



Elektrotehnički fakultet Univerziteta u Beogradu
Katedra za računarsku tehniku i informatiku

DIPLOMSKI RAD

Jedan pristup osnovnim konceptima arhitekture računara

student:
Iva Maksimović 351/02

profesor:
dr. Jovan Đorđević

Beograd
26.09.2008.

Sadržaj

SADRŽAJ	1
1. UVOD	4
2. ARHITEKTURA I ORGANIZACIJA RAČUNARA	6
2.1. Arhitektura računara	6
2.1.1. Skup programski dostupnih registara	6
2.1.2. Tipovi podataka	7
2.1.3. Format instrukcija	7
2.1.4. Načini adresiranja	9
2.1.5. Skup instrukcija	10
2.1.5.1. Opis instrukcija	10
2.1.5.2. Kodiranje instrukcija	14
2.1.6. Mehanizam prekida	17
2.1.7. Memorija	18
2.1.8. Ulazno/Izlazni uređaji	18
2.1.8.1. Registri kontrolera bez direktnog pristupa memoriji	19
2.1.8.2. Registri kontrolera sa direktnim pristupom memoriji	20
2.1.8.3. Registri kontrolera za generisanje impulsa	21
3. SOFTVERSKI PAKET IGoVSoDS	22
3.1. IGoVSoDS - opšte napomene	22
3.2. Korisnički interfejs softverskog paketa IGoVSoDS	22
3.2.1. Glavne komponente programa	22
3.2.2. Opšte funkcije programa	22
3.2.3. Biblioteka dostupnih kola	23
3.3. Funkcije grafičkog editora digitalne strukture	24
3.3.1. Funkcije za rad sa podlogom za prikaz strukture	24
3.3.2. Funkcije za rad sa objektima	24
3.3.3. Funkcije za rad sa signalima	25
3.3.4. Detaljni pregled funkcija	26
3.3.4.1. Portovi i simboli modula (<i>Port Menager...</i>)	26
3.3.4.2. Posebne vrste objekata	27
3.3.4.2.1. Moduli	28
3.3.4.2.2. Generatori signala	28

3.3.4.2.3. Memorijski moduli.....	29
3.3.4.3. Funkcija <i>Expand Connectors</i>	29
3.3.4.4. Pregled tipova signala i transformacija signala.....	29
3.3.4.4.1. Clone/PartClone signali.....	30
3.3.4.4.2. Magistrala.....	30
3.3.4.5. Povezivanje signala i modula.....	31
3.4. Upravljanje simulatorom.....	33
4.Simulator CPU.....	36
4.1. Procesor.....	36
4.1.1. Operaciona jedinica.....	36
4.1.1.1. Blok 'bus'.....	37
4.1.1.2. Blok 'addr'.....	38
4.1.1.3. Blok 'Instrukcije'.....	38
4.1.1.4. Blok 'Operacije'.....	40
4.1.1.5. Blok 'Prekid'.....	44
4.1.2. Upravljačka jedinica.....	47
4.1.2.1. Algoritam generisanja upravljačkih signala.....	47
4.1.2.2. Struktura upravljačke jedinice.....	47
4.2. Memorija.....	49
4.2.1. Operaciona jedinica.....	49
4.2.2. Upravljačka jedinica.....	50
5. Simulator U/I.....	51
5.1. Glavna šema.....	51
5.2. Ulazno/Izlazni uređaji.....	51
5.3. Periferija.....	52
5.3.1. Operaciona jedinica.....	52
5.3.2. Upravljačka jedinica.....	53
5.4. Kontroler periferije bez direktnog pristupa memoriji.....	53
5.4.1. Operaciona jedinica.....	54
5.4.1.1. Blok 'Registri'.....	54
5.4.1.2. Blok 'Interfejs'.....	56
5.4.2. Upravljačka jedinica.....	57
5.4.2.1. Upravljačka jedinica 'Uprav_bus'.....	57
5.4.2.2. Upravljačka jedinica 'Uprav_per'.....	58
5.5. Kontroler periferije sa direktnim pristupom memoriji.....	59

5.5.1. Operaciona jedinica.....	59
5.5.1.1. Blok 'Registri'.....	60
5.5.1.2. Blok 'Interfejs'.....	61
5.5.2. Upravljačka jedinica.....	62
5.5.2.1. Upravljačka jedinica 'Uprav_bus'.....	62
5.5.2.2. Upravljačka jedinica 'Uprav_per'.....	63
5.5.2.3. Upravljačka jedinica 'Uprav_mem'.....	63
5.6. Kontroler periferije za generisanje impulsa.....	64
5.6.1. Operaciona jedinica.....	65
5.6.2. Upravljačka jedinica.....	66
6. LABORATORIJA.....	67
6.1. Primeri instrukcija i načina adresiranja.....	67
6.1.1. Instrukcija ADD.....	67
6.1.1.1. Inicijalizacija programa.....	67
6.1.1.2. Izvršavanje programa.....	68
6.1.2. Instrukcije JSR i RTS.....	73
6.1.2.1. Inicijalizacija programa.....	73
6.1.2.2. Izvršavanje programa.....	74
6.2. Primeri mehanizma prekida.....	77
6.2.1. Prekid usled greške u načinu adresiranja.....	77
6.2.1.1. Inicijalizacija programa.....	77
6.2.1.2. Izvršavanje programa.....	78
6.3. Primeri programiranog ulaza i izlaza.....	83
6.3.1. Ulaz sa periferije tehnikom ispitivanja bita spremnosti.....	83
6.3.1.1. Inicijalizacija programa.....	84
6.3.1.2. Izvršavanje programa.....	85
7. ZAKLJUČAK.....	90
8. LITERATURA.....	91
9. PRILOG.....	92

1. UVOD

Kada bi birali jednu stvar za koju bi trebalo da kažemo da današnji svet bez nje ne može, definitivno se nameću računari. Današnjica je u znaku računara. Svugde oko nas i u svakom segmentu života okruženi smo računarima koji nam pomažu i olakšavaju živote ljudi. Oni se nalaze u automobilima, avionima, aparatima za domaćinstvo, medicinskom instrumentima, mobilnim telefonima, idustrijskim uređajima i telekomunikacionim sistemima. Mi ih koristimo svakodnevno i više, jednostavno, nismo toga ni svesni. Oni danas omogućavaju da se svet okreće. Zato je proučavanje i izrada računara danas veliko i široko područje. Međutim, svi ti računari imaju neku osnovnu zajedničku strukturu koju treba prvo dobro shvatiti i razumeti osnovne mehanizme njenog funkcionisanja, da bi se dalje ti mehanizmi koristili i nadograđivali u neke posebne svrhe, kojih je danas mnogo.

Iz tog razloga se na Elektrotehničkom fakultetu u Beogradu izučava arhitektura i organizacija računara. To se radi kroz teorijska razmatranja nekih osnovnih principa, mehanizama i gradivnih blokova današnjih računara. Ovo teorijsko znanje ipak ne prikazuje stvarnost računara dovoljno plastično. Stoga se u izučavanju koriste ogledni procesori i njihovi simulatori radi što realističnijeg i jasnijeg za razumevanje pristupa ovom znanju. Na Elektrotehničkom fakultetu u Beogradu već su izrađivani mnogi simulatori koji svojom eksplicitnošću pomažu studentima u razumevanju rada računarskih sistema i njihovih komponenata.

Ovaj diplomski rad je imao dva osnovna cilja. Prvi cilj rada je bio izrada ulazno-izlaznog podsistema jednog postojećeg računarskog sistema koji je već izrađen u okviru [9] i čija je arhitektura unapred data u [1]. U okviru ulazno-izlaznog podsistema, u softverskom paketu IgoVSoDS, čije su mogućnosti detaljno opisane u [2], izrađeni su simulatori kontrolera periferije bez direktnog pristupa memoriji, kontrolera periferije sa direktnim pristupom memoriji, kontrolera za generisanje impulsa i periferije. Takođe računarski sistem iz [9] je proširen sa tri na sedam ulaza za prekide da bi omogućio adekvatno povezivanje svih izrađenih ulazno-izlaznih uređaja. Drugi cilj rada je bio izrada asemblerskih programa koji se mogu izvršavati na već pomenutom računarskom sistemu i koji će ilustrovati neke osnovne koncepte arhitekture računara. Ti koncepti su izvršavanje instrukcija procesora, različiti načini adresiranja memorije, vektorisan mehanizam prekida i programirani ulaz i izlaz iz periferije u procesor i obrnuto i to upotrebom kontrolera bez direktnog pristupa memoriji korišćenjem tehnika ispitivanja bita spremnosti i generisanja prekida i upotrebom kontrolera sa direktnim pristupom memoriji. Instrukcije programa su kodirane u saglasnosti sa pravilima ovog procesora za laboratorijske vežbe i time prilagođene izvršavanju na njemu. Uz progame postoje i komentari i kratki opisi ideja koje iza njih stoje koji će takođe pomoći u lakšem razumevanju. Radi interakcije sa ovim sistemom i ukazivanja na važne činjenice data su i sugestivna pitanja povezana sa svakim programom. Ona ukazuju na značajne trenutke u izvršavanju programa i važne vrednosti i rezultate funkcionisanja koji se dobijaju u različitim delovima ovog sistema. Pitanja omogućavaju da se kroz kreativan rad jednostavno i zanimljivo studenti u laboratorijskim vežbama praktično upoznaju sa teorijskim znanjem koje stiču na predavanjima iz predmeta arhitekture i organizacije računara.

U sledećem odeljku uvoda dat je kratak sadržaj svih sledećih poglavlja rada da bi se čitaoci upoznali sa onim što sledi u narednom tekstu.

U drugoj glavi dat je kratak pregled arhitekture i organizacije računarskog sistema za laboratorijske vežbe datog u celosti u [1]. Elementi na koje je obraćena pažnja su: skup programski dostupnih registara, tipovi podataka, format instrukcija, načini adresiranja, skup instrukcija, opis instrukcija, kodiranje instrukcija, mehanizam prekida, poseban opis arhitekture memorije i ulazno/Izlaznih uređaja.

U trećoj glavi dat je kratak opis mogućnosti i načina upotrebe korišćenog softverskog paketa IGoVSoDS [2]. Pažnja je obraćena na najkorisnije mogućnosti sistema i ostale koje su korišćene u izradi, povezivanju i proveru ispravnosti simulatora sistema.

U četvrtoj glavi dat je detaljan pregled svih blokova procesora i memorije računarskog sistema, kao i opis njihovih funkcionalnosti u okviru sistema. Ovaj deo sistema nije realizovan u okviru rada nego je preuzet uz male izmene iz [9].

U petoj glavi dat je detaljan opis svih delova ulazno-izlaznih uređaja čiji su simulatori izrađeni, zajedno sa objašnjenjima njihove svrhe funkcionisanja kao celine. Simulatori su izrađeni takođe prema unapred datoj arhitekturi u [1].

U šestoj glavi dat je prikaz laboratorijskih vežbi koje su za simulator napravljene. One prolaze kroz najvažnije raspoložive instrukcije računarskog sistema sa svim mogućim načinima adresiranja, kroz sve vidove komunikacije procesorskog podsistema sa ulazno/izlaznim podsistemom i mehanizme prekida i njegove mogućnosti. Posebno je dat pregled tri odabrana primera.

U sedmoj glavi dat je zaključak kao kratak komentar svega o čemu je bilo reći prethodno kao i niz zapažanja vezanih za korišćeni softverski paket.

U prilogu su dati kompletni napravljeni asemblerski programi, zajedno sa pojašnjenjima i pitanjima.

2. ARHITEKTURA I ORGANIZACIJA RAČUNARA

U ovoj glavi je dat pregled na koji način je organizovan računar čiji simulator se razmatra. Delovi teksta i slike su preuzeti iz literature [1] uz saglasnost autora. Dat je detaljan pregled arhitekture računara kao i u kratkim crtama njegova organizacija. Detaljna organizacija računara će biti opisana u poglavljima koji slede.

2.1. Arhitektura računara

Arhitekturu procesora čine: skup programski dostupnih registara, tipovi podataka, format instrukcija, načini adresiranja, skup instrukcija i mehanizam prekida.

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

2.1.1. Skup programski dostupnih registara

Registri procesora koji su programski dostupni su: programski brojač PC, akumulator AB, akumulator AW, 32 registra opšte namene GPR, ukazivač na vrh steka SP, programska statusna reč PSW, registar maske IMR i ukazivač na tabelu adresa prekidnih rutina IVTP.

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

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

Registar AW je 16-to razredni akumulator za operacije prenosa 8-mo bitnih veličina u kojima se koristi kao implicitno izvoriste ili odredište operanda.

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

Registar SP je standardan 16-to razredni ukazivač na vrh steka. Stek je organizovan u operativnoj memoriji i raste prema višim adresama. Registar SP ukazuje na poslednju zauzetu lokaciju na steku. Adrese generisane na osnovu vrednosti registra SP su adrese 8-mo bitnih reči. Njime se mogu generisati adrese unutar adresnog prostora od 2^{16} 8-mo bitnih reči.

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

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

Struktura registra PSW

Bitovi registra PSW su: N—bit koji se postavlja na 1 u slučaju da je rezultat operacije negativan, Z—bit koji se postavlja na 1 u slučaju da je rezultat operacije jednak 0, C—bit koji se postavlja na 1 u slučaju prenosa/pozajmice u aritmetici celobrojnih veličina bez znaka, V—bit koji se postavlja na 1 u slučaju prekoračenja u aritmetici celobrojnih veličina sa znakom, L₂, L₁, L₀—bitovi kojima se pamti nivo prioriteta tekućeg programa, T—bit koji je jednak 1 ako procesor treba da posle svake instrukcije generiše prekid i I—bit koji je jednak 1 ako treba da budu dozvoljeni maskirajući prekidi.

Bitovi N, Z, C i V se postavljaju hardverski na osnovu rezultata izvršavanja instrukcija. Bitovi L₂, L₁ i L₀ se postavljaju hardverski u okviru opsluživanja prekida i softverski kao rezultat izvršavanja instrukcije povratak iz prekidne rutine. Bitovi I i T se postavljaju softverski kao rezultat izvršavanja posebnih instrukcija, hardverski u okviru opsluživanja prekida i softverski kao rezultat izvršavanja instrukcije povratak iz prekidne rutine.

Registar IMR je standardni 16-to razredni registar maske za selektivno maskiranje maskirajućih prekida. Koriste se samo razredi 3, 2 i 1 koji su dodeljeni prekidima koji dolaze po linijama 3, 2 i 1 od ulazno/izlaznih uređaja, respektivno.

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

2.1.2. Tipovi podataka

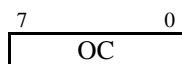
Tipovi podataka koji se koriste u ovom procesoru su celobrojne 8-mo bitne veličine sa znakom i bez znaka, 8-mo bitne binarne reči i 16-to bitne binarne reči. Celobrojne 8-mo bitne veličine sa znakom i bez znaka se koriste kod operacije prenosa, aritmetičkih operacija i operacija pomeranja i rotiranja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

2.1.3. Format instrukcija

U procesoru se koriste tri formata instrukcija: format bezadresnih instrukcija, format instrukcija relativnog skoka i format jednoadresnih instrukcija.

Format bezadresnih instrukcija

Format ovih instrukcija je dat na slici 1.

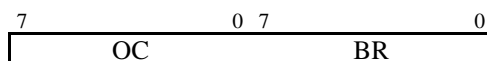


Slika 1: Format bezadresnih instrukcija

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

Format instrukcije prekida

Format ove instrukcije je dat na slici 2.

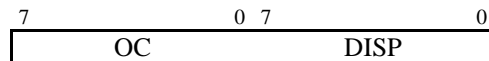


Slika 2: Format instrukcija prekida

Poljem OC se specificira operacija prekida, a poljem BR broj ulaza u tabelu sa adresama prekidnih rutina. Broj ulaza je 8-mo bitna celobrojna veličina bez znaka.

Format instrukcija relativnog skoka

Format ovih instrukcija je dat na slici 3.



Slika 3: Format instrukcija relativnog skoka

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

Format instrukcija apsolutnog skoka

Format ovih instrukcija je dat na slici 4.

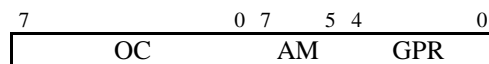


Slika 4: Format instrukcija relativnog skoka

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

Format jednoadresnih registarskih instrukcija

Format ovih instrukcija je dat na slici 5.

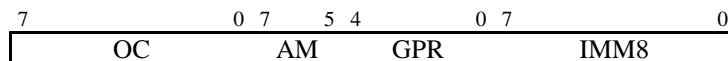


Slika 5: Format jednoadresnih registarskih instrukcija

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

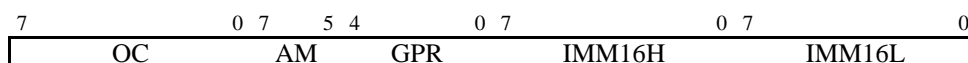
Format jednoadresnih instrukcija

Postoji više vrsta formata ovih instrukcija i oni su dati na slikama od 6 do 8.



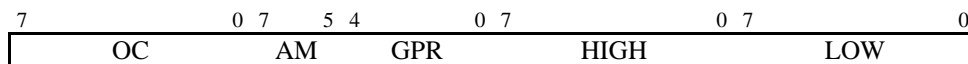
Slika 6: Format jednoadresnih instrukcija - **IB format**

Slika 6 - Poljem OC se specificira kod operacije jednoadresne instrukcije nad 8-mo bitnim veličinama, polje AM neposredno adresiranje, polje GPR se ne koristi i poljem IMM8 neposredna 8-mo bitna veličina.



Slika 7: Format jednoadresnih instrukcija - **IW format**

Slika 7 - Poljem OC se specificira kod operacije jednoadresne instrukcije nad 16-to bitnim veličinama, polje AM neposredno adresiranje, polje GPR se ne koristi i poljima IMM16H i IMM16L 8 starijih i 8 mlađih bitova neposredne 16-to bitne veličine.



Slika 8: Format jednoadresnih instrukcija - **AP format**

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

2.1.4. Načini adresiranja

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

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

Registarsko direktno adresiranje je adresiranje kod koga se operand nalazi u jednom od registara opšte namene. Registar opšte namene je specificiran poljem GPR. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 16-to bitnim veličinama ili 8-mo bitnim veličinama koristi se svih 16 ili 8 nižih razreda registra.

Registarsko indirektno adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi određenoj sadržajem jednog od registara opšte namene. Polje GPR specificira registar opšte namene. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

Memorijsko direktno adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi datoj u samoj instrukciji. Polje GPR se ne koristi. Polja HIGH i LOW sadrže starijih 8 i mladih 8 bitova adrese. U zavisnosti od toga da li se poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

Indirektno memorijsko adresiranje je adresiranje kod koga se operand nalazi u memoriji na adresi određenoj sadržajem memorijske lokacije čija je adresa data u samoj instrukciji. Polje GPR se ne koristi. Polja HIGH i LOW sadrže starijih 8 i mladih 8 bitova adrese sa koje se čita adresa operanda. U zavisnosti od toga da li se

poljem OC instrukcije specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama iz memorije se čita jedan ili dva bajta.

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

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

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

Neposredno adresiranje je adresiranje kod koga se operand nalazi u samoj instrukciji. Poljem OC se specificira operacija nad 8-mo bitnim veličinama ili 16-to bitnim veličinama. Polje GPR se ne koristi. U slučaju da je operacija nad 8-mo bitnim veličinama polje IMM8 predstavlja 8-mo bitni podatak, a u slučaju 16-to bitnih veličina polja IMM16H i IMM16L sadrže starijih 8 i mlađih 8 bitova 16-to bitnog podatka.

2.1.5. Skup instrukcija

U ovom delu se daje opis instrukcija i pregled kodiranja instrukcija.

2.1.5.1. Opis instrukcija

Instrukcije ovog procesora se mogu svrstati u sledećih deset grupa: instrukcija bez dejstva, instrukcija zaustavljanja, instrukcije skoka, instrukcije prenosa, aritmetičke instrukcije, logičke instrukcije, instrukcije pomeranja i rotiranja, instrukcije postavljanja indikatora u PSW.

Instrukcija bez dejstva

Instrukcija **NOP** ne proizvodi nikakvo dejstvo. Ona ima format bezadresnih instrukcija. Ova instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija zaustavljanja

Instrukcija **HALT** zaustavlja izvršavanje instrukcija. Ova instrukcija ne utiče na indikatore N, Z, C i V registra PSW i ona takođe ima format bezadresnih instrukcija.

Instrukcije skoka

Instrukcije skoka se svrstavaju u sledeće grupe: instrukcije uslovnog skoka, instrukcije bezuslovnog skoka, instrukcije skoka na potprogram i povratka iz potprograma i instrukcija prekida i povratka iz prekidne rutine

Instrukcije uslovnog skoka

Instrukcije uslovnog skoka **BEQL disp**, **BNEQL disp**, **BNEG disp**, **BNNEG disp**, **BOVF disp**, **BNOVF disp**, **BCAR disp**, **BNCAR disp**, **BGRT disp**, **BGRTE disp**, **BLSS disp**, **BLSSE disp**, **BGRTU disp**, **BGRTEU disp**, **BLSSU disp** i **BLSSEU disp** realizuju relativni skok sa pomerajem *disp* u odnosu na programski brojač PC ukoliko je uslov specificiran kodom operacije ispunjen (tabela 1). Pomeraj *disp* je 8-mo bitna celobrojna veličina sa znakom. Ni jedna od ovih instrukcija ne utiče na indikatore N, Z, C i V registra PSW, a njihov format je format instrukcija relativnog skoka.

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

Tabela 1: Instrukcije uslovnog skoka

Instrukcija bezuslovnog skoka

Instrukcija bezuslovnog skoka **JMP a** realizuje skok na adresu *a* koja je data u samoj instrukciji. Ona ima format instrukcija apsolutnog skoka. Ova instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcije skoka na potprogram i povratka iz potprograma

Instrukcija **JSR a** realizuje skok na potprogram tako što čuva vrednost programskog brojača PC na steku i realizuje skok na adresu *a* koja je data u samoj instrukciji. Ova instrukcija ima format instrukcija apsolutnog skoka.

Instrukcija **RTS** realizuje povratak iz potprograma tako što restaurira vrednost programskog brojača PC vrednošću sa steka. Ima format bezadresnih instrukcija.

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

Instrukcije prekida i povratka iz prekidne rutine

Instrukcija **INT** *br* programskim putem realizuje prekid i skok na prekidnu rutinu čija se adresa nalazi u ulazu *br* tabele sa adresama prekidnih rutina.

Instrukcija **RTI** realizuje povratak iz prekidne rutine tako što restaurira vrednosti programske statusne reči PSW i programskog brojača PC vrednostima sa steka. Ima format bezadresnih instrukcija.

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

Instrukcije prenosa

Instrukcija **LDB** *src* prenosi sadržaj 8-bitnog operanda *src* u akumulator AB.

Instrukcija **LDW** *src* prenosi sadržaj 16-bitnog operanda *src* u akumulator AW.

Instrukcija **STB** *dst* prenosi sadržaj akumulatora AB u 8-bitni operand *dst*.

Instrukcija **STW** *dst* prenosi sadržaj akumulatora AW u 16-bitni operand *dst*.

Instrukcija **POPB** prenosi sadržaj 8-bitnog operanda sa vrha steka u akumulatora AB.

Instrukcija **POPW** prenosi sadržaj 16-bitnog operanda sa vrha steka u akumulatora AW.

Instrukcija **PUSHB** prenosi sadržaj akumulatora AB u 8-bitni operand na vrh steka. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **PUSHW** prenosi sadržaj akumulatora AW 16-bitni operand na vrh steka. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **LDIVTP** prenosi sadržaj registra IVTP u akumulator AW. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **STIVTP** prenosi sadržaj akumulatora AW u registar IVTP. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **LDIMR** prenosi sadržaj registra IMR u akumulator AW. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **STIMR** prenosi sadržaj akumulatora AW u registar IMR. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **LOADSP** prenosi sadržaj registra SP u akumulator AW. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Instrukcija **STORESP** prenosi sadržaj akumulatora AW u registar SP. Instrukcija ne utiče na indikatore N, Z, C i V registra PSW.

Poslednjih deset opisanih instrukcija imaju format bezadresnih instrukcija.

Aritmetičke instrukcije

Instrukcija **ADD** *src* sabira veličinu koja se nalazi u akumulatoru AB sa operandom *src*, a rezultat smešta u akumulator AB.

Instrukcija **SUB** *src* oduzima operand *src* od veličine koja se nalazi u akumulatoru AB, a rezultat smešta u akumulator AB.

Instrukcija **INC** inkrementira veličinu koja se nalazi u akumulatoru AB i rezultat smešta u akumulator AB.

Instrukcija **DEC** dekrementira veličinu koja se nalazi u akumulatoru AB i rezultat smešta u akumulator AB.

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

Logičke instrukcije

Instrukcija **AND** *src* izračunava logičko I veličine koja se nalazi u akumulatoru AB i operanda *src*, a rezultat smešta u akumulator AB.

Instrukcija **OR** *src* izračunava logičko ILI veličine koja se nalazi u akumulatoru AB i operanda *src*, a rezultat smešta u akumulator AB.

Instrukcija **XOR** *src* izračunava logičko EKSLUZIVNO ILI veličine koja se nalazi u akumulatoru AB i operanda, a rezultat smešta u akumulator AB.

Instrukcija **NOT** izračunava logičku NEGACIJU veličine koja se nalazi u akumulatoru AB, a rezultat smešta u akumulator AB.

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

Instrukcije pomeranja i rotiranja

Instrukcija **ASR** pomera sadržaj akumulatora AB udesno za jedno mesto, a rezultat smešta u akumulator AB. Pri pomeranju razred AB₇ ostaje nepromenjen, a u indikator C registra PSW se upisuje sadržaj razreda AB₀.

Instrukcija **LSR** pomera sadržaj akumulatora AB udesno za jedno mesto, a rezultat smešta u akumulator AB. Pri pomeranju u razred AB₇ se upisuje 0, a u indikator C registra PSW se upisuje sadržaj razreda AB₀.

Instrukcija **ROR** pomera sadržaj akumulatora AB udesno za jedno mesto, a rezultat smešta u akumulator AB. Pri pomeranju u razred AB₇ se upisuje sadržaj razreda AB₀, a u indikator C registra PSW se upisuje sadržaj razreda AB₀.

Instrukcija **RORC** pomera sadržaj akumulatora AB udesno za jedno mesto, a rezultat smešta u akumulator AB. Pri pomeranju u razred AB₇ se upisuje sadržaj indikatora C registra PSW, a u indikator C registra PSW se upisuje sadržaj razreda AB₀.

Instrukcija **ASL** pomera sadržaj akumulatora AB ulevo za jedno mesto, a rezultat smešta u akumulator AB. Pri pomeranju u razred AB₀ se upisuje 0, a u indikator C registra PSW se upisuje sadržaj razreda AB₇.

Instrukcija **LSL** pomera sadržaj akumulatora AB ulevo za jedno mesto, a rezultat smešta u akumulator AB. Pri pomeranju u razred AB₀ se upisuje 0, a u indikator C registra PSW se upisuje sadržaj razreda AB₇.

Instrukcija **ROL** pomera sadržaj akumulatora AB ulevo za jedno mesto, a rezultat smešta u akumulator AB. Pri pomeranju u razred AB₀ se upisuje sadržaj razreda AB₇, a u indikator C registra PSW se upisuje sadržaj razreda AB₇.

Instrukcija **ROLC** pomera akumulatora AB ulevo za jedno mesto, a rezultat smešta u akumulator AB. Pri pomeranju u razred AB₀ se upisuje sadržaj indikatora C registra PSW, a u indikator C registra PSW se upisuje sadržaj razreda AB₇.

Instrukcije postavljaju indikatore N i Z registra PSW saglasno vrednosti koja se upisuje u akumulator AB, dok se indikator V registra PSW postavlja na 0. Sve instrukcije pomeranja i rotiranja imaju format bezadresnih instrukcija.

Instrukcije postavljanja indikatora u PSW

Instrukcija **INTD** postavlja nulu u razred I registra PSW.

Instrukcija **INTE** postavlja jedinicu u razred I registra PSW.

Instrukcija **TRPD** postavlja nulu u razred T registra PSW.

Instrukcija **TRPE** postavlja jedinicu u razred T registra PSW.

Ni jedna od ovih instrukcija ne utiče na indikatore N, Z, C i V registra PSW. Sve instrukcije postavljanja indikatora u PSW imaju format bezadresnih instrukcija.

2.1.5.2. Kodiranje instrukcija

Kodiranje instrukcija je izvršeno na takav način da bitovi: OC_{7...6} predstavljaju četiri grupe od 64 instrukcije, OC_{5...3} predstavljaju osam podgrupa od 8 instrukcija i OC_{2...0} predstavljaju instrukciju u okviru podgrupe instrukcija.

Kodiranje četiri grupe instrukcija G0 do G3 je dato u tabeli 2:

OC _{7...6}	Oznaka	Napomena
00	G0	Koristi se
01	G1	Ne koristi se
10	G2	Ne koristi se
11	G3	Ne koristi se

Tabela 2: Kodiranje četiri grupe instrukcija

Kodiranje osam podgrupa instrukcija G0_PG0 do G0_PG7 u okviru grupe instrukcija G0 je dato u tabeli 3.

OC _{5...3}	Oznaka	Napomena
000	G0_PG0	Instrukcija bez dejstva, instrukcija zaustavljanja i instrukcije postavljanja indikatora u PSW
001	G0_PG1	Instrukcija bezuslovnog skoka, instrukcije skoka na potprogram i povratka iz potprograma, instrukcija prekida i povratka iz prekidne rutine
010	G0_PG2	Instrukcije uslovnog skoka
011	G0_PG3	Instrukcije uslovnog skoka

100	G0_PG4	Instrukcije prenosa
101	G0_PG5	Instrukcije prenosa
110	G0_PG6	Aritmetičke instrukcije i logičke instrukcije
111	G0_PG7	Instrukcije pomeranja i rotiranja

Tabela 3: Kodiranje osam podgrupa instrukcija iz grupe G0

Kodiranje instrukcija u okviru osam podgrupa instrukcija G0_PG0 do G0_PG7 je dato u tabelama 4 do 11.

Kodiranje podgrupe instrukcija G0_PG0 koju čine instrukcija bez dejstva, instrukcija zaustavljanja i instrukcije postavljanja indikatora u PSW je dato u tabeli 4.

OC _{2...0}	Oznaka	dužina u bajtovima
000	NOP	1
001	HALT	1
010	-	-
011	-	-
100	INTD	1
101	INTE	1
110	TRPD	1
111	TRPE	1

Tabela 4: Kodiranje podgrupe instrukcija G0_PG0

Kodiranje podgrupe instrukcija G0_PG1 koju čine instrukcija bezuslovnog skoka, instrukcije skoka na potprogram i povratka iz potprograma i instrukcije prekida i povratka iz prekidne rutine je dato u tabeli 5.

OC _{2...0}	mnemonik	dužina u bajtovima
000	-	-
001	JMP	3
010	JSR	3
011	RTS	1
100	INT	1
101	RTI	1
110	-	-
111	-	-

Tabela 5: Kodiranje podgrupa instrukcija G0_PG1

Kodiranje podgrupe instrukcija G0_PG2 koju čine instrukcije uslovnog skoka je dato u tabeli 6.

OC _{2...0}	mnemonik	dužina u bajtovima
000	BEQL	2
001	BNEQL	2
010	BNEG	2
011	BNNEG	2
100	BOVF	2
101	BNOVF	2
110	BCAR	2

111	BNCAR	2
-----	--------------	---

Tabela 6: Kodiranje podgrupa instrukcija G0_PG2

Kodiranje podgrupe instrukcija G0_PG3 koju čine instrukcije uslovnog skoka je dato u tabeli 7.

OC _{2...0}	mnemonik	dužina u bajtovima
000	BGRT	2
001	BGRTE	2
010	BLSS	2
011	BLSSE	2
100	BGRTU	2
101	BGRTEU	2
110	BLSSU	2
111	BLSSEU	2

Tabela 7: Kodiranje podgrupa instrukcija G0_PG3

Kodiranje podgrupe instrukcija G0_PG4 koju čine instrukcije prenosa je dato u tabeli 8.

OC _{2...0}	mnemonik	dužina u bajtovima
000	LDB	2, 3 ili 4
001	LDW	2 ili 4
010	STB	2 ili 4
011	STW	2 ili 4
100	POPB	1
101	POPW	1
110	PUSHB	1
111	PUSHW	1

Tabela 8: Kodiranje podgrupa instrukcija G0_PG4

Kodiranje podgrupe instrukcija G0_PG5 koju čine instrukcije prenosa je dato u tabeli 9.

OC _{2...0}	mnemonik	dužina u bajtovima
000	LDIVTP	1
001	STIVTP	1
010	LDIMR	1
011	STIMR	1
100	LDSP	1
101	STSP	1
110	-	-
111	-	-

Tabela 9: Kodiranje podgrupa instrukcija G0_PG5

Kodiranje podgrupe instrukcija G0_PG6 koju čine aritmetičke instrukcije (odeljak 0) i logičke instrukcije je dato u tabeli 10.

OC _{2...0}	mnemonik	dužina u bajtovima
000	ADD	2, 3 ili 4
001	SUB	2, 3 ili 4
010	INC	1
011	DEC	1
100	AND	1
101	OR	1
110	XOR	1
111	NOT	1

Tabela 10: Kodiranje podgrupa instrukcija G0_PG6

Kodiranje podgrupe instrukcija G0_PG7 koju čine instrukcije pomeranja i rotiranja je dato u tabeli 11.

OC _{2...0}	mnemonik	dužina u bajtovima
000	ASR	1
001	LSR	1
010	ROR	1
011	RORC	1
100	ASL	1
101	LSL	1
110	ROL	1
111	ROLC	1

Tabela 11: Kodiranje podgrupa instrukcija G0_PG7

2.1.6. Mehanizam prekida

Zahteve za prekid mogu da generišu:

- 1) sedam kontrolera periferija po linijama intr₇ do intr₁ da bi signalizirali spremnost za prenos podataka (maskirajući prekidi),
- 2) jedan uređaj računara koji kontroliše ispravnost napona napajanja (nemaskirajući prekid),
- 3) procesor, kao rezultat otkrivene nekorektnosti u izvršavanju tekuće instrukcije (nelegalan kod operacije i nelegalno adresiranje),
- 4) procesor, ako je zadat takav režim rada procesora, kroz postavljanje indikatora T u programskoj statusnoj reči PSW, da se posle svake instrukcije skače na određenu prekidnu rutinu i
- 5) procesor, kao rezultat izvršavanja instrukcije prekida INT.

Prekidi 1 i 2 su spoljašnji prekidi, a prekidi 3, 4 i 5 su unutrašnji. Opsluživanje zahteva za prekid se sastoji iz tri grupe koraka.

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

U okviru druge grupe koraka utvrđuje se adresa prekidne rutine. Utvrđivanje adrese prekidne rutine se realizuje na osnovu sadržaja tabele adresa prekidnih rutina (IV tabela) i broja ulaza u IV tabelu. Stoga je u postupku inicijalizacije celog sistema u memoriji, počev od adrese na koju ukazuje sadržaj registra procesora IVTP

(*Interrupt Vector Table Pointer*), kreirana IV tabela sa 256 ulaza u kojima se nalaze adrese prekidnih rutina za sve vrste prekida. Broj ulaza u IV tabelu se dobija na više načina i to:

- predstavlja fiksnu vrednost za prekide iz tačke 1 i generiše ga sam procesor,
- predstavlja fiksnu vrednost za prekide iz tačaka 2, 3, 4 i generiše ga sam procesor i
- specificiran je adresnim delom instrukcije INT za prekid iz tačke 5.

Ulazi 0 do 3 i 9 do 15 u IV tabeli su rezervisani za adrese prekidnih rutina za sledeće vrste prekida:

0 – prekid zbog režima rada sa prekidom posle svake instrukcije,

1 – nemaskirajući prekid,

2 – prekid zbog greške u adresiranju,

3 – prekid zbog greške u kodu operacije,

9 do 15 – maskirajući prekidi po linijama intr_1 do intr_7 , respektivno.

Ulazi 4 do 8 i 16 do 255 u IV tabeli su slobodni za adrese prekidnih rutina za prekide izazvane instrukcijom INT.

Više prekida se može javiti istovremeno, a može se prihvatiti samo jedan zahtev za prekid i skočiti na njegovu prekidnu rutinu. Zato su zahtevima za prekid dodeljeni prioriteti, pa se od generisanih zahteva za prekid prihvata onaj zahtev za prekid koji je među njima najvišeg prioriteta. Redosled zahteva za prekid po opadajućim prioritetima je sledeći: najviši je 5, zatim redom 3, 2, 1 i na kraju najniži je 4.

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

U okviru treće grupe koraka se: brišu indikatori I i T registra PSW kod prekida svih vrsta i upisuje u bitove L_2 do L_0 registra PSW nivo prioriteta prekidne rutine na koju se skače u slučaju maskirajućeg prekida.

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

Kada procesor izvršava prekidnu rutinu može stići novi zahtev za prekid. Na ovaj zahtev za prekid može se reagovati na sledeće načine: prekida se izvršavanje tekuće prekidne rutine i skače na novu prekidnu rutinu ili ne prekida se izvršavanje prekidne rutine, već se zahtev za prekid prihvata tek po završetku prekidne rutine i povratku u prekinuti program.

2.1.7. Memorijska

Arhitekturu memorije čine 8-mo bitne programski dostupne memorijske lokacije kapaciteta 60K reči, koje zauzimaju opseg od 0 do EFFFh memorijskog adresnog prostora.

2.1.8. Ulazno/Izlazni uređaji

Arhitekturu ulazno/izlaznih uređaja čine 8-mo bitni programski dostupni registri kontrolera ulazno/izlaznih uređaja kapaciteta 4K reči, koji zauzimaju opseg od F000h do FFFFh memorijskog adresnog prostora. Iz registara kontrolera se čita i u registre kontrolera se upisuje na identičan način na koji se čita iz memorijskih lokacija i upisuje u memorijske lokacije. Dozvoljeno je čitanje iz svih registara kontrolera i upisivanje u sve registre kontrolera. Adresa registra kontrolera se dobija na identičan način na koji se dobija adresa memorijske lokacije.

Adresiranje pojedinih registara kontrolera se obezbeđuje preko tri komponente koje formiraju adrese registara (slika 9): adresiranje ulazno-izlaznog adresnog prostora (UIP), adresiranje uređaja u ulazno-izlaznom adresnom prostoru (UIU) i adresiranje registara unutar adresiranog uređaja (UIUR).

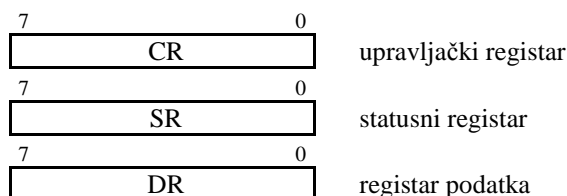


Slika 9 Formiranje adrese registara kontrolera periferije

Usvojeno je da je za ulazno-izlazni adresni prostor rezervisano najviših 2^{12} adresa te je sadržaj polja UIP jednak 1111b. Ova vrednost polja UIP je ista za sve ulazno-izlazne uređaje i njihove registre. Polje UIU je širine 6 bita, što omogućava adresiranje do 2^6 ulazno-izlaznih uređaja. Ova vrednost se postavlja mikroprekidačima pri povezivanju ulazno-izlaznog uređaja na računarski sistem. Poljem UIUR se adresira do 2^6 registara unutar ulazno-izlaznog uređaja. Ovakav format adresiranja registara ulazno-izlaznih uređaja je usvojen za sve uređaje.

2.1.8.1. Registri kontrolera bez direktnog pristupa memoriji

Programski dostupni registri kontrolera su CR, SR i DR (slika 10).



Slika 1 Programski dostupni registri kontrolera bez direktnog pristupa memoriji

Registar CR je upravljački registar širine 8 bita od kojih se koriste samo najmlađa 3 bita. Ovaj registar se koristi da se programskim putem upisivanjem odgovarajućih vrednosti kontroler startuje i zadaje režima rada i da se kontroler zaustavlja. Bitovi registra CR su:

- CR_0 koji vrednošću: 0 ukazuje da je zadat režim rada sa izlaznom periferijom i 1 ukazuje da je zadat režim rada sa ulaznom periferijom,
- CR_1 koji vrednošću: 0 ukazuje da zadat režim rada bez generisanjem prekida i 1 ukazuje da zadat režim rada sa generisanjem prekida i
- CR_2 koji vrednošću: 0 ukazuje da je kontroler zaustavljen i 1 ukazuje da je kontroler startovan.

Registar SR je statusni registar širine 8 bita od kojih se koristi samo najmlađi bit. Ovaj registar se koristi da se programskim putem čitanjem njegovog sadržaja dobijaju informacije o stanju u kome se nalazi kontroler. Bit statusnog registra je:

- SR_0 koji vrednošću: 0 ukazuje da registar DR nije raspoloživ i 1 ukazuje da je registar DR raspoloživ.

Registar DR je registar podatka širine 8 bita. Ovaj registar se koristi kao prihvatni registar za podatke koje razmenjuju kontroler i procesor. Slanje podatka u izlazni uređaj se realizuje programskim putem izvršavanjem instrukcija koje upisuju podatak u registar DR. Unos podatka iz ulaznog uređaja se realizuje programskim putem izvršavanjem instrukcija koje čitaju podatak iz registra DR.

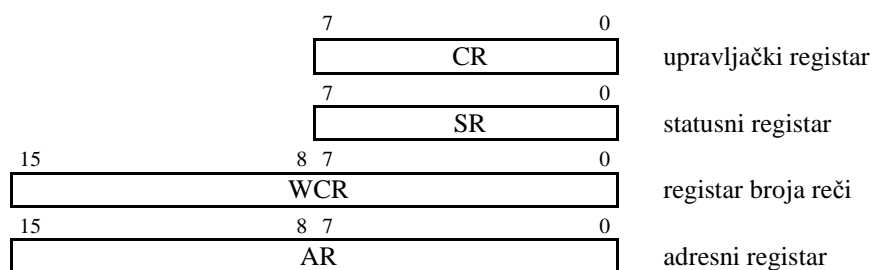
U tabeli 2 su date relativne adrese svih programski dostupnih registara kontrolera bez direktnog pristupa memoriji u okviru opsega od 64 adrese datog kontrolera.

Tabela 2 Relativne adrese registara kontrolera bez direktnog pristupa memoriji

Registar	Adresa
CR	0
SR	1
DR	2

2.1.8.2. Registri kontrolera sa direktnim pristupom memoriji

Programski dostupni registri kontrolera su CR, SR, WCR i AR (slika 2).



Slika 2 Programski dostupni registri kontrolera sa direktnim pristupom memoriji

Registar CR je upravljački registar širine 8 bita od kojih se koriste samo najmlađa 4 bita. Ovaj registar se koristi da se programskim putem upisivanjem odgovarajućih vrednosti kontroler startuje i zadaje režima rada i da se kontroler zaustavlja. Bitovi registra CR su:

- CR_0 koji vrednošću: 0 ukazuje da je zadat režim prenosa memorija – izlazna periferija i 1 ukazuje da je zadat režim prenosa ulazna periferija - memorija,
- CR_1 koji vrednošću: 0 ukazuje da zadat režim rada bez generisanjem prekida i 1 ukazuje da zadat režim rada sa generisanjem prekida,
- CR_2 koji vrednošću: 0 ukazuje da je kontroler zaustavljen i 1 ukazuje da je kontroler startovan i
- CR_3 koji vrednošću: 0 ukazuje da je zadat režim rada sa pojedinačnim prenosom i 1 ukazuje da je zadat režim rada sa paketskim prenosom.

Registar SR je statusni registar širine 8 bita od kojih se koriste samo najmlađi bit SR_0 . Ovaj registar se koristi da se programskim putem čitanjem njegovog sadržaja dobijaju informacije o stanju u kome se nalazi kontroler. Bit statusnog registra je:

- SR_0 koji vrednošću: 0 ukazuje da prenos podataka nije obavljen do kraja i 1 ukazuje da je prenos podataka obavljen do kraja.

Registar WCR je registar broja reči koje treba preneti širine 16 bita.

Registar AR je adresni registar širine 16 bita koji se koristi izvorišni adresni registar za adresiranje izvorišnog podatka u memoriji u režimu prenosa memorija–izlazni periferija i odredišni adresni registar za adresiranje odredišnog podatka u memoriji u režimu prenosa ulazna periferija – memorija.

U tabeli 3 su date relativne adrese svih programski dostupnih registara kontrolera sa direktnim pristupom memoriji u okviru opsega od 64 adrese datog kontrolera.

Tabela 3 Relativne adrese registara kontrolera sa direktnim pristupom memoriji

Registar	Adresa
CR	0
SR	1
WCR _{15...8}	4
WCR _{7...0}	5
AR _{15...8}	6
AR _{7...0}	7

2.1.8.3. Registri kontrolera za generisanje impulsa

Programski dostupni registri kontrolera su CR i WCR (slika 3).



Slika 3 Programski dostupni registri kontrolera za generisanje impulsa

Registar CR je upravljački registar širine 8 bita od kojih se koristi samo bit CR₂. Ovaj registar se koristi da se programskim putem upisivanjem odgovarajućih vrednosti kontroler startuje i da se kontroler zaustavlja. Bit registra CR je:

- CR₂ koji vrednošću: 0 ukazuje da je kontroler zaustavljen i 1 ukazuje da je kontroler startovan.

Registar WCR je 16-to razredni registar čekanja u kome se čuva vreme, zadato kao broj perioda signala takta, koje treba da protekne od trenutka startovanja kontrolera do trenutka generisanja signala prekida trajanja jedna perioda signala takta.

U tabeli 4 su date relativne adrese svih programski dostupnih registara kontrolera sa direktnim pristupom memoriji u okviru opsega od 64 adrese datog kontrolera.

Tabela 4 Relativne adrese registara kontrolera kontrolera za generisanje impulsa

Registar	Adresa
CR	0
WCR _{15...8}	2
WCR _{7...0}	3

3. SOFTVERSKI PAKET IGoVSoDS

U ovoj glavi dat je opis softverskog alata *IGoVSoDS* koji je korišćen za realizaciju ulazno-izlaznih jedinica, kao i celog opisanog računarskog sistema. U sledećim poglavljima dat je opis najkorisnijih funkcionalnosti softverskog alata kako za izradu simulatora tako i za pregled simulacija.

3.1. *IGoVSoDS* - opšte napomene

IGoVSoDS (**Interactive Generator of Visual Simulators of Digital Structures**) je program koji pruža korisnički interfejs za kreiranje i modifikaciju digitalnih struktura i omogućava detaljno podešavanje značajnih parametara svih digitalnih kola i modula, i obezbeđuje jednostavno upravljanje simulacijom i efikasan sistem pregleda rezultata rada simulacije projektovane digitalne strukture. To je alat koji je razvijen na Elektrotehničkom fakultetu u Beogradu sa ciljem da omogući jednostavan način generisanja svih vrsta simulatora koji će se koristiti za rad u laboratoriji na predmetima arhitekture i organizacije računara. Autor ovog programa je **mr Nenad M. Grbanović**, a deo teksta je preuzet iz literature [3] uz saglasnost autora.

3.2. Korisnički interfejs softverskog paketa *IGoVSoDS*

U ovom poglavlju će biti dat najosnovniji pregled svega onog sto pruža ovaj softverski paket, a u narednim poglavljima detaljno su prikazane mogućnosti svih delova osnovnog prozora softverskog paketa, kao i svih postupaka za upravljanje radom softverskog paketa.

3.2.1. Glavne komponente programa

Glavne komponente softverskog paketa su:

1) Osnovni prozor softverskog paketa, čiji su sastavni delovi: Podloga za prikaz digitalne strukture, Glavni meni i grafički skup poziva funkcije, Biblioteke kola i hijerarhijsko stablo modula, Navigacija i pregled rezultata rada simulatora, Pregled poruka i sadržaja strukture podataka, Sistem za pomoć pri korišćenju softverskog paketa

2) Funkcije grafičkog editora digitalne strukture

3) Rad sa parametrima simulacije i upravljanje simulatorom

3.2.2. Osnovne funkcije programa

Osnovne funkcije koje program pruža su:

- Dodavanje, pomeranje i brisanje standardnih sekvencijalnih i kombinacionih kola, kao i specijalnih kola i modula;
- Dodavanje, pomeranje i brisanje signala i linija signala;
- Dodavanje, pomeranje i brisanje natpisa opšte namene (*General Purpose Labels*), natpisa naziva signala (*Signal Name Labels*) i natpisa logičkih stanja na

linijama signala koji su širi od jednog bita (*Signal State Labels*);

- Poravnavanje kola u odnosu na odabrano referentno kolo po levoj, desnoj, gornjoj ili donjoj ivici, kao i poništavanje poslednjeg poravnavanja (funkcija *Alignment*);
- Promenu veličine i izgleda *Port Module*-a i *NonPort Module*-a;
- Jednostavnu zamenu *Port Module*-a i *NonPort Module*-a (funkcije *Replace With ALL Connections...* i *Replace Without ANY Connection...*).

Navedene funkcije se detaljno opisuju u poglavljima koji slede.

3.2.3. Biblioteka dostupnih kola

U programu je dostupna biblioteka realizovanih logičkih elemenata - kombinacionih i sekvencijalnih kola uz pomoć kojih je moguće konstruisati složenije elemente - module koji se kasnije mogu koristiti u konstruisanju još složenijih struktura. Na raspolaganju su sledeći tipovi kola i modula:

- Standardna kombinaciona kola, sa označenim brojem ulaznih i izlaznih konektora:
K-NOT, K-AND_i, K-NAND_i, K-OR_i, K-NOR_i, K-ExOR2, K-ExNOR2,
K-DC_{k-m}, K-CDp_{k-m}, K-MP_{k-m}, K-DP_{k-m},
K-SHIFT, K-INC-DEC, K-ADD, K-CG4, K-CMP i K-ALU.

Nakon izbora iz biblioteke, na ulazne konektore ovih kola mogu da se povežu signali širine 1 bit, dok su izlazni signali širine 1 bit. Primenom određene funkcije *Expand Connectors...* moguće je promeniti širinu ovih konektora. O ovoj funkciji će biti više reči u nastavku.

