JOVAN ĐORĐEVIĆ

ARHITEKTURA I ORGANIZACIJA RAČUNARA

ZBIRKA REŠENIH ZADATAKA

B, 2011.

PREDGOVOR

Knjiga sadrži rešene zadatke iz predmeta Arhitektura i organizacija računara. Zadaci su grupisani po oblastima i to izvršavanje instrukcija, ...

Knjiga je napisana u veoma kratkom vremenskom periodu, pa su, i pored izvršenih provera, moguće određene greške. Autor će biti zahvalan svima onima koji budu ukazili na otkrivene greške.

Autor

Beograd

29.11.2011.

SADRŽAJ

1 ZADATAK 1

Posmatra se blok *fetch* procesora u kome se realizuje faza čitanja instrukcije. Procesor je tako realizovan da se faze čitanje instrukcije, formiranje adrese i čitanje operanda, izvršavanje operacije i opsluživanje prekida realizuju u posebnim blokovima. Pored procesora u posmatranim računaru postoji i memorija. Memorija je kapaciteta 2^{16} bajtova. Širina memorijske reči je 1 bajt. Procesor je sa jednoadresnim formatom instrukcija. Podaci su celobrojne veličine bez znaka dužine jedan bajt.

U bloku *fetch* postoji registar programskog brojača PC dužine dva bajta, adresni registar memorije MAR dužine dva bajta, prihvatni registar podatka memorije MBR dužine jedan bajt, prihvatni registar instrukcije IR dužine četiri bajta. Instrukcije su dužine jedan, dva, tri ili četiri bajta.

Prvi bajt instrukcije sadrži polje koda operacije.

U procesoru postije bezadresne instrukcije, instrukcije uslovnog skoka, instrukcije bezuslovnog skoka i adresne instrukcije.

Bezadresne instrukcije su instrukcija stavljanja sadržaja akumulatora na stek (PUSH), instrukcija skidanja sadržaja sa steka u akumulator (POP), instrukcija povratka iz potprograma (RTS), instrukcija povratka iz prekidne rutine (RTI) i instrukcija aritmetičkog pomeranja sadržaja akumulatora udesno za jedno mesto (ASR). Za bezadresne instrukcije PUSH, POP, RTS i RTI i ASR su usvojeni kodovi operacija 00000000, 00000001, 00000010, 0000000011 i 00000100, respektivno. Dužina instrukcija je jedan bajt.

Instrukcija uslovnog skoka je instrukcija uslovnog skoka ukoliko je rezultat nula (BZ). Za instrukciju BZ je usvojen kod operacije 00000101. Instrukcija BZ se realizuje kao relativni skok u odnosu na tekuću vrednost programskog brojača PC, a pomeraj je 8-mo bitna celobrojna veličina sa znakom data drugim bajtom instrukcije. Dužina instrukcije je dva bajta.

Instrukcije bezuslovnog skoka su instrukcija bezuslovnog skoka (JMP) i instrukcija skoka na potprogram (JSR). Za instrukcije JMP i JSR su usvojeni kodovi operacija 00000110 i 00000111, respektivno. Instrukcije JMP i JSR se realizuju kao apsolutni skokovi, a adresa skoka je data drugim i trećim bajtom instrukcije, pri čemu je stariji bajt adrese skoka dat drugim bajtom instrukcije a mlađi bajt adrese skoka trećim bajtom instrukcije. Dužina instrukcija je tri bajta.

Adresne instrukcije su instrukcija prenosa u akumulator (LD), instrukcija prenosa iz akumulatora (ST), aritmetička instrukcija sabiranja (ADD), logička instrukcija logički proizvod (AND) i instrukcija bezuslovnog skoka na sračunatu adresu (JADR). U instrukciji ST nije dozvoljeno neposredno adresiranje, a u instrukciji JADR nije dozvoljeno direktno registarsko i neposredno adresiranje, pa ukoliko se jave ova adresiranja u ovim instrukcijama, instrukcije treba da budu bez dejstva. Za instrukcije LD, ST, ADD, AND, ASR i JADR usvojeni kodovi operacija 00001000, 00001001, 00001010, 00001011, 00001100 i 00001101, respektivno. Dužina instrukcija je dva, tri ili četiri bajta i zavisi od specificiranog načina adresiranja.

Za adresne instrukcije se bitovima 7, 6 i 5 drugog bajta instrukcije specificira načina adresiranja i to na sledeći način: 000-registarsko direktno adresiranje (regdir), 001-registarsko indirektno adresiranje (regind), 010-memorijsko direktno adresiranje (memdir), 011-memorijsko indirektno adresiranje (memind), 100-registarsko indirektno sa pomerajem

adresiranje (regindpom), 101-bazno indeksno sa pomerajem adresiranje (bxpom), 110- PC relativno sa pomerajem adresiranje (pcrel) i 111-neposredno adresiranje (immed). Registarsko direktno i registarsko indirektno adresiranje koriste neki od registara opšte namene R[0] i R[31] specificiran bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcija je dva bajta. Neposredno adresiranje ima i treći bajt instrukcije u kome se nalazi operand, ali ne koristi registre opšte namene R[0] i R[31] specificirane bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcije je tri bajta.

Memorijsko direktno, memorijsko indirektno, registarsko indirektno sa pomerajem adresiranje, bazno indeksno sa pomerajem adresiranje i PC relativno sa pomerajem adresiranje imaju treći i četvrti bajt instrukcije. Kod memorijskog direktnog i memorijskog indirektnog adresiranja treći i četvrti bajt instrukcije sadrže adresu memorijske lokacije, pri čemu je stariji bajt adrese dat trećim bajtom a mlađi bajt adrese četvrtim bajtom. Kod memorijskog indirektnog adresiranja adresa dužine 16 bita zauzima dve susedne memorijske lokacije, pri čemu je stariji bajt adrese nalazi na nižoj a mlađi bajt adrese na višoj lokaciji. Bitovi 4 do 0 drugog bajta instrukcije se ne koriste. Dužina instrukcija je četiri bajta. Kod registarskog indirektnog sa pomerajem adresiranje, bazno indeksnog sa pomerajem adresiranjem i kod PC relativnog adresiranja treći i četvrti bajt instrukcije sadrže pomeraj, pri čemu je stariji bajt pomeraja dat trećim bajtom a mlađi bajt pomeraja četvrtim bajtom. Kod registarskog indirektnog sa pomerajem adresiranja registara opšte namene R[0] i R[31] koji se koristi specificiran je bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcija je četiri bajta. Kod bazno indeksnog sa pomerajem adresiranjem kao bazni registar se koristi jedan od registara opšte namene R[0] i R[31] koji je specificiran bitovima 4 do 0 drugog bajta instrukcije, dok se kao indeksni registar koristi prvi sledeći registar opšte namene.

Realizovati blok *fetch* procesora koji na osnovu sadržaja programskog brojača PC čita instrukcije i smešta u prihvatni registar instrukcije. Blok kreće sa čitanjem instrukcije ukoliko se u flip-flopu FETCH nalazi vrednost 1. Po završenom čitanju instrukcije upisivanjem vrednosti 1 u flip-flop ADDR ili EXEC startuje se blok *addr* ili blok *exec* i produžava sa izvršavanjem faze formiranje adrese i čitanje operanda ili izvršavanje operacije, respektivno, dok se upisivanjem vrednost 0 u flip-flop FETCH zaustavlja blok *fetch*.

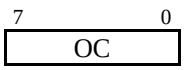
REŠENJE

Formati instrukcija

U procesoru se koriste sledeći formati instrukcija:

Format bezadresnih instrukcija – RTS, RTI, PUSH, POP, ASR

Format ovih instrukcija je dat na slici 1.

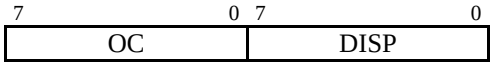


Slika 1 Format bezadresnih instrukcija

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

Format instrukcije relativnog skoka – BZ

Format ove instrukcije je dat na slici 2.



Slika 2 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.

Format instrukcija apsoolutnog skoka – JMP, JSR

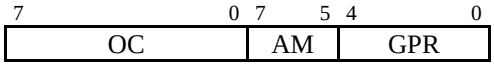
Format ovih instrukcija je dat na slici 3.

Slika 3 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.

Format jednoadresnih registarskih instrukcija – LD, ST, ADD, AND, JMPADR sa registarskim direktnim i registarskim indirektnim adresiranjima

Format ovih instrukcija je dat na slici 4.

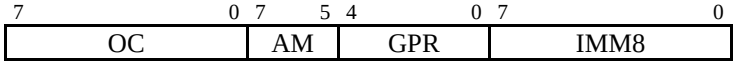


Slika 4 Format jednoadresnih registarskih instrukcija – R format

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

Format jednoadresnih neposrednih instrukcija – LD, ST, ADD, AND, JMPADR sa neposrednim adresiranjem

Format ovih instrukcija je dat na slici 5.

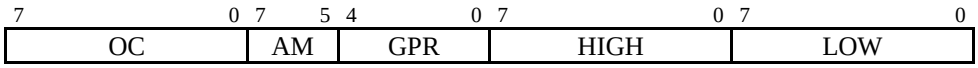


Slika 5 Format jednoadresnih instrukcija – IB format

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.

Format jednoadresnih memorijskih instrukcija – LD, ST, ADD, AND, JMPADR sa memorijskim direktnim, memorijskim indirektnim, registarskim indirektnim sa pomerajem, baznim indeksnim sa pomerajem i PC relativnim sa pomerajem adresiranjima

Format ovih instrukcija je dat na slici 6.

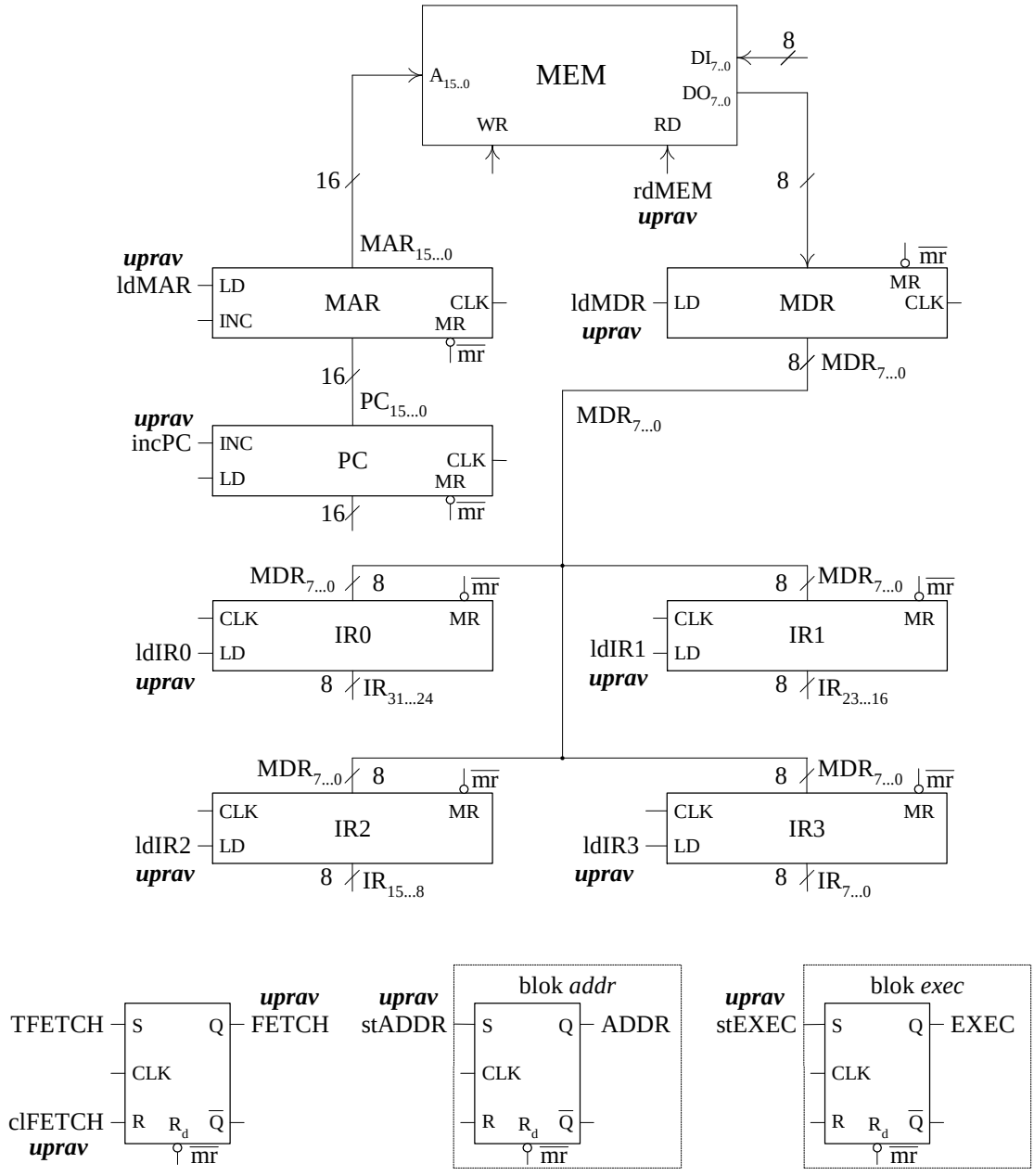


Slika 6 Format jednoadresnih instrukcija – AP format

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.

Operaciona jedinica bloka *fetch*

Operaciona jedinica bloka *fetch* (slika 7) sadrži adresni registar $MAR_{15...0}$, prihvatni registar podatka $MDR_{7...0}$, programski brojač $PC_{15...0}$, prihvatni registar instrukcije $IR_{31...0}$, kombinacione mreže za generisanje signala logičkih uslova operacija, adresiranja i dužina instrukcija i flip-flop FETCH.



Slika 7 Operaciona jedinica

Registar $MAR_{15...0}$ je 16-to razredni adresni registar čiji se sadržaj koristi kao adresa memorijske lokacije memorije ***MEM*** sa koje se čita jedan bajt. Sadržaj programskog brojača PC se vodi na ulaze registra $MAR_{15...0}$ i u njega upisuje vrednošću 1 signala **ldMAR**. Sadržaj registra $MAR_{15...0}$ se vodi na adresne linije $A_{15...0}$ memorije ***MEM***.

Registar $MDR_{7...0}$ je 8-mo razredni prihvatni registar podatka. Pri čitanju iz memorije ***MEM*** se vrednošću 1 signala **ldMDR** sadržaj sa izlaznih linija podataka $DO_{7...0}$ memorije ***MEM*** upisuje u registar $MDR_{7...0}$. Sadržaj registra $MDR_{7...0}$ se vodi na ulaze 8-mo razrednih

registara IR0, IR1, IR2 i IR3 u koje se smeštaju razredi 31…24, 23…16, 15…8 i 7…0 prihvatnog registra instrukcije $IR_{31...0}$.

Registar $PC_{15...0}$ je 16-to razredni programski brojač čiji sadržaj predstavlja adresu memorijske lokacije počev od koje treba pročitati jedan do četiri bajta instrukcije. Sadržaj registra $PC_{15...0}$ se inkrementira vrednošću 1 signala **incPC** prilikom čitanja svakog bajta instrukcije koji se nalaze u susednim 8-mo bitnim memorijskim lokacijama.

Registri IR0, IR1, IR2 i IR3 su 8-mo razredni registri koji formiraju razrede 31…24, 23…16, 15…8 i 7…0 prihvatnog registra instrukcije $IR_{31...0}$. Instrukcije mogu, u zavisnosti od formata instrukcije, da budu dužine 1, 2, 3 ili 4 bajta (slike 1 do 8). 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 vrednošću 1 jednog od signala **ldIR0**, **ldIR1**, **ldIR2** i **ldIR3**, respektivno. Prvi bajt instrukcije se uvek čita i upisuje u razrede $IR_{31...24}$ čiji sadržaj predstavlja kod operacije. Broj preostalih bajtova instrukcije koji se čitaju i upisuje u razrede $IR_{23...16}$, $IR_{15...8}$ i $IR_{7...0}$ zavisi od vrednosti koda operacije, a u slučaju aritmetičkih i logičkih operacija, i od načina adresiranja i njihov sadržaj ima različito značenje (slike 1 do 9).

Signali logičkih uslova operacija, adresiranja i dužina instrukcija, generišu se prema izrazima datim u tabelama 1, 2 i 3, respektivno.

Tabela 1 Signali operacija

PUSH = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot \overline{IR_{27}} \cdot \overline{IR_{26}} \cdot \overline{IR_{25}} \cdot \overline{IR_{24}}$

POP = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot \overline{IR_{27}} \cdot \overline{IR_{26}} \cdot \overline{IR_{25}} \cdot IR_{24}$

RTS = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot IR_{27} \cdot \overline{IR_{26}} \cdot IR_{25} \cdot \overline{IR_{24}}$

RTI = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot \overline{IR_{27}} \cdot \overline{IR_{26}} \cdot IR_{25} \cdot IR_{24}$

ASR = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot \overline{IR_{27}} \cdot IR_{26} \cdot \overline{IR_{25}} \cdot \overline{IR_{24}}$

BZ = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot \overline{IR_{27}} \cdot IR_{26} \cdot \overline{IR_{25}} \cdot IR_{24}$

JMP = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot \overline{IR_{27}} \cdot IR_{26} \cdot IR_{25} \cdot \overline{IR_{24}}$

JSR = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot \overline{IR_{27}} \cdot IR_{26} \cdot IR_{25} \cdot IR_{24}$

LD = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot IR_{27} \cdot \overline{IR_{26}} \cdot \overline{IR_{25}} \cdot \overline{IR_{24}}$

ST = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot IR_{27} \cdot \overline{IR_{26}} \cdot \overline{IR_{25}} \cdot IR_{24}$

ADD = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot IR_{27} \cdot \overline{IR_{26}} \cdot IR_{25} \cdot \overline{IR_{24}}$

AND = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot IR_{27} \cdot \overline{IR_{26}} \cdot IR_{25} \cdot IR_{24}$

JADR = $\overline{IR_{31}} \cdot \overline{IR_{30}} \cdot \overline{IR_{29}} \cdot \overline{IR_{28}} \cdot IR_{27} \cdot IR_{26} \cdot \overline{IR_{25}} \cdot \overline{IR_{24}}$

Tabela 2 Signali adresiranja

regdir = $\overline{IR_{23}} \cdot \overline{IR_{22}} \cdot \overline{IR_{21}}$

regind = $\overline{IR_{23}} \cdot \overline{IR_{22}} \cdot IR_{21}$

memdir = $\overline{IR_{23}} \cdot IR_{22} \cdot \overline{IR_{21}}$

memind = $\overline{IR_{23}} \cdot IR_{22} \cdot IR_{21}$

regindpom = $IR_{23} \cdot \overline{IR_{22}} \cdot \overline{IR_{21}}$

bxpom = $\mathbf{IR_{23}} \cdot \overline{\mathbf{IR_{22}}} \cdot \mathbf{IR_{21}}$

pcrel = $\mathbf{IR_{23}} \cdot \mathbf{IR_{22}} \cdot \overline{\mathbf{IR_{21}}}$

immed = $\mathbf{IR_{23}} \cdot \mathbf{IR_{22}} \cdot \mathbf{IR_{21}}$

Tabela 3 Signali dužina instrukcija

l1 = **PUSH** + **POP**+ **RTS** + **RTI** + **ASR**
l2_brnch = **BEQL**
l2_arlog = (**LD** + **ST** + **ADD** + **AND** + **JADR**)·(**regdir**+**regind**)
l3_jump = **JMP** + **JSR**
l3_arlog = (**LD** + **ST** + **ADD** + **AND** + **JADR**)·**immed**

Signali operacija **PUSH**, **POP**, **RTS**, **RTI**, **ASR**, **BZ**, **JMP**, **JSR** LD, ST, ADD, AND i JMPADR (tabela 1), od kojih u datom trenutku samo jedan ima vrednost 1, ukazuju koja operacija se realizuje.

Signali adresiranja **regdir**, **regind**, **memdir**, **memind**, **regindpom**, **bxpom**, **bcpom** i **imm** (tabela 2), od kojih u datom trenutku samo jedan ima vrednost 1, ukazuju kojim načinom adresiranja se dolazi do operanda.

Signali dužina instrukcija l1, l2_brnch, l2_arlog, l3_jump i **l3_arlog** ukazuje kolika je dužina instrukcije u bajtovima (tabela 3).

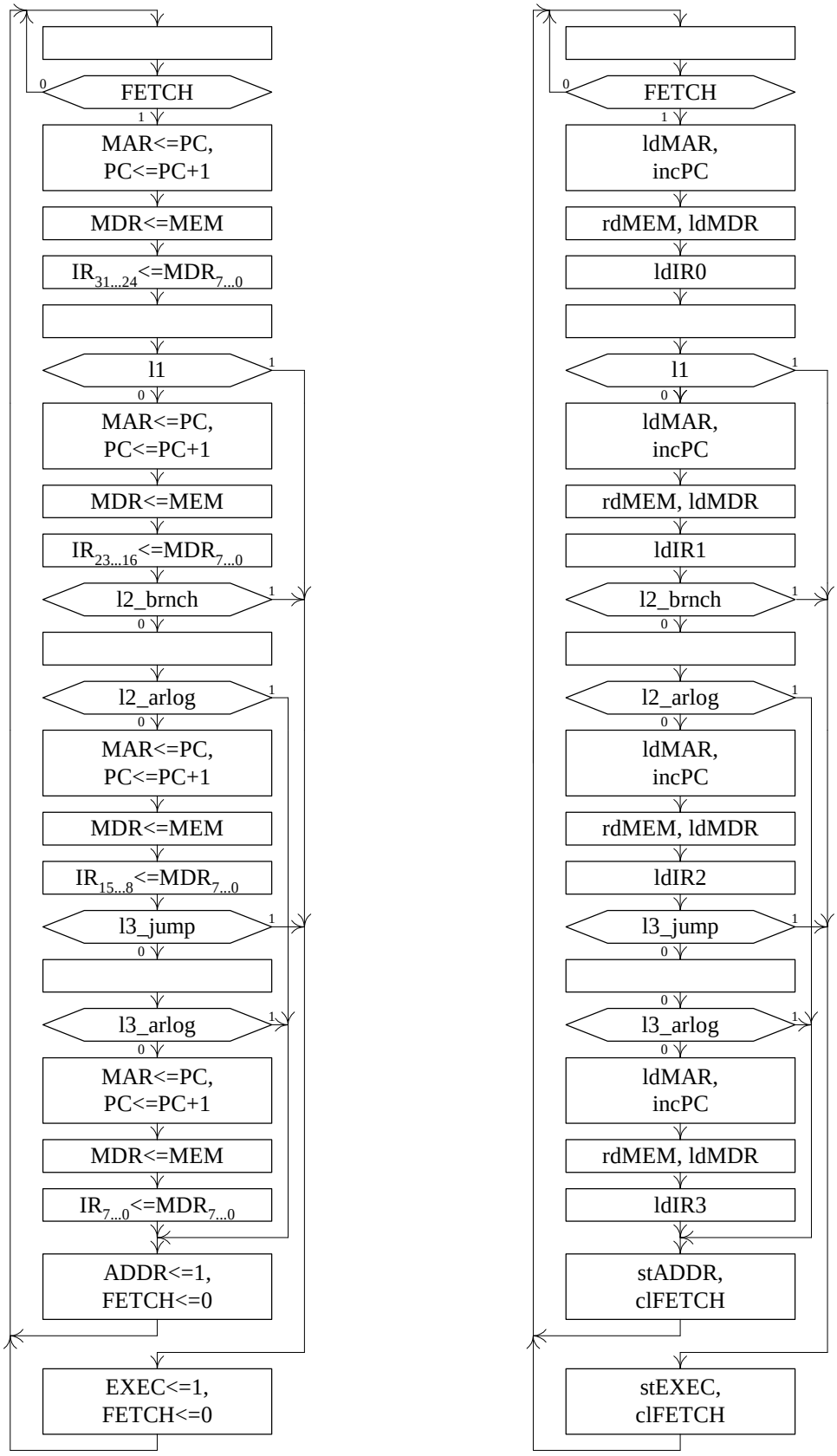
Signal logičkog uslova dužina instrukcije jedan bajt **l1** svojom vrednošću 1 određuje da je dužina instrukcije jedan bajt, a vrednošću 0 da je dužina instrukcije dva, tri ili četiri bajta. Signal **l1** ima vrednost 1 ukoliko se radi o instrukciji povratka iz potprograma RTS, instrukciji povratka iz prekidne rutine RTI, o nekoj od instrukcija prenosa PUSH ili POP, ili o instrukciji pomeranja ASR, 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 instrukcija uslovnog skoka BZ. Signal **l2_arlog** ima vrednost 1 ukoliko se radi o aritmetičkoj instrukciji ADD, logičkoj instrukcija AND, o nekoj od instrukcija prenosa LD ili ST, ili o instrukciji bezuslovnog skoka na sračunatu adresu JADR 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 aritmetičkoj instrukciji ADD, logičkoj instrukcija AND, o nekoj od instrukcija prenosa LD ili ST, ili o instrukciji bezuslovnog skoka na sračunatu adresu JADR za koju je specificirano neposredno adresiranje.

Dijagrami toka mikrooperacija i upravljačkih signala i sekvenca upravljačkih signala

Dijagrami toka mikrooperacija i upravljačkih signala dati su na slici 10, a sekvenca upravljačkih signala u tabeli 4.



Slika 10 Dijagrami toka mikrooperacija i upravljačkih signala

Tabela 4 Sekvenca upravljačkih signala

! Čitanje instrukcije !

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

$step_{00}$ *br* (*if* $\overline{\textbf{FETCH}}$ *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 $IR_{31...0}$. U koraku $step_{01}$ se vrednostima 1 signala **ldMAR** i **incPC** sadržaj registra $PC_{15..0}$ upisuje u adresni registar $MAR_{15..0}$ i sadržaj registra $PC_{15..0}$ inkrementira, respektivno, i prelazi na korak $step_{02}$. U koraku $step_{02}$ se realizuje čitanje jednog bajta i upisivanje u registar podatka $MDR_{7..0}$. Vrednošću 1 signala **rdMEM** se sa adrese određene sadržajem adresnog registra $MAR_{15..0}$, koji je povezan na adresne linije $A_{15..0}$ memorije ***MEM***, u memoriji ***MEM*** realizuje operacija čitanja. Pročitani sadržaj, koji memorija ***MEM*** pušta na izlazne linije podataka $DO_{7...0}$, se vrednošću 1 signala **ldMDR** upisuje u registar podatka $MDR_{7..0}$ i prelazi na korak $step_{03}$. U koraku $step_{03}$ se vrednošću 1 signala **ldIR0** pročitani sadržaj prebacuje iz registra $MDR_{7..0}$ u razrede $IR_{31...24}$ prihvatnog registra instrukcije $IR_{31...0}$ i prelazi na korak $step_{04}$. !

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

! U koraku $step_{04}$ se proverava vrednost signala **l1** 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 $step_{11}$ i startovanje bloka za izvršavanje operacije *exec* i zaustavljanje bloka *fetch* ili korak $step_{05}$ i produžava sa čitanjem bajtova instrukcije. !

$step_{04}$ *br* (*if* **l1** *then* $step_{11}$);

! 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**;
$step_{06}$ **rdMEM, ldMDR**;

! U koraku $step_{07}$ se vrednošću 1 signala **ldIR1** pročitani bajt prebacuje iz registra $MDR_{7..0}$ u razrede $IR_{23...16}$ prihvatnog registra instrukcije $IR_{31...0}$. !

! U koracima $step_{07}$ i $step_{08}$ se proverava vrednost signala **l2_brnch** i **l2_arlog** 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 $step_{11}$ i startovanje bloka za izvršavanje operacije *exec* i zaustavljanje bloka *fetch* ili korak $step_{08}$ 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 startovanje bloka za formiranje adrese i čitanje operanda *addr* i zaustavljanje bloka *fetch* ili korak $step_{09}$ i produžava sa čitanjem bajtova instrukcije.!

$step_{07}$ **ldIR1**,
br (*if* **l2_brnch** *then* $step_{11}$);
$step_{08}$ *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**;
$step_{0A}$ **rdMEM, ldMDR**;

! U koraku $step_{0B}$ se vrednošću 1 signala **ldIR2** pročitani bajt prebacuje iz registra $MDR_{7..0}$ u razrede $IR_{15...8}$ prihvatnog registra instrukcije instrukcije $IR_{31...0}$. !

! U koracima $step_{0B}$ i $step_{0C}$ se proverava vrednost signala **l3_jump** i **l3_arlog** 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 $step_{11}$ i startovanje bloka za izvršavanje operacije *exec* i zaustavljanje bloka *fetch* ili korak $step_{0C}$ 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 startovanje bloka za formiranje adrese i čitanje operanda *addr* i zaustavljanje bloka *fetch* ili na korak $step_{0D}$ i produžava sa čitanjem četvrtog bajta instrukcije.!

$step_{0B}$ **ldIR2**,
br (*if* **l3_jump** *then* $step_{11}$);
$step_{0C}$ *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 vrednošću 1 signala **ldIR3** pročitani bajt prebacuje iz registra $MDR_{7..0}$ u razrede $IR_{7...0}$ prihvatnog registra instrukcije $IR_{31...0}$. Iz koraka $step_{0F}$ se prelazi na korak $step_{10}$ i startovanje bloka za formiranje adrese i čitanje operanda *addr* i zaustavljanje bloka *fetch*. !

$step_{0D}$ **ldMAR**, **incPC**;
$step_{0E}$ **rdMEM, ldMDR**;
$step_{0F}$ **ldIR3**;

! U korak $step_{10}$ se dolazi iz koraka $step_{08}$ i $step_{0C}$. U koraku $step_{10}$ se vrednošću 1 signala **clFETCH** upisuje vrednost 0 u flip flop FETCH bloka *fetch*, čime se zaustavlja blok *fetch*, dok se vrednošću 1 signala **stADDR** upisuje vrednost 1 u flip flop ADDR bloka *addr*, čime se strartuje blok *addr*. !

$step_{10}$ **clFETCH, stADDR**,
br $step_{00}$;

! U korak $step_{11}$ se dolazi iz koraka $step_{04}$, $step_{07}$ i $step_{0B}$. U koraku $step_{11}$ se vrednošću 1 signala **clFETCH** upisuje vrednost 0 u flip flop FETCH bloka *fetch*, čime se zaustavlja blok *fetch*, dok se vrednošću 1 signala **stEXEC** upisuje vrednost 1 u flip flop EXEC bloka *exec*, čime se startuje blok *exec*. !

$step_{11}$ **clFETCH, stEXEC**,
br $step_{00}$;

Upravljačka jedinica

Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu sekvence upravljačkih signala po koracima (tabela 4). Mikroprogram se formira tako što se svakom koraku u sekvenci upravljačkih signala po koracima pridruži binarna reč sa slike 11. 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
ldMAR	incPC	rdMEM	ldMDR	$ldIR_0$	$ldIR_1$	$ldIR_2$	$ldIR_3$

8	9	10	11	12	13	14	15
clFETCH	-	stADDR	stEXEC	*cc*			

16	17	18	19	20	21	22	23
ba							

Slika 11 Mikroinstrukcija

Mikroinstrukcija ima dva dela i to operacini deo i upravljački deo. Operacioni deo čine bitovi 0 do 11, a upravljački deo čine bitovi 12 do 23. 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 i uslovni skok.

