, DSout, incSP !
        madr79  M12;                            ! read !
        madr7A  M15, M31, M44, M65,  M76;       ! ldMBR, ldMAR, upSPout, DSout,  incSP !
        madr7B  M14, M57;                       ! MBRout, ldBhigh !
        madr7C  M12;                            ! read !
        madr7D  M15, M31, M24;                  ! ldMBR, ldMAR, Aout !
        madr7E  M14, M32;                       ! MBRout, ldBlow !
        madr7F  M61, M71;                       ! Bout, ldX !
        madr80  M88, M74;                       ! trans, ldPSWALU !
        madr81  cnt, bruncond, madr92;
! INC !
        madr82  cnt, brimmed, madr9A;
        madr83  M61, M71;                       ! Bout, ldX !
        madr84  M85, M22, M57, M32, M74;        ! inc, ALUout, ldBhigh, ldBlow, ldPSWALU !
        madr85  cnt, bruncond, madr92;
! DEC !
        madr86  cnt, brimmed, madr9A;
        madr87  M61, M71;                       ! Bout, ldX !
        madr88  M86, M22, M57, M32, M74;        ! dec, ALUout, ldBhigh, ldBlow, ldPSWALU !
        madr89  cnt, bruncond, madr92;
! INTE !
        madr8A  M8C;                            ! setI !
        madr8B  cnt, bruncond, madr9B;
! INTD !
        madr8C  M8D;                            ! resetI !
        madr8D  cnt, bruncond, madr9B;
! TRPE !
        madr8E  M17;                            ! setT !
        madr8F  cnt, bruncond, madr9B;
! TRPD !
        madr90  M37;                            ! resetT !
        madr91  cnt, bruncond, madr9B;
! Vraćanje podatka !
        madr92  cnt, brregdir, madr98;
        madr93  M61, M15, M35;                  ! Bout, ldMBR, mxMBR !
        madr94  M13;                            ! write !
        madr95  M26, M61, M15, M35, M46;        ! decMAR, Bout, ldMBR, mxMBR, MBRhigh !
        madr96  M13;                            ! write !
        madr97  cnt, bruncond, madr9B;
        madr98  M61, M45, M36;                  ! Bout, ldREG, fvo !
        madr99  cnt, bruncond, madr9B;

! Opsluživanje prekida !
        madr9A  M8B;                            ! setADR !
        madr9B  cnt, brnotPREKID, madr00;
        madr9C  M77, M23, M35, M15, M47, M67;
                    ! decSP, PCout, mxMBR, ldMBR, ldBR, intack !
        madr9D  M44, M65, M31, M77;             ! upSPout, DSout, ldMAR, decSP !
        madr9E  M13;                            ! write !
        madr9F  M44, M65,  M31, M77;            ! upSPout, DSout,  ldMAR, decSP !
```

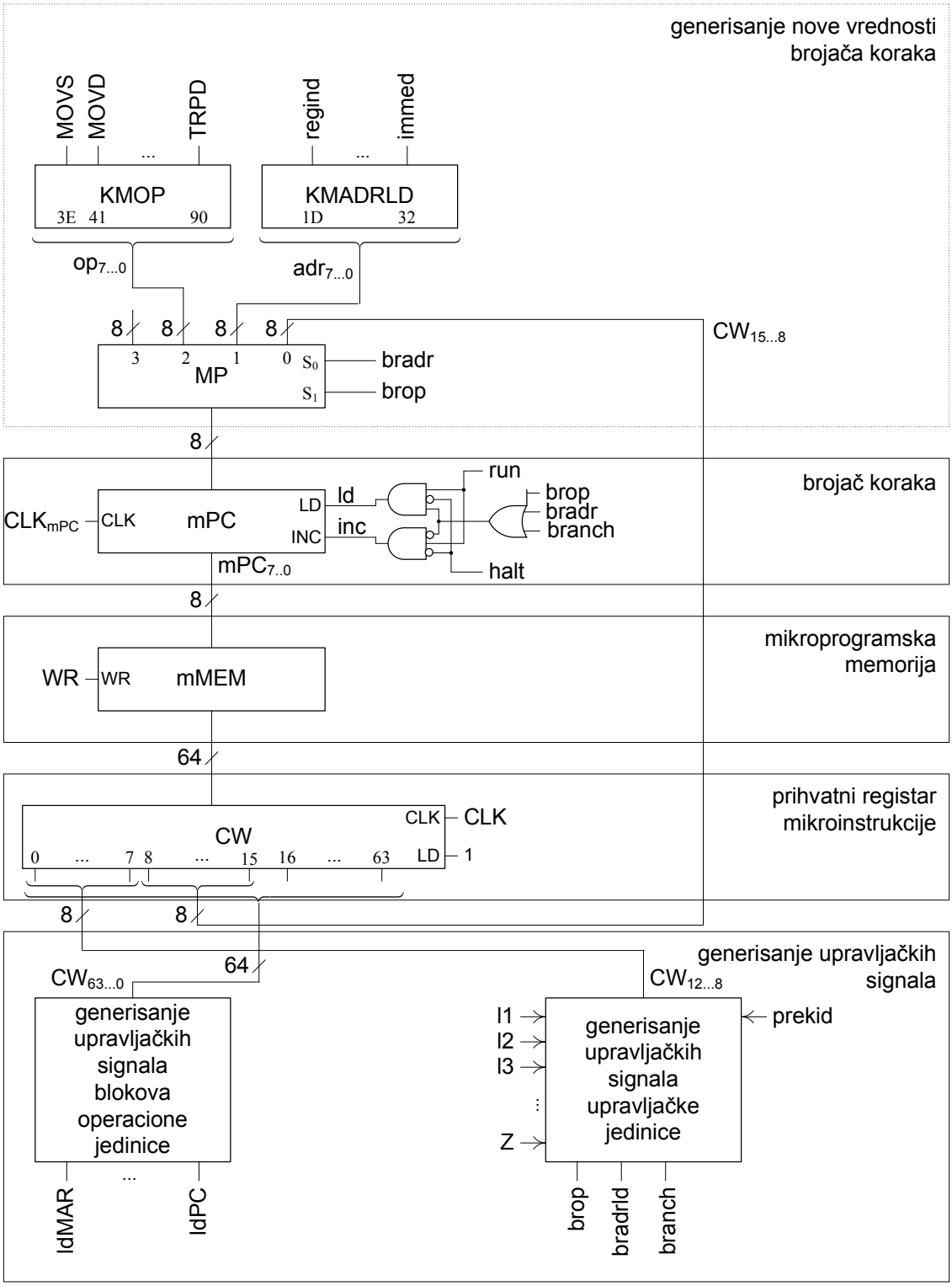
$madr_{A0}$ $M2_3$, $M3_5$, $M1_5$**,** $M4_6$;	**! PCout**, **mxMBR**, **ldMBR, MBRhigh !**
$madr_{A1}$ $M1_3$**;**	**! write !**
$madr_{A2}$ $M4_4$, $M6_5$**,** $M3_1$;	**! upSPout**, **DSout, ldMAR !**
$madr_{A3}$ $M6_2$, $M3_5$, $M1_5$;	**! PSWout**, **mxMBR**, **ldMBR !**
$madr_{A4}$ $M1_3$**;**	**! write !**
$madr_{A5}$ $M6_3$, $M7_1$	**! BRout**, **ldX !**
$madr_{A6}$ $M8_7$**,** $M2_2$**,** $M6_5$**,** $M7_1$**;**	**! shl, ALUout, DSout, ldX !**
$madr_{A7}$ $M6_4$, $M7_2$;	**! IVTPout**, **ldY !**
$madr_{A8}$ $M8_2$, $M2_2$**,** $M3_1$**,** $M6_5$ **;**	**! add**, **ALUout, ldMAR, DSout !**
$madr_{A9}$ $M1_2$**;**	**! read !**
$madr_{AA}$ $M1_5$;	**! ldMBR !**
$madr_{AB}$ $M1_4$, $M7_5$**,** $M3_4$;	**! MBRout**, **ldPChigh, incMAR !**
$madr_{AC}$ $M1_2$**;**	**! read !**
$madr_{AD}$ $M1_5$;	**! ldMBR !**
$madr_{AE}$ $M1_4$, $M8_1$;	**! MBRout**, **ldPClow !**
$madr_{AF}$ **cnt, bruncond, madr**$_{00}$;	

Struktura upravljačke jedinice je slična strukturi upravljačke jedinice sa dva tipa mikroinstrukcija i horizontalnim kodiranjem upravljačkih signala operacione jedinice i data je na slici 17. Mikroprogramski brojač se na isti način inkrementira, u mikroprogramski brojač se na isti način upisuje nova vrednost i mikroprogramski brojač prolazi kroz isti broj stanja. Iste su i vrednosti koje generišu kombinacione mreže KMOPR i KMADR i vrednosti $CW_{k+1...k+n}$ koje se upisuju iz upravljačke mikroinstrukcije.

Upravljački signali operacione jedinice se generišu ukoliko je $\mathbf{CW_0} = 0$, jer se tada u prihvatnom registru mikroinstrukcije $CW_{0...k+n}$ nalazi operaciona mikroinstrukcija. Upravljački signali operacione jedinice se generišu na sledeći način (tabela 25):

- **ldMAR** = $\overline{CW_0} \cdot \overline{CW_7} \cdot \overline{CW_8} \cdot \mathbf{CW_9}$
- **PCout** = $\overline{CW_0} \cdot \overline{CW_4} \cdot \mathbf{CW_5} \cdot \mathbf{CW_9}$
- **SDout** = $\overline{CW_0} \cdot \mathbf{CW_{16}} \cdot \mathbf{CW_{17}} - \overline{CW_{18}}$ itd.

