JOVAN ĐORĐEVIĆ

OSNOVI RAČUNARSKE TEHNIKE II

PROJEKTOVANJE UREĐAJA

BEOGRAD, 2013.

SADRŽAJ

1 DIGITALNI UREĐAJI

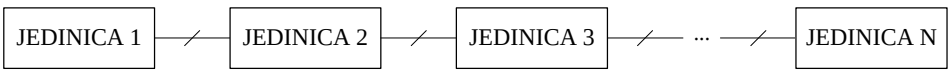
1.1 OSNOVNI POJMOVI

U ovom poglavlju se uvode osnovni pojmovi koji se koriste prilikom razmatranja postupaka projektovanja digitalnih uređaja. U okviru toga se najpre daje usvojena struktura digitalnih uređaja, zatim struktura jedinica digitalnog uređaja kao i mogući načini njihovog povezivanja. Na kraju se daju karakteristike jedinica sa jednom operacijom i sa više operacija koje se u naredna dva poglavlja koriste za ilustraciju opštih postupaka realizacije jedinica digitalnih uređaja.

1.1.1 Struktura digitalnog uređaja

Digitalni uređaj su prekidačke mreže koje realizuju jednu ili više operacija nad podacima koji su predstavljeni u binarnom obliku. Ukoliko uređaj može da realizuje ne jednu već više operacija, onda se i informacije i tome koju operaciju treba realizovati takođe zadaju binarnim vrednostima. Digitalni uređaji su najčešće toliko složene prekidačke mreže da se projektovanje digitalnih uređaja ne može realizovati formalnim postupcima sinteze prekidačkih mreža. Zbog toga postoje samo određeni postupci kojih se treba pridržavati prilikom projektovanja digitalnih uređaja. To dovodi do toga da prilikom projektovanja digitalnih uređaja veliku ulogu ima iskustvo projektanta i da može da bude projektovano više uređaja koji realizuju istu operaciju ili iste operacije a srukturne šeme im se razlikuju.

Jedan od često primenjivanih postupaka je da se izvršavanje operacije ili operacija podeli na logičke celine koje se nazivaju faze i da za izvršavanje svake faze u digitalnom uređaju postoji posebna jedinica (slika 1). Ukoliko je izvršavanje operacije ili operacija podeljeno na N faza, uređaj mora da ima N jedinica. Izvršavanje svake operacije započinje u jedinici 1 na osnovu početnih vrednosti podataka koji se nalaze u registrima i/ili memorijskim lokacijama jedinice 1. Međurezultati faze 1 se prebacuju iz jedinice 1 u jedinicu 2 u kojoj se nastavlja sa izvršavanjem faze 2 operacije. Na sličan način se izvršavaju i ostale faze operacije u ostalim jedinicama uređaja i vrši prebacivanje međurezultata između jedinica. Rezultat operacije se nalazi u registrima i/ili memorijskim lokacijama jedinice N. Primer je izvršavanje instrukcije računara koje se deli na fazu očitavanje instrukcije, fazu dekodovanje instrukcije i čitanje podataka, fazu izvršavanje operacije i fazu opsluživanje prekida.



Slika 1 Struktura digitalnog uređaja

Kod ovog pristupa realizacije uređaja treba razmotriti moguće načine realizacije jedinica i moguće načine njihovog povezivanja.

1.1.2 Realizacija jedinica digitalnog uređaja

U ovom poglavlju se razmatra postupak realizacije jedinice digitalnog uređaja po kome se jedinica sastoji iz operacione i upravljačke jedinice. Stoga se najpre daju strukture operacione i upravljačke jedinice, zatim njihovo međusobno povezivanje i na kraju postupak njihove realizacije.

1.1.2.1 Operaciona jedinica

Operaciona jedinica se sastoji od kombinacionih i sekvencijalnih prekidačkih mreža koje se prema svojoj funkciji mogu podeliti u tri grupe. U prvu grupu spadaju sekvencijalne prekidačke mreže koje služe za pamćenje binarnih reči kojima se definišu operacije koje treba realizovati i podaci nad kojima treba da se realizuju operacije, kao i binarnih reči koje predstavljaju međurezultate i rezultate operacija. To su uglavnom registri i memorije. U drugu grupu spadaju kombinacione i sekvencijalne prekidačke mreže koje realizuju izabrani skup mikrooperacija. To su uglavnom sabirači, aritmetičko logičke jedinice, pomerači, dekoderi, multiplekseri, itd. U treću grupu spadaju kombinacione i sekvencijalne prekidačke mreže koje formiraju signale logičkih uslova. To su uglavnom komparatori i brojači, ali i posebno konstruisane kombinacione i sekvencijalne prekidačke mreže.

Operacije koje treba da se realizuju u jedinici digitalnog uređaja se razlažu na određen broj koraka i to tako da u svaki korak uđe jedna ili više mikrooperacija. Svaki od koraka na koje je razložena operacija se izvršava za vreme trajanja jedne periode signala takta, pa realizacija operacija zahteva onoliko taktova na koliko koraka je razložena operacija. Pri tome se pod mikrooperacijama podrazumevaju operacije koje se realizuju za vreme trajanja jedne periode signala takta u kombinacionim ili sekvencijalnim mrežama kao što su aritmetičko logička jedinica, pomerač ulevo ili udesno, multiplekser, inkrementirajući i dekrementirajući brojači, itd. Skup mikrooperacija dovoljnih da se na njih razloži bilo koja operacija su

- aritmetičke mikrooperacije F=A+B, F=A–B, F=A+1 i F=A–1,
- logičke mikrooperacije F=A$\vee$B, F=A&B, F=A$\oplus$B i F=$\overline{A}$,
- pomeračke mikrooperacije za jedno mesto udesno F=I_R.A(n-1:1) i jedno mesto ulevo F=A(n-2:0).I_L i
- mikrooperacije prenosa kroz multiplekser F=A.

Ove mikrooperacije se tipično realizuju tako što se nad binarnim vrednostima koje se uzimaju iz registara označenih sa A i B u određenoj kombinacionoj mreži (aritmetička jedinica, logička jedinica, pomerač, multiplekser) izvrši određena transformacija i novoformirana vrednost upisuju u registar označen sa F.

Vreme izvršavanja operacije se dobija kao proizvod broja koraka na koje je operacija razložena i vrednosti periode signala takta. Ovo vreme se može smanjiti

- smanjivanjem broja koraka neophodnih za izvršavanje mikrooperacija na koje je jedna operacija razložena i
- smanjivanjem periode signala takta.

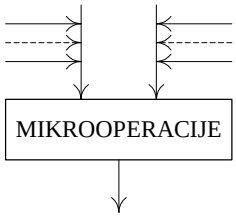
Smanjivanjem broja koraka na koje se razlaže neka operacije se može postići ukoliko se u jednom koraku izvršava ne jedna već više mikrooperacija. Međutim, broj mikrooperacija koje je moguće izvršavati u istom koraku zavisi od toga da li postoji međuzavisnost po podacima između mikrooperacija i da li postoje prekidačke mreže u kojima bi se u istom koraku izvršavale sve one mikrooperacije između kojih ne postoji međuzavisnost po podacima. Prilikom razmatranja mogućnosti da se u jednom koraku realizuje više od jedne mikrooperacije mogu da se jave dve situacije. Prva situacija je da mikrooperacije F=A+B i E=C+D mogu da se realizuju u istom koraku, jer nema međuzavisnosti. Međutim, da bi se to iskoristilo potrebno je da postoje dva sabirača. Ukoliko postoje dva sabirača, u istom koraku se može u jednom sabiraču sabiranjem A i B dobiti F, a u drugom sabiraču sabiranjem C i D dobiti E. Ukoliko postoji samo jedan sabirač u njemu bi se u jednom koraku najpre sabirali A i B, a zatim u sledećem koraku u istoma sabiraču bi se sabirali C i D. Druga je situacija da mikrooperacije F=A+B i E=F+C ne mogu da se realizuju u istom koraku i pored toga što

možda postoje dva sabirača, jer ima međuzavisnosti. Ovde je u jednom koraku potrebno najpre sabiranjem A i B dobiti F, a zatim u sledećem koraku sabiranjem F i C dobiti E.

Smanjivanje periode signala takta se postiže korišćenjem logičkih i memorijskih elemenata sa manjim kašnjenjem. Time se smanjuje kašnjenje prekidačkih mreža koje se koriste za realizaciju mikrooperacija na koje je razložena neka operacija. Međutim, logički i memorijski elementi sa manjim kašnjenjem su zbog samog tehnološkog postupka fabrikacije skuplji i zbog veće disipacije zahtevaju skuplje realizacije sistema za hlađenje nego logički i memorijski elementi sa većim kašnjenjem.

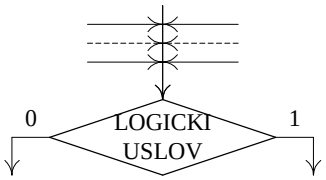
Redosled koraka u kojima se izvršavaju mikrooperacije na koje je razložena neka operacija u opštem slučaju zavisi i od tzv. logičkih uslova koji pokazuju neke karakteristike binarnih reči nad kojima se realizuje operacija. Ovde su tipične dve situacije koje mogu da se jave priliko razlaganja operacija na mikrooperacije. Prva situacija se javlja kada se posle određenog broja koraka neke operacije sračunaju vrednosti A i B, pa posle toga treba produžiti sa jednim od dva moguća niza koraka u zavisnosti od toga da li su A i B jednaki ili ne. U operacionoj jedinici se formira signal logičkog uslova koji ima vrednost 1 ili 0 u zavisnosti od toga da li su A i B jednaki ili ne. Na osnovu vrednosti ovog signala upravljačka jedinica generiše signale kojima se u operacionoj jedinici omogućava izvršavanje jednog od dva moguća niza koraka. Druga situacija se javlja u slučaju operacija gde se određen niz koraka ponavlja više puta. U ovom slučaju se u neki brojač postavlja vrednost koja predstavlja broj ponavljanja. Vrednost brojača se dekrementira prilikom svako izvršavanja datog niza koraka. U operacionoj jedinici se formira signal logičkog uslova koji ima vrednost 1 ili 0 u zavisnosti od toga da li je brojač dekrementiranjem stigao do nule ili nije stigao do nule. Na osnovu vrednosti ovog signala upravljačka jedinica generiše signale kojima se u operacionoj jedinici ili omogućava ponovno izvršavanje datog niza koraka ili prestaje sa njegovim ponavljanjem i prelazi na izvršavanje drugih mikroinstrukcija.

Uređeni niz koraka na koje je razložena neka operacija i logički uslovi na osnovu kojih se određuje redosled koraka definišu algoritam operacije. Za predstavljanje algoritama operacija koriste se dijagrami toka koji se crtaju pomoću operacionih i upravljačkih blokova. U operacioni blok se upisuju mikrooperacije koje se realizuju u istom koraku algoritma (slika 2). Operacioni blok ima jedan ili više ulaza i jedan izlaz. U uslovni blok se upisuje logički uslov koje definiše grananje algoritma, pri čemu 1 označava da je uslov zadovoljen, a 0 da nije (slika 3). Upravljački blok ima jedan ili više ulaza i dva izlaza. Operacioni i uslovni blokovi se povezuju linijama tako da izlaz nekog bloka vodi na ulaz jednog i samo jednog bloka.



Slika 2 Operacioni blok

Mikrooperacije u operacionim blokovima dijagramu toka tipično uzimaju binarne vrednosti iz registara, transformišu ih propuštanjem kroz kombinacione mreže i novoformirane vrednosti upisuju u registre. Zbog toga se kaže da dijagrami toka operacija predstavljaju algoritme operacija na nivou registarskog prenosa (Register Transfer Level).



Slika 3 Upravljački blok

Funkcije operacione jedinice su definisane

- skupom registara i memorija koje mogu da se koriste za čuvanje binarnih reči nad kojima se izvršavaju mikrooperacije i u koje mogu da se smeštaju rezultati mikrooperacija,
- skupom mikrooperacija na koje mogu da se razlože operacije koje treba da se izvršavaju i
- skupom signala logičkih uslova koje može da koristi upravljačka jedinica prilikom određivanja redosleda izvršavanja mikrooperacija.

1.1.2.2 Upravljačka jedinica

Upravljačka jedinica se sastoji iz sekvencijalnih i kombinacionih prekidačkih mreža koje saglasno algoritmu operacije i vrednostima signala logičkih uslova koji se formiraju u operacionoj jedinici generišu upravljačke signale za operacionu jedinicu. Ovi signali određuju koje mikrooperacije i po kom redosledu treba izvršavati u operacionoj jedinici da bi se kao rezultat posle određenog broja signala takta realizovala operacija. Upravljački signali su signali kojima se u operacionoj jedinici

- određuju sadržaji koji u nekom koraku treba da se propuste kroz multipleksere i pojave ne izlazima multipleksera,
- određuje koja aritmetička ili logička operacija treba da se realizuje u ALU,
- vrši upis u registre,
- inkrementira ili dekrementira sadržaj brojača itd.

Upravljačka jedinica se realizuje kao sekvencijalna mreža koja ima onoliko stanja koliko ima koraka na koje je razložena operacija. Posebno stanje sekvencijalne mreže se dodeljuje svakom koraku. U zavisnosti od toga kako se stanje sekvencijalne mreže koristi za generisanje upravljačkih signala operacione jedinice razlikuju se dve grupe realizacija upravljačkih jedinica i to realizacije sa ožičenim i mikroprogramskim generisanjem upravljačkih signala. U slučaju ožičenih realizacija signal dekodovanog stanja sekvencijalne mreže pridružen nekom koraku se koristi za generisanje onih upravljačkih signala operacione jedinice koji u datom koraku treba da imaju vrednost 1. U slučaju mikroprogramskih realizacija stanje sekvencijalne mreže se koristi kao adresa mikroprogramske memorije iz koje se čita kontrolna reč formirana za dati korak i na osnovu njenog sadržaja generišu odgovarajuće vrednosti upravljačkih signala operacione jedinice.

Sekvencijalna mreža upravljačke jedinice treba da prelazi iz stanja u stanje u skladu sa algoritmom operacije i vrednostima signala logičkih uslova. Stoga upravljačka jedinica generiše ne samo upravljačke operacione jedinice kojima se određuje koje mikrooperacije i po kom redosledu treba realizovati, već i upravljačke signale upravljačke jedinice kojima se određuju odgovarajuće promene stanja sekvencijalne mreže.

Funkcije upravljačke jedinice su definisane skupom algoritama odabranih operacija za čiju je realizaciju upravljačka jedinica projektovana da generiše sekvence upravljačkih signala kojima se obezbeđuje izvršavanje odgovarajućih nizova mikrooperacija na koje su odabrane operacije razložene u skladu sa usvojenim algoritmima operacija. Kada u jedinici treba da se izvrši jedna od odabranih operacija, operaciona jedinica generiše poseban signal logičkog

uslova date operacije. Upravljačka jedinica u određenom trenutku proverava vrednosti signala logičkih uslova odabranih operacija i na osnovu toga koji od tih signala ima vrednost jedan generiše sekvencu upravljačkih signala kojom se obezbeđuje izvršavanje odgovarajućeg niza mikrooperacija na koje je data operacija razložena u skladu sa usvojenom algoritmom operacije.

1.1.2.3 Projektovanje operacione i upravljačke jedinice

Projektovanje operacionih i upravljačkih jedinica se realizuje u dve faze.

U prvoj fazi se paralelno projektuju operaciona jedinica i dijagrami toka zadatih operacija. Ovo se obično realizuje u više iteracija i zasniva se na veštini i iskustvu projektanata, jer formalnih metoda nema. Na početku se na osnovu algoritama operacija kreće sa određenom strukturom operacione jedinice tako što se uključi određen broj prekidačkih mreža međusobno povezanih na način neophodan da se u njima čuvaju binarne reči nad kojima treba da se izvršavaju mikrooperacije i binarne reči koje predstavljaju rezultate izvršenih mikrooperacija, izvršavaju mikrooperacije na koje treba da se razlažu operacije i generišu signali logičkih uslova za potrebna grananja prilikom izvršavanja mikrooperacija. Na osnovu toga se kreće sa formiranjem dijagrama toka. U nekom trenutku formiranja dijagrama toka može se utvrditi da je potreban još neki registar za čuvanje rezultata mikrooperacija, zatim još neka kombinaciona mreža za realizaciju neke od mikrooperacija koja nije bila predviđena na početku ili dodatna kombinaciona mreža neophodna da bi se veći broj mikrooperacija realizovao u istom koraku i na kraju neka kombinaciona i/ili sekvencijalna mreža za generisanje signala logičkih uslova za koje na početku nije bilo jasno da su potrebni. Posle ovih izmena u operacionoj jedinici produžava se sa formiranjem dijagrama toka. U nekom trenutku kasnije može opet da se javi potreba za određenim izmenama u strukturnoj šemi opracione jedinice i to da se dodaju i/ili drugačije povežu prekidačke mreže. Posle određenog broja ovakvih iteraciaja, dolazi se do konačne strukturne šeme operadione jedinice i konačnog dijagrama toka.

U drugoj fazi se na osnovu dijagrama toka formiranog u prvoj fazi projektuje upravljačka jedinica. Ova faza se realizuje u jednom prolazu po formalnom postupku definisanom za odabrani tip realizacije upravljačke jedinice.

1.1.3 Povezivanje jedinica digitalnog uređaja

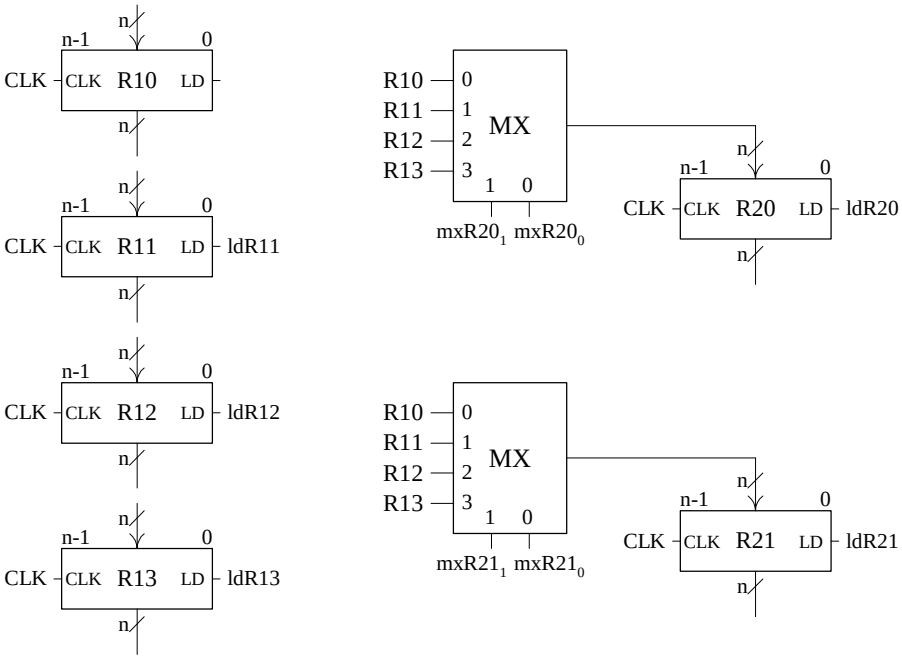
U ovom poglavlju se razmatra postupak povezivanja jedinica digitalnog uređaja kojim se prebacuju binarne reči iz jedne jedinice u drugu jedinicu. Prebacivanje binarnih reči između jedinica uređaja se svodi na postupak povezivanja izlaza registara iz jedinice iz koje se šalju binarne reči na ulaze registara jedinice u koju se šalje binarne reči.

Povezivanja registara se može realizovati na više načina, pri čemu su oni varijante dva osnovna načina povezivanja i to povezivanje direktnim vezama i povezivanje magistralom. Kao primer su uzeta četiri registra R10, R11, R12 i R13 iz jedinice 1 čije sadržaje treba prebacivati u registre R20 i R21 jedinice 2. Svi registri su dužine n razreda.

1.1.3.1 Povezivanje direktnim vezama

Kod ovog načina povezivanja postoje posebne veze kojima se izlazi registara R10, R11, R12 i R13 jedinice 1 vode preko multipleksera mxR20 i mxR21 na ulaze registara R20 i R21 jedinice 2, respektivno. Multiplekserima mxR20 i mxR21 koji su ispred registara R20 i R21 jedinice 2 se nezavisno za svaki od registara R20 i R21 jedinice 2 selektuje po jedan od registara R10, R11, R12 i R13 jedinice 1. Signalima $mxR20_1$ i $mxR20_0$ se selektuje jedan od registara R10, R11, R12 i R13 jedinice 1 i signalom ldR20 upisuje u registar R20 jedinice 2.

Signalima $mxR21_1$ i $mxR21_0$ se selektuje jedan od registara R10, R11, R12 i R13 jedinice 1 i signalom ldR21 upisuje u registar R21 jedinice 2.



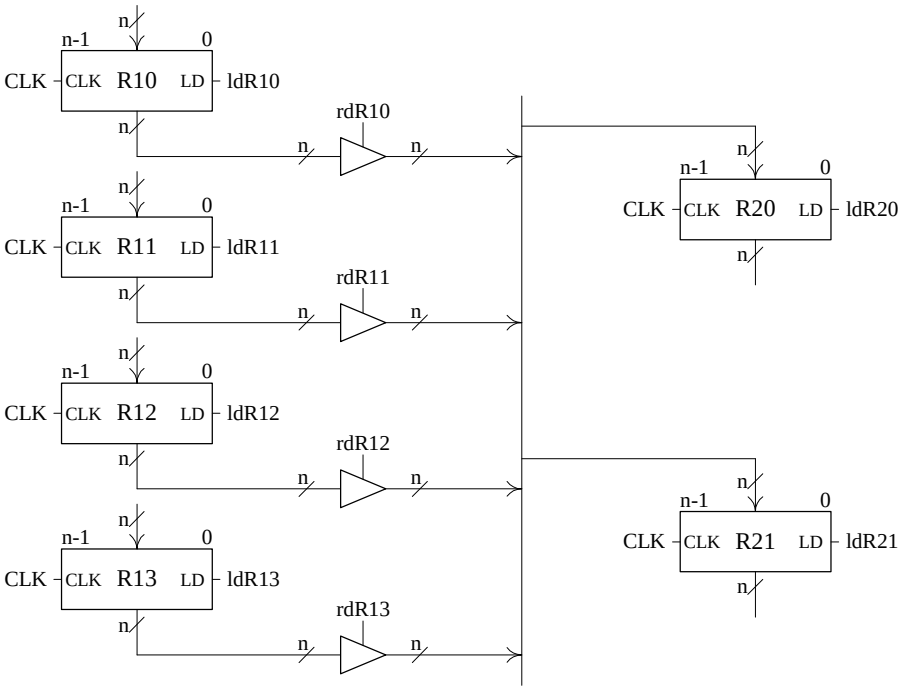
Slika 4 Povezivanje jedinica direktnim vezama

Dobra strana povezivanja direktnim vezama je da se u istom koraku može realizovati mikrooperacija prenosa sadržaja jednog od registara R10, R11, R12 i R13 jedinice 1 u registar R20 jedinice 2 i mikrooperacija prenosa sadržaj jednog od registara R10, R11, R12 i R13 jedinice 1 u registar R21 jedinice 2. Ukoliko treba preneti sadržaj registra R11 jedinice 1 u registar R20 jedinice 2 i sadržaj registra R12 jedinice 1 u registar R21 jedinice 2 to se može realizovati u istom koraku generisanjem vrednosti 0, 1 i 1 signala $mxR20_1$, $mxR20_0$ i ldR20, respektivno, i vrednosti 1, 0 i 1 signala $mxR21_1$, $mxR21_0$ i ldR20, respektivno. Vrednostima 0, 1 i 1 signala $mxR20_1$, $mxR20_0$ i ldR20, respektivno, se sadržaj registra R11 propušta kroz multiplekser mxR20 i upisuje u registar R20. Istovremeno se vrednostima 1, 0 i 1 signala $mxR21_1$, $mxR21_0$ i ldR21, respektivno, sadržaj registra R12 propušta kroz multiplekser mxR21 i upisuje u registar R21.

Loša strana povezivanja direktnim vezama je da je potreban veliki broj multipleksera. Registri su sa *n* razreda i za svaki razred je potreban jedan multiplekser sa četiri ulaza i jednim izlazom. Stoga je za prenos sadržaja jednog od registara R10, R11, R12 i R13 jedinice 1 u registar R20 jedinice 2 potrebno *n* multipleksera sa četiri ulaza i jednim izlazom. Takođe je za prenos sadržaj jednog od registara R10, R11, R12 i R13 jedinice 1 u registar R21 jedinice 2 potrebno *n* multipleksera sa četiri ulaza i jednim izlazom.

1.1.3.2 Povezivanje magistralom

Kod ovog načina povezivanja ne postoje posebne veze između registara R10, R11, R12 i R13 jedinice 1 i registara R20 i R21 jedinice 2. Umesto toga registri R10, R11, R12 i R13 jedinice 1 se preko bafera sa tri stanja povezuju na linije magistrale, a linije magistrale se vode na ulaze registara R20 i R21 jedinice 2. Vrednošću 1 samo jednog od signala rdR10, rdR11, rdR12 i rdR13 jedinice 1 sadržaj samo jednog od registara R10, R11, R12 i R13 jedinice 1 se pušta na linije magistrale, a vrednošću 1 jednog od signala ldR20 ili ldR21 ili oba signala, sadržaj sa linija magistrale se upisuje u jedan od registara R20 ili R21 ili u oba registra.



Slika 5 Povezivanje jedinica magistralom

Dobra strana povezivanja magistralom je da ne postoji potreba za multiplekserima, već da se umesto njih koriste baferi sa tri stanja. Registri su sa *n* razreda i za svaki razred svakog od registara R10, R11, R12 i R13 jedinice 1 je potreban jedan bafer sa tri stanja. Stoga je za prenos sadržaja jednog od registara R10, R11, R12 i R13 jedinice 1 u registre R20 i R21 jedinice 2 potrebno četiri puta po *n* bafera sa tri stanja.

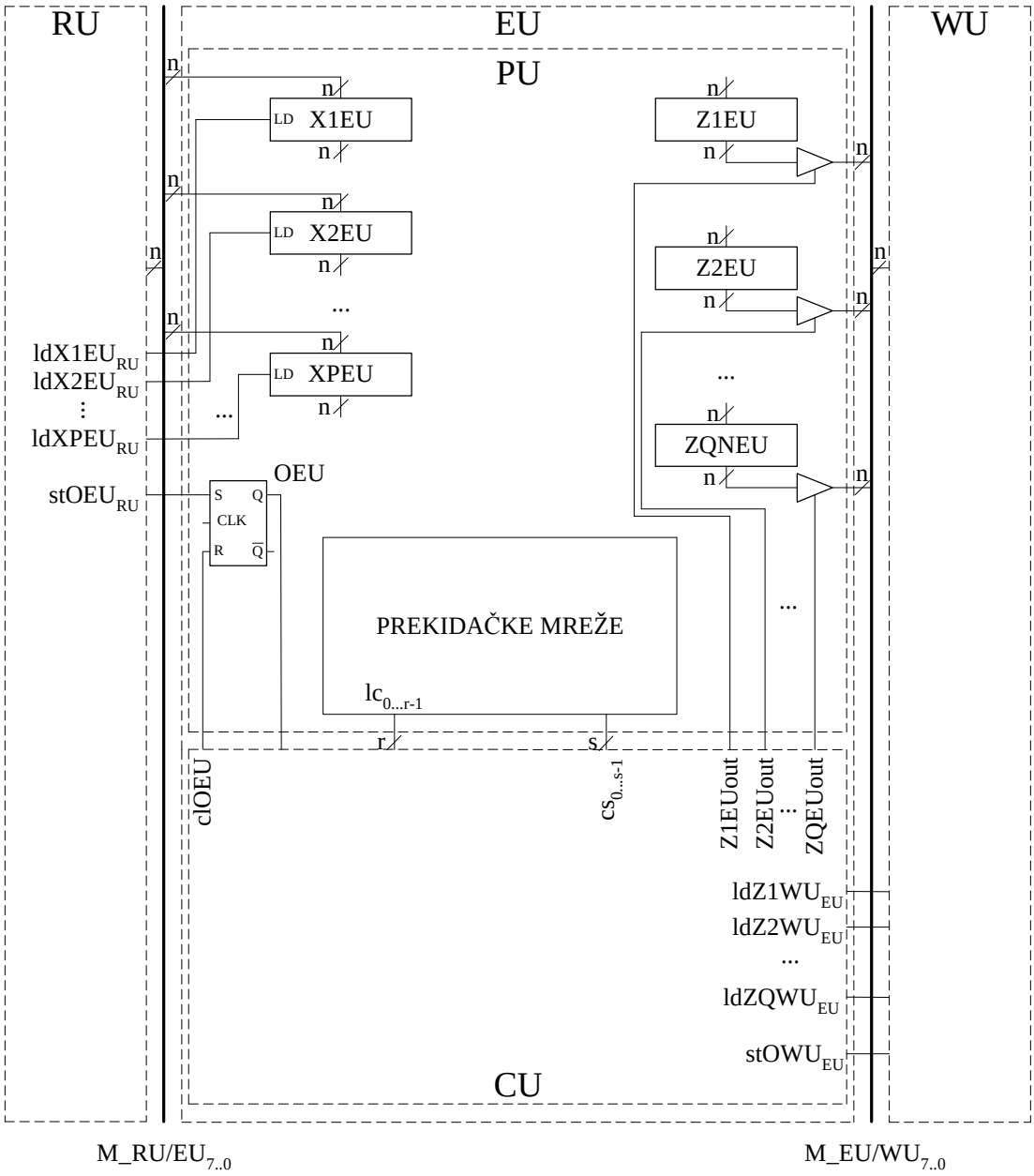
Loša strana je da se istovremeno ne može prenosti sadržaj jednog od registara R10, R11, R12 i R13 jedinice 1 u registar R20 jedinice 2 i jednog od registara R10, R11, R12 i R13 jedinice 1 u registar R21 jedinice 2, već se to mora uraditi u dva posebna koraka. To je zbog toga što se u jednom koraku vrednošću 1 jednog od signala rdR10, rdR11, rdR12 i rdR13 jedinice 1 sadržaj samo jednog od registara R10, R11, R12 i R13 jedinice 1 pušta na linije magistrale, a vrednošću 1 signala ldR20 i/ili ldR21 sadržaj sa linija magistrale upisuje u registar R20 i/ili registar R21. Stoga ukoliko treba preneti sadržaj registra R11 jedinice 1 u registar R20 jedinice 2 i sadržaj registra R12 jedinice 1 u registar R21 jedinice 2 to se mora realizovati u dva posebna koraka generisanjem vrednosti 1 signala rdR11 i ldR20 i jednom koraku i vrednosti 1 signala rdR12 i ldR21 u drugom koraku. U jednom koraku se vrednostima 1 signala rdR11 i ldR20 sadržaj registra R11 pušta na linije magistrale i upisuje u registar R20, respektivno, a u drugom koraku se vrednostima 1 signala rdR12 i ldR21 sadržaj registra R12 pušta na linije magistrale i upisuje u registar R21, respektivno,

1.1.4 Realizacija jedinica sa jednom i više operacija

U naredna dva poglavlja se razmatraju najpre jedinica sa jednom operacijiom a zatim i jedinica sa više operacija. Jedinica sa jednom operacijom realizuje aritmetičku operaciju množenja dve 8 bitne celobrojne vrednosti bez znaka. Jedinica sa više operacija realizuje aritmetičke operacije i to sabiranje, oduzimanje, inkrementiranje i dekrementiranje, logičke operacije i to I, ILI, ekskluzivno ILI i invertovanje, operacije pomeranja udesno i to aritmetičko pomeranje, logičko pomeranje, rotiranje i rotiranje sa indikatorom prenosa, operacije pomeranja ulevo i to aritmetičko pomeranje, logičko pomeranje, rotiranje i rotiranje sa indikatorom prenosa i aritmetičke operacije množenja i deljenja i to nad 8 bitnim

celobrojnim vrednostima bez znaka u slučaju aritmetičkih operacija i 8 bitnim binarnim vrednostima u slučaju logičkih operacija i operacija pomeranja.

Realizacija jedinica sa jednom i više operacija podrazumevaće da je jedinica koja se razmatra deo nekog uređaja koji se sastoji iz više jedinica (slika 6). Stoga jedinica koja se razmatra najpre dobija binarne vrednosti nad kojima treba da realizuje određenu operaciju od jedinice koja je njen prethodnik, zatim u skladu sa usvojenin algoritmom izvršava određenu operaciju i na kraju rezultat izvršene operacije predaje jedinici koja je njen sledbenik. Jedinica koja se razmatra se dalje naziva jedinica EU (Execution Unit), jedinica koja je njen prethodnik jedinica RU (Read Unit) i jedinica koja je njem sledbenik jedinica WU (Write Unit). Ovi nazivi jedinica su uzeti uz pretpostavku da jedinci RU čita podatke sa određenih lokacija i prosleđuje jedinici EU koja nad njima izvršava određene operacije i dobijene rezultate prosleđuje jedinici WU koja ih upisuje u određene lokacije.



Slika 6 Povezivanje jedinice EU sa jedinicama RU i WU i struktura jedinice EU

U određenom trenutku samo jedna od ove tri jedinice može da bude aktivna. Stoga u jedinicama RU, EU i WU postoje flip-flopovi ORU (Operational Read Unit), OEU (Operational Execution Unit) i OWU (Operational Write Unit) koji vrednostima 1 i 0 određuju da li je određena jedinica aktivna ili neaktivna, respektivno. Dok je jedinica RU aktivna u njenom flip-flopu ORU je vrednost jedan, a u flip-flopovima OEU i OWU jedinica EU i WU koje teba da budu neaktivne su nule. Na kraju svog rada jedinica RU u svoj flip-flop ORU upisuje vrednost nula a u flip-flop OEU vrednost 1. Time jedinica RU postaje neaktivna, a jedinica EU aktivna. Na kraju svog rada jedinica EU u flip-flop OEU upisuje vrednost nula a u flip-flop OWU vrednost 1. Time jedinica EU postaje neaktivna, a jedinica WU aktivna.

Jedinica RU i EU su povezane linijama magistrale $M\_RU/EU_{n-1..0}$ dužine n bitova. Početne vrednosti X1, X2,..., XP u p posebnih ciklusa jedinica RU šalje po linijama magistrale $M\_RU/EU_{n-1..0}$ i u registre $X1EU_{n-1..0}$, $X2EU_{n-1..0}$,..., $XPEU_{n-1..0}$ jedinice EU upisuje vrednostima 1 upravljačkih signala $ldX1EU_{RU}$, $ldX2EU_{RU}$, ..., $ldXPEU_{RU}$, respektivno. Jedinica RU generiše vrednost 1 upravljačkog signala $ldX1EU_{RU}$ onda kada na linije $M\_RU/EU_{n-1..0}$ pušta vrednost X1 koja treba da se upiše u registar $X1EU_{n-1..0}$ jedinice EU, vrednost 1 upravljačkog signala $ldX2EU_{RU}$ onda kada na linije $M\_RU/EU_{n-1..0}$ pušta sadržaj koji treba da se upiše u registar $X2EU_{n-1..0}$ i tako redom do vrednosti 1 upravljačkog signala $ldXPEU_{RU}$ onda kada na linije $M\_RU/EU_{n-1..0}$ pušta sadržaj koji treba da se upiše u registar $XPEU_{n-1..0}$. Izvršavanje operacije u jedinici EU se startuje kada jedinica RU vrednošću 1 upravljačkog signala signala $stOEU_{RU}$ u flip-flop OEU jedinice EU upiše vrednost 1.

Jedinica EU i WU su povezane linijama magistrale $M\_EU/WU_{n-1..0}$ dužine n bitova. Tokom izvršavanja određene operacije vrednosti Z1, Z2,..., ZQ koje predstavljaju rezultat izvršavanja operacije jedinica EU upisuje u registre $Z1EU_{n-1..0}$, $Z2EU_{n-1..0}$,..., $ZQEU_{n-1..0}$. Pri kraju izvršavanja operacije jedinica EU u q posebnih ciklusa šalje vrednosti Z1, Z2,..., ZQ iz registara $Z1EU_{n-1..0}$, $Z2EU_{n-1..0}$,..., $ZQEU_{n-1..0}$ na linije magistrale $M\_EU/WU_{n-1..0}$ i u registre $Z1WU_{n-1..0}$, $Z2WU_{n-1..0}$,..., $ZQWU_{n-1..0}$ jedinice WU upisuje vrednostima 1 upravljačkih signala $ldZ1WU_{EU}$, $ldZ2WU_{EU}$, ..., $ldZQ_{EU}$, respektivno. Jedinica EU generiše vrednost 1 upravljačkog signala Z1EUout da bi otvaranjem bafera sa tri stanje na linije $M\_EU/WU_{n-1..0}$ pustila vrednost Z1 i vrednost 1 upravljačkog signala $ldZ1WU_{EU}$ da bi datu vrednost upisala u registar $Z1WU_{n-1..0}$ jedinice WU, vrednost 1 upravljačkog signala Z2EUout da bi otvaranjem bafera sa tri stanje na linije $M\_EU/WU_{n-1..0}$ pustila vrednost Z2 i vrednost 1 upravljačkog signala $ldZ2WU_{EU}$ da bi datu vrednost upisala u registar $Z2WU_{n-1..0}$ jedinice WU i tako redom do vrednosti 1 upravljačkog signala ZQEUout da bi otvaranjem bafera sa tri stanje na linije $M\_EU/WU_{n-1..0}$ pustila vrednost ZQ i vrednost 1 upravljačkog signala $ldZQWU_{EU}$ da bi datu vrednost upisala u registar $ZQWU_{n-1..0}$ jedinice WU. Izvršavanje operacije u jedinici EU se zaustavlja kada jedinica EU vrednošću 1 upravljačkog signala signala clOEU u flip-flop OEU jedinice EU upiše vrednost 0, dok se izvršavanje operacije u jedinici WU startuje kada jedinica EU vrednošću 1 upravljačkog signala $stOWU_{EU}$ u flip-flop OWU jedinice WU upiše vrednost 1.

Jedinca EU se sasatoji od operacione jedinice PU (Processing Unit) i upravljačke jedinice CU (Control Unit).

Operaciona jedinica ima registre $X1EU_{n-1..0}$, $X2EU_{n-1..0}$,..., $XPEU_{n-1..0}$ za čuvanje početnih vrednosti nad kojima operacija treba da se izvršava, zatim registre $Z1EU_{n-1..0}$, $Z2EU_{n-1..0}$,..., $ZQEU_{n-1..0}$ u koje smešta rezultate operacije, kao i registre, koji se nalaze u bloku PREKIDAČKE MREŽE, za smeštanje međurezultata tokom izvršavanja operacije. U bloku PREKIDAČKE MREŽE se nalaze i kombinacione prekidačke mreže, kao što su sabirači, aritmetičko logičke jedinice, pomerači, dekoderi, multiplekseri, itd., koje realizuju izabrani

skup mikrooperacija na koje se operacije razlažu, i kombinacione i sekvencijalne prekidačke mreže, kao što su komparatori, brojači, itd., koje formiraju signale logičkih uslova. Jedan od signala logičkih uslova je signal OEU koji vrednostima 1 i 0 određuje da li upravljačka jedinica treba ili ne treba da generiše upravljačke signale operacione jedinice. Preostali signali logičkih uslova, koji dolaze iz bloka PREKIDAČKE MREŽE i koji su označeni kao signali $lc_{0...r-1}$, vrednostima 0 i 1 definišu da li generisanje upravljačkih signala operacione jedinice treba produžiti sa tekućom sekvencom ili treba preći na novu sekvencu.

Upravljačka jedinica se sastoji iz sekvencijalnih i kombinacionih prekidačkih mreža koje saglasno algoritmu operacije i vrednostima signala logičkih uslova generišu upravljačke signale koji određuju koje mikrooperacije i po kom redosledu treba izvršavati u operacionoj jedinici da bi se kao rezultat posle određenog broja signala takta realizovala operacija. Neki od upravljačkih signala operacione jedinice su signali Z1EUout, Z2EUout, ..., ZQEUout kojima se otvaraju baferi sa tri stanja onda kada se sadržaji registara $Z1EU_{n-1..0}$, $Z2EU_{n-1..0}$,..., $ZQEU_{n-1..0}$ upuštaju na linije magistrale $M\_EU/WU_{n-1..0}$ dužine, kao i signali $ldZ1WU_{EU}$, $ldZ2WU_{EU}$, ..., $ldZQWU_{EU}$, kojima se sadržaj sa magistrale upisuje u registre $Z1WU_{n-1..0}$, $Z2WU_{n-1..0}$,..., $ZQWU_{n-1..0}$ jedinice WU. Upravljački signali su i signal clOEU kojim se upisuje vrednost 0 u flip-flop OEU i time jedinica EU zaustavlja, kao i signal $stOWU_{EU}$, kojim se upisuje vrednost 1 u flip-flop OWU i time jedinica WU startuje. Preostali upravljački signali operacione jedinice, koji odlaze u blok PREKIDAČKE MREŽE i označeni su kao signali $cs_{0...s-1}$, vrednostima 0 i 1 određuju koje sadržaje treba prupuštati kroz određene multipleksere, koje aritmetičke i logičke operacije treba realizovati u ALU, u koje registre treba upisivati nove vrednosti itd.

Za svaku jedinicu se daje realizacija operacione jedinice i upravljačke jedinice. U okviru realizacije operacione jedinice se daju usvojeni algoritmi operacija, strukturna šema operacione jedinice sa direktnim vezama, jednom, dve i tri interne magistrale i dijagrami toka operacija. U okviru realizacije upravljačke jedinice se daju tri tehnike ožičene realizacije i dve tehnike mikroprogramske realizacije.

Prilikom ovih razmatranja ukazaće se na značaj izbora odgovarajučeg algoritma. Pokazaće se da u zavisnosti od izbora algoritma zavisi složenost operacione jedinice i brzina izvršavanja operacija. Pored toga ukazaće se da je operacionu i upravljačku jedinicu moguće realizovati na više različitih načina. Pokazaće se da su neki načini realizacije ovih jedinica pogodniji ukoliko se radi o jednostavnim operacijama a drugi načini realizacije pogodniji ukoliko se radi o složenijim operacijama

1.2 JEDINICA SA JEDNOM OPERACIJIOM

U ovom poglavlju se daju moguće postupci realizacije operacione i upravljačke jedinice za slučaj jedinice sa jednom operacijom koja realizuje aritmetičku operaciju množenja dve 8 bitne celobrojne vrednosti bez znaka.

1.2.1 OPERACIONA JEDINICA

U ovom odeljku se najpre daje algoritam operacije, zatim strukturna šema operacione jedinice i na kraju algoritam generisanja upravljačkih signala.

1.2.1.1 Algoritam operacije

Aritmetička operacija MULU množi 8 bitne celobrojne vrednosti bez znaka $X_7X_6...X_1X_0$ i $Y_7Y_6...Y_1Y_0$ i kao rezultat formira 16 bitnu celobrojnu vrednost bez znaka proizvoda $P_{15}P_{14}...P_1P_0$. Algoritam za izračunavanje proizvoda je definisan je sa

$$P = \sum_{i=0}^{7} XY_i 2^i$$

a može se napisati i kao

$$P=0+X\cdot Y_0\cdot 2^0+X\cdot Y_1\cdot 2^1+...+X\cdot Y_i\cdot 2^i+...X\cdot Y_7\cdot 2^7$$

gde je

$$P= P_{15}P_{14}...P_1P_0$$

i ima dužinu 16 bita.

Po ovom postupku množenje se realizuje u 8 iteracija tako što se računaju trenutni proizvodi P(0), P(1) do P(7), pri čemu je konačan proizvod P jednak zadnjem trenutnom proizvodu P(7).

Ako se trenutni proizvod

$$XY_02^0+XY_12^1+...+XY_i2^i$$

označi sa P(i), onda se proizvod može računati na sledeći način:

$$P(0)=0+X\cdot Y_0\cdot 2^0$$
$$P(1)=P(0)+X\cdot Y_1\cdot 2^1$$
...
$$P(i)=P(i-1)+X\cdot Y_i\cdot 2^i$$
...
$$P(7)=P(6)+X\cdot Y_7\cdot 2^7$$

gde je

$$P(7)=P.$$

i za realizaciju operacije množenja po ovom algoritmu potreban je ALU dužine 16 razreda koja realizuje sabiranje na dužini od 16 razreda.

Međutim, ovaj izraz se može modifikovati na sledeći način

$$P=\sum_{i=0}^{7} X\cdot Y_i\cdot 2^i = \sum_{i=0}^{7} X\cdot Y_i\cdot 2^i\cdot 2^8\cdot 2^{-8} = \sum_{i=0}^{7} (X\cdot Y_i)\cdot 2^8\cdot 2^{-(8-i)}$$

U tom slučaju umesto da se piše

$$P=0+X\cdot Y_0\cdot 2^0 \quad +X\cdot Y_1\cdot 2^1 \quad +...+X\cdot Y_i\cdot 2^i \quad +...+X\cdot Y_7\cdot 2^7$$

može se napisati i kao

$$P=0+X\cdot Y_0\cdot 2^8\cdot 2^{-8}+X\cdot Y_1\cdot 2^8\cdot 2^{-7}+...+X\cdot Y_i\cdot 2^8\cdot 2^{-(8-i)}+...+X\cdot Y_7\cdot 2^8\cdot 2^{-1}$$

ili dalje i kao

$$P=(...(...((0+X\cdot Y_0\cdot 2^8)\cdot 2^{-1}+X\cdot Y_1\cdot 2^8)\cdot 2^{-1}+...+ X\cdot Y_i\cdot 2^8)\cdot 2^{-1} +...+ X\cdot Y_7\cdot 2^8)2^{-1}$$

Ako se trenutni proizvod

$(\cdots((0+XY_0 2^8)2^{-1}+XY_1 2^8)2^{-1}+\cdots+XY_i 2^8)2^{-1}$ označi sa P(i), onda se proizvod može računati na sledeći način:

$$P(0)=(0+XY_0 2^8)2^{-1},$$
$$P(1)=(P(0)+XY_1 2^8)2^{-1},$$
$$\vdots$$
$$P(i)=(P(i-1)+XY_i 2^8)2^{-1},$$
$$\vdots$$
$$P(7)=(P(6)+XY_7 2^8)2^{-1}.$$

Proizvod P se izračunava u osam iteracija tako što se u svakoj iteraciji izračunava trenutni proizvod. Najpre se u prvoj iteraciji izračunava P(0) tako što se na početnu vrednost trenutnog proizvoda koja je 0 dodaje $XY_0 2^8$ i dobijena suma S(0) pomera udesno za jedno mesto. Potom se u drugoj iteraciji izračunava P(1) tako što se na trenutni proizvod P(0) dodaje $XY_1 2^8$ i dobijena suma S(1) pomera udesno za jedno mesto. U opštem slučaju se u (i+1)-oj iteraciji izračunava P(i) tako što se na trenutni proizvod P(i-1) dodaje $XY_i 2^8$ i dobijena suma S(i) pomera udesno za jedno mesto. Na kraju se u osmoj iteraciji izračunava P(7) tako što se na trenutni proizvod P(6) dodaje $XY_7 2^8$ i dobijena suma S(7) pomera udesno za jedno mesto. Trenutni proizvod posle osme iteracije P(7) je i konačni proizvod $P_{15}P_{14}...P_1P_0$. Za izračunavanje suma S(0) do S(7) je potreban sabirač dužine 16 razreda.

Iz prvih nekoliko koraka realizacije operacije MULU (slika 7) može se uočiti da se pri formiranju $S(0)=0+XY_0 2^8$, $S(1)= P(0)+XY_1 2^8$, $S(2)= P(1)+XY_1 2^8$ itd. kao nove cifre u nekoj iteraciji formiraju samo cifre 8 do 16, dok se cifre 7 do 0 nasleđuju od trenutnog proizvoda iz prethodne iteracije. To je posledica toga što se kod formiranja trenutnog proizvoda neke iteracije na trenutni proizvod prethodne iteracije dodaje vrednost X pomerena 8 mesta ulevo koja kao cifre 7 do 0 ima nule. Zbog toga je za njegovu realizaciju je dovoljan sabirač sa osam razreda, pri čemu cifru 7 iz P(0), cifre 7 i 6 iz P(1) itd. treba čuvati u nekom pomeračkom registru.

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$XY_0 2^8$		X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	0	0	0	0	0	0	0	0
$S(0)=0+XY_0 2^8$	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	0	0	0	0	0	0	0	0
$S(0)2^{-1}$		S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	0	0	0	0	0	0	0
$P(0)=S(0)2^{-1}$		P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8	P_7	0	0	0	0	0	0	0
$XY_0 2^8$		X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	0	0	0	0	0	0	0	0
$S(1)= P(0)+XY_1 2^8$	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	0	0	0	0	0	0	0
$S(1)2^{-1}$		S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	0	0	0	0	0	0
$P(1)=S(1)2^{-1}$		P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8	P_7	P_6	0	0	0	0	0	0
$XY_0 2^8$		X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	0	0	0	0	0	0	0	0
$S(2)= P(1)+XY_1 2^8$	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	S_6	0	0	0	0	0	0
$S(2)2^{-1}$		S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	S_6	0	0	0	0	0
$P(2)=S(2)2^{-1}$		P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8	P_7	P_6	P_5	0	0	0	0	0

...

Slika 7 Operacija MULU

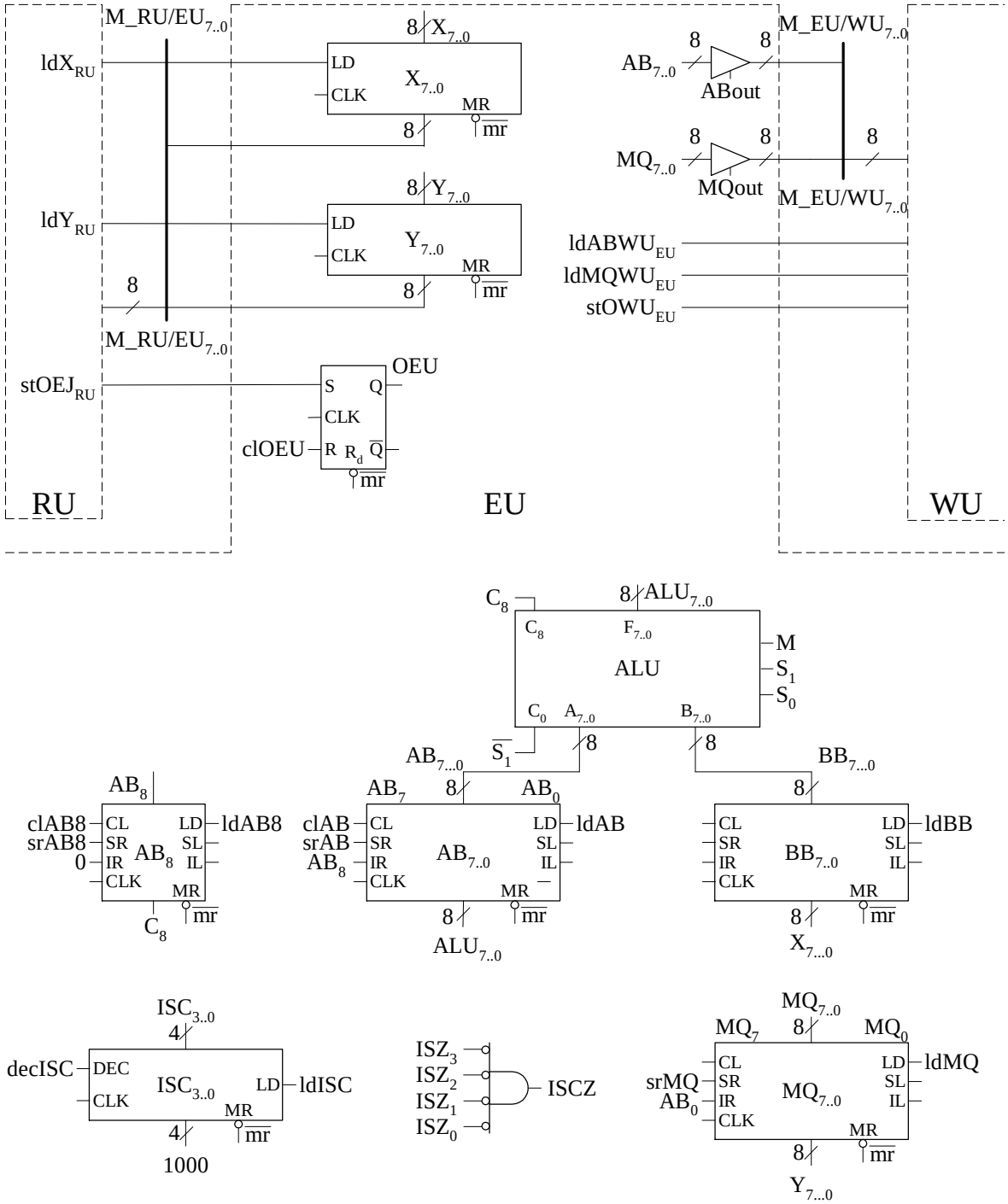
Neka razmatranja vezana se realizaciju operacije MULU data su u odeljku 1.4.1.

1.2.1.2 Strukturna šema

Strukturne šeme operacionih jedinica se razlikuju po tome da li se za povezivanje prekidačkih mreža unutar operacione jedinice koriste direktne veze ili interne magistrale.

1.2.1.2.1 Direktne veze

Moguća strukturna šema operacione jedinice sa direktnim vezama za jedinicu sa jednom operacijiom koja se razmatra je data na slici 8.



Slika 8 Operaciona jedinica sa direktnim vezama za jedinicu sa jednom operacijiom

Operaciona jedinica je realizovana u skladu sa usvojenom strukturom digitalnih uređaja kojom se podrazumeva da je jedinica koja se razmatra deo nekog uređaja koji se sastoji iz više jedinica (poglavlje 1.1.4). Stoga jedinica koja se razmatra najpre dobija binarne vrednosti nad kojima treba da realizuje određenu operaciju od jedinice koja je njen prethodnik, zatim u skladu sa usvojenin algoritmom izvršava određenu operaciju i na kraju rezultat izvršene operacije predaje jedinici koja je njen sledbenik. Jedinica koja se razmatra se dalje naziva

jedinica EU (Execution Unit), jedinica koja je njen prethodnik jedinica RU (Read Unit) i jedinica koja je njem sledbenik jedinica WU (Write Unit). U određenom trenutku samo jedna od ove tri jedinice može da bude aktivna. Stoga u jedinicama RU, EU i WU postoje flip-flopovi ORU (Operational Read Unit), OEU (Operational Execution Unit) i OWU (Operational Write Unit) koji vrednostima 1 i 0 određuju da li je određena jedinica aktivna ili neaktivna, respektivno.

Jedinica RU i EU su povezane linijama magistrale $M\_RU/EU_{7..0}$ dužine 8 bitova. Početne vrednosti X i Y u dva posebna ciklusa jedinica RU šalje po linijama magistrale $M\_RU/EU_{7..0}$ i u registre $X_{7..0}$ i $Y_{7..0}$ jedinice EU upisuje vrednostima 1 upravljačkih signala ldX_{RU} i ldY, respektivno. Izvršavanje operacije u jedinici EU se startuje kada jedinica RU vrednošću 1 upravljačkog signala signala $stOEU_{RU}$ u flip-flop OEU jedinice EU upiše vrednost 1.

Za realizaciju aritmetičkih i logičkih mikrooperacija koristi se aritmetičko logička jedinica ALU. Jedinica ALU mikrooperacije realizuje nad sadržajima sa ulaza $A_{7..0}$ i $B_{7..0}$, a mikrooperacije se biraju upravljačkim signalima M, S_1 i S_0 (slika 9). Na ulaze $A_{7..0}$ i $B_{7..0}$ dolaze sadržaji registara $AB_{7..0}$ i $BB_{7..0}$, respektivno.

M	S_1	S_0	operacija
0	0	0	$A + C_0$
0	0	1	$A - B - \overline{C_0}$
0	1	0	$A + B + C_0$
0	1	1	$A - \overline{C_0}$
1	0	0	$A \cdot B$
1	0	1	$A \oplus B$
1	1	0	$A + B$
1	1	1	$\overline{A}$

Slika 9 Mikrooperacije u jedinici ALU

Aritmetičke mikrooperacije imaju dva značenja u zavisnosti od vrednosti upravljačkog signala C_0. Tako na primer u slučaju prve vrste na izlazima ALU će biti sadržaj sa ulaza A ukoliko je C_0 nula i sadržaj sa ulaz A uvećan za jedan ukoliko je C_0 jedan. U slučaju druge vrste na izlazima ALU će biti razlika sadržaja sa ulaza A i B ukoliko je C_0 jedan i razlika sadržaja sa ulaza A i B umanjena za jedan ukoliko je C_0 nula. U slučaju treće vrste na izlazima ALU će biti suma sadržaja sa ulaza A i B ukoliko je C_0 nula i suma sadržaja sa ulaza A i B uvećana za jedan ukoliko je C_0 jedan. U slućaju četvrte vrste na izlazima ALU će biti sadržaj sa ulaza A ukoliko je C_0 jedan i sadržaj sa ulaz A umanjen za jedan ukoliko je C_0 nula. Ovo je u saglasnosti sa ranijim razmatranjima realizacije inkrementiranja, oduzimanja, sabiranja i dekrementiranja korišćenjem sabirača.

Operacija množenja dve 8 bitne celobrojne vrednosti bez znaka. se realizuje u 8 iteracija tako što se u svakoj iteraciji izvršavanjem mikrooperacija sabiranja i pomeranja udesno izračunava devet starijih cifara trenutnog proizvoda date iteracije i jedna za drugom osam najmlađih cifara konačnog proizvoda. Devet starijih cifara trenutnog proizvoda se formira u registrima AB_8 i $AB_{7..0}$, a osam najmlađih cifara konačnog proizvoda u registru $MQ_{7..0}$. Vrednostima 1 signala clAB8 i clAB registri AB_8 i $AB_{7..0}$ se dovode na početne vrednosti 0. Vrednost $X_{7..0}$ koju treba u svakoj iteraciji dodavati na trenutni proizvod u registru $AB_{7..0}$ i time dobiti novu vrednost trenutnog proizvoda u registrima AB_8 i $AB_{7..0}$, treba da bude u registru $BB_{7..0}$. Zbog toga se sadržaj registra $X_{7..0}$ vodi na ulaze registra $BB_{7..0}$, a prenos C_8 i rezultat $ALU_{7..0}$ se sa izlaza C_8 i $F_{7..0}$ vode u registare AB_8 i $AB_{7..0}$. U svako iteraciji se

proveravaju redom cifre Y_0 do Y_7 i u zavisnosti od toga da li je vrednost cifre 1 ili 0, vrednost $X_{7..0}$ se prilikom sračunavanja trenutnog proizvoda date iteracije ili dodaje ili ne dodaje na vrednost trenutnog proizvoda prethodne iteracije. Cifre Y_0 do Y_7 se redom u svakoj iteraciji pojavlju u razredu MQ_0. Zbog toga se sadržaj registra $Y_{7..0}$ vodi na ulaze registra $MQ_{7..0}$. Registri AB_8, $AB_{7..0}$ i $MQ_{7..0}$ se u svakoj iteraciji pomeraju jedno mesto udesno. U registar AB_8 se upisuje 0, u registar $AB_{7..0}$ osam najstarijih cifara trenutnog proizvoda koje učestvuju u formiranju trenutnog proizvoda sledeće iteracije, u razred MQ_7 se upisuje cifra trenutnog proizvoda koja ne učestvuje u formiranju cifara trenutnog proizvoda u sledećim iteracijama, a mlađe cifre trenutnog proizvoda i preostale cifre $Y_{7..0}$ koje se nalaze u $MQ_{7..0}$ pomeraju se jedno mesto udesno. Pomeranjem registra $MQ_{7..0}$ za jedno mesto udesno u registar $MQ_{7..0}$ se u osam iteracija ubacuju jedna za drugom osam najmlađih cifara proizvoda, a u razredu MQ_0 pojavljuju redom cifre Y_0 do Y_7. Posle osam iteracija u registru $AB_{7..0}$ se nalazi osam starijih cifara konačnog proizvoda, a u registru $MQ_{7..0}$ osam mlađih cifara konačnog proizvoda.

Algoritam operacije moženja zasniva se na ponavljanju koraka sabiranja i pomeranja udesno u 8 iteracija. Za brojanje ponavljanja koraka koristi se brojač ciklusa $ISC_{3..0}$. U brojač $ISC_{3..0}$ se signalom ldISC na početku paralelno upisuje vrednost 8. Sadržaj brojača $ISC_{3..0}$ se signalom decISC dekrementira u svakoj iteraciji. Signal ISCZ na izlazu I elementa ima vrednost 0 ukoliko sadržaj brojača $ISC_{3..0}$ nije nula i vrednost 1 ukoliko je sadržaj brojača $ISC_{3..0}$ nula. Signal ISCZ je signal logičkog uslova koji koristi upravljačka jedinica.

Rezultat izvršene operacije množenja koji se nalazi u registrima $AB_{7..0}$ i $MQ_{7..0}$ prenosi se u dva posebna ciklusa u registre $ABWU_{7..0}$ i $MQWU_{7..0}$ jedinice WU preko linija magistrale $M\_EU/WU_{7..0}$ dužine 8 bitova kojima su jedinice EU i WU povezane. Pri kraju svog rada jedinica EU generiše vrednost 1 signala ABout kojom otvara bafere sa tri stanja i sadržaj registra $AB_{7..0}$ pušta na linije magistrale $M\_EU/WU_{7..0}$ i vrednost 1 signala $ldABWU_{EU}$ kojim sadržaj sa magistrale $M\_EU/WU_{7..0}$ upisuje u registar $ABWU_{7..0}$ jedinice WU. Zatim se na identičan način vrednostima 1 signala MQout i $ldMQWU_{EU}$ sadržaj registra $MQ_{7..0}$ pušta na linije magistrale $M\_EU/WU_{7..0}$ i upisuje u registar $MQWU_{7..0}$ jedinice WU. Na kraju se vrednostima 1 signala clOEU i $stOWU_{EJ}$ u flip-flopove OEU i OWU upisuju vrednosti 0 i 1, respektivno, i time zaustavlja jedinica OEU i startuje jedinica OWU.

U odnosu na opštu strukturu operacione jedinice, razmatrana operaciona jedinica ima prekidačke mreže za

- čuvanje binarnih reči (registri $X_{7..0}$, $Y_{7..0}$, $AB_{7..0}$, $MQ_{7..0}$ itd.),
- realizaciju aritmetičkih, logičkih i pomeračkih mikrooperacija (jedinica ALU, registri AB_8, $AB_{7..0}$, $MQ_{7..0}$ sa mogućnošću pomeranja udesno itd.) i
- generisanje signala logičkih uslova OEU, MQ_0 i ISCZ (flip-flop OEU, registrar $MQ_{7..0}$, brojač $ISC_{3..0}$ itd.).

Iz operacione jedinice u upravljačku jedinicu dolaze signale logičkih uslova OEU, MQ_0 i ISCZ, dok iz upravljačke jedinice u operacionu jedinicu dolaze upravljački signali ldMQ, srAB itd.

U odnosu na opštu strukturu uređaja po kojoj se uređaj sastoji iz više jedinica koje su međusobno povezane, jedinica EU koja se razmatra je linijama magistrale $M\_RU/EU_{7..0}$ dužine 8 bitova povezana sa jedinicom RU koja je prethodnik i linijama magistrale $M\_EU/WU_{7..0}$ dužine 8 bitova sa jedinicom WU koja je sledbenik. Takođe po opštoj strukturi jedinica EU razmenjuje signale sa jedinicom RU i jedinicom WU. Iz jedinice RU u jedinicu EU po linijama magistrale $M\_RU/EU_{7..0}$ dolaze sadržaji koji se upisuju u registre $X_{7..0}$ i $Y_{7..0}$. Iz jedinice RU dolaze i upravljački signali ldX_{RU} i ldY_{RU} kojima se sadržaj sa linija magistrale $M\_RU/EU_{7..0}$ upisuje u registre $X_{7..0}$ i $Y_{7..0}$, respektivno, i upravljački signal $stOEU_{RU}$ kojim

se upisuje vrednost 1 u flip-flop OEU. Iz jedinice EU se u jedinicu WU po linijama magistrale $M\_EU/WU_{7..0}$ šalju sadržaji registara $AB_{7..0}$ i $MQ_{7..0}$ koji se upisuju u registre $ABWU_{7..0}$ i $MQWU_{7..0}$ jedinice WU. Iz jedinice EU se šalju i upravljački signali $ldABWU_{RU}$ i $ldMQWU_{RU}$ kojima se sadržaj sa linija magistrale $M\_EU/WU_{7..0}$ upisuje u registre $ABWU_{7..0}$ i $MQWU_{7..0}$ jedinice WU, respektivno, i upravljački signal $stOWU_{EU}$ kojim se upisuje vrednost 1 u flip-flop OWU jedinice WU.

1.2.1.2.2 Interne magistrale

Za povezivanje prekidačkih mreža unutar jedinice mogu da se koriste i interne magistrale i to jedna, dve ili tri interne magistrale.