Bezuslovni skokovi se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 4) 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 5.

Tabela 5 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 **Error! Reference source not found.**.

Tabela 6 Signali uslovnih skokova

signal uslovnog skoka	polje *cc*	signal uslova
brnotFETCH	2	$\overline{\textbf{FETCH}}$
brl1	3	**l1**
brl2_brnch	4	**l2_brnch**

signal uslovnog skoka	polje *cc*	signal uslova
brl2_arlog	5	**l2_arlog**
brl3_jump	6	**l3_jump**
brl3_arlog	**7**	**l3_arlog**

Vrednosti 0, 8, 9, A, B, C, D, E i F polja *cc* koje nisu dodeljene signalu bezuslovnog skoka i signalima uslovnih skokova određuju 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 4). 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 24 bitova. Za kodiranje operacionog dela mikroinstrukcije koristi se 12 bitova. Upravljačkih signala operacione jedinice ima 11, ali je umesto 11 bita usvojena dužina operacionog dela mikroinstrukcije 12 bitova. Za kodiranje upravljačkog dela mikroinstrukcije koristi se 12 bitova. Signala bezuslovnog skoka i signala uslovnih skokova ima 7, ali je umesto 3 bita usvojena dužina polja *cc* upravljačkog dela mikroinstrukcije 4 bita. Ukupan broj koraka u sekvenci upravljačkih signala po koracima je 18, ali je umesto 5 bita usvojena dužina polja *ba* upravljačkog dela mikroinstrukcije 8 bitova. Ovo je učinjeno da bi se dobio pregledniji mikroprogram predstavljen u heksadecimalnom obliku.

Mikroprogram se formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima (tabela 4) 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 ili 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* $step_A$ i *br* (*if* **uslov** *then* $step_A$). 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 5. 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 7.

Tabela 7 Koraci $step_i$, $step_A$, vrednosti $\mathbf{madr_A}$ i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	$\mathbf{madr_A}$	A
$step_{10}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{11}$	$step_{00}$	$\mathbf{madr_{00}}$	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 **Error! Reference source not found.**. 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 8.

Tabela 8 Koraci $step_i$, uslovi **uslov**, koraci $step_A$, vrednosti $\mathbf{madr_A}$ i vrednosti A za uslovne skokove

$step_i$	uslov	$step_A$	$\mathbf{madr_A}$	A
$step_{00}$	$\overline{\mathbf{FETCH}}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{04}$	**I1**	$step_{11}$	$\mathbf{madr_{11}}$	11
$step_{07}$	**I2_brnch**	$step_{11}$	$\mathbf{madr_{11}}$	11

$step_i$	uslov	$step_A$	$\mathbf{madr_A}$	A
$step_{08}$	**I2_arlog**	$step_{10}$	$\mathbf{madr_{10}}$	10
$step_{0B}$	**I3_jump**	$step_{11}$	$\mathbf{madr_{11}}$	11
$step_{0C}$	**I3_arlog**	$step_{10}$	$\mathbf{madr_{10}}$	10

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu mikroprogramske realizacije signal bezuslovnog skoka (tabela 5), signali uslovnih skokova (tabela **Error! Reference source not found.**) i signali vrednosti A za bezuslovne skokokve (tabela 7) i uslovne skokove (tabela 8).

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 4) formiran mikroprogram (tabela 9). 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 9 Mikroprogram za upravljačku jedinicu mikroprogramske realizacije

0 0	0 0 0 2 0 0	! **brnotFETCH**, **madr$_{00}$** !
0 1	C 0 0 0 0 0	! **ldMAR**, **incPC** !
0 2	3 0 0 0 0 0	! **rdMEM, ldMDR** !
0 3	0 8 0 0 0 0	! **ldIR0** !
0 4	0 0 0 3 1 1	! **brl1**, **madr$_{11}$** !
0 5	C 0 0 0 0 0	! **ldMAR**, **incPC**!
0 6	3 0 0 0 0 0	! **rdMEM, ldMDR** !
0 7	0 4 0 4 1 1	! **ldIR1**, **brl2_brnch, madr$_{11}$** !
0 8	0 0 0 5 1 0	! **brl2_arlog**, **madr$_{10}$** !
0 9	C 0 0 0 0 0	! **ldMAR**, **incPC**!
0 A	3 0 0 0 0 0	! **rdMEM, ldMDR** !
0 B	0 2 0 6 1 1	! **ldIR2**, **brl3_jump**, **madr$_{11}$**!
0 C	0 0 0 7 1 0	! **brl3_arlog**, **madr$_{10}$** !
0 D	C 0 0 0 0 0	! **ldMAR**, **incPC**!
0 E	3 0 0 0 0 0	! **rdMEM, ldMDR** !
0 F	0 1 0 0 0 0	! **ldIR3**!
1 0	0 0 A 1 0 0	! **clFETCH**, **stADDR**, **bruncnd**, **madr$_{00}$** !
1 1	0 0 9 1 0 0	! **clFETCH**, **stEXEC**, **bruncnd**, **madr$_{00}$** !

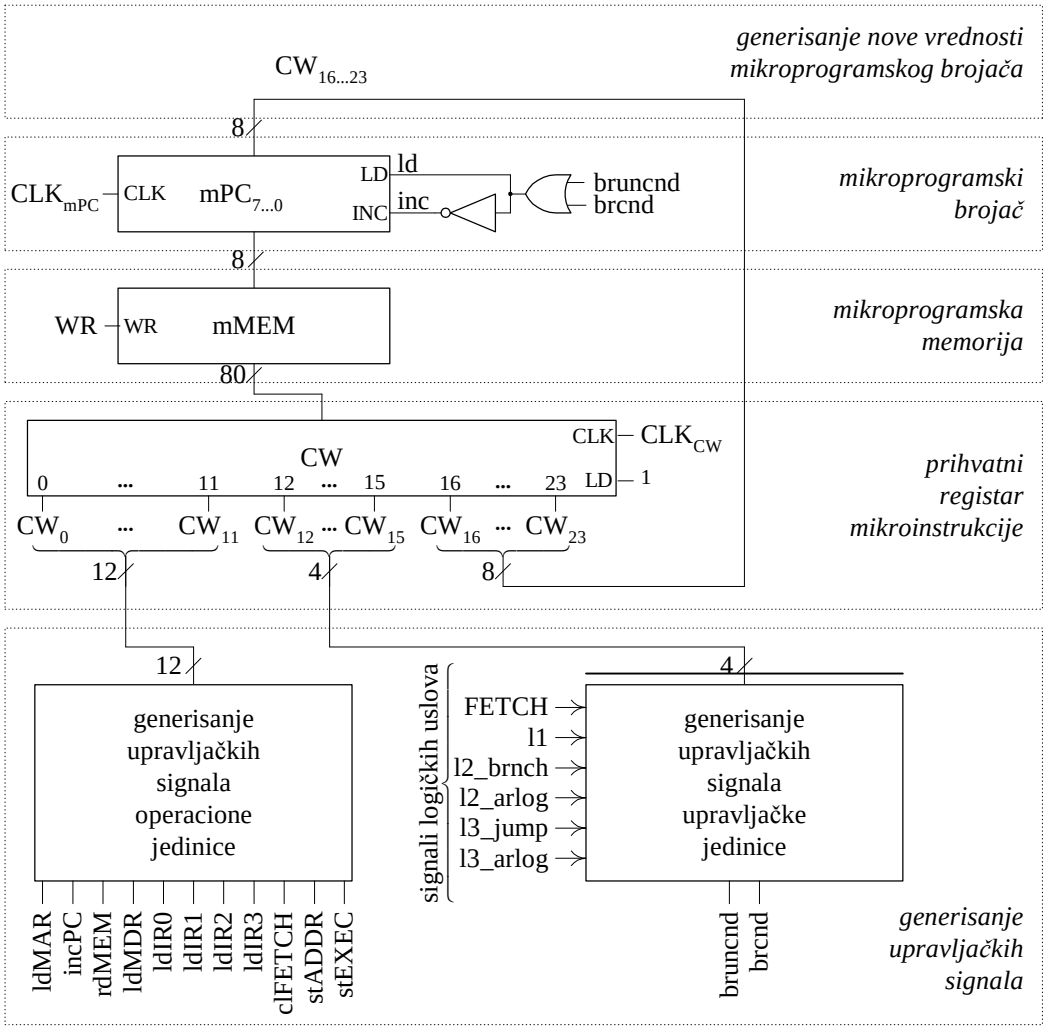
Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 12. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti mikroprogramskog brojača*, blok *mikroprogramski brojač*, blok *mikroprogramska memorija*, blok *prihvatni registar mikroinstrukcije* i blok *generisanje upravljačkih signala*. Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.

Blok *generisanje nove vrednosti mikroprogramskog brojača* se sastoji od linija po kojima se dovodi vrednosti koju treba upisati u mikroprogramski brojač $mPC_{7...0}$. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikroprograma. Prihvatni registar mikroinstrukcije $CW_{0...23}$ u svojim razredima $CW_{16...23}$ sadrži vrednost za upis u mikroprogramski brojač $mPC_{7...0}$ za bezuslovne skokove (tabela 7) i uslovne skokove (tabela 8) u mikroprogramu (tabela 9). Sadržaj razreda $CW_{16...23}$ je prisutan 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 9). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko signali **brcnd** i **bruncnd** imaju

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



Slika 12 Struktura upravljačke jedinice mikroprogramske realizacije

U režimu skoka pri pojavi signala takta **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 9). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ukoliko jedan od signala **brcnd** i **bruncnd** ima vrednost 1. Signali **brcnd** i **bruncnd** normalno imaju vrednost 0 sem prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran 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 9). S obzirom da se mikroprogram svih faza izvršavanja instrukcija nalazi u opsegu od adrese 00 do adrese 11, 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 24 bita, a kapacitet veličinom mikroprograma (tabela 9) i iznosi 256 lokacija. Adresiranje mikroprogramske memorije se realizuje sadržajem mikroprogramskog brojača $mPC_{7...0}$.

Blok *prihvatni registar mikroinstrukcije* sadrži prihvatni registar mikroinstrukcije $CW_{0...23}$. Prihvatni registar mikroinstrukcije $CW_{0...23}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...11}$ i $CW_{12...15}$ se koriste u bloku *generisanje upravljačkih signala* za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{16...23}$ 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 $CW_{0...11}$ prihvatnog registra mikroinstrukcije generišu upravljačke signale operacione jedinice i na osnovu sadržaja razreda $CW_{12...15}$ prihvatnog registra mikroinstrukcije i signala logičkih uslova **FETCH**, **l1**, ..., **l3_arlog** koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice ***oper*** se generišu na sledeći način:

- **ldMAR** = $\mathbf{CW_0}$
- **incPC** = $\mathbf{CW_1}$
- **rdMEM** = $\mathbf{CW_2}$
- **ldMDR** = $\mathbf{CW_3}$
- **ldIR0** = $\mathbf{CW_4}$
- **ldIR1** = $\mathbf{CW_5}$
- **ldIR2** = $\mathbf{CW_6}$
- **ldIR3** = $\mathbf{CW_7}$
- **clFETCH** = $\mathbf{CW_8}$
- **stADDR** = $\mathbf{CW_9}$
- **stEXEC** = $\mathbf{CW_{10}}$

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

- **bruncnd** = $\overline{\mathbf{CW_{12}}} \cdot \overline{\mathbf{CW_{13}}} \cdot \overline{\mathbf{CW_{14}}} \cdot \mathbf{CW_{15}}$
- **brcnd** = **brnotFETCH**$\cdot \overline{\mathbf{FETCH}}$ + **brl1**$\cdot$**l1** + **brl2_brnch**$\cdot$ **l2_brnch** + **brl2_arlog**$\cdot$ **l2_arlog** + **brl3_jump**$\cdot$ **l3_jump** + **brl3_arlog**$\cdot$ **l3_arlog** +
- **brnotSTART** = $\overline{\mathbf{CW_{12}}} \cdot \overline{\mathbf{CW_{13}}} \cdot \mathbf{CW_{14}} \cdot \overline{\mathbf{CW_{15}}}$
- **brl1**= $\overline{\mathbf{CW_{12}}} \cdot \overline{\mathbf{CW_{13}}} \cdot \mathbf{CW_{14}} \cdot \mathbf{CW_{15}}$
- **brl2_brnch**= $\overline{\mathbf{CW_{12}}} \cdot \mathbf{CW_{13}} \cdot \overline{\mathbf{CW_{14}}} \cdot \overline{\mathbf{CW_{15}}}$
- **brl2_arlog**= $\overline{\mathbf{CW_{12}}} \cdot \mathbf{CW_{13}} \cdot \overline{\mathbf{CW_{14}}} \cdot \mathbf{CW_{15}}$
- **brl3_jump**= $\overline{\mathbf{CW_{12}}} \cdot \mathbf{CW_{13}} \cdot \mathbf{CW_{14}} \cdot \overline{\mathbf{CW_{15}}}$
- **brl3_arlog**= $\overline{\mathbf{CW_{12}}} \cdot \mathbf{CW_{13}} \cdot \mathbf{CW_{14}} \cdot \mathbf{CW_{15}}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice ***oper*** i to **FETCH**, **l1**, **l2_brnch**, **l2_arlog**, **l3_jump** i **l3_arlog**.

2 ZADATAK 2

Posmatra se blok *addr* procesora u kome se realizuje faza formiranje adrese i čitanje operanda. Procesor je tako realizovan da se faze čitanje instrukcije, formiranje adrese i čitanje operanda, izvršavanje operacije i opsluživanje prekida realizuju u posebnim blokovima. Pored procesora u posmatranim računaru postoji i memorija. Memorija je kapaciteta 2^{16} bajtova. Širina memorijske reči je 1 bajt. Procesor je sa jednoadresnim formatom instrukcija. Podaci su celobrojne veličine bez znaka dužine jedan bajt.

U bloku *addr* postoji registar programskog brojača PC dužine dva bajta, adresni registar memorije MAR dužine dva bajta, prihvatni registar podatka memorije MBR dužine jedan bajt, prihvatni registar instrukcije IR dužine četiri bajta i prihvatni registar operanda BB dužine jedan bajt. Instrukcije su dužine jedan, dva, tri ili četiri bajta. Pretpostaviti da se u registru IR nalazi pročitana instrukcija i da u bloku *addr* procesora treba da se realizuje faza formiranje adrese i čitanje operanda.

Prvi bajt instrukcije sadrži polje koda operacije.

U procesoru postije bezadresne instrukcije, instrukcije uslovnog skoka, instrukcije bezuslovnog skoka i adresne instrukcije.

Bezadresne instrukcije su instrukcija stavljanja sadržaja akumulatora na stek (PUSH), instrukcija skidanja sadržaja sa steka u akumulator (POP), instrukcija povratka iz potprograma (RTS), instrukcija povratka iz prekidne rutine (RTI) i instrukcija aritmetičkog pomeranja sadržaja akumulatora udesno za jedno mesto (ASR). Za bezadresne instrukcije PUSH, POP, RTS i RTI i ASR su usvojeni kodovi operacija 00000000, 00000001, 00000010, 0000000011 i 00000100, respektivno. Dužina instrukcija je jedan bajt.

Instrukcija uslovnog skoka je instrukcija uslovnog skoka ukoliko je rezultat nula (BZ). Za instrukciju BZ je usvojen kod operacije 00000101. Instrukcija BZ se realizuje kao relativni skok u odnosu na tekuću vrednost programskog brojača PC, a pomeraj je 8-mo bitna celobrojna veličina sa znakom data drugim bajtom instrukcije. Dužina instrukcije je dva bajta.

Instrukcije bezuslovnog skoka su instrukcija bezuslovnog skoka (JMP) i instrukcija skoka na potprogram (JSR). Za instrukcije JMP i JSR su usvojeni kodovi operacija 00000110 i 00000111, respektivno. Instrukcije JMP i JSR se realizuju kao apsolutni skokovi, a adresa skoka je data drugim i trećim bajtom instrukcije, pri čemu je stariji bajt adrese skoka dat drugim bajtom instrukcije a mlađi bajt adrese skoka trećim bajtom instrukcije. Dužina instrukcija je tri bajta.

Adresne instrukcije su instrukcija prenosa u akumulator (LD), instrukcija prenosa iz akumulatora (ST), aritmetička instrukcija sabiranja (ADD), logička instrukcija logički proizvod (AND) i instrukcija bezuslovnog skoka na sračunatu adresu (JADR). U instrukciji ST nije dozvoljeno neposredno adresiranje, a u instrukciji JADR nije dozvoljeno direktno registarsko i neposredno adresiranje, pa ukoliko se jave ova adresiranja u ovim instrukcijama, instrukcije treba da budu bez dejstva. Za instrukcije LD, ST, ADD, AND, ASR i JADR usvojeni kodovi operacija 00001000, 00001001, 00001010, 00001011, 00001100 i 00001101, respektivno. Dužina instrukcija je dva, tri ili četiri bajta i zavisi od specificiranog načina adresiranja.

Za adresne instrukcije se bitovima 7, 6 i 5 drugog bajta instrukcije specificira načina adresiranja i to na sledeći način: 000-registarsko direktno adresiranje (regdir), 001-registarsko

indirektno adresiranje (regind), 010-memorijsko direktno adresiranje (memdir), 011-memorijsko indirektno adresiranje (memind), 100-registarsko indirektno sa pomerajem adresiranje (regindpom), 101-bazno indeksno sa pomerajem adresiranje (bxpom), 110- PC relativno sa pomerajem adresiranje (pcrel) i 111-neposredno adresiranje (immed). Registarsko direktno i registarsko indirektno adresiranje koriste neki od registara opšte namene R[0] i R[31] specificiran bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcija je dva bajta. Neposredno adresiranje ima i treći bajt instrukcije u kome se nalazi operand, ali ne koristi registre opšte namene R[0] i R[31] specificirane bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcije je tri bajta.

Memorijsko direktno, memorijsko indirektno, registarsko indirektno sa pomerajem adresiranje, bazno indeksno sa pomerajem adresiranje i PC relativno sa pomerajem adresiranje imaju treći i četvrti bajt instrukcije. Kod memorijskog direktnog i memorijskog indirektnog adresiranja treći i četvrti bajt instrukcije sadrže adresu memorijske lokacije, pri čemu je stariji bajt adrese dat trećim bajtom a mlađi bajt adrese četvrtim bajtom. Kod memorijskog indirektnog adresiranja adresa dužine 16 bita zauzima dve susedne memorijske lokacije, pri čemu je stariji bajt adrese nalazi na nižoj a mlađi bajt adrese na višoj lokaciji. Bitovi 4 do 0 drugog bajta instrukcije se ne koriste. Dužina instrukcija je četiri bajta. Kod registarskog indirektnog sa pomerajem adresiranje, bazno indeksnog sa pomerajem adresiranjem i kod PC relativnog adresiranja treći i četvrti bajt instrukcije sadrže pomeraj, pri čemu je stariji bajt pomeraja dat trećim bajtom a mlađi bajt pomeraja četvrtim bajtom. Kod registarskog indirektnog sa pomerajem adresiranja registara opšte namene R[0] i R[31] koji se koristi specificiran je bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcija je četiri bajta. Kod bazno indeksnog sa pomerajem adresiranjem kao bazni registar se koristi jedan od registara opšte namene R[0] i R[31] koji je specificiran bitovima 4 do 0 drugog bajta instrukcije, dok se kao indeksni registar koristi prvi sledeći registar opšte namene.

Realizovati blok *addr* procesora koji na osnovu sadržaja prihvatnog registra instrukcije $IR_{31...0}$ za adresne instrukcije, sem instrukcija ST i JADR, čita operand saglasno specificiranom načinu adresiranja i smešta ga u registar $BB_{7...0}$. Operand može da bude u nekom od registara opšte namene, u memorijskoj lokaciji ili samoj instrukciji. U slučaju direktnog registarskog adresiranja ili neposrednog adresiranja, kod kojih se operand nalazi u nekom od registara opšte namene ili u instrukciji, prolaz instrukcije kroz blok *addr* se svodi na prebacivanje operanda iz registra opšte namene ili iz prihvatnog registra instrukcije u registar $BB_{7...0}$. U slučaju adresiranja kod kojih se operand nalazi u memoriji, prolazak instrukcije kroz blok *addr* se sastoji od koraka u kojima se prvo formira adresa operanda u memoriji i zatim operand čita i prebacuje u registar $BB_{7...0}$. Izuzetak su instrukcije ST i JADR. U slučaju instrukcije ST sa registarskim direktnim adresiranjem zaustavlja se blok *addr* i startuje blok za izvršavanje operacije *exec* u kome se sadržaj registra akumulatora $AB_{7...0}$ upisuje u registar opšte namene. U slučaju instrukcije ST sa nekim od memorijskih adresiranja u bloku *addr* se samo formira adresa operanda i smešta u registr $MAR_{15...0}$, pa se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec* u kome se sadržaj registra akumulatora $AB_{7...0}$ upisuje u memoriju na formiranoj adresi. Pretpostavljeno je da se u slučaju instrukcije ST neće javiti neposredno adresiranje. U slučaju instrukcije JADR sa nekim od memorijskih adresiranja, u ovoj fazi se samo formira adresa operanda i smešta u registr $MAR_{15...0}$, pa se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec* u kome se sadržaj registra $MAR_{15...0}$ upisuje u programski brojač $PC_{15...0}$. Pretpostavljeno je da se slučaju instrukcije JADR neće javiti neposredno adresiranje ili direktno registarsko adresiranje.

Blok *addr* kreće sa formiranjem adrese operanda i čitanjem operanda ukoliko se u flip-flopu ADDR nalazi vrednost 1. Po završenom čitanju instrukcije upisivanjem vrednosti 1 u

flip-flop EXEC startuje se blok *exec* i produžava sa izvršavanjem faze izvršavanje operacije, dok se upisivanjem vrednost 0 u flip-flop ADDR zaustavlja blok *addr*.

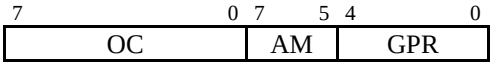
REŠENJE

Formati instrukcija

Kroz blok *addr* procesora prolaze jedino adresne instrukcije koje imaju sledeće formate instrukcija:

Format jednoadresnih registarskih instrukcija – LD, ST, ADD, AND, JMPADR sa registarskim direktnim i registarskim indirektnim adresiranjima

Format ovih instrukcija je dat na slici 13.

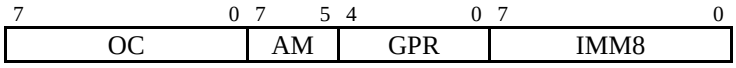


Slika 13 Format jednoadresnih registarskih instrukcija – R format

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

Format jednoadresnih neposrednih instrukcija – LD, ST, ADD, AND, JMPADR sa neposrednim adresiranjem

Format ovih instrukcija je dat na slici 14.

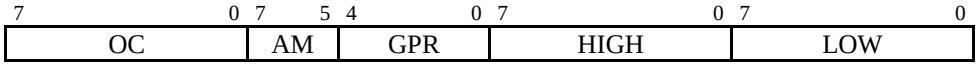


Slika 14 Format jednoadresnih instrukcija – IB format

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.

Format jednoadresnih memorijskih instrukcija – LD, ST, ADD, AND, JMPADR sa memorijskim direktnim, memorijskim indirektnim, registarskim indirektnim sa pomerajem, baznim indeksnim sa pomerajem i PC relativnim sa pomerajem adresiranjima

Format ovih instrukcija je dat na slici 15.



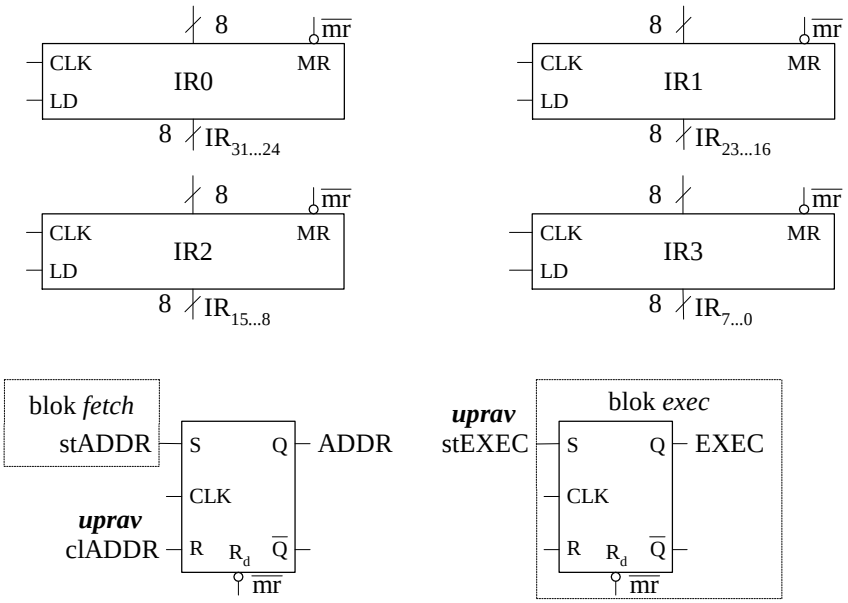
Slika 15 Format jednoadresnih instrukcija – AP format

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.

Operaciona jedinica bloka *addr*

Operaciona jedinica bloka *addr* sadrži prihvatni registar instrukcije $IR_{31...0}$, flip-flop ADDR (slika 16), adresni registar $MAR_{15...0}$ sa multiplekserom MX1, prihvatni registar podatka $MDR_{7...0}$, pomoćni registar $DW_{15...0}$, registarski fajl GPR sa inkrementerom INC i

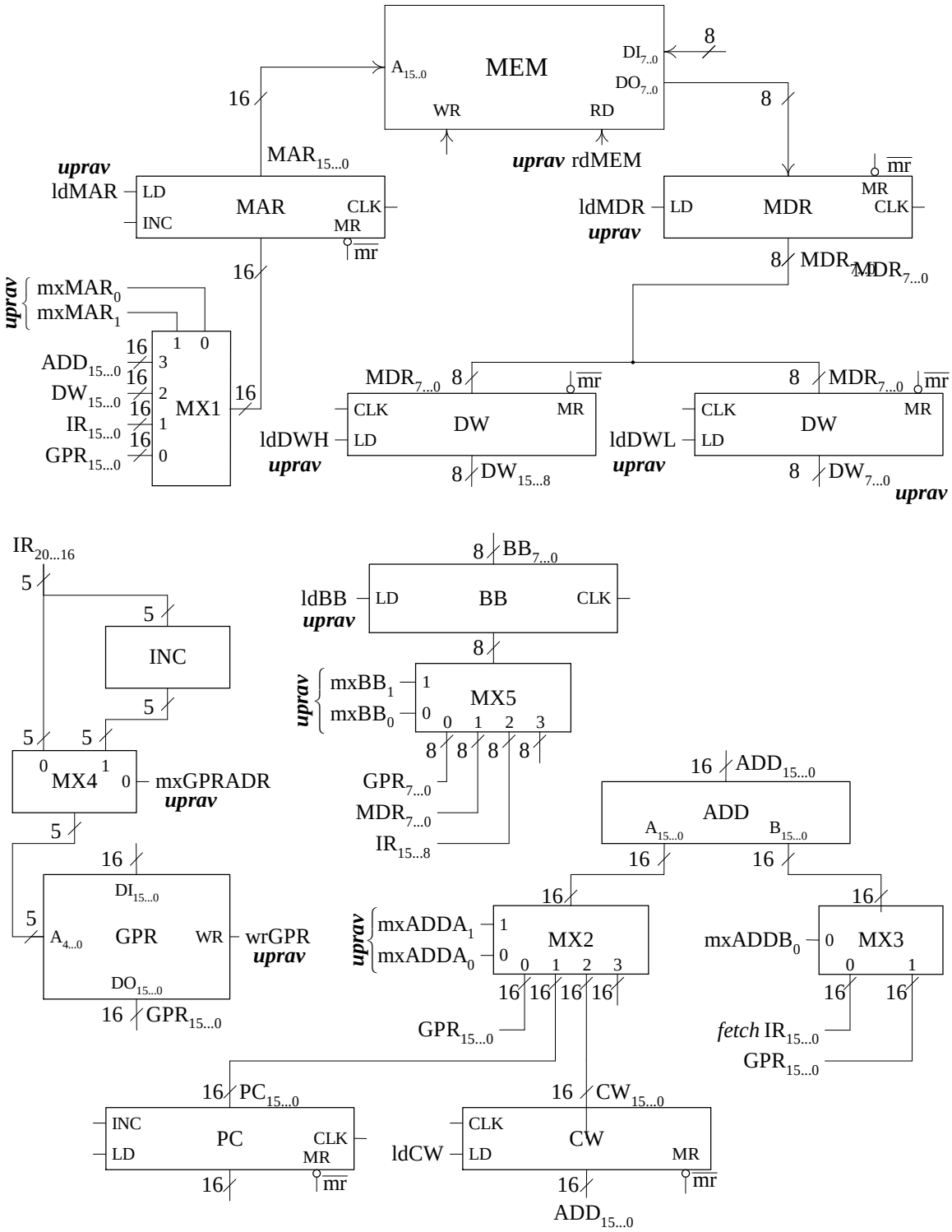
multiplekserom MX4, sabirač ADD sa multiplekserima MX2 i MX3, pomoćnim registrom $CW_{15...0}$ i programskim brojačem $PC_{15...0}$, prihvatni registar operanda $BB_{7...0}$ sa multiplekserom MX2 i kombinacione mreže za generisanje signala logičkih uslova operacija i adresiranja (slika 17).



Slika 16 Operaciona jedinica (prvi deo)

Registri IR0, IR1, IR2 i IR3 su 8-mo razredni registri koji formiraju razrede 31…24, 23…16, 15…8 i 7…0 prihvatnog registra instrukcije $IR_{31...0}$ (slika 16). Adresne instrukcije mogu, u zavisnosti od načina adresiranja, da budu dužine 2, 3 ili 4 bajta (slike 13, 14 i 15). U bloku *fetch* se saglasno formatu instrukcije prvi, drugi, treći i četvrti bajt instrukcije smeštaju redom u registre IR0, IR1, IR2 i IR3. Prvi bajt instrukcije se uvek čita i upisuje u razrede $IR_{31...24}$ čiji sadržaj predstavlja kod operacije. Broj preostalih bajtova instrukcije koji se čitaju i upisuje u razrede $IR_{23...16}$, $IR_{15...8}$ i $IR_{7...0}$ zavisi od vrednosti koda operacije, a u slučaju adresnih instrukcija, i od načina adresiranja i njihov sadržaj ima različito značenje. Prepostavljeno je da je instrukcija očitana u bloku *fetch* i da se sadržaj registra $IR_{31...0}$ može koristiti u bloku *addr* za realizaciju faze formiranje adrese i čitanje operanda adresnih instrukcija.

