JOVAN ĐORĐEVIĆ

ARHITEKTURA I ORGANIZACIJA RAČUNARA

ORGANIZACIJA RAČUNARA

BGD, 2011.

SADRŽAJ

1 STRUKTURA

Računarski sistem sadrži sledeće module: procesor ***CPU***, memoriju ***MEM*** i ulazno/izlazne uređaje ***U/I0*** do ***U/I7***. Procesor, memorija i ulazno/izlazni uređaji su međusobno povezani sistemskom magistralom ***BUS*** (slika 1). Moduli računarskog sistema rade sinhrono na isti signal takta **CLK**.

Arhitektura procesora od programski dostupnih registara ima registre opšte namene. Tipovi podataka sa kojima se radi su celobrojne 8-mo bitne veličine sa znakom i bez znaka. Format instrukcija je jednoadresni. Načini adresiranja uključuju registarsko direktno, registarsko indirektno, memorijsko direktno, memorijsko indirektno, registarsko indirektno sa pomerajem, bazno indeksno sa pomerajem, PC relativno sa pomerajem i neposredno adresiranje. Skup instrukcija uključuje instrukcije prenosa, aritmetičke instrukcije, logičke instrukcije, instrukcije pomeranja i rotiranja kao i instrukcije skoka. Prekidi uključuju spoljašnje prekide sa maskiranjem i prioritiranjem prekida, pri čemu se kontekst procesora se čuva na steku, a adresa prekidne rutine utvrđuje tehnikom vektorisanog mehanizma prekida.

Organizacija procesora je tako odabrana da je čine dve odvojene jedinice i to operaciona jedinica i upravljačka jedinica. Operaciona jedinica se sastoji od blokova povezivanje na magistralu, čitanje instrukcije, formiranje adrese i čitanje operanda, izvršavanje operacija i opsluživanje prekida, međusobno povezanih direktnim vezama. Upravljačka jedinica je realizovana tehnikom ožičene realizacije sa brojačem koraka i dekoderom.

Ulazno/izlaznih uređaja ima 8. Ulazno/izlazni uređaj ***U/I0*** do ***U/I7***se sastoji od periferije i kontrolera periferije bez direktnog pristupa memoriji. Procesor po linijama $\mathbf{intr_0}$ do $\mathbf{intr_7}$ dobija signale maskirajućih zahteva za prekid od ulazno/izlaznih uređaja ***U/I0*** do ***U/I7***, respektivno.

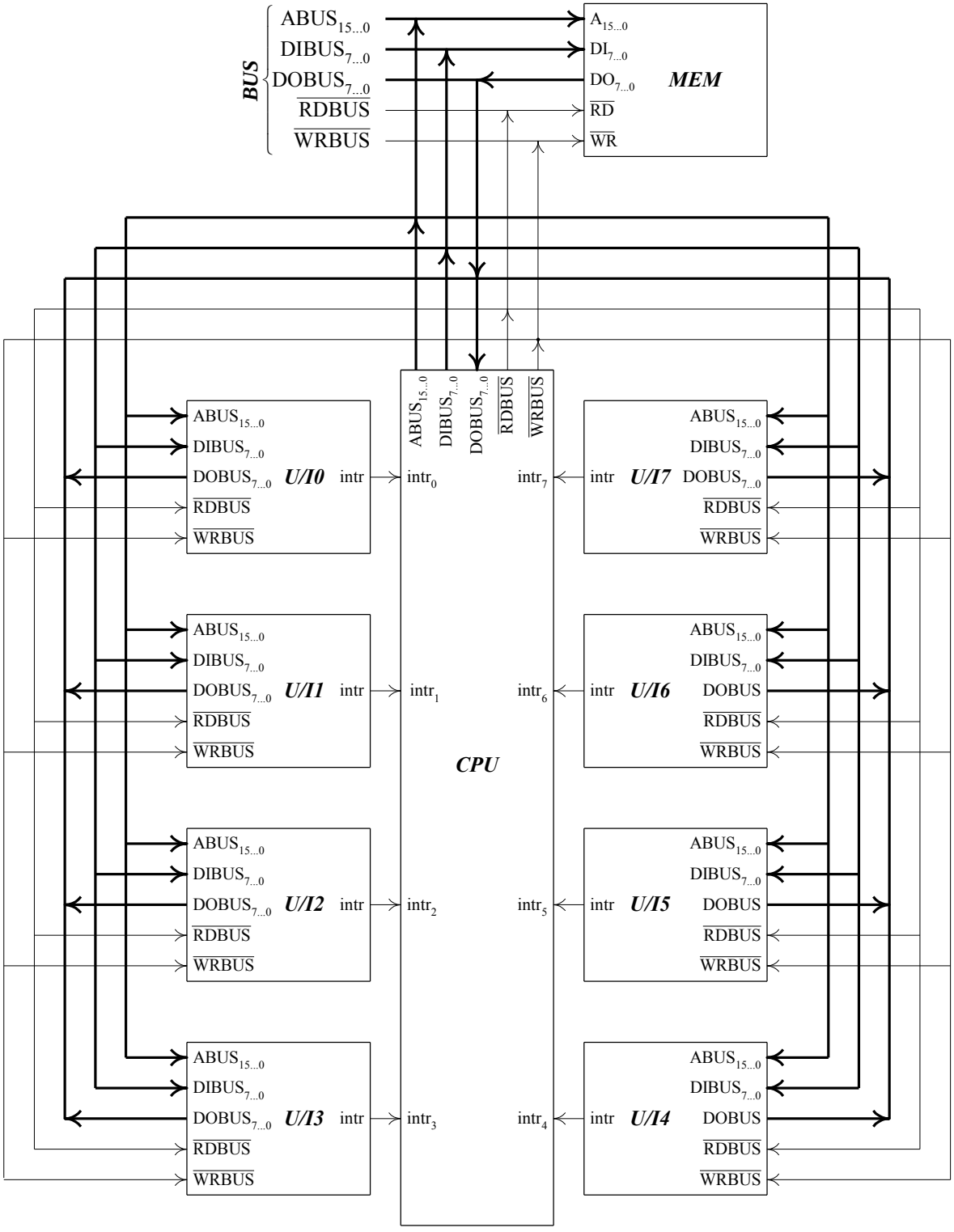
Ulazno/izlazni adresni prostor je memorijski preslikan. Opseg adresa od 0 do 60K-1 se koristi za adresiranje memorijskih lokacija, dok se opseg adresa 60K do 64K-1 koristi za adresiranje registara po kontrolerima periferija ulazno/izlaznih uređaja ***U/I0*** do ***U/I7***.

Memorija je kapaciteta 60K 8–mo bitnih reči.

Procesor, memorija i ulazno/izlazni uređaji su povezani asinhronom sistemskom magistralom ***BUS*** koju čine adresne linije $ABUS_{15...0}$, ulazne linije podataka $DIBUS_{7...0}$, izlazne linije podataka $DOBUS_{7...0}$ i upravljačke linije $\overline{\mathbf{RDBUS}}$ i $\overline{\mathbf{WRBUS}}$. Na magistrali mogu da se realizuju ciklus čitanja i ciklus upisa. Procesor realizuje cikluse čitanja prilikom čitanja instrukcija i podataka iz memorije i prilikom čitanja statusnih informacija i podataka iz registara kontrolera ulazno/izlaznih uređaja. Procesor realizuje cikluse upisa prilikom upisa podataka u memoriju i prilikom upisa upravljačkih informacija i podataka u registare kontrolera ulazno/izlaznih uređaja.

Uređaj koji započinje neki ciklusa na magistrali naziva se gazda, a uređaj koji realizuje ciklus sluga. Pri ciklusu čitanja gazda šalje adresu na adresne linije $ABUS_{15...0}$ i vrednošću 0 signala na upravljačkoj liniji $\overline{\mathbf{RDBUS}}$ startuje čitanje u slugi. Po završenom čitanju sluga šalje očitani podatak na linije podataka $DOBUS_{7...0}$. Pri ciklusu upisa gazda šalje adresu na adresne linije $ABUS_{15...0}$, podatak na linije podataka $DIBUS_{17...0}$ i vrednošću 0 signala na

upravljačkoj liniji **$\overline{\text{WRBUS}}$** startuje upis u slugi. Ciklusi čitanja i upisa traju jednu periodu signala takta.



Slika 1 Konfiguracija sistema

.

2 ARHITEKTURA

U ovoj glavi se razmatra arhitektura računarskog sistema. Pri tome se pod arhitekturom računarskog sistema podrazumevaju svi oni njegovi elementi koji treba da budu poznati da bi za njega mogao da se napiše asemblerski program koji će se uspešno izvršavati i uvek davati isti rezultat bez obzira na to kako je dati računarski sistem realizovan. U okviru arhitekture računarskog sistema razmatraju se arhitekture procesora, memorije i ulazno/izlaznih uređaja.

2.1 PROCESOR

Arhitekturu procesora čine:

- programski dostupni registri,
- tipovi podataka,
- formati instrukcija,
- načini adresiranja,
- skup instrukcija i
- mehanizam prekida.

U daljem tekstu biće razmotren svaki od ovih elemenata arhitekture procesora.

2.1.1 Programski dostupni registri

Programski dostupni registri procesora su:

- programski brojač PC,
- akumulator AB,
- akumulator AW,
- 32 registra opšte namene GPR,
- ukazivač na vrh steka SP,
- programska statusna reč PSW i
- ukazivač na tabelu adresa prekidnih rutina IVTP.

Registar PC je standardni 16-to razredni programski brojač procesora. Adrese generisane na osnovu vrednosti registra PC su adrese 8-mo bitnih reči. Njime se mogu generisati adrese unutar adresnog prostora od 2^{16} 8-mo bitnih reči.

Registar AB je 8-mo razredni akumulator za operacije prenosa, aritmetičke operacije, logičke operacije i operacije pomeranja i rotiranja nad 8-mo bitnim veličinama u kojima se koristi kao implicitno izvorište i/ili odredište operanda.

Registar AW je 16-to razredni akumulator za operacije prenosa 16-to bitnih veličina u kojima se koristi kao implicitno izvorište ili odredište operanda.

32 registra opšte namene su 16-to razredni registri koji se koriste kao registri podataka kod direktnog registarskog adresiranja, adresni registri kod registarskog indirektnog adresiranja, bazno/indeksni registri kod registarskog indirektnog adresiranja sa pomerajem, i bazni i indeksni registri kod bazno/indeksnog adresiranja sa pomerajem. Donjih osam razreda ovih registara se koristi kod direktnog registarskog adresiranja u operacijama za rad sa 8-mo bitnim veličinama.

Registar SP je standardan 16-to razredni ukazivač na vrh steka. Stek je organizovan u operativnoj memoriji i raste prema višim adresama. Registar SP ukazuje na poslednju zauzetu

lokaciju na steku. Adrese generisane na osnovu vrednosti registra SP su adrese 8-mo bitnih reči. Njime se mogu generisati adrese unutar adresnog prostora od 2^{16} 8-mo bitnih reči.

Registar PSW je standardna 16-to razredni programska statusna reč procesora čiji razredi, koji se obično nazivaju indikatori, sadrže bitove statusnog i upravljačkog karaktera (slika 2).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I	-	-	-	-	-	-	-	-	-	-	-	V	C	Z	N

Slika 2 Struktura registra PSW

Bitovi statusnog karaktera su:

- N—bit (indikator *negative*) koji se postavlja na 1 kada je rezultat operacije negativan,
- Z—bit (indikator *zero*) koji se postavlja na 1 kada je rezultat operacije nula,
- C—bit (indikator *carry/borrow*) koji se postavlja na 1 kada je takav rezultat operacije da postoji prenos/pozajmica u aritmetici celobrojnih veličina bez znaka,
- V—bit (indikator *overflow*) koji se postavlja na 1 kada je takav rezultat operacije da postoji prekoračenje u aritmetici celobrojnih veličina sa znakom i

Bitovi upravljačkog karaktera su:

- I—bit (indikator *interrupt*) koji se postavlja na 1 kada se zada režim rada procesora sa prihvatanjem maskirajućih prekida.

Bitovi statusnog karaktera N, Z, C i V se postavljaju hardverski na osnovu rezultata izvršavanja instrukcija. Bit upravljačkog karaktera I se postavlja softverski kao rezultat izvršavanja posebnih instrukcija i softverski kao rezultat izvršavanja instrukcije povratak iz prekidne rutine.

Registar IVTP je standardni 16-to razredni ukazivač na tabelu adresa prekidnih rutina koja se koristi u okviru vektorisanog mehanizma prekida. Adresa generisana na osnovu vrednosti registra IVTP je adresa 8-mo bitne reči.

2.1.2 Tipovi podataka

Tipovi podataka koji se koriste u ovom procesoru su celobrojne 8-mo bitne veličine sa znakom i bez znaka, 8-mo bitne binarne reči i 16-to bitne binarne reči. Celobrojne 8-mo bitne veličine sa znakom i bez znaka se koriste kod operacije prenosa, aritmetičkih operacija i operacija pomeranja i rotiranja nad 8-mo bitnim veličinama, 8-mo bitne binarne reči se koriste kod logičkih operacija i operacija pomeranja i rotiranja i 16-to bitne binarne reči se koriste kod operacija prenosa.

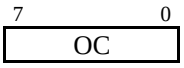
2.1.3 Formati instrukcija

U procesoru se koriste sledeći formati instrukcija:

- format bezadresnih instrukcija,
- format instrukcija relativnog skoka – B format,
- format instrukcija apsolutnog skoka – J format,
- format jednoadresnih registarskih instrukcija – R format,
- format jednoadresnih neposrednih instrukcija – IB format,
- format jednoadresnih neposrednih instrukcija – IW format i
- format jednoadresnih memorijskih instrukcija – AP format.

2.1.3.1 Format bezadresnih instrukcija

Format ovih instrukcija je dat na slici 3.

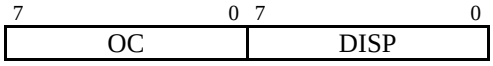


Slika 3 Format bezadresnih instrukcija

Poljem OC se specificira operacija koja se izvršava kao i registri koji se eventualno implicitno koriste.

2.1.3.2 Format instrukcija relativnog skoka – B format

Format ovih instrukcija je dat na slici 4.



Slika 4 Format instrukcija relativnog skoka – B format

Poljem OC se specificira operacija koja se izvršava, a poljem DISP pomeraj koji se sabira sa PC da bi se dobila adresa skoka. Pomeraj je 8-mo bitna celobrojna veličina sa znakom.

2.1.3.3 Format instrukcija apslolutnog skoka – J format

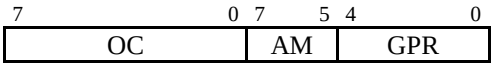
Format ovih instrukcija je dat na slici 5.

Slika 5 Format instrukcija relativnog skoka – J format

Poljem OC se specificira operacija koja se izvršava, a poljima DISPH i DISPL 8 starijih i 8 mlađih bitova 16-to bitne adrese skoka.

2.1.3.4 Format jednoadresnih registarskih instrukcija – R format

Format ovih instrukcija je dat na slici 6.

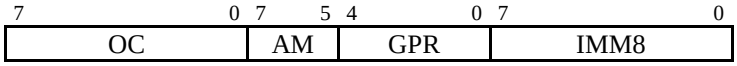


Slika 6 Format jednoadresnih registarskih instrukcija – R format

Poljem OC se specificira kod operacije jednoadresne instrukcije, polje AM registarsko direktno ili registarsko indirektno adresiranje i poljem GPR jedan od 32 registra opšte namene.

2.1.3.5 Format jednoadresnih neposrednih instrukcija – IB format

Format ovih instrukcija je dat na slici 7.

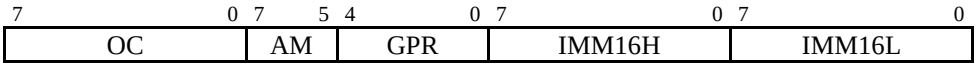


Slika 7 Format jednoadresnih instrukcija – IB format

Poljem OC se specificira kod operacije jednoadresne instrukcije nad 8-mo bitnim veličinama, polje AM neposredno adresiranje, polje GPR se ne koristi i poljem IMM8 neposredna 8-mo bitna veličina.

2.1.3.6 Format jednoadresnih instrukcija – IW format

Format ovih instrukcija je dat na slici 8.

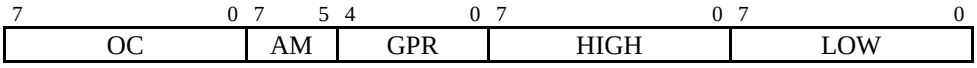


Slika 8 Format jednoadresnih instrukcija – IW format

Poljem OC se specificira kod operacije jednoadresne instrukcije nad 16-to bitnim veličinama, polje AM neposredno adresiranje, polje GPR se ne koristi i poljima IMM16H i IMM16L 8 starijih i 8 mlađih bitova neposredne 16-to bitne veličine.

2.1.3.7 Format jednoadresnih memorijskih instrukcija – AP format

Format ovih instrukcija je dat na slici 9.



Slika 9 Format jednoadresnih instrukcija – AP format

Poljem OC se specificira kod operacije jednoadresne instrukcije nad 8-mo bitnim veličinama, polje AM memorijsko direktno, memorijsko indirektno, registarsko indirektno sa pomerajem, bazno indeksno i PC relativno adresiranje. Polje GPR se ne koristi kod memorijskog direktnog i memorijsko indirektnog adresiranja dok kod registarsko indirektnog sa pomerajem, bazno indeksnog i PC relativnog adresiranje predstavlja registar opšte namene. Polja HIGH i LOW predstavljaju 8 starijih i 8 mlađih bitova 16-to bitne veličine koja predstavlja memorijsku adresu za memorijsko direktno i memorijsko indirektno adresiranje i 16-to bitni pomeraj za registarsko indirektno sa pomerajem, bazno indeksno i PC relativno adresiranje.

2.1.4 Načini adresiranja

Procesor poseduje 8 različitih načina adresiranja. Kodiranje polja AM za različite načine adresiranja i nazivi načina adresiranja dati su u tabeli 1.

Tabela 1 Načini adresiranja

AM	**Načini adresiranja**
000	registarsko direktno
001	registarsko indirektno
010	memorijsko direktno
011	memorijsko indirektno
100	registarsko indirektno sa pomerajem
101	bazno indeksno sa pomerajem
110	PC relativno sa pomerajem
111	neposredno

Registarsko direktno adresiranje je adresiranje kod koga se operand nalazi u jednom od registara opšte namene. Instrukcija sa registarskim direktnim adresiranjem ima R format (poglavlje 2.1.3.4). Registar opšte namene je specificiran poljem GPR. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 16-to bitnim veličinama ili 8-mo bitnim veličinama koristi se svih 16 ili 8 nižih razreda registra.

Registarsko indirektno adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi određenoj sadržajem jednog od registara opšte namene. Instrukcija sa registarskim direktnim adresiranjem ima R format (poglavlje 2.1.3.4). Polje GPR specificira registar opšte

namene. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

Memorijsko direktno adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi datoj u samoj instrukciji. Instrukcija sa registarskim direktnim adresiranjem ima AP format (poglavlje 2.1.3.7). Polje GPR se ne koristi. Polja HIGH i LOW sadrže starijih 8 i mlađih 8 bitova adrese. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

Memorijsko indirektno adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi određenoj sadržajem memorijske lokacije čija je adresa data u samoj instrukciji. Instrukcija sa memorijskim indirektnim adresiranjem ima AP format (poglavlje 2.1.3.7). Polje GPR se ne koristi. Polja HIGH i LOW sadrže starijih 8 i mlađih 8 bitova adrese sa koje se čita adresa operanda. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

Registarsko indirektno sa pomerajem adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi koja se dobija sabiranjem sadržaja jednog od registara opšte namene i pomeraja. Instrukcija sa registarskim indirektnim sa pomerajem adresiranjem ima AP format (poglavlje 2.1.3.7). Polje GPR specificira registar opšte namene. Polja HIGH i LOW sadrže starijih 8 i mlađih 8 bitova pomeraja. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

Bazno indeksno sa pomerajem adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi koja se dobija sabiranjem sadržaja dva registara opšte namene i pomeraja. Instrukcija sa baznim indeksnim sa pomerajem adresiranjem ima AP format (poglavlje 2.1.3.7). Polje GPR specificira adresu sa koje se čita registar opšte namene koji se koristi kao bazni registar, dok se registar opšte namene koji se koristi kao indeksni registar čita sa prve sledeće adrese. Polja HIGH i LOW sadrže starijih 8 i mlađih 8 bitova pomeraja. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

PC relativno sa pomerajem adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi koja se dobija sabiranjem sadržaja programskog brojača PC i pomeraja. Instrukcija sa PC relativnim sa pomerajem adresiranjem ima AP format (poglavlje 2.1.3.7). Polje GPR se ne koristi. Polja HIGH i LOW sadrže starijih 8 i mlađih 8 bitova pomeraja. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

Neposredno adresiranje je adresiranje kod koga se operand nalazi u samoj instrukciji. Instrukcija sa neposrednim adresiranjem u zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama ima IB format (poglavlje 2.1.3.5) ili IW format (poglavlje 2.1.3.6). U oba slučaja polje GPR se ne koristi. U slučaju IB formata polje IMM8 predstavlja 8-mo bitni podatak, a u slučaju IW formata polja IMM16H i IMM16L sadrže starijih 8 i mlađih 8 bitova 16-to bitnog podatka.

2.1.5 Skup instrukcija

U ovom poglavlju se, najpre, daje opis instrukcija, a zatim tabelarni pregled kodiranja instrukcija.

2.1.5.1 Opis instrukcija

Instrukcije procesora se mogu svrstati u sledećih šest grupa:

- instrukcije skoka,
- instrukcije prenosa,
- aritmetičke instrukcije,
- logičke instrukcije,
- instrukcije pomeranja i rotiranja,
- instrukcije postavljanja indikatora u PSW,

2.1.5.1.1 Instrukcije skoka

Instrukcije skoka se svrstavaju u sledeće grupe:

- instrukcije uslovnog skoka,
- instrukcije bezuslovnog skoka,
- instrukcije skoka na potprogram i povratka iz potprograma i
- instrukcija povratka iz prekidne rutine

2.1.5.1.1.1 Instrukcije uslovnog skoka

Instrukcije uslovnog skoka **BEQL** *disp*, **BNEQL** *disp*, **BNEG** *disp*, **BNNEG** *disp*, **BOVF** *disp*, **BNOVF** *disp*, **BCAR** *disp*, **BNCAR** *disp*, **BGRT** *disp*, **BGRTE** *disp*, **BLSS** *disp*, **BLSSE** *disp*, **BGRTU** *disp*, **BGRTEU** *disp*, **BLSSU** *disp* i **BLSSEU** *disp* realizuju relativni skok sa pomerajem *disp* u odnosu na programski brojač PC ukoliko je uslov specificiran kodom operacije ispunjen (tabela 2). Pomeraj *disp* je 8-mo bitna celobrojna veličina sa znakom. Format ovih instrukcija je dat u poglavlju 2.1.3.2. Ni jedna od ovih instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Tabela 2 Instrukcije uslovnog skoka

Instrukcija	**Značenje**	**Uslov**
BEQL	skok na jednako	Z = 1
BNEQ	skok na nejednako	Z = 0
BNEG	skok na N = 1	N = 1
BNNG	skok na N = 0	N = 0
BOVF	skok na V = 1	V = 1
BNVF	skok na V = 0	V = 0
BCR	skok na C = 1	C = 1
BNCR	skok na C = 0	C = 0
BGRT	skok na veće nego (sa znakom)	$(N \oplus V) \vee Z = 0$
BGRE	skok na veće nego ili jednako (sa znakom)	$N \oplus V = 0$
BLSS	skok na manje nego (sa znakom)	$(N \oplus V) = 1$
BLEQ	skok na manje nego ili jednako (sa znakom)	$(N \oplus V) \vee Z = 1$
BGRTU	skok na veće nego (bez znaka)	$C \vee Z = 0$
BGREU	skok na veće nego ili jednako (bez znaka)	C = 0
BLSSU	skok na manje nego (bez znaka)	C = 1
BLEQU	skok na manje nego ili jednako (bez znaka)	$C \vee Z = 1$

2.1.5.1.1.2 Instrukcija bezuslovnog skoka

Instrukcija bezuslovnog skoka **JMP** *a* realizuje skok na adresu *a* koja je data u samoj instrukciji. Format ove instrukcije je dat u poglavlju 2.1.3.3. Ova instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

2.1.5.1.1.3 Instrukcije skoka na potprogram i povratka iz potprograma

Instrukcija **JSR** *a* realizuje skok na potprogram tako što čuva vrednost programskog brojača PC na steku i realizuje skok na adresu *a* koja je data u samoj instrukciji. Format ove instrukcije je dat u odeljku 2.1.3.3.

Instrukcija **RTS** realizuje povratak iz potprograma tako što restaurira vrednost programskog brojača PC vrednošću sa steka. Format ove instrukcije je dat u poglavlju 2.1.3.1.

Ni jedna od ovih instrukcija ne utiče na indikatore N, Z, C i V iz registra PSW.

2.1.5.1.1.4 Instrukcije povratka iz prekidne rutine

Instrukcija **RTI** realizuje povratak iz prekidne rutine tako što restaurira vrednosti programske statusne reči PSW i programskog brojača PC vrednostima sa steka. Format ove instrukcije je dat u poglavlju 2.1.3.1.

Ni jedna od ovih instrukcija ne utiče na indikatore N, Z, C i V iz registra PSW.

2.1.5.1.2 Instrukcije prenosa

Instrukcija **LDB** *src* prenosi sadržaj 8-bitnog operanda *src* u akumulator AB. Kao operand *src*, u slučaju direktnog registarskog adresiranja, se koristi samo 8 nižih razreda adresiranog registra opšte namene. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4, 2.1.3.5 i 2.1.3.7. Istrukcija postavlja indikatore N i Z registra PSW saglasno vrednosti koja se upisuje u akumulator AB, dok se indikatori C i V registra PSW postavljaju na 0.

Instrukcija **LOADW** *src* prenosi sadržaj 16-bitnog operanda *src* u akumulator AW. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4, 2.1.3.6 i 2.1.3.7. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **STB** *dst* prenosi sadržaj akumulatora AB u 8-bitni operand *dst*. Kao operand *dst*, u slučaju direktnog registarskog adresiranja, se koristi samo 8 nižih razreda adresiranog registra opšte namene. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4 i 2.1.3.7. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **STW** *dst* prenosi sadržaj akumulatora AW u 16-bitni operand *dst*. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4 i 2.1.3.7. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **POPB** prenosi sadržaj 8-bitnog operanda sa vrha steka u akumulatora AB. Format ove instrukcije dat je u poglavlju 2.1.3.1. Istrukcija postavlja indikatore N i Z registra PSW saglasno vrednosti koja se upisuje u akumulator AB, dok se indikatori C i V registra PSW postavljaju na 0.

Instrukcija **POPW** prenosi sadržaj 16-bitnog operanda sa vrha steka u akumulatora AW. Format ove instrukcije dat je u poglavlju 2.1.3.1. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **PUSHB** prenosi sadržaj akumulatora AB u 8-bitni operand na vrh steka. Format ove instrukcije dat je u poglavlju 2.1.3.1. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **PUSHW** prenosi sadržaj akumulatora AW 16-bitni operand na vrh steka. Format ove instrukcije dat je u poglavlju 2.1.3.1. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **LDIVTP** prenosi sadržaj registra IVTP u akumulator AW. Format ove instrukcije dat je u poglavlju 2.1.3.1. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **STIVTP** prenosi sadržaj akumulatora AW u registar IVTP. Format ove instrukcije dat je u poglavlju 2.1.3.1. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **LOADSP** prenosi sadržaj registra SP u akumulator AW. Format ove instrukcije dat je u poglavlju 2.1.3.1. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **STORESP** prenosi sadržaj akumulatora AW u registar SP. Format ove instrukcije dat je u poglavlju 2.1.3.1. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

2.1.5.1.3 Aritmetičke instrukcije

Instrukcija **ADD** *src* sabira celobrojnu 8-bitnu veličinu koja se nalazi u akumulatoru AB sa operandom *src* koji je celobrojna 8-bitna veličina, a rezultat koji je celobrojna 8-bitna veličina smešta u akumulator AB. Kao operand *src*, u slučaju direktnog registarskog adresiranja, se koristi samo 8 nižih razreda adresiranog registra opšte namene. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4, 2.1.3.5 i 2.1.3.7.

Instrukcija **SUB** *src* oduzima operand *src* koji je celobrojna 8-bitna veličina od celobrojne 8-bitne veličine koja se nalazi u akumulatoru AB, a rezultat koji je celobrojna 8-bitna veličina smešta u akumulator AB. Kao operand *src*, u slučaju direktnog registarskog adresiranja, se koristi samo 8 nižih razreda adresiranog registra opšte namene. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4, 2.1.3.5 i 2.1.3.7.

Instrukcija **INC** inkrementira celobrojnu 8-bitnu veličinu koja se nalazi u akumulatoru AB i rezultat koji je celobrojna 8-bitna veličina smešta u akumulator AB. Format instrukcije dat je u poglavlju 2.1.3.1.

Instrukcija **DEC** dekrementira celobrojnu 8-bitnu veličinu koja se nalazi u akumulatoru AB i rezultat koji je celobrojna 8-bitna veličina smešta u akumulator AB. Format instrukcije dat je u poglavlju 2.1.3.1.

Istrukcije postavljaju indikatore N, Z, C i V registra PSW saglasno dobijenoj vrednosti koja se upisuje u akumulator AB.

2.1.5.1.4 Logičke instrukcije

Instrukcija **AND** *src* izračunava logičko I 8-bitne veličine koja se nalazi u akumulatoru AB i operanda *src* koji je 8-bitna veličina, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Kao operand *src*, u slučaju direktnog registarskog adresiranja, se koristi samo 8 nižih razreda adresiranog registra opšte namene. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4, 2.1.3.5 i 2.1.3.7.

Instrukcija **OR** *src* izračunava logičko ILI 8-bitne veličine koja se nalazi u akumulatoru AB i operanda *src* koji je 8-bitna veličina, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Kao operand *src*, u slučaju direktnog registarskog adresiranja, se koristi samo 8 nižih

razreda adresiranog registra opšte namene. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4, 2.1.3.5 i 2.1.3.7.

Instrukcija **XOR** *src* izračunava logičko EKSLUZIVNO ILI 8-bitne veličine koja se nalazi u akumulatoru AB i operanda *src* koji je 8-bitna veličina, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Kao operand *src*, u slučaju direktnog registarskog adresiranja, se koristi samo 8 nižih razreda adresiranog registra opšte namene. Formati ove instrukcije zavise od specificiranog načina adresiranja i dati su u poglavljima 2.1.3.4, 2.1.3.5 i 2.1.3.7.

Instrukcija **NOT** izračunava logičku NEGACIJU 8-bitne veličine koja se nalazi u akumulatoru AB, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Format instrukcije dat je u poglavlju 2.1.3.1.

Istrukcije postavljaju indikatore N i Z registra PSW saglasno vrednosti koja se upisuje u akumulator AB, dok se indikatori C i V registra PSW postavljaju na 0.

2.1.5.1.5 Instrukcije pomeranja

Instrukcija **ASR** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB udesno za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Pri pomeranju razred AB_7 ostaje nepromenjen, a u indikator C registra PSW se upisuje sadržaj razreda AB_0. Format instrukcije dat je u poglavlju 2.1.3.1.

Instrukcija **LSR** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB udesno za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Pri pomeranju u razred AB_7 se upisuje 0, a u indikator C registra PSW se upisuje sadržaj razreda AB_0. Format instrukcije dat je u poglavlju 2.1.3.1.

Instrukcija **ROR** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB udesno za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Pri pomeranju u razred AB_7 se upisuje sadržaj razreda AB_0, a u indikator C registra PSW se upisuje sadržaj razreda AB_0. Format instrukcije dat je u poglavlju 2.1.3.1.

Instrukcija **RORC** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB udesno za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Pri pomeranju u razred AB_7 se upisuje sadržaj indikatora C registra PSW, a u indikator C registra PSW se upisuje sadržaj razreda AB_0. Format instrukcije dat je u poglavlju 2.1.3.1.

Instrukcija **ASL** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB ulevo za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Pri pomeranju u razred AB_0 se upisuje 0, a u indikator C registra PSW se upisuje sadržaj razreda AB_7. Format instrukcije dat je u poglavlju 2.1.3.1.

Instrukcija **LSL** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB ulevo za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Pri pomeranju u razred AB_0 se upisuje 0, a u indikator C registra PSW se upisuje sadržaj razreda AB_7. Format instrukcije dat je u poglavlju 2.1.3.1.

Instrukcija **ROL** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB ulevo za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Pri pomeranju u razred AB_0 se upisuje sadržaj razreda AB_7, a u indikator C registra PSW se upisuje sadržaj razreda AB_7. Format instrukcije dat je u poglavlju 2.1.3.1.

Instrukcija **ROLC** pomera sadržaj 8-bitne veličine koja se nalazi u akumulatoru AB ulevo za jedno mesto, a rezultat koji je 8-bitna veličina smešta u akumulator AB. Pri pomeranju u

razred AB_0 se upisuje sadržaj indikatora C registra PSW, a u indikator C registra PSW se upisuje sadržaj razreda AB_7. Format instrukcije dat je u poglavlju 2.1.3.1.

Istrukcije postavljaju indikatore N i Z registra PSW saglasno vrednosti koja se upisuje u akumulator AB, dok se indikator V registra PSW postavlja na 0.

2.1.5.1.6 Instrukcije postavljanja indikatora u PSW

Instrukcija **INTD** postavlja nulu u razred I registra PSW. Format instrukcije je dat u poglavlju 2.1.3.1.

Instrukcija **INTE** postavlja jedinicu u razred I registra PSW. Format instrukcije je dat u poglavlju 2.1.3.1.

Ni jedna od ovih instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

2.1.5.2 Kodiranje instrukcija

Kodiranje instrukcija je izvršeno na takav način da bitovi:

- $OC_{7...6}$ predstavljaju četiri grupe od 64 instrukcije,
- $OC_{5...3}$ predstavljaju osam podgrupa od 8 instrukcija i
- $OC_{2...0}$ predstavljaju instrukciju u okviru podgrupe instrukcija.

Kodiranje četiri grupe instrukcija G0 do G3 je dato u tabeli 3.

Tabela 3 Kodiranje četiri grupe instrukcija

$OC_{7...6}$	**Oznaka**	**Napomena**
00	G0	Koristi se
01	G1	Ne koristi se
10	G2	Ne koristi se
11	G3	Ne koristi se

Kodiranje osam podgrupa instrukcija G0_PG0 do G0_PG7 u okviru grupe instrukcija G0 je dato u tabeli 4.

Tabela 4 Kodiranje osam podgrupa instrukcija iz grupe G0

$OC_{5...3}$	**Oznaka**	**Napomena**
000	G0_PG0	Instrukcija instrukcije postavljanja indikatora u PSW (poglavlje 2.1.5.1.6)
001	G0_PG1	Instrukcija bezuslovnog skoka (poglavlje 2.1.5.1.1.2), instrukcije skoka na potprogram i povratka iz potprograma (poglavlje 2.1.5.1.1.3), instrukcija povratka iz prekidne rutine (poglavlje 2.1.5.1.1.4)
010	G0_PG2	Instrukcije uslovnog skoka (poglavlje 2.1.5.1.1.1)
011	G0_PG3	Instrukcije uslovnog skoka (poglavlje 2.1.5.1.1.1)
100	G0_PG4	Instrukcije prenosa (poglavlje 2.1.5.1.2)
101	G0_PG5	Instrukcije prenosa (poglavlje 2.1.5.1.2)
110	G0_PG6	Aritmetičke instrukcije (poglavlje 2.1.5.1.3) i logičke instrukcije (poglavlje 2.1.5.1.4)
111	G0_PG7	Instrukcije pomeranja i rotiranja (poglavlje 2.1.5.1.5)

Kodiranje instrukcija u okviru osam podgrupa instrukcija G0_PG0 do G0_PG7 je dato u tabelama 5 do 12. U svakoj tabeli su date binarne vrednosti tri najmlađa bita **$OC_{2...0}$** polja koda operacije kojima se određuje instrukcija unutar date podgrupe instrukcija, oznaka za svaku instrukcije, dužina instrukcije u bajtovima i poglavlje u kome je opisana svaka instrukcija.

Kodiranje podgrupe instrukcija G0_PG0 koju čine instrukcije postavljanja indikatora u PSW (poglavlje 2.1.5.1.6) je dato u tabeli 5.

Tabela 5 Kodiranje podgrupe instrukcija G0_PG0

$OC_{2...0}$	Oznaka	Dužina	Poglavlje
000	-	-	-
001	-	-	-
010	-	-	-
011	-	-	-
100	**INTD**	1	2.1.5.1.6
101	**INTE**	1	2.1.5.1.6
110	-	-	-
111	-	-	-

Kodiranje podgrupe instrukcija G0_PG1 koju čine instrukcija bezuslovnog skoka (poglavlje 2.1.5.1.1.2), instrukcije skoka na potprogram i povratka iz potprograma (poglavlje 2.1.5.1.1.3) i instrukcija povratka iz prekidne rutine (poglavlje 2.1.5.1.1.4) je dato u tabeli 6.

Tabela 6 Kodiranje podgrupe instrukcija G0_PG1

$OC_{2...0}$	Oznaka	Dužina	Poglavlje
000	-	-	-
001	**JMP**	3	2.1.5.1.1.2
010	**JSR**	3	2.1.5.1.1.3
011	**RTS**	1	2.1.5.1.1.3
100	-	-	-
101	**RTI**	1	2.1.5.1.1.4
110	-	-	-
111	-	-	-

Kodiranje podgrupe instrukcija G0_PG2 koju čine instrukcije uslovnog skoka (poglavlje 2.1.5.1.1.1) je dato u tabeli 7.

Tabela 7 Kodiranje podgrupe instrukcija G0_PG2

$OC_{2...0}$	Oznaka	Dužina	Poglavlje
000	**BEQL**	2	2.1.5.1.1.1
001	**BNEQL**	2	2.1.5.1.1.1
010	**BNEG**	2	2.1.5.1.1.1
011	**BNNEG**	2	2.1.5.1.1.1
100	**BOVF**	2	2.1.5.1.1.1
101	**BNOVF**	2	2.1.5.1.1.1
110	**BCAR**	2	2.1.5.1.1.1
111	**BNCAR**	2	2.1.5.1.1.1

Kodiranje podgrupe instrukcija G0_PG3 koju čine instrukcije uslovnog skoka (poglavlje 2.1.5.1.1.1) je dato u tabeli 8.

Tabela 8 Kodiranje podgrupe instrukcija G0_PG3

$OC_{2...0}$	Oznaka	Dužina	Poglavlje
000	**BGRT**	2	2.1.5.1.1.1
001	**BGRTE**	2	2.1.5.1.1.1
010	**BLSS**	2	2.1.5.1.1.1
011	**BLSSE**	2	2.1.5.1.1.1
100	**BGRTU**	2	2.1.5.1.1.1
101	**BGRTEU**	2	2.1.5.1.1.1
110	**BLSSU**	2	2.1.5.1.1.1
111	**BLSSEU**	2	2.1.5.1.1.1

Kodiranje podgrupe instrukcija G0_PG4 koju čine instrukcije instrukcije prenosa (poglavlje 2.1.5.1.2) je dato u tabeli 9.

Tabela 9 Kodiranje podgrupe instrukcija G0_PG4

$OC_{2...0}$	Oznaka	Dužina	Poglavlje
000	**LDB**	2, 3 ili 4	2.1.5.1.2
001	**LDW**	2 ili 4	2.1.5.1.2
010	**STB**	2 ili 4	2.1.5.1.2
011	**STW**	2 ili 4	2.1.5.1.2
100	**POPB**	1	2.1.5.1.2
101	**POPW**	1	2.1.5.1.2
110	**PUSHB**	1	2.1.5.1.6
111	**PUSHW**	1	2.1.5.1.6

Kodiranje podgrupe instrukcija G0_PG5 koju čine instrukcije instrukcije prenosa (poglavlje **Error! Reference source not found.**) je dato u tabeli 10.

Tabela 10 Kodiranje podgrupe instrukcija G0_PG5

$OC_{2...0}$	Oznaka	Dužina	Poglavlje
000	**LDIVTP**	1	2.1.5.1.4
001	**STIVTP**	1	2.1.5.1.4
010	-	-	-
011	-	-	-
100	**LDSP**	1	2.1.5.1.6
101	**STSP**	1	2.1.5.1.6
110	-	-	-
111	-	-	-

Kodiranje podgrupe instrukcija G0_PG6 koju čine aritmetičke instrukcije (poglavlje 2.1.5.1.3) i logičke instrukcije (poglavlje 2.1.5.1.4) je dato u tabeli 11.

Tabela 11 Kodiranje podgrupe instrukcija G0_PG6

$OC_{2...0}$	Oznaka	Dužina	Poglavlje
000	**ADD**	2, 3 ili 4	2.1.5.1.4
001	**SUB**	2, 3 ili 4	2.1.5.1.4
010	**INC**	1	2.1.5.1.4
011	**DEC**	1	2.1.5.1.4
100	**AND**	2, 3 ili 4	2.1.5.1.6
101	**OR**	2, 3 ili 4	2.1.5.1.6
110	**XOR**	2, 3 ili 4	2.1.5.1.6
111	**NOT**	1	2.1.5.1.6

Kodiranje podgrupe instrukcija G0_PG7 koju čine instrukcije pomeranja i rotiranja (poglavlje 2.1.5.1.5) je dato u tabeli 12.

Tabela 12 Kodiranje podgrupe instrukcija G0_PG7

$OC_{2...0}$	Oznaka	Dužina	Poglavlje
000	**ASR**	1	2.1.5.1.4
001	**LSR**	1	2.1.5.1.4
010	**ROR**	1	2.1.5.1.4
011	**RORC**	1	2.1.5.1.4
100	**ASL**	1	2.1.5.1.6
101	**LSL**	1	2.1.5.1.6
110	**ROL**	1	2.1.5.1.6
111	**ROLC**	1	2.1.5.1.6

2.1.6 Mehanizam prekida

Zahteve za prekid može da generiše osam kontrolera periferija po linijama $intr_7$ do $intr_0$ da bi signalizirali spremnost za prenos podataka. Ovi prekidi se nazivaju spoljašnji prekidi jer dolaze od uređaja van procesora. Ovi prekidi se nazivaju i maskirajući prekidi, jer su dozvoljeni ili maskirani i procesor na njih reaguje ili ne reaguje u zavisnosti od toga da li se u razredu I registra PSW nalazi vrednost 1 ili 0, respektivno. Dozvoljavanje i maskiranje prekida se realizuje programskim putem izvršavanjem instrukcija INTE i INTD (odeljak 2.1.5.1.6) kojima se u razred I registra PSW upisuju vrednosti 1 ili 0, respektivno.

Izvršavanje svake instrukcije pored faza čitanje instrukcije, formiranje adrese operanda i izvršavanje operacije sadrži i fazu opsluživanje zahteva za prekid. Na početku faze opsluživanje zahteva za prekid vrši se provera da li je u toku izvršavanja prethodne tri faze tekuće instrukcije stigao zahtev za prekid po nekoj od linija $intr_7$ do $intr_0$ i da li se u razredu I registra PSW nalazi vrednost 1. Ukoliko nije, vraća se na fazu čitanje instrukcije sledeće instrukcije. Ukoliko jeste, izvršavanje tekuće instrukcije se produžava sa koracima faze opsluživanje zahteva za prekid. Opsluživanje zahteva za prekid se sastoji iz dve grupe koraka.

U okviru prve grupe koraka na steku se čuvaju programski brojač PC i programska statusna reči PSW.

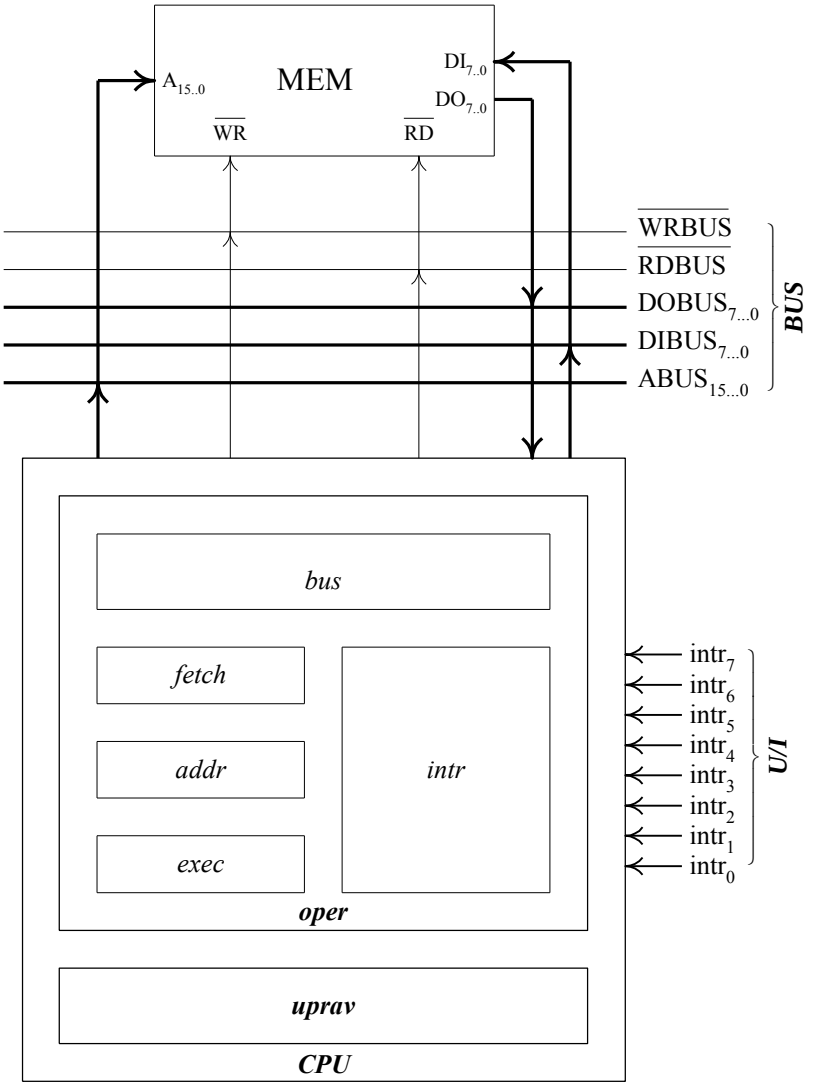
U okviru druge grupe koraka utvrđuje se adresa prekidne rutine. Utvrđivanje adrese prekidne rutine se realizuje na osnovu sadržaja tabele adresa prekidnih rutina koja se obično naziva IV tabela (*Interrupt Vector Table*) i broja ulaza u IV tabelu. Stoga je u postupku inicijalizacije celog sistema u memoriji, počev od adrese na koju ukazuje sadržaj registra procesora IVTP (*Interrupt Vector Table Pointer*), kreirana IV tabela sa 8 ulaza u kojima se nalaze adrese prekidnih rutina za svaki od prekida koji dolaze po linijama $intr_7$ do $intr_0$. Adrese prekidnih rutina za prekide koji dolaze po linijama $intr_7$ do $intr_0$ nalaze se u ulazima 7 do 0 IV tabele, respektivno. Prekidi koji dolaze po linijama $intr_7$ do $intr_0$ uređeni su po prioriteima pri čemu linija $intr_7$ ima najviši a linija $intr_0$ najniži nivo prioriteta. Broj ulaza u IV tabelu generiše procesor na osnovu pozicije linije $intr_7$ do $intr_0$ najvišeg nivoa prioriteta na kojoj postoji zahtev za prekid.