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



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

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

- **bropr = $CW_0 \cdot CW_4 \cdot CW_5 \cdot \overline{CW}_6 \cdot CW_7$**

- **bradr = $CW_0 \cdot CW_4 \cdot CW_5 \cdot CW_6 \cdot \overline{CW_7}$**
- **branch = bruncnd**

  + brl1*l1 + brl2* $\overline{l2}$ **+ brl3*l3 + brnotl4*** $\overline{l4}$ **+ brl5*l5 + brMOVDorPOP*(MOVD+POP) + brimmed*immed + brZero*Z + brIMMEDorREGIND*(immed+regind) + brREGDIR*regdir + brnotPREKID*** $\overline{PREKID}$

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

- **bruncond = $CW_0 \cdot \overline{CW_4} \cdot \overline{CW_5} \cdot \overline{CW_6} \cdot CW_7$**
- **brl1 = $CW_0 \cdot \overline{CW_4} \cdot \overline{CW_5} \cdot CW_6 \cdot \overline{CW_7}$**
- **brnotl2 = $CW_0 \cdot \overline{CW_4} \cdot \overline{CW_5} \cdot CW_6 \cdot CW_7$**
- **brl3 = $CW_0 \cdot \overline{CW_4} \cdot CW_5 \cdot \overline{CW_6} \cdot \overline{CW_7}$**
- **brnotl4 = $CW_0 \cdot \overline{CW_4} \cdot CW_5 \cdot \overline{CW_6} \cdot CW_7$**
- **brl5 = $CW_0 \cdot \overline{CW_4} \cdot CW_5 \cdot CW_6 \cdot \overline{CW_7}$**
- **brMOVDorPOP = $CW_0 \cdot \overline{CW_4} \cdot CW_5 \cdot CW_6 \cdot CW_7$**
- **brimmed = $CW_0 \cdot CW_4 \cdot \overline{CW_5} \cdot \overline{CW_6} \cdot \overline{CW_7}$**
- **brZero = $CW_0 \cdot CW_4 \cdot \overline{CW_5} \cdot \overline{CW_6} \cdot CW_7$**
- **brIMMEDorREGIND = $CW_0 \cdot CW_4 \cdot \overline{CW_5} \cdot CW_6 \cdot \overline{CW_7}$**
- **brREGDIR = $CW_0 \cdot CW_4 \cdot \overline{CW_5} \cdot CW_6 \cdot CW_7$**
- **brnotPREKID = $CW_0 \cdot CW_4 \cdot CW_5 \cdot \overline{CW_6} \cdot \overline{CW_7}$**

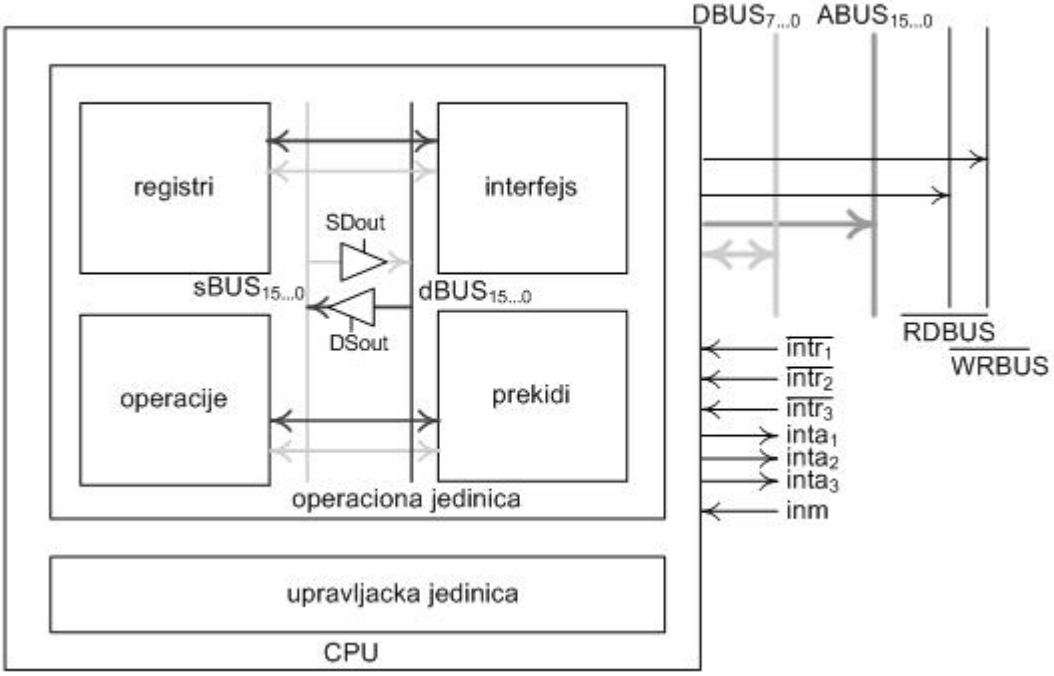
Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice i to: **l1**, **l2**, **l3, l4, l5, MOVD, POP, immed, Z, immed, regind, regdir, PREKID**.

Moguće je mešovito kodiranje signala i za slučaj kada postoji spajanje koraka i samo jedan tip mikroinstrukcije.Sličnim postupkom bi se formirao mikroprogram u kome bi u operacionom delu mikroinstrukcije upravljački signali operacione jedinice bili mešovito kodirani na način prikazan ranije.

Struktura upravljačke jedinice je veoma slična strukturi razmatrane upravljačke jedinice sa dva tipa mikroinstrukcija. Razlika je samo u delu u kome se generišu upravljački signali operacione jedinice koji se dobijaju dekodovanjem odgovarajućih grupa bitova operacionog dela mikroinstrukcije. Upravljački signali upravljačke jedinice se generišu na identičan način kao i u slučaju razmatrane upravljačke jedinice sa dva tipa mikroinstrukcija.

5 PRILOZI

U ovoj glavi se daje organizacija procesora ***CPU*** koji se sastoji iz operacione jedinice ***operaciona_jedinica*** i upravljačke jedinice ***upravljačka_jedinica***.



Operaciona jedinica ***operaciona_jedinica*** je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova upravljačke jedinice ***upravljačka_jedinica***. Upravljačka jedinica ***upravljačka_jedinica*** je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala operacione jedinice ***operaciona_jedinica*** na osnovu algoritama operacija i signala logičkih uslova.

Struktura i opis operacione i upravljačke jedinice se daju u daljem tekstu.

OPERACIONA JEDINICA

Operaciona jedinica ***operaciona_jedinica*** se sastoji od sledećih blokova:

- blok *registri*,
- blok *operacije*,
- blok *interfejs* i
- blok *prekidi*.

Ovi blokovi su međusobno povezani internim magistralama *bus* koje sadrže 16 linija $sBUS_{15...0}$ i $dBUS_{15...0}$.

Blok *registri* služi za realizaciju programskog brojača, prihvatnog registra instrukcije, ukazivača na vrh steka, registara AX – BP, pomoćnih registara A i B i programske statusne reči. Blok *operacije* služi za realizaciju aritmetičkih, logičkih i pomeračkih mikrooperacija. Blok *interfejs* služi za arbitraciju procesora i ulazno/izlaznih uređaja ***U/I*** sa kontrolerima za direktan pristup memoriji pri korišćenju magistrale ***BUS*** i realizaciju ciklusa na magistrali

BUS kada je procesor gazda. Blok *prekidi* služi za prihvatanje prekida, generisanje broja ulaza u tabelu sa adresama prekidnih rutina, kao i za arbitraciju.

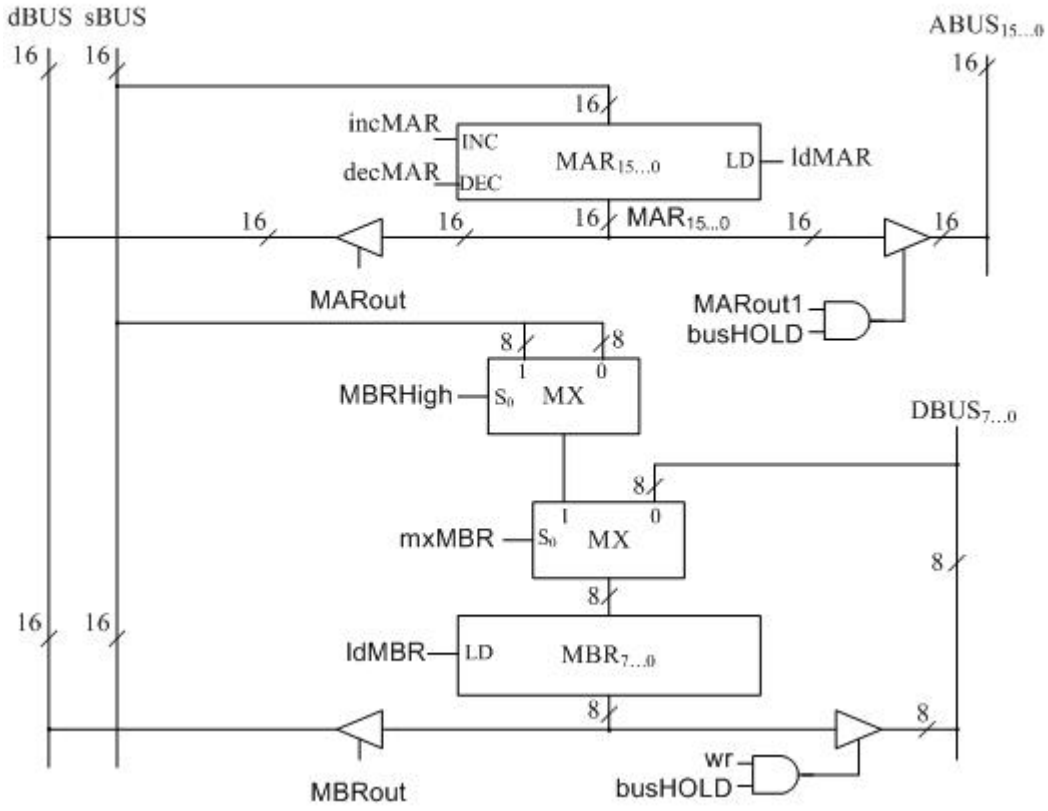
Struktura i opis blokova operacione jedinice ***operaciona_jedinica*** se daju u daljem tekstu.

5.1 BLOK INTERFEJS

Na slici 5.1 prikazana je šema blok interfejsa. Šema se sastoji od registara **MAR** i **MBR**, dva multipleksera i četiri trostatička bafera za izlaz na odgovarajuću magistarlu.