Flip-flop ADDR vrednostima 1 i 0 određuje da li je blok *addr* aktivan ili neaktivan, respektivno. U bloku *fetch* se po završetku faze čitanje instrukcije vrednošću 1 signala **stADDR** u flip-flop ADDR upisuje vrednost 1 i time u bloku *addr* pokreće faza formiranje adrese i čitanje operanda. U bloku *addr* se na kraju faze formiranje adrese i čitanje operanda vrednošću 1 signala **clADDR** upisuje vrednost 0 u flip-flop ADDR i time zaustavlja blok *addr*, dok se vrednošću 1 signala **stEXEC** u flip-flop EXEC u bloku *exec* upisuje vrednost 1 i time u bloku *exec* pokreće fazu izvršavanje operacije.



Slika 17 Operaciona jedinica (drugi deo)

Registar $MAR_{15...0}$ je 16-to razredni adresni registar čiji se sadržaj koristi kao adresa memorijske lokacije memorije ***MEM*** sa koje se čita jedan bajt. Adresa se iz nekog od blokova operacione jedinice ***oper*** preko multipleksera MX1 vodi na ulaze registra $MAR_{15...0}$ i u njega upisuje vrednošću 1 signala **ldMAR**. Sadržaj registra $MAR_{15...0}$ se inkrementira vrednošću 1 signala **incMAR**. Ovo se koristi u situacijama kada treba pročitati 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. Sadržaj registra $MAR_{15...0}$ se vodi na adresne linije $A_{15...0}$ memorije ***MEM***.

Multiplekser MX1 je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 3 multipleksera se vode sadržaji $GPR_{15...0}$, $IR_{15...0}$, $DW_{15...0}$, i $ADD_{15...0}$, a selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 3, respektivno, upravljačkih signala $\mathbf{mxMAR_1}$ i $\mathbf{mxMAR_0}$. 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 saglasno pravilima formiranja adrese za data adresiranja. Selektovani sadržaja sa izlaza multipleksera MX1 se vodi na paralelne ulaze registra $MAR_{15...0}$.

Registar $MDR_{7...0}$ je 8-mo razredni prihvatni registar podatka. Pri čitanju iz memorije ***MEM*** se vrednošću 1 signala **ldMDR** sadržaj sa izlaznih linija podataka $DO_{7...0}$ memorije ***MEM*** upisuje u registar $MDR_{7...0}$. Sadržaj registra $MDR_{7...0}$ se vodi na ulaze 8-mo razrednih registara $DW_{15...8}$ i $DW_{7...0}$, kao i multipleksera MX5.

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 $DO_{7...0}$ 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 $DO_{7...0}$ magistrale. 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 $MAR_{15...0}$.

Registri opšte namene GPR su realizovani kao registarski fajl sa 32 registra širine 16 bita. Adresa registra opšte namene koji se čita određena je sadržajem na izlazu multipleksera MX4 koji se vodi na adresne linije $A_{4...0}$ registarskog fajla GPR. To je ili sadržaj razreda $IR_{20...16}$ ili sadržaj sa izlaza inkrementera INC. Sadržaj koji se čita $GPR_{15...0}$ pojavljuje se na izlaznim linijama podataka $DO_{15...0}$ registarskog fajla. Vrednošću 0 signala **wrGPR** na upravljačkoj liniji WR registarskog fajla realizuje se čitanje iz registarskog fajla.

Inkrementer INC na svojim izlazima daje sadržaj razreda $IR_{20...16}$ uvećan za 1, što se koristi samo ukoliko se radi o bazno indeksnom sa pomerajem adresiranju. Tada se registar opšte namene čija je adresa zadata razreda $IR_{20...16}$ drugog bajta instrukcije koristi kao bazni registar, a registar opšte namene sa prve sledeće adrese kao indeksni registar.

Multiplekser MX4 je 5-to razredni multiplekser sa 2 ulaza. Na ulaze 0 i 1 multipleksera se vode sadržaji sa izlaza registra $IR_{20...16}$ i inkrementera INC. Selekcija jednog od ova dva sadržaja se realizuje binarnim vrednostima 0 i 1, respektivno, upravljačkog signala **mxGPRADR**. Kod svih adresiranja kod kojih se koristi neki od registara opšte namene, sem bazno indeksnog sa pomerajem, na izlaze multipleksera se pušta sadržaj $IR_{20...16}$. Kod bazno indeksnog sa pomerajem adresiranja prvo se pušta sadržaj sadržaj $IR_{20...16}$ koji predstavlja adresu registra opšte namene koji se koristi kao bazni registar, a zatim sadržaj sa izlaza inkrementera INC koji predstavlja adresu registra opšte namene koji se koristi kao indeksno registar.

Sabirač ADD je 16-to razredni sabirač koji se koristi za formiranje16-to bitne vrednosti koja predstavlja adresu sa koje treba da se pročita ili na kojoj treba da se upiše operand ili adresu 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 preko multipleksera MX1 vodi u adresni registar $MAR_{15...0}$. Sadržaj $ADD_{15...0}$ sa izlaza sabirača se upisuje i 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 $MAR_{15...0}$.

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 vrednošću 1 signala **ldCW**. Sadržaj registra $CW_{15...0}$ se vodi na ulaze multipleksera MX2.

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. U bloku *addr* se koristi kod PC relativnog adresiranja.

Multiplekser MX2 je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 2 multipleksera se vode sadržaji $GPR_{15...0}$, $PC_{15...0}$ i $CW_{15...0}$. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 0 do 2 upravljačkih signala $\mathbf{mxADDA_1}$ i $\mathbf{mxADDA_0}$. Sadržaj sa izlaza multipleksera MX2 se vodi na ulaze $A_{15...0}$ sabirača ADD.

Multiplekser MX3 je 16-to razredni multiplekser sa 2 ulaza. Na ulaze 0 i 1 multipleksera se vode sadržaji $IR_{15...0}$ i $GPR_{15...0}$. Selekcija jednog od ova dva sadržaja se realizuje binarnim vrednostima 0 i 1 upravljačkog signala $\mathbf{mxADDB_0}$. Sadržaj sa izlaza multipleksera MX3 se vodi na ulaze $B_{15...0}$ sabirača ADD.

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.

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 vrednošću 1 signala **ldBB**.

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 **mxBB$_1$** i **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.

Kombinacione mreže za generisanje signala logičkih uslova operacija i adresiranja generišu signale se prema izrazima datim u tabelama 10 i 11, respektivno.

Tabela 10 Signali operacija

ST = $\overline{\mathbf{IR_{31}}} \cdot \overline{\mathbf{IR_{30}}} \cdot \overline{\mathbf{IR_{29}}} \cdot \overline{\mathbf{IR_{28}}} \cdot \mathbf{IR_{27}} \cdot \overline{\mathbf{IR_{26}}} \cdot \overline{\mathbf{IR_{25}}} \cdot \mathbf{IR_{24}}$

JADR = $\overline{\mathbf{IR_{31}}} \cdot \overline{\mathbf{IR_{30}}} \cdot \overline{\mathbf{IR_{29}}} \cdot \overline{\mathbf{IR_{28}}} \cdot \mathbf{IR_{27}} \cdot \mathbf{IR_{26}} \cdot \overline{\mathbf{IR_{25}}} \cdot \overline{\mathbf{IR_{24}}}$

Tabela 11 Signali adresiranja

regdir = $\overline{\mathbf{IR_{23}}} \cdot \overline{\mathbf{IR_{22}}} \cdot \overline{\mathbf{IR_{21}}}$

regind = $\overline{\mathbf{IR_{23}}} \cdot \overline{\mathbf{IR_{22}}} \cdot \mathbf{IR_{21}}$

memdir = $\overline{\mathbf{IR_{23}}} \cdot \mathbf{IR_{22}} \cdot \overline{\mathbf{IR_{21}}}$

memind = $\overline{\mathbf{IR_{23}}} \cdot \mathbf{IR_{22}} \cdot \mathbf{IR_{21}}$

regindpom = $\mathbf{IR_{23}} \cdot \overline{\mathbf{IR_{22}}} \cdot \overline{\mathbf{IR_{21}}}$

bxpom = $\mathbf{IR_{23}} \cdot \overline{\mathbf{IR_{22}}} \cdot \mathbf{IR_{21}}$

pcrel = $\mathbf{IR_{23}} \cdot \mathbf{IR_{22}} \cdot \overline{\mathbf{IR_{21}}}$

immed = $\mathbf{IR_{23}} \cdot \mathbf{IR_{22}} \cdot \mathbf{IR_{21}}$

Signali operacija ST i JADR (tabela 10) vrednošću 1 ukazuju da se radi o instrukciji ST ili instrukciji JADR, a koriste se za formiranje signala **noread** = **ST**+**JADR**. Signal **noread** vrednošću 1 ukazuje da se radi o instrukcijama kod kojih se sa sračunate adrese memorijske lokacije ne čita operand, već se adresa koristi u bloku *exec*.

Signali adresiranja **regdir**, **regind**, **memdir**, **memind**, **regindpom**, **bxpom**, **bcpom** i **imm** (tabela 11), od kojih u datom trenutku samo jedan ima vrednost 1, ukazuju kojim načinom adresiranja se dolazi do operanda.

Dijagrami toka mikrooperacija i upravljačkih signala i sekvenca upravljačkih signala

Dijagrami toka mikrooperacija i upravljačkih signala dati su na slikama 18, 19 i 20, a sekvenca upravljačkih signala u tabeli 12.

Tabela 12 Sekvenca upravljačkih signala

! Formiranje adrese i čitanje operanda !

! Kroz korake bloka *addr* prolaze jedino adresne instrukcije. U koracima bloka *addr* se saglasno specificiranom načinu adresiranja za sve instrukcije, sem instrukcija ST i JMPADR, čita operand i smešta u registar $BB_{7...0}$. Operand može da bude u nekom od registara opšte namene, u memorijskoj lokaciji ili samoj instrukciji. U slučaju direktnog registarskog adresiranja ili neposrednog adresiranja, kod kojih se operand nalazi u nekom od registara opšte namene ili u instrukciji, prolaz instrukcije kroz blok *addr* se svodi na prebacivanje operanda iz registra opšte namene ili iz prihvatnog registra instrukcije u registar $BB_{7...0}$. U slučaju adresiranja kod kojih se operand nalazi u memoriji, prolazak instrukcije kroz blok *addr* se sastoji od koraka u kojima se prvo formira adresa operanda u memoriji i zatim operand čita i prebacuje u registar $BB_{7...0}$. Izuzetak su instrukcije ST i JADR. U slučaju instrukcije ST sa registarskim direktnim adresiranjem zaustavlja se blok *addr* i startuje blok za

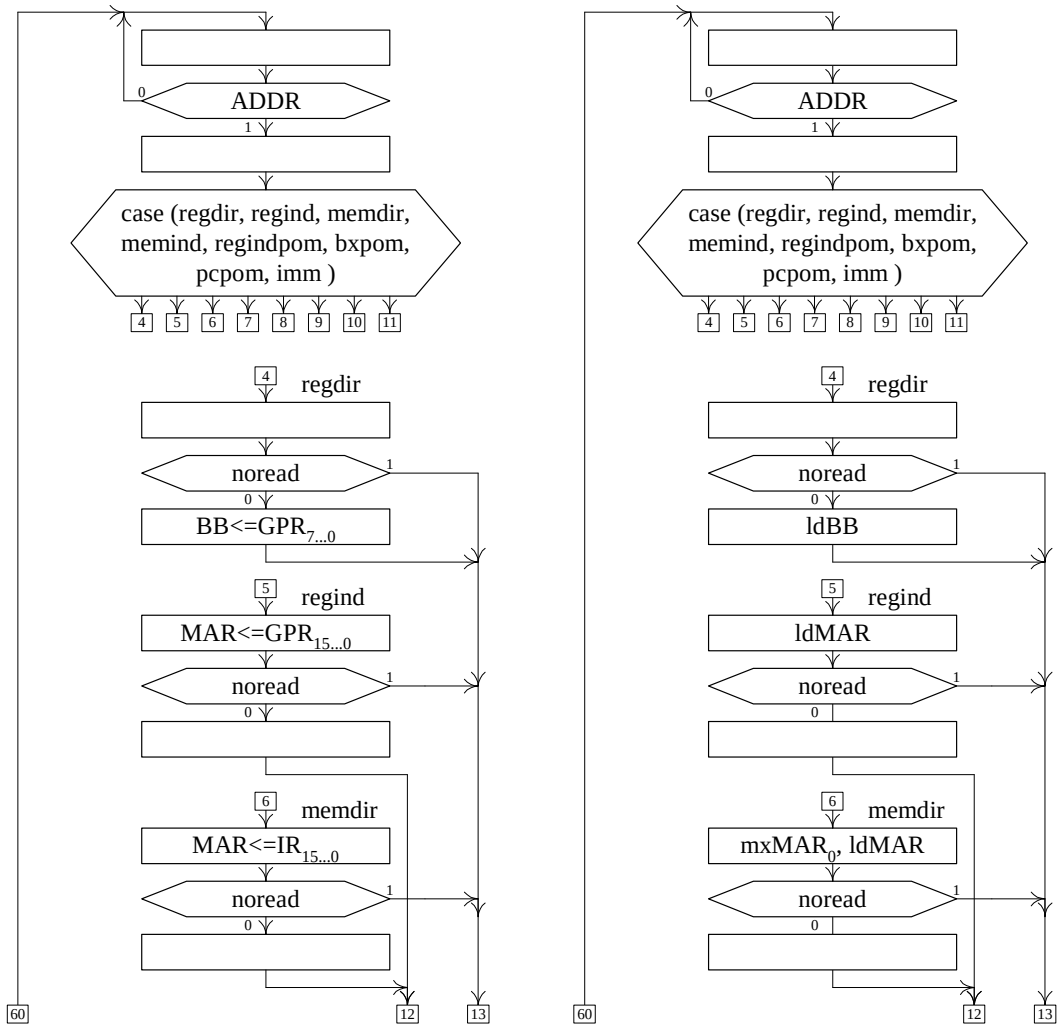
izvršavanje operacije *exec* u kome se sadržaj registra akumulatora $AB_{7...0}$ upisuje u registar opšte namene. U slučaju instrukcije ST sa nekim od memorijskih adresiranja u bloku *addr* se samo formira adresa operanda i smešta u registr $MAR_{15...0}$, pa se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec* u kome se sadržaj registra akumulatora $AB_{7...0}$ upisuje u memoriju na formiranoj adresi. Pretpostavljeno je da se u slučaju instrukcije ST neće javiti neposredno adresiranje. U slučaju instrukcije JADR sa nekim od memorijskih adresiranja, u ovoj fazi se samo formira adresa operanda i smešta u registr $MAR_{15...0}$, pa se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec* u kome se sadržaj registra $MAR_{15...0}$ upisuje u programski brojač $PC_{15...0}$. Pretpostavljeno je da se slučaju instrukcije JADR neće javiti neposredno adresiranje ili direktno registarsko adresiranje. !

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

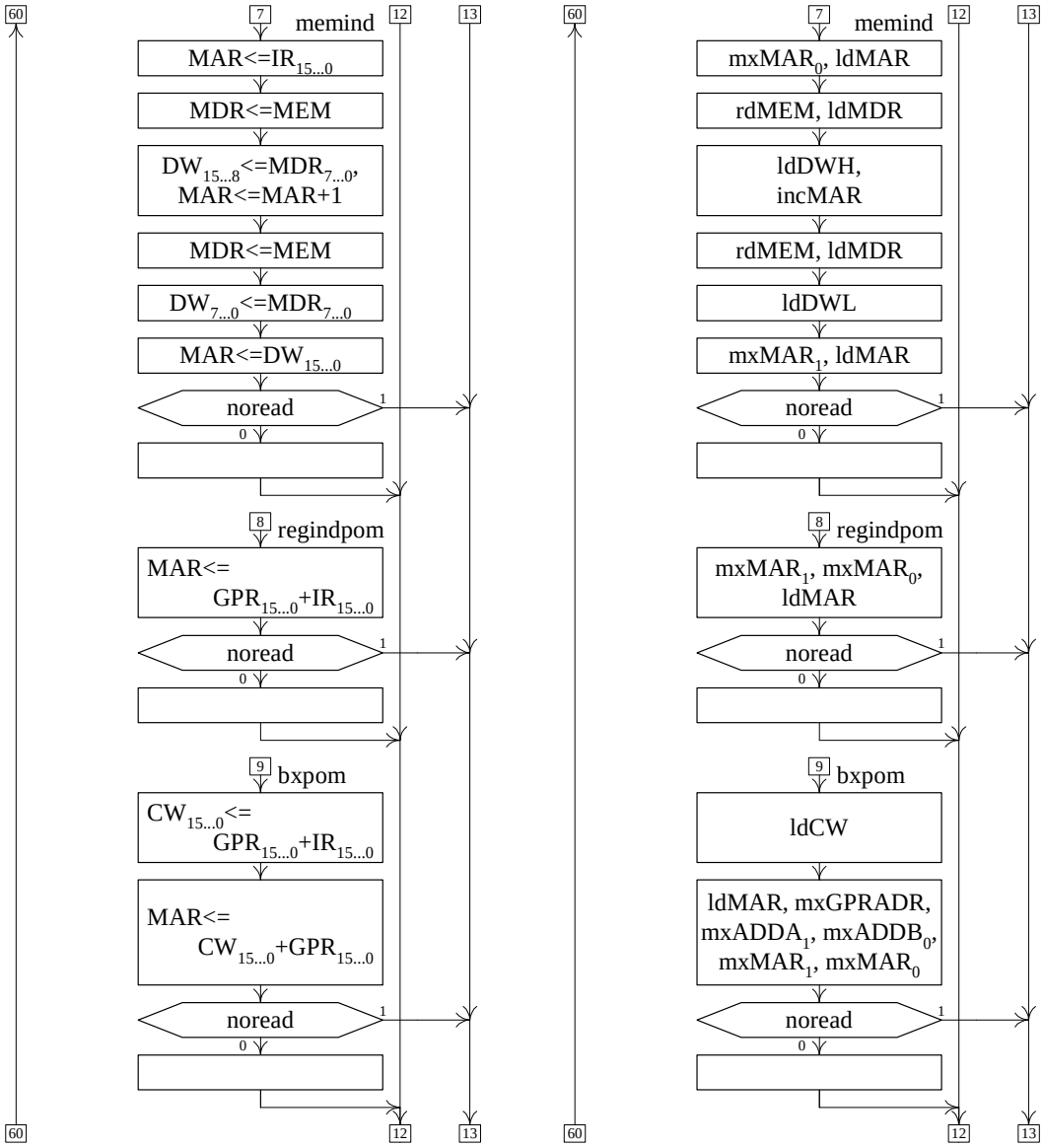
$step_{00}$ *br* (*if* $\overline{\textbf{ADDR}}$ *then* $step_{00}$);

! U koraku $step_{01}$ se realizuje višestruki uslovni skok na jedan od koraka $step_{02}$, $step_{04}$, …, $step_{17}$ u zavisnosti od toga koji od signala adresiranja **regdir**, **regind**, **memdir**, **memind**, **regindpom**, **bxpom**, **bcpom**, **imm** bloka *addr* ima vrednost 1. !

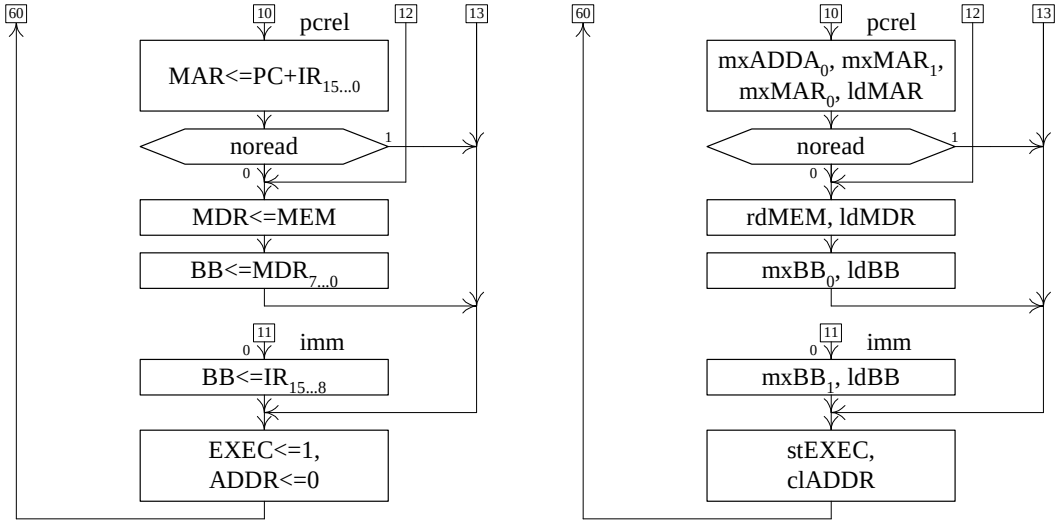
$step_{01}$ *br* (*case* (**regdir**, **regind**, **memdir**, **memind**, **regindpom**, **bxpom**, **bcpom**, **imm**) *then* (**regdir**, $step_{02}$), (**regind**, $step_{04}$), (**memdir**, $step_{06}$), (**memind**, $step_{08}$), (**regindpom**, $step_{0F}$), (**bxpom**, $step_{11}$), (**pcpom**, $step_{14}$), (**imm**, $step_{17}$));



Slika 18 Dijagrami toka mikrooperacija i upravljačkih signala (prvi deo)



Slika 19 Dijagrami toka mikrooperacija i upravljačkih signala (drugi deo)



Slika 20 Dijagrami toka mikrooperacija i upravljačkih signala (treći deo)

! Registarsko direktno adresiranje !
! U korak $step_{02}$ se dolazi iz $step_{01}$ ukoliko signal za registarsko direktno adresiranje **regdir** ima vrednost 1. Ukoliko signal **noread** ima vrednost 1, što znači da se radi o instrukcijama ST ili JADR za koje nema čitanja operanda, prelazi se na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*. U suprotnom slučaju se prelazi na korak $step_{03}$ u kome se vrednostima 0 signala $\mathbf{mxBB_1}$ i $\mathbf{mxBB_0}$ i vrednošću 1 signala **ldBB** niži bajt adresiranog registra opšte namene $GPR_{7...0}$ propušta kroz multiplekser MX5 upisuje u registar $BB_{7...0}$ i prelazi na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*. !

$step_{02}$ *br* (*if* **noread** *then* $step_{18}$);
$step_{03}$ **ldBB**,
br $step_{18}$;

! Registarsko indirektno adresiranje !
! U korak $step_{04}$ se dolazi iz koraka $step_{01}$ ukoliko signal za registarsko indirektno adresiranje **regind** ima vrednost 1. Vrednostima 0 signala $\mathbf{mxMAR_1}$ i $\mathbf{mxMAR_0}$ se sadržaj adresiranog registra opšte namene $GPR_{15...0}$ propušta kroz multiplekser MX1 i 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. Ukoliko signal **noread** ima vrednost 1, što znači da se radi o instrukcijama ST ili JADR za koje nema čitanja operanda, prelazi se na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*. U suprotnom slučaju se prelazi na korak $step_{05}$ iz koga se bezuslovno prelazi na korak $step_{15}$ i čitanje operanda. !

$step_{04}$ **ldMAR**,
br (*if* **noread** *then* $step_{18}$);
$step_{05}$ *br* $step_{15}$;

! Memorijsko direktno adresiranje !
! U korak $step_{06}$ se dolazi iz koraka $step_{01}$ ukoliko signal za memorijsko direktno adresiranje **memdir** ima vrednost 1. Vrednošću 1 signala $\mathbf{mxMAR_0}$ se sadržaj registra $IR_{15...0}$ propušta kroz multiplekser MX1 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 **noread** ima vrednost 1, što znači da se radi o instrukcijama ST ili JADR za koje nema čitanja operanda, prelazi se na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*. U suprotnom slučaju se prelazi na korak $step_{07}$ iz koga se bezuslovno prelazi na korak $step_{15}$ i čitanje operanda. !

$step_{06}$ $\mathbf{mxMAR_0}$, **ldMAR**,
br (*if* **noread** *then* $step_{18}$);
$step_{07}$ *br* $step_{15}$;

! Memorijsko indirektno adresiranje !
! U korak $step_{08}$ se dolazi iz $step_{01}$ ukoliko signal za memorijsko indirektno adresiranje **memind** ima vrednost 1. Vrednošću 1 signala $\mathbf{mxMAR_0}$ se sadržaj registra $IR_{15...0}$ propušta kroz multiplekser MX1 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. U koraku $step_{09}$ se realizuje čitanje prvog bajta i upisivanje u registar podatka $MDR_{7..0}$. Vrednošću 1 signala **rdMEM** se sa adrese određene sadržajem adresnog registra $MAR_{15..0}$, koji je povezan na adresne linije $A_{15..0}$ memorije ***MEM***, u memoriji ***MEM*** realizuje operacija čitanja. Pročitani sadržaj, koji memorija ***MEM*** pušta na izlazne linije podataka $DO_{7...0}$, se vrednošću 1 signala **ldMDR** upisuje u registar podatka $MDR_{7..0}$ i prelazi na korak $step_{0A}$. U koraku $step_{0A}$ se iz registra $MDR_{7...0}$ vrednošću 1 signala **ldDWH** prvi bajt upisuje u viši bajt registra $DW_{15...8}$, a vrednošću 1 signala **incMAR** adresni registar $MAR_{15...0}$ inkrementira na adresu sledećeg bajta. U koraku $step_{0B}$ se realizuje čitanje brugog bajta i upisivanje u registar podatka $MDR_{7..0}$. To se radi na isti način kao i u koraku $step_{09}$ kada se realizuje čitanje prvog bajta. U koraku $step_{0C}$ se iz registra $MDR_{7...0}$ vrednošću 1 signala **ldDWL** drugi bajt upisuje u niži bajt registra $DW_{7...0}$. Na kraju se u koraku $step_{0D}$ vrednošću 1 signala $\mathbf{mxMAR_1}$ sadržaj registra $DW_{15...0}$ koji predstavlja adresu operanda propušta kroz multiplekser MX1 i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$. Ukoliko signal **noread** ima vrednost 1, što znači da se radi o instrukcijama ST ili JADR za koje nema čitanja operanda, prelazi se na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok

za izvršavanje operacije *exec*. U suprotnom slučaju se prelazi na korak $step_{0E}$ iz koga se bezuslovno prelazi na korak $step_{15}$ i čitanje operanda. !

$step_{08}$ **mxMAR$_0$**, **ldMAR**;
$step_{09}$ **rdMEM**, **ldMDR**;
$step_{0A}$ **ldDWH**, **incMAR**;
$step_{0B}$ **rdMEM**, **ldMDR**;
$step_{0C}$ **ldDWL**;
$step_{0D}$ **mxMAR$_1$**, **ldMAR**,
br (*if* **noread** *then* $step_{18}$);
$step_{0E}$ *br* $step_{15}$;

! Registarsko indirektno adresiranje sa pomerajem !

! U korak $step_{0F}$ se dolazi iz $step_{01}$ ukoliko signal za registarsko indirektno adresiranje sa pomerajem **regindpom** ima vrednost 1. Vrednostima 0 signala **mxADDA$_1$**, **mxADDA$_0$** i **mxADDB$_0$** se najpre kroz multipleksere MX2 i MX3 na ulaze sabirača ADD propuštaju adresirani registar opšte namene $GPR_{15...0}$ i pomeraj iz $IR_{15...0}$, zatim se vrednostima 1 signala **mxMAR$_1$** i **mxMAR$_0$** dobijeni sadržaj sa izlaza sabirača ADD propušta kroz multiplekser MX1 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 **noread** ima vrednost 1, što znači da se radi o instrukcijama ST ili JADR za koje nema čitanja operanda, prelazi se na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*. U suprotnom slučaju se prelazi na korak $step_{10}$ iz koga se bezuslovno prelazi na korak $step_{15}$ i čitanje operanda. !

$step_{0F}$ **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**,
br (*if* **noread** *then* $step_{18}$);
$step_{10}$ *br* $step_{15}$;

! Bazno indeksno adresiranje sa pomerajem !

! U korak $step_{11}$ se dolazi iz koraka $step_{01}$ ukoliko signal za bazno indeksno adresiranje sa pomerajem **bxpom** ima vrednost 1. U koraku $step_{11}$ se vrednostima 0 signala **mxADDA$_1$**, **mxADDA$_0$** i **mxADDB$_0$** se kroz multipleksere MX2 i MX3 na ulaze sabirača ADD propuštaju adresirani registar opšte namene $GPR_{15...0}$ koji se koristi kao bazni registar i pomeraj iz $IR_{15...0}$, a vrednošću 1 signala **ldCW** sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD upisuje u registar $CW_{15...0}$. U koraku $step_{12}$ se vrednošću 1 signala **mxGPRADR** kroz multiplekser MX4 na adresne linije $A_{4...0}$ registarskog fajla GPR propušta sadržaj razreda $IR_{20...16}$ uvećan u inkrementeru INC za 1, vrednostima 1 signala **mxADDA$_1$** i **mxADDB$_0$** kroz multipleksere MX2 i MX3 na ulaze sabirača ADD propuštaju sadržaji registra $CW_{15...0}$ i adresiranog registra opšte namene $GPR_{15...0}$ koji se koristi kao indeksni registar, vrednostima 1 signala **mxMAR$_1$** i **mxMAR$_0$** se dobijeni sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD propušta kroz multiplekser MX1 i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj bazno indeksnog adresiranja sa pomerajem. Ukoliko signal **noread** bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama ST ili JADR za koje nema čitanja operanda, prelazi se na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*. U suprotnom slučaju se prelazi na korak $step_{13}$ iz koga se bezuslovno prelazi na korak $step_{15}$ i čitanje operanda. !

$step_{11}$ **ldCW**;
$step_{12}$ **mxGPRADR**, **mxADDA$_1$**, **mxADDB$_0$**, **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**,
br (*if* **noread** *then* $step_{18}$);
$step_{13}$ *br* $step_{15}$;

! PC relativno adresiranje !

! U korak $step_{14}$ se dolazi iz koraka $step_{01}$ ukoliko signal za PC relativno adresiranje **pcpom** ima vrednost 1. Vrednostima 0, 1 i 0 signala **mxADDA$_1$**, **mxADDA$_0$** i **mxADDB$_0$** se kroz multipleksere MX2 i MX3 na ulaze sabirača ADD propuštaju programski brojač $PC_{15...0}$ i pomeraj iz $IR_{15...0}$, vrednostima 1 signala **mxMAR$_1$** i **mxMAR$_0$** se dobijeni sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD propušta kroz multiplekser MX1 i vrednošću 1 signala **ldMAR** upisuje u registar $MAR_{15...0}$. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj PC relativnog adresiranja. Ukoliko signal **noread**

bloka *addr* ima vrednost 1, što znači da se radi o instrukcijama ST ili JADR za koje nema čitanja operanda, prelazi se na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*. U suprotnom slučaju se prelazi na korak $step_{15}$ i čitanje operanda. !