Kako je memorijska reč 8-mo bitna veličina a adresa prekidne rutine 16-to bitna veličina, to svaki ulaz u IV tabeli zauzima po dve susedne memorijske lokacije. Zbog toga se najpre broj ulaza množenjem sa dva pretvara u pomeraj, pa zatim pomeraj sabira sa sadržajem registra IVTP i na kraju dobijena vrednost koristiti kao adresu sa koje se čita adresa prekidne rutine i upisuje u registar PC.

Povratak iz prekidne rutine se realizuje instrukcijom **RTI**. Ovom instrukcijom se sa steka restauriraju registri PSW i PC.

3 ORGANIZACIJA

U ovoj glavi se daje organizacija procesora ***CPU*** koji se sastoji iz operacione jedinice ***oper*** i upravljačke jedinice ***uprav*** (slika 10). Operaciona jedinica ***oper*** je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova upravljačke jedinice ***uprav***. Upravljačka jedinica ***uprav*** je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala operacione jedinice ***oper*** na osnovu algoritama operacija i signala logičkih uslova. Struktura i opis operacione i upravljačke jedinice se daju u daljem tekstu.



Slika 10 Organizacija procesora ***CPU***

3.1 OPERACIONA JEDINICA

Operaciona jedinica ***oper*** (slika 10) se sastoji od sledećih blokova:

- blok *bus*,
- blok *fetch*,

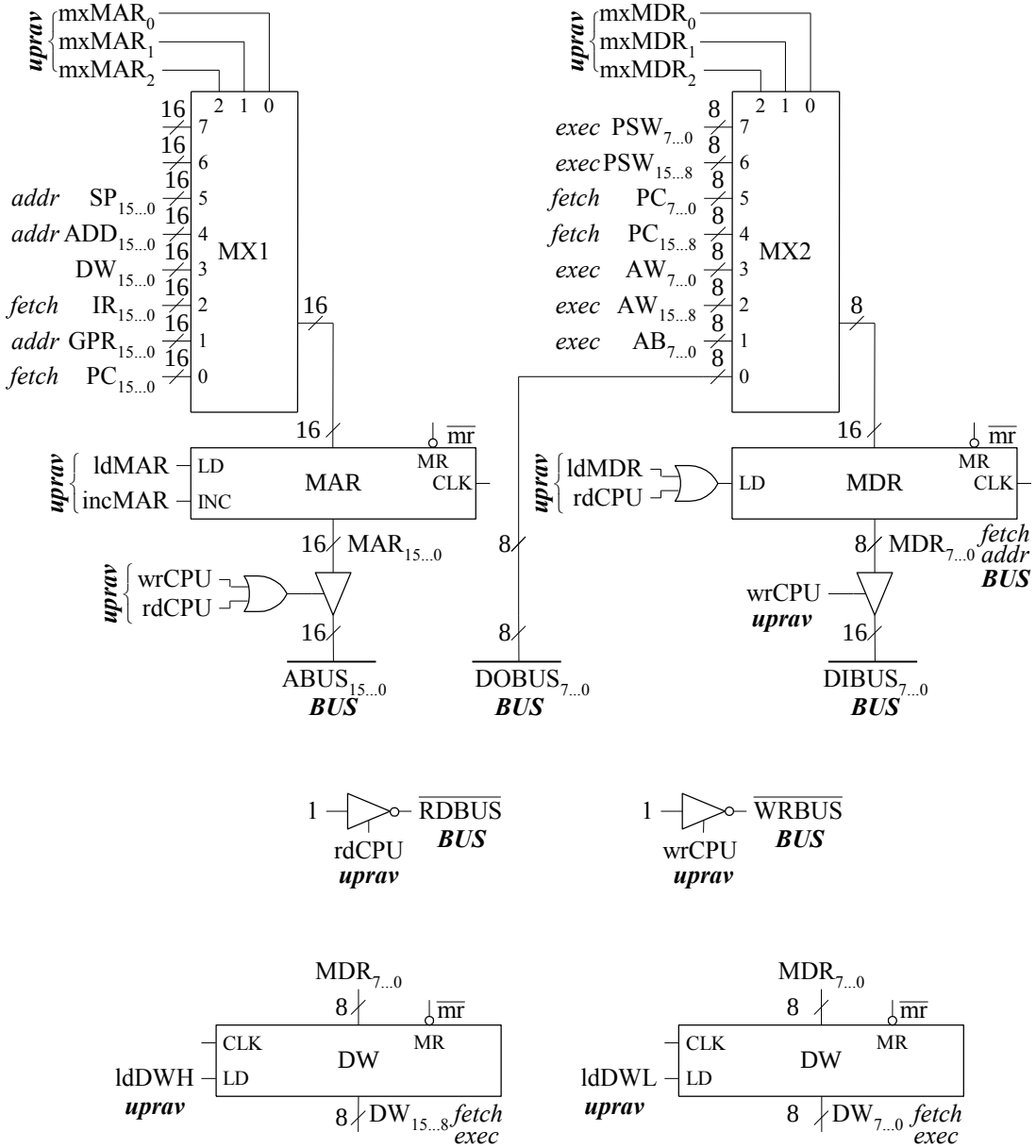
- blok *addr*,
- blok *exec* i
- blok *intr*.

Ovi blokovi su međusobno povezani direktnim vezama.

Blok *bus* (slika 11) služi za realizaciju ciklusa na magistrali ***BUS***. Blok *fetch* (slike 12, 13, 14) služi za čitanje instrukcije i smeštanje u prihvatni registar instrukcije. Blok *addr* (slika 15) služi za formiranje adrese operanda i čitanje operanda. Blok *exec* (slike 16, 17, 18 i 19) služi za izvršavanje operacija. Blok *intr* (slike 20) služi za prihvatanje prekida i generisanje broja ulaza u tabelu sa adresama prekidnih rutina Struktura i opis blokova operacione jedinice ***oper*** se daju u daljem tekstu.

3.1.1 Blok bus

Blok *bus* (slika 11) sadrži registre $MAR_{15...0}$ i $MDR_{7...0}$ sa multiplekserima MX1 i MX2, respektivno i prihvatni registar $DW_{15...0}$.



Slika 11 Blok *bus*

Registar $MAR_{15...0}$ je 16-to razredni adresni registar, čiji se sadržaj normalno koristi pri realizaciji ciklusa čitanja ili upisa na magistrali ***BUS***. Adresa se iz nekog od blokova operacione jedinice ***oper*** preko multipleksera MX1 vodi na ulaze registra $MAR_{15...0}$ i u njega upisuje generisanjem vrednosti 1 signala **ldMAR**. Sadržaj registra $MAR_{15...0}$ se inkrementira generisanjem vrednosti 1 signala **incMAR**. Ovo se koristi u situacijama kada treba pročitati ili upisati 16-to bitnu veličinu koja se nalazi u dvema susednim 8-mo bitnim lokacijama. Tada se najpre u registar $MAR_{15...0}$ upisuje adresa niže lokacije, a posle se inkrementiranjem dobija adresa više lokacije. Pri realizaciji ciklusa čitanja ili upisa signal generiše se vrednost 1 signala rdCPU ili wrCPU, čime se sadržaj registra $MAR_{15...0}$ propušta kroz bafere sa tri stanja na adresne linije $ABUS_{15...0}$ magistrale ***BUS***.

Multiplekser MX1 je 16-to razredni multiplekser sa 8 ulaza. Na ulaze 0 do 5 multipleksera se vode sadržaji $PC_{15...0}$, $GPR_{15...0}$, $IR_{15...0}$, $DW_{15...0}$, $ADD_{15...0}$ i $SP_{15...0}$, a selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 5, respektivno, upravljačkih signala $\mathbf{mxMAR_2}$, $\mathbf{mxMAR_1}$ i $\mathbf{mxMAR_0}$. Sadržaj $PC_{15...0}$ predstavlja adresu instrukcije. Sadržaj $GPR_{15...0}$ predstavlja adresu operanda u slučaju registarskog indirektnog adresiranja. Sadržaj $IR_{15...0}$ predstavlja adresu operanda u slučaju memorijskog direktnog adresiranja. Sadržaj $IR_{15...0}$ predstavlja i adresu memorijske lokacije na kojoj se nalazi adresa operanda u slučaju memorijskog indirektnog adresiranja, pa se tada sa ove adrese čita iz memorije adresa operanda i upisuje u registar $DW_{15...0}$, tako da sadržaj $DW_{15...0}$ predstavlja adresu operanda u slučaju memorijskog indirektnog adresiranja. Sadržaj $ADD_{15...0}$ predstavlja adresu operanda u slučaju registarskog indirektnog adresiranja sa pomerajem, bazno indeksnog sa pomerajem adresiranja i PC relativnog sa pomerajem adresiranju i formira se na izlazu sabirača ADD bloka *addr* saglasno pravilima formiranja adrese za data adresiranja. Sadržaj $SP_{15...0}$ se koristi prilikom stavljanja sadržaja na vrh steka i skidanja sadržaja sa vrha steka. Selektovani sadržaja sa izlaza multipleksera MX1 se vodi na paralelne ulaze registra $MAR_{15...0}$.

Registar $MDR_{7...0}$ je 8-mo razredni registar podatka u koji se generisanjem vrednosti 1 jednog od signala **rdCPU** i **ldMDR** upisuje sadržaj sa izlaza multipleksera MX2. Pri realizaciji ciklusa čitanja generiše se vrednosti 1 signala **rdCPU** i binarna vrednost 000 signala $\mathbf{mxMDR_2}$, $\mathbf{mxMDR_1}$ **i** $\mathbf{mxMDR_0}$, čime se sadržaj sa izlaznih linija podataka $DOBUS_{7...0}$ magistrale ***BUS*** propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. U nekom koraku pre koraka u kome se realizuje ciklus upisa generiše se vrednost 1 signala **ldMDR** i jedna od binarnih vrednosti 001 do 111 signala $\mathbf{mxMDR_2}$, $\mathbf{mxMDR_1}$ **i** $\mathbf{mxMDR_0}$ čime se jedan od sadržaja $AB_{7...0}$, $AW_{15...8}$, $AW_{7...0}$, $PC_{15...8}$, $PC_{7...0}$, $PSW_{15...8}$ i $PSW_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. U koraku u kome treba da se realizuje ciklus upisa generiše se vrednost 1 signala wrCPU, čime se sadržaj registra $MDR_{7...0}$ propušta kroz bafere sa tri stanja na ulazne linije podataka $DBUS_{7...0}$ magistrale ***BUS***.

Multiplekser MX2 je 16-to razredni multiplekser sa 8 ulaza. Na ulaze 0 do 7 multipleksera se vode sadržaji $DO_{7...0}$, $AB_{7...0}$, $AW_{15...8}$, $AW_{7...0}$, $PC_{15...8}$, $PC_{7...0}$, $PSW_{15...8}$ i $PSW_{7...0}$, a selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 7, respektivno, upravljačkih signala $\mathbf{mxMBR_2}$, $\mathbf{mxMBR_1}$ i $\mathbf{mxMBR_0}$. Sadržaj $DO_{7...0}$ se propušta kod ciklusa čitanja i predstavlja pročitanu vrednost memorijske lokacije koja po linijama podataka magistrale ***BUS*** dolazi u procesor ***CPU***. Sadržaji $AB_{7...0}$, $AW_{15...8}$, $AW_{7...0}$, $PC_{15...8}$, $PC_{7...0}$, $PSW_{15...8}$ i $PSW_{7...0}$ se propuštaju u slučaju ciklusa upisa u memorijsku lokaciju. Sadržaj $AB_{7...0}$ se propušta u slučajevima u kojima se sadržaj akumulatora $AB_{7...0}$ upisuje, sadržaji $AW_{15...8}$ i $AW_{7...0}$ u slučajevima u kojima se u dva ciklusa na magistrali sadržaji višeg i nižeg bajta akumulatora $AW_{15...0}$ upisuju, sadržaji $PC_{15...8}$ i $PC_{7...0}$ u slučajevima u kojima se u dva ciklusa na magistrali sadržaji višeg i nižeg bajta programskog brojača $PC_{15...0}$ stavljaju na stek i sadržaji $PSW_{15...8}$ i $PSW_{7...0}$ u slučajevima u kojima se u dva ciklusa na magistrali sadržaji

višeg i nižeg bajta programske statusne reči $PSW_{15...0}$ stavljaju na stek. Selektovani sadržaja sa izlaza multipleksera MX2 se vodi na paralelne ulaze registra $MDR_{7...0}$.

Registar $DW_{15...0}$ je 16-to razredni pomoćni registar koji se koristi za prihvatanje 16-to bitne veličine koja se dobija iz dve susedne 8-mo bitne memorijske lokacije u dva posebna ciklusa na magistrali. Vrednošću 1 signala **ldDWH** se u 8 starijih razreda registra $DW_{15...8}$ upisuje sadržaj registra $MDR_{7...0}$ u koji je prethodno upisan sadržaj sa linija podataka $DBUS_{7...0}$ magistrale, dok se vrednošću 1 signala **ldDWL** u 8 mlađih razreda registra $DW_{7...0}$ upisuje sadržaj registra $MDR_{7...0}$ u koji je prethodno upisan sadržaj sa linija podataka $DBUS_{7...0}$ magistrale. Sadržaj registra $DW_{15...0}$ se potom kao 16-to bitna veličina upisuje u odgovarajući 16-to razredni registar. Registar $DW_{15...0}$ se koristi da se prihvati 16-to bitna adresa operanda u slučaju indirektnog memorijskog adresiranja i da se posle upiše u registar $MAR_{15...0}$. Pored toga registar $DW_{15...0}$ se koristi da se prihvati 16-to bitni operand u slučaju instrukcija koje se izvršavaju nad 16-to bitnim veličinama i da se posle upiše u registar $BW_{15...0}$. Registar $DW_{15...0}$ se koristi i da se prihvati 16-to bitna vrednost sa steka ili 16-to bitna vrednost adrese prekidne rutine iz tabele sa adresama prekidnih rutina i da se posle upiše u programski brojač $PC_{15...0}$.

Pri realizaciji ciklusa čitanja na magistrali procesor otvara bafere sa tri stanja za adresne linije $ABUS_{15..0}$ i upravljačku liniju $\overline{\textbf{RDBUS}}$ magistrale i na njih propušta adresu i vrednost 0 signala čitanja, respektivno, čime se u memoriji startuje čitanje adresirane lokacije. Memorija otvara bafere sa tri stanja za linije podataka $DOBUS_{7..0}$ i na njih propušta sadržaj adresirane lokacije. Procesor prihvata sadržaj sa linija podataka i zatvara bafere sa tri stanja za adresne linije $ABUS_{15..0}$ i upravljačku liniju $\overline{\textbf{RDBUS}}$ magistrale, dok memorija zatvara bafere sa tri stanja za linije podataka $DBUS_{7..0}$.

Pri realizaciji ciklusa upisa na magistrali procesor otvara bafere sa tri stanja za adresne linije $ABUS_{15..0}$, linije podataka $DBUS_{7..0}$ i upravljačku liniju $\overline{\textbf{WRBUS}}$ magistrale i na njih propušta adresu, podatak i vrednost 0 signala upisa, respektivno, čime se u memoriji startuje upis u adresiranu lokaciju. Po završenom upisu procesor zatvara bafere sa tri stanja za adresne linije $ABUS_{15..0}$, linije podataka $DBUS_{7..0}$ i upravljačku liniju $\overline{\textbf{WRBUS}}$ magistrale.

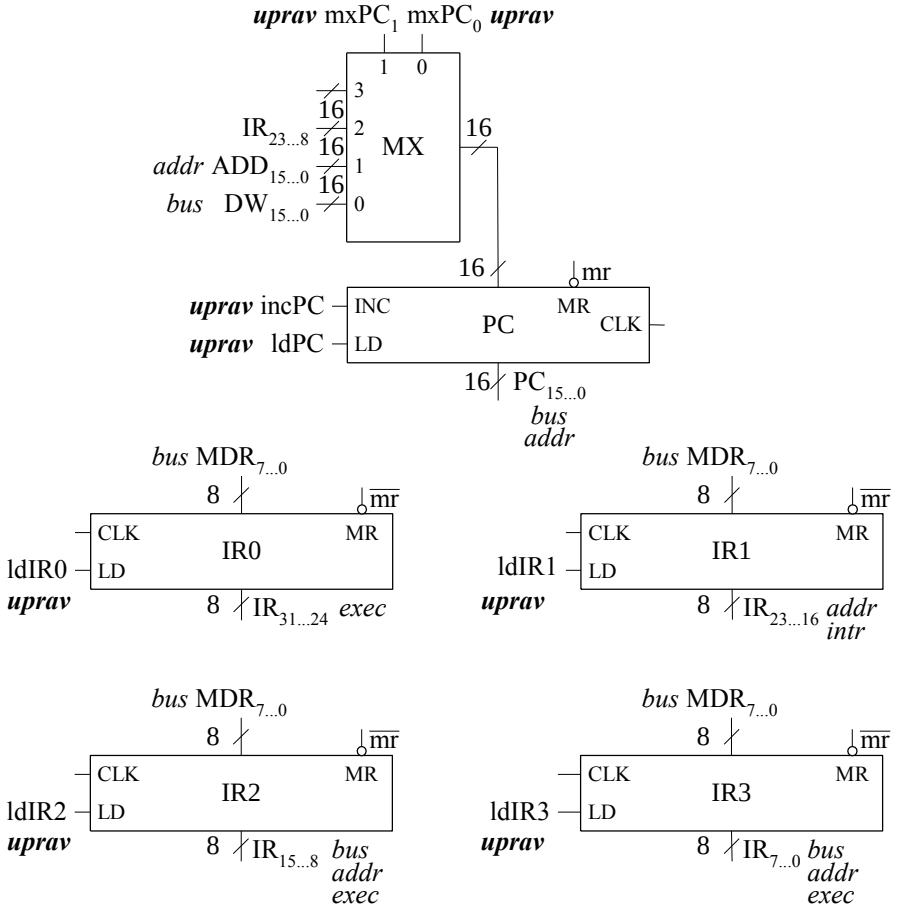
Kombinacione mreže za generisanje upravljačkih signala $\overline{\textbf{RDBUS}}$ i $\overline{\textbf{WRBUS}}$ magistrale generišu ove signale na osnovu signala **rdCPU** i **wrCPU**, respektivno. Pri vrednosti 0 signala **rdCPU** na liniji signala $\overline{\textbf{RDBUS}}$ je stanje visoke impedance, dok je pri vrednosti 1 signala **rdCPU** na liniji signala $\overline{\textbf{RDBUS}}$ vrednost 0. Signal **rdCPU** ima vrednost 1 ili 0 u zavisnosti od toga da li je u memoriji operacija čitanja u toku ili je završena, respektivno. Pri vrednosti 0 signala **wrCPU** na liniji signala $\overline{\textbf{WRBUS}}$ je stanje visoke impedance, dok je pri vrednosti 1 signala **wrCPU** na liniji signala $\overline{\textbf{WRBUS}}$ vrednost 0. Signal **wrCPU** ima vrednost 1 ili 0 u zavisnosti od toga da li je u memoriji operacija upisa u toku ili je završena, respektivno.

3.1.2 Blok fetch

Blok *fetch* sadrži registar $PC_{15...0}$ sa multiplekserom MX, registre IR0, IR1, IR2 i IR3 (slika 12), dekodere DC1 do DC11 signala logičkih uslova operacija i načina adresiranja (slika 13) i kombinacione mreže signala logičkih uslova dužina instrukcija (slika 14).

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. Adresa skoka u programu se iz nekog od blokova operacione jedinice ***oper*** preko multipleksera MX vodi na ulaze registra $PC_{15...0}$ i u njega upisuje generisanjem vrednosti 1 signala **ldPC**. Sadržaj registra $PC_{15...0}$ se inkrementira generisanjem vrednosti 1 signala **incPC**. Ovo se

koristi prilikom čitanja svakog bajta instrukcije koji se nalaze u susednim 8-mo bitnim lokacijama. Sadržaj registra $PC_{15...0}$ se koristi u bloku *addr* i za formiranje adrese memorijske lokacije kada se za adresiranje operanda koristi PC relativno adresiranje.



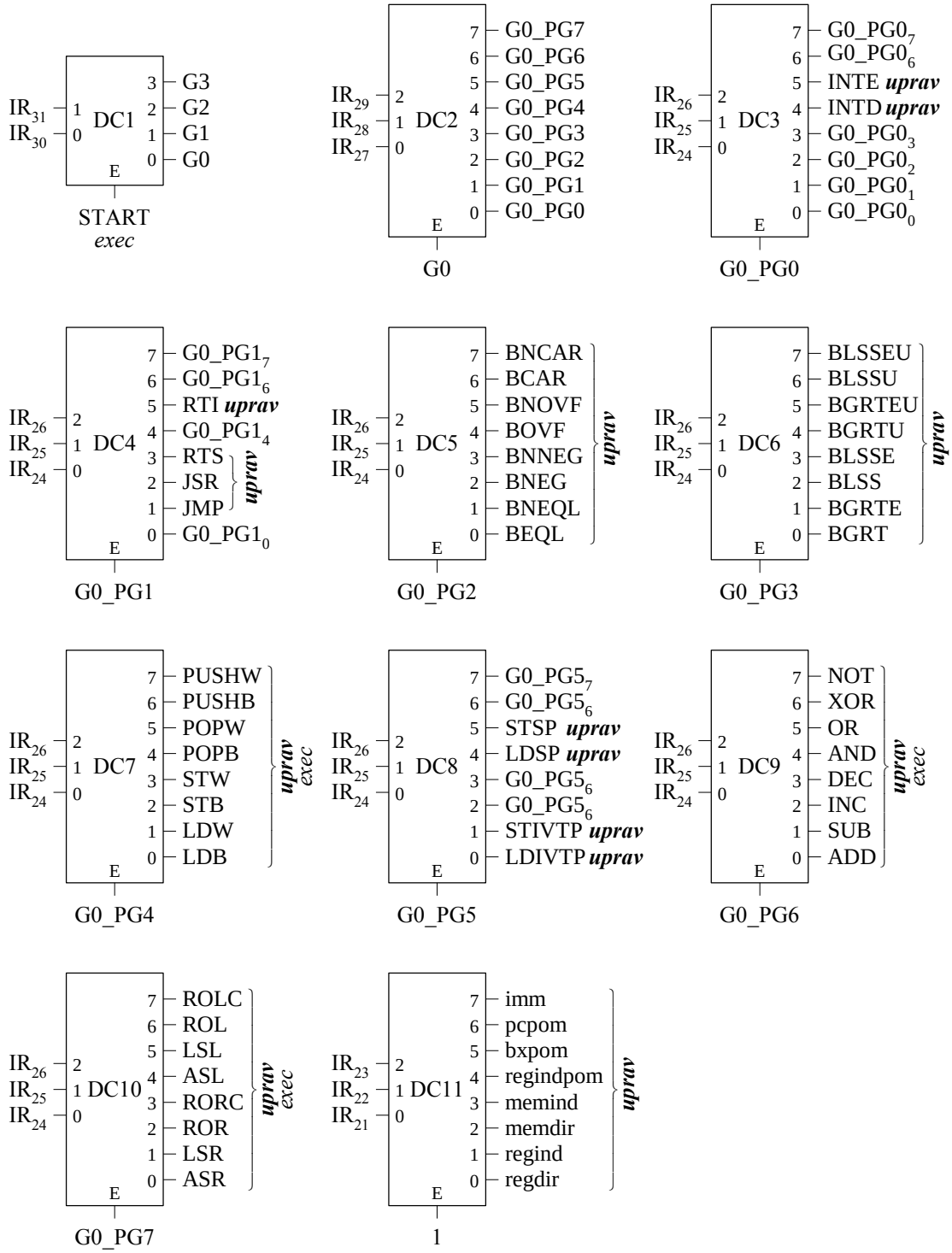
Slika 12 Blok fetch (prvi deo)

Multiplekser MX je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 2 multipleksera se vode sadržaji $DW_{15...0}$, $ADD_{15...0}$ i $IR_{23...8}$, koji predstavljaju adrese skoka za upis u registar $PC_{15...0}$. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 2, respektivno, upravljačkih signala **$mxPC_1$** i **$mxPC_0$**. Sadržaj $DW_{15...0}$ predstavlja ili vrednost koja se restaurira sa steka prilikom izvršavanja instrukcija povratka iz potprograma ili povratka iz prekidne rutine ili adresu prekidne rutine koja se čita iz tabele sa adresama prekidnih prilikom opsluživanja prekida. Sadržaj $ADD_{15...0}$ predstavlja adresu skoka koja se dobija sabiranjem sadržaja registra PC i pomeraja prilikom izvršavanja instrukcija uslovnog skoka i $IR_{23...8}$ predstavlja adresu skoka koja se uzima iz instrukcije prilikom izvršavanja instrukcija bezuslovnog skoka ili skoka na potprogram.

Registri IR0, IR1, IR2 i IR3 su 8-mo razredni registri koji formiraju razrede 31…24, 23…16, 15…8 i 7…0, respektivno, prihvatnog registra instrukcije $IR_{31...0}$. Instrukcije mogu, u zavisnosti od formata instrukcije, da budu dužine 1, 2, 3 ili 4 bajta (poglavlje 2.1.3). Saglasno formatu instrukcije prvi, drugi, treći i četvrti bajt instrukcije se smeštaju redom u registre IR0, IR1, IR2 i IR3. Paralelan upis sadržaja prihvatnog registra podatka $MDR_{7...0}$ u jedan od registara IR0, IR1, IR2 i IR3 se realizuje pri vrednosti 1 jednog od signala **ldIR0**, **ldIR1**, **ldIR2** i **ldIR3**, respektivno. Razredi $IR_{31...24}$ 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 (poglavlje 2.1.3).

Dekoderi DC1 do DC12 se koriste za dekodovanje istrukcija i formiranje signala logičkih uslova operacija **INTD**, ... , **ROLC** i načina adresiranja **regdir**, ..., **imm** (slika 13) saglasno načinu kodiranja instrukcija (poglavlje 2.1.5.2) i načina adresiranja (poglavlje 2.1.4).



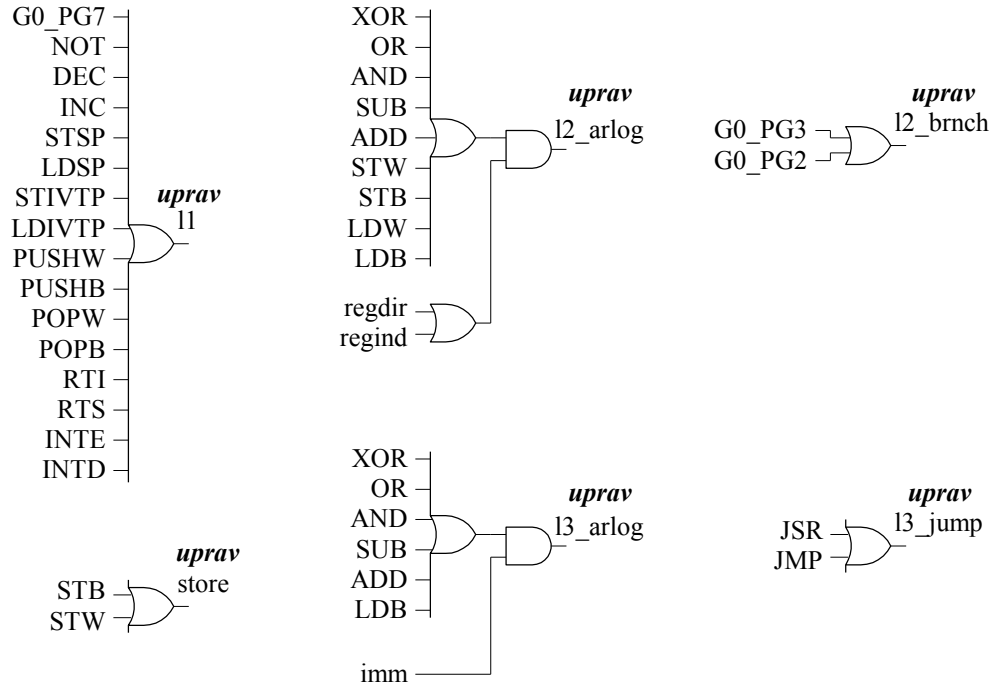
Slika 13 Blok fetch (drugi deo)

Dekoder DC1 služi za formiranje signala četiri grupe operacija G0, G1, G2 i G3 (tabela 3). Dekoder DC2 služi za formiranje signala osam podgrupa operacija G0_PG0 do G0_PG7 grupe G0 (tabela 4). Dekoder DC3 služi za formiranje signala **INTD** i **INTE** podgrupe

operacija G0_PG0 (tabela 5). Dekoder DC4 služi za formiranje signala **JMP**, **JSR**, **RTS** i **RTI** podgrupe operacija G0_PG1 (tabela 6). Dekoder DC5 služi za formiranje signala **BEQL**, **BNEQL**, **BNEG**, **BNNEG**, **BOVF**, **BNOVF**, **BCAR** i **BNCAR** podgrupe operacija G0_PG2 (tabela 7). Dekoder DC6 služi za formiranje signala **BGRT**, **BGRTE**, **BLSS**, **BLSSE**, **BGRTU**, **BGRTEU**, **BLSSU** i **BLSSEU** podgrupe operacija G0_PG3 (tabela 8). Dekoder DC7 služi za formiranje signala **LDB**, **LDW**, **STB**, **STW**, **POPB**, **POPW**, **PUSHB** i **PUSHW** podgrupe operacija G0_PG4 (tabela 9). Dekoder DC8 služi za formiranje signala **LDIVTP**, **STIVTP**, **LDSP** i **STSP** podgrupe operacija G0_PG5 (tabela 10). Dekoder DC9 služi za formiranje signala **ADD**, **SUB**, **INC**, **DEC**, **AND**, **OR**, **XOR** i **NOT** podgrupe operacija G0_PG6 (tabela 11). Dekoder DC10 služi za formiranje signala **ASR**, **LSR**, **ROR**, **RORC**, **ASL**, **LSL**, **ROL** i **ROLC** podgrupe operacija G0_PG7 (tabela 12).

Dekoder DC11 služi za formiranje signala **regdir**, **regind**, **memdir**, **memind**, **regindpom**, **bxpom**, **bcpom**, **imm** načina adresiranja (tabela 1).

Kombinacione mreže signala dužine instrukcija formiraju signale koji ukazuje kolika je dužina instrukcije u bajtovima (slika 14).



Slika 14 Blok fetch (treći deo)

Signal logičkog uslova dužina instrukcije jedan bajt **l1** svojom vrednošću 1 određuje da je dužina instrukcije jedan bajt, a vrednošću 0 da je dužina instrukcije dva, tri ili četiri bajta. Signal **l1** ima vrednost 1 ukoliko se radi o nekoj od instrukcija postavljanja indikatora INTD ili INTE, instrukciji povratka iz potprograma RTS, instrukciji povratka iz prekidne rutine RTI, o nekoj od instrukcija prenosa POPB, POPW, PUSHB, PUSHW, LDIVTP, STIVTP, LDSP ili STSP, nekoj od aritmetičkih instrukcija INC ili DEC, logičkoj instrukciji NOT ili o nekoj od instrukcija pomeranja iz grupe 0 podgrupa 7 (G0_PG7), dok u ostalim situacijama ima vrednost 0.

Signali logičkih uslova dužina instrukcije dva bajta **l2_brnch** i **l2_arlog** svojom vrednošću 1 određuje da je dužina instrukcije dva bajta, a vrednošću 0 da je dužina instrukcije tri ili četiri bajta. Signal **l2_brnch** ima vrednost 1 ukoliko se radi o nekoj od instrukcija uslovnog skoka iz grupe 0 podgrupa 2 ili 3. Signal **l2_arlog** ima vrednost 1 ukoliko se radi o nekoj od

aritmetičkih instrukcija ADD ili SUB, logičkih instrukcija AND, OR ili XOR ili instrukcija prenosa LDB, LDW, STB ili STW za koju je specificirano registarsko direktno (regdir) ili registarsko indirektno (regind) adresiranje.

Signali logičkih uslova dužina instrukcije dva bajta **l2_brnch** i **l2_arlog** svojom vrednošću 1 određuje da je dužina instrukcije dva bajta, a vrednošću 0 da je dužina instrukcije tri ili četiri bajta. Signal **l2_brnch** ima vrednost 1 ukoliko se radi o nekoj od instrukcija uslovnog skoka iz grupe 0 podgrupa 2 ili 3. Signal **l2_arlog** ima vrednost 1 ukoliko se radi o nekoj od aritmetičkih instrukcija ADD ili SUB, logičkih instrukcija AND, OR ili XOR ili instrukcija prenosa LDB, LDW, STB ili STW za koju je specificirano registarsko direktno (regdir) ili registarsko indirektno (regind) adresiranje.

Signali logičkih uslova dužina instrukcije tri bajta **l3_jump** i **l3_arlog** svojom vrednošću 1 određuje da je dužina instrukcije tri bajta, a vrednošću 0 da je dužina instrukcije četiri bajta. Signal **l3_jump** ima vrednost 1 ukoliko se radi o instrukciji bezuslovnog skoka JMP ili instrukciji skoka na potprogram JSR. Signal **l3_arlog** ima vrednost 1 ukoliko se radi o nekoj od aritmetičkih instrukcija ADD ili SUB, nekoj od logičkih instrukcija AND, OR ili XOR ili instrukcija prenosa LDB za koju je specificirano neposredno adresiranje.

Signal logičkog uslova **store** svojom vredošću 1 određuje da se radi o nekoj od instrukcija STB ili STW kojima se upisuje u neki od registara opšte namene ili memorijsku lokaciju.

3.1.3 Blok addr

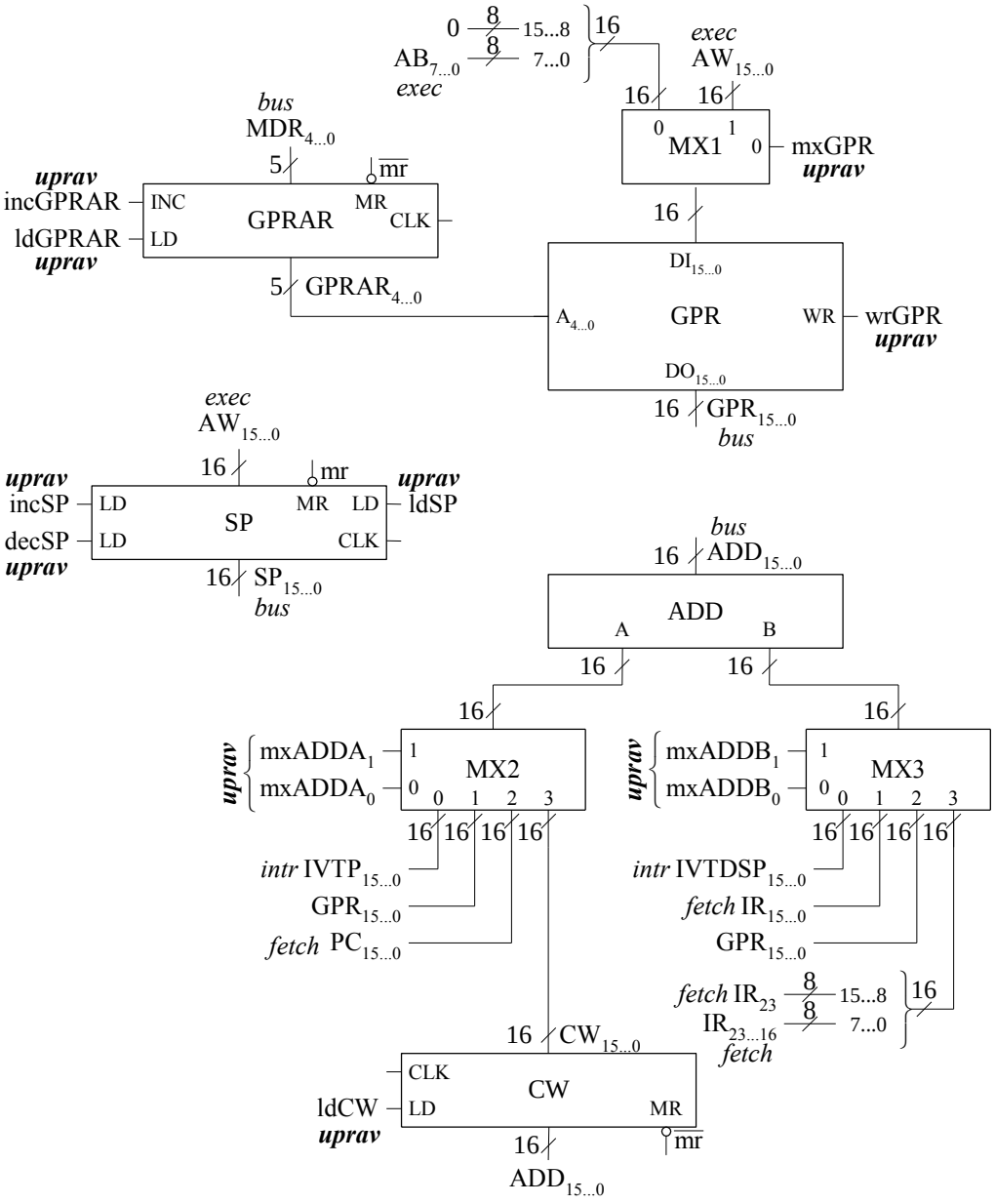
Blok *addr* sadrži registre opšte namene GPR sa adresnim registrom registara opšte namene $GPRAR_{4...0}$ i multiplekserom MX1, sabirač ADD sa multiplekserima MX2 i MX3, pomoćni registar $CW_{15...0}$ i ukazivač na vrh steka $SP_{15...0}$ (slika 15).

Registri opšte namene GPR su realizovani kao registarski fajl sa 32 registra širine 16 bita. Adresa registra opšte namene koji se čita ili u koji se upisuje određena je sadržajem registra $GPRAR_{4...0}$ čiji se sadržaj vodi na adresne linije $A_{4...0}$ registarskog fajla GPR. Sadržaj koji se upisuje vodi se sa izlaza multipleksera MX1 na ulazne linije podataka $DI_{15...0}$ registarskog fajla, a sadržaj koji se čita $GPR_{15...0}$ pojavljuje se na izlaznim linijama podataka $DO_{15...0}$ registarskog fajla. Pri vrednostima 1 i 0 signala **wrGPR** na upravljačkoj liniji WR registarskog fajla realizuje se upis u registarski fajl i čitanje iz registarskog fajla, respektivno.

Adresni registar opšte namene $GPRAR_{4...0}$ je 5-to razredni registar čiji se sadržaj koristi kao adresa registra registarskog fajla prilikom upisa u registarki fajl i čitanja iz registarskog fajla. U registar $GPRAR_{4...0}$ se pri vrednosti 1 signala **ldGPRAR** upisuju razredi 5...0 prihvatnog registra podatka $MDR_{7...0}$ bloka *bus*. Ovo se koristi samo u fazi čitanja instrukcije i to prilikom čitanja drugog bajta instrukcije. Tada ova grupa bitova, ukoliko se radi o aritmetičkim ili logičkim instrukcijama sa adresiranjima koja koriste registre registre opšte namene, predstavlja adresu registra opšte namene. Sadržaj registra $GPRAR_{4...0}$ se inkrementira pri vrednosti 1 signala **incGPRAR**, što se koristi samo ukoliko se radi o bazno indeksnom sa pomerajem adresiranju. Tada se registar opšte namene čija je adresa zadata bitovima 5...0 drugog bajta instrukcije koristi kao bazni registar, a registar opšte namene sa prve sledeće adrese kao indeksni registar. Sadržaj registra $GPRAR_{4...0}$ se vodi na adresne linije $A_{4...0}$ registarskog fajla GPR.

Multiplekser MX1 je 16-to razredni multiplekser sa 2 ulaza. Na ulaze 0 i 1 multipleksera se vode sadržaji registara $AB_{7...0}$ i $AW_{15...0}$, pri čemu je sadržaj registra $AB_{7...0}$ proširen nulama do dužine 16 bita. Selekcija jednog od ova dva sadržaja se realizuje binarnim vrednostima 0 i 1, respektivno, upravljačkog signala **mxGPR**. Ovo se korisi u instrukcijama prenosa STB i

STW sadržaja akumulatora $AB_{7...0}$ i $AW_{15...0}$, respektivno, kada je registarskim direktnim adresiranjem kao odredište specificiran neki od registara opšte namene. Selektovani sadržaj se vodi na ulazne linije podataka $DI_{15...0}$ registarskog fajla GPR.



Slika 15 Blok addr

Sabirač ADD je 16-to razredni sabirač koji se koristi za formiranje16-to bitne vrednosti koja može da bude adresa sa koje treba da se pročita adresa prekidne rutine, adresa sa koje treba da se pročita ili na kojoj treba da se upiše operand ili adresa skoka. Sabiranje se realizuje nad sadržajima koji sa izlaza multipksera MX2 i MX3 dolaze na ulaze $A_{15...0}$ i $B_{15...0}$, respektivno, sabirača ADD. Sadržaj $ADD_{15...0}$ sa izlaza sabirača se vodi u blok *bus* radi upisa u adresni registar $MAR_{15...0}$ i u blok *fetch* radi upisa u programski brojač $PC_{15...0}$. Sadržaj $ADD_{15...0}$ sa izlaza sabirača se u samom bloku *addr* upisuje u registar $CW_{15...0}$. Ovo se koristi kod bazno indeksnog adresiranja. Najpre se sabiraju jedan od registara opšte namene koji se koristi kao bazni registar i pomeraj i njihova suma upisuju u registar $CW_{15...0}$, a posle se sabiraju registar $CW_{15...0}$, i jedan od registara opšte namene koji se koristi kao indeksni registar i njihova suma upisuje u adresni registar $MAR_{15...0}$.

Multiplekser MX2 je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 3 multipleksera se vode sadržaji $IVTP_{15...0}$, $GPR_{15...0}$, $PC_{15...0}$ i $CW_{15...0}$. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 3, respektivno, upravljačkih signala **$mxADDA_1$** i **$mxADDA_0$**. Sadržaj sa izlaza multipleksera MX2 se vodi na ulaze $A_{15...0}$ sabirača ADD.

Multiplekser MX3 je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 3 multipleksera se vode sadržaji $IVTDSP_{15...0}$, $IR_{15...0}$, $GPR_{15...0}$ i $IR_{23...16}$ proširen znakom do dužine 16 bita. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 3, respektivno, upravljačkih signala **$mxADDB_1$** i **$mxADDB_0$**. Sadržaj sa izlaza multipleksera MX3 se vodi na ulaze $B_{15...0}$ sabirača ADD.

Sadržaji $IVTP_{15...0}$ i $IVTDSP_{15...0}$ se propuštaju kroz multipleksere MX2 i MX3, respektivno, da bi se njihovim sabiranjem na izlazu sabirača ADD dobila adresa sa koje treba da se pročita adresa prekidne rutine. Pri tome je $IVTP_{15...0}$ sadržaj registra koji ukazuje na početak tabele sa adresama prekidnih rutina i $IVTDSP_{15...0}$ pomeraj u odnosu na početak tabele formiran na osnovu broja ulaza u tabelu.

Sadržaji $GPR_{15...0}$ i $IR_{15...0}$ se propuštaju kroz multipleksere MX2 i MX3, respektivno, da bi se njihovim sabiranjem u slučaju registarskog indirektnog sa pomerajem adresiranja na izlazu sabirača ADD dobila adresa sa koje treba da se pročita ili na kojoj treba da se upiše operand. Pri tome je $GPR_{15...0}$ sadržaj registra opšte namene čija je adresa zadata instrukcijom i $IR_{15...0}$ pomeraj. Ovi sadržaji se propuštaju kroz multipleksere i sabiraju i u slučaju baznog indeksnog sa pomerajem adresiranja, pri čemu se rezultat sabiranja upisuje u registar $CW_{15...0}$.

Sadržaji $PC_{15...0}$ i $IR_{15...0}$ se propuštaju kroz multipleksere MX2 i MX3, respektivno, da bi se njihovim sabiranjem u slučaju PC relativnog sa pomerajem adresiranja na izlazu sabirača ADD dobila adresa sa koje treba da se pročita ili na kojoj treba da se upiše operand. Pri tome je $PC_{15...0}$ sadržaj programskog brojača i $IR_{15...0}$ pomeraj.

Sadržaji $CW_{15...0}$ i $GPR_{15...0}$ se propuštaju kroz multipleksere MX2 i MX3, respektivno, da bi se njihovim sabiranjem u slučaju baznog indeksnog sa pomerajem adresiranja na izlazu sabirača ADD dobila adresa sa koje treba da se pročita ili na kojoj treba da se upiše operand. Pri tome je $CW_{15...0}$ sadržaj pomoćnog registra opšte namene u kome se nalazi suma registra opšte namene čija je adresa zadata instrukcijom i pomeraja, dok je $GPR_{15...0}$ sadržaj registra opšte namene sa prve sledeće adrese u odnosu na adresu zadatu instrukcijom.

Sadržaji $PC_{15...0}$ i $IR_{23...16}$ proširen znakom do dužine 16 bita se propuštaju kroz multipleksere MX2 i MX3, respektivno, da bi se njihovim sabiranjem na izlazu sabirača ADD u slučaju instrukcija uslovnog skoka dobila adresa skoka. Pri tome je $PC_{15...0}$ sadržaj programskog brojača i $IR_{23...16}$ proširen znakom do dužine 16 bita pomeraj u odnosu na tekuću vrednost programskog brojača.