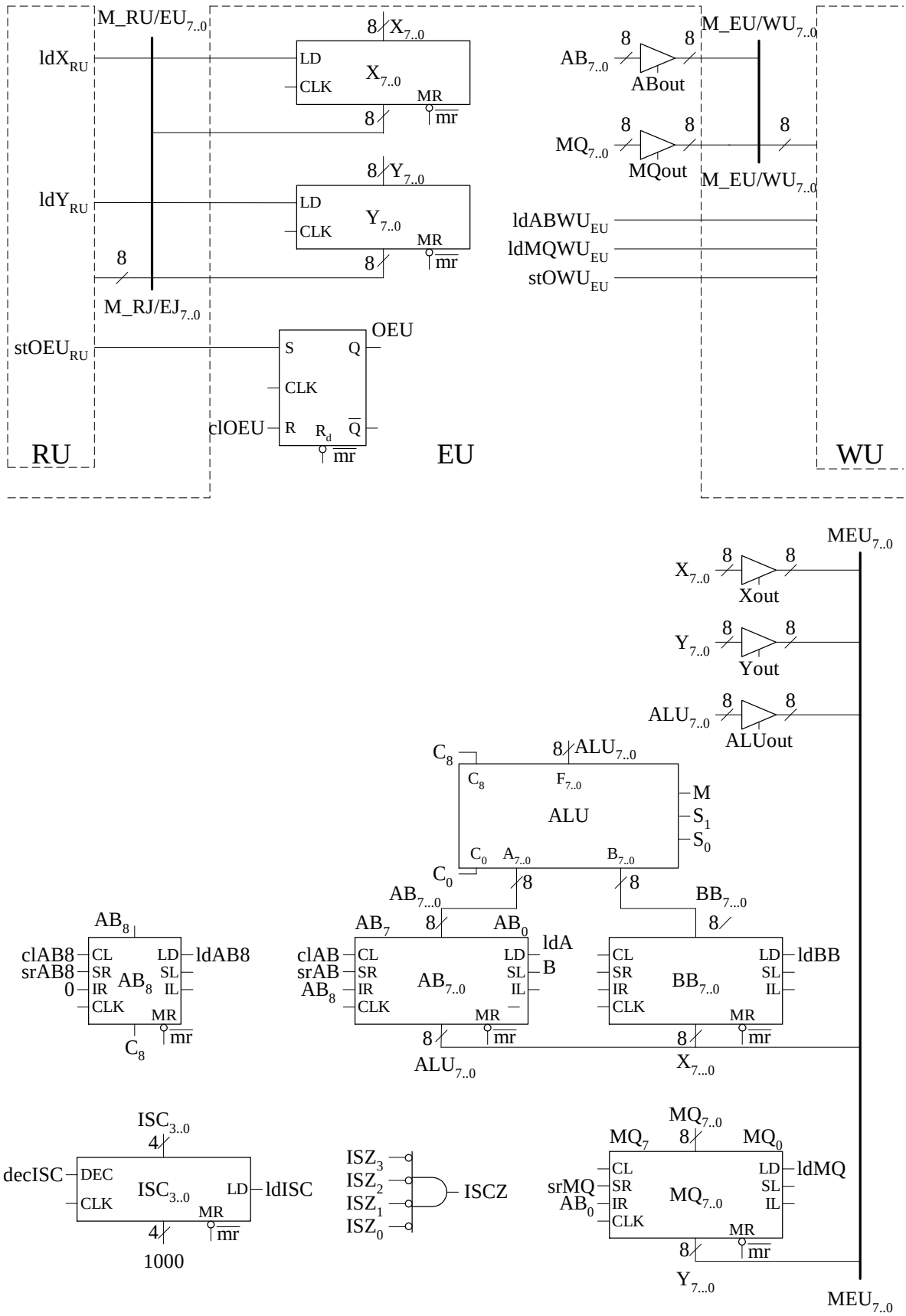
Moguća strukturna šema operacione jedinice sa jednom internom magistralom za jedinicu sa jednom operacijiom je data na slici 10

Izlazi registara $X_{7..0}$ i $Y_{7..0}$ i aritmetičko logičke jedinice $ALU_{7..0}$ povezani su preko bafera sa tri stanje na linije interne magistrala $MEU_{7..0}$. Sadržaj sa linija interne magistrala $MEU_{7..0}$. se vodi na paralelne ulaze registara $BB_{7..0}$, $MQ_{7..0}$ i $AB_{7..0}$. Upisivanje sadržaja registra $X_{7..0}$ u registar $BB_{7..0}$ se realizuje generisanjem vrednosti 1 signala Xout, kojim se sadržaj registra $X_{7..0}$ pušta na linije interne magistrale $MEU_{7..0}$, i signala ldBB, kojim se sadržaj sa linija interne magistrale $MEU_{7..0}$ upisuje u registar $BB_{7..0}$. Na sličan način se generisanjem vrednosti 1 signala Yout i ldMQ, realizuje upisivanje sadržaja registra $Y_{7..0}$ u registar $MQ_{7..0}$ i generisanjem vrednosti 1 signala ALUout i ldAB, realizuje upisivanje sadržaja sa izlaza $ALU_{7..0}$ u registar $AB_{7..0}$.

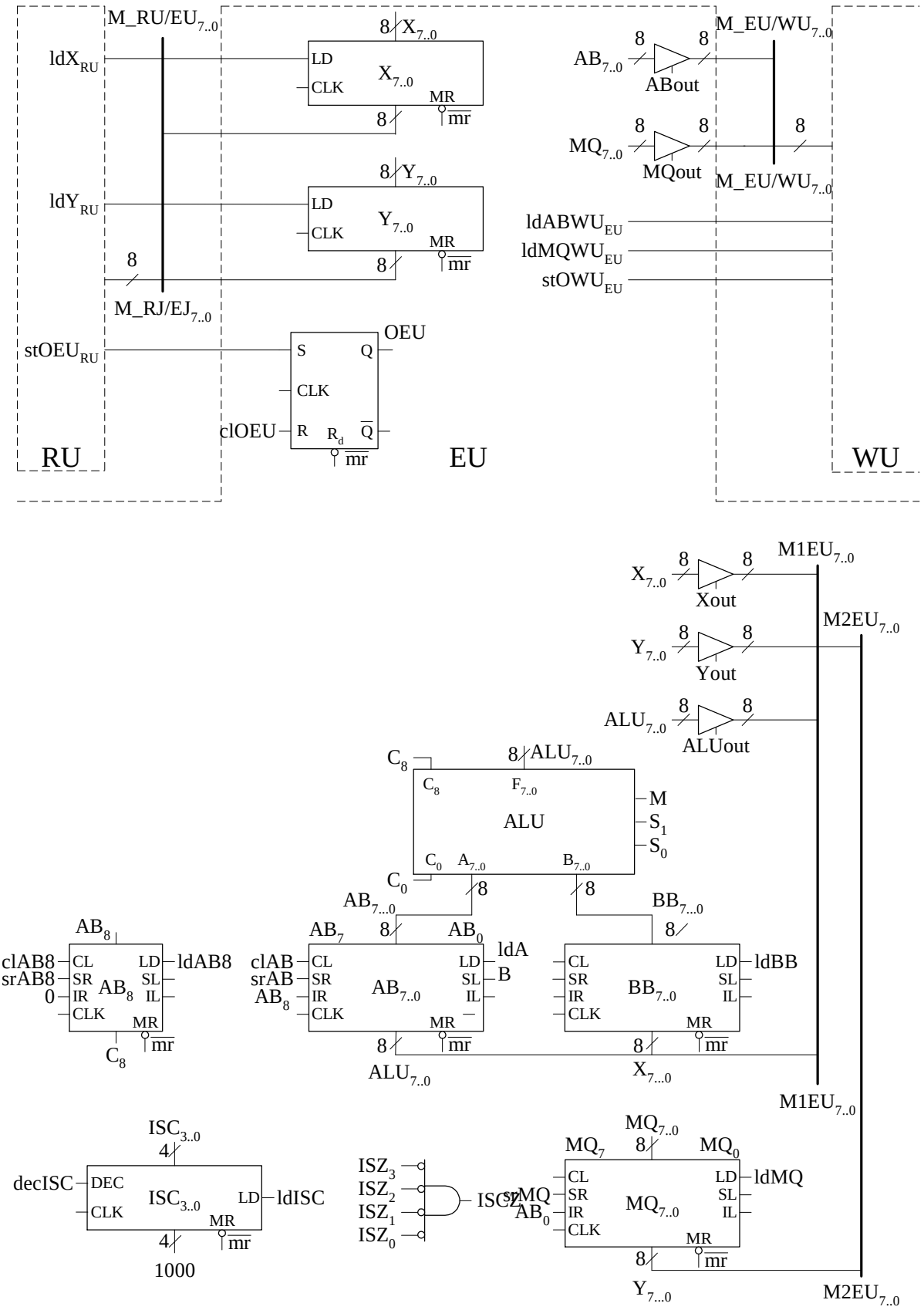
Treba uočiti da je strukturna šema operacione jedinice jednostavna i da povezivanje prekidačkih mreža internom magistralom nema nikakve prednosti u odnosu na povezivanje direktnim vezama. Strukturna šema sa internom magistralom je čak i složenija nego strukturna šema sa direktnim vezama, jer strukturna šema sa internom magistralom zaheva bafere sa tri stanja kojih nema u slučaju strukturne šeme sa direktnim vezama. Pored toga u slučaju strukturne šeme sa internom magistralom upisivanje sadržaja registra $X_{7..0}$ u registar $BB_{7..0}$ i registra $Y_{7..0}$ u registar $MQ_{7..0}$ se realizuje u dve posebne periode signala takta, dok je u slučaju strukturne šeme sa direktnim vezama to moguće uraditi u jednoj periodi signala takta.

Međutim u slučaju složenijih operacionih jedinica strukturna šema sa internom magistralom je jednostavnija od strukturne šeme sa direktnim vezama, što će biti ilustrovano prilikom razmatranja mogućih realizacija operacione jedinice za jedinicu sa više operacija. Nedostatak strukturnih šema operacionih jedinica sa jednom internom magistralom da je za vreme jedne periode signala takta moguće samo jedan prenos sadržaja između prekidačkih mreža se može ublažiti realizacijom strukturne šeme operacione jedinice sa dve interne magistrale.

Moguća strukturna šema operacione jedinice sa dve interne magistrale za jedinicu sa jednom operacijiom je data na slici 11.



Slika 10 Operaciona jedinica sa jednom internom magistralom
za jedinicu sa jednom operacijom



Slika 11 Operaciona jedinica sa dve interne magistrale za jedinicu sa jednom operacijom

Izlazi registra $X_{7..0}$ i aritmetičko logičke jedinice $ALU_{7..0}$ povezani su preko bafera sa tri stanje na linije interne magistrala $M1EU_{7..0}$. Sadržaj sa linija interne magistrala $M1EU_{7..0}$ se vodi na paralelne ulaze registara $BB_{7..0}$ i $AB_{7..0}$. Upisivanje sadržaja registra $X_{7..0}$ u registar

$BB_{7..0}$ se realizuje generisanjem vrednosti 1 signala Xout, kojim se sadržaj registra $X_{7..0}$ pušta na linije interne magistrale $M1EU_{7..0}$, i signala ldBB, kojim se sadržaj sa linija interne magistrale $MEU_{7..0}$ upisuje u registar $BB_{7..0}$. Na sličan način se generisanjem vrednosti 1 signala ALUout i ldAB, realizuje upisivanje sadržaja sa izlaza $ALU_{7..0}$ u registar $AB_{7..0}$. Izlaz regist $Y_{7..0}$ povezan je preko bafera sa tri stanje na linije interne magistrala $M2EU_{7..0}$. Sadržaj sa linija interne magistrala $M2EU_{7..0}$ se vodi na paralelne ulaze registra $MQ_{7..0}$. Upisivanje sadržaja registra $Y_{7..0}$ u registar $MQ_{7..0}$ se realizuje generisanjem vrednosti 1 signala Yout, kojim se sadržaj registra $Y_{7..0}$ pušta na linije interne magistrale $M2EU_{7..0}$, i signala ldMQ, kojim se sadržaj sa linija interne magistrale $M2EU_{7..0}$ upisuje u registar $MQ_{7..0}$.

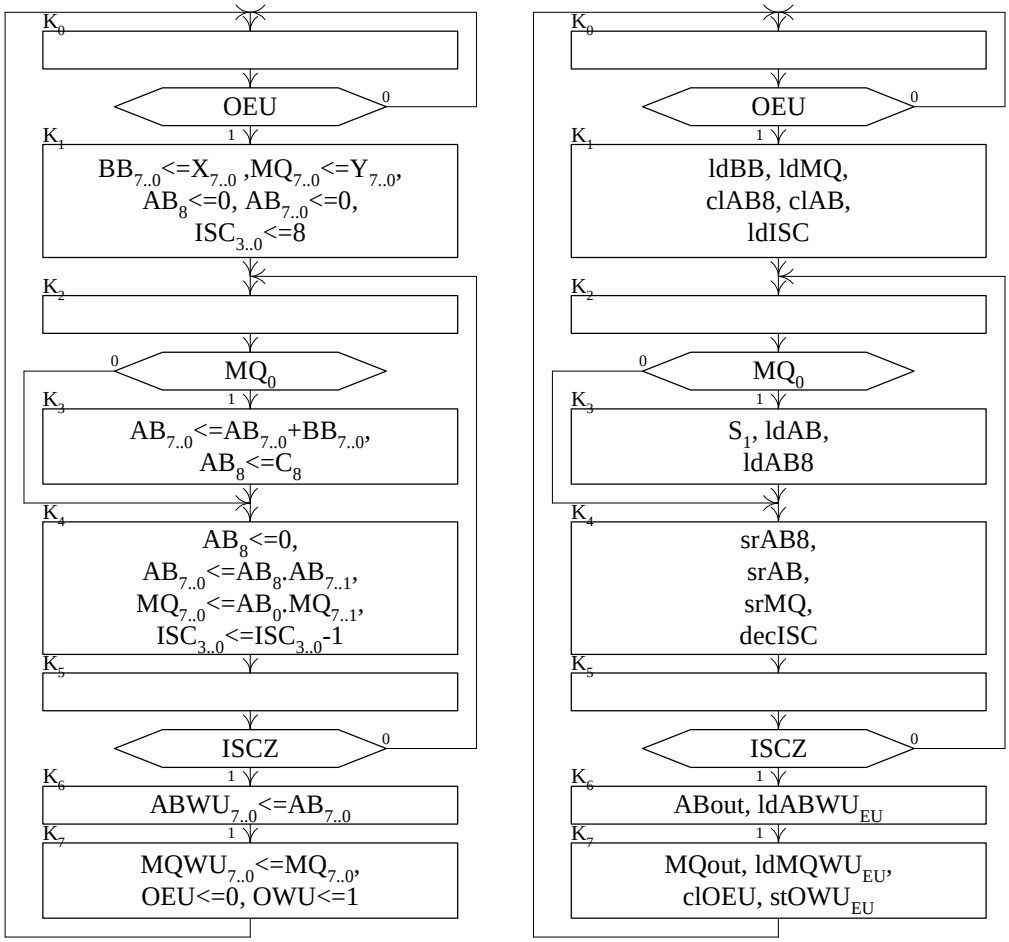
Treba i ovde uočiti da zbog toga što je strukturna šema operacione jedinice jednostavna i da povezivanje prekidačkih mreža sa dve interne magistrale nema nikakve prednosti u odnosu na povezivanje direktnim vezama. Strukturna šema sa dve interne magistrale je čak i složenija nego strukturna šema sa direktnim vezama, jer strukturna šema sa dve interne magistrale zaheva bafere sa tri stanja kojih nema u slučaju strukturne šeme sa direktnim vezama. Strukturna šema sa dve interne magistrale je nešto složenija nego strukturna šema sa jednom magistralom, jer pored linija jedne magistrale postoje i linije druge magistrale. Međutim, u slučaju strukturne šeme sa dve interne magistrale upisivanje sadržaja registra $X_{7..0}$ u registar $BB_{7..0}$ i registra $Y_{7..0}$ u registar $MQ_{7..0}$ se realizuje kao i u slučaju strukturne šeme sa direktnim vezama za vreme jedne periode signala takta, jer se upisivanje sadržaja registra $X_{7..0}$ u registar $BB_{7..0}$ i registra $Y_{7..0}$ u registar $MQ_{7..0}$ se realizuje preko posebnih magistrala $M1EU_{7..0}$ i $M2EU_{7..0}$, respektivno.

U nekim situacijama, koje se u slučaju jedinice za realizaciju operacije množenja celobrojnih vrednosti bez znaka ne javljaju, je poželjno da postoji mogućnost da se za vreme iste periode signala takta realizuje prebacivanje sadržaja u tri registra. To je moguće ostvariti ukoliko postoji i treća interna magistrala.

1.2.1.3 Algoritam generisanja upravljačkih signala

Algoritmi generisanja upravljačkih signala se razlikuju u zavisnosti od toga da li se za povezivanje prekidačkih mreža unutar operacione jedinice koriste direktne veze ili interne magistrale. U ovom odeljku se razmatra algoritam generisanja upravljačkih signala za jedinicu sa jednom operacijom u kojoj se za povezivanje prekidačkih mreža unutar operacione jedinice koriste direktne veze. Algoritam generisanja upravljačkih signala je formiran na osnovu usvojenog algoritama operacije (odeljak 1.2.1.1) i dat je u obliku dijagrama toka mikrooperacija i dijagrama toka upravljačkih signala operacione jedinice (slika 12) i sekvence upravljačkih signala po koracima (tabela 1).

Dijagram toka mikrooperacija i dijagram toka upravljačkih signala operacione jedinice su dati istovremeno i predstavljeni su operacionim i upravljačkim blokovima. U operacionim blokovima dijagrama toka mikrooperacija se nalaze mikrooperacije koje treba da se izvršavaju da bi se odgovarajuća operacija realizovala. Svakom operacionom bloku u dijagramu toka mikrooperacija odgovara operacioni blok u dijagramu toka upravljačkih signala operacione jedinice. U svakom od operacionih blokova dijagrama toka upravljačkih signala operacione jedinice se nalaze upravljački signali operacione jedinice koji treba da imaju vrednost 1 da bi se mikrooperacije iz odgovarajućeg operacionog bloka dijagrama toka mikrooperacija realizovale. U upravljačkim blokovima dijagrama toka mikrooperacija i dijagrama toka upravljačkih signala operacione jedinice se nalaze signali logičkih uslova od čijih vrednosti zavisi redosled generisanja vrednosti 1 upravljačkih signala operacione jedinice, a time i redosled izvršavanja mikrooperacija.



Slika 12 Dijagrami toka mikrooperacija i upravljačkih signala
za jedinicu sa jednom operacijom i operacionu jedinicu sa direktnim vezama

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

signali.

Ovaj iskaz sadrži spisak upravljačkih signala operacione jedinice i određuje koji signali treba da imaju vrednost 1. Iskazi za skokove su oblika

br $step_A$ i

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

Prvi iskaz sadrži korak $step_A$ na koji treba bezuslovno preći i u daljem tekstu se referiše kao bezuslovni skok. Drugi iskaz sadrži signal **uslov** i korak $step_A$ i određuje korak $step_A$ na koji treba preći ukoliko signal **uslov** ima vrednost 1 i u daljem tekstu se referiše kao uslovni skok. Objašnjenja vezana za generisanje upravljačkih signala su data zajednički za dijagram toka upravljačkih signala i sekvencu upravljačkih signala po koracima i to u okviru sekvence upravljačkih signala po koracima.

Tabela 1 Sekvenca upravljačkih signala po koracima za jedinicu sa jednom operacijom

! U koraku $step_{00}$ se vrši provera vrednosti signala OEU pa ukoliko signal OEU ima vrednosti 0, što znači da izvršavanje operacije nije startovano, ostaje se u koraku $step_{00}$. U suprotnom slučaju se prelazi na korak $step_{01}$, čime započinje izvršavanje operacije. !

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

! U koraku $step_{01}$ se sadržaji registara $BB_{7..0}$, $MQ_{7..0}$, AB_8 i $AB_{7..0}$ i brojača $ISC_{3..0}$ dovode u početno stanje tako što se vrednošću 1 signala **mxBB** i **ldBB** sadržaj registra $X_{7..0}$ upisuje u

registar $BB_{7..0}$, vrednošću 1 signala **ldMQ** sadržaj registra $Y_{7..0}$ upisuje u registar $MQ_{7..0}$, vrednošću 1 signala **clAB8** i **clAB** brišu sadržaji registara AB_8 i $AB_{7..0}$ i vrednošću 1 signala ldISC u brojač $ISC_{3..0}$ upisuje vrednost 8. !

$step_{01}$ **ldBB, ldMQ, clAB8, clAB, ldISC**;

! Koraci $step_{02}$ do $step_{05}$ se ponavljaju u 8 iteracija. U svakoj iteraciji u koraku $step_{02}$ se proverava vrednost razreda MQ_0 u kome se u iteracijama pojavljuju redom cifre Y_0 do Y_7. Ukoliko signal MQ_0 ima vrednost 1, iz koraka $step_{02}$ se prelazi se na korak $step_{03}$, dok se u suprotnom slučaju prelazi na korak $step_{04}$. U koraku $step_{03}$ se vrednošću 1 signala $\mathbf{S_1}$, ldAB i ldAB8 sabiraju trenutni proizvod iz registra $AB_{7..0}$ i vrednost X iz registra $BB_{7..0}$ i prenos i suma upisuju u registre AB_8 i $AB_{7..0}$, respektivno. U koraku $step_{04}$ se vrednošću 1 signala **srAB8**, srAB, srMQ sadržaji registara AB_8, $AB_{7..0}$ i $MQ_{7..0}$ pomeraju udesno za jedno mesto, dok se vrednošću 1 signala decISC dekrementira sadržaj brojača $ISC_{3..0}$. U koraku $step_{05}$ se proverava vrednost signala ISCZ. Vrednost 0 signala ISCZ je indikacija da brojač $ISC_{3..0}$ dekrementiranjem nije stigao do nule i da treba preći na korak $step_{02}$, dok u suprotnom slučaju treba preći na korak $step_{06}$. Pomeranjem sadržaja registara AB_8 i $AB_{7..0}$ za jedno mesto udesno u svakoj od 8 iteracija, u registre AB_8 i $AB_{7..0}$ se upisuje vrednost 0 i 8 starijih bitova trenutnog proizvoda date iteracije. Pomeranjem sadržaja registra $MQ_{7..0}$ za jedno mesto udesno u svakoj od 8 iteracija u registar $MQ_{7..0}$ se ubacuju bit po bit i to počev od najmlađeg bita 8 najmlađih bitova proizvoda i iz registra MQ izbacuju bit po bit počev od najmlađeg bita bitovi Y_0 do Y_7 vrednosti Y. Posle 8 ponavljanja uslovnog sabiranja 8 starijih bitova trenutnog proizvoda iz registra $AB_{7..0}$ i vrednosti X iz registra $AB_{7..0}$ i obaveznog pomeranja za jedno mesto udesno sadržaja registara AB_8, $AB_{7..0}$ i $MQ_{7..0}$ u registrima $AB_{7..0}$ i $MQ_{7..0}$ se nalazi 8 starijih i 8 mlađih bitova proizvoda, respektivno. !

$step_{02}$ *br* (*if* $\overline{\mathbf{MQ_0}}$ *then* $step_{04}$);

$step_{03}$ $\mathbf{S_1}$, ldAB, ldAB8;

$step_{04}$ **srAB8**, srAB, srMQ, decISC;

$step_{05}$ *br* (*if* $\overline{\mathbf{ISCZ}}$ *then* $step_{20}$);

! U koracima $step_{06}$ i $step_{07}$ se 8 starijih i 8 mlađih bitova proizvoda koji se nalazi u registrima $AB_{7..0}$ i $MQ_{7..0}$, respektivno, prebacuju iz jedinice EU u jedinicu WU. Vrednošću 1 signala ABout i $ldABWU_{EU}$ se sadržaj registra $AB_{7..0}$ pušta preko bafera sa tri stanja na linije magistrale $M\_EU\_WU_{7..0}$ i upisuje u registar $ABWU_{7..0}$ jedinice WU, dok se vrednošću 1 signala MQout i $ldMQWU_{EU}$ sadržaj registra $MQ_{7..0}$ pušta preko bafera sa tri stanja na linije magistrale $M\_EU\_WU_{7..0}$ i upisuje u registar $MQWU_{7..0}$ jedinice WU. U koraku $step_{07}$ se vrednošću 1 signala clOEU i $stOWU_{EU}$ upisuju vrednosti 0 i 1 u flip-flop OEU jedinice EU i flip-flop OWU jedinice WU, čime se jedinica EU zaustavlja i jedinica WU startuje. Pored toga u koraku se prelazi na početni korak $step_{00}$. !

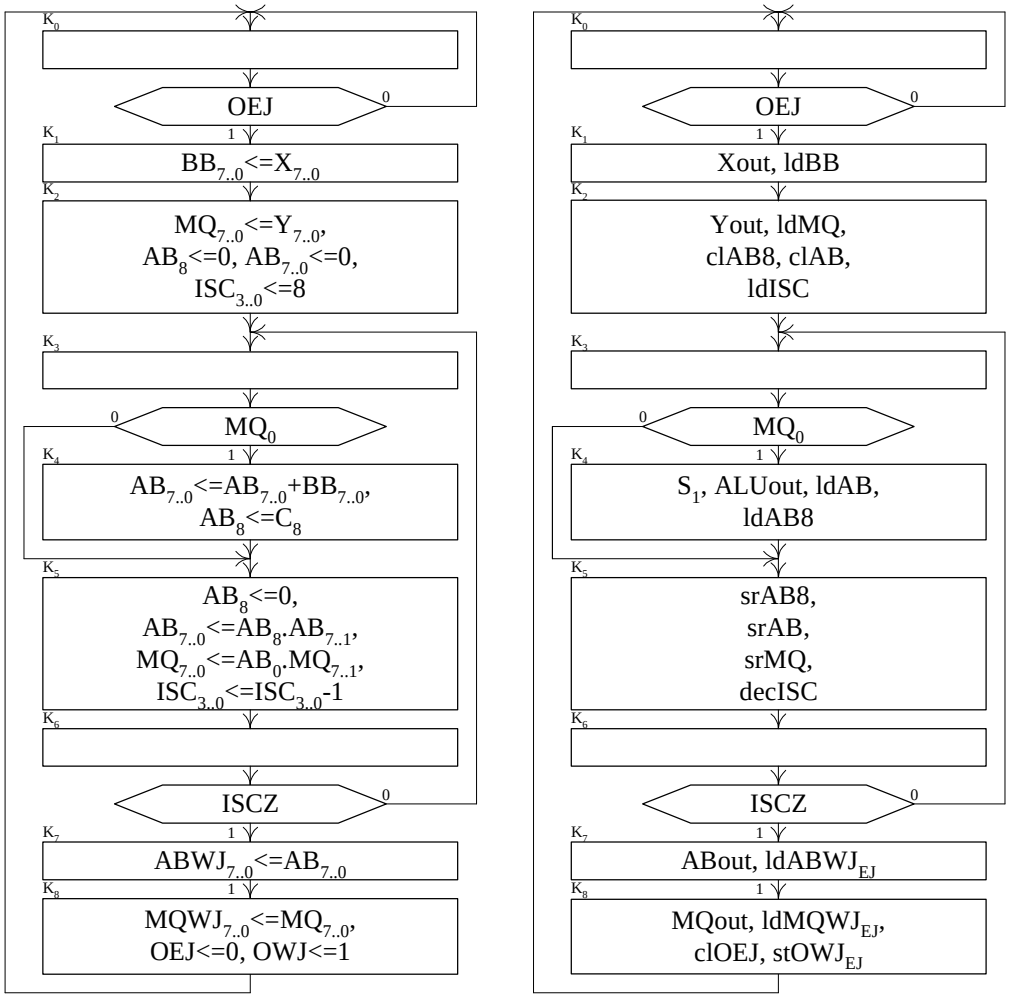
$step_{06}$ ABout, $ldABWU_{EU}$;

$step_{07}$ MQout, $ldMQWU_{EU}$, clOEU, $stOWU_{EU}$,
br $step_{00}$;

U dijagramu toka se pojavljuju dva prazna operaciona bloka i to prvi ispred upravljačkog bloka u kome se proverava signal logičkog uslova MQ_0 i drugi ispred upravljačkog bloka u kome se proverava signal logičkog uslova ISCZ. To je zbog toga što se na isti signal takta ne može i menjati stanje neke sekvencijalne mreže i proveravati signal logičkog uslova formiran na osnovu promenjenog stanja mreže radi prelaska u dijagramu toka i sekvenci upravljačkih signala po koracima. Zato se na određeni signal takta u jednom operacionom bloku upisuje sadržaj registra $BB_{7..0}$ u registar $MQ_{7..0}$ i na sledeći signal takta u sledećem operacionom bloku koji je prazan proverava MQ_0 i realizuje odgovarajući prelaz u dijagramu toka. Isto važi i za dekrementiranje brojača $ISCZ_{3..0}$ i proveru signala logičkog uslova ISCZ.

Dijagrami toka mikrooperacija i upravljačkih signala za realizaciju operacije množenja celobrojnih veličina bez znaka po modifikovanom algoritmu u kojoj se za povezivanje

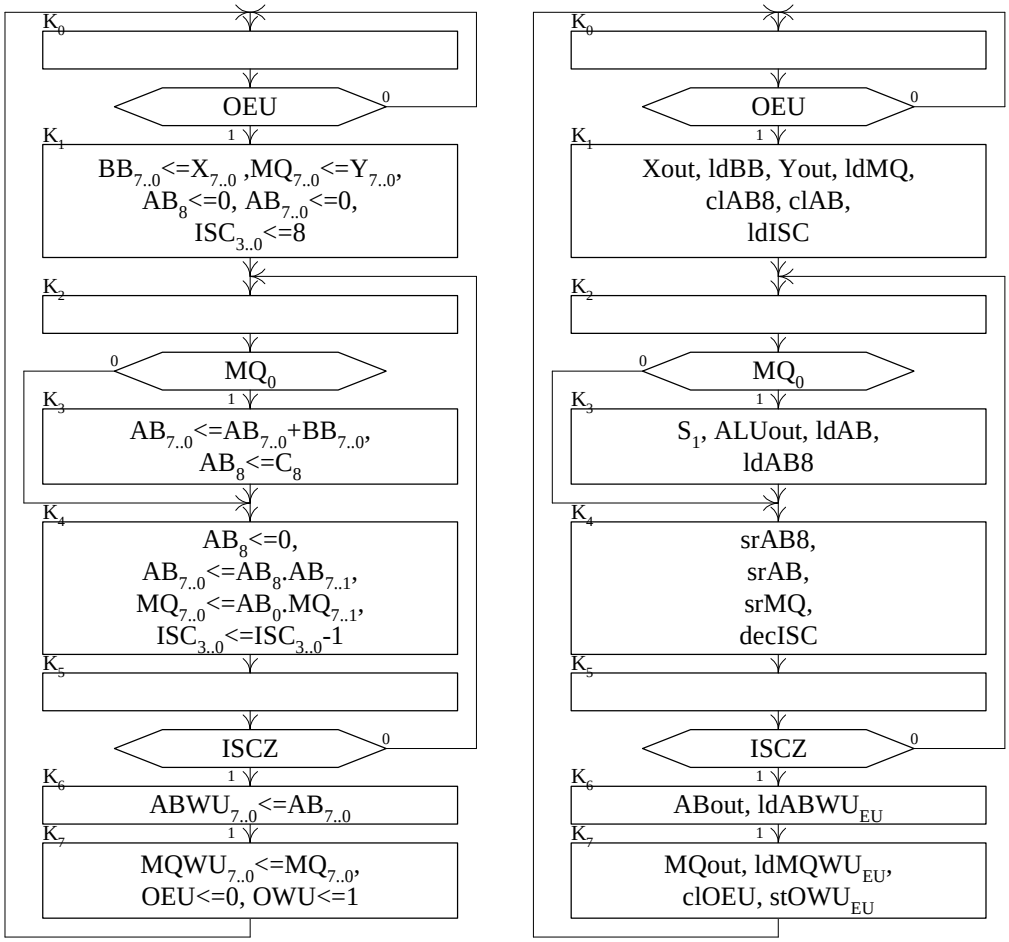
prekidačkih mreža unutar operacione jedinice koristi jedna interna magistrala je dat na slici 13.



Slika 13 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa jednom operacijom i operacionu jedinicu sa jednom internom magistralom

Ovaj dijagram toka je veoma sličan dijagramu toka kada se za povezivanje koriste direktne veze (slika 12) i razlike se javljaju samo zbog drugačijeg načina povezivanja prekidačkih mreža. U ovom slučaju sve ide preko interne magistrale, pa postoji potreba za generisanjem vrednosti 1 signala (Xout, Yout, ALUout) kojima se otvaraju baferi sa tri stanja da bi se sadržaj odgovarajuče prekidačke mreže ($X_{7..0}$, $Y_{7..0}$, $ALU_{7..0}$) puštao na linije interne magistrale. Pored toga u posebnim periodama signala takta vrši se upisivanje sadržaja registra $X_{7..0}$ u registar $BB_{7..0}$ i registra $Y_{7..0}$ u registar $MQ_{7..0}$. Zbog toga u dijagramu toka u slučaju povezivanja jednom internom magistralom postoji jedan operacioni blok više nego u slučaju povezivanja direktnim vezama.

Ukoliko bi postojale dve interne magistrale, upisivanje registra $X_{7..0}$ u registar $BB_{7..0}$ bi se realizovale preko jedne magistrale i prebacivanje registra $Y_{7..0}$ u registar $MQ_{7..0}$ preko druge magistrale, pa bi se operacioni blokovi K_1 i K_2 integrisali u jedan operacioni blok (slika 14). Tada bi broj operacionih blokova u slučaju povezivanja sa dve interne magistrale (slika 14) bio isti kao u slučaju povezivanja direktnim vezama (slika 12).



Slika 14 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa jednom operacijom i operacionu jedinici sa dve interne magistrale

1.2.2 UPRAVLJAČKA JEDINICA

Upravljačka jedinica se sastoji iz sekvencijalnih i kombinacionih prekidačkih mreža koje saglasno algoritmu operacije i vrednostima signala logičkih uslova koji dolaze iz operacione jedinice generišu upravljačke signale za operacionu jedinicu. Ovi signali određuju koje mikrooperacije i po kom redosledu treba izvršavati u operacionoj jedinici da bi se kao rezultat posle određenog broja signala takta realizovala operacija.

Osnovni deo svake upravljačke jedinice čini sekvencijalna prekidačke mreža koja ima onoliko stanja na koliko koraka je razloženo izvršavanje operacije. Svakom koraku u dijagramu toka odgovara posebno stanje sekvencijalne prekidačke mreže. Stanje sekvencijalne prekidačke mreže se koristi za generisanje dve grupe signala i to upravljačkih signala operacione jedinice koji omogućavaju izvršavanje mikrooperacija iz koraka dijagrama toka kome je dodeljeno dato stanje i upravljačkih signala upravljačke jedinice koji omogućavaju prelazak sekvencijalne mreže iz stanja u kome se nalazi u sledeće stanje saglasno dijagramu toka.

Postoje dve osnovne grupe realizacija upravljačkih jedinica i to sa ožičenim generisanjem upravljačkih signala i mikroprogramskim generisanjem upravljačkih signala, koje se razlikuju u načinu samog generisanja upravljačkih signala. U slučaju realizacija sa ožičenim generisanjem upravljačkih signala stanja sekvencijalne mreže se direktno koriste za generisanje upravljačkih signala. U slučaju realizacija sa mikroprogramskim generisanjem upravljačkih signala stanja sekvencijalne mreže se koriste kao adrese za čitanje kontrolnih reči

iz mikroprogramske memorije koja je sastavni deo upravljačke jedinice, dok se upravljački signali generišu na osnovu sadržaja kontrolnih reči.

U daljem tekstu se najpre razmatraju realizacije upravljačkih jedinica sa ožičenim generisanjem upravljačkih signala, a zatim realizacije upravljačkih jedinica sa mikroprogramskim generisanjem upravljačkih signala. U svim slučajevima se najpre daje opšti postupak realizacije, a zatim postupak realizacije za operaciju množenja dve celobrojne vrednosti bez znaka. Pri tome se koristi dijagram toka upravljačkih signala (slika 12) i sekvenca upravljačkih signala po koracima (slika 1) za operacionu jedinicu sa direktnim vezama (slika 8). Postupci realizacija upravljačkih jedinica za operacione jedinice sa jednom dve ili tri interne magistrale se ne daju jer su identični postupku za operacionu jedinicu sa direktnim vezama, pri čemu se koriste odgovarajući dijagrami toka upravljačkih signala i odgovarajuće sekvenca upravljačkih signala po koracima.

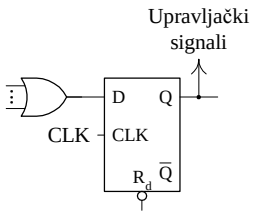
1.2.2.1 OŽIČENO GENERISNJE SIGNALA

Postoji više tipova upravljačke jedinice sa ožičenim generisnjem upravljačkih signala i to šetajuća jedinica, sekvencijalna mreža i brojač koraka.

1.2.2.1.1 ŠETAJUĆA JEDINICA

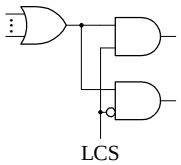
Postupak projektovanja strukturne šeme upravljačke jedinice tipa šetajuća jedinica zahteva da se svakom operacionom i upravljačkom bloku u dijagramu (slika 12) toka pridruži određeni tip prekidačke mreže.

Svakom operacionom bloku u dijagramu toka upravljačkih signala pridružuje se prekidačka mreža koju čine D flip-flop i u opštem slučaju ILI element (slika 15). Sa izlaza svakog od D flip-flopova uzimaju se upravljački signali naznačeni u operacionom bloku kojem je D flip-flop pridružen.



Slika 15 Prekidačka mreža pridružena operacionom bloku

Svakom upravljačkom bloku u dijagramu toka upravljačkih signala pridružuje se prekidačka mreža koju čine dva I elementa i u opštem slučaju ILI element (slika 16). Sa LCS je označen signal logičkog uslova koji se pojavljuje u upravljačkom bloku kojem je pridružena prekidačka mreža.



Slika 16 Prekidačka mreža pridružena upravljačkom bloku

Izlaz D flip-flopa pridruženog nekom operacionom bloku povezuje se ili

- na ulaz ILI elementa ili direktno na ulaz D flip-flopa pridruženih nekom operacionom bloku ili
- na ulaz ILI elementa ili direktno na ulaze I elemenata pridruženih nekom upravljačkom bloku

u zavisnosti od toga da li je u dijagramu toka izlaz operacionog bloka povezan na ulaz drugog operacionog bloka ili upravljačkog bloka.

Izlaz I elementa sa direktnom vrednošću signala LCS kombinacione mreže pridružene nekom upravljačkom bloku povezuje se ili

- na ulaz ILI elementa ili direktno na ulaz D flip-flopa pridruženih nekom operacionom bloku ili
- na ulaz ILI elementa ili direktno na ulaze I elemenata pridruženih nekom upravljačkom bloku

u zavisnosti od toga da li je u dijagramu toka izlaz upravljačkog bloka za slučaj kada je uslov ispunjen povezan na ulaz nekog drugog operacionog bloka ili upravljačkog bloka.

Izlaz I elementa sa komplementarnom vrednošću signala LCS kombinacione mreže pridružene nekom upravljačkom bloku povezuje se ili

- na ulaz ILI elementa ili direktno na ulaz D flip-flopa pridruženih nekom operacionom bloku ili
- na ulaz ILI elementa ili direktno na ulaze I elemenata pridruženih nekom upravljačkom bloku

u zavisnosti od toga da li je u dijagramu toka izlaz upravljačkog bloka za slučaj kada uslov nije ispunjen povezan na ulaz nekog drugog operacionog bloka ili upravljačkog bloka.

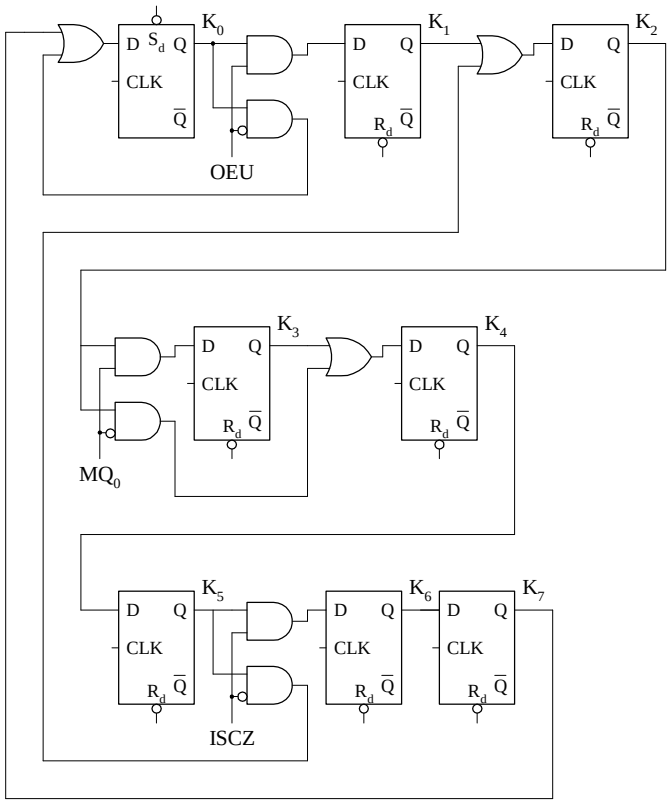
Ako je na ulaz nekog operacionog ili upravljačkog bloka vezan izlaz samo jednog operacionog ili upravljačkog bloka, ILI element ispred D flip-flopa ili I elemenata pridruženih datom operacionom ili upravljačkom bloku, respektivno, nije potreban. Međutim, ako su na ulaz nekog operacionog ili upravljačkog bloka vezani izlazi više od jednog bloka, bez obzira na to da li su to operacioni ili upravljački blokovi, ILI element ispred D flip-flopa ili I elemenata pridruženih datom operacionom ili upravljačkom bloku, respektivno, su potrebni.

U nekom trenutku jedinica se nalazi samo u jednom od flip-flopova. Ta jedinica u saglasnosti sa algoritmom operacije i vrednostima signala logičkih uslova šeta u ritmu signala takta iz flip-flopa u flip-flop. U nekom trenutku vrednost jedan imaju samo oni signali koji se uzimaju sa izlaza flip-flopa u kome je u datom trenutku vrednost jedan. Na početku jedinica se nalazi u flip-flopu koji je dodeljen početnom operacionom bloku.

U skladu sa iznetim postupkom projektovanja strukturne šeme upravljačke jedinice tipa šetajuća jedinica dolazi se do strukturne šeme upravljačke jedinice za operaciju množenja realizovane sa D flip-flopovima (slika 17).

U dijagramu toka (slika 12) postoji

- 8 operacionih blokova označenih sa K_0, K_1, K_2, K_3, K_4, K_5, K_6 i K_7, pa i u strukturnoj šemi upravljačke jedinice (slika 17) postoji 8 D flip-flopova označenih sa K_0, K_1, K_2, K_3, K_4, K_5, K_6 i K_7 pridruženih operacionim blokovima (slika 15) i
- 3 upravljačka bloka, pa u strukturnoj šemi upravljačke jedinice (slika 17) postoje tri kombinacione mreže pridružene upravljačkim blokovima (slika 16).



Slika 17 Upravljačka jedinica tipa šetajuća jedinica za jedinicu sa jednom operacijom

Ispred flip-flopa K_0 postoji ILI element sa dva ulaza jer se u operacioni blok K_0 kome je dodeljen flip-flop K_0 dolazi iz dva bloka i to upravljačkog bloka kome je dodeljena kombinaciona mreža sa signalom logičkog uslova OEU i operacionog bloka K_7 kome je dodeljen flip-flop K_7.

Ispred kombinacione mreže sa signalom logičkog uslova OEU ne postoji ILI element jer se u upravljački blok kome je dodeljena kombinaciona mreža sa signalom logičkog uslova OEU dolazi samo iz operacionog bloka K_0 kome je dodeljen flip-flop K_0.

Ispred flip-flopa K_1 ne postoji ILI element jer se u operacioni blok K_2 kome je dodeljen flip-flop K_2 dolazi samo iz upravljačkog bloka kome je dodeljena kombinaciona mreža sa signalom logičkog uslova OEU.

Ispred flip-flopa K_2 postoji ILI element jer se u operacioni blok kome je dodeljen flip-flop K_2 dolazi iz dva bloka i to operacionog bloka K_1 kome je dodeljen flip-flop K_1 i upravljačkog bloka kome je dodeljena kombinaciona mreža sa signalom logičkog uslova ISCZ.

Ispred kombinacione mreže sa signalom logičkog uslova MQ_0 ne postoji ILI element jer se u upravljački blok kome je dodeljena kombinaciona mreža sa signalom logičkog uslova MQ_0 dolazi samo iz operacionog bloka K_2 kome je dodeljen flip-flop K_2.

Iz istih razloga

- ispred flip-flopa K_3, flip-flopa K_5, kombinacione mreže sa signalom logičkog uslova ISCZ i flip-flopova K_6 i K_7 ne postoji ILI element i
- ispred flip-flopa K_4 postoji ILI element sa dva ulaza.

Na početku se jedinica nalazi u flip-flopu K_0, dok je u svim ostalim flip-flopovima nula. Ovo početno stanje se realizuje tako što se prilikom uključenja napajanja uređaja ili aktiviranja posebnog tastera za dovođenje uređaja u početno stanje generiše vrednost 0 signala

RESET određenog trajanja. Ovaj signal se vodi na asinhroni ulaz S_d flip-flopa K_0 i asinhrone ulaze R_d preostalih flip-flopova, pa se u flip-flop K_0 upisuje vrednost 1 a u ostale flip-flopove vrednost 0. Kada se signal RESET vrati na vrednost 1, on nadalje ostaje na vrednosti 1. Zbog toga se tokom rada uređaja preko asinhronog ulaza S_d flip-flopa K_0 i asinhronih ulaza R_d preostalih flip-flopova stanje flip-flopova više ne menja, već se to čini samo sinhrono signalom takta.

Jedinica ostaje u flip-flopu K_0 sve vreme dok je signal OEU nula. Tek kada signal OEU postane jedan, jedinica se prebacuje iz flip-flopa K_0 u flip-flop K_1, a u flip-flop K_0 se upisuje nula.

Kada se na signal takta jedinica pojavi u flip-flopu K_1, signali ldBB, ldMQ, clAB8, clAB i ldISC postaju jedan i ostaju jedan do pojave sledećeg signala takta na koji se jedinica prebacuje iz flip-flopa K_1 u flip-flop K_2 a u flip-flop K_1 se upisuje nula.

Kada se na signal takta jedinica pojavi u flip-flopu K_2, svi signali su nula do pojave sledećeg signala takta na koji se jedinica u zavisnosti od vrednosti signala logičkog uslova MQ_0 prebacuje iz flip-flopa K_2 ili u flip-flop K_3 ili u flip-flop K_4, a u flip-flop K_2 se upisuje nula.

Kada se na signal takta jedinica pojavi u flip-flopu K_3, signali S_1, ldAB i ldAB8 postaju jedan i ostaju jedan do pojave sledećeg signala takta na koji se jedinica prebacuje iz flip-flopa K_3 u flip-flop K_4, a u flip-flop K_3 se upisuje nula.

Kada se na signal takta jedinica pojavi u flip-flopu K_4, signali srAB8, srAB, srMQ i decISC postaju jedan i ostaju jedan do pojave sledećeg signala takta na kojia se jedinica prebacuje iz flip-flopa K_4 u flip-flop K_5, a u flip-flop K_4 se upisuje nula.

Kada se na signal takta jedinica pojavi u flip-flopu K_5, svi signali su nula do pojave sledećeg signala takta na koji se jedinica u zavisnosti od vrednosti najpre signala logičkog uslova ISCZ prebacuje iz flip-flopa K_5 ili u flip-flop K_2 ili u flip-flop K_6, a u flip-flop K_5 se upisuje nula.

Kada se na signal takta jedinica pojavi u flip-flopu K_6, signali ABout i $ldABWU_{EU}$ postaju jedan i ostaju jedan do pojave sledećeg signala takta na koji se jedinica prebacuje iz flip-flopa K_6 u flip-flop K_7 a u flip-flop K_6 se upisuje nula.

Kada se na signal takta jedinica pojavi u flip-flopu K_7, signali clOEF i stOCF postaju jedan i ostaju jedan do pojave sledećeg signala takta na koji se jedinica prebacuje iz flip-flopa K_7 u flip-flop K_0, a u flip-flop K_7 se upisuje nula.

Upravljački signali operacione jedinice se generišu prema izrazima
ldBB = K_1, ldMQ = K_1, clAB8 = K_1, clAB = K_1, ldISC = K_1,
S_1 = K_3, ldAB8 = K_3, ldAB = K_3,
srAB8 = K_4, srAB = K_4, srMQ = K_4, decISC = K_4,
ABout = K_6, $ldABWU_{EU}$ = K_6,
MQout = K_7, $ldMQWU_{EU}$ = K_7, clOEU = K_7 i $stOWU_{EU}$ = K_7
у којима K_0, K_1, ..., K_7 представљају сигнале са излаза флип-флопова придружених операционим блоковима у дијаграму тока управљачких сигнала u kojima se dati signali javljaju (слика 12). Сви остали управљачки сигнали имају вредност 0.

1.2.2.1.2 SEKVENCIJALNA MREŽA

Realizacija upravljačke jedinice tipa šetajuća jedinica je pogodna ukoliko u dijagramu toka ima manji broj operacionih blokova. Međutim, s obzirom na to da se svakom operacionom

bloku pridružuje jedan flip-flop, sa povećanjem broja operacionih blokova povećava se i broj flip-flopova potrebnih za pridruživanje operacionim blokovima.

Ukoliko se posmatraju signali na izlazima flip-flopova pridruženih operacionim blokovima, uočava se da u nekom trenutku samo jedan od tih signala ima vrednost 1 dok svi ostali imaju vrednost 0. Takva ista situacija sa signalima pridruženih operacionim blokovima može se postići i korišćenjem sekvencijalne prekidačke mreže i dekodera koji bi se koristio za dekodovanje stanja sekvencijalne prekidačke mreže. Ukoliko se posmatraju signali na izlazima dekodera, uočava se da u nekom trenutku, u zavisnosti od stanja u kome se nalazi sekvencijalna prekidačka mreža, samo jedan od tih signala ima vrednost 1 dok svi ostali signali imaju vrednost 0.

U slučaju dijagrama toka sa slike 12 u kome ima 8 operacionih blokova potrebna je sekvencijalna prekidačka mreža sa 8 stanja. Za realizaciju ove sekvencijalne prekidačke mreže dovoljna su tri flip-flopa, pri čemu bi se koristilo svih 8 mogućih stanja. Pored toga potreban je i dekoder sa tri ulaza i 8 izlaza. Signale 8 dekodovanih stanja sekvencijalne prekidačke mreže sa izlaza dekodera potrebno je pojedinačno pridružiti svakom od 8 operacionih blokova i koristiti za generisanje vrednosti 1 upravljačkih signala operacione jedinice koji se pojavlju u operacionom bloku kome je pridružen dati signal dekodovanog stanja. Pored toga, potrebno je tako generisati signale pobuda sekvencijalne prekidačke mreže da ona menja stanja u saglasnosti sa algoritmom operacije i vrednostima signala logičkih uslova predstavljenih dijagramom toka. Tada signali dekodovanih stanja sekvencijalne prekidačke mreže sa izlaza dekodera pridruženi operacionim blokovima u dijagramu toka dobijaju vrednosti 1 i 0 na identičan način kao i signali na izlazima flip-flopova pridruženi operacionim blokovima u slučaju realizacije upravljačke jedinice tipa šetajuća jedinica.

Postupak projektovanja strukturne šeme upravljačke jedinice tipa sekvencijalna mreža zahteva da se svakom operacionom bloku u dijagramu toka upravljačkih signala pridruži posebno stanje sekvencijalne prekidačke mreže. Tada se dijagram toka upravljačkih signala (slika 12) pretvara u graf stanja redukovanih dimenzija sekvencijalne mreže u kome se svakom putu iz sadašnjeg stanja i sledeće stanje
- koji prolazi kroz upravljački blok, pridružuje neki od signala logičkih uslova i to
  - direktna vrednost signala ako put prolazi kroz izlaz označen sa 1 i
  - komplementarna vrednost signala ako put prolazi kroz izlaz označen sa 0, dok putu
- koji ne prolazi kroz upravljačke blokove, pridružuje vrednost 1.

Na osnovu grafa stanja redukovanih dimenzija se konstruiše tablica koja sadrži tablicu stanja redukovanih dimenzija i tablicu prelaza/izlaza redukovanih dimenzija. U ovoj tablici postoji posebna vrsta za svako stanje sekvencijalne mreže, sa simboličkim oznakama stanja. U tablici se za svako stanje daje kako mreža prelazi iz stanja u sadašnjem trenutku u stanje u sledećem trenutku. Stoga u tablici postoji posebna kolona u kojoj je za dato stanje vrednost 1 ukoliko prelazak na stanje u sledećem trenutku ne zavisi ni od jednog od signala logičkih uslova, dok su u slučajevima kada taj prelazak zavisi od nekog od signala logičkih uslova data stanja sekvencijalne mreže u sledećem trenutku za obe moguće vrednosti određenog signala logičkog uslova. U tablici postoji i posebna kolona koja je, za one vrste koje odgovaraju stanjima pridruženim operacionim blokovima u kojima se pojavlju neki od upravljačkih signala operacione jedinice, popunjena oznakama datih signala, dok je data kolona, za one vrste koje odgovaraju stanjima pridruženim praznim operacionim blokovima, ostavljena bez oznaka signala.

U daljem postupku sinteze treba kodirati stanja sekvencijalne prekidačke mreže. U opštem slučaju stanja se mogu kodirati na proizvoljan način. Međutim, trebalo bi stanja sekvencijalne

prekidačke mreže da budu kodirana na takav način da se pri prelasku sekvencijalne prekidačke mreže iz stanja u sadašnjem u stanje u sledećem trenutku menja što je moguće manji broj koordinata vektora stanja. Za tako kodirana stanja kombinaciona mreža koja generiše signale pobuda za flip-flopove dodeljene koordinatama vektora stanja bi trebalo da bude najjednostavnija u odnosu na neke druge moguće načine kodiranja stanja. Usvojene binarne vrednosti stanja treba staviti u zagradi pored simboličkih oznaka stanja.

Zatim treba usvojiti tip flip-flopa za realizaciju stanja sekvencijalne mreže i u skladu sa tim dodati kolone sa vrednostima signala pobuda flip-flopova kojima se obezbeđuju prelasci mreže iz stanja u sadašnjem trenutku u stanje u sledećem trenutku. Na osnovu vrednosti signala pobuda treba odrediti i izraze za signale pobuda kao funkcije signala stanja i signala logičkih uslova.

Na osnovu dobijenih izraza za signale pobuda realizuje se sinteza kombinacionih mreža za signale pobuda i signali sa izlaza kombinacione mreže povezuju na ulaze odabranih flip-flopa kojima se realizuju potrebna stanja mreže.

Signali dekodovanih stanja mreže se dobijaju na izlazima dekodera na čije ulaze se dovode signali sa izlaza flip-flopova kojima se realizuju stanja. Ovi signali se sa signalima logičkih uslova koriste u kombinacionoj mreže koja generiše signale pobuda flip-flopova kojima se realizuju stanja mreže. Pored toga signali dekodovanih stanja mreže se koriste i za generianje upravljačkih signala operacione jedinice. Za svaki upravljački signal operacione jedinice treba posmatrati u kojim operacionim blokovima se pojavlju i dati signal formirati kao uniju signala dekodovanih stanja mreže pridruženih datim operacionim blokovima.

U skladu sa iznetim postupkom projektovanja strukturne šeme upravljačke jedinice tipa sekvencijalna mreža dolazi se do strukturne šeme upravljačke jedinice za operaciju množenja realizovane kao sekvencijalna mreža (slika 20).

Najpre je graf stanja redukovanih dimenzija formiran od dijagrama toka upravljačkih signala (slika 12) tako što je svakom od 8 operacionih blokova označenih sa K_0, K_1, K_2, K_3, K_4, K_5, K_6 i K_7 pridruženo jedno od 8 stanja sekvencijalne prekidačke mreže označeno sa K_0, K_1, K_2, K_3, K_4, K_5, K_6 i K_7, respektivno.

U daljem postupku sinteze su kodirana stanja K_0, K_1, K_2, K_3, K_4, K_5, K_6 i K_7 sekvencijalne prekidačke mreže. U opštem slučaju stanja se mogu kodirati na proizvoljan način. Međutim, trebalo bi stanja sekvencijalne prekidačke mreže da budu kodirana na takav način da se pri prelasku sekvencijalne prekidačke mreže iz stanja u sadašnjem u stanje u sledećem trenutku menja što je moguće manji broj koordinata vektora stanja (slika 18). Za tako kodirana stanja kombinaciona mreža koja generiše signale pobuda za flip-flopove dodeljene koordinatama vektora stanja bi trebalo da bude najjednostavnija u odnosu na neke druge moguće načine kodiranja stanja.

Q_1Q_2 / Q_3	00	01	11	10
0	K_0	K_3	K_4	K_7
1	K_1	K_2	K_5	K_6

Slika 18 Kodiranje stanja za jedinicu sa jednom operacijom

U skladu sa ovom sugestijom 8 mogućih stanja koja se koriste su kodirana na sledeći način:

- K_0=000, K_1=001, K_2=011, K_3=010, K_4=110, K_5=111, K_6=101 i K_7 =100

Ako se posmatraju prelasci u grafu redukovanih dimenzija (slika 12), vidi se da se pri prelasku mreže

- iz stanja K_0(000) u stanje K_1(001) menja sa 0 na 1 samo treća koordinata vektora stanja dok su prva i druga 0 i 0,
- iz stanja K_1(001) u stanje K_2(011) menja sa 0 na 1 samo druga koordinata vektora stanja dok su prva i treća 0 i 1,
- iz stanja K_2(011) u stanje K_3(010) menja sa 1 na 0 samo treća koordinata vektora stanja dok su prva i druga 0 i 1,

Međutim pri prelasku mreže

- iz stanja K_2(011) u stanje K_4(110) menjaju se sa 0 na 1 prva koordinata i sa 1 na 0 treća koordinata vektora stanja, dok je druga 1.

U svim ostalim prelascima (K_3(010) u K_4(110), K_4(110) u K_5(111), K_5(111) u K_2(011), K_5(111) u K_6(101), K_6(101) u K_7(100) i K_7(100) u K_0(000)) menja se samo jedna koordinata vektora stanja.

Na osnovu grafa stanja redukovanih dimenzija (slika 12), kodiranih stanja sekvencijalne prekidačke mreže upravljačke jedinice (slika 18) i izbora flip-flopa JK tipa za realizaaciju stanja sekvencijalne mreže, konstruisana je tablica (slika 19) koja sadrži tablicu stanja redukovanih dimenzija, tablicu prelaza/izlaza redukovanih dimenzija i kombinacionu tablicu funkcija pobude flip-flopova redukovanih dimenzija.

Q	Z	Q(t+1)	X	J_1	K_1	J_2	K_2	J_3	K_3
K_0 (000)	/	K_0 (000)	$\overline{OEU}$	0	b	0	b	0	b
		K_1 (001)	OEU	0	b	0	b	1	b
K_1 (001)	$ldBB_1$, ldMQ, clAB8, clAB, ldISC	K_2 (011)	1	0	b	1	b	b	0
K_2 (011)	/	K_3 (010)	MQ_0	0	b	b	0	b	1
		K_4 (110)	$\overline{MQ_0}$	1	b	b	0	b	1
K_3 (010)	S_1, ldAB, ldAB8	K_4 (110)	1	1	b	b	0	0	b
K_4 (110)	srAB8, srAB, srMQ	K_5 (111)	1	b	0	b	0	1	b
K_5 (111)	/	K_2 (011)	$\overline{ISCZ}$	b	1	b	0	b	0
		K_6 (101)	ISCZ	b	0	b	1	b	0
K_6 (101)	ABout, $ldABWU_{EU}$	K_7 (100)	1	b	0	0	b	b	1
K_7 (100)	MQout, $ldMQWU_{EU}$, clOEU, $stOWU_{EU}$	K_0 (000)	1	b	1	0	b	0	b

Slika 19 Tablica za upravljačku jedinicu tipa sekvencijalna mreža sa JK flip-flopovima za jedinicu sa jednom operacijom

Ova tablica je konstruisana iz tri koraka. Najpre je na osnovu grafa stanja (slika 12) konstruisana je tablica stanja (slika 19) u kojoj se pojavljuju samo simboličke oznake stanja K_0, K_1, ..., K_7 a ne i njihove binarne vrednosti i nema kolona sa signalima pobuda J_1, K_1, J_2, K_2, J_3 i K_3. Potom su pored simboličkih oznaka stanja K_0, K_1, ..., K_7 stavljene i odgovarajuće binarne vrednosti 000, 001, ..., 100 saglasno usvojenom načinu kodiranja stanja i dobijena i tablica prelaza/izlaza. Na kraju su popunjene kolone sa signalima pobuda J_1, K_1, J_2, K_2, J_3 i K_3 i dobijena kombinaciona tablica funkcija pobude flip-flopova. Prilikom popunjavanja kolona J_1 i K_1, posmatra se koordinata vektora stanja u sadašnjem trenutku Q_1 i sledećem trenutku Q_1(t+1) i saglasno funkcijama pobuda JK flip-flopa upisuju vrednosti signala pobuda J_1 i K_1 kojima se dati prelasci realizuju. Na isti način se popunjavaju i kolone za J_2 i K_2, pri čemu se posmatraju Q_2 i Q_2(t+1) i kolone za J_3 i $K3$, pri čemu se posmatraju Q_3 i Q_3(t+1). Ovo je urađeno uz pretpostavku da treba koristiti JK flip-flopove za realizaciju stanja sekvencijalne prekidačke mreža. Stanja sekvencijalne mreže mogu da se realizuju i sa nekim drugim tipom flip-flopa (D ili T ili RS), pa bi se tada pojavilie kolone koje treba popunjavati sa vrednostima

signala pobuda za dati tip flip-flopa (D_1, D_2, D_3 ili T_1, T_2, T_3 ili R_1, S_1, $R2$, S_2, R_3, S_3) da bi se isti prelasci iz stanja u sadašnjem trenutku u stanje u sledećem trenutku realizovali.

Iz tablice sa slike 19 se dobijaju izrazi sa signale pobuda J_1, K_1, J_2, K_2, J_3 i K_3.

Iz kolone za J_1 se vidi da signal pobude J_1 treba da ima vrednost 1 kada se sekvencijalna mreža nalazi u stanju K_2 i signal logičkog uslova MQ_0 ima vrednost 0 (pa $\overline{MQ_0}$ ima vrednost 1) i kada se sekvencijalna mreža nalazi u stanju K_3, dok u ostalim situacijama ili treba da ima vrednost 0 ili nije bitno da li ima vrednost 0 ili 1, jer se u tim situacijama pojavljuje "b". Stoga se može uzeti da je $J_1 = K_2 \cdot \overline{MQ_0} + K_3$.

Iz kolone za K_1 se vidi da signal pobude K_1 treba da ima vrednost 1 kada se sekvencijalna mreža nalazi u stanju K_5 i signal logičkog uslova ISCZ ima vrednost 0 i kada se sekvencijalna mreža nalazi u stanju K_7, dok u ostalim situacijama ili treba da ima vrednost 0 ili nije bitno da li ima vrednost 0 ili 1, jer se u tim situacijama pojavljuje "b". Stoga se može uzeti da je $K_1 = K_5 \cdot \overline{ISCZ} + K_7$.

Iz kolone za J_2 se vidi da signal pobude J_2 treba da ima vrednost 1 kada se sekvencijalna mreža nalazi u stanju K_1, dok u ostalim situacijama ili treba da ima vrednost 0 ili nije bitno da li ima vrednost 0 ili 1, jer se u tim situacijama pojavljuje "b". Stoga se može uzeti da je $J_2 = K_1$.

Istim postupkom se utvrđuju i izrazi za signale pobuda K_2, J_3 i K_3.

Na osnovu toga se dobijaju sledeći izrazi za signale pobuda flip-flopova:

$J_1 = K_2 \cdot \overline{MQ_0} + K_3$

$K_1 = K_5 \cdot \overline{ISCZ} + K_7$

$J_2 = K_1$

$K_2 = K_5 \cdot ISCZ$

$J_3 = K_0 \cdot OEU + K_4$

$K_3 = K_2 + K_6$

S obzirom na to da su u grafu stanja redukovanih dimenzija (slika 12) stanja sekvencijalne mreže kodirana na sledeći način (slika 18)

- $K_{\mathbf{0}} = 000$, $K_{\mathbf{1}} = 001$, $K_{\mathbf{2}} = 011$, $K_{\mathbf{3}} = 010$, $K_{\mathbf{4}} = 110$, $K_{\mathbf{5}} = 111$, $K_{\mathbf{6}} = 101$ i $K_{\mathbf{7}} = 100$.

pa je

- $K_{\mathbf{0}} = \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3}$,
- $K_{\mathbf{1}} = \overline{Q_1} \cdot \overline{Q_2} \cdot Q_3$,
- $K_{\mathbf{2}} = \overline{Q_1} \cdot Q_2 \cdot Q_3$
- $K_{\mathbf{3}} = \overline{Q_1} \cdot Q_2 \cdot \overline{Q_3}$
- $K_{\mathbf{4}} = Q_1 \cdot Q_2 \cdot \overline{Q_3}$
- $K_{\mathbf{5}} = Q_1 \cdot Q_2 \cdot Q_3$,
- $K_{\mathbf{6}} = Q_1 \cdot \overline{Q_2} \cdot Q_3$,
- $K_{\mathbf{7}} = Q_1 \cdot \overline{Q_2} \cdot \overline{Q_3}$,

izrazi za signale pobuda se mogu napisati i u obliku

- $J_1 = \overline{Q_1} \cdot Q_2 \cdot Q_3 \cdot \overline{MQ_0} + \overline{Q_1} \cdot Q_2 \cdot \overline{Q_3}$
- $K_1 = Q_1 \cdot Q_2 \cdot Q_3 \cdot \overline{ISCZ} + Q_1 \cdot \overline{Q_2} \cdot \overline{Q_3}$
- $J_2 = \overline{Q_1} \cdot \overline{Q_2} \cdot Q_3$
- $K_2 = Q_1 \cdot Q_2 \cdot Q_3 \cdot ISCZ$

- $J_3 = \overline{Q_1} \cdot \overline{Q_2} \cdot \overline{Q_3} \cdot OEU + Q_1 \cdot Q_2 \cdot \overline{Q_3}$
- $K_3 = \overline{Q_1} \cdot Q_2 \cdot Q_3 + Q_1 \cdot \overline{Q_2} \cdot Q_3$

ukoliko ukoliko se u njima i umesto signala dekodovanih stanja K_0, K_1, ..., K_7 stave signali koordinata vektora stanja Q_1, Q_2 i Q_3 saglasno načinu kodiranja stanja.