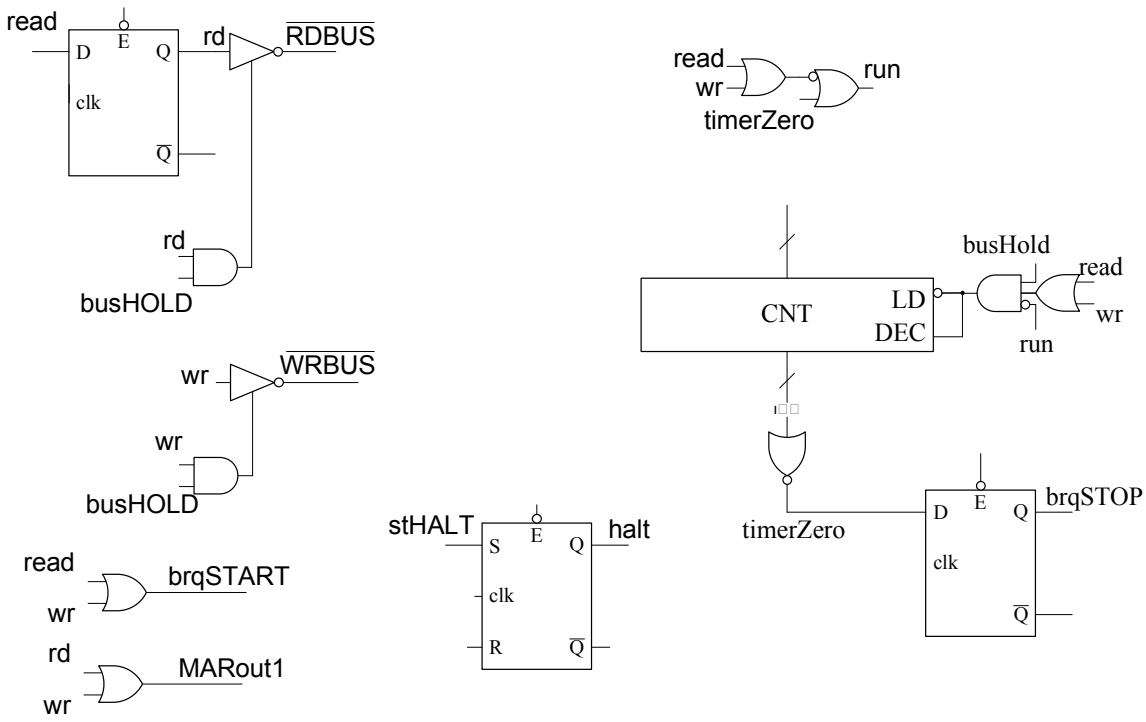
Registar **MAR** je šesnaestobitni registar u kojem se nalazi adresa koja se izbacuje sistemsku adresnu magistralu. Ovaj registar sadrži još i **ldMAR** kontrolni ulaz za učitavanje vrednosti sa interne sBUS magistrale, zatim kontrolne ulaze za inkrementiranje (**incMAR**) i dekrementiranje (**decMAR**). Kontrolni ulaz za inkrementiranje se koristi za dovlačenje jedne reči iz memorije, jer je za to potrebno dva pristupa memoriji. Nakon prvog pristupa biće dovučen donji bajt podatka, zatim će registar **MAR** biti inkrementiran i zatim će biti učitan gornji bajt podatka. Kontrolni ulaz za dekrementiranje se koristi prilikom vraćanja podatka u memoriju na sledeći način: nakon faze dohvatanja operanda u registru **MAR** nalaziće se adresa drugog bajta podatka koja će biti korišćena prilikom vraćanja podatka. Ukoliko je potrebno vratiti podatak u memoriju nakon izvršenja instrukcije, sačuvana adresa iz registra **MAR** će se iskoristiti za smeštanje gornjeg bajta podatka u memoriju, zatim će se registar **MAR** dekrementirati i na kraju će biti smešten i donji bajt podatka u memoriju. Signalom **MARout** koji dolazi iz upravljačke jedinice se sadržaj registra **MAR** izbacuje na internu dBUS magistralu. To je potrebno u slučajevima kada u toku izvršavanja instrukcije postoje ciklusi sa memorijom, stoga je potrebno prvo sačuvati registar **MAR** u registru **A**, a zatim se registar **MAR** koristi na uobičajeni način. Trostatički bafer koji izalzi na adresnu sistemsku magistralu propušta signal ukoliko je u toku ciklus čitanja i ukoliko je dobijeno pravo za korišćenje magistrale (**busHOLD**). Signal **MARout1** je ili funkcija signala **rd** i **wr**. Signal **rd** je zakasnjeni signal za jedan takt **read** koji se generiše iz upravljačke jedinice. Potrebno je da on bude zakašnjen da bi u slučaju čitanja iz memorije podatak na sistemskoj magistrali podataka bio dostupan i u koraku posle da bi mogao da se generiše signal **ldMBR** kako bi podatak sa sistemske magistrale podataka bio učitan u registar **MBR**.

Registar **MBR** je osmobitni registar koji se koristi kao prihvatni registar podatka prilikom čitanja iz memorije, dok se u slučaju upisa u njemu nalazi podatak koji će biti izbačen na sistemsku magistralu podataka.Signal **wr** se generiše iz upravljačke jedinice u slučaju upisa podatka u memoriju. Signal **busHOLD** je signal koji potvrđuje da je procesor dobio dozvolu za korišćenje magistrale. Trostatički bafer koji je vezan na internu dBUS magistralu podatka aktivira se signalom **MBRout** koji dolazi iz upravljačke jedinice i time se sadržaj registra **MBR** propušta na internu dBUS magistralu. S' obzirom da je registar **MBR** osmobitni a interna dBUS magistrala šesnaestobitna vrednost registra **MBR** se izbacuje i na gornji i na donji bajt interne dBUS magistrale. Trostatički bafer koji je vezan na sistemsku magistralu podataka aktivira se ukoliko je aktivan signal **wr** i signal **busHOLD** čija je uloga objašnjena gore. Kako je sistemska sBUS magistrala široka šesnaest bita, a registar **MBR** osam bita, gornji bajt interne sBUS magistrale se vodi na jedan, a donji bajt na drugi ulaz multipleksera. Ukoliko je aktivan signal **MBRHigh** koji dolazi iz upravljačke jedinice tada se propušta gornji bajt, u protivnom se propušta donji bajt. Signalom **mxMBR** koji dolazi iz upravljačke jedinice i njegovom aktivnom vrednošću propušta se izlaz gornjeg multipleksera na ulaz registra **MBR**, dok se u protivnom na ulaz registra **MBR** propušta vrednost sa sistemske magistrale podataka.



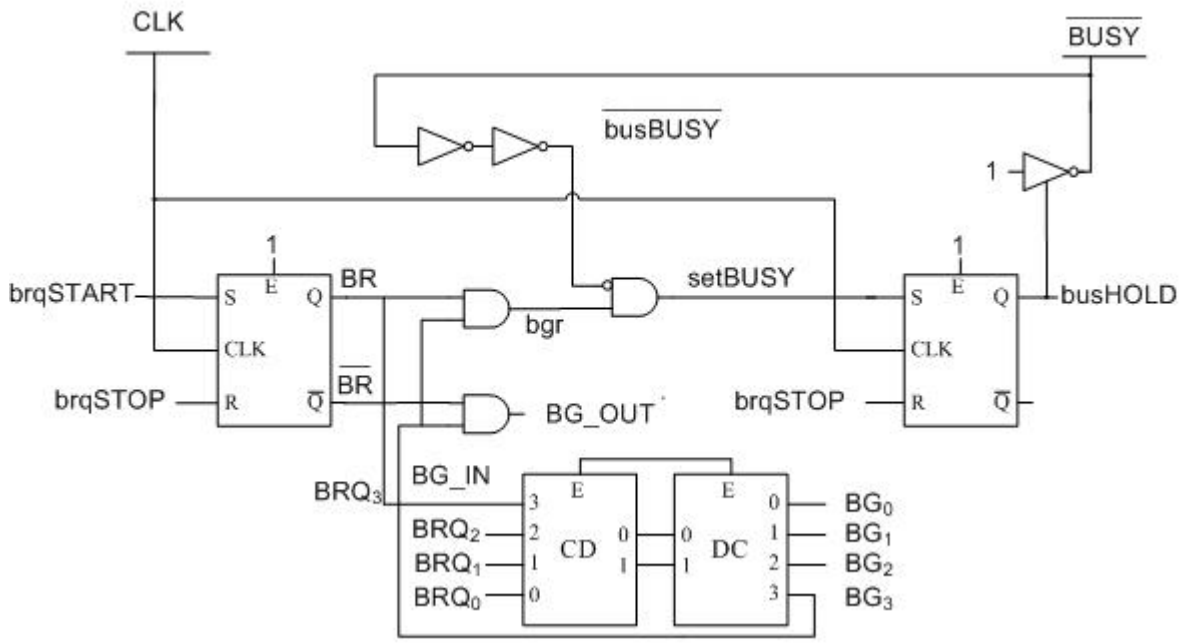
Slika 5.1

Na slici 5.2 prikazan je deo za sinhronizaciju. Signal **rd** je zakašnjeni signal za jedan takt **read** koji se generiše iz upravljačke jedinice. Potrebno je da on bude zakašnjen da bi u slučaju čitanja iz memorije podatak na sistemskoj magistrali podataka bio dostupan i u koraku posle da bi mogao da se generiše signal **ldMBR** kako bi podatak sa sistemske magistrale podataka bio učitan u registar **MBR**. Signal **wr** se generiše iz upravljačke jedinice u slučaju čitanja podatka iz memorije. Signal **busHOLD** je signal koji potvrđuje da je procesor dobio dozvolu za korišćenje magistrale. Ukoliko je potrebno pročitati podatak, tj. ukoliko je aktivan signal **rd** i ukoliko je dobijena dozvola za korišćenje magistrale biće aktivirana linija **RDBUS**. Ukoliko je potrebno upisati podatak u memoriju, tj. ukoliko je aktivan signal **wr** i ukoliko je dobijena dozvola za korišćenje magistrale biće aktivirana linija **WRBUS**. Na taj način se memoriji šalje signal koju operaciju treba da izvrši koristeći izbačenu adresu na sistemsku adresnu magistralu. SR flip flop koji se setuje signalom **stHALT** služi za zaustavljanje procesora u slučaju instrukcije **HALT**. Signal **run** je aktivan kada nije u toku ni jedan ciklus na magistrali i kada nije u toku čekanje na dozvolu za korišćenjem magistrale. Ukoliko je **run** signal neaktivan i ukoliko je dobijena dozvola za korišćenje magistrale nakon zadavanja operacije čitanja signalom **read** ili operacije upisa signalom **wr** brojač **CNT** će biti dekrementiran. Nakon što brojač dođe do nule biće aktivan signal **timerZero** kojim se setuje D flip flop. Izlaz D flip flopa je signal **brqSTOP** kojim se arbitratoru javlja da je ciklus na magistrali završen i da procesor osobađa magistralu. Ukoliko je signal **run** aktivan u brojač **CNT** će biti stalno upisivana vrednost sa ulaza koja se hardverski postavlja u zavisnosti od trajanja ciklusa na magistrali. Kako je magistrala sinhrona ciklus na magistrali će trajati određen broj taktova, a znak da je ciklus gotov je signal **timerZero** koji koji će biti aktivan kada borjač **CNT** odbroji do nule.