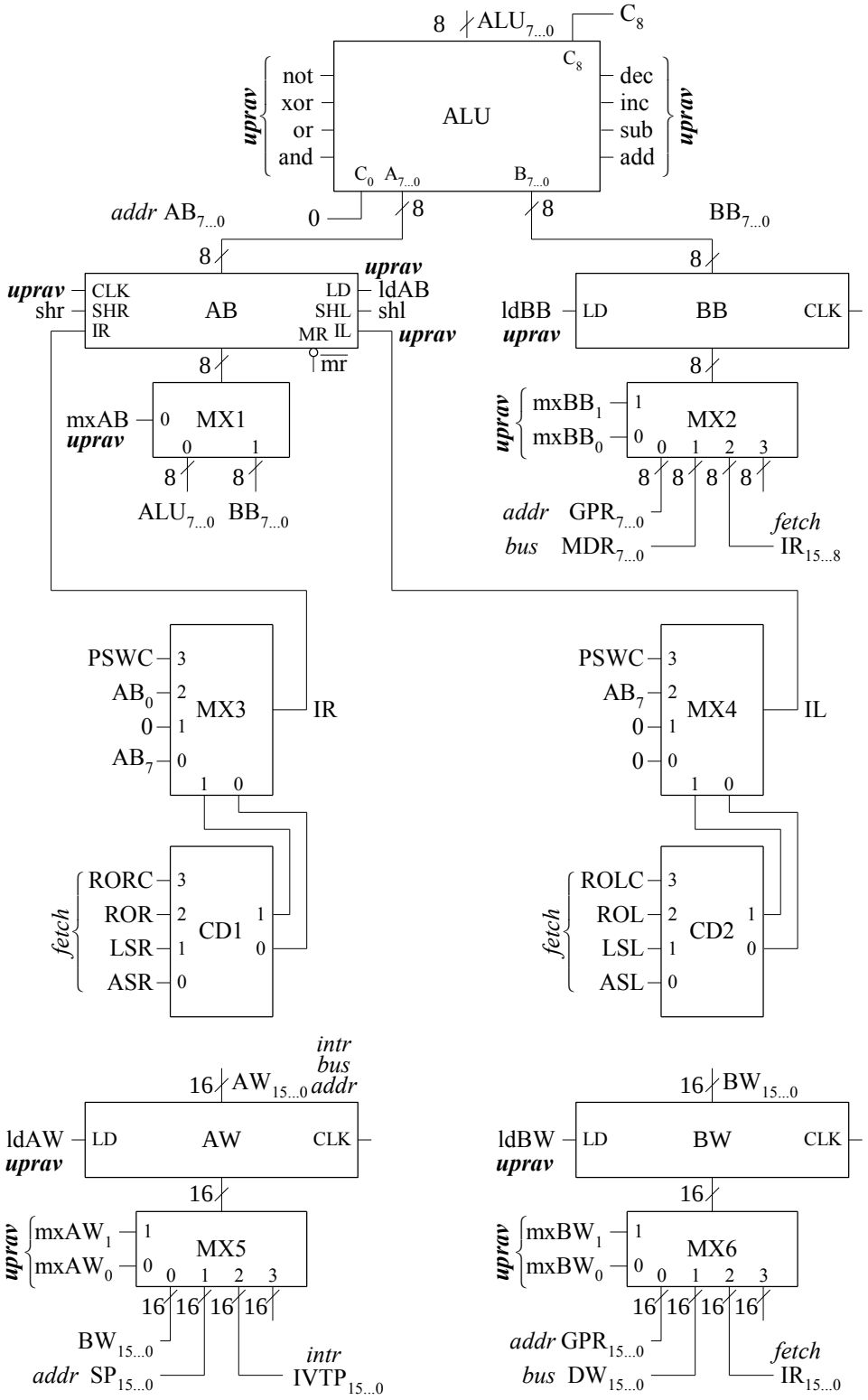
Registar $CW_{15...0}$ je 16-to razredni pomoćni registar u kome se u slučaju baznog indeksnog sa pomerajem adresiranja čuva suma registra opšte namene i pomeraja. Ova suma se dobija sa izlaza sabirača ADD i u registar $CW_{15...0}$ upisuje pri vrednosti 1 signala **ldCW**. Sadržaj registra $CW_{15...0}$ se vodi na ulaze multipleksera MX2.

Registar $SP_{15...0}$ je 16-to razredni ukazivač na vrh steka čiji sadržaj predstavlja adresu memorijske lokacije prilikom upisa na stek i čitanja sa steka. U registar $SP_{15...0}$ se pri vrednosti 1 signala **ldSP** u fazi izvršavanja instrukcije prenosa upisuje sadržaj 16-to razrednog akumulatora $AW_{15...0}$. Sadržaj registra $SP_{15...0}$ se inkrementira i dekrementira pri vrednostima 1 signala **incSP** i **decSP**, respektivno. Ovo se koristi pri upisu na stek i čitanju sa steka,

respektivno. Sadržaj registra $SP_{15...0}$ se vodi na ulaze multiplekser MX1 radi upisa u adresni registar memorije $MAR_{15...0}$ bloka *bus*.

3.1.4 Blok exec

Blok *exec* sadrži aritmetičko logičku jedinicu ALU, registre $AB_{7...0}$ i $BB_{7...0}$, multipleksere MX1, MX2, MX3 i MX4, kodere DC1 i DC2, registre $AW_{15...0}$ i $BW_{15...0}$, multipleksere MX5 i MX6 (slika 16), registar $PSW_{15...0}$, flip-flop START (slika 17), kombinacione mreže za formiranje signala postavljanja indikatora N, Z, C i V (slika 18) i kombinacione mreže za formiranje signala rezultata operacija **eql**, ..., **nneg**, **brpom** (slika 19).



Slika 16 Blok exec (prvi deo)

Aritmetičko logička jedinica ALU realizacije četiri aritmetičke i četiri logičke mikrooperacije nad sadržajima registara $AB_{7...0}$ i $BB_{7...0}$ (slike 16). Mikrooperacija koju treba realizovati se specificira vrednošću 1 ili jednog od upravljačkih signala **add**, **sub**, **inc** i **dec** za jednu od aritmetičkih mikrooperacija sabiranja, oduzimanja, inkementiranja i dekrementiranja, respektivno, ili jednog od upravljačkih signala **and**, **or**, **xor** i **not** za jednu od logičkih mikrooperacija I, ILI, ekskluzivno ILI i komplementiranja, respektivno. Rezultat realizovane mikrooperacije se dobija na linijama $ALU_{7...0}$. Na ulaz C_0 je dovedena vrednost 0, pri čemu je vrednost signala C_0, bitna samo u slučaju aritmetičkih mikrooperacija, dok ta vrednost nije bitna u slučaju logičkih mikrooperacija. U slučaju aritmetičkih mikrooperacija sabiranja i inkrementiranja vrednost 1 na izlazu C_8 predstavlja prenos, a u slučaju aritmetičkih mikrooperacija oduzimanja i dekrementiranja vrednost 0 na izlazu C_8 predstavlja pozajmicu. U slučaju logičkih mikrooperacija signal na izlazu C_8 nema smisla.

Registar $AB_{7...0}$ je 8-mo razredni akumulator koji se koristi kao implicitno izvorište i odredište u aritmetičkim, logičkim i pomeračkim instrukcijama. Sadržaj sa izlaza multipleksera MX1 se vodi na ulaze registra $AB_{7...0}$ i u njega upisuje generisanjem vrednosti 1 signala **ldAB**. Sadržaj registra $AB_{7...0}$ se pomera udesno za jedno mesto generisanjem vrednosti 1 signala **shR**. Tada se u najstariiji razred registra AB_7 upisuje signal **IR** koji dolazi sa izlaza multipleksera MX3. Sadržaj registra $AB_{7...0}$ se pomera ulevo za jedno mesto generisanjem vrednosti 1 signala **shL**. Tada se u najmlađi razred registra AB_0 upisuje signal **IL** koji dolazi sa izlaza multipleksera MX4. Sadržaj registra $AB_{7...0}$ se vodi na ulaze ALU gde se koristi kao prvi izvorišni operand u slučaju aritmetičkih i logičkih operacija.

Multiplekser MX1 je 8-mo razredni multiplekser sa 2 ulaza. Na ulaze 0 i 1 multipleksera se vode sadržaji $ALU_{7...0}$ i $BB_{7...0}$ koji se propuštaju kroz multiplekser i upisuju u registar $AB_{7...0}$. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 i 1, respektivno, upravljačkog signala **mxAB**. Sadržaj $ALU_{7...0}$ predstavlja rezultat aritmetičke ili logičke operacije koje kao odredište implicitno koriste registar akumulatora $AB_{7...0}$. Sadržaj $BB_{7...0}$ predstavlja operand koji se u fazi izvršavanja instrukcije LDB prebacuje iz prihvatnog registra operanda $BB_{7...0}$ u registar akumulatora $AB_{7...0}$.

Registar $BB_{7...0}$ je 8-mo razredni prihvatni registar u koji se privremeno smešta izvorišni operand specificiran adresnim delom svih instrukcija sa jednoadresnim formatom. Sadržaj sa izlaza multipleksera MX2 se vodi na ulaze registra $BB_{7...0}$ i u njega upisuje generisanjem vrednosti 1 signala **ldBB**. Sadržaj registra $BB_{7...0}$ se vodi na ulaze ALU, gde se koristi kao drugi izvorišni operand u slučaju aritmetičkih i logičkih instrukcija, i na ulaze multipleksera MX1 kroz koji se propušta i upisuje u registar $AB_{7...0}$ u slučaju instrukcije LDB.

Multiplekser MX2 je 8-mo razredni multiplekser sa 4 ulaza. Na ulaze 0 do 2 multipleksera se vode sadržaji $GPR_{7...0}$, $MDR_{7...0}$ i $IR_{15...8}$ koji se propuštaju kroz multiplekser i upisuju u registar $BB_{7...0}$. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 2, respektivno, upravljačkih signala $\mathbf{mxBB_1}$ i $\mathbf{mxBB_0}$. Sadržaj $GPR_{7...0}$ predstavlja operand u slučaju registarskog direktnog adresiranja, $MDR_{7...0}$ u slučaju memorijskih adresiranja i $IR_{15...8}$ u slučaju neposrednog adresiranja.

Multiplekser MX3 je 1-no razredni multiplekser sa 4 ulaza. Na ulaze 0 do 3 multipleksera se dovode signali $\mathbf{AB_7}$, **0**, $\mathbf{AB_0}$ i **PSWC** koji se propuštaju kroz multiplekser i po liniji IR upisuju u razred AB_7 registra $AB_{7...0}$ pri realizaciji neke od četiri instrukcije pomeranja za jedno mesto udesno sadržaja registra $AB_{7...0}$. Selekcija jednog od ova četiri signala se realizuje binarnim vrednostima 0 do 3, respektivno, sadržaja sa izlaza kodera CD1. Signal $\mathbf{AB_7}$ se propušta za instrukciju aritmetičkog pomereranja udesno kada signal operacije ASR ima vrednost 1. Signal **0** se propušta za instrukciju logičkog pomereranja udesno kada signal

operacije LSR ima vrednost 1. Signal $\mathbf{AB_0}$ se propušta za instrukciju rotiranja udesno kada signal operacije ROR ima vrednost 1. Signal **PSWC** se propušta za instrukciju rotiranja kroz indikator PSWC udesno kada signal operacije RORC ima vrednost 1.

Multiplekser MX4 je 1-no razredni multiplekser sa 4 ulaza. Na ulaze 0 do 3 multipleksera se dovode signali **0**, **0**, $\mathbf{AB_7}$ i **PSWC** koji se propuštaju kroz multiplekser i po liniji IL upisuju u razred AB_0 registra $AB_{7...0}$ pri realizaciji neke od četiri instrukcije pomeranja za jedno mesto ulevo sadržaja registra $AB_{7...0}$. Selekcija jednog od ova četiri signala se realizuje binarnim vrednostima 0 do 3, respektivno, sadržaja sa izlaza kodera CD2. Signal **0** se propušta za instrukciju aritmetičkog pomereranja ulevo kada signal operacije ASL ima vrednost 1. Signal **0** se propušta za instrukciju logičkog pomereranja ulevo kada signal operacije LSL ima vrednost 1. Signal $\mathbf{AB_7}$ se propušta za instrukciju rotiranja ulevo kada signal operacije ROL ima vrednost 1. Signal **PSWC** se propušta za instrukciju rotiranja kroz indikator PSWC ulevo kada signal operacije ROLC ima vrednost 1.

Koder CD1 je koder sa četiri ulaza i dva izlaza koji na izlazima daje binarnu vrednost 0, 1, 2 ili 3 u zavisnosti od toga koji od signala operacija pomeranja i rotiranja udesno za jedno mesto ASR, LSR, ROR i RORC ima vrednost 1. Ukoliko signal operacije aritmetičkog pomeranja udesno ASR koji je povezan na ulaz 0 kodera ima vrednost 1, na izlazima kodera se pojavljuje binarna vrednost 0, koja omogućuje da se kroz multiplekser MX1 na liniju IR registra $AB_{7...0}$ propusti signal $\mathbf{AB_7}$. Ukoliko signal operacije logičkog pomeranja udesno LSR koji je povezan na ulaz 1 kodera ima vrednost 1, na izlazima kodera se pojavljuje binarna vrednost 1, koja omogućuje da se kroz multiplekser MX1 na liniju IR registra $AB_{7...0}$ propusti signal **0**. Ukoliko signal operacije rotiranja udesno ROR koji je povezan na ulaz 2 kodera ima vrednost 1, na izlazima kodera se pojavljuje binarna vrednost 2, koja omogućuje da se kroz multiplekser MX1 na liniju IR registra $AB_{7...0}$ propusti signal $\mathbf{AB_0}$. Ukoliko signal operacije rotiranja udesno kroz indikator PSWC RORC koji je povezan na ulaz 3 kodera ima vrednost 1, na izlazima kodera se pojavljuje binarna vrednost 3, koja omogućuje da se kroz multiplekser MX1 na liniju IR registra $AB_{7...0}$ propusti signal **PSWC**.

Koder CD2 je koder sa četiri ulaza i dva izlaza koji na izlazima daje binarnu vrednost 0, 1, 2 ili 3 u zavisnosti od toga koji od signala operacija pomeranja i rotiranja ulevo za jedno mesto ASL, LSL, ROL i ROLC ima vrednost 1. Ukoliko signal operacije aritmetičkog pomeranja ulevo ASL koji je povezan na ulaz 0 kodera ima vrednost 1, na izlazima kodera se pojavljuje binarna vrednost 0, koja omogućuje da se kroz multiplekser MX2 na liniju IL registra $AB_{7...0}$ propusti signal **0**. Ukoliko signal operacije logičkog pomeranja ulevo LSL koji je povezan na ulaz 1 kodera ima vrednost 1, na izlazima kodera se pojavljuje binarna vrednost 1, koja omogućuje da se kroz multiplekser MX2 na liniju IL registra $AB_{7...0}$ propusti signal **0**. Ukoliko signal operacije rotiranja ulevo ROL koji je povezan na ulaz 2 kodera ima vrednost 1, na izlazima kodera se pojavljuje binarna vrednost 2, koja omogućuje da se kroz multiplekser MX2 na liniju IL registra $AB_{7...0}$ propusti signal $\mathbf{AB_7}$. Ukoliko signal operacije rotiranja ulevo kroz indikator PSWC ROLC koji je povezan na ulaz 3 kodera ima vrednost 1, na izlazima kodera se pojavljuje binarna vrednost 3, koja omogućuje da se kroz multiplekser MX2 na liniju IL registra $AB_{7...0}$ propusti signal **PSWC**.

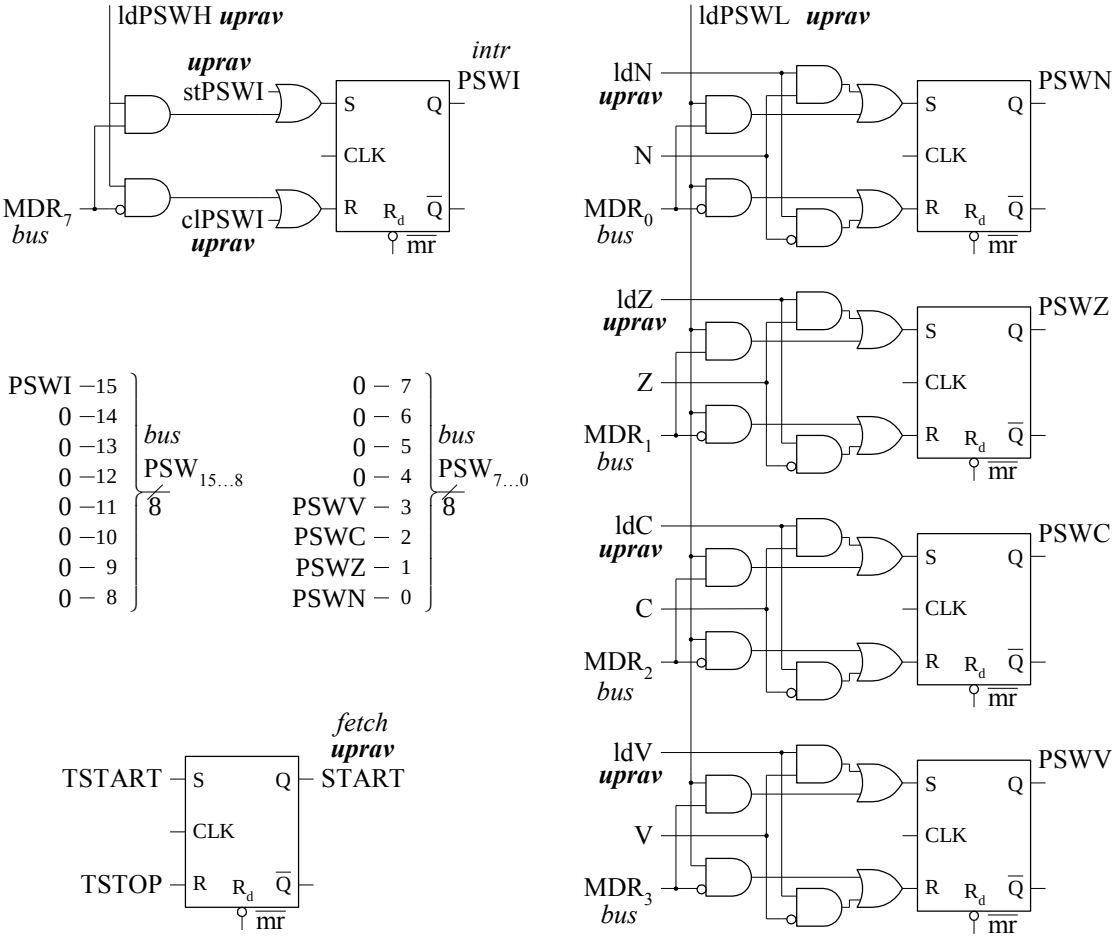
Registar $AW_{15...0}$ je 16-to razredni akumulator koji se koristi kao implicitno izvorište u instrukcijama prenosa STW, PUSHW, STIVTP i STSP i implicitno odredište u instrukcijama prenosa LDW, POPW, LDIVTP i LDSP. Sadržaj sa izlaza multipleksera MX5 se vodi na ulaze registra $AW_{15...0}$ i u njega upisuje generisanjem vrednosti 1 signala **ldAW**. Sadržaj registra $AW_{15...0}$ se vodi u blok *addr* gde se, ukoliko se izvršava instrukcija prenosa STW sa zadatim direktnim registarskim adresiranjem, upisuje u registarski fajl GPR i ukoliko se izvršava

instrukcija STSP upisuje u registar SP, u blok *bus* gde se, ukoliko se izvršava instrukcija prenosa STW sa zadatim nekim od memorijskih adresiranja ili instrukcija prenosa PUSHW, upisuje u memoriju, i u blok *intr* gde se, ukoliko se izvršava instrukcija STIVTP, upisuje u registar IVTP.

Multiplekser MX5 je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 2 multipleksera se vode sadržaji $BW_{15...0}$, $SP_{15...0}$ i $IVTP_{15...0}$. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 2, respektivno, upravljačkih signala **mxAW$_1$** i **mxAW$_0$**. Sadržaj sa izlaza multipleksera MX5 se vodi na ulaze registra $AW_{15...0}$. Sadržaj $BW_{15...0}$ se propušta kroz multiplekser u slučaju instrukcije LDW, sadržaj $SP_{15...0}$ u slučaju instrukcije LDSP i sadržaj $IVTP_{15...0}$ u slučaju instrukcije LDIVTP.

Registar $BW_{15...0}$ je 16-to razredni prihvatni registar u koji se privremeno smešta izvorišni operand specificiran adresnim delom instrukcije prenosa LDW. Sadržaj sa izlaza multipleksera MX6 se vodi na ulaze registra $BW_{15...0}$ i u njega upisuje vrednošću 1 signala **ldBW**. Sadržaj registra $BW_{15...0}$ se vodi na ulaze multipleksera MX5 kroz koji se propušta i upisuje u registar $AW_{15...0}$ u slučaju instrukcije LDW.

Multiplekser MX6 je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 2 multipleksera se vode sadržaji $GPR_{15...0}$, $DW_{15...0}$ i $IR_{15...0}$. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 2, respektivno, upravljačkih signala **mxBW$_1$** i **mxBW$_0$**. Sadržaj sa izlaza multipleksera MX6 se vodi na ulaze registra $BW_{15...0}$. Sadržaj $GPR_{15...0}$ predstavlja operand u slučaju direktnog registarskog adresiranja, sadržaj $DW_{15...0}$ u slučaju memorijskih adresiranja i sadržaj $IR_{15...0}$ u slučaju neposrednog adresiranja.



Slika 17 Blok exec (drugi deo)

Registar $PSW_{15...0}$ je 16-to razredni registar koji sadrži indikatore programske statusne reči procesora (slika 17). Registar $PSW_{15...0}$ se sastoji od flip-flopova PSWI, PSWV, PSWC, PSWZ i PSWN, koji predstavljaju razrede PSW_{15} i $PSW_{3...0}$, respektivno, dok razredi $PSW_{14...4}$ ne postoje.

Flip-flop PSWI sadrži indikator *interrupt*. Flip-flop se postavlja na vrednost 1 vrednošću 1 signala **stPSWI** u okviru faze izvršavanja instrukcije INTE i na vrednost 0 vrednošću 1 signala **clPSWI** u okviru faze izvršavanja instrukcije INTD.

Flip-flop PSWV sadrži indikator *overflow*. U flip-flop PSWV se vrednošću 1 signala **ldV** u okviru faze izvršavanja određenih instrukcija (odeljak 2.1.5) upisuje vrednost signala **V**.

Flip-flop PSWC sadrži indikator *carry/borrow*. U flip-flop PSWC se vrednošću 1 signala **ldC** u okviru faze izvršavanja određenih instrukcija (odeljak 2.1.5) upisuje vrednost signala **C**.

Flip-flop PSWZ sadrži indikator *zero*. U flip-flop PSWZ se vrednošću 1 signala **ldZ** u okviru faze izvršavanja određenih instrukcija (odeljak 2.1.5) upisuje vrednost signala **Z**.

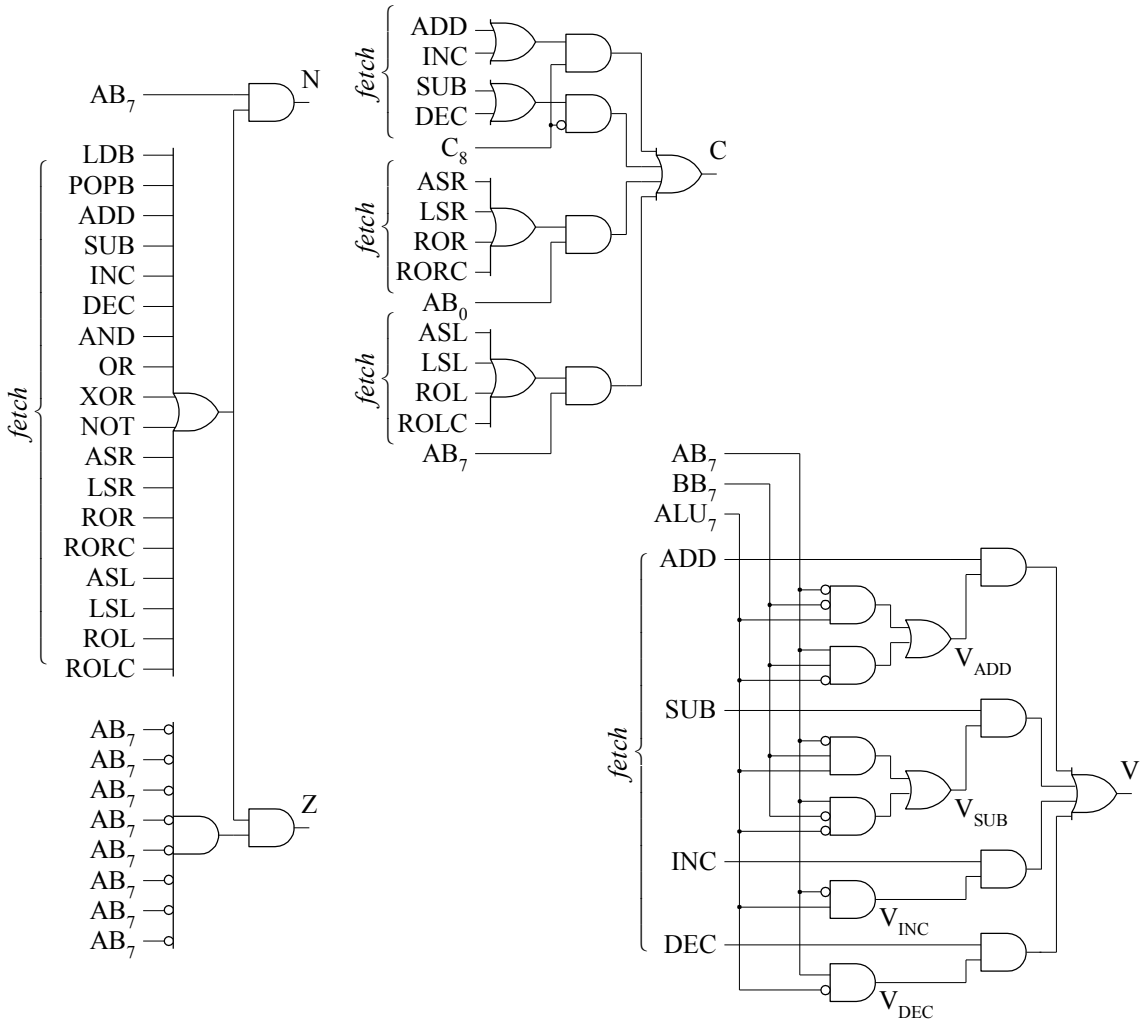
Flip-flop PSWN sadrži indikator *negative*. U flip-flop PSWN se vrednošću 1 signala **ldN** u okviru faze izvršavanja određenih instrukcija (odeljak 2.1.5) upisuje vrednost signala **N**.

U razred PSW_{15} koji predstavlja najstariji razred registra $PSW_{15...0}$ se vrednošću 1 signala **ldPSWH** upisuje sadržaj razreda MDR_7 prihvatnog registra podatka $MDR_{7...0}$ bloka *bus*, dok se u razrede $PSW_{3...0}$ koji predstavljaju 4 najmlađa razreda registra $PSW_{15...0}$ vrednošću 1 signala **ldPSWL** upisuje sadržaj razreda $MDR_{3...0}$ prihvatnog registra podatka $MDR_{7...0}$. Ovo se koristi prilikom izvršavanja instrukcije RTI da se sadržajem sa vrha steka restaurira sadržaj registra $PSW_{15...0}$. Sadržaj najstarijeg razreda PSW_{15}, 4 najmlađa razreda $PSW_{3...0}$ i vrednosti 0 za nepostojeće razrede $PSW_{14...4}$ registra $PSW_{15...0}$ se vode posebno kao 8 starijih razreda $PSW_{15...8}$ i 8 mlađih razreda $PSW_{7...0}$ registra $PSW_{15...0}$ u blok *bus* u kome se upisuju u prihvatni registar podatka $MDR_{7...0}$. Ovo se koristi prilikom opsluživanja zahteva za prekida da se sadržaj registra $PSW_{15...0}$ stavi na vrh steka.

Flip-flop START svojom vrednošću 1 omogućava izvršavanje instrukcija, dok vrednošću 0 zaustavlja izvršavanje instrukcija. U flip-flop START se upisuje vrednost 1 pri vrednosti 1 signala **TSTART** koja se generiše ili pri uključenju napajanja ili aktiviranjem tastera START. U flip-flop se upisuje vrednost 0 vrednosti 1 signala **TSTOP** koja se generiše aktiviranjem tastera START.

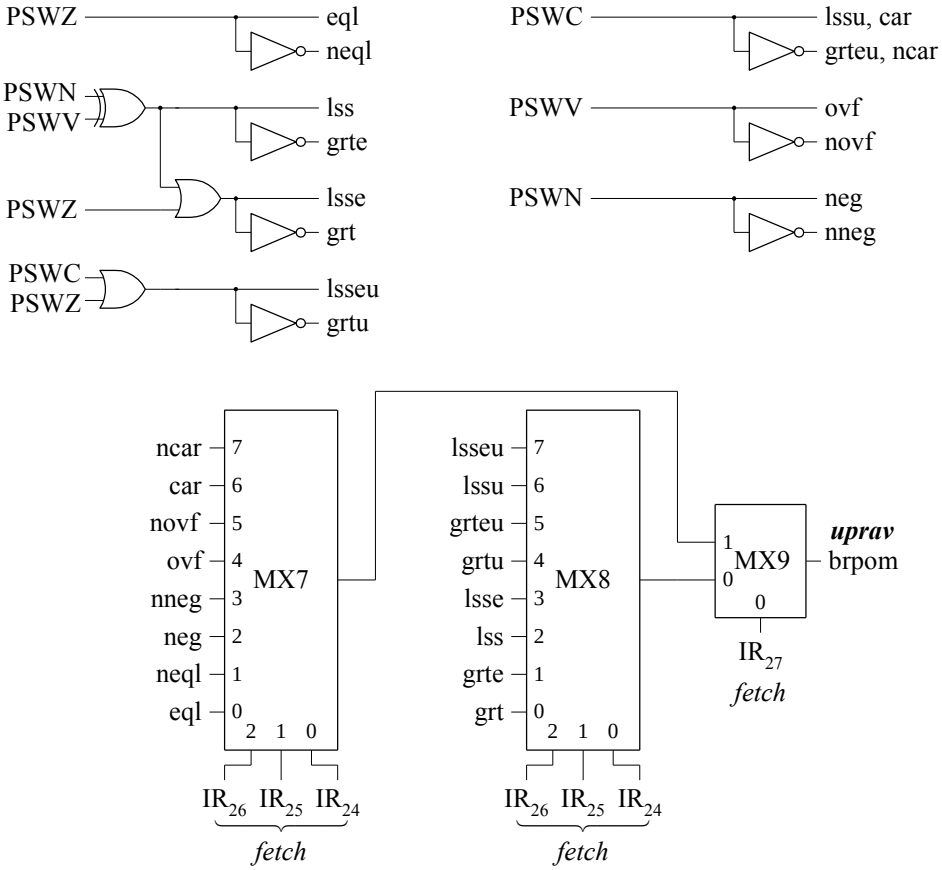
Kombinacione mreže signala postavljanja indikatora N, Z, C i V se sastoje od logičkih elemenata (slika 18). Na izlazima kombinacionih mreža se formiraju signali koji se upisuju u flip-flopove PSWN, PSWZ, PSWC i PSWV programske statusne reči u okviru izvršavanja određenih instrukcija (odeljak 2.1.5). Signal **N** ima vrednost koja odgovara vrednosti razreda AB_7 ukoliko je vrednost 1 jednog od signala operacija LDB, ..., ROLC, dok je u svim ostalim situacija vrednost signala **N** je 0. Signal **Z** ima vrednost 1 ukoliko vrednost 0 imaju svi razredi registra $AB_{7...0}$ i ukoliko je vrednost 1 ima jedan od signala operacija LDB, ..., ROLC, dok u svim ostalim situacija vrednost signala **Z** je 0. Signal **C** ima vrednost koja odgovara vrednosti signala $\mathbf{C_8}$ ukoliko je vrednost 1 jednog od signala operacija ADD ili INC i vrednost koja odgovara komplementu vrednost signala $\mathbf{C_8}$ ukoliko je vrednost 1 jednog od signala operacija SUB ili DEC. Pored toga signal **C** ima vrednost koja odgovara vrednosti signala $\mathbf{AB_0}$ ukoliko je vrednost 1 jednog od signala operacija ASR, LSR, ROR ili RORC i vrednost koja odgovara vrednosti signala $\mathbf{AB_7}$ ukoliko je vrednost 1 jednog od signala operacija ASL, LSL, ROL ili ROLC. U svim ostalim situacija vrednost signala **C** je 0. Signal **V** ima vrednost koja odgovara

vrednosti jednog od signala $\mathbf{V_{ADD}}$, $\mathbf{V_{SUB}}$, $\mathbf{V_{INC}}$ i $\mathbf{V_{DEC}}$, ukoliko je vrednost 1 signala operacija ADD, SUB, INC ili DEC, respektivno, dok u svim ostalim situacija vrednost signala **V** je 0. Signal $\mathbf{V_{ADD}}$ ima vrednost 1 ukoliko ili signali $\mathbf{AB_7}$ i $\mathbf{BB_7}$ imaju vrednost 0 i signal $\mathbf{ALU_7}$ ima vrednost 1 ili ukoliko signali $\mathbf{AB_7}$ i $\mathbf{BB_7}$ imaju vrednost 1 i signal $\mathbf{ALU_7}$ vrednost 0. Signal $\mathbf{V_{SUB}}$ ima vrednost 1 ukoliko ili signali $\mathbf{ALU_7}$ i $\mathbf{BB_7}$ imaju vrednost 0 i signal $\mathbf{AB_7}$ vrednost 1 ili signali $\mathbf{ALU_7}$ i $\mathbf{BB_7}$ imaju vrednost 1 i signal $\mathbf{AB_7}$ ima vrednost 0. Signal $\mathbf{V_{INC}}$ ima vrednost 1 ukoliko je signal $\mathbf{AB_7}$ ima vrednost 0 i signal $\mathbf{ALU_7}$ vrednost 1. Signal $\mathbf{V_{DEC}}$ ima vrednost 1 ukoliko signal $\mathbf{AB_7}$ ima vrednost 1 i signal $\mathbf{ALU_7}$ vrednost 0.



Slika 18 Blok exec (treći deo)

Kombinacione mreže za formiranje signala rezultata operacija **eql**, ..., **nneg** i **brpom** se sastoje od logičkih elemenata i multipleksera MX7, MX8 i MX9 (slika 19). Logičkim elementima se na osnovu sadržaja flip-flopova PSWN, PSWZ, PSWC i PSWV programske statusne reči formiraju signali **eql**, ..., **nneg**. Multiplekserima MX7, MX8 i MX9 se na osnovu razreda $IR_{27...24}$ prihvatnog registra instrukcije kojima se specificira kod operacije instrukcije uslovnog skoka BEQL, ..., BNNEQ selektuje jedan od signala **eql**, ..., **nneg** i pojavljuje kao signal **brpom** koji se koristi u upravljačkoj jedinici ***uprav*** da bi se prilikom izvršavanja instrukcija uslovnog skoka utvrdilo da li je uslov za skok ispunjen ili nije. Selekcija jednog od signala **eql**, ..., **nneg** i formiranje signala **brpom** se realizuje na osnovu kodiranja odgovarajućih kodova operacija instrukcije uslovnog skoka BEQL, ..., BNNEQ. Kodiranje instrukcija je tako realizovano da se razredom IR_{27} selektuje jedna od dve podgrupe G0_PG2 (poglavlje 2.1.5.1.1.1) i G0_PG3 (poglavlje 2.1.5.1.1.1) instrukcija uslovnog skoka (tabela 4), dok se razredima $IR_{26...24}$ unutar selektovane podrupe selektuje odgovarajuća instrukcija (tabele 7 i 8).



Slika 19 Blok exec (četvrti deo)

Signali rezultata operacija **eql**, ..., **nneg** imaju sledeće značenje:
eql — rezultat nula,
neq — rezultat različit od nule,
grt — rezultat veći od nule u aritmetici sa znakom,
gre — rezultat veći od nule ili jednak nuli u aritmetici sa znakom,
lss — rezultat manji od nule u aritmetici sa znakom,
leq — rezultat manji od nule ili jednak nuli u aritmetici sa znakom,
grtu — rezultat veći od nule u aritmetici bez znaka,
greu — rezultat veći od nule ili jednak nuli u aritmetici bez znaka,
lssu — rezultat manji od nule u aritmetici bez znaka,

lequ — rezultat manji od nule ili jednak nuli u aritmetici bez znaka,
neg — razred PSWN je jedan,
nng — razred PSWN nije jedan,
ovf — razred PSWV je jedan i
nvf — razred PSWV nije jedan.

Signali rezultata operacija **eql**, ..., **nvf** se realizuju prema sledećim izrazima:
eql = **PSWZ**,
neq = $\overline{\mathbf{PSWZ}}$,
grt = $\overline{((\mathbf{PSWN} \oplus \mathbf{PSWV}) + \mathbf{PSWZ})}$,
gre = $\overline{(\mathbf{PSWN} \oplus \mathbf{PSWV})}$,
lss = **PSWN** ⊕ **PSWV**,
leq = (**PSWN** ⊕ **PSWV**) + **PSWZ**,
grtu = $\overline{\mathbf{PSWC}} + \overline{\mathbf{PSWZ}}$,
greu = $\overline{\mathbf{PSWC}}$,
lssu = **PSWC**,
lequ = **PSWC** + **PSWZ**,
neg = **PSWN**,
nng = $\overline{\mathbf{PSWN}}$,
ovf = **PSWV** i
nvf = $\overline{\mathbf{PSWV}}$.

3.1.5 Blok intr

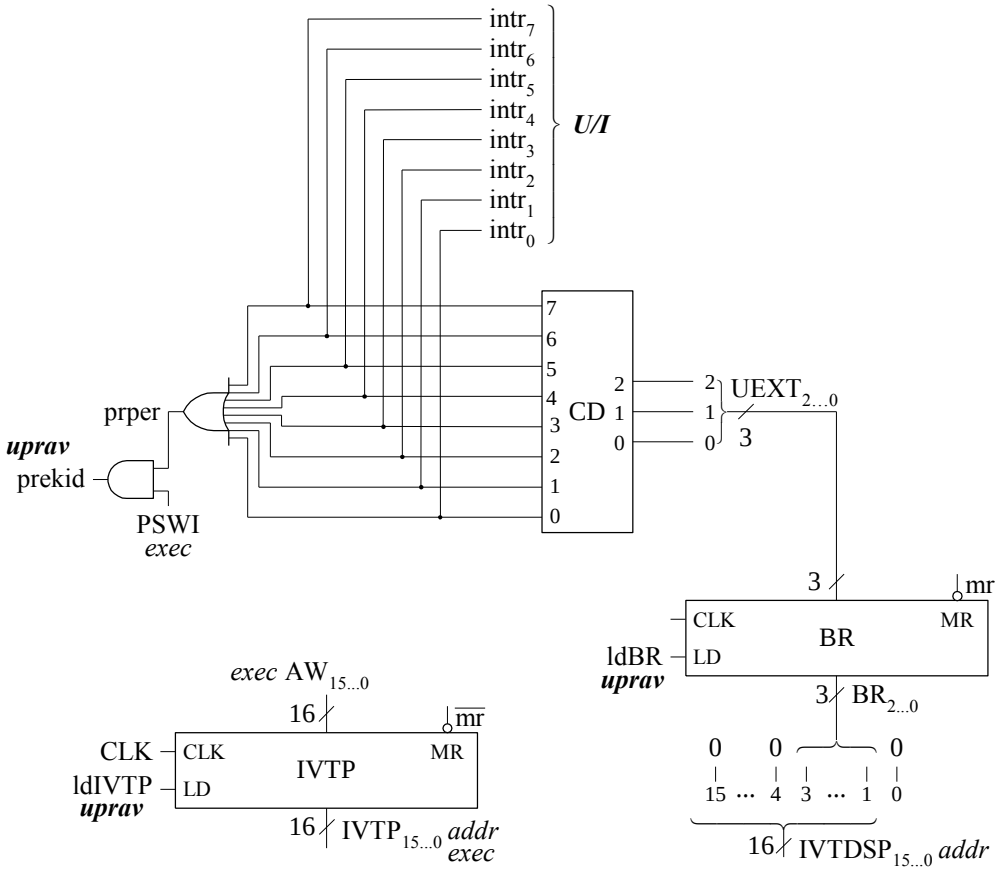
Blok *intr* sadrži logički ILI i I element za formiranje signala prekida **prekid**, koder prioriteta CD za formiranje broja ulaza $UEXT_{2...0}$ u IV tabelu, registar $BR_{2...0}$ za pamćenje vrednosti broja ulaza $UEXT_{2...0}$, mrežu za formiranje pomeraja za tabelu sa adresama prekidnih rutina $IVTDSP_{15...0}$ i registar ukazivača na tabelu sa adresama prekidnih rutina $IVTP_{15...0}$ (slika 20).

Signal prekida **prekid** se formira kao I funkcija signala **prper** i PSWI. Signal **prper** se formira kao ILI funkcija signala prekida $\mathbf{intr_0}$ do $\mathbf{intr_7}$. Signal PSWI se instrukcijama INTE i INTD postavlja na vrednostima 1 i 0, respektivno, i time zadaje režim rada procesora sa prihvatanjem ili maskiranjem prekida $\mathbf{intr_0}$ do $\mathbf{intr_7}$, respektivno. Da bi signal prekida **prekid** imao vrednost 1 potrebno je da barem jedan od signala $\mathbf{intr_0}$ do $\mathbf{intr_7}$ ima vrednost 1 i da signal PSWI ima vrednost 1. sa prihvatanjem maskirajućih prekida.

Koder CD služi za formiranje broja ulaza u tabelu sa adresama prekidnih rutina za spoljašnje maskirajuće prekide $intr_0$ do $intr_7$. Signali $intr_0$ do $intr_7$ se vode na ulaze 0 do 7 kodera prioriteta CD po rastućim prioritetima. Na izlazu kodera dobija se 3-bitna binarna vrednost $UEXT_{2...0}$ ulaza kodera najvišeg prioriteta na kome signal zahteva za prekid ima vrednost 1. Vrednost $UEXT_{2...0}$, koja predstavlja broj ulaza u IV tabelu u kome se nalazi adresa prekidne rutine za zahtev za prekid najvišeg prioriteta, se vodi na ulaze registra $BR_{2...0}$.

Registar $BR_{2...0}$ služi za pamćenje vrednosti broja ulaza u tabelu sa adresama prekidnih rutina $UEXT_{2...0}$. Vrednosti $UEXT_{2...0}$ se upisuje u registar $BR_{2...0}$ vrdnošću 1 signala **ldBR**. Sadržaj registra $BR_{2...0}$ se koristi za formiranje pomeraja $IVTDSP_{15...0}$ za ulazak u tabelu sa adresama prekidnih rutina. Kako adresa prekidne rutine zauzima dve susedne memorijske lokacije potrebno je dobiti pomeraj množenjem broja ulaza sa 2, što se realizuje pomeranjem sadržaja registra $BR_{2...0}$ za jedno mesto ulevo. Pomeraj $IVTDSP_{15...0}$ je formiran tako što se bit

$IVTDSP_0$ postavlja na vrednost 0, bitovi $IVTDSP_{3...1}$ na vrednost bitova $BR_{2...0}$ i bitovi $IVTDSP_{15...4}$ na vrednost 0.



Slika 20 Blok intr (prvi deo)

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

3.2 UPRAVLJAČKA JEDINICA

U ovom odeljku se daju dijagram toka izvršavanja instrukcija, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice ***uprav***.

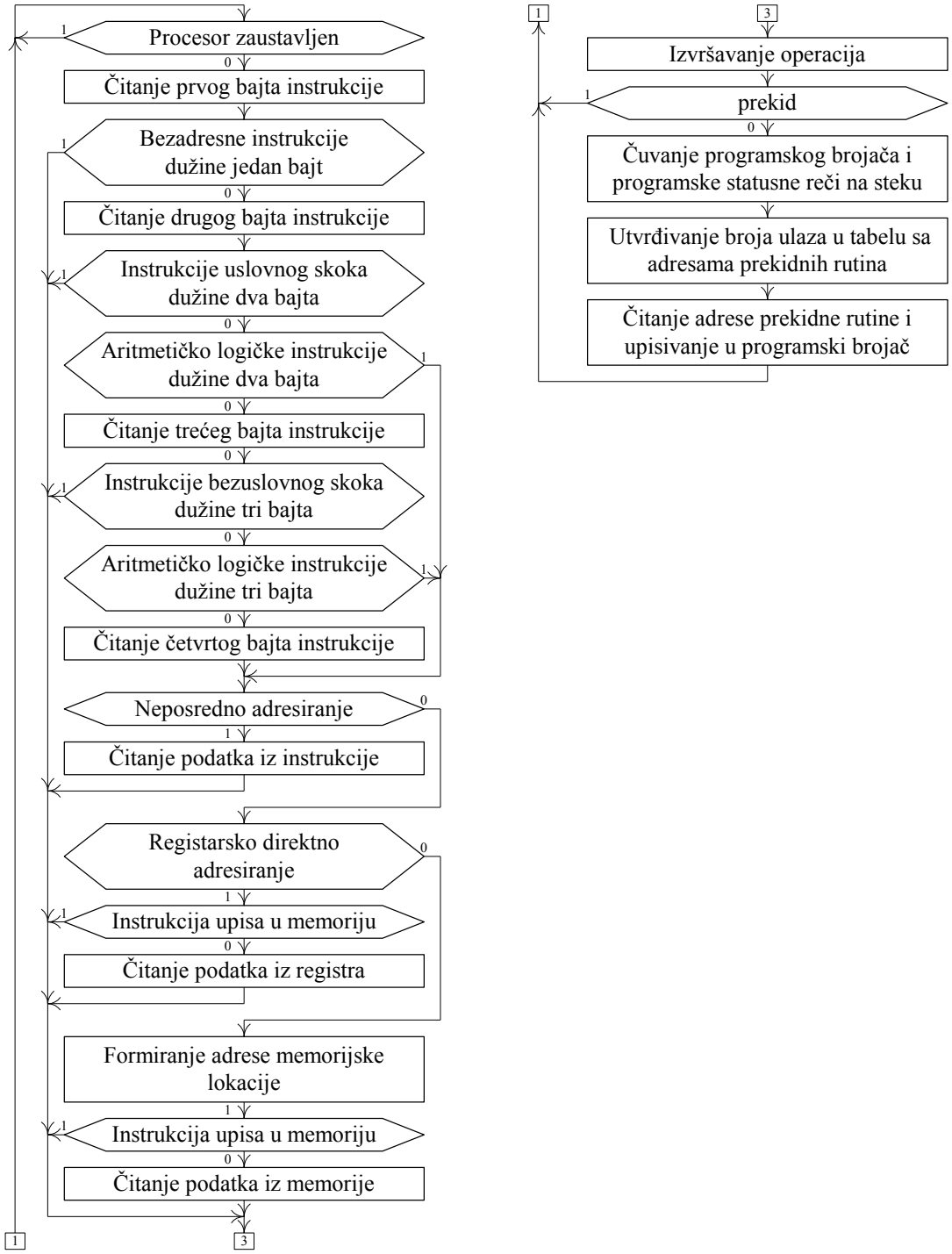
3.2.1 Dijagram toka izvršavanja instrukcija

Dijagram toka izvršavanja instrukcija je predstavljen operacionim i uslovnim blokovima (slika 21). U operacionim blokovima se nalaze opisi mikrooperacija koje treba realizovati. U uslovnim blokovima se nalaze opisi logičkih uslova koji definišu grananja algoritma.