- Standardna kombinaciona kola, sa modifikovanim ulaznim konektorima: K-gAND, K-gNAND, K-gOR, K-gNOR, K-gDC_{k-m}, K-gCDp_{k-m}, K-gMP_{k-m} i K-gDP_{k-m}. Sva kola iz ove grupe imaju po jedan ulazni konektor na koji može da se poveže signal sa širinom od 1 do 32 bita, dok je izlazni signal uvek širine 1 bit. Nakon izbora iz biblioteke, ulazni konektori ovih kola prihvataju signale širine 16 bita. Primenom funkcije *Expand Connectors...* menjaju se širine samo ulaznih konektora.
- Standardna sekvencijalna kola: S-DFF, S-RSFF, S-TFF, S-JKFF, S-REG, S-ShiftREG, S-COUNTERinc i S-COUNTER. Nakon izbora flip-flopova iz biblioteke kola, na ulazne konektore mogu da se povežu signali širine 1 bit, dok su izlazni signali širine 1 bit. Primenom funkcije *Expand Connectors...* menjaju se širine svih konektora. Nakon izbora registara i brojača iz biblioteke kola, ulazni i izlazni signali za podatke su širine 16 bita, dok su svi kontrolni ulazni i izlazni signali širine 1 bit. Primenom funkcije *Expand Connectors...* menjaju se širine samo ulaznih i izlaznih signala za podatke.
- Specijalni moduli:
 - Sp-SubModule – *NonPort Module*.
 - Sp-TriState – trostatički bafer koji omogućava povezivanje signala na magistralu. Isključivo izlazni signal ovog modula može da se poveže na magistralu. Nakon izbora iz biblioteke svi ulazni i izlazni signali su širine

1 bit. Primenom funkcije *Expand Connectors...* menjaju se širine ulaznog i izlaznog signala, dok kontrolni signal zadržava širinu 1 bit.

- Sp–Generator – generator signala.
- Sp–Memory – memorijski modul.

3.3. Funkcije grafickog editora digitalne strukture

U ovom poglavlju je dat detaljan uvid u sve mogućnosti koji pruža ovaj program. Dat je skup svih relevantnih funkcija uz pomoć kojih se konstruiše simulator kao i opis šta te funkcije rade. Biće takođe, prikazani postupci za rad sa objektima koji predstavljaju standardna kola, sa objektima koji predstavljaju module kao i sa signalima i linijama signala.

3.3.1. Funkcije za rad sa podlogom za prikaz strukture

Ova grupa funkcija je dostupna korisniku kada se pokazivač misa nalazi na slobodnoj površini podloge za prikaz strukture i kada se aktivira desni taster misa. Obezbeđene su funkcije koje vrše:

- 1) Prelazak na prikaz hijerarhijski nadređenog modula;
- 2) Prelazak na prikaz hijerarhijski najvišeg modula, odnosno glavnog modula;
- 3) Rad sa natpisima opšte namene: kreiranje natpisa (*Create*), modifikacija teksta natpisa (*Edit*), kreiranje natpisa koji je identičan natpisu iznad koga je otvoren padajući meni (*Create Copy*), smanjenje i povećanje veličine slova natpisa (*Font Size Down* i *Font Size Up*, respektivno), modifikacija izgleda slova natpisa (*Bold*, *Italic* i *Underline*) i brisanje natpisa (*Delete*);
- 4) Rad sa natpisima za prikaz naziva signala: kreiranje natpisa (*New...*), modifikacija teksta natpisa (*Edit*), poravnavanje natpisa po vertikali i horizontali sa linijom signala (*Align to Line*), centriranje natpisa sa sredinom linije signala (*Align to Center*) i brisanje natpisa (*Delete...*);
- 5) Rad sa natpisima za prikaz stanja signala, za signale veće širine od jednog bita: kreiranje natpisa (*New...*), aktiviranje binarnog i heksadecimalnog formata natpisa (*Binary* i *Hexadecimal*, respektivno), poravnavanje natpisa po vertikali i horizontali sa linijom signala (*Align to Line*), centriranje natpisa sa sredinom linije signala (*Align to Center*) i brisanje natpisa (*Delete...*);
- 6) Otvaranje dijaloga za kreiranje, modifikaciju i brisanje podataka o portovima trenutno aktivne strukture. Ovako definisani portovi koriste se samo kada se trenutno aktivna digitalna struktura integriše u neku drugu digitalnu strukturu, aktiviranjem funkcije *Add Module/SubModule with Ports...*;
- 7) Otvaranje dijaloga za modifikaciju i podesavanje karakteristike modula.

3.3.2. Funkcije za rad sa objektima

Ova grupa funkcija je dostupna korisniku kada se pokazivač misa postavi na neko kolo ili modul i aktivira desni taster miša. Obezbeđene su:

- 1) Skup funkcija za rad sa modulima:
 - *Edit Symbol Title* – otvaranje polja za modifikaciju teksta natpisa na simbolu koji predstavlja modul;
 - *Replace With ALL Connections...* – zamena modula trenutno aktivnog projekta drugim modulom, SA očuvanjem konekcija modula koji se zamenjuje, na modul koji se učitava;
 - *Replace Without ANY Connection...* – zamena modula trenutno aktivnog projekta drugim modulom, BEZ očuvanja konekcija modula koji se zamenjuje, na modul koji se učitava;
- 2) Skup funkcija za rad sa generatorom signala (generator signala je element koji proizvodi signal koji ima vrednost koja je zadata funkcijom):
 - *Function...* – otvaranje dijaloga za definisanje parametara funkcije kojom se generiše signal na izlazu generatora signala;
 - *Switch Value* – funkcija koja za generator signala čiji izlaz ima samo jedan bit prebacuje vrednost iz vrednosti logičke nule u vrednost logičke jedinice, i obrnuto. Funkcija nije dostupna za generator signala čiji je izlazni signal veće širine od jednog bita;
 - *None, Unknown i TriState* – parametri koji određuje da li je na izlazu generatora vrednost signala koja je saglasna rezultatu rada funkcije generatora signala (*None*), ili je nametnuta jedna od specijalnih vrednosti: nepoznata vrednost (*Unknown*) ili visoka impedansa (*TriState*);
- 3) Skup funkcija za rad sa memorijskim modulom (*Memory Module*): otvaranje dijaloga za pregled sadržaja memorijskih lokacija (*HEX Dump List...*), otvaranje dijaloga za pregled i modifikaciju sadržaja memorijskih lokacija (*Change Data...*), kao i dijalozi za učitavanje i snimanje sadržaja memorijskih lokacija iz, odnosno u datoteku (*Import Memory Data...* i *Export Memory Data...*, respektivno). Funkcija memorijskog modula će biti opisana u nastavku;
- 4) Funkcija za modifikaciju broja bitova na konektorima kola i modula (*Expand Connectors...*). Funkcija nije dostupna za *Port Module* i *NonPort Module* objekte. Detaljni opis mogućnosti funkcije *Expand Connectors...* se daje u nastavku;
- 5) Funkcija za dodavanje (*CREATE*) i brisanje (*REMOVE*) kružića koji se integriše sa konektorom kola i modula, kojim se ostvaruje negacija signala na konektoru (*Connector Not Sign*). Funkcija nije dostupna za specijalni modul Sp-Generator, kao i za *Port Module* i *NonPort Module* objekte.

3.3.3. Funkcije za rad sa signalima

Ova grupa funkcija je dostupna korisniku kada se pokazivač misa postavi na neku liniju signala i aktivira desni taster misa. Obezbeđeni su:

- 1) Funkcija za transformaciju signala u magistralu (*Make New BUS*). Funkcija je dostupna samo za signale koji nemaju izvor vrednosti;

- 2) Funkcija koja vrši promenu broja bitova magistrale (*Expand BUS...*). Funkcija je dostupna samo za signale koji su transformisani u magistrale, kada nema signala koji su povezani na magistralu;
- 3) Funkcije koje vrse brisanje određene linije signala (ako se signal sastoji iz više linija) i brisanje celog signala (svih linija signala);
- 4) Funkcija koja na pravoj liniji signala kreira prelomnu tačku (*New Break Point*), nakon čega je moguće da se napravi složenija putanja linije signala. Prelomna tačka koja se kreira nalazi se na mestu gde je otvoren meni (aktiviran traster misa);
- 5) Funkcije za brisanje poslednje linije signala na signalu koji se sastoji iz više linija, kao i dodavanje jedne linije signala na kraj linija koji čine jedan signal;
- 6) Funkcija za otvaranje dijaloga za podešavanje osnovnih parametara vezanih za signal nad kojim je funkcija aktivirana (*Properties...*).

3.3.4. Detaljni pregled funkcija

U poglavljima koja slede daje se detaljni opis najznacajnijih karakteristika i funkcija koje postoje u ovom programu. Karakteristike koje će biti opisane su:

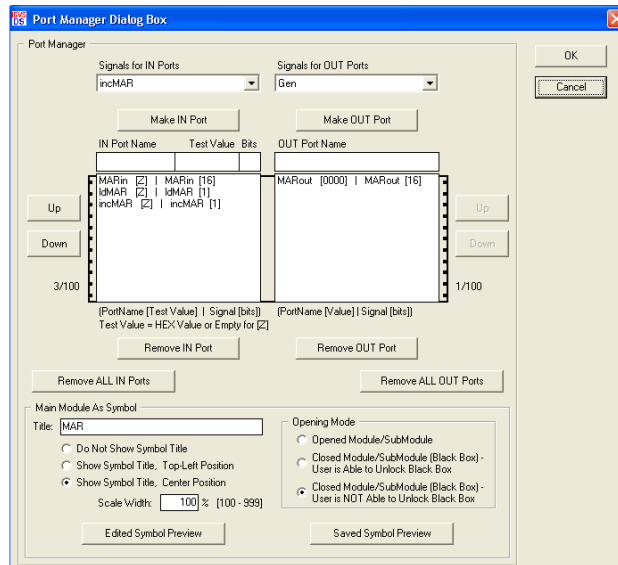
- Portovi i simboli modula (*Port Menager...*);
- Posebne vrste objekata: moduli, generatori signala, memorijski moduli;
- Funkcija *Expand Connectors...* ;
- Osobine signala i kreiranje *Clone/PartClone* signala;

3.3.4.1. Portovi i simboli modula (*Port Menager...*)

Jedna od prednosti ovog programa je mogućnost kreiranja modula na osnovu trenutno aktivne digitalne strukture. Tako kreiran modul može da se integriše u neku drugu digitalnu strukturu. Na slici 14 prikazan je dijalog za kreiranje i modifikaciju podataka o portovima. U tom dijalogu se nalazi lista kandidata za ulazne i izlazne portove (*Signal for IN ports* i *Signal for OUT ports*). Aktiviranjem tastera *Make IN Port* i *Make OUT Port* signal koji je trenutno odabran u odgovarajućoj listi kandidata promovira se u ulazni (*IN*), odnosno izlazni (*OUT*) port, respektivno. Signali koji se odrede za ulazne i izlazne portove mogu da se obrisu čime se uklanjaju iz liste portova. U ovom dijalogu postoji mogućnost za podešavanje izgleda modula. Obezbeđena je modifikacija natpisa na simbolu, mesta gde će biti napisan naziv, procentualne vrednosti za deformaciju veličine simbola, kao i parametra koji određuje da li će korisnik moći da uđe i pregleda strukturu modula, ili to neće biti dopušteno (*Opening Mode*). Omogućen je izbor tri vrednosti parametra *Opening Mode*:

- *Opened Module/SubModule* – dozvoljeno je da se uđe i pregleda struktura modula
- *Closed Module/SubModule (Black Box), User is Able to Unlock Black Box* – nije dozvoljeno da se uđe i pregleda struktura modula, ali korisnik može da promeni dozvolu za ulazak u modul
- *Closed Module/SubModule (Black Box), User is NOT Able to Unlock*

Black Box – nije dozvoljeno da se ude i pregleda struktura modula



Slika 13: Dijalog za kreiranje i modifikaciju podataka o portovima

Ukoliko su liste ulaznih i izlaznih portova prazne, modul će se koristiti kao *NonPort Module*. Prema dijalogu sa slike 13 kreiran je *Port Module*, čiji je izgled prikazan na slici 14.



Slika 14: Jedan mogući prikaz *Port Module*-a

Postupak integracije modula u drugu elektronsku digitalnu strukturu obavlja se aktiviranjem funkcija *Add Module/SubModule with Ports...* i *Add Module/SubModule w/o Ports...* Ukoliko je aktivirana funkcija *Add Module/SubModule with Ports...* modul će biti integrisan kao simbol sa portovima samo ako su portovi definisani, u suprotnom, modul će biti integrisan kao simbol bez portova. Ukoliko je aktivirana funkcija *Add Module/SubModule w/o Ports...* modul će biti integrisan kao simbol bez portova, bez obzira da li su portovi definisani ili nisu definisani.

Pored definisanja portova koji se koriste za povezivanje signala i modula, moguće je i povezivanje signala i modula bez definisanih portova. Moduli na koje su povezani signali mogu da se zamene drugim modulima tako da se ostvarene konekcije sačuvaju ili da se raskinu. Zamena modula sa portovima (*Port Module*) ostvaruje se korišćenjem informacija o portovima, ali takve informacije ne postoje za module bez portova (*NonPort Module*). Za potrebe ostvarivanja zamene modula bez portova, sa mogućnošću da se očuvaju uspostavljene konekcije, obezbeđena je mogućnost numeracije signala u modulu.

3.3.4.2. Posebne vrste objekata

3.3.4.2.1. Moduli

Jedna od važnijih funkcija koji ovaj program pruža je i zamena odabranog modula na digitalnoj strukturi drugim modulom (funkcije *Replace With ALL Connections...* i *Replace Without ANY Connection...*).

Osnovna namena funkcija za zamenu modula je obezbeđivanje mogućnosti da se u okviru aktivne digitalne strukture proizvoljni modul zameni nekim drugim modulom uz očuvanje ostvarenih konekcija signala van modula i signala unutar modula, ili bez očuvanja navedenih konekcija. Ponašanje funkcija delimično se razlikuje za module sa portovima (*Port Module*) i module bez portova (*NonPort Module*).

U zavisnosti od tipa funkcije i postojanja portova modula, razlikuju se četiri slučaja zamene module.

- zamena modula sa očuvanjem konekcija (*Replace With ALL Connections...*), za module sa portovima (*Port Module*) i za module bez portova (*NonPort Module*):
- za module sa portovima (*Port Module*): proverava se da li novi modul ima dovoljno portova, tako da sve postojeće konekcije signala sa starim modulom mogu da se ostvare sa novim modulom. Saglasnost portova se ne utvrđuje po nazivima portova, već po poziciji portova u nizu ulaznih, odnosno izlaznih portova, posebno za svaki tip portova.
- za module bez portova (*NonPort Module*): proverava se da li stari i novi modul imaju numerisane signale, tako da sve postojeće konekcije signala sa starim modulom mogu da se ostvare sa novim modulom. Novi modul može da ima proizvoljan broj signala, ali svi numerisani signali starog modula na koje su povezani signali van modula, moraju da imaju odgovarajući signal u novom modulu, sa identičnom numeracijom.

Modul koji je nasledio položaj starog modula u hijerarhijskoj strukturi modula, integrisan je u aktivnu digitalnu strukturu na identičan način kao i modul koji je zamenjen.

- zamena modula bez očuvanja konekcija (*Replace Without ANY Connection...*), za module sa portovima (*Port Module*) i za module bez portova (*NonPort Module*):

Modul koji je nasledio položaj starog modula u hijerarhijskoj strukturi modula, integrisan je u aktivnu digitalnu strukturu na identičan način kao i modul koji je zamenjen, osim konekcija koje ne postoje.

3.3.4.2.2. Generatori signala

U ranijem delu teksta je već napomenuto, u kraćim crtama, šta je generator signala. Generator signala je realizovan kao modul koji ima samo jedan izlazni signal, gde širina signala može da se podešava. Izlazni signal menja vrednost u svakom ciklusu signala takta pridruženog modulu kome generator pripada. Na raspolaganju su tri funkcije koje upravljaju radom generatora signala:

- *Constant* – izlazni signal ne menja vrednost

- *Counter* – izlazni signal menja vrednost – povećanje (*Increment*), odnosno smanjivanje (*Decrement*) vrednosti izlaznog signala za vrednost polja *Increment Value*, odnosno *Decrement Value*, respektivno
- *Oscillator* – izlazni signal menja vrednost tako što se svi bitovi izlaznog signala invertuju

3.3.4.2.3. Memorijski moduli

Memorijski modul je realizovan kao modul sa standardnim kontrolnim ulazima, ulazima za adresu i podatak, kao i izlazom za podatak koji je pročitao. Ulazne linije za adrese i podatke imaju po 16 bita. Širine ulaznih i izlaznih linija za podatke mogu da se podešavaju sa dozvoljenim vrednostima broja linija na ulaznim i izlaznim linijama za podatke 8, 4, 2 i 1.

Skup funkcija za rad sa sadržajem memorijskih modula (*Object/Memory Module*), obezbeđuje funkcije:

- *HEX Dump List...* – funkcija za otvaranje dijaloga za pregled sadržaja memorijskih lokacija memorijskog modula. Taster *Change* transformiše *HEX Dump List...* dijalog u *Change Data...* dijalog.
- *Change Data...* – funkcija za otvaranje dijaloga za pregled i modifikaciju sadržaja memorijskih lokacija memorijskog modula.

Postoji mogućnost da se promeni sadržaj samo jedne ili svih memorijskih lokacija.

3.3.4.3. Funkcija *Expand Connectors...*

Funkcija *Expand Connectors...* omogućava promenu širine signala na konektorima od 1 do 32 bita, u koracima po 1 bit. Funkcija može da se koristi proizvoljan broj puta za svako kolo koje nema povezane signale na svojim konektorima, bez obzira da li su postojali povezani signali koji su obrisani.

Postoje dve grupe kola i modula na kojima ova funkcija drugačije vrši modifikaciju širina signala. Jednoj grupi kola funkcija *Expand Connectors...* menja sve širine konektora, dok kod druge grupe menja širinu samo nekih konektora. Time se postiže velika raznovrsnost kola koji su dostupni iz osnovne biblioteke.

3.3.4.4. Pregled tipova signala i transformacija signala

U zavisnosti od načina kreiranja signala, korišćenja funkcija za povezivanje kola, modula i signala, kao i upotrebe funkcija za transformaciju tipova signala, signali mogu da pripadaju jednom od sledećih tipova signala:

- *Undefined* – signal koji nema izvor vrednosti (konektor kola ili signal iz modula)
- *Allocated* – signal koji ima izvor vrednosti (konektor kola ili signal iz modula)
- *BUS* – magistrala

- *External/Internal BUS* – poseban tip magistrale koja se koristi pri povezivanju magistrala koje se nalaze na različitim modulima i podmodulima
- *Clone* – signal čija vrednost je identična vrednosti nekog drugog signala (*Master Signal*)
- *PartClone* – signal čiji pojedini bitovi dobijaju vrednosti identične vrednostima bitova drugih signala (*Master Signal*)

Kao mogući izvori vrednosti za *Undefined* i *Allocated* signale označeni su signali iz modula.

Undefined signali imaju najmanje jedan slobodan kraj na podlozi za prikaz strukture, dok drugi kraj može biti slobodan ili povezan na ulazni konektor kola ili ulazni port modula. *Allocated* signali imaju jedan kraj povezan na izlazni konektor kola ili izlazni port modula. *BUS* signal ne može kao izvor vrednosti da ima izlazni konektor proizvoljnog kola ili izlazni port modula, već kao izvor vrednosti može da ima isključivo izlazni signal *Sp-TriState* kola. *External/Internal BUS* signali se koriste isključivo pri povezivanju magistrala u jedinstvenu magistralu koja se prostire na više modula i podmodula. *Clone* i *PartClone* signali kao izvore vrednosti imaju sve bitove jednog signala (*Clone*) ili pojedine bitove koji ne moraju da pripadaju samo jednom signalu (*PartClone*). Signali čiji bitovi postaju komponente *Clone* ili *PartClone* signala imaju status *Master Signal*, ali ovakav status ne prevodi signal u posebnu vrstu signala.

Pored osnovnih tipova signala, na raspolaganju je i poseban skup tipova signala za potrebe povezivanja signala i *Port Module* objekata. Na raspolaganju su *IN Port* i *OUT Port* signali, za realizaciju ulaznih i izlaznih portova *Port Module* objekata, respektivno. *IN Port* signal može da se kreira korišćenjem *Undefined* signala, dok *OUT Port* može da se kreira korišćenjem *Allocated*, *Clone* i *PartClone* signala.

3.3.4.4.1. Clone/PartClone signali

Verovatno najznačajnija funkcija softverskog paketa *IGoVSoDS* je upravo mogućnost dodele vrednosti signalima na osnovu vrednosti drugih signala. Realizacijom ove funkcije, obezbeđena je mogućnost kreiranja signala sastavljenih od proizvoljnih delova svih vidljivih signala. Ukoliko zavisni signal ima samo jednu komponentu u vidu svih linija jednog signala, dobija status *Clone Signal*, dok u svim ostalim slučajevima dobija status *PartClone Signal*. Signali čije se linije koriste kao komponente u kreiranju *Clone* ili *PartClone* signala dobijaju status *Master Signal*. Signali u statusu *Clone* i *PartClone* mogu da budu *Master* signali za druge *Clone* i *PartClone* signale.

Ova funkcija praktično omogućava da jedan signal iz jednog modula, čija je širina recimo 32 bita, ima vrednost sastavljenu od 32 različitih signala širine jednog bita koji se nalaze u različitim modulima.

3.3.4.4.2. Magistrala

Magistrala (*BUS Signal*) je posebna vrsta signala koja se koristi za povezivanje više drugih signala. Za magistrale se primenjuju posebna pravila za

kreiranje, promenu broja bitova, povezivanje drugih signala, kao i povezivanje magistrale sa kolima i modulima.

Za ostvarivanje funkcije povezivanja signala na magistralu, u biblioteci kola je obezbeđen specijalni modul *Sp-TriState*. Isključivo izlazni signal specijalnog modula *Sp-TriState* može da se poveže na magistralu.

Magistrala može da se povezuje na ulazne konektore kola i ulazne portove *Port Module* objekata, kao i svaki drugi signal. Posebne osobine koje ima magistrala, u odnosu na ostale signale, ne prenose se na signal koje se nalazi iza ulaznog porta. Isto tako, magistrala može da se povezuje sa signalima unutar *NonPort Module* objekta, i to na dva načina. Prvi način je da se magistrala poveže kao i svaki drugi signal, čime se posebne osobine magistrale ne prenose na signal unutar *NonPort Module* objekta (*Internal NonBUS signal*). Drugi način je da se magistrala poveže sa magistralom unutar *NonPort Module* objekta (*Internal BUS signal*), čime se dve magistrale integrišu u jedinstvenu magistralu koja se prostire na više modula i podmodula.

3.3.4.5. Povezivanje signala i modula

Povezivanje signala i modula je jedna od najznačajnijih funkcija softverskog paketa *IGoVSoDS*. Ovom funkcijom omogućava se razvoj hijerarhijske organizacije modula aktivne digitalne strukture. Realizovane su dve vrste modula: moduli sa portovima (*Port Module*) i moduli bez portova (*NonPort Module*).

Port Module objekti imaju definisane ulazne i izlazne portove, za povezivanje ulaznih i izlaznih signala, respektivno. Nijedan signal iz *Port Module* objekta nije vidljiv za nadređene module. Isto tako, nijedan signal van *Port Module* objekta nije vidljiv unutar tog objekta. Stoga, povezivanje signala i *Port Module* objekta može da se ostvari jedino povezivanjem signala na ulazne i izlazne portove.

NonPort Module objekti nemaju definisane portove, i nije moguće da se signali i moduli povezuju preko unapred određenih tačaka za povezivanje. Stoga je dopušteno da se signal van modula poveže sa bilo kojim signalom unutar modula, uz poštovanje pravila za povezivanje odgovarajućih tipova signala. Svaki signal van modula koji ima izvor vrednosti (*Allocated*, *Clone* i *PartClone*), kao i svaka magistrala van modula (*BUS*), može da se poveže sa svakim signalom unutar modula, koji nema izvor vrednosti (*Undefined*). Isto tako, svaka magistrala van modula (*BUS*) može da se poveže sa svakom magistralom unutar modula (*BUS*), i na taj način može da se kreira magistrala koja se prostire na više modula. Osnovni uslov za povezivanje je da signali koji se povezuju imaju jednak broj bitova. Signali *Port Module* objekata, podređenih *NonPort Module* objektu sa kojim treba povezati signal, ne mogu da se koriste za povezivanje, saglasno pravilima za vidljivost signala koji pripadaju *Port Module* objektima.

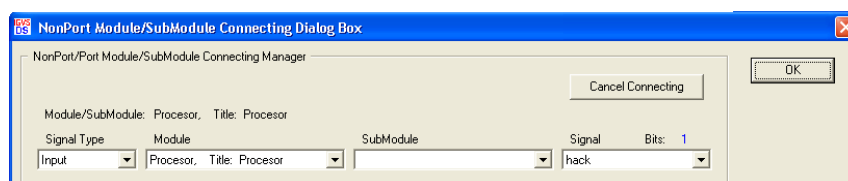
Povezivanje signala i modula je realizovano kroz dva postupka: 1)povezivanje signala i modula sa portovima (*Port Module*) i 2)povezivanje signala i modula bez portova (*NonPort Module*)

Kreiranje linija signala koje će povezivati ulazne i izlazne portove *Port Module* objekta, obavlja se na identičan način kao i za konektore kola i specijalnih

modula. Jedino ograničenje u odnosu na konektore je da nije moguće kreirati kružić za negaciju signala na portu.

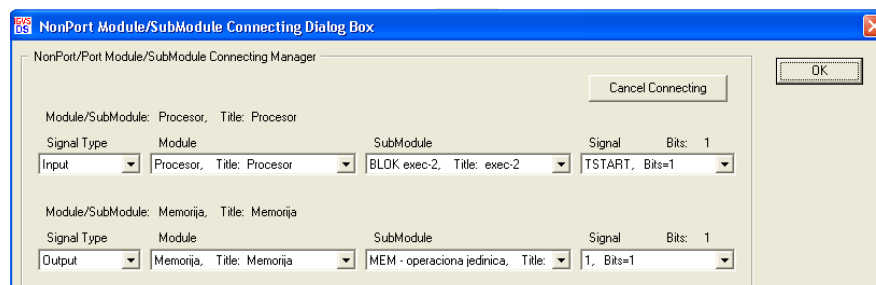
NonPort Module objekti nemaju unapred definisane portove za povezivanje signala. Stoga je omogućeno da se ostvari veza svakog signala van modula sa odgovarajućim signalom unutar modula, koji je vidljiv van modula.

Prilikom povezivanja nekog signala *Port Module*-a i *NonPort Module*-a otvara se dijalog prikazan na slici 15. U takvom dijalogu je moguće izabrati jedan signal iz *NonPort Module*-a koji će se povezati sa ulaznim ili izlaznim portom *Port Module*-a, do koga je dovučena linija signala. Ukoliko korisnik proba da poveže dva ulazna ili dva izlazna signala, greška će biti prijavljena.



Slika 15: Funkcija za povezivanje signala i *NonPort Module*-a

Prilikom povezivanja signala između dva *NonPort Module*-a otvara se dijalog prikazan na slici 16. U takvom dijalogu je moguće izabrati jedan izlazni signal iz jednog *NonPort Module*-a koji će se povezati samo sa ulaznim signalom drugog *NonPort Module*-a. Ukoliko korisnik proba da poveže dva ulazna ili dva izlazna signala, greška će biti prijavljena.



Slika 16: Funkcija za povezivanja signala dva *NonPort Module*-a

Za potrebe povezivanja magistrala i signala unutar *NonPort Module*-a, na raspolaganju su dve vrste povezivanja. Određuje se da li se magistrala povezuje bez prenošenja osobina magistrale na signal unutar *NonPort Module*-a (*Internal NonBUS signal*), ili se magistrala povezuje sa prenošenjem osobina magistrale na signal unutar *NonPort Module*-a (*Internal BUS signal*). Nakon izbora vrste povezivanja, otvara se dijalog, sa listama odgovarajućih signala unutar *NonPort Module*-a. Ukoliko je odabrana vrsta povezivanja *Internal NonBUS signal*, postavlja se *Signal Type=Input*, i prikazuje se lista *Undefined* signala unutar *NonPort Module*-a. Na ovaj način, magistrala postaje izvor vrednosti za *Undefined* signala unutar *NonPort Module*-a. Ukoliko je odabrana vrsta povezivanja *Internal BUS signal*, postavlja se *Signal Type=Output*, i prikazuje se lista *BUS* signala unutar *NonPort Module*-a. Nakon povezivanja, magistrala unutar *NonPort Module*-a postaje sastavni deo magistrale koja se povezuje na *NonPort Module*. Magistrala van *NonPort Module*-a dobija

dodatni status *External BUS*, dok magistrala unutar *NonPort Module*–a dobija dodatni status *Internal BUS*. Obe magistrale se prikazuju i ponašaju kao da postoji samo *External BUS* magistrala.

3.4. Upravljanje simulatorom

Upravljanje simulatorom softverskog paketa *IGoVSoEDS* obuhvata kontrolu režima rada simulatora i upravljanje simulacijom. U cilju pogodnijeg prikaza detaljnog pregleda mogućnosti za upravljanje simulatorom neophodno je da se definišu odgovarajuća stanja simulatora. Značajna stanja simulatora i odgovarajuće oznake su: 1) T_0 – početno stanje simulatora. Ovo stanje može da se dobije aktiviranjem funkcije za vraćanje simulacije u početno stanje, kao i višestrukim aktiviranjem funkcija za vraćanje simulacije unazad, 2) T_{sim} – stanje simulatora koje se trenutno prikazuje u softverskom paketu. Aktiviranjem bilo koje od funkcija za upravljanje simulacijom, u opštem slučaju, stanje simulatora T_{sim} se menja, 3) T_{max} – maksimalno dostignuti vremenski trenutak simulacije za trenutno aktivnu digitalnu strukturu. Upotrebom funkcija simulatora za pokretanje simulacije unapred, u opštem slučaju, vremenski trenutak T_{max} pomera se ka većim vrednostima.

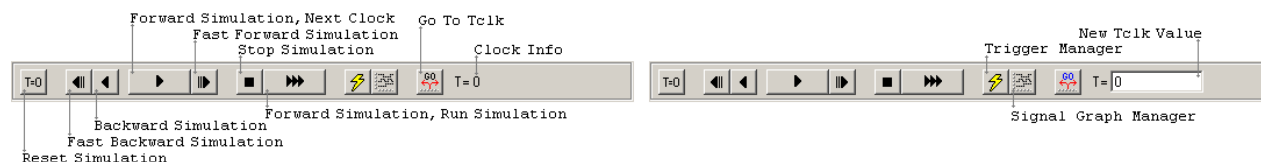
Simulator softverskog paketa ima dva režima rada, koji se označavaju kao režimi rada: 1) *Play* – režim rada kada se stanje simulatora i pregledi rezultata rada simulatora prikazuju za vremenski trenutak $T_{sim}=T_{max}$, 2) *RePlay* – režim rada kada se stanje simulatora i pregledi rezultata rada simulatora prikazuju za vremenski trenutak $T_{sim}<T_{max}$.

Režim rada *Play* je režim rada kada se simulacija ostvaruje samo aktiviranjem tastera za obradu jednog ili više narednih signala takta, bez korišćenja funkcija za vraćanje simulacije unazad. U režimu rada *Play* vremenski trenutak simulacije je uvek T_{max} . U ovom režimu rada na raspolaganju su pregled vremenskih oblika odabranih signala, kao i funkcije za promenu vrednosti svih memorijskih elemenata digitalne strukture.

Aktiviranjem neke od funkcija za vraćanje simulacije u vremenske trenutke koji su već obrađeni, prelazi se u režim rada *RePlay*, kada se kombinacijom memorisanih stanja simulatora, i simulacije rada digitalne strukture za taktove između odabranog memorisanog stanja simulatora i vremenskog trenutka simulacije koji treba prikazati, stanje simulatora dovodi u traženi vremenski trenutak. Režim rada *RePlay* koristi se za prikaz stanja simulatora za vremenske trenutke simulacije $T_{sim}<T_{max}$. Na raspolaganju je maksimalno 10 memorisanih stanja simulatora, koji čuvaju do 8 stanja simulatora, približno ravnomerno raspoređenih od T_0 do T_{max} , kao i 2 stanja simulatora, veoma bliskih vremenskom trenutku T_{max} . Prva grupa memorisanih stanja simulatora koristi se za velika pomeranja simulacije unazad, dok se druga grupa koristi za pomeranja simulacije za jedan takt unazad. Sva memorisana stanja simulatora imaju važeća stanja samo za simulacije koje prelaze više desetina hiljada taktova, dok se za simulacije sa manje obrađenih taktova koriste samo neka od memorisanih stanja. U režimu rada *RePlay* nije moguće pregledati vremenske oblike signala, i nisu na raspolaganju funkcije za promenu vrednosti memorijskih elemenata digitalne strukture. Korišćenjem funkcija za pomeranje stanja simulatora unapred, u

trenutku kada se simulator postavi na vremenski trenutak simulacije T_{\max} , režim rada *RePlay* implicitno prelazi u režim rada *Play*.

Upravljanje simulacijom ostvaruje se korišćenjem tastera grafičkog skupa funkcija *Simulation Toolbar*, koji je prikazan na slici 17.



Slika 17: Grafički skup funkcija za rad sa funkcijama simulatora (*Simulation Toolbar*)

Skup tastera za upravljanje radom simulacije (slika 18–levo), obezbeđuje funkcije za: 1) postavljanje početnog stanja simulatora, odnosno stanja T_0 , za $T_{\max}=0$ (*Reset Simulation*), 2) postavljanje simulatora na prethodno dostignute vremenske trenutke, čime se pokreće *RePlay* režim rada simulatora (*Fast Backward Simulation* i *Backward Simulation*), 3) pokretanje simulatora za jedan takt unapred (*Forward Simulation, Next Clock*), 4) postavljanje simulatora na vremenske trenutke koji se nalaze iza vremenskih trenutaka koji se posmatraju u *RePlay* režimu rada simulatora (*Fast Forward Simulation*), 5) zaustavljanje simulacije (*Stop Simulation*), 6) pokretanje simulatora za više taktova unapred (*Forward Simulation, Run Simulation*), 7) postavljanje simulatora na proizvoljan vremenski trenutak pre ili iza trenutno dostignutog vremenskog trenutka simulacije T_{\max} (*Go To Tclk*).

Na krajnjoj desnoj strani nalazi se natpis na kome se prikazuje vrednost dostignutog trenutka simulacije T_{\max} (*Clock Info*). U režimu rada *RePlay* vrednost parametra *Clock Info* prikazuje se kao T_{sim}/T_{\max} . Na raspolaganju su i tasteri (slika 17 – desno) za otvaranje dijaloga za definisanje događaja za zaustavljanje simulacije (*Trigger Manager*) i za rad sa listom signala za pregled promena vremenskih oblika signala (*Signal Graph Manager*).

Reset Simulation funkcija vraća stanje simulatora na početno stanje T_0 , odnosno na stanje $T_{\max}=0$. Aktiviranjem funkcije *Reset Simulation* brišu se sva memorisana stanja simulatora i sačuvane promene signala za potrebe prikaza vremenskih oblika odabranih signala. Ova funkcija je jedina funkcija koja vraća simulaciju unazad, ali ne postavlja simulator u *RePlay* režim rada.

Fast Backward Simulation postavlja stanje simulatora vremenski unazad, na vremenski najbliže memorisano stanje simulatora, u odnosu na trenutno prikazani vremenski trenutak simulacije. Ukoliko postoji odgovarajuće memorisano stanje simulatora, simulator se prevodu u to stanje i nastavlja da radi u *RePlay* režimu rada. Ukoliko ne postoji odgovarajuće memorisano stanje simulatora, simulator se ne postavlja u drugo stanje, i režim rada ostaje isti kao i pre aktiviranja funkcije.

Backward Simulation postavlja stanje simulatora vremenski unazad, na vremenski trenutak za 1 manji od trenutno prikazanog vremenskog trenutka, i prevodi simulator u *RePlay* režim rada. Ova funkcija ostvaruje simulaciju za jedan takt unazad, i može da dovede simulator u stanje T_0 , ali ostaje $T_{\max}>0$.

Forward Simulation, Next Clock aktivira simulaciju jednog narednog sistemskog signala takta, ali i svih narednih sistemskih signala takta do trenutka pojavljivanja uzlazne ivice prvog narednog signala takta, za koji u sistemu definisanja događaja za

zaustavljanje simulacije (*Trigger Manager*) nije postavljeno *Do Not Stop Simulation*. Funkcija je dostupna u oba režima rada simulatora.

Fast Forward Simulation postavlja stanje simulatora vremenski unapred, na vremenski najbliže memorisano stanje simulatora, u odnosu na trenutno prikazani vremenski trenutak simulacije. Ukoliko postoji odgovarajuće memorisano stanje simulatora, simulator se prevodu u to stanje. Ukoliko ne postoji odgovarajuće memorisano stanje, simulator se ne postavlja u drugo stanje, i režim rada ostaje isti kao i pre aktiviranja funkcije. Samo u *RePlay* režimu rada postoje odgovarajuća memorisana stanja simulatora za rad funkcije *Fast Forward Simulation*.

Stop Simulation funkcija zaustavlja simulaciju, za sve simulacije koje obuhvataju obradu više sistemskih signala takta. Ova funkcija ne utiče na promenu režima rada simulatora.

Forward Simulation, *Run Simulation* funkcija koristi se za startovanje simulacije, odnosno aktiviranje simulacije za obradu proizvoljnog broja sistemskih signala takta do prvog sledećeg kriterijuma za zaustavljanje simulacije (*Trigger Manager*), odnosno do aktiviranja funkcije *Stop Simulation*. Funkcija je dostupna u oba režima rada simulatora.

Go To Tclk funkcija stanje simulatora dovodi u stanje jednako postavljenoj vrednosti sistemskog signala takta u polju *New Tclk Value*. Prvo aktiviranje funkcije *Go To Tclk* kreira polje *New Tclk Value* (slika 17–desno). Naredno aktiviranje izvršava funkciju postavljanja simulatora na vremenski trenutak koji je postavljen u polju *New Tclk Value*, i potom uklanja polje *New Tclk Value*. Ukoliko je vrednost polja *New Tclk Value* manja od vrednosti trenutno prikazanog vremenskog trenutka, simulacija se vremenski pomera unazad, a u suprotnom simulacija se vremenski pomera unapred. Bez obzira na režim rada simulatora, koji je aktivan neposredno pre izvršavanje funkcije *Go To Tclk*, nakon izvršavanja funkcije, režim rada zavisi od vremenskog trenutka u koji je doveden simulator. Ukoliko je traženi vremenski trenutak stanja simulatora *New Tclk Value* manji od T_{\max} , nakon završetka rada funkcije postavlja se režim rada simulatora *RePlay*, a ako je jednak ili veći od T_{\max} , postavlja se režim rada simulatora *Play*, i T_{\max} dobija vrednost postavljenu u polju *New Tclk Value*. Izvršavanje funkcije *Go To Tclk* sastoji se u pronalaženju memorisanog stanja simulatora koje čuva stanje simulatora za najbliži vremenski trenutak manji od traženog vremenskog trenutka postavljenog u *New Tclk Value*, i izvršavanja simulacije rada sistemskih signala takta od memorisanog stanja do traženog stanja simulatora.

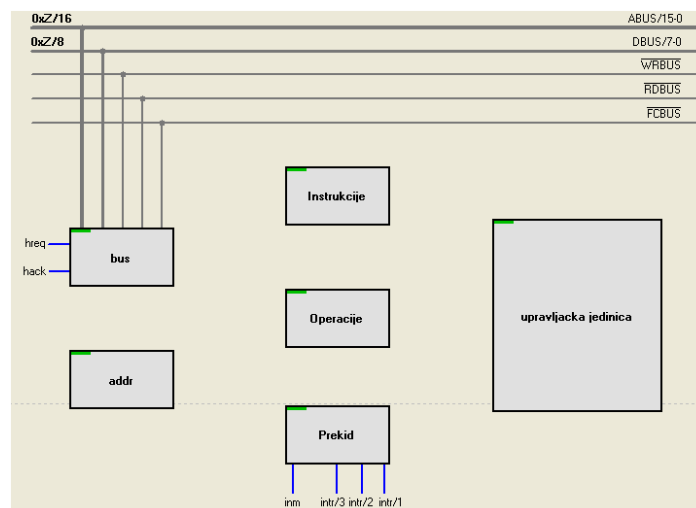
Za potrebe efikasnijeg kreiranja i modifikacije veoma složenih digitalnih struktura, realizovan je parametar *ReConfiguration Always ON*. Ukoliko ovaj parametar nije potvrđen, sa promenom digitalne strukture ne ažurira se struktura podataka koja čuva opis strukture, i ne ažurira se stanje simulatora. U ovom režimu rada, skup funkcija za upravljanje radom simulacije nije dostupan, i pokriven je porukom *ReConfiguration is Stopped. Some values likely are incorrect*. Stanja signala ne moraju da budu odgovarajuća, kao i vrednosti u svim ostalim pregledima rezultata rada simulatora. Nakon potvrđivanja parametra *ReConfiguration Always ON*, stanje strukture podataka za čuvanje opisa digitalne strukture postaje važeće, i funkcije za upravljanje radom simulacije postaju dostupne.

4. SIMULATOR CPU

U ovoj glavi daje se detaljan opis procesora i memorije ovog računarskog sistema. Opis uključuje sve module i podmodule sve do nivoa osnovnih logičkih kola, što omogućava lakše razumevanje rada ovog sistema i uvid u stanje svih relevantnih delova sistema u svakom trenutku. Ovaj deo simulatora sistema realizovao je Nikola Todorović u okviru svog diplomskog rada na Elektrotehničkom Fakultetu u Beogradu, ali opis se daje radi lakšeg razumevanja programskih primera u glavi 6. Delovi teksta su preuzeti iz [1] uz saglasnost autora, a slike su preuzeti iz [9].

4.1. Procesor

Procesor se sastoji iz operacione jedinice i upravljačke jedinice. U daljem tekstu će se dati detaljan opis iz kojih delova, odnosno blokova se sastoji procesor kao i način na koji funkcioniše procesor. Sto se operacione jedinice tiče, biće objašnjeno na koji način se podaci razmenjuju između pojedinih blokova unutar procesora kao i na koji način je procesor povezan sa ostatkom računara. U delu za upravljačku jedinicu biće opisan algoritam generisanja upravljačkih signala i struktura upravljačke jedinice.



Slika 18: Šema procesora

4.1.1. Operaciona jedinica

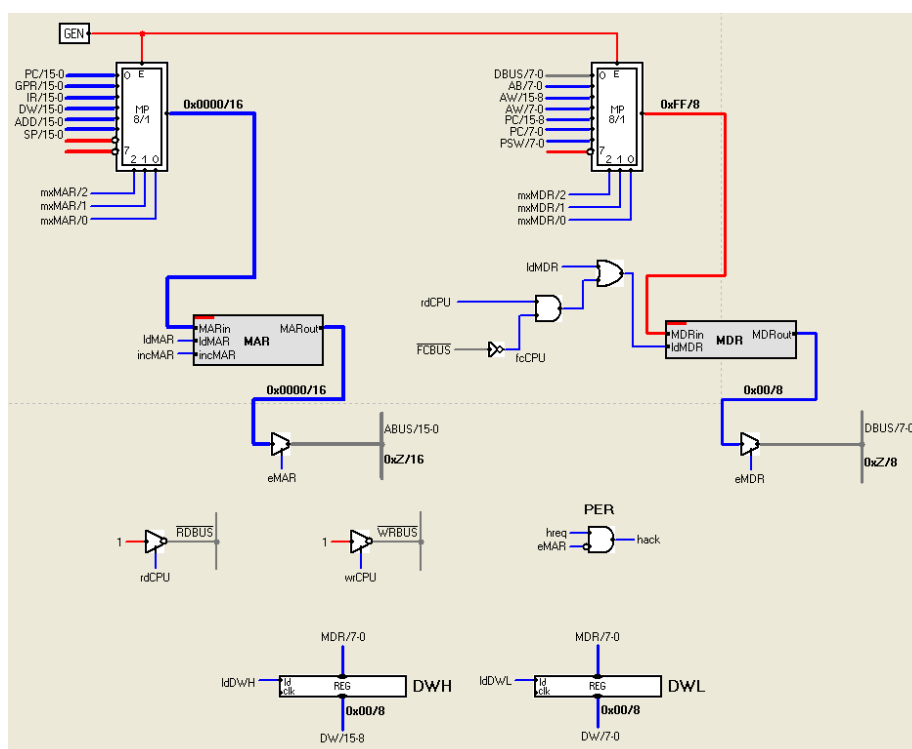
Operaciona jedinica se sastoji iz pet modula i više podmodula:

- 1) bus;
- 2) addr;
- 3) Instrukcije: a)fetch-1; b)fetch-2; c)fetch-3;
- 4) Operacije: a)exec-1; b)exec-2; c)exec-3; d)exec-4;
- 5) Prekidi: a)intr-1; b)intr-2;

U nastavku će ukratko biti opisan svaki od ovih modula odnosno podmodula uz čije će opise biti priložene i slike, realizovane u programu IGoVSoDS.

4.1.1.1. Blok 'bus'

Blok 'bus' predstavlja jedini modul u procesoru koji je direktno povezan na magistralu. On sadrži registre $MAR_{15...0}$ i $MDR_{7...0}$ sa multiplekserima MP, kombinacionom i sekvencijalnom logikom za realizaciju ciklusa na magistrali. Registar $MAR_{15...0}$ je 16-razredni adresni registar na čiji se ulaz dovodi jedna od vrednosti koja se propušta kroz multiplekser. Na ulaz registra $MAR_{15...0}$ prema tome može doći sadržaj registra PC-a, SP-a, sabirača, jednog od registara opšte namene. Ovaj sadržaj se upisuje u registar $MAR_{15...0}$ aktiviranjem signala **ldMAR** koji se dovodi iz upravljačke jedinice. Takođe, moguće je inkrementirati sadržaj registra $MAR_{15...0}$ aktiviranjem signala **incMAR**. Sadržaj registra $MAR_{15...0}$ se, generisanjem aktivne vrednosti signala **eMAR**, propušta kroz bafere sa tri stanja na adresnu magistralu. Ovaj podatak se koristi za dopremanje podatka iz memorije sa lokacije čija se adresa nalazi u ovom registru.



Slika 19: Blok 'bus'

Pored ovog registra, u bloku 'bus' nalazi se i prihvatni registar za podatke $MDR_{7...0}$. On je 8-razredni registar, čiji se sadržaj normalno koristi pri realizaciji ciklusa čitanja ili upisa na magistrali **BUS**. Generisanjem aktivne vrednosti signala **ldMDR** u registar $MDR_{7...0}$ se upisuje sadržaj sa izlaza multipleksera MP. U ovaj registar je moguće upisati kako vrednosti sa magistrale podataka, tako i vrednost registra PC ili registra PSW ili akumulatora. Kada je potrebno upisati neku od ovih vrednosti u registar $MDR_{7...0}$, potrebno je da bude aktivan samo signal **ldMDR**. Ako se u ovaj registar upisuje podatak sa magistrale podataka, aktivni moraju biti signali **rdCPU** i **fcCPU**. To znači da je podatak koji je procesor zahtevao od memorije spreman i da se nalazi na magistrali podataka.

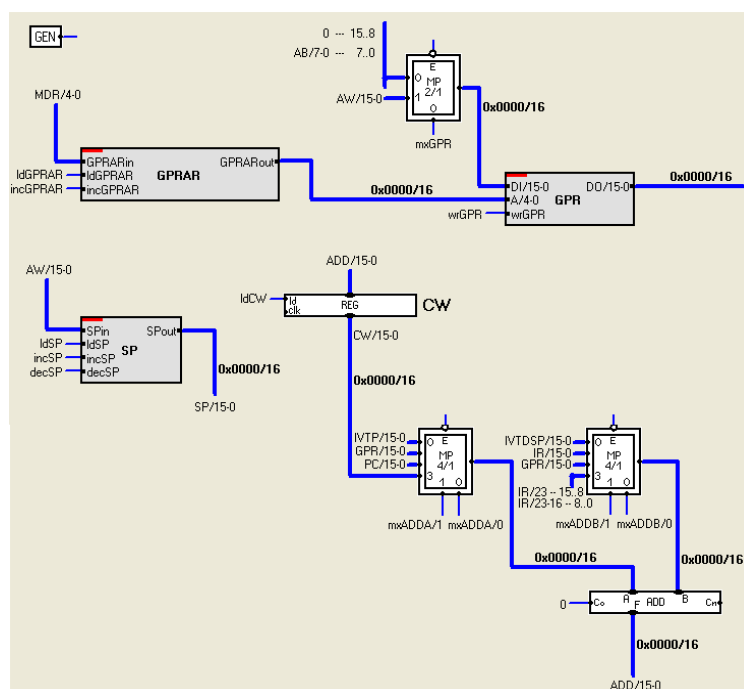
U ovom delu se nalazi i registar za podatke kome se prosleđuje sadržaj registra $MDR_{7...0}$, i on je realizovan iz dva registra $DWH_{7...0}$ i $DWL_{7...0}$ koji praktično čine

jednu celinu - registar $DW_{15...0}$. Izlaz tog registra se dovodi na ulaze mnogih registara u procesoru koji nemaju pristup magistrali već im je ovo jedini način da dođu do podataka iz memorije koji su im potrebni.

4.1.1.2. Blok 'addr'

U ovom modulu se nalazi registar Stack Pointer – SP. Njegova vrednost može da se učita iz akumulatora, aktiviranjem signala **ldSP**, a može i da se inkrementira i dekrementira aktiviranjem odgovarajućih signala. U ovom delu su realizovana i 32 registra opšte namene koja su smeštena u jedan portModul - GPR. Na njegov ulaz se dovodi podatak (ako se želi izvršiti upis u neki od registara) i adresa registra u kome se želi smestiti taj podatak. Aktiviranjem upravljačkog signala **wrGPR** se vrši upis. Ukoliko se vrši čitanje podatka iz nekog od registara, na ulaz se samo dovodi adresa tog registra, dok će se na izlazu pojaviti traženi podatak (sadržaja datog registra).