$step_{14}$ **$mxADDA_0$**, **$mxMAR_1$**, **$mxMAR_0$**, **ldMAR**,
br (*if* **noread** *then* $step_{18}$);

! Čitanje operanda !

! U korak $step_{15}$ se dolazi iz $step_{05}$ kod registarskog indirektnog adresiranja, iz $step_{07}$ kod memorijskog direktnog adresiranja, iz $step_{0E}$ kod memorijskog indirektnog adresiranja, iz $step_{10}$ kod registarskog indirektnog adresiranja sa pomerajem, iz $step_{13}$ kod baznog indeksnog adresiranja sa pomerajem i iz $step_{14}$ 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 $MAR_{15...0}$. Za sve instrukcije za koje se čita operand iz memorije dužina operanda je jedan bajt. Zbog toga se u koraku $step_{15}$ čita jedan bajt i to na isti način na koji se to radi u koraku $step_{09}$. Zatim se u koraku $step_{16}$ vrednostima 1 signala **$mxBB_0$** i **ldBB** sadržaj registra $MDR_{7...0}$ propušta kroz multiplekser MX5 i upisuje u registar $BB_{7...0}$. Iz koraka $step_{16}$ se prelazi na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*. !

$step_{15}$ **rdMEM**, **ldMDR**;
$step_{16}$ **$mxBB_0$**, **ldBB**,
br $step_{18}$;

! Neposredno adresiranje !

! U korak $step_{17}$ se dolazi iz $step_{01}$ ukoliko signal za neposredno adresiranje **imm** ima vrednost 1. U koraku $step_{17}$ se vrednostima 1 signala **$mxBB_1$** i **ldBB** 8-mo razredna neposredna veličina koja se nalazi u razredima $IR_{15...8}$ propušta kroz multiplekser MX5 i upisuje u registar $BB_{7...0}$ i prelazi na korak $step_{18}$ u kome se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec*.. !

$step_{17}$ **$mxBB_1$**, **ldBB**;

! U korak $step_{18}$ se dolazi iz koraka $step_{02}$, $step_{03}$, $step_{04}$, $step_{06}$, $step_{0D}$, $step_{0F}$, $step_{12}$, $step_{14}$, $step_{16}$ i $step_{17}$. U koraku $step_{18}$ se vrednošću 1 signala **clADDR** upisuje vrednost 0 u flip flop ADDR bloka *addr*, čime se zaustavlja blok *addr*, dok se vrednošću 1 signala **stEXEC** upisuje vrednost 1 u flip flop EXEC bloka *exec*, čime se startuje blok *exec*. !

$step_{18}$ **clADDR, stEXEC**,
br $step_{00}$;

Upravljačka jedinica

Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu sekvence upravljačkih signala po koracima (tabela 12). Mikroprogram se formira tako što se svakom koraku u sekvenci upravljačkih signala po koracima pridruži binarna reč sa slike 21. 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
ldMAR	incMAR	$mxMAR_1$	$mxMAR_0$	rdMEM	ldMDR	ldDWH	ldDWL

8	9	10	11	12	13	14	15
ldCW	$mxADDB_0$	$mxADDA_1$	$mxADDA_0$	-	ldBB	$mxBB_1$	$mxBB_0$

16	17	18	19	20	21	22	23
mxGPRADR	wrGPR	clADDR	stEXEC	*cc*			

24	25	26	27	28	29	30	31
ba							

Slika 21 Mikroinstrukcija

Mikroinstrukcija ima dva dela i to operacini deo i upravljački deo. Operacioni deo čine bitovi 0 do 19, a upravljački deo čine bitovi 20 do 31. 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.

Bezuslovni skokovi se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 12) 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 24.

Tabela 13 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 14.

Tabela 14 Signali uslovnih skokova

signal uslovnog skoka	polje *cc*	signal uslova
brnotADDR	2	$\overline{\textbf{ADDR}}$
brnoread	3	**noread**

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 15.

Tabela 15 Signali višestrukih uslovnih skokova

signal višestrukog uslovnog skoka	polje *cc*	korak
bradr	4	$step_{01}$

Vrednosti 0, 5, 6, …, F polja *cc* koje nisu dodeljene signalu bezuslovnog skoka (tabela 24), signalima uslovnih skokova (tabela 14) i signalu višestrukog uslovnog skoka (tabela 15) određuju 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 12). 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 32 bitova. Za kodiranje operacionog dela mikroinstrukcije koristi se 24 bita. Upravljačkih signala operacione jedinice ima 23, ali je umesto 23 bita usvojena dužina operacionog dela mikroinstrukcije 24 bitova. Za kodiranje upravljačkog dela mikroinstrukcije koristi se 12 bitova. Signala bezuslovnog skoka, signala uslovnih skokova i signala višestrukih uslovnih skokova ima 4, ali je umesto 2 bita usvojena dužina polja *cc* upravljačkog dela mikroinstrukcije 4 bita. Ukupan broj koraka u sekvenci upravljačkih signala po koracima je 25, ali je umesto 5 bita usvojena dužina polja *ba* upravljačkog dela mikroinstrukcije 8 bitova. Ovo je učinjeno da bi se dobio pregledniji mikroprogram predstavljen u heksadecimalnom obliku.

Mikroprogram se formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima (tabela 12) 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* $step_A$) ili *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 24. 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 16.

Tabela 16 Koraci $step_i$, $step_A$, vrednosti **$madr_A$** i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	**$madr_A$**	A
$step_{03}$	$step_{18}$	**$madr_{18}$**	18
$step_{05}$	$step_{15}$	**$madr_{15}$**	15
$step_{07}$	$step_{15}$	**$madr_{15}$**	15
$step_{0E}$	$step_{15}$	**$madr_{15}$**	15

$step_i$	$step_A$	**$madr_A$**	A
$step_{10}$	$step_{15}$	**$madr_{15}$**	15
$step_{13}$	$step_{15}$	**$madr_{15}$**	15
$step_{16}$	$step_{18}$	**$madr_{18}$**	18
$step_{18}$	$step_{00}$	**$madr_{00}$**	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 14. 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 17.

Tabela 17 Koraci $step_i$, uslovi **uslov**, koraci $step_A$, vrednosti $\mathbf{madr_A}$ i vrednosti A za uslovne skokove

$step_i$	uslov	$step_A$	$\mathbf{madr_A}$	A
$step_{00}$	$\overline{\mathbf{ADDR}}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{02}$	**noread**	$step_{18}$	$\mathbf{madr_{18}}$	18
$step_{04}$	**noread**	$step_{18}$	$\mathbf{madr_{18}}$	18
$step_{06}$	**noread**	$step_{18}$	$\mathbf{madr_{18}}$	18
$step_{0D}$	**noread**	$step_{18}$	$\mathbf{madr_{18}}$	18

$step_i$	uslov	$step_A$	$\mathbf{madr_A}$	A
$step_{0F}$	**noread**	$step_{18}$	$\mathbf{madr_{18}}$	18
$step_{12}$	**noread**	$step_{18}$	$\mathbf{madr_{18}}$	18
$step_{14}$	**noread**	$step_{18}$	$\mathbf{madr_{18}}$	18
$step_{16}$	**noread**	$step_{18}$	$\mathbf{madr_{18}}$	18

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 15. Signali uslova **uslov**$_1$, ..., **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 **uslov**$_1$, ..., **uslov**$_n$ ima vrednost 1 i vrednosti A1,..., An od kojih jednu treba upisati u mikroprogramski brojač za jedan iskaz ovog tipa koji se javlja u sekvenci upravljačkih signala po koracima dati su u tabeli 18.

Tabela 18 Signali uslova, koraci na koje se skače i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{01}$

uslov	$step_A$	A
regdir	$step_{02}$	02
regind	$step_{04}$	04
memdir	$step_{06}$	06
memind	$step_{08}$	08
regindpom	$step_{0F}$	0F
bxpom	$step_{11}$	11
pcpom	$step_{14}$	14
imm	$step_{17}$	17

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu mikroprogramske realizacije signal bezuslovnog skoka (tabela 24), signali uslovnih skokova (tabela 14), signali višestrukih uslovnih skokova (tabela 15) i signali vrednosti A za bezuslovne skokokve (tabela 16), uslovne skokove (tabela 17) i višestruke uslovne skokove (tabela 18).

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 12) formiran mikroprogram (tabela 19). 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 .

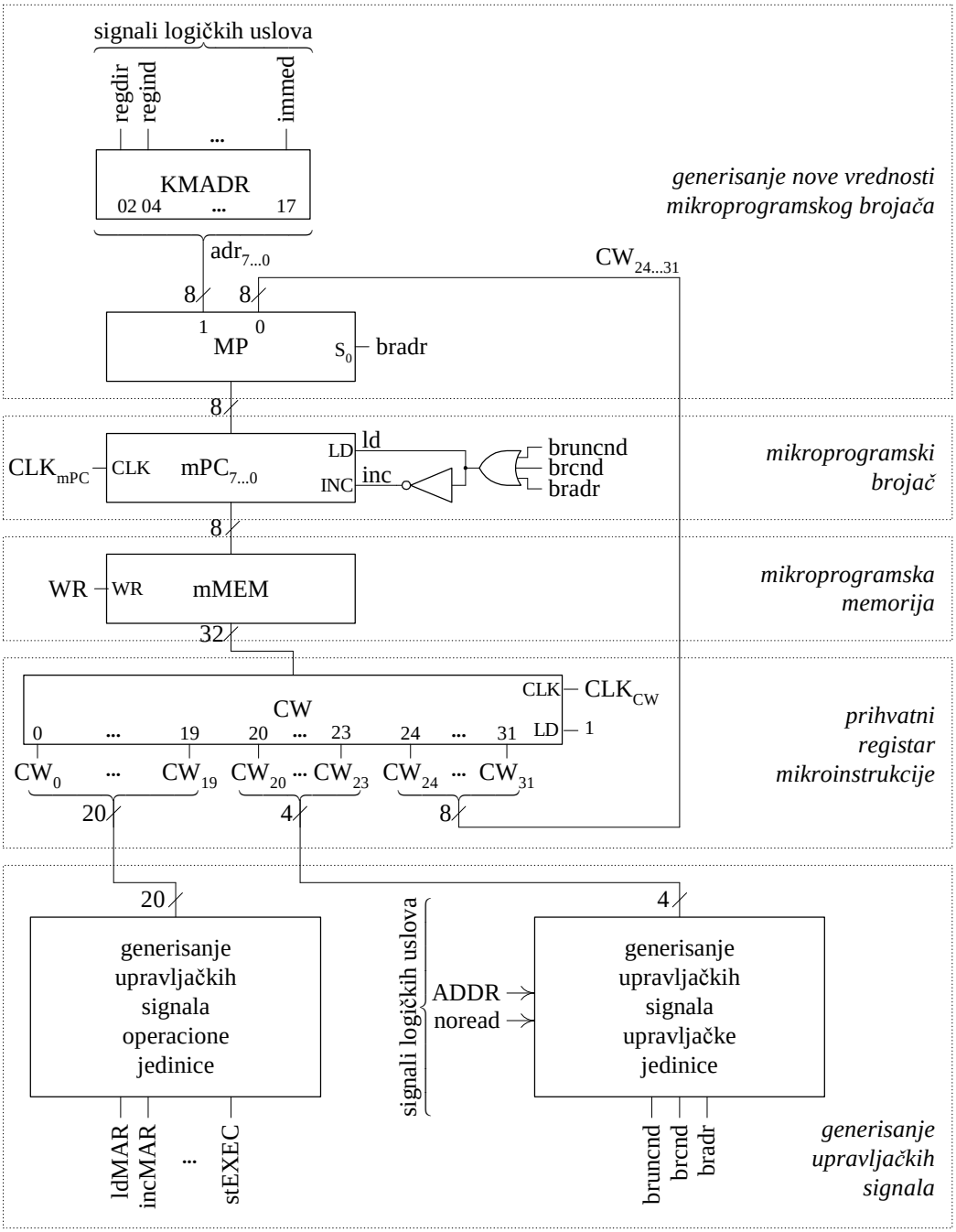
Tabela 19 Mikroprogram za upravljačku jedinicu mikroprogramske realizacije

0 0	0 0 0 0 0 2 0 0	! **brnotADDR**, **madr$_{00}$** !
0 1	0 0 0 0 0 4 0 0	! **bradr** !
0 2	0 0 0 0 0 3 1 8	! **brnoread**, **madr$_{18}$**!
0 3	0 0 0 4 0 1 1 8	! **ldBB, bruncnd, madr$_{18}$** !
0 4	8 0 0 0 0 3 1 8	! **ldMAR**, **brnoread**, **madr$_{18}$** !
0 5	0 0 0 0 0 1 1 5	! **bruncnd, madr$_{15}$** !
0 6	9 0 0 0 0 3 1 8	! **mxMAR$_0$, ldMAR, brnoread**, **madr$_{18}$** !
0 7	0 0 0 0 0 1 1 5	! **bruncnd, madr$_{15}$** !
0 8	9 0 0 0 0 0 0 0	! **mxMAR$_0$, ldMAR**!
0 9	0 C 0 0 0 0 0 0	! **rdMEM, ldMDR** !
0 A	4 2 0 0 0 0 0 0	! **ldDWH, incMAR** !
0 B	0 C 0 0 0 0 0 0	! **rdMEM, ldMDR** !
0 C	0 1 0 0 0 0 0 0	! **ldDWL** !
0 D	A 0 0 0 0 3 1 8	! **mxMAR$_1$, ldMAR, brnoread, madr$_{18}$,** !
0 E	0 0 0 0 0 1 1 5	! **bruncnd, madr$_{15}$** !
0 F	B 0 0 0 0 3 1 8	! **mxMAR$_1$, mxMAR$_0$, ldMAR, brnoread**, **madr$_{18}$** !
1 0	0 0 0 0 0 1 1 5	! **bruncnd, madr$_{15}$** !
1 1	0 0 8 0 0 0 0 0	! **ldCW** !
1 2	B 0 6 0 8 3 1 8	! **mxGPRADR**, **mxADDA$_1$**, **mxADDB$_0$**, **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**, **brnoread**, **madr$_{18}$** !
1 3	0 0 0 0 0 1 1 5	! **bruncnd, madr$_{15}$** !
1 4	B 0 1 0 0 3 1 8	**!** **mxADDA$_0$**, **mxMAR$_1$**, **mxMAR$_0$**, **ldMAR**, **brnoread**, **madr$_{18}$** !
1 5	0 C 0 0 0 0 0 0	**!** **rdMEM**, **ldMDR !**
1 6	0 0 0 5 0 1 1 8	**!** **mxBB$_0$**, **ldBB**, **bruncnd, madr$_{18}$** !
1 7	0 0 0 6 0 0 0 0	**!** **mxBB$_1$**, **ldBB !**
1 8	0 0 0 0 3 1 0 0	**!** **clADDR, stEXEC**, **bruncnd, madr$_{00}$** !

Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 22. 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.

Blok *generisanje nove vrednosti mikroprogramskog brojača* se sastoji od kombinacione mreže 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 dva načina i to pomoću kombinacione mreže KMADR koja formira signale **adr$_{7...0}$** i razreda $CW_{24...31}$ prihvatnog registra mikroinstrukcije $CW_{0...31}$. Selekcija jedne od dve grupe signala koje daju novu vrednost mikroprogramskog brojača obezbeđuje se

signalom **bradr** i to signali **adr$_{7...0}$** ako signal **bradr** ima vrednost 1 i signali CW$_{24...31}$ ako signal **bradr** ima vrednost 0.



Slika 22 Struktura upravljačke jedinice mikroprogramske realizacije

Kombinacionom mrežom KMADR generišu se vrednosti (tabela 18) za realizaciju višestrukog uslovnog skoka na adresi 01 mikroprograma (tabela 19). U zavisnosti od toga koji od signala **dirreg**, **indreg**,..., **immed** ima vrednost 1 zavisi koja će od vrednosti iz tabele 18 da se pojavi tada na linijama **adr$_{7...0}$**. S obzirom da se na adresi 01 mikroprograma nalazi mikroinstrukcija sa tako kodiranim poljem *cc* da njeno izvršavanje daje vrednost 1 signala višestrukog uslovnog skoka **bradr**, vrednost na linijama **adr$_{7...0}$** prolazi kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača mPC.

Prihvatni registar mikroinstrukcije CW$_{0...31}$ u svojim razredima CW$_{24...31}$ sadrži vrednost za upis u mikroprogramski brojač mPC$_{7...0}$ za bezuslovne skokove (tabela 16) i uslovne skokove (tabela 17) u mikroprogramu (tabela 19). Signal višestrukog uslovnog skok **bradr** ima

vrednost 1 samo prilikom izvršavanja mikroinstrukcije na adresi 01 mikroprograma, a u svim ostalim situacijama imaj vrednost 0. S obzirom da ovaj signal nema vrednost 1 prilikom izvršavanja mikroinstrukcija kojima se realizuju bezuslovni ili neki od uslovnih skokova u mikroprogramu, vrednost određena razredima $CW_{24...31}$ 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 19). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko svi signali **bradr**, **brcnd** i **bruncnd** imaju vrednost 0. Signali **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 19). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ukoliko jedan od signala **bradr**, **brcnd** i **bruncnd** ima vrednost 1. Signali **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 19). S obzirom da se mikroprogram svih faza izvršavanja instrukcija nalazi u opsegu od adrese 00 do adrese 18, 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 32 bita, a kapacitet veličinom mikroprograma (tabela 19) i iznosi 256 lokacija. Adresiranje mikroprogramske memorije se realizuje sadržajem mikroprogramskog brojača $mPC_{7...0}$.

Blok *prihvatni registar mikroinstrukcije* sadrži prihvatni registar mikroinstrukcije $CW_{0...23}$. Prihvatni registar mikroinstrukcije $CW_{0...31}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...19}$ i $CW_{20...23}$ se koriste u bloku *generisanje upravljačkih signala* za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{24...31}$ 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 $CW_{0...19}$ prihvatnog registra mikroinstrukcije generišu upravljačke signale operacione jedinice i na osnovu sadržaja razreda $CW_{20...23}$ prihvatnog registra mikroinstrukcije i signala

logičkih uslova **ADDR** i **noread** koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice ***oper*** se generišu na sledeći način:

- **ldMAR** = $\mathbf{CW_0}$
- **incMAR** = $\mathbf{CW_1}$
- $\mathbf{mxMAR_1} = \mathbf{CW_2}$
- $\mathbf{mxMAR_0} = \mathbf{CW_3}$
- **rdMEM** = $\mathbf{CW_4}$
- **ldMDR** = $\mathbf{CW_5}$
- **ldDWH** = $\mathbf{CW_6}$
- **ldDWL** = $\mathbf{CW_7}$
- **ldCW** = $\mathbf{CW_8}$
- $\mathbf{mxADDB_0} = \mathbf{CW_9}$
- $\mathbf{mxADDA_1} = \mathbf{CW_{10}}$
- $\mathbf{mxADDA_0} = \mathbf{CW_{11}}$
- **ldBB** = $\mathbf{CW_{13}}$
- $\mathbf{mxBB_1} = \mathbf{CW_{14}}$
- $\mathbf{mxBB_0} = \mathbf{CW_{15}}$
- **mxGPRADR** = $\mathbf{CW_{16}}$
- **wrGPR** = $\mathbf{CW_{17}}$
- **clADDR** = $\mathbf{CW_{18}}$
- **stEXEC** = $\mathbf{CW_{19}}$

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

- **bruncnd** = $\overline{\mathbf{CW_{20}}} \cdot \overline{\mathbf{CW_{21}}} \cdot \overline{\mathbf{CW_{22}}} \cdot \mathbf{CW_{23}}$
- **brcnd** = **brnotADDR** $\cdot$ $\overline{\mathbf{ADDR}}$ + **brnoread** $\cdot$ **noread**
- **brnotADDR** = $\overline{\mathbf{CW_{20}}} \cdot \overline{\mathbf{CW_{21}}} \cdot \mathbf{CW_{22}} \cdot \overline{\mathbf{CW_{23}}}$
- **brnoread**= $\overline{\mathbf{CW_{20}}} \cdot \overline{\mathbf{CW_{21}}} \cdot \mathbf{CW_{22}} \cdot \mathbf{CW_{23}}$
- **bradr**= $\overline{\mathbf{CW_{20}}} \cdot \mathbf{CW_{21}} \cdot \overline{\mathbf{CW_{22}}} \cdot \overline{\mathbf{CW_{23}}}$

Pri generisanju signala **branch** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice ***oper*** i to **ADDR** i **noread**.

3 ZADATAK 3

Posmatra se blok *exec* procesora u kome se realizuje faza izvršavanje operacije. Pored procesora u posmatranim računaru postoji i memorija. Memorija je kapaciteta 2^{16} bajtova. Širina memorijske reči je 1 bajt. Procesor je sa jednoadresnim formatom instrukcija. Podaci su celobrojne veličine bez znaka dužine jedan bajt.

Prvi bajt instrukcije sadrži polje koda operacije.

U procesoru postije bezadresne instrukcije, instrukcije uslovnog skoka, instrukcije bezuslovnog skoka i adresne instrukcije.

Bezadresne instrukcije su instrukcija stavljanja sadržaja akumulatora na stek (PUSH), instrukcija skidanja sadržaja sa steka u akumulator (POP), instrukcija povratka iz potprograma (RTS), instrukcija povratka iz prekidne rutine (RTI) i instrukcija aritmetičkog pomeranja sadržaja akumulatora udesno za jedno mesto (ASR). Za bezadresne instrukcije PUSH, POP, RTS i RTI i ASR su usvojeni kodovi operacija 00000000, 00000001, 00000010, 0000000011 i 00000100, respektivno. Dužina instrukcija je jedan bajt.

Instrukcija uslovnog skoka je instrukcija uslovnog skoka ukoliko je rezultat nula (BZ). Za instrukciju BZ je usvojen kod operacije 00000101. Instrukcija BZ se realizuje kao relativni skok u odnosu na tekuću vrednost programskog brojača PC, a pomeraj je 8-mo bitna celobrojna veličina sa znakom data drugim bajtom instrukcije. Dužina instrukcije je dva bajta.

Instrukcije bezuslovnog skoka su instrukcija bezuslovnog skoka (JMP) i instrukcija skoka na potprogram (JSR). Za instrukcije JMP i JSR su usvojeni kodovi operacija 00000110 i 00000111, respektivno. Instrukcije JMP i JSR se realizuju kao apsolutni skokovi, a adresa skoka je data drugim i trećim bajtom instrukcije, pri čemu je stariji bajt adrese skoka dat drugim bajtom instrukcije a mlađi bajt adrese skoka trećim bajtom instrukcije. Dužina instrukcija je tri bajta.

Adresne instrukcije su instrukcija prenosa u akumulator (LD), instrukcija prenosa iz akumulatora (ST), aritmetička instrukcija sabiranja (ADD), logička instrukcija logički proizvod (AND) i instrukcija bezuslovnog skoka na sračunatu adresu (JADR). U instrukciji ST nije dozvoljeno neposredno adresiranje, a u instrukciji JADR nije dozvoljeno direktno registarsko i neposredno adresiranje, pa ukoliko se jave ova adresiranja u ovim instrukcijama, instrukcije treba da budu bez dejstva. Za instrukcije LD, ST, ADD, AND, ASR i JADR usvojeni kodovi operacija 00001000, 00001001, 00001010, 00001011, 00001100 i 00001101, respektivno. Dužina instrukcija je dva, tri ili četiri bajta i zavisi od specificiranog načina adresiranja.

Za adresne instrukcije se bitovima 7, 6 i 5 drugog bajta instrukcije specificira načina adresiranja i to na sledeći način: 000-registarsko direktno adresiranje (regdir), 001-registarsko indirektno adresiranje (regind), 010-memorijsko direktno adresiranje (memdir), 011-memorijsko indirektno adresiranje (memind), 100-registarsko indirektno sa pomerajem adresiranje (regindpom), 101-bazno indeksno sa pomerajem adresiranje (bxpom), 110- PC relativno sa pomerajem adresiranje (pcrel) i 111-neposredno adresiranje (immed). Registarsko direktno i registarsko indirektno adresiranje koriste neki od registara opšte namene R[0] i R[31] specificiran bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcija je dva bajta. Neposredno adresiranje ima i treći bajt instrukcije u kome se nalazi operand, ali ne koristi

registre opšte namene R[0] i R[31] specificirane bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcije je tri bajta.

Memorijsko direktno, memorijsko indirektno, registarsko indirektno sa pomerajem adresiranje, bazno indeksno sa pomerajem adresiranje i PC relativno sa pomerajem adresiranje imaju treći i četvrti bajt instrukcije. Kod memorijskog direktnog i memorijskog indirektnog adresiranja treći i četvrti bajt instrukcije sadrže adresu memorijske lokacije, pri čemu je stariji bajt adrese dat trećim bajtom a mlađi bajt adrese četvrtim bajtom. Kod memorijskog indirektnog adresiranja adresa dužine 16 bita zauzima dve susedne memorijske lokacije, pri čemu je stariji bajt adrese nalazi na nižoj a mlađi bajt adrese na višoj lokaciji. Bitovi 4 do 0 drugog bajta instrukcije se ne koriste. Dužina instrukcija je četiri bajta. Kod registarskog indirektnog sa pomerajem adresiranje, bazno indeksnog sa pomerajem adresiranjem i kod PC relativnog adresiranja treći i četvrti bajt instrukcije sadrže pomeraj, pri čemu je stariji bajt pomeraja dat trećim bajtom a mlađi bajt pomeraja četvrtim bajtom. Kod registarskog indirektnog sa pomerajem adresiranja registara opšte namene R[0] i R[31] koji se koristi specificiran je bitovima 4 do 0 drugog bajta instrukcije. Dužina instrukcija je četiri bajta. Kod bazno indeksnog sa pomerajem adresiranjem kao bazni registar se koristi jedan od registara opšte namene R[0] i R[31] koji je specificiran bitovima 4 do 0 drugog bajta instrukcije, dok se kao indeksni registar koristi prvi sledeći registar opšte namene.

Procesor je tako realizovan da se faza čitanje instrukcije realizuje u bloku *fetch*, faza formiranje adrese i čitanje operanda u bloku *addr*, faza izvršavanje operacije u bloku *exec* i faza opsluživanje prekida u bloku *intr*. Svaka instrukcija prolazi kroz blokove neophodne za realizaciju određenih faza. U blok *exec* dolaze iz bloka *fetch* bezadresne instrukcije, instrukcije uslovnog skoka i instrukcije bezuslovnog skoka, dok iz bloka *addr* dolaze adresne instrukcije. Pretpostavlja se da se u bloku *exec* nalaze svi registri neophodni za izvršavanje faze izvršavanje operacije i da se njima nalaze informacije koje su rezultat prethodnih faza izvršavanja instrukcije. Od registara postoji programski brojač $PC_{15...0}$, adresni registar memorije $MAR_{15...0}$, prihvatni registar podatka memorije $MDR_{7...0}$, prihvatni registar instrukcije $IR_{31...0}$, ukazivač na vrh steka $SP_{15...0}$, registar akumulatora $AB_{15...0}$, prihvatni registar podatka $BB_{7...0}$, registarski fajl GPR i programska statusna reč $PSW_{15...0}$. U prihvatnom registru instrukcije $IR_{31...0}$ nalazi se očitana instrukcija, u prihvatnom registru podatka $BB_{7...0}$ očitani operand specificiran adresnim delom adresnih instrukcija LD, ADD i AND i u adresnim registru $MAR_{15...0}$ sračunata adresa memorijske lokacije neophodna za izvršavanje instrukcija ST i JAMP.

Blok *exec* kreće sa fazom izvršavanje operacije ukoliko se u flip-flopu EXEC nalazi vrednost 1. Po završenom izvršavanju operacije upisivanjem vrednosti 1 u flip-flop INTR startuje se blok *intr* produžava sa izvršavanjem faze opsluživanje prekida, dok se upisivanjem vrednost 0 u flip-flop EXEC zaustavlja blok *exec*.

Realizovati blok *exec* procesora. Operacione jedinica treba da bude realizovana direktnim povezivanjem prekidačkih mreža a upravljačka jedinica mikroprogramiranjem.

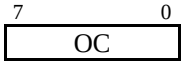
REŠENJE

Formati instrukcija

U procesoru se koriste sledeći formati instrukcija:

Format bezadresnih instrukcija – RTS, RTI, PUSH, POP, ASR

Format ovih instrukcija je dat na slici 23.

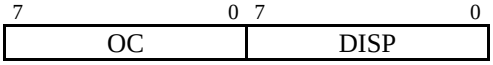


Slika 23 Format bezadresnih instrukcija

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

Format instrukcije relativnog skoka – BZ

Format ove instrukcije je dat na slici 24.



Slika 24 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.

Format instrukcija apslolutnog skoka – JMP, JSR

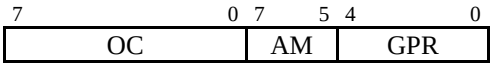
Format ovih instrukcija je dat na slici 25.

Slika 25 Format instrukcija apsolutnog 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.

Format jednoadresnih registarskih instrukcija – LD, ST, ADD, AND, JMPADR sa registarskim direktnim i registarskim indirektnim adresiranjima

Format ovih instrukcija je dat na slici 26.

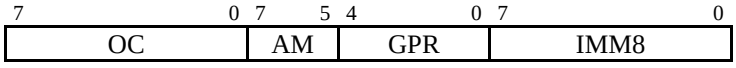


Slika 26 Format jednoadresnih registarskih instrukcija – R format

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

Format jednoadresnih neposrednih instrukcija – LD, ST, ADD, AND, JMPADR sa neposrednim adresiranjem

Format ovih instrukcija je dat na slici 27.

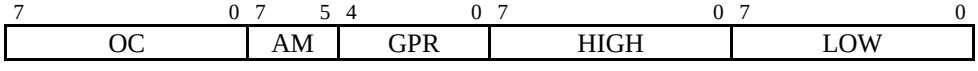


Slika 27 Format jednoadresnih instrukcija – IB format

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.

Format jednoadresnih memorijskih instrukcija – LD, ST, ADD, AND, JMPADR sa memorijskim direktnim, memorijskim indirektnim, registarskim indirektnim sa pomerajem, baznim indeksnim sa pomerajem i PC relativnim sa pomerajem adresiranjima

Format ovih instrukcija je dat na slici 28.