Upravljački signali operacione jedinice se generišu prema izrazima koji su identični sa odgovarajućim izrazima za upravljačku jedinice realizovanu kao šetajuća jedinica (odeljak 1.3.2.1.1). U datim izrazima sada K_0, K_1, ..., K_7 представљају сигнале декодованих стања sekvencijalne mreže придружених операционим блоковима у дијаграму тока управљачких сигнала (slike 12). Ovi izrazi se mogu napisati i u obliku

ldBB= $\overline{Q_1} \cdot \overline{Q_2} \cdot Q_3$, ldMQ= $\overline{Q_1} \cdot \overline{Q_2} \cdot Q_3$, clAB8= $\overline{Q_1} \cdot \overline{Q_2} \cdot Q_3$, clAB= $\overline{Q_1} \cdot \overline{Q_2} \cdot Q_3$,

ldISC= $\overline{Q_1} \cdot \overline{Q_2} \cdot Q_3$,

S_1= $\overline{Q_1} \cdot Q_2 \cdot \overline{Q_3}$, ldAB= $\overline{Q_1} \cdot Q_2 \cdot \overline{Q_3}$, ldAB8= $\overline{Q_1} \cdot Q_2 \cdot \overline{Q_3}$,

srAB8= $Q_1 \cdot Q_2 \cdot \overline{Q_3}$, srAB= $Q_1 \cdot Q_2 \cdot \overline{Q_3}$, srMQ= $Q_1 \cdot Q_2 \cdot \overline{Q_3}$, decISC= $Q_1 \cdot Q_2 \cdot \overline{Q_3}$,

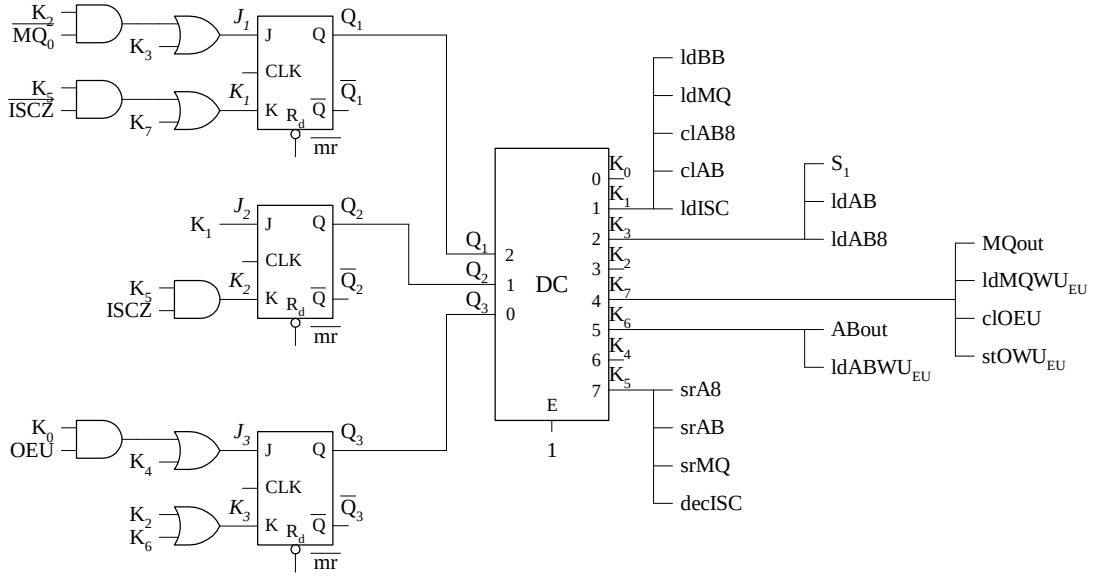
ABout= $Q_1 \cdot \overline{Q_2} \cdot Q_3$, ldABWU$_{EU}$= $Q_1 \cdot \overline{Q_2} \cdot Q_3$,

MQout= $Q_1 \cdot \overline{Q_2} \cdot \overline{Q_3}$, ldMQWU$_{EU}$= $Q_1 \cdot \overline{Q_2} \cdot \overline{Q_3}$,

clOEU= $Q_1 \cdot \overline{Q_2} \cdot \overline{Q_3}$, stOWU$_{EU}$= $Q_1 \cdot \overline{Q_2} \cdot \overline{Q_3}$

ukoliko se u njima i umesto signala dekodovanih stanja K_0, K_1, ..., K_7 stave signali koordinata vektora stanja Q_1, Q_2 i Q_3 saglasno načinu kodiranja stanja.

Na osnovu izraza sa signale pobuda pobuda flip-flopova i usvojenog načina kodiranja stanja, dolazi se do strukturne šeme upravljačke jedinice date na slici 20.



Slika 20 Upravljačka jedinica tipa sekvencijalna mreža za jedinicu sa jednom operaciojom

U strukturnoj šemi postoje

- 3 flip-flopa Q_1, Q_2 i Q_3 za realizaciju stanja sekvencijalne prekidačke mreže,
- dekoder DC za realizaciju signala stanja K_0, K_1, ..., K_7 i
- kombinaciona mreža za generisanje signala pobuda J_1, K_1, J_2, K_2, J_3 i K_3.

Usvojena je najjednostavnija realizacija kod koje se

- signali pobuda generišu na osnovu signala stanja K_0, K_1, ..., K_7 i signala logičkih uslova OEU, MQ_0 i ISCZ prema izrazima

$J_1 = K_2 \cdot \overline{MQ_0} + K_3$

$K_1 = K_5 \cdot \overline{ISCZ} + K_7$
$J_2 = K_1$
$K_2 = K_5 \cdot ISCZ$
$J_3 = K_0 \cdot OEU + K_4$
$K_3 = K_2 + K_6$

i

- upravljački signali operacione jedinice generišu na osnovu signala stanja K_0, K_1, ..., K_7 prema izrazima

ldBB= K_1, ldMQ= K_1, clAB8= K_1, clAB= K_1, ldISC= K_1,
S_1= K_3, ldAB= K_3, ldAB8= K_3,
srAB8= K_4, srAB= K_4, srMQ= K_4, decISC= K_4,
ABout= K_6, ld$ABWU_{EU}$= K_6,
MQout= K_7, ld$MQWU_{EU}$= K_7,
clOEU= K_7, stOWU_{EU}= K_7

1.2.2.1.3 BROJAČ KORAKA

Stanja K_0, K_1, K_2, K_3, K_4, K_5, K_6 i K_7 u grafu redukovanih dimenzija (slika 12) mogu se kodirati i na sledeći način:

- K_0=000, K_1=001, K_2=010, K_3=011, K_4=100, K_5=101, K_6=110 i K_7=111.

Ako se posmatraju prelasci u grafu redukovanih dimenzija (slika 12), vidi se da je pri prelasku mreže

- iz stanja K_0=000 u stanje K_1=001, iz stanja K_1=001 u stanje K_2=010, iz stanja K_2=010 u stanje K_3=011, iz stanja K_3=011 u stanje K_4=100, iz stanja K_4=100 u stanje K_5=101, iz stanja K_5=101u stanje K_6=110 i iz stanja K_6=110 u stanje K_7=111, stanje u sledećem trenutku posmatrano kao binarni broj jednako stanju u sadašnjem trenutku posmatrano kao binarni broj plus jedan dok
- dok pri prelascima iz stanja K_0=000 u stanje K_0=000, iz stanja K_2=010 u stanje K_4=100, iz stanja K_5=101 u stanje K_2=010 i iz stanja K_7=111u stanje K_0=000 to nije slučaj.

U prvom slučaju prelasci su sekvencijalni, dok se u drugom slučaju ili ostaje u istom stanju ili se odstupa od sekvencijalnog prelaska iz stanja u sadašnjem trenutku u stanje u sledećem trenutku.

Kod ovakvog načina kodiranja stanja, sekvencijalna mreža bi mogla da se realizuje korišćenjem inkrementirajućeg brojača koraka

- čiji bi se sadržaj inkrementirao prilikom sekvencijalnih prelazaka i u
- koji bi se upisivala ili ista vrednost za slučaj kada se ostaje u istom stanju ili bi se upisivala nova vrednost stanja prilikom prelazaka koji nisu sekvencijalni.

Postupak projektovanja strukturne šeme upravljačke jedinice tipa brojač koraka zahteva da se operacionim blokovima K_0, K_1, ..., K_7 u dijagramu toka upravljačkih signala (slika 12) ili koracima $step_0$, $step_1$, ..., $step_7$ u sekvenci upravljačkih signala po koracima (tabela 1) pridruže signali dekodovanih stanja brojača koraka K_0, K_1, ..., K_7 (slika 21) koji odgovaraju stanjima 000, 001, ..., 111 brojača koraka, respektivno.

Stanja brojača koraka bi se dekodovala korišćenjem dekodera. Signali K_0, K_1, ..., K_7 sa izlaza dekodera bi se koristili za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice.

Upravljački signali operacione jedinice se generišu na osnovu dekodovanih stanja brojača koraka K_0, K_1, ..., K_7 dodeljenih operacionim blokovima u kojima dati signali treba da imaju vrednost 1, pa je:

ldBB= K_1, ldMQ= K_1, clAB8= K_1, clAB= K_1, ldISC= K_1,
S_1= K_3, ldAB= K_3, ldAB8= K_3,
srAB8= K_4, srAB= K_4, srMQ= K_4, decISC= K_4,
ABout= K_6, ld$ABWU_{EU}$= K_6,
MQout= K_7, ld$MQWU_{EU}$= K_7, clOEU= K_7, stOWU_{EU}= K_7

Upravljački signali upravljačke jedinice bi trebalo da omoguće inkrementiranje brojača koraka u svim situacijama osim u koraku K_0 kada treba uslovno, ako $\overline{OEU}$ ima vrednost 1, upisati 0, u koraku K_2 kada treba uslovno, ako je $\overline{MQ_0}$ ima vrednost 1, upisati 4, u koraku K_5 kada treba uslovno, ako $\overline{ISCZ}$ ima vrednost 1, upisati 2, i koraku K_7 kada treba bezuslovno upisati 0. To se postiže generisanjem upravljačkog signala za upis u brojač koraka koji ima vrednost 0 u koracima u kojima treba inkrementirati brojač koraka, vrednost 1 ili 0 u koracima K_0, K_2 i K_5 u kojima treba uslovno u zavisnosti od vrednosti signala logičkih uslova $\overline{OEU}$, $\overline{MQ_0}$ i $\overline{ISCZ}$, respektivno, upisivati novu vrednost ili ikrementirati brojač koraka i vrednost 1 u koraku K_7 u kome bezuslovno treba upisivati novu vrednost. Pored toga na paralelnim ulazima brojača korak treba za upis u koracima K_0 i K_7 generisati vrednost 0, u koraku K_2 vrednost 4 i u koraku K_5 vrednost 2.

Upravljačka jedinica generiše dve vrste upravljačkih signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice. Upravljački signali operacione jedinice se koriste u operacionoj jedinici radi izvršavanja mikrooperacija. Upravljački signali upravljačke jedinice se koriste u upravljačkoj jedinici radi inkrementiranja brojača koraka ili upisa nove vrednosti u brojač koraka i radi generisanja vrednosti za upis u brojač koraka.

Upravljački signali operacione jedinice bi mogli da se generišu na osnovu sekvence upravljačkih signala po koracima (tabela 1). Za svaki upravljački signal operacione jedinice trebalo bi proći kroz sekvencu upravljačkih signala po koracima, tražiti korake u kojima se pojavljuje dati signal i izraz za dati signal formirati kao uniju signala dekodovanih stanja brojača koraka koji odgovaraju koracima u kojima se pojavljuje dati signal.

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

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

Za upravljačke signale upravljačke jedinice treba u sekvenci upravljačkih signala po koracima tražiti iskaze *br* $step_A$ i *br* (*if* **uslov** *then* $step_A$) i

Umesto iskaza *br* $step_A$ treba u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije staviti signal bezuslovnog skoka koji određuje da se bezuslovno prelazi na korak $step_A$ i signal $\mathbf{val_A}$ koji određuje da treba formirati binarnu vrednost A za upis u brojač koraka. Simbolička oznaka signala bezuslovnog skoka je **bruncnd**. Koraci $step_i$ u kojima se

bezuslovni skokovi javljaju, koraci $step_A$ na koje se bezuslovno skače, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A u heksadecimalnom dati su u tabeli 2.

Tabela 2 Koraci $step_i$, $step_A$, signali $\mathbf{val_A}$ i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	$\mathbf{val_A}$	A
$step_{07}$	$step_{00}$	$\mathbf{val_{00}}$	00

Umesto iskaza *br* (*if* **uslov** *then* $step_A$) treba u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije staviti signal uslovnog skoka pridružen signalu **uslov** koji treba da ima vrednost 1 da bi se realizovao prelaz na korak $step_A$ i signal $\mathbf{val_A}$ koji određuje da treba formirati binarnu vrednost A za upis u brojač koraka u slučaju da signal **uslov** ima vrednost 1. Simboličke oznake pridruženih signala uslovnih skokova i signala uslova za sve iskaze ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima, dati su u tabeli 3. Koraci $step_i$ u kojima se uslovni skokovi javljaju, signali uslova **uslov**, koraci $step_A$ na koje se uslovno skače, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A u heksadecimalnom obliku dati su u tabeli 4.

Tabela 3 Signali uslovnih skokova i signali uslova

signal uslovnog skoka	signal uslova
brnotOEU	$\overline{\mathbf{OEU}}$
brnotMQ0	$\overline{\mathbf{MQ_0}}$
brnotISCZ	$\overline{\mathbf{ISCZ}}$

Tabela 4 Koraci $step_i$, signali uslova **uslov**, koraci $step_A$, signali $\mathbf{val_A}$ i vrednosti A za uslovne skokove

$step_i$	uslov	$step_A$	$\mathbf{val_A}$	A
$step_{00}$	$\overline{\mathbf{OEU}}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{02}$	$\overline{\mathbf{MQ_0}}$	$step_{04}$	$\mathbf{val_{04}}$	04
$step_{05}$	$\overline{\mathbf{ISCZ}}$	$step_{02}$	$\mathbf{val_{02}}$	02

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu ožičene realizacije signal bezuslovnog skoka **bruncnd**, signali uslovnih skokova (tabela 3) i signali $\mathbf{val_A}$ za bezuslovne (tabela 2) i uslovne (tabela 4) skokove.

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela tabela 1), formirana sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 5). Jedna linija u toj sekvenci ima sledeću formu: na levoj strani nalazi se signal dekodovanog stanja brojača koraka, u sredini je niz upravljačkih signala operacione i upravljačke jedinice koji imaju vrednost 1 kada dati signal dekodovanog stanja brojača koraka ima vrednost 1, dok komentar, tamo gde postoji, počinje uskličnikom (!) i proteže se do sledećeg uskličnika (!).

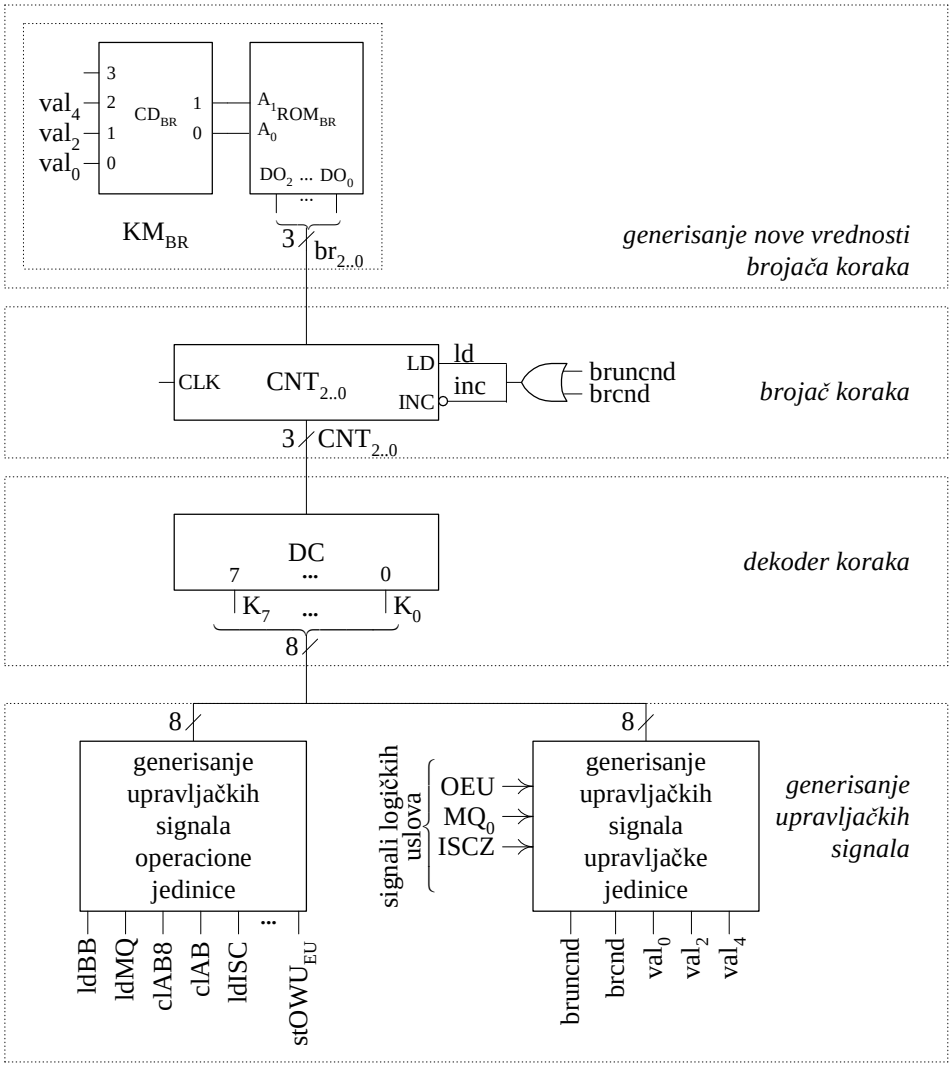
Tabela 5 Sekvenca upravljačkih signala za upravljačku jedinicu tipa brojač koraka za jedinicu sa jednom operacijom

K_0 **brnotOEU**, val_0;
K_1 **ldBB, ldMQ, clAB8, clAB, ldISC**;
K_2 **brnotMQ0**, val_4;
K_3 $\mathbf{S_1}$, ldAB, ldAB8;
K_4 **srAB8**, srAB, srMQ, decISC;
K_5 **brnotISCZ**, val_2;
K_6 ABout, $ldABWU_{EU}$;

K_7 MQout, ld$MQWU_{EU}$, clOEU, stOWU_{EU},
bruncnd, val_0;

Upravljački signali operacione jedinice i upravljačke jedinice se generišu na identičan način na osnovu sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 5). Za svaki upravljački signal operacione jedinice i upravljačke jedinice treba proći kroz sekvencu upravljačkih signala za upravljačku jedinicu ožičene realizacije, tražiti korake u kojima se pojavljuje dati signal i izraz za dati signal formirati kao uniju signala dekodovanih stanja brojača koraka koji odgovaraju koracima u kojima se pojavljuje dati signal.

Struktura upravljačke jedinice tipa brojač koraka je prikazana na slici 21. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti brojača koraka*, blok *brojač koraka*, blok *dekoder koraka* i blok *generisanje upravljačkih signala*. Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.



Slika 21 Upravljačka jedinica tipa brojač koraka za jedinicu sa jednom operacijom

Blok *generisanje nove vrednosti brojača koraka* se sastoji od kombinacione mreže KM_{BR} i služi za generisanje vrednosti koju treba upisati u brojač koraka $CNT_{2..0}$. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikrooperacija. Kombinacionom mrežom KM_{BR} generišu se vrednosti za upis u brojač koraka $CNT_{7...0}$ za bezuslovne skokove (tabela 2) i uslovne skokove (tabela 4) u sekvenci upravljačkih signala po koracima. Kombinacionu mrežu KM_{BR} čine koder CD_{BR} i ROM memorija ROM_{BR}. Na ulaze 0, 1 i 2

kodera CD_{BR} vezani su signali **val_0**, **val_2** i **val_4**, respektivno, a na adresama 0, 1 i 2 memorije ROM_{BR} nalaze se vrednosti 0, 2 i 4, respektivno. U zavisnosti od toga koji od signala **val_0**, **val_2** i **val_4** ima vrednost 1 na izlazima kodera se pojavljuje adresa memorijske lokacije sa koje se čita i na linijama **$br_{2..0}$** pojavljuje vrednost koji treba upisati u brojač koraka $CNT_{2..0}$.

Blok *brojač koraka* sadrži brojač $CNT_{2..0}$. Brojač $CNT_{2..0}$ svojom trenutnom vrednošću određuje koji će upravljački signali da imaju vrednost 1. Brojač $CNT_{2..0}$ može da radi u sledećim režimima: režim inkrementiranja i režim skoka.

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača $CNT_{2..0}$ za jedan čime se obezbeđuje sekvencijalno generisanje upravljačkih signala iz sekvence upravljačkih signala (tabela 5). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko svi signali **bruncnd** i **brcnd** imaju vrednost 0. Signali **bruncnd** i **brcnd** normalno imaju vrednost 0 sem u stanjima brojača koraka koja odgovaraju koracima kada treba realizovati bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

U režimu skoka pri pojavi signala takta vrši se upis nove vrednosti u brojač $CNT_{2..0}$ čime se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz sekvence upravljačkih signala (tabela 5). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ako jedan od signala **bruncnd** i **brcnd** ima vrednost 1. Signali **bruncnd** i **brcnd** normalno imaju vrednost 0 sem u stanjima brojača koraka koja odgovaraju koracima kada treba treba realizovati bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

Brojač koraka $CNT_{2..0}$ je dimenzionisan prema broju koraka u sekvenci upravljačkih signala (tabela 5). S obzirom da upravljački signali treba da se realizuju u opsegu od koraka K_0 do koraka K_7, usvojena je dužina brojača koraka $CNT_{2..0}$ od tri bita.

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača $CNT_{2..0}$. Dekodovana stanja brojača $CNT_{2..0}$ pojavljuju se kao signali **K_0**, **K_1**, ..., **K_7** na izlazima dekodera DC. Svakom koraku iz sekvence upravljačkih signala po koracima (tabela 5) dodeljeno je po jedno stanje brojača $CNT_{2..0}$ određeno vrednošću signala **K_0** do **K_7** i to koraku $step_0$ signal **K_0**, koraku $step_1$ signal **K_1**, itd.

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala **K_0**, **K_1**, ..., **K_7** koji dolaze sa bloka *dekoder koraka*, signala logičkih uslova **OEU**, **MQ_0** i **ISCZ** koji dolaze iz operacione jedinice i saglasno sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 5) generišu dve grupe upravljačkih signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice se generišu prema izrazima koji su identični sa odgovarajućim izrazima za upravljačku jedinice realizovanu kao šetajuća jedinica (odeljak 1.3.2.1.1). U datim izrazima sada **K_0**, **K_1**, ..., **K_7** представљају сигнале декодованих стања брojача koraka придружених операционим блоковима у дијаграму тока управљачких сигнала (slika 13) ili koracima u sekvenci upravljačkih signala po koracima (tabela 1).

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **bruncnd** = K_{07}
- **brcnd** = **brnotOEU**$\cdot \overline{\mathbf{OEU}}$ + **brnotMQ0**$\cdot \overline{\mathbf{MQ_0}}$ +**brnotISCZ**$\cdot \overline{\mathbf{ISCZ}}$
- **brnotOEU** = K_{00}
- **brnotMQ0** = K_{02}
- **brnotISCZ** = K_{05}

- $\mathbf{val_{00}} = K_{00}+K_{07}$
- $\mathbf{val_{02}} = K_{05}$
- $\mathbf{val_{04}} = K_{02}$

Pri generisanju signala **brcnd** koriste se signali logičkih uslova OEU, $\mathbf{MQ_0}$ i **ISCZ** koji dolaze iz operacione jedinice.

1.2.2.2 MIKROPROGRAMSKO GENERISANJE SIGNALA

Postoji više varijanti upravljačke jedinice sa mikroprogramskim generisanjem upravljačkih signala. U zavisnosti od toga kako upravljačka jedinica generiše upravljačke signale operacione jedinice i kako se upravlja radom upravljačke jedinice postoje realizacije sa jednim tipom mikroinstrukcije i sa dva tipa mikroinstrukcija. Pored toga, u zavisnosti od toga kako se vrši kodiranje upravljačkih signala operacione jedinice postoje realizacije sa horizontalnim kodiranjem signala, vertikalnim kodiranjem signala i mešovitim kodiranjem signala.

Kod mikroprogramske realizacije upravljačke jedinice, upravljački signali operacione jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu sekvence upravljačkih signala po koracima (tabela 1). Mikroprogram se nalazi u mikroprogramskoj memoriji koja predstavlja sastavni deo strukturne šeme upravljačke jedinice mikroprogramske realizacije. Zbog toga se u daljem tekstu daje najpre postupak formiranja mikroprograma, zatim postupak realizacije strukturne šeme upravljačke jedinice i na kraju funkcionisanje upravljačke jedinice. Razmatraju se realizacije sa jednim tipom mikroinstrukcije i sa dva tipa mikroinstrukcija, pri čemu se u oba slučaja daju realizacije sa horizontalnim kodiranjem upravljačkih signala operacione jedinice.

1.2.2.2.1 JEDAN TIP MIKROINSTRUKCIJE

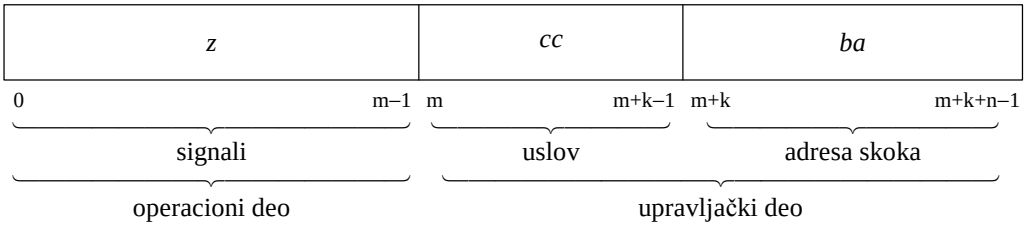
U ovom poglavlju se daje najpre postupak formiranja mikroprograma, zatim postupak realizacije strukturne šeme upravljačke jedinice i na kraju funkcionisanje upravljačke jedinice.

1.2.2.2.1.1 Formiranje mikroprograma

Upravljačka jedinica generiše dve vrste upravljačkih signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice. Upravljački signali operacione jedinice se koriste u operacionoj jedinici radi izvršavanja mikrooperacija. Upravljački signali upravljačke jedinice se koriste u upravljačkoj jedinici radi inkrementiranja mikroprogramskog brojača ili upisa nove vrednosti u mikroprogramski brojač i radi generisanja vrednosti za upis u mikroprogramski brojač.

Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu sekvence upravljačkih signala po koracima (tabela 1). Mikroprogram se formira tako što se svakom koraku u sekvenci upravljačkih signala po koracima pridružuje binarna reč sa slike 22. Ta 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 se sastoji od polja *z* dužine m bitova, a upravljački deo od polja *cc* dužine k bitova i polja *ba* dužine n bitova. Operacioni deo se koristi za generisanje upravljačkih signala operacione jedinice, a upravljački deo se koristi za generisanje upravljačkih signala upravljačke jedinice.



Slika 22 Mikroinstrukcija za upravljačku jedinicu sa jednim tipom mikroinstrukcije

Poljem *z* se određuje za svaki od upravljačkih signala operacione jedinice da li treba da ima vrednost 1 ili 0. U polju *z* postoji poseban bit pridružen svakom od upravljačkih signala. Bitovi polja *z* treba da imaju vrednost 1 ili 0 u zavisnosti od toga da li u koraku za koji se formira mikroinstrukcija upravljački signali operacione jedinice kojima su pridruženi dati bitovi imaju vrednost 1 ili 0, respektivno. Ovakav način kodiranja upravljačkih signala operacione jedinice se naziva horizontalni način kodiranja. Dužina polja *z* u bitovima treba da bude takva da postoji poseban bit za svaki upravljački signal operacione jedinice.

Polje *z* operacionog dela mikroinstrukcije se formira ukoliko u datom koraku ima upravljačkih signala operacione jedinice. U suprotnom slučaju svi bitovi operacionog dela se postavljaju na vrednost 0.

Poljem *cc* se određuje da li treba produžiti sa sekvencijalnim izvršavanjem mikroprograma ili treba realizovati bezuslovni ili uslovni skok u mikroprogramu. U slučajevima kada treba realizovati uslovni skok u mikroprogramu vrši se provera da li određeni signal logičkog uslova ima vrednost 1 ili 0 i na osnovu toga se ili realizuje skok u mikroprogramu ili produžava sa sekvencijalnim izvršavanjem mikroprograma, respektivno. Stoga se posebne vrednosti polja *cc* pridružuju za svaki od mogućih slučajeva izvršavanja mikroprograma i to za slučaj kada treba produžiti sa sekvencijalnim izvršavanjem mikroprograma, zatim za slučaj kada treba realizovati bezuslovni skok u mikroprogramu, kao i za svaki od mogućih slučajeva uslovnog skoka kada se vrednošću polja *cc* određuje signal logičkog uslova na osnovu čije vrednosti se ili skače u mikroprogramu ili produžava sa sekvencijalnim izvršavanjem mikroprograma. Dužina polja *cc* u bitovima treba da bude takva da se njima može kodirati posebna binarna vrednost za svaku od ovih situacija.

Poljem *ba* se određuje adresa mikroinstrukcije u mikroprogramu na koju se skače u slučaju da je vrednošću polja *cc* predviđeno da se bezuslovano skače ili da signal logičkog uslova određen vrednošću polja *cc* ima vrednost 1. Dužina polja *ba* u bitovima treba da bude takva da se njome može realizovati skok u bilo koju adresu mikroprogramske memorije.

Polja *cc* i *ba* upravljačkog dela mikroinstrukcije se formiraju ukoliko u datom koraku ima iskaza za bezuslovni skok ili uslovni skok. U suprotnom slučaju se svi bitovi upravljačkog dela postavljaju na vrednost 0, čime se određuje da treba produžiti sa sekvencijalnim izvršavanjem mikroprograma.

Usvojena struktura mikroinstrukcije za jedinicu sa jednom operacijom je data na slici 23.

Bit 0 operacionog dela mikroinstrukcije se ne koristi. Bitovi 1 do 21 operacionog dela mikroinstrukcije bili dodeljeni polju *z* i to bit 1 za signal ldBB, bit 2 za signal ldMQ i tako redom do bita 21 za signal $\mathrm{stOWU_{EU}}$, pri čemu se ovi bitovi dodeljuju upravljačkim signalima operacione jedinice na proizvoljan način. Bitovi 22 i 23 se, takođe, ne koriste, a uključeni su da bi, time što je broj bitova mikroinstrukcije deljiv sa četiri, mikroprogram napisan u heksadecimalnom sistemu bio pregledniji.

0	1	2	3	4	5	6	7
0	ldBB	ldMQ	srMQ	M	S_1	S_0	C_0

8	9	10	11	12	13	14	15
ldISC	clAB8	srAB8	ldAB8	decISC	clAB	srAB	ldAB

16	17	18	19	20	21	22	23
ABout	$ldABWU_{EU}$	MQout	$ldMQWU_{EU}$	clOEU	$stOWU_{EU}$	0	0

24	25	26	27	28	29	30	31
cc				*ba*			

Slika 23 Mikroinstrukcija za upravljačku jedinicu sa jednim tipom mikroinstrukcije za jedinicu sa jednom operacijom

Bitovi 24 do 27 upravljačke mikroinstrukcije su dodeljeni polju *cc* koje je kodirano na sledeći način: 0000 – sekvencijalno izvršavanja mikroprograma, 0001 – bezuslovni skok u mikroprogramu na adresu određenu poljem *ba*, 0010 – uslovni skok u mikroprogramu na adresu određenu poljem *ba* ukoliko je signal logičkog uslova OEU nula, 0011 – uslovni skok u mikroprogramu na adresu određenu poljem *ba* ukoliko je signal logičkog uslova MQ_0 nula i 0100 – uslovni u mikroprogramu skok na adresu određenu poljem *ba* ukoliko je signal logičkog uslova ISCZ nula. Ukoliko se javi neka od vrednosti 0101 do 1111 realizuje se sekvencijalno izvršavanja mikroprograma. Za kodiranje polja *cc* su dovoljna tri bita, ali je usvojeno da dužina bude četiri bita da bi mikroprogram napisan u heksadecimalnom sistemu bio pregledniji.

Bitovi 28 do 31 upravljačkog dela mikroinstrukcije su dodeljeni polju *ba*. U sekvenci upravljačkih signala po koracima postoji 8 koraka, pa će se i mikroprogram sastojati iz 8 mikroinstrukcija. Za kodiranje polja *ba* su dovoljna tri bita, ali je usvojeno da dužina bude četiri bita da bi mikroprogram napisan u heksadecimalnom sistemu bio pregledniji.

Na osnovu usvojenih dužina polja *z*, *cc* i *ba*, dužina mikroinstrukcije je 32 bita.

Sa usvojenim formatom mikroinstrukcije (slika 23) se za sekvencu upravljačkih signala po koracima (slika 1) za jedinicu sa jednom operacijom formira mikroprogram (tabela 10).

Kod formiranja operacionog dela mikroinstrukcije bitovi ovog dela mikroinstrukcije koji odgovaraju upravljačkim signalima operacione jedinice koji se javljaju u datom koraku postavljaju se na 1, dok se bitovi ovog dela mikroinstrukcije koji odgovaraju upravljačkim signalima operacione jedinice 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* $step_A$). Bezuslovni skokovi se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 1) u kojima se pojavljuju iskazi tipa *br* $step_A$. Uslovni skokovi se realizuju u onim koracima sekvence upravljačkih signala po koracima u kojima se pojavljuju iskazi tipa *br* (*if* **uslov** *then* $step_A$). Za korake u kojima se ovi iskazi javljaju, bitovi polja *cc* i *ba* se kodiraju u zavisnosti od toga koji se od ova dva 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 6.

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 6 Signal bezuslovnog skoka

signal bezuslovnog skoka	*cc*
bruncnd	1

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
$step_7$	$step_0$	$\mathbf{madr_0}$	0

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 8. 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 9.

Tabela 8 Signali uslovnih skokova

signal uslovnog skoka	polje *cc*	signal uslova
brnotOEU	2	$\overline{\mathbf{OEU}}$
brnotMQ0	3	$\overline{\mathbf{MQ_0}}$
brnotISCZ	4	$\overline{\mathbf{ISCZ}}$

Tabela 9 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
$step_0$	$\overline{\mathbf{OEU}}$	$step_{00}$	$\mathbf{madr_0}$	0
$step_2$	$\overline{\mathbf{MQ_0}}$	$step_{04}$	$\mathbf{madr_4}$	4
$step_5$	$\overline{\mathbf{ISCZ}}$	$step_{02}$	$\mathbf{madr_2}$	2

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu mikroprogramske realizacije signal bezuslovnog skoka (tabela 6) i signali uslovnih skokova (tabela 8), i signali vrednosti A za bezuslovne skokokve (tabela 7) i uslovne skokove (tabela 9).

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 1) formiran mikroprogram (tabela 10). 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 10 Mikroprogram za upravljačku jedinicu sa jednim tipom mikroinstrukcije za jedinicu sa jednom operacijom

0	000000	2	0	**! brnotOEU**, $madr_0$ **!**
1	60C400	0	0	**! ldBB, ldMQ, clAB8, clAB, ldISC** !
2	000000	3	4	! **brnotMQ0**, $madr_4$!
3	041100	0	0	**! S_1**, ldAB, ldAB8!
4	102A10	0	0	**! srAB8**, srAB, srMQ, decISC !
5	000000	4	2	! **brnotISCZ**, $madr_2$!
6	0000C0	0	0	**!** ABout, **ldABWU**$_{FII}$!
7	00003C	1	0	**!** MQout, **ldMQWU**$_{FII}$, clOEU, stOWU$_{FII}$ **, bruncnd**, **$madr_0$**!

U kolonama tabele su dati adrese mikroprogramske memorije, vrednosti polja *z*, *cc* i *ba* mikroinstrukcije i komentar. Adrese mikroprogramske memorije i vrednosti polja *z*, *cc* i *ba* mikroinstrukcije su dati u heksadecimalnom obliku. U komentaru za operacioni deo mikroinstrukcije pojavljuju se samo upravljački signali operacione jedinice koji imaju vrednost 1, dok signala koji imaju vrednost 0 nema. U komentaru za upravljački deo mikroinstrukcije pojavljuje se upravljački signal upravljačke jedinice koji ima vrednost 1 i simbolička oznaka vrednosti koja se upisuje u mikroprogramski brojač u situacijama kada se realizuje bezuslovni ili uslovni skok, dok ovih signala nema u situacijama kada nema ovih skokova.

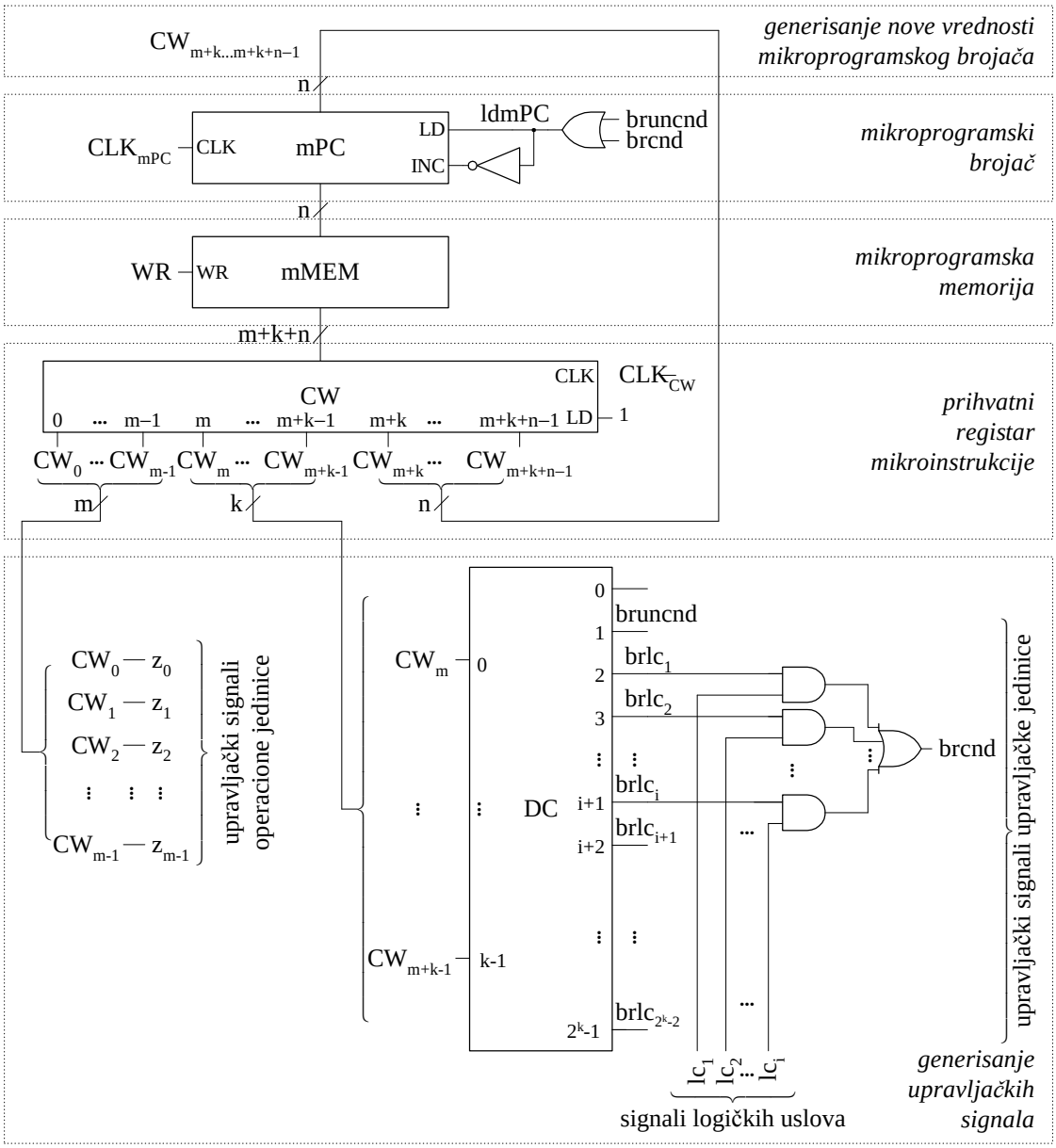
1.2.2.2.1.2 Strukturna šema upravljačke jedinice

Upravljačku jedinicu (slika 24) u opštem slučaju čine mikroprogramski brojač mPC, mikroprogramska memorija mMEM, prihvatni registar mikroinstrukcije CW, kombinaciona mreža za generisanje upravljačkih signala i kombinaciona mreže za generisanje nove vrednosti mikroprogramskog brojača.

Mikroprogramski brojač mPC određuje adresu mikroinstrukcije u mikroprogramskoj memoriji. Sadržaj mikroprogramskog brojača mPC se može ili inkrementirati ili se u mikroprogramski brojač mPC može upisivati nova vrednost i time realizovati skok u mikroprogramskoj memoriji. Vrednost 0 signala ldmPC određuje da sadržaj mikroprogramskog brojača mPC treba inkrementirati, dok vrednost 1 određuje da u njega treba upisati novu vrednost. Signal ldmPC ima vrednost 0 ili 1 u zavisnosti od sadržaja razreda $CW_{m...m+k-1}$ prihvatnog registra mikroinstrukcije CW koji predstavlju polje *cc* upravljačkog dela mikroinstrukcije i signala logičkih uslova. Vrednost koja se upisuje određena je razredima $CW_{m+k...m+k+n-1}$ prihvatnog registra mikroinstrukcije CW i predstavlja polje *ba* upravljačkog dela mikroinstrukcije. Mikroprogramski brojač ima n razreda uz pretpostavku da je kapacitet mikroprogramske memorije 2^n reči.

Mikroprogramska memorije mMEM služi za smeštanje mikroprograma. Kapacitet mikroprogramske memorije je određen veličinom mikroprograma i iznosi 2^n reči. Širina reči mikroprogramske memorije određena je brojem bitova operacionog i upravljačkog dela mikroinstrukcije. Uz pretpostavku da je operacioni deo mikroinstrukcije m bita i upravljački deo k+n bita, širina reči mikroprogramske memorije je m+k+n bita.

Prihvatni registar mikroinstrukcije CW služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM sa adrese određene sadržajem mikroprogramskog brojača mPC. U razrede 0 do m-1 se smeštaju bitovi polja *z* operacionog dela mikroinstrukcije, dok se u razrede m do m+k-1 i m+k do m+k+n-1 smeštaju bitovi polja *cc* i *ba*, respektivno, upravljačkog dela mikroinstrukcije.



Slika 24 Upravljačka jedinica sa jednim tipom mikroinstrukcije

Kombinaciona mreža za generisanje upravljačkih signala generiše dve grupe signala i to upravljačke signale operacione jedinice i upravljački signal upravljačke jedinice.

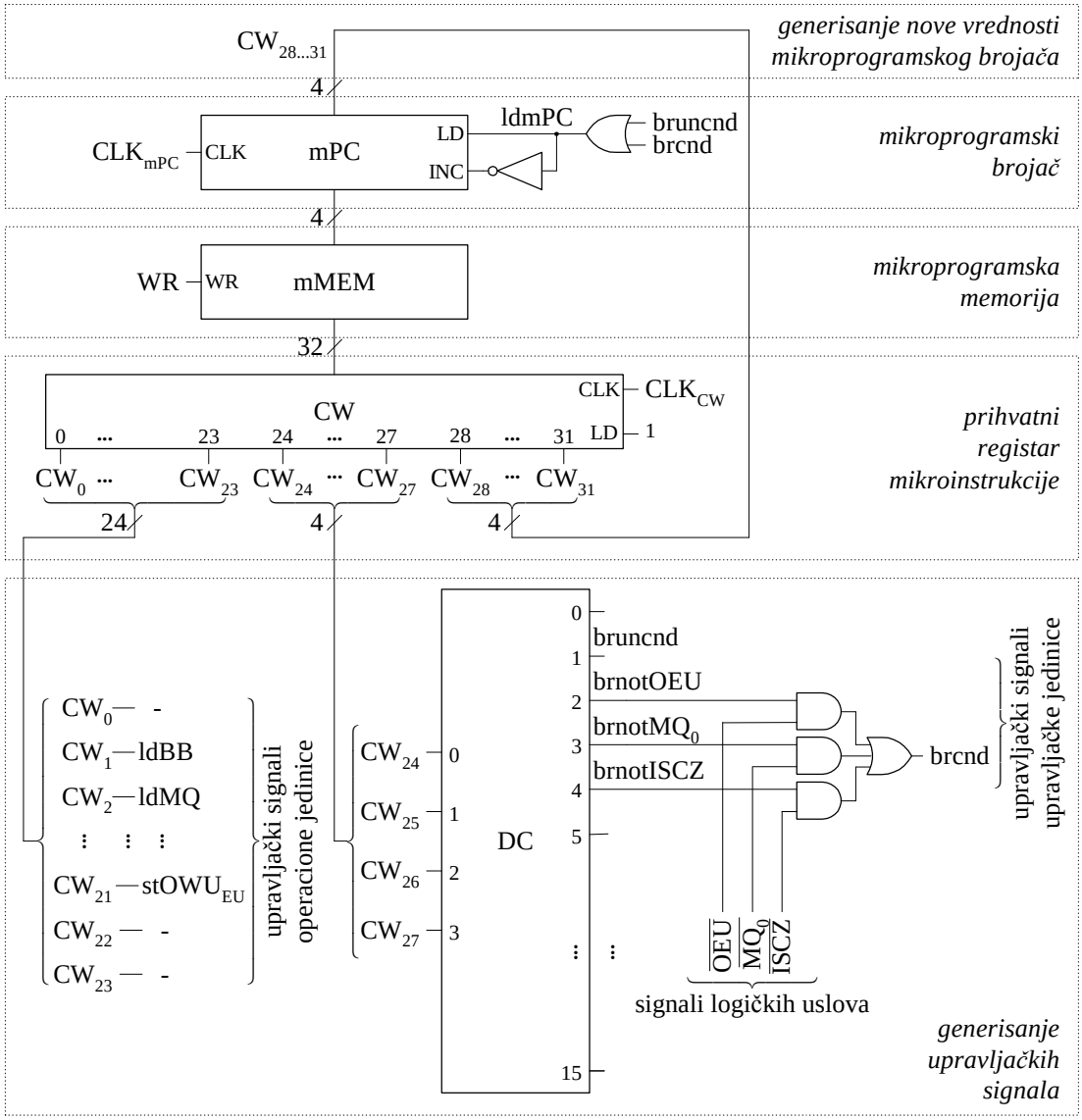
Upravljački signali operacione jedinice z_0 do z_{m-1} se generišu na osnovu sadržaja razreda CW_0 do CW_{m-1}.

Upravljački signali upravljačke jedinice su bruncnd, brcnd i $brlc_1$ do $brlc_i$. Ukoliko signali bruncnd i brcnd imaju vrednost 0 i signal mPC ima vrednost 0, pa se sadržaj mPC inkrementira. Ukoliko signal bruncnd ili signal brcnd ima vrednost 1 i signal mPC ima vrednost 1, pa se u mPC upisuje sadržaj razreda CW_{m+k} do $CW_{m+k+n-1}$. Signali bruncnd i brcnd imaju vrednost 0 ukoliko je u razredima CW_m do CW_{m+k-1} vrednost 0 koja je pridružena situaciji kada treba produžiti sa sekvencijalnim izvršavanjem mikroprograma ili neka od vrednosti i+2 do 2^k-1 koje su ostale neiskorišćene. Signal bruncnd ima vrednost 1 ukoliko je u razredima CW_m do CW_{m+k-1} vrednost 1 koja je pridružena situaciji kada treba napraviti bezuslovni skok u mikroprogramu, dok u svim ostalim situacijama ime vrednost 0. Signal brcnd ima vrednost 1 ukoliko je u razredima CW_m do CW_{m+k-1} neka od vrednost 2 do i+1

kojima se kodiraju signali uslovnog skoka $brlc_1$ do $brlc_i$ pridruženi signalima logičkih uslova lc_1 do lc_i, respektivno, i odgovarajući signal logičkog uslovi ima vrednost 1, dok u svim ostalim situacijama ime vrednost 0.

Kombinaciona mreža za generisanje nove vrednosti mikroprogramskog brojača se sastoji samo od linija po kojima na paralelne ulaze mikroprogramskog brojača mPC dolaze razredi CW_{m+k} do $CW_{m+k+n-1}$ prihvatnog registra mikroinstrukcije CW. Ovaj sadržaj se upisuje u mikroprogramski brojač ukoliko signal ldmPC ima vrednost 1.

Strukturnu šemu upravljačke jedinice za jedinicu sa jednom operacijom (slika 25) čine: mikroprogramski brojač $mPC_{3..0}$, mikroprogramska memorija mMEM, prihvatni registar mikroinstrukcije $CW_{0..31}$, kombinaciona mreža za generisanje upravljačkih signala i kombinaciona mreža za generisanje nove vrednosti mikroprogramskog brojača.



Slika 25 Upravljačka jedinica sa jednim tipom mikroinstrukcije
za jedinicu sa jednom operacijom

Mikroprogramski brojač $mPC_{3..0}$ ima 4 razreda i njime može da se vrši adresiranje mikroprogramske memorije u opsegu 0 do 15. Sadržaj mikroprogramskog brojača mPC se vrednošću 0 signala ldmPC inkrementira, dok se i mikroprogramski brojač mPC vrednošću 1 signala ldmPC upisuje nova vrednost iz CW_{28} do CW_{31}.

Mikroprogramska memorije mMEM može da sadrži mikroprogram maksimalne veličine 16 mikroinstrukcija širine 32 bita, pri čemu su popunjene samo lokacije na adresam 0 do 7.

Prihvatni registar mikroinstrukcije $CW_{0..31}$ ima 32 razreda u skladu sa usvojenom dužinom mikroinstrukcije od 32 bita.

Kombinaciona mreža za generisanje upravljačkih signala generiše dve grupe signala i to upravljačke signale operacione jedinice i upravljački signal upravljačke jedinice.

Upravljački signali operacione jedinice ldBB do $stOWU_{EU}$ se generišu na osnovu sadržaja razreda CW_1 do CW_{21}.

Upravljački signali upravljačke jedinice bruncnd, brcnd, brnotOEU, brnotMQ0 i brnotISCZ se generiše na osnovu sadržaja razreda CW_{24} do CW_{27} i signala logičkih uslova OEU, MQ_0 i ISCZ. Signali bruncnd, brcnd, brnotOEU, brnotMQ0 i brnotISCZ, a time i signal ldmPC, imaju vrednost 0 ukoliko je binarna vrednost razreda CW_{24} do CW_{27} jednaka 0000. Signal bruncnd, a time i signal ldmPC, ima vrednost 1 ukoliko je binarna vrednost razreda CW_{24} do CW_{27} jednaka 0001. Ukoliko je binarna vrednost razreda CW_{24} do CW_{27} jednaka 0010, 0011 ili 0100, jedan od signala brnotOEU, brnotMQ0 i brnotISCZ ima vrednost 1, pa signala brcnd, a time i signal ldmPC, ima vrednost 0 ili 1 u zavisnosti od toga da li odgovarajući signal logičkog uslova OEU, MQ_0 i ISCZ ima vrednost 1 ili 0, respektivno. Signali bruncnd, brcnd, brnotOEU, brnotMQ0 i brnotISCZ, a time i signal ldmPC, imaju vrednost 0 ukoliko je binarna vrednost razreda CW_{24} do CW_{27} jednaka 0101 do 1111.

Kombinaciona mreža za generisanje nove vrednosti mikroprogramskog brojača se sastoji samo od linija po kojima na paralelne ulaze mikroprogramskog brojača mPC dolaze razredi CW_{28} do CW_{31} prihvatnog registra mikroinstrukcije CW. Ovaj sadržaj se upisuje u mikroprogramski brojač ukoliko signal ldmPC ima vrednost 1.

1.2.2.2.1.3 Funkcionisanje upravljačke jedinice

Funkcionisanje upravljačke jedinice se u opštem slučaju odvija po postupku datom u daljem tekstu. Mikroprogramski brojaču mPC, čiji početni sadržaj ukazuje na početnu adresu mikroprograma operacije za koju treba generisati upravljačke signale, koristi se kao adresa mikroprogramske memorije iz koje se čita mikroinstrukcija i smešta u prihvatni registar mikroinstrukcije CW. Operacioni deo mikroinstrukcija iz razreda CW_0 do CW_{m-1} i upravljački deo mikroinstrukcije iz razreda CW_m do CW_{m+k-1} i CW_{m+k} do $CW_{m+k+n-1}$ se koriste nezavisno za generisanje upravljačkih signala operacione jedinice z_0 do z_{m-1} i upravljačkih signala upravljačke jedinice bruncnd, brcnd i $brlc_1$ do $brlc_i$, respektivno.

Razredi CW_0 do CW_{m-1} predstavljaju bitove polja *z* operacionog dela mikroinstrukcije, pa se na osnovu sadržaja razreda CW_0 do CW_{m-1} generišu vrednosti 0 i 1 upravljačkih signala operacione jedinice z_0 do z_{m-1}. S obzirom na to da je svakom od *m* upravljačkih signala operacione jedinice dodeljen jedan od m bitova operacionog dela mikroinstrukcije koji se nalaze u jednom od razreda CW_0 do CW_{m-1} odgovarajući upravljački signal operacione jedinice imaće vrednost 0 ili 1 u zavisnosti od toga da li se u odgovarajućem razredu nalazi vrednost 0 ili 1, respektivno.

Razredi CW_m do CW_{m+k-1} i CW_{m+k} do $CW_{m+k+n-1}$ predstavljaju bitove polja *cc* i *ba* upravljačkog dela mikroinstrukcije, respektivno, pa se na osnovu sadržaja razreda CW_m do CW_{m+k-1}, a u nekim situacijama i na osnovu vrednosti signala logičkih uslova lc_1 do lc_i, generišu vrednost 0 ili 1 upravljačkih signala upravljačke jedinice bruncnd, brcnd i $brlc_1$ do $brlc_i$. Ukoliko je binarna vrednost razreda CW_m do CW_{m+k-1} jednaka 0, signali bruncnd, brcnd i $brlc_1$ do $brlc_i$, a time i signal mPC, imaju vrednost 0, pa se mikroprogramski brojač mPC

inkrementira. Time se vrednošću 0 polja *cc* upravljačkog dela mikroinstrukcije realizuje sekvencijalno izvršavanje mikroprograma. Ukoliko je binarna vrednost razreda CW_m do CW_{m+k-1} jednaka 1, signal bruncnd, a time i signal mPC, ima vrednost 1, pa se u mikroprogramski brojač mPC upisuje sadržaj razreda CW_{m+k} do $CW_{m+k+n-1}$. Time se vrednošću 1 polja *cc* upravljačkog dela mikroinstrukcije realizuju bezuslovni skok u mikroprogramu.

Ukoliko je binarna vrednost razreda CW_m do CW_{m+k-1} u opsegu vrednosti 2 do i+1, jedan od signala uslovnih skokova $brlc_1$ do $brlc_i$ pridružen jednom od signala logičkih uslova lc_1 do lc_i, respektivno, ima vrednost 1, pa signal brcnd, a time i signal ldmPC, ima vrednost 0 ili 1 u zavisnosti od toga da li odgovarajući signal logičkog uslova lc_1 do lc_i ima vrednost 0 ili 1, respektivno. Ukoliko signal ldmPC ima vrednost 0, mikroprogramski brojač mPC se inkrementira, pa se produžava sa sekvencijalnim izvršavanjem mikroprograma. Ukoliko signal ldmPC ima vrednost 1, u mikroprogramski brojač mPC se upisuje sadržaj razreda CW_{m+k} do $CW_{m+k+n-1}$, pa se realizuje skok u mikroprogramu. Time se vrednostima 2 do i+1 polja *cc* upravljačkog dela mikroinstrukcije, a na osnovu vrednosti odgovarajućih signala logičkih uslova lc_1 do lc_i, realizuje uslovni skok u mikroprogramu. Ukoliko je binarna vrednost razreda CW_m do CW_{m+k-1} u opsegu vrednosti i+2 do 2^k-1, signali bruncnd, brcnd i $brlc_1$ do $brlc_i$, a time i signal mPC, imaju vrednost 0, pa se mikroprogramski brojač mPC inkrementira. Time se vrednošću i+2 do 2^k-1 polja *cc* upravljačkog dela mikroinstrukcije, koje nisu pridružene ni jednom signalu logičkih uslova, realizuje sekvencijalno izvršavanje mikroprograma.

Funkcionisanje upravljačke jedinice za jedinicu sa jednom operaciojom se odvija po po postupku datom u daljem tekstu.

Sa adrese 0000 mikroprogramske memorije čita se mikroinstrukcija. Bitovi 1 do 21 polja *z* mikroinstrukcije imaju vrednost 0, pa svi upravljački signali operacione jedinice imaju vrednost 0. Bitovi 24 do 27 polja *cc* mikroinstrukcije imaju vrednost 0010. Time se određuje da treba proveriti vrednost signala logičkog uslova OEU i ukoliko ima vrednost 0 upisati bitove 28 do 31 polja *ba* mikroinstrukcije koji imaju vrednost 0000 u mikroprogramski brojač mPC, dok u suprotnom slučaju treba inkrementirati mikroprogramski brojač mPC.

Sa adrese 0001 mikroprogramske memorije čita se mikroinstrukcija. Bitovi 1, 2, 8, 9 i 13 polja *z* mikroinstrukcije imaju vrednost 1 dok ostali bitovi imaju vrednost 0. Time se određuje da upravljački signali operacione jedinice ldBB, ldMQ, clAB8, clAB i ldISC treba da imaju vrednost 1, dok svi ostali upravljački signali operacione jedinice vrednost 0. Bitovi 24 do 27 polja *cc* mikroinstrukcije imaju vrednost 0000, pa mikroprogramski brojač mPC treba inkrementirati.

Sa adrese 0010 mikroprogramske memorije čita se mikroinstrukcija. Bitovi 1 do 21 polja *z* mikroinstrukcije imaju vrednost 0, pa svi upravljački signali operacione jedinice imaju vrednost 0. Bitovi 24 do 27 polja *cc* mikroinstrukcije imaju vrednost 0011. Time se određuje da treba proveriti vrednost signala logičkog uslova MQ_0 i ukoliko ima vrednost 0 upisati bitove 28 do 31 polja *ba* mikroinstrukcije koji imaju vrednost 0100 u mikroprogramski brojač mPC, dok u suprotnom slučaju treba inkrementirati mikroprogramski brojač mPC.

Sa adrese 0011 mikroprogramske memorije čita se mikroinstrukcija. Bitovi 5, 11 i 15 polja *z* mikroinstrukcije imaju vrednost 1 dok ostali bitovi imaju vrednost 0. Time se određuje da upravljački signali operacione jedinice S_1, ldAB i ldAB8 treba da imaju vrednost 1, dok svi ostali upravljački signali operacione jedinice vrednost 0. Bitovi 24 do 27 polja *cc* mikroinstrukcije imaju vrednost 0000, pa mikroprogramski brojač mPC treba inkrementirati.

Sa adrese 0100 mikroprogramske memorije čita se mikroinstrukcija. Bitovi 3, 10, 12 i 14 polja *z* mikroinstrukcije imaju vrednost 1, dok ostali bitovi imaju vrednost 0. Time se određuje da upravljački signali operacione jedinice srMQ, srAB8, decISC i srAB treba da imaju vrednost 1, dok svi ostali upravljački signali operacione jedinice vrednost 0. Bitovi 24 do 27 polja *cc* mikroinstrukcije imaju vrednost 0000, pa mikroprogramski brojač mPC treba inkrementirati.

Sa adrese 0101 mikroprogramske memorije čita se mikroinstrukcija. Bitovi 1 do 21 polja *z* mikroinstrukcije imaju vrednost 0, pa svi upravljački signali operacione jedinice imaju vrednost 0. Bitovi 24 do 27 polja *cc* mikroinstrukcije imaju vrednost 0100. Time se određuje da treba proveriti vrednost signala logičkog uslova ISCZ i ukoliko ima vrednost 0 upisati bitove 28 do 31 polja *ba* mikroinstrukcije koji imaju vrednost 0010 u mikroprogramski brojač mPC, dok u suprotnom slučaju treba inkrementirati mikroprogramski brojač mPC.

Sa adrese 0110 mikroprogramske memorije čita se mikroinstrukcija. Bitovi 16 i 17 polja *z* mikroinstrukcije imaju vrednost 1, dok ostali bitovi imaju vrednost 0. Time se određuje da upravljački signali operacione jedinice ABout i $\text{ldABWU}_{\text{EU}}$ treba da imaju vrednost 1, dok svi ostali upravljački signali operacione jedinice vrednost 0. Bitovi 24 do 27 polja *cc* mikroinstrukcije imaju vrednost 0000, pa mikroprogramski brojač mPC treba inkrementirati..

Sa adrese 0111 mikroprogramske memorije čita se mikroinstrukcija. Bitovi 18, 19, 20 i 21 polja *z* mikroinstrukcije imaju vrednost 1, dok ostali bitovi imaju vrednost 0. Time se određuje da upravljački signali operacione jedinice MQout, $\text{ldMQWU}_{\text{EU}}$, clOEU i stOWU_{EU} treba da imaju vrednost 1, dok svi ostali upravljački signali operacione jedinice vrednost 0. Bitovi 24 do 27 polja *cc* mikroinstrukcije imaju vrednost 0001. Time se određuje da treba bezuslovno upisati bitove 28 do 31 polja *ba* mikroinstrukcije koji imaju vrednost 0000 u mikroprogramski brojač mPC. Ovim se realizuje povratak na početak mikroprograma.

1.2.2.2.2 DVA TIPA MIKROINSTRUKCIJA

U ovom poglavlju se daje najpre postupak formiranja mikroprograma, zatim postupak realizacije strukturne šeme upravljačke jedinice i na kraju funkcionisanje upravljačke jedinice.

1.2.2.2.2.1 Formiranje mikroprograma

Upravljačka jedinica generiše dve vrste upravljačkih signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice. Upravljački signali operacione jedinice se koriste u operacionoj jedinici radi izvršavanja mikrooperacija. Upravljački signali upravljačke jedinice se koriste u upravljačkoj jedinici radi inkrementiranja mikroprogramskog brojača ili upisa nove vrednosti u mikroprogramski brojač i radi generisanja vrednosti za upis u mikroprogramski brojač.

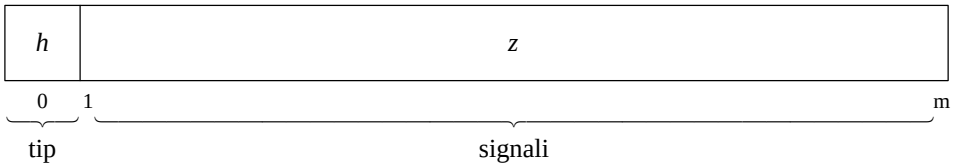
Za generisanje upravljačkih signala koriste se dva tipa mikroinstrukcija i to operaciona mikroinstrukcija za generisanje upravljačkih signala operacione jedinice i upravljačka mikroinstrukcija za generisnje upravljačkih signala upravljačke jedinice. Pri tome struktura operacione mikroinstrukcije odgovara strukturi opracionog dela mikroinstrukcije upravljačkih jedinica sa jednim tipom mikroinstrukcije, dok struktura upravljačke mikroinstrukcije odgovara strukturi upravljačkog dela mikroinstrukcije upravljačkih jedinica sa jednim tipom mikroinstrukcije. Da bi mogao da se formira mikroprogram u kome se pojavljuju posebno operacione i posebno upravljačke mikroinstrukcije potrebno je da se modifikuje sekvenca upravljačkih signala po koracima (tabela 1). Svaki korak u kome se samo generišu upravljački signali operacione jedinice i ne realizuje skok, ostaje i u modifikovanoj sekvenci upravljačkih signala i u mikroprogramu se predstavlja jednom operacionom mikroinstrukcijom. Svaki korak u kome se ne generišu upravljački signali operacione jedinice već samo realizuje skok,

ostaje i u modifikovanoj sekvenci upravljačkih signala i u mikroprogramu se predstavlja jednom upravljačkom mikroinstrukcijom. Svaki korak u kome se ne samo generišu upravljački signali operacione jedinice već realizuje i skok, u modifikovanoj sekvenci upravljačkih signala se zamenjuje sa dva posebna koraka i to jednim korakom za generisanje upravljačkih signala operacione jedinice i jednim korakom za realizaciju skoka i u mikroprogramu se predstavlja sa dve mikroinstrukcije i to jednom operacionom mikroinstrukcijom i jednom upravljačkom mikroinstrukcijom. Zbog toga je potrebno od originalne sekvence upravljačkih signala po koracima (tabela 1) formirati modifikovanu sekvencu upravljačkih signala po koracima (tabela 11).

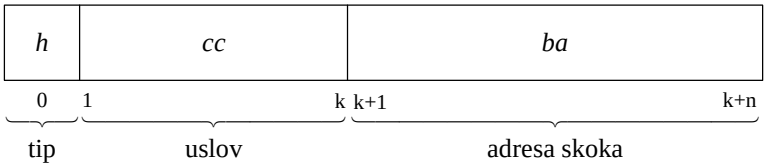
Tabela 11 Modifikovana sekvenca upravljačkih signala po koracima za jedinicu sa jednom operacijom

$step_0$ *br* (*if* $\overline{\textbf{OEU}}$ *then* $step_0$);
$step_1$ **ldBB, ldMQ, clAB8, clAB, ldISC**;
$step_2$ *br* (*if* $\overline{\textbf{MQ}_\textbf{0}}$ *then* $step_4$);
$step_3$ $\textbf{S}_\textbf{1}$, ldAB, ldAB8;
$step_4$ **srAB8**, srAB, srMQ, decISC;
$step_5$ *br* (*if* $\overline{\textbf{ISCZ}}$ *then* $step_2$);
$step_6$ ABout, $ldABWU_{EU}$;
$step_7$ MQout, $ldMQWU_{EU}$, clOEU, $stOWU_{EU}$;
$step_8$ *br* $step_0$;

Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu modifikovane sekvence upravljačkih signala po koracima. Mikroprogram se formira tako što se u sekvenci upravljačkih signala po koracima svakom koraku u kome su navedeni upravljački signali operacione jedinice koji treba da imaju vrednost 1 pridružuje operaciona mikroinstrukcija (slika 26) i svakom koraku u kome su realizuju skokovi pridružuje upravljačka mikroinstrukcija (slika 27).



Slika 26 Operaciona mikroinstrukcija za upravljačku jedinicu sa dva tipa mikroinstrukcija



Slika 27 Upravljačka mikroinstrukcija za upravljačku jedinicu sa dva tipa mikroinstrukcija

Operacione i upravljačke mikroinstrukcije imaju polje *h* dužine 1 bit koje vrednostim 0 i 1 određuje da li se radi o operacionoj ili upravljačkoj mikroinstrukciji, respektivno. Operacione mikroinstrukcije imaju polje *z* dužine m bitova, čije je značenje identično značenju bitova polja z operacionog dela mikroinstrukcije za slučaj upravljačke jedinice sa jednim tipom mikroinstrukcije. Upravljačke mikroinstrukcije imaju polja *cc* i *ba* dužine k i n bitova, respektivno, čije je značenje identično značenju bitova polja *cc* i *ba* upravljačkog dela mikroinstrukcije za slučaj upravljačke jedinice sa jednim tipom mikroinstrukcije.

Usvojena struktura operacione mikroinstrukcije za jedinicu sa jednom operacijom je data na slici 28.