U početnom koraku dijagram toka operacija vrši se provera da li je procesor zaustavljen ili ne. Ako je procesor zaustavljen ostaje se u početnom koraku i čeka startovanje procesora. Ako procesor nije zaustavljen prelazi se na korake u kojima se realizuje faza čitanje instrukcije.

U okviru faze čitanje instrukcije najpre se čita prvi bajt instrukcije u kome se nalazi kod operacije.

U nastavku faze čitanje instrukcije proverava se da li vrednost koda operacije odgovara vrednostima bezadresnih instrukcija čija je dužina jedan bajt. Ukoliko nije nastavlja se sa fazom čitanje instrukcije, a ukoliko jeste prelazi na fazu izvršavanje operacije.



Slika 21 Dijagram toka izvršavanja instrukcija

U nastavku faze čitanje instrukcije čita se drugi bajt instrukcije.

U nastavku faze čitanje instrukcije proverava se da li vrednost koda operacije odgovara vrednostima instrukcija uslovnog skoka čija je dužina dva bajta. Ukoliko nije nastavlja se sa fazom čitanje instrukcije, a ukoliko jeste prelazi na fazu izvršavanje operacije.

U nastavku faze čitanje instrukcije proverava se da li vrednost koda operacije odgovara vrednostima aritmetičko logičkih instrukcija i da li vrednost načina adresiranja odgovara vrednostima za direktno registarsko adresiranje i indirektno registarsko adresiranje čija je

dužina dva bajta. Ukoliko nije nastavlja se sa fazom čitanje instrukcije, a ukoliko jeste prelazi na fazu formiranje adrese operanda.

U nastavku faze čitanje instrukcije čita se treći bajt instrukcije i proverava se da li vrednost koda operacije odgovara vrednostima instrukcija bezuslovnog skoka čija je dužina tri bajta. Ukoliko nije nastavlja se sa fazom čitanje instrukcije, a ukoliko jeste prelazi na fazu izvršavanje operacije.

U nastavku faze čitanje instrukcije proverava se da li vrednost koda operacije odgovara vrednostima aritmetičko logičkih instrukcija i da li vrednosti načina adresiranja odgovara vrednosti za neposredno adresiranje čija je dužina tri bajta. Ukoliko nije nastavlja se sa fazom čitanje instrukcije, a ukoliko jeste prelazi na fazu formiranje adrese operanda.

U nastavku faze čitanje instrukcije čita se četvrti bajt instrukcije i prelazi na fazu formiranje adrese operanda.

U okviru faze formiranje adrese operanda se postupa različito u zavisnosti od toga da li se radi o instrukcijama upisa ili čitanja operanda, a u slučaju da se radi o instrukcijama čitanja operanda i od toga da li se operand nalazi u instrukciji, nekom od registara opšte namene ili memorijskoj lokaciji. U slučaju neposrednog adresiranja operand se čita neposredno iz instrukcije i prelazi na fazu izvršavanje operacije. U slučaju direktnog registarskog adresiranja operand se čita iz registra opšte namene i prelazi na fazu izvršavanje operacije ukoliko se ne radi o instrukciji upisa i odmah se prelazi na fazu izvršavanje operacije ukoliko se radi o instrukciji upisa. U svim ostalim slučajevima se radi o nekom od memorijskih adresiranja, pa se saglasno specificiranom načinu adresiranja formira adresa memorijske lokacije. Operand se čita iz memorijske lokacije i prelazi na fazu izvršavanje operacije ukoliko se ne radi o instrukciji upisa i odmah se prelazi na fazu izvršavanje operacije ukoliko se radi o instrukciji upisa.

U okviru faze izvršavanje operacija prelazi se na izvršavanje odgovarajuće operacije saglasno vrednosti koda operacije i prelazi na fazu opsluživanje prekida. Instrukcijama prenosa se ostvaruju prenosi iz različitih izvorišta u akumulatore procesora i obrnuto. Aritmetičkim instrukcijama se realizuju aritmetičke operacije nad sadržajima akumulatora i specificiranog operanda, a rezulta smešta u akumulator. Logičkim instrukcijama se realizuju logičke operacije nad sadržajima akumulatora i specificiranog operanda, a rezulta smešta u akumulator. Instrukcijama pomeranja i rotiranja se vrše pomeranja ulevo i udesno sadržaja akumulatora, a rezultat smešta u akumulator. Instrukcijama skoka se realizuju bezuslovni i uslovni skokovi u programu, skok na potprogram, povratak iz potprograma i povratak iz prekidne rutine. Instrukcijama postavljanja indikatora u PSW se dozvoljava ili zabranjuje reakcija na maskirajuće prekide.

U okviru faze opsluživanje prekida proverava se da li postoji zahtev za prekid koji je dozvoljen. Ukoliko ne postoji vraća se u početni korak i kreće sa fazom čitanje instrukcije prve sledeće instrukcije, dok se u suprotnom slučaju nastavlja sa fazom opsluživaje prekida. U nastavku faze opsluživanje prekida sa najpre na stek stavljaju programski brojač i programska statusna reč. Posle toga se formira broj ulaza u tabelu sa adresama prekidnih rutina. Na kraju se, sabiranjem broja ulaza pretvorenog u pomeraj i registra koji ukazuje na početnu adresu tabele sa adresama prekidnih rutina, dobija adresa memorijske lokacije sa koje se čita adresa prekidne rutine koja se upisuje u programski brojač i vraća u početni korak. Time se kreće sa fazom čitanje instrukcije prve instrukcije prekidne rutine.

3.2.2 Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu dijagrama toka operacija (slika 21) i dat je u obliku dijagrama toka mikrooperacija, dijagrama toka upravljačkih signala (slike 22 do 48) i sekvence upravljačkih signala (tabela 13).

Dijagram toka mikrooperacija i dijagram toka upravljačkih signala su dati istovremeno i predstavljeni su operacionim i uslovnim blokovima. U operacionim blokovima dijagrama toka mikrooperacija se nalaze mikrooperacije i uslovi pod kojima se one izvršavaju, dok se u operacionim blokovima dijagrama toka upravljačkih signala nalaze upravljački signali i uslovi pod kojima se oni generišu. U uslovnim blokovima dijagrama toka mikrooperacija i dijagrama toka upravljačkih signala se nalaze signali logičkih uslova.

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

signali.

Ovaj iskaz sadrži spisak upravljačkih signala blokova operacione jedinice ***oper*** 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.

Objašnjenja vezana za generisanje upravljačkih signala su data zajednički za dijagram toka upravljačkih signala i sekvencu upravljačkih signala i to u okviru sekvence upravljačkih signala.

Tabela 13 Sekvenca upravljačkih signala

! Sekvenca upravljačkih signala ima četiri celine koje odgovaraju fazama čitanje instrukcije, formiranje adrese i čitanje operanda, izvršavanje operacije i opsluživanje prekida. Faza čitanje instrukcije se realizuje u koracima $step_{00}$ do $step_{0F}$ koji su zajednički za sve instrukcije. Faza formiranje adrese i čitanje operanda se realizuje u koracima $step_{10}$ do $step_{1F}$ pri čemu postoje posebni koraci za svaki način adresiranja operanda. Faza izvršavanje operacije se realizuje u koracima $step_{30}$ do $step_{88}$, pri čemu postoje posebni koraci za svaku operaciju. Faza opsluživanje prekida se realizuje u koracima $step_{89}$ do $step_{9C}$ koji su zajednički za sve instrukcije. !

! Čitanje instrukcije !

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

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

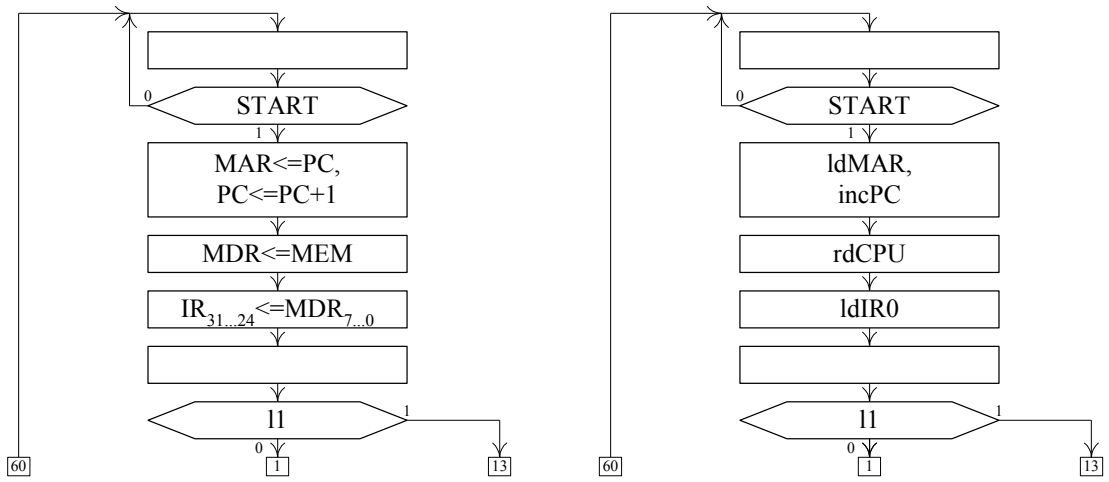
! U koracima $step_{01}$ do $step_{03}$ se čita prvi bajt instrukcije i smešta u razrede $IR_{31...24}$ prihvatnog registra instrukcije $IR_{31...0}$. Vrednostima 0 signala **$mxMAR_2$**, **$mxMAR_1$** i **$maMAR_0$** bloka *bus* se sadržaj registra $PC_{15..0}$ bloka *fetch* propušta kroz multiplekser MX1 i vrednošću 1 signala **ldMAR** upisuje u adresni registar $MAR_{15..0}$ bloka *bus*. Pored toga, vrednošću 1 signala **incPC** se sadržaj registra $PC_{15..0}$ inkrementira i prelazi na korak $step_{02}$. U koraku $step_{02}$ se realizuje čitanje jednog bajta i upisivanje u registar podatka $MDR_{7..0}$ bloka *bus*. Vrednošću 1 signala **rdCPU** se sadržaj registra $MAR_{15...0}$ pušta na adresne linije $ABUS_{15...0}$ magistrale ***BUS*** i generiše vrednost 0 signala na upravljačkoj liniji $\overline{\textbf{RDBUS}}$ magistrale ***BUS***, čime se u memoriji ***MEM*** startuje operacija čitanja. Pročitani sadržaj

koji memorija ***MEM*** pušta na izlazne linije podataka $DOBUS_{7...0}$ magistrale ***BUS*** se vrednošću 1 signala **rdCPU** upisuje u registar podatka $MDR_{7..0}$ i prelazi na korak $step_{03}$. U koraku $step_{03}$ se vrednošću 1 signala **ldIR0** bloka *fetch* pročitani bajt prebacuje iz registra $MDR_{7..0}$ bloka *bus* u prihvatni registar instrukcije i to u razrede $IR_{31...24}$ bloka *bus*. !

$step_{01}$ **ldMAR**, **incPC**;
$step_{02}$ **rdCPU;**
$step_{03}$ **ldIR0**;

! U koraku $step_{04}$ se proverava vrednost signala **l1** bloka *fetch* da bi se utvrdilo da li je dužina instrukcije jedan bajt ili više bajtova. U zavisnosti od toga da li signal **l1** ima vrednost 1 ili 0, prelazi se ili na korak $step_{30}$ i fazu izvršavanje operacije ili korak $step_{05}$ i produžava sa čitanjem bajtova instrukcije. !

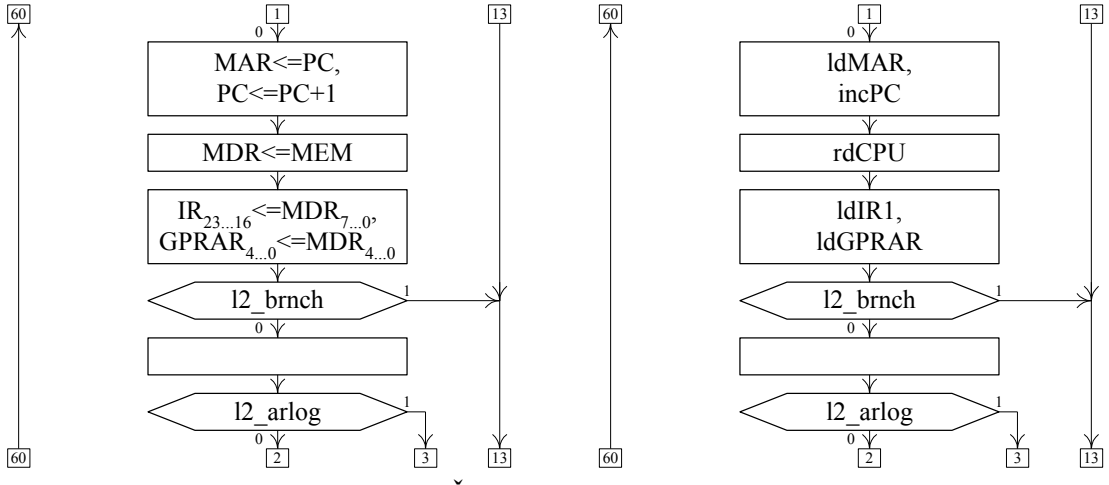
$step_{04}$ *br* (*if* **l1** *then* $step_{30}$);



Slika 22 Čitanje instrukcije (prvi deo)

! U koracima $step_{05}$ do $step_{07}$ se čita drugi bajt instrukcije i smešta u razrede $IR_{23...16}$ prihvatnog registra instrukcije $IR_{31...0}$. Koraci $step_{05}$ i $step_{06}$ u kojima se čita drugi bajt instrukcije su isti kao koraci $step_{01}$ i $step_{02}$ u kojima se čita prvi bajt instrukcije. !

$step_{05}$ **ldMAR**, **incPC**;
$step_{06}$ **rdCPU**;



Slika 23 Čitanje instrukcije (drugi deo)

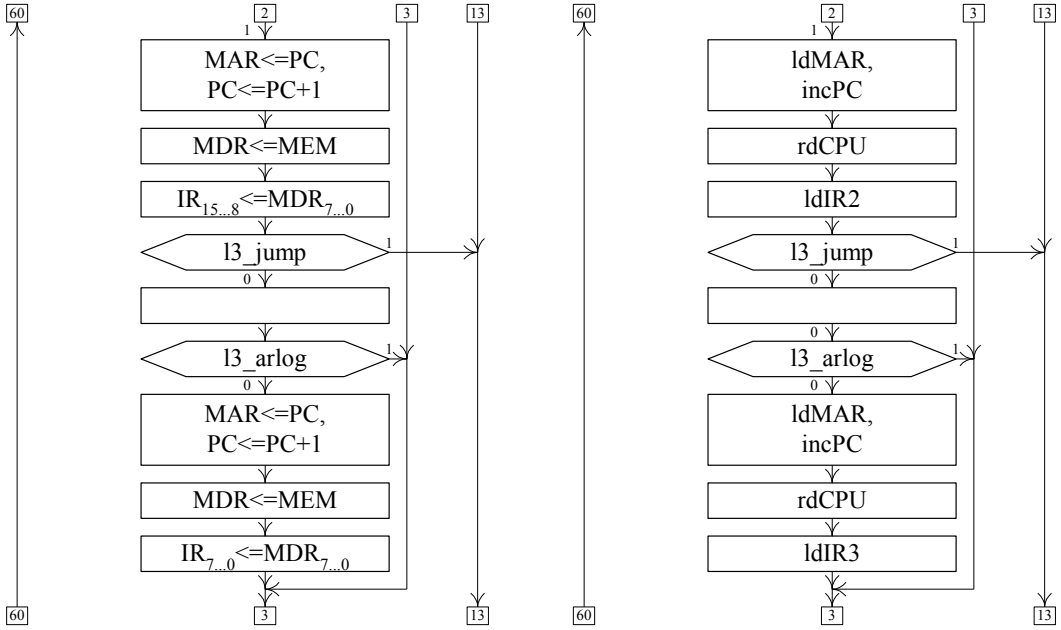
! U koraku $step_{07}$ se vrednošću 1 signala **ldIR1** bloka *fetch* pročitani bajt prebacuje iz registra $MDR_{7..0}$ bloka *bus* u prihvatni registar instrukcije i to u razrede $IR_{23...16}$ bloka *bus*. Istovremeno se vrednošću 1 signala **ldGPRADR** bloka *addr* razredi $MDR_{4..0}$ upisuju u registar $GPRADR_{4..0}$. Time se u registru $GPRADR_{4..0}$ nalazi adresa registra opšte namene koja se koristi samo u slučaju da se radi o instrukciji kojoj je potrebna ta adresa da bi se saglasno specificiranom načinu adresiranja došlo do operanda. !

! U koracima $step_{07}$ i $step_{08}$ se proverava vrednost signala **l2_brnch** i **l2_arlog** bloka *fetch* da bi se utvrdilo da li je dužina instrukcije dva bajta ili više bajtova. Iz koraka $step_{07}$ se u zavisnosti od toga da li signal **l2_brnch** ima vrednost 1 ili 0, ili prelazi na korak $step_{30}$ i fazu izvršavanje operacije ili $step_{08}$ i proveru vrednosti signala **l2_arlog**. Iz koraka $step_{08}$ se, u zavisnosti od toga da li signal **l2_arlog** ima vrednost 1 ili 0, ili prelazi na korak $step_{10}$ i fazu formiranje adrese i čitanje operanda ili korak $step_{09}$ i produžava sa čitanjem bajtova instrukcije.!

$step_{07}$ **ldIR1**, **ldGPRADR**,
br (*if* **l2_brnch** *then* $step_{30}$);
$step_{08}$ *br* (*if* **l2_arlog** *then* $step_{10}$);

! U koracima $step_{09}$ do $step_{0B}$ se čita treći bajt instrukcije i smešta u razrede $IR_{15...8}$ prihvatnog registra instrukcije $IR_{31...0}$. Koraci $step_{09}$ i $step_{0A}$ u kojima se čita treći bajt instrukcije su isti kao koraci $step_{01}$ i $step_{02}$ u kojima se čita prvi bajt instrukcije.!

$step_{09}$ **ldMAR**, **incPC**;
$step_{0A}$ **rdCPU**;



Slika 24 Čitanje instrukcije (treći deo)

! U koraku $step_{0B}$ se signalom **ldIR2** bloka *fetch* pročitani bajt prebacuje iz registra $MDR_{7..0}$ bloka *bus* u prihvatni registar instrukcije i to u razrede $IR_{15...8}$ bloka *bus*. !

! U koracima $step_{0B}$ i $step_{0C}$ se proverava vrednost signala **l3_jump** i **l3_arlog** bloka *fetch* da bi se utvrdilo da li je dužina instrukcije tri ili četiri bajta. Iz koraka $step_{0B}$ se, u zavisnosti od toga da li signal **l3_jump** ima vrednost 1 ili 0, ili prelazi na korak $step_{30}$ i fazu izvršavanje operacije ili $step_{0C}$ i proveru vrednosti signala **l3_arlog**. Iz koraka $step_{0C}$ se u zavisnosti od toga da li signal **l3_arlog** ima vrednost 1 ili 0, ili prelazi na korak $step_{10}$ i fazu formiranje adrese i čitanje operanda ili na korak $step_{0D}$ i produžava sa čitanjem četvrtog bajta instrukcije.!

$step_{0B}$ **ldIR2**,
br (*if* **l3_jump** *then* $step_{30}$);
$step_{0C}$ *br* (*if* **l3_arlog** *then* $step_{10}$);

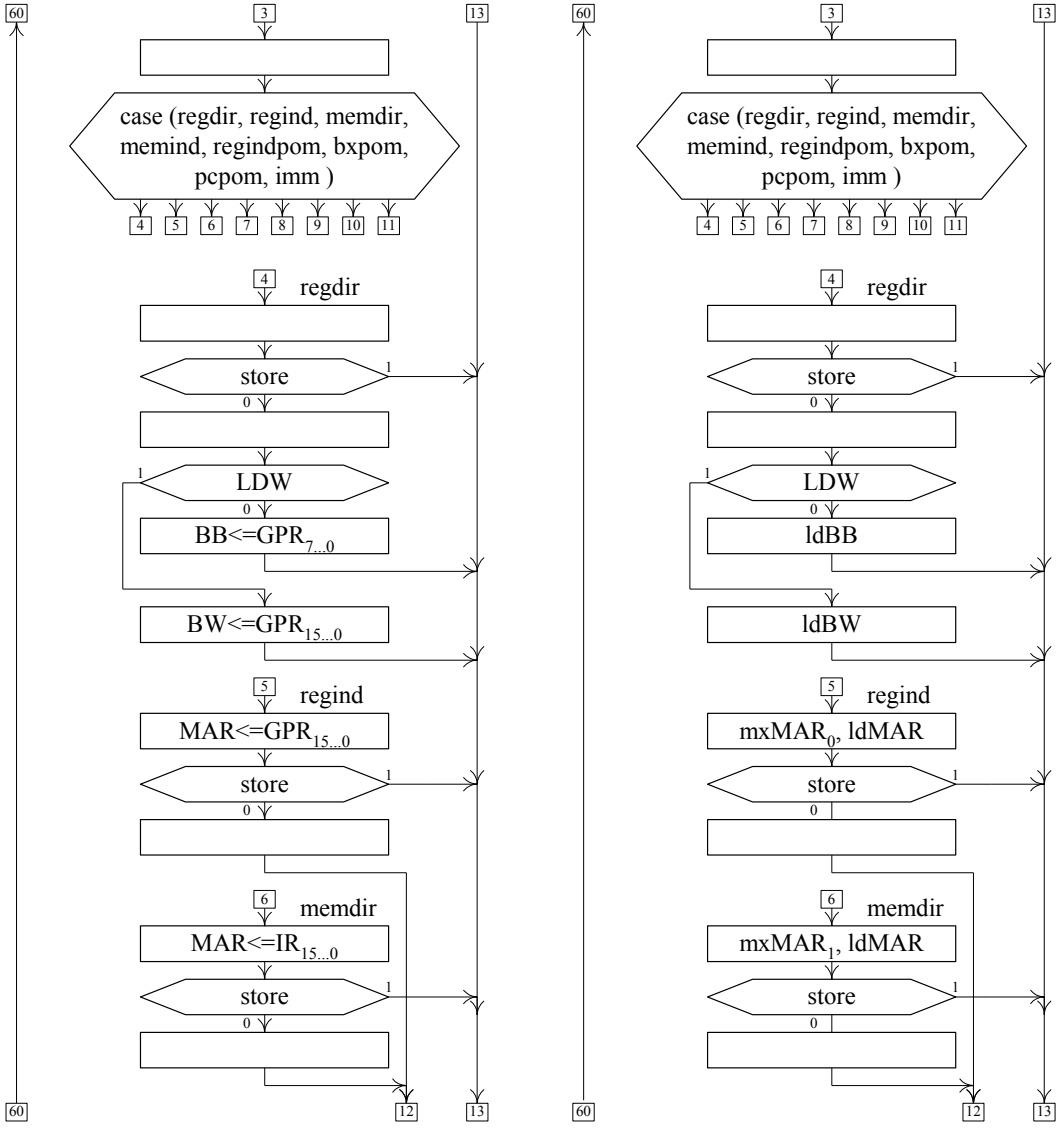
! U koracima $step_{0D}$ do $step_{0F}$ se čita četvrti bajt instrukcije i smešta u razrede $IR_{7...0}$ prihvatnog registra instrukcije $IR_{31...0}$. Koraci $step_{0D}$ i $step_{0E}$ u kojima se čita četvrti bajt instrukcije su isti kao koraci $step_{01}$ i $step_{02}$ u kojima se čita prvi bajt instrukcije. U koraku $step_{0F}$ se signalom **ldIR3** bloka *fetch* pročitani bajt prebacuje iz registra $MDR_{7..0}$ bloka *bus* u prihvatni registar instrukcije i to u razrede $IR_{7...0}$ bloka *bus*. Iz koraka $step_{0F}$ se prelazi na korak $step_{10}$ i fazu formiranje adrese i čitanje operanda. !

$step_{0D}$ **ldMAR**, **incPC**;
$step_{0E}$ **rdCPU**;

$step_{0F}$ **ldIR3**;

! Formiranje adrese i čitanje operanda !

! U korak $step_{20}$ se dolazi iz koraka $step_{08}$, $step_{0C}$ i $step_{0F}$ ukoliko se radi o instrukcijama dužine dva, tri i četiri bajta koje zahtevaju da se do operanda dođe saglasno specificiranom načinu adresiranja. Za sve instrukcije, sem instrukcija STB i STW, operand se smešta u registar $BB_{7...0}$ ili registar $BW_{15...0}$. Operand može da bude u nekom od registara opšte namene, u memorijskoj lokaciji ili samoj instrukciji. 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 registar $BB_{7...0}$ ili registar $BW_{15...0}$. 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 operand čita i prebacuje u registar $BB_{7...0}$ ili registar $BW_{15...0}$. Izuzetak su instrukcije STB i STW kod kojih se operand upisuje. U slučaju adresiranja kod koga se operand upisuje u registar opšte namene, odmah se prelazi na fazu izvršavanje operacije u kojoj se operand upisuje u registar opšte namene. U slučaju nekog od adresiranja kod kojih se operand upisuje u memoriju, u ovoj fazi se samo formira adresa operanda u registru $MAR_{15...0}$, pa se prelazi na fazu izvršavanje operacije u kojoj se operand upisuje u memoriju na formiranoj adresi. Pretpostavljeno je da se slučaju instrukcija STB i STW neće javiti neposrednos adresiranje i da se zato slučaj kada se upisuje u samu instrukciju ne javlja. U koraku $step_{10}$ se realizuje višestruki uslovni skok na jedan od koraka $step_{11}$, $step_{15}$, …, $step_{2D}$ u zavisnosti od toga koji od signala adresiranja **regdir**, **regind**, **memdir**, **memind**, **regindpom**, **bxpom**, **bcpom**, **imm** bloka *addr* ima vrednost 1. !



Slika 25 Formiranje adrese i čitanje operanda (prvi deo)

$step_{10}$ *br* (*case* (**regdir**, **regind**, **memdir**, **memind**, **regindpom**, **bxpom**, **bcpom**, **imm**) *then* (**regdir**, $step_{11}$), (**regind**, $step_{15}$), (**memdir**, $step_{17}$), (**memind**, $step_{19}$), (**regindpom**, $step_{20}$), (**bxpom**, $step_{22}$), (**pcpom**, $step_{25}$), (**imm**, $step_{2D}$));

! Registarsko direktno adresiranje !

! U korak $step_{11}$ se dolazi iz $step_{10}$ ukoliko signal za registarsko direktno adresiranje **regdir** ima vrednost 1. Ukoliko signal **store** bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama STB ili STW za koje nema čitanja operanda, prelazi se na korak $step_{30}$ i fazu izvršavanje operacije. U suprotnom slučaju se prelazi na korak $step_{12}$ u kome se proverava vrednost signala operacije **LDW**. Ukoliko signal **LDW** ima vrednost 1, prelazi se na korak $step_{14}$ u kome se vrednošću 1 signala **ldBW** 16-to razredni sadržaj adresiranog registra opšte namene $GPR_{15...0}$ bloka *addr* upisuje u registar $BW_{15...0}$ bloka *exec*. U suprotnom slučaju se prelazi na korak $step_{13}$ u kome se vrednošću 1 signala **ldBB** niži bajt adresiranog registra opšte namene $GPR_{7...0}$ upisuje u registar $BB_{7...0}$ bloka *exec*. U oba slučaja se prelazi na korak $step_{30}$ i fazu izvršavanje operacije. !

$step_{11}$ *br* (*if* **store** *then* $step_{30}$);

$step_{12}$ *br* (*if* **LDW** *then* $step_{14}$);

$step_{13}$ **ldBB**,
br $step_{30}$;

$step_{14}$ **ldBW**,
br $step_{30}$;

! Registarsko indirektno adresiranje !

! U korak $step_{15}$ se dolazi iz koraka $step_{10}$ ukoliko signal za registarsko indirektno adresiranje **regind** ima vrednost 1. Vrednošću 1 signala $\mathbf{mxMAR_0}$ se sadržaj adresiranog registra opšte namene $GPR_{15...0}$ bloka *addr* propušta kroz multiplekser MPX1 bloka *bus* i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$ bloka *bus*. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj registarskog indirektnog adresiranja. Ukoliko signal **store** bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama STB ili STW za koje nema čitanja operanda, prelazi se na korak $step_{30}$ i fazu izvršavanje operacije. U suprotnom slučaju se prelazi na korak $step_{16}$ iz koga se bezuslovno prelazi na korak $step_{26}$ i čitanje operanda. !

$step_{15}$ $\mathbf{mxMAR_0}$, **ldMAR**,
br (*if* **store** *then* $step_{30}$);

$step_{16}$ *br* $step_{26}$;

! Memorijsko direktno adresiranje !

! U korak $step_{17}$ se dolazi iz koraka $step_{10}$ ukoliko signal za memorijsko direktno adresiranje **memdir** ima vrednost 1. Vrednošću 1 signala $\mathbf{mxMAR_1}$ se sadržaj registra $IR_{15...0}$ bloka *fetch* propušta kroz multiplekser MX1 bloka *bus* i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj memorijskog direktnog adresiranja. Ukoliko signal **store** bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama STB ili STW za koje nema čitanja operanda, prelazi se na korak $step_{30}$ i fazu izvršavanje operacije. U suprotnom slučaju se prelazi na korak $step_{18}$ iz koga se bezuslovno prelazi na korak $step_{26}$ i čitanje operanda. !

$step_{17}$ $\mathbf{mxMAR_1}$, **ldMAR**,
br (*if* **store** *then* $step_{30}$);

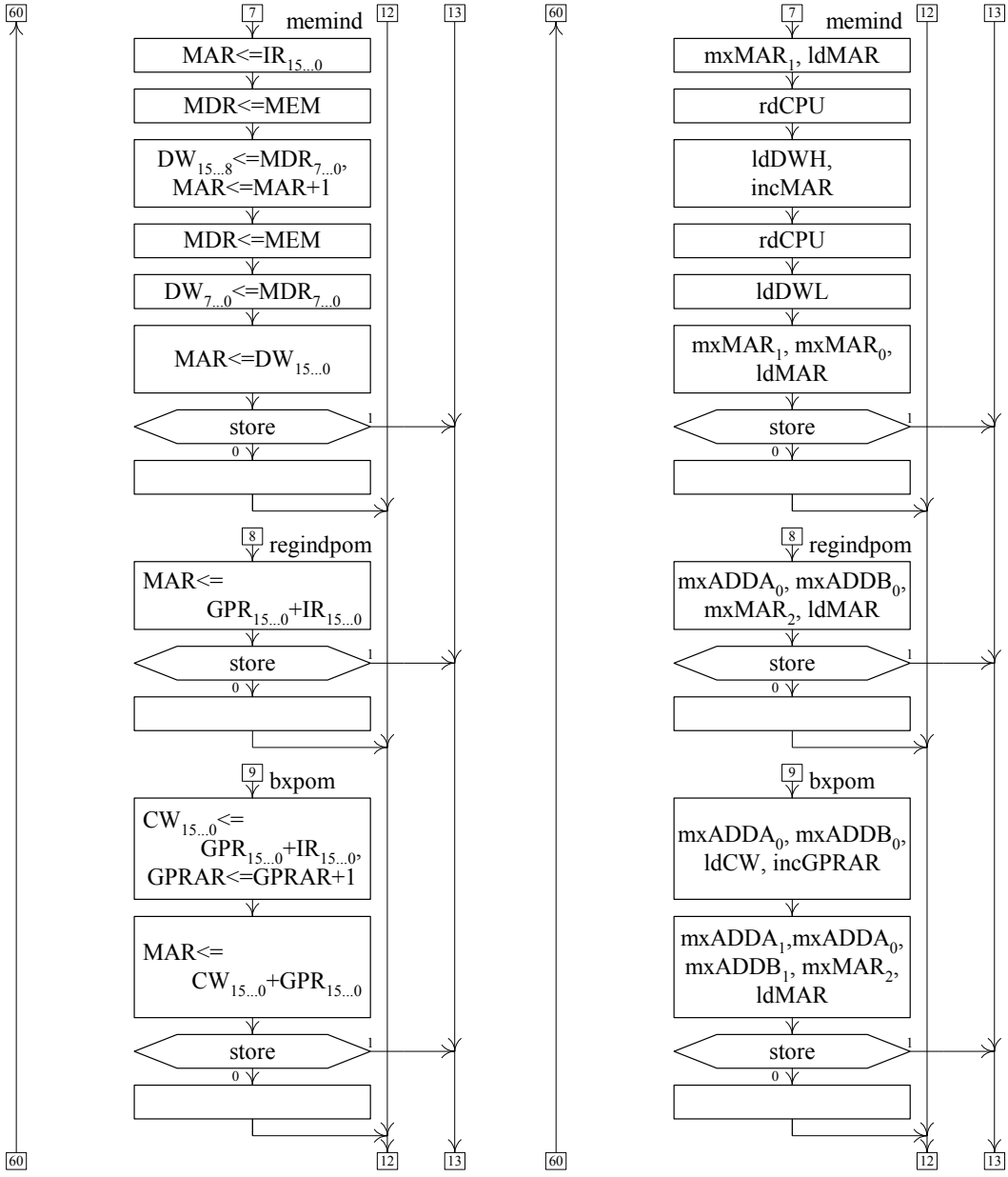
$step_{18}$ *br* $step_{26}$;

! Memorijsko indirektno adresiranje !

! U korak $step_{19}$ se dolazi iz $step_{10}$ ukoliko signal za memorijsko indirektno adresiranje **memind** ima vrednost 1. Vrednošću 1 signala $\mathbf{mxMAR_1}$ se sadržaj registra $IR_{15...0}$ bloka *fetch* propušta kroz multiplekser MX1 bloka *bus* i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa memorijske lokacije počev od koje treba pročitati dva bajta koji predstavljaju viši i niži bajt adrese operanda za slučaj memorijskog indirektnog adresiranja. Čitanje prvog bajta se realizuje u koraku $step_{1A}$, a drugog bajta u koracima $step_{1C}$, na isti način kao u koraku $step_{02}$ kod čitanja prvog bajta instrukcije. U koraku $step_{1B}$ se iz registra $MDR_{7...0}$ bloka *bus* vrednošću 1 signala **ldDWH** prvi bajt upisuje u viši bajt registra $DW_{15...0}$ bloka *bus*, a vrednošću 1 signala **incMAR** adresni registar $MAR_{15...0}$ bloka *bus* inkrementira na adresu sledećeg bajta. U koraku $step_{1D}$ se iz registra $MDR_{7...0}$ bloka *bus* vrednošću 1 signala **ldDWL** drugi bajt upisuje u niži bajt registra

$DW_{15...0}$. Na kraju se u koraku $step_{1E}$ vrednostima 1 signala **mxMAR$_1$** i **mxMAR$_0$** sadržaj registra $DW_{15...0}$ koji predstavlja adresu operanda propušta kroz multiplekser MX1 bloka *bus* i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$. Ukoliko signal **store** bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama STB ili STW za koje nema čitanja operanda, prelazi se na korak $step_{30}$ i fazu izvršavanje operacije. U suprotnom slučaju se prelazi na korak $step_{1F}$ iz koga se bezuslovno prelazi na korak $step_{26}$ i čitanje operanda. !

$step_{19}$ **mxMAR$_1$**, **ldMAR**;
$step_{1A}$ **rdCPU**;
$step_{1B}$ **ldDWH**, **incMAR**;
$step_{1C}$ **rdCPU**;
$step_{1D}$ **ldDWL**;
$step_{1E}$ **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**,
br (*if* **store** *then* $step_{30}$);
$step_{1F}$ *br* $step_{26}$;



Slika 26 Formiranje adrese i čitanje operanda (drugi deo)

! Registarsko indirektno adresiranje sa pomerajem !

! U korak $step_{20}$ se dolazi iz $step_{10}$ ukoliko signal za registarsko indirektno adresiranje sa pomerajem **regindpom** ima vrednost 1. Vrednostima 1 signala $\mathbf{mxADDA_0}$ i $\mathbf{mxADDB_0}$ se najpre kroz multipleksere MX2 i MX3 na ulaze sabirača ADD bloka *addr* propuštaju adresirani registar opšte namene $GPR_{15...0}$ bloka *addr* i pomeraj iz $IR_{15...0}$ bloka *fetch*, zatim se vrednošću 1 signala $\mathbf{mxMAR_2}$ dobijeni sadržaja sa izlaza sabirača ADD propušta kroz multiplekser MX1 bloka *bus* i na kraju vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj registarskog indirektnog adresiranja sa pomerajem. Ukoliko signal **store** bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama STB ili STW za koje nema čitanja operanda, prelazi se na korak $step_{30}$ i fazu izvršavanje operacije. U suprotnom slučaju se prelazi na korak $step_{21}$ iz koga se bezuslovno prelazi na korak $step_{26}$ i čitanje operanda. !

$step_{20}$ $\mathbf{mxADDA_0}$, $\mathbf{mxADDB_0}$, $\mathbf{mxMAR_2}$, **ldMAR**,
br (*if* **store** *then* $step_{30}$);
$step_{21}$ *br* $step_{26}$;

! Bazno indeksno adresiranje sa pomerajem !
! U korak $step_{22}$ se dolazi iz koraka $step_{10}$ ukoliko signal za bazno indeksno adresiranje sa pomerajem **bxpom** ima vrednost 1. U koraku $step_{22}$ se vrednostima 1 signala $\mathbf{mxADDA_0}$ i $\mathbf{mxADDB_0}$ kroz multipleksere MX2 i MX3 na ulaze sabirača ADD bloka *addr* propuštaju adresirani registar opšte namene $GPR_{15...0}$ bloka *addr* i pomeraj iz $IR_{15...0}$ bloka *fetch*, a vrednošću 1 signala **ldCW** sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD upisuje u registar $CW_{15...0}$ bloka *addr*. Pored toga vrednošću 1 signala **incGPRAR** bloka *addr* se vrši inkrementiranje adresnog registra $GPRAR_{4...0}$ registara opšte namene $GPR_{15...0}$. U koraku $step_{23}$ se vrednostima 1 signala $\mathbf{mxADDA_1}$, $\mathbf{mxADDA_0}$ i $\mathbf{mxADDB_1}$ kroz multipleksere MX2 i MX3 na ulaze sabirača ADD propuštaju sadržaji registra $CW_{15...0}$ i adresiranog registra opšte namene $GPR_{15...0}$, vrednošću 1 signala $\mathbf{mxMAR_2}$ se dobijeni sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD propušta kroz multiplekser MX1 bloka *bus* i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$ bloka *bus*. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj bazno indeksnog adresiranja sa pomerajem. Ukoliko signal **store** bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama STB ili STW za koje nema čitanja operanda, prelazi se na korak $step_{30}$ i fazu izvršavanje operacije. U suprotnom slučaju se prelazi na korak $step_{24}$ iz koga se bezuslovno prelazi na korak $step_{26}$ i čitanje operanda. !

$step_{22}$ $\mathbf{mxADDA_0}$, $\mathbf{mxADDB_0}$, **ldCW**, **incGPRAR**;
$step_{23}$ $\mathbf{mxADDA_1}$, $\mathbf{mxADDA_0}$, $\mathbf{mxADDB_1}$, $\mathbf{mxMAR_2}$, **ldMAR**,
br (*if* **store** *then* $step_{30}$);
$step_{24}$ *br* $step_{26}$;

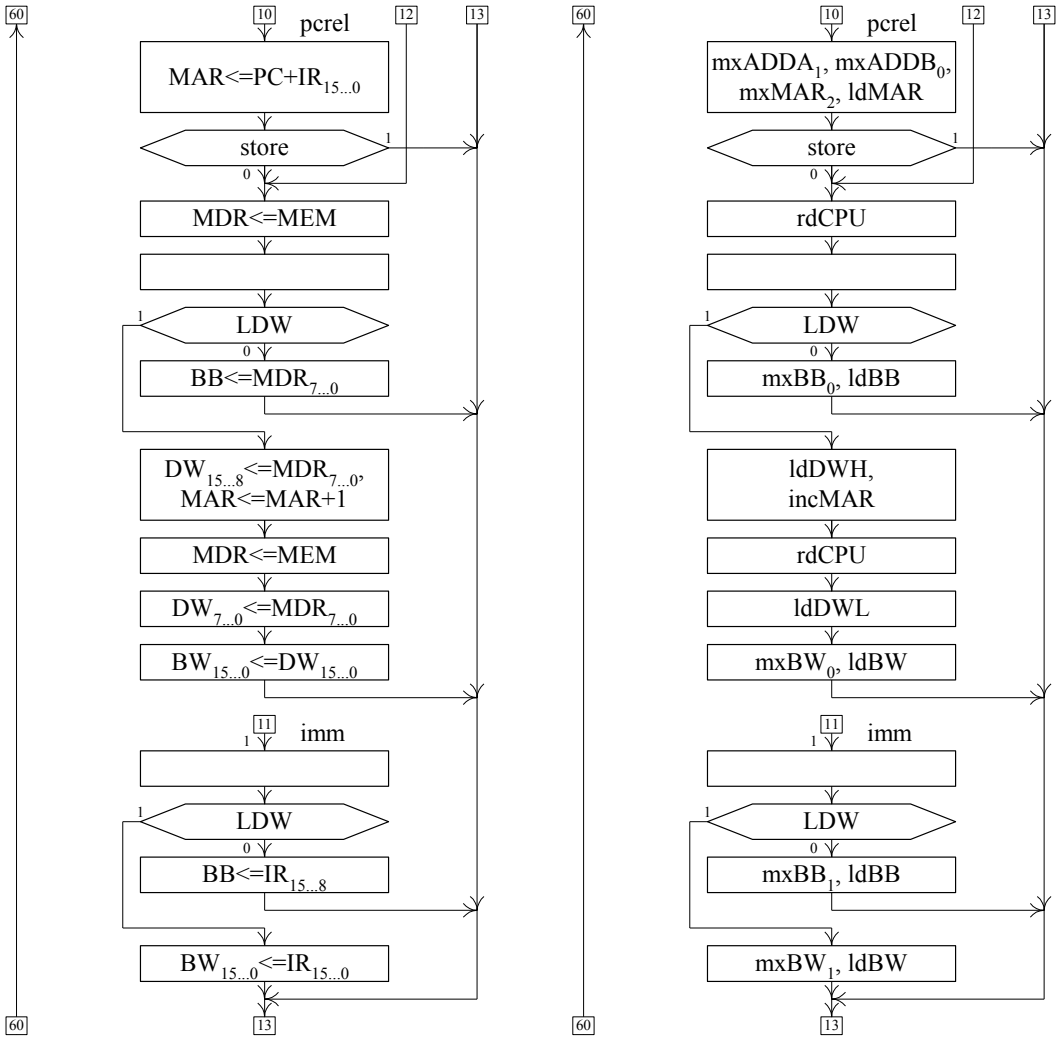
! PC relativno adresiranje !
! U korak $step_{25}$ se dolazi iz koraka $step_{10}$ ukoliko signal za PC relativno adresiranje **pcpom** ima vrednost 1. Vrednostima 1 signala $\mathbf{mxADDA_1}$ i $\mathbf{mxADDB_0}$ se kroz multipleksere MX2 i MX3 na ulaze sabirača ADD bloka *addr* propuštaju programski brojač $PC_{15...0}$ bloka *fetch* i pomeraj iz $IR_{15...0}$ bloka *fetch*, vrednošću 1 signala $\mathbf{mxMAR_2}$ se dobijeni sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD propušta kroz multiplekser MX1 bloka *bus* i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$ bloka *bus*. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj PC relativnog adresiranja. Ukoliko signal **store** bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama STB ili STW za koje nema čitanja operanda, prelazi se na korak $step_{30}$ i fazu izvršavanje operacije. U suprotnom slučaju se prelazi na korak $step_{26}$ i čitanje operanda sa adrese koja se nalazi u registru $MAR_{15...0}$. !

$step_{25}$ $\mathbf{mxADDA_1}$, $\mathbf{mxADDB_0}$, $\mathbf{mxMAR_2}$, **ldMAR**,
br (*if* **store** *then* $step_{30}$);

! Čitanje operanda !
! U korak $step_{26}$ se dolazi iz $step_{16}$ kod registarskog indirektnog adresiranja, iz $step_{18}$ kod memorijskog direktnog adresiranja, iz $step_{1F}$ kod memorijskog indirektnog adresiranja, iz $step_{21}$ kod registarskog indirektnog adresiranja sa pomerajem, iz $step_{24}$ kod baznog indeksnog adresiranja sa pomerajem i iz $step_{25}$ kod PC relativnog adresiranja sa pomerajem. U svim ovim situacijama adresa memorijske lokacije sa koje treba pročitati operand je sračunata u saglasnosti sa specificiranim načinom adresiranja i nalazi se u registru $MAR_{15...0}$ bloka *bus*. Za sve instrukcije za koje se čita operand iz memorije dužina operanda je jedan bajt sem za instrukciju LDW za koju je dužina operanda dva bajta.