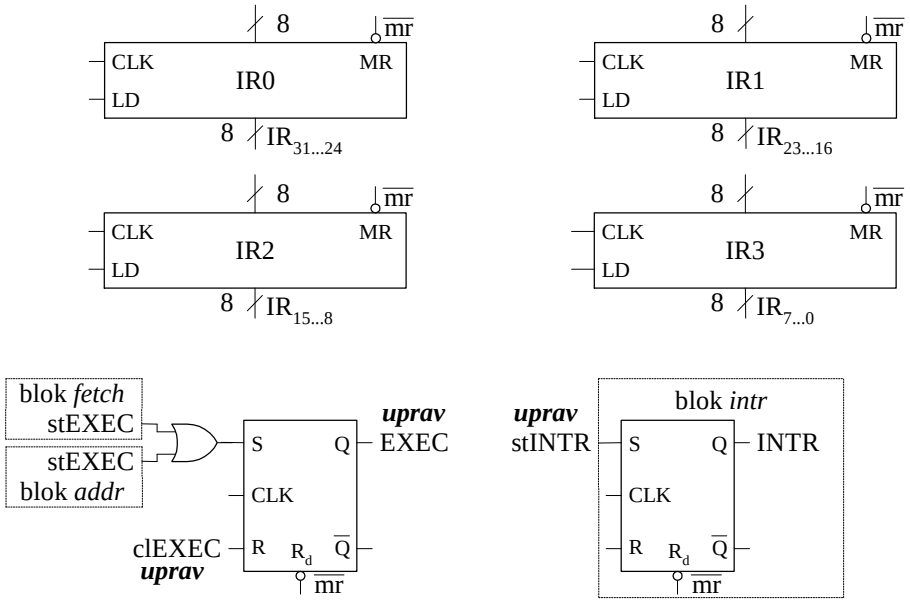


Slika 28 Format jednoadresnih instrukcija – AP format

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.

Operaciona jedinica bloka *exec*

Operaciona jedinica bloka *exec* sadrži prihvatni registar instrukcije $IR_{31...0}$, flip-flop FETCH (slika 29), adresni registar $MAR_{15...0}$, prihvatni registar podatka $MDR_{7...0}$, programski brojač $PC_{15...0}$, ukazivač na vrh steka $SP_{15...0}$, sabirač ADD, registar $DW_{15...0}$ (slika 30), aritmetičko logičku jedinicu ALU, registre $AB_{7...0}$ i $BB_{7...0}$, registarski fajl GPR (slika 31), programsku statusnu reč $PSW_{15...0}$ sa kombinacionim mrežama za postavljanje indikatora (slika 32) i kombinacione mreže za generisanje signala logičkih uslova operacija, adresiranja i rezultata operacija.



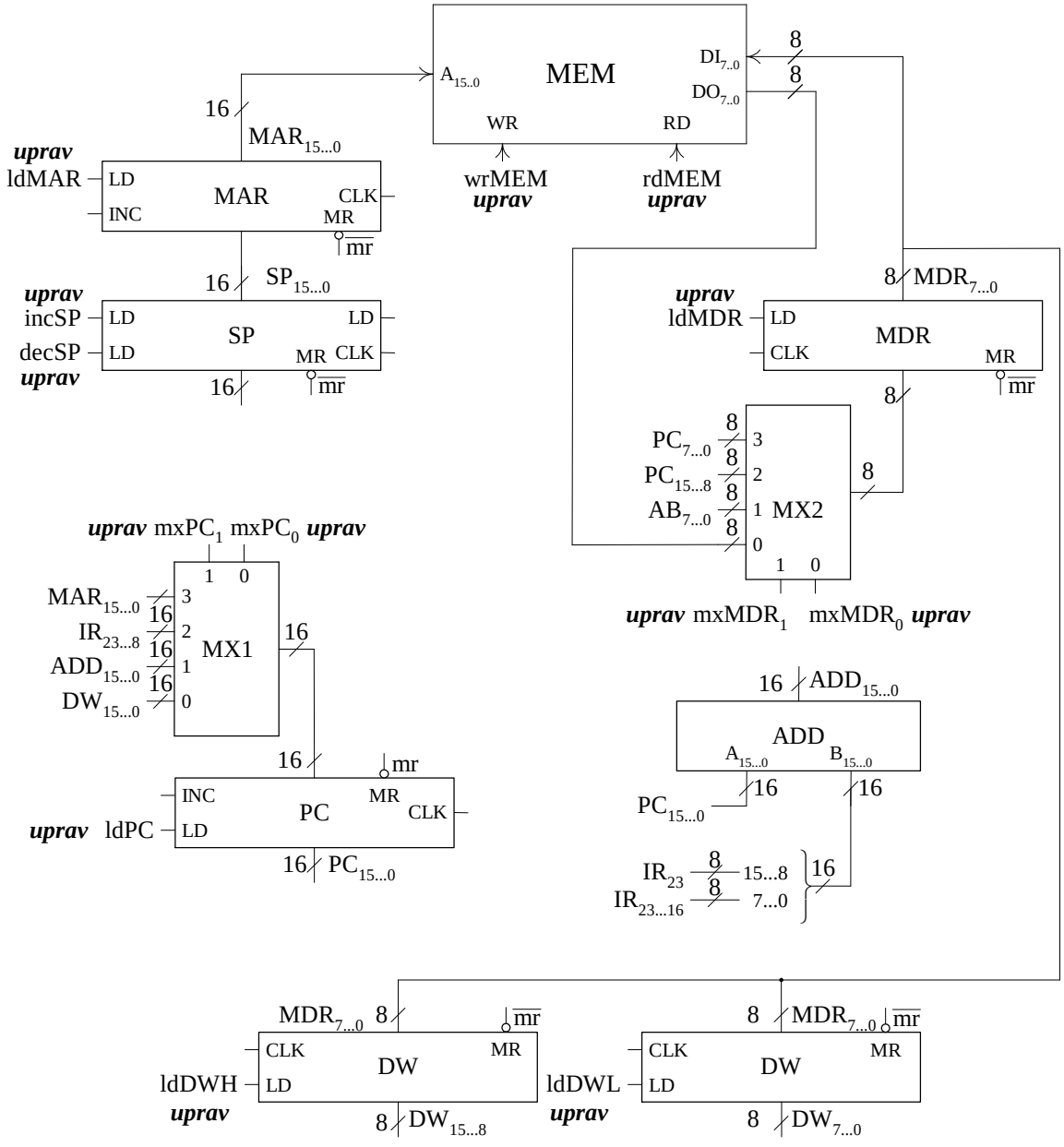
Slika 29 Operaciona jedinica (prvi deo)

Registri IR0, IR1, IR2 i IR3 su 8-mo razredni registri koji formiraju razrede 31…24, 23…16, 15…8 i 7…0 prihvatnog registra instrukcije $IR_{31...0}$ (slika 29). Adresne instrukcije mogu, u zavisnosti od načina adresiranja, da budu dužine 2, 3 ili 4 bajta (slike 26, 27 i 28). U bloku *fetch* se saglasno formatu instrukcije prvi, drugi, treći i četvrti bajt instrukcije smeštaju redom u registre IR0, IR1, IR2 i IR3. Prvi bajt instrukcije se uvek čita i upisuje u razrede $IR_{31...24}$ čiji sadržaj predstavlja kod operacije. Broj preostalih bajtova instrukcije koji se čitaju i upisuje u razrede $IR_{23...16}$, $IR_{15...8}$ i $IR_{7...0}$ zavisi od vrednosti koda operacije, a u slučaju adresnih instrukcija, i od načina adresiranja i njihov sadržaj ima različito značenje. Prepostavljeno je da je instrukcija očitana u bloku *fetch* i da se sadržaj registra $IR_{31...0}$ može koristiti u bloku *exec* za realizaciju faze izvršavanje operacije.

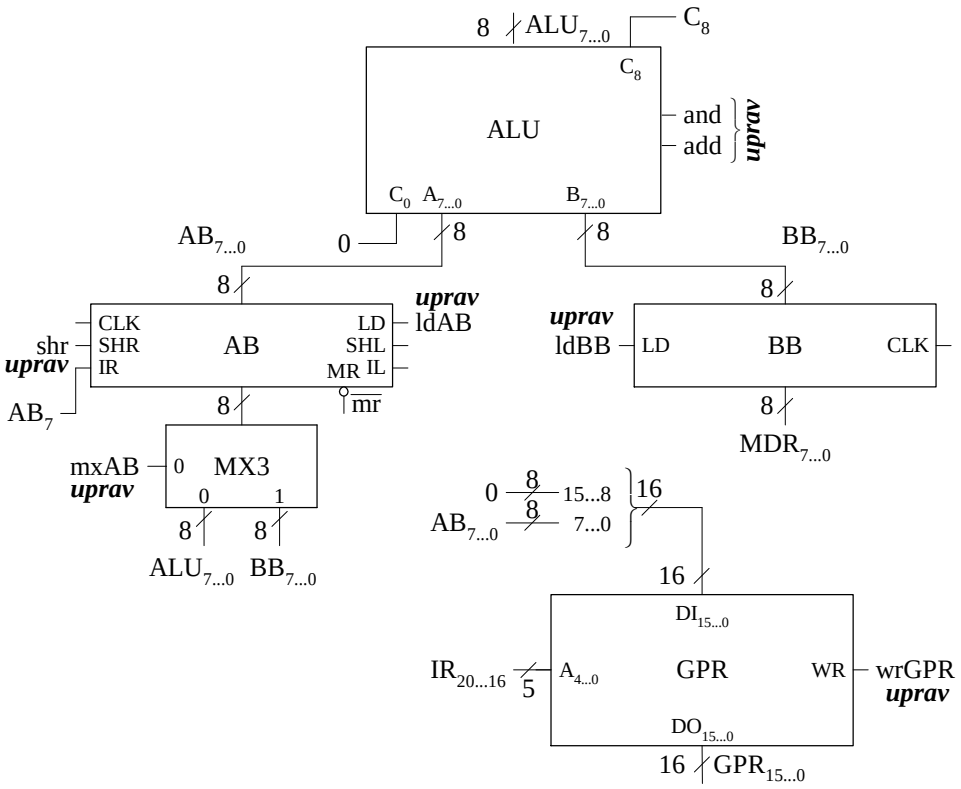
Flip-flop EXEC vrednostima 1 i 0 određuje da li je blok *exec* aktivan ili neaktivan, respektivno. U bloku *fetch* se na kraju faze čitanje bezadresnih instrukcija, instrukcija uslovnog skoka i instrukcija bezuslovnog skoka, vrednošću 1 signala **stEXEC** u flip-flop

EXEC upisuje vrednost 1, dok se u bloku *addr* na kraju faze formiranje adrese i čitanje operanda adresnih instrukcija vrednošću 1 signala **stEXEC** u flip-flop EXEC upisuje vrednost 1. Time se u bloku *exec* pokreće faza izvršavanje operacije. U bloku *exec* se na kraju faze izvršavanje operacije vrednošću 1 signala **clEXEC** upisuje vrednost 0 u flip-flop EXEC i time zaustavlja blok *exec*, dok se vrednošću 1 signala **stINTR** u flip-flop INTR u bloku *intr* upisuje vrednost 1 i time u bloku *intr* pokreće faza opsluživanje prekida.

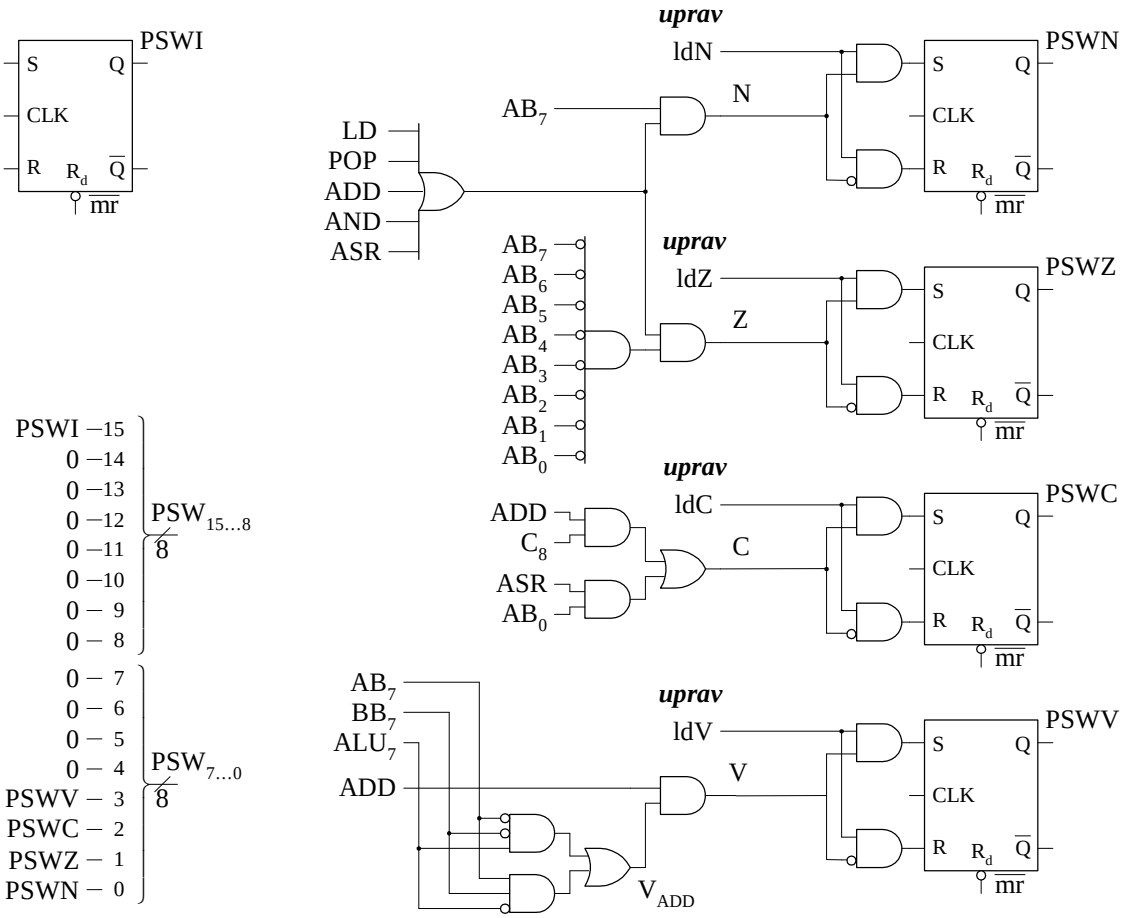
Registar $MAR_{15...0}$ je 16-to razredni adresni registar čiji se sadržaj koristi kao adresa memorijske lokacije memorije ***MEM*** sa koje se čita jedan bajt (slika 30). Prilikom prolaska instrukcija ST i JAMP kroz blok *addr* u registar $MAR_{15...0}$ je upisana sračunata adresa memorijske lokacije neophodna za njihovo izvršavanje u bloku *exec*. Sadržaj ukazivača na vrh steka $SP_{15...0}$ se vodi na ulaze registra $MAR_{15...0}$ i u njega upisuje vrednošću 1 signala **ldMAR** prilikom izvršavanja instrukcija kojima se na stek stavlja ili sa steka skida neki sadržaj. Sadržaj registra $MAR_{15...0}$ se vodi na adresne linije $A_{15...0}$ memorije ***MEM***.



Slika 30 Operaciona jedinica (drugi deo)



Slika 31 Operaciona jedinica (treći deo)



Slika 32 Operaciona jedinica (četvrti deo)

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. Sadržaj registra $SP_{15...0}$ se inkrementira i dekrementira vrednostima 1 signala **incSP** i **decSP**, pri upisu na stek i čitanju sa steka, respektivno. Sadržaj registra $SP_{15...0}$ se vodi na ulaze adresnog registra memorije $MAR_{15...0}$.

Registar $MDR_{7...0}$ je 8-mo razredni registar podatka memorije u koji se vrednošću 1 signala **ldMDR** upisuje sadržaj sa izlaza multipleksera MX2. Pri realizaciji ciklusa čitanja generiše se vrednosti 1 signala **rdMDR** i binarna vrednost 00 signala $\mathbf{mxMDR_1}$ **i** $\mathbf{mxMDR_0}$, čime se sadržaj sa izlaznih linija podataka $DO_{7...0}$ memorije ***MEM*** 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 01 do 11 signala $\mathbf{mxMDR_1}$ **i** $\mathbf{mxMDR_0}$ čime se jedan od sadržaja $AB_{7...0}$, $PC_{15...8}$ i $PC_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Sadržaj registra $MDR_{7...0}$ se vodi na ulazne linije podataka $DI_{7...0}$ memorije ***MEM***.

Multiplekser MX2 je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 3 multipleksera se vode sadržaji $DO_{7...0}$, $AB_{7...0}$, $PC_{15...8}$, $PC_{7...0}$, a selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 00 do 11, respektivno, upravljačkih signala $\mathbf{mxMBR_1}$ i $\mathbf{mxMBR_0}$. Sadržaj $DO_{7...0}$ se propušta kod ciklusa čitanja i predstavlja pročitanu vrednost memorijske lokacije koja po izlaznim linijama podataka $DO_{7...0}$ memorije ***MEM*** dolazi u procesor ***CPU***. Sadržaji $AB_{7...0}$, $PC_{15...8}$ i $PC_{7...0}$ 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 i 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. Selektovani sadržaj sa izlaza multipleksera MX2 se vodi na ulaze registra $MDR_{7...0}$.

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 preko multipleksera MX1 vodi na ulaze registra $PC_{15...0}$ i u njega upisuje vrednošću 1 signala **ldPC**. Sadržaj registra $PC_{15...0}$ se u slučaju instrukcije uslovnog skoka BZ koristi u sabiraču ADD za formiranje adrese skoka.

Multiplekser MX1 je 16-to razredni multiplekser sa 4 ulaza. Na ulaze 0 do 3 multipleksera se vode sadržaji $DW_{15...0}$, $ADD_{15...0}$, $IR_{23...8}$ i $MAR_{15...0}$,, koji predstavljaju adrese skoka za upis u registar $PC_{15...0}$. Selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 00 do 11, respektivno, upravljačkih signala $\mathbf{mxPC_1}$ i $\mathbf{mxPC_0}$. Sadržaj $DW_{15...0}$ predstavlja vrednost koja se restaurira sa steka prilikom izvršavanja instrukcije povratka iz potprograma RTS. Sadržaj $ADD_{15...0}$ predstavlja adresu skoka koja se dobija sabiranjem sadržaja registra $PC_{15...0}$ i pomeraja prilikom izvršavanja instrukcija uslovnog skoka BZ, $IR_{23...8}$ predstavlja adresu skoka koja se uzima iz instrukcije prilikom izvršavanja instrukcija bezuslovnog skoka JMP ili skoka na potprogram JSR i $MAR_{15...0}$ predstavlja adresu skoka prilikom izvršavanja instrukcije bezuslovnog skoka na sračunatu adresu (JADR).

Sabirač ADD je 16-to razredni sabirač koji se koristi za formiranje16-to bitne adrese relativnog skoka prilikom izvršavanja instrukcije uslovnog skoka BZ. Sadržaji $PC_{15...0}$ i $IR_{23...16}$ proširen znakom do dužine 16 bita se vode na ulaze A i B sabirača ADD. 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. Sadržaj $ADD_{15...0}$ sa izlaza sabirača se propušta kroz multiplekser MX1 i upisuje u programski brojač $PC_{15...0}$.

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 izlaznih linija podataka $DO_{7...0}$ memorije, 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 izlaznih linija podataka $DO_{7...0}$ memorije. Registar $DW_{15...0}$ se koristi da se prilikom izvršavanja instrukcije RTS prihvati 16-to bitna vrednost sa steka. Sadržaj $DW_{15...0}$ se propušta kroz multiplekser MX1 i upisuje u programski brojač $PC_{15...0}$.

Aritmetičko logička jedinica ALU realizacije jednu aritmetičku i jednu logičku mikrooperacije nad sadržajima registara $AB_{7...0}$ i $BB_{7...0}$ (slike 31). Mikrooperacija koju treba realizovati se specificira vrednošću 1 upravljačkog signala **add** za aritmetičku mikrooperaciju sabiranja i upravljačkog signala **and** za logičku mikrooperaciju I. 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 aritmetičke mikrooperacije, dok ta vrednost nije bitna u slučaju logičke mikrooperacije. U slučaju aritmetičke mikrooperacije sabiranja vrednost 1 na izlazu C_8 predstavlja prenos. U slučaju logičke mikrooperacije I 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 MX3 se vodi na ulaze registra $AB_{7...0}$ i u njega upisuje vrednošću 1 signala **ldAB**. Sadržaj registra $AB_{7...0}$ se pomera udesno za jedno mesto vrednošću 1 signala **shr**. Tada se u najstariiji razred registra AB_7 po liniji IR upisuje signal $\mathbf{AB_7}$. Sadržaj registra $AB_{7...0}$ se vodi na ulaze ALU gde se koristi kao prvi izvorišni operand u slučaju aritmetičkih i logičkih operacija.

Multiplekser MX3 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 operanda u koji se privremeno smešta izvorišni operand specificiran adresnim delom instrukcija LD, ADD i AND. Prilikom prolaska instrukcija LD, ADD i AND kroz blok *addr* u prihvatni registar podatka $BB_{7...0}$ je upisan očitani operand specificiran adresnim delom ovih instrukcija. Sadržaj registra $BB_{7...0}$ se vodi na ulaze ALU, gde se koristi kao drugi izvorišni operand u slučaju aritmetičkih i logičkih instrukcija ADD i AND, i na ulaze multipleksera MX3 kroz koji se propušta i upisuje u registar $AB_{7...0}$ u slučaju instrukcije LDB. Sadržaj prihvatnog registra podatka memorije $MDR_{7...0}$ se vodi na ulaze registra $BB_{7...0}$ i u njega upisuje vrednošću 1 signala **ldBB** prilikom izvršavanja instrukcije POP.

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 $IR_{20...16}$ koji se vodi na adresne linije $A_{4...0}$ registarskog fajla GPR. Sadržaj koji se upisuje vodi se 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. Na ulazne linije podataka $DI_{15...0}$ registarskog fajla se vodi sadržaj registra $AB_{7...0}$ proširen nulama do dužine 16 bita. Ovo se korisi u instrukciji prenosa STB sadržaja akumulatora $AB_{7...0}$, kada je registarskim direktnim adresiranjem kao odredište specificiran neki od registara opšte namene..

Registar $PSW_{15...0}$ je 16-to razredni registar koji sadrži indikatore programske statusne reči procesora (slika 32). 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 PSWV sadrži indikator *overflow*. U flip-flop PSWV se vrednošću 1 signala **ldV** u okviru faze izvršavanja određenih instrukcija 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 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 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 upisuje vrednost signala **N**.

Kombinacione mreže signala postavljanja indikatora N, Z, C i V se sastoje od logičkih elemenata. 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. Signal **N** ima vrednost koja odgovara vrednosti razreda AB_7 ukoliko je vrednost 1 jednog od signala operacija LD, POP, ADD, AND i ASR, 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 vrednost 1 ima jedan od signala operacija LD, POP, ADD, AND i ASR, 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 signala operacije ADD i vrednost koja odgovara vrednosti signala $\mathbf{AB_0}$ ukoliko je vrednost 1 signala operacija ASR, dok u svim ostalim situacija vrednost signala **C** je 0. Signal **V** ima vrednost koja odgovara vrednosti jednog od signala $\mathbf{V_{ADD}}$, ukoliko je vrednost 1 signala operacija ADD, 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.

Signali logičkih uslova operacija, adresiranja i rezultata operacija generišu se prema izrazima datim u tabelama 20, 21 i 22, respektivno.

Tabela 20 Signali operacija

$$\mathbf{PUSH} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \overline{\mathbf{IR}_{27}} \cdot \overline{\mathbf{IR}_{26}} \cdot \overline{\mathbf{IR}_{25}} \cdot \overline{\mathbf{IR}_{24}}$$

$$\mathbf{POP} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \overline{\mathbf{IR}_{27}} \cdot \overline{\mathbf{IR}_{26}} \cdot \overline{\mathbf{IR}_{25}} \cdot \mathbf{IR}_{24}$$

$$\mathbf{RTS} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \mathbf{IR}_{27} \cdot \overline{\mathbf{IR}_{26}} \cdot \mathbf{IR}_{25} \cdot \overline{\mathbf{IR}_{24}}$$

$$\mathbf{RTI} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \overline{\mathbf{IR}_{27}} \cdot \overline{\mathbf{IR}_{26}} \cdot \mathbf{IR}_{25} \cdot \mathbf{IR}_{24}$$

$$\mathbf{ASR} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \overline{\mathbf{IR}_{27}} \cdot \mathbf{IR}_{26} \cdot \overline{\mathbf{IR}_{25}} \cdot \overline{\mathbf{IR}_{24}}$$

$$\mathbf{BZ} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \overline{\mathbf{IR}_{27}} \cdot \mathbf{IR}_{26} \cdot \overline{\mathbf{IR}_{25}} \cdot \mathbf{IR}_{24}$$

$$\mathbf{JMP} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \overline{\mathbf{IR}_{27}} \cdot \mathbf{IR}_{26} \cdot \mathbf{IR}_{25} \cdot \overline{\mathbf{IR}_{24}}$$

$$\mathbf{JSR} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \overline{\mathbf{IR}_{27}} \cdot \mathbf{IR}_{26} \cdot \mathbf{IR}_{25} \cdot \mathbf{IR}_{24}$$

$$\mathbf{LD} = \overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \mathbf{IR}_{27} \cdot \overline{\mathbf{IR}_{26}} \cdot \overline{\mathbf{IR}_{25}} \cdot \overline{\mathbf{IR}_{24}}$$

ST = $\overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \mathbf{IR}_{27} \cdot \overline{\mathbf{IR}_{26}} \cdot \overline{\mathbf{IR}_{25}} \cdot \mathbf{IR}_{24}$

ADD = $\overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \mathbf{IR}_{27} \cdot \overline{\mathbf{IR}_{26}} \cdot \mathbf{IR}_{25} \cdot \overline{\mathbf{IR}_{24}}$

AND = $\overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \mathbf{IR}_{27} \cdot \overline{\mathbf{IR}_{26}} \cdot \mathbf{IR}_{25} \cdot \mathbf{IR}_{24}$

JADR = $\overline{\mathbf{IR}_{31}} \cdot \overline{\mathbf{IR}_{30}} \cdot \overline{\mathbf{IR}_{29}} \cdot \overline{\mathbf{IR}_{28}} \cdot \mathbf{IR}_{27} \cdot \mathbf{IR}_{26} \cdot \overline{\mathbf{IR}_{25}} \cdot \overline{\mathbf{IR}_{24}}$

Tabela 21 Signali adresiranja

regdir = $\overline{\mathbf{IR}_{23}} \cdot \overline{\mathbf{IR}_{22}} \cdot \overline{\mathbf{IR}_{21}}$

Tabela 22 Signali rezultata operacija

eql = **PSWZ**

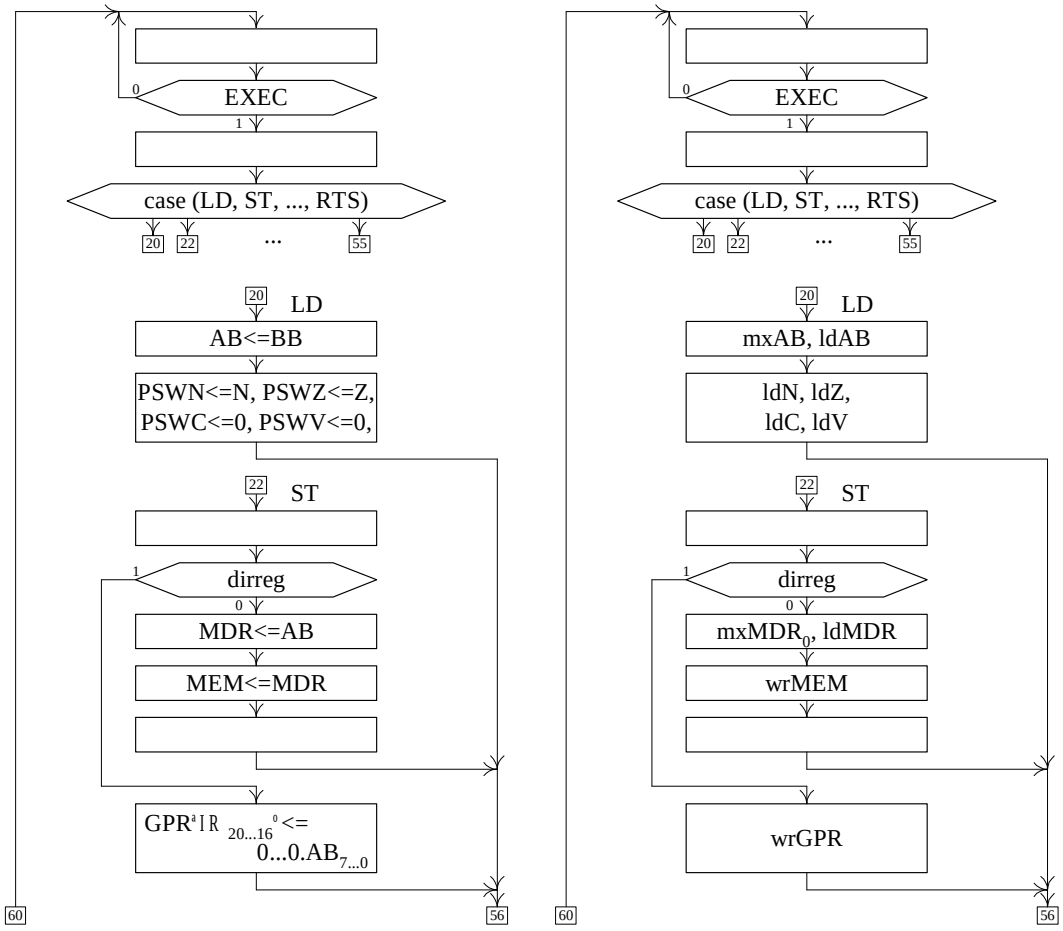
Signali operacija **PUSH**, **POP**, **RTS**, **RTI**, **ASR**, **BZ**, **JMP**, **JSR**, **LD**, **ST**, **ADD**, **AND** i **JADR** (tabela 20), od kojih u datom trenutku samo jedan ima vrednost 1, ukazuju koja operacija se realizuje.

Signal adresiranja **regdir** (tabela 21) vrednostima 1 i 0 ukazuju da li se u slučaju adresnih instrukcija radi o direktnom registarskom adresiranju ili o nekom od preostalih adresiranja, respektivno.