0	1	2	3	4	5	6	7
0	ldBB	ldMQ	srMQ	M	S_1	S_0	C_0

8	9	10	11	12	13	14	15
ldISC	clAB8	srAB8	ldAB8	decISC	clAB	srAB	ldAB

16	17	18	19	20	21	22	23
ABout	$ldABWU_{EU}$	MQout	$ldMQWU_{EU}$	clOEU	$stOWU_{EU}$	0	0

Slika 28 Operaciona mikroinstrukcija za upravljačku jedinicu sa dva tipa mikroinstrukcija za jedinicu sa jednom operacijom

Bit 0 operacione mikroinstrukcije dodeljen je polju *h* i za operacionu mikroinstrukciju ima vrednost 0. Bitovi 1 do 23 operacione mikroinstrukcije dodeljeni su polju *z* i imaju identično značenje kao bitovi 1 do 23 operacionog dela mikroinstrukcije za slučaj upravljačke jedinice sa jednim tipom mikroinstrukcije.

Usvojena struktura upravljačke mikroinstrukcije za za jedinicu sa jednom operacijom je data na slici 29.

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

8	9	10	11	12	13	14	15
ba				0	0	0	0

16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0

Slika 29 Upravljačka mikroinstrukcija za upravljačku jedinicu sa dva tipa mikroinstrukcija za jedinicu sa jednom operacijom

Bit 0 upravljačke mikroinstrukcije dodeljen je polju *h* i za upravljačku mikroinstrukciju ima vrednost 1. Bitovi 1, 2 i 3 upravljačke mikroinstrukcije se ne koriste. Bitovi 4 do 7 i 8 do 11 upravljačke mikroinstrukcije su dodeljeni poljima *cc* i *ba*, respektivno, i imaju identično značenju bitova polja *cc* i *ba* upravljačkog dela mikroinstrukcije za slučaj upravljačke jedinice sa jednim tipom mikroinstrukcije. Polja *h*, *cc* i *ba* upravljačke mikroinstrukcije počinju na granici od četiri bita, da bi mikroprogram napisan u heksadecimalnom sistemu bio pregledniji.

Operaciona mikroinstrukcija je duža od upravljačke mikroinstrukcije, pa je dužina mikroinstrukcija, a time i širina reči mikroprogramske memorije, određena dužinom operacione mikroinstrukcije i iznosi 24 bita. Stoga se bitovi 12 do 23 upravljačke mikroinstrukcije se ne koriste.

Sa usvojenim formatom mikroinstrukcija (slike 28 i 29) se za modifikovanu sekvencu upravljačkih signala po koracima (slika 11) za jedinicu sa jednom operacijom formira mikroprogram (tabela 16). Mikroprogram se formira tako što se za svaki korak u modifikovanoj sekvenci upravljačkih signala po koracima (tabela 11) formira jedna mikroinstrukcija. Operacione mikroinstrukcije se formiraju ukoliko u datom koraku ima

upravljačkih signala operacione jedinice. Upravljačka mikroinstrukcija se formira ukoliko u datom koraku ima iskaza za bezuslovni skok, uslovni skok ili višestruki uslovni skok.

Kod formiranja operacione mikroinstrukcije bitovi 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 upravljačke mikroinstrukcije za dati korak se proverava da li se javlja neki od iskaza *br* $step_A$ i *br* (*if* **uslov** *then* $step_A$). Bezuslovni skokovi se realizuje u onim koracima modifikovane sekvence upravljačkih signala po koracima (tabela 11) u kojima se pojavljuju iskazi tipa *br* $step_A$. Uslovni skokovi se realizuju u onim koracima modifikovane sekvence upravljačkih signala po koracima u kojima se pojavljuju iskazi tipa *br* (*if* **uslov** *then* $step_A$). Za korake u kojima se ovi iskazi javljaju, bitovi polja *cc* i *ba* se kodiraju u zavisnosti od toga koji se od ova dva iskaza javlja u datom koraku.

Za iskaz *br* $step_A$ se upravljačka mikroinstrukcija kodira tako što se za polje *cc* uzima kod dodeljen signalu bezuslovnog skoka koji određuje da se bezuslovno 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 12. 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 13.

Tabela 12 Signal bezuslovnog skoka

signal bezuslovnog skoka	*cc*
bruncnd	1

Tabela 13 Koraci $step_i$, $step_A$, vrednosti $\mathbf{madr_A}$ i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	$\mathbf{madr_A}$	A
$step_8$	$step_0$	$\mathbf{madr_0}$	0

Za iskaz *br* (*if* **uslov** *then* $step_A$) se upravljačka mikroinstrukcija kodira tako što se za polje *cc* uzima kod dodeljen signalu uslovnog skoka koji određuje signal **uslov** koji treba da 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 modifikovanoj 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 15.

Tabela 14 Signali uslovnih skokova

signal uslovnog skoka	polje *cc*	signal uslova
brnotOEU	2	$\overline{\mathbf{OEU}}$
brnotMQ0	3	$\overline{\mathbf{MQ_0}}$
brnotISCZ	4	$\overline{\mathbf{ISCZ}}$

Tabela 15 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
$step_0$	$\overline{\mathbf{OEU}}$	$step_{00}$	$\mathbf{madr_0}$	0
$step_2$	$\overline{\mathbf{MQ_0}}$	$step_{04}$	$\mathbf{madr_4}$	4
$step_5$	$\overline{\mathbf{ISCZ}}$	$step_{02}$	$\mathbf{madr_2}$	2

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu mikroprogramske realizacije signal bezuslovnog skoka (tabela 12) i signali uslovnih skokova (tabela 14), i signali vrednosti A za bezuslovne skokokve (tabela 13) i uslovne skokove (tabela 15).

Po opisanom postupku je, na osnovu modifikovane sekvence upravljačkih signala po koracima (tabela 11) formiran mikroprogram (tabela 16). 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 .

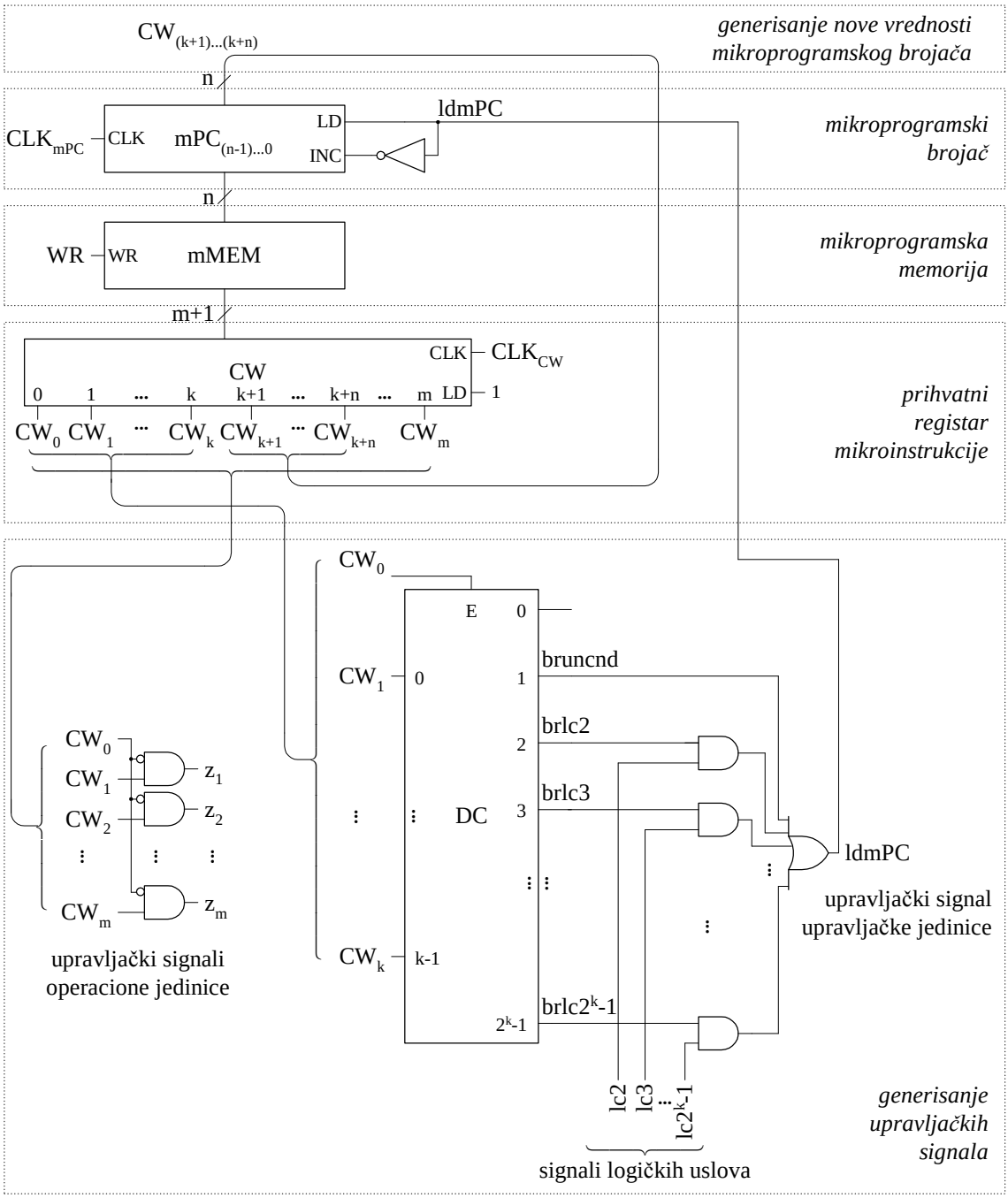
Tabela 16 Mikroprogram za upravljačku jedinicu sa dva tipa mikroinstrukcija za jedinicu sa jednom operacijom

```
0  820000   ! brnotOEU, madr0 !
1  60C400   ! ldBB, ldMQ, clAB8, clAB, ldISC !
2  834000   ! brnotMQ0, madr4 !
3  041100   ! S1, ldAB, ldAB8!
4  102A10   ! srAB8, srAB, srMQ, decISC !
5  842000   ! brnotISCZ, madr2!
6  0000C0   ! ABout, ldABWUFIJ!
7  00003C   ! MQout, ldMQWUFIJ, clOEU, stOWUFIJ !
8  810000   ! bruncnd, madr0!
```

U kolonama tabele su dati adrese mikroprogramske memorije, vrednosti polja *z*, *cc* i *ba* mikroinstrukcije i komentar. Adrese mikroprogramske memorije i vrednosti polja *z*, *cc* i *ba* mikroinstrukcije su dati u heksadecimalnom obliku. U komentaru za operacioni deo mikroinstrukcije pojavljuju se samo upravljački signali operacione jedinice koji imaju vrednost 1, dok signala koji imaju vrednost 0 nema. U komentaru za upravljački deo mikroinstrukcije pojavljuje se upravljački signal upravljačke jedinice koji ima vrednost 1 i simbolička oznaka vrednosti koja se upisuje u mikroprogramski brojač u situacijama kada se realizuje bezuslovni ili uslovni skok, dok ovih signala nema u situacijama kada nema ovih skokova.

1.2.2.2.2.2 Strukturna šema upravljačke jedinice

Upravljačku jedinicu (slika 30) u opštem slučaju čine mikroprogramski brojač mPC, mikroprogramska memorija mMEM, prihvatni registar mikroinstrukcije CW, kombinaciona mreža za generisanje upravljačkih signala i kombinaciona mreže za generisanje nove vrednosti mikroprogramskog brojača koji imaju istu funkciju i sličnu strukturu kao i u slučaju upravljačke jedinice sa jednim tipom mikroinstrukcije.



Slika 30 Upravljačka jedinica sa dva tipa mikroinstrukcija

Sadržaj mikroprogramskog brojača mPC se inkrementira ili se u mikroprogramski brojač mPC upisuje nova vrednost u zavisnosti od toga da li signal ldmPC ima vrednost 0 ili 1, respektivno. Signal ldmPC ima vrednost 0 prilikom izvršavanja operacionih mikroinstrukcija i vrednost 0 ili 1 prilikom izvršavanja upravljačkih mikroinstrukcija. Mikroprogramski brojač ima *n* razreda uz pretpostavku da je kapacitet mikroprogramske memorije 2^n reči.

Kapacitet mikroprogramske memorije je u skladu sa veličinom mikroprograma i iznosi 2^n reči, a širina reči mikroprogramske memorije određena je brojem bitova operacione ili upravljačke mikroinstrukcije u zavisnosti od toga koja je duža od ove dve mikroinstrukcije. Uz pretpostavku da je operaciona mikroinstrukcija duža od upravljačke mikroinstrukcije, uzeto je da je širina reči mikroprogramske memorije m+1 bitova, pa se stoga svi bitovi reči mikroprogramske memorije koriste u slučaju operacione mikroinstrukcije i to 1 bit za polje *h* i

m bitova za polje *z*, a samo k+n+1 bitova u slučaju upravljačke mikroinstrukcije i to1 bit za polje *h*, k bitova za polje *cc* i n bitova za polje *ba*.

Sadržaj prihvatnog registra mikroinstrukcije CW se interpretira na dva načina u zavisnosti od vrednosti razreda CW_0. Ukoliko CW_0 ima vrednost 0, razredi 1 do m registra CW se interpretiraju kao bitovi polja *z* operacione mikroinstrukcije. Ukoliko CW_0 ima vrednost 1, razredi 1 do k i k+1 do k+n registra CW se interpretiraju kao bitovi polja *cc* i *ba*, respektivno, upravljačke mikroinstrukcije

Upravljački signali operacione jedinice z_1 do z_m se generišu na osnovu sadržaja razreda CW_1 do CW_m ukoliko razred CW_0 ima vrednost 0. Upravljački signali upravljačke jedinice bruncnd, brcnd i $brlc_1$ do $brlc_i$ se generiše na osnovu sadržaja razreda CW_1 do CW_k i signala logičkih uslova lc_1 do lc_i ukoliko razred CW_0 ima vrednost 1.

U kombinacionoj mreži za generisanje nove vrednosti mikroprogramskog brojača na paralelne ulaze mikroprogramskog brojača mPC dolaze razredi CW_{k+1} do CW_{k+n} prihvatnog registra mikroinstrukcije CW.

Strukturnu šemu upravljačke jedinice za jedinicu sa jednom operacijom (slika 31) čine: mikroprogramski brojač $mPC_{3..0}$, mikroprogramska memorija mMEM, prihvatni registar mikroinstrukcije $CW_{0..31}$, kombinaciona mreža za generisanje upravljačkih signala i kombinaciona mreža za generisanje nove vrednosti mikroprogramskog brojača koji imaju istu funkciju i sličnu strukturu kao i u slučaju upravljačke jedinice sa jednim tipom mikroinstrukcije.

Mikroprogramski brojač $mPC_{3..0}$ ima 4 razreda i njime može da se vrši adresiranje mikroprogramske memorije u opsegu 0 do 15. Sadržaj mikroprogramskog brojača mPC se vrednošću 0 signala ldmPC inkrementira, dok se i mikroprogramski brojač mPC vrednošću 1 signala ldmPC upisuje nova vrednost iz CW_8 do CW_{15}.

Mikroprogramska memorije mMEM može da sadrži mikroprogram maksimalne veličine 16 mikroinstrukcija širine 24 bita, pri čemu su popunjene samo lokacije na adresama 0 do 8.

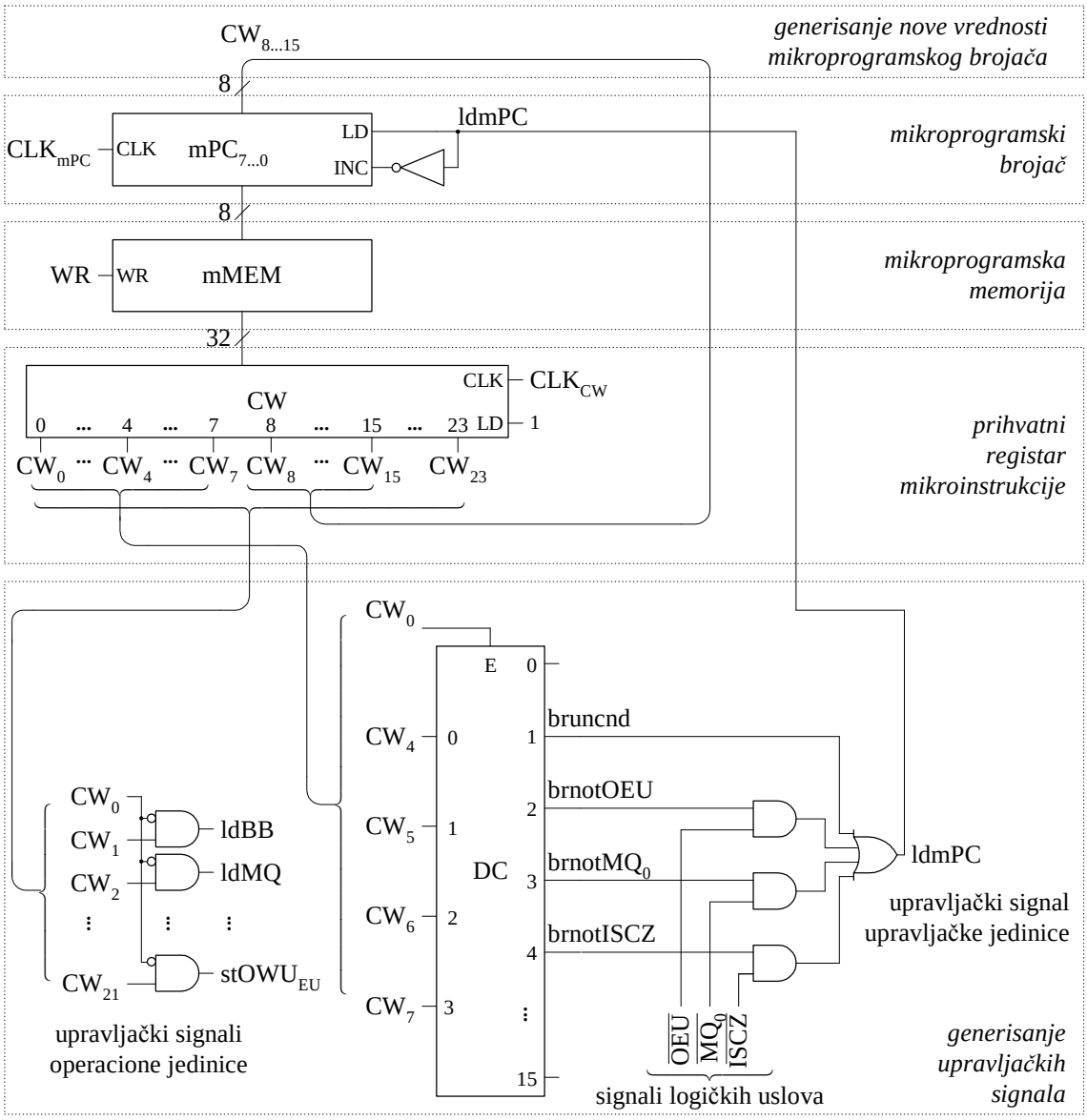
Prihvatni registar mikroinstrukcije $CW_{0..23}$ ima 24 razreda u skladu sa usvojenom dužinom mikroinstrukcije od 24 bita.

Kombinaciona mreža za generisanje upravljačkih signala generiše dve grupe signala i to upravljačke signale operacione jedinice i upravljački signal upravljačke jedinice.

Upravljački signali operacione jedinice ldBB do $stOWU_{EU}$ se generišu na osnovu sadržaja razreda CW_1 do CW_{21} ukoliko razred CW_0 ima vrednost 0.

Upravljački signali upravljačke jedinice bruncnd, brcnd, brnotOEU, brnotMQ0 i brnotISCZ se generiše na osnovu sadržaja razreda CW_4 do CW_7 i signala logičkih uslova OEU, MQ_0 i ISCZ ukoliko razred CW_0 ima vrednost 1.

U kombinacionoj mreži za generisanje nove vrednosti mikroprogramskog brojača na paralelne ulaze mikroprogramskog brojača mPC dolaze razredi CW_8 do CW_{11} prihvatnog registra mikroinstrukcije CW.



Slika 31 Upravljačka jedinica sa dva tipa mikroinstrukcija za jedinicu sa jednom operacijom

1.2.2.2.2.3 Funkcionisanje upravljačke jedinice

Funkcionisanje upravljačke jedinice se u opštem slučaju odvija po postupku datom u daljem tekstu.

Mikroprogramski brojaču mPC, čiji početni sadržaj ukazuje na početnu adresu mikroprograma operacije za koju treba generisati upravljačke signale, koristi se kao adresa mikroprogramske memorije iz koje se čita mikroinstrukcija i smešta u prihvatni registar mikroinstrukcije CW. Na osnovu toga da li sadržaj razreda CW_0, u kome se nalazi polje *h* i operacione i upravljačke mikroinstrukcije, ima vrednost 0 ili 1 utvrđuje se da li se radi o operacionoj mikroinstrukciji ili upravljačkoj mikroinstrukciji, respektivno.

Ukoliko se radi o operacionoj mikroinstrukciji, sadržaj razreda CW_1 do CW_m se interpretira kao bitovi polja *z* operacione mikroinstrukcije, pa se na osnovu sadržaja razreda CW_1 do CW_m generišu vrednosti 0 i 1 upravljačkih signala operacione jedinice z_1 do z_m. S obzirom na to da je svakom od *m* upravljačkih signala operacione jedinice dodeljen jedan od m bitova operacione mikroinstrukcije koji se nalaze u jednom od razreda CW_1 do CW_m,

odgovarajući upravljački signal operacione jedinice imaće vrednost 0 ili 1 u zavisnosti od toga da li se u odgovarajućem razredu nalazi vrednost 0 ili 1, respektivno. Pošto se radi o operacionoj mikroinstrukciji signal ldmPC ima vrednost 0, pa se mikroprogramski brojač mPC inkrementira. Time se produžava sa sekvencijalnim izvršavanjem mikroprograma.

Ukoliko se radi o upravljačkoj mikroinstrukciji, sadržaji razreda CW_1 do CW_k i CW_{k+1} do CW_{k+n} se interpretiraju kao bitovi polja *cc* i *ba* upravljačke mikroinstrukcije, respektivno, pa se na osnovu sadržaja razreda CW_1 do CW_k, a u nekim situacijama i na osnovu vrednosti signala logičkih uslova lc_1 do lc_i, generišu vrednost 0 ili 1 upravljačkih signala upravljačke jedinice bruncnd, brcnd i $brlc_1$ do $brlc_i$.

Ukoliko je binarna vrednost razreda CW_1 do CW_k jednaka 0, signali bruncnd, brcnd i $brlc_1$ do $brlc_i$, a time i signal mPC, imaju vrednost 0, pa se mikroprogramski brojač mPC inkrementira. Time se vrednošću 0 polja *cc* upravljačke mikroinstrukcije realizuje sekvencijalno izvršavanje mikroprograma.

Ukoliko je binarna vrednost razreda CW_1 do CW_k jednaka 1, signal bruncnd, a time i signal mPC, ima vrednost 1, pa se u mikroprogramski brojač mPC upisuje sadržaj razreda CW_{k+1} do CW_{k+n}. Time se vrednošću 1 polja *cc* upravljačkog dela mikroinstrukcije realizuju bezuslovni skok u mikroprogramu.

Ukoliko je binarna vrednost razreda CW_1 do CW_k u opsegu vrednosti 2 do i+1, jedan od signala uslovnih skokova $brlc_1$ do $brlc_i$ pridružen jednom od signala logičkih uslova lc_1 do lc_i, respektivno, ima vrednost 1, pa signal brcnd, a time i signal ldmPC, ima vrednost 0 ili 1 u zavisnosti od toga da li odgovarajući signal logičkog uslova lc_1 do lc_i ima vrednost 0 ili 1, respektivno. Ukoliko signal ldmPC ima vrednost 0, mikroprogramski brojač mPC se inkrementira, pa se produžava sa sekvencijalnim izvršavanjem mikroprograma. Ukoliko signal ldmPC ima vrednost 1, u mikroprogramski brojač mPC se upisuje sadržaj razreda CW_{k+1} do CW_{k+n}, pa se realizuje skok u mikroprogramu. Time se vrednostima 2 do i+1 polja *cc* upravljačke mikroinstrukcije, a na osnovu vrednosti odgovarajućih signala logičkih uslova lc_1 do lc_i, realizuje uslovni skok u mikroprogramu.

Ukoliko je binarna vrednost razreda CW_1 do CW_k u opsegu vrednosti i+2 do 2^k-1, signali bruncnd, brcnd i $brlc_1$ do $brlc_i$, a time i signal mPC, imaju vrednost 0, pa se mikroprogramski brojač mPC inkrementira. Time se vrednošću i+2 do 2^k-1 polja *cc* upravljačke mikroinstrukcije, koje nisu pridružene ni jednom signalu logičkih uslova, realizuje sekvencijalno izvršavanje mikroprograma.

Funkcionisanje upravljačke jedinice za jedinicu sa jednom operaciojom se odvija po po postupku datom u daljem tekstu.

Sa adrese 00000000 mikroprogramske memorije čita se mikroinstrukcija koja je upravljačka jer bit 0 polja *h* mikroinstrukcije ima vrednost 1. Bitovi 4 do 7 polja *cc* mikroinstrukcije imaju vrednost 0010 čime se određuje da treba proveriti vrednost signala logičkog uslova OEU i ukoliko ima vrednost 0 upisati bitove 8 do 11 polja *ba* mikroinstrukcije koji imaju vrednost 0000 u mikroprogramski brojač mPC, dok u suprotnom slučaju treba inkrementirati mikroprogramski brojač mPC.

Sa adrese 0001 mikroprogramske memorije čita se mikroinstrukcija koja je operaciona jer bit 0 polja *h* mikroinstrukcije ima vrednost 0. Bitovi 1, 2, 8, 9 i 13 polja *z* mikroinstrukcije imaju vrednost 1 dok ostali bitovi imaju vrednost 0 čime se određuje da upravljački signali operacione jedinice ldBB, ldMQ, clAB8, clAB i ldISC treba da imaju vrednost 1 dok svi ostali upravljački signali operacione jedinice vrednost 0. Pored toga, mikroprogramski brojač mPC treba inkrementirati.

Sa adrese 0010 mikroprogramske memorije čita se mikroinstrukcija koja je upravljačka jer bit 0 polja *h* mikroinstrukcije ima vrednost 1. Bitovi 4 do 7 polja *cc* mikroinstrukcije imaju vrednost 0011 čime se određuje da treba proveriti vrednost signala logičkog uslova MQ_0 i ukoliko ima vrednost 0 upisati bitove 8 do 11 polja *ba* mikroinstrukcije koji imaju vrednost 0100 u mikroprogramski brojač mPC, dok u suprotnom slučaju treba inkrementirati mikroprogramski brojač mPC.

Sa adrese 0011 mikroprogramske memorije čita se mikroinstrukcija koja je operaciona jer bit 0 polja *h* mikroinstrukcije ima vrednost 0. Bitovi 5, 11 i 15 polja *z* mikroinstrukcije imaju vrednost 1 dok ostali bitovi imaju vrednost 0 čime se određuje da upravljački signali operacione jedinice S_1, ldAB i ldAB8 treba da imaju vrednost 1 dok svi ostali upravljački signali operacione jedinice vrednost 0. Pored toga, mikroprogramski brojač mPC treba inkrementirati.

Sa adrese 0100 mikroprogramske memorije čita se mikroinstrukcija koja je operaciona jer bit 0 polja *h* mikroinstrukcije ima vrednost 0. Bitovi 3, 10, 12 i 14 polja *z* mikroinstrukcije imaju vrednost 1 dok ostali bitovi imaju vrednost 0 čime se određuje da upravljački signali operacione jedinice srMQ, srAB8, decISC i srAB i treba da imaju vrednost 1 dok svi ostali upravljački signali operacione jedinice vrednost 0. Pored toga, mikroprogramski brojač mPC treba inkrementirati.

Sa adrese 0101 mikroprogramske memorije čita se mikroinstrukcija koja je upravljačka jer bit 0 polja *h* mikroinstrukcije ima vrednost 1. Bitovi 4 do 7 polja *cc* mikroinstrukcije imaju vrednost 0100 čime se određuje da treba proveriti vrednost signala logičkog uslova ISCZ i ukoliko ima vrednost 0 upisati bitove 8 do 11 polja *ba* mikroinstrukcije koji imaju vrednost 0010 u mikroprogramski brojač mPC, dok u suprotnom slučaju treba inkrementirati mikroprogramski brojač mPC.

Sa adrese 0110 mikroprogramske memorije čita se mikroinstrukcija koja je operaciona jer bit 0 polja *h* mikroinstrukcije ima vrednost 0. Bitovi 16 i 17 polja *z* mikroinstrukcije imaju vrednost 1 dok ostali bitovi imaju vrednost 0 čime se određuje da upravljački signali operacione jedinice ABout i $ldABWU_{EU}$ treba da imaju vrednost 1 dok svi ostali upravljački signali operacione jedinice vrednost 0. Pored toga, mikroprogramski brojač mPC treba inkrementirati.

Sa adrese 0111 mikroprogramske memorije čita se mikroinstrukcija koja je operaciona jer bit 0 polja *h* mikroinstrukcije ima vrednost 0. Bitovi 18, 19, 20 i 21 polja *z* mikroinstrukcije imaju vrednost 1 dok ostali bitovi imaju vrednost 0 čime se određuje da upravljački signali operacione jedinice Mqout, $ldMQWU_{EU}$, clOEU i $stOWU_{EU}$ treba treba da imaju vrednost 1 dok svi ostali upravljački signali operacione jedinice vrednost 0. Pored toga, mikroprogramski brojač mPC treba inkrementirati.

Sa adrese 01000 mikroprogramske memorije čita se mikroinstrukcija koja je upravljačka jer bit 0 polja *h* mikroinstrukcije ima vrednost 1. Bitovi 4 do 7 polja *cc* mikroinstrukcije imaju vrednost 0001 čime se određuje da treba bezuslovno upisati bitove 8 do 15 polja *ba* mikroinstrukcije koji imaju vrednost 0000 u mikroprogramski brojač mPC. Ovim se realizuje povratak na početak mikroprograma.

Dobra strana realizacije sa dva tipa mikroinstrukcija umesto sa jednim tipom mikroinstrukcije je da se smanjuje dužina mikroinstrukcije, ali je loša strana da se povećava broj mikroinstrukcija a time i broja koraka mikroprograma što za posledicu ima povećanje vremena izvršavanja operacije. U slučaju razmatrane operacije množenja dužina mikroinstrukcije se smanjila sa 36 bitova za slučaj realizacije sa dva tipa mikroinstrukcija na

24 bita za slučaj realizacije sa jednim tipom mikroinstrukcije, ali se broj mikroinstrukcija a time i broja koraka mikroprograma povećao sa 8 na 9.

1.3 JEDINICA SA VIŠE OPERACIJA

U ovom poglavlju se daju moguće postupci realizacije operacione i upravljačke jedinice za slučaj jedinice sa više operacija. Jedinica sa više operacija koja se razmatra realizuje četiri aritmetičke operacije i to sabiranje (ADD), oduzimanje (SUB), inkrementiranje (INC) i dekrementiranje (DEC), četiri logičke operacije i to I (AND), ILI (OR), ekskluzivno ILI (XOR) i invertovanje (NOT), četiri operacije pomeranja udesno i to aritmetičko pomeranje (ASR), logičko pomeranje (LSR), rotiranje (ROR) i rotiranje sa indikatorom prenosa (RORC), četiri operacije pomeranja ulevo i to aritmetičko pomeranje (ASL), logičko pomeranje (LSL), rotiranje (ROL) i rotiranje sa indikatorom prenosa (ROLC) i aritmetičke operacije množenja (MULU) i deljenja (DIVU) i to nad 8 bitnim celobrojnim vrednostima bez znaka slučaju aritmetičkih operacija i 8 bitnim binarnim vrednostima u slučaju logičkih operacija i operacija pomeranja. Operacije ADD, SUB, INC, DEC, AND, OR, XOR, NOT, ASR, LSR, ROR, RORC, ASL, LSL, ROL, ROLC se se izvršavaju u jednoj iteraciji i za njihovu realizaciju se pored mikrooperacija prenosa, kojima se sadržaji dovode u odgovarajuće registre jedinice, koristi još samo jedna odgovarajuća aritmetička, logička ili pomerača mikrooperacija. Operacije MULU i DIVU se izvršavaju u više iteracija i za njihovu realizaciju se pored mikrooperacija prenosa, kojima se sadržaji dovode u odgovarajuće registre jedinice, koristi odgovarajuća kombinacija aritmetičkih i pomeračkih mikrooperacija. Na kraju svake operacije vrši se i provera da li je prilikom formiranja rezultata dobijena korektna vrednost i u skladu sa tim indikator C programske statusne reči PSW postavlja na vrednost 1 ili 0. Takođe se vrši provera da li je dobijena vrednost nula i u skladu sa tim indikator Z programske statusne reči PSW postavlja na vrednost 1 ili 0.

1.3.1 OPERACIONA JEDINICA

U ovom odeljku se najpre daju algoritmi operacija, zatim strukturna šema operacione jedinice i na kraju algoritam generisanja upravljačkih signala.

1.3.1.1 Algoritmi operacija

Algoritmi operacija se daju najpre za operacije koje se izvršavaju u jednoj iteraciji a zatim i za operacije koje se izvršavaju u više iteracija.

1.3.1.1.1 Operacije koje se izvršavaju u jednoj iteraciji

Aritmetičke operacije ADD, SUB, INC i DEC, logičke operacije AND, OR, XOR i NOT i operacije pomeranja ASR, LSR, ROR, RORC, ASL, LSL, ROL i ROLC se realizuju u jednoj iteraciji.

Aritmetičke operacije ADD, SUB, INC i DEC

Aritmetička operacija ADD sabira 8 bitne celobrojne vrednosti bez znaka $X_7X_6...X_1X_0$ i $Y_7Y_6...Y_1Y_0$ i formira 8 bitnu celobrojnu vrednost bez znaka rezultata $F_7F_6...F_1F_0$. Ukoliko rezultat sabiranja može da se predstavi sa 8 cifara, nema prenosa u 9-ti razred, cifre $F_7F_6...F_1F_0$ predstavljaju korektne cifre rezultata, pa se vrednost 0 upisuje u indikator C registra PSW. Ukoliko je za predstavljanje rezultata sabiranja potrebno 9 cifara, postoji prenos u 9-ti razred i cifre $F_7F_6...F_1F_0$ predstavljaju ne predstavljaju korektne cifre rezultata, pa se vrednost 1 upisuje u indikator C registra PSW. Kod operacije ADD vrednost 1 indikatora C registra PSW ukazuje da postoji prenos u 9-ti razred i da $F_7F_6...F_1F_0$ ne predstavlja korektan rezultat.

Aritmetička operacija SUB oduzima 8 bitnu celobrojnu vrednost bez znaka $Y_7Y_6...Y_1Y_0$ od 8 bitne celobrojne vrednosti bez znaka $X_7X_6...X_1X_0$ i formira 8 bitnu celobrojnu vrednost bez

znaka rezultata $F_7F_6...F_1F_0$. Ukoliko je vrednost $X_7X_6...X_1X_0$ veća od vrednosti $Y_7Y_6...Y_1Y_0$ rezultat oduzimanja može da se predstavi sa 8 cifara, nema pozajmice u 9-ti razred, cifre $F_7F_6...F_1F_0$ predstavljaju korektne cifre rezultata, pa se vrednost 0 upisuje u indikator C registra PSW. Ukoliko je vrednost $X_7X_6...X_1X_0$ manja od vrednosti $Y_7Y_6...Y_1Y_0$ i nema smisla realizovati oduzimanje jer rezultat nije celobrojna vrednost bez znaka, postoji pozajmica u 9-ti razred i cifre $F_7F_6...F_1F_0$ ne predstavljaju korektne cifre rezultata, pa se vrednost 1 upisuje u indikator C registra PSW. Kod operacije SUB vrednost 1 indikatora C registra PSW ukazuje da postoji pozajmica u 9-ti razred i da $F_7F_6...F_1F_0$ ne predstavlja korektan rezultat.

Aritmetička operacija INC sabira 8 bitnu celobrojnu vrednosti bez znaka $X_7X_6...X_1X_0$ i vrednost 1 i formira 8 bitnu celobrojnu vrednost bez znaka rezultata $F_7F_6...F_1F_0$. Ukoliko rezultat sabiranja može da se predstavi sa 8 cifara, nema prenosa u 9-ti razred, cifre $F_7F_6...F_1F_0$ predstavljaju korektne cifre rezultata, pa se vrednost 0 upisuje u indikator C registra PSW. Ukoliko je za predstavljanje rezultata sabiranja potrebno 9 cifara, postoji prenos u 9-ti razred i cifre $F_7F_6...F_1F_0$ ne predstavljaju korektne cifre rezultata, pa se vrednost 1 upisuje u indikator C registra PSW. Kod operacije INC vrednost 1 indikatora C registra PSW ukazuje da postoji prenos u 9-ti razred i da $F_7F_6...F_1F_0$ ne predstavlja korektan rezultat.

Aritmetička operacija DEC oduzima vrednost 1od 8 bitne celobrojne vrednosti bez znaka $X_7X_6...X_1X_0$ i formira 8 bitnu celobrojnu vrednost bez znaka rezultata $F_7F_6...F_1F_0$. Ukoliko je vrednost $X_7X_6...X_1X_0$ veća od vrednosti 1 rezultat oduzimanja može da se predstavi sa 8 cifara, nema pozajmice u 9-ti razred, cifre $F_7F_6...F_1F_0$ predstavljaju korektne cifre rezultata, pa se vrednost 0 upisuje u indikator C registra PSW. Ukoliko je vrednost $X_7X_6...X_1X_0$ manja od vrednosti 1 i nema smisla realizovati oduzimanje jer rezultat nije celobrojna vrednost bez znaka, postoji pozajmica u 9-ti razred i cifre $F_7F_6...F_1F_0$ ne predstavljaju korektne cifre rezultata, pa se vrednost 1 upisuje u indikator C registra PSW. Kod operacije DEC vrednost 1 indikatora C registra PSW ukazuje da postoji pozajmica u 9-ti razred i da $F_7F_6...F_1F_0$ ne predstavlja korektan rezultat.

Kod sve četiri aritmetičke operacije ADD, SUB, INC i DEC vrši se i provera da li je vrednost binarne reči rezultata $F_7F_6...F_1F_0$ nula ili različita od nule i na osnovu toga se u indikator Z registra PSW upisuje vrednost 1 ili 0, respektivno.

Logičke operacije AND, OR, XOR i NOT

Logička operacija AND realizuje logičku I operaciju nad X_i i Y_i i formira F_i, i=7,...0, pri čemu su $X_7X_6...X_1X_0$ i $Y_7Y_6...Y_1Y_0$ 8 bitne binarne rečima nad kojima se operacija realizuje i $F_7F_6...F_1F_0$ formirana 8 bitnu binarna reč rezultata.

Logička operacija OR realizuje logičku ILI operaciju nad X_i i Y_i i formira F_i, i=7,...0, pri čemu su $X_7X_6...X_1X_0$ i $Y_7Y_6...Y_1Y_0$ 8 bitne binarne rečima nad kojima se operacija realizuje i $F_7F_6...F_1F_0$ formirana 8 bitnu binarna reč rezultata.

Logička operacija XOR realizuje logičku ekskluzivno ILI operaciju nad X_i i Y_i i formira F_i, i=7,...0, pri čemu su $X_7X_6...X_1X_0$ i $Y_7Y_6...Y_1Y_0$ 8 bitne binarne rečima nad kojima se operacija realizuje i $F_7F_6...F_1F_0$ formirana 8 bitnu binarna reč rezultata.

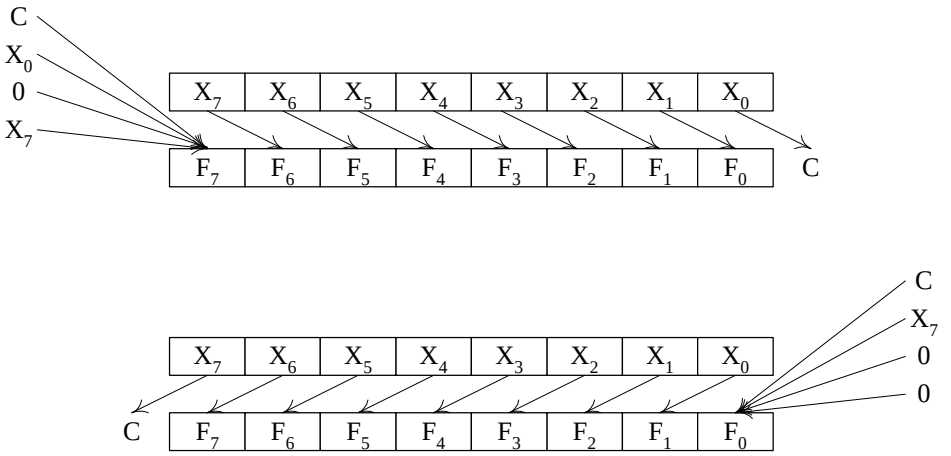
Logička operacija NOT realizuje logičku NE operaciju nad X_i i formira F_i, i=7,...0, pri čemu je $X_7X_6...X_1X_0$ 8 bitna binarna reč nad kojom se operacija realizuje i $F_7F_6...F_1F_0$ formirana 8 bitnu binarna reč rezultata.

Kod sve četiri logičke operacije AND, OR, XOR i NOT u indikator C registra PSW upisuje se vrednost 0. Poret toga za sve četiri logičke operacije AND, OR, XOR i NOT vrši se i

provera da li je vrednost binarne reči rezultata $F_7F_6...F_1F_0$ nula ili različita od nule i na osnovu toga se u indikator Z registra PSW upisuje vrednost 1 ili 0, respektivno.

Operacije pomeranja ASR, LSR, ROR, RORC, ASL, LSL, ROL i ROLC

Operacije pomeranja realizuju pomeranje 8 bitne binarne reči $X_{7..0}$ za jedno mesto udesno ili jedno mesto ulevo i formiraju 8 bitnu binarnu reč $F_{7..0}$ za (slika 32).



Slika 32 Pomeranje udesno i ulevo

Kod pomeranja udesno binarne reči $X_7X_6...X_1X_0$ razredi binarne reči $X_7X_6...$ X_2X_1 postaju razredi $F_6F_5...F_1F_0$ binarne reči rezultata $F_7F_6...F_1F_0$. Najstariji bit F_7 binarne reči rezultata može da dobije jednu od vrednosti X_7, 0, X_0 ili C i u zavisnosti koju od te četiri vrednosti dobije razlikuju se četiri operacije pomeranja udesno i to aritmetičko pomeranje udesno, logičko pomeranje udesno, rotiranje udesno i rotiranje udesno kroz indikator C, respektivno. U sva četiri slučaja najmlađi bit X_0 binarne reči $X_7X_6...$ X_1X_0 se upisuje u indikator C registra PSW.

Kod pomeranja ulevo binarne reči $X_7X_6...X_1X_0$ razredi binarne reči $X_6X_5...X_1X_0$ postaju razredi $F_7F_6...F_2F_1$ binarne reči rezultata $F_7F_6...F_1F_0$. Najmlađi bit F_0 binarne reči rezultata može da dobije jednu od vrednosti 0, 0, X_7 ili C i u zavisnosti koju od te četiri vrednosti dobije razlikuju se četiri operacije pomeranja ulevo i to aritmetičko pomeranje ulevo, logičko pomeranje ulevo, rotiranje ulevo i rotiranje ulevo kroz indikator C, respektivno. U sva četiri slučaja najstariji bit X_7 binarne reči $X_7X_6...X_1X_0$ se upisuje u indikator C registra PSW.

Kod sve četiri operacije pomeranja udesno i sve četiri operacije pomeranja ulevo vrši se i provera da li je vrednost binarne reči rezultata $F_7F_6...F_1F_0$ nula ili različita od nule i na osnovu toga se u indikator Z registra PSW upisuje vrednost 1 ili 0, respektivno.

1.3.1.1.2 Operacije koje se izvršavaju u više iteracija

Aritmetičke operacije MULU i DIVU se realizuju u više iteracija. Algoritam operacije MULU je dat u odeljku 0, pa se u ovom odeljku razmatra samo algoritam operacije DIVU.

Aritmetička operacija DIVU realizuje deljenje celobrojnih vrednosti bez znaka pri čemu su deljenik $X=X_{15}X_{14}…X_0$ i delilac $Y=Y_7Y_6…Y_0$, dok je rezultat operacije količnik $Q=Q_7Q_6…Q_0$ i ostatak $Z=Z_7Z_6…Z_0$.

Ako se u algoritmu deljenja sa $Z(8)$ do $Z(0)$ označe trenutni ostaci, a sa Q_7 do Q_0 cifre količnika $Q_{7..0}$, onda se algoritam deljenja može opisati na sledeći način:

1. $Z(8) = X - 2^8Y$

if $Z(8) \geq 0$ then prekoračenje

else $Z(7) = (Z(8) + 2^8Y) - 2^7Y$

2. if $Z(i) \geq 0$ then $Q_i = 1$, $Z(i-1) = Z(i) - 2^{i-1}Y$, $i = 7, 6, \ldots, 1$

else $Q_i = 0$, $Z(i-1) = (Z(i) + 2^iY) - 2^{i-1}Y$, $i = 7, 6, \ldots, 1$

3. if $Z(0) \geq 0$ then $Q_0 = 1$, $Z = Z(0)$

else $Q_0 = 0$, $Z = Z(0) + 2^0Y$

Na početku je potrebno da se izvrši provera da li deljenik $X = X_{15}X_{14}\ldots X_0$ i delilac $Y = Y_7Y_6\ldots Y_0$ imaju takve vrednosti da količnik $Q=Q_7Q_6\ldots Q_0$ može da se predstavi sa osam cifara. Stoga se, najpre, izračunava trenutni ostatak $Z(8)$, tako što se od deljenika X oduzme 2^8Y, a zatim vrši provera da li je $Z(8)\geq 0$ ili $Z(8)<0$. Ako je $Z(8)\geq 0$, količnik bi imao više od osam cifara. U tom slučaju dolazi do prekoračenja i operacija deljenja se završava. Ako je $Z(8)<0$, količnik će imati osam cifara. U tom slučaju operacija deljenja započinje sračunavanjem trenutnog ostatka Z(7) tako što se izvrši, najpre, obnavljanje trenutnog ostatka Z(8) dodavanjem 2^8Y, a zatim od dobijene vrednosti oduzimanje 2^7Y. Na osnovu toga se za trenutni ostatak Z(7) dobija $Z(7)=(Z(8)+2^8Y)-2^7Y$.

Slede koraci realizovani u sedam iteracija, $i = 7, 6, \ldots, 1$, u kojima se cifre količnika Q_7 do Q_1 dobijaju na osnovu vrednosti trenutnih ostataka Z(7) do Z(1). U njima se izračunavaju i trenutni ostaci Z(6) do Z(0). Ako je u i-toj iteraciji $Z(i)\geq 0$, tada je cifra $Q_i = 1$. U istoj iteraciji se, u cilju dobijanja trenutnog ostatka $Z(i-1)$, mora od trenutnog ostatka $Z(i)$ izvršiti oduzimanje $2^{i-1}Y$. Na osnovu toga se dobija $Z(i-1)= Z(i)-2^{i-1}Y$. Ako je i-toj iteraciji $Z(i) < 0$, tada je cifra $Q_i = 0$. U istom koraku se, u cilju dobijanja trenutnog ostatka $Z(i-1)$, mora izvršiti obnavljanje trenutnog ostatka Z(i) dodavanjem 2^iY i od dobijene vrednosti oduzimanje $2^{i-1}Y$. Na osnovu toga se dobija $Z(i-1) = (Z(i)+2^iY)-2^{i-1}Y$.

Na kraju se dobija cifra količnika Q_0 na osnovu vrednosti trenutnog ostatka Z(0) i izačunava ostatak Z. Ako je $Z(0)\geq 0$, tada je cifra $Q_0 = 1$. Pored toga ostatak Z dobija vrednost trenutnog ostatka $Z(0)$. Ako je $Z(0)<0$, tada je cifra $Q_0 = 0$. Pored toga ostatak Z dobija vrednost obnovljenog trenutnog ostatka Z(0) koji se dobija dodavanjem 2^0Y.

Za realizaciju operacije deljenja po ovom algoritmu potrebna je ALU dužine 16 razreda, koja realizuje sabiranje i oduzimanje na dužini od 16 razreda.

Međutim, algoritam deljenja se može modifikovati i opisati na sledeći način:

1. $Z(8) = X - 2^8Y$

if $Z(8) \geq 0$ *then* prekoračenje

else $Z(7)^* = 2^1(Z(8) + 2^8Y)-2^8Y = 2^1Z(8)+2^8Y$

pri čemu je $Z(7)^* = 2^1Z(7)$

2. if $Z(i)^* \geq 0$ *then* $Q_i=1$, $Z(i-1)^* =2^1Z(i)^*-2^8Y$, $i = 7, 6, \ldots, 1$

else $Q_i =0$, $Z(i-1)^*= 2^1(Z(i)^*+2^8Y)-2^8Y =2^1Z(i)^*+2^8Y$, $i = 7, 6, \ldots, 1$

pri čemu je $Z(i)^* = 2^{8-i}Z(i)$

3. if $Z(0)^* \geq 0$ *then* $Q_0=1$, $Z^*=Z(0)^*$

else $Q_0=0$, $Z^*=Z(0)^*+2^8Y$

pri čemu je $Z(0)^*=2^8Z(0)$

Na početku je potrebno da se izvrši provera da li deljenik X = $X_{15}X_{14}…X_0$ i delilac $Y = Y_7Y_6…Y_0$ imaju takve vrednosti da količnik $Q=Q_7Q_6…Q_0$ može da se predstavi sa osam cifara. Stoga se, najpre, izračunava trenutni ostatak $Z(8)$, tako što se od deljenika X oduzme 2^8Y, a zatim vrši provera da li je $Z(8)\geq 0$ ili $Z(8)<0$. Ako je $Z(8) \geq 0$, količnik bi imao više od osam cifara. U tom slučaju dolazi do prekoračenja i operacija deljenja se završava. Ako je $Z(8)<0$, količnik će imati osam cifara. U tom slučaju operacija deljenja započinje sračunavanjem trenutnog ostatka $Z(7)^*$ tako što se izvrši, najpre, obnavljanje trenutnog ostatka $Z(8)$ dodavanjem 2^8Y, zatim njegovo pomeranje za jedno mesto ulevo i na kraju od dobijene vrednosti oduzimanje 2^8Y. Na osnovu toga se za trenutni ostatak $Z(7)^*$ dobija $Z(7)^* = 2^1(Z(8)+2^8Y)-2^8Y = 2^1Z(8)+2^8Y$.

Slede koraci realizovani u sedam iteracija, $i = 7, 6, \dots, 1$, u kojima se cifre količnika Q_7 do Q_1 dobijaju na osnovu vrednosti trenutnih ostataka Z(7)* do Z(1)*. U njima se izračunavaju i trenutni ostaci Z(6)* do Z(0)*. Ako je u i-toj iteraciji $Z(i)^*\geq 0$, tada je cifra $Q_i=1$. U istoj iteraciji se, u cilju dobijanja trenutnog ostatka Z(i-1)*, mora izvršiti pomeranje trenutnog ostatka Z(i)* za jedno mesto ulevo pa onda od dobijene vrednosti oduzimanje 2^8Y. Na osnovu toga je $Z(i-1)^*=2^1Z(i)^*-2^8Y$. Ako je i-toj iteraciji $Z(i)^*<0$, tada je cifra $Q_i = 0$. U istom koraku se, u cilju dobijanja trenutnog ostatka Z(i-1)*, mora izvršiti obnavljanje trenutnog ostatka Z(i)* dodavanjem 2^8Y, njegovo pomeranje za jedno mesto ulevo i od dobijene vrednosti oduzimanje 2^8Y. Na osnovu toga je $Z(i-1)^*=2^1(Z(i)^*+2^8Y)-2^8Y=2^1Z(i)^*+ 2^8Y$

Na kraju se dobija cifra količnika Q_0 na osnovu vrednosti trenutnog ostatka Z(0)* i izačunava ostatak Z*. Ako je $Z(0)^*\geq 0$, tada je cifra $Q_0 = 1$. Pored toga ostatak Z* dobija vrednost trenutnog ostatka $Z(0)^*$. Ako je $Z(0)^*<0$, tada je cifra $Q_0 = 0$. Pored toga ostatak Z* dobija vrednost obnovljenog trenutnog ostatka Z(0)* koji se dobija dodavanjem 2^8Y.

Kako je $Z(0)^*=2^8Z(0)$, to je i $Z^*=2^8Z$, pa ostatak Z je dat sa osam starijih cifara Z*.

Iz prvih nekoliko koraka realizacije operacije DIVU (slika 33) može se uočiti da se pri formiranju $Z(8)= X-Y2^8$, $Z(7)^*= Z(8)2^1\pm Y2^8$, $Z(6)^*= Z(7)^*2^1\pm Y2^8$, itd. kao nove cifre u nekoj iteraciji formiraju samo cifre 8 do 16, dok se cifre 7 do 0 nasleđuju od X, $Z(8)2^1$, $Z(7)^*2^1$ itd. iz prethodne iteracije. To je posledica toga što se kod formiranja $Z(8)= X-Y2^8$, $Z(7)^*=Z(8)2^1\pm Y2^8$, $Z(6)^*=Z(7)^*2^1\pm Y2^8$, itd. neke iteracije na X, $Z(8)2^1$, $Z(7)^*2^1$ itd. prethodne iteracije dodaje vrednost Y pomerena 8 mesta ulevo koja kao cifre 7 do 0 ima nule. Zbog toga je za njegovu realizaciju je dovoljan ALU sa osam razreda, pri čemu komplemente S_{16} iz $Z(7)^*=Z(8)2^1\pm Y2^8$, $Z(6)^*=Z(7)^*2^1\pm Y2^8$, $Z(5)^*= Z(6)^*2^1\pm Y2^8$ itd. treba čuvati u nekom pomeračkom registru jer predstavljaju cifre 7, 6, 5 itd. količnika.

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	0	X_{15}	X_{14}	X_{13}	X_{12}	X_{11}	X_{10}	X_9	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0
$Y2^8$	0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	0	0	0	0	0	0	0	0
$Z(8)= X-Y2^8$	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0
$Z(8)2^1$	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	0
$Y2^8$	0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	0	0	0	0	0	0	0	0
$Z(7)^*= Z(8)2^1\pm Y2^8$	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	X_6	X_5	X_4	X_3	X_2	X_1	X_0	0
$Z(7)^*2^1$	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	X_6	X_5	X_4	X_3	X_2	X_1	X_0	0	0
$Y2^8$	0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	0	0	0	0	0	0	0	0
$Z(6)^*= Z(7)^*2^1\pm Y2^8$	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	X_5	X_4	X_3	X_2	X_1	X_0	0	0
$Z(6)^*2^1$	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	X_5	X_4	X_3	X_2	X_1	X_0	0	0	0
$Y2^8$	0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	0	0	0	0	0	0	0	0
$Z(5)^*= Z(6)^*2^1\pm Y2^8$	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	X_4	X_3	X_2	X_1	X_0	0	0	0

...

Slika 33 Operacija DIVU

1.3.1.2 Strukturna šema

Strukturne šeme operacionih jedinica se razlikuju po tome da li se za povezivanje prekidačkih mreža unutar operacione jedinice koriste direktne veze ili interne magistrale.

1.3.1.2.1 Direktne veze

Moguća strukturna šema operacione jedinice sa direktnim vezama za jedinicu sa više operacija koja se razmatra je data na slikama 34, 35 i 37.

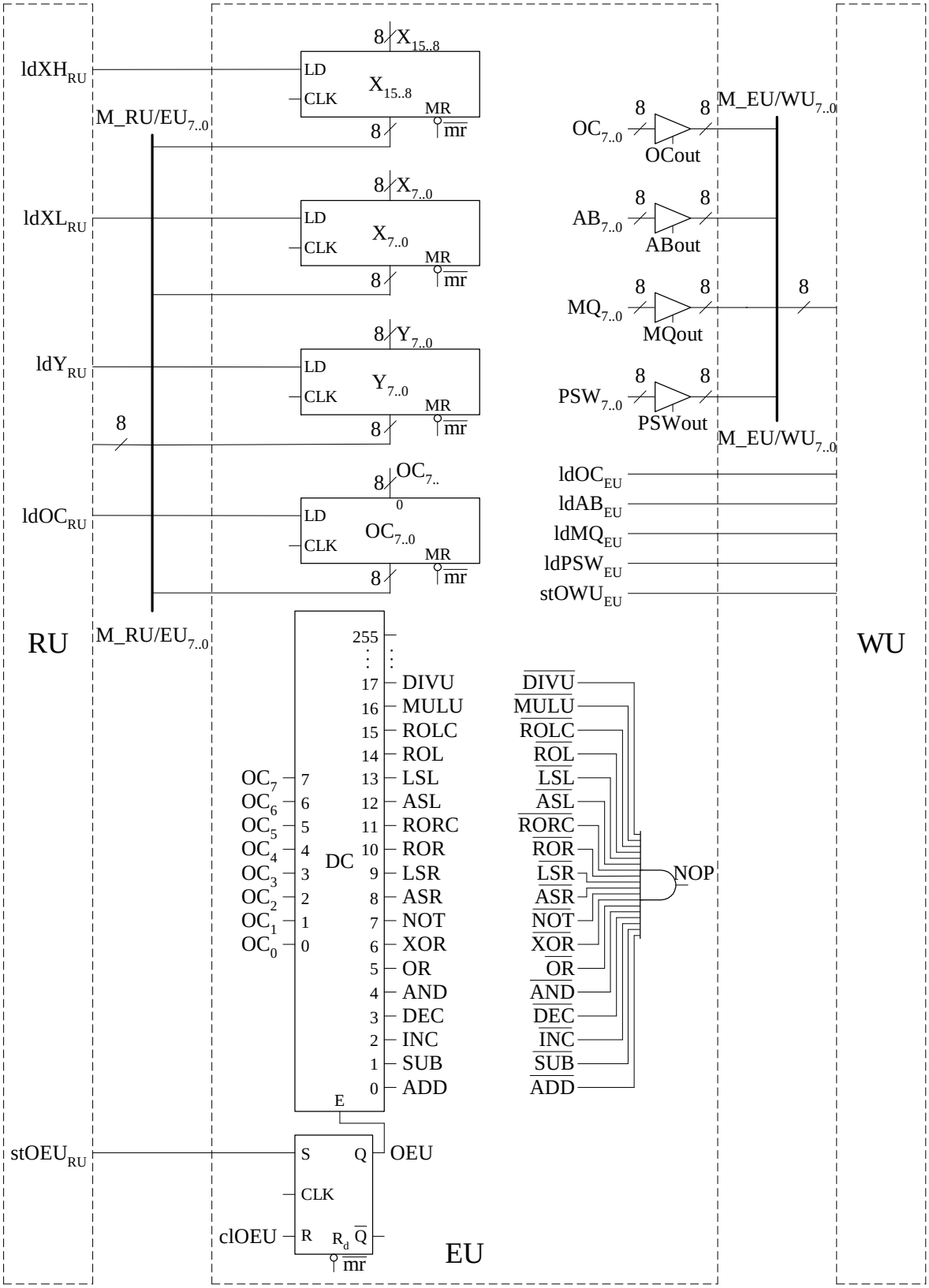
Operaciona jedinica je realizovana u skladu sa usvojenom strukturom digitalnih uređaja kojom se podrazumeva da je jedinica koja se razmatra deo nekog uređaja koji se sastoji iz više jedinica (poglavlje 1.1.4). Stoga jedinica koja se razmatra najpre dobija binarne vrednosti nad kojima treba da realizuje određenu operaciju od jedinice koja je njen prethodnik, zatim u skladu sa usvojenin algoritmom izvršava određenu operaciju i na kraju rezultat izvršene operacije predaje jedinici koja je njen sledbenik. Jedinica koja se razmatra se dalje naziva jedinica EU (Execution Unit), jedinica koja je njen prethodnik jedinica RU (Read Unit) i jedinica koja je njem sledbenik jedinica WU (Write Unit). U određenom trenutku samo jedna od ove tri jedinice može da bude aktivna. Stoga u jedinicama RU, EU i WU postoje flip-flopovi ORU (Operational Read Unit), OEU (Operational Execution Unit) i OWU (Operational Write Unit) koji vrednostima 1 i 0 određuju da li je određena jedinica aktivna ili neaktivna, respektivno.

Jedinica RU i EU su povezane linijama magistrale M_RU/$EU_{7..0}$ dužine 8 bitova po kojima jedinica RU šalje jedinicu EU 8 bitne binarne vrednosti koje predstavljaju podatke nad kojima operacija treba da se razlikuje i binarnu vrednost koja određuje operaciju koja treba da se realizuje. Registri $X_{15..8}$, $X_{7..0}$ i $Y_{7..0}$ dužine 8 razreda služe za smeštanje početnih vrednosti nad kojima odgovarajuće operacija treba da se realizuje (slika 34). U slučaju aritmetičkih operacija ADD, SUB i MULU i logičkih operacija AND, OR i XOR u registre $X_{7..0}$ i $Y_{7..0}$ se upisuju vrednosti X i Y dužine 8 bitova, respektivno, nad kojima se date operacije realizuju. U slučaju aritmetičkih operacija INC i DEC, logičke operacije NOT i operacija pomeranja ASR, LSR, ROR, RORC, ASL, LSL, ROL i ROLC u registar $X_{7..0}$ se upisuje vrednost X dužine 8 bitova nad kojom se data operacija realizuje. U slučaju aritmetičke operacije DIVU u registre $X_{15..8}$ i $X_{7..0}$ dužine 8 bitova se upisuje 8 starijih i 8 mlađih bitova, respektivno, deljenika X dužine 16 bitova i u registar $Y_{7..0}$ dužine 8 bitova se upisuje 8 delioca Y dužine 8 bitova. Pošto jedinica može da realizuje 18 operacija, postoji i registar $OC_{7..0}$ dužine 8 razreda u koji se upisuje binarna vrednosti operacije koju treba realizovati (slika 34).

Vrednosti $X_{15..8}$, $X_{7..0}$, $Y_{7..0}$ i $OC_{7..0}$ u posebnim ciklusima šalje jedinica RU po linijama magistrale M_RU/$EU_{7..0}$ i u registre $X_{15..8}$, $X_{7..0}$, $Y_{7..0}$ i $OC_{7..0}$ upisuju vrednostima 1 signala $ldXH_{RU}$, $ldXL_{RU}$, ldY_{RU} i $ldOCR_{RU}$, respektivno. Signal $ldXH_{RU}$ jedinica RU generiše onda kada na linije M_RU/$EU_{7..0}$ pušta sadržaj koji treba da se upiše u registar $X_{15..8}$, signal $ldXL_{RU}$ jedinica RU generiše onda kada na linije M_RU/$EU_{7..0}$ pušta sadržaj koji treba da se upiše u registar $X_{7..0}$, signal ldY_{RU} onda kada na linije M_RU/$EU_{7..0}$ pušta sadržaja koji treba da se upiše u registar $Y_{7..0}$ i signal $ldOCR_{RU}$ onda kada na linije M_RU/$EU_{7..0}$ pušta sadržaja koji treba da se upiše u registar $OCR_{7..0}$. Izvršavanje odgovarajuće operacije se startuje kada jedinica RU vrednošću 1 signala $stOEU_{RU}$ u flip-flop OEU upiše vrednost 1 (slika 34).

Dok signal OEU ima vrednost 0 svi signali operacija od ADD do DIVU koji se na osnovu sadržaja registra $OCR_{7..0}$ formiraju na izlazima dekodera DC imaju vrednost 0. Međutim, po upisivanju vrednost 1 u flip-flop OEU, a u zavisnosti od toga koja se binarna vrednost nalazi u registru $OCR_{7..0}$, jedan od signala operacija ADD do DIVU postaje 1, čime se startuje izvršavanje odgovarajuće operacije. Signal NOP ima vrednost 1 ukoliko svi signali od ADD

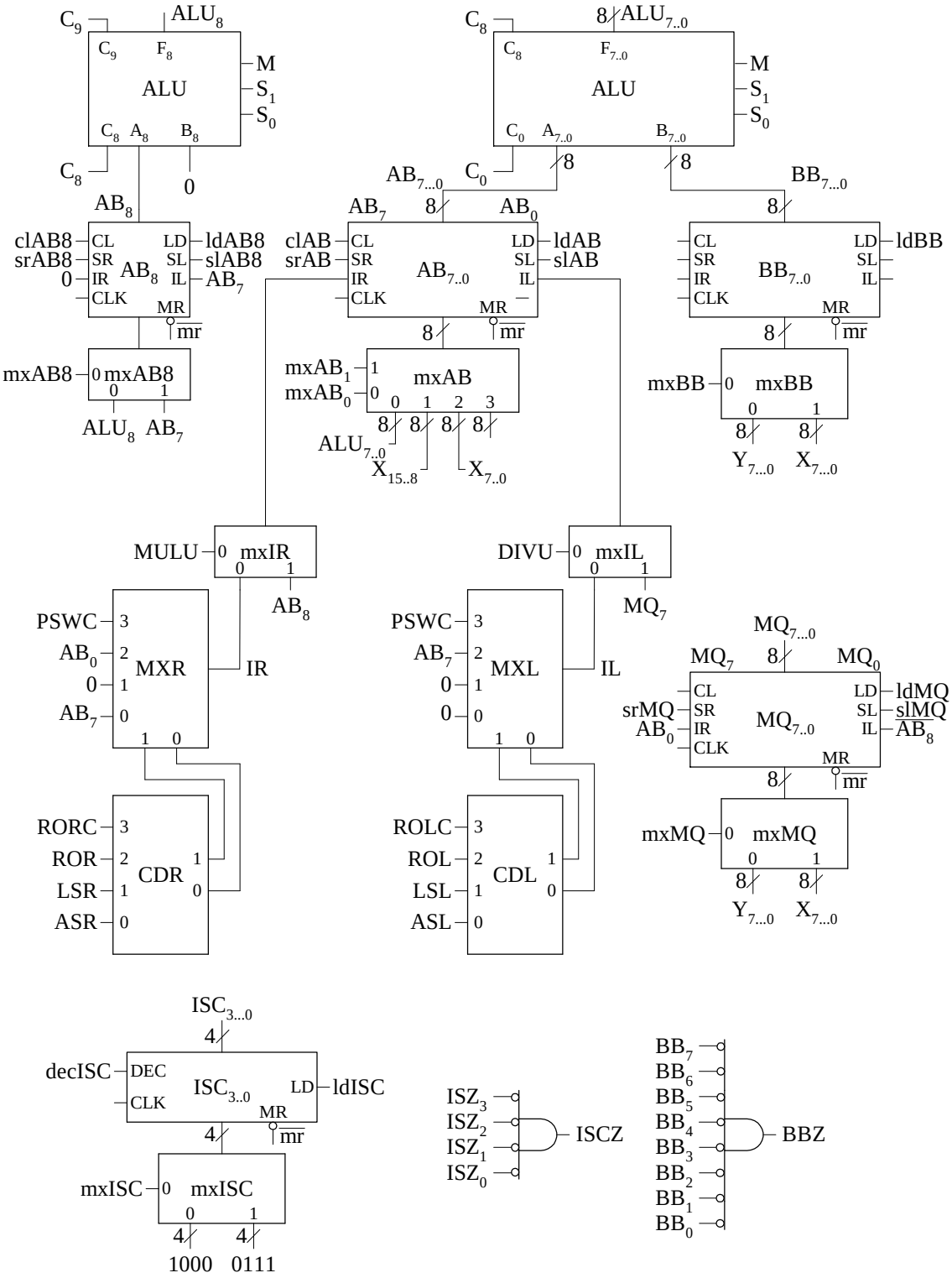
do DIVU imaju vrednost 0 i vrednost 1 ukoliko neko od signala ADD do DIVU ima vrednost 1. Signali ADD do DIVU i NOP su signali logičkih uslova koje koristi upravljačka jedinica.