Ovaj modul takođe sadrži i realizovan sabirač koji se koristi u ovom procesoru za izračunavanje adresa gde se nalazi podatak kod različitih načina adresiranja. Izlaz sabirača se dovodi na ulaz registra $MAR_{15...0}$ aktiviranjem odgovarajućih upravljačkih signala na ulazu multiplexera bloka 'bus'.



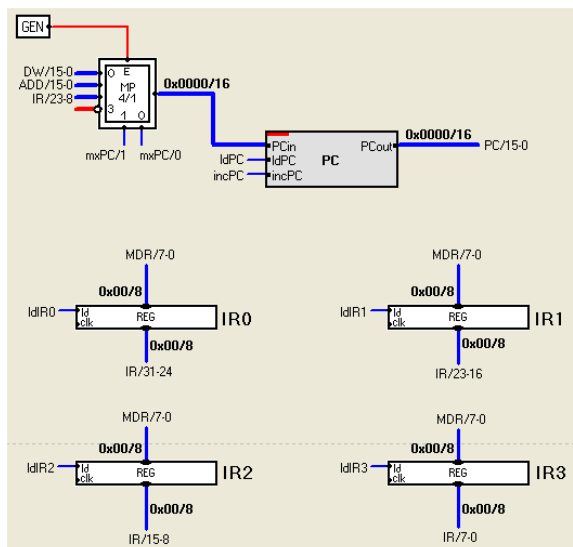
Slika 20: Blok 'addr'

4.1.1.3. Blok 'Instrukcije'

Ovaj modul se sastoji iz tri podmodula i njegov cilj je da instrukciju koja je dohvaćena iz memorije razloži na delove i da ustanovi koja se operacija izvršava i koji je tip adresiranja u pitanju.

U podmodulu **fetch-1** se nalaze registri PC, IR1, IR2, IR3 i IR4. Registar $PC_{15...0}$ je programski brojač i u njemu se čuva podatak koji označava mesto u memoriji gde se nalazi sledeća instrukcija koja se treba izvršavati. Registri $IR_{0...0}$,

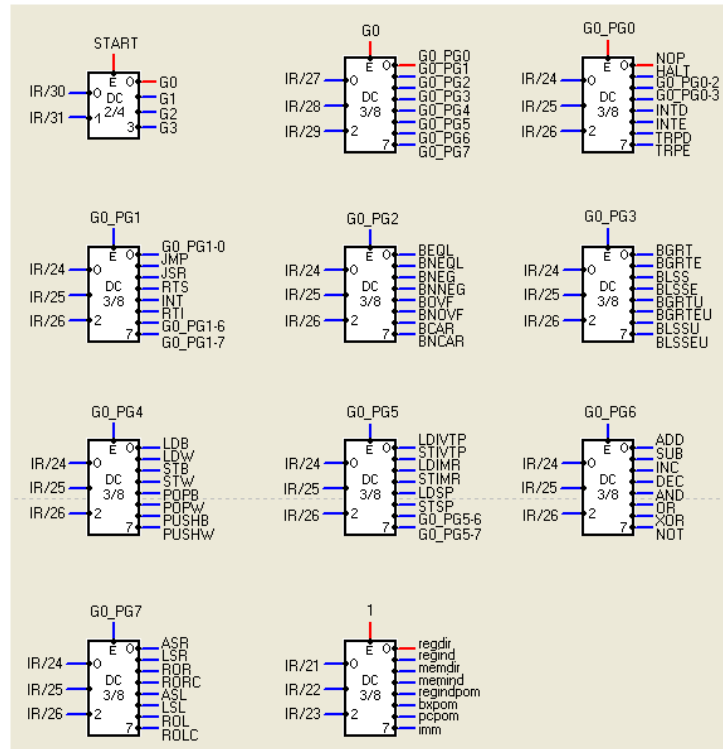
IR1_{7...0}, IR2_{7...0} i IR3_{7...0} formiraju prihvatni registar instrukcije, pri čemu se za instrukcije dužine 32 bita koriste sva četiri registra, za instrukcije dužine 24 bita se koriste registri IR0_{7...0}, IR1_{7...0} i IR2_{7...0}, za instrukcije dužine 16 bita registri IR0_{7...0} i IR1_{7...0} a za instrukcije dužine 8 bita samo registar IR0_{7...0}.



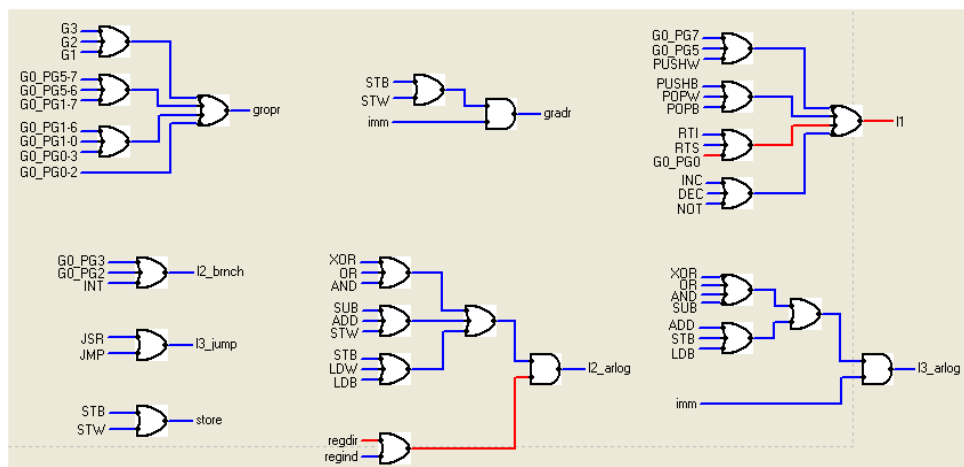
Slika 21: Blok 'fetch-1'

Podmodul **fetch-2** na osnovu prvog bajta instrukcije, određuje koja se instrukcija iz koje grupe instrukcija izvršava. Na osnovu tri najvise bita drugog bajta instrukcije se određuje koji je tip adresiranja u pitanju u tekućoj instrukciji, kod višeadresnih instrukcija. Kod instrukcija čija je dužina samo 8 bajta postoji samo jedan način adresiranja – neposredno adresiranje. Ova logika je realizovana uz pomoć dekodera 3/8.

U podmodulu **fetch-3** se na osnovu instrukcije, koja je određena u prethodnom koraku, ispituje kolika je dužina cele instrukcije. Signali koji određuju ovu dužinu su: **I1**, **I2arlog**, **I2brnch**, **I3arlog**, **I3brnch**. Aktivna vrednost signala **I1** znači da je instrukcija dužine 8 bita (1 reč), aktivna vrednost signala **I2arlog** znači da je u pitanju jednoadresna registarska instrukcija dužine 16 bita (2 reči). Aktivna vrednost signala **I2brnch** označava da je u pitanju instrukcija prekida ili relativnog skoka čija je dužina takođe 16 bita (2 reči). **I3arlog** signal je aktivan kada je instrukcija dužine 24 bita (3 reči) i kod te vrste instrukcija moguće je samo neposredan način adresiranja. Signal **I3brnch** je aktivan kod instrukcija apsolutnog skoka čija je dužina takođe 3 reči. Na kraju, za instrukcije čija je dužina 32 bita (4 reči) nije aktivan nijedan od ovih signala. Signali **gopr** i **gradr** su aktivni ako je došlo do neke greske u instrukciji, na primer ako se pokušava izvršavanje instrukcija STORE sa neposrednim načinom adresiranja.



Slika 22: Blok 'fetch-2'



Slika 23: Blok 'fetch-3'

4.1.1.4. Blok 'Operacije'

Ovaj modul se sastoji iz četiri podmodula u kojima se izvršavaju operacije u instrukciji koja je određena u prethodnom koraku kao i azuriranje pratećih indikatora tih instrukcija.

Podmodul **exec-1** je glavni u ovom bloku jer se u njemu nalazi aritmeticko-logicka jedinica koja je zadužena za izvršavanje mikrooperacija. Te mikrooperacije se izvršavaju nad sadržajima registra $AB_{7...0}$ i $BB_{7...0}$ i one se specificiraju upravljačkim

signalima **add**, **and**, **inc**, **xor**, **sub**, **or**, **dec**, **not** koji se dovode na ulaz portModula realizovane ALU jedinice. Rezultat mikrooperacije se dobija na linijama ALU_{7...0} a u slučaju aritmetičkih mikrooperacija prenos se dobija na liniji C₈. Osim aritmetičko-logičkih operacija, moguće je izvršavati i pomeracke operacije. Za to se koristi pomerački registar AB na čiji se ulaz dovode upravljački signali kojima se specificira koja vrsta pomeračke mikrooperacije se treba izvršiti.

Aritmetički deo služi za realizaciju mikrooperacija:

- sabiranje sadržaja registara AB_{7...0} i BB_{7...0} (signal **add** aktivan),
- oduzimanje sadržaja registara AB_{7...0} od registra BB_{7...0} (signal **sub** aktivan),
- sabiranje vrednosti jedan sa sadržajem registra AB_{15...0} (signal **inc** aktivan) i
- oduzimanje vrednosti jedan od sadržaja registra AB_{15...0} (signal **dec** aktivan).

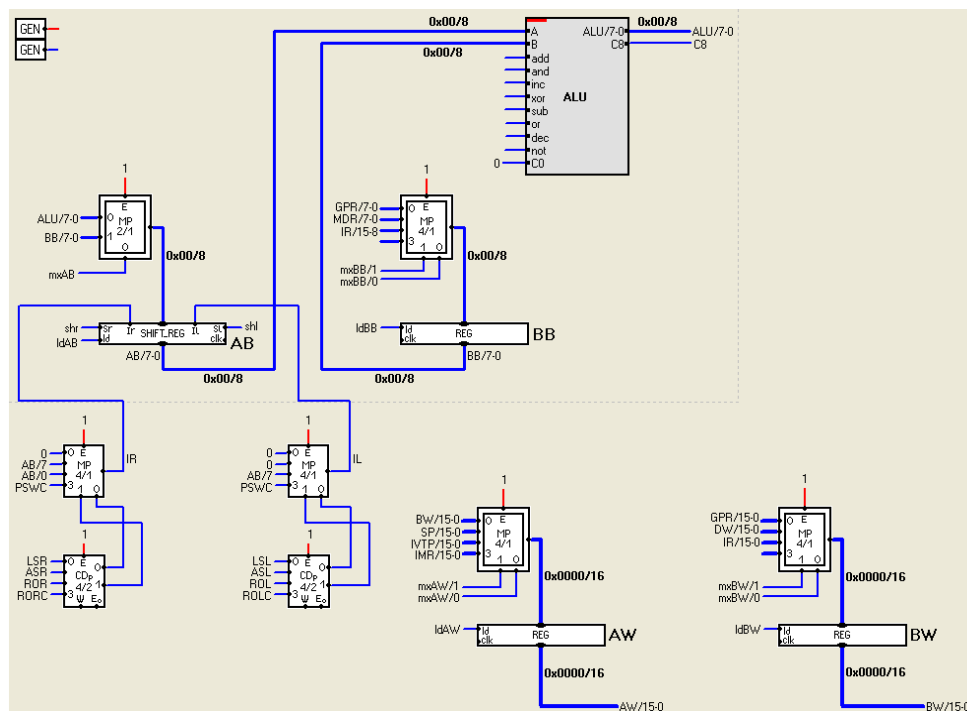
Logički deo služi za realizaciju sledećih mikrooperacija:

- logička funkcija komplementiranja sadržaja registra AB_{7...0} (signal **not** aktivan).
- logička funkcija ILI sadržaja registara AB_{7...0} i BB_{7...0} (signal **or** aktivan)
- logička funkcija ekskluzivno ILI sadržaja registara AB_{7...0} i BB_{7...0} (signal **xor** aktivan)

Pomerački deo služi za realizaciju mikrooperacija:

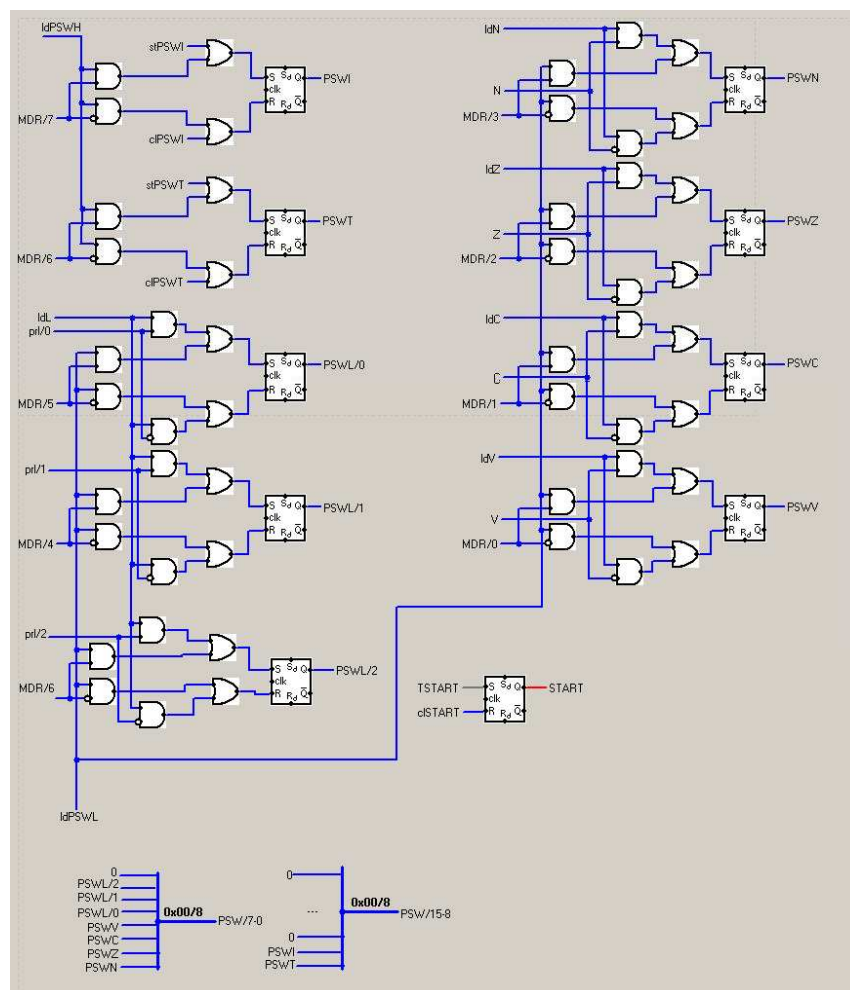
- logičko i aritmetičko pomeranje ulevo za jedno mesto sadržaja registra AB_{7...0} (signal **shl** aktivan)
- logičko i aritmetičko pomeranje udesno za jedno mesto sadržaja registra AB_{7...0} (signal **lsr** aktivan)

Prilikom izvršavanja ovih pomerackih mikrooperacija, na ulaze IR i IL se dovode odgovarajuće vrednosti u zavisnosti od toga da li se izvršava aritmetičko ili logičko pomeranje.



Slika 24 : Blok 'exec-1'

U narednom podmodulu ovog bloka **exec-2**, flip-flopovi PSWI, PSWT, PSWL₂, PSWL₁, PSWL₀, PSWV i PSWC, PSWZ, PSWN formiraju razrede 15, 14, 6, 5, 4, 3, 2, 1 i 0 programske statusne reči PSW_{15...0}. U registar PSW_{15...0} se može upisivati na dva načina i to ili u sve razrede istovremeno ili u grupe od jednog ili nekoliko razreda. Upis u sve razrede istovremeno se realizuje na signal takta tako što se u registar MDR_{7...0} postavi sadržaj koji treba upisati i generiše aktivna vrednost signala **ldPSWL** i **ldPSWH**. Upis u grupu od jednog ili nekoliko razreda registra PSW_{7...0} se neznatno razlikuje za pojedine grupe razreda.



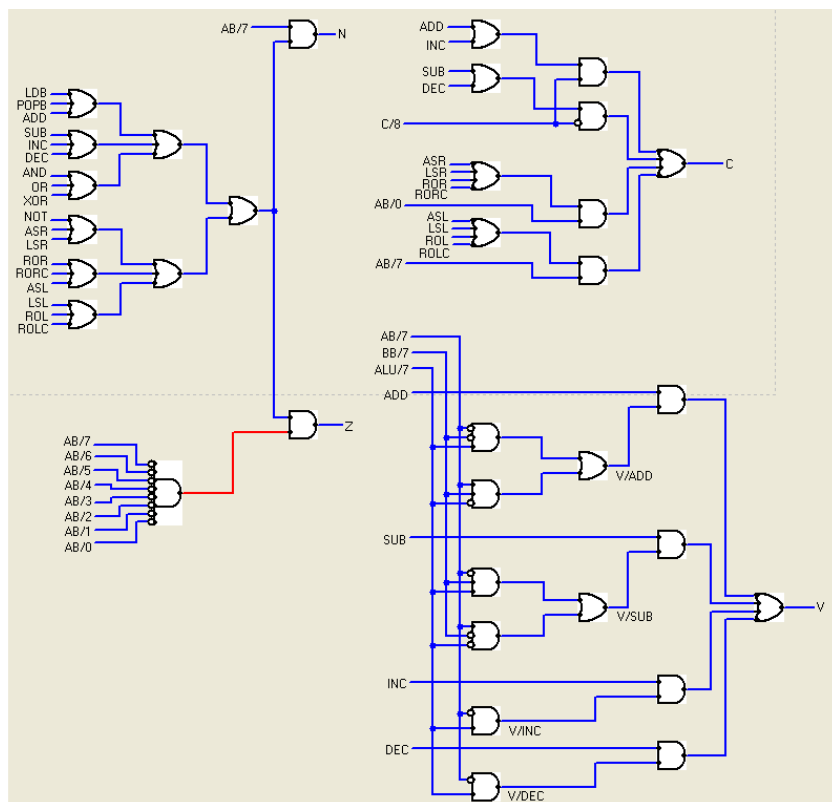
Slika 25 : Blok 'exec-2'

Upis u razrede PSWN, PSWZ, PSWC i PSWV se realizuje onda kada na osnovu rezultata izvršene operacije treba ažurirati ove indikatore. Vrednosti N, Z, C i V se upisuju na signal takta generisanjem aktivnih vrednosti signala **ldN**, **ldZ**, **ldC**, **ldV**.

Upis u razrede PSWL₂, PSWL₁ i PSWL₀ se realizuje prilikom opsluživanja maskirajućih prekida. Vrednosti prl₂, prl₁ i prl₀ se upisuju na signal takta generisanjem aktivne vrednosti signala **ldL**. Upis u razrede PSWI i PSWT se realizuje posebnim instrukcijama. Aktivna vrednost se upisuje generisanjem aktivne vrednosti signala **stPSWI** ili **stPSWT** a neaktivna vrednost generisanjem aktivne vrednosti signala **clPSWI** ili **clPSWT**.

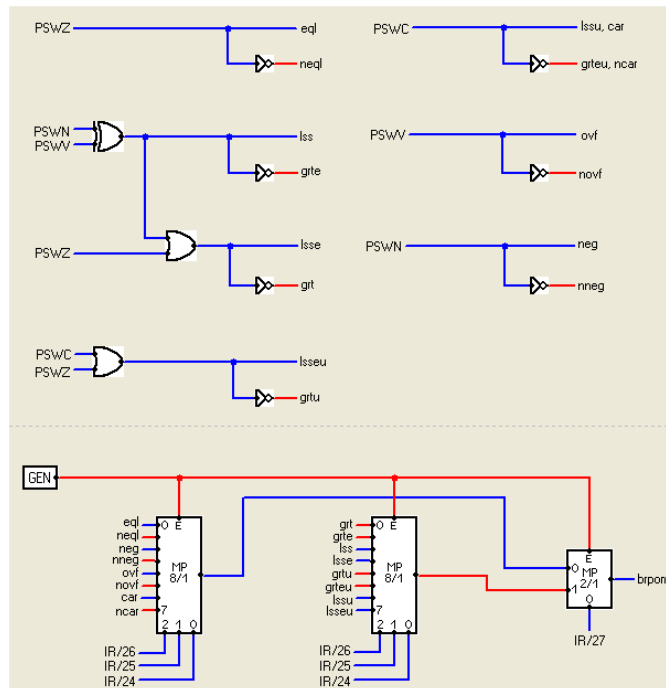
Prilikom čitanja sadržaja registra $PSW_{15..0}$, ukoliko je potrebna njegova vrednost za izvršavanje neke operacije, izlaz ovog registra se direktno dovodi na ulaz tog kola.

Podmodul **exec-3** azurira vrednosti indikatora N, Z, C, V na osnovu instrukcije koja se trenutno izvršava.



Slika 26: Blok 'exec-3'

Podmodul **exec-4** služi za generisanje signala **brpom**. Ukoliko se trenutno izvršava instrukcija uslovnog skoka, potrebno je proveriti da li je aktivan onaj indikator u registru PSW za koji se skok zahteva. Ako je takav signal aktivan, biće aktivan i signal **brpom** čime se signalizira da je uslov ispunjen i biće izvršen skok.

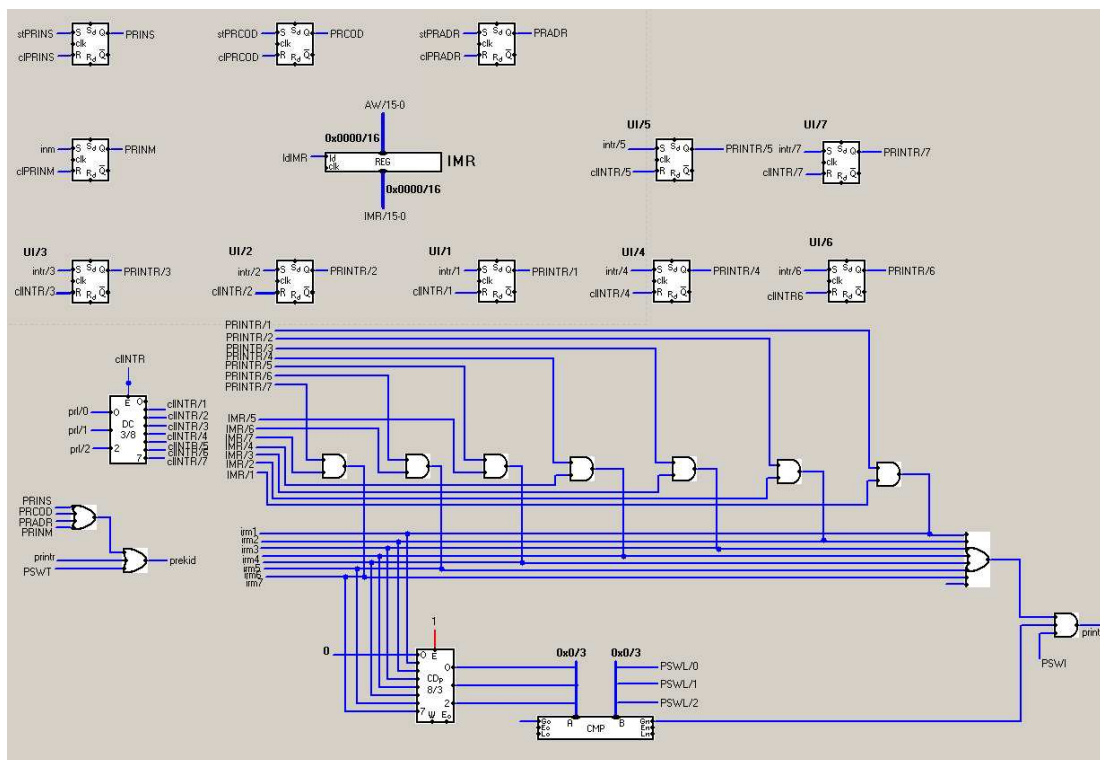


4.1.1.5. Blok 'Prekid'

U prvom podmodulu **intr-1** nalazi se jedan logički ILI element koji služi za formiranje signala prekida **prekid**. Ovaj signal ima aktivnu vrednost ukoliko je aktivan barem jedan od signala prekida. To može da bude neki od signala unutrašnjih prekida i to **PRINS** za prekid zbog izvršavanja instrukcije prekida INT, **PRCOD** za prekid usled čitanja instrukcije sa nepostojećim kodom operacije, **PRADR** zbog greske u adresiranju, zatim signal spoljašnjeg prekida **PRINM** zbog generisanja nemaskirajućeg prekida, potom signal spoljašnjeg prekida **printr** zbog prisustva nekog od maskirajućih prekida i nakraju signal unutrašnjeg prekida **PSWT** usled prekida zbog zadatog režim rada procesora prekid posle svake instrukcije.

Flip-flop PRCOD pamti unutrašnji zahtev za prekid zbog čitanja instrukcije sa nepostojećim kodom operacije. Ukoliko je vrednost signala **gopr** aktivna ili neaktivna (u zavisnosti od toga da li je po očitavanju instrukcije otkriven ili nije otkriven nepostojeći kod operacije), aktivira se signal **stPRCOD** pri čemu se setuje

flip-flop **PRCOD**. Ovaj flip-flop se postavlja na neaktivnu vrednost signalom **cIPRCOD** u okviru opsluživanja ovog zahteva za prekid.



Slika 28: Blok 'intr - 1'

Slična je situacija kod flip-flopa **PRADR**. Ukoliko je vrednost signala **gradr** aktivna, aktivira se signal **stPRADR** čime se ovja flip-flop setuje. Resetovanje se vrši aktiviranjem signala **cIPRADR** u okviru opsluživanja ovog zahteva za prekid.

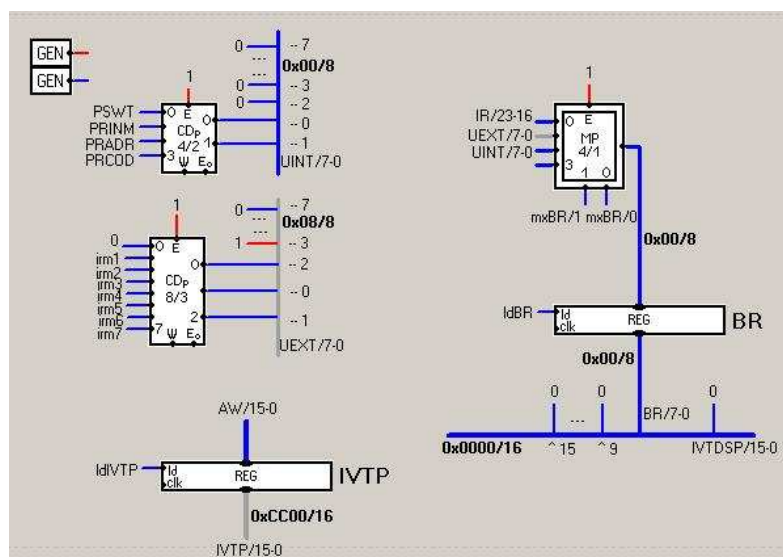
Kombinacione i sekvencijalne mreže za prihvatanje spoljašnjih nemaskirajućih i maskirajućih prekida se sastoje od flip-flova **PINM**, **PRINM**, **PRINTR₁**, **PRINTR₂**, **PRINTR₃**, registra maske **IMR_{15...0}**, koda prioriteta **CD**, komparatora **CMP** i logičkih elemenata.

Flip-flop **PRINM** služi za prihvatanje spoljašnjeg nemaskirajućeg zahteva za prekid **inm**. Zahtev za prekid **inm** dolazi od uređaja **FAULT** kao impuls i pamti se u flip-flopu **PRINM** na prvi signal takta. Ovaj flip-flop se postavlja na neaktivnu vrednost aktivnom vrednošću signala **cliPRINM** u okviru opsluživanja ovog zahteva za prekid.

Flip-flovi **PRINTR₁** do **PRINTR₇** služe za prihvatanje spoljašnjih maskirajućih zahteva za prekid **intr₁** do **intr₇**. Zahtevi za prekid **intr₁** do **intr₇** dolaze od ulazno/izlaznih uređaja **U/I** kao impulsi i pamte se u flip-flovima za prihvatanje prekida **PRINTR₁** do **PRINTR₇**, respektivno. Kada se u okviru opsluživanja zahteva za prekid prihvati neki od maskirajućih zahteva za prekid odgovarajući flip-flop **PRINTR₁** do **PRINTR₇** se postavlja na neaktivnu vrednost. Postavljanje odgovarajućeg para flip-flova na neaktivnu vrednost se realizuje tako što se u okviru opsluživanja spoljašnjeg maskirajućeg zahteva za prekid generiše aktivna vrednost signala **clINTR₁** do **clINTR₇**. Signali **prl₀**, **prl₁** i **prl₂** se koriste da selektuju ove signale.

Signal maskirajućeg prekida **printr** ima aktivnu vrednost ukoliko signali **imrprintr**, **acc** i **PSWI** imaju aktivne vrednosti. Signal **imrprintr** ima aktivnu vrednost ukoliko je aktivna vrednost barem jednog od signala **irm₁** do **irm₇**. Ovo će se desiti ukoliko se barem u jednom od flip-flopora **PRINTR₁** do **PRINTR₇** nalazi aktivna vrednost i ukoliko je u odgovarajućem razredu **IMR₁** do **IMR₇** registra maske **IMR_{15...0}** takođe aktivna vrednost. Signal **acc** na izlazu komparatora **CMP** ima aktivnu vrednost ukoliko je prioritet pristiglog spoljašnjeg maskirajućeg zahteva za prekid najvišeg prioriteta koji nije maskiran viši od prioriteta tekućeg programa. Prioritet pristiglog spoljašnjeg maskirajućeg zahteva za prekid najvišeg prioriteta koji nije maskiran određen je signalima **prl₂**, **prl₁** i **prl₀** na izlazu koda prioriteta **CD**, dok je prioritet tekućeg programa određenog signalima **PSWL₂**, **PSWL₁** i **PSWL₀** registra **PSW_{15...0}**. Signal **PSWI** dolazi sa izlaza odgovarajućeg razreda registra **PSW_{15...0}** i njegovom aktivnom i neaktivnom vrednošću se dozvoljavaju i maskiraju svi maskirajući prekidi. Potvrda periferiji, u slučaju da je njen prekid prihvaćen, se šalje softverski na početku odgovarajuće prekidne rutine.

Registar IVTP_{15...0} je ukazivač na tabelu sa adresama prekidnih rutina i sadrži početnu adresu tabele. Upis sadržaja iz akumulatora AW_{15...0} u registar IVTP_{15...0} se obavlja ukoliko je aktivan signal **ldIVTP**.



Koder CD1 služi za formiranje broja ulaza u tabelu sa adresama prekidnih rutina za unutrašnje prekide i spoljašnji nemaskirajući prekid. Ovi prekidi su određeni signalima na izlazima flip-flopova PSWT za unutrašnji prekid usled zadatog režima

rada prekid posle svake instrukcije, PRINM za spoljašnji nemaskirajući prekid, PRADR za unutrašnji prekid zbog korišćenja neposrednog adresiranja za određeni operand i PRCOD za unutrašnji prekid zbog čitanja instrukcije sa nepostojećim kodom operacije. Signali sa izlaza flip-flopova PSWT, PRINM, PRADR i PRCOD dovedeni su na ulaze 0 do 3 kodera prioriteta CD1 po rastućim prioritetima. Na izlazu kodera dobija se broj zahteva za prekid najvišeg prioriteta binarno kodiran. Ova dvobitna vrednost se proširuje na ukupno 16 bitova (bitovi 2 do 15 broja ulaza imaju vrednost 0) i ta vrednost se dovodi na ulaz multipleksera.

Koder CD2 služi za formiranje broja ulaza u tabelu sa adresama prekidnih rutina za spoljašnje maskirajuće prekide $intr_1$ do $intr_7$ u slučaju kada su ulazi za ove prekide fiksni. Signali spoljašnjih maskirajućih prekida $intr_1$, do $intr_7$ posle eventualnog maskiranja razredima IMR_1 do IMR_7 registra maske $IMR_{15...0}$ pojavljuju se kao signali **irm₁** do **irm₇**. Ovi signali se vode na ulaze 1 do 7 kodera prioriteta CD2 po rastućim prioritetima. Na izlazu kodera dobija se broj zahteva za prekid najvišeg prioriteta binarno kodiran. I ova trobitna vrednost se proširuje na 16 bitova (bit 3 ima vrednost 1, ostali 0) i prosleđuje na ulaz multipleksera.

Treća vrednost koja se dovodi na ulaz multipleksera je ustvari druga reč instrukcije, koja predstavlja pomeraj koji se sabira sa vrednoscu PC-a da bi se dobila adresa skoka kod instrukcija relativnog skoka. Jedna od ovih vrednosti će biti propuštena kroz multiplekser i sabiraće se u sabiraču sa drugim operandom, bilo da je to PC bilo sama vrednost registra IVTP.

4.1.2. Upravljačka jedinica

U ovom poglavlju je prikazana struktura upravljačke jedinice sva svim delovima iz kojih se sastoji.

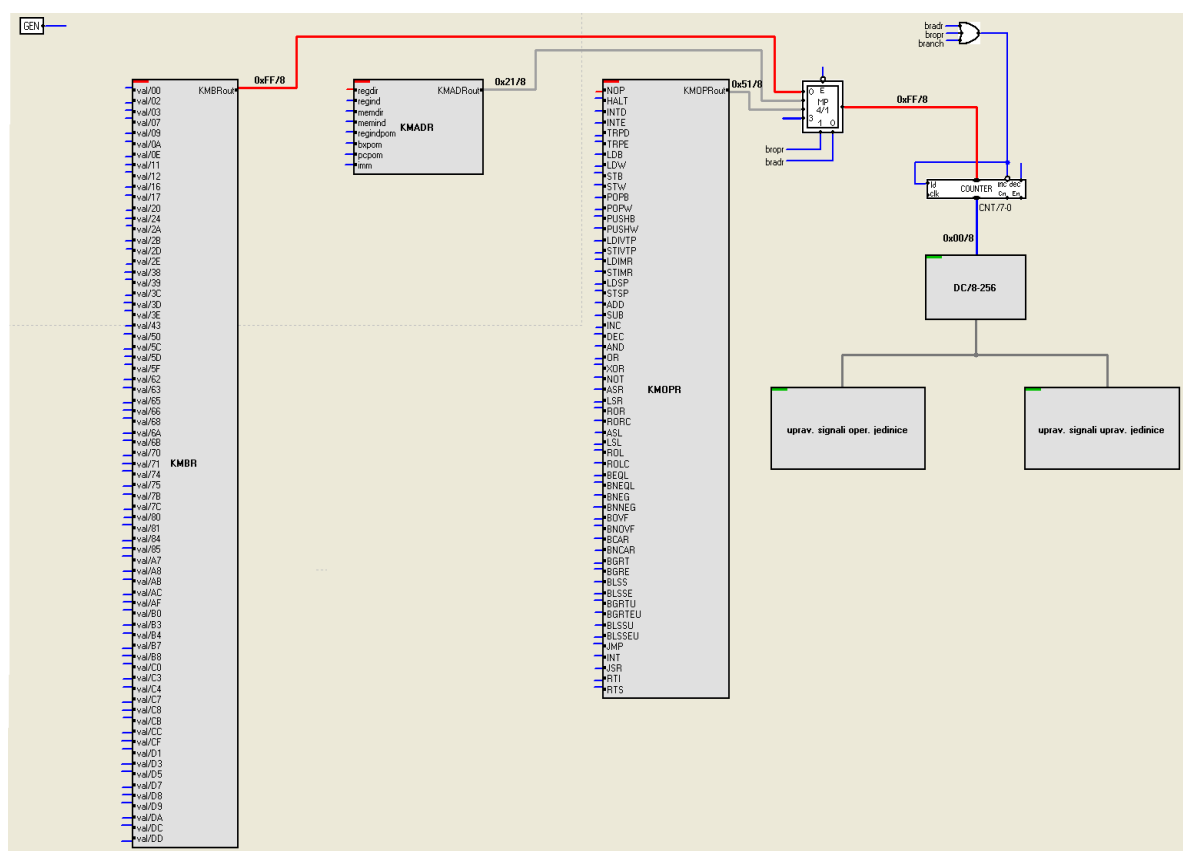
4.1.2.1. Algoritam generisanja upravljačkih signala

Upravljačka jedinica, odnosno blokovi za generisanje upravljačkih signala operacione i upravljačke jedinice se konstruiše na osnovu algoritma generisanja upravljačkih signala koji je kompletno dat u [1].

4.1.2.2. Struktura upravljačke jedinice

Struktura upravljačke jedinice se sastoji iz više delova. U jednom delu se vrši *generisanje nove vrednosti brojača koraka* i taj deo se sastoji od kombinacionih mreža KMOPR, KMADR i KMBR sa multiplekserom MP i služi za generisanje i selekciju vrednosti koju treba upisati u brojač koraka. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikrooperacija. Selekcija jedne od tri grupe signala koji daju novu vrednost brojača koraka obezbeđuje se signalima **bropr** i **bradr**. Ako je aktivan signal **bropr**, propušta se vrednost iz mreže KMOPR, ako je aktivan signal **bradr**, propušta se vrednost iz mreže KMADR, a ako nije aktivan nijedan od ovih signala, propušta se vrednost iz mreže KMBR.

Najzad, poslednju celinu upravljačke jedinice čini deo za *generisanje upravljačkih signala* koji sadrži kombinacionu mrežu koja pomoću signala $\mathbf{T}_0, \mathbf{T}_1, \dots, \mathbf{T}_{FF}$ koji dolaze iz bloka *dekoder koraka*, signala logičkih uslova $\mathbf{I1}, \mathbf{I2}, \dots, \mathbf{prekid}$ koji dolaze iz operacione jedinice i saglasno sekvenci upravljačkih signala za upravljačku jedinicu generišu dve grupe upravljačkih signala i to: upravljačke signale operacione jedinice i upravljačke signale upravljačke jedinice.



Slika 30: Upravljačka jedinica

4.2. Memorija

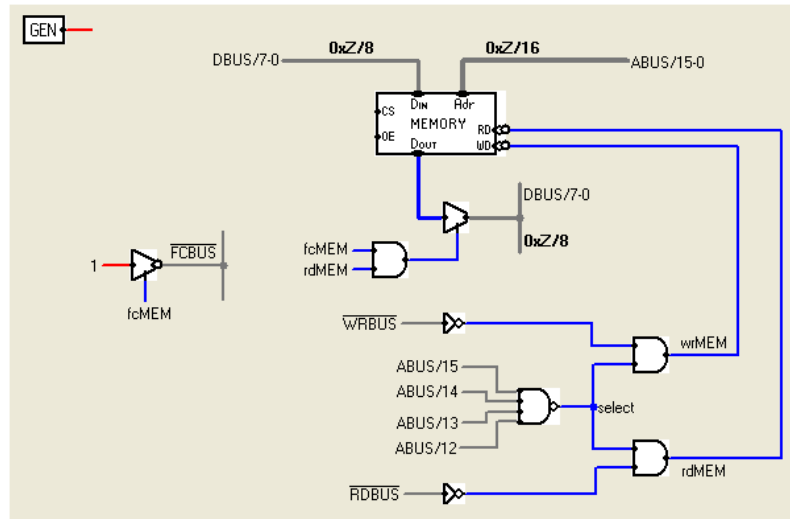
Memorija je realizovana i sastoji se iz dve celine: operacione jedinice i upravljačke jedinice. Upravljačka jedinica služi za generisanje upravljačkih signala koji se koriste u operacionoj jedinici, a operaciona jedinica služi za realizaciju ciklusa čitanja i upisa iniciranih od strane procesora. U nastavku je dato detaljno objašnjenje koje su uloge ovih delova u procesu čitanja, odnosno upisa podatka.

4.2.1. Operaciona jedinica

Operaciona jedinica se sastoji od jednog memorijskog RAM modula i kombinacione mreže koja omogućava čitanje i upis podataka u taj modul. Postoji ukupno 2^{16} memorijskih lokacija i svaka lokacija je veličine jednog bajta (jedna reč). Najvisih 2^{12} lokacija (F000-FFFF) je rezervisano za periferne uređaje i njima se ne može pristupiti niti u njih upisivati podaci.

Dok ne krene sa realizacijom ciklusa čitanja procesor drži adresne linije **ABUS** i upravljačku liniju čitanja **RDBUS** u stanju visoke impedanse. To isto čini i memorija sa linijama podataka **DBUS** i upravljačkom linijom završetka ciklusa **FCBUS**. Procesor kreće sa realizacijom ciklusa čitanja tako što otvara bafere sa tri stanja za adresne linije **ABUS** i upravljačku liniju čitanja **RDBUS**. Na linijama **ABUS** je adresa lokacije, a na liniji **RDBUS** neaktivna vrednost signala čitanja. Sadržaj sa linija **ABUS** i signal sa linije **RDBUS** prima memorija i tada započinje proces čitanja podatka. Na linijama **DBUS** je dok traje čitanje nedefinisani sadržaj, a na liniji **FCBUS** neaktivna vrednost signala. Kada se u slugi završi čitanje na linijama **DBUS** se pojavljuje očitani sadržaj, a na liniji **FCBUS** aktivna vrednost. Na aktivnu vrednost signala **FCBUS** reaguje procesor i upisuje sadržaj sa linija **DBUS** u svoj prihvatni registar podatka. Po završetku upisa procesor postavlja liniju **RDBUS** na neaktivnu vrednost, što je indikacija memoriji da sadržaj sa linija **DBUS** nije više potreban. Na neaktivnu vrednost signala **RDBUS** reaguje memorija tako što ukida sadržaj sa linija podataka **DBUS** i prebacuje ove linije u stanje visoke impedanse, i postavlja liniju **FCBUS** najpre na neaktivnu vrednost a zatim i u stanje visoke impedanse. Na neaktivnu vrednost signala **FCBUS** reaguje memorija tako što ukida sadržaj sa adresnih linija **ABUS** i prebacuje ove linije u stanje visoke impedanse. Time je ciklus čitanja kompletiran.

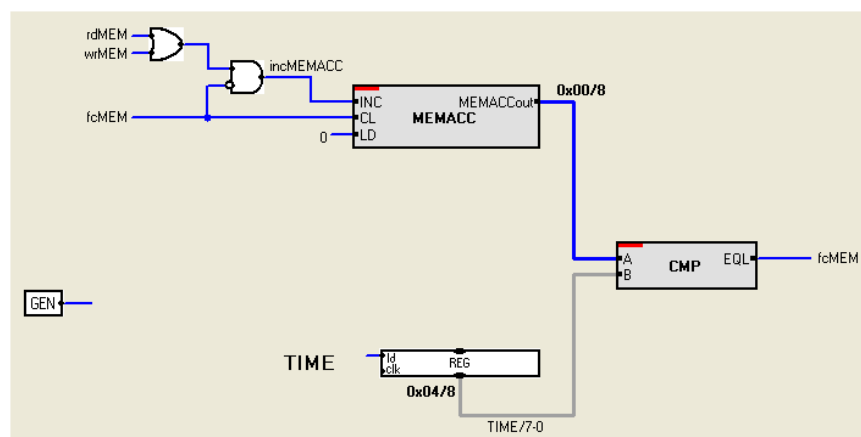
Kod ciklusa upisa u memoriju, procesor osim što postavlja podatak na adresne linije **ABUS**, on postavlja i podatak koji treba da bude upisan u memoriju na linije podataka **DBUS**. Dalje je ceo postupak veoma sličan sa procesom čitanja podataka osim što procesor ovde šalje signal upisa **WRBUS** umesto signala čitanja **RDBUS**.



Slika 31: *MEM* – operaciona jedinica

4.2.2. Upravljačka jedinica

Upravljačka jedinica se sastoji od kombinacione logike koja ima zadatak da generiše aktivnu vrednost signala **fMEM** koji govori o tome da je vreme za pristup memoriji isteklo i da se podatak, koji je učitao, može staviti na magistralu podataka. Vreme koje je potrebno memoriji da pročita podatak sa tražene lokacije se nalazi u registru **TIME**. Kada od strane procesora stigne zahtev za čitanje ili upis podatka u memoriju, na svaki signal takta se inkrementira sadržaj registra **MEMACC** (koji je na početku imao vrednost nula). Na svaki signal takta se takođe vrši provera da li je isteklo predviđeno vreme za čitanje ili upis podatka. To se postiže tako što se u komparatoru vrši provera da li registri **TIME** i **MEMACC** imaju jednake vrednosti. Ako imaju, vreme je isteklo, podatak je ili pročitan ili upisan u traženu lokaciju i generiše se aktivna vrednost signala **fMEM**. Aktiviranjem ovog signala se u operacionoj jedinici otvara bafer sa tri stanja čime se na magistralu podataka izbacuje traženi podatak ako je u pitanju bio ciklus čitanja podatka.



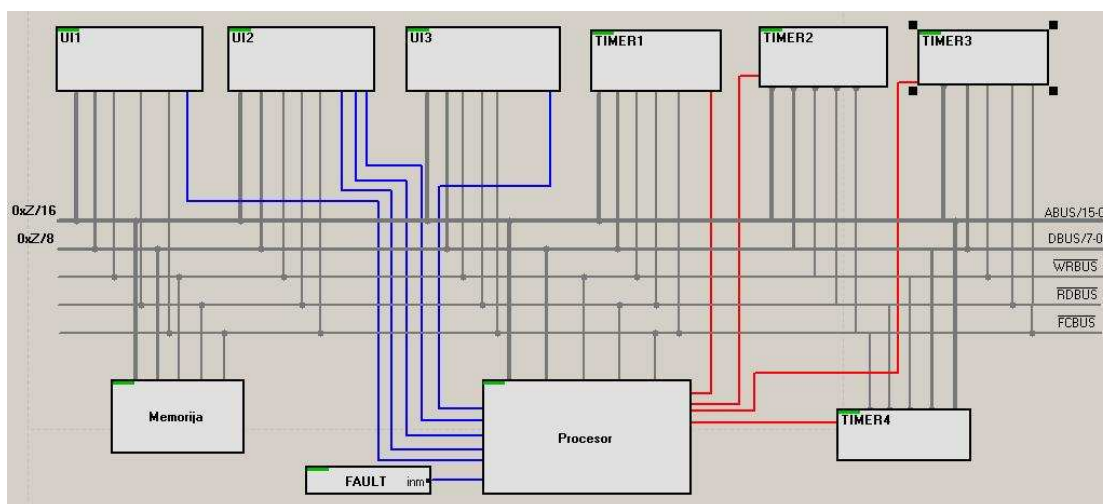
Slika 32: *MEM* – upravljačka jedinica

5. SIMULATOR U/I

U ovoj glavi daje se detaljan opis ulazno/izlaznih uređaja ovog računarskog sistema. Opis uključuje sve module i submodule sve do nivoa osnovnih logičkih kola, što omogućava lakše razumevanje rada ovog sistema i uvid u stanje svih relevantnih delova sistema u svakom trenutku. Simulator je izrađen u alatu IgoVSoDS, prema već projektovanom računarskom sistemu prikazanom u [1] uz saglasnost autora. Delovi teksta su preuzeti iz literature [1] uz saglasnost autora.

5.1. Glavna šema

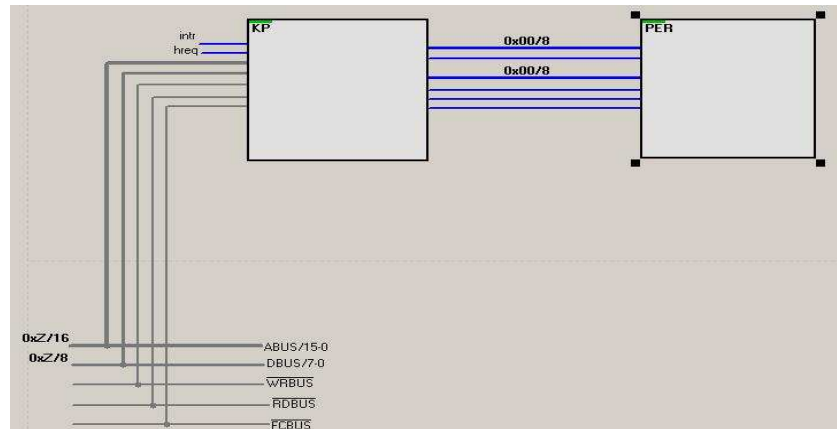
Na glavnoj šemi (slika 18) se vide osnovni delovi projektovanog računarskog sistema. To su memorija, procesor i ulazno/izlazne celine. Oni su povezani preko systemske magistrale, preko koje se odvija razmena podataka između registara procesora, memorije i registara kontrolera periferija. Kontroleri periferija su sa periferijama povezani direktnim vezama. Kontroleri periferija su sa procesorom povezani takođe linijama prekida, što im omogućava da izazovu prekid u procesoru u proizvoljnom trenutku.



Slika 33: Glavna šema

5.2. Ulazno/Izlazni uređaji

Ulazno/Izlazni uređaji se sastoje od periferije PER i kontrolera periferije KP kao na slici 19. U daljem tekstu biće dati opisi svih vrsta kontrolera periferije i same periferije. Postoje kontroleri periferije sa direktnim pristupom memoriji, bez direktnog pristupa memoriji i kontroler za generisanje impulsa.

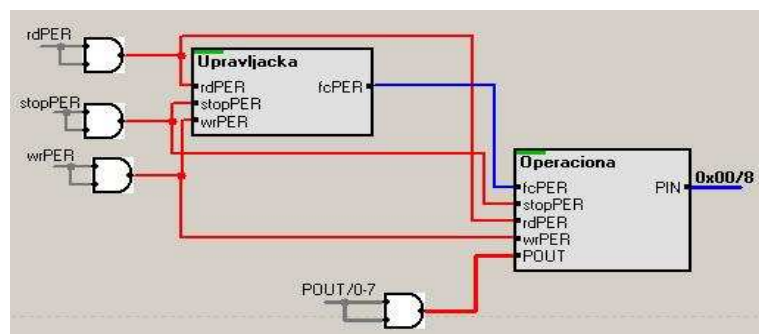


Slika 34: Struktura ulazno/izlaznog uređaja

5.3. Periferija

Periferija se sastoji iz operacione i upravljačke jedinice (slika 20). Operaciona jedinica se sastoji iz sekvencijalnih i kombinacionih mreža koje služe za pamćenje binarnih reči, izvršavanje mikroopreacija i generisanje signala logičkih uslova.