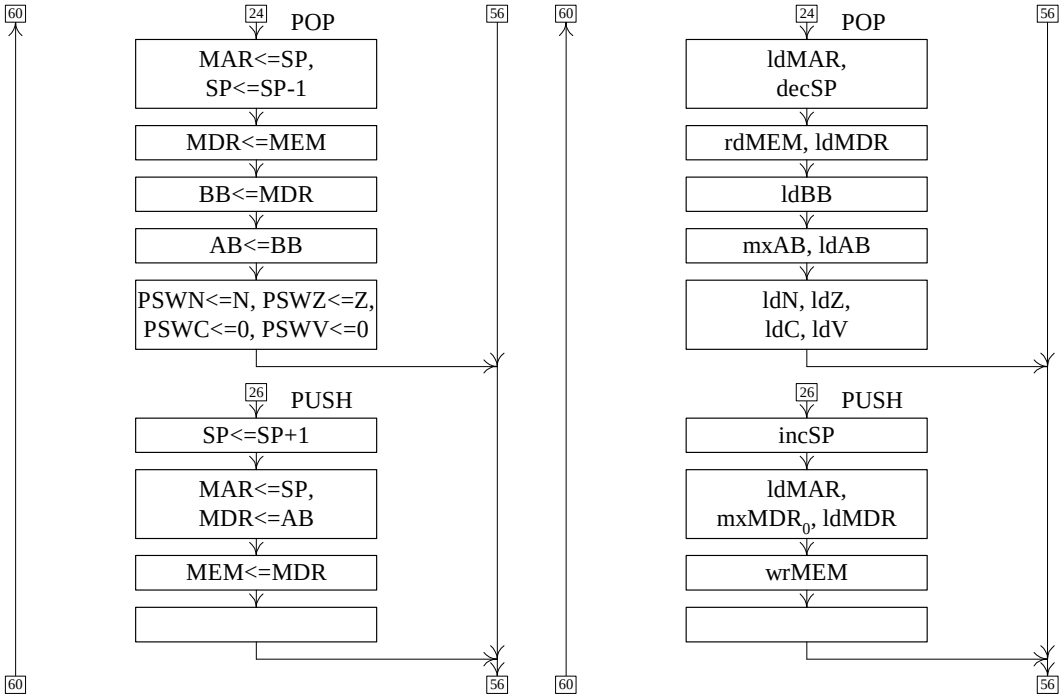
Signal rezultata operadije **eql** (tabela 22) vrednostima 1 i 0 ukazuju da li je rezultat zadnje izvršene instrukcije koja postavlja indikatore PSWN, PSWZ, PSWC i PSWV u programskoj statusnoj reči $PSW_{15...0}$ nula ili različit od nule, respektivno, i ima vrednost koja odgovara vrednosti signala PSWZ. Signal eql se koristi kod izvršavanja instrukcije uslovnog skoka BZ.

Dijagrami toka mikrooperacija i upravljačkih signala i sekvenca upravljačkih signala

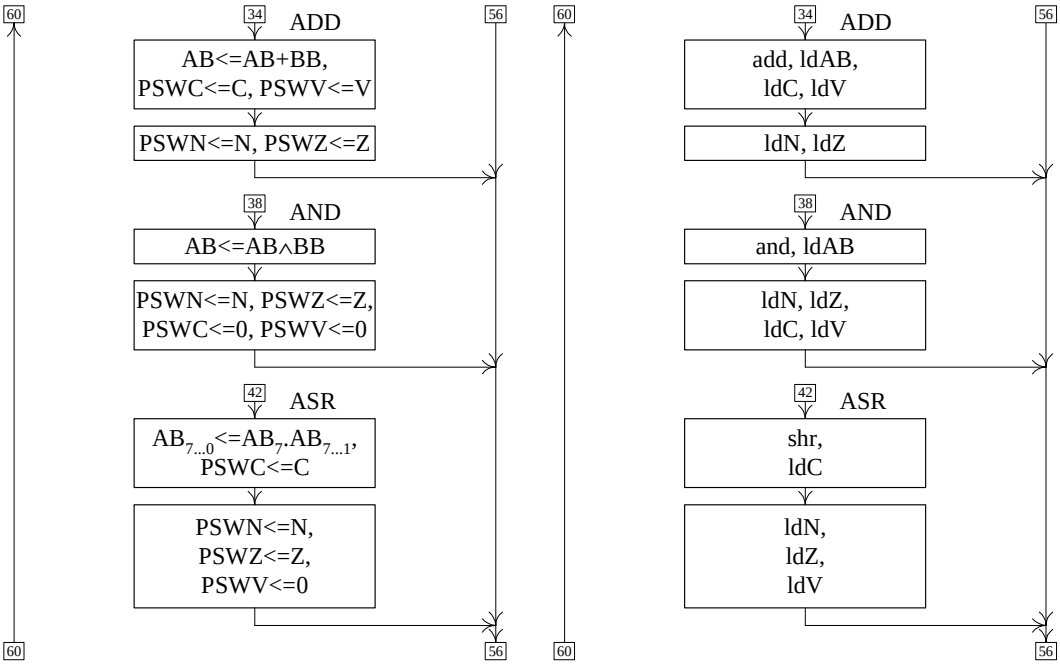
Dijagrami toka mikrooperacija i upravljačkih signala dati su na slikama 33 do 37, a sekvenca upravljačkih signala u tabeli 23.



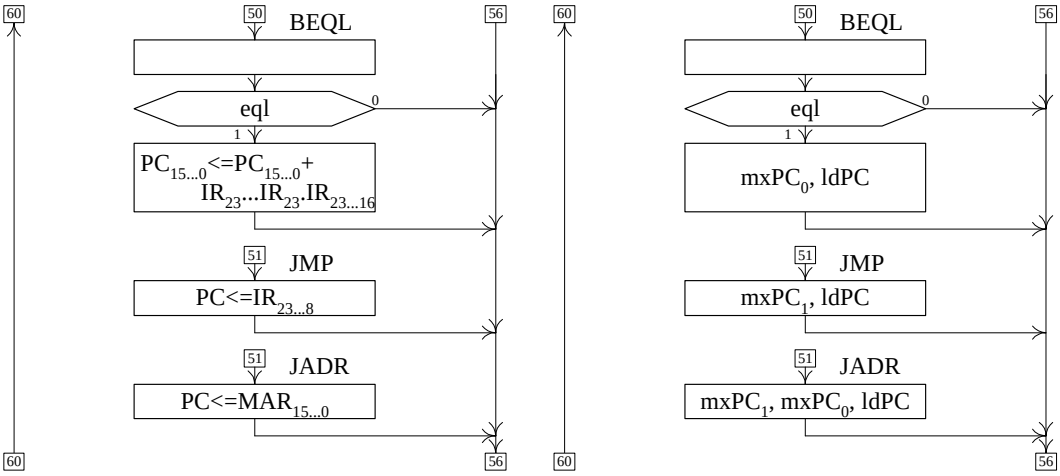
Slika 33 Dijagrami toka mikrooperacija i upravljačkih signala (prvi deo)



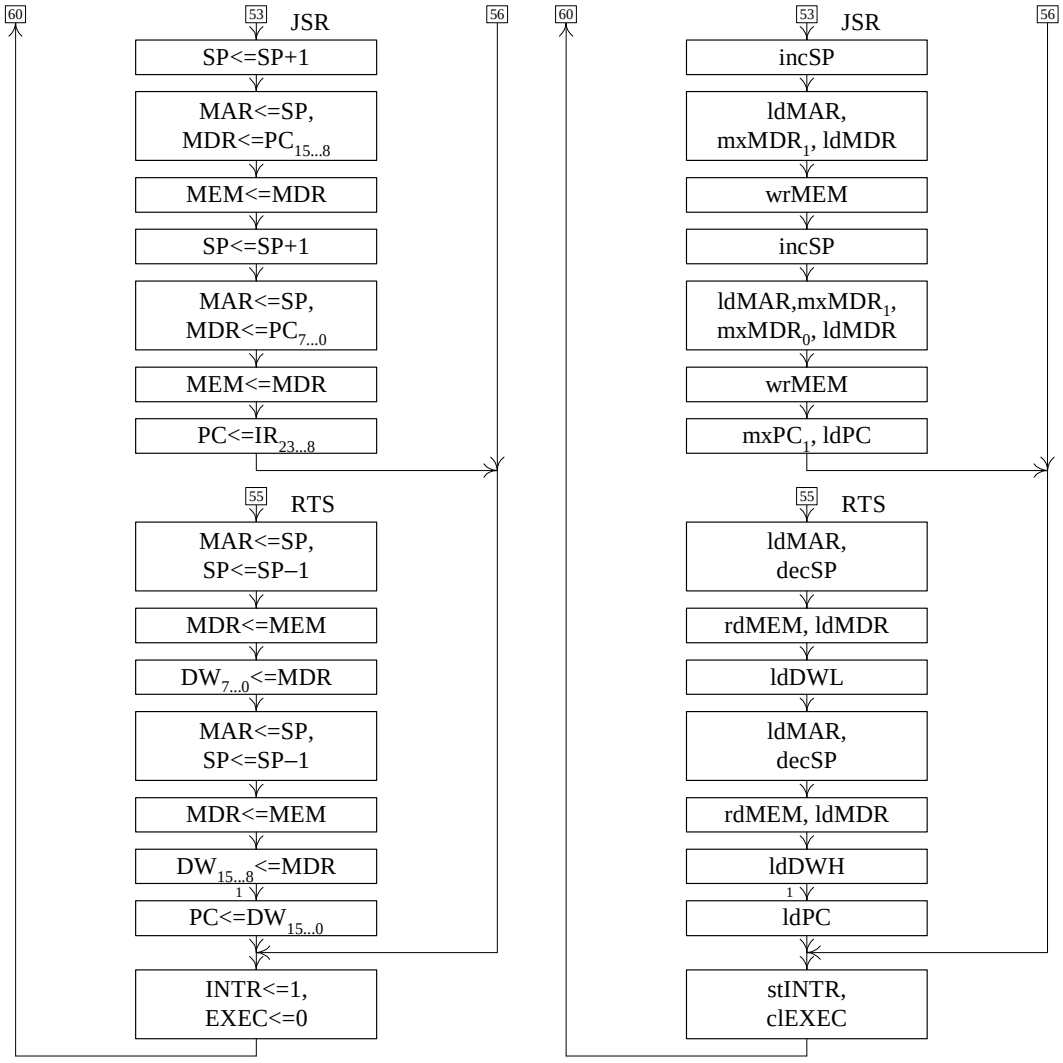
Slika 34 Dijagrami toka mikrooperacija i upravljačkih signala (drugi deo)



Slika 35 Dijagrami toka mikrooperacija i upravljačkih signala (treći deo)



Slika 36 Dijagrami toka mikrooperacija i upravljačkih signala (četvrti deo)



Slika 37 Dijagrami toka mikrooperacija i upravljačkih signala (peti deo)

Tabela 23 Sekvenca upravljačkih signala

! Izvršavanje operacije !

! Kroz korake bloka *exec* prolaze sve instrukcije, pri čemu bezadresne instrukcije i instrukcije skoka dolaze iz bloka *fetch*, a adresne iz bloka *addr*. U koracima bloka *addr* se saglasno specificiranom načinu adresiranja za sve instrukcije, sem instrukcija ST i JMPADR, čita operand i smešta u registar $BB_{7...0}$. Izuzetak su instrukcije ST i JADR. U slučaju instrukcije ST sa registarskim direktnim adresiranjem zaustavlja se blok *addr* i startuje blok za izvršavanje operacije *exec* u kome se sadržaj registra akumulatora $AB_{7...0}$ upisuje u registar opšte namene. U slučaju instrukcije ST nekim od memorijskih adresiranja u bloku *addr* se samo formira adresa operanda i smešta u registr $MAR_{15...0}$, pa se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec* u kome se sadržaj registra akumulatora $AB_{7...0}$ upisuje u memoriju na formiranoj adresi. Pretpostavljeno je da se u slučaju instrukcije ST neće javiti neposredno adresiranje. U slučaju instrukcije JADR nekim od memorijskih adresiranja, u ovoj fazi se samo formira adresa operanda i smešta u registr $MAR_{15...0}$, pa se zaustavlja blok *addr* i startuje blok za izvršavanje operacije *exec* u kome se sadržaj registra $MAR_{15...0}$ upisuje u programski brojač $PC_{15...0}$. Pretpostavljeno je da se slučaju instrukcije JADR neće javiti neposredno adresiranje ili direktno registarsko adresiranje. !

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

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

! U koraku $step_{01}$ se realizuje višestruki uslovni skok na jedan od koraka $step_{02}$, $step_{04}$, ..., $step_{82}$ u zavisnosti od toga koji od signala operacija **LD**, **ST**, ..., **RTS** ima aktivnu vrednost. !

$step_{01}$ *br* (*case* (**LD**, **ST**, **POP**, **PUSH**,
ADD, **AND**, **ASR**,
BEQL, **JMP**, **JADR**, **JSR**, **RTS**)
then
(**LD**, $step_{02}$), (**ST**, $step_{04}$), (**POP**, $step_{42}$), (**PUSH**, $step_{4F}$),
(**ADD**, $step_{5E}$), (**AND**, $step_{66}$), (**ASR**, $step_{6E}$),
(**BEQL**, $step_{72}$), (**JMP**, $step_{74}$), (**JADR**, $step_{7C}$), (**JSR**, $step_{75}$), (**RTS**, $step_{82}$));

! LD !

! U korak $step_{02}$ se dolazi iz $step_{01}$ ukoliko signal operacije **LD** ima vrednost 1. U korak $step_{02}$ se operand specificiran adresnim delom instrukcije, koji je prilikom prolaska instrukcije kroz blok *addr* doveden u registar $BB_{7...0}$, prebacuje u registar $AB_{7...0}$. Stoga se vrednostima 1 signala **mxAB** i **ldAB** sadržaj registra $BB_{7...0}$ propušta kroz multiplekser MX3 i upisuje u registar $AB_{7...0}$. Potom se u koraku $step_{03}$ vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** 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 $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{02}$ **mxAB**, **ldAB**;
$step_{03}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{2A}$;

! STB !

! U korak $step_{04}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije **ST** ima vrednost 1. Iz ovog koraka se u zavisnosti od toga da li signal **dirreg** ima vrednost 0 ili 1 prelazi na korak $step_{05}$ ili $step_{08}$.!

$step_{04}$ *br* (*if* **dirreg** *then* $step_{08}$);

! U koracima $step_{05}$ i $step_{06}$ se sadržaj registra $AB_{7...0}$ upisuje u memorijsku lokaciju na adresi koja je formirana prilikom prolaska instrukcije kroz blok *addr* i koja se nalazi i registru $MAR_{15...0}$. Stoga se, najpre, u koraku $step_{05}$ vrednostima 1 signalima $\textbf{mxMDR}_{\textbf{0}}$ i **ldMDR** sadržaj registra $AB_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. U koraku $step_{06}$ se vrednošću 1 signala **wrMEM** realizuje upis jednog bajta iz registra podatka $MDR_{7..0}$, koji je povezan na ulazne linije podataka $DI_{7...0}$ memorije ***MEM***, na adresi određenoj sadržajem adresnog registra $MAR_{15..0}$, koji je povezan na adresne linije $A_{15..0}$ memorije ***MEM***, i prelazi na korak $step_{07}$. Iz koraka $step_{07}$ se bezuslovno prelazi na korak $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{05}$ **mxMDR$_0$**, **ldMDR**;
$step_{06}$ **wrMEM**;
$step_{07}$ *br* $step_{2A}$;

! U koraku $step_{08}$ se sadržaj registra $AB_{7...0}$ proširen nulama do dužine 16 bita vrednošću 1 signala **wrGPR** upisuje u registar opšte namene $GPR_{15...0}$ adresiran razredima $IR_{20...16}$ prihvatnog registra instrukcije $IR_{31...0}$ i bezuslovno prelazi na korak $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{08}$ **wrGPR**,
br $step_{2A}$;

! POP !

! U korak $step_{09}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije **POP** ima vrednost 1. U fazi izvršavanja ove instrukcije sa steka se skida bajt podatka i upisuje u registar $AB_{7...0}$. Stoga se, najpre, u koraku $step_{09}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. U koraku $step_{0A}$ se realizuje čitanje jednog bajta i upisivanje u registar podatka $MDR_{7..0}$. Vrednošću 1 signala **rdMEM** se sa adrese određene sadržajem adresnog registra $MAR_{15..0}$, koji je povezan na adresne linije $A_{15..0}$ memorije ***MEM***, u memoriji ***MEM*** realizuje operacija čitanja, pročitani sadržaj, koji memorija ***MEM*** šalje po izlaznim linijama podataka $DO_{7...0}$, se vrednošću 1 signala **ldMDR** upisuje u registar podatka $MDR_{7..0}$ i prelazi na korak $step_{0B}$. U koraku $step_{0B}$ se vrednošću 1 signala **ldBB** sadržaj registra $MDR_{7...0}$ upisuje u registar $BB_{7...0}$. Zatim se u koraku $step_{0C}$ vrednostima 1 signala **mxAB** i **ldAB** sadržaj registra $BB_{7...0}$ propušta kroz multiplekser MX3 i upisuje u registar $AB_{7...0}$. Na kraju se u koraku $step_{0D}$ vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** upisuju u razrede PSWN i PSWZ programske statusne reči 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 $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{09}$ **ldMAR**, **decSP**;
$step_{0A}$ **rdMEM**, **ldMDR**;
$step_{0B}$ **ldBB**;
$step_{0C}$ **mxAB**, **ldAB**;
$step_{0D}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{2A}$;

! PUSH !

! U korak $step_{0E}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije **PUSH** ima vrednost 1. U fazi izvršavanja ove instrukcije se sadržaj registra $AB_{7...0}$ stavlja na vrh steka. Stoga se najpre u koraku $step_{0E}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{0F}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ 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 $step_{10}$ na isti način kao što se to radi u koraku $step_{06}$ instrukcije STB i prelazi na korak $step_{11}$. Na kraju se iz koraka $step_{11}$ bezuslovno prelazi na korak $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{0E}$ **incSP**;
$step_{0F}$ **ldMAR**, **mxMDR$_0$**, **ldMDR**;
$step_{10}$ **wrMEM**;
$step_{11}$ *br* $step_{2A}$;

! ADD !

! U korak $step_{12}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije **ADD** ima vrednost 1. U fazi izvršavanja ove instrukcije se sabiraju sadržaji registra $AB_{7...0}$, koji se koristi kao akumulator, i registra $BB_{7...0}$, u kome se nalazi operand specificiran adresnim delom instrukcije koji je prilikom prolaska instrukcije kroz blok *addr* doveden u registar $BB_{7...0}$, i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **add** 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 MX3 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$. Istovremeno se vrednostima

1 signala **ldC** i **ldV** u razrede PSWC i PSWV programske statusne reči $PSW_{15...0}$ upisuju vrednosti signala **C** i **V** formirane na osnovu dobijenog rezultata na izlazima $ALU_{7...0}$ i prelazi na korak $step_{13}$. U koraku $step_{13}$ se vrednostima 1 signala **ldN** i **ldZ** 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 prelazi na korak $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{12}$ **add**, **ldAB**, **ldC**, **ldV**;
$step_{13}$ **ldN**, **ldZ**,
br $step_{2A}$;

! AND !

! U korak $step_{14}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije **AND** ima vrednost 1. U fazi izvršavanja ove instrukcije se nad sadržajima registra $AB_{7...0}$, koji se koristi kao akumulator, i registra $BB_{7...0}$, u kome se nalazi operand specificiran adresnim delom instrukcije koji je prilikom prolaska instrukcije kroz blok *addr* doveden u registar $BB_{7...0}$, realizuje logička I operacija i rezultat upisuje u registar $AB_{7...0}$. Stoga se vrednošću 1 signala **and** 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 MX3 i vrednošću 1 signala **ldAB** upisuje u registar $AB_{7...0}$ i prelazi na korak $step_{15}$. U koraku $step_{15}$ se vrednostima 1 signala **ldN**, **ldZ**, **ldC** i **ldV** 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 $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{14}$ **and**, **ldAB**;
$step_{15}$ **ldN**, **ldZ**, **ldC**, **ldV**,
br $step_{2A}$;

! ASR !

! U korak $step_{16}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije ASR ima vrednost 1. U fazi izvršavanja ove instrukcije se bitovi registra $AB_{7...0}$ koji se koristi kao akumulator aritmetički pomeraju za jedno mesto udesno. Stoga se vrednošću 1 signala **shr** bitovi registra $AB_{7...0}$ pomeraju udesno za jedno mesto, pri čemu se u u razred AB_7 preko ulaza **IR** upisuje vrednost signala $\mathbf{AB_7}$. Istovremeno se vrednošću 1 signala **ldC** u razred PSWC programske statusne reči $PSW_{15...0}$ upisuje vrednost signala **C**, koja u slučaju svih pomeranja i rotiranja za jedno mesto udesno predstavlja vrednost signala $\mathbf{AB_0}$. Iz koraka $step_{16}$ se prelazi na korak $step_{17}$. U koraku $step_{17}$ se vrednostima 1 signala **ldN** i **ldZ** 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 vrednošću 1 signala **ldV** u razred PSWV programske statusne reči $PSW_{15...0}$ upisuje vrednost 0 signala **V** i bezuslovno prelazi na korak $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{16}$ **shr**, **ldC**;
$step_{17}$ **ldN**, **ldZ**, **ldV**,
br $step_{2A}$;

! BEQL !

! U korak $step_{18}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije uslovnog skoka **BEQL** ima vrednost 1 i prelazi na $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr* ukoliko signal **eql** ima vrednost 0 i na korak $step_{19}$ ukoliko ima vrednost 1. !

$step_{18}$ *br* (*if* $\overline{\mathbf{eql}}$ *then* $step_{2A}$);

! U korak $step_{19}$ se dolazi iz koraka $step_{18}$ ukoliko signal **eql** ima vrednost 1, čime je uslov za skok ispunjen. U ovom koraku se signali $ADD_{15...0}$ sa izlaza sabirača ADD, koji predstavljaju adresu skoka i koji se dobijaju kao suma sadržaja registra $PC_{15...0}$ i registra $IR_{23...16}$ proširenog znakom na 16 bita, vrednostima 1 signala $\mathbf{mxPC_0}$ i **ldPC** propuštaju kroz multiplekser MX1 i upisuju u registar $PC_{15...0}$. Pored toga iz koraka $step_{19}$ se bezuslovno prelazi na korak $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{19}$ $\mathbf{mxPC_0}$, **ldPC**,
br $step_{2A}$;

! JMP !

! U korak $step_{1A}$ se dolazi iz koraka $step_{01}$ 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 $IR_{23...8}$ koji predstavlja adresu skoka vrednostima 1 signala **$mxPC_1$** i **ldPC** propušta kroz multiplekser MX1 i upisuje u registar $PC_{15...0}$. Pored toga iz koraka $step_{1A}$ se bezuslovno prelazi na korak $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{1A}$ **$mxPC_1$**, **ldPC**,
br $step_{2A}$;

! JADR !

! U korak $step_{1B}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije **JADR** ima vrednost 1. U fazi izvršavanja ove instrukcije se realizuje bezuslovni skok na adresu koja je sračunata prilikom prolaska instrukcije kroz blok *addr* i koja se nalazi u registru $MAR_{15...0}$. Stoga se sadržaj registra $MAR_{15...0}$ vrednostima 1 signala **$mxPC_1$**, **$mxPC_0$** i **ldPC** propušta kroz multiplekser MX1 i upisuje u registar $PC_{15...0}$. Pored toga iz koraka $step_{1B}$ se bezuslovno prelazi na korak $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{1B}$ **$mxPC_1$**, **$mxPC_0$**, **ldPC**,
br $step_{2A}$;

! JSR !

! U korak $step_{1C}$ se dolazi iz koraka $step_{01}$ 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 $step_{1C}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$ bloka *addr*. Zatim se u koraku $step_{1D}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **$mxMDR_1$** 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 $step_{1E}$ na isti način kao što se to radi u koraku $step_{06}$ instrukcije ST. Potom se u koraku $step_{1F}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{20}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala **$mxMDR_1$**, **$mxMDR_0$**, **ldMDR** sadržaj nižeg bajta registra $PC_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{21}$ na isti način kao što se to radi u koraku $step_{1E}$ prilikom upisa višeg bajta. Na kraju se u koraku $step_{22}$ sadržaj registra $IR_{23...8}$ koji predstavlja adresu skoka vrednostima 1 signala **$mxPC_1$** i **ldPC** propušta kroz multiplekser MX1 i upisuje u registar $PC_{15...0}$ i bezuslovno prelazi na $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{1C}$ **incSP**;
$step_{1D}$ **ldMAR**, **$mxMDR_1$**, **ldMDR**;
$step_{1E}$ **wrMEM**;
$step_{1F}$ **incSP**;
$step_{20}$ **ldMAR**, **$mxMDR_1$**, **$mxMDR_0$**, **ldMDR**;
$step_{21}$ **wrMEM**;
$step_{22}$ **$mxPC_1$**, **ldPC**,
br $step_{2A}$;

! RTS !

! U korak $step_{23}$ se dolazi iz koraka $step_{01}$ ukoliko signal operacije **RTS** ima vrednost 1. U koracima $step_{23}$ do $step_{29}$ se vrednošću sa steka restaurira programski brojač $PC_{15...0}$. Sa steka se najpre skida niži a zatim i viši bajt registra $PC_{15...0}$. Stoga se u koraku $step_{23}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koraku $step_{24}$ na isti načina kao što se to radi u koraku $step_{0A}$ instrukcije POP. Na kraju se u koraku $step_{25}$ vrednošću 1 signala **ldDWL** sadržaj registra $MDR_{7...0}$ upisuje u niži bajt registra $DW_{7...0}$. Potom se u koraku $step_{26}$ vrednošću 1 signala **ldMAR** sadržaj registra $SP_{15...0}$ upisuje u registar $MAR_{15...0}$. Pored toga se i vrednošću 1 signala **decSP** dekrementira sadržaj registra $SP_{15...0}$. Čitanje se realizuje u koraku $step_{27}$ na isti načina kao što se to radi u koraku $step_{24}$ u kome se čita niži bajt. Na kraju se u koraku $step_{28}$ vrednošću 1 signala **ldDWH** sadržaj registra $MDR_{7...0}$ upisuje u viši bajt registra $DW_{15...8}$. Time je 16-to bitna vrednost skinuta sa

steka i smeštena u registar $DW_{15...0}$. Konačno se u koraku $step_{29}$ vrednostima 0 signala $\mathbf{mxPC_1}$ i $\mathbf{mxPC_0}$ sadržaj registra $DW_{15...0}$ propušta kroz multiplekser MX1 i vrednošću 1 signala **ldPC** upisuje u registar $PC_{15...0}$ i bezuslovno prelazi na $step_{2A}$ u kome se zaustavlja blok *exec* i startuje blok za opsluživanje prekida *intr*. !

$step_{23}$ **ldMAR**, **decSP**;
$step_{24}$ **rdMEM**, **ldMDR**;
$step_{25}$ **ldDWL**;
$step_{26}$ **ldMAR**, **decSP**;
$step_{27}$ **rdMEM**, **ldMDR**;
$step_{28}$ **ldDWH**;
$step_{29}$ **ldPC**;

! U korak $step_{2A}$ se dolazi iz koraka $step_{03}$, $step_{07}$, $step_{08}$, $step_{0D}$, $step_{11}$, $step_{13}$, $step_{15}$, $step_{17}$, $step_{18}$, $step_{19}$, $step_{1A}$, $step_{1B}$, $step_{22}$ i $step_{29}$. U koraku $step_{2A}$ se vrednošću 1 signala **clEXEC** upisuje vrednost 0 u flip flop EXEC bloka *exec*, čime se zaustavlja blok *exec*, dok se vrednošću 1 signala **stINTR** upisuje vrednost 1 u flip flop INTR bloka *intr*, čime se startuje blok *intr*. !

$step_{2A}$ **clEXEC, stINTR**,
br $step_{00}$;

Upravljačka jedinica

Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu sekvence upravljačkih signala po koracima (tabela 23). Mikroprogram se formira tako što se svakom koraku u sekvenci upravljačkih signala po koracima pridruži binarna reč sa slike 38. 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
ldMAR	stINTR	rdMEM	wrMEM	ldMDR	-	$mxMDR_1$	$mxMDR_0$

8	9	10	11	12	13	14	15
incSP	decSP	ldDWH	ldDWL	-	ldPC	$mxPC_1$	$mxPC_0$

16	17	18	19	20	21	22	23
shr	clEXEC	ldAB	mxAB	and	add	ldBB	wrGPR

24	25	26	27	28	29	30	31
ldN	ldZ	ldC	ldV	*cc*			

32	33	34	35	36	37	38	39
ba							

Slika 38 Mikroinstrukcija

Mikroinstrukcija ima dva dela i to operacini deo i upravljački deo. Operacioni deo čine bitovi 0 do 27, a upravljački deo čine bitovi 28 do 40. 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.

Bezuslovni skokovi se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 23) 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 24.

Tabela 24 Signal bezuslovnog skoka

signal bezuslovnog skoka	*cc*
bruncnd	1

Uslovni skokovi se realizuju u onim koracima sekvence upravljačkih signala po koracima (tabela 23) 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 25.

Tabela 25 Signali uslovnih skokova

signal uslovnog skoka	polje *cc*	signal uslova
brnotADDR	2	$\overline{\textbf{ADDR}}$
brdirreg	3	**dirreg**
brnoteql	4	$\overline{\textbf{eql}}$

Višestruki uslovni skokovi se realizuju u onim koracima sekvence upravljačkih signala po koracima (tabela 23) 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 26.

Tabela 26 Signali višestrukih uslovnih skokova

signal višestrukog uslovnog skoka	polje *cc*	korak
bropr	5	$step_{01}$

Vrednosti 0, 6, 7, 8, 9, A, B, C, D, E i F polja *cc* koje nisu dodeljene signalu bezuslovnog skoka i signalima uslovnih skokova određuju 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 23). 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 40 bitova. Za kodiranje operacionog dela mikroinstrukcije koristi se 28 bitova. Upravljačkih signala operacione jedinice ima 26, ali je umesto 26 bitova usvojena dužina operacionog dela mikroinstrukcije 28 bitova. Za kodiranje upravljačkog dela mikroinstrukcije koristi se 12 bitova. Signala bezuslovnog skoka, signala uslovnih skokova i

signala višestrukih uslovnih skokova ima 5, ali je umesto 2 bita usvojena dužina polja *cc* upravljačkog dela mikroinstrukcije 4 bita. Ukupan broj koraka u sekvenci upravljačkih signala po koracima je 42, ali je umesto 6 bitova usvojena dužina polja *ba* upravljačkog dela mikroinstrukcije 8 bitova. Ovo je učinjeno da bi se dobio pregledniji mikroprogram predstavljen u heksadecimalnom obliku.

Mikroprogram se formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima (tabela 23) 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* $step_A$) ili *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 24. 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 27.

Tabela 27 Koraci $step_i$, $step_A$, vrednosti **$madr_A$** i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	**$madr_A$**	A
$step_{03}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{07}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{08}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{0D}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{11}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{13}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{15}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{17}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{19}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{1A}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{1B}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{22}$	$step_{2A}$	**$madr_{2A}$**	2A
$step_{2A}$	$step_{00}$	**$madr_{00}$**	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 25. 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 28.

Tabela 28 Koraci $step_i$, uslovi **uslov**, koraci $step_A$, vrednosti $\mathbf{madr_A}$ i vrednosti A za uslovne skokove

$step_i$	uslov	$step_A$	$\mathbf{madr_A}$	A
$step_{00}$	$\overline{\mathbf{ADDR}}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{04}$	**dirreg**	$step_{08}$	$\mathbf{madr_{08}}$	08
$step_{18}$	$\overline{\mathbf{eql}}$	$step_{11}$	$\mathbf{madr_{11}}$	11

Za iskaz *br* (*case* ($\mathbf{uslov_1}$, ..., $\mathbf{uslov_n}$) *then* ($\mathbf{uslov_1}$, $step_{A1}$), ..., ($\mathbf{uslov_n}$, $step_{An}$)) se upravljački deo mikroinstrukcije kodira tako što se za polje *cc* uzima kod dodeljen signalu višestrukog uslovnog skoka koji određuje signale $\mathbf{uslov_1}$, ..., $\mathbf{uslov_n}$ 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 $\mathbf{uslov_1}$, ..., $\mathbf{uslov_n}$ 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 26. 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 jedan iskaz ovog tipa koji se javlja u sekvenci upravljačkih signala po koracima dati su u tabeli 29.