Slika 34 Operaciona jedinica sa direktnim vezama za jedinicu sa više operacija (prvi deo)

Za realizaciju aritmetičkih i logičkih mikrooperacija koristi se aritmetičko logička jedinica ALU (slika 35). Jedinica ALU mikrooperacije realizuje nad sadržajima sa ulaza $A_{8..0}$ i $B_{8..0}$, a

mikrooperacije se biraju upravljačkim signalima M, S_1 i S_0 (slika 36). Na ulaze $A_{7..0}$ i $B_{7..0}$ dolaze sadržaji registara $AB_{7..0}$ i $BB_{7..0}$, respektivno, na ulaze A_8 i B_8 dolazi sadržaj jednorazrednog registra AB_8 i vrednost 0, respektivno, i na ulaz C_8 signal prenosa C_8.



Slika 35 Operaciona jedinica sa direktnim vezama za jedinicu sa više operacija (drugi deo)

Aritmetičke mikrooperacije imaju dva značenja u zavisnosti od vrednosti upravljačkog signala C_0. Tako na primer u slučaju prve vrste na izlazima ALU će biti sadržaj sa ulaza A ukoliko je C_0 nula i sadržaj sa ulaz A uvećan za jedan ukoliko je C_0 jedan. U slučaju druge

vrste na izlazima ALU će biti razlika sadržaja sa ulaza A i B ukoliko je C_0 jedan i razlika sadržaja sa ulaza A i B umanjena za jedan ukoliko je C_0 nula. U slučaju treće vrste na izlazima ALU će biti suma sadržaja sa ulaza A i B ukoliko je C_0 nula i suma sadržaja sa ulaza A i B uvećana za jedan ukoliko je C_0 jedan. U slućaju četvrte vrste na izlazima ALU će biti sadržaj sa ulaza A ukoliko je C_0 jedan i sadržaj sa ulaz A umanjen za jedan ukoliko je C_0 nula. Ovo je u saglasnosti sa ranijim razmatranjima realizacije inkrementiranja, oduzimanja, sabiranja i dekrementiranja korišćenjem sabirača.

M	S_1	S_0	operacija
0	0	0	$A + C_0$
0	0	1	$A - B - \overline{C_0}$
0	1	0	$A + B + C_0$
0	1	1	$A - \overline{C_0}$
1	0	0	$A \cdot B$
1	0	1	$A \oplus B$
1	1	0	$A + B$
1	1	1	$\overline{A}$

Slika 36 Mikrooperacije u jedinici ALU

Za realizaciju pomeračkih mikrooperacija koristi se pomerački registar $AB_{7..0}$ čiji se sadržaj vrednošću 1 signala srAB pomera jedno mesto udesno i vrednošću 1 signala slAB pomera jedno mesto ulevo.

Aritmetičke operacije ADD i SUB i logičke operacije AND, OR i XOR se realizuju nad sadržajima registara $AB_{7..0}$ i $BB_{7..0}$ i rezultat $ALU_{7..0}$ upisuje u registar $AB_{7..0}$. Zbog toga se sadržaji registara $X_{7..0}$ i $Y_{7..0}$ preko multipleksera mxAB i mxBB vode na ulaze registara $AB_{7..0}$ i $BB_{7..0}$, respektivno, a rezultat $ALU_{7..0}$ se sa izlaza $F_{7..0}$ preko multipleksera mxAB vodi u registar $AB_{7..0}$.

Aritmetičke operacije operacija INC i DEC i logička operacija NOT se realizuju nad sadržajem registra $AB_{7..0}$ i rezultat $ALU_{7..0}$ upisuje u registar $AB_{7..0}$. Zbog toga se sadržaj registra $X_{7..0}$ preko multipleksera mxAB vodi na ulaze registra $AB_{7..0}$, a rezultat $ALU_{7..0}$ se sa izlaza $F_{7..0}$ preko multipleksera mxAB vodi u registar $AB_{7..0}$.

Operacije pomeranja ASR, LSR, ROR i RORC se realizuju nad sadržajem registra $AB_{7..0}$ tako što se sadržaj registra $AB_{7..0}$ pomera jedno mesto udesno, dok se operacije pomeranja ASL, LSL, ROL i ROLC se realizuju nad sadržajem registra $AB_{7..0}$ tako što se sadržaj registra $AB_{7..0}$ pomera jedno mesto ulevo. Zbog toga se sadržaj registra $X_{7..0}$ preko multipleksera mxAB vodi na ulaze registra $AB_{7..0}$, a rezultat posle pomeranja ostaje u registru $AB_{7..0}$. Kod operacija pomeranja udesno vrednost IR koja treba da se upiše u AB_7 se propušta kroz multiplekser mxIR. Kod operacija pomeranja ulevo vrednost IL koja treba da se upiše u AB_0 se propušta kroz multiplekser mxIL. Vrednost IR je AB_7, 0, AB_0 ili PSWC u zavisnosti od toga koji od signala operacija ASR, LSR, ROR i RORC ima vrednost 1. Vrednost IL je 0, 0, AB_7 ili PSWC u zavisnosti od toga koji od signala operacija ASL, LSL, ROL i ROLC ima vrednost 1.

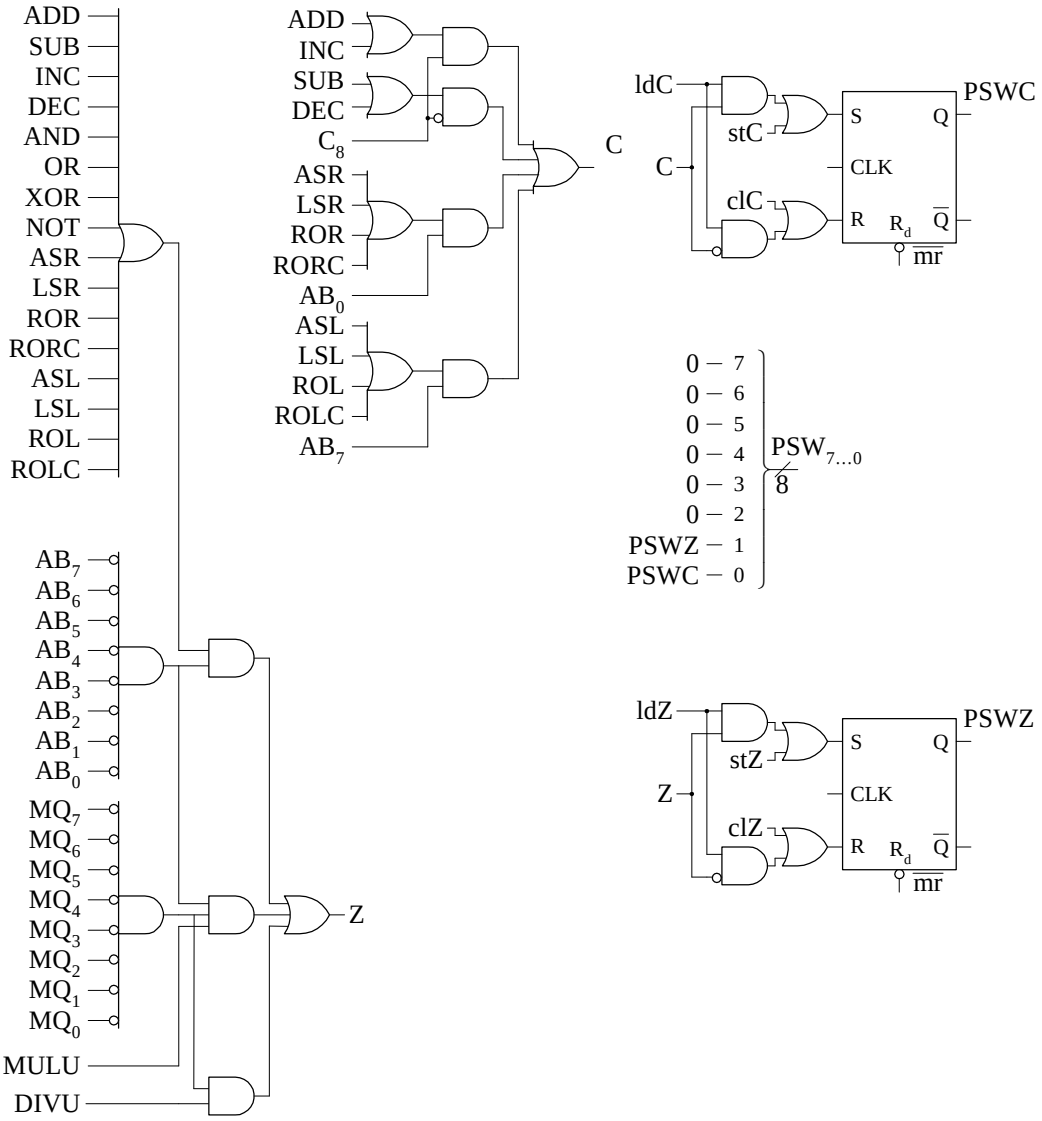
Aritmetička operacija MULU se realizuje u 8 iteracija tako što se u svakoj iteraciji izvršavanjem mikrooperacija sabiranja i pomeranja udesno izračunava devet starijih cifara trenutnog proizvoda date iteracije i jedna za drugom osam najmlađih cifara konačnog proizvoda. Devet starijih cifara trenutnog proizvoda se formira u registrima AB_8 i $AB_{7..0}$, a

osam najmlađih cifara konačnog proizvoda u registru $MQ_{7..0}$. Vrednostima 1 signala clAB8 i clAB registri AB_8 i $AB_{7..0}$ se dovode na početne vrednosti 0. Vrednost $X_{7..0}$ koju treba u svakoj iteraciji dodavati na trenutni proizvod u registru $AB_{7..0}$ i time dobiti novu vrednost trenutnog proizvoda u registrima AB_8 i $AB_{7..0}$, treba da bude u registru $BB_{7..0}$. Zbog toga se sadržaj registra $X_{7..0}$ preko multipleksera mxBB vodi na ulaze registra $BB_{7..0}$, a rezultat ALU_8 i $ALU_{7..0}$ se sa izlaza F_8 i $F_{7..0}$ preko multipleksera mxAB8 i mxAB vode u registare AB_8 i $AB_{7..0}$. U svako iteraciji se proveravaju redom cifre Y_0 do Y_7 i u zavisnosti od toga da li je vrednost cifre 1 ili 0, vrednost $X_{7..0}$ se prilikom sračunavanja trenutnog proizvoda date iteracije ili dodaje ili ne dodaje na vrednost trenutnog proizvoda prethodne iteracije. Cifre Y_0 do Y_7 se redom u svakoj iteraciji pojavlju u razredu MQ_0. Zbog toga se sadržaj registra $Y_{7..0}$ preko multipleksera mxMQ vodi na ulaze registra $MQ_{7..0}$. Registri AB_8, $AB_{7..0}$ i $MQ_{7..0}$ se u svakoj iteraciji pomeraju jedno mesto udesno. U registar AB_8 se upisuje 0, u registar $AB_{7..0}$ osam najstarijih cifara trenutnog proizvoda koje učestvuju u formiranju trenutnog proizvoda sledeće iteracije, u razred MQ_7 se upisuje cifra trenutnog proizvoda koja ne učestvuje u formiranju cifara trenutnog proizvoda u sledećim iteracijama, a mlađe cifre trenutnog proizvoda i preostale cifre $Y_{7..0}$ koje se nalaze u $MQ_{7..0}$ pomeraju se jedno mesto udesno. Pomeranjem registra $MQ_{7..0}$ za jedno mesto udesno u registar $MQ_{7..0}$ se u osam iteracija ubacuju jedna za drugom osam najmlađih cifara proizvoda, a u razredu MQ_0 pojavljuju redom cifre Y_0 do Y_7. Posle osam iteracija u registru $AB_{7..0}$ se nalazi osam starijih cifara konačnog proizvoda, a u registru $MQ_{7..0}$ osam mlađih cifara konačnog proizvoda.

Aritmetička operacija DIVU se realizuje u 8 iteracija tako što se izvršavanjem mikrooperacija pomeranja ulevo i sabiranja ili oduzimanja u svakoj iteraciji izračunava devet starijih cifara trenutnog ostatka date iteracije i jedna za drugom cifre Q_7 do Q_0 količnika. Devet starijih cifara trenutnog ostatka se formira u registrima AB_8 i $AB_{7..0}$. Vrednošću 1 signala clAB8 registar AB_8 se dovodi na početnu vrednost 0. Početna vrednost ostatka je vrednost deljenika X dužine 16 bitova čijih se 8 starijih i 8 mlađih bitova iz registara $X_{15..8}$ i $X_{7..0}$ dužine 8 bitova preko multipleksera mxAB i mxMQ vodi na ulaze registara $AB_{7..0}$ i $MQ_{7..0}$. Vrednost delioca $Y_{7..0}$ dužine 8 bitova koju treba u svakoj iteraciji dodavati na ili oduzimati od trenutne vrednost ostatka u registru $AB_{7..0}$ i time dobiti novu vrednost trenutnog ostatka u registrima AB_8 i $AB_{7..0}$, treba da bude u registru $BB_{7..0}$. Zbog toga se sadržaj registra $Y_{7..0}$ preko multipleksera mxBB vodi na ulaze registra $BB_{7..0}$, a rezultat ALU_8 i $ALU_{7..0}$ se sa izlaza F_8 i $F_{7..0}$ preko multipleksera mxAB8 i mxAB vodi u registre AB_8 i $AB_{7..0}$. Registri AB_8, $AB_{7..0}$ i $MQ_{7..0}$ se u svakoj iteraciji pomeraju jedno mesto ulevo. U registar AB_8 se upisuje AB_7, u registar $AB_{7..0}$ osam najstarijih cifara trenutnog ostatka koje učestvuju u formiranju trenutnog ostatka sledeće iteracije, u razred MQ_0 se upisuje komplement AB_8 koji predstavlja cifru količnika koja se formira u datoj iteraciji, a mlađe cifre trenutnog ostatka i cifre količnika formirane u prethodnim iteracijam koje se nalaze u $MQ_{7..0}$ pomeraju se jedno mesto ulevo. Pomeranjem registra $MQ_{7..0}$ za jedno mesto ulevo u registar $MQ_{7..0}$ se u osam iteracija ubacuju jedna za drugom osam cifara količnika Q_7 do Q_0. U svako iteraciji se proveravaju redom formirane cifre Q_7 do Q_0 i u zavisnosti od toga da li je vrednost cifre 0 ili 1, vrednost $Y_{7..0}$ se prilikom sračunavanja trenutnog ostatka date iteracije ili dodaje ili ne dodaje na trenutnu vrednost ostatka prethodne iteracije. Posle osam iteracija u registru $AB_{7..0}$ se nalazi osam cifara konačnog ostatka, a u registru $MQ_{7..0}$ osam cifara količnika.

Tokom izvršavanja operacija formiraju se indikatori C i Z i vrednostima 1 signala ldC i ldZ upisuju u razrede PSWC i PSWZ programske statusne reči $PSW_{7..0}$, respektivno (slika 37). Formiranje vrednosti indikatora C i Z je posebno definisano za svaku od 18 operacija. U slučaju aritmetičkih operacija ADD i INC vrednost indikatora C zavisi od vrednosti signala prenosa koji je određen vrednošću signala C_8, dok u slučaju aritmetičkih operacija SUB i

DEC vrednost indikatora C zavisi od vrednosti signala pozajmice koji je određen vrednošću komplementa signala C_8. U slučaju operacija pomeranja udesno ASR, LSR, ROR i RORC vrednost indikatora C zavisi od vrednosti razreda AB_0 pre pomeranja sadržaja registra $AB_{7..0}$ udesno, dok u slučaju operacija pomeranja ulevo ASL, LSL, ROL i ROLC vrednost indikatora C zavisi od vrednosti razreda AB_7 pre pomeranja sadržaja registra $AB_{7..0}$ ulevo. U slučaju svih ostalih operacija indikator C ima vrednost 0. U slučaju operacija ADD, SUB, INC, DEC, AND, OR, XOR, NOT, ASR, LSR, ROR, RORC, ASL, LSL, ROL i ROLC vrednost indikatora Z je 1 ili 0 u zavisnosti od toga da li je sadržaja svih razreda registra $AB_{7..0}$ nula ili nije nula, respektivno. U slučaju operacije MULU vrednost indikatora Z je 1 ili 0 u zavisnosti od toga da li je sadržaja svih razreda registara $AB_{7..0}$ i $MQ_{7..0}$ nula ili nije nula, respektivno. U slučaju operacije DIVU vrednost indikatora Z je 1 ili 0 u zavisnosti od toga da li je sadržaj svih razreda registra $MQ_{7..0}$ nula ili nije nula, respektivno.



Slika 37 Operaciona jedinica sa direktnim vezama za jedinicu sa više operacija (treći deo)

Algoritmi operacija MULU i DIVU se zasnivaju na ponavljanju koraka sabiranja i pomeranja udesno i pomeranja ulevo i sabiranja ili oduzimanja u 8 i 7 iteracija, respektivno. Za brojanje ponavljanja koraka koristi se brojač ciklusa $ISC_{3..0}$ (slika 34). Vrednost 8 za operaciju MULU i vrednost 7 za operaciju DIVU se propuštaju kroz multiplekser mxISC i u brojač $ISC_{3..0}$ paralelno upisuje signalom ldISC. Sadržaj brojača $ISC_{3..0}$ se signalom decISC

dekrementira u svakoj iteraciji. Signal ISCZ na izlazu I elementa ima vrednost 0 ukoliko sadržaj brojača $ISC_{3..0}$ nije nula i vrednost 1 ukoliko je sadržaj brojača $ISC_{3..0}$ nula. Signal ISCZ je signal logičkog uslova koji koristi upravljačka jedinica.

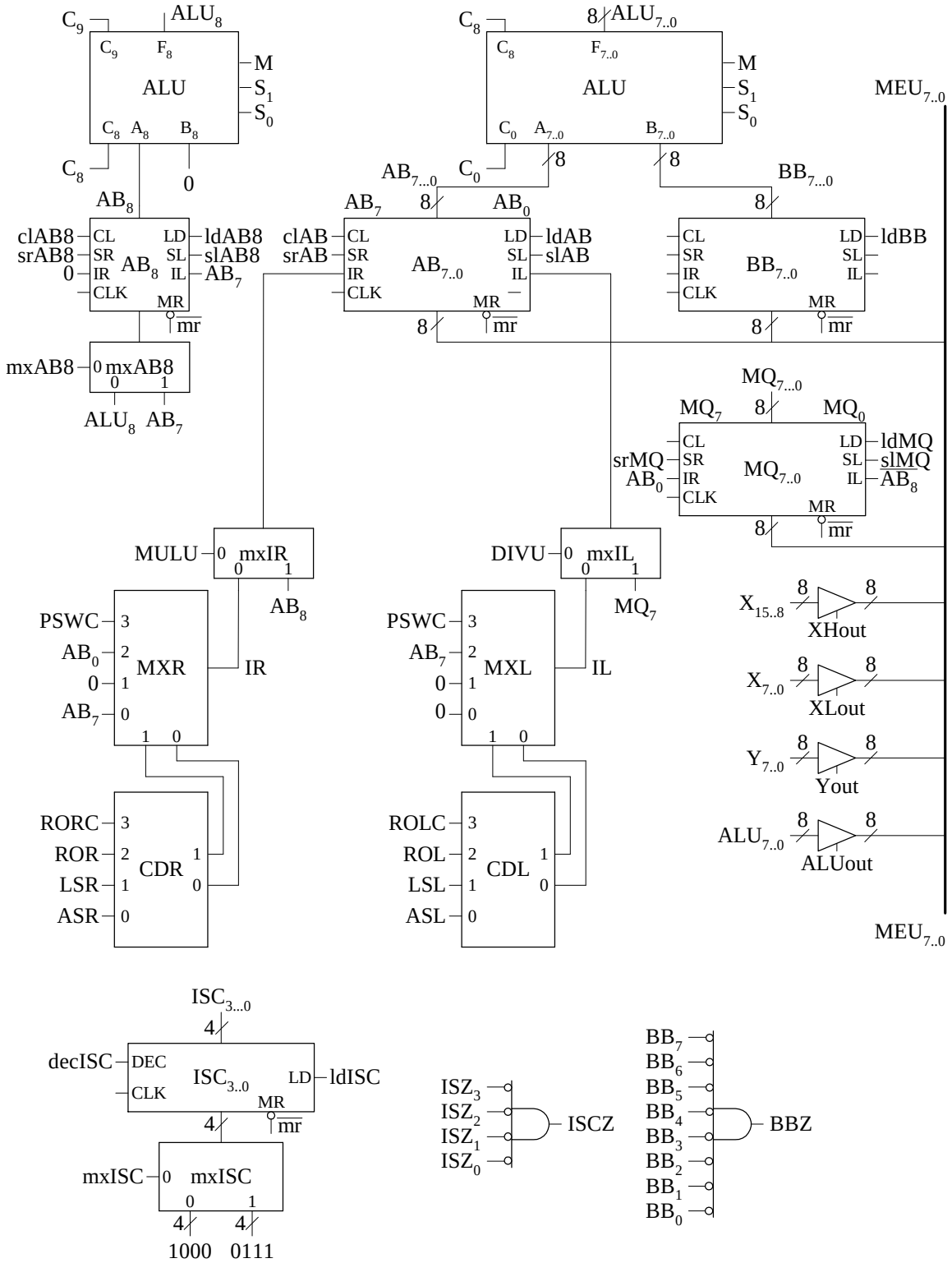
Pri kraju svog rada jedinica EU prenosi u posebnim ciklusima sadržaje registara u kojima je rezultat operacije u odgovarajuće registre jedinice WU preko linija magistrale $M\_EU/WU_{7..0}$ dužine 8 bitova kojima su jedinice EU i WU povezane (slika 34). U slučaju operacija ADD, SUB, INC, DEC, AND, OR, XOR, NOT ASR, LSR, ROR, RORC, ASL, LSL, ROL i ROLC rezultat je u registru $AB_{7..0}$. Stoga jedinica EU generiše vrednost 1 signala ABout kojom otvara bafere sa tri stanja i sadržaj registra $AB_{7..0}$ pušta na linije magistrale $M\_EU/WU_{7..0}$ i vrednost 1 signala $ldABWU_{EU}$ kojim sadržaj sa magistrale $M\_EU/WU_{7..0}$ upisuje u registar $ABWU_{7..0}$ jedinice WU. U slučaju operacije MULU proizvod je u registrima $AB_{7..0}$ i $MQ_{7..0}$, a u slučaju operacije DIVU količnik je u registru $MQ_{7..0}$ i ostatak u registru $AB_{7..0}$. Stoga u slučaju operacija MULU i DIVU jedinica EU na identičan način kao i u slučaju prenošenja sadržaja registra $AB_{7..0}$ vrednostima 1 signala MQout i $ldMQWU_{EU}$ sadržaj registra $MQ_{7..0}$ pušta na linije magistrale $M\_EU/WU_{7..0}$ i upisuje u registar $MQWU_{7..0}$ jedinice WU. Pored toga jedinica EU u slučaju svih operacija vrednostima 1 signala PSWout i $ldPSWWU_{EU}$ sadržaj registra $PSW_{7..0}$ pušta na linije magistrale $M\_EU/WU_{7..0}$ i upisuje u registar $PSWWU_{7..0}$ jedinice WU. Na kraju jedinica EU vrednostima 1 signala clOEU i $stOWU_{EJ}$ u flip-flopove OEU jedince EU i OWU jedinice WU upisuju vrednosti 0 i 1, respektivno, i time zaustavlja jedinicu OEU i startuje jedinicu OWU.

1.3.1.2.2 Interne magistrale

Za povezivanje prekidačkih mreža unutar operacione jedinice mogu da se koriste i interne magistrale i to jedna, dve ili tri interne magistrale.

Moguća strukturna šema operacione jedinice sa jednom internom magistralom za jedinicu sa više operacija je data na slici 38.

Izlazi registara $X_{15..8}$, $X_{7..0}$ i $Y_{7..0}$ i aritmetičko logičke jedinice $ALU_{7..0}$ povezani su preko bafera sa tri stanje na linije interne magistrala $MEU_{7..0}$, a sadržaj sa linija interne magistrale $MEU_{7..0}$ se vodi na paralelne ulaze registara $AB_{7..0}$, $BB_{7..0}$ i $MQ_{7..0}$. Kao posledica ovakvog načina povezivanja prekidačkih mreža, upisivanje sadržaja sa izlaza jednog od registara registra $X_{15..8}$, $X_{7..0}$,$Y_{7..0}$ ili aritmetičko logičke jedinice $ALU_{7..0}$ u neki od registara $AB_{7..0}$, $BB_{7..0}$ i $MQ_{7..0}$ se realizuje generisanjem vrednosti 1 jednog od signala signala XHout, XLout, Yout ili ALUout kojim se sadržaj sa izlaza jednog od registara registra $X_{15..8}$, $X_{7..0}$,$Y_{7..0}$ ili aritmetičko logičke jedinice $ALU_{7..0}$, respektivno, pušta na linije interne magistrale $MEU_{7..0}$ i jednog od signala signala ldAB, ldBB ili ldMQ kojim se sadržaj sa linija interne magistrale $MEU_{7..0}$ upisuje u jedan od registara $AB_{7..0}$, $BB_{7..0}$ i $MQ_{7..0}$, respektivno. Treba uočiti da je strukturna šema operacione jedinice sa više operacija nešto složenija od operacione jedinice za jednu operaciju (slika 8) i da povezivanje prekidačkih mreža jednom internom magistralom (slika 38) ima prednost u odnosu na povezivanje direktnim vezama (slika 35), jer se eliminišu multiplekseri mxAB, mxBB i mxMQ. Međutim, u nekim situacijama nedostatak može da bude to što je za vreme trajanja jedne periode signala takta moguće po linijama jedne interne magistrale prebaciti u neki od registara $AB_{7..0}$, $BB_{7..0}$ i $MQ_{7..0}$ samo sadržaj sa jednog od izlaza registara $X_{15..8}$, $X_{7..0}$,$Y_{7..0}$ ili aritmetičko logičke jedinice $ALU_{7..0}$.



Slika 38 Operaciona jedinica sa jednom internom magistralom za jedinicu sa više operacija

U slučaju operacija INC, DEC, NOT, ASR, LSR, ROR, RORC, ASL, LSL, ROL i ROLC postoji potreba da se na početku samo vrednost $X_{7..0}$ prebaci u $AB_{7..0}$, pa je za to potrebna jedna perioda signala takta bez obzira na to da li se za povezivanje koriste direktne veze ili jedna interna magistrala. Međutim, u slučaju operacija ADD, SUB, AND, OR i XOR postoji potreba da se na početku vrednosti $X_{7..0}$ i $Y_{7..0}$ prebace u $AB_{7..0}$ i $BB_{7..0}$, respektivno, a u slučaju operacije MULU da se vrednosti $X_{7..0}$ i $Y_{7..0}$ prebace u $BB_{7..0}$ i $MQ_{7..0}$, respektivno, pa su, za razliku od povezivanja direktnim vezama gde je to moguće ostvariti za vreme jedne

periode signala takta, u slučaju povezivanja jednom internom magistralom potrebne dve periode signala takta. Situacija je još nepovoljnija u slučaju operacije DIVU gde postoji potreba da se na početku vrednosti $X_{15..8}$, $X_{7..0}$ i $Y_{7..0}$ prebace u $AB_{7..0}$, $MQ_{7..0}$ i $BB_{7..0}$, respektivno, pa su, za razliku od povezivanja direktnim vezama gde je to moguće ostvariti za vreme jedne periode signala takta, u slučaju povezivanja jednom internom magistralom potrebne tri periode signala takta. U slučaju operacija ADD, SUB, INC, DEC, AND, OR, XOR, NOT, MULU i DIVU postoji potreba da se tokom njihovog izvršavanja vrednost $ALU_{7..0}$ prebaci u $AB_{7..0}$, pa je za to potrebna jedna perioda signala takta bez obzira na to da li se za povezivanje koriste direktne veze ili jedna interna magistrala.

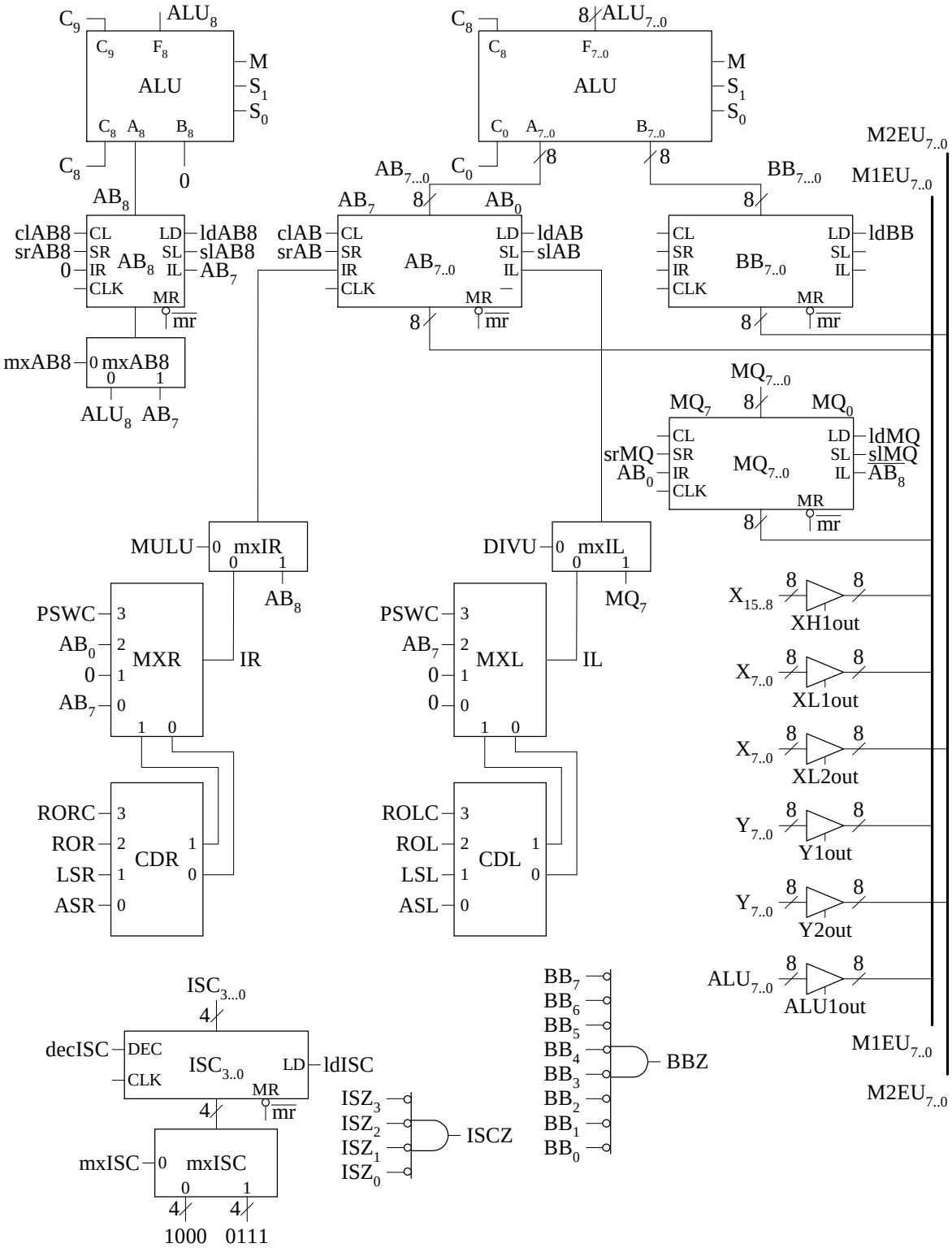
Nedostatak strukturnih šema operacionih jedinica sa jednom internom magistralom je da je za vreme jedne periode signala takta moguće samo jedan prenos sadržaja između prekidačkih mreža se može ublažiti realizacijom strukturne šeme operacione jedinice sa dve ili tri interne magistrale.

Moguća strukturna šema operacione jedinice sa dve interne magistrale za jedinicu sa više operacija je data na slici 39. Izlazi registara $X_{15..8}$, $X_{7..0}$ i $Y_{7..0}$ i aritmetičko logičke jedinice $ALU_{7..0}$ povezani su preko bafera sa tri stanje na linije interne magistrala $M1EU_{7..0}$ i na njih puštaju vrednošću 1 jednog od signala XH1out, XL1out, Y1out i ALU1out, respektivno. Sadržaj sa linija interne magistrale $M1EU_{7..0}$ se vodi na paralelne ulaze registara $AB_{7..0}$ i $MQ_{7..0}$ i u njih upisuje vrednošću 1 signala ldAB i ldMQ, respektivno. Izlazi registara $X_{7..0}$ i $Y_{7..0}$ povezani su preko bafera sa tri stanje i na linije interne magistrala $M2EU_{7..0}$ i na njih puštaju vrednošću 1 jednog od signala XL2out i Y2out, respektivno. Sadržaj sa linija interne magistrale $M2EU_{7..0}$ se vodi na paralelne ulaze registra $BB_{7..0}$ i u njega upisuje vrednošću 1 signala ldBB.

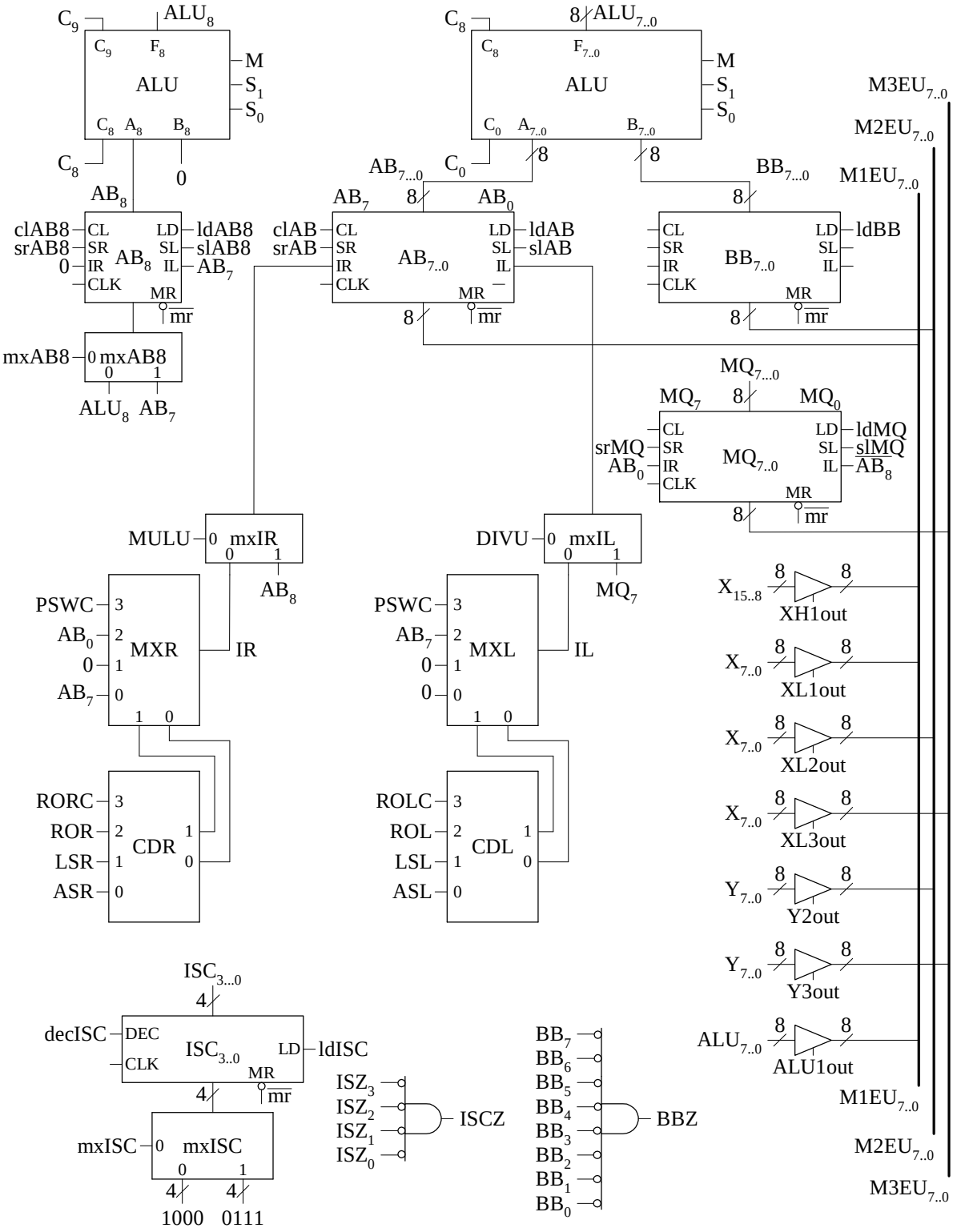
Sada je za operacije ADD, SUB, AND, OR i XOR, za koje postoji potreba da se na početku vrednosti $X_{7..0}$ i $Y_{7..0}$ prebace u $AB_{7..0}$ i $BB_{7..0}$, respektivno, moguće to uraditi za vreme trajanja jedne periode signala takta tako što se vrednosti $X_{7..0}$ i $Y_{7..0}$ prebacuju u $AB_{7..0}$ i $BB_{7..0}$ preko dve interne magistrale $M1EU_{7..0}$ i $M2EU_{7..0}$, respektivno. Takođe i za operaciju MULU, za koju postoji potreba da se na početku vrednosti $X_{7..0}$ i $Y_{7..0}$ prebace u $BB_{7..0}$ i $MQ_{7..0}$, respektivno, moguće je to uraditi za vreme trajanja jedne periode signala takta tako što se vrednosti $X_{7..0}$ i $Y_{7..0}$ prebacuju u $BB_{7..0}$ i $MQ_{7..0}$ preko dve interne magistrale $M2EU_{7..0}$ i $M1EU_{7..0}$, respektivno. Međutim, za operaciju DIVU, gde postoji potreba da se na početku vrednosti $X_{15..8}$, $X_{7..0}$ i $Y_{7..0}$ prebace u $AB_{7..0}$, $MQ_{7..0}$ i $BB_{7..0}$, respektivno, u slučaju povezivanja sa dve interne magistralom potrebne su dve periode signala takta. Za vreme prve periode signala takta moguće je da se $X_{15..8}$ i $Y_{7..0}$ prebace u $AB_{7..0}$ i $BB_{7..0}$ preko dve interne magistrale $M1EU_{7..0}$ i $M2EU_{7..0}$, respektivno, a za vreme druge periode signala takta da se $X_{7..0}$ prebaci u $MQ_{7..0}$ preko interne magistrale $M1EU_{7..0}$. Ovo je bolje rešenje u odnosu na povezivanje sa jednom internom magistralom, ali još uvek lošije u odnosu na povezivanje direktnim vezama.

Moguća strukturna šema operacione jedinice sa tri interne magistrale za jedinicu sa više operacija je data na slici 40. Izlazi registara $X_{15..8}$ i $X_{7..0}$ i aritmetičko logičke jedinice $ALU_{7..0}$ povezani su preko bafera sa tri stanje na linije interne magistrala $M1EU_{7..0}$ i na njih puštaju vrednošću 1 jednog od signala XH1out, XL1out i ALU1out, respektivno. Sadržaj sa linija interne magistrale $M1EU_{7..0}$ se vodi na paralelne ulaze registra $AB_{7..0}$ i u njega upisuje vrednošću 1 signala ldAB. Izlazi registara $X_{7..0}$ i $Y_{7..0}$ povezani su preko bafera sa tri stanje i na linije interne magistrala $M2EU_{7..0}$ i na njih puštaju vrednošću 1 jednog od signala XL2out i Y2out, respektivno. Sadržaj sa linija interne magistrale $M2EU_{7..0}$ se vodi na paralelne ulaze registra $BB_{7..0}$ i u njega upisuje vrednošću 1 signala ldBB. Izlazi registara $X_{7..0}$ i $Y_{7..0}$ povezani

su preko bafera sa tri stanje i na linije interne magistrala $M3EU_{7..0}$ i na njih puštaju vrednošću 1 jednog od signala XL3out i Y3out, respektivno. Sadržaj sa linija interne magistrale $M3EU_{7..0}$ se vodi na paralelne ulaze registra $MQ_{7..0}$ i u njega upisuje vrednošću 1 signala ldMQ. Sada je i za operaciju DIVU, za koju postoji potreba da se na početku vrednosti $X_{15..8}$, $X_{7..0}$ i $Y_{7..0}$ prebace u $AB_{7..0}$, $MQ_{7..0}$ i $BB_{7..0}$, respektivno, moguće to uraditi za vreme trajanja jedne periode signala takta tako što se vrednosti $X_{15..8}$, $X_{7..0}$ i $Y_{7..0}$ prebacuju u $AB_{7..0}$, $MQ_{7..0}$ i $BB_{7..0}$, preko tri interne magistrale $M1EU_{7..0}$, $M3EU_{7..0}$ i $M2EU_{7..0}$, respektivno.



Slika 39 Operaciona jedinica sa dve interne magistrale za jedinicu sa više operacija



Slika 40 Operaciona jedinica sa tri interne magistrale za jedinicu sa više operacija

1.3.1.3 Algoritam generisanja upravljačkih signala

Algoritmi generisanja upravljačkih signala se razlikuju u zavisnosti od toga da li se za povezivanje prekidačkih mreža unutar operacione jedinice koriste direktne veze ili interne magistrale. U ovom odeljku se razmatra algoritam generisanja upravljačkih signala za jedinicu sa više operacija u kojoj se za povezivanje prekidačkim mreža unutar operacione jedinice koriste direktne veze. Algoritam generisanja upravljačkih signala je formiran na osnovu usvojenih algoritama operacija (odeljak 1.3.1.1) i dat je u obliku dijagrama toka mikrooperacija, dijagrama toka upravljačkih signala operacione jedinice (slike 41 do 47) i sekvence upravljačkih po koracima (tabela 17).

Dijagram toka mikrooperacija i dijagram toka upravljačkih signala operacione jedinice su dati istovremeno i predstavljeni su operacionim i upravljačkim blokovima. U operacionim blokovima dijagrama toka mikrooperacija se nalaze mikrooperacije koje treba da se izvršavaju da bi se odgovarajuća operacija realizovala. Svakom operacionom bloku u dijagramu toka mikrooperacija odgovara operacioni blok u dijagramu toka upravljačkih signala operacione jedinice. U svakom od operacionih blokova dijagrama toka upravljačkih signala operacione jedinice se nalaze upravljački signali operacione jedinice koji treba da imaju vrednost 1 da bi se mikrooperacije iz odgovarajućeg operacionog bloka dijagrama toka mikrooperacija realizovale. U upravljačkim blokovima dijagrama toka mikrooperacija i dijagrama toka upravljačkih signala operacione jedinice se nalaze signali logičkih uslova od čijih vrednosti zavisi redosled generisanja vrednosti 1 upravljačkih signala operacione jedinice, a time i redosled izvršavanja mikrooperacija.

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

signali.

Ovaj iskaz sadrži spisak upravljačkih signala operacione jedinice i određuje koji signali treba da imaju vrednost 1. Iskazi za skokove su oblika

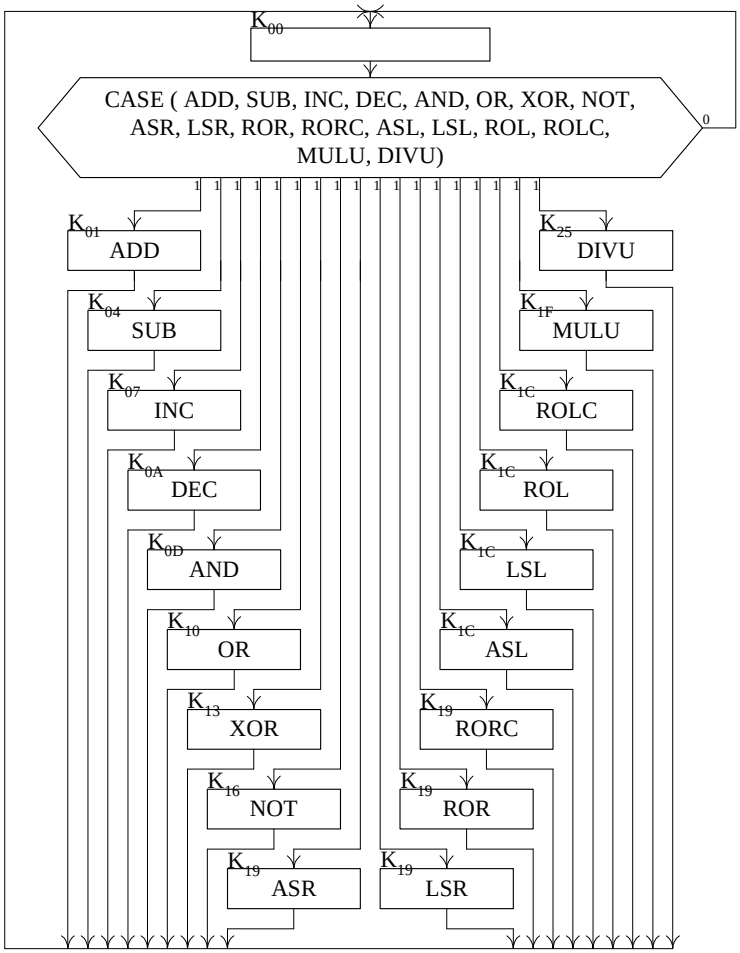
br $step_A$,

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

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

Prvi iskaz sadrži korak $step_A$ na koji treba bezuslovno preći i u daljem tekstu se referiše kao bezuslovni skok. Drugi iskaz sadrži signal **uslov** i korak $step_A$ i određuje korak $step_A$ na koji treba preći ukoliko signal **uslov** ima vrednost 1 i u daljem tekstu se referiše kao uslovni skok. Treći iskaz sadrži signale **$uslov_1$**, ..., **$uslov_n$** i korake $step_{A1}$, ..., $step_{An}$ i određuje na koji od koraka $step_{A1}$, ..., $step_{An}$ treba preći u zavisnosti od toga koji od signala **$uslov_1$**, ..., **$uslov_n$** ima vrednost 1 i u daljem tekstu se referiše kao višestruki uslovni skok.

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

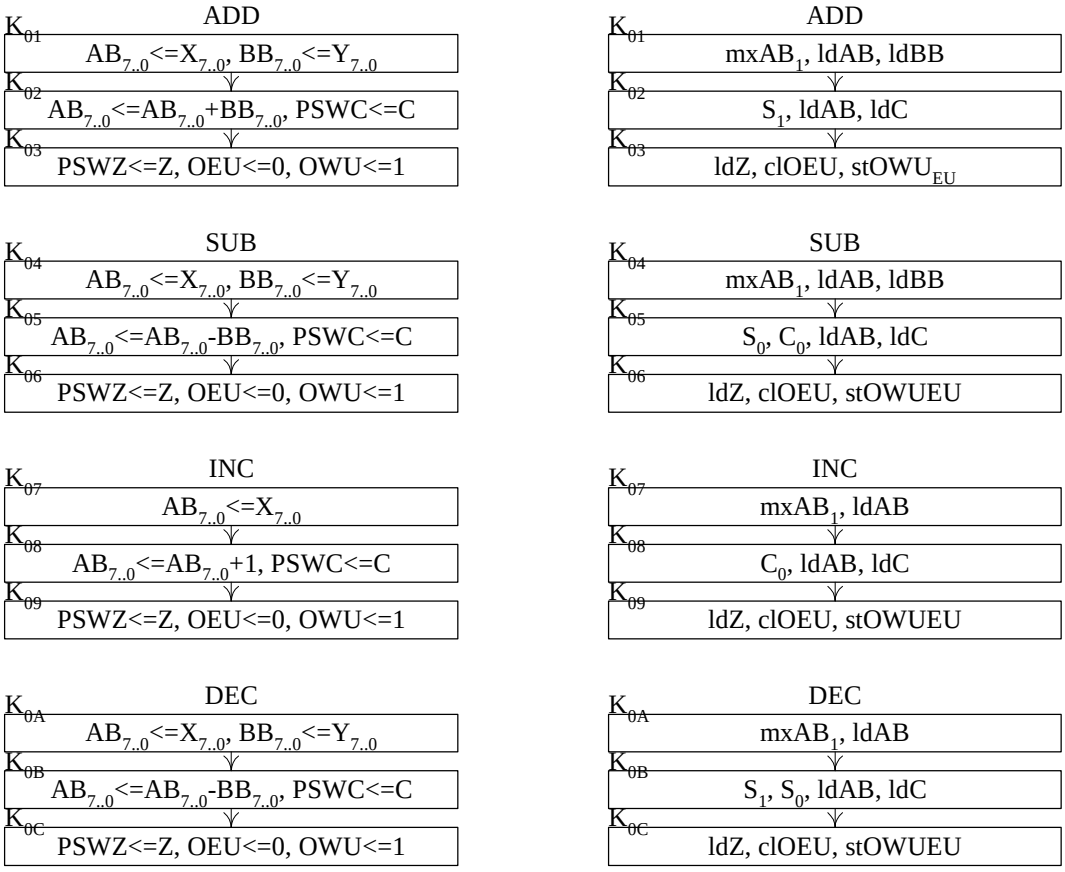


Slika 41 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa više operacija (prvi deo)

Tabela 17 Sekvenca upravljačkih signala po koracima

! U koraku $step_{00}$ (slika 41) se realizuje višestruki uslovni skok na jedan od koraka $step_{01}$, $step_{04}$, ..., $step_{82}$ u zavisnosti od toga koji od signala operacija **ADD**, **SUB**, ..., **DIVU** ima vrednost 1 i ostaje u koraku $step_{00}$ ukoliko signal **OEU** ima vrednost 0, pa stoga i svi signali **ADD**, **SUB**, ..., **DIVU** imaju vrednost 0 . !

$step_{00}$ *br* (*case* (**ADD**, **SUB**, **INC**, **DEC**, **AND**,**OR**, **XOR**, **NOT**,
ASR, **LSR**, **ROR**, **RORC**, **ASL**, **LSL**, **ROL**, **ROLC**,
MULU, **DIVU, NOP**)
then
(**ADD**, $step_{01}$), (**SUB**, ste_{04}), (**INC**, $step_{07}$), (**DEC**, $step_{0A}$),
(**AND**, $step_{0D}$), (**OR**, $step_{10}$), (**XOR**, $step_{13}$), (**NOT**, $step_{16}$),
(**ASR**, $step_{19}$), (**LSR**, $step_{19}$), (**ROR**, $step_{19}$), (**RORC**, $step_{19}$),
(**ASL**, $step_{1C}$), (**LSL**, $step_{1C}$), (**ROL**, $step_{1C}$), (**ROLC**, $step_{1C}$),
(**MULU**, $step_{1F}$), (**DIVU**, $step_{25}$), (**NOP**, $step_{00}$));



Slika 42 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa više operacija (drugi deo)

! ADD !

! U korak $step_{01}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije **ADD** ima vrednost 1 (slika 42). U koraku $step_{01}$ se vrednošću 1 signala **$mxAB_1$** i **ldAB** se sadržaj registra $X_{7..0}$ upisuje u registar $AB_{7..0}$, dok se vrednošću 1 signala **ldBB** sadržaj registra $Y_{7..0}$ upisuje u registar $BB_{7..0}$. U koraku $step_{02}$ se vrednosti sume i prenosa sadržaja registara $AB_{7..0}$ i $BB_{7..0}$, koji se vrednošću 1 signala **S_1**, **ldAB** i **ldC** formiraju na izlazima $ALU_{7..0}$ i C_8, upisuju u registar $AB_{7..0}$, i indikator PSWC. U koraku $step_{03}$ se vrednošću 1 signala ldZ u indikator PSWZ upisuje vrednost 1 ili 0 u zavisnosti od toga da li je rezultat operacije sabiranja takav da je vrednost sadržaja registra $AB_{7..0}$ nula ili različit od nule, respektivno. U ovom koraku se i vrednošću 1 signala **clOEU** u **$stOWU_{EU}$** u flip-flop OEU jedinice EU i flip-flop OWU jedinice WU upisuju vrednosti 0 i 1, čime se jedinica EU zaustavlja a jedinica WU pokreće. Pored toga iz koraka $step_{03}$ se bezulovno prelazi na poćeni koraka $step_{00}$. !

$step_{01}$ **$mxAB_1$**, **ldAB**, **ldBB**;
$step_{02}$ S_1, ldAB, ldC;
$step_{03}$ **ldZ**, clOEU, $stOWU_{EU}$,
br $step_{00}$;

! SUB !

! U korak $step_{04}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije **SUB** ima vrednost 1 (slika 42). Koraci $step_{04}$, $step_{05}$ i $step_{06}$ operacije SUB su slični koracima $step_{04}$, $step_{05}$ i $step_{06}$ operacije ADD. Jedina razlika je da se umesto vrednosti 1 signala **S_1** generiše vrednost 1 signala **S_0**, pa se na izlazima $ALU_{7..0}$ i C_8, pojavljuju vrednosti razlike i pozajmice sadržaja registara $AB_{7..0}$ i $BB_{7..0}$. Operacija oduzimanja se u ALU realizuje na takav način da vrednosti 1 i 0 signala C_8, ukazuju da nema i ima pozajmice, respektivno. !

$step_{04}$ **$mxAB_1$**, **ldAB**, **ldBB**;
$step_{05}$ S_0, C_0, ldAB, ldC;
$step_{06}$ **ldZ**, clOEU, $stOWU_{EU}$,
br $step_{00}$

! INC !

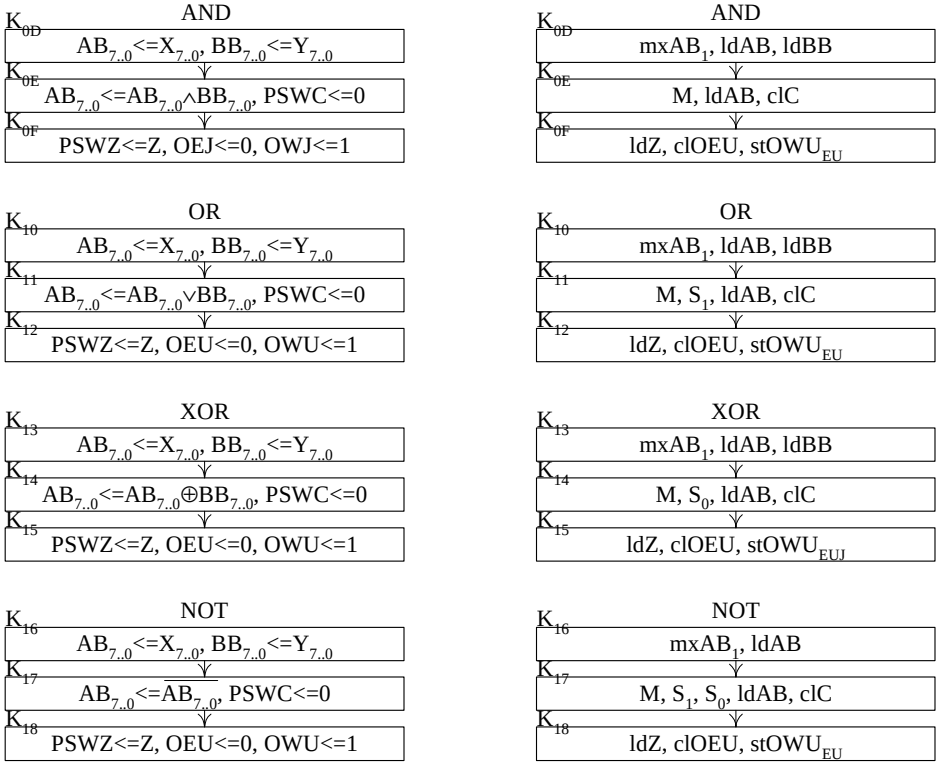
! U korak $step_{07}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije **INC** ima vrednost 1 (slika 42). U koraku $step_{07}$ se vrednošću 1 signala **$mxAB_1$** i **ldAB** se sadržaj registra $X_{7..0}$ upisuje u registar $AB_{7..0}$. U koraku $step_{08}$ se vrednosti sume i prenosa sadržaja registra $AB_{7..0}$ i vrednosti 1, koji se vrednošću 0 signala **S_1** i **S_0** i vrednošću 1 signala **ldAB** i **ldC** formiraju na izlazima $ALU_{7..0}$ i C_8, upisuju u registar $AB_{7..0}$ i indikator PSWC. Korak $step_{09}$ je isti kao korak $step_{03}$ operacije ADD. !

$step_{07}$ **$mxAB_1$**, **ldAB**, **ldBB**;
$step_{08}$ C_0, ldAB, ldC;
$step_{09}$ **ldZ**, clOEU, $stOWU_{EU}$,
br $step_{00}$;

! DEC !

! U korak $step_{0A}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije **DEC** ima vrednost 1 (slika 42). Koraci $step_{0A}$, $step_{0B}$ i $step_{0C}$ operacije DEC su slični koracima $step_{07}$, $step_{08}$ i $step_{09}$ operacije INC. Jedina razlika je da se umesto vrednosti 0 signala **S_1** i **S_0** generiše vrednost 1 ovih signala, pa se na izlazima $ALU_{7..0}$ i C_8 pojavljuju vrednosti razlike i pozajmice sadržaja registara $AB_{7..0}$ i vrednosti 1. Operacija oduzimanja se u ALU realizuje na takav način da vrednosti 1 i 0 signala C_8, ukazuju da nema i ima pozajmice, respektivno. !

$step_{0A}$ **$mxAB_1$**, **ldAB**, **ldBB**;
$step_{0B}$ S_1, S_0, ldAB, ldC;
$step_{0C}$ **ldZ**, clOEU, $stOWU_{EU}$,
br $step_{00}$;



Slika 43 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa više operacija (treci deo)

! AND !

! U korak $step_{0D}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije **AND** ima vrednost 1 (slika 43). U koraku $step_{0D}$ se vrednošću 1 signala **$mxAB_1$** i **ldAB** se sadržaj registra $X_{7..0}$ upisuje u registar $AB_{7..0}$, dok se vrednošću 1 signala **ldBB** sadržaj registra $Y_{7..0}$ upisuje u registar $BB_{7..0}$. U koraku $step_{0E}$ se vrednost koja predstavlja rezultat logičke I operacije sadržaja registara $AB_{7..0}$ i $BB_{7..0}$, koja se vrednošću 1 signala M i **ldAB** formira na izlazima $ALU_{7..0}$, upisuje u registar $AB_{7..0}$. U ovom koraku se vrednošću 1 signala clC upisuje vrednost 0 u indikator PSWC. U koraku $step_{0F}$ se vrednošću 1 signala ldZ u indikator PSWZ upisuje vrednost 1 ili 0 u zavisnosti od toga da li je rezultat operacije

sabiranja takav da je vrednost sadržaja registra $AB_{7..0}$ nula ili različit od nule, respektivno. U ovom koraku se i vrednošću 1 signala **clOEU** u **stOWU$_{EU}$** u flip-flop OEU jedinice EU i flip-flop OWU jedinice WU upisuju vrednosti 0 i 1, čime se jedinica EU zaustavlja a jedinica WU pokreće. Pored toga iz koraka $step_{03}$ se bezulovno prelazi na poćeni koraka $step_{00}$. !

$step_{0D}$ **mxAB$_1$**, **ldAB**, **ldBB**;
$step_{0E}$ M, ldAB, clC;
$step_{0F}$ **ldZ**, clOEU, stOWU$_{EU}$,
br $step_{00}$;

! OR !

! U korak $step_{10}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije **OR** ima vrednost 1 (slika 43). Koraci $step_{0D}$, $step_{0E}$ i $step_{0E}$ operacije OR su slični koracima $step_{0D}$, $step_{0E}$ i $step_{0F}$ operacije AND. Jedina razlika je da se u koraku $step_{0E}$ generiše vrednost 1 signala **S$_1$**, pa se na izlazima $ALU_{7..0}$ pojavljuju vrednosti koja predstavlja rezultat logičke ILI operacije sadržaja registara $AB_{7..0}$ i $BB_{7..0}$. !

$step_{10}$ **mxAB$_1$**, **ldAB**, **ldBB**;
$step_{11}$ M, **S$_1$**, ldAB, clC;
$step_{12}$ **ldZ**, clOEU, stOWU$_{EU}$,
br $step_{00}$;

! XOR !

! U korak $step_{13}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije **XOR** ima vrednost 1 (slika 43). Koraci $step_{13}$, $step_{14}$ i $step_{15}$ operacije XOR su slični koracima $step_{0D}$, $step_{0E}$ i $step_{0F}$ operacije AND. Jedina razlika je da se u koraku $step_{14}$ generiše vrednost 1 signala **S$_0$**, pa se na izlazima $ALU_{7..0}$ pojavljuju vrednosti koja predstavlja rezultat logičke ekskluzivno ILI operacije sadržaja registara $AB_{7..0}$ i $BB_{7..0}$. !

$step_{13}$ **mxAB$_1$**, **ldAB**, **ldBB**;
$step_{14}$ M, **S$_0$**, ldAB, clC;
$step_{15}$ **ldZ**, clOEU, stOWU$_{EU}$,
br $step_{00}$;

! NOT !

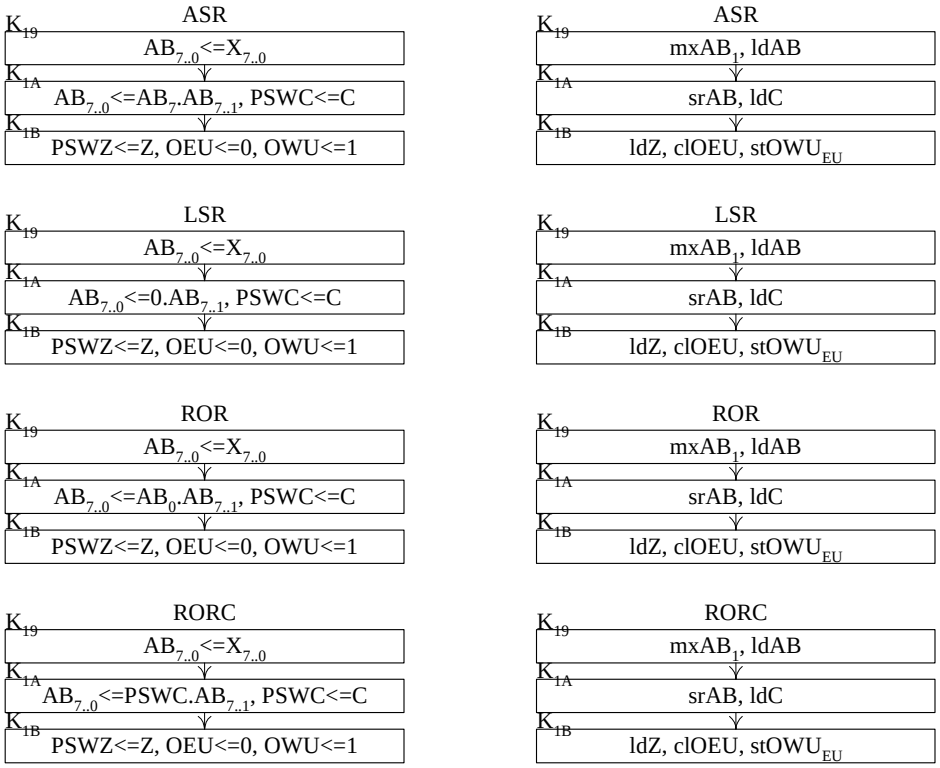
! U korak $step_{16}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije **NOT** ima vrednost 1 (slika 43). U koraku $step_{16}$ se vrednošću 1 signala **mxAB$_1$** i **ldAB** se sadržaj registra $X_{7..0}$ upisuje u registar $AB_{7..0}$. U koraku $step_{17}$ se invertovana vrednost sadržaja registra $AB_{7..0}$, koja se vrednošću 1 signala M, **S$_1$** i **S$_0$** formira na izlazima $ALU_{7..0}$, vrednošću 1 signala ldAB upisuje u registar $AB_{7..0}$. U ovom koraku se vrednošću 1 signala clC upisuje vrednost 0 u indikator PSWC. Korak $step_{18}$ je isti kao korak $step_{0F}$ operacije AND. !

$step_{16}$ **mxAB$_1$**, **ldAB**, **ldBB**;
$step_{17}$ M, **S$_1$**, S_0, ldAB, clC;
$step_{18}$ **ldZ**, clOEU, stOWU$_{EU}$,
br $step_{00}$;

! ASR, LSR, ROR, RORC !

! U korak $step_{19}$ se dolazi iz koraka $step_{00}$ ukoliko neki od signala operacije ASR, LSR, ROR ili RORC ima vrednost 1 (slika 44). U koraku $step_{19}$ se vrednošću 1 signala **mxAB$_1$** i **ldAB** sadržaj registra $X_{7..0}$ upisuje u registar $AB_{7..0}$. U koraku $step_{1A}$ se vrednošću 1 signala srA sadržaj registra $AB_{7..0}$ pomera udesno za jedno mesto i time u razrede $AB_{6..0}$ upisuju vrednosti razreda $AB_{7..1}$, respektivno. Tom prilikom se, u zavisnosti od toga koji od signala ASR, LSR, ROR ili RORC ima vrednost 1, u razred AB_7 upisuje prethodna vrednost razreda AB_7, 0, prethodna vrednost razreda AB_0 ili prethodna vrednost indikatora PSWC, respektivno. Tada se vrednošću 1 signala ldC u indikator PSWC upisuje prethodna vrednost razreda AB_0. Korak $step_{1B}$ je isti kao korak $step_{03}$ operacije ADD.!