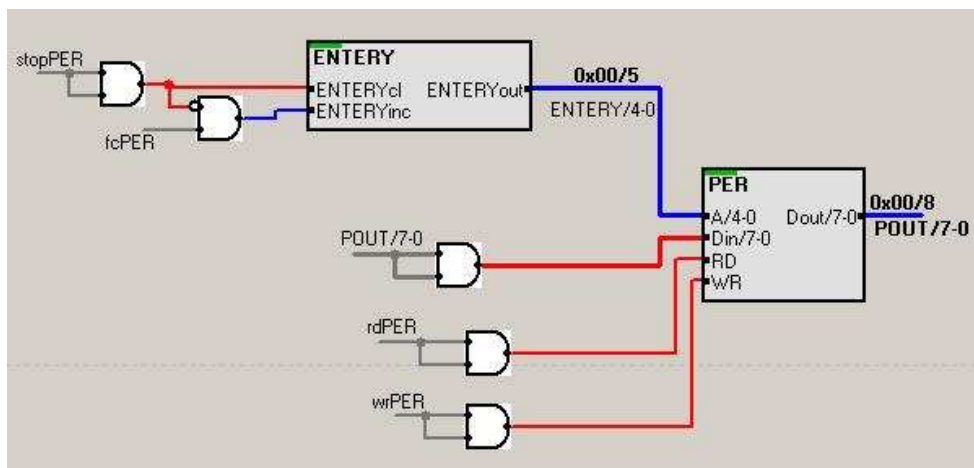
Upravljačka jedinica je skup mreža koje služe za generisanje upravljačkih signala po algoritmu generisanja upravljačkih signala operacione jedinice i vrednosti signala logičkih uslova.



Slika 35: Struktura periferije

5.3.1. Operaciona jedinica

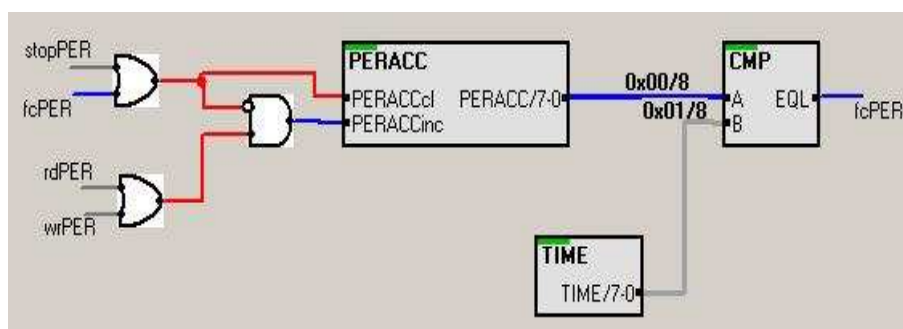
Operaciona jedinica sadrži brojač **ENTERY** i memoriju **PER**. Sadržaj brojača **ENTERY** se koristi za adresiranje memorijskih lokacija memorije **PER** prilikom čitanja ili upisa. Ovaj brojač se inkrementira da bi ukazivao na sledeću memorijsku lokaciju prilikom svakog čitanja ili se puni svim nulama kao priprema za početak sledećeg prenosa u zavisnosti od signala **fcPER** i **stopPER**. Memorija **PER** služi za simuliranje periferije određenog kapaciteta. Na njene adresne linije vodi se izlaz brojača **ENTERY**, na kontrolne linije **RD** i **WR** vode se upravljački signali **rdPER** i **wrPER** koje obezbeđuje kontroler periferije, izlazne i ulazne linije podataka se takođe vode na ulaz kontrolera periferije jer omogućuju razmenu podataka sa njim prilikom ulaza ili izlaza sa periferije. Struktura operacione jedinice prikazana je na slici 21.



Slika 36: Struktura operacione jedinice

5.3.2. Upravljačka jedinica

Upravljačka jedinica se sastoji od mikroprekidača **TIME**, brojača **PERACC**, komparatora i logičkih elemenata koji na osnovu upravljačkih signala kontrolera periferije **rdPER**, **wrPER** i **stopPER** i upravljačkog signala upravljačke jedinice **fcPER** generišu signale za upisivanje svih nula ili inkrementiranje brojača **PERACC**. Mikroprekidačem **TIME** simulira se vreme pristupa memoriji i u njega se upisuje broj taktova koliko treba da traje vreme pristupa memoriji. Brojačem **PERACC** se broje taktovi od početka pristupa memoriji prilikom čitanja ili upisa što se signalizira aktivnom vrednošću signala **rdPER** ili **wrPER** i kada postanu jednaki sadržaju mikroprekidača **TIME**, što se utvrđuje komparatorom, generiše se signal **fcPER** koji govori da je završen pristup memoriji. Kada je pristup memoriji završen ili kontroler periferije šalje signal o zaustavljanju periferije aktivnom vrednošću signala **stopPER** u brojač **PERACC** se upisuju sve nule radi pripreme za naredni pristup memoriji. Struktura upravljačke jedinice prikazana je na slici 22.



Slika 37: Struktura upravljačke jedinice

5.4. Kontroler periferije bez direktnog pristupa memoriji

Kontroler periferije bez direktnog pristupa memoriji se sastoji od operacione i upravljačke jedinice.

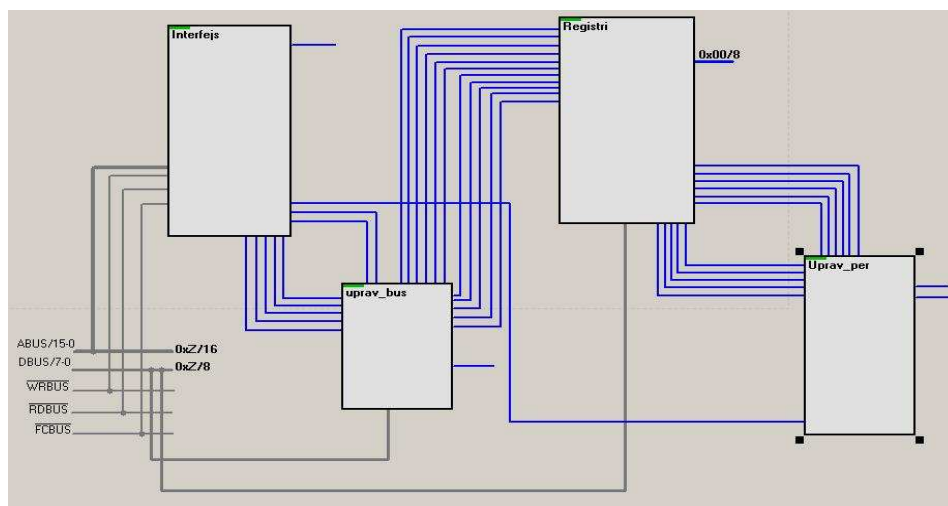
Operaciona jedinica je skup kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova.

Upravljačka jedinica se sastoji iz dva dela i to iz upravljačke jedinice magistrale **uprav_bus** i upravljačke jedinice periferije **uprav_per**.

5.4.1. Operaciona jedinica

Operaciona jedinica se sastoji iz blokova **'Registri'** i **'Interfejs'**. Blok **'Registri'** služi za čuvanje podataka, upravljačkih i statusnih informacija. Blok **'Interfejs'** služi za realizaciju ciklusa na magistrali u kojima je kontroler sluga i realizaciju prekida. Detaljna struktura i opis blokova operacione jedinice data je u sledeća dva poglavlja.

5.4.1.1. Blok **'Registri'**



Slika 38:Struktura operacione i upravljačke jedinice

Blok **'Registri'** sadrži registre **DR** i **DRAUX** zajedno sa multiplekserima **MP1** i **MP2**, kao i registre **CR** i **SR** i flip-flobove **WRDR** i **RDDR**.

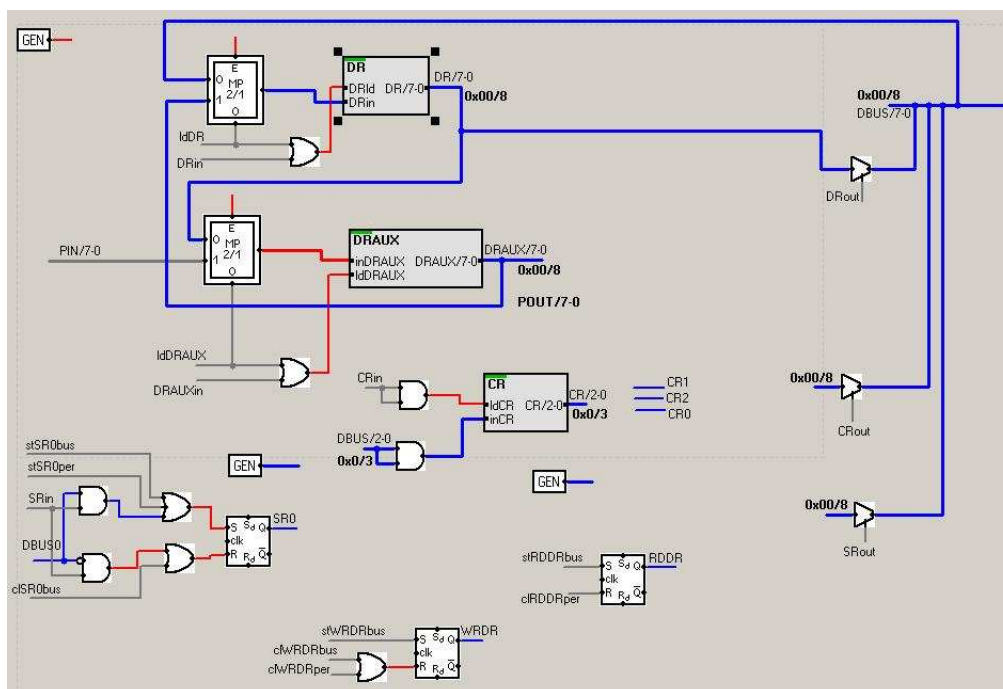
Registri **DR** i **DRAUX** zajedno sa multiplekserima **MP1** i **MP2** služe za prenos podataka iz memorije u periferiju i obrnuto. Pri prenosu podataka iz periferije u memoriju podataka je sa ulaznih linija **PIN** propušta kroz multiplekser **MP2** u registar **DRAUX** pa se iz njega kroz multiplekser **MP1** propušta u registar **DR**, da bi se iz njega po potrebi propustio kroz bafere sa tri stanja na linije magistrale podataka **DBUS**. Postojanje ova dva registra i ovakvim načinom prenosa podataka omogućava se da se istovremeno jedan podatak prenosi iz periferije u registar **DRAUX** i da se drugi podatak prenosi iz registra **DR** na magistralu. Naravno, omogućena je sinhronizacija stanja u registru **DR** i **DRAUX** da se ne bi dogodilo da se podatak koji još nije izašao na magistralu prepíše sledećim podatkom iz registra **DRAUX**. Pri prenosu podataka iz memorije u periferiju podataka se sa magistrale podataka **DBUS** prvo kroz multiplekser **MP1** propusti i upiše u registar **DR**, da bi se potom propustio kroz multiplekser **MP2** i upisao u registar **DRAUX** odakle se prenosi u periferiju po

potrebi. Ovakvim načinom prenosa i postojanjem registara **DR** i **DRAUX** omogućen je istovremeni prenos jednog podatka sa magistrale podataka **DBUS** u registar **DR** i drugog podatka iz registra **DRAUX** u periferiju. Međutim, omogućena je i sinhronizacija da se ne bi desilo da se podatak u registru **DR** koji još nije prenet u periferiju prepíše sledećim podatkom iz registra **DR**. Upis podataka u registar **DR** i propuštanje podataka kroz multiplekser **MP1** vrši se signalima **ldDR** i **DRin** dok se isti posao za registar **DRAUX** i multiplekser **MP2** obavlja signalima **ldDRAUX** i **DRAUXin**. Signali **ldDR**, **ldDRAUX** i **DRAUXin** dolaze iz upravljačke jedinice periferija, dok signal **DRin** dolazi iz upravljačke jedinice magistrale.

Registar **CR** predstavlja kontrolni registar kontrolera periferije koji sadrži start, u/i i bit prekida. Njegov sadržaj se kroz bafere sa tri stanja može propustiti na linije podataka magistrale **DBUS** proširen nulama na odgovarajuću dužinu uz pomoć signala **CRout** ili se može sa njih upisati uz pomoć signala **CRin**.

Registar **SR** je upravljački registar koji sadrži bit spremnosti. U njega se uz pomoć signala **SRin** može upisati vrednost sa magistrale podataka **DBUS**, a on se može takođe proširen nulama na odgovarajuću dužinu upisati na linije podataka magistrale **DBUS** kroz bafere sa tri stanja uz pomoć signala **SRout**. Pri prenosu podataka iz periferije u memoriju njegova aktivna vrednost je indikacija da je novi podatak prebačen iz registra **DRAUX** u registar **DR** i da odatle može biti prebačen u memoriju. Pri prenosu podataka iz memorije u periferiju njegova aktivna vrednost je indikacija da je podatak prebačen iz registra **DR** u registar **DRAUX** i da se u registar **DR** može upisati sledeća vrednost iz memorije.

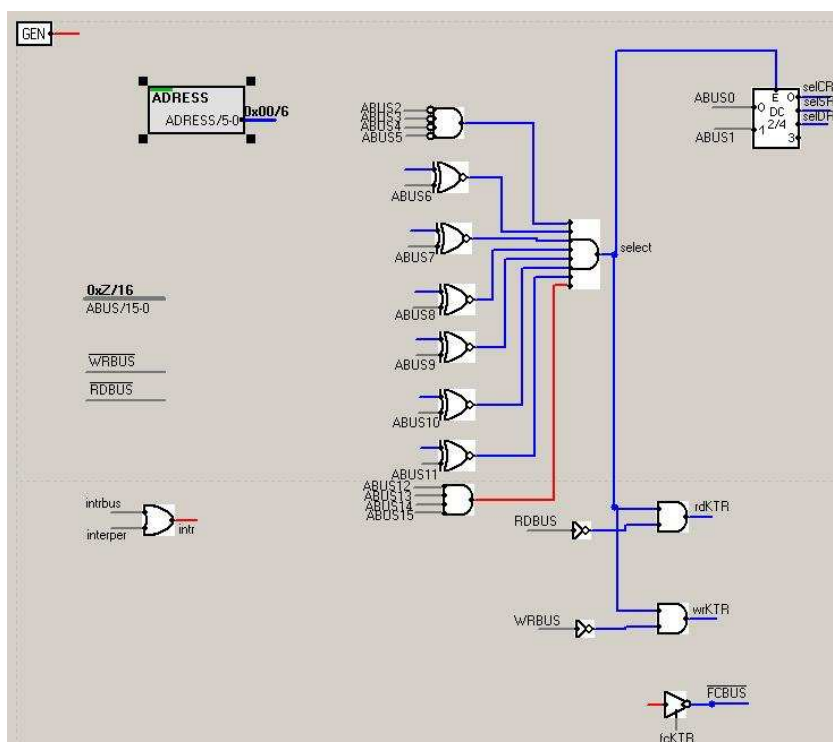
Flip-flopovi **RDDR** i **WRDR** služe za neophodnu sinhronizaciju podataka pri prenosu iz periferije u memoriju i iz memorije u periferiju. Flip-flop **RDDR** se koristi pri prenosu podataka iz periferije u memoriju i njegova aktivna vrednost je indikacija da novi podatak može da se prenese iz registra **DRAUX** u registar **DR**, a neaktivna da tekući podatak još uvek nije prenet iz registra **DR** u memoriju i da novi podatak ne može da se iz registra **DRAUX** prenese u registar **DR**. Flip-flop **WRDR** se koristi pri prenosu podataka iz memorije u periferiju i njegova aktivna vrednost je indikacija da se u registru **DR** nalazi podatak i da se on može preneti iz registra **DR** u registar **DRAUX**, a neaktivna vrednost da novi podatak još uvek nije prenet iz memorije u registar **DR** i da sadržaj registra **DR** ne može da se prenese u registar **DRAUX**.



Slika 39: Struktura bloka 'Registri'

5.4.1.2. Blok 'Interfejs'

Blok 'Interfejs' sadrži kombinacione i sekvencijalne mreže za realizaciju ciklusa na magistrali u kojima je kontroler sluga i generisanje prekida. To su mikroprekidač **ADRESS**, dekodler i kombinacione logičke mreže prikazani na slici 25.



Slika 40: Struktura bloka 'Interfejs'

Kombinacione i skevencijalne mreže za realizaciju ciklusa na magistrali u kojima je kontroler sluga formiraju signale **rdKTR** i **wrKTR** upravljačke jedinice **uprav_bus** i **FCBUS** magistrale. Signali **rdKTR** i **wrKTR** se formiraju na osnovu adresnih linija magistrale **ABUS** i **WRBUS** i **RDBUS** upravljačkih linija magistarle. Signal **FCBUS** se formira na osnovu signala **fcKTR** upravljačke jedinice **uprav_bus**.

Signali **rdKTR** i **wrKTR** imaju vrednosti upravljačkih signala **RDBUS** i **WRBUS** magistrale kada je signal **select** aktivan i neaktivne vrednosti kada je signal **select** neaktivan. Signal **rdKTR** ima aktivnu vrednost za vreme čitanja nekog od registara kontrolera, a signal **wrKTR** ima aktivnu vrednost za vreme upisa u neki od registara kontrolera.

Signal **select** ima aktivnu vrednost ako se na linijama adresne magistrale **ABUS** nalazi adresa iz opsega adresa dodeljenih registrima kontrolera periferije. Celokupni opseg adresa 0000h-FFFFh podeljen je na dva podopsega. Adrese od 0000h-EFFFh dodeljene su memoriji, a adrese od F000h-FFFFh dodeljene su registrima kontrolera periferija. Opseg adresa dodeljen registrima kontrolera periferija je predviđen za adresiranje najviše 64 kontrolera sa po najviše 64 registra. Pri tome prilikom adresiranja nekog od registara kontrolera sadržaj na adresnim linijama **ABUS**_{15...0} magistrale ima sledeću strukturu: bitovi **ABUS**_{15...12} su sve jedinice, bitovi **ABUS**_{11...6} određuju broj kontrolera i bitovi **ABUS**_{5...0} adresu registra unutar kontrolera. Broj kontrolera periferije za određeni kontroler periferije se postavlja mikroprekidačima **ADDRESS** na jednu od vrednosti u opsegu 0 do 63. Registrima **CR**, **SR** i **DR** su unutar opsega adresa datog kontrolera dodeljene adrese 0 do 2, pa bitovi **ABUS**_{5...2} mora da budu nule dok se njihovo pojedinačno adresiranje realizuje bitovima **ABUS**₁ i **ABUS**₀. Opisan način realizovanja adresiranja registara kontrolera obavlja se logičkim kolima i dekodeorom na čijem izlazu se generišu signali **selCR**, **selDR** i **selSR** potrebni za adresiranje odgovarajućih registara.

Kada kontroler treba u procesoru da izazove prekid, generiše se aktivna vrednost signala **intr**. Signala **intr** ima aktivnu vrednost ukoliko ili signal **intrbus** upravljačke jedinice **uprav_bus** ili signal **intrper** upravljačke jedinice **uprav_per** ima aktivnu vrednost.

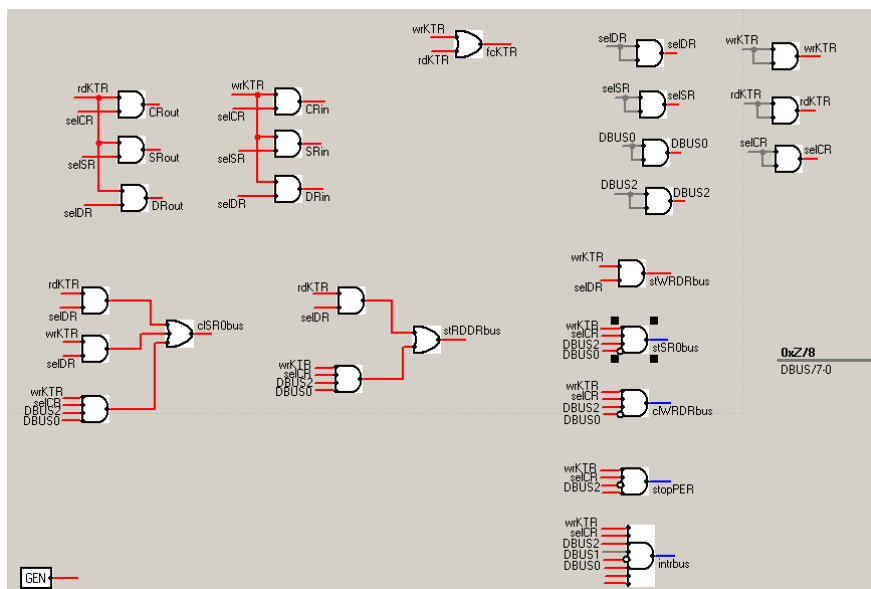
5.4.2. Upravljačka jedinica

Upravljačka jedinica se sastoji iz dva dela i to upravljačke jedinice magistrale **uprav_bus** i upravljačke jedinice periferije **uprav_per**. Ove dve upravljačke jedinice rade istovremeno i omogućavaju paralelan rad kontrolera periferije kao sluge u ciklusima na magistrali i u komunikaciji sa periferijom.

5.4.2.1. Upravljačka jedinica 'Uprav_bus'

Upravljačka jedinica se sastoji od logičkih elemenata koji na osnovu signala čitanja **rdKTR** i signala upisa **wrKTR** bloka '**Interfejs**' i signala logičkih uslova generišu sledeće signale: signal **fcKTR** bloka '**Interfejs**' završetka ili čitanja nekog od registara kontrolera **CR**, **SR** ili **DR** bloka '**Registri**' ili upisa u neki od ovih registara kontrolera, signale **CRout**, **SRout** i **DRout** bloka '**Registri**' čitanja registara kontrolera **CR**, **SR** ili **DR**, signale **CRin**, **SRin** i **DRin** bloka '**Registri**' upisa u registre kontrolera **CR**, **SR** ili **DR**, signale **clSR0bus** i **stSR0bus** bloka *registri*

postavljanja na neaktivnu i aktivnu vrednost, respektivno, bita spremnost SR_0 statusnog registra SR_0 kontrolera, signal **stRDDRbus** bloka '**Registri**' postavljanja na aktivnu vrednost flip-flopa RDDR i signale **stWRDR** i **clWRDR** bloka '**Registri**' postavljanja na aktivnu i neaktivnu vrednost, respektivno, flip-flopa WRDR, signal prekida **intrbus** bloka '**Interfejs**' i signal zaustavljanja periferije **stopPER** bloka '**Interfejs**'.



Slika 41: Struktura upravljačke jedinice **uprav_bus**

5.4.2.2. Upravljačka jedinica 'Uprav_per'

Upravljačka jedinica **uprav_per** se sastoji iz četiri bloka i to su: blok **generisanje nove vrednosti brojača koraka**, blok **brojač koraka**, blok **dekoder koraka** i blok **generisanje upravljačkih signala**.

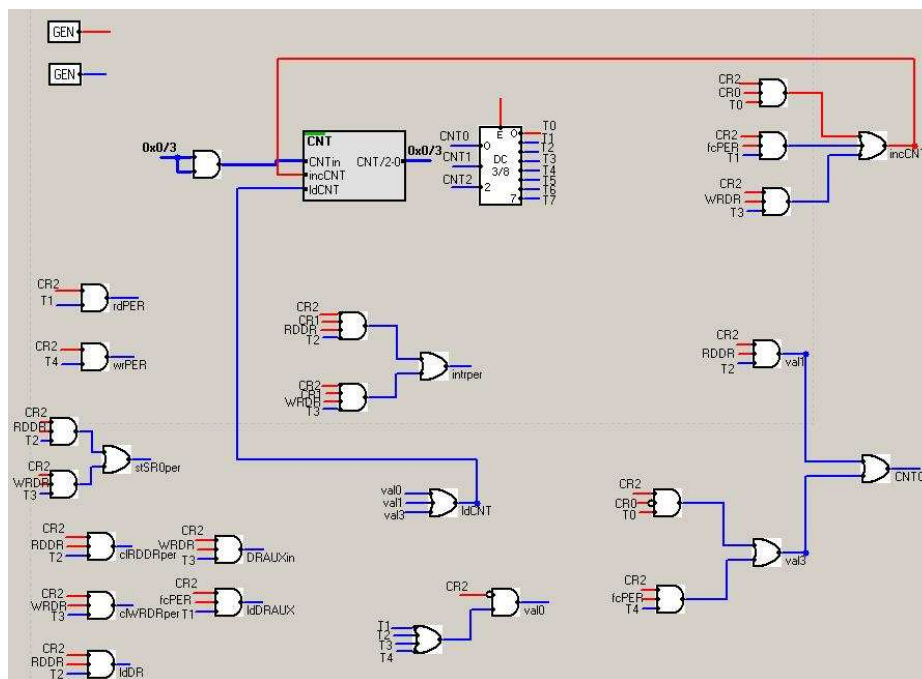
Blok **generisanje nove vrednosti brojača koraka** služi za generisanje vrednosti koju treba upisati u brojač koraka. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala.

Blok **brojač koraka** sadrži brojač koji svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač može da radi u režimima: režim inkrementiranja, režim skoka i režim mirovanja. U režimu inkrementiranja obezbeđuje se sekvencijalno generisanje upravljačkih signala iz algoritma generisanja upravljačkih signala. Ovaj režim se obezbeđuje aktivnom vrednošću signala **incCNT**, koji je inače neaktivan. U režimu skoka obezbeđuje se odstupanje od sekvencijalnog generisanja upravljačkih signala. Ovaj režim se obezbeđuje aktivnom vrednošću signala **ldCNT**, koji je inače neaktivan. Ako su neaktivna oba signala **incCNT** i **ldCNT** brojač koraka se nalazi u režimu čekanja u kome se ne menja vrednost brojača koraka jer se čeka na neki spoljašnji signal.

Blok **dekoder koraka** sadrži dekoder na čije ulaze se dovodi izlaz brojača koraka. Na izlazima dekodera se pojavljuju dekodovana stanja brojača kao signali koji odgovaraju svakom koraku iz algoritma generisanja upravljačkih signala.

Blok **generisanje upravljačkih signala** generiše tri grupe upravljačkih signala i to: upravljačke signale periferije **rdPER** i **wrPER**, upravljačke signale blokova

operacione jedinice **stSR0per**, **clRDDRper**, **clWRDRper**, **ldDR**, **DRAUXin** i **ldDRAUX** i upravljačke signale upravljačke jedinice **uprav_per** **ldCNT**, **incCNT**, **val0**, **val1** i **val3**.



Slika 42: Struktura upravljačke jedinice **uprav_per**

5.5. Kontroler periferije sa direktnim pristupom memoriji

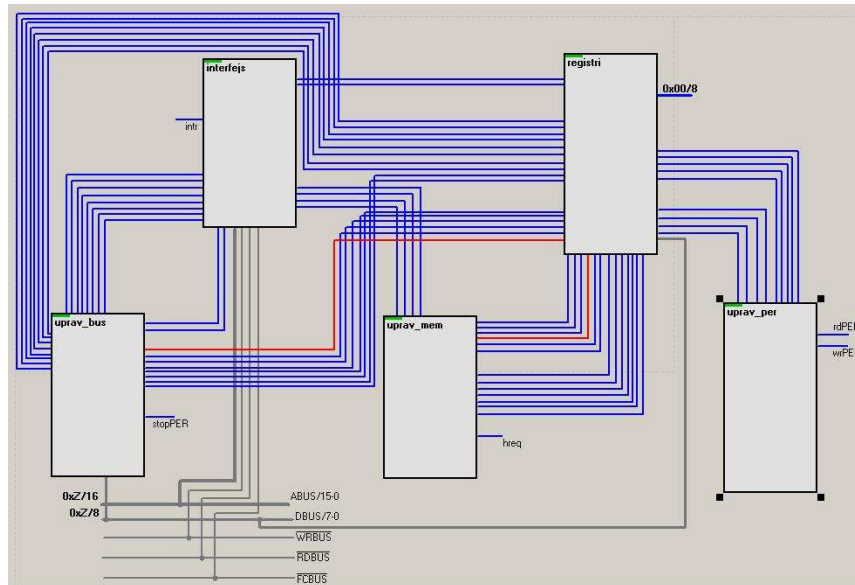
Kontroler periferije sa direktnim pristupom memoriji se sastoji iz operacione i upravljačke jedinice.

Operaciona jedinica se sastoji iz kombinacionih i sekvencijalnih prekidačkih mreža za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova.

Upravljačka jedinica se sastoji iz upravljačke jedinice periferije **uprav_per**, upravljačke jedinice magistrale **uprav_bus** i upravljačke jedinice memorije **uprav_mem**.

4.5.1. Operaciona jedinica

Operaciona jedinica se sastoji od bloka **'Registri'** i bloka **'Interfejs'**. Blok **'Registri'** služi za čuvanje podataka, upravljačkih i statusnih informacija. Blok **'Interfejs'** služi za realizaciju ciklusa na magistrali kada je kontroler sluga, realizaciju prekida, dobijanje dozvole korišćenja magistrale i realizaciju ciklusa na magistrali kada je kontroler gazda.



Slika 43: Struktura kontrolera sa direktnim pristupom memoriji

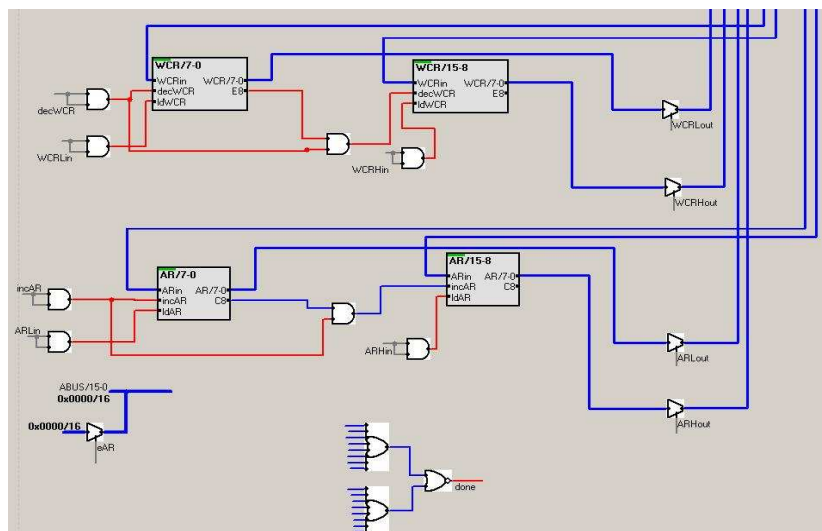
5.5.1.1. Blok 'Registri'

Blok '**Registri**' sadrži registre **DR** i **DRAUX** sa multiplekserima **MP1** i **MP2**, registre **CR** i **SR**, flip-flopove **WRDR** i **RDDR** i brojači **WCR** i **AR**.

Uloga registara **DR** i **DRAUX** sa multiplekserima **MP1** i **MP2**, kao i flip-flopova **WRDR** i **RDDR** i registara **CR** i **SR** su slične kao kod kontrolera bez direktnog pristupa memoriju tako da se u ovom poglavlju neće dublje ulaziti u njihovu funkciju. Novi elementi su brojači **WCR** i **AR** pa će njihova funkcija biti detaljnije opisana.

U brojaču **WCR** se čuva broj reči koje treba preneti pri zahtevu čitanja ili upisa. U brojač se može upisivati sa magistarle podataka, dakle programskim putem od strane procesora, i to u dva ciklusa na magistrali, jer je brojač dužine dva bajta. Ovo je omogućeno aktivnikom vrednostima signala **WCRLin** i **WCRHin**. Takođe sadržaj brojača se može postaviti na magistralu podataka kroz bafere sa tri stanja uz pomoć signala **WCRLout** i **WCRHout**. Pri prenosu svakog podatka iz bloka podataka brojač se dekrementira i time prikazuje trenutni broj podataka koje treba preneti. To se postiže aktivnom vrednošću signala **decWCR**. Vrednost ovog brojača različita od nule je indikacija da ceo blok podataka još uvek nije prenet, a kada on postane nula prenos je završen, što se vidi aktivnom vrednošću signala **done**.

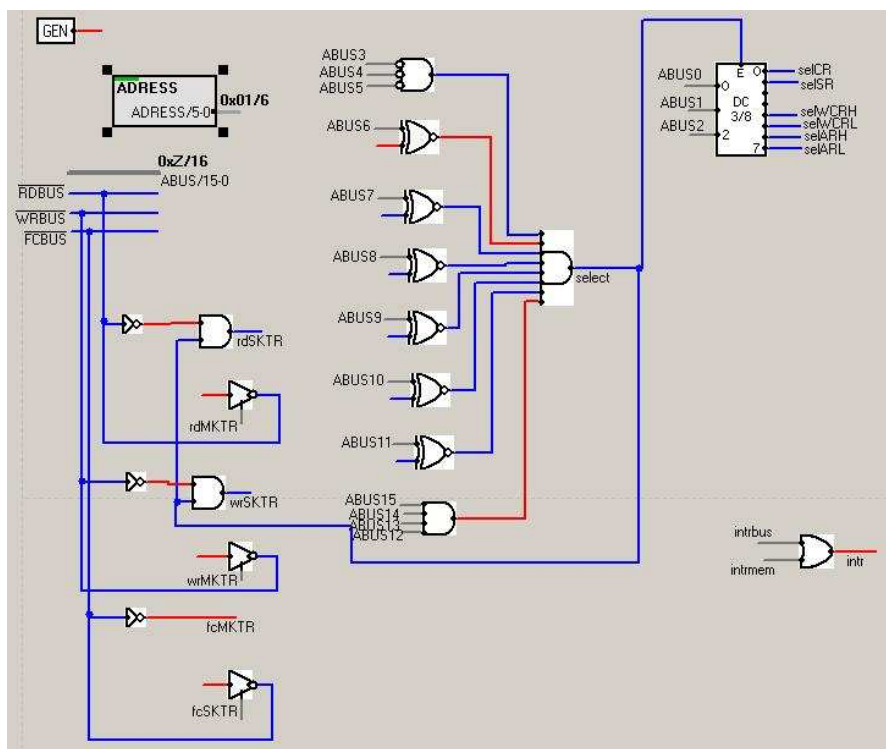
U brojaču **AR** se čuva dresa sledeće reči bloka podataka koja treba da se prenese. U brojač se može upisivati sa magistarle podataka i to u dva ciklusa na magistrali. Procesor to, dakle, može raditi programskim putem, što je omogućeno aktivnim vrednostima signala **ARHin** i **ARLin**. Takođe sadržaj brojača se može upisati na magistralu podataka kroz bafere sa tri stanja uz pomoć aktivne vrednosti signala **ARHout** i **ARLout**. A može se upisati i na adresnu magistralu što se koristi za adresiranje pri prenosu podataka i to aktivnom vrednošću signala **eAR**. Pri prenosu svake reči bloka podataka ovaj brojač se inkrementira da bi u svakom trenutku prikazivao adresi podatka kojeg treba sledećeg preneti. Ovo je omogućeno aktivnom vrednošću signala **incAR**.



Slika 44: Struktura bloka 'Registri'

5.5.1.2. Blok 'Interfejs'

Bloka 'Interfejs' sadrži kombinacione i sekvencijalne mreže za realizaciju ciklusa na magistrali kada je kontroler sluga, realizaciju prekida, dobijanje dozvole korišćenja magistrale i realizaciju ciklusa na magistrali kada je kontroler gazda. To su mikroprekidač ADDRESS, dekodler i kombinacione mreže prikazane na slici 30.



Slika 45: Struktura bloka 'Interfejs'

Kombinacione i sekvencijalne mreže za realizaciju ciklusa na magistrali u kojima je kontroler sluga formiraju signale **rdSKTR** i **wrSKTR** upravljačke jedinice

uprav_bus i **FCBUS** magistrale. Signali **rdSKTR** i **wrSKTR** se formiraju na osnovu adresnih linija magistrale **ABUS** i **WRBUS** i **RDBUS** upravljačkih linija magistarle. Signal **FCBUS** se formira na osnovu signala **fcSKTR** upravljačke jedinice **uprav_bus**.

Signali **rdSKTR** i **wrSKTR** imaju vrednosti upravljačkih signala **RDBUS** i **WRBUS** magistrale kada je signal **select** aktivan i neaktivne vrednosti kada je signal **select** neaktivan. Signal **rdSKTR** ima aktivnu vrednost za vreme čitanja nekog od registara kontrolera, a signal **wrSKTR** ima aktivnu vrednost za vreme upisa u neki od registara kontrolera.

Kombinacione i skevencijalne mreže za realizaciju ciklusa na magistrali u kojima je kontroler gazda formiraju signale **RDBUS** i **WRBUS** kontrolne magistrale i to na osnovu signala **rdMKTR** i **wrMKTR** upravljačke jedinice **uprav_mem**. Oni imaju aktivnu vrednost kada kontroler želi kako gazda na magistrali direktno da vrši ciklus čitanja ili upisa u memoriju.

Signal **select** ima aktivnu vrednost kada se na adresnoj magistarli nalazi adresa iz opsega adresa registara kontrolera. Način adresiranja registara kontrolera i upotreba linija adresne magistarle detaljno su opisani u poglavlju o bloku '**Interfejs**' kontrolera bez direktnog pristupa memoriji tako da ovde neće biti ponovo navođene. Razlika u odnosu na kontroler bez direktnog pristupa memoriji je što ovde postoji šest registara umesto dva tako da postoji i šest signala selekcije na izlazu dekodera umesto dva. Svaki od signala selekcije odgovara pojavljivanju tačno njegove adrese na adresnim linijama magisrale i to su signali **selCR**, **selSR**, **selWCRH**, **selWCRL**, **selARH** i **selARL**.

Kada kontroler treba u procesoru da izazove prekid, generiše se aktivna vrednost signala **intr**. Signala **intr** ima aktivnu vrednost ukoliko ili signal **intrbus** upravljačke jedinice **uprav_bus** ili signal **intrmem** upravljačke jedinice **uprav_mem** ima aktivnu vrednost.

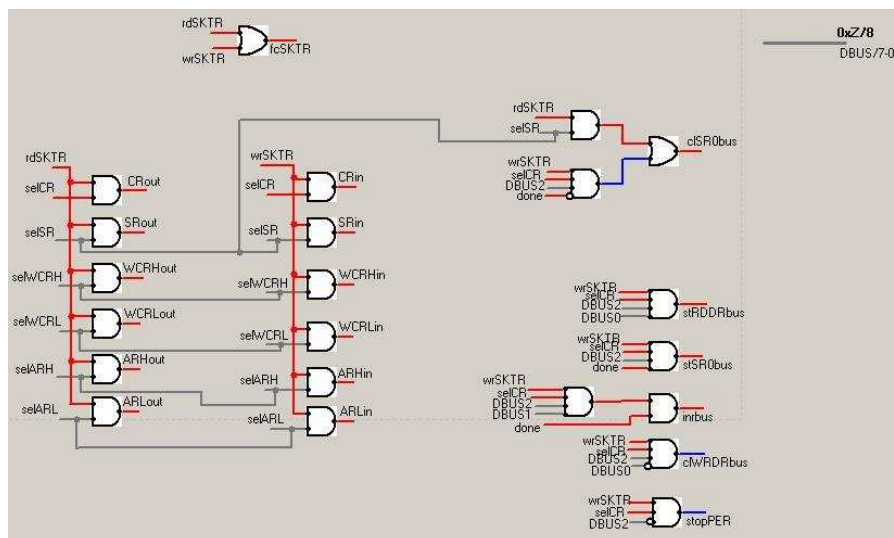
5.5.2. Upravljačka jedinica

Upravljačka jedinica se sastoji iz tri dela: upravljačke jedinice magistrale **uprav_bus**, upravljačke jedinice periferije **uprav_per** i upravljačke jedinice memorije **uprav_mem**.

5.5.2.1. Blok 'Uprav_bus'

Upravljačka jedinica magistrale se sastoji od logičkih elemenata koji na osnovu upravljačkih signala čitanja **rdSKTR** i upisa **wrSKTR** bloka '**Interfejs**' i signala logičkih uslova generišu sledeće upravljačke signale: signal **fcSKTR** bloka '**Interfejs**' završetka ili čitanja nekog od registara kontrolera **CR**, **SR**, **WCRH**, **WCRL**, **ARH** ili **ARL** bloka '**Registri**' ili upisa u neki od ovih registara kontrolera, signale **CRout**, **SRout**, **WCRHout**, **WCRLout**, **ARHout** ili **ARLout** bloka '**Registri**' čitanja registara kontrolera **CR**, **SR**, **WCRH**, **WCRL**, **ARH** ili **ARL**, signale **CRin**, **SRin**, **WCRHin**, **WCRLin**, **ARHin** ili **ARLin** bloka '**Registri**' upisa u registre kontrolera **CR**, **SR**, **WCRH**, **WCRL**, **ARH** ili **ARL**, signale **clSR0bus** i **stSR0bus** bloka '**Registri**' postavljanja na neaktivnu i aktivnu vrednost bita

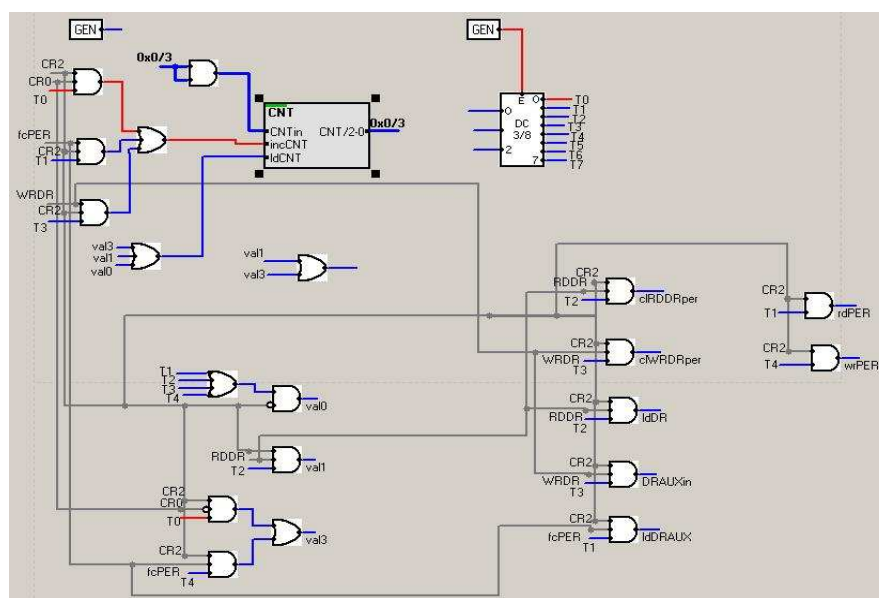
spremnosti statusnog registra kontrolera **SR**, signal **stRDDRbus** bloka '**Registri**' postavljanja na aktivnu vrednost flip-flopa **RDDR** i signal **clWRDR** bloka '**Registri**' postavljanja na neaktivnu vrednost flip-flopa **WRDR**, signal prekida **intrbus** bloka '**Interfejs**' i signal zaustavljanja periferije **stopPER** bloka '**Interfejs**'.



Slika 46: Struktura upravljačke jedinice magistrale

5.5.2.2. Blok 'Uprav_per'

Struktura upravljačke jedinice periferije je slična onoj kod kontrolera bez direktnog pristupa memoriji tako da ona neće ovde biti ponovo navedena, ali se ona može videti i uporediti sa slike 32.



Slika 47: Struktura upravljačke jedinice periferije

5.5.2.3. Blok 'Uprav_mem'

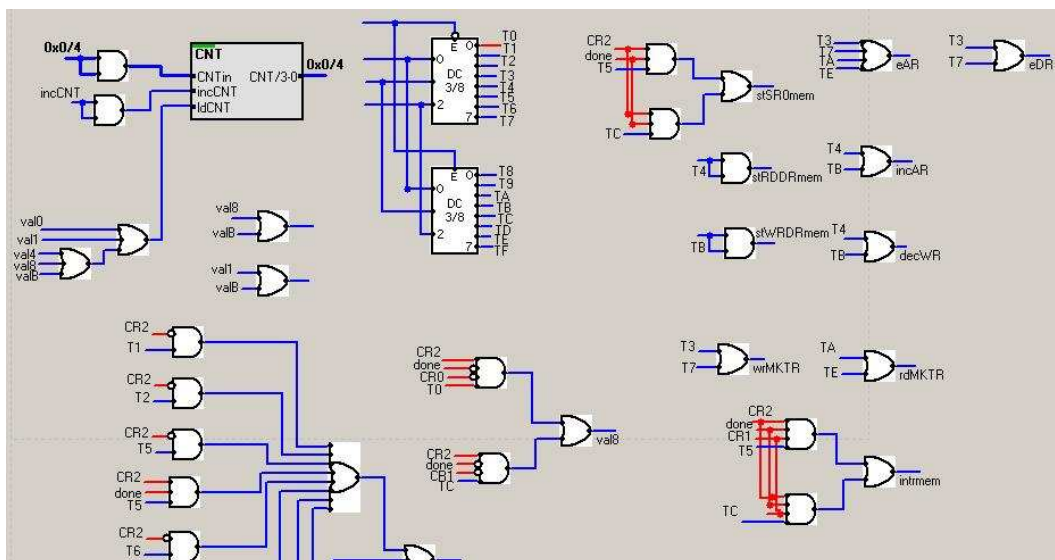
Upravljačka jedinica memorije ožičene realizacije se sastoji iz četiri bloka i to su: blok **generisanje nove vrednosti brojača koraka**, blok **brojač koraka**, blok **dekoder koraka** i blok **generisanje upravljačkih signala**.

Blok **generisanje nove vrednosti brojača koraka** služi za generisanje vrednosti koju treba upisati u brojač koraka. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala.

Blok **brojač koraka** sadrži brojač koji svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač može da radi u režimima: režim inkrementiranja, režim skoka i režim mirovanja. U režimu inkrementiranja obezbeđuje se sekvencijalno generisanje upravljačkih signala iz algoritma generisanja upravljačkih signala. Ovaj režim se obezbeđuje aktivnom vrednošću signala **incCNT**, koji je inače neaktivan. U režimu skoka obezbeđuje se odstupanje od sekvencijalnog generisanja upravljačkih signala. Ovaj režim se obezbeđuje aktivnom vrednošću signala **ldCNT**, koji je inače neaktivan. Ako su neaktivna oba signala **incCNT** i **ldCNT** brojač koraka se nalazi u režimu čekanja u kome se ne menja vrednost brojača koraka jer se čeka na neki spoljašnji signal.

Blok **dekoder koraka** sadrži dekodeer na čije ulaze se dovodi izlaz brojača koraka. Na izlazima dekodera se pojavljuju dekodovana stanja brojača kao signali koji odgovaraju svakom koraku iz algoritma generisanja upravljačkih signala.

Blok **generisanje upravljačkih signala** generiše tri grupe upravljačkih signala i to: upravljačke signale bloka '**Registri**' operacione jedinice **stSR0mem**, **stRDDRmem**, **stWRDRmem**, **eAR**, **eDR**, **incAR**, **decWR** upravljačke signale bloka '**Interfejs**' operacione jedinice **wrMKTR**, **rdMKTR** i **intrmem** i upravljačke signale upravljačke jedinice **uprav_mem** **ldCNT**, **incCNT**, **val0**, **val1**, **val4**, **val8** i **valB**.



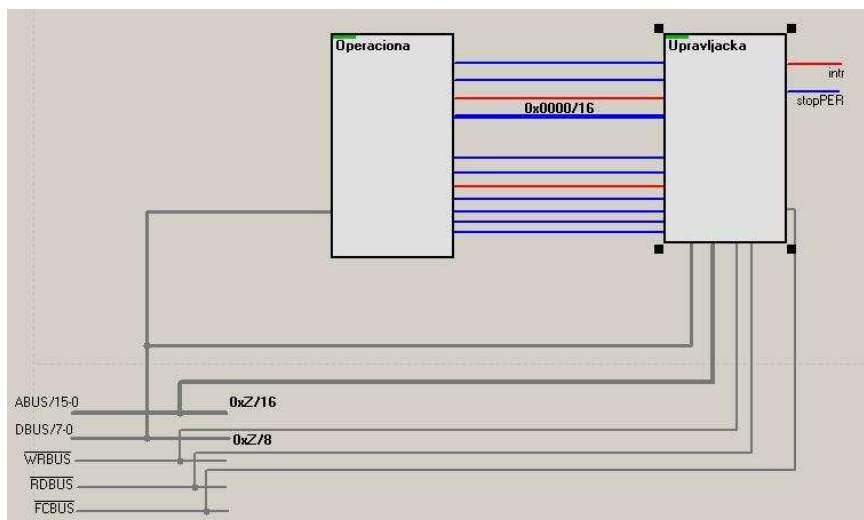
Slika 48: Struktura dela upravljačke jedinice memorije

5.6. Kontroler priferije za generisanje impulsa

Kontroler za generisanje impulsa se sastoji od operacione i upravljačke jedinice.

5.6.1. Operaciona jedinica

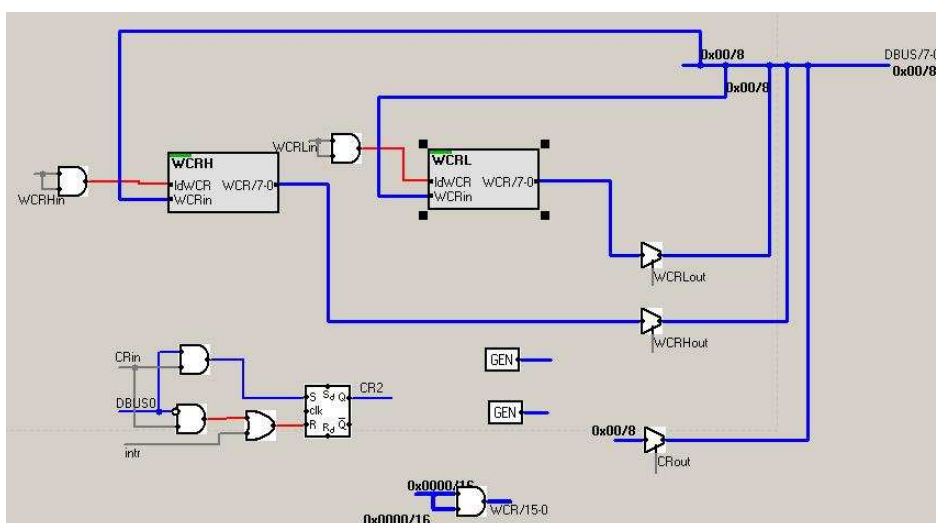
Operaciona jedinica sadrži registre **CR** i **WCR** i kombinacione i sekvencijalne prekidačke mreže za realizaciju ciklusa na magistrali u kojima je kontroler sluga i za generisanje prekida.