Zbog toga se najpre u koraku $step_{26}$ čita jedan bajt i to na isti način na koji se to radi u koraku $step_{02}$ u kome se čita prvi bajt instrukcije. Potom se u koraku $step_{27}$ u zavisnost od toga da li signal **LDW** ima vrednost 0 ili 1 prelazi na korak $step_{28}$ ili korak $step_{29}$, respektivno. Kroz korak $step_{28}$ se prolazi samo kada je dužina operanda jedan bajt. Tada se vrednostima 1 signala $\mathbf{mxBB_0}$ i **ldBB** bloka *exec* sadržaj registra $MDR_{7...0}$ bloka *bus* propušta kroz multiplekser MX2 i upisuje u registar $BB_{7...0}$. Iz koraka $step_{28}$ se prelazi na korak $step_{30}$ i fazu izvršavanje operacije. Kroz korake $step_{29}$ do $step_{2C}$ se prolazi samo kada je dužina operanda dva bajta. Najpre se u koraku $step_{29}$ vrednošću 1 signala **ldDWH** bloka *bus* sadržaj registra $MDR_{7...0}$ upisuje u stariji bajt registra $DW_{15...0}$ i vrednošću 1 signala **incMAR** inkrementira sadržaj registra $MAR_{15...0}$ da bi ukazivao na memorijsku lokaciju na kojoj se nalazi mlađi bajt 16-to razrednog operanda. Potom se u koraku $step_{2A}$ čita jedan bajt i to na isti način na koji se to radi u koraku $step_{02}$ u kome se čita prvi bajt instrukcije. Zatim se u koraku $step_{2B}$ vrednošću 1 signala **ldDWL** bloka *bus* sadržaj registra $MDR_{7...0}$ upisuje u mlađi bajt registra $DW_{15...0}$. Time se u registru $DW_{15...0}$ nalazi 16-to bitni operand. Na kraju se u koraku $step_{2C}$ vrednostima 1 signala $\mathbf{mxBW_0}$ i **ldBW** bloka *exec* sadržaj registra $DW_{15...0}$ propušta kroz multiplekser MX6 i upisuje u registar $BW_{15...0}$. Iz koraka $step_{2C}$ se prelazi na korak $step_{30}$ i fazu izvršavanje operacije. !

$step_{26}$ **rdCPU**;
$step_{27}$ *br* (*if* **LDW** *then* $step_{29}$);
$step_{28}$ $\mathbf{mxBB_0}$, **ldBB**,
br $step_{30}$;
$step_{29}$ **ldDWH**, **incMAR**;
$step_{2A}$ **rdCPU**;
$step_{2B}$ **ldDWL**;
$step_{2C}$ $\mathbf{mxBW_0}$, **ldBW**,
br $step_{30}$;



Slika 27 Formiranje adrese i čitanje operanda (treći deo)

! Neposredno adresiranje !

! U korak $step_{2D}$ se dolazi iz $step_{10}$ ukoliko signal za neposredno adresiranje **imm** ima vrednost 1. Iz ovog koraka se ukoliko signal **LDW** ima vrednost 1 prelazi na korak $step_{2F}$ u kome se vrednostima 1 signala $\mathbf{mxBW_1}$ i **ldBW** bloka *exec* 16-to razredna neposredna veličina koja se nalazi u razredima $IR_{15...0}$ bloka *fetch* propušta kroz multiplekser MX6 i upisuje u registar $BW_{15...0}$. U suprotnom slučaju se prelazi na korak $step_{2E}$ u kome se vrednostima 1 signala $\mathbf{mxBB_1}$ i **ldBB** bloka *exec* 8-mo razredna neposredna veličina koja se nalazi u razredima $IR_{15...8}$ propušta kroz multiplekser MX2 i upisuje u registar $BB_{7...0}$. U oba slučaja se prelazi na korak $step_{30}$ i fazu izvršavanja operacije. !

$step_{2D}$ *br* (*if* **LDW** *then* $step_{2F}$);
$step_{2E}$ $\mathbf{mxBB_1}$, **ldBB**,
br $step_{30}$;
$step_{2F}$ $\mathbf{mxBW_1}$, **ldBW**;

! Izvršavanje operacije !

! U korak $step_{30}$ se dolazi iz koraka $step_{04}$, $step_{07}$, $step_{1B}$, $step_{11}$, $step_{13}$, $step_{14}$, $step_{15}$, $step_{17}$, $step_{1E}$, $step_{20}$, $step_{23}$, $step_{25}$, $step_{28}$, $step_{2C}$, $step_{2F}$ i $step_{2E}$ radi izvršavanja operacije. U koraku $step_{30}$ se realizuje višestruki uslovni skok na jedan od koraka $step_{31}$, $step_{32}$, ..., $step_{82}$ u zavisnosti od toga koji od signala operacija **INTD**, **INTE**, ..., **RTS** ima aktivnu vrednost. !

$step_{30}$ *br* (*case* (**INTD**, **INTE**, **TRPD**, **TRPE**,
LDB, **LDW**, **STB**, **STW**, **POPB**, **POPW**, **PUSHB**, **PUSHW**,
LDIVTP, **STIVTP**, **LDSP**, **STSP**,
ADD, **SUB**, **INC**, **DEC**, **AND**,**OR**, **XOR**, **NOT**,
ASR, **LSR**, **ROR**, **RORC**, **ASL**, **LSL**, **ROL**, **ROLC**,
BEQL, **BNEQL**, **BNEG**, **BNNEG**, **BOVF**, **BNOVF**, **BCAR**, **BNCAR**,
BGRT, **BGRTE**, **BLSS**, **BLSSE**, **BGRTU**, **BGRTEU**, **BLSSU**, **BLSSEU**,
JMP, **JSR**, **RTI**, **RTS**)
then
(**INTD**, $step_{31}$), (**INTE**, $step_{32}$),
(**LDB**, $step_{33}$), (**LDW**, $step_{35}$), (**STB**, $step_{36}$), (**STW**, $step_{3B}$),
(**POPB**, $step_{42}$), (**POPW**, $step_{47}$), (**PUSHB**, $step_{4F}$), (**PUSHW**, $step_{53}$),
(**LDIVTP**, $step_{5A}$), (**STIVTP**, $step_{5B}$), (**LDSP**, $step_{5C}$), (**STSP**, $step_{5D}$),
(**ADD**, $step_{5E}$), (**SUB**, ste_{60}), (**INC**, $step_{62}$), (**DEC**, $step_{64}$),
(**AND**, $step_{66}$), (**OR**, $step_{68}$), (**XOR**, $step_{6A}$), (**NOT**, $step_{6C}$),
(**ASR**, $step_{6E}$), (**LSR**, $step_{6E}$), (**ROR**, $step_{6E}$), (**RORC**, $step_{6E}$),
(**ASL**, $step_{70}$), (**LSL**, $step_{70}$), (**ROL**, $step_{70}$), (**ROLC**, $step_{70}$),
(**BEQL**, $step_{72}$), (**BNEQL**, $step_{72}$), (**BNEG**, $step_{72}$), (**BNNEG**, $step_{72}$),
(**BOVF**, $step_{72}$), (**BNOVF**, $step_{72}$), (**BCAR**, $step_{72}$), (**BNCAR**, $step_{72}$),
(**BGRT**, $step_{72}$), (**BGRE**, $step_{72}$), (**BLSS**, $step_{72}$), (**BLSSE**, $step_{72}$),
(**BGRTU**, $step_{72}$), (**BGRTEU**, $step_{72}$), (**BLSSU**, $step_{72}$), (**BLSSEU**, $step_{72}$),
(**JMP**, $step_{74}$), (**JSR**, $step_{75}$), (**RTI**, $step_{7C}$), (**RTS**, $step_{82}$));

! INTD !

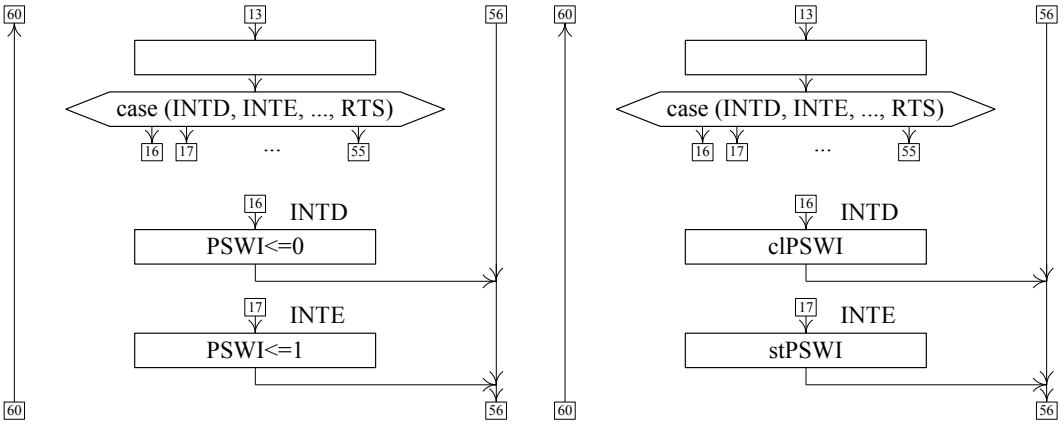
! U korak $step_{31}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **INTD** ima vrednost 1. Vrednošću 1 signala **clPSWI** se razred PSWI bloka *exec* postavlja na vrednost 0. Iz koraka $step_{31}$ se bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{31}$ **clPSWI**,
br $step_{89}$;

! INTE !

! U korak $step_{32}$ se dolazi iz koraka $step_{30}$ ukoliko je signal operacije **INTE** ima vrednost 1. Vrednošću 1 signala **stPSWI** se razred PSWI bloka *exec* postavlja na vrednost 1. Iz koraka $step_{32}$ se bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida.!

$step_{32}$ **stPSWI**,
br $step_{89}$;

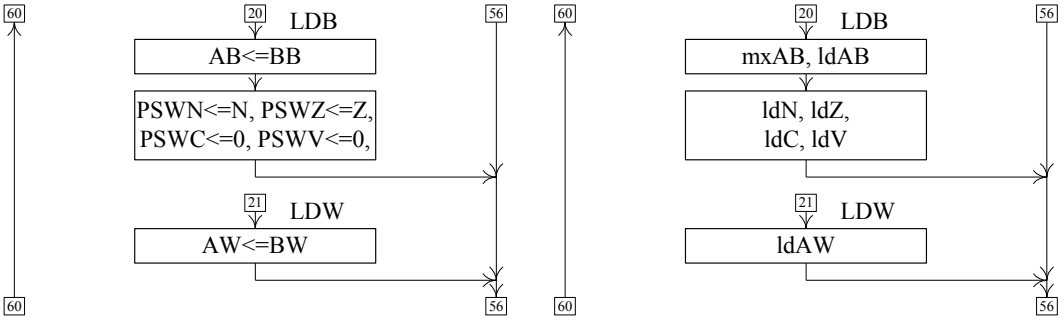


Slika 28 Izvršavanje operacije (prvi deo)

! LDB !

! U korak $step_{33}$ se dolazi iz $step_{30}$ ukoliko signal operacije **LDB** ima vrednost 1. U fazi izvršavanja ove instrukcije se operand specificiran adresnim delom instrukcije prebacuje u registar $AB_{7...0}$. Stoga se vrednostima 1 signala **mxAB** i **ldAB** bloka *exec* sadržaj registra $BB_{7...0}$ propušta kroz multiplekser MX i upisuje u registar $AB_{7...0}$. Potom se u koraku $step_{34}$ vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** bloka *exec* upisuju u razrede PSWN i PSWZ programske statusne reči PSW vrednosti signala N i Z formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i u razrede PSWC i PSWV vrednosti 0 i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{33}$ **mxAB**, **ldAB**;
$step_{34}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{89}$;



Slika 29 Izvršavanje operacije (drugi deo)

! LDW !

! U korak $step_{35}$ se dolazi iz $step_{30}$ ukoliko signal operacije **LDW** ima vrednost 1. U ovom koraku se vrednošću 1 signala **ldAW** bloka *exec* sadržaj registra $BW_{15...0}$ upisuje u registar $AW_{15...0}$ i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{35}$ **ldAW**,
br $step_{89}$;

! STB !

! U korak $step_{36}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **STB** ima vrednost 1. Iz ovog koraka se u zavisnosti od toga da li signal **dirreg** bloka *fetch* ima vrednost 0 ili 1 prelazi na korak $step_{37}$ ili $step_{3A}$. !

$step_{36}$ *br* (*if* **dirreg** *then* $step_{3A}$);

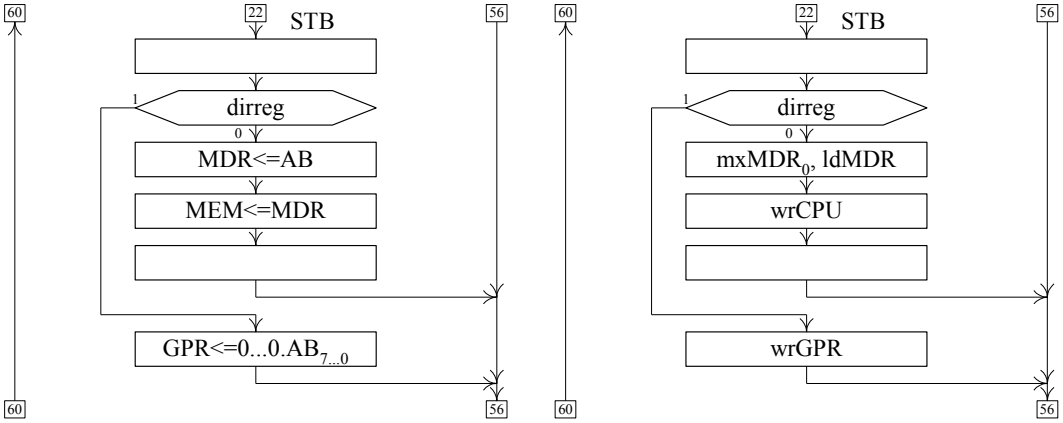
! U koracima $step_{37}$ i $step_{38}$ se sadržaj registra $AB_{7...0}$ bloka *exec* upisuje u memorijsku lokaciju na adresi koja je formirana u fazi formiranje adrese i koja se nalazi i registr $MAR_{15...0}$ bloka *bus*. Stoga se, najpre, u koraku $step_{37}$ vrednostima 1 signalima $\mathbf{mxMDR_0}$ i **ldMDR** bloka *bus* sadržaj registra $AB_{7...0}$ propušta kroz multiplekser MX i upisuje u registar $MDR_{7...0}$. U koraku $step_{38}$ se vrednošću 1 signala **wrCPU** bloka *bus* sadržaji registara $MAR_{15...0}$ i $MDR_{7...0}$ puštaju na adresne linije $ABUS_{15...0}$ i ulazne linije podataka $DIBUS_{7...0}$ magistrale ***BUS*** i formira vrednost 0 signala na upravljačkoj

liniji $\overline{\textbf{WRBUS}}$ magistrale ***BUS***, čime se u memoriji ***MEM*** realizuje operacija upisa i prelazi na korak $step_{39}$. Iz koraka $step_{39}$ se bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{37}$ **mxMDR$_0$, ldMDR**;
$step_{38}$ **wrCPU**;
$step_{39}$ *br* $step_{89}$;

! U koraku $step_{3A}$ se sadržaj registra $AB_{7...0}$ bloka *exec* proširen nulama do dužine 16 bita vrednošću 1 signala **wrGPR** bloka *addr* upisuje u adresirani registar opšte namene $GPR_{15...0}$ i bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{3A}$ **wrGPR**,
br $step_{89}$;



Slika 30 Izvršavanje operacije (treći deo)

! STW !

! U korak $step_{3B}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **STW** ima vrednost 1. Iz ovog koraka se u zavisnosti od toga da li signal **dirreg** bloka *fetch* ima vrednost 0 ili 1 prelazi na korak $step_{3C}$ ili $step_{41}$. !

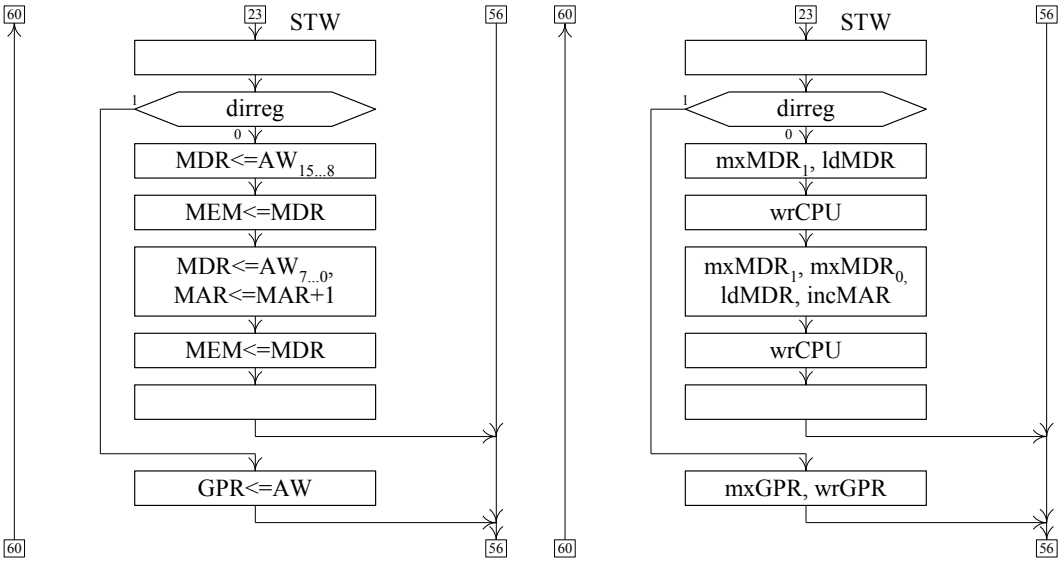
$step_{3B}$ *br* (*if* **dirreg** *then* $step_{41}$);

! U koracima $step_{3C}$ do $step_{40}$ se sadržaj najpre višeg bajta a zatim i nižeg bajta registra $AW_{15...0}$ bloka *exec* upisuje u dve susedne memorijske lokacije počev od memorijske lokacije čija je adresa formirana u fazi formiranje adrese i koja se nalazi i registru $MAR_{15...0}$ bloka *bus*. Stoga se, najpre, u koraku $step_{3C}$ vrednostima 1 signala **mxMDR$_1$** i **ldMDR** bloka *bus* sadržaj registra $AW_{15...8}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{3D}$ na isti načina kao što se to radi u koraku $step_{38}$ instrukcije STB. Potom se u koraku $step_{3E}$ vrednostima 1 signala **mxMDR$_1$**, **mxMDR$_0$** i **ldMDR** bloka *bus* sadržaj registra $AW_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{3F}$ na isti načina kao što se to radi u koraku $step_{3D}$ za prethodni bajt i prelazi na korak $step_{40}$. Iz koraka $step_{40}$ se bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{3C}$ **mxMDR$_1$**, **ldMDR**;
$step_{3D}$ **wrCPU**;
$step_{3E}$ **mxMDR$_1$**, **mxMDR$_0$**, **ldMDR**, **incMAR**;
$step_{3F}$ **wrCPU**;
$step_{40}$ *br* $step_{89}$;

! U koraku $step_{41}$ se vrednostima 1 signala **mxGPR** i **wrGPR** bloka *addr* sadržaj registra $AW_{15...0}$ bloka *exec* propušta kroz multiplekser MX1 i upisuje u adresirani registar opšte namene $GPR_{15...0}$ i bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{41}$ **mxGPR**, **wrGPR**,
br $step_{89}$;

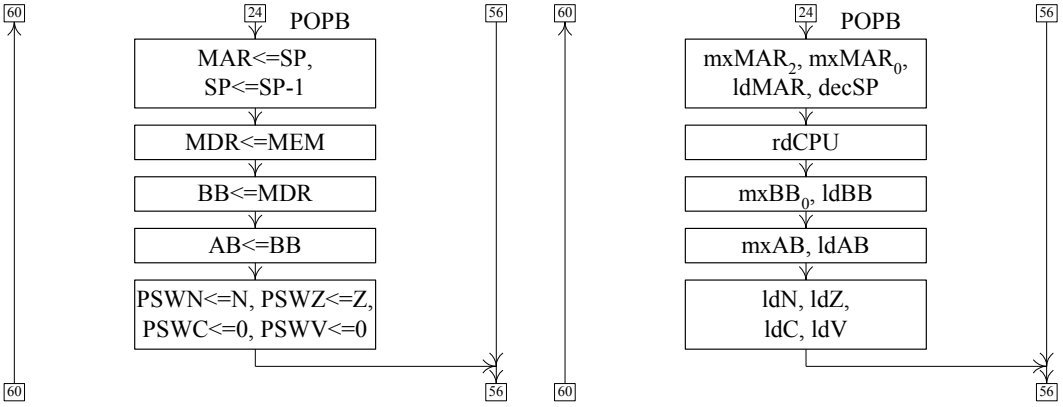


Slika 31 Izvršavanje operacije (četvrti deo)

! POPB !

! U korak $step_{42}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **POPB** ima vrednost 1. U fazi izvršavanja ove instrukcije sa steka se skida bajt podatka i upisuje u registar $AB_{7...0}$ bloka *exec*. Stoga se, najpre, u koraku $step_{42}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR** bloka *bus* sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$ bloka *addr*. Čitanje se realizuje u koraku $step_{43}$ na isti načina kao što se to radi u koraku $step_{02}$ u kome se čita prvi bajt instrukcije. U koraku $step_{44}$ se vrednostima 1 signala **mxBB$_0$** i **ldBB** sadržaj registra $MDR_{7...0}$ bloka *bus* propušta kroz multiplekser MX2 i upisuje u registar $BB_{7...0}$ bloka *exec*. Zatim se u koraku $step_{45}$ vrednostima 1 signala **mxAB** i **ldAB** sadržaj registra $BB_{7...0}$ propušta kroz multiplekser MX1 i upisuje u registar $AB_{7...0}$ bloka *exec*. Na kraju se vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** bloka *exec* upisuju u razrede PSWN i PSWZ programske statusne reči vrednosti signala N i Z formirane na osnovu sadržaja upisanog u registar AB i u razrede PSWC i PSWV vrednosti 0 i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{42}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **decSP**;
$step_{43}$ **rdCPU**;
$step_{44}$ **mxBB$_0$, ldBB**;
$step_{45}$ **mxAB**, **ldAB**;
$step_{46}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{89}$;



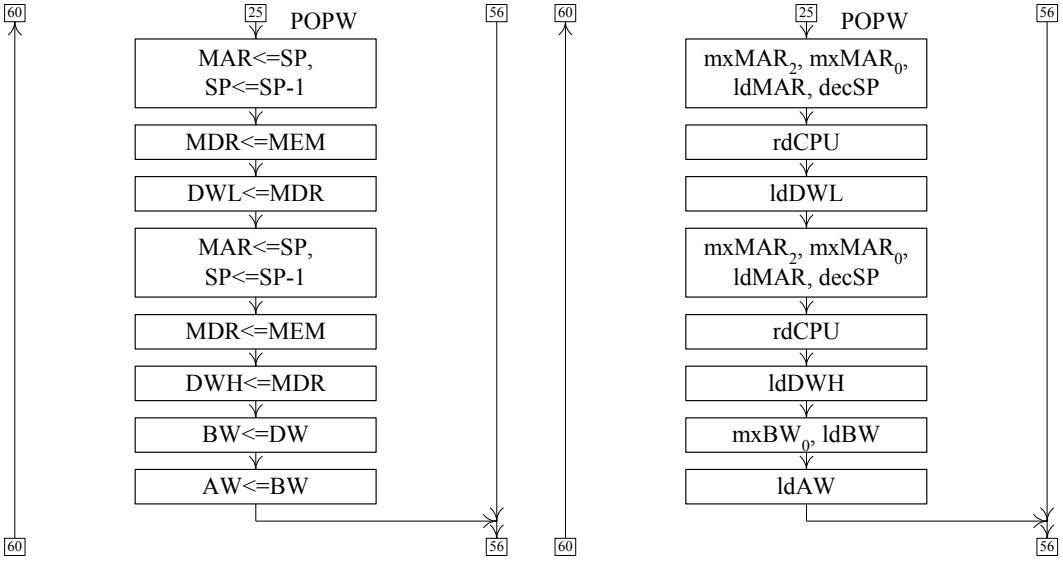
Slika 32 Izvršavanje operacije (peti deo)

! POPW !

! U korak $step_{47}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **POPW** ima vrednost 1. U fazi izvršavanja ove instrukcije sa steka se skidaju dva bajta podatka i upisuju u registar $AW_{15...0}$ bloka

exec. Stoga se, najpre, u koraku $step_{47}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR** bloka *bus* sadržaj registra $SP_{15...0}$ bloka *addr* propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koraku $step_{48}$ na isti načina kao što se to radi u koraku $step_{02}$ u kome se čita prvi bajt instrukcije. U koraku $step_{49}$ se vrednošću 1 signala **ldDWL** sadržaj registra $MDR_{7...0}$ bloka *bus* upisuje u niži bajt registra $DW_{7...0}$ bloka *exec*. Zatim se u koraku $step_{4A}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR** sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koraku $step_{4B}$ na isti načina kao što se to radi u koraku $step_{48}$ u kome se čita prethodni bajt. Potom se u koraku $step_{4C}$ vrednošću 1 signala **ldDWH** sadržaj registra $MDR_{7...0}$ upisuje u viši bajt registra $DW_{15...8}$. Time se u registru $DW_{15...0}$ nalazi 16-to bitna vrednost skinuta sa steka. Na kraju se najpre u koraku $step_{4D}$ vrednostima 1 signala **mxBW$_0$** i **ldBW** sadržaj registra $DW_{15...0}$ propušta kroz multiplekser MX6 i upisuje u registar $BW_{15...0}$ bloka *exec* i zatim u koraku $step_{4E}$ vrednostima 0 signala **mxAW$_1$** i **mxAW$_0$** i vrednošću 1 signala **ldAW** sadržaj registra $BW_{15...0}$ propušta kroz multiplekser MX5 i upisuje u registar $AW_{15...0}$ i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{47}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **decSP**;
$step_{48}$ **rdCPU**;
$step_{49}$ **ldDWL**;
$step_{4A}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **decSP**;
$step_{4B}$ **rdCPU**;
$step_{4C}$ **ldDWH**;
$step_{4D}$ **mxBW$_0$, ldBW**;
$step_{4E}$ **ldAW**,
br $step_{89}$;

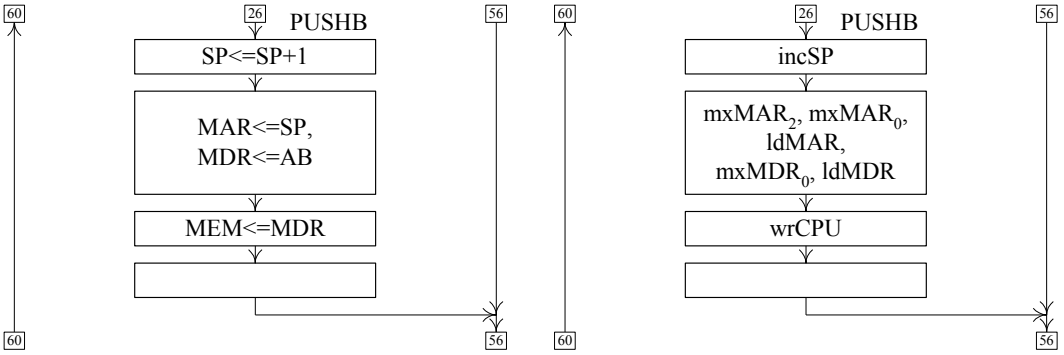


Slika 33 Izvršavanje operacije (šesti deo)

! PUSHB !

! U korak $step_{4F}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **PUSHB** ima vrednost 1. U fazi izvršavanja ove instrukcije se sadržaj registra $AB_{7...0}$ bloka *exec* stavlja na vrh steka. Stoga se najpre u koraku $step_{4F}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$ bloka *addr*. Zatim se u koraku $step_{50}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **mxMDR$_0$** i **ldMDR** sadržaj registra $AB_{7...0}$ propušta kroz multiplekser MX2 bloka *bus* i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{51}$ na isti načina kao što se to radi u koraku $step_{38}$ instrukcije STB i prelazi na korak $step_{52}$. Na kraju se iz koraka $step_{52}$ bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

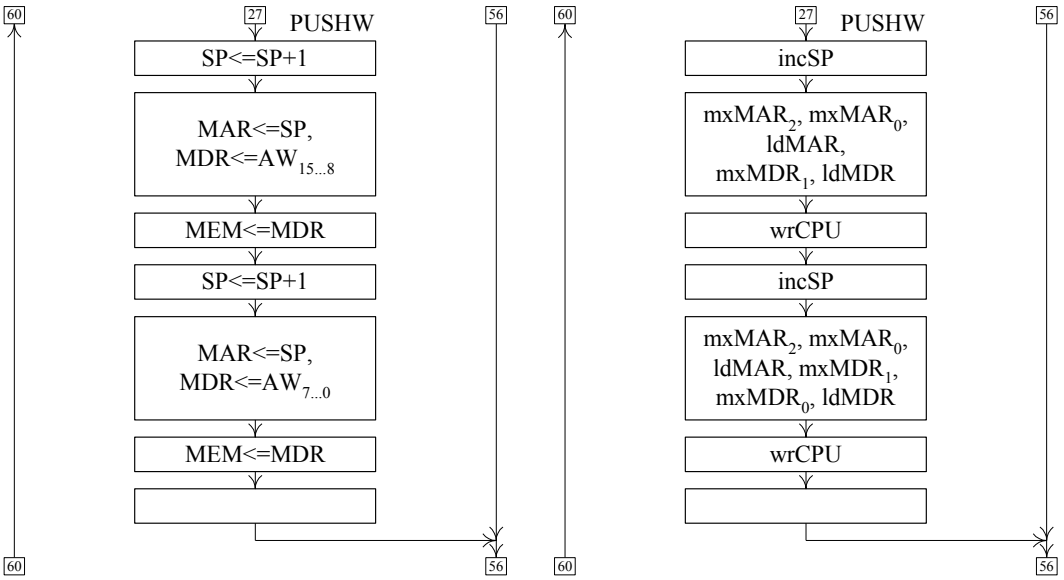
$step_{4F}$ **incSP**;
$step_{50}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **mxMDR$_0$, ldMDR**;
$step_{51}$ **wrCPU**;
$step_{52}$ *br* $step_{89}$;



Slika 34 Izvršavanje operacije (sedmi deo)

! PUSHW !

! U korak $step_{53}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **PUSHW** ima vrednost 1. U fazi izvršavanja ove instrukcije se sadržaj registra $AW_{15...0}$ bloka *exec* stavlja na stek i to prvo viši a zatim i niži bajt. Stoga se u koraku $step_{53}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$ bloka *addr*. Zatim se u koraku $step_{54}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **mxMDR$_1$** i **ldMDR** sadržaj registra $AW_{15...8}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{55}$ na isti načina kao što se to radi u koraku $step_{38}$ instrukcije STB. Ponovo se sada u koraku $step_{56}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{57}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR** sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **mxMDR$_1$**, **mxMDR$_0$** i **ldMDR** sadržaj registra $AW_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{58}$ na isti načina kao što se to radi u koraku $step_{55}$ i prelazi na korak $step_{59}$. Na kraju se iz koraka $step_{59}$ bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !



Slika 35 Izvršavanje operacije (osmi deo)

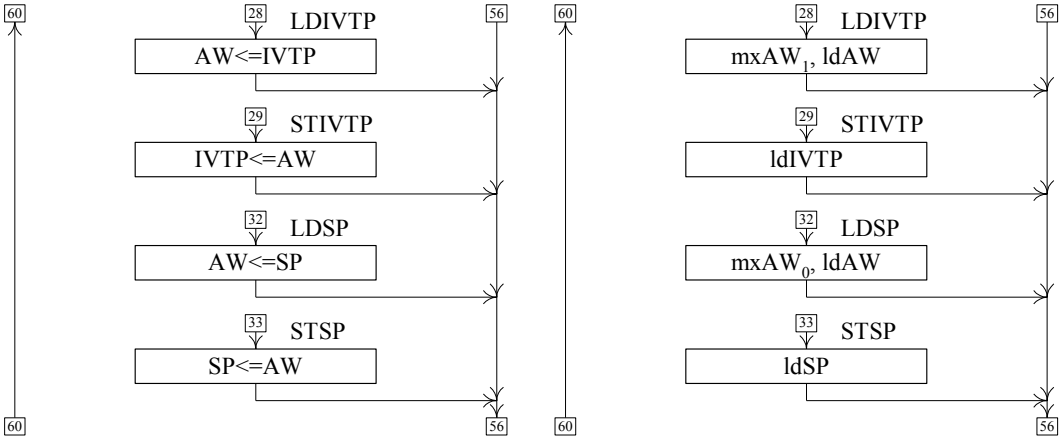
$step_{53}$ **incSP**;
$step_{54}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **mxMDR$_1$, ldMDR**;
$step_{55}$ **wrCPU**;
$step_{56}$ **incSP**;

step$_{57}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **mxMDR$_1$, mxMDR$_0$, ldMDR**;
step$_{58}$ **wrCPU**;
step$_{59}$ *br* step$_{89}$;

! LDIVTP !

! U korak step$_{5A}$ se dolazi iz koraka step$_{30}$ ukoliko signal operacije LDIVTP ima vrednost 1. U fazi izvršavanja ove instrukcije se sadržaj registra IVTP$_{15...0}$ bloka *intr* upisuje u registar AW$_{15...0}$ bloka *exec*. Stoga se vrednostima 1 signala **mxAW$_1$** i **ldAW** sadržaj registra IVTP$_{15...0}$ propušta kroz multiplekser MX5 bloka *exec* i upisuje u registar AW$_{15...0}$. Na kraju se iz koraka step$_{5A}$ bezuslovno prelazi na korak step$_{89}$ i fazu opsluživanje prekida. !

step$_{5A}$ **mxAW$_1$, ldAW**,
br step$_{89}$;



Slika 36 Izvršavanje operacije (deveti deo)

! STIVTP !

! U korak step$_{5B}$ se dolazi iz koraka step$_{30}$ ukoliko signal operacije STIVTP ima vrednost 1. U fazi izvršavanja ove instrukcije se sadržaj registra AW$_{15...0}$ bloka *exec* upisuje u registar IVTP$_{15...0}$ bloka *intr*. Stoga se vrednošću 1 signala **ldIVTP** sadržaj registra AW$_{15...0}$ upisuje u registar IVTP$_{15...0}$. Na kraju se iz koraka step$_{5B}$ bezuslovno prelazi na korak step$_{89}$ i fazu opsluživanje prekida. !

step$_{5B}$ **ldIVTP**,
br step$_{89}$;

! LDSP !

! U korak step$_{5C}$ se dolazi iz koraka step$_{30}$ ukoliko signal operacije LDSP ima vrednost 1. U fazi izvršavanja ove instrukcije se sadržaj registra SP$_{15...0}$ bloka *addr* upisuje u registar AW$_{15...0}$ bloka *exec*. Stoga se vrednostima 1 signala **mxAW$_0$** i **ldAW** sadržaj registra SP$_{15...0}$ propušta kroz multiplekser MX5 bloka *exec* i upisuje u registar AW$_{15...0}$. Na kraju se iz koraka step$_{5C}$ bezuslovno prelazi na korak step$_{89}$ i fazu opsluživanje prekida. !

step$_{5C}$ **mxAW$_0$, ldAW**,
br step$_{89}$;

! STSP !

! U korak step$_{5D}$ se dolazi iz step$_{30}$ ukoliko signal operacije STSP ima vrednost 1. U fazi izvršavanja ove instrukcije se sadržaj registra AW$_{15...0}$ bloka *exec* upisuje u registar SP$_{15...0}$ bloka *addr*. Stoga se vrednošću 1 signala **ldSP** sadržaj registra AW$_{15...0}$ upisuje u registar SP$_{15...0}$. Na kraju se iz koraka step$_{5D}$ bezuslovno prelazi na korak step$_{89}$ i fazu opsluživanje prekida. !

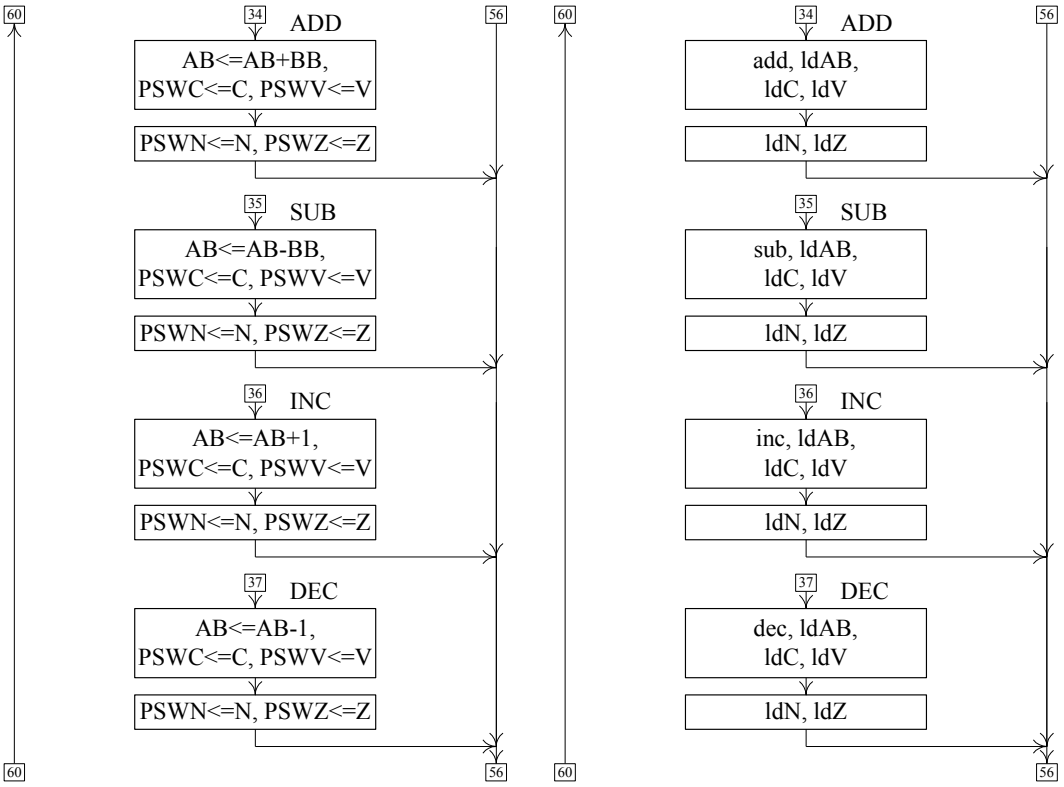
step$_{5D}$ **ldSP**,
br step$_{89}$;

! ADD !

! U korak step$_{5E}$ se dolazi iz koraka step$_{30}$ ukoliko signal operacije **ADD** ima vrednost 1. U fazi izvršavanja ove instrukcije se sabiraju sadržaji registra AB$_{7...0}$ bloka *exec* koji se koristi kao akumulator i registra BB$_{7...0}$ bloka *exec* u kome se nalazi operand specificiran adresnim delom instrukcije i rezultat upisuje u registar AB$_{7...0}$. Stoga se vrednošću 1 signala **add** bloka *exec* na izlazima ALU$_{7...0}$ aritmetičko logičke jedinice ALU formira suma sadržaja registara AB$_{7...0}$ i BB$_{7...0}$

koja se dalje vrednošću 0 signala **mxAB** propušta kroz multiplekser MX1 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$. Istovremeno se vrednostima 1 signala **ldC** i **ldV** bloka *exec* u razrede PSWC i PSWV programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **C** i **V** bloka *exec* formirane na osnovu dobijenog rezultata na izlazima $ALU_{7...0}$ i prelazi na korak $step_{5F}$. U koraku $step_{5F}$ se vrednostima 1 signala **ldN** i **ldZ** bloka *exec* u razrede PSWN i PSWZ programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **N** i **Z** bloka *exec* formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{5E}$ **add**, **ldAB**, **ldC**, **ldV**;
$step_{5F}$ **ldN**, **ldZ**,
br $step_{89}$;



Slika 37 Izvršavanje operacije (deseti deo)

! SUB !

! U korak $step_{60}$ se dolazi iz $step_{30}$ ukoliko signal operacije **SUB** ima vrednost 1. U fazi izvršavanja ove instrukcije se od sadržaja registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator oduzima sadržaj registra $BB_{7...0}$ bloka *exec* u kome se nalazi operand specificiran adresnim delom instrukcije i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **sub** bloka *exec* na izlazima $ALU_{7...0}$ formira razlika sadržaja registara $AB_{7...0}$ i $BB_{7...0}$ koja se dalje vrednošću 0 signala **mxAB** propušta kroz multiplekser MX1 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$. Istovremeno se vrednostima 1 signala **ldC** i **ldV** bloka *exec* u razrede PSWC i PSWV programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **C** i **V** bloka *exec* formirane na osnovu dobijenog rezultata na izlazima $ALU_{7...0}$ i prelazi na korak $step_{61}$. U koraku $step_{61}$ se vrednostima 1 signala **ldN** i **ldZ** bloka *exec* u razrede PSWN i PSWZ programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **N** i **Z** bloka *exec* formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{60}$ **sub**, **ldAB**, **ldC**, **ldV**;
$step_{61}$ **ldN**, **ldZ**,
br $step_{89}$;

! INC !

! U korak $step_{62}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **INC** ima vrednost 1. U fazi izvršavanja ove instrukcije se sabira sadržaj registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator i

vrednost 1 i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **inc** bloka *exec* na izlazima $ALU_{7...0}$ formira suma sadržaja registra $AB_{7...0}$ i vrednosti 1 koja se dalje vrednošću 0 signala **mxAB** propušta kroz multiplekser MX1 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$. Istovremeno se vrednostima 1 signala **ldC** i **ldV** bloka *exec* u razrede PSWC i PSWV programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **C** i **V** bloka *exec* formirane na osnovu dobijenog rezultata na izlazima $ALU_{7...0}$ i prelazi na korak $step_{63}$. U koraku $step_{63}$ se vrednostima 1 signala **ldN** i **ldZ** bloka *exec* u razrede PSWN i PSWZ programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **N** i **Z** bloka *exec* formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{62}$ **inc**, **ldAB**, **ldC**, **ldV**;
$step_{63}$ **ldN**, **ldZ**,
br $step_{89}$;

! DEC !

! U korak $step_{64}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **DEC** ima vrednost 1. U fazi izvršavanja ove instrukcije se od sadržaj registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator i oduzima vrednost 1 i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **dec** bloka *exec* na izlazima $ALU_{7...0}$ formira razlika sadržaja registra $AB_{7...0}$ i vrednosti 1 koja se dalje vrednošću 0 signala **mxAB** propušta kroz multiplekser MX1 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$. Istovremeno se vrednostima 1 signala **ldC** i **ldV** bloka *exec* u razrede PSWC i PSWV programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **C** i **V** bloka *exec* formirane na osnovu dobijenog rezultata na izlazima $ALU_{7...0}$ i prelazi na korak $step_{65}$. U koraku $step_{65}$ se vrednostima 1 signala **ldN** i **ldZ** bloka *exec* u razrede PSWN i PSWZ programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **N** i **Z** bloka *exec* formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{64}$ **dec**, **ldAB**, **ldC**, **ldV**;
$step_{65}$ **ldN**, **ldZ**,
br $step_{89}$;

! AND !

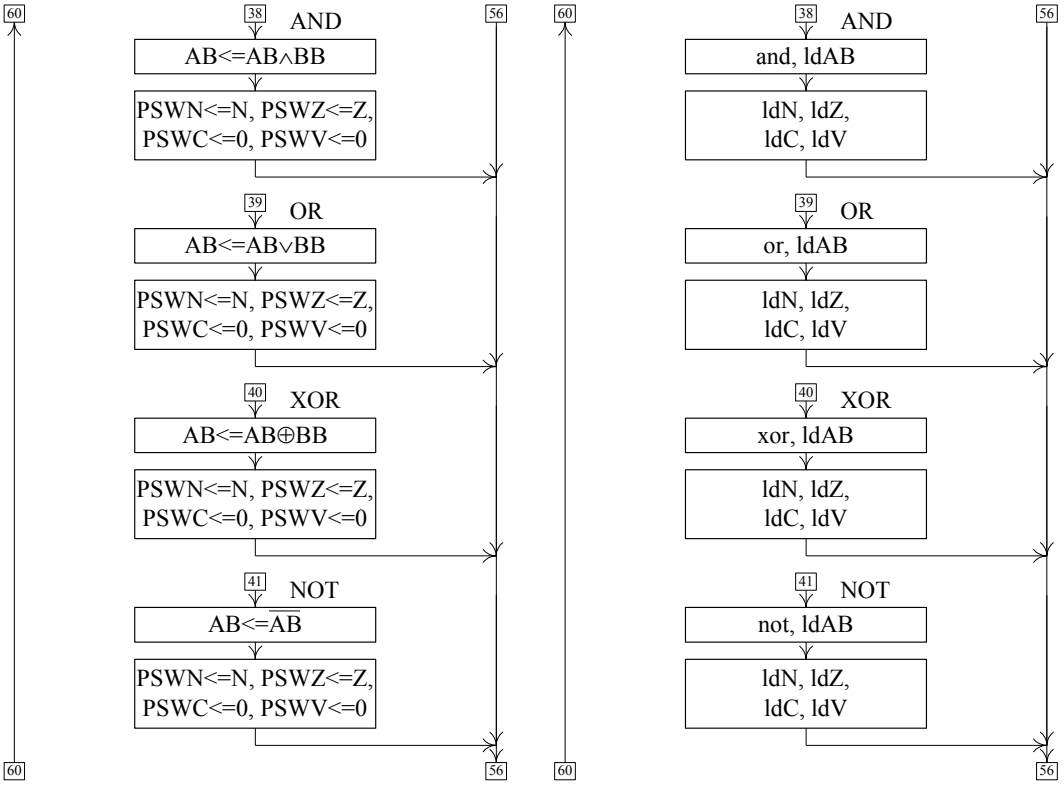
! U korak $step_{66}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **AND** ima vrednost 1. U fazi izvršavanja ove instrukcije se nad sadržajima registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator i registra $BB_{7...0}$ bloka *exec* u kome se nalazi operand specificiran adresnim delom instrukcije realizuje logička I operacija i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **and** bloka *exec* na izlazima $ALU_{7...0}$ aritmetičko logičke jedinice ALU formira rezultat logičke I operacije sadržaja registara $AB_{7...0}$ i $BB_{7...0}$ koji se dalje vrednošću 0 signala **mxAB** propušta kroz multiplekser MX1 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$ i prelazi na korak $step_{67}$. U koraku $step_{67}$ se vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** bloka *exec* u razrede PSWN i PSWZ programske statusne $PSW_{15...0}$ reči upisuju vrednosti signala **N** i **Z** formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i u razrede PSWC i PSWV vrednosti 0 i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{66}$ **and**, **ldAB**;
$step_{67}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{89}$;

! OR !