Slika 5.2

Slika 5.3 predstavlja arbitrator za 4 uređaja. Na slici je iskorišćen samo jedan ulaz i to ulaz najvećeg prioriteta za potrebe procesora, dok se na ostala tri mogu vezati neke periferije sa direktnim pristupom memoriji. Procesor traži korišćenje magistrale signalom **brqSTART**. Ukoliko je magistrala slobodna, tj. ukoliko je signal $\overline{\text{BUSY}}$ neaktivan, a signal **brqSTART** je aktivan biće aktivan i signal **setBUSY** kojim se postavlja signal $\overline{\text{BUSY}}$ na aktivnu vrednost, a procesoru se dozvoljava korišćenje magistrale signalom **busHOLD**. Nakon završenog korišćenja magistrale postavlja se signal **brqSTOP** kojim se resetuju oba flip flopa i na taj način se signal $\overline{\text{BUSY}}$ deaktivira. Za arbitriranje se koristi prioritetni koder sa četiri ulaza. Kada je na ulazu kodera aktivan neki signal na njegovom izlazu će se pojaviti kod ulaza sa najvećim prioritetom koji se nakon dekodera vraća radi formiranja signala **setBUSY**. Signal **BG_OUT** se može iskoristiti za ulančavanje arbitratora ukoliko je potreban veći broj periferija nego što dati arbitrator može da podrži. Na izlazu dekodera će se pojaviti neka vrednost samo ukoliko na ulazu prioritetnog kodera postoji neki zahtev. To je obezbeđeno time što je izlaz **W** prioritetnog kodera vezan za ulaz **E** dekodera.


CLK
BUSY
busBUSY
1
1
1
brqSTART
S
E
Q
BR
setBUSY
bgr
busHOLD
CLK
brqSTOP
R
Q
BR
BG_OUT
brqSTOP
BG_IN
BRQ3
BRQ2
BRQ1
BRQ0
CD
DC
BG0
BG1
BG2
BG3

Slika 5.3

5.2 BLOK REGISTRI

Blok *registri* sadrži registre PC, IR1, …, IR4, SP, A i B kombinacionu mrežu KM, registre AX, BX, CX, DX, SI, DI, BP dekodera, multipleksera, registar PSW.

Registar $PC_{15...0}$ je programski brojač (slika 5.10). Registri $IR1_{7...0}$ … $IR4_{7...0}$ formiraju prihvatni registar instrukcije, pri čemu se za saglasno dužini instrukcije koriste registri do indeksa dužine instrukcije, uključujući i taj registar.

Paralelan upis i čitanje registara $PC_{15...0}$, $IR1_{7...0}$ … $IR4_{7...0}$ se realizuje na identičan način za sve registre. Upis sadržaja sa interne magistrale $dBUS_{15...0}$ u registar $PC_{15...0}$ se obavlja na signal takta ukoliko je aktivan signal **ldPC**. Izlazi registra $PC_{15...0}$ su vezani na internu magistralu $sBUS_{15...0}$ preko bafera sa tri stanja, pa se njegovo paralelno čitanje i propuštanje sadržaja na linije $sBUS_{15...0}$ obavlja aktivnom vrednošću signala **PCout**. Za registre $IR1_{7...0}$ … $IR4_{7...0}$ odgovarajući signali za paralelan upis i čitanje su **ldIR1, IR1out, …, ldIR4**.

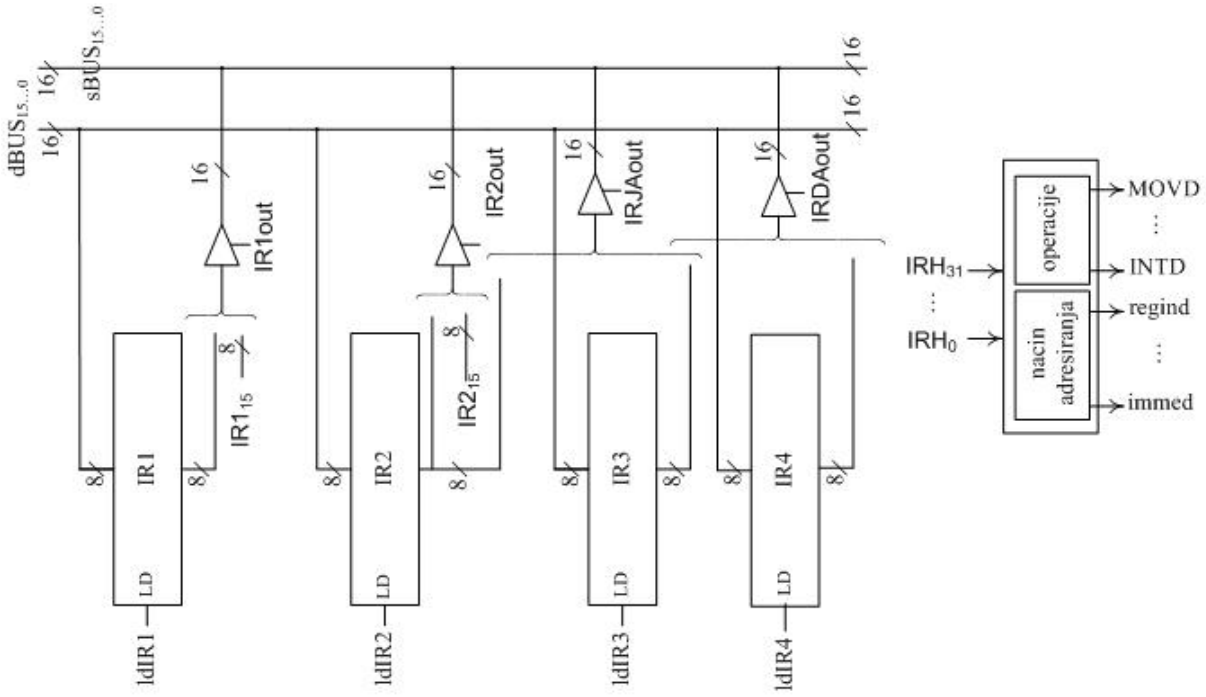
Registar $SP_{15...0}$ je ukazivač na vrh steka. Paralelan upis i čitanje registra $SP_{15...0}$ se realizuju na sličan način kao i za registre $PC_{15...0}$, $IRH_{7...0}$ i $IR4_{7...0}$. Odgovarajući signali za paralelan upis i paralelno čitanje su **ldSP** i **SPout**. Signali **ldSP** i **SPout** su unija signala **upldSP** i **ldREG*SPsel**, odnosno **upSPout** i **REGout*SPsel**. Paralelan upis i čitanje registra $SP_{15...0}$ se normalno realizuju signalima **ldSP** i **SPout**, respektivno.

Kombinaciona mreža KM na osnovu sadržaja registra $IRH_{32...0}$ generiše signale operacija i načina adresiranja.

Signali operacija su **MOVS**, **MOVD**, …, **HALT** i njihove oznake odgovaraju mnemonicima odgovarajućih instrukcija. Operacije se kodiraju saglasno uslovu zadatka. Signali **MOVS**, **MOVD**, ..., **HALT** se realizuju prema izrazima:

MOVS = $IRH_{31} * \overline{IRH}_{30} * \overline{IRH}_{29} * \overline{IRH}_{28} * \overline{IRH}_{27} * \overline{IRH}_{26} * \overline{IRH}_{25} * IRH_{24}$,

MOVD = $IRH_{31} * \overline{IRH}_{30} * \overline{IRH}_{29} * \overline{IRH}_{28} * \overline{IRH}_{27} * \overline{IRH}_{26} * IRH_{25} * \overline{IRH}_{24}$, itd.



Slika 5.4

Signali operacija **MOVS**, **MOVD**, …, **HALT** se koriste u upravljačkoj jedinici kao signali logičkih uslova za realizaciju odgovarajućih operacija.

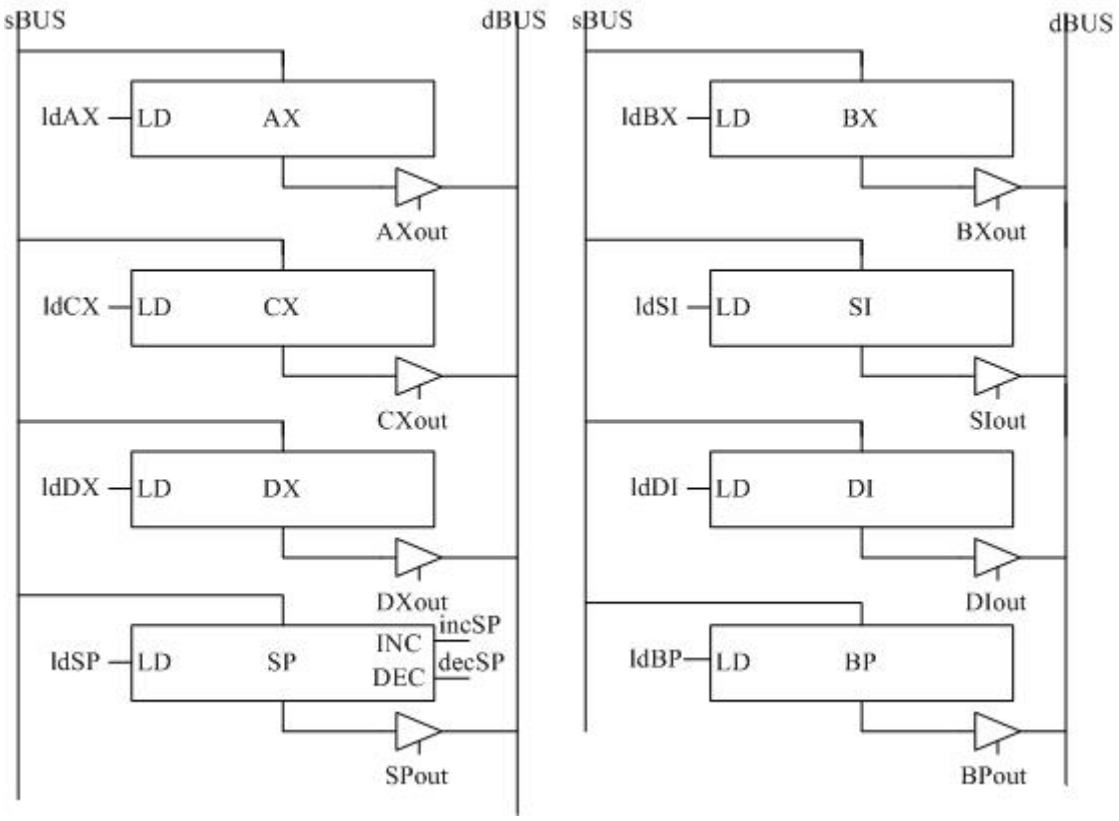
Signali načina adresiranja su **regdir** za registarsko direktno adresiranje, **regind** za registarsko indirektno adresiranje, **rindpom** za registarsko indirektno adresiranje sa pomerajem, **memdir** za memorijsko direktno adresiranje, **memind** za memorijsko indirektno adresiranje, **rel** za relativno adresiranje i **immed** za neposredno adresiranje. Načini adresiranja se kodiraju saglasno uslovima zadatka. Signali **regind**, **rindpom**, …, **immed** se realizuju prema izrazima

regdir = $\overline{IRH_{23}} * \overline{IRH_{22}}$

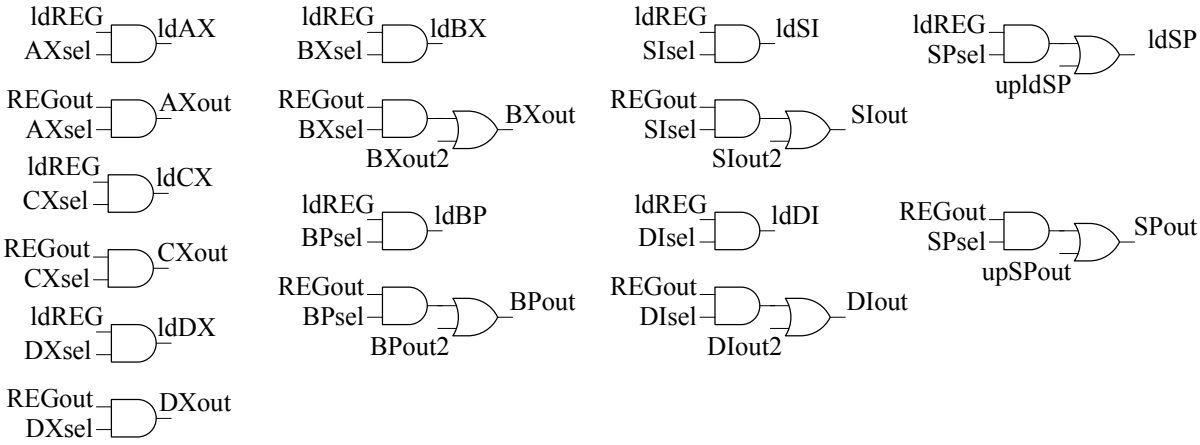
regind = $\overline{IRH_{23}} * IRH_{22}$ itd.