$step_{19}$ **mxAB$_1$**, **ldAB**;
$step_{1A}$ srAB, ldC;
$step_{1B}$ **ldZ**, clOEU, stOWU$_{EU}$,
br $step_{00}$;

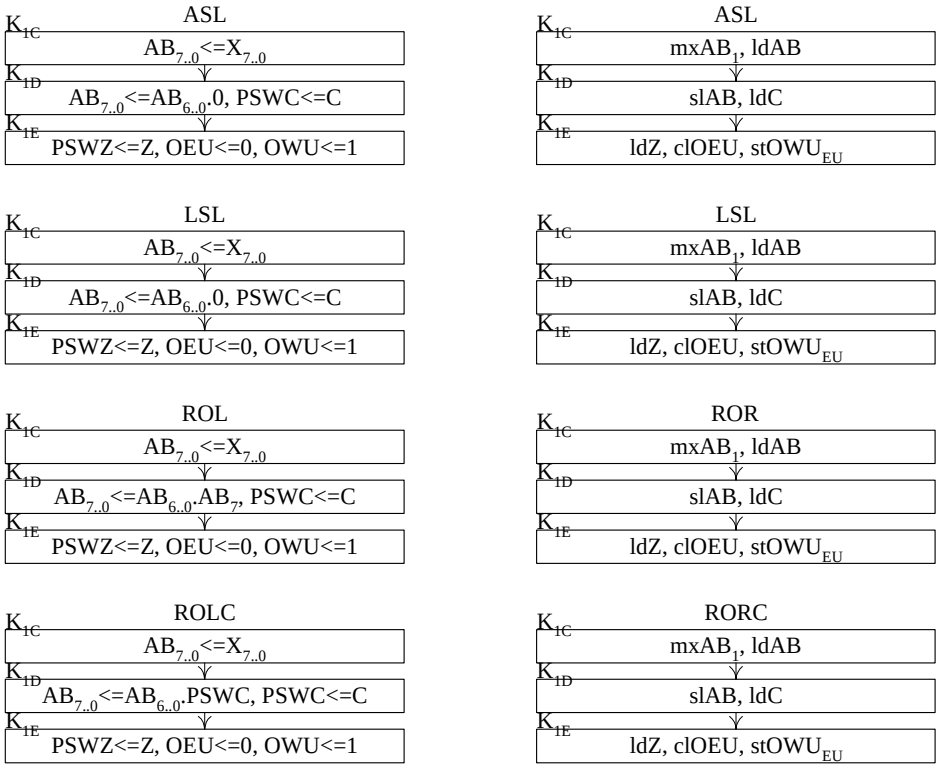


Slika 44 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa više operacija (cetvrti deo)

! ASL, LSL, ROL, ROLC !

! U korak $step_{1C}$ se dolazi iz koraka $step_{00}$ ukoliko neki od signala operacije ASL, LSL, ROL ili ROLC ima vrednost 1 (slika 45). U koraku $step_{1C}$ se vrednošću 1 signala **$mxAB_1$** i **ldAB** sadržaj registra $X_{7..0}$ upisuje u registar $AB_{7..0}$. U koraku $step_{1D}$ se vrednošću 1 signala slA sadržaj registra $AB_{7..0}$ pomera ulevoo za jedno mesto i time u razrede $AB_{7..1}$ upisuju vrednosti razreda $AB_{6..0}$, respektivno. Tom prilikom se, u zavisnosti od toga koji od signala ASL, LSL, ROL ili ROLC ima vrednost 1, u razred AB_0 upisuje 0, 0, prethodna vrednost razreda AB_7 ili prethodna vrednost indikatora PSWC. Tada se vrednošću 1 signala ldC u indikator PSWC upisuje prethodna vrednost razreda AB_7. Korak $step_{1B}$ je isti kao korak $step_{03}$ operacije ADD. !

$step_{1C}$ **$mxAB_1$**, **ldAB**;
$step_{1D}$ slAB, ldC;
$step_{1E}$ **ldZ**, clOEU, $stOWU_{EU}$,
br $step_{00}$;



Slika 45 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa više operacija (peti deo)

! MULU !

! U korak $step_{1F}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije MULU ima vrednost 1 (slika 46). U koraku $step_{1F}$ se vrednošću 1 signala **mxBB** i **ldBB** se sadržaj registra $X_{7..0}$ upisuje u registar $BB_{7..0}$, vrednošću 1 signala **ldMQ** sadržaj registra $Y_{7..0}$ upisuje u registar $MQ_{7..0}$., vrednošću 1 signala **clAB8** i **clAB** brišu sadržaji registara AB_8 i $AB_{7..0}$ i vrednošću 1 signala ldISC u brojač $ISC_{3..0}$ upisuje vrednost 8. !

$step_{1F}$ **mxBB**, **ldBB, ldMQ, clAB8, clAB, ldISC**;

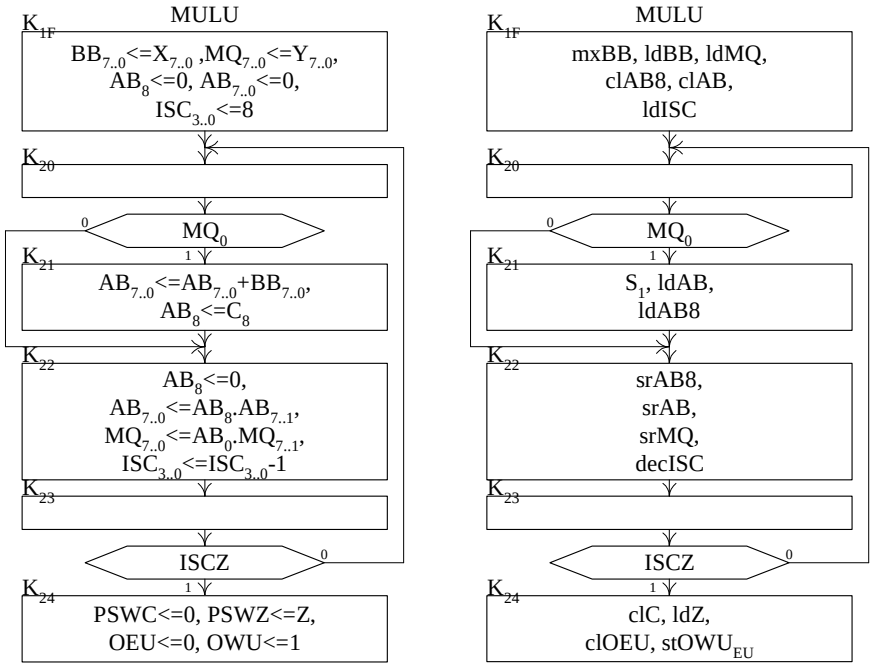
$step_{20}$ *br* (*if* $\overline{\mathbf{MQ_0}}$ *then* $step_{22}$);

$step_{21}$ $\mathbf{S_1}$, ldAB, ldAB8;

$step_{22}$ **srAB8**, srAB, srMQ, decISC;

$step_{23}$ *br* (*if* $\overline{\mathbf{ISCZ}}$ *then* $step_{20}$);

$step_{24}$ clC, **ldZ**, clOEU, $stOWU_{EU}$,
br $step_{00}$;

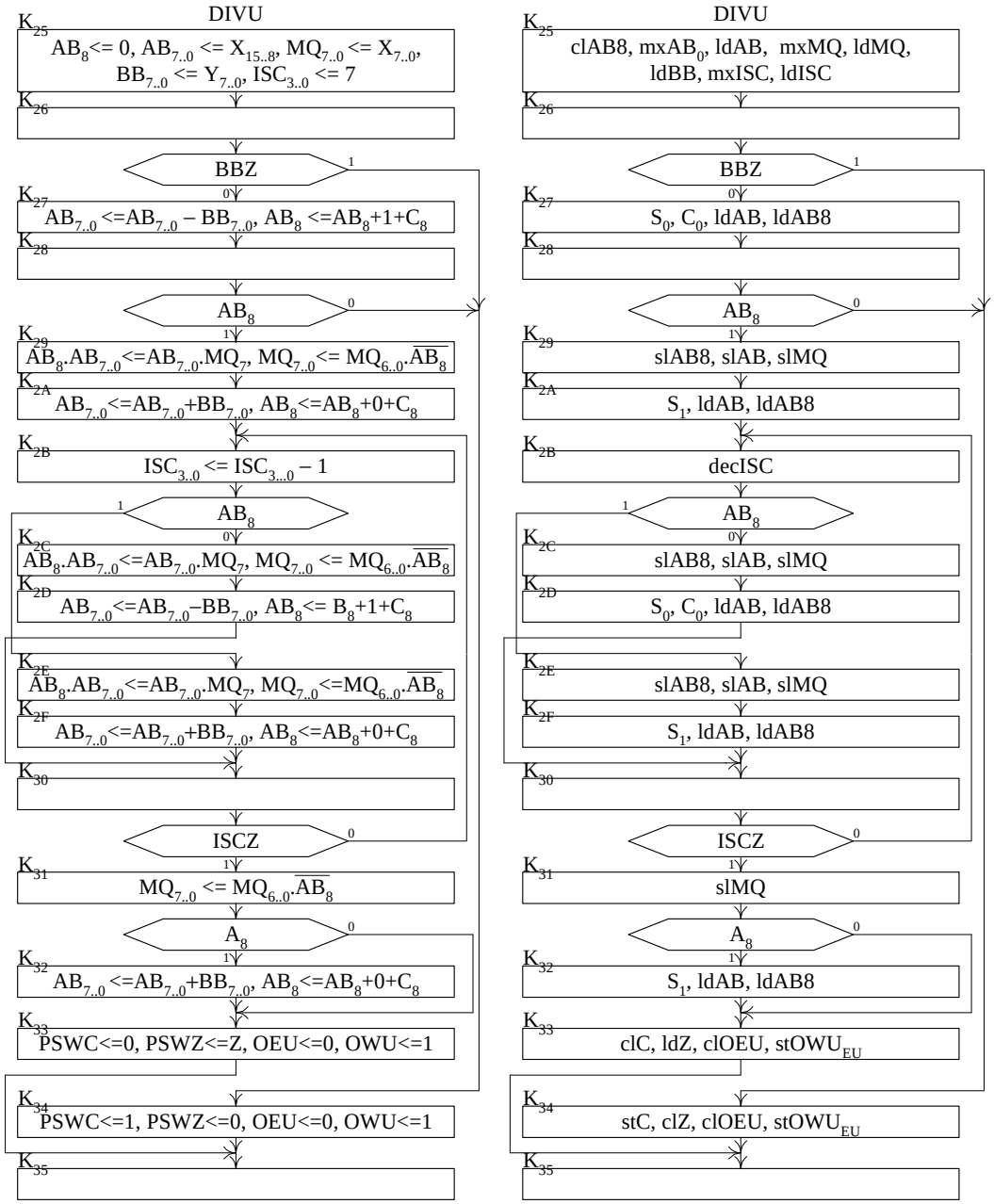


Slika 46 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa više operacija (šesti deo)

! DIVU !

! U korak $step_{25}$ se dolazi iz koraka $step_{00}$ ukoliko signal operacije DIVU ima vrednost 1 (slika 47). U koraku $step_{25}$ se vrednošću 1 signala **mxBB** i **ldBB** se sadržaj registra $X_{7..0}$ upisuje u registar $BB_{7..0}$, vrednošću 1 signala **ldMQ** sadržaj registra $Y_{7..0}$ upisuje u registar $MQ_{7..0}$., vrednošću 1 signala **clAB8** i **clAB** brišu sadržaji registara AB_8 i $AB_{7..0}$ i vrednošću 1 signala ldISC u brojač $ISC_{3..0}$ upisuje vrednost 8. !

$step_{25}$ **clAB8, mxAB$_0$**, **ldAB, mxMQ**, **ldMQ, ldBB, mxISC, ldISC**;
$step_{26}$ *br* (*if* **BBZ** *then* $step_{34}$);
$step_{27}$ **S$_0$**, C_0, ldAB, ldAB8;
$step_{28}$ *br* (*if* $\overline{\mathbf{AB_8}}$ *then* $step_{34}$);
$step_{29}$ **slAB8**, slAB, slMQ;
$step_{2A}$ **S$_1$**, ldAB, ldAB8;
$step_{2B}$ decISC
br (*if* AB_8 *then* $step_{2E}$);
$step_{2C}$ **slAB8**, slAB, slMQ;
$step_{2D}$ **S$_0$**, C_0, ldAB, ldAB8,
br $step_{30}$;
$step_{2E}$ **slAB8**, slAB, slMQ;
$step_{2F}$ **S$_1$**, ldAB, ldAB8;
$step_{30}$ *br* (*if* $\overline{\mathbf{ISCZ}}$ *then* $step_{2B}$);
$step_{31}$ **slMQ**,
br (*if* $\overline{\mathbf{AB_8}}$ *then* $step_{33}$);
$step_{32}$ **S$_1$**, ldAB, ldAB8,
$step_{33}$ clC, **ldZ**, clOEU, $stOWU_{EU}$,
br $step_{35}$;
$step_{34}$ stC, **ldZ**, clOEU, $stOWU_{EU}$,
$step_{35}$ *br* $step_{00}$;



Slika 47 Dijagrami toka mikrooperacija i upravljačkih signala za jedinicu sa više operacija (sedmi deo)

1.3.2 UPRAVLJAČKA JEDINICA

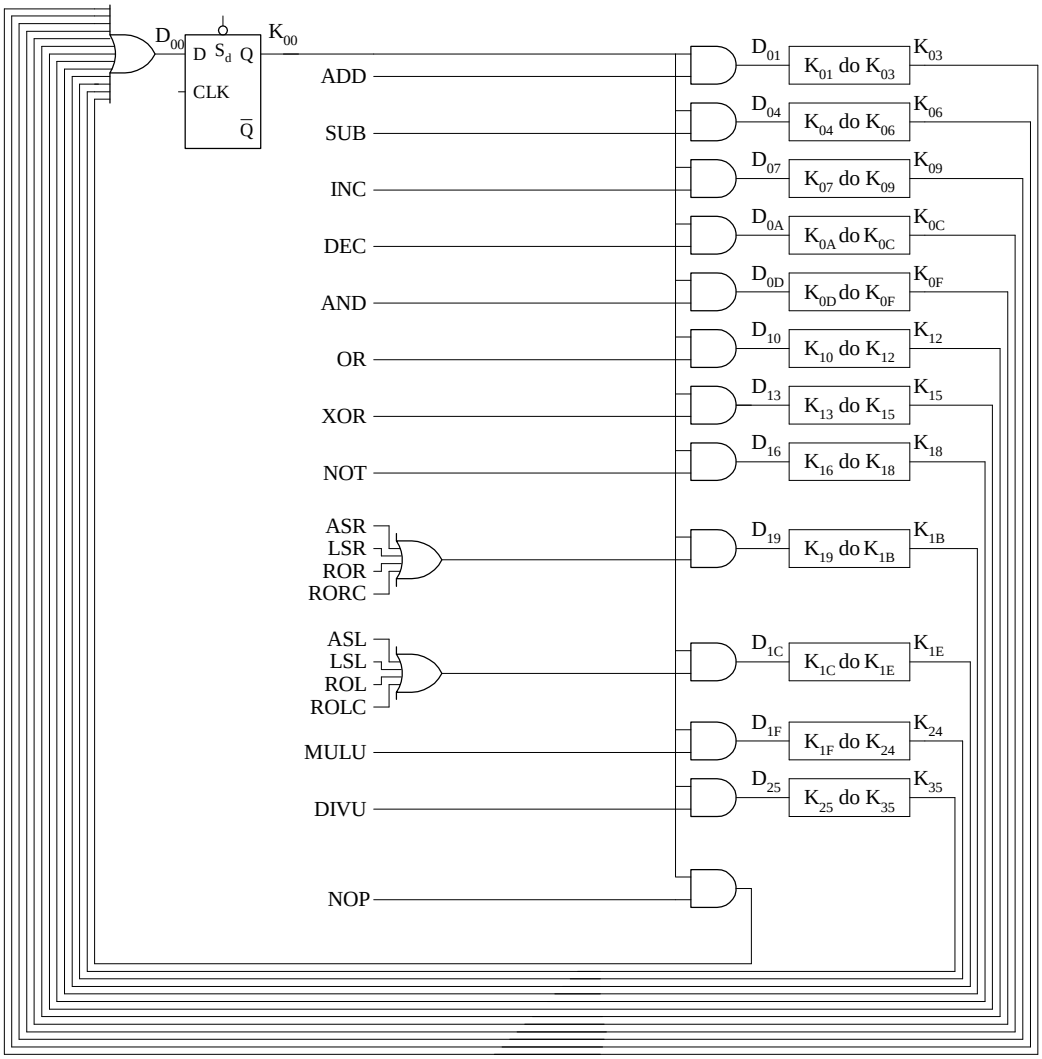
U ovom odeljku se razmatraju realizacije upravljačkih jedinica i to najpre sa ožičenim generisanjem upravljačkih signala a zatim i sa mikroprogramskim generisanjem upravljačkih signala.

1.3.2.1 OŽIČENO GENERISANJE SIGNALA

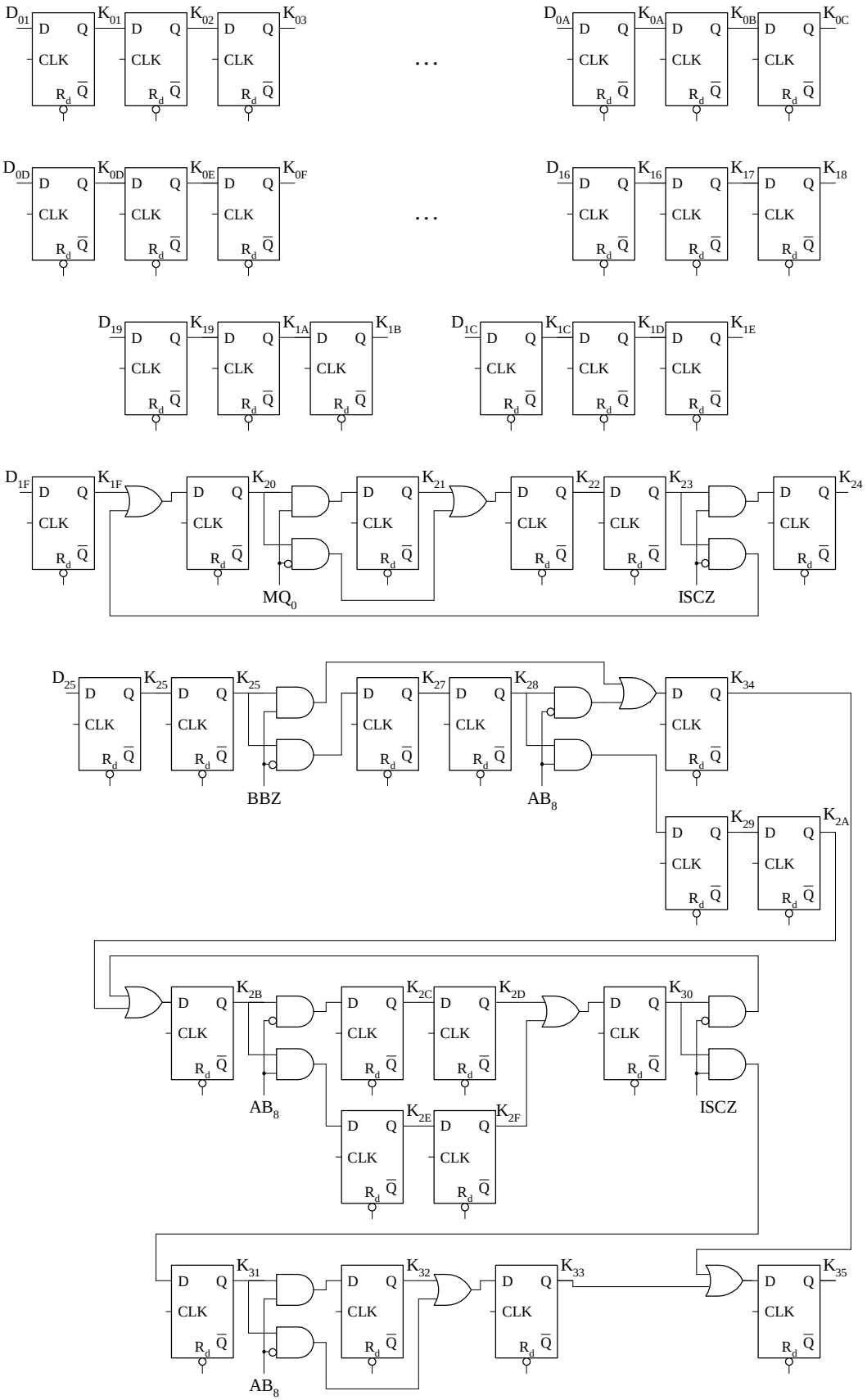
U ovom odeljku se daju postupci realizacija upravljačkih jedinica sa ožičenim generisanjem upravljačkih signala koje mogu da budu tipa šetajuća jedinica, sekvencijalna mreža i brojač koraka.

1.3.2.1.1 ŠETAJUĆA JEDINICA

Strukturna šema upravljačke jedinice tipa šetajuća jedinica (slike 48 i 49) je formirana na osnovu dijagrama toka upravljačkih signala (slike 41 do 47) i sekvence upravljačkih signala po koracima (tabela 17) po postupku datim u odeljku 1.2.2.1.1.



Slika 48 Upravljačka jedinica šetajuća jedinica (prvi deo)



Slika 49 Upravljačka jedinica tipa šetajuća jedinica za jedinicu sa više operacija (drugi deo)

Управљачки сигнали операционе јединице се генеришу према изразима

$mxAB_1 = K_{01} + K_{04} + K_{07} + K_{0A} + K_{0D} + K_{10} + K_{13} + K_{16} + K_{19} + K_{1C}$

$mxAB_0 = K_{25}$

$ldAB = K_{01}+K_{02}+K_{04}+K_{05}+K_{07}+K_{08}+K_{0A}+K_{0B}+K_{0D}+K_{0E}+K_{10}+K_{11}+K_{13}+K_{14}+K_{16}+K_{17}+K_{19}+K_{1C}+K_{21}+K_{25}+K_{27}+K_{2A}+K_{2D}+K_{2F}+K_{32}$

$M = K_{0E} + K_{11} + K_{14} + K_{17}$

$S_1 = K_{02} + K_{0B} + K_{11} + K_{17} + K_{21} + K_{2A} + K_{2F} + K_{32}$

$S_0 = K_{05} + K_{0B} + K_{14} + K_{17} + K_{27} + K_{2D}$

$srAB = K_{1A} + K_{22}$

$slAB = K_{1D} + K_{29} + K_{2C} + K_{2E}$

$decISC = K_{22} + K_{2B}$ itd.

у којима K_{00}, K_{01}, ..., K_{35} представљају сигнале са излаза флип-флопова придружених операционим блоковима у дијаграму тока управљачких сигнала (слике 41 do 47).

1.3.2.1.2 SEKVENCIJALNA MREŽA

Strukturna šema upravljačke jedinice tipa sekvencijalna mreža (slike 53 i 54) je formirana na osnovu dijagrama toka upravljačkih signala (slike 41 do 47) i sekvence upravljačkih signala po koracima (tabela 17) po postupku datim u odeljku 1.2.2.1.2. U okviru tog postupka najpre je konstruisana tablica (slika 51) koja sadrži tablicu stanja redukovanih dimenzija, tablicu prelaza/izlaza redukovanih dimenzija i kombinacionu tablicu funkcija pobude flip-flopova redukovanih dimenzija.

U tablici se pored simbličkih oznaka stanja sekvencijalne mreže K_{00}, K_{01}, .., K_{35}, koja su dodeljena operacionim blokovima K_{00}, K_{01}, ..., K_{35} u dijagramu toka upravljačkih signala (slike 41 do 47), pojavljuju i binarne vrednosti signala stanja sekvencijalne mreže. U opštem slučaju stanja se mogu kodirati na proizvoljan način. Međutim, stanja sekvencijalne prekidačke mreže su kodirana na takav način da se pri prelasku sekvencijalne mreže iz stanja u sadašnjem u stanje u sledećem trenutku menja što je moguće manji broj koordinata vektora stanja (slika 50). Za tako kodirana stanja kombinaciona mreža koja generiše signale pobuda za flip-flopove dodeljene koordinatama vektora stanja bi trebalo da bude najjednostavnija u odnosu na neke druge moguće načine kodiranja stanja.

$Q_4Q_5Q_6$ \ $Q_1Q_2Q_3$	000	001	011	010	110	111	101	100
000	K_{00}	K_{0F}	K_{10}	K_{1F}	K_{20}	K_{2F}	K_{30}	K_{3F}
001	K_{01}	K_{0E}	K_{11}	K_{1E}	K_{21}	K_{2E}	K_{31}	K_{3E}
011	K_{02}	K_{0D}	K_{12}	K_{1D}	K_{22}	K_{2D}	K_{32}	K_{3D}
010	K_{03}	K_{0C}	K_{13}	K_{1C}	K_{23}	K_{2C}	K_{33}	K_{3C}
110	K_{04}	K_{0B}	K_{14}	K_{1B}	K_{24}	K_{2B}	K_{34}	K_{3B}
111	K_{05}	K_{0A}	K_{15}	K_{1A}	K_{25}	K_{2A}	K_{35}	K_{3A}
101	K_{06}	K_{09}	K_{16}	K_{19}	K_{26}	K_{29}	K_{36}	K_{39}
100	K_{07}	K_{08}	K_{17}	K_{18}	K_{27}	K_{28}	K_{37}	K_{38}

Slika 50 Kodiranje stanja za upravljačku jedinicu tipa sekvencijalna mreža za jedinicu sa više operacija

Q	Z	Q(t+1)	X	D_1	D_2	D_3	D_4	D_5	D_6
K_{00}(000000)	/	K_{00}(000000)	NOP	0	0	0	0	0	0
		K_{01}(000001)	ADD	0	0	0	0	0	1
		K_{04}(000110)	SUB	0	0	0	1	1	0
		K_{07}(000100)	INC	0	0	0	1	0	0
		K_{0A}(001111)	DEC	0	0	1	1	1	1
		K_{0D}(001011)	AND	0	0	1	0	1	1
		K_{10}(011000)	OR	0	1	1	0	0	0
		K_{13}(011010)	XOR	0	1	1	0	1	0
		K_{16}(011101)	NOT	0	1	1	1	0	1
		K_{19}(010101)	ASR+LSR+ROR+RORC	0	1	0	1	0	1
		K_{1C}(010010)	ASL+LSL+ROL+ROLC	0	1	0	0	1	0
		K_{1F}(010000)	MULU	0	1	0	0	0	0
		K_{25}(110111)	DIVU	1	1	0	1	1	1
K_{01}(000001)	$mxAB_1$, ldAB, ldBB	K_{02}(000011)	1	0	0	0	0	1	1
K_{02}(000011)	S_1, ldAB, ldC	K_{03}(000010)	1	0	0	0	0	1	0
K_{03}(000010)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{04}(000110)	$mxAB_1$, ldAB, ldBB	K_{05}(000111)	1	0	0	0	1	1	1
K_{05}(000111)	S_0, C_0, ldAB, ldC	K_{06}(000101)	1	0	0	0	1	0	1
K_{06}(000101)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{07}(000100)	$mxAB_1$, ldAB	K_{08}(001100)	1	0	0	1	1	0	0
K_{08}(001100)	C_0, ldAB, ldC	K_{09}(001101)	1	0	0	1	1	0	1
K_{09}(001101)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{0A}(001111)	$mxAB_1$, ldAB	K_{0B}(001110)	1	0	0	1	1	1	0
K_{0B}(001110)	S_1, S_0, ldAB, ldC	K_{0C}(001010)	1	0	0	1	0	1	0
K_{0C}(001010)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{0D}(001011)	$mxAB_1$, ldAB, ldBB	K_{0E}(001001)	1	0	0	1	0	0	1
K_{0E}(001001)	M, ldAB, clC	K_{0F}(001000)	1	0	0	1	0	0	0
K_{0F}(001000)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{10}(011000)	$mxAB_1$, ldAB, ldBB	K_{11}(011001)	1	0	1	1	0	0	1
K_{11}(011001)	M, S_1, ldAB, ldC	K_{12}(011011)	1	0	1	1	0	1	1
K_{12}(011011)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{13}(011010)	$mxAB_1$, ldAB, ldBB	K_{14}(011110)	1	0	1	1	1	1	0
K_{14}(011110)	M, S_0, ldAB, ldC	K_{15}(011111)	1	0	1	1	1	1	1
K_{15}(011111)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{16}(011101)	$mxAB_1$, ldAB	K_{17}(011100)	1	0	1	1	1	0	0
K_{17}(011100)	M, S_1, S_0, ldAB, ldC	K_{18}(010100)	1	0	1	0	1	0	0
K_{18}(010100)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{19}(010101)	$mxAB_1$, ldAB	K_{1A}(010111)	1	0	1	0	1	1	1
K_{1A}(010111)	srAB, ldC	K_{1B}(010110)	1	0	1	0	1	1	0
K_{1B}(010110)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{1C}(010010)	$mxAB_1$, ldAB	K_{1D}(010011)	1	0	1	0	0	1	1
K_{1D}(010011)	slAB, ldC	K_{1E}(010001)	1	0	1	0	0	0	1
K_{1E}(010001)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0
K_{1F}(010000)	mxBB, ldBB, ldMQ, clAB8,clAB, ldISC	K_{20}(110000)	1	1	1	0	0	0	0
K_{20}(110000)	/	K_{21}(110001)	MQ_0	1	1	0	0	0	1
		K_{22}(110011)	$\overline{MQ_0}$	1	1	0	0	1	1
K_{21}(110001)	S_1, ldAB, ldAB8	K_{22}(110011)	1	1	1	0	0	1	1
K_{22}(110011)	srAB8, srAB, srMQ, decISC	K_{23}(110010)	1	1	1	0	0	1	0
K_{23}(110010)	/	K_{24}(110110)	ISCZ	1	1	0	1	1	0
		K_{20}(110000)	$\overline{ISCZ}$	1	1	0	0	0	0
K_{24}(110110)	clC, ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	0	0

Q	Z	Q(t+1)	X	D_1	D_2	D_3	D_4	D_5	D_6
K_{25}(110111)	clAB8, mxAB_0, ldAB, mxMQ, ldMQ, ldBB, mxISC, ldISC	K_{26}(110101)	1	1	1	0	1	0	1
K_{26}(110101)	/	K_{27}(110100)	$\overline{BBZ}$	1	1	0	1	0	0
		K_{34}(101110)	BBZ	1	0	1	1	1	0
K_{27}(110100)	S_0, C_0, ldAB, ldAB8	K_{28}(111100)	1	1	1	1	1	0	0
K_{28}(111100)	/	K_{29}(111101)	AB_8	1	1	1	1	0	1
		K_{34}(101110)	$\overline{AB_8}$	1	0	1	1	1	0
K_{29}(111101)	slAB8,slAB, slMQ	K_{2A}(111111)	1	1	1	1	1	1	1
K_{2A}(111111)	S_1, ldAB, ldAB8	K_{2B}(111110)	1	1	1	1	1	1	0
K_{2B}(111110)	decISC	K_{2C}(111010)	$\overline{AB_8}$	1	1	1	0	1	0
		K_{2E}(111001)	AB_8	1	1	1	0	0	1
K_{2C}(111010)	slAB8,slAB, slMQ	K_{2D}(111011)	1	1	1	1	0	1	1
K_{2D}(111011)	S_0, C_0, ldAB, ldAB8	K_{30}(101000)		1	0	1	0	0	0
K_{2E}(111001)	slAB8,slAB, slMQ	K_{2F}(111000)	1	1	1	1	0	0	0
K_{2F}(111000)	S_1, ldAB, ldAB8	K_{30}(101000)		1	0	1	0	0	0
K_{30}(101000)	/	K_{31}(101001)	ISCZ	1	0	1	0	0	1
		K_{2B}(111110)	$\overline{ISCZ}$	1	1	1	1	1	0
K_{31}(101001)	slMQ	K_{32}(101011)	AB_8	1	0	1	0	1	1
		K_{33}(101010)	$\overline{AB_8}$	1	0	1	0	1	0
K_{32}(101011)	S_1, ldAB, ldAB8	K_{33}(101010)	1	1	0	1	0	1	0
K_{33}(101010)	clC, ldZ, clOEU, stOWU_{EU}	K_{35}(101111)	1	1	0	1	1	1	1
K_{34}(101110)	stC, clZ, clOEU, stOWU_{EU}	K_{35}(101111)	1	1	0	1	1	1	1
K_{35}(101111)	/	K_{00}(000000)	1	0	0	0	0	0	0

Slika 51 Tablica za upravljačku jedinicu tipa sekvencijalna mreža realizovanu sa D flip-flopovima za jedinicu sa više operacija

Za realizaciju stanja sekvencijalne mreže se koriste D flip-flopovi, pa se u tablici (slika 51) pojavljuju i kolone sa vrednostima signala pobuda D_1, D_2, ..., D_6.

Da bi se pokazalo kako izbor flip-flopova za realizaciju stanja sekvencijalnu mreže može da utiče na složenost kombinacione mreže koja generiše signale pobuda, uzeto je i da se za realizaciju stanja umesto D flip-flopova koriste T flip-flopovi. Stoga je na osnovu tablice (slika 51) formirana tablica (slika 52) u kojoj se umesto kolona sa signalima pobuda D_1, D_2, ..., D_6 pojavljuju kolone sa signalima pobuda T_1, T_2, ..., T_6.

Iz tablice sa D flip-flopovima (slika 51) se utvrđuje da vrednost 1 signal D_1 ima 29 puta, signal D_2 ima 35 puta, signal D_3 ima 35 puta, signal D_4 ima 35 puta, signal D_5 ima 33 puta i signal D_6 ima 29 puta. Međutim, iz tablice sa T flip-flopovima (slika 52) se utvrđuje da vrednost 1 signal T_1 ima 4 puta, signal T_2 ima 18 puta, signal T_3 ima 15 puta, signal T_4 ima 19 puta, signal T_5 ima 31 puta i signal T_6 ima 35 puta. Na osnovu ove analize se utvrđuje da je strukturna šema kombinacione mreže koja generiše signale pobuda jednostavnija ukoliko se za realizaciju stanja sekvencijalne mreže koriste T flip-flopovi a ne D flip-flopovi. Na sličan način bi trebalo formirati i tablice sa RS i JK flip-flopovima.

Q	Z	Q(t+1)	X	T_1	T_2	T_3	T_4	T_5	T_6
K_{00}(000000)	/	K_{00}(000000)	NOP	0	0	0	0	0	0
		K_{01}(000001)	ADD	0	0	0	0	0	1
		K_{04}(000110)	SUB	0	0	0	1	1	0
		K_{07}(000100)	INC	0	0	0	1	0	0
		K_{0A}(001111)	DEC	0	0	1	1	1	1
		K_{0D}(001011)	AND	0	0	1	0	1	1
		K_{10}(011000)	OR	0	1	1	0	0	0
		K_{13}(011010)	XOR	0	1	1	0	1	0
		K_{16}(011101)	NOT	0	1	1	1	0	1
		K_{19}(010101)	ASR+LSR+ROR+RORC	0	1	0	1	0	1
		K_{1C}(010010)	ASL+LSL+ROL+ROLC	0	1	0	0	1	0
		K_{1F}(010000)	MULU	0	1	0	0	0	0
		K_{25}(110111)	DIVU	1	1	0	1	1	1
K_{01}(000001)	$mxAB_1$, ldAB, ldBB	K_{02}(000011)	1	0	0	0	0	1	0
K_{02}(000011)	S_1, ldAB, ldC	K_{03}(000010)	1	0	0	0	0	0	1
K_{03}(000010)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	0	1	0
K_{04}(000110)	$mxAB_1$, ldAB, ldBB	K_{05}(000111)	1	0	0	0	0	0	1
K_{05}(000111)	S_0, C_0, ldAB, ldC	K_{06}(000101)	1	0	0	0	0	1	0
K_{06}(000101)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	0	1	0	1
K_{07}(000100)	$mxAB_1$, ldAB	K_{08}(001100)	1	0	0	1	0	0	0
K_{08}(001100)	C_0, ldAB, ldC	K_{09}(001101)	1	0	0	0	0	0	1
K_{09}(001101)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	1	1	0	1
K_{0A}(001111)	$mxAB_1$, ldAB	K_{0B}(001110)	1	0	0	0	0	0	1
K_{0B}(001110)	S_1, S_0, ldAB, ldC	K_{0C}(001010)	1	0	0	0	1	0	0
K_{0C}(001010)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	1	0	1	0
K_{0D}(001011)	$mxAB_1$, ldAB, ldBB	K_{0E}(001001)	1	0	0	0	0	1	0
K_{0E}(001001)	M, ldAB, clC	K_{0F}(001000)	1	0	0	0	0	0	1
K_{0F}(001000)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	0	1	0	0	0
K_{10}(011000)	$mxAB_1$, ldAB, ldBB	K_{11}(011001)	1	0	0	0	0	0	1
K_{11}(011001)	M, S_1, ldAB, ldC	K_{12}(011011)	1	0	0	0	0	1	0
K_{12}(011011)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	1	1	0	1	1
K_{13}(011010)	$mxAB_1$, ldAB, ldBB	K_{14}(011110)	1	0	0	0	1	0	0
K_{14}(011110)	M, S_0, ldAB, ldC	K_{15}(011111)	1	0	0	0	0	0	1
K_{15}(011111)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	1	1	1	1	1
K_{16}(011101)	$mxAB_1$, ldAB	K_{17}(011100)	1	0	0	0	0	0	1
K_{17}(011100)	M, S_1, S_0, ldAB, ldC	K_{18}(010100)	1	0	0	1	0	0	0
K_{18}(010100)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	1	0	1	0	0
K_{19}(010101)	$mxAB_1$, ldAB	K_{1A}(010111)	1	0	0	0	0	1	0
K_{1A}(010111)	srAB, ldC	K_{1B}(010110)	1	0	0	0	0	1	0
K_{1B}(010110)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	1	0	1	1	0
K_{1C}(010010)	$mxAB_1$, ldAB	K_{1D}(010011)	1	0	0	0	0	0	1
K_{1D}(010011)	slAB, ldC	K_{1E}(010001)	1	0	0	0	0	1	0
K_{1E}(010001)	ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	0	1	0	0	0	1
K_{1F}(010000)	mxBB, ldBB, ldMQ, clAB8,clAB, ldISC	K_{20}(110000)	1	1	0	0	0	0	0
K_{20}(110000)	/	K_{21}(110001)	MQ_0	0	0	0	0	0	1
		K_{22}(110011)	$\overline{MQ_0}$	0	0	0	0	1	1
K_{21}(110001)	S_1, ldAB, ldAB8	K_{22}(110011)	1	0	0	0	0	1	0
K_{22}(110011)	srAB8, srAB, srMQ, decISC	K_{23}(110010)	1	0	0	0	0	0	1
K_{23}(110010)	/	K_{24}(110110)	ISCZ	0	0	0	1	0	0
		K_{20}(110000)	$\overline{ISCZ}$	0	0	0	0	1	0
K_{24}(110110)	clC, ldZ, clOEU, $stOWU_{EU}$	K_{00}(000000)	1	1	1	0	1	1	0

Q	Z	Q(t+1)	X	T_1	T_2	T_3	T_4	T_5	T_6
K_{25}(110111)	clAB8, mxAB_0, ldAB, mxMQ, ldMQ, ldBB, mxISC, ldISC	K_{26}(110101)	1	0	0	0	0	1	0
K_{26}(110101)	/	K_{27}(110100)	$\overline{BBZ}$	0	0	0	0	0	1
		K_{34}(101110)	BBZ	0	1	1	0	1	1
K_{27}(110100)	S_0, C_0, ldAB, ldAB8	K_{28}(111100)	1	0	0	1	0	0	0
K_{28}(111100)	/	K_{29}(111101)	AB_8	0	0	0	0	0	1
		K_{34}(101110)	$\overline{AB_8}$	0	1	0	0	1	0
K_{29}(111101)	slAB8,slAB, slMQ	K_{2A}(111111)	1	0	0	0	0	1	0
K_{2A}(111111)	S_1, ldAB, ldAB8	K_{2B}(111110)	1	0	0	0	0	0	1
K_{2B}(111110)	decISC	K_{2C}(111010)	$\overline{AB_8}$	0	0	0	1	0	0
		K_{2E}(111001)	AB_8	0	0	0	1	1	1
K_{2C}(111010)	slAB8,slAB, slMQ	K_{2D}(111011)	1	0	0	0	0	0	1
K_{2D}(111011)	S_0, C_0, ldAB, ldAB8	K_{30}(101000)		0	1	0	0	1	1
K_{2E}(111001)	slAB8,slAB, slMQ	K_{2F}(111000)	1	0	0	0	0	0	1
K_{2F}(111000)	S_1, ldAB, ldAB8	K_{30}(101000)		0	1	0	0	0	0
K_{30}(101000)	/	K_{31}(101001)	ISCZ	0	0	0	0	0	1
		K_{2B}(111110)	$\overline{ISCZ}$	0	1	0	1	1	0
K_{31}(101001)	slMQ	K_{32}(101011)	AB_8	0	0	0	0	1	0
		K_{33}(101010)	$\overline{AB_8}$	0	0	0	0	1	1
K_{32}(101011)	S_1, ldAB, ldAB8	K_{33}(101010)	1	0	0	0	0	0	1
K_{33}(101010)	clC, ldZ, clOEU, stOWU_{EU}	K_{35}(101111)	1	0	0	0	1	0	1
K_{34}(101110)	stC, clZ, clOEU, stOWU_{EU}	K_{35}(101111)	1	0	0	0	0	0	1
K_{35}(101111)	/	K_{00}(000000)	1	1	0	1	1	1	1

Slika 52 Tablica za upravljačku jedinicu tipa sekvencijalna mreža realizovanu sa T flip-flopovima za jedinicu sa više operacija

Uzeto je da se za realizaciju stanja sekvencijalne mreže koriste T flip-flopovi, pa se na osnovu tablice sa T flip-flopovima (slika 52) dobijaju sledeći izrazi za signale pobuda flip-flopova:

$$T_1 = K_{00}\cdot DIVU+K_{1F}+K_{24}+K_{35}$$

$$T_2 = K_{00}\cdot OR+ K_{00}\cdot XOR+ K_{00}\cdot NOT+K_{00}\cdot(ASR+LSR+ROR+RORC)+ K_{00}\cdot(ASL+LSL+ROL+ROLC)+K_{00}\cdot MULU+ K_{00}\cdot DIVU+K_{12}+K_{15}+ K_{18}+ K_{1B}+K_{1E}+K_{24}+K_{26}\cdot BBZ+K_{28}\cdot \overline{AB_8} +K_{2D}+K_{2F}+K_{30}\cdot \overline{ISCZ}$$

$$T_3 = K_{00}\cdot DEC+K_{00}\cdot AND+K_{00}\cdot OR+K_{00}\cdot XOR+K_{00}\cdot NOT+K_{07}+K_{09}+K_{0C}+K_{0F}+K_{12}+K_{15}+ K_{17}+K_{26}\cdot BBZ+K_{27}+K_{35}$$

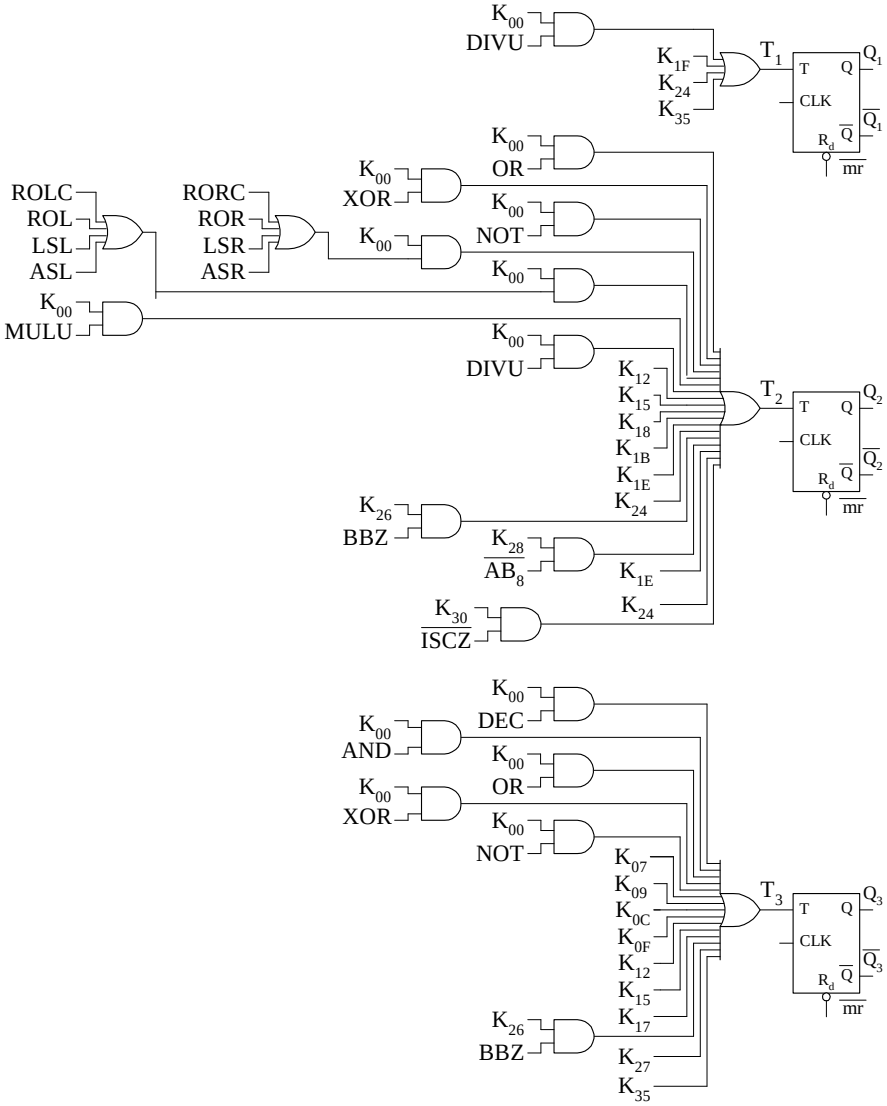
$$T_4 = K_{00}\cdot SUB+K_{00}\cdot INC+K_{00}\cdot DEC+K_{00}\cdot NOT+K_{00}\cdot(ASR+LSR+ROR+RORC)+K_{00}\cdot DIVU+ K_{06}+K_{09}+K_{0B}+K_{13}+K_{15}+K_{18}+K_{1B}+K_{23}\cdot ISCZ+K_{24}+K_{2B}+K_{30}\cdot \overline{ISCZ} +K_{33}+K_{35}$$

$$T_5 = K_{00}\cdot SUB+K_{00}\cdot DEC+K_{00}\cdot AND+K_{00}\cdot XOR+K_{00}\cdot(ASL+LSL+ROL+ROLC)+ K_{00}\cdot DIVU+K_{01}+K_{03}+K_{05}+K_{0C}+K_{0D}+K_{11}+K_{12}+K_{15}+K_{19}+K_{1A}+K_{1B}+K_{1D}+K_{20}\cdot \overline{MQ_0} + K_{21}+K_{23}\cdot \overline{ISCZ} +K_{24}+K_{25}+K_{26}\cdot BBZ+K_{28}\cdot \overline{AB_8} +K_{29}+K_{2B}\cdot AB_8+ K_{2D}+ K_{30}\cdot \overline{ISCZ} +K_{31}+K_{35}$$

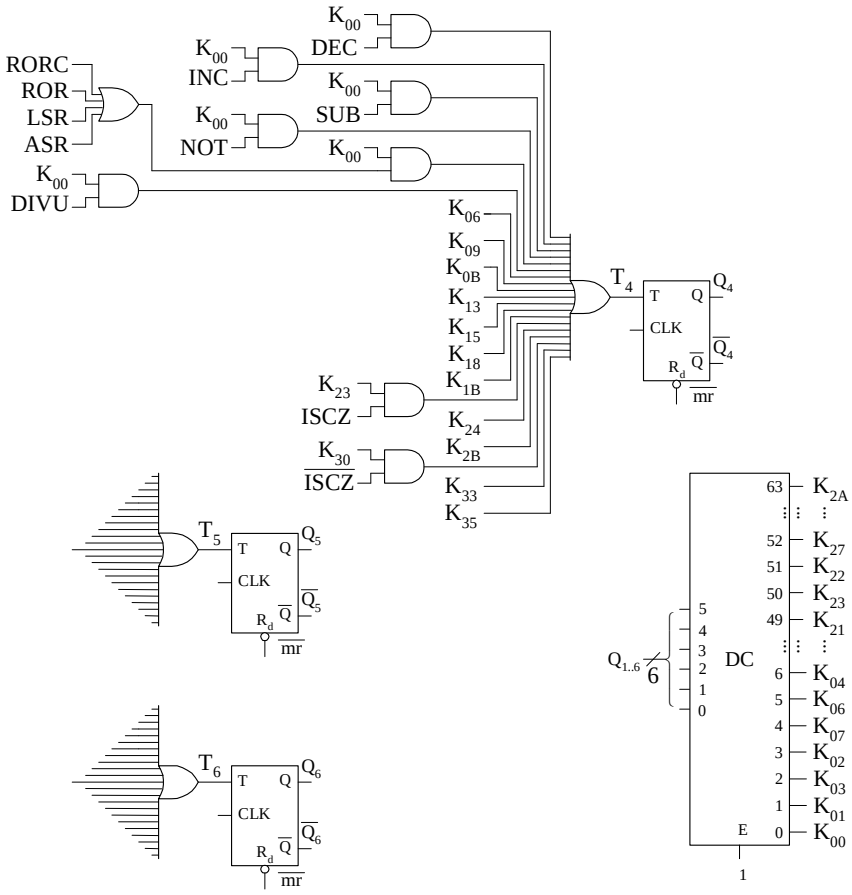
$$T_6 = K_{00}\cdot ADD+K_{00}\cdot DEC+K_{00}\cdot AND+K_{00}\cdot NOT+K_{00}\cdot(ASR+LSR+ROR+RORC)+ K_{00}\cdot DIVU+K_{02}+K_{04}+K_{06}+K_{08}+K_{09}+K_{0A}+K_{0E}+K_{10}+K_{12}+K_{14}+K_{15}+K_{16}+K_{1C}+K_{1E}+ K_{20}+K_{22}+K_{26}+K_{28}\cdot AB_8+K_{2A}+K_{2B}\cdot AB_8+K_{2C}+K_{2D}+K_{2E}+K_{30}\cdot ISCZ+K_{31}\cdot \overline{AB_8} + K_{32}+ K_{33}+K_{34}+K_{35}$$

Strukturna šema upravljačke jedinice (slike 53 i 54) sastoji iz kombinacione mreže koja je realizuje signale pobuda na osnovu izraza za T_1, T_2,..., T_6, flip-flopova Q_1, Q_2,..., Q_6 kojima se realizuju stanja sekvencijalne mreže i dekodera na čijim se izlazima formiraju signali stanja K_{00}, K_{10}, ..., K_{35}, pri čemu se izlazi kombinacione mreže T_1, T_2,..., T_6 vode na ulaze T flip-flopova Q_1, Q_2,..., Q_6, a izlazi flip-flopova na ulaze dekodera.

Upravljački signali operacione jedinice se generišu prema izrazima koji su identični sa odgovarajućim izrazima za upravljačku jedinice realizovanu kao šetajuća jedinica (odeljak 1.3.2.1.1). U datim izrazima sada K_{00}, K_{01}, ..., K_{35} представљају сигнале декодованих стања секвенцијалне мреже придружених операционим блоковима у дијаграму тока управљачких сигнала (slike 41 do 47). Сви остали управљачки сигнали имају вредност 0.



Slika 53 Upravljačka jedinice tipa sekvencijalna mreža za jedinicu sa više operacija (prvi deo)



Slika 54 Upravljačka jedinica tipa sekvencijalna mreža
za jedinicu sa više operacija (drugi deo)

1.3.2.1.3 BROJAČ KORAKA

Strukturna šema upravljačke jedinice je realizovana korišćenjem inkrementirajućeg brojača i dekodera (slike 55) na osnovu dijagrama toka upravljačkih signala (slike 41 do 47) i sekvence upravljačkih signala po koracima (tabela 17) po postupku datim u odeljku 1.2.2.1.3.

Upravljačka jedinica generiše dve vrste upravljačkih signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice. Upravljački signali operacione jedinice se koriste u operacionoj jedinici radi izvršavanja mikrooperacija. Upravljački signali upravljačke jedinice se koriste u upravljačkoj jedinici radi inkrementiranja brojača koraka ili upisa nove vrednosti u brojač koraka i radi generisanja vrednosti za upis u brojač koraka.

Upravljački signali operacione jedinice bi mogli da se generišu na osnovu sekvence upravljačkih signala po koracima (tabela 17). Za svaki upravljački signal operacione jedinice trebalo bi proći kroz sekvencu upravljačkih signala po koracima, tražiti korake u kojima se pojavljuje dati signal i izraz za dati signal formirati kao uniju signala dekodovanih stanja brojača koraka koji odgovaraju koracima u kojima se pojavljuje dati signal.

Upravljački signali upravljačke jedinice se ne mogu generisati na osnovu sekvence upravljačkih signala po koracima (tabela 17), jer se u njoj ne pojavljuju upravljački signali upravljačke jedinice, već samo iskazi za skokove. Zbog toga je potrebno na osnovu sekvence upravljačkih signala po koracima formirati sekvencu upravljačkih signala za upravljačku jedinicu ožičene realizacije. U njoj treba da se pored upravljačkih signala operacione jedinice pojave i upravljački signali upravljačke jedinice neophodni za realizaciju bezuslovnih, uslovnih i višestruih uslovnih skokova specificiranih iskazima za skokove. Prilikom njenog

formiranja primenjuje se različiti postupak za upravljačke signale operacione jedinice i za upravljačke signale upravljačke jedinice.

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

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

Umesto iskaza *br* $step_A$ treba u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije staviti signal bezuslovnog skoka koji određuje da se bezuslovno prelazi na korak $step_A$ i signal $\mathbf{val_A}$ koji određuje da treba formirati binarnu vrednost A za upis u brojač koraka. Simbolička oznaka signala bezuslovnog skoka je **bruncnd**. Koraci $step_i$ u kojima se bezuslovni skokovi javljaju, koraci $step_A$ na koje se bezuslovno skače, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A u heksadecimalnom dati su u tabeli 18.

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

$step_i$	$step_A$	$\mathbf{val_A}$	A
$step_{03}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{06}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{09}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{0C}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{0F}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{12}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{15}$	$step_{00}$	$\mathbf{val_{00}}$	00

$step_i$	$step_A$	$\mathbf{val_A}$	A
$step_{18}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{1B}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{1E}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{24}$	$step_{00}$	$\mathbf{val_{00}}$	00
$step_{2D}$	$step_{30}$	$\mathbf{val_{30}}$	30
$step_{33}$	$step_{35}$	$\mathbf{val_{35}}$	35
$step_{35}$	$step_{00}$	$\mathbf{val_{00}}$	00

Umesto iskaza *br* (*if* **uslov** *then* $step_A$) treba u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije staviti signal uslovnog skoka pridružen signalu **uslov** koji treba da ima vrednost 1 da bi se realizovao prelaz na korak $step_A$ i signal $\mathbf{val_A}$ koji određuje da treba formirati binarnu vrednost A za upis u brojač koraka u slučaju da signal **uslov** ima vrednost 1. Simboličke oznake pridruženih signala uslovnih skokova i signala uslova za sve iskaze ovog tipa koji se javljaju u sekvenci upravljačkih signala po koracima, dati su u tabeli 19. Koraci $step_i$ u kojima se uslovni skokovi javljaju, signali uslova **uslov**, koraci $step_A$ na koje se uslovno skače, simboličke oznake signala $\mathbf{val_A}$ i vrednosti A u heksadecimalnom obliku dati su u tabeli 20.

Tabela 19 Signali uslovnih skokova i signali uslova

signal uslovnog skoka	signal uslova
brnotMQ0	$\overline{\mathbf{MQ_0}}$
brnotISCZ	$\overline{\mathbf{ISCZ}}$
brBBZ	**BBZ**
brnotAB8	$\overline{\mathbf{AB_8}}$
brAB8	$\mathbf{AB_8}$

Tabela 20 Koraci $step_i$, signali uslova **uslov**, koraci $step_A$, signali $\mathbf{val_A}$ i vrednosti A za uslovne skokove

$step_i$	uslov	$step_A$	$\mathbf{val_A}$	A
$step_{20}$	$\overline{\mathbf{MQ_0}}$	$step_{22}$	$\mathbf{val_{22}}$	22
$step_{23}$	$\overline{\mathbf{ISCZ}}$	$step_{20}$	$\mathbf{val_{20}}$	20
$step_{26}$	**BBZ**	$step_{34}$	$\mathbf{val_{34}}$	34
$step_{28}$	$\overline{\mathbf{AB_8}}$	$step_{34}$	$\mathbf{val_{34}}$	34
$step_{2B}$	$\mathbf{AB_8}$	$step_{2E}$	$\mathbf{val_{2E}}$	2E
$step_{30}$	$\overline{\mathbf{ISCZ}}$	$step_{2B}$	$\mathbf{val_{2B}}$	2B
$step_{31}$	$\overline{\mathbf{AB_8}}$	$step_{33}$	$\mathbf{val_{33}}$	33

Umesto iskaza *br* (*case* ($\mathbf{uslov_1}$, ..., $\mathbf{uslov_n}$) *then* ($\mathbf{uslov_1}$, $step_{A1}$), ..., ($\mathbf{uslov_n}$, $step_{An}$) treba u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije staviti signal višestrukog uslovnog skoka pridružen signalima $\mathbf{uslov_1}$, $\mathbf{uslov_2}$,..., $\mathbf{uslov_n}$ od kojih jedan treba da ima vrednost 1 da bi se realizovao prelazak na jedan od koraka $step_{A1}$, $step_{A2}$, ..., $step_{An}$. Korak $step_i$ u kome se višestruki uslovni skok javlja i simbolička oznaka pridruženog signala višestrukog uslovnog skoka za korak ovog tipa koji se javlja u sekvenci upravljačkih signala po koracima je dat u tabeli 21. Signali uslova $\mathbf{uslov_1}$, $\mathbf{uslov_2}$,..., $\mathbf{uslov_n}$, koraci $step_{A1}$, $step_{A2}$, ..., $step_{An}$ na koje se uslovno skače i vrednosti A1, A2, ..., An u heksadecimalnom koje treba da se upišu u brojač koraka u zavisnosti od toga koji od signala uslova $\mathbf{uslov_1}$, $\mathbf{uslov_2}$,..., $\mathbf{uslov_n}$ ima vrednost 1 za višestruki uslovni skok u koraku $step_{00}$ dati su tabeli 22.

Tabela 21 Korak sa višestrukim uslovnim skokom i signal višestrukog uslovnog skoka

$step_i$	signal
$step_{00}$	**brop**

Tabela 22 Signali uslova, koraci na koje se skače i vrednosti za upis u brojač koraka za višestruki uslovni skok u koraku $step_{00}$

uslov	$step_A$	A
NOP	$step_{00}$	00
ADD	$step_{01}$	01
SUB	$step_{04}$	04
INC	$step_{07}$	07
DEC	$step_{0A}$	0A
AND	$step_{0D}$	0D
OR	$step_{10}$	10

uslov	$step_A$	A
XOR	$step_{13}$	13
NOT	$step_{16}$	16
ASR	$step_{19}$	19
LSR	$step_{19}$	19
ROR	$step_{19}$	19
RORC	$step_{19}$	19

uslov	$step_A$	A
ASL	$step_{1C}$	1C
LSL	$step_{1C}$	1C
ROL	$step_{1C}$	1C
ROLC	$step_{1C}$	1C
MULU	$step_{1F}$	1F
DIVU	$step_{25}$	25

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu ožičene realizacije signal bezuslovnog skoka **bruncnd**, signali uslovnih skokova (tabela 19), signal višestrukog uslovnog skoka (tabela 21) i signali $\mathbf{val_A}$ za bezuslovne (tabela 18) i uslovne (tabela 20) skokove.

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela tabela 17), formirana sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 23). Jedna linija u toj sekvenci ima sledeću formu: na levoj strani nalazi se signal dekodovanog stanja brojača koraka, u sredini je niz upravljačkih signala operacione i upravljačke jedinice koji imaju vrednost 1 kada dati signal dekodovanog stanja brojača koraka ima vrednost 1, dok komentar, tamo gde postoji, počinje uskličnikom (!) i proteže se do sledećeg uskličnika (!).

Upravljački signali operacione jedinice i upravljačke jedinice se generišu na identičan način na osnovu sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 23). Za svaki upravljački signal operacione jedinice i upravljačke jedinice treba proći kroz sekvencu upravljačkih signala za upravljačku jedinicu ožičene realizacije, tražiti korake u kojima se pojavljuje dati signal i izraz za dati signal formirati kao uniju signala dekodovanih stanja brojača koraka koji odgovaraju koracima u kojima se pojavljuje dati signal.

Tabela 23 Sekvenca upravljačkih signala za upravljačku jedinicu ožičene realizacije

K_{00} **brop**;
! ADD !
K_{01} **mxAB$_1$**, **ldAB**, **ldBB**;
K_{02} S_1, ldAB, ldC;
K_{03} **ldZ**, clOEU, stOWU$_{EU}$,
bruncnd, val$_{00}$;
! SUB !
K_{04} **mxAB$_1$**, **ldAB**, **ldBB**;
K_{05} S_0, C_0, ldAB, ldC;
K_{06} **ldZ**, clOEU, stOWU$_{EU}$,
bruncnd, val$_{00}$;
! INC !
K_{07} **mxAB$_1$**, **ldAB**;
K_{08} C_0, ldAB, ldC;
K_{09} **ldZ**, clOEU, stOWU$_{EU}$,
bruncnd, val$_{00}$;
! DEC !
K_{0A} **mxAB$_1$**, **ldAB**;
K_{0B} **S$_1$**, S_0, ldAB, ldC;
K_{0C} **ldZ**, clOEU, stOWU$_{EU}$,
bruncnd, val$_{00}$;
! AND !
K_{0D} **mxAB$_1$**, **ldAB**, **ldBB**;
K_{0E} M, ldAB, clC;
K_{0F} **ldZ**, clOEU, stOWU$_{EU}$,
bruncnd, val$_{00}$;
! OR !
K_{10} **mxAB$_1$**, **ldAB**, **ldBB**;
K_{11} M, **S$_1$**, ldAB, clC;
K_{12} **ldZ**, clOEU, stOWU$_{EU}$,
bruncnd, val$_{00}$;
! XOR !
K_{13} **mxAB$_1$**, **ldAB**, **ldBB**;
K_{14} M, **S$_0$**, ldAB, clC;
K_{15} **ldZ**, clOEU, stOWU$_{EU}$,
bruncnd, val$_{00}$;
! NOT !
K_{16} **mxAB$_1$**, **ldAB**;
K_{17} M, **S$_1$**, S_0, ldAB, clC;
K_{18} **ldZ**, clOEU, stOWU$_{EU}$,
bruncnd, val$_{00}$;

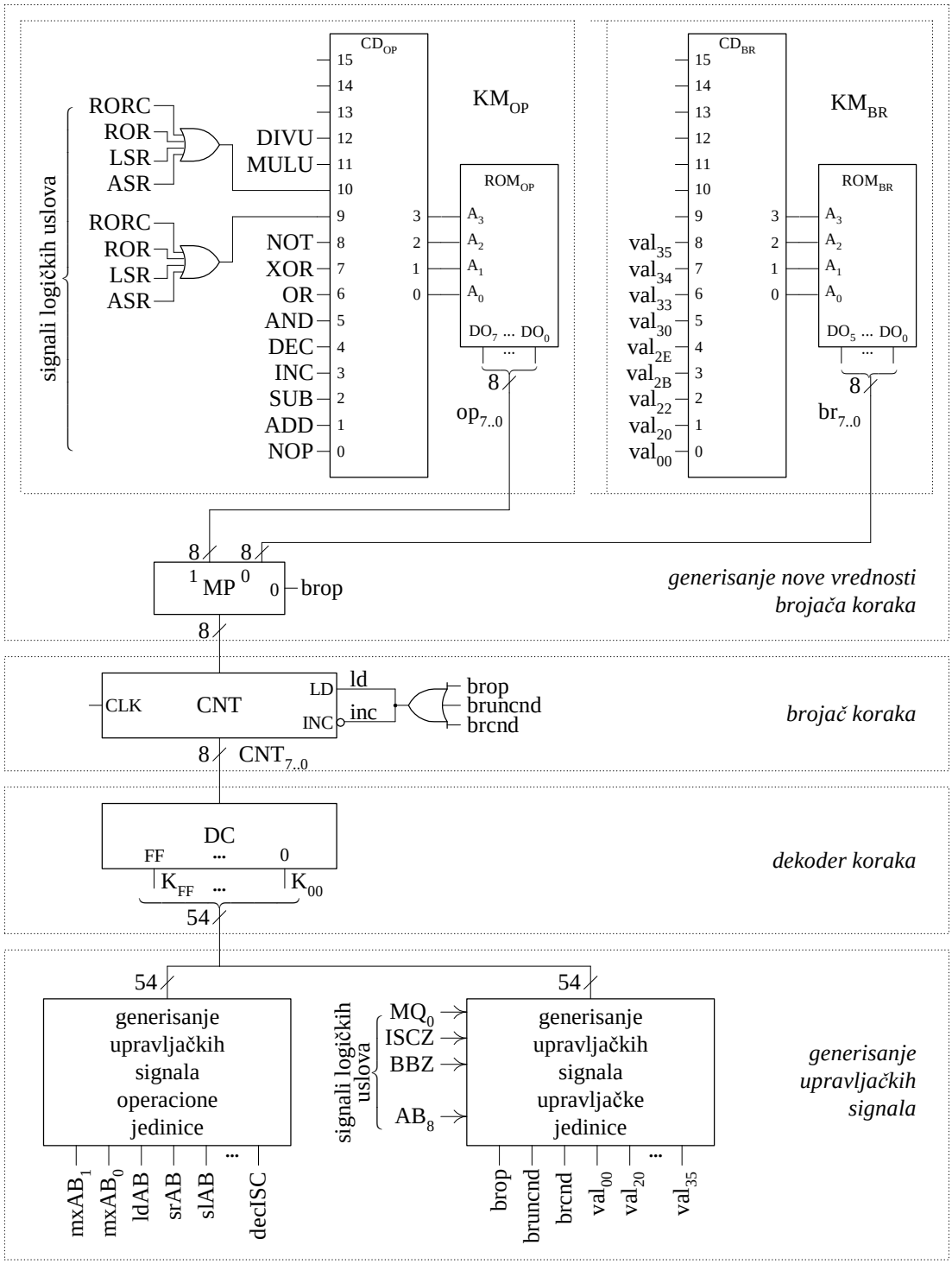
! ASR, LSR, ROR, RORC !
K_{19} **$mxAB_1$**, **ldAB**;
K_{1A} srAB, ldC;
K_{1B} **ldZ**, clOEU, $stOWU_{EU}$,
bruncnd, val_{00};

! ASL, LSL, ROL, ROLC !
K_{1C} **$mxAB_1$**, **ldAB**;
K_{1D} slAB, ldC;
K_{1E} **ldZ**, clOEU, $stOWU_{EU}$,
bruncnd, val_{00};

! MULU !
K_{1F} **mxBB**, **ldBB, ldMQ, clAB8, clAB, ldISC**;
K_{20} **brnotMQ0**, val_{22};
K_{21} **S_1**, ldAB, ldAB8;
K_{22} **srAB8**, srAB, srMQ, decISC;
K_{23} **brnotISCZ**, val_{20};
K_{24} clC, **ldZ**, clOEU, $stOWU_{EU}$,
bruncnd, val_{00};

! DIVU !
K_{25} **clAB8, $mxAB_0$**, **ldAB, mxMQ**, **ldMQ, ldBB, mxISC, ldISC**;
K_{26} **brBBZ**, val_{34};
K_{27} **S_0**, C_0, ldAB, ldAB8;
K_{28} **brnotAB8**, val_{34};
K_{29} **slAB8**, slAB, slMQ;
K_{2A} **S_1**, ldAB, ldAB8;
K_{2B} decISC,
brAB8, val_{2E};
K_{2C} **slAB8**, slAB, slMQ;
K_{2D} **S_0**, C_0, ldAB, ldAB8,
bruncnd, val_{30};
K_{2E} **slAB8**, slAB, slMQ;
K_{2F} **S_1**, ldAB, ldAB8;
K_{30} brnotISCZ, val_{2B};
K_{31} **slMQ**,
brnotAB8, val_{33};
K_{32} **S_1**, ldAB, ldAB8,
K_{33} clC, **ldZ**, clOEU, $stOWU_{EU}$,
bruncnd, val_{35};
K_{34} stC, **ldZ**, clOEU, $stOWU_{EU}$,
K_{35} **bruncnd**, val_{00};

Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 55. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti brojača koraka*, blok *brojač koraka*, blok *dekoder koraka* i blok *generisanje upravljačkih signala*. Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.