Slika 49: Struktura kontrolera za generisanje impulsa

Registar **CR** je upravljački registar koji sadrži strat bit. U njega se programskim putem sa magistrale podataka može upisati vrednost uz pomoć signala **CRin**. Njegova vrednost se proširena odgovarajućim brojem bita može upisati na magistralu podataka kroz bafere sa tri stanja uz pomoć signala **CRout**.

Registar **WCR** sadrži vrednost broja signala takta koja treba da protekne od startovanja kontrolera do generisanja prekida trajanja jedne periode signala takta. U ovaj registar se vrednost može upisati programskim putem i to posebno u viši, a posebno u niži bajt signalima **WCRHin** i **WCRLin**. Takođe, vrednost ovog registar se može upisati na magistralu podataka posebno viši i posebno niži bajt i to signalima **WCRHout** i **WCRLout**.



Slika 50: Struktura operacione jedinice

5.6.2. Upravljačka jedinica

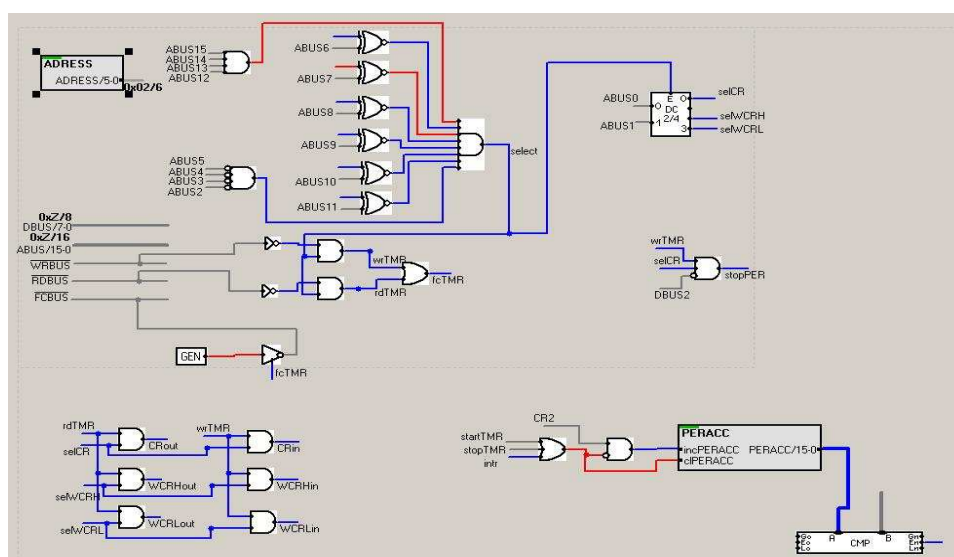
Upravljačka jedinica sadrži kombinacione i sekvencijalne mreže za realizaciju ciklusa na magistrali kada je kontroler sluga i realizaciju prekida. To su mikroprekidač **ADRESS**, dekodler, brojač **PERACC**, komparator i kombinacione mreže prikazane na slici 39.

Kombinacione i sekvencijalne mreže za realizaciju ciklusa na magistrali u kojima je kontroler sluga formiraju signale **rdTMR** i **wrTMR**. Signali **rdTMR** i **wrTMR** se formiraju na osnovu adresnih linija magistrale **ABUS** i **WRBUS** i **RDBUS** upravljačkih linija magistrale. Signal **FCBUS** se formira na osnovu signala **fcTMR** koji je formiran na osnovu signala **wrTMR** i **rdTMR**.

Signali **rdTMR** i **wrTMR** imaju vrednosti upravljačkih signala **RDBUS** i **WRBUS** magistrale kada je signal **select** aktivan i neaktivne vrednosti kada je signal **select** neaktivan. Signal **rdTMR** ima aktivnu vrednost za vreme čitanja nekog od registara kontrolera, a signal **wrTMR** ima aktivnu vrednost za vreme upisa u neki od registara kontrolera.

Signal **select** ima aktivnu vrednost kada se na adresnoj magistrali nalazi adresa iz opsega adresa registara kontrolera. Način adresiranja registara kontrolera i upotreba linija adresne magistrale detaljno su opisani u poglavlju o bloku '**Interfejs**' kontrolera bez direktnog pristupa memoriji tako da ovde neće biti ponovo navođene. Razlika u odnosu na kontroler bez direktnog pristupa memoriji je što ovde postoji tri registara umesto dva tako da postoji i tri signala selekcije na izlazu dekodera umesto dva. Svaki od signala selekcije odgovara pojavljivanju tačno njegove adrese na adresnim linijama magistrale i to su signali **selCR**, **selWCRH** i **selWCRL**.

Kada kontroler treba u procesoru da izazove prekid, generiše se aktivna vrednost signala **intr**. Signala **intr** ima aktivnu vrednost ukoliko vrednost brojača **PERACC** dostigne vrednost registra **WCR** operacione jedinice. Kada se kontroler startuje brojač **PERACC** se inkrementira u svakoj periodi signala takta i time ih broji kada dostigne vrednost upisanu u registar **WCR** generiše se prekid.



Slika 51: Struktura upravljačke jedinice

6. LABORATORIJA

U ovoj glavi dati su asemblerski programi koji ilustruju kompletnu arhitekturu sistema datog u [1] i sve njegove najvažnije funkcionalnosti. U prvoj glavi opisani su primeri programa koji ilustruju sve načine adresiranja i sve instrukcije procesora. U drugoj glavi opisani su primeri programa koji ilustruju tehnike programiranog ulaza i izlaza. U trećoj glavi opisani su primeri koji prikazuju mehanizam prekida.

6.1. Primeri instrukcija i načina adresiranja

Primeri programa koji opisuju raspoložive instrukcije i načine adresiranja kompletno su dati u prilogu. Pripremljeni su opisi svih aritmetičkih, logičkih, pomeračkih, instrukcija skoka, instrukcija skoka na potprogram i instrukcija za rad sa stekom. Uz programe dati su kraći opisi izvršavanja programa i pomoćna pitanja koja treba da sugerišu najvažnije principe upotrebljene pri izvršavanju programa.

6.1.1. Instrukcija ADD

U ovom poglavlju dat je detaljan opis izvršavanja programa koji ilustruje ilustruje aritmetičku operaciju sabiranja kao i neposredno, memorijsko direktno i PC relativno adresiranje. Opis je dat do nivoa aktivnih signala koji učestvuju u izvršavanju mikrooperacija. Ovde na početku dat je kratak opis izvršavanja programa.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 05h puni u akumulator. Sadržaj akumulatora se zatim u okviru izvršavanja instrukcije ADD, koja koristi memorijsko direktno adresiranje, sabira sa sadržajem memorijske lokacije 010Bh, koji ima vrednost 03h i rezultat se smešta u akumulator. Na kraju se instrukcijom STB, koja koristi PC relativnim adresiranje, rezultat prebacuje iz akumulatora u memorijsku lokaciju 010Bh.

6.1.1.1. Inicijalizacija programa

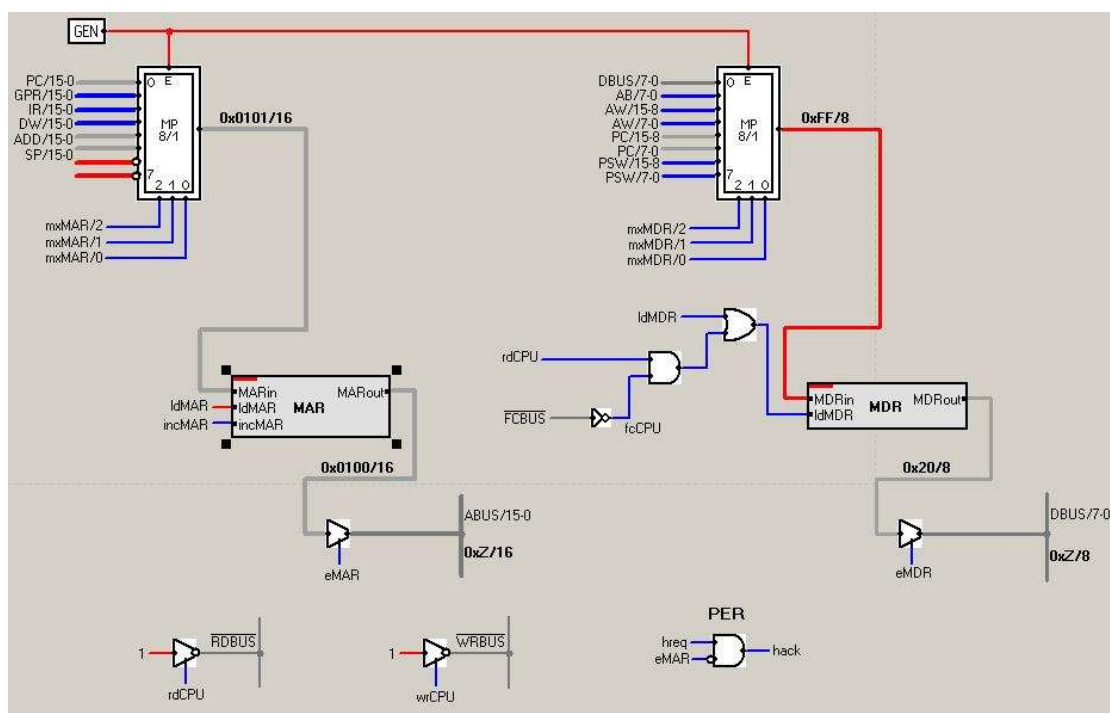
Pre početka izvršavanja programa neophodno je da odgovarajući registri budu inicijalizovani početnim vrednostima. U ovom slučaju inicijalizovani su registri IVTP, PC i SP i to vrednostima CC00h, C000h i B000h. Takođe mora se inicijalizovati deo memorije u kome treba da se nalaze instrukcije programa odgovarajućim kodiranim vrednostima. Ove vrednosti zajedno sa simboličkim instrukcijama programa date su na slici 52.

100	LDB imm(05)	Acc8 = 05	20 E0 05
103	ADD mem(010B)	Acc8 = Acc8 + mem[010B]	30 40 01 0B
107	STB pcrel(0000)	mem[PC+0000] =	22 C0 00 00

6.1.1.2. Izvršavanje programa

Pre nego što počne izvršavanje programa, proverava se vrednost signala **START** bloka *Operacije* koji vrednostima 0 i 1 ukazuje da li je procesor neaktivan ili aktivan, respektivno. Ukoliko je procesor aktivan, nastavlja se sa izvršavanjem programa.

Na početku procesor je u fazi čitanja instrukcije. Prvo se čita prvi bajt instrukcije i smešta u razrede $IR_{31...24}$ prihvatnog registra instrukcije, tako što se neaktivnim vrednostima signala **mxMAR₂**, **mxMAR₁** i **mxMAR₀** bloka *bus* sadržaj registra PC bloka *fetch* propušta kroz multiplekser MX1 i signalom **ldMAR** upisuje u adresni registar MAR bloka *bus*, kao što je prikazano na slici 53.

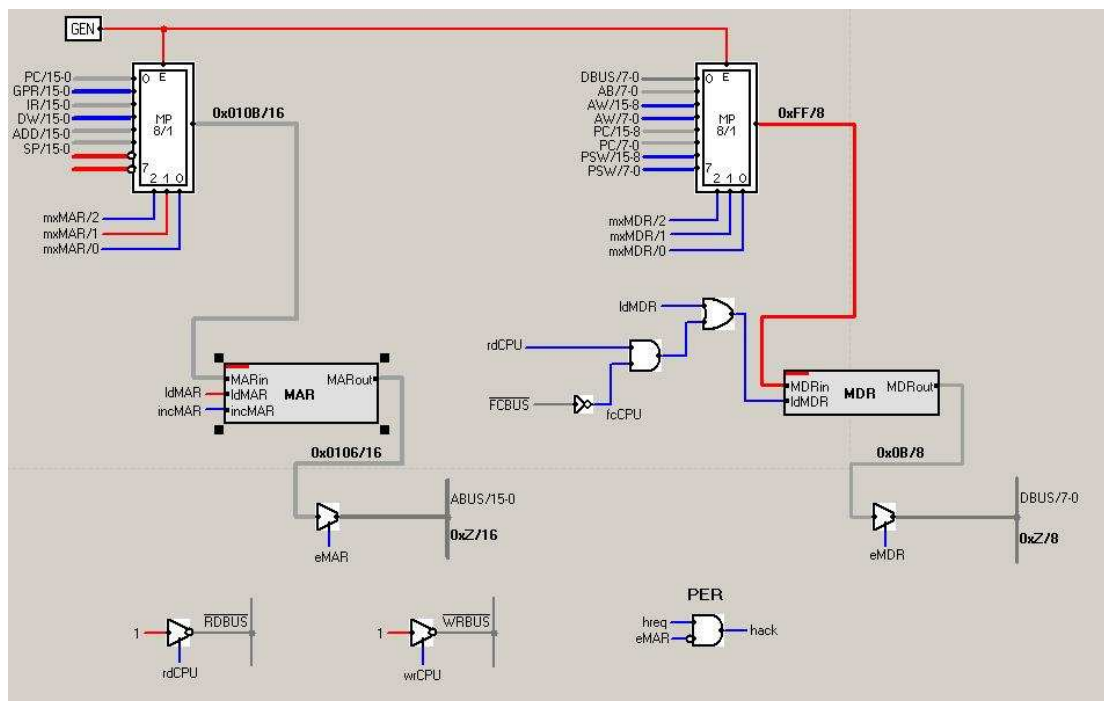


Slika 53. Upis adrese prvog bajta instrukcije u registar MAR

Pored toga, signalom **incPC** se sadržaj registra PC inkrementira. Sledi ciklus čitanja na magistrali koji se obezbeđuje aktivnim vrednostima signala **eMAR** i **rdCPU**. Pročitana vrednost se iz registra MDR prebacuje u registar IR_{31..24} aktivnom vrednošću signala **ldIRO**, kao što je prikazano na slici 54.

greška u načinu adresiranja. Dalje se proveravaju signali **l2_arlog** i **l2_brnch** koji su neaktivni jer je instrukcija dužine tri bajta. Dalje se na sličan način čita i treći bajt instrukcije i smešta u registar $IR_{15..8}$. Proverava se vrednost signala **l3_jump** koja je neaktivna i vrednost signala **l3_arlog** koja je aktivna. Prelazi se u fazu formiranja adrese i čitanja operanda. Kako je signal **imm** koji označava neposredno adresiranje aktivan, proverava se signal operacije LDW. On je neaktivan i aktivnim vrednostima signala **mxBB₁** i **ldBB** se 8-mo razredna neposredna veličina iz registra $IR_{15..8}$ prebacuje u registar BB. Kako je signal operacije LDB aktivan prelazi se u fazu njenog izvršavanja. U fazi izvršavanja ove instrukcije se operand specificiran adresnim delom instrukcije prebacuje u registar $AB_{7..0}$. Stoga se signalima **mxAB** i **ldAB** bloka *exec* sadržaj registra $BB_{7..0}$ propušta kroz multiplexer MX i upisuje u registar $AB_{7..0}$. Potom se u sledećem koraku signalima **ldN**, **ldZ**, **ldC** i **ldV** bloka *exec* upisuju u razrede PSWN i PSWZ programske statusne reči PSW vrednosti signala N i Z formirane na osnovu sadržaja upisanog u registar AB i u razrede PSWC i PSWV neaktivne vrednosti. Dalje se prelazi u fazu izvršavanja prekida. Kako nema prekida, odnosno signal **prekid** je neaktivan, odmah se prelazi u fazu čitanja sledeće instrukcije. Na ovaj način je instrukcijom LDB, koja koristi neposredno adresiranje, učitana je vrednost 05h u akumulator AB.

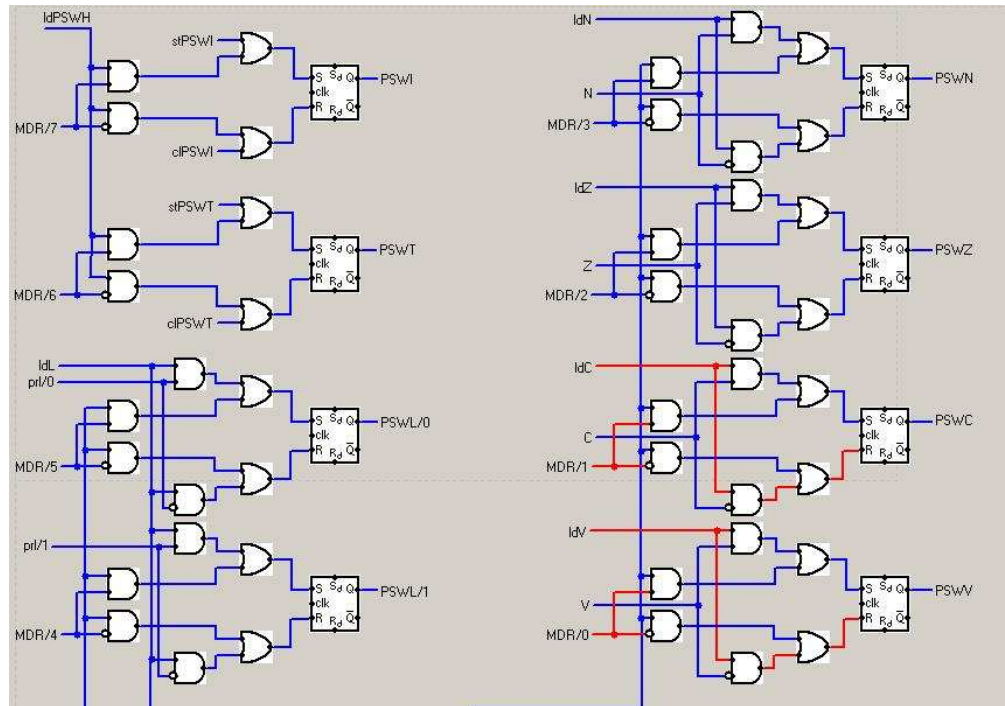
Na sličan način kao u slučaju prethodne instrukcije pročitaju se četiri bajta ove instrukcije i prelazi se u fazu formiranja adrese i čitanja operanda. Zbog aktivne vrednosti signala **memdir** radi se o memorijskom direktnom adresiranju. Signalom **mxMAR₁** se sadržaj registra $IRL_{15..0}$ bloka *registri* propušta kroz multiplexer bloka *bus* i signalom **ldMAR** upisuje u registar MAR bloka *bus*. Time se u registru $MAR_{15..0}$ nalazi adresa operanda za slučaj memorijskog direktnog adresiranja, kao što je prikazano na slici 56.



Slika 56. Upis adrese operanda u registar MAR

Dalje se prelazi na čitanje operanda. Jednim ciklusom na magistrali se vrednost operanda prebacuje u registar MDR. To se postiže aktivnim vrednostima

Istovremeno se signalima **ldC** i **ldV** bloka *exec* u razrede PSWC i PSWV programske statusne reči PSW upisuju vrednosti signala **C** i **V** bloka *exec* formirane na osnovu dobijenog rezultata na izlazima ALU, kao na slici 58.

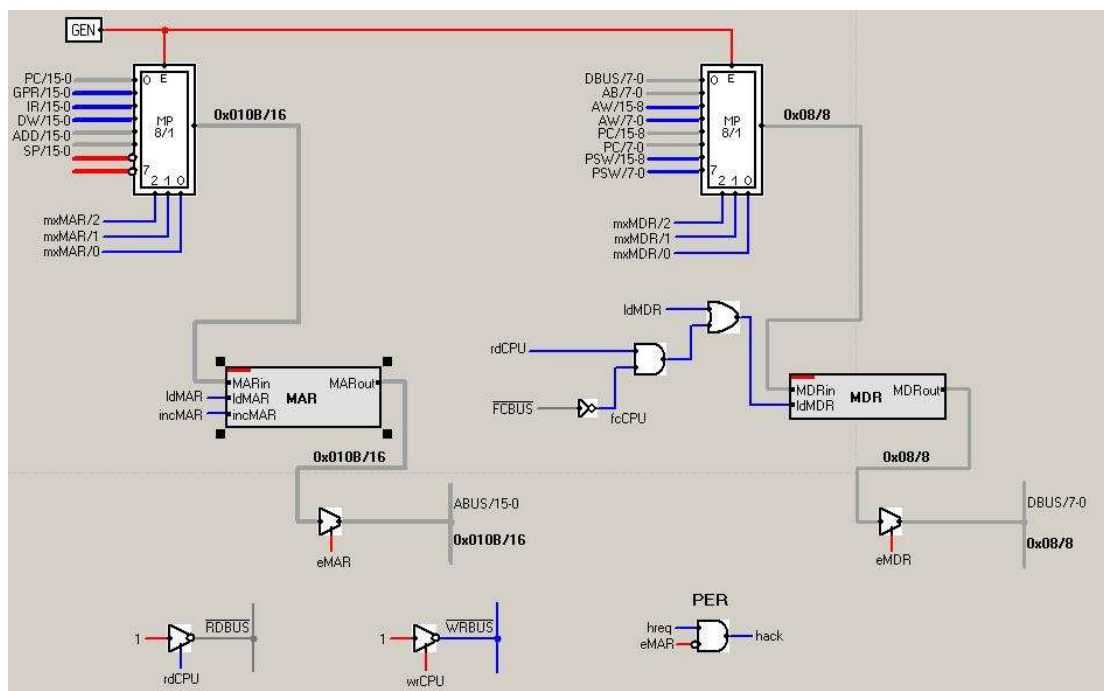


Slika 58. Upis u indikatore registra PSW

U sledećem koraku se signalima **IdN** i **IdZ** bloka *exec* u razrede PSWN i PSWZ programske statusne PSW reči upisuju vrednosti signala **N** i **Z** bloka *exec* formirane na osnovu sadržaja upisanog u registar AB i prelazi u fazu opsluživanje prekida. Kako nema prekida, odnosno signal **prekid** je neaktivan, odmah se prelazi u fazu čitanja sledeće instrukcije. Na ovaj način je instrukcijom ADD, koja koristi memorijsko direktno adresiranje, sabran sadržaj akumulatora sa sadržajem memorijske lokacije 010Bh i rezultat upisan u akumulator.

Na sličan način kao i prethodne dve instrukcije i za ovu instrukciju se pročitaju četiri bajta koliko je instrukcija dugačka i prelazi se na fazu formiranja adrese i čitanja operanda. Signal **pcpom** je aktivan što znači da je u instrukciji naznačen PC relativan način adresiranja. Signalima **mxADDA₁** i **mxADDB₀** se najpre kroz multipleksere i na ulaze sabirača ADD bloka *addr* propuštaju programski brojač PC i pomeraj iz IR_{15..0}, zatim se signalom **mxMAR₂** dobijeni sadržaj sa izlaza sabirača propušta kroz multiplekser bloka *bus* i na kraju signalom **IdMAR** upisuje u registar MAR bloka *bus*. Time se u registru MAR nalazi adresa operanda za slučaj PC relativnog adresiranja. Signal **store** bloka *addr* je aktivan, jer se radi o instrukciji STB. Za ovu instrukciju nema čitanja operanda i stoga se odmah prelazi na fazu izvršenja instrukcije. U fazi izvršavanja sadržaj registra AB bloka *exec* upisuje se u memorijsku lokaciju na adresi koja je formirana u fazi formiranje adrese i koja se nalazi i registru MAR bloka *bus*. Stoga se, najpre, signalima **mxMDR₀** i **IdMDR** bloka *bus* sadržaj registra AB propušta kroz multiplekser MX i upisuje u registar MDR. Pre upisa se proverava vrednost signala **hack** bloka *bus* koji vrednostima 1 i 0 ukazuje da li je processor prepustio magistralu kontroleru sa direktnim pristupom memoriji ili je magistrala u posedu procesora, respektivno. Dalje se realizuje upis. Signalima **eMAR** i **eMDR** bloka *bus* sadržaji registara MAR i MDR se puštaju na adresne linije ABUS i linije DBUS podataka magistrale **BUS** i signalom **wrCPU** formira aktivna vrednost signala

na upravljačkoj liniji $\overline{\text{WRBUS}}$ magistrale **BUS**, čime se u memoriji **MEM** startuje operacija upisa, kao što je prikazano na slici 59.



Slika 59. Upis u memoriju

Čeka se onoliko perioda signala takta koliko je neophodno da se upis realizuje. Po završenom upisu memorija **MEM** generiše aktivnu vrednost signala na upravljačkoj liniji **FCBUS** magistrale **BUS**. Proverava se vrednost signala **fcCPU** bloka *bus* koji vrednostima 0 i 1 ukazuje da upis nije završen i da je upis završen, respektivno. Po završetku upisa prelazi se na fazu opsluživanja prekida. Kako nema prekida, odnosno signal **prekid** je neaktivan, i nema više instrukcija programa izvršavanje programa se završava. I ovom poslednjom instrukcijom STB, koja koristi PC relativno adresiranje, je sadržaj akumulatora prebačen u memorijsku lokaciju sa adresom 010Bh.

6.1.2. Instrukcije JSR i RTS

U ovom poglavlju dat je detaljan opis izvršavanja programa koji ilustruje instrukcije skoka i povratka iz potprograma. Opis je dat do nivoa aktivnih signala koji učestvuju u izvršavanju mikrooperacija. Ovde, na početku, dat je kratak opis izvršavanja programa.

U programu se u okviru izvršavanja instrukcije JSR skače na potprogram na adresi 0295h, a u potprogramu se u okviru izvršavanja instrukcije RTS vraća nazad na adresu 0293h gde se nastavlja izvršavanje programa.

6.1.2.1. Inicijalizacija programa

Pre početka izvršavanja programa neophodno je da odgovarajući registri budu inicijalizovani početnim vrednostima. U ovom slučaju inicijalizovani su registri IVTP, PC i SP i to vrednostima CC00h, C000h i B000h. Takođe mora se inicijalizovati deo

memorije u kome treba da se nalaze instrukcije programa odgovarajućim kodiranim vrednostima. Ove vrednosti zajedno sa simboličkim instrukcijama programa date su na slici 53.

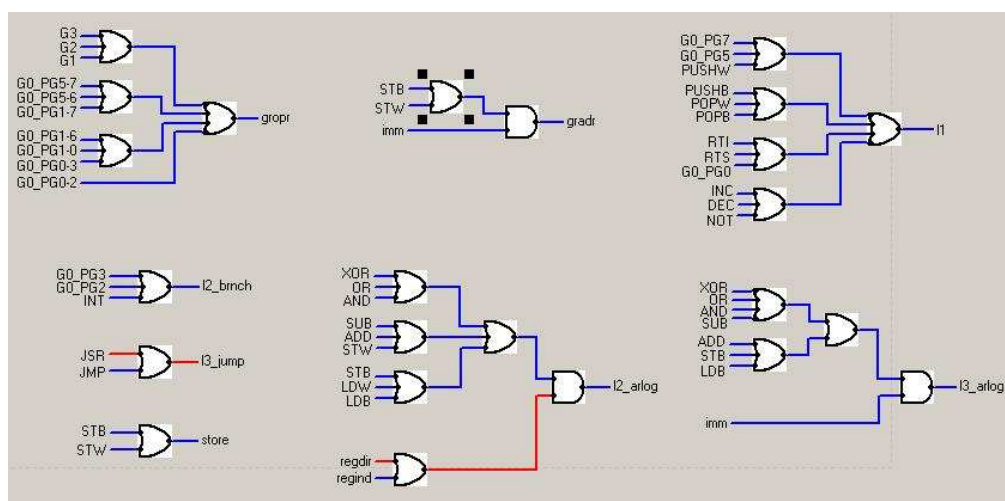
290	JSR imm(0295)	0A 02 05
293		
294		
295	RTS	0B

Slika 60. instrukcije programa

6.1.2.2. Izvršavanje programa

Pre nego što počne izvršavanje programa, proverava se vrednost signala **START** bloka *Operacije* koji vrednostima 0 i 1 ukazuje da li je procesor neaktivan ili aktivan, respektivno. Ukoliko je procesor aktivan, nastavlja se sa izvršavanjem programa.

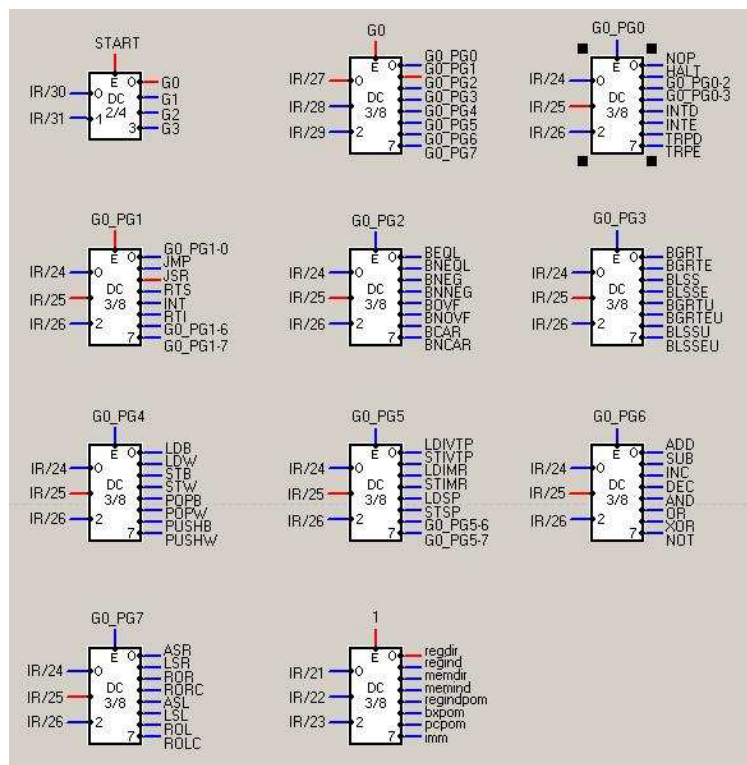
Na početku programa procesor je u fazi čitanja instrukcije. Kao u prethodnoj instrukciji najpre se pročitaju tri bajta koliko ih instrukcija ima. Pri pročitanoj trećem bajtu instrukcije signal **I3_jump** ima aktivnu vrednost, jer se radi o instrukciji skoka, kao što je prikazano na slici 61.



Slika 61. Vrednost signala **I3_jump**

Kako izvršenje ove instrukcije ne obuhvata faze formiranja adrese i čitanja operanda prelazi se direktno na fazu izvršenja instrukcije. Signal **JSR** ima aktivnu vrednost, jer se radi o instrukciji skoka na potprogram, kao što je prikazano na slici 62. U fazi izvršavanja ove instrukcije se realizuje skok na potprogram tako što se prvo na stek stavi tekući sadržaj programskog brojača i zatim realizuje bezuslovni skok na adresu koja je data u samoj instrukciji. Na stek se stavlja prvo viši a zatim i niži bajt registra $PC_{15...0}$. Stoga se najpre signalom **incSP** vrši inkrementiranje registra $SP_{15...0}$ bloka *addr*. Zatim se signalima **mxMAR₂**, **mxMAR₀** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i signalima **mxMDR₂** i **ldMDR** sadržaj višeg bajta registra $PC_{15...8}$ propušta kroz multiplekser MX2 i upisuje u registar $MDR_{7...0}$. Upis se realizuje na sličan način kao i upis rezultata u prethodnom primeru instrukcije ADD. Potom se signalom **incSP** vrši

inkrementiranje registra $SP_{15...0}$. Zatim se u signalima **mxMAR₂**, **mxMAR₀** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplexer MX1 bloka *bus* i upisuje u registar $MAR_{15...0}$ i signalima **mxMDR₂**, **mxMDR₀**, **ldMDR** sadržaj nižeg bajta registra $PC_{7...0}$ propušta kroz multiplexer MX2 i upisuje u registar $MDR_{7...0}$.



Slika 62. Vrednost signala operacije **JSR**

Upis se realizuje na isti način kao što se to radi prilikom upisa višeg bajta. Na kraju se sadržaj registra $IR_{23...8}$ bloka *fetch* koji predstavlja adresu skoka signalima **mxPC₁** i **ldPC** propušta kroz multiplexer MX bloka *fetch* i upisuje u registar $PC_{15...0}$ i bezuslovno prelazi fazu opsluživanje prekida. Upis u registar PC pikazan je na slici 63. Kako signal **prekid** ima neaktivnu vrednost, prekida nema, pa se prelazi na fazu čitanja sledeće instrukcije.

Potom se u signalima **mxMAR₂**, **mxMAR₀** i **ldMAR** sadržaj registra SP_{15...0} propušta kroz multiplekser MX1 i upisuje u registar MAR_{15...0}. Pored toga se i signalom **decSP** dekrementira sadržaj registra SP_{15...0}. Čitanje se realizuje na isti načina kao što se to radi kada se čita niži bajt. Na kraju se signalom **ldDWH** sadržaj registra MDR_{7...0} upisuje u viši bajt registar DW_{15...8}. Time je 16-to bitna vrednost skinuta sa steka i smeštena u registar DW_{15...0}. Konačno se sadržaj registra DW_{15...0} propušta kroz multiplekser MX bloka *fetch* i signalom **ldPC** upisuje u registar PC_{15...0} i bezuslovno prelazi na fazu opsluživanje prekida. Kako signal **prekid** ima neaktivnu vrednost, prekida nema, pa se prelazi na fazu čitanja sledeće instrukcije. Kako je ovo poslednja instrukcija programa time se završava i opis njegovog izvršavanja.

6.2.Primeri mehanizma prekida

Primeri programa koji opisuju sve mogućnosti mehanizma prekida kompletno su dati u prologu. Pripremljeni su opisi opsluživanja prekida i povratka iz prekidne rutine, maskiranja svih maskirajućih prekida, selektivnog maskiranja maskirajućih prekida, prioriteta maskirajućih prekida, režim rada prekid posle svake instrukcije, korišćenje instrukcije za programsko izazivanje prekida INT, gnežđenje prekida i prekid zbog greške u načinu adresiranja. Uz programe dati su kraći opisi izvršavanja programa i pomoćna pitanja koja treba da sugerišu najvažnije principe upotrebljene pri izvršavanju programa.

6.2.1. Prekid usled greške u načinu adresiranja

U sledećem poglavlju dat je detaljan opis izvršavanja programa koji ilustruje opsluživanje prekida usled greške u načinu adresiranja. Opis je dat do nivoa aktivnih signala koji učestvuju u izvršavanju mikrooperacija.

Ovde je dat i karatak opis izvršavanja programa. U programu se u tabelu prekidnih rutina u ulaz 2 koji odgovara prekidu usled greške u načinu adresiranja prvo upisuje adresa odgovarajuće prekidne rutine 0820. Dalje se instrukcijom STB sa neposrednim odredištem izaziva greška u načinu adresiranja i skače se u prekidnu rutinu čija je adresa upisana u tabeli.

6.2.1.1. Inicijalizacija programa

Pre početka izvršavanja programa neophodno je da odgovarajući registri budu inicijalizovani početnim vrednostima. U ovom slučaju inicijalizovani su registri IVTP, PC i SP i to vrednostima CC00h, C000h i B000h. Takođe mora se inicijalizovati deo memorije u kome treba da se nalaze instrukcije programa odgovarajućim kodiranim vrednostima. Ove vrednosti zajedno sa simboličkim instrukcijama programa date su na slikama 65. i 66.

810	LDW imm(0820)	Acc16 = 0820h	21 E0 08 20
814	STW mem(CC04)	mem[CC04] = 08h, mem[CC05] = 20h	23 40 CC 04
818	STB imm(01)	STB imm(01)	22 E0 01

Slika 65. Glavni program

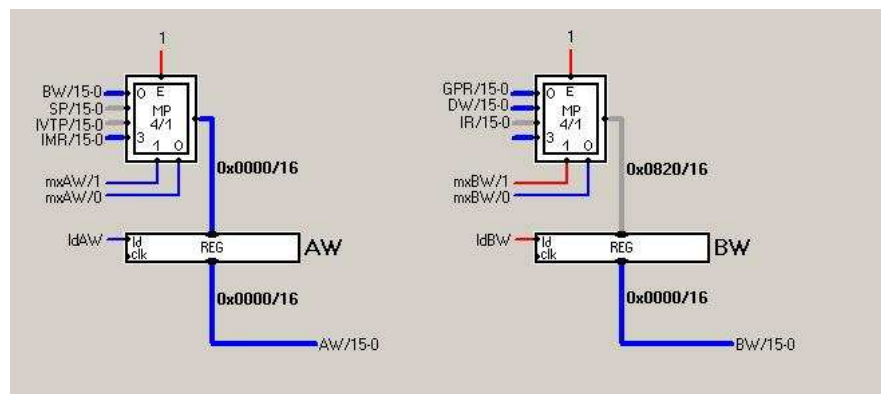
820	RTI	0D
-----	-----	----

Slika 66. Prekidna rutina

6.2.1.2. Izvršavanje programa

Pre nego što počne izvršavanje programa, proverava se vrednost signala **START** bloka *Operacije* koji vrednostima 0 i 1 ukazuje da li je procesor neaktivan ili aktivan, respektivno. Ukoliko je procesor aktivan, nastavlja se sa izvršavanjem programa.

Na početku programa procesor je u fazi čitanja instrukcije. Kao i kod prethodnih programa pročitaju se četiri bajta instrukcije i prelazi se u fazu formiranja adrese i čitanja operanda. Kako je aktivan signal **imm** instrukcija koristi neposredno adresiranje. Kako je signal **LDW** aktivan radi se o instrukciji LDW i signalima **mxBW₁** i **ldBW** 16-to razredna neposredna veličina koja se nalazi u razredima **IR_{15...0}** propušta kroz multiplexer i upisuje u registar BW, kao što je prilazano na slici 67.

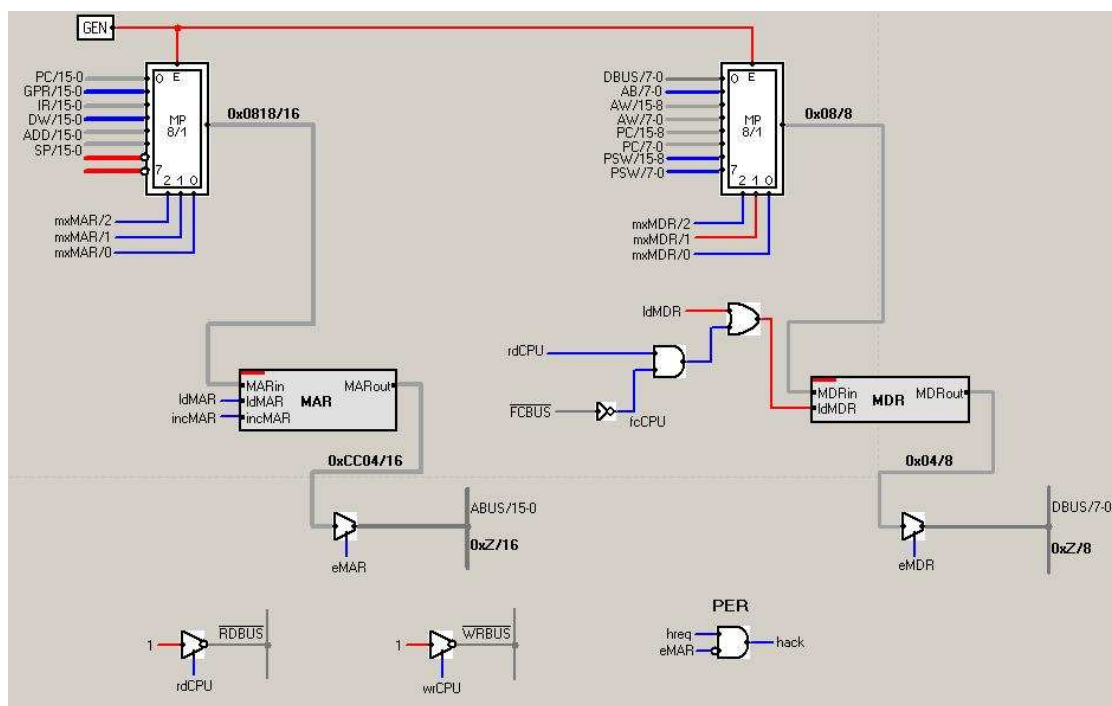


Slika 67. Upis podataka u registra BW

Nastavlja se u fazu izvršavanja operacije LDW. Ovde se signalom **ldAW** bloka *exec* sadržaj registra BW i upisuje u registar AW i prelazi na fazu opsluživanje prekida. Kako nema prekida, odnosno signal **prekid** je neaktivan nastavlja se sa izvršavanjem sledeće instrukcije. Ovom instrukcijom LDW, koja koristi neposredno adresiranje, učitana je vrednost 0820h u akumulator AW. Ova vrednost predstavlja adresu prekidne rutine, koja će biti upotrebljena za upisivanje u tabelu prekidnih rutina.

Dalje se pročitaju četiri bajta instrukcije i prelazi na fazu formiranja adrese i čitanja operanda. Kako je aktivan signal **memdir** instrukcija koristi memorijsko direktno adresiranje. Signalom **mxMAR₁** se sadržaj registra **IRL_{15...0}** bloka *registri* propušta kroz multiplexer bloka *bus* i signalom **ldMAR** upisuje u registar **MAR_{15...0}**

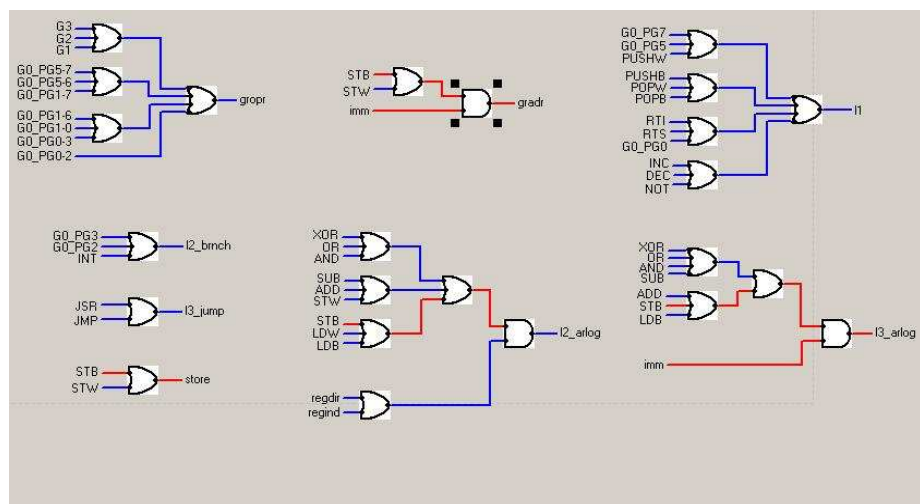
bloka *bus*. Time se u registru $MAR_{15...0}$ nalazi adresa operanda za slučaj memorijskog direktnog adresiranja. Kako je signal **store** aktivan nema čitanja operanda i odmah se prelazi u fazu izvršenja instrukcije. Signal STW je aktivan takođe, tako da se izvršava instrukcija STW. U fazi izvršavanja se, najpre, sadržaj višeg bajta a zatim i nižeg bajta registra AW bloka *exec* upisuje u dve susedne memorijske lokacije počev od memorijske lokacije čija je adresa formirana u fazi formiranje adrese i koja se nalazi i registru MAR bloka *bus*. Stoga se, najpre, signalima **mxMDR₁** i **ldMDR** bloka *bus* sadržaj registra $AW_{15..8}$ propušta kroz multiplekser MX i upisuje u registar MDR, kao što je prikazano na slici 68.



Slika 68. Upis podatka u registar MDR

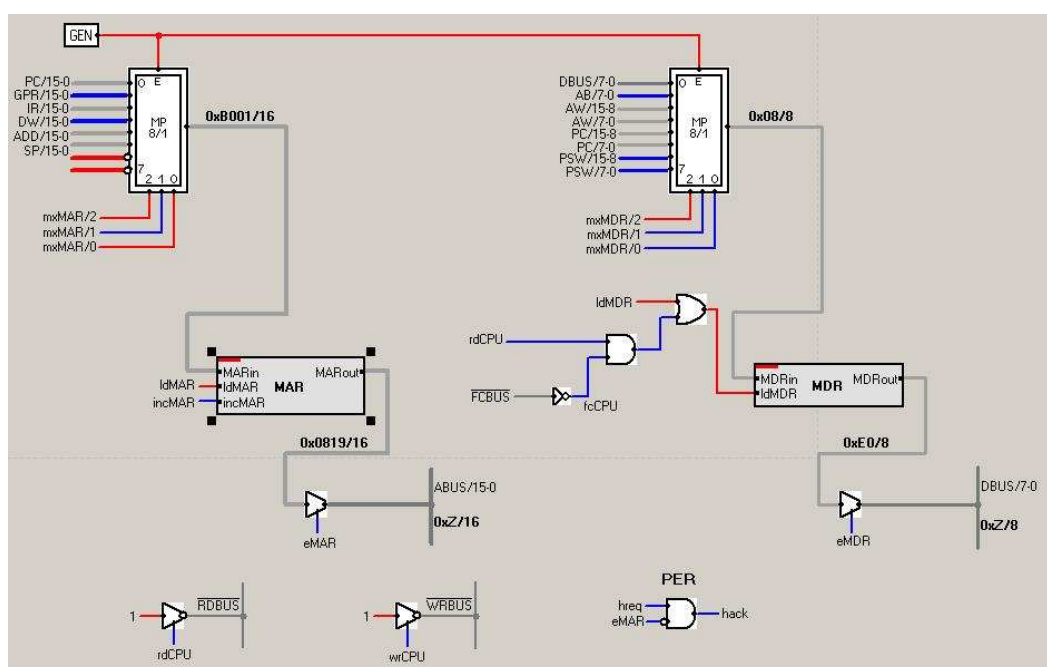
Upis se realizuje ciklusom na magistrali. Aktivnim vrednostima signala eMAR, eMDR i wrCPU započinje se ciklus. Čeka se na odgovor memorije i aktivnu vrednost signala fcCPU. Potom se signalima **mxMDR₁**, **mxMDR₀** i **ldMDR** bloka *bus* sadržaj registra $AW_{7...0}$ propušta kroz multiplekser MX i upisuje u registar MDR. Upis se realizuje na sličan način kao za prethodni bajt. Po završetku upisa se prelazi na fazu opsluživanja prekida. Kako nema prekida, odnosno signal **prekid** je neaktivan nastavlja se sa izvršavanjem sledeće instrukcije. Ovom instrukcijom STW, koja koristi memorijsko direktno adresiranje, upisan je sadržaj akumulatora AW na memorijsku lokaciju CC04h i CC05h. Ovim je upisana vrednost u ulaz tabele prekidnih rutina za prekid greške u načinu adresiranja.

U fazi čitanja ove instrukcije pročita se prvi od tri bajta instrukcije. Kako je signal **gradr** aktivan radi se o grešci u načinu adresiranja, što je prikazano na slici 69.



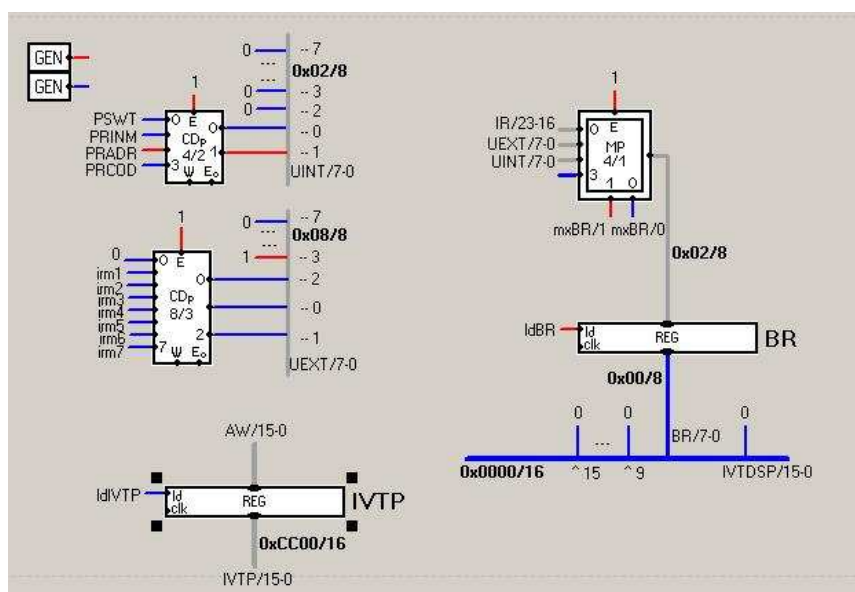
Slika 69. Vrednost signala **gradr**

Dalje se signalom **stPRCOD** bloka *intr* u flip-flop **PRCOD** upisuje aktivna vrednost i bezuslovno prelazi na fazu opsluživanja prekida. Signal **prekid** je aktivan, što znači da je došlo do prekida u toku izvršavanja ove operacije. Obrada prekida se sastoji iz tri koraka i to su čuvanje konteksta procesora, utvrđivanje broja ulaza i utvrđivanje adrese prekidne rutine. Kontekst procesora se čuva tako što se na stek stavljaju $PC_{15...0}$ i $PSW_{7...0}$ registri. Na stek se stavlja prvo viši a zatim i niži bajt registra $PC_{15...0}$. Stoga se najpre signalom **incSP** vrši inkrementiranje registra $SP_{15...0}$. Zatim se signalima **mxMAR₂**, **mxMAR₀** i **ldMAR**, sadržaj registra $SP_{15...0}$ propušta kroz multiplekser MX bloka *bus* i upisuje u registar $MAR_{15...0}$, što je prikazano na slici 70. i signalima **mxMDR₂** i **ldMDR** sadržaj višeg bajta registra $PC_{15...8}$ propušta kroz multiplekser MX i upisuje u registar $MDR_{7...0}$.