Ovi signali se koriste u upravljačkoj jedinici kao signali logičkih uslova za realizaciju odgovarajućih načina adresiranja.

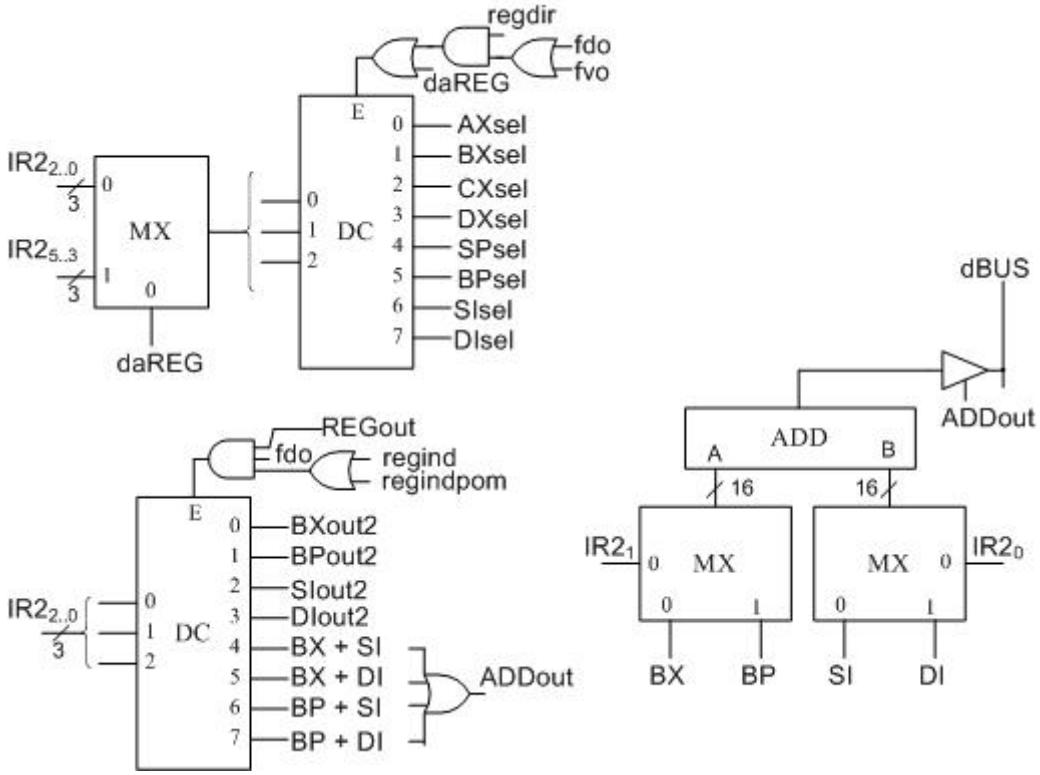
Registri **AX** do **BP** su registri opšte namene (slika 5.5). Upis sadržaja sa interne magistrale $sBUS_{15...0}$ u jedan od registara $AX_{15...0}$ do $BP_{15...0}$ se realizuje na signal takta pri aktivnoj vrednosti signala **ldREG** i odgovarajućen **sel** signalu tog registra. Čitanje sadržaja jednog od registara i njegovo slanje preko bafera sa tri stanja na internu magistralu $dBUS_{15...0}$ se realizuje pri aktivnoj vrednosti signala **REGout** (gde REG označava odgovarajući registar). Selekcija jednog od registara za upis ili čitanje se realizuje aktivnom vrednošću jednog od signala **AXsel** do **BPsel**. Signali **AXsel** do **BPsel** se formiraju na izlazima dekodera DC i to na osnovu vrednosti koje se na njegove ulaze dovode sa izlaza mulripleksera MP. Na izlazima multipleksera se pojavljuju vrednosti $IR2_2$ do $IR2_0$, $IR2_5$ do $IR2_3$ i njihova selekcija se realizuje aktivnom vrednošću signala **daREG**, respektivno. Ovakav način selekcije jednog od registara je posledica uslova zadatka. Zbog toga se pri aktivnoj vrednosti signala **ldREG** (gde REG označava odgovarajući registar) dobija aktivna vrednost jednog od signala **ldAX** do **ldBP** i upis sadržaja sa interne magistrale $sBUS_{15...0}$ realizuje u jedan od registara **AX** do **BP**, dok se pri aktivnoj vrednosti signala **REGout** (gde REG označava odgovarajući registar) realizuje čitanje sadržaja jednog od registara **AX** do **BP** i njegovo slanje preko bafera sa tri stanja na internu magistralu $dBUS_{15...0}$.



Slika 5.5



Slika 5.6



Slika 5.7

Flip-flopovi PSWN, PSWZ, PSWC, PSWV, $PSWL_0$, $PSWL_1$, PSWT i PSWI formiraju razrede programske statusne reči $PSW_{15...0}$, U registar $PSW_{15...0}$ se može upisivati na dva načina i to ili u sve razrede istovremeno ili u grupe od jednog ili nekoliko razreda.

Upis u sve rezrede istovremeno se realizuje na signal takta tako što se na internu magistralu $sBUS_{15...0}$ postavi sadržaj koji treba upisati i generiše aktivna vrednost signala **ldPSW**. Upis u registar $PSW_{15...0}$ se normalno realizuje signalom **ldPSW**.

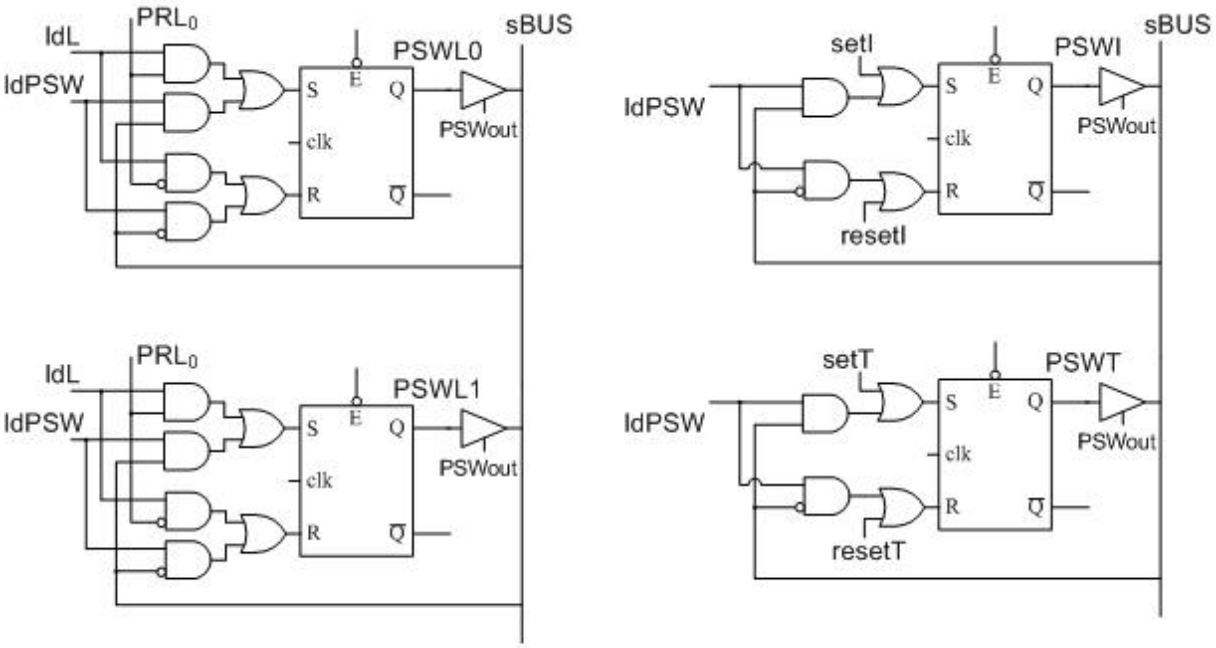
Upis u grupu od jednog ili nekoliko razreda registra $PSW_{15...0}$ se neznatno razlikuje za pojedine grupe razreda.

Upis u razrede PSWN, PSWZ, PSWC i PSWV (slika 5.9) se realizuje onda kada na osnovu rezultata izvršene operacije treba ažurirati ove indikatore. Vrednosti N, Z, C i V se upisuju na signal takta generisanjem aktivna vrednost signala **ldPSWALU**.