! U korak $step_{68}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **OR** ima vrednost 1. U fazi izvršavanja ove instrukcije se nad sadržajima registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator i registra $BB_{7...0}$ bloka *exec* u kome se nalazi operand specificiran adresnim delom instrukcije realizuje logička ILI operacija i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **or** bloka *exec* na izlazima $ALU_{7...0}$ formira rezultat logičke ILI operacije sadržaja registara $AB_{7...0}$ i $BB_{7...0}$ koji se dalje vrednošću 0 signala **mxAB** propušta kroz multiplekser MX1 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$ i prelazi na korak $step_{69}$. U koraku $step_{69}$ vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** bloka *exec* u razrede PSWN i PSWZ programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **N** i **Z** formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i u razrede PSWC i PSWV vrednosti 0 i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{68}$ **or**, **ldAB**;
$step_{69}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{89}$;



Slika 38 Izvršavanje operacije (jedanaesti deo)

! XOR !

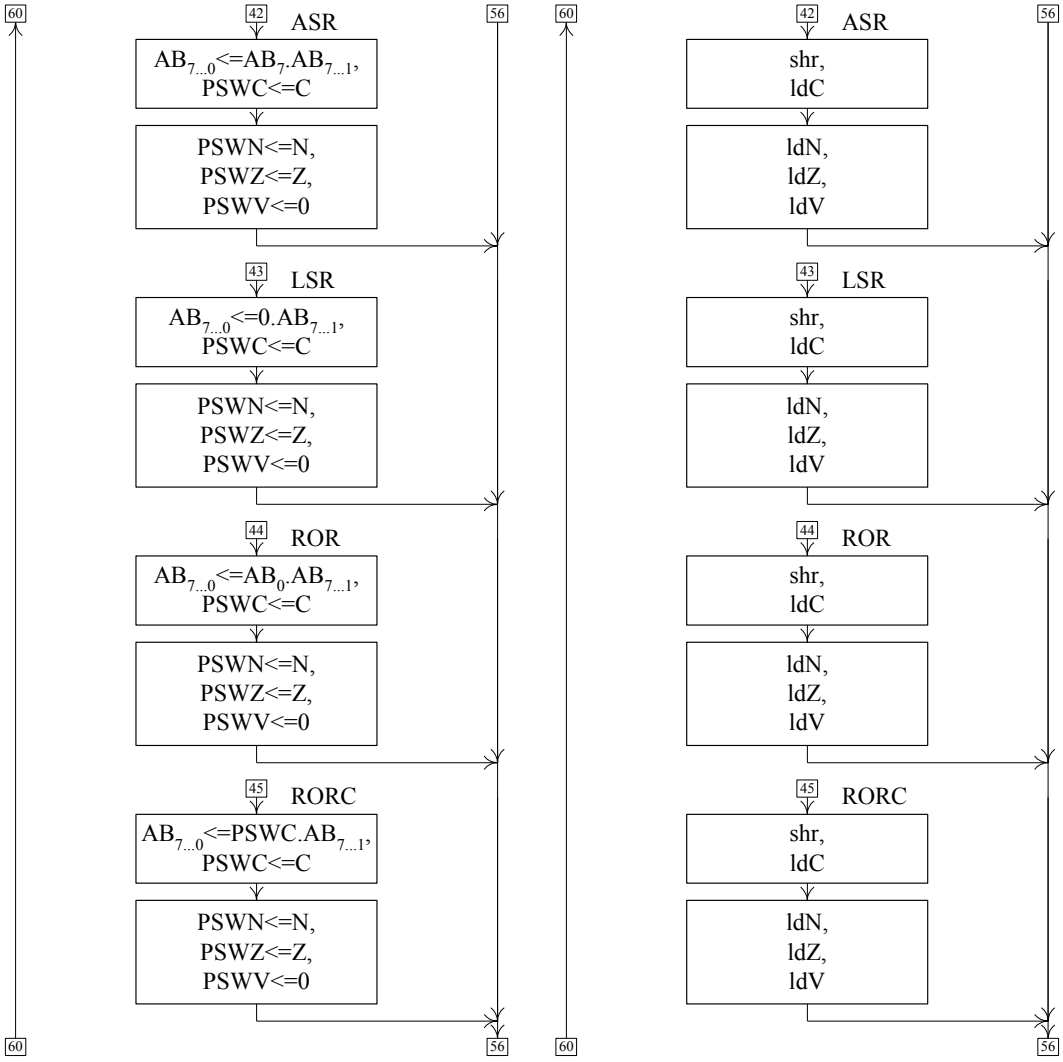
! U korak $step_{6A}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **XOR** ima vrednost 1. U fazi izvršavanja ove instrukcije se nad sadržajima registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator i registra $AB_{7...0}$ bloka *exec* u kome se nalazi operand specificiran adresnim delom instrukcije realizuje logička ekskluzivno ILI operacija i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **xor** bloka *exec* na izlazima $ALU_{7...0}$ formira rezultat logičke ekskluzivno ILI operacije sadržaja registara $AB_{7...0}$ i $BB_{7...0}$ koji se dalje vrednošću 0 signala **mxAB** propušta kroz multiplekser MX1 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$ i prelazi na korak $step_{6B}$. U koraku $step_{6B}$ se vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** bloka *exec* u razrede PSWN i PSWZ programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **N** i **Z** formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i u razrede PSWC i PSWV vrednosti 0 i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{6A}$ **xor**, **ldAB**;
$step_{6B}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{89}$;

! NOT !

! U korak $step_{6C}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **NOT** ima vrednost 1. U fazi izvršavanja ove instrukcije se invertuju bitovi registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **not** bloka *exec* na izlazima $ALU_{7...0}$ formiraju invertovane vrednosti bitova registra $AB_{7...0}$ koje se dalje vrednošću 0 signala **mxAB** propuštaju kroz multiplekser MX1 i vrednošću 1 signala **ldAB** upisuju u registar $AB_{7...0}$ i prelazi na korak $step_{6D}$. U koraku $step_{6D}$ se vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** bloka *exec* u razrede PSWN i PSWZ programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **N** i **Z** formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i u razrede PSWC i PSWV vrednosti 0 i prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{6C}$ **not**, **ldAB**;
$step_{6D}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{89}$**;**



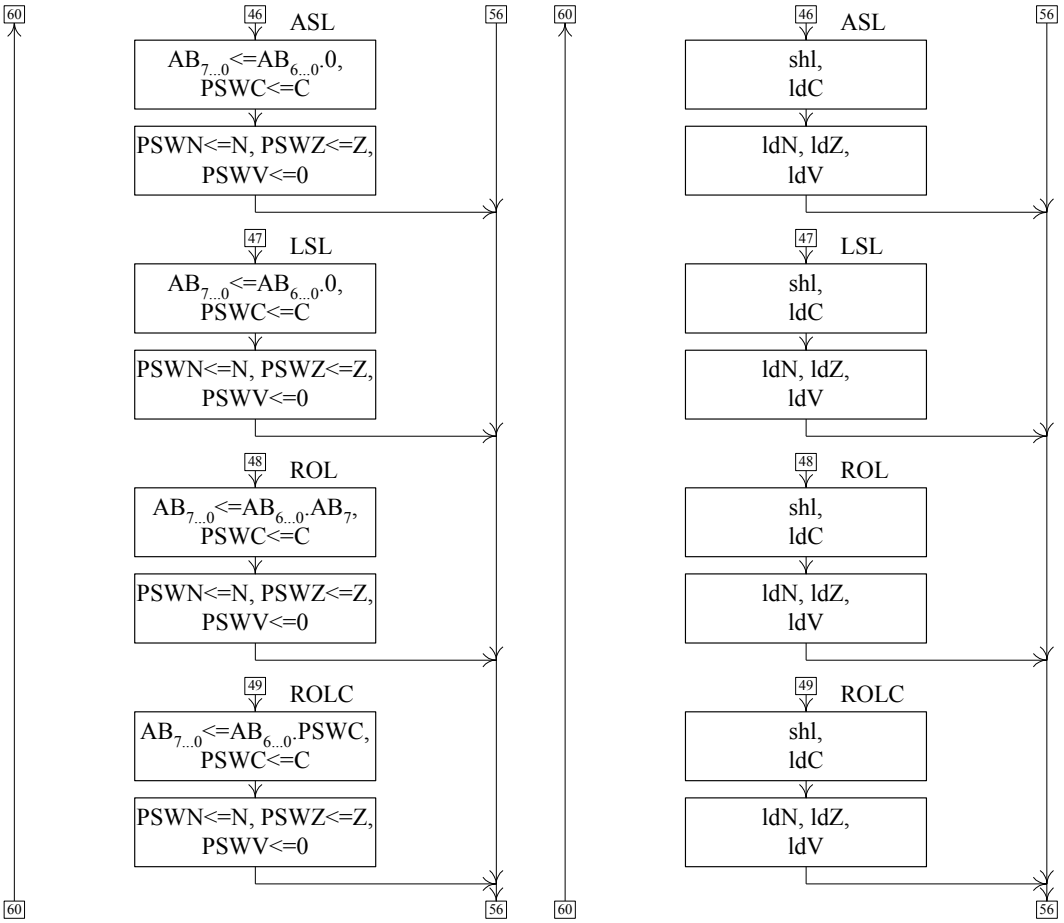
Slika 39 Izvršavanje operacije (dvanaesti deo)

! ASR, LSR, ROR i ROLC !

! U korak $step_{6E}$ se dolazi iz koraka $step_{30}$ ukoliko neki od signala operacija ASR, LSR, ROR i ROLC ima vrednost 1. U fazi izvršavanja ove instrukcije se bitovi registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator aritmetički pomeraju za jedno mesto udesno ukoliko signal **ASR** ima vrednost 1, logički pomeraju za jedno mesto udesno ukoliko signal **LSR** ima vrednost 1, rotiraju za jedno mesto udesno ukoliko signal **ROR** ima vrednost 1 i rotiraju zajedno sa razredom PSWC programske statusne reči $PSW_{15...0}$ za jedno mesto udesno ukoliko signal **RORC** ima vrednost 1. Stoga se vrednošću 1 signala **shr** bloka *exec* bitovi registra $AB_{7...0}$ pomeraju udesno za jedno mesto, pri čemu se u u razred AB_7 upisuje signal **IR** sa izlaza multipleksera MX3 bloka *exec*. U slučaju aritmetičkog pomeranja za jedno mesto udesno to je signal $\mathbf{AB_7}$. Ovaj signal se selektuje na izlazu IR multipleksera MX3 binarnom vrednošću 0 signala selekcije multipleksera MX3 koji se dobijaju na izlazu kodera CD1 zbog vrednosti 1 signala **ASR** na ulazu 0 kodera CD1. U slučaju logičkog pomeranja za jedno mesto udesno to je signal **0**. Ovaj signal se selektuje na izlazu IR multipleksera MX3 binarnom vrednošću 1 signala selekcije multipleksera MX3 koji se dobijaju na izlazu kodera CD1 zbog vrednosti 1 signala **LSR** na ulazu 1 kodera CD1. U slučaju rotiranja za jedno mesto udesno to je signal $\mathbf{AB_0}$. Ovaj signal se selektuje na izlazu IR multipleksera MX3 binarnom vrednošću 2 signala selekcije multipleksera MX3 koji se dobijaju na izlazu kodera CD1 zbog vrednosti 1 signala **ROR** na ulazu 2 kodera CD1. U slučaju rotiranja zajedno sa razredom PSWC programske statusne reči $PSW_{15...0}$ za jedno mesto

udesno to je signal **PSWC**. Ovaj signal se selektuje na izlazu IR multipleksera MX3 binarnom vrednošću 3 signala selekcije multipleksera MX3 koji se dobijaju na izlazu kodera CD1 zbog vrednosti 1 signala **RORC** na ulazu 3 kodera CD1. Istovremeno se vrednošću 1 signala **ldC** bloka *exec* u razred PSWC programske statusne reči $PSW_{15...0}$ upisuje vrednost signala **C** bloka *exec*. U slučaju svih pomeranja i rotiranja za jedno mesto udesno to je signal $\mathbf{AB_0}$. Iz koraka $step_{6E}$ se prelazi na korak $step_{6F}$. U koraku $step_{6F}$ se vrednostima 1 signala **ldN** i **ldZ** bloka *exec* u razrede PSWN i PSWZ programske statusne $PSW_{15...0}$ reči upisuju vrednosti signala **N** i **Z** bloka *exec* formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i vrednošću 1 signala **ldV** bloka *exec* u razred PSWV programske statusne reči $PSW_{15...0}$ upisuje vrednost 0 signala **V** i bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{6E}$ **shr**, **ldC**;
$step_{6F}$ **ldN**, **ldZ**, **ldV**,
br $step_{89}$;



Slika 40 Izvršavanje operacije (trinaesti deo)

! ASL, LSL, ROL i ROLC !

! U korak $step_{70}$ se dolazi iz koraka $step_{30}$ ukoliko neki od signala operacija ASL, LSL, ROL i ROLC ima vrednost 1. U fazi izvršavanja ove instrukcije se bitovi registra $AB_{7...0}$ bloka *exec* koji se koristi kao akumulator aritmetički pomeraju za jedno mesto ulevo ukoliko signal **ASL** ima vrednost 1, logički pomeraju za jedno mesto ulevo ukoliko signal **LSL** ima vrednost 1, rotiraju za jedno mesto ulevo ukoliko signal **ROL** ima vrednost 1 i rotiraju zajedno sa razredom PSWC programske statusne reči $PSW_{15...0}$ za jedno mesto ulevo ukoliko signal **ROLC** ima vrednost 1. Stoga se vrednošću 1 signala **shl** bloka *exec* bitovi registra $AB_{7...0}$ pomeraju ulevo za jedno mesto, pri čemu se u u razred AB_0 upisuje signal **IL** sa izlaza multipleksera MX4 bloka *exec*. U slučaju aritmetičkog pomeranja za jedno mesto ulevo to je signal **0**. Ovaj signal se selektuje na izlazu IL multipleksera MX4 binarnom vrednošću 0 signala selekcije multipleksera MX3 koji se dobijaju na izlazu kodera CD2 zbog vrednosti 1 signala **ASL** na ulazu 0 kodera CD2. U slučaju logičkog pomeranja za jedno mesto ulevo

to je signal **0**. Ovaj signal se selektuje na izlazu IL multipleksera MX4 binarnom vrednošću 1 signala selekcije multipleksera MX4 koji se dobijaju na izlazu kodera CD2 zbog vrednosti 1 signala **LSL** na ulazu 1 kodera CD2. U slučaju rotiranja za jedno mesto ulevo to je signal $\mathbf{AB_7}$. Ovaj signal se selektuje na izlazu IL multipleksera MX4 binarnom vrednošću 2 signala selekcije multipleksera MX4 koji se dobijaju na izlazu kodera CD2 zbog vrednosti 1 signala **ROL** na ulazu 2 kodera CD2. U slučaju rotiranja zajedno sa razredom PSWC programske statusne reči $PSW_{15...0}$ za jedno mesto ulevo to je signal **PSWC**. Ovaj signal se selektuje na izlazu IL multipleksera MX4 binarnom vrednošću 3 signala selekcije multipleksera MX4 koji se dobijaju na izlazu kodera CD2 zbog vrednosti 1 signala **ROLC** na ulazu 3 kodera CD2. Istovremeno se vrednošću 1 signala **ldC** bloka *exec* u razred PSWC programske statusne reči $PSW_{15...0}$ upisuje vrednost signala **C** bloka *exec*. U slučaju svih pomeranja i rotiranja za jedno mesto ulevo to je signal $\mathbf{AB_7}$. Iz koraka $step_{70}$ se prelazi na korak $step_{71}$. U koraku $step_{71}$ se vrednostima 1 signala **ldN** i **ldZ** bloka *exec* u razrede PSWN i PSWZ programske statusne $PSW_{15...0}$ reči upisuju vrednosti signala **N** i **Z** bloka *exec* formirane na osnovu sadržaja upisanog u registar $AB_{7...0}$ i vrednošću 1 signala **ldV** bloka *exec* u razred PSWV programske statusne reči $PSW_{15...0}$ upisuje vrednost 0 signala **V** i bezuslovano prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{70}$ **shl**, **ldC**;
$step_{71}$ **ldN**, **ldZ**, **ldV**,
br $step_{89}$;

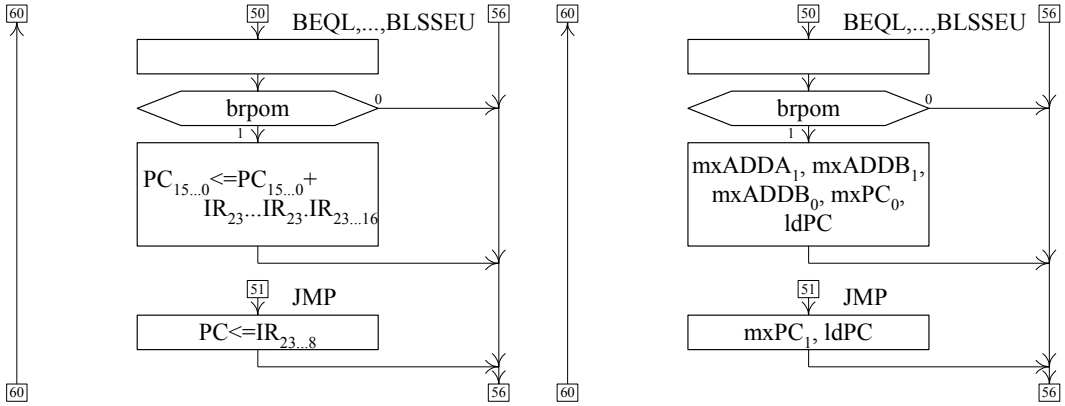
! BEQL,…, BLSSEU !

! U korak $step_{72}$ se dolazi iz koraka $step_{30}$ ukoliko neki od signala operacija uslovnog skoka **BEQL**, **BNEQL**, **BNEG**, **BNNEG**, **BOVF**, **BNOVF**, **BCAR**, **BNCAR**, **BGRT**, **BGRTE**, **BLSS**, **BLSSE**, **BGRTU**, **BGRTEU**, **BLSSU**, **BLSSEU** ima vrednost 1 i prelazi na $step_{89}$ i fazu opsluživanje prekida ukoliko signal **brpom** bloka *exec* ima vrednost 0 i na sledeći korak $step_{73}$ ukoliko ima vrednost 1. Signal **brpom** ima vrednost 1 ukoliko vrednost 1 ima signal rezultata operacije **eql**, **neql**, **neg**, **nneg**, **ovf**, **novf**, **car**, **ncar**, **grt**, **grte**, **lss**, **lsse**, **grtu**, **grteu**, **lssu**, **lsseu** određen vrednošću 1 signala operacije uslovnog skoka **BEQL**, **BNEQL**, **BNEG**, **BNNEG**, **BOVF**, **BNOVF**, **BCAR**, **BNCAR**, **BGRT**, **BGRTE**, **BLSS**, **BLSSE**, **BGRTU**, **BGRTEU**, **BLSSU**, **BLSSEU**. !

$step_{72}$ *br* (*if* $\overline{\textbf{brpom}}$ *then* $step_{C0}$);

! U korak $step_{73}$ se dolazi iz koraka $step_{72}$ ukoliko signal **brpom** ima vrednost 1, čime je uslov za skok ispunjen. U ovom koraku se vrednostima 1 signala $\mathbf{mxADDA_1}$, $\mathbf{mxADDB_1}$ i $\mathbf{mxADDB_0}$ kroz multipleksere MX2 i MX3 bloka *addr* na ulaze sabirača ADD propuštaju sadržaji registra $PC_{15...0}$ bloka *fetch* i registra $IR_{23...16}$ bloka *fetch* proširen znakom na 16 bita. Signali $ADD_{15...0}$ sa izlaza sabirača ADD koji predstavljaju adresu skoka se vrednostima 1 signala $\mathbf{mxPC_0}$ i **ldPC** propuštaju kroz multiplekser MX bloka *fetch* i upisuju u registar $PC_{15...0}$. Pored toga iz koraka $step_{73}$ se bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{73}$ $\mathbf{mxADDA_1}$, $\mathbf{mxADDB_1}$, $\mathbf{mxADDB_0}$, $\mathbf{mxPC_0}$, **ldPC**,
br $step_{89}$;



Slika 41 Izvršavanje operacije (četrnaesti deo)

! JMP !

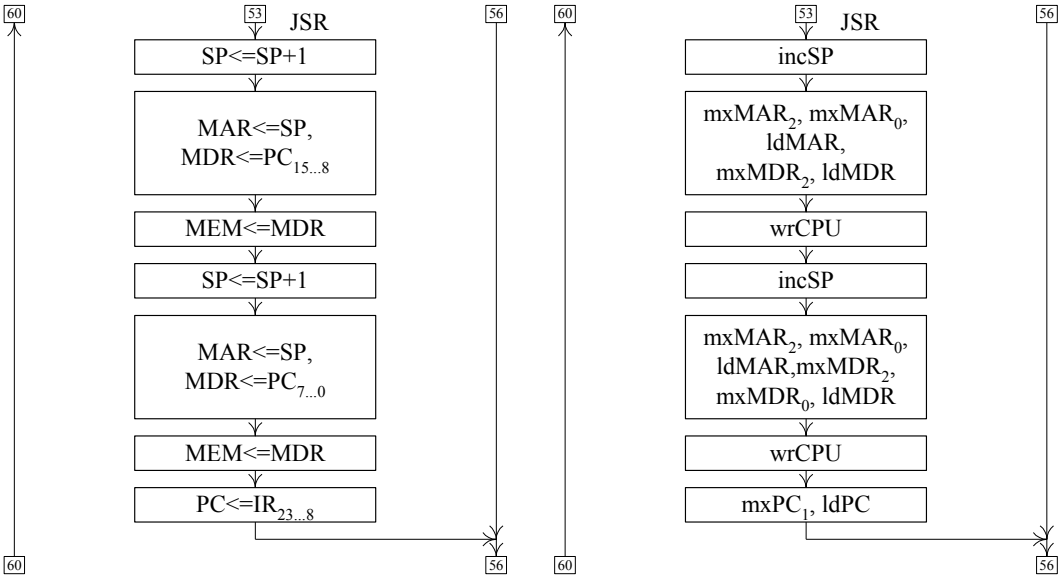
! U korak $step_{74}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **JMP** ima vrednost 1. U fazi izvršavanja ove instrukcije se realizuje bezuslovni skok na adresu koja je data u samoj instrukciji. Stoga se sadržaj registra $IR_{23...8}$ bloka *fetch* koji predstavlja adresu skoka vrednostima 1 signala **mxPC$_1$** i **ldPC** propušta kroz multiplekser MX bloka *fetch* i upisuje u registar $PC_{15...0}$. Pored toga iz koraka $step_{74}$ se bezuslovno prelazi na korak $step_{89}$ i fazu opsluživanje prekida. !

$step_{74}$ **mxPC$_1$, ldPC**,
br $step_{89}$;

! JSR !

! U korak $step_{75}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **JSR** ima vrednost 1. U fazi izvršavanja ove instrukcije se realizuje skok na potprogram tako što se prvo na stek stavi tekući sadržaj programskog brojača i zatim realizuje bezuslovni skok na adresu koja je data u samoj instrukciji. Na stek se stavlja prvo viši a zatim i niži bajt registra $PC_{15...0}$. Stoga se najpre u koraku $step_{75}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$ bloka *addr*. Zatim se u koraku $step_{76}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **mxMDR$_2$** i **ldMDR** sadržaj višeg bajta registra $PC_{15...8}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{77}$ na isti načina kao što se to radi u koraku $step_{38}$ instrukcije STB. Potom se u koraku $step_{78}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{79}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **mxMDR$_2$**, **mxMDR$_0$**, **ldMDR** sadržaj nižeg bajta registra $PC_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{7A}$ na isti načina kao što se to radi u koraku $step_{77}$ prilikom upisa višeg bajta. Na kraju se u koraku $step_{7B}$ sadržaj registra $IR_{23...8}$ bloka *fetch* koji predstavlja adresu skoka vrednostima 1 signala **mxPC$_1$** i **ldPC** propušta kroz multiplekser MX bloka *fetch* i upisuje u registar $PC_{15...0}$ i bezuslovno prelazi na $step_{89}$ i fazu opsluživanje prekida. !

$step_{75}$ **incSP**;
$step_{76}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **mxMDR$_2$, ldMDR**;
$step_{77}$ **wrCPU**;
$step_{78}$ **incSP**;
$step_{79}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **mxMDR$_2$, mxMDR$_0$, ldMDR**;
$step_{7A}$ **wrCPU**,
$step_{7B}$ **mxPC$_1$, ldPC**,
br $step_{89}$;

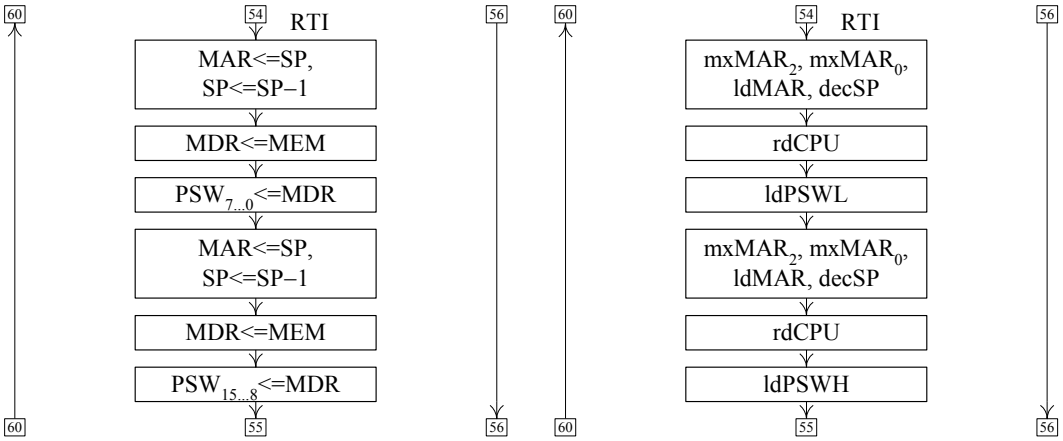


Slika 42 Izvršavanje operacije (petnaesti deo)

! RTI !

! U korak $step_{7C}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **RTI** ima vrednost 1. U fazi izvršavanja ove instrukcije vrednostima sa steka se restauriraju programska statusna reč $PSW_{15..0}$ i programski brojač $PC_{15...0}$. U koracima $step_{7C}$ do $step_{81}$ se najpre vrednošću sa steka restaurira programska statusna reč $PSW_{15...0}$ bloka *exec*. Sa steka se najpre skida niži a zatim i viši bajt registra $PSW_{15...0}$. Stoga se u koraku $step_{7C}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR** bloka *bus* sadržaj registra $SP_{15...0}$ bloka *addr* propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koraku $step_{7D}$ na isti načina kao što se to radi u koraku $step_{02}$ u kome se čita prvi bajt instrukcije. Na kraju se u koraku $step_{7E}$ vrednošću 1 signala **ldPSWL** sadržaj registra $MDR_{7...0}$ bloka *bus* upisuje u niži bajt registra $PSW_{7...0}$ bloka *exec*. Potom se u koraku $step_{7F}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR** sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koraku $step_{80}$ na isti načina kao što se to radi u koraku $step_{7D}$ u kome se čita niži bajt. Na kraju se u koraku $step_{81}$ vrednošću 1 signala **ldPSWH** sadržaj registra $MDR_{7...0}$ bloka *bus* upisuje u viši bajt registra $PSW_{15...8}$. Time je 16-to bitna vrednost skinuta sa steka i smeštena u registar $PSW_{15...0}$, pa se prelazi na korak $step_{82}$ počev od koga se kao i u slučaju instrukcije RTS sa steka skida 16-to bitna vrednost i smešta u registar $PC_{15...0}$. !

$step_{7C}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, decSP**;
$step_{7D}$ **rdCPU**;
$step_{7E}$ **ldPSWL**;
$step_{7F}$ **mxMAR$_2$, mxMAR$_0$, ldMAR, decSP**;
$step_{80}$ **rdCPU**;
$step_{81}$ **ldPSWH**;



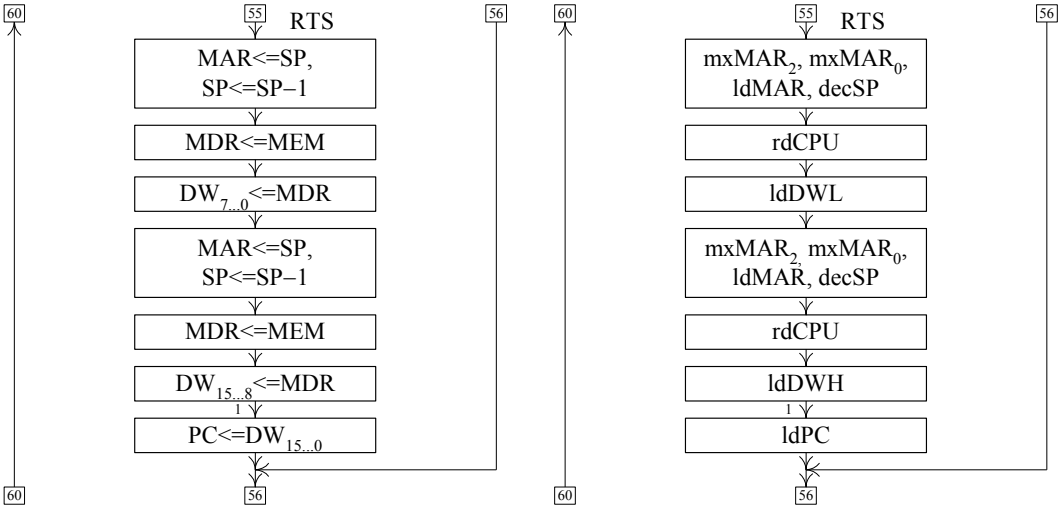
Slika 43 Izvršavanje operacije (šesnaesti deo)

! RTS !

! U korak $step_{82}$ se dolazi iz koraka $step_{30}$ ukoliko signal operacije **RTS** ima vrednost 1. Pored toga u korak $step_{82}$ se dolazi i iz koraka $step_{81}$ kada signal operacije **RTI** ima vrednost 1. U oba slučaja se u koracima $step_{82}$ do $step_{88}$ vrednošću sa steka restaurira programski brojač $PC_{15...0}$. Sa steka se najpre skida niži a zatim i viši bajt registra $PC_{15...0}$. Stoga se u koraku $step_{82}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR** bloka *bus* sadržaj registra $SP_{15...0}$ bloka *addr* propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koraku $step_{83}$ na isti načina kao što se to radi u koraku $step_{02}$ u kome se čita prvi bajt instrukcije. Na kraju se u koraku $step_{84}$ vrednošću 1 signala **ldDWL** sadržaj registra $MDR_{7...0}$ bloka *bus* upisuje u niži bajt registra $DW_{7...0}$ bloka *bus*. Potom se u koraku $step_{85}$ vrednostima 1 signala **mxMAR$_2$**, **mxMAR$_0$** i **ldMAR** sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koraku $step_{86}$ na isti načina kao što se to radi u koraku $step_{83}$ u kome se čita niži bajt. Na kraju se u koraku $step_{87}$ vrednošću 1 signala **ldDWH** sadržaj registra $MDR_{7...0}$ upisuje u viši bajt registar $DW_{15...8}$. Time je 16-to bitna vrednost skinuta sa

steka i smeštena u registar $DW_{15...0}$. Konačno se u koraku $step_{88}$ sadržaj registra $DW_{15...0}$ propušta kroz multiplekser MX bloka *fetch* i vrednošću 1 signala **ldPC** upisuje u registar $PC_{15...0}$ i bezuslovno prelazi na $step_{89}$ i fazu opsluživanje prekida. !

$step_{82}$ **$mxMAR_2$, $mxMAR_0$, ldMAR**, **decSP**;
$step_{83}$ **rdCPU**;
$step_{84}$ **ldDWL**;
$step_{85}$ **$mxMAR_2$, $mxMAR_0$, ldMAR**, **decSP**;
$step_{86}$ **rdCPU**;
$step_{87}$ **ldDWH**;
$step_{88}$ **ldPC**;



Slika 44 Izvršavanje operacije (sedamnaesti deo)

! Opsluživanje prekida !

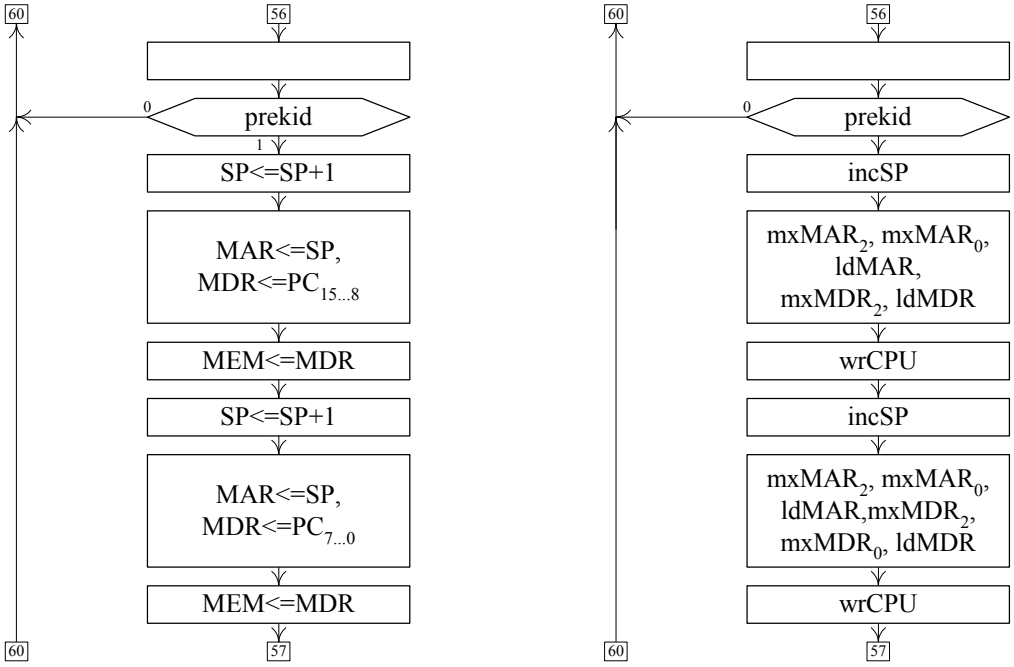
! U korak $step_{89}$ se dolazi iz koraka $step_{31}$, $step_{32}$, ..., $step_{88}$ na završetku faze izvršavanje instrukcije. U koraku $step_{89}$ se, u zavisnosti od toga da li signal **prekid** bloka *intr* ima vrednost 0 ili 1, ili završava izvršavanje tekuće instrukcije i prelaskom na korak $step_{00}$ započinje faza čitanje instrukcije sledeće instrukcije ili se produžava izvršavanje tekuće instrukcije i prelaskom na korak $step_{8A}$ produžava faza opsluživanje prekida tekuće instrukcije. !

$step_{89}$ br (*if* $\overline{\textbf{prekid}}$ *then* $step_{00}$);

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

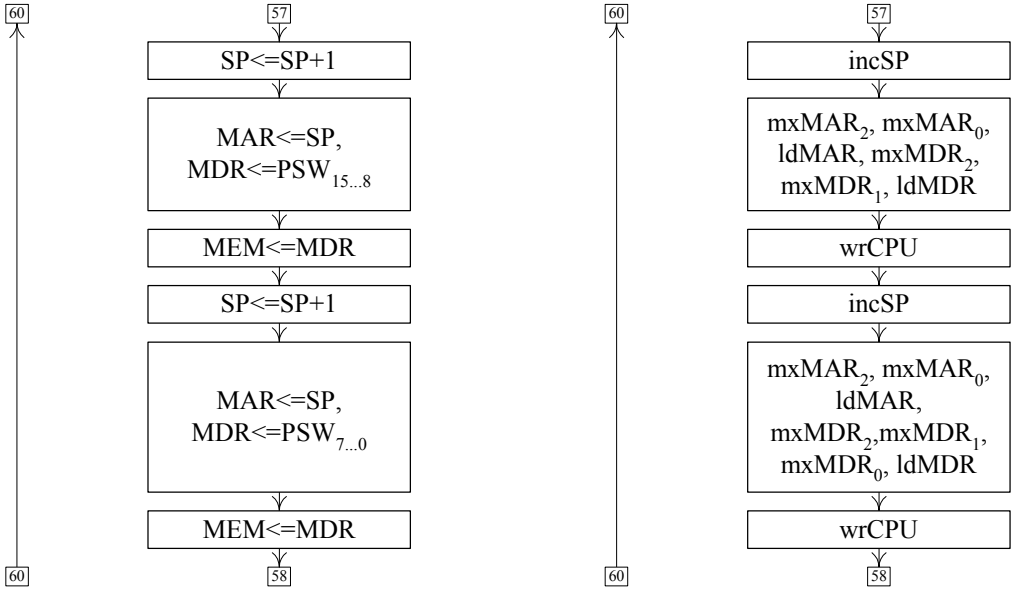
! Čuvanje konteksta procesora !

! Kontekst procesora i to $PC_{15...0}$ i $PSW_{15...0}$ se čuva u koracima $step_{8A}$ do $step_{95}$. U koracima $step_{8A}$ do ste_{8F} se na stek stavlja programski brojač $PC_{15...0}$. Na stek se stavlja prvo viši a zatim i niži bajt registra $PC_{15...0}$. Stoga se najpre u koraku $step_{8A}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$ bloka *addr*. Zatim se u koraku $step_{8B}$ vrednostima 1 signala **$mxMAR_2$**, **$mxMAR_0$** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **$mxMDR_2$** i **ldMDR** sadržaj višeg bajta registra $PC_{15...8}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{8C}$ na isti načina kao što se to radi u koraku $step_{38}$ instrukcije STB. Potom se u koraku $step_{8D}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{8E}$ vrednostima 1 signala **$mxMAR_2$**, **$mxMAR_0$** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **$mxMDR_2$**, **$mxMDR_0$**, **ldMDR** sadržaj nižeg bajta registra $PC_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{8F}$ na isti načina kao što se to radi u koraku $step_{8C}$ prilikom upisa višeg bajta.!



Slika 45 Opsluživanje prekida (prvi deo)

step$_{8A}$ **incSP**;
step$_{8B}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **mxMDR$_2$, ldMDR**;
step$_{8C}$ **wrCPU**;
step$_{8D}$ **incSP**;
step$_{8E}$ **mxMAR$_2$, mxMAR$_0$, ldMAR**, **mxMDR$_2$, mxMDR$_0$, ldMDR**;
step$_{8F}$ **wrCPU**;



Slika 46 Opsluživanje prekida (drugi deo)

! U koracima step$_{90}$ do step$_{95}$ se na stek stavlja programska statusna reč PSW$_{15...0}$ bloka *exec*. Na stek se stavlja prvo viši a zatim i niži bajt registra PSW$_{15...0}$. Stoga se najpre u koraku step$_{90}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra SP$_{15...0}$ bloka *addr*. Zatim se u koraku step$_{91}$ vrednostima 1 signala **mxMAR$_2$, mxMAR$_0$** i **ldMAR**, sadržaj registra SP$_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar MAR$_{15...0}$ i vrednostima 1 signala **mxMDR$_2$, mxMDR$_0$** i **ldMDR** sadržaj višeg bajta registra PSW$_{15...8}$ propušta kroz multiplekser MX2 bloka *bus* i upisuje u registar MDR$_{7...0}$. Upis se realizuje u koraku step$_{92}$ na isti način kao što se to radi u koraku step$_{38}$ instrukcije STB. Potom se u koraku step$_{93}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra SP$_{15...0}$. Zatim se u koraku

$step_{94}$ vrednostima 1 signala **$mxMAR_2$**, **$mxMAR_0$** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **$mxMDR_2$**, **$mxMDR_1$**, **$mxMDR_0$**, **ldMDR** sadržaj nižeg bajta registra $PSW_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{95}$ na isti načina kao što se to radi u koraku $step_{92}$ prilikom upisa višeg bajta. !

$step_{90}$ **incSP**;
$step_{91}$ **$mxMAR_2$**, **$mxMAR_0$**, **ldMAR**, **$mxMDR_2$**, **$mxMDR_1$**, **ldMDR**;
$step_{92}$ **wrCPU**;
$step_{93}$ **incSP**;
$step_{94}$ **$mxMAR_2$**, **$mxMAR_0$**, **ldMAR**, **$mxMDR_2$**, **$mxMDR_1$**, **$mxMDR_0$**, **ldMDR**;
$step_{95}$ **wrCPU**;

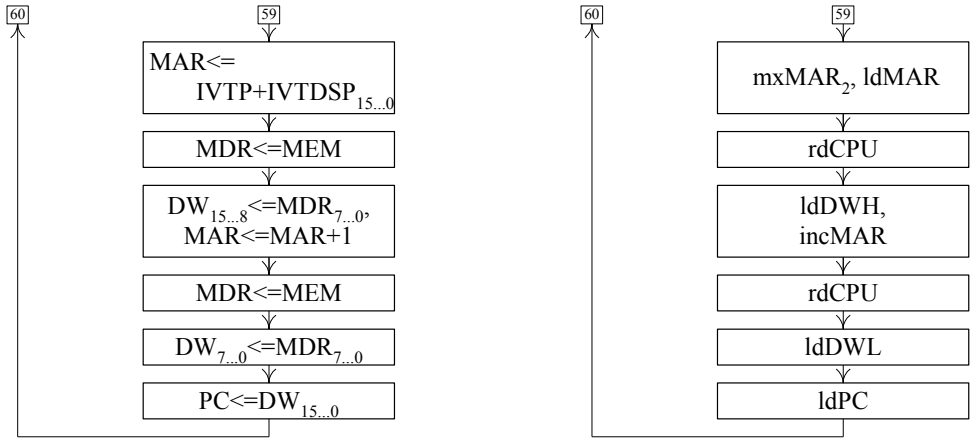
! Utvrđivanje broja ulaza !

! U korak $step_{96}$ se dolazi iz $step_{95}$. U koraku $step_{96}$ se sadržaj $UEXT_{2...0}$ sa izlaza kodera CD bloka *intr*, koji sadrži broj ulaza u tabelu sa adresama prekidnih rutina, vrednošću 1 signala **ldBR** upisuje u registar $BR_{2...0}$. Iz koraka $step_{96}$ se prelazi na korak $step_{97}$ radi utvrđivanja adrese prekidne rutine. !

$step_{96}$ **ldBR**;

Slika 47 Opsluživanje prekida (treći deo)

! Utvrđivanje adrese prekidne rutine !



Slika 48 Opsluživanje prekida (četvrti deo)

! U korak $step_{97}$ se dolazi iz koraka $step_{96}$. U koracima $step_{97}$ do $step_{9C}$ se na osnovu dobijenog broja ulaza i sadržaja registra koji ukazuje na početnu adresu tabele sa adresama pekidnih rutina, iz odgovarajućeg ulaza čita adresa prekidne rutine i upisuje u programski brojač $PC_{15...0}$ bloka *fetch*. U koraku $step_{97}$ se vrednostima 0 signala **$mxADDA_1$**, **$mxADDA_0$**, **$mxADDB_1$** i **$MXADDB_0$** bloka *addr* kroz multipleksere MX2 i MX3 na ulaze sabirača ADD propuštaju sadržaj registra $IVTP_{15...0}$ i sadržaj $IVTDSP_{15...0}$ koji predstavlja sadržaj registra $BR_{2...0}$ pomeren ulevo za jedno mesto i proširen nulama do dužine 16 bita i vrednostima 1 signala **$mxMAR_2$** i **ldMAR** bloka *bus* se sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa memorijske lokacije počev od koje treba pročitati dva bajta koji predstavljaju viši i niži bajt adrese prekidne rutine. Čitanje prvog bajta se realizuje u koraku $step_{98}$, a drugog bajta u koraku $step_{9A}$ na isti način kao u koraku $step_{02}$ kod čitanja prvog bajta instrukcije. U koraku $step_{99}$ se prvi bajt vrednošću 1 signala **ldDWH** upisuje u viši bajt registra $DW_{15...8}$ bloka *bus*, a vrednošću 1 signala **incMAR** adresni registar $MAR_{15...0}$ inkrementira na adresu sledećeg bajta. U koraku $step_{9B}$ se drugi bajt vrednošću 1 signala **ldDWL** upisuje u niži bajt registra $DW_{7...0}$. Time se u registru $DW_{15...0}$ nalazi adresa prekidne rutine. Na kraju se u koraku $step_{9C}$ vrednostima 0 signala **$mxPC_1$** i **$mxPC_0$** sadržaj registra $DW_{15...0}$ propušta kroz multiplekser MX1 bloka *fetch* i vrednošću 1

signala **ldPC** upisuje u registar $PC_{15...0}$. Time se u registru $PC_{15...0}$ nalazi adresa prve instrukcije prekidne rutine. Iz koraka $step_{9C}$ se bezuslovno prelazi na $step_{00}$. !

$step_{97}$ **mxMAR$_2$**, **ldMAR**;
$step_{98}$ **rdCPU**;
$step_{99}$ **ldDWH**, **incMAR**;
$step_{9A}$ **rdCPU**;
$step_{9B}$ **ldDWL**;
$step_{9C}$ **ldPC**,
br $step_{00}$;

3.2.3 Struktura upravljačke jedinice

U ovom odeljku se daje struktura upravljačke jedinice ožičene realizacije i struktura upravljačke jedinice mikroprogramske realizacije.

3.2.3.1 Struktura upravljačke jedinice ožičene realizacije

Upravljačka jedinice generiše dve vrste upravljačkih signala i to:

- upravljačke signale blokova operacione jedinice ***oper*** i
- upravljačke signale upravljačke jedinice ***uprav***.

Upravljački signali blokova operacione jedinice ***oper*** se koriste u blokovima operacione jedinice ***oper*** radi izvršavanja mikrooperacija. Upravljački signali upravljačke jedinice ***uprav*** se koriste u upravljačkoj jedinici ***uprav*** radi inkrementiranja brojača koraka ili upisa nove vrednosti u brojač koraka i radi generisanja vrednosti za upis u brojač koraka.

Upravljački signali operacione jedinice bi mogli da se generišu na osnovu sekvence upravljačkih signala po koracima (tabela 13). Za svaki upravljački signal operacione jedinice trebalo bi proći kroz sekvencu upravljačkih signala po koracima, tražiti korake u kojima se pojavljuje dati signal i izraz za dati signal formirati kao uniju signala dekodovanih stanja brojača koraka koji odgovaraju koracima u kojima se pojavljuje dati signal.