Slika 55 Upravljačka jedinica tipa brojač koraka za jedinicu sa više operacija

Blok *generisanje nove vrednosti brojača koraka* se sastoji od kombinacionih mreža KM_{OP} i KM_{BR} sa multiplekserom MP i služi za generisanje i selekciju vrednosti koju treba upisati u brojač koraka $CNT_{7...0}$. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikrooperacija. Vrednosti koje treba upisati u brojač koraka generišu se na dva načina i to pomoću kombinacione mreže KM_{OP} koja formira signale $\mathbf{op_{7...0}}$ i kombinacione mreže KM_{BR} koja formira signale $\mathbf{br_{7...0}}$. Selekcija jedne od dve grupe signala koji daju novu vrednost brojača koraka $CNT_{7...0}$ obezbeđuje se signalom **brop** i to signali $\mathbf{op_{7...0}}$ ako signal **brop** ima vrednost 1 i signali $\mathbf{br_{7...0}}$ ako signal **brop** ima vrednost 0.

Kombinacionom mrežom KM_{OP} generišu se vrednosti (tabela 22) za realizaciju višestrukog uslovnog skoka u koraku $step_{00}$ sekvence upravljačkih signala. Kombinacionu mrežu KM_{OP} čine koder CD_{OP} i ROM memorija ROM_{OP}. Na ulaze 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 i 12 kodera CD_{OP} vezani su signali **NOP**, **ADD**, **SUB**, INC, DEC, AND, OR, XOR, NOT, (ASR+LSR+ROR+RORC), (ASL+LSL+ROL+ROLC), MULU i **DIVU**, respektivno, a na adresama 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 i 12 memorije ROM_{OP} nalaze se vrednosti 00, 01, 04, 07, **0A**, **0D**, **10**, **13**, **16**, **19**, **1C,** 1F i **25**, respektivno. U zavisnosti od toga koji od signala **NOP**, **ADD**, ..., **DIVU** ima vrednost 1 na izlazima kodera se pojavljuje adresa memorijske lokacije sa koje se čita i na linijama $op_{7...0}$ pojavljuje vrednost koji treba upisati u brojač koraka. S obzirom da vrednost 1 signala dekovanog stanje brojača koraka K_{00} daje vrednost 1 signala višestrukog uslovnog skoka **brop**, vrednost na linijama $\mathbf{op_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima brojača koraka $CNT_{7...0}$.

Kombinacionom mrežom KM_{BR} generišu se vrednosti za upis u brojač koraka $CNT_{7...0}$ za bezuslovne skokove (tabela 18) i uslovne skokove (tabela 20) u sekvenci upravljačkih signala po koracima. Kombinacionu mrežu KM_{BR} čine koder CD_{BR} i ROM memorija ROM_{BR}. Na ulaze 0, 1, 2, 3, 4, 5, 6, 7 i 8 kodera CD_{BR} vezani su signali $\mathbf{val_{00}}$, $\mathbf{val_{20}}$, $\mathbf{val_{22}}$, $\mathbf{val_{2B}}$, $\mathbf{val_{2E}}$, $\mathbf{val_{30}}$, $\mathbf{val_{33}}$, $\mathbf{val_{34}}$ i $\mathbf{val_{35}}$, respektivno, a na adresama 0, 1, 2, 3, 4, 5, 6, 7 i 8 memorije ROM_{BR} nalaze se vrednosti 00, 20, 22, 2B, **2E**, **30**, **33**, **34** i **35**, respektivno. U zavisnosti od toga koji od signala $\mathbf{val_{00}}$, $\mathbf{val_{02}}$, ..., $\mathbf{val_{35}}$ ima vrednost 1 na izlazima kodera se pojavljuje adresa memorijske lokacije sa koje se čita i na linijama $\mathbf{br_{7...0}}$ pojavljuje vrednost koji treba upisati u brojač koraka. Signal višestrukog uslovnog skoka **brop** ima vrednost 1 samo kada signal dekodovanog stanja brojača koraka K_{00} ima vrednost 1, dok u svim ostalim situacijama ima vrednost 0. S obzirom da ovaj signal nema vrednost 1 u stanjima brojača koraka kada treba realizovati bezuslovni ili neki od uslovnih skokova, vrednost na linijama $\mathbf{br_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima brojača koraka $CNT_{7...0}$.

Blok *brojač koraka* sadrži brojač $CNT_{7...0}$. Brojač $CNT_{7...0}$ svojom trenutnom vrednošću određuje koji će upravljački signali da imaju vrednost 1. Brojač $CNT_{7...0}$ može da radi u sledećim režimima: režim inkrementiranja i režim skoka.

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača $CNT_{7...0}$ za jedan čime se obezbeđuje sekvencijalno generisanje upravljačkih signala iz sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 23). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko svi signali **brop**, **brcnd** i **bruncnd** imaju vrednost 0. Signali **brop**, **brcnd** i **bruncnd** normalno imaju vrednost 0 sem u stanjima brojača koraka koja odgovaraju koracima kada treba treba realizovati višestruki uslovni skok, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

U režimu skoka pri pojavi signala takta vrši se upis nove vrednosti u brojač $CNT_{7...0}$ čime se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz sekvence upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 23). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ako jedan od signala **brop**, **brcnd** i **bruncnd** ima vrednost 1. Signali **brop**, **brcnd** i **bruncnd** normalno imaju vrednost 0 sem u stanjima brojača koraka koja odgovaraju koracima kada treba treba realizovati višestruki uslovni skok, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

Brojač koraka $CNT_{7...0}$ je dimenzionisan prema broju koraka u sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 23). S obzirom da se upravljački

signali svih faza izvršavanja instrukcija realizuju u opsegu od koraka K_{00} do koraka K_{35} usvojena je dužina brojača koraka $CNT_{7...0}$ od 8 bitova.

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača $CNT_{7...0}$. Dekodovana stanja brojača $CNT_{7...0}$ pojavljuju se kao signali $\mathbf{K_{00}}$, $\mathbf{K_{01}}$, ..., $\mathbf{K_{FF}}$ na izlazima dekodera DC. Svakom koraku iz sekvence upravljačkih signala po koracima (tabela 17) dodeljeno je po jedno stanje brojača $CNT_{7...0}$ određeno vrednošću signala $\mathbf{K_{00}}$ do $\mathbf{K_{FF}}$ i to koraku $step_{00}$ signal $\mathbf{K_{00}}$, koraku $step_{01}$ signal $\mathbf{K_{01}}$, itd. (tabela 23).

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala $\mathbf{K_{00}}$, $\mathbf{K_{01}}$, ..., $\mathbf{K_{35}}$ koji dolaze sa bloka *dekoder koraka*, signala logičkih uslova $\mathbf{MQ_0}$, **ISCZ**, **BBZ** i $\mathbf{AB_8}$ koji dolaze iz operacione jedinice i saglasno sekvenci upravljačkih signala za upravljačku jedinicu ožičene realizacije (tabela 23) generišu dve grupe upravljačkih signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice se generišu prema izrazima koji su identični sa odgovarajućim izrazima za upravljačku jedinice realizovanu kao šetajuća jedinica (odeljak 1.3.2.1.1). U datim izrazima sada $\mathbf{K_{00}}$, $\mathbf{K_{01}}$, ..., $\mathbf{K_{35}}$ представљају сигнале декодованих стања brojača koraka придружених операционим блоковима у дијаграму тока управљачких сигнала (slike 41 do 47).

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **brop** = K_{00}
- **bruncnd** = $K_{03}+K_{06}+K_{09}+K_{0C}+K_{0F}+K_{12}+K_{15}+\mathbf{K_{18}}+K_{1B}+\mathbf{K_{1E}}+\mathbf{K_{24}}+K_{2D}+\mathbf{K_{33}}+K_{35}$
- **brcnd** = $\mathbf{brnotMQ0}\cdot\overline{\mathbf{MQ_0}}+\mathbf{brnotISCZ}\cdot\overline{\mathbf{ISCZ}}+\mathbf{brBBZ}\cdot\mathbf{BBZ}+\mathbf{brnotAB8}\cdot\overline{\mathbf{AB_8}}+\mathbf{brAB8}\cdot\mathbf{AB_8}$
- **brnotMQ0** = $\mathbf{K_{20}}$
- **brnotISCZ** = $K_{23} + K_{30}$
- **brBBZ** = K_{26}
- **brnotAB8**= $K_{28} + K_{31}$
- **brAB8**= K_{2B}
- $\mathbf{val_{00}} = K_{03}+K_{06}+K_{09}+K_{0C}+K_{0F}+K_{12}+K_{15}+K_{18}+K_{1B}+K_{1E}+K_{24}+K_{35}$
- $\mathbf{val_{20}} = K_{23}$
- $\mathbf{val_{22}} = K_{20}$
- $\mathbf{val_{2B}} = K_{30}$
- $\mathbf{val_{2E}} = K_{2B}$
- $\mathbf{val_{30}} = K_{2D}$
- $\mathbf{val_{33}} = K_{31}$
- $\mathbf{val_{34}} = K_{26}+K_{28}$
- $\mathbf{val_{35}} = \mathbf{T_{33}}$

Pri generisanju signala **brcnd** koriste se signali logičkih uslova $\mathbf{MQ_0}$, **ISCZ**, **BBZ** i $\mathbf{AB_8}$ koji dolaze iz operacione jedinice.

1.3.2.2 MIKROPROGRAMSKO GENERISANJE SIGNALA

U ovom odeljku se daju postupci realizacija upravljačkih jedinica sa mikroprogramskim generisanjem upravljačkih signala koje mogu da budu sa jednim tipom mikroinstrukcije i sa dva tipa mikroinstrukcija. U oba slučaja se daje najpre postupak formiranja mikroprograma a

zatim i strukturna šema upravljačke jedinice. Upravljačka jedinica se realizuje po postupku datim u odeljku 1.2.2.2.

1.3.2.2.1 JEDAN TIP MIKROINSTRUKCIJE

U ovom odeljku se daje konkretan postupak realizacije upravljačke jedinice sa mikroprogramskim generisanjem upravljačkih signala i to sa jednim tipom mikroinstrukcije. Strukturna šema upravljačke jedinice (slike 57) je realizovana na osnovu dijagrama toka upravljačkih signala (slike 41 do 47) i sekvence upravljačkih signala po koracima (tabela 17) po postupku datim u odeljku 1.2.2.2.1.

Upravljačka jedinica generiše dve vrste upravljačkih signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice. Upravljački signali operacione jedinice se koriste u operacionoj jedinici radi izvršavanja mikrooperacija. Upravljački signali upravljačke jedinice se koriste u upravljačkoj jedinici radi inkrementiranja mikroprogramskog brojača ili upisa nove vrednosti u mikroprogramski brojač i radi generisanja vrednosti za upis u mikroprogramski brojač.

Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu sekvence upravljačkih signala po koracima (tabela 17). Mikroprogram se formira tako što se svakom koraku u sekvenci upravljačkih signala po koracima pridružuje binarna reč sa slike 56. 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
-	mxISC	ldISC	decISC	M	S_1	S_0	C_0

8	9	10	11	12	13	14	15
-	mxAB8	ldAB8	srAB8	slAB8	clAB8	mxBB	ldBB

16	17	18	19	20	21	22	23
$mxAB_1$	$mxAB_0$	ldAB	srAB	slAB	clAB	mxMQ	ldMQ

24	25	26	27	28	29	30	31
slMQ	srMQ	ldC	stC	clC	ldZ	stZ	clZ

32	33	34	35	36	37	38	39
OCout	$ldOCWU_{EU}$	ABout	$ldABWU_{EU}$	MQout	$ldMQWU_{EU}$	PSWout	$ldPSWWU_{EU}$

40	41	42	43	44	45	46	47
clOEU	$stOWU_{EU}$	-	-	*cc*			

48	49	50	51	52	53	54	55
ba							

Slika 56 Mikroinstrukcija za jedinicu sa više operacija
horizontalno kodiranje signala i jedan tip mikroinstrukcije

Mikroinstrukcija ima dva dela i to operacini deo i upravljački deo. Operacioni deo čine bitovi 0 do 43, a upravljački deo čine bitovi 44 do 55. Operacioni deo se koristi za generisanje upravljačkih signala operacione jedinice, a upravljački deo se koristi za generisanje upravljačkih signala upravljačke jedinice.

Operacioni deo ima poseban bit za svaki upravljački signal operacione jedinice. Određeni bit operacionog dela mikroinstrukcije treba da ima vrednost 1 ili 0 u zavisnosti od toga da li u

koraku za koji se formira mikroinstrukcija upravljački signal operacione jedinice kome je pridružen dati bit ima vrednost 1 ili 0, respektivno.

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

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

Bezuslovni skokovi se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 17) 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 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
brnotMQ0	3	$\overline{\mathbf{MQ_0}}$
brnotISCZ	4	$\overline{\mathbf{ISCZ}}$
brBBZ	5	**BBZ**
brnotAB8	6	$\overline{\mathbf{AB_8}}$
brAB8	7	$\mathbf{AB_8}$

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 jedan takav korak treba generisati, način njegovog kodiranja bitovima polja *cc* mikroinstrukcije i korak u sekvenci upravljačkih signala po koracima u kojima se pojavljuje iskaz ovog tipa su dati u tabeli 26.

Tabela 26 Signal višestrukog uslovnog skoka

signal višestrukog uslovnog skoka	polje *cc*	korak
brop	2	$step_{00}$

Vrednosti 0 i 8 do F polja *cc* koje nisu dodeljene signalu bezuslovnog skoka, signalu višestrukog uslovnog skoka i signalima uslovnih skokova određuje da treba preći na sledeću mikroinstrukciju.

Bitovi polja *ba* mikroinstrukcije koriste se za specificiranje adrese mikroinstrukcije na koju treba skočiti kod bezuslovnih skokova i uslovnih skokova ukoliko odgovarajući signal uslova ima vrednost 1 u sekvenci upravljačkih signala po koracima (tabela 17). 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 56 bitova. Ukupan broj upravljačkih signala operacione jedinice 41 ali je usvojeno da se za kodiranje operacionog dela mikroinstrukcije koriste 44 bita. Ukupan broj signala bezuslovnih skokova, uslovnih skokova i višestrukih uslovnih skokova je 7 ali je usvojeno da se za kodiranje polja *cc* upravljačkog dela mikroinstrukcije koriste 4 bita. Ukupan broj koraka u sekvenci upravljačkih signala po koracima 54 ali je usvojeno da se za kodiranje polja *ba* upravljačkog dela mikroinstrukcije koristi 8 bitova. Ovo je učinjeno samo zbog toga da bi se omogućio takav način kodiranja ovih polja kojim se dobija pregledniji mikroprogram predstavjen u heksadecimalnom obliku.

Mikroprogram se formira tako što se za svaki korak u sekvenci upravljačkih signala po koracima (tabela 17) 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 mikroinstrukcije koji odgovaraju upravljačkim signalima operacione jedinice koji se javljaju u datom koraku postavljaju se na 1, dok se bitovi ovog dela mikroinstrukcije koji odgovaraju upravljačkim signalima operacione jedinice koji se ne javljaju u datom koraku postavljaju na 0.

Kod formiranja upravljačkog dela mikroinstrukcije za dati korak se proverava da li se javlja neki od iskaza *br* $step_A$, *br* (*if* **uslov** *then* $step_A$) i *br* (*case* (**$uslov_1$**, ..., **$uslov_n$**) *then* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$)). Za korake u kojima se javljaju, bitovi polja *cc* i *ba* se kodiraju u zavisnosti od toga koji se od ova tri iskaza javlja u datom koraku.

Za iskaz *br* $step_A$ se upravljački deo mikroinstrukcije kodira tako što se za polje *cc* uzima kod dodeljen signalu bezuslovnog skoka koji određuje da se bezuslovno skače na korak $step_A$ i za polje *ba* binarna vrednosti A koju treba upisati u mikroprogramski brojač. Simbolička oznaka signala bezuslovnog skoka i način njegovog kodiranja poljem *cc* dati su u tabeli 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
$step_{03}$	$step_{00}$	**$madr_{00}$**	00
$step_{06}$	$step_{00}$	**$madr_{00}$**	00
$step_{09}$	$step_{00}$	**$madr_{00}$**	00
$step_{0C}$	$step_{00}$	**$madr_{00}$**	00
$step_{0F}$	$step_{00}$	**$madr_{00}$**	00
$step_{12}$	$step_{00}$	**$madr_{00}$**	00
$step_{15}$	$step_{00}$	**$madr_{00}$**	00

$step_i$	$step_A$	**$madr_A$**	A
$step_{18}$	$step_{00}$	**$madr_{00}$**	00
$step_{1B}$	$step_{00}$	**$madr_{00}$**	00
$step_{1E}$	$step_{00}$	**$madr_{00}$**	00
$step_{24}$	$step_{00}$	**$madr_{00}$**	00
$step_{2D}$	$step_{30}$	**$madr_{30}$**	30
$step_{33}$	$step_{35}$	**$madr_{35}$**	35
$step_{35}$	$step_{00}$	**$madr_{00}$**	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
$step_{20}$	$\overline{\mathbf{MQ_0}}$	$step_{22}$	$\mathbf{madr_{22}}$	22
$step_{23}$	$\overline{\mathbf{ISCZ}}$	$step_{20}$	$\mathbf{madr_{20}}$	20
$step_{26}$	**BBZ**	$step_{34}$	$\mathbf{madr_{34}}$	34
$step_{28}$	$\overline{\mathbf{AB_8}}$	$step_{34}$	$\mathbf{madr_{34}}$	34
$step_{2B}$	$\mathbf{AB_8}$	$step_{2E}$	$\mathbf{madr_{2E}}$	2E
$step_{30}$	$\overline{\mathbf{ISCZ}}$	$step_{2B}$	$\mathbf{madr_{2B}}$	2B
$step_{31}$	$\overline{\mathbf{AB_8}}$	$step_{33}$	$\mathbf{madr_{33}}$	33

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. Simbolička oznaka signala višestrukog uslovnog skoka, način njegovog kodiranja poljem *cc* i kora u sekvenci upravljačkih signala po koracima u kojima se javlja iskaz ovog tipa dati su u tabeli 24. 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 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 ste_{300}

uslov	$step_A$	A
NOP	$step_{00}$	00
ADD	$step_{01}$	01
SUB	$step_{04}$	04
INC	$step_{07}$	07
DEC	$step_{0A}$	0A
AND	$step_{0D}$	0D
OR	$step_{10}$	10

uslov	$step_A$	A
XOR	$step_{13}$	13
NOT	$step_{16}$	16
ASR	$step_{19}$	19
LSR	$step_{19}$	19
ROR	$step_{19}$	19
RORC	$step_{19}$	19

uslov	$step_A$	A
ASL	$step_{1C}$	1C
LSL	$step_{1C}$	1C
ROL	$step_{1C}$	1C
ROLC	$step_{1C}$	1C
MULU	$step_{1F}$	1F
DIVU	$step_{25}$	25

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

Po opisanom postupku je, na osnovu sekvence upravljačkih signala po koracima (tabela 17) 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 .

Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 57. 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.

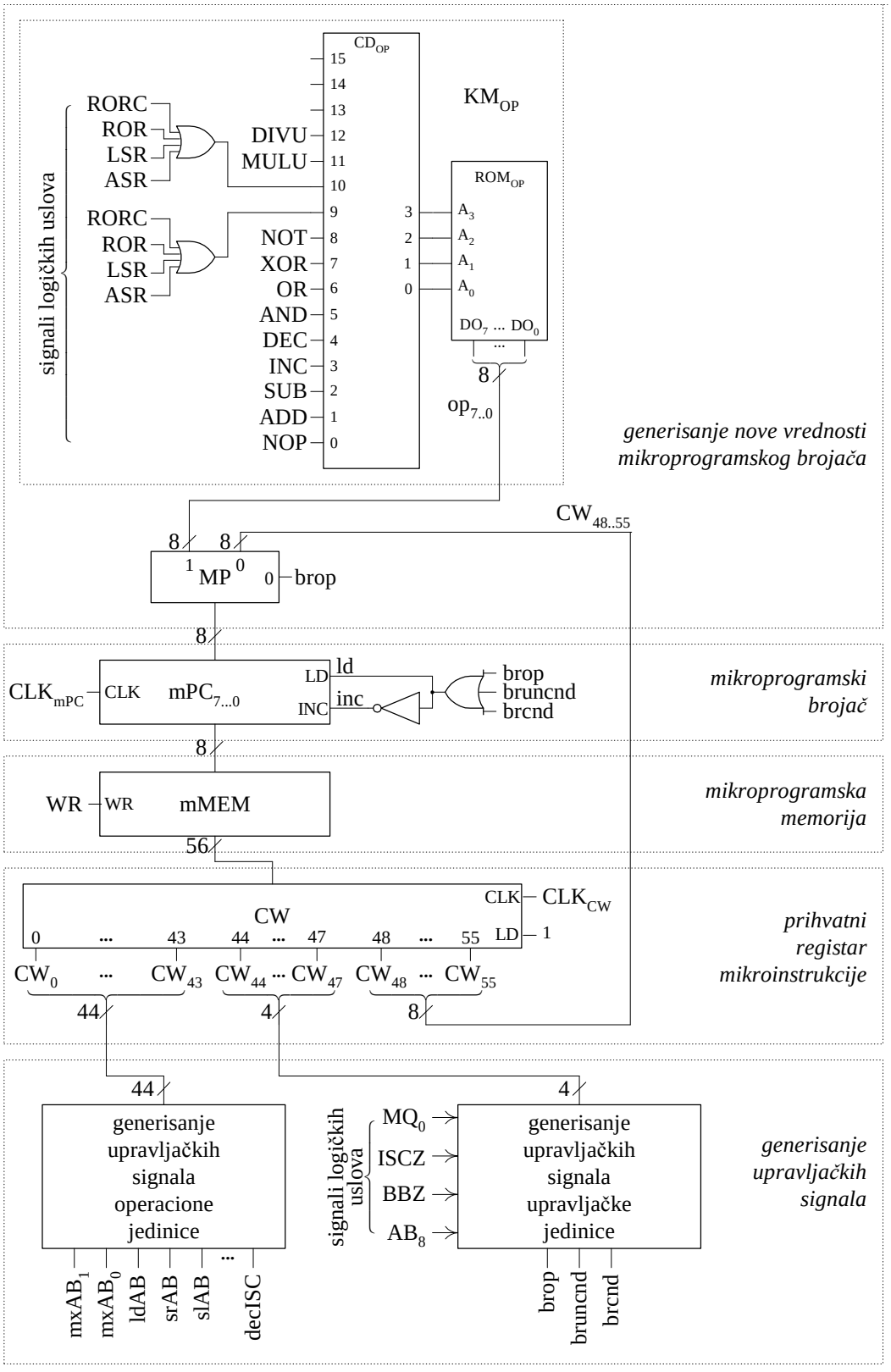
Blok *generisanje nove vrednosti mikroprogramskog brojača* se sastoji od kombinacione mreže KM_{OP} sa multiplekserom MP i služi za generisanje i selekciju vrednosti koju treba upisati u mikroprogramski brojač $mPC_{7...0}$. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikroprograma. Vrednosti koje treba upisati u mikroprogramski brojač generišu se na dva načina i to pomoću kombinacione mreže KM_{OP} koja formira signale $\mathbf{op_{7...0}}$ i razreda $CW_{48...55}$ prihvatnog registra mikroinstrukcije $CW_{0...55}$. Selekcija jedne od dve grupe signala koji daju novu vrednost mikroprogramskog brojača obezbeđuje se signalom **brop** i to signali $\mathbf{op_{7...0}}$ ako signal **brop** ima vrednost 1 i i signali $CW_{48...55}$ ako signal **brop** ima vrednost 0.

Kombinacionom mrežom KM_{OP} generišu se vrednosti (tabela 29) za realizaciju višestrukog uslovnog skoka na adresi 00 mikroprograma (tabela 30). Kombinacionu mrežu KM_{OP} čine koder CD_{OP} i ROM memorija ROM_{OP}. Na ulaze 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 i 12 kodera CD_{OP} vezani su signali **NOP**, **ADD**, **SUB**, INC, DEC, AND, OR, XOR, NOT, (ASR+LSR+ROR+RORC), (ASL+LSL+ROL+ROLC), MULU i **DIVU**, respektivno, a na adresama 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 i 12 memorije ROM_{OP} nalaze se vrednosti 00, 01, 04, 07, **0A**, **0D**, **10**, **13**, **16**, **19**, **1C,** 1F i **25**, respektivno. U zavisnosti od toga koji od signala **NOP**, **ADD**, ..., **DIVU** ima vrednost 1 na izlazima kodera se pojavljuje adresa memorijske lokacije sa koje se čita i na linijama $\mathbf{op_{7...0}}$ pojavljuje vrednost koji treba upisati u mikroprogramski brojač. S obzirom da se na adresi 00 mikroprograma nalazi mikroinstrukcija sa tako kodiranim poljem *cc* da njeno izvršavanje daje vrednost 1 signala višestrukog uslovnog skoka **brop**, vrednost na linijama $\mathbf{op_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača mPC.

Prihvatni registar mikroinstrukcije $CW_{0...55}$ u svojim razredima $CW_{48...55}$ 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 skoka **brop** ima vrednost 1 samo prilikom izvršavanja mikroinstrukcije na adresi 00 mikroprograma, a u svim ostalim situacijama imaju 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 $CW_{48...55}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača $mPC_{7...0}$.

Tabela 30 Mikroprogram za upravljačku jedinicu sa mikroprogramskim generisanjem upravljačkih signala i jednim tipom mikroinstrukcije

00	00000000000	2	00	**!brop!**
! ADD !				
01	0001C000000	0	00	**! mxAB$_1$, ldAB, ldBB!**
02	04002020000	0	00	**!** S$_1$, ldAB, ldC**!**
03	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! SUB !				
04	0001C000000	0	00	**! mxAB$_1$, ldAB, ldBB!**
05	03002020000	0	00	**!** S$_0$, C$_0$, ldAB, ldC**!**
06	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! INC !				
07	0000C000000	0	00	**! mxAB$_1$, ldAB!**
08	01002020000	0	00	**!** C$_0$, ldAB, ldC**!**
09	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! DEC !				
0A	0000C000000	0	00	**! mxAB$_1$, ldAB!**
0B	06002020000	0	00	**!** S$_1$, S$_0$, ldAB, ldC**!**
0C	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! AND !				
0D	0001C000000	0	00	**! mxAB$_1$, ldAB, ldBB!**
0E	08002008000	0	00	**!** M, ldAB, clC**!**
0F	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! OR !				
10	0001C000000	0	00	**! mxAB$_1$, ldAB, ldBB!**
11	0C002008000	0	00	**!** M, S$_1$, ldAB, clC**!**
12	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! XOR !				
13	0001C000000	0	00	**! mxAB$_1$, ldAB, ldBB!**
14	0A002008000	0	00	**!** M, S$_0$, ldAB, clC**!**
15	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! NOT !				
16	0000C000000	0	00	**! mxAB$_1$, ldAB!**
17	0E002008000	0	00	**!** M, S$_1$, S$_0$, ldAB, clC**!**
18	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! ASR, LSR, ROR, RORC !				
19	0000C000000	0	00	**! mxAB$_1$, ldAB!**
1A	00001020000	0	00	**! sr**AB, ldC**!**
1B	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! ASL, LSL, ROL, ROLC !				
1C	0000C000000	0	00	**! mxAB$_1$, ldAB!**
1D	00000820000	0	00	**! sl**AB, ldC**!**
1E	00000004400C	1	00	! **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, **madr$_{00}$!**
! MULU !				
1F	20070500000	0	00	**! mxBB, ldBB, ldMQ, clAB8, clAB, ldISC**!
20	00000000000	3	22	! **brnotMQ0**, madr$_{22}$!
21	04202000000	0	00	**! S$_1$**, ldAB, ldAB8!
22	10101040000	0	00	**! srAB8**, srAB, srMQ, decISC !
23	00000000000	4	20	! **brnotISCZ**, madr$_{20}$!
24	0000000C00C	1	00	**!** clC, **ldZ**, clOEU, stOWU$_{FIL}$ **, bruncnd**, **madr$_{00}$**!
! DIVU !				
25	60056300000	0	00	! **clAB8, mxAB$_0$, ldAB, mxMQ**, **ldMQ, ldBB, mxISC, ldISC**!
26	00000000000	5	34	**! brBBZ**, madr$_{34}$!
27	03202000000	0	00	**! S$_0$, C$_0$**, ldAB, ldAB8!
28	00000000000	6	34	**! brnotAB8**, **madr$_{34}$** !
29	00080880000	0	00	**! slAB8**, slAB, slMQ!
2A	04202000000	0	00	**! S$_1$**, ldAB, ldAB8 !
2B	10000000000	7	2E	**!** decISC, brAB8, madr$_{2E}$!
2C	00080880000	0	00	! **slAB8**, slAB, slMQ !
2D	03202000000	1	30	**! S$_0$, C$_0$**, ldAB, ldAB8, **bruncnd**, madr$_{30}$!
2E	00080880000	0	00	**! slAB8**, slAB, slMQ!
2F	04202000000	0	60	**! S$_1$**, ldAB, ldAB8 !
30	00000000000	4	2B	**!** brnotISCZ, madr$_{2B}$!
31	00000080000	6	33	**! slMQ**, **brnotAB8**, madr$_{33}$!
32	04202000000	0	00	! **S$_1$**, ldAB, ldAB8 !
33	0000000C00C	1	35	**!** clC, **ldZ**, clOEU, stOWU$_{FIL}$, **bruncnd**, madr$_{35}$;
34	00000014400C	0	00	! stC, **ldZ**, clOEU, stOWU$_{FIL}$!
35	00000000000	1	00	**! bruncnd**, madr$_{00}$!



Slika 57 Upravljačke jedinica sa mikroprogramskim generisanjem upravljačkih signala i jednim tipom mikroinstrukcije

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

U režimu skoka pri pojavi signala takta $\mathbf{CLK_{mPC}}$ vrši se upis nove vrednosti u mikroprogramski brojač $mPC_{7...0}$ čime se obezbeđuje odstupanje od sekvencijalnog očitavanja mikroinstrukcija iz mikroprogramske memorije (tabela 30). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ukoliko jedan od signala **brop**, **bruncnd** i **brcnd** ima vrednost 1. Signali **brop**, **bruncnd** i **brcnd** normalno imaju vrednost 0 sem prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran višestruki uslovni skok, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

Mikroprogramski brojač $mPC_{7...0}$ se dimenzioniše prema veličini mikroprograma (tabela 30). S obzirom da se mikroprogram nalazi u opsegu adresa od 00 do 35, dovoljna bi bila dužina mikroprogramskog brojača $mPC_{7...0}$ od 8 bitova. Međutim, da bi mikroprogram predstavljen u heksadecimalnom obliku bio pregledniji usvojeno je da adrese budu dužine 8 bitova, pa je usvojena dužina mikroprogramskog brojača $mPC_{7...0}$ od 8 bitova.

Blok *mikroprogramska memorija* sadrži mikroprogramsku memoriju mMEM u kojoj je smešten mikroprogram. Širina reči mikroprogramske memorije je određena usvojenom dužinom mikroinstrukcije i iznosi 56 bitova, 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 $CW_{0...55}$. Prihvatni registar mikroinstrukcije $CW_{0...55}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...43}$ i $CW_{44...47}$ se koriste u bloku *generisanje upravljačkih signala* za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{48...55}$ 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 $CW_{0...43}$ prihvatnog registra mikroinstrukcije generišu upravljačke signale operacione jedinice i na osnovu sadržaja razreda $CW_{44...47}$ prihvatnog registra mikroinstrukcije i signala

logičkih uslova **MQ_0**, **ISCZ**, **BBZ** i **AB_8** koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice se generišu prema izrazima
$mxAB_1 = CW_{16}$
$mxAB_0 = CW_{17}$
$ldAB = CW_{18}$
$M = CW_4$
$S_1 = CW_5$
$S_0 = CW_6$
$srAB = CW_{19}$
$slAB = CW_{20}$
$decISC = CW_3$ itd.
у којима CW_{16}, CW_{17}, ..., CW_3 представљају сигнале са излаза razreda operacionog dela prihvatnog registra mikroinstrukcije saglasno formatu mikroinstrukcije (слика 56).

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **brop** = $\overline{CW_{44}} \cdot \overline{CW_{45}} \cdot CW_{46} \cdot \overline{CW_{47}}$
- **bruncnd** = $\overline{CW_{44}} \cdot \overline{CW_{45}} \cdot \overline{CW_{46}} \cdot CW_{47}$
- **brcnd** = **brnotMQ0**$\cdot \overline{\mathbf{MQ_0}}$ +**brnotISCZ**$\cdot \overline{\mathbf{ISCZ}}$ +**brBBZ**$\cdot$**BBZ**+
  brnotAB8$\cdot \overline{\mathbf{AB_8}}$ +**brAB8**$\cdot \mathbf{AB_8}$
- **brnotMQ0** = $\overline{CW_{44}} \cdot \overline{CW_{45}} \cdot CW_{46} \cdot CW_{47}$
- **brnotISCZ** = $\overline{CW_{44}} \cdot CW_{45} \cdot \overline{CW_{46}} \cdot \overline{CW_{47}}$
- **brBBZ** = $\overline{CW_{44}} \cdot CW_{45} \cdot \overline{CW_{46}} \cdot CW_{47}$
- **brnotAB8**= $\overline{CW_{44}} \cdot CW_{45} \cdot CW_{46} \cdot \overline{CW_{47}}$
- **brAB8**= $\overline{CW_{44}} \cdot CW_{45} \cdot CW_{46} \cdot CW_{47}$

у којима CW_{44} do CW_{47} представљају сигнале са излаза razreda upravljačkog dela prihvatnog registra mikroinstrukcije kojima se saglasno formatu mikroinstrukcije kodira polje *cc* (слика 56), a **MQ_0**, **ISCZ**, **BBZ** i **AB_8** signale logičkih uslova koji dolaze iz operacione jedinice.

1.3.2.2.2 DVA TIPA MIKROINSTRUKCIJA

U ovom odeljku se daje konkretan postupak realizacije upravljačke jedinice sa mikroprogramskim generisanjem upravljačkih signala i to sa dva tipa mikroinstrukcija.

Upravljačka jedinica generiše dve vrste upravljačkih signala i to upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice. Upravljački signali operacione jedinice se koriste u operacionoj jedinici radi izvršavanja mikrooperacija. Upravljački signali upravljačke jedinice se koriste u upravljačkoj jedinici radi inkrementiranja mikroprogramskog brojača ili upisa nove vrednosti u mikroprogramski brojač i radi generisanja vrednosti za upis u mikroprogramski brojač.

Za generisanje upravljačkih signala koriste se dva tipa mikroinstrukcija i to operaciona mikroinstrukcija za generisanje upravljačkih signala operacione jedinice i upravljačka mikroinstrukcija za generisnje upravljačkih signala upravljačke jedinice. Pri tome struktura operacione mikroinstrukcije odgovara strukturi opracionog dela mikroinstrukcije upravljačkih jedinica sa jednim tipom mikroinstrukcije, dok struktura upravljačke mikroinstrukcije odgovara strukturi upravljačkog dela mikroinstrukcije upravljačkih jedinica sa jednim tipom mikroinstrukcije. Da bi mogao da se formira mikroprogram u kome se pojavljuju posebno

operacione i posebno upravljačke mikroinstrukcije potrebno je da se modifikuje sekvenca upravljačkih signala po koracima (tabela 17). Svaki korak u kome se samo generišu upravljački signali operacione jedinice i ne realizuje skok, ostaje i u modifikovanoj sekvenci upravljačkih signala i u mikroprogramu se predstavlja jednom operacionom mikroinstrukcijom. Svaki korak u kome se ne generišu upravljački signali operacione jedinice već samo realizuje skok, ostaje i u modifikovanoj sekvenci upravljačkih signala i u mikroprogramu se predstavlja jednom upravljačkom mikroinstrukcijom. Svaki korak u kome se ne samo generišu upravljački signali operacione jedinice već realizuje i skok, u modifikovanoj sekvenci upravljačkih signala se zamenjuje sa dva posebna koraka i to jednim korakom za generisanje upravljačkih signala operacione jedinice i jednim korakom za realizaciju skoka i u mikroprogramu se predstavlja sa dve mikroinstrukcije i to jednom operacionom mikroinstrukcijom i jednom upravljačkom mikroinstrukcijom. Zbog toga je potrebno od originalne sekvence upravljačkih signala po koracima (tabela 17) formirati modifikovanu sekvencu upravljačkih signala po koracima (tabela 31).

Tabela 31 Modifikovana sekvenca upravljačkih signala po koracima

step$_{00}$ *br* (*case* (**ADD**, **SUB**, **INC**, **DEC**, **AND**,**OR**, **XOR**, **NOT**, **ASR**, **LSR**, **ROR**, **RORC**, **ASL**, **LSL**, **ROL**, **ROLC**, **MULU**, **DIVU, NOP**) *then* (**ADD**, step$_{01}$), (**SUB**, ste$_{05}$), (**INC**, step$_{09}$), (**DEC**, step$_{0D}$), (**AND**, step$_{11}$), (**OR**, step$_{15}$), (**XOR**, step$_{19}$), (**NOT**, step$_{1D}$), (**ASR**, step$_{21}$), (**LSR**, step$_{21}$), (**ROR**, step$_{21}$), (**RORC**, step$_{21}$), (**ASL**, step$_{25}$), (**LSL**, step$_{25}$), (**ROL**, step$_{25}$), (**ROLC**, step$_{25}$), (**MULU**, step$_{29}$), (**DIVU**, step$_{30}$), (**NOP**, step$_{00}$));

! ADD !

step$_{01}$ **mxAB$_1$**, **ldAB**, **ldBB**;
step$_{02}$ S$_1$, ldAB, ldC;
step$_{03}$ **ldZ**, clOEU, stOWU$_{EU}$,
step$_{04}$ *br* step$_{00}$;

! SUB !

step$_{05}$ **mxAB$_1$**, **ldAB**, **ldBB**;
step$_{06}$ S$_0$, C$_0$, ldAB, ldC;
step$_{07}$ **ldZ**, clOEU, stOWU$_{EU}$;
step$_{08}$ *br* step$_{00}$;

! INC !

step$_{09}$ **mxAB$_1$**, **ldAB**, **ldBB**;
step$_{0A}$ C$_0$, ldAB, ldC;
step$_{0B}$ **ldZ**, clOEU, stOWU$_{EU}$;
step$_{0C}$ *br* step$_{00}$;

! DEC !

step$_{0D}$ **mxAB$_1$**, **ldAB**, **ldBB**;
step$_{0E}$ S$_1$, S$_0$, ldAB, ldC;
step$_{0F}$ **ldZ**, clOEU, stOWU$_{EU}$;
step$_{10}$ *br* step$_{00}$;

! AND !

step$_{11}$ **mxAB$_1$**, **ldAB**, **ldBB**;
step$_{12}$ M, ldAB, clC;
step$_{13}$ **ldZ**, clOEU, stOWU$_{EU}$;
step$_{14}$ *br* step$_{00}$;

! OR !

step$_{15}$ **mxAB$_1$**, **ldAB**, **ldBB**;
step$_{16}$ M, **S$_1$**, ldAB, clC;
step$_{17}$ **ldZ**, clOEU, stOWU$_{EU}$;
step$_{18}$ *br* step$_{00}$;

! XOR !

step$_{19}$ **mxAB$_1$**, **ldAB**, **ldBB**;
step$_{1A}$ M, **S$_0$**, ldAB, clC;
step$_{1B}$ **ldZ**, clOEU, stOWU$_{EU}$;
step$_{1C}$ *br* step$_{00}$;

! NOT !

step$_{1D}$ **mxAB$_1$**, **ldAB**, **ldBB**;
step$_{1E}$ M, **S$_1$**, S$_0$, ldAB, clC;
step$_{1F}$ **ldZ**, clOEU, stOWU$_{EU}$;
step$_{20}$ *br* step$_{00}$;

! ASR, LSR, ROR, RORC !

step$_{21}$ **mxAB$_1$**, **ldAB**;

$step_{22}$ srAB, ldC;
$step_{23}$ **ldZ**, clOEU, $stOWU_{EU}$;
$step_{24}$ *br* $step_{00}$;
! ASL, LSL, ROL, ROLC !
$step_{25}$ **$mxAB_1$**, **ldAB**;
$step_{26}$ slAB, ldC;
$step_{27}$ **ldZ**, clOEU, $stOWU_{EU}$;
$step_{28}$ *br* $step_{00}$;
! MULU !
$step_{29}$ **mxBB**, **ldBB, ldMQ, clAB8, clAB, ldISC**;
$step_{2A}$ *br* (*if* $\overline{\mathbf{MQ_0}}$ *then* $step_{2C}$);
$step_{2B}$ **S_1**, ldAB, ldAB8;
$step_{2C}$ **srAB8**, srAB, srMQ, decISC;
$step_{2D}$ *br* (*if* $\overline{\mathbf{ISCZ}}$ *then* $step_{2A}$);
$step_{2E}$ clC, **ldZ**, clOEU, $stOWU_{EU}$;
$step_{2F}$ *br* $step_{00}$;
! DIVU !
$step_{30}$ **clAB8, $mxAB_0$, ldAB, mxMQ**, **ldMQ, ldBB, mxISC, ldISC**;
$step_{31}$ *br* (*if* **BBZ** *then* $step_{43}$);
$step_{32}$ **S_0**, C_0, ldAB, ldAB8;
$step_{33}$ *br* (*if* $\overline{\mathbf{AB_8}}$ *then* $step_{43}$);
$step_{34}$ **slAB8**, slAB, slMQ;
$step_{35}$ **S_1**, ldAB, ldAB8;
$step_{36}$ decISC;
$step_{37}$ *br* (*if* AB_8 *then* $step_{3B}$);
$step_{38}$ **slAB8**, slAB, slMQ;
$step_{39}$ **S_0**, C_0, ldAB, ldAB8;
$step_{3A}$ *br* $step_{3D}$;
$step_{3B}$ **slAB8**, slAB, slMQ;
$step_{3C}$ **S_1**, ldAB, ldAB8;
$step_{3D}$ *br* (*if* $\overline{\mathbf{ISCZ}}$ *then* $step_{36}$);
$step_{3E}$ **slMQ**;
$step_{3F}$ *br* (*if* $\overline{\mathbf{AB_8}}$ *then* $step_{41}$);
$step_{40}$ **S_1**, ldAB, ldAB8,
$step_{41}$ clC, **ldZ**, clOEU, $stOWU_{EU}$;
$step_{42}$ *br* $step_{44}$;
$step_{43}$ stC, **ldZ**, clOEU, $stOWU_{EU}$;
$step_{44}$ *br* $step_{00}$;

Strukturna šema upravljačke jedinice (slike 60) je realizovana na osnovu modifikovane sekvence upravljačkih signala po koracima (tabela 31) po postupku datom u odeljku 1.2.2.2.2. Upravljački signali operacione i upravljačke jedinice se generišu korišćenjem mikroprograma koji se formira na osnovu modifikovane sekvence upravljačkih signala po koracima. Mikroprogram se formira tako što se u sekvenci upravljačkih signala po koracima svakom koraku u kome su navedeni upravljački signali operacione jedinice koji treba da imaju vrednost 1 pridružuje binarna reč čiji je format dat na slici 58 i svakom koraku u kome su realizuju skokovi pridružuje binarna reč čiji je format dat na slici 59. Te binarne reči se nazivaju mikroinstrukcijama, mikronaredbama ili mikrokomandama. Mikroinstrukcije pridružene koracima u kojima su navedeni upravljački signali operacione jedinice koji treba da imaju vrednost 1 nazivaju se operacione mikroinstrukcije, dok se mikroinstrukcije pridružene koracima u kojima su realizuju skokovi nazivaju upravljačke mikroinstrukcije.

Uređeni niz ovako formiranih mikroinstrukcija naziva se mikroprogram.

0	1	2	3	4	5	6	7
0	mxISC	ldISC	decISC	M	S_1	S_0	C_0

8	9	10	11	12	13	14	15
-	mxAB8	ldAB8	srAB8	slAB8	clAB8	mxBB	ldBB

16	17	18	19	20	21	22	23
$mxAB_1$	$mxAB_0$	ldAB	srAB	slAB	clAB	mxMQ	ldMQ

24	25	26	27	28	29	30	31
slMQ	srMQ	ldC	stC	clC	ldZ	stZ	clZ

32	33	34	35	36	37	38	39
OCout	$ldOCWU_{EU}$	ABout	$ldABWU_{EU}$	MQout	$ldMQWU_{EU}$	PSWout	$ldPSWWU_{EU}$

40	41	42	43				
clOEU	$stOWU_{EU}$	-	-				

Slika 58 Operaciona mikroinstrukcija za jedinicu sa više operacija

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

8	9	10	11	12	13	14	15
ba							

16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0

24	25	26	27	28	29	30	31
0	0	0	0	0	0	0	0

32	33	34	35	36	37	38	39
0	0	0	0	0	0	0	0

40	41	42	43				
0	0	0	0				

Slika 59 Upravljačka mikroinstrukcija za jedinicu sa više operacija

Bit 0 svake mikroinstrukcije vrednostima 0 i 1 određuje da li se radi o operacionoj ili upravljačkoj mikroinstrukciji, respektivno.

Bitovi 1 do 41 operacione mikroinstrukcije dodeljeni su upravljačkim signalima operacione jedinice i to bit 1 signalu mxISC, bit 2 signalu ldISC i tako redom do bita 41 koji je dodeljen signalu $stOWU_{EU}$. Ovi bitovi operacione mikroinstrukcije se dodeljuju upravljačkim signalima operacione jedinice na proizvoljan način. Bitovi 8, 42 i 43 se ne koristi, a uključeni su da bi, time što je broj bitova mikroinstrukcije deljiv sa četiri, mikroprogram napisan u heksadecimalnom sistemu bio pregledniji.

Bitovi 4 do 7 i 8 do 15 upravljačke mikroinstrukcije se koriste za kodiranje polja *cc* i *ba*, respektivno. I kod upravljačke mikroinstrukcije broj bitova dodeljen pojedinim poljima mikroinstrukcije je deljiv sa četiri da bi mikroprogram napisan u heksadecimalnom sistemu bio pregledniji. Operaciona mikroinstrukcija je duža od upravljačke mikroinstrukcije, pa je dužina mikroinstrukcija, a time i širina reči mikroprogramske memorije, određena dužinom

operacione mikroinstrukcije i iznosi 44 bita. Bitovi 16 do 43 upravljačke mikroinstrukcije se ne koriste.

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

Bezuslovni skokovi se realizuje u onim koracima sekvence upravljačkih signala po koracima (tabela 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 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
brnotMQ0	3	$\overline{\mathbf{MQ_0}}$
brnotISCZ	4	$\overline{\mathbf{ISCZ}}$
brBBZ	5	**BBZ**
brnotAB8	6	$\overline{\mathbf{AB_8}}$
brAB8	7	$\mathbf{AB_8}$

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 jedan takav korak treba generisati, način njegovog kodiranja bitovima polja *cc* mikroinstrukcije i korak u sekvenci upravljačkih signala po koracima u kojima se pojavljuje iskaz ovog tipa su dati u tabeli 34.

Tabela 34 Signal višestrukog uslovnog skoka

signal višestrukog uslovnog skoka	polje *cc*	korak
brop	2	$step_{00}$

Vrednosti 0 i 8 do F polja *cc* koje nisu dodeljene signalu bezuslovnog skoka, signalu višestrukog uslovnog skoka i signalima uslovnih skokova određuje da treba preći na sledeću mikroinstrukciju.

Bitovi polja *ba* mikroinstrukcije koriste se za specificiranje adrese mikroinstrukcije na koju treba skočiti kod bezuslovnih skokova i uslovnih skokova ukoliko odgovarajući signal uslova ima vrednost 1 u modifikovanoj 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 56 bitova. Za kodiranje operacionog dela mikroinstrukcije koristi se 44 bita. Upravljačkih signala operacione jedinice ima 41, ali je umesto 41 bita usvojena dužina operacionog dela mikroinstrukcije 44 bita, da bi se omogućio takav način pridruživanja bitova upravljačkim signalima operacione jedinice kojim se dobija pregledniji mikroprogram predstavjen u heksadecimalnom obliku. Za kodiranje polja *cc* upravljačkog dela mikroinstrukcije usvojena su 4 bita, jer je ukupan broj signala bezuslovnih skokova, uslovnih skokova i višestrukih uslovnih skokova 7. Za kodiranje polja *ba* upravljačkog dela mikroinstrukcije usvojeno je 8 bitova, jer je ukupan broj koraka u sekvenci upravljačkih signala po koracima 54.

Mikroprogram se formira tako što se za svaki korak u modifikovanoj sekvenci upravljačkih signala po koracima (tabela 31) formira jedna mikroinstrukcija. Operacione mikroinstrukcije se formiraju ukoliko u datom koraku ima upravljačkih signala operacione jedinice. Upravljačka mikroinstrukcija se formira ukoliko u datom koraku ima iskaza za bezuslovni skok, uslovni skok ili višestruki uslovni skok.

Kod formiranja operacione mikroinstrukcije bitovi 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 upravljačke mikroinstrukcije za dati korak se proverava da li se javlja neki od iskaza *br* $step_A$, *br* (*if* **uslov** *then* $step_A$) i *br* (*case* (**$uslov_1$**, ..., **$uslov_n$**) *then* (**$uslov_1$**, $step_{A1}$), ..., (**$uslov_n$**, $step_{An}$)). Za korake u kojima se javljaju, bitovi polja *cc* i *ba* se kodiraju u zavisnosti od toga koji se od ova tri iskaza javlja u datom koraku.

Za iskaz *br* $step_A$ se upravljačka mikroinstrukcija 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 35.

Tabela 35 Koraci $step_i$, $step_A$, vrednosti $\mathbf{madr_A}$ i vrednosti A za bezuslovne skokove

$step_i$	$step_A$	$\mathbf{madr_A}$	A
$step_{04}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{08}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{0C}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{10}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{14}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{18}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{1C}$	$step_{00}$	$\mathbf{madr_{00}}$	00

$step_i$	$step_A$	$\mathbf{madr_A}$	A
$step_{20}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{24}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{28}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{2F}$	$step_{00}$	$\mathbf{madr_{00}}$	00
$step_{3A}$	$step_{3D}$	$\mathbf{madr_{3D}}$	3D
$step_{42}$	$step_{44}$	$\mathbf{madr_{44}}$	44
$step_{44}$	$step_{00}$	$\mathbf{madr_{00}}$	00

Za iskaz *br* (*if* **uslov** *then* $step_A$) se upravljačka mikroinstrukcija kodira tako što se za polje *cc* uzima kod dodeljen signalu uslovnog skoka koji određuje signal **uslov** koji treba da 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 modifikovanoj 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 36.

Tabela 36 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
$step_{2A}$	$\overline{\mathbf{MQ_0}}$	$step_{2C}$	$\mathbf{madr_{2C}}$	2C
$step_{2D}$	$\overline{\mathbf{ISCZ}}$	$step_{2A}$	$\mathbf{madr_{20}}$	2A
$step_{31}$	**BBZ**	$step_{43}$	$\mathbf{madr_{43}}$	43
$step_{33}$	$\overline{\mathbf{AB_8}}$	$step_{33}$	$\mathbf{madr_{43}}$	43
$step_{37}$	$\mathbf{AB_8}$	$step_{3B}$	$\mathbf{madr_{3B}}$	3B
$step_{3D}$	$\overline{\mathbf{ISCZ}}$	$step_{36}$	$\mathbf{madr_{36}}$	36
$step_{3F}$	$\overline{\mathbf{AB_8}}$	$step_{41}$	$\mathbf{madr_{41}}$	41

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. Simbolička oznaka signala višestrukog uslovnog skoka, način njegovog kodiranja poljem *cc* i korak u sekvenci upravljačkih signala po koracima u kojima se javlja iskaz ovog tipa dati su u tabeli 34. 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 dat je u tabeli 37.

Tabela 37 Signali uslova, koraci na koje se skače i vrednosti za upis u mikroprogramski brojač za višestruki uslovni skok u koraku $step_{00}$

uslov	$step_A$	A
NOP	$step_{00}$	00
ADD	$step_{01}$	01
SUB	$step_{05}$	05
INC	$step_{09}$	09
DEC	$step_{0D}$	0D
AND	$step_{11}$	11
OR	$step_{15}$	15

uslov	$step_A$	A
XOR	$step_{19}$	19
NOT	$step_{1D}$	1D
ASR	$step_{21}$	21
LSR	$step_{21}$	21
ROR	$step_{21}$	21
RORC	$step_{21}$	21

uslov	$step_A$	A
ASL	$step_{25}$	25
LSL	$step_{25}$	25
ROL	$step_{25}$	25
ROLC	$step_{25}$	25
MULU	$step_{29}$	29
DIVU	$step_{30}$	30

Iz izloženog se vidi da su upravljački signali za upravljačku jedinicu mikroprogramske realizacije signal bezuslovnog skoka (tabela 32), signali uslovnih skokova (tabela 36), signal višestrukog uslovnog skoka (34) i signali vrednosti A za bezuslovne skokove (tabela 35), uslovne skokove (tabela 36) i višestruki uslovn skok (tabela 37).

Po opisanom postupku je, na osnovu modifikovane sekvence upravljačkih signala po koracima (tabela 31) formiran mikroprogram (tabela 38). 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 .

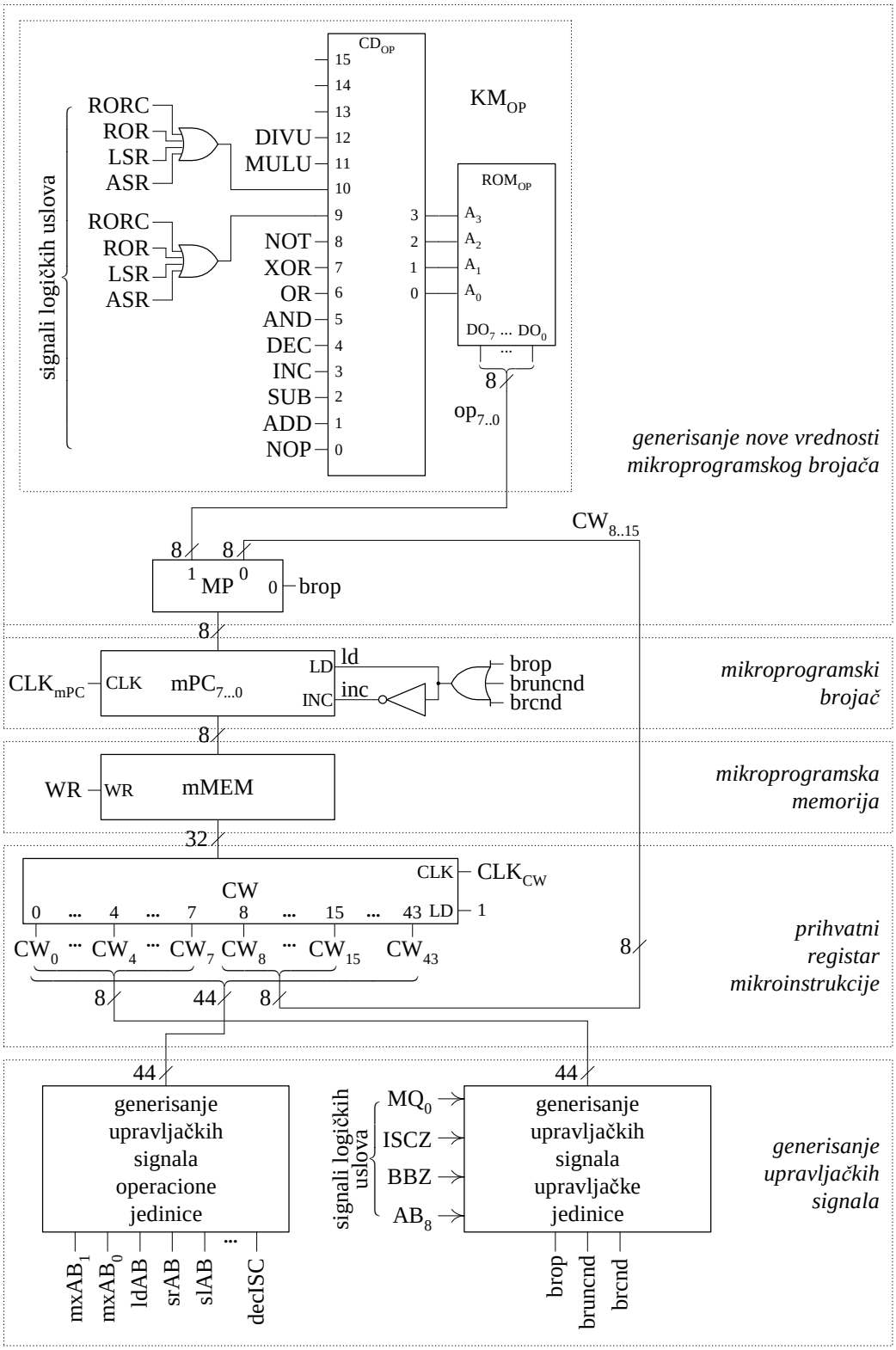
Struktura upravljačke jedinice mikroprogramske realizacije je prikazana na slici 60. Upravljačka jedinica se sastoji iz sledećih blokova: blok *generisanje nove vrednosti mikroprogramskog brojača*, blok *mikroprogramski brojač*, blok *mikroprogramska memorija*, blok *prihvatni registar mikroinstrukcije* 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 KM_{OP} 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 KM_{OP} koja formira signale $\mathbf{op_{7...0}}$ i razreda $CW_{48...55}$ prihvatnog registra mikroinstrukcije $CW_{0...55}$. Selekcija jedne od dve grupe signala koji daju novu vrednost mikroprogramskog brojača obezbeđuje se signalom **brop** i to signali $\mathbf{op_{7...0}}$ ako signal **brop** ima vrednost 1 i i signali $CW_{0...55}$ ako signal **brop** ima vrednost 0.

Tabela 38 Mikroprogram za upravljačku jedinicu sa mikroprogramskim generisanjem upravljačkih signala i dva tipa mikroinstrukcija

00 820000000000	**!brop!**
! ADD !	
01 0001C0000000	**! mxAB$_1$, ldAB, ldBB !**
02 040020200000	**!** S$_1$, ldAB, ldC **!**
03 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
04 810000000000	! **bruncnd**, **madr$_{00}$!**
! SUB !	
05 0001C0000000	**! mxAB$_1$, ldAB, ldBB !**
06 030020200000	**!** S$_0$, C$_0$, ldAB, ldC **!**
07 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
08 810000000000	! **bruncnd**, **madr$_{00}$!**
! INC !	
09 0000C0000000	**! mxAB$_1$, ldAB !**
0A 010020200000	**!** C$_0$, ldAB, ldC **!**
0B 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
0C 810000000000	! **bruncnd**, **madr$_{00}$!**
! DEC !	
0D 0000C0000000	**! mxAB$_1$, ldAB !**
0E 060020200000	**!** S$_1$, S$_0$, ldAB, ldC **!**
0F 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
10 810000000000	! **bruncnd**, **madr$_{00}$!**
! AND !	
11 0001C0000000	**! mxAB$_1$, ldAB, ldBB!**
12 080020080000	**!** M, ldAB, clC**!**
13 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
14 810000000000	! **bruncnd**, **madr$_{00}$!**
! OR !	
15 0001C0000000	**! mxAB$_1$, ldAB, ldBB!**
16 0C0020080000	**!** M, S$_1$, ldAB, clC**!**
17 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
18 810000000000	! **bruncnd**, **madr$_{00}$!**
! XOR !	
19 0001C0000000	**! mxAB$_1$, ldAB, ldBB!**
1A 0A0020080000	**!** M, S$_0$, ldAB, clC**!**
1B 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
1C 810000000000	! **bruncnd**, **madr$_{00}$!**
! NOT !	
1D 0000C0000000	**! mxAB$_1$, ldAB!**
1E 0E0020080000	**!** M, S$_1$, S$_0$, ldAB, clC**!**
1F 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
20 810000000000	! **bruncnd**, **madr$_{00}$!**
! ASR, LSR, ROR, RORC !	
21 0000C0000000	**! mxAB$_1$, ldAB!**
22 000010200000	**! sr**AB, ldC**!**
23 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
24 810000000000	! **bruncnd**, **madr$_{00}$!**
! ASL, LSL, ROL, ROLC !	
25 0000C0000000	**! mxAB$_1$, ldAB !**
26 000008200000	**! sl**AB, ldC **!**
27 000000400C	! **ldZ**, clOEU, stOWU$_{FIT}$ **!**
28 810000000000	! **bruncnd**, **madr$_{00}$!**
! MULU !	
29 200705000000	**! mxBB, ldBB, ldMQ, clAB8, clAB, ldISC**!
2A 832200000000	! **brnotMO0**, madr$_{22}$!
2B 042020000000	**! S$_1$**, ldAB, ldAB8!
2C 101010400000	**! srAB8**, srAB, srMQ, decISC !
2D 842000000000	! **brnotISCZ**, madr$_{20}$!
2E 0000000C00C	**!** clC, **ldZ**, clOEU, stOWU$_{FIT}$!
2F 810000000000	! **bruncnd**, **madr$_{00}$!**
! DIVU !	
30 600563000000	! **clAB8, mxAB$_0$, ldAB, mxMQ, ldMQ, ldBB, mxISC, ldISC**!
31 854300000000	**! brBBZ**, madr$_{43}$!
32 032020000000	**! S$_0$, C$_0$**, ldAB, ldAB8!
33 864300000000	**! brnotAB8**, **madr$_{43}$** !
34 000808800000	**! slAB8**, slAB, slMQ!
35 042020000000	**! S$_1$**, ldAB, ldAB8 !

```
36  100000000000        ! decISC!
37  873B00000000        ! brAB8, madr3B !
38  000808800000        ! slAB8, slAB, slMQ !
39  032020000000        ! S0, C0, ldAB, ldAB8!
3A  813D00000000        ! bruncnd, madr3D !
3B  000808800000        ! slAB8, slAB, slMQ!
3C  042020000000        ! S1, ldAB, ldAB8 !
3D  843600000000        ! brnotISCZ, madr36 !
3E  000000800000        ! slMQ, brnotAB8 !
3F  864100000000        ! brnotAB8, madr41!
40  042020000000        ! S1, ldAB, ldAB8 !
41  00000000C00C        ! clC, ldZ, clOEU, stOWUFFF ;
42  814400000000        ! bruncnd, madr44 ;
43  000000140 0C        ! stC, ldZ, clOEU, stOWUFFF!
44  810000000000        ! bruncnd, madr00!
```



Slika 60 Upravljačka jedinica sa mikroprogramskim generisanjem upravljačkih signala i dva tipa mikroinstrukcija

Kombinacionom mrežom KM_{OP} generišu se vrednosti (tabela 37) za realizaciju višestrukog uslovnog skoka na adresi 00 mikroprograma (tabela 38). Kombinacionu mrežu KM_{OP} čine koder CD_{OP} i ROM memorija ROM_{OP}. Na ulaze 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 i 12 kodera CD_{OP} vezani su signali **NOP**, **ADD**, **SUB**, INC, DEC, AND, OR, XOR, NOT, (ASR+LSR+ROR+RORC), (ASL+LSL+ROL+ROLC), MULU i **DIVU**, respektivno, a na adresama 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 i 12 memorije ROM_{OP} nalaze se vrednosti 00, 01, 05 09, **0D**, 11, **15**, **19**, **1D**, 21, 25**,** 29 i **30**, respektivno. U zavisnosti od toga koji od signala **NOP**, **ADD**, ..., **DIVU** ima vrednost 1 na izlazima kodera se pojavljuje adresa memorijske lokacije sa koje se čita i na linijama $op_{7...0}$ pojavljuje vrednost koji treba upisati u mikroprogramski brojač. S obzirom da se na adresi 00 mikroprograma nalazi mikroinstrukcija sa tako kodiranim poljem *cc* da njeno izvršavanje daje vrednost 1 signala višestrukog uslovnog skoka **brop**, vrednost na linijama $\mathbf{op_{7...0}}$ prolazi tada kroz multiplekser MP i pojavljuje se na ulazima mikroprogramskog brojača mPC.

Prihvatni registar mikroinstrukcije $CW_{0...55}$ u svojim razredima $CW_{48...55}$ sadrži vrednost za upis u mikroprogramski brojač $mPC_{7...0}$ za bezuslovne skokove (tabela 35) i uslovne skokove (tabela 36) u mikroprogramu (tabela 38). Signal višestrukog uslovnog skoka **brop** ima vrednost 1 samo prilikom izvršavanja mikroinstrukcije na adresi 00 mikroprograma, a u svim ostalim situacijama imaju 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 $CW_{48...55}$ 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 38). Ovaj režim rada se obezbeđuje vrednošću 1 signala **inc**. Signal **inc** ima vrednost 1 ukoliko svi signali **brop**, **brcnd** i **bruncnd** imaju vrednost 0. Signali **brop**, **brcnd** i **bruncnd** normalno imaju vrednost 0 sem prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran višestruki uslovni skok, bezuslovni skok ili neki od uslovnih skokova i uslov skoka je ispunjen, pa jedan od ovih signala ima vrednost 1.