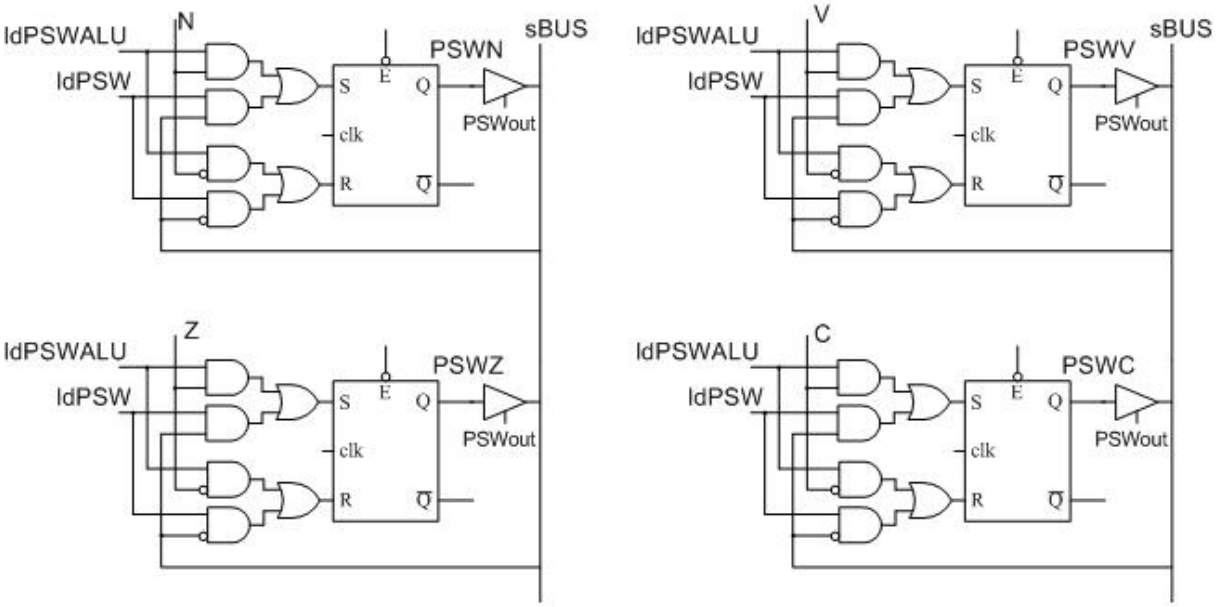
Upis u razrede $PSWL_1$ i $PSWL_0$ (slika 5.8) se realizuje prilikom opsluživanja maskirajućih prekida. Vrednosti prl_1 i prl_0 se upisuju na signal takta generisanjem aktivne vrednosti signala **ldPSW**.

Upis u razrede PSWI, PSWT (slika 5.8) se realizuje posebnim instrukcijama INTE, INTD, TRPE i TRPD. Aktivna vrednost se upisuje generisanjem aktivne vrednosti signala **ldPSW**.

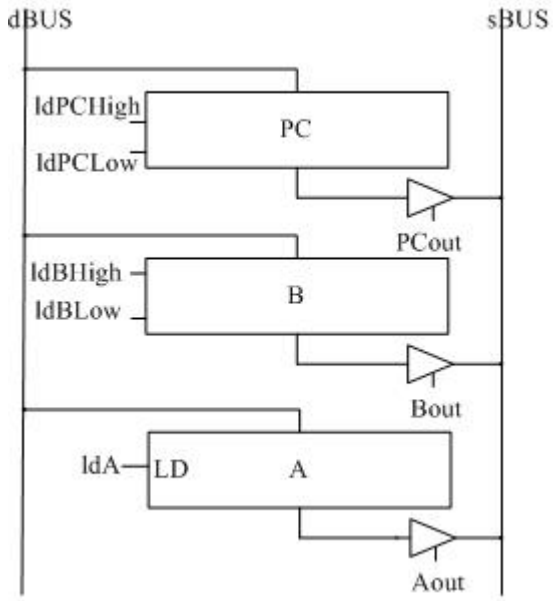
Čitanje sadržaja registra $PSW_{15...0}$ i njegovo slanje preko bafera sa tri stanja na internu magistralu $sBUS_{15...0}$ se obavlja aktivnom vrednošću signala **PSWout**.



Slika 5.8



Slika 5.9


dBUS
sBUS
ldPCHigh
ldPCLow
PC
PCout
ldBHigh
ldBLow
B
Bout
ldA
LD
A
Aout

Slika 5.10

5.3 BLOK OPERACIJE

Blok *operacije* sadrži registre $X_{15...0}$ i $Y_{15...0}$ (slika 5.11) i kombinacione mreže ALUSHIFT i KM (slika 5.12).

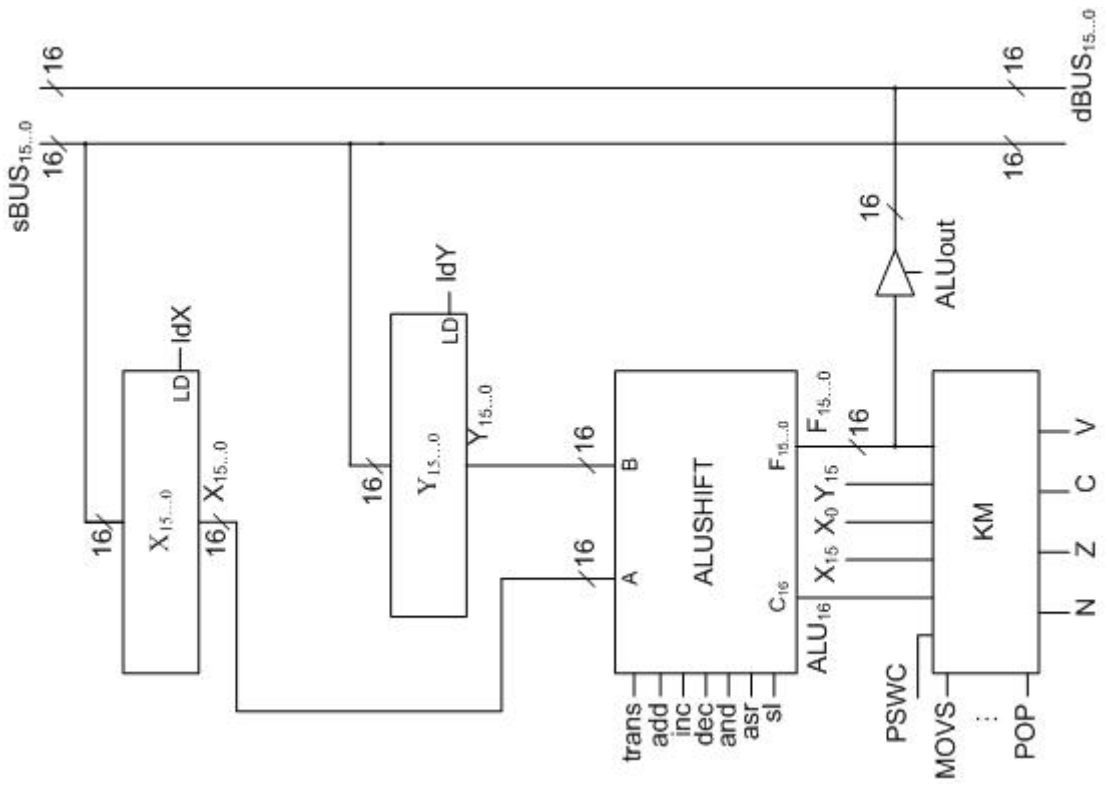
Registri $X_{15...0}$ i $Y_{15...0}$ služi za prihvatanje operanada nad kojima će se izvršiti neka mikrooperacija u kombinacionoj mreži ALUSHIFT (slika 5.11). Izlazi registara $X_{15...0}$ i $Y_{15...0}$ su povezani direktno na ulaze A i B kombinacione mreže ALUSHIFT. U registre $X_{15...0}$ i $Y_{15...0}$ se upisuje sadržaj sa interne magistrale $sBUS_{15...0}$ generisanjem aktivnih vrednosti signala **Xin** ili **Yin**, respektivno.

Kombinaciona mreža ALUSHIFT realizacije aritmetičke, logičke i pomeračke mikrooperacije. Mikrooperacije se realizuju nad sadržajima registara $X_{15...0}$ i $Y_{15...0}$. Mikrooperacija koju treba izvršiti specificira se upravljačkim signalima **trans**, **add**, …, **sl**. Rezultat mikrooperacije se dobija na linijama $F_{15...0}$, a u slučaju aritmetičkih mikrooperacija prenos se dobija na liniji C_{16} odnosno ALU_{16}.

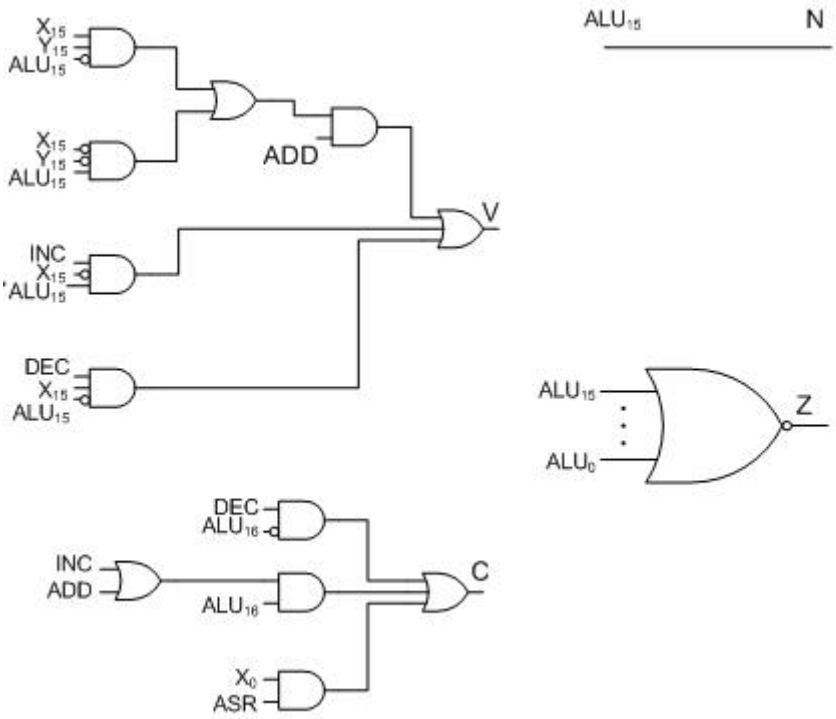
Na osnovu signala mikrooperacija **trans**, **add**, …, **sl**, od kojih samo jedan može u nekom trenutku da bude aktivan, formiraju se signali rezultata $\mathbf{F_{15...0}}$. Signali $\mathbf{F_{15...0}}$ se vode direktno u kombinacionu mrežu KM. Pored toga signali $\mathbf{F_{15...0}}$ se pri aktivnoj vrednosti signala **ALUSHIFTout** propuštaju preko bafera sa tri stanja na linije $sBUS_{15...0}$ interne magistrale.