Slika 70. Upis sadržaja registra SP u registar MAR

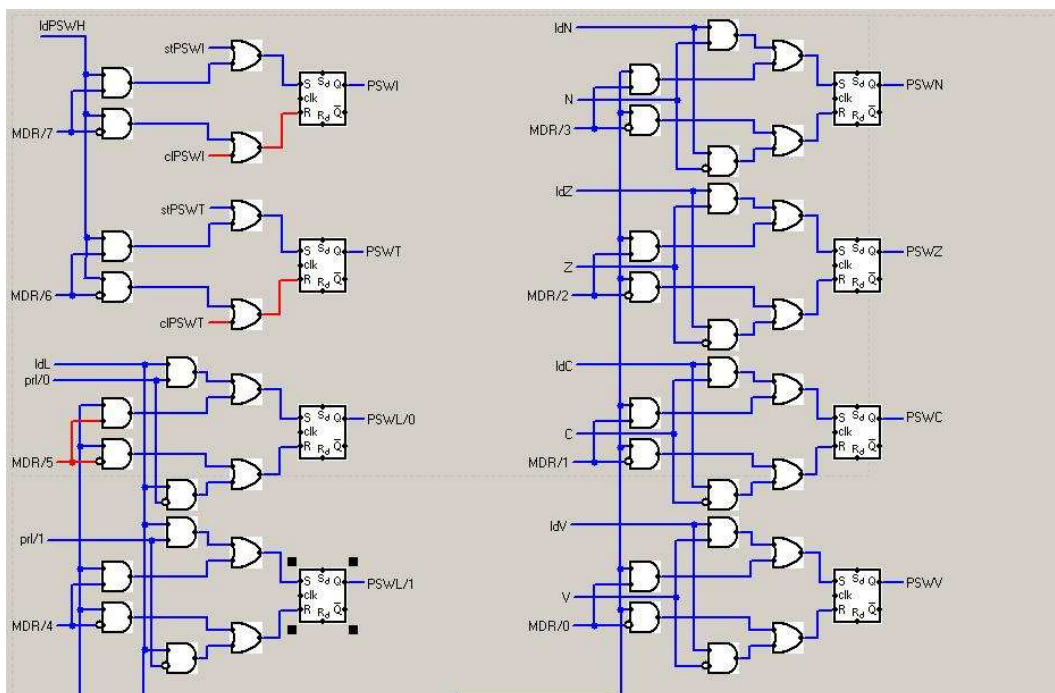
Upis se realizuje na sličan način kao kod instrukcije STW. Potom se signalom **incSP** vrši inkrementiranje registra SP_{15...0}. Zatim se signalima **mxMAR₂**, **mxMAR₀** i **ldMAR**, sadržaj registra SP_{15...0} propušta kroz multiplexer MX bloka *bus* i upisuje u registar MAR_{15...0} i signalima **mxMDR₂**, **mxMDR₀**, **ldMDR** sadržaj nižeg bajta registra PC_{7...0} propušta kroz multiplexer MX i upisuje u registar MDR_{7...0}. Upis se vrši na isti način kao i prethodni bajt. Sledeća se na stek stavlja programska statusna reč PSW. Stoga se najpre signalom **incSP** vrši inkrementiranje registra SP_{15...0}. Zatim se signalima **mxMAR₂**, **mxMAR₀** i **ldMAR**, sadržaj registra SP_{15...0} propušta kroz multiplexer MX bloka *bus* i upisuje u registar MAR_{15...0} i signalima **mxMDR₂** i **mxMDR₁**, **ldMDR** sadržaj višeg bajta registra PSW_{15...8} propušta kroz multiplexer MX i upisuje u registar MDR_{7...0}. Upis se realizuje kao kod PC registra. Na sličan način se na stek stavlja i niži bajt registra PSW_{7...0}. Prelazi se na utvrđivanje broja ulaza i to kako je signal **pradr** aktivan aktivnom vrednošću signala **mxBR₁** bloka *intr* sadržaj UINT_{7...0} sa izlaza kodera CD2, koji sadrži broj ulaza, propušta kroz multiplexer MX i signalom **ldBR** upisuje u registar BR_{7...0}, kao što je prikazano na slici 71.



Slika 71. Upis broja ulaza IVT

Istovremeno se signalom **clPRADR** bloka *intr* flip-flop PRADR postavlja na neaktivnu vrednost. U nastavku se vrši određivanje adrese prekidne rutine i upisivanje u programski brojač. Neaktivnim vrednostima signala **mxADDA₁**, **MXADDA₀**, **mxADDB₁** i **MXADDB₀**, bloka *addr* kroz multiplexere MX2 i MX3 na ulaze sabirača ADD propuštaju sadržaj registra IVTP_{15...0} i sadržaj IVTDSP_{15...0} koji predstavlja sadržaj registra BR_{7...0} pomeren ulevo za jedno mesto i proširen nulama do dužine 16 bita. Signalima **mxMAR₂** i **ldMAR** bloka *bus* se sadržaj ADD_{15...0} sa izlaza sabirača ABB propušta kroz multiplexer MX bloka *bus* i upisuje u registar MAR_{15...0}. Time se u registru MAR_{15...0} nalazi adresa memorijske lokacije počev od koje treba pročitati dva bajta koji predstavljaju viši i niže bajt adrese prekidne rutine. Čitanje prvog i drugog bajta se realizuje kao čitanje prvog bajta instrukcije. Prvi bajt se signalom **ldDWH** upisuje u viši bajt registra DW_{15...0} bloka *bus*, a signalom **incMAR** adresni registar MAR_{15...0} inkrementirana na adresu sledećeg bajta. Drugi bajt se signalom **ldDWL** upisuje u niži bajt registra DW_{15...0}. Time se u registru DW_{15...0} nalazi adresa prekidne rutine. Na kraju se neaktivnim signalima **mxPC₁** i **mxPC₀**

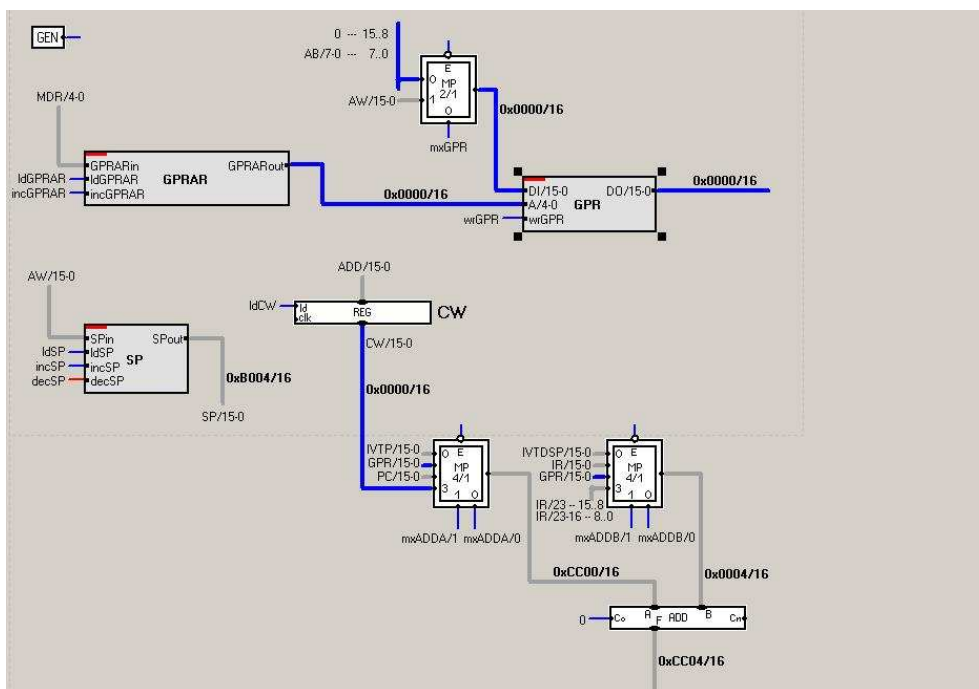
sadržaj registra $DW_{15...0}$ propušta kroz multiplexer MX i signalom **ldPC** upisuje u registar $PC_{15...0}$. Time se u registru $PC_{15...0}$ nalazi adresa prve instrukcije prekidne rutine. Signalima **clPSWI** i **clPSWT** se u razrede PSWI i PSWT bloka *exec* upisuju neaktivne vrednosti, kao što je prikazano na slici 72.



Slika 72. Branjanje bitova PSWI i PSWT

Time se u prekidnu rutinu ulazi sa režimom rada u kome su maskirani svi maskirajući prekidi i u kome nema prekida posle svake instrukcije. Ovo je kraj faze opsluživanja prekida i prelazi se na fazu čitanja sledeće instrukcije.

Posle pročitano prvog bajta instrukcije signal **II** je aktivan i prelazi se u fazu izvršavanja instrukcije. Kako je signal **RTI** aktivan, radi se o izvršavanju te instrukcije. U fazi izvršavanja ove instrukcije vrednostima sa steka se restauriraju programska statusna reč $PSW_{7...0}$ i programski brojač $PC_{15...0}$. Najpre se vrednošću sa steka restaurira programska statusna reč $PSW_{15...0}$. Stoga se signalima **mxMAR₂**, **mxMAR₀** i **ldMAR** bloka *bus* sadržaj registra $SP_{15...0}$ propušta kroz multiplexer i upisuje u registar $MAR_{15...0}$. Pored toga se i signalom **decSP** dekrementira sadržaj registra $SP_{15...0}$ bloka *exec*, kao što je prikazano na slici 73. Čitanje se realizuje slično kao i čitanje prvog bajta instrukcije. Potom se signalom **ldPSW** sadržaj registra $MDR_{7...0}$ bloka *bus* upisuje u registar $PSW_{7...0}$ bloka *exec*. Na sličan način se postavljaju prvo viši bajt registra $PSW_{15..8}$, pa niži i viši bajt registra PC . Dalje se prelazi u fazu opsluživanja prekida. Kako prekida nema, jer je signal prekid neaktivan, prelazi se na izvršavanje instrukcije čija se adresa nalazi u registru PC . Ovo je i poslednja instrukcija programa tako da se završava i njegov opis.



Slika 73. Dekrementiranje registra SP

6.3. Primeri programiranog ulaza i izlaza

Primeri programa koji opisuju sve tehnike programiranog ulaza i uzlaza kompletno su dati u prilogu. Pripremljeni su opisi programiranog ulaza sa periferije korišćenjem kontrolera bez direktnog pristupa memoriji i tehnike ispitivanja bita spremnosti, izlaz na periferiju korišćenjem takođe kontrolera bez direktnog pristupa memoriji i tehnike generisanja prekida i konačno ulaz sa periferije korišćenjem kontrolera sa direktnim pristupom memoriji. Uz programe dati su kraći opisi izvršavanja programa i pomoćna pitanja koja treba da sugerišu najvažnije principe upotrebljene pri izvršavanju programa.

6.3.1. Ulaz sa periferije tehnikom ispitivanja bita spremnosti

U ovom poglavlju dat je detaljan opis izvršavanja programa koji ilustruje ulaz sa periferije korišćenjem kontrolera bez direktnog pristupa memoriji i tehnike ispitivanja bita spremnosti. Opis je dat do nivoa aktivnih signala koji učestvuju u izvršavanju mikrooperacija.

Na početku dat je kratak opis izvršavanja programa. U programu se koristi uređaj broj 0 (0-64). Za startovanje periferije u odgovarajućem režimu i njeno zaustavljanje koriste se kontrolne reči: Start reč 05h, Stop reč 01h. Za ispitivanje bita spremnosti statusnog registra periferije koristi se maska 01h. Adrese registara kontrolera korišćene periferije su: Adrese Kontrolnog registra F000, Statusnog registra F001, Registra podataka F002. Adresa bloka podataka 0543 se puni u R1, dužina bloka podataka se puni u R2, a vrednost 0 se puni u R3. Start reč se upisuje u kontrolni

registar periferije i ona se startuje. Proverava se bit spremnost dok podatak ne bude spreman. Podatak se učitava i smešta na željenu adresu pomoću registra R1 i R3 i sadržaji registara R3 i R2 se ažuriraju. Proverava se da li je ostalo još podataka za prenos. Ako jeste ponovo se čeka na statusni registar, a ako nije program se završava.

6.3.1.1. Inicijalizacija programa

Pre početka izvršavanja programa neophodno je da odgovarajući registri budu inicijalizovani početnim vrednostima. U ovom slučaju inicijalizovani su registri IVTP, PC i SP i to vrednostima CC00h, C000h i B000h. Takođe mora se inicijalizovati deo memorije u kome treba da se nalaze instrukcije programa odgovarajućim kodiranim vrednostima. Ove vrednosti zajedno sa simboličkim instrukcijama programa date su na slici 74.

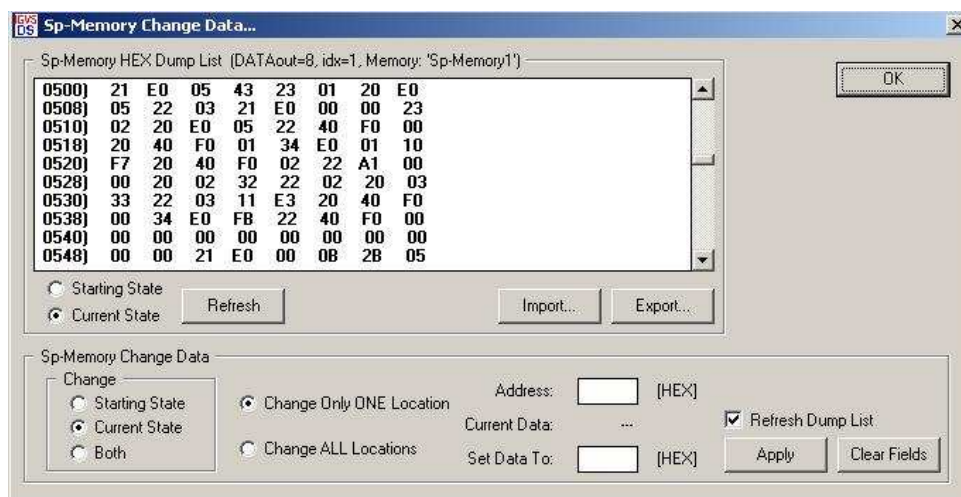
500	LDW imm(0543)	Acc16 = 0543h	21 E0 05 43
504	STW reg(R1)	R1 = Acc16	23 01
506	LDB imm(05)	Acc8 = 05h	20 E0 05
509	STB reg(R3)	R3 = Acc8	22 03
50B	LDW imm(0000)	Acc16 = 0000h	21 E0 00 00
50F	STW reg(R2)	R2 = Acc16	23 02
511	LDB imm(05)	Acc8 = 05h	20 E0 05
514	STB mem(F000)	mem[F000] = Acc8	22 40 F0 00
518	LDB mem(F001)	Acc8 = mem[F001]	20 40 F0 01
51C	AND imm(01)	Acc8 = 01h AND Acc8	34 E0 01
51F	BEQL imm(F7)	skok na adresu 0518	10 F7
521	LDB mem(F002)	Acc8 = mem[F002]	20 40 F0 02
525	STB bi(R1, R2)	mem[R1+R2] = Acc8	22 A1 00 00
529	LDB reg(R2)	Acc8 = R2	20 02
52B	INC	Acc8 = Acc8 +1	32
52C	STB reg(R2)	R2 = Acc8	22 02
52E	LDB reg(R3)	Acc8 = R3	20 03
530	DEC	Acc8 = Acc8 -1	33
531	STB reg(R3)	R3 = Acc8	22 03

533	BNEQL imm(E3)	skok na adresu 0518	11 E3
535	LDB mem(F000)	Acc8 = mem[F000]	20 40 F0 00
539	AND imm(FB)	Acc8 = FBh AND Acc8	34 E0 FB
53C	STB mem(F000)	mem[F000] = Acc8	22 40 F0 00
542	db 00		00
543	db 00		00
544	db 00		00
545	db 00		00
546	db 00		00
547	db 00		00

Slika 74. Instrukcije programa

6.3.1.2. Izvršavanje programa

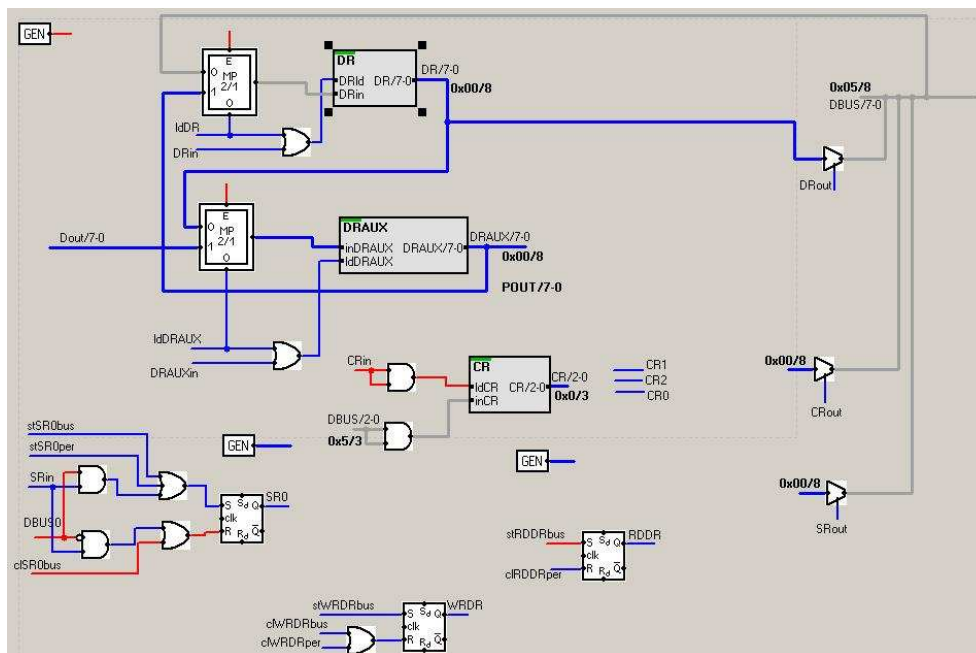
Na početku programa u okviru izvršavanja instrukcije LDW, koja koristi neposredno adresiranje, vrednost 0543h se prebacuje u 16 bitni akumulator AW. Ova vrednost predstavlja adresu početka bloka za smeštanje podataka sa periferije u memoriji. Sadržaj 16 bitnog akumulatora AW se dalje u okviru izvršavanja instrukcije STW, koja koristi direktno registarsko adresiranje, prebacuje u registra R1, odakle će se koristiti za adresiranje bloka za smeštanje podataka.



Slika 75. Stanje bloka memorije pre početka izvršavanja

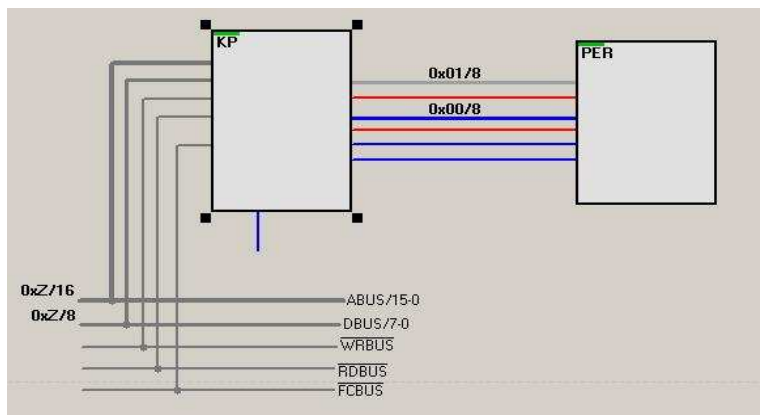
U okviru izvršavanja instrukcije STB, koja koristi neposredno adresiranje, vrednost 05h se smešta u 8 bitni akumulator AB. Ova vrednost predstavlja dužinu bloka podataka za smeštanje sa periferije. Dalje se u okviru izvršavanja instrukcije STB, koja koristi direktno registarsko adresiranje, sadržaj 8 bitnog akumulatora AB prebacuje u registra R3. Ova vrednost će se koristiti za detektovanje kraja prenosa

podataka. U okviru izvršavanja instrukcije LDW, koja koristi neposredno adresiranje, vrednost 0000h se prebacuje u 16 bitni akumulator AW. Ova vrednost predstavlja indeks podatka koji se prebacuje sa periferije u memoriju. Dalje se u okviru izvršavanja instrukcije STW, koja koristi registrasko direktno adresiranje, sadržaj 16 bitnog akumulatora AW prebacuje u registar R3. Ovaj će se registar koristiti pri adresiranju bloka za smeštanje podataka sa periferije. Instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 05h se prebacuje u 8 bitni akumulator AB. Ova vrednost predstavlja start reč kontrolera periferije bez direktnog pristupa memoriji. Ovom reči kontroler se startuje za prenos podataka iz periferije u memoriju mehanizmom provere statusnog bita, kao što je prikazano na slici 76.



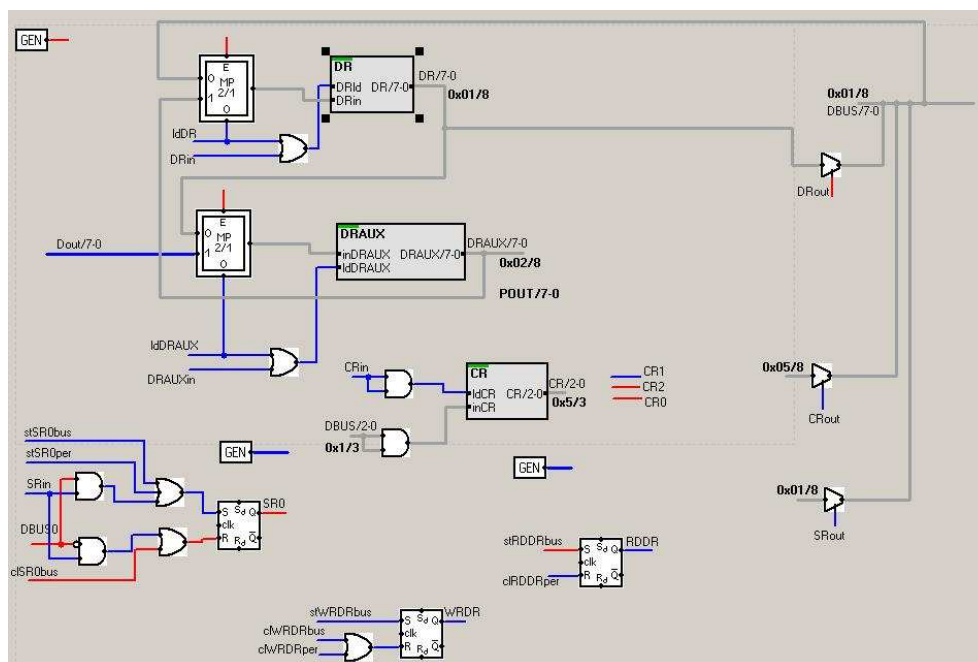
Slika 76. Upis u kontrolni registra kontrolera

Da bi kontroler bio startovan ova vrednost se prebacuje u njegov kontrolni registar. Ovo se dešava u okviru izvršavanja instrukcije STB, koja koristi direktno memorijsko adresiranje. Dakle, sadržaj 8 bitnog akumulatora se prebacuje u memorijsku lokaciju sa adresom F000h. Ova adresa je adresa iz ulazno/izlaznog memorijskog prostora i predstavlja adresu kontrolnog registra kontrolera periferije bez direktnog pristupa memoriji koji se koristi u prenosu.



Slika 77. Prenos podatka iz periferije u kontroler

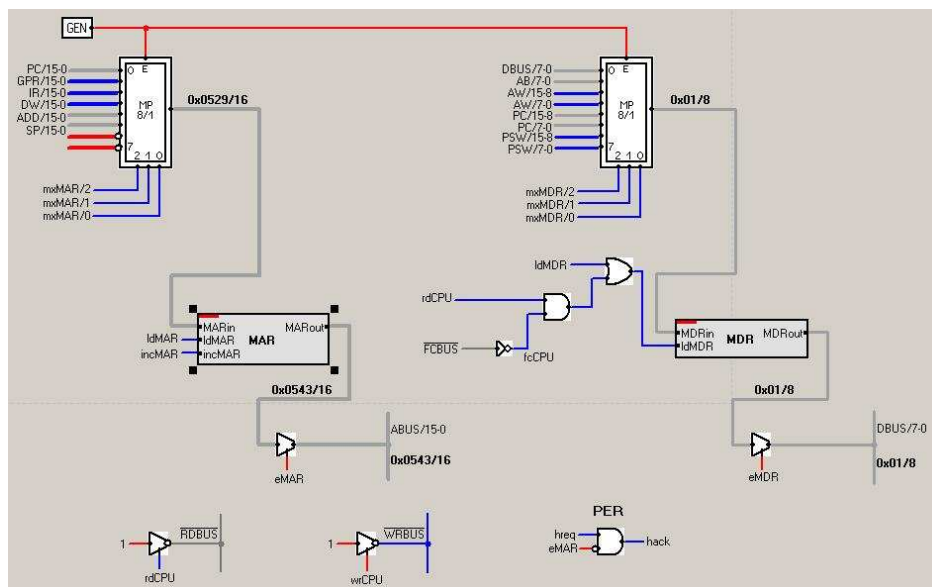
Dalje se, u okviru izvršavanja instrukcije LDB, koja koristi memorijsko direktno adresiranje, sadržaj memorijske lokacije sa adresom F001h prebacuje u 8 bitni akumulator AB. Ova adresa je, takođe, adresa iz ulazno/izlaznog memorijskog prostora i predstavlja adresu statusnog registra kontrolera periferije koji se koristi u prenosu. U okviru izvršavanja ove instrukcije sadržaj statusnog registra kontrolera se prebacuje u 8 bitni akumulatro da bi se proverilo stanje statusnog ready bita koji govori da li je završen prenos jednog podatka iz periferije u registra podataka kontrolera periferije. To se proverava u okviru izvršavanja sledeće instrukcije AND, koja koristi neposredno adresiranje. Dakle, nad sadržajem 8 bitnog akumulatora i vrednosti 01h izvršava se operacija logičkog I i pri tome se dobija rezultat 00h ako statusni bit nije postavljen i vrednost 01h ako je postavljen u 8 bitnom akumulatoru kao rezultat. Na osnovu sadržaja akumulatora postavljaju se i indikatori statusnog registra procesora PSW NF, CF, OF, ZF. Dalje u okviru izvršavanja instrukcije BEQL, proverava se uslov skoka koji se generiše na osnovu postavljenih indikatora PSW. Kako je vreme odziva memorije periferije postavljeno na jedan takt, podatak će u ovom trenutku biti spreman u registru podataka kontrolera i uslov skoka neće biti ispunjen. Izvršavanje programa će se nastaviti sekvencijelono, jer je podatak u kontroleru spreman i nije potrebna ponovna provera statusnog bita ready kontrolera. U okviru izvršavanja instrukcije LDB, koja koristi memorijsko direktno adresiranje, sadržaj memorijske lokacije sa adresom F002h se prebacuje u 8 bitni akumulatro AB. Ova vrednost predstavlja sadržaj registra podataka kontrolera, jer je F002h njegova adresa. Ovu vrednost je sada neophodno prebaciti u memoriju u blok za smeštanje podataka, kao što je prikazano na slici 78.



Slika 78. Podatak u registru podataka kotrolera

U okviru izvršavanja instrukcije STB, koja koristi bazno-indeksno adresiranje sa pomerajem, sadržaj 8 bitnog akumulatora se prebacuje na odgovarajuću adresu u memoriji. Pri adresiranju koriste se registri R1 i R2 koji sadrže adresu početka bloka u memoriji i indeksa prebačenog podatka i njihovim sabiranjem se lako adresira željena lokacija u bloku podataka, kao što je prikazano na slici 79. Sada je neophodno

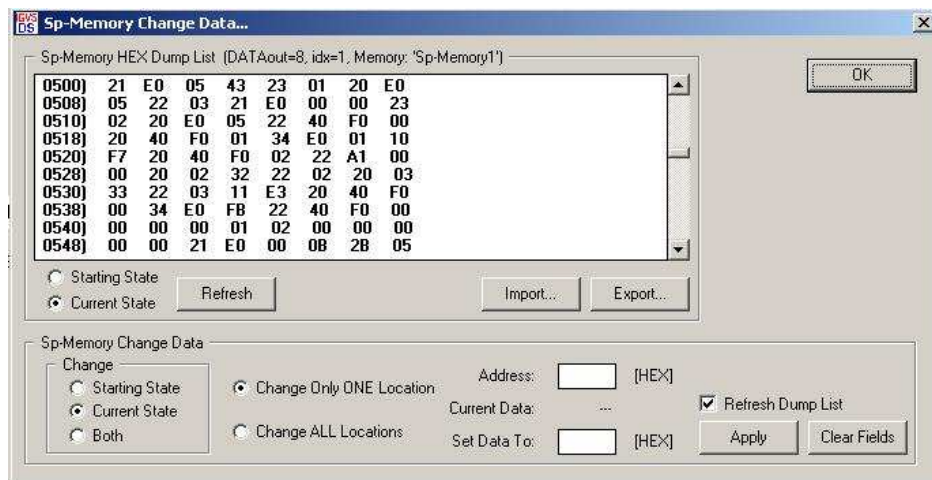
ažurirati registre R2, i R3 da bi prikazivali trenutno stanje prenosa, jer je upravo prebačen jedan podatak.



Slika 79. Upis podatka u memoriju

U okviru instrukcije LDB, koja koristi registarsko direktno adresiranje, sadržaj registra R2 se prebacuje u 8 bitni akumulator AB. Dalje se u okviru izvršavanja bezadresne instrukcije INC, sadržaj 8 bitnog akumulatora inkrementira i rezultat smešta u 8 bitni akumulator AB. U okviru izvršavanja instrukcije STB, koja koristi registarsko direktno adresiranje, sadržaj 8 bitnog akumulatora AB se smešta u registra R2. U okviru izvršavanja prethodne tri instrukcije inkrementiran je sadržaj registra R2. Dalje se, u okviru izvršavanja instrukcije LDB, koja koristi registarsko direktno adresiranje, sadržaj registra R3 prebacuje u 8 bitni akumulator AB. U okviru izvršavanja bezadresne instrukcije DEC se sadržaj 8 bitnog akumulatora AB dekrementira i rezultat ponovo smešta u 8 bitni akumulator AB. Takođe, se postavljaju indikatori statusnog registra procesora PSW. U okviru izvršavanja instrukcije STB, koja koristi direktno registarsko adresiranje, sadržaj 8 bitnog akumulatora AB se smešta u registra R3. U okviru izvršavanja prethodne tri instrukcije dekrementiran je sadržaj registra R3. Kada je sadržaj odgovarajućih registara ažuriran, treba proveriti da li su preneti svi podaci. To se postiže instrukcijom uslovnog skoka, jer su, prethodno, pri dekrementiranju registra dužine bloka podataka postavljeni indikatori statusnog registra procesora PSW. U okviru izvršavanja instrukcije BNEQL proverava se uslov skoka koji je ispunjen, jer je prenet samo jedan podatak od pet koliko je dužina bloka, tako da se izvršavanje programa nastavlja na adresi 0518h. Na ovoj adresi ponovo se započinje prenos sledećeg podatka koji se odvija na sličan način kao prenos prethodnog. Na ovaj način se prenesu svi podaci sa periferije u memoriju i ostaje još samo da se isključi kontroler periferije jer je prenos završen. U okviru izvršavanja instrukcije LDB, koja koristi memorijsko direktno adresiranje, sadržaj adrese F000h se prenosi u 8 bitni akumulator AB. Ovo predstavlja sadržaj kontrolnog registra kontrolera. U okviru izvršavanja instrukcije AND, koja koristi neposredno adresiranje, nad sadržajem 8 bitnog akumulatora AB i vrednosti FBh vrši se operacija logičkog I i rezultat smešta u 8 bitni akumulator. Time se stop reč kontrolera nalazi u 8 bitnom akumulatoru. Ostaje još samo da se ona prenese u kontrolni registar kontrolera. To se postiže u okviru

izvršavanja instrukcije STB, koja koristi memorijsko direktno adresiranje. Blok podataka je prenešen sa periferije u memoriju i kontroler je zaustavljen, pa ovo predstavlja kraj izvršavanja programa i kraj opisa izvršavanja.



Slika 80. Dva preneti podatka u memoriji

7. ZAKLJUČAK

U šest prethodnih poglavlja ovog rada dat je kratak pregled arhitekture projektovanog računarskog sistema, pregled najkorisnijih i upotrebljenih mogućnosti i funkcija softverskog alata IgoVSoDS korišćenog za projektovanje i izvršavanje priloženih programa i posmatranje interakcije delova računarskog sistema, dat je kratak pregled strukture realizovanih delova simulatora kao i nekoliko detaljnih primara programa koji se na njemu izvršavaju i u prilogu je dat kompletan skup realizovanih programa sa pitanjima i objašnjenjima.

Kroz prethodna poglavlja ostvaren je cilj ovog rada koji je bio je da se konstruiše simulator ulazno/izlaznog podsistema jednog računarskog sistema za laboratoriju unapred date arhitekture. Takođe, da se za taj sistem naprave programi koji najbolje ilustruju sve glavne principe na kojima je računarski sistem projektovan kao i sve mogućnosti koje on pruža. Kroz primere asemblerskih programa isprobane su sve najvažnije instrukcije iz skupa instrukcija datog procesora, svi mogući načini adresiranja podataka, komunikacija procesora sa ulazno/izlaznim uređajuma sa direktnim pristupom memoriji kao i onih bez direktnog pristupa memoriji kao i mehanizam prekida. Uz primere data su objašnjenja suštine primera kao i kratak opis njihovog odvijanja i pitanja koja treba da pomognu u razumevanju funkcionisanja datog računarskog sistema.

Takođe cilj ovog rada je bio i provera ispravnosti arhitekture konstruisanog računarskog sistema i provera funkcionisanja softverskog alata u kom je simulator sistema projektovan.

Zaključak autora ovog rada je da je arhitektura računarskog sistema dobra i da pruža najjednostavniji za razumevanje pogled u osnovne načine adresiranja, skup instrukcija, mehanizam prekida, cikluse na magistrali i komunikaciju sa ulazno/izlaznim uređajima. Sistem funkcioniše prema predviđenim pravilima kao celina i predstavlja dobar primer za laboratorijske vežbe kroz napravljeni simulator i programe.

Takođe, autor je mišljenja da korišćeni softverski paket IgoVSoDS predstavlja ne mnogo zahtevan, jednostavan i lak za upotrebu interfejs za izradu simulatora računarskih sistema i komponenti i njihovo korišćenje za simulacije. Alat ima nekoliko važnijih funkcionalnosti koje se brzo savladavaju uz detaljan priručnik i koje pružaju dovoljno mogućnosti za projektovanje najrazličitijih sistema. Autor je, međutim, ipak primetio neke nedostatke alata za koje je siguran da će biti otklonjenje u budućnosti njegovog razvijanja. Takvi nedostaci su recimo mala raznovrsnost ugrađenih osnovnih kola, nemogućnost pomeranja logičkih kola već povezanih signalima, nemogućnost povezivanja otvorenih signala na portove submodula, komplikovanost izrade magistrala i crtanja samostalnih signala i mala nestabilnost programa u pojedinim situacijama. Takođe, autor smatra da bi uz program i priručnik za korišćenje trebalo da ide neki skup osnovnih primera radi lakšeg i bržeg razumevanja mogućnosti sistema.

8. LITERATURA

1. J. Đorđević, *Arhitektura i organizacija računara, Računarski sistem za rad u laboratoriji, Arhitektura i organizacija računarskog sistema, Interni materijal u pripremi za štampu, Elektrotehnički fakultet Beograd, 2007*
2. N. Grbanović, *Interaktivni generator vizuelnih simulatora digitalnih sistema – Doktorska disertacija u izradi, Interni materijal u pripremi za štampu, 2007*
3. N. Grbanović, *Interaktivni Generator Vizuelnih Simulatora Digitalnih Struktura (IGoVSoDS) – priručnik za upotrebu, Elektrotehnički fakultet Beograd, 2007*
4. <http://rti.etf.bg.ac.yu/rti/oe3aor/literatura/literatura.html>
5. <http://rti.etf.bg.ac.yu/rti/ri3aor/literatura/predavanja/>
6. J. Đorđević, *Arhitektura računara, edukacioni računarski sistem, Arhitektura i organizacija računarskog sistema, Akademska misao, Beograd, 2003*
7. J. Đorđević, N. Grbanović, B. Nikolić, Z. Radivojević, *Arhitektura računara, edukacioni računarski sistem, priručnik za simulaciju sa zadacima, Akademska misao Beograd, 2008*
8. J. Đorđević, B. Nikolić, *Arhitektura i organizacija računara – skripta sa predavanja, Elektrotehnički fakultet Beograd, 2007*
9. N. Todorović, *Implementacija simulatora računara za rad u laboratoriji korišćenjem softverskog paketa IgoVSoDS – Diplomski rad, 2007*
10. J. Djordjevic, A. Milenkovic, N. Grbanovic, *An Integrated Educational Environment for Teaching Computer Architecture and Organization, IEEE MICRO, Vol. 20, No. 3, pp. 66-74, May/June 2000.*
11. J. Djordjevic, B. Nikolic, A. Milenkovic, *Flexible Web-based Educational System for Teaching Computer Architecture and Organization, IEEE Transactions on Education, Vol. 48, No. 2, pp. 264-273, May 2005*
12. J. Djordjevic, M. Mitrovic, B. Nikolic, *A Memory System for Education, The Computer Journal, Vol. 48, No. 6, pp. 630-641, November 2005.*
13. A. Stojkovic, J. Djordjevic, B. Nikolic, *WASP – A Web-Based Simulator for an Educational Pipelined Processor, International Journal on Electrical Engineering Education, Vol. 44, No. 3, 2007, pp. 197-215.*
14. J. Djordjevic, B. Nikolic, T. Borožan, A. Milenković, *CAL²: Computer Aided Learning in Computer Architecture Laboratory, Computer Applications in Engineering Education, Vol. 16, No. 3, 2008, pp. 172-188.*

8. PRILOG

Napomena: U sledećim primerima assembleriskih programa se uz simboličke oznake instrukcija, daju kratka objašnjenja i heksadekadni zapis kodiranih instrukcija programa.

U svakom programu neophodno je pre izvršavanja upisati na adrese C001h i C002h adresu početka programa. Izvršavanje programa počinje od adrese C000h gde se nalazi безусловni skok na adresu koja se nalazi na adresama C001h i C002h, zbog toga je neophodno u ove adrese upisati adresu početka programa.

1. Instrukcije i načini adresiranja

Sledeći programi ilustruju korišćenje raspoloživih instrukcija procesora, kao i načina adresiranja. Za svaku instrukciju postoji poseban program koji je ilustruje, a u programima se koriste i svi raspoloživi načini adresiranja.

1) Aritmetičke instrukcije

1. Instrukcija ADD

Cilj ovog primera je da se ilustruje aritmetička operacija sabiranja kao i neposredno, memorijsko direktno i PC relativno adresiranje.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 05h puni u akumulator AB. Sadržaj akumulatora AB se zatim u okviru izvršavanja instrukcije ADD, koja koristi memorijsko direktno adresiranje, sabira sa sadržajem memorijske lokacije 010Bh, koji ima vrednost 03h i rezultat se smešta u akumulator. Na kraju se instrukcijom STB, koja koristi PC relativnim adresiranje, rezultat prebacuje iz akumulatora AB u memorijsku lokaciju 010Bh.

U memorijskim lokacijama 0100h do 010Ah nalaze se instrukcije, a u memorijskoj lokaciji 010Bh nalazi se 8 bitni operand 03h i to je adresa na koju se na kraju smešta 8 bitni rezultat.

0100 LDB imm(05)	;AB = 05	;20 E0 05
0103 ADD memdir(010B)	;AB = AB + mem[010B]	;30 40 01 0B
0107 STB pcpc(0000)	;mem[PC+0000] = AB	;22 C0 00 00
010B db 03	;8 bitni operand	;03

Pitanja:

1. U trenutku T=32 šta se nalazi u memorijskoj lokaciji 010Bh?
2. Koja je vrednost signala ldMAR u trenutku T=34 i koja je njegova uloga? U istom trenutku koja je vrednost signala incPC i šta se time postiže?
3. U trenutku T=36 koja je vrednost signala eMAR i rdCPU i šta oni rade?
4. Da li se u trenutku T=40 vrednost signala fcCPU promenila i šta to pokazuje?
5. U trenutku T=41 koja je vrednost signala ldIR0 i šta se njome postiže?
6. Da li je signal gropr aktivan u trenutku T=42 i zašto?
7. Koja je vrednost signala I1 u trenutku T=42 i šta se njome dobija?
8. Da li signal gradr ima aktivnu vrednost u trenutku T=42 i zašto?
9. U trenutku T=42 koji od signala l2_branch i l2_arlog je aktivan i šta se njime utvrđuje?
10. U trenutku T=51 koja je vrednost signala ldIR1 i šta se njime postiže?

11. U trenutku T=52 da li je signal l3_jump ili signal l3_arlog aktivan i zašto?
12. Koja je vrednost signala mxBB1 i ldBB u trenutku T=66 i šta se njima postiže? Koja se vrednost upisuje u registar BB i zašto?
13. Da li su signali mxAB i ldAB aktivni u trenutku T=68 i šta se njima postiže? Koja je vrednost u registru AB i šta ona predstavlja?
14. U trenutku T=90 da li je vrednost signala memdir aktivna i šta to znači?
15. Zašto su u trenutku T=93 signali mxMAR1 i ldMAR aktivni?
16. Koje su vrednosti signala eMAR i rdCPU u trenutku T=96?
17. Da li su vrednosti signala mxBB0 i ldBB aktivne u trenutku T=120? U istom trenutku zašto je signal ADD aktivan?
18. Šta se nalazi u registrima AB i BB u trenutku T=121? Šta se u istom trenutku pojavljuje na izlazima ALU?
19. Koje su vrednosti signala mxAB i ldAB u trenutku T=122 i šta se njima postiže?
20. Zašto su u trenutku T=123 signali ldN i ldZ aktivni?
21. Koje su vrednosti signala mxADDA1 i mxADDB0 u trenutku T=165? U kakvoj je to vezi sa vrednošću signala mxMAR2 i ldMAR?
22. U trenutku T=170 šta se nalazi u memorijskoj lokaciji 010Bh i zašto se ona promenila u odnosu na trenutak T=32?

2. Instrukcija SUB

Cilj ovog primera je da se ilustruje aritmetička operacija oduzimanja kao i neposredno, memorijsko indirektno i registarsko direktno adresiranje.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 06h puni u akumulator AB. Od sadržaja akumulatora AB se, u okviru izvršavanja instrukcije SUB, koja koristi memorijsko indirektno adresiranje, oduzima vrednost 02h sa adrese 011Ch čija se adresa nalazi na adresi 011Ah i rezultat se smešta u akumulator AB. Na kraju se instrukcijom STB, koja koristi registarsko direktno adresiranje, rezultat prebacuje iz akumulatora u registar R5.

U memorijskim lokacijama 0110h do 0118h nalaze se instrukcije, memorijska lokacija 0119h nije iskorišćena, u memorijskim lokacijama 011Ah i 011Bh nalaze se viši i niži bajt 16 bitne adrese operanda 011Ch i u memorijskoj lokaciji 011Ch nalazi se 8 bitni operand 02h.

0110 LDB imm(06)	;AB = 06h	;20 E0 06
0113 SUB memind(011A)	;AB = AB – mem[[011A]]	;31 60 01 1A
0117 STB regdir(R5)	;R5 = AB	;22 05
0119		
011A dw 01 1C	;16 bitna adresa operanda	;01 1C
011C db 02	;8 bitni operand	;02

Pitanja:

1. U trenutku T=32 koja je vrednost u memorijskoj lokaciji 011C?
2. Koja je vrednost signala ldMAR u trenutku T=34 i koja je njegova uloga? U istom trenutku koja je vrednost signala incPC i šta se time postiže?
3. U trenutku T=41 koja je vrednost signala ldIR0 i šta se njome postiže?
4. Da li je signal gropr aktivan u trenutku T=42 i zašto?
5. Koja je vrednost signala lI u trenutku T=42 i šta se njome dobija?
6. Da li signal gradr ima aktivnu vrednost u trenutku T=42 i zašto?

7. U trenutku T=42 koji od signala l2_branch i l2_arlog je aktivan i šta se njime utvrđuje?
8. U trenutku T=52 da li je signal imm aktivan i šta to znači?
9. Koje su vrednosti signala mxBB1 i ldBB u trenutku T=66 i šta one omogućuju? Koja se vrednost upisuje u registar BB?
10. Koje su vrednosti signala mxMAR1 i ldMAR u trenutku T=111 i šta se njima postiže?
11. U trenutku T=118 koja je vrednost signala incMAR i za šta se ona koristi?
12. U trenutku T=126 koje su vrednosti signala mxMAR1, mxMAR0 i ld MAR i zašto? Šta se nalazi u MAR?
13. U trenutku T=135 koje su vrednosti signala mxBB0 i ldBB i šta one omogućavaju? Šta se upisuje u registar BB?
14. U trenutku T=137 koja je vrednost signala ldAB? Šta se upisuje u registar AB?
15. Koja je vrednost signala ldGPADR u trenutku T=158? Za šta se ona koristi?
16. U trenutku T=166 koja je vrednost signala wrGPR? Šta se upisuje u registar R5?

3. Instrukcija INC

Cilj ovog primera je da se ilustruje aritmetička instrukcija inkrementiranja kao i neposredno, registarsko direktno, registarsko indirektno i memorijsko direktno adresiranje.

U programu se najpre instrukcijom LDW, koja koristi neposredno adresiranje, 16 bitni akumulator AW puni vrednošću 0131h. Ova vrednost predstavlja adresu kojom se puni registar R5, u okviru izvršavanja instrukcije STW, koja koristi registarsko direktno adresiranje. Registar R5 se koristi za adresiranje lokacije 0131h, čiji se sadržaj puni u 8 bitni akumulator AB, u okviru izvršavanja instrukcije LDB, koja koristi registarsko indirektno adresiranje i rezultat se smešta u 8 bitni akumulator AB. 8 bitni akumulator AB se inkrementira u okviru izvršavanja instrukcije INC i rezultat se smešta u 8 bitni akumulator AB. Na kraju se instrukcijom STB, koja koristi memorijsko direktno adresiranje, rezultat prebacije iz 8 bitnog akumulatora AB u memorijsku lokaciju 0131h.

U memorijskim lokacijama 0120h do 012Eh nalaze se instrukcije, memorijske lokacije 012Dh do 0130h nisu iskorišćene i u memorijskoj lokaciji 0131h nalazi se 8 bitni operand 01h i u nju se na kraju smešta rezultat.

0120 LDW imm(0131)	;AW = 0131	;21 E0 01 31
0124 STW regdir(R5)	;R5 = AW	;23 05
0126 LDB regind(R5)	;AB = [R5]	;20 25
0128 INC	;AB = INC AB	;32
0129 STB memdir(0131)	;mem[0131] = AB	;22 40 01 31
012D		
012E		
012F		
0130		
0131 db 01	;8 bitni operand	;01

Pitanja:

1. U trenutku T=32 šta se nalazi u memorijskoj lokaciji na adresi 0131h?
2. U trenutku T=74 koje su vrednosti signala mxBW1 i ldBW? Koji je sadržaj registra BW ?

3. U trenutku T=76 koja je vrednost signala ldAW? Koji je sadržaj registra AW?
4. U trenutku T=96 koja je vrednost signala ldGPRAR? Zašta se ona koristi?
5. U trenutku T=104 koje se vrednosti signala mxGPR i wrGPR? Koja se vrednost upisuje u registar R5?
6. U trenutku T=107 koje su vrednosti signala mxMAR0 i ldMAR? Koja se vrednost upisuje u registar MAR?
7. U trenutku T=109 koje su vrednosti signala eMAR i rdCPU? Koji je ciklus na magistarli u toku i zašto?
8. U trenutku T=138 koje su vrednosti signala mxBB0 i ldBB? Koja se vrednost upisuje u registar BB?
9. U trenutku T=140 koja je vrednost signala mxAB i ldAB? Koja se vrednost upisuje u registar AB?
10. U trenutku T=141 koje su vrednosti signala ldN, ldZ, ldC i ldV? Zašto?
11. U trenutku T=155 koje su vrednosti signala inc, mxAB i ldAB? Šta se upisuje u registar AB?
12. U trenutku T=155 koje su vrednosti signala ldC i ldV? Zašto?
13. U trenutku T=156 koje su vrednosti signala ld Ni ldZ? Zašto?
14. U trenutku T=198 koje su vrednosti signala mxMAR1 i ldMAR? Koja se vrednost upisuje u registar MAR?
15. U trenutku T=201 koje su vrednosti signala mxMDR i ldMDR? Koja vrednost se upisuje u registar MDR?
16. U trenutku T=203 koje su vrednosti signala eMAR, eMDR i wrCPU? Koji je ciklus na magistrali u toku ? Zašto?
17. U trenutku T=208 koji je sadržaj memorijske lokacije 0131h?

4. Instrukcija DEC

Cilj ovog primera je da se ilustruje aritmetička instrukcija dekrementiranja kao i memorijsko direktno, registarsko direktno i registarsko indirektno sa adresiranjem pomerajem.

U programu se najpre instrukcijom LDW, koja koristi memorijsko direktno adresiranje, 16 bitni akumulator AW puni vrednošću sa adresa 0151h i 0152h. Ova vrednost predstavlja adresu kojom se puni registar R6, u okviru izvršavanja instrukcije STW, koja koristi registarsko direktno adresiranje. Registar R6 se koristi u okviru izvršavanja instrukcije LDB, koja koristi registarsko indirektno adresiranje sa pomerajem, za punjenje 8 bitnog akumulatora AB. Akumulator AB se dekrementira u okviru izvršavanja instrukcije DEC i rezultat se smešta nazad u akumulator AB. Na kraju se instrukcijom STB, koja koristi memorijsko direktno adresiranje, rezultat prebacuje iz akumulatora AB u memorijsku lokaciju 0153h.