Tabela 29 Signali uslova, koraci na koje se skače i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{01}$

uslov	$step_A$	A
LD	$step_{02}$	02
ST	$step_{04}$	04
POP	$step_{09}$	09
PUSH	$step_{0E}$	0E
ADD	$step_{12}$	12
AND	$step_{14}$	14
ASR	$step_{16}$	16
BEQL	$step_{18}$	18
JMP	$step_{1A}$	1A
JADR	$step_{1B}$	1B
JSR	$step_{22}$	22
RTS	$step_{23}$	23

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu mikroprogramske realizacije signal bezuslovnog skoka (tabela 24), signali uslovnih skokova (tabela 25), signali višestrukih uslovnih skokova (tabela 26) i signali vrednosti A za bezuslovne skokokve (tabela 27), uslovne skokove (tabela 28) i višestruke uslovne skokove (tabela 29).

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 23) formiran mikroprogram (tabela 30). 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 30 Mikroprogram za upravljačku jedinicu mikroprogramske realizacije

00	0000000 2 00	! **brnotADDR**, **madr$_{00}$** !
01	0000000 5 00	! **bropr** !
02	0000300 0 00	! **mxAB**, **ldAB** !
03	000000F 1 2A	! **ldN**, **ldZ**, **ldC**, **ldV**, **bruncnd**, **madr$_{2A}$** !
04	0000000 3 08	! **brdirreg**, **madr$_{08}$** !
05	0900000 0 00	! **mxMDR$_0$**, **ldMDR** !
06	1000000 0 00	! **wrMEM** !
07	0000000 1 2A	! **bruncnd**, **madr$_{2A}$** !
08	0000010 1 2A	! **wrGPR**, **bruncnd**, **madr$_{2A}$** !
09	8040000 0 00	! **ldMAR**, **decSP** !
0A	2800000 0 00	! **rdMEM**, **ldMDR** !
0B	0000020 0 00	! **ldBB** !
0C	0000300 0 00	! **mxAB**, **ldAB** !
0D	000000F 1 2A	! **ldN**, **ldZ**, **ldC**, **ldV**, **bruncnd**, **madr$_{2A}$** !
0E	0080000 0 00	! **incSP**!
0F	8900000 0 00	! **ldMAR**, **mxMDR$_0$**, **ldMDR**!
10	1000000 0 00	! **wrMEM** !
11	0000000 1 2A	! **bruncnd**, **madr$_{2A}$** !
12	0000243 0 00	! **add**, **ldAB**, **ldC**, **ldV** !
13	000000C 1 2A	! **ldN**, **ldZ**, **bruncnd**, **madr$_{2A}$** !
14	0000280 0 00	! **and**, **ldAB**!
15	000000F 1 2A	! **ldN**, **ldZ**, **ldC**, **ldV**, **bruncnd**, **madr$_{2A}$** !
16	0000802 0 00	! **shr**, **ldC** !
17	000000D 1 2A	! **ldN**, **ldZ**, **ldV**, **bruncnd**, **madr$_{2A}$** !
18	0000000 4 2A	! **brnoteql**, **madr$_{2A}$** !
19	0005000 1 2A	! **mxPC$_0$**, **ldPC**, **bruncnd**, **madr$_{2A}$** !
1A	0006000 1 2A	! **mxPC$_1$**, **ldPC**, **bruncnd**, **madr$_{2A}$** !
1B	0007000 1 2A	! **mxPC$_1$**, **mxPC$_0$**, **ldPC**, **bruncnd**, **madr$_{2A}$** !
1C	0080000 0 00	**! incSP !**
1D	8A00000 0 00	**! ldMAR**, **mxMDR$_1$**, **ldMDR !**
1E	1000000 0 00	**! wrMEM !**
1F	0080000 0 00	**! incSP !**
20	8B00000 0 00	**! ldMAR**, **mxMDR$_1$**, **mxMDR$_0$**, **ldMDR !**
21	1000000 0 00	**! wrMEM !**
22	0006000 1 2A	! **mxPC$_1$**, **ldPC**, **bruncnd**, **madr$_{2A}$** !
23	8040000 0 00	**! ldMAR**, **decSP !**
24	2800000 0 00	**! rdMEM**, **ldMDR !**
25	0010000 0 00	**! ldDWL !**

```
26  8040000 0 00 ! ldMAR, decSP !
27  2800000 0 00 ! rdMEM, ldMDR !
28  0020000 0 00 ! ldDWH !
29  0004000 0 00 ! ldPC !
2A  4000400 1 00 ! clEXEC, stINTR, bruncnd, madr00 !
```

Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 39. 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.

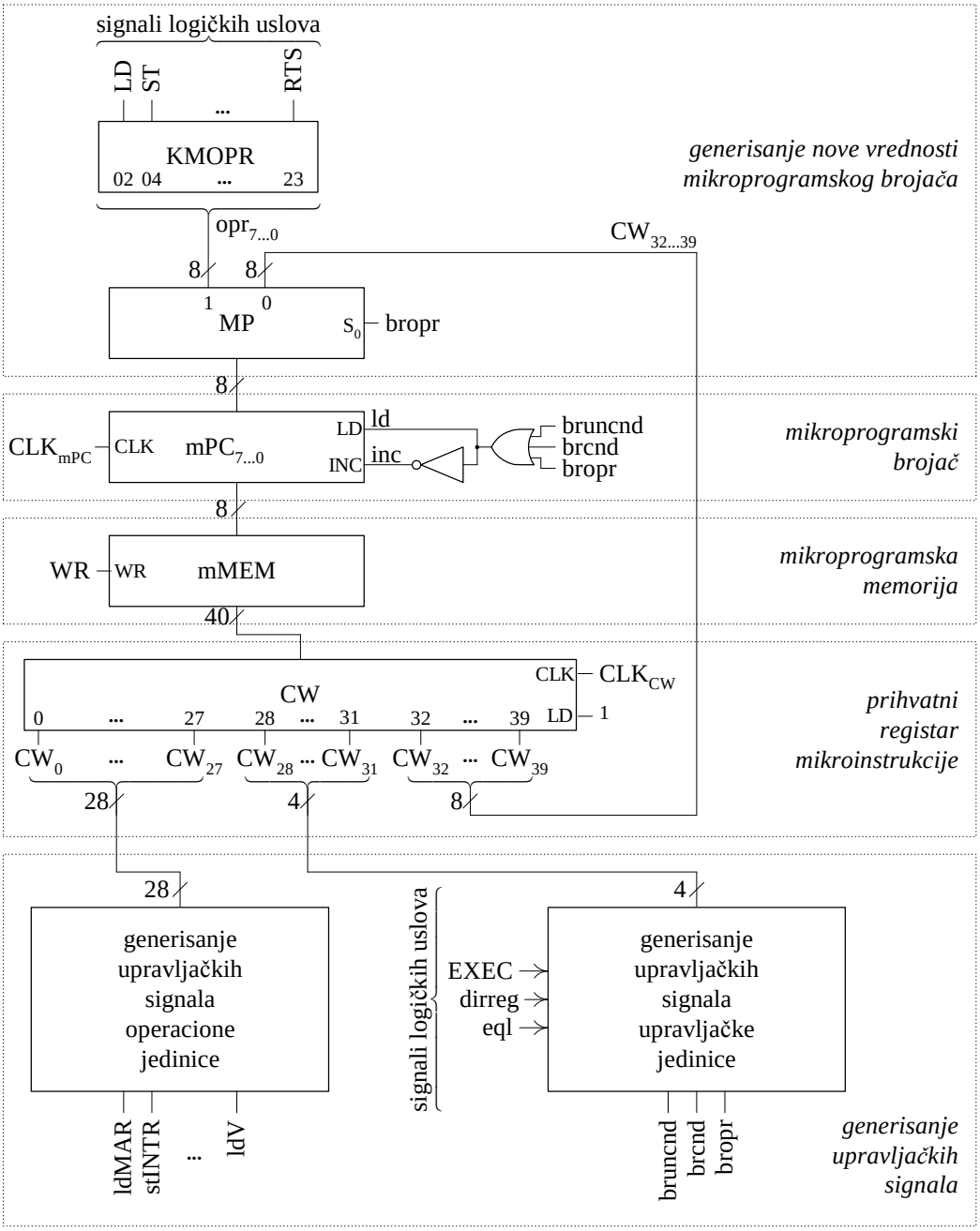
Blok *generisanje nove vrednosti mikroprogramskog brojača* se sastoji od kombinacione mreže KMOPR sa multiplekserom MP i služi za generisanje i selekciju vrednosti koju treba upisati u mikroprogramski brojač $mPC_{7...0}$. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikroprograma. Vrednosti koje treba upisati u mikroprogramski brojač generišu se na dva načina i to pomoću kombinacione mreže KMOPR koja formira signale $\mathbf{opr_{7...0}}$ i razreda $CW_{32...39}$ prihvatnog registra mikroinstrukcije $CW_{0...39}$. Selekcija jedne od dve grupe signala koje daju novu vrednost mikroprogramskog brojača obezbeđuje se signalom **bropr** i to signali $\mathbf{opr_{7...0}}$ ako signal **bropr** ima vrednost 1 i signali $CW_{32...39}$ ako signal **bropr** ima vrednost 0.

Kombinacionom mrežom KMOPR generišu se vrednosti (tabela 29) za realizaciju višestrukog uslovnog skoka na adresi 01 mikroprograma (tabela 30). U zavisnosti od toga koji od signala **LD**, **ST**,..., **RTS** ima vrednost 1 zavisi koja će od vrednosti iz tabele 29 da se pojavi tada na linijama $\mathbf{opr_{7...0}}$. S obzirom da se na adresi 01 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 kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača mPC.

Prihvatni registar mikroinstrukcije $CW_{0...39}$ u svojim razredima $CW_{32...39}$ sadrži vrednost za upis u mikroprogramski brojač $mPC_{7...0}$ za bezuslovne skokove (tabela 27) i uslovne skokove (tabela 28) u mikroprogramu (tabela 30). Signal višestrukog uslovnog skok **bropr** ima vrednost 1 samo prilikom izvršavanja mikroinstrukcije na adresi 01 mikroprograma, a u svim ostalim situacijama imaj vrednost 0. S obzirom da ovaj signal nema vrednost 1 prilikom izvršavanja mikroinstrukcija kojima se realizuju bezuslovni ili neki od uslovnih skokova u mikroprogramu, vrednost određena razredima $CW_{32...39}$ 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 30). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko svi signali **bropr**, **brcnd** i **bruncnd** imaju vrednost 0. Signali **bropr**, **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.



Slika 39 Struktura upravljačke jedinice mikroprogramske realizacije

U režimu skoka pri pojavi signala takta **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 30). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ukoliko jedan od signala **bropr**, **brcnd** i **bruncnd** ima vrednost 1. Signali **bropr**, **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 30). S obzirom da se mikroprogram svih faza izvršavanja instrukcija nalazi u opsegu od adrese 00 do adrese 2A, 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 40 bita, a kapacitet veličinom mikroprograma (tabela 30) i iznosi 256 lokacija. Adresiranje mikroprogramske memorije se realizuje sadržajem mikroprogramskog brojača $mPC_{7...0}$.

Blok *prihvatni registar mikroinstrukcije* sadrži prihvatni registar mikroinstrukcije $CW_{0...39}$. Prihvatni registar mikroinstrukcije $CW_{0...39}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...27}$ i $CW_{28...31}$ se koriste u bloku *generisanje upravljačkih signala* za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{32...39}$ 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 $CW_{0...27}$ prihvatnog registra mikroinstrukcije generišu upravljačke signale operacione jedinice i na osnovu sadržaja razreda $CW_{28...31}$ prihvatnog registra mikroinstrukcije i signala logičkih uslova **ADDR**, **dirreg** i **eql** koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice ***oper*** se generišu na sledeći način:

- $\mathbf{ldMAR = CW_0}$
- $\mathbf{stINTR = CW_1}$
- $\mathbf{rdMEM = CW_2}$
- $\mathbf{wrMEM = CW_3}$
- $\mathbf{ldMDR = CW_4}$
- $\mathbf{mxMDR_1 = CW_6}$
- $\mathbf{mxMDR_0 = CW_7}$
- $\mathbf{incSP = CW_8}$
- $\mathbf{decSP = CW_9}$
- $\mathbf{ldDWH = CW_{10}}$
- $\mathbf{ldDWL = CW_{11}}$
- $\mathbf{incPC = CW_{12}}$
- $\mathbf{ldPC = CW_{13}}$
- $\mathbf{mxPC_1 = CW_{14}}$
- $\mathbf{mxPC_0 = CW_{15}}$
- $\mathbf{shr = CW_{16}}$
- $\mathbf{clEXEC = CW_{17}}$
- $\mathbf{ldAB = CW_{18}}$
- $\mathbf{mxAB = CW_{19}}$
- $\mathbf{and = CW_{20}}$
- $\mathbf{add = CW_{21}}$
- $\mathbf{ldBB = CW_{22}}$

- **wrGPR** = $\mathbf{CW_{23}}$
- **ldN** = $\mathbf{CW_{24}}$
- **ldZ** = $\mathbf{CW_{25}}$
- **ldC** = $\mathbf{CW_{26}}$
- **ldV** = $\mathbf{CW_{27}}$

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

- **bruncnd** = $\overline{\mathbf{CW_{28}}} \cdot \overline{\mathbf{CW_{29}}} \cdot \overline{\mathbf{CW_{30}}} \cdot \mathbf{CW_{31}}$
- **brcnd** = **brnotEXEC**$\cdot \overline{\mathbf{EXEC}}$ + **brdirreg**$\cdot$**dirreg** + **brnoteql**$\cdot \overline{\mathbf{eql}}$
- **brnotEXEC** = $\overline{\mathbf{CW_{28}}} \cdot \overline{\mathbf{CW_{29}}} \cdot \mathbf{CW_{30}} \cdot \overline{\mathbf{CW_{31}}}$
- **brdirreg**= $\overline{\mathbf{CW_{28}}} \cdot \overline{\mathbf{CW_{29}}} \cdot \mathbf{CW_{30}} \cdot \mathbf{CW_{31}}$
- **brnoteql**= $\overline{\mathbf{CW_{28}}} \cdot \mathbf{CW_{29}} \cdot \overline{\mathbf{CW_{30}}} \cdot \overline{\mathbf{CW_{31}}}$
- **bropr**= $\overline{\mathbf{CW_{28}}} \cdot \mathbf{CW_{29}} \cdot \overline{\mathbf{CW_{30}}} \cdot \mathbf{CW_{31}}$

Pri generisanju signala **brcnd** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice ***oper*** i to **EXEC**, **dirreg** i **eql**.

4 ZADATAK 4

Posmatra se blok *intr* procesora u kome se realizuje faza opsluživanje prekida. Pored procesora u posmatranim računaru postoji i memorija. Memorija je kapaciteta 2^{16} bajtova. Širina memorijske reči je 1 bajt. Sadržaji dužine 16 bita smeštaju se u dve susedne memorijske lokacije i to tako da se na nižoj adresi smešta stariji bajt a na višoj adresi mlađi bajt.

Procesor je tako realizovan da se faza čitanje instrukcije realizuje u bloku *fetch*, faza formiranje adrese i čitanje operanda u bloku *addr*, faza izvršavanje operacije u bloku *exec* i faza opsluživanje prekida u bloku *intr*. Sve instrukcije po realizaciji faze izvršavanje operacije u bloku *exec* dolaze u blok *intr* radi realizacije faze opsluživanje prekida. Pretpostavlja se da se u bloku *intr* nalaze svi registri neophodni za izvršavanje faze opsluživanje prekida i da se njima nalaze informacije koje su rezultat prethodnih faza izvršavanja instrukcije. Od registara postoji programski brojač $PC_{15...0}$, programska statusna reč $PSW_{15...0}$, ukazivač na vrh steka $SP_{15...0}$, adresni registar memorije $MAR_{15...0}$, prihvatni registar podatka memorije $MDR_{7...0}$ i ukazivač na tabelu sa adresama prekidnih rutina $IVTP_{15...0}$.

Blok *intr* kreće sa fazom opsluživanje prekida ukoliko se u flip-flopu INTR nalazi vrednost 1. Po završetku faze opsluživanje prekida upisivanjem vrednosti 1 u flip-flop FETCH startuje se blok *fetch* i kreće sa fazom čitanja sledeće instrukcije, dok se upisivanjem vrednost 0 u flip-flop INTR zaustavlja blok *intr*.

Zahteve za prekid može da generiše osam kontrolera periferija po linijama $intr_7$ do $intr_0$. Ovi prekidi se nazivaju spoljašnji prekidi jer dolaze od uređaja van procesora, kao 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 PSWI registra programske statusne reči $PSW_{15...0}$ nalazi vrednost 1 ili 0, respektivno. Dozvoljavanje i maskiranje prekida se realizuje programskim putem izvršavanjem posebnih instrukcija kojima se u razred PSWI registra $PSW_{15...0}$ upisuju vrednosti 1 ili 0, respektivno.

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 PSWI registra $PSW_{15...0}$ nalazi vrednost 1. Ukoliko nije, završava se sa fazom opsluživanje prekida u bloku *intr* i prelazi na fazu čitanje instrukcije sledeće instrukcije u bloku *fetch*. Ukoliko jeste, izvršavanje tekuće instrukcije se produžava sa koracima faze opsluživanje prekida. Opsluživanje zahteva za prekid se sastoji iz dve grupe koraka.

U okviru prve grupe koraka na steku se čuvaju programski brojač $PC_{15...0}$ i programska statusna reči $PSW_{15...0}$. Stek raste prema višim adresama a ukazivač na vrh steka $SP_{15...0}$ ukazuje na zadnu zauzetu lokaciju.

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 $IVTP_{15...0}$ (*Interrupt Vector Table Pointer*), kreirana IV tabela sa 8 ulaza, tako da se u ulazima 7 do 0 nalaze adrese prekidnih rutina za svaki od prekida koji dolaze po linijama $intr_7$ do $intr_0$, respektivno. Prekidi koji dolaze po linijama $intr_7$ do $intr_0$ uređeni su po prioritetima 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 $IVTP_{15...0}$ i na kraju dobijena vrednost koristiti kao adresu sa koje se čita adresa prekidne rutine i upisuje u registar $PC_{15...0}$.

Realizovati blok *exec* procesora. Operacione jedinica treba da bude realizovana direktnim povezivanjem prekidačkih mreža a upravljačka jedinica mikroprogramiranjem.

REŠENJE

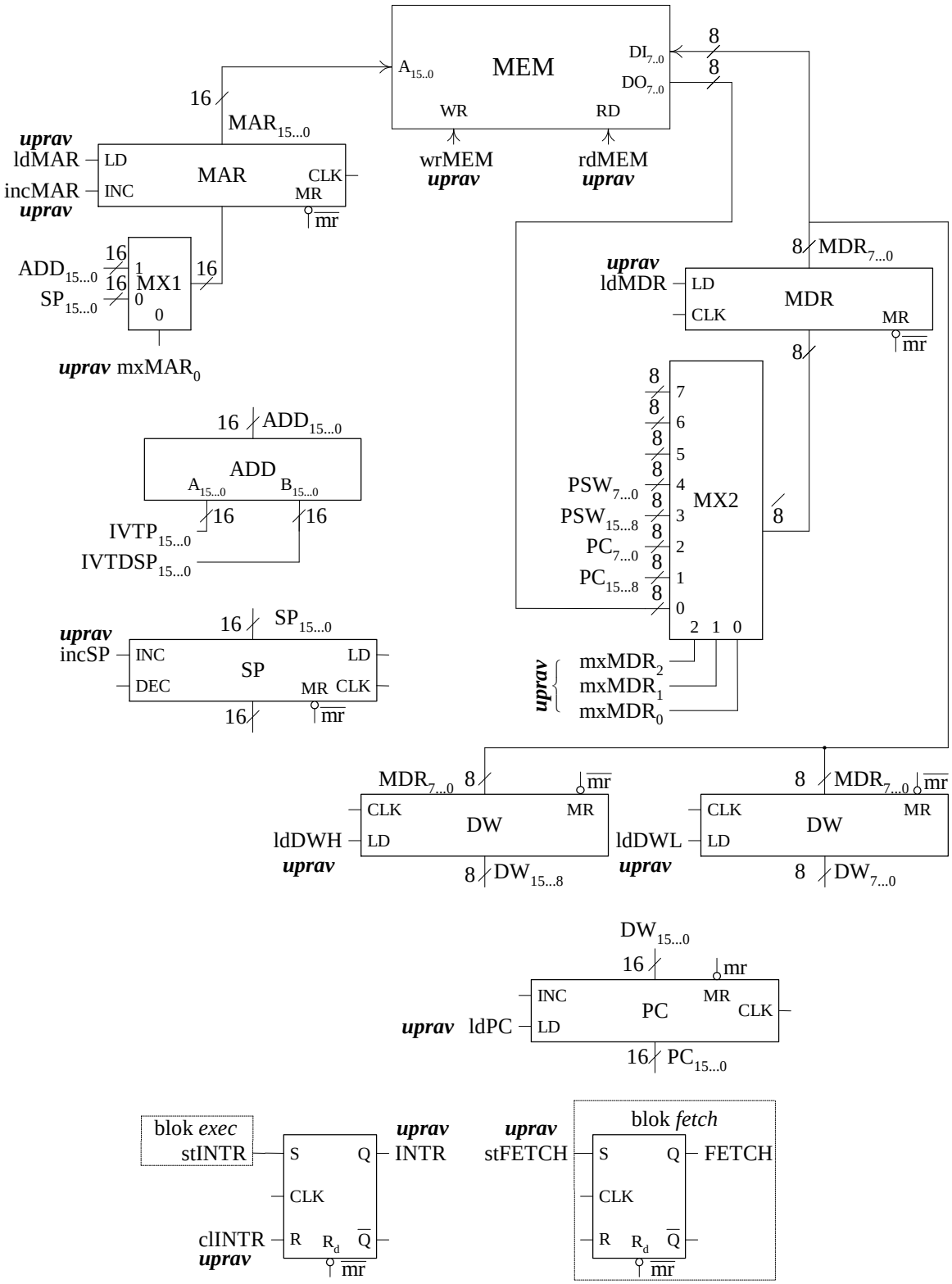
Operaciona jedinica bloka *intr*

Operaciona jedinica bloka *intr* sadrži adresni registar $MAR_{15...0}$ sa multiplekserom MX1, prihvatni registar podatka $MDR_{7...0}$, sa multiplekserom MX2, ukazivač na vrh steka $SP_{15...0}$, sabirač ADD, registar $DW_{15...0}$, programski brojač $PC_{15...0}$, flip-flop INTR (slika 40), registar $PSW_{15...0}$ (slika 41), logičke ILI i I elemene 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 42).

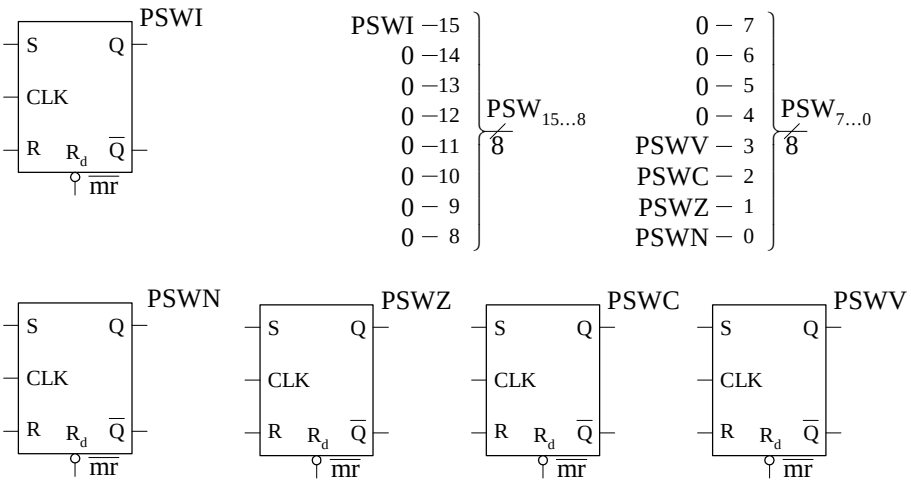
Registar $MAR_{15...0}$ je 16-to razredni adresni registar čiji se sadržaj koristi kao adresa memorijske lokacije memorije ***MEM*** u koju se upisuje ili sa koje se čita jedan bajt. Adresa se sa izlaza registra $SP_{15...0}$ i sabirača ADD preko multipleksera MX1 vodi na ulaze registra $MAR_{15...0}$ i u njega upisuje vrednošću 1 signala **ldMAR**. Sadržaj registra $MAR_{15...0}$ se inkrementira vrednošću 1 signala **incMAR**. Ovo se koristi u situacijama kada treba pročitati 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. Sadržaj registra $MAR_{15...0}$ se vodi na adresne linije $A_{15...0}$ memorije ***MEM***.

Multiplekser MX1 je 16-to razredni multiplekser sa 2 ulaza. Na ulaze 0 i 1 multipleksera se vode sadržaji $SP_{15...0}$ i $ADD_{15...0}$, a selekcija jednog od ova dva sadržaja se realizuje vrednostima 0 i 1, respektivno, upravljačkog signala $\mathbf{mxMAR_0}$. Sadržaj $SP_{15...0}$ predstavlja adresu memorijske lokacije koja predstavlja vrh stek, a koristi se priliko stavljanja registara $PC_{15...0}$ i $PSW_{15...0}$ na stek. Sadržaj $ADD_{15...0}$ predstavlja adresu memorijske lokacije počev od koje treba pročitati dva bajta 16-to bitne adrese prekidne rutine.

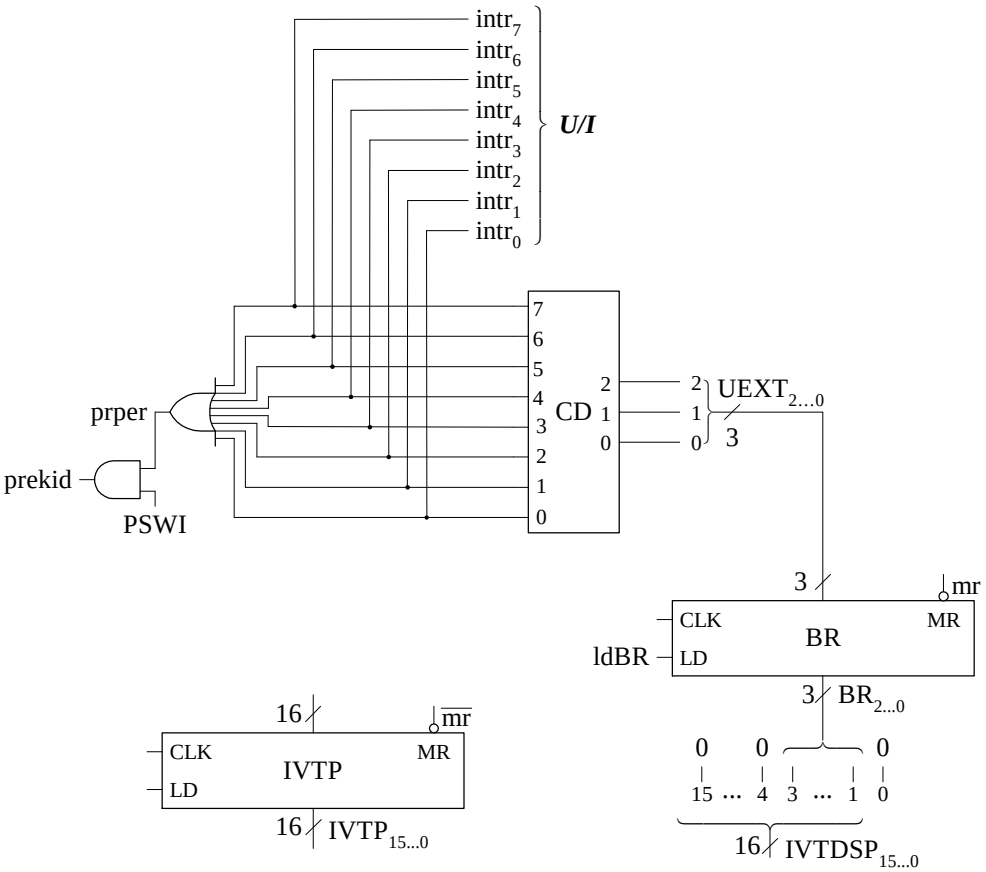
Registar $MDR_{7...0}$ je 8-mo razredni registar podatka u koji se vrednošću 1 signala **ldMDR** upisuje sadržaj sa izlaza multipleksera MX2. Pri realizaciji ciklusa čitanja generiše se vrednosti 1 signala **ldMDR** i binarna vrednost 000 signala $\mathbf{mxMDR_2}$, $\mathbf{mxMDR_1}$ **i** $\mathbf{mxMDR_0}$, čime se sadržaj sa izlaznih linija podataka $DO_{7...0}$ memorije ***MEM*** 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 100 signala $\mathbf{mxMDR_2}$, $\mathbf{mxMDR_1}$ **i** $\mathbf{mxMDR_0}$ čime se jedan od sadržaja $PC_{15...8}$, $PC_{7...0}$, $PSW_{15...8}$ i $PSW_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Sadržaj registra $MDR_{7...0}$ se vodi na ulazne linije podataka $DI_{7...0}$ memorije ***MEM***.



Slika 40 Operaciona jedinica (prvi deo)



Slika 41 Operaciona jedinica (drugi deo)



Slika 42 Operaciona jedinica (treći deo)

Multiplekser MX2 je 16-to razredni multiplekser sa 8 ulaza. Na ulaze 0 do 4 multipleksera se vode sadržaji $DO_{7...0}$, $PC_{15...8}$, $PC_{7...0}$, $PSW_{15...8}$ i $PSW_{7...0}$, a selekcija jednog od ovih sadržaja se realizuje binarnim vrednostima 000 do 100, respektivno, upravljačkih signala **mxMBR$_2$**, **mxMBR$_1$** i **mxMBR$_0$**. Sadržaj $DO_{7...0}$ se propušta kod ciklusa čitanja i predstavlja pročitanu vrednost memorijske lokacije koja po izlaznim linijama podataka $DO_{7...0}$ memorije ***MEM*** dolazi u procesor ***CPU***. Sadržaji $PC_{15...8}$, $PC_{7...0}$, $PSW_{15...8}$ i $PSW_{7...0}$ se propuštaju u slučaju ciklusa upisa u memorijsku lokaciju. Sadržaji $PC_{15...8}$ i $PC_{7...0}$ se propuštaju 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 $SP_{15...0}$ je 16-to razredni ukazivač na vrh steka čiji sadržaj predstavlja adresu memorijske lokacije prilikom upisa na stek registara $PC_{15...0}$ i $PSW_{15...0}$. Sadržaj registra $SP_{15...0}$ se inkrementira vrednošću 1 signala **incSP** pri upisu na stek. Sadržaj registra $SP_{15...0}$ se propušta kroz multiplekser MX1 i upisuje mamorijski adresni registar $MAR_{15...0}$.