Kombnaciona mreža KM generiše signale **N**, **Z**, **C** i **V** koji se upisuju u razrede PSWN, PSWZ, PSWC i PSWV registra $PSW_{15...0}$. Ovi signali se formiraju na osnovu signala

- operacija **MOVS**, **MOVD**, …, **SL**
- operanada $\mathbf{X_{15}}$, $\mathbf{X_0}$ i $\mathbf{Y_{15}}$ iz registara $X_{15...0}$ i $Y_{15...0}$,
- rezultata $\mathbf{F_{15...0}}$ i prenosa $\mathbf{ALU_{16}}$ iz ALUSHIFT i
- razreda **PSWC** registra $PSW_{15...0}$.



Slika 5.11



Slika 5.12

i to prema pravilima postavljanja indikatora N, Z, C i V definisanim za svaku instrukciju posebno. Instrukcije MOVS, MOVD, …, SL postavljaju indikatore N, Z, C i V, dok instrukcije TRPE, …, INTD, HALT ne postavljaju. Za instrukcije koje postavljaju indikatore N, Z, C i V formiraju se vrednosti signala **N**, **Z**, **C** i **V**. dobijaju sledeći izrazi za signale **N**, **Z**, **C** i **V**:

$\mathbf{N} = ALU_{15}$

$\mathbf{Z} = \overline{ALU_{15} * ALU_{14} * ... * ALU_{0}}$

$\mathbf{C} = DEC * \overline{ALU}_{16} + (INC + ADD) * ALU_{16} + X_0 * ASR$ i

$\mathbf{V} = (X_{15} * Y_{15} * \overline{ALU}_{15} + \overline{X}_{15} * \overline{Y}_{15} * ALU_{15}) * ADD$

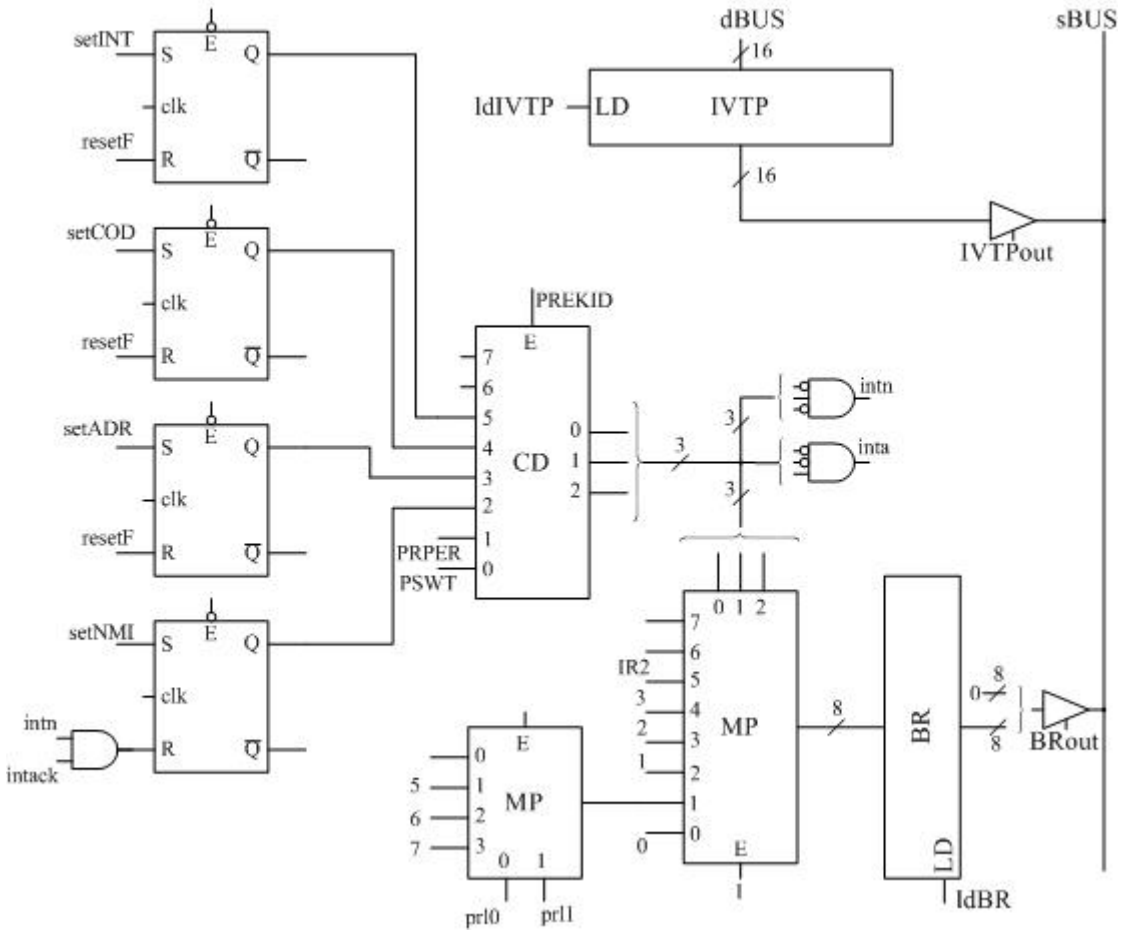
$+ INC * \overline{X}_{15} * ALU_{15} + DEC * X_{15} * \overline{ALU}_{15}.$

Za instrukcije koje postavljaju ove indikatore generiše se signal **ldPSWALU** kojim se upisuju vrednosti signala **N**, **Z**, **C** i **V** u razrede PSWN, PSWZ, PSWC i PSWV registra $PSW_{15...0.}$ Za instrukcije koje ne postavljaju ove indikatore ne generiše se signal **ldPSWALU**, pa se time ne menjaju vrednosti ovih razreda.

5.4 BLOK PREKIDI

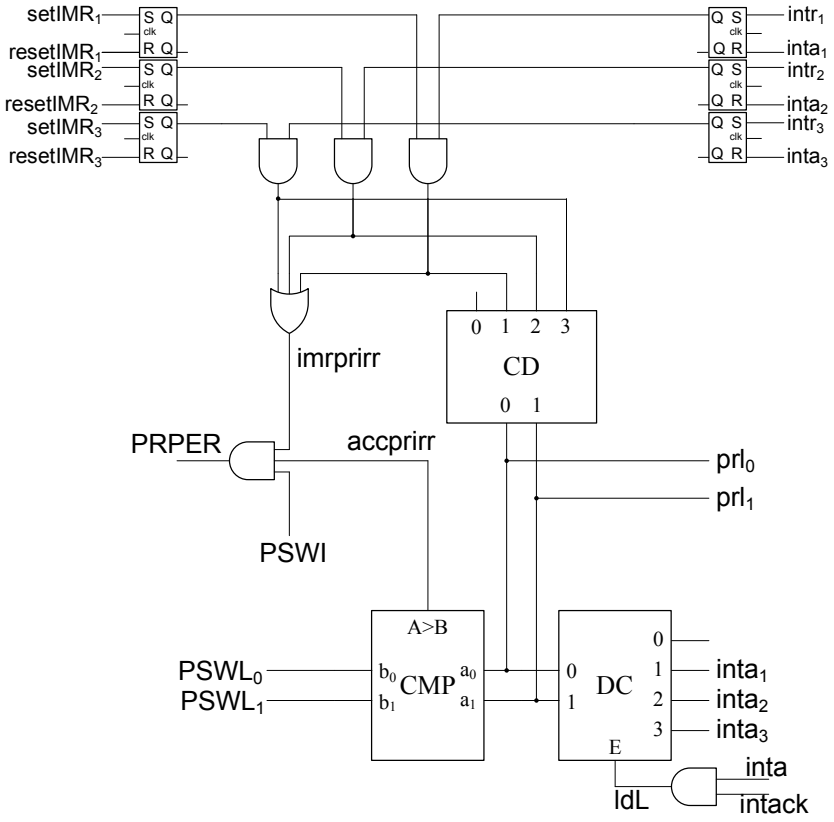
Na slici 5.13 prikazan je blok za prioritiranje zahteva za prekid i njihovo prihvatanje. Na ulazte flip flopova dovode se signali **setINT, setCOD, setADR** i **setNMI** kojima se neki od flip flopova setuje. Svaki od flip flopova služi za pamćenje da je stigao neki od zahteva za prekid. To može biti prekid usled instrukcije **INT**, prekid usled greške u kodu operacije, prekid usled greške u načinu adresiranja ili nemaskirajući prekid koji dolazi spolja u procesor. Maskirajući prekidi od periferija će biti objasnjeni na slici 5.14. na ulaz prioritetnog kodera dolaze signali postojanja nekog prekida. Izlaz **W** prioritetnog kodera će biti aktivan ukoliko postoji neki od prekida i taj izlaz vodi se u upravljačku jedinicu i na osnovu njega se ulazi u fazu opsluživanja prekida ili se prelazi na izvršavanje sledeće instrukcije. Na ulaz 0 prioritetnog kodera vezan je **T** bit iz **PSW**-a koji predstavlja zahtev za prekidom posle svake instrukcije. Na ulaz 1 prioritetnog kodera vezana je linija koja govori da postoji zahtev za maskirajući prekid. Na ulaz 2 prioritetnog kodera vezana je linija koja govori da postoji zahtev za maskirajućim prekidom. Na ulaz 3 prioritetnog kodera vezana je linija koja govori da postoji greška u načinu adresiranja. Na ulazu 4 prioritetnog kodera vezana je linija koja govori da postoji greška u kodu operacije. Na ulazu 5 prioritetnog kodera vezana je linija koja govori da je u toku instrukcija **INT**. Na osnovu zahteva sa najvećim prioritetom na izlazu kodera će se pojaviti kod kojim se selektuje jedan od ulaza multipleksera. Na ulaze multipleksera dovedene su fiksne vrednosti broja ulaza u tabelu prekidnih rutina, osim ulaza 5 na koji se dovodi vrednost registra **IR2** koji se koristi u slučaju da se radi o istrukciji **INT**. Izlaz multipleksera se vodi na ulaz registra **BR** u kojem se čuva broj ulaza u tabelu prekidnih rutina. Ukoliko je prihvaćen nemaskirajući prekid na izlazu multipleksera pojaviće se vrednost dva što će učiniti da signal **intn** bude aktivan. Ovaj signal se dovodi na i kolo zajedno sa signalom **intack** koji se generiše iz upravljačke jedinice u fazi opsluživanja zahteva za prekid. Izlaz ovog i kola se vodi na reset ulaz flip flopa čime se resetuje flip flop za nemaskirajuće prekide jer je nemaskirajući prekid upravo prihvaćen.