U memorijskim lokacijama 0140h do 014Eh nalaze se instrukcije, memorijske lokacije 014Fh i 0150h nisu iskorišćene, u memorijskim lokacijama 0151h i 0152h nalaze se viši i niži bajt 16 bitne adrese operanda sa adrese 0153h i na adresi 0153h nalazi se 8 bitni operand 03h.

0140 LDW memdir(0151)	;AW = mem[0151]	;21 40 01 51
0144 STW regdir(R6)	;R6 = AW	;23 06
0146 LDB regindpom(R6, 0001)	;AB = mem[R6+0001]	;20 86 00 01
014A DEC	;AB = DEC AB	;33
014B STB memdir(0153)	;mem[0153] = AB	;22 40 01 53
014F		

0150

0151 dw 01 53

;16 bitna adresa operanda ;01 53

0153 db 03

;8 bitni operand ;03

Pitanja:

1. U trenutku T=32 koji je sadržaj memorijske lokacije na adresi 0153h?
2. U trenutku T=32 koji je sadržaj memorijskih lokacija na adresama 0151h i 0152h? Šta predstavlja njihov sadržaj?
3. U trenutku T=73 koje su vrednosti signala mxMAR1 i ldMAR? Koja se vrednost upisuje u registar MAR?
4. U trenutku T=76 koji ciklus je u toku na magistrali? Zašto?
5. U trenutku T=90 koje su vrednosti signala mxBW i ldBW? Koja vrednost se upisuje u registar BW?
6. U trenutku T=92 koja je vrednost signala ldAW? Koja vrednost se upisuje u registra AW?
7. U trenutku T=112 koja je vrednost signala ldGPRAR? Koja je njena uloga?
8. U trenutku T=120 koje su vrednosti signala mxGPR i wr GPR? Koja se vrednost upisuje u registar R6?
9. U trenutku T=162 koje su vrednosti signala mxADDA0, mxADDDB0, mxMAR2 i ldMAR? Koja vrednost se upisuje u registar MAR?
10. U trenutku T=165 koji je ciklus u toku na magistrali i zašto?
11. U trenutku T=171 koja je vrednost signala mxBB0 i ldBB? Koja vrednost se upisuje u registra BB?
12. U trenutku T=173 koje su vrednosti signala mxAB i ldAB? Koja se vrednost upisuje u registar AB?
13. U trenutku T=188 koje su vrednosti signala dec, ldAB? Šta se upisuje u registar AB? Koje su vrednosti signala ldC i ldV i zašto?
14. U trenutku T=189 koje su vrednosti signala ldN i ldZ?
15. U trenutku T=231 koje su vrednosti signala mxMAR1 i ldMAR? Koja vrednost se upisuje u registar MAR?
16. U trenutku T=234 koje su vrednosti signala mxMDR0 i ldMDR? Koja vrednost se upisuje u registar MDR?
17. U trenutku T=236 koji ciklus na magistrali je u toku? Koje su vrednosti signala eMAR, eMDR i wrCPU?
18. U trenutku T=241 koji je sadržaj memorijske lokacije sa adresom 0153h?

2)Logičke instrukcije

1.Instrukcija AND

Cilj ovog primera je da se ilustruje logička operacija I kao i neposredno, registarsko direktno, memorijsko direktno i bazno-indeksno adresiranje sa pomerajem.

U programu se najpre instrukcijom LDW, koja koristi neposredno adresiranje, vrednost 0170h puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registarsko direktno adresiranje, sadržaj 16 bitnog akumulatora AW se upisuje u registar R2. Instrukcijom LDW, koja koristi neposredno adresiranje, se vrednost 0006h puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registarsko direktno adresiranje, se sadržaj 16 bitnog akumulatora AW upisuje u registar R3. Instrukcijom LDB, koja koristi neposredno adresiranje, 8 bitni akumulator AB se puni

vrednošću $10100101_2 = A5h$. Registri R2 i R3 se koriste za adresiranje lokacije 0177h, u okviru izvršavanja instrukcije AND, koja koristi bazno-indeksno adresiranje sa pomerajem. Sadržaj lokacije 0177h, odnosno vrednost $01011100_2 = 5Ch$, se koristi u operaciji logičkog I zajedno sa sadržajem 8 bitnog akumulatora AB, u okviru izvršavanja instrukcije AND, i rezultat $00000100_2 = 04h$ se smešta nazad u 8 bitni akumulator AB. Na kraju se instrukcijom STB, koja koristi memorijsko direktno adresiranje, rezultat prebacuje iz 8 bitnog akumulatora AB u memorijsku lokaciju 0177h.

U memorijskim lokacijama 0160h do 0176h nalaze se instrukcije, a na adresi 0177h nalazi se 8 bitni operand 5Ch na čije se mesto na kraju smešta 8 bitni rezultat.

0160 LDW imm(0170)	;AW = 0170	;21 E0 01 70
0164 STW regdir(R2)	;R2 = AW	;23 02
0166 LDW imm(0006)	;AW = 0006	;21 E0 00 06
016A STW regdir(R3)	;R3 = AW	;23 03
016C LDB imm(A5)	;AB = A5	;20 E0 A5
016F AND bxpom(R2, R3, 0001)	;AB = AB AND mem[R2+R3+0001]	;34 A2 00 01
0173 STB memdir(0177)	;mem[0177] = AB	;22 40 01 77
0177 db 5C	;8 bitni operand	;5C

Pitanja:

1. U trenutku T=32 koji je sadržaj memorijske lokacije 0177h?
2. U trenutku T=74 koje su vrednosti signala mxBW1 i ldBW? Koja se vrednost upisuje u registar BW?
3. U trenutku T=76 koja je vrednost signala ldAW? Koja se vrednost upisuje u registar AW?
4. U trenutku T=96 koja je vrednost signala ldGPRAR? Zašta se ona koristi?
5. U trenutku T=104 koje su vrednosti signala mxGPR i wrGPR? Koja se vrednost upisuje u registar R2?
6. U trenutku T=177 koja se vrednost upisuje u registar R3?
7. U trenutku T=212 koje su vrednosti signala mxBB1 i ldBB? Koja se vrednost upisuje u registar BB?
8. U trenutku T=214 koja je vrednost signala mxAB i ldAB? Koja se vrednost upisuje u registar AB?
9. U trenutku T=256 koje su vrednosti signala mxADD0, mxADDB0, incCW i incGPRAR? Koji način adresiranja je u pitanju?
10. U trenutku T=258 koje su vrednosti signala mxADDA1, mxADDA0, mxADDB1, mxMAR2 i ldMAR? Koja se vrednost upisuje u registar MAR?
11. U trenutku T=261 koji je ciklus na magistrali u toku? Zašto? U trenutku T=267 koja vrednost se upisuje u registar BB?
12. U trenutku T=269 koje su vrednosti signala and i ldAB? Koja vrednost se upisuje u registar AB?
13. U trenutku T=270 koje su vrednosti signala ldN, ldZ, ldC i ldV?
14. U trenutku T=312 koje su vrednosti signala mxMAR1 i ldMAR? Koja vrednost se upisuje u registar MAR?
15. U trenutku T=315 koje su vrednosti signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registar MDR?
16. U trenutku T=318 koji je ciklus u toku na magistrali i zašto?

17. U trenutku $T=322$ koja se vrednost nalazi u memorijskoj lokaciji na adresi 0177h?

2. Instrukcija OR

Cilj ovog primera je da se ilustruje logička operacija ILI kao i neposredno, memorijsko direktno i PC relativno adresiranje.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, vrednost $10011010_2 = 9Ah$ puni u 8 bitni akumulator AB. Nad sadržajem 8 bitnog akumulatora AB i sadržajem memorijske lokacije 019Bh, koja ima vrednost $10100111_2 = A7h$, se u okviru izvršavanja instrukcije OR, koja koristi memorijskog direktnog adresiranja, izvršava logičko ILI i rezultat $10111111_2 = BFh$ smešta u 8 bitni akumulator AB. Na kraju se rezultat prebacuje na lokaciju 019Bh instrukcijom STB, koja koristi PC relativno adresiranje.

U memorijskim lokacijama 0190h do 019Ah nalaze se instrukcije, a na adresi 019Bh nalazi se 8 bitni operand A7h na čije se mesto na kraju smešta 8 bitni rezultat.

0190 LDB imm(9A)	;AB = 9A	;20 E0 9A
0193 OR memdir(019B)	;AB = AB OR mem[019B]	;35 40 01 9B
0197 STB pcpom(0000)	;mem[PC+0000] = AB	;22 00 00 00
019B db A7	;8 bitni operand	;A7

Pitanja:

1. U trenutku $T=32$ koja se vrednost nalazi u memorijskoj lokaciji 019Bh?
2. U trenutku $T=66$ koja je vrednost signala mxBB1 i ldBB? Koja se vrednost upisuje u registar BB?
3. U trenutku $T=68$ koje su vrednosti signala mxAB i ldAB? Koja vrednost se upisuje u registar AB?
4. U trenutku $T=111$ koje su vrednosti signala mxMAR1 i ldMAR? Koja vrednost se upisuje u registar MAR?
5. U trenutku $T=114$ koji je ciklus na magistrali u toku? Zašto?
6. U trenutku $T=120$ koje su vrednosti signala mxBB0 i ldBB? Koja vrednost se upisuje u registar BB?
7. U trenutku $T=122$ koje su vrednosti signala or i ldAB? Koja vrednost se upisuje u registra AB? Zašto?
8. U trenutku $T=123$ koje su vrednosti signala ldN, ldZ, ldC i ldV? Zašto?
9. U trenutku $T=165$ koje su vrednosti signala mxADDA1, mxADDDB0, mxMAR2 i ldMAR? Koja se vrednost upisuje u registar MAR?
10. U trenutku $T=168$ koje su vrednosti signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registar MDR?
11. U trenutku $T=171$ koji je ciklus na magistrali u toku? Zašto?
12. U trenutku $T=175$ koja je vrednost u memorijskoj lokaciji 019Bh?

3. Instrukcija XOR

Cilj ovog primera je da se ilustruje logička operacija EKSKLUZIVNO ILI kao i neposredno, memorijsko indirektno i registersko direktno adresiranje.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, 8 bitni akumulator AB puni vrednošću $10011010_2 = 9Ah$. Nad sadržajem 8 bitnog akumulatora AB i lokacije čija se adresa nalazi na adresi 01AAh, koja ima vrednost $10100111_2 = A7h$, u okviru izvršavanja instrukcije XOR, koja koristi memorijsko indirektno adresiranje, vrši se operacija EKSKLUZIVNO ILI i rezultat $00111101_2 =$

3Dh se smešta u 8 bitni akumulator AB. Na kraju se rezultat prebacuje u registar R5 instrukcijom STB, koja koristi registarsko direktno adresiranje.

U memorijskim lokacijama 01A0h do 01A8h nalaze se instrukcije, memorijska lokacija 01A9h nije iskorišćena, u memorijskim lokacijama 01ABh i 01ACh nalaze se viši i niži bajt 16 bitne adrese operanda sa adrese 01ACh i na adresi 01ACh nalazi se 8 bitni operand A7h.

01A0 LDB imm(9A)	;AB = 9A	;20 E0 9A
01A3 XOR memind(01AA)	;AB = AB XOR mem[[01AA]]	;36 60 01 AA
01A7 STB regdir(R5)	;R5 = AB	;22 05
01A9		
01AA dw 01 AC	;16 bitna adresa operanda	;01 AC
01AC db A7	;8 bitni operand	;A7

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi na adresi 01ACh?
2. U trenutku T=32 koje vrednosti se nalaze na adresama 01AAh i 01ABh? Šta ove vrednosti predstavljaju?
3. U trenutku T=66 koje su vrednosti signala mxBB1 i ldBB? Koja vrednost se upisuje u registar BB?
4. U trenutku T=68 koje su vrednosti signala mxAB i ldAB? Koja se vrednost upisuje u registar AB?
5. U trenutku T=111 koje su vrednosti signala mxMAR1 i ldMAR? Koje se vrednosti upisuju u registar MAR?
6. U trenutku T=113 koji je ciklus u toku na magistrali? Zašto?
7. U trenutku T=118 koja je vrednost signala incMAR? Zašta se on akoristi?
8. U trenutku T=120 koji je ciklus u toku na magistrali? Zašto ?
9. U trenutku T=126 koje su vrednosti signala mxMAR1, mxMAR0 i ldMAR? Koja se vrednost upisuje u registar MAR?
10. U trenutku T=135 koje su vrednosti signala mxBB0 i ldBB? Koja se vrednost upisuje u registra BB ?
11. U trenutku T=137 koje su vrednosti signala xor i ldAB? Koja se vrednost upisuje u registar AB ?
12. U trenutku T=158 koja je vrednost signala ldGPRAR? Zašta se ona koristi?
13. U trenutku T=166 koja je vrednost signala wrGPR ? Koja se vrednost upisuje u registar R5 ?

4.Instrukcija NOT

Cilj ovog primera je da se ilustruje logička operacija NE kao i neposredno, registarsko direktno i indirektno i memorijsko direktno adresiranje.

U programu se najpre instrukcijom LDW, koja koristi neposredno adresiranje, vrednost 01BEh puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registarsko direktno adresiranje, vrednost se prebacije iz 16 bitnog akumulatora AW u registar R6. Instrukcijom LDB, koja koristi registarsko indirektno adresiranje, adresira se lokacija 01BEh, koristeći registar R6, i njenim sadržajem se puni 8 bitni akumulator AB. Instrukcijom NOT se nad sadržajem 8 bitnog akumulatora AB izvršava logička operacija NE i rezultat se smešta u 8 bitni akumulator AB. Na kraju se instrukcijom STB, koja koristi memorijsko direktno adresiranje, rezultat puni u memorijsku lokaciju 01BFh.

U memorijskim lokacijama 01B0h do 01BCh nalaze se instrukcije, memorijska lokacija 01BDh nije iskorišćena, u memorijskim lokacijama 01BEh i 01BFh nalaze se 8 bitni operand 01h i 8 bitni rezultat operacije 00h.

01B0 LDW imm(01BE)	;AW = 01BE	;21 E0 01 BE
01B4 STW regdir(R6)	;R6 = AW	;23 06
01B6 LDB regind(R6)	;AB = mem[R6]	;20 26
01B8 NOT	;AB = NOT AB	;37
01B9 STB memdir(01BF)	;mem[01BF] = AB	;22 40 01 BF
01BD		
01BE db 01	;8 bitni operand	;01
01BF db 00	;8 bitni rezultat	;00

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi u memorijskoj lokaciji 01BEh?
2. U trenutku T=32 koja vrednost se nalazi u memorijskoj lokaciji 01BFh?
3. U trenutku T=74 koje su vrednosti signala mxBW i ldBW? Koja se vrednost upisuje u registra BW ?
4. U trenutku T=96 koja je vrednost signala ldGPAR? Zašta se ona koristi?
5. U trenutku T=104 koja je vrednost signala mxGPR i wrGPR? Koja se vrednost upisuje u registar R6?
6. U trenutku T=129 koje su vrednosti signala mxMAR0 i ldMAR? Koja se vrednost upisuje u registar MAR?
7. U trenutku T=132 koji je ciklus u toku na magistrali? Zašto? U trenutku T=138 koja se vrednost upisuje u registar BB?
8. U trenutku T=140 koja je vrednost signala mxAB i ldAB? Koja se vrednost upisuje u registar AB?
9. U trenutku T=155 koja je vrednost signala not i ldAB? Koja se vrednost upisuje u registar AB?
10. U trenutku T=156 koje su vrednosti signala ldN, ldZ, ldC i ldV i zašto?
11. U trenutku T=198 koje su vrednosti signala mxMAR1 i ldMAR? Koja se vrednost upisuje u registar MAR?
12. U trenutku T=201 koja je vrednost signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registar MDR?
13. U trenutku T=203 koji je ciklus na magistrali u toku? Zašto?
14. U trenutku T=208 koja se vrednost nalazi u memorijskoj lokaciji 01BFh?

3)Pomeračke instrukcije

1.Instrukcija ASR

Cilj ovog primera je da se ilustruje operacija aritmetičkog pomeranja za jedno mesto udesno, kao i memorijsko direktno, registarsko direktno i indirektno i PC relativno adresiranje.

U programu se najpre instrukcijom LDW, koja koristi memorijsko direktno adresiranje, sadržaj memorijskih lokacija 01E1h i 01E2h, koje imaju vrednost 01E3h, puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registarsko direktno adresiranje, se sadržaj 16 bitnog akumulatora AW prebacuje u registra R7. Instrukcijom LDB, koja koristi registrasko indirektno adresiranje sa pomerajem, se

sadržaj lokacije 01E3h, koji ima vrednost 05h, puni u 8 bitni akumulator AB. Instrukcijom ASR, se nad sadržajem 8 bitnog akumulatora AB se izvršava operacija aritmetičkog pomeranja za jedno mesto udesno i rezultat 02h se smešta u 8 bitni akumulator AB. U okviru izvršavanja ove instrukcije se postavlja bit CF statusnog registra PSW, što se proverava u uslovnom skoku, odnosno instrukcijom BCR. Pri izvršavanju instrukcije BCR, uslov skoka nije ispunjen i prelazi se na izvršavanje sledeće instrukcije. Na kraju se instrukcijom STB, koja koristi PC relativno adresiranje, sadržaj 8 bitnog akumulatora AB se puni u lokaciju 01E3h.

U memorijskim lokacijama 01D0h do 01E0h nalaze se instrukcije, u memorijskim lokacijama 01E1h i 01E2h nalaze se viši i niži bajt 16 bitne adrese operanda 01E3h i u memorijskoj lokaciji 01E3h nalazi se 8 bitni operand 05h. U lokaciju 01E3h se na kraju smešta i rezultat operacije.

01D0 LDW memdir(01E1)	;AW = mem[01E0]	;21 40 01 E1
01D4 STW regdir(R7)	;R7 = AW	;23 07
01D6 LDB regindpom(R7, 0001)	;AB = mem[R7+0001]	;20 87 00 01
01DA ASR	;AB = ASR AB	;38
01DB BCR imm(FD)	;skok na adresu 01DA	;16 FD
01DD STB pcpc(0002)	;mem[PC+0002] = AB	;22 C0 00 02
01E1 dw 01 E3	;16 bitna adresa operanda	;01 E3
01E3 db 05	;8 bitni operand	;05

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi u memorijskoj lokaciji na adresi 01E3h?
2. U trenutku T=32 koja se vrednost nalazi u memorijskim lokacijama 01E1h i 01E2h? Šta one predstavljaju?
3. U trenutku T=73 koje su vrednosti signala mxMAR1 i ldMAR? Koja vrednost se upisuje u registar MAR?
4. U trenutku T=76 koji je ciklus u toku na magistrali i zašto?
5. U trenutku T=82 koja je vrednost signala incMAR? Zašta se ovaj signal koristi?
6. U trenutku T=84 koji je ciklus u toku na magistrali i zašto?
7. U trenutku T=90 koje su vrednosti signala mxBW0 i ldBW? Koja se vrednost upisuje u registar BW?
8. U trenutku T=112 koja je vrednost signala ldGPRAR? Zašta se on koristi?
9. U trenutku T=120 koja je vrednost signala mxGPR i wrGPR? Koja se vrednost upisuje u registra R7?
10. U trenutku T=162 koje su vrednosti signala mxADDA0, mxADDB0, mxMAR2 i ldMAR? Koja vrednost se upisuje u registar MAR?
11. U trenutku T=165 koji ciklus na magistrali je u toku? Zašto?
12. U trenutku T=171 koje su vrednosti signala mxBB0 i ldBB? Koja vrednost se upisuje u registar BB?
13. U trenutku T=172 koje su vrednosti signala mxAB i ldAB? Koja vrednost signala se upisuje u registar AB?
14. U trenutku T=188 koja je vrednost signala shr i ldC? Koja se vrednost upisuje u registar AB?
15. U trenutku T=189 koje su vrednosti signala ldN, ldZ i ldV? Zašta se one koriste?
16. U trenutku T=255 koje su vrednosti signala mxADDA1, mxADDB0, mxMAR2 i ldMAR? Koja vrednost se upisuje u registar MAR?

17. U trenutku T=258 koje su vrednosti signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registar MDR?
18. U trenutku T=260 koji ciklus je u toku na magistrali? Zašto?
19. U trenutku T=265 koja vrednost se nalazi u memorijskoj lokaciji 01E3h?

2. Instrukcija LSR

Cilj ovog primera je da se ilustruje pomeračka operacija logičkog pomeranja za jedno mesto udesno, kao i neposredno, registarsko direktno i bazno-indeksno adresiranje sa pomerajem.

U programu se najpre instrukcijom LDW, koja koristi neposredno adresiranje, vrednost 0200h puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registarsko direktno adresiranje, se sadržaj 16 bitnog akumulatora AW prebacuje u registar R2. Instrukcijom LDW, koja koristi neposredno adresiranje, se vrednost 0003h puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registrasko direktno adresiranje, se sadržaj 16 bitnog akumulatora AW prebacuje u registar R3. Instrukcijom LDB, koja koristi bazno-indeksno adresiranje sa pomerajem, se pomoću registara R2 i R3 8 bitni akumulator AB puni sadržajem memorijske lokacije 0205h, koji ima vrednost 05h. Instrukcijom LSR se nad sadržajem 8 bitnog akumulatora AB izvršava opracija logičkog pomeranja za jedno mesto udesno i rezultat se smešta u 8 bitni akumulator AB. Na kraju se instrukcijom STB, koja koristi direktno memorijsko adresiranje, sadržaj 8 bitnog akumulatora AB prebacuje u memorijsku lokaciju 0205h.

U memorijskim lokacijama 01F0h do 0204h nalaze se instrukcije i u memorijskoj lokaciji 0205h nalazi se 8 bitni operand 05h. U lokaciju 0205h se na kraju smešta i rezultat operacije.

01F0 LDW imm(0200)	;AW = 0200	;21 E0 02 00
01F4 STW regdir(R2)	;R2 = AW	;23 02
01F6 LDW imm(0003)	;AW = 0003	;21 E0 00 03
01FA STW regdir(R3)	;R3 = AW	;23 03
01FC LDB bxpom(R2, R3, 0002)	;AB = mem[R2+R3+0002]	;20 A2 00 02
0200 LSR	;AB = LSR AB	;39
0201 STB memdir(0205)	;mem[0205] = AB	;22 40 02 05
0205 db 05	;8 bitni operand	;05

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi u memorijskoj lokaciji na adresi 0205h?
2. U trenutku T=74 koje su vrednosti signala mxBW1 i ldBW? Koja se vrednost upisuje u registra BW?
3. U trenutku T=96 koja je vrednost signala ldGPRAR? Zašta se ona koristi?
4. U trenutku T=104 koje su vrednosti signala mxGPR i wrGPR? Koja se vrednost upisuje u registar R2?
5. U trenutku T=177 koja se vrednost upisuje u registar R3?
6. U trenutku T=218 koje su vrednosti signala mxADDA0, mxADDB0, ldCW i incGPRAR? Koji način adresiranja je u pitanju?
7. U trenutku T=220 koje su vrednosti signala mxADDA1, mxADDA0, mxADDB1, mxMAR2 i ldMAR? Koja vrednost se upisuje u registar MAR?
8. U trenutku T=223 koji je ciklus na magistrali u toku? Zašto?
9. U trenutku T=229 koje su vrednosti signala mxBB0 i ldBB? Koja vrednost se upisuje u registar BB?

10. U trenutku T=231 koje su vrednosti signala mxAB i ldAB? Koja vrednost signala se upisuje u registar AB?
11. U trenutku T=246 koje su vrednosti signala shr i ldC? Koja se vrednost upisuje u registar AB?
12. U trenutku T=247 koje su vrednosti signala ldN, ldZ i ldV?
13. U trenutku T=289 koje su vrednosti signala mxMAR1 i ldMAR? Koja se vrednost upisuje u registar MAR?
14. U trenutku T=292 koje su vrednosti signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registar MDR?
15. U trenutku T=294 koji je ciklus na magistrali u toku? Zašto?
16. U trenutku T=299 koja vrednost je na lokaciji 0205h?

3. Instrukcija ROR

Cilj ovog primera je da se ilustruje pomeračka operacija rotacije udesno za jedno mesto, kao i neposredno i PC relativno adresiranje.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 05h puni u 8 bitni akumulator AB. Nad sadržajem 8 bitnog akumulatora AB se, u okviru izvršavanja instrukcije ROR, izvršava operacija rotiranja za jedno mesto udesno i rezultat se smešta u 8 bitni akumulator AB. U okviru izvršavanja instrukcije ROR se postavlja indikator CF, što se proverava uslovnim skokom u okviru izvršavanja instrukcije BCR. Pri izvršavanju instrukcije BCR uslov skoka je ispunjen i izvršavanje programa se nastavlja na adresi 0213h. Instrukcija ROR se sadržaj 8 bitnog akumulatora AB ponovo rotira za jedno mesto udesno i rezultat upisuje u 8 bitni akumulator AB. U okviru izvršavanja instrukcije BCR, uslov skoka sada nije ispunjen i prelazi se na izvršavanje sledeće instrukcije. Na kraju se rezultat 41h, u okviru izvršavanja instrukcije STB, koja koristi PC relativno adresiranje, smešta na lokaciju 021Ah.

U memorijskim lokacijama 0210h do 0219h nalaze se instrukcije, a u memorijskoj lokaciji 021Ah nalazi se 8 bitni rezultat operacije.

0210 LDB imm(05)	;AB = 05	;20 E0 05
0213 ROR	;AB = ROR AB	;3A
0214 BCR imm(FD)	;skok na adresu 0213	;16 FD
0216 STB pcpom(0000)	;mem[PC+0000] = AB	;22 C0 00 00
021A db 00	;8 bitni rezultat	;00

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi u memorijskoj lokaciji na adresi 021A?
2. U trenutku T=66 koje su vrednosti signala mxBB1 i ldBB? Koja se vrednost upisuje u registar BB?
3. U trenutku T=68 koje su vrednosti signala mxAB i ldAB? Koja se vrednost upisuje u registar AB?
4. U trenutku T=69 koje su vrednosti signala ldN, ldZ, ldC i ldV? Zašto se one koriste?
5. U trenutku T=83 koje su vrednosti signala shr i ldC? Koja se vrednost upisuje u registar AB?
6. U trenutku T=84 koje su vrednosti signala ldN, ldZ i ldV? Zašto se one koriste?
7. U trenutku T=109 da li je ispunjen uslov skoka? Zašto se registra PC puni vrednošću 0213h?

8. U trenutku T=123 koje su vrednosti signala shr i ldC? Koja se vrednost upisuje u registar AB?
9. U trenutku T=124 da li je ispunjen uslov skoka ? Na kojoj adresi se nastavlja izvršavanje programa?
10. U trenutku T=190 koje su vrednosti signala mxADDA1, mxADDDB0, mxMAR2 i ldMAR? Koja se vrednost upisuje u registar MAR?
11. U trenutku T=193 koje su vrednosti signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registar MDR?
12. U trenutku T=196 koji ciklus na magistrali je utoku i zašto?
13. U trenutku T=200 koja vrednost se nalazi na adresi 021Ah?

4. Instrukcija RORC

Cilj ovog primera je da se ilustruje pomeračka operacija rotiranja za jedno mesto udesno kroz CF bit statusnog registra PSW, kao i neposredno i registarsko direktno adresiranje.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 05h puni u 8 bitni akumulator AB. U okviru izvršavanja instrukcije RORC se sadržaj 8 bitnog akumulatora AB rotira za jedno mesto udesno kroz CF bit statusnog registra PSW i rezultat se upisuje u 8 bitni akumulator AB. Na kraju se, instrukcijom STB, koja koristi registrasko direktno adresiranje, rezultat se smešta u registar R3.

U memorijskim lokacijama 0220h do 0225h nalaze se instrukcije programa.

0220 LDB imm(05)	;AB = 05h	;20 E0 05
0223 RORC	;AB = RORC AB	;3B
0224 STB regdir(R3)	;R3 = AB	;22 03

Pitanja:

1. U trenutku T=66 koje su vrednosti signala mxBB1 i ldBB? Koja vrednost se upisuje u registra BB?
2. U trenutku T=68 koje su vrednosti sugnala mxAB i ldAB? Koja se vrednost upisuje u registra AB ?
3. U trenutku T=83 koje su vrednosti signala shr i ldC? Koja se vrednost upisuje u registar AB ?
4. U trenutku T=84 koje su vrednosti signala ldN, ldZ i ldV? Zašta se one koriste?
5. U trenutku T=104 koja je vrednost signala ldGPRAR? Zašta se ona koristi?
6. U trenutku T=112 koja je vrednost signala wrGPR? Koja se vrednost upisuje u registra R3?

5. Instrukcija ASL

Cilj ovog primera je da se ilustruje operacija aritmetičkog pomeranja za jedno mesto ulevo, kao i neposredno, registarsko direktno i indirektno i memorijsko direktno adresiranje.

U programu se najpre instrukcijom LDW, koja koristi neposredno adresiranje, vrednost 023Eh puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registarsko direktno adresiranje, ova vrednost, koja predstavlja adresu, se prebacuje u registar R4. Registar R4 se koristi za adresiranje sadržaja memorijske lokacije 023Eh, koji ima vrednost 01h, u okviru izvršavanja instrukcije LDB, koja koristi registarsko indirektno adresiranje. Sadržaj ove lokacije se u okviru izvršavanja instrukcije LDB

puni u 8 bitni akumulator AB. Nad sadržajem 8 bitnog akumulatora AB vrši se operacija aritmetičkog pomeranja za jedno mesto ulevo, u okviru izvršavanja instrukcije ASL, i rezultat 02h se smešta u 8 bitni akumulator AB. Na kraju se sadržaj 8 bitnog akumulatora AB, instrukcijom STB, koja koristi memorijsko direktno adresiranje, smešta u memorijsku lokaciju 023Eh.

U memorijskim lokacijama 0230h do 023Ch nalaze se instrukcije, memorijska lokacija 023D nije iskorišćena, a u memorijskoj lokaciji 023Eh nalazi se 8 bitni operand 01h. U lokaciju 023Eh se na kraju programa smešta i rezultat.

0230 LDW imm(023E)	;AW = 023E	;21 E0 02 3E
0234 STW regdir(R4)	;R4 = AW	;23 04
0236 LDB regind(R4)	;AB = mem[R4]	;20 24
0238 ASL	;AB = ASL AB	;3C
0239 STB memdir(023E)	;mem[023E] = AB	;22 40 02 3E
023D		
023E db 01	;8 bitni rezultat	;01

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi u memorijskoj lokaciji na adresi 023Eh?
2. U trenutku T=74 koje su vrednosti signala mxBW1 i ldBW? Koja se vrednost upisuje u registar BW ?
3. U trenutku T=76 koja je vrednost signala ldAW? Koja se vrednost upisuje u registar AW ?
4. U trenutku T=96 koja je vrednost signala ldGPRAR? Zašta se ona koristi?
5. U trenutku T=104 koje su vrednosti signala mxGPR i ldGPR ? Koja se vrednost upisuje u registra R4?
6. U trenutku T=129 koje su vrednosti signala mxMAR0 i ldMAR? Koja se vrednost upisuje u registar MAR?
7. U trenutku T=132 koji ciklus na magistrali je u toku ? Zašto ?
8. U trenutku T=138 koja je vrednost signala mxBB0 i ldBB ? Koja se vrednost upisuje u registar BB?
9. U trenutku T=140 koje su vrednosti signala mxAB i ldAB ? Koja se vrednost upisuje u registar AB?
10. U trenutku T=155 koje su vrednosti signala shl i ldC? Koja se vrednost upisuje u registar AB?
11. U trenutku T=156 koje su vrednosti signala ldN, ldZ i ldV? Zašta se oni koriste?
12. U trenutku T=198 koje su vrednosti signala mxMAR1 i ldMAR? Koja se vrednost upisuje u registar MAR?
13. U trenutku T=201 koje su vrednosti signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registra MDR?
14. U trenutku T=203 koji je ciklus na magistrali u toku? Zašto?
15. U trenutku T=208 koja se vrednost nalazi u memorijskoj lokaciji sa adresom 023Eh?

6. Instrukcija LSL

Cilj ovog primera je da se ilustruje operacija logičkog pomeranja za jedno mesto ulevo, kao i neposredno, registarsko direktno i indirektno adresiranje sa pomerajem i memorijsko direktno adresiranje.

U programu se najpre instrukcijom LDW, koja koristi neposredno adresiranje, vrednost 0250h puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registrasko direktno adresiranje, se sadržajem 16 bitnog akumulatora AW puni registar R4. Registar R4 se koristi za adresiranje sadržaja lokacije 0252h, koja ima vrednost 01h, u okviru izvršavanja instrukcije LDB, koja koristi registrasko indirektno adresiranje sa pomerajem. U okviru izvršavanja instrukcije LDB se sadržaj pomenute memorijske lokacije puni u 8 bitni akumulator AB. Nad sadržajem 8 bitnog akumulatora AB se izvršava operacija logičkog pomeranja u levo za jedno mesto u okviru izvršavanja instrukcije LSL i rezultat se smešta u 8 bitni akumulator AB. Instrukcija LSL postavlja indikator CF. Proverom ovog indikatora utvrđuje se da uslov skoka nije ispunjen u okviru izvršavanja instrukcije BCR i prelazi se na izvršavanje sledeće instrukcije. Na kraju se instrukcijom STB, koja koristi memorijsko direktno adresiranje, rezultat 02h smešta na lokaciju 0252h.

U memorijskim lokacijama 0240h do 0250h nalaze se instrukcije, memorijska lokacija 0251h se ne koristi a u memorijskoj lokaciji 0252h se nalazi 8 bitni operand 01h. U lokaciju 0252h se na kraju izvršavanja programa smešta i rezultat.

0240 LDW imm(0250)	;AW = 0250h	;21 E0 02 50
0244 STW regdir(R4)	;R4 = 0250h	;23 04
0246 LDB regindpom(R4, 0002)	;AB = mem[R4+0002]	;20 84 00 02
024A LSL	;AB = LSL AB	;3D
024B BCR imm(FD)	;skok na adresu 024Ah	;16 FD
024D STB memdir(0252)	;mem[0252] = AB	;22 40 02 52
0251		
0252 db 01	;8 bitni operand	;01

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi u memorijskoj lokaciji sa adresom 0251h?
2. U trenutku T=32 koja se vrednost nalazi u memorijskoj lokaciji sa adresom 0252h?
3. U trenutku T=74 koje su vrednosti signala mxBW1 i ldBW? Koja se vrednost upisuje u registar BW ?
4. U trenutku T=76 koja je vrednost signala ldAW? Koja se vrednost upisuje u registra AW?
5. U trenutku T=96 koja je vrednost signala ldGPRAR? Zašta se ona koristi?
6. U trenutku T=104 koje su vrednosti signala mxGPR i wrGPR? Koja se vrednost upisuje u registar R4?
7. U trenutku T=146 koje su vrednosti signala mxADDA0, mxADDB0, mxMAR2 i ldMAR? Koja se vrednost upisuje u MAR registar?
8. U trenutku T=149 koje je ciklus na magistrali u toku? Zašto?
9. U trenutku T=155 koje su vrednosti signala mxBB0 i ldBB? Koja se vrednost upisuje u registra BB?
10. U trenutku T=157 koje su vrednosti signala mxAB i ldAB? Koja se vrednost upisuje u registra AB?
11. U trenutku T=172 koje su vrednosti signala shl i ldC? Koja se vrednost upisuje u registra AB?
12. U trenutku T=173 koje su vrednosti signal ldN, ldZ i ldV? Zašta se one koriste?
13. U trenutku T=174 da li je ispunjen uslov skoka? Zašto?
14. U trenutku T=239 koje su vrednosti signala mxMAR1 i ldMAR ? Koja se vrednost upisuje u registar MAR?

15. U trenutku T=242 koje su vrednosti signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registra MDR?
16. U trenutku T=244 koji je ciklus na magistrali u toku? Zašto?
17. U trenutku T=249 koja je vrednost na adresi 0252h?

7. Instrukcija ROL

Cilj ovog primera je da se ilustruje operacija rotiranja za jedno mesto ulevo, kao i neposredno, registarsko direktno i bazno-indeksno adresiranje sa pomerajem i PC relativno adresiranje.

U programu se najpre instrukcijom LDW, koja koristi neposredno adresiranje, vrednost 0270h puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registarsko direktno adresiranje se sadržaj 16 bitnog akumulatora AW prebacuje u registar R5. Instrukcijom LDW, koja koristi neposredno adresiranje, se vrednost 0003h puni u 16 bitni akumulator AW. Instrukcijom STW, koja koristi registarsko direktno adresiranje, se sadržaj 16 bitnog akumulatora AW prebacuje u registar R6. U okviru izvršavanja instrukcije LDB, koja koristi bazno-indeksno adresiranje sa pomerajem, registri R5 i R6 se koriste za adresiranje sadržaja memorijske lokacije 0275h, koja ima vrednost 12h. U okviru izvršavanja instrukcije LDB se sadržaj pomenute memorijske lokacije prebacuje u 8 bitni akumulator AB. Nad sadržajem 8 bitnog akumulatora AB izvršava se operacija rotiranja za jedno mesto ulevo, u okviru izvršavanja instrukcije ROL, i rezultat 24h se smešta u 8 bitni akumulator AB. Na kraju se instrukcijom STB, koja koristi PC relativno adresiranje, sadržaj 8 bitnog akumulatora AB se smešta na lokaciju 0275h.

U memorijskim lokacijama 0260h do 0274h nalaze se instrukcije, a u memorijskoj lokaciji 0275h nalazi se 8 bitni operand.

0260 LDW imm(0270)	;AW = 0270	;21 E0 02 70
0264 STW regdir(R5)	;R5 = AW	;23 05
0266 LDW imm(0003)	;AW = 0003	;21 E0 00 03
026A STW regdir(R6)	;R6 = AW	;23 06
026C LDB bxpom(R5, R6, 0002)	;AB = mem[R5+R6+0002]	;20 A5 00 02
0270 ROL	;AB = ROL AB	;3E
0271 STB pcpom(0000)	;mem[PC+0000] = AB	;22 C0 00 00
0275 db 12	;8 bitni operand	;12

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi na adresi 0275h?
2. U trenutku T=74 koje su vrednosti signala mxBW1 i ldBW? Koja se vrednost upisuje u registra BW ?
3. U trenutku T=76 koja je vrednost signala ldAW? Koja se vrednost upisuje u registra AW?
4. U trenutku T=96 koja je vrednost signala ldGPRAR? Zašta se ona koristi?
5. U trenutku T=104 koja je vrednost signala mxGPR i wrGPR? Koja se vrednost upisuje u registra R5 ?
6. U trenutku T=177 koja se vrednost upisuje u registar R6 ?
7. U trenutku T=218 koje su vrednosti signala mxADDA=, mxADDDB0, ldCW i inc GPRAR ? Koji način adresiranja je pitanju ?
8. U trenutku T=220 koje su vrednosti signala mxADDA1, mxADDA0, mxADDDB1, mxMAR2 i ldMAR ? Koja se vrednost upisuje u registar MAR?

9. U trenutku T=223 koji je ciklus u toku na magistrali i zašto?
10. U trenutku T=229 koje su vrednosti signala mxBB0 i ldBB ? Koja se vrednost upisuje u registar BB?
11. U trenutku T=231 koje su vrednosti signala mxAB i ldAB ? Koja se vrednost upisuje u registra AB ?
12. U trenutku T=246 koje su vrednosti signala shl i ldC ? Koja se vrednost upisuje u registra AB ?
13. U trenutku T=247 koja je vrednost signala ldN, ldZ i ldV ? Zašta se oni koriste?
14. U trenutku T=289 koja je vrednost signala mxADDA1, mxADDDB0, mxMAR2 i ldMAR ? Koja se vrednost upisuje u registra MAR?
15. U trenutku T=292 koja je vrednost signala mxMDR0 i ldMDR ? Koja se vrednost upisuje u registar MDR ?
16. U trenutku T=294 koji je ciklus na magistrali u toku ? Zašto ?
17. U trenutku T=299 koja se vrednost nalazi na lokaciji 0275h ?

8. Instrukcija ROLC

Cilj ovog primera je da se ilustruje operacija rotiranja ulevo kroz bit CF statusnog registra PSW, kao i neposredno i PC relativno adresiranje.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 12h puni u 8 bitni akumulator AB. U okviru izvršavanja instrukcije ROLC, se nad sadržajem 8 bitnog akumulatora AB izvršava operacija rotiranja u levo za jedno mesto kroz bit CF statusnog registra PSW i rezultat 24h se smešta u 8 bitni akumulator AB. Na kraju se instrukcijom STB, koja koristi PC relativno adresiranje, sadržaj 8 bitnog akumulatora AB se smešta na adresu 0288h.

U memorijskim lokacijama 0280h do 0287h nalaza se instrukcije, a u memorijskoj lokaciji 0288h se smešta rezultat.

0280 LDB imm(12)	;AB = 12	;20 E0 12
0283 ROLC	;AB = ROLC AB	;3F
0284 STB pcgom(0000)	;mem[PC+0000] = AB	;22 C0 00 00
0288 db 00	;8 bitni rezultat	;00

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi u memorijskoj lokaciji na adresi 0288h?
2. U trenutku T=66 koje su vrednosti signala mxBB1 i ldBB? Koja se vrednost upisuje u registar BB?
3. U trenutku T=68 koje su vrednosti signala mxAB i ldAB? Koje se vrednosti upisuju u registar AB?
4. U trenutku T=83 koje su vrednosti signala shl i ldC ? Koje se vrednosti upisuju u registar AB?
5. U trenutku T=84 koje su vrednosti signala ldN, ldZ i ldV? Zašto?
6. U trenutku T=126 koje su vrednosti signala mxADDA1, mxADDDB0, mxMAR2 i ldMAR? Koja se vrednost upisuje u registar MAR?
7. U trenutku T=129 koje su vrednosti signala mxMDR0 i ldMDR? Koja se vrednost upisuje u registra MDR?
8. U trenutku T=131 koji je ciklus na magistrali u toku? Zašto?
9. U trenutku T=136 koja se vrednost nalazi na lokaciji 0288h?

4) Instrukcije skoka i skoka na potprogram

1. Instrukcije JSR i RTS

Cilj ovog primera je da se ilustruju instrukcije skoka i povratka iz potprograma JSR i RTS.

U programu se u okviru izvršavanja instrukcije JSR skače na potprogram na adresi 0295h, a u potprogramu se u okviru izvršavanja instrukcije RTS vraća nazad na adresu 0293h gde se nastavlja izvršavanje programa.

U memorijskim lokacijama 0290h do 0292h i 0295h su smeštene instrukcije, a memorijske lokacije 0293h i 0294h nisu iskorišćene.

```
0290 JSR imm(0295)      ;0A 02 05
0293
0294
0295 RTS                ;0B
```

Pitanja:

1. U trenutku T=34 koje su vrednosti signala ldMAR i incPC? Šta se njima postiže?
2. U trenutku T=36 koji je ciklus u toku na magistrali? Zašto?
3. U trenutku T=41 koja je vrednost signala ldIR0? Zašta se on koristi?
4. U trenutku T=42 koja je vrednost signala gropr ? Šta ona pokazuje?
5. U trenutku T=44 koje su vrednosti signala ldMAR i incPC? Šta se njima postiže?
6. U trenutku T=46 koji je ciklus u toku na magistrali? Zašto?
7. U trenutku T=51 koja je vrednost signala ldIR1 ? Zašta se on koristi ?
8. U trenutku T=52 koja je vrednost signala gradr ? Šta ona pokazuje ?
9. U trenutku T=53 koje su vrednosti signala l2_brnch i l2_arlog ? Šta te vrednosti govore ?
10. U trenutku T=62 koje su vrednosti signala ldIR2, l3_jump i l3_arlog? Šta se događa ?
11. U trenutku T=64 koja je vrednost signala incSP? Zašta ona služi?
12. U trenutku T=65 koje su vrednosti signala mxMAR2, mxMAR0, ldMAR, mxMADR2 i ldMADR ? Koje se vrednosti upisuju u registra MAR ? A koje u registra MDR ?
13. U trenutku T=67 koji je ciklus na magistrali u toku? Zašto?
14. U trenutku T=80 koje su vrednosti signala mxPC1 i ldPC ? Koja se vrednost upisuje u registra PC ? Na kojoj se adresi nastavlja izvršavanje instrukcija?
15. U trenutku T=94 koje su vrednosti signala mxMAR2, mxMAR0, ldMAR i decSP ? Koja se vrednost upisuje u registra MAR? Zašto se dekrementira registar SP?
16. U trenutku T=96 koji je ciklus u toku na magistrali? Zašto?
17. U trenutku T=114 koja je vrednost signala ldPC ? Koja se vrednost upisuje u registra PC ? Zašto ?

2. Instrukcije JMP i BNEQL

Cilj ovog primera je da se ilustruje korišćenje naredbi bezuslovnog skoka JMP i uslovnog skoka BNEQL, kao i neposrednog i memorijskog direktnog adresiranja.

U programu se najpre instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 05h puni u 8 bitni akumulator AB. Od sadržaja 8 bitnog akumulatora AB se, u okviru izvršavanja instrukcije SUB, koja koristi memorijsko direktno adresiranje, oduzima sadržaj lokacije 02B9h, koji ima vrednost 05h i rezultat smešta u 8 bitni akumulator AB. U okviru izvršavanja instrukcije SUB postavlja se indikator ZF.

Vrednost indikatora se proverava uslovnim skokom, u okviru izvršavanja instrukcije BNEQL. Pri izvršavanju instrukcije BNEQL uslov skoka nije ispunjen i izvršavanje se nastavlja na sledećoj instrukciji programa. Instrukcijom LDB, koja koristi neposredno adresiranje, vrednost 06h se puni u 8 bitni akumulator AB. U okviru izvršavanja instrukcije SUB, koja koristi memorijsko direktno adresiranje, se od sadržaja 8 bitnog akumulatora AB oduzima sadržaj lokacije 02B9h i rezultat smešta u 8 bitni akumulator AB. Instrukcijom LDB, koja koristi neposredno adresiranje, se vrednost 06h puni u 8 bitni akumulator AB. Instrukcijom STB, koja koristi memorijsko direktno adresiranje, sadržaj 8 bitnog akumulatora AB se prebacuje u memorijsku lokaciju 02B9h. U okviru izvršavanja instrukcije BNEQL uslov skoka je sada ispunjen i izvršavanje programa se nastavlja na adresi 02BAh. U okviru izvršavanja instrukcije JMP se, sa adrese 02BAh, bezuslovno skače nazad na adresu 02A9h. Na ovoj adresi instrukcijom LDB, koja koristi neposredno adresiranje, 8 bitni akumulator AB se ponovo puni vrednošću 06h. U okviru izvršavanja instrukcije SUB, koja koristi memorijsko direktno adresiranje, od sadržaja 8 bitnog akumulatora AB se oduzima sadržaj lokacije 02B9h, koji sada ima vrednost 06h i rezultat se smešta u 8 bitni akumulator AB. U okviru izvršavanja instrukcije BNEQL uslov skoka sada nije ispunjen i izvršavanje se nastavlja na sledećoj adresi.

U memorijskim lokacijama 02A0h do 02B8h i 02BAh do 02BCh nalaze se instrukcije, a u memorijskoj lokaciji 02B9h nalazi se 8 bitni operand 05h.

02A0 LDB imm(05)	;AB = 05	;20 E0 05
02A3 SUB memdir(02B9)	;AB = AB - mem[02B9]	;31 40 02 B9
02A7 BNEQL imm(11)	;skok na adresu 02BA	;11 11
02A9 LDB imm(06)	;AB = 06h	;20 E0 06
02AC SUB memdir(02B9)	;AB = AB - mem[02B4]	;31 40 02 B9
02B0 LDB imm(06)	;AB = 06h	;20 E0 06
02B3 STB memdir(02B9)	;mem[02B9] = AB	;22 40 02 B9
02B7 BNEQL imm(01)	;skok na adresu 02BA	;11 01
02B9 db 05	;8 bitni operand	;05
02BA JMP imm(02A9)	;skok na adresu 02A9	;09 02 A9

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi na adresi 02B9h?
2. U trenutku T=66 koje su vrednosti signala mxBB1 i ldBB? Koja se vrednost upisuje u registra BB?
3. U trenutku T=111 koje su vrednosti signala mxMAR1 i ldMAR? Koja se vrednost upisuje u registar MAR?
4. U trenutku T=114 koji je ciklus na magistrali u toku? Zašto?
5. U trenutku T=120 koje su vrednosti signala mxBB0 i ldBB? Koja se vrednost upisuje u registra BB ?
6. U trenutku T=122 koje su vrednosti signala sub i ldAB ? Koja se vrednost upisuje u registar AB?
7. U trenutku T=123 koja je vrednost signala ldZ ? Zašta se on koristi?
8. U trenutku T=123 koja je vrednost signala brpom ? Da li je ispunjen uslov skoka? Zašto ?
9. U trenutku T=344 koja je vrednost signala brpom ? Da li je sada ispunjen uslov skoka ? Zašto?

10. U trenutku T=354 koje su vrednosti signala mxADDA1, mxADDB1, mxADDB0, mxPC0 i ldPC? Koja se vrednost upisuje u registar PC? Gde se nastavlja izvršavanje programa?

11. U trenutku T= 387koje su vrednosti signala mxPC1 i ldPC? Koja se vrednost upisuje u registar PC? Gde se nastavlja izvršavanje programa?

12. U trenutku T=479 koja je vrednost signala brpom? Da li je uslov skoka ispunjen? Zašto?

5)Instrukcije za rad sa stekom

1.Instrukcija PUSHB

Cilj ovog primera je da se ilustruje korišćenje operacije stavljanja bajta na stek, kao i PC relativnog i memorijskog direktnog adresiranja.