Upravljački signali upravljačke jedinice se ne mogu generisati na osnovu sekvence upravljačkih signala po koracima (tabela 13), 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 u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije 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 u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije staviti signal bezuslovnog skoka koji određuje da se bezuslovno prelazi na korak $step_A$ i signal **val$_A$** koji određuje da treba formirati binarnu vrednost A za upis u brojač koraka. Simbolička oznaka signala bezuslovnog skoka je **bruncnd**. Koraci $step_i$ u kojima se

bezuslovni skokovi javljaju, koraci $step_A$ na koje se bezuslovno skače, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A u heksadecimalnom dati su u tabeli 14.

Tabela 14 Koraci $step_i$, $step_A$, signali $\mathbf{val_A}$ i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	$\mathbf{val_A}$	A
$step_{13}$	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{14}$	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{16}$	$step_{26}$	$\mathbf{val_{26}}$	26
$step_{18}$	$step_{26}$	$\mathbf{val_{26}}$	26
$step_{1F}$	$step_{26}$	$\mathbf{val_{26}}$	26
$step_{21}$	$step_{26}$	$\mathbf{val_{26}}$	26
$step_{24}$	$step_{26}$	$\mathbf{val_{26}}$	26
$step_{28}$	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{2C}$	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{2E}$	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{31}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{32}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{34}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{35}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{39}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{40}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{41}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{46}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{4E}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{52}$	$step_{89}$	$\mathbf{val_{89}}$	89

$step_i$	$step_A$	$\mathbf{val_A}$	A
$step_{59}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{5A}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{5B}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{5C}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{5D}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{5F}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{61}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{63}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{65}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{67}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{69}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{6B}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{6D}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{6F}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{71}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{73}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{74}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{7B}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{9C}$	$step_{00}$	$\mathbf{val_{00}}$	00

Umesto iskaza *br* (*if* **uslov** *then* $step_A$) treba u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije staviti signal uslovnog skoka pridružen signalu **uslov** koji treba da ima vrednost 1 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 signal **uslov** ima vrednost 1. Simboličke oznake pridruženih signala uslovnih skokova i signala uslova za sve iskaze ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima, dati su u tabeli 15. Koraci $step_i$ u kojima se uslovni skokovi javljaju, signali uslova **uslov**, koraci $step_A$ na koje se uslovno skače, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A u heksadecimalnom obliku dati su u tabeli 16.

Tabela 15 Signali uslovnih skokova i signali uslova

signal uslovnog skoka	signal uslova
brnotSTART	$\overline{\mathbf{START}}$
brl1	**l1**
brl2_brnch	**l2_brnch**
brl2_arlog	**l2_arlog**
brl3_jump	**l3_jump**
brl3_arlog	**l3_arlog**

signal uslovnog skoka	signal uslova
brstore	**store**
brLDW	**LDW**
brdirreg	**dirreg**
brnotbrpom	$\overline{\mathbf{brpom}}$
brnotprekid	$\overline{\mathbf{prekid}}$

Tabela 16 Koraci $step_i$, signali uslova **uslov**, koraci $step_A$, signali $\mathbf{val_A}$ i vrednosti A za uslovne skokove

$step_i$	uslov	$step_A$	$\mathbf{val_A}$	A
$step_{00}$	$\overline{\mathbf{START}}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{04}$	**l1**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{07}$	**l2_brnch**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{08}$	**l2_arlog**	$step_{10}$	$\mathbf{val_{10}}$	10
$step_{0B}$	**l3_jump**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{0C}$	**l3_arlog**	$step_{10}$	$\mathbf{val_{10}}$	10

$step_i$	uslov	$step_A$	$\mathbf{val_A}$	A
$step_{1E}$	**store**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{20}$	**store**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{23}$	**store**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{25}$	**store**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{27}$	**LDW**	$step_{29}$	$\mathbf{val_{29}}$	29
$step_{2D}$	**LDW**	$step_{2F}$	$\mathbf{val_{2F}}$	2F

$step_{11}$	**store**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{12}$	**LDW**	$step_{14}$	$\mathbf{val_{14}}$	14
$step_{15}$	**store**	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{17}$	**store**	$step_{30}$	$\mathbf{val_{30}}$	30

$step_{36}$	**dirreg**	$step_{3A}$	$\mathbf{val_{3A}}$	3A
$step_{3B}$	**dirreg**	$step_{41}$	$\mathbf{val_{41}}$	41
$step_{72}$	$\overline{\textbf{brpom}}$	$step_{89}$	$\mathbf{val_{89}}$	89
$step_{89}$	$\overline{\textbf{prekid}}$	$step_{00}$	$\mathbf{val_{00}}$	00

Umesto iskaza *br* (*case* (**uslov$_1$**, ..., **uslov$_n$**) *then* (**uslov$_1$**, $step_{A1}$), ..., (**uslov$_n$**, $step_{An}$) treba u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije staviti signal višestrukog uslovnog skoka pridružen signalima **uslov$_1$**, **uslov$_2$**,..., **uslov$_n$** od kojih jedan treba da ima vrednost 1 da bi se realizovao prelazak na jedan od koraka $step_{A1}$, $step_{A2}$, ..., $step_{An}$. Koraci $step_i$ u kojima se višestruki uslovni skokovi javljaju i simboličke oznake pridruženih signala višestrukih uslovnih skokova za sve korake ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima dati su u tabeli 17. Signali uslova **uslov$_1$**, **uslov$_2$**,..., **uslov$_n$**, koraci $step_{A1}$, $step_{A2}$, ..., $step_{An}$ na koje se uslovno skače i vrednosti A1, A2, ..., An u heksadecimalnom koje treba da se upišu u brojač koraka u zavisnosti od toga koji od signala uslova **uslov$_1$**, **uslov$_2$**,..., **uslov$_n$** ima vrednost 1 za višestruke uslovne skokove u koracima $step_{10}$ i $step_{30}$ dati su tabelama 18 i 19.

Tabela 17 Koraci sa višestrukim uslovnim skokovima i signali višestrukih uslovnih skokova

$step_i$	signal
$step_{10}$	**bradr**
$step_{30}$	**bropr**

Tabela 18 Signali uslova, koraci na koje se skače i vrednosti A za upis u brojač koraka za višestruki uslovni skok u koraku $step_{10}$

uslov	$step_A$	A
regdir	$step_{11}$	11
regind	$step_{15}$	15
memdir	$step_{17}$	17
memind	$step_{19}$	19
regindpom	$step_{20}$	20
bxpom	$step_{22}$	22
pcpom	$step_{25}$	25
imm	$step_{2D}$	2D

Tabela 19 Signali uslova, koraci na koje se skače i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{30}$

uslov	$step_A$	A
INTD	$step_{31}$	31
INTE	$step_{32}$	32
LDB	$step_{33}$	33
LDW	$step_{35}$	35
STB	$step_{36}$	36
STW	$step_{3B}$	3B
POPB	$step_{42}$	42
POPW	$step_{47}$	47
PUSHB	$step_{4F}$	4F
PUSHW	$step_{53}$	53
LDIVTP	$step_{5A}$	5A
STIVTP	$step_{5B}$	5B
LDSP	$step_{5C}$	5C
STSP	$step_{5D}$	5D
ADD	$step_{5E}$	5E
SUB	$step_{60}$	60
INC	$step_{62}$	62
DEC	$step_{64}$	64

uslov	$step_A$	A
LSL	$step_{70}$	70
ROL	$step_{70}$	70
ROLC	$step_{70}$	70
BEQL	$step_{72}$	72
BNEQ	$step_{72}$	72
BNEG	$step_{72}$	72
BNNEG	$step_{72}$	72
BOVF	$step_{72}$	72
BNOVF	$step_{72}$	72
BCAR	$step_{72}$	72
BNCAR	$step_{72}$	72
BGRT	$step_{72}$	72
BGRE	$step_{72}$	72
BLSS	$step_{72}$	72
BLSSE	$step_{72}$	72
BGRT	$step_{72}$	72
BGRE	$step_{72}$	72
BLSS	$step_{72}$	72

AND	$step_{66}$	66
OR	$step_{68}$	68
XOR	$step_{6A}$	6A
NOT	$step_{6C}$	6C
ASR	$step_{6E}$	6E
LSR	$step_{6E}$	6E
ROR	$step_{6E}$	6E
RORC	$step_{6E}$	6E
ASL	$step_{70}$	70

BLSSE	$step_{72}$	72
BGRTU	$step_{72}$	72
BGRTEU	$step_{72}$	72
BLSSU	$step_{72}$	72
BLSSEU	$step_{72}$	72
JMP	$step_{74}$	74
JSR	$step_{75}$	75
RTI	$step_{7C}$	7C
RTS	$step_{82}$	82

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu ožičene realizacije signal bezuslovnog skoka **bruncnd**, signali uslovnih skokova (tabela 15), signali višestrukih uslovnih skokova (tabela 17) i signali $\mathbf{val_A}$ za bezuslovne (tabela 14) i uslovne (tabela 16) skokove.

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela tabela 13), formirana sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 20). Jedna linija u toj sekvenci ima sledeću formu: na levoj strani nalazi se signal dekodovanog stanja brojača koraka, u sredini je niz upravljačkih signala operacione i upravljačke jedinice koji imaju vrednost 1 kada dati signal dekodovanog stanja brojača koraka ima vrednost 1, dok komentar, tamo gde postoji, počinje uskličnikom (!) i proteže se do sledećeg uskličnika (!).

Upravljački signali operacione jedinice i upravljačke jedinice se generišu na identičan način na osnovu sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 20). Za svaki upravljački signal operacione jedinice i upravljačke jedinice treba proći kroz sekvencu upravljačkih signala za upravljačku jedinicu ožičene realizacije, tražiti korake u kojima se pojavljuje dati signal i izraz za dati signal formirati kao uniju signala dekodovanih stanja brojača koraka koji odgovaraju koracima u kojima se pojavljuje dati signal.

Tabela 20 Sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije

! Čitanje instrukcije !

T_{00} **brnotSTART**, $\mathbf{val_{00}}$;
T_{01} **ldMAR**, **incPC**;
T_{02} **rdCPU;**
T_{03} **ldIR0**;
T_{04} **brl1**, $\mathbf{val_{30}}$;
T_{05} **ldMAR**, **incPC**;
T_{06} **rdCPU**;
T_{07} **ldIR1**, **ldGPRADR**,
brl2_brnch, $\mathbf{val_{30}}$;
T_{08} **brl2_arlog**, $\mathbf{val_{10}}$;
T_{09} **ldMAR**, **incPC**;
T_{0A} **rdCPU**;
T_{0B} **ldIR2**,
brl3_jump, $\mathbf{val_{30}}$;
T_{0C} **brl3_arlog**, $\mathbf{val_{10}}$;
T_{0D} **ldMAR**, **incPC**;
T_{0E} **rdCPU**;
T_{0F} **ldIR3**;

! Formiranje adrese i čitanje operanda !

T_{10} **bradr**;

! Registarsko direktno adresiranje !

T_{11} **brstore**, **val$_{30}$**;
T_{12} **brLDW**, **val$_{14}$**;
T_{13} **ldBB**,
bruncnd, **val$_{30}$**;
T_{14} **ldBW**,
bruncnd, **val$_{30}$**;

! Registarsko indirektno adresiranje !

T_{15} **mxMAR$_0$**, **ldMAR**,
brstore, **val$_{30}$**;
T_{16} **bruncnd**, **val$_{26}$**;

! Memorijsko direktno adresiranje !

T_{17} **mxMAR$_1$**, **ldMAR**,
brstore, **val$_{30}$**;
T_{18} **bruncnd**,**val$_{26}$**;

! Memorijsko indirektno adresiranje !

T_{19} **mxMAR$_1$**, **ldMAR**;
T_{1A} **rdCPU**;
T_{1B} **ldDWH**, **incMAR**;
T_{1C} **rdCPU**;
T_{1D} **ldDWL**;
T_{1E} **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**,
brstore, **val$_{30}$**;
T_{1F} **bruncnd**, **val$_{26}$**;

! Registarsko indirektno adresiranje sa pomerajem !

T_{20} **mxADDA$_0$**, **mxADDB$_0$**, **mxMAR$_2$**, **ldMAR**,
brstore, **val$_{30}$**;
T_{21} **bruncnd**, **val$_{26}$**;

! Bazno indeksno adresiranje sa pomerajem !

T_{22} **mxADDA$_0$**, **mxADDB$_0$**, **ldCW**, **incGPRAR**;
T_{23} **mxADDA$_1$**, **mxADDA$_0$**, **mxADDB$_1$**, **mxMAR$_2$**, **ldMAR**,
brstore, **val$_{30}$**;
T_{24} **bruncnd**, **val$_{26}$**;

! PC relativno adresiranje !

T_{25} **mxADDA$_1$**, **mxADDB$_0$**, **mxMAR$_2$**, **ldMAR**,
brstore, **val$_{30}$**;

! Čitanje operanda !

T_{26} **rdCPU**;
T_{27} **brLDW**, **val$_{29}$**;
T_{28} **mxBB$_0$**, **ldBB**,
bruncnd, **val$_{30}$**;
T_{29} **ldDWH**, **incMAR**;
T_{2A} **rdCPU**;
T_{2B} **ldDWL**;
T_{2C} **mxBW$_0$**, **ldBW**,
bruncnd, **val$_{30}$**;

! Neposredno adresiranje !

T_{2D} **brLDW**, **val$_{2F}$**;
T_{2E} **mxBB$_1$**, **ldBB**,
bruncnd, **val$_{30}$**;
T_{2F} **mxBW$_1$**, **ldBW**;

! Izvršavanje operacije !

T_{30} **bropr** ;

! INTD !

T_{31} **clPSWI**,
bruncnd, **val$_{89}$**;

! INTE !

T_{32} **stPSWI**,
bruncnd, **val$_{89}$**;

! LDB !

T_{33} **mxAB**, **ldAB**;
T_{34} **ldN**, **ldZ**, **ldC**, **ldV**,
bruncnd, **val$_{89}$**;

! LDW !

T_{35} **ldAW**,
bruncnd, **val$_{89}$**;

! STB !

T_{36} **brdirreg**, **val$_{3A}$**;
T_{37} **mxMDR$_0$**, **ldMDR**;
T_{38} **wrCPU**;
T_{39} **bruncnd**, **val$_{89}$**;
T_{3A} **wrGPR**,
bruncnd, **val$_{89}$**;

! STW !

T_{3B} **brdirreg**, **val$_{41}$**;
T_{3C} **mxMDR$_1$**, **ldMDR**;
T_{3D} **wrCPU**;
T_{3E} **mxMDR$_1$**, **mxMDR$_0$**, **ldMDR**, **incMAR**;
T_{3F} **wrCPU**;
T_{40} **bruncnd**, **val$_{89}$**;
T_{41} **mxGPR**, **wrGPR**,
bruncnd, **val$_{89}$**;

! POPB !

T_{42} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**;
T_{43} **rdCPU**;
T_{44} **mxBB$_0$**, **ldBB**;
T_{45} **mxAB**, **ldAB**;
T_{46} **ldN**, **ldZ**, **ldC**, **ldV**,
bruncnd, **val$_{89}$**;

! POPW !

T_{47} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**;
T_{48} **rdCPU**;
T_{49} **ldDWL**;
T_{4A} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**;
T_{4B} **rdCPU**;
T_{4C} **ldDWH**;
T_{4D} **mxBW$_0$**, **ldBW**;
T_{4E} **ldAW**,
bruncnd, **val$_{89}$**;

! PUSHB !

T_{4F} **incSP**;
T_{50} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_0$**, **ldMDR**;
T_{51} **wrCPU**;
T_{52} **bruncnd**, **val$_{89}$**;

! PUSHW !

T_{53} **incSP**;

T_{54} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_1$**, **ldMDR**;
T_{55} **wrCPU**;
T_{56} **incSP**;
T_{57} **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_1$**, **mxMDR$_0$**, **ldMDR**;
T_{58} **wrCPU**;
T_{59} **bruncnd**, **val$_{89}$**;

! LDIVTP !

T_{5A} **mxAW$_1$**, **ldAW**,
bruncnd,**val$_{89}$**;

! STIVTP !

T_{5B} **ldIVTP**,
bruncnd, **val$_{89}$**;

! LDSP !

T_{5C} **mxAW$_0$**, **ldAW**,
bruncnd, **val$_{89}$**;

! STSP !

T_{5D} **ldSP**,
bruncnd, **val$_{89}$**;

! ADD !

T_{5E} **add**, **ldAB**, **ldC**, **ldV**;
T_{5F} **ldN**, **ldZ**,
bruncnd, **val$_{89}$**;

! SUB !

T_{60} **sub**, **ldAB**, **ldC**, **ldV**;
T_{61} **ldN**, **ldZ**,
bruncnd, **val$_{89}$**;

! INC !

T_{62} **inc**, **ldAB**, **ldC**, **ldV**;
T_{63} **ldN**, **ldZ**,
bruncnd, **val$_{89}$**;

! DEC !

T_{64} **dec**, **ldAB**, **ldC**, **ldV**;
T_{65} **ldN**, **ldZ**,
bruncnd, **val$_{89}$**;

! AND !

T_{66} **and**, **ldAB**;
T_{67} **ldN**, **ldZ**, **ldC**, **ldV**,
bruncnd, **val$_{89}$**;

! OR !

T_{68} **or**, **ldAB**;
T_{69} **ldN**, **ldZ**, **ldC**, **ldV**,
bruncnd, **val$_{89}$**;

! XOR !

T_{6A} **xor**, **ldAB**;
T_{6B} **ldN**, **ldZ**, **ldC**, **ldV**,
bruncnd, **val$_{89}$**;

! NOT !

T_{6C} **not**, **ldAB**;
T_{6D} **ldN**, **ldZ**, **ldC**, **ldV**,
bruncnd, **val$_{89}$**;

! ASR, LSR, ROR i ROLC !

T_{6E} **shr**, **ldC**;
T_{6F} **ldN**, **ldZ**, **ldV**,

bruncnd, **val$_{89}$**;

! ASL, LSL, ROL i ROLC !

T$_{70}$ **shl**, **ldC**;

T$_{71}$ **ldN**, **ldZ**, **ldV**,
bruncnd, **val$_{89}$**;

! BEQL,…, BLSSEU !

T$_{72}$ **brnotbrprom**, **val$_{89}$**;

T$_{73}$ **mxADDA$_1$**, **mxADDB$_1$**, **mxADDB$_0$**, **mxPC$_0$**, **ldPC**,
bruncnd, **val$_{89}$**;

! JMP !

T$_{74}$ **mxPC$_1$**, **ldPC**,
bruncnd, **val$_{89}$**;

! JSR !

T$_{75}$ **incSP**;

T$_{76}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **ldMDR**;

T$_{77}$ **wrCPU**;

T$_{78}$ **incSP**;

T$_{79}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **mxMDR$_0$**, **ldMDR**;

T$_{7A}$ **wrCPU**,

T$_{7B}$ **mxPC$_1$**, **ldPC**,
bruncnd, **val$_{30}$**;

! RTI !

T$_{7C}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**;

T$_{7D}$ **rdCPU**;

T$_{7E}$ **ldPSWL**;

T$_{7F}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**;

T$_{80}$ **rdCPU**;

T$_{81}$ **ldPSWH**;

! RTS !

T$_{82}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**;

T$_{83}$ **rdCPU**;

T$_{84}$ **ldDWL**;

T$_{85}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**;

T$_{86}$ **rdCPU**;

T$_{87}$ **ldDWH**;

T$_{88}$ **ldPC**;

! Opsluživanje prekida !

T$_{89}$ **brnptprekid**, **val$_{00}$**;

! Čuvanje konteksta procesora !

T$_{8A}$ **incSP**;

T$_{8B}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **ldMDR**;

T$_{8C}$ **wrCPU**;

T$_{8D}$ **incSP**;

T$_{8E}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **mxMDR$_0$**, **ldMDR**;

T$_{8F}$ **wrCPU**;

T$_{90}$ **incSP**;

T$_{91}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **mxMDR$_1$**, **ldMDR**;

T$_{92}$ **wrCPU**;

T$_{93}$ **incSP**;

T$_{94}$ **mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **mxMDR$_1$**, **mxMDR$_0$**, **ldMDR**;

T$_{95}$ **wrCP**;

! Utvrđivanje broja ulaza !

T_{96} **ldBR**;

! Utvrđivanje adrese prekidne rutine !

T_{97} **mxMAR$_2$**, **ldMAR**;

T_{98} **rdCPU**;

T_{99} **ldDWH**, **incMAR**;

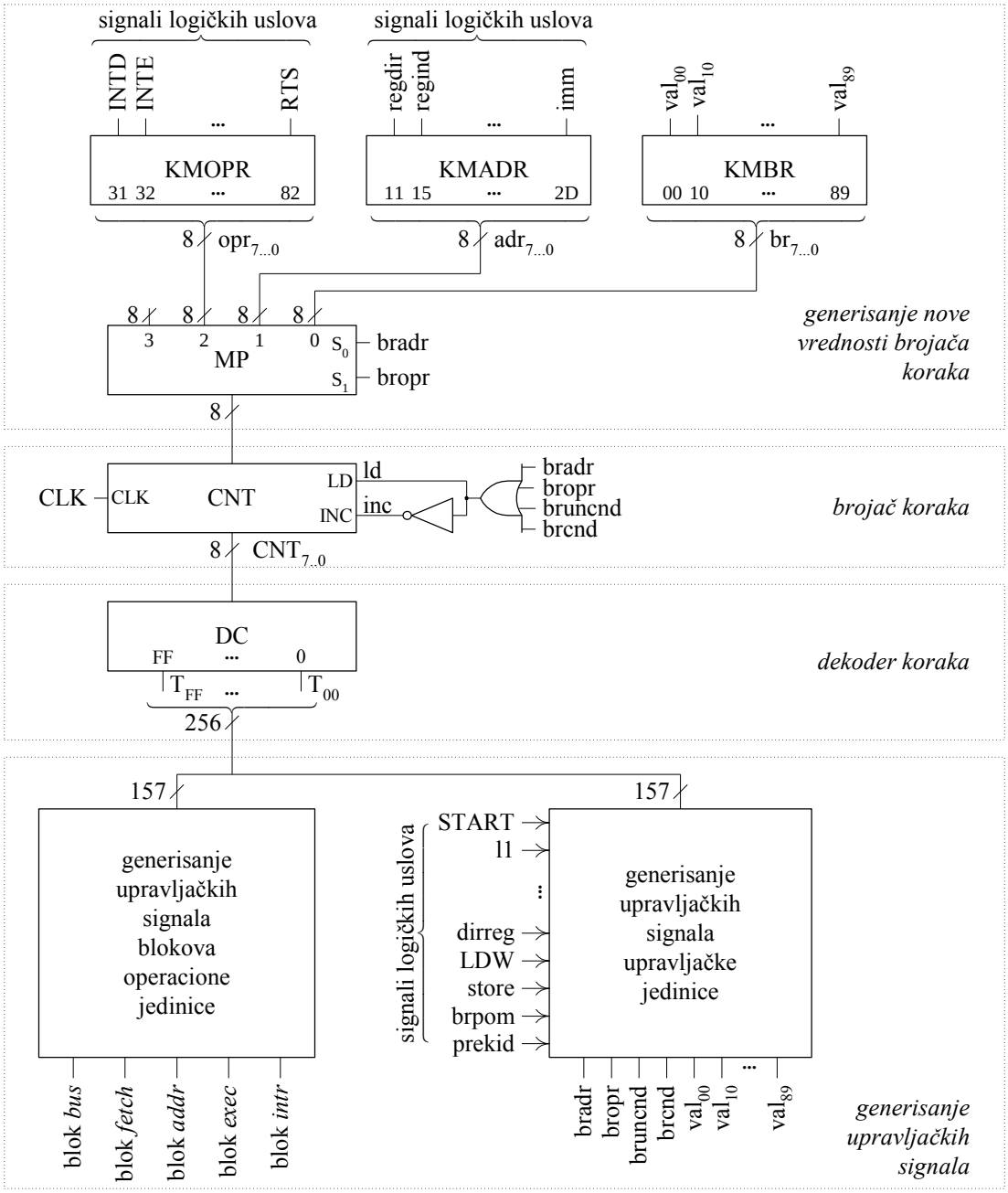
T_{9A} **rdCPU**;

T_{9B} **ldDWL**;

T_{9C} **ldPC**,

bruncnd, **val$_{00}$**;

Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 49. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti brojača koraka*, blok *brojač koraka*, blok *dekoder koraka* i blok *generisanje upravljačkih signala*. Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.



Slika 49 Struktura upravljačke jedinice ožičene realizacije

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 $CNT_{7...0}$. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikrooperacija. Vrednosti koje treba upisati u brojač koraka generišu se na tri načina i to pomoću kombinacione mreže KMOPR koja formira signale $\mathbf{opr_{7...0}}$, kombinacione mreže KMADR koja formira signale $\mathbf{adr_{7...0}}$ i kombinacione mreže KMBR koja formira signale $\mathbf{br_{7...0}}$. Selekcija jedne od tri grupe signala koji daju novu vrednost brojača koraka $CNT_{7...0}$ obezbeđuje se signalima **bropr** i **bradr** i to signali $\mathbf{opr_{7...0}}$ ako signal **bropr** ima vrednost 1, signali $\mathbf{adr_{7...0}}$ ako signal **bradr** ima vrednost 1 i signali $\mathbf{br_{7...0}}$ ako oba signala **bropr** i **bradr** imaju vrednost 0.

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 19) za realizaciju višestrukog uslovnog skoka u koraku $step_{30}$ sekvence upravljačkih signala. U zavisnosti od toga koji od signala **INTD**, …, **RTS** ima vrednost 1 zavisi koja će od vrednosti A iz tabele 19 da se pojavi tada na linijama $\mathbf{op_{7...0}}$. S obzirom da vrednost 1 signala dekovanog stanje brojača koraka T_{30} daje vrednost 1 signala višestrukog uslovnog skoka **bropr**, vrednost na linijama $\mathbf{op_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima brojača koraka $CNT_{7...0}$.

Kombinacionom mrežom KMADR generišu se vrednosti (tabela 18) za realizaciju višestrukog uslovnog skoka u koraku $step_{10}$ sekvence upravljačkih signala. U zavisnosti od toga koji od signala **regdir**,..., **imm** ima vrednost 1 zavisi koja će od vrednosti A iz tabele 18 da se pojavi tada na linijama $\mathbf{adr_{7...0}}$. S obzirom da vrednost 1 signala dekodovanog stanja brojača koraka T_{10} daje vrednost 1 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 $CNT_{7...0}$ za bezuslovne skokove (tabela 14) i uslovne skokove (tabela 16) u sekvenci upravljačkih signala po koracima. U zavisnosti od toga koji od signala $\mathbf{val_{00}}$, $\mathbf{val_{10}}$, ..., $\mathbf{val_{89}}$ ima vrednost 1 zavisi koja će od vrednosti iz tabela 14 i 16 tada da se pojavi na linijama $\mathbf{br_{7...0}}$. Signali višestrukih uslovnih skokova **bropr** i **bradr** imaju vrednost 1 samo pri vrednostima 1 signala dekodovanih stanja brojača koraka T_{30} i T_{10}, respektivno, dok u svim ostalim situacijama imaju vrednost 0. S obzirom da nijedan od ova dva signala nema vrednost 1 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 određuje koji će upravljački signali da imaju vrednost 1. 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 (tabela 20). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko svi signali **bropr**, **bradr**, **brcnd** i **bruncnd** imaju vrednost 0. Signali **bropr**, **bradr**, **brcnd** i **bruncnd** normalno imaju vrednost 0 sem u stanjima brojača koraka koja odgovaraju koracima kada treba treba realizovati višestruki uslovni skok, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

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 (tabela 20). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ako jedan od signala **bropr**, **bradr**, **brcnd** i **bruncnd** ima vrednost 1. Signali **bropr**, **bradr**, **brcnd** i **bruncnd** normalno imaju vrednost 0 sem u stanjima brojača koraka koja odgovaraju koracima kada treba treba realizovati višestruki uslovni skok, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

Brojač koraka $CNT_{7...0}$ je dimenzionisan prema broju koraka u sekvenci upravljačkih signala (tabela 20). S obzirom da se upravljački signali svih faza izvršavanja instrukcija realizuju u opsegu od koraka T_{00} do koraka T_{9C} usvojena je dužina brojača koraka $CNT_{7...0}$ od 8 bitova.

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 13) 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 20).

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala $\mathbf{T_0}$, $\mathbf{T_1}$, ..., $\mathbf{T_{9C}}$ koji dolaze sa bloka *dekoder koraka*, signala logičkih uslova **START**, **I1**, ..., **prekid** koji dolaze iz operacione jedinice i saglasno sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 20) generišu dve grupe upravljačkih signala i to: upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice.

Upravljački signali blokova operacione jedinice ***oper*** se daju posebno za svaki blok.

Upravljački signali bloka *bus* se generišu na sledeći način:

- $\mathbf{mxMAR_2} = \mathbf{T_{20}} + \mathbf{T_{23}} + \mathbf{T_{25}} + \mathbf{T_{42}} + \mathbf{T_{47}} + \mathbf{T_{4A}} + \mathbf{T_{50}} + \mathbf{T_{54}} + \mathbf{T_{57}} + \mathbf{T_{76}} + \mathbf{T_{79}} + \mathbf{T_{7C}} + \mathbf{T_{7F}} + \mathbf{T_{82}} + \mathbf{T_{85}} + \mathbf{T_{8B}} + \mathbf{T_{8E}} + \mathbf{T_{91}} + \mathbf{T_{94}} + \mathbf{T_{97}}$
- $\mathbf{mxMAR_1} = \mathbf{T_{17}} + \mathbf{T_{19}} + \mathbf{T_{1E}}$
- $\mathbf{mxMAR_0} = \mathbf{T_{15}} + \mathbf{T_{1E}} + \mathbf{T_{42}} + \mathbf{T_{47}} + \mathbf{T_{4A}} + \mathbf{T_{50}} + \mathbf{T_{54}} + \mathbf{T_{57}} + \mathbf{T_{76}} + \mathbf{T_{79}} + \mathbf{T_{7C}} + \mathbf{T_{7F}} + \mathbf{T_{82}} + \mathbf{T_{85}} + \mathbf{T_{8B}} + \mathbf{T_{8E}} + \mathbf{T_{91}} + \mathbf{T_{94}}$
- $\mathbf{ldMAR} = \mathbf{T_{01}} + \mathbf{T_{05}} + \mathbf{T_{09}} + \mathbf{T_{0D}} + \mathbf{T_{15}} + \mathbf{T_{17}} + \mathbf{T_{19}} + \mathbf{T_{1E}} + \mathbf{T_{20}} + \mathbf{T_{23}} + \mathbf{T_{25}} + \mathbf{T_{42}} + \mathbf{T_{47}} + \mathbf{T_{4A}} + \mathbf{T_{50}} + \mathbf{T_{54}} + \mathbf{T_{57}} + \mathbf{T_{76}} + \mathbf{T_{79}} + \mathbf{T_{7C}} + \mathbf{T_{7F}} + \mathbf{T_{82}} + \mathbf{T_{85}} + \mathbf{T_{8B}} + \mathbf{T_{8E}} + \mathbf{T_{91}} + \mathbf{T_{94}} + \mathbf{T_{97}}$
- $\mathbf{incMAR} = \mathbf{T_{1B}} + \mathbf{T_{29}} + \mathbf{T_{3E}} + \mathbf{T_{99}}$
- $\mathbf{mxMDR_2} = \mathbf{T_{76}} + \mathbf{T_{79}} + \mathbf{T_{8B}} + \mathbf{T_{8E}} + \mathbf{T_{91}} + \mathbf{T_{94}}$
- $\mathbf{mxMDR_1} = \mathbf{T_{3C}} + \mathbf{T_{3E}} + \mathbf{T_{54}} + \mathbf{T_{57}} + \mathbf{T_{91}} + \mathbf{T_{94}}$
- $\mathbf{mxMDR_0} = \mathbf{T_{37}} + \mathbf{T_{3E}} + \mathbf{T_{50}} + \mathbf{T_{57}} + \mathbf{T_{79}} + \mathbf{T_{8E}} + \mathbf{T_{94}}$
- $\mathbf{ldMDR} = \mathbf{T_{37}} + \mathbf{T_{3C}} + \mathbf{T_{3E}} + \mathbf{T_{50}} + \mathbf{T_{54}} + \mathbf{T_{57}} + \mathbf{T_{76}} + \mathbf{T_{79}} + \mathbf{T_{8B}} + \mathbf{T_{8E}} + \mathbf{T_{91}} + \mathbf{T_{94}}$
- $\mathbf{rdCPU} = \mathbf{T_{02}} + \mathbf{T_{06}} + \mathbf{T_{0A}} + \mathbf{T_{0E}} + \mathbf{T_{1A}} + \mathbf{T_{1C}} + \mathbf{T_{26}} + \mathbf{T_{2A}} + \mathbf{T_{43}} + \mathbf{T_{48}} + \mathbf{T_{4B}} + \mathbf{T_{7D}} + \mathbf{T_{80}} + \mathbf{T_{83}} + \mathbf{T_{86}} + \mathbf{T_{98}} + \mathbf{T_{9A}}$
- $\mathbf{wrCPU} = \mathbf{T_{38}} + \mathbf{T_{3D}} + \mathbf{T_{3F}} + \mathbf{T_{51}} + \mathbf{T_{55}} + \mathbf{T_{58}} + \mathbf{T_{77}} + \mathbf{T_{7A}} + \mathbf{T_{8C}} + \mathbf{T_{8F}} + \mathbf{T_{92}} + \mathbf{T_{95}}$
- $\mathbf{ldDWH} = \mathbf{T_{1B}} + \mathbf{T_{29}} + \mathbf{T_{4C}} + \mathbf{T_{87}} + \mathbf{T_{99}}$
- $\mathbf{ldDWL} = \mathbf{T_{1D}} + \mathbf{T_{2B}} + \mathbf{T_{49}} + \mathbf{T_{84}} + \mathbf{T_{9B}}$

Upravljački signali bloka *fetch* se generišu na sledeći način:

- $\mathbf{mxPC_1} = \mathbf{T_{74}} + \mathbf{T_{7B}}$
- $\mathbf{mxPC_0} = \mathbf{T_{73}}$
- $\mathbf{ldPC} = \mathbf{T_{73}} + \mathbf{T_{74}} + \mathbf{T_{7B}} + \mathbf{T_{88}} + \mathbf{T_{9C}}$
- $\mathbf{incPC} = \mathbf{T_{01}} + \mathbf{T_{05}} + \mathbf{T_{09}} + \mathbf{T_{0D}}$

- **ldIR0** = $\mathbf{T_{03}}$
- **ldIR1** = $\mathbf{T_{07}}$
- **ldIR2** = $\mathbf{T_{0B}}$
- **ldIR3** = $\mathbf{T_{0F}}$

Upravljački signali bloka *addr* se generišu na sledeći način:

- **ldGPRAR** = $\mathbf{T_{07}}$
- **incGPRAR** = $\mathbf{T_{22}}$
- **mxGPR** = $\mathbf{T_{41}}$
- **wrGPR** = $\mathbf{T_{3A} + T_{41}}$
- **ldSP** = $\mathbf{T_{5D}}$
- **incSP** = $\mathbf{T_{4F} + T_{53} + T_{56} + T_{75} + T_{78} + T_{8A} + T_{8D} + T_{90} + T_{93}}$
- **decSP** = $\mathbf{T_{42} + T_{47} + T_{4A} + T_{7C} + T_{7F} + T_{82} + T_{85}}$
- **mxADDA**$_1$ = $\mathbf{T_{23} + T_{25} + T_{73}}$
- **mxADDA**$_0$ = $\mathbf{T_{20} + T_{22} + T_{23}}$
- **mxADDB**$_1$ = $\mathbf{T_{23} + T_{73}}$
- **mxADDB**$_0$ = $\mathbf{T_{20} + T_{22} + T_{25} + T_{73}}$
- **ldCW** = $\mathbf{T_{22}}$

Upravljački signali bloka *exec* se generišu na sledeći način:

- **mxAB** = $\mathbf{T_{33} + T_{45}}$
- **ldAB** = $\mathbf{T_{33} + T_{45} + T_{5E} + T_{60} + T_{62} + T_{64} + T_{66} + T_{68} + T_{6A} + T_{6C}}$
- **shr** = $\mathbf{T_{61}}$
- **shl** = $\mathbf{T_{70}}$
- **mxBB**$_1$ = $\mathbf{T_{2E}}$
- **mxBB**$_0$ = $\mathbf{T_{28} + T_{44}}$
- **ldBB** = $\mathbf{T_{13} + T_{28} + T_{2E} + T_{44}}$
- **mxAW**$_1$ = $\mathbf{T_{5A}}$
- **mxAW**$_0$ = $\mathbf{T_{5C}}$
- **ldAW** = $\mathbf{T_{35} + T_{4E} + T_{5A} + T_{5C}}$
- **mxBW**$_1$ = $\mathbf{T_{2F}}$
- **mxBW**$_0$ = $\mathbf{T_{2C} + T_{4D}}$
- **ldBW** = $\mathbf{T_{2C} + T_{2F} + T_{4D}}$
- **stPSWI** = $\mathbf{T_{32}}$
- **clPSWI** = $\mathbf{T_{31}}$
- **ldN** = $\mathbf{T_{34} + T_{46} + T_{56} + T_{61} + T_{63} + T_{65} + T_{67} + T_{69} + T_{6B} + T_{6D} + T_{6F} + T_{71}}$
- **ldZ** = $\mathbf{T_{34} + T_{46} + T_{56} + T_{61} + T_{63} + T_{65} + T_{67} + T_{69} + T_{6B} + T_{6D} + T_{6F} + T_{71}}$
- **ldC** = $\mathbf{T_{34} + T_{46} + T_{5E} + T_{60} + T_{62} + T_{64} + T_{67} + T_{69} + T_{6B} + T_{6D} + T_{6E} + T_{70}}$
- **ldV** = $\mathbf{T_{34} + T_{46} + T_{5E} + T_{60} + T_{62} + T_{64} + T_{67} + T_{69} + T_{6B} + T_{6D} + T_{6F} + T_{71}}$
- **ldPSWL** = $\mathbf{T_{7E}}$
- **ldPSWH** = $\mathbf{T_{81}}$
- **add** = $\mathbf{T_{5E}}$
- **sub** = $\mathbf{T_{60}}$
- **inc** = $\mathbf{T_{62}}$
- **dec** = $\mathbf{T_{64}}$
- **and** = $\mathbf{T_{66}}$
- **or** = $\mathbf{T_{68}}$
- **xor** = $\mathbf{T_{6A}}$

- **not** = $\mathbf{T_{6C}}$

Upravljački signali bloka *intr* se generišu na sledeći način:

- **ldIVTP** = $\mathbf{T_{5B}}$
- **ldBR** = $\mathbf{T_{96}}$

Upravljački signali upravljačke jedinice ***uprav*** se generišu na sledeći način:

- **bradr** = $\mathbf{T_{10}}$
- **bropr** = $\mathbf{T_{30}}$
- **bruncnd** = $\mathbf{T_{13} + T_{14} + T_{16} + T_{18} + T_{1F} + T_{21} + T_{24} + T_{28} + T_{2C} + T_{2E} + T_{31} + T_{32} + T_{34} + T_{35} + T_{39} + T_{3A} + T_{40} + T_{41} + T_{46} + T_{4E} + T_{52} + T_{59} + T_{5A} + T_{5B} + T_{5C} + T_{5D} + T_{5F} + T_{61} + T_{63} + T_{65} + T_{67} + T_{69} + T_{6B} + T_{6D} + T_{6F} + T_{71} + T_{73} + T_{74} + T_{7B} + T_{9C} +$
- **brcnd** = **brnotSTART**$\cdot \overline{\textbf{START}}$ + **brl1**$\cdot$**l1** + **brl2_brnch**$\cdot$ **l2_brnch** + **brl2_arlog**$\cdot$ **l2_arlog** + **brl3_jump**$\cdot$ **l3_jump** + **brl3_arlog**$\cdot$ **l3_arlog** + **brstore**$\cdot$ **store** + **brLDW**$\cdot$ **LDW** + **brdirreg**$\cdot$ **dirreg** + **brnotbrpom** $\cdot \overline{\textbf{brpom}}$
- **brnotSTART** = $\mathbf{T_{00}}$
- **brl1**= $\mathbf{T_{04}}$
- **brl2_brnch**= $\mathbf{T_{07}}$
- **brl2_arlog**= $\mathbf{T_{08}}$
- **brl3_jump**= $\mathbf{T_{0B}}$
- **brl3_arlog**= $\mathbf{T_{0C}}$
- **brstore**= $\mathbf{T_{11} + T_{15} + T_{17} + T_{1E} + T_{20} + T_{23} + T_{25}}$
- **brLDW**= $\mathbf{T_{12} + T_{27} + T_{2D} +}$
- **brdirreg**= $\mathbf{T_{36} + T_{3B}}$
- **brnotpom**= $\mathbf{T_{72}}$
- **brnotprekid**= $\mathbf{T_{89}}$
- $\mathbf{val_{00}} = \mathbf{T_{00} + T_{89} + T_{9C} +}$
- $\mathbf{val_{30}} = \mathbf{T_{04} + T_{07} + T_{0B} + T_{11} + T_{13} + T_{14} + T_{15} + T_{17} + T_{1E} + T_{20} + T_{23} + T_{25} + T_{28} + T_{2C} + T_{2E}}$
- $\mathbf{val_{10}} = \mathbf{T_{08} + T_{0C}}$
- $\mathbf{val_{14}} = \mathbf{T_{12}}$
- $\mathbf{val_{26}} = \mathbf{T_{16} + T_{18} + T_{1F} + T_{21} + T_{24}}$
- $\mathbf{val_{29}} = \mathbf{T_{27}}$
- $\mathbf{val_{2F}} = \mathbf{T_{2D}}$
- $\mathbf{val_{89}} = \mathbf{T_{31} + T_{32} + T_{34} + T_{35} + T_{39} + T_{3A} + T_{40} + T_{41} + T_{46} + T_{4E} + T_{52} + T_{59} + T_{5A} + T_{5B} + T_{5C} + T_{5D} + T_{5F} + T_{61} + T_{63} + T_{65} + T_{67} + T_{69} + T_{6B} + T_{6D} + T_{6F} + T_{71} + T_{72} + T_{73} + T_{74} + T_{7B}}$
- $\mathbf{val_{3A}} = \mathbf{T_{36}}$
- $\mathbf{val_{41}} = \mathbf{T_{3B}}$

Pri generisanju signala **brcnd** koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice ***oper*** i to:

- **START** — blok *exec*,
- **l1** — blok *fetch*,
- **l2_brnch** — blok *fetch*,
- **l2_arlog** — blok *fetch*,
- **l3_jump** — blok *fetch*,
- **l3_arlog** — blok *fetch*,
- **store** — blok *fetch*,

- **LDW** — blok *fetch*,
- **dirreg** — blok *fetch*,
- **brpom** — blok *exec*
- **prekid** — blok *intr*

3.2.3.2 Struktura upravljačke jedinice mikroprogramske realizacije

Upravljačka jedinica generiše dve vrste upravljačkih signala i to:

- upravljačke signale blokova operacione jedinice ***oper*** i
- upravljačke signale upravljačke jedinice ***uprav***.

Upravljački signali blokova operacione jedinice ***oper*** se koriste u blokovima operacione jedinice ***oper*** radi izvršavanja mikrooperacija. Upravljački signali upravljačke jedinice ***uprav*** se koriste u upravljačkoj jedinici ***uprav*** radi inkrementiranja mikroprogramskog brojača ili upisa nove vrednosti u mikroprogramski brojač i radi generisanja vrednosti za upis u mikroprogramski brojač.

Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu sekvence upravljačkih signala po koracima (tabela 13). Mikroprogram se formira tako što se svakom koraku u sekvenci upravljačkih signala po koracima pridruži binarna reč sa slike 50. Te binarna reči se naziva mikroinstrukcija, mikronaredba ili mikrokomanda. Uređeni niz mikroinstrukcija pridruženih koracima u sekvenci upravljačkih signala po koracima naziva se mikroprogram.

0	1	2	3	4	5	6	7
-	$mxMAR_2$	$mxMAR_1$	$mxMAR_0$	-	$mxMDR_2$	$mxMDR_1$	$mxMDR_0$

8	9	10	11	12	13	14	15
ldMAR	incMAR	ldDWH	ldDWL	ldMDR	-	rdCPU	wrCPU

16	17	18	19	20	21	22	23
incPC	ldPC	$mxPC_1$	$mxPC_0$	$ldIR_0$	$ldIR_1$	$ldIR_2$	$ldIR_3$

24	25	26	27	28	29	30	31
ldSP	ldCW	mxGPR	wrGPR	ldGPRAR	incGPRAR	$mxADDA_1$	$mxADDA_0$

32	33	34	35	36	37	38	39
incSP	decSP	$mxADDB_1$	$mxADDB_0$	dec	inc	sub	add

40	41	42	43	44	45	46	47
not	xor	or	and	shr	shl	mxAB	ldAB

48	49	50	51	52	53	54	55
ldIVTP	ldBB	$mxBB_1$	$mxBB_0$	-	ldAW	$mxAW_1$	$mxAW_0$

56	57	58	59	60	61	62	63
ldBR	ldBW	$mxBW_1$	$mxBW_0$	ldN	ldZ	ldC	ldV

64	65	66	67	68	69	70	71
clPSWI	stPSWI	ldPSWH	ldPSWL	*cc*			

72	73	74	75	76	77	78	79
ba							

Slika 50 Mikroinstrukcija

Mikroinstrukcija ima dva dela i to operacini deo i upravljački deo. Operacioni deo čine bitovi 0 do 67, a upravljački deo čine bitovi 68 do 79. Operacioni deo se koristi za generisanje upravljačkih signala operacione jedinice, a upravljački deo se koristi za generisanje upravljačkih signala upravljačke jedinice.

Operacioni deo ima poseban bit za svaki upravljački signal operacione jedinice. Određeni bit operacionog dela mikroinstrukcije treba da ima vrednost 1 ili 0 u zavisnosti od toga da li u koraku za koji se formira mikroinstrukcija upravljački signal operacione jedinice kome je pridružen dati bit ima vrednost 1 ili 0, respektivno.

Upravljački deo ima dva polja i to polje *cc* i polje *ba*.

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 skokovi se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 13) 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 je dat u tabeli 21.