Registar $IVTP_{15...0}$ je ukazivač na tabelu sa adresama prekidnih rutina i sadrži početnu adresu tabele. Upis sadržaja sa linija $dBUS_{15..0}$ interne magistrale u registar $IVTP_{15...0}$ se obavlja ukoliko je aktivan signal **ldIVTP**. Upis u registar $IVTP_{15...0}$ se realizuje kod izvršavanja pripreme za obradu prekida. Čitanje sadržaja registra $IVTP_{15...0}$ i njegovo slanje preko bafera sa tri stanja na internu magistralu $sBUS_{15...0}$ se obavlja aktivnom vrednošću signala **IVTPout**. Čitanje sadržaja registra $IVTP_{15...0}$ se normalno realizuje signalom **IVTPout**. U slučaju prihvatanja prekida vrednost ulaza u tabelu prekidnih rutina se sabira sa vrednošću registra **BR** pomerenog za jedno mesto ulevo. Multiplekser sa četiri ulaza služi za izbor adrese prekidne rutine na osnovu toga sa koje je periferije stigao maskirajući zahtev za prekid. Kontrolni ulazi **prl0** i **prl1** služe za selektovanje jednog od ulaza na osnovu nivoa prioriteta periferije. Generisanje signala **prl0** i **prl1** biće prikazano na slici 5.14.



Slika 5.13

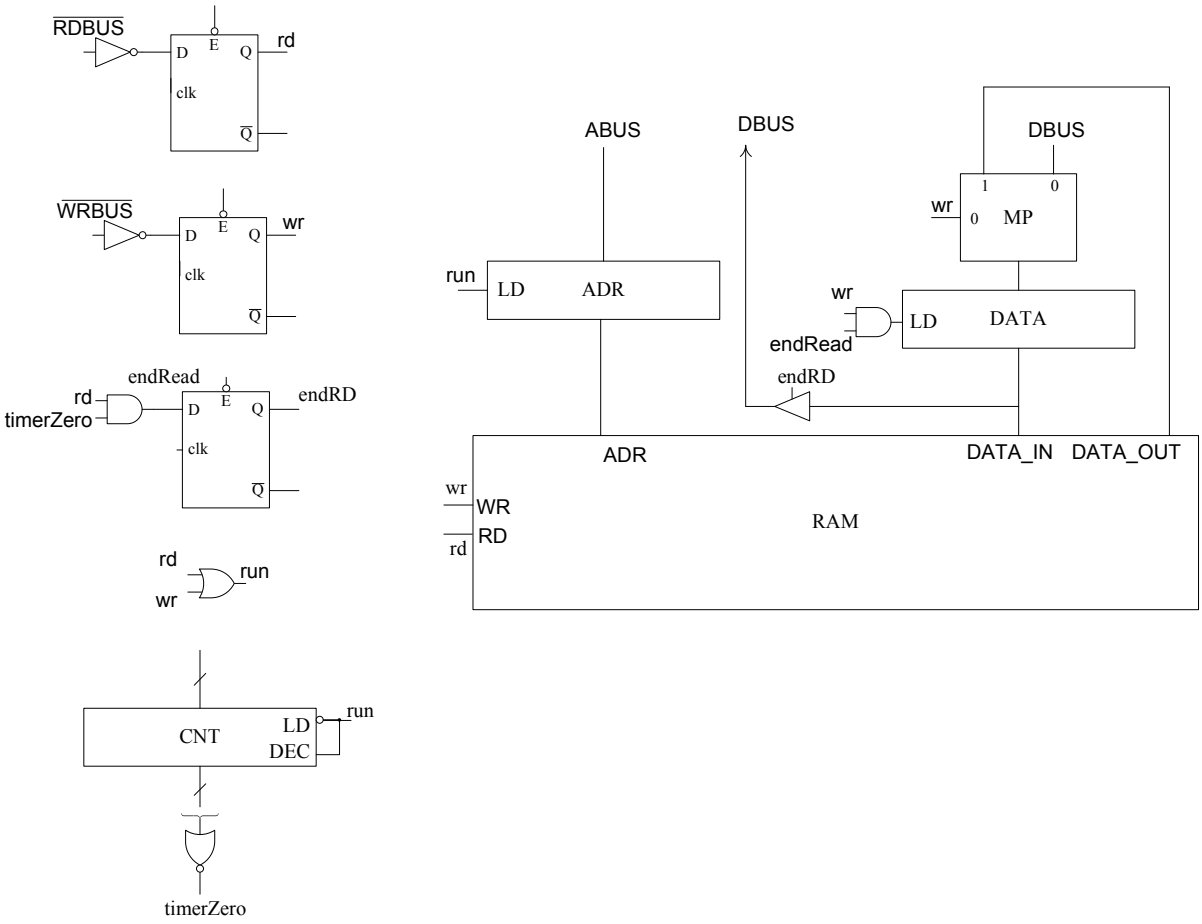
Na slici 5.14 prikazan je način na koji se prihvataju maskirajući prekidi od strane periferija. Registrima **IMR** moguće je selektivno maskirati prekide. Registri **IMR** setuju se signalima **setIMR**i i resetuju signalima **resetIMR**i. Zahtevi za prekid dolaze po linija **intr1** – **intr3**, a potvrde periferijama da je prekid prihvaćen šalju se po linijama **inta1 – inta3**. Ukoliko je stigao zahtev za prekid koji nije selektivno maskiran biće aktivan jedan od ulaza prioritetnog kodera I biće aktivan i signal **imrprirr** koji govori da postoji zahtev za maskirajućim prekidom. Na osnovu ulaza u prioritetni koder pojaviće se kod na njegovom izlazu, tj. biće postavljeni biti **prl0** i **prl1** u zavisnosti od prioriteta periferije koja je poslala zahtev za prekid. Signali **prl0** i **prl1** vode se na ulaz multipleksera sa slike 5.13, zatim na ulaze **a0** i **a1** komparatora gde se porede sa trenutnim vrednostima **L** flegova. Ukoliko je zahtev za prekid većeg prioriteta od trenutno izvršavanog programa, tj. ukoliko je izlaz komparatora aktivan prekid će biti prihvaćen, osim u slučaju da je bit **I** neaktivan. Ovo je realizovano pomoću i kola čiji izlaz predstavlja signal **PRPER** koji se vodi na prioritetni koder sa slike 5.13. Izlazi kodera **prl0** i **prl1** vode se još i na dekoder. Ukoliko je aktivan signal **inta** koji dolazi sa slike 5.13 u zavisnosti koji je prekid prihvaćen, i ukoliko je aktivan signal **intack** koji dolazi iz upravljačke jedinice u fazi opsluživanja zahteva za prekid, biće aktivan jedan od signala **inta1 – inta3** čime se resetuje flip flop zahteva da postoji zahtev za prekid od strane te periferije i istovremeno se salje potvrda periferiji da je zahtev prihvaćen. Signal koji ide na ulaz dekodera je signal **ldL** koji se vodi i na **L** bite **PSW-**a kojim se upisuje vrednost bita **prl0** i **prl1** čime se menja prioritet trenutno izvršavanog programa. Zahtevi za prekid od periferija se mogu slati kao nivo ili kao impuls jer će oni biti zapamćeni u registrima i dovoljno je njihovo postojanje samo u trenutku signala takta.


setIMR1
resetIMR1
setIMR2
resetIMR2
setIMR3
resetIMR3
S Q
clk
R Q
Q S
Q R
intr1
inta1
intr2
inta2
intr3
inta3
imrprirr
0 1 2 3
CD
0 1
PRPER
accprirr
prl0
prl1
PSWI
A>B
PSWL0
PSWL1
b0
b1
CMP
a0
a1
0
1
DC
0
1
2
3
E
inta1
inta2
inta3
inta
intack
IdL

Slika 5.14

6 MEMORIJA

Na slici 6.1 je prikazana realizacija memorije. Memorija je kapaciteta 64KB. Kada je aktivan neki od signala sa kontrolne magistrale, **RDBUS** ili **WRBUS**, bice setovan neki od **D** flip flopova i time će postati aktivan jedan od signala **rd** ili **wr.** Ovi signali se vode na **RD** i **WR** RAM memorije respektivno. Ti signali govore memoriji koja je operacija trenutno u toku, da li operacija čitanja ili pisanja. Signali **wr** i **rd** se dovode na ili kolo i izlaz ili kola predstavlja signal **run**. S' obzirom da je magistrala sinhrona memorija sadrži borjač **CNT** koji odbrojava na svaki signal takta sve do nule što označava da je završena zadata operacija. Ukoliko signal **run** nije aktivan vrednost sa ulaza u brojač **CNT** se upisuje u brojač. Ta vrednost se hardverski postavlja i zavisi od brzine memorije. Kada brojač odbroji do nule generiše se signal **timerZero**. Taj signal se vodi na i kolo sa signalom **rd**. Izlaz tog i kola je signal **endRead**, koji se zatim dovodi na D flip flop i time zakasni jedan takt. Zakasnjena verzija signala **endRead** je signal **endRD**. Korišćenje ova dva signala će biti opisano u nastavku. Memorija poseduje i dva prihvatna registra, jedan za adrese, drugi za podatke. U adresni registar se učitava sa adresne magistrale kada je aktivan signal **run.** Izlaz adresnog prihvatnog registra se vodi na ulaz **ADR** RAM memorije. Izlaz prihvatnog registra podataka se vodi na **DATA_IN** RAM memorije i na trostatički bafer koji je svojim izlazom vezan na magistralu podataka. Kontrolni izlaz trostatičkog bafera vezan je za signal **endRD** čime se obezbeđuje da se u taktu nakon što se pročitan podataka iz memorije upiše u prihvatni registar podataka taj podatak izbaci na sistemsku magistralu podataka. Na ulaz prihvatnog registra podataka dovodi se izlaz multipleksera na čijem ulazu se nalazi sistemska magistrala podataka i izlaz podataka iz memorije **DATA_OUT**. Signalom **wr** se propušta vrednost sa sistemske magistrale podataka do prihvatnog registra podataka. Vrednost sa ulaza u prihvatni registar podataka upisuje se kada je aktivan signal **wr** ili **endRead**. Kada je u pitanju ciklus čitanja nakon pročitanog podatka biće generisan signal **endRead** čime se vrednost sa linije **DATA_OUT** upisuje u prihvatni registar. U slučaju ciklusa upisa u memoriju biće aktivan signal **wr** tako da će se vrednost sa magistrale podataka stalno učitavati u prihvatni registar podataka za sve vreme trajanja ciklusa upisa.


RDBUS
D
E
Q
rd
clk
Q
WRBUS
wr
endRead
rd
timerZero
endRD
run
wr
CNT
LD
DEC
timerZero
ABUS
DBUS
DBUS
1
0
wr
0
MP
run
LD
ADR
wr
LD
DATA
endRead
endRD
ADR
DATA_IN
DATA_OUT
wr
WR
RD
rd
RAM

slika 6.1