U programu se najpre instrukcijom LDW, koja koristi memorijsko direktno adresiranje, puni 16 bitni akumulator AW vrednošću 02CAh, koja pokazuje iza kraja programa. Dalje se instrukcijom STSP sadržaj 16 bitnog akumulatora AW prebacuje u registar SP. Instrukcijom LDB, koja koristi PC relativno adersiranje, 8 bitni akumulator AB se puni sadržajem memorijske lokacije 02CAh, koja ima vrednost 02h. U okviru izvršavanja instrukcije PUSHB sadržaj 8 bitnog akumulatora AB se stavlja na stek, odnosno u memorijsku lokaciju 02CBh.

U memorijskim lokacijama 02C0h do 02C9h nalaze se instrukcije, a u lokacijama 02CAh i 02CBh 16 bitna adresa operanda.

02C0 LDW memdir(02CA)	;AW = mem[02CA]	;21 40 02 CA
02C4 STSP	;SP = AW	;2D
02C5 LDB pcpom(0001)	;AB = mem[PC+0001]	;20 C0 00 01
02C9 PUSHB	;mem[SP+1] = AB	;26
02CA dw 02 CA	;16 bitna adresa operanda	;02 CA

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi na adresi 02CAh? A koja na adresi 02CBh? Šta ove vrednosti predstavljaju?

2. U trenutku T=90 koja je vrednsot signala mxBW1 i ldBW? Koja se vrednost upisuje u registra BW?

3. U trenutku T=92 koja je vrednsot signala ldAW? Koja se vrednost upisuje u registra AW?

4. U trenutku T=106 koja je vrednost signala ldSP? Koja se vrednost upisuje u registra SP?

5. U trenutku T=148 koje su vrednosti signala mxADDA1, mxADDB0, mxMAR2 i ldMAR? Koja se vrednost upisuje u registra MAR?

6. U trenutku T=150 koji je ciklus na magistrali u toku? Zašto?

7. U trenutku T=156 koje su vrednosti signala mxBB0 i ldBB? Koja se vrednost upisuje u registra BB?

8. U trenutku T=158 koje su vrednosti signala mxAB i ldAB? Koja se vrednost upisuje u registra AB?

9. U trenutku T=173 koja je vrednost signlala incSP? Zašta se on koristi?

10. U trenutku T=174 koje su vrednosti signala mxMAR2, mxMAR0, ldMAR, mxMDR0 i ldMADR? Koja se vrednost upisuje u registar MAR? A koja u registar MDR?

11. U trenutku T=177 koji je ciklus na magistrali u toku? Zašto ?
12. U trenutku T=181 koja se vrednost nalazi na adresi 02CBh?

2. Instrukcija POPB

Cilj ovog primera je da se ilustruje korišćenje naredbe skidanja bajta sa steka, kao i PC relativnog i memorijskog direktnog adresiranja.

U programu se najpre instrukcijom LDW, koja koristi neposredno adresiranje, 16 bitni akumulator AW puni vrednošću 02DBh, koja pokazuje iza kraja programa. Instrukcijom STSP se sadržaj 16 bitnog akumulatora AW prebacuje u registar SP. U okviru izvršavanja instrukcije POPB vrednost 01h se skida sa steka, odnosno memorijske lokacije 02DBh, i smešta u 8 bitni akumulator AB. Na kraju se instrukcijom STB, koja koristi memorijsko direktno adresiranje, sadržaj 8 bitnog akumulatora AB smešta u memorijsku lokaciju 02DAh.

U memorijskim lokacijama 02D0h do 02D9h nalaze se instrukcije a u memorijskim lokacijama 02DAh i 02DBh 8 bitni rezultat operacije i operand.

02D0 LDW imm(02DB)	;AW = 02DAB	;21 E0 02 DB
02D4 STSP	;SP = AW	;2D
02D5 POPB	;AB = mem[SP+1]	;24
02D6 STB memdir(02DA)	;mem[02DA] = AB	;22 40 02 DA
02DA db 02	;8 bitni operand	;02
02DB db 01	;8 bitni rezultat	;01

Pitanja:

1. U trenutku T=32 koja je vrednost na adresi 02DAh? A koja na adresi 02DBh? Šta ove adrese predstavljaju?
2. U trenutku T=74 koje su vrednost signala mxBW1 i ldBW? Koja se vrednost upisuje u registra BW?
3. U trenutku T=76 koja je vrednost signala ldAW? Koja se vrednost upisuje u registra AW?
4. U trenutku T=90 koja je vrednost signala ldSP? Koja se vrednost upisuje u registra SP?
5. U trenutku T=104 koja je vrednost signala mxMAR2, mxMAR0, ldMAR i dec SP? Koja se vrednost upisuje u registar MAR? Zašto se dekrementira registrar SP?
6. U trenutku T=106 koji se ciklus na magistrali odvija? Zašto?
7. U trenutku T=111 koje su vrednosti signala mxBB0 i ldBB? Koja se vrednost upisuje u registra BB?
8. U trenutku T=112 koje su vrednosti signala mxAB i ldAB? Koja se vrednost upisuje u registra AB ?
9. U trenutku T=155 koje su vrednosti signala mxMAR1 i ldMAR ? Koja se vrednost upisuje u registra MAR?
10. U trenutku T=158 koje su vrednosti signala mxMDR0 i ldMDR ? Koja se vrednost upisuje u registar MDR?
11. U trenutku T=160 koji je ciklus na magistrali u toku? Zašto?
12. U trenutku T=165 koja je vrednost na adresi 02DAh ?

2. Programi za testiranje ulaza i izlaza

Sledeća tri programa ilustruju organizaciju ulaza i izlaza sa i na periferije ovog računarskog sistema. U sistemu postoje periferije sa kontrolerima sa i bez direktnog pristupa meoriji. Prvi program ilustruje ulaz sa periferije bez DMA kontrolera u memoriju pomoću tehnike ispitivanja bita spremnosti. Drugi primer ilustruje izlaz na periferiju sa kontrolerom bez DMA iz memorije pomoću tehnike generisanja prekida. Treći program ilustruje ulaz sa periferije sa DMA kontrolerom u memoriju.

1) Ulaz sa testiranjem bita spremnosti sa kontrolerom bez DMA

Cilj primera je da se ilustruje ulaz niza bajtova sa periferije u memoriju korišćenjem kontrolera periferije bez DMA i ispitivanjem bita spremnosti.

U programu se koristi uređaj broj 0 (0-64). Za startovanje periferije u odgovarajućem režimu i njeno zaustavljanje koriste se kontrolne reči: Start reč 05h, Stop reč 01h. Za ispitivanje bita spremnosti statusnog registra periferije koristi se maska 01h. Adrese registara kontrolera korišćene periferije su: Adrese Kontrolnog registra F000, Statusnog registra F001, Registra podataka F002. Adresa bloka podataka 0543 se puni u R1, dužina bloka podataka se puni u R2, a vrednost 0 se puni u R3. Start reč se upisuje u kontrolni registar periferije i ona se startuje. Proverava se bit spremnost dok podatak ne bude spreman. Podatak se učitava i smešta na željenu adresu pomoću registra R1 i R3 i sadržaji registara R3 i R2 se ažuriraju. Proverava se da li je ostalo još podataka za prenos. Ako jeste ponovo se čeka na statusni registar, a ako nije program se završava.

U memorijskim lokacijama 0500h do 053Fh nalaze se instrukcije, memorijske lokacije 0540h do 0542h nisu iskorišćene a lokacije 0543h do 0547h se koriste za smeštanje bloka podataka sa periferije.

0500 LDW imm(0543)	;AW = 0543h;adresa bloka	;21 E0 05 43
0504 STW regdir(R1)	;R1 = AW	;23 01
0506 LDB imm(05)	;AB = 05h;dužina bloka	;20 E0 05
0509 STB regdir(R3)	;R3 = AB	;22 03
050B LDW imm(0000)	;AW = 0000h;indeks	;21 E0 00 00
050F STW regdir(R2)	;R2 = AW	;23 02
0511 LDB imm(05)	;AB = 05h ;start reč	;20 E0 05
0514 STB memdir(F000)	;mem[F000] = AB;CR	;22 40 F0 00
0518 LDB memdir(F001)	;AB = mem[F001];SR	;20 40 F0 01
051C AND imm(01)	;AB = 01h AND AB	;34 E0 01
051F BEQL imm(F7)	;skok na adresu 0518	;10 F7
0521 LDB memdir(F002)	;AB = mem[F002];DR	;20 40 F0 02
0525 STB bxpom(R1, R2)	;mem[R1+R2] = AB	;22 A1 00 00
0529 LDB regdir(R2)	;AB = R2	;20 02
052B INC	;AB = AB + 1	;32
052C STB regdir(R2)	;R2 = AB	;22 02
052E LDB regdir(R3)	;AB = R3	;20 03
0530 DEC	;AB = AB - 1	;33
0531 STB regdir(R3)	;R3 = AB	;22 03
0533 BNEQL imm(E3)	;skok na adresu 0518	;11 E3
0535 LDB memdir(F000)	;AB = mem[F000];CR	;20 40 F0 00
0539 AND imm(FB)	;AB = FBh AND AB;stop reč	;34 E0 FB
053C STB memdir(F000)	;mem[F000] = AB;CR	;22 40 F0 00

0540		
0541		
0542		
0543 db 00	;8 bitni podatak	;00
0544 db 00	;8 bitni podatak	;00
0545 db 00	;8 bitni podatak	;00
0546 db 00	;8 bitni podatak	;00
0547 db 00	;8 bitni podatak	;00

Pitanja:

1. U trenutku T=32 koje se vrednosti nalaze na memorijskim lokacijama 0543h, 0544h, 0545h, 0546h i 0547h?
2. U trenutku T=104 koja je vrednost registar R1 i šta ona predstavlja?
3. U trenutku T=170 koja je vrednost registra R3 i šta ona predstavlja?
4. U trenutku T=243 koja je vrednost registra R2 i šta ona predstavlja?
5. U trenutku T=328 šta se upisuje na adresu F000h? Šta predstavlja ta adresa? Šta predstavlja upisana vrednost?
6. U trenutku T=374 šta se nalazi na adresi F001h? Šta predstavlja ta adresa i zašto se njena vrednost učitava u akumulator?
7. U trenutku T=435 da li je ispunjen uslov skoka i zašto?
8. U trenutku T=486 šta se nalazi na adresi F002h? Šta predstavlja ta adresa i zašto se taj podatak učitava u akumulator?
9. U trenutku T=617 koja se vrednost upisuje u registar R2?
10. U trenutku T=690 koja se vrednost upisuje u registar R3?
11. Na koji način su se ove vrednosti promenile i zašto?
12. U trenutku T=717 da li je ispunjen uslov skoka i zašto?
13. U trenutku T=2263 da li je ispunjen uslov skoka i zašto? Kako su se promenile vrednosti memorijskih lokacija 0543h, 0544h, 0545h, 0546h i 0547h od trenutka T=32?
14. U trenutku T=2393 šta se upisuje na adresu F000 i šta predstavlja ta vrednost?

2) Izlaz sa generisanjem prekida sa kontrolerom bez DMA

Cilj primera je da se ilustruje izlaz niza bajtova iz memorije u periferiju korišćenjem kontrolera periferije bez DMA sa mehanizmom prekida.

U programu se koristi uređaj broj 0 (0-64). Za startovanje periferije u odgovarajućem režimu i njeno zaustavljanje koriste se kontrolne reči: Start reč 06h, Stop reč 02h. Za ispitivanje bita spremnosti statusnog registra periferije koristi se maska 01h. Adrese registara kontrolera korišćene periferije su: Adrese Kontrolnog registra F000, Statusnog registra F001, Registra podataka F002. Primer se sastoji od glavnog programa i prekidne rutine. U glavnom programu se prvo postavljaju jedinica u registar maske na odgovarajuću poziciju da bi se demaskirao prekid i postavlja se jedinica u PSWI da bi se prekidi uopšte omogućili. Na odgovarajuće memorijske lokacije se dalje postavljaju adresa početka memorijskog bloka i njegova dužina. Postavlja se i vrednost promenljive semafor u memoriji na 01h. Periferija se pokrene upisivanjem startne kontrolne reči u kontrolni registar kontrolera periferije i proverava se vrednost semafora. U prekidnoj rutini se podaci prebacuju na periferiju, ažuriraju memorijske lokacije sa trenutnom adresom za prebacivanje i dužinom niza podataka. Kada se prebace svi podaci postavlja se vrednost semafora na 00h i periferija se

zaustavlja. Kako se koristi mehanizam prekida treba obezbediti da se pri prekidu skoči na pravu prekidnu rutinu. Pre početka izvršavanja programa treba postati ulaz 5 u tabeli prekidnih rutina, koji ova periferija fiksno koristi, na adresu prekidne rutine 0550h koja treba pri prekidu da se izvrši.

U memorijskim lokacijama 054Ah do 057Bh i 0590h do 05ABh se nalaze instrukcije, memorijske lokacije 057Ch i 057Dh su neiskorišćene, u memorijskim lokacijama 057Fh i 0580h se nalaze viši i niži bajt 16 bitne adrese bloka podataka, u memorijskoj lokaciji 0581h nalazi se 8 bitna dužina bloka podataka, u memorijskoj lokaciji 0582h se nalazi 8 bitna semafora promenljiva a u memorijskim lokacijama od 0583h do 0587h se nalazi blok podataka za izlaz na periferiju.

;Glavni Program

054A LDW imm(000B)	;AW = 000Bh;demaskiranje prekida	;21 E0 00 0B
054E STIMR	;IMR = AW	;2B
054F INTE	;PSWI = 1	;05
0550 LDB imm(05)	;AB = 05h;adresa bloka high	;20 E0 05
0553 STB memdir(057F)	;mem[057F] = AB	;22 40 05 7F
0557 LDB imm(83)	;AB = 83h;adresa bloka low	;20 E0 83
055A STB memdir(0580)	;mem[0580] = AB	;22 40 05 80
055E LDB imm(05)	;AB = 05h;dužina bloka	;20 E0 05
0561 STB memdir(0581)	;mem[0581] = AB	;22 40 05 81
0565 LDB imm(01)	;AB = 01h;semafor	;20 E0 01
0568 STB memdir(0582)	;mem[0582] = AB	;22 40 05 82
056C LDB imm(06)	;AB = 06h;start reč	;20 E0 06
056F STB memdir(F000)	;mem[F000] = AB;CR	;22 40 F0 00
0573 LDB memdir(0582)	;AB = mem[0582]	;20 40 05 82
0577 AND imm(00)	;AB = AB AND 00h	;34 E0 00
057A BNEQL imm(F7)	;skok na adresu 0573h	;11 F7
057C		
057D		
057F dw 00 00	;16 bitna adresa bloka	;00 00
0581 db 00	;8 bitna dužina bloka	;00
0582 db 00	;8 bitni semafor	;00
0583 db 08	;8 bitni podatak	;08
0584 db 09	;8 bitni podatak	;09
0585 db 0A	;8 bitni podatak	;0A
0586 db 0B	;8 bitni podatak	;0B
0587 db 0C	;8 bitni podatak	;0C

;Prekidna Rutina

0590 LDB memind(057F)	;AB = mem[[057F]]	;20 60 05 7F
0594 STB memdir(F002)	;mem[F002] = AB;registar podataka	;22 40 F0 02
0598 LDB memdir(0580)	;AB = mem[0580];adresa bloka high	;20 40 05 80
059C INC	;AB = AB + 1	;32
059D STB memdir(0580)	;mem[0580] = AB	;22 40 05 80
05A1 LDB memdir(0581)	;AB = mem[0581];dužina bloka	;20 40 05 81
05A5 DEC	;AB = AB - 1	;33
05A6 STB memdir(0581)	;mem[0581] = AB	;22 40 05 81
05AA BNEQL imm(12)	;skok na adresu 05BE	;11 12

05AC LDB imm(00)	;AB = 00h	;20 E0 00
05AF STB memdir(0582)	;mem[0582] = AB;semafor	;22 40 05 82
05B3 LDB imm(02)	;AB = 02h;stop reč	;20 E0 02
05B6 STB memdir(F000)	;mem[F000] = AB	;22 40 F0 00
05BA RTI		;0D

Pitanja:

1. U trenutku T=32 koje su vrednosti na memorijskim lokacijama 0583h, 0584h, 0585h, 0585h i 0587h?
2. U trenutku T=189 šta se upisuje u memorijsku lokaciju sa adresom 057Fh? Šta predstavlja ova vrednost?
3. U trenutku T=279 šta se upisuje u memorijsku lokaciju sa adresom 0580h? Šta predstavlja ova vrednost?
4. U trenutku T=369 šta se upisuje u memorijsku lokaciju sa adresom 0581h? Šta predstavlja ova vrednost?
5. U trenutku T=459 šta se upisuje u memorijsku lokaciju sa adresom 0582? Šta predstavlja ova vrednost?
6. U trenutku T=549 šta se upisuje na adresu F000? Šta predstavlja ova adresa, a upisana vrednost? Da li je neka od linija prekida procesora aktivna i zašto?
7. Zašto se u trenutku T=598 skače sa izvršavanja instrukcije na adresi 0573h na onu na adresi 0590h?
8. U trenutku T=712 šta se upisuje u memorijsku lokaciju sa adresom F002? Šta predstavlja ova adresa, a šta upisana vrednost?
9. U trenutku T=829 šta se nalazi na memorijskoj lokaciji 0580h? Kako se promenila ova vrednost u odnosu na trenutak T=279?
10. U trenutku T=950 šta se nalazi na memorijskoj lokaciji 0581h? Kako se promenila ova vrednost u odnosu na trenutak T=369?
11. U trenutku T=982 da li je uslov skoka ispunjen i zašto?
12. U trenutku T=2840 da li je uslov skoka ispunjen i zašto?
13. U trenutku T=3076 šta se upisuje na adresu F000? Šta predstavlja ova vrednost?
14. U trenutku T=3076 šta se nalazi na prvih pet lokacija u memoriji periferije? Odakle ove vrednosti potiču?

3)Ulaz sa DMA kontrolerom

Cilj primera je da se ilustruje ulaz niza bajtova sa periferije u memoriju korišćenjem kontrolera periferije sa DMA sa mehanizmom prekida.

U programu se koristi uređaj broj 2 (0-64). Za startovanje periferije u odgovarajućem režimu i njeno zaustavljanje koriste se kontrolne reči: Start reč 0Fh, Stop reč 0Bh. Adrese registara kontrolera korišćene periferije su: Adrese Kontrolnog registra F080, Brojač low F084, Brojač high F085, Adresnog registra high F086, Adresnog registra low F087. Primer se sastoji od glavnog programa i prekidne rutine. U glavnom programu postavljaju se memorijske lokacije sa adresom niza podataka, dužinom i semafora promenljiva na 01h. Periferija se startuje i proverava se semafor. U prekidnoj rutini semafor se postavlja na nulu i periferija se zaustavlja. Kako se koristi mehanizam prekida treba obezbediti da se pri prekidu skoči na pravu prekidnu rutinu. Pre početka izvršavanja programa treba postaiti ulaz 6 u tabeli adresa prekidnih rutina, koji ova periferija fiksno koristi, na adresu prekidne rutine 0600h koja treba pri prekidu da se izvrši.

U memorijskim lokacijama 05C0h do 05ECh i 0600h do 060Eh nalaze se instrukcije, u memorijskoj lokaciji 05EDh se nalazi 8 bitna semafora promenljiva a u memorijskim lokacijama 05EEh do 05F2h se smešta blok podataka koji se prima iz periferije.

;Glavni Program

05C0 LDW imm(000B)	;AW = 000Bh;demaskiranje prekida	;21 E0 00 0B
05C4 STMR	;IMR = AW	;2B
05C5 INTE	;PSWI = 1	;05
05C6 LDW imm(05EE)	;AW = 05EEh	;21 E0 05 EE
05CA STW memdir(F086)	;mem[F086] = AW;adresa bloka	;23 40 F0 86
05CE LDW imm(0005)	;AW = 0005h	;21 E0 00 05
05D2 STW memdir(F084)	;mem[F084] = AW;dužina bloka	;23 40 F0 84
05D6 LDB imm(01)	;AB = 01h	;20 E0 01
05D9 STB memdir(05ED)	;mem[05ED] = AB;semafor	;22 40 05 ED
05DD LDB imm(0F)	;AB = 0Fh	;20 E0 0F
05E0 STB memdir(F080)	;mem[F080] = AB;start reč	;22 40 F0 80
05E4 LDB memdir(05ED)	;AB = mem[05ED]	;20 40 05 ED
05E8 AND imm(00)	;AB = AB AND 00h	;43 E0 00
05EB BNQL imm(F7)	;skok na adresu 05E4	;11 F7
05ED db 00	;8 bitni semafor	;00
05EE db 01	;8 bitni podatak	;01
05EF db 02	;8 bitni podatak	;02
05F0 db 03	;8 bitni podatak	;03
05F1 db 04	;8 bitni podatak	;04
05F2 db 05	;8 bitni podatak	;05

;Prekidna Rutina

0600 LDB imm(00)	;AB = 00h	;20 E0 00
0603 STB memdir(05ED)	;mem[05ED] = AB;semafor	;22 40 05 ED
0607 LDB imm(0B)	;AB = 0Bh;stop reč	;20 E0 0B
060A STB memdir(F080)	;mem[F080] = AB;CR	;22 40 F0 80
060E RTI		;0D

Pitanja:

1. U trenutku T=32 šta se nalazi u memorijskim lokacijama sa adresama 05EEh, 05EFh, 05F0h, 05F1h, 05F2h?
2. U trenutku T=196 šta se upisuje na adresi F086h? Šta predstavlja ova adresa, a šta vrednost koja se u nju upisuje?
3. U trenutku T=292 šta se upisuje na adresi F084h? Šta predstavlja ova adresa, a šta vrednost koja se u nju upisuje?
4. U trenutku T=381 šta se upisuje u memorijsku lokaciju 05EDh? Šta predstavlja ova lokacija, a šta znači vrednost koja se u nju upisuje?
5. U trenutku T=471 šta se upisuje na adresu F080h? Šta predstavlja ova adresa, a šta vrednost koja se u nju upisuje?
6. Šta se događa na magistarli u trenucima T=483, T=491, T=507 i T=516? Koji ciklus je u toku i ko ga je inicirao?
7. U trenutku T=521 da li je neka linija za prekide procesora aktivna i zašto?

8. U trenutku T=609 se skače na adresu 600h. Šta predstavlja ova adresa i zašto se tamo nastavlja izvršavanje instrukcija?
9. U trenutku T=692 šta se upisuje u memorijsku lokaciju sa adresom 05EDh? Zašto se tamo upisuje vrednost?
10. U trenutku T=782 šta se upisuje na adresu F080? Šta predstavlja ta vrednost?
11. U trenutku T=888 da li je ispunjen uslov skoka i zašto?
12. U trenutku T=888 šta se nalazi na memorijskim lokacijama sa adresama 05EEh, 05EFh, 05F0h, 05F1h, 05F2h? Zašto su se vrednosti promenile od trenutka T=32?

3. Programi za testiranje mehanizma prekida

Sledećih devet primera ilustruju rad mehanizama prekida procesora računarskog sistema.

1) Opsluživanje prekida i povratak iz prekidne rutine

Cilj ovog primera je da se pokaže šta se dešava u procesoru od trenutka kada stigne zahtev za prekid do trenutka kada se u registru PC nađe adresa prve instrukcije prekidne rutine, kao i od čega se sastoji izvršavanje instrukcije RTI.

U programu se prvo puni tabela prekidnih rutina adresom prekidne rutine periferije koja će da izazove prekid 0605h u njoj odgovarajući ulaz tabele 12 (adresa CC18). Nakon toga se demaksiraju svi maskirajući prekidi upisom svih jedinica u registar maske i dozvolom svih maskirajućih prekida postavljanjem jedinice u bit I statusnog registra. Nakon toga se puni registar WCR periferije da se prekid izazove nakon 75 taktova od startovanja periferije i periferija se startuje. Dalje se glavni program vrti u petlji oduzimajuću od vrednosti 05h vrednost memorijske lokacije 0645h koja je 06h sve dok ne dođe prekid od periferije i vrednost te lokacije se u prekidnoj rutini ne postavi na 05h. Po povratku iz prekidne rutine program izlazi iz petlje.

U memorijskim lokacijama 0620h do 0644h i 0650h do 0657h nalaze se instrukcije, a u memorijskoj lokaciji 0645h nalazi se 8 bitni operand 06h.

;Glavni program

0620 LDW imm(0605)	;AW = 0605h	;21 E0 06 50
0624 STW memdir(CC18)	;mem[CC18] = 06h, mem[CC19] = 05h	
		;23 40 CC 18
0628 LDW imm(0010)	;AW = 0010h	;21 E0 00 10
062C STIMR	;IMR = AW	;2B
062D INTE	;PSWI = 1	;05
062E LDB imm(75)	;AB = 75h	;20 E0 75
0631 STB memdir(F143)	;mem[F143] = AB	;22 40 F1 43
0635 LDB imm(01)	;AB = 01h	;20 E0 01
0638 STB memdir(F140)	;mem[F140] = AB	;22 40 F1 40
063C LDB imm(05)	;AB = 05h	;20 E0 05
063F SUB memdir(0645)	;AB = AB – mem[0645]	;31 40 06 45
0643 BNEQ imm(F7)	;skok na adresu 063C	;11 F7
0645 db 06	;8 bitni operand	;06

;Prekidna rutina

0650 LDB imm(05)	;AB = 05h	;20 E0 05
0653 STB memdir(0645)	;mem[0645] = AB	;22 40 06 45

Pitanja:

1. U trenutku T=32 koja se vrednost nalazi na adresi 0645h?
2. Šta predstavlja adresa CC18h?
3. Šta predstavlja adresa 0650h? Zašto se ova vrednost upisuje na adrese CC18h i CC19h?
4. Šta predstavlja adresa F143? Šta predstavlja vrednost 50h koja se upisuje na tu adresu?
5. Šta predstavlja adresa F140? Zašto se upisuje vrednost 01h na tu adresu?
6. U trenutku T=496 da li je ispunjen uslov skoka? Zašto?
7. U trenutku T=498 koja je vrednost signala print? Šta to znači?
8. U trenutku T=499 koja je vrednost signala incSP? Zašto se inkrementira registar SP?
9. U trenutku T=500 koje su vrednosti signala mxMAR2, mxMAR0, ldMAR, mxMDR2 i ldMDR? Koja se vrednost upisuje u registra MAR? A koja u registar MDR?
10. U trenutku T=502 koji je ciklus na magistrali u toku? Zašto?
11. U trenutku T=507 koja je vrednost signala incSP? Zašto se inkrementira registra SP?
12. U trenutku T=508 koje su vrednosti signala mxMAR2, mxMAR0, ldMAR, mxMDR0, mxMDR2 i ldMDR? Koja se vrednost upisuje u registra MAR? A koja u registar MDR?
13. U trenutku T=510 koji je ciklus u toku na magistrali? Zašto?
14. U trenutku T=515 koja je vrednost signala incSP? Zašto se inkrementira registar SP?
15. U trenutku T=516 koje su vrednosti signala mxMAR2, mxMAR0, ldMAR, mxMDR2, mxMDR0 i ldMDR? Koja se vrednost upisuje u registar MAR? A koja u registar MDR?
16. U trenutku T=518 koji je ciklus na magistrali u toku? Zašto?
17. U trenutku T=536 koje su vrednosti signala mxBR0, ldBR, clINTR i ldL? Šta se postiže ovim signalima?
18. U trenutku T=537 koje su vrednosti signala mxMAR i ldMAR? Koja se vrednost upisuje u registra MAR?
19. U trenutku T=539 koji je ciklus na magistrali u toku? Zašto?
20. U trenutku T=552 koja je vrednost signala ldPC? Koja se vrednost upisuje u registar PC? Odakle se nastavlja izvršavanje instrukcija?
21. U trenutku T=641 koja je vrednost na adresi 0645h?
22. U trenutku T=655 koje su vrednosti signala mxMAR2, mxMAR0, ldMAR i decSP? Koja se vrednost upisuje u registar MAR? Zašto se dekrementira registar SP?
23. U trenutku T=657 koji ciklus na magistrali je u toku? Zašto?
24. U trenutku T=662 koja je vrednost signala ldPSWL? Šta se njime postiže?
25. U trenutku T=687 koja je vrednost signala ldPC? Koja se vrednost upisuje u PC? Gde se nastavlja izvršavanje instrukcija?
26. U trenutku T=803 da li je ispunjen uslov skoka? Zašto?

2) Maskiranje svih maskirajućih prekida

Cilj ovog primera je da pokaže kako se instrukcijama INTE i INTD programskim putem odlučuje u kojim delovima programa će se reagovati na maskirajuće prekide, a

u kojima ne. Instrukcijom INTE se upisuje 1 u bit I statusnog registra PSW čime se demaskiraju svi maskirajući prekidi, a instrukcijom INTD 0, čime se svi maskiraju.

U programu se prvo u tabelu prekidnih rutina upisuje adresa prekidne rutine periferije koja će da izazove prekid, 0690h, u njen ulaz broj 12 (adresa CC18). Onda se demaskiraju svi maskirajući prekidi upisivanjem svih jedinica u registar maske, ali se svi maskiraju upisivanjem 0 u bit I statusnog registra. U periferiju se u registra WCR upisuje vrednost 05h da bi se prekid dogodio nakon 5 taktova od startovanja periferije i periferija se startuje. Program nastavlja izvršavanje instrukcije LDW kada prekid stiže ali se ne skače u prekidnu rutinu, jer su prekidi maskirani. Prekidi se demaskiraju postavljanjem 1 u bit I statusnog registra i tek tada se skače u prekidnu rutinu periferije, jer tek tada postoji mogućnost da se prekid prihvati.

U memorijskim lokacijama 0660h do 0684h i 0690h nalaze se instrukcije programa.

;Glavni program

0660 LDW imm(0690)	;AW = 0690h	;21 E0 06 90
0664 STW memdir(CC18)	;mem[CC18] = 06h, mem[CC19] = 90h	
		;23 40 CC 0A
0668 LDW imm(0010)	;AW = 0010	;21 E0 00 10
066C STIMR	;IMR = AW	;2B
066D INTD	;PSWI = 0	;04
066E LDB imm(05)	;AB = 05h	;20 E0 05
0671 STB memdir(F143)	;mem[F143] = AB	;22 40 F1 43
0675 LDB imm(01)	;AB = 01h	;20 E0 01
0678 STB memdir(F140)	;mem[F140] = AB	;22 40 F1 40
067C LDW imm(0025)	;AW = 0025h	;21 E0 00 25
0680 INTE	;PSWI = 1	;05
0681 LDW imm(0025)	;AW = 0025h	;21 E0 00 25

;Prekidna rutina

0690 RTI		;0D
----------	--	-----

Pitanja:

1. Šta predstavlja adresa CC18h?
2. Šta predstavlja adresa 0690h? Zašto se ova vrednost upisuje na prethodnu adresu?
3. U trenutku T=208 koja je vrednost signala clPSWI? Šta se njime postiže?
4. Šta predstavlja adresa F143h? A šta predstavlja vrednost 05h koja se u nju upisuje?
5. Šta predstavlja adresa F140h? Zašto se na ovu adresu upisuje vrednost 01h?
6. U trenutku T=386 koja je vrednost signala printr? A koja je vrednost signala printr4? Zašto se razlikuju?
7. U trenutku T=439 koja je vrednost signala stPSWI? Šta se njime postiže?
8. U trenutku T=440 koja je vrednost signala printr? Da li je jednaka vrednosti print4?
9. U trenutku T=494 koja je vrednost signala ldPC? Koja se vrednost upisuje u registar PC? Na kojoj adresi se nastavlja izvršavanje instrukcija?

3)Selektivno maskiranje maskirajućih prekida

Cilj ovog primera je da se pokaže da kada u procesor stignu zahtevi za prekid od više periferija istovremeno, skače se u prekidnu rutinu one periferije najvišeg

prioriteta od koje je stigao zahtev za prekid koji nije selektivno maskiran odgovarajućim bitom 0 u registru maske.

U programu se prvo u tabelu prekidnih rutina u odgovarajuće ulaze upišu adrese prekidnih rutina periferija. Onda se selektivno maskiraju maskirajući prekidi upisivanjem odgovarajuće vrednosti u registar maske tako da je demaskiran prekid samo od periferije 2. Dalje se demaskiraju svi prekidi upisivanjem 1 u bit I statusnog registra. Onda se u registre WCR periferija upišu odgovarajuće vrednosti tako da sve izazovu prekid u istom taktu i periferije se startuju. Sve periferije generišu prekid u isto vreme, ali se skače u prekidnu rutinu periferije 2, jer je ona jedina demaskirana u registru maske.

U memorijskim lokacijama 0700h do 0745h i 0750h do 0752h nalaze se instrukcije programa.

;Glavni program

0700 LDW imm(0750)	;AW = 0750h	;21 E0 07 50
0704 STW memdir(CC18)	;mem[CC18] = 07, mem[CC19] = 50h	
		;23 40 CC 18
0708 LDW imm(0751)	;AW = 0751h	;21 E0 07 51
070C STW memdir(CC1A)	;mem[CC1A] = 07, mem[CC1B] = 51h	
		;23 40 CC 1A
0710 LDW imm(0752)	;AW = 0752	;21 E0 07 52
0714 STW memdir(CC1C)	;mem[CC1C] = 07, mem[CC1D] = 52h	
		;23 40 CC 1C
0718 LDW imm(0010)	;AW = 0010h	;21 E0 00 10
071C STIMR	;IMR = AW	;2B
071D INTE	;PSWI = 1	;05
071E LDB imm(35)	;AB = 35h	;20 E0 35
0721 STB memdir(F143)	;mem[F143] = AB	;22 40 F1 43
0725 LDB imm(15)	;AB = 15h	;20 E0 15
0728 STB memdir(F183)	;mem[F183] = AB	;22 40 F1 83
072C LDB imm(05)	;AB = 05h	;20 E0 05
072F STB memdir(F1C3)	;mem[F1C3] = AB	;22 40 F1 C3
0733 LDB imm(01)	;AB = 01h	;20 E0 01
0736 STB memdir(F140)	;mem[F140] = AB	;22 40 F1 40
073A STB memdir(F180)	;mem[F180] = AB	;22 40 F1 80
073E STB memdir(F1C0)	;mem[F1C0] = AB	;22 40 F1 C0
0742 LDW imm(0025)	;AW = 0025h	;21 E0 00 25

;Prekidne rutine

0750 RTI	;0D
0751 RTI	;0D
0752 RTI	;0D

Pitanja:

1. Šta predstavljaju adrese CC18h, CC1Ah i CC1Ch ?
2. Šta predstavljaju adrese 0750h, 0751h i 0752h ? Zašto se ove vrednosti upisuju na prethodne adrese?
3. Šta predstavljaju adrese F143h, F183h i F1C3h? A šta vrednosti 35h, 15h i 05h koje se na njih upisuju?

4. Šta predstavljaju adrese F140h, F180h i F1C0h? Zašto se na njih upisuje vrednost 01h?
5. U trenutku T=862 koja je vrednost signala printr? A koje su vrednosti signala printr4, printr5 i printr6?
6. U trenutku T=911 koja je vrednost signala ldPC? Koja se vrednost upisuje u registar PC? Zašto?

4) Prioriteti maskirajućih prekida

Cilj ovog primera je da se pokaže da kada u procesor stignu zahtevi za prekid od više periferija i svi su demaskirani skače se u prekidnu rutinu one periferije koja je najvišeg prioriteta.

U programu se prvo u tabelu prekidnih rutina u odgovarajuće ulaze upišu adrese prekidnih rutina periferija. Onda se selektivno demaskiraju svi maskirajući prekidi upisivanjem svih jedinica u registar maske. Dalje se demaskiraju svi prekidi upisivanjem 1 u bit I statusnog registra. Onda se u registre WCR periferija upišu odgovarajuće vrednosti tako da sve izazovu prekid u istom taktu i periferije se startuju. Sve periferije generišu prekid u isto vreme, ali se skače u prekidnu rutinu periferije 1, jer je ona najvišeg prioriteta.

U memorijskim lokacijama 06A0h do 06E5h i 06F0h do 06F2h nalaze se instrukcije programa.

;Glavni program

06A0 LDW imm(06F0)	;AW = 06F0h	;21 E0 06 F0
06A4 STW memdir(CC18)	;mem[CC18] = 06h, mem[CC19] = F0	
		;23 40 CC 18
06A8 LDW imm(06F1)	;AW = 06F1h	;21 E0 40 06 F1
06AC STW memdir(CC1A)	;mem[CC1A] = 06h, mem[CC1B] = F1	
		;23 40 CC 1A
06B0 LDW imm(06F2)	;AW = 06F2h	;21 E0 06 F2
06B4 STW memdir(CC1C)	;mem[CC1C] = 06h, mem[CC1D] = F2	
		;23 40 CC 1C
06B8 LDW imm(00FF)	;AW = 00FFh	;21 E0 00 FF
06BC STIMR	;IMR = AW	;2B
06BD INTE	;PSWI = 1	;05
06BE LDB imm(35)	;AB = 35h	;20 E0 35
06C1 STB memdir(F143)	;mem[F143] = 05h	;22 40 F1 43
06C5 LDB imm(15)	;AB = 15h	;20 E0 15
06C8 STB memdir(F183)	;mem[F183] = 15h	;22 40 F1 83
06CC LDB imm(05)	;AB = 05h	;20 E0 05
06CF STB memdir(F1C3)	;mem[F1C3] = AB	;22 40 F1 C3
06D3 LDB imm(01)	;AB = 01h	;20 E0 01
06D6 STB memdir(F140)	;mem[F140] = AB	;22 40 F1 40
06DA STB memdir(F180)	;mem[F180] = AB	;22 40 F1 80
06DE STB memdir(F1C0)	;mem[F1C0] = AB	;22 40 F1 C0
06E2 LDW imm(0025)	;AW = 0025h	;21 E0 00 25

;Prekidne rutine

06F0 RTI	;0D
06F1 RTI	;0D

06F2 RTI

;0D

Pitanja:

1. Šta predstavljaju adrese CC18h, CC1Ah i CC1Ch ?
2. Šta predstavljaju adrese 06F0h, 06F1h i 06F2h ? Zašto se ove vrednosti upisuju na prethodne adrese?
3. Šta predstavlja vrednost 00FF koja se upisuje u registar maske instrukcijom STIMR?
4. Šta predstavljaju adrese F143h, F183h i F1C3h? A šta vrednosti 05h, 15h i 25h koje se na njih upisuju?
5. Šta predstavljaju adrese F140h, F180h i F1C0h? Zašto se na njih upisuje vrednost 01h?
6. U trenutku T=862 koja je vrednost signala printr? A koje su vrednosti signala printr4, printr5 i printr6?
7. U trenutku T=911 koja je vrednost signala ldPC? Koja se vrednost upisuje u registra PC? Zašto?

5)Prekid posle svake instrukcije

Cilj ovog primera je da se pokaže rad trap mehanizma. Kada je ovaj mehanizam aktiviran posle izvršavanja svake instrukcije skače se u odgovarajuću prekidnu rutinu. Ovaj mehanizam aktivira se instrukcijom TRPE koja postavlja 1 u bit T statusnog registra, a deaktivira se instrukcijom TRPD koja u isti bit postavlja 0.

U programu se prvo u tabelu prekidnih rutina u ulay nula koji odgovara prekidu trap mehanizma stavlja adres prekidne rutine 0780. Omogućava se trap mehanizam i izvršavaju tri DLW instrukcije. Posle svake od instrukcija skočiće se na trap prekidnu rutinu. Dalje se ovaj mehanizam isključuje i izvršava još jedna LDW instrukcija posle koje se neće skočiti na prekidnu rutinu, jer je mehanizam trap isključen.

U memorijskim lokacijama 0760h do 0780h nalaze se instrukcije programa.

;Glavni program

0760 LDW imm(0780)	;AW = 0780h	;21 E0 07 80
0764 STW memdir(CC00)	;mem[CC00] = 07h, mem[CC01] = 80	;23 40 CC 00
0768 TRPE	;PSWT = 1	;07
0769 LDW imm(0056)	;AW = 0056h	;21 E0 00 56
076D LDW imm(0057)	;AW = 0057h	;21 E0 00 57
0771 LDW imm(0058)	;AW = 0058h	;21 E0 00 58
0775 TRPD	;PSWT = 0	;06
0776 LDW imm(0059)	;AW = 0059h	;21 E0 00 59

;Prekidna rutina

0780 RTI

;0D

Pitanja:

1. Šta predstavlja adresa CC00h?
2. Šta predstavlja adresa 0780h? Zašto se ova vrednost upisuje na prethodnu adresu?
3. U trenutku T=149 koja je vrednost signala stPSWT? Šta se njime postiže?

4. U trenutku T=204 koja je vrednost signala ldPC? Koja vrednost se upisuje u registar PC? Zašto?
5. U trenutku T=250 koja je vrednost signala ldPC? Gde se nastavlja izvršavanje programa? Zašto?
6. U trenutku T=350 koja je vrednost signala ldPC? Koja vrednost se upisuje u registar PC? Zašto?
7. U trenutku T=698 koja je vrednost signala clPSWT? Šta se njime postiže?
8. U trenutku T=702 na kojoj adresi se nastavlja izvršavanje programa?

6) Instrukcija INT

Cilj ovog primera je da se pokaže kako se programskim putem preko instrukcije INT može izazvati prekid. Skače se u onu prekidnu rutinu čiji se broj ulaza nalazi u instrukciji INT.

U programu se prvo u tabelu prekidnih rutina u ulaz 4 opstavlja adresa željene prekidne rutine 07A0h. Dalje se prekid izaziva programskim putem instrukcijom INT sa argumentom 04h, što predstavlja ulaz 4 u tabeli prekidnih rutina. Dalje se skače na željenu prekidnu rutinu.

U memorijskim lokacijama 0790h do 07A0h nalaze se instrukcije programa.

```

;Glavni program
0790 LDW imm(07A0)           ;AW = 07A0h           ;21 E0 07 A0
0794 STW memdir(CC08)        ;mem[CC08] = 07h, mem[CC09] = A0
                                ;23 40 CC 08
0798 INT imm(04)             ;0C 04

;Prekidna rutina
07A0 RTI                     ;0D

```

Pitanja:

1. Šta predstavlja adresa CC08h?
2. Šta predstavlja adresa 07A0h? Zašto se ova vrednost upisuje na prethodnu adresu?
3. U trenutku T=159 koja je vrednost signala stPRINS? Šta se njime postiže?
4. U trenutku T=210 koja je vrednost signala ldPC? Koja se vrednost upisuje u registar PC? Na kojoj adresi se nastavlja izvršavanje instrukcija?

7) Gnježđenje prekida

Cilj ovog primera je da se pokaže šta se događa kada usred izvršavanja jedne prekidne rutine stigne zahtev za prekid druge periferije. Druga periferija može biti višeg ili nižeg prioriteta od one čija se prekidna rutina izvršava.

U programu se prvo u tabeli prekidnih rutina u ulazima 12, 13 i 14 koji odgovaraju periferijama 4, 5 i 6 upisuju odgovarajuće adrese prekidnih rutina. Dalje se svi maskirajući prekidi selektivno demaskiraju upisivanjem svih jedinica u registar maske i globalno demaskiraju upisivanjem jedinice u bit I statusnog registra. Dalje se u register WCR periferija 4, 5 i 6 upisuju vrednosti tako da periferije 4 i 5 izazovu prekid u isto vreme, a periferija 6 kasnije. Dalje se periferije startuju. Za vreme izvršavanja instrukcije LDW stiže zahtev za prekid od periferija 4 i 5. Ulazi se u prekidnu rutinu periferije 5, jer je ona višeg prioriteta. Kada se vrati na glavni program nastavlja se u prekidnu rutinu periferije 4. Međutim pri izvršavanju prekidne rutine periferije 4 stiže zahtev za prekid od periferije 6 i on se prihvata jer je višeg

prioriteta i skače se na njenu prekidnu rutinu. Kada se završi ova prekidna rutina skače se na prekidnu rutinu periferije 4 koja se završava i nastavlja se u glavni program.

U memorijskim lokacijama 07B0h do 07F5h i 0802h do 0809h nalaze se instrukcije programa.

;Glavni program

07B0 LDW imm(0802)	;AW = 0802h	;21 E0 08 02
07B4 STW memdir(CC18)	;mem[CC18] = 08h ,mem[CC19] = 02h	;23 40 CC 18
07B8 LDW imm(0800)	;AW = 0800h	;21 E0 08 00
07BC STW memdir(CC1A)	;mem[CC1A] = 08, mem[CC1B] = 00h	;23 40 CC 1A
07C0 LDW imm(0808)	;AW = 0808h	;21 E0 08 08
07C4 STW memdir(CC1C)	;mem[CC1C] = 08h, mem[CC1D] = 08h	;23 40 CC 1C
07C8 LDW imm(00FF)	;AW = 00FFh	;21 E0 00 FF
07CC STMR	;IMR = AW	;2B
07CD INTE	;PSWI = 1	;05
07CE LDB imm(35)	;AB = 35h	;20 E0 35
07D1 STB memdir(F143)	;mem[F143] = AB	;22 40 F1 43
07D5 LDB imm(05)	;AB = 05h	;20 E0 05
07D8 STB memdir(F183)	;mem[F183] = AB	;22 40 F1 83
07DC LDB imm(99)	;AB = 99h	;20 E0 99
07DF STB memdir(F1C3)	;mem[F1C3] = AB	;22 40 F1 C3
07E3 LDB imm(01)	;AB = 01h	;20 E0 01
07E6 STB memdir(F140)	;mem[F140] = AB	;22 40 F1 40
07EA STB memdir(F180)	;mem[F180] = AB	;22 40 F1 80
07EE STB memdir(F1C0)	;mem[F1C0] = AB	;22 40 F1 C0
07F2 LDW imm(0056)	;AW = 0056h	;21 E0 00 56

;Prekidna rutina per 2

0800 INTE	;05
0801 RTI	;0D

;Prekidna rutina per 1

0802 INTE	;05
0803 LDW imm(0056)	;AW = 0056h ;21 E0 00 56
0807 RTI	;0D

;Prekidna rutina per 3

0808 INTE	;05
0809 RTI	;0D

Pitanja:

1. Šta predstavljaju adrese CC18h, CC1Ah i CC1Ch ?
2. Šta predstavljaju adrese 0802h, 0800h i 0808h? Zašto se ove vrednosti upisuju na prethodne adrese?

3. Šta predstavljaju adrese F143h, F183h i F1C3h? A šta vrednosti 05h, 15h i 05h koje se na njih upisuju?
4. Šta predstavljaju adrese F140h, F180h i F1C0h? Zašto se na njih upisuje vrednost 01h?
5. U trenutku T=852 koja je vrednost signala printr? A koja je vrednost signala printr4 i printr5?
6. U trenutku T=912 na kojoj adresi se nastavlja izvršavanje instrukcija? Zašto?
7. Zašto se na početku svake prekidne rutine nalazi instrukcija INTE ?
8. U trenutku T=1026 na kojoj adresi se nastavlja izvršavanje programa? Zašto?
9. U trenutku T=1094 na kojoj adresi se nastavlja izvršavanje programa? Zašto?
10. U trenutku T=1153 na kojoj adresi se nastavlja izvršavanje programa? Zašto?
11. U trenutku T=1244 na kojoj adresi se nastavlja izvršavanje programa? Zašto?

8)Prekid usled greške u načinu adresiranja

Cilj ovog primera je da se pokaže kako procesor reaguje na unutrašnje prekide i to na grešku u načinu adresiranja.

U programu se u tabelu prekidnih rutina u ulaz 2 koji odgovara prekidu usled greške u načinu adresiranja prvo upisuje adresa odgovarajuće prekidne rutine 0820h. Dalje se instrukcijom STB sa neposrednim odredištem izaziva greška u načinu adresiranja i skače se u prekidnu rutinu čija je adresa upisana u tabeli.

U memorijskim lokacijama 0810h do 081Ah i 0820h nalaze se instrukcije programa.

;Glavni Program

0810 LDW imm(0820)	;AW = 0820h	;21 E0 08 20
0814 STW memdir(CC04)	;mem[CC04] = 08, mem[CC05] = 20h	;23 40 CC 04
0818 STB imm(01)		;22 E0 01

;Prekidna rutina

0820 RTI	;0D
----------	-----

Pitanja:

1. Šta predstavlja adresa CC04h?
2. Šta predstavlja adresa 0820h? Zašto se ova vrednost upisuje na prethodnu adresu?
3. U trenutku T=156 koja je vrednost signala gradr? Šta to govori?
4. U trenutku T=157 koja je vrednost signala stPRADR? Zašto?
5. U trenutku T=210 koja je vrednost signala ldPC? Koja se vrednost upisuje u registar PC? Na kojoj adresi se nastavlja izvršavanje programa?
6. U trenutku T=256 koja se instrukcija izvršava nakon instrukcije RTI? Zašto?

9)Prekid usled greške u kodu operacije

Cilj ovog primera je da se pokaže kako procesor reaguje na unutrašnje prekide i to na grešku u u kodu operacije.

U programu se u tabelu prekidnih rutina u ulaz 3 koji odgovara prekidu usled greške u kodu operacije prvo upisuje adresa odgovarajuće prekidne rutine 0840h. Dalje se nekorektnim kodom operacije izaziva greška u kodu operacije i skače se u prekidnu rutinu čija je adresa upisana u tabeli.

U memorijskim lokacijama 0830h do 083Ah i 0840h nalaze se instrukcije programa.

;Glavni Program

0830 LDW imm(0840)	;AW = 0840h	;21 E0 08 40
0834 STW memdir(CC06)	;mem[CC06] = 08, mem[CC07] = 40h	;23 40 CC 06
0838	;nekorektan kod operacije	;0E E0 01

;Prekidna rutina

0840 RTI	;0D
----------	-----

Pitanja:

1. Šta predstavlja adresa CC06h?
2. Šta predstavlja adresa 0840h? Zašto se ova vrednost upisuje na prethodnu adresu?
3. U trenutku T=156 koja je vrednost signala gropr? Šta to govori?
4. U trenutku T=157 koja je vrednost signala stPRCOD? Zašto?
5. U trenutku T=210 koja je vrednost signala ldPC? Koja se vrednost upisuje u registar PC? Na kojoj adresi se nastavlja izvršavanje programa?
6. U trenutku T=256 koja se instrukcija izvršava nakon instrukcije RTI? Zašto?