Sabirač ADD je 16-to razredni sabirač koji se koristi da bi se sabiranjem $IVTP_{15...0}$ i $IVTDSP_{15...0}$ dobila adrese sa koje treba očitati adresu 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. Sadržaj $ADD_{15...0}$ sa izlaza sabirača se propušta kroz multiplekser MX1 i upisuje mamorijski adresni registar $MAR_{15...0}$.

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 izlaznih linija podataka $DO_{7...0}$ memorije, 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 izlaznih linija podataka $DO_{7...0}$ memorije. Registar $DW_{15...0}$ se koristi da se prihvati 16-to bitna vrednost adrese prekidne rutine iz tabele sa adresama prekidnih rutina i da se posle upiše u programski brojač $PC_{15...0}$.

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. Razredi $PC_{15...8}$ i $PC_{7...0}$ se propuštaju kroz multiplekser MX2 i upisuju u registar $MDR_{7...0}$ prilikom stavljanja registra $PC_{15...0}$ na stek. Prilikom utvrđivanja adrese prekidne rutine, registar $DW_{15...0}$ se koristi da se prihvati 16-to bitna vrednost adrese prekidne rutine iz tabele sa adresama prekidnih rutina i da se posle vrednošću 1 signala **ldPC** upiše u programski brojač $PC_{15...0}$. .

Registar $PSW_{15...0}$ je 16-to razredni registar koji sadrži indikatore programske statusne reči procesora. 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 dok flip-flopovi razreda $PSW_{14...4}$ ne postoje. Flip-flop PSWI sadrži indikator *interrupt*. Flip-flop PSWV sadrži indikator *overflow*. Flip-flop PSWC sadrži indikator *carry/borrow*. Flip-flop PSWZ sadrži indikator *zero*. PSWN sadrži indikator *negative*. Pretpostavlja se da je flip-flop PSWI postavljen na vrednost 1 ili 0 kao rezultat prethodnog izvršavanja posebnih instrukcija kojima se upisivanjem vrednosti 1 ili 0 u ovaj indikator programske statusne reči $PSW_{15...0}$ procesora zadaje režima rada procesora sa reakcijiom na prekide ili sa maskiranjem prekida, respektivno. Takođe se pretpostavlja da se u flip-flopovima PSWV, PSWC, PSWZ i PSWN nalaze vrednosti koje odgovaraju rezultatu zadnje izvršene instrukcije koja postavlja ove indikatore programske statusne reči $PSW_{15...0}$. Razredi $PSW_{15...8}$ i $PSW_{7...0}$ se propuštaju kroz multiplekser MX2 i upisuju u registar $MDR_{7...0}$ prilikom stavljanja registra $PSW_{15...0}$ na stek.

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 $\mathbf{intr_7}$. 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.

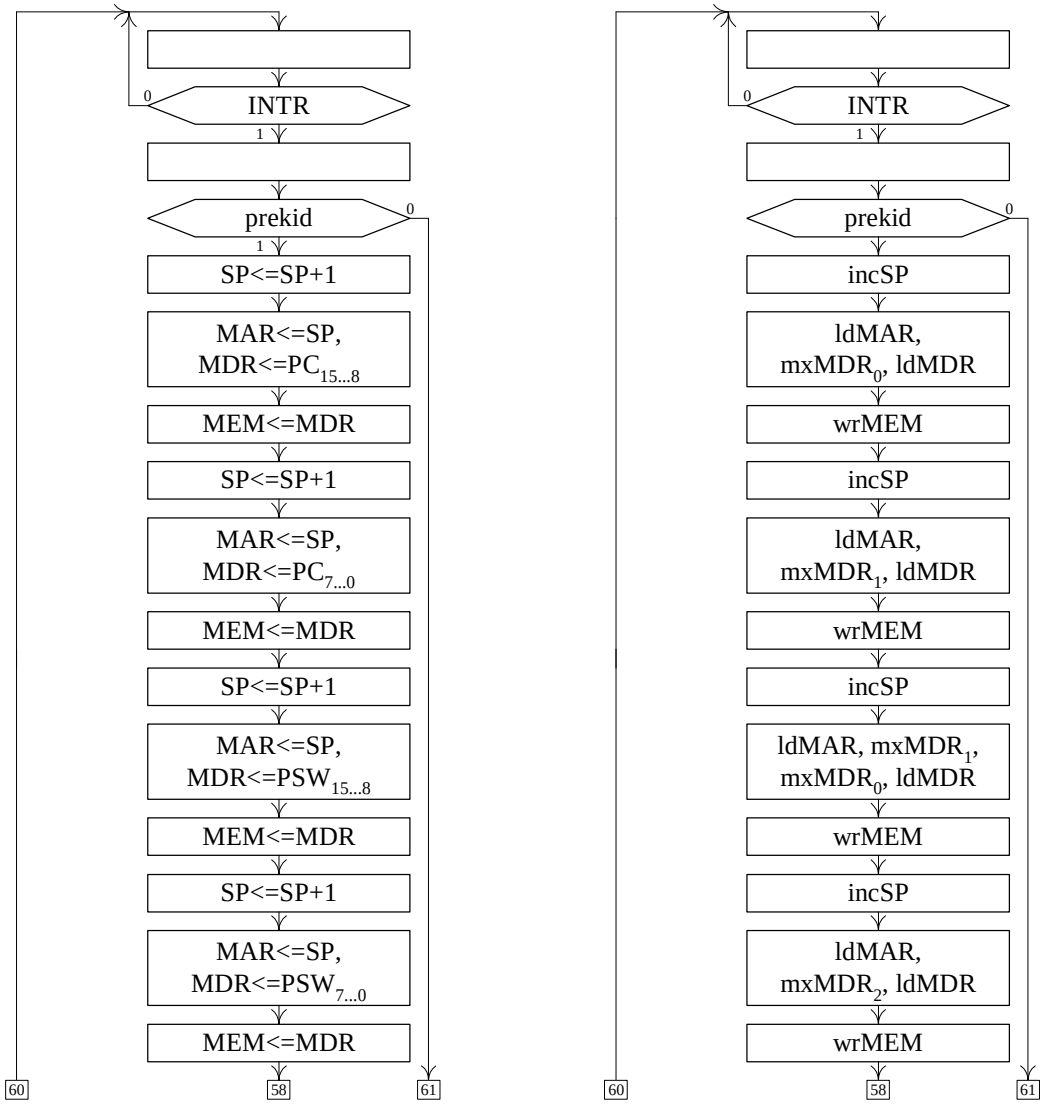
Koder CD služi za formiranje broja ulaza u tabelu sa adresama prekidnih rutina za spoljašnje maskirajuće prekide $intr_0$ do $intr_7$. Signali $intr_0$ do $intr_7$ se vode na ulaze 0 do 7 kodera prioriteta CD po rastućim prioritetima. Na izlazu kodera dobija se 3-bitna binarna vrednost $UEXT_{2...0}$ 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. Pomeraj $IVTDSP_{15...0}$ se vodi na ulaz B sabirača ADD da bi se sabiranjem sa $IVTP_{15...0}$ dobila adresa na kojoj se nalazi adresa prekidne rutine.

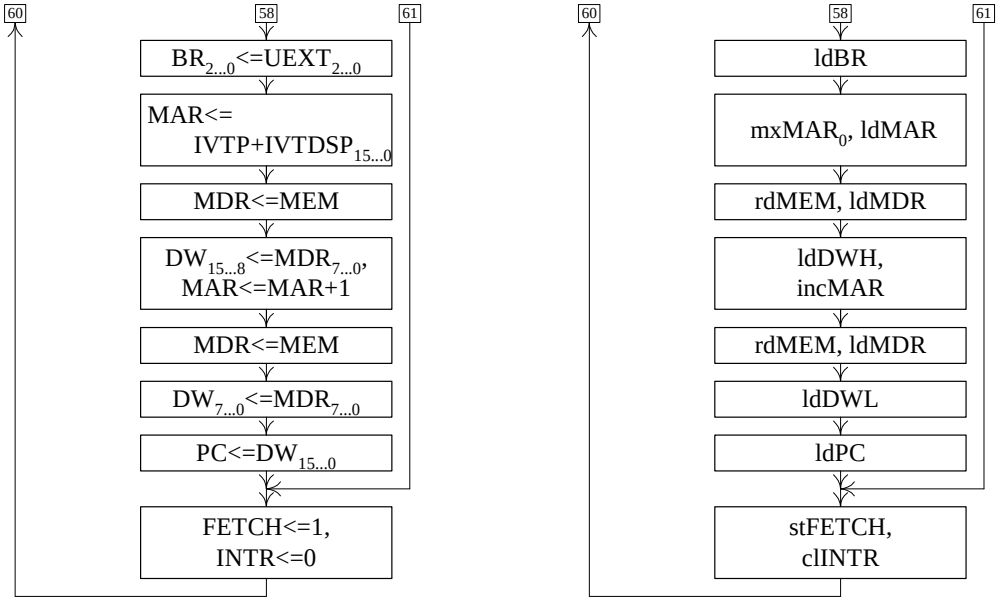
Registar $IVTP_{15...0}$ je ukazivač na tabelu sa adresama prekidnih rutina i sadrži početnu adresu IV tabele. Registar $IVTP_{15...0}$ se vodi na ulaz A sabirača ADD da bi se sabiranjem sa pomerajem $IVTDSP_{15...0}$ dobila adresa na kojoj se nalazi adresa prekidne rutine.

Dijagrami toka mikrooperacija i upravljačkih signala i sekvenca upravljačkih signala

Dijagrami toka mikrooperacija i upravljačkih signala dati su na slikama 43 i 44, a sekvenca upravljačkih signala u tabeli 31.



Slika 43 Dijagrami toka mikrooperacija i upravljačkih signala (prvi deo)



Slika 44 Dijagrami toka mikrooperacija i upravljačkih signala (drugi deo)

Tabela 31 Sekvenca upravljačkih signala

! Opsluživanje prekida !

! Kroz korake bloka *intr* prolaze sve instrukcije. U koracima bloka *intr* se najpre na stek stavljaju programski brojač $PC_{15...0}$ i programska statusna reč $PSW_{15...0}$, a zatim utvrđuje broj ulaza u tabelu sa adresama prekidnih rutina, iz njega čita adresa prekidne rutine i upisuje u programski brojač $PC_{15...0}$. !

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

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

! U koraku $step_{01}$ se, u zavisnosti od toga da li signal **prekid** ima vrednost 0 ili 1, ili završava izvršavanje tekuće instrukcije i prelazi na korak $step_{15}$ u kome se zaustavlja blok *intr* i startuje blok za čitanje sledeće instrukcije *fetch* ili se produžava izvršavanje tekuće instrukcije i prelaskom na korak $step_{02}$ produžava faza opsluživanje prekida tekuće instrukcije. !

$step_{01}$ br (*if* $\overline{\textbf{prekid}}$ *then* $step_{15}$);

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

! Čuvanje konteksta procesora !

! Kontekst procesora i to $PC_{15...0}$ i $PSW_{15...0}$ se čuva u koracima $step_{02}$ do $step_{0D}$. U koracima $step_{02}$ do ste_{07} 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 $step_{02}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{03}$ vrednostima 0 i 1 signala $\textbf{mxMAR}_\textbf{0}$ i **ldMAR**, respektivno, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala $\textbf{mxMDR}_\textbf{0}$ i **ldMDR** sadržaj višeg bajta registra $PC_{15...8}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. U koraku $step_{04}$ se vrednošću 1 signala **wrMEM** realizuje upis jednog bajta iz registra podatka $MDR_{7..0}$, koji je povezan na ulazne linije podataka $DI_{7...0}$ memorije ***MEM***, na adresi određenoj sadržajem adresnog registra $MAR_{15..0}$, koji je povezan na adresne linije $A_{15..0}$ memorije ***MEM***, i prelazi na korak $step_{05}$. Potom se u koraku $step_{05}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{06}$ vrednostima 0 i 1 signala $\textbf{mxMAR}_\textbf{0}$ i **ldMAR**, respektivno, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala $\textbf{mxMDR}_\textbf{1}$ i **ldMDR** sadržaj nižeg bajta registra $PC_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{07}$ na isti načina kao što se to radi u koraku $step_{04}$ prilikom upisa višeg bajta.!

$step_{02}$ **incSP**;
$step_{03}$ **ldMAR**, $\textbf{mxMDR}_\textbf{0}$, **ldMDR**;
$step_{04}$ **wrMEM**;
$step_{05}$ **incSP**;
$step_{06}$ **ldMAR**, $\textbf{mxMDR}_\textbf{1}$, **ldMDR**;
$step_{07}$ **wrMEM**;

! U koracima $step_{08}$ do $step_{0D}$ 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 $step_{08}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{09}$ vrednostima 0 i 1 signala $\textbf{mxMAR}_\textbf{0}$ i **ldMAR**, respektivno, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 i upisuje u registar $MAR_{15...0}$ i vrednostima 1 signala $\textbf{mxMDR}_\textbf{1}$, $\textbf{mxMDR}_\textbf{0}$ i **ldMDR** sadržaj višeg bajta registra $PSW_{15...8}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{0A}$ na isti načina načina kao što se to radi u koraku $step_{04}$ prilikom upisa višeg bajta $PC_{15...8}$. Potom se u koraku $step_{0B}$ vrednošću 1 signala **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se u koraku $step_{0C}$ i vrednostima 1 signala $\textbf{mxMDR}_\textbf{2}$ i **ldMDR** sadržaj nižeg bajta registra $PSW_{7...0}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje u koraku $step_{0D}$ na isti načina kao što se to radi u koraku $step_{0A}$ prilikom upisa višeg bajta. !

$step_{08}$ **incSP**;
$step_{09}$ **ldMAR**, $\textbf{mxMDR}_\textbf{1}$, $\textbf{mxMDR}_\textbf{0}$, **ldMDR**;
$step_{0A}$ **wrMEM**;

$step_{0B}$ **incSP**;
$step_{0C}$ **ldMAR**, **mxMDR$_2$**, **ldMDR**;
$step_{0D}$ **wrMEM**;
! Utvrđivanje broja ulaza !
! U koraku $step_{0E}$ se sadržaj $UEXT_{2...0}$ sa izlaza kodera CD, koji sadrži broj ulaza u tabelu sa adresama prekidnih rutina, vrednošću 1 signala **ldBR** upisuje u registar $BR_{2...0}$. !
$step_{0E}$ **ldBR**;
! Utvrđivanje adrese prekidne rutine !
! U koracima $step_{0F}$ do $step_{14}$ 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 $PC_{15...0}$. U koraku $step_{0F}$ se 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$_0$** i **ldMAR** se sadržaj $ADD_{15...0}$ sa izlaza sabirača ADD propušta kroz multiplekser MX1 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. U koraku $step_{10}$ se vrednošću 1 signala **rdMEM** sa adrese određene sadržajem adresnog registra $MAR_{15..0}$, koji je povezan na adresne linije $A_{15..0}$ memorije ***MEM***, u memoriji ***MEM*** realizuje operacija čitanja, pročitani sadržaj, koji memorija ***MEM*** šalje po izlaznim linijama podataka $DO_{7...0}$, se vrednošću 1 signala **ldMDR** upisuje u registar podatka $MDR_{7..0}$ i prelazi na korak $step_{11}$. U koraku $step_{11}$ se prvi bajt vrednošću 1 signala **ldDWH** upisuje u viši bajt registra $DW_{15...8}$, a vrednošću 1 signala **incMAR** adresni registar $MAR_{15...0}$ inkrementira na adresu sledećeg bajta. U koraku $step_{12}$ se čita drugi bajt i to na isti način kao što se u koraku čita prvi bajt i prelazi na korak $step_{13}$. U koraku $step_{13}$ 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 sadržaj registra $DW_{15...0}$ vrednošću 1 signala **ldPC** upisuje u registar $PC_{15...0}$. Time se u registru $PC_{15...0}$ nalazi adresa prve instrukcije prekidne rutine. Iz koraka $step_{9C}$ se prelazi na $step_{15}$. !
$step_{0F}$ **mxMAR$_0$**, **ldMAR**;
$step_{10}$ **rdMEM**, **ldMDR**;
$step_{11}$ **ldDWH**, **incMAR**;
$step_{12}$ **rdMEM**, **ldMDR**;
$step_{13}$ **ldDWL**;
$step_{14}$ **ldPC**;
! U korak $step_{15}$ se dolazi iz koraka $step_{01}$ i $step_{14}$. U koraku $step_{15}$ se vrednošću 1 signala **clINTR** upisuje vrednost 0 u flip flop INTR bloka *intr*, čime se zaustavlja blok *intr*, dok se vrednošću 1 signala **stFETCH** upisuje vrednost 1 u flip flop FETCH bloka *fetch*, čime se startuje blok *fetch*. !
$step_{15}$ **clINTR, stFETCH**,
br $step_{00}$;

Upravljačka jedinica

Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu sekvence upravljačkih signala po koracima (tabela 31). Mikroprogram se formira tako što se svakom koraku u sekvenci upravljačkih signala po koracima pridruži binarna reč sa slike 45. Te binarna reči se naziva mikroinstrukcija, mikronaredba ili mikrokomanda. Uređeni niz mikroinstrukcija pridruženih koracima u sekvenci upravljačkih signala po koracima naziva se mikroprogram.

Mikroinstrukcija ima dva dela i to operacini deo i upravljački deo. Operacioni deo čine bitovi 0 do 15, a upravljački deo čine bitovi 16 do 27. Operacioni deo se koristi za generisanje upravljačkih signala operacione jedinice, a upravljački deo se koristi za generisanje upravljačkih signala upravljačke jedinice.

0	1	2	3	4	5	6	7
incSP	incMAR	ldMAR	$mxMAR_0$	ldMDR	$mxMDR_2$	$mxMDR_1$	$mxMDR_0$

8	9	10	11	12	13	14	15
wrMEM	rdMEM	ldDWH	ldDWL	ldPC	ldBR	clINTR	stFETCH

16	17	18	19
cc			

20	21	22	23	24	25	26	27
ba							

Slika 45 Mikroinstrukcija

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 i uslovni skok.

Bezuslovni skokovi se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 31) 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 32.

Tabela 32 Signal bezuslovnog skoka

signal bezuslovnog skoka	*cc*
bruncnd	1

Uslovni skokovi se realizuju u onim koracima sekvence upravljačkih signala po koracima (tabela 31) 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 33.

Tabela 33 Signali uslovnih skokova

signal uslovnog skoka	polje *cc*	signal uslova
brnotINTR	2	$\overline{\textbf{INTR}}$
brnotprekid	3	$\overline{\textbf{prekid}}$

Vrednosti 0, 4, 5, 6, 7, 8, 9, A, B, C, D, E i F polja *cc* koje nisu dodeljene signalu bezuslovnog skoka i signalima uslovnih skokova određuju 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 31). 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 28 bitova. Za kodiranje operacionog dela mikroinstrukcije koristi se 16 bitova. Upravljačkih signala operacione jedinice ima 16, pa je usvojena dužina operacionog dela mikroinstrukcije 16 bitova. Za kodiranje upravljačkog dela mikroinstrukcije koristi se 12 bitova. Signala bezuslovnog skoka i signala uslovnih skokova ima 3, ali je umesto 2 bita usvojena dužina polja *cc* upravljačkog dela mikroinstrukcije 4 bita. Ukupan broj koraka u sekvenci upravljačkih signala po koracima je 16, ali je umesto 4 bita usvojena dužina polja *ba* upravljačkog dela mikroinstrukcije 8 bitova. Ovo je učinjeno da bi se dobio pregledniji mikroprogram predstavljen u heksadecimalnom obliku.

Mikroprogram se formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima (tabela 31) 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 ili 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* $step_A$ i *br* (*if* **uslov** *then* $step_A$). 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 32. 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 34.

Tabela 34 Koraci $step_i$, $step_A$, vrednosti $\mathbf{madr_A}$ i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	$\mathbf{madr_A}$	A
$step_{15}$	$step_{00}$	$\mathbf{madr_{00}}$	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 33. 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 35.

Tabela 35 Koraci $step_i$, uslovi **uslov**, koraci $step_A$, vrednosti $\mathbf{madr_A}$ i vrednosti A za uslovne skokove

$step_i$	uslov	$step_A$	$\mathbf{madr_A}$	A
$step_{00}$	$\overline{\mathbf{INTR}}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{01}$	$\overline{\mathbf{prekid}}$	$step_{15}$	$\mathbf{madr_{15}}$	15

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu mikroprogramske realizacije signal bezuslovnog skoka (tabela 32), signali uslovnih skokova (tabela 33) i signali vrednosti A za bezuslovne skokove (tabela 34) i uslovne skokove (tabela 35).

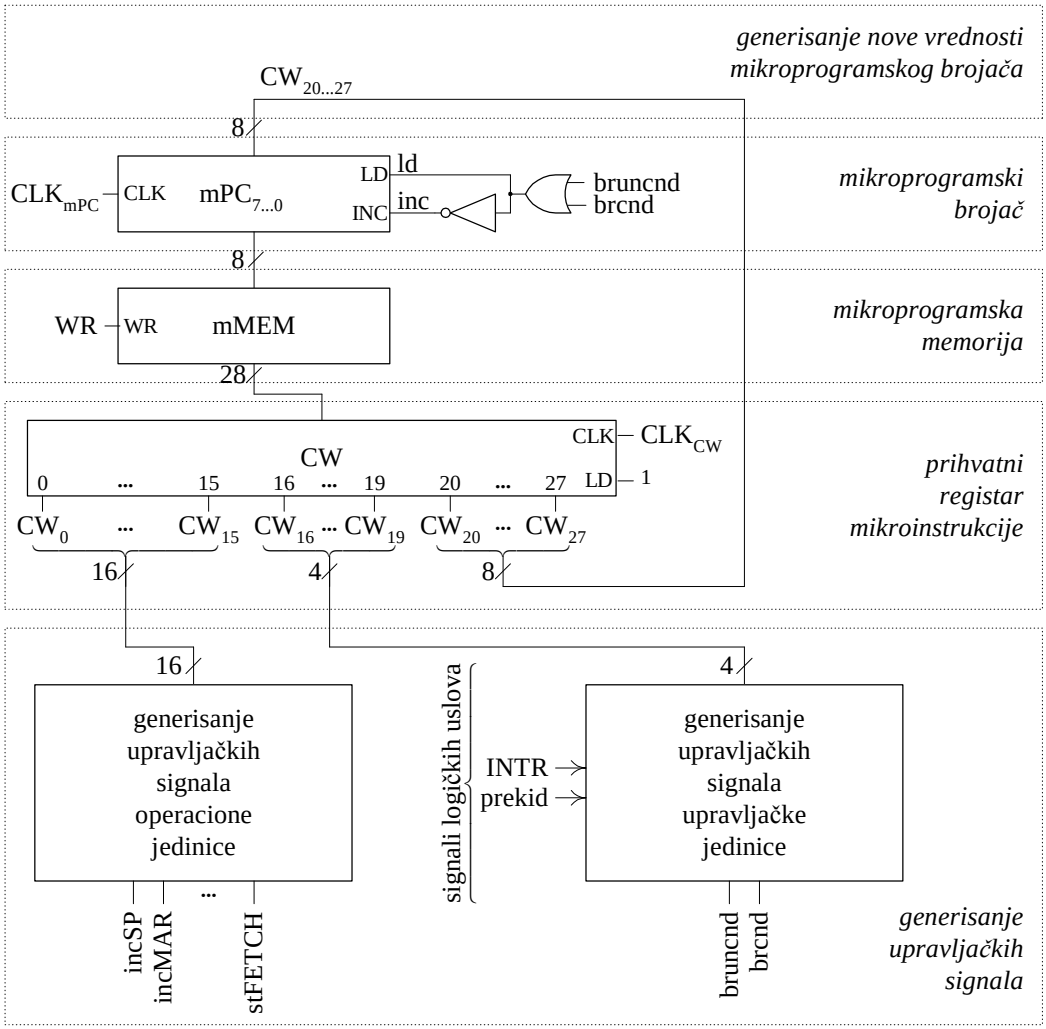
Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 31) formiran mikroprogram (tabela 36). On ima sledeću formu:

- 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 36 Mikroprogram za upravljačku jedinicu mikroprogramske realizacije

```
0 0   0 0 0 0  2  0 0 ! brnotINTR, madr00 !
0 1   0 0 0 0  3  1 5 ! brnotprekid, madr15 !
0 2   8 0 0 0  0  0 0 ! incSP !
0 3   2 9 0 0  0  0 0 ! ldMAR, mxMDR0, ldMDR !
0 4   0 0 8 0  0  0 0 ! wrMEM !
0 5   8 0 0 0  0  0 0 ! incSP !
0 6   2 A 0 0  0  0 0 ! ldMAR, mxMDR1, ldMDR!
0 7   0 0 8 0  0  0 0 ! wrMEM !
0 8   8 0 0 0  0  0 0 ! incSP !
0 9   2 B 0 0  0  0 0 ! ldMAR, mxMDR1, mxMDR0, ldMDR !
0 A   0 0 8 0  0  0 0 ! wrMEM !
0 B   8 0 0 0  0  0 0 ! incSP !
0 C   2 C 0 0  0  0 0 ! ldMAR, mxMDR2, ldMDR !
0 D   0 0 8 0  0  0 0 ! wrMEM !
0 E   0 0 0 4  0  0 0 ! ldBR !
0 F   3 0 0 0  0  0 0 ! mxMAR0, ldMAR !
1 0   0 8 4 0  0  0 0 ! rdMEM, ldMDR !
1 1   4 0 2 0  0  0 0 ! ldDWH, incMAR !
1 2   0 8 4 0  0  0 0 ! rdMEM, ldMDR !
1 3   0 0 1 0  0  0 0 ! ldDWL !
1 4   0 0 0 8  0  0 0 ! ldPC !
1 5   0 0 0 3  1  0 0 ! clINTR, stFETCH, bruncnd, madr00 !
```

Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 46. 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 46 Struktura upravljačke jedinice mikroprogramske realizacije

Blok *generisanje nove vrednosti mikroprogramskog brojača* se sastoji od linija po kojima se dovodi vrednosti koju treba upisati u mikroprogramski brojač $mPC_{7...0}$. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikroprograma. Prihvatni registar mikroinstrukcije $CW_{0...27}$ u svojim razredima $CW_{20...27}$ sadrži vrednost za upis u mikroprogramski brojač $mPC_{7...0}$ za bezuslovne skokove (tabela 34) i uslovne skokove (tabela 35) u mikroprogramu (tabela 36). Sadržaj razreda $CW_{20...27}$ je prisutan 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 36). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko signali **brcnd** i **bruncnd** imaju vrednost 0. Signali **brcnd** i **bruncnd** normalno imaju vrednost 0 sem prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran 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 36). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ukoliko jedan od signala **brcnd** i **bruncnd** ima vrednost 1. Signali **brcnd** i **bruncnd** normalno imaju vrednost 0 sem prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran 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 36). S obzirom da se mikroprogram svih faza izvršavanja instrukcija nalazi u opsegu od adrese 00 do adrese 15, 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 30 bitova, a kapacitet veličinom mikroprograma (tabela 36) i iznosi 256 lokacija. Adresiranje mikroprogramske memorije se realizuje sadržajem mikroprogramskog brojača $mPC_{7...0}$.

Blok *prihvatni registar mikroinstrukcije* sadrži prihvatni registar mikroinstrukcije $CW_{0...27}$. Prihvatni registar mikroinstrukcije $CW_{0...27}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...15}$ i $CW_{16...19}$ se koriste u bloku *generisanje upravljačkih signala* za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{20...27}$ 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 $CW_{0...15}$ prihvatnog registra mikroinstrukcije generišu upravljačke signale operacione jedinice i na osnovu sadržaja razreda $CW_{16...19}$ prihvatnog registra mikroinstrukcije i signala logičkih uslova **INTR** i **prekid** koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice ***oper*** se generišu na sledeći način:

- **incSP** = $\mathbf{CW_0}$
- **incMAR** = $\mathbf{CW_1}$
- **ldMAR** = $\mathbf{CW_2}$
- $\mathbf{mxMAR_0}$ = $\mathbf{CW_3}$
- **ldMDR** = $\mathbf{CW_4}$
- $\mathbf{mxMDR_2}$ = $\mathbf{CW_5}$
- $\mathbf{mxMDR_1}$ = $\mathbf{CW_6}$
- $\mathbf{mxMDR_0}$ = $\mathbf{CW_7}$
- **wrMEM** = $\mathbf{CW_8}$
- **rdMEM** = $\mathbf{CW_9}$
- **ldDWH** = $\mathbf{CW_{10}}$
- **ldDWL** = $\mathbf{CW_{11}}$
- **ldPC** = $\mathbf{CW_{12}}$
- **ldBR** = $\mathbf{CW_{13}}$

- **clINTR** = $\mathbf{CW_{14}}$
- **stFETCH** = $\mathbf{CW_{15}}$

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

- **bruncnd** = $\overline{\mathbf{CW_{16}}} \cdot \overline{\mathbf{CW_{17}}} \cdot \overline{\mathbf{CW_{18}}} \cdot \mathbf{CW_{19}}$
- **brcnd** = **brnotINTR**$\cdot \overline{\textbf{INTR}}$ + **brnotprekid**$\cdot \overline{\textbf{prekid}}$
- **brnotINTR** = $\overline{\mathbf{CW_{16}}} \cdot \overline{\mathbf{CW_{17}}} \cdot \mathbf{CW_{18}} \cdot \overline{\mathbf{CW_{19}}}$
- **brnotprekid** = $\overline{\mathbf{CW_{16}}} \cdot \overline{\mathbf{CW_{17}}} \cdot \mathbf{CW_{18}} \cdot \mathbf{CW_{19}}$

Pri generisanju signala **brcnd** koriste se sledeći signali logičkih uslova koji dolaze iz operacione jedinice ***oper*** i to **INTR** i **prekid**.

5 LITERATURA

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

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

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

4. 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.

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

6. N. Grbanovic, J. Djordjevic, B. Nikolić, *The Software Package for an Educational Computer System*, International Journal on Electrical Engineering Education, Vol. 40, No. 4, Oct 2003, pp. 270-284.

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

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

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

10. J. Đorđević, *Arhitektura računara*, *Edukacioni računarski sistem*, *Arhitektura i organizacija računrskog sistema*, Elektrotehnički fakultet, Beograd, 2002.

11. J. Đorđević, N. Grbanović, B. Nikolić, Z. Radivojević, *Arhitektura računara*, *Edukacioni računarski sistem*, *Priručnik za simulaciju sa zadacima*, Elektrotehnički fakultet, Beograd, 2004.

12. J. Djordjevic, B. Nikolic, A. Milenkovic, "*Flexible Web-based Educational System for Teaching Computer Architecture and Organization*," IEEE, Transactions on Education, Vol. 48, No. 2, 2005.

13. J. Djordjevic, B. Nikolic, M. Mitrovic, "*A Memory System for Education*," Computer Journal, (to appear)