U režimu skoka pri pojavi signala takta $\mathbf{CLK_{mPC}}$ vrši se upis nove vrednosti u mikroprogramski brojač $mPC_{7...0}$ čime se obezbeđuje odstupanje od sekvencijalnog očitavanja mikroinstrukcija iz mikroprogramske memorije (tabela 38). Ovaj režim rada se obezbeđuje vrednošću 1 signala **ld**. Signal **ld** ima vrednost 1 ukoliko jedan od signala **brop**, **brcnd** i **bruncnd** ima vrednost 1. Signali **brop**, **brcnd** i **bruncnd** normalno imaju vrednost 0 sem prilikom izvršavanja mikroinstrukcije koja ima takvo polje *cc* da je specificiran višestruki uslovni skok, 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 38). S obzirom da se mikroprogram svih operacija nalazi u opsegu od adrese 00 do adrese 35, 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 56 bitova, a kapacitet veličinom mikroprograma svih instrukcija procesora (tabela 38) i iznosi 256 lokacija. Adresiranje mikroprogramske memorije se realizuje sadržajem mikroprogramskog brojača $mPC_{7...0}$.

Blok *prihvatni registar mikroinstrukcije* sadrži prihvatni registar mikroinstrukcije $CW_{0...55}$. Prihvatni registar mikroinstrukcije $CW_{0...55}$ služi za prihvatanje mikroinstrukcije očitane iz mikroprogramske memorije mMEM. Na osnovu sadržaja ovog registra generišu se upravljački signali. Razredi $CW_{0...43}$ i $CW_{44...47}$ se koriste u bloku *generisanje upravljačkih signala* za generisanje upravljačkih signala operacione jedinice i upravljačke jedinice, respektivno, dok se razredi $CW_{48...55}$ 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 $CW_{0...43}$ prihvatnog registra mikroinstrukcije generišu upravljačke signale operacione jedinice i na osnovu sadržaja razreda $CW_{44...47}$ prihvatnog registra mikroinstrukcije i signala logičkih uslova $\mathbf{MQ_0}$, **ISCZ**, **BBZ** i $\mathbf{AB_8}$ koji dolaze iz operacione jedinice generišu upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice se generišu na sledeći način:

$mxAB_1 = \overline{CW_0} \cdot CW_{16}$

$mxAB_0 = \overline{CW_0} \cdot CW_{17}$

$ldAB = \overline{CW_0} \cdot CW_{18}$

$M = \overline{CW_0} \cdot CW_4$

$S_1 = \overline{CW_0} \cdot CW_5$

$S_0 = \overline{CW_0} \cdot CW_6$

$srAB = \overline{CW_0} \cdot CW_{19}$

$slAB = \overline{CW_0} \cdot CW_{20}$

$decISC = \overline{CW_0} \cdot CW_3$ itd.

saglasno formatu operacione mikroinstrukcije (слика 58).

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **brop** = $CW_0 \cdot \overline{CW_4} \cdot \overline{CW_5} \cdot CW_6 \cdot \overline{CW_7}$
- **bruncnd** = $CW_0 \cdot \overline{CW_4} \cdot \overline{CW_5} \cdot \overline{CW_6} \cdot CW_7$
- **brcnd** = **brnotMQ0**$\cdot \overline{\mathbf{MQ_0}}$ +**brnotISCZ**$\cdot \overline{\mathbf{ISCZ}}$ +**brBBZ**$\cdot$**BBZ**+ **brnotAB8**$\cdot \overline{\mathbf{AB_8}}$ +**brAB8**$\cdot \mathbf{AB_8}$
- **brnotMQ0** = $CW_0 \cdot \overline{CW_4} \cdot \overline{CW_5} \cdot CW_6 \cdot CW_7$
- **brnotISCZ** = $CW_0 \cdot \overline{CW_4} \cdot CW_5 \cdot \overline{CW_6} \cdot \overline{CW_7}$
- **brBBZ** = $CW_0 \cdot \overline{CW_4} \cdot CW_5 \cdot \overline{CW_6} \cdot CW_7$
- **brnotAB8**= $CW_0 \cdot \overline{CW_4} \cdot CW_5 \cdot CW_6 \cdot \overline{CW_7}$
- **brAB8**= $CW_0 \cdot \overline{CW_4} \cdot CW_5 \cdot CW_6 \cdot CW_7$

saglasno formatu upravljačke mikroinstrukcije (слика 59).

Pri generisanju signala **brcnd** koriste se signali logičkih uslova $\mathbf{MQ_0}$, **ISCZ**, **BBZ** i $\mathbf{AB_8}$ koji dolaze iz operacione jedinice.

1.4 DODATAK

1.4.1 Aritmetička operacija MULU

Algoritam po kome će se realizovati operacija množenja dve celobrojne veličine bez znaka odgovara postupku koji se koristi kada se množenje realizuje ručno na papiru. Po tom postupku proizvod P celobrojnih veličina bez znaka

$X=X_{n-1}X_{n-2}...X_1X_0$ i

$Y=Y_{n-1}Y_{n-2}...Y_1Y_0$

dužine n bita definiše se sa:

$$P=\sum_{i=0}^{n-1} \mathbf{X \cdot Y_i \cdot 2^i}$$

a može se napisati i kao

$P=0+X\cdot Y_0\cdot 2^0+X\cdot Y_1\cdot 2^1+...+X\cdot Y_i\cdot 2^i+...X\cdot Y_{n-1}\cdot 2^{n-1}$

gde je

$P=P_{2n-1}P_{2n-2}...P_0$

i ima dužinu 2*n* bita.

Po ovom postupku množenje se realizuje u *n* iteracija tako što se računaju parcijalni proizvodi P(0), P(1) do P(n-1), pri čemu je konačan proizvod P jednak zadnjem parcijalnom proizvodu P(n-1).

Ako se parcijalni proizvod

$XY_02^0+XY_12^1+...+XY_i2^i$

označi sa P(i), onda se proizvod može računati na sledeći način:

$P(0)=0+X\cdot Y_0\cdot 2^0$

$P(1)=P(0)+X\cdot Y_1\cdot 2^1$

...

$P(i)=P(i-1)+X\cdot Y_i\cdot 2^i$

...

$P(n-1)=P(i-2)+X\cdot Y_{n-1}\cdot 2^{n-1}$

gde je

$P(n-1)=P$.

Po ovom postupku se, kao i kada se množenje realizuje ručno na papiru, u prvoj iteracija najpre računa parcijalni proizvod P(0) tako što se na vrednost 0 kao početnu vrednost parcijalnog proizvoda dodaje $X \cdot Y_0 \cdot 2^0$. To praktično znači da

- kada binarna cifra Y_0 ima vrednost 1 treba uzeti $X=X_{n-1}X_{n-2}...X_1X_0$ pomnožiti sa 2^0 (u ovom slučaju to je množenje sa 1 pa se ništa ne menja) i sabrati sa 0 i
- kada binarna cifra Y_0 ima vrednost 0 treba sabrati 0 sa 0.

U drugoj iteracija se računa parcijalni proizvod P(1) tako što se na prethodno sračunati parcijalni proizvod P(0) dodaje $X \cdot Y_1 \cdot 2^1$. To praktično znači da

- kada binarna cifra Y_1 ima vrednost 1 treba uzeti $X=X_{n-1}X_{n-2}...X_1X_0$ pomnožiti sa 2^1 (u ovom slučaju to je množenje sa 2 što se realizuje pomeranjem $X=X_{n-1}X_{n-2}...X_1X_0$ za jedno mesto ulevo) i sabrati sa P(0) i
- kada binarna cifra Y_1 ima vrednost 0 treba sabrati 0 sa P(0).

U trećoj iteracija se računa parcijalni proizvod P(2) tako što se na prethodno sračunati parcijalni proizvod P(1) dodaje $X \cdot Y_2 \cdot 2^2$. To praktično znači da

- kada binarna cifra Y_2 ima vrednost 1 treba uzeti $X=X_{n-1}X_{n-2}...X_1X_0$ pomnožiti sa 2^2 (u ovom slučaju to je množenje sa 4 što se realizuje pomeranjem $X=X_{n-1}X_{n-2}...X_1X_0$ za dva mesta ulevo) i sabrati sa P(1) i
- kada binarna cifra Y_2 ima vrednost 0 treba sabrati 0 sa P(1).

Po istom postupku se računaju i parcijalni proizvodi za binarne cifre Y_3 do Y_{n-1}, pa se konačno u n-toj iteraciji računa parcijalni proizvod P(n-1) tako što se na prethodno sračunati parcijalni proizvod P(n-2) dodaje $X \cdot Y_{n-1} \cdot 2^{n-1}$. To praktično znači da

- kada binarna cifra Y_{n-1} ima vrednost 1 treba uzeti $X=X_{n-1}X_{n-2}...X_1X_0$ pomnožiti sa 2^{n-1} (u ovom slučaju to je množenje sa 2^{n-1} što se realizuje pomeranjem $X=X_{n-1}X_{n-2}...X_1X_0$ za 2^{n-1} mesta ulevo) i sabrati sa P(n-2) i
- kada binarna cifra Y_{n-1} ima vrednost 0 treba sabrati 0 sa P(n-2).

Parcijalni proizvod P(n-1) je i konačan proizvod P.

Za realizaciju algoritma je potreban sabirač dužine 2n razreda.

Kao ilustracija množenja po ovom algoritmu uzeto je da je X=1001 i Y=1111. Pošto je X jednako 9 a Y jednako 15, kao proizvod treba da se dobije 135. Koraci kojima se po ovom potupku dobija proizvod dati su na slici 61.

	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0
$X \cdot Y_0 \cdot 2^0$					1	0	0	1
$P(0)=0 + X \cdot Y_0 \cdot 2^0$	0	0	0	0	1	0	0	1
$X \cdot Y_1 \cdot 2^1$				1	0	0	1	0
$P(1)=P(0)+X \cdot Y_1 \cdot 2^1$	0	0	0	1	1	0	1	1
$X \cdot Y_2 \cdot 2^2$			1	0	0	1	0	0
$P(2)=P(1)+X \cdot Y_2 \cdot 2^2$	0	0	1	1	1	1	1	1
$X \cdot Y_3 \cdot 2^3$		1	0	0	1	0	0	0
$P(3)=P(2)+X \cdot Y_3 \cdot 2^3$	1	0	0	0	0	1	1	1

Slika 61 Množenje po originalnom algoritmu

U prvoj iteraciji se po ovom postupku najpre računa parcijalni proizvod P(0) tako što se na vrednost 0 kao početnu vrednost parcijalnog proizvoda dodaje $X \cdot Y_0 \cdot 2^0$. To praktično znači da se

- pošto binarna cifra Y_0 ima vrednost 1, uzima X=1001, množi sa 2^0 (u ovom slučaju to je množenje sa 1 pa se ništa ne menja) i sabira sa 00000000 i
- kao rezultat dobija 00001001.

U drugoj iteracija se računa parcijalni proizvod P(1) tako što se na prethodno sračunati parcijalni proizvod P(0) dodaje $X \cdot Y_1 \cdot 2^1$. To praktično znači da se

- pošto binarna cifra Y_1 ima vrednost 1 uzima X=1001, množi sa 2^1 (u ovom slučaju to je množenje sa 2 što se realizuje pomeranjem X=1001 za jedno mesto ulevo koje daje 10010) i sabira sa 00001001 i
- kao rezultat dobija 00011011.

U trećoj iteracija se računa parcijalni proizvod P(2) tako što se na prethodno sračunati parcijalni proizvod P(1) dodaje $X \cdot Y_2 \cdot 2^2$. To praktično znači da se

- pošto binarna cifra Y_2 ima vrednost 1 uzima X=1001, množi sa 2^2 (u ovom slučaju to je množenje sa 4 što se realizuje pomeranjem X=1001 za dva mesta ulevo koje daje 100100) i sabira sa 00011011 i
- kao rezultat dobija 00111111.

U četvrtoj iteraciji se računa parcijalni proizvod P(3) tako što se na prethodno sračunati parcijalni proizvod P(2) dodaje $X \cdot Y_3 \cdot 2^3$. To praktično znači da se

- pošto binarna cifra Y_3 ima vrednost 1 uzima X=1001, množi sa 2^3 (u ovom slučaju to je množenje sa 8 što se realizuje pomeranjem X=1001 za tri mesta ulevo koje daje 1001000) i sabira sa 00111111 i
- kao rezultat dobija 10000111.

Parcijalni proizvod P(3) je i konačan proizvod P.

Rezultat množenja je 135 a za realizaciju algoritma je potreban sabirač dužine 8 razreda.

Analizom koraka prilikom množenja može se sa slike zapaziti sledeće:

Prilikom formiranja parcijalnog proizvoda P(0), koji je 00001001, se u odnosu na početnu vrednost parcijalnog proizvoda, koji je 00000000, kao nove cifre formiraju samo cifre 01001 razreda 4, 3, 2, 1 i 0, dok su cifre 000 razreda 7, 6 i 5 iste. Nove cifre 01001 razreda 4, 3, 2, 1 i 0 predstavljaju prenos i četiri bita sume koji nastaju sabiranjem cifara razreda 0 do 3. Ovde je prenos nula pa je zato cifra u razredu 4 parcijalnog proizvoda P(0) ista kao cifra u razredu 4 početne vrednosti parcijalnog proizvoda.

Prilikom formiranja parcijalnog proizvoda P(1), koji je 00011011, se u odnosu na vrednost parcijalnog proizvoda P(0), koji je 00001001, kao nove cifre formiraju samo cifre 01101 razreda 5, 4, 3, 2 i 1, dok su cifre 00 razreda 7 i 6 i cifra 0 razreda 0 iste. Nove cifre 01101 razreda 5, 4, 3, 2 i 1 predstavljaju prenos i četiri bita sume koji nastaju sabiranjem cifara razreda 1 do 4. Ovde je prenos nula pa je zato cifra u razredu 5 parcijalnog proizvoda P(1) ista kao cifra u razredu 4 parcijalnog proizvoda P(0).

Prilikom formiranja parcijalnog proizvoda P(2), koji je 00111111, se u odnosu na vrednost parcijalnog proizvoda P(1), koji je 00011011, kao nove cifre formiraju samo cifre 01111 razreda 6, 5, 4, 3 i 2, dok su cifre 0 razreda 7 i cifre 11 razreda 1 i 0 iste. Nove cifre 01111 razreda 6, 5, 4, 3 i 2 predstavljaju prenos i četiri bita sume koji nastaju sabiranjem cifara razreda 2 do 5. Ovde je prenos nula pa je zato cifra u razredu 6 parcijalnog proizvoda P(2) ista kao cifra u razredu 6 parcijalnog proizvoda P(1).

Prilikom formiranja parcijalnog proizvoda P(3), koji je 10000111, se u odnosu na vrednost parcijalnog proizvoda P(2), koji je 00111111, kao nove cifre formiraju samo cifre 10000 razreda 7, 6, 5, 4 i 3, dok su cifre 111 razreda 2, 1 i 0 iste. Nove cifre 10000 razreda 7, 6, 5, 4 i 3 predstavljaju prenos i četiri bita sume koji nastaju sabiranjem cifara razreda 3 do 6. Ovde je prenos jedan pa je zato cifra u razredu 7 parcijalnog proizvoda P(3) jedan dok je cifra u razredu 7 parcijalnog proizvoda P(2) nula.

Iz ovoga se vidi da bi bilo moguće umesto sabirača dužine 8 razreda koristiti sabirač dužine 4 razreda sa čijih izlaza bi se koristili signal prenosa i četiri signala sume za formiranje pet novih cifara prilikom formiranja parcijalnih proizvoda P(0), P(1), P(2) i P(3). Međutim, razmatrani algoritam nije pogodan za realizaciju jer bi

- prilikom računaja P(0) trebalo na ulaze sabirača dužine 4 razreda voditi signale razreda 3 do 0, a signal prenosa i četiri signala sume sa izlaza sabirača voditi u razrede 4 do 0 parcijalnog proizvoda P(0),
- prilikom računaja P(1) trebalo na ulaze sabirača dužine 4 razreda voditi signale razreda 4 do 1, a signal prenosa i četiri signala sume sa izlaza sabirača voditi u razrede 5 do 1 parcijalnog proizvoda P(1),
- prilikom računaja P(2) trebalo na ulaze sabirača dužine 4 razreda voditi signale razreda 5 do 2, a signal prenosa i četiri signala sume sa izlaza sabirača voditi u razrede 6 do 2 parcijalnog proizvoda P(2),
- prilikom računaja P(3) trebalo na ulaze sabirača dužine 4 razreda voditi signale razreda 6 do 3, a signal prenosa i četiri signala sume sa izlaza sabirača voditi u razrede 7 do 3 parcijalnog proizvoda P(3).

Originalni izraz za izračunavanje proizvoda P se može transformisati u oblik koji je pogodniji za realizaciju operacije množenja korišćenjem ne sabirača dužine 2n razreda već samo n razreda.

Ako se izraz za P napiše u obliku:

$$P=\sum_{i=0}^{n-1} X \cdot Y_i \cdot 2^i = \sum_{i=0}^{n-1} X \cdot Y_i \cdot 2^i \cdot 2^n \cdot 2^{-n} = \sum_{i=0}^{n-1} (X \cdot Y_i) \cdot 2^n \cdot 2^{-(n-i)}$$

odnosno

$$P=(...(...((0+X\cdot Y_0\cdot 2^n)\cdot 2^{-1}+X\cdot Y_1\cdot 2^n)\cdot 2^{-1}+...+X\cdot Y_i\cdot 2^n)\cdot 2^{-1}+...+X\cdot Y_{n-1}\cdot 2^n)\cdot 2^{-1}$$

i parcijalni proizvod

$$(...((0+X\cdot Y_0\cdot 2^n)\cdot 2^{-1}+X\cdot Y_1\cdot 2^n)\cdot 2^{-1}+...+X\cdot Y_i\cdot 2^n)\cdot 2^{-1}$$

označi sa P(i), onda se proizvod može računati na sledeći način:

$P(0)=(0 + X\cdot Y_0\cdot 2^n)\cdot 2^{-1}$

$P(1)=(P(0)+X\cdot Y_1\cdot 2^n)\cdot 2^{-1}$

...

$P(i)=(P(i-1)+X\cdot Y_i\cdot 2^n)\cdot 2^{-1}$

...

$P(n-1)=(P(i-2)+X\cdot Y_{n-1}\cdot 2^n)\cdot 2^{-1}$

gde je

$P(n-1)=P$

Parcijalni proizvod P(n-1) je i konačan proizvod P.

Za realizaciju algoritma je i u ovom slučaju potreban sabirač dužine 2n razreda.

Po ovom postupku se u prvoj iteracija najpre računa parcijalni proizvod P(0) tako što se na vrednost 0 kao početnu vrednost parcijalnog proizvoda dodaje $X \cdot Y_0 \cdot 2^n$ i dobijena suma množi sa 2^{-1}. To praktično znači da

- kada binarna cifra Y_0 ima vrednost 1 treba uzeti $X=X_{n-1}X_{n-2}...X_1X_0$ pomnožiti sa 2^n (što se realizuje pomeranjem $X=X_{n-1}X_{n-2}...X_1X_0$ za n mesta ulevo) i sabrati sa 0 i
- kada binarna cifra Y_0 ima vrednost 0 treba sabrati 0 sa 0 i
- u oba slučaja dobijenu sumu pomnožiti sa 2^{-1} što se realizuje pomeranjem dobijene sume udesno za jedno mesto.

U drugoj iteracija se računa parcijalni proizvod P(1) tako što se na prethodno sračunati parcijalni proizvod P(0) dodaje $X \cdot Y_1 \cdot 2^n$ i dobijena suma množi sa 2^{-1}. To praktično znači da

- kada binarna cifra Y_1 ima vrednost 1 treba uzeti $X=X_{n-1}X_{n-2}...X_1X_0$ pomnožiti sa 2^n (što se realizuje pomeranjem $X=X_{n-1}X_{n-2}...X_1X_0$ za n mesto ulevo) i sabrati sa P(0) i
- kada binarna cifra Y_1 ima vrednost 0 treba sabrati 0 sa P(0) i
- u oba slučaja dobijenu sumu pomnožiti sa 2^{-1} što se realizuje pomeranjem dobijene sume udesno za jedno mesto.

U trećoj iteracija se računa parcijalni proizvod P(2) tako što se na prethodno sračunati parcijalni proizvod P(1) dodaje $X \cdot Y_2 \cdot 2^n$ i dobijena suma množi sa 2^{-1}. To praktično znači da

- kada binarna cifra Y_2 ima vrednost 1 treba uzeti $X=X_{n-1}X_{n-2}...X_1X_0$ pomnožiti sa 2^n (što se realizuje pomeranjem $X=X_{n-1}X_{n-2}...X_1X_0$ za n mesta ulevo) i sabrati sa P(1) i
- kada binarna cifra Y_2 ima vrednost 0 treba sabrati 0 sa P(1) i
- u oba slučaja dobijenu sumu pomnožiti sa 2^{-1} što se realizuje pomeranjem dobijene sume udesno za jedno mesto.

Po istom postupku se računaju i parcijalni proizvodi za binarne cifre Y_3 do Y_{n-1}, pa se konačno u n-toj iteraciji računa parcijalni proizvod P(n-1) tako što se na prethodno sračunati parcijalni proizvod P(n-2) dodaje $X \cdot Y_{n-1} \cdot 2^n$ i dobijena suma množi sa 2^{-1}. To praktično znači da

- kada binarna cifra Y_{n-1} ima vrednost 1 treba uzeti $X=X_{n-1}X_{n-2}...X_1X_0$ pomnožiti sa 2^n (što se realizuje pomeranjem $X=X_{n-1}X_{n-2}...X_1X_0$ za n mesta ulevo) i sabrati sa P(n-1) i
- kada binarna cifra Y_{n-1} ima vrednost 0 treba sabrati 0 sa P(n-1) i
- u oba slučaja dobijenu sumu pomnožiti sa 2^{-1} što se realizuje pomeranjem dobijene sume udesno za jedno mesto.

Parcijalni proizvod P(n-1) je i konačan proizvod P.
Za realizaciju algoritma je potreban sabirač dužine 2n razreda.

Kao ilustracija množenja po modifikovanom algoritmu ponovo je uzeto da je X=1001 i Y=1111. Pošto je X jednako 9 a Y jednako 15, kao proizvod treba da se dobije 135. Koraci kojima se po ovom postupku dobija proizvod dati su na slici 62.

	8	7	6	5	4	3	2	1	0
0		0	0	0	0	0	0	0	0
$X \cdot Y_0 \cdot 2^4$		1	0	0	1	0	0	0	0
$0 + X \cdot Y_0 \cdot 2^4$	0	1	0	0	1	0	0	0	0
$P(0)=(0+X \cdot Y_0 \cdot 2^4) \cdot 2^{-1}$		0	1	0	0	1	0	0	0
$X \cdot Y_1 \cdot 2^4$		1	0	0	1	0	0	0	0
$P(0)+X \cdot Y_1 \cdot 2^4$	0	1	1	0	1	1	0	0	0
$P(1)=(P(0)+X \cdot Y_1 \cdot 2^4) \cdot 2^{-1}$		0	1	1	0	1	1	0	0
$X \cdot Y_2 \cdot 2^4$		1	0	0	1	0	0	0	0
$P(1)+X \cdot Y_2 \cdot 2^4$	0	1	1	1	1	1	1	0	0
$P(2)=(P(1)+X \cdot Y_2 \cdot 2^4) \cdot 2^{-1}$		0	1	1	1	1	1	1	0
$X \cdot Y_3 \cdot 2^4$		1	0	0	1	0	0	0	0
$P(2)+X \cdot Y_3 2^4$	1	0	0	0	0	1	1	1	0
$P(3)=(P(2)+X \cdot Y_3 \cdot 2^4) \cdot 2^{-1}$		1	0	0	0	0	1	1	1

Slika 62 Množenje po modifikovanom algoritmu

U prvoj iteraciji se po ovom postupku najpre računa parcijalni proizvod P(0) tako što se na vrednost 0 kao početnu vrednost parcijalnog proizvoda dodaje $X \cdot Y_0 \cdot 2^4$ i dobijena suma množi sa 2^{-1}. To praktično znači da se

- pošto binarna cifra Y_0 ima vrednost 1, uzima X=1001, množi sa 2^4 (u ovom slučaju to je množenje sa 16 što se realizuje pomeranjem X=1001 za četiri mesta ulevo i dobija 10010000), sabira sa 00000000 i dobija 010010000, pomera za jedno mesto udesno i
- kao rezultat dobija 01001000.

U drugoj iteracija se računa parcijalni proizvod P(1) tako što se na prethodno sračunati parcijalni proizvod P(0) dodaje $X \cdot Y_1 \cdot 2^4$ i dobijena suma množi sa 2^{-1}. To praktično znači da se

- pošto binarna cifra Y_1 ima vrednost 1 uzima X=1001, množi sa 2^4 (u ovom slučaju to je množenje sa 16 što se realizuje pomeranjem X=1001 za četiri mesta ulevo i dobija 10010000), sabira sa 01001000 i dobija 011011000, pomera za jedno mesto udesno i
- kao rezultat dobija 01101100.

U trećoj iteracija se računa parcijalni proizvod P(2) tako što se na prethodno sračunati parcijalni proizvod P(1) dodaje $X \cdot Y_2 \cdot 2^4$ i dobijena suma množi sa 2^{-1}. To praktično znači da se

- pošto binarna cifra Y_2 ima vrednost 1 uzima X=1001, množi sa 2^4 (u ovom slučaju to je množenje sa 16 što se realizuje pomeranjem X=1001 za četiri mesta ulevo i dobija 10010000), sabira sa 01101100 i dobija 011111100, pomera za jedno mesto udesno i
- kao rezultat dobija 01111110.

U četvrtoj iteraciji se računa parcijalni proizvod P(3) tako što se na prethodno sračunati parcijalni proizvod P(2) dodaje $X \cdot Y_3 \cdot 2^4$ i dobijena suma množi sa 2^{-1}. To praktično znači da se

- pošto binarna cifra Y_3 ima vrednost 1 uzima X=1001, množi sa 2^4 (u ovom slučaju to je množenje sa 16 što se realizuje pomeranjem X=1001 za četiri mesta ulevo i dobija 10010000), sabira sa 01111110 i dobija 100001110, pomera za jedno mesto udesno i
- kao rezultat dobija 10000111.

Parcijalni proizvod P(3) je i konačan proizvod P.

Rezultat množenja je 135 a za realizaciju algoritma je potreban sabirač dužine 8 razreda.

Analizom koraka prilikom množenja može se sa slike zapaziti sledeće:

Prilikom formiranja $0+X\cdot Y_0\cdot 2^4$, koji je 010010000, se u odnosu na početnu vrednost parcijalnog proizvoda, koji je 00000000, kao nove cifre formiraju samo cifre 01001 razreda 8, 7, 6, 5 i 4, dok su cifre 0000 razreda 3, 2, 1 i 0 iste. Nove cifre 01001 razreda 8, 7, 6, 5 i 4, predstavljaju bit prenosa i četiri bita sume koji nastaju sabiranjem cifara razreda 4 do 7. Treba uočiti da su ovo iste cifre kao cifre koje se dobijaju sabiranjem cifara razreda 0 do 3 u originalnom algoritmu.

Prilikom formiranja $P(0)+X\cdot Y_1\cdot 2^4$, koji je 011011000, se u odnosu na vrednost parcijalnog proizvoda P(0), koji je 01001000, kao nove cifre formiraju samo cifre 01101 razreda 8, 7, 6, 5 i 4, dok su cifre 1000 razreda 3, 2, 1 i 0 iste. Nove cifre 01101 razreda 8, 7, 6, 5 i 4, predstavljaju bit prenosa i četiri bita sume koji nastaju sabiranjem cifara razreda 4 do 7. Treba uočiti da su ovo iste cifre kao cifre koje se dobijaju sabiranjem cifara razreda 1 do 4 u originalnom algoritmu.

Prilikom formiranja $P(1)+X\cdot Y_2\cdot 2^4$, koji je 011111100, se u odnosu na vrednost parcijalnog proizvoda P(1), koji je 01101100, kao nove cifre formiraju samo cifre 01111 razreda 8, 7, 6, 5 i 4, dok su cifre 1100 razreda 3, 2, 1 i 0 iste. Nove cifre 01111 razreda 8, 7, 6, 5 i 4, predstavljaju bit prenosa i četiri bita sume koji nastaju sabiranjem cifara razreda 4 do 7. Treba uočiti da su ovo iste cifre kao cifre koje se dobijaju sabiranjem cifara razreda 2 do 5 u originalnom algoritmu.

Prilikom formiranja $P(2)+X\cdot Y_3\cdot 2^4$, koji je 100001110, se u odnosu na vrednost parcijalnog proizvoda P(2), koji je 01111110, kao nove cifre formiraju samo cifre 10000 razreda 8, 7, 6, 5 i 4, dok su cifre 1110 razreda 3, 2, 1 i 0 iste. Nove cifre 10000 razreda 8, 7, 6, 5 i 4, predstavljaju bit prenosa i četiri bita sume koji nastaju sabiranjem cifara razreda 4 do 7. Treba uočiti da su ovo iste cifre kao cifre koje se dobijaju sabiranjem cifara razreda 3 do 6 u originalnom algoritmu.

Modifikovani algoritam je pogodniji u odnosu na originalni algoritam za realizaciju operacione jedinice koja bi umesto sabirača dužine 8 razreda koristila sabirač dužine 4 razreda.

U originalnom algoritmu se 4 cifre koje se dovede na ulaze sabirača dužine 4 razreda pomeraju za svaku od četiri iteracije, pa se u prvoj iteraciji dovode cifre 3 do 0, u drugoj cifre 4 do 1, u trećoj cifre 5 do 2 i u četvrtoj iteraciji cifre 6 do 3. Stoga bi, ukoliko bi se koristio sabirač dužine 4 razreda, bilo potrebno multipleksiranje odgovarajućih cifara na ulaze sabirača za svaku od četiri iteracije. Pored toga signal prenosa i četiri signala sume predstavljaju cifre 4 do 0 u parcijalnom proizvodu P(0) prve iteracije, cifre 5 do 1 u parcijalnom proizvodu P(1) druge iteracije, cifre 6 do 2 u parcijalnom proizvodu P(2) treće iteracije i cifre 7 do 3 u parcijalnom proizvodu P(3) treće iteracije. Stoga bi, ukoliko bi se koristio sabirač dužine 4 razreda, bilo potrebno distribuiranje signal prenosa i četiri signala sume u odgovarajuće cifre elementarnih proizvoda P(0), P(1), P(2) i P(3).

U modifikovanom algoritmu se na ulaze sabirača dužine 4 razreda u sve četiri iteracije dovode cifre 4 do 7. Pored toga u sve četiri iteracije signal prenosa i četiri signala sume predstavljaju razrede 8 do 4 suma $0+X\cdot Y_0\cdot 2^4$, $P(0)+X\cdot Y_1\cdot 2^4$, $P(1)+X\cdot Y_2\cdot 2^4$ i $P(2)+X\cdot Y_3\cdot 2^4$ koje se formiraju u prvoj drugoj, trećoj i četvrtoj iteraciji, respektivno. U modifikovanom algoritmu postoji potreba da se pomeraju $0+X\cdot Y_0\cdot 2^4$, $P(0)+X\cdot Y_1\cdot 2^4$, $P(1)+X\cdot Y_2\cdot 2^4$ i $P(2)+X\cdot Y_3\cdot 2^4$ za jedno mesto udesno. S toga je pored sabirača dužine 4 razreda potreban i pomerački registar udesno dužine 4 razreda. U njega bi se pomeranjem udesno posle svakog sabiranja $0+X\cdot Y_0\cdot 2^4$, $P(0)+X\cdot Y_1\cdot 2^4$, $P(1)+X\cdot Y_2\cdot 2^4$ i $P(2)+X\cdot Y_3\cdot 2^4$ ubacivala cifra koja ne treba da učestvuje u sabiranju u sledećoj iteraciji.

U posmatranom primeru početna vrednost pomeračkog registra je 0000 a na kraju prve iteracije posle pomeranja $0+X\cdot Y_0\cdot 2^4$ za jedno mesto udesno postaje 1000. Na kraju druge iteracije posle pomeranja $P(0)+X\cdot Y_1\cdot 2^4$ za jedno mesto udesno postaje 1100. Na kraju treće iteracije posle pomeranja $P(1)+X\cdot Y_2\cdot 2^4$ za jedno mesto udesno postaje 1110. Na kraju četvrte iteracije posle pomeranja $P(2)+X\cdot Y_3\cdot 2^4$ za jedno mesto udesno postaje 0111.

Na osnovu ovih razmatranja dolazi se do moguće strukturne šeme operacione jedinice i dijagrama toka.

Potrebni su registri $A=A_3A_2A_1A_0$ i $B=B_3B_2B_1B_0$ dužine 4 razreda za smeštanje početnih vrednosti $X=X_3X_2X_1X_0$ i $Y=Y_3Y_2Y_1Y_0$ dužine 4 bita, koje su X=1001 i Y=1111, respektivno.

Registar $B=B_3B_2B_1B_0$ treba da ima mogućnost pomeranje udesno. Pomeranje udesno treba da se realizuje na kraju svake od četiri iteracije. Kao rezultat u razredu B_0 će u prvoj, drugoj, trećoj i četvrtoj iteraciji da budu bitovi Y_0, Y_1, Y_2, Y_3, respektivno.

Potreban je sabirač dužine 4 razreda koji daje bit prenosa i četiri bita sume.

Potreban je registar $P=P_3P_2P_1P_0$ dužine 4 razreda za smeštanje sume dužine 4 bita sa izlaza sabirača i dodatni jednorazredni registar P_4 za smeštanje dodatnog petog bita prenosa sa izlaza sabirača. Početne vrednosti registara P_4 i $P_3P_2P_1P_0$ treba da budu 0 i 0000, respektivno.

Potreban je dodatni registar $MQ=MQ_3MQ_2MQ_1MQ_0$ dužine 4 razreda u koji se u 4 iteracije pomeranjem udesno za jedno mesto registara P_4, $P_3P_2P_1P_0$ i $MQ_3MQ_2MQ_1MQ_0$ upisuje bit iz razreda P_0 koji ne treba da učestvuje u sabiranju u sledećoj iteraciji. Posle četiri iteracije u razredima $P_3P_2P_1P_0$ se nalaze četiri starija a u razredima $MQ_3MQ_2MQ_1MQ_0$ četiri mlađa bita proizvoda dužine osam bitova.

Na ulaze sabirača treba da se dovedu registar $P_3P_2P_1P_0$, čiji sadržaj predstavlja bitove 7, 6, 5 i 4 parcijalnog proizvoda koji treba da učestvuju u sabiranju, i registar $A_3A_2A_1A_0$, čiji sadržaj predstavlja veličinu $X_3X_2X_1X_0$.

U svako iteraciji treba proveriti razred B_0 u kome se u prvoj, drugoj, trećoj i četvrtoj iteraciji nalaze bitovi Y_0, Y_1, Y_2, Y_3, respektivno. Ukoliko je u razredu B_0 vrednost 1 treba $P_3P_2P_1P_0$ i $A_3A_2A_1A_0$ sabrati, bit prenosa i četiri bita sume upisati u registre P_4 i $P_3P_2P_1P_0$, respektivno, i registre P_4, $P_3P_2P_1P_0$ i $MQ_3MQ_2MQ_1MQ_0$ pomeriti udesno za jedno mesto. Ukoliko je u razredu B_0 vrednost 0 ne treba $P_3P_2P_1P_0$ i $A_3A_2A_1A_0$ sabirati, već odmah registre P_4, $P_3P_2P_1P_0$ i $MQ_3MQ_2MQ_1MQ_0$ pomeriti udesno za jedno mesto.

Stanja registara u operacionoj jedinici u svakoj od četiri iteracije operacije množenja prema modifikovanom algoritmu mogu se predstaviti slikom 63. Stanja registara prate modifikovani algoritam množenja sa slike 62.

	Kolone									
	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	Vrsta 1
$X \cdot Y_0 \cdot 2^4$		1	0	0	1	0	0	0	0	Vrsta 2
$0 + X \cdot Y_0 \cdot 2^4$	0	1	0	0	1	0	0	0	0	Vrsta 3
$P(0)=(0+X \cdot Y_0 \cdot 2^4) \cdot 2^{-1}$	0	0	1	0	0	1	0	0	0	Vrsta 4
$X \cdot Y_1 \cdot 2^4$		1	0	0	1	0	0	0	0	Vrsta 5
$P(0)+X \cdot Y_1 \cdot 2^4$	0	1	1	0	1	1	0	0	0	Vrsta 6
$P(1)=(P(0)+X \cdot Y_1 \cdot 2^4) \cdot 2^{-1}$	0	0	1	1	0	1	1	0	0	Vrsta 7
$X \cdot Y_2 \cdot 2^4$		1	0	0	1	0	0	0	0	Vrsta 8
$P(1)+X \cdot Y_2 \cdot 2^4$	0	1	1	1	1	1	1	0	0	Vrsta 9
$P(2)=(P(1)+X \cdot Y_2 \cdot 2^4) \cdot 2^{-1}$	0	0	1	1	1	1	1	1	0	Vrsta 10
$X \cdot Y_3 \cdot 2^4$		1	0	0	1	0	0	0	0	Vrsta 11
$P(2)+X \cdot Y_3 \cdot 2^4$	1	0	0	0	0	1	1	1	0	Vrsta 12
$P(3)=(P(2)+X \cdot Y_3 \cdot 2^4) \cdot 2^{-1}$		1	0	0	0	0	1	1	1	Vrsta 13

Slika 63 Stanja prilikom množenja po modifikovanom algoritmu

Vrsta 1. Kolona 8 predstavlja P_4, kolone 7, 6, 5 i 4 predstavljaju P_3, P_2, P_1 i P_0 i kolone 3, 2, 1 i 0 predstavljaju MQ_3, MQ_2, MQ_1 i MQ_0. Početne vrednosti u svim razredima ovih registara su 0.

Vrsta 2. Kolone 7, 6, 5 i 4 predstavljaju sadržaj razreda A_3, A_2, A_1 i A_0 u kojima se nalaze bitovi 1001 broja X. Ovi bitovi učestvuju u sabiranju sa P_3, P_2, P_1 i P_0 iz kolona 7, 6, 5 i 4 iz vrste 1, jer bit Y_0 koji se tada nalazi u B_0 ima vrednost 1.

Vrsta 3. Bit prenosa koji je 0 i četiri bita sume koji su 1001 i koji nastaju kao rezultat sabiranja bitova iz kolona 7, 6, 5 i 4 iz vrste 1 i vrste 2 upisuju se u P_4 kolone 8 i P_3, P_2, P_1 i P_0 kolona 7, 6, 5 i 4 vrste 3. U kolonama 3, 2, 1 i 0 vrste 3 nalaze se MQ_3, MQ_2, MQ_1 i MQ_0 koji ne učestvuju u sabiranju i ostaju nepromenjeni 0000 kao u kolonama 3, 2, 1 i 0 vrste 1.

Vrsta 4. Registar P_4 (kolona 8) koji je 0, registar $P_3P_2P_1P_0$ (kolone 7, 6, 5 i 4) koji je 1001 i registar $MQ_3MQ_2MQ_1MQ_0$ (kolone 3, 2, 1 i 0) koji je 0000 u vrsti 3 pomera se udesno za jedno mesto, a u P_4 upisuje nula. Zbog toga je sada u vrsti 4 u registru P_4 (kolona 8) vrednost 0, u registru $P_3P_2P_1P_0$ (kolone 7, 6, 5 i 4) vrednost 0100 i u registru $MQ_3MQ_2MQ_1MQ_0$ (kolone 3, 2, 1 i 0) vrednost 1000. Time je u registrima $P_3P_2P_1P_0$ (kolone 7, 6, 5 i 4) i $MQ_3MQ_2MQ_1MQ_0$ (kolone 3, 2, 1 i 0) formiran parcijalni proizvod P(0). Sada se pomera udesno i registar $B_3B_2B_1B_0$, pa se u razredu B_0 sada nalazi bit Y_1. Time je završena prva iteracija.

Vrste 5 i 6. U drugoj iteraciji se na osnovu kolona 7, 6, 5 i 4 vrste 4 (0100) i vrste 5 (1001) formiraju najpre kolone 8, 7, 6, 5 i 4 vrste 6 (01101) dok kolone 3, 2, 1 i 0 vrste 6 (1000) ostaju iste kao i za vrstu 4. I ovde je u kolonama 7, 6, 5 i 4 vrste 5 vrednost 1001, zbog toga što Y_1 koje se nalazi u B_0 ima vrednost 1.

Vrsta 7. Registar P_4 (kolona 8) koji je 0, registar $P_3P_2P_1P_0$ (kolone 7, 6, 5 i 4) koji je 1101 i registar $MQ_3MQ_2MQ_1MQ_0$ (kolone 3, 2, 1 i 0) koji je 1000 u vrsti 6 pomera se udesno za jedno mesto, a u P_4 upisuje nula. Zbog toga je sada u vrsti 7 u registru P_4 (kolona 8) vrednost 0, u registru $P_3P_2P_1P_0$ (kolone 7, 6, 5 i 4) vrednost 0110 i u registru $MQ_3MQ_2MQ_1MQ_0$ (kolone 3, 2, 1 i 0) vrednost 1100. Time je u registrima $P_3P_2P_1P_0$ (kolone 7, 6, 5 i 4) i $MQ_3MQ_2MQ_1MQ_0$ (kolone 3, 2, 1 i 0) formiran parcijalni proizvod P(1). Sada se pomera udesno i registar $B_3B_2B_1B_0$, pa se u razredu B_0 sada nalazi bit Y_2. Time je završena druga iteracija.

Postupak kojim je u drugoj iteraciji na osnovu vrsta 4 i 5 formirana najpre vrsta 6 a potom pomeranjem udesno i vrsta 7 sa parcijalnim proizvodom P(1), je isti i kao postupak kojim je u prvoj iteraciji na osnovu vrsta 1 i 2 formirana najpre vrsta 3 a potom pomeranjem udesno i vrsta 4 sa parcijalnim proizvodom P(0). Na isti način se formiraju i vrsta 10 sa parcijalnim proizvodima P(2) i vrsta 13 sa parcijalnim proizvodom P(3) koji je i konačan proizvod.

Po analogiji se realizuje i množenje binarnih reči dužine n bitova.

1.4.2 Aritmetička operacija DIVU

Aritmetička operacija DIVU realizuje deljenje celobrojnih vrednosti bez znaka pri čemu su deljenik $X=X_{15}X_{14}\ldots X_0$ i delilac $Y=Y_7Y_6\ldots Y_0$, dok je rezultat operacije količnik $Q=Q_7Q_6\ldots Q_0$ i ostatak $Z=Z_7Z_6\ldots Z_0$.

1.4.2.1 Originalni algoritam

Ako se u algoritmu deljenja sa $Z(8)$ do $Z(0)$ označe trenutni ostaci, a sa Q_7 do Q_0 cifre količnika $Q_{7..0}$, onda se algoritam deljenja može opisati na sledeći način:

1. $Z(8) = X - 2^8Y$

   if $Z(8) \geq 0$ *then* prekoračenje

   else $Z(7) = (Z(8) + 2^8Y) - 2^7Y$

2. if $Z(7) \geq 0$ then $Q_7 = 1$, $Z(6) = Z(7) - 2^6Y$, $i = 7, 6, \ldots, 1$

   else $Q_7 = 0$, $Z(6) = (Z(7) + 2^7Y) - 2^6Y$, $i = 7$

   if $Z(6) \geq 0$ then $Q_6 = 1$, $Z(5) = Z(6) - 2^5Y$, $i = 6$

   else $Q_6 = 0$, $Z(5) = (Z(6) + 2^6Y) - 2^5Y$, $i = 6$

   if $Z(5) \geq 0$ then $Q_5 = 1$, $Z(4) = Z(5) - 2^4Y$, $i = 5$

   else $Q_5 = 0$, $Z(4) = (Z(5) + 2^5Y) - 2^4Y$, $i = 5$

   if $Z(4) \geq 0$ then $Q_4 = 1$, $Z(3) = Z(4) - 2^3Y$, $i = 4$

else $Q_4 = 0$, $Z(3) = (Z(4) + 2^4Y) - 2^3Y$, $i = 4$

if $Z(3) \geq 0$ then $Q_3 = 1$, $Z(2) = Z(3) - 2^2Y$, $i = 3$

else $Q_3 = 0$, $Z(2) = (Z(3) + 2^3Y) - 2^2Y$, $i = 3$

if $Z(2) \geq 0$ then $Q_2 = 1$, $Z(1) = Z(2) - 2^1Y$, $i = 2$

else $Q_2 = 0$, $Z(1) = (Z(2) + 2^2Y) - 2^1Y$, $i = 2$

if $Z(1) \geq 0$ then $Q_1 = 1$, $Z(0) = Z(1) - 2^0Y$, $i = 1$

else $Q_1 = 0$, $Z(0) = (Z(1) + 2^1Y) - 2^0Y$, $i = 1$

3. if $Z(0) \geq 0$ then $Q_0 = 1$, $Z = Z(0)$

else $Q_0 = 0$, $Z = Z(0) + 2^0Y$

Na početku je potrebno da se izvrši provera da li deljenik $X = X_{15}X_{14}\ldots X_0$ i delilac $Y = Y_7Y_6\ldots Y_0$ imaju takve vrednosti da količnik $Q=Q_7Q_6\ldots Q_0$ može da se predstavi sa osam cifara. Stoga se, najpre, izračunava trenutni ostatak $Z(8)$, tako što se od deljenika X oduzme 2^8Y, a zatim vrši provera da li je $Z(8)\geq 0$ ili $Z(8)<0$. Ako je $Z(8)\geq 0$, količnik bi imao više od osam cifara. U tom slučaju dolazi do prekoračenja i operacija deljenja se završava. Ako je $Z(8)<0$, količnik će imati osam cifara. U tom slučaju operacija deljenja započinje sračunavanjem trenutnog ostatka Z(7) tako što se izvrši, najpre, obnavljanje trenutnog ostatka Z(8) dodavanjem 2^8Y, a zatim od dobijene vrednosti oduzimanje 2^7Y. Na osnovu toga se za trenutni ostatak Z(7) dobija $Z(7)=(Z(8)+2^8Y)-2^7Y$.

Slede koraci realizovani u sedam iteracija, $i = 7, 6, \ldots, 1$, u kojima se cifre količnika Q_7 do Q_1 dobijaju na osnovu vrednosti trenutnih ostataka Z(7) do Z(1). U njima se izračunavaju i trenutni ostaci Z(6) do Z(0). Ako je u i-toj iteraciji $Z(i)\geq 0$, tada je cifra $Q_i = 1$. U istoj iteraciji se, u cilju dobijanja trenutnog ostatka $Z(i\text{-}1)$, mora od trenutnog ostatka $Z(i)$ izvršiti oduzimanje $2^{i-1}Y$. Na osnovu toga se dobija $Z(i\text{-}1)= Z(i)-2^{i-1}Y$. Ako je i-toj iteraciji $Z(i) < 0$, tada je cifra $Q_i = 0$. U istom koraku se, u cilju dobijanja trenutnog ostatka $Z(i\text{-}1)$, mora izvršiti obnavljanje trenutnog ostatka Z(i) dodavanjem 2^iY i od dobijene vrednosti oduzimanje $2^{i-1}Y$. Na osnovu toga se dobija $Z(i\text{-}1) = (Z(i)+2^iY)-2^{i-1}Y$.

Na kraju se dobija cifra količnika Q_0 na osnovu vrednosti trenutnog ostatka Z(0) i izačunava ostatak Z. Ako je $Z(0)\geq 0$, tada je cifra $Q_0 = 1$. Pored toga ostatak Z dobija vrednost trenutnog ostatka $Z(0)$. Ako je $Z(0)<0$, tada je cifra $Q_0 = 0$. Pored toga ostatak Z dobija vrednost obnovljenog trenutnog ostatka Z(0) koji se dobija dodavanjem 2^0Y.

Za realizaciju operacije deljenja po ovom algoritmu potrebna je ALU dužine 16 razreda, koja realizuje sabiranje i oduzimanje na dužini od 16 razreda.

Ovaj algoritam se može koncizno predstaviti na sledeći način:

1. $Z(8) = X - 2^8Y$

if $Z(8) \geq 0$ *then* prekoračenje

else $Z(7) = (Z(8) + 2^8Y) - 2^7Y$

2. if $Z(i) \geq 0$ *then* $Q_i = 1$, $Z(i\text{-}1) = Z(i) - 2^{i-1}Y$, $i = 7, 6, \ldots, 1$

else $Q_i = 0$, $Z(i\text{-}1) = (Z(i) + 2^iY) - 2^{i-1}Y$, $i = 7, 6, \ldots, 1$

3. if $Z(0) \geq 0$ *then* $Q_0 = 1$, $Z = Z(0)$

else $Q_0 = 0$, $Z = Z(0) + 2^0Y$

1.4.2.2 Modifikovani algoritam

Ako se u algoritmu deljenja sa $Z(8)$ do $Z(0)$ označe trenutni ostaci, a sa Q_7 do Q_0 cifre količnika $Q_{7..0}$, onda se algoritam deljenja može opisati na sledeći način:

1. $Z(8) = X - 2^8Y$

   if $Z(8) \geq 0$ *then* prekoračenje
   else $Z(7) = (Z(8) + 2^8Y) - 2^7Y$

$Z(7)\cdot 2^1 = (Z(8) + 2^8Y)\cdot 2^1 - 2^7Y\cdot 2^1$
$Z(7)^* = Z(7)\cdot 2^1 = (Z(8) + 2^8Y)\cdot 2^1 - 2^7Y\cdot 2^1$
$Z(7)^* = Z(7)\cdot 2^1 = Z(8)\cdot 2^1 + 2^8Y\cdot 2^1 - 2^8Y$
$Z(7)^* = Z(7)\cdot 2^1 = Z(8)\cdot 2^1 + 2^8Y$

---

2. *if* $Z(7) \geq 0$ *then* $Q_7 = 1,\ Z(6) = Z(7) - 2^6Y,\ i = 7$
   else $Q_7 = 0,\ Z(6) = (Z(7) + 2^7Y) - 2^6Y,\ i = 7$

---

then $Q_7 = 1,\ Z(6) = Z(7) - 2^6Y,\ i = 7$
$Z(6)\cdot 2^2 = Z(7)\cdot 2^2 - 2^6Y\cdot 2^2,\ i = 7$
$Z(6)^* = Z(6)\cdot 2^2 = Z(7)\cdot 2^1 \cdot 2^1 - 2^8Y,\ i = 7$
$Z(6)^* = Z(6)\cdot 2^2 = Z(7)^*\cdot 2^1 - 2^8Y,\ i = 7$
else $Q_7 = 0,\ Z(6) = (Z(7) + 2^7Y) - 2^6Y,\ i = 7$
$Z(6)\cdot 2^2 = (Z(7) + 2^7Y)\cdot 2^2 - 2^6Y\cdot 2^2,\ i = 7$
$Z(6)^* = Z(6)\cdot 2^2 = (Z(7)\cdot 2^1 + 2^7\cdot 2^1Y)\cdot 2^1 - 2^8Y,\ i = 7$
$Z(6)^* = Z(6)\cdot 2^2 = Z(7)^*\cdot 2^1 + 2^8Y\cdot 2^1 - 2^8Y,\ i = 7$
$Z(6)^* = Z(6)\cdot 2^2 = Z(7)^*\cdot 2^1 + 2^8Y,\ i = 7$

---

if $Z(6) \geq 0$ *then* $Q_6 = 1,\ Z(5) = Z(6) - 2^5Y,\ i = 6$
else $Q_6 = 0,\ Z(5) = (Z(6) + 2^6Y) - 2^5Y,\ i = 6$

---

then $Q_6 = 1,\ Z(5) = Z(6) - 2^5Y,\ i = 6$
$Z(5)\cdot 2^3 = Z(6)\cdot 2^3 - 2^5Y\cdot 2^3,\ i = 6$
$Z(5)^* = Z(5)\cdot 2^3 = Z(6)\cdot 2^2\cdot 2^1 - 2^8Y,\ i = 6$
$Z(5)^* = Z(5)\cdot 2^3 = Z(6)^*\cdot 2^1 - 2^8Y,\ i = 6$
else $Q_6 = 0,\ Z(5) = (Z(6) + 2^6Y) - 2^5Y,\ i = 6$
$Z(5)\cdot 2^3 = (Z(6) + 2^6Y)\cdot 2^3 - 2^5Y\cdot 2^3,\ i = 6$
$Z(5)^* = Z(5)\cdot 2^3 = (Z(6)\cdot 2^2 + 2^6\cdot 2^2Y)\cdot 2^1 - 2^8Y,\ i = 6$
$Z(5)^* = Z(5)\cdot 2^3 = Z(6)^*\cdot 2^1 + 2^8Y\cdot 2^1 - 2^8Y,\ i = 6$
$Z(5)^* = Z(5)\cdot 2^3 = Z(6)^*\cdot 2^1 + 2^8Y,\ i = 6$

---

if $Z(5) \geq 0$ *then* $Q_5 = 1,\ Z(4) = Z(5) - 2^4Y,\ i = 5$
else $Q_5 = 0,\ Z(4) = (Z(5) + 2^5Y) - 2^4Y,\ i = 5$

---

then $Q_5 = 1,\ Z(4) = Z(5) - 2^4Y,\ i = 5$
$Z(4)\cdot 2^4 = Z(5)\cdot 2^4 - 2^4Y\cdot 2^4,\ i = 5$
$Z(4)^* = Z(4)\cdot 2^4 = Z(5)\cdot 2^3\cdot 2^1 - 2^8Y,\ i = 5$
$Z(4)^* = Z(4)\cdot 2^4 = Z(5)^*\cdot 2^1 - 2^8Y,\ i = 5$

else $Q_5 = 0$, $Z(4) = (Z(5) + 2^5Y) - 2^4Y$, $i = 5$

$Z(4) \cdot 2^4 = (Z(5) + 2^5Y) \cdot 2^4 - 2^4Y \cdot 2^4$, $i = 5$

$Z(4)^* = Z(4) \cdot 2^4 = (Z(5) \cdot 2^3 + 2^5 \cdot 2^3Y) \cdot 2^1 - 2^8Y$, $i = 5$

$Z(4)^* = Z(4) \cdot 2^4 = Z(5)^* \cdot 2^1 + 2^8Y \cdot 2^1 - 2^8Y$, $i = 5$

$Z(4)^* = Z(4) \cdot 2^4 = Z(5)^* \cdot 2^1 + 2^8Y$, $i = 5$

---

if $Z(4) \geq 0$ *then* $Q_4 = 1$, $Z(3) = Z(4) - 2^3Y$, $i = 4$

else $Q_4 = 0$, $Z(3) = (Z(4) + 2^4Y) - 2^3Y$, $i = 4$

---

then $Q_4 = 1$, $Z(3) = Z(4) - 2^3Y$, $i = 4$

$Z(3) \cdot 2^5 = Z(4) \cdot 2^5 - 2^3Y \cdot 2^5$, $i = 4$

$Z(3)^* = Z(3) \cdot 2^5 = Z(4) \cdot 2^4 \cdot 2^1 - 2^8Y$, $i = 4$

$Z(3)^* = Z(3) \cdot 2^5 = Z(4)^* \cdot 2^1 - 2^8Y$, $i = 4$

else $Q_4 = 0$, $Z(3) = (Z(4) + 2^4Y) - 2^3Y$, $i = 4$

$Z(3) \cdot 2^5 = (Z(4) + 2^4Y) \cdot 2^5 - 2^3Y \cdot 2^5$, $i = 4$

$Z(3)^* = Z(3) \cdot 2^5 = (Z(4) \cdot 2^4 + 2^4 \cdot 2^4Y) \cdot 2^1 - 2^8Y$, $i = 4$

$Z(3)^* = Z(3) \cdot 2^5 = Z(4)^* \cdot 2^1 + 2^8Y \cdot 2^1 - 2^8Y$, $i = 4$

$Z(3)^* = Z(3) \cdot 2^5 = Z(4)^* \cdot 2^1 + 2^8Y$, $i = 4$

---

if $Z(3) \geq 0$ *then* $Q_3 = 1$, $Z(2) = Z(3) - 2^2Y$, $i = 3$

else $Q_3 = 0$, $Z(2) = (Z(3) + 2^3Y) - 2^2Y$, $i = 3$

---

then $Q_3 = 1$, $Z(2) = Z(3) - 2^2Y$, $i = 3$

$Z(2) \cdot 2^6 = Z(3) \cdot 2^6 - 2^2Y \cdot 2^6$, $i = 3$

$Z(2)^* = Z(2) \cdot 2^6 = Z(3) \cdot 2^5 \cdot 2^1 - 2^8Y$, $i = 3$

$Z(2)^* = Z(2) \cdot 2^6 = Z(3)^* \cdot 2^1 - 2^8Y$, $i = 3$

else $Q_3 = 0$, $Z(2) = (Z(3) + 2^3Y) - 2^2Y$, $i = 3$

$Z(2) \cdot 2^6 = (Z(3) + 2^3Y) \cdot 2^6 - 2^2Y \cdot 2^6$, $i = 3$

$Z(2)^* = Z(2) \cdot 2^6 = (Z(3) \cdot 2^5 + 2^3 \cdot 2^5Y) \cdot 2^1 - 2^8Y$, $i = 3$

$Z(2)^* = Z(2) \cdot 2^6 = Z(3)^* \cdot 2^1 + 2^8Y \cdot 2^1 - 2^8Y$, $i = 3$

$Z(2)^* = Z(2) \cdot 2^6 = Z(3)^* \cdot 2^1 + 2^8Y$, $i = 3$

---

if $Z(2) \geq 0$ *then* $Q_2 = 1$, $Z(1) = Z(2) - 2^1Y$, $i = 2$

else $Q_2 = 0$, $Z(1) = (Z(2) + 2^2Y) - 2^1Y$, $i = 2$

---

then $Q_2 = 1$, $Z(1) = Z(2) - 2^1Y$, $i = 2$

$Z(1) \cdot 2^7 = Z(2) \cdot 2^7 - 2^1Y \cdot 2^7$, $i = 2$

$Z(1)^* = Z(1) \cdot 2^7 = Z(2) \cdot 2^6 \cdot 2^1 - 2^8Y$, $i = 2$

$Z(1)^* = Z(1) \cdot 2^7 = Z(2)^* \cdot 2^1 - 2^8Y$, $i = 2$

else $Q_2 = 0$, $Z(1) = (Z(2) + 2^2Y) - 2^1Y$, $i = 2$

$Z(1) \cdot 2^7 = (Z(2) + 2^2Y) \cdot 2^7 - 2^1Y \cdot 2^7$, $i = 2$

$Z(1)^* = Z(1) \cdot 2^7 = (Z(2) \cdot 2^6 + 2^2 2^6Y) \cdot 2^1 - 2^8Y$, $i = 2$

$Z(1)^* = Z(1) \cdot 2^7 = Z(2)^* \cdot 2^1 + 2^8Y \cdot 2^1 - 2^8Y$, $i = 2$

$Z(1)^* = Z(1) \cdot 2^7 = Z(2)^* \cdot 2^1 + 2^8Y$, $i = 2$

---

if $Z(1) \geq 0$ *then* $Q_1 = 1,\ Z(0) = Z(1) - 2^0Y,\ i = 1$

else $Q_1 = 0,\ Z(0) = (Z(1) + 2^1Y) - 2^0Y,\ i = 1$

---

then $Q_1 = 1,\ Z(0) = Z(1) - 2^0Y,\ i = 1$

$Z(0)\cdot 2^8 = Z(1)\cdot 2^8 - 2^0Y\cdot 2^8,\ i = 1$

$Z(0)^* = Z(0)\cdot 2^8 = Z(1)\cdot 2^7\cdot 2^1 - 2^8Y,\ i = 1$

$Z(0)^* = Z(0)\cdot 2^8 = Z(1)^*\cdot 2^1 - 2^8Y,\ i = 1$

else $Q_1 = 0,\ Z(0) = (Z(1) + 2^1Y) - 2^0Y,\ i = 1$

$Z(0)\cdot 2^8 = (Z(1) + 2^1Y)\cdot 2^8 - 2^0Y\cdot 2^8,\ i = 1$

$Z(0)^* = Z(0)\cdot 2^8 = (Z(1)\cdot 2^7 + 2^1 2^7 Y)\cdot 2^1 - 2^8Y,\ i = 1$

$Z(0)^* = Z(0)\cdot 2^8 = Z(1)^*\cdot 2^1 + 2^8Y\cdot 2^1 - 2^8Y,\ i = 1$

$Z(0)^* = Z(0)\cdot 2^8 = Z(1)^*\cdot 2^1 + 2^8Y,\ i = 1$

---

3. *if* $Z(0) \geq 0$ *then* $Q_0 = 1,\ Z = Z(0)$

else $Q_0 = 0,\ Z = Z(0) + 2^0Y$

---

then $Q_0 = 1,\ Z = Z(0)$

$Z\cdot 2^8 = Z(0)\cdot 2^8$

$Z^* = Z\cdot 2^8 = Z(0)\cdot 2^8 = Z(0)^*$

else $Q_0 = 0,\ Z = Z(0) + 2^0Y$

$Z\cdot 2^8 = Z(0)\cdot 2^8 + 2^0Y\cdot 2^8$

$Z^* = Z\cdot 2^8 = Z(0)^* + Y\cdot 2^8$

Posle ovih modifikacija algoritam se može napisati i na sledeći način:

1. $Z(8) = X - 2^8Y$

if $Z(8) \geq 0$ *then* prekoračenje
else
$Z(7)^* = Z(7)\cdot 2^1 = (Z(8) + 2^8Y)\cdot 2^1 - 2^8Y$
$Z(7) = Z(7)\cdot 2^1 = Z(8)\cdot 2^1 + 2^8Y$

---

2. *if* $Z(7)^* \geq 0$
then $Q_7 = 1$,
$Z(6)^* = Z(6)\cdot 2^2 = Z(7)^*\cdot 2^1 - 2^8Y,\ i = 7$
else $Q_7 = 0$,
$Z(6)^* = Z(6)\cdot 2^2 = (Z(7)^* + 2^8Y)\cdot 2^1 - 2^8Y,\ i = 7$
$Z(6)^* = Z(6)\cdot 2^2 = Z(7)^*\cdot 2^1 + 2^8Y,\ i = 7$

---

if $Z(6)^* \geq 0$
then $Q_6 = 1$,
$Z(5)^* = Z(5)\cdot 2^3 = Z(6)^*\cdot 2^1 - 2^8Y,\ i = 6$
else $Q_6 = 0$,
$Z(5)^* = Z(5)\cdot 2^3 = (Z(6)^* + 2^8Y)\cdot 2^1 - 2^8Y,\ i = 6$
$Z(5)^* = Z(5)\cdot 2^3 = Z(6)^*\cdot 2^1 + 2^8Y,\ i = 6$

---

if $Z(5)^* \geq 0$
then $Q_5 = 1$,
$Z(4)^* = Z(4)\cdot 2^4 = Z(5)^*\cdot 2^1 - 2^8Y,\ i = 5$
else $Q_5 = 0$,
$Z(4)^* = Z(4)\cdot 2^4 = (Z(5)^* + 2^8Y)\cdot 2^1 - 2^8Y,\ i = 5$
$Z(4)^* = Z(4)\cdot 2^4 = Z(5)^*\cdot 2^1 + 2^8Y,\ i = 5$

---

if $Z(4)^* \geq 0$
then $Q_4 = 1$,
$Z(3)^* = Z(3)\cdot 2^5 = Z(4)^*\cdot 2^1 - 2^8Y,\ i = 4$
else $Q_4 = 0$,
$Z(3)^* = Z(3)\cdot 2^5 = (Z(4)^* + 2^8Y)\cdot 2^1 - 2^8Y,\ i = 4$
$Z(3)^* = Z(3)\cdot 2^5 = Z(4)^*\cdot 2^1 + 2^8Y,\ i = 4$

---

if $Z(3)^* \geq 0$
then $Q_3 = 1$,
$Z(2)^* = Z(2)\cdot 2^6 = Z(3)^*\cdot 2^1 - 2^8Y,\ i = 3$
else $Q_3 = 0$,
$Z(2)^* = Z(2)\cdot 2^6 = (Z(3)^* + 2^8Y)\cdot 2^1 - 2^8Y,\ i = 3$
$Z(2)^* = Z(2)\cdot 2^6 = Z(3)^*\cdot 2^1 + 2^8Y,\ i = 3$

---

if $Z(2)^* \geq 0$

then $Q_2 = 1$,
$Z(1)^* = Z(1) \cdot 2^7 = Z(2)^* \cdot 2^1 - 2^8 Y, \ i = 2$
else $Q_2 = 0$,
$Z(1)^* = Z(1) \cdot 2^7 = (Z(2)^* + 2^8 Y) \cdot 2^1 - 2^8 Y, \ i = 2$
$Z(1)^* = Z(1) \cdot 2^7 = Z(2)^* \cdot 2^1 + 2^8 Y, \ i = 2$

---

if $Z(1)^* \geq 0$

then $Q_1 = 1$,
$Z(0)^* = Z(0) \cdot 2^8 = Z(1)^* \cdot 2^1 - 2^8 Y, \ i = 1$
else $Q_1 = 0$,
$Z(0)^* = Z(0) \cdot 2^8 = (Z(1)^* + 2^8 Y) \cdot 2^1 - 2^8 Y, \ i = 1$
$Z(0)^* = Z(0) \cdot 2^8 = Z(1)^* \cdot 2^1 + 2^8 Y, \ i = 1$

---

3. *if* $Z(0)^* \geq 0$

then $Q_0 = 1$,
$Z^* = Z \cdot 2^8 = Z(0)^*$
else
$Z^* = Z \cdot 2^8 = Z(0)^* + Y \cdot 2^8$

Ovaj algoritam se može koncizno predstaviti na sledeći način:

1. $Z(8) = X - 2^8Y$

   if $Z(8) \geq 0$ *then* prekoračenje

   else $Z(7)^* = 2^1(Z(8) + 2^8Y) - 2^8Y = 2^1Z(8) + 2^8Y$

   pri čemu je $Z(7)^* = 2^1Z(7)$

2. if $Z(i)^* \geq 0$ *then* $Q_i=1$, $Z(i-1)^* = 2^1Z(i)^* - 2^8Y$, $i = 7, 6, \ldots, 1$

   else $Q_i = 0$, $Z(i-1)^* = 2^1(Z(i)^* + 2^8Y) - 2^8Y = 2^1Z(i)^* + 2^8Y$, $i = 7, 6, \ldots, 1$

   pri čemu je $Z(i)^* = 2^{8-i}Z(i)$

3. if $Z(0)^* \geq 0$ *then* $Q_0=1$, $Z^*=Z(0)^*$

   else $Q_0=0$, $Z^*=Z(0)^* + 2^8Y$

   pri čemu je $Z(0)^*=2^8Z(0)$

2 LITERATURA

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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