Tabela 21 Signal bezuslovnog skoka

signal bezuslovnog skoka	*cc*
bruncnd	1

Uslovni skokovi se realizuju u onim koracima sekvence upravljačkih signala po koracima u kojima se pojavljuju iskazi tipa *br* (*if* **uslov** *then* $step_A$). Simbolička oznaka signala uslovnog skoka koji za svaki od njih treba generisati, način njegovog kodiranja bitovima polja *cc* mikroinstrukcije i signal **uslov** koji treba da ima vrednost 1 da bi se realizovao skok dati su u tabeli 22.

Tabela 22 Signali uslovnih skokova

signal uslovnog skoka	polje *cc*	signal uslova
brnotSTART	2	$\overline{\textbf{START}}$
brl1	3	**l1**
brl2_brnch	4	**l2_brnch**
brl2_arlog	5	**l2_arlog**
brl3_jump	6	**l3_jump**
brl3_arlog	**7**	**l3_arlog**

signal uslovnog skoka	polje *cc*	signal uslova
brstore	8	**store**
brLDW	9	**LDW**
brdirreg	A	**dirreg**
brnotbrpom	B	$\overline{\textbf{brpom}}$
brnotprekid	C	$\overline{\textbf{prekid}}$

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* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$). Simbolička oznaka signala višestrukog uslovnog skoka koji za svaki od njih treba generisati, način njegovog kodiranja bitovima polja *cc* mikroinstrukcije i koraci u sekvenci upravljačkih signala po koracima u kojima se pojavljuju iskazi ovog tipa dati su u tabeli 23.

Tabela 23 Signali višestrukih uslovnih skokova

signal višestrukog uslovnog skoka	polje *cc*	korak
bradr	D	$step_{10}$
bropr	E	$step_{30}$

Vrednosti 0 i F polja *cc* koje nisu dodeljene signalu bezuslovnog skoka, signalima uslovnih skokova i signalima višestrukih uslovnih skokova određuje da treba preći na sledeću mikroinstrukciju.

Bitovi polja *ba* mikroinstrukcije koriste se za specificiranje adrese mikroinstrukcije na koju treba skočiti kod bezuslovnih skokova i uslovnih skokova ukoliko odgovarajući signal uslova ima vrednost 1 u sekvenci upravljačkih signala po koracima (tabela 13). Ovim bitovima se predstavlja vrednost koju treba upisati u mikroprogramski brojač u slučaju bezuslovnih skokova i uslovnih skokova ukoliko odgovarajući signal uslova ima vrednost 1. 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.

Dužina mikroinstrukcije je 80 bitova. Za kodiranje operacionog dela mikroinstrukcije koristi se 68 bitova. Upravljačkih signala operacione jedinice ima 64, ali je umesto 64 bita usvojena dužina operacionog dela mikroinstrukcije 68 bitova, da bi se omogućio takav način pridruživanja bitova upravljačkim signalima operacione jedinice kojim se dobija pregledniji mikroprogram predstavjen u heksadecimalnom obliku. Za kodiranje polja *cc* upravljačkog dela mikroinstrukcije usvojena su 4 bita, jer je ukupan broj signala bezuslovnih skokova, uslovnih skokova i višestrukih uslovnih skokova 14. Za kodiranje polja *ba* upravljačkog dela mikroinstrukcije usvojeno je 8 bitova, jer je ukupan broj koraka u sekvenci upravljačkih signala po koracima 157.

Mikroprogram se formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima (tabela 13) formira jedna mikroinstrukcija. Operacioni deo mikroinstrukcije se formira ukoliko u datom koraku ima upravljačkih signala operacione jedinice. U suprotnom slučaju svi bitovi operacionog dela se postavljaju na vrednost 0. Upravljački deo mikroinstrukcije se formira ukoliko u datom koraku ima iskaza za bezuslovni skok, uslovni skok ili višestruki uslovni skok. U suprotnom slučaju svi bitovi upravljačkog dela se postavljaju na vrednost 0.

Kod formiranja operacionog dela mikroinstrukcije bitovi ovog dela koji odgovaraju upravljačkim signalima operacione koji se javljaju u datom koraku postavljaju se na 1, dok se bitovi ovog dela koji odgovaraju upravljačkim signalima operacione koji se ne javljaju u datom koraku postavljaju na 0.

Kod formiranja upravljačkog dela mikroinstrukcije za dati korak se proverava da li 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 korake u kojima se javljaju, bitovi polja *cc* i *ba* se kodiraju u zavisnosti od toga koji se od ova tri iskaza javlja u datom koraku.

Za iskaz *br* $step_A$ se upravljački deo mikroinstrukcije kodira tako što se za polje *cc* uzima kod dodeljen signalu bezuslovnog skoka koji određuje da se bezuslovno skače 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 21. Korak $step_i$ u kome se javlja bezuslovni skok, korak $step_A$ na koji treba preći, 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 24.

Tabela 24 Koraci $step_i$, $step_A$, vrednosti **$madr_A$** i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	**$madr_A$**	A
$step_{13}$	$step_{30}$	**$madr_{30}$**	30

$step_i$	$step_A$	**$madr_A$**	A
$step_{59}$	$step_{89}$	**$madr_{89}$**	89

$step_{14}$	$step_{30}$	**$madr_{30}$**	30
$step_{16}$	$step_{26}$	**$madr_{26}$**	26
$step_{18}$	$step_{26}$	**$madr_{26}$**	26
$step_{1F}$	$step_{26}$	**$madr_{26}$**	26
$step_{21}$	$step_{26}$	**$madr_{26}$**	26
$step_{24}$	$step_{26}$	**$madr_{26}$**	26
$step_{28}$	$step_{30}$	**$madr_{30}$**	30
$step_{2C}$	$step_{30}$	**$madr_{30}$**	30
$step_{2E}$	$step_{30}$	**$madr_{30}$**	30
$step_{31}$	$step_{89}$	**$madr_{89}$**	89
$step_{32}$	$step_{89}$	**$madr_{89}$**	89
$step_{34}$	$step_{89}$	**$madr_{89}$**	89
$step_{35}$	$step_{89}$	**$madr_{89}$**	89
$step_{39}$	$step_{89}$	**$madr_{89}$**	89
$step_{40}$	$step_{89}$	**$madr_{89}$**	89
$step_{41}$	$step_{89}$	**$madr_{89}$**	89
$step_{46}$	$step_{89}$	**$madr_{89}$**	89
$step_{4E}$	$step_{89}$	**$madr_{89}$**	89
$step_{52}$	$step_{89}$	**$madr_{89}$**	89

$step_{5A}$	$step_{89}$	**$madr_{89}$**	89
$step_{5B}$	$step_{89}$	**$madr_{89}$**	89
$step_{5C}$	$step_{89}$	**$madr_{89}$**	89
$step_{5D}$	$step_{89}$	**$madr_{89}$**	89
$step_{5F}$	$step_{89}$	**$madr_{89}$**	89
$step_{61}$	$step_{89}$	**$madr_{89}$**	89
$step_{63}$	$step_{89}$	**$madr_{89}$**	89
$step_{65}$	$step_{89}$	**$madr_{89}$**	89
$step_{67}$	$step_{89}$	**$madr_{89}$**	89
$step_{69}$	$step_{89}$	**$madr_{89}$**	89
$step_{6B}$	$step_{89}$	**$madr_{89}$**	89
$step_{6D}$	$step_{89}$	**$madr_{89}$**	89
$step_{6F}$	$step_{89}$	**$madr_{89}$**	89
$step_{71}$	$step_{89}$	**$madr_{89}$**	89
$step_{73}$	$step_{89}$	**$madr_{89}$**	89
$step_{74}$	$step_{89}$	**$madr_{89}$**	89
$step_{7B}$	$step_{89}$	**$madr_{89}$**	89
$step_{9C}$	$step_{00}$	**$madr_{00}$**	00

Za iskaz *br* (*if* **uslov** *then* $step_A$) se upravljački deo mikroinstrukcije kodira tako što se za polje *cc* uzima kod dodeljen signalu uslovnog skoka koji određuje signal **uslov** koji treba da ima vrednost 1 da bi se realizovao skok na korak $step_A$ i za polje *ba* binarna vrednosti A koju treba upisati u mikroprogramski brojač u slučaju da signal **uslov** ima vrednost 1. 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 22. Korak $step_i$ u kome se javlja uslovni skok, signal **uslov** čija se vrednost proverava, korak $step_A$ na koji treba preći u slučaju da signal **uslov** ima vrednost 1, 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 25.

Tabela 25 Koraci $step_i$, uslovi **uslov**, koraci $step_A$, vrednosti **$madr_A$** i vrednosti A za uslovne skokove

$step_i$	uslov	$step_A$	**$madr_A$**	A
$step_{00}$	$\overline{\textbf{START}}$	$step_{00}$	**$madr_{00}$**	00
$step_{04}$	**l1**	$step_{30}$	**$madr_{30}$**	30
$step_{07}$	**l2_brnch**	$step_{30}$	**$madr_{30}$**	30
$step_{08}$	**l2_arlog**	$step_{10}$	**$madr_{10}$**	10
$step_{0B}$	**l3_jump**	$step_{30}$	**$madr_{30}$**	30
$step_{0C}$	**l3_arlog**	$step_{10}$	**$madr_{10}$**	10
$step_{11}$	**store**	$step_{30}$	**$madr_{30}$**	30
$step_{12}$	**LDW**	$step_{14}$	**$madr_{14}$**	14
$step_{15}$	**store**	$step_{30}$	**$madr_{30}$**	30
$step_{17}$	**store**	$step_{30}$	**$madr_{30}$**	30

$step_i$	uslov	$step_A$	**$madr_A$**	A
$step_{1E}$	**store**	$step_{30}$	**$madr_{30}$**	30
$step_{20}$	**store**	$step_{30}$	**$madr_{30}$**	30
$step_{23}$	**store**	$step_{30}$	**$madr_{30}$**	30
$step_{25}$	**store**	$step_{30}$	**$madr_{30}$**	30
$step_{27}$	**LDW**	$step_{29}$	**$madr_{29}$**	29
$step_{2D}$	**LDW**	$step_{2F}$	**$madr_{2F}$**	2F
$step_{36}$	**dirreg**	$step_{3A}$	**$madr_{3A}$**	3A
$step_{3B}$	**dirreg**	$step_{41}$	**$madr_{41}$**	41
$step_{72}$	$\overline{\textbf{brpom}}$	$step_{89}$	**$madr_{89}$**	89
$step_{89}$	$\overline{\textbf{prekid}}$	$step_{00}$	**$madr_{00}$**	00

Za iskaz *br* (*case* (**$uslov_1$**, ..., **$uslov_n$**) *then* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$)) se upravljački deo mikroinstrukcije 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 ima vrednost 1 da bi se na osnovu toga realizovao skok na jedan od koraka $step_{A1}$, ..., $step_{An}$ i za polje *ba* 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 i obezbedi selekciju jedne od vrednosti A1,..., An u zavisnosti od toga koji od signala uslova **$uslov_1$**, ..., **$uslov_n$** ima vrednost 1.

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 23. Signali uslova $\mathbf{uslov_1}$, ..., $\mathbf{uslov_n}$ za koje treba izvršiti proveru koji je od njih ima vrednost 1, koraci $step_{A1}$, ..., $step_{An}$ na jedan od kojih se skače u zavisnosti od toga koji od signala uslova $\mathbf{uslov_1}$, ..., $\mathbf{uslov_n}$ ima vrednost 1 i vrednosti A1,..., An od kojih jednu treba upisati u mikroprogramski brojač za dva iskaza ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima dati su u tabelama 26 i 27.

Tabela 26 Signali uslova, koraci na koje se skače i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{10}$

uslov	$step_A$	A
regdir	$step_{11}$	11
regind	$step_{15}$	15
memdir	$step_{17}$	17
memind	$step_{19}$	19
regindpom	$step_{20}$	20
bxpom	$step_{22}$	22
pcpom	$step_{25}$	25
imm	$step_{2D}$	2D

Tabela 27 Signali uslova, koraci na koje se skače i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{30}$

uslov	$step_A$	A
INTD	$step_{31}$	31
INTE	$step_{32}$	32
LDB	$step_{33}$	33
LDW	$step_{35}$	35
STB	$step_{36}$	36
STW	$step_{3B}$	3B
POPB	$step_{42}$	42
POPW	$step_{47}$	47
PUSHB	$step_{4F}$	4F
PUSHW	$step_{53}$	53
LDIVTP	$step_{5A}$	5A
STIVTP	$step_{5B}$	5B
LDSP	$step_{5C}$	5C
STSP	$step_{5D}$	5D
ADD	$step_{5E}$	5E
SUB	$step_{60}$	60
INC	$step_{62}$	62
DEC	$step_{64}$	64
AND	$step_{66}$	66
OR	$step_{68}$	68
XOR	$step_{6A}$	6A
NOT	$step_{6C}$	6C
ASR	$step_{6E}$	6E
LSR	$step_{6E}$	6E
ROR	$step_{6E}$	6E
RORC	$step_{6E}$	6E
ASL	$step_{70}$	70

uslov	$step_A$	A
LSL	$step_{70}$	70
ROL	$step_{70}$	70
ROLC	$step_{70}$	70
BEQL	$step_{72}$	72
BNEQ	$step_{72}$	72
BNEG	$step_{72}$	72
BNNEG	$step_{72}$	72
BOVF	$step_{72}$	72
BNOVF	$step_{72}$	72
BCAR	$step_{72}$	72
BNCAR	$step_{72}$	72
BGRT	$step_{72}$	72
BGRE	$step_{72}$	72
BLSS	$step_{72}$	72
BLSSE	$step_{72}$	72
BGRT	$step_{72}$	72
BGRE	$step_{72}$	72
BLSS	$step_{72}$	72
BLSSE	$step_{72}$	72
BGRTU	$step_{72}$	72
BGRTEU	$step_{72}$	72
BLSSU	$step_{72}$	72
BLSSEU	$step_{72}$	72
JMP	$step_{74}$	74
JSR	$step_{75}$	75
RTI	$step_{7C}$	7C
RTS	$step_{82}$	82

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu mikroprogramske realizacije signal bezuslovnog skoka (tabela 21), signali uslovnih skokova (tabela 22), signali višestrukih uslovnih skokova (23) i signali vrednosti A za bezuslovne skokokve (tabela 24), uslovne skokove (tabela 25) i višestruke uslovne skokove (tabele 26 i 27).

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 13) formiran mikroprogram (tabela 28). On ima sledeću formu:

- na levoj strani su adrese mikroinstrukcija u mikroprogramskoj memoriji predstavljene u heksadekadnom obliku,
- u sredini su mikroinstrukcije predstavljene u heksadekadnom obliku i
- na desnoj strani je komentar koji počinje uskličnikom (!) i proteže se do sledećeg uskličnika (!) i koji se sastoji od simboličkih oznaka samo upravljačkih signala

operacione i/ili upravljačke jedinice razdvojenih zapetama koji u datom koraku imaju vrednost 1 .

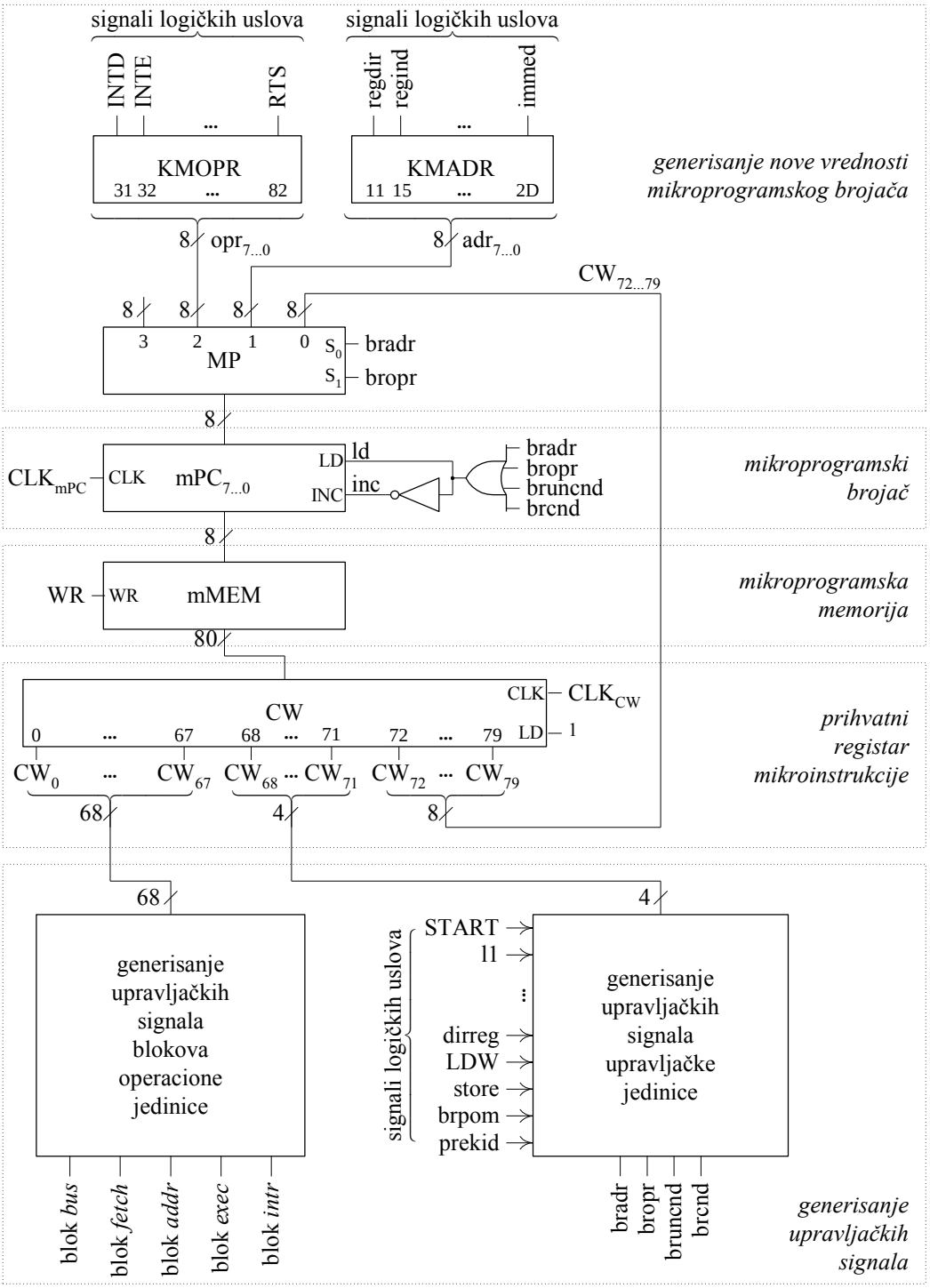
Tabela 28 Mikroprogram za upravljačku jedinicu mikroprogramske realizacije

! Čitanje instrukcije !
00 00000000000000000200 **!brnotSTART, maddr$_{00}$!**
01 00808000000000000000 **!ldMAR, incPC!**
02 00020000000000000000 **!rdCPU!**
03 00000800000000000000 **!ldIR0!**
04 00000000000000000330 **!brl1, maddr$_{30}$!**
05 00808000000000000000 **!ldMAR, incPC!**
06 00020000000000000000 **!rdCPU!**
07 00000408000000000430 **!ldIR1, ldGPRADR, brl2_brnch, maddr$_{30}$!**
08 00000000000000000510 **!brl2_arlog, maddr$_{10}$!**
09 00808000000000000000 **!ldMAR, incPC!**
0A 00020000000000000000 **!rdCPU!**
0B 00000200000000000630 **!ldIR2, brl3_jump, maddr$_{30}$!**
0C 00000000000000000710 **!brl3_arlog, maddr$_{10}$!**
0D 00808000000000000000 **!ldMAR, incPC!**
0E 00020000000000000000 **!rdCPU!**
0F 00000100000000000000 **!ldIR3!**
! Formiranje adrese i čitanje operanda !
10 00000000000000000D00 **!bradr!**
! Registarsko direktno adresiranje !
11 00000000000000000830 **!brstore, maddr$_{30}$!**
12 00000000000000000914 **!brLDW, maddr$_{14}$!**
13 00000000000004000130 **!ldBB, bruncnd, maddr$_{30}$**
14 00000000000000400130 **!ldBW, bruncnd, maddr$_{30}$!**
! Registarsko indirektno adresiranje !
15 10800000000000000830 **!mxMAR$_0$, ldMAR, brstore, maddr$_{30}$!**
16 00000000000000000126 **!bruncnd, maddr$_{26}$!**
! Memorijsko direktno adresiranje !
17 20800000000000000830 **!mxMAR$_1$, ldMAR, brstore, maddr$_{30}$!**
18 00000000000000000126 **!bruncnd, maddr$_{26}$!**
! Memorijsko indirektno adresiranje !
19 20800000000000000000 **!mxMAR$_1$, ldMAR!**
1A 00020000000000000000 **!rdCPU!**
1B 00600000000000000000 **!ldDWH, incMAR!**
1C 00020000000000000000 **!rdCPU!**
1D 00100000000000000000 **!ldDWL!**
1E 30800000000000000830 **!mxMAR$_1$, mxMAR$_0$, ldMAR, brstore, maddr$_{30}$!**
1F 00000000000000000126 **!bruncnd, maddr$_{26}$!**
! Registarsko indirektno adresiranje sa pomerajem !
20 40800001100000000830 !mxADDA$_0$, mxADDB$_0$, mxMAR$_2$, ldMAR, brstore, madr$_{30}$!
21 00000000000000000126 **!bruncnd, maddr$_{26}$!**
! Bazno indeksno adresiranje sa pomerajem !
22 00000045100000000000 **!mxADDA$_0$, mxADDB$_0$, ldCW, incGPRAR!**
23 40800003200000000830 !mxADDA$_1$, mxADDA$_0$,mxADDB$_1$,mxMAR$_2$,ldMAR, brstore, madr$_{30}$!
24 00000000000000000126 **!bruncnd, madr$_{26}$!**
! PC relativno adresiranje !
25 40800002100000000830 !mxADDA$_1$, mxADDB$_0$, mxMAR$_2$, ldMAR, brstore, madr$_{30}$!
! Čitanje operanda !
26 00020000000000000000 **!rdCPU!**
27 00000000000000000929 **!brLDW, maddr$_{29}$!**
28 00000000000050000130 **!mxBB$_0$, ldBB, bruncnd, maddr$_{30}$!**
29 00600000000000000000 **!ldDWH, incMAR!**
2A 00020000000000000000 **!rdCPU!**
2B 00100000000000000000 **!ldDWL!**
2C 00000000000005000130 !mxBW$_0$, ldBW, bruncnd, maddr$_{30}$!
! Neposredno adresiranje !
2D 00000000000000000920 **!brLDW, maddr$_{20}$!**
2E 00000000000060000130 **!mxBB$_1$, ldBB, bruncnd, maddr$_{30}$!**
2F 00000000000006000000 **!mxBW$_1$, ldBW!**
! Izvršavanje operacije !
30 00000000000000000E00 **!bropr !**
! INTD !
31 00000000000000008189 **!clPSWI, bruncnd, maddr$_{89}$!**
! INTE !
32 00000000000000004189 !stPSWI, bruncnd, maddr$_{89}$!

```
! LDB !
33  00000000000300000000 !mxAB, ldAB!
34  00000000000000F0189 !ldN, ldZ, ldC, ldV, bruncnd, maddr89!
! LDW !
35  00000000000004000189 !ldAW, bruncnd, maddr89!
! STB !
36  0000000000000000A3A !brdirreg, maddr3A!
37  01080000000000000000 !mxMDR0, ldMDR!
38  00010000000000000000 !wrCPU!
39  00000000000000000189 !bruncnd, maddr89!
3A  00000010000000000189 !wrGPR, bruncnd, maddr89!
! STW !
3B  0000000000000000A41 !brdirreg, maddr41!
3C  02080000000000000000 !mxMDR1, ldMDR!
3D  00010000000000000000 !wrCPU!
3E  03480000000000000000 !mxMDR1, mxMDR0, ldMDR, incMAR!
3F  00010000000000000000 !wrCPU!
40  00000000000000000189 !bruncnd, maddr89!
41  00000030000000000189 !mxGPR, wrGPR, bruncnd, maddr89!
! POPB !
42  50800000400000000000 !mxMAR7, mxMAR0, ldMAR, decSP!
43  00020000000000000000 !rdCPU!
44  00000000000050000000 !mxBB0, ldBB!
45  00000000000300000000 !mxAB, ldAB!
46  00000000000000F0189 !ldN, ldZ, ldC, ldV, bruncnd, maddr89!
! POPW !
47  50800000400000000000 !mxMAR7, mxMAR0, ldMAR, decSP!
48  00020000000000000000 !rdCPU!
49  00100000000000000000 !ldDWL!
4A  50800000400000000000 !mxMAR7, mxMAR0, ldMAR, decSP!
4B  00020000000000000000 !rdCPU!
4C  00200000000000000000 !ldDWH!
4D  00000000000000500000 !mxBW0, ldBW!
4E  00000000000004000189 !ldAW, bruncnd, maddr89!
! PUSHB !
4F  00000000800000000000 !incSP!
50  51880000000000000000 !mxMAR7, mxMAR0, ldMAR, mxMDR0, ldMDR!
51  00010000000000000000 !wrCPU!
52  00000000000000000189 !bruncnd, maddr89!
! PUSHW !
53  00000000800000000000 !incSP!
54  52880000000000000000 !mxMAR7, mxMAR0, ldMAR, mxMDR1, ldMDR!
55  00010000000000000000 !wrCPU!
56  00000000800000000000 !incSP!
57  53880000000000000000 !mxMAR7, mxMAR0, ldMAR, mxMDR1, mxMDR0, ldMDR!
58  00010000000000000000 !wrCPU!
59  00000000000000000189 !bruncnd, maddr89!
! LDIVTP !
5A  00000000000006000189 !mxAW1, ldAW, bruncnd,maddr89!
! STIVTP !
5B  00000000000080000189 !ldIVTP, bruncnd, maddr89!
! LDSP !
5C  00000000000005000189 !mxAW0, ldAW, bruncnd, maddr89!
! STSP !
5D  00000080000000000189 !ldSP, bruncnd, maddr89!
! ADD !
5E  00000000010100030000 !add, ldAB, ldC, ldV!
5F  0000000000000000C0189 !ldN, ldZ, bruncnd, maddr89!
! SUB !
60  00000000020100030000 !sub, ldAB, ldC, ldV!
61  0000000000000000C0189 !ldN, ldZ, bruncnd, maddr89!
! INC !
62  00000000040100030000 !inc, ldAB, ldC, ldV!
63  0000000000000000C0189 !ldN, ldZ, bruncnd, maddr89!
! DEC !
64  00000000080100030000 !dec, ldAB, ldC, ldV!
65  0000000000000000C0189 !ldN, ldZ, bruncnd, maddr89!
! AND !
66  00000000001100000000 !and, ldAB!
67  0000000000000000F0189 !ldN, ldZ, ldC, ldV, bruncnd, maddr89!
```

! OR !
68 00000000002100000000 **!or**, **ldAB**!
69 0000000000000000F0189 !ldN, ldZ, ldC, ldV, bruncnd, maddr$_{89}$!
! XOR !
6A 00000000004100000000 **!xor**, **ldAB**!
6B 0000000000000000F0189 !ldN, ldZ, ldC, ldV, bruncnd, maddr$_{89}$!
! NOT !
6C 00000000008100000000 **!not**, **ldAB**!
6D 0000000000000000F0189 !ldN, ldZ, ldC, ldV, bruncnd, maddr$_{89}$!
! ASR, LSR, ROR i ROLC !
6E 00000000000800020000 **!shr**, **ldC**!
6F 0000000000000000D0189 !ldN, ldZ, ldV, bruncnd, maddr$_{89}$!
! ASL, LSL, ROL i ROLC !
70 00000000000400020000 **!shl**, **ldC**!
71 0000000000000000D0189 !ldN, ldZ, ldV, bruncnd, maddr$_{89}$!
! BEQL,…, BLSSEU !
72 0000000000000000B089 **!brnotbrpom**, **maddr$_{89}$**!
73 0000500230000000189 **!mxADDA$_1$**,**mxADDB$_1$**,**mxADDB$_0$**, **mxPC$_0$**, **ldPC**, **bruncnd**, **maddr$_{89}$**!
! JMP !
74 0000600000000000189 **!mxPC$_1$**, **ldPC**, **bruncnd**, **maddr$_{89}$**!
! JSR !
75 00000000800000000000 **!incSP**!
76 54880000000000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **ldMDR**!
77 00010000000000000000 **!wrCPU**!
78 00000000800000000000 **!incSP**!
79 55880000000000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **mxMDR$_0$**, **ldMDR**!
7A 00010000000000000000 **!wrCPU!**
7B 0000600000000000189 !mxPC$_1$, ldPC, bruncnd, maddr$_{89}$!
! RTI !
7C 50800000400000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**!
7D 00020000000000000000 **!rdCPU**!
7E 00000000000000001000 **!ldPSWL**!
7F 50800000400000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**!
80 00020000000000000000 **!rdCPU**!
81 00000000000000002000 **!ldPSWH**!
! RTS !
82 50800000400000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**!
83 00020000000000000000 **!rdCPU**!
84 00100000000000000000 **!ldDWL**!
85 50800000400000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **decSP**!
86 00020000000000000000 **!rdCPU**!
87 00200000000000000000 **!ldDWH**!
88 00004000000000000000 **!ldPC**!
! Opsluživanje prekida !
89 0000000000000000C00 **!brnotprekid**, **maddr$_{00}$**!
! Čuvanje konteksta procesora !
8A 00000000800000000000 **!incSP**!
8B 54880000000000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **ldMDR**!
8C 00010000000000000000 **!wrCPU**!
8D 00000000800000000000 **!incSP**!
8E 55880000000000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **mxMDR$_0$**, **ldMDR**!
8F 00010000000000000000 **!wrCPU**!
90 00000000800000000000 !incSP!
91 56880000000000000000 **!mxMAR$_2$**, **mxMAR$_0$**, **ldMAR**, **mxMDR$_2$**, **mxMDR$_1$**, **ldMDR**!
92 00010000000000000000 **!wrCPU**!
93 00000000800000000000 **!incSP**!
94 57880000000000000000 **!mxMAR$_2$**,**mxMAR$_0$**,**ldMAR**,**mxMDR$_2$**,**mxMDR$_1$**,**mxMDR$_0$**,**ldMDR**!
95 00010000000000000000 **!wrCPU**!
! Utvrđivanje broja ulaza !
96 00000000000000800000 **!ldBR**!
! Utvrđivanje adrese prekidne rutine !
97 40800000000000000000 **!mxMAR$_2$**, **ldMAR**!
98 00020000000000000000 **!rdCPU**!
99 00600000000000000000 **!ldDWH**, **incMAR**!
9A 00020000000000000000 **!rdCPU**!
9B 00100000000000000000 **!ldDWL**!
9C 00004000000000000100 **!ldPC**, **bruncnd**, **maddr$_{00}$**!

Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 51. 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*. Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.



Slika 51 Struktura upravljačke jedinice mikroprogramske realizacije

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 $mPC_{7...0}$. 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 $CW_{72...79}$ prihvatnog registra mikroinstrukcije $CW_{0...79}$. 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 signal **bropr** ima vrednost 1, signali $\mathbf{adr_{7...0}}$ ako signal **bradr** ima vrednost 1 i signali $CW_{72...79}$ ako oba signala **bropr** i **bradr** imaju vrednost 0.

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 27) za realizaciju višestrukog uslovnog skoka na adresi 30 mikroprograma (tabela 28). U zavisnosti od toga koji od signala **INTD**, **INTE**, …, **RTS** ima vrednost 1 zavisi koja će od vrednosti iz tabele 27 da se pojavi na linijama $\mathbf{opr_{7...0}}$. S obzirom da se na adresi 30 mikroprograma nalazi mikroinstrukcija sa tako kodiranim poljem *cc* da njeno izvršavanje daje vrednost 1 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 26) za realizaciju višestrukog uslovnog skoka na adresi 10 mikroprograma (tabela 28). U zavisnosti od toga koji od signala **dirreg**, **indreg**,..., **immed** ima vrednost 1 zavisi koja će od vrednosti iz tabele 26 da se pojavi tada na linijama $\mathbf{adr_{7...0}}$. S obzirom da se na adresi 10 mikroprograma nalazi mikroinstrukcija sa tako kodiranim poljem *cc* da njeno izvršavanje daje vrednost 1 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 $CW_{0...79}$ u svojim razredima $CW_{72...79}$ sadrži vrednost za upis u mikroprogramski brojač $mPC_{7...0}$ za bezuslovne skokove (tabela 24) i uslovne skokove (tabela 25) u mikroprogramu (tabela 28). Signali višestrukih uslovnin skokova **bropr** i **bradr** imaju vrednost 1 samo prilikom izvršavanja mikroinstrukcija na adresama 30 i 10 mikroprograma, respektivno, a u svim ostalim situacijama imaju vrednost 0. S obzirom da nijedan od ova dva signala nema vrednost 1 prilikom izvršavanja mikroinstrukcija kojima se realizuju bezuslovni ili neki od uslovnih skokova u mikroprogramu, vrednost određena razredima $CW_{72...79}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača $mPC_{7...0}$.

Blok *mikroprogramski brojač* sadrži mikroprogramski brojač $mPC_{7...0}$. Mikroprogramski brojač $mPC_{7...0}$ svojom trenutnom vrednošću određuje adresu mikroprogramske memorije mMEM sa koje treba očitati mikroinstrukciju. Mikroprogramski brojač $mPC_{7...0}$ 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 $mPC_{7...0}$ za jedan čime se obezbeđuje sekvencijalno očitavanje mikroinstrukcija iz mikroprogramske memorije (tabela 28). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko svi signali **bropr**, **bradr**, **brcnd** i **bruncnd** imaju vrednost 0. Signali **bropr**, **bradr**, **brcnd** i **bruncnd** normalno imaju vrednost 0 sem prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran neki višestrukih uslovnih skokova, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

U režimu skoka pri pojavi signala takta $\mathbf{CLK_{mPC}}$ vrši se upis nove vrednosti u mikroprogramski brojač $mPC_{7...0}$ čime se obezbeđuje odstupanje od sekvencijalnog očitavanja mikroinstrukcija iz mikroprogramske memorije (tabela 28). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ukoliko jedan od signala **bropr**, **bradr**, **brcnd** i **bruncnd** ima vrednost 1. Signali **bropr**, **bradr**, **brcnd** i **bruncnd** normalno imaju

vrednost 0 sem prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran neki višestrukih uslovnih skokova, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

Mikroprogramski brojač $mPC_{7...0}$ je dimenzionisan prema veličini mikroprograma (tabela 28). S obzirom da se mikroprogram svih faza izvršavanja instrukcija nalazi u opsegu od adrese 00 do adrese 9C, usvojena je dužina mikroprogramskog brojača $mPC_{7...0}$ od 8 bita.

Blok *mikroprogramska memorija* sadrži mikroprogramsku memoriju mMEM, koja služi za smeštanje mikroprograma. Širina reči mikroprogramske memorije je određena dužinom mikroinstrukcija i iznosi 80 bita, a kapacitet veličinom mikroprograma svih instrukcija procesora (tabela 28) i iznosi 256 lokacija. Adresiranje mikroprogramske memorije se realizuje sadržajem mikroprogramskog brojača $mPC_{7...0}$.

Blok *prihvatni registar mikroinstrukcije* sadrži prihvatni registar mikroinstrukcije $CW_{0...79}$. Prihvatni registar mikroinstrukcije $CW_{0...79}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...67}$ i $CW_{68...71}$ se koriste u bloku *generisanje upravljačkih signala* za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{72...79}$ 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 $CW_{0...67}$ prihvatnog registra mikroinstrukcije generišu upravljačke signale operacione jedinice i na osnovu sadržaja razreda $CW_{68...71}$ prihvatnog registra mikroinstrukcije i signala logičkih uslova **l1**, **l2**, ..., **prekid** koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali blokova operacione jedinice ***oper*** se daju posebno za svaki blok.

Upravljački signali bloka *bus* se generišu na sledeći način:

- $\mathbf{mxMAR_2} = \mathbf{CW_1}$
- $\mathbf{mxMAR_1} = \mathbf{CW_2}$
- $\mathbf{mxMAR_0} = \mathbf{CW_3}$
- $\mathbf{ldMAR} = \mathbf{CW_8}$
- $\mathbf{incMAR} = \mathbf{CW_9}$
- $\mathbf{mxMDR_2} = \mathbf{CW_5}$
- $\mathbf{mxMDR_1} = \mathbf{CW_6}$
- $\mathbf{mxMDR_0} = \mathbf{CW_7}$
- $\mathbf{ldMDR} = \mathbf{CW_{12}}$
- $\mathbf{rdCPU} = \mathbf{CW_{14}}$
- $\mathbf{wrCPU} = \mathbf{CW_{15}}$
- $\mathbf{ldDWH} = \mathbf{CW_{10}}$
- $\mathbf{ldDWL} = \mathbf{CW_{11}}$

Upravljački signali bloka *fetch* se generišu na sledeći način:

- $\mathbf{mxPC_1} = \mathbf{CW_{18}}$
- $\mathbf{mxPC_0} = \mathbf{CW_{19}}$
- $\mathbf{ldPC} = \mathbf{CW_{17}}$

- **incPC** = **CW_{16}**
- **ldIR0** = **CW_{20}**
- **ldIR1** = **CW_{21}**
- **ldIR2** = **CW_{22}**
- **ldIR3** = **CW_{23}**

Upravljački signali bloka *addr* se generišu na sledeći način:
- **ldGPRAR** = **CW_{28}**
- **incGPRAR** = **CW_{29}**
- **mxGPR** = **CW_{26}**
- **wrGPR** = **CW_{27}**
- **ldSP** = **CW_{24}**
- **incSP** = **CW_{32}**
- **decSP** = **CW_{33}**
- **$mxADDA_1$** = **CW_{30}**
- **$mxADDA_0$** = **CW_{31}**
- **$mxADDB_1$** = **CW_{34}**
- **$mxADDB_0$** = **CW_{35}**
- **ldCW** = **CW_{25}**

Upravljački signali bloka *exec* se generišu na sledeći način:
- **mxAB** = **CW_{46}**
- **ldAB** = **CW_{47}**
- **shr** = **CW_{44}**
- **shl** = **CW_{45}**
- **$mxBB_1$** = **CW_{50}**
- **$mxBB_0$** = **CW_{51}**
- **ldBB** = **CW_{49}**
- **$mxAW_1$** = **CW_{54}**
- **$mxAW_0$** = **CW_{55}**
- **ldAW** = **CW_{53}**
- **$mxBW_1$** = **CW_{58}**
- **$mxBW_0$** = **CW_{59}**
- **ldBW** = **CW_{57}**
- **stPSWI** = **CW_{65}**
- **clPSWI** = **CW_{64}**
- **ldN** = **CW_{60}**
- **ldZ** = **CW_{61}**
- **ldC** = **CW_{62}**
- **ldV** = **CW_{63}**
- **ldPSWL** = **CW_{67}**
- **ldPSWH** = **CW_{66}**
- **add** = **CW_{39}**
- **sub** = **CW_{38}**
- **inc** = **CW_{37}**
- **dec** = **CW_{36}**
- **and** = **CW_{43}**
- **or** = **CW_{42}**

- **xor** = $\mathbf{CW_{41}}$
- **not** = $\mathbf{CW_{40}}$

Upravljački signali bloka *intr* se generišu na sledeći način:

- **ldIVTP** = $\mathbf{CW_{48}}$
- **ldBR** = $\mathbf{CW_{56}}$

Upravljački signali upravljačke jedinice ***uprav*** se generišu na sledeći način: $\mathbf{CW_{68} \cdot CW_{69} \cdot CW_{70} \cdot CW_{71}}$

- **bradr** = $\mathbf{CW_{68} \cdot CW_{69} \cdot \overline{CW_{70}} \cdot CW_{71}}$
- **bropr** = $\mathbf{CW_{68} \cdot CW_{69} \cdot CW_{70} \cdot \overline{CW_{71}}}$
- **bruncnd** = $\mathbf{\overline{CW_{68}} \cdot \overline{CW_{69}} \cdot \overline{CW_{70}} \cdot CW_{71}}$
- **brcnd** = **brnotSTART**$\cdot \overline{\textbf{START}}$ + **brl1**·**l1** + **brl2_brnch**· **l2_brnch** + **brl2_arlog**· **l2_arlog** + **brl3_jump**· **l3_jump** + **brl3_arlog**· **l3_arlog** + **brstore**· **store** + **brLDW**· **LDW** + **brdirreg**· **dirreg** + **brnotbrpom** $\cdot \overline{\textbf{brpom}}$
- **brnotSTART** = $\mathbf{\overline{CW_{68}} \cdot \overline{CW_{69}} \cdot CW_{70} \cdot \overline{CW_{71}}}$
- **brl1**= $\mathbf{\overline{CW_{68}} \cdot \overline{CW_{69}} \cdot CW_{70} \cdot CW_{71}}$
- **brl2_brnch**= $\mathbf{\overline{CW_{68}} \cdot CW_{69} \cdot \overline{CW_{70}} \cdot \overline{CW_{71}}}$
- **brl2_arlog**= $\mathbf{\overline{CW_{68}} \cdot CW_{69} \cdot \overline{CW_{70}} \cdot CW_{71}}$
- **brl3_jump**= $\mathbf{\overline{CW_{68}} \cdot CW_{69} \cdot CW_{70} \cdot \overline{CW_{71}}}$
- **brl3_arlog**= $\mathbf{\overline{CW_{68}} \cdot CW_{69} \cdot CW_{70} \cdot CW_{71}}$
- **brstore**= $\mathbf{CW_{68} \cdot \overline{CW_{69}} \cdot \overline{CW_{70}} \cdot \overline{CW_{71}}}$
- **brLDW**= $\mathbf{CW_{68} \cdot \overline{CW_{69}} \cdot \overline{CW_{70}} \cdot CW_{71}}$
- **brdirreg**= $\mathbf{CW_{68} \cdot \overline{CW_{69}} \cdot CW_{70} \cdot \overline{CW_{71}}}$
- **brnotpom**= $\mathbf{CW_{68} \cdot \overline{CW_{69}} \cdot CW_{70} \cdot CW_{71}}$
- **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 blokova operacione jedinice ***oper*** i to:

- **START** — blok *exec*,
- **l1** — blok *fetch*,
- **l2_brnch** — blok *fetch*,
- **l2_arlog** — blok *fetch*,
- **l3_jump** — blok *fetch*,
- **l3_arlog** — blok *fetch*,
- **store** — blok *fetch*,
- **LDW** — blok *fetch*,
- **dirreg** — blok *fetch*,
- **brpom** — blok *exec*
- **prekid** — blok *intr*

4 LITERATURA

1. Lazić, B., *Logičko projektovanje računara*, Nauka—Elektrotehnički fakultet, Beograd, Srbija, Jugoslavija, **1994**

2. Živković, D., Popović, D., *Impulsna i digitalna elektronika*, Nauka—Elektrotehnički fakultet, Beograd, Srbija, Jugoslavija, **1992**.

3. Aleksić, T., *Računari—organizacija i arhitektura*, Naučna knjiga, Beograd, Srbija, Jugoslavija, **1985**.

4. Stallings, W., *Computer Organization and Architecture*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, **1996**.

5. Patterson, D., Hennessy, J., Goldberg, D., *Computer Architecture—A Quantitative Approach*, Morgan Kaufmann Publishers Inc., San Francisco, California, USA, **1996**.

6. Flynn, M., *Computer Architecture: Pipelined and Parallel Processor Design*, Jones and Bartlett, USA, **1995**

7. J. Djordjevic, A. Milenkovic, N. Grbanovic, "*An Integrated Educational Environment for Teaching Computer Architecture and Organisation*," IEEE MICRO, May 2000.pp. 66-74.

8. J. Djordjevic, M. Bojovic, A. Milenković, *An Integrated Educational Environment for Computer Architecture and Organisation*, Proceedings of the Symposium on Education and Employment, France, September, 1998.

9. J. Djordjevic, A. Milenkovic, S. Prodanovic "*A Hierarchical Memory System Environment*," IEEE TC Computer Architecture Newsletter, March 1999.

10. J. Đorđević, B. Nikolić, *Vizuelni simulator edukacionog računara*, Zbornik radova IT 2001, Žabljak, Jugoslavija, Mart 2001.

11. J. Djordjevic, R. N. Ibbett, M. R. Barbacci, *Evaluation of computer architectures using ISPS*, Proc. of IEE, Vol. 127, Pt. E. No. 4, pp. 126-131, July 1980.

12. J. Djordjevic, M. R. Barbacci, B. Hosler, *A PMS Level Notation for the Description and Simulation of Digital Systems*, The Computer Journal, Vol. 28, No. 4, pp. 357-365, 1985.

13. S. Miladinović, J. Đorđević, A. Milenković, *Programski sistem za grafički opis i simulaciju digitalnih sistema*, Zbornik radova ETRAN 1997, Zlatibor, Jugoslavija, Jun 1997.

14. N. Grbanovic, J. Djordjevic, B. Nikolić, *The Software Package Of An Educational Computer System*, prijavljen za objavljivanje u IJEEE, England, October, 2002.

15. J. Đorđević, B. Nikolić, *Neki Aspekti Realizacije Vizuelnog Simulatora Edukacionog Računara, ,* Zbornik radova IT 2001, Žabljak, Jugoslavija, Mart 2001.

16. J. Đorđević, T. Borozan, B. Nikolić, *Softversko okruženje za simulaciju računara*, Zbornik radova ETRAN 2001, Bukovička Banja, Jugoslavija, Jun 2001.

17. J. Djordjevic, A. Milenkovic, I. Todorovic, D. Marinov, "CALCAS: *A Computer Architecture Learning and Knowledge Assessment System*," IEEE TC Computer Architecture Newsletter, March 1999.

18. A. Milenkovic, Nikolić, B., J. Djordjevic, "CASTLE, *Computer Architecture Self-Testing and Learning System*," WCAE 2002, Workshop on Computer Architecture Education, Anchorage, Alaska, May 26, 2002.

19. Đorđević, J., *Priručnik iz arhitekture računara*, Elektrotehnički fakultet, Beograd, **1997**.

20. Đorđević, J., *Priručnik iz arhitekture i organizacije računara*, Elektrotehnički fakultet, Beograd, **1997**.

21. Đorđević, J., Grbanović, N., Nikolić, B., *Arhitektura računara, Edukacioni računarski sistem, Priručnik za simulaciju sa zadacima,* Elektrotehnički fakultet, Beograd, **2002**.

1
2